



SOURCE CODE

KHGT TIMES V8.0

APLIKASI HISAB ASTRONOMI
DAN KALENDER ISLAM GLOBAL

DENGAN MULTIBAHASA, ANALISIS GRAFIK,
DAN INTEGRASI PETA LANGIT DUNIA

KASMUI

Source Code

KHGT TIMES

V8.0

<https://hisabmu.com/khgttimes/>

KASMUI

OKTOBER 2026

KATA PENGANTAR

Bismillahirrahmanirrahim

Alhamdulillah rabbil ‘alamin. Segala puji bagi Allah Swt. atas limpahan rahmat, karunia, ilmu, dan kesempatan yang diberikan sehingga pengembangan **KHGT Times V8.0** beserta penyusunan dokumentasi *source code*-nya dapat diselesaikan dan disajikan dalam bentuk buku ini. Shalawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad saw., keluarga, para sahabat, serta seluruh umat yang mengikuti tuntunan beliau hingga akhir zaman.

Buku **Source Code KHGT Times V8.0** (<https://hisabmu.com/khgttimes/>) disusun sebagai dokumentasi teknis, ilmiah, dan edukatif atas pengembangan perangkat lunak **KHGT Times**, sebuah aplikasi komputasi astronomi dan ilmu falak yang dirancang untuk mendukung perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal secara global, pengelolaan kalender Islam, serta kajian dan implementasi **Kalender Hijriah Global Tunggal (KHGT)**. Di dalam source code dinyatakan bahwa aplikasi ini sekaligus dimaksudkan sebagai sarana edukasi publik di bidang ilmu falak Islami dan sebagai perangkat pendukung bagi transisi menuju penerapan KHGT di masa depan. Pada versi 8.0, sistem telah berkembang menjadi **41 modul menu terintegrasi**, dilengkapi antarmuka multibahasa Indonesia–Inggris–Arab, Peta Langit Dunia, Laporan Astronomi, dan HIJRI_DB Auto-Builder untuk kebutuhan administrator.

Pengembangan KHGT Times berangkat dari kebutuhan untuk mempertemukan tradisi ilmu falak dengan perkembangan komputasi astronomi modern. Persoalan penentuan awal bulan Hijriah, posisi Matahari dan Bulan, waktu-waktu astronomis, visibilitas hilal, arah kiblat, gerhana, serta berbagai fenomena langit tidak lagi cukup dipahami hanya sebagai persoalan perhitungan manual. Perkembangan data ephemeris numerik, komputasi posisi benda langit, pemetaan digital, visualisasi tiga dimensi, dan pengolahan data skala besar membuka kesempatan untuk membangun suatu lingkungan komputasi falak yang lebih terpadu.

Karena itu, KHGT Times V8.0 tidak dirancang hanya sebagai kalkulator untuk menghasilkan angka akhir. Struktur program dikembangkan sebagai sebuah **platform Falak Digital** yang menggabungkan kalkulasi astronomi, kalender Hijriah, visualisasi, pemetaan, simulasi, analisis grafik, pengolahan data, dan sejumlah fasilitas pendukung dalam satu antarmuka aplikasi desktop. Dokumentasi internal program menyebutkan orientasi penggunaannya mencakup observatorium, lembaga riset akademik, dan pihak-pihak yang membutuhkan analisis terkait persoalan kalender dan astronomi Islam. Source code juga memisahkan perhitungan **geosentris**, yaitu parameter yang ditinjau dari pusat Bumi, dan **toposentris**, yaitu parameter yang ditinjau dari lokasi pengamat di permukaan Bumi.

Salah satu fondasi utama KHGT Times V8.0 adalah penggunaan pustaka astronomi **Skyfield**. Dalam struktur kode, Skyfield digunakan melalui sejumlah komponen seperti `Loader`, `wgs84`, `almanac`, `eclipselib`, dan `Geocentric`. Selain itu, aplikasi mengintegrasikan berbagai pustaka Python untuk mendukung antarmuka grafis, analisis data, visualisasi, interpolasi, pemrosesan gambar, komunikasi jaringan, dan berbagai kebutuhan komputasi lainnya. Source code secara

eksplisit mengimpor Skyfield untuk komputasi astronomi serta Matplotlib untuk visualisasi grafik dan kanvas astronomi.

Pada **V8.0**, arsitektur ephemeris Matahari dan Bulan juga mengalami penyempurnaan penting. Sistem dirancang menggunakan mekanisme **auto-switch ephemeris**, yaitu memilih `de440s.bsp` untuk rentang tahun modern **1850–2149**, sedangkan `de441-new.bsp` digunakan untuk rentang **1–1849** dan **2150–2999**. Pendekatan tersebut dimaksudkan agar periode modern dapat menggunakan kernel yang lebih ringkas, sementara kemampuan perhitungan jangka panjang aplikasi tetap dipertahankan. Batas konfigurasi tersebut didefinisikan langsung di dalam source code melalui parameter file dan rentang tahun ephemeris.

Pengembangan ini penting karena kebutuhan komputasi KHGT bukan hanya menentukan posisi satu benda langit pada satu lokasi, tetapi juga berkaitan dengan pemindaian kondisi astronomis pada cakupan geografis yang luas. Dokumentasi internal source code menjelaskan adanya mekanisme pemindaian banyak kota untuk melacak titik pertama terpenuhinya kriteria KHGT serta pengolahan parameter astronomis geosentris dan toposentris secara terpisah. Dengan demikian, aplikasi diarahkan untuk menjembatani kebutuhan kajian astronomi global dengan kebutuhan pengamatan berbasis lokasi.

Dari sisi teknologi perangkat lunak, KHGT Times V8.0 merupakan hasil integrasi berbagai pustaka Python. **CustomTkinter** dan **Tkinter** digunakan untuk membangun antarmuka desktop yang interaktif; **Skyfield** menjadi salah satu fondasi utama komputasi posisi astronomis; **Matplotlib** mendukung visualisasi grafik; **NumPy** digunakan dalam pengolahan data numerik; **SciPy** mendukung fungsi matematis dan interpolasi; sedangkan **Pillow/PIL** digunakan dalam pengolahan dan pembuatan citra. Pada bagian-bagian tertentu aplikasi juga memanfaatkan pustaka lain sesuai kebutuhan masing-masing modul. Dengan desain tersebut, kode sumber bukan sekadar kumpulan fungsi kalkulasi, tetapi merupakan suatu arsitektur perangkat lunak yang menghubungkan lapisan data, perhitungan, visualisasi, antarmuka, dan keluaran pengguna.

Karakter integratif tersebut semakin terlihat pada struktur menu KHGT Times V8.0. Versi ini secara eksplisit menetapkan jumlah menu menjadi **41 modul**. Pembaruan V8.0 menempatkan **Peta Langit Dunia** sebagai **Menu 39**, **Laporan Astronomi** sebagai **Menu 40**, dan **HIJRI_DB Auto-BUILDER (Admin)** sebagai **Menu 41**. Peta Langit Dunia terhubung dengan layanan daring KHGT Times, sedangkan Laporan Astronomi menyediakan akses ke aplikasi pendamping. Penyesuaian tersebut sekaligus menunjukkan bahwa pengembangan KHGT Times bergerak dari aplikasi kalkulasi tunggal menuju ekosistem perangkat astronomi yang lebih luas.

Pada sisi dokumentasi dan aksesibilitas, KHGT Times V8.0 juga mengembangkan antarmuka **multibahasa Indonesia, Inggris, dan Arab**. Sistem penerjemahan tidak hanya diarahkan pada elemen navigasi, tetapi juga pada bagian dokumentasi dan informasi rilis aplikasi. Dengan pendekatan ini, aplikasi diharapkan lebih mudah digunakan oleh pengguna dari latar belakang bahasa yang berbeda serta memiliki potensi penggunaan yang lebih luas dalam pembelajaran dan kajian astronomi Islam internasional. Dokumentasi V8.0 secara khusus menyatakan adanya penyelarasan struktur 41 menu dengan manual, informasi rilis, dan panduan navigasi aplikasi.

Pengembangan KHGT Times juga tidak berhenti pada perhitungan posisi Matahari dan Bulan. Jejak pengembangan pada source code menunjukkan adanya berbagai fitur yang berkaitan dengan visualisasi dan analisis astronomi, antara lain **Altitude Chart Analyzer**, visualisasi dan pelacakan benda langit, sinkronisasi dengan Stellarium, visualisasi rasi bintang dan planet pada Live Animasi, Generator Analemma Matahari, analisis gerhana, perhitungan arah dan waktu kiblat, astrofotografi, prediktor pasang surut, evaluasi SQM fajar, serta berbagai fungsi kalender dan pengolahan data. Keseluruhan fungsi tersebut menunjukkan upaya membangun aplikasi yang dapat digunakan tidak hanya untuk mendapatkan hasil numerik, tetapi juga untuk memahami fenomena astronomi melalui grafik, peta, simulasi, dan representasi visual.

Aspek lain yang menjadi ciri pengembangan KHGT Times adalah integrasi **WinAI** sebagai asisten berbasis kecerdasan buatan. Dalam source code, WinAI dikembangkan agar dapat mendeteksi maksud atau *intent* pengguna serta menghubungkannya dengan pertanyaan maupun perintah perhitungan yang berkaitan dengan menu-menu sistem. Dengan pendekatan tersebut, interaksi manusia dengan aplikasi tidak lagi sepenuhnya bergantung pada navigasi menu konvensional, tetapi mulai diarahkan menuju interaksi yang lebih natural dan kontekstual. Fitur ini tetap harus dipahami sebagai fasilitas pendukung; hasil kalkulasi astronomi perlu ditelusuri kembali kepada parameter, metode komputasi, dan data ephemeris yang digunakan oleh mesin perhitungan.

Keberadaan **HIJRI_DB** juga mempunyai posisi penting dalam arsitektur program. Database ini digunakan untuk mendukung fungsi kalender Hijriah dan konversi yang terdapat dalam aplikasi. Source code memuat struktur data bulan Hijriah beserta tanggal awal dan jumlah hari yang menjadi bagian dari sistem kalender. Di sisi lain, pengembangan **HIJRI_DB Auto-Builder** menunjukkan adanya upaya untuk menyediakan perangkat administratif yang dapat membantu proses pembentukan dan pemutakhiran struktur database kalender secara lebih sistematis. Contoh struktur data kalender tersebut terlihat langsung di dalam source code.

Dalam konteks akademik, buku source code ini diharapkan mempunyai beberapa fungsi sekaligus. Pertama, sebagai **dokumentasi pengembangan perangkat lunak**, sehingga struktur dan logika program dapat ditelusuri. Kedua, sebagai **bahan pembelajaran pemrograman astronomi**, khususnya penerapan Python, Skyfield, data ephemeris JPL, visualisasi, dan GUI dalam kajian ilmu falak. Ketiga, sebagai **bahan evaluasi ilmiah**, karena algoritma dan parameter yang digunakan dapat dibaca, diperiksa, diuji ulang, dan dibandingkan dengan sumber atau perangkat astronomi lainnya. Keempat, sebagai dokumentasi perkembangan KHGT Times dari satu versi ke versi berikutnya sehingga perubahan algoritma, fitur, struktur menu, maupun basis ephemeris dapat dilacak secara lebih transparan.

Keterbukaan source code memiliki nilai penting dalam perangkat lunak ilmiah. Sebuah hasil komputasi astronomi seharusnya tidak dipahami hanya berdasarkan angka yang muncul pada layar. Pengguna perlu mengetahui dari mana data berasal, bagaimana koordinat didefinisikan, bagaimana waktu dikonversi, kernel ephemeris apa yang digunakan, bagaimana posisi pengamat ditentukan, serta bagaimana suatu parameter astronomis dihitung dan divisualisasikan. Karena itulah buku ini diharapkan dapat menjadi jembatan antara **hasil komputasi** dan **mekanisme komputasi** yang melatarbelakanginya.

Meskipun KHGT Times V8.0 telah mengintegrasikan beragam fungsi, perangkat lunak ilmiah pada hakikatnya merupakan produk yang terus berkembang. Pembaruan pustaka Python, perubahan lingkungan sistem operasi, pembaruan database, perkembangan metode astronomi, koreksi terhadap algoritma, serta kebutuhan pengguna dapat menuntut penyempurnaan lebih lanjut. Oleh karena itu, source code yang disajikan hendaknya dipandang sebagai dokumentasi teknis **versi 8.0**, bukan sebagai titik akhir pengembangan. Validasi silang terhadap hasil observasi, ephemeris, perangkat astronomi independen, serta literatur ilmiah tetap diperlukan ketika hasil aplikasi digunakan untuk penelitian atau pengambilan keputusan.

Penulis berharap buku **Source Code KHGT Times V8.0** dapat memberikan manfaat bagi mahasiswa, dosen, peneliti, programmer, praktisi ilmu falak, pengelola observatorium, pemerhati kalender Islam, serta masyarakat yang ingin memahami bagaimana astronomi modern dapat diimplementasikan dalam perangkat lunak komputasi. Bagi pembaca yang berlatar belakang ilmu falak, buku ini diharapkan dapat memperkenalkan struktur pemrograman astronomi secara lebih konkret. Bagi pembaca yang berlatar belakang pemrograman, buku ini diharapkan dapat memberikan contoh bagaimana Python, data ephemeris, algoritma astronomi, antarmuka grafis, visualisasi, dan basis data dapat dipadukan untuk menyelesaikan persoalan-persoalan astronomi Islam.

Lebih jauh, pengembangan KHGT Times diharapkan dapat ikut mendorong tumbuhnya budaya **falak digital yang terbuka, terukur, dapat diuji, dan dapat dikembangkan kembali**. Transformasi ilmu falak ke lingkungan digital bukan berarti meninggalkan tradisi keilmuan yang telah berkembang selama berabad-abad, melainkan memperkaya metode kajian dengan kemampuan komputasi modern. Observasi tetap diperlukan, teori tetap menjadi dasar, metodologi tetap harus dapat dipertanggungjawabkan, sedangkan teknologi menjadi alat untuk memperluas kemampuan manusia dalam menghitung, menganalisis, memvisualisasikan, dan memahami fenomena langit.

Penyusunan buku ini tentu masih memungkinkan adanya kekurangan, baik pada dokumentasi kode, penjelasan teknis, konsistensi istilah, maupun aspek lain yang terus berkembang bersama aplikasi. Oleh sebab itu, kritik, koreksi, validasi, dan masukan ilmiah dari para ahli astronomi, ahli falak, akademisi, programmer, serta pengguna KHGT Times sangat berharga untuk penyempurnaan versi berikutnya.

Akhirnya, semoga pengembangan **KHGT Times V8.0** dan penyusunan buku source code ini dapat menjadi bagian kecil dari ikhtiar pengembangan ilmu pengetahuan, khususnya pada persimpangan antara **astronomi, ilmu falak, kalender Hijriah, teknologi informasi, dan komputasi ilmiah**. Semoga karya ini memberikan manfaat nyata bagi pendidikan, penelitian, pengembangan perangkat lunak astronomi, dan penguatan literasi ilmu falak pada era digital.

Kepada Allah Swt. segala usaha dan hasil akhirnya dikembalikan. Semoga setiap ilmu yang bermanfaat menjadi amal yang terus memberikan kebaikan.

Semarang, 2026

KASMUI

DAFTAR ISI

KATA PENGANTAR	4
DAFTAR ISI	7
SOURCE CODE KHGT TIMES V8.0	13
BAGIAN I — IMPORT, KONFIGURASI, DAN DEPENDENSI SISTEM	
1. Import Library Dasar Python dan Dependensi Utama	13
1.1. Tambahan Import untuk WinAI	13
1.2. Konfigurasi URL Server WinAI	13
1.3. Konfigurasi Ephemeris V8.0 — Fokus Matahari & Bulan	13
1.4. Konfigurasi de440s.bsp dan de441-new.bsp	14
1.5. Terjemahan Highlight Rilis V8.0	14
1.6. GUI Libraries	19
1.7. Skyfield Libraries	19
1.8. Matplotlib Libraries	19
1.9. Dependensi Peta HD dan Interpolasi SciPy	19
1.10. Dependensi Modul Alarm dan Notifikasi	19
1.11. Perbaikan Crash GUI dan SSL untuk EXE	20
BAGIAN II — UTILITAS, DATABASE, DAN ENGINE DASAR	
2. Fungsi Tahun Aman dan Kalender BCE/CE	20
2.1. Fungsi Utilitas Gambar Jadwal Shalat	20
2.2. Parser Data Jadwal Shalat	21
2.3. Generator Gambar Jadwal Shalat	21
2.4. Database Hijriah & Kalender	24
2.5. Struktur BULAN_MASEHI, BULAN_HIJRIAH, dan HIJRI_DB	24
2.6. Auto-Load Database Hijriah dari JSON	35
2.7. Database Kota	36
2.8. Penambahan Referensi Gisborne, New Zealand	40
2.9. Fungsi Unduh Ephemeris & Map	45
2.10. Fungsi-Fungsi Formatting & Safety	45
2.11. Database Bintang Terang — Koordinat J2000	47
2.12. Penambahan Metrik Astronomi	49
2.13. Panel Matahari	49
2.14. Panel Bulan	49
2.15. Perbaikan Iluminasi Matahari-Bulan untuk Stellarium	51
2.16. Kalkulasi Ekstra: Elongasi, Iluminasi, dan Magnitudo	52
BAGIAN III — KELAS UTAMA DAN ARSITEKTUR GUI	
3. Kelas Utama Aplikasi GUI	54
3.1. Inisialisasi Sistem Bahasa	54
3.2. Konfigurasi Ikon Aplikasi	55
3.3. Inisialisasi Sistem Alarm	55
3.4. Variabel Simulasi 3D Matplotlib	56
3.5. Variabel Kurva Alaska	56
4. Sidebar dan Sistem Navigasi	60
4.1. UI Pemilih Bahasa pada KHGT Engine	60
4.2. Modern Scrollable Menu Selection	60
4.3. Kelompok Kalender dan Konversi	60
4.4. Kelompok Analisis dan Komparasi 50 Tahun	61
4.5. Ephemeris dan Posisi Benda Langit	61
4.6. Waktu Shalat dan Arah Kiblat	61
4.7. Password dan Otorisasi HIJRI_DB Auto-Builder	62
5. Container Input dan Pengaturan Modul	63
5.1. Container Database Kota	63
5.2. Container Alarm Salat	64
5.3. Container Visibility Inputs	64
5.4. Pilihan Waktu Observasi	65
5.5. Container Ephemeris 3D	66
5.6. Input Mizwala/Tongkat Istiwa	67

5.7. Auto-Builder Database Hijriah	68
5.8. Status Kriteria Batas.....	68
5.9. Frame Input Modul 31–34	69
5.10. Astrofotografi dan Prediktor Okultasi	69
5.11. Pasang Surut Gravitasi	70
5.12. Evaluasi Fajar Shadiq & SQM	70
5.13. Analemma Matahari.....	71
6. Main Area — Ruang Output Kanan	71
6.1. Widget Cuaca.....	72
6.2. Tombol dan Sistem Ekspor CSV	73
6.3. Frame Output Grafik.....	73
6.4. Input Kalender Hijriah	74
6.5. Input Kalender Masehi.....	74
6.6. Setup WinAI	75
BAGIAN IV — INPUT MODUL ASTRONOMI DAN FALAK	
7. Moonphase.....	76
8. Ephemeris.....	76
8.1. Waktu Awal	76
8.2. Waktu Akhir	77
8.3. Lokasi dan Interval.....	78
9. Qiblah Direction.....	78
9.1. Celestial Qiblah.....	79
10. Moon Times	80
11. Sun Times.....	80
12. Prayer Times.....	81
12.1. In-Flight Prayer Times	83
13. Crescent Visibility Map	83
14. Pilihan Kriteria Visibilitas	84
15. Konversi Kalender.....	85
16. Qibla Time / Rashdul Qiblah Lokal	86
17. Analisis Gerhana	87
18. Live Animasi 2D	87
19. Sistem 3D berbasis Skyfield.....	89
20. GHA dan Koordinat Astronomis	89
21. Parameter Kurva Alaska	90
22. Altitude Chart.....	91
22.1. Grafik Harian 24 Jam	91
22.2. Grafik Tren Bulanan Hijriah	92
22.3. Grafik Kombinasi.....	92
23. First Point / Kota Pertama KHGT	92
24. Equinox & Solstice.....	93
25. Planetary Times.....	94
BAGIAN V — LOGIKA GUI, ROUTING, DAN INFORMASI RILIS	
26. Perhitungan Posisi Toposentris Skyfield.....	98
27. Perhitungan Posisi Geosentris Skyfield.....	99
28. Plotting 3D Matplotlib.....	99
29. Sinkronisasi Timezone	100
30. Logika Pergantian Menu	103
31. Routing Menu dan Output Modul	106
32. Integrasi WinAI pada Pergantian Menu	109
33. Logika Alarm dan Notifikasi Salat	112
33.1. Sinkronisasi Zona Waktu Independen.....	113
33.2. Presisi Batas Waktu T0 dan T1	114
33.3. Konversi Timezone Aman	116
33.4. Detak Jam Lokal Dinamis.....	116
34. Informasi Lokasi dan Cuaca.....	120
35. Update Lokasi Pengamat.....	121
36. Halaman Informasi Rilis & Dokumentasi.....	127
36.1. Header Informasi Rilis.....	127
36.2. Executive Summary Dashboard	127

36.3. Integrasi Kalender Hijriah Interaktif.....	128
36.4. Legend Hari Penting Islam.....	129
36.5. Parameter Astronomi dan Lokasi.....	130
36.6. Fitur Print/Ekspor Kalender.....	130
36.7. Kalkulasi Fase Bulan Bulanan.....	132
36.8. Kalkulasi Astronomi saat Maghrib.....	133
36.9. Intro — KHGT Times V8.0.....	136
36.10. Bagian 1: Teknologi & Engine.....	137
36.11. Bagian 2: Konsep Dasar KHGT.....	138
36.12. Bagian 3: Penjelasan 41 Modul Menu V8.0.....	139
36.13. Bagian 4: Highlight Pembaruan.....	147
36.14. Bagian 5: Panduan Penggunaan.....	148
36.15. Footer Copyleft.....	150
37. Routing Tombol Proses dan Eksekusi Modul.....	154
BAGIAN VI — ENGINE ANALISIS HILAL DAN GERHANA	
38. Analisis Hilal Global Iteratif — PyEphem.....	156
38.1. Sinkronisasi Pencarian Ijtimak.....	157
38.2. Loop Database Kota.....	157
38.3. Evaluasi PKG 1 dan PKG 2.....	159
38.4. Penyusunan Output Analisis Global.....	162
39. Analisis Gerhana Algoritmik.....	162
39.1. Pencarian Gerhana Bulan dengan Vector Math.....	165
39.2. Pencarian Gerhana Matahari.....	167
BAGIAN VII — LIVE ANIMASI DAN VISUALISASI 3D	
40. Live Animasi — Full Scroll & Rasi Bintang.....	174
40.1. Scrollable Frame.....	174
40.2. State Kotak Informasi Interaktif.....	175
40.3. Event Binding Canvas.....	175
40.4. Kalkulasi Posisi Planet.....	178
40.5. Proyeksi Visual Normal dan Zoom.....	179
40.6. Mode Zoom Hilal.....	179
40.7. Mode Normal Horizon.....	181
40.8. Render Rasi Bintang.....	182
40.9. Render Planet.....	184
40.10. Garis Elongasi Matahari–Bulan.....	185
40.11. Render Matahari.....	185
40.12. Render Bulan dan Fase Mask.....	185
40.13. Proyeksi Tinggi ke Ufuk.....	187
40.14. Kotak Status Tracking Hilal.....	187
41. 3D Space System with Zoom.....	192
BAGIAN VIII — KONVERSI KALENDER DAN VISIBILITAS KHGT	
42. Fungsi Konversi KHGT.....	198
42.1. Normalisasi Bulan Multibahasa.....	199
42.2. Masehi → Hijriah KHGT.....	203
42.3. Hijriah KHGT → Masehi.....	204
42.4. Masehi → Hijriah Urfi/Kuwait.....	205
42.5. Hijriah Urfi/Kuwait → Masehi.....	206
42.6. Render Laporan Konversi.....	206
43. Analisis Visibilitas Hilal Lokal dan Global.....	207
43.1. Sunset Lokal Titik Observasi.....	207
43.2. Parameter Lokal pada Waktu Evaluasi.....	209
43.3. Pemindaian Global KHGT — PKG 1 & PKG 2.....	211
43.4. Prioritas Titik Lokal.....	212
43.5. Scan Global PKG 1.....	212
43.6. Evaluasi PKG 2 di Benua Amerika.....	213
BAGIAN IX — WAKTU BENDA LANGIT, SHALAT, DAN PEMETAAN	
44. Kalkulasi Rise, Transit, dan Set.....	228
45. Moon Times — Laporan Lokal.....	229
46. Sun Times — Laporan Lokal.....	231

47. Prayer Times — Engine Jadwal Shalat	233
47.1. Metode dan Mazhab	233
47.2. Safe Month Range	234
47.3. Looping Hisab Jadwal	236
47.4. Output Teks dan Generator Gambar	238
48. Crescent Visibility Map dan Peta KHGT	243
48.1. Peta KHGT Dua Hari Seamless	243
48.2. Render Peta Visibilitas Standar	245
48.3. Plotting Peta	247
48.4. Kalkulasi Data Astronomi Peta	254
48.5. Legenda dan Keterangan Dinamis	256
48.6. Evaluasi PKG Terintegrasi Konversi Tanggal	261
BAGIAN X — LOKASI, EKSPOR, DAN KOMPARASI 50 TAHUN	
49. Sinkronisasi dan Update Lokasi	270
50. Ekspor Data dan Laporan	277
50.1. Ekspor Kalender	277
50.2. Ekspor Modul Teks, 3D, Map, Chart, dan Gerhana	277
50.3. Ekspor Kalender Masehi	279
51. Fungsi Pembersih Saat Aplikasi Ditutup	282
52. Komparasi KHGT dan Neo-MABIMS 50 Tahun	283
52.1. Zona Waktu Indonesia dan Alaska	283
52.2. Komparasi Ramadan	290
52.3. Tabel Tinggi/Elongasi Ramadan	290
52.4. Komparasi Syawal	294
52.5. Tabel Tinggi/Elongasi Syawal	299
52.6. Komparasi Zulhijah	302
BAGIAN XI — KALENDER INTERAKTIF DAN ALTITUDE CHART	
53. Kalender Hijriah Berjalan	312
53.1. Navigasi dan Cetak	313
53.2. Grid Kalender	313
53.3. Hari Penting Islam	313
53.4. Highlight Hari Ini	315
53.5. Event Sejarah Islam	316
53.6. Ekspor Kalender Hijriah	319
54. Kalender Masehi Berjalan	320
54.1. Grid Kalender	321
54.2. Hari Penting Nasional dan Internasional	324
54.3. Ekspor Kalender Masehi	328
55. Altitude Chart Analyser — Final & Anti-Freeze	329
55.1. Kalkulasi Background Thread	330
55.2. Data Toposentris	330
55.3. Data Geosentris	330
55.4. Data Iluminasi	330
55.5. Common Styling	333
55.6. Tooltip Hover	333
BAGIAN XII — WINAI DAN SIMULATOR GERHANA	
56. WinAI — AI Chatbot Assistant	335
56.1. Variabel dan Konfigurasi API	335
56.2. Main Frame Output	336
56.3. Pembersihan dan Format Dokumen	339
56.4. Smart Interceptor / Agentic AI Control	340
56.5. Routing Intent ke Modul	340
56.6. Pengamanan Output AI	344
56.7. Rotasi Anti-Macet	345
56.8. Fallback Hisab saat AI Gagal	346
57. Simulator Visual Gerhana Matahari & Bulan	348
57.1. UI Simulator Gerhana	349
57.2. Pengaman math.acos()	350
57.3. Gerhana Bulan Geosentris	352

57.4. Fallback Pandangan Geosentris	353
57.5. Render Gerhana Bulan	355
BAGIAN XIII — MODUL FISIS DAN OBSERVASI LANJUTAN	
58. Kalkulasi Zuhur dan Dhuha	358
59. Sistem Flag dan Pembacaan Waktu	359
60. Live/Custom Animation Rendering	362
61. Loop Animasi Real-Time	365
62. Kalkulator Astrofotografi & Kontras	372
62.1. Helper Cuaca Universal	373
62.2. Integrasi API Cuaca	374
63. Prediktor Pasang Surut Gravitasi	376
64. Evaluasi Fajar Shadiq & SQM Teoritis	379
64.1. Integrasi API Cuaca	379
64.2. Pelacakan Altitude -18° dan -20°	380
65. Generator Analemma Matahari	381
65.1. Integrasi API Cuaca	382
65.2. Logika Astronomi Analemma	382
66. Fungsi Detektor Okultasi	386
67. Kalkulator Shalat Penerbangan — In-Flight Prayer Times	388
68. Kurva Alaska	395
69. Tren Bulanan Hijriah — Rajab sampai Zulhijah	399
69.1. Kalkulasi Toposentris	401
69.2. Kalkulasi Geosentris	401
69.3. Iluminasi	402
69.4. Plot Kombinasi	403
69.5. Tooltip Hover Matplotlib	404
BAGIAN XIV — SISTEM MULTIBAHASA	
70. Sistem Multi-Bahasa — JSON Translator	405
70.1. Lokasi Cache Bahasa	406
70.2. Indeks dan Reverse Translation	407
70.3. Cache UI Utama	408
70.4. Built-in Translation Dictionary	409
70.5. Terjemahan Panduan Release	410
70.6. Terjemahan 41 Menu	412
70.7. Terjemahan Label dan Parameter GUI	414
70.8. Release Language Cache	425
70.9. Translation Regex dan Fallback	426
70.10. Translation Output Cache	428
70.11. Localized Textbox dan Output Module	431
70.12. Language Hook untuk Secondary Textbox	432
70.13. Translasi Rekursif Widget GUI	438
AKHIR SOURCE CODE KHGT TIMES V8.0	438
DAFTAR PUSTAKA DAN REFERENSI TEKNIS	439
GLOSARIUM	444
SINGKATAN PENTING	457
CATATAN PENGGUNAAN GLOSARIUM	461

SOURCE CODE KHGT TIMES V8.0

```
import sys
import io
import math
import datetime
import calendar
import os
import threading
import urllib.request
import ssl
import textwrap
from collections import defaultdict
import random
import csv

# ---> TAMBAHKAN 2 BARIS INI DI SINI <---
import platform
import subprocess
# -----

import numpy as np
import pytz
import ephem

import warnings
warnings.filterwarnings("ignore")

# ---> TAMBAHAN IMPORT UNTUK WINAI <---
from duckduckgo_search import DDGS
from bs4 import BeautifulSoup
import requests
import json
import hashlib
import webbrowser
import urllib.parse
import re

# --- KONFIGURASI URL SERVER WINAI ---
API_URL = "https://hisabmu.com/aifikih/system_php.php"
SYSTEM_PROMPT_URL = "https://hisabmu.com/aifikih/system_prompt.php"
CALENDAR_DATA_URL = "https://hisabmu.com/aifikih/tahun4.php"
GOOGLE_SESSION_COOKIES = {}
WORLD_SKY_MAP_URL = "https://kasmui.cloud/langit/"

# --- KONFIGURASI EPHEMERIS V8.0 (FOKUS MATAHARI & BULAN) ---
```

SOURCE CODE KHGT TIMES V8.0


```

# de440s dipakai untuk rentang modern yang seluruh tahunnya tercakup penuh.
# de440s sendiri membentang sekitar 26-12-1849 s/d 22-01-2150; karena banyak
# modul memilih kernel berdasarkan TAHUN, batas aman tahun penuh adalah 1850-2149.
# Di luar rentang aman tersebut aplikasi otomatis memakai de441-new.bsp.
EPHEMERIS_SHORT_FILE = "de440s.bsp"
EPHEMERIS_LONG_FILE = "de441-new.bsp"
EPHEMERIS_SHORT_MIN_YEAR = 1850
EPHEMERIS_SHORT_MAX_YEAR = 2149
EPHEMERIS_LONG_MIN_YEAR = 1
EPHEMERIS_LONG_MAX_YEAR = 2999
# -----
# -----

# V8.0: terjemahan lengkap highlight rilis, tersedia tanpa layanan daring.
# Nama file, nomor menu, dan versi historis tetap mengikuti teks sumber.
RELEASE_HIGHLIGHT_TRANSLATIONS = {
    "[ 4. HIGHLIGHT PEMBARUAN EKSKLUSIF (VERSI 8.0) ]": {
        "en": "[ 4. EXCLUSIVE UPDATE HIGHLIGHTS (VERSION 8.0) ]",
        "ar": "[ أبرز التحديثات الحصرية (الإصدار 8.0) ]"
    },
    "[V7.7 - MULTIBAHASA & SIDEBAR] Antarmuka Indonesia, Inggris & Arab": {
        "en": "[V7.7 - MULTILINGUAL & SIDEBAR] Indonesian, English & Arabic Interface",
        "ar": "[واجهة بالإنجليزية والعربية (تعدد اللغات والشريط الجانبي - V7.7)]"
    },
    "Sidebar dan elemen antarmuka utama kini mendukung penerjemahan ke Bahasa Indonesia, Inggris, dan Arab sehingga navigasi modul dapat digunakan secara multibahasa.": {
        "en": "The sidebar and main interface elements now support translation into Indonesian, English, and Arabic, allowing multilingual navigation between modules.",
        "ar": "يدعم الشريط الجانبي وعناصر الواجهة الرئيسية الآن الترجمة إلى الإندونيسية والإنجليزية والعربية، مما يتيح التنقل بين الوحدات بلغات متعددة."
    },
    "[V7.7 - ALTITUDE CHART ANALYZER] Analisis Grafik & Kombinasi": {
        "en": "[V7.7 - ALTITUDE CHART ANALYZER] Chart Analysis & Data Combinations",
        "ar": "[تحليل المخططات ودمج البيانات (محلل مخطط الارتفاع - V7.7)]"
    },
    "Menu 9 (Altitude Chart Analyser) menyediakan analisis grafik ketinggian dan kombinasi data untuk membantu pembacaan posisi benda langit secara lebih visual dan terstruktur.": {
        "en": "Menu 9 (Altitude Chart Analyser) provides altitude chart analysis and data combinations to help interpret celestial object positions in a more visual and structured way.",
        "ar": "توفر القائمة 9 (محلل مخطط الارتفاع) تحليل مخططات الارتفاع ودمج البيانات للمساعدة في قراءة مواقع الأجرام السماوية بطريقة أكثر وضوحاً وتنظيماً."
    },
    "[V8.0 - PETA LANGIT DUNIA] Integrasi Peta Langit Online": {
        "en": "[V8.0 - WORLD SKY MAP] Online Sky Map Integration",
        "ar": "[تكامل خريطة السماء عبر الإنترنت (خريطة السماء العالمية - V8.0)]"
    },
}

```

```

"Menu 39 adalah Peta Langit Dunia. Saat menu dipilih, KHGT Times membuka layanan peta langit online pada https://kasmui.cloud/langit/ melalui browser default.": {
  "en": "Menu 39 is World Sky Map. When selected, KHGT Times opens the online sky map service at https://kasmui.cloud/langit/ in the default browser.",
  "ar": "خدمة خريطة السماء عبر الإنترنت على KHGT Times القائمة 39 هي خريطة السماء العالمية. عند اختيارها، يفتح https://kasmui.cloud/langit/ في المتصفح الافتراضي."
},
"[ 3. MANUAL BOOK: PENJELASAN 41 MODUL MENU V8.0 ]": {
  "en": "[ 3. MANUAL BOOK: EXPLANATION OF 41 V8.0 MENU MODULES ]",
  "ar": "[ 3. وحدة قائمة في V8.0 دليل الاستخدام: شرح 41 وحدة قائمة في V8.0 ]"
},
"• Kegunaan: Membuka Peta Langit Dunia KHGT Times berbasis web untuk eksplorasi langit secara online.": {
  "en": "• Purpose: Opens the web-based KHGT Times World Sky Map for online sky exploration.",
  "ar": "عبر الويب لاستكشاف السماء عبر الإنترنت KHGT Times الاستخدام: فتح خريطة السماء العالمية لـ."
},
"Layanan dibuka melalui browser default pada alamat https://kasmui.cloud/langit/.": {
  "en": "The service opens in the default browser at https://kasmui.cloud/langit/.",
  "ar": "https://kasmui.cloud/langit/ تُفتح الخدمة في المتصفح الافتراضي على العنوان."
},
"• Sub-Menu Input: Tidak memerlukan input khusus pada aplikasi desktop.": {
  "en": "• Sub-Menu Input: No special input is required in the desktop application.",
  "ar": "إدخال القائمة الفرعية: لا يتطلب إدخالاً خاصاً في تطبيق سطح المكتب."
},
"• Aksi & Output: Membuka Peta Langit Dunia online pada browser default saat menu dipilih.": {
  "en": "• Action & Output: Opens the online World Sky Map in the default browser when the menu is selected.",
  "ar": "الإجراء والمخرجات: فتح خريطة السماء العالمية عبر الإنترنت في المتصفح الافتراضي عند اختيار القائمة."
},
"[V8.0 - LAPORAN ASTRONOMI] Penyesuaian Nomor Menu": {
  "en": "[V8.0 - ASTRONOMY REPORT] Menu Number Adjustment",
  "ar": "تعديل رقم القائمة [التقرير الفلكي - V8.0]"
},
"Menu 40 adalah Laporan Astronomi. Menu ini membuka aplikasi pendamping 'LaporanAstronomi.exe' atau 'LaporanAstronomi.exe' dari folder aplikasi KHGT Times dan menampilkan pesan kesalahan jika file pendamping tidak tersedia atau gagal dijalankan.": {
  "en": "Menu 40 is Astronomy Report. It opens the companion application 'LaporanAstronomi.exe' or 'LaporanAstronomi.exe' from the KHGT Times application folder and displays an error message if the companion file is unavailable or fails to launch.",
  "ar": "أو 'LaporanAstronomi.exe' القائمة 40 هي التقرير الفلكي. تفتح هذه القائمة التطبيق المرافق 'LaporanAstronomi.exe' وتعرض رسالة خطأ إذا كان ملف التطبيق المرافق غير متوفر أو KHGT Times من مجلد تطبيق 'LaporanAstronomi.exe' تعذر تشغيله."
},
"[V8.0 - MENUS & DOKUMENTASI] Penyelarasan 41 Modul": {
  "en": "[V8.0 - MENUS & DOCUMENTATION] Alignment of 41 Modules",
  "ar": "مواءمة 41 وحدة [القوائم والتوثيق - V8.0]"
},

```

"Daftar menu aktual V8.0 terdiri atas 41 menu. Menu 39 adalah Peta Langit Dunia, Menu 40 adalah Laporan Astronomi, dan Menu 41 adalah HIJRI_DB Auto-Builder (Admin). Manual Book, highlight rilis, dan panduan navigasi pada halaman ini telah diselaraskan dengan struktur aktual tersebut.": {

"en": "The actual V8.0 menu list contains 41 menus. Menu 39 is World Sky Map, Menu 40 is Astronomy Report, and Menu 41 is HIJRI_DB Auto-Builder (Admin). The manual book, release highlights, and navigation guide on this page have been aligned with this actual structure.",

"ar": "الفعالية 41 قائمة. القائمة 39 هي خريطة السماء العالمية، والقائمة 40 هي التقرير الفلكي، والقائمة V8.0 تتضمن قائمة"،
للمسؤول (تمت مواءمة دليل الاستخدام وأبرز تحديثات الإصدار ودليل التنقل في هذه HIJRI_DB Auto-Builder هي 41 الصفحة مع هذا الهيكل الفعلي".

},

"[V8.0 - ENGINE] Auto-Switch Ephemeris Matahari & Bulan": {

"en": "[V8.0 - ENGINE] Automatic Sun & Moon Ephemeris Switching",

"ar": "التبديل التلقائي لجداول الشمس والقمر الفلكية [المحرك - V8.0]"

},

"Sistem otomatis menggunakan de440s.bsp untuk tahun 1850-2149 dan beralih ke de441-new.bsp untuk tahun 1-1849 atau 2150-2999, sehingga perhitungan modern Matahari-Bulan memakai kernel ringkas berpresisi tinggi tanpa kehilangan jangkauan panjang aplikasi.": {

"en": "The system automatically uses de440s.bsp for years 1850-2149 and switches to de441-new.bsp for years 1-1849 or 2150-2999, so modern Sun-Moon calculations use the compact high-precision kernel without losing the application's long time span.",

"ar": "للسنوات 1- de441-new.bsp للسنوات 1850-2149، ويتحول إلى de440s.bsp يستخدم النظام تلقائياً الملف"،
أو 2150-2999، بحيث تستخدم حسابات الشمس والقمر الحديثة نواة مدمجة عالية الدقة مع الحفاظ على المدى الزمني 1849 الطويل للتطبيق".

},

"[V7.6 - LIVE ANIMASI] Sinkronisasi Stellarium": {

"en": "[V7.6 - LIVE ANIMATION] Stellarium Synchronization",

"ar": "المزامنة مع [التحريك المباشر - V7.6] Stellarium"

},

"Pada output Live Animasi ditambahkan tombol Sinkronisasi Stellarium untuk mengaktifkan Stellarium dan membandingkan data Matahari dan Bulan antara output Stellarium dengan hasil hitung sistem KHGT berbasis ephemeris DE441.": {

"en": "The Live Animation output now includes a Stellarium Synchronization button to launch Stellarium and compare its Sun and Moon data with calculation results from the KHGT system based on DE441 ephemeris.",

"ar": "لتشغيله ومقارنة بيانات الشمس والقمر في مخرجاته Stellarium أُضيف إلى مخرجات التحريك المباشر زر المزامنة مع"،
DE441 المعتمدة على الجداول الفلكية KHGT بنتائج حسابات نظام".

},

"[V7.6 - LIVE ANIMASI] Tracker Benda Langit": {

"en": "[V7.6 - LIVE ANIMATION] Celestial Object Tracker",

"ar": "متتبع الأجرام السماوية [التحريك المباشر - V7.6]"

},

"Pada output Live Animasi ditambahkan tombol Tracker untuk pelacakan posisi benda langit, baik menggunakan GPS maupun tanpa GPS.": {

"en": "The Live Animation output now includes a Tracker button for tracking celestial object positions, with or without GPS.",

"ar": "أو بدون GPS أُضيف إلى مخرجات التحريك المباشر زر التتبع لرصد مواقع الأجرام السماوية، سواء باستخدام"،

},

"[V7.5 - LIVE ANIMASI] Rasi Bintang & Planet": {

```

"en": "[V7.5 - LIVE ANIMATION] Constellations & Planets",
"ar": "[V7.5 - التحويلات والكواكب [التحريك المباشر - V7.5]]",
},
"Pembaruan pada Modul 36 (Live Animasi) dilengkapi visualisasi Rasi Bintang (Constellations) dan planet-planet tata surya secara dinamis dan real-time.": {
  "en": "The update to Module 36 (Live Animation) adds dynamic, real-time visualizations of constellations and the planets of the solar system.",
  "ar": "يضيف تحديث الوحدة 36 (التحريك المباشر) عرضاً مرئياً ديناميكياً للكوكبات وكواكب المجموعة الشمسية في الوقت الفعلي.",
},
"[V7.5 - WINAI] Asisten AI Universal": {
  "en": "[V7.5 - WINAI] Universal AI Assistant",
  "ar": "[V7.5 - WINAI] مساعد ذكاء اصطناعي شامل",
},
"Peningkatan kecerdasan WinAI Interceptor agar mampu mendeteksi niat (intent) pengguna dan mengeksekusi pertanyaan atau perintah perhitungan yang berkaitan dengan menu di dalam sistem.": {
  "en": "WinAI Interceptor has been enhanced to detect user intent and execute questions or calculation commands related to menus within the system.",
  "ar": "ليتعرف على قصد المستخدم وينفذ الاستفسارات أو أوامر الحساب المتعلقة WinAI Interceptor تم تحسين ذكاء بالقوائم داخل النظام.",
},
"[V7.5 - ANALEMMA] Grafik Kurva Matahari": {
  "en": "[V7.5 - ANALEMMA] Solar Curve Chart",
  "ar": "[V7.5 - أناليمّا] مخطط منحنى الشمس",
},
"Peningkatan visual pada Modul 33 (Generator Analemma Matahari) yang menampilkan grafik kurva lintasan Matahari berbentuk angka 8.": {
  "en": "The visual presentation of Module 33 (Solar Analemma Generator) has been enhanced to display the Sun's path as a figure-eight curve.",
  "ar": "تم تحسين العرض المرئي في الوحدة 33 (مولد أناليمّا الشمس) لإظهار مخطط منحنى مسار الشمس على شكل الرقم 8.",
},
"[V7.5 - BUG FIXES] Gerhana, Kiblat & Optimalisasi": {
  "en": "[V7.5 - BUG FIXES] Eclipses, Qibla & Optimization",
  "ar": "[V7.5 - الكسوف والخسوف والقبلة والتحسين [إصلاح الأخطاء - V7.5]]",
},
"Perbaikan bug pada rentang waktu komputasi di Modul 30 (Analisis Gerhana), perbaikan akurasi pada Modul 26 (Qiblah Direction and Times), serta berbagai penyesuaian stabilitas sistem.": {
  "en": "Bugs in computation time ranges in Module 30 (Eclipse Analysis) have been fixed, accuracy in Module 26 (Qiblah Direction and Times) has been corrected, and various system stability adjustments have been made.",
  "ar": "تم إصلاح أخطاء النطاقات الزمنية للحساب في الوحدة 30 (تحليل الكسوف والخسوف)، وتصحيح الدقة في الوحدة 26 (اتجاه القبلة وأوقاتها)، وإجراء تعديلات متنوعة لتحسين استقرار النظام.",
},
"[V7.4 - ENGINE] Penyederhanaan Ephemeris DE441": {
  "en": "[V7.4 - ENGINE] DE441 Ephemeris Simplification",
  "ar": "[V7.4 - المحرك] تبسيط الجداول الفلكية [DE441]",
},

```



```

"Sistem menggunakan file ephemeris 'de441-new.bsp' untuk kalkulasi pada rentang waktu yang
didukung aplikasi, sehingga pengelolaan basis ephemeris menjadi lebih sederhana.": {
  "en": "The system uses the ephemeris file 'de441-new.bsp' for calculations over the time range
supported by the application, simplifying ephemeris data management.",
  "ar": "الحسابات ضمن النطاق الزمني الذي يدعمه التطبيق، مما 'de441-new.bsp' يستخدم النظام ملف الجداول الفلكية
للسهولة. يبسط إدارة بيانات الجداول الفلكية."
},
"[V7.3 - HIGHLIGHTS] Modul Fisis & Auto-Builder": {
  "en": "[V7.3 - HIGHLIGHTS] Physical Modules & Auto-Builder",
  "ar": "الوحدات الفيزيائية والبناء التلقائي [أبرز التحديثات - V7.3]"
},
"Modul Astrofotografi, Prediktor Pasang Surut, Evaluasi SQM Fajar, dan HIJRI_DB Auto-Builder untuk
Administrator merupakan pengembangan yang diperkenalkan pada rilis sebelumnya.": {
  "en": "The Astrophotography, Tide Predictor, Dawn SQM Evaluation, and HIJRI_DB Auto-Builder
modules for administrators were introduced in earlier releases.",
  "ar": "للمسؤولين هي HIJRI_DB Auto-Builder عند الفجر، و SQM وحدات التصوير الفلكي، والتنبؤ بالمد والجزر، وتقييم."
  "تطويرات قدمت في إصدارات سابقة."
},
'Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak komprehensif dan mutakhir untuk
perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal global, dan penyatuan penanggalan Islam
internasional serta menjadi sarana edukasi publik terkait ilmu falak Islami dan transisi penerapan KHGT
di masa depan. Pada V8.0, sistem menyediakan 41 modul menu terintegrasi, antarmuka multibahasa
Indonesia-Inggris-Arab, Peta Langit Dunia online pada Menu 39, akses ke aplikasi pendamping Laporan
Astronomi pada Menu 40, serta HIJRI_DB Auto-Builder (Admin) pada Menu 41.': {
  "en": 'This application is specifically developed as comprehensive and up-to-date software for high-
precision astrometric calculations, global crescent-visibility mapping, and the unification of international
Islamic calendaring, while also serving as a public educational tool for Islamic astronomy and the future
transition toward KHGT implementation. In V8.0, the system provides 41 integrated menu modules, an
Indonesian-English-Arabic multilingual interface, an online World Sky Map in Menu 39, access to the
Astronomy Report companion application in Menu 40, and HIJRI_DB Auto-Builder (Admin) in Menu 41.',
  "ar": "تم تطوير هذا التطبيق خصيصاً ليكون برنامجاً شاملاً وحديثاً للحسابات الفلكية عالية الدقة، ورسم خرائط عالمية لرؤية
الهلال، وتوحيد التقويم الإسلامي الدولي، كما يُستخدم أداةً للتحقيق العام في علم الفلك الإسلامي والانتقال المستقبلي إلى تطبيق
يوفر النظام 41 وحدة قائمة متكاملة، وواجهة متعددة اللغات بالإنجليزية والعربية، وخريطة V8.0 في الإصدار KHGT.
HIJRI_DB Auto-Builder السماء العالمية عبر الإنترنت في القائمة 39، والوصول إلى تطبيق التقرير الفلكي المرافق في القائمة 40، و
للمسؤول (في القائمة 41).'
},
"INFORMASI RILIS & DOKUMENTASI SISTEM KHGT TIMES V8.0": {
  "en": "KHGT TIMES V8.0 RELEASE INFORMATION & SYSTEM DOCUMENTATION",
  "ar": "KHGT TIMES V8.0 معلومات الإصدار وتوثيق نظام"
},
"Selamat datang di KHGT Times V8.0": {
  "en": "Welcome to KHGT Times V8.0",
  "ar": "KHGT Times V8.0 مرحباً بكم في"
}
}

```

--- GUI LIBRARIES ---

SOURCE CODE KHGT TIMES V8.0

```

import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox, filedialog, ttk
from PIL import Image, ImageTk, ImageDraw, ImageFont, ImageGrab
from CtkToolTip import *

# --- SKYFIELD LIBRARIES ---
if not hasattr(CtkToolTip, 'block_update_dimensions_event'):
    setattr(CtkToolTip, 'block_update_dimensions_event', lambda self: None)
if not hasattr(CtkToolTip, 'unblock_update_dimensions_event'):
    setattr(CtkToolTip, 'unblock_update_dimensions_event', lambda self: None)
# -----
from skyfield.api import Loader, wgs84
from skyfield import almanac, eclipselib
from skyfield.positionlib import Geocentric

# --- MATPLOTLIB LIBRARIES ---
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
from matplotlib.colors import ListedColormap, BoundaryNorm

# --- TAMBAHAN DEPENDENSI UNTUK PETA HD & INTERPOLASI ---
try:
    import scipy.interpolate as interp
    HAS_MAP_TOOLS = True
except ImportError:
    HAS_MAP_TOOLS = False

# --- TAMBAHAN DEPENDENSI MODUL 08: ALARM & NOTIFIKASI ---
try:
    from win10toast import ToastNotifier
    HAS_TOAST = True
except ImportError:
    HAS_TOAST = False

try:
    import pygame
    HAS_PYGAME = True
except ImportError:
    HAS_PYGAME = False
# -----

# =====
# PERBAIKAN CRASH GUI & SSL UNTUK EXE
# =====

```

SOURCE CODE KHGT TIMES V8.0

```

if sys.stdout is None: sys.stdout = io.StringIO()
if sys.stderr is None: sys.stderr = io.StringIO()

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

if getattr(sys, 'frozen', False):
    BASE_DIR = os.path.dirname(sys.executable)
else:
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))

ctk.set_appearance_mode("Dark")
ctk.set_default_color_theme("blue")

# TAMBAHKAN FUNGSI INI DI BAGIAN ATAS (BAWAH IMPORT)
def safe_montrange(year, month):
    """Pengganti calendar.montrange yang kebal tahun minus/BCE"""
    is_leap = year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
    days_in_month = [31, 29 if is_leap else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    return 0, days_in_month[month - 1]

def format_tahun_aman(year):
    """Format tahun agar 0 menjadi 1 SM, -1 menjadi 2 SM, dst"""
    return f"{year} M" if year > 0 else f"{abs(year-1)} SM"

# =====
# FUNGSI UTILITAS GAMBAR JADWAL SHALAT
# (Disisipkan dari aplikasi terpisah)
# =====

def _util_get_nama_bulan(angka_bulan):
    bulan = {
        '01': 'JANUARI', '02': 'FEBRUARI', '03': 'MARET', '04': 'APRIL',
        '05': 'MEI', '06': 'JUNI', '07': 'JULI', '08': 'AGUSTUS',
        '09': 'SEPTEMBER', '10': 'OKTOBER', '11': 'NOVEMBER', '12': 'DESEMBER'
    }
    return bulan.get(angka_bulan, "")

def _util_parse_jadwal_text(raw_text):
    data = []
    lines = raw_text.strip().split('\n')

    bulan_angka = "01"
    tahun = "2026"

```

```

koordinat_str = ""

import re
for line in lines:
    # PERBAIKAN: Tangkap LOKASI lengkap dengan nama kota dan provinsinya
    if "- LOKASI:" in line:
        koordinat_str = line.split("- LOKASI:")[1].strip()

    match_judul = re.search(r'Prayer times for:\s*.*?(\d{2})/(\d{4})', line)
    if match_judul:
        bulan_angka = match_judul.group(1)
        tahun = match_judul.group(2)

    # Ekstrak data tabel menghindari pergeseran spasi
    match_row = re.match(r'^\s*(\d{2}/\d{2}/\d{4}.*)\s{2,}(.*)', line)
    if match_row:
        tanggal_full = match_row.group(1).strip()
        hari = str(int(tanggal_full.split('/')[0]))

        times_str = match_row.group(2)
        parts = [p for p in times_str.strip().split(' ') if p]

        if len(parts) >= 8: # Ada 8 Kolom (Subuh s/d Midnight)
            data.append({
                'hari': hari,
                'subuh': parts[0],
                'terbit': parts[1],
                'duha': parts[2],
                'zuhur': parts[3],
                'asar': parts[4],
                'magrib': parts[5],
                'isya': parts[6],
                'midnight': parts[7]
            })

    nama_bulan_upper = _util_get_nama_bulan(bulan_angka)
    return data, nama_bulan_upper, bulan_angka, tahun, koordinat_str

def _util_generate_image(raw_text, template_path, output_path):
    jadwal_data, bulan_upper, bulan_angka, tahun, koordinat_str = _util_parse_jadwal_text(raw_text)

    if not jadwal_data:
        raise ValueError("Data jadwal harian tidak ditemukan dalam teks laporan. Gambar tidak dibuat.")

    try:
        img = Image.open(template_path)
    except FileNotFoundError:
        raise FileNotFoundError(f"File template '{template_path}' tidak ditemukan di folder aplikasi.")

```

```

draw = ImageDraw.Draw(img)

try:
    font_judul = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 60)
    font_sub_judul1 = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 36)
    font_sub_judul2 = ImageFont.truetype(os.path.join(BASE_DIR, "arial.ttf"), 26)
    font_header = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 34)
    font_sub_header = ImageFont.truetype(os.path.join(BASE_DIR, "arial.ttf"), 22)
    font_tabel = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 34)
except IOError:
    font_judul = font_sub_judul1 = font_sub_judul2 = font_header = font_sub_header = font_tabel =
    ImageFont.load_default()

center_x = img.width // 2
col_x = {
    'hari': center_x - 860,
    'tanggal': center_x - 650,
    'subuh': center_x - 455,
    'terbit': center_x - 265,
    'duha': center_x - 80,
    'zuhur': center_x + 120,
    'asar': center_x + 305,
    'magrib': center_x + 495,
    'isya': center_x + 680,
    'midnight': center_x + 880
}

start_y_judul = 120
row_spacing_judul = 60
start_y_header = 330
row_spacing_header = 45
start_y_data = 430
row_spacing_data = 40

warna_putih = (255, 255, 255)
warna_emas = (255, 204, 0)
warna_kotak_gelap = (20, 20, 20)

draw.text((center_x, start_y_judul), "JADWAL SHALAT", font=font_judul, fill=warna_emas,
anchor="mm")
teks_bulan_tahun = f"BULAN {bulan_upper} {tahun}"
draw.text((center_x, start_y_judul + row_spacing_judul), teks_bulan_tahun, font=font_sub_judul1,
fill=warna_putih, anchor="mm")

# PERBAIKAN: Menampilkan Kota dan Provinsi dengan rapi
if koordinat_str:
    teks_wilayah = f"LOKASI: {koordinat_str.upper()}"

```



```

else:
    teks_wilayah = "LOKASI: DATA TIDAK DITEMUKAN"

    draw.text((center_x, start_y_judul + 2 * row_spacing_judul), teks_wilayah, font=font_sub_judul2,
    fill=warna_putih, anchor="mm")

    padding_kotak_header = 1100
    kotak_header_x0 = center_x - padding_kotak_header
    kotak_header_y0 = start_y_header - 35
    kotak_header_x1 = center_x + padding_kotak_header
    kotak_header_y1 = start_y_header + 35
    draw.rounded_rectangle([kotak_header_x0, kotak_header_y0, kotak_header_x1, kotak_header_y1],
    fill=warna_kotak_gelap, radius=10)

    headers = [
        ("NOMOR", 'hari'), ("TANGGAL", 'tanggal'), ("SUBUH", 'subuh'), ("TERBIT", 'terbit'),
        ("DUHA", 'duha'), ("ZUHUR", 'zuhur'), ("ASAR", 'asar'), ("MAGRIB", 'magrib'),
        ("ISYA", 'isya'), ("TENGAH MALAM", 'midnight')
    ]
    for text, key in headers:
        draw.text((col_x[key], start_y_header), text, font=font_header, fill=warna_putih, anchor="mm")

    sub_headers = [
        ("", 'hari'), ("", 'tanggal'), ("B. Twi.", 'subuh'), ("Sunrise", 'terbit'),
        ("+4.5°", 'duha'), ("Transit", 'zuhur'), ("-----", 'asar'), ("Sunset", 'magrib'),
        ("E. Twi.", 'isya'), ("(Mid/Last 1/3)", 'midnight')
    ]
    for text, key in sub_headers:
        draw.text((col_x[key], start_y_header + row_spacing_header), text, font=font_sub_header,
    fill=warna_putih, anchor="mm")

    for i, row in enumerate(jadwal_data):
        current_y = start_y_data + (i * row_spacing_data)
        tanggal_hari = row['hari'].zfill(2)
        tanggal_str = f"{tanggal_hari}/{bulan_angka}/{tahun}"

        color = warna_emas if row['hari'] == '1' else warna_putih

        draw.text((col_x['hari'], current_y), row['hari'], font=font_tabel, fill=color, anchor="mm")
        draw.text((col_x['tanggal'], current_y), tanggal_str, font=font_tabel, fill=warna_putih,
    anchor="mm")
        draw.text((col_x['subuh'], current_y), row['subuh'], font=font_tabel, fill=warna_putih,
    anchor="mm")
        draw.text((col_x['terbit'], current_y), row['terbit'], font=font_tabel, fill=warna_putih, anchor="mm")
        draw.text((col_x['duha'], current_y), row['duha'], font=font_tabel, fill=warna_putih, anchor="mm")
        draw.text((col_x['zuhur'], current_y), row['zuhur'], font=font_tabel, fill=warna_putih,
    anchor="mm")
        draw.text((col_x['asar'], current_y), row['asar'], font=font_tabel, fill=warna_putih, anchor="mm")

```

```

        draw.text((col_x['magrib'], current_y), row['magrib'], font=font_tabel, fill=warna_putih,
        anchor="mm")
        draw.text((col_x['isya'], current_y), row['isya'], font=font_tabel, fill=warna_putih, anchor="mm")
        draw.text((col_x['midnight'], current_y), row['midnight'], font=font_tabel, fill=warna_putih,
        anchor="mm")

    img.save(output_path)
    return output_path

# =====
# DATABASE HIJRIAH & KALENDER
# =====
BULAN_MASEHI = ["Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus", "September",
"Oktober", "November", "Desember"]
BULAN_HIJRIAH = ["Muharam", "Safar", "Rabiulawal", "Rabiulakhir", "Jumadilawal", "Jumadilakhir",
"Rajab", "Syakban", "Ramadan", "Syawal", "Zulkaidah", "Zulhijah"]

HIJRI_DB = {
    1446: [{"Muharam", "Ahad Kliwon", "07-Jul-2024", 29}, {"Safar", "Senin", "05-Aug-2024", 30},
{"Rabiulawal", "Rabu", "04-Sep-2024", 30}, {"Rabiulakhir", "Jumat", "04-Oct-2024", 30}, {"Jumadilawal",
"Ahad", "03-Nov-2024", 29}, {"Jumadilakhir", "Senin", "02-Dec-2024", 30}, {"Rajab", "Rabu", "01-Jan-
2025", 30}, {"Syakban", "Jumat", "31-Jan-2025", 29}, {"Ramadan", "Sabtu", "01-Mar-2025", 30},
{"Syawal", "Ahad", "31-Mar-2025", 30}, {"Zulkaidah", "Selasa", "29-Apr-2025", 29}, {"Zulhijah", "Rabu",
"28-May-2025", 29}],
    1447: [{"Muharam", "Kamis Wage", "26-Jun-2025", 30}, {"Safar", "Sabtu Wage", "26-Jul-2025", 29},
{"Rabiulawal", "Ahad Pon", "24-Aug-2025", 30}, {"Rabiulakhir", "Selasa Pon", "23-Sep-2025", 30},
{"Jumadilawal", "Kamis Pon", "23-Oct-2025", 29}, {"Jumadilakhir", "Jumat Pahing", "21-Nov-2025", 30},
{"Rajab", "Ahad Pahing", "21-Dec-2025", 30}, {"Syakban", "Selasa Pahing", "20-Jan-2026", 29},
{"Ramadan", "Rabu Legi", "18-Feb-2026", 30}, {"Syawal", "Jumat Legi", "20-Mar-2026", 29}, {"Zulkaidah",
"Sabtu Kliwon", "18-Apr-2026", 30}, {"Zulhijah", "Senin Kliwon", "18-May-2026", 29}],
    1448: [{"Muharam", "Selasa Wage", "16-Jun-2026", 29}, {"Safar", "Rabu", "15-Jul-2026", 30},
{"Rabiulawal", "Jumat", "14-Aug-2026", 29}, {"Rabiulakhir", "Sabtu", "12-Sep-2026", 30}, {"Jumadilawal",
"Senin", "12-Oct-2026", 29}, {"Jumadilakhir", "Selasa", "10-Nov-2026", 30}, {"Rajab", "Kamis", "10-Dec-
2026", 30}, {"Syakban", "Sabtu", "09-Jan-2027", 30}, {"Ramadan", "Senin", "08-Feb-2027", 29},
{"Syawal", "Selasa", "09-Mar-2027", 30}, {"Zulkaidah", "Kamis", "08-Apr-2027", 29}, {"Zulhijah", "Jumat
Wage", "07-May-2027", 30}],
    #... [Data Hijriah disingkat untuk keterbacaan, logikanya tetap memuat data lengkap seperti aslinya]
}

# (Memasukkan sisa HIJRI_DB tahun 1449-1497 agar converter tetap berjalan sempurna sesuai kode
master Anda)
HIJRI_DB.update({
    1449: [{"Muharam", "Ahad Wage", "06-Jun-2027", 29}, {"Safar", "Senin", "05-Jul-2027", 29},
{"Rabiulawal", "Selasa", "03-Aug-2027", 30}, {"Rabiulakhir", "Kamis", "02-Sep-2027", 29},
{"Jumadilawal", "Jumat", "01-Oct-2027", 30}, {"Jumadilakhir", "Ahad", "31-Oct-2027", 29}, {"Rajab",
"Senin", "29-Nov-2027", 30}, {"Syakban", "Rabu", "29-Dec-2027", 29}, {"Ramadan", "Jumat", "28-Jan-

```

2028", 30], ["Syawal", "Sabtu", "26-Feb-2028", 30], ["Zulkaidah", "Senin", "27-Mar-2028", 30], ["Zulhijah", "Rabu", "26-Apr-2028", 29]],

1450: [{"Muharam", "Kamis", "25-May-2028", 30}, {"Safar", "Sabtu", "24-Jun-2028", 29}, {"Rabiulawal", "Ahad", "23-Jul-2028", 29}, {"Rabiulakhir", "Senin", "21-Aug-2028", 30}, {"Jumadilawal", "Rabu", "20-Sep-2028", 29}, {"Jumadilakhir", "Kamis", "19-Oct-2028", 30}, {"Rajab", "Sabtu", "18-Nov-2028", 29}, {"Syakban", "Ahad", "17-Dec-2028", 30}, {"Ramadan", "Selasa", "16-Jan-2029", 29}, {"Syawal", "Rabu", "14-Feb-2029", 30}, {"Zulkaidah", "Jumat", "16-Mar-2029", 30}, {"Zulhijah", "Ahad", "15-Apr-2029", 29}],

1451: [{"Muharam", "Senin", "14-May-2029", 30}, {"Safar", "Rabu", "13-Jun-2029", 29}, {"Rabiulawal", "Kamis", "12-Jul-2029", 30}, {"Rabiulakhir", "Sabtu", "11-Aug-2029", 29}, {"Jumadilawal", "Ahad", "09-Sep-2029", 30}, {"Jumadilakhir", "Selasa", "09-Oct-2029", 29}, {"Rajab", "Rabu", "07-Nov-2029", 30}, {"Syakban", "Jumat", "07-Dec-2029", 29}, {"Ramadan", "Sabtu", "05-Jan-2030", 30}, {"Syawal", "Senin", "04-Feb-2030", 29}, {"Zulkaidah", "Selasa", "05-Mar-2030", 30}, {"Zulhijah", "Kamis", "04-Apr-2030", 29}],

1452: [{"Muharam", "Jumat", "03-May-2030", 30}, {"Safar", "Ahad", "02-Jun-2030", 30}, {"Rabiulawal", "Selasa", "02-Jul-2030", 30}, {"Rabiulakhir", "Kamis", "01-Aug-2030", 29}, {"Jumadilawal", "Jumat", "30-Aug-2030", 29}, {"Jumadilakhir", "Sabtu", "28-Sep-2030", 30}, {"Rajab", "Senin", "28-Oct-2030", 29}, {"Syakban", "Selasa", "26-Nov-2030", 30}, {"Ramadan", "Kamis", "26-Dec-2030", 29}, {"Syawal", "Jumat", "24-Jan-2031", 29}, {"Zulkaidah", "Sabtu", "22-Feb-2031", 30}, {"Zulhijah", "Senin", "24-Mar-2031", 30}],

1453: [{"Muharam", "Rabu", "23-Apr-2031", 29}, {"Safar", "Kamis", "22-May-2031", 30}, {"Rabiulawal", "Sabtu", "21-Jun-2031", 30}, {"Rabiulakhir", "Senin", "21-Jul-2031", 29}, {"Jumadilawal", "Selasa", "19-Aug-2031", 30}, {"Jumadilakhir", "Kamis", "18-Sep-2031", 29}, {"Rajab", "Jumat", "17-Oct-2031", 30}, {"Syakban", "Ahad", "16-Nov-2031", 30}, {"Ramadan", "Senin", "15-Dec-2031", 29}, {"Syawal", "Rabu", "14-Jan-2032", 29}, {"Zulkaidah", "Kamis", "12-Feb-2032", 29}, {"Zulhijah", "Jumat", "12-Mar-2032", 30}],

1454: [{"Muharam", "Ahad", "11-Apr-2032", 30}, {"Safar", "Selasa", "11-May-2032", 29}, {"Rabiulawal", "Rabu", "09-Jun-2032", 30}, {"Rabiulakhir", "Jumat", "09-Jul-2032", 29}, {"Jumadilawal", "Sabtu", "07-Aug-2032", 30}, {"Jumadilakhir", "Senin", "06-Sep-2032", 29}, {"Rajab", "Selasa", "05-Oct-2032", 30}, {"Syakban", "Kamis", "04-Nov-2032", 30}, {"Ramadan", "Sabtu", "04-Dec-2032", 30}, {"Syawal", "Ahad", "02-Jan-2033", 29}, {"Zulkaidah", "Selasa", "01-Feb-2033", 29}, {"Zulhijah", "Rabu", "02-Mar-2033", 30}],

1455: [{"Muharam", "Jumat", "01-Apr-2033", 29}, {"Safar", "Sabtu", "30-Apr-2033", 29}, {"Rabiulawal", "Ahad", "29-May-2033", 30}, {"Rabiulakhir", "Selasa", "28-Jun-2033", 29}, {"Jumadilawal", "Rabu", "27-Jul-2033", 30}, {"Jumadilakhir", "Jumat", "26-Aug-2033", 29}, {"Rajab", "Sabtu", "24-Sep-2033", 30}, {"Syakban", "Senin", "24-Oct-2033", 30}, {"Ramadan", "Rabu", "23-Nov-2033", 30}, {"Syawal", "Jumat", "23-Dec-2033", 29}, {"Zulkaidah", "Sabtu", "21-Jan-2034", 30}, {"Zulhijah", "Senin", "20-Feb-2034", 29}],

1456: [{"Muharam", "Selasa", "21-Mar-2034", 30}, {"Safar", "Kamis", "20-Apr-2034", 29}, {"Rabiulawal", "Jumat", "19-May-2034", 29}, {"Rabiulakhir", "Sabtu", "17-Jun-2034", 30}, {"Jumadilawal", "Senin", "17-Jul-2034", 29}, {"Jumadilakhir", "Selasa", "15-Aug-2034", 29}, {"Rajab", "Rabu", "13-Sep-2034", 30}, {"Syakban", "Jumat", "13-Oct-2034", 30}, {"Ramadan", "Ahad", "12-Nov-2034", 30}, {"Syawal", "Selasa", "12-Dec-2034", 30}, {"Zulkaidah", "Kamis", "11-Jan-2035", 29}, {"Zulhijah", "Jumat", "09-Feb-2035", 30}],

1457: [{"Muharam", "Ahad", "11-Mar-2035", 29}, {"Safar", "Senin", "09-Apr-2035", 30}, {"Rabiulawal", "Rabu", "09-May-2035", 29}, {"Rabiulakhir", "Kamis", "07-Jun-2035", 29}, {"Jumadilawal", "Jumat", "06-Jul-2035", 29}, {"Jumadilakhir", "Sabtu", "04-Aug-2035", 30}, {"Rajab", "Senin", "03-Sep-2035", 30}, {"Syakban", "Rabu", "03-Oct-2035", 29}, {"Ramadan", "Kamis", "01-Nov-2035", 30}, {"Syawal", "Sabtu", "01-Dec-2035", 30}, {"Zulkaidah", "Senin", "31-Dec-2035", 30}, {"Zulhijah", "Rabu", "30-Jan-2036", 29}],

1458: [{"Muharam", "Kamis", "28-Feb-2036", 30}, {"Safar", "Sabtu", "29-Mar-2036", 29}, {"Rabiulawal", "Ahad", "27-Apr-2036", 30}, {"Rabiulakhir", "Selasa", "27-May-2036", 29}, {"Jumadilawal",

"Rabu", "25-Jun-2036", 29], ["Jumadilakhir", "Kamis", "24-Jul-2036", 30], ["Rajab", "Sabtu", "23-Aug-2036", 29], ["Syakban", "Ahad", "21-Sep-2036", 29], ["Ramadan", "Senin", "20-Oct-2036", 30], ["Syawal", "Rabu", "19-Nov-2036", 30], ["Zulkaidah", "Jumat", "19-Dec-2036", 30], ["Zulhijah", "Ahad", "18-Jan-2037", 29]],

1459: [{"Muharam", "Senin", "16-Feb-2037", 30}, {"Safar", "Rabu", "18-Mar-2037", 30}, {"Rabiulawal", "Jumat", "17-Apr-2037", 29}, {"Rabiulakhir", "Sabtu", "16-May-2037", 29}, {"Jumadilawal", "Ahad", "14-Jun-2037", 30}, {"Jumadilakhir", "Selasa", "14-Jul-2037", 29}, {"Rajab", "Rabu", "12-Aug-2037", 30}, {"Syakban", "Jumat", "11-Sep-2037", 29}, {"Ramadan", "Sabtu", "10-Oct-2037", 30}, {"Syawal", "Ahad", "08-Nov-2037", 30}, {"Zulkaidah", "Selasa", "08-Dec-2037", 30}, {"Zulhijah", "Kamis", "07-Jan-2038", 29}],

1460: [{"Muharam", "Jumat", "05-Feb-2038", 30}, {"Safar", "Ahad", "07-Mar-2038", 30}, {"Rabiulawal", "Senin", "05-Apr-2038", 29}, {"Rabiulakhir", "Rabu", "05-May-2038", 30}, {"Jumadilawal", "Jumat", "04-Jun-2038", 29}, {"Jumadilakhir", "Sabtu", "03-Jul-2038", 30}, {"Rajab", "Senin", "02-Aug-2038", 29}, {"Syakban", "Selasa", "31-Aug-2038", 30}, {"Ramadan", "Kamis", "30-Sep-2038", 29}, {"Syawal", "Jumat", "29-Oct-2038", 29}, {"Zulkaidah", "Sabtu", "27-Nov-2038", 30}, {"Zulhijah", "Senin", "27-Dec-2038", 30}],

1461: [{"Muharam", "Rabu", "26-Jan-2039", 29}, {"Safar", "Kamis", "24-Feb-2039", 30}, {"Rabiulawal", "Sabtu", "26-Mar-2039", 29}, {"Rabiulakhir", "Ahad", "24-Apr-2039", 30}, {"Jumadilawal", "Selasa", "24-May-2039", 29}, {"Jumadilakhir", "Rabu", "22-Jun-2039", 30}, {"Rajab", "Jumat", "22-Jul-2039", 30}, {"Syakban", "Ahad", "21-Aug-2039", 29}, {"Ramadan", "Senin", "19-Sep-2039", 30}, {"Syawal", "Rabu", "19-Oct-2039", 29}, {"Zulkaidah", "Kamis", "17-Nov-2039", 30}, {"Zulhijah", "Sabtu", "17-Dec-2039", 29}],

1462: [{"Muharam", "Ahad", "15-Jan-2040", 29}, {"Safar", "Senin", "13-Feb-2040", 30}, {"Rabiulawal", "Rabu", "14-Mar-2040", 29}, {"Rabiulakhir", "Kamis", "12-Apr-2040", 30}, {"Jumadilawal", "Sabtu", "12-May-2040", 29}, {"Jumadilakhir", "Ahad", "10-Jun-2040", 30}, {"Rajab", "Selasa", "10-Jul-2040", 30}, {"Syakban", "Kamis", "09-Aug-2040", 30}, {"Ramadan", "Jumat", "08-Sep-2040", 30}, {"Syawal", "Ahad", "07-Oct-2040", 30}, {"Zulkaidah", "Selasa", "06-Nov-2040", 29}, {"Zulhijah", "Rabu", "05-Dec-2040", 30}],

1463: [{"Muharam", "Jumat", "04-Jan-2041", 29}, {"Safar", "Sabtu", "02-Feb-2041", 29}, {"Rabiulawal", "Ahad", "03-Mar-2041", 30}, {"Rabiulakhir", "Selasa", "02-Apr-2041", 29}, {"Jumadilawal", "Rabu", "01-May-2041", 30}, {"Jumadilakhir", "Jumat", "31-May-2041", 29}, {"Rajab", "Sabtu", "29-Jun-2041", 30}, {"Syakban", "Senin", "29-Jul-2041", 29}, {"Ramadan", "Selasa", "27-Aug-2041", 30}, {"Syawal", "Kamis", "26-Sep-2041", 30}, {"Zulkaidah", "Sabtu", "26-Oct-2041", 30}, {"Zulhijah", "Senin", "25-Nov-2041", 29}],

1464: [{"Muharam", "Selasa", "24-Dec-2041", 30}, {"Safar", "Kamis", "23-Jan-2042", 29}, {"Rabiulawal", "Jumat", "21-Feb-2042", 30}, {"Rabiulakhir", "Ahad", "23-Mar-2042", 29}, {"Jumadilawal", "Senin", "21-Apr-2042", 29}, {"Jumadilakhir", "Selasa", "20-May-2042", 30}, {"Rajab", "Kamis", "19-Jun-2042", 29}, {"Syakban", "Jumat", "18-Jul-2042", 30}, {"Ramadan", "Ahad", "17-Aug-2042", 29}, {"Syawal", "Senin", "16-Sep-2042", 30}, {"Zulkaidah", "Rabu", "15-Oct-2042", 30}, {"Zulhijah", "Jumat", "14-Nov-2042", 30}],

1465: [{"Muharam", "Ahad", "14-Dec-2042", 29}, {"Safar", "Senin", "12-Jan-2043", 30}, {"Rabiulawal", "Rabu", "11-Feb-2043", 29}, {"Rabiulakhir", "Kamis", "12-Mar-2043", 30}, {"Jumadilawal", "Sabtu", "11-Apr-2043", 29}, {"Jumadilakhir", "Ahad", "10-May-2043", 30}, {"Rajab", "Senin", "08-Jun-2043", 29}, {"Syakban", "Selasa", "07-Jul-2043", 30}, {"Ramadan", "Kamis", "06-Aug-2043", 29}, {"Syawal", "Jumat", "04-Sep-2043", 30}, {"Zulkaidah", "Ahad", "04-Oct-2043", 30}, {"Zulhijah", "Selasa", "03-Nov-2043", 30}],

1466: [{"Muharam", "Kamis", "03-Dec-2043", 29}, {"Safar", "Jumat", "01-Jan-2044", 30}, {"Rabiulawal", "Ahad", "31-Jan-2044", 30}, {"Rabiulakhir", "Selasa", "01-Mar-2044", 29}, {"Jumadilawal", "Rabu", "30-Mar-2044", 30}, {"Jumadilakhir", "Jumat", "29-Apr-2044", 29}, {"Rajab", "Sabtu", "28-May-2044", 29}, {"Syakban", "Ahad", "26-Jun-2044", 29}, {"Ramadan", "Senin", "25-Jul-2044", 30}, {"Syawal", "Rabu", "24-Aug-2044", 29}, {"Zulkaidah", "Kamis", "22-Sep-2044", 30}, {"Zulhijah", "Sabtu", "22-Oct-2044", 30}],

1467: [{"Muharam", "Senin", "21-Nov-2044", 29}, {"Safar", "Selasa", "20-Dec-2044", 30}, {"Rabiulawal", "Kamis", "19-Jan-2045", 30}, {"Rabiulakhir", "Sabtu", "18-Feb-2045", 30}, {"Jumadilawal",

"Senin", "20-Mar-2045", 29], ["Jumadilakhir", "Selasa", "18-Apr-2045", 30], ["Rajab", "Kamis", "18-May-2045", 29], ["Syakban", "Jumat", "16-Jun-2045", 29], ["Ramadan", "Sabtu", "15-Jul-2045", 29], ["Syawal", "Ahad", "13-Aug-2045", 30], ["Zulkaidah", "Selasa", "12-Sep-2045", 29], ["Zulhijah", "Rabu", "11-Oct-2045", 30]],

1468: [{"Muharam", "Jumat", "10-Nov-2045", 30}, {"Safar", "Ahad", "10-Dec-2045", 29}, {"Rabiulawal", "Senin", "08-Jan-2046", 30}, {"Rabiulakhir", "Rabu", "07-Feb-2046", 30}, {"Jumadilawal", "Jumat", "09-Mar-2046", 29}, {"Jumadilakhir", "Sabtu", "07-Apr-2046", 30}, {"Rajab", "Senin", "07-May-2046", 29}, {"Syakban", "Selasa", "05-Jun-2046", 30}, {"Ramadan", "Kamis", "05-Jul-2046", 29}, {"Syawal", "Jumat", "03-Aug-2046", 30}, {"Zulkaidah", "Ahad", "02-Sep-2046", 29}, {"Zulhijah", "Senin", "01-Oct-2046", 29}],

1469: [{"Muharam", "Selasa", "30-Oct-2046", 30}, {"Safar", "Kamis", "29-Nov-2046", 29}, {"Rabiulawal", "Sabtu", "29-Dec-2046", 30}, {"Rabiulakhir", "Ahad", "27-Jan-2047", 29}, {"Jumadilawal", "Selasa", "26-Feb-2047", 30}, {"Jumadilakhir", "Kamis", "28-Mar-2047", 29}, {"Rajab", "Jumat", "26-Apr-2047", 30}, {"Syakban", "Ahad", "26-May-2047", 29}, {"Ramadan", "Selasa", "25-Jun-2047", 30}, {"Syawal", "Rabu", "24-Jul-2047", 29}, {"Zulkaidah", "Kamis", "22-Aug-2047", 30}, {"Zulhijah", "Sabtu", "21-Sep-2047", 29}],

1470: [{"Muharam", "Ahad", "20-Oct-2047", 30}, {"Safar", "Selasa", "19-Nov-2047", 29}, {"Rabiulawal", "Rabu", "18-Dec-2047", 29}, {"Rabiulakhir", "Kamis", "16-Jan-2048", 30}, {"Jumadilawal", "Sabtu", "15-Feb-2048", 29}, {"Jumadilakhir", "Ahad", "15-Mar-2048", 30}, {"Rajab", "Selasa", "14-Apr-2048", 30}, {"Syakban", "Kamis", "14-May-2048", 30}, {"Ramadan", "Sabtu", "13-Jun-2048", 29}, {"Syawal", "Ahad", "12-Jul-2048", 30}, {"Zulkaidah", "Selasa", "11-Aug-2048", 29}, {"Zulhijah", "Rabu", "09-Sep-2048", 30}],

1471: [{"Muharam", "Jumat", "09-Oct-2048", 29}, {"Safar", "Sabtu", "07-Nov-2048", 30}, {"Rabiulawal", "Senin", "07-Dec-2048", 29}, {"Rabiulakhir", "Selasa", "05-Jan-2049", 29}, {"Jumadilawal", "Rabu", "03-Feb-2049", 30}, {"Jumadilakhir", "Jumat", "05-Mar-2049", 29}, {"Rajab", "Sabtu", "03-Apr-2049", 30}, {"Syakban", "Senin", "03-May-2049", 30}, {"Ramadan", "Rabu", "02-Jun-2049", 29}, {"Syawal", "Kamis", "01-Jul-2049", 30}, {"Zulkaidah", "Sabtu", "31-Jul-2049", 30}, {"Zulhijah", "Senin", "30-Aug-2049", 29}],

1472: [{"Muharam", "Selasa", "28-Sep-2049", 30}, {"Safar", "Kamis", "28-Oct-2049", 29}, {"Rabiulawal", "Jumat", "26-Nov-2049", 30}, {"Rabiulakhir", "Ahad", "26-Dec-2049", 29}, {"Jumadilawal", "Senin", "24-Jan-2050", 29}, {"Jumadilakhir", "Selasa", "22-Feb-2050", 30}, {"Rajab", "Kamis", "24-Mar-2050", 29}, {"Syakban", "Jumat", "22-Apr-2050", 30}, {"Ramadan", "Ahad", "22-May-2050", 29}, {"Syawal", "Senin", "20-Jun-2050", 30}, {"Zulkaidah", "Rabu", "20-Jul-2050", 30}, {"Zulhijah", "Jumat", "19-Aug-2050", 29}],

1473: [{"Muharam", "Sabtu", "17-Sep-2050", 30}, {"Safar", "Senin", "17-Oct-2050", 30}, {"Rabiulawal", "Rabu", "16-Nov-2050", 29}, {"Rabiulakhir", "Kamis", "15-Dec-2050", 30}, {"Jumadilawal", "Sabtu", "14-Jan-2051", 29}, {"Jumadilakhir", "Ahad", "12-Feb-2051", 30}, {"Rajab", "Selasa", "14-Mar-2051", 29}, {"Syakban", "Rabu", "12-Apr-2051", 29}, {"Ramadan", "Kamis", "11-May-2051", 30}, {"Syawal", "Sabtu", "10-Jun-2051", 29}, {"Zulkaidah", "Ahad", "09-Jul-2051", 30}, {"Zulhijah", "Selasa", "08-Aug-2051", 29}],

1474: [{"Muharam", "Rabu", "06-Sep-2051", 30}, {"Safar", "Jumat", "06-Oct-2051", 30}, {"Rabiulawal", "Ahad", "05-Nov-2051", 29}, {"Rabiulakhir", "Senin", "04-Dec-2051", 30}, {"Jumadilawal", "Rabu", "03-Jan-2052", 30}, {"Jumadilakhir", "Jumat", "02-Feb-2052", 29}, {"Rajab", "Sabtu", "02-Mar-2052", 30}, {"Syakban", "Senin", "01-Apr-2052", 29}, {"Ramadan", "Selasa", "30-Apr-2052", 29}, {"Syawal", "Rabu", "29-May-2052", 30}, {"Zulkaidah", "Jumat", "28-Jun-2052", 29}, {"Zulhijah", "Sabtu", "27-Jul-2052", 30}],

1475: [{"Muharam", "Senin", "26-Aug-2052", 29}, {"Safar", "Selasa", "24-Sep-2052", 30}, {"Rabiulawal", "Kamis", "24-Oct-2052", 29}, {"Rabiulakhir", "Jumat", "22-Nov-2052", 30}, {"Jumadilawal", "Ahad", "22-Dec-2052", 30}, {"Jumadilakhir", "Selasa", "21-Jan-2053", 29}, {"Rajab", "Kamis", "20-Feb-2053", 29}, {"Syakban", "Jumat", "21-Mar-2053", 30}, {"Ramadan", "Ahad", "20-Apr-2053", 29}, {"Syawal", "Senin", "19-May-2053", 29}, {"Zulkaidah", "Selasa", "17-Jun-2053", 30}, {"Zulhijah", "Kamis", "17-Jul-2053", 29}],

1476: [{"Muharam", "Jumat", "15-Aug-2053", 29}, {"Safar", "Sabtu", "13-Sep-2053", 30}, {"Rabiulawal", "Senin", "13-Oct-2053", 29}, {"Rabiulakhir", "Selasa", "11-Nov-2053", 30}, {"Jumadilawal", "Kamis", "11-Dec-2053", 30}, {"Jumadilakhir", "Sabtu", "10-Jan-2054", 30}, {"Rajab", "Senin", "09-Feb-2054", 29}, {"Syakban", "Rabu", "11-Mar-2054", 29}, {"Ramadan", "Kamis", "09-Apr-2054", 30}, {"Syawal", "Sabtu", "09-May-2054", 29}, {"Zulkaidah", "Ahad", "07-Jun-2054", 29}, {"Zulhijah", "Senin", "06-Jul-2054", 30}],

1477: [{"Muharam", "Rabu", "05-Aug-2054", 29}, {"Safar", "Kamis", "03-Sep-2054", 29}, {"Rabiulawal", "Jumat", "02-Oct-2054", 30}, {"Rabiulakhir", "Ahad", "01-Nov-2054", 29}, {"Jumadilawal", "Senin", "30-Nov-2054", 30}, {"Jumadilakhir", "Rabu", "30-Dec-2054", 30}, {"Rajab", "Jumat", "29-Jan-2055", 30}, {"Syakban", "Ahad", "28-Feb-2055", 29}, {"Ramadan", "Senin", "29-Mar-2055", 30}, {"Syawal", "Rabu", "28-Apr-2055", 30}, {"Zulkaidah", "Jumat", "28-May-2055", 29}, {"Zulhijah", "Sabtu", "26-Jun-2055", 29}],

1478: [{"Muharam", "Ahad", "25-Jul-2055", 30}, {"Safar", "Selasa", "24-Aug-2055", 29}, {"Rabiulawal", "Rabu", "22-Sep-2055", 29}, {"Rabiulakhir", "Kamis", "21-Oct-2055", 30}, {"Jumadilawal", "Sabtu", "20-Nov-2055", 29}, {"Jumadilakhir", "Ahad", "19-Dec-2055", 30}, {"Rajab", "Selasa", "18-Jan-2056", 30}, {"Syakban", "Kamis", "17-Feb-2056", 29}, {"Ramadan", "Jumat", "17-Mar-2056", 30}, {"Syawal", "Ahad", "16-Apr-2056", 30}, {"Zulkaidah", "Selasa", "16-May-2056", 29}, {"Zulhijah", "Rabu", "14-Jun-2056", 30}],

1479: [{"Muharam", "Jumat", "14-Jul-2056", 29}, {"Safar", "Sabtu", "12-Aug-2056", 30}, {"Rabiulawal", "Senin", "11-Sep-2056", 29}, {"Rabiulakhir", "Selasa", "10-Oct-2056", 30}, {"Jumadilawal", "Rabu", "08-Nov-2056", 30}, {"Jumadilakhir", "Jumat", "08-Dec-2056", 29}, {"Rajab", "Sabtu", "06-Jan-2057", 30}, {"Syakban", "Senin", "05-Feb-2057", 29}, {"Ramadan", "Selasa", "06-Mar-2057", 30}, {"Syawal", "Kamis", "05-Apr-2057", 30}, {"Zulkaidah", "Sabtu", "05-May-2057", 29}, {"Zulhijah", "Ahad", "03-Jun-2057", 30}],

1480: [{"Muharam", "Selasa", "03-Jul-2057", 30}, {"Safar", "Kamis", "02-Aug-2057", 29}, {"Rabiulawal", "Jumat", "31-Aug-2057", 30}, {"Rabiulakhir", "Sabtu", "29-Sep-2057", 30}, {"Jumadilawal", "Senin", "29-Oct-2057", 30}, {"Jumadilakhir", "Rabu", "28-Nov-2057", 29}, {"Rajab", "Kamis", "27-Dec-2057", 29}, {"Syakban", "Jumat", "25-Jan-2058", 30}, {"Ramadan", "Ahad", "24-Feb-2058", 29}, {"Syawal", "Senin", "25-Mar-2058", 30}, {"Zulkaidah", "Rabu", "24-Apr-2058", 29}, {"Zulhijah", "Kamis", "23-May-2058", 30}],

1481: [{"Muharam", "Sabtu", "22-Jun-2058", 29}, {"Safar", "Ahad", "21-Jul-2058", 30}, {"Rabiulawal", "Selasa", "20-Aug-2058", 30}, {"Rabiulakhir", "Kamis", "19-Sep-2058", 30}, {"Jumadilawal", "Sabtu", "19-Oct-2058", 29}, {"Jumadilakhir", "Ahad", "17-Nov-2058", 30}, {"Rajab", "Selasa", "17-Dec-2058", 29}, {"Syakban", "Rabu", "15-Jan-2059", 29}, {"Ramadan", "Kamis", "13-Feb-2059", 30}, {"Syawal", "Sabtu", "15-Mar-2059", 29}, {"Zulkaidah", "Ahad", "13-Apr-2059", 30}, {"Zulhijah", "Selasa", "13-May-2059", 29}],

1482: [{"Muharam", "Rabu", "11-Jun-2059", 30}, {"Safar", "Jumat", "11-Jul-2059", 29}, {"Rabiulawal", "Sabtu", "09-Aug-2059", 30}, {"Rabiulakhir", "Senin", "08-Sep-2059", 30}, {"Jumadilawal", "Rabu", "08-Oct-2059", 30}, {"Jumadilakhir", "Jumat", "07-Nov-2059", 29}, {"Rajab", "Sabtu", "06-Dec-2059", 30}, {"Syakban", "Senin", "05-Jan-2060", 29}, {"Ramadan", "Selasa", "03-Feb-2060", 30}, {"Syawal", "Kamis", "04-Mar-2060", 29}, {"Zulkaidah", "Jumat", "02-Apr-2060", 29}, {"Zulhijah", "Sabtu", "01-May-2060", 30}],

1483: [{"Muharam", "Senin", "31-May-2060", 29}, {"Safar", "Selasa", "29-Jun-2060", 29}, {"Rabiulawal", "Rabu", "28-Jul-2060", 30}, {"Rabiulakhir", "Jumat", "27-Aug-2060", 30}, {"Jumadilawal", "Ahad", "26-Sep-2060", 30}, {"Jumadilakhir", "Selasa", "26-Oct-2060", 29}, {"Rajab", "Rabu", "24-Nov-2060", 30}, {"Syakban", "Jumat", "24-Dec-2060", 30}, {"Ramadan", "Ahad", "23-Jan-2061", 29}, {"Syawal", "Senin", "21-Feb-2061", 30}, {"Zulkaidah", "Rabu", "23-Mar-2061", 29}, {"Zulhijah", "Kamis", "21-Apr-2061", 29}],

1484: [{"Muharam", "Jumat", "20-May-2061", 29}, {"Safar", "Sabtu", "18-Jun-2061", 30}, {"Rabiulawal", "Senin", "18-Jul-2061", 29}, {"Rabiulakhir", "Selasa", "16-Aug-2061", 30}, {"Jumadilawal", "Kamis", "15-Sep-2061", 30}, {"Jumadilakhir", "Sabtu", "15-Oct-2061", 29}, {"Rajab", "Ahad", "13-Nov-2061", 30}, {"Syakban", "Selasa", "13-Dec-2061", 30}, {"Ramadan", "Kamis", "12-Jan-2062", 30}],

["Syawal", "Sabtu", "11-Feb-2062", 29], ["Zulkaidah", "Ahad", "12-Mar-2062", 30], ["Zulhijah", "Selasa", "11-Apr-2062", 29]],

1485: [{"Muharam", "Rabu", "10-May-2062", 29}, {"Safar", "Kamis", "08-Jun-2062", 29}, {"Rabiulawal", "Jumat", "07-Jul-2062", 30}, {"Rabiulakhir", "Ahad", "06-Aug-2062", 29}, {"Jumadilawal", "Senin", "04-Sep-2062", 30}, {"Jumadilakhir", "Rabu", "04-Oct-2062", 29}, {"Rajab", "Kamis", "02-Nov-2062", 30}, {"Syakban", "Sabtu", "02-Dec-2062", 30}, {"Ramadan", "Senin", "01-Jan-2063", 30}, {"Syawal", "Rabu", "31-Jan-2063", 29}, {"Zulkaidah", "Kamis", "01-Mar-2063", 30}, {"Zulhijah", "Sabtu", "31-Mar-2063", 29}],

1486: [{"Muharam", "Ahad", "29-Apr-2063", 30}, {"Safar", "Selasa", "29-May-2063", 29}, {"Rabiulawal", "Rabu", "27-Jun-2063", 30}, {"Rabiulakhir", "Jumat", "27-Jul-2063", 29}, {"Jumadilawal", "Sabtu", "25-Aug-2063", 29}, {"Jumadilakhir", "Ahad", "23-Sep-2063", 30}, {"Rajab", "Selasa", "23-Oct-2063", 29}, {"Syakban", "Rabu", "21-Nov-2063", 30}, {"Ramadan", "Jumat", "21-Dec-2063", 30}, {"Syawal", "Ahad", "20-Jan-2064", 29}, {"Zulkaidah", "Senin", "18-Feb-2064", 30}, {"Zulhijah", "Rabu", "19-Mar-2064", 30}],

1487: [{"Muharam", "Jumat", "18-Apr-2064", 29}, {"Safar", "Sabtu", "17-May-2064", 30}, {"Rabiulawal", "Senin", "16-Jun-2064", 29}, {"Rabiulakhir", "Selasa", "15-Jul-2064", 30}, {"Jumadilawal", "Kamis", "14-Aug-2064", 29}, {"Jumadilakhir", "Jumat", "12-Sep-2064", 29}, {"Rajab", "Sabtu", "11-Oct-2064", 30}, {"Syakban", "Senin", "10-Nov-2064", 29}, {"Ramadan", "Selasa", "09-Dec-2064", 30}, {"Syawal", "Kamis", "08-Jan-2065", 29}, {"Zulkaidah", "Jumat", "06-Feb-2065", 30}, {"Zulhijah", "Ahad", "08-Mar-2065", 30}],

1488: [{"Muharam", "Selasa", "07-Apr-2065", 29}, {"Safar", "Rabu", "06-May-2065", 30}, {"Rabiulawal", "Jumat", "05-Jun-2065", 30}, {"Rabiulakhir", "Ahad", "05-Jul-2065", 29}, {"Jumadilawal", "Senin", "03-Aug-2065", 30}, {"Jumadilakhir", "Rabu", "02-Sep-2065", 29}, {"Rajab", "Kamis", "01-Oct-2065", 29}, {"Syakban", "Jumat", "30-Oct-2065", 30}, {"Ramadan", "Ahad", "29-Nov-2065", 29}, {"Syawal", "Senin", "28-Dec-2065", 30}, {"Zulkaidah", "Rabu", "27-Jan-2066", 29}, {"Zulhijah", "Kamis", "25-Feb-2066", 30}],

1489: [{"Muharam", "Sabtu", "27-Mar-2066", 29}, {"Safar", "Ahad", "25-Apr-2066", 30}, {"Rabiulawal", "Selasa", "25-May-2066", 30}, {"Rabiulakhir", "Kamis", "24-Jun-2066", 30}, {"Jumadilawal", "Sabtu", "24-Jul-2066", 29}, {"Jumadilakhir", "Ahad", "22-Aug-2066", 30}, {"Rajab", "Selasa", "21-Sep-2066", 29}, {"Syakban", "Rabu", "20-Oct-2066", 30}, {"Ramadan", "Jumat", "19-Nov-2066", 29}, {"Syawal", "Sabtu", "18-Dec-2066", 29}, {"Zulkaidah", "Ahad", "16-Jan-2067", 30}, {"Zulhijah", "Selasa", "15-Feb-2067", 29}],

1490: [{"Muharam", "Rabu", "16-Mar-2067", 30}, {"Safar", "Jumat", "15-Apr-2067", 29}, {"Rabiulawal", "Sabtu", "14-May-2067", 30}, {"Rabiulakhir", "Senin", "13-Jun-2067", 30}, {"Jumadilawal", "Rabu", "13-Jul-2067", 29}, {"Jumadilakhir", "Kamis", "11-Aug-2067", 30}, {"Rajab", "Sabtu", "10-Sep-2067", 30}, {"Syakban", "Ahad", "10-Oct-2067", 29}, {"Ramadan", "Selasa", "08-Nov-2067", 30}, {"Syawal", "Kamis", "08-Dec-2067", 29}, {"Zulkaidah", "Jumat", "06-Jan-2068", 29}, {"Zulhijah", "Sabtu", "04-Feb-2068", 30}],

1491: [{"Muharam", "Senin", "05-Mar-2068", 29}, {"Safar", "Selasa", "03-Apr-2068", 30}, {"Rabiulawal", "Kamis", "03-May-2068", 29}, {"Rabiulakhir", "Jumat", "01-Jun-2068", 30}, {"Jumadilawal", "Ahad", "01-Jul-2068", 29}, {"Jumadilakhir", "Senin", "30-Jul-2068", 30}, {"Rajab", "Rabu", "29-Aug-2068", 30}, {"Syakban", "Jumat", "28-Sep-2068", 29}, {"Ramadan", "Sabtu", "27-Oct-2068", 30}, {"Syawal", "Senin", "26-Nov-2068", 30}, {"Zulkaidah", "Rabu", "26-Dec-2068", 29}, {"Zulhijah", "Kamis", "24-Jan-2069", 29}],

1492: [{"Muharam", "Jumat", "22-Feb-2069", 30}, {"Safar", "Ahad", "24-Mar-2069", 29}, {"Rabiulawal", "Senin", "22-Apr-2069", 30}, {"Rabiulakhir", "Rabu", "22-May-2069", 29}, {"Jumadilawal", "Kamis", "20-Jun-2069", 29}, {"Jumadilakhir", "Jumat", "19-Jul-2069", 30}, {"Rajab", "Ahad", "18-Aug-2069", 30}, {"Syakban", "Selasa", "17-Sep-2069", 29}, {"Ramadan", "Rabu", "16-Oct-2069", 30}, {"Syawal", "Jumat", "15-Nov-2069", 30}, {"Zulkaidah", "Ahad", "15-Dec-2069", 29}, {"Zulhijah", "Senin", "13-Jan-2070", 30}],

1493: [{"Muharam", "Rabu", "12-Feb-2070", 29}, {"Safar", "Jumat", "14-Mar-2070", 29}, {"Rabiulawal", "Sabtu", "12-Apr-2070", 29}, {"Rabiulakhir", "Ahad", "11-May-2070", 30}, {"Jumadilawal", "Selasa", "10-Jun-2070", 29}, {"Jumadilakhir", "Rabu", "09-Jul-2070", 29}, {"Rajab", "Kamis", "07-Aug-2070", 30}, {"Syakban", "Sabtu", "06-Sep-2070", 29}, {"Ramadan", "Ahad", "05-Oct-2070", 30}, {"Syawal", "Selasa", "04-Nov-2070", 30}, {"Zulkaidah", "Kamis", "04-Dec-2070", 29}, {"Zulhijah", "Jumat", "02-Jan-2071", 30}],

1494: [{"Muharam", "Ahad", "01-Feb-2071", 30}, {"Safar", "Selasa", "03-Mar-2071", 30}, {"Rabiulawal", "Kamis", "02-Apr-2071", 29}, {"Rabiulakhir", "Jumat", "01-May-2071", 29}, {"Jumadilawal", "Sabtu", "30-May-2071", 30}, {"Jumadilakhir", "Senin", "29-Jun-2071", 29}, {"Rajab", "Selasa", "28-Jul-2071", 29}, {"Syakban", "Rabu", "26-Aug-2071", 30}, {"Ramadan", "Jumat", "25-Sep-2071", 29}, {"Syawal", "Sabtu", "24-Oct-2071", 30}, {"Zulkaidah", "Senin", "23-Nov-2071", 29}, {"Zulhijah", "Selasa", "22-Dec-2071", 30}],

1495: [{"Muharam", "Kamis", "21-Jan-2072", 30}, {"Safar", "Sabtu", "20-Feb-2072", 30}, {"Rabiulawal", "Senin", "21-Mar-2072", 29}, {"Rabiulakhir", "Selasa", "19-Apr-2072", 30}, {"Jumadilawal", "Kamis", "19-May-2072", 29}, {"Jumadilakhir", "Jumat", "17-Jun-2072", 30}, {"Rajab", "Ahad", "17-Jul-2072", 29}, {"Syakban", "Senin", "15-Aug-2072", 29}, {"Ramadan", "Selasa", "13-Sep-2072", 30}, {"Syawal", "Kamis", "13-Oct-2072", 29}, {"Zulkaidah", "Jumat", "11-Nov-2072", 30}, {"Zulhijah", "Ahad", "11-Dec-2072", 29}],

1496: [{"Muharam", "Senin", "09-Jan-2073", 30}, {"Safar", "Rabu", "08-Feb-2073", 30}, {"Rabiulawal", "Jumat", "10-Mar-2073", 30}, {"Rabiulakhir", "Ahad", "09-Apr-2073", 29}, {"Jumadilawal", "Senin", "08-May-2073", 30}, {"Jumadilakhir", "Rabu", "07-Jun-2073", 29}, {"Rajab", "Kamis", "06-Jul-2073", 30}, {"Syakban", "Sabtu", "05-Aug-2073", 29}, {"Ramadan", "Ahad", "03-Sep-2073", 29}, {"Syawal", "Senin", "02-Oct-2073", 30}, {"Zulkaidah", "Rabu", "01-Nov-2073", 29}, {"Zulhijah", "Kamis", "30-Nov-2073", 30}],

1497: [{"Muharam", "Sabtu", "30-Dec-2073", 29}, {"Safar", "Ahad", "28-Jan-2074", 30}, {"Rabiulawal", "Selasa", "27-Feb-2074", 30}, {"Rabiulakhir", "Kamis", "29-Mar-2074", 29}, {"Jumadilawal", "Jumat", "27-Apr-2074", 30}, {"Jumadilakhir", "Ahad", "27-May-2074", 30}, {"Rajab", "Selasa", "26-Jun-2074", 29}, {"Syakban", "Rabu", "25-Jul-2074", 30}, {"Ramadan", "Jumat", "24-Aug-2074", 29}, {"Syawal", "Sabtu", "22-Sep-2074", 29}, {"Zulkaidah", "Ahad", "21-Oct-2074", 30}, {"Zulhijah", "Selasa", "20-Nov-2074", 30}],

1498: [
 [
 "Muharam",
 "Selasa",
 "19-Dec-2074",
 30
],
 [
 "Safar",
 "Kamis",
 "18-Jan-2075",
 29
],
 [
 "Rabiulawal",
 "Jumat",
 "16-Feb-2075",
 30
],
 [
 "Rabiulakhir",
 "Ahad",

```

    "18-Mar-2075",
    29
  ],
  [
    "Jumadilawal",
    "Senin",
    "16-Apr-2075",
    30
  ],
  [
    "Jumadilakhir",
    "Rabu",
    "16-May-2075",
    29
  ],
  [
    "Rajab",
    "Kamis",
    "14-Jun-2075",
    30
  ],
  [
    "Syakban",
    "Sabtu",
    "14-Jul-2075",
    30
  ],
  [
    "Ramadan",
    "Senin",
    "13-Aug-2075",
    29
  ],
  [
    "Syawal",
    "Selasa",
    "11-Sep-2075",
    30
  ],
  [
    "Zulkaidah",
    "Kamis",
    "11-Oct-2075",
    29
  ],
  [
    "Zulhijah",
    "Jumat",

```

```

        "09-Nov-2075",
        30
    ],
    "1499": [
        [
            "Muharam",
            "Ahad",
            "09-Dec-2075",
            29
        ],
        [
            "Safar",
            "Senin",
            "07-Jan-2076",
            30
        ],
        [
            "Rabiulawal",
            "Rabu",
            "06-Feb-2076",
            29
        ],
        [
            "Rabiulakhir",
            "Kamis",
            "06-Mar-2076",
            30
        ],
        [
            "Jumadilawal",
            "Sabtu",
            "05-Apr-2076",
            29
        ],
        [
            "Jumadilakhir",
            "Ahad",
            "04-May-2076",
            30
        ],
        [
            "Rajab",
            "Selasa",
            "03-Jun-2076",
            29
        ],
        [

```



```

        "Syakban",
        "Rabu",
        "02-Jul-2076",
        30
    ],
    [
        "Ramadan",
        "Jumat",
        "01-Aug-2076",
        29
    ],
    [
        "Syawal",
        "Sabtu",
        "30-Aug-2076",
        30
    ],
    [
        "Zulkaidah",
        "Senin",
        "29-Sep-2076",
        30
    ],
    [
        "Zulhijah",
        "Rabu",
        "29-Oct-2076",
        29
    ]
],
"1500": [
    [
        "Muharam",
        "Kamis",
        "27-Nov-2076",
        30
    ],
    [
        "Safar",
        "Sabtu",
        "27-Dec-2076",
        29
    ],
    [
        "Rabiulawal",
        "Ahad",
        "25-Jan-2077",
        30
    ]
]

```

```

],
[
  "Rabiulakhir",
  "Selasa",
  "24-Feb-2077",
  29
],
[
  "Jumadilawal",
  "Rabu",
  "25-Mar-2077",
  30
],
[
  "Jumadilakhir",
  "Jumat",
  "24-Apr-2077",
  29
],
[
  "Rajab",
  "Sabtu",
  "23-May-2077",
  29
],
[
  "Syakban",
  "Ahad",
  "21-Jun-2077",
  30
],
[
  "Ramadan",
  "Selasa",
  "21-Jul-2077",
  29
],
[
  "Syawal",
  "Rabu",
  "19-Aug-2077",
  30
],
[
  "Zulkaidah",
  "Jumat",
  "18-Sep-2077",
  30
]

```

```

    ],
    [
        "Zulhijah",
        "Ahad",
        "18-Oct-2077",
        30
    ]
]
})

# ---> TAMBAHAN: AUTO-LOAD DARI DATABASE JSON <---
import json
try:
    if getattr(sys, 'frozen', False):
        DB_JSON_PATH = os.path.join(os.path.dirname(sys.executable), "db_hijriah.json")
    else:
        DB_JSON_PATH = os.path.join(os.path.dirname(os.path.abspath(__file__)), "db_hijriah.json")

    if os.path.exists(DB_JSON_PATH):
        with open(DB_JSON_PATH, "r", encoding="utf-8") as f:
            external_db = json.load(f)

            # Konversi key JSON dari string menjadi integer
            external_db = {int(k): v for k, v in external_db.items()}

            # Jangan menimpa data bawaan yang sudah tervalidasi
            for year, months in external_db.items():
                if year not in HIJRI_DB:
                    HIJRI_DB[year] = months
except Exception as e:
    print(f"Peringatan: Gagal memuat db_hijriah.json -> {e}")
# -----

class HijriConverter:
    @staticmethod
    def parse_date(date_str):
        try:
            m = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10,
                "Nov":11, "Dec":12}
            parts = date_str.split('-')
            return datetime.date(int(parts[2]), m.get(parts[1], 1), int(parts[0]))
        except: return None
    @staticmethod
    def get_hijri_date(today):
        for year, months in HIJRI_DB.items():
            for m_data in months:
                start = HijriConverter.parse_date(m_data[2])

```

```

        if start and start <= today < start + datetime.timedelta(days=m_data[3]):
            return f"{{(today - start).days + 1}} {{m_data[0]}} {{year}} H"
        return "N/A"

```

```

# =====
# DATABASE KOTA (Diambil dari master Anda)
# =====
CITY_DB = {
    "Aceh": {"Banda Aceh": (5.5483, 95.3238), "Lhokseumawe": (5.1801, 97.1507), "Langsa": (4.4725,
97.9658), "Meulaboh": (4.1438, 96.1265), "Sabang": (5.8942, 95.3184), "Subulussalam": (2.6312,
98.0012), "Takengon": (4.6212, 96.8412), "Kutacane": (3.4812, 97.8112), "Singkil": (2.2851, 97.7942),
"Calang": (4.6312, 95.5812), "Blangpidie": (3.7412, 96.8412), "Karang Baru": (4.3112, 98.0512),
"Blangkejeren": (3.9912, 97.3312), "Sigli": (5.3812, 95.9612), "Meureudu": (5.2412, 96.2512), "Bireuen":
(5.2031, 96.7011), "Simpang Tiga Redelong": (4.7212, 96.8512), "Suka Makmue": (4.1512, 96.3312),
"Sinabang": (2.4812, 96.3712)},
    "Bali": {"Denpasar": (-8.6705, 115.2126), "Singaraja": (-8.1120, 115.0882), "Mangupura": (-8.5812,
115.1712), "Gianyar": (-8.5412, 115.3212), "Tabanan": (-8.5312, 115.1212), "Semarapura": (-8.5312,
115.4012), "Amlapura": (-8.4412, 115.6112), "Bangli": (-8.4512, 115.3512), "Negara": (-8.3512,
114.6212)},
    "Bangka Belitung": {"Pangkal Pinang": (-2.1300, 106.1100), "Sungailiat": (-1.8512, 106.1112), "Tanjung
Pandan": (-2.7312, 107.6312)},
    "Banten": {"Serang": (-6.1104, 106.1601), "Cilegon": (-6.0234, 106.0152), "Tangerang": (-6.1702,
106.6403), "Tangerang Selatan": (-6.2886, 106.7179), "Tigaraksa": (-6.2712, 106.4712), "Pandeglang": (-
6.3112, 106.1012), "Rangkasbitung": (-6.3512, 106.2412)},
    "Bengkulu": {"Bengkulu": (-3.8004, 102.2655), "Curup": (-3.4612, 102.5212), "Manna": (-4.4712,
102.9012), "Argamakmur": (-3.4212, 102.1912), "Mukomuko": (-2.5812, 101.1212)},
    "DI Yogyakarta": {"Yogyakarta": (-7.7956, 110.3695), "Sleman": (-7.7212, 110.3644), "Bantul": (-
7.8922, 110.3322), "Wates": (-7.8512, 110.1512), "Wonosari": (-7.9612, 110.6012)},
    "DKI Jakarta": {"Jakarta Pusat": (-6.1865, 106.8270), "Jakarta Selatan": (-6.2615, 106.8106), "Jakarta
Barat": (-6.1674, 106.7637), "Jakarta Timur": (-6.2250, 106.9004), "Jakarta Utara": (-6.1214, 106.8745),
"Kepulauan Seribu": (-5.7412, 106.6112)},
    "Gorontalo": {"Gorontalo": (0.5435, 123.0568), "Limboto": (0.6212, 122.9812), "Marisa": (0.4512,
121.9412)},
    "Jambi": {"Jambi": (-1.6099, 103.6057), "Sungai Penuh": (-2.0722, 101.3789), "Muara Bulian": (-
1.7212, 103.2712), "Bangko": (-2.0712, 102.2612), "Muara Bungo": (-1.4812, 102.1212)},
    "Jawa Barat": {"Bandung": (-6.9175, 107.6191), "Bekasi": (-6.2383, 106.9756), "Depok": (-6.4025,
106.7942), "Bogor": (-6.5971, 106.7986), "Cimahi": (-6.8722, 107.5422), "Tasikmalaya": (-7.3274,
108.2207), "Garut": (-7.2272, 107.9086), "Ciamis": (-7.3211, 108.3512), "Banjar": (-7.3676, 108.5364),
"Cirebon": (-6.7063, 108.5570), "Indramayu": (-6.3271, 108.3201), "Majalengka": (-6.8312, 108.2212),
"Kuningan": (-6.9712, 108.4812), "Sukabumi": (-6.9242, 106.9350), "Cianjur": (-6.8201, 107.1394),
"Karawang": (-6.3012, 107.3011), "Subang": (-6.5712, 107.7612), "Purwakarta": (-6.5512, 107.4411),
"Sumedang": (-6.8412, 107.9211)},
    "Jawa Tengah": {"Semarang": (-7.0667, 110.4100), "Demak": (-6.8906, 110.6389), "Kudus": (-6.8048,
110.8400), "Pati": (-6.7509, 111.0384), "Rembang": (-6.7058, 111.3414), "Jepara": (-6.5888, 110.6675),
"Blora": (-6.9697, 111.4184), "Grobogan": (-7.0264, 110.9187), "Surakarta": (-7.5703, 110.8297),
"Sukoharjo": (-7.6833, 110.8333), "Wonogiri": (-7.8122, 110.9253), "Karanganyar": (-7.5961, 110.9511),
"Sragen": (-7.4267, 111.0211), "Boyolali": (-7.5317, 110.5961), "Klaten": (-7.7031, 110.6025),
"Magelang": (-7.4706, 110.2178), "Temanggung": (-7.3167, 110.1667), "Wonosobo": (-7.3597,

```

109.9111), "Purworejo": (-7.7122, 110.0078), "Kebumen": (-7.6697, 109.6522), "Purwokerto": (-7.4244, 109.2300), "Cilacap": (-7.7277, 109.0159), "Purbalingga": (-7.3875, 109.3675), "Banjarnegara": (-7.3961, 109.6967), "Pekalongan": (-6.8886, 109.6753), "Batang": (-6.9111, 109.7289), "Pemalang": (-6.8919, 109.3814), "Tegal": (-6.8676, 109.1371), "Brebes": (-6.8722, 109.0436), "Kendal": (-6.9189, 110.2039), "Salatiga": (-7.3305, 110.5084)},

"Jawa Timur": {"Surabaya": (-7.2575, 112.7521), "Sidoarjo": (-7.4412, 112.7112), "Gresik": (-7.1612, 112.6511), "Malang": (-7.9797, 112.6304), "Batu": (-7.8671, 112.5239), "Kediri": (-7.8172, 112.0116), "Blitar": (-8.0983, 112.1681), "Tulungagung": (-8.0612, 111.9012), "Nganjuk": (-7.6012, 111.9012), "Trenggalek": (-8.0512, 111.7112), "Madiun": (-7.6298, 111.5239), "Ngawi": (-7.4012, 111.4411), "Magetan": (-7.6512, 111.3312), "Ponorogo": (-7.8712, 111.4612), "Pacitan": (-8.2012, 111.0912), "Jember": (-8.1712, 113.7011), "Banyuwangi": (-8.2112, 114.3611), "Bondowoso": (-7.9112, 113.8212), "Situbondo": (-7.7012, 114.0012), "Probolinggo": (-7.7554, 113.2162), "Lumajang": (-8.1312, 113.2212), "Bojonegoro": (-7.1512, 111.8811), "Tuban": (-6.9012, 112.0511), "Lamongan": (-7.1212, 112.4111), "Bangkalan": (-7.0412, 112.7411), "Sampang": (-7.1812, 113.2411), "Pamekasan": (-7.1512, 113.4711), "Sumenep": (-7.0112, 113.8611), "Pasuruan": (-7.6449, 112.9061), "Mojokerto": (-7.4712, 112.4311), "Jombang": (-7.5512, 112.2312)},

"Kalimantan Barat": {"Pontianak": (-0.0263, 109.3425), "Singkawang": (0.9023, 108.9711), "Mempawah": (0.3512, 108.9612), "Sambas": (1.3512, 109.3012), "Sintang": (0.0712, 111.4912), "Ketapang": (-1.8412, 109.9712)},

"Kalimantan Selatan": {"Banjarmasin": (-3.3167, 114.5900), "Banjarbaru": (-3.4431, 114.8286), "Pelaihari": (-3.7912, 114.7712), "Kotabaru": (-3.2412, 116.2212), "Tanjung": (-2.1812, 115.3912)},

"Kalimantan Tengah": {"Palangkaraya": (-2.2107, 113.9213), "Sampit": (-2.5312, 112.9512), "Pangkalan Bun": (-2.6812, 111.6212), "Muara Teweh": (-0.9512, 114.8912), "Kapuas": (-3.0112, 114.3912)},

"Kalimantan Timur": {"Samarinda": (-0.4949, 117.1492), "Balikpapan": (-1.2654, 116.8312), "Bontang": (0.1328, 117.4728), "Tenggarong": (-0.4312, 116.9812), "Sangatta": (0.5012, 117.5512)},

"Kalimantan Utara": {"Tanjung Selor": (2.8333, 117.3667), "Tarakan": (3.3033, 117.5878), "Nunukan": (4.1312, 117.6512), "Malinau": (3.5812, 116.6312)},

"Kepulauan Riau": {"Tanjung Pinang": (0.9167, 104.4500), "Batam": (1.1301, 104.0520), "Karimun": (0.9912, 103.4212), "Natuna": (3.9412, 108.3812)},

"Lampung": {"Bandar Lampung": (-5.4292, 105.2611), "Metro": (-5.1133, 105.3067), "Kalianda": (-5.7412, 105.5912), "Kotabumi": (-4.8212, 104.8912)},

"Maluku": {"Ambon": (-3.6954, 128.1814), "Tual": (-5.6322, 132.7389), "Masohi": (-3.3012, 128.9512), "Langgur": (-5.6512, 132.7112), "Saumlaki": (-7.9512, 131.3012)},

"Maluku Utara": {"Ternate": (0.7901, 127.3821), "Tidore": (0.6457, 127.4422), "Sofifi": (0.7312, 127.5812), "Tobelo": (1.7212, 128.0112)},

"NTB": {"Mataram": (-8.5833, 116.1167), "Praya": (-8.7112, 116.2712), "Selong": (-8.6512, 116.5312), "Bima": (-8.4605, 118.7265), "Sumbawa Besar": (-8.5012, 117.4212), "Dompu": (-8.5312, 118.4612)},

"NTT": {"Kupang": (-10.1772, 123.6070), "Soe": (-9.8612, 124.2812), "Kefamenanu": (-9.4512, 124.4812), "Waingapu": (-9.6512, 120.2612), "Labuan Bajo": (-8.5012, 119.8812), "Ruteng": (-8.6112, 120.4712), "Ende": (-8.8412, 121.6512), "Maukere": (-8.6254, 122.2132), "Larantuka": (-8.3412, 122.9812)},

"Papua": {"Jayapura": (-2.5916, 140.6690), "Biak": (-1.1712, 136.0812), "Serui": (-1.8812, 136.2312)},

"Papua Barat": {"Manokwari": (-0.8614, 134.0620), "Fakfak": (-3.2212, 132.2912), "Kaimana": (-3.6612, 133.7712)},

"Papua Barat Daya": {"Sorong": (-0.8765, 131.2558), "Waisai": (-0.4112, 130.8112)},

"Papua Selatan": {"Merauke": (-8.4912, 140.4012), "Agats": (-5.5412, 138.1312)},

"Papua Tengah": {"Nabire": (-3.3612, 135.5012), "Timika": (-4.5412, 136.8812)},

"Papua Pegunungan": {"Wamena": (-4.0612, 138.9412)},
 "Riau": {"Pekanbaru": (0.5071, 101.4478), "Dumai": (1.6712, 101.4455), "Bangkinang": (0.3312, 101.0312), "Siak": (0.7912, 102.0412)},
 "Sulawesi Barat": {"Mamuju": (-2.6712, 118.8812), "Majene": (-3.5412, 118.9712), "Polewali": (-3.4312, 119.3212)},
 "Sulawesi Selatan": {"Makassar": (-5.1476, 119.4327), "Parepare": (-4.0135, 119.6247), "Palopo": (-2.9926, 120.1916), "Gowa": (-5.2012, 119.4512), "Bone": (-4.5412, 120.3212)},
 "Sulawesi Tengah": {"Palu": (-0.8917, 119.8707), "Luwuk": (-0.9512, 122.7812), "Poso": (-1.3912, 120.7512), "Donggala": (-0.6712, 119.7412)},
 "Sulawesi Tenggara": {"Kendari": (-3.9722, 122.5149), "Baubau": (-5.4612, 122.6112), "Kolaka": (-4.0512, 121.5912)},
 "Sulawesi Utara": {"Manado": (1.4748, 124.8404), "Bitung": (1.4412, 125.1212), "Tomohon": (1.3212, 124.8312), "Kotamobagu": (0.7212, 124.3112)},
 "Sumatera Barat": {"Padang": (-0.9471, 100.4172), "Bukittinggi": (-0.3041, 100.3695), "Payakumbuh": (-0.2241, 100.6358), "Pariaman": (-0.6171, 100.1239), "Solok": (-0.7937, 100.6599)},
 "Sumatera Selatan": {"Palembang": (-2.9761, 104.7754), "Lubuklinggau": (-3.2952, 102.8611), "Prabumulih": (-3.4308, 104.2255), "Lahat": (-3.7812, 103.5412), "Pagar Alam": (-4.0112, 103.2712)},
 "Sumatera Utara": {"Medan": (3.5952, 98.6722), "Binjai": (3.6010, 98.4854), "Pematangsiantar": (2.9620, 99.0664), "Tebing Tinggi": (3.3364, 99.1625), "Sibolga": (1.7392, 98.7776)},

"Amerika Serikat": {
 "New York City": (40.7128, -74.0060),
 "Los Angeles": (34.0522, -118.2437),
 "Chicago": (41.8781, -87.6298),
 "Houston": (29.7604, -95.3698),
 "Phoenix": (33.4484, -112.0740),
 "Seattle": (47.6062, -122.3321),
 "Miami": (25.7617, -80.1918),
 "Washington D.C.": (38.9072, -77.0369),
 "Anchorage": (61.2181, -149.9003),
 "Fairbanks": (64.8378, -147.7164),
 "King Salmon": (58.6833, -156.6667),
 "Port Heiden": (56.937449, -158.611570),
 "Aleutians East Borough-1": (56.807956, -158.944580),
 "Aleutians East Borough-2": (56.013387, -160.413219),
 "Cold Bay-1": (55.2000, -162.7167),
 "Cold Bay-2": (55.194384, -162.728017),
 "Homer": (59.6425, -151.5483),
 "Valdez": (61.1308, -146.3483)
 },
 "Kanada": {
 "Toronto": (43.6510, -79.3470),
 "Vancouver": (49.2827, -123.1207),
 "Montreal": (45.5017, -73.5673)
 },
 "Meksiko": {
 "Mexico City": (19.4326, -99.1332),
 "Monterrey": (25.6866, -100.3161),


```

"Guadalajara": (20.6597, -103.3496)
},
# MENGGABUNGKAN SELURUH AMERIKA SELATAN (Brasil, Argentina, Kolombia, Peru, Chili, dsb)
"Amerika Selatan": {
    "Lima": (-12.0464, -77.0428),
    "Santiago": (-33.4489, -70.6693),
    "Bogota": (4.7110, -74.0721),
    "Quito": (-0.1807, -78.4678),
    "La Paz": (-16.4897, -68.1193),
    "Guayaquil": (-2.1894, -79.8891),
    "Valparaiso": (-33.0472, -71.6127),
    "Arequipa": (-16.4090, -71.5375),
    "Cali": (3.4516, -76.5320),
    "Antofagasta": (-23.6500, -70.4000),
    "Sao Paulo": (-23.5505, -46.6333),
    "Rio de Janeiro": (-22.9068, -43.1729),
    "Buenos Aires": (-34.6037, -58.3816),
    "Montevideo": (-34.9011, -56.1645),
    "Brasilia": (-15.8267, -47.9218),
    "Salvador": (-12.9714, -38.5014),
    "Fortaleza": (-3.7319, -38.5267),
    "Rosario": (-32.9468, -60.6393),
    "Asuncion": (-25.2637, -57.5759),
    "Caracas": (10.4806, -66.9036),
    "Medellin": (6.2442, -75.5812),
    "Cartagena": (10.3910, -75.4794),
    "Cuenca": (-2.9001, -79.0059),
    "Manta": (-0.9621, -80.7127),
    "Trujillo": (-8.1159, -79.0282),
    "Piura": (-5.1945, -80.6328),
    "Cusco": (-13.5226, -71.9673),
    "Arica": (-18.4783, -70.3126),
    "Iquique": (-20.2208, -70.1431),
    "Concepcion": (-36.8201, -73.0444),
    "Temuco": (-38.7359, -72.5904),
    "Manaus": (-3.1190, -60.0217),
    "Cuiaba": (-15.6014, -56.0979),
    "Iquitos": (-3.7491, -73.2538),
    "Santa Cruz de la Sierra": (-17.7833, -63.1821),
    "Cochabamba": (-17.3895, -66.1568),
    "Sucre": (-19.0333, -65.2627),
    "Potosi": (-19.5836, -65.7531),
    "Cordoba": (-31.4201, -64.1888),
    "Mendoza": (-32.8908, -68.8272)
},
"Arab Saudi": {
    "Tumair": (25.7039, 45.8678), "Majmaah": (25.8970, 45.3372), "Shaqra": (25.2524, 45.2536),
    "Makkah": (21.4225, 39.8262), "Madinah": (24.4672, 39.6112), "Riyadh": (24.7136, 46.6753),

```

```

    "Jeddah": (21.4858, 39.1925), "Taif": (21.2703, 40.4158), "Tabuk": (28.3835, 36.5662),
    "Dammam": (26.4207, 50.0888), "Abha": (18.2164, 42.5053), "Jazan": (16.8892, 42.5511),
    "Buraidah": (26.3260, 43.9750), "Hail": (27.5219, 41.6907), "Najran": (17.4933, 44.1277),
    "Al-Baha": (20.0129, 41.4677), "Sakaka": (29.9697, 40.2064), "Arar": (30.9833, 40.0333),
    "Yanbu": (24.0891, 38.0637), "Al Ahsa": (25.3795, 49.5858)
},
# MENGGABUNGKAN KUNCI "Mesir" YANG SEBELUMNYA DITULIS 2 KALI
"Mesir": {
    "Kairo": (30.0444, 31.2357),
    "Iskandariyah / Alexandria": (31.2001, 29.9187),
    "Aswan": (24.0889, 32.8998),
    "Sallum": (31.5658, 25.1481),
    "Abu Simbel": (22.3372, 31.6258),
    "Kharga": (25.4390, 30.5586),
    "Sohag": (26.5591, 31.6957),
    "Faiyum": (29.3084, 30.8428),
    "Qena": (26.1551, 32.7160),
    "Sharm El-Sheikh": (27.9158, 34.3299),
    "Marsa Matruh": (31.3525, 27.2453)
},
# MENGGABUNGKAN KUNCI "Maroko" YANG SEBELUMNYA DITULIS 2 KALI
"Maroko": {
    "Rabat": (34.0209, -6.8416),
    "Casablanca": (33.5731, -7.5898),
    "Marrakech": (31.6295, -7.9811),
    "Fez": (34.0331, -5.0003),
    "Dakhla": (23.6848, -15.9580),
    "Laayoune": (27.1253, -13.1625),
    "Agadir": (30.4278, -9.5981),
    "Tangier": (35.7595, -5.8340),
    "Oujda": (34.6814, -1.9086),
    "Ouarzazate": (30.9189, -6.8934)
},
"Kepulauan Pasifik (Timur Jauh)": {
    "Kiritimati": (1.8709, -157.3962),
    "Apia": (-13.8333, -171.7667),
    "Nuku'alofa": (-21.1393, -175.2018),
    "Suva": (-18.1248, 178.4501)
},
# MENAMBAHKAN GISBORNE (NEW ZEALAND)
"Selandia Baru": {
    "Gisborne": (-38.6623, 178.0176),
    "Waitangi": (-35.2681, 174.0803),
    "Auckland": (-36.8485, 174.7633),
    "Wellington": (-41.2865, 174.7762),
    "Christchurch": (-43.5321, 172.6362),
    "Dunedin": (-45.8788, 170.5028)
},

```

```

"Australia": {
  "Sydney": (-33.8688, 151.2093),
  "Brisbane": (-27.4698, 153.0251),
  "Melbourne": (-37.8136, 144.9631),
  "Hobart": (-42.8821, 147.3272),
  "Adelaide": (-34.9285, 138.6007),
  "Alice Springs": (-23.6980, 133.8807),
  "Darwin": (-12.4634, 130.8456),
  "Perth": (-31.9505, 115.8605)
},
"Jepang": {
  "Tokyo": (35.6762, 139.6503),
  "Osaka": (34.6937, 135.5023),
  "Sapporo": (43.0618, 141.3545),
  "Naha": (26.2124, 127.6809)
},
"Turki": {
  "Istanbul": (41.0082, 28.9784),
  "Ankara": (39.9334, 32.8597),
  "Izmir": (38.4192, 27.1287),
  "Konya": (37.8746, 32.4932),
  "Erzurum": (39.9043, 41.2679),
  "Antalya": (36.8969, 30.7133)
},
"Inggris Raya": {
  # London dan sekitarnya
  "London": (51.5074, -0.1278),
  "Greenwich": (51.4826, -0.0077),

  # Inggris
  "Birmingham": (52.4862, -1.8904),
  "Manchester": (53.4808, -2.2426),
  "Liverpool": (53.4084, -2.9916),
  "Leeds": (53.8008, -1.5491),
  "Sheffield": (53.3811, -1.4701),
  "Bristol": (51.4545, -2.5879),
  "Newcastle upon Tyne": (54.9783, -1.6178),
  "Nottingham": (52.9548, -1.1581),
  "Leicester": (52.6369, -1.1398),
  "Coventry": (52.4068, -1.5197),
  "Bradford": (53.7950, -1.7594),
  "Hull": (53.7676, -0.3274),
  "Stoke-on-Trent": (53.0027, -2.1794),
  "Wolverhampton": (52.5862, -2.1287),
  "Derby": (52.9225, -1.4746),
  "Southampton": (50.9097, -1.4044),
  "Portsmouth": (50.8198, -1.0880),
  "Plymouth": (50.3755, -4.1427),

```

```

"Brighton": (50.8225, -0.1372),
"Oxford": (51.7520, -1.2577),
"Cambridge": (52.2053, 0.1218),
"York": (53.9590, -1.0815),
"Exeter": (50.7184, -3.5339),
"Norwich": (52.6309, 1.2974),
"Reading": (51.4543, -0.9781),
"Milton Keynes": (52.0406, -0.7594),
"Peterborough": (52.5695, -0.2405),

# Skotlandia
"Glasgow": (55.8642, -4.2518),
"Edinburgh": (55.9533, -3.1883),
"Aberdeen": (57.1497, -2.0943),
"Dundee": (56.4620, -2.9707),
"Inverness": (57.4778, -4.2247),
"Stirling": (56.1165, -3.9369),

# Wales
"Cardiff": (51.4816, -3.1791),
"Swansea": (51.6214, -3.9436),
"Newport": (51.5842, -2.9977),
"Wrexham": (53.0430, -2.9916),

# Irlandia Utara
"Belfast": (54.5973, -5.9301),
"Derry": (54.9966, -7.3086),
"Lisburn": (54.5162, -6.0580),
"Newry": (54.1751, -6.3402)
},
"Eropa Daratan": {
  "Paris": (48.8566, 2.3522),
  "Berlin": (52.5200, 13.4050),
  "Roma": (41.9028, 12.4964),
  "Madrid": (40.4168, -3.7038),
  "Cordoba": (37.8882, -4.7794),
  "Sarajevo": (43.8563, 18.4131)
},
"Afrika Selatan": {
  "Cape Town": (-33.9249, 18.4241),
  "Johannesburg": (-26.2041, 28.0473),
  "Durban": (-29.8587, 31.0218),
  "Pretoria": (-25.7479, 28.2293),
  "Bloemfontein": (-29.1141, 26.2206),
  "Port Elizabeth": (-33.9608, 25.6022)
},
"Rusia": {
  "Makhachkala": (42.9831, 47.5046), "Grozny": (43.3194, 45.6949), "Sochi": (43.5853, 39.7203),

```

```

    "Astrakhan": (46.3497, 48.0408), "Rostov-na-Donu": (47.2357, 39.7015), "Volgograd": (48.7000,
44.5167),
    "Kazan": (55.7963, 49.1088), "Ufa": (54.7388, 55.9721), "Samara": (53.2415, 50.2212),
    "Nizhny Novgorod": (56.3269, 44.0059), "Moskow": (55.7558, 37.6173), "Sankt-Peterburg":
(59.9343, 30.3351),
    "Murmansk": (68.9585, 33.0827), "Arkhangelsk": (64.5399, 40.5167), "Yekaterinburg": (56.8389,
60.6057),
    "Chelyabinsk": (55.1644, 61.4368), "Omsk": (54.9885, 73.3242), "Novosibirsk": (55.0084, 82.9357),
    "Krasnoyarsk": (56.0153, 92.8932), "Irkutsk": (52.2978, 104.2964), "Khabarovsk": (48.4814,
135.0721),
    "Vladivostok": (43.1198, 131.8869)
},
"Aljazair": {
    "Aljir": (36.7538, 3.0588),
    "Oran": (35.6987, -0.6308),
    "Constantine": (36.3650, 6.6147),
    "Tamanrasset": (22.7850, 5.5228),
    "Tindouf": (27.6711, -8.1474)
},
"Tunisia": {
    "Tunis": (36.8065, 10.1815),
    "Sfax": (34.7406, 10.7603),
    "Tozeur": (33.9197, 8.1335)
},
"Libya": {
    "Tripoli": (32.8872, 13.1913),
    "Benghazi": (32.1167, 20.0667),
    "Sebha": (27.0377, 14.4283)
},
"Afrika Barat dan Tengah": {
    "Lagos": (6.5244, 3.3792), "Dakar": (14.7167, -17.4677), "Accra": (5.5560, -0.1969),
    "Abidjan": (5.3600, -4.0083), "Bamako": (12.6392, -8.0029), "Kano": (12.0022, 8.5920),
    "Port Harcourt": (4.8156, 7.0498), "Conakry": (9.5092, -13.7122), "Freetown": (8.4657, -13.2317),
    "Monrovia": (6.3156, -10.8074), "Ouagadougou": (12.3714, -1.5197), "Niamey": (13.5116, 2.1254),
    "Kinshasa": (-4.4419, 15.2663), "Brazzaville": (-4.2634, 15.2429), "Yaounde": (3.8480, 11.5021),
    "Douala": (4.0511, 9.7679), "Libreville": (0.4162, 9.4673), "Bangui": (4.3947, 18.5582),
    "N'Djamena": (12.1348, 15.0557), "Lubumbashi": (-11.6609, 27.4794), "Malabo": (3.7504, 8.7860),
    "Pointe-Noire": (-4.7692, 11.8664)
},
"China": {
    "Beijing": (39.9042, 116.4074), "Tianjin": (39.0842, 117.2010), "Shijiazhuang": (38.0428, 114.5149),
    "Taiyuan": (37.8706, 112.5489), "Hohhot": (40.8423, 111.6708), "Shenyang": (41.8057, 123.4315),
    "Dalian": (38.9140, 121.6147), "Changchun": (43.8171, 125.3235), "Harbin": (45.8038, 126.5350),
    "Tangshan": (39.6309, 118.1802), "Baoding": (38.8739, 115.4648), "Handan": (36.6256, 114.5395),
    "Qinhuangdao": (39.9354, 119.5865), "Datong": (40.0768, 113.2914), "Baotou": (40.6522,
109.8222),
    "Anshan": (41.1078, 122.9945), "Jilin": (43.8378, 126.5494), "Qiqihar": (47.3543, 123.9182),

```

"Chengde": (40.9515, 117.9338), "Zhangjiakou": (40.8244, 114.8797), "Shanghai": (31.2304, 121.4737),
 "Nanjing": (32.0603, 118.7969), "Hangzhou": (30.2741, 120.1551), "Jinan": (36.6512, 117.1201),
 "Qingdao": (36.0671, 120.3826), "Fuzhou": (26.0745, 119.2965), "Xiamen": (24.4798, 118.0894),
 "Hefei": (31.8206, 117.2272), "Nanchang": (28.6820, 115.8579), "Suzhou": (31.2989, 120.5853),
 "Wuxi": (31.4912, 120.3119), "Ningbo": (29.8683, 121.5439), "Wenzhou": (27.9938, 120.6994),
 "Quanzhou": (24.8739, 118.6758), "Xuzhou": (34.2646, 117.1859), "Yantai": (37.4638, 121.4479),
 "Weifang": (36.7068, 119.1618), "Jinhua": (29.0781, 119.6495), "Shaoxing": (30.0024, 120.5763),
 "Nantong": (32.0147, 120.8646), "Guangzhou": (23.1291, 113.2644), "Shenzhen": (22.5431, 114.0579),
 "Dongguan": (23.0205, 113.7518), "Foshan": (23.0215, 113.1214), "Nanning": (22.8146, 108.3200),
 "Haikou": (20.0174, 110.3212), "Sanya": (18.2528, 109.5119), "Zhuhai": (22.2707, 113.5767),
 "Zhongshan": (22.5176, 113.3928), "Shantou": (23.3541, 116.6820), "Zhanjiang": (21.2707, 110.3590),
 "Guilin": (25.2536, 110.2902), "Liuzhou": (24.3146, 109.4160), "Beihai": (21.4812, 109.1194),
 "Huizhou": (23.1118, 114.4162), "Jiangmen": (22.5787, 113.0816), "Zhaoqing": (23.0515, 112.4651),
 "Chaozhou": (23.6570, 116.6226), "Maoming": (21.6620, 110.9255), "Yangjiang": (21.8565, 111.9826),
 "Wuhan": (30.5928, 114.3055), "Zhengzhou": (34.7466, 113.6253), "Changsha": (28.2282, 112.9388),
 "Luoyang": (34.6198, 112.4539), "Yichang": (30.6919, 111.2865), "Xiangyang": (32.0084, 112.1224),
 "Jingzhou": (30.3349, 112.2397), "Yueyang": (29.3730, 113.1287), "Zhuzhou": (27.8274, 113.1330),
 "Xiangtan": (27.8296, 112.9332), "Hengyang": (26.8954, 112.5719), "Kaifeng": (34.7969, 114.3076),
 "Xinxiang": (35.3031, 113.9268), "Anyang": (36.0967, 114.3920), "Xuchang": (34.0253, 113.8223),
 "Pingdingshan": (33.7371, 113.3005), "Nanyang": (32.9908, 112.5283), "Xinyang": (32.1458, 114.0847),
 "Shangqiu": (34.4150, 115.6562), "Zhumadian": (32.9773, 114.0254), "Chengdu": (30.5728, 104.0668),
 "Chongqing": (29.5630, 106.5515), "Xi'an": (34.3416, 108.9398), "Kunming": (25.0453, 102.7100),
 "Guiyang": (26.6470, 106.6302), "Urumqi": (43.8256, 87.6168), "Lanzhou": (36.0611, 103.8343),
 "Xining": (36.6171, 101.7782), "Yinchuan": (38.4872, 106.2309), "Lhasa": (29.6500, 91.1000),
 "Mianyang": (31.4675, 104.6796), "Nanchong": (30.8378, 106.0849), "Zunyi": (27.6868, 106.9272),
 "Qujing": (25.4900, 103.7978), "Dali": (25.5916, 100.2227), "Lijiang": (26.8778, 100.2277),
 "Kashgar": (39.4677, 75.9938), "Turpan": (42.9452, 89.1835), "Jiuquan": (39.7289, 98.5042),
 "Baoji": (34.3619, 107.1448)
 },
 "Asia Selatan dan Timur Tengah": {
 "New Delhi": (28.6139, 77.2090), "Mumbai": (19.0760, 72.8777), "Bengaluru": (12.9716, 77.5946),
 "Chennai": (13.0827, 80.2707), "Kolkata": (22.5726, 88.3639), "Hyderabad": (17.3850, 78.4867),
 "Ahmedabad": (23.0225, 72.5714), "Pune": (18.5204, 73.8567), "Jaipur": (26.9124, 75.7873),
 "Surat": (21.1702, 72.8311), "Lucknow": (26.8467, 80.9462), "Kanpur": (26.4499, 80.3319),
 "Nagpur": (21.1458, 79.0882), "Indore": (22.7196, 75.8577), "Bhopal": (23.2599, 77.4126),
 "Patna": (25.5941, 85.1376), "Vadodara": (22.3072, 73.1812), "Agra": (27.1767, 78.0081),
 "Varanasi": (25.3176, 82.9739), "Ludhiana": (30.9010, 75.8573), "Karachi": (24.8607, 67.0011),
 "Lahore": (31.5204, 74.3587), "Islamabad": (33.6844, 73.0479), "Rawalpindi": (33.5984, 73.0441),
 "Faisalabad": (31.4504, 73.1350), "Multan": (30.1575, 71.5249), "Peshawar": (34.0151, 71.5249),
 "Quetta": (30.1798, 66.9750), "Gujranwala": (32.1617, 74.1883), "Sialkot": (32.4945, 74.5229),
 }


```

    "Hyderabad_PK": (25.3960, 68.3578), "Bahawalpur": (29.3957, 71.6833), "Sargodha": (32.0740,
72.6861),
    "Sukkur": (27.7052, 68.8574), "Larkana": (27.5570, 68.2028), "Sheikhupura": (31.7167, 73.9850),
    "Rahim Yar Khan": (28.4212, 70.2989), "Jhang": (31.2681, 72.3181), "Dera Ghazi Khan": (30.0489,
70.6317),
    "Gujrat": (32.5736, 74.0741), "Tehran": (35.6892, 51.3890), "Mashhad": (36.2972, 59.6067),
    "Isfahan": (32.6539, 51.6660), "Karaj": (35.8327, 50.9915), "Shiraz": (29.5918, 52.5836),
    "Tabriz": (38.0773, 46.2919), "Qom": (34.6416, 50.8746), "Ahvaz": (31.3183, 48.6706),
    "Kermanshah": (34.3142, 47.0650), "Urmia": (37.5527, 45.0761), "Rasht": (37.2799, 49.5886),
    "Zahedan": (29.4963, 60.8629), "Hamadan": (34.7982, 48.5146), "Kerman": (30.2839, 57.0834),
    "Yazd": (31.8974, 54.3569), "Ardabil": (38.2514, 48.2973), "Bandar Abbas": (27.1832, 56.2666),
    "Arak": (34.0954, 49.6909), "Zanjan": (36.6736, 48.4787), "Sanandaj": (35.3144, 46.9923),
    "Kabul": (34.5553, 69.2075), "Kandahar": (31.6200, 65.7158), "Herat": (34.3529, 62.2040),
    "Mazar-i-Sharif": (36.7000, 67.1167), "Jalalabad": (34.4265, 70.4515), "Kunduz": (36.7286, 68.8681),
    "Taloqan": (36.7361, 69.5345), "Puli Khumri": (35.9446, 68.7151), "Sheberghan": (36.6676,
65.7529),
    "Zaranj": (30.9596, 61.8604), "Maymana": (35.9214, 64.7836), "Ghazni": (33.5539, 68.4203),
    "Khost": (33.3338, 69.9172), "Shir Khan Bandar": (37.1704, 68.5833), "Charikar": (35.0136,
69.1715),
    "Lashkargah": (31.5938, 64.3716), "Farah": (32.3745, 62.1164), "Asadabad": (34.8731, 71.1469),
    "Gardez": (33.5856, 69.2259), "Bamyan": (34.8216, 67.8242), "Baghdad": (33.3152, 44.3661),
    "Basra": (30.5081, 47.7835), "Mosul": (36.3400, 43.1300), "Erbil": (36.1901, 44.0090),
    "Kirkuk": (35.4670, 44.3831), "Najaf": (31.9961, 44.3147), "Karbala": (32.6160, 44.0249),
    "Sulaymaniyah": (35.5618, 45.4328), "Nasiriyah": (31.0580, 46.2573), "Amarah": (31.8413, 47.1436),
    "Diwaniyah": (31.9868, 44.9215), "Kut": (32.5115, 45.8247), "Hilla": (32.4800, 44.4336),
    "Ramadi": (33.4243, 43.2989), "Fallujah": (33.3491, 43.7828), "Samarra": (34.1979, 43.8906),
    "Baqubah": (33.7466, 44.6247), "Halabja": (35.1778, 45.9861), "Zakho": (37.1415, 42.6848),
    "Duhok": (36.8679, 42.9884)
}
}

```

```

# =====
# FUNGSI UNDUH EPHEMERIS & MAP
# =====
def download_custom_bsp(filename, url):
    filepath = os.path.join(BASE_DIR, filename)
    if not os.path.exists(filepath):
        try:
            req = urllib.request.Request(url, headers={'User-Agent': 'Mozilla/5.0'})
            with urllib.request.urlopen(req) as response, open(filepath, 'wb') as out_file:
                out_file.write(response.read())
        except Exception: pass

# =====
# FUNGSI-FUNGSI FORMATTING & SAFETY
# =====
def get_safe_events(t_obj, y_obj):
    y_arr = np.atleast_1d(y_obj)

```

```

if len(y_arr) == 0: return []
events = []
for k in range(len(y_arr)):
    try: t_val = t_obj[k]
    except Exception: t_val = t_obj
    events.append((t_val, int(y_arr[k])))
return events

def format_angle(deg, is_ra=False):
    if deg is None or math.isnan(deg): return "+00°:00':00\""
    if hasattr(deg, 'item'): deg = deg.item()
    sign = "+" if deg >= 0 else "-"
    deg = abs(deg)
    if is_ra:
        hours = deg / 15.0
        h = int(hours)
        m = int((hours - h) * 60)
        s = int(round((hours - h - m/60.0) * 3600))
        if s == 60: s = 0; m += 1
        if m == 60: m = 0; h += 1
        return f"{sign}{h:02d}H {m:02d}M {s:02d}S"
    else:
        d = int(deg)
        m = int((deg - d) * 60)
        s = int(round((deg - d - m/60.0) * 3600))
        if s == 60: s = 0; m += 1
        if m == 60: m = 0; d += 1
        return f"{sign}{d:02d}°:{m:02d}':{s:02d}\"

def format_eph_angle(deg, is_ra=False, is_sd=False):
    if deg is None or math.isnan(deg): return "00H 00M 00S" if is_ra else "+00:00:00"
    if hasattr(deg, 'item'): deg = deg.item()
    sign = "+" if deg >= 0 else "-"
    deg = abs(deg)
    if is_ra:
        hours = deg / 15.0
        h = int(hours)
        m = int((hours - h) * 60)
        s = int(round((hours - h - m/60.0) * 3600))
        if s >= 60: s -= 60; m += 1
        if m >= 60: m -= 60; h += 1
        if h >= 24: h -= 24
        return f"{sign}{h:02d}H {m:02d}M {s:02d}S"
    else:
        d = int(deg)
        m = int((deg - d) * 60)
        s = int(round((deg - d - m/60.0) * 3600))
        if s >= 60: s -= 60; m += 1

```

```

    if m >= 60: m -= 60; d += 1
    if is_sd: return f"{d:02d}:{m:02d}:{s:02d}"
    if not is_sd and d >= 100: return f"{d:03d}:{m:02d}:{s:02d}"
    return f"{sign}{d:02d}:{m:02d}:{s:02d}"

def format_time_hms(delta_hours):
    if hasattr(delta_hours, 'item'): delta_hours = delta_hours.item()
    sign = "+" if delta_hours >= 0 else "-"
    delta_hours = abs(delta_hours)
    h = int(delta_hours)
    m = int(round((delta_hours - h) * 60))
    if m == 60: m = 0; h += 1
    return f"{sign}{h:02d}H {m:02d}M"

def get_hms_str(time_obj, tz_offset):
    if time_obj is None: return "--:--"
    dt_utc = time_obj.utcnow() # Gunakan output datetime asli
    dt_local = dt_utc + datetime.timedelta(hours=tz_offset)
    return dt_local.strftime("%H.%M")

def get_live_weather_status(lat, lon):
    """ Mengambil data tutupan awan dari Open-Meteo. """
    url = f"https://api.open-
meteo.com/v1/forecast?latitude={lat}&longitude={lon}&current=cloud_cover,relative_humidity_2m&ti
mezone=auto"
    try:
        response = requests.get(url, timeout=5)
        response.raise_for_status()
        data = response.json()

        cloud_cover = data['current']['cloud_cover']
        if cloud_cover > 80:
            return f"🔴 AWAN {cloud_cover}% - Fisik Berisiko Terhalang!", "#FF5252"
        elif cloud_cover > 50:
            return f"🟡 AWAN {cloud_cover}% - Ufuk Sebagian Terhalang", "#FFD54F"
        else:
            return f"🟢 CUACA CERAH: Awan {cloud_cover}% - Ideal untuk Rukyat", "#00E676"
    except Exception:
        return f"🕒 Data cuaca tidak tersedia (Offline)", "#9E9E9E"

def calculate_saemundsson_refraction(alt_true, temp, press):
    import math
    if alt_true < -2.0: return 0.0
    r_arcmin = 1.02 / math.tan(math.radians(alt_true + 10.3 / (alt_true + 5.11)))
    correction = (press / 1010.0) * (283.15 / (273.15 + temp))
    return (r_arcmin * correction) / 60.0

```

```
# Database Bintang Terang (Magnitude < 2.0) - Koordinat J2000
```

```
BRIGHT_STARS = [  
    {"name": "Sirius", "ra": (6, 45, 8.9), "dec": (-16, 42, 58), "mag": -1.46},  
    {"name": "Canopus", "ra": (6, 23, 57.1), "dec": (-52, 41, 45), "mag": -0.74},  
    {"name": "Rigel Kentaurus", "ra": (14, 39, 36.5), "dec": (-60, 50, 2), "mag": -0.27},  
    {"name": "Arcturus", "ra": (14, 15, 39.7), "dec": (19, 10, 57), "mag": -0.05},  
    {"name": "Vega", "ra": (18, 36, 56.3), "dec": (38, 47, 1), "mag": 0.03},  
    {"name": "Capella", "ra": (5, 16, 41.4), "dec": (45, 59, 53), "mag": 0.08},  
    {"name": "Rigel", "ra": (5, 14, 32.3), "dec": (-8, 12, 6), "mag": 0.13},  
    {"name": "Procyon", "ra": (7, 39, 18.1), "dec": (5, 13, 30), "mag": 0.34},  
    {"name": "Achernar", "ra": (1, 37, 42.8), "dec": (-57, 14, 12), "mag": 0.46},  
    {"name": "Betelgeuse", "ra": (5, 55, 10.3), "dec": (7, 24, 25), "mag": 0.50},  
    {"name": "Hadar", "ra": (14, 3, 49.4), "dec": (-60, 22, 23), "mag": 0.61},  
    {"name": "Altair", "ra": (19, 50, 47.0), "dec": (8, 52, 6), "mag": 0.76},  
    {"name": "Aldebaran", "ra": (4, 35, 55.2), "dec": (16, 30, 33), "mag": 0.86},  
    {"name": "Antares", "ra": (16, 29, 24.4), "dec": (-26, 25, 55), "mag": 0.96},  
    {"name": "Spica", "ra": (13, 25, 11.6), "dec": (-11, 9, 41), "mag": 0.97},  
    {"name": "Pollux", "ra": (7, 45, 18.9), "dec": (28, 1, 34), "mag": 1.14},  
    {"name": "Fomalhaut", "ra": (22, 57, 39.1), "dec": (-29, 37, 20), "mag": 1.16},  
    {"name": "Deneb", "ra": (20, 41, 25.9), "dec": (45, 16, 49), "mag": 1.25},  
    {"name": "Regulus", "ra": (10, 8, 22.3), "dec": (11, 58, 2), "mag": 1.35},  
    {"name": "Adhara", "ra": (6, 58, 37.5), "dec": (-28, 58, 19), "mag": 1.50},  
    {"name": "Castor", "ra": (7, 34, 36.0), "dec": (31, 53, 18), "mag": 1.58}  
]
```

```
class KomparasiObservatorium(ctk.CTkToplevel):
```

```
    def __init__(self, master, eph, ts):  
        super().__init__(master)
```

```
        self.title("Observatorium Live: Komparasi Stellarium vs Ephemeris NASA")  
        self.geometry("1100x650")  
        self.attributes("-topmost", True)  
        self.configure(fg_color="#0A0A10")
```

```
        # Menggunakan mesin Skyfield dari KHGTApp agar tidak perlu load file .bsp 2 kali  
        self.ts = ts  
        self.eph = eph  
        self.ephemeris_loaded = True if self.eph else False
```

```
        if self.ephemeris_loaded:  
            self.earth = self.eph['earth']  
            self.sun_calc = self.eph['sun']  
            self.moon_calc = self.eph['moon']
```

```
        self.title_label = ctk.CTkLabel(self, text="🔭 KOMPARASI DATA: STELLARIUM vs KOMPUTASI  
ENGINE", font=("Segoe UI", 22, "bold"), text_color="#FFD54F")
```

```

self.title_label.pack(pady=(20, 5))

self.info_label = ctk.CTkLabel(self, text="Memeriksa sistem dan memuat data API...", font=("Segoe
UI", 14), text_color="#AAAAAA")
self.info_label.pack(pady=(0, 20))

self.cek_dan_jalankan_stellarium()

self.main_container = ctk.CTkFrame(self, fg_color="transparent")
self.main_container.pack(fill="both", expand=True, padx=20, pady=(0, 10))
self.main_container.grid_columnconfigure(0, weight=1, uniform="group1")
self.main_container.grid_columnconfigure(1, weight=1, uniform="group1")
self.main_container.grid_rowconfigure(0, weight=1)

# --- PENAMBAHAN METRIK BARU ---
self.metrics = [
    "Ketinggian (Alt)", "Arah (Azimuth)", "Jarak (AU)",
    "RA", "Deklinasi (Dec)", "Elongasi", "Iluminasi", "Magnitude"
]

# --- PANEL MATAHARI ---
self.sun_frame = ctk.CTkFrame(self.main_container, corner_radius=15, fg_color="#1E1E1E",
border_width=1, border_color="#FFD700")
self.sun_frame.grid(row=0, column=0, padx=(0, 10), sticky="nsew")
ctk.CTkLabel(self.sun_frame, text="☀ MATAHARI", font=("Segoe UI", 18, "bold"),
text_color="#FFD700").pack(pady=15)

self.sun_table = ctk.CTkFrame(self.sun_frame, fg_color="transparent")
self.sun_table.pack(fill="both", expand=True, padx=15, pady=(0, 15))
self.sun_widgets = {}
self.buat_header_tabel(self.sun_table)
self.buat_baris_tabel(self.sun_table, self.metrics, self.sun_widgets)

# --- PANEL BULAN ---
self.moon_frame = ctk.CTkFrame(self.main_container, corner_radius=15, fg_color="#1E1E1E",
border_width=1, border_color="#00E5FF")
self.moon_frame.grid(row=0, column=1, padx=(10, 0), sticky="nsew")
ctk.CTkLabel(self.moon_frame, text="🌙 BULAN", font=("Segoe UI", 18, "bold"),
text_color="#00E5FF").pack(pady=15)

self.moon_table = ctk.CTkFrame(self.moon_frame, fg_color="transparent")
self.moon_table.pack(fill="both", expand=True, padx=15, pady=(0, 15))
self.moon_widgets = {}
self.buat_header_tabel(self.moon_table)
self.buat_baris_tabel(self.moon_table, self.metrics, self.moon_widgets)

self.status_label = ctk.CTkLabel(self, text="Status: Inisialisasi...", font=("Segoe UI", 13, "bold"))

```

```

self.status_label.pack(side="bottom", pady=15)

# Pengaman agar loop berhenti saat jendela ditutup
self.protocol("WM_DELETE_WINDOW", self.on_close)
self.is_running = True
self.update_dashboard()

def on_close(self):
    self.is_running = False
    self.destroy()

def cek_dan_jalankan_stellarium(self):
    try:
        import subprocess
        output = subprocess.check_output('tasklist', creationflags=0x08000000)
        if b'stellarium.exe' not in output.lower():
            self.info_label.configure(text="Mendeteksi Stellarium belum aktif. Mencoba membuka
otomatis...")
            jalur_instalasi = [
                r"C:\Program Files\Stellarium\stellarium.exe",
                r"C:\Program Files (x86)\Stellarium\stellarium.exe",
                os.path.expandvars(r"%LOCALAPPDATA%\Programs\Stellarium\stellarium.exe")
            ]
            berhasil_buka = False
            for jalur in jalur_instalasi:
                if os.path.exists(jalur):
                    os.startfile(jalur)
                    berhasil_buka = True
                    break
            if not berhasil_buka:
                self.info_label.configure(text="Gagal menemukan stellarium.exe. Silakan buka aplikasi secara
manual.")
        except Exception:
            pass

def buat_header_tabel(self, parent):
    headers = ["Metrik", "Stellarium", "Sistem KHGT", "Δ Selisih"]
    for i in range(4):
        parent.grid_columnconfigure(i, weight=1)
        lbl = ctk.CTkLabel(parent, text=headers[i], font=("Segoe UI", 12, "bold"), text_color="#FFFFFF")
        lbl.grid(row=0, column=i, pady=(0, 10), padx=5, sticky="w" if i==0 else "e")

def buat_baris_tabel(self, parent, metrics, widget_dict):
    for i, metric in enumerate(metrics, start=1):
        parent.grid_rowconfigure(i, weight=1)
        ctk.CTkLabel(parent, text=metric, font=("Segoe UI", 12), text_color="#AAAAAA").grid(row=i,
column=0, sticky="w", padx=5, pady=8)
        lbl_stel = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#00FFFF")

```



```

lbl_stel.grid(row=i, column=1, sticky="e", padx=5, pady=8)
lbl_calc = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#00E676")
lbl_calc.grid(row=i, column=2, sticky="e", padx=5, pady=8)
lbl_delta = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#FF5252")
lbl_delta.grid(row=i, column=3, sticky="e", padx=5, pady=8)
widget_dict[metric] = {"stel": lbl_stel, "calc": lbl_calc, "delta": lbl_delta}

```

```

def fetch_stellarium_state(self):
    import datetime
    import pytz
    import requests
    try:
        res = requests.get("http://localhost:8090/api/main/status", timeout=2)
        if res.status_code == 200:
            data = res.json()
            lat = data.get('location', {}).get('latitude', 0)
            lon = data.get('location', {}).get('longitude', 0)
            alt = data.get('location', {}).get('altitude', 0)
            loc_name = data.get('location', {}).get('name', 'Unknown')

            utc_str = data.get('time', {}).get('utc', "")
            try:
                dt_utc = datetime.datetime.strptime(utc_str[:19], "%Y-%m-%dT%H:%M:%S")
                dt_utc = dt_utc.replace(tzinfo=pytz.utc)
            except Exception as e:
                dt_utc = datetime.datetime.now(pytz.utc)

            return {"lat": lat, "lon": lon, "alt": alt, "loc_name": loc_name, "time": dt_utc, "ok": True}
        except Exception as e:
            pass
    return {"ok": False}

```

```

def fetch_stellarium_object(self, name):
    import requests
    import math
    try:
        res = requests.get("http://localhost:8090/api/objects/info", params={'name': name, 'format':
'json'}, timeout=2)
        if res.status_code == 200:
            d = res.json()

            # Stellarium API default mengembalikan elongasi dalam bentuk Radian
            elong_rad = d.get('elongation', 0)
            elong_deg = math.degrees(elong_rad) if elong_rad else 0.0

            # ---> PERBAIKAN ILUMINASI MATAHARI & BULAN STELLARIUM <---
            if name == "Sun":
                iluminasi_stel = 100.0 # Matahari selalu 100%

```

```

else:
    iluminasi_stel = d.get('illuminated')
    if iluminasi_stel is None:
        iluminasi_stel = d.get('phase', 0.0) * 100.0
# -----

return {
    "Ketinggian (Alt)": d.get('altitude', 0),
    "Arah (Azimuth)": d.get('azimuth', 0),
    "Jarak (AU)": d.get('distance', 0),
    "RA": d.get('ra', 0),
    "Deklinasi (Dec)": d.get('dec', 0),
    "Elongasi": elong_deg,
    "Iluminasi": iluminasi_stel,
    "Magnitude": d.get('vmag', 0)
}
except:
    pass
return None

def kalkulasi_skyfield(self, target_name, state):
    if not self.ephemeris_loaded: return None
    from skyfield.api import wgs84
    from skyfield import almanac
    import math

    t = self.ts.from_datetime(state['time'])
    observer = self.earth + wgs84.latlon(state['lat'], state['lon'], state['alt'])

    target_eph = self.sun_calc if target_name == "Sun" else self.moon_calc
    apparent = observer.at(t).observe(target_eph).apparent()
    alt, az, dist = apparent.altaz()
    ra, dec, _ = apparent.radec(epoch='date')

    # --- KALKULASI EKSTRA (ELONGASI, ILUMINASI, MAGNITUDE) ---
    elongation = 0.0
    illumination = 100.0
    magnitude = 0.0

    if target_name == "Moon":
        sun_app = observer.at(t).observe(self.sun_calc).apparent()
        elongation = sun_app.separation_from(apparent).degrees

    illum_val = almanac.fraction_illuminated(self.eph, 'moon', t)
    illumination = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0

    # Aproksimasi Magnitude Visual Bulan (Rumus Astronomis Standar)

```

```

        phase_angle = math.degrees(math.acos(2 * (illumination / 100.0) - 1)) if illumination <= 100.0
    else 0
        magnitude = -12.73 + 0.026 * abs(phase_angle) + (4e-9) * (phase_angle**4)

    elif target_name == "Sun":
        elongation = 0.0
        illumination = 100.0
        magnitude = -26.74

    return {
        "Ketinggian (Alt)": alt.degrees,
        "Arah (Azimuth)": az.degrees,
        "Jarak (AU)": dist.au,
        "RA": ra._degrees,
        "Deklinasi (Dec)": dec.degrees,
        "Elongasi": elongation,
        "Iluminasi": illumination,
        "Magnitude": magnitude
    }

def perbarui_ui_tabel(self, widgets, data_stel, data_calc):
    for metrik in self.metrics:
        if data_stel and data_calc:
            v_stel = data_stel.get(metrik, 0)
            v_calc = data_calc.get(metrik, 0)
            delta = abs(v_stel - v_calc)

            # Format Tampilan Berbeda Berdasarkan Jenis Metrik
            if metrik == "Jarak (AU)":
                fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.6f}", f"{v_calc:.6f}", f"{delta:.6f}"
            elif metrik == "Iluminasi":
                fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.2f}%", f"{v_calc:.2f}%", f"{delta:.2f}%"
            elif metrik == "Magnitude":
                fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.2f}", f"{v_calc:.2f}", f"{delta:.2f}"
            else:
                fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.4f}°", f"{v_calc:.4f}°", f"{delta:.4f}°"

            widgets[metrik]["stel"].configure(text=fmt_stel)
            widgets[metrik]["calc"].configure(text=fmt_calc)
            widgets[metrik]["delta"].configure(text=fmt_delta)
        else:
            widgets[metrik]["stel"].configure(text="--")
            widgets[metrik]["calc"].configure(text="--")
            widgets[metrik]["delta"].configure(text="--")

def update_dashboard(self):
    if not self.is_running: return
    state = self.fetch_stellarium_state()

```

```

if state['ok'] and self.ephemeris_loaded:
    waktu_str = state['time'].strftime("%d-%m-%Y %H:%M:%S UTC")
    self.info_label.configure(text=f"Lokasi: {state['loc_name']} (Lat: {state['lat']:.2f}, Lon: {state['lon']:.2f}) | Waktu Komputasi: {waktu_str}")

    sun_stel = self.fetch_stellarium_object("Sun")
    sun_calc = self.kalkulasi_skyfield("Sun", state)
    self.perbarui_ui_tabel(self.sun_widgets, sun_stel, sun_calc)

    moon_stel = self.fetch_stellarium_object("Moon")
    moon_calc = self.kalkulasi_skyfield("Moon", state)
    self.perbarui_ui_tabel(self.moon_widgets, moon_stel, moon_calc)

    self.status_label.configure(text="● Sinkronisasi Data Berjalan Baik", text_color="#00E676")
else:
    if not self.ephemeris_loaded:
        self.status_label.configure(text="✗ Error: Mesin Ephemeris belum dimuat",
text_color="#FF5252")
    else:
        self.status_label.configure(text="● Menunggu API... (Pastikan Plugin 'Remote Control' AKTIF di Stellarium -> [F2] -> Plugins)", text_color="#FFD700")
        for widgets in [self.sun_widgets, self.moon_widgets]:
            for key in widgets:
                widgets[key]["stel"].configure(text="--")
                widgets[key]["calc"].configure(text="--")
                widgets[key]["delta"].configure(text="--")

self.after(1000, self.update_dashboard)

# =====
# KELAS UTAMA APLIKASI GUI
# =====
class KHGTApp(ctk.CTk):
    def __init__(self):
        super().__init__()

        # ---> INISIALISASI SISTEM BAHASA <---
        self.current_lang = "id"
        self.translations = {}
        self.load_translations()
        self._last_raw_report = ""

        # Penanda alur bahasa:
        # False = kondisi awal/beranda rilis, belum ada menu 1–41 yang dipilih pengguna.
        # True = pengguna sudah memilih salah satu menu 1–41, sehingga pergantian
        #       bahasa harus mengikuti mekanisme lama: sidebar kiri + tampilan modul.

```

```

self.user_has_selected_main_menu = False

# Cache JSON khusus halaman INFORMASI RILIS.
# Tujuan: saat klik ID/EN/AR, halaman kanan tidak menerjemahkan ulang
# seluruh teks, tetapi membaca hasil terjemahan yang sudah tersimpan.
self.release_lang_cache_path = self.get_release_lang_cache_path()
self.release_translation_cache = {}
self._release_translation_cache_dirty = False
self.load_release_language_cache()
# -----

self.title("KHGT Times: Kalkulator Integrasi KHGT, Ephemeris, Qiblah & Prayer Times")
self.geometry("1150x850")

# --- KODE UNTUK MENGGANTI ICON POJOK KIRI ATAS ---
try:
    # Pastikan logo.png berada di folder yang sama dengan script ini
    icon_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), "logo.png")

    # Buka gambar menggunakan Pillow dan convert ke format PhotoImage Tkinter
    icon_image = ImageTk.PhotoImage(Image.open(icon_path))

    # Terapkan ke window (True berarti berlaku untuk window utama dan semua sub-window)
    self.wm_iconphoto(True, icon_image)
except Exception as e:
    print(f"Peringatan: Gagal memuat ikon logo.png ({e})")
# -----

self.minsize(900, 700)

self.protocol("WM_DELETE_WINDOW", self.on_closing)

self.load_obj = Loader(BASE_DIR, verbose=False)
self.ts = self.load_obj.timescale()
self.eph = None
self.ephemeris_name = None
# Cache dua kernel agar perpindahan tahun tidak perlu membuka file berulang.
# Kernel panjang baru dimuat ketika benar-benar diperlukan.
self._ephemeris_cache = {}
self._ephemeris_lock = threading.RLock()

self.lokasi_nama = ctk.StringVar(value="Semarang, Jawa Tengah")

# --- Inisialisasi Sistem Alarm ---
self.alarm_enabled = ctk.BooleanVar(value=True)
default_audio = os.path.join(BASE_DIR, "adhan.mp3")
if os.path.exists(default_audio):
    self.adhan_audio_path = ctk.StringVar(value=default_audio)

```

```

else:
    self.adzan_audio_path = ctk.StringVar(value="")

self.daily_prayer_schedule = {}
self.last_calculated_date = None
self.last_triggered_prayer = None
self.snooze_time = None
self.snooze_prayer = None

if HAS_TOAST:
    self.toaster = ToastNotifier()
if HAS_PYGAME:
    try:
        pygame.mixer.init()
    except Exception as e:
        print("Pygame mixer init failed:", e)

# --- Variabel Simulasi 3D Matplotlib ---
self.eph3d_updating = False
self.eph3d_is_live = False
self.pe_obs = ephemeris.Observer()
self.pe_sun = ephemeris.Sun()
self.pe_moon = ephemeris.Moon()

# ---> TAMBAHAN VARIABEL KURVA ALASKA <---
self.valid_days_alaska = []
self.valid_alt_alaska = []
self.valid_elong_alaska = []
self.hover_data_alaska = {}

self.setup_ui()

threading.Thread(target=self.load_ephemeris, daemon=True).start()
self.update_calendar_widget()

# Start Alarm Tick Loop
self.after(1000, self.alarm_tick)

# Start Matplotlib 3D Ephemeris Loop
self.eph3d_live_update_loop()

def get_header(self, width):
    lines = [
        "",
        "KALENDER HIJRIAH GLOBAL TUNGGAL",
        "KHGT Times 8.0, By Kasmui"
    ]
    return "\n".join(line.center(width) for line in lines)

```



```

def _ephemeris_filename_for_year(self, target_year):
    """Pilih kernel JPL berdasarkan tahun tanpa mengubah logika modul lain."""
    y = int(target_year)
    if EPHEMERIS_SHORT_MIN_YEAR <= y <= EPHEMERIS_SHORT_MAX_YEAR:
        return EPHEMERIS_SHORT_FILE
    return EPHEMERIS_LONG_FILE

def _load_ephemeris_file(self, filename):
    """Muat satu kernel JPL; kernel yang pernah dibuka disimpan dalam cache."""
    filepath = os.path.join(BASE_DIR, filename)
    if not os.path.exists(filepath):
        raise FileNotFoundError(
            f"File ephemeris '{filename}' tidak ditemukan di folder aplikasi: {BASE_DIR}"
        )

    with self._ephemeris_lock:
        if filename in self._ephemeris_cache:
            kernel = self._ephemeris_cache[filename]
        else:
            kernel = self.load_obj(filename)
            if 'earth' not in kernel or 'sun' not in kernel or 'moon' not in kernel:
                raise ValueError(
                    f"Data utama Earth/Sun/Moon tidak ditemukan dalam '{filename}'."
                )
            self._ephemeris_cache[filename] = kernel

    self.eph = kernel
    self.ephemeris_name = filename
    return kernel

def load_ephemeris(self, target_year=None):
    """
    Muat ephemeris otomatis.
    - 1850-2149 : de440s.bsp
    - 1-1849 dan 2150-2999 : de441-new.bsp

    Pemilihan 1850-2149 sengaja memakai tahun penuh karena de440s mulai
    sekitar 26 Desember 1849 dan berakhir sekitar 22 Januari 2150.
    """
    try:
        if target_year is None:
            target_year = datetime.datetime.now().year

        target_year = int(target_year)
        if target_year < EPHEMERIS_LONG_MIN_YEAR or target_year > EPHEMERIS_LONG_MAX_YEAR:
            raise ValueError(
                f"Tahun {target_year} di luar rentang aplikasi "
            )
    
```

```

        f"({EPHEMERIS_LONG_MIN_YEAR}-{EPHEMERIS_LONG_MAX_YEAR} M)."
    )

    filename = self._ephemeris_filename_for_year(target_year)
    filepath = os.path.join(BASE_DIR, filename)

    if not os.path.exists(filepath):
        self.lbl_status.configure(
            text=f"Error: {filename} Tidak Ditemukan!",
            text_color="#FF1744"
        )
        self.after(0, lambda f=filename: messagebox.showerror(
            "File Ephemeris Hilang",
            f"File '{f}' wajib ada di folder aplikasi.\n\n"
            f"KHGT Times V8.0 memakai:\n"
            f"• {EPHEMERIS_SHORT_FILE} untuk {EPHEMERIS_SHORT_MIN_YEAR}-{EPHEMERIS_SHORT_MAX_YEAR} M\n"
            f"• {EPHEMERIS_LONG_FILE} untuk {EPHEMERIS_LONG_MIN_YEAR}-{EPHEMERIS_SHORT_MIN_YEAR - 1} M dan "
            f"{EPHEMERIS_SHORT_MAX_YEAR + 1}-{EPHEMERIS_LONG_MAX_YEAR} M"
        ))
        return

    self.lbl_status.configure(
        text=f"Memuat {filename} untuk tahun {target_year}...",
        text_color="#00E5FF"
    )

    self._load_ephemeris_file(filename)

    self.lbl_status.configure(
        text=f"Sistem Siap ({self.ephemeris_name})",
        text_color="#00E676"
    )
    self.btn_hitung.configure(state="normal")

except Exception as e:
    self.lbl_status.configure(text="Gagal memuat Ephemeris!", text_color="#FF1744")
    self.after(0, lambda err=str(e): messagebox.showerror(
        "Error Mesin", f"Gagal membaca file ephemeris: {err}"
    ))

def toggle_sidebar(self):
    if self.sidebar_visible:
        self.sidebar.grid_remove()
        self.sidebar_visible = False
    else:
        self.sidebar.grid(row=0, column=0, sticky="nsew")

```

```

self.sidebar_visible = True

def open_peta_langit_dunia(self):
    """Membuka Peta Langit Dunia KHGT Times melalui browser default."""
    try:
        opened = webbrowser.open_new_tab(WORLD_SKY_MAP_URL)
        if not opened:
            opened = webbrowser.open(WORLD_SKY_MAP_URL)
        self.lbl_status.configure(
            text="Peta Langit Dunia dibuka di browser",
            text_color="#00E676"
        )
        if not opened:
            messagebox.showinfo(
                "Peta Langit Dunia",
                f"Silakan buka alamat berikut pada browser:\n\n{WORLD_SKY_MAP_URL}"
            )
    except Exception as e:
        messagebox.showerror(
            "Gagal Membuka Peta Langit Dunia",
            f"Layanan online tidak dapat dibuka melalui browser.\n\n{WORLD_SKY_MAP_URL}\n\n{e}"
        )
        self.lbl_status.configure(
            text="Gagal membuka Peta Langit Dunia",
            text_color="#FF1744"
        )

def open_laporan_astronomi(self):
    """Membuka aplikasi pendamping LaporanAstronomi.exe dari folder aplikasi."""
    kandidat_nama = ("LaporanAstronomi.exe", "LaporanAstronomi.exe")
    exe_path = next(
        (os.path.join(BASE_DIR, nama) for nama in kandidat_nama
         if os.path.isfile(os.path.join(BASE_DIR, nama))),
        None
    )

    if exe_path is None:
        lokasi_utama = os.path.join(BASE_DIR, kandidat_nama[0])
        messagebox.showerror(
            "Laporan Astronomi Tidak Ditemukan",
            "File aplikasi pendamping tidak ditemukan.\n\n"
            f"Letakkan file 'LaporanAstronomi.exe' di folder yang sama dengan "
            f"KHGT Times.\n\nLokasi yang diperiksa:\n{lokasi_utama}"
        )
        return

    try:
        if os.name == "nt" and hasattr(os, "startfile"):

```

```

        os.startfile(exe_path)
    else:
        subprocess.Popen([exe_path], cwd=os.path.dirname(exe_path))
    self.lbl_status.configure(
        text="LaporanAstronomi.exe berhasil dibuka",
        text_color="#00E676"
    )
except Exception as e:
    messagebox.showerror(
        "Gagal Membuka Laporan Astronomi",
        f"File ditemukan, tetapi gagal dijalankan.\n\n{e}"
    )
    self.lbl_status.configure(
        text="Gagal membuka LaporanAstronomi.exe",
        text_color="#FF1744"
    )

def setup_ui(self):
    now = datetime.datetime.now()
    curr_y = str(now.year)
    curr_m = f"{now.month:02d}"
    curr_d = f"{now.day:02d}"
    curr_m_np = str(now.month)
    curr_d_np = str(now.day)

    self.grid_columnconfigure(0, weight=0)
    self.grid_columnconfigure(1, weight=1)
    self.grid_rowconfigure(0, weight=1)

    # ===== SIDEBAR =====
    self.sidebar = ctk.CTkScrollableFrame(self, width=370, corner_radius=0, fg_color="#181818")
    self.sidebar.grid(row=0, column=0, sticky="nsew")

    # ---> TAMBAHAN UI BAHASA DI KANAN "KHGT ENGINE" <---
    header_app_frame = ctk.CTkFrame(self.sidebar, fg_color="transparent")
    header_app_frame.pack(fill="x", padx=15, pady=(20, 5))

    ctk.CTkLabel(header_app_frame, text="KHGT ENGINE", font=("Segoe UI", 24, "bold"),
    text_color="#00E5FF").pack(side="left")

    self.lang_var = ctk.StringVar(value="id")
    self.combo_lang = ctk.CTkOptionMenu(header_app_frame, values=["id", "en", "ar"],
    variable=self.lang_var, width=60, height=28, command=self.change_language)
    self.combo_lang.pack(side="right")
    # -----

    # =====
    # PENGGANTI MENU SELECTION (MODERN SCROLLABLE LIST)

```

```
# =====
# Menggunakan CTKScrollableFrame agar desainnya 100% senada dengan tema aplikasi
self.frame_menu_container = ctk.CTkScrollableFrame(
    self.sidebar,
    height=280, # Tinggi kotak scroll
    fg_color="#121212",
    corner_radius=8,
    border_width=1,
    border_color="#333333",
    scrollbar_button_color="#424242",
    scrollbar_button_hover_color="#616161"
)
self.frame_menu_container.pack(fill="x", padx=15, pady=(5, 10))

menu_items = [
    # Kalender & Konversi
    "1) Kalender Hijriah Berjalan",
    "2) Kalender Masehi Berjalan",
    "3) Konversi Kalender",

    # Analisis & Visibilitas Hilal (KHGT)
    "4) Visibility Hilal (KHGT)",
    "5) Analisis Hilal Global",
    "6) Status Kriteria Batas",
    "7) Kota Pertama KHGT (Mainland)",
    "8) Crescent Visibility Map",
    "9) Altitude Chart Analyser",

    # Data Histori & Komparasi 50 Tahun
    "10) Komparasi 50 Tahun (Ramadhan)",
    "11) Tabel Tinggi Hilal 50 Thn (Ramadhan)",
    "12) Tabel Elongasi Hilal 50 Thn (Ramadhan)",
    "13) Komparasi 50 Tahun (Syawal)",
    "14) Tabel Tinggi Hilal 50 Thn (Syawal)",
    "15) Tabel Elongasi Hilal 50 Thn (Syawal)",
    "16) Komparasi 50 Tahun (Zulhijah)",
    "17) Tabel Tinggi Hilal 50 Thn (Zulhijah)",
    "18) Tabel Elongasi Hilal 50 Thn (Zulhijah)",
    "19) Komparasi 50 Tahun (12 Bulan)",

    # Ephemeris & Posisi Benda Langit
    "20) Fase Bulan (Moonphase)",
    "21) Moon Times",
    "22) Sun Times",
    "23) Sun Moon Ephemeris",
    "24) Sistem Sun Moon Earth",

    # Waktu Shalat & Arah Kiblat
```

```

"25) Prayer Times",
"26) Qiblah Direction & Times",
"27) Qibla Time (Rashdul Lokal)",
"28) Kalkulator Mizwala (Bayangan)",
"29) Evaluasi Fajar Shadiq & SQM",

# Fenomena Astronomi & Observasi
"30) Analisis Gerhana",
"31) Equinox & Solstice",
"32) Planetary Times",
"33) Generator Analemma Matahari",
"34) Prediktor Pasang Surut Gravitasi",
"35) Astrofotografi & Kontras Visibilitas",

# Simulasi, Ekstra & Admin
"36) Live Animasi",
"37) Simulasi Ephemeris 3D",
"38) WinAI: Aplikasi AI Windows",
"39) Peta Langit Dunia",
"40) Laporan Astronomi",
"41) HIJRI_DB Auto-Builder (Admin)"
]

self.menu_buttons = {}
self.selected_mode = ctk.StringVar(value=menu_items[0])

def on_menu_click(mode_name):
    # ---> TAMBAHAN PASSWORD UNTUK MENU AUTO-BUILDER <---
    if "Auto-Builder" in mode_name:
        from tkinter import simpledialog
        # Gunakan simpledialog agar input bisa disamarkan dengan tanda bintang (show='*')
        pwd = simpledialog.askstring("Otorisasi Admin", "Masukkan Password Admin:", show='*')
        if pwd != "27021966":
            messagebox.showerror("Akses Ditolak", "Password salah atau dibatalkan!")
            return # Batal pindah menu dan kembali ke menu sebelumnya
    # -----

    # Tandai bahwa setelah fase startup, pengguna benar-benar sudah masuk
    # ke salah satu menu 1–41. Saat startup flag ini akan di-reset kembali
    # sesudah show_release_info() agar halaman awal tetap dianggap beranda rilis.
    self.user_has_selected_main_menu = True

    self.selected_mode.set(mode_name)
    # Update warna tombol (Highlight menu yang sedang aktif)
    for name, btn in self.menu_buttons.items():
        if name == mode_name:
            btn.configure(fg_color="#1565C0", text_color="#FFFFFF") # Warna aktif (Background Biru,
Teks Putih)

```

```

        else:
            btn.configure(fg_color="transparent", text_color="#00E5FF") # Warna normal (Teks Cyan)

# Panggil fungsi pergantian UI utama
self.switch_mode(mode_name)

# Generate daftar menu ke dalam kotak scroll
for item in menu_items:
    btn = ctk.CTkButton(
        self.frame_menu_container,
        text=item,
        anchor="w", # Rata kiri
        fg_color="transparent",
        text_color="#00E5FF",
        hover_color="#1F1F1F",
        font=("Segoe UI", 14, "bold"),
        height=35,
        corner_radius=6,
        command=lambda m=item: on_menu_click(m)
    )
    btn.pack(fill="x", pady=2, padx=2)
    self.menu_buttons[item] = btn

# Class jembatan agar fungsi 'self.combo_mode.get()' bawaan aplikasi tetap berfungsi normal tanpa
error
class MenuBridge:
    def __init__(self, var):
        self.var = var
    def get(self):
        return self.var.get()
    def set(self, val):
        pass

self.combo_mode = MenuBridge(self.selected_mode)
# =====

# --- CONTAINER DATABASE KOTA ---
self.frame_city_select = ctk.CTkFrame(self.sidebar, fg_color="#212121")
self.frame_city_select.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(self.frame_city_select, text="PILIH KOTA (DEFAULT SEMARANG)", font=("Segoe UI",
11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))

self.opt_prov = ctk.CTkOptionMenu(self.frame_city_select, values=sorted(list(CITY_DB.keys())),
command=self.on_prov_change)
self.opt_prov.pack(fill="x", padx=10, pady=5)
self.opt_prov.set("Jawa Tengah")

```



```

self.opt_city = ctk.CTkOptionMenu(self.frame_city_select, values=sorted(list(CITY_DB["Jawa
Tengah"].keys()))), command=self.on_city_change)
self.opt_city.pack(fill="x", padx=10, pady=(0, 10))
self.opt_city.set("Semarang")

self.btn_auto_loc = ctk.CTkButton(self.frame_city_select, text="📍 Deteksi Lokasi Otomatis",
fg_color="#00695C", hover_color="#004D40", command=self.auto_detect_location)
self.btn_auto_loc.pack(fill="x", padx=10, pady=(0, 10))

# --- CONTAINER ALARM SALAT ---
self.frame_alarm_settings = ctk.CTkFrame(self.sidebar, fg_color="#212121", border_width=1,
border_color="#D32F2F")
self.frame_alarm_settings.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(self.frame_alarm_settings, text="PENGATURAN ALARM SALAT", font=("Segoe UI", 11,
"bold"), text_color="#FF5252").pack(anchor="w", padx=10, pady=(8, 2))

self.switch_alarm = ctk.CTkSwitch(self.frame_alarm_settings, text="Aktifkan Alarm",
variable=self.alarm_enabled, command=self.on_alarm_toggle, progress_color="#D32F2F")
self.switch_alarm.pack(anchor="w", padx=15, pady=5)

btn_pilih_audio = ctk.CTkButton(self.frame_alarm_settings, text="🎵 Pilih Audio Adzan",
fg_color="#424242", hover_color="#616161", command=self.pilih_file_audio)
btn_pilih_audio.pack(fill="x", padx=10, pady=5)

current_audio = self.adzan_audio_path.get()
if current_audio and os.path.exists(current_audio):
    initial_text = os.path.basename(current_audio)
else:
    initial_text = "Default System Sound"

self.lbl_audio_path = ctk.CTkLabel(self.frame_alarm_settings, text=initial_text, font=("Segoe UI",
10, "italic"), text_color="#9E9E9E")
self.lbl_audio_path.pack(anchor="w", padx=10, pady=(0, 8))

# --- CONTAINER 1: VISIBILITY INPUTS ---
self.frame_vis_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_vdate = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
frame_vdate.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_vdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# SATUKAN LABEL DAN INPUT DALAM SATU FRAME GRID
vdate_grid = ctk.CTkFrame(frame_vdate, fg_color="transparent")
vdate_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk

```

```

        ctk.CTkLabel(vdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
        ctk.CTkLabel(vdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
        ctk.CTkLabel(vdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_vday = ctk.CTkEntry(vdate_grid, width=45, placeholder_text="dd", justify="center")
self.entry_vday.insert(0, curr_d)
self.entry_vday.grid(row=1, column=0, padx=(0, 5))

self.entry_vmonth = ctk.CTkEntry(vdate_grid, width=45, placeholder_text="mm", justify="center")
self.entry_vmonth.insert(0, curr_m)
self.entry_vmonth.grid(row=1, column=1, padx=5)

self.entry_vyear = ctk.CTkEntry(vdate_grid, width=70, placeholder_text="yyyy", justify="center")
self.entry_vyear.insert(0, curr_y)
self.entry_vyear.grid(row=1, column=2, padx=(5, 0))

# ---> AWAL TAMBAHAN UI PILIHAN WAKTU OBSERVASI <---
frame_vtime_mode = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
frame_vtime_mode.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_vtime_mode, text="WAKTU OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

self.radio_vis_time_mode = ctk.StringVar(value="maghrib")

def on_vis_time_mode_change():
    if self.radio_vis_time_mode.get() == "kustom":
        self.entry_vtime.pack(fill="x", padx=10, pady=(0, 10))
    else:
        self.entry_vtime.pack_forget()

        ctk.CTkRadioButton(frame_vtime_mode, text="1) Waktu Maghrib (Sunset)",
variable=self.radio_vis_time_mode, value="maghrib",
command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=5)
        ctk.CTkRadioButton(frame_vtime_mode, text="2) Waktu Tertentu (Kustom)",
variable=self.radio_vis_time_mode, value="kustom",
command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=5)
        ctk.CTkRadioButton(frame_vtime_mode, text="3) Waktu Live (Sekarang)",
variable=self.radio_vis_time_mode, value="live",
command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=(5, 10))

self.entry_vtime = ctk.CTkEntry(frame_vtime_mode, placeholder_text="HH:MM:SS (Contoh:
17:45:00)", justify="center")
# Catatan: entry_vtime disembunyikan secara default karena mode awalnya "maghrib"
# ---> AKHIR TAMBAHAN UI <---

```

```

frame_vloc = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
frame_vloc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_vloc, text="KOORDINAT LOKASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_vlat = self.create_input_row(frame_vloc, "Lat (Lintang):", "-7.0667")
self.entry_vlon = self.create_input_row(frame_vloc, "Lon (Bujur):", "110.4100")
self.entry_velev = self.create_input_row(frame_vloc, "Elevasi (m):", "230.0")
self.entry_vtz = self.create_input_row(frame_vloc, "Timezone:", "7.0")

frame_vatm = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
frame_vatm.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_vatm, text="PARAMETER ATMOSFER", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_vtemp = self.create_input_row(frame_vatm, "Suhu (°C):", "10.0")
self.entry_vpres = self.create_input_row(frame_vatm, "Tekanan (mb):", "1010.0")
self.entry_vhum = self.create_input_row(frame_vatm, "Kelembapan (%)ate:", "60.0")

# --- CONT 2 SD 18 (SAMA SEPerti MASTER, DI RINGKAS UNTUK TEMPAT, TAPI DI-INCLUDE DI
BAWAH) ---
self._setup_other_inputs(curr_y, curr_m, curr_d, curr_m_np, curr_d_np)

# --- CONTAINER 19: 3D EPHEMERIS (SCRIPT A) INPUTS ---
self.frame_eph3d_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

# Lokasi khusus (Terhubung ke update on_city_change)
frame_eph3d_loc = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="#212121")
frame_eph3d_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_eph3d_loc, text="KOORDINAT OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_eph3d_lat = self.create_input_row(frame_eph3d_loc, "Lat (Lintang):", "-7.0667")
self.entry_eph3d_lon = self.create_input_row(frame_eph3d_loc, "Lon (Bujur):", "110.4100")
self.entry_eph3d_tz = self.create_input_row(frame_eph3d_loc, "Timezone:", "7.0") # <---
TAMBAHKAN BARIS INI

frame_eph3d_waktu = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="#212121")
frame_eph3d_waktu.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_eph3d_waktu, text="PENGATURAN WAKTU & ELEVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

self.var_eph3d_tahun, self.lbl_eph3d_tahun = self.create_eph3d_slider(frame_eph3d_waktu,
"Tahun", -3000, 3000, int(curr_y), 1)
self.var_eph3d_bulan, self.lbl_eph3d_bulan = self.create_eph3d_slider(frame_eph3d_waktu,
"Bulan", 1, 12, int(curr_m_np), 1)
self.var_eph3d_hari, self.lbl_eph3d_hari = self.create_eph3d_slider(frame_eph3d_waktu,
"Tanggal", 1, 31, int(curr_d_np), 1)
self.var_eph3d_jam, self.lbl_eph3d_jam = self.create_eph3d_slider(frame_eph3d_waktu, "Jam
(Lokal)", 0, 23.99, 12.0, 0.01)

```

```

self.var_eph3d_elev, self.lbl_eph3d_elev = self.create_eph3d_slider(frame_eph3d_waktu, "Elevasi (m)", 0, 5000, 230, 1)

frame_eph3d_btn = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="transparent")
frame_eph3d_btn.pack(fill="x", padx=15, pady=10)

btn_eph3d_sunset = ctk.CTkButton(frame_eph3d_btn, text="Cari Sunset", width=120,
fg_color="#C62828", hover_color="#B71C1C", command=self.eph3d_cari_sunset)
btn_eph3d_sunset.pack(side="left", fill="x", expand=True, padx=(0,5))

btn_eph3d_live = ctk.CTkButton(frame_eph3d_btn, text="🔴 Waktu Live", width=120,
fg_color="#2E7D32", hover_color="#1B5E20", command=self.eph3d_start_live)
btn_eph3d_live.pack(side="left", fill="x", expand=True, padx=(5,0))

# -- MIZWALA / TONGKAT ISTIWA INPUTS (MENU 29) --
self.frame_mizwala_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_miz_date = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
frame_miz_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_miz_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_miz_year, self.entry_miz_month, self.entry_miz_day =
self.create_ymd_row(frame_miz_date, curr_y, curr_m_np, curr_d_np)

frame_miz_loc = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
frame_miz_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_miz_loc, text="LOKASI OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_miz_lat = self.create_input_row(frame_miz_loc, "Lat (Lintang):", "-7.0667")
self.entry_miz_lon = self.create_input_row(frame_miz_loc, "Lon (Bujur):", "110.4100")
self.entry_miz_elev = self.create_input_row(frame_miz_loc, "Elevasi (m):", "230.0")
self.entry_miz_tz = self.create_input_row(frame_miz_loc, "Timezone:", "7.0")

frame_miz_opt = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
frame_miz_opt.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_miz_opt, text="PARAMETER TONGKAT & WAKTU", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_miz_tinggi = self.create_input_row(frame_miz_opt, "Tinggi (cm):", "100.0")

# ---> TAMBAHKAN BARIS INI UNTUK INPUT WAKTU SIMULASI <---
self.entry_miz_waktu = self.create_input_row(frame_miz_opt, "Waktu (HH:MM):", "LIVE")

# Tombol Pemicu Simulasi Mizwala
self.btn_simulasi_miz = ctk.CTkButton(
    self.frame_mizwala_inputs,
    text="🔍 Buka Simulasi Visual Mizwala",
    font=("Segoe UI", 12, "bold"),

```

```

        fg_color="#F57C00",
        hover_color="#E65100",
        command=self.buka_simulasi_mizwala
    )
    self.btn_simulasi_miz.pack(fill="x", padx=15, pady=(10, 5))
    self.entry_miz_step = self.create_input_row(frame_miz_opt, "Interval (Menit):", "10")

# -- AUTO BUILDER DATABASE HIJRIAH (MENU 30) --
self.frame_autobuild_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

    frame_ab_cfg = ctk.CTkFrame(self.frame_autobuild_inputs, fg_color="#212121", border_width=1,
border_color="#F57C00")
    frame_ab_cfg.pack(fill="x", padx=15, pady=10)
    ctk.CTkLabel(frame_ab_cfg, text="⚙️ GENERATOR DATABASE KHGT", font=("Segoe UI", 12, "bold"),
text_color="#FFD54F").pack(anchor="w", padx=10, pady=(8, 2))

    self.entry_ab_start = self.create_input_row(frame_ab_cfg, "Tahun Awal (H):", "1446")
    self.entry_ab_end = self.create_input_row(frame_ab_cfg, "Tahun Akhir (H):", "1546")

# -- MENU 31: STATUS KRITERIA BATAS --
self.frame_kriteria_batas_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

# 1. Input Tanggal Observasi
frame_kb_date = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
frame_kb_date.pack(fill="x", padx=15, pady=5)
    ctk.CTkLabel(frame_kb_date, text="TANGGAL OBSERVASI (SAAT MAGHRIB)", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
    self.entry_kb_year, self.entry_kb_month, self.entry_kb_day = self.create_ymd_row(frame_kb_date,
curr_y, curr_m_np, curr_d_np)

# 2. Input Lokasi Observasi
frame_kb_loc = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
frame_kb_loc.pack(fill="x", padx=15, pady=5)
    ctk.CTkLabel(frame_kb_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
    self.entry_kb_lat = self.create_input_row(frame_kb_loc, "Lat (Lintang):", "-7.0667")
    self.entry_kb_lon = self.create_input_row(frame_kb_loc, "Lon (Bujur):", "110.4100")
    self.entry_kb_elev = self.create_input_row(frame_kb_loc, "Elevasi (m):", "230.0")
    self.entry_kb_tz = self.create_input_row(frame_kb_loc, "Timezone:", "7.0")

# 3. Slider Manual (Jika ingin simulasi sendiri setelah perhitungan)
frame_kb_opt = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
frame_kb_opt.pack(fill="x", padx=15, pady=10)
    ctk.CTkLabel(frame_kb_opt, text="SIMULATOR PARAMETER (MANUAL)", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

```

```

self.var_kb_elongasi, self.lbl_kb_elongasi = self.create_eph3d_slider(frame_kb_opt, "Elongasi", 0.0,
15.0, 8.5, 0.1)
self.var_kb_ketinggian, self.lbl_kb_ketinggian = self.create_eph3d_slider(frame_kb_opt,
"Ketinggian", -5.0, 15.0, 6.0, 0.1)

# Trigger update grafik otomatis saat slider digeser
self.var_kb_elongasi.trace_add('write', lambda *args: self.update_kriteria_batas_plot())
self.var_kb_ketinggian.trace_add('write', lambda *args: self.update_kriteria_batas_plot())

# =====
# MODUL 31 - 34: FRAME INPUT (SIDEBAR)
# =====
# -- 31. ASTROFOTOGRAFI --
self.frame_astro_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_astro_date = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
frame_astro_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_astro_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_astro_year, self.entry_astro_month, self.entry_astro_day =
self.create_ymd_row(frame_astro_date, curr_y, curr_m_np, curr_d_np)

frame_astro_loc = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
frame_astro_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_astro_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_astro_lat = self.create_input_row(frame_astro_loc, "Lat (Lintang):", "-7.0667")
self.entry_astro_lon = self.create_input_row(frame_astro_loc, "Lon (Bujur):", "110.4100")
self.entry_astro_elev = self.create_input_row(frame_astro_loc, "Elevasi (m):", "230.0")
self.entry_astro_tz = self.create_input_row(frame_astro_loc, "Timezone:", "7.0")

# ---> TAMBAHAN LABEL CUACA 31 <---
self.lbl_cuaca_astro = ctk.CTkLabel(frame_astro_loc, text="Cuaca: Menunggu...", font=("Consolas",
10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_astro.pack(anchor="w", padx=10, pady=(0, 5))

# ---> TAMBAHAN LABEL CUACA 31 <---
self.lbl_cuaca_astro = ctk.CTkLabel(frame_astro_loc, text="Cuaca: Menunggu...", font=("Consolas",
10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_astro.pack(anchor="w", padx=10, pady=(0, 5))

# =====
# ---> SISIPAN BARU: UI PREDIKTOR OKULTASI (ASTROFOTOGRAFI)
# =====
frame_astro_occ = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
frame_astro_occ.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_astro_occ, text="🔭 PREDIKTOR OKULTASI PLANET", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

```

```

self.combo_astro_occ_target = ctk.CTkOptionMenu(frame_astro_occ, values=["Venus", "Mars",
"Jupiter", "Saturn", "Mercury", "Uranus", "Neptune"])
self.combo_astro_occ_target.pack(fill="x", padx=10, pady=(0, 10))

self.btn_cari_occ = ctk.CTkButton(frame_astro_occ, text="Cari Okultasi Bulan Ini", font=("Segoe UI",
12, "bold"), fg_color="#673AB7", hover_color="#512DA8", command=self.calculate_occultation)
self.btn_cari_occ.pack(fill="x", padx=10, pady=(0, 10))
# =====

# =====
# -- 32. PASANG SURUT GRAVITASI --
# =====
self.frame_pasang_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_pasang_date = ctk.CTkFrame(self.frame_pasang_inputs, fg_color="#212121")
frame_pasang_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_pasang_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_pasang_year, self.entry_pasang_month, self.entry_pasang_day =
self.create_ymd_row(frame_pasang_date, curr_y, curr_m_np, curr_d_np)

frame_pasang_loc = ctk.CTkFrame(self.frame_pasang_inputs, fg_color="#212121")
frame_pasang_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_pasang_loc, text="KOORDINAT PANTAI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_pasang_lat = self.create_input_row(frame_pasang_loc, "Lat (Lintang):", "-7.0667")
self.entry_pasang_lon = self.create_input_row(frame_pasang_loc, "Lon (Bujur):", "110.4100")
self.entry_pasang_tz = self.create_input_row(frame_pasang_loc, "Timezone:", "7.0")

# ---> TAMBAHAN LABEL CUACA 32 <---
self.lbl_cuaca_pasang = ctk.CTkLabel(frame_pasang_loc, text="Cuaca: Menunggu...",
font=("Consolas", 10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_pasang.pack(anchor="w", padx=10, pady=(0, 5))

# =====
# -- 33. EVALUASI FAJAR SHADIQ & SQM --
# =====
self.frame_fajar_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_fajar_date = ctk.CTkFrame(self.frame_fajar_inputs, fg_color="#212121")
frame_fajar_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_fajar_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_fajar_year, self.entry_fajar_month, self.entry_fajar_day =
self.create_ymd_row(frame_fajar_date, curr_y, curr_m_np, curr_d_np)

frame_fajar_loc = ctk.CTkFrame(self.frame_fajar_inputs, fg_color="#212121")

```



```

frame_fajar_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_fajar_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_fajar_lat = self.create_input_row(frame_fajar_loc, "Lat (Lintang):", "-7.0667")
self.entry_fajar_lon = self.create_input_row(frame_fajar_loc, "Lon (Bujur):", "110.4100")
self.entry_fajar_tz = self.create_input_row(frame_fajar_loc, "Timezone:", "7.0")

# ---> TAMBAHAN LABEL CUACA 33 <---
self.lbl_cuaca_fajar = ctk.CTkLabel(frame_fajar_loc, text="Cuaca: Menunggu...", font=("Consolas",
10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_fajar.pack(anchor="w", padx=10, pady=(0, 5))

# -- 34. ANALEMMA MATAHARI --
self.frame_analemma_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_analemma_date = ctk.CTkFrame(self.frame_analemma_inputs, fg_color="#212121")
frame_analemma_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_analemma_date, text="TAHUN & JAM ANALISIS", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# Baris Tahun
self.entry_analemma_year = self.create_input_row(frame_analemma_date, "Tahun (YYYY):",
curr_y)
# Baris Jam (Lokal)
self.entry_analemma_hour = self.create_input_row(frame_analemma_date, "Jam (0-23):", "12.0")

frame_analemma_loc = ctk.CTkFrame(self.frame_analemma_inputs, fg_color="#212121")
frame_analemma_loc.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_analemma_loc, text="KOORDINAT PENGAMAT", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
self.entry_analemma_lat = self.create_input_row(frame_analemma_loc, "Lat (Lintang):", "-7.0667")
self.entry_analemma_lon = self.create_input_row(frame_analemma_loc, "Lon (Bujur):",
"110.4100")
self.entry_analemma_tz = self.create_input_row(frame_analemma_loc, "Timezone:", "7.0")

# ---> LABEL CUACA MODUL 34 (WAJIB ADA) <---
self.lbl_cuaca_analemma = ctk.CTkLabel(frame_analemma_loc, text="Cuaca: Menunggu...",
font=("Consolas", 10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_analemma.pack(anchor="w", padx=10, pady=(0, 5))
# =====

# --- Tombol Aksi Utama ---
self.btn_hitung = ctk.CTkButton(self.sidebar, text="▶ PROSES DATA", font=("Segoe UI", 14,
"bold"), height=45, fg_color="#1565C0", hover_color="#0D47A1", command=self.run_calculation,
state="disabled")
self.btn_hitung.pack(fill="x", padx=15, pady=(20, 10))

```

```

self.lbl_status = ctk.CTkLabel(self.sidebar, text="Memuat Ephemeris...", font=("Consolas", 11),
text_color="#FFAB40")
self.lbl_status.pack(pady=5)

# ===== MAIN AREA (KANAN) =====
self.main_frame = ctk.CTkFrame(self, fg_color="transparent")
self.main_frame.grid(row=0, column=1, sticky="nsew", padx=20, pady=20)
self.main_frame.grid_rowconfigure(1, weight=1)
self.main_frame.grid_columnconfigure(0, weight=1)

header_frame = ctk.CTkFrame(self.main_frame, fg_color="transparent")
header_frame.grid(row=0, column=0, sticky="ew", pady=(0, 10))
header_frame.grid_columnconfigure(0, weight=1)
header_frame.grid_columnconfigure(1, weight=1)
header_frame.grid_columnconfigure(2, weight=1)

title_frame = ctk.CTkFrame(header_frame, fg_color="transparent")
title_frame.grid(row=0, column=0, sticky="w")

self.btn_toggle = ctk.CTkButton(title_frame, text="☰", font=("Segoe UI", 22), width=40, height=40,
fg_color="#1F1F1F", hover_color="#333333", text_color="#00E5FF", command=self.toggle_sidebar)
self.btn_toggle.pack(side="left", padx=(0, 15))

title_text_frame = ctk.CTkFrame(title_frame, fg_color="transparent")
title_text_frame.pack(side="left", fill="y")

self.lbl_main_title = ctk.CTkLabel(title_text_frame, text="Laporan Analisis Hilal & KHGT",
font=("Segoe UI", 20, "bold"))
self.lbl_main_title.pack(anchor="w")

self.lbl_countdown = ctk.CTkLabel(title_text_frame, text="Alarm Salat: Memuat Jadwal...",
font=("Consolas", 12, "bold"), text_color="#FFAB40")
self.lbl_countdown.pack(anchor="w", pady=(2, 0))

# ---> TAMBAHAN WIDGET CUACA <---
self.lbl_status_cuaca = ctk.CTkLabel(title_text_frame, text="Cuaca: Menunggu deteksi...",
font=("Consolas", 11, "bold"), text_color="#9E9E9E")
self.lbl_status_cuaca.pack(anchor="w", pady=(2, 0))
# -----

self.f_cal = ctk.CTkFrame(header_frame, fg_color="#1A1A1A", corner_radius=8, border_width=1,
border_color="#333")
self.f_cal.grid(row=0, column=1, sticky="n")

self.lbl_cal_masehi = ctk.CTkLabel(self.f_cal, text="", font=("Segoe UI", 13, "bold"),
text_color="#00E5FF")
self.lbl_cal_masehi.pack(side="left", padx=(15, 5), pady=4)

```

```

lbl_sep = ctk.CTkLabel(self.f_cal, text="|", font=("Segoe UI", 13, "bold"), text_color="#555")
lbl_sep.pack(side="left", pady=4)

self.lbl_cal_hijri = ctk.CTkLabel(self.f_cal, text="", font=("Georgia", 14, "italic"),
text_color="#FFD54F")
self.lbl_cal_hijri.pack(side="left", padx=(5, 15), pady=4)

btn_frame = ctk.CTkFrame(header_frame, fg_color="transparent")
btn_frame.grid(row=0, column=2, sticky="e")

self.btn_simpan = ctk.CTkButton(btn_frame, text="💾", font=("Segoe UI", 18), width=35,
height=35, fg_color="#2E7D32", hover_color="#1B5E20", command=self.save_to_txt)
self.btn_simpan.pack(side="left", padx=(0, 8))
CTkToolTip(self.btn_simpan, delay=0.5, message="Simpan Data ke TXT")

self.btn_png = ctk.CTkButton(btn_frame, text="🖨️", font=("Segoe UI", 16), width=35, height=35,
fg_color="#F57C00", hover_color="#EF6C00", command=self.export_to_png)
self.btn_png.pack(side="left", padx=(0, 8))
CTkToolTip(self.btn_png, delay=0.5, message="Simpan Laporan sbg PNG")

self.btn_pdf = ctk.CTkButton(btn_frame, text="📄", font=("Segoe UI", 16), width=35, height=35,
fg_color="#1976D2", hover_color="#1565C0", command=self.export_to_pdf)
self.btn_pdf.pack(side="left", padx=(0, 8))
CTkToolTip(self.btn_pdf, delay=0.5, message="Simpan sebagai PDF")

# ---> SISIPKAN KODE TOMBOL CSV DI SINI <---
self.btn_csv = ctk.CTkButton(btn_frame, text="📊", font=("Segoe UI", 16), width=35, height=35,
fg_color="#00695C", hover_color="#004D40", command=self.export_to_csv)
self.btn_csv.pack(side="left", padx=(0, 8))
CTkToolTip(self.btn_csv, delay=0.5, message="Simpan ke CSV (Excel / SPSS)")
# -----

self.btn_home = ctk.CTkButton(btn_frame, text="🏠", font=("Segoe UI", 18), width=35, height=35,
fg_color="#E65100", hover_color="#BF360C", command=self.show_release_info)
self.btn_home.pack(side="left", padx=(0, 8))
CTkToolTip(self.btn_home, delay=0.5, message="Kembali ke Beranda / Info Rilis")

# Output Container Utama (Text)
self.textbox = ctk.CTkTextbox(self.main_frame, font=("Consolas", 13), fg_color="#101010",
text_color="#00E5FF", wrap="none")
self.textbox.grid(row=1, column=0, sticky="nsew")
self.textbox.insert("1.0", "Menunggu input dari pengguna...")
self.textbox.configure(state="disabled")
self.install_textbox_language_hooks()

self.setup_gerhana_out_frame()

```

```

self.setup_animasi_out_frame()
self.setup_3d_out_frame()

self.setup_kriteria_batas_out_frame()
self.setup_analemma_out_frame()
self.setup_kurva_alaska_out_frame() # <--- TAMBAHKAN BARIS INI

# ---> TAMBAHKAN 1 BARIS INI (RUMAH KHUSUS UNTUK GRAFIK CHART) <---
self.frame_chart_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")

# =====
# 1. PINDAHKAN INISIALISASI KALENDER KE SINI
# (Wajib dibuat sebelum switch_mode dipanggil!)
# =====

# -- KALENDER HIJRIAH INPUTS (MENU 26) --
self.frame_kalender_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
kal_info = ctk.CTkFrame(self.frame_kalender_inputs, fg_color="#212121")
kal_info.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(kal_info, text="PENGATURAN KALENDER", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# Tambahan Dropdown Bulan dan Entry Tahun (Hijriah)
row_h_bln = ctk.CTkFrame(kal_info, fg_color="transparent")
row_h_bln.pack(fill="x", padx=10, pady=5)
ctk.CTkLabel(row_h_bln, text="Bulan:", font=("Segoe UI", 12)).pack(side="left")
self.combo_kal_h_bulan = ctk.CTkComboBox(row_h_bln, values=BULAN_HIJRIAH, width=120)
self.combo_kal_h_bulan.pack(side="right")

row_h_thn = ctk.CTkFrame(kal_info, fg_color="transparent")
row_h_thn.pack(fill="x", padx=10, pady=5)
ctk.CTkLabel(row_h_thn, text="Tahun (H):", font=("Segoe UI", 12)).pack(side="left")
self.entry_kal_h_tahun = ctk.CTkEntry(row_h_thn, width=80, justify="center")
self.entry_kal_h_tahun.pack(side="right")

ctk.CTkButton(kal_info, text="🔍 Terapkan Pencarian", fg_color="#1565C0",
hover_color="#0D47A1", command=self.apply_kalender_kustom).pack(fill="x", padx=10, pady=(10, 5))
ctk.CTkButton(kal_info, text="🔄 Reset ke Bulan Ini", fg_color="#00695C", hover_color="#004D40",
command=self.reset_kalender).pack(fill="x", padx=10, pady=(0, 10))

# -- KALENDER MASEHI INPUTS (MENU 27) --
self.frame_kalmasehi_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
kalm_info = ctk.CTkFrame(self.frame_kalmasehi_inputs, fg_color="#212121")
kalm_info.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(kalm_info, text="PENGATURAN KAL. MASEHI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

```

```

# Tambahan Dropdown Bulan dan Entry Tahun (Masehi)
row_m_bln = ctk.CTkFrame(kalm_info, fg_color="transparent")
row_m_bln.pack(fill="x", padx=10, pady=5)
ctk.CTkLabel(row_m_bln, text="Bulan:", font=("Segoe UI", 12)).pack(side="left")
self.combo_kal_m_bulan = ctk.CTkComboBox(row_m_bln, values=BULAN_MASEHI, width=120)
self.combo_kal_m_bulan.pack(side="right")

row_m_thn = ctk.CTkFrame(kalm_info, fg_color="transparent")
row_m_thn.pack(fill="x", padx=10, pady=5)
ctk.CTkLabel(row_m_thn, text="Tahun (M):", font=("Segoe UI", 12)).pack(side="left")
self.entry_kal_m_tahun = ctk.CTkEntry(row_m_thn, width=80, justify="center")
self.entry_kal_m_tahun.pack(side="right")

ctk.CTkButton(kalm_info, text="🔍 Terapkan Pencarian", fg_color="#1565C0",
hover_color="#0D47A1", command=self.apply_kalmasehi_kustom).pack(fill="x", padx=10, pady=(10, 5))
ctk.CTkButton(kalm_info, text="🔄 Reset ke Bulan Ini", fg_color="#00695C",
hover_color="#004D40", command=self.reset_kalmasehi).pack(fill="x", padx=10, pady=(0, 10))

# Render dan Siapkan Frame Output Kalender
self.reset_kalender(update_ui=False)
self.setup_kalender_out_frame()
self.reset_kalmasehi(update_ui=False)
self.setup_kalmasehi_out_frame()

# Setup Frame Output Mode 19 (Matplotlib 3D Ephemeris)
self.setup_eph3d_out_frame()
# =====

# ---> TAMBAHKAN SETUP WINAI DI SINI <---
self.setup_winai_ui()
# =====

# 2. BARU PANGGIL SWITCH MODE & STARTUP LAINNYA DI SINI
on_menu_click("1) Visibility Hilal (KHGT)")
self.show_release_info()
# Simpan teks asli seluruh widget sidebar setelah semua frame input dibuat.
# Ini membuat klik ID/EN/AR setelah memilih menu 1–41 selalu dapat
# menerjemahkan sidebar kiri dari sumber Indonesia asli, bukan dari teks
# yang sedang tampil atau dari hasil terjemahan sebelumnya.
self.capture_sidebar_original_texts()
self.force_update_sidebar_language()
self.update_ui_texts(self)
self.install_secondary_output_textbox_hooks()
self.apply_language_direction()
# Startup tetap menampilkan INFORMASI RILIS. Karena pengguna belum
# memilih menu 1–41 secara manual, klik ID/EN/AR pada kondisi awal
# harus hanya menerjemahkan ruang kanan dari cache JSON rilis.

```

```

self.user_has_selected_main_menu = False

self.sidebar_visible = False
self.sidebar.grid_remove()

self.anim_running = False
self.is_viewing_3d = False

# ---> TAMBAHKAN BARIS INI UNTUK MEMICU CUACA SAAT STARTUP <---
self.on_city_change(self.opt_city.get())
# -----

def _setup_other_inputs(self, curr_y, curr_m, curr_d, curr_m_np, curr_d_np):
    # Meringkas pembuatan UI master yang panjang untuk mode 2 sampai 18
    # -- MOONPHASE --
    self.frame_moon_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
    frame_myyear = ctk.CTkFrame(self.frame_moon_inputs, fg_color="#212121")
    frame_myyear.pack(fill="x", padx=15, pady=10)
    ctk.CTkLabel(frame_myyear, text="TAHUN MASEHI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(10, 5))
    self.entry_moon_year = ctk.CTkEntry(frame_myyear, width=100, justify="center")
    self.entry_moon_year.insert(0, curr_y)
    self.entry_moon_year.pack(padx=10, pady=(0, 15))

    # -- EPHEMERIS --
    self.frame_eph_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
    frame_eph_cfg = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
    frame_eph_cfg.pack(fill="x", padx=15, pady=5)
    row_obj = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
    row_obj.pack(fill="x", padx=10, pady=2)
    ctk.CTkLabel(row_obj, text="Object:", font=("Segoe UI", 12)).pack(side="left")
    self.combo_eph_obj = ctk.CTkOptionMenu(row_obj, values=["Sun", "Moon"], width=100)
    self.combo_eph_obj.pack(side="right")
    row_tref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
    row_tref.pack(fill="x", padx=10, pady=2)
    ctk.CTkLabel(row_tref, text="Time Ref:", font=("Segoe UI", 12)).pack(side="left")
    self.combo_eph_tref = ctk.CTkOptionMenu(row_tref, values=["Local (UT1)", "Local (TDT)"],
width=100)
    self.combo_eph_tref.pack(side="right")
    row_lref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
    row_lref.pack(fill="x", padx=10, pady=2)
    ctk.CTkLabel(row_lref, text="Location:", font=("Segoe UI", 12)).pack(side="left")
    self.combo_eph_lref = ctk.CTkOptionMenu(row_lref, values=["Topocentric", "Geocentric"],
width=100)
    self.combo_eph_lref.pack(side="right")

    # --- WAKTU AWAL (START) ---
    frame_eph_start = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")

```

```

frame_eph_start.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_eph_start, text="WAKTU AWAL (START)", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# Grid Tanggal
self.entry_eph_sy, self.entry_eph_smon, self.entry_eph_sd =
self.create_ymd_row(frame_eph_start, curr_y, curr_m_np, curr_d_np)

# Grid Waktu (HH:MM:SS)
start_hms_grid = ctk.CTkFrame(frame_eph_start, fg_color="transparent")
start_hms_grid.pack(fill="x", padx=10, pady=(0, 10))

ctk.CTkLabel(start_hms_grid, text="hh", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(start_hms_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(start_hms_grid, text="ss", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

self.entry_eph_sh = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="hh",
justify="center")
self.entry_eph_sh.insert(0, "22")
self.entry_eph_sh.grid(row=1, column=0, padx=(0, 5))

self.entry_eph_smin = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_eph_smin.insert(0, "0")
self.entry_eph_smin.grid(row=1, column=1, padx=5)

self.entry_eph_ssec = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="ss",
justify="center")
self.entry_eph_ssec.insert(0, "0")
self.entry_eph_ssec.grid(row=1, column=2, padx=(5, 0))

# --- WAKTU AKHIR (END) ---
frame_eph_end = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
frame_eph_end.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_eph_end, text="WAKTU AKHIR (END)", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# Grid Tanggal
self.entry_eph_ey, self.entry_eph_emon, self.entry_eph_ed =
self.create_ymd_row(frame_eph_end, curr_y, curr_m_np, curr_d_np)

# Grid Waktu (HH:MM:SS)
end_hms_grid = ctk.CTkFrame(frame_eph_end, fg_color="transparent")
end_hms_grid.pack(fill="x", padx=10, pady=(0, 10))

```



```

        ctk.CTkLabel(end_hms_grid, text="hh", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
        ctk.CTkLabel(end_hms_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
        ctk.CTkLabel(end_hms_grid, text="ss", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

        self.entry_eph_eh = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="hh",
justify="center")
        self.entry_eph_eh.insert(0, "22")
        self.entry_eph_eh.grid(row=1, column=0, padx=(0, 5))

        self.entry_eph_emin = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="mm",
justify="center")
        self.entry_eph_emin.insert(0, "0")
        self.entry_eph_emin.grid(row=1, column=1, padx=5)

        self.entry_eph_esec = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="ss",
justify="center")
        self.entry_eph_esec.insert(0, "0")
        self.entry_eph_esec.grid(row=1, column=2, padx=(5, 0))

        # --- LOKASI & INTERVAL ---
        frame_eph_step = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
        frame_eph_step.pack(fill="x", padx=15, pady=5)

        row_inc = ctk.CTkFrame(frame_eph_step, fg_color="transparent")
        row_inc.pack(fill="x", padx=10, pady=(10, 2))
        self.entry_eph_step = ctk.CTkEntry(row_inc, width=50, justify="center")
        self.entry_eph_step.insert(0, "1")
        self.entry_eph_step.pack(side="left")
        self.combo_eph_step = ctk.CTkOptionMenu(row_inc, values=["Day", "Hour", "Minute", "Second"],
width=80)
        self.combo_eph_step.pack(side="left", padx=5)

        self.entry_eph_lat = self.create_input_row(frame_eph_step, "Lat:", "-7.0667")
        self.entry_eph_lon = self.create_input_row(frame_eph_step, "Lon:", "110.4100")
        self.entry_eph_elev = self.create_input_row(frame_eph_step, "Elev:", "230.0")
        self.entry_eph_tz = self.create_input_row(frame_eph_step, "TZ:", "7.0")

        # -- QIBLAH DIR --
        self.frame_qiblah_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_qdate = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
        frame_qdate.pack(fill="x", padx=15, pady=5)
        ctk.CTkLabel(frame_qdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

        # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---

```

```

qdate_grid = ctk.CTkFrame(frame_qdate, fg_color="transparent")
qdate_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(qdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(qdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(qdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_qday = ctk.CTkEntry(qdate_grid, width=45, placeholder_text="dd", justify="center")
self.entry_qday.insert(0, curr_d)
self.entry_qday.grid(row=1, column=0, padx=(0, 5))

self.entry_qmonth = ctk.CTkEntry(qdate_grid, width=45, placeholder_text="mm", justify="center")
self.entry_qmonth.insert(0, curr_m)
self.entry_qmonth.grid(row=1, column=1, padx=5)

self.entry_qyear = ctk.CTkEntry(qdate_grid, width=70, placeholder_text="yyyy", justify="center")
self.entry_qyear.insert(0, curr_y)
self.entry_qyear.grid(row=1, column=2, padx=(5, 0))

frame_qloc = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
frame_qloc.pack(fill="x", padx=15, pady=5)
self.entry_qlat = self.create_input_row(frame_qloc, "Lat (Lintang):", "-7.0667")
self.entry_qlon = self.create_input_row(frame_qloc, "Lon (Bujur):", "110.4100")
self.entry_qtz = self.create_input_row(frame_qloc, "Timezone:", "7.0")
frame_qmode = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
frame_qmode.pack(fill="x", padx=15, pady=5)
self.radio_qiblah_var = ctk.StringVar(value="sun")
ctk.CTkRadioButton(frame_qmode, text="Sun is at Qiblah direction", variable=self.radio_qiblah_var,
value="sun").pack(anchor="w", padx=15, pady=5)
ctk.CTkRadioButton(frame_qmode, text="Sun's shadow is at Qiblah",
variable=self.radio_qiblah_var, value="shadow").pack(anchor="w", padx=15, pady=(5, 10))
self.btn_qmap = ctk.CTkButton(self.frame_qiblah_inputs, text="🗺 Show Qiblah Map",
fg_color="#00695C", hover_color="#004D40", command=self.show_qiblah_map)
self.btn_qiblat_online = ctk.CTkButton(
    self.frame_qiblah_inputs,
    text="🌐 Akses Kiblat Online",
    fg_color="#1565C0",
    hover_color="#0D47A1",
    command=self.open_kiblat_online
)

# =====

```

```

# ---> SISIPAN BARU: UI CELESTIAL QIBLAH (MENU 11)
# =====
frame_cel_q = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
frame_cel_q.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_cel_q, text="🌌 CELESTIAL QIBLAH (BINTANG)", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# Daftar bintang navigasi populer
self.combo_q_star = ctk.CTkOptionMenu(frame_cel_q, values=[
    "Canopus (Suhail)", "Sirius (Asy-Syira)", "Rigel (Orion)",
    "Betelgeuse (Orion)", "Acrux (Crux/Salib Sel.)", "Vega",
    "Altair", "Antares", "Spica"
])
self.combo_q_star.pack(fill="x", padx=10, pady=(0, 10))

self.btn_calc_cel_q = ctk.CTkButton(frame_cel_q, text="Cari Waktu Bintang di Kiblat",
    font=("Segoe UI", 12, "bold"), fg_color="#673AB7",
    hover_color="#512DA8", command=self.calculate_celestial_qiblah)
self.btn_calc_cel_q.pack(fill="x", padx=10, pady=(0, 10))
self.btn_qmap.pack(fill="x", padx=15, pady=(10, 5))
self.btn_qiblat_online.pack(fill="x", padx=15, pady=(5, 10))

# -- MOON TIMES --
self.frame_mt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_mtdate = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
frame_mtdate.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_mtdate, text="BULAN OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
mt_date_grid = ctk.CTkFrame(frame_mtdate, fg_color="transparent")
mt_date_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(mt_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(mt_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))

# Baris 1: Kotak Input (Perhitungan sebulan penuh)
self.entry_mt_month = ctk.CTkEntry(mt_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_mt_month.insert(0, curr_m)
self.entry_mt_month.grid(row=1, column=0, padx=(0, 5))

self.entry_mt_year = ctk.CTkEntry(mt_date_grid, width=70, placeholder_text="yyyy",
justify="center")

```

```

self.entry_mt_year.insert(0, curr_y)
self.entry_mt_year.grid(row=1, column=1, padx=(5, 0))

frame_mtloc = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
frame_mtloc.pack(fill="x", padx=15, pady=5)
self.entry_mt_lat = self.create_input_row(frame_mtloc, "Lat (Lintang):", "-7.0667")
self.entry_mt_lon = self.create_input_row(frame_mtloc, "Lon (Bujur):", "110.4100")
self.entry_mt_elev = self.create_input_row(frame_mtloc, "Elevasi (m):", "230.0")
self.entry_mt_tz = self.create_input_row(frame_mtloc, "Timezone:", "7.0")

# -- SUN TIMES --
self.frame_st_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_stdate = ctk.CTkFrame(self.frame_st_inputs, fg_color="#212121")
frame_stdate.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_stdate, text="BULAN OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
st_date_grid = ctk.CTkFrame(frame_stdate, fg_color="transparent")
st_date_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(st_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(st_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))

# Baris 1: Kotak Input (Perhitungan sebulan penuh)
self.entry_st_month = ctk.CTkEntry(st_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_st_month.insert(0, curr_m)
self.entry_st_month.grid(row=1, column=0, padx=(0, 5))

self.entry_st_year = ctk.CTkEntry(st_date_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_st_year.insert(0, curr_y)
self.entry_st_year.grid(row=1, column=1, padx=(5, 0))

frame_stloc = ctk.CTkFrame(self.frame_st_inputs, fg_color="#212121")
frame_stloc.pack(fill="x", padx=15, pady=5)
self.entry_st_lat = self.create_input_row(frame_stloc, "Lat (Lintang):", "-7.0667")
self.entry_st_lon = self.create_input_row(frame_stloc, "Lon (Bujur):", "110.4100")
self.entry_st_elev = self.create_input_row(frame_stloc, "Elevasi (m):", "230.0")
self.entry_st_tz = self.create_input_row(frame_stloc, "Timezone:", "7.0")

# -- PRAYER TIMES --
self.frame_pt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_pt_mode = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")

```

```

frame_pt_mode.pack(fill="x", padx=15, pady=5)
self.combo_pt_mode = ctk.CTkOptionMenu(frame_pt_mode, values=["Bulanan (1 Bulan Penuh)",
"Harian (1 Hari)"])
self.combo_pt_mode.pack(fill="x", padx=10, pady=(0, 10))

frame_ptdate = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
frame_ptdate.pack(fill="x", padx=15, pady=5)

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
ptdate_grid = ctk.CTkFrame(frame_ptdate, fg_color="transparent")
ptdate_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(ptdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(ptdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(ptdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_pt_day = ctk.CTkEntry(ptdate_grid, width=45, placeholder_text="dd", justify="center")
self.entry_pt_day.insert(0, curr_d)
self.entry_pt_day.grid(row=1, column=0, padx=(0, 5))

self.entry_pt_month = ctk.CTkEntry(ptdate_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_pt_month.insert(0, curr_m)
self.entry_pt_month.grid(row=1, column=1, padx=5)

self.entry_pt_year = ctk.CTkEntry(ptdate_grid, width=70, placeholder_text="yyyy", justify="center")
self.entry_pt_year.insert(0, curr_y)
self.entry_pt_year.grid(row=1, column=2, padx=(5, 0))

frame_ptloc = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
self.entry_pt_lat = self.create_input_row(frame_ptloc, "Lat (Lintang):", "-7.0667")
self.entry_pt_lon = self.create_input_row(frame_ptloc, "Lon (Bujur):", "110.4100")
self.entry_pt_elev = self.create_input_row(frame_ptloc, "Elevasi (m):", "230.0")
self.entry_pt_tz = self.create_input_row(frame_ptloc, "Timezone:", "7.0")

frame_ptatm = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
frame_ptatm.pack(fill="x", padx=15, pady=5)
self.entry_pt_temp = self.create_input_row(frame_ptatm, "Suhu (°C):", "10.0")
self.entry_pt_pres = self.create_input_row(frame_ptatm, "Tekanan (mb):", "1010.0")
self.entry_pt_hum = self.create_input_row(frame_ptatm, "Kelembapan (%)ate", "60.0")
frame_pt_method = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
frame_pt_method.pack(fill="x", padx=15, pady=5)

```

```

self.combo_pt_mazhab = ctk.CTkOptionMenu(frame_pt_method, values=["Standard (Syafi'i, Maliki, Hanbali)", "Hanafi"])
self.combo_pt_mazhab.pack(fill="x", padx=10, pady=(0, 5))
self.combo_pt_method = ctk.CTkOptionMenu(frame_pt_method, values=["MUHAMMADIYAH (Fajr 18°, Isha 18°)", "Kemenag RI (Fajr 20°, Isha 18°)", "Muslim World League (Fajr 18°, Isha 17°)", "ISNA (Fajr 15°, Isha 15°)", "Egypt (Fajr 19.5°, Isha 17.5°)"])
self.combo_pt_method.pack(fill="x", padx=10, pady=(0, 10))

# Penambahan Opsi Metode Lintang Ekstrem
self.combo_pt_highlat = ctk.CTkOptionMenu(
    frame_pt_method,
    values=[
        "Lintang Normal (Abaikan)",
        "Nisf al-Lail (1/2 Malam)",
        "Subeh Sadiq (1/7 Malam)",
        "Proporsi Sudut (Angle-Based)",
        "Aqrab al-Balad (Estimasi 45°)"
    ]
)
self.combo_pt_highlat.pack(fill="x", padx=10, pady=(0, 10))
self.combo_pt_highlat.set("Lintang Normal (Abaikan)")

frame_pt_opt = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
frame_pt_opt.pack(fill="x", padx=15, pady=5)
self.entry_pt_ikhtiyat = self.create_input_row(frame_pt_opt, "Ikhtiyat (Detik):", "16")

row_fmt = ctk.CTkFrame(frame_pt_opt, fg_color="transparent")
row_fmt.pack(fill="x", padx=10, pady=2)
self.combo_pt_fmt = ctk.CTkOptionMenu(row_fmt, values=["HH:MM:SS (Jam:Menit:Detik)", "HH:MM (Jam:Menit)"])
self.combo_pt_fmt.pack(fill="x")

# ---> TAMBAHAN FITUR: TOMBOL IN-FLIGHT PRAYER TIMES <---
self.btn_pt_inflight = ctk.CTkButton(self.frame_pt_inputs, text="✈ Shalat Penerbangan (In-Flight)",
                                     font=("Segoe UI", 12, "bold"), fg_color="#D84315", hover_color="#BF360C",
                                     command=self.buka_kalkulator_penerbangan)
self.btn_pt_inflight.pack(fill="x", padx=15, pady=(15, 5))

# -- VISMAP --
self.frame_vismap_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

# 1. FRAME TANGGAL DENGAN GRID SYSTEM (Agar dd, mm, yyyy presisi)
frame_vmdate = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
frame_vmdate.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_vmdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))

```

```

vmdate_grid = ctk.CTkFrame(frame_vmdate, fg_color="transparent")
vmdate_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(vmdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(vmdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(vmdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_vmday = ctk.CTkEntry(vmdate_grid, width=45, placeholder_text="dd", justify="center")
self.entry_vmday.insert(0, curr_d)
self.entry_vmday.grid(row=1, column=0, padx=(0, 5))

self.entry_vmmmonth = ctk.CTkEntry(vmdate_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_vmmmonth.insert(0, curr_m)
self.entry_vmmmonth.grid(row=1, column=1, padx=5)

self.entry_vmyear = ctk.CTkEntry(vmdate_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_vmyear.insert(0, curr_y)
self.entry_vmyear.grid(row=1, column=2, padx=(5, 0))

# 2. FRAME TOGGLE (Lanjutan kode asli)
frame_vmtoggle = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
frame_vmtoggle.pack(fill="x", padx=15, pady=5)
self.radio_vmphase = ctk.StringVar(value="new")
ctk.CTkRadioButton(frame_vmtoggle, text="New Crescent (Evening)", variable=self.radio_vmphase,
value="new").pack(anchor="w", padx=15, pady=5)
ctk.CTkRadioButton(frame_vmtoggle, text="Old Crescent (Morning)", variable=self.radio_vmphase,
value="old").pack(anchor="w", padx=15, pady=(0, 8))
frame_vmtime = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
frame_vmtime.pack(fill="x", padx=15, pady=5)
self.radio_vmtime = ctk.StringVar(value="sunset")
row_t1 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
row_t1.pack(fill="x", padx=5)
ctk.CTkRadioButton(row_t1, text="Sunset", variable=self.radio_vmtime, value="sunset",
width=80).pack(side="left", padx=5)
ctk.CTkRadioButton(row_t1, text="Moonset", variable=self.radio_vmtime, value="moonset",
width=80).pack(side="left", padx=5)
row_t2 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
row_t2.pack(fill="x", padx=5, pady=5)
ctk.CTkRadioButton(row_t2, text="Best Time", variable=self.radio_vmtime,
value="best").pack(side="left", padx=5)

```



```

frame_vmcrit = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
frame_vmcrit.pack(fill="x", padx=15, pady=5)

# ---> PERBARUI LIST KRITERIA DI SINI <---
daftar_kriteria = [
    "KHGT / Diyanet Turki (Alt>=5, Eln>=8)",
    "Neo MABIMS (Alt>=3, Eln>=6.4)",
    "Yallop Criterion (q-Parameter)",
    "Ilyas (Alt>=4, Eln>=10.5)",
    "Danjon Limit (Eln>=7)",
    "Odeh Criterion"
]
self.combo_vmcrit = ctk.CTkOptionMenu(frame_vmcrit, values=daftar_kriteria)
self.combo_vmcrit.pack(fill="x", padx=10, pady=(0, 10))

frame_vmparam = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
frame_vmparam.pack(fill="x", padx=15, pady=5)

# Ganti dengan opsi Interseksi (Peta KHGT 2D)
self.combo_vmparam = ctk.CTkOptionMenu(frame_vmparam, values=["Kriteria Visibilitas",
"Altitude Bulan", "Elongasi", "Iluminasi", "Interseksi", "Interseksi (Peta KHGT 2D)"])
self.combo_vmparam.pack(fill="x", padx=10, pady=(0, 10))

# -- KONVERSI --
self.frame_conv_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_conv_mode = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
frame_conv_mode.pack(fill="x", padx=15, pady=5)
self.radio_conv_var = ctk.StringVar(value="m2h")

# Tambahan Menu Urfi
ctk.CTkRadioButton(frame_conv_mode, text="Masehi → Hijriah (KHGT)",
variable=self.radio_conv_var, value="m2h", command=self.update_conv_ui).pack(anchor="w",
padx=15, pady=5)
ctk.CTkRadioButton(frame_conv_mode, text="Hijriah (KHGT) → Masehi",
variable=self.radio_conv_var, value="h2m", command=self.update_conv_ui).pack(anchor="w",
padx=15, pady=5)
ctk.CTkRadioButton(frame_conv_mode, text="Masehi → Hijriah (Urfi)",
variable=self.radio_conv_var, value="m2u", command=self.update_conv_ui).pack(anchor="w",
padx=15, pady=5)
ctk.CTkRadioButton(frame_conv_mode, text="Hijriah (Urfi) → Masehi",
variable=self.radio_conv_var, value="u2m", command=self.update_conv_ui).pack(anchor="w",
padx=15, pady=(5, 10))

frame_conv_date = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
frame_conv_date.pack(fill="x", padx=15, pady=5)

```

```

self.lbl_conv_date = ctk.CTkLabel(frame_conv_date, text="TANGGAL MASEHI", font=("Segoe UI",
12, "bold"))
self.lbl_conv_date.pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
convdate_grid = ctk.CTkFrame(frame_conv_date, fg_color="transparent")
convdate_grid.pack(fill="x", padx=10, pady=(0, 10))

days = [str(d).zfill(2) for d in range(1, 32)]

# Baris 0: Label Petunjuk
ctk.CTkLabel(convdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2), sticky="w", padx=(5,0))
ctk.CTkLabel(convdate_grid, text="Bulan", font=("Segoe UI", 10),
text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2), sticky="w", padx=(5,0))
ctk.CTkLabel(convdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2), sticky="w", padx=(5,0))

# Baris 1: Kotak Input (Karena ini dropdown, ukurannya sedikit beda)
self.combo_conv_day = ctk.CTkComboBox(convdate_grid, values=days, width=60)
self.combo_conv_day.set(curr_d)
self.combo_conv_day.grid(row=1, column=0, padx=(0, 5))

self.combo_conv_month = ctk.CTkComboBox(convdate_grid, values=BULAN_MASEHI, width=120)
self.combo_conv_month.set(BULAN_MASEHI[datetime.datetime.now().month - 1])
self.combo_conv_month.grid(row=1, column=1, padx=5)

self.entry_conv_year = ctk.CTkEntry(convdate_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_conv_year.insert(0, curr_y)
self.entry_conv_year.grid(row=1, column=2, padx=(5, 0))

# ---> PERBAIKAN: Menambahkan Label Keterangan Ikhtiyat <---
frame_conv_adj = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
frame_conv_adj.pack(fill="x", padx=15, pady=5)

# Label judul untuk kotak "0"
ctk.CTkLabel(frame_conv_adj, text="KOREKSI HARI (IKHTIYAT)", font=("Segoe UI", 11,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

self.combo_conv_adj = ctk.CTkComboBox(frame_conv_adj, values=["-2", "-1", "0", "+1", "+2"])
self.combo_conv_adj.set("0")
self.combo_conv_adj.pack(fill="x", padx=10, pady=(0, 10))

# -- QIBLA TIME --
self.frame_qt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_qtdate = ctk.CTkFrame(self.frame_qt_inputs, fg_color="#212121")
frame_qtdate.pack(fill="x", padx=15, pady=5)

```

```

        ctk.CTkLabel(frame_qtdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
qt_date_grid = ctk.CTkFrame(frame_qtdate, fg_color="transparent")
qt_date_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(qt_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(qt_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(qt_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_qtday = ctk.CTkEntry(qt_date_grid, width=45, placeholder_text="dd", justify="center")
self.entry_qtday.insert(0, curr_d)
self.entry_qtday.grid(row=1, column=0, padx=(0, 5))

self.entry_qtmonth = ctk.CTkEntry(qt_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_qtmonth.insert(0, curr_m)
self.entry_qtmonth.grid(row=1, column=1, padx=5)

self.entry_qtyear = ctk.CTkEntry(qt_date_grid, width=70, placeholder_text="yyyy", justify="center")
self.entry_qtyear.insert(0, curr_y)
self.entry_qtyear.grid(row=1, column=2, padx=(5, 0))

frame_qtloc = ctk.CTkFrame(self.frame_qt_inputs, fg_color="#212121")
frame_qtloc.pack(fill="x", padx=15, pady=5)
self.entry_qtlat = self.create_input_row(frame_qtloc, "Lat (Lintang):", "-7.0667")
self.entry_qtlon = self.create_input_row(frame_qtloc, "Lon (Bujur):", "110.4100")
self.entry_qttz = self.create_input_row(frame_qtloc, "Timezone:", "7.0")

# -- GERHANA --
self.frame_gerhana_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_gdate = ctk.CTkFrame(self.frame_gerhana_inputs, fg_color="#212121")
frame_gdate.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(frame_gdate, text="PILIH TAHUN GERHANA", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
gerhana_grid = ctk.CTkFrame(frame_gdate, fg_color="transparent")
gerhana_grid.pack(fill="x", padx=10, pady=(0, 10))

ctk.CTkLabel(gerhana_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))

```

```

# ---> UBAH RANGE MENJADI (1, 3000) DI SINI <---
self.combo_tahun_gerhana = ctk.CTkComboBox(gerhana_grid, values=[str(y) for y in range(1,
3000)], justify="center", width=80)
self.combo_tahun_gerhana.set(curr_y)
self.combo_tahun_gerhana.grid(row=1, column=0, padx=(0, 5))

# -- ANIMASI LIVE 2D --
self.frame_animasi_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_anim_date = ctk.CTkFrame(self.frame_animasi_inputs, fg_color="#212121")
frame_anim_date.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(frame_anim_date, text="WAKTU SIMULASI 2D", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
anim_date_grid = ctk.CTkFrame(frame_anim_date, fg_color="transparent")
anim_date_grid.pack(fill="x", padx=10, pady=(0, 5))

# Baris 0: Label Petunjuk
ctk.CTkLabel(anim_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(anim_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(anim_date_grid, text="yyyy", font=("Segoe UI", 10),
text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))

# Baris 1: Kotak Input
self.entry_anim_day = ctk.CTkEntry(anim_date_grid, width=45, placeholder_text="dd",
justify="center")
self.entry_anim_day.insert(0, curr_d)
self.entry_anim_day.grid(row=1, column=0, padx=(0, 5))

self.entry_anim_month = ctk.CTkEntry(anim_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_anim_month.insert(0, curr_m)
self.entry_anim_month.grid(row=1, column=1, padx=5)

self.entry_anim_year = ctk.CTkEntry(anim_date_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_anim_year.insert(0, curr_y)
self.entry_anim_year.grid(row=1, column=2, padx=(5, 0))

# ---> PERBAIKAN: Tambahan Label hh:mm:ss <---
ctk.CTkLabel(frame_anim_date, text="hh:mm:ss", font=("Segoe UI", 10),
text_color="#9E9E9E").pack(anchor="w", padx=12, pady=(5, 0))

self.entry_anim_time = ctk.CTkEntry(frame_anim_date, placeholder_text="HH:MM:SS")

```

```

self.entry_anim_time.insert(0, datetime.datetime.now().strftime("%H:%M:%S"))
self.entry_anim_time.pack(fill="x", padx=10, pady=(0, 10))

frame_anim_btn = ctk.CTkFrame(self.frame_animasi_inputs, fg_color="transparent")
frame_anim_btn.pack(fill="x", padx=15, pady=5)

self.btn_anim_terapkan = ctk.CTkButton(frame_anim_btn, text="Terapkan Waktu Kustom",
fg_color="#1976D2", hover_color="#1565C0", command=self.anim_set_kustom)
self.btn_anim_terapkan.pack(fill="x", pady=5)

self.btn_anim_sunset = ctk.CTkButton(frame_anim_btn, text="🌅 Set ke Waktu Sunset",
fg_color="#F57C00", hover_color="#E65100", command=self.anim_cari_sunset)
self.btn_anim_sunset.pack(fill="x", pady=5)

self.btn_anim_live = ctk.CTkButton(frame_anim_btn, text="🔴 Kembali ke Live (Sekarang)",
fg_color="#2E7D32", hover_color="#1B5E20", command=self.anim_start_live)
self.btn_anim_live.pack(fill="x", pady=5)

self.anim_is_live = True
self.anim_custom_time = None

# -- 3D SYSTEM (SKYFIELD) MENU 16 --
self.frame_3d_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

frame_3d_date = ctk.CTkFrame(self.frame_3d_inputs, fg_color="#212121")
frame_3d_date.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(frame_3d_date, text="SET TANGGAL SIMULASI 3D", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

f3d_row = ctk.CTkFrame(frame_3d_date, fg_color="transparent")
f3d_row.pack(fill="x", padx=10, pady=(0, 10))

# Dropdown Tanggal
self.combo_3d_d = ctk.CTkComboBox(f3d_row, values=[f"{i:02d}" for i in range(1, 32)], width=60,
command=self.on_3d_date_change)
self.combo_3d_d.set(curr_d)
self.combo_3d_d.pack(side="left", padx=2)

# Dropdown Bulan
self.combo_3d_m = ctk.CTkComboBox(f3d_row, values=[f"{i:02d}" for i in range(1, 13)], width=60,
command=self.on_3d_date_change)
self.combo_3d_m.set(curr_m)
self.combo_3d_m.pack(side="left", padx=2)

# Dropdown Tahun
years_range = [str(i) for i in range(1, 3000)]

```

```

self.combo_3d_y = ctk.CTkComboBox(f3d_row, values=years_range, width=85,
command=self.on_3d_date_change)
self.combo_3d_y.set(curr_y)
self.combo_3d_y.pack(side="left", padx=2)

# ---> TAMBAHAN: TOMBOL PROSES KHUSUS MENU 16 <---
self.btn_apply_3d = ctk.CTkButton(self.frame_3d_inputs, text="▶ TERAPKAN TANGGAL",
font=("Segoe UI", 14, "bold"), height=40, fg_color="#1565C0", hover_color="#0D47A1",
command=self.force_update_3d)
self.btn_apply_3d.pack(fill="x", padx=15, pady=(5, 10))

# Info navigasi
ctk.CTkLabel(self.frame_3d_inputs, text="Klik & Drag layar untuk rotasi.\nScroll/Mousewheel untuk
Zoom.",
font=("Segoe UI", 11, "italic"), text_color="#FFD54F").pack(pady=5)

# -- GHA --
self.frame_gha_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
frame_gha_date = ctk.CTkFrame(self.frame_gha_inputs, fg_color="#212121")
frame_gha_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_gha_date, text="TANGGAL REFERENSI (UTC)", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

gha_date_grid = ctk.CTkFrame(frame_gha_date, fg_color="transparent")
gha_date_grid.pack(fill="x", padx=10, pady=(0, 5))

ctk.CTkLabel(gha_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(gha_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(gha_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

self.entry_gha_day = ctk.CTkEntry(gha_date_grid, width=45, placeholder_text="dd",
justify="center")
self.entry_gha_day.insert(0, curr_d)
self.entry_gha_day.grid(row=1, column=0, padx=(0, 5))

self.entry_gha_month = ctk.CTkEntry(gha_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_gha_month.insert(0, curr_m)
self.entry_gha_month.grid(row=1, column=1, padx=5)

self.entry_gha_year = ctk.CTkEntry(gha_date_grid, width=70, placeholder_text="yyyy",
justify="center")

```

```

self.entry_gha_year.insert(0, curr_y)
self.entry_gha_year.grid(row=1, column=2, padx=(5, 0))

ctk.CTkLabel(frame_gha_date, text="hh:mm:ss", font=("Segoe UI", 10),
text_color="#9E9E9E").pack(anchor="w", padx=12, pady=(5, 0))
self.entry_gha_time = ctk.CTkEntry(frame_gha_date, placeholder_text="Waktu HH:MM:SS (Def:
12:00:00)")
self.entry_gha_time.insert(0, "12:00:00")
self.entry_gha_time.pack(fill="x", padx=10, pady=(0, 10))

# ---> TAMBAHAN: PARAMETER & TOMBOL KURVA ALASKA <---
frame_alaska_loc = ctk.CTkFrame(self.frame_gha_inputs, fg_color="#212121", border_width=1,
border_color="#00E5FF")
frame_alaska_loc.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(frame_alaska_loc, text="📍 PARAMETER KURVA ALASKA", font=("Segoe UI", 11,
"bold"), text_color="#00E5FF").pack(anchor="w", padx=10, pady=(8, 2))

self.entry_gha_lat = self.create_input_row(frame_alaska_loc, "Lat:", "56.8135")
self.entry_gha_lon = self.create_input_row(frame_alaska_loc, "Lon:", "-158.8622")
self.entry_gha_tz = self.create_input_row(frame_alaska_loc, "TZ:", "-11.0")

self.btn_kurva_alaska = ctk.CTkButton(frame_alaska_loc, text="GENERATE KURVA ALASKA",
font=("Segoe UI", 12, "bold"), fg_color="#00C853", hover_color="#00E676",
command=self.proses_kurva_alaska)
self.btn_kurva_alaska.pack(fill="x", padx=10, pady=(10, 10))

# -- ALT CHART --
self.frame_chart_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

# --- A. GRAFIK HARIAN (24 JAM) ---
c_date = ctk.CTkFrame(self.frame_chart_inputs, fg_color="#212121")
c_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(c_date, text="GRAFIK HARIAN (24 JAM)", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

chart_date_grid = ctk.CTkFrame(c_date, fg_color="transparent")
chart_date_grid.pack(fill="x", padx=10, pady=(0, 10))

ctk.CTkLabel(chart_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(chart_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(chart_date_grid, text="yyyy", font=("Segoe UI", 10),
text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))

self.entry_chart_d = ctk.CTkEntry(chart_date_grid, width=45, placeholder_text="dd",
justify="center")

```



```

self.entry_chart_d.insert(0, curr_d)
self.entry_chart_d.grid(row=1, column=0, padx=(0, 5))

self.entry_chart_m = ctk.CTkEntry(chart_date_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_chart_m.insert(0, curr_m)
self.entry_chart_m.grid(row=1, column=1, padx=5)

self.entry_chart_y = ctk.CTkEntry(chart_date_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_chart_y.insert(0, curr_y)
self.entry_chart_y.grid(row=1, column=2, padx=(5, 0))

# ---> PERBAIKAN: Menggunakan Grid 2x2 agar ke-4 tombol muat dengan rapi di sidebar <---
frame_chart_btns = ctk.CTkFrame(self.frame_chart_inputs, fg_color="transparent")
frame_chart_btns.pack(fill="x", padx=15, pady=(5, 5))

frame_chart_btns.grid_columnconfigure(0, weight=1)
frame_chart_btns.grid_columnconfigure(1, weight=1)

btn_alt = ctk.CTkButton(frame_chart_btns, text="📈 Alt Chart", fg_color="#1565C0",
hover_color="#0D47A1", command=lambda: self.trigger_chart("altitude"))
btn_alt.grid(row=0, column=0, padx=(0, 5), pady=5, sticky="ew")

btn_elong = ctk.CTkButton(frame_chart_btns, text="📈 Elongasi", fg_color="#2E7D32",
hover_color="#1B5E20", command=lambda: self.trigger_chart("elongation"))
btn_elong.grid(row=0, column=1, padx=(5, 0), pady=5, sticky="ew")

btn_illum = ctk.CTkButton(frame_chart_btns, text="📈 Iluminasi", fg_color="#F57C00",
hover_color="#E65100", command=lambda: self.trigger_chart("illumination"))
btn_illum.grid(row=1, column=0, padx=(0, 5), pady=5, sticky="ew")

btn_il_el = ctk.CTkButton(frame_chart_btns, text="📈 Ill vs Eln", fg_color="#673AB7",
hover_color="#512DA8", command=lambda: self.trigger_chart("illum_vs_elong"))
btn_il_el.grid(row=1, column=1, padx=(5, 0), pady=5, sticky="ew")

# --- B. GRAFIK TREN BULANAN HIJRIAH (RAJAB - ZULHIJAH) ---
frame_chart_hijri = ctk.CTkFrame(self.frame_chart_inputs, fg_color="#212121", border_width=1,
border_color="#D32F2F")
frame_chart_hijri.pack(fill="x", padx=15, pady=(15, 10))
ctk.CTkLabel(frame_chart_hijri, text="TREN HIJRIAH (RAJAB - ZULHIJAH)", font=("Segoe UI", 12,
"bold"), text_color="#FF5252").pack(anchor="w", padx=10, pady=(8, 2))

row_hy = ctk.CTkFrame(frame_chart_hijri, fg_color="transparent")
row_hy.pack(fill="x", padx=10, pady=5)
ctk.CTkLabel(row_hy, text="Tahun Hijriah:", font=("Segoe UI", 12)).pack(side="left")
self.entry_chart_hy = ctk.CTkEntry(row_hy, width=80, justify="center")

```

```

self.entry_chart_hy.insert(0, "1447")
self.entry_chart_hy.pack(side="right")

btn_trend_alt = ctk.CTkButton(frame_chart_hijri, text="🏔️ Tren Altitude (Geo vs Topo)",
fg_color="#D84315", hover_color="#BF360C", command=lambda: self.trigger_trend("altitude"))
btn_trend_alt.pack(fill="x", padx=10, pady=5)

btn_trend_elong = ctk.CTkButton(frame_chart_hijri, text="📏 Tren Elongasi (Geo vs Topo)",
fg_color="#00838F", hover_color="#006064", command=lambda: self.trigger_trend("elongation"))
btn_trend_elong.pack(fill="x", padx=10, pady=5)

btn_trend_illum = ctk.CTkButton(frame_chart_hijri, text="💡 Tren Iluminasi Siklik",
fg_color="#4A148C", hover_color="#38006B", command=lambda: self.trigger_trend("illumination"))
btn_trend_illum.pack(fill="x", padx=10, pady=(5, 10))

# ---> TOMBOL KE-4 BARU: KOMBINASI <---
btn_trend_all = ctk.CTkButton(frame_chart_hijri, text="🏔️📏💡 Tren Kombinasi (Alt, Eln, Illum)",
fg_color="#B71C1C", hover_color="#7F0000", command=lambda: self.trigger_trend("kombinasi"))
btn_trend_all.pack(fill="x", padx=10, pady=(5, 10))

# -- FIRST POINT --
self.frame_first_point_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
fp_date = ctk.CTkFrame(self.frame_first_point_inputs, fg_color="#212121")
fp_date.pack(fill="x", padx=15, pady=10)

ctk.CTkLabel(fp_date, text="TANGGAL PENCARIAN", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
fp_inputs_grid = ctk.CTkFrame(fp_date, fg_color="transparent")
fp_inputs_grid.pack(fill="x", padx=10, pady=(0, 10))

# Membuat list untuk Tanggal, Bulan, dan Tahun
days_list = [str(d).zfill(2) for d in range(1, 32)]
months_list = [str(m).zfill(2) for m in range(1, 13)]

# ---> UBAH RANGE MENJADI (1, 3000) DI SINI <---
years_list = [str(y) for y in range(1, 3000)] # Range tahun 1 s/d 2999 M

# Baris 0: Label Petunjuk
ctk.CTkLabel(fp_inputs_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(fp_inputs_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
ctk.CTkLabel(fp_inputs_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

```

```

# Baris 1: Kotak Input (Dropdown)
self.entry_fp_d = ctk.CTkComboBox(fp_inputs_grid, values=days_list, width=65)
self.entry_fp_d.set(curr_d) # Otomatis set ke tanggal hari ini
self.entry_fp_d.grid(row=1, column=0, padx=(0, 5))

self.entry_fp_m = ctk.CTkComboBox(fp_inputs_grid, values=months_list, width=65)
self.entry_fp_m.set(curr_m) # Otomatis set ke bulan hari ini
self.entry_fp_m.grid(row=1, column=1, padx=5)

self.entry_fp_y = ctk.CTkComboBox(fp_inputs_grid, values=years_list, width=80)
self.entry_fp_y.set(curr_y) # Otomatis set ke tahun hari ini
self.entry_fp_y.grid(row=1, column=2, padx=(5, 0))

# -- SEASON (EQUINOX & SOLSTICE) --
self.frame_season_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_s_year = ctk.CTkFrame(self.frame_season_inputs, fg_color="#212121")
frame_s_year.pack(fill="x", padx=15, pady=10)
ctk.CTkLabel(frame_s_year, text="TAHUN OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
season_grid = ctk.CTkFrame(frame_s_year, fg_color="transparent")
season_grid.pack(fill="x", padx=10, pady=(0, 10))

ctk.CTkLabel(season_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))

self.entry_season_year = ctk.CTkEntry(season_grid, width=70, placeholder_text="yyyy",
justify="center")
self.entry_season_year.insert(0, curr_y)
self.entry_season_year.grid(row=1, column=0, padx=(0, 5))

# -- PLANETARY --
self.frame_planet_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
frame_pl_date = ctk.CTkFrame(self.frame_planet_inputs, fg_color="#212121")
frame_pl_date.pack(fill="x", padx=15, pady=5)
ctk.CTkLabel(frame_pl_date, text="BULAN OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))

# --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
planet_grid = ctk.CTkFrame(frame_pl_date, fg_color="transparent")
planet_grid.pack(fill="x", padx=10, pady=(0, 10))

# Baris 0: Label Petunjuk
ctk.CTkLabel(planet_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
ctk.CTkLabel(planet_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))

```

```

# Baris 1: Kotak Input
self.entry_pl_month = ctk.CTkEntry(planet_grid, width=45, placeholder_text="mm",
justify="center")
self.entry_pl_month.insert(0, curr_m)
self.entry_pl_month.grid(row=1, column=0, padx=(0, 5))

self.entry_pl_year = ctk.CTkEntry(planet_grid, width=70, placeholder_text="yyyy", justify="center")
self.entry_pl_year.insert(0, curr_y)
self.entry_pl_year.grid(row=1, column=1, padx=(5, 0))

self.combo_pl_target = ctk.CTkOptionMenu(frame_pl_date, values=["Mercury", "Venus", "Mars",
"Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"])
self.combo_pl_target.pack(fill="x", padx=10, pady=(0, 10))

def on_3d_date_change(self, event=None):
    """
    Fungsi ini dipanggil setiap kali dropdown tanggal/bulan/tahun diubah.
    Memicu pembaruan instan pada animasi 3D di ruang kanan.
    """
    if self.is_viewing_3d:
        self.update_3d_animation()

def anim_start_live(self):
    self.anim_is_live = True
    now = datetime.datetime.now()
    self.entry_anim_year.delete(0, 'end'); self.entry_anim_year.insert(0, str(now.year))
    self.entry_anim_month.delete(0, 'end'); self.entry_anim_month.insert(0, f'{now.month:02d}')
    self.entry_anim_day.delete(0, 'end'); self.entry_anim_day.insert(0, f'{now.day:02d}')
    self.entry_anim_time.delete(0, 'end')
    messagebox.showinfo("Live Mode", "Simulasi dikembalikan ke Waktu Real-Time saat ini.")

def anim_set_kustom(self):
    self.anim_is_live = False
    try:
        y = int(self.entry_anim_year.get())
        m = int(self.entry_anim_month.get())
        d = int(self.entry_anim_day.get())
        time_str = self.entry_anim_time.get().strip()

        if not time_str:
            time_str = "12:00:00"
            self.entry_anim_time.insert(0, time_str)

        parts = time_str.split(":")
        jam = int(parts[0])
        menit = int(parts[1]) if len(parts) > 1 else 0

```

```

detik = int(parts[2]) if len(parts) > 2 else 0

# Ambil timezone berdasarkan bujur
try: lon_val = float(self.entry_vlon.get())
except: lon_val = 110.4100
tz_offset = int(self.get_tz_from_lon(lon_val))

dt_lokal = datetime.datetime(y, m, d, jam, menit, detik)
dt_utc = dt_lokal - datetime.timedelta(hours=tz_offset)

self.anim_custom_time = self.ts.from_datetime(dt_utc.replace(tzinfo=pytz.utc))
except Exception as e:
    messagebox.showerror("Error Input", f"Format waktu/tanggal tidak valid!\nError: {e}")
    self.anim_start_live()

def create_eph3d_slider(self, parent, label_text, from_, to_, valinit, res=1):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)

    ctk.CTkLabel(row, text=label_text, width=80, anchor="w", font=("Segoe UI", 12)).pack(side="left")

    # Inilah label yang akan kita kontrol
    val_label = ctk.CTkLabel(row, text=str(valinit), width=40, anchor="e", font=("Consolas", 12))

    var = ctk.DoubleVar(value=valinit)

    def on_slide(val):
        fmt = "{:.0f}" if res >= 1 else "{:.2f}"
        val_label.configure(text=fmt.format(val))
        if not getattr(self, 'eph3d_updating', False):
            self.eph3d_is_live = False
            self.eph3d_update_plot()

    slider = ctk.CTkSlider(row, from_=from_, to=to_, variable=var, number_of_steps=int((to_ -
from_)/res), command=on_slide)
    slider.pack(side="left", fill="x", expand=True, padx=(5, 10))
    val_label.pack(side="right")

    # PERBAIKAN: Kembalikan val_label, BUKAN slider
    return var, val_label

def setup_eph3d_out_frame(self):
    # MENGGUNAKAN CtkScrollableFrame agar konten di ruang kanan bisa di-scroll
    self.frame_eph3d_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#000000",
label_text="VISUALISASI EPHEMERIS 3D")

```

```

# Panel Info Teks (Diletakkan di atas)
self.lbl_eph3d_info = ctk.CTkLabel(self.frame_eph3d_out, text="Memuat data...",
                                   font=("Courier", 13), text_color="#00E676",
                                   justify="left", anchor="nw")
self.lbl_eph3d_info.pack(fill="x", padx=15, pady=10)

# Setup Figure Matplotlib
plt.style.use('dark_background')
# Ukuran figsize disesuaikan agar proporsional saat di-scroll
self.fig_eph3d = plt.figure(figsize=(8, 8), facecolor='#000000')
self.ax_eph3d = self.fig_eph3d.add_subplot(111, projection='3d')
self.fig_eph3d.subplots_adjust(left=0, right=1, bottom=0, top=1)

self.canvas_eph3d = FigureCanvasTkAgg(self.fig_eph3d, master=self.frame_eph3d_out)
canvas_widget = self.canvas_eph3d.get_tk_widget()
canvas_widget.pack(fill="x", expand=True, padx=10, pady=10)

# Inisialisasi Objek Statis 3D
u = np.linspace(0, 2 * np.pi, 60)
v = np.linspace(0, np.pi/2, 30)
self.ax_eph3d.plot_wireframe(10 * np.outer(np.cos(u), np.sin(v)),
                             10 * np.outer(np.sin(u), np.sin(v)),
                             10 * np.outer(np.ones(np.size(u)), np.cos(v)),
                             color='gray', alpha=0.1)

xx, yy = np.meshgrid(np.linspace(-10, 10, 2), np.linspace(-10, 10, 2))
self.ax_eph3d.plot_surface(xx, yy, np.zeros_like(xx), color='green', alpha=0.3)

self.ax_eph3d.text(0, 11, 0, 'Utara', color='white', ha='center', fontsize=12)
self.ax_eph3d.text(0, -11, 0, 'Selatan', color='white', ha='center', fontsize=12)
self.ax_eph3d.text(11, 0, 0, 'Timur', color='white', ha='center', fontsize=12)
self.ax_eph3d.text(-11, 0, 0, 'Barat', color='white', ha='center', fontsize=12)

# Objek Dinamis
self.titik_matahari, = self.ax_eph3d.plot([], [], [], 'o', color='yellow', markersize=15)
self.titik_bulan, = self.ax_eph3d.plot([], [], [], 'o', color='white', markersize=10)
self.garis_matahari, = self.ax_eph3d.plot([], [], [], color='yellow', linestyle='--', alpha=0.5)
self.garis_bulan, = self.ax_eph3d.plot([], [], [], color='white', linestyle='--', alpha=0.5)

self.ax_eph3d.set_xlim([-10, 10])
self.ax_eph3d.set_ylim([-10, 10])
self.ax_eph3d.set_zlim([-2, 10])
self.ax_eph3d.set_axis_off()

self.canvas_eph3d.draw()

def eph3d_r_ke_xyz(self, alt, az, r=10):
    # PERBAIKAN: Wajib konversi derajat ke radian agar trigonometri tidak kacau

```

```

import math
alt_rad = math.radians(alt)
az_rad = math.radians(az)

x = r * math.sin(az_rad) * math.cos(alt_rad)
y = r * math.cos(az_rad) * math.cos(alt_rad)
z = r * math.sin(alt_rad)
return x, y, z

def eph3d_update_plot(self):
    try:
        # 1. Ambil nilai Input dari UI
        lat_val = float(self.entry_eph3d_lat.get())
        lon_val = float(self.entry_eph3d_lon.get())
        elev_val = float(self.var_eph3d_elev.get())

        # --- PERBAIKAN: Fallback Timezone Lapis Ganda ---
        try:
            tz_offset = float(self.entry_eph3d_tz.get())
        except ValueError:
            try: tz_offset = float(self.entry_vtz.get())
            except ValueError: tz_offset = 7.0 # Default WIB

        y = int(self.var_eph3d_tahun.get())
        m = int(self.var_eph3d_bulan.get())
        jam_desimal = self.var_eph3d_jam.get()

        max_d = calendar.monthrange(y, m)[1]
        d = min(int(self.var_eph3d_hari.get()), max_d)

        # --- PERBAIKAN: Rekonstruksi Waktu Presisi Tinggi ---
        total_seconds = int(round(jam_desimal * 3600))
        jam = total_seconds // 3600
        menit = (total_seconds % 3600) // 60
        detik = total_seconds % 60

        if jam >= 24:
            jam = 23; menit = 59; detik = 59

        waktu_lokal = datetime.datetime(y, m, d, jam, menit, detik)
        waktu_utc = waktu_lokal - datetime.timedelta(hours=tz_offset)

        # Buat Objek Waktu Skyfield
        t_skyfield = self.ts.utc(waktu_utc.year, waktu_utc.month, waktu_utc.day,
                                   waktu_utc.hour, waktu_utc.minute, waktu_utc.second)

        bumi = self.eph['earth']
        matahari = self.eph['sun']

```



```

bulan = self.eph['moon']

# Setup Lokasi Pengamat dengan WGS84 Skyfield
lokasi_obs = bumi + wgs84.latlon(lat_val, lon_val, elevation_m=elev_val)

# 2. PERHITUNGAN TOPOSENTRIK MURNI (SKYFIELD)
app_sun_topo = lokasi_obs.at(t_skyfield).observe(matahari).apparent()
app_moon_topo = lokasi_obs.at(t_skyfield).observe(bulan).apparent()

alt_s_topo_obj, az_s_topo_obj, _ = app_sun_topo.altaz(temperature_C=25,
pressure_mbar=1010)
alt_m_topo_obj, az_m_topo_obj, _ = app_moon_topo.altaz(temperature_C=25,
pressure_mbar=1010)

alt_topo_sun = alt_s_topo_obj.degrees
az_topo_sun = az_s_topo_obj.degrees
alt_topo_moon = alt_m_topo_obj.degrees
az_topo_moon = az_m_topo_obj.degrees

elong_topo = app_sun_topo.separation_from(app_moon_topo).degrees

# Fraksi Iluminasi Fase Bulan
illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_skyfield)
iluminasi_fase = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0

# 3. PERHITUNGAN GEOSENTRIK (SKYFIELD)
sun_geo = bumi.at(t_skyfield).observe(matahari).apparent()
moon_geo = bumi.at(t_skyfield).observe(bulan).apparent()

elong_geo = sun_geo.separation_from(moon_geo).degrees

ra_sun, dec_sun, _ = sun_geo.radec()
ra_moon, dec_moon, _ = moon_geo.radec()

gast = t_skyfield.gast
lst = gast + (lon_val / 15.0)
lat_rad = math.radians(lat_val)

HA_sun_rad = math.radians((lst - ra_sun.hours) * 15.0)
sin_alt_geo_sun = math.sin(lat_rad) * math.sin(dec_sun.radians) + math.cos(lat_rad) *
math.cos(dec_sun.radians) * math.cos(HA_sun_rad)
alt_geo_sun = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_sun))))

HA_moon_rad = math.radians((lst - ra_moon.hours) * 15.0)
sin_alt_geo_moon = math.sin(lat_rad) * math.sin(dec_moon.radians) + math.cos(lat_rad) *
math.cos(dec_moon.radians) * math.cos(HA_moon_rad)
alt_geo_moon = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_moon))))

```

```

# 4. PLOTTING 3D MATPLOTLIB & UPDATE INFO UI
dip_deg = (1.76 * math.sqrt(max(0, elev_val))) / 60.0

# Terapkan posisi toposentrik ke Plot 3D (Sekarang sudah memakan Radian)
sx, sy, sz = self.eph3d_r_ke_xyz(alt_topo_sun, az_topo_sun)
mx, my, mz = self.eph3d_r_ke_xyz(alt_topo_moon, az_topo_moon)

self.titik_matahari.set_data([sx], [sy])
self.titik_matahari.set_3d_properties([sz])
self.garis_matahari.set_data([0, sx], [0, sy])
self.garis_matahari.set_3d_properties([0, sz])

self.titik_bulan.set_data([mx], [my])
self.titik_bulan.set_3d_properties([mz])
self.garis_bulan.set_data([0, mx], [0, my])
self.garis_bulan.set_3d_properties([0, mz])

status_waktu = "🔴 SEDANG LIVE" if getattr(self, 'eph3d_is_live', False) else "🔴 WAKTU
KUSTOM"
tz_str = f"UTC+{tz_offset}" if tz_offset >= 0 else f"UTC{tz_offset}"

try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov}"
except:
    lokasi_text = "CUSTOM LOCATION"

teks = (
    f"LOKASI AKTIF: {lokasi_text}\n"
    f"Lat: {lat_val} | Lon: {lon_val} | Elevasi: {elev_val}m\n"
    f"WAKTU LOKAL ({tz_str}): \n"
    f"{waktu_lokal.strftime('%d-%b-%Y %H:%M:%S')} - {status_waktu}\n\n"
    f"UFUK MAR'I (Batas Dip): -{dip_deg:.3f}°\n"
    f"[MATAHARI]\n"
    f"└─ Toposentrik -> Tinggi: {alt_topo_sun:>6.2f}° | Azimuth: {az_topo_sun:>6.2f}°\n"
    f"└─ Geosentrik -> Tinggi: {alt_geo_sun:>6.2f}°\n"
    f"[BULAN / HILAL]\n"
    f"└─ Toposentrik -> Tinggi: {alt_topo_moon:>6.2f}° | Azimuth: {az_topo_moon:>6.2f}°\n"
    f"└─ Geosentrik -> Tinggi: {alt_geo_moon:>6.2f}°\n"
    f"[RELASI & KHGT]\n"
    f"└─ Elongasi Toposentrik : {elong_topo:>6.2f}°\n"
    f"└─ Elongasi Geosentrik : {elong_geo:>6.2f}°\n"
    f"└─ Fase Bulan (Illuminasi): {iluminasi_fase:>5.1f}%\n"
)

self.lbl_eph3d_info.configure(text=teks)

```

```

self.canvas_eph3d.draw_idle()

except Exception as e:
    import traceback
    print(f"Error 3D Plot: {e}\n{traceback.format_exc()}")

def eph3d_cari_sunset(self):
    self.eph3d_is_live = False
    try:
        lat_val = float(self.entry_eph3d_lat.get())
        lon_val = float(self.entry_eph3d_lon.get())
        elev_val = float(self.var_eph3d_elev.get())

        # --- PERBAIKAN: Ambil TZ dari UI ---
        try:
            tz_offset = float(self.entry_eph3d_tz.get())
        except:
            tz_offset = float(self.entry_vtz.get())

        y = int(self.var_eph3d_tahun.get())
        m = int(self.var_eph3d_bulan.get())
        d = min(int(self.var_eph3d_hari.get()), calendar.monthrange(y, m)[1])

        # Setup parameter Skyfield
        loc = wgs84.latlon(lat_val, lon_val, elevation_m=elev_val)
        t0 = self.ts.utc(y, m, d, -tz_offset)
        t1 = self.ts.utc(y, m, d, 24 - tz_offset)

        # Deteksi Sunset menggunakan Skyfield
        f_rs = almanac.sunrise_sunset(self.eph, loc)
        t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

        waktu_sunset_lokal = None
        for t_ev, y_ev in zip(np.atleast_1d(t_rs), np.atleast_1d(y_rs)):
            if y_ev == 0: # 0 berarti Sunset
                dt_utc = t_ev.utc_datetime()
                waktu_sunset_lokal = dt_utc + datetime.timedelta(hours=tz_offset)
                break

        if waktu_sunset_lokal:
            jam_desimal = waktu_sunset_lokal.hour + (waktu_sunset_lokal.minute / 60.0) +
(waktu_sunset_lokal.second / 3600.0)

            self.eph3d_updating = True
            self.var_eph3d_tahun.set(waktu_sunset_lokal.year)
            self.var_eph3d_bulan.set(waktu_sunset_lokal.month)
            self.var_eph3d_hari.set(waktu_sunset_lokal.day)
            self.var_eph3d_jam.set(jam_desimal)

```

```

self.lbl_eph3d_tahun.configure(text=str(waktu_sunset_lokal.year))
self.lbl_eph3d_bulan.configure(text=str(waktu_sunset_lokal.month))
self.lbl_eph3d_hari.configure(text=str(waktu_sunset_lokal.day))
self.lbl_eph3d_jam.configure(text=f"{jam_desimal:.2f}")

self.eph3d_updating = False
self.eph3d_update_plot()
else:
    messagebox.showwarning("Peringatan", "Matahari tidak terbenam pada hari/lokasi tersebut (Anomali Kutub).")

except Exception as e:
    print(f"Gagal mencari sunset 3D: {e}")

def eph3d_start_live(self):
    self.eph3d_is_live = True
    self.eph3d_update_plot() # Panggil 1x update plot seketika (mencegah delay)

def eph3d_live_update_loop(self):
    # 1. Cek apakah menu yang aktif benar-benar Menu 19
    current_mode = self.combo_mode.get()

    if "Simulasi Ephemeris 3D" in current_mode and getattr(self, 'eph3d_is_live', False):
        self.eph3d_updating = True
        try:
            # --- PERBAIKAN: Dapatkan Timezone yang presisi dari UI ---
            try:
                tz_offset = float(self.entry_eph3d_tz.get())
            except ValueError:
                try: tz_offset = float(self.entry_vtz.get())
            except ValueError: tz_offset = 7.0

            # Gunakan UTC murni untuk menghindari limitasi tahun 1-9999 pada datetime
            now_utc = datetime.datetime.now(datetime.timezone.utc).replace(tzinfo=None)
            now_target = now_utc + datetime.timedelta(hours=tz_offset)

            # Update slider/variable tanpa memicu event 'command' slider agar tidak rekursif
            self.var_eph3d_tahun.set(now_target.year)
            self.var_eph3d_bulan.set(now_target.month)
            self.var_eph3d_hari.set(now_target.day)

            jam_des = now_target.hour + (now_target.minute / 60.0) + (now_target.second / 3600.0)
            self.var_eph3d_jam.set(jam_des)

            # Update label teks pendamping slider
            self.lbl_eph3d_tahun.configure(text=str(now_target.year))
            self.lbl_eph3d_bulan.configure(text=str(now_target.month))

```

```

self.lbl_eph3d_hari.configure(text=str(now_target.day))
self.lbl_eph3d_jam.configure(text=f"{jam_des:.2f}")

# Jalankan render plot
self.eph3d_update_plot()
except Exception as e:
    print(f"Silent error in 3D loop: {e}")
finally:
    self.eph3d_updating = False

# 2. Pemanggilan ulang loop (HANYA satu kali self.after)
self.after(1000, self.eph3d_live_update_loop)

def switch_mode(self, mode):
    # Setiap pemilihan menu membersihkan halaman muka/laporan sebelumnya.
    # Untuk modul berbasis tombol, ruang kanan menampilkan petunjuk PROSES DATA.
    self.prepare_module_right_output()

    # 1. Pastikan tombol PROSES DATA muncul kembali saat pindah menu
    self.btn_hitung.pack(fill="x", padx=15, pady=(20, 10))

    # 2. Sembunyikan semua input module di sidebar
    for f in [self.frame_vis_inputs, self.frame_moon_inputs, self.frame_eph_inputs,
self.frame_qiblah_inputs,
            self.frame_mt_inputs, self.frame_st_inputs, self.frame_pt_inputs, self.frame_vismap_inputs,
            self.frame_conv_inputs, self.frame_qt_inputs, self.frame_gerhana_inputs,
self.frame_animasi_inputs,
            self.frame_3d_inputs, self.frame_gha_inputs, self.frame_chart_inputs,
self.frame_first_point_inputs,
            self.frame_season_inputs, self.frame_planet_inputs, self.frame_eph3d_inputs,
self.frame_kalender_inputs,
            self.frame_kalmasehi_inputs, self.frame_winai_inputs, self.frame_autobuild_inputs,
self.frame_mizwala_inputs,
            self.frame_kriteria_batas_inputs,
            self.frame_astro_inputs, self.frame_pasang_inputs, self.frame_fajar_inputs,
self.frame_analemma_inputs]:
        f.pack_forget()

    # 3. Sembunyikan semua output module di ruang kanan secara aman
    self.textbox.grid_remove()

    if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
    if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
    if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
    if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
    if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
    if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
    if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()

```

```

if hasattr(self, 'frame_winai_out'): self.frame_winai_out.grid_remove()
if hasattr(self, 'frame_kriteria_batas_out'): self.frame_kriteria_batas_out.grid_remove()
if hasattr(self, 'frame_analemma_out'): self.frame_analemma_out.grid_remove()
if hasattr(self, 'frame_alaska_out'): self.frame_alaska_out.grid_remove() # <--- TAMBAHKAN INI

self.anim_running = False
self.is_viewing_3d = False

# Bersihkan frame chart agar tidak menumpuk
if hasattr(self, 'frame_chart_out'):
    for widget in self.frame_chart_out.winfo_children():
        widget.destroy()

# 4. LOGIKA PERGANTIAN MENU
if "Peta Langit Dunia" in mode:
    self.lbl_main_title.configure(text=self.tr("Peta Langit Dunia"))
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E5FF", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert(
        "1.0",
        self.tr("Membuka Peta Langit Dunia melalui browser default...") + "\n\n" +
        WORLD_SKY_MAP_URL
    )
    self.textbox.configure(state="disabled")
    self.btn_hitung.pack_forget()
    self.after(100, self.open_peta_langit_dunia)

elif "Laporan Astronomi" in mode:
    self.lbl_main_title.configure(text="Laporan Astronomi")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E5FF", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert(
        "1.0",
        "Membuka aplikasi Laporan Astronomi...\n\n"
        "Pastikan file 'LaporanAstronomi.exe' berada di folder yang sama "
        "dengan aplikasi KHGT Times."
    )
    self.textbox.configure(state="disabled")
    self.btn_hitung.pack_forget()
    self.after(100, self.open_laporan_astronomi)

elif "Visibility Hilal" in mode:
    self.frame_vis_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Crescent Analysis & KHGT")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")

```

```

elif "Visibility Map" in mode:
    self.frame_vismap_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Crescent Visibility Map Scanner")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik 'PROSES DATA' untuk melakukan pemindaian peta global
HD. \nProses ini menghitung data spasial dan melakukan interpolasi bicubic. Harap tunggu...")
    self.textbox.configure(state="disabled")
elif "Analisis Hilal Global" in mode:
    self.frame_gha_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Global Hilal Visibility Analyzer")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E676")
elif "Moonphase" in mode:
    self.frame_moon_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Moonphase Cycle Data")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E5FF")
elif "Sun Moon Ephemeris" in mode:
    self.frame_eph_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Sun and Moon Ephemeris Data")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")
elif "Qiblah Direction" in mode:
    self.frame_qiblah_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Qiblah Direction & Alignment Times")
    self.textbox.grid(row=1, column=0, sticky="nsew") # Menampilkan box teks laporan
    self.textbox.configure(text_color="#FFD54F")
elif "Moon Times" in mode:
    self.frame_mt_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Moon Rise, Transit, and Set Times")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")
elif "Sun Times" in mode:
    self.frame_st_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Sun Rise, Transit, and Set Times")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")
elif "Prayer Times" in mode:
    self.frame_pt_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Islamic Prayer Times & Twilight")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#E0E0E0")
elif "Konversi" in mode:
    self.frame_conv_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Gregorian & KHGT Calendar Conversion")

```



```

        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E676")
    elif "Qibla Time" in mode:
        self.frame_qt_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Rashdul Qiblah Lokal")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#FFD54F")
    elif "Analisis Gerhana" in mode:
        self.frame_gerhana_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Precision Eclipse Calculator")
        self.frame_gerhana_out.grid(row=1, column=0, sticky="nsew")
    elif "Live Animasi" in mode:
        self.frame_animasi_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Live Celestial Object Tracker")
        self.frame_animasi_out.grid(row=1, column=0, sticky="nsew")
        self.btn_hitung.pack_forget() # Sembunyikan tombol proses

    # Reset ke waktu berjalan
    self.anim_start_live()

    self.anim_running = True
    self.draw_static_background()
    self.update_animation()
    elif "Sistem Sun Moon" in mode:
        self.frame_3d_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="3D Geocentric Space System")
        self.frame_3d_out.grid(row=1, column=0, sticky="nsew")
        self.btn_hitung.pack_forget()
        self.is_viewing_3d = True
        self.update_3d_animation()
    elif "Altitude Chart" in mode:
        self.frame_chart_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Celestial Altitude Graphics")
        self.textbox.grid_remove()
        self.frame_chart_out.grid(row=1, column=0, sticky="nsew")
        self.show_pending_message_in_frame(self.frame_chart_out)
        self.lbl_status.configure(
            text=self.tr("Klik PROSES DATA untuk render grafik"),
            text_color="#00E5FF"
        )
    elif "Kota Pertama" in mode:
        self.frame_first_point_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="KHGT first Point (Mainland)")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Klik 'PROSES DATA' untuk memulai scanning global mencari titik pertama di daratan utama...")

```

```

        self.textbox.configure(state="disabled")
    elif "Equinox" in mode:
        self.frame_season_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Equinox & Solstice Calculator")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#FFD54F")
    elif "Planetary Times" in mode:
        self.frame_planet_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Rise, Transit & Set Planet")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E5FF")

# ---> INI DIA BLOK MENU 19 YANG SEMPAT TERHAPUS <---
    elif "Simulasi Ephemeris 3D" in mode:
        self.frame_eph3d_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="3D Sun & Moon Ephemeris Simulation")

        self.textbox.grid_remove()
        self.frame_eph3d_out.grid(row=1, column=0, sticky="nsew")

        # Sembunyikan tombol proses karena ini live interaktif
        self.btn_hitung.pack_forget()

        # Langsung jalankan animasi live saat menu diklik
        self.eph3d_start_live()

# ---> URUTAN YANG BENAR (SYAWAL HARUS DI ATAS) <---
    elif "Komparasi 50 Tahun (Syawal)" in mode:
        self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Syawal")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E676", state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai komputasi 50 tahun penentuan 1 Syawal.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
        self.textbox.configure(state="disabled")
        self.frame_eph3d_inputs.pack_forget()

# ---> TAMBAHKAN MENU 35 DI SINI <---
    elif "Komparasi 50 Thn (Zulhijah)" in mode:
        self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Zulhijah")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E676", state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai komputasi 50 tahun penentuan 1 Zulhijah.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
        self.textbox.configure(state="disabled")
        self.frame_eph3d_inputs.pack_forget()

```

```

elif "Komparasi 50 Tahun (12 Bulan)" in mode:
    self.lbl_main_title.configure(text="Analisis Probabilitas & Komparasi Global")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E676", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Klik '  PROSES DATA' untuk memulai audit 12 bulan Hijriah selama 50 tahun.\nSistem akan menghitung probabilitas sinkronisasi KHGT vs MABIMS Sabang.")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

elif "Komparasi 50 Tahun" in mode:
    self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Ramadhan")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#00E676", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai komputasi 50 tahun.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

elif "Tabel Tinggi Hilal 50 Thn (Syawal)" in mode:
    self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Syawal)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Ketinggian 1 Syawal).\nProses ini menggunakan Akselerasi Database...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

# ---> TAMBAHKAN MENU 36 DI SINI <---
elif "Tabel Tinggi Hilal 50 Thn (Zulhijah)" in mode:
    self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Zulhijah)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Ketinggian 1 Zulhijah).\nProses ini menggunakan Akselerasi Database...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

elif "Tabel Tinggi Hilal" in mode:
    self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Ramadhan)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")

```

```

        self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian
global 50 tahun (1 Ramadhan).\nProses ini sangat cepat berkat Akselerasi Database...")
        self.textbox.configure(state="disabled")
        self.frame_eph3d_inputs.pack_forget()

elif "Tabel Elongasi Hilal" in mode:
    self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Ramadhan)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian
global 50 tahun (Fokus Elongasi).\nProses ini sangat cepat berkat Akselerasi Database...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()
# ---> SISIPKAN BLOK MENU 25 DI SINI <---
elif "Tabel Elongasi Hilal 50 Thn (Syawal)" in mode:
    self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Syawal)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian
global 50 tahun (Fokus Elongasi 1 Syawal).\nProses ini menggunakan Akselerasi Database...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

# ---> TAMBAHKAN MENU 37 DI SINI <---
elif "Tabel Elongasi Hilal 50 Thn (Zulhijah)" in mode:
    self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Zulhijah)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Silakan klik tombol '  PROSES DATA' untuk memulai pemindaian
global 50 tahun (Fokus Elongasi 1 Zulhijah).\nProses ini menggunakan Akselerasi Database...")
    self.textbox.configure(state="disabled")
    self.frame_eph3d_inputs.pack_forget()

# Pastikan ini berada di bawah Menu 25
elif "Tabel Elongasi Hilal" in mode:
    self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Ramadhan)")

elif "Kalender Hijriah" in mode:
    self.frame_kalender_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Kalender Hijriah Global Tunggal")

    self.textbox.grid_remove()
    self.frame_kalender_out.grid(row=1, column=0, sticky="nsew")

```

```

# Sembunyikan tombol proses karena kalender me-render dirinya sendiri secara instan
self.btn_hitung.pack_forget()
self.render_kalender()

elif "Kalender Masehi" in mode:
    self.frame_kalmasehi_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Kalender Masehi Terintegrasi KHGT")

    self.textbox.grid_remove()
    self.frame_kalmasehi_out.grid(row=1, column=0, sticky="nsew")

    # Sembunyikan tombol proses karena dirender instan
    self.btn_hitung.pack_forget()
    self.render_kalmasehi()

# ---> TAMBAHKAN BLOK WINAI INI <---
elif "WinAI" in mode:
    self.frame_winai_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="WinAI: Aplikasi AI Windows")

    self.textbox.grid_remove()
    self.frame_winai_out.grid(row=1, column=0, sticky="nsew")
    self.btn_hitung.pack_forget()

# ---> INJEKSI OTOMATIS TANGGAL & LOKASI KE LAYAR <---
loc_prov = self.opt_prov.get()
loc_city = self.opt_city.get()
now = datetime.datetime.now()
tgl_str = now.strftime("%d %B %Y")

pesan_konteks = (f"Parameter Sistem Dimuat: Anda berada di {loc_city}, {loc_prov}.\n"
                 f"Tanggal Saat Ini: {tgl_str}.\n"
                 f"💡 Tips: Anda bisa langsung mengetik 'hitung visibilitas hilal hari ini' atau 'cari kota"
perama KHGT' "
                 f"untuk menjalankan kalkulasi KHGT secara instan.")

self.winai_output_box.configure(state="normal")
# ---> PERBAIKAN: Ubah tag dari "info" menjadi "bold" agar hitam pekat dan tebal <---
self.winai_output_box.insert("end", f"💡 Sistem:\n{pesan_konteks}\n\n", "bold")
self.winai_output_box.see("end")
self.winai_output_box.configure(state="disabled")

elif "Mizwala" in mode:
    self.frame_mizwala_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Kalkulator Bayangan Tongkat Istiwa (Mizwala)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F")

```

```

elif "Auto-Builder" in mode:
    self.frame_autobuild_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="HIJRI_DB Auto-Builder (Generator Siklus KHGT)")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", " ⚙️ TOOL DEVELOPER / ADMIN ⚙️ \n\nKlik '▶ PROSES DATA' untuk
me-run iterasi engine Ephemeris NASA.\nSistem akan merekonstruksi ulang penanggalan Kalender
Hijriah Global Tunggal (KHGT) untuk rentang tahun yang Anda input, kemudian menuliskannya secara
otomatis ke dalam file 'db_hijriah.json'.\n\n(Peringatan: Pemrosesan 100 Tahun akan memakan waktu
beberapa saat).")
    self.textbox.configure(state="disabled")

elif "Status Kriteria Batas" in mode:
    self.frame_kriteria_batas_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Simulasi Interaktif: Anatomi Kriteria KHGT 5-8")
    self.frame_kriteria_batas_out.grid(row=1, column=0, sticky="nsew")

    # HAPUS BARIS INI -> self.btn_hitung.pack_forget()
    # Biarkan tombol hitung tetap muncul!

    self.update_kriteria_batas_plot() # Panggil plot pertama kali

elif "Astrofotografi" in mode:
    self.frame_astro_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Astrofotografi Hilal & Visibilitas Kontras")
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.textbox.configure(text_color="#FFD54F", state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", "Klik '▶ PROSES DATA' untuk menghitung Limit Danjon dan Jendela
Kontras Astrofotografi...")
    self.textbox.configure(state="disabled")

elif "Pasang Surut" in mode:
    self.frame_pasang_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Prediktor Pasang Surut Gravitasi")
    self.textbox.grid(row=1, column=0, sticky="nsew")

elif "Fajar Shadiq" in mode:
    self.frame_fajar_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Evaluasi Fajar Shadiq & SQM Teoritis")
    self.textbox.grid(row=1, column=0, sticky="nsew")

elif "Analemma" in mode:
    self.frame_analemma_inputs.pack(fill="x", before=self.btn_hitung)
    self.lbl_main_title.configure(text="Generator Analemma Matahari")

```

```

self.textbox.grid_remove() # Sembunyikan textbox default
self.frame_analemma_out.grid(row=1, column=0, sticky="nsew") # Tampilkan Scrollable Frame
Baru

```

```

#elif "Analemma Matahari" in mode:
# self.clear_main_frame()

# Container Utama
container = ctk.CTkFrame(self.main_frame, fg_color="transparent")
container.pack(fill="both", expand=True, padx=20, pady=20)

# --- SIDEBAR INPUT (KIRI) ---
input_frame = ctk.CTkFrame(container, width=300)
input_frame.pack(side="left", fill="y", padx=(0, 20))

ctk.CTkLabel(input_frame, text="PENGATURAN ANALEMMA", font=ctk.CTkFont(size=16,
weight="bold")).pack(pady=15)

# Input Tahun & Jam
self.entry_analemma_year = self.create_input(input_frame, "Tahun Analisis:",
str(datetime.datetime.now().year))
self.entry_analemma_hour = self.create_input(input_frame, "Jam Pengamatan (Lokal):", "12")

# Input Lokasi (Akan otomatis terisi oleh sistem sinkronisasi)
self.entry_analemma_lat = self.create_input(input_frame, "Lintang (Latitude):",
self.entry_vlat.get())
self.entry_analemma_lon = self.create_input(input_frame, "Bujur (Longitude):",
self.entry_vlon.get())
# Cari bagian pembentukan sidebar Menu 34 di setup_ui
self.entry_analemma_tz = self.create_input_row(frame_analemma_loc, "Timezone:", "7.0")

# Tambahkan label cuaca khusus menu 34 di sini
self.lbl_cuaca_analemma = ctk.CTkLabel(frame_analemma_loc, text="Cuaca: Menunggu
deteksi...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
self.lbl_cuaca_analemma.pack(anchor="w", padx=10, pady=(0, 5))

btn_proses = ctk.CTkButton(input_frame, text="PROSES ANALEMMA", fg_color="#FF6D00",
hover_color="#E65100",
command=self.calculate_analemma)
btn_proses.pack(pady=20, padx=20, fill="x")

# --- AREA HASIL (KANAN) ---
result_frame = ctk.CTkFrame(container)
result_frame.pack(side="right", fill="both", expand=True)

self.txt_analemma_report = ctk.CTkTextbox(result_frame, font=ctk.CTkFont(family="Courier",
size=12))
self.txt_analemma_report.pack(fill="both", expand=True, padx=10, pady=10)

```

```

# Widget yang baru ditampilkan/dibuat langsung mengikuti bahasa aktif.
self.install_secondary_output_textbox_hooks()
self.force_update_sidebar_language()
self.update_ui_texts(self)
self.apply_language_direction()

# =====
# LOGIKA MODUL 08: ALARM & NOTIFIKASI SALAT
# =====
def on_alarm_toggle(self):
    if self.alarm_enabled.get():
        self.lbl_countdown.configure(text="Menginisiasi Jadwal Alarm...", text_color="#00E676")
        self._is_calculating_alarm = True
        threading.Thread(target=self.update_silent_schedule, daemon=True).start()
    else:
        self.lbl_countdown.configure(text="Alarm Salat: Dinonaktifkan", text_color="#FF5252")
        if HAS_PYGAME:
            try: pygame.mixer.music.stop()
            except: pass

def pilih_file_audio(self):
    filepath = filedialog.askopenfilename(title="Pilih Audio Adzan", filetypes=[("Audio Files", "*.mp3
*.wav")])
    if filepath:
        self.adzan_audio_path.set(filepath)
        filename = os.path.basename(filepath)
        if len(filename) > 20: filename = filename[:17] + "..."
        self.lbl_audio_path.configure(text=filename)

def update_silent_schedule(self):
    """Menghitung jadwal salat harian tanpa menampilkan ke layar utama"""
    if not self.eph:
        self.after(0, lambda: self.lbl_countdown.configure(text="Menunggu Ephemeris...",
text_color="#FFAB40"))
        self._is_calculating_alarm = False
        return

    try:
        # --- PERBAIKAN 1: Sinkronisasi Zona Waktu Independen ---
        try: tz_offset = float(self.entry_pt_tz.get())
        except: tz_offset = 7.0

        tz_info = datetime.timezone(datetime.timedelta(hours=tz_offset))
        now_utc = datetime.datetime.now(datetime.timezone.utc)
        now_local = now_utc.astimezone(tz_info).replace(tzinfo=None)

        waktu_kalkulasi = now_local

```



```

# Jika dipanggil otomatis saat jadwal hari ini habis, gunakan tanggal besok
if getattr(self, '_force_tomorrow', False):
    waktu_kalkulasi = now_local + datetime.timedelta(days=1)
    self._force_tomorrow = False

year, month, day = waktu_kalkulasi.year, waktu_kalkulasi.month, waktu_kalkulasi.day
# -----

try:
    lat = float(self.entry_pt_lat.get())
    lon = float(self.entry_pt_lon.get())
    elev = float(self.entry_pt_elev.get())
except:
    lat, lon, elev = -7.0667, 110.4100, 230.0

method_val = self.combo_pt_method.get()
if "Kemenag" in method_val:
    fajr_angle, isha_angle = -20.0, -18.0
elif "Muslim World League" in method_val:
    fajr_angle, isha_angle = -18.0, -17.0
elif "ISNA" in method_val:
    fajr_angle, isha_angle = -15.0, -15.0
elif "Egypt" in method_val:
    fajr_angle, isha_angle = -19.5, -17.5
else:
    fajr_angle, isha_angle = -18.0, -18.0

asr_factor = 2.0 if "Hanafi" in self.combo_pt_mazhab.get() else 1.0
try: ikhtiyat_sec = int(self.entry_pt_ikhtiyat.get())
except: ikhtiyat_sec = 16

earth = self.eph['earth']
sun = self.eph['sun']
loc = wgs84.latlon(lat, lon, elevation_m=elev)

# --- PERBAIKAN 2: Batas waktu T0 dan T1 yang presisi ---
t0_dt = datetime.datetime(year, month, day, 0, 0, 0, tzinfo=tz_info)
t1_dt = t0_dt + datetime.timedelta(days=1)

t0 = self.ts.from_datetime(t0_dt)
t1 = self.ts.from_datetime(t1_dt)
# -----

tt_array = np.linspace(t0.tt, t1.tt, 2880)
t_search = self.ts.tt_jd(tt_array)

alt_deg = (earth + loc).at(t_search).observe(sun).apparent().altaz()[0].degrees
idx_noon = np.argmax(alt_deg)

```

```

tt_dhohur = tt_array[idx_noon]
alt_noon = alt_deg[idx_noon]

alt_am = alt_deg[:idx_noon]
tt_am = tt_array[:idx_noon]
alt_pm = alt_deg[idx_noon:]
tt_pm = tt_array[idx_noon:]

zenith_noon = 90.0 - alt_noon
shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
shadow_asr = asr_factor + shadow_noon
alt_asr = math.degrees(math.atan(1.0 / shadow_asr))

def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
    diffs = alt_arr - target_alt
    for i in range(len(diffs)-1):
        if direction == 'up' and diffs[i] <= 0 and diffs[i+1] > 0:
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
        elif direction == 'down' and diffs[i] >= 0 and diffs[i+1] < 0:
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
    return None

val_fajr = find_crossing(alt_am, tt_am, fajr_angle, 'up')
val_shuroq = find_crossing(alt_am, tt_am, -0.833, 'up')
val_asr = find_crossing(alt_pm, tt_pm, alt_asr, 'down')
val_maghreb = find_crossing(alt_pm, tt_pm, -0.833, 'down')
val_isha = find_crossing(alt_pm, tt_pm, isha_angle, 'down')

highlat_method = self.combo_pt_highlat.get()

if "Lintang Normal" not in highlat_method:
    # 1. Jika Matahari masih terbit & terbenam, hitung durasi malam
    if val_maghreb is not None and val_shuroq is not None:
        # Durasi siang (Julian Date Float: 1.0 = 24 Jam)
        durasi_siang = val_maghreb - val_shuroq
        durasi_malam = 1.0 - durasi_siang

    if "1/2 Malam" in highlat_method:
        porsi = durasi_malam / 2.0
        if val_isha is None: val_isha = val_maghreb + porsi
        if val_fajr is None: val_fajr = val_shuroq - porsi

    elif "1/7 Malam" in highlat_method:
        porsi = durasi_malam / 7.0
        if val_isha is None: val_isha = val_maghreb + porsi
        if val_fajr is None: val_fajr = val_shuroq - porsi

```

```

elif "Proporsi Sudut" in highlat_method:
    porsi_isha = durasi_malam * (abs(isha_angle) / 60.0)
    porsi_fajr = durasi_malam * (abs(fajr_angle) / 60.0)
    if val_isha is None: val_isha = val_maghreb + porsi_isha
    if val_fajr is None: val_fajr = val_shuroq - porsi_fajr

elif "Aqrab al-Balad" in highlat_method:
    # Estimasi matematis cepat untuk Lintang 45° di Musim Panas
    # Rata-rata selisih Maghrib-Isya adalah 1.5 jam (1.5 / 24.0 JD)
    if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
    if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)

# 2. Penanganan Ekstrem: Midnight Sun / Polar Night (Matahari tidak terbit/terbenam)
if val_maghreb is None or val_shuroq is None:
    if tt_dhohur is not None:
        # Aqrab al-Ayyam Asumsi Dasar (Menarik garis 6 jam dari waktu Dzuhur/Transit)
        if val_shuroq is None: val_shuroq = tt_dhohur - (6.0 / 24.0)
        if val_maghreb is None: val_maghreb = tt_dhohur + (6.0 / 24.0)
        if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
        if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
        if val_dhuha is None: val_dhuha = val_shuroq + (1.0 / 24.0)
        if val_asr is None: val_asr = tt_dhohur + (3.0 / 24.0)

# -----

schedule = {}
def add_to_schedule(name, tt_val, is_shuroq=False):
    if tt_val is not None:
        # --- PERBAIKAN 3: Konversi TZ aman ---
        dt_utc_aware = self.ts.tt_jd(tt_val).utc_datetime()
        dt_local_aware = dt_utc_aware.astimezone(tz_info)

        if is_shuroq: dt_local_aware -= datetime.timedelta(seconds=ikhtiyat_sec)
        else: dt_local_aware += datetime.timedelta(seconds=ikhtiyat_sec)
        schedule[name] = dt_local_aware.replace(tzinfo=None)

add_to_schedule("Subuh", val_fajr)
add_to_schedule("Terbit/Syuruq", val_shuroq, is_shuroq=True)
add_to_schedule("Dzuhur", tt_dhohur)
add_to_schedule("Ashar", val_asr)
add_to_schedule("Maghrib", val_maghreb)
add_to_schedule("Isya", val_isha)

self.daily_prayer_schedule = schedule
self.last_calculated_date = now_local.date()

except Exception as e:
    print(f"Error Silent Schedule: {e}")

```

```

        self.after(0, lambda e: self.lbl_countdown.configure(text="Gagal Kalkulasi Alarm!",
text_color="#FF5252"))
    finally:
        self._is_calculating_alarm = False

def alarm_tick(self):
    """Thread GUI (Main Thread) 1-detik loop untuk Cek Alarm dan Update Countdown Header"""

    # --- PERBAIKAN: Menyamakan detak jam dengan Zona Waktu Lokal secara Dinamis ---
    try: tz_offset = float(self.entry_pt_tz.get())
    except: tz_offset = 7.0

    tz_info = datetime.timezone(datetime.timedelta(hours=tz_offset))
    now_utc = datetime.datetime.now(datetime.timezone.utc)
    now_local = now_utc.astimezone(tz_info).replace(tzinfo=None)
    # -----

    # Update harian (refresh jam 00:00)
    if self.alarm_enabled.get() and self.last_calculated_date != now_local.date() and self.eph is not
None:
        if not getattr(self, '_is_calculating_alarm', False):
            self._is_calculating_alarm = True
            threading.Thread(target=self.update_silent_schedule, daemon=True).start()

    if self.alarm_enabled.get():
        if self.daily_prayer_schedule:
            # 1. Handle Snooze Logic
            if self.snooze_time and now_local >= self.snooze_time:
                target_time = self.snooze_time
                self.snooze_time = None
                self.trigger_alarm(self.snooze_prayer, target_time, is_snooze=True)

            # 2. Cari Salat Selanjutnya
            future_prayers = {k: v for k, v in self.daily_prayer_schedule.items() if v > now_local}

            if future_prayers:
                next_prayer = min(future_prayers, key=future_prayers.get)
                next_time = future_prayers[next_prayer]
                diff = next_time - now_local

                hours, remainder = divmod(diff.seconds, 3600)
                minutes, seconds = divmod(remainder, 60)
                self.lbl_countdown.configure(text=f"Waktu {next_prayer} dalam
{hours:02d}:{minutes:02d}:{seconds:02d}", text_color="#00E676")

                if diff.total_seconds() <= 1 and self.last_triggered_prayer != next_prayer:
                    self.trigger_alarm(next_prayer, next_time)
            else:

```

```

        # Langsung beralih ke jadwal besok agar countdown ke Subuh berjalan
        self.lbl_countdown.configure(text="Beralih ke jadwal esok hari...", text_color="#FFAB40")
        if not getattr(self, '_is_calculating_alarm', False):
            self._force_tomorrow = True
            self._is_calculating_alarm = True
            threading.Thread(target=self.update_silent_schedule, daemon=True).start()
        elif not getattr(self, '_is_calculating_alarm', False):
            self.lbl_countdown.configure(text="Menunggu Data...", text_color="#FFAB40")
        elif not self.alarm_enabled.get():
            self.lbl_countdown.configure(text="Alarm Salat: Dinonaktifkan", text_color="#FF5252")

        self.after(1000, self.alarm_tick)

def trigger_alarm(self, prayer_name, target_time, is_snooze=False):
    if not is_snooze:
        self.last_triggered_prayer = prayer_name

        city_name = self.opt_city.get()
        title = f"Waktu Salat {prayer_name}"
        msg = f"Telah masuk waktu {prayer_name} untuk wilayah {city_name} dan sekitarnya."

        if HAS_TOAST:
            threading.Thread(target=lambda: self.toaster.show_toast(title, msg, duration=10,
                                                                    threaded=True), daemon=True).start()

        if HAS_PYGAME and self.alarm_enabled.get():
            audio_path = self.adzan_audio_path.get()
            try:
                if audio_path and os.path.exists(audio_path):
                    pygame.mixer.music.load(audio_path)
                    pygame.mixer.music.play()
            except:
                else:
                    import winsound
                    winsound.MessageBeep()
            except Exception as e:
                pass

        self.show_alarm_popup(prayer_name, msg)

def show_alarm_popup(self, prayer_name, msg):
    popup = ctk.CTkToplevel(self)
    popup.title(f"Notifikasi Waktu Salat")
    popup.geometry("400x250")
    popup.attributes("-topmost", True)

    ctk.CTkLabel(popup, text=f"WAKTU {prayer_name.upper()}", font=("Segoe UI", 24, "bold"),
                  text_color="#00E5FF").pack(pady=(20, 10))

```

```

        ctk.CTkLabel(popup, text=msg, font=("Segoe UI", 12), wraplength=350,
        justify="center").pack(pady=10)

        btn_frame = ctk.CTkFrame(popup, fg_color="transparent")
        btn_frame.pack(pady=20)

        def matikan_alarm():
            if HAS_PYGAME:
                try: pygame.mixer.music.stop()
                except: pass
            popup.destroy()

        def set_snooze(menit):
            if HAS_PYGAME:
                try: pygame.mixer.music.stop()
                except: pass
            self.snooze_time = datetime.datetime.now() + datetime.timedelta(minutes=menit)
            self.snooze_prayer = prayer_name
            self.lbl_countdown.configure(text=f"Snooze {prayer_name} ({menit}m)...", text_color="#FFAB40")
            popup.destroy()

        ctk.CTkButton(btn_frame, text="Matikan", fg_color="#D32F2F", hover_color="#B71C1C",
        width=100, command=matikan_alarm).pack(side="left", padx=5)
        ctk.CTkButton(btn_frame, text="Snooze 5m", fg_color="#F57C00", hover_color="#E65100",
        width=100, command=lambda: set_snooze(5)).pack(side="left", padx=5)
        ctk.CTkButton(btn_frame, text="Snooze 10m", fg_color="#1976D2", hover_color="#1565C0",
        width=100, command=lambda: set_snooze(10)).pack(side="left", padx=5)

# =====
# LOGIKA PADA FORM/UI LAINNYA
# =====
def on_prov_change(self, prov):
    cities = sorted(list(CITY_DB[prov].keys()))
    self.opt_city.configure(values=cities)
    self.opt_city.set(cities[0])
    self.on_city_change(cities[0])

def on_city_change(self, city):
    prov = self.opt_prov.get()
    if prov in CITY_DB and city in CITY_DB[prov]:
        lat, lon = CITY_DB[prov][city]
        tz_val = self.get_tz_from_lon(lon)

    # Daftar lengkap variabel input koordinat (1 s.d 34)
    lat_entries = [
        'entry_vlat', 'entry_eph_lat', 'entry_qlat', 'entry_mt_lat', 'entry_st_lat', 'entry_pt_lat',
        'entry_qtlat', 'entry_eph3d_lat', 'entry_miz_lat', 'entry_kb_lat', 'entry_astro_lat',
        'entry_pasang_lat', 'entry_fajar_lat', 'entry_analemma_lat'

```

```

]
lon_entries = [
    'entry_vlon', 'entry_eph_lon', 'entry_qlon', 'entry_mt_lon', 'entry_st_lon', 'entry_pt_lon',
    'entry_qtln', 'entry_eph3d_lon', 'entry_miz_lon', 'entry_kb_lon', 'entry_astro_lon',
    'entry_pasang_lon', 'entry_fajar_lon', 'entry_analemma_lon'
]
tz_entries = [
    'entry_vtz', 'entry_eph_tz', 'entry_qtz', 'entry_mt_tz', 'entry_st_tz', 'entry_pt_tz',
    'entry_qttz', 'entry_miz_tz', 'entry_kb_tz', 'entry_astro_tz', 'entry_pasang_tz',
    'entry_fajar_tz', 'entry_analemma_tz', 'entry_eph3d_tz' # <--- TAMBAHKAN DI AKHIR
]

# Update semua kotak input di sidebar secara serentak
for name in lat_entries:
    if hasattr(self, name):
        e = getattr(self, name)
        e.delete(0, 'end'); e.insert(0, str(lat))

for name in lon_entries:
    if hasattr(self, name):
        e = getattr(self, name)
        e.delete(0, 'end'); e.insert(0, str(lon))

for name in tz_entries:
    if hasattr(self, name):
        e = getattr(self, name)
        e.delete(0, 'end'); e.insert(0, str(tz_val))

self.lokasi_nama.set(f'"{city}, {prov} (Lat: {lat}, Lon: {lon}, TZ: +{int(tz_val)})"')

# ---> TAMBAHKAN KODE INI UNTUK UPDATE CUACA OTOMATIS SAAT GANTI KOTA <---
if hasattr(self, 'fetch_and_update_weather'):
    labels_cuaca = [
        'lbl_status_cuaca', 'lbl_cuaca_astro',
        'lbl_cuaca_pasang', 'lbl_cuaca_fajar', 'lbl_cuaca_analemma'
    ]
    for lbl_name in labels_cuaca:
        if hasattr(self, lbl_name):
            label_obj = getattr(self, lbl_name)
            self.fetch_and_update_weather(lat, lon, label_obj)
# -----

if getattr(self, 'alarm_enabled', None) and self.alarm_enabled.get():
    threading.Thread(target=self.update_silent_schedule, daemon=True).start()

def update_calendar_widget(self):
    now = datetime.datetime.now()

```

```

# Array Nama Hari Masehi dan Pasaran Jawa
hari_masehi = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
pasaran_jawa = ["Legi", "Pahing", "Pon", "Wage", "Kliwon"]

# Ambil nama hari biasa
nama_hari = hari_masehi[now.weekday()]

# Hitung pasaran jawa secara matematis menggunakan fungsi ordinal
# Siklus 5 harian terkalibrasi presisi dengan toordinal() % 5
nama_pasaran = pasaran_jawa[now.date().toordinal() % 5]

# Format Tanggal Masehi dengan sisipan pasaran
nama_bulan = BULAN_MASEHI[now.month - 1]
tanggal_masehi = f"{nama_hari} {nama_pasaran}, {now.day} {nama_bulan} {now.year}"

# Terapkan ke Label GUI
self.lbl_cal_masehi.configure(text=tanggal_masehi)

# Format Tanggal Hijriah (Tetap seperti semula)
hijri_str = HijriConverter.get_hijri_date(now.date())
self.lbl_cal_hijri.configure(text=hijri_str)

def update_conv_ui(self):
    now = datetime.datetime.now()
    mode = self.radio_conv_var.get()

    if mode in ["m2h", "m2u"]:
        self.lbl_conv_date.configure(text="TANGGAL MASEHI")
        self.combo_conv_month.configure(values=[self.tr(v) for v in BULAN_MASEHI])
        self.combo_conv_month.set(self.tr(BULAN_MASEHI[now.month - 1]))
        self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 32)])
        self.combo_conv_day.set(f"{now.day:02d}")
        self.entry_conv_year.delete(0, 'end')
        self.entry_conv_year.insert(0, str(now.year))
    else:
        self.lbl_conv_date.configure(text="TANGGAL HIJRIAH")
        self.combo_conv_month.configure(values=[self.tr(v) for v in BULAN_HIJRIAH])
        self.combo_conv_month.set(self.tr("Ramadan"))
        self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 31)])

def build_home_dashboard(self):
    """Membuat Widget Dashboard Executive Summary untuk Beranda"""
    dash_container = ctk.CTkFrame(self.textbox, fg_color="transparent")

    # Title Dashboard
    ctk.CTkLabel(dash_container, text="🇮🇩 EXECUTIVE SUMMARY", font=("Segoe UI", 16, "bold"),
text_color="#FFD54F").pack(pady=(0, 2))

```



```

# ---> TAMBAHAN: LABEL INFORMASI LOKASI <---
lbl_dash_lokasi = ctk.CTkLabel(dash_container, text="📍 Memuat Lokasi...", font=("Segoe UI", 12,
"italic"), text_color="#00E5FF")
lbl_dash_lokasi.pack(pady=(0, 10))
# -----

grid_frame = ctk.CTkFrame(dash_container, fg_color="transparent")
grid_frame.pack(fill="x")

# BOX 1: Cuaca Lokal
box_cuaca = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1,
border_color="#333333")
box_cuaca.grid(row=0, column=0, padx=5, sticky="nsew")
ctk.CTkLabel(box_cuaca, text="☁️ Cuaca Lokal", font=("Segoe UI", 12, "bold"),
text_color="#9E9E9E").pack(pady=(10, 5))
lbl_dash_cuaca = ctk.CTkLabel(box_cuaca, text="Memuat...", font=("Segoe UI", 14, "bold"))
lbl_dash_cuaca.pack(pady=(0, 10), padx=10)

# BOX 2: Status Hilal Sore Ini (KHGT)
box_hilal = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1,
border_color="#333333")
box_hilal.grid(row=0, column=1, padx=5, sticky="nsew")
ctk.CTkLabel(box_hilal, text="🌙 Status Hilal Nanti Sore", font=("Segoe UI", 12, "bold"),
text_color="#9E9E9E").pack(pady=(10, 5))
lbl_dash_hilal = ctk.CTkLabel(box_hilal, text="Menganalisis...", font=("Segoe UI", 14, "bold"))
lbl_dash_hilal.pack(pady=(0, 5), padx=10)
lbl_dash_hilal_sub = ctk.CTkLabel(box_hilal, text="Menghitung Ephemeris", font=("Segoe UI", 11),
text_color="#9E9E9E")
lbl_dash_hilal_sub.pack(pady=(0, 10))

# BOX 3: Countdown Ijtima Berikutnya
box_ijt = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1,
border_color="#333333")
box_ijt.grid(row=0, column=2, padx=5, sticky="nsew")
ctk.CTkLabel(box_ijt, text="⌚ Countdown Ijtima", font=("Segoe UI", 12, "bold"),
text_color="#9E9E9E").pack(pady=(10, 5))
lbl_dash_ijt = ctk.CTkLabel(box_ijt, text="Memuat...", font=("Segoe UI", 14, "bold"))
lbl_dash_ijt.pack(pady=(0, 5), padx=10)
lbl_dash_ijt_sub = ctk.CTkLabel(box_ijt, text="Mencari fase bulan", font=("Segoe UI", 11),
text_color="#9E9E9E")
lbl_dash_ijt_sub.pack(pady=(0, 10))

for i in range(3): grid_frame.grid_columnconfigure(i, weight=1)

# Thread Background untuk memuat data cerdas
def load_dash_data():

```

```

import time
while self.eph is None: time.sleep(1) # Tunggu ephemeris selesai loading

try:
    lat = float(self.entry_vlat.get())
    lon = float(self.entry_vlon.get())
    tz = float(self.entry_vtz.get())
except:
    lat, lon, tz = -7.0667, 110.4100, 7.0

# ---> TAMBAHAN: UPDATE TEKS LOKASI <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_str = f" 📍 {nama_kota}, {nama_prov} (Lat: {lat}, Lon: {lon})"
except:
    lokasi_str = f" 📍 Koordinat Kustom (Lat: {lat}, Lon: {lon})"

self.after(0, lambda: lbl_dash_lokasi.configure(text=lokasi_str))
# -----

# 1. Update Cuaca
status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
cuaca_pendek = status_cuaca.split("-")[0].strip() # Ambil teks ringkasnya saja
self.after(0, lambda: lbl_dash_cuaca.configure(text=cuaca_pendek, text_color=warna_cuaca))

try:
    import math
    now = datetime.datetime.now()

    # 2. Update Countdown Ijtimak (Menggunakan PyEphem agar sangat cepat)
    waktu_sekarang = ephem.Date(now)
    next_nm = ephem.next_new_moon(waktu_sekarang)
    dt_nm_lokal = next_nm.datetime() + datetime.timedelta(hours=tz)
    diff = dt_nm_lokal - now

    hari, sisa_detik = diff.days, diff.seconds
    jam = sisa_detik // 3600
    teks_ijt = f"{hari} Hari, {jam} Jam lagi"
    teks_ijt_sub = dt_nm_lokal.strftime("%d %b %Y, %H:%M LT")

    self.after(0, lambda: lbl_dash_ijt.configure(text=teks_ijt, text_color="#FFD54F"))
    self.after(0, lambda: lbl_dash_ijt_sub.configure(text=teks_ijt_sub))

    # 3. Update Status Hilal Sore Ini (Geosentrik KHGT via Skyfield)
    earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

```

```

t0 = self.ts.utc(now.year, now.month, now.day, -int(tz))
t1 = self.ts.utc(now.year, now.month, now.day, 24 - int(tz))
loc = wgs84.latlon(lat, lon)

t_rs, y_rs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, loc))
t_sunset = None
for t_ev, y_ev in zip(np.atleast_1d(t_rs), np.atleast_1d(y_rs)):
    if y_ev == 0:
        t_sunset = t_ev
        break

if t_sunset is None:
    self.after(0, lambda: lbl_dash_hilal.configure(text="Anomali Kutub", text_color="#FF5252"))
else:
    obs_center = earth.at(t_sunset)
    m_app_geo = obs_center.observe(moon).apparent()
    s_app_geo = obs_center.observe(sun).apparent()

    elong_geo = s_app_geo.separation_from(m_app_geo).degrees

    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset)
    gast = t_sunset.gast
    lst_deg = (gast * 15.0) + lon
    ha_deg = lst_deg - (ra_m.hours * 15.0)

    lat_rad = math.radians(lat)
    ha_rad = math.radians(ha_deg)
    d_rad = dec_m.radians

    sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

    if alt_geo >= 5.0 and elong_geo >= 8.0:
        teks_hilal = "☑ WUJUD (KHGT)"
        warna_hilal = "#00E676"
    elif alt_geo > 0:
        teks_hilal = "⚠ BAWAH KRITERIA"
        warna_hilal = "#FFB74D"
    else:
        teks_hilal = "✗ DI BAWAH UFUK"
        warna_hilal = "#FF5252"

    teks_hilal_sub = f"Geo Alt: {alt_geo:.1f}° | Eln: {elong_geo:.1f}°"
    self.after(0, lambda: lbl_dash_hilal.configure(text=teks_hilal, text_color=warna_hilal))
    self.after(0, lambda: lbl_dash_hilal_sub.configure(text=teks_hilal_sub))

```

```

except Exception as e:
    self.after(0, lambda: lbl_dash_hilal.configure(text="Gagal Hitung", text_color="#FF5252"))
    self.after(0, lambda: lbl_dash_ijt.configure(text="Gagal Hitung", text_color="#FF5252"))

threading.Thread(target=load_dash_data, daemon=True).start()
return dash_container

def _translate_release_highlight_piece(self, text, lang):
    """Terjemahkan highlight lengkap dengan mempertahankan tag/pembatas teks.

    Kamus V8.0 menjadi acuan untuk bagian ini agar cache lama yang masih
    berbahasa Indonesia atau hanya diterjemahkan sebagian dapat diperbaiki.
    None berarti teks bukan bagian highlight yang diperbarui.
    """
    if lang not in ("en", "ar"):
        return None
    raw = text if isinstance(text, str) else str(text)
    result = raw
    matched = False
    for source_text, targets in RELEASE_HIGHLIGHT_TRANSLATIONS.items():
        if source_text in raw:
            result = result.replace(source_text, targets[lang])
            matched = True
    return result if matched else None

def _translate_release_value_full(self, text):
    """Terjemahkan prosa rilis memakai bahasa aktif dan cache yang sama."""
    if not isinstance(text, str) or not text.strip():
        return text
    if getattr(self, "current_lang", "id") not in ("en", "ar"):
        return text
    return self.translate_release_text_cached(text)

def _translate_release_piece_preserve_urls(self, text):
    """Translate prose while leaving URLs unchanged."""
    if not isinstance(text, str) or not text:
        return text
    url_re = re.compile(r'https?://[^\s<>"\']+|www\.[^\s<>"\']+')
    parts = url_re.split(text)
    urls = url_re.findall(text)
    translated = [self._translate_release_value_full(p) for p in parts]
    out = []
    for i, part in enumerate(translated):
        out.append(part)
        if i < len(urls):
            out.append(urls[i])
    return "".join(out)

```

```

def show_release_info(self):
    # Penanda khusus: halaman kanan sedang berada pada INFORMASI RILIS.
    # Saat bahasa ID/EN/AR diganti, halaman ini harus DIGAMBAR ULANG
    # dengan tag, link, widget, dan layout asli; bukan diisi ulang sebagai teks polos.
    self.active_right_view = "release_info"
    self._rendering_release_info = True
    self._last_raw_report = ""

    # 1. Reset state dan sembunyikan frame lain (Dibuat kebal error / Anti-Crash)
    self.textbox.grid(row=1, column=0, sticky="nsew")
    if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
    if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
    if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
    if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
    if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
    if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
    if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()
    if hasattr(self, 'frame_winai_out'): self.frame_winai_out.grid_remove()
    if hasattr(self, 'frame_kriteria_batas_out'): self.frame_kriteria_batas_out.grid_remove()
    if hasattr(self, 'frame_analemma_out'): self.frame_analemma_out.grid_remove()

    self.anim_running = False
    self.is_viewing_3d = False
    if hasattr(self, 'eph3d_is_live'): self.eph3d_is_live = False

    # 2. Pengaturan Dasar Textbox
    self.textbox.configure(state="normal", font=("Segoe UI", 16), wrap="word")
    self.textbox.delete("1.0", "end")

    # 3. Konfigurasi Tag Warna & Tipografi
    tb = self.textbox._textbox

    # Menggunakan font bawaan OS agar warna asli emoji keluar
    import platform
    emoji_font = "Segoe UI Emoji" if platform.system() == "Windows" else "Apple Color Emoji"

    # Tag khusus Emoji (Tanpa warna foreground agar dirender native oleh OS)
    tb.tag_config("emoji_header", font=(emoji_font, 28), justify="center", spacing1=10, spacing3=5)
    # Tag khusus Link T3 (Warna Putih Murni)
    tb.tag_config("link_white", font=("Segoe UI", 16), foreground="#FFFFFF", justify="center",
    spacing1=5, spacing3=15)

    tb.tag_config("title", font=("Segoe UI", 24, "bold"), foreground="#FFD54F", justify="center",
    spacing1=5, spacing3=5)
    tb.tag_config("subtitle", font=("Segoe UI", 20, "bold"), foreground="#00E5FF", justify="center",
    spacing1=20, spacing3=10)
    tb.tag_config("h4", font=("Segoe UI", 18, "bold"), foreground="#FFD54F", lmargin1=15,
    lmargin2=15, spacing1=15, spacing3=5)

```

```

        release_text_justify = "right" if getattr(self, "current_lang", "id") == "ar" else "left"
        tb.tag_config("body", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=15, lmargin2=15,
        spacing1=5, spacing3=10, justify=release_text_justify)

        # Format Khusus untuk Daftar Menu
        tb.tag_config("menu_title", font=("Segoe UI", 18, "bold"), foreground="#00E676", lmargin1=25,
        lmargin2=25, spacing1=15, spacing3=2, justify=release_text_justify)
        tb.tag_config("menu_desc", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=45,
        lmargin2=45, spacing1=2, spacing3=8, justify=release_text_justify)

        tb.tag_config("list_item", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=35, lmargin2=55,
        spacing1=5, spacing3=5, justify=release_text_justify)
        tb.tag_config("tech", font=("Consolas", 16, "bold"), foreground="#00E676")
        tb.tag_config("highlight", font=("Segoe UI", 16, "bold"), foreground="#FFD54F")
        tb.tag_config("box_info", font=("Segoe UI", 16), foreground="#00E5FF", lmargin1=40, lmargin2=60,
        spacing1=5, spacing3=5)
        tb.tag_config("box_warn", font=("Segoe UI", 16), foreground="#FFD54F", lmargin1=40,
        lmargin2=60, spacing1=5, spacing3=5)
        tb.tag_config("footer", font=("Segoe UI", 14, "italic"), foreground="#ffffff", justify="center")

        # ---> TAG BARU UNTUK CENTER WINDOW <---
        tb.tag_configure("center_window", justify="center")
        # Tambahkan baris ini di area konfigurasi tag
        tb.tag_config("welcome_title", font=("Segoe UI", 22, "bold"), foreground="#00E676",
        justify="center", spacing1=15, spacing3=10)

        # Pembatas Garis Proporsional (Anti Terpotong Saat Toggle Layar Sempit)
        garis_sama = "_" * 110
        garis_strip = "-" * 94

        # ===== HEADER =====
        t1 = "🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 \n"
        t2 = "INFORMASI RILIS & DOKUMENTASI SISTEM KHGT TIMES V8.0\n"

        self.textbox.insert("end", t1, "emoji_header")
        self.textbox.insert("end", t2, "title")

        # =====
        # ---> SISIPAN DASHBOARD EXECUTIVE SUMMARY <---
        # =====
        self.textbox.insert("end", "\n")
        pos_dash = self.textbox.index("end-1c")
        self.dashboard_widget = self.build_home_dashboard()
        tb.window_create("end", window=self.dashboard_widget)
        tb.tag_add("center_window", pos_dash, "end")
        self.textbox.insert("end", "\n\n")
        # =====

```

```

# ---> 1. KONFIGURASI TAG LINK UNTUK HEADER <---
import webbrowser
tb = self.textbox._textbox
tb.tag_config("link_aktif_header", foreground="#00B0FF", underline=True)
tb.tag_bind("link_aktif_header", "<Button-1>", lambda e:
webbrowser.open("https://hisabmu.com/khgttimes/KHGT_Times.zip"))
tb.tag_bind("link_aktif_header", "<Enter>", lambda e: self.textbox.configure(cursor="hand2"))
tb.tag_bind("link_aktif_header", "<Leave>", lambda e: self.textbox.configure(cursor="xterm"))

# ---> 2. MASUKKAN TEKS SEBELUM LINK <---
self.textbox.insert("end", "Akses dan download melalui link: ", "link_white")

# ---> 3. MASUKKAN LINK-NYA (Warna Biru & Bisa Diklik) <---
self.textbox.insert("end", "https://hisabmu.com/khgttimes/KHGT_Times.zip", ("link_white",
"link_aktif_header"))

# ---> 4. MASUKKAN GARIS BARU (Enter) <---
self.textbox.insert("end", "\n\n", "link_white")

# ===== INTEGRASI WIDGET KALENDER HIJRIAH INTERAKTIF
=====
# State Kalender Preview (Deteksi Otomatis Berdasarkan Bulan Masehi Saat Ini)
import datetime

today = datetime.date.today()
self.demo_h_year = 1447 # Nilai fallback (cadangan)
self.demo_h_month = 9 # Nilai fallback (cadangan)

# Mencari bulan Hijriah di HIJRI_DB yang beririsan dengan hari ini
is_found = False
for y, months in HIJRI_DB.items():
    for m, m_data in enumerate(months):
        nama_bulan, _, start_date_str, jumlah_hari = m_data
        parts = start_date_str.split('-')
        bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
            "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

        start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
        end_date = start_date + datetime.timedelta(days=jumlah_hari - 1)

        # Jika tanggal hari ini berada dalam rentang bulan Hijriah tersebut
        if start_date <= today <= end_date:
            self.demo_h_year = y
            self.demo_h_month = m
            is_found = True
            break
    if is_found:

```

```

        break

self.demo_cal_wrapper = ctk.CTkFrame(self.textbox, fg_color="transparent")

# Header Navigasi Kalender (Tengah)
demo_header = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
demo_header.pack(pady=(5, 15))

def demo_prev():
    self.demo_h_month -= 1
    if self.demo_h_month < 0:
        self.demo_h_month = 11
        self.demo_h_year -= 1
    if self.demo_h_year not in HIJRI_DB:
        self.demo_h_year += 1
        self.demo_h_month = 0
    return
    render_demo()

def demo_next():
    self.demo_h_month += 1
    if self.demo_h_month > 11:
        self.demo_h_month = 0
        self.demo_h_year += 1
    if self.demo_h_year not in HIJRI_DB:
        self.demo_h_year -= 1
        self.demo_h_month = 11
    return
    render_demo()

btn_prev = ctk.CTkButton(demo_header, text="◀ Sebelumnya", font=("Segoe UI", 12, "bold"),
width=110, fg_color="#1F1F1F", hover_color="#333333", command=demo_prev)
btn_prev.pack(side="left", padx=10)

self.lbl_demo_title = ctk.CTkLabel(demo_header, text="MEMUAT...", font=("Segoe UI", 18, "bold"),
text_color="#FFD54F", justify="center")
self.lbl_demo_title.pack(side="left", expand=True)

btn_next = ctk.CTkButton(demo_header, text="Berikutnya ▶", font=("Segoe UI", 12, "bold"),
width=110, fg_color="#1F1F1F", hover_color="#333333", command=demo_next)
btn_next.pack(side="right", padx=10)

self.demo_cal_grid = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="#050510", corner_radius=10,
border_width=1, border_color="#333333")
self.demo_cal_grid.pack(padx=10, pady=(0, 10))

# --- TAMBAHAN: Frame Legend Hari Penting Islam ---

```



```

self.demo_frame_legend = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="#1E1E1E",
corner_radius=8)
self.demo_frame_legend.pack(pady=(5, 5))

ctk.CTkLabel(self.demo_frame_legend, text="Legenda:", font=("Segoe UI", 12, "bold"),
text_color="#FFD54F").pack(side="left", padx=15, pady=10)

# Helper function untuk menambah item legend
def add_legend_item(parent, color, text):
    container = ctk.CTkFrame(parent, fg_color="transparent")
    container.pack(side="left", padx=10, pady=10)
    dot = ctk.CTkFrame(container, width=12, height=12, corner_radius=6, fg_color=color)
    dot.pack(side="left", padx=(0, 5))
    ctk.CTkLabel(container, text=text, font=("Segoe UI", 11)).pack(side="left")

add_legend_item(self.demo_frame_legend, "#FF5252", "Hari Haram Puasa / Hari Raya")
add_legend_item(self.demo_frame_legend, "#00E676", "Puasa Sunnah Utama")
add_legend_item(self.demo_frame_legend, "#00B0FF", "Puasa Ayyamul Bidh")
add_legend_item(self.demo_frame_legend, "#FFA000", "Puasa Senin Kamis")
add_legend_item(self.demo_frame_legend, "#8BC34A", "Ramadhan")
add_legend_item(self.demo_frame_legend, "#E040FB", "Hari Besar Islam")

# ---> TAMBAHAN: KETERANGAN PARAMETER ASTRONOMI & LOKASI <---
self.demo_frame_keterangan = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
self.demo_frame_keterangan.pack(pady=(0, 10))

teks_ket = "* Kiri: Umur Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas), Tinggi
Hilal Geo (Bawah). Dihitung saat Maghrib."
ctk.CTkLabel(self.demo_frame_keterangan, text=teks_ket, font=("Segoe UI", 13, "italic"),
text_color="ffffff").pack()

# Label Dinamis untuk Tanggal, Kota, dan Koordinat
self.lbl_demo_lokasi = ctk.CTkLabel(self.demo_frame_keterangan, text="📍 Memuat data lokasi
dan tanggal...", font=("Segoe UI", 14, "bold"), text_color="#00E5FF")
self.lbl_demo_lokasi.pack(pady=(5, 0))

# =====
# ---> TAMBAHAN FITUR PRINT (EKSPORT) KHUSUS DEMO KALENDER
# =====
self.demo_frame_export = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
self.demo_frame_export.pack(pady=(15, 5))

btn_demo_pdf = ctk.CTkButton(self.demo_frame_export, text="📄 Cetak PDF (Portrait)",
font=("Segoe UI", 13, "bold"), command=lambda: self.export_demo_kalender("pdf"),
fg_color="#D32F2F", hover_color="#B71C1C")
btn_demo_pdf.pack(side="left", padx=10)

```

```

        btn_demo_png = ctk.CTkButton(self.demo_frame_export, text="📷 Simpan PNG (Portrait)",
font=("Segoe UI", 13, "bold"), command=lambda: self.export_demo_kalender("png"),
fg_color="#F57C00", hover_color="#E65100")
        btn_demo_png.pack(side="left", padx=10)
        # =====

if not hasattr(self, '_demo_moon_imgs'):
    self._demo_moon_imgs = []

def render_demo():
    # --- PENGAMAN STARTUP: TUNGGU EPHEMERIS SELESAI DIMUAT DARI BACKGROUND THREAD --
-
    if getattr(self, 'eph', None) is None:
        for w in self.demo_cal_grid.winfo_children():
            w.destroy()
        ctk.CTkLabel(self.demo_cal_grid, text="⌚ Sedang Memuat Mesin Ephemeris NASA, harap
tunggu...", font=("Segoe UI", 14, "italic"), text_color="#FFD54F").grid(row=0, column=0, padx=50,
pady=50)
        self.after(500, render_demo) # Ulangi / Cek lagi 0.5 detik kemudian
        return
    # -----

    for w in self.demo_cal_grid.winfo_children():
        w.destroy()

    y = self.demo_h_year
    m = self.demo_h_month
    if y not in HIJRI_DB: return

    m_data = HIJRI_DB[y][m]
    nama_bulan, _, start_date_str, jumlah_hari = m_data

    parts = start_date_str.split('-')
    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
    start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
    end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

    hijri_t = f"{nama_bulan.upper()} {y} H"
    greg_t = f"{BULAN_MASEHI[start_date.month-1].upper()} {start_date.year}"
    if start_date.month != end_date.month:
        greg_t += f" - {BULAN_MASEHI[end_date.month-1].upper()} {end_date.year}"

    self.lbl_demo_title.configure(text=f"{hijri_t}\n({greg_t})")

    # ---> UPDATE INFORMASI TANGGAL, KOTA, DAN KOORDINAT KE LAYAR <---
    try:

```

```

lat_val = float(self.entry_vlat.get())
lon_val = float(self.entry_vlon.get())
elev_val = float(self.entry_velev.get())
tz_val = float(self.entry_vtz.get())
nama_kota = self.opt_city.get()
nama_prov = self.opt_prov.get()
except:
    lat_val, lon_val, elev_val, tz_val = -7.0667, 110.4100, 230.0, 7.0
    nama_kota = "Semarang"
    nama_prov = "Jawa Tengah"

teks_info_bawah = f" 📍 Kota: {nama_kota}, {nama_prov} | Koordinat: (Lat: {lat_val}, Lon:
{lon_val})"
self.lbl_demo_lokasi.configure(text=teks_info_bawah)
# -----

days_header = ["AHAD", "SENIN", "SELASA", "RABU", "KAMIS", "JUMAT", "SABTU"]
for col, day_name in enumerate(days_header):
    txt_color = "#FF5252" if col == 0 else ("#00E676" if col == 5 else "#FFFFFF")
    lbl = ctk.CTkLabel(self.demo_cal_grid, text=day_name, font=("Segoe UI", 14, "bold"),
text_color=txt_color)
    lbl.grid(row=0, column=col, padx=4, pady=(5, 2), sticky="nsew")

offset = (start_date.weekday() + 1) % 7
row_cal, col_cal = 1, offset
current_g_date = start_date

self._demo_moon_imgs.clear()

import ephem
import math
import pytz
from skyfield import almanac

obs_ephem = ephem.Observer()
obs_ephem.lat = str(lat_val)
obs_ephem.lon = str(lon_val)
obs_ephem.elevation = elev_val
obs_ephem.pressure = 1010
obs_ephem.temp = 25

matahari_ephem = ephem.Sun()

# Persiapkan objek Skyfield untuk akurasi tertinggi
earth_sf = self.eph['earth']
sun_sf = self.eph['sun']
moon_sf = self.eph['moon']

```

```

# =====
# ---> TAMBAHAN: KALKULASI FASE BULAN UTAMA UNTUK SEBULAN PENUH <---
# =====
t0_month = self.ts.utc(start_date.year, start_date.month, start_date.day, -int(tz_val))
t1_month = self.ts.utc(end_date.year, end_date.month, end_date.day, 24 - int(tz_val))
t_phases, y_phases = almanac.find_discrete(t0_month, t1_month,
almanac.moon_phases(self.eph))

phase_events = {}
if t_phases is not None:
    for t_obj, p_idx in get_safe_events(t_phases, y_phases):
        dt_loc = t_obj.utc_datetime() + datetime.timedelta(hours=tz_val)
        phase_events[dt_loc.date()] = int(p_idx)

# Map index fase ke nama string
phase_names = {0: "New Moon", 1: "First Quarter", 2: "Full Moon", 3: "Last Quarter"}
# =====

for h_day in range(1, jumlah_hari + 1):
    # --- 1. PENANDA HARI INI (CURRENT DAY HIGHLIGHT) ---
    is_today = (current_g_date == datetime.date.today())

    if is_today:
        b_color = "#FFEA00"
        b_width = 2
    else:
        b_color = "#FF5252" if col_cal == 0 else ("#00E676" if col_cal == 5 else "#555555")
        b_width = 1.0 if (col_cal == 0 or col_cal == 5) else 1

    # Height diubah ke 85 agar teks fase bulan di bawah tidak terpotong
    cell = ctk.CTkFrame(self.demo_cal_grid, fg_color="#0F172A", corner_radius=6,
border_width=b_width, border_color=b_color, width=200, height=85)
    cell.grid(row=row_cal, column=col_cal, padx=10, pady=2, sticky="nsew")
    cell.grid_propagate(False)

    frame_top = ctk.CTkFrame(cell, fg_color="transparent")
    frame_top.pack(fill="x", padx=10, pady=(5, 0))

    # --- KALKULASI DATA ASTRONOMI SAAT MAGHRIB (SUNSET) LOKAL ---
    dt_noon_local = datetime.datetime(current_g_date.year, current_g_date.month,
current_g_date.day, 12, 0, 0)
    dt_noon_utc = dt_noon_local - datetime.timedelta(hours=tz_val)
    obs_ephem.date = ephem.Date(dt_noon_utc.strftime("%Y/%m/%d %H:%M:%S"))

    try:
        waktu_sunset_ephem = obs_ephem.next_setting(matahari_ephem)
    except:

```

```

        waktu_sunset_ephem = obs_ephem.date + 0.25

    dt_sunset_utc = waktu_sunset_ephem.datetime().replace(tzinfo=pytz.utc)
    t_sunset = self.ts.from_datetime(dt_sunset_utc)

    geo_earth = earth_sf.at(t_sunset)
    app_moon_geo = geo_earth.observe(moon_sf).apparent()
    app_sun_geo = geo_earth.observe(sun_sf).apparent()

    sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
    elongasi = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg

    ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)
    gast = t_sunset.gast
    lst_deg = (gast * 15.0) + lon_val

    ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
    dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
    ha_deg = lst_deg - (ra_h * 15.0)
    lat_rad, ha_rad = math.radians(lat_val), math.radians(ha_deg)

    sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)
    tinggi_hilal = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

    try:
        prev_nm = ephemer.previous_new_moon(waktu_sunset_ephem)
        umur_bulan = waktu_sunset_ephem - prev_nm
    except:
        umur_bulan = 0.0

    illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
    iluminasi = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0

    box_kiri = ctk.CTkFrame(frame_top, fg_color="transparent")
    box_kiri.pack(side="left", anchor="nw")

    lbl_h = ctk.CTkLabel(box_kiri, text=str(h_day), font=("Segoe UI", 24, "bold"),
text_color="#FFFFFF")
    lbl_h.pack(anchor="w")

    lbl_age = ctk.CTkLabel(box_kiri, text=f"{umur_bulan:.1f}", font=("Consolas", 9, "bold"),
text_color="#FFFFFF")
    lbl_age.pack(anchor="w", pady=(0, 0))

    lbl_elong = ctk.CTkLabel(box_kiri, text=f"{elongasi:.1f}°", font=("Consolas", 9, "bold"),
text_color="#FFFFFF")
    lbl_elong.pack(anchor="w", pady=(0, 0))

```

```

box_kanan = ctk.CTkFrame(frame_top, fg_color="transparent")
box_kanan.pack(side="right", anchor="ne")

try:
    angle = (h_day / 29.53059) * 360.0
    moon_idx = int(angle) % 360
    filepath = os.path.join(BASE_DIR, "moon", f"m{moon_idx:03d}.png")
    if os.path.exists(filepath):
        from PIL import Image, ImageDraw
        img = Image.open(filepath).convert("RGBA")
        mask = Image.new('L', img.size, 0)
        draw = ImageDraw.Draw(mask)
        draw.ellipse((0, 0) + img.size, fill=255)
        img.putalpha(mask)

        ctk_img = ctk.CTkImage(img, size=(24, 24))
        self._demo_moon_imgs.append(ctk_img)
        lbl_moon = ctk.CTkLabel(box_kanan, image=ctk_img, text="")
    else:
        lbl_moon = ctk.CTkLabel(box_kanan, text=" 🌙 ", font=("Segoe UI", 14))
except:
    lbl_moon = ctk.CTkLabel(box_kanan, text=" 🌙 ", font=("Segoe UI", 14))

lbl_moon.pack(anchor="e", pady=(2, 0))

lbl_ilu = ctk.CTkLabel(box_kanan, text=f"{iluminasi:.1f}%", font=("Consolas", 9, "bold"),
text_color="#FFFFFF")
lbl_ilu.pack(anchor="e", pady=(0, 0))

lbl_alt = ctk.CTkLabel(box_kanan, text=f"{tinggi_hilal:.1f}°", font=("Consolas", 9, "bold"),
text_color="#FFFFFF")
lbl_alt.pack(anchor="e", pady=(0, 0))

frame_bottom = ctk.CTkFrame(cell, fg_color="transparent")
frame_bottom.pack(expand=True, fill="both", padx=10, pady=(2, 8))

m_str = f"{current_g_date.day} {BULAN_MASEHI[current_g_date.month-1].upper()}
{current_g_date.year}"
lbl_g = ctk.CTkLabel(frame_bottom, text=m_str, font=("Segoe UI", 10, "bold"),
text_color="#FFD54F")
lbl_g.pack(expand=True, anchor="center")

# =====
# ---> TAMBAHAN WIDGET FASE BULAN DI BAWAH TANGGAL MASEHI <---
# =====
if current_g_date in phase_events:

```

```

        phase_text = phase_names[phase_events[current_g_date]]
        lbl_phase = ctk.CTkLabel(frame_bottom, text=phase_text, font=("Consolas", 9, "bold"),
text_color="#00E5FF")
        lbl_phase.pack(expand=True, anchor="center", pady=(0, 2))
        # =====

        event_color, _ = self._get_islamic_event(h_day, m + 1, current_g_date.weekday())
        if event_color:
            dot = ctk.CTkFrame(frame_bottom, width=6, height=6, corner_radius=3,
fg_color=event_color)
            dot.pack(side="bottom", pady=(0, 2))

        col_cal += 1
        if col_cal > 6:
            col_cal = 0
            row_cal += 1
        current_g_date += datetime.timedelta(days=1)

    for i in range(7): self.demo_cal_grid.grid_columnconfigure(i, weight=1)

render_demo()

# --- CARA MEMBUAT WIDGET CENTER DI DALAM TEXTBOX ---
self.textbox.insert("end", "\n")
pos_awal = self.textbox.index("end-1c")

tb.window_create("end", window=self.demo_cal_wrapper)
self.textbox.insert("end", "\n\n")

# Menerapkan tag agar rata tengah (Center)
tb.tag_add("center_window", pos_awal, "end")
#
=====

# ===== INTRO =====
welcome_text = "Selamat datang di KHGT Times V8.0\n"
intro_1 = ("Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak "
"komprehensif dan mutakhir untuk perhitungan astrometri presisi tinggi, pemetaan visibilitas
hilal global, "
"dan penyatuan penanggalan Islam internasional serta menjadi sarana edukasi publik terkait
ilmu falak Islami dan transisi penerapan KHGT di masa depan. "
"Pada V8.0, sistem menyediakan 41 modul menu terintegrasi, antarmuka multibahasa
Indonesia-Inggris-Arab, Peta Langit Dunia online pada Menu 39, akses ke aplikasi pendamping Laporan
Astronomi pada Menu 40, serta HIJRI_DB Auto-Builder (Admin) pada Menu 41.\n")

self.textbox.insert("end", welcome_text, "welcome_title")
self.textbox.insert("end", intro_1, "body")

```

```

self.textbox.insert("end", "Melampaui Batas Pendahulunya (Beyond Accurate Times)\n", "h4")
intro_2 = ("Selama bertahun-tahun, dunia falak Islam sangat bergantung pada perangkat lunak
pionir pendahulu. Aplikasi tersebut sangat berjasa meletakkan standar dasar komputasi waktu salat dan
peta hilal di era awal komputasi.\n")
    "Namun, generasi pendahulu tersebut umumnya dibangun di atas teori analitik klasik
(algoritma VSOP87), menggunakan arsitektur single-thread yang lambat, dan dikonsep murni untuk
rukyatul hilal lokal/zonal, bukan untuk integrasi kalender global.\n")
self.textbox.insert("end", intro_2, "body")

self.textbox.insert("end", "Mengapa KHGT Times Hadir dan Sangat Penting?\n", "h4")
intro_3 = ("KHGT Times hadir sebagai Quantum Leap (Lompatan Besar) komputasi untuk menjawab
tantangan astronomi modern dan urgensi implementasi Kalender Hijriah Global Tunggal (KHGT) di
seluruh dunia. Keunggulan absolut sistem ini meliputi:\n")
self.textbox.insert("end", intro_3, "body")

self.textbox.insert("end", "► ", "highlight")
self.textbox.insert("end", "[1] Presisi Ephemeris JPL NASA: ", "tech")
self.textbox.insert("end", "Meninggalkan kalkulasi rumus hampiran matematika klasik, beralih
penuh ke integrasi numerik orbit astronomis real-time dari Development Ephemeris (DE) NASA.\n",
"list_item")

self.textbox.insert("end", "► ", "highlight")
self.textbox.insert("end", "[2] Pemindai Kesatuan Matlak Global: ", "tech")
self.textbox.insert("end", "Sistem memindai (scanning) ratusan kota di seluruh benua dalam
hitungan detik untuk melacak Titik Pertama pemenuhan kriteria KHGT (PKG 1 & PKG 2) di daratan utama
bumi.\n", "list_item")

self.textbox.insert("end", "► ", "highlight")
self.textbox.insert("end", "[3] Pemisahan Ruang Geosentris & Toposentris: ", "tech")
self.textbox.insert("end", "Menghitung secara simultan dan independen parameter dari inti bumi
(Geosentris) yang disyaratkan KHGT, dan dari mata pengamat (Toposentris) yang disyaratkan
MABIMS.\n", "list_item")

intro_4 = ("\nDengan arsitektur ini, KHGT Times adalah sebuah karya kontemporer Falak Digital
yang didedikasikan untuk observatorium, lembaga riset akademik, dan pengambil kebijakan syariah di
era modern.\n")
self.textbox.insert("end", intro_4, "body")

# ===== BAGIAN 1: TEKNOLOGI & ENGINE =====
self.textbox.insert("end", "\n" + garis_strip + "\n[ 1. REQUIREMENTS LIBRARY & ENGINE
ASTROMETRI ]\n" + garis_strip + "\n", "subtitle")

intro_tech = ("Sistem ini menggabungkan berbagai pustaka (library) saintifik standar industri untuk
mencapai keseimbangan antara akurasi tingkat tinggi (High Precision) dan kecepatan pemrosesan (High
Performance).\n")
self.textbox.insert("end", intro_tech, "body")

techs = [

```


("Skyfield API", "Library Python astronomi modern. Bertugas menghitung posisi benda langit dengan akurasi setara USNO. Mengeksekusi rotasi koordinat Geosentrik (pusat bumi) maupun Toposentrik (permukaan bumi) dengan koreksi refraksi, nutasi, presesi, dan aberrasi cahaya."),

("JPL Ephemeris", "Bukan library kode, melainkan Basis Data fisik (.bsp) dari NASA/JPL. Sistem V8.0 memakai de440s.bsp secara otomatis untuk tahun 1850-2149 dan de441-new.bsp untuk tahun di luar rentang tersebut (hingga batas aplikasi 1-2999 M), sehingga fokus Matahari-Bulan era modern menggunakan kernel yang lebih ringkas tanpa kehilangan jangkauan panjang."),

("PyEphem (ephem)", "Library engine C-based turunan XEphem. Dipakai khusus sebagai akselerator kalkulasi brutal (Brute-Force), seperti iterasi mencari Waktu Terbenam (Sunset) dan Ijtimak Global di ratusan kota pada Modul 3 dan Modul 5 secara instan."),

("SciPy (scipy.interpolate)", "Library komputasi matematika kompleks. Bertugas menggunakan algoritma 'Bicubic Griddata' untuk merajut puluhan titik data kasar (Latitude/Longitude) menjadi Peta Heatmap Visual (Crescent Visibility HD Map) yang mulus (Seamless) di Modul 2."),

("NumPy", "Library manipulasi matriks. Digunakan berdampingan dengan SciPy dan Matplotlib untuk memproses Array Kalkulasi 50 Tahun serta data Linspace (pengiris waktu per detik) tanpa membebani RAM CPU."),

("Matplotlib", "Engine rendering visual grafis. Membangun ruang kanvas 3D Spasial, kurva garis ketinggian harian (Altitude Chart), dan instrumen kompas bayangan kutub (Mizwala)."),

("CustomTkinter & Tk", "Framework Graphical User Interface (GUI). Menyulap kode terminal Python menjadi aplikasi Desktop yang modern, memiliki Dark-Mode, Sidebar interaktif, dan Frame yang dinamis."),

("Pillow (PIL)", "Engine pemroses gambar raster digital. Bertugas me-render dan melukis (ImageDraw) secara internal pada modul Generator Gambar Jadwal Shalat, Kalender Hijriah/Masehi beresolusi tinggi, serta kalkulasi masking rotasi fase bulan."),

("Requests & Bs4", "Library koneksi HTTP. Digunakan pada Modul WinAI untuk menjangkau API Gemini Google, sekaligus melakukan 'Scraping' otomatis (Google Search/DuckDuckGo) secara Real-time saat AI butuh fakta terbaru di luar database."),

("FPDF & CSV", "Library manajemen format data. FPDF mencetak String text dari terminal ke file dokumen PDF yang terstruktur, sementara CSV memecah string tabel komparasi 50 tahun menjadi pemisah titik-koma (;) agar siap dibuka di Microsoft Excel.")

]

for name, desc in techs:

self.textbox.insert("end", "> ", "highlight")

self.textbox.insert("end", f"{name} : ", "tech")

self.textbox.insert("end", f"{desc}\n\n", "list_item")

===== BAGIAN 2: KONSEP DASAR KHGT =====

self.textbox.insert("end", "\n" + garis_strip + "\n[2. KONSEP DASAR, PRINSIP & FORMULASI KHGT]\n" + garis_strip + "\n\n", "subtitle")

b2_intro = ("Kalender Hijriah Global Tunggal (KHGT) merupakan tonggak peradaban sistem penanggalan Islam unifikatif hasil Muktamar Internasional Turki 2016. Berbeda dengan sistem lokal/zonal, KHGT bertujuan menyatukan umat Islam di bawah satu sistem waktu global.\n\n")

self.textbox.insert("end", b2_intro, "body")

self.textbox.insert("end", "3 Prinsip Utama KHGT:\n", "h4")

```

self.textbox.insert("end", " ▶ ", "highlight")
self.textbox.insert("end", "Kesatuan Matlak (Ittihad al-Matali): ", "tech")
self.textbox.insert("end", "Seluruh bumi dianggap sebagai satu kesatuan zona wilayah.\n",
"list_item")

self.textbox.insert("end", " ▶ ", "highlight")
self.textbox.insert("end", "Kaidah Transfer Visibilitas (Naql al-Rukyat): ", "tech")
self.textbox.insert("end", "Jika hilal secara astronomis mungkin dirukyat di SATU titik daratan di
bumi, maka visibilitas tersebut sah dan berlaku untuk SELURUH dunia.\n", "list_item")

self.textbox.insert("end", " ▶ ", "highlight")
self.textbox.insert("end", "Keterpaduan Hari: ", "tech")
self.textbox.insert("end", "Bulan Hijriah dimulai serentak di seluruh dunia tanpa ada negara yang
berbeda tanggal.\n\n", "list_item")

self.textbox.insert("end", "Parameter Kriteria (Imkanur Rukyat Istanbul 2016):\n", "h4")
self.textbox.insert("end", "1. Tinggi Hilal Geosentris (Geocentric Altitude) minimal 5 derajat (Alt >=
5°).\n", "box_info")
self.textbox.insert("end", "2. Sudut Elongasi Geosentris (Geocentric Elongation) minimal 8 derajat
(Elong >= 8°).\n\n", "box_info")

self.textbox.insert("end", "Mekanisme Eksekusi (Parameter Kalender Global / PKG):\n", "h4")
self.textbox.insert("end", " 1) [PKG 1] Kondisi Normal (Wilayah Timur hingga Tengah Bumi):\n",
"highlight")
self.textbox.insert("end", "Sistem memindai apakah kriteria visibilitas (Alt >= 5° & Elong >= 8°)
terpenuhi di daratan mana pun di bumi SEBELUM pukul 00:00 UTC. Jika terpenuhi, maka besok adalah
awal bulan baru secara global.\n\n", "box_info")
self.textbox.insert("end", " 2) [PKG 2] Kondisi Kritis (Wilayah Ekstrem Barat / Benua Amerika):\n",
"highlight")
self.textbox.insert("end", "Jika kriteria visibilitas baru terpenuhi SETELAH pukul 00:00 UTC, maka
sistem otomatis membandingkan waktu Ijtimak (Konjungsi) terhadap Terbit Fajar di Gisborne, Selandia
Baru.\n", "box_warn")
self.textbox.insert("end", " • Jika Ijtimak SEBELUM fajar Gisborne: Maka besok tetap masuk bulan
baru.\n", "list_item")
self.textbox.insert("end", " • Jika Ijtimak SETELAH fajar Gisborne: Maka masuk bulan baru DITUNDA
lusa (Lihat hasil hitung pada tanggal = ijtimak + 1) agar wilayah Timur bumi tidak berpuasa sebelum
ijtimak terjadi.\n", "list_item")

# ===== BAGIAN 3: DAFTAR FITUR LENGKAP (41 MENU)
=====

self.textbox.insert("end", "\n" + garis_strip + "\n[ 3. MANUAL BOOK: PENJELASAN 41 MODUL MENU
V8.0 ]\n" + garis_strip + "\n", "subtitle")
self.textbox.insert("end", "Gunakan Dropdown Menu di Sidebar Kiri untuk berpindah antar modul.
Berikut rinciannya:\n", "body")

detailed_features = [
    ("1) Kalender Hijriah Berjalan",
    " • Kegunaan: Kalender visual interaktif yang terintegrasi dengan database KHGT.\n")

```

"Menyajikan tampilan kalender bulanan yang dinamis. Dilengkapi dengan penanda hari penting Islam, puasa sunnah, serta informasi astronomi hilal (umur, elongasi, iluminasi, tinggi) yang dihitung otomatis saat maghrib.\n"

- "• Sub-Menu Input: Pilih Bulan dan Tahun Hijriah.\n"

- "• Aksi & Output: Grid visual kalender HD dengan sistem navigasi offset otomatis. Dilengkapi tombol cetak PDF / PNG Resolusi Tinggi.")

("2) Kalender Masehi Berjalan",

- "• Kegunaan: Kalender visual penanggalan sipil internasional.\n"

"Menampilkan matriks kalender Gregorian dengan penyematan tanggal KHGT sebagai pelengkap. Menghitung rotasi fase bulan visual untuk setiap harinya.\n"

- "• Sub-Menu Input: Pilih Bulan dan Tahun Masehi.\n"

- "• Aksi & Output: Mesin render grafis kalender yang mendukung ekspor dokumen cetak.")

("3) Konversi Kalender",

- "• Kegunaan: Transformasi tanggal antara Masehi (Gregorian/Julian) dan Hijriah (KHGT).\n"

"Melakukan konversi tanggal dengan akurasi tinggi menggunakan database KHGT yang sudah terintegrasi. Menghitung secara astronomis jatuhnya fase bulan baru (bukan rumus pembagian hari manual).\n"

- "• Sub-Menu Input: Mode Konversi (M2H / H2M), Input Tanggal, Koreksi Hari Ikhtiyat.\n"

- "• Aksi & Output: Tampilan tanggal hasil konversi lengkap dengan nama hari dan hari pasaran.")

("4) Visibility Hilal (KHGT)",

- "• Kegunaan: Analisis komprehensif visibilitas hilal berdasarkan multi-kriteria global.\n"

"Modul ini melakukan perhitungan astrometri presisi tinggi untuk membandingkan posisi Bulan dan Matahari saat terbenam. Sistem mengevaluasi data dari dua sudut pandang: Geosentrik (pusat bumi) dan Toposentrik (permukaan bumi/lokasi pengamat), guna memberikan akurasi data yang sesuai dengan standar hisab internasional.\n"

- "• Sub-Menu Input: Tanggal Observasi (Y/M/D), Titik Koordinat Pengamat (Lat, Lon, Elevasi, TZ), dan Kondisi Atmosfer.\n"

- "• Aksi & Output: Menghitung umur bulan sejak ijtimak dan memisahkan nilai Altitude/Elongasi ke dalam wujud Geosentris dan Toposentris. Mengevaluasi langsung terhadap 5 kriteria sekaligus (KHGT, MABIMS, Ilyas, Danjon, Yallop.")

("5) Analisis Hilal Global",

- "• Kegunaan: Pelacakan visibilitas hilal di seluruh zona waktu dunia secara kronologis.\n"

"Algoritma ini melakukan iterasi per menit pada garis tanggal internasional untuk mendeteksi di mana dan kapan kriteria hilal pertama kali terpenuhi di muka bumi. Fokus utamanya adalah memvalidasi prinsip kesatuan matlak global.\n"

- "• Sub-Menu Input: Tanggal Referensi dan Waktu Acuan UTC.\n"

- "• Aksi & Output: Mencetak tabel raksasa yang men-sorting 300+ Kota di database dari nilai umur bulan dan ketinggian hilal (Geosentris & Toposentris). Membantu pengguna melihat negara mana yang lebih dulu berpeluang merukyat.")

("6) Status Kriteria Batas",

- "• Kegunaan: Simulator Anatomi Kriteria Imkanur Rukyat (KHGT).\n"

"Instrumen edukasi untuk membedah parameter fisis kriteria KHGT (5-8). Pengguna dapat melihat bagaimana posisi bulan berinteraksi dengan zona kritis seperti Batas Danjon dan Bias Cahaya Senja.\n"

- Sub-Menu Input: Tanggal, Lokasi, dan dua buah Slider Interaktif (Parameter Elongasi & Ketinggian).\n"

- Aksi & Output: Sistem akan menaruh titik Bintang hasil kalkulasi astrometri aktual pada grafik Kuadran 2D. Jika pengguna menggeser slider secara manual, letak titik Bintang akan berpindah ke zona-zona pembatas (Zona Merah Danjon, Zona Jingga Bias Senja, atau Zona Hijau KHGT). Status Kelulusan Teks di bawah grafik akan ikut berubah-ubah menjelaskan alasan saintifik kegagalan/kelolosan posisi Bintang tersebut."),

("7) Kota Pertama KHGT (Mainland)",

- Kegunaan: Melacak titik daratan utama (Mainland) pertama di dunia yang masuk bulan baru.\n"

"Merupakan fitur krusial dalam Kalender Hijriah Global Tunggal untuk menentukan dimulainya hari pertama bulan Hijriah. Sistem secara otomatis mengabaikan pulau-pulau kecil di Pasifik dan fokus pada daratan utama benua yang memenuhi syarat 5-8 (Tinggi 5°, Elongasi 8°).\n"

- Sub-Menu Input: Tanggal Konjungsi/Ijtima'.\n"

- Aksi & Output: Laporan teks dan peta topografi global yang menandai lokasi spesifik (bintang emas) lengkap dengan analisis PKG 1 dan PKG 2 (Kriteria Gisborne)."),

("8) Crescent Visibility Map",

- Kegunaan: Pemindaian peta visibilitas hilal global dalam resolusi tinggi (HD).\n"

"Menggunakan mesin iterasi spasial untuk memetakan visibilitas hilal ke seluruh permukaan bumi. Modul ini mendukung pemetaan berbagai lapisan data astronomi (layers) secara seamless, mulai dari ketinggian hilal hingga area interseksi kriteria fisis tertentu.\n"

- Sub-Menu Input: Tanggal Observasi, Mode Fase Bulan, Target Pemetaan (Sunset/Moonset/Best Time).\n"

- Aksi & Output: Window grafik interaktif yang menampilkan peta dunia dengan gradien warna visibilitas. Output dapat diekspor ke PNG/SVG resolusi tinggi untuk kebutuhan publikasi ilmiah."),

("9) Altitude Chart Analyser",

- Kegunaan: Visualisasi kurva ketinggian harian benda langit.\n"

"Modul ini merelasikan pergerakan semu harian Matahari dan Bulan dalam sebuah grafik sinusoida. Berguna untuk melihat titik puncak (transit), waktu terbit-terbenam, serta gap ketinggian antara Matahari dan Bulan sepanjang 24 jam penuh di lokasi tertentu.\n"

- Sub-Menu Input: Tanggal, Titik Koordinat.\n"

- Aksi & Output: Menggambar grafik garis yang memotong Sumbu Nol (Garis Ufuk). Pengguna dapat melihat secara matematis jam berapa titik temu kedua benda langit terjadi, lengkap dengan penanda interaktif 'Waktu Sekarang' (Real-time)."),

("10) Komparasi 50 Tahun (Ramadhan)",

- Kegunaan: Studi perbandingan penentuan awal bulan antara KHGT dan kriteria lokal (MABIMS).\n"

"Melakukan simulasi hisab selama 5 dekade ke depan untuk melihat sinkronisasi atau perbedaan tanggal antara standar global (KHGT) dan standar regional Asia Tenggara (MABIMS) di titik Sabang.\n"

- Sub-Menu Input: Hanya dieksekusi sekali jalan (1447 H - 1496 H).\n"

"• Aksi & Output: Tabel laporan komparasi per tahun dengan status 'Serentak' atau 'Beda (Mundur 1 Hari)'.")

("11) Tabel Tinggi Hilal 50 Thn (Ramadhan)",

"• Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Ramadhan selama 50 tahun.\n"

"Menyajikan data statistik ketinggian hilal jangka panjang, baik secara geosentrik maupun toposentrik.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel.")

("12) Tabel Elongasi Hilal 50 Thn (Ramadhan)",

"• Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Ramadhan selama 50 tahun.\n"

"Menyajikan data statistik elongasi hilal jangka panjang, baik secara geosentrik maupun toposentrik.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel.")

("13) Komparasi 50 Tahun (Syawal)",

"• Kegunaan: Analisis potensi beda Hari Raya Idul Fitri umat Islam masa depan.\n"

"Identik dengan modul 20, namun dikhususkan sepenuhnya untuk melacak tanggal masuknya 1 Syawal selama 50 tahun ke depan menggunakan algoritma KHGT vs MABIMS.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Laporan matriks perbandingan 50 tahun.")

("14) Tabel Tinggi Hilal 50 Thn (Syawal)",

"• Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Syawal selama 50 tahun.\n"

"Menyajikan data statistik ketinggian hilal jangka panjang, khusus untuk penentuan Idul Fitri.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel.")

("15) Tabel Elongasi Hilal 50 Thn (Syawal)",

"• Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Syawal selama 50 tahun.\n"

"Menyajikan data statistik elongasi hilal jangka panjang, khusus untuk penentuan Idul Fitri.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel.")

("16) Komparasi 50 Tahun (Zulhijah)",

"• Kegunaan: Analisis potensi beda Hari Raya Idul Adha umat Islam masa depan.\n"

"Modul ini mensimulasikan data 5 dekade ke depan untuk melihat sinkronisasi atau perbedaan tanggal masuknya 1 Zulhijah (Idul Adha) antara standar KHGT vs MABIMS Sabang.\n"

"• Sub-Menu Input: Eksekusi Otomatis.\n"

"• Aksi & Output: Laporan matriks perbandingan 50 tahun untuk bulan Zulhijah.")

("17) Tabel Tinggi Hilal 50 Thn (Zulhijah)",
 "• Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Zulhijah selama 50 tahun.\n"
 "Menyajikan data statistik ketinggian hilal jangka panjang, khusus untuk penentuan Idul Adha.\n"
 "• Sub-Menu Input: Eksekusi Otomatis.\n"
 "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),

("18) Tabel Elongasi Hilal 50 Thn (Zulhijah)",
 "• Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Zulhijah selama 50 tahun.\n"
 "Menyajikan data statistik elongasi hilal jangka panjang, khusus untuk penentuan Idul Adha.\n"
 "• Sub-Menu Input: Eksekusi Otomatis.\n"
 "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),

("19) Komparasi 50 Tahun (12 Bulan)",
 "• Kegunaan: Analisis probabilitas sinkronisasi global KHGT vs MABIMS.\n"
 "Mengekstrak dan membandingkan data awal bulan Hijriah selama 50 tahun penuh (12 bulan x 50 tahun = 600 bulan) untuk menghitung statistik keseragaman secara menyeluruh.\n"
 "• Sub-Menu Input: Eksekusi Otomatis.\n"
 "• Aksi & Output: Tabel komprehensif 600 bulan beserta ringkasan persentase probabilitas keseragaman global."),

("20) Fase Bulan (Moonphase)",
 "• Kegunaan: Prediktor siklus fase bulan bulanan dan tahunan.\n"
 "Menghitung fraksi iluminasi dan sudut fase untuk menentukan bentuk bulan yang tampak dari bumi, mulai dari New Moon, First Quarter, Full Moon, hingga Last Quarter.\n"
 "• Sub-Menu Input: Tahun Masehi (Mendukung tahun 0 M s/d 3000 M).\n"
 "• Aksi & Output: Mencetak matriks tabel 4 Fase Utama (New Moon/Ijtimaq, First Quarter, Full Moon/Purnama, Last Quarter) secara berurutan beserta Jam:Menit-nya."),

("21) Moon Times",
 "• Kegunaan: Jadwal terbit, transit, dan terbenam Bulan selama satu bulan penuh.\n"
 "Menghitung dinamika harian Bulan dengan mempertimbangkan koreksi refraksi atmosfer dan paralaks. Data ini sangat penting bagi pengamat langit malam dan astronom.\n"
 "• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
 "• Aksi & Output: Mencetak tabel Waktu Terbit (Moonrise), Titik Puncak/Kulminasi (Transit), dan Waktu Terbenam (Moonset) dengan efek refraksi atmosfer lokal."),

("22) Sun Times",
 "• Kegunaan: Jadwal harian Matahari (Sunrise, Transit, Sunset).\n"
 "Menyediakan data presisi mengenai waktu fajar, tengah hari, dan terbenamnya matahari. Modul ini menjadi dasar perhitungan waktu-waktu fenomena siang hari.\n"
 "• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
 "• Aksi & Output: Tabel bulanan waktu matahari terbit hingga terbenam (Sunrise, Transit Zawal, Sunset)."),

("23) Sun Moon Ephemeris",
 "• Kegunaan: Ekstraksi data koordinat langit (RA/Dec) secara detail.\n"

"Menyediakan data posisi murni objek (ephemeris) per interval waktu tertentu (detik/menit/jam). Data mencakup Right Ascension, Declination, Azimuth, Altitude, hingga jarak benda langit ke bumi (Distance).\n"

- Sub-Menu Input: Target (Sun/Moon), Ref Waktu (UT1/TDT), Lokasi (Geosentris/Toposentris), dan Set Waktu Interval.\n"

- Aksi & Output: Tabel raksasa data saintifik murni berisi parameter R.A, Deklinasi, Altitude, Azimuth, Jarak Aktual, Semi-Diameter, dan Delta-T."),

("24) Sistem Sun Moon Earth",

- Kegunaan: Visualisasi 3D sistem tata surya geosentrik sederhana.\n"

"Menampilkan model 3D interaktif yang menunjukkan posisi Bulan dan Matahari dalam mengelilingi Bumi sebagai pusat pengamatan (Geosentrik). Termasuk proyeksi kerucut bayangan bumi (Umbra).\n"

- Sub-Menu Input: Set Tanggal Simulasi 3D.\n"

- Aksi & Output: Kanvas 3D interaktif yang menunjukkan orientasi orbit benda langit. Bisa di-Rotasi dan di-Zoom menggunakan mouse."),

("25) Prayer Times",

- Kegunaan: Jadwal salat 5 waktu dan fenomena senja (Twilight).\n"

"Modul ini menghitung waktu Subuh, Terbit, Duha, Zuhur, Asar, Magrib, dan Isya. Mendukung berbagai standar otoritas internasional dan metode khusus untuk wilayah lintang tinggi.\n"

- Sub-Menu Input: Tanggal, Koordinat Lokasi, Parameter Suhu dan Ketinggian tempat, Mazhab, dan Metode Lembaga.\n"

- Aksi & Output: Menghasilkan Tabel Jadwal harian/bulanan lengkap dengan Midnight. Mendukung ekspor ke Gambar (Jadwal Dinding)."),

("26) Qiblah Direction & Times",

- Kegunaan: Kalkulator arah kiblat dan peta navigasi Ka'bah.\n"

"Selain menghitung sudut derajat arah kiblat dari arah Utara-Barat menggunakan model Bumi Ellipsoid WGS84, modul ini menyediakan visualisasi garis kiblat pada peta dunia untuk memastikan jalur terpendek (Great Circle) menuju Makkah.\n"

- Sub-Menu Input: Titik Koordinat.\n"

- Aksi & Output: Menampilkan derajat sudut Ka'bah dari Utara Sejati dan proyeksi Garis Lengkung Great Circle di peta topografi Dunia."),

("27) Qibla Time (Rashdul Lokal)",

- Kegunaan: Menentukan waktu saat matahari berada tepat di atas arah kiblat suatu lokasi.\n"

"Menggunakan algoritma segitiga bola (Spherical Trigonometry) untuk mencari waktu harian di mana azimuth matahari berhimpit dengan azimuth kiblat lokasi tersebut, sehingga bayangan benda tegak dapat digunakan untuk verifikasi arah Ka'bah secara manual.\n"

- Sub-Menu Input: Tanggal Observasi, Titik Koordinat.\n"

- Aksi & Output: Mencetak Waktu Pertama dan Waktu Kedua terjadinya fenomena Rashdul Kiblat lokal."),

("28) Kalkulator Mizwala (Bayangan)",

- Kegunaan: Simulator instrumen tongkat istiwa' atau Mizwala.\n"

"Menghitung panjang dan arah bayangan benda tegak sepanjang hari. Sangat berguna untuk penentuan arah kiblat manual dan praktikum falak di lapangan.\n"

- Sub-Menu Input: Tinggi Tongkat (cm), dan Waktu Simulasi (Live/Kustom).\n"
- Aksi & Output: Tabel harian pergerakan bayangan dan visualisasi kompas Polar 2D yang berputar selaras dengan matahari.)",

("29) Evaluasi Fajar Shadiq & SQM",
"• Kegunaan: Analisis fajar fisis dan kegelapan langit (Sky Quality).\n"
"Modul ini mengevaluasi kemunculan fajar shadiq berdasarkan sudut depresi matahari dan estimasi teoritis nilai SQM (Sky Quality Meter).\n"
"• Sub-Menu Input: Tanggal dan Lokasi.\n"
"• Aksi & Output: Nilai SQM estimasi dan grafik waktu kemunculan fajar secara fisis-astronomis.")",

("30) Analisis Gerhana",
"• Kegunaan: Pemindaian otomatis fenomena Gerhana Matahari dan Gerhana Bulan.\n"
"Menggunakan pustaka mesin pelacak otonom fenomena Syzygy untuk mendeteksi waktu puncak gerhana, jenis gerhana (Total, Cincin, Penumbra), serta visibilitasnya di wilayah Indonesia dan global.\n"
"• Sub-Menu Input: Tahun Kalkulasi (1 M hingga 2999 M).\n"
"• Aksi & Output: Tabel daftar gerhana setahun. Tersedia tombol untuk mengeksport Jalur Totalitas Gerhana Matahari ke format Google Earth (KML).")",

("31) Equinox & Solstice",
"• Kegunaan: Menghitung waktu pergantian musim astronomis.\n"
"Mendeteksi waktu presisi saat matahari melintasi ekuator langit (Equinox) dan saat mencapai deklinasi maksimum/minimum (Solstice).\n"
"• Sub-Menu Input: Tahun.\n"
"• Aksi & Output: Tanggal dan jam terjadinya Musim Semi, Panas, Gugur, dan Dingin.")",

("32) Planetary Times",
"• Kegunaan: Jadwal visibilitas 7 planet utama (Merkurius s/d Neptunus).\n"
"Menghitung waktu terbit, kulminasi, dan terbenam setiap planet untuk membantu observasi astronomi planetari selama sebulan penuh.\n"
"• Sub-Menu Input: Pemilihan Target Planet, Bulan, dan Tahun.\n"
"• Aksi & Output: Tabel waktu penampakan planet di langit malam (Rise/Set).")",

("33) Generator Analemma Matahari",
"• Kegunaan: Pemetaan lintasan analemma Matahari dalam satu tahun.\n"
"Memvisualisasikan lintasan 'angka 8' yang dibentuk oleh posisi matahari di jam yang sama sepanjang tahun akibat kemiringan sumbu bumi dan orbit elips.\n"
"• Sub-Menu Input: Tahun Analisis, Jam Pengamatan (Lokal), dan Lokasi.\n"
"• Aksi & Output: Laporan titik ekstrem analemma (solstice/equinox) dan grafik lintasan analemma tahunan.")",

("34) Prediktor Pasang Surut Gravitasi",
"• Kegunaan: Memprediksi fenomena pasang surut air laut akibat gaya gravitasi Bulan dan Matahari.\n"
"Memberikan estimasi waktu pasang tertinggi dan terendah berdasarkan posisi relatif benda langit untuk membantu keamanan observasi di pesisir.\n"

- "• Sub-Menu Input: Tanggal dan Koordinat Pantai.\n"
- "• Aksi & Output: Laporan prediksi fluktuasi gravitasi dan estimasi waktu pasang harian."),

("35) Astrofotografi & Kontras Visibilitas",

- "• Kegunaan: Analisis teknis untuk pengambilan foto hilal (imaging).\n"
- "Modul ini menghitung peluang hilal terekam kamera CCD/CMOS dengan menganalisis jendela kontras (contrast window) antara kecerahan hilal dan latar belakang langit senja.\n"
- "• Sub-Menu Input: Tanggal dan Koordinat Lokasi.\n"
- "• Aksi & Output: Rekomendasi waktu terbaik pemotretan dan status kontras visibilitas fisis."),

("36) Live Animasi",

- "• Kegunaan: Simulator posisi benda langit secara real-time dan terintegrasi observatorium.\n"
- "Menampilkan visualisasi pergerakan Matahari, Bulan, Rasi Bintang, dan Planet di atas cakrawala pengamat secara dinamis. Kini dilengkapi dengan fitur pelacakan (tracking) benda langit secara online via internet.\n"
- "• Integrasi Stellarium: Memiliki akses langsung ke engine Stellarium. Terdapat panel komparasi data astronomis (Matahari dan Bulan) secara real-time antara hasil kalkulasi presisi internal sistem KHGT (JPL de440s/de441-new otomatis) dengan output aplikasi Stellarium.\n"
- "• Sub-Menu Input: Waktu Simulasi (Tanggal & Jam), Mode LIVE, Tombol Tracker Online, dan Tombol Komparasi Stellarium.\n"
- "• Aksi & Output: Melukis garis ufuk, memproyeksikan benda langit secara presisi, fitur 'Zoom Hilal', serta menampilkan jendela tabel komparasi parameter astrometri (Alt, Az, Jarak, RA, Dec, dll)."),

("37) Simulasi Ephemeris 3D",

- "• Kegunaan: Simulator pergerakan benda langit 3D menggunakan mesin Matplotlib.\n"
- "Memberikan pengalaman visual yang lebih ilmiah dan presisi. Pengguna dapat melihat posisi aktual bintang, matahari, dan bulan dalam koordinat bola langit dengan grid yang dapat diputar secara bebas.\n"
- "• Sub-Menu Input: Slider Kontrol Waktu (Tahun, Bulan, Tgl, Jam) dan Tombol Cari Sunset Otomatis.\n"
- "• Aksi & Output: Grafik 3D bola langit dengan garis ekuator dan ufuk, melukis titik objek pada koordinat Toposentrik."),

("38) WinAI: Aplikasi AI Windows",

- "• Kegunaan: Asisten cerdas berbasis kecerdasan buatan (AI) untuk konsultasi astronomi dan fikih.\n"
- "Mengintegrasikan model bahasa besar (Gemini) yang terhubung dengan konteks aplikasi. AI mampu membaca kueri internet dan merujuk dokumen Tarjih secara offline.\n"
- "• Sub-Menu Input: Kotak Chat untuk mengetik Prompt.\n"
- "• Aksi & Output: Jawaban teks cerdas dan kemampuan menjalankan fitur aplikasi secara background (Smart Interceptor)."),

("39) Peta Langit Dunia",

- "• Kegunaan: Membuka Peta Langit Dunia KHGT Times berbasis web untuk eksplorasi langit secara online.\n"
- "Layanan dibuka melalui browser default pada alamat <https://kasmui.cloud/langit/>.\n"
- "• Sub-Menu Input: Tidak memerlukan input khusus pada aplikasi desktop.\n"
- "• Aksi & Output: Membuka Peta Langit Dunia online pada browser default saat menu dipilih."),

```

("40) Laporan Astronomi",
  "• Kegunaan: Membuka aplikasi pendamping Laporan Astronomi dari KHGT Times.\n"
  "Modul ini menjalankan file 'LaporanAstronomi.exe' (atau 'LaporanAstronomi.exe') yang berada
di folder yang sama dengan aplikasi KHGT Times untuk mengakses laporan astronomi.\n"
  "• Sub-Menu Input: Tidak memerlukan input khusus pada KHGT Times; aplikasi pendamping
dibuka otomatis saat menu dipilih.\n"
  "• Aksi & Output: Menjalankan aplikasi Laporan Astronomi eksternal. Jika file pendamping tidak
ditemukan atau gagal dijalankan, sistem menampilkan pesan kesalahan."),

("41) HIJRI_DB Auto-Builder (Admin)",
  "• Kegunaan: Fitur khusus Developer untuk merakit fondasi Kalender Aplikasi (Siklus Pembuatan
Data JSON).\n"
  "Modul tingkat lanjut untuk membangun file database penanggalan KHGT secara otomatis dari
iterasi ribuan tahun mesin Ephemeris NASA.\n"
  "• Sub-Menu Input: Tahun Awal Hijriah s/d Tahun Akhir Hijriah (Memerlukan Otorisasi
Admin).\n"
  "• Aksi & Output: Menulis seluruh hasilnya menjadi File 'db_hijriah.json' baru yang secara
otomatis dimuat ulang ke dalam memori aplikasi.")
]

```

```

# Render list menu dengan dua tag berbeda agar berwarna dan rapi
for title, desc in detailed_features:
    self.textbox.insert("end", f"▶ {title}\n", "menu_title")
    self.textbox.insert("end", f"{desc}\n\n", "menu_desc")

# ===== BAGIAN 4: HIGHLIGHT PEMBARUAN =====
self.textbox.insert("end", "\n" + garis_strip + "\n[ 4. HIGHLIGHT PEMBARUAN EKSKLUSIF (VERSI 8.0)
]\n" + garis_strip + "\n", "subtitle")

updates = [
    ("[V7.7 - MULTIBAHASA & SIDEBAR] Antarmuka Indonesia, Inggris & Arab", "Sidebar dan elemen
antarmuka utama kini mendukung penerjemahan ke Bahasa Indonesia, Inggris, dan Arab sehingga
navigasi modul dapat digunakan secara multibahasa."),
    ("[V7.7 - ALTITUDE CHART ANALYZER] Analisis Grafik & Kombinasi", "Menu 9 (Altitude Chart
Analyser) menyediakan analisis grafik ketinggian dan kombinasi data untuk membantu pembacaan
posisi benda langit secara lebih visual dan terstruktur."),
    ("[V8.0 - PETA LANGIT DUNIA] Integrasi Peta Langit Online", "Menu 39 adalah Peta Langit Dunia.
Saat menu dipilih, KHGT Times membuka layanan peta langit online pada https://kasmui.cloud/langit/
melalui browser default."),
    ("[V8.0 - LAPORAN ASTRONOMI] Penyesuaian Nomor Menu", "Menu 40 adalah Laporan
Astronomi. Menu ini membuka aplikasi pendamping 'LaporanAstronomi.exe' atau
'LaporanAstronomi.exe' dari folder aplikasi KHGT Times dan menampilkan pesan kesalahan jika file
pendamping tidak tersedia atau gagal dijalankan."),
    ("[V8.0 - MENUS & DOKUMENTASI] Penyeragaman 41 Modul", "Daftar menu aktual V8.0 terdiri
atas 41 menu. Menu 39 adalah Peta Langit Dunia, Menu 40 adalah Laporan Astronomi, dan Menu 41

```

adalah HIJRI_DB Auto-Builder (Admin). Manual Book, highlight rilis, dan panduan navigasi pada halaman ini telah diselaraskan dengan struktur aktual tersebut."),

("[V8.0 - ENGINE] Auto-Switch Ephemeris Matahari & Bulan", "Sistem otomatis menggunakan de440s.bsp untuk tahun 1850-2149 dan beralih ke de441-new.bsp untuk tahun 1-1849 atau 2150-2999, sehingga perhitungan modern Matahari-Bulan memakai kernel ringkas berpresisi tinggi tanpa kehilangan jangkauan panjang aplikasi."),

("[V7.6 - LIVE ANIMASI] Sinkronisasi Stellarium", "Pada output Live Animasi ditambahkan tombol Sinkronisasi Stellarium untuk mengaktifkan Stellarium dan membandingkan data Matahari dan Bulan antara output Stellarium dengan hasil hitung sistem KHGT berbasis ephemeris DE441."),

("[V7.6 - LIVE ANIMASI] Tracker Benda Langit", "Pada output Live Animasi ditambahkan tombol Tracker untuk pelacakan posisi benda langit, baik menggunakan GPS maupun tanpa GPS."),

("[V7.5 - LIVE ANIMASI] Rasi Bintang & Planet", "Pembaruan pada Modul 36 (Live Animasi) dilengkapi visualisasi Rasi Bintang (Constellations) dan planet-planet tata surya secara dinamis dan real-time."),

("[V7.5 - WINAI] Asisten AI Universal", "Peningkatan kecerdasan WinAI Interceptor agar mampu mendeteksi niat (intent) pengguna dan mengeksekusi pertanyaan atau perintah perhitungan yang berkaitan dengan menu di dalam sistem."),

("[V7.5 - ANALEMMA] Grafik Kurva Matahari", "Peningkatan visual pada Modul 33 (Generator Analemma Matahari) yang menampilkan grafik kurva lintasan Matahari berbentuk angka 8."),

("[V7.5 - BUG FIXES] Gerhana, Kiblat & Optimalisasi", "Perbaikan bug pada rentang waktu komputasi di Modul 30 (Analisis Gerhana), perbaikan akurasi pada Modul 26 (Qiblah Direction and Times), serta berbagai penyesuaian stabilitas sistem."),

("[V7.4 - ENGINE] Penyederhanaan Ephemeris DE441", "Sistem menggunakan file ephemeris 'de441-new.bsp' untuk kalkulasi pada rentang waktu yang didukung aplikasi, sehingga pengelolaan basis ephemeris menjadi lebih sederhana."),

("[V7.3 - HIGHLIGHTS] Modul Fisis & Auto-Builder", "Modul Astrofotografi, Prediktor Pasang Surut, Evaluasi SQM Fajar, dan HIJRI_DB Auto-Builder untuk Administrator merupakan pengembangan yang diperkenalkan pada rilis sebelumnya.")

]

for key, val in updates:

key_render = self.translate_release_text_cached(key)

val_render = self._translate_release_piece_preserve_urls(val)

self.textbox.insert("end", " > ", "highlight")

self.textbox.insert("end", f"{key_render}: ", "highlight")

self.textbox.insert("end", f"{val_render}\n", "list_item")

===== BAGIAN 5: PANDUAN PENGGUNAAN =====

self.textbox.insert("end", garis_strip + "\n[5. PANDUAN OPERASIONAL & EXPORT]\n" + garis_strip + "\n", "subtitle")

---> 1. TAMBAHKAN KONFIGURASI TAG LINK DI SINI <---

import webbrowser

tb = self.textbox._textbox

HANYA gunakan link_aktif untuk styling (warna dan garis bawah)

JANGAN tambahkan tag_bind <Button-1> di sini agar tidak dobel klik

tb.tag_config("link_aktif", foreground="#00B0FF", underline=True)

```

guides = [
    ("DOWNLOAD BUKU KHGT TIMES", "Silahkan download aplikasi KHGT Times melalui link:
https://s.id/buku-khggtimes"),
    ("PYTHON MASTERY (TUTORIAL)", "Akses tutorial Python dasar melalui link:
https://kasmui.cloud/python/"),
    ("NAVIGASI MODUL", "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk
beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input
yang tidak relevan agar layar selalu rapi dan fokus."),
    ("INPUT LOKASI", "Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan
aplikasi, atau gunakan tombol '📍 Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar
sistem melacak titik GPS dan mendeteksi Timezone otomatis."),
    ("EKSEKUSI DATA", "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi
parameter lalu klik tombol biru besar '▶ PROSES DATA' di panel bawah. Tombol ini otomatis
bersembunyi jika Anda sedang memasuki modul Live/Real-time."),
    ("INTERAKSI GRAFIS", "Khusus pada kanvas 3D Matplotlib dan Simulasi 3D Spasial, biasakan diri
menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta
Roda Mouse (Scroll) untuk melakukan perbesaran (Zoom In/Out)."),
    ("SISTEM EXPORT", "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di
bawah tabel/kanvas): \n [📄 TXT] Menyimpan data tabel saat ini ke file Notepad.\n [🖨️ PNG]
Menyimpan kanvas visual / laporan ke gambar PNG.\n [📄 PDF] Mencetak dokumen laporan standar
akademik fpdf.\n [📊 CSV] Mengekstrak data komparasi 50 Tahun ke Excel.\n [🏠 RILIS] Tombol reset
darurat kembali ke Dokumentasi.")
]

```

---> 2. PERBAIKAN LOOPING AGAR TIAP LINK BERBEDA & BISA DIKLIK <---

for i, (key, val) in enumerate(guides):

```

# Panduan Release adalah isi Textbox, bukan widget UI. Karena itu
# terjemahkan key dan val secara eksplisit melalui subsistem cache
# release agar EN/AR benar-benar diterapkan tanpa mengubah URL.
key_render = self.translate_release_text_cached(key)
val_render = self.translate_release_text_cached(val)

```

```

self.textbox.insert("end", "> ", "highlight")
self.textbox.insert("end", f"{key_render} : ", "tech")

```

```

# Deteksi jika ada link URL di dalam teks
if "https://" in val_render:
    start_idx = val_render.find("https://")
    end_idx = val_render.find(" ", start_idx)

    if end_idx == -1:
        end_idx = len(val)

    link_url = val_render[start_idx:end_idx]
    bagian = val_render.split(link_url)

```

```

# Buat nama TAG UNIK untuk setiap link (contoh: url_0, url_1)
tag_unik = f"url_{i}"

# Teks sebelum link
self.textbox.insert("end", bagian[0], "list_item")

# Teks link yang disisipkan dengan TAG UNIK dan styling "link_aktif"
self.textbox.insert("end", link_url, ("list_item", tag_unik, "link_aktif"))

# Binding event klik kiri (<Button-1>) KHUSUS untuk tag unik ini menuju URL aslinya
self.textbox.tag_bind(tag_unik, "<Button-1>", lambda event, url=link_url:
webbrowser.open(url))

# Mengubah kursor jadi tangan saat di-hover KHUSUS untuk tag unik ini
tb.tag_bind(tag_unik, "<Enter>", lambda e: tb.configure(cursor="hand2"))
tb.tag_bind(tag_unik, "<Leave>", lambda e: tb.configure(cursor="xterm"))

# Teks setelah link
self.textbox.insert("end", bagian[1] + "\n", "list_item")
else:
    self.textbox.insert("end", f"{val_render}\n", "list_item")
# -----

# ===== FOOTER COPYLEFT =====
footer_text = "\n\n" + garis_sama + "\n\n"
footer_text += "Copyleft (c) Kasmui, 2026. All Left Reserved.\n"
footer_text += "This software is fully dedicated to the advancement of global Islamic astronomy and
science.\n\n"
footer_text += "KHGT Times Development Team \nLead Programmer & Developer: Kasmui
\nProgramming Tools (In order of role dominance): Gemini, DeepSeek, Qwen, ChatGPT/Copilot, and
Grok.\n"
footer_text += garis_sama + "\n"
self.textbox.insert("end", footer_text, "footer")

# 4. Kunci Textbox dan update judul layar utama
self.textbox.configure(state="disabled")
self.lbl_main_title.configure(text="KHGT Times V8.0")

# Widget CTk yang dibuat di dalam halaman rilis (tombol kalender, legenda, dll.)
# juga diterjemahkan setelah render, tetapi hanya pada wrapper halaman ini.
try:
    if getattr(self, "current_lang", "id") != "id" and hasattr(self, "demo_cal_wrapper"):
        self.update_ui_texts(self.demo_cal_wrapper)
except Exception:
    pass
finally:
    # Tulis cache JSON setelah halaman rilis selesai digambar.
    # Pada klik berikutnya, EN/AR cukup dibaca dari file ini.

```

```

        self.save_release_language_cache()
        self.save_output_language_cache()
        self._rendering_release_info = False

def create_input_row(self, parent, label_text, default_val):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)
    ctk.CTkLabel(row, text=label_text, font=("Segoe UI", 12)).pack(side="left")
    entry = ctk.CTkEntry(row, width=80, justify="right")
    entry.delete(0, 'end')
    entry.insert(0, default_val)
    entry.pack(side="right")
    return entry

def create_ymd_row(self, parent, dy, dm, dd):
    # --- PERBAIKAN: Menggunakan sistem Grid terpusat ---
    grid_frame = ctk.CTkFrame(parent, fg_color="transparent")
    grid_frame.pack(fill="x", padx=10, pady=(0, 10))

    # Baris 0: Label Petunjuk
    ctk.CTkLabel(grid_frame, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=0, pady=(0, 2))
    ctk.CTkLabel(grid_frame, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=1, pady=(0, 2))
    ctk.CTkLabel(grid_frame, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
column=2, pady=(0, 2))

    # Baris 1: Kotak Input
    d = ctk.CTkEntry(grid_frame, width=45, placeholder_text="dd", justify="center")
    d.delete(0, 'end')
    d.insert(0, dd)
    d.grid(row=1, column=0, padx=(0, 5))

    m = ctk.CTkEntry(grid_frame, width=45, placeholder_text="mm", justify="center")
    m.delete(0, 'end')
    m.insert(0, dm)
    m.grid(row=1, column=1, padx=5)

    y = ctk.CTkEntry(grid_frame, width=70, placeholder_text="yyyy", justify="center")
    y.delete(0, 'end')
    y.insert(0, dy)
    y.grid(row=1, column=2, padx=(5, 0))

    return y, m, d

def create_hms_row(self, parent, dh, dm, ds):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)

```

```

h = ctk.CTkEntry(row, width=35, placeholder_text="h")
h.delete(0, 'end')
h.insert(0, dh)
h.pack(side="left", padx=2)
m = ctk.CTkEntry(row, width=35, placeholder_text="m")
m.delete(0, 'end')
m.insert(0, dm)
m.pack(side="left", padx=2)
s = ctk.CTkEntry(row, width=35, placeholder_text="s")
s.delete(0, 'end')
s.insert(0, ds)
s.pack(side="left", padx=2)
return h, m, s

```

```

def auto_switch_ephemeris(self, target_year):

```

```

    """

```

```

    Pilih ephemeris otomatis berdasarkan tahun target.

```

```

    Rentang aplikasi tetap 1-2999 M:

```

```

    - 1850-2149 -> de440s.bsp (kernel pendek, fokus era modern)

```

```

    - lainnya -> de441-new.bsp (kernel panjang)

```

```

    """

```

```

    try:

```

```

        target_year = int(target_year)

```

```

    except (TypeError, ValueError):

```

```

        raise ValueError(f"Tahun tidak valid: {target_year}")

```

```

    if target_year < EPHEMERIS_LONG_MIN_YEAR or target_year > EPHEMERIS_LONG_MAX_YEAR:

```

```

        err_msg = f"Tahun {target_year} di luar jangkauan!"

```

```

        detail_msg = (

```

```

            f"Sistem mendukung rentang tahun {EPHEMERIS_LONG_MIN_YEAR} s/d "

```

```

            f"{EPHEMERIS_LONG_MAX_YEAR} Masehi.\n\n"

```

```

            f"Input Anda: {target_year}\n\n"

```

```

            f"Pemilihan ephemeris otomatis:\n"

```

```

            f"• {EPHEMERIS_SHORT_FILE}: {EPHEMERIS_SHORT_MIN_YEAR}-"

```

```

{EPHEMERIS_SHORT_MAX_YEAR} M\n"

```

```

            f"• {EPHEMERIS_LONG_FILE}: di luar rentang tersebut."

```

```

        )

```

```

        self.after(0, lambda: messagebox.showwarning(

```

```

            "Rentang Tahun Tidak Didukung", detail_msg

```

```

        ))

```

```

        self.after(0, lambda: self.lbl_status.configure(

```

```

            text="Tahun Tidak Valid!", text_color="#FF1744"

```

```

        ))

```

```

        self.after(0, lambda: self.btn_hitung.configure(state="normal"))

```

```

        raise ValueError(err_msg)

```

```

    desired = self._ephemeris_filename_for_year(target_year)

```

```

if self.eph is None or self.ephemeris_name != desired:
    # Pesan status dibuat melalui event loop agar aman bila dipanggil thread kalkulasi.
    self.after(0, lambda f=desired, y=target_year: self.lbl_status.configure(
        text=f"Mengalihkan ephemeris ke {f} ({y})...",
        text_color="#00E5FF"
    ))
    try:
        self._load_ephemeris_file(desired)
    except Exception as e:
        self.after(0, lambda err=str(e): messagebox.showerror(
            "Gagal Mengganti Ephemeris", err
        ))
        self.after(0, lambda: self.lbl_status.configure(
            text="Gagal memuat Ephemeris!", text_color="#FF1744"
        ))
        raise

    self.after(0, lambda f=desired: self.lbl_status.configure(
        text=f"Sistem Siap ({f})", text_color="#00E676"
    ))

return self.eph

def run_calculation(self):
    self.lbl_status.configure(text="Menghitung...", text_color="#FFAB40")
    self.btn_hitung.configure(state="disabled")

    self.textbox.configure(font=("Consolas", 13), wrap="none")

    mode = self.combo_mode.get()

    if "Altitude Chart" in mode:
        threading.Thread(target=self.calculate_chart_analyser, daemon=True).start()
        return

    if "Visibility Map" not in mode and "Analisis Gerhana" not in mode and "Live Animasi" not in mode
    and "Sistem Sun Moon" not in mode and "Simulasi Ephemeris" not in mode:
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memproses data astrometri...\n")
        self.textbox.configure(state="disabled")

    if "Visibility Hilal" in mode:
        threading.Thread(target=self.calculate_visibility, daemon=True).start()
    elif "Visibility Map" in mode:
        threading.Thread(target=self.calculate_visibility_map, daemon=True).start()
    elif "Analisis Hilal Global" in mode:

```



```

        threading.Thread(target=self.calculate_global_hilal, daemon=True).start()
    elif "Moonphase" in mode:
        threading.Thread(target=self.calculate_moonphase, daemon=True).start()
    elif "Sun Moon Ephemeris" in mode:
        threading.Thread(target=self.calculate_ephemeris, daemon=True).start()
    elif "Qiblah Direction" in mode:
        threading.Thread(target=self.calculate_qiblah, daemon=True).start()
    elif "Moon Times" in mode:
        threading.Thread(target=self.calculate_moontimes, daemon=True).start()
    elif "Sun Times" in mode:
        threading.Thread(target=self.calculate_suntimes, daemon=True).start()
    elif "Prayer Times" in mode:
        threading.Thread(target=self.calculate_prayertimes, daemon=True).start()
    elif "Konversi" in mode:
        threading.Thread(target=self.calculate_conversion, daemon=True).start()
    elif "Qibla Time" in mode:
        threading.Thread(target=self.calculate_qiblatime, daemon=True).start()
    elif "Analisis Gerhana" in mode:
        tahun = int(self.combo_tahun_gerhana.get())
        threading.Thread(target=self._calc_gerhana_thread, args=(tahun,), daemon=True).start()
    elif "Kota Pertama" in mode:
        threading.Thread(target=self.calculate_first_point, daemon=True).start()
    elif "Equinox" in mode:
        threading.Thread(target=self.calculate_seasons, daemon=True).start()
    elif "Planetary Times" in mode:
        threading.Thread(target=self.calculate_planetary_times, daemon=True).start()
    # ---> URUTAN YANG BENAR DI TOMBOL PROSES (SYAWAL HARUS DI ATAS) <---
    elif "Komparasi 50 Tahun (Syawal)" in mode:
        self.analisis_komparasi_syawal_50_tahun()

    # ---> TAMBAHKAN ROUTING EKSEKUSI ZULHIJAH DI SINI <---
    elif "Komparasi 50 Tahun (Zulhijah)" in mode:
        self.analisis_komparasi_zulhijah_50_tahun()

    elif "Komparasi 50 Tahun (12 Bulan)" in mode:
        self.analisis_komparasi_global_50_tahun()

    elif "Komparasi 50 Tahun" in mode:
        self.analisis_komparasi_ramadhan_50_tahun()

    elif "Tabel Tinggi Hilal 50 Thn (Syawal)" in mode:
        threading.Thread(target=self._proses_tabel_ketinggian_syawal_50_tahun, daemon=True).start()

    # ---> TAMBAHKAN ROUTING EKSEKUSI TABEL TINGGI ZULHIJAH DI SINI <---
    elif "Tabel Tinggi Hilal 50 Thn (Zulhijah)" in mode:
        threading.Thread(target=self._proses_tabel_ketinggian_zulhijah_50_tahun, daemon=True).start()

```

```

elif "Tabel Tinggi Hilal" in mode:
    threading.Thread(target=self._proses_tabel_ketinggian_50_tahun, daemon=True).start()

# ---> SISIPKAN ROUTING MENU 25 DI SINI <---
elif "Tabel Elongasi Hilal 50 Thn (Syawal)" in mode:
    threading.Thread(target=self._proses_tabel_elongasi_syawal_50_tahun, daemon=True).start()

# ---> TAMBAHKAN ROUTING EKSEKUSI TABEL ELONGASI ZULHIJAH DI SINI <---
elif "Tabel Elongasi Hilal 50 Thn (Zulhijah)" in mode:
    threading.Thread(target=self._proses_tabel_elongasi_zulhijah_50_tahun, daemon=True).start()

# Pastikan ini berada di bawah Menu 25
elif "Tabel Elongasi Hilal" in mode:
    threading.Thread(target=self._proses_tabel_elongasi_50_tahun, daemon=True).start()

elif "Mizwala" in mode:
    threading.Thread(target=self.calculate_mizwala, daemon=True).start()

elif "Auto-Builder" in mode:
    threading.Thread(target=self._build_hijri_db_thread, daemon=True).start() # <--- TAMBAHKAN
INI

# ---> TAMBAHKAN ROUTING MENU 31 DI SINI <---
elif "Status Kriteria Batas" in mode:
    threading.Thread(target=self.calculate_kriteria_batas, daemon=True).start()

# ---> TAMBAHAN PEMICU MODUL 31 <---
elif "Astrofotografi" in mode:
    threading.Thread(target=self.calculate_astrofotografi, daemon=True).start()

elif "Pasang Surut" in mode:
    threading.Thread(target=self.calculate_pasang_surut, daemon=True).start()

# ---> TAMBAHAN PEMICU MODUL 33 <---
elif "Fajar Shadiq" in mode:
    threading.Thread(target=self.calculate_fajar_sqm, daemon=True).start()

elif "Analemma" in mode:
    threading.Thread(target=self.calculate_analemma, daemon=True).start()

# MODUL TAMBAHAN 5: ANALISIS HILAL GLOBAL ITERATIF (EPHEM)
# =====
def calculate_global_hilal(self):
    # ---> PERBAIKAN UI: Mengembalikan tampilan ke mode Tabel (Teks) <---
    def _siapkan_ui_tabel():
        if hasattr(self, 'frame_alaska_out'):
            self.frame_alaska_out.grid_remove() # Sembunyikan kurva alaska
            self.textbox.grid(row=1, column=0, sticky="nsew") # Munculkan tabel

```

```

        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Menganalisis data hilal global lintas benua dan samudra (PKG 1 & PKG
2)...\\nMohon tunggu...\\n")
        self.textbox.configure(state="disabled")

    self.after(0, _siapkan_ui_tabel)

    try:
        import math
        import pytz
        import datetime
        import numpy as np
        from skyfield import almanac

        y = int(self.entry_gha_year.get())
        m = int(self.entry_gha_month.get())
        d = int(self.entry_gha_day.get())
        time_str = self.entry_gha_time.get()
        if not time_str: time_str = "12:00:00"

        h, mn, s = map(int, time_str.split(':'))
        tanggal_referensi_utc = f"{y}/{m:02d}/{d:02d} {time_str}"

        self.auto_switch_ephemeris(y)
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

        # Waktu referensi UTC sebagai patokan mencari ijtimak
        t_ref = self.ts.utc(y, m, d, h, mn, s)
        t_ref_tt = t_ref.tt.item() if hasattr(t_ref.tt, 'item') else t_ref.tt

        # --- 1. SINKRONISASI PENCARIAN IJTIMAK ---
        t_start_conj = self.ts.tt_jd(t_ref_tt - 20.0)
        t_end_conj = self.ts.tt_jd(t_ref_tt + 20.0)

        t_phases, y_phases = almanac.find_discrete(t_start_conj, t_end_conj,
almanac.moon_phases(self.eph))
        t_ijtimak = None
        jarak_terdekat = float('inf')

        for t_ev, phase in get_safe_events(t_phases, y_phases):
            if phase == 0: # 0 = Konjungsi / New Moon
                ev_tt = t_ev.tt.item() if hasattr(t_ev.tt, 'item') else t_ev.tt
                selisih_hari = abs(ev_tt - t_ref_tt)
                if selisih_hari < jarak_terdekat:
                    jarak_terdekat = selisih_hari

```

```

t_ijtimak = t_ev

if t_ijtimak is None: raise ValueError("Ijtimak tidak ditemukan dalam rentang waktu ini.")

str_ijtimak = t_ijtimak.utc_datetime().strftime("%Y/%m/%d %H:%M:%S")
ti_tt = t_ijtimak.tt.item() if hasattr(t_ijtimak.tt, 'item') else t_ijtimak.tt

hasil_kalkulasi = []

# LOOP DATABASE KOTA (HANYA UNTUK DITAMPILKAN DI TABEL)
for negara, kota_dict in CITY_DB.items():
    for nama_kota, koordinat in kota_dict.items():
        lintang, bujur = koordinat
        loc = wgs84.latlon(lintang, bujur, elevation_m=0)
        tz_approx = int(self.get_tz_from_lon(bujur))

        t0_ss = self.ts.utc(y, m, d, 0 - tz_approx)
        t1_ss = self.ts.utc(y, m, d, 24 - tz_approx)

        t_ss_evs, y_ss_evs = almanac.find_discrete(t0_ss, t1_ss, almanac.sunrise_sunset(self.eph,
loc))

        t_sunset = None
        for t_ev, is_sunrise in get_safe_events(t_ss_evs, y_ss_evs):
            if not is_sunrise:
                t_sunset = t_ev
                break

        if t_sunset is None:
            hasil_kalkulasi.append({
                'negara': negara, 'kota': nama_kota, 'waktu_sort': 999999.0,
                'baris_teks': f'{"negara[:15]:<15} | {"nama_kota[:14]:<14} | {"Anomali Ekstrem":<16} | {"-':<9} | {"-':<8} | {"-':<8} | {"-':<8} | {"-':<8} | Lintang Tinggi (Midnight Sun)",
                'is_khgt': False, 't_sunset_obj': None, 'lon': bujur, 'lat': lintang
            })
            continue

        ts_tt = t_sunset.tt.item() if hasattr(t_sunset.tt, 'item') else t_sunset.tt

        geo_earth = earth.at(t_sunset)
        m_app_geo = geo_earth.observe(moon).apparent()
        s_app_geo = geo_earth.observe(sun).apparent()

        sep_geo = s_app_geo.separation_from(m_app_geo).degrees
        elong_geo = sep_geo.item() if hasattr(sep_geo, 'item') else sep_geo

        ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset)
        gast = t_sunset.gast

```

```

lst_deg = (gast * 15.0) + bujur

ra_h = ra_m.hours.item() if hasattr(ra_m.hours, 'item') else ra_m.hours
dec_r = dec_m.radians.item() if hasattr(dec_m.radians, 'item') else dec_m.radians

ha_deg = lst_deg - (ra_h * 15.0)
ha_rad = math.radians(ha_deg)
lat_rad = math.radians(lintang)

sin_alt_geo = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)
sin_alt_geo = max(-1.0, min(1.0, sin_alt_geo))
alt_geo = math.degrees(math.asin(sin_alt_geo))

topo_obs = (earth + loc).at(t_sunset)
app_moon_topo = topo_obs.observe(moon).apparent()
app_sun_topo = topo_obs.observe(sun).apparent()

alt_moon_topo_obj, _ = app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)
alt_moon_topo = alt_moon_topo_obj.degrees

sep_topo = app_sun_topo.separation_from(app_moon_topo).degrees
elong_topo = sep_topo.item() if hasattr(sep_topo, 'item') else sep_topo

umur_bulan_desimal = (ts_tt - ti_tt) * 24.0

if umur_bulan_desimal < 0:
    format_umur = "B. Ijtimak"
    status_visibilitas = "Negatif (Bulan Tua)"
else:
    jam = int(umur_bulan_desimal)
    menit = int((umur_bulan_desimal - jam) * 60)
    format_umur = f"{jam}j {menit}m"

    if alt_geo >= 5.0 and elong_geo >= 8.0: status_visibilitas = "Memenuhi Kriteria KHGT"
    elif alt_moon_topo >= 3.0 and elong_geo >= 6.4: status_visibilitas = "Memenuhi Kriteria
MABIMS"
    elif alt_geo > 0 or alt_moon_topo > 0: status_visibilitas = "Wujudul Hilal"
    else: status_visibilitas = "Negatif (Bawah Ufuk)"

dt_sunset_lokal = t_sunset.utc_datetime() + datetime.timedelta(hours=tz_approx)
sunset_str = dt_sunset_lokal.strftime("%d/%m/%Y %H:%M")

alt_g_str, elong_g_str = f"{alt_geo:.2f}°", f"{elong_geo:.2f}°"
alt_t_str, elong_t_str = f"{alt_moon_topo:.2f}°", f"{elong_topo:.2f}°"

```

```

        baris_teks = f"{negara[:15]:<15} | {nama_kota[:14]:<14} | {sunset_str[:16]:<16} |
{format_umur[:9]:<9} | {alt_g_str[:8]:<8} | {elong_g_str[:8]:<8} | {alt_t_str[:8]:<8} | {elong_t_str[:8]:<8}
| {status_visibilitas}"

```

```

is_khgt_valid = umur_bulan_desimal > 0 and alt_geo >= 5.0 and elong_geo >= 8.0

```

```

hasil_kalkulasi.append({
    'negara': negara, 'kota': nama_kota, 'waktu_sort': ts_tt, 'baris_teks': baris_teks,
    'is_khgt': is_khgt_valid, 't_sunset_obj': t_sunset, 'lon': bujur, 'lat': lintang,
    'alt_geo': alt_geo, 'elong_geo': elong_geo
})

```

```

hasil_kalkulasi.sort(key=lambda x: (x['negara'].lower(), x['kota'].lower()))

```

```

# =====
# EVALUASI PKG 1 & PKG 2 (MENYELURUH DARATAN DAN LAUTAN)
# =====
import ephem
obs_date_utc = datetime.date(y, m, d)
batas_00_utc_ephem = ephem.Date(datetime.datetime(obs_date_utc.year,
obs_date_utc.month, obs_date_utc.day) + datetime.timedelta(days=1))

```

```

# Hitung Fajar Gisborne untuk PKG 2
gisborne = ephem.Observer()
gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
gisborne.elevation, gisborne.pressure = 0, 0
gisborne.horizon = '-18'
matahari_ephem = ephem.Sun()
gisborne.date = ephem.Date(batas_00_utc_ephem - 0.5)
try:
    fajar_gisborne = gisborne.next_rising(matahari_ephem, use_center=True)
except:
    fajar_gisborne = ephem.Date(batas_00_utc_ephem + 0.6)

```

```

dt_utc_ijtima = t_ijtimak.utc_datetime()
is_ijtima_before_fajar = ephem.Date(dt_utc_ijtima) < fajar_gisborne

```

```

pkg1_met = False
pkg2_met = False
pkg_reason = ""
titik_pertama_global = None

```

```

# 1. EVALUASI PKG 1 (Dimanapun di dunia, Daratan / Lautan asalkan Sunset < 00:00 UTC)
# Kita lakukan scanning grid 10 derajat mencakup daratan dan lautan
for lon_g in range(180, -181, -10):
    if pkg1_met: break
    for lat_g in range(-60, 61, 15):
        loc_g = wgs84.latlon(lat_g, lon_g)

```

```

est_sunset_utc = 12.0 - (lon_g / 15.0)
t0_g = self.ts.utc(y, m, d, est_sunset_utc - 2)
t1_g = self.ts.utc(y, m, d, est_sunset_utc + 2)

try:
    t_evs, y_evs = almanac.find_discrete(t0_g, t1_g, almanac.sunrise_sunset(self.eph, loc_g))
    for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
        if not is_sr:
            t_ss_sky = self.ts.tt_jd(t_ev)
            if ephemer.Date(t_ss_sky.utc_datetime()) < batas_00_utc_ephem:
                obs_center = earth.at(t_ss_sky)
                m_app = obs_center.observe(moon).apparent()
                s_app = obs_center.observe(sun).apparent()
                elong_g = s_app.separation_from(m_app).degrees

                if elong_g >= 8.0:
                    ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
                    ha_deg = (t_ss_sky.gast * 15.0) + lon_g - (ra_m.hours * 15.0)
                    sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_g)) +
                    math.cos(dec_m.radians) * math.cos(math.radians(lat_g)) * math.cos(math.radians(ha_deg))
                    alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

                    if alt_g >= 5.0:
                        pkg1_met = True
                        pkg_reason = f"PKG 1 (Titik Koordinat: Lintang {lat_g}°, Bujur {lon_g}°
[Darat/Lautan])"
                        titik_pertama_global = f"Lautan/Daratan (Lat: {lat_g}°, Lon: {lon_g}°) | Sunset
UTC: {t_ss_sky.utc_strftime('%H:%M:%S')}"
                        break
            except Exception:
                pass

# 2. EVALUASI PKG 2 (Hanya Daratan Amerika, Sunset >= 00:00 UTC, Ijtima < Fajar NZ)
if not pkg1_met:
    negara_mainland_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan"]
    # Memfilter kota kepulauan (Hawaii, Polinesia, dll)
    kota_kepulauan = ["Honolulu", "Papeete", "Alofi", "Pago Pago", "Fernando de Noronha",
"Nuuk", "San Juan", "Santo Domingo", "Havana", "Praia"]

    for hasil in hasil_kalkulasi:
        if pkg2_met: break
        if hasil['t_sunset_obj'] is not None:
            is_mainland_amerika = hasil['negara'] in negara_mainland_amerika and hasil['kota'] not in
kota_kepulauan
            if is_mainland_amerika:
                if ephemer.Date(hasil['t_sunset_obj'].utc_datetime()) >= batas_00_utc_ephem:
                    if hasil['alt_geo'] >= 5.0 and hasil['elong_geo'] >= 8.0:
                        if is_ijtima_before_fajr:

```

```

        pkg2_met = True
        pkg_reason = f"PKG 2 (Daratan Benua Amerika: {hasil['kota']}, {hasil['negara']}) &
ljtima < Fajar NZ)"
        titik_pertama_global = f"{hasil['kota']}, {hasil['negara']} | Sunset UTC:
{hasil['t_sunset_obj'].utc_strftime('%H:%M:%S')}"
    else:
        pkg_reason = f"PKG 2 GAGAL (Kriteria tembus di Amerika: {hasil['kota']}, TAPI
ljtima > Fajar NZ)"

    # Penetapan Nama Bulan dan Tahun Hijriah untuk Output
    nm_tt_val = self.ts.from_datetime(dt_utc_ljtima.replace(tzinfo=pytz.utc)).tt
    h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
    hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
    hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"

    tgl_1_mas = ephemer.Date(batas_00_utc_ephem)
    gy, gm, gd, _, _ = tgl_1_mas.tuple()
    masehi_str_besok = f"{int(gd):02d} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"

    tgl_lusa = ephemer.Date(batas_00_utc_ephem + 1.0)
    gy2, gm2, gd2, _, _ = tgl_lusa.tuple()
    masehi_str_lusa = f"{int(gd2):02d} {BULAN_MASEHI[int(gm2)-1]} {format_tahun_aman(int(gy2))}"

    if pkg1_met:
        status_khgt = f"PKG 1 Terpenuhi (Global Daratan/Lautan, Sunset < 00:00 UTC)\n* Awal Bulan
Baru : BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_besok})"
    elif pkg2_met:
        status_khgt = f"PKG 2 Terpenuhi (Daratan Benua Amerika & ljtima < Fajar NZ)\n* Awal Bulan
Baru : BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_besok})"
    else:
        status_khgt = f"Gagal PKG 1 & PKG 2 ({pkg_reason if pkg_reason else 'Kriteria tidak
terpenuhi'})\n* Awal Bulan Baru : LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_lusa})"

    if titik_pertama_global is None:
        info_kota_pertama = " 🌞 TITIK PERTAMA KHGT : Tidak Ditemukan (Kriteria 5-8 belum
terpenuhi di seluruh dunia)"
    else:
        info_kota_pertama = f" 🌞 TITIK PERTAMA KHGT : {titik_pertama_global}\n* Status Global :
{status_khgt}"

    # ---> PENYUSUNAN HEADER DAN OUTPUT AKHIR <---
    output_lines = []
    header = f"{self.get_header(138)}\n"
    header += f"[ Analisis Hilal Global (KHGT, MABIMS, Wujudul Hilal) - Skyfield ]'.center(138)}\n\n"
    header += f"* Tanggal Referensi Pencarian (UTC): {tanggal_referensi_utc}\n"
    header += f"* Waktu ljtimak/Konjungsi Terdekat (UTC): {str_ijtimak}\n"
    header += f"* {info_kota_pertama}\n"

```



```

        header += f"* Catatan: Tabel di bawah diurutkan Berdasarkan Abjad Negara, lalu Abjad Kota.\n"
        header += "="*138 + "\n"
        header += f"{'Negara':<15} | {'Kota':<14} | {'Sunset (LT)':<16} | {'Umur Bln':<9} | {'Alt(Geo)':<8} | {'Eln(Geo)':<8} | {'Alt(Top)':<8} | {'Eln(Top)':<8} | Status\n"
        header += "-"*138

    output_lines.append(header)

    for hasil in hasil_kalkulasi:
        output_lines.append(hasil['baris_teks'])

    output_lines.append("="*138)
    self.after(0, self.display_result, "\n".join(output_lines))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

# =====
# MODUL TAMBAHAN 11: ANALISIS GERHANA ALGORITMIK
# =====
def setup_gerhana_out_frame(self):
    self.frame_gerhana_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")

    style = ttk.Style(self)
    style.theme_use("clam")

    # 1. Mengubah font Judul Kolom (Heading) menjadi ukuran 13
    style.configure("Gerhana.Treeview.Heading", font=('Segoe UI', 13, 'bold'), background="#2b5797",
    foreground="white")

    # 2. Mengubah font Isi Tabel menjadi ukuran 13 dan tinggi baris (rowheight) menjadi 32
    style.configure("Gerhana.Treeview", font=('Segoe UI', 13), rowheight=32, background="#1e1e1e",
    foreground="white", fieldbackground="#1e1e1e")

    style.map('Gerhana.Treeview', background=[('selected', '#0078D7')], foreground=[('selected',
    'white')])

    frame_tabel = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
    frame_tabel.pack(fill="both", expand=True, pady=(0, 10))

    kolom_gerhana = ("objek", "jenis", "mulai", "puncak", "akhir", "wilayah_global", "indo_vis")
    self.tabel_gerhana = ttk.Treeview(frame_tabel, columns=kolom_gerhana, show="headings",
    style="Gerhana.Treeview")

    self.tabel_gerhana.heading("objek", text="Objek")
    self.tabel_gerhana.heading("jenis", text="Jenis Gerhana")
    self.tabel_gerhana.heading("mulai", text="Kontak Awal (WIB)")

```

```

self.tabel_gerhana.heading("puncak", text="Puncak (WIB)")
self.tabel_gerhana.heading("akhir", text="Kontak Akhir (WIB)")
self.tabel_gerhana.heading("wilayah_global", text="Karakteristik & Wilayah")
self.tabel_gerhana.heading("indo_vis", text="Visibilitas (WIB Siang/Malam)")

self.tabel_gerhana.column("objek", width=80, anchor="center")
self.tabel_gerhana.column("jenis", width=120, anchor="center")
self.tabel_gerhana.column("mulai", width=130, anchor="center")
self.tabel_gerhana.column("puncak", width=130, anchor="center")
self.tabel_gerhana.column("akhir", width=130, anchor="center")
self.tabel_gerhana.column("wilayah_global", width=220, anchor="w")
self.tabel_gerhana.column("indo_vis", width=260, anchor="w")

self.tabel_gerhana.tag_configure('ganjil', background='#2b2b2b')
self.tabel_gerhana.tag_configure('genap', background='#1e1e1e')

sb_gerhana = ttk.Scrollbar(frame_tabel, orient="vertical", command=self.tabel_gerhana.yview)
self.tabel_gerhana.configure(yscroll=sb_gerhana.set)

self.tabel_gerhana.pack(side="left", fill="both", expand=True)
sb_gerhana.pack(side="right", fill="y")

frame_btn = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
frame_btn.pack(pady=10)

btn_detail = ctk.CTkButton(frame_btn, text="🔍 Lihat Detail Lokal Saat Puncak", font=("Segoe UI",
12, "bold"), command=self.tampilkan_detail_lokal_gerhana)
btn_detail.pack(side="left", padx=10)

self.btn_kml = ctk.CTkButton(frame_btn, text="📄 Export Jalur Totalitas/Anularitas ke KML",
font=("Segoe UI", 12, "bold"), fg_color="#F57C00", hover_color="#EF6C00",
command=self.export_kml_solar)
self.btn_kml.pack(side="left", padx=10)

self.btn_simulasi_gerhana = ctk.CTkButton(frame_btn, text="🌐 Simulasi Visual Gerhana",
font=("Segoe UI", 12, "bold"), fg_color="#673AB7", hover_color="#512DA8",
command=self.buka_simulator_gerhana)
self.btn_simulasi_gerhana.pack(side="left", padx=10)

def insert_gerhana_wrapped_row(self, values, tags):
    char_limits = [10, 16, 18, 18, 18, 30, 36]
    wrapped_cols = []
    max_lines = 1

    for i, val in enumerate(values):
        wrapped = textwrap.wrap(str(val), width=char_limits[i])
        if not wrapped: wrapped = [""]

```

```

        wrapped_cols.append(wrapped)
        if len(wrapped) > max_lines: max_lines = len(wrapped)

    for line_idx in range(max_lines):
        row_data = [col[line_idx] if line_idx < len(col) else "" for col in wrapped_cols]
        self.tabel_gerhana.insert("", "end", values=row_data, tags=tags)

    self.tabel_gerhana.insert("", "end", values=["", "", "", "", "", "", ""], tags=tags)

def analisis_visibilitas_indonesia(self, jam_utc):
    if 10 <= jam_utc <= 14: return "Terlihat Jelas: Papua, Maluku, Sulawesi. (Sumatra/Jawa saat terbit)"
    elif 15 <= jam_utc <= 19: return "Terlihat Jelas Seluruh Indonesia (Tengah Malam)"
    elif 20 <= jam_utc <= 22: return "Terlihat Jelas: Sumatera, Jawa, Kalimantan. (Indonesia Timur saat
    terbenam)"
    else: return "Tidak Terlihat (Terjadi Siang Hari di Indonesia)"

def _calc_gerhana_thread(self, tahun):
    try:
        self.auto_switch_ephemeris(tahun)
        # Gunakan ts.utc murni tanpa timedelta agar aman untuk tahun berapapun
        t0 = self.ts.utc(tahun, 1, 1)
        t1 = self.ts.utc(tahun, 12, 31, 23, 59, 59)
        all_eclipses = []
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

        # Titik Referensi Indonesia (Diambil Tengah: Balikpapan) untuk cek visibilitas lokal
        indo_loc = earth + wgs84.latlon(-1.2, 116.8)

        # =====
        # PRE-KOMPUTASI DATABASE NEGARA (Agar Scanning Super Cepat)
        # =====
        perwakilan_locs = {}
        prov_indo = ["Aceh", "Bali", "Bangka Belitung", "Banten", "Bengkulu", "DI Yogyakarta", "DKI
        Jakarta", "Gorontalo", "Jambi", "Jawa Barat", "Jawa Tengah", "Jawa Timur", "Kalimantan Barat",
        "Kalimantan Selatan", "Kalimantan Tengah", "Kalimantan Timur", "Kalimantan Utara", "Kepulauan Riau",
        "Lampung", "Maluku", "Maluku Utara", "NTB", "NTT", "Papua", "Papua Barat", "Papua Barat Daya",
        "Papua Selatan", "Papua Tengah", "Papua Pegunungan", "Riau", "Sulawesi Barat", "Sulawesi Selatan",
        "Sulawesi Tengah", "Sulawesi Tenggara", "Sulawesi Utara", "Sumatera Barat", "Sumatera Selatan",
        "Sumatera Utara"]

        for region, cities in CITY_DB.items():
            if not cities: continue
            nama_tampil = region
            if region in prov_indo:
                nama_tampil = "Indonesia"
            elif "Amerika Selatan" in region: nama_tampil = "Amerika Sel."
            elif "Afrika" in region: nama_tampil = "Afrika"
            elif "Asia Selatan" in region: nama_tampil = "Timteng & Asia Sel."

```

```

elif "Eropa" in region or "Inggris" in region: nama_tampil = "Eropa"

first_city = list(cities.keys())[0]
lat, lon = cities[first_city]
if nama_tampil not in perwakilan_locs:
    perwakilan_locs[nama_tampil] = []

if len(perwakilan_locs[nama_tampil]) < 3:
    loc_obj = earth + wgs84.latlon(lat, lon)
    perwakilan_locs[nama_tampil].append(loc_obj)

# Cari seluruh fase bulan di tahun tersebut (0 = Ijtimak/New Moon, 2 = Purnama/Full Moon)
t_phases, y_phases = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))

if t_phases is None:
    raise ValueError(f"Data Ephemeris tidak mencakup fase bulan untuk tahun {tahun}.")

# --- 1. PENCARIAN GERHANA BULAN (MANUAL VECTOR MATH - MENGGANTIKAN eclipselib) ---
t_full_moons = [t for t, phase in zip(t_phases, y_phases) if phase == 2]
for t_fm in t_full_moons:
    tt_array = np.linspace(t_fm.tt - 0.25, t_fm.tt + 0.25, 300)
    t_arr = self.ts.tt_jd(tt_array)
    e_pos = earth.at(t_arr)
    s_app = e_pos.observe(sun).apparent()
    m_app = e_pos.observe(moon).apparent()

    seps_from_sun = s_app.separation_from(m_app).degrees
    seps_from_shadow = 180.0 - seps_from_sun

    min_idx = np.argmin(seps_from_shadow)
    min_sep = seps_from_shadow[min_idx]

    t_p = t_arr[min_idx]
    s_dist = earth.at(t_p).observe(sun).apparent().distance().km
    m_dist = earth.at(t_p).observe(moon).apparent().distance().km

    sd_m = math.degrees(math.asin(1737.4 / m_dist))
    pi_m = math.degrees(math.asin(6378.14 / m_dist))
    sd_s = math.degrees(math.asin(696000.0 / s_dist))
    pi_s = 8.794 / 3600.0

    r_umbra = (pi_m + pi_s - sd_s) * 1.02
    r_penumbra = (pi_m + pi_s + sd_s) * 1.02

# Deteksi jika masuk bayangan bumi
if min_sep < (r_penumbra + sd_m):
    if min_sep < (r_umbra - sd_m): jenis = "Total"
    elif min_sep < (r_umbra + sd_m): jenis = "Sebagian (Partial)"

```

```

else: jenis = "Penumbra"

m_alt_indo, _, _ = indo_loc.at(t_p).observe(moon).apparent().altaz()
indo_vis = "Terlihat Jelas (Bulan di atas ufuk)" if m_alt_indo.degrees > 0 else "Tidak Terlihat
(Siang di Indonesia)"

wilayah_terlihat = set()
for nama_wil, list_loc in perwakilan_locs.items():
    for loc in list_loc:
        alt_obj, _, _ = loc.at(t_p).observe(moon).apparent().altaz()
        if alt_obj.degrees > 0:
            wilayah_terlihat.add(nama_wil)
            break

wil_str = ", ".join(sorted(list(wilayah_terlihat))) if wilayah_terlihat else "Hanya terlihat di
Samudra Terpencil"

all_eclipses.append({
    'objek': 'Bulan',
    'jenis': jenis,
    't_peak': t_p,
    't_mulai': self.ts.tt_jd(t_p.tt - 0.15),
    't_akhir': self.ts.tt_jd(t_p.tt + 0.15),
    'wilayah': wil_str,
    'indo_vis': indo_vis
})

# --- 2. PENCARIAN GERHANA MATAHARI ---
t_new_moons = [t for t, phase in zip(t_phases, y_phases) if phase == 0]
for t_nm in t_new_moons:
    tt_array = np.linspace(t_nm.tt - 0.25, t_nm.tt + 0.25, 300)
    t_arr = self.ts.tt_jd(tt_array)
    e_pos = earth.at(t_arr)
    seps =
e_pos.observe(sun).apparent().separation_from(e_pos.observe(moon).apparent()).degrees
    min_idx = np.argmin(seps)

    if seps[min_idx] < 1.6:
        t_p = t_arr[min_idx]
        dist_s = earth.at(t_p).observe(sun).apparent().distance().km
        dist_m = earth.at(t_p).observe(moon).apparent().distance().km
        sd_s = math.degrees(math.asin(696000.0 / dist_s))
        sd_m = math.degrees(math.asin(1737.4 / dist_m))

        jenis = "Total" if sd_m > sd_s else "Cincin (Annular)"
        if seps[min_idx] > abs(sd_s - sd_m): jenis = "Sebagian (Partial)"

    s_app_indo = indo_loc.at(t_p).observe(sun).apparent()

```

```

m_app_indo = indo_loc.at(t_p).observe(moon).apparent()
alt_s_indo, _, _ = s_app_indo.altaz()
sep_indo = s_app_indo.separation_from(m_app_indo).degrees

if alt_s_indo.degrees > 0:
    if sep_indo < (sd_s + sd_m + 0.6):
        indo_vis = "Terlihat (Bayangan Gerhana melintasi Indonesia)"
    else:
        indo_vis = "Tidak Terlihat (Meleset dari lintang Indonesia)"
else:
    indo_vis = "Tidak Terlihat (Terjadi saat Malam di Indonesia)"

wilayah_terlihat = set()
for nama_wil, list_loc in perwakilan_locs.items():
    for loc in list_loc:
        s_app = loc.at(t_p).observe(sun).apparent()
        alt_s_wil, _, _ = s_app.altaz()
        if alt_s_wil.degrees > 0:
            m_app = loc.at(t_p).observe(moon).apparent()
            sep_wil = s_app.separation_from(m_app).degrees
            if sep_wil < (sd_s + sd_m + 0.8):
                wilayah_terlihat.add(nama_wil)
            break

wil_str = ", ".join(sorted(list(wilayah_terlihat))) if wilayah_terlihat else "Hanya melintasi
Samudra / Antartika"

all_eclipses.append({
    'objek': 'Matahari',
    'jenis': jenis,
    't_peak': t_p,
    't_mulai': self.ts.tt_jd(t_p.tt - 0.12),
    't_akhir': self.ts.tt_jd(t_p.tt + 0.12),
    'wilayah': wil_str,
    'indo_vis': indo_vis
})

# Jika data gerhana benar-benar kosong
if len(all_eclipses) == 0:
    self.after(0, lambda: messagebox.showinfo("Informasi Ephemeris",
        f"Tidak ada peristiwa gerhana yang ditemukan pada tahun {tahun}.\n\n"
        "Hal ini wajar jika terjadi pada tahun tertentu, atau pastikan file ephemeris 'de440s.bsp' /
'de441-new.bsp' sesuai rentang tidak korup. "
        "Rentang yang dipakai aplikasi adalah 1 hingga 2999 M.))
    self.after(0, lambda: self.lbl_status.configure(text=f"Tidak ada data gerhana {tahun}",
text_color="#FFAB40"))
else:
    all_eclipses.sort(key=lambda x: x['t_peak'].tt)

```

```

        self.after(0, self._post_hitung_gerhana, all_eclipses, tahun)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Gagal Scan Gerhana Tahun {tahun}: {str(e)}\n\nPastikan format tahun benar dan rentang berada di dalam file .bsp (Misal: 1 hingga 3000).\n\n{traceback.format_exc()}")

def _post_hitung_gerhana(self, all_eclipses, tahun):
    self.tabel_gerhana.delete(*self.tabel_gerhana.get_children())
    count = 0

    def format_waktu(t_obj):
        if t_obj is None: return "---"
        t_local = self.ts.tt_jd(t_obj.tt + 7.0/24.0) # Konversi ke WIB
        y, m, d, h, mn, s = t_local.utc
        yr = int(y)
        txt_yr = f"{yr:04d}" if yr > 0 else f"{abs(yr-1):04d} SM"
        return f"{int(d):02d}-{int(m):02d}-{txt_yr} {int(h):02d}:{int(mn):02d}"

    for ev in all_eclipses:
        w_mulai = format_waktu(ev['t_mulai'])
        w_puncak = format_waktu(ev['t_peak'])
        w_akhir = format_waktu(ev['t_akhir'])

        tag = 'ganjil' if count % 2 == 0 else 'genap'
        self.insert_gerhana_wrapped_row((
            ev['objek'], ev['jenis'], w_mulai, w_puncak, w_akhir, ev['wilayah'], ev['indo_vis']
        ), (tag,))
        count += 1

    self.lbl_status.configure(text=f"Analisis Gerhana {tahun} Selesai", text_color="#00E676")
    self.btn_hitung.configure(state="normal")

def _parse_waktu_tabel_gerhana(self, waktu_str):
    """Fungsi Anti-Crash untuk mengkonversi string tabel kembali ke format Waktu Skyfield (Kebal Tahun SM)"""
    import re
    # Format string dari tabel: "dd-mm-yyyy HH:MM" atau "dd-mm-yyyy SM HH:MM"
    match = re.match(r"(\d+)-(\d+)-(\d+)\s*(SM)?\s*(\d+):(\d+)", waktu_str)
    if not match: return None
    d, m, y, is_sm, h, mn = match.groups()
    y = int(y)
    if is_sm: y = -(y - 1)
    # Kurangi 7 Jam karena format tabel adalah WIB (UTC+7)
    return self.ts.utc(y, int(m), int(d), int(h) - 7, int(mn))

def export_kml_solar(self):

```

```

selected_item = self.tabel_gerhana.selection()
if not selected_item:
    messagebox.showwarning("Peringatan", "Silakan pilih jadwal Gerhana Matahari di tabel terlebih
dahulu.")
    return

vals = self.tabel_gerhana.item(selected_item[0])['values']
if "Matahari" not in str(vals[0]):
    messagebox.showinfo("Info", "Pilih data Gerhana Matahari, bukan Bulan.")
    return
if "Sebagian" in str(vals[1]):
    messagebox.showinfo("Info", "Gerhana Sebagian (Partial) tidak memiliki Jalur Sentral
Umbra/Antumbra (Totality/Annular).")
    return

waktu_puncak_str = str(vals[3]).strip()
jenis = str(vals[1]).strip()

try:
    # Gunakan Parser Anti-Crash
    t_peak = self._parse_waktu_tabel_gerhana(waktu_puncak_str)
    if t_peak is None: raise ValueError("Gagal membaca format waktu dari tabel.")

    t_start = self.ts.tt_jd(t_peak.tt - 4.0/24.0)
    t_end = self.ts.tt_jd(t_peak.tt + 4.0/24.0)
    t_arr = self.ts.tt_jd(np.linspace(t_start.tt, t_end.tt, 1000))

    earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

    e_pos = earth.at(t_arr)
    M = e_pos.observe(moon).position.km
    S = e_pos.observe(sun).position.km

    V = M - S
    norm_V = np.linalg.norm(V, axis=0)
    v_hat = V / norm_V

    M_dot_v = np.sum(M * v_hat, axis=0)
    M_sq = np.sum(M**2, axis=0)

    R_E = 6371.0
    b = 2.0 * M_dot_v
    c = M_sq - R_E**2
    delta = b**2 - 4.0 * c

    valid_idx = delta >= 0
    if not np.any(valid_idx):

```



```

        messagebox.showwarning("Peringatan", "Jalur sentral meleset dari permukaan Bumi (tidak ada
        daratan yang dilewati sumbu bayangan pusat).")
        return

```

```

        b_val = b[valid_idx]
        delta_val = delta[valid_idx]
        M_val = M[:, valid_idx]
        v_hat_val = v_hat[:, valid_idx]
        t_valid = t_arr.tt[valid_idx]

```

```

        k = (-b_val - np.sqrt(delta_val)) / 2.0
        P_km = M_val + v_hat_val * k

```

```

        P_au = P_km / 149597870.7

```

```

        coords_list = []
        from skyfield.positionlib import Geocentric
        for i in range(len(t_valid)):
            geo = Geocentric(position_au=P_au[:, i], t=self.ts.tt_jd(t_valid[i]))
            subpt = wgs84.subpoint(geo)
            coords_list.append(f"{subpt.longitude.degrees},{subpt.latitude.degrees},0")

```

```

        kml_coords = "\n        ".join(coords_list)

```

```

        kml_content = f'<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>Jalur Gerhana Matahari {waktu_puncak_str}</name>
  <description>Tipe: {jenis}</description>
  <Style id="pathStyle">
    <LineStyle>
      <color>7f0000ff</color> <width>5</width>
    </LineStyle>
  </Style>
  <Placemark>
    <name>Path of Totality / Annularity (Center Line)</name>
    <styleUrl>#pathStyle</styleUrl>
    <LineString>
      <tessellate>1</tessellate>
      <coordinates>
        {kml_coords}
      </coordinates>
    </LineString>
  </Placemark>
</Document>
</kml>'

```

```

        # Rapikan nama file agar bersih dari spasi dan simbol

```

SOURCE CODE KHGT TIMES V8.0

```

safe_name = waktu_puncak_str.replace(" ", "_").replace(":", "")
filepath = filedialog.asksaveasfilename(
    initialfile=f"Jalur_Gerhana_Matahari_{safe_name}.kml",
    defaultextension=".kml",
    filetypes=[("KML Files", "*.kml")]
)

if filepath:
    with open(filepath, 'w', encoding='utf-8') as f:
        f.write(kml_content)
        messagebox.showinfo("Sukses", f"File Jalur Gerhana KML berhasil
diekspor:\n{filepath}\n\nSilakan buka file ini menggunakan Google Earth untuk melihat simulasi
jalurnya.")

except Exception as e:
    import traceback
    traceback.print_exc()
    messagebox.showerror("Error", f"Gagal export KML: {e}")

def tampilkan_detail_lokal_gerhana(self):
    selected_item = self.tabel_gerhana.selection()
    if not selected_item:
        messagebox.showwarning("Peringatan", "Silakan klik/pilih salah satu jadwal gerhana di tabel
terlebih dahulu.")
    return

    item_values = self.tabel_gerhana.item(selected_item[0])['values']
    if not item_values or item_values[0] == "": return

    objek = str(item_values[0]).strip()
    waktu_puncak_str = str(item_values[3]).strip()
    if not waktu_puncak_str: return

    try:
        # Gunakan Parser Anti-Crash
        waktu_puncak_skyfield = self._parse_waktu_tabel_gerhana(waktu_puncak_str)
        if waktu_puncak_skyfield is None: raise ValueError("Gagal membaca format waktu dari tabel.")

        earth = self.eph['earth']
        target_objek = self.eph['moon'] if "Bulan" in objek else self.eph['sun']

        try:
            lat = float(self.entry_vlat.get())
            lon = float(self.entry_vlon.get())
        except:
            lat, lon = -7.0667, 110.4100

        lokasi_observasi = earth + wgs84.latlon(lat, lon)

```

```

astrometric = lokasi_observasi.at(waktu_puncak_skyfield).observe(target_objek)
alt, az, distance = astrometric.apparent().altaz()

extra_info = ""
if "Bulan" in objek:
    iluminasi = almanac.fraction_illuminated(self.eph, 'moon', waktu_puncak_skyfield) * 100.0
    dt_utc_peak = waktu_puncak_skyfield.utc_datetime()
    t0 = self.ts.utc(dt_utc_peak - datetime.timedelta(days=35))
    t1 = waktu_puncak_skyfield

    fase_t, fase_y = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
    waktu_new_moon = [t for t, y in zip(fase_t, fase_y) if y == 0]

    if waktu_new_moon:
        last_new_moon = waktu_new_moon[-1]
        moon_age_actual_days = waktu_puncak_skyfield.tt - last_new_moon.tt

        app_moon = earth.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
        app_sun = earth.at(waktu_puncak_skyfield).observe(self.eph['sun']).apparent()
        _, m_lon, _ = app_moon.ecliptic_latlon()
        _, s_lon, _ = app_sun.ecliptic_latlon()

        phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
        moon_age_phase_days = (phase_angle / 360.0) * 29.530588861

        extra_info = f"\nIluminasi (Fase): {iluminasi:.2f}%\nUmur Aktual (True Time Elapsed):
{moon_age_actual_days:.4f} Hari\nUmur Fase Sudut (Siklik/Elongasi): {moon_age_phase_days:.4f} Hari"
        else:
            extra_info = f"\nIluminasi (Fase): {iluminasi:.2f}%\nUmur Bulan: N/A"
        else:
            app_moon = lokasi_observasi.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
            m_alt, m_az, _ = app_moon.altaz()
            elong = astrometric.separation_from(app_moon).degrees
            extra_info = f"\nAltitude Bulan: {m_alt.degrees:.2f}°\nElongasi Sudut (Toposentrik):
{elong:.2f}°\n"

            if alt.degrees > 0:
                extra_info += ">> MATAHARI DI ATAS UFUK: Gerhana mungkin teramati dari lokasi ini jika
elongasi mendekati 0°."
            else:
                extra_info += ">> MATAHARI DI BAWAH UFUK: Gerhana terjadi saat malam di lokasi ini."

    pesan = (
        f"LOKASI OBSERVASI (GPS)\n"
        f"{self.lokasi_nama.get()}\n"
        f"{'-'*40}\n"
        f"Waktu Puncak Global (WIB): {waktu_puncak_str}\n"

```

```

        f"Waktu Puncak Global (UTC): {waktu_puncak_skyfield.utc_strftime('%Y-%m-%d
%H:%M:%S')}\n"
        f"{' '*40}\n"
        f"PARAMETER VISUAL {objek.upper()} (TOPOSENTRIK)\n"
        f"Altitudo (Tinggi ufuk): {alt.degrees:.2f}°\n"
        f"Azimuth (Arah kompas): {az.degrees:.2f}°\n"
        f"Jarak Bumi-{objek}: {distance.km:,.0f} km\n"
        f"{extra_info}"
    )

    win_detail = ctk.CTkToplevel(self)
    win_detail.title(f"Detail Parameter Lokal - {objek}")
    win_detail.geometry("550x450")
    win_detail.attributes("-topmost", True)

    lbl_title = ctk.CTkLabel(win_detail, text=f"Detail Visual {objek} Saat Puncak", font=("Segoe UI", 16,
"bold"), text_color="#00E5FF")
    lbl_title.pack(pady=(15, 5))

    textbox = ctk.CTkTextbox(win_detail, width=500, height=350, font=("Consolas", 13),
wrap="word", fg_color="#1e1e1e")
    textbox.pack(padx=20, pady=10, fill="both", expand=True)
    textbox.insert("1.0", pesan)
    textbox.configure(state="disabled")

    except Exception as e:
        messagebox.showerror("Error", f"Terjadi kesalahan saat menghitung detail parameter: {e}")

# =====
# MODUL TAMBAHAN 15/36: LIVE ANIMASI (FULL SCROLL & RASI BINTANG)
# =====
def setup_animasi_out_frame(self):
    # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA SCROLL ATAS BAWAH
    self.frame_animasi_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#050510",
corner_radius=10)

    frame_atas = ctk.CTkFrame(self.frame_animasi_out, fg_color="#181818", corner_radius=8)
    frame_atas.pack(fill="x", padx=15, pady=10)

    ctk.CTkLabel(frame_atas, text="🔴 LIVE SIMULATOR", font=("Segoe UI", 16, "bold"),
text_color="#FF5252").pack(side="left", padx=15, pady=10)

    self.is_anim_zoomed = False

    # Penempatan tombol-tombol (Urutan pack(side="right") ditumpuk dari Kanan ke Kiri)
    # 1. Tombol Zoom (Paling Kanan)

```

```

self.btn_anim_zoom = ctk.CTkButton(frame_atas, text="🌐 Zoom Hilal", font=("Segoe UI", 12,
"bold"), fg_color="#673AB7", hover_color="#512DA8", command=self.toggle_anim_zoom)
self.btn_anim_zoom.pack(side="right", padx=(5, 15), pady=10)

# 2. Tombol Tracker (Sebelah kiri Zoom)
self.btn_tracker_link = ctk.CTkButton(
    frame_atas,
    text="🌐 Tracker",
    font=("Segoe UI", 12, "bold"),
    fg_color="#00838F",
    hover_color="#006064",
    command=lambda: webbrowser.open("https://hisabmu.com/khggtimes/tracker/")
)
self.btn_tracker_link.pack(side="right", padx=(5, 5), pady=10)

# 3. Tombol Komparasi Stellarium (Samping kiri Tracker)
self.btn_komparasi_stellarium = ctk.CTkButton(
    frame_atas,
    text="🔭 Stellarium Sync",
    font=("Segoe UI", 12, "bold"),
    fg_color="#E65100",
    hover_color="#BF360C",
    command=self.buka_komparasi_stellarium
)
self.btn_komparasi_stellarium.pack(side="right", padx=(5, 5), pady=10)

# 4. Label Lokasi (Paling Kiri sebelum tombol-tombol)
self.lbl_anim_lokasi = ctk.CTkLabel(frame_atas, textvariable=self.lokasi_nama, font=("Consolas",
12), text_color="#00E5FF")
self.lbl_anim_lokasi.pack(side="right", padx=15)

# 2. SET TINGGI CANVAS (1400px) AGAR RENTANG 90° HINGGA -90° TERLIHAT
self.anim_canvas = tk.Canvas(self.frame_animasi_out, bg='#030514', highlightthickness=0,
height=1400)
self.anim_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))
self.anim_canvas.bind("<Configure>", self.draw_static_background)

# --- VARIABEL STATE UNTUK KOTAK INFO INTERAKTIF ---
self.show_anim_info_box = True
self.anim_cached_date = None
self.anim_cached_events = {}

# --- EVENT BINDING UNTUK INTERAKSI KANVAS ---
self.anim_canvas.tag_bind("btn_close", "<Button-1>", lambda e: self.toggle_anim_info(False))
self.anim_canvas.tag_bind("btn_close", "<Enter>", lambda e:
self.anim_canvas.config(cursor="hand2"))
self.anim_canvas.tag_bind("btn_close", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))

```

```

        self.anim_canvas.tag_bind("sun_obj", "<Button-1>", lambda e: self.toggle_anim_info(True))
        self.anim_canvas.tag_bind("sun_obj", "<Enter>", lambda e:
self.anim_canvas.config(cursor="hand2"))
        self.anim_canvas.tag_bind("sun_obj", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))

        self.anim_canvas.tag_bind("moon_obj", "<Button-1>", lambda e: self.toggle_anim_info(True))
        self.anim_canvas.tag_bind("moon_obj", "<Enter>", lambda e:
self.anim_canvas.config(cursor="hand2"))
        self.anim_canvas.tag_bind("moon_obj", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))

def buka_komparasi_stellarium(self):
    """Memanggil jendela interaktif komparasi data Live Stellarium vs Ephemeris"""
    KomparasiObservatorium(self, self.eph, self.ts)

def draw_static_background(self, event=None):
    """Menggambar Ufuk, Garis Zenith/Nadir, dan Skala Kompas"""
    if getattr(self, 'is_anim_zoomed', False):
        self.anim_canvas.delete("static")
        return

    self.anim_canvas.delete("static")
    width = self.anim_canvas.winfo_width()
    height = self.anim_canvas.winfo_height()
    if height <= 10: return

    horizon_y = height / 2
    pad_x = 45
    usable_width = width - (2 * pad_x)

    # Ubah warna area bawah ufuk menjadi transisi warna bumi gelap
    self.anim_canvas.create_rectangle(0, horizon_y, width, height, fill="#0A1010", outline="",
tags="static")

    # Garis Ufuk 0 derajat
    self.anim_canvas.create_line(0, horizon_y, width, horizon_y, fill="#00E676", width=2, dash=(4, 2),
tags="static")
    self.anim_canvas.create_text(10, horizon_y - 12, text="GARIS UFUK (0°)", fill="#00E676",
font=("Consolas", 10, "bold"), anchor="w", tags="static")

    # Gambar Bintang Latar Acak di SELURUH KANVAS (+90 s/d -90)
    for _ in range(350):
        x = random.randint(0, width)
        y = random.randint(0, height)
        r = random.uniform(0.5, 1.5)
        color = random.choice(["#FFFFFF", "#E0F7FA", "#FFFDE7", "#555555"])
        self.anim_canvas.create_oval(x-r, y-r, x+r, y+r, fill=color, outline="", tags="static")

```

```

# Skala Kompas Standar (Tetap di ufuk)
kompas = {0: "U (0°)", 45: "TL (45°)", 90: "T (90°)", 135: "TG (135°)", 180: "S (180°)", 225: "BD
(225°)", 270: "B (270°)", 315: "BL (315°)", 360: "U (360°)"}
for az, label in kompas.items():
    x = pad_x + (az / 360.0) * usable_width
    self.anim_canvas.create_text(x, horizon_y + 15, text=label, fill="white", font=("Consolas", 9),
tags="static")
    self.anim_canvas.create_line(x, horizon_y, x, horizon_y + 6, fill="white", tags="static")

# Garis Zenith (+90) dan Nadir (-90)
self.anim_canvas.create_line(0, 30, width, 30, fill="#333333", width=1, dash=(2, 4), tags="static")
self.anim_canvas.create_text(10, 40, text="ZENITH (+90°)", fill="#777777", font=("Consolas", 10,
"bold"), anchor="w", tags="static")

self.anim_canvas.create_line(0, height-30, width, height-30, fill="#333333", width=1, dash=(2, 4),
tags="static")
self.anim_canvas.create_text(10, height-40, text="NADIR (-90°)", fill="#777777", font=("Consolas",
10, "bold"), anchor="w", tags="static")

def update_animation(self):
    if not getattr(self, 'anim_running', False): return

    width = self.anim_canvas.winfo_width()
    height = self.anim_canvas.winfo_height()
    if width < 10 or height < 10:
        self.after(500, self.update_animation)
    return

self.anim_canvas.delete("dyn")

def get_compass_dir(deg):
    dirs = ["U", "UTL", "TL", "TTL", "T", "TM", "TG", "SM", "S", "SBD", "BD", "BBD", "B", "BBL", "BL",
"UBL"]
    idx = int(round((deg % 360) / 22.5)) % 16
    return dirs[idx]

try:
    try: lat, lon = float(self.entry_vlat.get()), float(self.entry_vlon.get())
    except: lat, lon = -7.0667, 110.4100
    tz_offset = int(self.get_tz_from_lon(lon))

    if getattr(self, 'anim_is_live', True):
        t_sim = self.ts.now()
        status = " ● LIVE (Waktu Berjalan)"
    else:
        if getattr(self, 'anim_custom_time', None) is None:
            self.anim_custom_time = self.ts.now()

```

```

t_sim = self.anim_custom_time
status = "  KUSTOM (Waktu Dihentikan)"

dt_lokal = t_sim.utc_datetime() + datetime.timedelta(hours=tz_offset)
tz_str = f"UTC{'+' if tz_offset >= 0 else ''}{tz_offset}"
waktu_str = f"{dt_lokal.strftime('%d-%m-%Y %H:%M:%S')} ({tz_str})\nStatus: {status}"

earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
loc = wgs84.latlon(lat, lon)
observer = earth + loc

# Ambil suhu dan tekanan default atau dari input UI jika Anda sudah menyediakannya
try:
    temp_c = float(self.entry_vtemp.get())
    pres_mb = float(self.entry_vpres.get())
except Exception:
    temp_c = 25.0
    pres_mb = 1010.0

astro_sun = observer.at(t_sim).observe(sun).apparent()
alt_sun, az_sun, _ = astro_sun.altaz(temperature_C=temp_c, pressure_mbar=pres_mb)

astro_moon = observer.at(t_sim).observe(moon).apparent()
alt_moon, az_moon, _ = astro_moon.altaz(temperature_C=temp_c, pressure_mbar=pres_mb)

# ---> TAMBAHAN: KALKULASI SEMUA PLANET <---
planets_target = [
    ('mercury barycenter', 'Merkurius', '#B0BEC5'),
    ('venus barycenter', 'Venus', '#FFCC80'),
    ('mars barycenter', 'Mars', '#EF5350'),
    ('jupiter barycenter', 'Jupiter', '#FFB74D'),
    ('saturn barycenter', 'Saturnus', '#FFE082'),
    ('uranus barycenter', 'Uranus', '#81D4FA'),
    ('neptune barycenter', 'Neptunus', '#5C6BC0'),
    ('pluto barycenter', 'Pluto', '#CFD8DC')
]

planet_positions = []
for p_key, p_name, p_color in planets_target:
    try:
        try:
            p_obj = self.eph[p_key]
        except KeyError:
            p_obj = self.eph[p_key.split()][0]

        p_astro = observer.at(t_sim).observe(p_obj).apparent()
        p_alt, p_az, _ = p_astro.altaz()

```



```

        planet_positions.append({
            'name': p_name,
            'color': p_color,
            'alt': p_alt.degrees,
            'az': p_az.degrees
        })
    except Exception:
        pass
# -----

geo_sun = earth.at(t_sim).observe(sun).apparent()
geo_moon = earth.at(t_sim).observe(moon).apparent()

ra_s, dec_s, _ = geo_sun.radec(epoch=t_sim)
ra_m, dec_m, _ = geo_moon.radec(epoch=t_sim)

gast = t_sim.gast
lst_deg = (gast * 15.0) + lon
lat_rad = math.radians(lat)

def calc_geo_alt(ra, dec):
    ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
    dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians
    ha_rad = math.radians(lst_deg - (ra_h * 15.0))
    sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)
    return math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

alt_sun_geo = calc_geo_alt(ra_s, dec_s)
alt_moon_geo = calc_geo_alt(ra_m, dec_m)

elong_geo = geo_sun.separation_from(geo_moon).degrees

_, m_lon, _ = astro_moon.ecliptic_latlon()
_, s_lon, _ = astro_sun.ecliptic_latlon()
phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
moon_idx = int(phase_angle) % 360

waktu_sekarang = ephemeris.Date(t_sim.utc_datetime())
ijtimak = ephemeris.previous_new_moon(waktu_sekarang)
moon_age = waktu_sekarang - ijtimak

# =====
# PROYEKSI VISUAL (NORMAL VS ZOOM)
# =====

alt_s_deg = alt_sun.degrees
az_s_deg = az_sun.degrees
alt_m_deg = alt_moon.degrees

```

```

az_m_deg = az_moon.degrees

alt_top_sun = alt_s_deg + 0.833
alt_top_moon = alt_m_deg + 0.833

pad_x = 45
usable_width = width - (2 * pad_x)

if getattr(self, 'is_anim_zoomed', False):
    # ---- 1. MODE ZOOM HILAL ----
    diff_az = az_m_deg - az_s_deg
    if diff_az > 180: diff_az -= 360
    elif diff_az < -180: diff_az += 360

    center_az = az_s_deg + (diff_az / 2.0)
    center_alt = (alt_s_deg + alt_m_deg) / 2.0

    delta_az = abs(diff_az)
    delta_alt = abs(alt_m_deg - alt_s_deg)

    if delta_az < 2.0: delta_az = 2.0
    if delta_alt < 2.0: delta_alt = 2.0

    pad_zoom_x = 120
    pad_zoom_y = 120
    available_w = max(100, width - (2 * pad_zoom_x))
    available_h = max(100, height - (2 * pad_zoom_y))

    scale_x = available_w / delta_az
    scale_y = available_h / delta_alt
    px_per_deg = min(scale_x, scale_y)

    if px_per_deg > 80.0: px_per_deg = 80.0
    if px_per_deg < 5.0: px_per_deg = 5.0

    px_per_deg_x = px_per_deg
    px_per_deg_y = px_per_deg

    horizon_y = (height / 2.0) + (center_alt * px_per_deg_y)

    r_sun = max(30, min(80, int(0.8 * px_per_deg)))
    r_moon = max(20, min(60, int(0.6 * px_per_deg)))

    def get_zoomed_x(az_target):
        d_az = az_target - center_az
        if d_az > 180: d_az -= 360
        elif d_az < -180: d_az += 360
        return (width / 2.0) + (d_az * px_per_deg_x)

```

```

x_sun = get_zoomed_x(az_s_deg)
x_moon = get_zoomed_x(az_m_deg)

self.anim_canvas.create_line(0, horizon_y, width, horizon_y, fill="#00E676", width=3,
tags="dyn")
self.anim_canvas.create_text(10, horizon_y - 12, text="GARIS UFUK (0°)", fill="#00E676",
font=("Consolas", 12, "bold"), anchor="w", tags="dyn")

start_az = int(center_az - (width/2 / px_per_deg_x)) - 1
end_az = int(center_az + (width/2 / px_per_deg_x)) + 1

for az_mark in range(start_az, end_az):
    mx = get_zoomed_x(az_mark)
    real_az = az_mark % 360
    if 0 <= mx <= width:
        tick_h = 5
        if az_mark % 5 == 0: tick_h = 10
        if az_mark % 10 == 0: tick_h = 15
        self.anim_canvas.create_line(mx, horizon_y, mx, horizon_y + tick_h, fill="white", width=1.5
if tick_h > 5 else 1.0, tags="dyn")
        if az_mark % 10 == 0:
            self.anim_canvas.create_text(mx, horizon_y + 28, text=f"{real_az}°", fill="white",
font=("Consolas", 10), tags="dyn")
            if real_az % 45 == 0:
                arah_dict = {0: "U", 45: "TL", 90: "T", 135: "TG", 180: "S", 225: "BD", 270: "B", 315: "BL"}
                dir_str = arah_dict.get(real_az, "")
                if dir_str:
                    self.anim_canvas.create_text(mx, horizon_y - 20, text=f"▲ {dir_str}", fill="#B0BEC5",
font=("Segoe UI", 14, "bold"), tags="dyn")

        self.anim_canvas.create_line(x_sun, horizon_y, x_sun, horizon_y + 45, fill="#FFD54F", width=2,
tags="dyn")
        self.anim_canvas.create_text(x_sun, horizon_y + 55, text=f"AZIMUT MTHRI\n{az_s_deg:.2f}°",
fill="#FFD54F", font=("Consolas", 11, "bold"), justify="center", anchor="n", tags="dyn")

    y_offset_moon = 45
    if abs(x_sun - x_moon) < 110:
        y_offset_moon = 90
        self.anim_canvas.create_line(x_moon, horizon_y, x_moon, horizon_y + y_offset_moon,
fill="#00E5FF", width=2, dash=(4,2), tags="dyn")
    else:
        self.anim_canvas.create_line(x_moon, horizon_y, x_moon, horizon_y + y_offset_moon,
fill="#00E5FF", width=2, tags="dyn")

    self.anim_canvas.create_text(x_moon, horizon_y + y_offset_moon + 10, text=f"AZIMUT
HILAL\n{az_m_deg:.2f}°", fill="#00E5FF", font=("Consolas", 11, "bold"), justify="center", anchor="n",
tags="dyn")

```

```

else:
    # ---- 2. MODE NORMAL (TAMPILAN PENUH +90° HINGGA -90°) ----
    horizon_y = height / 2.0
    px_per_deg_y = horizon_y / 90.0
    px_per_deg_x = usable_width / 360.0

    r_sun = max(36, int(3.6 * px_per_deg_y))
    r_moon = max(22, int(2.2 * px_per_deg_y))

    x_sun = pad_x + (az_s_deg * px_per_deg_x)
    x_moon = pad_x + (az_m_deg * px_per_deg_x)

# Kalkulasi Posisi Y Awal
y_sun = horizon_y - (alt_s_deg * px_per_deg_y)
y_moon = horizon_y - (alt_m_deg * px_per_deg_y)

offset_visual_y = r_sun - (0.833 * px_per_deg_y)
y_sun += offset_visual_y
y_moon += offset_visual_y

# --- RENDER DATABASE RASI BINTANG (CONSTELLATIONS) ---
from skyfield.api import Star
CONSTELLATIONS = {
    "ORION (Si Pemburu)": [
        {"name": "Betelgeuse", "ra": (5, 55, 10.3), "dec": (7, 24, 25), "mag": 0.50},
        {"name": "Bellatrix", "ra": (5, 25, 7.9), "dec": (6, 20, 59), "mag": 1.64},
        {"name": "Alnilam", "ra": (5, 36, 12.8), "dec": (-1, 12, 6.9), "mag": 1.69},
        {"name": "Rigel", "ra": (5, 14, 32.3), "dec": (-8, 12, 6), "mag": 0.18}
    ],
    "URSA MAJOR (Beruang Besar)": [
        {"name": "Alkaid", "ra": (13, 47, 32.4), "dec": (49, 18, 48), "mag": 1.85},
        {"name": "Mizar", "ra": (13, 23, 55.5), "dec": (54, 55, 31), "mag": 2.23},
        {"name": "Alioth", "ra": (12, 54, 1.7), "dec": (55, 57, 35), "mag": 1.76},
        {"name": "Dubhe", "ra": (11, 3, 43.7), "dec": (61, 45, 4), "mag": 1.81}
    ],
    "CRUX (Salib Selatan)": [
        {"name": "Gacrux", "ra": (12, 31, 9.9), "dec": (-57, 6, 47), "mag": 1.59},
        {"name": "Mimosa", "ra": (12, 47, 43.3), "dec": (-59, 41, 19), "mag": 1.25},
        {"name": "Acrux", "ra": (12, 26, 35.9), "dec": (-63, 5, 57), "mag": 0.77}
    ],
    "SCORPIUS (Kalajengking)": [
        {"name": "Dschubba", "ra": (16, 0, 20.0), "dec": (-22, 37, 18), "mag": 2.29},
        {"name": "Antares", "ra": (16, 29, 24.4), "dec": (-26, 25, 55), "mag": 1.06},
        {"name": "Shaula", "ra": (17, 33, 36.5), "dec": (-37, 6, 14), "mag": 1.62}
    ],
    "CANIS MAJOR (Anjing Besar)": [
        {"name": "Sirius", "ra": (6, 45, 8.9), "dec": (-16, 42, 58), "mag": -1.46},

```

```

    {"name": "Wezen", "ra": (7, 8, 23.5), "dec": (-26, 23, 35), "mag": 1.83},
    {"name": "Adhara", "ra": (6, 58, 37.5), "dec": (-28, 58, 19), "mag": 1.50}
  ],
  "TAURUS (Banteng)": [
    {"name": "Aldebaran", "ra": (4, 35, 55.2), "dec": (16, 30, 33), "mag": 0.87},
    {"name": "Elnath", "ra": (5, 26, 17.5), "dec": (28, 36, 27), "mag": 1.65}
  ],
  "LEO (Singa)": [
    {"name": "Regulus", "ra": (10, 8, 22.3), "dec": (11, 58, 2), "mag": 1.36},
    {"name": "Denebola", "ra": (11, 49, 3.3), "dec": (14, 34, 19), "mag": 2.14}
  ],
  "CENTAURUS": [
    {"name": "Rigil Kentaurus", "ra": (14, 39, 36.5), "dec": (-60, 50, 2), "mag": -0.27},
    {"name": "Hadar", "ra": (14, 3, 49.4), "dec": (-60, 22, 23), "mag": 0.61}
  ],
  "GEMINI (Si Kembar)": [
    {"name": "Castor", "ra": (7, 34, 36.0), "dec": (31, 53, 18), "mag": 1.58},
    {"name": "Pollux", "ra": (7, 45, 18.9), "dec": (28, 1, 34), "mag": 1.16}
  ],
  "CYGNUS (Angsa)": [
    {"name": "Deneb", "ra": (20, 41, 25.9), "dec": (45, 16, 49), "mag": 1.25},
    {"name": "Sadr", "ra": (20, 22, 13.7), "dec": (40, 15, 24), "mag": 2.23},
    {"name": "Albireo", "ra": (19, 30, 43.3), "dec": (27, 57, 34), "mag": 3.05}
  ],
  "LYRA (Sitar)": [
    {"name": "Vega", "ra": (18, 36, 56.3), "dec": (38, 47, 1), "mag": 0.03}
  ],
  "AQUILA (Elang)": [
    {"name": "Altair", "ra": (19, 50, 47.0), "dec": (8, 52, 6), "mag": 0.77}
  ],
  "CARINA (Lunas Kapal)": [
    {"name": "Canopus", "ra": (6, 23, 57.1), "dec": (-52, 41, 45), "mag": -0.74},
    {"name": "Miaaplacidus", "ra": (9, 13, 11.9), "dec": (-69, 43, 1), "mag": 1.67}
  ],
  "VIRGO (Gadis)": [
    {"name": "Spica", "ra": (13, 25, 11.6), "dec": (-11, 9, 41), "mag": 0.98}
  ],
  "AURIGA (Kusir)": [
    {"name": "Capella", "ra": (5, 16, 41.4), "dec": (45, 59, 53), "mag": 0.08}
  ],
  "BOÖTES (Pengembala)": [
    {"name": "Arcturus", "ra": (14, 15, 39.7), "dec": (19, 10, 57), "mag": -0.05}
  ],
  "PISCIS AUSTRINUS": [
    {"name": "Fomalhaut", "ra": (22, 57, 39.1), "dec": (-29, 37, 20), "mag": 1.16}
  ],
  "ERIDANUS (Sungai)": [
    {"name": "Achernar", "ra": (1, 37, 42.8), "dec": (-57, 14, 12), "mag": 0.46}
  ]
}

```

```

    ]
}

for const_name, stars_data in CONSTELLATIONS.items():
    points = []
    for s_data in stars_data:
        bintang = Star(ra_hours=s_data['ra'], dec_degrees=s_data['dec'])
        astrometric_star = observer.at(t_sim).observe(bintang).apparent()
        alt_s, az_s, _ = astrometric_star.altaz()
        s_alt_deg = alt_s.degrees
        s_az_deg = az_s.degrees

        if getattr(self, 'is_anim_zoomed', False):
            x_star = get_zoomed_x(s_az_deg)
            y_star = horizon_y - (s_alt_deg * px_per_deg_y) + offset_visual_y
        else:
            x_star = pad_x + (s_az_deg * px_per_deg_x)
            y_star = horizon_y - (s_alt_deg * px_per_deg_y) + offset_visual_y

        r_star = max(1, int(5 - s_data['mag']))

        if 0 <= x_star <= width and 0 <= y_star <= height:
            points.append((x_star, y_star))
            self.anim_canvas.create_oval(x_star-r_star, y_star-r_star, x_star+r_star, y_star+r_star,
                                         fill="#FFFFFF", outline="#B0BEC5", tags="dyn")
            # Tulis Nama Bintang
            if s_data['mag'] < 2.0 or getattr(self, 'is_anim_zoomed', False):
                self.anim_canvas.create_text(x_star, y_star + r_star + 6, text=s_data['name'],
                                             fill="#9E9E9E", font=("Segoe UI", 9), tags="dyn")

    # Gambar Garis Rasi Bintang & Label jika lebih dari 1 bintang terlihat (khusus mode Normal)
    if len(points) > 1 and not getattr(self, 'is_anim_zoomed', False):
        for i in range(len(points) - 1):
            # Mencegah garis melintang dari ujung kiri ke kanan (Azimuth Wrap Around 360 -> 0)
            if abs(points[i][0] - points[i+1][0]) < (width / 2):
                self.anim_canvas.create_line(points[i][0], points[i][1], points[i+1][0], points[i+1][1],
                                             fill="#4A6572", width=1.5, dash=(2, 2), tags="dyn")

        self.anim_canvas.create_text(points[0][0], points[0][1] - 18, text=const_name,
                                     fill="#FFD54F", font=("Segoe UI", 11, "bold"), tags="dyn")

    if not hasattr(self, '_live_images'):
        self._live_images = {}

    # ---> TAMBAHAN: MENGGAMBAR SEMUA PLANET <---
    for p in planet_positions:
        if getattr(self, 'is_anim_zoomed', False):
            d_az = p['az'] - center_az

```

```

        if d_az > 180: d_az -= 360
        elif d_az < -180: d_az += 360
        p_x = (width / 2.0) + (d_az * px_per_deg_x)
    else:
        p_x = pad_x + (p['az'] * px_per_deg_x)

    p_y = horizon_y - (p['alt'] * px_per_deg_y) + offset_visual_y

    if 0 <= p_x <= width and 0 <= p_y <= height:
        r_p = 5 if getattr(self, 'is_anim_zoomed', False) else 3
        self.anim_canvas.create_oval(p_x-r_p, p_y-r_p, p_x+r_p, p_y+r_p, fill=p['color'],
        outline="white", tags="dyn")
        self.anim_canvas.create_text(p_x, p_y+r_p+8, text=p['name'], fill=p['color'],
        font=("Consolas", 10 if getattr(self, 'is_anim_zoomed', False) else 8, "bold"), tags="dyn")
        # -----

    # --- GAMBAR GARIS ELONGASI (Tengah Matahari & Bulan) ---
    self.anim_canvas.create_line(x_sun, y_sun, x_moon, y_moon, fill="#00E676", dash=(4, 2),
    width=2, tags="dyn")
    mid_x = (x_sun + x_moon) / 2
    mid_y = (y_sun + y_moon) / 2

    if getattr(self, 'is_anim_zoomed', False):
        self.anim_canvas.create_text(mid_x+1, mid_y - 14, text=f"Eln: {elong_geo:.2f}°", fill="black",
        font=("Consolas", 15, "bold"), tags="dyn")
        self.anim_canvas.create_text(mid_x, mid_y - 15, text=f"Eln: {elong_geo:.2f}°", fill="#00E676",
        font=("Consolas", 15, "bold"), tags="dyn")
    else:
        self.anim_canvas.create_text(mid_x, mid_y - 12, text=f"Eln:{elong_geo:.1f}°", fill="#00E676",
        font=("Consolas", 10, "bold"), tags="dyn")

    # --- RENDER MATAHARI ---
    try:
        sun_path = os.path.join(BASE_DIR, "matahari.png")
        sun_img = Image.open(sun_path).convert("RGBA")
        sun_img = sun_img.resize((r_sun*2, r_sun*2), Image.Resampling.LANCZOS)
        self._live_images['sun'] = ImageTk.PhotoImage(sun_img)
        self.anim_canvas.create_image(x_sun, y_sun, image=self._live_images['sun'], tags=("dyn",
        "sun_obj"))
    except Exception:
        self.anim_canvas.create_oval(x_sun-r_sun-4, y_sun-r_sun-4, x_sun+r_sun+4, y_sun+r_sun+4,
        fill="#FF6D00", outline="", stipple="gray25", tags=("dyn", "sun_obj"))
        self.anim_canvas.create_oval(x_sun-r_sun, y_sun-r_sun, x_sun+r_sun, y_sun+r_sun,
        fill="#FFD54F", outline="#FFF59D", width=2, tags=("dyn", "sun_obj"))

    sun_text = f"Matahari\nAlt(Topo): {alt_s_deg:.2f}°\nAlt(Geo) : {alt_sun_geo:.2f}°\nAzimuth :
    {az_s_deg:.1f}°"
    if not getattr(self, 'is_anim_zoomed', False):

```

```

        if x_sun > width - 120:
            self.anim_canvas.create_text(x_sun - r_sun - 15, y_sun, text=sun_text, fill="#FFD54F",
font=("Consolas", 10, "bold"), justify="right", anchor="e", tags="dyn")
        else:
            self.anim_canvas.create_text(x_sun + r_sun + 15, y_sun, text=sun_text, fill="#FFD54F",
font=("Consolas", 10, "bold"), justify="left", anchor="w", tags="dyn")

# --- RENDER BULAN & FASE MASK (DENGAN ROTASI KEMIRINGAN) ---
try:
    moon_filename = f"m{moon_idx:03d}.png"
    moon_path = os.path.join(BASE_DIR, "moon", moon_filename)
    moon_img = Image.open(moon_path).convert("RGBA")
    target_size = int(r_moon * 2)
    moon_img = moon_img.resize((target_size, target_size), Image.Resampling.LANCZOS)

    dx = x_sun - x_moon
    dy = y_sun - y_moon
    sudut_kemiringan = math.degrees(math.atan2(dy, dx))

    if moon_idx >= 180:
        sudut_kemiringan -= 180

    moon_img = moon_img.rotate(-sudut_kemiringan, resample=Image.Resampling.BICUBIC)

    mask = Image.new('L', (target_size, target_size), 0)
    draw = ImageDraw.Draw(mask)
    draw.ellipse((0, 0, target_size - 1, target_size - 1), fill=255)
    moon_img.putalpha(mask)

    self._live_images['moon'] = ImageTk.PhotoImage(moon_img)
    self.anim_canvas.create_image(x_moon, y_moon, image=self._live_images['moon'],
tags=("dyn", "moon_obj"))
except Exception:
    self.anim_canvas.create_oval(x_moon-r_moon-4, y_moon-r_moon-4, x_moon+r_moon+4,
y_moon+r_moon+4, fill="#80DEEA", outline="", stipple="gray12", tags=("dyn", "moon_obj"))
    self.anim_canvas.create_oval(x_moon-r_moon, y_moon-r_moon, x_moon+r_moon,
y_moon+r_moon, fill="#E0E0E0", outline="white", width=2, tags=("dyn", "moon_obj"))

    moon_text = f"Bulan\nAlt(Topo): {alt_m_deg:.2f}°\nAlt(Geo) : {alt_moon_geo:.2f}°\nAzimuth :
{az_m_deg:.1f}°\nFase: {moon_idx}°"
    if getattr(self, 'is_anim_zoomed', False):
        self.anim_canvas.create_text(x_moon + 16, (y_moon + horizon_y) / 2 + 1, text=f"Alt:
{alt_moon_geo:.2f}°", fill="black", font=("Consolas", 15, "bold"), anchor="w", tags="dyn")
        self.anim_canvas.create_text(x_moon + 15, (y_moon + horizon_y) / 2, text=f"Alt:
{alt_moon_geo:.2f}°", fill="#FFD54F", font=("Consolas", 15, "bold"), anchor="w", tags="dyn")
    else:
        if x_moon > width - 120:

```



```

        self.anim_canvas.create_text(x_moon - r_moon - 15, y_moon, text=moon_text,
fill="#00E5FF", font=("Consolas", 10, "bold"), justify="right", anchor="e", tags="dyn")
    else:
        self.anim_canvas.create_text(x_moon + r_moon + 15, y_moon, text=moon_text,
fill="#00E5FF", font=("Consolas", 10, "bold"), justify="left", anchor="w", tags="dyn")

    # --- GARIS PROYEKSI KE UFUK MATAHARI (HANYA NORMAL MODE) ---
    if not getattr(self, 'is_anim_zoomed', False):
        if (y_sun + r_sun) < horizon_y:
            self.anim_canvas.create_line(x_sun, y_sun + r_sun, x_sun, horizon_y, fill="#FFD54F",
dash=(2, 2), tags="dyn")
        elif (y_sun - r_sun) > horizon_y:
            self.anim_canvas.create_line(x_sun, y_sun - r_sun, x_sun, horizon_y, fill="#FFD54F", dash=(2,
2), tags="dyn")

    # --- GARIS PROYEKSI TINGGI HILAL KE UFUK ---
    if (y_moon + r_moon) < horizon_y:
        self.anim_canvas.create_line(x_moon, y_moon + r_moon, x_moon, horizon_y, fill="#00E5FF",
dash=(2, 2), tags="dyn")
    elif (y_moon - r_moon) > horizon_y:
        self.anim_canvas.create_line(x_moon, y_moon - r_moon, x_moon, horizon_y, fill="#00E5FF",
dash=(2, 2), tags="dyn")

    # --- KOTAK INFO STATUS TRACKING HILAL ---
    is_sun_set_visually = alt_top_sun <= 0
    is_moon_set_visually = alt_top_moon <= 0

    txt_fisik_sun = "Terbenam Total" if is_sun_set_visually else "Di Atas Ufuk"
    txt_fisik_moon = "Terbenam Total" if is_moon_set_visually else "Di Atas Ufuk"

    arah_sun = get_compass_dir(az_s_deg)
    arah_moon = get_compass_dir(az_m_deg)
    beda_az = abs(az_m_deg - az_s_deg)

    delta_az_tilt = az_s_deg - az_m_deg
    if delta_az_tilt > 180: delta_az_tilt -= 360
    elif delta_az_tilt < -180: delta_az_tilt += 360

    delta_alt_tilt = alt_s_deg - alt_m_deg

    tilt_rad = math.atan2(delta_az_tilt * math.cos(math.radians(alt_m_deg)), delta_alt_tilt)
    tilt_deg = math.degrees(tilt_rad)

    jam_arah = round(tilt_deg / 30.0)
    if jam_arah <= 0: jam_arah += 12
    if jam_arah > 12: jam_arah %= 12
    if jam_arah == 0: jam_arah = 12

```

```

jam_cusp = (jam_arah + 6) % 12
if jam_cusp == 0: jam_cusp = 12

tilt_info = f"Perut Sabit Jam {jam_arah} | Ujung Cusp Jam {jam_cusp}"

illum = almanac.fraction_illuminated(self.eph, 'moon', t_sim)
illum_val = (illum.item() if hasattr(illum, 'item') else illum) * 100.0
evs = self._get_anim_daily_events(dt_lokal.year, dt_lokal.month, dt_lokal.day, tz_offset, lat, lon)

self.latest_anim_info_text = (
    f"TRACKING HILAL, MATAHARI & PLANET (REAL-TIME DATA)\n{waktu_str}\n"
    f"Umur Bulan    : {moon_age:.2f} hari\n"
    f"Fase/Illuminasi : {illum_val:.1f}%\n"
    f"Elongasi (Geo) : {elong_geo:.2f}° | Beda Azimuth : {beda_az:.2f}°\n"
    f"Kemiringan    : Zenit {tilt_deg:+.1f}° ({tilt_info})\n"
    f"-----\n"
    f"MATAHARI : {txt_fisik_sun:<15} | Azimut: {az_s_deg:06.2f}° ({arah_sun})\n"
    f"    Terbit: {evs['sun_rise']} | Terbenam: {evs['sun_set']}\n"
    f"HILAL    : {txt_fisik_moon:<15} | Azimut: {az_m_deg:06.2f}° ({arah_moon})\n"
    f"    Terbit: {evs['moon_rise']} | Terbenam: {evs['moon_set']}"
)

self.draw_anim_info_box_now()

elong_topo = astro_sun.separation_from(astro_moon).degrees

if not getattr(self, 'anim_is_live', True):
    if alt_moon_geo >= 5.0 and elong_geo >= 8.0:
        box_w_khgt, box_h_khgt = 380, 40
        y1_khgt = height - 55
        img_khgt = Image.new("RGBA", (box_w_khgt, box_h_khgt), (26, 26, 26, 160))
        self._live_images['khgt_bg'] = ImageTk.PhotoImage(img_khgt)
        self.anim_canvas.create_image(15, y1_khgt, image=self._live_images['khgt_bg'],
anchor="nw", tags="dyn")
        self.anim_canvas.create_rectangle(15, y1_khgt, 15 + box_w_khgt, y1_khgt + box_h_khgt,
fill="", outline="#00E676", width=1.5, tags="dyn")
        self.anim_canvas.create_text(25, y1_khgt + 20, text="Memenuhi kriteria KHGT",
fill="#00E676", font=("Consolas", 20, "bold"), anchor="w", tags="dyn")

    if alt_m_deg >= 3.0 and elong_topo >= 6.4:
        box_w_mabims, box_h_mabims = 450, 40
        x1_mabims = width - box_w_mabims - 15
        y1_mabims = height - 55
        img_mab = Image.new("RGBA", (box_w_mabims, box_h_mabims), (26, 26, 26, 160))
        self._live_images['mabims_bg'] = ImageTk.PhotoImage(img_mab)
        self.anim_canvas.create_image(x1_mabims, y1_mabims,
image=self._live_images['mabims_bg'], anchor="nw", tags="dyn")

```

```

        self.anim_canvas.create_rectangle(x1_mabims, y1_mabims, x1_mabims + box_w_mabims,
y1_mabims + box_h_mabims, fill="", outline="#00E5FF", width=1.5, tags="dyn")
        self.anim_canvas.create_text(x1_mabims + 20, y1_mabims + 20, text="Memenuhi kriteria
Neo Mabims", fill="#00E5FF", font=("Consolas", 20, "bold"), anchor="w", tags="dyn")

    except Exception as e:
        self.anim_canvas.create_text(width/2, height/2, text=f"Sedang memuat data ephemeris... ({e})",
fill="red", font=("Consolas", 12), tags="dyn")

    self.after(1000, self.update_animation)

def toggle_anim_info(self, state):
    """Menyembunyikan/menampilkan kotak info secara dinamis"""
    self.show_anim_info_box = state
    self.draw_anim_info_box_now()

def draw_anim_info_box_now(self):
    """Mencetak UI Kotak Info di Kanvas"""
    self.anim_canvas.delete("dyn_info")
    if getattr(self, 'show_anim_info_box', True):
        if hasattr(self, 'latest_anim_info_text'):
            txt_id = self.anim_canvas.create_text(25, 25, text=self.latest_anim_info_text, fill="white",
font=("Consolas", 12, "bold"), anchor="nw", tags=("dyn", "dyn_info"))
            bbox = self.anim_canvas.bbox(txt_id)
            if bbox:
                # Latar Belakang Kotak
                bg = self.anim_canvas.create_rectangle(bbox[0]-10, bbox[1]-10, bbox[2]+30, bbox[3]+10,
fill="#121212", outline="#00E5FF", width=1.5, tags=("dyn", "dyn_info", "info_bg"))
                self.anim_canvas.tag_lower(bg, txt_id)

                # Tombol Close (X)
                btn_x, btn_y = bbox[2] + 5, bbox[1] - 5
                self.anim_canvas.create_rectangle(btn_x, btn_y, btn_x+20, btn_y+20, fill="#FF5252",
outline="white", tags=("dyn", "dyn_info", "btn_close"))
                self.anim_canvas.create_text(btn_x+10, btn_y+10, text="X", fill="white", font=("Consolas",
10, "bold"), tags=("dyn", "dyn_info", "btn_close"))
            else:
                self.anim_canvas.create_text(25, 25, text="💡 [ Panel Info Disembunyikan - Klik Gambar
Matahari/Bulan untuk Membuka Kembali ]", fill="#00E5FF", font=("Consolas", 11, "italic"), anchor="nw",
tags=("dyn", "dyn_info"))

def _get_anim_daily_events(self, year, month, day, tz, lat, lon):
    """Helper cerdas untuk caching jadwal Rise & Set harian agar animasi tidak lag"""
    date_key = f"{year}-{month}-{day}-{lat}-{lon}"
    if getattr(self, 'anim_cached_date', None) == date_key:
        return getattr(self, 'anim_cached_events', {})

```

```

try:
    t0 = self.ts.utc(year, month, day, -int(tz))
    t1 = self.ts.utc(year, month, day, 24 - int(tz))
    loc = wgs84.latlon(lat, lon)

    # Waktu Matahari
    f_s = almanac.sunrise_sunset(self.eph, loc)
    t_s, y_s = almanac.find_discrete(t0, t1, f_s)
    sun_r, sun_s = "--:--", "--:--"
    if t_s is not None:
        for t_val, y_val in zip(np.atleast_1d(t_s.tt), np.atleast_1d(y_s)):
            dt_loc = self.ts.tt_jd(t_val + tz/24.0).utc_datetime()
            if y_val == 1: sun_r = dt_loc.strftime("%H:%M")
            else: sun_s = dt_loc.strftime("%H:%M")

    # Waktu Bulan
    f_m = almanac.risings_and_settings(self.eph, self.eph['moon'], loc)
    t_m, y_m = almanac.find_discrete(t0, t1, f_m)
    moon_r, moon_s = "--:--", "--:--"
    if t_m is not None:
        for t_val, y_val in zip(np.atleast_1d(t_m.tt), np.atleast_1d(y_m)):
            dt_loc = self.ts.tt_jd(t_val + tz/24.0).utc_datetime()
            if y_val == 1: moon_r = dt_loc.strftime("%H:%M")
            else: moon_s = dt_loc.strftime("%H:%M")
except Exception:
    sun_r, sun_s, moon_r, moon_s = "Err", "Err", "Err", "Err"

self.anim_cached_date = date_key
self.anim_cached_events = {'sun_rise': sun_r, 'sun_set': sun_s, 'moon_rise': moon_r, 'moon_set':
moon_s}
return self.anim_cached_events

```

(Fungsi toggle_anim_zoom dan draw_static_background biarkan tetap seperti sebelumnya)

```

def toggle_anim_zoom(self):
    """Fungsi untuk beralih antara Mode Normal dan Mode Zoom Hilal"""
    self.is_anim_zoomed = not getattr(self, 'is_anim_zoomed', False)
    if self.is_anim_zoomed:
        self.btn_anim_zoom.configure(text="🔍 Normal View", fg_color="#D32F2F",
hover_color="#B71C1C")
        self.anim_canvas.delete("static") # Bersihkan bintang/kompas agar fokus
    else:
        self.btn_anim_zoom.configure(text="🔍 Zoom Hilal", fg_color="#673AB7",
hover_color="#512DA8")

```

```

self.draw_static_background() # Gambar ulang bintang dan kompas

# Paksa update frame seketika
self.update_animation()

def anim_cari_sunset(self):
    self.anim_is_live = False
    try:
        # 1. Ambil data input tanggal
        y = int(self.entry_anim_year.get())
        m = int(self.entry_anim_month.get())
        d = int(self.entry_anim_day.get())

        # 2. Ambil Lokasi
        try:
            lat_val = float(self.entry_vlat.get())
            lon_val = float(self.entry_vlon.get())
            elev_val = float(self.entry_velev.get())
        except Exception:
            lat_val, lon_val, elev_val = -7.0667, 110.4100, 230.0

        tz_offset = int(self.get_tz_from_lon(lon_val))

        # 3. MENGGUNAKAN PYEPHEM MURNI (Bukan Skyfield)
        # PyEphem memproses waktu secara skalar, bebas dari error "single Time / array"
        import ephem
        pengamat = ephem.Observer()
        pengamat.lat = str(lat_val)
        pengamat.lon = str(lon_val)
        pengamat.elevation = elev_val
        pengamat.pressure = 1010 # Standar atmosfer
        pengamat.temp = 25      # Standar suhu
        matahari = ephem.Sun()

        # Mulai pencarian dari jam 12:00 siang Waktu Lokal (diubah ke UTC)
        waktu_mulai_cari = datetime.datetime(y, m, d, 12, 0, 0) - datetime.timedelta(hours=tz_offset)
        pengamat.date = ephem.Date(waktu_mulai_cari)

        # Cari waktu sunset (Terbenam)
        try:
            # Menghasilkan datetime UTC murni
            waktu_sunset_utc = pengamat.next_setting(matahari).datetime()

            # Konversi ke waktu lokal untuk ditampilkan di form GUI
            dt_lokal = waktu_sunset_utc + datetime.timedelta(hours=tz_offset)

```

```

# Konversi ke format Skyfield Time HANYA untuk kebutuhan Animasi
self.anim_custom_time = self.ts.from_datetime(waktu_sunset_utc.replace(tzinfo=pytz.utc))

# 4. Update Angka di Form UI
self.entry_anim_year.delete(0, 'end')
self.entry_anim_year.insert(0, str(dt_lokal.year))

self.entry_anim_month.delete(0, 'end')
self.entry_anim_month.insert(0, f"{dt_lokal.month:02d}")

self.entry_anim_day.delete(0, 'end')
self.entry_anim_day.insert(0, f"{dt_lokal.day:02d}")

self.entry_anim_time.delete(0, 'end')
self.entry_anim_time.insert(0, dt_lokal.strftime("%H:%M:%S"))

except (ephem.AlwaysUpError, ephem.NeverUpError):
    messagebox.showwarning("Peringatan", "Matahari tidak terbenam pada hari/lokasi tersebut (Anomali Kutub).")
    self.anim_start_live()

except Exception as e:
    import traceback
    # Memunculkan log lengkap jika gagal, agar mudah dilacak
    messagebox.showerror("Error", f"Gagal mencari waktu sunset:\n{e}\n\nDetail Error:\n{traceback.format_exc()}")
    self.anim_start_live()

# =====
# MODUL TAMBAHAN 13 & 14: 3D SPACE SYSTEM W/ ZOOM
# =====
def setup_3d_out_frame(self):
    self.frame_3d_out = ctk.CTkFrame(self.main_frame, fg_color="#020205")

    header = ctk.CTkFrame(self.frame_3d_out, fg_color="#101018", corner_radius=8)
    header.pack(fill="x", padx=15, pady=10)

    ctk.CTkLabel(header, text="🌌 3D GEOCENTRIC VIEW", font=("Montserrat", 16, "bold"),
text_color="#00E5FF").pack(side="left", padx=20)

    self.btn_peak_eclipse = ctk.CTkButton(header, text="🌑 ECLIPSE 2026", width=140,
fg_color="#D32F2F", hover_color="#B71C1C", command=self.set_eclipse_time_3d)
    self.btn_peak_eclipse.pack(side="right", padx=(10, 15), pady=10)

# --- TOMBOL ZOOM ---

```

```

self.anim_3d_canvas = tk.Canvas(self.frame_3d_out, bg='#010105', highlightthickness=0)
self.anim_3d_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))

# Inisialisasi Kamera & Skala
self.cam_angle_x = 0.8
self.cam_angle_y = 0.3
self.cam_zoom = 1.0

# Binding Interaksi Mouse (Rotasi & Scroll Zoom Cross-Platform)
self.anim_3d_canvas.bind("<B1-Motion>", self.rotate_3d_view)
self.anim_3d_canvas.bind("<MouseWheel>", self.on_mouse_wheel_3d) # Windows/Mac
self.anim_3d_canvas.bind("<Button-4>", self.on_mouse_wheel_3d) # Linux Scroll Up
self.anim_3d_canvas.bind("<Button-5>", self.on_mouse_wheel_3d) # Linux Scroll Down

def zoom_in(self):
    if hasattr(self, 'cam_zoom'):
        self.cam_zoom *= 1.2
        if self.cam_zoom > 12.0: self.cam_zoom = 12.0 # Batas maksimal perbesaran

def zoom_out(self):
    if hasattr(self, 'cam_zoom'):
        self.cam_zoom *= 0.8
        if self.cam_zoom < 0.1: self.cam_zoom = 0.1 # Batas maksimal pengecilan

def on_mouse_wheel_3d(self, event):
    # Logika scroll menangkap pergerakan roda mouse dengan aman
    if str(event.type) == 'MouseWheel':
        if event.delta > 0: self.zoom_in()
        else: self.zoom_out()
    else:
        if getattr(event, 'num', 0) == 4: self.zoom_in()
        elif getattr(event, 'num', 0) == 5: self.zoom_out()

def get_shared_date_components(self):
    """Memastikan tanggal diambil dari Menu 16 jika sedang aktif (Anti Macet)"""
    try:
        mode_aktif = self.combo_mode.get()
        if "Sistem Sun Moon" in mode_aktif:
            return int(self.combo_3d_y.get()), int(self.combo_3d_m.get()), int(self.combo_3d_d.get())
        else:
            return int(self.entry_vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
    except:
        now = datetime.datetime.now()
        return now.year, now.month, now.day

def force_update_3d(self):
    """Mengeksekusi animasi saat tombol 'TERAPKAN TANGGAL' diklik"""
    try:

```

```

y = int(self.combo_3d_y.get())
m = int(self.combo_3d_m.get())
d = int(self.combo_3d_d.get())

# Sinkronisasi ke form input general (Menu 1) agar sinkron ke seluruh modul
try:
    self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, str(y))
    self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, f"{m:02d}")
    self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, f"{d:02d}")
except: pass

self.lbl_status.configure(text=f"Menerapkan Data Ephemeris {y}...", text_color="#FFAB40")

# Switch data satelit/ephemeris secara real-time
self.auto_switch_ephemeris(y)

self.lbl_status.configure(text="Data 3D Berhasil Diterapkan", text_color="#00E676")
except Exception as e:
    messagebox.showerror("Error", f"Gagal menerapkan tanggal:\n{e}")

def set_eclipse_time_3d(self):
    """Tombol pintar untuk Gerhana 2026"""
    self.combo_3d_y.set("2026")
    self.combo_3d_m.set("03")
    self.combo_3d_d.set("03")

    self.force_update_3d()
    messagebox.showinfo("Eclipse Mode", "Simulasi 3D sukses disetel ke posisi Gerhana Matahari 3
Maret 2026.")

def project_3d_raw(self, x, y, z):
    # Murni melakukan rotasi sumbu 3D dan kalkulasi jarak kamera tanpa terpengaruh Zoom
    x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
    z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
    y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
    z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)

    camera_dist = 1000.0
    factor = camera_dist / (camera_dist + z2) if (camera_dist + z2) != 0 else 1

    return x1 * factor, -y2 * factor, z2

def adjust_zoom(self, factor):
    self.cam_zoom *= factor
    # Membatasi skala zoom agar tidak terlalu jauh atau tembus ke dalam
    if self.cam_zoom < 0.1: self.cam_zoom = 0.1
    if self.cam_zoom > 8.0: self.cam_zoom = 8.0

```



```

def on_mouse_wheel_3d(self, event):
    # Logika menangkap pergerakan roda mouse
    if event.num == 4 or getattr(event, 'delta', 0) > 0:
        self.adjust_zoom(1.1)
    elif event.num == 5 or getattr(event, 'delta', 0) < 0:
        self.adjust_zoom(0.9)

def rotate_3d_view(self, event):
    w = self.anim_3d_canvas.winfo_width()
    h = self.anim_3d_canvas.winfo_height()
    if w > 0 and h > 0:
        self.cam_angle_x = (event.x / w) * 2 * math.pi
        self.cam_angle_y = (event.y / h) * math.pi

def project_3d(self, x, y, z, width, height):
    x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
    z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
    y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
    z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)

    factor = 500 / (500 + z2)
    px = x1 * factor + (width / 2)
    py = -y2 * factor + (height / 2)
    return px, py, z2

def update_3d_animation(self):
    if not self.is_viewing_3d: return

    canvas = self.anim_3d_canvas
    w = canvas.winfo_width()
    h = canvas.winfo_height()

    if w <= 1:
        self.after(200, self.update_3d_animation)
        return

    canvas.delete("all")

    # Gambar background bintang (statis)
    random.seed(42)
    for _ in range(100):
        rx, ry = random.randint(0, w), random.randint(0, h)
        canvas.create_oval(rx, ry, rx+1, ry+1, fill="#555555")

    try:
        y, m, d = self.get_shared_date_components()

```

```

now = datetime.datetime.now()
t_sim = self.ts.utc(y, m, d, now.hour, now.minute, now.second)

earth_obj = self.eph['earth']

# Daftar benda langit yang divisualisasikan
celestial_bodies = [
    ('moon', "#ECEFF1", 15, 180, "BULAN"),
    ('mercury', "#B0BEC5", 10, 230, "MERKURIUS"),
    ('venus', "#FFCC80", 18, 290, "VENUS"),
    ('sun', "#FFD600", 50, 350, "MATAHARI"),
    ('mars', "#EF5350", 16, 450, "MARS"),
    ('jupiter', "#FFB74D", 35, 600, "JUPITER"),
    ('saturn', "#FFC107", 30, 750, "SATURNUS"),
    ('uranus', "#81D4FA", 25, 900, "URANUS"),
    ('neptune', "#5C6BC0", 24, 1050, "NEPTUNUS"),
    ('pluto', "#CFD8DC", 8, 1200, "PLUTO")
]

draw_list = []

# 1. Menggambar Bumi di Pusat (Geocentric)
p_earth = self.project_3d(0, 0, 0, w, h)
draw_list.append(('earth', p_earth, "#2196F3", 30 * self.cam_zoom, "BUMI"))

# 2. Menggambar Umbra (Bayangan Gelap Bumi)
try:
    pos_sun = earth_obj.at(t_sim).observe(self.eph['sun']).position.km
    dist_s = np.linalg.norm(pos_sun)
    # Jarak visual umbra diskalakan dengan zoom
    ux, uy, uz = -(pos_sun / dist_s) * (180 * self.cam_zoom)
    p_umbra = self.project_3d(ux, uy, uz, w, h)
    draw_list.append(('umbra', p_umbra, "#1A1A1A", 22 * self.cam_zoom, ""))
except:
    pass

# 3. Kalkulasi dan Skala Zoom untuk Semua Objek
for name, color, size, vis_dist, label in celestial_bodies:
    try:
        target_key = f'{name} barycenter' if name not in ['sun', 'moon'] else name
        try:
            target_obj = self.eph[target_key]
        except KeyError:
            target_obj = self.eph[name]

        pos_km = earth_obj.at(t_sim).observe(target_obj).position.km
        dist_km = np.linalg.norm(pos_km)

```

```

        # Kalikan jarak visual dengan faktor zoom kamera saat ini
        px, py, pz = (pos_km / dist_km) * (vis_dist * self.cam_zoom)

        proj_p = self.project_3d(px, py, pz, w, h)
        # Kalikan juga ukuran objek (size) dengan zoom
        draw_list.append((name, proj_p, color, size * self.cam_zoom, label))

    except Exception as e:
        pass

    # Algoritma Z-Buffer: Urutkan objek dari yang terjauh ke yang terdekat dari pandangan kamera
    draw_list.sort(key=lambda x: x[1][2], reverse=True)

    # 4. Merender objek ke Kanvas
    for tag, p, color, size, label in draw_list:
        if tag == 'umbra':
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill="", outline="#333",
dash=(4,4))
        else:
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill=color, outline="white" if
tag=='earth' else "")
            # Tampilkan Label Nama Planet (Sembunyikan jika di zoom out terlalu jauh agar tidak
numpuk)
            if label and size > 4:
                font_size = max(7, int(9 * min(self.cam_zoom, 1.5)))
                canvas.create_text(p[0], p[1]+size+12, text=label, fill="white", font=("Consolas", font_size,
"bold"))

    # Menggambar Garis Orbit Panduan (Ikut ter-zoom)
    r_moon = 180 * self.cam_zoom
    r_sun = 350 * self.cam_zoom
    r_out = 600 * self.cam_zoom
    canvas.create_oval(w/2-r_moon, h/2-r_moon, w/2+r_moon, h/2+r_moon, outline="#222")
    canvas.create_oval(w/2-r_sun, h/2-r_sun, w/2+r_sun, h/2+r_sun, outline="#222")
    canvas.create_oval(w/2-r_out, h/2-r_out, w/2+r_out, h/2+r_out, outline="#111")

    info = f"3D GEOCENTRIC SOLAR SYSTEM (LIVE)\n" \
        f"Date: {y}-{m:02d}-{d:02d}\n" \
        f"Zoom Level: {self.cam_zoom:.1f}x\n" \
        f"* Klik Kiri & Geser Mouse untuk Rotasi\n" \
        f"* Scroll Mouse / Tombol UI untuk Zoom"

    canvas.create_rectangle(20, 20, 360, 110, fill="#050510", outline="#00E5FF", width=1)
    canvas.create_text(35, 35, text=info, fill="#00E5FF", font=("Consolas", 10), anchor="nw")

    except Exception as e:
        canvas.create_text(w/2, h/2, text=f"System Updating Data... {e}", fill="gray")

```

```

self.after(50, self.update_3d_animation)

def set_eclipse_time_3d(self):
    self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, "2026")
    self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, "03")
    self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, "03")
    messagebox.showinfo("Eclipse Mode", "Waktu diset ke 3 Maret 2026.")

# =====
# FUNGSI KONVERSI KHGT
# =====
def get_new_moons_in_range(self, start_tt_float, end_tt_float):
    t0 = self.ts.tt_jd(start_tt_float)
    t1 = self.ts.tt_jd(end_tt_float)
    t_obj, y_obj = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
    if t_obj is None: return []
    if getattr(t_obj, 'shape', ()) == ():
        tt_list = [float(t_obj.tt)]
        y_list = [int(y_obj)]
    else:
        tt_list = t_obj.tt.tolist()
        y_list = y_obj.tolist()
    new_moons_tt = []
    for i in range(len(y_list)):
        if y_list[i] == 0:
            new_moons_tt.append(tt_list[i])
    return new_moons_tt

def get_hijri_month_from_tt(self, tt_float):
    delta_months = round((tt_float - 2451550.0) / 29.530588853)
    abs_month = 17038 + delta_months
    y = (abs_month - 1) // 12 + 1
    m = (abs_month - 1) % 12 + 1
    return int(y), int(m)

def get_approx_nm_tt(self, y, m):
    abs_month = (y - 1) * 12 + m
    delta_months = abs_month - 17038
    return 2451550.0 + delta_months * 29.530588853

def calculate_khgt_1st_of_month(self, nm_tt_float):
    # Menggunakan Julian Date (Float TT) agar kebal dari error tahun minus Python
    t_nm = self.ts.tt_jd(nm_tt_float)
    y, m, d, h, mi, s = t_nm.utc

    # Cari titik waktu 00:00 UTC pada hari konjungsi
    t_midnight = self.ts.utc(y, m, d)

```

```

# Aturan KHGT: Jika ijtimaq sebelum jam 15:00 UTC (Fajar Selandia Baru),
# bulan baru masuk besok (+1 hari). Jika tidak, lusa (+2 hari).
if h < 15:
    return t_midnight.tt + 1.0
else:
    return t_midnight.tt + 2.0

def calculate_conversion(self):
    try:
        import math
        mode_conv = self.radio_conv_var.get()
        d = int(self.combo_conv_day.get())

        # --- PERBAIKAN FINAL: Kembalikan teks ke bahasa asli & Anti-Crash ---
        raw_m_str = self.combo_conv_month.get()
        m_str = self.reverse_tr(raw_m_str)

        y = int(self.entry_conv_year.get())
        adj = int(self.combo_conv_adj.get())

        # Fungsi bantuan deteksi kabisat Masehi
        def is_leap(year):
            return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

        # -----
        # NORMALISASI BULAN MULTI-BAHASA UNTUK KONVERSI KALENDER
        # -----
        # Database internal aplikasi memakai nama bulan Indonesia:
        # BULAN_MASEHI dan BULAN_HIJRIAH. Saat UI diganti ke English
        # atau Arabic, nilai ComboBox menjadi terjemahan, sehingga
        # pemanggilan list.index(...) langsung dapat memicu error:
        # 'Ramadan' is not in list / 'رمضان' is not in list.
        # Helper berikut mengembalikan indeks bulan 1..12 secara aman
        # tanpa mengubah database, algoritma, atau modul lain.
        def _clean_month_key(value):
            value = str(value or "").strip()
            value = value.replace(":", "") # hapus tatwil Arab
            value = re.sub(r"\s+", " ", value)
            return value.casefold()

        def get_gregorian_month_index(value):
            key = _clean_month_key(value)
            masehi_map = {}

            # Indonesia/internal
            for idx, name in enumerate(BULAN_MASEHI, 1):
                masehi_map[_clean_month_key(name)] = idx

```

```

# English
en_names = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]
en_abbr = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
for idx, name in enumerate(en_names, 1):
    masehi_map[_clean_month_key(name)] = idx
for idx, name in enumerate(en_abbr, 1):
    masehi_map[_clean_month_key(name)] = idx

# Arabic Gregorian umum
ar_names = [
    "يناير", "فبراير", "مارس", "أبريل", "مايو", "يونيو",
    "يوليو", "أغسطس", "أغسطس", "سبتمبر", "أكتوبر", "أكتوبر",
    "نوفمبر", "ديسمبر"
]
ar_idx = [1, 2, 3, 4, 4, 5, 6, 7, 8, 8, 9, 10, 10, 11, 12]
for name, idx in zip(ar_names, ar_idx):
    masehi_map[_clean_month_key(name)] = idx

# Coba dari file translator JSON bila tersedia
for idx, id_name in enumerate(BULAN_MASEHI, 1):
    for lang_val in getattr(self, "translations", {}).get(id_name, {}).values():
        masehi_map[_clean_month_key(lang_val)] = idx

if key in masehi_map:
    return masehi_map[key]

# Fallback untuk format singkatan English seperti Sept.
key3 = key[:3]
for k, idx in masehi_map.items():
    if len(k) >= 3 and k[:3] == key3:
        return idx

raise ValueError(f>Nama bulan Masehi tidak dikenali: {value}")

def get_hijri_month_index(value):
    key = _clean_month_key(value)
    hijri_map = {}

# Indonesia/internal
for idx, name in enumerate(BULAN_HIJRIAH, 1):
    hijri_map[_clean_month_key(name)] = idx

# Variasi Indonesia yang sering dipakai
id_aliases = {

```

```

1: ["Muharram", "Muharam"],
3: ["Rabiul Awal", "Rabiulawal", "Rabi'ul Awal"],
4: ["Rabiul Akhir", "Rabiulakhir", "Rabi'ul Akhir"],
5: ["Jumadil Awal", "Jumadilawal", "Jumada Awal"],
6: ["Jumadil Akhir", "Jumadilakhir", "Jumada Akhir"],
8: ["Syaban", "Sya'ban", "Syakban"],
9: ["Ramadan", "Ramadhan"],
11: ["Zulkaidah", "Dzulqaidah", "Dzulkaidah", "Zulqaidah"],
12: ["Zulhijah", "Dzulhijjah", "Zulhijjah", "Dzulhijjah"]
}

# English Hijri names
en_aliases = {
    1: ["Muharram", "Muharam"],
    2: ["Safar"],
    3: ["Rabi al-Awwal", "Rabi' al-Awwal", "Rabi I", "Rabiul Awwal"],
    4: ["Rabi al-Thani", "Rabi' al-Thani", "Rabi al-Akhir", "Rabi II"],
    5: ["Jumada al-Awwal", "Jumada I", "Jumada al-Ula"],
    6: ["Jumada al-Thani", "Jumada II", "Jumada al-Akhirah"],
    7: ["Rajab"],
    8: ["Sha'ban", "Shaban", "Shaaban"],
    9: ["Ramadan", "Ramadhan"],
    10: ["Shawwal", "Syawal"],
    11: ["Dhu al-Qadah", "Dhul Qadah", "Dhu al-Qi'dah", "Dhul Qi'dah", "Zulkaidah"],
    12: ["Dhu al-Hijjah", "Dhul Hijjah", "Zulhijah", "Zulhijjah"]
}

# Arabic Hijri names
ar_aliases = {
    1: ["محرم", "محرم"],
    2: ["صفر"],
    3: ["ربيع الأول", "ربيع الاول", "ربيع أول", "ربيع اول"],
    4: ["ربيع الآخر", "ربيع الاخر", "ربيع الثاني"],
    5: ["جمادى الأولى", "جمادى الاولى", "جمادى أولى", "جمادى اولى"],
    6: ["جمادى الآخرة", "جمادى الاخرة", "جمادى الثانية"],
    7: ["رجب"],
    8: ["شعبان"],
    9: ["رمضان"],
    10: ["شوال", "شَوَال"],
    11: ["ذو القعدة", "ذو القعدة", "ذو القعدة", "ذو القعدة"],
    12: ["ذو الحجة", "ذو الحجة", "ذو الحجة", "ذو الحجة"]
}

for alias_dict in (id_aliases, en_aliases, ar_aliases):
    for idx, names in alias_dict.items():
        for name in names:
            hijri_map[_clean_month_key(name)] = idx

```

```

# Coba dari file translator JSON bila tersedia
for idx, id_name in enumerate(BULAN_HIJRIAH, 1):
    for lang_val in getattr(self, "translations", {}).get(id_name, {}).values():
        hijri_map[_clean_month_key(lang_val)] = idx

if key in hijri_map:
    return hijri_map[key]

raise ValueError(f"Nama bulan Hijriah tidak dikenali: {value}")

# Algoritma Julian Day untuk Kalender Urfi
def get_jd(year, month, day):
    if month <= 2:
        year -= 1
        month += 12
    A = math.floor(year / 100)
    B = 2 - A + math.floor(A / 4)
    return math.floor(365.25 * (year + 4716)) + math.floor(30.6001 * (month + 1)) + day + B -
1524.5

def jd_to_urfi(jd):
    jd = math.floor(jd) + 0.5
    days = math.floor(jd - 1948439.5)
    y_h = math.floor((30 * days + 10646) / 10631)
    d_h = days - math.floor((11 * y_h + 3) / 30) - 354 * y_h
    m_h = math.floor((11 * d_h + 330) / 325)
    d_final = d_h - math.floor((325 * m_h - 320) / 11)
    return int(y_h), int(m_h), int(d_final)

def urfi_to_jd(y_h, m_h, d_h):
    return d_h + math.floor((325 * m_h - 320) / 11) + math.floor((11 * y_h + 3) / 30) + 354 * y_h +
1948439.5

def jd_to_gregorian(jd):
    Z = math.floor(jd + 0.5)
    if Z < 2299161:
        A = Z
    else:
        alpha = math.floor((Z - 1867216.25) / 36524.25)
        A = Z + 1 + alpha - math.floor(alpha / 4)
    B = A + 1524
    C = math.floor((B - 122.1) / 365.25)
    D = math.floor(365.25 * C)
    E = math.floor((B - D) / 30.6001)
    day = B - D - math.floor(30.6001 * E)
    month = E - 1 if E < 14 else E - 13
    year = C - 4716 if month > 2 else C - 4715
    return int(year), int(month), int(day)

```



```

# Array Master
hari_masehi = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
pasaran_jawa = ["Legi", "Pahing", "Pon", "Wage", "Kliwon"]

catatan_tambahan = ""

# =====
# LOGIKA 1: MASEHI -> HIJRIAH (KHGT)
# =====
if mode_conv == "m2h":
    m = get_gregorian_month_index(m_str)
    m_id = BULAN_MASEHI[m - 1]

    mode_text = "Masehi → Hijriah (KHGT)"
    greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
    input_text = f"{d} {m_id} {greg_year_str_input}"

    self.auto_switch_ephemeris(y)
    days_in_month = [31, 29 if is_leap(y) else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    d = min(d, days_in_month[m-1])

    target_t_obj = self.ts.utc(y, m, d)
    target_tt = target_t_obj.tt

    new_moons_tt_list = self.get_new_moons_in_range(target_tt - 35.0, target_tt + 2.0)
    current_nm_tt = None
    for nm_tt in reversed(new_moons_tt_list):
        start_tt = self.calculate_khgt_1st_of_month(nm_tt)
        if target_tt >= start_tt - 0.1:
            current_nm_tt = nm_tt
            break

    if current_nm_tt is None:
        raise ValueError(f"Tanggal target di luar jangkauan Ephemeris ({self.ephemeris_name}).")

    h_y, h_m = self.get_hijri_month_from_tt(current_nm_tt)
    start_tt = self.calculate_khgt_1st_of_month(current_nm_tt)

    h_d = int(round(target_tt - start_tt)) + 1
    h_d += adj

    while h_d <= 0:
        h_m -= 1
        if h_m <= 0: h_m, h_y = 12, h_y - 1
        h_d += 30
    while h_d > 30:

```

```

        h_d -= 30
        h_m += 1
        if h_m > 12: h_m, h_y = 1, h_y + 1

    try:
        if y > 0:
            dt_obj = datetime.date(y, m, d)
            hari = f"{hari_masehi[dt_obj.weekday()]} {pasaran_jawa[dt_obj.toordinal() % 5]}"
        else:
            hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
    except:
        hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]

    hijri_year_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
    hasil = f"{hari}, {h_d} {BULAN_HIJRIAH[h_m - 1]} {hijri_year_str} H"
    catatan_tambahan = "- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).\n- Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.\n- Kalkulasi menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak)."
```

=====

LOGIKA 2: HIJRIAH (KHGT) -> MASEHI

=====

```

elif mode_conv == "h2m":
    m = get_hijri_month_index(m_str)
    m_id = BULAN_HIJRIAH[m - 1]
    h_y = y
    mode_text = "Hijriah (KHGT) → Masehi"
    hijri_year_str_input = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
    input_text = f"{d} {m_id} {hijri_year_str_input} H"

    approx_greg_year = int(h_y * 0.970224 + 622.54)
    self.auto_switch_ephemeris(approx_greg_year)

    approx_tt = self.get_approx_nm_tt(h_y, m)
    new_moons_tt_list = self.get_new_moons_in_range(approx_tt - 5.0, approx_tt + 5.0)
    if not new_moons_tt_list:
        raise ValueError(f"Tanggal tersebut di luar jangkauan Ephemeris ({self.ephemeris_name}).")

    exact_nm_tt = new_moons_tt_list[0]
    start_tt = self.calculate_khgt_1st_of_month(exact_nm_tt)

    target_tt = start_tt + (d - 1) - adj
    target_t_obj = self.ts.tt_jd(target_tt)
    ty, tm, td, _, _, _ = target_t_obj.utc

    try:
        ty_int, tm_int, td_int = int(ty), int(tm), int(td)

```

```

    if ty_int > 0:
        dt_obj = datetime.date(ty_int, tm_int, td_int)
        hari = f"{hari_masehi[dt_obj.weekday()]}" {pasaran_jawa[dt_obj.toordinal() % 5]}"
    else:
        hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
except:
    hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]

greg_year_str = f"{int(ty)}" if ty > 0 else f"{abs(int(ty)-1)} SM"
hasil = f"{hari}, {int(td)} {BULAN_MASEHI[int(tm) - 1]} {greg_year_str}"
catatan_tambahan = "- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).\n-
Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.\n- Kalkulasi
menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak)."

# =====
# LOGIKA 3: MASEHI -> HIJRIAH (URFI / KUWEIT)
# =====
elif mode_conv == "m2u":
    m = get_gregorian_month_index(m_str)
    m_id = BULAN_MASEHI[m - 1]

    mode_text = "Masehi → Hijriah (Urfi)"
    greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
    input_text = f"{d} {m_id} {greg_year_str_input}"

    jd = get_jd(y, m, d)
    jd += adj # Koreksi hari
    h_y, h_m, h_d = jd_to_urfi(jd)

    try:
        if y > 0:
            dt_obj = datetime.date(y, m, d)
            hari = f"{hari_masehi[dt_obj.weekday()]}" {pasaran_jawa[dt_obj.toordinal() % 5]}"
        else:
            hari = hari_masehi[int((jd + 1.5) % 7)]
    except:
        hari = hari_masehi[int((jd + 1.5) % 7)]

    hijri_year_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
    hasil = f"{hari}, {h_d} {BULAN_HIJRIAH[h_m - 1]} {hijri_year_str} H"
    catatan_tambahan = "- Konversi ini menggunakan algoritma Tabular (Urfi/Kuweit Standard).\n-
Bersifat murni matematis dari siklus 30 tahun (11 tahun kabisat).\n- Tidak berdasarkan visibilitas/fase
bulan aktual (Bukan Hisab Hakiki)."

# =====
# LOGIKA 4: HIJRIAH (URFI / KUWEIT) -> MASEHI
# =====

```

```

elif mode_conv == "u2m":
    m_h = get_hijri_month_index(m_str)
    m_id = BULAN_HIJRIAH[m_h - 1]
    y_h = y
    mode_text = "Hijriah (Urfi) → Masehi"
    hijri_year_str_input = str(y_h) if y_h > 0 else f"{abs(y_h-1)} SH"
    input_text = f"{d} {m_id} {hijri_year_str_input} H"

    d_adj = d - adj # Inversi koreksi hari
    jd = urfi_to_jd(y_h, m_h, d_adj)
    y_m, m_m, d_m = jd_to_gregorian(jd)

    try:
        if y_m > 0:
            dt_obj = datetime.date(y_m, m_m, d_m)
            hari = f"{hari_masehi[dt_obj.weekday()] {pasaran_jawa[dt_obj.toordinal() % 5]}"
        else:
            hari = hari_masehi[int((jd + 1.5) % 7)]
    except:
        hari = hari_masehi[int((jd + 1.5) % 7)]

    greg_year_str = f"{y_m}" if y_m > 0 else f"{abs(y_m-1)} SM"
    hasil = f"{hari}, {d_m} {BULAN_MASEHI[m_m - 1]} {greg_year_str}"
    catatan_tambahan = "- Konversi ini menggunakan algoritma Tabular (Urfi/Kuweit Standard).\n- Bersifat murni matematis dari siklus 30 tahun (11 tahun kabisat).\n- Tidak berdasarkan visibilitas/fase bulan aktual (Bukan Hisab Hakiki)."

    # --- RENDER LAPORAN ---
    report = f"""\{self.get_header(85)}
    {[ Konversi Kalender Terpadu ]".center(85)}

    * Settings:-
    - Mode Konversi : {mode_text}
    - Tanggal Input : {input_text}
    - Koreksi Hari : {adj} Hari (Penyesuaian Manual)

    =====

    HASIL KONVERSI:

    -----
    >> {hasil} <<
    -----

    * Catatan:
    {catatan_tambahan}

    =====
    """""

    self.after(0, self.display_result, report)

```

```

except Exception as e:
    import traceback
    trace_err = traceback.format_exc()
    self.after(0, self.display_error, f"{str(e)}\n\n(Details):\n{trace_err}")

def generate_visibility_report(self):
    """Fungsi inti kalkulasi hilal yang me-return string report murni (Bisa dipanggil oleh WinAI)"""
    import math
    import pytz
    import datetime
    import numpy as np
    import ephem
    from skyfield import almanac
    from skyfield.api import wgs84

    year = int(self.entry_vyear.get())
    month = int(self.entry_vmonth.get())
    day = int(self.entry_vday.get())

    self.auto_switch_ephemeris(year)

    lat = float(self.entry_vlat.get())
    lon = float(self.entry_vlon.get())
    elev = float(self.entry_velev.get())
    tz = float(self.entry_vtz.get())

    temp = float(self.entry_vtemp.get())
    pres = float(self.entry_vpres.get())
    hum = float(self.entry_vhum.get())

    earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
    lokasi_obs = wgs84.latlon(lat, lon, elevation_m=elev)

    # 1. CARI SUNSET LOKAL UNTUK TITIK OBSERVASI
    t0 = self.ts.utc(year, month, day, 0 - int(tz))
    t1 = self.ts.utc(year, month, day, 23 - int(tz))

    t_sunset = None
    t_evs, y_evs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, lokasi_obs))
    for t_ev, is_sunrise in get_safe_events(t_evs, y_evs):
        if not is_sunrise:
            t_sunset = t_ev
            break

    if t_sunset is None:
        raise ValueError("Sunset (Matahari Terbenam) tidak ditemukan pada tanggal/koordinat tersebut.")

```

```

waktu_mode = getattr(self, 'radio_vis_time_mode', None)
mode_val = waktu_mode.get() if waktu_mode else "maghrib"

if mode_val == "live":
    t_eval = self.ts.now()
    waktu_kalkulasi_str = "LIVE (Real-Time)"
elif mode_val == "kustom":
    time_str = self.entry_vtime.get().strip()
    if not time_str: time_str = "12:00:00"
    parts = time_str.split(":")
    jam = int(parts[0])
    menit = int(parts[1]) if len(parts) > 1 else 0
    detik = int(parts[2]) if len(parts) > 2 else 0

    dt_lokal = datetime.datetime(year, month, day, jam, menit, detik)
    dt_utc = dt_lokal - datetime.timedelta(hours=tz)
    t_eval = self.ts.utc(dt_utc.year, dt_utc.month, dt_utc.day, dt_utc.hour, dt_utc.minute,
dt_utc.second)
    waktu_kalkulasi_str = f"{time_str} LT (Kustom)"
else:
    t_eval = t_sunset
    waktu_kalkulasi_str = f"{get_hms_str(t_sunset, tz)} LT (Sunset)"

ts_tt = t_sunset.tt.item() if hasattr(t_sunset.tt, 'item') else t_sunset.tt
t_eval_tt = t_eval.tt.item() if hasattr(t_eval.tt, 'item') else t_eval.tt

t_bound_moon = self.ts.tt_jd(ts_tt + 1.5)
t_moonset = None
t_mevs, y_mevs = almanac.find_discrete(t0, t_bound_moon, almanac.risings_and_settings(self.eph,
moon, lokasi_obs))

for t_ev, is_moonrise in get_safe_events(t_mevs, y_mevs):
    tm_tt = t_ev.tt.item() if hasattr(t_ev.tt, 'item') else t_ev.tt
    if not is_moonrise and tm_tt > (ts_tt - 0.5):
        t_moonset = t_ev
        break

# 2. CARI IJTIMAK GLOBAL (+/- 20 Hari)
t_start_conj = self.ts.tt_jd(t_eval_tt - 20.0)
t_end_conj = self.ts.tt_jd(t_eval_tt + 20.0)

t_phases, y_phases = almanac.find_discrete(t_start_conj, t_end_conj,
almanac.moon_phases(self.eph))
t_ijtimak = None
jarak_terdekat = float('inf')

for t_ev, phase in get_safe_events(t_phases, y_phases):

```

```

if phase == 0:
    selisih_hari = abs(t_ev.tt - t_eval_tt)
    if selisih_hari < jarak_terdekat:
        jarak_terdekat = selisih_hari
        t_ijtima = t_ev

# 3. KALKULASI PARAMETER LOKAL SAAT WAKTU EVALUASI
geo_earth = earth.at(t_eval)
app_moon_geo = geo_earth.observe(moon).apparent()
app_sun_geo = geo_earth.observe(sun).apparent()

ra_moon, dec_moon, dist_moon = app_moon_geo.radec(epoch=t_eval)
ra_sun, dec_sun, dist_sun = app_sun_geo.radec(epoch=t_eval)

gast = t_eval.gast
lst_deg = (gast * 15.0) + lon

def get_geo_altaz(ra, dec):
    ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
    dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians
    ha_deg = lst_deg - (ra_h * 15.0)
    lat_rad, dec_rad, ha_rad = math.radians(lat), dec_r, math.radians(ha_deg)
    sin_alt = math.sin(dec_rad) * math.sin(lat_rad) + math.cos(dec_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    sin_alt = max(-1.0, min(1.0, sin_alt))
    alt = math.degrees(math.asin(sin_alt))
    y = -math.sin(ha_rad)
    x = math.tan(dec_rad) * math.cos(lat_rad) - math.sin(lat_rad) * math.cos(ha_rad)
    az = (math.degrees(math.atan2(y, x)) + 360) % 360
    return alt, az

alt_moon_geo, az_moon_geo = get_geo_altaz(ra_moon, dec_moon)
alt_sun_geo, az_sun_geo = get_geo_altaz(ra_sun, dec_sun)

topo_obs = (earth + lokasi_obs).at(t_eval)
app_moon_topo = topo_obs.observe(moon).apparent()
app_sun_topo = topo_obs.observe(sun).apparent()

alt_moon_topo_obj, az_moon_topo_obj, _ = app_moon_topo.altaz(temperature_C=temp,
pressure_mbar=pres)
alt_sun_topo_obj, az_sun_topo_obj, _ = app_sun_topo.altaz(temperature_C=temp,
pressure_mbar=pres)

alt_moon_topo = alt_moon_topo_obj.degrees
alt_sun_topo = alt_sun_topo_obj.degrees

ref_moon = calculate_saemundsson_refraction(alt_moon_topo, temp, pres)
ref_sun = calculate_saemundsson_refraction(alt_sun_topo, temp, pres)

```

```

if t_ijtima is not None:
    ti_tt = t_ijtima.tt.item() if hasattr(t_ijtima.tt, 'item') else t_ijtima.tt
    moon_age_hours = (t_eval_tt - ti_tt) * 24.0
else:
    moon_age_hours = 0.0

if t_moonset is not None:
    tm_tt = t_moonset.tt.item() if hasattr(t_moonset.tt, 'item') else t_moonset.tt
    lag_time_hours = (tm_tt - ts_tt) * 24.0
else:
    lag_time_hours = 0.0

sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
elongation = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg
rel_alt_topo = alt_moon_topo - alt_sun_topo

def moon_sun_elongation(t):
    e = earth.at(t)
    _, slon, _ = e.observe(sun).apparent().ecliptic_latlon(epoch=t)
    _, mlon, _ = e.observe(moon).apparent().ecliptic_latlon(epoch=t)
    s_deg = slon.degrees.item() if hasattr(slon.degrees, 'item') else slon.degrees
    m_deg = mlon.degrees.item() if hasattr(mlon.degrees, 'item') else mlon.degrees
    return (m_deg - s_deg) % 360.0

phase_angle = moon_sun_elongation(t_eval)
illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_eval)
illumination = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0

dist_km = dist_moon.km.item() if hasattr(dist_moon.km, 'item') else dist_moon.km
sd_moon = math.degrees(math.asin(1737.4 / dist_km))

W_arcmin = (sd_moon * 60.0) * (1 - math.cos(math.radians(elongation)))
q_yallop = rel_alt_topo - (11.8371 - 6.3226 * W_arcmin + 0.7319 * (W_arcmin**2) - 0.1018 *
(W_arcmin**3))

if q_yallop > 0.216: yallop_stat = "A (Mudah Terlihat / Mata Telanjang)"
elif q_yallop > -0.014: yallop_stat = "B (Terlihat dgn Cuaca Sempurna)"
elif q_yallop > -0.160: yallop_stat = "C (Butuh Alat Optik utk Menemukan)"
elif q_yallop > -0.232: yallop_stat = "D (Hanya Terlihat via Alat Optik)"
else: yallop_stat = "F (TIDAK TERLIHAT / Bawah Limit)"

if t_ijtima is not None:
    dt_utc_ijtima = t_ijtima.utc_datetime()
    dt_lokal_ijtima = dt_utc_ijtima + datetime.timedelta(hours=tz)
    y_i, m_i, d_i = dt_lokal_ijtima.year, dt_lokal_ijtima.month, dt_lokal_ijtima.day
    h_i, mn_i = dt_lokal_ijtima.hour, dt_lokal_ijtima.minute
    y_str = f"{y_i}" if y_i > 0 else f"{abs(y_i-1)} SM"

```



```

    str_ijtima = f"{d_i:02d}/{m_i:02d}/{y_str}, {int(h_i):02d}:{int(mn_i):02d} LT"
    str_ijtima_utc = dt_utc_ijtima.strftime('%d/%m/%Y, %H:%M UTC')
else:
    str_ijtima, str_ijtima_utc = "N/A", "N/A"

# Variabel Logika Kriteria Lokal
is_khgt_local = alt_moon_geo >= 5.0 and elongation >= 8.0

is_mabims = alt_moon_topo >= 3.0 and elongation >= 6.4
str_mabims = "TERPENUHI" if is_mabims else "Gagal"

is_ilyas = alt_moon_topo >= 4.0 and elongation >= 10.5
str_ilyas = "TERPENUHI" if is_ilyas else "Gagal"

is_danjon = elongation >= 7.0
str_danjon = "LOLOS (Sabit Terbentuk)" if is_danjon else "GAGAL (Sabit Tidak Terbentuk)"

# =====
# 4. PEMINDAIAN GLOBAL KHGT (PKG 1 & PKG 2)
# =====
status_khgt_global = "GAGAL (Belum Terpenuhi di belahan bumi manapun)"
pkg_reason = ""
titik_pertama_global = None

if t_ijtima is not None:
    obs_date_utc = datetime.date(dt_utc_ijtima.year, dt_utc_ijtima.month, dt_utc_ijtima.day)
    batas_00_utc_ephem = ephem.Date(datetime.datetime(obs_date_utc.year,
obs_date_utc.month, obs_date_utc.day) + datetime.timedelta(days=1))

    # Cari Fajar Gisborne untuk PKG 2
    gisborne = ephem.Observer()
    gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
    gisborne.elevation, gisborne.pressure, gisborne.horizon = 0, 0, '-18'
    matahari_ephem = ephem.Sun()
    gisborne.date = ephem.Date(batas_00_utc_ephem - 0.5)
    try:
        fajar_gisborne = gisborne.next_rising(matahari_ephem, use_center=True)
    except:
        fajar_gisborne = ephem.Date(batas_00_utc_ephem + 0.6)

    is_ijtima_before_fajar = ephem.Date(dt_utc_ijtima) < fajar_gisborne

    pkg1_met = False
    pkg2_met = False

# ---> LANGKAH 1: JADIKAN TITIK LOKAL SEBAGAI PRIORITAS PERTAMA (Akselerasi) <---
nama_kota = getattr(self, 'opt_city', None)
nama_kota = nama_kota.get() if nama_kota else "Lokasi Anda"

```

```

nama_prov = getattr(self, 'opt_prov', None)
nama_prov = nama_prov.get() if nama_prov else ""

if is_khgt_local:
    if ephemeris.Date(t_sunset.utc_datetime()) < batas_00_utc_ephem:
        pkg1_met = True
        pkg_reason = f"PKG 1 (Titik Lokal Anda: {nama_kota})"
        titik_pertama_global = f"{nama_kota} | Sunset UTC: {t_sunset.utc_strftime('%H:%M:%S')}"
    else:
        # Jika sunset lokal lewat 00:00 UTC, apakah ini di Benua Amerika?
        negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan", "Brasil",
"Argentina", "Kolombia", "Peru", "Chili"]
        if nama_prov in negara_amerika or "Amerika" in nama_prov:
            if is_ijtima_before_fajr:
                pkg2_met = True
                pkg_reason = f"PKG 2 (Titik Lokal Anda: {nama_kota} & Ijtima < Fajar NZ)"
                titik_pertama_global = f"{nama_kota} | Sunset UTC:
{t_sunset.utc_strftime('%H:%M:%S')}"

# ---> LANGKAH 2: JIKA LOKAL GAGAL, SCAN SELURUH DUNIA (PKG 1) <---
if not pkg1_met and not pkg2_met:
    for lon_g in range(180, -181, -15):
        if pkg1_met: break
        for lat_g in range(-60, 61, 15):
            loc_g = wgs84.latlon(lat_g, lon_g)
            est_sunset_utc = 12.0 - (lon_g / 15.0)
            t0_g = self.ts.utc(year, month, day, est_sunset_utc - 2)
            t1_g = self.ts.utc(year, month, day, est_sunset_utc + 2)

            try:
                t_evs, y_evs = almanac.find_discrete(t0_g, t1_g, almanac.sunrise_sunset(self.eph,
loc_g))

                for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
                    if not is_sr:
                        t_ss_sky = self.ts.tt_jd(t_ev)
                        if ephemeris.Date(t_ss_sky.utc_datetime()) < batas_00_utc_ephem:
                            obs_center = earth.at(t_ss_sky)
                            m_app = obs_center.observe(moon).apparent()
                            s_app = obs_center.observe(sun).apparent()
                            elong_g = s_app.separation_from(m_app).degrees

                            if elong_g >= 8.0:
                                ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
                                ha_deg = (t_ss_sky.gast * 15.0) + lon_g - (ra_m.hours * 15.0)
                                sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_g)) +
math.cos(dec_m.radians) * math.cos(math.radians(lat_g)) * math.cos(math.radians(ha_deg))
                                alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

```

```

        if alt_g >= 5.0:
            pkg1_met = True
            pkg_reason = f"PKG 1 (Titik Pindai: Lat {lat_g}°, Lon {lon_g}° [Darat/Lautan])"
            titik_pertama_global = f"Lat {lat_g}°, Lon {lon_g}° | Sunset UTC:
{t_ss_sky.utc_strftime('%H:%M:%S')}"
            break
    except Exception:
        pass

# ---> LANGKAH 3: JIKA PKG 1 TETAP GAGAL, EVALUASI PKG 2 DI BENUA AMERIKA <---
if not pkg1_met and is_ijtima_before_fajr:
    negara_mainland_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan"]
    kota_kepulauan = ["Honolulu", "Papeete", "Alofi", "Pago Pago", "Fernando de Noronha",
"Nuuk", "San Juan", "Santo Domingo", "Havana", "Praia"]

    for negara in negara_mainland_amerika:
        if pkg2_met: break
        if negara in CITY_DB:
            for nama_kota_am, coords in CITY_DB[negara].items():
                if nama_kota_am in kota_kepulauan: continue
                if pkg2_met: break

            lat_a, lon_a = coords
            loc_a = wgs84.latlon(lat_a, lon_a)
            tz_a = int(self.get_tz_from_lon(lon_a))

            t0_a = self.ts.utc(year, month, day, 0 - tz_a)
            t1_a = self.ts.utc(year, month, day, 24 - tz_a)
            try:
                t_evs, y_evs = almanac.find_discrete(t0_a, t1_a, almanac.sunrise_sunset(self.eph,
loc_a))

                for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
                    if not is_sr:
                        t_ss_sky = self.ts.tt_jd(t_ev)
                        if ephemer.Date(t_ss_sky.utc_datetime()) >= batas_00_utc_ephem:
                            obs_center = earth.at(t_ss_sky)
                            m_app = obs_center.observe(moon).apparent()
                            s_app = obs_center.observe(sun).apparent()
                            elong_g = s_app.separation_from(m_app).degrees

                        if elong_g >= 8.0:
                            ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
                            ha_deg = (t_ss_sky.gast * 15.0) + lon_a - (ra_m.hours * 15.0)
                            sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_a)) +
math.cos(dec_m.radians) * math.cos(math.radians(lat_a)) * math.cos(math.radians(ha_deg))
                            alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

                            if alt_g >= 5.0:

```

```

        pkg2_met = True
        pkg_reason = f"PKG 2: Mainland Amerika ({nama_kota_am}, {negara}) &
ljtima < Fajar NZ"
        titik_pertama_global = f"{nama_kota_am}, {negara} | Sunset UTC:
{t_ss_sky.utc_strftime('%H:%M:%S')}"
        break
    except:
        pass

# Konklusi Global Akhir
nm_tt_val = self.ts.from_datetime(dt_utc_ljtima.replace(tzinfo=pytz.utc)).tt
h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"

tgl_1_mas = ephem.Date(batas_00_utc_ephem)
gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
masehi_str_besok = f"{int(gd):02d} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"

tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)
gy2, gm2, gd2, _, _, _ = tgl_lusa.tuple()
masehi_str_lusa = f"{int(gd2):02d} {BULAN_MASEHI[int(gm2)-1]} {format_tahun_aman(int(gy2))}"

# --- PERBAIKAN FORMATTING: INDENTASI 47 SPASI AGAR RATA KANAN ---
indent = ' ' * 47
if pkg1_met:
    status_khgt_global = f"TERPENUHI GLOBAL (PKG 1: Sunset < 00 UTC)\n{indent}* Titik
Pemenuhan : {titik_pertama_global}\n{indent}* Awal Bulan Baru : BESOK (1 {hijri_bulan_str}
{hijri_tahun_str} H = {masehi_str_besok})"
    elif pkg2_met:
        status_khgt_global = f"TERPENUHI GLOBAL (PKG 2: Amerika & ljtima < Fajar NZ)\n{indent}*
Titik Pemenuhan : {titik_pertama_global}\n{indent}* Awal Bulan Baru : BESOK (1 {hijri_bulan_str}
{hijri_tahun_str} H = {masehi_str_besok})"
    else:
        alasan = "Kriteria tidak tercapai di bumi" if is_ljtima_before_fajr else "ljtima > Terbit Fajar NZ"
        status_khgt_global = f"GAGAL GLOBAL PKG 1 & 2 ({alasan})\n{indent}* Awal Bulan Baru : LUSA
(1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_lusa})"

# =====

moonset_str = get_hms_str(t_moonset, tz) if t_moonset is not None else '--.'
lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")

hari_idx = int(t0.whole % 7)
nama_hari_tampil = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]
tahun_str = f"{year} CE" if year > 0 else f"{abs(year-1)} BCE (SM)"
display_date = f"{nama_hari_tampil}, {day:02d}/{month:02d}/{tahun_str}"

```

```

nama_kota = getattr(self, 'opt_city', None)
nama_kota = nama_kota.get() if nama_kota else "Lokasi Anda"
nama_prov = getattr(self, 'opt_prov', None)
nama_prov = nama_prov.get() if nama_prov else ""

# --- PERBAIKAN FORMATTING: ALIGNMENT TITIK DUA (: <45) ---
label_1 = "1. Penentuan KHGT (PKG 1 & PKG 2)"
label_2 = "2. Neo MABIMS (T.Alt>=3°, G.Eln>=6.4°)"
label_3 = "3. Ilyas Criterion (T.Alt>=4°, G.Eln>=10.5°)"
label_4 = "4. Danjon Limit (G.Eln >= 7.0°)"
label_5 = f"5. Yallop (1998) (q-Value = {q_yallop:+.3f})"

# Perbarui Format Output Laporan
report = f""{self.get_header(85)}

```

* Settings:-

- Calculations for Waxing Crescent (New, Evening).
- Crescent Visibility on: {display_date}
- Calculations are Done at Time: {waktu_kalkulasi_str}
- Calculations are Geocentric & Topocentric.
- Refraction Model: Sæmundsson (1986)
- Local Weather : {temp}°C, {pres}mb, {hum}% Hum.
- LOKASI: {nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz})

=====

```

- G. Conjunction (UTC) : {str_ijtima_utc}
- L. Conjunction (LT) : {str_ijtima}
- Julian Date at Time of Calculations: {t_eval_tt:.5f}

- {"Sunset":<22}: {(get_hms_str(t_sunset, tz) + " LT"):<18} {"G. Moon Age":<22}:
{format_time_hms(moon_age_hours):<18}
- {"Moonset":<22}: {(moonset_str + " LT"):<18} {"Moon Lag Time":<22}:
{format_time_hms(lag_time_hours):<18}

- G. Moon Altitude : {format_angle(alt_moon_geo):<16} T. Moon Altitude :
{format_angle(alt_moon_topo):<16} Refraction (Local) : {ref_moon:.4f}°
- G. Sun Altitude : {format_angle(alt_sun_geo):<16} T. Sun Altitude :
{format_angle(alt_sun_topo):<16} Refraction (Local) : {ref_sun:.4f}°

- G. Elongation : {format_angle(elongation):<16} G. Phase Angle :
{format_angle(phase_angle):<16}
- G. Illumination : {f"{illumination:05.4f} %":<16} G. Distance : {f"{dist_km:,.2f} Km":<16}

```

[MULTI-CRITERIA GLOBAL RESEARCH RESULT]

```
{label_1:<45}: {status_khgt_global}
{label_2:<45}: {str_mabims}
{label_3:<45}: {str_ilyas}
{label_4:<45}: {str_danjon}
{label_5:<45}: {yallop_stat}
```

```
=====
"""
    return report

def calculate_visibility(self):
    try:
        # =====
        # TAMBAHAN API CUACA
        # =====
        # Ubah "self.entry_vlat" dan "self.entry_vlon" sesuai nama kotak input di aplikasi Anda
        # (misal: self.entry_lat atau self.lat_entry)
        try:
            lat = float(self.entry_vlat.get())
            lon = float(self.entry_vlon.get())

            # Set status sedang memuat
            self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...",
text_color="#00E5FF"))

            # Panggil API dan Update UI
            status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
            self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca,
text_color=warna_cuaca))
        except Exception as e_cuaca:
            self.after(0, lambda: self.lbl_status_cuaca.configure(text="⊙ Gagal memuat cuaca (Periksa
input koordinat)", text_color="#9E9E9E"))
        # =====

        # --- KODE ASLI ANDA ---
        report = self.generate_visibility_report()
        self.after(0, self.display_result, report)

    except Exception as e:
        import traceback
        self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def calculate_moonphase(self):
    try:
        tahun = int(self.entry_moon_year.get())
        self.auto_switch_ephemeris(tahun)
```

```

# --- TAMBAHAN PENGAMAN BATAS EPHEMERIS (ANTI-ERROR 0) ---
# Jika tahun 0, mulai dari akhir Januari sesuai batas DE406
if tahun == 0:
    t0 = self.ts.utc(0, 1, 30)
else:
    t0 = self.ts.utc(tahun, 1, 1)

# Jika tahun 3000, jangan sampai lewat dari bulan Mei
if tahun == 3000:
    t1 = self.ts.utc(3000, 5, 5)
else:
    t1 = self.ts.utc(tahun, 12, 31, 23, 59, 59)
# -----

t_phases, y_phases = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))

tz_offset = 7.0
rows = []
current_row = ["", "", "", ""]

# Pastikan t_phases tidak kosong
if t_phases is not None:
    for t_obj, phase_idx in get_safe_events(t_phases, y_phases):
        t_lokal = self.ts.tt_jd(t_obj.tt + (tz_offset / 24.0))
        y_l, m_l, d_l, h_l, min_l, s_l = t_lokal.utc

        tahun_str = f"{int(y_l)} M" if y_l > 0 else f"{abs(int(y_l)-1)} SM"
        date_str = f"{int(d_l):02d}/{int(m_l):02d}/{tahun_str} {int(h_l):02d}.{int(min_l):02d}"

        if current_row[phase_idx] != "":
            rows.append(current_row)
            current_row = ["", "", "", ""]
        current_row[phase_idx] = date_str

    if any(current_row):
        rows.append(current_row)

tahun_header = f"{tahun} M" if tahun > 0 else f"{abs(tahun-1)} SM"
info_batas = " (Mulai 30 Jan)" if tahun == 0 else ""

header = f""{self.get_header(103)}
{ "[ Kalkulator Fase Bulan ] ".center(103)}

* Settings:-
- Geocentric Phases of The Moon for The Year: {tahun_header}{info_batas}
- Time Reference is Local WIB (UTC+7)
- Ephemeris segment covers: 0-01-29 to 3000-05-06

```

```

=====
=====
{"New Moon".center(25)}{"First Quarter".center(25)}{"Full Moon".center(25)}{"Last
Quarter".center(25)}
"""

    output_lines = [header]
    for r in rows:
        col0 = r[0].center(25) if r[0] else " " * 25
        col1 = r[1].center(25) if r[1] else " " * 25
        col2 = r[2].center(25) if r[2] else " " * 25
        col3 = r[3].center(25) if r[3] else " " * 25
        output_lines.append(f" {col0}{col1}{col2}{col3}")

    footer = f""""
=====
=====
* Remarks:-
- Date format: dd/mm/yyyy.
- Calculated using Python Skyfield & JPL Ephemeris ({self.ephemeris_name}).
"""

    output_lines.append(footer)
    self.after(0, self.display_result, "\n".join(output_lines))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Kesalahan Rentang Data: {str(e)}\n\n(Catatan: File ephemeris
Anda hanya mendukung hingga 29 Jan 0 SM)")

def calculate_ephemeris(self):
    try:
        import datetime
        import pytz
        import math

        obj_target = self.combo_eph_obj.get()
        time_ref = self.combo_eph_tref.get()
        loc_ref = self.combo_eph_lref.get()

        # 1. Ambil Input (Aman untuk tahun minus)
        sy, sm, sd = int(self.entry_eph_sy.get()), int(self.entry_eph_smon.get()),
int(self.entry_eph_sd.get())
        sh, smin, ssec = int(self.entry_eph_sh.get()), int(self.entry_eph_smin.get()),
int(self.entry_eph_ssec.get())

        ey, em, ed = int(self.entry_eph_ey.get()), int(self.entry_eph_emon.get()),
int(self.entry_eph_ed.get())

```



```

    eh, emin, esec = int(self.entry_eph_eh.get()), int(self.entry_eph_emin.get()),
int(self.entry_eph_esec.get())

    step_val = float(self.entry_eph_step.get())
    step_unit = self.combo_eph_step.get()

    lat = float(self.entry_eph_lat.get())
    lon = float(self.entry_eph_lon.get())
    elev = float(self.entry_eph_elev.get())
    tz = float(self.entry_eph_tz.get())

    # Auto-Switch Ephemeris berdasarkan tahun awal
    self.auto_switch_ephemeris(sy)
    earth = self.eph['earth']

    if obj_target == "Sun":
        target = self.eph['sun']
        radius_km = 696000.0
    else:
        target = self.eph['moon']
        radius_km = 1737.4

    loc = wgs84.latlon(lat, lon, elevation_m=elev)

    # =====
    # 2. PERBAIKAN: KONVERSI WAKTU LOKAL KE JULIAN DATE DENGAN TREF
    # =====
    # Ubah input user (Lokal) menjadi UTC murni dengan mengurangi TZ
    dt_start_local = datetime.datetime(sy, sm, sd, sh, smin, ssec)
    dt_end_local = datetime.datetime(ey, em, ed, eh, emin, esec)

    dt_start_base = dt_start_local - datetime.timedelta(hours=tz)
    dt_end_base = dt_end_local - datetime.timedelta(hours=tz)

    # Eksekusi TREF (Terrestrial Dynamical Time vs Universal Time)
    if "TDT" in time_ref or "TT" in time_ref:
        t_start = self.ts.tt(dt_start_base.year, dt_start_base.month, dt_start_base.day,
                             dt_start_base.hour, dt_start_base.minute, dt_start_base.second)
        t_end = self.ts.tt(dt_end_base.year, dt_end_base.month, dt_end_base.day,
                           dt_end_base.hour, dt_end_base.minute, dt_end_base.second)
    else:
        # Default UT1/UTC
        t_start = self.ts.utc(dt_start_base.year, dt_start_base.month, dt_start_base.day,
                              dt_start_base.hour, dt_start_base.minute, dt_start_base.second)
        t_end = self.ts.utc(dt_end_base.year, dt_end_base.month, dt_end_base.day,
                            dt_end_base.hour, dt_end_base.minute, dt_end_base.second)

    # Pengaman batas bawah ephemeris de406 (0-01-29)

```

```

jd_limit = self.ts.utc(0, 1, 29, 12).tt
current_jd = max(t_start.tt, jd_limit)
end_jd = t_end.tt

if step_unit == "Day": step_jd = step_val
elif step_unit == "Hour": step_jd = step_val / 24.0
elif step_unit == "Minute": step_jd = step_val / 1440.0
else: step_jd = step_val / 86400.0

output_lines = []
lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")

try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele:{elev},
Zone:{tz:.2f})"
except:
    lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f})"

header = f""{self.get_header(128)}
{ "[ Sun & Moon Ephemeris ]".center(128)}

* Settings:-
- {obj_target} Ephemeris
- LOKASI: {lokasi_text}
- Ephemeris are {loc_ref}. Time Reference: {time_ref}.
=====
=====

Time & Date      R. A.    Dec.    Altitude  Azimuth  Latitude Longitude  Distance    SD    Parallax
DT
""

output_lines.append(header)

loop_count = 0
while current_jd <= end_jd:
    loop_count += 1
    if loop_count > 2000: # Batas iterasi keamanan
        output_lines.append("\n[Batas maksimum iterasi tercapai. Ekspor ke CSV jika butuh data
lebih dari 2000 baris]")
        break

    t = self.ts.tt_jd(current_jd)

    geo_apparent = earth.at(t).observe(target).apparent()
    geo_dist_km = geo_apparent.distance().km

```

```

hp_deg = math.degrees(math.asin(6378.137 / geo_dist_km))

if loc_ref == "Topocentric":
    astrometric = (earth + loc).at(t).observe(target)
    apparent = astrometric.apparent()
    ra, dec, distance = apparent.radec(epoch=t) # Ditambahkan epoch=t agar J2000 dinamis ke
Apparent of Date
    # Ephemeris murni umumnya Geometric (tanpa refraksi atmosfer)
    alt_obj, az_obj, _ = apparent.altaz()
    alt_deg, az_deg = alt_obj.degrees, az_obj.degrees
else:
    apparent = geo_apparent
    ra, dec, distance = apparent.radec(epoch=t)
    gast = t.gast
    lst_deg = (gast * 15.0) + lon
    ha_rad = math.radians(lst_deg - (ra.hours * 15.0))
    dec_r = dec.radians
    lat_r = math.radians(lat)
    sin_alt = math.sin(dec_r)*math.sin(lat_r) +
math.cos(dec_r)*math.cos(lat_r)*math.cos(ha_rad)
    alt_deg = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
    az_deg = (math.degrees(math.atan2(-math.sin(ha_rad), math.tan(dec_r)*math.cos(lat_r)-
math.sin(lat_r)*math.cos(ha_rad))) + 360) % 360

    lat_ecl, lon_ecl, _ = apparent.ecliptic_latlon(epoch=t)
    sd_deg = math.degrees(math.asin(radius_km / distance.km))

# =====
# 3. PERBAIKAN: FORMAT PRINT WAKTU KEMBALI KE LOKAL DENGAN AMAN
# =====
dt_utc_out = t.utc_datetime()
dt_display = dt_utc_out + datetime.timedelta(hours=tz)

time_str = f"{dt_display.hour:02d}.{dt_display.minute:02d}.{dt_display.second:02d}
{dt_display.day:02d}/{dt_display.month:02d}/{format_tahun_aman(dt_display.year)}"

ra_str = format_eph_angle(ra.hours * 15.0, is_ra=True)
dec_str = format_eph_angle(dec.degrees)
alt_str = format_eph_angle(alt_deg)
az_str = format_eph_angle(az_deg)
lat_e_str = format_eph_angle(lat_ecl.degrees)
lon_e_str = format_eph_angle(lon_ecl.degrees)
dist_str = f"{distance.km:.1f}".replace(".", ",")
sd_str = format_eph_angle(sd_deg, is_sd=True)
hp_str = format_eph_angle(hp_deg, is_sd=True)
dt_str = f"{t.delta_t:.1f}".replace(".", ",")

```

```

        line = f"{time_str:<21} {ra_str} {dec_str:>9} {alt_str:>9} {az_str:>9} {lat_e_str:>9}
{lon_e_str:>9} {dist_str:>11} {sd_str} {hp_str} {dt_str:>4}"
        output_lines.append(line)

        if loop_count % 4 == 0: output_lines.append("")
        current_jd += step_jd

        output_lines.append("=" * 128)
        output_lines.append("* Remarks:")
        output_lines.append("- Calculated using JPL Ephemeris NASA.")
        output_lines.append("- Altitude data is purely geometric (does not include atmospheric
refraction).")
        self.after(0, self.display_result, "\n".join(output_lines))

    except Exception as e:
        import traceback
        self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def calculate_qiblah(self):
    try:
        year = int(self.entry_qyear.get())
        month = int(self.entry_qmonth.get())
        day = int(self.entry_qday.get())
        target_date = datetime.date(year, month, day)

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_qlat.get())
        lon = float(self.entry_qlon.get())
        tz = float(self.entry_qtz.get())
        mode_qiblah = self.radio_qiblah_var.get()

        phi_k = math.radians(21.4225)
        lam_k = math.radians(39.8262)
        phi = math.radians(lat)
        lam = math.radians(lon)

        y_val = math.sin(lam_k - lam)
        x_val = math.cos(phi) * math.tan(phi_k) - math.sin(phi) * math.cos(lam_k - lam)
        qiblah_angle = math.degrees(math.atan2(y_val, x_val))
        qiblah_angle = (qiblah_angle + 360.0) % 360.0

        target_azimuth = qiblah_angle
        if mode_qiblah == "shadow":
            target_azimuth = (qiblah_angle + 180.0) % 360.0

        earth = self.eph['earth']
        sun = self.eph['sun']

```

```

loc = wgs84.latlon(lat, lon, elevation_m=0.0)

t0 = self.ts.utc(target_date.year, target_date.month, target_date.day, -int(tz))
t1 = self.ts.utc(target_date.year, target_date.month, target_date.day, 24 - int(tz))

tt_array = np.linspace(t0.tt, t1.tt, 1440)
t_search = self.ts.tt_jd(tt_array)

astrometric = (earth + loc).at(t_search).observe(sun)
apparent = astrometric.apparent()
alt_arr, az_arr, _ = apparent.altaz()

az_degrees = az_arr.degrees
alt_degrees = alt_arr.degrees

diffs = (az_degrees - target_azimuth + 180.0) % 360.0 - 180.0

crossings = []
for i in range(len(diffs) - 1):
    if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
        if abs(diffs[i] - diffs[i+1]) < 180.0:
            fraction = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            t_cross_tt = tt_array[i] + fraction * (tt_array[i+1] - tt_array[i])
            alt_cross = alt_degrees[i] + fraction * (alt_degrees[i+1] - alt_degrees[i])

            if alt_cross > 0:
                crossings.append(self.ts.tt_jd(t_cross_tt))

dt_val = t0.delta_t.item() if hasattr(t0.delta_t, 'item') else t0.delta_t

lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")
lon_clean = lon_str.replace(":", ".").replace("'", '0')
lat_clean = lat_str.replace(":", ".").replace("'", '0')

# ---> AMBIL NAMA KOTA & PROVINSI <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{{nama_kota}}, {{nama_prov}} (Long: {{lon_clean}}, Lat: {{lat_clean}}, Zone:{{tz:.2f}})"
except:
    lokasi_text = f"CUSTOM LOCATION (Long: {{lon_clean}}, Lat: {{lat_clean}}, Zone:{{tz:.2f}})"

mode_text = "The Time At Which the SUN is at Qiblah Direction" if mode_qiblah == "sun" else
"The Time At Which the Sun's SHADOW is at Qiblah"
qiblah_str = f"{{qiblah_angle:.1f}}°".replace('.', ',')

header = f"{{self.get_header(84)}}
```

```
{ "[ Qiblah Direction ]".center(84)}
```

```
* Settings:-
```

- Qiblah Direction is: {qiblah_str} From True North
- Qiblah Time you Chose is {mode_text}
- Qiblah Direction for: {target_date.strftime('%d/%m/%Y CE')}
- LOKASI: {lokasi_text}
- No Summer Time.
- Delta T: {dt_val:.1f} Second(s)

```
=====
```

```

Date           Qiblah Time           Qiblah Time
              First Time             Second Time
=====
time_1 = "----"
time_2 = "----"

if len(crossings) > 0:
    dt1 = crossings[0].utc_datetime() + datetime.timedelta(hours=tz)
    time_1 = dt1.strftime("%H:%M:%S")
if len(crossings) > 1:
    dt2 = crossings[1].utc_datetime() + datetime.timedelta(hours=tz)
    time_2 = dt2.strftime("%H:%M:%S")

data_row = f"\n{target_date.strftime('%d/%m/%Y')}      {time_1:^10}      {time_2:^10}"
\n"

footer =
=====
=
```

```
* Remarks:-
```

- Date format: dd/mm/yyyy.
- The Symbol '*' before the date, refers to Summer Time.
- '----' Means that the Sun / Sun's Shadow does not reach the Qiblah Direction on that day.
- Kindly notice that at certain locations and on certain days, the Sun / Sun's Shadow reaches Qiblah direction twice on the same day!

```
=====
```

```
report = header + data_row + footer
self.after(0, self.display_result, report)
```

```
except Exception as e:
```

```
import traceback
self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
```

```
def open_kiblat_online(self):
```

```
    """Menu 26: buka layanan Kiblat Online hanya jika koneksi internet tersedia."""
```

SOURCE CODE KHGT TIMES V8.0

```

url = "https://kasmui.cloud/kiblat/"

def _check_connection_and_open():
    try:
        request = urllib.request.Request(
            url,
            headers={"User-Agent": "Mozilla/5.0 KHGT-Times/7.7.3"}
        )
        with urllib.request.urlopen(request, timeout=7) as response:
            status = getattr(response, "status", 200)
            if status >= 400:
                raise urllib.error.URLError(f"HTTP status {status}")

        # Buka sebagai tab baru pada browser default setelah koneksi berhasil.
        self.after(0, lambda: webbrowser.open_new_tab(url))

    except Exception as exc:
        detail = str(exc)
        self.after(
            0,
            lambda msg=detail: messagebox.showerror(
                "Koneksi Internet Diperlukan",
                "Akses Kiblat Online memerlukan sambungan internet.\n\n"
                "Pastikan komputer terhubung ke internet lalu coba lagi.\n\n"
                f"Detail: {msg}"
            )
        )

threading.Thread(target=_check_connection_and_open, daemon=True).start()

def show_qiblah_map(self):
    try:
        import math
        import numpy as np
        from PIL import Image
        from matplotlib.figure import Figure
        from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
        import customtkinter as ctk
        import os
        from tkinter import messagebox

        map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
        map_file = "map_topografi.jpg"

        # Panggil fungsi unduh
        download_custom_bsp(map_file, map_url)
        full_map_path = os.path.join(BASE_DIR, map_file)

```

```

lons = np.linspace(-180, 180, 360)
lats = np.linspace(-90, 90, 180)
LONS, LATS = np.meshgrid(lons, lats)

phi_k = math.radians(21.4225)
lam_k = math.radians(39.8262)
phi = np.radians(LATS)
lam = np.radians(LONS)

y = np.sin(lam_k - lam)
x = np.cos(phi) * np.tan(phi_k) - np.sin(phi) * np.cos(lam_k - lam)
Q = np.degrees(np.arctan2(y, x))
Q = (Q + 360) % 360

# --- PERBAIKAN: BUAT JENDELA GUI BARU (Menghindari plt.show() yang membuka grafik lain) ---
win_plot = ctk.CTkToplevel(self)
win_plot.title("Accurate Times: Qiblah World Map")
win_plot.geometry("1100x700")
win_plot.attributes("-topmost", True)

# --- PERBAIKAN: GUNAKAN OBJECT-ORIENTED MATPLOTLIB (Mencegah bentrok dengan grafik
3D/Menu lain) ---
fig = Figure(figsize=(11, 6), dpi=100)
ax = fig.add_subplot(111)

if os.path.exists(full_map_path):
    try:
        img = Image.open(full_map_path)
        ax.imshow(img, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
    except Exception as e:
        print("Gagal memuat gambar peta via Pillow:", e)
else:
    messagebox.showwarning("Peringatan", f"File gambar {map_file} gagal diunduh atau tidak
ditemukan di {full_map_path}")

# Arsiran warna peta kiblat
c = ax.contourf(LONS, LATS, Q, levels=np.arange(0, 361, 10), cmap='hsv', alpha=0.4, zorder=1)

# Garis kontur kiblat
ax.contour(LONS, LATS, Q, levels=np.arange(0, 361, 10), colors='black', linewidths=0.6, alpha=0.8,
zorder=2)

# Titik Makkah
ax.scatter(39.8262, 21.4225, color='black', marker='*', s=200, zorder=10, label='Makkah (Kaaba)')

ax.set_title("Accurate Times: Qiblah World Map", fontweight='bold')
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")

```



```

ax.set_xticks(np.arange(-180, 181, 30))
ax.set_yticks(np.arange(-80, 81, 20))
ax.grid(True, linestyle='-', color='gray', linewidth=0.5, zorder=2)
ax.legend(loc='upper right')

fig.tight_layout()

# --- MASUKKAN KE DALAM CANVAS TKINTER ---
frame_canvas = ctk.CTkFrame(win_plot)
frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)

canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
canvas_plot.draw()
canvas_plot.get_tk_widget().pack(fill="both", expand=True)

# Toolbar Navigasi Matplotlib untuk Zoom/Pan/Save
toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
toolbar.update()

except Exception as e:
    from tkinter import messagebox
    messagebox.showerror("Error Map", f"Gagal memuat peta Kiblat: {e}")

def calculate_moontimes(self):
    try:
        year = int(self.entry_mt_year.get())
        month = int(self.entry_mt_month.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_mt_lat.get())
        lon = float(self.entry_mt_lon.get())
        elev = float(self.entry_mt_elev.get())
        tz = float(self.entry_mt_tz.get())

        earth, moon = self.eph['earth'], self.eph['moon']
        loc = wgs84.latlon(lat, lon, elevation_m=elev)

        # Gunakan fungsi safe_monthrange agar kebal tahun minus
        _, num_days = safe_monthrange(year, month)

        # --- PENGAMAN RENTANG DATA ---
        # Menghindari error batas ephemeris 0-01-29
        safe_day_start = 1

```

```

if year == 0 and month == 1:
    safe_day_start = 30

t0_month = self.ts.utc(year, month, safe_day_start, -int(tz))
t1_month = self.ts.utc(year, month, num_days, 24 - int(tz))

f_rs = almanac.risings_and_settings(self.eph, moon, loc)
t_rs, y_rs = almanac.find_discrete(t0_month, t1_month, f_rs)

# Inisialisasi dictionary untuk menampung kejadian per tanggal
# Kita gunakan string "Tahun-Bulan-Hari" sebagai kunci (Key) agar aman
events_by_date = defaultdict(lambda: {'rise': '----', 'transit': '----', 'set': '----'})

# 1. PROSES TERBIT & TERBENAM
if t_rs is not None:
    for t_val, y_val in get_safe_events(t_rs, y_rs):
        # Geser Julian Date secara manual untuk mendapatkan waktu Lokal
        t_lokal = self.ts.tt_jd(t_val.tt + (tz / 24.0))
        yl, ml, dl, hl, minl, sl = t_lokal.utc

        date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
        time_str = f"{int(hl):02d}:{int(minl):02d}"

        if y_val == 1: # Rising
            events_by_date[date_key]['rise'] = time_str
        else: # Setting
            events_by_date[date_key]['set'] = time_str

# 2. PROSES TRANSIT (KULMINASI) - Menggunakan metode pencarian titik tertinggi
for d_idx in range(safe_day_start, num_days + 1):
    t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
    # Ambil 1440 sampel (per menit) dalam 24 jam untuk mencari puncak altitude
    tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
    t_search = self.ts.tt_jd(tt_array)

    alt_arr, _, _ = (earth + loc).at(t_search).observe(moon).apparent().altaz()
    max_idx = np.argmax(alt_arr.degrees)

    t_max = self.ts.tt_jd(tt_array[max_idx])
    # Konversi ke lokal
    t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
    yl, ml, dl, hl, minl, sl = t_max_lokal.utc

    date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
    events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"

# --- PEMBUATAN LAPORAN ---
output_lines = []

```

```

lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "").replace("-", "-")
tahun_header = format_tahun_aman(year)

# ---> MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR GUI <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone:
UTC{'+' if tz >= 0 else ''}{tz})"
except:
    lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if
tz >= 0 else ''}{tz})"

    header = f""{self.get_header(84)}
    {"[ Moon Times ]".center(84)}

* Settings:-
- Moon Times for: {month:02d}/{tahun_header}
- LOKASI: {lokasi_text}
- Refraction is included.
=====

Date      Moonrise      Transit      Moonset
""""

    output_lines.append(header)

    for d in range(1, num_days + 1):
        date_key = f"{year}-{month}-{d}"
        ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})

        # Format tanggal display (dd/mm/yyyy)
        date_display = f"{d:02d}/{month:02d}/{year:04d}"
        if year <= 0:
            date_display = f"{d:02d}/{month:02d}/{abs(year-1):04d} SM"

        line = f"{date_display:<14} {ev['rise']:^12} {ev['transit']:^12} {ev['set']:^12}"
        output_lines.append(line)

    footer =
""""=====
=
* Remarks:-
- Date format: dd/mm/yyyy.
- '----' Means that the event does not occur on that day.
- Calculated using JPL Ephemeris Engine.""""
    output_lines.append(footer)

```

```

self.after(0, self.display_result, "\n".join(output_lines))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def calculate_suntimes(self):
    try:
        year = int(self.entry_st_year.get())
        month = int(self.entry_st_month.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_st_lat.get())
        lon = float(self.entry_st_lon.get())
        elev = float(self.entry_st_elev.get())
        tz = float(self.entry_st_tz.get())

        earth, target_obj = self.eph['earth'], self.eph['sun']
        loc = wgs84.latlon(lat, lon, elevation_m=elev)

        # Gunakan fungsi safe_monthrange agar kebal tahun minus/BCE
        _, num_days = safe_monthrange(year, month)

        # --- PENGAMAN RENTANG DATA ---
        # Menghindari error batas ephemeris NASA de406 (0-01-29)
        safe_day_start = 1
        if year == 0 and month == 1:
            safe_day_start = 30

        t0_month = self.ts.utc(year, month, safe_day_start, -int(tz))
        t1_month = self.ts.utc(year, month, num_days, 24 - int(tz))

        # Cari Terbit & Terbenam Matahari
        f_rs = almanac.sunrise_sunset(self.eph, loc)
        t_rs, y_rs = almanac.find_discrete(t0_month, t1_month, f_rs)

        # Inisialisasi dictionary untuk menampung kejadian per tanggal
        # Key: "Tahun-Bulan-Hari" (String) agar aman dari limitasi datetime
        events_by_date = defaultdict(lambda: {'rise': '----', 'transit': '----', 'set': '----'})

        # 1. PROSES SUNRISE & SUNSET
        if t_rs is not None:
            for t_val, y_val in get_safe_events(t_rs, y_rs):
                # Geser Julian Date secara manual untuk mendapatkan waktu LOKAL
                t_lokal = self.ts.tt_jd(t_val.tt + (tz / 24.0))
                yl, ml, dl, hl, minl, sl = t_lokal.utc

```

```

date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
time_str = f"{int(hl):02d}:{int(minl):02d}"

if y_val == 1: # Sunrise
    events_by_date[date_key]['rise'] = time_str
else: # Sunset
    events_by_date[date_key]['set'] = time_str

# 2. PROSES TRANSIT (ZAWAL) - Mencari titik kulminasi atas
for d_idx in range(safe_day_start, num_days + 1):
    t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
    # Sampel 1440 titik (per menit) untuk akurasi tinggi
    tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
    t_search = self.ts.tt_jd(tt_array)

    alt_arr, _, _ = (earth + loc).at(t_search).observe(target_obj).apparent().altaz()
    max_idx = np.argmax(alt_arr.degrees)

    t_max = self.ts.tt_jd(tt_array[max_idx])
    # Konversi ke waktu lokal
    t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
    yl, ml, dl, hl, minl, sl = t_max_lokal.utc

    date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
    events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"

# --- GENERATE LAPORAN TEKS ---
output_lines = []
lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")
tahun_header = format_tahun_aman(year)

# ---> MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR GUI <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone:
UTC{'+' if tz >= 0 else ''}{tz})"
except:
    lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if
tz >= 0 else ''}{tz})"

    header = f""{self.get_header(84)}
    {"[ Sun Times ]".center(84)}

* Settings:-
- Sun Times for: {month:02d}/{tahun_header}
- LOKASI: {lokasi_text}

```

- Refraction is included.

```
=====

Date          Sunrise      Transit      Sunset
=====

output_lines.append(header)

for d in range(1, num_days + 1):
    date_key = f"{year}-{month}-{d}"
    ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})

    # Format tampilan tanggal (Bebas crash tahun minus)
    date_display = f"{d:02d}/{month:02d}/{year:04d}"
    if year <= 0:
        # Contoh: 0 -> 3001 SM
        date_display = f"{d:02d}/{month:02d}/{abs(year-1):04d} SM"

    line = f"{date_display:<14} {ev['rise']:^12} {ev['transit']:^12} {ev['set']:^12}"
    output_lines.append(line)

    footer =
    """=====
    =
    * Remarks:-
    - Date format: dd/mm/yyyy.
    - '----' Means that the event does not occur on that day.
    - Calculated using JPL Ephemeris NASA."""
    output_lines.append(footer)

    self.after(0, self.display_result, "\n".join(output_lines))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def calculate_prayertimes(self):
    try:
        # 1. AMBIL INPUT UI (DIPERTAHANKAN UTUH)
        year = int(self.entry_pt_year.get())
        month = int(self.entry_pt_month.get())
        day = int(self.entry_pt_day.get())

        # Engine auto-switch ephemeris
        self.auto_switch_ephemeris(year)

        lat = float(self.entry_pt_lat.get())
        lon = float(self.entry_pt_lon.get())
        elev = float(self.entry_pt_elev.get())
```

```

tz = float(self.entry_pt_tz.get())

temp = float(self.entry_pt_temp.get())
pres = float(self.entry_pt_pres.get())
hum = float(self.entry_pt_hum.get())

mode_pt = self.combo_pt_mode.get()
ikhtiyat_sec = int(self.entry_pt_ikhtiyat.get())
fmt_pt = self.combo_pt_fmt.get()

# --- LOGIKA MAZHAB & METODE ASLI (DIPERTAHANKAN) ---
mazhab_val = self.combo_pt_mazhab.get()
method_val = self.combo_pt_method.get()

if "Hanafi" in mazhab_val:
    asr_factor = 2.0
    mazhab_name = "Hanafi"
else:
    asr_factor = 1.0
    mazhab_name = "Standard (Syafi'i, Maliki, Hanbali)"

if "Kemenag" in method_val:
    fajr_angle, isha_angle = -20.0, -18.0
elif "Muslim World League" in method_val:
    fajr_angle, isha_angle = -18.0, -17.0
elif "ISNA" in method_val:
    fajr_angle, isha_angle = -15.0, -15.0
elif "Egypt" in method_val:
    fajr_angle, isha_angle = -19.5, -17.5
else:
    fajr_angle, isha_angle = -18.0, -18.0

earth = self.eph['earth']
sun = self.eph['sun']
loc = wgs84.latlon(lat, lon, elevation_m=elev)

# --- PERBAIKAN: GUNAKAN safe_monthrange UNTUK TAHUN MINUS ---
_, num_days = safe_monthrange(year, month)

# Pengaman batas bawah ephemeris de406 (0-01-29)
start_d_iter = 1
if year == 0 and month == 1: start_d_iter = 30

if "Bulanan" in mode_pt:
    days_to_calc = list(range(start_d_iter, num_days + 1))
    date_info = f"{month:02d}/{format_tahun_aman(year)}"
else:
    days_to_calc = [day]

```

```

date_info = f"{day:02d}/{month:02d}/{format_tahun_aman(year)}"

# Ambil Delta T secara aman untuk tahun minus
dt_val = self.ts.utc(year, month, start_d_iter).delta_t
lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")
lon_clean = lon_str.replace(":", "").replace("'", '0')
lat_clean = lat_str.replace(":", "").replace("'", '0')

# ---> AMBIL NAMA KOTA & PROVINSI UNTUK GAMBAR <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_clean}, Lat: {lat_clean}, Ele:{elev}m, Zone:
UTC{'+' if tz >= 0 else ''}{tz})"
except:
    lokasi_text = f"CUSTOM LOCATION (Long: {lon_clean}, Lat: {lat_clean}, Ele:{elev}m, Zone:
UTC{'+' if tz >= 0 else ''}{tz})"

# --- FORMAT HEADER TABEL YANG DIPERBAIKI (ANTI BERANTAKAN) ---
if "HH:MM:SS" in fmt_pt:
    pad = 10
    hdr_line1 = f" {'Date':<15} {'Fajer':^{pad}} {'Shuroq':^{pad}} {'Dhuha':^{pad}}
{'Dhohur':^{pad}} {'Aser':^{pad}} {'Maghreb':^{pad}} {'Isha':^{pad}} {'Midnight':^15}"
    hdr_line2 = f" {'':<15} {'B. Twi.':^{pad}} {'Sunrise':^{pad}} {'+4.5°':^{pad}} {'Transit':^{pad}}
{'-----':^{pad}} {'Sunset':^{pad}} {'E. Twi.':^{pad}} {'(Mid/Last 1/3)':^15}"
    sep = "=" * 118
else:
    pad = 8
    hdr_line1 = f" {'Date':<15} {'Fajer':^{pad}} {'Shuroq':^{pad}} {'Dhuha':^{pad}}
{'Dhohur':^{pad}} {'Aser':^{pad}} {'Maghreb':^{pad}} {'Isha':^{pad}} {'Midnight':^15}"
    hdr_line2 = f" {'':<15} {'B. Twi.':^{pad}} {'Sunrise':^{pad}} {'+4.5°':^{pad}} {'Transit':^{pad}}
{'-----':^{pad}} {'Sunset':^{pad}} {'E. Twi.':^{pad}} {'(Mid/Last 1/3)':^15}"
    sep = "=" * 102

report_lines = []
header_text = f""{self.get_header(len(sep))}
{"[ Islamic Prayer Times ]".center(len(sep))}

* Settings:-
- Prayer times for: {date_info}
- LOKASI: {lokasi_text}
- No Summer Time.
- Method: {method_val}
- Fajer Angle: {abs(fajr_angle)}°, Isha Angle: {abs(isha_angle)}°
- Refraction: Temp.: {temp} °C Pres.: {pres} mb Humidity: {hum} % Temp.Rate: 0.0065 K/m
- Ikhtiyat Time: {ikhtiyat_sec} Second(s)
- Mazhab (Asr): {mazhab_name} (Shadow multiplier: {asr_factor}x)

```



```

- Delta T: {float(dt_val):.1f} Second(s)
{sep}

{hdr_line1}
{hdr_line2}"""
    report_lines.append(header_text)

def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
    diffs = alt_arr - target_alt
    for i in range(len(diffs)-1):
        if direction == 'up' and diffs[i] <= 0 and diffs[i+1] > 0:
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
        elif direction == 'down' and diffs[i] >= 0 and diffs[i+1] < 0:
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
    return None

def fmt_time(tt_val, is_shuroq=False):
    if tt_val is None: return "----".center(pad)
    offset_i = (ikhtiyat_sec / 86400.0)
    if is_shuroq:
        t_res = self.ts.tt_jd(tt_val + (tz / 24.0) - offset_i)
    else:
        t_res = self.ts.tt_jd(tt_val + (tz / 24.0) + offset_i)

    _, _, h_f, m_f, s_f = t_res.utc

    if "HH:MM:SS" in fmt_pt:
        return f"{int(h_f):02d}:{int(m_f):02d}:{int(s_f):02d}"
    else:
        return f"{int(h_f):02d}.{int(m_f):02d}"

# --- LOOPING UTAMA (LOGIKA HISAB UTUH) ---
for d in days_to_calc:
    t0 = self.ts.utc(year, month, d, -int(tz))
    t1 = self.ts.utc(year, month, d, 24 - int(tz))
    tt_array = np.linspace(t0.tt, t1.tt, 2880)
    t_search = self.ts.tt_jd(tt_array)

    alt_deg = (earth + loc).at(t_search).observe(sun).apparent().altaz(temperature_C=temp,
pressure_mbar=pres)[0].degrees

    idx_noon = np.argmax(alt_deg)
    tt_dhohur = tt_array[idx_noon]
    alt_noon = alt_deg[idx_noon]

    alt_am, tt_am = alt_deg[:idx_noon], tt_array[:idx_noon]

```

```

alt_pm, tt_pm = alt_deg[idx_noon:], tt_array[idx_noon:]

zenith_noon = 90.0 - alt_noon
shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
shadow_asr = asr_factor + shadow_noon
alt_asr_target = math.degrees(math.atan(1.0 / shadow_asr))

val_fajr = find_crossing(alt_am, tt_am, fajr_angle, 'up')
val_shuroq = find_crossing(alt_am, tt_am, -0.833, 'up')
val_dhuha = find_crossing(alt_am, tt_am, 4.5, 'up')
val_asr = find_crossing(alt_pm, tt_pm, alt_asr_target, 'down')
val_maghreb = find_crossing(alt_pm, tt_pm, -0.833, 'down')
val_isha = find_crossing(alt_pm, tt_pm, isha_angle, 'down')

highlat_method = self.combo_pt_highlat.get()

if "Lintang Normal" not in highlat_method:
    if val_maghreb is not None and val_shuroq is not None:
        durasi_siang = val_maghreb - val_shuroq
        durasi_malam = 1.0 - durasi_siang

    if "1/2 Malam" in highlat_method:
        porsi = durasi_malam / 2.0
        if val_isha is None: val_isha = val_maghreb + porsi
        if val_fajr is None: val_fajr = val_shuroq - porsi

    elif "1/7 Malam" in highlat_method:
        porsi = durasi_malam / 7.0
        if val_isha is None: val_isha = val_maghreb + porsi
        if val_fajr is None: val_fajr = val_shuroq - porsi

    elif "Proporsi Sudut" in highlat_method:
        porsi_isha = durasi_malam * (abs(isha_angle) / 60.0)
        porsi_fajr = durasi_malam * (abs(fajr_angle) / 60.0)
        if val_isha is None: val_isha = val_maghreb + porsi_isha
        if val_fajr is None: val_fajr = val_shuroq - porsi_fajr

    elif "Aqrab al-Balad" in highlat_method:
        if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
        if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)

if val_maghreb is None or val_shuroq is None:
    if tt_dhohur is not None:
        if val_shuroq is None: val_shuroq = tt_dhohur - (6.0 / 24.0)
        if val_maghreb is None: val_maghreb = tt_dhohur + (6.0 / 24.0)
        if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
        if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
        if val_dhuha is None: val_dhuha = val_shuroq + (1.0 / 24.0)

```

```

        if val_asr is None: val_asr = tt_dhohur + (3.0 / 24.0)

    str_midnight = "----"
    if val_maghreb is not None and val_fajr is not None:
        durasi_malam = (val_fajr + 1.0) - val_maghreb
        t_mid = self.ts.tt_jd(val_maghreb + (durasi_malam/2.0) + (tz/24.0))
        t_last = self.ts.tt_jd(val_maghreb + (durasi_malam*2/3.0) + (tz/24.0))
        str_midnight =
f"{int(t_mid.utc[3]):02d}:{int(t_mid.utc[4]):02d}/{int(t_last.utc[3]):02d}:{int(t_last.utc[4]):02d}"

    str_fajr = fmt_time(val_fajr)
    str_shuroq = fmt_time(val_shuroq, is_shuroq=True)
    str_dhuha = fmt_time(val_dhuha)
    str_dhohur = fmt_time(tt_dhohur)
    str_asr = fmt_time(val_asr)
    str_maghreb = fmt_time(val_maghreb)
    str_isha = fmt_time(val_isha)

    date_str = f"{d:02d}/{month:02d}/{format_tahun_aman(year)}"

    line = f" {date_str:<15} {str_fajr:^{pad}} {str_shuroq:^{pad}} {str_dhuha:^{pad}}
{str_dhohur:^{pad}} {str_asr:^{pad}} {str_maghreb:^{pad}} {str_isha:^{pad}} {str_midnight:^15}"

    report_lines.append(line)

    report_lines.append(f"{sep}")
    report_lines.append("""
* Remarks:-
- Date format: dd/mm/yyyy.
- Fajer: Beginning of Astronomical Twilight. | Shuroq: Sunrise.
- Dhohur: Transit of Sun. | Maghreb: Sunset. | Isha: End of Astronomical Twilight.
- Calculated using JPL NASA Engine. Supported Year 0 to 3000.""")

    final_report = "\n".join(report_lines)

    # --- 1. TAMPILKAN TEKS KE LAYAR GUI ---
    self.after(0, self.display_result, final_report)

    # --- 2. GENERATE GAMBAR OTOMATIS & BUKA ---
    # Perbaikan: Gambar akan dibuat baik itu Bulanan maupun Harian
    self.after(0, lambda: self.lbl_status.configure(text="Sedang membuat gambar jadwal...",
text_color="#FFAB40"))

    # Auto-Generate background template jika file hilang
    template_path = os.path.join(BASE_DIR, "template_kosong.jpg")
    if not os.path.exists(template_path):
        try:
            img_bg = Image.new("RGB", (2100, 2400), "#0F172A")

```

```

        draw_bg = ImageDraw.Draw(img_bg)
        draw_bg.rectangle([0, 0, 2100, 280], fill="#1E88E5")
        img_bg.save(template_path)
    except Exception as e:
        print("Gagal membuat template otomatis:", e)

    try: nama_kota_t = self.opt_city.get()
    except: nama_kota_t = "Custom"

    output_filename = f"Jadwal_Shalat_{nama_kota_t.replace(' ', '_')}_month:02d}_{year}.png"
    output_path = os.path.join(BASE_DIR, output_filename)

    try:
        _util_generate_image(final_report, template_path, output_path)

        if platform.system() == 'Windows':
            os.startfile(output_path)
        elif platform.system() == 'Darwin':
            subprocess.call(['open', output_path])
        else:
            subprocess.call(['xdg-open', output_path])

        self.after(0, lambda f=output_filename: self.lbl_status.configure(text=f"Selesai. Gambar
disimpan sbg: {f}", text_color="#00E676"))
    except Exception as e_img:
        self.after(0, lambda e=e_img: messagebox.showwarning("Generate Gambar Gagal", f"Gagal
membuat gambar jadwal:\n{e}"))

    except Exception as e:
        import traceback
        self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def calculate_visibility_map(self):
    if not HAS_MAP_TOOLS:
        self.after(0, self.display_error, "Fitur Peta HD dinonaktifkan.\nLibrary 'scipy' tidak ditemukan.")
        return

    try:
        year = int(self.entry_vmyear.get())
        month = int(self.entry_vmmonth.get())
        day = int(self.entry_vmday.get())

        crit = self.combo_vmcrit.get()
        param_target = self.combo_vmparam.get()

        self.auto_switch_ephemeris(year)
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

```

```

lons_coarse = []
lats_coarse = []
alt_data = []
elong_data = []
arcv_data = []
illum_data = []

grid_step = 5
is_khgt_map = param_target == "Interseksi (Peta KHGT 2D)"

# Rentang Longitude: 0-360 untuk Peta KHGT 2D, -180 to 180 untuk standar
lon_range = range(0, 361, grid_step) if is_khgt_map else range(-180, 181, grid_step)

for lat in range(-90, 91, grid_step):
    lat_rad = math.radians(lat)

    for plot_lon in lon_range:
        if is_khgt_map:
            if plot_lon < 180:
                real_lon = plot_lon
                # Belahan Timur (Kiri Peta) dihitung memakai Hari Besoknya (D+1)
                dt_calc = datetime.date(year, month, day) + datetime.timedelta(days=1)
            else:
                real_lon = plot_lon - 360
                # Belahan Barat (Kanan Peta) dihitung memakai Hari Input (D)
                dt_calc = datetime.date(year, month, day)
            calc_y, calc_m, calc_d = dt_calc.year, dt_calc.month, dt_calc.day
        else:
            real_lon = plot_lon
            calc_y, calc_m, calc_d = year, month, day

        t_noon_utc = self.ts.utc(calc_y, calc_m, calc_d, 12)
        sun_geo = earth.at(t_noon_utc).observe(sun).apparent()
        _, dec_sun, _ = sun_geo.radec()
        dec_rad = dec_sun.radians

        noon_utc_hour = 12.0 - (real_lon / 15.0)

        cos_h = -math.tan(lat_rad) * math.tan(dec_rad)
        if cos_h < -1 or cos_h > 1:
            sunset_utc_hour = noon_utc_hour + 6.0
        else:
            h_rad = math.acos(cos_h)
            h_hours = math.degrees(h_rad) / 15.0
            sunset_utc_hour = noon_utc_hour + h_hours

        t_sunset = self.ts.utc(calc_y, calc_m, calc_d, sunset_utc_hour)

```

```

obs = earth.at(t_sunset)
s_app = obs.observe(sun).apparent()
m_app = obs.observe(moon).apparent()

gmst = t_sunset.gast # Perbaikan: Gunakan GAST (Apparent), bukan GMST
lst_deg = (gmst * 15.0) + real_lon

def quick_alt_geo(app_obj, l_rad):
    # Perbaikan: Tambahkan epoch=t_sunset untuk menghindari pergeseran J2000
    ra, dec_o, _ = app_obj.radec(epoch=t_sunset)
    ha_deg = lst_deg - (ra.hours * 15.0)
    d_rad = dec_o.radians
    ha_rad = math.radians(ha_deg)
    sin_alt = math.sin(d_rad) * math.sin(l_rad) + math.cos(d_rad) * math.cos(l_rad) *
math.cos(ha_rad)
    return math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

# 1. Kalkulasi GEOSENTRIK (Untuk KHGT)
m_alt_geo = quick_alt_geo(m_app, lat_rad)
s_alt_geo = quick_alt_geo(s_app, lat_rad)
elong_geo = s_app.separation_from(m_app).degrees

# 2. Kalkulasi TOPOSENTRIK (Untuk MABIMS, Yallop, Ilyas)
# Menggunakan Native Skyfield agar efek Parallax & Refraksi terhitung sempurna
loc_topo = wgs84.latlon(lat, real_lon, elevation_m=0)
topo_obs = (earth + loc_topo).at(t_sunset)
m_alt_topo = topo_obs.observe(moon).apparent().altaz(temperature_C=25,
pressure_mbar=1010)[0].degrees
s_alt_topo = topo_obs.observe(sun).apparent().altaz(temperature_C=25,
pressure_mbar=1010)[0].degrees

arcv_topo = m_alt_topo - s_alt_topo

illum = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
illum_val = illum.item() if hasattr(illum, 'item') else illum

lons_coarse.append(plot_lon)
lats_coarse.append(lat)

# Simpan data Toposentrik dan Geosentrik berdasarkan mode peta
if is_khgt_map or "KHGT" in crit or "Danjon" in crit:
    alt_data.append(m_alt_geo) # Peta KHGT murni pakai Geosentrik
    arcv_data.append(m_alt_geo - s_alt_geo)
else:
    alt_data.append(m_alt_topo) # Peta MABIMS/Yallop murni pakai Toposentrik
    arcv_data.append(arcv_topo)

elong_data.append(elong_geo) # Elongasi selalu Geosentris

```

```

        illum_data.append(illum_val * 100.0)

    data_dict = {
        'lons': np.array(lons_coarse),
        'lats': np.array(lats_coarse),
        'alt': np.array(alt_data),
        'elong': np.array(elong_data),
        'arcv': np.array(arcv_data),
        'illum': np.array(illum_data)
    }

    self.after(0, self.show_hd_vismap, data_dict, f"{day} {calendar.month_name[month]} {year}", crit,
    param_target)

    except Exception as e:
        import traceback
        self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def show_hd_vismap(self, plot_data, date_str, crit_name, param_target):
    try:
        points = np.column_stack((plot_data['lons'], plot_data['lats']))
        is_khgt_map = param_target == "Interseksi (Peta KHGT 2D)"

        if is_khgt_map:
            lon_fine = np.arange(0, 360.25, 0.2)
        else:
            lon_fine = np.arange(-180, 180.25, 0.2)

        lat_fine = np.arange(-90, 90.25, 0.2)
        LONS_fine, LATS_fine = np.meshgrid(lon_fine, lat_fine)

        grid_alt = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine), method='cubic')
        grid_elong = interp.griddata(points, plot_data['elong'], (LONS_fine, LATS_fine), method='cubic')
        grid_arcv = interp.griddata(points, plot_data['arcv'], (LONS_fine, LATS_fine), method='cubic')
        grid_illum = interp.griddata(points, plot_data['illum'], (LONS_fine, LATS_fine), method='cubic')

        grid_alt_near = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine), method='nearest')
        grid_alt[np.isnan(grid_alt)] = grid_alt_near[np.isnan(grid_alt)]

        grid_elong_near = interp.griddata(points, plot_data['elong'], (LONS_fine, LATS_fine),
        method='nearest')
        grid_elong[np.isnan(grid_elong)] = grid_elong_near[np.isnan(grid_elong)]

        grid_arcv_near = interp.griddata(points, plot_data['arcv'], (LONS_fine, LATS_fine),
        method='nearest')
        grid_arcv[np.isnan(grid_arcv)] = grid_arcv_near[np.isnan(grid_arcv)]

```

```

        grid_illum_near = interp.griddata(points, plot_data['illum'], (LONS_fine, LATS_fine),
method='nearest')
        grid_illum[np.isnan(grid_illum)] = grid_illum_near[np.isnan(grid_illum)]

        win_plot = ctk.CTkToplevel(self)
        win_plot.title("High Resolution Crescent Map" if not is_khgt_map else "Peta Kalender Hijriah
Global Tunggal (KHGT 2D)")
        win_plot.geometry("1100x750")
        win_plot.attributes("-topmost", True)

        fig, ax = plt.subplots(figsize=(12, 6.5), dpi=110)

        map_url = "https://hisabmu.com/aifiki/berbagi/map_topografi.jpg"
        map_file = "map_topografi.jpg"
        download_custom_bsp(map_file, map_url)
        full_map_path = os.path.join(BASE_DIR, map_file)

        if os.path.exists(full_map_path):
            try:
                img = Image.open(full_map_path)
                img_arr = np.array(img)

                if is_khgt_map:
                    # Potong dan geser gambar topografi agar Pasifik di tengah (0 hingga 360)
                    mid = img_arr.shape[1] // 2
                    img_shifted = np.concatenate((img_arr[:, mid:], img_arr[:, :mid]), axis=1)
                    ax.imshow(img_shifted, extent=[0, 360, -90, 90], aspect='auto', alpha=1.0, zorder=0)
                else:
                    # Format standar -180 hingga 180
                    ax.imshow(img_arr, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
            except Exception as e:
                print("Gagal memuat gambar peta:", e)

        # =====
        # PLOTTING KHUSUS KHGT MAP (GABUNGAN 2 HARI SEAMLESS)
        # =====
        if is_khgt_map:
            grid_score = np.zeros_like(grid_alt)
            grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1

            cmap_inter = ListedColormap([(0, 0, 0, 0), '#00E676'])
            bounds_inter = [-0.5, 0.5, 1.5]
            norm_inter = BoundaryNorm(bounds_inter, cmap_inter.N)

            ax.contourf(LONS_fine, LATS_fine, grid_score, cmap=cmap_inter, norm=norm_inter, alpha=0.2,
zorder=2, antialiased=True)

```



```

cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0], colors=['#1A237E'],
linewidths=1.5, linestyle='solid', zorder=3)
ax.clabel(cs_alt, inline=True, fontsize=10, fmt='Alt 5°')

cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0], colors=['#D32F2F'],
linewidths=1.5, linestyle='solid', zorder=3)
ax.clabel(cs_elong, inline=True, fontsize=10, fmt='Elong 8°')

# Garis Pembatas Penanggalan di tengah (Bujur 180)
ax.axvline(180, color='black', linewidth=1.5, zorder=4)

# Cetak Teks Tanggal di Puncak Garis
try:
    d = int(self.entry_vmday.get())
    m = int(self.entry_vmmonth.get())
    y = int(self.entry_vmyear.get())
    dt_kanan = datetime.date(y, m, d)
    dt_kiri = dt_kanan + datetime.timedelta(days=1)

    bulan_nama = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]

    lbl_kanan = f"{dt_kanan.day} {bulan_nama[dt_kanan.month]} {dt_kanan.year}"
    lbl_kiri = f"{dt_kiri.day} {bulan_nama[dt_kiri.month]} {dt_kiri.year}"
except:
    lbl_kanan = "Hari 1"
    lbl_kiri = "Hari 2"

bbox_props = dict(boxstyle="square,pad=0.3", fc="white", ec="black", alpha=0.85)
ax.text(178, 85, lbl_kiri, color='black', ha='right', va='top', fontsize=11, fontweight='bold',
bbox=bbox_props, zorder=5)
ax.text(182, 85, lbl_kanan, color='black', ha='left', va='top', fontsize=11, fontweight='bold',
bbox=bbox_props, zorder=5)

ax.set_xlim(0, 360)
ax.set_ylim(-75, 90) # Potong sedikit Antartika agar proporsional
ax.set_title(f"Peta Kalender Hijriah Global Tunggal (KHGT)\nMode: Altitude (Geo), Elongasi
(Geo)", fontweight='bold', pad=15)

# Mengubah Sumbu X menjadi Format Bujur
ticks_x = np.arange(0, 361, 30)
labels_x = [f"{val}° BT" if val < 180 else f"{360 - val}° BB" if val > 180 else "180°" for val in
ticks_x]

ax.set_xticks(ticks_x)
ax.set_xticklabels(labels_x)
ax.set_yticks(np.arange(-60, 61, 30))

```

```

        p_interseksi = mpatches.Patch(color='#00E676', alpha=0.2, label='Area Hijau: Memenuhi
Kriteria KHGT')
        line_alt = plt.Line2D([0], [0], color='#1A237E', lw=1.5, linestyle='solid', label='Batas Altitude 5°
(G)')
        line_elong = plt.Line2D([0], [0], color='#D32F2F', lw=1.5, linestyle='solid', label='Batas Elongasi
8° (G)')
        ax.legend(handles=[p_interseksi, line_alt, line_elong], loc='lower center', ncol=3,
bbox_to_anchor=(0.5, -0.15), fontsize='small')

# =====
# RENDER UNTUK MODE PETA STANDAR LAINNYA
# =====
else:
    if param_target == "Kriteria Visibilitas":
        grid_score = np.zeros_like(grid_alt)

        if "KHGT" in crit_name or "Diyanet" in crit_name:
            grid_score[(grid_alt >= 0) & (grid_arcv >= 0)] = 1
            grid_score[(grid_alt >= 5) & (grid_elong >= 8)] = 2
            cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
            bounds = [-0.5, 0.5, 1.5, 2.5]
            norm = BoundaryNorm(bounds, cmap.N)
            cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

            p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi KHGT (Geo Alt>=5°, Geo
Eln>=8°)')
            p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
            p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible (Bawah Ufuk)')
            ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18),
fontsize='small')

        elif "MABIMS" in crit_name:
            grid_score[(grid_alt >= 0) & (grid_arcv >= 0)] = 1
            grid_score[(grid_alt >= 3) & (grid_elong >= 6.4)] = 2
            cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
            bounds = [-0.5, 0.5, 1.5, 2.5]
            norm = BoundaryNorm(bounds, cmap.N)
            cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

            p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi Neo MABIMS (Topo
Alt>=3°, Geo Eln>=6.4°)')
            p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
            p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
            ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18),
fontsize='small')

```

```

elif "Ilyas" in crit_name:
    grid_score[(grid_alt >= 0) & (grid_arcv >= 0)] = 1
    grid_score[(grid_alt >= 4) & (grid_elong >= 10.5)] = 2
    cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
    bounds = [-0.5, 0.5, 1.5, 2.5]
    norm = BoundaryNorm(bounds, cmap.N)
    cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

    p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi Ilyas (Topo Alt>=4°, Geo
Eln>=10.5°)')
    p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
    p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
    ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18),
fontsize='small')

elif "Danjon" in crit_name:
    grid_score[(grid_alt >= 0) & (grid_arcv >= 0)] = 1
    grid_score[(grid_elong >= 7.0)] = 2
    cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
    bounds = [-0.5, 0.5, 1.5, 2.5]
    norm = BoundaryNorm(bounds, cmap.N)
    cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

    p1 = mpatches.Patch(color='#00E676', label='Green: Melewati Limit Danjon (Geo Eln >=
7.0°)')
    p2 = mpatches.Patch(color='#FFFFFF', label='White: Bawah Limit Danjon')
    p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
    ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18),
fontsize='small')

elif "Yallop" in crit_name:
    W_grid = 15.5 * (1 - np.cos(np.radians(grid_elong)))
    q_grid = grid_arcv - (11.8371 - 6.3226 * W_grid + 0.7319 * (W_grid**2) - 0.1018 *
(W_grid**3))

    grid_score[:] = 0 # F
    grid_score[(q_grid > -0.232)] = 1 # D
    grid_score[(q_grid > -0.160)] = 2 # C
    grid_score[(q_grid > -0.014)] = 3 # B
    grid_score[(q_grid > 0.216)] = 4 # A

    cmap = ListedColormap(['#FF5252', '#2979FF', '#E040FB', '#FFD54F', '#00E676'])
    bounds = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5]
    norm = BoundaryNorm(bounds, cmap.N)
    cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

```

```

p1 = mpatches.Patch(color='#FF5252', label='F: Not Visible (Topo q-Value)')
p2 = mpatches.Patch(color='#2979FF', label='D: Only with Optical Aid (Topo q)')
p3 = mpatches.Patch(color='#E040FB', label='C: Need Optical Aid First (Topo q)')
p4 = mpatches.Patch(color='#FFD54F', label='B: Visible / Perfect Cond. (Topo q)')
p5 = mpatches.Patch(color='#00E676', label='A: Easily Visible / Naked Eye (Topo q)')
ax.legend(handles=[p1, p2, p3, p4, p5], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -
0.22), fontsize='small')

```

```

elif "Odeh" in crit_name:
    grid_score[:] = 1
    grid_score[(grid_arcv < 0)] = 0
    grid_score[(grid_arcv > 6.0) & (grid_elong > 8.0)] = 2
    grid_score[(grid_arcv > 8.0) & (grid_elong > 10.0)] = 3
    grid_score[(grid_arcv > 10.4) & (grid_elong > 12.0)] = 4

    cmap = ListedColormap(['#FF5252', '#FFFFFF', '#2979FF', '#E040FB', '#00E676'])
    bounds = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5]
    norm = BoundaryNorm(bounds, cmap.N)
    cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, antialiased=True)

```

```

p1 = mpatches.Patch(color='#FF5252', label='Red: Impossible (Topo V-Value)')
p2 = mpatches.Patch(color='#FFFFFF', label='White: Not Possible (Topo V)')
p3 = mpatches.Patch(color='#2979FF', label='Blue: Need Optical Aid (Topo V)')
p4 = mpatches.Patch(color='#E040FB', label='Magenta: Naked Eye / Perfect Cond (Topo
V)')
p5 = mpatches.Patch(color='#00E676', label='Green: Easily Visible (Topo V)')
ax.legend(handles=[p1, p2, p3, p4, p5], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -
0.22), fontsize='small')

```

```

# PLOTTING DILAKUKAN SEKALI SAJA DI SINI (DILUAR IF-ELIF)
# PERBAIKAN: Penambahan parameter `levels=bounds` sangat krusial!
norm = BoundaryNorm(bounds, cmap.N)
cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm,
alpha=0.2, zorder=2, antialiased=True)

```

```

elif param_target == "Interseksi":
    grid_score = np.zeros_like(grid_alt)
    grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1
    cmap_inter = ListedColormap([(0, 0, 0, 0), '#00E676'])
    bounds_inter = [-0.5, 0.5, 1.5]
    norm_inter = BoundaryNorm(bounds_inter, cmap_inter.N)
    cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds_inter, cmap=cmap_inter,
norm=norm_inter, alpha=0.2, zorder=2, antialiased=True)
    cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0], colors=['#FFD54F'],
linewidths=2.5, linestyle='solid', zorder=3)
    ax.clabel(cs_alt, inline=True, fontsize=11, fmt='Geo Alt 5°')

```

```

        cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0], colors=['#00E5FF'],
linewidths=2.5, linestyle='dashed', zorder=3)
        ax.clabel(cs_elong, inline=True, fontsize=11, fmt='Geo Eln 8°')

        p_interseksi = mpatches.Patch(color='#00E676', alpha=0.2, label='Area Pemenuhan KHGT
(Geo)')
        line_alt = plt.Line2D([0], [0], color='#FFD54F', lw=2.5, linestyle='solid', label='Batas Altitude 5°
(Geo)')
        line_elong = plt.Line2D([0], [0], color='#00E5FF', lw=2.5, linestyle='dashed', label='Batas
Elongasi 8° (Geo)')
        ax.legend(handles=[p_interseksi, line_alt, line_elong], loc='lower center', ncol=3,
bbox_to_anchor=(0.5, -0.20), fontsize='small')

    else:
        if param_target == "Altitude Bulan":
            target_grid, cmap_name, lbl = grid_alt, 'plasma', "Altitude Bulan (°)"
        elif param_target == "Elongasi":
            target_grid, cmap_name, lbl = grid_elong, 'inferno', "Sudut Elongasi (°)"
        else:
            target_grid, cmap_name, lbl = grid_illum, 'magma', "Fraksi Iluminasi (%)"

        contours = ax.contour(LONS_fine, LATS_fine, target_grid, cmap=cmap_name, levels=20,
linewidths=1.5, alpha=1.0, zorder=3)
        ax.clabel(contours, inline=True, fontsize=12, fmt='%1.1f')
        cbar = fig.colorbar(contours, ax=ax, orientation='horizontal', fraction=0.046, pad=0.1)
        cbar.set_label(f"Legend: {lbl}", fontweight='bold')

        if not is_khgt_map:
            ax.set_title(f"High-Res Visibility Scanner\n{crit_name} - {date_str} - Layer: {param_target}",
fontweight='bold', pad=10)
            ax.set_ylim(-90, 90)
            ax.set_xlabel("Longitude")
            ax.set_ylabel("Latitude")
            ax.set_xticks(np.arange(-180, 181, 30))
            ax.set_yticks(np.arange(-60, 61, 20))
            ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=4)

    fig.tight_layout()

    # Embed ke Tkinter
    frame_canvas = ctk.CTkFrame(win_plot)
    frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)

    canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
    canvas_plot.draw()
    canvas_plot.get_tk_widget().pack(fill="both", expand=True)

    toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")

```

```

toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))

toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
toolbar.update()

def simpan_gambar_kustom():
    filepath = filedialog.asksaveasfilename(
        initialfile=f"HD_VisMap_KHGT_Seamless.png" if is_khgt_map else
f"HD_VisMap_{param_target.replace(' ', '')}.png",
        defaultextension=".png",
        filetypes=[("PNG Image", "*.png"), ("SVG Vector", "*.svg")]
    )
    if filepath:
        fig.savefig(filepath, dpi=300, bbox_inches='tight')
        messagebox.showinfo("Sukses", f"Peta HD berhasil diekspor ke:\n{filepath}")

    btn_export = ctk.CTkButton(toolbar_frame, text="📄 Export Gambar HD", font=("Segoe UI", 12,
"bold"), fg_color="#E65100", hover_color="#BF360C", command=simpan_gambar_kustom)
    btn_export.pack(side="right", padx=10)

    self.lbl_status.configure(text=f"Render Peta HD Selesai.", text_color="#00E676")
    self.btn_hitung.configure(state="normal")

    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", f"[{datetime.datetime.now().strftime('%H:%M:%S')}] Pemrosesan Data
Selesai.\nPeta Visualisasi Kualitas Tinggi (HD) sedang ditampilkan pada Window baru...\n\nLayer
Heatmap : {param_target}")
    self.textbox.configure(state="disabled")

except Exception as e:
    import traceback
    self.display_error(f"Gagal menampilkan HD Peta: {e}\n\n{traceback.format_exc()}")

def calculate_qiblatime(self):
    try:
        # 1. Ambil Input (Aman untuk tahun minus)
        year = int(self.entry_qtyear.get())
        month = int(self.entry_qtmonth.get())
        day = int(self.entry_qtday.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_qtlat.get())
        lon = float(self.entry_qtlon.get())
        tz = float(self.entry_qttz.get())

```

```

# 2. Hitung Azimut Kiblat (Rumus Spherical Trigonometry)
phi_k, lam_k = math.radians(21.4225), math.radians(39.8262) # Ka'bah
phi, lam = math.radians(lat), math.radians(lon)

y_q = math.sin(lam_k - lam)
x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
q_az = math.degrees(math.atan2(y_q, x_q)) % 360
shadow_az = (q_az + 180) % 360

earth, sun = self.eph['earth'], self.eph['sun']
loc = wgs84.latlon(lat, lon)

# 3. Tentukan Rentang Pencarian (00:00 s.d 24:00 waktu lokal dalam UTC)
t0 = self.ts.utc(year, month, day, -int(tz))
t1 = self.ts.utc(year, month, day, 24 - int(tz))

# Gunakan resolusi tinggi (setiap menit) untuk mencari persilangan azimut
tt_array = np.linspace(t0.tt, t1.tt, 1440)
t_search = self.ts.tt_jd(tt_array)

obs = (earth + loc).at(t_search).observe(sun).apparent()
alt_arr, az_arr, _ = obs.altaz()
az_deg_list = az_arr.degrees
alt_deg_list = alt_arr.degrees

def find_crossing_times(target_az):
    # Hitung selisih sudut (handling 360 wrap around)
    diffs = (az_deg_list - target_az + 180) % 360 - 180
    results = []
    for i in range(len(diffs)-1):
        # Jika ada perubahan tanda, berarti ada persilangan
        if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
            # Pastikan matahari di atas ufuk saat itu terjadi
            if alt_deg_list[i] > 0:
                frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
                tt_res = tt_array[i] + frac * (tt_array[i+1] - tt_array[i])

                # Konversi ke waktu lokal
                t_final = self.ts.tt_jd(tt_res + (tz / 24.0))
                _, _, h, mn, s = t_final.utc
                results.append(f"{int(h):02d}:{int(mn):02d}:{int(s):02d}")
    return results

res_sun = ", ".join(find_crossing_times(q_az)) if find_crossing_times(q_az) else "----"
res_shd = ", ".join(find_crossing_times(shadow_az)) if find_crossing_times(shadow_az) else "----"

# Format Tanggal untuk Laporan (Anti Crash)
tgl_str = f"{day:02d}/{month:02d}/{format_tahun_aman(year)}"

```

```

# ---> AMBIL NAMA KOTA & PROVINSI <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Lat {lat}, Lon {lon}, TZ {tz:.2f})"
except:
    lokasi_text = f"CUSTOM LOCATION (Lat {lat}, Lon {lon}, TZ {tz:.2f})"

report = f""{self.get_header(85)}
{ "[ Rashdul Qiblah Lokal / Qibla Time ]".center(85)}

* Settings:-
- Tanggal      : {tgl_str}
- LOKASI       : {lokasi_text}
- Azimut Kiblat : {q_az:.2f}° (dari Utara ke Timur)
- Azimut Bayangan : {shadow_az:.2f}°
=====

1. WAKTU MATAHARI DI ARAH KIBLAT:
(Saat matahari terlihat tepat di arah Ka'bah)
>> {res_sun} LT

2. WAKTU BAYANGAN DI ARAH KIBLAT:
(Saat bayangan benda tegak lurus mengarah ke Ka'bah)
>> {res_shd} LT

* Catatan:
- "----" berarti fenomena tidak terjadi di lokasi pada tanggal tersebut.
- Rashdul Qiblah hanya terjadi saat matahari berada di atas ufuk (>0°).
- Kalkulasi berbasis JPL Ephemeris NASA ({self.ephemeris_name}).
=====
"""

    self.after(0, self.display_result, report)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def _generate_demo_calendar_portrait(self):
    """Merender kanvas Kalender Info Rilis lewat Pillow dengan format Portrait HD persis seperti UI"""
    import os
    import math
    import datetime
    import ephemeris
    import pytz
    from skyfield import almanac

```



```

from PIL import Image, ImageDraw, ImageFont

# Ukuran kanvas portrait resolusi tinggi
img = Image.new("RGB", (1400, 2000), "#0A0A10")
draw = ImageDraw.Draw(img)

try:
    f_title = ImageFont.truetype("arialbd.ttf", 50)
    f_sub = ImageFont.truetype("arial.ttf", 30)
    f_day = ImageFont.truetype("arialbd.ttf", 26)

    # Font khusus untuk data di dalam cell & footer
    f_h_large = ImageFont.truetype("arialbd.ttf", 46) # Angka Hijriah
    f_astro = ImageFont.truetype("courbd.ttf", 15) # Data Astronomi
    f_greg = ImageFont.truetype("arialbd.ttf", 18) # Angka Masehi Bawah
    f_leg = ImageFont.truetype("arial.ttf", 24) # Legenda

    # ---> FONT KHUSUS KETERANGAN & LOKASI AGAR PAS DI KANVAS <---
    f_ket = ImageFont.truetype("arial.ttf", 20) # Keterangan Parameter
    f_loc = ImageFont.truetype("arialbd.ttf", 26) # Nama Kota & Koordinat
except Exception:
    f_title = f_sub = f_day = f_h_large = f_astro = f_greg = f_leg = f_ket = f_loc =
ImageFont.load_default()

# 1. Ambil Data Tanggal & Lokasi
y = self.demo_h_year
m = self.demo_h_month
m_data = HIJRI_DB[y][m]
nama_bulan, _, start_date_str, jumlah_hari = m_data

bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
parts = start_date_str.split('-')
start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

try:
    lat_val = float(self.entry_vlat.get())
    lon_val = float(self.entry_vlon.get())
    elev_val = float(self.entry_velev.get())
    tz_val = float(self.entry_vtz.get())
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
except:
    lat_val, lon_val, elev_val, tz_val = -7.0667, 110.4100, 230.0, 7.0
    nama_kota, nama_prov = "Semarang", "Jawa Tengah"

# 2. Setup Engine Ephemeris & Skyfield

```

```

obs_ephem = ephem.Observer()
obs_ephem.lat, obs_ephem.lon = str(lat_val), str(lon_val)
obs_ephem.elevation, obs_ephem.pressure, obs_ephem.temp = elev_val, 1010, 25
matahari_ephem = ephem.Sun()

earth_sf, sun_sf, moon_sf = self.eph['earth'], self.eph['sun'], self.eph['moon']

# 3. Cetak Header
hijri_title = f"Kalender Hijriah KHGT: {nama_bulan.upper()} {y} H"
greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
if start_date.month != end_date.month:
    greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"

draw.text((700, 80), hijri_title, font=f_title, fill="#FFD54F", anchor="mm")
draw.text((700, 140), greg_title, font=f_sub, fill="#00E5FF", anchor="mm")

# 4. Konfigurasi Grid Portrait
start_x = 85
start_y = 230
cell_w = 175
cell_h = 190
days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]

for i, d_name in enumerate(days_header):
    x0 = start_x + i * cell_w
    y0 = start_y
    x1 = x0 + cell_w - 5
    y1 = y0 + 55
    color_bg = "#FF5252" if i == 0 else ("#00E676" if i == 5 else "#1E88E5")
    draw.rectangle([x0, y0, x1, y1], fill=color_bg, outline=color_bg, width=2)
    draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")

offset = (start_date.weekday() + 1) % 7
row, col = 0, offset
curr_g = start_date
grid_y_start = start_y + 65

# 5. Looping Hari & Kalkulasi Astronomi
for h_day in range(1, jumlah_hari + 1):
    x0 = start_x + col * cell_w
    y0 = grid_y_start + row * cell_h
    x1 = x0 + cell_w - 5
    y1 = y0 + cell_h - 5

    is_today = (curr_g == datetime.date.today())
    if is_today:
        b_color, b_width = "#FFEA00", 4
    else:

```

```

b_color = "#FF5252" if col == 0 else ("#00E676" if col == 5 else "#555555")
b_width = 2 if (col == 0 or col == 5) else 1

draw.rectangle([x0, y0, x1, y1], fill="#0F172A", outline=b_color, width=b_width)

# --- KALKULASI DATA ASTRONOMI ---
dt_noon_local = datetime.datetime(curr_g.year, curr_g.month, curr_g.day, 12, 0, 0)
dt_noon_utc = dt_noon_local - datetime.timedelta(hours=tz_val)
obs_ephem.date = ephem.Date(dt_noon_utc.strftime("%Y/%m/%d %H:%M:%S"))

try:
    waktu_sunset_ephem = obs_ephem.next_setting(matahari_ephem)
except:
    waktu_sunset_ephem = obs_ephem.date + 0.25

dt_sunset_utc = waktu_sunset_ephem.datetime().replace(tzinfo=pytz.utc)
t_sunset = self.ts.from_datetime(dt_sunset_utc)

geo_earth = earth_sf.at(t_sunset)
app_moon_geo = geo_earth.observe(moon_sf).apparent()
app_sun_geo = geo_earth.observe(sun_sf).apparent()

sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
elongasi = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg

ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)
gast = t_sunset.gast
lst_deg = (gast * 15.0) + lon_val

ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
ha_rad = math.radians(lst_deg - (ra_h * 15.0))
lat_rad = math.radians(lat_val)

sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)
tinggi_hilal = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

try:
    prev_nm = ephem.previous_new_moon(waktu_sunset_ephem)
    umur_bulan = waktu_sunset_ephem - prev_nm
except:
    umur_bulan = 0.0

illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
iluminasi = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0

# Format teks astronomi

```

```

umur_str = f"{umur_bulan:.1f}" if type(umur_bulan) == float else umur_bulan
elong_str = f"{elongasi:.1f}°"
ilu_str = f"{iluminasi:.1f}%"
alt_str = f"{tinggi_hilal:.1f}°"

# --- MENGGAMBAR KE DALAM KOTAK (CELL) ---
# 1. Angka Hijriah & Kiri Atas
draw.text((x0 + 10, y0 + 10), str(h_day), font=f_h_large, fill="#FFFFFF", anchor="lt")
draw.text((x0 + 10, y0 + 65), umur_str, font=f_astro, fill="#FFFFFF", anchor="lt")
draw.text((x0 + 10, y0 + 85), elong_str, font=f_astro, fill="#FFFFFF", anchor="lt")

# 2. Gambar Bulan & Kanan Atas
moon_size = 45
angle = (h_day / 29.53059) * 360.0
moon_idx = int(angle) % 360
moon_path = os.path.join(BASE_DIR, "moon", f"m{moon_idx:03d}.png")

if os.path.exists(moon_path):
    img_moon = Image.open(moon_path).convert("RGBA")
    img_moon = img_moon.resize((moon_size, moon_size), Image.Resampling.LANCZOS)
    mask = Image.new('L', (moon_size, moon_size), 0)
    draw_mask = ImageDraw.Draw(mask)
    draw_mask.ellipse((0, 0, moon_size - 1, moon_size - 1), fill=255)
    img_moon.putalpha(mask)
    # Tempelkan gambar bulan di pojok kanan atas kotak
    img.paste(img_moon, (int(x1 - moon_size - 8), int(y0 + 8)), img_moon)
else:
    draw.text((x1 - 10, y0 + 10), " 🌙 ", font=f_day, fill="#FFFFFF", anchor="rt")

draw.text((x1 - 10, y0 + 65), ilu_str, font=f_astro, fill="#FFFFFF", anchor="rt")
draw.text((x1 - 10, y0 + 85), alt_str, font=f_astro, fill="#FFFFFF", anchor="rt")

# 3. Tanggal Masehi & Indikator Event di Bawah
m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3].upper()} {curr_g.year}"
draw.text(((x0+x1)/2, y1 - 25), m_str, font=f_greg, fill="#FFD54F", anchor="mm")

event_color, _ = self._get_islamic_event(h_day, m + 1, curr_g.weekday())
if event_color:
    dot_r = 7
    dot_cx, dot_cy = (x0+x1)/2, y1 - 45
    draw.ellipse([dot_cx - dot_r, dot_cy - dot_r, dot_cx + dot_r, dot_cy + dot_r], fill=event_color)

col += 1
if col > 6:
    col = 0
    row += 1
curr_g += datetime.timedelta(days=1)

```

```

# =====
# 6. MENGGAMBAR LEGENDA & KETERANGAN SECARA DINAMIS (ANTI TERPOTONG)
# =====

# Posisi Y dinamis diatur berdasarkan jumlah baris di kalender saat ini
legend_y = grid_y_start + (row + 1 if col > 0 else row) * cell_h + 30

draw.text((start_x, legend_y), "LEGENDA WARNA:", font=f_sub, fill="#FFD54F", anchor="lm")

legend_items = [
    ("#FF5252", "Hari Haram Puasa / Hari Raya"),
    ("#00E676", "Puasa Sunnah Utama"),
    ("#00B0FF", "Puasa Ayyamul Bidh"),
    ("#FFA000", "Puasa Senin Kamis"),
    ("#8BC34A", "Ramadhan"),
    ("#E040FB", "Hari Besar Islam")
]

ly = legend_y + 40
lx = start_x
for i, (c, text) in enumerate(legend_items):
    draw.ellipse([lx, ly-10, lx+20, ly+10], fill=c)
    draw.text((lx+35, ly), text, font=f_leg, fill="white", anchor="lm")
    ly += 40
    if i == 2:
        ly = legend_y + 40
        lx = start_x + 450

# ---> BAGIAN KETERANGAN DAN LOKASI DICETAK LENGKAP <---
# Kita tempatkan elemen footer menyesuaikan akhir dari letak legenda (dinamis)
footer_y = legend_y + 175

# 1. Teks Keterangan Parameter (Teks Lengkap)
teks_ket = "* Kiri: Umur Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas), Tinggi Hilal Geo (Bawah). Dihitung saat Maghrib."
draw.text((700, footer_y), teks_ket, font=f_ket, fill="FFFFFF", anchor="mm")

# 2. Teks Info Lokasi (Kota dan Koordinat)
ket_lokasi = f"📍 Kota: {nama_kota}, {nama_prov} | Koordinat: (Lat: {lat_val}, Lon: {lon_val})"
draw.text((700, footer_y + 45), ket_lokasi, font=f_loc, fill="#00E5FF", anchor="mm")

# 3. Copyright / Watermark
draw.text((700, footer_y + 105), "KHGT Times Engine V8.0 - By Kasmui", font=f_leg, fill="#555555", anchor="mm")

```

```

return img

def export_demo_kalender(self, format_type):
    """Fungsi pembantu untuk trigger simpan PDF atau PNG khusus Kalender Rilis"""
    img = self._generate_demo_calendar_portrait()

    y = self.demo_h_year
    m = self.demo_h_month
    nama_bulan = HIJRI_DB[y][m][0]

    default_file = f"Kalender_KHGT_{nama_bulan}_{y}H_Portrait"

    if format_type == "png":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG Portrait
di:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")

        elif format_type == "pdf":
            filepath = filedialog.asksaveasfilename(
                initialfile=f"{default_file}.pdf",
                defaultextension=".pdf",
                filetypes=[("PDF Document", "*.pdf")]
            )
            if filepath:
                try:
                    img.save(filepath, "PDF", resolution=150.0)
                    messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi PDF Portrait
di:\n{filepath}")
                except Exception as e:
                    messagebox.showerror("Error", f"Gagal menyimpan PDF: {e}")

    def calculate_first_point(self):
        try:
            # 1. Ambil input sebagai integer (Aman untuk tahun minus)
            y = int(self.entry_fp_y.get())
            m = int(self.entry_fp_m.get())
            d = int(self.entry_fp_d.get())

            # --- PERBAIKAN: Panggil auto_switch_ephemeris agar tidak crash/macet ---

```

```

self.auto_switch_ephemeris(y)

# 2. Bypass string, gunakan Tuple PyEphem!
waktu_tuple = (y, m, d, 12, 0, 0)

# Update UI via main thread
self.after(0, lambda: self.lbl_status.configure(text="Melacak Titik Pertama (Iterasi Multi-Hari)...",
text_color="#00E5FF"))

matahari = ephem.Sun()
bulan = ephem.Moon()

try:
    ijtimak_1 = ephem.previous_new_moon(waktu_tuple)
    ijtimak_2 = ephem.next_new_moon(waktu_tuple)
except Exception:
    self.after(0, self.display_error, "Format tanggal tidak valid. Harap periksa input.")
    return

titik_hasil = []
# Kita abaikan pulau kecil / wilayah timur jauh yang bukan daratan utama (Mainland)
zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "Maluku", "Maluku
Utara", "NTT"]

# Loop untuk 2 siklus ijtimak
for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "Bulan Berjalan/Depan")]:

    # Cek dari H+0 (Hari Ijtimak) sampai maksimal H+3
    for offset_hari in range(4):
        kandidat_hari_ini = []
        waktu_pencarian = ephem.Date(w_ijtimak + offset_hari)

        for negara, kota_dict in CITY_DB.items():
            if negara in zona_dikecualikan:
                continue

            for nama_kota, koordinat in kota_dict.items():
                lintang, bujur = koordinat
                pengamat = ephem.Observer()
                pengamat.lat = str(lintang)
                pengamat.lon = str(bujur)
                pengamat.elevation = 0

                pengamat.date = waktu_pencarian
                try:
                    waktu_sunset = pengamat.next_setting(matahari)
                except (ephem.AlwaysUpError, ephem.NeverUpError):
                    continue # Lewati jika di kutub / tidak ada sunset

```

```

pengamat.date = waktu_sunset
matahari.compute(pengamat)
bulan.compute(pengamat)

# Kalkulasi Geo (DISELELARASKAN DENGAN MENU 2 - SKYFIELD)
import pytz
t_sunset_sky = self.ts.from_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))
obs_center = self.eph['earth'].at(t_sunset_sky)
s_app_sky = obs_center.observe(self.eph['sun']).apparent()
m_app_sky = obs_center.observe(self.eph['moon']).apparent()

elong_geo = s_app_sky.separation_from(m_app_sky).degrees

gmst = t_sunset_sky.gast
lst_deg = (gmst * 15.0) + bujur
ra_m, dec_m, _ = m_app_sky.radec(epoch=t_sunset_sky)

ha_deg = lst_deg - (ra_m.hours * 15.0)
ha_rad = math.radians(ha_deg)
lat_rad = math.radians(lintang)
d_rad = dec_m.radians

sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad)
* math.cos(ha_rad)
alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

if alt_geo >= 5.0 and elong_geo >= 8.0:
    kandidat_hari_ini.append({
        'kota': nama_kota,
        'negara': negara,
        'lat': lintang,
        'lon': bujur,
        'sunset_utc': waktu_sunset,
        'alt_geo': alt_geo,
        'elong_geo': elong_geo,
        'ijtimak': w_ijtimak,
        'label': label_siklus
    })

# Jika di hari ini sudah ada minimal 1 kota yang memenuhi kriteria,
# Berhenti mencari di hari berikutnya! Ambil yang paling pertama terbenam.
if kandidat_hari_ini:
    kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
    titik_hasil.append(kandidat_hari_ini[0])
    break

```

if not titik_hasil:


```

        self.after(0, self.display_error, "Tidak ada daratan utama yang memenuhi kriteria KHGT pada 2
siklus ini.")
        return

    self.after(0, self.show_first_point_map, titik_hasil)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Gagal menghitung Titik
Pertama:\n{str(e)}\n\nTraceback:\n{traceback.format_exc()}")

def show_first_point_map(self, list_titik):
    try:
        win_plot = ctk.CTkToplevel(self)
        win_plot.title("Peta Titik Pertama Visibilitas KHGT (2 Siklus) & Analisis Gisborne")
        win_plot.geometry("1100x750")
        win_plot.attributes("-topmost", True)

        fig, ax = plt.subplots(figsize=(11, 6.5), dpi=100)

        # Load Peta Latar Belakang
        map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
        map_file = "map_topografi.jpg"
        download_custom_bsp(map_file, map_url)
        full_map_path = os.path.join(BASE_DIR, map_file)

        if os.path.exists(full_map_path):
            try:
                img = Image.open(full_map_path)
                ax.imshow(img, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
                ax.set_ylim(-90, 90)
            except Exception as e:
                print("Gagal memuat gambar peta:", e)

        # Plot semua kota dari database sebagai noktah merah
        all_lats = []
        all_lons = []
        for negara, kota_dict in CITY_DB.items():
            for nama_kota, koordinat in kota_dict.items():
                all_lats.append(koordinat[0])
                all_lons.append(koordinat[1])

        ax.scatter(all_lons, all_lats, color='red', marker='o', s=8, alpha=0.6, zorder=3)

        # Konfigurasi visual marker
        warna_bintang = ['#FFD54F', '#00E676']
        warna_tepi = ['red', 'white']

```

```
text_summary = "Pencarian Multi-Siklus Selesai.\n\nTitik daratan utama pertama di dunia yang
masuk bulan baru KHGT:\n\n"
```

```
for idx, pt in enumerate(list_titik):
    c_star = warna_bintang[idx % len(warna_bintang)]
    c_edge = warna_tepi[idx % len(warna_tepi)]

    # Plot Bintang Titik Pertama
    ax.scatter(pt['lon'], pt['lat'], color=c_star, marker='*', s=600, edgecolor=c_edge, linewidth=1.5,
zorder=5)

    tz_approx = int(self.get_tz_from_lon(pt['lon']))
    tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"

    # KALKULASI WAKTU LOKAL (Bypass Python Datetime, pakai Ephem Math)
    dt_local_ephem = ephemeris.Date(pt['sunset_utc'] + (tz_approx / 24.0))
    y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()

    nama_bulan = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]
    tgl_lokal_str = f"{int(d_l):02d} {nama_bulan[int(m_l)]} {format_tahun_aman(int(y_l))}"
    jam_lokal_str = f"{int(h_l):02d}:{int(min_l):02d}:{int(s_l):02d}"

    y_i, m_i, d_i, h_i, min_i, s_i = pt['ijtimak'].tuple()
    ij_jam = f"{int(h_i):02d}:{int(min_i):02d}:{int(s_i):02d}"

    y_s, m_s, d_s, h_s, min_s, s_s = pt['sunset_utc'].tuple()
    ss_jam = f"{int(h_s):02d}:{int(min_s):02d}:{int(s_s):02d}"

    # =====
    # EVALUASI PKG 1 & PKG 2 (TERINTEGRASI KONVERSI TANGGAL)
    # =====
    # PERBAIKAN: Bungkus operasi penambahan 1 hari ke dalam ephemeris.Date()
    batas_00_utc_ephem = ephemeris.Date(ephemeris.Date(int(y_l), int(m_l), int(d_l), 0, 0, 0) + 1.0)
    bt_jam = "00:00:00"
    fajar_info = ""

    negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil", "Argentina", "Kolombia",
"Peru", "Chili"]
    is_amerika = pt['negara'] in negara_amerika

    import pytz
    dt_ij_utc = pt['ijtimak'].datetime().replace(tzinfo=pytz.utc)
    nm_tt_val = self.ts.from_datetime(dt_ij_utc).tt
    h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
    hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
    hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
```

```

# EVALUASI PKG 1 (Sunset < 00:00 UTC)
if pt['sunset_utc'] < batas_00_utc_ephem:
    tgl_1_mas = ephem.Date(batas_00_utc_ephem)
    gy, gm, gd, _, _ = tgl_1_mas.tuple()
    masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
    status_ijtimak = f"Kriteria PKG 1 Terpenuhi\n(Terpenuhi sbnm {bt_jam} UTC)\n» Kesimpulan:
BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"

# EVALUASI PKG 2 (Jika PKG 1 gagal)
else:
    gisborne = ephem.Observer()
    gisborne.lat = math.radians(-38.6623)
    gisborne.lon = math.radians(178.0176)
    gisborne.elevation = 0
    gisborne.pressure = 0
    gisborne.horizon = '-18' # Fajar (Astronomical Twilight)
    matahari = ephem.Sun()

    gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
    matahari.compute(gisborne)

    try:
        fajar_gisborne_ephem = gisborne.next_rising(matahari, use_center=True)
        _, _, h_f, min_f, s_f = fajar_gisborne_ephem.tuple()
        fj_jam = f"{int(h_f):02d}:{int(min_f):02d}:{int(s_f):02d}"

        fajar_info = f" Fajar Gisb. : {fj_jam} UTC\n"

        if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
            tgl_1_mas = ephem.Date(batas_00_utc_ephem)
            gy, gm, gd, _, _ = tgl_1_mas.tuple()
            masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
            status_ijtimak = f"Kriteria PKG 2 Terpenuhi\n(Titik Amerika & Ijtima < Fajar)\n»
Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
        else:
            tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)
            gy, gm, gd, _, _ = tgl_lusa.tuple()
            masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
            alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima > Terbit Fajar"
            status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi\n({alasan})\n» Kesimpulan: LUSA (1
{hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"

    except Exception as e:
        fj_jam = "N/A"
        fajar_info = f" Fajar Gisb. : Error/N/A\n"

        tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)
        gy, gm, gd, _, _ = tgl_lusa.tuple()

```

```

masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi\n(Gagal Menghitung Fajar)\n»
Kesimpulan: LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"

```

```

info_text = (
    f"TITIK KHGT ({pt['label'].upper()})\n"
    f"{'-'*45}\n"
    f"Lokasi : {pt['kota']}, {pt['negara']}\n"
    f"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
    f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
    f"{'-'*45}\n"
    f"Parameter Waktu (UTC):\n"
    f" Waktu Maghrib: {ss_jam} UTC\n"
    f" Batas PKG 1 : {bt_jam} UTC\n"
    f" Ijtimak : {ij_jam} UTC\n"
    f"{fajar_info}"
    f"{'-'*45}\n"
    f"{status_ijtimak}"
)

```

```

props = dict(boxstyle='round,pad=0.6', facecolor='#1A1A1A', alpha=0.85, edgecolor=c_star,
linewidth=1.5)

```

```

if idx == 0:
    text_pos = (0.02, 0.04)
    halign = 'left'
    valign = 'bottom'
    curve = "arc3,rad=0.1"
else:
    text_pos = (0.98, 0.04)
    halign = 'right'
    valign = 'bottom'
    curve = "arc3,rad=-0.1"

```

```

ax.annotate(
    info_text,
    xy=(pt['lon'], pt['lat']),
    xycoords='data',
    xytext=text_pos,
    textcoords='axes fraction',
    color='white',
    fontsize=9,
    fontfamily='monospace',
    verticalalignment=valign,
    horizontalalignment=halign,
    bbox=props,
    arrowprops=dict(
        arrowstyle="->",

```

```

        connectionstyle=curve,
        color=c_star,
        lw=1.5,
        alpha=0.8
    ),
    zorder=6
)

text_summary += f"[{pt['label']}] >> {pt['kota']}, {pt['negara']} (Maghrib Lokal: {tgl_lokal_str},
{jam_lokal_str})\nStatus: {status_ijtimak.replace(chr(10), ' ')}\n\n"

ax.set_title("Titik Awal Pemenuhan Kriteria KHGT & Komparasi Fajar Gisborne",
fontweight='bold', pad=10)
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
ax.set_xticks(np.arange(-180, 181, 30))
ax.set_yticks(np.arange(-60, 61, 20))
ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=1)

fig.tight_layout()

frame_canvas = ctk.CTkFrame(win_plot)
frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)

canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
canvas_plot.draw()
canvas_plot.get_tk_widget().pack(fill="both", expand=True)

toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
toolbar.update()

self.lbl_status.configure(text=f"Pencarian Titik Pertama & Analisis Selesai",
text_color="#00E676")
self.btn_hitung.configure(state="normal")

self.textbox.configure(state="normal")
self.textbox.delete("1.0", "end")
text_summary += "Detail visual komparasi telah ditambahkan pada peta (Text Box melayang)."
self.textbox.insert("1.0", text_summary)
self.textbox.configure(state="disabled")

except Exception as e:
    import traceback
    self.display_error(f"Gagal menampilkan peta Titik Pertama: {e}\n\n{traceback.format_exc()}")
    self.textbox.insert("1.0", text_summary)
    self.textbox.configure(state="disabled")

```

```

def generate_first_point_report(self, y, m, d):
    waktu_tuple = (y, m, d, 12, 0, 0)
    self.auto_switch_ephemeris(y)

    matahari = ephemeris.Sun()
    bulan = ephemeris.Moon()

    try:
        ijtimak_1 = ephemeris.previous_new_moon(waktu_tuple)
        ijtimak_2 = ephemeris.next_new_moon(waktu_tuple)
    except Exception:
        return "Format tanggal tidak valid. Harap periksa input."

    titik_hasil = []
    zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "Maluku", "Maluku Utara", "NTT"]

    for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "Bulan Berjalan/Depan")]:
        for offset_hari in range(4):
            kandidat_hari_ini = []
            waktu_pencarian = ephemeris.Date(w_ijtimak + offset_hari)

            for negara, kota_dict in CITY_DB.items():
                if negara in zona_dikecualikan: continue
                for nama_kota, koordinat in kota_dict.items():
                    lintang, bujur = koordinat
                    pengamat = ephemeris.Observer()
                    pengamat.lat, pengamat.lon = str(lintang), str(bujur)
                    pengamat.elevation = 0
                    pengamat.pressure = 1010
                    pengamat.temp = 25
                    pengamat.date = waktu_pencarian

                    try: waktu_sunset = pengamat.next_setting(matahari)
                    except: continue

                    pengamat.date = waktu_sunset
                    matahari.compute(pengamat)
                    bulan.compute(pengamat)

            import pytz
            t_sunset_sky = self.ts.from_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))
            obs_center = self.eph['earth'].at(t_sunset_sky)
            s_app_sky = obs_center.observe(self.eph['sun']).apparent()
            m_app_sky = obs_center.observe(self.eph['moon']).apparent()

```

```

elong_geo = s_app_sky.separation_from(m_app_sky).degrees

gmst = t_sunset_sky.gast
lst_deg = (gmst * 15.0) + bujur
ra_m, dec_m, _ = m_app_sky.radec(epoch=t_sunset_sky)

ha_deg = lst_deg - (ra_m.hours * 15.0)
ha_rad = math.radians(ha_deg)
lat_rad = math.radians(lintang)
d_rad = dec_m.radians

sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

if alt_geo >= 5.0 and elong_geo >= 8.0:
    kandidat_hari_ini.append({
        'kota': nama_kota, 'negara': negara, 'lat': lintang, 'lon': bujur,
        'sunset_utc': waktu_sunset, 'alt_geo': alt_geo, 'elong_geo': elong_geo,
        'ijtimak': w_ijtimak, 'label': label_siklus
    })

if kandidat_hari_ini:
    kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
    titik_hasil.append(kandidat_hari_ini[0])
    break

if not titik_hasil:
    return "Tidak ada daratan utama (Mainland) yang memenuhi kriteria KHGT pada 2 siklus ini."

report = "Pencarian Multi-Siklus Selesai.\nTitik daratan utama pertama di dunia yang masuk bulan
baru KHGT:\n\n"

for pt in titik_hasil:
    tz_approx = int(self.get_tz_from_lon(pt['lon']))
    tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"

    dt_local_ephem = ephemeris.Date(pt['sunset_utc'] + (tz_approx / 24.0))
    y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()

    nama_bulan_masehi = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt",
"Nov", "Des"]
    tgl_lokal_str = f"{int(d_l):02d} {nama_bulan_masehi[int(m_l)]} {format_tahun_aman(int(y_l))}"
    jam_lokal_str = f"{int(h_l):02d}:{int(min_l):02d}:{int(s_l):02d}"

    y_i, m_i, d_i, h_i, min_i, s_i = pt['ijtimak'].tuple()
    ij_jam = f"{int(h_i):02d}:{int(min_i):02d}:{int(s_i):02d}"

```

```

y_s, m_s, d_s, h_s, min_s, s_s = pt['sunset_utc'].tuple()
ss_jam = f"{int(h_s):02d}:{int(min_s):02d}:{int(s_s):02d}"

# PERBAIKAN: Bungkus dengan ephem.Date()
batas_00_utc_ephem = ephem.Date(ephem.Date((int(y_l), int(m_l), int(d_l), 0, 0, 0)) + 1.0)
bt_jam = "00:00:00"
fajar_info = ""

negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil", "Argentina", "Kolombia",
"Peru", "Chili"]
is_amerika = pt['negara'] in negara_amerika

import pytz
dt_ij_utc = pt['ijtimak'].datetime().replace(tzinfo=pytz.utc)
nm_tt_val = self.ts.from_datetime(dt_ij_utc).tt
h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"

if pt['sunset_utc'] < batas_00_utc_ephem:
    tgl_1_mas = ephem.Date(batas_00_utc_ephem)
    gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
    masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
    status_ijtimak = f"Kriteria PKG 1 Terpenuhi (Terpenuhi sbml {bt_jam} UTC)\n» Kesimpulan:
BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
else:
    gisborne = ephem.Observer()
    gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
    gisborne.elevation, gisborne.pressure = 0, 0
    gisborne.horizon = '-18'
    matahari_g = ephem.Sun()
    gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
    matahari_g.compute(gisborne)

    try:
        fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)
        _, _, h_f, min_f, s_f = fajar_gisborne_ephem.tuple()
        fj_jam = f"{int(h_f):02d}:{int(min_f):02d}:{int(s_f):02d}"
        fajar_info = f" Fajar Gisb. : {fj_jam} UTC\n"

    if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
        tgl_1_mas = ephem.Date(batas_00_utc_ephem)
        gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
        masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
        status_ijtimak = f"Kriteria PKG 2 Terpenuhi (Titik Amerika & Ijtima < Fajar Gisb.)\n»
Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
    else:
        tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)

```



```

        gy, gm, gd, _, _, _ = tgl_lusa.tuple()
        masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
        alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima > Terbit Fajar"
        status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi ({alasan})\n» Kesimpulan: LUSA (1
{hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
    except:
        tgl_lusa = ephemer.Date(batas_00_utc_ephem + 1.0)
        gy, gm, gd, _, _, _ = tgl_lusa.tuple()
        masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
        fajar_info = " Fajar Gisb. : Error/N/A\n"
        status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi (Gagal Menghitung Fajar)\n» Kesimpulan:
LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"

    report += (
        f"[{pt['label'].upper()}]\n"
        f"{'-'*45}\n"
        f"Lokasi : {pt['kota']}, {pt['negara']}\n"
        f"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
        f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
        f"{'-'*45}\n"
        f"Parameter Waktu (UTC):\n"
        f" Waktu Maghrib: {ss_jam} UTC\n"
        f" Batas PKG 1 : {bt_jam} UTC\n"
        f" Ijtimak : {ij_jam} UTC\n"
        f"{fajar_info}"
        f"{'-'*45}\n"
        f"Status: {status_ijtimak}\n"
        f"{'-'*45}\n\n"
    )
    return report.strip()

def calculate_seasons(self):
    try:
        year = int(self.entry_season_year.get())
        self.auto_switch_ephemeris(year)

        t0 = self.ts.utc(year, 1, 1)
        t1 = self.ts.utc(year, 12, 31)
        t_seasons, y_seasons = almanac.find_discrete(t0, t1, almanac.seasons(self.eph))

        tz_wib = pytz.timezone('Asia/Jakarta')
        musim_nama = ["Vernal Equinox (Musim Semi Utara)",
            "Summer Solstice (Musim Panas Utara)",
            "Autumnal Equinox (Musim Gugur Utara)",
            "Winter Solstice (Musim Dingin Utara)"]

        report = f"{self.get_header(85)}\n"
        report += f"[ Equinox & Solstice Calculator ].center(85)}\n\n"

```

```

report += f"* Tahun Astronomis: {year} CE\n"
report += f"* Zona Waktu: WIB (UTC+7)\n"
report += "="*85 + "\n\n"

for t_ev, y_ev in zip(t_seasons, y_seasons):
    dt_wib = t_ev.utctime().replace(tzinfo=pytz.utc).astimezone(tz_wib)
    waktu_str = dt_wib.strftime('%d %B %Y - %H:%M:%S WIB')
    nama = musim_nama[y_ev]
    report += f">> {nama:<38} : {waktu_str}\n"

report += "\n" + "="*85
self.after(0, self.display_result, report)
except Exception as e:
    self.after(0, self.display_error, str(e))

def calculate_planetary_times(self):
    try:
        year = int(self.entry_pl_year.get())
        month = int(self.entry_pl_month.get())
        target_name = self.combo_pl_target.get().lower()

        # Sinkronisasi Ephemeris sesuai rentang tahun
        self.auto_switch_ephemeris(year)

        # Khusus planet, JPL Ephemeris biasanya menggunakan suffix 'barycenter'
        target_key = f'{target_name} barycenter' if target_name != 'moon' else 'moon'
        target_obj = self.eph[target_key]
        earth = self.eph['earth']

        # Ambil koordinat lokasi
        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        tz = float(self.entry_vtz.get())
        elev = float(self.entry_velev.get())
        loc = wgs84.latlon(lat, lon, elevation_m=elev)

        # Gunakan fungsi safe_monthrange agar kebal tahun minus/BCE
        _, num_days = safe_monthrange(year, month)

        # Tentukan rentang pencarian (00:00 hari pertama s.d 24:00 hari terakhir)
        t0 = self.ts.utc(year, month, 1, -int(tz))
        t1 = self.ts.utc(year, month, num_days, 24 - int(tz))

        # Hitung kejadian Rise/Set
        f_rs = almanac.risings_and_settings(self.eph, target_obj, loc)
        t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

        # ---> PERBAIKAN: MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR <---

```

```

try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov} (Lat {lat}, Lon {lon}, TZ UTC{'+' if tz >= 0 else ''}{tz})"
except:
    lokasi_text = f"Lat {lat}, Lon {lon}, TZ UTC{'+' if tz >= 0 else ''}{tz}"

# Inisialisasi Header Laporan
report = f"{self.get_header(85)}\n"
report += f"[ Planetary Times: {target_name.upper()} ].center(85)}\n\n"
report += f"Bulan/Tahun : {month:02d}/{format_tahun_aman(year)}\n"
report += f"Lokasi : {lokasi_text}\n"
report += "="*85 + "\n"
report += "Tanggal          Terbit (Rise)          Terbenam (Set)\n"
report += "-"*85 + "\n"

# Loop setiap hari untuk mencocokkan event
for d in range(1, num_days + 1):
    found_rise = "----"
    found_set = "----"

    if t_rs is not None:
        for t_val, y_val in get_safe_events(t_rs, y_rs):
            # Konversi waktu kejadian ke waktu lokal untuk pengecekan tanggal
            t_loc = self.ts.tt_jd(t_val.tt + (tz / 24.0))
            _, _, dl, hl, mnl, _ = t_loc.utc

            if int(dl) == d:
                time_str = f"{int(hl):02d}:{int(mnl):02d}"
                if y_val == 1: found_rise = time_str
                else: found_set = time_str

# Format tampilan tanggal (Bebas crash tahun minus)
date_display = f"{d:02d}/{month:02d}/{format_tahun_aman(year)}"

# Tambahkan baris data ke laporan
line = f"{date_display:<18} {found_rise:^15} {found_set:^15}\n"
report += line

report += "="*85
report += "\n* Remarks: '----' means the event does not occur on that day.\n"
report += f"* Calculated using JPL NASA Engine ({self.ephemeris_name})."

# Tampilkan ke ruang kanan (Textbox)
self.after(0, self.display_result, report)

except Exception as e:
    import traceback

```

```

self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def display_result(self, report_text):
    # Simpan laporan mentah berbahasa Indonesia agar ruang kanan bisa
    # digambar ulang 100% ketika bahasa diganti ke ID/EN/AR.
    self._last_raw_report = report_text
    self._last_raw_textbox_content = report_text

    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", report_text)
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text=self.tr("Kalkulasi Selesai"), text_color="#00E676")
    self.btn_hitung.configure(state="normal", text=self.tr("▶ PROSES DATA"))

def display_error(self, error_msg):
    # Error juga disimpan mentah agar saat EN/AR dipilih teks debug
    # ikut berganti bahasa dan tidak memicu translasi ganda.
    self._last_raw_report = f"TERJADI KESALAHAN (DEBUG):\n{error_msg}"
    self._last_raw_textbox_content = self._last_raw_report

    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", self._last_raw_report)
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text=self.tr("Error Kalkulasi"), text_color="#FF1744")
    self.btn_hitung.configure(state="normal", text=self.tr("▶ PROSES DATA"))

def auto_detect_location(self):
    self.lbl_status.configure(text="Mendeteksi lokasi...", text_color="#00E5FF")
    def fetch_location():
        try:
            import json
            import urllib.request
            req = urllib.request.Request("http://ip-api.com/json/", headers={'User-Agent': 'Mozilla/5.0'})
            with urllib.request.urlopen(req, timeout=5) as response:
                data = json.loads(response.read().decode())
            if data['status'] == 'success':
                lat, lon, city = data['lat'], data['lon'], data['city']
                self.after(0, self._update_all_loc_entries, lat, lon, city)
            else:
                self.after(0, lambda: messagebox.showerror("Gagal", "Tidak dapat mendeteksi lokasi."))
        except Exception as e:
            self.after(0, lambda: messagebox.showerror("Error", f"Koneksi Geolocation Gagal: {e}"))
    finally:
        self.after(0, lambda: self.lbl_status.configure(text="Sistem Siap", text_color="#00E676"))
    threading.Thread(target=fetch_location, daemon=True).start()

```

```

def _update_all_loc_entries(self, lat, lon, city):
    lat_entries = ['entry_vlat', 'entry_eph_lat', 'entry_qlat', 'entry_mt_lat', 'entry_st_lat', 'entry_pt_lat',
'entry_qtlat', 'entry_eph3d_lat', 'entry_miz_lat', 'entry_kb_lat', 'entry_astro_lat', 'entry_pasang_lat',
'entry_fajar_lat', 'entry_analemma_lat']
    lon_entries = ['entry_vlon', 'entry_eph_lon', 'entry_qlon', 'entry_mt_lon', 'entry_st_lon',
'entry_pt_lon', 'entry_qtlon', 'entry_eph3d_lon', 'entry_miz_lon', 'entry_kb_lon', 'entry_astro_lon',
'entry_pasang_lon', 'entry_fajar_lon', 'entry_analemma_lon']
    tz_entries = ['entry_vtz', 'entry_eph_tz', 'entry_qtz', 'entry_mt_tz', 'entry_st_tz', 'entry_pt_tz',
'entry_qttz', 'entry_miz_tz', 'entry_kb_tz', 'entry_astro_tz', 'entry_pasang_tz', 'entry_fajar_tz',
'entry_analemma_tz']

    tz_val = self.get_tz_from_lon(lon)
    for name in lat_entries:
        if hasattr(self, name):
            e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(lat))
    for name in lon_entries:
        if hasattr(self, name):
            e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(lon))
    for name in tz_entries:
        if hasattr(self, name):
            e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(tz_val))

self.lbl_status.configure(text=f"📍 Lokasi Terdeteksi: {city} ({lat}, {lon})", text_color="green")

# ---> TAMBAHKAN BARIS INI DI AKHIR FUNGSI _update_all_loc_entries <---
if hasattr(self, 'fetch_and_update_weather'):
    labels_cuaca = [
        'lbl_status_cuaca', 'lbl_cuaca_astro',
        'lbl_cuaca_pasang', 'lbl_cuaca_fajar', 'lbl_cuaca_analemma'
    ]
    for lbl_name in labels_cuaca:
        if hasattr(self, lbl_name):
            self.fetch_and_update_weather(lat, lon, getattr(self, lbl_name))
# -----

# 2. Cari Provinsi dari CITY_DB berdasarkan nama kota hasil deteksi API
ditemukan_prov = None
ditemukan_kota = city

for prov_name, cities_dict in CITY_DB.items():
    for db_city in cities_dict.keys():
        # Pencarian fleksibel
        if city.lower() in db_city.lower() or db_city.lower() in city.lower():
            ditemukan_prov = prov_name
            ditemukan_kota = db_city
            break

```

```

        if ditemukan_prov:
            break

# 3. Sesuaikan Dropdown Provinsi dan Kota di Sidebar
if ditemukan_prov:
    self.opt_prov.set(ditemukan_prov)
    cities_in_prov = sorted(list(CITY_DB[ditemukan_prov].keys()))
    self.opt_city.configure(values=cities_in_prov)
    self.opt_city.set(ditemukan_kota)
else:
    # Jika kota dari GPS TIDAK ada di database bawaan, buat menu kustom
    prov_values = list(self.opt_prov.cget("values"))
    if "Lokasi Otomatis (GPS)" not in prov_values:
        self.opt_prov.configure(values=["Lokasi Otomatis (GPS)"] + prov_values)
    self.opt_prov.set("Lokasi Otomatis (GPS)")

    city_values = list(self.opt_city.cget("values"))
    if city not in city_values:
        self.opt_city.configure(values=[city] + city_values)
    self.opt_city.set(city)

# 4. Update Label atas dan munculkan pop-up sukses
self.lokasi_nama.set(f"{{ditemukan_kota}} (Auto-Detected)")
messagebox.showinfo("Lokasi Ditemukan", f"Sistem GPS mendeteksi lokasi Anda di
{{ditemukan_kota}}\nLat: {{lat}}\nLon: {{lon}}\n\nSeluruh modul telah diperbarui ke koordinat ini.")

def export_to_ics(self):
    # Cek apakah jadwal salat sudah ada di memori
    if not self.daily_prayer_schedule:
        messagebox.showwarning("Peringatan", "Jadwal Salat belum tersedia di memori!\nSilakan
aktifkan 'Alarm Salat' di sidebar agar sistem menghitung jadwal hari ini terlebih dahulu.")
    return

# Mengambil nilai zona waktu dari input untuk penyesuaian UTC
try:
    tz_offset = float(self.entry_pt_tz.get())
except ValueError:
    tz_offset = 7.0 # Default ke WIB (UTC+7) jika kosong

# Membuat struktur Header file iCalendar murni
ics_content = [
    "BEGIN:VCALENDAR",
    "VERSION:2.0",
    "PRODID:-//KHGT Times By Kasmui//ID",
    "CALSCALE:GREGORIAN"
]

# Waktu pembuatan file (Stamp) dalam format UTC

```

```

now_utc_str = datetime.datetime.now(datetime.timezone.utc).strftime('%Y%m%dT%H%M%SZ')

# Looping setiap jadwal salat yang ada di memori
for prayer_name, dt_local in self.daily_prayer_schedule.items():
    # Konversi waktu lokal (naive) kembali ke UTC karena standar ICS wajib UTC
    dt_utc = dt_local - datetime.timedelta(hours=tz_offset)

    # Format waktu sesuai standar ICS: YYYYMMDDThhmmssZ
    dt_start_str = dt_utc.strftime('%Y%m%dT%H%M%SZ')

    # Asumsi durasi event salat di kalender adalah 15 menit
    dt_end_str = (dt_utc + datetime.timedelta(minutes=15)).strftime('%Y%m%dT%H%M%SZ')

    # Membuat Unique ID untuk kalender
    uid = f"{dt_start_str}-{prayer_name.replace('/', ' ').replace(' ', '')}@khggtimes"

    # Memasukkan data event salat
    ics_content.extend([
        "BEGIN:VEVENT",
        f"UID:{uid}",
        f"DTSTAMP:{now_utc_str}",
        f"SUMMARY:Waktu Salat {prayer_name}",
        f"DTSTART:{dt_start_str}",
        f"DTEND:{dt_end_str}",
        "BEGIN:VALARM",
        "ACTION:DISPLAY",
        f"DESCRIPTION:Telah masuk waktu {prayer_name}",
        "TRIGGER:-PT10M", # Setel pengingat (notifikasi) 10 menit sebelum waktu salat
        "END:VALARM",
        "END:VEVENT"
    ])

ics_content.append("END:VCALENDAR")
final_ics = "\n".join(ics_content)

# Dialog penyimpanan file
filepath = filedialog.asksaveasfilename(
    initialfile=f"Jadwal_Salat_{datetime.datetime.now().strftime('%Y%m%d')}.ics",
    defaultextension=".ics",
    filetypes=[("iCalendar Files", "*.ics")]
)

# Eksekusi penyimpanan ke format .ics
if filepath:
    try:
        with open(filepath, 'w', encoding='utf-8') as f:
            f.write(final_ics)
    
```

```

        messagebox.showinfo("Sukses", f"File Kalender ICS berhasil dibuat murni tanpa library
eksternal!\nDisimpan di:\n{filepath}\n\nFile ini siap diimpor ke Google Calendar, Apple Calendar, atau
Outlook.")
    except Exception as e:
        messagebox.showerror("Error", f"Gagal menyimpan file ICS: {e}")

def save_to_txt(self):
    mode = self.combo_mode.get()
    report_data = self.textbox.get("1.0", "end-1c")

    # Handle khusus untuk Modul Gerhana (Mengekstrak data dari Treeview)
    if "Analisis Gerhana" in mode:
        report_data = "DATA ANALISIS GERHANA\n" + "="*80 + "\n"
        headers = [self.tabel_gerhana.heading(c)["text"] for c in self.tabel_gerhana["columns"]]
        report_data += " | ".join(headers) + "\n" + "-"*80 + "\n"
        for item_id in self.tabel_gerhana.get_children():
            row = self.tabel_gerhana.item(item_id)["values"]
            report_data += " | ".join(str(val) for val in row) + "\n"

    if not report_data.strip() or "TERJADI KESALAHAN" in report_data:
        messagebox.showwarning("Peringatan", "Tidak ada data kalkulasi valid yang bisa disimpan.")
        return

    filepath = filedialog.asksaveasfilename(
        initialfile="Laporan_KHGT_Times.txt",
        defaulttextextension=".txt",
        filetypes=[("Text Files", "*.txt")]
    )

    if filepath:
        try:
            with open(filepath, "w", encoding="utf-8") as f:
                f.write(report_data)
            messagebox.showinfo("Sukses", f"Data berhasil disimpan di:\n{filepath}")
            if getattr(self, 'sidebar_visible', False):
                self.toggle_sidebar()
        except Exception as e:
            messagebox.showerror("Error", f"Gagal menyimpan file:\n{str(e)}")

def export_to_csv(self):
    mode = self.combo_mode.get()
    report_data = self.textbox.get("1.0", "end-1c")

    if (not report_data or "TERJADI KESALAHAN" in report_data) and "Analisis Gerhana" not in mode:
        messagebox.showwarning("Peringatan", "Tidak ada data kalkulasi valid.")
        return

    filepath = filedialog.asksaveasfilename(

```



```

        initialfile="Data_Ekstraksi_Falak.csv",
        defaulttextextension=".csv",
        filetypes=[("CSV Files", "*.csv"), ("All Files", "*.")]
    )

    if filepath:
        try:
            # utf-8-sig memastikan karakter derajat (°) terbaca sempurna di Microsoft Excel
            with open(filepath, mode='w', newline="", encoding='utf-8-sig') as f:
                writer = csv.writer(f, delimiter=';')

                # Logika 1: Jika berada di Modul Gerhana (Treeview)
                if "Analisis Gerhana" in mode:
                    headers = [self.tabel_gerhana.heading(c)["text"] for c in self.tabel_gerhana["columns"]]
                    writer.writerow(headers)
                    for item_id in self.tabel_gerhana.get_children():
                        row = self.tabel_gerhana.item(item_id)["values"]
                        writer.writerow(row)

                # Logika 2: Modul teks biasa (Mencari karakter pemisah '|')
                else:
                    lines = report_data.split('\n')
                    tabel_ditemukan = False
                    for line in lines:
                        if '|' in line and not line.startswith('=') and not line.startswith('-'):
                            row = [col.strip() for col in line.split('|')]
                            writer.writerow(row)
                            tabel_ditemukan = True

                # Fallback jika ternyata layar tidak memuat tabel kolom '|'
                if not tabel_ditemukan:
                    for line in lines:
                        writer.writerow([line.strip()])

            messagebox.showinfo("Sukses", f"Data berhasil diekstrak dan diekspor ke CSV:\n{filepath}")
        except Exception as e:
            messagebox.showerror("Error CSV", f"Gagal membuat file CSV: {e}")

    def export_to_pdf(self):
        mode = self.combo_mode.get()

        # --- 1. JIKA SEDANG DI MODUL KALENDER (Gunakan Render Kustom HD agar tajam) ---
        if "Kalender Hijriah" in mode:
            img = self._generate_calendar_image()
            filepath = filedialog.asksaveasfilename(
                initialfile=f"Kalender_Hijriah_{self.cal_h_year}H.pdf",
                defaulttextextension=".pdf",
                filetypes=[("PDF Document", "*.pdf")]
            )

```

```

)
if filepath:
    try:
        # PDF tidak mendukung RGBA, harus convert ke RGB
        img.convert('RGB').save(filepath, "PDF", resolution=150.0)
        messagebox.showinfo("Sukses", f"Kalender Hijriah berhasil diekspor ke PDF:\n{filepath}")
    except Exception as e:
        messagebox.showerror("Error", f"Gagal menyimpan PDF Kalender: {e}")
    return

if "Kalender Masehi" in mode:
    img = self._generate_calmasehi_image()
    nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
    filepath = filedialog.asksaveasfilename(
        initialfile=f"Kalender_Masehi_{nama_bulan}_{self.cal_m_year}.pdf",
        defaultextension=".pdf",
        filetypes=[("PDF Document", "*.pdf")]
    )
    if filepath:
        try:
            img.convert('RGB').save(filepath, "PDF", resolution=150.0)
            messagebox.showinfo("Sukses", f"Kalender Masehi berhasil diekspor ke PDF:\n{filepath}")
        except Exception as e:
            messagebox.showerror("Error", f"Gagal menyimpan PDF Kalender: {e}")
        return

# --- 2. UNTUK SEMUA MODUL LAINNYA (TEKS, 3D, MAP, CHART, GERHANA) ---
# Menangkap screenshot main_frame dan menyimpannya sebagai halaman PDF
filepath = filedialog.asksaveasfilename(
    initialfile="Laporan_KHGT_Times.pdf",
    defaultextension=".pdf",
    filetypes=[("PDF Document", "*.pdf")]
)

if filepath:
    try:
        import time
        # Pastikan UI stabil
        self.update_idletasks()
        time.sleep(0.2)

        # Ambil koordinat main_frame (ruang kanan)
        x0 = self.main_frame.winfo_rootx()
        y0 = self.main_frame.winfo_rooty()
        w = self.main_frame.winfo_width()
        h = self.main_frame.winfo_height()

        # Tangkap area tersebut

```

```

bbox = (x0, y0, x0 + w, y0 + h)
screenshot = ImageGrab.grab(bbox=bbox)

# Konversi hasil screenshot ke PDF (Wajib convert ke RGB)
screenshot.convert('RGB').save(filepath, "PDF", resolution=100.0)

messagebox.showinfo("Sukses", f"Laporan visual berhasil diekspor ke PDF:\n{filepath}")

except Exception as e:
    import traceback
    messagebox.showerror("Error PDF", f"Gagal membuat PDF:\n{e}\n\n{traceback.format_exc()}")

def export_to_png(self):
    mode = self.combo_mode.get()

    # --- 1. JIKA SEDANG DI MODUL KALENDER HIJRIAH (Gunakan Render Kustom HD) ---
    if "Kalender Hijriah" in mode:
        img = self._generate_calendar_image()
        filepath = filedialog.asksaveasfilename(
            initialfile=f"Kalender_Hijriah_{self.cal_h_year}H.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender Hijriah berhasil disimpan sbg PNG:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG Kalender: {e}")
        return

    # --- 2. JIKA SEDANG DI MODUL KALENDER MASEHI (Gunakan Render Kustom HD) ---
    if "Kalender Masehi" in mode:
        img = self._generate_calmasehi_image()
        nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
        filepath = filedialog.asksaveasfilename(
            initialfile=f"Kalender_Masehi_{nama_bulan}_{self.cal_m_year}.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender Masehi berhasil disimpan sbg PNG:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG Kalender: {e}")
        return

```

```

# --- 3. UNTUK SEMUA MODUL LAINNYA (TEKS, GRAFIK 3D, MAP, GERHANA, DLL) ---
# Menangkap (Screenshot) seluruh area 'main_frame' (ruang kanan) secara eksak beserta warnanya
filepath = filedialog.asksaveasfilename(
    initialfile="Output_KHGT_Times.png",
    defaultextension=".png",
    filetypes=[("PNG Image", "*.png")]
)

if filepath:
    try:
        import time

        # Sembunyikan tooltip/kursor sejenak & pastikan UI sudah ter-render sempurna
        self.update_idletasks()
        time.sleep(0.2)

        # Ambil koordinat layar dari main_frame (ruang kanan)
        x0 = self.main_frame.winfo_rootx()
        y0 = self.main_frame.winfo_rooty()
        w = self.main_frame.winfo_width()
        h = self.main_frame.winfo_height()

        # Tentukan Bounding Box (Kiri, Atas, Kanan, Bawah)
        bbox = (x0, y0, x0 + w, y0 + h)

        # Tangkap gambar menggunakan Pillow ImageGrab
        img = ImageGrab.grab(bbox=bbox)

        # Simpan ke PNG
        img.save(filepath, "PNG")

        messagebox.showinfo("Sukses", f"Tampilan layar (Ruang Kanan) berhasil disimpan sebagai PNG:\n{filepath}")

    except Exception as e:
        import traceback
        messagebox.showerror("Error PNG", f"Gagal menangkap layar:\n{e}\n\n{traceback.format_exc()}")

def analisis_komparasi_ramadhan_50_tahun(self):
    # Jalankan di thread terpisah agar GUI tidak Not Responding (Freeze)
    threading.Thread(target=self._proses_komparasi_ramadhan, daemon=True).start()

def _proses_komparasi_ramadhan(self):
    # Update UI Status

```

```

        self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Ramadhan...",
text_color="#00E5FF"))
        self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
        self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
        self.after(0, lambda: self.textbox.delete("1.0", "end"))
        self.after(0, lambda: self.textbox.insert("end", "Memulai mesin Ephemeris untuk 50 Tahun (1447 H -
1496 H)...\nMohon tunggu...\n\n"))

        output_lines = []
        output_lines.append(self.get_header(125))
        output_lines.append("[ REKAPITULASI KOMPARASI AWAL RAMADHAN (1447 H - 1496 H
]".center(125))
        output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°,
Eln>=6.4°)".center(125))
        output_lines.append("=" * 125)
        output_lines.append(f'{"Tahun":<8} | {"Waktu Ijtimak (UTC)":<19} | {"1 Ramadhan (KHGT)":<20} | {"1
Ramadhan (MABIMS Sabang)":<28} | {"Status / Selisih"}')
        output_lines.append("-" * 125)

        # Setup Observer Sabang untuk Neo MABIMS
        sabang = ephemeris.Observer()
        sabang.lat = math.radians(5.8942)
        sabang.lon = math.radians(95.3184)
        sabang.elevation = 0
        sabang.pressure = 1010
        sabang.temp = 25

        matahari = ephemeris.Sun()
        bulan = ephemeris.Moon()

        # Iterasi dari 1447 H sampai 1496 H
        for thn_h in range(1447, 1497):
            try:
                # 1. Ambil data 1 Ramadhan KHGT dari Database Bawaan
                # Index 8 adalah bulan ke-9 (Ramadhan)
                data_ramadan = HIJRI_DB[thn_h][8]
                tgl_khgt_str = data_ramadan[2] # Format: DD-MMM-YYYY
                dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")

                # 2. Tanggal Rukyat (29 Syakban) adalah 1 hari sebelum 1 Ramadhan KHGT
                dt_rukyat = dt_khgt - datetime.timedelta(days=1)

                # 3. Cari Waktu Ijtimak di sekitar tanggal Rukyat
                waktu_pencarian = ephemeris.Date(dt_rukyat)
                ijtimak = ephemeris.previous_new_moon(waktu_pencarian + 5) # +5 hari sbg buffer pencarian
                dt_ijtimak_utc = ijtimak.datetime()
                str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")

```

```
# 4. Kalkulasi Sunset di Sabang pada hari Rukyat
sabang.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try:
    waktu_sunset = sabang.next_setting(matahari)
except:
    waktu_sunset = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0)) # Fallback
perkiraan UTC Sunset Sabang
```

```
# 5. Hitung Posisi Bulan saat Sunset Sabang (Toposentrik)
```

```
sabang.date = waktu_sunset
matahari.compute(sabang)
bulan.compute(sabang)
```

```
alt_topo = math.degrees(bulan.alt)
elong_topo = math.degrees(ephemeris.separation(matahari, bulan))
```

```
# 6. Evaluasi Neo MABIMS (Alt >= 3° dan Elong >= 6.4°)
```

```
# Serta pastikan ijtimak sudah terjadi sebelum sunset (umur bulan positif)
```

```
umur_bulan = waktu_sunset - ijtimak
```

```
if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
```

```
    dt_mabims = dt_khgt
    status = "Serentak (Sama)"
```

```
else:
```

```
    # Jika gagal memenuhi MABIMS di Sabang, bulan Syakban digenapkan 30 hari
```

```
    dt_mabims = dt_khgt + datetime.timedelta(days=1)
    status = "Beda (MABIMS Mundur 1 Hari)"
```

```
# 7. Format Output String
```

```
str_khgt = dt_khgt.strftime("%d %b %Y")
```

```
str_mabims = dt_mabims.strftime("%d %b %Y")
```

```
baris = f"{thn_h} H | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
output_lines.append(baris)
```

```
# Sisipkan pemisah per 10 Tahun (Dekade)
```

```
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
```

```
    output_lines.append("-" * 125)
```

```
except Exception as e:
```

```
    output_lines.append(f"{thn_h} H | Error kalkulasi: {str(e)}")
```

```
output_lines.append("-" * 125)
```

```
output_lines.append("* Catatan Metodologi:")
```

```
output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari
struktur HIJRI_DB.")
```

```
output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia
(Sabang, Aceh).")
```

SOURCE CODE KHGT TIMES V8.0

output_lines.append("- Perbedaan biasanya terjadi jika KHGT tervalidasi di Benua Amerika, namun Sabang belum memenuhi syarat MABIMS.")

laporan_final = "\n".join(output_lines)

Kembalikan ke UI (Main Thread)

self.after(0, lambda: self.textbox.insert("1.0", laporan_final))

self.after(0, lambda: self.textbox.configure(state="disabled"))

self.after(0, lambda: self.lbl_status.configure(text="Rekapitulasi 50 Tahun Selesai",
text_color="#00E676"))

self.after(0, lambda: self.btn_hitung.configure(state="normal"))

=====

FUNGSI PEMBERSIH SAAT APLIKASI DITUTUP

=====

def on_closing(self):

try:

self.save_release_language_cache()

self.save_output_language_cache()

except Exception:

pass

self.anim_running = False

self.is_viewing_3d = False

self.eph3d_is_live = False

self.alarm_enabled.set(False)

if HAS_PYGAME:

try:

import pygame

pygame.mixer.music.stop()

pygame.quit()

except:

pass

self.destroy()

import os

os._exit(0)

def get_tz_from_lon(self, lon_val):

"""Menentukan Zona Waktu Administratif (Indonesia & Override Kustom) atau Geografis Global."""

1. OVERRIDE KOTA/NEGARA SPESIFIK (Administratif & DST)

Karena rumus matematis (lon / 15) tidak mendeteksi Daylight Saving Time (DST),

Bapak bisa mendaftarkan nilai bujur kota-kota Eropa/Amerika di sini.

Saat ini London diset ke 1.0 (BST). Jika masuk musim dingin (GMT), cukup ubah ke 0.0.

override_tz = {

-0.1278: 1.0, # London

-1.8904: 1.0, # Birmingham

```

-2.2426: 1.0, # Manchester
-4.2518: 1.0, # Glasgow
-3.1883: 1.0 # Edinburgh
}

# Cek apakah bujur saat ini terdaftar di kamus pengecualian
if lon_val in override_tz:
    return override_tz[lon_val]

# 2. ZONA WAKTU INDONESIA (WIB, WITA, WIT)
if 94.0 <= lon_val < 115.0:
    return 7.0
elif 115.0 <= lon_val < 126.0:
    return 8.0
elif 126.0 <= lon_val <= 141.0:
    return 9.0

# 3. OVERRIDE ZONA WAKTU ALASKA -> UTC-11
elif -180.0 <= lon_val <= -140.0:
    return -11.0

# 4. ZONA WAKTU GLOBAL (Default Matematis)
else:
    return float(int(round(lon_val / 15.0)))

def analisis_komparasi_syawal_50_tahun(self):
    threading.Thread(target=self._proses_komparasi_syawal, daemon=True).start()

def _proses_komparasi_syawal(self):
    # Update UI Status
    self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Syawal...",
text_color="#00E5FF"))
    self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
    self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
    self.after(0, lambda: self.textbox.delete("1.0", "end"))
    self.after(0, lambda: self.textbox.insert("end", "Memulai komputasi Ephemeris untuk penentuan 1
SYAWAL (1447 H - 1496 H)...\nMohon tunggu...\n\n"))

    output_lines = []
    output_lines.append(self.get_header(125))
    output_lines.append("[ REKAPITULASI KOMPARASI AWAL SYAWAL / IDUL FITRI (1447 H - 1496 H)
]").center(125))
    output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°,
Eln>=6.4°)".center(125))
    output_lines.append("=" * 125)

```



```

output_lines.append(f'{"Tahun":<8} | {"Waktu Ijtimak (UTC)":<19} | {"1 Syawal (KHGT)":<20} | {"1 Syawal (MABIMS Sabang)":<28} | {"Status / Selisih"}')
output_lines.append("-" * 125)

# Setup Observer Sabang untuk Neo MABIMS
sabang = ephemeris.Observer()
sabang.lat = math.radians(5.8942)
sabang.lon = math.radians(95.3184)
sabang.elevation = 0
sabang.pressure = 1010
sabang.temp = 25

matahari = ephemeris.Sun()
bulan = ephemeris.Moon()

# Iterasi dari 1447 H sampai 1496 H
for thn_h in range(1447, 1497):
    try:
        # 1. Ambil data 1 Syawal KHGT dari Database Bawaan (Index 9 adalah Syawal)
        data_syawal = HIJRI_DB[thn_h][9]
        tgl_khgt_str = data_syawal[2] # Format: DD-MMM-YYYY
        dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")

        # 2. Tanggal Rukyat (29 Ramadan) adalah 1 hari sebelum 1 Syawal KHGT
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # 3. Cari Waktu Ijtimak Akhir Ramadan
        waktu_pencarian = ephemeris.Date(dt_rukyat)
        ijtimak = ephemeris.previous_new_moon(waktu_pencarian + 5)
        dt_ijtimak_utc = ijtimak.datetime()
        str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")

        # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
        sabang.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
        try:
            waktu_sunset = sabang.next_setting(matahari)
        except:
            waktu_sunset = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

        # 5. Hitung Posisi Bulan saat Sunset Sabang
        sabang.date = waktu_sunset
        matahari.compute(sabang)
        bulan.compute(sabang)

        alt_topo = math.degrees(bulan.alt)
        elong_topo = math.degrees(ephemeris.separation(matahari, bulan))

        # 6. Evaluasi Neo MABIMS

```

```

umur_bulan = waktu_sunset - ijtimak

if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
    dt_mabims = dt_khgt
    status = "Serentak (Sama)"
else:
    # Jika MABIMS Sabang belum tembus, maka Idul Fitri mundur 1 hari (Ramadan istikmal 30
hari)
    dt_mabims = dt_khgt + datetime.timedelta(days=1)
    status = "Beda (MABIMS Mundur 1 Hari)"

# 7. Format Output Tabel
str_khgt = dt_khgt.strftime("%d %b %Y")
str_mabims = dt_mabims.strftime("%d %b %Y")

baris = f"{thn_h} H | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
output_lines.append(baris)

# Sisipkan pemisah per 10 Tahun
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
    output_lines.append("-" * 125)

except Exception as e:
    output_lines.append(f"{thn_h} H | Error kalkulasi: {str(e)}")

output_lines.append("=" * 125)
output_lines.append("* Catatan Metodologi Penentuan 1 Syawal:")
output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari
struktur HIJRI_DB.")
output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia
(Sabang, Aceh).")
output_lines.append("- Jika status 'Beda (MABIMS Mundur 1 Hari)', artinya KHGT sudah merayakan
Idul Fitri, sedangkan wilayah MABIMS masih berpuasa (istikmal).")

laporan_final = "\n".join(output_lines)

# Lempar ke Main Thread
self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
self.after(0, lambda: self.textbox.configure(state="disabled"))
self.after(0, lambda: self.lbl_status.configure(text="Komparasi 50 Tahun Syawal Selesai",
text_color="#00E676"))
self.after(0, lambda: self.btn_hitung.configure(state="normal"))

def analisis_komparasi_zulhijah_50_tahun(self):
    threading.Thread(target=self._proses_komparasi_zulhijah, daemon=True).start()

def _proses_komparasi_zulhijah(self):
    # Update UI Status

```

```

        self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Zulhijah...",
text_color="#00E5FF"))
        self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
        self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
        self.after(0, lambda: self.textbox.delete("1.0", "end"))
        self.after(0, lambda: self.textbox.insert("end", "Memulai komputasi Ephemeris untuk penentuan 1
ZULHIJAH (1447 H - 1496 H)...\nMohon tunggu...\n\n"))

        output_lines = []
        output_lines.append(self.get_header(125))
        output_lines.append("[ REKAPITULASI KOMPARASI AWAL ZULHIJAH / IDUL ADHA (1447 H - 1496 H)
]".center(125))
        output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°,
Eln>=6.4°)".center(125))
        output_lines.append("=" * 125)
        output_lines.append(f'{"Tahun":<8} | {"Waktu Ijtimak (UTC)":<19} | {"1 Zulhijah (KHGT)":<20} | {"1
Zulhijah (MABIMS Sabang)":<28} | {"Status / Selisih"}')
        output_lines.append("-" * 125)

        # Setup Observer Sabang untuk Neo MABIMS
        sabang = ephem.Observer()
        sabang.lat = math.radians(5.8942)
        sabang.lon = math.radians(95.3184)
        sabang.elevation = 0
        sabang.pressure = 1010
        sabang.temp = 25

        matahari = ephem.Sun()
        bulan = ephem.Moon()

        # Iterasi dari 1447 H sampai 1496 H
        for thn_h in range(1447, 1497):
            try:
                # 1. Ambil data 1 Zulhijah KHGT dari Database Bawaan (Index 11 adalah Zulhijah)
                data_zulhijah = HIJRI_DB[thn_h][11]
                tgl_khgt_str = data_zulhijah[2] # Format: DD-MMM-YYYY
                dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")

                # 2. Tanggal Rukyat (29 Zulkaidah) adalah 1 hari sebelum 1 Zulhijah KHGT
                dt_rukyat = dt_khgt - datetime.timedelta(days=1)

                # 3. Cari Waktu Ijtimak Akhir Zulkaidah
                waktu_pencarian = ephem.Date(dt_rukyat)
                ijtimak = ephem.previous_new_moon(waktu_pencarian + 5)
                dt_ijtimak_utc = ijtimak.datetime()
                str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")

                # 4. Kalkulasi Sunset di Sabang pada hari Rukyat

```

```

sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try:
    waktu_sunset = sabang.next_setting(matahari)
except:
    waktu_sunset = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

# 5. Hitung Posisi Bulan saat Sunset Sabang
sabang.date = waktu_sunset
matahari.compute(sabang)
bulan.compute(sabang)

alt_topo = math.degrees(bulan.alt)
elong_topo = math.degrees(ephem.separation(matahari, bulan))

# 6. Evaluasi Neo MABIMS
umur_bulan = waktu_sunset - ijtimak

if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
    dt_mabims = dt_khgt
    status = "Serentak (Sama)"
else:
    # Jika MABIMS Sabang belum tembus, maka Idul Adha mundur 1 hari (Zulkaidah istikmal 30
hari)
    dt_mabims = dt_khgt + datetime.timedelta(days=1)
    status = "Beda (MABIMS Mundur 1 Hari)"

# 7. Format Output Tabel
str_khgt = dt_khgt.strftime("%d %b %Y")
str_mabims = dt_mabims.strftime("%d %b %Y")

baris = f"{thn_h} H | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
output_lines.append(baris)

# Sisipkan pemisah per 10 Tahun
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
    output_lines.append("-" * 125)

except Exception as e:
    output_lines.append(f"{thn_h} H | Error kalkulasi: {str(e)}")

output_lines.append("-" * 125)
output_lines.append("* Catatan Metodologi Penentuan 1 Zulhijah:")
output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari
struktur HIJRI_DB.")
output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia
(Sabang, Aceh).")
output_lines.append("- Jika status 'Beda (MABIMS Mundur 1 Hari)', artinya KHGT sudah wukuf/Idul
Adha lebih dulu, sedangkan MABIMS istikmal Zulkaidah.")

```

```

laporan_final = "\n".join(output_lines)

# Lempar ke Main Thread
self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
self.after(0, lambda: self.textbox.configure(state="disabled"))
self.after(0, lambda: self.lbl_status.configure(text="Komparasi 50 Tahun Zulhijah Selesai",
text_color="#00E676"))
self.after(0, lambda: self.btn_hitung.configure(state="normal"))

def analisis_komparasi_global_50_tahun(self):
    """Memicu perhitungan komparasi 12 bulan dalam thread terpisah"""
    threading.Thread(target=self._proses_komparasi_global_50_tahun, daemon=True).start()

def _proses_komparasi_global_50_tahun(self):
    self.after(0, lambda: self.lbl_status.configure(text="Audit Global 12 Bulan (50 Thn)...",
text_color="#00E5FF"))
    self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
    self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
    self.after(0, lambda: self.textbox.delete("1.0", "end"))
    self.after(0, lambda: self.textbox.insert("end", "Memproses data 600 bulan Hijriah (1447H-
1496H)...\nMohon tunggu...\n\n"))

    output_lines = []
    output_lines.append(self.get_header(130))
    output_lines.append("[ ANALISIS PROBABILITAS GLOBAL & KOMPARASI 12 BULAN (1447 H - 1496 H)
]").center(130))
    output_lines.append("KHGT (Global Alt>=5, Eln>=8) vs Neo MABIMS (Lokal Sabang Alt>=3,
Eln>=6.4)".center(130))
    output_lines.append("=" * 130)
    output_lines.append(f'{'Tahun':<7} | {'Bulan Hijriah':<15} | {'1 KHGT (Masehi)':<18} | {'1 MABIMS
(Sabang)':<20} | {'Status'}}")
    output_lines.append("-" * 130)

    total_bulan, total_serentak, total_beda = 0, 0, 0
    sabang = ephem.Observer()
    sabang.lat, sabang.lon = math.radians(5.8942), math.radians(95.3184)
    sabang.pressure, sabang.temp = 1010, 25
    sun, moon = ephem.Sun(), ephem.Moon()
    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}

    for thn_h in range(1447, 1497):
        if thn_h not in HIJRI_DB: continue
        output_lines.append(f"--- TAHUN {thn_h} H ---")
        for m_idx in range(12):
            try:
                total_bulan += 1

```

```

db_entry = HIJRI_DB[thn_h][m_idx]
nama_bln = db_entry[0]
parts = db_entry[2].split('-')
d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
dt_khgt = datetime.datetime(y, m, d)
dt_rukyat = dt_khgt - datetime.timedelta(days=1)
ijtimak = ephemer.previous_new_moon(ephemer.Date(dt_rukyat) + 5)
sabang.date = ephemer.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try: ss_sabang = sabang.next_setting(sun)
except: ss_sabang = ephemer.Date(dt_rukyat.replace(hour=11, minute=30))
sabang.date = ss_sabang
sun.compute(sabang); moon.compute(sabang)
alt_t, eln_t = math.degrees(moon.alt), math.degrees(ephemer.separation(sun, moon))
if (ss_sabang - ijtimak)*24 > 0 and alt_t >= 3.0 and eln_t >= 6.4:
    dt_mabims, status, total_serentak = dt_khgt, "Serentak", total_serentak + 1
else:
    dt_mabims, status, total_beda = dt_khgt + datetime.timedelta(days=1), "Beda", total_beda
+ 1
    output_lines.append(f'{"":<7} | {nama_bln:<15} | {dt_khgt.strftime("%d %b %Y"):<18} |
{dt_mabims.strftime("%d %b %Y"):<20} | {status}')
    except: continue
    output_lines.append("-" * 130)

prob = (total_serentak / total_bulan * 100) if total_bulan > 0 else 0
res_summary = (f'\n{" "*130}\nRINGKASAN STATISTIK (50 TAHUN):\n'
    f'- Total Bulan: {total_bulan} | Serentak: {total_serentak} | Berbeda: {total_beda}\n'
    f'- Probabilitas Keseragaman Global: {prob:.2f}%\n{" "*130}')
output_lines.append(res_summary)
self.after(0, lambda: self.display_result("\n".join(output_lines)))

def _proses_tabel_elongasi_50_tahun(self):
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe / Anti-Kosong)
    def _set_ui_loading():
        self.lbl_status.configure(text="Menganalisis Elongasi 50 Tahun...", text_color="#00E5FF")
        self.textbox.configure(state="normal", wrap="none")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal
Ramadhan)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
        self.textbox.configure(state="disabled")

    def _set_ui_done(hasil_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", hasil_teks)
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Kalkulasi Elongasi Ramadhan Selesai", text_color="#00E676")
        self.btn_hitung.configure(state="normal")

```

```

def _set_ui_error(err_teks):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
    self.btn_hitung.configure(state="normal")

# Panggil status loading ke layar
self.after(0, _set_ui_loading)

# 2. Proses Komputasi Utama
try:
    output_lines = []
    output_lines.append(self.get_header(148))
    output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (RAMADHAN 1447H - 1496H)
]".center(148))
    output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Ramadhan
di Sabang)".center(148))
    output_lines.append("=" * 148)
    # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
    output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} |
{'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Ramadhan'}")
    output_lines.append("-" * 148)

# Dictionary untuk parsing tanggal aman
bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
              "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

import math
import ephemeris
import pytz

# Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
engine = ephemeris.Observer()
engine.lat = math.radians(5.8942)
engine.lon = math.radians(95.3184)
engine.elevation = 0
engine.pressure = 1010
engine.temp = 25

sun = ephemeris.Sun()
moon = ephemeris.Moon()

for thn_h in range(1447, 1497):
    try:
        if thn_h not in HIJRI_DB:
            continue

```

```

# INDEX 8 ADALAH BULAN RAMADHAN
data_ramadan = HIJRI_DB[thn_h][8]

# Parsing tanggal KHGT
parts = data_ramadan[2].split('-')
d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

dt_khgt = datetime.datetime(y, m, d)

# PERBAIKAN: Hari Rukyat adalah 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT)
dt_rukyat = dt_khgt - datetime.timedelta(days=1)

# Cari waktu Ijtimak
t_ref = ephemeris.Date(dt_rukyat)
ijtimak = ephemeris.previous_new_moon(t_ref + 2)
str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

# Evaluasi Sunset di Sabang pada Hari Rukyat
engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try:
    ss_sabang = engine.next_setting(sun)
except:
    # Fallback jika error (Anomali ephemeris), asumsi jam 18:30 LT (11:30 UTC)
    ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

# Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

# =====
# A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
# =====
engine.date = ss_sabang
sun.compute(engine)
moon.compute(engine)

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephemeris.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    kesimpulan = "Serentak (Sama)"
else:
    kesimpulan = "Beda (MABIMS Mundur 1 Hari)"

# Format Elongasi ditaruh di depan untuk Menu 23
str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"

```



```

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
# Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    # Elongasi Geosentris
    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

    # Altitude Geosentris Relatif Terhadap Sabang
    gmst = t_sunset_sky.gast
    lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

    ha_deg = lst_deg - (ra_m.hours * 15.0)
    lat_rad = math.radians(5.8942) # Lintang Sabang
    d_rad = dec_m.radians
    ha_rad = math.radians(ha_deg)

    sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

    str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
except Exception as e_geo:
    str_khgt = "Error Kalkulasi Geo"

# Menulis baris ke laporan
output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} |
{str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

# Garis pembatas per dekade
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
    output_lines.append("-" * 148)

except Exception as err_baris:
    output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

output_lines.append("=" * 148)
output_lines.append("* Catatan Pembacaan Tabel:")
output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29
Syakban (H-1 sebelum 1 Ramadhan versi KHGT).")

```

```
        output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
```

```
        output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
```

```
        output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga harus istikmal (genap 30 hari).")
```

```
        # --- MENGIRIM TEKS KE LAYAR ---
```

```
        final_text = "\n".join(output_lines)
```

```
        self.after(0, lambda: _set_ui_done(final_text))
```

```
except Exception as e:
```

```
    import traceback
```

```
    self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
```

```
def _proses_tabel_ketinggian_50_tahun(self):
```

```
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe / Anti-Kosong)
```

```
    def _set_ui_loading():
```

```
        self.lbl_status.configure(text="Menganalisis Ketinggian 50 Tahun...", text_color="#00E5FF")
```

```
        self.textbox.configure(state="normal", wrap="none")
```

```
        self.textbox.delete("1.0", "end")
```

```
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Ramadhan)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
```

```
        self.textbox.configure(state="disabled")
```

```
    def _set_ui_done(hasil_teks):
```

```
        self.textbox.configure(state="normal")
```

```
        self.textbox.delete("1.0", "end")
```

```
        self.textbox.insert("1.0", hasil_teks)
```

```
        self.textbox.configure(state="disabled")
```

```
        self.lbl_status.configure(text="Kalkulasi Ketinggian Ramadhan Selesai", text_color="#00E676")
```

```
        self.btn_hitung.configure(state="normal")
```

```
    def _set_ui_error(err_teks):
```

```
        self.textbox.configure(state="normal")
```

```
        self.textbox.delete("1.0", "end")
```

```
        self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
```

```
        self.textbox.configure(state="disabled")
```

```
        self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
```

```
        self.btn_hitung.configure(state="normal")
```

```
    # Panggil status loading ke layar
```

```
    self.after(0, _set_ui_loading)
```

```
    # 2. Proses Komputasi Utama
```

```
    try:
```

```
        output_lines = []
```

```
        output_lines.append(self.get_header(148))
```

```

        output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (RAMADHAN 1447H - 1496H)
]".center(148))
        output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Ramadhan
di Sabang)".center(148))
        output_lines.append("=" * 148)
        # PERBAIKAN HEADER TABEL AGAR JELAS
        output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} |
{'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Ramadhan'}")
        output_lines.append("-" * 148)

# Dictionary untuk parsing tanggal aman
bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
             "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

import math
import ephemeris
import pytz

# Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
engine = ephemeris.Observer()
engine.lat = math.radians(5.8942)
engine.lon = math.radians(95.3184)
engine.elevation = 0
engine.pressure = 1010
engine.temp = 25

sun = ephemeris.Sun()
moon = ephemeris.Moon()

for thn_h in range(1447, 1497):
    try:
        if thn_h not in HIJRI_DB:
            continue

        # INDEX 8 ADALAH BULAN RAMADHAN
        data_ramadan = HIJRI_DB[thn_h][8]

        # Parsing tanggal KHGT
        parts = data_ramadan[2].split('-')
        d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

        dt_khgt = datetime.datetime(y, m, d)

        # PERBAIKAN: Hari Rukyat adalah 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT)
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # Cari waktu Ijtimak
        t_ref = ephemeris.Date(dt_rukyat)

```

```

ijtimak = ephemer.previous_new_moon(t_ref + 2)
str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

# Evaluasi Sunset di Sabang pada Hari Rukyat
engine.date = ephemer.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try:
    ss_sabang = engine.next_setting(sun)
except:
    # Fallback jika error, jam 18:30 LT (11:30 UTC)
    ss_sabang = ephemer.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

# Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

# =====
# A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
# =====
engine.date = ss_sabang
sun.compute(engine)
moon.compute(engine)

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephemer.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    status_mabims = "Lolos"
    kesimpulan = "Serentak (Sama)"
else:
    status_mabims = "Gagal"
    kesimpulan = "Beda (MABIMS Mundur 1 Hari)"

str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
# Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    # Elongasi Geosentris
    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

```

```

# Altitude Geosentris Relatif Terhadap Sabang
gmst = t_sunset_sky.gast
lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

ha_deg = lst_deg - (ra_m.hours * 15.0)
lat_rad = math.radians(5.8942) # Lintang Sabang
d_rad = dec_m.radians
ha_rad = math.radians(ha_deg)

sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
except Exception as e_geo:
    str_khgt = "Error Kalkulasi Geo"

# Menulis baris ke laporan
output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} |
{str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

# Garis pembatas per dekade
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
    output_lines.append("-" * 148)

except Exception as err_baris:
    output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

output_lines.append("=" * 148)
output_lines.append("* Catatan Pembacaan Tabel:")
output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29
Syakban (H-1 sebelum 1 Ramadhan versi KHGT).")
output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi
(Geocentric) pada saat Waktu Referensi tersebut.")
output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari
permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai
kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga harus istikmal (genap 30 hari).")

# --- MENGIRIM TEKS KE LAYAR ---
final_text = "\n".join(output_lines)
self.after(0, lambda: _set_ui_done(final_text))

except Exception as e:
    import traceback
    self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

```

```

def _proses_tabel_ketinggian_syawal_50_tahun(self):
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
    def _set_ui_loading():
        self.lbl_status.configure(text="Menganalisis Ketinggian Syawal...", text_color="#00E5FF")
        self.textbox.configure(state="normal", wrap="none")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Syawal)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
        self.textbox.configure(state="disabled")

    def _set_ui_done(hasil_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", hasil_teks)
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Kalkulasi Ketinggian Syawal Selesai", text_color="#00E676")
        self.btn_hitung.configure(state="normal")

    def _set_ui_error(err_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
        self.btn_hitung.configure(state="normal")

    self.after(0, _set_ui_loading)

    # 2. Proses Komputasi Utama Syawal
    try:
        output_lines = []
        output_lines.append(self.get_header(148))
        output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (SYAWAL 1447H - 1496H)
]".center(148))
        output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Syawal di Sabang)".center(148))
        output_lines.append("=" * 148)
        # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
        output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Syawal'}")
        output_lines.append("-" * 148)

        bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
                    "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

        import math
        import ephem

```

```

import pytz

# Setup Observer untuk Sabang (Referensi MABIMS)
engine = ephem.Observer()
engine.lat = math.radians(5.8942)
engine.lon = math.radians(95.3184)
engine.elevation = 0
engine.pressure = 1010
engine.temp = 25

sun = ephem.Sun()
moon = ephem.Moon()

for thn_h in range(1447, 1497):
    try:
        if thn_h not in HIJRI_DB:
            continue

        # INDEX 9 ADALAH BULAN SYAWAL
        data_syawal = HIJRI_DB[thn_h][9]

        # Parsing tanggal KHGT
        parts = data_syawal[2].split('-')
        d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

        dt_khgt = datetime.datetime(y, m, d)

        # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal KHGT)
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # Cari waktu Ijtimak
        t_ref = ephem.Date(dt_rukyat)
        ijtimak = ephem.previous_new_moon(t_ref + 2)
        str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

        # Evaluasi Sunset di Sabang pada akhir Ramadhan (Hari Rukyat)
        engine.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
        try:
            ss_sabang = engine.next_setting(sun)
        except:
            # Fallback jika error, jam 18:30 LT (11:30 UTC)
            ss_sabang = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

        # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
        dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
        str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

        # =====

```

```

# A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
# =====
engine.date = ss_sabang
sun.compute(engine)
moon.compute(engine)

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephem.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    kesimpulan = "Serentak (Idul Fitri Sama)"
else:
    kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"

# Format Altitude ditaruh di depan untuk Menu 24
str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
# Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    # Elongasi Geosentris
    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

    # Altitude Geosentris Relatif Terhadap Sabang
    gmst = t_sunset_sky.gast
    lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

    ha_deg = lst_deg - (ra_m.hours * 15.0)
    lat_rad = math.radians(5.8942) # Lintang Sabang
    d_rad = dec_m.radians
    ha_rad = math.radians(ha_deg)

    sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

    str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
except Exception as e_geo:

```



```

        str_khgt = "Error Kalkulasi Geo"

    # Menulis baris ke laporan
    output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

    # Garis pembatas per dekade
    if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
        output_lines.append("-" * 148)

    except Exception as err_baris:
        output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

    output_lines.append("=" * 148)
    output_lines.append("* Catatan Pembacaan Tabel:")
    output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Ramadhan (H-1 sebelum 1 Syawal versi KHGT).")
    output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
    output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
    output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Ramadhan harus diistimkan (digenapkan 30 hari).")

    # --- MENGIRIM TEKS KE LAYAR ---
    final_text = "\n".join(output_lines)
    self.after(0, lambda: _set_ui_done(final_text))

    except Exception as e:
        import traceback
        self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

def _proses_tabel_ketinggian_zulhijah_50_tahun(self):
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
    def _set_ui_loading():
        self.lbl_status.configure(text="Menganalisis Ketinggian Zulhijah...", text_color="#00E5FF")
        self.textbox.configure(state="normal", wrap="none")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Zulhijah)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
        self.textbox.configure(state="disabled")

    def _set_ui_done(hasil_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", hasil_teks)
        self.textbox.configure(state="disabled")

```

```

self.lbl_status.configure(text="Kalkulasi Ketinggian Zulhijah Selesai", text_color="#00E676")
self.btn_hitung.configure(state="normal")

def _set_ui_error(err_teks):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
    self.btn_hitung.configure(state="normal")

self.after(0, _set_ui_loading)

# 2. Proses Komputasi Utama Zulhijah
try:
    output_lines = []
    output_lines.append(self.get_header(148))
    output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (ZULHIJAH 1447H - 1496H)
]".center(148))
    output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Zulhijah di
Sabang)".center(148))
    output_lines.append("=" * 148)
    output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} |
{'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Zulhijah'}")
    output_lines.append("-" * 148)

    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
                  "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

    import math
    import ephemeris
    import pytz

    # Setup Observer untuk Sabang (Referensi MABIMS)
    engine = ephemeris.Observer()
    engine.lat = math.radians(5.8942)
    engine.lon = math.radians(95.3184)
    engine.elevation = 0
    engine.pressure = 1010
    engine.temp = 25

    sun = ephemeris.Sun()
    moon = ephemeris.Moon()

    for thn_h in range(1447, 1497):
        try:
            if thn_h not in HIJRI_DB:
                continue

```

```

# INDEX 11 ADALAH BULAN ZULHIJAH
data_zulhijah = HIJRI_DB[thn_h][11]

# Parsing tanggal KHGT
parts = data_zulhijah[2].split('-')
d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

dt_khgt = datetime.datetime(y, m, d)

# Hari rukyat Zulhijah adalah tanggal 29 Zulkaidah (H-1 dari 1 Zulhijah KHGT)
dt_rukyat = dt_khgt - datetime.timedelta(days=1)

# Cari waktu Ijtimak
t_ref = ephemeris.Date(dt_rukyat)
ijtimak = ephemeris.previous_new_moon(t_ref + 2)
str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

# Evaluasi Sunset di Sabang pada akhir Zulkaidah (Hari Rukyat)
engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
try:
    ss_sabang = engine.next_setting(sun)
except:
    # Fallback jika error, jam 18:30 LT (11:30 UTC)
    ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

# Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

# =====
# A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
# =====
engine.date = ss_sabang
sun.compute(engine)
moon.compute(engine)

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephemeris.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    kesimpulan = "Serentak (Idul Adha Sama)"
else:
    kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"

# Format Altitude ditaruh di depan untuk Menu Ketinggian
str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"

```

```

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

    gmst = t_sunset_sky.gast
    lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

    ha_deg = lst_deg - (ra_m.hours * 15.0)
    lat_rad = math.radians(5.8942) # Lintang Sabang
    d_rad = dec_m.radians
    ha_rad = math.radians(ha_deg)

    sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

    str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
except Exception as e_geo:
    str_khgt = "Error Kalkulasi Geo"

# Menulis baris ke laporan
output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} |
{str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

# Garis pembatas per dekade
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
    output_lines.append("-" * 148)

except Exception as err_baris:
    output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

output_lines.append("=" * 148)
output_lines.append("* Catatan Pembacaan Tabel:")
output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29
Zulkaidah (H-1 sebelum 1 Zulhijah versi KHGT).")
output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi
(Geocentric) pada saat Waktu Referensi tersebut.")

```

```

        output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari
permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
        output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai
kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Zulkaidah harus diistimikan.")

```

```

        final_text = "\n".join(output_lines)
        self.after(0, lambda: _set_ui_done(final_text))

```

```

except Exception as e:
    import traceback
    self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

```

```

def _proses_tabel_elongasi_syawal_50_tahun(self):
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
    def _set_ui_loading():
        self.lbl_status.configure(text="Menganalisis Elongasi Syawal...", text_color="#00E5FF")
        self.textbox.configure(state="normal", wrap="none")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal
Syawal)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
        self.textbox.configure(state="disabled")

```

```

    def _set_ui_done(hasil_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", hasil_teks)
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Kalkulasi Elongasi Syawal Selesai", text_color="#00E676")
        self.btn_hitung.configure(state="normal")

```

```

    def _set_ui_error(err_teks):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
        self.btn_hitung.configure(state="normal")

```

```

    self.after(0, _set_ui_loading)

```

```

    # 2. Proses Komputasi Utama Syawal (Fokus Elongasi)
    try:
        output_lines = []
        output_lines.append(self.get_header(148))
        output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (SYAWAL 1447H - 1496H)
]".center(148))
        output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Syawal di
Sabang)".center(148))

```

```

output_lines.append("=" * 148)
# PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} |  
{'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Syawal'}")
output_lines.append("-" * 148)

bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,  
             "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

import math
import ephem
import pytz

# Setup Observer untuk Sabang (Referensi MABIMS)
engine = ephem.Observer()
engine.lat = math.radians(5.8942)
engine.lon = math.radians(95.3184)
engine.elevation = 0
engine.pressure = 1010
engine.temp = 25

sun = ephem.Sun()
moon = ephem.Moon()

for thn_h in range(1447, 1497):
    try:
        if thn_h not in HIJRI_DB:
            continue

        # INDEX 9 ADALAH BULAN SYAWAL
        data_syawal = HIJRI_DB[thn_h][9]

        # Parsing tanggal KHGT
        parts = data_syawal[2].split('-')
        d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

        dt_khgt = datetime.datetime(y, m, d)

        # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal KHGT)
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # Cari waktu Ijtimak
        t_ref = ephem.Date(dt_rukyat)
        ijtimak = ephem.previous_new_moon(t_ref + 2)
        str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

        # Evaluasi Sunset di Sabang pada akhir Ramadhan (Hari Rukyat)
        engine.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))

```

```

try:
    ss_sabang = engine.next_setting(sun)
except:
    # Fallback jika error, jam 18:30 LT (11:30 UTC)
    ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

# Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

# =====
# A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
# =====
engine.date = ss_sabang
sun.compute(engine)
moon.compute(engine)

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephemeris.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    kesimpulan = "Serentak (Idul Fitri Sama)"
else:
    kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"

# Format Elongasi ditaruh di depan untuk Menu 25
str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
# Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    # Elongasi Geosentris
    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

    # Altitude Geosentris Relatif Terhadap Sabang
    gmst = t_sunset_sky.gast
    lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

```

```

        ha_deg = lst_deg - (ra_m.hours * 15.0)
        lat_rad = math.radians(5.8942) # Lintang Sabang
        d_rad = dec_m.radians
        ha_rad = math.radians(ha_deg)

        sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
        alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

        str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
    except Exception as e_geo:
        str_khgt = "Error Kalkulasi Geo"

    # Menulis baris ke laporan
    output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} |
{str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

    # Garis pembatas per dekade
    if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
        output_lines.append("-" * 148)

    except Exception as err_baris:
        output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

    output_lines.append("=" * 148)
    output_lines.append("* Catatan Pembacaan Tabel:")
    output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29
Ramadhan (H-1 sebelum 1 Syawal versi KHGT).")
    output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi
(Geocentric) pada saat Waktu Referensi tersebut.")
    output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari
permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
    output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai
kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Ramadhan harus diistimkan (digenapkan 30
hari).")

    # --- MENGIRIM TEKS KE LAYAR ---
    final_text = "\n".join(output_lines)
    self.after(0, lambda: _set_ui_done(final_text))

    except Exception as e:
        import traceback
        self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

def _proses_tabel_elongasi_zulhijah_50_tahun(self):
    # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
    def _set_ui_loading():
        self.lbl_status.configure(text="Menganalisis Elongasi Zulhijah...", text_color="#00E5FF")

```



```

self.textbox.configure(state="normal", wrap="none")
self.textbox.delete("1.0", "end")
self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal
Zulhijah)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
self.textbox.configure(state="disabled")

def _set_ui_done(hasil_teks):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", hasil_teks)
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Kalkulasi Elongasi Zulhijah Selesai", text_color="#00E676")
    self.btn_hitung.configure(state="normal")

def _set_ui_error(err_teks):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
    self.btn_hitung.configure(state="normal")

self.after(0, _set_ui_loading)

# 2. Proses Komputasi Utama Zulhijah (Fokus Elongasi)
try:
    output_lines = []
    output_lines.append(self.get_header(148))
    output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (ZULHIJAH 1447H - 1496H)
]".center(148))
    output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Zulhijah di
Sabang)".center(148))
    output_lines.append("=" * 148)
    # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
    output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} |
{'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Zulhijah'}")
    output_lines.append("-" * 148)

    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
                  "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}

    import math
    import ephemeris
    import pytz

    # Setup Observer untuk Sabang (Referensi MABIMS)
    engine = ephemeris.Observer()
    engine.lat = math.radians(5.8942)

```

```

engine.lon = math.radians(95.3184)
engine.elevation = 0
engine.pressure = 1010
engine.temp = 25

sun = ephemeris.Sun()
moon = ephemeris.Moon()

for thn_h in range(1447, 1497):
    try:
        if thn_h not in HIJRI_DB:
            continue

        # INDEX 11 ADALAH BULAN ZULHIJAH
        data_zulhijah = HIJRI_DB[thn_h][11]

        # Parsing tanggal KHGT
        parts = data_zulhijah[2].split('-')
        d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])

        dt_khgt = datetime.datetime(y, m, d)

        # Hari rukyat Zulhijah adalah tanggal 29 Zulkaidah (H-1 dari 1 Zulhijah KHGT)
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # Cari waktu Ijtimak
        t_ref = ephemeris.Date(dt_rukyat)
        ijtimak = ephemeris.previous_new_moon(t_ref + 2)
        str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

        # Evaluasi Sunset di Sabang pada akhir Zulkaidah (Hari Rukyat)
        engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
        try:
            ss_sabang = engine.next_setting(sun)
        except:
            # Fallback jika error, jam 18:30 LT (11:30 UTC)
            ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

        # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
        dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
        str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")

        # =====
        # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
        # =====
        engine.date = ss_sabang
        sun.compute(engine)
        moon.compute(engine)

```

```

alt_topo = math.degrees(moon.alt)
eln_topo = math.degrees(ephem.separation(sun, moon))
umur_s = (ss_sabang - ijtimak) * 24

if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
    kesimpulan = "Serentak (Idul Adha Sama)"
else:
    kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"

# Format Elongasi ditaruh di depan untuk Menu Elongasi
str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"

# =====
# B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
# =====
try:
    t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
    obs_center = self.eph['earth'].at(t_sunset_sky)

    s_app_geo = obs_center.observe(self.eph['sun']).apparent()
    m_app_geo = obs_center.observe(self.eph['moon']).apparent()

    eln_geo = s_app_geo.separation_from(m_app_geo).degrees

    gmst = t_sunset_sky.gast
    lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
    ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)

    ha_deg = lst_deg - (ra_m.hours * 15.0)
    lat_rad = math.radians(5.8942) # Lintang Sabang
    d_rad = dec_m.radians
    ha_rad = math.radians(ha_deg)

    sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

    str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
except Exception as e_geo:
    str_khgt = "Error Kalkulasi Geo"

# Menulis baris ke laporan
output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} |
{str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")

# Garis pembatas per dekade
if (thn_h - 1446) % 10 == 0 and thn_h != 1496:

```

```

        output_lines.append("-" * 148)

    except Exception as err_baris:
        output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")

    output_lines.append("=" * 148)
    output_lines.append("* Catatan Pembacaan Tabel:")
    output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29
Zulkaidah (H-1 sebelum 1 Zulhijah versi KHGT).")
    output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi
(Geocentric) pada saat Waktu Referensi tersebut.")
    output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari
permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
    output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai
kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Zulkaidah harus diistimikan.")

    final_text = "\n".join(output_lines)
    self.after(0, lambda: _set_ui_done(final_text))

except Exception as e:
    import traceback
    self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

# =====
# MODUL TAMBAHAN 26: KALENDER HIJRIAH BERJALAN
# =====
def reset_kalender(self, update_ui=True):
    today = datetime.date.today()
    found_y, found_m = 1447, 0 # Default aman

    # Deteksi otomatis bulan hijriah saat ini
    for y, months in HIJRI_DB.items():
        for m_idx, m_data in enumerate(months):
            start = HijriConverter.parse_date(m_data[2])
            if start and start <= today < start + datetime.timedelta(days=m_data[3]):
                found_y, found_m = y, m_idx
                break
    self.cal_h_year = found_y
    self.cal_h_month = found_m

    if update_ui:
        self.render_kalender()

def setup_kalender_out_frame(self):
    # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA DI-SCROLL KE BAWAH
    self.frame_kalender_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="transparent")

    # --- Header Navigasi & Cetak ---

```

```

header_cal = ctk.CTkFrame(self.frame_kalender_out, fg_color="#101018", corner_radius=8)
header_cal.pack(fill="x", padx=15, pady=(0, 10))

btn_prev = ctk.CTkButton(header_cal, text="◀ Sebelumnya", command=self.cal_prev_month,
fg_color="#1F1F1F", hover_color="#333333", width=120)
btn_prev.pack(side="left", padx=15, pady=10)

self.lbl_cal_title = ctk.CTkLabel(header_cal, text="", font=("Segoe UI", 22, "bold"),
text_color="#00E5FF", justify="center")
self.lbl_cal_title.pack(side="left", expand=True)

btn_next = ctk.CTkButton(header_cal, text="Berikutnya ▶", command=self.cal_next_month,
fg_color="#1F1F1F", hover_color="#333333", width=120)
btn_next.pack(side="right", padx=15, pady=10)

frame_export = ctk.CTkFrame(self.frame_kalender_out, fg_color="transparent")
frame_export.pack(fill="x", padx=15, pady=(0, 10))

btn_pdf = ctk.CTkButton(frame_export, text="🖨️ Cetak PDF", font=("Segoe UI", 12, "bold"),
command=self.export_to_pdf, fg_color="#D32F2F", hover_color="#B71C1C")
btn_pdf.pack(side="right", padx=5)

btn_png = ctk.CTkButton(frame_export, text="💾 Simpan PNG", font=("Segoe UI", 12, "bold"),
command=self.export_to_png, fg_color="#F57C00", hover_color="#E65100")
btn_png.pack(side="right", padx=5)

# --- Area Grid Kalender ---
self.grid_cal_frame = ctk.CTkFrame(self.frame_kalender_out, fg_color="#050510",
corner_radius=10)
self.grid_cal_frame.pack(fill="both", expand=True, padx=15, pady=(0, 15))

# --- 2. TAMBAHAN AREA CATATAN HARI PENTING ISLAM ---
self.frame_kalender_events = ctk.CTkFrame(self.frame_kalender_out, fg_color="#1E1E1E",
corner_radius=8, border_width=1, border_color="#333333")
self.frame_kalender_events.pack(fill="x", padx=15, pady=(0, 15))

self.lbl_kalender_events_title = ctk.CTkLabel(self.frame_kalender_events, text="📅 Catatan & Hari
Penting Islam:", font=("Segoe UI", 16, "bold"), text_color="#FFD54F", anchor="w")
self.lbl_kalender_events_title.pack(fill="x", padx=15, pady=(10, 5))

self.lbl_kalender_events_text = ctk.CTkLabel(self.frame_kalender_events, text="", font=("Segoe UI",
14), text_color="#E0E0E0", justify="left", anchor="w")
self.lbl_kalender_events_text.pack(fill="x", padx=15, pady=(0, 15))

def cal_prev_month(self):
    self.cal_h_month -= 1
    if self.cal_h_month < 0:

```

```

        self.cal_h_month = 11
        self.cal_h_year -= 1
        if self.cal_h_year not in HIJRI_DB:
            self.cal_h_year += 1 # Kembalikan batas
            self.cal_h_month = 0
            messagebox.showinfo("Batas Data", "Telah mencapai batas awal database.")
            return
        self.render_kalender()

    def cal_next_month(self):
        self.cal_h_month += 1
        if self.cal_h_month > 11:
            self.cal_h_month = 0
            self.cal_h_year += 1
            if self.cal_h_year not in HIJRI_DB:
                self.cal_h_year -= 1
                self.cal_h_month = 11
                messagebox.showinfo("Batas Data", "Telah mencapai batas akhir database.")
            return
        self.render_kalender()

    def render_kalender(self):
        # Bersihkan grid lama
        for widget in self.grid_cal_frame.winfo_children():
            widget.destroy()

        y = self.cal_h_year
        m = self.cal_h_month

        if y not in HIJRI_DB: return

        m_data = HIJRI_DB[y][m]
        nama_bulan, _, start_date_str, jumlah_hari = m_data

        # Konversi string Masehi (e.g. 26-Jun-2025) ke datetime
        bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
        parts = start_date_str.split('-')
        start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
        end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

        # Update Judul Header
        hijri_title = f"{nama_bulan} {y} H"
        greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
        if start_date.month != end_date.month:
            greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"

        self.lbl_cal_title.configure(text=f"{hijri_title}\n({greg_title})")

```

```

# Kolom Header Hari
days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
for col, day_name in enumerate(days_header):
    lbl = ctk.CTkLabel(self.grid_cal_frame, text=day_name, height=35, font=("Segoe UI", 14, "bold"),
fg_color="#1E88E5", corner_radius=5)
    lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")

# Cari titik awal kolom (Hari Masehi Senin=0, Ahad=6. Kita ubah Ahad=0, Senin=1)
offset = (start_date.weekday() + 1) % 7

row, col = 1, offset
current_g_date = start_date

# Konstruksi Grid Hari
for h_day in range(1, jumlah_hari + 1):

    # ---> Deteksi hari ini untuk border kuning tebal <---
    if current_g_date == datetime.date.today():
        border_warna = "#FFEA00" # Kuning Terang
        border_tebal = 3 # Lebih tebal
    else:
        border_warna = "#424242" # Warna default (abu-abu gelap)
        border_tebal = 1 # Ketebalan standar

    # Frame Per Kotak (Terapkan variabel warna & ketebalan di sini)
    cell = ctk.CTkFrame(self.grid_cal_frame, fg_color="#1E1E1E", corner_radius=8,
border_width=border_tebal, border_color=border_warna)
    cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")

    # Angka Hijriah (Besar di Tengah)
    lbl_h = ctk.CTkLabel(cell, text=str(h_day), font=("Consolas", 32, "bold"), text_color="#FFD54F")
    lbl_h.pack(expand=True)

    # Angka Masehi (Kecil di Bawah Kanan)
    m_str = f"{current_g_date.day} {BULAN_MASEHI[current_g_date.month-1][:3]}
{current_g_date.year}"

    # Tandai Jumat warna sedikit beda
    color_masehi = "#00E676" if col == 5 else "#B0BEC5"
    lbl_g = ctk.CTkLabel(cell, text=m_str, font=("Segoe UI", 11), text_color=color_masehi)
    lbl_g.pack(side="bottom", anchor="e", padx=8, pady=3)

    # Update Posisi
    col += 1
    if col > 6:
        col = 0
        row += 1

```

```

        current_g_date += datetime.timedelta(days=1)

# Sinkronisasi nilai UI di sidebar
try:
    self.combo_kal_h_bulan.set(BULAN_HIJRIAH[self.cal_h_month])
    self.entry_kal_h_tahun.delete(0, 'end')
    self.entry_kal_h_tahun.insert(0, str(self.cal_h_year))
except Exception:
    pass

# Proporsi Seragam Grid Matriks
for i in range(7):
    self.grid_cal_frame.grid_columnconfigure(i, weight=1)

# KUNCI tinggi baris header (row 0) agar tidak menyusut
self.grid_cal_frame.grid_rowconfigure(0, weight=0, minsize=45)
for i in range(1, 7): # Sisa baris tanggal tetap fleksibel
    self.grid_cal_frame.grid_rowconfigure(i, weight=1)

# =====
# INJEKSI LOGIKA HARI PENTING & SEJARAH ISLAM DUNIA
# =====
h_month_num = m + 1
events_list = []

if h_month_num == 1:
    events_list.extend([
        "1 Muharam: Tahun Baru Hijriah (Penetapan Penanggalan Islam oleh Umar bin Khattab, 17 H)",
        "9 Muharam: Puasa Sunnah Tasu'a",
        "10 Muharam: Puasa Sunnah Asyura (Keselamatan Musa AS & Tragedi Karbala, 61 H)",
        "13, 14, 15 Muharam: Puasa Ayyamul Bidh"
    ])
elif h_month_num == 2:
    events_list.extend([
        "13, 14, 15 Safar: Puasa Ayyamul Bidh",
        "27 Safar: Momen awal Rasulullah SAW hijrah meninggalkan Makkah menuju Madinah"
    ])
elif h_month_num == 3:
    events_list.extend([
        "8 Rabiulawal: Kedatangan Nabi SAW di Quba & pendirian Masjid Quba (1 H)",
        "12 Rabiulawal: Maulid Nabi Muhammad SAW (Tahun Gajah) & Wafatnya Rasulullah SAW (11 H)",
        "13, 14, 15 Rabiulawal: Puasa Ayyamul Bidh"
    ])
elif h_month_num == 4:
    events_list.extend([
        "13, 14, 15 Rabiulakhir: Puasa Ayyamul Bidh"
    ])

```



```

elif h_month_num == 5:
    events_list.extend([
        "8 Jumadilawal: Terjadinya Perang Mu'tah melawan Bizantium (8 H)",
        "13, 14, 15 Jumadilawal: Puasa Ayyamul Bidh"
    ])
elif h_month_num == 6:
    events_list.extend([
        "13, 14, 15 Jumadilakhir: Puasa Ayyamul Bidh",
        "22 Jumadilakhir: Wafatnya Khalifah Pertama, Abu Bakar As-Siddiq (13 H)"
    ])
elif h_month_num == 7:
    events_list.extend([
        "13, 14, 15 Rajab: Puasa Ayyamul Bidh",
        "27 Rajab: Peristiwa Agung Isra' Mi'raj Nabi Muhammad SAW",
        "28 Rajab: Pembebasan Baitul Maqdis (Yerusalem) oleh Salahuddin Al-Ayyubi (583 H)"
    ])
elif h_month_num == 8:
    events_list.extend([
        "13, 14, 15 Syakban: Puasa Ayyamul Bidh",
        "15 Syakban: Turunnya perintah peralihan Kiblat dari Masjidil Aqsa ke Ka'bah (2 H)"
    ])
elif h_month_num == 9:
    events_list.extend([
        "1-30 Ramadan: Puasa Wajib Bulan Ramadan",
        "17 Ramadan: Nuzulul Quran & Kemenangan besar Perang Badar (2 H)",
        "20 Ramadan: Fathu Makkah / Penaklukan & Pembebasan Makkah (8 H)",
        "10 Hari Terakhir: I'tikaf & Pencarian Malam Lailatul Qadar"
    ])
elif h_month_num == 10:
    events_list.extend([
        "1 Syawal: Hari Raya Idul Fitri (Haram Berpuasa)",
        "13, 14, 15 Syawal: Puasa Ayyamul Bidh",
        "15 Syawal: Terjadinya Perang Uhud (3 H)",
        "6 Hari di Bulan Syawal: Puasa Sunnah Syawal"
    ])
elif h_month_num == 11:
    events_list.extend([
        "5 Zulkaidah: Disepakatinya Perjanjian Hudaibiyah (6 H)",
        "13, 14, 15 Zulkaidah: Puasa Ayyamul Bidh"
    ])
elif h_month_num == 12:
    events_list.extend([
        "1-9 Zulhijah: Keutamaan Amal & Puasa Sunnah Dzulhijjah",
        "8 Zulhijah: Puasa Tarwiyah",
        "9 Zulhijah: Puasa Arafah & Penyampaian Khutbah Wada' / Perpisahan Nabi SAW (10 H)",
        "10 Zulhijah: Hari Raya Idul Adha / Kurban (Haram Berpuasa)",
        "11, 12, 13 Zulhijah: Hari Tasyriq (Haram Berpuasa)",
        "14, 15 Zulhijah: Puasa Ayyamul Bidh",
    ])

```

```

        "18 Zulhijah: Wafatnya Khalifah Ketiga, Utsman bin Affan (35 H)"
    ])

    if events_list:
        # Menggunakan spasi antar baris yang lega agar nyaman dibaca saat di scroll
        events_text = "\n\n".join(f" ♦ {ev}" for ev in events_list)
    else:
        events_text = " ♦ Tidak ada hari besar spesifik pada bulan ini selain puasa sunnah."

    # Perbarui teks label dengan fitur wraplength agar teks tidak terpotong ke samping
    self.lbl_kalender_events_text.configure(text=events_text, wraplength=950)

def _generate_calendar_image(self):
    """Merender kanvas elegan murni lewat Pillow untuk resolusi Ultra HD"""
    img = Image.new("RGB", (1400, 1000), "#0A0A10")
    draw = ImageDraw.Draw(img)

    try:
        f_title = ImageFont.truetype("arialbd.ttf", 46)
        f_sub = ImageFont.truetype("arial.ttf", 26)
        f_day = ImageFont.truetype("arialbd.ttf", 22)
        f_h = ImageFont.truetype("arialbd.ttf", 64)
        f_m = ImageFont.truetype("arial.ttf", 16)
    except Exception:
        f_title = f_sub = f_day = f_h = f_m = ImageFont.load_default()

    y = self.cal_h_year
    m = self.cal_h_month
    m_data = HIJRI_DB[y][m]
    nama_bulan, _, start_date_str, jumlah_hari = m_data

    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
    parts = start_date_str.split('-')
    start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
    end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

    hijri_title = f"Kalender Hijriah KHGT: {nama_bulan} {y} H"
    greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
    if start_date.month != end_date.month:
        greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"

    draw.text((700, 60), hijri_title, font=f_title, fill="#00E5FF", anchor="mm")
    draw.text((700, 110), greg_title, font=f_sub, fill="#B0BEC5", anchor="mm")

    start_x, start_y = 50, 180
    cell_w, cell_h = 185, 120

```

```

days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]

for i, d_name in enumerate(days_header):
    x0 = start_x + i * cell_w
    y0 = start_y
    x1 = x0 + cell_w - 10
    y1 = y0 + 45
    draw.rectangle([x0, y0, x1, y1], fill="#1E88E5", outline="#1565C0", width=2)
    draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")

offset = (start_date.weekday() + 1) % 7
row, col = 0, offset
curr_g = start_date
grid_y_start = start_y + 60

for h_day in range(1, jumlah_hari + 1):
    x0 = start_x + col * cell_w
    y0 = grid_y_start + row * cell_h
    x1 = x0 + cell_w - 10
    y1 = y0 + cell_h - 10

    draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)

    # --- CEK EVENT EXPORT HIJRIAH ---
    event_color, _ = self._get_islamic_event(h_day, m + 1, curr_g.weekday())
    color_hijri = event_color if event_color else "#FFD54F"

    if event_color:
        draw.ellipse([x0+15, y0+15, x0+25, y0+25], fill=color_hijri) # Titik penanda warna puasa/event

    draw.text(((x0+x1)/2, y0 + 40), str(h_day), font=f_h, fill=color_hijri, anchor="mm")

    # Angka Masehi
    m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3]} {curr_g.year}"
    draw.text((x1 - 10, y1 - 15), m_str, font=f_m, fill="#B0BEC5", anchor="rm")

    col += 1
    if col > 6:
        col = 0
        row += 1
        curr_g += datetime.timedelta(days=1)

    draw.text((700, 960), "KHGT Times Engine V8.0 - By Kasmui", font=f_sub, fill="#555555",
    anchor="mm")
    return img

```

=====

SOURCE CODE KHGT TIMES V8.0

```

# MODUL TAMBAHAN 27: KALENDER MASEHI BERJALAN
# =====
def reset_kalmasehi(self, update_ui=True):
    today = datetime.date.today()
    self.cal_m_year = today.year
    self.cal_m_month = today.month
    if update_ui:
        self.render_kalmasehi()

def setup_kalmasehi_out_frame(self):
    # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA DI-SCROLL KE BAWAH
    self.frame_kalmasehi_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="transparent")

    # --- Header Navigasi & Cetak ---
    header_cal = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#101018", corner_radius=8)
    header_cal.pack(fill="x", padx=15, pady=(0, 10))

    btn_prev = ctk.CTkButton(header_cal, text="◀ Sebelumnya",
command=self.calmasehi_prev_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
    btn_prev.pack(side="left", padx=15, pady=10)

    self.lbl_calm_title = ctk.CTkLabel(header_cal, text="", font=("Segoe UI", 22, "bold"),
text_color="#00E5FF", justify="center")
    self.lbl_calm_title.pack(side="left", expand=True)

    btn_next = ctk.CTkButton(header_cal, text="Berikutnya ▶",
command=self.calmasehi_next_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
    btn_next.pack(side="right", padx=15, pady=10)

    frame_export = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="transparent")
    frame_export.pack(fill="x", padx=15, pady=(0, 10))

    btn_pdf = ctk.CTkButton(frame_export, text="📄 Cetak PDF", font=("Segoe UI", 12, "bold"),
command=self.export_to_pdf, fg_color="#D32F2F", hover_color="#B71C1C")
    btn_pdf.pack(side="right", padx=5)

    btn_png = ctk.CTkButton(frame_export, text="💾 Simpan PNG", font=("Segoe UI", 12, "bold"),
command=self.export_to_png, fg_color="#F57C00", hover_color="#E65100")
    btn_png.pack(side="right", padx=5)

    # --- Area Grid Kalender ---
    self.grid_calm_frame = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#050510",
corner_radius=10)
    self.grid_calm_frame.pack(fill="both", expand=True, padx=15, pady=(0, 15))

    # --- 2. TAMBAHAN AREA CATATAN HARI PENTING MASEHI ---

```

```

self.frame_kalmasehi_events = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#1E1E1E",
corner_radius=8, border_width=1, border_color="#333333")
self.frame_kalmasehi_events.pack(fill="x", padx=15, pady=(0, 15))

self.lbl_kalmasehi_events_title = ctk.CTkLabel(self.frame_kalmasehi_events, text="🔴 Catatan &
Hari Penting (Nasional/Internasional):", font=("Segoe UI", 16, "bold"), text_color="#00E5FF",
anchor="w")
self.lbl_kalmasehi_events_title.pack(fill="x", padx=15, pady=(10, 5))

self.lbl_kalmasehi_events_text = ctk.CTkLabel(self.frame_kalmasehi_events, text="", font=("Segoe
UI", 14), text_color="#E0E0E0", justify="left", anchor="w")
self.lbl_kalmasehi_events_text.pack(fill="x", padx=15, pady=(0, 15))

def cal_masehi_prev_month(self):
    self.cal_m_month -= 1
    if self.cal_m_month < 1:
        self.cal_m_month = 12
        self.cal_m_year -= 1
    self.render_kalmasehi()

def cal_masehi_next_month(self):
    self.cal_m_month += 1
    if self.cal_m_month > 12:
        self.cal_m_month = 1
        self.cal_m_year += 1
    self.render_kalmasehi()

def render_kalmasehi(self):
    # Bersihkan grid lama
    for widget in self.grid_calm_frame.winfo_children():
        widget.destroy()

    y = self.cal_m_year
    m = self.cal_m_month

    _, jumlah_hari = calendar.monthrange(y, m)
    start_date = datetime.date(y, m, 1)
    end_date = datetime.date(y, m, jumlah_hari)

    # Cari rentang bulan Hijriah untuk judul menggunakan data KHGT
    h_start = HijriConverter.get_hijri_date(start_date)
    h_end = HijriConverter.get_hijri_date(end_date)

    greg_title = f"{BULAN_MASEHI[m-1]} {y}"

    # Ekstrak bulan dan tahun Hijriah (misal: "1 Muharam 1446 H" -> "Muharam 1446")
    try:

```

```

h1_parts = h_start.split(" ")
h2_parts = h_end.split(" ")
h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
if h1_m_y != h2_m_y:
    hijri_title = f"{h1_m_y} - {h2_m_y} H"
else:
    hijri_title = f"{h1_m_y} H"
except:
    hijri_title = "Data Hijriah KHGT"

self.lbl_calm_title.configure(text=f"{greg_title}\n({hijri_title})")

# Header Hari Masehi (Senin di awal, Ahad di akhir)
days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
for col, day_name in enumerate(days_header):
    color_bg = "#D84315" if col == 6 else "#E65100" # Ahad warnanya lebih gelap
    lbl = ctk.CTkLabel(self.grid_calm_frame, text=day_name, height=35, font=("Segoe UI", 14, "bold"),
fg_color=color_bg, corner_radius=5)
    lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")

# Offset kalender Masehi (Monday=0, Sunday=6)
offset = start_date.weekday()
row, col = 1, offset

curr_g = start_date
for m_day in range(1, jumlah_hari + 1):

    # ---> Deteksi hari ini untuk border kuning tebal <---
    if curr_g == datetime.date.today():
        border_warna = "#FFEA00" # Kuning Terang
        border_tebal = 3 # Lebih tebal
    else:
        border_warna = "#424242" # Warna default (abu-abu gelap)
        border_tebal = 1 # Ketebalan standar

    # Frame Per Kotak
    cell = ctk.CTkFrame(self.grid_calm_frame, fg_color="#1E1E1E", corner_radius=8,
border_width=border_tebal, border_color=border_warna)
    cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")

    # Angka Masehi Besar (Warna merah khusus untuk Ahad)
    color_masehi = "#FF5252" if col == 6 else "#00E5FF"
    lbl_m = ctk.CTkLabel(cell, text=str(m_day), font=("Consolas", 32, "bold"),
text_color=color_masehi)
    lbl_m.pack(expand=True)

    # Angka Hijriah Kecil di bawah

```

```

h_str = HijriConverter.get_hijri_date(curr_g)
if h_str == "N/A": h_str = "-"

# Singkat nama bulan agar muat
parts = h_str.split()
if len(parts) >= 4:
    h_str_short = f"{parts[0]} {parts[1][:3]} {parts[2]} H"
else:
    h_str_short = h_str

lbl_h = ctk.CTkLabel(cell, text=h_str_short, font=("Segoe UI", 11), text_color="#FFD54F")
lbl_h.pack(side="bottom", anchor="e", padx=8, pady=3)

col += 1
if col > 6:
    col = 0
    row += 1
curr_g += datetime.timedelta(days=1)

# Sinkronisasi nilai UI di sidebar
try:
    self.combo_kal_m_bulan.set(BULAN_MASEHI[self.cal_m_month - 1])
    self.entry_kal_m_tahun.delete(0, 'end')
    self.entry_kal_m_tahun.insert(0, str(self.cal_m_year))
except Exception:
    pass

# Proporsi Seragam
for i in range(7):
    self.grid_calm_frame.grid_columnconfigure(i, weight=1)

# KUNCI tinggi baris header Masehi
self.grid_calm_frame.grid_rowconfigure(0, weight=0, minsize=45)
for i in range(1, 7):
    self.grid_calm_frame.grid_rowconfigure(i, weight=1)

# =====
# INJEKSI LOGIKA HARI PENTING NASIONAL & INTERNASIONAL
# =====
m_events_list = []
tahun_str = str(y) if y > 0 else f"{abs(y-1)} SM"

if m == 1:
    m_events_list.extend([
        f"1 Januari {tahun_str}: Tahun Baru Masehi",
        f"15 Januari {tahun_str}: Hari Peringatan Laut dan Samudera Nasional",
        f"25 Januari {tahun_str}: Hari Gizi Nasional Indonesia"
    ])

```

```

elif m == 2:
    m_events_list.extend([
        f"4 Februari {tahun_str}: Hari Kanker Sedunia",
        f"9 Februari {tahun_str}: Hari Pers Nasional (HPN)",
        f"21 Februari {tahun_str}: Hari Peduli Sampah Nasional"
    ])
elif m == 3:
    m_events_list.extend([
        f"1 Maret {tahun_str}: Peringatan Serangan Umum 1 Maret (Bersejarah 1949)",
        f"8 Maret {tahun_str}: Hari Perempuan Internasional (International Women's Day)",
        f"9 Maret {tahun_str}: Hari Musik Nasional (Kelahiran W.R. Supratman)",
        f"22 Maret {tahun_str}: Hari Air Sedunia",
        f"30 Maret {tahun_str}: Hari Film Nasional"
    ])
elif m == 4:
    m_events_list.extend([
        f"7 April {tahun_str}: Hari Kesehatan Sedunia",
        f"18 April {tahun_str}: Hari Peringatan Konferensi Asia-Afrika (KAA Bandung 1955)",
        f"21 April {tahun_str}: Hari Kartini (Pahlawan Emansipasi Wanita, Lahir 1879)",
        f"22 April {tahun_str}: Hari Bumi (Earth Day)",
        f"23 April {tahun_str}: Hari Buku Sedunia (World Book Day)"
    ])
elif m == 5:
    m_events_list.extend([
        f"1 Mei {tahun_str}: Hari Buruh Internasional (May Day)",
        f"2 Mei {tahun_str}: Hari Pendidikan Nasional (Kelahiran Ki Hajar Dewantara, 1889)",
        f"20 Mei {tahun_str}: Hari Kebangkitan Nasional (Berdirinya Budi Utomo, 1908)",
        f"17 Mei {tahun_str}: Hari Buku Nasional",
        f"29 Mei {tahun_str}: Hari Penaklukan Konstantinopel oleh Sultan Mehmed (Muhammad Al-Fatih) tahun
1453 M)"
    ])
elif m == 6:
    m_events_list.extend([
        f"1 Juni {tahun_str}: Hari Lahir Pancasila (Pidato Bung Karno, 1945)",
        f"5 Juni {tahun_str}: Hari Lingkungan Hidup Sedunia",
        f"22 Juni {tahun_str}: Hari Ulang Tahun Kota Jakarta"
    ])
elif m == 7:
    m_events_list.extend([
        f"12 Juli {tahun_str}: Hari Koperasi Indonesia",
        f"23 Juli {tahun_str}: Hari Anak Nasional"
    ])
elif m == 8:
    m_events_list.extend([
        f"8 Agustus {tahun_str}: Hari Berdirinya ASEAN (1967)",
        f"10 Agustus {tahun_str}: Hari Veteran Nasional",
        f"14 Agustus {tahun_str}: Hari Pramuka (Praja Muda Karana)",
        f"17 Agustus {tahun_str}: Hari Proklamasi Kemerdekaan Republik Indonesia (1945)",

```



```

        f"18 Agustus {tahun_str}: Hari Konstitusi Republik Indonesia"
    })
    elif m == 9:
        m_events_list.extend([
            f"9 September {tahun_str}: Hari Olahraga Nasional (Haornas)",
            f"17 September {tahun_str}: Hari Palang Merah Indonesia (PMI)",
            f"30 September {tahun_str}: Hari Peringatan G30S/PKI (1965)"
        ])
    elif m == 10:
        m_events_list.extend([
            f"1 Oktober {tahun_str}: Hari Kesaktian Pancasila",
            f"2 Oktober {tahun_str}: Hari Batik Nasional",
            f"5 Oktober {tahun_str}: Hari Ulang Tahun Tentara Nasional Indonesia (TNI, Berdiri 1945)",
            f"28 Oktober {tahun_str}: Hari Sumpah Pemuda (Kongres Pemuda II, 1928)"
        ])
    elif m == 11:
        m_events_list.extend([
            f"10 November {tahun_str}: Hari Pahlawan Nasional (Pertempuran Surabaya, 1945)",
            f"12 November {tahun_str}: Hari Ayah Nasional",
            f"25 November {tahun_str}: Hari Guru Nasional (Berdirinya PGRI, 1945)"
        ])
    elif m == 12:
        m_events_list.extend([
            f"1 Desember {tahun_str}: Hari AIDS Sedunia",
            f"22 Desember {tahun_str}: Hari Ibu Nasional (Kongres Perempuan Indonesia, 1928)",
            f"25 Desember {tahun_str}: Hari Raya Natal"
        ])

    if m_events_list:
        # Gunakan spasi antar baris ganda agar lebih leluasa saat dibaca
        events_text = "\n\n".join(f" ♦ {ev}" for ev in m_events_list)
    else:
        events_text = " ♦ Tidak ada catatan hari besar nasional yang spesifik pada bulan ini."

    # Terapkan teks dengan fitur wraplength agar rapi ke bawah
    self.lbl_kalmasehi_events_text.configure(text=events_text, wraplength=950)

def _generate_calmasehi_image(self):
    """Merender kanvas kalender masehi lewat Pillow untuk ekspor PDF & PNG HD"""
    img = Image.new("RGB", (1400, 1000), "#0A0A10")
    draw = ImageDraw.Draw(img)

    try:
        f_title = ImageFont.truetype("arialbd.ttf", 46)
        f_sub = ImageFont.truetype("arial.ttf", 26)
        f_day = ImageFont.truetype("arialbd.ttf", 22)
        f_m = ImageFont.truetype("arialbd.ttf", 64)

```

```

        f_h = ImageFont.truetype("arial.ttf", 16)
    except Exception:
        f_title = f_sub = f_day = f_m = f_h = ImageFont.load_default()

    y = self.cal_m_year
    m = self.cal_m_month

    _, jumlah_hari = calendar.monthrange(y, m)
    start_date = datetime.date(y, m, 1)
    end_date = datetime.date(y, m, jumlah_hari)

    h_start = HijriConverter.get_hijri_date(start_date)
    h_end = HijriConverter.get_hijri_date(end_date)

    greg_title = f"Kalender Masehi: {BULAN_MASEHI[m-1]} {y}"

    try:
        h1_parts = h_start.split(" ")
        h2_parts = h_end.split(" ")
        h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
        h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
        if h1_m_y != h2_m_y:
            hijri_title = f"Data KHGT: {h1_m_y} - {h2_m_y} H"
        else:
            hijri_title = f"Data KHGT: {h1_m_y} H"
    except:
        hijri_title = "Data Hijriah KHGT"

    # Cetak Header Text
    draw.text((700, 60), greg_title, font=f_title, fill="#00E5FF", anchor="mm")
    draw.text((700, 110), hijri_title, font=f_sub, fill="#FFD54F", anchor="mm")

    # Konfigurasi Dimensi Grid
    start_x, start_y = 50, 180
    cell_w, cell_h = 185, 120
    days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]

    # Gambar Header Hari
    for i, d_name in enumerate(days_header):
        x0 = start_x + i * cell_w
        y0 = start_y
        x1 = x0 + cell_w - 10
        y1 = y0 + 45
        color_bg = "#D84315" if i == 6 else "#E65100"
        draw.rectangle([x0, y0, x1, y1], fill=color_bg, outline="#E65100", width=2)
        draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")

    # Gambar Isi Tanggal

```

```

offset = start_date.weekday()
row, col = 0, offset
curr_g = start_date
grid_y_start = start_y + 60

for m_day in range(1, jumlah_hari + 1):
    x0 = start_x + col * cell_w
    y0 = grid_y_start + row * cell_h
    x1 = x0 + cell_w - 10
    y1 = y0 + cell_h - 10

    # Background Box
    draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)

    # --- CEK EVENT EXPORT HIJRIAH ---
    event_color, _ = self._get_islamic_event(h_day, m + 1, curr_g.weekday())
    color_hijri = event_color if event_color else "#FFD54F"

    if event_color:
        draw.ellipse([x0+15, y0+15, x0+25, y0+25], fill=color_hijri) # Titik penanda

    draw.text(((x0+x1)/2, y0 + 40), str(h_day), font=f_h, fill=color_hijri, anchor="mm")

    # Angka Masehi
    m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3]} {curr_g.year}"

    # Masehi Besar
    color_masehi = "#FF5252" if col == 6 else "#00E5FF"
    draw.text(((x0+x1)/2, y0 + 40), str(m_day), font=f_m, fill=color_masehi, anchor="mm")

    # Hijriah Kecil
    h_str = HijriConverter.get_hijri_date(curr_g)
    if h_str == "N/A": h_str = "-"

    parts = h_str.split()
    if len(parts) >= 4:
        h_str_short = f"{parts[0]} {parts[1][:3]} {parts[2]} H"
    else:
        h_str_short = h_str

    # --- CEK EVENT EXPORT MASEHI ---
    color_hijri = "#FFD54F"
    if len(parts) >= 4:
        try:
            h_day_val = int(parts[0])
            h_month_val = BULAN_HIJRIAH.index(parts[1]) + 1
            ev_color, _ = self._get_islamic_event(h_day_val, h_month_val, curr_g.weekday())
            if ev_color:

```

```

        color_hijri = ev_color
        draw.ellipse([x1-25, y0+15, x1-15, y0+25], fill=color_hijri) # Titik penanda
    except: pass

    draw.text((x1 - 10, y1 - 15), h_str_short, font=f_h, fill=color_hijri, anchor="rm")

    col += 1
    if col > 6:
        col = 0
        row += 1
    curr_g += datetime.timedelta(days=1)

# Footer
draw.text((700, 960), "KHGT Times Engine V8.0 - By Kasmui", font=f_sub, fill="#555555",
anchor="mm")
return img

def export_kalmasehi(self, export_format):
    img = self._generate_calmasehi_image()

    y = self.cal_m_year
    nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
    default_file = f"Kalender_Masehi_{nama_bulan}_{y}"

    if export_format == "png":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG
di:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")

        elif export_format == "pdf":
            filepath = filedialog.asksaveasfilename(
                initialfile=f"{default_file}.pdf",
                defaultextension=".pdf",
                filetypes=[("PDF Document", "*.pdf")]
            )
            if filepath:
                try:
                    # Konversi otomatis Image PIL menjadi PDF resolusi tinggi
                    img.save(filepath, "PDF", resolution=150.0)

```

```

        messagebox.showinfo("Sukses", f"Kalender HD berhasil diekspor menjadi PDF
di:\n{filepath}")
    except Exception as e:
        messagebox.showerror("Error", f"Gagal menyimpan PDF: {e}")

# =====
# MODUL 04: ALTITUDE CHART ANALYSER (FINAL & ANTI-FREEZE)
# =====
def trigger_chart(self, chart_type):
    """Memantik render grafik ke background thread berdasarkan tombol yang diklik"""
    self.lbl_status.configure(text="Memproses Data Grafik...", text_color="#00E5FF")
    import threading
    threading.Thread(target=self.calculate_chart_analyser, args=(chart_type,), daemon=True).start()

def calculate_chart_analyser(self, chart_type="altitude"):
    try:
        import math
        import numpy as np
        from skyfield import almanac
        from skyfield.api import wgs84

        # 1. Ambil input UI
        y = int(self.entry_chart_y.get())
        m = int(self.entry_chart_m.get())
        d = int(self.entry_chart_d.get())

        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        tz = float(self.entry_vtz.get())
        elev = float(self.entry_velev.get())

        self.after(0, lambda: self.lbl_status.configure(text="Mengumpulkan data Ephemeris 24 Jam...",
        text_color="#00E5FF"))

        # 2. KALKULASI DI BACKGROUND THREAD
        self.auto_switch_ephemeris(y)
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
        loc = wgs84.latlon(lat, lon, elevation_m=elev)
        observer = earth + loc

        hours = np.linspace(0, 24, 144) # Resolusi per 10 menit

        # Wadah Data Array
        sun_alts, moon_alts, moon_geo_alts = [], [], []
        elongations, illuminations = [], []

        for h in hours:

```

```

t = self.ts.utc(y, m, d, h - tz)

# --- A. Data Toposentrik (Altitude MABIMS) ---
sun_app = observer.at(t).observe(sun).apparent()
moon_app = observer.at(t).observe(moon).apparent()
sun_alts.append(sun_app.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)
moon_alts.append(moon_app.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)

# --- B. Data Geosentrik (Elongasi KHGT) ---
obs_center = earth.at(t)
s_app_geo = obs_center.observe(sun).apparent()
m_app_geo = obs_center.observe(moon).apparent()
elongations.append(s_app_geo.separation_from(m_app_geo).degrees)

# --- C. Data Iluminasi (Fase Bulan) ---
illum_val = almanac.fraction_illuminated(self.eph, 'moon', t)
illuminations.append((illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0)

# --- D. Data Altitude Geosentrik (Tinggi KHGT) ---
gmst = t.gast
lst_deg = (gmst * 15.0) + lon
ra_m, dec_m, _ = m_app_geo.radec(epoch=t)

ha_deg = lst_deg - (ra_m.hours * 15.0)
ha_rad = math.radians(ha_deg)
lat_rad = math.radians(lat)
d_rad = dec_m.radians

sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
moon_geo_alts.append(math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo)))))

# 3. KIRIM HASIL BUNDLE KE MAIN THREAD
data_bundle = {
    'hours': hours, 'sun_alts': sun_alts, 'moon_alts': moon_alts,
    'moon_geo_alts': moon_geo_alts, 'elongations': elongations,
    'illuminations': illuminations, 'y': y, 'm': m, 'd': d, 'type': chart_type
}
self.after(0, self.plot_chart_dispatcher, data_bundle)

except Exception as e:
    import traceback
    self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")

def plot_chart_dispatcher(self, data):
    """Merender canvas matriks berdasarkan tipe chart (Altitude / Elongasi / Iluminasi)"""
    try:
        for widget in self.frame_chart_out.winfo_children():

```

```

        widget.destroy()

from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np
import datetime

fig = Figure(figsize=(8, 5), dpi=100)
fig.patch.set_facecolor('#181818')
ax = fig.add_subplot(111)
ax.set_facecolor('#101010')

chart_type = data['type']
hours = data['hours']
y, m, d = data['y'], data['m'], data['d']

tahun_str = f"{y} M" if y > 0 else f"{abs(y-1)} SM"
nama_bulan = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]
bulan_str = nama_bulan[m] if 1 <= m <= 12 else str(m)
date_display = f"{d:02d} {bulan_str} {tahun_str}"

self.chart_type = chart_type
self.chart_data = data # Simpan referensi array untuk hover interaktif

if chart_type == "altitude":
    ax.plot(hours, data['sun_alts'], color='#FFD54F', label='Matahari (Topo)', linewidth=2)
    ax.plot(hours, data['moon_alts'], color='#80DEEA', label='Bulan (Topo MABIMS)', linewidth=2)
    ax.plot(hours, data['moon_geo_alts'], color='#00E676', label='Bulan (Geo KHGT)', linewidth=2,
linestyle='--')
    ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')

    ax.set_title(f"Grafik Ketinggian Benda Langit: {date_display}", color='white', fontsize=14,
fontweight='bold', pad=15)
    ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)
    ax.set_xlabel("Waktu Lokal (Jam)", color='ffffff', fontsize=12)
    ax.set_xlim(0, 24)
    ax.set_xticks(np.arange(0, 25, 2))
    ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])

    now = datetime.datetime.now()
    if y == now.year and m == now.month and d == now.day:
        curr_h = now.hour + now.minute/60.0
        ax.axvline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
        ax.text(curr_h + 0.2, min(data['sun_alts']), 'Waktu Sekarang', color='#00E676', fontsize=10,
rotation=90, verticalalignment='bottom')

    elif chart_type == "elongation":

```

```

        ax.plot(hours, data['elongations'], color='#FFD700', label='Elongasi Bulan-Matahari (Geo)',
linewidth=2.5)
        ax.axhline(8, color='#00E676', linestyle='--', linewidth=1.5, label='Batas Minimal KHGT (8°)')
        ax.axhline(6.4, color='#FF5252', linestyle=':', linewidth=1.5, label='Batas Minimal MABIMS
(6.4°)')

        ax.set_title(f"Grafik Elongasi (Jarak Sudut): {date_display}", color='white', fontsize=14,
fontweight='bold', pad=15)
        ax.set_ylabel("Elongasi (°)", color='ffffff', fontsize=12)
        ax.set_xlabel("Waktu Lokal (Jam)", color='ffffff', fontsize=12)
        ax.set_xlim(0, 24)
        ax.set_xticks(np.arange(0, 25, 2))
        ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])

    elif chart_type == "illumination":
        ax.plot(hours, data['illuminations'], color='#E040FB', label='Fraksi Iluminasi Bulan (%)',
linewidth=2.5)
        ax.set_title(f"Grafik Iluminasi (Cahaya Fase): {date_display}", color='white', fontsize=14,
fontweight='bold', pad=15)
        ax.set_ylabel("Iluminasi (%)", color='ffffff', fontsize=12)
        ax.set_xlabel("Waktu Lokal (Jam)", color='ffffff', fontsize=12)
        ax.set_xlim(0, 24)
        ax.set_xticks(np.arange(0, 25, 2))
        ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])

    elif chart_type == "illum_vs_elong":
        ax.plot(data['elongations'], data['illuminations'], color='#2979FF', label='Korelasi Iluminasi thdp
Elongasi', linewidth=2.5)
        ax.set_title(f"Iluminasi sbg Fungsi Elongasi: {date_display}", color='white', fontsize=14,
fontweight='bold', pad=15)
        ax.set_ylabel("Iluminasi (%)", color='ffffff', fontsize=12)
        ax.set_xlabel("Elongasi (°)", color='ffffff', fontsize=12)

        # Dinamis limit berdasarkan data
        min_e, max_e = min(data['elongations']), max(data['elongations'])
        ax.set_xlim(min_e - (max_e-min_e)*0.05, max_e + (max_e-min_e)*0.05)

    # --- COMMON STYLING ---
    ax.tick_params(colors='ffffff', labels=11)
    ax.grid(True, color='ffffff', linestyle=':', alpha=0.3)
    ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='best')
    fig.tight_layout()

    # Render Matplotlib ke Tkinter
    self.chart_ax = ax
    self.chart_canvas = FigureCanvasTkAgg(fig, master=self.frame_chart_out)

    # --- INISIALISASI TOOLTIP HOVER CANGGIH ---

```



```

self.annot_chart = ax.annotate(
    "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
    bbox=dict(boxstyle="round,pad=0.6", fc="#1E1E1E", ec="#00E5FF", lw=1.5, alpha=0.9),
    color="white", fontsize=10, zorder=10
)
self.annot_chart.set_visible(False)
self.chart_canvas.mpl_connect("motion_notify_event", self.on_hover_chart)

self.chart_canvas.draw()
self.chart_canvas.get_tk_widget().pack(fill="both", expand=True)

self.lbl_status.configure(text="Grafik Berhasil Dimuat", text_color="#00E676")
self.btn_hitung.configure(state="normal")

except Exception as e:
    import traceback
    self._force_show_error(f"Gagal Merender Grafik
Matplotlib:\n{str(e)}\n{traceback.format_exc()}")

def on_hover_chart(self, event):
    """Sistem Interpolasi Hover Interaktif yang dinamis berdasarkan jenis chart"""
    if event.inaxes == getattr(self, 'chart_ax', None):
        x_val = event.xdata
        y_val = event.ydata
        if x_val is None or y_val is None: return

        data = getattr(self, 'chart_data', None)
        if not data: return

        chart_type = self.chart_type
        import numpy as np

        if chart_type in ["altitude", "elongation", "illumination"]:
            if x_val < 0 or x_val > 24:
                self.annot_chart.set_visible(False)
                self.chart_canvas.draw_idle()
                return

            # Format waktu
            jam = int(x_val)
            sisa_menit = (x_val - jam) * 60.0
            menit = int(sisa_menit)
            detik = int(round((sisa_menit - menit) * 60.0))
            if detik >= 60: menit += 1; detik -= 60
            if menit >= 60: jam += 1; menit -= 60
            time_str = f"{jam:02d}:{menit:02d}:{detik:02d}"

            hours = data['hours']

```

```

if chart_type == "altitude":
    c_sun = np.interp(x_val, hours, data['sun_alts'])
    c_moon = np.interp(x_val, hours, data['moon_alts'])
    c_geo = np.interp(x_val, hours, data['moon_geo_alts'])
    text = (f" 🕒 Waktu Lokal : {time_str}\n"
            f" 🌞 Alt Matahari : {c_sun:+.2f}° (Topo)\n"
            f" 🌙 Alt Bulan : {c_moon:+.2f}° (Topo)\n"
            f" 🌍 Alt Bln (Geo): {c_geo:+.2f}° (KHGT)")

elif chart_type == "elongation":
    c_elong = np.interp(x_val, hours, data['elongations'])
    text = (f" 🕒 Waktu Lokal : {time_str}\n"
            f" 📏 Elongasi (Geo) : {c_elong:.2f}°")

elif chart_type == "illumination":
    c_ill = np.interp(x_val, hours, data['illuminations'])
    text = (f" 🕒 Waktu Lokal : {time_str}\n"
            f" 💡 Iluminasi Fase : {c_ill:.2f}%")

elif chart_type == "illum_vs_elong":
    min_e, max_e = min(data['elongations']), max(data['elongations'])
    if x_val < min_e or x_val > max_e:
        self.annot_chart.set_visible(False)
        self.chart_canvas.draw_idle()
        return

    # Pengurutan data penting agar np.interp tidak error pada data X
    x_arr = np.array(data['elongations'])
    y_arr = np.array(data['illuminations'])
    sort_idx = np.argsort(x_arr)

    c_ill = np.interp(x_val, x_arr[sort_idx], y_arr[sort_idx])
    text = (f" 📏 Elongasi : {x_val:.2f}°\n"
            f" 💡 Iluminasi: {c_ill:.2f}%")

    self.annot_chart.xy = (x_val, y_val)
    self.annot_chart.set_text(text)
    self.annot_chart.set_visible(True)
    self.chart_canvas.draw_idle()
else:
    if getattr(self, 'annot_chart', None) and self.annot_chart.get_visible():
        self.annot_chart.set_visible(False)
        self.chart_canvas.draw_idle()

def _force_show_error(self, err_msg):
    """Memaksa textbox muncul kembali jika terjadi error di mode grafik"""
    if hasattr(self, 'frame_chart_out'):

```

```

        self.frame_chart_out.grid_remove()
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.display_error(err_msg)

# =====
# MODUL 28: WINAI (AI CHATBOT ASSISTANT)
# =====
def setup_winai_ui(self):
    # --- VARIABEL & API KEY WINAI (DIRECT GEMINI API) ---
    self.winai_api_keys = [
        'AlzaSyAVIBjSEWKdyeAYnhYue6Zw6tSJKzQpfEw',
        'AlzaSyB8jOg5MPMrbpIWtMRT51rydJYLMFhRaXs',
        'AlzaSyD2UOrD7VaYU21FI6z8Ps1N4oEh86IrvJ4', 'AlzaSyCS_HjbzrEN5ddevYNjSvbw0K4oONBOHuY',
        'AlzaSyBGRz7pausR71hjj_WyVh2FxsqVZedj3dc'
    ]
    self.winai_primary_model = 'gemini-2.5-flash'
    self.winai_tracker_file = 'last_successful_index_winai.txt'
    self.winai_db_items = []
    self.winai_db_loaded = False

    self.winai_jawaban_terakhir = ""
    self.winai_pertanyaan_terakhir = ""
    self.winai_memori = ""
    self.winai_instruksi = ""
    self.winai_data_kal = ""

    # Muat database lokal di background
    threading.Thread(target=self.load_winai_databases, daemon=True).start()

    # ===== SIDEBAR INPUT (KIRI) =====
    self.frame_winai_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

    winai_info = ctk.CTkFrame(self.frame_winai_inputs, fg_color="#212121")
    winai_info.pack(fill="x", padx=15, pady=10)
    ctk.CTkLabel(winai_info, text="WINAI ASSISTANT", font=("Segoe UI", 12, "bold"),
text_color="#00E5FF").pack(anchor="w", padx=10, pady=(8, 2))

    winai_date_frame = ctk.CTkFrame(winai_info, fg_color="transparent")
    winai_date_frame.pack(fill="x", padx=10, pady=(0, 10))

    now = datetime.datetime.now()
    self.winai_entry_y, self.winai_entry_m, self.winai_entry_d =
self.create_ymd_row(winai_date_frame, str(now.year), f"{now.month:02d}", f"{now.day:02d}")

    self.winai_input_entry = ctk.CTkEntry(winai_info, placeholder_text="Ketik pertanyaan...",
font=("Segoe UI", 13), height=40)

```

```

self.winai_input_entry.pack(fill="x", padx=10, pady=10)
self.winai_input_entry.bind("<Return>", lambda e: self.proses_pertanyaan_winai())

self.btn_winai_kirim = ctk.CTkButton(winai_info, text="Kirim Pertanyaan", font=("Segoe UI", 12,
"bold"), fg_color="#1565C0", hover_color="#0D47A1", height=35,
command=self.proses_pertanyaan_winai)
self.btn_winai_kirim.pack(fill="x", padx=10, pady=(0, 5))

self.btn_winai_reset = ctk.CTkButton(winai_info, text="🔄 Reset Memori", font=("Segoe UI", 12,
"bold"), fg_color="#dc3545", hover_color="#c82333", height=35, command=self.reset_memori_winai)
self.btn_winai_reset.pack(fill="x", padx=10, pady=(5, 10))

# ===== MAIN FRAME OUTPUT (KANAN) =====
self.frame_winai_out = ctk.CTkFrame(self.main_frame, fg_color="#050510", corner_radius=10)

header_winai = ctk.CTkFrame(self.frame_winai_out, fg_color="#181818", corner_radius=8)
header_winai.pack(fill="x", padx=15, pady=10)

ctk.CTkLabel(header_winai, text="🤖 Ruang Interaksi AI", font=("Segoe UI", 16, "bold"),
text_color="#00E5FF").pack(side="left", padx=15, pady=10)

self.btn_winai_salin = ctk.CTkButton(header_winai, text="📄 Salin", fg_color="#28a745",
hover_color="#218838", width=80, command=self.salin_teks_winai)
self.btn_winai_salin.pack(side="right", padx=(5, 15), pady=10)
self.btn_winai_berbagi = ctk.CTkButton(header_winai, text="🔗 Share", fg_color="#17a2b8",
hover_color="#138496", width=80, command=self.berbagi_teks_winai)
self.btn_winai_berbagi.pack(side="right", padx=5, pady=10)
self.btn_winai_buka = ctk.CTkButton(header_winai, text="📄 Log Teks", fg_color="#6c757d",
hover_color="#5a6268", width=80, command=self.buka_output_winai)
self.btn_winai_buka.pack(side="right", padx=5, pady=10)

# Ruang Chat - Background Putih & Teks Hitam Standar Dokumen
self.winai_output_box = ctk.CTkTextbox(self.frame_winai_out, wrap="word", fg_color="FFFFFF",
text_color="000000", font=("Segoe UI", 16))
self.winai_output_box.pack(fill="both", expand=True, padx=15, pady=(0, 15))

font_nama = "Segoe UI"
ukuran_normal = 16

tb = self.winai_output_box._textbox
tb.tag_configure("info", foreground="#856404", background="#FFF3CD", font=(font_nama, 14,
"italic"), spacing1=5, spacing3=10, justify="center")
tb.tag_configure("info_internet", foreground="#0056b3", background="#e7f1ff", font=(font_nama,
14), lmargin1=15, lmargin2=15, spacing1=5, spacing3=10, borderwidth=1, relief="solid")
tb.tag_configure("pembuka", foreground="000000", font=(font_nama, 18, "bold"), spacing1=0,
spacing2=0, spacing3=5, justify="left")

```



```

        tb.tag_configure("user_header", foreground="#0056b3", font=(font_nama, 16, "bold"),
spacing1=15)
        tb.tag_configure("user_teks", background="#E3F2FD", foreground="#000000", font=(font_nama,
ukuran_normal), lmargin1=10, lmargin2=10, spacing1=5, spacing3=10)
        tb.tag_configure("ai_header", foreground="#28A745", font=(font_nama, 17, "bold"), spacing1=15)
        tb.tag_configure("normal", font=(font_nama, ukuran_normal), foreground="#000000",
justify="left")
        tb.tag_configure("bold", font=(font_nama, ukuran_normal, "bold"), foreground="#000000",
justify="left")
        tb.tag_configure("paragraf", spacing1=5, spacing3=10, lmargin1=10, lmargin2=10, justify="left")
        tb.tag_configure("kutipan", font=(font_nama, ukuran_normal, "italic"), lmargin1=30, lmargin2=30,
foreground="#333333", background="#F8F9FA")
        tb.tag_configure("kode", font=("Consolas", 14), foreground="#000000", background="#E8F5E9",
spacing1=2, spacing3=2, lmargin1=15, lmargin2=15)

```

```

pesan_pembuka = (
    "Selamat datang di WinAI: Aplikasi AI Windows V8.0.\n"
    "Terintegrasi penuh dengan KHGT Times Engine dan Database Tarjih Muhammadiyah.\n"
)
self.winai_output_box.insert("0.0", pesan_pembuka, "pembuka")
self.winai_output_box.configure(state="disabled")

```

```

def load_winai_databases(self):
    try:
        base_url = "https://hisabmu.com/aifikih/db_fikih_tarjih"
        res = requests.get(f"{base_url}.json", timeout=10)
        if res.status_code == 200: self.winai_db_items.extend(res.json())
        for i in range(2, 21):
            try:
                res = requests.get(f"{base_url}_{i}.json", timeout=5)
                if res.status_code == 200: self.winai_db_items.extend(res.json())
                else: break
            except: break
        res_txt = requests.get("https://hisabmu.com/aifikih/tanyajawabagama.txt", timeout=10)
        if res_txt.status_code == 200: self.winai_db_items.extend(res_txt.json())
        self.winai_db_loaded = True
    except:
        pass

```

```

def retrieve_winai_context(self, user_question):
    if not self.winai_db_loaded: return "Database belum siap."
    clean_text = re.sub(r'[^a-zA-Z0-9]', "", user_question.lower())
    words = clean_text.split()
    keywords = [w for w in words if len(w) > 3]

    scored_items = []
    for item in getattr(self, 'winai_db_items', []):
        judul = item.get('judul', "")

```

```

isi = item.get('isi_konten', '')
text = f"Q: {judul}\nA: {isi}\n\n"
text_lower = text.lower()
score = sum(1 for kw in keywords if kw in text_lower)
if score > 0: scored_items.append({'score': score, 'text': text})

scored_items.sort(key=lambda x: x['score'], reverse=True)
internal_db_text = ""
for si in scored_items:
    if len(internal_db_text) + len(si['text']) > 80000: break
    internal_db_text += si['text']
return internal_db_text if internal_db_text.strip() else "Tidak ada referensi lokal yang relevan."

def sisipkan_teks_winai(self, teks, pengirim="AI"):
    self.winai_output_box.configure(state="normal")
    if pengirim == "Anda":
        self.winai_pertanyaan_terakhir = teks
        self.winai_output_box.insert("end", "👤 Anda:\n", "user_header")
        self.winai_output_box.insert("end", f" {teks} \n\n", "user_teks")
    elif pengirim == "Sistem":
        self.winai_output_box.insert("end", "🕒 " + teks + "\n\n", ("info", "status_loading"))
    else:
        self.winai_output_box.insert("end", "🤖 WinAI:\n", "ai_header")
        self.winai_jawaban_terakhir = teks

    try:
        with open("output_winai.txt", "a", encoding="utf-8") as file_out:
            waktu_sekarang = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            file_out.write(f"=== TANGGAL: {waktu_sekarang} ===\nPERTANYAAN: {self.winai_pertanyaan_terakhir}\nJAWABAN:\n{teks}\n{' '*50}\n\n")
        except Exception: pass

    # --- PEMBERSIHAN DAN FORMAT DOKUMEN STANDAR ---
    text_cleaned = re.sub(r'(?i)Asisten Fikih ini.*', '', teks, flags=re.DOTALL).strip()

    lines = text_cleaned.split('\n')
    in_code_block = False # <-- TAMBAHAN: Deteksi Blok Kode

    for i, line in enumerate(lines):
        # Deteksi markdown code block (```)
        if line.strip().startswith("```"):
            in_code_block = not in_code_block
            continue # Lewati baris ``` nya sendiri

        # Jika sedang di dalam blok kode, format sebagai Consolas murni
        if in_code_block:
            self.winai_output_box.insert("end", line + "\n", "kode")

```

```

        continue

    # Format khusus blok kode (kalkulasi) versi lama
    if (line.startswith(" ") and "|" in line) or line.startswith("===") or line.startswith("---"):
        self.winai_output_box.insert("end", line + "\n", "kode")
    # Deteksi Heading/Judul (#, ##, ###) untuk ditebalkan tanpa menampilkan '#'
    elif re.match(r'^\s*#+\s+', line):
        clean_line = re.sub(r'^\s*#+\s+', "", line)
        self.winai_output_box.insert("end", clean_line + "\n", "bold")
    else:
        # Deteksi teks tebal (**)
        parts = line.split('**')
        for j, part in enumerate(parts):
            if j % 2 == 1:
                self.winai_output_box.insert("end", part, "bold")
            else:
                # Hapus simbol aneh, biarkan teks normal
                part_clean = part.replace('*', '')
                self.winai_output_box.insert("end", part_clean, "normal")

    # Baris baru
    if i < len(lines) - 1:
        self.winai_output_box.insert("end", "\n", "normal")

    self.winai_output_box.insert("end", "\n\n" + "="*80 + "\n\n", "normal")

    self.winai_output_box.see("end")
    self.winai_output_box.configure(state="disabled")

def tampilkan_jejak_internet(self, teks):
    self.winai_output_box.configure(state="normal")
    self.winai_output_box.insert("end", "🌐 BERHASIL MENDAPATKAN DATA INTERNET  
TERBARU:\n\n" + teks + "\n", "info_internet")
    self.winai_output_box.see("end")
    self.winai_output_box.configure(state="disabled")

def hapus_status_loading_winai(self):
    self.winai_output_box.configure(state="normal")
    ranges = self.winai_output_box.tag_ranges("status_loading")
    if ranges:
        self.winai_output_box.delete(ranges[0], ranges[-1])
    self.winai_output_box.configure(state="disabled")

def proses_pertanyaan_winai(self):
    pertanyaan = self.winai_input_entry.get().strip()
    if not pertanyaan: return

```

```

self.winai_input_entry.delete(0, "end")
self.sisipkan_teks_winai(pertanyaan, pengirim="Anda")

teks_lower = pertanyaan.lower()

# =====
# ---> SMART INTERCEPTOR (AGENTIC AI CONTROL) <---
# =====
# AI mendeteksi niat (intent) pengguna dari kata kerjanya
is_perintah = any(kata in teks_lower for kata in ["hitung", "kalkulasi", "berapa", "cek", "cari",
"tampilkan", "simulasi", "buka"])

# Deteksi Objek / Modul Sasaran
is_tanya_hilal = is_perintah and ("visibilitas" in teks_lower or "hilal" in teks_lower)
is_tanya_kota = is_perintah and any(kata in teks_lower for kata in ["kota pertama", "titik pertama",
"awal bulan khgt"])
is_tanya_okultasi = is_perintah and ("okultasi" in teks_lower or "menutupi" in teks_lower)
is_tanya_gerhana = is_perintah and "gerhana" in teks_lower
is_tanya_pasang = is_perintah and any(kata in teks_lower for kata in ["pasang surut", "air laut",
"gravitasi"])
is_tanya_fajar = is_perintah and any(kata in teks_lower for kata in ["fajar", "shadiq", "sqm",
"kegelapan langit"])
is_tanya_animasi = is_perintah and any(kata in teks_lower for kata in ["animasi", "live", "posisi real-
time"])
is_tanya_3d = is_perintah and ("3d" in teks_lower or "tata surya" in teks_lower)

# Matikan tombol kirim sementara proses komputasi berlangsung
self.btn_winai_kirim.configure(state="disabled")

# --- EKSEKUSI INTERCEPTOR ROUTING ---
if is_tanya_hilal:
    self.sisipkan_teks_winai("Menjalankan Modul 1 (Kalkulasi Visibilitas) secara background...",
pengirim="Sistem")
    threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "hilal"),
daemon=True).start()

elif is_tanya_kota:
    self.sisipkan_teks_winai("Menjalankan Modul 5 (Pelacakan Kota Pertama) secara background...",
pengirim="Sistem")
    threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "kota"),
daemon=True).start()

elif is_tanya_okultasi:
    self.sisipkan_teks_winai("Memicu Prediktor Okultasi di layar utama...", pengirim="Sistem")
    # Auto-pindah UI ke Menu 35
    self.combo_mode.set("35) Astrofotografi & Kontras Visibilitas")
    self.switch_mode("35) Astrofotografi & Kontras Visibilitas")
    self.calculate_occultation() # Eksekusi grafik okultasi

```

```

        # AI tetap menjawab teorinya di latar belakang
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Tolong jelaskan secara teori
astronomi apa itu fenomena okultasi.", None), daemon=True).start()

    elif is_tanya_gerhana:
        self.sisipkan_teks_winai("Memicu Scanner Gerhana Global di layar utama...", pengirim="Sistem")
        # Auto-pindah UI ke Menu 14
        self.combo_mode.set("14) Analisis Gerhana")
        self.switch_mode("14) Analisis Gerhana")
        tahun = int(self.combo_tahun_gerhana.get())
        threading.Thread(target=self._calc_gerhana_thread, args=(tahun,), daemon=True).start()
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Berikan edukasi tentang
mekanisme gerhana.", None), daemon=True).start()

    elif is_tanya_pasang:
        self.sisipkan_teks_winai("Memicu Modul Pasang Surut Gravitasi...", pengirim="Sistem")
        self.combo_mode.set("36) Prediktor Pasang Surut Gravitasi")
        self.switch_mode("36) Prediktor Pasang Surut Gravitasi")
        self.calculate_pasang_surut()
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()

    elif is_tanya_fajar:
        self.sisipkan_teks_winai("Memicu Modul Evaluasi Fajar & SQM...", pengirim="Sistem")
        self.combo_mode.set("37) Evaluasi Fajar Shadiq & SQM")
        self.switch_mode("37) Evaluasi Fajar Shadiq & SQM")
        self.calculate_fajar_sqm()
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()

    elif is_tanya_animasi:
        self.sisipkan_teks_winai("Membuka Modul Live Animasi...", pengirim="Sistem")
        self.combo_mode.set("15) Live Animasi")
        self.switch_mode("15) Live Animasi")
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Katakan: Saya telah
membuka layar Live Animasi untuk Anda di layar utama.", None), daemon=True).start()

    elif is_tanya_3d:
        self.sisipkan_teks_winai("Membuka Modul Simulasi Tata Surya 3D...", pengirim="Sistem")
        self.combo_mode.set("19) Simulasi Ephemeris 3D")
        self.switch_mode("19) Simulasi Ephemeris 3D")
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Katakan: Sistem Tata Surya
3D telah disiapkan untuk Anda.", None), daemon=True).start()

    else:
        # Pertanyaan umum fikh/astronomi biasa (Tanpa pindah layar)
        self.sisipkan_teks_winai("Mengakses Database & Menghubungi AI...", pengirim="Sistem")
        threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()

def hitung_dan_tanya_ai(self, pertanyaan, mode):

```

```

"""Fungsi yang memproses hitungan terlebih dahulu, lalu mengirim datanya ke AI untuk
diinterpretasi"""
report_data = ""
if mode == "hilal":
    try:
        y, m, d = self.winai_entry_y.get(), self.winai_entry_m.get(), self.winai_entry_d.get()
        # Sinkronkan dengan menu 1
        self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, y)
        self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, m)
        self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, d)
        report_data = self.generate_visibility_report()
    except Exception as e:
        report_data = f"Gagal menghitung visibilitas: {e}"
elif mode == "kota":
    try:
        y, m, d = int(self.winai_entry_y.get()), int(self.winai_entry_m.get()),
int(self.winai_entry_d.get())
        report_data = self.generate_first_point_report(y, m, d)
    except Exception as e:
        report_data = f"Gagal melacak kota pertama: {e}"

# Teruskan ke AI dengan membawa data hasil hitungan
self.tanya_ai_thread(pertanyaan, report_tambahan=report_data)

def tanya_ai_thread(self, teks_pertanyaan, report_tambahan=None):
    self.after(0, lambda: self.sisipkan_teks_winai("Membaca instruksi & Database Internal...",
pengirim="Sistem"))

    try:
        headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"}
        resp_sys = requests.get("https://hisabmu.com/aifikih/system_prompt.php", headers=headers,
timeout=15)
        resp_sys.encoding = 'utf-8'
        if resp_sys.status_code == 200 and "$systemPersona" not in resp_sys.text:
            self.winai_instruksi = resp_sys.text.strip()
        else:
            self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447
H."
    except:
        self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447 H."

    if not self.winai_data_kal:
        try:
            resp_cal = requests.get(CALENDAR_DATA_URL, headers=headers, timeout=10)
            if resp_cal.status_code == 200: self.winai_data_kal = resp_cal.text.strip()
        except: pass

    teks_bersih = teks_pertanyaan.strip()

```

```

gabungan_internet = ""

if not teks_bersih.startswith("/"):
    self.after(0, lambda: self.sisipkan_teks_winai("Menelusuri informasi pendukung di internet...",
pengirim="Sistem"))
    hasil_google = ""
    try:
        kata_kunci = urllib.parse.quote(teks_bersih)
        url_g =
f"https://www.google.com/search?&lr=lang_id&hl=id&nem=143&udm=50&q={kata_kunci}"
        resp_g = requests.get(url_g, headers=headers, cookies=GOOGLE_SESSION_COOKIES,
timeout=15)
        if resp_g.status_code == 200:
            soup = BeautifulSoup(resp_g.text, 'html.parser')
            snippets = soup.find_all(['div', 'span'], class_='BNeawe', 'VwiC3b', 's3v9rd', 'HwtZe'])
            kumpulan = []
            for s in snippets:
                t = s.get_text(strip=True)
                if t and len(t) > 60 and t not in kumpulan: kumpulan.append(t)
            if kumpulan: hasil_google = "[Data Referensi Google Search]:\n" + "\n".join(f"- {txt}" for txt in
kumpulan[:3]) + "\n"
        except: pass

    hasil_ddg = ""
    try:
        with DDGS() as ddgs:
            results = list(ddgs.text(teks_bersih, max_results=2))
            if results: hasil_ddg = "[Data Referensi DuckDuckGo]:\n" + "\n".join(f"- {r['body']}" for r in
results) + "\n"
        except: pass

    if hasil_google or hasil_ddg:
        gabungan_internet = f"{hasil_google}\n{hasil_ddg}".strip()
        self.after(0, self.tampilkan_jejak_internet, gabungan_internet)

# RAG / Local DB
konteks_lokal = self.retrieve_winai_context(teks_bersih)

self.after(0, lambda: self.sisipkan_teks_winai("Menganalisis pertanyaan dan memanggil Model AI
Gemini...", pengirim="Sistem"))

loc_prov = self.opt_prov.get()
loc_city = self.opt_city.get()
win_y = self.winai_entry_y.get()
win_m = self.winai_entry_m.get()
win_d = self.winai_entry_d.get()

```

```

        prompt_pintar = f"=== KONTEKS PENGGUNA ===\nLokasi: {loc_city}, {loc_prov}.\nTanggal Tinjauan:
{win_d}/{win_m}/{win_y}.\nWaktu Aktual: {datetime.datetime.now().strftime('%A, %d %B %Y
%H:%M:%S')}\n\n"
        if self.winai_data_kal: prompt_pintar += f"=== DATA KALENDER KHGT
===\n{self.winai_data_kal}\n\n"
        if gabungan_internet: prompt_pintar += f"=== REFERENSI INTERNET
===\n{gabungan_internet}\n\n"
        if self.winai_memori: prompt_pintar += f"=== RIWAYAT OBROLAN LALU
===\n{self.winai_memori}\n\n"

        # Injeksi Data Report Sistem (Bila Ada Perintah Hitung)
        if report_tambahan:
            prompt_pintar += f"=== HASIL KALKULASI SISTEM SAAT INI ===\n{report_tambahan}\n\n"

        prompt_pintar += f"=== PERTANYAAN USER ===\n{teks_bersih}\n\n"

        # --- PERBAIKAN ATURAN: Melarang AI menyalin ulang tabel ---
        aturan = "ATURAN MENJAWAB:\n1. Jawab pertanyaan user secara luwes, ramah, dan solutif.
Jangan kaku.\n2. JANGAN gunakan simbol markdown kotor (* atau #) di tengah kalimat, susun paragraf
dengan rapi.\n3. Jika ada 'HASIL KALKULASI SISTEM' yang diberikan di atas, Anda CUKUP menjelaskan
kesimpulannya saja secara ringkas. DILARANG KERAS menyalin atau mengetik ulang tabel laporannya,
karena UI antarmuka sistem akan melampirkannya secara otomatis di layar."

        final_instruction = self.winai_instruksi + f"\n\n===== \nREFERENSI DATABASE
INTERNAL LOKAL:\n\"\"\"{konteks_lokal}\"\"\" \n===== \n{aturan}"

        payload = {
            "systemInstruction": [{"text": final_instruction}],
            "contents": [{"role": "user", "parts": [{"text": prompt_pintar}]}],
            "generationConfig": {"temperature": 0.2, "topP": 0.8}
        }

        start_index = 0
        if os.path.exists(self.winai_tracker_file):
            with open(self.winai_tracker_file, 'r') as f:
                start_index = int(f.read().strip() or 0)

        total_keys = len(self.winai_api_keys)
        jawaban = ""
        sukses_dijawab = False
        last_error = ""

        for i in range(total_keys):
            idx = (start_index + i) % total_keys
            api_key = self.winai_api_keys[idx]
            api_url =
f"https://generativelanguage.googleapis.com/v1beta/models/{self.winai_primary_model}:generateCont
ent?key={api_key}"

```



```

try:
    res = requests.post(api_url, headers={'Content-Type': 'application/json'}, json=payload,
timeout=20)
    if res.status_code == 200:
        data = res.json()
        jawaban = data['candidates'][0]['content']['parts'][0]['text']
        with open(self.winai_tracker_file, 'w') as f:
            f.write(str(idx))
        sukses_dijawab = True
        break
    else:
        err = res.json()
        last_error = err.get('error', {}).get('message', f'HTTP {res.status_code}')

        # --- PERBAIKAN 2: Logika Rotasi Anti-Macet ---
        # Jika error 429 (Quota Habis) atau 403 (Key Mati) atau 400 (Bad Request),
        # ABAIKAN dan LANGSUNG LANJUT uji API Key berikutnya!
        if res.status_code in [429, 403, 400]:
            continue

        # Berhenti total HANYA JIKA model benar-benar tidak ditemukan (404)
        if res.status_code == 404:
            break

except requests.exceptions.RequestException as e:
    last_error = str(e)
    continue

if not sukses_dijawab:
    jawaban = f" ⚠️ API Bermasalah atau Koneksi Gagal.\nDetail: {last_error}"

    # --- TAMBAHAN PERBAIKAN: Menampilkan hisab walau AI Gagal ---
    if report_tambahan:
        jawaban += f"\n\nNamun, sistem HISAB KHGT tetap berhasil melakukan kalkulasi untuk\nAnda:\n\n``text\n{report_tambahan}\n``"

    if sukses_dijawab:
        # --- INJEKSI OTOMATIS LAPORAN MURNI DI BELAKANG JAWABAN AI ---
        if report_tambahan:
            jawaban += f"\n\n[Laporan Asli Sistem]:\n\n``text\n{report_tambahan}\n``"

        if len(jawaban) < 2000:
            self.winai_memori += f"User: {teks_bersih}\nAI: {jawaban}\n\n"
        else:
            self.winai_memori += f"User: {teks_bersih}\nAI: [Telah memberikan jawaban\npanjang/dokumen]\n\n"

```

```

        if len(self.winai_memori) > 15000:
            self.winai_memori = self.winai_memori[-15000:]

        self.after(0, self.selesai_memproses_winai, jawaban)

    def selesai_memproses_winai(self, jawaban):
        self.hapus_status_loading_winai()
        self.sisipkan_teks_winai(jawaban, pengirim="AI")
        self.btn_winai_kirim.configure(state="normal")

    def salin_teks_winai(self):
        teks_salin = self.winai_output_box.get("1.0", "end-1c").strip()
        if not teks_salin:
            messagebox.showwarning("Peringatan", "Tidak ada teks untuk disalin.")
            return

        self.clipboard_clear()
        self.clipboard_append(teks_salin)
        messagebox.showinfo("Sukses", "Seluruh teks di layar berhasil disalin ke Clipboard!")

    def berbagi_teks_winai(self):
        teks_share = self.winai_output_box.get("1.0", "end-1c").strip()
        if not teks_share: return

        self.btn_winai_berbagi.configure(text="⌚ Memproses...", state="disabled")
        threading.Thread(target=self.proses_berbagi_thread, args=(teks_share,), daemon=True).start()

    def proses_berbagi_thread(self, teks_share):
        payload = {'action': 'share_content', 'text': teks_share}
        try:
            # Karena PHP Server WinAI tetap butuh endpoint untuk berbagi URL
            response = requests.post("https://hisabmu.com/aifikih/system_php.php", data=payload,
            timeout=15)
            if response.status_code == 200:
                data = response.json()
                if data.get('status') == 'success':
                    self.after(0, self.berbagi_sukses, data.get('url'))
                else:
                    self.after(0, self.berbagi_gagal, data.get('message', 'Gagal membuat link.'))
            else:
                self.after(0, self.berbagi_gagal, "Error koneksi.")
        except Exception as e:
            self.after(0, self.berbagi_gagal, str(e))

    def berbagi_sukses(self, url):
        self.btn_winai_berbagi.configure(text="🔗 Share", state="normal")

```

```

webbrowser.open(url)
self.clipboard_clear()
self.clipboard_append(url)
messagebox.showinfo("Sukses", f"Link berhasil dibuat!\n\n{url}")

def berbagi_gagal(self, pesan_error):
    self.btn_winai_berbagi.configure(text="🔗 Share", state="normal")
    messagebox.showerror("Gagal", f"Gagal membagikan teks:\n{pesan_error}")

def buka_output_winai(self):
    nama_file = "output_winai.txt"
    if not os.path.exists(nama_file):
        with open(nama_file, "w", encoding="utf-8") as f:
            f.write("=== LOG OUTPUT WinAI ===\n\n")
    try: os.startfile(nama_file)
    except Exception as e: messagebox.showerror("Error", f"Gagal membuka:\n{e}")

def reset_memori_winai(self):
    self.winai_memori = ""
    messagebox.showinfo("Reset Berhasil", "Ingatan dikosongkan. AI siap!")

def apply_kalender_kustom(self):
    try:
        thn = int(self.entry_kal_h_tahun.get())
        # ---> KEMBALIKAN KE BAHASA INDONESIA <---
        bln_str = self.reverse_tr(self.combo_kal_h_bulan.get())
        bln_idx = BULAN_HIJRIAH.index(bln_str)

        if thn in HIJRI_DB:
            self.cal_h_year = thn
            self.cal_h_month = bln_idx
            self.render_kalender()
        else:
            messagebox.showwarning("Peringatan Database", f"Data awal bulan untuk tahun {thn} H
belum tersedia di database internal.")
    except ValueError:
        messagebox.showerror("Error", "Tahun Hijriah harus berupa angka!")

def apply_kalmasehi_kustom(self):
    try:
        thn = int(self.entry_kal_m_tahun.get())
        # ---> KEMBALIKAN KE BAHASA INDONESIA <---
        bln_str = self.reverse_tr(self.combo_kal_m_bulan.get())
        bln_idx = BULAN_MASEHI.index(bln_str) + 1

        self.cal_m_year = thn

```

```

        self.cal_m_month = bln_idx
        self.render_kalmasehi()
    except ValueError:
        messagebox.showerror("Error", "Tahun Masehi harus berupa angka!")

# =====
# EKPANSI MODUL 14: SIMULATOR VISUAL GERHANA (MATAHARI & BULAN)
# =====
def buka_simulator_gerhana(self):
    selected_item = self.tabel_gerhana.selection()
    if not selected_item:
        messagebox.showwarning("Peringatan", "Silakan klik/pilih salah satu jadwal gerhana di tabel
terlebih dahulu.")
        return

    item_values = self.tabel_gerhana.item(selected_item[0])['values']
    if not item_values or item_values[0] == "": return

    objek = str(item_values[0]).strip()
    jenis = str(item_values[1]).strip()
    waktu_puncak_str = str(item_values[3]).strip()
    wilayah_global = str(item_values[5]).strip() # Mengambil data wilayah terdampak
    if not waktu_puncak_str: return

    # Buka Jendela TopLevel
    self.win_sim = ctk.CTkToplevel(self)
    self.win_sim.title(f"Telescope View - Simulasi Gerhana {objek}")
    self.win_sim.geometry("600x740")
    self.win_sim.attributes("-topmost", True)
    self.win_sim.configure(fg_color="#000000")

    # Header UI
    header_frame = ctk.CTkFrame(self.win_sim, fg_color="transparent")
    header_frame.pack(fill="x", pady=10)
    ctk.CTkLabel(header_frame, text=f"SIMULASI GERHANA {objek.upper()}", font=("Segoe UI", 16,
"bold"), text_color="#FFD54F").pack()

    # ---> PERBAIKAN: Menambahkan wraplength agar teks wilayah panjang bisa rapi <---
    self.lbl_sim_info = ctk.CTkLabel(header_frame, text="Mengkalkulasi parameter astronomis...",
font=("Consolas", 12), text_color="#00E676", wraplength=550)
    self.lbl_sim_info.pack()

    # Canvas Simulator
    self.sim_canvas = tk.Canvas(self.win_sim, bg="#050510", width=500, height=500,
highlightthickness=1, highlightbackground="#333333")
    self.sim_canvas.pack(pady=10)

    # Frame Kontrol Animasi

```

```

ctrl_frame = ctk.CTkFrame(self.win_sim, fg_color="transparent")
ctrl_frame.pack(fill="x", padx=30, pady=10)

self.sim_slider = ctk.CTkSlider(ctrl_frame, from_=0, to=100, command=self._update_sim_frame)
self.sim_slider.pack(fill="x", pady=5)
self.sim_slider.set(0)

btn_play = ctk.CTkButton(ctrl_frame, text="▶ Play / Pause Animasi", fg_color="#1565C0",
hover_color="#0D47A1", command=self._toggle_sim_play)
btn_play.pack(pady=5)

self.sim_data = []
self.sim_is_playing = False

# Mulai Kalkulasi di Background Thread (Kirim wilayah_global juga)
threading.Thread(target=self._kalkulasi_data_simulator, args=(waktu_puncak_str, objek,
wilayah_global), daemon=True).start()

def _kalkulasi_data_simulator(self, waktu_puncak_str, objek, wilayah_global):
    try:
        from skyfield.api import wgs84
        tz_wib = pytz.timezone('Asia/Jakarta')
        dt_naive = datetime.datetime.strptime(waktu_puncak_str, "%d-%m-%Y %H:%M")
        dt_wib = tz_wib.localize(dt_naive)
        t_peak = self.ts.from_datetime(dt_wib)

    try:
        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        elev = float(self.entry_velev.get())
        tz = float(self.entry_vtz.get())
    except:
        lat, lon, elev, tz = -7.0667, 110.4100, 230.0, 7.0

    earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
    lokasi_awal = earth + wgs84.latlon(lat, lon, elevation_m=elev)

    # Kalkulasi +/- 4 Jam dari puncak (agar cover seluruh fase)
    tt_start = t_peak.tt - (4.0 / 24.0)
    tt_end = t_peak.tt + (4.0 / 24.0)
    tt_array = np.linspace(tt_start, tt_end, 400)

    # ---> FUNGSI PENGAMAN AREA MATH.ACOS AGAR TIDAK ERROR <---
    def calc_obscuratation(sep, r_moon, r_target):
        if sep >= (r_target + r_moon): return 0.0
        if sep <= abs(r_target - r_moon):
            return 100.0 if r_moon >= r_target else (r_moon**2 / r_target**2) * 100.0

```

```

r, R, d = r_moon, r_target, sep
# Clamping domain arccos ke [-1.0, 1.0] agar tidak crash karena floating point error
val1 = max(-1.0, min(1.0, (d**2 + r**2 - R**2) / (2 * d * r)))
val2 = max(-1.0, min(1.0, (d**2 + R**2 - r**2) / (2 * d * R)))

part1 = r**2 * math.acos(val1)
part2 = R**2 * math.acos(val2)
part3 = 0.5 * math.sqrt(max(0.0, (-d + r + R) * (d + r - R) * (d - r + R) * (d + r + R)))

return ((part1 + part2 - part3) / (math.pi * R**2)) * 100.0

def generate_frames(target_lokasi, is_geocentric=False):
    frames = []
    if "Matahari" in objek:
        for tt in tt_array:
            t_frame = self.ts.tt_jd(tt)

            if is_geocentric:
                s_app = earth.at(t_frame).observe(sun).apparent()
                m_app = earth.at(t_frame).observe(moon).apparent()

                s_ra, s_dec, s_dist = s_app.radec()
                m_ra, m_dec, m_dist = m_app.radec()

                sd_sun = math.degrees(math.asin(696000.0 / s_dist.km))
                sd_moon = math.degrees(math.asin(1737.4 / m_dist.km))

                d_dec = m_dec.degrees - s_dec.degrees
                d_ra_raw = (m_ra.hours - s_ra.hours + 12.0) % 24.0 - 12.0
                d_ra = d_ra_raw * 15.0 * math.cos(s_dec.radians)

                sep = s_app.separation_from(m_app).degrees
                obs_pct = calc_obscurtion(sep, sd_moon, sd_sun)

                dt_lokal = t_frame.utc_datetime() + datetime.timedelta(hours=tz)
                frames.append({
                    'tipe': 'Matahari', 'waktu': dt_lokal.strftime("%H:%M:%S"),
                    'd_x': d_ra, 'd_y': d_dec, 'sd_sun': sd_sun, 'sd_moon': sd_moon,
                    'obs': obs_pct, 'sep': sep, 'alt_matahari': 90.0 # Pseudo-altitude
                })
            else:
                s_app = target_lokasi.at(t_frame).observe(sun).apparent()
                m_app = target_lokasi.at(t_frame).observe(moon).apparent()

                s_alt, s_az, s_dist = s_app.altaz()
                m_alt, m_az, m_dist = m_app.altaz()

```

```

sd_sun = math.degrees(math.asin(696000.0 / s_dist.km))
sd_moon = math.degrees(math.asin(1737.4 / m_dist.km))

d_alt = m_alt.degrees - s_alt.degrees
d_az_raw = (m_az.degrees - s_az.degrees + 180.0) % 360.0 - 180.0
d_az = d_az_raw * math.cos(math.radians(s_alt.degrees))

sep = s_app.separation_from(m_app).degrees
obs_pct = calc_obscurations(sep, sd_moon, sd_sun)

dt_lokal = t_frame.utc_datetime() + datetime.timedelta(hours=tz)
frames.append({
    'tipe': 'Matahari', 'waktu': dt_lokal.strftime("%H:%M:%S"),
    'd_x': d_az, 'd_y': d_alt, 'sd_sun': sd_sun, 'sd_moon': sd_moon,
    'obs': obs_pct, 'sep': sep, 'alt_matahari': s_alt.degrees
})
else:
    # GERHANA BULAN (Selalu Geosentrik karena bayangan bumi berpusat di sana)
    for tt in tt_array:
        t_frame = self.ts.tt_jd(tt)
        m_geo = earth.at(t_frame).observe(moon).apparent()
        s_geo = earth.at(t_frame).observe(sun).apparent()

        m_ra, m_dec, m_dist = m_geo.radec()
        s_ra, s_dec, s_dist = s_geo.radec()

        as_ra = (s_ra.hours + 12.0) % 24.0
        as_dec = -s_dec.degrees

        d_ra = (m_ra.hours - as_ra) * 15.0 * math.cos(math.radians(as_dec))
        d_dec = m_dec.degrees - as_dec
        sep = math.sqrt(d_ra**2 + d_dec**2)

        sd_m = math.degrees(math.asin(1737.4 / m_dist.km))
        pi_m = math.degrees(math.asin(6378.14 / m_dist.km))
        sd_s = math.degrees(math.asin(696000.0 / s_dist.km))
        pi_s = 8.794 / 3600.0

        r_umbra = (pi_m + pi_s - sd_s) * 1.02
        r_penumbra = (pi_m + pi_s + sd_s) * 1.02

        obs_pct = calc_obscurations(sep, sd_m, r_umbra)

        dt_lokal = t_frame.utc_datetime() + datetime.timedelta(hours=tz)
        frames.append({
            'tipe': 'Bulan', 'waktu': dt_lokal.strftime("%H:%M:%S"),
            'd_x': d_ra, 'd_y': d_dec, 'sd_moon': sd_m, 'r_umbra': r_umbra,
            'r_penumbra': r_penumbra, 'obs': obs_pct, 'sep': sep
        })

```

```

    })
    return frames

# Buat kalkulasi pertama (berbasis lokasi user yang dipilih - Toposentrik)
sim_frames = generate_frames(lokalasi_awal, is_geocentric=False)
is_fallback = False

# Saring frame
if "Matahari" in objek:
    valid_frames = [f for f in sim_frames if f['obs'] > 0.0 and f['alt_matahari'] > -1.0]
else:
    valid_frames = [f for f in sim_frames if f['sep'] < (f['r_penumbra'] + f['sd_moon'])]

# ---> PERBAIKAN: JIKA GAGAL LOKAL, GUNAKAN PANDANGAN GEOSENTRIS (Pusat Bumi) <---
# Ini dijamin 100% selalu melihat Gerhana Matahari di manapun dia terjadi di bumi.
if not valid_frames:
    sim_frames = generate_frames(None, is_geocentric=True)

    if "Matahari" in objek:
        valid_frames = [f for f in sim_frames if f['obs'] > 0.0]
    else:
        valid_frames = [f for f in sim_frames if f['sep'] < (f['r_penumbra'] + f['sd_moon'])]

    is_fallback = True

if not valid_frames:
    self.after(0, lambda: self.lbl_sim_info.configure(text=f"Gerhana {objek} gagal disimulasikan
    secara sistem.", text_color="#FF5252"))
    return

peak_frame = max(valid_frames, key=lambda x: x['obs'])

# Buffer 15 frame sebelum dan sesudah agar animasi mulus
idx_start = max(0, sim_frames.index(valid_frames[0]) - 15)
idx_end = min(len(sim_frames), sim_frames.index(valid_frames[-1]) + 15)
self.sim_data = sim_frames[idx_start:idx_end]

# Tentukan Pesan Peringatan Teks
if is_fallback:
    pesan_info = f"⚠️ TIDAK TERLIHAT LOKAL (Dialihkan ke View Pusat Bumi)\nWilayah
    Terdampak: {wilayah_global}"
else:
    pesan_info = f"Wilayah Terdampak: {wilayah_global}"

self.after(0, self._siapkan_ui_animasi, peak_frame, pesan_info)

except Exception as e:

```



```

import traceback
self.after(0, lambda: self.lbl_sim_info.configure(text=f"Gagal memuat simulasi: {e}",
text_color="#FF5252"))
print(traceback.format_exc())

def _siapkan_ui_animasi(self, peak_frame, pesan_info):
    self.sim_slider.configure(from_=0, to=len(self.sim_data)-1, number_of_steps=len(self.sim_data)-1)
    self.sim_slider.set(0)

    info = f"Maksimum Obscuration: {peak_frame['obs']:.1f}%"

    # Tambahkan label Geosentris jika ini berasal dari fallback pusat bumi
    if peak_frame['tipe'] == 'Matahari':
        alt_txt = "Geosentris" if peak_frame['alt_matahari'] == 90.0 else
        f"{peak_frame['alt_matahari']:.1f}°"
        info += f" | Alt Matahari: {alt_txt}"

    # Tambahkan teks info peringatan lokasi
    info += f"\n{pesan_info}"

    self.lbl_sim_info.configure(text=info)

    self._update_sim_frame(0)

def _update_sim_frame(self, index):
    idx = int(float(index))
    if not self.sim_data or idx >= len(self.sim_data): return

    frame = self.sim_data[idx]
    canvas = self.sim_canvas
    canvas.delete("all")
    cx, cy = 250, 250 # Pusat Kanvas

    if frame['tipe'] == 'Matahari':
        r_sun_px = 120
        scale = r_sun_px / frame['sd_sun']
        r_moon_px = frame['sd_moon'] * scale
        moon_x = cx + (frame['d_x'] * scale)
        moon_y = cy - (frame['d_y'] * scale) # Minus karena Y canvas terbalik

    # Efek warna langit gelap perlahan
    kecerahan_langit = int(max(5, 255 - (frame['obs'] * 2.5)))
    warna_langit = f"#{kecerahan_langit:02x}{kecerahan_langit:02x}1a" if kecerahan_langit < 20 else
    "#050510"
    canvas.configure(bg=warna_langit)

    canvas.create_line(cx, 0, cx, 500, fill="#333333", dash=(4,4))
    canvas.create_line(0, cy, 500, cy, fill="#333333", dash=(4,4))

```

```

        canvas.create_oval(cx - r_sun_px - 5, cy - r_sun_px - 5, cx + r_sun_px + 5, cy + r_sun_px + 5, fill="",
outline="#FFD54F", width=2, stipple="gray25")
        canvas.create_oval(cx - r_sun_px, cy - r_sun_px, cx + r_sun_px, cy + r_sun_px, fill="#FFEA00",
outline="#FFC107")
        canvas.create_oval(moon_x - r_moon_px, moon_y - r_moon_px, moon_x + r_moon_px, moon_y
+ r_moon_px, fill="#0F0F0F", outline="#222222")

        info_teks = f"Waktu Lokal: {frame['waktu']}\nTertutup (Obscuration): {frame['obs']:.2f}%\nAlt
Matahari: {frame['alt_matahari']:.1f}°"
        canvas.create_text(15, 20, text=info_teks, fill="#00E676", font=("Consolas", 12, "bold"),
anchor="nw")

    else:
        # RENDER GERHANA BULAN
        scale = 130 / frame['r_umbra'] # Jari-jari Umbra dipatok 130 px
        r_umbra_px = 130
        r_pen_px = frame['r_penumbra'] * scale
        r_moon_px = frame['sd_moon'] * scale

        moon_x = cx + (frame['d_x'] * scale)
        moon_y = cy - (frame['d_y'] * scale)

        canvas.configure(bg="#050510")
        canvas.create_line(cx, 0, cx, 500, fill="#222222", dash=(4,4))
        canvas.create_line(0, cy, 500, cy, fill="#222222", dash=(4,4))

        # Gambar Penumbra & Umbra (Bayangan Bumi)
        canvas.create_oval(cx - r_pen_px, cy - r_pen_px, cx + r_pen_px, cy + r_pen_px, fill="#15151A",
outline="#333344", width=2)
        canvas.create_text(cx, cy - r_pen_px - 10, text="Batas Penumbra", fill="#777777",
font=("Consolas", 10))

        canvas.create_oval(cx - r_umbra_px, cy - r_umbra_px, cx + r_umbra_px, cy + r_umbra_px,
fill="#2E0E0E", outline="#4A1C1C", width=2)
        canvas.create_text(cx, cy - r_umbra_px - 10, text="Batas Umbra", fill="#AA4444",
font=("Consolas", 10))

        # Efek Perubahan Warna Bulan (Blood Moon) jika masuk Umbra
        obs = frame['obs']
        if obs > 95: moon_color = "#5C1010" # Merah Darah Gelap (Totalitas)
        elif obs > 40: moon_color = "#B71C1C" # Merah
        elif obs > 0: moon_color = "#E57373" # Kemerahan
        else: moon_color = "#E0E0E0" # Bulan Purnama Normal

        canvas.create_oval(moon_x - r_moon_px, moon_y - r_moon_px, moon_x + r_moon_px, moon_y
+ r_moon_px, fill=moon_color, outline="#FFFFFF")

```

```

info_teks = f"Waktu Lokal: {frame['waktu']}\nTertutup Umbra: {frame['obs']:.2f}%"
canvas.create_text(15, 20, text=info_teks, fill="#FFD54F", font=("Consolas", 12, "bold"),
anchor="nw")

def _toggle_sim_play(self):
    if not self.sim_data: return
    self.sim_is_playing = not self.sim_is_playing
    if self.sim_is_playing:
        self._anim_loop()

def _anim_loop(self):
    if not self.sim_is_playing or not getattr(self, 'win_sim', None) or not self.win_sim.winfo_exists():
        self.sim_is_playing = False
        return

    curr_val = int(self.sim_slider.get())
    next_val = curr_val + 1

    if next_val >= len(self.sim_data):
        self.sim_is_playing = False
        return

    self.sim_slider.set(next_val)
    self._update_sim_frame(next_val)
    self.win_sim.after(60, self._anim_loop) # Kecepatan frame animasi

def _get_islamic_event(self, h_day, h_month, weekday):
    """
    Mendeteksi hari libur Islam dan jadwal puasa sunnah.
    h_month: 1-12 (Muharam - Zulhijah)
    weekday: 0-6 (0=Senin, 6=Ahad)
    """
    events = []
    color = None
    is_haram = False

    # 1. Cek Hari Haram Puasa (Warna Merah)
    if (h_month == 10 and h_day == 1):
        events.append("1 Syawal: Idul Fitri (Haram Puasa)")
        color = "#FF5252"
        is_haram = True
    elif (h_month == 12 and h_day == 10):
        events.append("10 Zulhijah: Idul Adha (Haram Puasa)")
        color = "#FF5252"
        is_haram = True
    elif h_month == 12 and h_day in [11, 12, 13]:
        events.append("Hari Tasyriq (Haram Puasa)")

```

```

color = "#FF5252"
is_haram = True

# 2. Cek Peringatan Hari Besar Islam (Warna Ungu Muda)
if h_month == 1 and h_day == 1: events.append("Tahun Baru Hijriah")
if h_month == 3 and h_day == 12: events.append("Maulid Nabi Muhammad SAW")
if h_month == 7 and h_day == 27: events.append("Isra' Mi'raj")
if h_month == 8 and h_day == 15: events.append("Nisfu Syakban")

# 3. Cek Puasa Sunnah (Hanya jika bukan hari haram)
if not is_haram:
    if h_month == 1 and h_day == 9:
        events.append("Puasa Sunnah Tasu'a")
        if not color: color = "#00E676" # Hijau
    if h_month == 1 and h_day == 10:
        events.append("Puasa Sunnah Asyura")
        if not color: color = "#00E676"
    if h_month == 12 and h_day == 8:
        events.append("Puasa Sunnah Tarwiyah")
        if not color: color = "#00E676"
    if h_month == 12 and h_day == 9:
        events.append("Puasa Sunnah Arafah")
        if not color: color = "#00E676"
    if h_month == 9:
        events.append("Puasa Ramadhan")
        if not color: color = "#8BC34A" # Hijau Muda
    if h_day in [13, 14, 15] and h_month != 9:
        events.append("Puasa Ayyamul Bidh")
        if not color: color = "#00B0FF" # Biru Muda (Cyan)

# Cek Puasa Senin & Kamis
if weekday in [0, 3] and h_month != 9: # 0 = Senin, 3 = Kamis
    nama_hari = "Senin" if weekday == 0 else "Kamis"
    events.append(f"Puasa Sunnah {nama_hari}")
    if not color: color = "#FFA000" # Orange

# Set warna peringatan biasa jika tidak ada puasa/haram
if events and not color:
    color = "#E040FB" # Ungu

tooltip_text = "\n".join(events) if events else ""
return color, tooltip_text

def calculate_mizwala(self):
    try:
        year = int(self.entry_miz_year.get())
        month = int(self.entry_miz_month.get())
        day = int(self.entry_miz_day.get())

```

```

self.auto_switch_ephemeris(year)

lat = float(self.entry_miz_lat.get())
lon = float(self.entry_miz_lon.get())
elev = float(self.entry_miz_elev.get())
tz = float(self.entry_miz_tz.get())

tinggi_tongkat = float(self.entry_miz_tinggi.get())

# --- MEMAKSA INTERVAL MENJADI 1 MENIT (Sesuai kode awal) ---
step_mnt = 1

earth, sun = self.eph['earth'], self.eph['sun']
loc = wgs84.latlon(lat, lon, elevation_m=elev)
observer = earth + loc

# 1. Cari Waktu Terbit dan Terbenam Lokal untuk membatasi tabel
t0 = self.ts.utc(year, month, day, -int(tz))
t1 = self.ts.utc(year, month, day, 24 - int(tz))

f_rs = almanac.sunrise_sunset(self.eph, loc)
t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

t_rise, t_set = None, None
for t_ev, y_ev in get_safe_events(t_rs, y_rs):
    if y_ev == 1: t_rise = t_ev
    else: t_set = t_ev

if t_rise is None or t_set is None:
    self.after(0, self.display_error, "Matahari tidak terbit/terbenam di lokasi ini pada tanggal
tersebut (Anomali Lintang).")
    return

# 2. Sampel Perjalanan Matahari Harian
tt_array = np.linspace(t_rise.tt, t_set.tt, 1440)
t_search = self.ts.tt_jd(tt_array)
alt_arr = observer.at(t_search).observe(sun).apparent().altaz()[0].degrees

# --- KALKULASI DZUHUR ---
idx_noon = np.argmax(alt_arr)
alt_noon = alt_arr[idx_noon]

t_noon = self.ts.tt_jd(tt_array[idx_noon])
noon_dt = t_noon.utc_datetime() + datetime.timedelta(hours=tz)

zenith_noon = 90.0 - alt_noon
bayangan_dzuhur = tinggi_tongkat * math.tan(math.radians(max(0, zenith_noon)))

```

```

target_shadow_asr = tinggi_tongkat + bayangan_dzuhur

# --- KALKULASI WAKTU DHUHA ---
alt_am = alt_arr[:idx_noon]
tt_am = tt_array[:idx_noon]

tt_dhuha = None
diffs = alt_am - 4.5
for i in range(len(diffs)-1):
    if diffs[i] <= 0 and diffs[i+1] > 0:
        frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
        tt_dhuha = tt_am[i] + frac * (tt_am[i+1] - tt_am[i])
        break

dhuha_str = "----"
if tt_dhuha is not None:
    dhuha_dt_exact = self.ts.tt_jd(tt_dhuha).utc_datetime() + datetime.timedelta(hours=tz)
    dhuha_str = dhuha_dt_exact.strftime("%H:%M:%S")

# 3. Format Output Header
output_lines = []
lat_str = format_angle(lat).replace("+", "").replace("-", "-")
lon_str = format_angle(lon).replace("+", "")

# ---> PERBAIKAN: MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR <---
try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{{nama_kota}}, {{nama_prov}} (Lat {{lat_str}}, Lon {{lon_str}}, Elev {{elev}}m, TZ {{tz}})"
except:
    lokasi_text = f"Lat {{lat_str}}, Lon {{lon_str}}, Elev {{elev}}m, TZ {{tz}}"

output_lines.append(self.get_header(90))
output_lines.append("[ Tabel Bayangan Tongkat Istiwa / Mizwala ].center(90))
output_lines.append("")
output_lines.append(f"* Tanggal : {{day:02d}}/{{month:02d}}/{{format_tahun_aman(year)}}")
output_lines.append(f"* Lokasi : {{lokasi_text}}")
output_lines.append(f"* Tongkat : {{tinggi_tongkat}} cm")
output_lines.append(f"* Waktu Dhuha: {{dhuha_str}} (Matahari Naik 4.5°)")
output_lines.append(f"* Byg Dzhur : {{bayangan_dzuhur:.2f}} cm (Bayangan Terpendek)")
output_lines.append(f"* Target Ashr: {{target_shadow_asr:.2f}} cm (Tongkat + Byg Dzhur)")
output_lines.append("=*90)
output_lines.append(f"{'Waktu (LT)':<12} | {'Alt Matahari':<14} | {'Arah Bayangan':<15} |
{'Panjang Bayangan':<20}")
output_lines.append("-" * 90)

# 4. Generate Data Tabel berdasarkan Step Waktu
start_dt = t_rise.utc_datetime() + datetime.timedelta(hours=tz)

```

```

end_dt = t_set.utc_datetime() + datetime.timedelta(hours=tz)

menit_awal = (start_dt.minute // step_mnt) * step_mnt + step_mnt
curr_dt = start_dt.replace(minute=0, second=0, microsecond=0) +
datetime.timedelta(minutes=menit_awal)

# --- SISTEM BENDERA (FLAG) ---
dhuha_marked = False
dzuhur_marked = False
ashar_marked = False

while curr_dt <= end_dt:
    t_calc = self.ts.from_datetime((curr_dt -
datetime.timedelta(hours=tz)).replace(tzinfo=pytz.utc))

    app = observer.at(t_calc).observe(sun).apparent()
    alt, az, _ = app.altaz()

    alt_deg = alt.degrees
    az_deg = az.degrees

    if alt_deg > 0:
        zenith = 90.0 - alt_deg
        shadow_len = tinggi_tongkat * math.tan(math.radians(zenith))
        shadow_az = (az_deg + 180.0) % 360.0

        waktu_str = curr_dt.strftime("%H:%M")
        alt_str = f"{alt_deg:.2f}°"
        arah_str = f"{shadow_az:.2f}°"
        panjang_str = f"{shadow_len:.2f} cm"

        marker = ""

        # 1. Dhuha
        if curr_dt < noon_dt and alt_deg >= 4.5 and not dhuha_marked:
            marker = " << MASUK DHUHA"
            dhuha_marked = True

        # 2. Dzuhur
        elif curr_dt >= noon_dt and not dzuhur_marked:
            marker = " << ZAWAL / DZUHUR"
            dzuhur_marked = True

        # 3. Ashar
        elif curr_dt > noon_dt and shadow_len >= target_shadow_asr and not ashar_marked:
            marker = " << MASUK ASHAR"
            ashar_marked = True

```

```

        output_lines.append(f"{waktu_str:<12} | {alt_str:<14} | {arah_str:<15} |
{panjang_str}{marker}")

        curr_dt += datetime.timedelta(minutes=step_mnt)

        output_lines.append("=" * 90)
        output_lines.append("* Keterangan: Arah bayangan dihitung dari Utara (0°) searah jarum jam.")

        self.after(0, self.display_result, "\n".join(output_lines))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")

def buka_simulasi_mizwala(self):
    try:
        # 1. Ambil Parameter Input Tanggal, Lokasi, dan Tongkat
        year = int(self.entry_miz_year.get())
        month = int(self.entry_miz_month.get())
        day = int(self.entry_miz_day.get())
        lat = float(self.entry_miz_lat.get())
        lon = float(self.entry_miz_lon.get())
        elev = float(self.entry_miz_elev.get())
        tz = float(self.entry_miz_tz.get())
        tinggi_tongkat = float(self.entry_miz_tinggi.get())

        # ---> LOGIKA PINTAR PEMBACAAN WAKTU <---
        waktu_input = self.entry_miz_waktu.get().strip().upper()

        is_live_mode = False
        jam_awal_desimal = 12.0

        if waktu_input == "" or waktu_input == "LIVE" or waktu_input == "SEKARANG":
            is_live_mode = True
        else:
            try:
                waktu_input = waktu_input.replace(".", ":")
                if ":" in waktu_input:
                    h_str, m_str = waktu_input.split(":")
                    jam_awal_desimal = int(h_str) + (int(m_str) / 60.0)
                else:
                    jam_awal_desimal = float(waktu_input)
            except Exception:
                is_live_mode = True # Fallback ke Live jika format ngawur

        self.auto_switch_ephemeris(year)
        earth, sun = self.eph['earth'], self.eph['sun']
        loc = wgs84.latlon(lat, lon, elevation_m=elev)

```



```

observer = earth + loc

# 2. Hitung Waktu Terbit & Terbenam sebagai Batas Slider
t0 = self.ts.utc(year, month, day, -int(tz))
t1 = self.ts.utc(year, month, day, 24 - int(tz))
f_rs = almanac.sunrise_sunset(self.eph, loc)
t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

t_rise, t_set = None, None
for t_ev, y_ev in get_safe_events(t_rs, y_rs):
    if y_ev == 1: t_rise = t_ev
    else: t_set = t_ev

if t_rise is None or t_set is None:
    messagebox.showerror("Error", "Matahari tidak terbit/terbenam pada tanggal ini.")
    return

# Konversi waktu batas ke jam desimal lokal secara presisi
dt_rise = t_rise.utc_datetime() + datetime.timedelta(hours=tz)
dt_set = t_set.utc_datetime() + datetime.timedelta(hours=tz)
start_hr = dt_rise.hour + (dt_rise.minute / 60.0)
end_hr = dt_set.hour + (dt_set.minute / 60.0)

# 3. Buat Jendela GUI Baru
win_sim = ctk.CTkToplevel(self)
win_sim.title("Simulasi Visual Kompas Mizwala")
win_sim.geometry("650x800")
win_sim.attributes("-topmost", True)
win_sim.configure(fg_color="#0A0A0A")

ctk.CTkLabel(win_sim, text="SIMULASI ARAH BAYANGAN MATAHARI", font=("Segoe UI", 16,
"bold"), text_color="#FFD54F").pack(pady=10)

# 4. Inisialisasi Matplotlib Polar
fig = plt.figure(figsize=(6, 6), facecolor='#0A0A0A')
ax = fig.add_subplot(111, projection='polar')

canvas_sim = FigureCanvasTkAgg(fig, master=win_sim)
canvas_sim.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=5)

# 5. Kontrol UI Bawah
ctrl_frame = ctk.CTkFrame(win_sim, fg_color="#1E1E1E", corner_radius=10)
ctrl_frame.pack(fill="x", padx=20, pady=10)

top_ctrl = ctk.CTkFrame(ctrl_frame, fg_color="transparent")
top_ctrl.pack(fill="x", padx=10, pady=5)

```

```

lbl_waktu = ctk.CTkLabel(top_ctrl, text="Waktu Lokal: --:--:--", font=("Consolas", 16, "bold"),
text_color="#00E676")
lbl_waktu.pack(side="left", padx=10)

# ---> TOMBOL TOGGLE LIVE / KUSTOM <---
self.miz_is_live = is_live_mode

def toggle_live():
    self.miz_is_live = True
    btn_live.configure(fg_color="#2E7D32", text="● MODE LIVE")

btn_live = ctk.CTkButton(top_ctrl, text="● MODE LIVE" if is_live_mode else "⏸ MODE
KUSTOM",
                        fg_color="#2E7D32" if is_live_mode else "#555555",
                        width=120, command=toggle_live)
btn_live.pack(side="right", padx=10)

lbl_info = ctk.CTkLabel(ctrl_frame, text="Mengkalkulasi...", font=("Consolas", 12))
lbl_info.pack(pady=5)

slider_var = ctk.DoubleVar()
max_radius = tinggi_tongkat * 3.0

# ---> FUNGSI RENDER UTAMA <---
def update_plot(val, update_slider=False):
    try:
        jam_desimal = float(val)
        jam = int(jam_desimal)

        # Ekstrak presisi hingga level detik
        sisa_menit = (jam_desimal - jam) * 60.0
        menit = int(sisa_menit)
        detik = int(round((sisa_menit - menit) * 60.0))

        if detik >= 60:
            menit += 1; detik -= 60
        if menit >= 60:
            jam += 1; menit -= 60

        if update_slider:
            slider_var.set(jam_desimal)

        lbl_waktu.configure(text=f"Waktu Lokal: {jam:02d}:{menit:02d}:{detik:02d}")

    # Cegah time out of bounds yang memicu error kalender
    jam_safe = min(23, max(0, jam))

```

```

base_local = datetime.datetime(year, month, day)
curr_local = base_local + datetime.timedelta(hours=jam_safe, minutes=menit,
seconds=detik)
curr_utc = curr_local - datetime.timedelta(hours=tz)

t_calc = self.ts.from_datetime(curr_utc.replace(tzinfo=pytz.utc))
app = observer.at(t_calc).observe(sun).apparent()
alt, az, _ = app.altaz()

ax.clear()
ax.set_theta_zero_location('N')
ax.set_theta_direction(-1)
ax.set_facecolor('#121212')
ax.tick_params(colors='white')
ax.grid(color='#333333', linestyle='--')
ax.set_ylim(0, max_radius)

ax.set_xticks(np.radians([0, 45, 90, 135, 180, 225, 270, 315]))
ax.set_xticklabels(['U (0°)', 'TL', 'T (90°)', 'TG', 'S (180°)', 'BD', 'B (270°)', 'BL'])
ax.set_yticklabels([])

if alt.degrees > 0:
    zenith = 90.0 - alt.degrees
    shadow_len = tinggi_tongkat * math.tan(math.radians(zenith))
    shadow_az = (az.degrees + 180.0) % 360.0

    ax.plot(0, 0, marker='o', color='white', markersize=8, label=f"Tongkat
({tinggi_tongkat}cm)")

    display_shadow_len = min(shadow_len, max_radius)
    ax.plot([0, math.radians(shadow_az)], [0, display_shadow_len], color='#00E5FF',
linewidth=3.5, label="Arah Bayangan")
    ax.plot(math.radians(az.degrees), max_radius * 0.9, marker='o', color='#FFD54F',
markersize=14, label="Matahari")

    lbl_info.configure(text=f"Alt Mth: {alt.degrees:.1f}° | Arah Byg: {shadow_az:.1f}° | Pjg Byg:
{shadow_len:.1f} cm", text_color="white")
    ax.legend(loc='lower left', bbox_to_anchor=(-0.1, -0.15), facecolor='#0A0A0A',
edgecolor='#333', labelcolor='white')
else:
    lbl_info.configure(text="Matahari di bawah Ufuk (Malam Hari)", text_color="#FF5252")

canvas_sim.draw_idle()

except Exception as ex:
    print("Error rendering:", ex)

# ---> JIKA SLIDER DIGESER MANUAL <---

```

```

def on_slider_drag(val):
    self.miz_is_live = False
    btn_live.configure(fg_color="#555555", text="  MODE KUSTOM")
    update_plot(val, update_slider=False)

slider = ctk.CTkSlider(
    ctrl_frame,
    from_=start_hr,
    to=end_hr,
    variable=slider_var,
    command=on_slider_drag,
    progress_color="#F57C00",
    button_color="#E65100"
)
slider.pack(fill="x", padx=20, pady=(10, 20))

# ---> LOOP ANIMASI REAL-TIME (TICK PER DETIK) <---
def live_loop():
    if not win_sim.wininfo_exists():
        return # Stop memori memanggil loop jika window sudah disilang/ditutup

    if getattr(self, 'miz_is_live', False):
        now = datetime.datetime.now()
        jam_sekarang_desimal = now.hour + (now.minute / 60.0) + (now.second / 3600.0)

        # Jika waktu real-time komputer saat ini masih berada di dalam jadwal siang hari (terbit -
        terbenam)
        if start_hr <= jam_sekarang_desimal <= end_hr:
            update_plot(jam_sekarang_desimal, update_slider=True)
        else:
            # Jika sudah malam, stop rendernya agar garis tidak kacau, cukup gerakkan angkanya
            lbl_info.configure(text="Matahari di bawah Ufuk (Malam Hari)", text_color="#FF5252")
            lbl_waktu.configure(text=f"Waktu Lokal: {now.strftime('%H:%M:%S')}")
            ax.clear()
            ax.set_theta_zero_location('N')
            ax.set_theta_direction(-1)
            ax.set_facecolor('#121212')
            ax.set_ylim(0, max_radius)
            ax.set_xticks(np.radians([0, 45, 90, 135, 180, 225, 270, 315]))
            ax.set_xticklabels(['U (0°)', 'TL', 'T (90°)', 'TG', 'S (180°)', 'BD', 'B (270°)', 'BL'])
            ax.set_yticklabels([])
            canvas_sim.draw_idle()
            slider_var.set(start_hr)

    # Ulangi setiap 1 detik
    win_sim.after(1000, live_loop)

```

```

# Nyalakan engine live loop
live_loop()

# ---> PENYETELAN WAKTU AWAL SAAT PERTAMA KALI JENDELA DIBUKA <---
if not is_live_mode:
    # Mode Kustom (Sesuai dengan yang Anda input misal "09:00")
    if start_hr <= jam_awal_desimal <= end_hr:
        update_plot(jam_awal_desimal, update_slider=True)
    else:
        messagebox.showwarning("Peringatan Waktu", f"Waktu yang Anda input ({waktu_input})
        terjadi saat malam hari (matahari di bawah ufuk).\n\nSimulasi dikembalikan ke mode Live.")
        self.miz_is_live = True
        btn_live.configure(fg_color="#2E7D32", text="● MODE LIVE")
    else:
        # Jika user mengisi dengan "LIVE" atau membiarkannya kosong
        now = datetime.datetime.now()
        jam_sekarang_desimal = now.hour + (now.minute / 60.0) + (now.second / 3600.0)
        if not (start_hr <= jam_sekarang_desimal <= end_hr):
            # Jika dibuka saat malam hari, arahkan slider ke posisi Dzuhur agar user bisa melihat simulasi
            default
            tengah_hari = (start_hr + end_hr) / 2.0
            self.miz_is_live = False
            btn_live.configure(fg_color="#555555", text="⏸ MODE KUSTOM")
            update_plot(tengah_hari, update_slider=True)

except Exception as e:
    import traceback
    messagebox.showerror("Error", f"Gagal memuat simulasi: {e}\n{traceback.format_exc()}")

def _build_hijri_db_thread(self):
    try:
        start_year = int(self.entry_ab_start.get())
        end_year = int(self.entry_ab_end.get())
    except ValueError:
        self.after(0, lambda: messagebox.showerror("Error", "Tahun harus berupa angka. "))
        self.after(0, lambda: self.btn_hitung.configure(state="normal"))
        return

    self.after(0, lambda: self.lbl_status.configure(text=f"Membangun DB Hijriah {start_year}-{end_year}...", text_color="#00E5FF"))
    self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
    self.after(0, lambda: self.textbox.delete("1.0", "end"))
    self.after(0, lambda: self.textbox.insert("end", f"Memulai iterasi mesin KHGT dari tahun {start_year}
    H hingga {end_year} H...\nProses ini memerlukan waktu pemrosesan CPU, mohon tunggu...\n\n"))
    self.after(0, lambda: self.btn_hitung.configure(state="disabled"))

new_db = {}

```

```

hari_nama = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
bulan_masehi_singkat = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec"]

```

```

try:
    for y_h in range(start_year, end_year + 1):
        # =====
        # PERBAIKAN: Hitung estimasi tahun Masehi per iterasi (bukan di luar loop).
        # Auto-Switch otomatis memakai de440s.bsp atau de441-new.bsp sesuai tahun
        # saat tahun target keluar dari rentang de440s, sehingga tidak terjadi EphemerisRangeError.
        # =====
        approx_greg_curr = int(y_h * 0.970224 + 622.54)
        self.auto_switch_ephemeris(approx_greg_curr)

        year_data = []
        for m_h in range(1, 13):
            # 1. Cari TT Ijtimak bulan ini
            approx_tt_curr = self.get_approx_nm_tt(y_h, m_h)
            nm_list_curr = self.get_new_moons_in_range(approx_tt_curr - 5.0, approx_tt_curr + 5.0)
            if not nm_list_curr: raise ValueError(f"Fase Bulan Baru tidak ditemukan untuk {y_h}-{m_h}")
            nm_curr = nm_list_curr[0]
            start_tt_curr = self.calculate_khgt_1st_of_month(nm_curr)

            # 2. Cari TT Ijtimak bulan depan (untuk menghitung jumlah hari)
            next_m = m_h + 1
            next_y = y_h
            if next_m > 12:
                next_m = 1
                next_y += 1

            approx_tt_next = self.get_approx_nm_tt(next_y, next_m)
            nm_list_next = self.get_new_moons_in_range(approx_tt_next - 5.0, approx_tt_next + 5.0)
            nm_next = nm_list_next[0]
            start_tt_next = self.calculate_khgt_1st_of_month(nm_next)

            # 3. Hitung Jumlah Hari (Selisih waktu mulai bulan depan dan bulan ini)
            jumlah_hari = int(round(start_tt_next - start_tt_curr))

            # 4. Format String Tanggal Masehi dan Hari Pasaran (Standar)
            t_obj = self.ts.tt_jd(start_tt_curr)
            gy, gm, gd, _, _, _ = t_obj.utc
            hari_idx = int(t_obj.whole % 7)
            nama_h = hari_nama[hari_idx]

            tgl_str = f"{int(gd):02d}-{bulan_masehi_singkat[int(gm)-1]}-{int(gy)}"

            year_data.append([BULAN_HIJRIAH[m_h-1], nama_h, tgl_str, jumlah_hari])

```

```

new_db[y_h] = year_data

# Update log GUI tanpa freeze
self.after(0, lambda yr=y_h: self.textbox.insert("end", f"✓ Tahun {yr} H selesai diproses...\n"))
self.after(0, lambda: self.textbox.see("end"))

# Save ke JSON
db_path = os.path.join(BASE_DIR, "db_hijriah.json")
with open(db_path, "w", encoding="utf-8") as f:
    json.dump(new_db, f, indent=4, ensure_ascii=False)

# Update Memori Aplikasi (Live tanpa perlu restart)
global HIJRI_DB
HIJRI_DB.update(new_db)

self.after(0, lambda: self.textbox.insert("end", f"\n[+] SUKSES! Database berhasil dibangun dan
disimpan ke:\n{db_path}\nData telah dimuat ke dalam memori aplikasi sehingga Kalender sekarang
sudah ter-update secara Live.))
self.after(0, lambda: self.lbl_status.configure(text="Auto-Builder Selesai", text_color="#00E676"))

except Exception as e:
    import traceback
    self.after(0, lambda err=str(e), tb=traceback.format_exc(): self.textbox.insert("end", f"\nERROR:
{err}\n{tb}"))
    self.after(0, lambda: self.lbl_status.configure(text="Auto-Builder Gagal", text_color="#FF1744"))
finally:
    self.after(0, lambda: self.btn_hitung.configure(state="normal"))
    self.after(0, lambda: self.textbox.configure(state="disabled"))

def setup_kriteria_batas_out_frame(self):
    """Mempersiapkan area canvas untuk Matplotlib Kriteria Batas 31"""
    self.frame_kriteria_batas_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")

    # Setup Figure Matplotlib (Theme disesuaikan untuk Dark Mode)
    self.fig_kb, self.ax_kb = plt.subplots(figsize=(8, 6), facecolor='#101010')
    self.ax_kb.set_facecolor('#101010')

    elongation_max = 15 # Diubah menjadi 15 agar lebih lebar
    altitude_max = 15
    altitude_min = -5 # Diubah agar nilai minus (bawah ufuk) kelihatan

    # Pembuatan Zona Parameter
    self.ax_kb.fill_between([0, 7], altitude_min, altitude_max, color='#FF5252', alpha=0.3, label='Batas
Danjon (< 7°)')
    self.ax_kb.fill_between([7, elongation_max], altitude_min, 5, color='#FFB74D', alpha=0.3,
label='Bias Senja (< 5°)')

```

```

self.ax_kb.fill_between([7, 8], 5, altitude_max, color='#FFF176', alpha=0.3, label='Margin Keamanan
(7°-8°)')
self.ax_kb.fill_between([8, elongation_max], 5, altitude_max, color='#69F0AE', alpha=0.3,
label='Kriteria KHGT Terpenuhi')

self.ax_kb.axhline(y=0, color='gray', linestyle='-', linewidth=1.5) # Garis Nol (Horizon)
self.ax_kb.axvline(x=8, color='#00E676', linestyle='--', linewidth=2)
self.ax_kb.axhline(y=5, color='#00E676', linestyle='--', linewidth=2)
self.ax_kb.axvline(x=7, color='#FF1744', linestyle=':', linewidth=2)

self.ax_kb.text(3.5, 6.5, 'MUSTAHIL\n(Secara Fisis)', ha='center', va='center', fontsize=12,
fontweight='bold', color='#FF5252')
self.ax_kb.text(8.5, 2.5, 'TIDAK TERLIHAT\n(Tertutup Cahaya Senja)', ha='center', va='center',
fontsize=10, fontweight='bold', color='#FFB74D')
self.ax_kb.text(7.5, 8, 'RAWAN', ha='center', va='center', fontsize=10, fontweight='bold',
color='#FFF176', rotation=90)
self.ax_kb.text(9, 8, 'RUKYAT BIL \ILMI', ha='center', va='center', fontsize=10, fontweight='bold',
color='#00E676')

self.titik_hilal_kb, = self.ax_kb.plot([8.5], [6.0], 'w*', markersize=18, zorder=5,
markededgecolor='black', label='Titik Simulasi Real-time')

self.teks_info_bintang = self.ax_kb.text(8.5, 6.6, 'Eln: 8.50° | Alt: 6.00°',
color='white', fontsize=11, fontweight='bold',
ha='center', va='bottom', zorder=6,
bbox=dict(boxstyle="round,pad=0.4", fc="#1E1E1E", ec="#00E5FF",
alpha=0.9))

self.ax_kb.set_xlim(0, elongation_max)
self.ax_kb.set_ylim(altitude_min, altitude_max)
self.ax_kb.set_xlabel('Elongasi / Jarak Sudut Bulan-Matahari (Derajat)', fontsize=12, color='white')
self.ax_kb.set_ylabel('Ketinggian Hilal / Altitude (Derajat)', fontsize=12, color='white')
self.ax_kb.set_title('Anatomi Kriteria KHGT 5-8', fontsize=14, fontweight='bold', pad=15,
color='white')
self.ax_kb.grid(True, linestyle='--', alpha=0.2, color='white')
self.ax_kb.tick_params(colors='white')

# Legend
legend = self.ax_kb.legend(loc='upper left', fontsize='9', facecolor='#1E1E1E', edgecolor='#333333')
for text in legend.get_texts():
    text.set_color("white")

self.fig_kb.tight_layout()

# Integrasi ke Tkinter
self.canvas_kb = FigureCanvasTkAgg(self.fig_kb, master=self.frame_kriteria_batas_out)
self.canvas_kb.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=10)

```



```

# Status Box di bawah grafik
status_frame = ctk.CTkFrame(self.frame_kriteria_batas_out, fg_color="#1A1A1A", corner_radius=8,
border_width=1, border_color="#333333")
status_frame.pack(fill="x", padx=15, pady=(0, 15))

self.kb_status_label = ctk.CTkLabel(status_frame, text="STATUS: -", font=("Consolas", 16, "bold"))
self.kb_status_label.pack(pady=(10, 5))

self.kb_alasan_label = ctk.CTkLabel(status_frame, text="ALASAN: -", font=("Segoe UI", 13),
justify="center", wraplength=700)
self.kb_alasan_label.pack(pady=(0, 10), padx=20)

def update_kriteria_batas_plot(self):
    """Memperbarui grafik dan penjelasan teks Menu 31 saat slider digeser"""
    if not hasattr(self, 'ax_kb'): return

    x = self.var_kb_elongasi.get()
    y = self.var_kb_ketinggian.get()

    self.titik_hilal_kb.set_xdata([x])
    self.titik_hilal_kb.set_ydata([y])

# ---> TAMBAHAN: Update Isi Angka & Posisi Kotak Info <---
# Logika pintar: jika bintang terlalu tinggi (dekat batas atas grafik),
# teks ditaruh di bawah bintang agar tidak terpotong (keluar layar).
offset_y = -1.2 if y > 13 else 0.6
self.teks_info_bintang.set_position((x, y + offset_y))
self.teks_info_bintang.set_text(f"Eln: {x:.2f}° | Alt: {y:.2f}°")
# -----

if x < 7:
    status_msg = "STATUS: MUSTAHIL (Berada di Bawah Batas Danjon - Fisis)"
    color_msg = '#FF5252'
    penjelasan_msg = ("ALASAN: Sabit bulan secara fisis belum terbentuk. Pada sudut elongasi di
bawah 7 derajat,\n"
        "bayangan pegunungan di permukaan bulan memutus pantulan cahaya matahari (Limit
Danjon).\n"
        "Pengamatan hilal pada area ini mustahil dilakukan secara optik maupun fisis.")
    elif x >= 7 and y < 5:
        status_msg = "STATUS: TIDAK TERLIHAT (Kalah Terang oleh Bias Senja - Optis)"
        color_msg = '#FFB74D'
        penjelasan_msg = ("ALASAN: Sabit mungkin sudah terbentuk secara fisis, namun ketinggian di
bawah 5 derajat\n"
            "membuat hilal kalah terang oleh cahaya senja (Twilight Glare) di atmosfer bumi.\n"
            "Kontras cahaya sabit tidak mencukupi untuk mengalahkan bias cahaya ufuk barat.")
    elif 7 <= x < 8 and y >= 5:
        status_msg = "STATUS: RAWAN (Masuk dalam Margin Keamanan Observasi)"
        color_msg = '#FFF176'

```

```

        penjelasan_msg = ("ALASAN: Berada di zona transisi. Meskipun ketinggian di atas ufuk sudah
cukup,\n"
                        "elongasi di bawah 8 derajat dianggap belum aman dari ketidakteraturan permukaan
bulan.\n"
                        "Kriteria KHGT menetapkan 8 derajat sebagai batas margin keamanan (Safety
Margin).")
    else:
        status_msg = "STATUS: RUKYAT BIL 'ILMI (Hilal Sah & Awal Bulan Serentak!)"
        color_msg = '#00E676'
        penjelasan_msg = ("ALASAN: Memenuhi kriteria global Istanbul 2016. Pada posisi ini, hilal
memiliki elongasi\n"
                        "yang cukup untuk membentuk sabit utuh dan ketinggian yang cukup untuk
mengalahkan bias\n"
                        "cahaya senja. Secara saintifik terbukti wujud (Rukyat bil 'Ilmi).")

```

```

self.kb_status_label.configure(text=status_msg, text_color=color_msg)
self.kb_alasan_label.configure(text=penjelasan_msg)
self.canvas_kb.draw_idle()

```

```

def calculate_kriteria_batas(self):

```

```

    """Menghitung ketinggian dan elongasi real-time berdasarkan input di Menu 31"""

```

```

    try:

```

```

        y = int(self.entry_kb_year.get())
        m = int(self.entry_kb_month.get())
        d = int(self.entry_kb_day.get())

```

```

        self.auto_switch_ephemeris(y)

```

```

        lat = float(self.entry_kb_lat.get())
        lon = float(self.entry_kb_lon.get())
        elev = float(self.entry_kb_elev.get())
        tz = float(self.entry_kb_tz.get())

```

```

        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
        lokasi_obs = wgs84.latlon(lat, lon, elevation_m=elev)

```

```

        # Cari Waktu Maghrib (Sunset)
        t0 = self.ts.utc(y, m, d, 4 - int(tz))
        t1 = self.ts.utc(y, m, d, 24 - int(tz))

```

```

        t_sunset = None
        t_evs, y_evs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, lokasi_obs))
        for t_ev, is_sunrise in get_safe_events(t_evs, y_evs):
            if not is_sunrise:
                t_sunset = t_ev
                break

```

```

        if t_sunset is None:

```

```

        raise ValueError("Matahari tidak terbenam pada tanggal dan lokasi tersebut (Anomali Kutub).")

# Kalkulasi Geosentris (Karena Kriteria KHGT menggunakan Geosentrik)
geo_earth = earth.at(t_sunset)
app_moon_geo = geo_earth.observe(moon).apparent()
app_sun_geo = geo_earth.observe(sun).apparent()

ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)

gast = t_sunset.gast
lst_deg = (gast * 15.0) + lon

# Rumus Manual Altitude Geosentris
ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
ha_deg = lst_deg - (ra_h * 15.0)
lat_rad, ha_rad = math.radians(lat), math.radians(ha_deg)
sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)

alt_moon_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
elong_geo = app_sun_geo.separation_from(app_moon_geo).degrees

# Lempar ke UI (Atur Slider sesuai hisab real-time, grafik akan otomatis mengikuti karena ada
fungsi Trace)
self.after(0, lambda: self.var_kb_elongasi.set(round(elong_geo, 2)))
self.after(0, lambda: self.var_kb_ketinggian.set(round(alt_moon_geo, 2)))
self.after(0, lambda: self.lbl_status.configure(text="Data Real-Time berhasil dimuat!",
text_color="#00E676"))
self.after(0, lambda: self.btn_hitung.configure(state="normal"))

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Gagal menghitung kriteria
batas:\n{str(e)}\n\n{traceback.format_exc()}")

# =====
# MODUL 31: KALKULATOR ASTROFOTOGRAFI & KONTRAS
# =====
def setup_menu_31_astrofotografi(self):
    # Bersihkan frame utama (Sesuaikan dengan cara Anda membersihkan frame)
    for widget in self.main_frame.winfo_children():
        widget.destroy()

# Header Modul
header_frame = ctk.CTkFrame(self.main_frame, fg_color="#1a1a2e", corner_radius=10)
header_frame.pack(fill="x", padx=15, pady=10)

```

```

        ctk.CTkLabel(header_frame, text="🌤️ Modul 31: Astrofotografi Hilal & Visibilitas Kontras",
font=("Segoe UI", 18, "bold"), text_color="#00E5FF").pack(pady=10)

# Frame Input Data
input_frame = ctk.CTkFrame(self.main_frame, fg_color="transparent")
input_frame.pack(fill="x", padx=15, pady=5)

# Input Tanggal
ctk.CTkLabel(input_frame, text="Tanggal (YYYY-MM-DD):", font=("Segoe UI", 12)).grid(row=0,
column=0, padx=5, pady=5, sticky="e")
self.entry_astro_date = ctk.CTkEntry(input_frame, width=120)
self.entry_astro_date.grid(row=0, column=1, padx=5, pady=5)
now = datetime.datetime.now()
self.entry_astro_date.insert(0, f"{now.year}-{now.month:02d}-{now.day:02d}")

# Tombol Hitung
btn_hitung_astro = ctk.CTkButton(input_frame, text="Hitung Jendela Astrofotografi",
command=self.hitung_astrofotografi, fg_color="#E65100", hover_color="#EF6C00")
btn_hitung_astro.grid(row=0, column=2, padx=15, pady=5)

# Area Teks Laporan
self.txt_astro_report = ctk.CTkTextbox(self.main_frame, width=600, height=350, font=("Consolas",
14), fg_color="#0f0f1a", text_color="FFFFFF")
self.txt_astro_report.pack(fill="both", expand=True, padx=15, pady=10)
self.txt_astro_report.insert("0.0", "Tekan 'Hitung' untuk menganalisis kontras hilal dan Danjon
Limit...")

# =====
# HELPER CUACA UNIVERSAL (ANTI-BUG TKINTER)
# =====
def fetch_and_update_weather(self, lat, lon, target_label):
    def task():
        try:
            import requests
            # Update status awal (menggunakan keyword argument agar aman)
            self.after(0, lambda lbl=target_label: lbl.configure(text="Mengambil data satelit...",
text_color="#00E5FF"))

            url = f"https://api.open-
meteo.com/v1/forecast?latitude={lat}&longitude={lon}&current=cloud_cover&timezone=auto"
            res = requests.get(url, timeout=5)

            if res.status_code == 200:
                cc = res.json()['current']['cloud_cover']
                if cc > 80:
                    teks, warna = f"🔴 AWAN {cc}% - Cuaca Buruk!", "#FF5252"
                elif cc > 50:

```

```

        teks, warna = f" 🟡 AWAN {cc}% - Ufuk Berawan", "#FFD54F"
    else:
        teks, warna = f" 🟢 CERAH: Awan {cc}% - Aman", "#00E676"

    # Mencegah Bug Closure dengan binding parameter ke fungsi lambda (t=teks, w=warna)
    self.after(0, lambda lbl=target_label, t=teks, w=warna: lbl.configure(text=t, text_color=w))
    else:
        self.after(0, lambda lbl=target_label: lbl.configure(text=" ⚪ Gagal akses API cuaca",
text_color="#9E9E9E"))
        except Exception as e:
            self.after(0, lambda lbl=target_label: lbl.configure(text=" ⚪ Data Cuaca Offline",
text_color="#9E9E9E"))

import threading
threading.Thread(target=task, daemon=True).start()

# =====
# MODUL 31: KALKULATOR ASTROFOTOGRAFI & KONTRAS
# =====
def calculate_astrofotografi(self):
    try:
        import numpy as np
        from skyfield import almanac
        from skyfield.api import wgs84
        import datetime

        y = int(self.entry_astro_year.get())
        m = int(self.entry_astro_month.get())
        d = int(self.entry_astro_day.get())
        lat = float(self.entry_astro_lat.get())
        lon = float(self.entry_astro_lon.get())
        tz = float(self.entry_astro_tz.get())

        # =====
        # TAMBAHAN API CUACA (SAMA PERSIS DENGAN MENU 1)
        # =====
        try:
            # 1. Tampilkan status loading di Sidebar & Layar Utama
            self.after(0, lambda: self.lbl_cuaca_astro.configure(text="Mengambil data cuaca...",
text_color="#00E5FF"))
            self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...",
text_color="#00E5FF"))

            # 2. Panggil fungsi cuaca bawaan aplikasi
            status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)

            # 3. Update hasilnya ke UI

```

```

        self.after(0, lambda: self.lbl_cuaca_astro.configure(text=status_cuaca,
text_color=warna_cuaca))
        self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca,
text_color=warna_cuaca))
    except Exception:
        self.after(0, lambda: self.lbl_cuaca_astro.configure(text="○ Gagal memuat cuaca",
text_color="#9E9E9E"))
        self.after(0, lambda: self.lbl_status_cuaca.configure(text="○ Gagal memuat cuaca",
text_color="#9E9E9E"))
    # =====

    self.auto_switch_ephemeris(y)

    earth = self.eph['earth']
    sun = self.eph['sun']
    moon = self.eph['moon']

    loc = wgs84.latlon(lat, lon)
    topos_obs = earth + loc

    t0 = self.ts.utc(y, m, d, -int(tz))
    t1 = self.ts.utc(y, m, d, 24 - int(tz))
    f_rs = almanac.sunrise_sunset(self.eph, loc)
    t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

    tt_vals = np.atleast_1d(t_rs.tt)
    y_vals = np.atleast_1d(y_rs)

    t_sunset = None
    for tt_val, y_ev in zip(tt_vals, y_vals):
        if y_ev == 0:
            t_sunset = self.ts.tt(jd=tt_val)
            break

    if t_sunset is None:
        self.after(0, self.display_error, "Matahari tidak terbenam pada tanggal ini di lokasi tersebut.")
        return

    app_moon = topos_obs.at(t_sunset).observe(moon).apparent()
    app_sun = topos_obs.at(t_sunset).observe(sun).apparent()

    alt_moon, az_moon, _ = app_moon.altaz()
    alt_sun, az_sun, _ = app_sun.altaz()

    elongation = app_sun.separation_from(app_moon).degrees

    if elongation < 7.0:

```

```

        danjon_status = f" 🚫 GAGAL: Elongasi {elongation:.2f}° (< 7°).\n Cahaya sabit belum terbentuk
(tertutup bayang-bayang kawah bulan)."
        elif elongation < 10.0:
            danjon_status = f" ⚠️ KRITIS: Elongasi {elongation:.2f}°. Perlu Teleskop + Kamera CCD
(Inframerah)."
        else:
            danjon_status = f" 🟢 OPTIMAL: Elongasi {elongation:.2f}°. Hilal cukup terang untuk
astrofotografi."

    t_end_golden = t_sunset.tt + (24.0 / (24 * 60))
    durasi_menit = round((t_end_golden - t_sunset.tt) * 24 * 60)

    dt_sunset = t_sunset.utc_datetime() + datetime.timedelta(hours=tz)

    report = f"{self.get_header(80)}\n"
    report += f"{'[ Astrofotografi Hilal & Visibilitas Kontras ]'.center(80)}\n\n"
    report += f"Tanggal Observasi : {d:02d}-{m:02d}-{y}\n"
    report += f"Waktu Sunset Lokal: {dt_sunset.strftime('%H:%M:%S')}\n"
    report += "-" * 80 + "\n"
    report += f"1. Ketinggian Hilal : {alt_moon.degrees:.2f}° (Toposentrik)\n"
    report += f"2. Batas Optik Danjon : {danjon_status}\n\n"
    report += "--- REKOMENDASI PEMOTRETAN ---\n"
    if alt_moon.degrees < 0:
        report += "Bulan terbenam lebih dulu sebelum Matahari. Tidak bisa difoto.\n"
    else:
        report += f"Mulai Bidik Kamera : {dt_sunset.strftime('%H:%M:%S')} (Saat Sunset)\n"
        report += f"Durasi Golden Hour : ~{durasi_menit} Menit pertama setelah sunset.\n"

    self.after(0, self.display_result, report)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Error:\n{str(e)}\n\n{traceback.format_exc()}")

# =====
# MODUL 32: PREDIKTOR PASANG SURUT GRAVITASI
# =====
def calculate_pasang_surut(self):
    try:
        import datetime
        import math
        from skyfield.api import wgs84

        y = int(self.entry_pasang_year.get())
        m = int(self.entry_pasang_month.get())
        d = int(self.entry_pasang_day.get())
        lat = float(self.entry_pasang_lat.get())

```

```

lon = float(self.entry_pasang_lon.get())
tz = float(self.entry_pasang_tz.get())

# =====
# TAMBAHAN API CUACA (SAMA PERSIS DENGAN MENU 1)
# =====
try:
    # 1. Tampilkan status loading di Sidebar & Layar Utama
    self.after(0, lambda: self.lbl_cuaca_pasang.configure(text="Mengambil data cuaca...",
text_color="#00E5FF"))
    self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...",
text_color="#00E5FF"))

    # 2. Panggil fungsi cuaca bawaan aplikasi
    status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)

    # 3. Update hasilnya ke UI
    self.after(0, lambda: self.lbl_cuaca_pasang.configure(text=status_cuaca,
text_color=warna_cuaca))
    self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca,
text_color=warna_cuaca))
except Exception:
    self.after(0, lambda: self.lbl_cuaca_pasang.configure(text="○ Gagal memuat cuaca",
text_color="#9E9E9E"))
    self.after(0, lambda: self.lbl_status_cuaca.configure(text="○ Gagal memuat cuaca",
text_color="#9E9E9E"))
# =====

self.auto_switch_ephemeris(y)

earth = self.eph['earth']
sun = self.eph['sun']
moon = self.eph['moon']

t_eval = self.ts.utc(y, m, d, 12 - int(tz))
loc = wgs84.latlon(lat, lon)
topos_obs = earth + loc

obs_moon = topos_obs.at(t_eval).observe(moon)
obs_sun = topos_obs.at(t_eval).observe(sun)

app_moon = obs_moon.apparent()
app_sun = obs_sun.apparent()

dist_moon_km = app_moon.distance().km
dist_sun_km = app_sun.distance().km
elongation = app_sun.separation_from(app_moon).degrees

```



```

if elongation < 25 or elongation > 335:
    fase = "New Moon (Ijtimak / Bulan Baru)"
    jenis_pasang = "🔴 PASANG PURNAMA (Spring Tide)"
    desc_pasang = "Gaya tarik Bulan dan Matahari SEARAH. Potensi pasang laut MAKSIMAL."
elif 155 < elongation < 205:
    fase = "Full Moon (Bulan Purnama)"
    jenis_pasang = "🔴 PASANG PURNAMA (Spring Tide)"
    desc_pasang = "Gaya tarik Bulan dan Matahari SEJAJAR. Potensi pasang laut MAKSIMAL."
elif 65 < elongation < 115 or 245 < elongation < 295:
    fase = "Kuartir (Bulan Paruh)"
    jenis_pasang = "🟢 PASANG PERBANI (Neap Tide)"
    desc_pasang = "Gaya tarik Bulan dan Matahari TEGAK LURUS. Kondisi pasang laut AMAN."
else:
    fase = "Fase Transisi (Crescent/Gibbous)"
    jenis_pasang = "🟡 PASANG MENENGAH (Normal Tide)"
    desc_pasang = "Kondisi pasang laut berada di batas normal."

if dist_moon_km < 365000:
    status_jarak = f"PERIGEE (Sangat Dekat: {dist_moon_km:,.0f} km)"
    if "PURNAMA" in jenis_pasang:
        desc_pasang += "\n ⚠️ BAHAYA: Terjadi KING TIDE (Pasang Perigean)!\n Waspada banjir
rob dan ombak ekstrem di pesisir pantai."
    elif dist_moon_km > 404000:
        status_jarak = f"APOGEE (Sangat Jauh: {dist_moon_km:,.0f} km)"
    else:
        status_jarak = f"Normal ({dist_moon_km:,.0f} km)"

moon_tide_index = (384400 / dist_moon_km)**3 * 100

report = f"{self.get_header(80)}\n"
report += f"{'[ Prediktor Pasang Surut Gravitasi Pantai (Tidal Force) ]'.center(80)}\n\n"
report += f"Tanggal Analisis : {d:02d}-{m:02d}-{y} (Tinjauan Siang Lokal)\n"
report += f"Koordinat Pantai : Lintang {lat}, Bujur {lon}\n"
report += "-" * 80 + "\n"

report += f"A. DATA ASTRONOMI FISIK\n"
report += f" - Elongasi Bulan-Matahari : {elongation:.2f}° ({fase})\n"
report += f" - Jarak Bumi-Bulan : {status_jarak}\n"
report += f" - Jarak Bumi-Matahari : {dist_sun_km:,.0f} km\n"
report += f" - Tarikan Gravitasi Bulan : {moon_tide_index:.1f}% (vs Rata-rata)\n\n"

report += f"B. STATUS KEAMANAN PESISIR & RUKYAT\n"
report += f" - Tipe Pasang Laut : {jenis_pasang}\n"
report += f" - Deskripsi Dampak : {desc_pasang}\n\n"

report += "Catatan: Prediksi ini bersifat astronomis murni (Efek Gravitasi Ekstraterestrial).\n"

```

```

report += "Waktu aktual puncak pasang di pesisir juga bergantung pada angin laut/topografi.\n"

self.after(0, self.display_result, report)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Error:\n{str(e)}\n\n{traceback.format_exc()}")

# =====
# MODUL 33: EVALUASI FAJAR SHADIQ & SQM TEORITIS
# =====
def calculate_fajar_sqm(self):
    try:
        import numpy as np
        from skyfield import almanac
        from skyfield.api import wgs84
        import datetime

        y = int(self.entry_fajar_year.get())
        m = int(self.entry_fajar_month.get())
        d = int(self.entry_fajar_day.get())
        lat = float(self.entry_fajar_lat.get())
        lon = float(self.entry_fajar_lon.get())
        tz = float(self.entry_fajar_tz.get())

        # =====
        # INTEGRASI API CUACA
        # =====
        try:
            self.after(0, lambda: self.lbl_cuaca_fajar.configure(text="Mengambil data cuaca...",
            text_color="#00E5FF"))
            self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...",
            text_color="#00E5FF"))

            status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)

            self.after(0, lambda: self.lbl_cuaca_fajar.configure(text=status_cuaca,
            text_color=warna_cuaca))
            self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca,
            text_color=warna_cuaca))
        except Exception:
            self.after(0, lambda: self.lbl_cuaca_fajar.configure(text="○ Gagal memuat cuaca",
            text_color="#9E9E9E"))
            self.after(0, lambda: self.lbl_status_cuaca.configure(text="○ Gagal memuat cuaca",
            text_color="#9E9E9E"))
        # =====

```

```

self.auto_switch_ephemeris(y)

earth = self.eph['earth']
sun = self.eph['sun']

loc = wgs84.latlon(lat, lon)
topos_obs = earth + loc

# Rentang waktu pencarian: 00:00 s.d 24:00 waktu lokal
t0 = self.ts.utc(y, m, d, -int(tz))
t1 = self.ts.utc(y, m, d, 24 - int(tz))

# Fungsi Kustom Skyfield untuk melacak ketinggian spesifik (-18° dan -20°)
def f_angle(angle_deg):
    def _sun_alt(t):
        alt, az, distance = topos_obs.at(t).observe(sun).apparent().altaz()
        return alt.degrees >= angle_deg
    _sun_alt.step_days = 0.04
    return _sun_alt

f_18 = f_angle(-18.0)
f_20 = f_angle(-20.0)

# Mencari event (1 = Matahari naik melewati batas, 0 = Matahari turun melewati batas)
t_18_ev, y_18_ev = almanac.find_discrete(t0, t1, f_18)
t_20_ev, y_20_ev = almanac.find_discrete(t0, t1, f_20)

fajar_18 = None
fajar_20 = None
isya_18 = None

# Anti-Bug Single Time dengan np.atleast_1d
for t_tt, y_val in zip(np.atleast_1d(t_18_ev.tt), np.atleast_1d(y_18_ev)):
    if y_val == 1 and fajar_18 is None: fajar_18 = self.ts.tt(jd=t_tt) # Matahari Naik (Pagi/Fajar)
    if y_val == 0 and isya_18 is None: isya_18 = self.ts.tt(jd=t_tt) # Matahari Turun (Malam/Isya)

for t_tt, y_val in zip(np.atleast_1d(t_20_ev.tt), np.atleast_1d(y_20_ev)):
    if y_val == 1 and fajar_20 is None: fajar_20 = self.ts.tt(jd=t_tt)

def format_time(t_obj):
    if t_obj is None: return "Tidak Terjadi/Extrem"
    dt = t_obj.utc_datetime() + datetime.timedelta(hours=tz)
    return dt.strftime('%H:%M:%S')

# Format Laporan
report = f"{{self.get_header(80)}}\n"
report += f"{{'[ Evaluasi Fajar Shadiq & SQM Teoritis ]'.center(80)}}\n\n"
report += f"Tanggal Observasi : {{d:02d}}-{{m:02d}}-{{y}}\n"

```

```

report += f"Koordinat Lokasi : Lat {lat}, Lon {lon}\n"
report += "-" * 80 + "\n\n"

report += "A. KOMPARASI WAKTU FAJAR SHADIQ (SUBUH)\n"
report += f" - Kemenag RI (-20°) : {format_time(fajar_20)} Waktu Lokal\n"
report += f" - Muhammadiyah/MABIMS (-18°): {format_time(fajar_18)} Waktu Lokal\n"

# ---> PERBAIKAN DI SINI: Gunakan 'is not None' <---
if fajar_18 is not None and fajar_20 is not None:
    selisih_fajar = round(abs((fajar_18.tt - fajar_20.tt) * 24 * 60))
    report += f" * Selisih waktu subuh : ~{selisih_fajar} menit\n\n"
else:
    report += f" * Selisih waktu subuh : Tidak dapat dihitung\n\n"

report += "B. WAKTU ISYA (-18°)\n"
report += f" - Awal Waktu Isya : {format_time(isya_18)} Waktu Lokal\n\n"

report += "C. PREDIKSI NILAI SQM (Sky Quality Meter) TEORITIS\n"
report += " - Saat Matahari -18° : ~21.0 - 21.5 mag/arcsec2 (Awal Fajar Astronomis)\n"
report += " - Saat Matahari -15° : ~19.0 - 19.5 mag/arcsec2 (Langit mulai teramati biru gelap)\n"
report += " - Saat Matahari -12° : ~17.0 - 17.5 mag/arcsec2 (Batas Nautical Twilight)\n"
report += " - Saat Matahari -6° : ~13.0 - 14.0 mag/arcsec2 (Batas Civil Twilight)\n\n"

report += "KESIMPULAN & CATATAN PENGUKURAN SQM:\n"
report += "1. Pengukuran dengan alat SQM fisik sangat dipengaruhi oleh polusi cahaya kota.\n"
report += "2. Angka referensi di atas berlaku HANYA untuk lokasi pengamatan yang\n"
report += " benar-benar gelap (Bortle Class 1-2) tanpa gangguan cahaya bulan purnama.\n"
report += "3. Di perkotaan, nilai SQM sering kali tertahan di angka 17 - 18 karena polusi.\n"

self.after(0, self.display_result, report)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Error:\n{str(e)}\n\n{traceback.format_exc()}")

# =====
# MODUL 34: GENERATOR ANALEMMA MATAHARI
# =====
def calculate_analemma(self):
    try:
        import numpy as np
        from skyfield.api import wgs84
        import datetime

        # Ambil data dari input sidebar
        y = int(self.entry_analemma_year.get())
        h_local = float(self.entry_analemma_hour.get())
        lat = float(self.entry_analemma_lat.get())

```

```

lon = float(self.entry_analemma_lon.get())
tz = float(self.entry_analemma_tz.get())

# =====
# INTEGRASI API CUACA (UPDATE INSTAN)
# =====
try:
    self.after(0, lambda: self.lbl_cuaca_analemma.configure(text="Mengambil data cuaca...",
text_color="#00E5FF"))
    self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...",
text_color="#00E5FF"))

    status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)

    self.after(0, lambda: self.lbl_cuaca_analemma.configure(text=status_cuaca,
text_color=warna_cuaca))
    self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca,
text_color=warna_cuaca))
except Exception:
    self.after(0, lambda: self.lbl_cuaca_analemma.configure(text="○ Gagal memuat cuaca",
text_color="#9E9E9E"))
# =====

# --- LOGIKA ASTRONOMI ANALEMMA ---
self.auto_switch_ephemeris(y)
earth = self.eph['earth']
sun = self.eph['sun']
loc = earth + wgs84.latlon(lat, lon)

alts, azs, dates = [], [], []

# Memproses lintasan setahun penuh (setiap 5 hari)
for day in range(0, 365, 5):
    date_dt = datetime.datetime(y, 1, 1) + datetime.timedelta(days=day)
    t = self.ts.utc(y, date_dt.month, date_dt.day, h_local - tz)

    astrometric = loc.at(t).observe(sun).apparent()
    alt, az, _ = astrometric.altaz()

    alts.append(alt.degrees)
    azs.append(az.degrees)
    dates.append(date_dt)

idx_max = np.argmax(alts)
idx_min = np.argmin(alts)

report = f"{self.get_header(80)}\n"

```

```

report += f"{'[ ANALISIS LINTASAN ANALEMMA MATAHARI ]'.center(80)}\n"
report += f"{'Pengamatan Tahunan pada Jam {h_local:02.0f}:00 Lokal'.center(80)}\n"
report += "-" * 80 + "\n\n"

report += f"A. TITIK EKSTRIM ANALEMMA (TAHUN {y})\n"
report += f" 1. Posisi Tertinggi : {dates[idx_max].strftime('%d %B')} -> Alt: {alts[idx_max]:.2f}°\n"
report += f" 2. Posisi Terendah : {dates[idx_min].strftime('%d %B')} -> Alt:
{alts[idx_min]:.2f}°\n\n"

report += "B. DATA SAMPEL LINTASAN BULANAN\n"
report += f"{'Tanggal':<20} | {'Ketinggian (Alt)':<18} | {'Arah (Azimuth)':<15}\n"
report += "-" * 60 + "\n"
for i in range(0, len(dates), 6):
    report += f"{'dates[i].strftime('%d %b %Y'):<20} | {alts[i]:>15.2f}° | {azs[i]:>12.2f}°\n"

report += "\nAnalemma membentuk angka 8 karena orbit Bumi yang elips dan kemiringan sumbu
rotasi.\n"
report += "Hal ini membuktikan perbedaan antara Waktu Matahari Sejati dan Waktu Rata-rata."

# Lempar ke GUI (Main Thread) agar merender teks beserta grafiknya
self.after(0, self.display_analemma_result, report, azs, alts, y)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Gagal menghitung
Analemma:\n{str(e)}\n\n{traceback.format_exc()}")

def setup_analemma_out_frame(self):
    """Mempersiapkan area output scrollable untuk teks dan grafik Analemma Menu 34"""
    self.frame_analemma_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#050510",
corner_radius=10)

    # Kotak Teks untuk Laporan
    self.txt_analemma_report = ctk.CTkTextbox(self.frame_analemma_out, height=350,
font=("Consolas", 13), fg_color="#101010", text_color="#00E5FF", wrap="none")
    self.txt_analemma_report.pack(fill="x", padx=15, pady=(15, 5))

    # Frame untuk menampung Grafik Matplotlib
    self.frame_analemma_canvas = ctk.CTkFrame(self.frame_analemma_out, fg_color="#181818")
    self.frame_analemma_canvas.pack(fill="both", expand=True, padx=15, pady=(5, 15))

def display_analemma_result(self, report_text, azs, alts, year):
    """Memasukkan teks ke textbox dan menggambar kurva Analemma"""
    # 1. Update Textbox
    self.txt_analemma_report.configure(state="normal")
    self.txt_analemma_report.delete("1.0", "end")
    self.txt_analemma_report.insert("1.0", report_text)
    self.txt_analemma_report.configure(state="disabled")

```

```

# 2. Render Grafik Matplotlib
for widget in self.frame_analemma_canvas.wininfo_children():
    widget.destroy()

from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np

fig = Figure(figsize=(8, 6), dpi=100)
fig.patch.set_facecolor('#181818')
ax = fig.add_subplot(111)
ax.set_facecolor('#101010')

# Plot garis Analemma (angka 8)
ax.plot(azs, alts, color='#FFD54F', marker='o', markersize=4, linestyle='-', linewidth=1.5, label='Jalur
Analemma')

# Tandai posisi tertinggi dan terendah
idx_max = np.argmax(alts)
idx_min = np.argmin(alts)
ax.plot(azs[idx_max], alts[idx_max], 'r^', markersize=10, label='Tertinggi (Summer Solstice)')
ax.plot(azs[idx_min], alts[idx_min], 'bv', markersize=10, label='Terendah (Winter Solstice)')

ax.set_title(f'Kurva Analemma Matahari (Tahun {year})', color='white', fontsize=14,
fontweight='bold', pad=15)
ax.set_xlabel("Azimuth Matahari (°)", color='ffffff', fontsize=12)
ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)
ax.tick_params(colors='ffffff', labels=11)
ax.grid(True, color='ffffff', linestyle=':', alpha=0.3)
ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white')
fig.tight_layout()

canvas = FigureCanvasTkAgg(fig, master=self.frame_analemma_canvas)
canvas.draw()
canvas.get_tk_widget().pack(fill="both", expand=True, pady=10)

# Selesaikan Status
self.lbl_status.configure(text="Kalkulasi & Grafik Analemma Selesai", text_color="#00E676")
self.btn_hitung.configure(state="normal")

def calculate_occultation(self):
    """Memicu pencarian okultasi di background thread agar aplikasi tidak macet"""
    threading.Thread(target=self._proses_okultasi, daemon=True).start()

def _proses_okultasi(self):
    self.after(0, lambda: self.lbl_status.configure(text="Memindai Okultasi Bulan & Planet...",
text_color="#00E5FF"))

```

```

self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
self.after(0, lambda: self.btn_cari_occ.configure(state="disabled"))
self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
self.after(0, lambda: self.textbox.delete("1.0", "end"))
self.after(0, lambda: self.textbox.insert("1.0", "Menganalisis lintasan Bulan dan Target Planet dalam
sebulan penuh...\nMohon tunggu...\n\n"))

```

```

try:
    import numpy as np
    from skyfield import almanac
    from skyfield.api import wgs84
    import math
    import datetime

    # Ambil data input Astrofotografi
    y = int(self.entry_astro_year.get())
    m = int(self.entry_astro_month.get())
    lat = float(self.entry_astro_lat.get())
    lon = float(self.entry_astro_lon.get())
    tz = float(self.entry_astro_tz.get())
    elev = float(self.entry_astro_elev.get())
    target_name = self.combo_astro_occ_target.get().lower()

    self.auto_switch_ephemeris(y)
    earth = self.eph['earth']
    moon = self.eph['moon']

    # Skyfield memakai kata "barycenter" untuk tata surya besar
    target_key = f'{target_name} barycenter' if target_name not in ['mercury', 'venus'] else
target_name
    try:
        planet = self.eph[target_key]
    except KeyError:
        planet = self.eph[target_name]

    loc = wgs84.latlon(lat, lon, elevation_m=elev)

    # Rentang pencarian: 1 bulan penuh
    _, num_days = safe_moonrange(y, m)
    t0 = self.ts.utc(y, m, 1, -int(tz))
    t1 = self.ts.utc(y, m, num_days, 24 - int(tz))

    # FUNGSI DETEKTOR OKULTASI
    # Menghitung jika jarak pisah (separasi) lebih kecil dari gabungan Radius Bulan + Radius Planet
    def occultation_func(t):
        obs = (earth + loc).at(t)
        m_app = obs.observe(moon).apparent()
        p_app = obs.observe(planet).apparent()

```



```

sep = m_app.separation_from(p_app).degrees

m_dist = m_app.distance().km
p_dist = p_app.distance().km

# Kalkulasi Radius Semu di Langit
m_rad = np.degrees(np.arcsin(1737.4 / m_dist))
radii = {'mercury': 2439.7, 'venus': 6051.8, 'mars': 3389.5, 'jupiter': 69911.0, 'saturn': 58232.0,
'uranus': 25362.0, 'neptune': 24622.0}
p_rad = np.degrees(np.arcsin(radii[target_name] / p_dist))

return sep < (m_rad + p_rad)

# Resolusi pencarian ~7 menit agar sangat akurat mendeteksi masuk/keluar planet
occultation_func.step_days = 0.005

# Lacak menggunakan Almanak
t_occ, y_occ = almanac.find_discrete(t0, t1, occultation_func)

output_lines = []
output_lines.append(self.get_header(90))
output_lines.append(f"[ PREDIKTOR OKULTASI: BULAN MENUTUPI {target_name.upper()}
]".center(90))
output_lines.append("=" * 90)

try:
    nama_kota = self.opt_city.get()
    nama_prov = self.opt_prov.get()
    lokasi_text = f"{nama_kota}, {nama_prov}"
except:
    lokasi_text = f"Lat {lat}, Lon {lon}"

output_lines.append(f"Lokasi Observasi : {lokasi_text}")
output_lines.append(f"Rentang Waktu   : Bulan {m:02d} Tahun {format_tahun_aman(y)}")
output_lines.append(f"Zona Waktu       : UTC{'+' if tz>=0 else ''}{tz}")
output_lines.append("-" * 90)

if len(t_occ) == 0:
    output_lines.append(f"\n>> TIDAK ADA okultasi {target_name.capitalize()} oleh Bulan yang
teramati")
    output_lines.append(f"   dari lokasi Anda pada bulan {m:02d}/{format_tahun_aman(y)}.")
else:
    # Group Ingress (Masuk) dan Egress (Keluar)
    events = []
    current_ingress = None

    for t_ev, state in zip(t_occ, y_occ):
        if state == 1: # Ingress (Mulai Tertutup / Kontak 1)

```

```

        current_ingress = t_ev
    elif state == 0 and current_ingress is not None: # Egress (Mulai Muncul Kembali / Kontak 2)
        events.append((current_ingress, t_ev))
        current_ingress = None

    if current_ingress is not None:
        # Okultasi terjadi dan menyeberang ke bulan berikutnya
        events.append((current_ingress, None))

    for idx, ev in enumerate(events, 1):
        t_in, t_eg = ev

        # Cek Altitude & Iluminasi saat Ingress
        obs_in = (earth + loc).at(t_in).observe(moon).apparent()
        alt_in = obs_in.altaz()[0].degrees

        dt_in = t_in.utcdatetime() + datetime.timedelta(hours=tz)
        str_in = dt_in.strftime("%d-%b-%Y %H:%M:%S")

        if t_eg is not None:
            obs_eg = (earth + loc).at(t_eg).observe(moon).apparent()
            alt_eg = obs_eg.altaz()[0].degrees
            dt_eg = t_eg.utcdatetime() + datetime.timedelta(hours=tz)
            str_eg = dt_eg.strftime("%d-%b-%Y %H:%M:%S")
            alt_eg_str = f"{alt_eg:.2f}°"
            durasi = round((t_eg.tt - t_in.tt) * 24 * 60, 1)
            durasi_str = f"{durasi} Menit"
        else:
            str_eg = "Terpotong batas bulan"
            alt_eg_str = "N/A"
            durasi_str = "N/A"

        # Iluminasi Bulan (Memudahkan kamera mengatur ISO/Shutter Speed)
        illum = almanac.fraction_illuminated(self.eph, 'moon', t_in)
        illum_val = (illum.item() if hasattr(illum, 'item') else illum) * 100.0

        output_lines.append(f"\n[ PERISTIWA OKULTASI #{idx} ]")
        output_lines.append(f" FASE INGRESS (Mulai Tertutup) : {str_in} | Alt Bulan: {alt_in:.2f}°")
        output_lines.append(f" FASE EGRESS (Muncul Kembali) : {str_eg} | Alt Bulan: {alt_eg_str}")
        output_lines.append(f" Durasi Tertutup Penuh : {durasi_str}")
        output_lines.append(f" Kondisi Iluminasi Bulan : {illum_val:.1f}%")

        if alt_in < 0:
            output_lines.append(f" ⚠ PERINGATAN: Terjadi saat Bulan berada di BAWAH UFUK (Tidak bisa dirukyat secara fisik).")

        output_lines.append("\n" + "=" * 90)

```

```

        output_lines.append("* Catatan Validasi:")
        output_lines.append("- Prediktor ini memperhitungkan semi-diameter (radius) Bulan dan Planet
secara dinamis")
        output_lines.append(" berdasarkan jarak real-time dari bumi.")
        output_lines.append("- Waktu sangat presisi dan disesuaikan dengan titik Toposentrik lokasi
GPS.")

        final_report = "\n".join(output_lines)
        self.after(0, lambda: self.display_result(final_report))

except Exception as e:
    import traceback
    self.after(0, lambda: self.display_error(f"Gagal memprediksi
Okultasi:\n{str(e)}\n\n{traceback.format_exc()}"))
    finally:
        self.after(0, lambda: self.btn_hitung.configure(state="normal"))
        self.after(0, lambda: self.btn_cari_occ.configure(state="normal"))

# =====
# MODUL EKSKLUSIF: KALKULATOR SHALAT PENERBANGAN (IN-FLIGHT PRAYER TIMES)
# =====
def buka_kalkulator_penerbangan(self):
    self.win_flight = ctk.CTkToplevel(self)
    self.win_flight.title(" ✈ Kalkulator Waktu Shalat Penerbangan (In-Flight)")
    self.win_flight.geometry("850x850")
    self.win_flight.attributes("-topmost", True)
    self.win_flight.configure(fg_color="#0A0A10")

    ctk.CTkLabel(self.win_flight, text="AEROSPACE PRAYER TIMES CALCULATOR", font=("Segoe UI", 18,
"bold"), text_color="#FFD54F").pack(pady=(15, 5))
    ctk.CTkLabel(self.win_flight, text="Kalkulasi Dinamis berbasis Rute Great Circle, Kecepatan, dan
Ketinggian", font=("Segoe UI", 12)).pack(pady=(0, 15))

    # --- FRAME INPUT ---
    input_container = ctk.CTkFrame(self.win_flight, fg_color="transparent")
    input_container.pack(fill="x", padx=20)

    # Kiri: Keberangkatan
    frame_dep = ctk.CTkFrame(input_container, fg_color="#1E1E1E", corner_radius=8)
    frame_dep.pack(side="left", fill="both", expand=True, padx=(0, 10))
    ctk.CTkLabel(frame_dep, text="DEPARTURE (KEBERANGKATAN)", font=("Segoe UI", 12, "bold"),
text_color="#00E5FF").pack(pady=10)
    self.fl_dep_lat = self.create_input_row(frame_dep, "Lat Berangkat:", "-6.1256") # Default CGK
    self.fl_dep_lon = self.create_input_row(frame_dep, "Lon Berangkat:", "106.6558")
    self.fl_dep_tz = self.create_input_row(frame_dep, "Timezone Awal:", "7.0")

```

```

        ctk.CTkLabel(frame_dep, text="Waktu Berangkat (Lokal):", font=("Segoe UI", 11)).pack(anchor="w",
padx=10, pady=(10,0))
        date_frame = ctk.CTkFrame(frame_dep, fg_color="transparent")
        date_frame.pack(fill="x", padx=10, pady=5)
        now = datetime.datetime.now()
        self.fl_y, self.fl_m, self.fl_d = self.create_ymd_row(date_frame, str(now.year), f"{now.month:02d}",
f"{now.day:02d}")
        self.fl_h_m_s = self.create_input_row(frame_dep, "Jam (HH:MM):", "08:00")

# Kanan: Kedatangan & Pesawat
frame_arr = ctk.CTkFrame(input_container, fg_color="#1E1E1E", corner_radius=8)
frame_arr.pack(side="right", fill="both", expand=True, padx=(10, 0))
ctk.CTkLabel(frame_arr, text="DESTINATION & AIRCRAFT", font=("Segoe UI", 12, "bold"),
text_color="#00E676").pack(pady=10)
self.fl_arr_lat = self.create_input_row(frame_arr, "Lat Tujuan:", "21.5433") # Default JED
self.fl_arr_lon = self.create_input_row(frame_arr, "Lon Tujuan:", "39.1728")

        ctk.CTkLabel(frame_arr, text="Spesifikasi Penerbangan:", font=("Segoe UI", 11)).pack(anchor="w",
padx=10, pady=(10,0))
        self.fl_duration = self.create_input_row(frame_arr, "Durasi (Jam):", "9.5")
        self.fl_alt = self.create_input_row(frame_arr, "Ketinggian (Kaki/ft):", "35000")

# Tombol Proses
self.btn_calc_flight = ctk.CTkButton(self.win_flight, text="▶ HITUNG JADWAL SELAMA
PENERBANGAN", font=("Segoe UI", 14, "bold"),
                                fg_color="#D84315", hover_color="#BF360C", height=40,
command=self.proses_kalkulator_penerbangan)
self.btn_calc_flight.pack(fill="x", padx=20, pady=20)

# Kotak Laporan
self.txt_flight_report = ctk.CTkTextbox(self.win_flight, font=("Consolas", 13), fg_color="#101010",
text_color="#E0E0E0", wrap="none")
self.txt_flight_report.pack(fill="both", expand=True, padx=20, pady=(0, 20))
self.txt_flight_report.insert("1.0", "Menunggu input parameter penerbangan...\nSistem akan
melacak posisi pesawat per menit di udara (Great Circle) dan menghitung penurunan ufuk akibat
altitude pesawat (Horizon Dip).")
self.txt_flight_report.configure(state="disabled")

def proses_kalkulator_penerbangan(self):
    self.btn_calc_flight.configure(state="disabled", text="⌚ MENGKALKULASI RUTE...")
    self.txt_flight_report.configure(state="normal")
    self.txt_flight_report.delete("1.0", "end")
    threading.Thread(target=self._hitung_penerbangan_thread, daemon=True).start()

def _slerp(self, lat1, lon1, lat2, lon2, frac):
    """Spherical Linear Interpolation untuk menghitung posisi pesawat secara presisi di rute
melengkung bumi (Great Circle)"""

```

```

import math
l1, lo1, l2, lo2 = map(math.radians, [lat1, lon1, lat2, lon2])

x1, y1, z1 = math.cos(l1)*math.cos(lo1), math.cos(l1)*math.sin(lo1), math.sin(l1)
x2, y2, z2 = math.cos(l2)*math.cos(lo2), math.cos(l2)*math.sin(lo2), math.sin(l2)

dot = max(-1.0, min(1.0, x1*x2 + y1*y2 + z1*z2))
omega = math.acos(dot)

if omega < 1e-5: return lat1, lon1

sin_omega = math.sin(omega)
a = math.sin((1.0 - frac) * omega) / sin_omega
b = math.sin(frac * omega) / sin_omega

x, y, z = a*x1 + b*x2, a*y1 + b*y2, a*z1 + b*z2
return math.degrees(math.asin(z)), math.degrees(math.atan2(y, x))

def _hitung_penerbangan_thread(self):
    try:
        # 1. Ambil Input
        lat1, lon1 = float(self.fl_dep_lat.get()), float(self.fl_dep_lon.get())
        lat2, lon2 = float(self.fl_arr_lat.get()), float(self.fl_arr_lon.get())
        durasi_jam = float(self.fl_duration.get())
        altitude_ft = float(self.fl_alt.get())
        tz_awal = float(self.fl_dep_tz.get())

        y, m, d = int(self.fl_y.get()), int(self.fl_m.get()), int(self.fl_d.get())
        jam_str = self.fl_h_m_s.get().split(":")
        hr, mn = int(jam_str[0]), int(jam_str[1])

        # Waktu Lokal ke UTC
        dt_dep_lokal = datetime.datetime(y, m, d, hr, mn, 0)
        dt_dep_utc = dt_dep_lokal - datetime.timedelta(hours=tz_awal)

        self.auto_switch_ephemeris(y)
        earth, sun = self.eph['earth'], self.eph['sun']

        # 2. Rumus Dip of Horizon (Penurunan Ufuk di Pesawat)
        # Pada ketinggian 35.000 kaki, ufuk tidak lagi di 0 derajat, tapi anjlok ke bawah.
        # Rumus Dip:  $1.76 * \sqrt{\text{Altitude\_ft} / 3.2808} / 60$ 
        altitude_m = altitude_ft / 3.2808
        dip_deg = (1.76 * math.sqrt(altitude_m)) / 60.0

        # Standar geometri Maghrib/Syuruq:  $-0.833^\circ$  (Refraksi standar + Semi-diameter).
        # Di pesawat, harus ditambah dengan Dip of the Horizon.
        batas_ufuk_pesawat = -0.833 - dip_deg
        batas_fajar_isya = -18.0 # Sudut Fajar/Isya standar astronomis

```

```

# 3. Simulasi Perjalanan Menit per Menit
total_menit = int(durasi_jam * 60)
timeline_events = []

t_dep_sky = self.ts.utc(dt_dep_utc.year, dt_dep_utc.month, dt_dep_utc.day, dt_dep_utc.hour,
dt_dep_utc.minute)

# Pelacakan status agar tidak terjadi cetak berulang
status_siang = None
isya_done, fajar_done = False, False

# Menyimpan rute dan waktu untuk laporan
log_rute = []

for menit_ke in range(total_menit + 1):
    frac = menit_ke / total_menit if total_menit > 0 else 0
    curr_dt_utc = dt_dep_utc + datetime.timedelta(minutes=menit_ke)
    curr_dt_lokal_asal = dt_dep_lokal + datetime.timedelta(minutes=menit_ke)

    # Interpolasi Posisi Pesawat
    curr_lat, curr_lon = self._slerp(lat1, lon1, lat2, lon2, frac)

    # Hitung Posisi Matahari dari Koordinat Pesawat Saat Ini
    t_curr = self.ts.utc(curr_dt_utc.year, curr_dt_utc.month, curr_dt_utc.day, curr_dt_utc.hour,
curr_dt_utc.minute)
    loc = wgs84.latlon(curr_lat, curr_lon, elevation_m=altitude_m)

    # Menghitung ketinggian dan arah kiblat dinamis pesawat
    app_sun = (earth + loc).at(t_curr).observe(sun).apparent()
    alt_sun, _, _ = app_sun.altaz(temperature_C=-50, pressure_mbar=250) # Atmosfer tipis di atas
    alt_deg = alt_sun.degrees

    # Hitung Kiblat Dinamis
    phi_k, lam_k = math.radians(21.4225), math.radians(39.8262)
    phi, lam = math.radians(curr_lat), math.radians(curr_lon)
    y_q = math.sin(lam_k - lam)
    x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
    qibla_dir = (math.degrees(math.atan2(y_q, x_q)) + 360) % 360

    # Cek Kondisi Siang/Malam berdasarkan Ufuk Pesawat
    cek_siang = alt_deg > batas_ufuk_pesawat

    # Catat Log setiap jam untuk tabel jalur
    if menit_ke % 60 == 0 or menit_ke == total_menit:
        log_rute.append(f'{curr_dt_lokal_asal.strftime('%H:%M')} | {curr_lat:>7.2f}°,
{curr_lon:>8.2f}° | Alt Mth: {alt_deg:>6.2f}° | Arah Kiblat: {qibla_dir:>5.1f}°')

```

```

if menit_ke == 0:
    status_siang = cek_siang
    continue

jam_kejadian = curr_dt_lokal_asal.strftime('%H:%M')

# Logika Penyilangan Event
if not status_siang and cek_siang:
    timeline_events.append(f"[{jam_kejadian}] 🌅 TERBIT MATAHARI (Syuruq) - Ketinggian:
{alt_deg:.2f}°")
    status_siang = True
    fajar_done = False # Reset fajar

elif status_siang and not cek_siang:
    timeline_events.append(f"[{jam_kejadian}] 🌇 MAGHRIB (Terbenam) - Ketinggian:
{alt_deg:.2f}° (Ufuk pesawat di {batas_ufuk_pesawat:.2f}°)")
    status_siang = False
    isya_done = False # Reset isya

# Logika Subuh / Isya
if alt_deg >= batas_fajar_isya and not cek_siang and not fajar_done and alt_deg > -18.5 and
alt_deg < batas_ufuk_pesawat:
    # Mendekati fajar (naik dari -18) -> SUBUH
    # Pastikan ini sebelum terbit
    if alt_deg - (-18.0) > 0:
        timeline_events.append(f"[{jam_kejadian}] 🌄 SUBUH (Fajar Shadiq) - Posisi Matahari:
{alt_deg:.2f}°")
        fajar_done = True

elif alt_deg <= batas_fajar_isya and not cek_siang and not isya_done:
    # Turun melewati -18 -> ISYA
    timeline_events.append(f"[{jam_kejadian}] 🌌 ISYA (Syafaq Hilang) - Posisi Matahari:
{alt_deg:.2f}°")
    isya_done = True

```

```

# 4. Susun Laporan Final
report = f""{self.get_header(90)}
{ "[ KALKULATOR WAKTU SHALAT PENERBANGAN (AEROSPACE) ] ".center(90)}

```

Penerbangan jarak jauh (Long-Haul) melintasi garis bujur dengan cepat menyebabkan fenomena waktu relatif. Di ketinggian {altitude_ft:,.0f} ft, ufuk mar'i turun sebesar {abs(dip_deg):.2f}°.

=====

=====

```

[ DATA KEBERANGKATAN (Waktu Acuan Lokal) ]
Lokasi Awal : Lat {lat1:.4f}, Lon {lon1:.4f}
Take Off   : {dt_dep_lokal.strftime('%d %B %Y - %H:%M')} (Timezone {tz_awal})

```

[KONDISI DI UDARA]

Durasi : {durasi_jam} Jam

Altitude : {altitude_ft:,.0f} ft (Batas Horizon Pesawat: {batas_ufuk_pesawat:.2f}°)

=====

[GARIS WAKTU SHALAT SELAMA PENERBANGAN]

* Seluruh jam di bawah ini menggunakan Waktu Jam Tangan Anda saat berangkat (Waktu Lokal Asal).

* Tidak perlu memutar jam tangan selama di udara.

"""

```
        if not timeline_events:
            report += ">> Pesawat tidak melintasi transisi waktu shalat (Misal: Terbang siang hari penuh).\n"
```

```
        else:
```

```
            for ev in timeline_events:
```

```
                report += f"{ev}\n"
```

```
        report +=
```

```
        "\n=====
```

```
=====\\n"
```

```
        report += "[ LOG JALUR PENERBANGAN & ARAH KIBLAT DINAMIS ]\n"
```

```
        report += "Jam(Asal) | Koordinat Pesawat | Tinggi Mth. | Arah Kiblat (Dari Utara Pesawat)\n"
```

```
        report += "- " * 90 + "\\n"
```

```
        for log in log_rute:
```

```
            report += f"{log}\n"
```

```
        report += "\n* Catatan: Arah Kiblat (Qibla) terus bergeser seiring pesawat berpindah benua/samudera.\n"
```

```
        self.after(0, lambda: self.txt_flight_report.insert("1.0", report))
```

```
    except Exception as e:
```

```
        import traceback
```

```
        self.after(0, lambda: self.txt_flight_report.insert("1.0", f"TERJADI
```

```
KESALAHAN:\n{str(e)}\n\n{traceback.format_exc()}"))
```

```
    finally:
```

```
        self.after(0, lambda: self.txt_flight_report.configure(state="disabled"))
```

```
        self.after(0, lambda: self.btn_calc_flight.configure(state="normal", text="▶ HITUNG JADWAL SELAMA PENERBANGAN"))
```

```
def calculate_celestial_qiblah(self):
```

```
    """Mencari jam di mana bintang tertentu melintasi Azimut Kiblat lokal"""
```

```
    try:
```

```
        from skyfield.api import Star, wgs84
```



```

# 1. Ambil Parameter Lokasi & Tanggal
lat, lon = float(self.entry_qlat.get()), float(self.entry_qlon.get())
tz = float(self.entry_qtz.get())
y, m, d = int(self.entry_qyear.get()), int(self.entry_qmonth.get()), int(self.entry_qday.get())

# 2. Hitung Azimut Kiblat (Spherical Trigonometry)
phi_k, lam_k = math.radians(21.4225), math.radians(39.8262)
phi, lam = math.radians(lat), math.radians(lon)
y_q = math.sin(lam_k - lam)
x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
q_az = math.degrees(math.atan2(y_q, x_q)) % 360

# 3. Katalog Bintang (RA/Dec J2000)
star_catalog = {
    "Canopus (Suhail)": Star(ra_hours=(6, 23, 57.1), dec_degrees=(-52, 41, 45)),
    "Sirius (Asy-Syira)": Star(ra_hours=(6, 45, 8.9), dec_degrees=(-16, 42, 58)),
    "Rigel (Orion)": Star(ra_hours=(5, 14, 32.3), dec_degrees=(-8, 12, 6)),
    "Betelgeuse (Orion)": Star(ra_hours=(5, 55, 10.3), dec_degrees=(7, 24, 25)),
    "Acrux (Crux/Salib Sel.)": Star(ra_hours=(12, 26, 35.9), dec_degrees=(-63, 5, 57)),
    "Vega": Star(ra_hours=(18, 36, 56.3), dec_degrees=(38, 47, 1)),
    "Altair": Star(ra_hours=(19, 50, 47), dec_degrees=(8, 52, 6)),
    "Antares": Star(ra_hours=(16, 29, 24.4), dec_degrees=(-26, 25, 55)),
    "Spica": Star(ra_hours=(13, 25, 11.6), dec_degrees=(-11, 9, 41))
}

target_star = star_catalog[self.combo_q_star.get()]
self.auto_switch_ephemeris(y)
earth = self.eph['earth']
loc = wgs84.latlon(lat, lon)
observer = earth + loc

# 4. Iterasi Pencarian (Rentang 24 jam)
t0 = self.ts.utc(y, m, d, -int(tz))
tt_array = np.linspace(t0.tt, t0.tt + 1.0, 1440) # Sampel per menit
t_search = self.ts.tt_jd(tt_array)

# Ambil Azimut & Altitude Bintang
app = observer.at(t_search).observe(target_star).apparent()
alt_arr, az_arr, _ = app.altaz()

diffs = (az_arr.degrees - q_az + 180) % 360 - 180
results = []

for i in range(len(diffs)-1):
    if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
        if alt_arr.degrees[i] > 5: # Minimal 5 derajat di atas ufuk agar terlihat
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1])) + 1e-9
            t_res = self.ts.tt_jd(tt_array[i] + frac * (tt_array[i+1] - tt_array[i]))

```

```

dt_lokal = t_res.utc_datetime() + datetime.timedelta(hours=tz)
results.append((dt_lokal.strftime("%H:%M:%S"), alt_arr.degrees[i]))

# 5. Output Report
rep = f"{self.get_header(85)}\n"
rep += f"{'[ Celestial Qiblah: Verifikasi via Bintang ]'.center(85)}\n"
rep += f"* Objek Bintang : {self.combo_q_star.get()}\n"
rep += f"* Azimut Kiblat : {q_az:.2f}° (dari Utara)\n"
rep += f"* Lokasi      : Lat {lat}, Lon {lon}, TZ {tz}\n"
rep += "="*85 + "\n\n"

if not results:
    rep += f">> Sayang sekali, {self.combo_q_star.get()} tidak melintasi jalur azimut kiblat\n"
    rep += f"di lokasi Anda pada tanggal {d:02d}-{m:02d}-{y}.\n"
else:
    rep += f"{'Waktu (LT)':<15} | {'Ketinggian Bintang':<20} | {'Catatan'}\n"
    rep += "-"*85 + "\n"
    for w, alt in results:
        rep += f"{'w':<15} | {'alt':>17.2f}° | Berada tepat di arah Ka'bah\n"
    rep += "\n* Gunakan waktu di atas untuk melihat bintang tersebut.\n"
    rep += "Jika bintang berada tepat di depan Anda, maka itulah arah Kiblat Sejati.\n"

self.after(0, self.display_result, rep)

except Exception as e:
    self.after(0, self.display_error, str(e))

# =====
# MODUL TAMBAHAN: KURVA ALASKA (TERINTEGRASI MENU 5)
# =====
def setup_kurva_alaska_out_frame(self):
    """Menyiapkan ruang output (kanan) untuk Kurva Alaska"""
    self.frame_alaska_out = ctk.CTkFrame(self.main_frame, fg_color="#101010")

    self.fig_alaska = plt.Figure(figsize=(8, 6), dpi=100, facecolor="#101010")
    self.ax_alaska = self.fig_alaska.add_subplot(111)
    self.ax_alaska.set_facecolor("#1E1E1E")

    self.canvas_alaska = FigureCanvasTkAgg(self.fig_alaska, master=self.frame_alaska_out)
    self.canvas_alaska.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=10)

# Tooltip Hover
self.annot_alaska = self.ax_alaska.annotate(
    "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
    bbox=dict(boxstyle="round,pad=0.5", fc="#2A2A2A", ec="white", lw=1, alpha=0.9),
    color="white", fontsize=10, zorder=10
)
self.annot_alaska.set_visible(False)

```

```

self.canvas_alaska.mpl_connect("motion_notify_event", self.on_hover_alaska)

def proses_kurva_alaska(self):
    self.lbl_status.configure(text="Menghitung Kurva Alaska...", text_color="#00E5FF")
    self.btn_kurva_alaska.configure(state="disabled", text="MENGHITUNG...")
    threading.Thread(target=self._kalkulasi_kurva_alaska, daemon=True).start()

def _kalkulasi_kurva_alaska(self):
    try:
        from skyfield import almanac
        import math
        import datetime

        y = int(self.entry_gha_year.get())
        m = int(self.entry_gha_month.get())
        lat = float(self.entry_gha_lat.get())
        lon = float(self.entry_gha_lon.get())
        tz_offset = float(self.entry_gha_tz.get())

        self.auto_switch_ephemeris(y)
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
        lokasi = wgs84.latlon(lat, lon)
        observer = earth + lokasi

        f_sunset = almanac.sunrise_sunset(self.eph, lokasi)
        _, num_days = calendar.monthrange(y, m)

        days, altitudes, elongations, maghrib_times = [], [], [], []

        for d in range(1, num_days + 1):
            t0 = self.ts.utc(y, m, d, -int(tz_offset))
            t1 = self.ts.utc(y, m, d + 1, -int(tz_offset))

            t_events, y_events = almanac.find_discrete(t0, t1, f_sunset)

            # Filter hanya saat Matahari Terbenam (y_events == 0)
            sunsets = [t for t, is_sr in zip(np.atleast_1d(t_events), np.atleast_1d(y_events)) if not is_sr]

            if sunsets:
                t_maghrib = sunsets[0]
                t_maghrib_local = t_maghrib.utc_datetime() + datetime.timedelta(hours=tz_offset)
                time_str = t_maghrib_local.strftime('%H:%M:%S')

                geo_sun = earth.at(t_maghrib).observe(sun).apparent()
                geo_moon = earth.at(t_maghrib).observe(moon).apparent()
                elongation = geo_sun.separation_from(geo_moon).degrees

                topo_moon = observer.at(t_maghrib).observe(moon).apparent()

```

```

alt_topo_m, _, _ = topo_moon.altaz(temperature_C=25, pressure_mbar=1010)

days.append(d)
altitudes.append(alt_topo_m.degrees)
elongations.append(elongation)
maghrib_times.append(time_str)
else:
    days.append(d)
    altitudes.append(None)
    elongations.append(None)
    maghrib_times.append(None)

self.after(0, self._render_kurva_alaska, days, altitudes, elongations, maghrib_times, lat, lon, y, m)

except Exception as e:
    import traceback
    self.after(0, self.display_error, f"Gagal menghitung kurva:\n{str(e)}\n{traceback.format_exc()}")
    self.after(0, lambda: self.btn_kurva_alaska.configure(state="normal", text="GENERATE KURVA
ALASKA"))

def _render_kurva_alaska(self, days, altitudes, elongations, maghrib_times, lat, lon, year, month):
    self.ax_alaska.clear()
    self.ax_alaska.set_facecolor("#1E1E1E")

    self.valid_days_alaska.clear()
    self.valid_alt_alaska.clear()
    self.valid_elong_alaska.clear()
    self.hover_data_alaska.clear()

    for d, a, e, t in zip(days, altitudes, elongations, maghrib_times):
        if a is not None:
            self.valid_days_alaska.append(d)
            self.valid_alt_alaska.append(a)
            self.valid_elong_alaska.append(e)
            self.hover_data_alaska[d] = {'time': t}

    # Plot Data
    self.ax_alaska.plot(self.valid_days_alaska, self.valid_alt_alaska, color="#00E5FF", label='Tinggi Hilal
(Topo)', linewidth=2.5, marker='o', markersize=5)
    self.ax_alaska.plot(self.valid_days_alaska, self.valid_elong_alaska, color="#FFD700", label='Elongasi
(Geo)', linewidth=2.5, marker='s', markersize=5)

    self.ax_alaska.axhline(5, color='#4CAF50', linestyle='--', linewidth=1.5, label='Batas Alt 5°')
    self.ax_alaska.axhline(8, color='#FF5722', linestyle='--', linewidth=1.5, label='Batas Elong 8°')

    for d, alt, elong in zip(self.valid_days_alaska, self.valid_alt_alaska, self.valid_elong_alaska):
        if -5 <= alt <= 20:

```

```

        self.ax_alaska.annotate(f"{alt:.1f}°", (d, alt), textcoords="offset points", xytext=(0,10),
ha='center', fontsize=9, color="#00E5FF")
        self.ax_alaska.annotate(f"{elong:.1f}°", (d, elong), textcoords="offset points", xytext=(0,-15),
ha='center', fontsize=9, color="#FFD700")

        self.annot_alaska = self.ax_alaska.annotate("", xy=(0,0), xytext=(15, 15), textcoords="offset points",
        bbox=dict(boxstyle="round,pad=0.5", fc="#2A2A2A", ec="white", lw=1, alpha=0.9),
        color="white", fontsize=10, zorder=10)
        self.annot_alaska.set_visible(False)

        self.ax_alaska.set_title(f"Posisi Bulan Waktu Maghrib (Lat: {lat}, Lon: {lon}) - {year}/{month}",
color="white", fontsize=14, pad=15)
        self.ax_alaska.set_xlabel("Tanggal", color="lightgray", fontsize=12)
        self.ax_alaska.set_ylabel("Derajat (°)", color="lightgray", fontsize=12)
        self.ax_alaska.set_xticks(self.valid_days_alaska)

        if self.valid_alt_alaska and self.valid_elong_alaska:
            min_y = max(min(self.valid_alt_alaska) - 5, -15)
            max_y = min(max(self.valid_elong_alaska) + 5, 40)
            self.ax_alaska.set_ylim(min_y, max_y)

        self.ax_alaska.grid(True, color="#444444", linestyle=':', linewidth=0.8)
        self.ax_alaska.legend(loc='upper left', facecolor="#222222", edgecolor="none", labelcolor="white")

        self.fig_alaska.tight_layout()
        self.canvas_alaska.draw()

        # ---> PERBAIKAN UI: Hanya sembunyikan textbox tanpa menghapus Header <---
        self.textbox.grid_remove()
        self.frame_alaska_out.grid(row=1, column=0, sticky="nsew")

        self.lbl_status.configure(text="Kurva Alaska Berhasil Dirender", text_color="#00E676")
        self.btn_kurva_alaska.configure(state="normal", text="GENERATE KURVA ALASKA")

    def on_hover_alaska(self, event):
        """Sistem Interpolasi Hover Interaktif Alaska"""
        if event.inaxes == self.ax_alaska:
            x_val = event.xdata
            y_val = event.ydata

            if x_val is None or y_val is None: return

            if not self.valid_days_alaska or x_val < min(self.valid_days_alaska) or x_val >
max(self.valid_days_alaska):
                if self.annot_alaska.get_visible():
                    self.annot_alaska.set_visible(False)
                    self.canvas_alaska.draw_idle()
                return

```

```

import numpy as np
calc_alt = np.interp(x_val, self.valid_days_alaska, self.valid_alt_alaska)
calc_elong = np.interp(x_val, self.valid_days_alaska, self.valid_elong_alaska)

closest_day = int(round(x_val))
time_str = self.hover_data_alaska.get(closest_day, {}).get('time', '--:--:--')

text = (f"Posisi X: {x_val:.2f}\n"
        f"Ref. Maghrib: Tgl {closest_day} ({time_str})\n"
        f"Est. Altitude: {calc_alt:.2f}°\n"
        f"Est. Elongasi: {calc_elong:.2f}°")

self.annot_alaska.xy = (x_val, y_val)
self.annot_alaska.set_text(text)
self.annot_alaska.set_visible(True)
self.canvas_alaska.draw_idle()
else:
    if self.annot_alaska.get_visible():
        self.annot_alaska.set_visible(False)
        self.canvas_alaska.draw_idle()

# =====
# MODUL 9 LANJUTAN: TREN BULANAN HIJRIAH (RAJAB - ZULHIJAH)
# =====
def trigger_trend(self, trend_type):
    """Memantik render grafik tren ke background thread"""
    self.lbl_status.configure(text="Mengekstrak Data Tren Bulanan...", text_color="#00E5FF")
    import threading
    threading.Thread(target=self.calculate_trend_hijri, args=(trend_type,), daemon=True).start()

def calculate_trend_hijri(self, trend_type):
    try:
        import math
        import numpy as np
        import datetime
        import pytz
        from skyfield import almanac
        from skyfield.api import wgs84

        h_year = int(self.entry_chart_hy.get())
        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        tz = float(self.entry_vtz.get())
        elev = float(self.entry_velev.get())

        if h_year not in HIJRI_DB:
            raise ValueError(f"Tahun Hijriah {h_year} tidak tersedia di database.")

```

```

# Mengambil data bulan 7 (Rajab) sampai 12 (Zulhijah) dari HIJRI_DB
# Index -> 6: Rajab, 7: Syakban, 8: Ramadan, 9: Syawal, 10: Zulkaidah, 11: Zulhijah
bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}

# 1. Cari Tanggal Mulai (1 Rajab)
r_data = HIJRI_DB[h_year][6]
p_start = r_data[2].split('-')
start_date = datetime.date(int(p_start[2]), bulan_map[p_start[1]], int(p_start[0]))

# 2. Cari Tanggal Akhir (Akhir Zulhijah)
z_data = HIJRI_DB[h_year][11]
p_end = z_data[2].split('-')
z_start = datetime.date(int(p_end[2]), bulan_map[p_end[1]], int(p_end[0]))
end_date = z_start + datetime.timedelta(days=z_data[3]-1)

num_days = (end_date - start_date).days + 1

# Load Ephemeris di tengah-tengah rentang waktu
mid_date = start_date + datetime.timedelta(days=num_days//2)
self.auto_switch_ephemeris(mid_date.year)

earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
loc = wgs84.latlon(lat, lon, elevation_m=elev)

dates_greg = []
alt_topo, alt_geo = [], []
elong_topo, elong_geo = [], []
illum = []

# Kumpulkan penanda tanggal awal bulan Hijriah
hijri_marks = []
for m_idx in range(6, 12):
    m_data = HIJRI_DB[h_year][m_idx]
    p_m = m_data[2].split('-')
    d_m = datetime.date(int(p_m[2]), bulan_map[p_m[1]], int(p_m[0]))
    hijri_marks.append({'date': d_m, 'name': m_data[0]})

# Looping per hari (Komputasi pada waktu Maghrib/Sunset Lokal)
for i in range(num_days):
    curr_date = start_date + datetime.timedelta(days=i)
    dates_greg.append(curr_date)

    t0 = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 0 - int(tz))
    t1 = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 24 - int(tz))

    f_rs = almanac.sunrise_sunset(self.eph, loc)

```

```

t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
t_sunset = None
if t_rs is not None:
    for tv, yv in zip(np.atleast_1d(t_rs.tt), np.atleast_1d(y_rs)):
        if yv == 0:
            t_sunset = self.ts.tt(jd=tv)
            break
if t_sunset is None:
    t_sunset = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 18 - int(tz))

# --- KALKULASI TOPOSENTRIK ---
obs_topo = (earth + loc).at(t_sunset)
app_moon_topo = obs_topo.observe(moon).apparent()
app_sun_topo = obs_topo.observe(sun).apparent()

alt_topo.append(app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)
elong_topo.append(app_sun_topo.separation_from(app_moon_topo).degrees)

# --- KALKULASI GEOSENTRIK ---
obs_geo = earth.at(t_sunset)
app_moon_geo = obs_geo.observe(moon).apparent()
app_sun_geo = obs_geo.observe(sun).apparent()

elong_geo.append(app_sun_geo.separation_from(app_moon_geo).degrees)

ra_m, dec_m, _ = app_moon_geo.radec(epoch=t_sunset)
lst_deg = (t_sunset.gast * 15.0) + lon
ha_rad = math.radians(lst_deg - (ra_m.hours * 15.0))
d_rad = dec_m.radians
lat_rad = math.radians(lat)

sin_alt_geo = math.sin(d_rad)*math.sin(lat_rad) +
math.cos(d_rad)*math.cos(lat_rad)*math.cos(ha_rad)
alt_geo.append(math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo)))))

# --- ILUMINASI ---
il = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
illum.append((il.item() if hasattr(il, 'item') else il) * 100.0)

data_bundle = {
    'dates': dates_greg,
    'alt_topo': alt_topo, 'alt_geo': alt_geo,
    'elong_topo': elong_topo, 'elong_geo': elong_geo,
    'illum': illum, 'hijri_marks': hijri_marks,
    'type': trend_type, 'h_year': h_year
}

self.after(0, self.plot_trend_dispatcher, data_bundle)

```



```

except Exception as e:
    import traceback
    self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")

def plot_trend_dispatcher(self, data):
    """Merender canvas grafik jangka panjang (Bulan berbulan-bulan)"""
    try:
        for widget in self.frame_chart_out.wininfo_children():
            widget.destroy()

        from matplotlib.figure import Figure
        from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
        import matplotlib.dates as mdates
        import numpy as np

        fig = Figure(figsize=(9, 5.5), dpi=100)
        fig.patch.set_facecolor('#181818')
        ax = fig.add_subplot(111)
        ax.set_facecolor('#101010')

        trend_type = data['type']
        dates = data['dates']
        h_year = data['h_year']

        x_dates = mdates.date2num(dates)

        self.trend_type = trend_type
        self.trend_data = data

        if trend_type == "altitude":
            ax.plot(x_dates, data['alt_topo'], color='#80DEEA', label='Altitude Toposentrik', linewidth=1.5)
            ax.plot(x_dates, data['alt_geo'], color='#00E676', label='Altitude Geosentrik', linewidth=1.5,
linestyle='--')
            ax.axhline(5, color='#00E676', linestyle=':', linewidth=1, label="Batas Tinggi KHGT (5°)")
            ax.axhline(3, color='#80DEEA', linestyle=':', linewidth=1, label="Batas Tinggi MABIMS (3°)")
            ax.axhline(0, color='#FF5252', linestyle='-', linewidth=1)
            ax.set_title(f"Tren Ketinggian Hilal Saat Maghrib (Rajab - Zulhijah {h_year} H)", color='white',
fontsize=14, fontweight='bold', pad=15)
            ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)

        elif trend_type == "elongation":
            ax.plot(x_dates, data['elong_topo'], color='#FFB74D', label='Elongasi Toposentrik',
linewidth=1.5)
            ax.plot(x_dates, data['elong_geo'], color='#FFD54F', label='Elongasi Geosentrik', linewidth=1.5,
linestyle='--')
            ax.axhline(8, color='#FFD54F', linestyle=':', linewidth=1, label="Batas Elongasi KHGT (8°)")

```

```

        ax.axhline(6.4, color='#FFB74D', linestyle=':', linewidth=1, label="Batas Elongasi MABIMS
(6.4°)")
        ax.set_title(f"Tren Elongasi Bulan-Matahari Saat Maghrib (Rajab - Zulhijah {h_year} H)",
color='white', fontsize=14, fontweight='bold', pad=15)
        ax.set_ylabel("Sudut Elongasi (°)", color='ffffff', fontsize=12)

    elif trend_type == "illumination":
        ax.plot(x_dates, data['illum'], color='#E040FB', label='Fraksi Iluminasi Bulan', linewidth=2)
        ax.set_title(f"Siklus Iluminasi / Fase Bulan Saat Maghrib (Rajab - Zulhijah {h_year} H)",
color='white', fontsize=14, fontweight='bold', pad=15)
        ax.set_ylabel("Iluminasi (%)", color='ffffff', fontsize=12)

    # ---> TAMBAHAN LOGIKA PLOT KOMBINASI <---
    elif trend_type == "kombinasi":
        ax.plot(x_dates, data['alt_topo'], color='#80DEEA', label='Alt Topo (°)', linewidth=1.5)
        ax.plot(x_dates, data['alt_geo'], color='#00E676', label='Alt Geo (°)', linewidth=1.5, linestyle='--
')
        ax.plot(x_dates, data['elong_topo'], color='#FFB74D', label='Eln Topo (°)', linewidth=1.5)
        ax.plot(x_dates, data['elong_geo'], color='#FFD54F', label='Eln Geo (°)', linewidth=1.5,
linestyle='-.')
        ax.plot(x_dates, data['illum'], color='#E040FB', label='Iluminasi (%)', linewidth=2, alpha=0.8)

        ax.axhline(0, color='#FF5252', linestyle='-', linewidth=1)
        ax.set_title(f"Tren Kombinasi (Alt, Eln, Illum) Saat Maghrib\nRajab - Zulhijah {h_year} H",
color='white', fontsize=14, fontweight='bold', pad=15)
        ax.set_ylabel("Nilai (Derajat / Persen)", color='ffffff', fontsize=12)
        # -----

    # Garis vertikal pembatas antar bulan Hijriah
    colors_h = ['#EF5350', '#AB47BC', '#29B6F6', '#66BB6A', '#FFA726', '#FF7043']
    for i, mark in enumerate(data['hijri_marks']):
        mx = mdates.date2num(mark['date'])
        ax.axvline(mx, color=colors_h[i], linestyle='-', linewidth=1.5, alpha=0.8)
        ax.text(mx + 1, ax.get_ylim()[1] * 0.95, f"1 {mark['name']}", color=colors_h[i], rotation=90,
verticalalignment='top', fontsize=9, fontweight='bold')

    # Atur format sumbu X menjadi tanggal
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %b'))
    ax.xaxis.set_major_locator(mdates.DayLocator(interval=15))

    ax.tick_params(colors='ffffff', labels=10)
    ax.grid(True, color='ffffff', linestyle=':', alpha=0.2)
    ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='lower
right')
    fig.autofmt_xdate()
    fig.tight_layout()

    self.trend_ax = ax

```

```

self.trend_canvas = FigureCanvasTkAgg(fig, master=self.frame_chart_out)

# Sistem Tooltip Hover Matplotlib
self.annot_trend = ax.annotate(
    "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
    bbox=dict(boxstyle="round,pad=0.5", fc="#1E1E1E", ec="#FFD54F", lw=1.5, alpha=0.9),
    color="white", fontsize=10, zorder=10
)
self.annot_trend.set_visible(False)
self.trend_canvas.mpl_connect("motion_notify_event", self.on_hover_trend)

self.trend_canvas.draw()
self.trend_canvas.get_tk_widget().pack(fill="both", expand=True)

self.lbl_status.configure(text="Grafik Tren Berhasil Dimuat", text_color="#00E676")
self.btn_hitung.configure(state="normal")

except Exception as e:
    import traceback
    self._force_show_error(f"Gagal Merender Grafik Tren:\n{str(e)}\n{traceback.format_exc()}")

def on_hover_trend(self, event):
    if event.inaxes == getattr(self, 'trend_ax', None):
        x_val = event.xdata
        y_val = event.ydata
        if x_val is None or y_val is None: return

        data = getattr(self, 'trend_data', None)
        if not data: return

        import matplotlib.dates as mdates
        import numpy as np

        x_dates = mdates.date2num(data['dates'])
        if x_val < x_dates[0] or x_val > x_dates[-1]:
            self.annot_trend.set_visible(False)
            self.trend_canvas.draw_idle()
            return

        # Cari hari / index terdekat
        idx = (np.abs(x_dates - x_val)).argmin()
        closest_date = data['dates'][idx]
        date_str = closest_date.strftime("%d %b %Y")

        if self.trend_type == "altitude":
            v_topo = data['alt_topo'][idx]
            v_geo = data['alt_geo'][idx]

```

```

        text = f"📅 {date_str} (Maghrib)\n 🌙 Alt Topo: {v_topo:+.2f}°\n 🌍 Alt Geo : {v_geo:+.2f}°"
    elif self.trend_type == "elongation":
        v_topo = data['elong_topo'][idx]
        v_geo = data['elong_geo'][idx]
        text = f"📅 {date_str} (Maghrib)\n 📐 Eln Topo: {v_topo:.2f}°\n 🌍 Eln Geo : {v_geo:.2f}°"
    elif self.trend_type == "illumination":
        v_ill = data['illum'][idx]
        text = f"📅 {date_str} (Maghrib)\n 💡 Iluminasi: {v_ill:.2f}%"
    # ---> TAMBAHAN HOVER UNTUK MODE KOMBINASI <---
    elif self.trend_type == "kombinasi":
        v_alt_t = data['alt_topo'][idx]
        v_alt_g = data['alt_geo'][idx]
        v_eln_t = data['elong_topo'][idx]
        v_eln_g = data['elong_geo'][idx]
        v_ill = data['illum'][idx]
        text = (f"📅 {date_str} (Maghrib)\n"
                f"🌙 Alt Topo: {v_alt_t:+.2f}° | Geo: {v_alt_g:+.2f}°\n"
                f"📐 Eln Topo: {v_eln_t:.2f}° | Geo: {v_eln_g:.2f}°\n"
                f"💡 Iluminasi: {v_ill:.2f}%")
    # -----

    self.annot_trend.xy = (x_dates[idx], y_val)
    self.annot_trend.set_text(text)
    self.annot_trend.set_visible(True)
    self.trend_canvas.draw_idle()
else:
    if getattr(self, 'annot_trend', None) and self.annot_trend.get_visible():
        self.annot_trend.set_visible(False)
        self.trend_canvas.draw_idle()
# =====
# SISTEM MULTI-BAHASA (JSON TRANSLATOR)
# =====
def _cache_base_dir(self):
    """Folder aplikasi, kompatibel untuk mode .py dan .exe."""
    try:
        if getattr(sys, 'frozen', False):
            return os.path.dirname(sys.executable)
        return os.path.dirname(os.path.abspath(__file__))
    except Exception:
        return os.getcwd()

def _find_cache_file(self, filename):
    """Cari file cache bahasa; menerima nama normal dan salinan seperti name(2).json."""
    base_path = self._cache_base_dir()
    direct = os.path.join(base_path, filename)
    if os.path.exists(direct):
        return direct

```

```

stem, ext = os.path.splitext(filename)
try:
    for name in os.listdir(base_path):
        if name == filename:
            return os.path.join(base_path, name)
        if name.startswith(stem + "(") and name.endswith(ext):
            return os.path.join(base_path, name)
except Exception:
    pass
return direct

def _merge_translation_mapping(self, data):
    """Gabungkan cache bahasa ke self.translations dalam format {teks_id: {en, ar}}."""
    if not isinstance(data, dict):
        return

    # Dukungan format release: {"en": {raw: trans}, "ar": {raw: trans}}
    if isinstance(data.get("en"), dict) or isinstance(data.get("ar"), dict):
        looks_like_release = any(
            isinstance(v, dict) and v and all(isinstance(x, str) for x in list(v.values())[:5])
            for k, v in data.items() if k in ("en", "ar")
        )
        if looks_like_release and all(k in ("en", "ar") for k in data.keys()):
            for lang in ("en", "ar"):
                for key, trans in (data.get(lang) or {}).items():
                    if isinstance(key, str) and isinstance(trans, str):
                        self.translations.setdefault(key, {})[lang] = trans
            return

    # Format utama UI: {teks_sumber: {"id": ..., "en": ..., "ar": ...}}
    # Kunci "id" opsional dipakai untuk membersihkan teks campuran Indonesia/Inggris
    # tanpa mengubah kunci internal menu maupun logika modul.
    for key, langs in data.items():
        if not isinstance(key, str) or not isinstance(langs, dict):
            continue
        entry = self.translations.setdefault(key, {})
        for lang in ("id", "en", "ar"):
            val = langs.get(lang)
            if isinstance(val, str) and val:
                entry[lang] = val

def _build_language_indexes(self):
    """Bangun indeks O(1), indeks balik, dan reset regex translasi."""
    self._translation_index = {"id": {}, "en": {}, "ar": {}}
    self._reverse_translation_map = {}

    for source_text, langs in (getattr(self, "translations", {}) or {}).items():
        if not isinstance(source_text, str):

```

```

        continue
    if not isinstance(langs, dict):
        langs = {}

    canonical_id = langs.get("id")
    if not isinstance(canonical_id, str) or not canonical_id:
        canonical_id = source_text

    variants = [source_text, canonical_id]
    for lang in ("en", "ar"):
        value = langs.get(lang)
        if isinstance(value, str) and value:
            variants.append(value)

    for value in variants:
        if isinstance(value, str) and value:
            self._reverse_translation_map[self._norm_lang_text(value)] = canonical_id

    for lang in ("id", "en", "ar"):
        if lang == "id":
            target = canonical_id
        else:
            target = langs.get(lang)
            if not isinstance(target, str) or not target:
                target = canonical_id
        self._translation_index[lang][source_text] = target
        self._translation_index[lang][canonical_id] = target

    self._translation_patterns = {}
    self._translation_sorted_keys_cache = None
    self._release_sorted_keys_cache = None

def load_translations(self):
    """Memuat cache UI eksternal dan cache output agar perpindahan ID/EN/AR cepat."""
    self.translations = {}
    self._translation_index = {"id": {}, "en": {}, "ar": {}}
    self._reverse_translation_map = {}
    self._translation_patterns = {}
    self._translation_sorted_keys_cache = None
    self._builtin_translations_installed = False
    self._lang_output_cache = {}
    self._lang_output_cache_dirty = False

    # 1) Cache UI utama: paling penting untuk tombol, label, dan frasa laporan.
    try:
        lang_path = self._find_cache_file("lang_cache_ui.json")
        if os.path.exists(lang_path):
            with open(lang_path, "r", encoding="utf-8") as f:

```

```

        self._merge_translation_mapping(json.load(f))
except Exception as e:
    print(f"Peringatan: Gagal memuat lang_cache_ui.json -> {e}")

# 2) Cache output penuh: hash sha256(f"{lang}\\0{raw}") -> hasil terjemahan.
try:
    output_path = self.get_output_lang_cache_path()
    if os.path.exists(output_path):
        with open(output_path, "r", encoding="utf-8") as f:
            data = json.load(f)
            if isinstance(data, dict):
                self._lang_output_cache = {str(k): str(v) for k, v in data.items() if isinstance(v, str)}
except Exception as e:
    print(f"Peringatan: Gagal memuat lang_output_cache.json -> {e}")

self.install_builtin_translations()

def install_builtin_translations(self):
    """Fallback kecil saja; kamus besar dipindah ke lang_cache_ui.json."""
    if getattr(self, "_builtin_translations_installed", False):
        return
    if not hasattr(self, "translations") or self.translations is None:
        self.translations = {}

    fallback = {
        "id": {"en": "en", "ar": "ar"},
        "en": {"en": "en", "ar": "en"},
        "ar": {"en": "ar", "ar": "ar"},
        "KHGT ENGINE": {"en": "KHGT ENGINE", "ar": "محرك KHGT"},
        "Peta Langit Dunia": {"en": "World Sky Map", "ar": "خريطة السماء العالمية"},
        "Membuka Peta Langit Dunia melalui browser default...": {
            "en": "Opening World Sky Map in the default browser...",
            "ar": "...جارٍ فتح خريطة السماء العالمية في المتصفح الافتراضي",
        },
        "▶ PROSES DATA": {"en": "▶ PROCESS DATA", "ar": "▶ معالجة البيانات"},
        "Kalkulasi Selesai": {"en": "Calculation Complete", "ar": "اكتمل الحساب"},
        "Error Kalkulasi": {"en": "Calculation Error", "ar": "خطأ في الحساب"},
        "Sistem Siap": {"en": "System Ready", "ar": "النظام جاهز"},
        "Mohon tunggu": {"en": "Please wait", "ar": "يرجى الانتظار"},
        "Memproses data astrometri": {"en": "Processing astrometric data", "ar": "جارٍ معالجة البيانات الفلكية"},
    }

    # Kamus inti mandiri untuk menu, label utama, judul, dan pesan output.
    # Cache JSON eksternal tetap didukung dan tidak dihapus.
    #
    # Panduan pada halaman Release dirender langsung ke Textbox (bukan widget
    # CTKLabel/CTKButton), sehingga frasa-frasa ini harus tersedia di kamus

```

agar dapat diterjemahkan ketika bahasa EN/AR dipilih. URL sengaja
dipertahankan apa adanya.

```
fallback.update({
  "DOWNLOAD BUKU KHGT TIMES": {
    "id": "DOWNLOAD BUKU KHGT TIMES",
    "en": "DOWNLOAD KHGT TIMES BOOK",
    "ar": "تحميل كتاب KHGT Times",
  },
  "PYTHON MASTERY (TUTORIAL)": {
    "id": "PYTHON MASTERY (TUTORIAL)",
    "en": "PYTHON MASTERY (TUTORIAL)",
    "ar": "دليل تعليمي Python إتقان",
  },
  "NAVIGASI MODUL": {
    "id": "NAVIGASI MODUL",
    "en": "MODULE NAVIGATION",
    "ar": "التنقل بين الوحدات",
  },
  "INPUT LOKASI": {
    "id": "INPUT LOKASI",
    "en": "LOCATION INPUT",
    "ar": "إدخال الموقع",
  },
  "EKSEKUSI DATA": {
    "id": "EKSEKUSI DATA",
    "en": "DATA EXECUTION",
    "ar": "تنفيذ البيانات",
  },
  "INTERAKSI GRAFIS": {
    "id": "INTERAKSI GRAFIS",
    "en": "GRAPHICAL INTERACTION",
    "ar": "التفاعل الرسومي",
  },
  "SISTEM EXPORT": {
    "id": "SISTEM EXPORT",
    "en": "EXPORT SYSTEM",
    "ar": "نظام التصدير",
  },
  "Silahkan download aplikasi KHGT Times melalui link: https://s.id/buku-khggtimes": {
    "id": "Silahkan download aplikasi KHGT Times melalui link: https://s.id/buku-khggtimes",
    "en": "Please download the KHGT Times application via: https://s.id/buku-khggtimes",
    "ar": "يرجى تنزيل تطبيق KHGT Times عبر الرابط: https://s.id/buku-khggtimes",
  },
  "Akses tutorial Python dasar melalui link: https://kasmui.cloud/python/": {
    "id": "Akses tutorial Python dasar melalui link: https://kasmui.cloud/python/",
    "en": "Access the basic Python tutorial via: https://kasmui.cloud/python/",
    "ar": "الأساسي Python للوصول إلى دليل: https://kasmui.cloud/python/",
  },
},
```


"Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input yang tidak relevan agar layar selalu rapi dan fokus.": {

"id": "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input yang tidak relevan agar layar selalu rapi dan fokus.",

"en": "Use the scrollable 'KHGT ENGINE' dropdown menu in the left sidebar to switch between 41 scientific calculation modules. Each module hides irrelevant input forms so the screen remains clean and focused.",

"ar": "المسندلة القابلة للتمرير في الشريط الجانبي الأيسر للتنقل بين 41 وحدة 'KHGT ENGINE' استخدم قائمة",
للحسابات العلمية. تعمل كل وحدة على إخفاء نماذج الإدخال غير ذات الصلة حتى تظل الشاشة مرتبة ومركزة",

},

"Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan aplikasi, atau gunakan tombol '  Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar sistem melacak titik GPS dan mendeteksi Timezone otomatis.": {

"id": "Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan aplikasi, atau gunakan tombol '  Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar sistem melacak titik GPS dan mendeteksi Timezone otomatis.",

"en": "Two options are available. You can select a location from the built-in Province/City dropdown, or use the '  Detect Automatic Location' button to call the IP Geolocation API so the system can track the GPS point and detect the timezone automatically.",

"ar": "يتوفر خياران. يمكنك اختيار الموقع من قائمة المحافظة/المدينة المنسندلة المدمجة في التطبيق، أو استخدام زر '  واكتشاف GPS حتى يتمكن النظام من تتبع نقطة IP اكتشاف الموقع تلقائيًا لاستدعاء واجهة تحديد الموقع الجغرافي عبر '  المنطقة الزمنية تلقائيًا",

},

"Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter lalu klik tombol biru besar '  PROSES DATA' di panel bawah. Tombol ini otomatis bersembunyi jika Anda sedang memasuki modul Live/Real-time.": {

"id": "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter lalu klik tombol biru besar '  PROSES DATA' di panel bawah. Tombol ini otomatis bersembunyi jika Anda sedang memasuki modul Live/Real-time.",

"en": "In most text- and graph-based modules, simply enter the parameters and click the large blue '  PROCESS DATA' button in the lower panel. This button automatically hides when you enter a Live/Real-time module.",

"ar": "في معظم الوحدات المعتمدة على النصوص والرسوم البيانية، ما عليك سوى إدخال المعلومات ثم النقر على الزر '  الأزرق الكبير '  معالجة البيانات ' في اللوحة السفلية. يختفي هذا الزر تلقائيًا عند الدخول إلى وحدة مباشرة في الوقت الفعلي",

},

"Khusus pada canvas 3D Matplotlib dan Simulasi 3D Spasial, biasakan diri menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta Roda Mouse (Scroll) untuk melakukan perbesaran (Zoom In/Out).": {

"id": "Khusus pada canvas 3D Matplotlib dan Simulasi 3D Spasial, biasakan diri menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta Roda Mouse (Scroll) untuk melakukan perbesaran (Zoom In/Out).",

"en": "For the Matplotlib 3D canvas and Spatial 3D Simulation, use the Left Mouse Button (hold and drag) to rotate the camera orientation/viewpoint, and the Mouse Wheel (Scroll) to zoom in or out.",

"ar": "ثلاثية الأبعاد ومحاكاة الفضاء ثلاثية الأبعاد، استخدم زر الفأرة الأيسر (اضغط واسحب) في Matplotlib في لوحة",
للتدوير اتجاه الكاميرا/زاوية العرض، وعجلة الفأرة (التمرير) للتكبير أو التصغير",

```

    },
    "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di bawah
tabel/kanvas): \n [📄 TXT] Menyimpan data tabel saat ini ke file Notepad.\n [🖼️ PNG] Menyimpan
kanvas visual / laporan ke gambar PNG.\n [📄 PDF] Mencetak dokumen laporan standar akademik
fpdf.\n [📊 CSV] Mengekstrak data komparasi 50 Tahun ke Excel.\n [🏠 RILIS] Tombol reset darurat
kembali ke Dokumentasi.": {
    "id": "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di bawah
tabel/kanvas): \n [📄 TXT] Menyimpan data tabel saat ini ke file Notepad.\n [🖼️ PNG] Menyimpan
kanvas visual / laporan ke gambar PNG.\n [📄 PDF] Mencetak dokumen laporan standar akademik
fpdf.\n [📊 CSV] Mengekstrak data komparasi 50 Tahun ke Excel.\n [🏠 RILIS] Tombol reset darurat
kembali ke Dokumentasi.",
    "en": "A floating Action Bar is available in the upper-right corner of the screen (or below a
table/canvas): \n [📄 TXT] Saves the current table data to a Notepad file.\n [🖼️ PNG] Saves the visual
canvas/report as a PNG image.\n [📄 PDF] Prints a standard academic fpdf report.\n [📊 CSV]
Extracts 50-year comparison data to Excel.\n [🏠 RELEASE] Emergency reset button that returns to
Documentation.",
    "ar": "في الزاوية العلوية اليمنى من الشاشة (أو أسفل الجدول/اللوحة) (Action Bar) توجد لوحة إجراءات عائمة": \n
[📄 TXT] حفظ البيانات الجدول الحالية في ملف Notepad.\n [🖼️ PNG] حفظ اللوحة المرئية/التقرير كصورة PNG.\n
[📄 PDF] طباعة تقرير أكاديمي قياسي بصيغة fpdf.\n [📊 CSV] استخراج بيانات المقارنة لمدة 50 سنة إلى Excel.\n [🏠
الإصدار [زر إعادة ضبط طارئ للعودة إلى التوثيق",
    },
    })
    fallback.update({'1) Kalender Hijriah Berjalan': {'id': '1) Kalender Hijriah Berjalan', 'en': '1) Current
Hijri Calendar', 'ar': '1) التقويم الهجري الجاري'},
'2) Kalender Masehi Berjalan': {'id': '2) Kalender Masehi Berjalan', 'en': '2) Current Gregorian Calendar',
'ar': '2) التقويم الميلادي الجاري'},
'3) Konversi Kalender': {'id': '3) Konversi Kalender', 'en': '3) Calendar Conversion', 'ar': '3) تحويل التقويم'},
'4) Visibility Hilal (KHGT)': {'id': '4) Visibilitas Hilal (KHGT)', 'en': '4) Crescent Visibility (KHGT)', 'ar': '4) رؤية
الهلال (KHGT)'},
'5) Analisis Hilal Global': {'id': '5) Analisis Hilal Global', 'en': '5) Global Crescent Analysis', 'ar': '5) تحليل
الهلال العالمي'},
'6) Status Kriteria Batas': {'id': '6) Status Kriteria Batas', 'en': '6) Threshold Criteria Status', 'ar': '6) حالة
معايير الحدود'},
'7) Kota Pertama KHGT (Mainland)': {'id': '7) Kota Pertama KHGT (Daratan Utama)',
'en': '7) First KHGT City (Mainland)',
'ar': '7) أول مدينة KHGT (اليابسة الرئيسية)'},
'8) Crescent Visibility Map': {'id': '8) Peta Visibilitas Hilal', 'en': '8) Crescent Visibility Map', 'ar': '8) خريطة
رؤية الهلال'},
'9) Altitude Chart Analyser': {'id': '9) Penganalisis Grafik Ketinggian', 'en': '9) Altitude Chart Analyzer',
'ar': '9) محلل مخطط الارتفاع'},
'10) Komparasi 50 Tahun (Ramadhan)': {'id': '10) Komparasi 50 Tahun (Ramadan)', 'en': '10) 50-Year
Comparison (Ramadan)', 'ar': '10) مقارنة 50 سنة (رمضان)'},
'11) Tabel Tinggi Hilal 50 Thn (Ramadhan)': {'id': '11) Tabel Tinggi Hilal 50 Tahun (Ramadan)',
'en': '11) 50-Year Crescent Altitude Table (Ramadan)',
'ar': '11) جدول ارتفاع الهلال 50 سنة (رمضان)'},
'12) Tabel Elongasi Hilal 50 Thn (Ramadhan)': {'id': '12) Tabel Elongasi Hilal 50 Tahun (Ramadan)',

```

'en': '12) 50-Year Crescent Elongation Table (Ramadan)',
 'ar': '12) جدول استطالة الهلال لـ 50 سنة (رمضان)',
 '13) Komparasi 50 Tahun (Syawal)': {'id': '13) Komparasi 50 Tahun (Syawal)', 'en': '13) 50-Year Comparison (Shawwal)', 'ar': '13) مقارنة 50 سنة (شوال)',
 '14) Tabel Tinggi Hilal 50 Thn (Syawal)': {'id': '14) Tabel Tinggi Hilal 50 Tahun (Syawal)', 'en': '14) 50-Year Crescent Altitude Table (Shawwal)', 'ar': '14) جدول ارتفاع الهلال لـ 50 سنة (شوال)',
 '15) Tabel Elongasi Hilal 50 Thn (Syawal)': {'id': '15) Tabel Elongasi Hilal 50 Tahun (Syawal)', 'en': '15) 50-Year Crescent Elongation Table (Shawwal)', 'ar': '15) جدول استطالة الهلال لـ 50 سنة (شوال)',
 '16) Komparasi 50 Tahun (Zulhijah)': {'id': '16) Komparasi 50 Tahun (Zulhijah)', 'en': '16) 50-Year Comparison (Dhu al-Hijjah)', 'ar': '16) مقارنة 50 سنة (ذو الحجة)',
 '17) Tabel Tinggi Hilal 50 Thn (Zulhijah)': {'id': '17) Tabel Tinggi Hilal 50 Tahun (Zulhijah)', 'en': '17) 50-Year Crescent Altitude Table (Dhu al-Hijjah)', 'ar': '17) جدول ارتفاع الهلال لـ 50 سنة (ذو الحجة)',
 '18) Tabel Elongasi Hilal 50 Thn (Zulhijah)': {'id': '18) Tabel Elongasi Hilal 50 Tahun (Zulhijah)', 'en': '18) 50-Year Crescent Elongation Table (Dhu al-Hijjah)', 'ar': '18) جدول استطالة الهلال لـ 50 سنة (ذو الحجة)',
 '19) Komparasi 50 Tahun (12 Bulan)': {'id': '19) Komparasi 50 Tahun (12 Bulan)', 'en': '19) 50-Year Comparison (12 Months)', 'ar': '19) مقارنة 50 سنة (12 شهرًا)',
 '20) Fase Bulan (Moonphase)': {'id': '20) Fase Bulan', 'en': '20) Moon Phases', 'ar': '20) أطوار القمر'},
 '21) Moon Times': {'id': '21) Waktu Bulan', 'en': '21) Moon Times', 'ar': '21) أوقات القمر'},
 '22) Sun Times': {'id': '22) Waktu Matahari', 'en': '22) Sun Times', 'ar': '22) أوقات الشمس'},
 '23) Sun Moon Ephemeris': {'id': '23) Ephemeris Matahari dan Bulan', 'en': '23) Sun and Moon Ephemeris', 'ar': '23) تقويم الشمس والقمر الفلكي'},
 '24) Sistem Sun Moon Earth': {'id': '24) Sistem Matahari–Bulan–Bumi', 'en': '24) Sun–Moon–Earth System', 'ar': '24) نظام الشمس والقمر والأرض'},
 '25) Prayer Times': {'id': '25) Waktu Salat', 'en': '25) Prayer Times', 'ar': '25) مواقيت الصلاة'},
 '26) Qiblah Direction & Times': {'id': '26) Arah dan Waktu Kiblat', 'en': '26) Qibla Direction and Times', 'ar': '26) اتجاه القبلة وأوقاتها'},
 '27) Qibla Time (Rashdul Lokal)': {'id': '27) Waktu Kiblat (Rashdul Kiblat Lokal)', 'en': '27) Qibla Time (Local Rashdul Qibla)', 'ar': '27) وقت القبلة (الرصد المحلي)'},
 '28) Kalkulator Mizwala (Bayangan)': {'id': '28) Kalkulator Mizwala (Bayangan)', 'en': '28) Mizwala Calculator (Shadow)', 'ar': '28) حاسبة المزالة (الظل)'},
 '29) Evaluasi Fajar Shadiq & SQM': {'id': '29) Evaluasi Fajar Sadik dan SQM', 'en': '29) True Dawn and SQM Evaluation', 'ar': '29) تقييم الفجر الصادق و SQM'},
 '30) Analisis Gerhana': {'id': '30) Analisis Gerhana', 'en': '30) Eclipse Analysis', 'ar': '30) تحليل الكسوف والخرسوف'},
 '31) Equinox & Solstice': {'id': '31) Ekuinoks dan Solstis', 'en': '31) Equinox and Solstice', 'ar': '31) الاعتدال والانقلاب'},
 '32) Planetary Times': {'id': '32) Waktu Planet', 'en': '32) Planetary Times', 'ar': '32) أوقات الكواكب'},
 '33) Generator Analemma Matahari': {'id': '33) Generator Analema Matahari', 'en': '33) Solar Analemma Generator', 'ar': '33) مولد أناليمّا الشمس'},
 '34) Prediktor Pasang Surut Gravitasi': {'id': '34) Prediktor Pasang Surut Gravitasi', 'en': '34) Gravitational Tide Predictor',

'ar': '34) متنبئ المد والجزر الجذبي (34)',
 '35) Astrofotografi & Kontras Visibilitas': {'id': '35) Astrofotografi dan Kontras Visibilitas',
 'en': '35) Astrophotography and Visibility Contrast',
 'ar': '35) التصوير الفلكي وتباين الرؤية'},
 '36) Live Animasi': {'id': '36) Animasi Langsung', 'en': '36) Live Animation', 'ar': '36) الرسوم المتحركة المباشرة'},
 '37) Simulasi Ephemeris 3D': {'id': '37) Simulasi Ephemeris 3D', 'en': '37) 3D Ephemeris Simulation', 'ar':
 '37) محاكاة التقويم الفلكي ثلاثية الأبعاد'},
 '38) WinAI: Aplikasi AI Windows': {'id': '38) WinAI: Aplikasi AI Windows',
 'en': '38) WinAI: Windows AI Application',
 'ar': '38) WinAI: تطبيق ذكاء اصطناعي لويندوز'},
 '39) Peta Langit Dunia': {'id': '39) Peta Langit Dunia',
 'en': '39) World Sky Map',
 'ar': '39) خريطة السماء العالمية'},
 '40) Laporan Astronomi': {'id': '40) Laporan Astronomi',
 'en': '40) Astronomy Report',
 'ar': '40) التقرير الفلكي'},
 '41) HIJRI_DB Auto-Builder (Admin)': {'id': '41) Pembuat Otomatis HIJRI_DB (Admin)',
 'en': '41) HIJRI_DB Auto-Builder (Admin)',
 'ar': '41) (المشرف) HIJRI_DB المنشئ التلقائي'},
 'KHGT ENGINE': {'id': 'MESIN KHGT', 'en': 'KHGT ENGINE', 'ar': 'محرك KHGT'},
 '▶ PROSES DATA': {'id': '▶ PROSES DATA', 'en': '▶ PROCESS DATA', 'ar': '▶ معالجة البيانات'},
 'Silakan klik tombol ▶ PROSES DATA untuk menampilkan output.': {'id': 'Silakan klik tombol ▶
 PROSES DATA untuk menampilkan output.',
 'en': 'Please click the ▶ PROCESS DATA button to display the
 output.',
 'ar': 'يرجى النقر على زر ▶ معالجة البيانات لعرض النتائج'},
 'PILIH KOTA (DEFAULT SEMARANG)': {'id': 'PILIH KOTA (BAWAAN: SEMARANG)', 'en': 'SELECT CITY
 (DEFAULT: SEMARANG)', 'ar': 'اختر المدينة (الافتراضي: سمارانغ)'},
 '📍 Deteksi Lokasi Otomatis': {'id': '📍 Deteksi Lokasi Otomatis', 'en': '📍 Detect Location
 Automatically', 'ar': '📍 اكتشاف الموقع تلقائياً'},
 'PENGATURAN ALARM SALAT': {'id': 'PENGATURAN ALARM SALAT', 'en': 'PRAYER ALARM SETTINGS', 'ar':
 'إعدادات منبه الصلاة'},
 'Aktifkan Alarm': {'id': 'Aktifkan Alarm', 'en': 'Enable Alarm', 'ar': 'تفعيل المنبه'},
 '🎵 Pilih Audio Adzan': {'id': '🎵 Pilih Audio Azan', 'en': '🎵 Select Adhan Audio', 'ar': '🎵 اختيار صوت
 الأذان'},
 'TANGGAL OBSERVASI': {'id': 'TANGGAL OBSERVASI', 'en': 'OBSERVATION DATE', 'ar': 'تاريخ الرصد'},
 'TANGGAL OBSERVASI (SAAT MAGHRIB)': {'id': 'TANGGAL OBSERVASI (SAAT MAGRIB)', 'en':
 'OBSERVATION DATE (AT SUNSET)', 'ar': 'تاريخ الرصد (عند الغروب)'},
 'TAHUN OBSERVASI': {'id': 'TAHUN OBSERVASI', 'en': 'OBSERVATION YEAR', 'ar': 'سنة الرصد'},
 'BULAN OBSERVASI': {'id': 'BULAN OBSERVASI', 'en': 'OBSERVATION MONTH', 'ar': 'شهر الرصد'},
 'KOORDINAT LOKASI': {'id': 'KOORDINAT LOKASI', 'en': 'LOCATION COORDINATES', 'ar': 'إحداثيات الموقع'},
 'KOORDINAT OBSERVASI': {'id': 'KOORDINAT OBSERVASI', 'en': 'OBSERVATION COORDINATES', 'ar':
 'إحداثيات الرصد'},
 'KOORDINAT PANTAI': {'id': 'KOORDINAT PANTAI', 'en': 'COASTAL COORDINATES', 'ar': 'إحداثيات الساحل'},
 'KOORDINAT PENGAMAT': {'id': 'KOORDINAT PENGAMAT', 'en': 'OBSERVER COORDINATES', 'ar':
 'إحداثيات الراصد'},
 'LOKASI OBSERVASI': {'id': 'LOKASI OBSERVASI', 'en': 'OBSERVATION LOCATION', 'ar': 'موقع الرصد'},

'WAKTU OBSERVASI': {'id': 'WAKTU OBSERVASI', 'en': 'OBSERVATION TIME', 'ar': 'وقت الرصد'},
'TAHUN MASEHI': {'id': 'TAHUN MASEHI', 'en': 'GREGORIAN YEAR', 'ar': 'السنة الميلادية'},
'TANGGAL MASEHI': {'id': 'TANGGAL MASEHI', 'en': 'GREGORIAN DATE', 'ar': 'التاريخ الميلادي'},
'PENGATURAN KALENDER': {'id': 'PENGATURAN KALENDER', 'en': 'CALENDAR SETTINGS', 'ar': 'إعدادات التقويم'},
'PENGATURAN KAL. MASEHI': {'id': 'PENGATURAN KALENDER MASEHI', 'en': 'GREGORIAN CALENDAR SETTINGS', 'ar': 'إعدادات التقويم الميلادي'},
'Bulan:': {'id': 'Bulan:', 'en': 'Month:', 'ar': 'الشهر:'},
'Tahun (H):': {'id': 'Tahun (H):', 'en': 'Year (AH):', 'ar': 'السنة هـ'},
'Tahun (M):': {'id': 'Tahun (M):', 'en': 'Year (CE):', 'ar': 'السنة م'},
'🔍 Terapkan Pencarian': {'id': '🔍 Terapkan Pencarian', 'en': '🔍 Apply Search', 'ar': 'تطبيق البحث'},
'📅 Reset ke Bulan Ini': {'id': '📅 Atur Ulang ke Bulan Ini', 'en': '📅 Reset to Current Month', 'ar': 'إعادة الضبط إلى الشهر الحالي'},
'Memuat Ephemeris...': {'id': 'Memuat Ephemeris...', 'en': 'Loading Ephemeris...', 'ar': 'جارٍ تحميل التقويم الفلكي...'},
'Sistem Siap': {'id': 'Sistem Siap', 'en': 'System Ready', 'ar': 'النظام جاهز'},
'Alarm Salat: Memuat Jadwal...': {'id': 'Alarm Salat: Memuat Jadwal...', 'en': 'Prayer Alarm: Loading Schedule...', 'ar': 'منبه الصلاة: جارٍ تحميل الجدول...'},
'Cuaca: Menunggu deteksi...': {'id': 'Cuaca: Menunggu deteksi...', 'en': 'Weather: Waiting for detection...', 'ar': 'الطقس: في انتظار تحديد الموقع...'},
'Cuaca: Menunggu...': {'id': 'Cuaca: Menunggu...', 'en': 'Weather: Waiting...', 'ar': 'الطقس: في الانتظار...'},
'Laporan Analisis Hilal & KHGT': {'id': 'Laporan Analisis Hilal dan KHGT', 'en': 'Crescent and KHGT Analysis Report', 'ar': 'تقرير تحليل الهلال و KHGT'},
'Simpan Data ke TXT': {'id': 'Simpan Data ke TXT', 'en': 'Save Data to TXT', 'ar': 'TXT حفظ البيانات بصيغة'},
'Simpan Laporan sbg PNG': {'id': 'Simpan Laporan sebagai PNG', 'en': 'Save Report as PNG', 'ar': 'حفظ التقرير بصيغة PNG'},
'Simpan sebagai PDF': {'id': 'Simpan sebagai PDF', 'en': 'Save as PDF', 'ar': 'PDF حفظ بصيغة'},
'Simpan ke CSV (Excel / SPSS)': {'id': 'Simpan ke CSV (Excel/SPSS)', 'en': 'Save to CSV (Excel/SPSS)', 'ar': 'حفظ CSV (Excel/SPSS) بصيغة'},
'Kembali ke Beranda / Info Rilis': {'id': 'Kembali ke Beranda / Informasi Rilis', 'en': 'Return to Home / Release Information', 'ar': 'العودة إلى الرئيسية / معلومات الإصدار'},
'Menunggu input dari pengguna...': {'id': 'Menunggu input dari pengguna...', 'en': 'Waiting for user input...', 'ar': 'في انتظار إدخال المستخدم...'},
'PENGATURAN WAKTU & ELEVASI': {'id': 'PENGATURAN WAKTU DAN ELEVASI', 'en': 'TIME AND ELEVATION SETTINGS', 'ar': 'إعدادات الوقت والارتفاع'},
'PARAMETER ATMOSFER': {'id': 'PARAMETER ATMOSFER', 'en': 'ATMOSPHERIC PARAMETERS', 'ar': 'معاملات الغلاف الجوي'},
'PARAMETER TONGKAT & WAKTU': {'id': 'PARAMETER TONGKAT DAN WAKTU', 'en': 'ROD AND TIME PARAMETERS', 'ar': 'معاملات العصا والوقت'},
'PENGATURAN ANALEMMA': {'id': 'PENGATURAN ANALEMA', 'en': 'ANALEMMA SETTINGS', 'ar': 'إعدادات الأنايما'},
'TAHUN & JAM ANALISIS': {'id': 'TAHUN DAN JAM ANALISIS', 'en': 'ANALYSIS YEAR AND TIME', 'ar': 'سنة ووقت التحليل'},
'🔭 PREDIKTOR OKULTASI PLANET': {'id': '🔭 PREDIKTOR OKULTASI PLANET', 'en': '🔭 PLANETARY OCCULTATION PREDICTOR', 'ar': 'متنبئ احتجاب الكواكب'},

'Cari Okultasi Bulan Ini': {'id': 'Cari Okultasi Bulan Ini', 'en': 'Find Occultations This Month', 'ar': 'البحث عن حالات الاحتجاب هذا الشهر'},

'PROSES ANALEMMA': {'id': 'PROSES ANALEMA', 'en': 'PROCESS ANALEMMA', 'ar': 'معالجة الأنالِمَا'},

'🖨 Export Gambar HD': {'id': '🖨 Ekspor Gambar HD', 'en': '🖨 Export HD Image', 'ar': 'تصدير صورة عالية الدقة'},

'📄 Cetak PDF': {'id': '📄 Cetak PDF', 'en': '📄 Print PDF', 'ar': 'طباعة PDF'},

'📄 Log Teks': {'id': '📄 Log Teks', 'en': '📄 Text Log', 'ar': 'سجل نصي'},

'📄 Salin': {'id': '📄 Salin', 'en': '📄 Copy', 'ar': 'نسخ'},

'🔗 Share': {'id': '🔗 Bagikan', 'en': '🔗 Share', 'ar': 'مشاركة'},

'Best Time': {'id': 'Waktu Terbaik', 'en': 'Best Time', 'ar': 'أفضل وقت'},

'Location:': {'id': 'Lokasi:', 'en': 'Location:', 'ar': 'الموقع:'},

'Object:': {'id': 'Objek:', 'en': 'Object:', 'ar': 'الجسم:'},

'Time Ref:': {'id': 'Referensi Waktu:', 'en': 'Time Reference:', 'ar': 'مرجع الوقت:'},

'Sunset': {'id': 'Matahari Terbenam', 'en': 'Sunset', 'ar': 'غروب الشمس'},

'Moonset': {'id': 'Bulan Terbenam', 'en': 'Moonset', 'ar': 'غروب القمر'},

'New Crescent (Evening)': {'id': 'Hilal Baru (Petang)', 'en': 'New Crescent (Evening)', 'ar': 'الهلال الجديد (مساءً)'},

'Old Crescent (Morning)': {'id': 'Sabit Tua (Pagi)', 'en': 'Old Crescent (Morning)', 'ar': 'الهلال القديم (صباحًا)'},

'Sun is at Qiblah direction': {'id': 'Matahari berada pada arah kiblat', 'en': 'The Sun is in the Qibla direction', 'ar': 'الشمس في اتجاه القبلة'},

"Sun's shadow is at Qiblah": {'id': 'Bayangan Matahari berada pada arah kiblat', 'en': '"The Sun's shadow is in the Qibla direction"', 'ar': 'ظل الشمس في اتجاه القبلة'},

'Masehi → Hijriah (KHGT)': {'id': 'Masehi → Hijriah (KHGT)', 'en': 'Gregorian → Hijri (KHGT)', 'ar': 'ميلادي → هجري (KHGT)'},

'Hijriah (KHGT) → Masehi': {'id': 'Hijriah (KHGT) → Masehi', 'en': 'Hijri (KHGT) → Gregorian', 'ar': 'هجري (KHGT) → ميلادي'},

'Masehi → Hijriah (Urfi)': {'id': 'Masehi → Hijriah (Urfi)', 'en': 'Gregorian → Hijri (Tabular)', 'ar': 'ميلادي → هجري (حسابي)'},

'Hijriah (Urfi) → Masehi': {'id': 'Hijriah (Urfi) → Masehi', 'en': 'Hijri (Tabular) → Gregorian', 'ar': 'هجري (حسابي) → ميلادي'},

'Kalkulasi Selesai': {'id': 'Kalkulasi Selesai', 'en': 'Calculation Complete', 'ar': 'اكتمل الحساب'},

'Error Kalkulasi': {'id': 'Kesalahan Kalkulasi', 'en': 'Calculation Error', 'ar': 'خطأ في الحساب'},

'Mohon tunggu': {'id': 'Mohon tunggu', 'en': 'Please wait', 'ar': 'يرجى الانتظار'},

'Memproses data astrometri': {'id': 'Memproses data astrometri', 'en': 'Processing astrometric data', 'ar': 'جارٍ معالجة البيانات الفلكية'},

'Crescent Analysis & KHGT': {'id': 'Analisis Hilal dan KHGT', 'en': 'Crescent Analysis and KHGT', 'ar': 'تحليل KHGT والهلال'},

'Crescent Visibility Map Scanner': {'id': 'Pemindai Peta Visibilitas Hilal', 'en': 'Crescent Visibility Map Scanner', 'ar': 'ماسح خريطة رؤية الهلال'},

'Global Hilal Visibility Analyzer': {'id': 'Penganalisis Visibilitas Hilal Global', 'en': 'Global Crescent Visibility Analyzer', 'ar': 'محلل رؤية الهلال العالمي'},

'Moonphase Cycle Data': {'id': 'Data Siklus Fase Bulan', 'en': 'Moon Phase Cycle Data', 'ar': 'بيانات دورة أطوار القمر'}

'Sun and Moon Ephemeris Data': {'id': 'Data Ephemeris Matahari dan Bulan', 'en': 'Sun and Moon Ephemeris Data', 'ar': 'بيانات تقويم الشمس والقمر الفلكي'},

'Qiblah Direction Calculator': {'id': 'Kalkulator Arah Kiblat', 'en': 'Qibla Direction Calculator', 'ar': 'حاسبة اتجاه القبلة'},

'Moon Rise, Transit & Set': {'id': 'Waktu Terbit, Transit, dan Terbenam Bulan', 'en': 'Moonrise, Transit and Moonset', 'ar': 'شروق القمر وعبوره وغروبه'},

'Sun Rise, Transit & Set': {'id': 'Waktu Terbit, Transit, dan Terbenam Matahari', 'en': 'Sunrise, Transit and Sunset', 'ar': 'شروق الشمس وعبورها وغروبها'},

'Prayer Times Calculator': {'id': 'Kalkulator Waktu Salat', 'en': 'Prayer Times Calculator', 'ar': 'حاسبة مواقيت الصلاة'},

'Calendar Conversion': {'id': 'Konversi Kalender', 'en': 'Calendar Conversion', 'ar': 'تحويل التقويم'},

'Local Qibla Time Calculator': {'id': 'Kalkulator Waktu Kiblat Lokal', 'en': 'Local Qibla Time Calculator', 'ar': 'حاسبة وقت القبلة المحلي'},

'Precision Eclipse Calculator': {'id': 'Kalkulator Gerhana Presisi', 'en': 'Precision Eclipse Calculator', 'ar': 'حاسبة الكسوف والخسوف الدقيقة'},

'Live Celestial Object Tracker': {'id': 'Pelacak Langsung Benda Langit', 'en': 'Live Celestial Object Tracker', 'ar': 'متتبع الأجرام السماوية المباشر'},

'3D Geocentric Space System': {'id': 'Sistem Ruang Geosentrik 3D', 'en': '3D Geocentric Space System', 'ar': 'نظام فضائي مركزي أرضي ثلاثي الأبعاد'},

'Celestial Altitude Graphics': {'id': 'Grafik Ketinggian Benda Langit', 'en': 'Celestial Altitude Graphics', 'ar': 'رسوم ارتفاع الأجرام السماوية'},

'KHGT first Point (Mainland)': {'id': 'Titik Pertama KHGT (Daratan Utama)', 'en': 'First KHGT Point (Mainland)', 'ar': 'الْيَابَسَةُ الرَّئِيسِيَّةُ أول نقطة'},

'Equinox & Solstice Calculator': {'id': 'Kalkulator Ekuinoks dan Solstis', 'en': 'Equinox and Solstice Calculator', 'ar': 'حاسبة الاعتدال والانقلاب'},

'Rise, Transit & Set Planet': {'id': 'Waktu Terbit, Transit, dan Terbenam Planet', 'en': 'Planet Rise, Transit and Set', 'ar': 'شروق الكواكب وعبورها وغروبها'},

'3D Sun & Moon Ephemeris Simulation': {'id': 'Simulasi Ephemeris Matahari dan Bulan 3D', 'en': '3D Sun and Moon Ephemeris Simulation', 'ar': 'محاكاة تقويم الشمس والقمر ثلاثية الأبعاد'},

'Kalkulator Bayangan Tongkat Istiwa (Mizwala)': {'id': 'Kalkulator Bayangan Tongkat Istiwa (Mizwala)', 'en': 'Istiwa Rod Shadow Calculator (Mizwala)', 'ar': 'حاسبة ظل عصا الاستواء (المزولة)'},

'Prediktor Pasang Surut Gravitasi': {'id': 'Prediktor Pasang Surut Gravitasi', 'en': 'Gravitational Tide Predictor', 'ar': 'متنبئ المد والجزر الجذبوي'},

'Evaluasi Fajar Shadiq & SQM Teoritis': {'id': 'Evaluasi Fajar Sadik dan SQM Teoretis', 'en': 'True Dawn and Theoretical SQM Evaluation', 'ar': 'النظري SQM تقييم الفجر الصادق و'},

'Generator Analemma Matahari': {'id': 'Generator Analema Matahari', 'en': 'Solar Analemma Generator', 'ar': 'مولد أنالِمَا الشمس'}}}

fallback.update({

 "Klik PROSES DATA untuk render grafik": {

 "id": "Klik PROSES DATA untuk merender grafik",

 "en": "Click PROCESS DATA to render the chart",

 "ar": "انقر على معالجة البيانات لرسم المخطط"

 }

})

fallback.update({'1) Waktu Maghrib (Sunset)': {'id': '1) Waktu Magrib (Matahari Terbenam)', 'en': '1) Sunset Time', 'ar': 'وقت غروب الشمس (1)},
 '2) Waktu Tertentu (Kustom)': {'id': '2) Waktu Tertentu (Kustom)', 'en': '2) Specific Time (Custom)', 'ar': 'وقت محدد (مخصص) (2)},
 '3) Waktu Live (Sekarang)': {'id': '3) Waktu Langsung (Sekarang)', 'en': '3) Live Time (Now)', 'ar': 'الوقت (المباشر) الآن'},
 'AEROSPACE PRAYER TIMES CALCULATOR': {'id': 'KALKULATOR WAKTU SALAT PENERBANGAN',
 'en': 'AEROSPACE PRAYER TIMES CALCULATOR',
 'ar': 'حاسبة مواقيت الصلاة أثناء الطيران'},
 'ALASAN: -': {'id': 'ALASAN: -', 'en': 'REASON: -', 'ar': 'السبب: -'},
 'Berikutnya ▶': {'id': 'Berikutnya ▶', 'en': 'Next ▶', 'ar': 'التالي ▶'},
 'Bulan': {'id': 'Bulan', 'en': 'Month', 'ar': 'الشهر'},
 'Cari Sunset': {'id': 'Cari Waktu Matahari Terbenam', 'en': 'Find Sunset', 'ar': 'البحث عن وقت الغروب'},
 'Cari Waktu Bintang di Kiblat': {'id': 'Cari Waktu Bintang pada Arah Kiblat',
 'en': 'Find Star Time in the Qibla Direction',
 'ar': 'البحث عن وقت النجم في اتجاه القبلة'},
 'DEPARTURE (KEBERANGKATAN)': {'id': 'KEBERANGKATAN', 'en': 'DEPARTURE', 'ar': 'المغادرة'},
 'DESTINATION & AIRCRAFT': {'id': 'TUJUAN DAN PESAWAT', 'en': 'DESTINATION AND AIRCRAFT', 'ar': 'الوجهة والطائرة'},
 'GENERATE KURVA ALASKA': {'id': 'BUAT KURVA ALASKA', 'en': 'GENERATE ALASKA CURVE', 'ar': 'إنشاء منحنى ألاسكا'},
 'GRAFIK HARIAN (24 JAM)': {'id': 'GRAFIK HARIAN (24 JAM)', 'en': 'DAILY CHART (24 HOURS)', 'ar': 'المخطط اليومي (24 ساعة)'},
 'Hitung Jendela Astrofotografi': {'id': 'Hitung Jendela Astrofotografi', 'en': 'Calculate Astrophotography Window', 'ar': 'حساب نافذة التصوير الفلكي'},
 'KOREKSI HARI (IKHTIYAT)': {'id': 'KOREKSI HARI (IKHTIYAT)', 'en': 'DAY CORRECTION (PRECAUTION)', 'ar': 'تصحيح اليوم (الاحتياط)'},
 'Kalkulasi Dinamis berbasis Rute Great Circle, Kecepatan, dan Ketinggian': {'id': 'Kalkulasi Dinamis Berbasis Rute Lingkaran Besar, Kecepatan, dan Ketinggian',
 'en': 'Dynamic Calculation Based on Great-Circle Route, Speed, and Altitude',
 'ar': 'حساب ديناميكي مبني على مسار الدائرة العظمى والسرعة والارتفاع'},
 'Kirim Pertanyaan': {'id': 'Kirim Pertanyaan', 'en': 'Send Question', 'ar': 'إرسال السؤال'},
 'Klik & Drag layar untuk rotasi.\nScroll/Mousewheel untuk Zoom.': {'id': 'Klik dan seret layar untuk rotasi.\nGulir/roda tetikus untuk memperbesar.',
 'en': 'Click and drag to rotate.\nScroll/mouse wheel to zoom.',
 'ar': 'استخدم عجلة الفأرة للتكبير.\nانقر واسحب للتدوير'},
 'Legenda': {'id': 'Legenda', 'en': 'Legend', 'ar': 'مفتاح الرسم'},
 'MEMUAT...': {'id': 'MEMUAT...', 'en': 'LOADING...', 'ar': 'جارٍ التحميل'},
 'Matikan': {'id': 'Matikan', 'en': 'Turn Off', 'ar': 'إيقاف'},
 'Memeriksa sistem dan memuat data API...': {'id': 'Memeriksa sistem dan memuat data API...',
 'en': 'Checking the system and loading API data...',
 'ar': 'API... جارٍ فحص النظام وتحميل بيانات'},
 'Memuat data...': {'id': 'Memuat data...', 'en': 'Loading data...', 'ar': 'جارٍ تحميل البيانات'},
 'Memuat...': {'id': 'Memuat...', 'en': 'Loading...', 'ar': 'جارٍ التحميل'},
 'Mencari fase bulan': {'id': 'Mencari fase Bulan', 'en': 'Searching for lunar phase', 'ar': 'جارٍ البحث عن طور القمر'},

'Menganalisis...': {'id': 'Menganalisis...', 'en': 'Analyzing...', 'ar': 'جارٍ التحليل...'},
'Menghitung Ephemeris': {'id': 'Menghitung Ephemeris', 'en': 'Calculating Ephemeris', 'ar': 'جارٍ حساب
التقويم الفلكي'},
'Mengkalkulasi parameter astronomis...': {'id': 'Menghitung parameter astronomis...',
'en': 'Calculating astronomical parameters...',
'ar': 'جارٍ حساب المعاملات الفلكية'},
'Mengkalkulasi...': {'id': 'Menghitung...', 'en': 'Calculating...', 'ar': 'جارٍ الحساب...'},
'PILIH TAHUN GERHANA': {'id': 'PILIH TAHUN GERHANA', 'en': 'SELECT ECLIPSE YEAR', 'ar': 'اختر سنة
الكسوف أو الخسوف'},
'SET TANGGAL SIMULASI 3D': {'id': 'ATUR TANGGAL SIMULASI 3D', 'en': 'SET 3D SIMULATION DATE', 'ar':
'تعيين تاريخ المحاكاة ثلاثية الأبعاد'},
'SIMULASI ARAH BAYANGAN MATAHARI': {'id': 'SIMULASI ARAH BAYANGAN MATAHARI', 'en': 'SUN
SHADOW DIRECTION SIMULATION', 'ar': 'محاكاة اتجاه ظل الشمس'},
'SIMULATOR PARAMETER (MANUAL)': {'id': 'SIMULATOR PARAMETER (MANUAL)', 'en': 'PARAMETER
SIMULATOR (MANUAL)', 'ar': 'محاكي المعاملات (يدوي)'},
'STATUS: -': {'id': 'STATUS: -', 'en': 'STATUS: -', 'ar': 'الحالة: -'},
'Snooze 10m': {'id': 'Tunda 10 menit', 'en': 'Snooze 10 min', 'ar': 'غفوة 10 دقائق'},
'Snooze 5m': {'id': 'Tunda 5 menit', 'en': 'Snooze 5 min', 'ar': 'غفوة 5 دقائق'},
'Spesifikasi Penerbangan:': {'id': 'Spesifikasi Penerbangan:', 'en': 'Flight Specifications:', 'ar': 'مواصفات
الرحلة'},
'Status: Inisialisasi...': {'id': 'Status: Inisialisasi...', 'en': 'Status: Initializing...', 'ar': 'الحالة: جارٍ التهيئة'},
'TANGGAL PENCARIAN': {'id': 'TANGGAL PENCARIAN', 'en': 'SEARCH DATE', 'ar': 'تاريخ البحث'},
'TANGGAL REFERENSI (UTC)': {'id': 'TANGGAL REFERENSI (UTC)', 'en': 'REFERENCE DATE (UTC)', 'ar':
'(UTC) التاريخ المرجعي'},
'TREN HIJRIAH (RAJAB - ZULHIJAH)': {'id': 'TREN HIJRIAH (RAJAB-ZULHIJAH)', 'en': 'HIJRI TREND (RAJAB-
DHU AL-HIJJAH)', 'ar': 'الاتجاه الهجري (رجب-ذو الحجة)'},
'Tahun Hijriah:': {'id': 'Tahun Hijriah:', 'en': 'Hijri Year:', 'ar': 'السنة الهجرية'},
'Tanggal (YYYY-MM-DD)': {'id': 'Tanggal (TTTT-BB-HH)', 'en': 'Date (YYYY-MM-DD)', 'ar': '(YYYY-MM-DD) التاريخ'},
'Terapkan Waktu Kustom': {'id': 'Terapkan Waktu Kustom', 'en': 'Apply Custom Time', 'ar': 'تطبيق الوقت
المخصص'},
'WAKTU AKHIR (END)': {'id': 'WAKTU AKHIR', 'en': 'END TIME', 'ar': 'وقت النهاية'},
'WAKTU AWAL (START)': {'id': 'WAKTU AWAL', 'en': 'START TIME', 'ar': 'وقت البداية'},
'WAKTU SIMULASI 2D': {'id': 'WAKTU SIMULASI 2D', 'en': '2D SIMULATION TIME', 'ar': 'وقت المحاكاة ثنائية
الأبعاد'},
'WINAI ASSISTANT': {'id': 'ASISTEN WINAI', 'en': 'WINAI ASSISTANT', 'ar': 'مساعد WINAI'},
'Waktu Berangkat (Lokal)': {'id': 'Waktu Berangkat (Lokal)', 'en': 'Departure Time (Local)', 'ar': 'وقت
المغادرة (محلي)'},
'Waktu Lokal: --:--:--': {'id': 'Waktu Lokal: --:--:--', 'en': 'Local Time: --:--:--', 'ar': 'الوقت المحلي: --:--:--'},
'⌚ Countdown Ijtimak': {'id': '⌚ Hitung Mundur Ijtimak', 'en': '⌚ Conjunction Countdown', 'ar': '⌚
العد التنازلي للاقتران'},
'⌚ Sedang Memuat Mesin Ephemeris NASA, harap tunggu...': {'id': '⌚ Sedang Memuat Mesin
Ephemeris NASA, harap tunggu...'},
'en': '⌚ Loading the NASA Ephemeris Engine, please wait...',
'ar': '⌚ ...جارٍ تحميل محرك التقويم الفلكي لناسا، يرجى الانتظار'},
'▶ HITUNG JADWAL SELAMA PENERBANGAN': {'id': '▶ HITUNG JADWAL SELAMA PENERBANGAN',
'en': '▶ CALCULATE IN-FLIGHT SCHEDULE', 'ar': '▶ حساب الجدول أثناء الرحلة'},

'▶ Play / Pause Animasi': {'id': '▶ Putar / Jeda Animasi', 'en': '▶ Play / Pause Animation', 'ar': '▶ تشغيل / إيقاف مؤقت للرسوم'},

'▶ TERAPKAN TANGGAL': {'id': '▶ TERAPKAN TANGGAL', 'en': '▶ APPLY DATE', 'ar': '▶ تطبيق التاريخ'},

'◀ Sebelumnya': {'id': '◀ Sebelumnya', 'en': '◀ Previous', 'ar': '◀ السابق'},

'☀ MATAHARI': {'id': '☀ MATAHARI', 'en': '☀ SUN', 'ar': '☀ الشمس'},

'⚙ GENERATOR DATABASE KHGT': {'id': '⚙ GENERATOR BASIS DATA KHGT', 'en': '⚙ KHGT DATABASE GENERATOR', 'ar': '⚙ مولد قاعدة بيانات KHGT'},

'✈ Shalat Penerbangan (In-Flight)': {'id': '✈ Salat dalam Penerbangan', 'en': '✈ In-Flight Prayer', 'ar': '✈ الصلاة أثناء الطيران'},

'🌅 Set ke Waktu Sunset': {'id': '🌅 Atur ke Waktu Matahari Terbenam', 'en': '🌅 Set to Sunset Time', 'ar': '🌅 الضبط على وقت الغروب'},

'🌐 3D GEOCENTRIC VIEW': {'id': '🌐 TAMPILAN GEOSENTRIK 3D', 'en': '🌐 3D GEOCENTRIC VIEW', 'ar': '🌐 عرض مركزي أرضي ثلاثي الأبعاد'},

'🌐 CELESTIAL QIBLAH (BINTANG)': {'id': '🌐 KIBLAT LANGIT (BINTANG)', 'en': '🌐 CELESTIAL QIBLA (STAR)', 'ar': '🌐 القبلة السماوية (النجم)'},

'🌐 Akses Kiblat Online': {'id': '🌐 Akses Kiblat Online', 'en': '🌐 Online Qibla Access', 'ar': '🌐 الوصول إلى القبلة عبر الإنترنت'},

'🌐 Tracker': {'id': '🌐 Pelacak', 'en': '🌐 Tracker', 'ar': '🌐 المتتبع'},

'🌑 Simulasi Visual Gerhana': {'id': '🌑 Simulasi Visual Gerhana', 'en': '🌑 Visual Eclipse Simulation', 'ar': '🌑 محاكاة بصرية للكسوف والخسوف'},

'🌑 ECLIPSE 2026': {'id': '🌑 GERHANA 2026', 'en': '🌑 ECLIPSE 2026', 'ar': '🌑 الكسوف والخسوف 2026'},

'🌙 BULAN': {'id': '🌙 BULAN', 'en': '🌙 MOON', 'ar': '🌙 القمر'},

'🌙 Status Hilal Nanti Sore': {'id': '🌙 Status Hilal Nanti Sore', 'en': '🌙 Crescent Status This Evening', 'ar': '🌙 حالة الهلال هذا المساء'},

'🌤 Cuaca Lokal': {'id': '🌤 Cuaca Lokal', 'en': '🌤 Local Weather', 'ar': '🌤 الطقس المحلي'},

'👁 Buka Simulasi Visual Mizwala': {'id': '👁 Buka Simulasi Visual Mizwala', 'en': '👁 Open Mizwala Visual Simulation', 'ar': '👁 افتح المحاكاة البصرية للمزولة'},

'📄 Cetak PDF (Portrait)': {'id': '📄 Cetak PDF (Potret)', 'en': '📄 Print PDF (Portrait)', 'ar': '📄 طباعة PDF (عمودي)'},

'📈 Alt Chart': {'id': '📈 Grafik Ketinggian', 'en': '📈 Altitude Chart', 'ar': '📈 مخطط الارتفاع'},

'📈 Elongasi': {'id': '📈 Elongasi', 'en': '📈 Elongation', 'ar': '📈 الاستطالة'},

'📈 Ill vs Eln': {'id': '📈 Iluminasi vs Elongasi', 'en': '📈 Illumination vs Elongation', 'ar': '📈 الإضاءة مقابل الاستطالة'},

'📈 Iluminasi': {'id': '📈 Iluminasi', 'en': '📈 Illumination', 'ar': '📈 الإضاءة'},

'📈 PARAMETER KURVA ALASKA': {'id': '📈 PARAMETER KURVA ALASKA', 'en': '📈 ALASKA CURVE PARAMETERS', 'ar': '📈 معاملات منحنى ألاسكا'},

'📊 EXECUTIVE SUMMARY': {'id': '📊 RINGKASAN EKSEKUTIF', 'en': '📊 EXECUTIVE SUMMARY', 'ar': '📊 الملخص التنفيذي'},

'📊 Tren Altitude (Geo vs Topo)': {'id': '📊 Tren Ketinggian (Geo vs Topo)', 'en': '📊 Altitude Trend (Geo vs Topo)', 'ar': '📊 اتجاه الارتفاع (مركزي أرضي مقابل موضعي)'},

'📊 Tren Elongasi (Geo vs Topo)': {'id': '📊 Tren Elongasi (Geo vs Topo)', 'en': '📊 Tren Elongasi (Geo vs Topo)', 'ar': '📊 اتجاه الاستطالة (مركزي أرضي مقابل موضعي)'}

'en': ' 📏 Elongation Trend (Geo vs Topo)',
 'ar': ' 📏 اتجاه الاستطالة (مركزي أرضي مقابل موضعي)',
 ' 📏 Tren Iluminasi Siklik': {'id': ' 📏 Tren Iluminasi Siklik', 'en': ' 📏 Cyclic Illumination Trend', 'ar': ' 📏 اتجاه الإضاءة الدوري'},
 ' 📏 Tren Kombinasi (Alt, Eln, Illum)': {'id': ' 📏 Tren Kombinasi (Tinggi, Elongasi, Iluminasi)',
 'en': ' 📏 Combined Trend (Altitude, Elongation, Illumination)',
 'ar': ' 📏 الاتجاه المركب (الارتفاع، الاستطالة، الإضاءة)'},
 ' 📌 Catatan & Hari Penting (Nasional/Internasional)': {'id': ' 📌 Catatan dan Hari Penting (Nasional/Internasional)',
 'en': ' 📌 Notes and Important Days (National/International)',
 'ar': ' 📌 ملاحظات وأيام مهمة (وطنية/دولية)'},
 ' 📍 Memuat Lokasi...': {'id': ' 📍 Memuat Lokasi...', 'en': ' 📍 Loading Location...', 'ar': ' 📍 جاري تحميل الموقع...'},
 ' 📍 Memuat data lokasi dan tanggal...': {'id': ' 📍 Memuat data lokasi dan tanggal...',
 'en': ' 📍 Loading location and date data...',
 'ar': ' 📍 جاري تحميل بيانات الموقع والتاريخ'},
 ' 📌 Catatan & Hari Penting Islam': {'id': ' 📌 Catatan dan Hari Penting Islam',
 'en': ' 📌 Notes and Important Islamic Days',
 'ar': ' 📌 ملاحظات وأيام إسلامية مهمة'},
 ' 📷 Modul 31: Astrofotografi Hilal & Visibilitas Kontras': {'id': ' 📷 Modul 31: Astrofotografi Hilal dan Visibilitas Kontras',
 'en': ' 📷 Module 31: Crescent Astrophotography and Visibility Contrast',
 'ar': ' 📷 الوحدة 31: تصوير الهلال وتباين الرؤية'},
 ' 📷 Simpan PNG': {'id': ' 📷 Simpan PNG', 'en': ' 📷 Save PNG', 'ar': ' 📷 حفظ PNG'},
 ' 📷 Simpan PNG (Portrait)': {'id': ' 📷 Simpan PNG (Potret)', 'en': ' 📷 Save PNG (Portrait)', 'ar': ' 📷 حفظ PNG (عمودي)'},
 ' 🔄 Reset Memori': {'id': ' 🔄 Atur Ulang Memori', 'en': ' 🔄 Reset Memory', 'ar': ' 🔄 إعادة ضبط الذاكرة'},
 ' 🔍 Zoom Hilal': {'id': ' 🔍 Perbesar Hilal', 'en': ' 🔍 Zoom Crescent', 'ar': ' 🔍 تكبير الهلال'},
 ' 📏 KOMPARASI DATA: STELLARIUM vs KOMPUTASI ENGINE': {'id': ' 📏 KOMPARASI DATA: STELLARIUM vs KOMPUTASI MESIN',
 'en': ' 📏 DATA COMPARISON: STELLARIUM vs ENGINE COMPUTATION',
 'ar': ' 📏 مقارنة البيانات: ستيلاريوم مقابل حساب المحرك'},
 ' 📏 Lihat Detail Lokal Saat Puncak': {'id': ' 📏 Lihat Detail Lokal Saat Puncak',
 'en': ' 📏 View Local Details at Maximum',
 'ar': ' 📏 عرض التفاصيل المحلية عند الذروة'},
 ' 📏 Stellarium Sync': {'id': ' 📏 Sinkronisasi Stellarium', 'en': ' 📏 Stellarium Sync', 'ar': ' 📏 مزامنة ستيلاريوم'},
 ' 🟢 Kembali ke Live (Sekarang)': {'id': ' 🟢 Kembali ke Waktu Langsung (Sekarang)', 'en': ' 🟢 Return to Live (Now)', 'ar': ' 🟢 العودة إلى البث المباشر (الآن)'},
 ' 🟢 LIVE SIMULATOR': {'id': ' 🟢 SIMULATOR LANGSUNG', 'en': ' 🟢 LIVE SIMULATOR', 'ar': ' 🟢 المحاكى المباشر'},
 ' 🟢 Waktu Live': {'id': ' 🟢 Waktu Langsung', 'en': ' 🟢 Live Time', 'ar': ' 🟢 الوقت المباشر'},
 ' 📄 Export Jalur Totalitas/Anularitas ke KML': {'id': ' 📄 Ekspor Jalur Totalitas/Anularitas ke KML',

```

'en': '🗺️ Export Totality/Annularity Path to KML',
'ar': '🗺️ تصدير مسار الكلية/الحلقية إلى KML'},
'🗺️ Show Qiblah Map': {'id': '🗺️ Tampilkan Peta Kiblat', 'en': '🗺️ Show Qibla Map', 'ar': '🗺️ عرض خريطة القبلة'},
'🤖 Ruang Interaksi AI': {'id': '🤖 Ruang Interaksi AI', 'en': '🤖 AI Interaction Space', 'ar': '🤖 مساحة التفاعل مع الذكاء الاصطناعي'},
'Bujur (Longitude)': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'خط الطول:'},
'Durasi (Jam)': {'id': 'Durasi (Jam):', 'en': 'Duration (Hours):', 'ar': 'المدة (ساعات):'},
'Elev.': {'id': 'Elevasi:', 'en': 'Elevation:', 'ar': 'الارتفاع:'},
'Elevasi (m)': {'id': 'Elevasi (m):', 'en': 'Elevation (m):', 'ar': 'الارتفاع (م):'},
'Ikhtiyat (Detik)': {'id': 'Ikhtiyat (Detik):', 'en': 'Precaution (Seconds):', 'ar': 'الاحتياط (ثوانٍ):'},
'Interval (Menit)': {'id': 'Interval (Menit):', 'en': 'Interval (Minutes):', 'ar': 'الفاصل (دقائق):'},
'Jam (0-23)': {'id': 'Jam (0-23):', 'en': 'Hour (0-23):', 'ar': 'الساعة (23-0):'},
'Jam (HH:MM)': {'id': 'Jam (JJ:MM):', 'en': 'Time (HH:MM):', 'ar': '(HH:MM) الوقت:'},
'Jam Pengamatan (Lokal)': {'id': 'Jam Pengamatan (Lokal):', 'en': 'Observation Time (Local):', 'ar': 'وقت الرصد (محلي):'},
'Kelembapan (%)': {'id': 'Kelembapan (%):', 'en': 'Humidity (%):', 'ar': 'الرطوبة (%):'},
'Ketinggian (Kaki/ft)': {'id': 'Ketinggian (Kaki/ft):', 'en': 'Altitude (Feet/ft):', 'ar': 'الارتفاع (قدم):'},
'Lat (Lintang)': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'خط العرض:'},
'Lat Berangkat': {'id': 'Lintang Keberangkatan:', 'en': 'Departure Latitude:', 'ar': 'خط عرض المغادرة:'},
'Lat Tujuan': {'id': 'Lintang Tujuan:', 'en': 'Destination Latitude:', 'ar': 'خط عرض الوجهة:'},
'Lat:': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'خط العرض:'},
'Lintang (Latitude)': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'خط العرض:'},
'Lon (Bujur)': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'خط الطول:'},
'Lon Berangkat': {'id': 'Bujur Keberangkatan:', 'en': 'Departure Longitude:', 'ar': 'خط طول المغادرة:'},
'Lon Tujuan': {'id': 'Bujur Tujuan:', 'en': 'Destination Longitude:', 'ar': 'خط طول الوجهة:'},
'Lon:': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'خط الطول:'},
'Suhu (°C)': {'id': 'Suhu (°C):', 'en': 'Temperature (°C):', 'ar': 'درجة الحرارة (°م):'},
'TZ:': {'id': 'Zona Waktu:', 'en': 'Time Zone:', 'ar': 'المنطقة الزمنية:'},
'Tahun (YYYY)': {'id': 'Tahun (TTTT):', 'en': 'Year (YYYY):', 'ar': 'السنة (YYYY):'},
'Tahun Akhir (H)': {'id': 'Tahun Akhir (H):', 'en': 'End Year (AH):', 'ar': 'سنة النهاية (هـ):'},
'Tahun Analisis:': {'id': 'Tahun Analisis:', 'en': 'Analysis Year:', 'ar': 'سنة التحليل:'},
'Tahun Awal (H)': {'id': 'Tahun Awal (H):', 'en': 'Start Year (AH):', 'ar': 'سنة البداية (هـ):'},
'Tekanan (mb)': {'id': 'Tekanan (mb):', 'en': 'Pressure (mb):', 'ar': 'الضغط (مليبار):'},
'Timezone Awal:': {'id': 'Zona Waktu Awal:', 'en': 'Initial Time Zone:', 'ar': 'المنطقة الزمنية الابتدائية:'},
'Timezone:': {'id': 'Zona Waktu:', 'en': 'Time Zone:', 'ar': 'المنطقة الزمنية:'},
'Tinggi (cm)': {'id': 'Tinggi (cm):', 'en': 'Height (cm):', 'ar': 'الارتفاع (سم):'},
'Waktu (HH:MM)': {'id': 'Waktu (JJ:MM):', 'en': 'Time (HH:MM):', 'ar': '(HH:MM) الوقت:'}}

```

```

# Cache eksternal yang sudah dimuat tetap menjadi prioritas.
for source_text, langs in fallback.items():
    if not isinstance(source_text, str) or not isinstance(langs, dict):
        continue
    entry = self.translations.setdefault(source_text, {})
    for lang in ("id", "en", "ar"):
        value = langs.get(lang)
        if isinstance(value, str) and value:

```

```

        entry.setdefault(lang, value)

self._builtin_translations_installed = True
self._build_language_indexes()

def _norm_lang_text(self, value):
    value = str(value or "").strip().replace("'", "")
    value = re.sub(r"\s+", " ", value)
    return value.casefold()

def tr(self, text):
    """Terjemahan exact-match dari varian ID/EN/AR ke bahasa aktif; O(1)."""
    if not text:
        return text
    self.install_builtin_translations()
    lang = getattr(self, "current_lang", "id")
    if lang not in ("id", "en", "ar"):
        lang = "id"

    canonical_id = self._reverse_translation_map.get(
        self._norm_lang_text(text), text
    )
    mapping = self._translation_index.get(lang, {})
    return mapping.get(
        canonical_id,
        mapping.get(text, canonical_id if lang == "id" else text)
    )

def reverse_tr(self, current_text):
    """Mengembalikan teks Indonesia asli dari tampilan ID/EN/AR; O(1)."""
    if not current_text:
        return current_text
    self.install_builtin_translations()
    return self._reverse_translation_map.get(self._norm_lang_text(current_text), current_text)

def change_language(self, choice):
    """Dipanggil saat pilihan bahasa id/en/ar diubah."""
    choice_norm = str(choice or "").strip().lower()
    self.current_lang = choice_norm if choice_norm in ("id", "en", "ar") else "id"

    # Bahasa berlaku ke seluruh antarmuka: sidebar kiri, header,
    # tombol, frame output, dan isi laporan di ruang kanan.
    self.force_update_sidebar_language()
    self.update_ui_texts(self)
    self.force_update_sidebar_language()
    self.apply_language_direction()

```

```

if getattr(self, "active_right_view", "") == "release_info":
    # Beranda digambar ulang hanya bila pengguna memang sedang di beranda.
    self.show_release_info()
else:
    self.refresh_right_output_language()
    self.refresh_secondary_output_languages()

try:
    if hasattr(self, "combo_lang"):
        self.combo_lang.configure(values=["id", "en", "ar"])
        self.combo_lang.set(self.current_lang)
except Exception:
    pass

def refresh_right_output_language(self):
    """Gambar ulang semua teks utama ruang kanan dari teks mentah.
    Prioritas: laporan kalkulasi terakhir; jika tidak ada, buffer mentah
    terakhir dari self.textbox.insert(...) dipakai. Dengan cara ini,
    output kanan tidak berhenti pada sidebar saja.
    """
    if not hasattr(self, "textbox"):
        return
    if getattr(self, "active_right_view", "") == "release_info":
        self.show_release_info()
        return
    raw_text = getattr(self, '_last_raw_report', '') or getattr(self, '_last_raw_textbox_content', '')
    if not raw_text:
        try:
            raw_text = self.textbox.get("1.0", "end-1c")
        except Exception:
            raw_text = ""
    if raw_text:
        try:
            self.textbox.configure(state="normal")
            self.textbox.delete("1.0", "end")
            self.textbox.insert("1.0", raw_text)
            self.textbox.configure(state="disabled")
        except Exception:
            pass

def get_release_lang_cache_path(self):
    """Lokasi file cache JSON khusus INFORMASI RILIS."""
    return self._find_cache_file("lang_cache_release.json")

def get_output_lang_cache_path(self):
    """Lokasi cache JSON untuk hasil translasi blok/output penuh."""
    return self._find_cache_file("lang_output_cache.json")

```

```

def _output_cache_key(self, lang, raw):
    return hashlib.sha256((str(lang) + "\0" + str(raw)).encode("utf-8", errors="ignore")).hexdigest()

def load_release_language_cache(self):
    """Membaca cache JSON terjemahan halaman INFORMASI RILIS."""
    self.release_translation_cache = {"en": {}, "ar": {}}
    try:
        cache_path = getattr(self, "release_lang_cache_path", None) or
self.get_release_lang_cache_path()
        self.release_lang_cache_path = cache_path
        if os.path.exists(cache_path):
            with open(cache_path, "r", encoding="utf-8") as f:
                data = json.load(f)
                if isinstance(data, dict):
                    for lang in ("en", "ar"):
                        if isinstance(data.get(lang), dict):
                            self.release_translation_cache[lang].update({
                                str(k): str(v) for k, v in data[lang].items() if isinstance(v, str)
                            })
            # Masukkan frasa release ke kamus UI agar fallback per-frasa tetap ada.
            self._merge_translation_mapping(self.release_translation_cache)
            self._release_translation_keys = set(self.release_translation_cache.get("en", {})) |
set(self.release_translation_cache.get("ar", {}))
            self._build_language_indexes()
        except Exception as e:
            print(f"Peringatan: gagal membaca lang_cache_release.json ({e})")

def save_release_language_cache(self, force=False):
    """Menyimpan cache JSON INFORMASI RILIS bila berubah."""
    if not force and not getattr(self, "_release_translation_cache_dirty", False):
        return
    try:
        cache_path = getattr(self, "release_lang_cache_path", None) or
self.get_release_lang_cache_path()
        self.release_lang_cache_path = cache_path
        data = getattr(self, "release_translation_cache", None) or {"en": {}, "ar": {}}
        with open(cache_path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=2)
        self._release_translation_cache_dirty = False
    except Exception as e:
        print(f"Peringatan: gagal menyimpan lang_cache_release.json ({e})")

def save_output_language_cache(self, force=False):
    """Menyimpan cache translasi blok/output penuh bila berubah."""
    if not force and not getattr(self, "_lang_output_cache_dirty", False):
        return
    try:
        cache_path = self.get_output_lang_cache_path()

```

```

        data = getattr(self, "_lang_output_cache", {}) or {}
        with open(cache_path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=2)
        self._lang_output_cache_dirty = False
    except Exception as e:
        print(f"Peringatan: gagal menyimpan lang_output_cache.json ({e})")

def _get_translation_regexes(self, lang):
    """Kompilasi regex sekali per bahasa untuk mengganti ribuan frasa dalam satu-dua lintasan."""
    if lang in getattr(self, "_translation_patterns", {}):
        return self._translation_patterns[lang]

    mapping = self._translation_index.get(lang, {}) if hasattr(self, "_translation_index") else {}
    simple_keys = []
    bounded_keys = []
    for key, trans in mapping.items():
        if not key or not trans or trans == key:
            continue
        # Kata alfabetis/pendek diberi batas kata agar tidak merusak kata lain.
        if len(key) <= 3 or key.isalpha():
            bounded_keys.append(key)
        else:
            simple_keys.append(key)
    simple_keys.sort(key=len, reverse=True)
    bounded_keys.sort(key=len, reverse=True)

    def compile_alt(keys, bounded=False):
        if not keys:
            return None
        alt = "|".join(re.escape(k) for k in keys)
        if bounded:
            alt = r"(?!\\w)(?:" + alt + r")"
        return re.compile(alt)

    try:
        patterns = (compile_alt(simple_keys, False), compile_alt(bounded_keys, True))
    except Exception:
        patterns = (None, None)
    self._translation_patterns[lang] = patterns
    return patterns

def _replace_by_language_map(self, raw, lang, release_only=False):
    result = str(raw)
    mapping = self._translation_index.get(lang, {}) if hasattr(self, "_translation_index") else {}
    if not mapping:
        return result

    if release_only:

```



```

keys = getattr(self, "_release_translation_keys", set()) or set(mapping.keys())
sorted_keys = getattr(self, "_release_sorted_keys_cache", None)
if sorted_keys is None:
    sorted_keys = sorted(keys, key=len, reverse=True)
    self._release_sorted_keys_cache = sorted_keys
for key in sorted_keys:
    trans = mapping.get(key)
    if trans and trans != key and key in result:
        result = result.replace(key, trans)
return result

simple_pattern, bounded_pattern = self._get_translation_regexes(lang)
def repl(match):
    return mapping.get(match.group(0), match.group(0))
try:
    if simple_pattern is not None:
        result = simple_pattern.sub(repl, result)
    if bounded_pattern is not None:
        result = bounded_pattern.sub(repl, result)
    return result
except Exception:
    # Fallback aman seperti metode lama, tetapi tetap hanya saat regex gagal.
    sorted_keys = getattr(self, "_translation_sorted_keys_cache", None)
    if sorted_keys is None:
        sorted_keys = sorted(mapping.keys(), key=len, reverse=True)
        self._translation_sorted_keys_cache = sorted_keys
    for key in sorted_keys:
        trans = mapping.get(key)
        if trans and trans != key and key in result:
            result = result.replace(key, trans)
    return result

def _translate_release_piece_fast(self, text):
    """Terjemahan cepat satu potongan release memakai cache + kamus release saja."""
    raw = str(text or "")
    lang = getattr(self, "current_lang", "id")
    if not raw or lang == "id":
        return raw
    self.install_builtin_translations()

    stripped = raw.strip()
    leading = raw[:len(raw) - len(raw.lstrip())]
    trailing = raw[len(raw.rstrip()):]
    exact = self._translation_index.get(lang, {}).get(stripped)
    if exact:
        return leading + exact + trailing

    if "\n" in raw:

```

```

translated_lines = []
changed = False
for line in raw.splitlines(keepends=True):
    newline = ""
    body = line
    if line.endswith("\r\n"):
        body, newline = line[:-2], "\r\n"
    elif line.endswith("\n"):
        body, newline = line[:-1], "\n"
    new_body = self._translate_release_piece_fast(body) if body.strip() else body
    changed = changed or (new_body != body)
    translated_lines.append(new_body + newline)
if changed:
    return "".join(translated_lines)

return self._replace_by_language_map(raw, lang, release_only=True)

def translate_release_text_cached(self, text):
    """Ambil/simpan terjemahan release dari lang_cache_release + lang_output_cache."""
    raw = text if isinstance(text, str) else str(text)
    lang = getattr(self, "current_lang", "id")
    if not raw or lang == "id" or lang not in ("en", "ar"):
        return raw

    cache = getattr(self, "release_translation_cache", None)
    if not isinstance(cache, dict):
        self.load_release_language_cache()
        cache = self.release_translation_cache
    cache.setdefault("en", {})
    cache.setdefault("ar", {})

    # Kamus highlight V8.0 mendahului kedua cache: jangan gunakan hasil
    # lama yang masih ID/campuran ketika pengguna memilih EN atau AR.
    translated = self._translate_release_highlight_piece(raw, lang)
    out_cache = getattr(self, "_lang_output_cache", {}) or {}
    out_key = self._output_cache_key(lang, raw)
    if translated is None:
        if raw in cache[lang]:
            return cache[lang][raw]
        if out_key in out_cache:
            translated = out_cache[out_key]
        else:
            translated = self._translate_release_piece_fast(raw)

    if cache[lang].get(raw) != translated:
        cache[lang][raw] = translated
        self._release_translation_cache_dirty = True
    if out_cache.get(out_key) != translated:

```

```

        out_cache[out_key] = translated
        self._lang_output_cache_dirty = True
    self._lang_output_cache = out_cache
    return translated

```

```

def _fix_visibility_id_alignment(self, text):

```

```

    """Rapikan alignment khusus output ID pada modul Visibilitas Hilal (KHGT).

```

```

    Translasi dilakukan setelah string laporan selesai diberi padding.

```

```

    Akibatnya penggantian "Sunset" -> "Matahari Terbenam" dan

```

```

    "Moonset" -> "Bulan Terbenam" membuat posisi titik dua bergeser.

```

```

    Fungsi ini hanya menyentuh dua baris tersebut pada laporan visibilitas,
    tanpa mengubah nilai waktu maupun kalkulasi astronomis.

```

```

    """

```

```

    if not text:

```

```

        return text

```

```

    result_lines = []

```

```

    for line in str(text).splitlines(keepends=True):

```

```

        # Baris Sunset pada report visibilitas memiliki pasangan "G. Moon Age".

```

```

        if "Matahari Terbenam" in line and "G. Moon Age" in line:

```

```

            line = re.sub(
                r"Matahari Terbenam\s*:",
                f'{"Matahari Terbenam":<22}: ',
                line,
                count=1,
            )

```

```

        # Baris Moonset pada report visibilitas memiliki pasangan "Moon Lag Time".

```

```

        elif "Bulan Terbenam" in line and "Moon Lag Time" in line:

```

```

            line = re.sub(
                r"Bulan Terbenam\s*:",
                f'{"Bulan Terbenam":<22}: ',
                line,
                count=1,
            )

```

```

        result_lines.append(line)

```

```

    return "".join(result_lines)

```

```

def translate_text_block(self, text):

```

```

    """Menerjemahkan blok string laporan/output kanan secara cepat dan cached."""

```

```

    if not text:

```

```

        return text

```

```

    lang = getattr(self, "current_lang", "id")

```

```

    if lang not in ("id", "en", "ar"):

```

```

        lang = "id"

```

```

raw = text if isinstance(text, str) else str(text)

# Hook Textbox juga harus memperbaiki cache lama untuk judul,
# heading, dan uraian highlight, termasuk teks dengan pembatas/tag.
is_release = (getattr(self, "active_right_view", "") == "release_info"
               or getattr(self, "_rendering_release_info", False))
if is_release and self._translate_release_highlight_piece(raw, lang) is not None:
    return self.translate_release_text_cached(raw)

out_cache = getattr(self, "_lang_output_cache", {}) or {}
out_key = self._output_cache_key(lang, raw)
if out_key in out_cache:
    cached = out_cache[out_key]
    if lang == "id":
        cached = self._fix_visibility_id_alignment(cached)
        out_cache[out_key] = cached
    return cached

if getattr(self, "active_right_view", "") == "release_info" or getattr(self, "_rendering_release_info",
False):
    translated = self.translate_release_text_cached(raw)
else:
    self.install_builtin_translations()
    translated = self._replace_by_language_map(raw, lang, release_only=False)

# KHUSUS ID - Visibilitas Hilal (KHGT):
# rapikan kembali dua label yang memanjang setelah translasi.
if lang == "id":
    translated = self._fix_visibility_id_alignment(translated)

# Batasi ukuran cache agar file JSON tidak tumbuh tak terkendali.
if len(out_cache) > 5000:
    for k in list(out_cache.keys())[:1000]:
        out_cache.pop(k, None)
    out_cache[out_key] = translated
    self._lang_output_cache = out_cache
    self._lang_output_cache_dirty = True
    return translated

def get_process_prompt_text(self):
    """Pesan standar ketika modul menunggu tombol PROSES DATA."""
    messages = {
        "id": "Silakan klik tombol  PROSES DATA untuk menampilkan output.",
        "en": "Please click the  PROCESS DATA button to display the output.",
        "ar": "يرجى النقر على زر  معالجة البيانات لعرض النتائج.",
    }

```

```

return messages.get(getattr(self, "current_lang", "id"), messages["id"])

def prepare_module_right_output(self):
    """Kosongkan sisa beranda/laporan lama ketika pengguna memilih menu."""
    self.active_right_view = "module_pending"
    self._last_raw_report = ""
    raw_prompt = "Silakan klik tombol  PROSES DATA untuk menampilkan output."
    self._last_raw_textbox_content = raw_prompt

    try:
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(state="normal", font=("Consolas", 13), wrap="word")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", raw_prompt)
        self.textbox.configure(state="disabled")
    except Exception:
        pass
    finally:
        self.active_right_view = "module_pending"

def show_pending_message_in_frame(self, frame):
    """Tampilkan petunjuk terlokalisasi pada frame output non-teks yang belum diproses."""
    try:
        for widget in frame.winfo_children():
            widget.destroy()
        label = ctk.CTkLabel(
            frame,
            text=self.get_process_prompt_text(),
            font=("Segoe UI", 17, "bold"),
            text_color="#00E5FF",
            wraplength=700,
            justify="right" if getattr(self, "current_lang", "id") == "ar" else "center",
        )
        label.pack(fill="both", expand=True, padx=30, pady=30)
        setattr(
            label,
            "_khgt_id_text",
            "Silakan klik tombol  PROSES DATA untuk menampilkan output."
        )
    except Exception:
        pass

def install_secondary_output_textbox_hooks(self):
    """Pasang hook bahasa pada CTkTextbox output selain textbox utama."""
    if not hasattr(self, "main_frame"):
        return

```

```

def walk(widget):
    try:
        children = widget.winfo_children()
    except Exception:
        children = []
    for child in children:
        try:
            if isinstance(child, ctk.CTkTextbox) and child is not getattr(self, "textbox", None):
                self._install_one_secondary_textbox_hook(child)
        except Exception:
            pass
        walk(child)

walk(self.main_frame)

def _install_one_secondary_textbox_hook(self, widget):
    if getattr(widget, "_khgt_lang_hook_installed", False):
        return
    widget._khgt_lang_hook_installed = True
    try:
        raw_insert = widget.insert
        raw_delete = widget.delete
        widget._khgt_raw_insert = raw_insert
        widget._khgt_raw_delete = raw_delete
        widget._khgt_raw_content = widget.get("1.0", "end-1c")
    except Exception:
        return

def _delete(index1, index2=None, *args):
    try:
        idx1 = str(index1).lower()
        idx2 = str(index2).lower() if index2 is not None else None
        if idx1 in ("0.0", "1.0") and (
            idx2 in ("end", "end-1c", "tk.end") or index2 is None
        ):
            widget._khgt_raw_content = ""
    except Exception:
        pass
    if index2 is None:
        return raw_delete(index1, *args)
    return raw_delete(index1, index2, *args)

def _insert(index, content, *args):
    raw = content if isinstance(content, str) else str(content)
    try:
        idx = str(index).lower()
        if idx in ("0.0", "1.0"):
            widget._khgt_raw_content = raw

```

```

        else:
            widget._khgt_raw_content = getattr(widget, "_khgt_raw_content", "") + raw
    except Exception:
        pass
    rendered = self.translate_text_block(raw) if isinstance(content, str) else content
    return raw_insert(index, rendered, *args)

widget.delete = _delete
widget.insert = _insert

def refresh_secondary_output_languages(self):
    """Gambar ulang semua CTKTextbox output kanan dari teks mentahnya."""
    self.install_secondary_output_textbox_hooks()
    if not hasattr(self, "main_frame"):
        return

    def walk(widget):
        try:
            children = widget.winfo_children()
        except Exception:
            children = []
        for child in children:
            try:
                if isinstance(child, ctk.CTKTextbox) and child is not getattr(self, "textbox", None):
                    raw = getattr(child, "_khgt_raw_content", None)
                    if raw is None:
                        raw = child.get("1.0", "end-1c")
                        child._khgt_raw_content = raw
                    state = str(child.cget("state"))
                    child.configure(state="normal")
                    child._khgt_raw_delete("1.0", "end")
                    child._khgt_raw_insert("1.0", self.translate_text_block(raw))
                    child.configure(state=state)
            except Exception:
                pass
            walk(child)

    walk(self.main_frame)

def apply_language_direction(self):
    """Atur perataan dasar untuk bahasa Arab tanpa mengubah logika widget."""
    is_ar = getattr(self, "current_lang", "id") == "ar"
    anchor = "e" if is_ar else "w"
    justify = "right" if is_ar else "left"

    def walk(widget):
        try:
            children = widget.winfo_children()

```

```

except Exception:
    children = []
for child in children:
    try:
        if isinstance(
            child,
            (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)
        ):
            txt = str(child.cget("text") or "")
            if txt not in ("≡", "📁", "📷", "📄", "📊", "🏠", "|"):
                child.configure(anchor=anchor)
            if isinstance(child, ctk.CTkTextbox):
                try:
                    child._textbox.tag_configure(
                        "_khgt_lang_direction",
                        justify=justify
                    )
                    child._textbox.tag_add(
                        "_khgt_lang_direction", "1.0", "end"
                    )
                except Exception:
                    pass
            except Exception:
                pass
        walk(child)

    try:
        walk(self)
    except Exception:
        pass

def install_textbox_language_hooks(self):
    """Membungkus self.textbox.insert/delete agar semua output kanan
    otomatis mengikuti bahasa aktif tanpa harus mengedit 100+ titik insert.
    """
    if not hasattr(self, "textbox") or getattr(self, "_textbox_lang_hooks_installed", False):
        return
    self._textbox_lang_hooks_installed = True
    try:
        self._raw_textbox_insert = self.textbox.insert
        self._raw_textbox_delete = self.textbox.delete
        self._last_raw_textbox_content = self.textbox.get("1.0", "end-1c")
    except Exception:
        return

    def _delete(index1, index2=None, *args):
        try:

```



```

        idx1 = str(index1).lower()
        idx2 = str(index2).lower() if index2 is not None else None
        if idx1 == "1.0" and (idx2 in ("end", "end-1c", "tk.end") or index2 is None):
            self._last_raw_textbox_content = ""
            if not getattr(self, "_rendering_release_info", False):
                self.active_right_view = ""
        except Exception:
            pass
        if index2 is None:
            return self._raw_textbox_delete(index1, *args)
        return self._raw_textbox_delete(index1, index2, *args)

def _insert(index, text, *args):
    raw_text = text if isinstance(text, str) else str(text)
    try:
        idx = str(index).lower()
        if idx == "end":
            self._last_raw_textbox_content = getattr(self, "_last_raw_textbox_content", "") + raw_text
        elif idx == "1.0":
            self._last_raw_textbox_content = raw_text
        else:
            self._last_raw_textbox_content = getattr(self, "_last_raw_textbox_content", "") + raw_text
    except Exception:
        pass
    if isinstance(text, str):
        text = self.translate_text_block(text)
    return self._raw_textbox_insert(index, text, *args)

self.textbox.delete = _delete
self.textbox.insert = _insert

def capture_sidebar_original_texts(self):
    """Menyimpan teks asli sidebar kiri (bahasa Indonesia) untuk translasi stabil.
    Dipanggil setelah seluruh widget sidebar selesai dibuat. Penyimpanan ini
    tidak mengubah algoritma modul; hanya memberi acuan agar tombol bahasa
    dapat mengembalikan/menerjemahkan UI kiri dengan benar.
    """
    self._sidebar_original_text_widgets = {}
    self._sidebar_original_value_widgets = {}

def walk(widget):
    try:
        children = widget.winfo_children()
    except Exception:
        children = []
    for child in children:
        try:
            if isinstance(child, (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)):

```

```

curr_text = child.cget("text")
if curr_text:
    # Simpan teks Indonesia asli. reverse_tr dipakai agar aman
    # bila fungsi ini dipanggil ulang saat UI sudah EN/AR.
    id_text = self.reverse_tr(curr_text)
    setattr(child, "_khgt_id_text", id_text)
    self._sidebar_original_text_widgets[child] = id_text

if isinstance(child, (ctk.CTkOptionMenu, ctk.CTkComboBox)):
    if getattr(self, 'combo_lang', None) and child == self.combo_lang:
        setattr(child, "_khgt_lang_selector", True)
    else:
        vals = child.cget("values") or []
        id_vals = [self.reverse_tr(v) for v in vals]
        setattr(child, "_khgt_id_values", id_vals)
        self._sidebar_original_value_widgets[child] = id_vals
except Exception:
    pass
walk(child)

try:
    walk(self.sidebar)
except Exception:
    pass

def _translate_id_text_for_current_lang(self, id_text):
    """Terjemahkan teks ID secara aman untuk widget UI kiri."""
    if not id_text:
        return id_text
    return self.tr(id_text)

def force_update_sidebar_language(self):
    """Memaksa sidebar kiri mengikuti bahasa aktif setelah menu 1–41 dipilih.
    Fungsi ini tidak mengubah algoritma modul; hanya memperbarui teks widget
    kiri berdasarkan teks Indonesia asli + lang_cache_ui.json + fallback internal.
    """
    try:
        self.install_built_in_translations()
    except Exception:
        pass

# Jika belum pernah direkam, rekam sekarang dari struktur widget yang ada.
if not getattr(self, "_sidebar_original_text_widgets", None):
    try:
        self.capture_sidebar_original_texts()
    except Exception:
        pass

```

1) Paksa semua Label/Button/Switch/Radio di sidebar dari teks ID asli.

```
try:
    for widget, id_text in list(getattr(self, "_sidebar_original_text_widgets", {}).items()):
        try:
            if widget.winfo_exists():
                widget.configure(text=self._translate_id_text_for_current_lang(id_text))
        except Exception:
            pass
except Exception:
    pass
```

2) Paksa tombol menu 1–41 dari kunci Indonesia asli yang dipakai saat dibuat.

```
try:
    for id_text, btn in getattr(self, "menu_buttons", {}).items():
        try:
            btn.configure(text=self._translate_id_text_for_current_lang(id_text))
        except Exception:
            pass
except Exception:
    pass
```

3) Paksa OptionMenu/ComboBox sidebar dari daftar ID asli, kecuali selector bahasa.

```
try:
    for widget, id_values in list(getattr(self, "_sidebar_original_value_widgets", {}).items()):
        try:
            if not widget.winfo_exists():
                continue
            new_values = [self._translate_id_text_for_current_lang(v) for v in id_values]
            widget.configure(values=new_values)
            curr = widget.get()
            id_curr = self.reverse_tr(curr)
            if id_curr in id_values:
                widget.set(self._translate_id_text_for_current_lang(id_curr))
        except Exception:
            pass
except Exception:
    pass
```

4) Selector bahasa tidak boleh berubah menjadi teks terjemahan.

```
try:
    if hasattr(self, "combo_lang"):
        self.combo_lang.configure(values=["id", "en", "ar"])
        self.combo_lang.set(getattr(self, "current_lang", "id"))
except Exception:
    pass
```

```
def update_ui_texts(self, widget):
```

```
    """Melakukan rekursi pada seluruh elemen GUI dan menerjemahkannya."""
```

SOURCE CODE KHGT TIMES V8.0

```

self.install_builtin_translations()
for child in widget.winfo_children():
    try:
        # Terjemahkan Label, Tombol, Radio, dan Checkbox
        if isinstance(child, (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)):
            curr_text = child.cget("text")
            if curr_text:
                orig = self.reverse_tr(curr_text)
                new_t = self.tr(orig)
                if new_t != curr_text:
                    child.configure(text=new_t)

        # Terjemahkan OptionMenus & ComboBoxes
        if isinstance(child, (ctk.CTkOptionMenu, ctk.CTkComboBox)):
            # Jangan terjemahkan dropdown bahasa itu sendiri.
            if getattr(self, 'combo_lang', None) and child == self.combo_lang:
                continue
            curr_vals = child.cget("values")
            if curr_vals:
                new_vals = [self.tr(self.reverse_tr(v)) for v in curr_vals]
                child.configure(values=new_vals)
                curr_val = child.get()
                child.set(self.tr(self.reverse_tr(curr_val)))

        self.update_ui_texts(child)
    except Exception:
        pass

if __name__ == "__main__":
    app = KHGTApp()
    app.mainloop()

```

DAFTAR PUSTAKA DAN REFERENSI TEKNIS

Bishop, S. (n.d.). *pytz: World timezone definitions for Python*. Python Package Index. [Akses aktif — PyPI pytz](#)

Butar-Butar, A. J. R. (2022, August 31). *Dinamika para ahli tentang kriteria 3–6,4*. Observatorium Ilmu Falak Universitas Muhammadiyah Sumatera Utara. [Akses aktif — OIF UMSU](#)

Caldwell, J. A. R., & Laney, C. D. (2001). First visibility of the lunar crescent. *African Skies*, 5, 15–23. Referensi bibliografis karya ini juga tercantum dalam daftar literatur visibilitas hilal U.S. Naval Observatory. [Akses aktif — USNO Crescent Moon Visibility](#)

CustomTkinter. (n.d.). *CustomTkinter documentation*. [Akses aktif — Dokumentasi CustomTkinter](#)

Doggett, L. E., & Schaefer, B. E. (1994). Lunar crescent visibility. *Icarus*, 107(2), 388–403. <https://doi.org/10.1006/icar.1994.1031>. [DOI/Akses aktif](#)

Duffett-Smith, P., & Zwart, J. (2011). *Practical astronomy with your calculator or spreadsheet* (4th ed.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511861161>. [DOI/Akses aktif](#)

Espenak, F., & Meeus, J. (2006). *Five millennium canon of solar eclipses: –1999 to +3000 (2000 BCE to 3000 CE)* (NASA/TP-2006-214141). National Aeronautics and Space Administration, Goddard Space Flight Center. [Akses aktif — NASA Five Millennium Canon](#)

Espenak, F., & Meeus, J. (2009). *Five millennium canon of lunar eclipses: –1999 to +3000 (2000 BCE to 3000 CE)* (NASA/TP-2009-214172). National Aeronautics and Space Administration, Goddard Space Flight Center. [Akses aktif — NASA Lunar Eclipse Archive](#)

Fatoohi, L. J., Stephenson, F. R., & Al-Dargazelli, S. S. (1998). The Danjon limit of first visibility of the lunar crescent. *The Observatory*, 118, 65–72. [Referensi aktif — USNO](#)

fpdf2 Development Team. (n.d.). *fpdf2 documentation*. [Akses aktif — fpdf2](#)

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>. [DOI/Akses aktif](#)

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>. [DOI/Akses aktif](#)

Ilyas, M. (1984). *A modern guide to astronomical calculations of Islamic calendar, times & qibla*. Berita Publishing. [Akses bibliografis — Google Books](#)

Ilyas, M. (1994). Lunar crescent visibility criterion and Islamic calendar. *Quarterly Journal of the Royal Astronomical Society*, 35, 425–461. [Akses artikel — NASA ADS](#)

International Astronomical Center. (n.d.). *Scientific papers on crescent visibility and astronomical calculations*. [Akses aktif — International Astronomical Center](#)

Karim, M. F. R., & Mahsun. (2024). Kriteria baru MABIMS 3–6,4: Upaya penyatuan kalender Hijriah di Indonesia dalam perspektif *maqāsid al-syarī'ah*. *Astroislamica: Journal of Islamic Astronomy*, 3(1), 44–69. <https://doi.org/10.47766/astroislamica.v3i1.2735>. DOI/Akses aktif

Karttunen, H., Kröger, P., Oja, H., Poutanen, M., & Donner, K. J. (Eds.). (2017). *Fundamental astronomy* (6th ed.). Springer. <https://doi.org/10.1007/978-3-662-53045-0>. DOI/Akses aktif

Matplotlib Development Team. (n.d.). *Matplotlib documentation*. [Akses aktif — Matplotlib](#)

Meeus, J. (1998). *Astronomical algorithms* (2nd ed.). Willmann-Bell. [Akses bibliografis — Astronomical Algorithms](#)

Montenbruck, O., & Pfleger, T. (2000). *Astronomy on the personal computer* (4th ed.). Springer. <https://doi.org/10.1007/978-3-642-03436-7>. DOI/Akses aktif

Muhammadiyah. (2022, June 24). *Mengapa Muhammadiyah memakai kriteria Istanbul 2016 untuk kalender Islam global?* [Akses aktif — Muhammadiyah](#)

Muhammadiyah. (2024, July 8). *Alasan Muhammadiyah menerima Kalender Hijriah Global Tunggal Kriteria Turki 2016*. [Akses aktif — Muhammadiyah](#)

Muhammadiyah. (n.d.). *KHGT—Kalender Hijriah Global Tunggal*. [Akses aktif — Portal KHGT](#)

NASA Goddard Space Flight Center. (n.d.). *NASA eclipse web site: Lunar eclipses*. National Aeronautics and Space Administration. [Akses aktif — NASA Lunar Eclipses](#)

NASA Jet Propulsion Laboratory. (n.d.). *JPL planetary and lunar ephemerides*. Solar System Dynamics. [Akses aktif — JPL Ephemerides](#)

NumPy Developers. (n.d.). *NumPy documentation*. [Akses aktif — NumPy Documentation](#)

Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18, 39–64. <https://doi.org/10.1007/s10686-005-9002-5>. DOI/Akses aktif

Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105. <https://doi.org/10.3847/1538-3881/abd414>. DOI/Akses aktif

Pillow Contributors. (n.d.). *Pillow documentation*. [Akses aktif — Pillow Documentation](#)

Pygame Community. (n.d.). *Pygame documentation*. [Akses aktif — Pygame Documentation](#)

PyInstaller Development Team. (n.d.). *PyInstaller manual*. [Akses aktif — PyInstaller Manual](#)

Python Software Foundation. (n.d.). *Python 3 documentation*. [Akses aktif — Python Documentation](#)

Python Software Foundation. (n.d.). *tkinter—Python interface to Tcl/Tk*. In *Python 3 documentation*. [Akses aktif — Tkinter Documentation](#)

Reda, I., & Andreas, A. (2004). Solar position algorithm for solar radiation applications. *Solar Energy*, 76(5), 577–589. <https://doi.org/10.1016/j.solener.2003.12.003>. DOI/Akses aktif

Requests Development Team. (n.d.). *Requests: HTTP for humans*. [Akses aktif — Requests Documentation](#)

Rhodes, B. C. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites generator* (ASCL:1907.024). Astrophysics Source Code Library. [Akses aktif — Skyfield](#)

Rhodes, B. C. (n.d.). *PyEphem documentation*. [Akses aktif — PyEphem Documentation](#)

Rhodes, B. C. (n.d.). *Skyfield documentation*. [Akses aktif — Skyfield Documentation](#)

Richardson, L. (n.d.). *Beautiful Soup documentation*. [Akses aktif — Beautiful Soup 4](#)

Schaefer, B. E. (1988). Visibility of the lunar crescent. *Quarterly Journal of the Royal Astronomical Society*, 29, 511–523. [Akses aktif — NASA NTRS](#)

SciPy Community. (n.d.). *scipy.interpolate.griddata*. In *SciPy documentation*. [Akses aktif — SciPy griddata](#)

SciPy Community. (n.d.). *SciPy documentation*. [Akses aktif — SciPy Documentation](#)

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory supplement to the Astronomical Almanac* (3rd ed.). University Science Books. [Akses aktif — Explanatory Supplement](#)

U.S. Naval Observatory. (n.d.). *Crescent Moon visibility*. Astronomical Applications Department. [Akses aktif — USNO Crescent Moon Visibility](#)

U.S. Naval Observatory, & His Majesty's Nautical Almanac Office. (n.d.). *The Astronomical Almanac*. [Akses aktif — USNO Astronomical Almanac](#)

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K.

J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>. [DOI/Akses aktif](#)

Yallop, B. D. (1997). *A method for predicting the first sighting of the new crescent moon* (NAO Technical Note No. 69). HM Nautical Almanac Office. [Akses dokumen — NAO Technical Note 69](#)

Referensi Perangkat Lunak Tambahan yang Digunakan Source Code

Bora, A. (n.d.). *CTkToolTip*. Python Package Index. [Akses aktif — CTkToolTip](#)

DuckDuckGo Search Project. (n.d.). *duckduckgo-search*. Python Package Index. **Catatan:** paket `duckduckgo_search` yang digunakan pada source code V8.0 saat ini telah berganti nama menjadi `ddgs`. [Akses aktif — duckduckgo-search](#)

Python Software Foundation. (n.d.). *csv—CSV file reading and writing*. In *Python 3 documentation*. [Akses aktif — Python csv](#)

Python Software Foundation. (n.d.). *datetime—Basic date and time types*. In *Python 3 documentation*. [Akses aktif — Python datetime](#)

Python Software Foundation. (n.d.). *threading—Thread-based parallelism*. In *Python 3 documentation*. [Akses aktif — Python threading](#)

Python Software Foundation. (n.d.). *urllib—URL handling modules*. In *Python 3 documentation*. [Akses aktif — Python urllib](#)

Catatan Bibliografis

Daftar pustaka ini disusun sebagai **referensi ilmiah dan teknis yang mendukung konsep, algoritma, library, ephemeris, kriteria kalender, serta implementasi perangkat lunak KHGT Times V8.0**. Tidak semua sumber buku dan jurnal di atas dapat dinyatakan secara historis sebagai bahan yang secara langsung dibaca penulis ketika setiap baris kode dibuat, karena source code tidak memuat catatan sitasi per fungsi. Oleh sebab itu, istilah yang paling tepat untuk bagian buku adalah **“Daftar Pustaka dan Referensi Teknis”**.

Untuk komponen astronomi presisi V8.0, rujukan primer yang sangat penting adalah **Park, Folkner, Williams, dan Boggs (2021)** mengenai ephemeris JPL DE440/DE441; **Urban dan Seidelmann (2013)** untuk astrometri dan standar *Astronomical Almanac*; **Meeus (1998)** untuk algoritma astronomi; serta dokumentasi **Skyfield** untuk implementasi komputasinya.

Untuk kajian visibilitas hilal, sumber utama meliputi **Ilyas (1994)**, **Schaefer (1988)**, **Doggett dan Schaefer (1994)**, **Yallop (1997)**, serta **Odeh (2004)**.

Untuk model Kalender Hijriah Global Tunggal yang diimplementasikan di KHGT Times, rujukan institusional utama adalah dokumentasi **Kriteria Turki/Istanbul 2016** dan **Portal KHGT Muhammadiyah**, khususnya parameter tinggi hilal minimal 5° dan elongasi minimal 8° dalam satu kesatuan matlak global.

Untuk komparasi dengan kalender regional, literatur **Neo-MABIMS 3° – $6,4^\circ$** digunakan sebagai sumber metodologis mengenai tinggi hilal minimum 3° dan elongasi minimum $6,4^\circ$.

Semua dokumentasi perangkat lunak dicantumkan untuk menjamin **reproducibility**, keterlacakan dependency, dan kemudahan pemeliharaan source code KHGT Times V8.0 pada pengembangan berikutnya.

GLOSARIUM

KHGT TIMES V8.0

Aberasi Cahaya (*Aberration of Light*)

Perubahan semu arah datang cahaya benda langit akibat kombinasi gerak pengamat dan kecepatan cahaya. Dalam dokumentasi KHGT Times, koreksi aberasi merupakan salah satu koreksi astronomis yang ditangani dalam komputasi berbasis Skyfield.

Admin

Pengguna dengan hak akses khusus terhadap fungsi administratif aplikasi. Dalam KHGT Times V8.0, Menu 41 **HIJRI_DB Auto-Builder (Admin)** merupakan fasilitas yang dilindungi otorisasi untuk pengelolaan atau pembangunan data kalender Hijriah.

AI (*Artificial Intelligence*)

Kecerdasan buatan. Pada KHGT Times V8.0 diterapkan melalui modul **WinAI**, yang dirancang untuk menerima pertanyaan pengguna, mendeteksi maksud (*intent*), serta mengarahkan permintaan tertentu ke modul perhitungan yang sesuai.

Altitude / Ketinggian Benda Langit

Sudut vertikal suatu benda langit terhadap horizon. Nilai 0° menunjukkan benda berada pada horizon, nilai positif menunjukkan benda berada di atas horizon, dan nilai negatif menunjukkan benda berada di bawah horizon.

Altitude Chart

Grafik perubahan ketinggian benda langit terhadap waktu. KHGT Times menyediakan **Altitude Chart Analyser** untuk menampilkan dan menganalisis posisi Matahari, Bulan, dan parameter astronomis lain selama suatu interval waktu.

Altitude Chart Analyser

Modul KHGT Times untuk menghitung dan menampilkan grafik ketinggian benda langit. Kode melakukan pengambilan data ephemeris selama 24 jam dan membentuk deret waktu dengan NumPy untuk divisualisasikan.

Analemma Matahari

Kurva berbentuk menyerupai angka delapan yang menggambarkan perubahan posisi semu Matahari apabila diamati pada waktu yang sama sepanjang tahun. KHGT Times menyediakan **Generator Analemma Matahari** sebagai salah satu modul astronomi.

Analisis Gerhana

Modul untuk melakukan pencarian dan penyajian kejadian gerhana dalam rentang waktu yang dipilih. Hasilnya dapat disajikan dalam struktur tabel dan diekspor oleh aplikasi.

Apparent Position / Posisi Tampak

Posisi benda langit setelah koreksi-koreksi astronomis yang relevan diterapkan. Dalam Skyfield,

kode KHGT Times menggunakan proses `.observe(...).apparent()` sebelum sejumlah parameter posisi dihitung.

Array

Struktur data numerik multidimensi. Dalam KHGT Times, array NumPy digunakan untuk mengolah kumpulan nilai astronomis, deret waktu, dan perhitungan dalam jumlah besar.

Astrofotografi

Teknik fotografi benda atau fenomena langit. Pada KHGT Times terdapat modul **Astrofotografi & Kontras Visibilitas**, termasuk fungsi yang berkaitan dengan kondisi visibilitas dan fenomena okultasi.

Astrometri

Cabang astronomi yang berhubungan dengan pengukuran posisi dan gerak benda langit. KHGT Times menggunakan Skyfield dan ephemeris JPL sebagai komponen penting engine astrometrinya.

Auto-Builder

Fasilitas yang secara otomatis membangun atau membantu menghasilkan struktur data tertentu. Dalam KHGT Times, istilah ini secara khusus digunakan pada **HIJRI_DB Auto-Builder**.

Auto-Switch Ephemeris

Mekanisme V8.0 yang secara otomatis memilih kernel ephemeris menurut tahun perhitungan. Sistem menggunakan `de440s.bsp` untuk tahun 1850–2149 dan `de441-new.bsp` untuk tahun di luar rentang tersebut sampai batas aplikasi.

Azimut (*Azimuth*)

Sudut arah horizontal suatu objek, diukur terhadap arah acuan pada horizon. Dalam sejumlah keluaran KHGT Times, azimut dinyatakan dalam derajat dan digunakan antara lain pada perhitungan posisi Matahari, Bulan, dan arah kiblat.

Beautiful Soup / Bs4

Library Python untuk membaca dan memproses struktur HTML. Source code mengimpor `BeautifulSoup` dari paket `bs4` sebagai salah satu komponen fungsi jaringan dan pemrosesan konten web.

BSP (*Binary SPK*)

Format file kernel ephemeris yang digunakan sistem SPICE/JPL untuk menyimpan data posisi benda tata surya. KHGT Times V8.0 menggunakan file `de440s.bsp` dan `de441-new.bsp`.

Bulan Hijriah

Satuan bulan dalam kalender Hijriah, terdiri atas Muharam, Safar, Rabiulawal, Rabiulakhir, Jumadilawal, Jumadilakhir, Rajab, Syakban, Ramadan, Syawal, Zulkaidah, dan Zulhijah.

Crescent

Bulan sabit. Dalam konteks KHGT Times umumnya merujuk kepada hilal, yaitu fase awal Bulan yang menjadi objek penting dalam penentuan awal bulan Hijriah.

Crescent Visibility

Kemungkinan atau kondisi keterlihatan hilal berdasarkan parameter astronomis. KHGT Times mempunyai sejumlah modul untuk kalkulasi, analisis global, status kriteria, serta pemetaan visibilitas hilal.

Crescent Visibility Map

Peta visual yang menggambarkan kondisi visibilitas hilal menurut wilayah geografis. KHGT Times menggunakan pengolahan data dan interpolasi untuk membentuk visualisasi peta tersebut.

CSV (*Comma-Separated Values*)

Format data teks tabular. KHGT Times menyediakan fungsi ekspor CSV agar hasil perhitungan dapat diolah lebih lanjut pada aplikasi spreadsheet.

CustomTkinter

Framework GUI berbasis Tkinter yang digunakan KHGT Times untuk membangun antarmuka desktop modern, termasuk *dark mode*, sidebar, frame, tombol, dan berbagai komponen input.

Dark Mode

Mode tampilan antarmuka dengan warna latar dominan gelap. Source code V8.0 menetapkan tampilan CustomTkinter pada mode gelap.

DE (*Development Ephemeris*)

Seri ephemeris numerik yang dikembangkan Jet Propulsion Laboratory untuk menentukan posisi dan gerak benda-benda tata surya.

DE440s

Versi ringkas dari JPL DE440. Dalam KHGT Times V8.0 file `de440s.bsp` digunakan secara otomatis untuk tahun penuh **1850–2149**.

DE441

Ephemeris JPL dengan rentang waktu sangat panjang. KHGT Times menggunakan file `de441-new.bsp` untuk tahun **1–1849** dan **2150–2999** dalam batas rentang yang ditetapkan aplikasi.

Deklinasi (*Declination* / *Dec*)

Koordinat sudut benda langit pada sistem koordinat ekuatorial yang menyatakan posisi utara atau selatan terhadap ekuator langit.

Dip Horizon

Penurunan geometris horizon akibat ketinggian pengamat di atas permukaan acuan. Dalam beberapa kalkulasi kode, koreksi ketinggian pengamat diperhitungkan untuk menentukan kondisi horizon.

Earth / Bumi

Objek pusat pengamatan utama dalam banyak perhitungan KHGT Times. Pada kernel Skyfield, objek `earth` digunakan untuk membentuk pengamat geosentris maupun toposentris.

Elevasi Pengamat

Ketinggian lokasi pengamat dari permukaan acuan, umumnya dinyatakan dalam meter. Parameter ini digunakan bersama lintang dan bujur ketika membuat lokasi dengan WGS84.

Elongasi (*Elongation*)

Sudut pisah antara Matahari dan Bulan sebagaimana terlihat dari titik acuan tertentu. Parameter ini merupakan salah satu variabel utama analisis hilal.

Elongasi Geosentris

Sudut pisah Matahari–Bulan yang dihitung dengan titik pengamatan dari pusat Bumi.

Elongasi Toposentris

Sudut pisah Matahari–Bulan yang dihitung dari lokasi tertentu di permukaan Bumi.

Engine

Bagian inti sistem yang melakukan proses kalkulasi. Istilah **KHGT Engine** digunakan pada antarmuka untuk kumpulan fungsi dan modul perhitungan astronomi.

Ephemeris

Data atau hasil perhitungan yang menyatakan posisi benda langit pada waktu tertentu. Dalam KHGT Times, ephemeris berperan sebagai dasar utama berbagai perhitungan posisi Matahari, Bulan, dan benda langit lainnya.

Ephemeris 3D

Representasi tiga dimensi posisi atau konfigurasi benda langit berdasarkan data ephemeris. KHGT Times menyediakan **Simulasi Ephemeris 3D**.

Equinox / Ekuinoks

Peristiwa ketika Matahari berada di sekitar perpotongan ekliptika dan ekuator langit, sehingga deklinasi Matahari mendekati 0° . KHGT Times menyediakan modul **Equinox & Solstice**.

Export / Ekspor

Proses menyimpan hasil perhitungan aplikasi ke format eksternal. KHGT Times mendukung beberapa keluaran seperti TXT, CSV, PNG, dan PDF sesuai modulnya.

Fajar Shadiq

Fajar yang digunakan sebagai acuan awal waktu Subuh. KHGT Times memiliki modul **Evaluasi Fajar Shadiq & SQM**.

Fase Bulan (*Moon Phase*)

Perubahan bentuk bagian Bulan yang tampak terang dari Bumi akibat perubahan konfigurasi Matahari–Bumi–Bulan.

FPDF

Library pembentukan dokumen PDF yang digunakan dalam sebagian fungsi ekspor laporan KHGT Times.

Geocentric / Geosentris

Sistem perhitungan dengan pusat Bumi sebagai titik acuan. Dalam KHGT Times, parameter geosentris dibedakan dari parameter yang dihitung berdasarkan posisi pengamat di permukaan Bumi.

GHA/GAST

Dalam konteks kode astronomi, GAST (*Greenwich Apparent Sidereal Time*) digunakan dalam proses transformasi antara koordinat ekuatorial dan koordinat yang berkaitan dengan posisi pengamat.

Global Crescent Visibility Analyzer

Fungsi analisis yang mengevaluasi kondisi hilal dalam lingkup geografis luas sebagai bagian dari sistem analisis kalender global.

GPS / Geolocation

Teknologi penentuan lokasi pengguna. KHGT Times menyediakan fasilitas lokasi untuk memperoleh atau membantu menentukan koordinat pengamat.

GUI (*Graphical User Interface*)

Antarmuka grafis yang memungkinkan pengguna berinteraksi dengan program melalui tombol, menu, formulir, grafik, dan panel, bukan hanya melalui terminal.

Heatmap

Peta visual yang menggunakan perubahan tampilan grafis untuk menunjukkan variasi nilai pada area tertentu. Dalam dokumentasi source code, SciPy digunakan membantu pembentukan peta visibilitas yang lebih mulus.

HIJRI_DB

Struktur basis data internal kalender Hijriah pada KHGT Times. Data ini memuat tahun Hijriah, nama bulan, tanggal awal bulan, jumlah hari, dan informasi terkait yang digunakan oleh modul kalender dan konversi.

HIJRI_DB Auto-Builder

Menu administratif untuk membantu membangun atau menghasilkan data **HIJRI_DB** melalui proses perhitungan aplikasi.

Hilal

Bulan sabit muda yang berkaitan dengan awal bulan Hijriah. Dalam KHGT Times, hilal dianalisis menggunakan parameter seperti tinggi, elongasi, iluminasi, umur Bulan, lokasi, dan waktu pengamatan.

Horizon / Ufuk

Batas semu antara langit dan permukaan Bumi bagi pengamat. Ketinggian benda langit dihitung relatif terhadap horizon.

Ijtimak

Konjungsi Matahari dan Bulan yang menandai keadaan astronomis ketika keduanya memiliki posisi arah yang sangat berdekatan. Dalam source code, waktu ijtimak digunakan dalam berbagai analisis awal bulan Hijriah.

Illuminasi Bulan

Persentase bagian permukaan Bulan yang tampak diterangi dari arah pengamat. Kode menggunakan fungsi Skyfield `fraction_illuminated()` dalam sejumlah kalkulasi.

Imkanur Rukyat

Konsep kemungkinan terlihatnya hilal berdasarkan kriteria astronomis tertentu. KHGT Times mengimplementasikan dan membandingkan lebih dari satu kerangka kriteria sesuai modul analisis kalender.

Intent

Maksud atau tujuan suatu perintah pengguna. WinAI menggunakan mekanisme deteksi *intent* untuk menentukan apakah permintaan pengguna berkaitan dengan hilal, gerhana, pasang surut, fajar, animasi, dan fungsi lain.

Interpolasi

Teknik memperkirakan nilai pada lokasi di antara titik-titik data yang diketahui. Source code menggunakan fasilitas SciPy untuk mendukung interpolasi pada visualisasi peta.

Istikmal

Penggenapan jumlah hari dalam bulan Hijriah menjadi 30 hari apabila kondisi atau kriteria yang digunakan belum terpenuhi. Istilah ini digunakan pada keluaran komparasi KHGT dan Neo-MABIMS.

JPL (*Jet Propulsion Laboratory*)

Lembaga yang menghasilkan Development Ephemeris yang digunakan sebagai basis data posisi benda tata surya oleh KHGT Times.

Kalender Hijriah

Kalender yang didasarkan pada siklus sinodik Bulan dan terdiri atas 12 bulan Hijriah.

Kalender Hijriah Global Tunggal

Lihat KHGT.

Kalender Masehi

Kalender Gregorian yang digunakan untuk representasi tanggal sipil dan sebagai salah satu sistem tanggal dalam fungsi konversi KHGT Times.

Kesatuan Matlak (*Ittihad al-Matali'*)

Prinsip yang dalam dokumentasi KHGT dipahami sebagai penggunaan Bumi sebagai satu kesatuan wilayah kalender. Dalam source code, konsep ini menjadi bagian dari kerangka penentuan kalender global.

KHGT

Singkatan dari **Kalender Hijriah Global Tunggal**, yaitu kerangka kalender Hijriah yang diarahkan pada penggunaan satu sistem kalender global.

KHGT Engine

Sebutan untuk sistem perhitungan dan kumpulan modul utama yang menjalankan fungsi astronomi, kalender, pemetaan, dan analisis KHGT Times.

Kriteria 5°–8°

Parameter yang digunakan source code dalam analisis KHGT: tinggi hilal geosentris minimal 5° dan elongasi geosentris minimal 8°.

Kriteria 3°–6,4°

Parameter yang digunakan source code dalam komparasi Neo-MABIMS: tinggi hilal minimal 3° dan elongasi minimal 6,4° pada skenario analisis yang diimplementasikan aplikasi.

Latitude / Lintang

Koordinat geografis yang menyatakan posisi utara atau selatan suatu titik terhadap ekuator Bumi.

Linspace

Fungsi NumPy untuk menghasilkan sejumlah nilai dengan interval teratur dalam suatu rentang. Digunakan dalam source code antara lain untuk membentuk sampling waktu.

Live Animasi

Modul visualisasi dinamis posisi benda langit. Pengembangan modul ini mencakup visualisasi Matahari, Bulan, planet, rasi bintang, dan fasilitas pelacakan.

Longitude / Bujur

Koordinat geografis yang menyatakan posisi timur atau barat terhadap meridian acuan.

Mainland

Daratan utama. Dalam konteks modul **Kota Pertama KHGT (Mainland)**, istilah ini berkaitan dengan pencarian lokasi daratan yang pertama memenuhi parameter tertentu.

MABIMS

Forum kerja sama keagamaan Brunei Darussalam, Indonesia, Malaysia, dan Singapura. Dalam KHGT Times istilah ini muncul terutama pada komparasi kriteria penentuan awal bulan dengan KHGT.

Matplotlib

Library visualisasi Python yang digunakan KHGT Times untuk membangun grafik astronomi, visualisasi spasial 3D, *Altitude Chart*, serta sejumlah instrumen grafis lainnya.

Mizwala

Instrumen atau konsep yang menggunakan bayangan Matahari untuk kebutuhan pengukuran arah/waktu. KHGT Times menyediakan **Kalkulator Mizwala (Bayangan)**.

Moon / Bulan

Satelit alami Bumi dan salah satu objek utama dalam kalkulasi KHGT Times, terutama pada kalender Hijriah, fase Bulan, hilal, gerhana, pasang surut, serta ephemeris Matahari–Bulan.

Moon Times

Modul yang berkaitan dengan waktu fenomena harian Bulan, termasuk terbit, transit, dan terbenam.

Moonphase

Fase Bulan sebagai akibat perubahan geometri Matahari–Bumi–Bulan. KHGT Times menyediakan modul khusus data siklus fase Bulan.

NASA

National Aeronautics and Space Administration. Dalam konteks dokumentasi aplikasi, istilah NASA/JPL berkaitan dengan sumber ephemeris numerik yang menjadi basis komputasi.

Neo-MABIMS

Kriteria baru MABIMS yang dalam implementasi komparasi source code menggunakan batas tinggi 3° dan elongasi $6,4^\circ$.

NumPy

Library komputasi numerik Python. Digunakan KHGT Times untuk array, manipulasi data numerik, pembentukan deret waktu, dan berbagai kalkulasi berbasis matriks.

Nutasi (*Nutation*)

Variasi periodik kecil orientasi sumbu rotasi Bumi. Disebut dalam dokumentasi engine astrometri sebagai salah satu koreksi posisi astronomis.

Observer / Pengamat

Titik referensi tempat posisi benda langit dihitung. Pengamat dapat berupa pusat Bumi atau lokasi tertentu di permukaan Bumi.

Okultasi (*Occultation*)

Fenomena ketika satu benda langit menutupi benda langit lainnya dari sudut pandang pengamat. Source code menghubungkan fungsi prediktor okultasi dengan modul astrofotografi dan visibilitas.

Peta Langit Dunia

Menu 39 pada KHGT Times V8.0 yang membuka layanan peta langit berbasis web untuk eksplorasi posisi benda langit.

Pasang Surut Gravitasi

Perubahan muka air yang berkaitan terutama dengan pengaruh gravitasi Bulan dan Matahari. KHGT Times menyediakan modul **Prediktor Pasang Surut Gravitasi**.

PDF (*Portable Document Format*)

Format dokumen yang digunakan sebagai salah satu bentuk keluaran atau laporan aplikasi.

Pillow / PIL

Library pemrosesan citra Python yang digunakan KHGT Times untuk membuat, menggambar, menyimpan, dan memanipulasi gambar kalender, jadwal salat, serta elemen visual lainnya.

PKG

Dalam dokumentasi aplikasi digunakan sebagai singkatan **Parameter Kalender Global**.

PKG 1

Mekanisme kondisi normal pada implementasi KHGT di source code: sistem memindai terpenuhinya kriteria tinggi hilal dan elongasi pada daratan sebelum batas waktu yang ditentukan dalam algoritma.

PKG 2

Mekanisme kondisi kritis yang digunakan source code apabila pemenuhan kriteria terjadi dalam kondisi waktu tertentu, dengan pemeriksaan tambahan terhadap waktu ijtimaq dan fajar pada lokasi referensi.

Planetary Times

Modul KHGT Times yang berkaitan dengan informasi waktu atau parameter posisi planet sesuai rancangan aplikasi.

PNG (*Portable Network Graphics*)

Format citra yang digunakan untuk menyimpan sejumlah grafik, kalender, atau visualisasi hasil perhitungan.

Prayer Times

Modul perhitungan jadwal salat. Output kode antara lain memuat Subuh, terbit Matahari, Duha, Zuhur, Asar, Magrib, Isya, dan informasi tengah malam.

Presesi (*Precession*)

Perubahan jangka panjang arah sumbu rotasi Bumi. Dalam dokumentasi engine, presesi merupakan salah satu koreksi yang diperhitungkan oleh sistem astronomi.

PyEphem / ephemeris

Library astronomi Python yang digunakan pada sejumlah kalkulasi dan iterasi, seperti pencarian ijtimak, Matahari terbenam, serta pemrosesan berulang pada banyak lokasi.

Python

Bahasa pemrograman utama yang digunakan untuk membangun KHGT Times V8.0.

Qiblah / Kiblat

Arah menuju Ka'bah dari lokasi pengamat. KHGT Times menyediakan modul perhitungan arah dan waktu terkait kiblat.

Qiblah Direction & Times

Modul yang menggabungkan kalkulasi arah kiblat dengan informasi waktu yang berkaitan dengan posisi Matahari.

Qibla Time / Rashdul Qiblah Lokal

Waktu ketika arah Matahari atau arah bayangan dapat digunakan untuk membantu menentukan arah kiblat pada lokasi tertentu.

RA / Right Ascension / Asensio Rekta

Koordinat ekuatorial benda langit yang berfungsi sebanding dengan bujur pada bola langit dan biasanya dinyatakan dalam jam, menit, dan detik.

Real-Time

Pemrosesan atau visualisasi yang diperbarui menurut waktu berjalan atau keadaan simulasi yang sedang berlangsung.

Refraksi Atmosfer

Pembelokan cahaya benda langit akibat atmosfer Bumi sehingga posisi tampak berbeda dari posisi geometrisnya. Parameter suhu dan tekanan dapat digunakan pada fungsi altitude–azimuth untuk koreksi ini.

Requests

Library HTTP Python yang digunakan source code untuk melakukan komunikasi dengan sumber atau layanan berbasis web.

Rukyat

Pengamatan hilal. Dalam aplikasi istilah ini terutama berkaitan dengan evaluasi posisi dan kondisi hilal menjelang awal bulan Hijriah.

SciPy

Library komputasi ilmiah Python. KHGT Times memanfaatkan komponen interpolasi SciPy untuk membantu pemrosesan dan visualisasi peta.

Sidebar

Panel navigasi pada antarmuka aplikasi yang berisi akses ke menu/modul KHGT Times.

Skyfield

Library astronomi Python yang menjadi salah satu komponen utama engine KHGT Times. Digunakan untuk menangani ephemeris JPL, waktu astronomi, lokasi WGS84, posisi geosentris/toposentris, almanak, gerhana, dan berbagai transformasi astronomis.

Solstice / Solstis

Peristiwa ketika Matahari mencapai deklinasi utara atau selatan maksimum dalam siklus tahunan. Dianalisis bersama ekuinoks pada modul **Equinox & Solstice**.

SPK

Spacecraft and Planet Kernel, struktur kernel SPICE yang digunakan untuk menyimpan informasi posisi benda langit; file JPL berekstensi `.bsp` yang digunakan aplikasi termasuk dalam keluarga data ini.

SQM (Sky Quality Meter)

Konsep/instrumen pengukuran tingkat kecerahan atau kegelapan langit. Dalam aplikasi digunakan pada modul **Evaluasi Fajar Shadiq & SQM**.

Stellarium

Perangkat lunak planetarium yang digunakan sebagai aplikasi pembanding/pendamping pada fungsi sinkronisasi dalam pengembangan Live Animasi KHGT Times.

Sun / Matahari

Bintang pusat Tata Surya dan salah satu objek utama ephemeris KHGT Times. Posisi Matahari diperlukan untuk jadwal salat, hilal, kiblat, analemma, ekuinoks, solstis, gerhana, dan sejumlah modul lainnya.

Sun Moon Earth System

Representasi hubungan geometris Matahari–Bulan–Bumi dalam salah satu modul KHGT Times.

Sun Moon Ephemeris

Modul penyajian data posisi dan parameter astronomis Matahari dan Bulan.

Sun Times

Modul yang berkaitan dengan waktu peristiwa harian Matahari, seperti terbit, transit, dan terbenam.

Sunrise

Waktu ketika Matahari terbit melewati horizon menurut definisi yang digunakan dalam perhitungan.

Sunset

Waktu Matahari terbenam. Pada berbagai modul hilal, sunset menjadi waktu referensi penting untuk mengevaluasi posisi Bulan.

Thread / Threading

Teknik menjalankan pekerjaan tertentu tanpa membekukan antarmuka utama. KHGT Times menggunakan Python `threading` untuk menjalankan kalkulasi yang membutuhkan proses lebih panjang.

Timezone / Zona Waktu

Selisih waktu lokal terhadap UTC. Parameter ini digunakan untuk mengonversi waktu perhitungan astronomis ke waktu lokal.

Topocentric / Toposentris

Sistem koordinat atau perhitungan dengan lokasi pengamat pada permukaan Bumi sebagai titik acuan. Source code menggunakan `wgs84.latlon()` untuk membentuk lokasi pengamat dan menghitung posisi tampak dari lokasi tersebut.

Transit

Saat benda langit melintasi meridian pengamat dan umumnya mencapai posisi tertinggi hariannya.

Twilight / Senja-Fajar Astronomis

Keadaan pencahayaan langit ketika Matahari berada di bawah horizon. Dalam aplikasi, parameter twilight berkaitan antara lain dengan jadwal salat dan evaluasi fajar.

TXT

Format teks sederhana yang digunakan sebagai salah satu format penyimpanan laporan atau hasil perhitungan.

UTC (*Coordinated Universal Time*)

Standar waktu internasional yang menjadi acuan utama untuk berbagai perhitungan astronomi dan kemudian dikonversi ke zona waktu lokal.

Visibility / Visibilitas

Tingkat atau kemungkinan suatu objek dapat terlihat. Dalam KHGT Times istilah ini terutama digunakan untuk hilal dan astrofotografi.

WGS84 (*World Geodetic System 1984*)

Sistem referensi geodesi global yang digunakan Skyfield untuk mendefinisikan lokasi pengamat berdasarkan lintang, bujur, dan elevasi.

WinAI

Modul asisten kecerdasan buatan pada KHGT Times. Source code mengembangkan WinAI sebagai *interceptor* yang dapat mengidentifikasi intent pengguna dan, untuk sejumlah permintaan, mengarahkan aplikasi ke modul perhitungan yang bersesuaian.

World Sky Map

Nama bahasa Inggris untuk **Peta Langit Dunia**, yaitu layanan peta langit daring yang terintegrasi melalui Menu 39.

XEphem

Perangkat lunak astronomi yang berkaitan secara historis dengan engine yang digunakan PyEphem. Dokumentasi internal KHGT Times menyebut PyEphem sebagai engine berbasis C turunan XEphem.

Zuhur / Solar Transit Reference

Dalam konteks jadwal salat, Zuhur berkaitan dengan saat Matahari telah melewati meridian lokal. Source code jadwal salat menampilkan kolom Zuhur dengan keterangan transit.

Δ / Delta

Simbol yang secara umum menunjukkan selisih atau perubahan suatu parameter. Dalam data astronomis dapat digunakan untuk menyatakan perbedaan nilai antarposisi, waktu, atau kondisi, sesuai konteks keluaran.

SINGKATAN PENTING

Bagian ini memuat singkatan, akronim, dan notasi teknis yang digunakan dalam **KHGT Times V8.0**. Penjelasan diberikan sesuai konteks penggunaannya dalam source code, antarmuka, keluaran perhitungan, dokumentasi teknis, dan modul astronomi aplikasi.

Singkatan	Kepanjangan / Istilah	Makna dalam KHGT Times V8.0
3D	<i>Three-Dimensional</i>	Representasi tiga dimensi. Digunakan antara lain pada Simulasi Ephemeris 3D dan visualisasi spasial benda langit.
AI	<i>Artificial Intelligence</i>	Kecerdasan buatan. Digunakan pada WinAI , asisten aplikasi yang menerima pertanyaan dan dapat mengarahkan permintaan pengguna menuju modul tertentu.
API	<i>Application Programming Interface</i>	Antarmuka komunikasi antarperangkat lunak atau layanan daring. Digunakan pada bagian aplikasi yang berkomunikasi dengan layanan eksternal.
AR	<i>Arabic</i>	Kode bahasa Arab pada sistem antarmuka multibahasa.
BCE	<i>Before Common Era</i>	Notasi tahun sebelum era Masehi yang relevan dengan fungsi penanganan tahun astronomis atau tahun negatif.
BS4	<i>Beautiful Soup 4</i>	Library Python untuk pemrosesan dan pembacaan struktur HTML.
BSP	<i>Binary SPK</i>	Format kernel ephemeris JPL/SPICE. KHGT Times V8.0 menggunakan <code>de440s.bsp</code> dan <code>de441-new.bsp</code> .
CE	<i>Common Era</i>	Notasi tahun era Masehi pada sejumlah representasi tahun astronomis.
CSV	<i>Comma-Separated Values</i>	Format data tabular untuk ekspor hasil kalkulasi agar dapat diolah lebih lanjut menggunakan aplikasi spreadsheet.
CTk	<i>CustomTkinter</i>	Framework GUI modern berbasis Tkinter yang digunakan untuk membangun antarmuka KHGT Times.
DE	<i>Development Ephemeris</i>	Seri ephemeris numerik JPL untuk menentukan posisi benda tata surya.
DE440s	<i>Development Ephemeris 440 short</i>	Kernel ringkas ephemeris yang digunakan otomatis oleh V8.0 untuk tahun 1850–2149.
DE441	<i>Development Ephemeris 441</i>	Ephemeris berjangka panjang yang digunakan KHGT Times untuk tahun di luar rentang DE440s hingga batas aplikasi 1–2999 M.

Dec	<i>Declination</i>	Deklinasi, yaitu koordinat ekuatorial yang menyatakan posisi benda langit di utara atau selatan ekuator langit.
EN	<i>English</i>	Kode bahasa Inggris pada antarmuka multibahasa.
Eln	<i>Elongation</i>	Elongasi atau sudut pisah Matahari–Bulan. Dalam berbagai laporan kode dapat muncul sebagai Eln, elong, atau elongation.
EXE	<i>Executable File</i>	Berkas aplikasi yang dapat dijalankan pada Windows. Source code memiliki beberapa penyesuaian khusus agar stabil setelah dikemas menjadi EXE.
GAST	<i>Greenwich Apparent Sidereal Time</i>	Waktu sideris tampak Greenwich yang digunakan dalam transformasi koordinat astronomis.
GPS	<i>Global Positioning System</i>	Istilah yang digunakan pada fasilitas penentuan atau pelacakan lokasi pengamat.
GUI	<i>Graphical User Interface</i>	Antarmuka grafis aplikasi yang terdiri atas sidebar, tombol, kotak input, grafik, tabel, dan area keluaran.
H	Hijriah	Penanda tahun kalender Hijriah, misalnya 1447 H.
HD	<i>High Definition</i>	Resolusi tinggi; digunakan dalam konteks visualisasi atau ekspor peta/gambar beresolusi tinggi.
HTML	<i>HyperText Markup Language</i>	Bahasa markup halaman web yang dapat diproses oleh komponen pencarian dan pemrosesan konten web aplikasi.
HTTP/HTTPS	<i>HyperText Transfer Protocol / Secure</i>	Protokol komunikasi web yang digunakan ketika aplikasi mengakses layanan daring.
ID	Indonesia	Kode bahasa Indonesia, sekaligus bahasa bawaan antarmuka KHGT Times V8.0.
JPL	<i>Jet Propulsion Laboratory</i>	Lembaga pengembang Development Ephemeris yang menjadi basis data astronomis de440s.bsp dan de441-new.bsp.
JSON	<i>JavaScript Object Notation</i>	Format data yang digunakan antara lain untuk cache terjemahan dan basis data pendukung. HIJRI_DB Auto-Builder juga berkaitan dengan pembuatan data JSON.
KHGT	Kalender Hijriah Global Tunggal	Sistem kalender Hijriah global yang menjadi salah satu fokus utama aplikasi KHGT Times.
Lat	<i>Latitude</i>	Lintang geografis lokasi pengamat.
Lon	<i>Longitude</i>	Bujur geografis lokasi pengamat.
LT	<i>Local Time</i>	Waktu lokal lokasi pengamat. Pada beberapa laporan kode digunakan sebagai pembeda terhadap UTC.

M	Masehi	Penanda tahun Masehi dalam format keluaran aplikasi.
MABIMS	Menteri Agama Brunei Darussalam, Indonesia, Malaysia, dan Singapura	Istilah yang digunakan pada modul komparasi kalender, terutama untuk membandingkan implementasi KHGT dengan kriteria Neo-MABIMS.
NASA	<i>National Aeronautics and Space Administration</i>	Nama lembaga antariksa Amerika Serikat yang dalam dokumentasi aplikasi sering muncul bersama JPL pada penyebutan sumber ephemeris.
N/A	<i>Not Available / Not Applicable</i>	Menandakan data tidak tersedia, tidak dapat dihitung, atau tidak berlaku pada suatu kondisi.
PDF	<i>Portable Document Format</i>	Salah satu format ekspor laporan dan kalender.
PIL	<i>Python Imaging Library</i>	Nama historis library pengolahan gambar yang saat ini digunakan melalui paket Pillow.
PKG	Parameter Kalender Global	Istilah internal dokumentasi KHGT Times untuk mekanisme evaluasi kalender global, khususnya PKG 1 dan PKG 2 .
PKG 1	Parameter Kalender Global 1	Jalur evaluasi utama ketika kriteria KHGT telah terpenuhi sebelum batas waktu yang ditetapkan algoritma.
PKG 2	Parameter Kalender Global 2	Jalur evaluasi tambahan ketika kondisi waktu tertentu memerlukan pemeriksaan terhadap ijtihak dan fajar di lokasi referensi.
PNG	<i>Portable Network Graphics</i>	Format gambar yang digunakan untuk menyimpan grafik, kalender, peta, dan visualisasi aplikasi.
RA	<i>Right Ascension</i>	Asensio Rekta, koordinat ekuatorial yang berfungsi seperti bujur pada bola langit.
RAM	<i>Random Access Memory</i>	Memori kerja komputer. Disebut dalam dokumentasi teknis dalam kaitannya dengan efisiensi pengolahan data numerik.
SM	Sebelum Masehi	Format tahun yang digunakan oleh fungsi penanganan tahun non-Masehi dalam antarmuka bahasa Indonesia.
SPK	<i>Spacecraft and Planet Kernel</i>	Jenis kernel SPICE untuk penyimpanan data posisi benda tata surya; file .bsp JPL termasuk dalam struktur data ini.
SQM	<i>Sky Quality Meter</i>	Istilah yang berkaitan dengan evaluasi tingkat kegelapan/kecerahan langit, digunakan pada modul Evaluasi Fajar Shadiq & SQM .
SSL	<i>Secure Sockets Layer</i>	Komponen keamanan komunikasi jaringan. Source code memiliki penyesuaian SSL agar akses jaringan tetap dapat berjalan pada lingkungan aplikasi tertentu.

Tk	<i>Tkinter/Tk</i>	Toolkit antarmuka grafis yang menjadi fondasi GUI aplikasi bersama CustomTkinter.
TT	<i>Terrestrial Time</i>	Skala waktu astronomis yang digunakan dalam berbagai kalkulasi Skyfield, termasuk representasi Julian Date TT.
TXT	<i>Text File</i>	Format teks sederhana untuk menyimpan hasil kalkulasi atau laporan.
TZ	<i>Time Zone</i>	Zona waktu atau offset waktu lokal terhadap UTC.
UI	<i>User Interface</i>	Antarmuka yang berinteraksi langsung dengan pengguna; dalam kode dapat mencakup GUI, tombol, label, textbox, sidebar, dan sebagainya.
URL	<i>Uniform Resource Locator</i>	Alamat sumber atau layanan web. Konfigurasi awal kode memuat beberapa URL server untuk WinAI, data kalender, dan Peta Langit Dunia.
UTC	<i>Coordinated Universal Time</i>	Standar waktu internasional yang menjadi salah satu acuan utama kalkulasi astronomi KHGT Times.
WGS84	<i>World Geodetic System 1984</i>	Sistem referensi geodesi yang digunakan Skyfield untuk membentuk lokasi pengamat berdasarkan lintang, bujur, dan elevasi.
WIB	Waktu Indonesia Barat	Zona waktu UTC+7 yang digunakan pada sejumlah keluaran dan kalkulasi aplikasi.
WinAI	<i>Windows Artificial Intelligence / WinAI Assistant</i>	Nama modul AI terintegrasi pada KHGT Times V8.0. Modul ini mendukung dialog, konsultasi, dan <i>Smart Interceptor</i> untuk menghubungkan permintaan pengguna dengan fungsi aplikasi.

Konfigurasi awal source code memang memuat URL server WinAI dan Peta Langit Dunia, serta konfigurasi kernel `de440s.bsp` dan `de441-new.bsp`. SOURCE CODE KHGT TIMES V8.0 Adapun struktur menu aplikasi menggunakan istilah seperti KHGT, SQM, Ephemeris 3D, WinAI, Peta Langit Dunia, dan HIJRI_DB Auto-Builder. SOURCE CODE KHGT TIMES V8.0(1)

CATATAN PENGGUNAAN GLOSARIUM

Glosarium dalam buku **Source Code KHGT Times V8.0** disusun untuk membantu pembaca memahami istilah astronomi, ilmu falak, kalender Hijriah, pemrograman Python, sistem koordinat, library, format data, serta istilah teknis yang muncul di dalam kode program dan antarmuka aplikasi. Glosarium terutama ditujukan sebagai **panduan terminologi kontekstual**, sehingga definisi yang diberikan mengacu pada cara suatu istilah digunakan dalam KHGT Times V8.0.

Istilah dalam glosarium tidak dimaksudkan untuk menggantikan definisi lengkap dalam buku teks astronomi, astrometri, ilmu falak, geodesi, ilmu komputer, ataupun dokumentasi resmi suatu library. Sebuah istilah dapat mempunyai cakupan yang lebih luas dalam disiplin ilmiah tertentu, sedangkan glosarium ini memusatkan penjelasan pada fungsi dan maknanya **di dalam arsitektur KHGT Times V8.0**.

Sebagai contoh, istilah **geosentris** dan **toposentris** dalam buku perlu dibaca sesuai dengan metode perhitungan yang digunakan oleh program. Geosentris mengacu pada perhitungan dengan pusat Bumi sebagai titik referensi, sedangkan toposentris mengacu pada perhitungan dari lokasi pengamat tertentu di permukaan Bumi. Source code secara nyata menggunakan `wgs84.latlon()` dan objek pengamat Skyfield untuk kalkulasi berbasis lokasi. Hal ini penting karena nilai ketinggian, azimut, dan beberapa parameter astronomis dapat berbeda menurut sistem referensinya.

Demikian pula, istilah **ephemeris** dalam buku terutama mengacu pada basis data dan engine posisi astronomis yang digunakan aplikasi. KHGT Times V8.0 memakai mekanisme pemilihan otomatis `de440s.bsp` untuk tahun 1850–2149 dan `de441-new.bsp` di luar rentang tersebut sampai batas aplikasi 1–2999 M. Karena itu, ketika pembaca menemukan istilah DE440s, DE441, BSP, JPL, atau ephemeris, istilah tersebut harus dibaca dalam hubungan dengan sistem komputasi posisi Matahari, Bulan, planet, dan fenomena astronomi aplikasi. SOURCE CODE KHGT TIMES V8.0

Istilah **KHGT**, **PKG 1**, dan **PKG 2** perlu dibaca sebagai bagian dari logika kalender global yang diimplementasikan dalam aplikasi. Glosarium menjelaskan fungsi operasionalnya pada program, sedangkan pembahasan metodologi, kriteria astronomis, dan argumentasi kalender tetap harus dirujuk kepada bagian dokumentasi KHGT serta daftar pustaka yang disediakan pada buku.

Begitu pula dengan **Neo-MABIMS**. Pada modul komparasi tertentu, kode menggunakan parameter tinggi dan elongasi yang ditetapkan di dalam algoritma aplikasi untuk membandingkan hasil dengan KHGT. Karena itu, hasil komputasi harus dibaca dengan memperhatikan apakah parameter yang ditampilkan adalah **geosentris** atau **toposentris**, lokasi referensi yang digunakan, waktu evaluasi, serta jenis kriteria yang sedang dipilih.

Untuk istilah library seperti **Skyfield**, **PyEphem**, **NumPy**, **SciPy**, **Matplotlib**, **Pillow**, **CustomTkinter**, **Requests**, **Beautiful Soup**, dan library lainnya, glosarium hanya menjelaskan fungsi utama yang relevan terhadap KHGT Times. Dokumentasi teknis dan sintaks lengkap setiap

library tetap mengacu pada dokumentasi resminya. Source code sendiri menjelaskan penggunaan library-library tersebut sebagai bagian dari engine astrometri, visualisasi, interpolasi, GUI, pemrosesan citra, dan komunikasi jaringan. SOURCE CODE KHGT TIMES V8.0

Nama fungsi, variabel, file, class, konstanta, dan objek kode yang ditulis dengan format seperti HIJRI_DB, de440s.bsp, de441-new.bsp, wgs84.latlon(), fraction_illuminated(), atau .apparent() sebaiknya **dipertahankan persis sebagaimana ditulis dalam source code**. Penamaan tersebut mempunyai arti teknis dan tidak dianjurkan diterjemahkan ketika digunakan untuk menjelaskan struktur program.

Istilah bahasa Inggris yang telah umum digunakan dalam pemrograman dan astronomi—misalnya *altitude*, *azimuth*, *elongation*, *observer*, *sunset*, *transit*, *moonphase*, *threading*, *cache*, *fallback*, *rendering*, dan *real-time*—dapat tetap dicantumkan bersama padanan bahasa Indonesianya. Pendekatan ini memudahkan pembaca menghubungkan penjelasan buku dengan nama variabel, dokumentasi library, dan kode program yang asli.

Untuk istilah **WinAI**, definisi dalam glosarium harus dipahami sebagai fungsi yang diimplementasikan pada KHGT Times V8.0. Source code menggambarkan WinAI sebagai asisten yang menerima prompt pengguna dan menggunakan mekanisme *Smart Interceptor* untuk mengarahkan permintaan tertentu ke fungsi aplikasi. Pada dokumentasi 41 modul, WinAI ditempatkan sebagai **Menu 38**, sedangkan Peta Langit Dunia, Laporan Astronomi, dan HIJRI_DB Auto-Builder masing-masing menjadi Menu 39, 40, dan 41. SOURCE CODE KHGT TIMES V8.0(1)

Singkatan **ID**, **EN**, dan **AR** pada konteks bahasa masing-masing merujuk pada Indonesia, Inggris, dan Arab. KHGT Times V8.0 menggunakan ketiganya pada sistem penerjemahan GUI dan dokumentasi. Penggunaan singkatan bahasa tersebut tidak boleh disamakan dengan singkatan lain yang kebetulan memiliki huruf yang sama dalam disiplin berbeda.

Nilai numerik dan satuan dalam glosarium harus selalu dibaca sesuai konteks. Simbol ° menyatakan derajat sudut; % digunakan antara lain untuk iluminasi; **m** dapat menunjukkan meter ketika berkaitan dengan elevasi; sedangkan jam dapat muncul sebagai UTC, LT, atau zona waktu lokal tertentu. Pembaca dianjurkan memeriksa judul kolom dan parameter modul sebelum membandingkan dua hasil perhitungan.

Glosarium juga membedakan antara **istilah ilmiah**, **nama modul**, dan **nama implementasi program**. Sebagai contoh, “gerhana” adalah istilah astronomis, “Analisis Gerhana” adalah nama modul, sedangkan nama fungsi Python yang menjalankannya merupakan implementasi perangkat lunak. Ketiganya berhubungan tetapi tidak selalu identik dalam ruang lingkup makna.

Istilah atau parameter yang berubah karena pembaruan library, API, basis data, atau algoritma perlu dibaca berdasarkan **versi KHGT Times V8.0**. Buku ini mendokumentasikan keadaan source code pada versi tersebut. Perubahan pada versi aplikasi selanjutnya dapat menyebabkan perubahan pada nama fungsi, struktur menu, dependency, metode kalkulasi, atau rentang ephemeris.

Jika terdapat perbedaan antara istilah dalam uraian buku dengan nama literal pada source code, **nama yang tertulis dalam source code V8.0 menjadi acuan teknis utama** untuk memahami implementasinya. Glosarium berfungsi membantu interpretasi, bukan mengubah struktur atau terminologi asli program.

Sumber: <https://hisabmu.com/khgttimes/>

- KASMUI -
Dosen Kimia, Komputasi, IT, AI UNNES

<https://hisabmu.com/>
<https://kasmui.cloud/>

Ketua PCM
Gunungpati 2

Anggota Majelis Tabligh
PDM Kota Semarang
& PWM Jawa Tengah

Tim Pengembang
Software KHGT
MTT PP Muhammadiyah

Praktisi ✨
Ilmu Falak

SOURCE CODE KHGT TIMES V8.0

APLIKASI HISAB ASTRONOMI DAN KALENDER ISLAM GLOBAL

DENGAN MULTIBAHASA, ANALISIS GRAFIK,
DAN INTEGRASI PETA LANGIT DUNIA

SINOPSIS

Buku **Source Code KHGT Times V8.0** ini merupakan dokumentasi lengkap dan sistematis dari pengembangan aplikasi KHGT Times, sebuah perangkat lunak komputasi astronomi dan ilmu falak yang dirancang untuk mendukung perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal secara global, pengelolaan kalender Islam, serta kajian dan implementasi Kalender Hijriah Global Tunggal (KHGT).

Pada versi 8.0, aplikasi ini telah berkembang menjadi **41 modul menu** terintegrasi yang mencakup berbagai kebutuhan perhitungan, analisis, visualisasi, pemetaan, dan simulasi fenomena langit. Buku ini menyajikan seluruh **source code Python** secara lengkap, disertai penjelasan fungsi, logika algoritma, struktur program, dan panduan implementasinya.

KHGT Times V8.0 menggunakan pustaka astronomi **Skyfield** dengan mekanisme **auto-switch ephemeris**, yaitu **de440s.bsp** untuk rentang tahun 1850–2149 dan **de441-new.bsp** untuk rentang 1–1849 serta 2150–2999. Aplikasi ini menggabungkan perhitungan **geosentris** dan **toposentris**, dilengkapi antarmuka grafis modern berbasis Python, dukungan multibahasa **Indonesia–Inggris–Arab**, **Peta Langit Dunia**, **Laporan Astronomi**, **HIJRI_DB Auto-Builder**, serta integrasi **WinAI** sebagai asisten kecerdasan buatan.

Buku ini tidak hanya menjadi dokumentasi teknis bagi pengembang, tetapi juga referensi edukatif bagi akademisi, peneliti, observatorium, dan masyarakat yang ingin memahami implementasi ilmu **falak** modern berbasis komputasi. Dengan pendekatan yang terstruktur dan mudah diikuti, pembaca dapat mempelajari cara kerja aplikasi, mengembangkan fitur lebih lanjut, serta memanfaatkan KHGT Times V8.0 sebagai sarana edukasi dan penelitian dalam bidang astronomi dan kalender Islam global.