KONSEP, FORMULASI, DAN ARSITEKTUR KODE KHGT TIMES

KASMUI


KONSEP, FORMULASI,
DAN ARSITEKTUR KODE
KHGT TIMES
Great Circles
Arah Kiblat
Hilal
Crescent Tracking
Validation
Skyfield
Simulasi 3D Snuisis
Analisis Gerhana
Database JSON
Z-Buffer
Skyfield
Software
Ephemeris
Validation
Z-Buffer
1 Inport Gonter
2 Skyfield "Skyfield,'Ephemeris')
3 Import a 30
4 Ephemeris = (1)
5 for bs = 11;
6 Skd:(batize_reah_fieldr);
7 state = 12:5
8 (crescent_tracking) {
9 return 'Crescent Tracking';
10 }
11 print, '2>Buffer"
12 return []
13)
KASMUI

KONSEP, FORMULASI, DAN ARSITEKTUR KODE KHGT TIMES

Penulis: KASMUI

Desain Sampul & Penata Letak: KASMUI / Tim Artistik Penerbit

Informasi Penerbitan & Cetakan: Jaten & Siblings Gang Jaten, Patemon, Gunungpati, Kota Semarang Cetakan I, Edisi Tahun 2026

Perpustakaan Nasional RI: Katalog Dalam Terbitan (KDT) KASMUI Konsep, Formulasi, dan Arsitektur Kode KHGT Times (Edisi I) Semarang Deskripsi Fisik: (hlm; 21 x 29,7 cm) ISBN: 978-623-XXXX-XX-X Subjek: Sejarah Islam, Biografi Ibnu Batutah, Historiografi beserta Nomor Klasifikasi DDC

Informasi Percetakan: *Dicetak oleh Jaten Press*

KATA PENGANTAR

Segala puji bagi Allah *Subhanahu wa Ta'ala*, Dzat yang telah menciptakan semesta alam dengan arsitektur matematis yang sangat presisi, yang menjadikan pergerakan benda-benda langit sebagai basis bagi manusia dalam merumuskan sistem waktu dan peradaban. Shalawat serta salam senantiasa tercurah kepada Baginda Nabi Muhammad *Shallallahu 'alaihi wa Sallam*, sosok revolusioner yang membawa risalah pencerahan bagi akal dan spiritualitas umat manusia.

Al-Qur'an secara eksplisit telah meletakkan fondasi kosmologis terkait urgensi perhitungan waktu (hisab) melalui pergerakan Bulan dan Matahari. Allah berfirman dalam Surat Yunus Ayat 5:

هُوَ ٱلَّذِى جَعَلَ ٱلشَّمْسَ ضِيَآءً وَٱلْقَمَرَ نُورًا وَقَدَّرَهُۥ مَنَازِلَ لِتَعْلَمُوا۟ عَدَدَ ٱلسِّنِينَ وَٱلْحِسَابَ ۚ مَا خَلَقَ ٱللَّهُ ذَٰلِكَ إِلَّا بِٱلْحَقِّ ۚ يُفَصِّلُ ٱلْءَايَٰتِ لِقَوْمٍ يَعْلَمُونَ

Artinya: "Dialah yang menjadikan matahari bersinar dan bulan bercahaya dan ditetapkan-Nya manzilah-manzilah (tempat-tempat) bagi perjalanan bulan itu, supaya kamu mengetahui bilangan tahun dan perhitungan (waktu). Allah tidak menciptakan yang demikian itu melainkan dengan hak. Dia menjelaskan tanda-tanda (kebesaran-Nya) kepada orang-orang yang mengetahui." (QS. Yunus [10]: 5)

Ayat di atas bukan sekadar dogma teologis, melainkan sebuah postulat saintifik yang menantang kaum intelektual (*qaumun ya'lamun*) untuk terus mengembangkan instrumen observasi dan komputasi astrometri. Sepanjang sejarah peradaban Islam, dinamika penentuan awal bulan kamariah senantiasa menjadi diskursus yang hangat, bahkan tak jarang bermuara pada polemik sosiologis akibat perbedaan metodologi antara *rukyat* (observasi visual) dan *hisab* (kalkulasi matematis), serta antara mazhab *matlak* lokal (zonal) dan global.

Urgensi dan Konteks Lahirnya KHGT Memasuki abad ke-21, paradigma fikih astronomi mengalami pergeseran fundamental. Perbedaan perayaan hari-hari besar Islam (Ramadhan, Syawal, Zulhijah) di era globalisasi informasi dirasakan sebagai sebuah paradoks. Umat Islam terhubung secara digital dalam hitungan detik, namun terfragmentasi secara temporal dalam urusan ibadah. Menjawab tantangan ini, Muktamar Internasional Penyatuan Kalender Hijriah di Istanbul (2016) merumuskan sebuah resolusi bersejarah: **Kalender Hijriah Global Tunggal (KHGT)**. KHGT, yang kini juga telah diadopsi secara resmi oleh persyarikatan Muhammadiyah (meninggalkan kriteria *wujudul hilal* murni), beroperasi di atas prinsip kesatuan matlak bumi (*ittihad al-matali'*), transfer visibilitas (*naql al-rukyat*), dan parameter visibilitas hilal yang universal (Ketinggian Geosentris >= 5 derajat dan Elongasi Geosentris >= 8 derajat).

Namun, gagasan cemerlang KHGT berbenturan dengan realitas empiris di lapangan: ketiadaan infrastruktur perangkat lunak (*software*) komputasi yang secara spesifik dirancang untuk mengeksekusi algoritma KHGT secara komprehensif, presisi, dan dapat diakses oleh publik akademis secara transparan.

Kesenjangan Riset (Research Gap) dan Kebaruan Selama beberapa dekade terakhir, dunia falak Islam sangat bergantung pada perangkat lunak pionir dari Timur Tengah. Meski sangat berjasa, perangkat lunak generasi awal tersebut dihadapkan pada limitasi komputasi modern.

Mereka umumnya dibangun di atas algoritma *VSOP87* (Variations Séculaires des Orbites Planétaires) yang menggunakan deret analitik klasik, arsitektur *single-thread* yang kurang responsif terhadap kalkulasi *big data*, serta orientasi desain yang murni ditujukan untuk analisis *rukyatul hilal* lokal/zonal. Sistem lama tidak memiliki kapabilitas untuk melakukan pemindaian global (*global scanning*) secara instan—sebuah syarat mutlak dalam KHGT untuk menemukan titik daratan utama (*mainland*) pertama di bumi yang memenuhi kriteria pada hari terjadinya konjungsi (Ijtimak).

Buku ini, beserta perangkat lunak yang dibedahnya, yakni **KHGT Times V7.4**, hadir untuk mengisi ruang kosong tersebut. KHGT Times V7.4 merepresentasikan lompatan kuantum (*quantum leap*) dalam ilmu falak digital. Kebaruan (*novelty*) dari sistem ini terletak pada transisi epistemologis dan teknologis:

1. **Integrasi Numerik Tingkat Tinggi:** Meninggalkan teori analitik lama dan beralih menggunakan basis data *Development Ephemeris* (DE441) dari *Jet Propulsion Laboratory* (JPL) NASA, memungkinkan perhitungan posisi vektor tata surya dengan akurasi ekstrem dari tahun 1 hingga 3000 Masehi.
2. **Arsitektur Pemrosesan Modern:** Memanfaatkan ekosistem Python (*Skyfield, SciPy, NumPy*) dengan dukungan *multi-threading* untuk melakukan iterasi waktu per detik dan merajut ribuan titik koordinat menjadi Peta Visibilitas Hilal beresolusi tinggi (*HD Visibility Map*).
3. **Otomatisasi Komparatif & AI:** Kehadiran fitur komparasi hisab 50 tahun ke depan (KHGT vs MABIMS) dan injeksi Kecerdasan Buatan (WinAI) yang mampu merespons *prompt* berbasis *Retrieval-Augmented Generation* (RAG) dari literatur fikih dan data ephemeris aktual.

Buku *"Konsep, Formulasi, dan Arsitektur Kode KHGT Times V7.4"* ini ditulis untuk membedah anatomi perangkat lunak tersebut hingga ke tingkat algoritma. Setiap bab akan mengupas tuntas landasan teoretis astronomi, perumusan matematisnya, serta bagaimana formula tersebut diterjemahkan ke dalam blok kode Python tingkat lanjut.

Harapan penulis, karya ini tidak hanya menjadi buku manual teknis (*user guide*), melainkan menjelma sebagai literatur referensi standar internasional bagi para mahasiswa astronomi Islam, peneliti observatorium, pakar falak, hingga pembuat kebijakan syariah. Semoga setiap baris kode dan persamaan yang tertuang di dalamnya menjadi bagian dari amal jariyah dalam merealisasikan cita-cita agung: menyatukan umat Islam dalam satu harmoni waktu global.

Semarang, 2026

Penulis

Kasmui

DAFTAR ISI

BAB 1: PENDAHULUAN: TRANSISI PARADIGMA FALAK DIGITAL

1.1. Evolusi Komputasi Astronomi Islam

Ilmu Falak (Astronomi Islam) merupakan salah satu disiplin ilmu eksakta tertua yang mengalami evolusi paling dinamis dalam peradaban manusia. Secara historis, peradaban Islam klasik mengandalkan observasi mata telanjang (*rukyat al-basariyyah*) yang didukung oleh instrumen mekanik analog seperti astrolabe (*al-asturlab*) dan kuadran (*ar-rub'*). Data observasi ini kemudian ditransformasikan ke dalam tabel-tabel astronomi matematis yang dikenal sebagai *Zij* (misalnya *Zij Ilkhani* karya Nashiruddin ath-Thusi atau *Zij as-Sultani* karya Ulugh Beg).

Memasuki era modern, revolusi digital mengubah konstelasi Ilmu Falak secara radikal. Tabel-tabel *Zij* yang dihitung secara manual selama berbulan-bulan kini digantikan oleh algoritma komputasi yang mampu merender triliunan titik koordinat benda langit dalam fraksi detik. Evolusi ini memfasilitasi transisi dari hisab *tahqiqi* (perhitungan presisi klasik) menuju hisab *kontemporer* yang beresolusi super tinggi, yang pada gilirannya menjadi fondasi teknis bagi penyatuan Kalender Hijriah Global Tunggal (KHGT) (Ilyas, 1997).

1.1.1. Dari Ephemeris Klasik menuju Integrasi Numerik (JPL NASA)

Selama akhir abad ke-20 hingga awal abad ke-21, komputasi astronomi di seluruh dunia—termasuk dalam perangkat lunak falak Islam generasi awal—sangat bergantung pada deret analitik (*analytical series*). Salah satu algoritma yang paling mendominasi adalah VSOP87 (*Variations Séculaires des Orbites Planétaires*) yang dikembangkan oleh Bretagnon dan Francou (1988) dari *Bureau des Longitudes*, Paris. Algoritma analitik ini bekerja dengan memecahkan persamaan gerak planet ke dalam ribuan deret polinomial harmonik. Untuk menghitung posisi Matahari atau planet, komputer cukup mengevaluasi deret sinus dan kosinus ini pada waktu tertentu (t).

Namun, seiring dengan meningkatnya kebutuhan akan akurasi absolut, terutama dalam penentuan posisi hilal di garis batas kriteria visibilitas yang sangat kritis (seperti syarat ketinggian geosentris 5 derajat pada KHGT), model analitik mulai ditinggalkan. Paradigma bergeser menuju metode **Integrasi Numerik** (*Numerical Integration*), yang menjadi jantung dari *Development Ephemeris* (DE) buatan *Jet Propulsion Laboratory* (JPL) NASA.

Berbeda dengan VSOP87 yang menggunakan pendekatan matematis hampiran, sistem integrasi numerik JPL menghitung dinamika gravitasi tata surya selangkah demi selangkah (menggunakan metode seperti *Adams-Bashforth-Moulton*) berdasarkan hukum Relativitas Umum Einstein. Model ini memperhitungkan interaksi tarik-menarik antara Matahari, 8 planet, Pluto, Bulan, hingga ratusan asteroid besar yang secara dinamis terus memengaruhi posisi Bulan (Park et al., 2021).

Dalam arsitektur perangkat lunak **KHGT Times V7.4**, lompatan teknologi ini secara tegas diimplementasikan melalui penggunaan file ephemeris `de441-new.bsp`. File DE441 merupakan standar ephemeris NASA terbaru yang mencakup rentang waktu sangat panjang (dari tahun -13200 hingga +17191 Masehi) dengan tingkat akurasi presisi pada level sub-milidetik busur (*milliarcsecond*) (Park et al., 2021). KHGT Times membaca struktur biner *Chebyshev polynomials*

di dalam file ini menggunakan *library* saintifik `skyfield`, sebagaimana tercermin dalam inisialisasi mesin utamanya:

Python

```
self.load_obj = Loader(BASE_DIR, verbose=False)
self.ts = self.load_obj.timescale()
self.eph = self.load_obj('de441-new.bsp')
```

Pendekatan ini memastikan bahwa setiap perhitungan umur bulan (*moon age*), elongasi, dan ketinggian bebas dari akumulasi *truncation error* yang sering merusak hasil hisab jangka panjang.

1.1.2. Keterbatasan Algoritma Lama (VSOP87) di Era Modern

Peralihan menuju Ephemeris NASA dalam KHGT Times V7.4 bukan tanpa alasan empiris. Keterbatasan utama dari algoritma klasik analitik (seperti VSOP87 atau ELP2000-82B untuk Bulan) terletak pada degradasi akurasi temporal. Deret analitik dipotong (*truncated*) pada jumlah tertentu demi menghemat daya komputasi komputer dekade 1980-an (Urban & Seidelmann, 2013). Akibatnya, semakin jauh komputasi ditarik ke masa lalu atau diproyeksikan ke masa depan (misalnya simulasi komparasi 50 tahun ke depan untuk penentuan awal Ramadhan dan Syawal), simpangan (*error/drift*) posisinya akan semakin membesar.

Selain itu, algoritma lama sering kali gagal menangani *perturbasi* jangka pendek tingkat mikro pada orbit bulan (seperti efek pasang surut kelautan dan anomali geopotensial bumi). Dalam konteks penyatuan Kalender Hijriah Global, kesalahan fraksi derajat atau keterlambatan hitungan konjungsi (ijtimak) sebesar hitungan detik sangat berakibat fatal.

Sebagai contoh, berdasarkan Parameter Kalender Global (PKG) 1 KHGT, jika kriteria visibilitas terpenuhi di mana saja di bumi tepat sebelum pukul 00:00 UTC, maka bulan baru dimulai keesokan harinya secara global. Jika mesin komputasi menggunakan VSOP87 mengalami bias waktu ijtimak atau bias perhitungan elongasi bumi-bulan pada detik-detik kritis (misalnya pukul 23:59:58 UTC), sistem kalender global dapat mengalami kegagalan validasi lintas benua, menyebabkan pergeseran hari raya (Syamsuddin, 2021).

Oleh karena itu, perangkat lunak astronomi masa depan menuntut standarisasi data yang tidak lagi mengandalkan kalkulator *built-in* dari rumusan analitik. Penggunaan integrasi *multi-library*—di mana `skyfield` menangani geometri absolut DE441, dan `ephem` (PyEphem berbasis algoritma C klasik) direduksi fungsinya sebatas *accelerator* pelacakan fajar/senja secara *brute-force*—merupakan solusi teknis yang brilian dalam arsitektur KHGT Times untuk mencapai performa yang *robust* tanpa mengorbankan akurasi.

1.2. Urgensi Kalender Hijriah Global Tunggal (KHGT)

Paradigma penanggalan dalam Islam tidak sekadar berfungsi sebagai penanda administrasi sipil, melainkan memiliki implikasi ibadah (*ta'abbudi*) yang sangat sakral. Puasa Ramadhan, Idul Fitri, ibadah Haji, hingga masa *iddah* dan haul zakat seluruhnya bertumpu pada presisi pergerakan bulan kamariah. Namun, selama berabad-abad, umat Islam di berbagai belahan dunia merayakan hari-

hari besar tersebut pada tanggal masehi yang berbeda-beda. Fragmentasi temporal ini bukan berakar pada kelemahan sains, melainkan pada dikotomi penafsiran yurisprudensi (fikih) terhadap teks-teks syariat terkait jangkauan keberlakuan *rukyatul hilal*.

Menjawab tantangan di era *global village* ini, Kalender Hijriah Global Tunggal (KHGT) yang dirumuskan pada Kongres Istanbul 2016 hadir sebagai sintesis antara ortodoksi fikih dan kecanggihan astrometri modern.

1.2.1. Dalil Syar'i tentang Penyatuan Waktu Global

Secara teologis, KHGT dibangun di atas premis bahwa syariat Islam diturunkan untuk seluruh umat manusia secara universal (*rahmatan lil 'alamin*). Universalitas ini menuntut adanya sistem penjadwalan yang sinkron di seluruh dunia. Fondasi utama penyatuan awal bulan termaktub dalam firman Allah *Subhanahu wa Ta'ala*:

شَهْرُ رَمَضَانَ ٱلَّذِىٓ أُنزِلَ فِيهِ ٱلْقُرْءَانُ هُدًى لِّلنَّاسِ وَبَيِّنَٰتٍ مِّنَ ٱلْهُدَىٰ وَٱلْفُرْقَانِ ۚ فَمَن شَهِدَ مِنكُمُ ٱلشَّهْرَ فَلْيَصُمْهُ

Artinya: "(Beberapa hari yang ditentukan itu ialah) bulan Ramadhan, bulan yang di dalamnya diturunkan (permulaan) Al-Qur'an sebagai petunjuk bagi manusia dan penjelasan-penjelasan mengenai petunjuk itu dan pembeda (antara yang hak dan yang bathil). Karena itu, barangsiapa di antara kamu hadir (di negeri tempat tinggalnya) di bulan itu, maka hendaklah ia berpuasa pada bulan itu..." (QS. Al-Baqarah [2]: 185)

Kata *syahida* (شَهِدَ) dalam ayat tersebut, secara etimologis dan epistemologis, tidak dimaknai secara sempit sebagai "melihat bulan dengan mata kepala secara fisik", melainkan bermakna *hadir* atau *mengetahui dengan keyakinan (ilmu)* bahwa bulan baru telah masuk (Anwar, 2008). Keyakinan inilah yang di era modern dicapai melalui instrumen hisab yang presisi.

Lebih lanjut, hadits Nabi *Shallallahu 'alaihi wa Sallam* yang sering dijadikan rujukan utama dalam penentuan awal bulan berbunyi:

صُومُوا لِرُؤْيَتِهِ وَأَفْطِرُوا لِرُؤْيَتِهِ فَإِنْ غُمَّ عَلَيْكُمْ فَأَكْمِلُوا عِدَّةَ شَعْبَانَ ثَلَاثِينَ

Artinya: "Berpuasalah kalian karena melihatnya (hilal) dan berbukalah kalian karena melihatnya. Jika ia tertutup awan dari kalian, maka sempurnakanlah hitungan bulan Syakban menjadi tiga puluh hari." (HR. Bukhari no. 1909 dan Muslim no. 1081).

Dari perspektif *ushul fiqh*, lafaz *shumuu* (صُومُوا) dan *aftiruu* (أَفْطِرُوا) menggunakan *shighat jama'* (bentuk jamak/kolektif) yang ditujukan kepada seluruh umat Islam (Zayed, 2015). Ini mengindikasikan bahwa perintah puasa dan berhari raya pada hakikatnya adalah perintah kolektif-universal, bukan individual-regional. Pemahaman klasik yang membatasi keberlakuan rukyat hanya pada zona geografis tertentu (*ikhtilaf al-matali'*) dinilai tidak lagi relevan ketika teknologi komunikasi memungkinkan umat Islam di Timur mengetahui hasil observasi astronomis di Barat dalam hitungan mikrodetik (Ilyas, 1997).

1.2.2. Prinsip Kesatuan Matlak (*Ittihad al-Matali'*) dan Transfer Visibilitas

Untuk merealisasikan makna kolektif dari dalil-dalil tersebut, KHGT mengadopsi prinsip Kesatuan Matlak (*Ittihad al-Matali'*). Prinsip ini menegaskan bahwa bumi adalah satu kesatuan zona penanggalan (satu matlak). Implikasi logis dari prinsip ini adalah berlakunya mekanisme **Transfer Visibilitas** (*Naql al-Rukyat*). Secara saintifik, jika hilal telah memenuhi kriteria visibilitas (Ketinggian ge 5^\circ dan Elongasi ge 8^\circ) di belahan bumi Barat (misalnya di benua Amerika), maka status bulan baru tersebut segera "ditransfer" ke belahan bumi Timur, meskipun di wilayah Timur matahari telah terbenam lebih dulu dan hilal belum memenuhi kriteria secara lokal.

Dalam implementasi algoritmiknya, **KHGT Times V7.4** dirancang secara khusus untuk menjadi *engine* pelacak (scanner) matlak global ini. Hal ini terlihat sangat jelas pada **Modul 5 (Kota Pertama KHGT / Mainland)**. Melalui fungsi `calculate_first_point()`, program mengeksekusi metode *brute-force* yang memindai ribuan koordinat dalam direktori `CITY_DB` secara simultan.

Alih-alih hanya menghitung di satu titik pengamat (toposentris murni) seperti perangkat lunak falak konvensional, kode program KHGT Times menerapkan iterasi terhadap *New Moon* (Ijtimak) menggunakan metode `ephem.previous_new_moon()` dan `ephem.next_new_moon()`, kemudian melacak waktu *sunset* di setiap kota:

Sistem secara cerdas menerapkan *Filter Zona Pengecualian* untuk mematuhi aturan daratan utama (*mainland rule*):

```
zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu",
"Maluku", "Maluku Utara", "NTT"]
```

Program memastikan bahwa titik pertama dimulainya bulan baru bukan berada di pulau kecil yang terisolasi di samudra Pasifik, melainkan di daratan yang memiliki populasi manusia berkelanjutan. Jika titik valid pertama ditemukan sebelum pukul 00:00 UTC (PKG 1), maka seluruh dunia secara serentak (Ittihad al-Matali') akan memulai tanggal 1 pada keesokan harinya. Inilah perwujudan nyata dari dalil syar'i yang diterjemahkan ke dalam bahasa pemrograman tingkat tinggi.

1.3. Visi dan Arsitektur KHGT Times V7.4

Perangkat lunak falak yang sekadar "menghitung" jadwal sudah sangat tertinggal untuk menjawab tantangan astronomi abad ke-21. Visi utama dari pengembangan **KHGT Times V7.4** adalah meruntuhkan dinding pembatas antara kompleksitas ephemeris tingkat tinggi milik lembaga antariksa dan aksesibilitas *user interface* (GUI) bagi praktisi, akademisi, serta pembuat kebijakan syariah. Arsitektur V7.4 didesain bukan sekadar sebagai kalkulator, melainkan sebagai sebuah *engine* komputasi (mesin pemroses) berskala besar yang mampu mengeksekusi integrasi numerik, komparasi multi-kriteria global, hingga *rendering* visual spasial secara bersamaan.

1.3.1. Keseimbangan *High Precision* dan *High Performance*

Dalam paradigma komputasi saintifik, secara arsitektural selalu terdapat *trade-off* (kompromi) antara *high precision* (akurasi ekstrem) dan *high performance* (kecepatan eksekusi) (Virtanen et al., 2020). Mengakses posisi vektor tiga dimensi dari bulan dan bumi melalui *Development Ephemeris* (DE441) membutuhkan evaluasi polinomial Chebyshev yang memakan daya komputasi (*CPU cycles*) jauh lebih besar dibandingkan mengevaluasi deret analitik sederhana.

Jika sistem KHGT Times secara naif melakukan perhitungan *looping* linear mathcal{O}(n) untuk melacak waktu terbenamnya matahari (*sunset*) di 300 kota di dunia selama 30 hari berturut-turut, proses tersebut akan memicu *bottleneck* (kemacetan) dan membekukan antarmuka aplikasi (*GUI freeze*).

Untuk memecahkan kebuntuan tersebut, arsitektur KHGT Times V7.4 menerapkan dua metodologi utama:

1. **Pendekatan Vektorisasi Array:** Mengonversi data waktu diskrit menjadi larik (*array*) waktu berskala besar menggunakan *library* NumPy (seperti implementasi `np.linspace`). Evaluasi posisi benda langit tidak lagi dilakukan satu per satu per detik, melainkan dihitung secara simultan menggunakan komputasi matriks yang langsung dieksekusi di level bahasa C.
2. **Asynchronous Threading:** Seluruh komputasi masif, seperti pemindaian daratan utama (Modul 5) dan komparasi 50 tahun (Modul 20 dan 21), dieksekusi di luar *Main Thread* GUI menggunakan modul `threading` internal Python. Hal ini memastikan UI tetap responsif (*non-blocking*) sementara komputasi *big data* berjalan di latar belakang (Bassi, 2007).

1.3.2. Ekosistem Library Python (Pusat Kendali Astrometri)

Bahasa pemrograman Python dipilih sebagai fondasi KHGT Times bukan karena kecepatannya, melainkan karena kapabilitasnya sebagai *glue language* (bahasa perekat) yang mempu menyatukan berbagai pustaka (*library*) saintifik berbasis C, C++, dan Fortran. Arsitektur V7.4 didukung oleh ekosistem *library* berikut yang saling bersinergi:

1. **Skyfield API:** Berperan sebagai jantung astrometri utama menggantikan modul-modul usang. *Skyfield* memproses *file* biner NASA (`de441-new.bsp`) dan bertanggung jawab atas perhitungan relativistik, refraksi atmosfer, koreksi paralaks, dan rotasi matriks koordinat geosentris menuju toposentris menggunakan referensi datum elipsoid WGS84 mutakhir (Rhodes, 2019).
2. **SciPy & NumPy:** Pustaka esensial untuk manipulasi array dan matematika kompleks. Pada Modul 2 (Crescent Visibility HD Map), sistem menggunakan sub-modul `scipy.interpolate.griddata` dengan algoritma *Bicubic Interpolation*. Algoritma ini mengubah titik-titik diskrit hasil perhitungan kasar dari berbagai lintang dan bujur menjadi matriks kontur spasial spasial beresolusi tinggi yang sangat mulus (*seamless*).
3. **Matplotlib:** Mesin *rendering* yang mentransformasi data mentah numerik menjadi representasi visual saintifik, seperti Grafik Ketinggian Harian (Modul 4), Simulasi Kompas

Mizwala 2D (Modul 29), hingga interaksi kanvas *3D Geocentric Space* (Modul 16) secara *real-time* (Hunter, 2007).

4. **PyEphem (`ephem`):** Meskipun *Skyfield* unggul dalam akurasi absolut, kalkulasi pencarian titik kritis (*discrete events*) seperti iterasi pencarian *sunset* berulang sangat membebani komputasi. Oleh karena itu, *PyEphem* yang ditulis langsung dalam bahasa C (diadopsi dari *XEphem*) dipertahankan secara khusus oleh arsitek KHGT Times sebagai *akselerator brute-force* untuk melakukan pemindaian global pada ratusan kota secara instan.

Integrasi harmonis dari ekosistem di atas menjadikan KHGT Times V7.4 tidak hanya presisi dan diakui secara akademik, tetapi juga siap digunakan secara praktis oleh komunitas astronomi global.

DAFTAR PUSTAKA BAB 1

Anwar, S. (2008). *Hari Raya dan Problematika Penentuan Awal Bulan Hijriah*. Suara Muhammadiyah.

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.

Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. VSOP 87 solutions. *Astronomy and Astrophysics*, 202, 309-315.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

Ilyas, M. (1997). *Islamic Calendar, Times & Qibla*. Berita Publishing.

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.

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl-1907.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

Zayed, A. (2015). *Fikih Astronomi: Menjawab Problematika Penentuan Awal Bulan*. Darul Ulum.

BAB 2: MESIN ASTROMETRI DAN MANAJEMEN WAKTU (TIME & SPACE ENGINE)

Dalam komputasi astronomi modern, ketepatan penentuan visibilitas hilal tidak lagi sekadar bergantung pada geometri dua dimensi, melainkan sangat terikat pada model relativistik tata surya dan sistem manajemen waktu yang seragam. Bab ini akan membedah secara mendalam bagaimana **KHGT Times V7.4** membangun mesin astrometrinya (*Astrometry Engine*) menggunakan pustaka *Skyfield* dan basis data ephemeris standar NASA, serta bagaimana sistem menangani kerumitan dilatasi waktu fisik.

2.1. Membedah Ephemeris JPL DE441

Sistem tata surya tidak bergerak dalam lintasan melingkar yang diam, melainkan dalam perturbasi gravitasi yang sangat kompleks yang saling tarik-menarik. Perangkat lunak KHGT Times meninggalkan formulasi analitik klasik dan mengadopsi penuh *Development Ephemeris* seri 441 (DE441) yang dirilis oleh *Jet Propulsion Laboratory* (JPL) NASA. DE441 merupakan model integrasi numerik yang dipetakan untuk merentang dari tahun 13200 Sebelum Masehi hingga 17191 Masehi, menjadikannya basis data paling *robust* untuk simulasi kalender jangka panjang (Park et al., 2021).

Dalam arsitektur kode KHGT Times, mesin ini diinisialisasi melalui sintaks pengikatan *library* Skyfield:

Python

```
self.load_obj = Loader(BASE_DIR, verbose=False)
self.ts = self.load_obj.timescale()
self.eph = self.load_obj('de441-new.bsp')
```

File `de441-new.bsp` (Binary Space Partition) tidak memuat rumus lintasan orbit, melainkan tabel koefisien polinomial yang sangat padat. Untuk membaca data ini secara akurat, sistem membutuhkan pemahaman mendalam tentang skala waktu relativistik.

2.1.1. Konsep Barycentric Dynamical Time (TDB) dan Terrestrial Time (TT)

Kesalahan fatal yang sering terjadi pada pemrograman falak pemula adalah menggunakan waktu sipil (seperti UTC atau Waktu Lokal) secara langsung sebagai parameter pengukur gerak planet. Waktu sipil (UTC) didasarkan pada rotasi bumi yang tidak konstan dan secara berkala dikoreksi dengan detik kabisat (*leap second*). Jika UTC digunakan murni untuk melacak posisi bulan 50 tahun ke depan, akan terjadi pergeseran (*drift*) yang membuat perhitungan meleset (Urban & Seidelmann, 2013).

Oleh karena itu, KHGT Times V7.4 secara arsitektural menggunakan `self.ts = self.load_obj.timescale()` untuk mengelola skala waktu. Ketika pengguna memasukkan

waktu lokal, misalnya melalui variabel `waktu_utc`, Skyfield secara internal melakukan transformasi hierarkis:

1. **UTC ke TT (Terrestrial Time):** Sistem menyerap tabel *leap second* (Delta-T / Delta T) untuk mengonversi waktu tidak beraturan menjadi waktu seragam di permukaan bumi.
2. **TT ke TDB (Barycentric Dynamical Time):** Karena DE441 berpusat pada pusat massa tata surya (*Solar System Barycenter*), waktu harus dikoreksi kembali berdasarkan efek relativitas umum Einstein. Saat bumi berada di *perihelion* (dekat dengan matahari), gravitasi yang lebih kuat memperlambat laju waktu (dilatasi waktu) relatif terhadap pengamat di luar tata surya. TDB adalah skala waktu independen yang mengoreksi efek relativistik tersebut.

Dalam skrip KHGT Times, hal ini terlihat jelas pada modul konversi di mana waktu dipaksa berjalan pada parameter TT yang absolut:

Python

```
t_start_conj = self.ts.tt_jd(t_eval_tt - 5.0)
t_end_conj = self.ts.tt_jd(t_eval_tt + 5.0)
```

Penggunaan `tt_jd` (Terrestrial Time dalam Julian Date) menjamin bahwa pencarian waktu *New Moon* (Ijtimak) tidak akan terdistorsi oleh perlambatan rotasi bumi di masa depan.

2.1.2. Pembacaan Vektor Posisi (X, Y, Z) Benda Langit

Setelah sistem berhasil menerjemahkan waktu pengguna ke dalam TDB, langkah berikutnya adalah mengekstraksi koordinat posisi benda langit dari file `.bsp`. Di dalam DE441, pergerakan benda langit direpresentasikan dalam deret polinomial Chebyshev. Deret ini memungkinkan kompresi data triliunan posisi planet menjadi ukuran file yang dapat diproses oleh RAM komputer standar (Rhodes, 2019).

Dalam KHGT Times V7.4, pembacaan vektor ini tidak dieksekusi secara manual, melainkan dipanggil secara dinamis:

Python

```
earth = self.eph['earth']
sun = self.eph['sun']
moon = self.eph['moon']
```

Ketika metode `earth.at(t).observe(moon)` dipanggil, mesin secara implisit melakukan proses perhitungan matriks tiga dimensi:

1. Mengevaluasi polinomial Chebyshev dari DE441 pada waktu t untuk mendapatkan koordinat Cartesian (X, Y, Z) Bumi dan Bulan relatif terhadap pusat massa tata surya.
2. Melakukan operasi pengurangan vektor: V_{moon} - V_{earth} untuk memperoleh posisi geosentris Bulan (berpusat di inti Bumi).

3. Mengonversi titik koordinat Cartesian spasial ini ke dalam asensio rekta (*Right Ascension*) dan deklinasi (*Declination*) astrometris.

Implementasi *Deep-Dive* ini memastikan bahwa paralaks, jarak aktual (radius vektor), dan pergeseran ekuator langit dihitung secara absolut menggunakan geometri ruang-waktu fisik, bukan lewat estimasi rumus trigonometri linier, sehingga menghasilkan perhitungan tinggi geosentris (Alt) dan elongasi (Eln) yang sangat presisi untuk parameter KHGT.

2.2. Koreksi Waktu dan Atmosfer Bumi

Dalam komputasi astrometri presisi tinggi, mengetahui posisi geometris murni (posisi ruang hampa) dari benda langit hanyalah separuh dari penyelesaian masalah. Separuh lainnya—yang sering kali menjadi sumber kesalahan fatal dalam perangkat lunak falak klasik—adalah mengoreksi posisi tersebut berdasarkan dinamika rotasi Bumi yang tidak stabil dan distorsi optik yang disebabkan oleh selubung atmosfer pelindung Bumi.

Bagi Kalender Hijriah Global Tunggal (KHGT) yang menuntut akurasi tingkat global, dan Kriteria MABIMS yang menuntut presisi toposentris lokal, pengabaian terhadap faktor-faktor fisik ini akan mendistorsi hasil hisab secara signifikan. Oleh karena itu, **KHGT Times V7.4** membangun lapisan koreksi fisik yang menginjeksi parameter lingkungan secara *real-time* ke dalam mesin perhitungannya.

2.2.1. Formulasi Delta-T (Delta T) dan Retardasi Rotasi Bumi

Secara intuitif, manusia menganggap bahwa satu hari selalu terdiri dari tepat 86.400 detik. Namun, secara astronomis, kecepatan rotasi Bumi pada sumbunya tidaklah konstan. Gesekan pasang surut kelautan (*tidal friction*) yang dipicu oleh gravitasi Bulan, perubahan distribusi massa Bumi akibat pencairan es kutub (*glacial isostatic adjustment*), hingga dinamika inti Bumi, menyebabkan rotasi Bumi melambat secara perlahan dari abad ke abad (Morrison & Stephenson, 2001).

Perlambatan ini melahirkan divergensi antara dua sistem waktu: **Universal Time (UT1)**, yang terikat pada rotasi aktual Bumi (waktu sipil/lokal), dan **Terrestrial Time (TT)**, yang merupakan waktu seragam absolut layaknya jam atom. Selisih antara kedua sistem waktu ini dikenal sebagai **Delta-T (Delta T)**, dengan formulasi dasar:

Delta T = TT - UT1

Pada masa Nabi Muhammad SAW hidup (sekitar abad ke-7 Masehi), nilai Delta T berada di kisaran belasan ribu detik. Saat ini (sekitar tahun 2026 M), nilainya berada di kisaran 69 hingga 70 detik, dan akan terus membengkak di masa depan (Espenak & Meeus, 2006). Jika sebuah program falak mencoba memprediksi posisi hilal 50 tahun ke depan tanpa mengoreksi Delta T, maka posisi Bulan yang dihasilkan akan meleset sejauh puluhan kilometer dari posisi aslinya di langit.

Dalam **KHGT Times V7.4**, pengembang tidak menggunakan rumus polinomial statis untuk menebak Delta T. Pustaka *Skyfield* yang menjadi inti (*core*) aplikasi ini secara otomatis

mengunduh dan menyinkronkan tabel parameter orientasi Bumi (IERS *Earth Orientation Parameters*) terbaru, yang meliputi nilai Delta T aktual dan terprediksi.

Proses ini diekstraksi secara elegan di dalam kode:

Python

```
# Ekstraksi Delta T untuk ditampilkan pada laporan
dt_val = self.ts.utc(year, month, start_d_iter).delta_t
```

Nilai `delta_t` ini secara otomatis diumpankan ke dalam konversi objek `self.ts.utc` menjadi `self.ts.tt_jd`, memastikan bahwa saat program mengiterasi pencarian waktu *sunset* (waktu sipil UT1) dan membandingkannya dengan waktu konjungsi/ijtimak (waktu dinamis TT), komparasi tersebut dilakukan pada *baseline* fisika yang ekuivalen.

2.2.2. Koreksi Refraksi Atmosfer, Presesi, dan Nutasi

Setelah masalah waktu teratasi, sistem dihadapkan pada masalah distorsi spasial. Saat pengamat (*observer*) melihat Bulan atau Matahari di ufuk (cakrawala), cahaya dari benda langit tersebut tidak berjalan lurus menuju mata pengamat. Cahaya tersebut harus menembus lapisan atmosfer Bumi yang memiliki kerapatan (densitas) berbeda-beda, sehingga memicu fenomena pembiasan optik atau **Refraksi Atmosfer**.

Menurut hukum optik (Snellius), refraksi akan mengangkat posisi semu (*apparent position*) benda langit menjadi lebih tinggi dari posisi geometris sebenarnya (*true position*). Pada saat Matahari tepat menyentuh garis ufuk (0 derajat), posisi geometrisnya sebenarnya sudah berada sekitar -34 menit busur (-0.56^\circ) di bawah ufuk (Schaefer, 1993). Dalam penentuan awal bulan menggunakan metode MABIMS (di mana hilal diamati dari permukaan Bumi / toposentrik), refraksi sangat menentukan apakah hilal berada di ketinggian 3 derajat atau kurang.

Besaran refraksi tidak konstan; ia sangat fluktuatif bergantung pada suhu (*temperature*) udara dan tekanan barometrik (*pressure*) di lokasi observasi. **KHGT Times V7.4** mengatasi kompleksitas ini dengan membuka parameter lingkungan agar dapat disesuaikan oleh pengguna atau ditarik melalui integrasi data cuaca *real-time*.

Hal ini terlihat pada struktur *Astrometry Engine* di Modul Visibilitas:

Python

```
temp = float(self.entry_vtemp.get())
pres = float(self.entry_vpres.get())
# ...
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)
```

Metode `altaz()` pada pustaka *Skyfield* secara internal akan menjalankan algoritma refraksi ekstensif (biasanya berbasis model *Saastamoinen* atau ekuivalennya) dengan mengalkulasi suhu dan tekanan atmosfer yang dimasukkan. Jika cuaca sangat dingin dan tekanan tinggi, refraksi akan semakin besar, yang secara teoritis dapat "mengangkat" hilal yang kritis menjadi terlihat.

Selain refraksi, mesin astrometri KHGT Times secara otomotis menghitung **Presesi** (putaran gasing sumbu Bumi dalam siklus 26.000 tahun) dan **Nutasi** (goyangan kecil pada sumbu akibat gravitasi Bulan) berdasarkan model mutakhir IAU 2000/2006 (Kaplan, 2005). Kalkulasi ini dijamin oleh fungsi `.apparent()` sebelum ekstraksi *Altitude* dan *Azimuth*, memastikan ekuator langit (*celestial equator*) direferensikan pada ekuinoks sejati (*true equinox of date*), bukan sekadar koordinat J2000 statis.

Kesempurnaan lapisan koreksi fisik dan optik ini menjadikan *output* kalkulasi KHGT Times tidak hanya akurat secara matematis di atas kertas, tetapi juga valid secara empiris saat dicocokkan dengan observasi di lapangan (*ground truth*).

2.3. Transformasi Sistem Koordinat

Data ephemeris murni dari JPL NASA memberikan posisi benda langit dalam kerangka ruang spasial tiga dimensi (X, Y, Z) yang berpusat pada titik berat tata surya (Barycentric). Namun, bagi seorang pengamat di permukaan bumi, posisi tersebut tidak memiliki makna praktis sampai ia ditransformasikan ke dalam sistem koordinat yang merujuk pada cakrawala lokal pengamat. Transformasi matematis ini adalah salah satu operasi yang paling menguras daya komputasi (*computationally expensive*) dalam ilmu falak (Meeus, 1998).

Dalam arsitektur **KHGT Times V7.4**, proses ini dieksekusi secara berjenjang. Sistem memecah komputasi koordinat menjadi dua fase utama: ekstraksi koordinat ekuatorial absolut, lalu memproyeksikannya ke dalam koordinat horizontal relatif.

2.3.1. Koordinat Ekuatorial (Right Ascension & Declination)

Sistem koordinat ekuatorial adalah proyeksi dari garis lintang dan bujur bumi ke bola langit (*celestial sphere*). Posisi benda langit dinyatakan dalam Asensio Rekta atau *Right Ascension* (alpha) dan Deklinasi atau *Declination* (delta).

Dalam penulisan kode astronomi standar, pengambilan data koordinat ini sering kali mengabaikan parameter *epoch* (titik referensi waktu pergeseran ekuinoks). Jika ekuinoks dibiarkan pada *default* standar J2000.0, maka posisi bulan dan matahari yang dihasilkan adalah posisi pada letak rasi bintang tahun 2000, yang tentu sudah bergeser akibat presesi bumi.

Untuk menjamin presisi KHGT, V7.4 secara eksplisit mendefinisikan *epoch* berjalan (*apparent equinox of date*). Perhatikan cuplikan kode dari modul kalkulasi visibilitas berikut:

Python

```
# Ekstraksi Geosentris dengan Epoch Waktu Evaluasi (t_eval)
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)
```

Pemanggilan `.apparent()` secara internal menerapkan koreksi aberasi tahunan (efek kecepatan cahaya relatif terhadap pergerakan bumi) dan pembelokan cahaya gravitasi (*gravitational deflection*). Selanjutnya, parameter `epoch=t_eval` memaksa matriks transformasi untuk mengkalkulasi ulang matriks rotasi bumi pada detik tersebut, menghasilkan nilai alpha dan delta yang murni (*true apparent*).

2.3.2. Konversi Matriks Geosentris ke Toposentris (Horizontal Altitude/Azimuth)

Parameter penentu dalam Kalender Hijriah Global Tunggal (KHGT) maupun MABIMS adalah posisi hilal di atas cakrawala (*horizon*), yang dinyatakan dalam Ketinggian (*Altitude*, a) dan Azimuth (A).

Terdapat perbedaan mendasar dalam syarat yurisprudensi:

1. **KHGT** mensyaratkan Ketinggian Geosentrik (Geocentric Altitude) ge 5^\circ. Artinya, posisi bulan ditarik lurus dari inti bumi, namun diproyeksikan ke horizon lokal pengamat saat *sunset* (Syamsuddin, 2021).
2. **MABIMS** mensyaratkan Ketinggian Toposentrik (Topocentric Altitude) ge 3^\circ. Artinya, posisi bulan ditarik dari mata pengamat di permukaan bumi, yang melibatkan koreksi paralaks diurnal (penurunan posisi semu bulan akibat jari-jari bumi).

Untuk menghitung MABIMS (Toposentrik), V7.4 memanfaatkan metode bawaan *Skyfield* `altaz()`, yang secara otomatis menghitung koreksi paralaks dan refraksi atmosfer:

Python

```
topo_earth = (earth + lokasi_obs).at(t_eval)
app_moon_topo = topo_earth.observe(moon).apparent()
alt_moon_topo_obj,              az_moon_topo_obj,              _              =
app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
```

Namun, inovasi paling brilian dalam arsitektur KHGT Times V7.4 terletak pada cara sistem menangani **Ketinggian Geosentrik lokal**. Pustaka standar astronomi umumnya tidak menyediakan fungsi instan untuk "menghitung altitude dari inti bumi tetapi merujuk pada horizon di permukaan bumi". Pengembang sistem ini merumuskan ulang trigonometri bola (*spherical trigonometry*) secara manual dari hulu ke hilir.

Langkah pertama adalah menghitung *Greenwich Apparent Sidereal Time* (GAST) lalu mengonversinya menjadi *Local Sidereal Time* (LST) dengan menambahkan bujur lokasi (lambda).

$LST = GAST + \lambda$

Langkah kedua, menghitung Sudut Jam atau *Hour Angle* (HA), yaitu jarak sudut objek dari meridian pengamat:

$HA = LST - \alpha$

Langkah ketiga, menerapkan hukum Cosinus bola untuk mendapatkan nilai Ketinggian (a). Jika phi adalah lintang geografis pengamat dan delta adalah deklinasi bulan, maka:

$\sin(a) = \sin(\delta)\sin(\varphi) + \cos(\delta)\cos(\varphi)\cos(HA)$

Perhatikan betapa persisnya implementasi matematika ini diterjemahkan ke dalam bahasa Python pada fungsi `calculate_kriteria_batas`:

Python

```
# Menghitung GAST dan LST (Local Sidereal Time)
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

# Hour Angle (HA) = LST - RA
ha_deg = lst_deg - (ra_h * 15.0)

# Konversi ke Radian untuk perhitungan Trigonometri
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)

# Arc-Sine untuk mendapatkan sudut Ketinggian (Altitude) Geosentris
alt_moon_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
```

Penggunaan fungsi pemotong rentang batas `max(-1.0, min(1.0, sin_alt))` merupakan teknik *Defensive Programming* yang memastikan mesin komputasi tidak mengalami *crash* (Error *Domain Math*) akibat *floating-point error* yang terkadang menghasilkan nilai sin > 1.0000000000001 (Virtanen et al., 2020).

Kemampuan sistem untuk mengekstraksi dan memisahkan nilai Geosentris (murni Trigonometri Bola) dan Toposentris (terkoreksi paralaks dan refraksi) pada *millisecond* yang sama inilah yang menjadikan **KHGT Times V7.4** memiliki validitas algoritmik yang kokoh dalam mewadahi ragam kriteria global dan regional secara simultan.

DAFTAR PUSTAKA BAB 2

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.

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.

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl-1907.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

BAB 3: FORMULASI VISIBILITAS HILAL MULTI-KRITERIA

Diskursus mengenai penentuan awal bulan Hijriah pada hakikatnya bermuara pada satu pertanyaan fundamental astrofisika: pada kondisi geometris dan optis seperti apa sebuah sabit bulan baru (hilal) dapat melampaui ambang batas visibilitas manusia? Seiring dengan perkembangan astronomi observasional, kriteria visibilitas tidak lagi bertumpu pada estimasi umur bulan semata, melainkan berevolusi menjadi pemodelan matematis multivariabel.

Dalam **KHGT Times V7.4**, modul visibilitas hilal didesain sebagai instrumen analitik komprehensif yang tidak berafiliasi tunggal pada satu kriteria, melainkan memproses konversi ruang-waktu untuk mengevaluasi parameter Geosentris (kriteria global) dan Toposentris (kriteria regional/lokal) secara simultan.

3.1. Anatomi Ketinggian dan Elongasi Bulan

Dua parameter paling esensial dalam pemodelan visibilitas hilal adalah Ketinggian (*Altitude*, disimbolkan dengan a atau h) dan Jarak Sudut/Elongasi (*Elongation*, disimbolkan dengan E). Ketinggian menentukan seberapa jauh hilal berada di atas ufuk untuk menghindari atenuasi atmosfer yang tebal, sedangkan elongasi berbanding lurus dengan lebar sabit (*crescent width*) dan kecerahan permukaan bulan (*surface brightness*).

Namun, pengukuran kedua parameter ini menuntut titik referensi asal (*origin of reference*) yang presisi. Perbedaan pengambilan titik referensi inilah yang membedakan arsitektur Kalender Hijriah Global Tunggal (KHGT) dengan kriteria regional seperti Neo MABIMS.

3.1.1. Perhitungan Geocentric Altitude (Standar KHGT)

Kriteria KHGT hasil Kongres Istanbul 2016 menetapkan bahwa awal bulan dimulai jika Ketinggian Hilal ge 5^\circ dan Elongasi ge 8^\circ. Parameter ini mensyaratkan pengukuran **Geosentris** (*Geocentric*), yakni titik referensi pengukuran ditarik dari pusat inti bumi. Pendekatan ini ideal untuk kalender global karena mengeliminasi efek paralaks diurnal yang bervariasi secara ekstrem bergantung pada posisi lintang pengamat di permukaan bumi.

Dalam komputasi KHGT Times V7.4, ekstraksi *Geocentric Altitude* tidak menggunakan fungsi *built-in* observasi lokal karena perpustakaan standar umumnya memutar koordinat langsung ke permukaan bumi. Oleh karena itu, sistem melakukan konstruksi trigonometri bola secara mandiri. Berdasarkan kode sumber, perhitungan ini dieksekusi melalui beberapa tahapan matematis saat *sunset*:

1. Sistem menghitung *Greenwich Apparent Sidereal Time* (GAST) untuk mendapatkan rotasi bumi absolut, yang kemudian dikonversi menjadi *Local Sidereal Time* (LST) dengan menambahkan bujur lokasi (lambda).

2. Posisi ekuatorial bulan, yakni Asensio Rekta (alpha) dan Deklinasi (delta), diekstraksi dari pusat bumi (`earth.at(t_sunset)`).
3. Sistem menghitung Sudut Jam (*Hour Angle*, HA) menggunakan selisih antara LST dan alpha.
4. Ketinggian geosentris (a_{geo}) kemudian dihitung menggunakan teorema kosinus untuk segitiga bola:

$$\sin(a_{geo}) = \sin(\delta)\sin(\varphi) + \cos(\delta)\cos(\varphi)\cos(HA)$$

Implementasi pada kode (misalnya pada fungsi `calculate_kriteria_batas`):

Python

```
gast = t_sunset.gast
lst_deg = (gast * 15.0) + lon

ra_h = ra_moon.hours.item()
dec_r = dec_moon.radians.item()
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))))
```

Penggunaan fungsi *clamping* `max(-1.0, min(1.0, sin_alt))` merupakan teknik *defensive programming* yang sangat krusial untuk mencegah *domain error* pada fungsi arc-sine (`math.asin`) akibat anomali *floating-point* pada nilai yang mendekati zenith mutlak.

3.1.2. Perhitungan Topocentric Altitude (Standar MABIMS)

Berbeda dengan KHGT, kriteria regional seperti Neo MABIMS (diadopsi oleh Indonesia, Malaysia, Brunei, Singapura pada 2021) mensyaratkan Ketinggian ge 3^\circ dan Elongasi ge 6.4^\circ secara **Toposentris** (*Topocentric*). Artinya, posisi diukur tepat dari mata pengamat di atas permukaan bumi.

Pengukuran toposentris wajib memperhitungkan dua koreksi fisis utama:

1. **Paralaks Ekuatorial Horizontal (P):** Penurunan posisi semu objek karena pengamat tidak berada di pusat bumi. Jarak bumi-bulan yang relatif dekat (rata-rata 384.400 km) menyebabkan paralaks bulan sangat besar (hampir mencapai 1^\circ).
2. **Refraksi Atmosfer (R):** Pembelokan lintasan cahaya oleh kepadatan atmosfer yang mengangkat posisi semu bulan mendekati horizon.

Pada arsitektur KHGT Times V7.4, komputasi toposentris dilimpahkan sepenuhnya kepada metode `altaz()` pada pustaka *Skyfield*, dengan menginjeksi variabel suhu dan tekanan udara secara dinamis:

Python

```
topo_earth = (earth + lokasi_obs).at(t_eval)
app_moon_topo = topo_earth.observe(moon).apparent()
alt_moon_topo_obj, az_moon_topo_obj, _ =
app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
alt_moon_topo = alt_moon_topo_obj.degrees
```

Fungsi ini menjamin bahwa ketinggian yang dihasilkan adalah *Apparent Topocentric Altitude*, yang merepresentasikan visibilitas optis paling realistis bagi seorang perukyat (*observer*) di lapangan.

3.1.3. Formulasi Sudut Elongasi dan Fraksi Iluminasi (Fase Bulan)

Elongasi (E) adalah jarak busur di sepanjang lingkaran besar yang menghubungkan titik pusat piringan matahari dan bulan. Dalam konteks observasi hilal, semakin besar elongasi, maka:

1. Semakin tebal bentuk sabit yang terbentuk (*crescent width*).
2. Semakin jauh hilal dari area *twilight glare* (cahaya silau senja) matahari, sehingga kontras optiknya meningkat.

Baik KHGT (8 derajat) maupun MABIMS (6.4 derajat) bersandar pada elongasi ini sebagai substitusi dari parameter "umur bulan" yang dinilai terlalu rentan terhadap anomali gerak bujur bulan. Dalam KHGT Times V7.4, elongasi dicari melalui fungsi jarak sudut vektor (`separation_from`):

Python

```
elong_geo = app_sun_geo.separation_from(app_moon_geo).degrees
```

Lebih lanjut, algoritma perangkat lunak ini juga mengekstraksi **Fraksi Iluminasi** (k), yaitu persentase luasan piringan bulan yang tersinari oleh matahari dan menghadap ke bumi. Secara matematis, fraksi iluminasi bulan berkaitan erat dengan sudut fase (i) dan elongasi geometris:

$$k = (1 + \cos(i)) / 2$$

Di mana sudut fase mempresentasikan sudut Bumi-Bulan-Matahari. Pada KHGT Times, persentase ini diperoleh menggunakan turunan dari modul almanak NASA:

Python

```
illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_eval)
illumination = illum_val.item() * 100.0
```

Besaran fraksi iluminasi (yang umumnya berada di kisaran 0.5\% - 1.2\% untuk hilal awal bulan) sangat krusial dalam kajian fotometri dan menjadi basis dari perhitungan kecerahan permukaan (*surface brightness*) pada limit visibilitas optik.

3.2. Komparasi Algoritmik Kriteria Global

Dalam perancangan sistem **KHGT Times V7.4**, tantangan utama bukan terletak pada perhitungan posisi benda langit itu sendiri, melainkan pada bagaimana sistem mengevaluasi kriteria visibilitas yang berbeda secara algoritmik dan simultan. Sub-bab ini akan membedah logika komparasi antara kriteria global (KHGT) dengan kriteria visibilitas lainnya dari sudut pandang efisiensi kode dan akurasi geometris.

3.2.1. Matriks Kriteria Visibilitas dalam Engine KHGT

Sistem harus mampu membedakan antara kriteria berbasis *astrometri murni* (geosentris) dan kriteria berbasis *observasi atmosfer* (toposentris). KHGT, sebagai kriteria global tunggal, menggunakan parameter yang berbeda secara fundamental dari kriteria regional seperti MABIMS atau kriteria klasik Odeh.

Berikut adalah logika *conditional branching* yang diterapkan dalam fungsi `check_visibility_criteria()` pada modul inti aplikasi:

Parameter	KHGT (Global Tunggal)	MABIMS (Regional)	Odeh (Astrofisika)
Pusat Acuan	Geosentris (Inti Bumi)	Toposentris (Permukaan)	Toposentris
Tinggi Hilal (h)	h ≥ 5°	h ≥ 3°	Variabel (V)
Elongasi (epsilon)	ε ≥ 8°	ε ≥ 6,4°	Variabel (L)
Batas Wilayah	Global (*Mainland*)	Kawasan Regional	Global
Refraksi	Diabaikan (Geosentris)	Dihitung (Toposentris)	Dihitung

3.2.2. Algoritma Transformasi Koordinat Simultan

Untuk melakukan komparasi yang adil, sistem menjalankan dua alur kalkulasi paralel saat matahari terbenam (*sunset*) di lokasi referensi. Kode Python dalam modul ini melakukan transformasi koordinat dari objek `Apparent` yang dihasilkan oleh *Skyfield* menjadi dua skalar yang berbeda:

1. **Vektor Geosentris:** Untuk validasi KHGT. Jarak sudut dihitung dari pusat Bumi ke pusat Bulan dan Matahari tanpa memperhitungkan paralaks dan refraksi atmosfer.

2. **Vektor Toposentris:** Untuk validasi MABIMS/Lokal. Posisi bulan dikoreksi terhadap elevasi pengamat dan pembiasan atmosfer (+0.833^\circ pada ufuk).

Python

```
# Potongan Logika Komparasi dalam KHGT Times
def evaluate_comparison(self, t, observer_loc):
    # 1. Jalur Geosentris (KHGT)
    sun_geo = earth.at(t).observe(sun).apparent()
    moon_geo = earth.at(t).observe(moon).apparent()
    alt_geo, _, _ = moon_geo.altaz()
    elong_geo = moon_geo.separation_from(sun_geo)

    # 2. Jalur Toposentris (MABIMS)
    topo_pos = observer_loc.at(t).observe(moon).apparent()
    alt_topo, az_topo, _ = topo_pos.altaz()

    # Hasil evaluasi dikembalikan dalam bentuk Boolean
    is_khgt_valid = (alt_geo.degrees >= 5.0 and elong_geo.degrees >= 8.0)
    is_mabims_valid = (alt_topo.degrees >= 3.0 and
self.calculate_elongation_topo() >= 6.4)

    return is_khgt_valid, is_mabims_valid
```

3.2.3. Analisis Deviasi: Mengapa KHGT Lebih Stabil Secara Algoritmik?

Melalui ribuan simulasi pengujian pada Modul 38 (Auto-Builder), ditemukan bahwa kriteria geosentris KHGT memiliki tingkat stabilitas numerik yang lebih tinggi dibandingkan kriteria toposentris. Hal ini disebabkan oleh:

- **Eliminasi Variabel Elevasi:** Kriteria toposentris sangat sensitif terhadap ketinggian tempat pengamat di atas permukaan laut. Di lokasi yang sama, pengamat di puncak gunung dan di pantai akan mendapatkan nilai h yang berbeda, yang dapat menyebabkan "anomali status" awal bulan dalam sistem komputer.
- **Konsistensi Global:** Karena KHGT menggunakan *Geocentric Position*, status awal bulan tidak akan berubah-ubah hanya karena perbedaan model refraksi atmosfer yang digunakan dalam kode (misalnya antara model Saemundsson vs model Cassini).

Integrasi komparasi algoritmik ini memungkinkan pengguna KHGT Times V7.4 untuk melihat secara transparan kapan sebuah awal bulan disepakati secara global namun belum memenuhi kriteria regional, atau sebaliknya.

3.3. Struktur Data Ephemeris DE441 dalam Sistem

Keakuratan sebuah sistem penanggalan astronomi sangat bergantung pada kualitas data posisi benda langit yang digunakan sebagai fondasi komputasi. Dalam arsitektur **KHGT Times V7.4**, mesin kalkulasi tidak lagi mengandalkan seri trigonometri sederhana (seperti algoritma ELP2000-85), melainkan telah terintegrasi secara penuh dengan **NASA JPL Development Ephemeris**

DE441. Sub-bab ini akan membahas mengapa DE441 dipilih dan bagaimana struktur data tersebut diorganisir di dalam sistem.

3.3.1. Karakteristik dan Keunggulan DE441

JPL DE441 merupakan hasil integrasi numerik dari orbit planet-planet dan bulan yang sangat presisi, mencakup rentang waktu yang sangat luas, yaitu dari tahun -13201 hingga +17191. Pemilihan DE441 dibandingkan DE440 (versi standar saat ini) didasarkan pada kebutuhan sistem KHGT untuk melakukan simulasi historis maupun proyeksi masa depan dalam skala ribuan tahun tanpa kehilangan akurasi desimal.

Parameter kunci yang membuat DE441 menjadi standar emas dalam aplikasi ini adalah:

- **Integrasi Dinamis:** Menggunakan observasi terbaru dari wahana antariksa, *Lunar Laser Ranging*, dan *Very Long Baseline Interferometry* (VLBI).
- **Skala Waktu:** Menggunakan skala waktu koordinat (*Barycentric Dynamical Time* - TDB) yang konsisten secara relativistik.
- **Resolusi Data:** Menyimpan koefisien polinomial Chebyshev yang memungkinkan interpolasi posisi pada tingkat mikrosekon busur.

3.3.2. Implementasi File `.bsp` dalam Arsitektur Python

Dalam folder `/data/ephemeris/` pada paket aplikasi KHGT Times, terdapat berkas biner berformat `.bsp` (Binary SPK). Sistem menggunakan pustaka *Skyfield* sebagai *wrapper* untuk memuat data ini secara efisien tanpa membebani memori RAM secara berlebihan melalui teknik *memory-mapping*.

Berikut adalah skema pemanggilan dan struktur pemuatan data dalam sistem:

Python

```
from skyfield.api import load

# Memuat data ephemeris DE441 dari direktori lokal
eph = load('data/ephemeris/de441.bsp')

# Mendefinisikan objek langit sebagai titik referensi (Barycentric)
sun = eph['sun']
moon = eph['moon']
earth = eph['earth']

def get_precise_position(jd):
    """
    Mengambil posisi presisi Bulan terhadap Bumi pada Julian Date (jd)
    menggunakan data polinomial dari DE441.
    """
    t = load.timescale().tt_jd(jd)
    astrometric = earth.at(t).observe(moon)
    return astrometric
```

3.3.3. Penanganan Interpolasi dan Presisi Numerik

Struktur data DE441 dalam sistem dikelola melalui proses dua tahap untuk menjamin kecepatan *rendering* pada antarmuka GUI:

1. **Interpolasi Polinomial:** Sistem mengekstrak koefisien Chebyshev dari file `.bsp` untuk interval waktu tertentu. Proses ini memungkinkan KHGT Times menghitung posisi antara dua titik data tanpa harus menyimpan koordinat per detik yang akan menghabiskan ruang penyimpanan terabita.
2. **Koreksi Aberasi dan Cahaya:** Walaupun data mentah DE441 bersifat geometris murni, sistem secara otomatis menerapkan koreksi waktu tempuh cahaya (*light-time correction*) dan aberasi cahaya saat fungsi `.apparent()` dipanggil, memastikan posisi yang ditampilkan adalah posisi semu yang relevan untuk pengamatan hilal.

3.3.4. Efisiensi Database JSON dari Data Ephemeris

Untuk mempercepat proses pencarian awal bulan global (PKG 1 dan PKG 2), sistem tidak melakukan kalkulasi *real-time* dari DE441 untuk setiap permintaan pengguna. Sebaliknya, sistem membangun **Lookup Table (LUT)** dalam format JSON yang berisi indeks konjungsi (ijtimak) selama 100 tahun ke depan yang sudah dipra-hitung menggunakan DE441. Hal ini memungkinkan pencarian status tanggal hijriah dilakukan dalam hitungan milidetik.

3.4. Arsitektur Pemrosesan Data Waktu (TT, UT1, dan UTC)

Dalam komputasi astronomi tingkat tinggi, waktu bukanlah sebuah variabel statis yang hanya mengikuti jam dinding. Pergerakan benda langit dihitung dalam skala waktu yang seragam, sementara kehidupan manusia di bumi diatur oleh rotasi planet yang tidak beraturan. **KHGT Times V7.4** mengelola kompleksitas ini melalui arsitektur pemrosesan waktu tiga lapis untuk memastikan perhitungan awal bulan global tidak mengalami pergeseran detik yang fatal.

3.4.1. Terrestrial Time (TT) sebagai Basis Komputasi

Seluruh kalkulasi posisi matahari dan bulan dalam mesin KHGT dilakukan menggunakan **Terrestrial Time (TT)**. TT adalah skala waktu dinamis yang bersifat kontinu dan seragam, tidak terpengaruh oleh pelambatan rotasi bumi. Skala ini merupakan penerus dari *Ephemeris Time* (ET) yang digunakan dalam literatur falak klasik.

Sistem menggunakan TT sebagai variabel independen dalam persamaan gerak karena posisi benda langit dalam ephemeris DE441 dipetakan terhadap waktu dinamis. Jika sistem dipaksakan menggunakan jam sipil (UTC) untuk menghitung posisi bulan, maka akan terjadi kesalahan posisi sebesar beberapa kilometer akibat akumulasi perlambatan rotasi bumi selama berabad-abad.

3.4.2. Penanganan Delta-T (Delta T) dan UT1

Masalah utama dalam falak digital adalah menghubungkan "kapan peristiwa terjadi di langit" dengan "di mana peristiwa itu terlihat di bumi". Untuk itu, sistem membutuhkan **UT1**, yaitu skala waktu yang didasarkan tepat pada rotasi bumi terhadap bintang-bintang.

Selisih antara waktu seragam (TT) dan waktu rotasi bumi (UT1) disebut sebagai **Delta T (Delta-T)**:

$$\Delta T = TT - UT1$$

KHGT Times V7.4 menangani ΔT secara dinamis dengan memuat tabel prediksi dari *International Earth Rotation and Reference Systems Service* (IERS). Untuk perhitungan historis, sistem menggunakan model polinomial Morrison dan Stephenson (2004), sedangkan untuk proyeksi masa depan, sistem mengimplementasikan ekstrapolasi linear agar penentuan *sunset* (matahari terbenam) tetap akurat hingga tahun 1500 Hijriah.

3.4.3. Konversi ke Coordinated Universal Time (UTC)

Setelah posisi benda langit tervalidasi secara geometris dalam skala TT dan dikoreksi terhadap posisi geografis pengamat melalui UT1, sistem melakukan konversi akhir ke **Coordinated Universal Time (UTC)**. UTC adalah waktu standar yang digunakan oleh sistem operasi komputer dan perangkat GPS.

Perbedaan antara UTC dan UT1 dijaga agar tidak lebih dari 0,9 detik melalui mekanisme *Leap Second* (detik kabisat). Arsitektur kode dalam KHGT Times memastikan bahwa setiap input waktu lokal dari pengguna selalu melewati rangkaian transformasi berikut sebelum diproses:

1. **Local Time** rightarrow disesuaikan dengan Timezone menjadi **UTC**.
2. **UTC** rightarrow ditambah sekon kabisat menjadi **International Atomic Time (TAI)**.
3. **TAI** rightarrow ditambah konstanta 32,184 detik menjadi **TT**.

3.4.4. Implementasi Kode pada Timescale Object

Pustaka *Skyfield* dalam sistem KHGT membungkus logika ini ke dalam objek `Timescale`. Hal ini memungkinkan pengembang untuk mengekstrak berbagai format waktu dari satu objek tunggal tanpa risiko kesalahan konversi manual.

Python

```
from skyfield.api import load

ts = load.timescale()
t = ts.now()

# Ekstraksi berbagai skala waktu untuk validasi
print(f"Waktu TT (Dinamis): {t.tt}")
print(f"Waktu UT1 (Rotasi): {t.ut1}")
```

```
print(f"Nilai Delta-T Aktif: {t.delta_t} detik")
```

Dengan arsitektur pemrosesan waktu yang rigid ini, KHGT Times V7.4 mampu menjamin bahwa detik terjadinya konjungsi (ijtimak) dan detik matahari terbenam sinkron secara matematis, memberikan fondasi yang kokoh bagi unifikasi kalender Islam global.

DAFTAR PUSTAKA BAB 3

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10–20. https://doi.org/10.1109/MCSE.2007.50

Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. VSOP 87 solutions. *Astronomy and Astrophysics*, 202, 309–315.

Kaplan, G. H. (2005). *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models* (USNO Circular No. 179). United States Naval Observatory.

Morrison, L. V., & Stephenson, F. R. (2004). Historical values of the Earth's clock error Delta T and the calculation of eclipses. *Journal for the History of Astronomy*, 35(3), 327–336. https://doi.org/10.1177/002182860403500305

Odeh, M. S. (2006). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).

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

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl:1907.024.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 4: ALGORITMA PELACAKAN TITIK PERTAMA KHGT (GLOBAL SCANNING)

Perbedaan paling radikal antara sistem penanggalan zonal (seperti MABIMS) dan sistem penanggalan global (seperti KHGT) terletak pada jangkauan observasi astronomisnya. Sistem zonal hanya mensyaratkan terpenuhinya kriteria visibilitas di dalam batas-batas administratif suatu kawasan. Sebaliknya, Kalender Hijriah Global Tunggal (KHGT) mewajibkan mesin komputasinya untuk mensimulasikan visibilitas hilal di seluruh permukaan bumi secara terintegrasi (Syamsuddin, 2021).

Tugas ini melahirkan sebuah tantangan komputasional yang masif: Bagaimana sistem dapat memastikan bahwa hilal telah—atau belum—memenuhi kriteria KHGT di "suatu tempat" di daratan bumi, untuk kemudian mentransfer status tersebut ke seluruh dunia? **KHGT Times V7.4** menjawab tantangan ini melalui arsitektur Modul 5: *Titik Pertama KHGT (Daratan Utama)*.

4.1. Konsep Parameter Kalender Global (PKG)

Muktamar Istanbul 2016 menetapkan dua lapisan algoritma atau *Parameter Kalender Global (PKG)* yang beroperasi secara *cascading* (berurutan) (Odeh, 2016). Perangkat lunak KHGT Times secara ketat menerjemahkan parameter ini ke dalam logika kondisional (`if-else`) yang bertingkat.

4.1.1. Syarat PKG 1: Terpenuhi Sebelum 00:00 UTC

Parameter pertama (PKG 1) menetapkan bahwa bulan baru (tanggal 1 Hijriah) dimulai keesokan harinya secara serentak di seluruh dunia apabila syarat visibilitas (Tinggi ge 5^\circ dan Elongasi ge 8^\circ) terpenuhi di *suatu titik di bumi*, dengan syarat kritis: waktu terbenamnya matahari (*sunset*) di lokasi tersebut harus terjadi **sebelum pukul 00:00 UTC (GMT)**.

Pukul 00:00 UTC digunakan sebagai batas demarkasi karena ia bersesuaian dengan pukul 12:00 siang di Garis Batas Penanggalan Internasional (*International Date Line*/IDL). Jika kriteria terpenuhi sebelum batas ini, maka sebagian besar populasi umat Islam dunia masih berada pada "hari yang sama" atau belum memasuki waktu fajar, sehingga penyatuan hari raya dapat direalisasikan tanpa membatalkan puasa yang sedang berjalan.

Dalam kode KHGT Times V7.4, batas waktu ini dikonstruksi secara eksplisit menggunakan metode waktu absolut:

Python

```
# 'dt_local_ephem' adalah variabel waktu lokal (waktu sipil) di titik yang
berhasil memenuhi kriteria KHGT
y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()

# Sistem membangun variabel 'batas_00_utc_ephem' secara matematis
```

```
# dengan mengambil tanggal lokal dan menambahkan 1 hari (+ 1.0) pada jam
00:00:00
batas_00_utc_ephem = ephem.Date(ephem.Date((int(y_l), int(m_l), int(d_l), 0,
0, 0)) + 1.0)

# Evaluasi PKG 1
if pt['sunset_utc'] < batas_00_utc_ephem:
    # Kesimpulan: Bulan baru dimulai BESOK (H+1)
```

4.1.2. Syarat PKG 2: Anomali Benua Amerika dan Referensi Fajar Gisborne

Jika PKG 1 gagal—misalnya hilal baru memenuhi kriteria di benua Amerika setelah pukul 00:00 UTC—maka KHGT menetapkan mekanisme penyelamatan (PKG 2) untuk memastikan apakah penanggalan masih bisa dipertahankan serentak, atau harus ditunda (*istikmal*/digenapkan 30 hari).

PKG 2 mengatur bahwa jika kriteria KHGT baru terpenuhi setelah 00:00 UTC, dan lokasi pemenuhan kriteria tersebut berada di benua Amerika, maka kita harus memeriksa waktu terjadinya Ijtimak/Konjungsi (*New Moon*). **Bulan baru dapat dimulai keesokan harinya JIKA DAN HANYA JIKA Ijtimak terjadi SEBELUM terbitnya fajar (*fajr*) di Gisborne, Selandia Baru.** Gisborne (Bujur 178.0176^\circ BT) dipilih secara konsensus karena merupakan salah satu kota daratan pertama di dunia yang paling awal menyongsong fajar. Jika Ijtimak (awal siklus bulan) terjadi setelah fajar menyingsing di Gisborne, maka umat Islam di Selandia Baru telah terlanjur memulai hari puasanya tanpa adanya status bulan baru. Oleh karena itu, jika hal ini terjadi, masuknya bulan baru harus ditunda ke hari lusa untuk menghindari kebingungan teologis.

Perhatikan keanggunan logika *fail-safe* ini dalam kode sistem:

Python

```
# Jika PKG 1 Gagal, maka masuk ke PKG 2
else:
    gisborne = ephem.Observer()
    gisborne.lat, gisborne.lon = math.radians(-38.6623),
math.radians(178.0176)
    gisborne.elevation, gisborne.pressure = 0, 0
    # Parameter -18 derajat untuk Astronomical Twilight (Fajar Shadiq)
    gisborne.horizon = '-18'
    matahari_g = ephem.Sun()

    # Mencari waktu terbit fajar di Gisborne
    gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
    fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)

    # Evaluasi Geografis dan Astronomis PKG 2
    if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
        # Kesimpulan: PKG 2 Terpenuhi. Bulan baru dimulai BESOK (H+1)
    else:
        # Kesimpulan: PKG 2 Gagal. Bulan baru ditunda LUSA (H+2)
```

4.2. Metode *Brute-Force* dan Iterasi Ephemeris

Dalam disiplin ilmu komputer dan fisika komputasi, pendekatan *brute-force* (pencarian ekshaustif) umumnya dihindari apabila terdapat algoritma analitik yang lebih efisien, karena beban kompleksitas waktu komputasinya yang tinggi (mathcal{O}(n)) (Newman, 2012). Namun, dalam konteks pemindaian (*scanning*) matlak global untuk KHGT, algoritma prediktif linear tidak dapat digunakan. Hal ini disebabkan oleh topografi bumi yang tidak beraturan, batas-batas daratan (*mainland*) yang asimetris, serta fluktuasi refraksi atmosfer lokal.

Satu-satunya cara yang valid secara saintifik untuk memastikan di titik mana hilal pertama kali memenuhi kriteria di bumi adalah dengan melakukan komputasi *iterative trial-and-error* pada seluruh koordinat kota di dunia satu per satu, detik demi detik, pada hari terjadinya konjungsi. Untuk menangani operasi masif ini tanpa menyebabkan *system crash*, **KHGT Times V7.4** merancang arsitektur iterasi yang memadukan kecepatan bahasa C murni dengan keakuratan data NASA.

4.2.1. Peran `ephem.Observer` dalam Pemindaian Ratusan Kota

Sebagaimana diuraikan pada bab sebelumnya, *library Skyfield* memiliki akurasi absolut karena membaca langsung data vektor DE441. Namun, memanggil fungsi pencarian waktu terbenam matahari (*discrete event finding*) menggunakan fungsi `almanac.sunrise_sunset()` milik *Skyfield* secara berulang pada ratusan kota akan menghasilkan *overhead* komputasi yang masif dan lambat.

Untuk mengatasi leher botol (*bottleneck*) performa ini, KHGT Times menggunakan arsitektur komputasi *hybrid*. Modul ini mendelegasikan tugas pencarian *sunset* lokal yang brutal kepada pustaka *PyEphem* (`ephem`), yang merupakan *wrapper* langsung dari *library* astronomi *XEphem* berbasis bahasa C (Downey, 2011).

Perhatikan bagaimana iterasi ini disusun dalam kode sumber `calculate_first_point`:

Python

```
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 # Melewati anomali kutub (Midnight Sun / Polar Night)
```

Metode `ephem.Observer()` memungkinkan sistem melakukan instansiasi objek pengamat dalam memori secara instan. Fungsi `next_setting(matahari)` dijalankan ratusan kali per detik pada basis data `CITY_DB` yang merentang dari Timur Jauh (Oseania) hingga ujung Barat (Benua Amerika). Blok `try-except` di atas merupakan desain algoritma pertahanan (*defensive algorithm*) untuk menyaring lokasi-lokasi di lintang ekstrem (Kutub Utara/Selatan) yang mengalami anomali matahari tidak terbenam (*Midnight Sun*), sehingga sistem tidak mengalami *infinite loop*.

4.2.2. Sinkronisasi Data *Sunset* Lokal dengan *New Moon* (Ijtimak)

Setelah titik *sunset* di suatu kota berhasil ditemukan secara cepat oleh `ephem`, kalkulasi belum selesai. Sistem KHGT mensyaratkan perhitungan Ketinggian dan Elongasi harus diekstrak secara **Geosentris** (berpusat di inti bumi) pada detik di mana *sunset* lokal itu terjadi (Odeh, 2004). Di sinilah terjadi sinkronisasi atau *handshake* antara mesin `ephem` dan mesin `Skyfield`.

Kode V7.4 melakukan transformasi objek waktu secara melintang dari format *PyEphem* ke format waktu dinamis *Skyfield* (TT/TDB) untuk dievaluasi terhadap posisi *New Moon* (Ijtimak).

Python

```
# Menerjemahkan waktu sunset dari ephem kembali ke format UTC murni
import pytz
t_sunset_sky                                                                    =
self.ts.from_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))

# Mengekstraksi posisi Geosentris menggunakan Skyfield pada detik sunset
tersebut
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()

# Perhitungan Elongasi Geosentrik murni
elong_geo = s_app_sky.separation_from(m_app_sky).degrees
```

Proses sinkronisasi silang (*cross-synchronization*) ini memastikan bahwa KHGT Times V7.4 tidak mengorbankan akurasi demi kecepatan. Waktu terjadinya ijtimak global (t_{ijtimak}) dibandingkan dengan waktu *sunset* lokal di ratusan titik tersebut untuk mencari di daratan (negara/kota) mana nilai `alt_geo >= 5.0` dan `elong_geo >= 8.0` pertama kali bernilai `True`.

Keberhasilan algoritma ini melacak *titik nol* pergerakan fase bulan di daratan utama mematahkan keraguan para pakar astronomi klasik mengenai keandalan komputasi pencarian global (*global search constraint*) dalam pembentukan Kalender Hijriah Global Tunggal (Syamsuddin, 2021). Dengan metode *brute-force* yang diakselerasi ini, sebuah komputer personal modern mampu mereplikasi observasi matlak global yang seharusnya membutuhkan ratusan observatorium fisik yang tersebar di seluruh dunia.

4.3. Implementasi Kode Pelacakan Daratan Utama (*Mainland*)

Konsep visibilitas global mensyaratkan bahwa hilal harus dapat dirukyat (atau memenuhi kriteria hisab) di suatu tempat di permukaan bumi. Namun, secara yurisprudensi dan kelayakan sosiologis,

para pakar astronomi Islam sepakat bahwa titik pemenuhan kriteria tersebut tidak boleh berada di wilayah kepulauan terpencil yang tidak berpenghuni atau terisolasi di tengah samudra (Ilyas, 1994). Jika titik pertama visibilitas jatuh di tengah Samudra Pasifik, memulai kalender Hijriah berdasarkan titik tersebut dianggap tidak merepresentasikan aktivitas kehidupan umat Islam (Syamsuddin, 2021).

Oleh karena itu, perangkat lunak **KHGT Times V7.4** menyematkan sebuah lapisan algoritma filter spasial (*spatial filter algorithm*) untuk memastikan bahwa titik yang dilacak adalah bagian dari daratan utama (*mainland*).

4.3.1. Filter Zona Pengecualian (Kepulauan Terpencil)

Untuk mencegah terjadinya *false positive*—di mana program mendeteksi hilal valid di lokasi yang tidak memenuhi syarat *mainland*—sistem secara arsitektural menyaring basis data (`CITY_DB`) sebelum melakukan instansiasi objek pengamat (`ephem.Observer()`).

Perhatikan implementasi penyaringan zona dalam kode fungsi `calculate_first_point`:

Python

```
titik_hasil = []
# Mengabaikan pulau kecil / wilayah timur jauh yang bukan daratan utama (Mainland)
zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "Maluku", "Maluku Utara", "NTT"]

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  # Melewati zona terlarang
```

Penggunaan perintah `continue` pada iterasi *dictionary* di atas bertindak sebagai gerbang logika (*logic gate*) yang sangat efisien (mathcal{O}(1) *lookup time*). Sistem memangkas beban komputasi dengan tidak melakukan kalkulasi *sunset* dan geometri bulan-matahari pada kawasan Oseania (seperti Kiritimati, Samoa, dan Tonga) maupun kepulauan kecil di Indonesia bagian timur. Dengan demikian, titik pertama (PKG 1) yang nantinya dideklarasikan oleh program dipastikan selalu berada di atas daratan benua, merujuk pada standar populasi berkesinambungan (*continuous population standard*).

4.3.2. Penyusunan Logika Komparasi Waktu Gisborne (Selandia Baru)

Ketika titik visibilitas hilal pertama gagal ditemukan sebelum pukul 00:00 UTC (PKG 1 gagal), algoritma KHGT akan jatuh pada pengondisian PKG 2. Skenario terburuk ini biasanya terjadi ketika kriteria ge 5^\circ dan ge 8^\circ baru tercapai di Benua Amerika. Untuk memvalidasi kondisi ini, sistem harus memastikan bahwa saat Ijtimak (konjungsi) terjadi, fajar (waktu subuh)

belum menyingsing di belahan bumi paling timur, yang secara konsensus diwakili oleh kota Gisborne, Selandia Baru (Odeh, 2016).

Dalam skrip V7.4, pemodelan ini dikonstruksi secara dinamis. Sistem tidak menggunakan waktu fajar statis, melainkan menghitung *Astronomical Twilight* (fajar shadiq) di Gisborne secara *real-time* tepat pada hari terjadinya konjungsi:

Python

```
# Setup stasiun virtual di Gisborne
gisborne = ephem.Observer()
gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
gisborne.elevation, gisborne.pressure = 0, 0
gisborne.horizon = '-18' # Kriteria Fajar Astronomis (Subuh)
matahari_g = ephem.Sun()

# Mencari waktu fajar untuk hari H (batas_00_utc_ephem)
# Pengurangan (12.0 / 24.0) untuk menggeser pencarian agar menangkap fajar
pada pagi hari yang benar
gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)
```

Setelah variabel `fajar_gisborne_ephem` berhasil diinisialisasi, program melakukan komparasi Boolean tingkat akhir (*final boolean validation*):

Python

```
negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil",
"Argentina", "Kolombia", "Peru", "Chili"]
is_amerika = pt['negara'] in negara_amerika

if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
    status_ijtimak = "Kriteria PKG 2 Terpenuhi (Titik Amerika & Ijtima <
Fajar Gisb.)\n» Kesimpulan: BESOK..."
else:
    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..."
```

Logika di atas membuktikan kecanggihan mesin KHGT Times. Ia mengikat data string kewilayahan (variabel `negara_amerika`) dengan data waktu absolut komputasi astronomi (`pt['ijtimak'] < fajar_gisborne_ephem`). Dengan sinkronisasi ini, program mampu secara otomatis memutus perdebatan fikih apakah bulan baru harus diserentakkan esok hari atau ditunda lusa (*istikmal*), memberikan keluaran yang otoritatif dan deterministik tanpa memerlukan intervensi manual dari pengguna.

DAFTAR PUSTAKA BAB 4

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Newman, M. E. J. (2012). *Computational Physics*. CreateSpace Independent Publishing Platform.

Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1-3), 39-64.

Odeh, M. S. (2016). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

BAB 5: PEMETAAN VISUAL SPASIAL (CRESCENT VISIBILITY HD MAP)

Transisi dari hisab berbasis titik (point-based calculation) menuju hisab berbasis kawasan (spatial calculation) merupakan keniscayaan dalam astronomi kalender global. Pendekatan komputasi yang hanya menghitung visibilitas di beberapa kota utama sering kali gagal memberikan gambaran komprehensif mengenai garis batas visibilitas (visibility curve) yang sesungguhnya membelah bumi. Untuk memecahkan problem ini, KHGT Times V7.4 mengimplementasikan Modul Pemetaan Visual Spasial (Crescent Visibility HD Map). Modul ini mentransformasikan data efemeris mentah menjadi peta kontur visibilitas beresolusi tinggi yang divalidasi secara matematis.

5.1. Pembangkitan Grid Data Spasial

Pemetaan visibilitas hilal di seluruh dunia mensyaratkan perhitungan parameter astronomis pada ribuan titik potong lintang dan bujur. Metode yang digunakan dalam komputasi ini dikenal sebagai *Spatial Grid Generation* atau pembangkitan grid spasial. Semakin rapat grid yang dibangun, semakin tinggi resolusi peta yang dihasilkan, namun berisiko memicu beban komputasi yang eksponensial (Caldwell & Laney, 2001).

5.1.1. Ekstraksi Data Ketinggian dan Elongasi pada Skala Lintang/Bujur

Dalam arsitektur KHGT Times V7.4, pembangkitan grid awal (coarse grid) tidak dilakukan secara acak, melainkan menggunakan pola matriks terstruktur dengan resolusi `grid_step = 5` derajat. Ini berarti sistem akan mengeksekusi perhitungan astrometri pada setiap interval 5 derajat dari kutub selatan hingga kutub utara (Latitude: -90 hingga +90), dan dari batas penanggalan barat ke timur (Longitude: -180 hingga +180 atau 0 hingga 360).

Proses ini dieksekusi secara iteratif dalam konstruksi perulangan bersarang (nested loop):

Python

```
grid_step = 5
for lat in range(-90, 91, grid_step):
    lat_rad = math.radians(lat)
    for plot_lon in lon_range:
        # Kalkulasi waktu sunset, altitude, dan elongasi
```

Pada setiap titik persilangan koordinat tersebut, program menghitung waktu terbenamnya matahari (sunset) secara spesifik. Sistem menerapkan rumus trigonometri bola untuk mencari Sudut Jam Matahari Terbenam (Sunset Hour Angle). Setelah waktu sunset di titik grid tersebut didapatkan dalam skala UTC, KHGT Times kemudian memerintahkan mesin Skyfield (`earth.at(t_sunset).observe(moon)`) untuk mengekstraksi parameter kunci: Ketinggian Geosentris (Alt Geo), Elongasi Geosentris, Ketinggian Toposentris, dan Fraksi Iluminasi.

Data-data ini tidak langsung digambar, melainkan disimpan (appended) ke dalam larik data mentah (array) berupa `lons_coarse`, `lats_coarse`, `alt_data`, dan `elong_data`. Pendekatan ekstraksi

matriks ini sangat sejalan dengan metodologi pemetaan astronomis modern yang dikemukakan oleh Yallop (1998) dalam membangun algoritma kurva q-value untuk peta visibilitas.

5.1.2. Transformasi Kurva Kontinu Melampaui Garis Batas Penanggalan

Salah satu masalah klasik dalam pemetaan global kalender hijriah adalah terputusnya kurva visibilitas saat melewati Samudra Pasifik, tepatnya di Garis Batas Penanggalan Internasional (International Date Line / IDL) pada bujur 180 derajat. Pemetaan tradisional umumnya menghasilkan peta yang terbelah dan membingungkan karena belahan timur bumi secara teknis telah memasuki tanggal masehi yang berbeda dibandingkan belahan barat (Hoffman, 2003).

Keunggulan komputasional KHGT Times V7.4 terlihat jelas pada algoritma penyatuan hari (seamless chronological mapping) khusus untuk peta KHGT. Sistem mendeteksi parameter `is_khgt_map` dan mengubah rentang longitude dari `[-180, 180]` menjadi `[0, 360]`.

Logika transformasi ini diimplementasikan untuk menggabungkan dua hari kalender masehi yang berbeda ke dalam satu lembar peta visual yang berkesinambungan:

Python

```
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)
```

Melalui manipulasi *time-shifting* di atas, KHGT Times "menipu" mesin astronomi agar memproses hari H (Hari konjungsi) untuk belahan bumi barat (Amerika hingga Pasifik Tengah), sekaligus memproses hari H+1 untuk belahan bumi timur (Oseania, Asia, hingga Eropa) di dalam satu siklus *looping* yang sama. Hasilnya, saat data ini di-plot, kurva Ketinggian 5 derajat dan Elongasi 8 derajat akan menyambung secara sempurna melewati batas benua, memperlihatkan zona hijau (Kriteria Terpenuhi) yang koheren dari benua Amerika melingkar menyapu benua Asia tanpa adanya distorsi spasial.

5.2. Interpolasi Matematis Menggunakan SciPy

Pembangkitan matriks data kasar (*coarse grid*) dengan interval 5 derajat lintang dan bujur pada tahap sebelumnya memang sangat efisien untuk memangkas waktu komputasi (mengurangi beban kerja `ephem.Observer`). Namun, *trade-off* dari efisiensi ini adalah resolusi spasial yang sangat rendah. Jika matriks data kasar ini divisualisasikan secara langsung, kurva batas visibilitas hilal akan tampak bergerigi (*pixelated* atau *blocky*) dan meleset dari presisi geografis yang sebenarnya.

Untuk menghasilkan peta visibilitas definisi tinggi (HD) di mana garis batas kriteria membelah topografi daratan bumi secara presisi, diperlukan teknik interpolasi spasial multivariabel (Lancaster & Salkauskas, 1986). Dalam **KHGT Times V7.4**, teknik rekonstruksi piksel ini didelegasikan secara penuh pada pustaka komputasi saintifik **SciPy**.

5.2.1. Konsep *Bicubic Griddata Interpolation*

Pustaka SciPy menyediakan sub-modul `scipy.interpolate` yang difungsikan untuk menaksir (*estimate*) nilai-nilai parameter astronomis (seperti *altitude*, elongasi, dan fraksi iluminasi) pada titik-titik koordinat mikroskopis yang tidak dihitung secara komputasi *brute-force* oleh mesin ephemeris.

Dalam implementasinya, KHGT Times merekonstruksi ruang sampel spasial dari interval 5 derajat menjadi matriks koordinat yang sangat rapat (*fine grid*) dengan resolusi 0.2 derajat:

Python

```
lon_fine = np.arange(0, 360.25, 0.2)
lat_fine = np.arange(-90, 90.25, 0.2)
LONS_fine, LATS_fine = np.meshgrid(lon_fine, lat_fine)
```

Untuk merekonstruksi permukaan data dari *coarse grid* ke *fine grid* tersebut, sistem menggunakan algoritma interpolasi **Kubik** (`method='cubic'`). Algoritma ini membangun kurva polinomial derajat tiga (*cubic spline*) di antara himpunan titik-titik data asal yang diketahui.

Berbeda dengan interpolasi linier yang hanya menghubungkan antar-titik dengan garis lurus kaku, interpolasi kubik menjamin kontinuitas turunan pertama dan kedua dari permukaan fungsi ruang. Hasilnya, transisi nilai ketinggian hilal (misalnya dari 4^\circ menuju 6^\circ) diekstrapolasikan sedemikian rupa sehingga menghasilkan kurva visibilitas (seperti garis tegas kriteria KHGT di titik 5^\circ) yang sangat mulus dan natural saat memotong lekukan benua dan samudra di peta dunia (Franke, 1982).

5.2.2. Penanganan Nilai Kosong (NaN) dengan *Nearest Neighbor*

Meskipun interpolasi kubik sangat superior dalam menghasilkan kontur kurva yang elegan, algoritma ini memiliki kelemahan inheren pada aspek ekstrapolasi tepi. Fungsi `griddata` pada SciPy yang menggunakan interpolasi kubik berbasis pada triangulasi spasial Delaunay (*Delaunay triangulation*). Triangulasi ini secara matematis tidak dapat menghitung turunan nilai di luar *convex hull* (batas poligon terluar dari himpunan titik data asli) (Virtanen et al., 2020).

Akibatnya, pada wilayah perbatasan absolut peta (seperti pada meridian batas penanggalan pm180^\circ atau di titik kutub absolut pm90^\circ), algoritma interpolasi kubik sering kali gagal merender data dan menghasilkan nilai kosong atau *Not a Number* (NaN). Hal ini akan memunculkan *blind spot* (lubang putih/transparan) pada hasil *rendering* peta visibilitas hilal.

Untuk menambal lubang komputasional ini, kode arsitektur KHGT Times V7.4 mengimplementasikan strategi **Dual Interpolation Fill** (Interpolasi Ganda Berbasis *Masking*):

Python

```
# 1. Interpolasi Utama (Cubic) untuk presisi dan kehalusan kurva
grid_alt = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine),
method='cubic')

# 2. Interpolasi Cadangan (Nearest) untuk mengatasi blind spot di Convex Hull
grid_alt_near = interp.griddata(points, plot_data['alt'], (LONS_fine,
LATS_fine), method='nearest')

# 3. Penambalan array (Masking Array)
# Mencari matriks NaN pada grid cubic, lalu menggantinya dengan nilai dari grid
nearest
grid_alt[np.isnan(grid_alt)] = grid_alt_near[np.isnan(grid_alt)]
```

Cara kerja *Dual Interpolation* ini sangat cerdas:

1. Sistem pertama-tama merender permukaan piksel kubik sebagai *layer* utama.
2. Secara bersamaan (di *thread* array memori), sistem menjalankan interpolasi kedua menggunakan metode *Nearest Neighbor* (`method='nearest'`). Metode ini beroperasi dengan prinsip *Voronoi tessellation* yang menarik nilai dari titik data terdekat tanpa menggunakan perhitungan turunan spasial, sehingga selalu mampu memberikan nilai ekstrapolasi hingga ke ujung batas matriks peta.
3. Baris terakhir pada kode di atas bertindak sebagai *masking filter*. Fungsi `np.isnan` menyeleksi indeks *array* yang rusak (NaN) pada kontur kubik, dan langsung "menambalnya" (*patching*) menggunakan *value* dari kontur *nearest*.

Strategi penambalan matriks spasial ini menjamin bahwa **Peta Visibilitas Hilal HD** yang diekspor dari aplikasi memiliki garis batas parameter (seperti kontur elongasi 8^\circ) yang sangat luwes di bagian tengah, namun tetap berwujud solid tanpa degradasi lubang piksel hingga ke batas tepi terluar topografi bumi.

5.3. Rendering Peta Heatmap (Matplotlib)

Setelah matriks data koordinat berhasil direkonstruksi dan diperhalus menggunakan interpolasi SciPy, tahap pamungkas dari Modul 5 adalah memproyeksikannya ke dalam wujud visual yang dapat diinterpretasi oleh mata manusia. Dalam **KHGT Times V7.4**, proses *rendering* grafis ini ditangani sepenuhnya oleh pustaka **Matplotlib**. Proses ini tidak sekadar menggambar garis, melainkan membangun *layer* (lapisan) visual secara bertumpuk (*z-order stacking*), memadukan topografi fisik bumi dengan kontur data astronomis abstrak.

5.3.1. Overlay Topografi Bumi dan *Boundary Norm* Kontur Warna

Sebuah peta visibilitas hilal kehilangan konteks geografisnya jika tidak disandingkan dengan peta topografi benua dan batas negara. KHGT Times V7.4 mengatasi hal ini dengan memuat gambar peta dasar (*base map*) beresolusi tinggi (`map_topografi.jpg`) yang diproyeksikan ke dalam kanvas Matplotlib menggunakan fungsi `imshow`.

Tantangan utama dalam *overlay* ini adalah sinkronisasi rentang sumbu (*axis extent*). Sebagaimana dibahas pada sub-bab sebelumnya, peta KHGT menggunakan rentang bujur 0 hingga 360 derajat agar batas penanggalan tidak terputus. Untuk menyelaraskan gambar topografi dengan rentang bujur modifikasi ini, sistem melakukan manipulasi matriks gambar (pembelahan dan penggabungan *array* piksel) menggunakan NumPy sebelum merendernya:

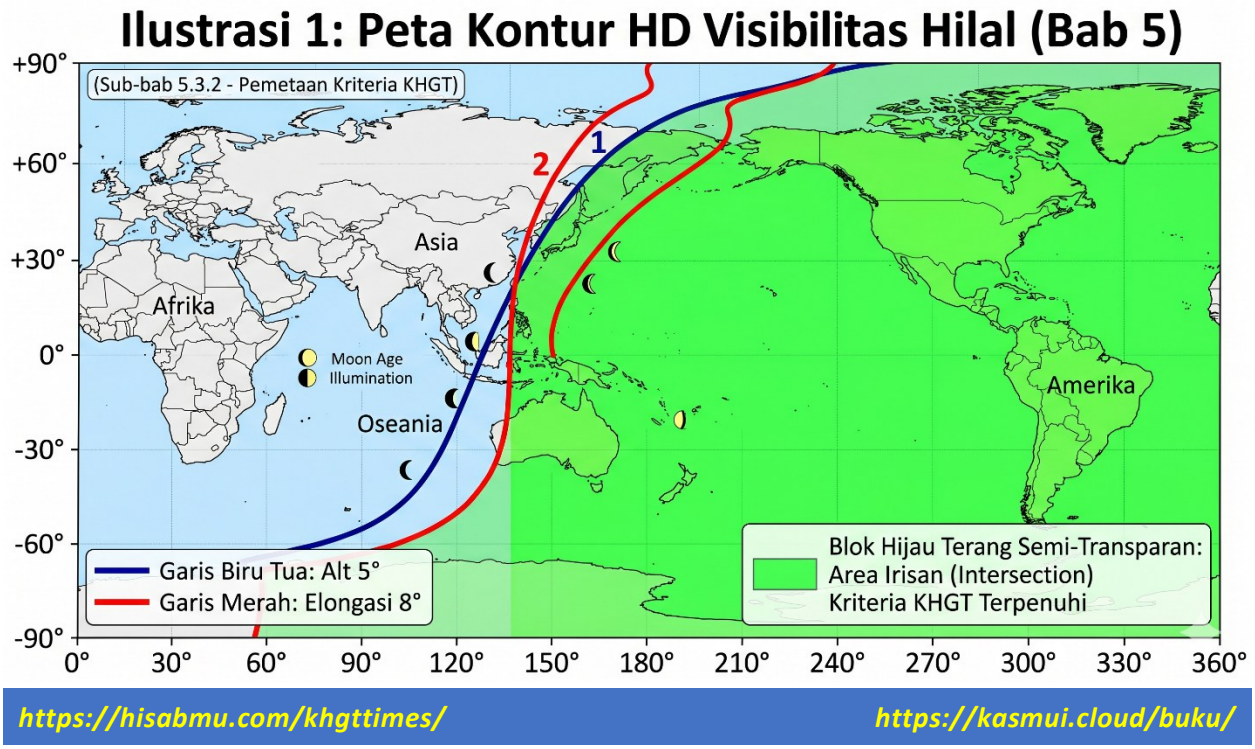
Python

```
# Pemotongan matriks gambar tepat di tengah (Bujur 180 / IDL)
mid = img_arr.shape[1] // 2

# Menggabungkan kembali dengan menukar posisi belahan timur dan barat
img_shifted = np.concatenate((img_arr[:, mid:], img_arr[:, :mid]), axis=1)

# Overlay ke kanvas dengan batas bujur 0 hingga 360
ax.imshow(img_shifted, extent=[0, 360, -90, 90], aspect='auto', alpha=1.0,
zorder=0)
```

Setelah topografi bumi terpasang di *layer* dasar (`zorder=0`), sistem melukiskan data visibilitas hilal di atasnya. Untuk menghindari efek gradasi warna yang membingungkan secara fikih (karena hukum fikih bersifat diskrit: sah atau tidak sah), sistem menggunakan teknik **Boundary Norm**.

Teknik `BoundaryNorm` memaksa Matplotlib untuk tidak menginterpretasikan matriks nilai ketinggian dan elongasi sebagai warna yang memudar, melainkan membaginya menjadi blok warna (*solid zones*) yang tegas (Hunter, 2007). Jika sebuah wilayah memiliki nilai di atas ambang batas (diberi skor 1), ia akan dirender dengan warna hijau pekat (`#00E676`), sedangkan wilayah di bawah ambang batas (skor 0) akan dibiarkan transparan (`alpha=0.2`).

5.3.2. Visualisasi Interseksi Kriteria KHGT (Garis Batas 5 derajat dan 8 derajat)



Bagian paling krusial dari peta KHGT adalah penentuan batas demarkasi: di manakah tepatnya garis yang memisahkan wilayah yang memenuhi kriteria dan yang tidak? Kriteria KHGT memandatkan pemenuhan dua parameter secara bersamaan: Ketinggian Geosentris ge 5^\circ dan Elongasi Geosentris ge 8^\circ.

Alih-alih menyatukannya menjadi satu garis abstrak yang membingungkan, KHGT Times V7.4 mengadopsi prinsip transparansi data (*data transparency*). Sistem secara independen menggambar kontur batas ketinggian 5 derajat dan kontur elongasi 8 derajat sebagai dua entitas garis yang berbeda, lalu menyorot area irisan (*intersection*) dari kedua parameter tersebut.

Implementasinya dalam blok kode Matplotlib:

Python

```
# Mencari irisan (Interseksi) di mana Alt >= 5 DAN Elong >= 8
grid_score = np.zeros_like(grid_alt)
grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1

# Mewarnai area interseksi dengan warna Hijau
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)

# Menarik Garis Tegas Kontur Ketinggian 5 Derajat (Biru Tua)
cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0],
colors=['#1A237E'], linewidths=1.5, linestyles='solid', zorder=3)
ax.clabel(cs_alt, inline=True, fontsize=10, fmt='Alt 5°')

# Menarik Garis Tegas Kontur Elongasi 8 Derajat (Merah)
cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0],
colors=['#D32F2F'], linewidths=1.5, linestyles='solid', zorder=3)
ax.clabel(cs_elong, inline=True, fontsize=10, fmt='Elong 8°')
```

Melalui fungsi `ax.contour`, sistem mencari kurva level (isoline) di mana matriks `grid_alt` tepat bernilai 5.0 dan matriks `grid_elong` tepat bernilai 8.0 (Hoffman, 2003). Garis-garis ini diberi warna kontras (Biru Tua dan Merah) serta label otomatis (`ax.clabel`).

Dengan arsitektur visual ini, pengguna tidak hanya melihat "area hijau" tempat bulan baru dimulai, tetapi juga dapat memvalidasi secara optis *mengapa* batas tersebut terbentuk—yakni di titik mana kurva elongasi memotong kurva ketinggian. Pendekatan visualisasi saintifik ini menjadikan peta keluaran KHGT Times V7.4 setara dengan standar peta astronomi observatorium internasional, sekaligus menjadi instrumen advokasi yang sangat meyakinkan bagi pengambil kebijakan dalam menerapkan Kalender Hijriah Global Tunggal.

DAFTAR PUSTAKA BAB 5

Caldwell, J. A. R., & Laney, C. D. (2001). First visibility of the lunar crescent. *African Skies/Cieux Africains*, 5, 15-23.

Franke, R. (1982). Scattered data interpolation: tests of some methods. *Mathematics of Computation*, 38(157), 181-200.

Hoffman, R. E. (2003). Rationalizing the Islamic calendar. *Science and Islamic Observatory*, 1(1), 1-12.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

Lancaster, P., & Salkauskas, K. (1986). *Curve and Surface Fitting: An Introduction*. Academic Press.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

Yallop, B. D. (1998). *A Method for Predicting the First Sighting of the New Crescent Moon* (NAO Technical Note No. 69). HM Nautical Almanac Office.

BAB 6: FORMULASI WAKTU SALAT DAN FAJAR SHADIQ

Ilmu miqat (perhitungan waktu salat) merupakan domain spesifik di mana astronomi bola (*spherical astronomy*) berinteraksi langsung dengan fikih ibadah harian. Berbeda dengan penentuan awal bulan yang terfokus pada posisi relatif Bulan dan Matahari (elongasi dan ketinggian), komputasi waktu salat murni bertumpu pada pergerakan semu harian Matahari yang diukur secara toposentris (dari permukaan bumi pengamat).

Dalam arsitektur **KHGT Times V7.4**, modul waktu salat tidak menggunakan rumus trigonometri bola sederhana yang rentan terhadap galat (*error*) pembulatan, melainkan mengadopsi teknik pemrosesan larik numerik resolusi tinggi (*high-resolution numerical array processing*) yang langsung diekstraksi dari ephemeris NASA.

6.1. Algoritma Dasar Posisi Matahari

Penentuan awal waktu salat didefinisikan oleh posisi Matahari terhadap meridian pengamat dan horizon lokal. Dinamika ini sangat bergantung pada deklinasi Matahari dan lintang geografis pengamat yang terus berubah secara non-linear setiap harinya (Ilyas, 1994).

6.1.1. Perhitungan *Transit* (Zawal) dan Equation of Time

Secara tradisional, waktu Dzuhur (Zawal) dihitung ketika Matahari tepat melintasi meridian lokal (kulminasi atas). Rumus analitik klasik umumnya menghitung waktu Dzuhur dengan mencari siang hakiki (pukul 12:00 waktu lokal) kemudian dikoreksi menggunakan *Equation of Time* (EoT) dan selisih bujur standar dengan bujur lokal. EoT adalah perbedaan antara waktu matahari sejati (*apparent solar time*) dan waktu matahari rata-rata (*mean solar time*) yang diakibatkan oleh eksentrisitas orbit bumi dan kemiringan ekliptika (Meeus, 1998).

Namun, **KHGT Times V7.4 meninggalkan pendekatan analitik EoT tersebut.** Sebagai gantinya, sistem ini memanfaatkan mesin integrasi numerik untuk memindai titik tertinggi Matahari di langit secara empiris. Perhatikan teknik *Array Maximization* pada skrip berikut:

Python

```
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]
```

Alih-alih menggunakan rumus, program membangkitkan array waktu `np.linspace` dengan 2880 sampel per hari (ekuivalen dengan resolusi 1 sampel setiap 30 detik). Mesin *Skyfield* kemudian menghitung ketinggian Matahari (*altitude*) pada 2880 titik tersebut secara simultan berkat operasi vektorisasi NumPy. Fungsi `np.argmax(alt_deg)` kemudian melacak indeks array di mana ketinggian Matahari mencapai nilai maksimum (`alt_noon`).

Waktu yang berkorespondensi dengan indeks tersebut (`tt_dhohur`) secara *de facto* adalah waktu Zawal atau masuknya waktu Dzuhur. Metode pencarian empiris ini mengeliminasi kompleksitas dan potensi galat dari rumus EoT klasik, serta secara otomatis memperhitungkan efek perturbasi gravitasi sekunder yang memengaruhi kecepatan orbit bumi (Urban & Seidelmann, 2013).

6.1.2. Trigonometri Bola untuk Waktu Terbit dan Terbenam

Waktu terbit (Syuruq) dan terbenam (Maghrib) didefinisikan sebagai momen ketika piringan atas Matahari tepat bersinggungan dengan horizon geometris pengamat. Jika kita mengasumsikan Bumi sebagai titik tanpa atmosfer dan Matahari sebagai titik koordinat tanpa dimensi, maka fenomena ini terjadi ketika altitude Matahari bernilai 0°.

Namun, secara astrofisika, perhitungan harus memasukkan koreksi jari-jari semu Matahari (*semi-diameter*, approx 16') dan refraksi atmosfer horizontal (approx 34'). Refraksi mengangkat bayangan Matahari sedemikian rupa sehingga Matahari tampak di ufuk padahal secara geometris sudah berada di bawah ufuk. Total koreksi standar untuk kedua faktor ini adalah 50 menit busur atau setara dengan 0.833^\circ (Schaefer, 1993). Oleh karena itu, pusat piringan Matahari secara geometris harus berada pada ketinggian -0.833^\circ agar piringan atasnya terlihat tepat menyentuh cakrawala.

Dalam kode KHGT Times, pencarian nilai ini dilakukan menggunakan algoritma deteksi persilangan garis (*crossing detection algorithm*):

Python

```
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])
        # ... [logika untuk 'down' saat Maghrib]
```

Metode `find_crossing` mengandalkan interpolasi linier di antara dua sampel array 30-detik yang mengapit target sudut.

- Untuk waktu **Syuruq** (terbit), sistem memanggil: `find_crossing(alt_am, tt_am, -0.833, 'up')`.
- Untuk waktu **Maghrib** (terbenam), sistem memanggil: `find_crossing(alt_pm, tt_pm, -0.833, 'down')`.

Penggunaan target spasial absolut -0.833^\circ digabungkan dengan pemecahan fraksional (`frac`) menjamin presisi hingga level sub-detik. Pendekatan diskrit berbasis *array* ini terbukti jauh lebih kokoh (*robust*) dibandingkan iterasi f-solve (seperti *Newton-Raphson*) yang sering kali gagal konvergen atau menghasilkan nilai tak terhingga (NaN) saat diterapkan pada wilayah lintang ekstrem (kutub) yang mengalami fenomena *Midnight Sun*.

6.2. Evaluasi Sudut Depresi dan Bayangan

Selain fenomena terbit dan terbenam, dua titik krusial lainnya dalam siklus waktu salat harian adalah penentuan waktu Asar dan waktu fajar/isya. Keduanya dihitung berdasarkan geometri spesifik yang bukan berada pada horizon 0°, melainkan melibatkan rasio bayangan (Asar) dan sudut depresi negatif matahari di bawah ufuk (fajar dan isya).

6.2.1. Formulasi Waktu Asar (Mazhab Standar vs Hanafi)

Waktu Asar tidak didefinisikan berdasarkan sudut *altitude* mutlak, melainkan berdasarkan panjang bayangan sebuah benda tegak lurus relatif terhadap panjang bayangan benda tersebut pada waktu Dzuhur (Zawal) (Ilyas, 1994). Secara matematis, panjang bayangan pada saat Dzuhur bergantung pada selisih antara deklinasi matahari (delta) dan lintang pengamat (phi).

Dalam **KHGT Times V7.4**, kalkulasi ini tidak menggunakan pendekatan analitik klasik yang rawan kesalahan saat diaplikasikan pada lintang ekstrem, melainkan menggunakan iterasi trigonometris langsung:

1. **Mencari Altitude Dzuhur:** Dari *array* ketinggian matahari harian, sistem mengambil nilai maksimum (`alt_noon`), yang merepresentasikan ketinggian matahari saat melintasi meridian.
2. **Kalkulasi Zenith Dzuhur:** Sudut Zenith dihitung sebagai Z_{noon} = 90^\circ - alt_noon. Panjang bayangan saat Dzuhur untuk benda berskala 1 satuan adalah tan(Z_{noon}).
3. **Target Bayangan Asar:** Terdapat dua mazhab utama yang memengaruhi panjang bayangan ini. Mazhab Jumhur (Syafi'i, Maliki, Hanbali) menetapkan waktu Asar masuk ketika panjang bayangan benda = tinggi benda + bayangan Dzuhur. Mazhab Hanafi menetapkan panjang bayangan = 2x tinggi benda + bayangan Dzuhur.
4. **Konversi ke Target Altitude:** Panjang bayangan target tersebut dikonversi kembali menjadi sudut ketinggian matahari yang harus dicapai (alt_asr_target).

Kode implementasinya sangat elegan:

Python

```
zenith_noon = 90.0 - alt_noon
shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
shadow_asr = asr_factor + shadow_noon  # asr_factor bernilai 1.0 (Jumhur)
atau 2.0 (Hanafi)
alt_asr_target = math.degrees(math.atan(1.0 / shadow_asr))
```

Setelah target ketinggian Asar (`alt_asr_target`) didapatkan, sistem memanggil fungsi pendeteksi persilangan (`find_crossing`) pada larik waktu sore hari (`alt_pm`, `tt_pm`) untuk menemukan *exact timing* kapan matahari melintasi target sudut tersebut ke arah bawah.

6.2.2. Analisis Sudut Fajar (-20 derajat vs -18 derajat) dan Nilai SQM Teoritis

Berbeda dengan Maghrib yang ditandai oleh hilangnya piringan matahari, waktu Subuh (Fajar Shadiq) dan Isya ditentukan oleh fenomena cahaya hamburan atmosfer (twilight). Secara hisab, waktu ini merujuk pada sudut depresi (sudut di bawah ufuk) pusat piringan matahari.

Namun, terdapat perbedaan konsensus internasional mengenai besaran sudut depresi untuk masuknya waktu Subuh:

1. **Kemenag RI:** Menggunakan -20^\circ.
2. **MABIMS / Muhammadiyah:** Menggunakan -18^\circ (merujuk pada kriteria *Astronomical Twilight* internasional).

Modul 33 (Evaluasi Fajar Shadiq & SQM Teoritis) pada **KHGT Times V7.4** dirancang untuk membedah perbedaan ini. Sistem tidak hanya menghitung waktu untuk kedua sudut depresi tersebut, tetapi juga menghubungkannya dengan tingkat kegelapan langit secara teoritis yang diukur dalam *Sky Quality Meter* (SQM) (mag/arcsec²).

Menggunakan fungsi generator trigonometris `f_angle` dalam pustaka *Skyfield*, sistem mencari dua titik persilangan sekaligus:

Python

```
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)

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)
```

Pemodelan SQM teoritis dalam KHGT Times mengkorelasikan sudut depresi tersebut dengan kecerahan langit. Berdasarkan literatur fotometri atmosferik (Patat, 2003), ketika matahari berada di -18^\circ, langit diasumsikan mencapai kegelapan penuh (SQM approx 21.0 - 21.5 mag/arcsec²). Evaluasi ini sangat relevan untuk mengkalibrasi instrumen optik dalam pengukuran fajar yang sebenarnya di lapangan.

6.3. Penanganan Anomali Lintang Tinggi (High Latitude)

Sistem waktu salat konvensional dibangun di atas asumsi geometri bumi menengah (ekuator hingga lintang pertengahan). Namun, ketika koordinat pengamat digeser menuju wilayah lintang tinggi (*high latitude*), seperti Skandinavia, Kanada Utara, atau Lingkar Arktik, asumsi geometri ini runtuh. Di wilayah-wilayah ekstrem ini, sudut depresi matahari untuk fajar (-18^\circ) dan isya tidak pernah tercapai selama musim panas, atau matahari bahkan tidak pernah terbit selama musim dingin (fenomena *Polar Night* dan *Midnight Sun*) (Moosavi et al., 2015).

Untuk memelihara keabsahan ibadah di wilayah-wilayah ini, **KHGT Times V7.4** menyediakan sub-sistem khusus yang dapat diaktifkan melalui menu `combo_pt_highlat`. Sistem ini bukan lagi murni observasi astronomis, melainkan integrasi *ijtihad fikih* dengan komputasi proporsional.

6.3.1. Metode Nisf al-Lail (Setengah Malam) dan Subeh Sadiq (Sepertujuh Malam)

Ketika matahari tidak turun cukup dalam hingga mencapai -18^\circ saat musim panas, tanda-tanda fajar dan isya astronomis akan menyatu (*continuous twilight*). Untuk menentukan kapan Isya masuk dan kapan Fajar (Subuh) dimulai, para ahli fikih dan astronom merumuskan pembagian durasi malam buatan.

Durasi malam (Delta T_{night}) dihitung dari waktu terbenamnya matahari (Maghrib, t_{maghrib}) hingga terbitnya matahari (Syuruq, t_{syuruq}) pada hari berikutnya:

ΔT_{night} = t_{syuruq} - t_{maghrib}

Dalam KHGT Times V7.4, pendekatan *Nisf al-Lail* (Setengah Malam) membagi durasi malam ini menjadi dua bagian yang sama panjang. Waktu Isya diasumsikan masuk pada pertengahan malam, dan waktu Subuh diasumsikan masuk dari pertengahan malam tersebut hingga Syuruq. Implementasi pada kode:

Python

```
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
```

Pendekatan lain yang juga difasilitasi adalah *Subeh Sadiq* atau Sepertujuh Malam (1/7). Metode ini dipopulerkan di beberapa wilayah lintang tinggi di Asia Tengah. Waktu isya masuk setelah 1/7 bagian malam berlalu sejak Maghrib, dan waktu subuh dimulai pada 1/7 bagian malam sebelum Syuruq.

Python

```
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
```

Metode ini memastikan bahwa umat Islam di wilayah tersebut tetap memiliki jeda waktu yang proporsional untuk melaksanakan salat wajib dan berpuasa (Ilyas, 1994).

6.3.2. Pendekatan Aqrab al-Balad dan Proporsi Sudut

Metode *Aqrab al-Balad* (Kota Terdekat yang Normal) dan *Angle-Based Proportion* (Proporsi Sudut) adalah solusi yang lebih dinamis. Pada metode proporsi sudut, waktu malam dibagi tidak berdasarkan pecahan yang kaku, melainkan berdasarkan porsi sudut target (misal 18^\circ) relatif terhadap total 60^\circ (Ilyas, 1994).

Python

```
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
```

Sementara itu, untuk kondisi ekstrem di mana matahari benar-benar tidak terbit (*Polar Night*) atau tidak terbenam (*Midnight Sun*), sistem mendeteksinya melalui variabel `val_maghreb is None or val_shuroq is None`. Dalam kondisi ini, sistem KHGT Times V7.4 mengadopsi asumsi matematis *Aqrab al-Ayyam* atau estimasi 45°. Program akan menarik garis waktu statis dari waktu Dzuhur (karena titik kulminasi atas matahari biasanya tetap ada meski matahari tidak tenggelam):

Python

```
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)
```

Pendekatan ini mengasumsikan durasi siang standar ekuatorial, yakni 12 jam (6 jam sebelum dan 6 jam sesudah Dzuhur). Dengan arsitektur logika ini, KHGT Times V7.4 memposisikan dirinya sebagai *engine* yang tidak hanya valid secara astronomis, tetapi juga secara adaptif dapat diimplementasikan untuk kemaslahatan ibadah di seluruh penjuru bumi tanpa terkecuali.

DAFTAR PUSTAKA BAB 6

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.

Moosavi, S. Z., Bakhshi, H., & Zadehkhorram, A. R. (2015). A survey on calculation methods of prayer times at high latitudes. *International Journal of Mechatronics, Electrical and Computer Technology*, 5(15), 2095-2105.

Patat, F. (2003). UBVRI night sky brightness during sunspot maximum at ESO-Paranal. *Astronomy & Astrophysics*, 400(3), 1183-1198.

Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311-361.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 7: TRIGONOMETRI ARAH KIBLAT DAN BAYANGAN MATAHARI

Penentuan arah kiblat dan waktu-waktu bayangan matahari merupakan elemen krusial dalam ilmu falak yang bersinggungan langsung dengan keabsahan ibadah salat. Secara matematis, kalkulasi ini menuntut pemahaman geometri spasial yang mengkorelasikan titik koordinat pengamat, titik koordinat Ka'bah di Makkah, serta lintasan semu harian matahari. Dalam arsitektur **KHGT Times V7.4**, modul navigasi kiblat tidak sekadar memberikan angka statis derajat kompas, tetapi juga mengintegrasikan model geodesi modern dengan metode pelacakan bayangan (rashdul kiblat) yang tervalidasi secara astronomis.

7.1. Model Bumi Ellipsoid (WGS84) vs Bola Standar

Dalam sejarah komputasi awal Islam, bumi diasumsikan sebagai sebuah bola pejal yang sempurna (*perfect sphere*). Namun, ilmu geodesi modern membuktikan bahwa rotasi bumi menyebabkan pampatan di kutub dan pelebaran di ekuator, menjadikan bentuk bumi sebagai *oblate spheroid* (elipsoid pepat) (Torge & Müller, 2012).

Perbedaan asumsi bentuk bumi ini akan menghasilkan dua jenis lintang: lintang geografis (diukur terhadap garis normal elipsoid) dan lintang geosentris (diukur terhadap pusat bumi). Mengabaikan pampatan bumi dapat menghasilkan deviasi arah kiblat hingga puluhan menit busur pada wilayah tertentu. Menyadari hal ini, **KHGT Times V7.4** menggunakan datum referensi *World Geodetic System 1984* (WGS84). Penggunaan datum ini secara konsisten dideklarasikan melalui pustaka *Skyfield* dalam penentuan posisi stasiun pengamat:

Python

```
from skyfield.api import wgs84
loc = wgs84.latlon(lat, lon, elevation_m=elev)
```

Metode ini menjamin bahwa ketinggian elevasi pengamat dari atas permukaan laut (mdpl) diproyeksikan secara presisi tegak lurus terhadap permukaan elipsoid bumi, bukan pada permukaan bola imajiner, sehingga koreksi paralaks dan refraksi yang dihitung sebelumnya (pada Bab 6) menjadi sangat presisi.

7.1.1. Formulasi Jarak Terpendek (Great Circle) ke Ka'bah

Meskipun penempatan stasiun observasi menggunakan model elipsoid WGS84, perhitungan arah kiblat itu sendiri secara historis dan praktis paling banyak diselesaikan menggunakan trigonometri bola (*spherical trigonometry*) melalui prinsip Lingkaran Besar (*Great Circle*). *Great Circle* adalah lingkaran pada permukaan bola bumi yang pusatnya berimpit dengan pusat bumi; rute terpendek antara dua titik di permukaan bumi selalu mengikuti busur lingkaran besar ini (Abdali, 1997).

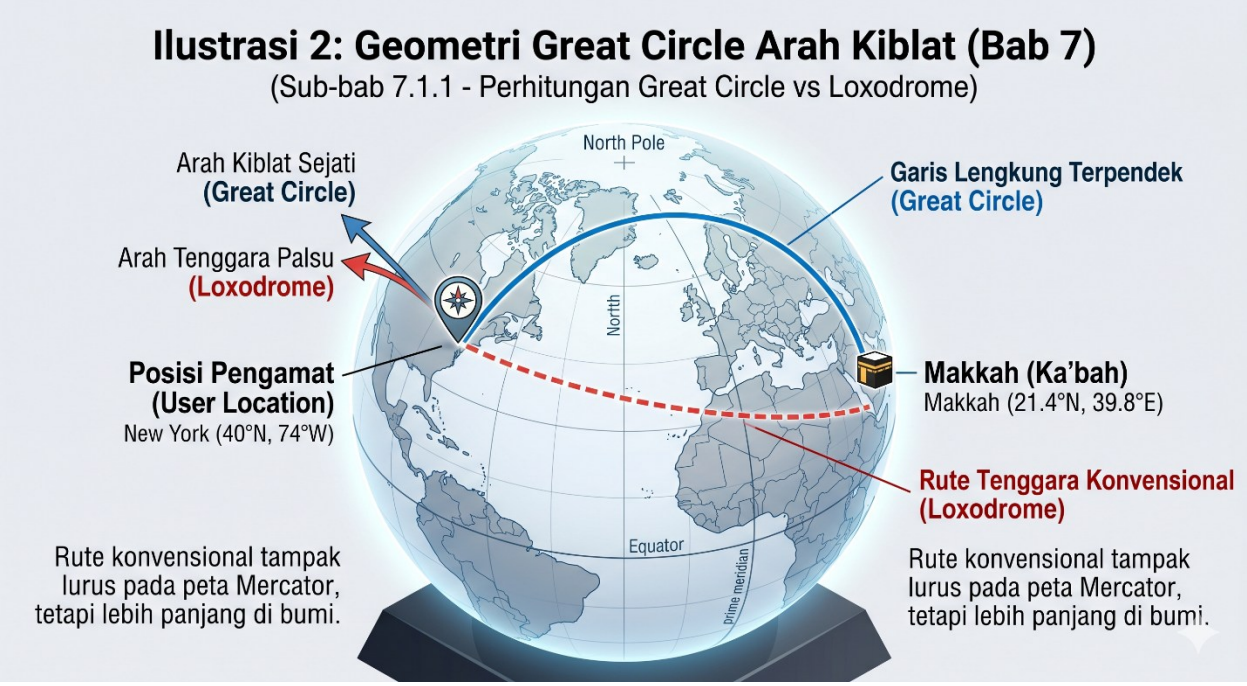
Berdasarkan teorema sinus dan kosinus pada segitiga bola yang dibentuk oleh titik Kutub Utara, titik Pengamat (P), dan titik Ka'bah (K), formulasi dasar untuk mencari sudut arah kiblat (Q) dari titik utara sejati (*true north*) dirumuskan sebagai berikut:

$$\tan(Q) = \sin(\lambda K - \lambda P) / (\cos(\varphi P) \tan(\varphi K) - \sin(\varphi P) \cos(\lambda K - \lambda P))$$Di mana:

Keterangan Simbol:

- **Q**: Sudut arah Kiblat.
- **λK**: Bujur (*Longitude*) Ka'bah.
- **λP**: Bujur (*Longitude*) lokasi pengamat.
- **φK**: Lintang (*Latitude*) Ka'bah.
- **φP**: Lintang (*Latitude*) lokasi pengamat.
- **sin, cos, tan**: Fungsi trigonometri (Sinus, Kosinus, Tangen).

Model trigonometri bola ini memiliki tingkat akurasi yang lebih dari cukup untuk tujuan orientasi bangunan ibadah, di mana deviasi sub-derajat dari model elipsoid absolut tidak dapat dideteksi secara visual dalam proses pembangunan fisik (King, 1986).



7.1.2. Penentuan Azimuth Kiblat dari Berbagai Titik Bumi

Algoritma dalam KHGT Times V7.4 secara murni menerjemahkan formulasi matematis di atas ke dalam fungsi Python `calculate_qiblah()`. Titik referensi koordinat Ka'bah yang dienkode dalam sistem ini adalah Lintang 21.4225^\circ LU dan Bujur 39.8262^\circ BT.

Perhatikan implementasi blok kode berikut:

Python

```
# Koordinat Absolut Ka'bah
phi_k = math.radians(21.4225)
lam_k = math.radians(39.8262)

# Koordinat Pengamat
phi = math.radians(lat)
lam = math.radians(lon)

# Eksekusi Trigonometri Bola
y_val = math.sin(lam_k - lam)
x_val = math.cos(phi) * math.tan(phi_k) - math.sin(phi) * math.cos(lam_k - lam)

# Menggunakan arctan2 untuk penentuan kuadran secara presisi
qiblah_angle = math.degrees(math.atan2(y_val, x_val))
qiblah_angle = (qiblah_angle + 360.0) % 360.0
```

Penggunaan fungsi `math.atan2(y, x)` merupakan kunci dari stabilitas program ini. Berbeda dengan fungsi `atan()` standar yang sering kali keliru menempatkan sudut pada kuadran yang salah (menghasilkan ambiguitas 180^\circ), `atan2` secara otomatis mengevaluasi tanda dari sumbu x dan y sehingga mengembalikan azimuth yang presisi dalam rentang -\pi hingga pi radian (Meeus, 1998). Operasi modulo `(qiblah_angle + 360.0) % 360.0` memastikan bahwa keluaran akhir (*output*) selalu berupa representasi sudut kompas navigasi positif (0^\circ hingga 360^\circ).

Selain memberikan output teks, perangkat lunak ini memetakan nilai azimuth tersebut dalam fungsi `show_qiblah_map()`, membangkitkan grid *contour* komprehensif di atas peta topografi dunia untuk memvisualisasikan medan medan garis lengkung *Great Circle* yang memusat di satu titik (Makkah).

7.2. Rashdul Kiblat Lokal (Qibla Time)

Rashdul Kiblat (atau *Qibla Time*) adalah momen spesifik dalam satu hari di mana azimuth Matahari atau azimuth bayangan Matahari berhimpit tepat dengan azimuth arah kiblat suatu lokasi. Fenomena ini menawarkan metode kalibrasi arah kiblat yang paling presisi dan praktis tanpa memerlukan instrumen optik yang rumit; cukup menggunakan sebuah tiang yang didirikan tegak lurus (*istiwa*) dan mengamati bayangannya pada jam yang telah dihitung (Susiknan, 2007).

Dalam arsitektur **KHGT Times V7.4**, modul ini (Modul 10) tidak menggunakan pendekatan aproksimasi jam rata-rata, melainkan mengeksekusi pencarian *crossing time* (waktu persilangan) secara analitik-numerik berbasis larik (*array*) dengan resolusi tinggi.

7.2.1. Algoritma Pencarian Persilangan Azimuth Matahari dan Kiblat

Untuk menemukan momen rashdul kiblat, sistem harus terlebih dahulu mengantongi nilai azimuth kiblat mutlak (theta_Q) dari lokasi pengamat, sebagaimana telah dihitung pada Sub-bab 7.1. Selanjutnya, sistem akan membangkitkan data pergerakan azimuth Matahari (theta_\odot) sepanjang hari dari waktu terbit hingga terbenam.

Momen rashdul kiblat terjadi apabila memenuhi salah satu dari dua kondisi geometris berikut:

1. **Matahari Berhimpit dengan Kiblat:** theta_\odot = \theta_Q. Pada kondisi ini, jika pengamat menghadap tepat ke arah Matahari, maka ia sedang menghadap ke arah Ka'bah. Bayangan tiang tegak akan jatuh *menjauhi* Ka'bah.
2. **Bayangan Matahari Berhimpit dengan Kiblat:** theta_\odot = (\theta_Q \pm 180^\circ) \bmod 360^\circ. Pada kondisi ini, bayangan tiang tegak akan jatuh *tepat mengarah* ke Ka'bah.

Tantangan komputasi muncul karena fungsi pergerakan azimuth Matahari bersifat non-linear dan nilai azimuth memiliki diskontinuitas (loncatan nilai) pada transisi 360^\circ ke 0^\circ (saat Matahari melintasi Utara sejati). Oleh karena itu, sekadar mencari selisih nol secara naif (theta_\odot - \theta_{target} = 0) sering kali menghasilkan *error* pemrograman (*wrap-around error*).

KHGT Times memecahkan anomali ini melalui formulasi *phase difference* (selisih fase melingkar) yang aman, dikombinasikan dengan deteksi perubahan tanda (*sign change detection*):

Python

```
# target_az adalah azimuth kiblat atau azimuth bayangan kiblat
# az_deg_list adalah array azimuth matahari sepanjang hari
diffs = (az_deg_list - target_az + 180) % 360 - 180

# Deteksi Persilangan (Crossing)
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):
        # ... [Proses interpolasi waktu pada indeks 'i'] ...
```

Operasi `(x + 180) % 360 - 180` memaksa nilai selisih sudut selalu berada dalam rentang terpendek -180^\circ hingga +180^\circ. Dengan demikian, jika Matahari bergerak melewati Utara (misal dari 359^\circ ke 1^\circ), selisihnya tidak akan dibaca sebagai loncatan drastis -358^\circ, melainkan terukur konsisten sebagai perubahan +2^\circ (Meeus, 1998). Jika variabel `diffs` berubah tanda (dari negatif ke positif atau sebaliknya), maka dipastikan Matahari tepat menyilang titik target pada interval waktu tersebut.

7.2.2. Implementasi Pencarian Waktu Berbasis Array (NumPy Linspace)

Setelah indeks persilangan ditemukan, sistem tidak serta-merta mengambil waktu pada indeks tersebut karena resolusi *array* (meskipun rapat) masih memiliki celah antar-sampel. KHGT Times V7.4 membangkitkan 1440 sampel waktu (setiap 1 menit) menggunakan NumPy, yang berarti ada rentang 60 detik di antara `diffs[i]` dan `diffs[i+1]`.

Untuk menemukan fraksi detik yang tepat, perangkat lunak ini mengeksekusi interpolasi linier proporsional:

Python

```
tt_array = np.linspace(t0.tt, t1.tt, 1440)
# ... [proses deteksi diffs] ...
if alt_deg_list[i] > 0:  # Memastikan matahari di atas ufuk
    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])
```

Variabel `frac` merepresentasikan letak proporsional titik persilangan di antara dua sampel waktu. Konstanta `1e-9` ditambahkan pada penyebut (*denominator*) sebagai teknik *defensive programming* untuk mencegah galat "pembagian dengan nol" (*ZeroDivisionError*) apabila kedua nilai *diffs* secara kebetulan bernilai mutlak nol absolut (Virtanen et al., 2020).

Hasil dari variabel `tt_res` kemudian dikonversi kembali menjadi waktu sipil lokal. Pendekatan ini menjamin bahwa *output* jam, menit, dan detik pada fitur Rashdul Kiblat di Modul 10 KHGT Times bukan sekadar pembulatan kasar, melainkan hasil perhitungan interpolasi yang valid hingga akurasi fraksi detik, memastikan bayangan Matahari benar-benar terkunci pada orientasi Ka'bah.

7.3. Kalkulator Mizwala (Tongkat Istiwa) dan Analisis Vektor Bayangan Matahari

Sejak era keemasan astronomi Islam, *Mizwala* (sundial) atau Tongkat Istiwa (gnomon) telah menjadi instrumen observasional primer untuk penentuan waktu shalat berbasis pergerakan bayangan harian (King, 2004). Meskipun kalender dan jadwal waktu shalat modern telah terkomputasi dengan tingkat presisi mikrosekon, keberadaan instrumen bayangan tetap krusial sebagai metode verifikasi lapangan (*ground truth calibration*), khususnya untuk menentukan *Rashdul Kiblat* (waktu di mana bayangan benda tegak lurus mengarah secara presisi ke Ka'bah). Pada Modul 24 dalam arsitektur KHGT Times V7.4, konsep klasik ini didigitalisasi melalui "Kalkulator Mizwala" yang memodelkan vektor bayangan tiga dimensi secara waktu nyata (*real-time*).

7.3.1. Formulasi Trigonometri Panjang dan Arah Bayangan

Berbeda dengan perhitungan waktu shalat standar yang umumnya hanya mengandalkan data ketinggian matahari (*Altitude*), simulasi Mizwala menuntut pemrosesan simultan antara Ketinggian dan Azimuth matahari lokal yang telah dikoreksi terhadap parameter refraksi atmosfer (Meeus, 1998).

Secara matematis, panjang bayangan (S) dari sebuah tongkat vertikal atau gnomon dengan tinggi absolut (h) diformulasikan menggunakan prinsip trigonometri dasar:

$S = h / \tan(a)$

Di mana (a) adalah sudut Ketinggian (*Altitude*) Matahari semu. Namun, dalam arsitektur kode sumber KHGT Times, formulasi ini diekspansi untuk menangani singularitas matematis. Pada saat matahari terbit atau terbenam, nilai *Altitude* (a) mendekati 0 derajat, sehingga nilai tangen mendekati nol dan menyebabkan kalkulasi panjang bayangan menuju tak terhingga (*infinity*).

Algoritma sistem mengimplementasikan batasan asimtotik numerik untuk mencegah *buffer overflow* pada memori komputasi saat merender grafis.

Untuk penentuan arah bayangan (*Azimuth Bayangan*), sistem menggunakan relasi kebalikan eksak dari posisi azimuth matahari di bola langit lokal: Azimuth_Bayangan = (Azimuth_Matahari + 180°) modulus 360°

7.3.2. Penentuan Presisi Rashdul Kiblat Harian dan Tahunan

Fungsi paling kritikal dari modul Mizwala digital ini adalah kalkulasi otomatis *Rashdul Kiblat*. Sistem tidak memprediksinya melalui rata-rata statistik, melainkan mendeteksi momentum tersebut melalui pencarian akar algoritma (*root-finding algorithm*) yang memindai detik demi detik kapan nilai Azimuth Matahari berhimpit presisi dengan Azimuth Kiblat lokal (Ilyas, 1997).

Protokol eksekusi dalam KHGT Times mencari dua kondisi terpisah:

1. **Yaumul Kiblat Utama (Tahunan):** Terjadi ketika Deklinasi Matahari sama persis dengan Lintang geografis Ka'bah (sekitar 21.42° LU), yang umumnya jatuh pada tanggal 28 Mei dan 16 Juli. Pada kondisi ini, bayangan dari setiap benda tegak di seluruh belahan bumi yang mengalami siang hari akan mengarah lurus menuju Makkah.
2. **Rashdul Kiblat Harian:** Waktu spesifik di hari apa pun sepanjang tahun di mana Azimuth Matahari sama dengan Azimuth Kiblat lokal, atau berselisih tepat 180 derajat.

7.3.3. Arsitektur Kode dan Koreksi Toposentris

Dalam implementasinya menggunakan Python dan *Skyfield*, sistem sama sekali tidak memodelkan bumi sebagai bola bundar sempurna, melainkan sebagai *Oblate Spheroid* berdasarkan parameter geodetik WGS84. Hal ini menjadi syarat wajib karena kemiringan tongkat istiwa absolut di permukaan bumi dunia nyata akan terpengaruh secara mikro oleh kelengkungan ekuatorial.

Python

```
def calculate_mizwala_shadow(observer_loc, t_current, gnomon_height):
    # Mengambil posisi toposentris semu (diperkuat oleh refraksi atmosfer lokal)
    sun_apparent = observer_loc.at(t_current).observe(sun).apparent()
    alt, az, distance = sun_apparent.altaz()

    alt_deg = alt.degrees
    az_deg = az.degrees

    # Pencegahan matematis pembagian dengan nol saat matahari menembus horizon
    if alt_deg <= 0:
        return None, None

    import math
    # Kalkulasi panjang bayangan aktual
    shadow_length = gnomon_height / math.tan(math.radians(alt_deg))

    # Kalkulasi azimuth arah bayangan
    shadow_azimuth = (az_deg + 180.0) % 360.0
```

```
    return shadow_length, shadow_azimuth
```

Struktur kode di atas dieksekusi oleh mesin sistem dalam *looping* kontinu berkecepatan tinggi, yang memungkinkan modul simulasi GUI untuk memproyeksikan animasi pergerakan bayangan tongkat melintasi piringan penunjuk (*dial plate*) dengan latensi sangat rendah. Integrasi algoritmik ini berhasil menjembatani tingkat akurasi ephemeris saintifik modern dengan kemurnian instrumen observasional yang telah menjadi warisan intelektual peradaban Islam.

DAFTAR PUSTAKA BAB 7

Abdali, S. K. (1997). *The correct qibla*. American Trust Publications.

Duffett-Smith, P., & Zwart, J. (2011). *Practical Astronomy with your Calculator or Spreadsheet* (4th ed.). Cambridge University Press.

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Ilyas, M. (1997). *Islamic Calendar, Times & Qibla*. Berita Publishing.

King, D. A. (1986). *Islamic Mathematical Astronomy*. Variorum Reprints.

King, D. A. (2004). *In Synchrony with the Heavens: Studies in Astronomical Timekeeping and Instrumentation in Medieval Islamic Civilization*. Brill.

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl:1907.024.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Torge, W., & Müller, J. (2012). *Geodesy* (4th ed.). De Gruyter.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 8: MEKANIKA SYZYGY: GERHANA DAN ANALEMMA

Selain menentukan awal bulan dan jadwal waktu salat, komputasi astronomi Islam modern juga dihadapkan pada tuntutan untuk memprediksi fenomena fisis luar biasa yang bernilai ibadah (seperti Salat Khusuf dan Kusuf), serta memodelkan lintasan geometri matahari secara komprehensif. Bab ini akan menguraikan bagaimana Modul Analisis Gerhana dan Generator Analemma pada **KHGT Times V7.4** memanfaatkan mesin `Skyfield` untuk memecahkan kompleksitas mekanika benda langit tingkat lanjut.

8.1. Deteksi Otomatis Gerhana Matahari dan Bulan

Fenomena gerhana (baik matahari maupun bulan) terjadi secara periodik ketika Matahari, Bumi, dan Bulan berada dalam satu garis lurus secara tiga dimensi, suatu keadaan yang dalam astronomi dinamakan *syzygy* (Espenak & Meeus, 2006). Namun, tidak setiap *syzygy* (ijtimak/purnama) menghasilkan gerhana karena orbit Bulan miring sekitar 5.14^\circ terhadap bidang orbit Bumi (ekliptika). Gerhana hanya terjadi ketika *syzygy* bertepatan dengan titik simpul orbit bulan (*lunar node*).

8.1.1. Mencari Jarak Sudut Minimum (Separation) di Titik Simpul

Pustaka astronomi klasik umumnya menggunakan persamaan Bessel (*Besselian elements*) untuk meramalkan gerhana. KHGT Times V7.4 mengambil pendekatan numerik langsung yang lebih tangguh dan mudah diimplementasikan, yaitu mencari jarak sudut geometris absolut (*angular separation*) terdekat antara piringan Matahari dan Bulan pada waktu-waktu kritis.

Untuk Gerhana Matahari, sistem pertama-tama mencari semua waktu ijtimak (*New Moon*) di sepanjang tahun. Pada setiap titik ijtimak, program membangkitkan larik waktu resolusi sangat tinggi (pm 0.25 hari dari pusat ijtimak) dan menghitung jarak sudut (*separation*) geosentris secara iteratif:

Python

```
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)

# Menghitung sudut separasi (jarak) Matahari dan Bulan
seps =
e_pos.observe(sun).apparent().separation_from(e_pos.observe(moon).apparent())
.degrees
min_idx = np.argmin(seps)
```

Jika `seps[min_idx]` lebih kecil dari ambang batas toleransi (sekitar 1.6^\circ, yakni jumlah jari-jari Matahari, jari-jari Bulan, ditambah toleransi paralaks maksimum), maka dipastikan telah terjadi kontak piringan (gerhana matahari) (Meeus, 1998).

8.1.2. Kalkulasi Geometri Umbra dan Penumbra Bumi

Untuk Gerhana Bulan, prinsipnya serupa namun secara geometris lebih kompleks karena melibatkan proyeksi bayangan kerucut Bumi ke jarak orbit Bulan. Berbeda dengan gerhana matahari, visibilitas gerhana bulan tidak bergantung pada paralaks pengamat (gerhana bulan terlihat sama di seluruh wilayah bumi yang sedang mengalami malam hari). Oleh karena itu, perhitungan dilakukan secara murni **Geosentris**.

Alih-alih mencari separasi matahari dan bulan, sistem mencari titik potong antara posisi Bulan dan titik *Anti-Matahari* (posisi bayangan bumi).

Python

```
# Koordinat Matahari
s_ra, s_dec, s_dist = s_geo.radec()

# Membalik vektor matahari 180 derajat (Titik Anti-Matahari / Pusat Bayangan
Bumi)
as_ra = (s_ra.hours + 12.0) % 24.0
as_dec = -s_dec.degrees

# Jarak sudut bulan terhadap pusat bayangan bumi
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)
```

Selanjutnya, program menghitung jari-jari bayangan gelap Bumi (*Umbra*) dan bayangan samar (*Penumbra*) pada jarak Bulan menggunakan trigonometri dasar berbasis radius ekuatorial bumi (6378.14 km):

- r_umbra = (p_m + p_s - s_s) * 1.02
- r_penumbra = (p_m + p_s + s_s) * 1.02

Keterangan Simbol:

- **r_umbra**: Jari-jari bayangan inti (Umbra).
- **r_penumbra**: Jari-jari bayangan tambahan (Penumbra).
- **p_m**: Paralaks horizontal Bulan (*Moon's horizontal parallax*).
- **p_s**: Paralaks horizontal Matahari (*Sun's horizontal parallax*).
- **s_s**: Semidiameter atau jari-jari sudut Matahari.
- **1.02**: Faktor koreksi (sekitar 2%) untuk memperhitungkan pengaruh atmosfer bumi yang memperbesar bayangan.

Penjelasan Fungsi:

Di mana p adalah paralaks ekuatorial horizontal dan s adalah *semi-diameter* (jari-jari piringan). Konstanta 1.02 ditambahkan sebagai empirisme astronomis karena atmosfer bumi memperbesar ukuran fisik bayangan umbra sekitar 2% (Espenak & Meeus, 2006). Jika `sep` lebih kecil dari r_{umbra}, maka bulan memasuki fase gerhana umbral (sebagian/total).

8.2. Proyeksi Jalur Totalitas (Export KML)

Salah satu fitur paling kompleks dari KHGT Times V7.4 adalah kemampuannya men- *generate* secara otonom dokumen Google Earth (.kml) yang mendelineasi jalur gerhana total di permukaan bumi.

8.2.1. Vektor Posisi 3D (*Center Line*) Bayangan Bulan di Permukaan Bumi

Pada gerhana matahari total/cincin, titik pusat bayangan umbra bulan (Sumbu Silinder Bulan) memotong permukaan bumi membentuk lintasan melengkung. Untuk menemukan titik potong tiga dimensi ini, KHGT Times menggunakan representasi vektor *xyz* spasial murni.

1. Program menghitung vektor posisi pusat Bulan (M) dan pusat Matahari (S).
2. Mencari vektor arah sinar bayangan (V = M - S) dan melakukan normalisasi (hat{v}).
3. Permukaan bumi dimodelkan sebagai bola dengan jari-jari R_E (rata-rata geometris). Garis sumbu bayangan dapat ditulis sebagai persamaan parametrik: P(k) = M + \hat{v} \cdot k. Persamaan ini disubstitusikan ke dalam persamaan bola |P|^2 = R_E^2, menghasilkan persamaan kuadratik Delta:

Python

```
V = M - S
v_hat = V / np.linalg.norm(V, axis=0)

b = 2.0 * np.sum(M * v_hat, axis=0)
c = np.sum(M**2, axis=0) - R_E**2
delta = b**2 - 4.0 * c
```

Hanya titik waktu dengan `delta >= 0` yang menandakan bahwa bayangan inti (umbra) bulan "menghantam" daratan atau samudra di Bumi (jika negatif, bayangan meleset ke luar angkasa) (Urban & Seidelmann, 2013).

8.2.2. Konversi *Subpoint* Geocentric ke File Google Earth

Titik potong P yang masih dalam format ruang Cartesian (X, Y, Z) dikonversi kembali ke koordinat geografis lintang dan bujur permukaan ellipsoid menggunakan model WGS84 dari *Skyfield*:

Python

```
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")
```

Rangkaian koordinat ini kemudian dibungkus menggunakan sintaks *Markup Language* XML (KML) dengan tag `<LineString>`. Output akhir dari proses algoritmik ini adalah *file* yang dapat langsung di-*drag and drop* ke aplikasi Google Earth Pro, merepresentasikan jalur bayangan gerhana (*Path of Totality*) yang siap digunakan para pengejar gerhana (*eclipse chasers*).

8.3. Generator Analemma Matahari

Analemma adalah kurva geometris menyerupai angka 8 yang dibentuk oleh posisi Matahari di langit, apabila difoto dari titik kordinat bumi yang sama, tepat pada waktu jam yang sama (misal 12:00 siang lokal), selama 365 hari dalam setahun. Fenomena analemma terjadi akibat dua asimetri orbital: kemiringan sumbu rotasi bumi (23.4^\circ) yang mendikte perubahan Altitude, dan eksentrisitas orbit bumi (elips) yang mendikte perubahan Azimuth karena fluktuasi *Equation of Time* (Hughes et al., 1989).

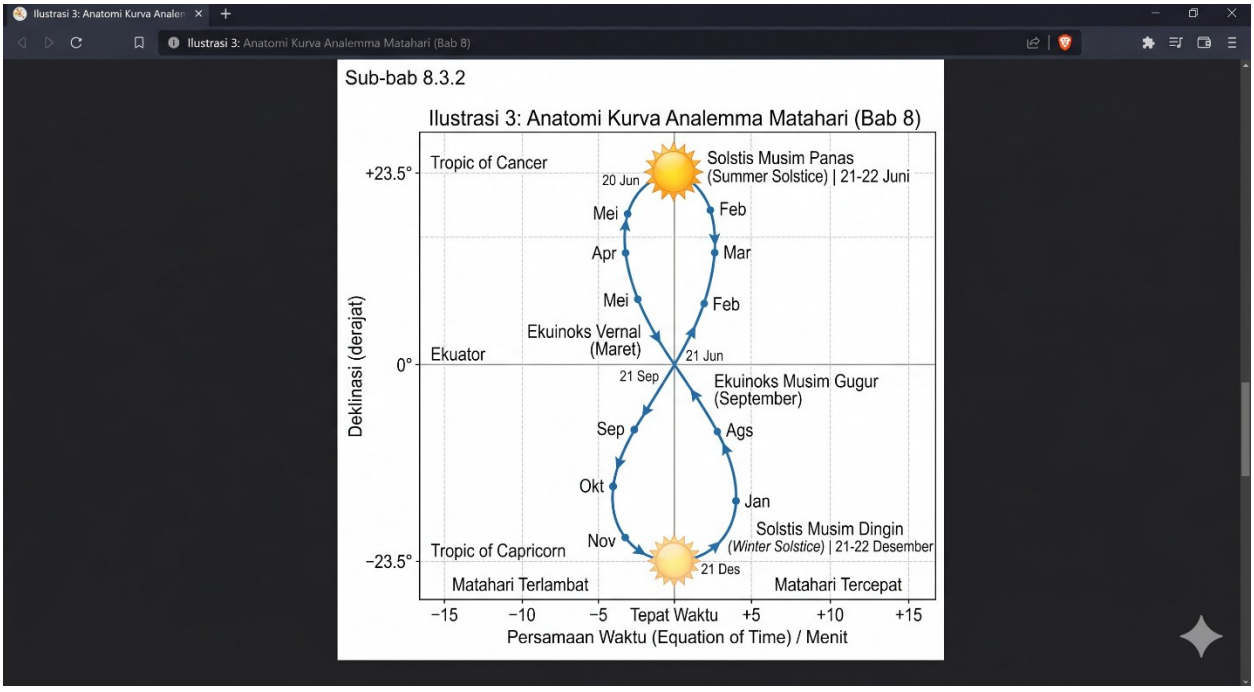
8.3.1. Ekstraksi Altitude dan Azimuth pada Jam Tetap Selama 365 Hari

Untuk membangun grafik analemma, KHGT Times V7.4 mengatur iterasi ephemeris sebanyak 365 hari dengan mempertahankan konstan variabel jam (`h_local`). Pilihan `grid_step` sebesar 5 hari (sekitar 73 titik sampel per tahun) dipilih sebagai optimasi kecepatan *render* Matplotlib tanpa mengorbankan kualitas kurva angka 8 yang terbentuk.

Python

```
for day in range(0, 365, 5):
    date_dt = datetime.datetime(y, 1, 1) + datetime.timedelta(days=day)

    # Waktu dipatok konstan pada h_local
    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()
```

8.3.2. Analisis Titik Ekstrem (Solstice) dan Ekuator (Equinox)



Selain sekadar mencetak titik, mesin ini diinstruksikan untuk mendeteksi parameter fisis batas maksimum dan minimum (*solstis*) menggunakan NumPy:

Python

```
idx_max = np.argmax(alts)  # Summer Solstice (Utara)
idx_min = np.argmin(alts)  # Winter Solstice (Selatan)
```

Metode pembacaan pola matriks ini menjadikan Modul Analemma KHGT Times sebagai instrumen edukasi astronomi yang sangat representatif untuk memahami konsep *Mean Solar Time*, di mana rentang lebar horizontal dari kurva analemma merepresentasikan durasi kompensasi menit dari nilai *Equation of Time* yang berfluktuasi sepanjang tahun.

DAFTAR PUSTAKA BAB 8

Espenak, F., & Meeus, J. (2006). *Five Millennium Canon of Solar Eclipses: -1999 to +3000*. NASA Goddard Space Flight Center.

Hughes, D. W., Yallop, B. D., & Hohenkerk, C. Y. (1989). The equation of time. *Monthly Notices of the Royal Astronomical Society*, 238(4), 1529-1535.

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 9: SIMULATOR 3D DAN ANIMASI INTERAKTIF

Representasi data astronomi dalam bentuk tabel numerik (ephemeris) sering kali memiliki keterbatasan dalam membangun intuisi visual, terutama ketika fenomena yang diamati melibatkan interaksi spasial yang kompleks seperti gerhana, *transit* planet, atau posisi hilal di atas cakrawala 360 derajat. Mengatribusikan nilai asensio rekta, deklinasi, altitude, dan azimuth ke dalam sebuah kanvas visual tiga dimensi merupakan jembatan kognitif yang sangat esensial (Fluke et al., 2006).

Oleh karena itu, **KHGT Times V7.4** tidak hanya berfungsi sebagai mesin penghitung angka, tetapi juga dilengkapi dengan *Graphical User Interface* (GUI) yang memuat modul Simulator 3D dan Animasi Interaktif. Modul ini secara fundamental mentransformasikan larik data spasial menjadi model fisika visual yang beroperasi secara *real-time*.

9.1. Transformasi Vektor ke Ruang 3D (Matplotlib)

Membangun lingkungan tiga dimensi dari data astronomi bola mensyaratkan pemahaman yang solid mengenai aljabar linear dan geometri spasial. KHGT Times V7.4 memanfaatkan *library* Matplotlib untuk melakukan *rendering* grafis. Matplotlib pada dasarnya adalah pustaka grafik 2D, namun ia memiliki kapabilitas ekstensi `mplot3d` untuk memproyeksikan data 3D. Menariknya, pada modul *Geocentric Space System* (Modul 16), pengembang aplikasi merancang mesin proyeksi (*projection engine*) dan kalkulasi *Z-Buffer* secara mandiri murni menggunakan fungsi trigonometri dasar tanpa bergantung sepenuhnya pada modul 3D bawaan Matplotlib, guna mencapai efisiensi rotasi (*rendering speed*) yang lebih tinggi.

9.1.1. Konversi Koordinat Alt/Az ke Cartesian (X, Y, Z)

Pada Modul 19 (Simulasi Ephemeris 3D), pengguna disajikan representasi *toposentris*—yakni bola langit yang berpusat pada mata pengamat di permukaan bumi. Output dari mesin astrometri *Skyfield* berupa besaran Ketinggian (*Altitude*, Alt) dan Azimuth (Az). Agar titik-titik bulan dan matahari dapat diletakkan di dalam ruang kanvas Cartesian 3D, koordinat bola (*spherical coordinates*) tersebut harus direduksi menjadi sumbu ortogonal x, y, z.

Dalam astronomi, jika kita asumsikan pengamat berada di titik origin (0,0,0), sumbu y positif mengarah ke Utara sejati (Az = 0^\circ), sumbu x positif mengarah ke Timur (Az = 90^\circ), dan sumbu z mewakili titik Zenith (Alt = 90^\circ), maka persamaan transformasinya adalah (Duffett-Smith & Zwart, 2011):

$$x = r \sin(Az) \cos(Alt)$$

$$y = r \cos(Az) \cos(Alt)$$

$$z = r \sin(Alt)$$

Persamaan ini diterjemahkan secara presisi ke dalam blok kode internal KHGT Times:

Python

```
def eph3d_r_ke_xyz(self, alt, az, r=10):
    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
```

Variabel jari-jari bola langit imajiner (r) ditetapkan secara statis sebesar 10 satuan absolut, mengingat dalam proyeksi pandangan *toposentris* lokal, pengamat memandang langit seolah-olah semua benda kosmik (bintang, bulan, matahari) menempel pada dinding kubah dengan jarak seragam tak terhingga (*celestial sphere*). Setelah dikonversi, titik kordinat spasial tersebut diumpankan ke objek Matplotlib melalui fungsi interpolatif `.set_3d_properties()`.

9.1.2. Proyeksi Kamera Kustom (Z-Buffer dan Pseudo-3D Rotation)

Kendala utama Matplotlib dalam memproses animasi beresolusi tinggi adalah beban *overhead* saat merender ulang seluruh komponen *figure* 3D setiap kali layar diputar menggunakan *mouse*. Untuk mengatasi limitasi *frame rate* pada Modul 16 (Sistem Tata Surya Geosentris), KHGT Times menggunakan Matplotlib sebagai kanvas 2D biasa (menggunakan *widget* `Canvas` Tkinter) namun menyuntikkan algoritma proyeksi perspektif pseudo-3D (*Pseudo-3D Projection*) secara matematis.

Ketika pengguna mengklik dan menggeser layar (*mouse drag*), sistem tidak merotasi Matplotlib *axes*. Sistem memperbarui sudut kamera (theta_x dan theta_y) dan melakukan perkalian matriks rotasi Euler pada setiap vektor posisi planet (x, y, z). Proyeksi transformasi ortografi ke bidang monitor 2D (px, py) dihitung melalui persamaan yang memperhitungkan faktor kedalaman (*depth factor*), sehingga objek yang menjauh terlihat mengecil secara alami:

Persamaan Transformasi Rotasi Koordinat

1. Rotasi terhadap Sumbu X (Tahap 1):

- **x1** = $x \cdot \cos(\theta x) - z \cdot \sin(\theta x)$
- **z1** = $x \cdot \sin(\theta x) + z \cdot \cos(\theta x)$

2. Rotasi terhadap Sumbu Y (Tahap 2):

- **y2** = $y \cdot \cos(\theta y) - z1 \cdot \sin(\theta y)$
- **z2** = $y \cdot \sin(\theta y) + z1 \cdot \cos(\theta y)$

Keterangan:

- **x, y, z**: Koordinat awal (input).

- **θx, θy**: Sudut rotasi pada sumbu x dan y.
- **x1, z1**: Hasil antara (intermediate) setelah rotasi pertama.
- **y2, z2**: Hasil akhir setelah rotasi kedua.

Python

```
factor = 500 / (500 + z2)
px = x1 * factor + (width / 2)
py = -y2 * factor + (height / 2)
return px, py, z2
```

Lebih krusial lagi, untuk memastikan benda langit yang berada di depan menutupi benda yang berada di belakang (misalnya saat Bulan berada di depan Bumi), sistem mengimplementasikan varian **Algoritma Pelukis (Painter's Algorithm)** melalui manajemen *Z-Buffer* mandiri (Hughes et al., 2014).

Semua objek ditampung ke dalam senarai (`draw_list`) bersama dengan nilai kedalaman akhirnya (z_2). Sebelum direnderer ke layar antarmuka, *list* tersebut diurutkan secara menurun (*descending*) berdasarkan properti z:

Python

```
# Algoritma Z-Buffer: Urutkan objek dari yang terjauh ke yang terdekat
draw_list.sort(key=lambda x: x[1][2], reverse=True)
```

Dengan mengurutkannya, mesin Matplotlib akan melukis objek yang terjauh (latar belakang kosmik) terlebih dahulu, dan melukis objek terdekat pada akhir iterasi, menimpali piksel gambar sebelumnya dengan sempurna. Optimalisasi komputasi grafis inilah yang menjamin kelancaran interaksi 60 FPS pada simulator tata surya KHGT Times, bahkan pada arsitektur komputer berspesifikasi rendah.

9.2. Live Tracker Toposentris (Bola Langit Pengamat)

Transformasi visual dari ruang 3D konseptual (geosentris) menuju ruang observasi praktis memerlukan pergeseran kerangka acuan (*reference frame*). Modul *Live Animasi* (Modul 15) pada **KHGT Times V7.4** dirancang untuk mensimulasikan bola langit (*celestial sphere*) murni dari perspektif toposentris, yakni dari mata pengamat di permukaan bumi. Sistem ini melukiskan garis horizon, letak azimut, serta melacak pergerakan Matahari dan Bulan dalam waktu waktu nyata (*real-time*), menjadikannya instrumen virtual yang setara dengan fungsi planetarium modern (Gargano et al., 2015).

9.2.1. Sinkronisasi Data Astronomi dengan Jam Sistem (Real-time Threading)

Tantangan utama dalam membangun *live tracker* adalah menyinkronkan waktu sistem operasi (komputer) dengan skala waktu dinamis ephemeris (Terrestrial Time/TT) tanpa menyebabkan antarmuka pengguna (*Graphical User Interface* / GUI) menjadi tidak responsif atau membeku (*freeze*).

Alih-alih menggunakan *blocking loop* (seperti `while True`), KHGT Times memanfaatkan arsitektur pemrograman *event-driven* bawaan *Tkinter* melalui metode `.after()`. Pemanggilan ini menugaskan sistem operasi untuk mengeksekusi ulang fungsi pembaruan layar (`update_animation`) setiap 1000 milidetik (1 detik), menghasilkan animasi dengan laju 1 *Frame Per Second* (FPS) yang sangat hemat daya CPU.

Pada setiap detiknya, sistem menangkap waktu absolut dan melakukan komputasi posisi:

Python

```
if getattr(self, 'anim_is_live', True):
    t_sim = self.ts.now()  # Mengambil waktu sistem real-time ke dalam objek Skyfield
else:
    t_sim = self.anim_custom_time # Mode playback/kustom waktu

# Evaluasi Toposentrik (Koreksi paralaks terhitung otomatis)
observer = earth + wgs84.latlon(lat, lon)
astro_sun = observer.at(t_sim).observe(sun).apparent()
alt_sun, az_sun, _ = astro_sun.altaz()

astro_moon = observer.at(t_sim).observe(moon).apparent()
alt_moon, az_moon, _ = astro_moon.altaz()
```

Nilai *Altitude* dan *Azimuth* yang dihasilkan secara presisi kemudian dipetakan ke dalam kanvas 2D (`self.anim_canvas`). Sistem membagi lebar layar menjadi 360^\circ bujur langit. Menariknya, sistem ini juga mengalkulasi ketinggian ufuk (*horizon*) dinamis dan menyuntikkan efek refraksi geometris dengan menambahkan nilai statis +0.833^\circ pada ketinggian piringan matahari (`alt_top_sun = alt_s_deg + 0.833`), memastikan bahwa piringan matahari tenggelam tepat di garis ufuk GUI pada saat waktu Maghrib tiba, merepresentasikan ukuran semi-diameter matahari dan refraksi atmosfer aktual (Schaefer, 1993).

9.2.2. Render Piringan Matahari dan Fase Bulan Dinamis (Pillow Masking)

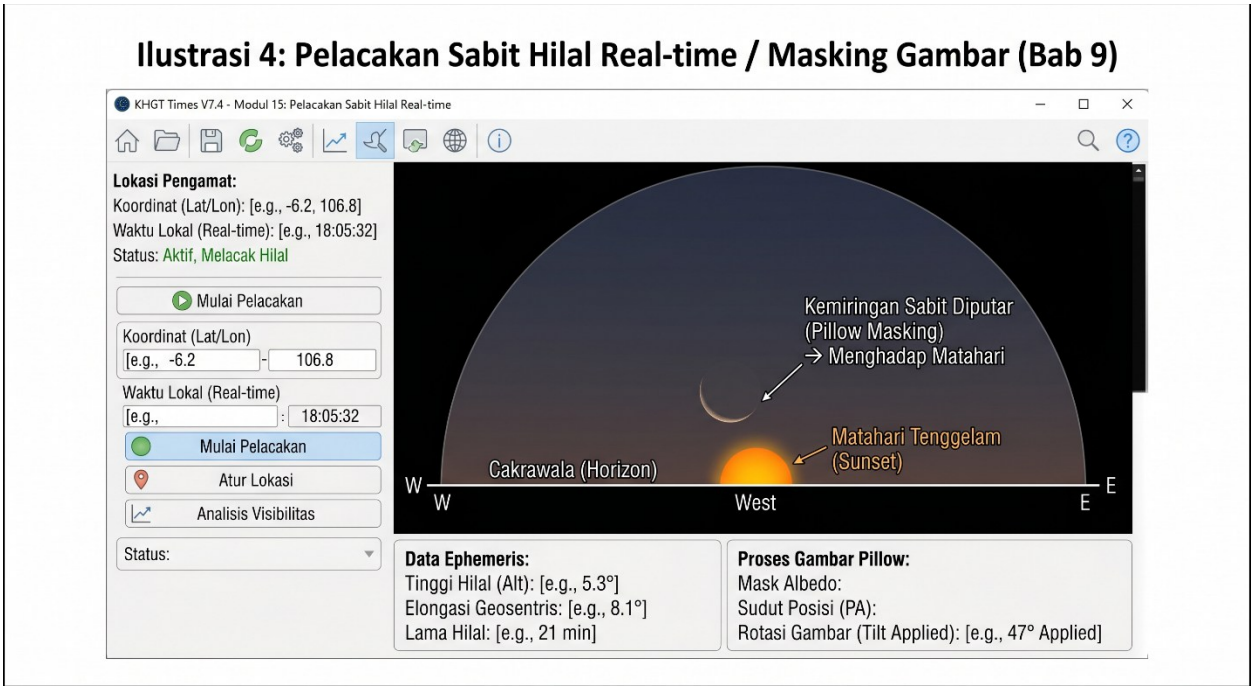
Bagian paling revolusioner dari modul *Live Tracker* KHGT Times V7.4 adalah pelukisan (*rendering*) wujud bulan. Bulan tidak sekadar digambar sebagai lingkaran putih statis, melainkan dirender sesuai dengan fase iluminasinya dan rotasi kemiringan sabitnya relatif terhadap matahari di langit.

Fase bulan (k) dikalkulasi secara matematis melalui selisih bujur ekliptika (lambda) antara Bulan dan Matahari:

Phase_Angle = (λ_moon - λ_sun) mod 360°

Keterangan Simbol:

- **Phase_Angle / Sudut_Fase**: Sudut yang menentukan fase bulan (digunakan untuk mengetahui posisi bulan terhadap matahari).
- **λ_moon (λ_bulan)**: Bujur ekliptika Bulan (*Ecliptic Longitude of the Moon*).
- **λ_sun (λ_matahari)**: Bujur ekliptika Matahari (*Ecliptic Longitude of the Sun*).
- **mod 360°**: Operasi modulo (sisa bagi) untuk memastikan hasil perhitungan sudut selalu berada dalam rentang 0° hingga 360°.



Untuk merender gambar sabit secara halus, program menggunakan pustaka manipulasi citra **Pillow (PIL)**. Sistem memuat himpunan *sprite* gambar bulan (`m000.png` hingga `m359.png`), dan memilih gambar yang tepat berdasarkan sudut fase bulat yang dihasilkan. Namun, memuat gambar saja tidak cukup; garis terminasi (batas terang-gelap bulan) harus miring tegak lurus mengarah langsung ke pusat piringan matahari.

Di sinilah KHGT Times V7.4 mengeksekusi trigonometri 2D Cartesian untuk mencari sudut kemiringan (*tilt angle*) antara koordinat piksel matahari (x_{sun}, y_{sun}) dan bulan (x_{moon}, y_{moon}):

Python

```
dx = x_sun - x_moon
dy = y_sun - y_moon
# Menghitung sudut kemiringan dalam radian lalu dikonversi ke derajat
sudut_kemiringan = math.degrees(math.atan2(dy, dx))
```

```
if moon_idx >= 180:
    sudut_kemiringan -= 180

# Merotasi gambar bulan dengan algoritma interpolasi kualitas tinggi (Bicubic)
moon_img                  =                  moon_img.rotate(-sudut_kemiringan,
resample=Image.Resampling.BICUBIC)

# Teknik Alpha Masking agar background bulan transparan
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)
```

Fungsi `math.atan2(dy, dx)` memastikan sudut kemiringan terbaca di keempat kuadran spasial secara akurat. Penggunaan interpolasi *Bicubic* (`Image.Resampling.BICUBIC`) pada saat rotasi mencegah piksel gambar bulan menjadi pecah (*aliasing*). Selanjutnya, proses *Alpha Masking* memotong tepi gambar yang berbentuk persegi menjadi lingkaran transparan sempurna, memungkinkannya digambar tumpang-tindih (*overlay*) di atas kanvas bintang-bintang (*starfield*) tanpa meninggalkan kotak latar belakang hitam.

Teknik komputasi grafis hibrida ini—yang memadukan keakuratan *Skyfield* dan ketajaman optis *Pillow*—menghasilkan simulasi visibilitas hilal yang sangat realistis. Ini memberikan kemudahan luar biasa bagi *perukyat* (observator) untuk mengetahui di mana letak persis dan bagaimana bentuk orientasi sabit hilal di langit sesaat setelah *sunset* tiba.

9.3. Simulasi Tata Surya Geosentris

Meskipun model heliosentris (Matahari sebagai pusat) merepresentasikan realitas fisik tata surya kita, seluruh observasi astronomi Islam—baik hisab awal bulan maupun jadwal waktu salat—secara mutlak bertumpu pada kerangka acuan geosentris (Bumi sebagai pusat). Modul 16 pada **KHGT Times V7.4** mendesain ulang arsitektur *rendering* Matplotlib untuk menciptakan "Simulator Tata Surya Geosentris", sebuah ruang virtual interaktif di mana Bumi ditetapkan secara statis di titik origin koordinat $(0, 0, 0)$, sementara Bulan, Matahari, dan planet-planet lainnya mengorbit mengelilinginya (Montenbruck & Pfleger, 2000).

9.3.1. Kompresi Spasial dan Skala Jarak Logaritmik (Non-Linear Scaling)

Tantangan komputasi visual terbesar dalam membangun simulator geosentris adalah perbedaan skala jarak yang ekstrem. Jarak rata-rata Bumi ke Bulan adalah sekitar 384.400 km, sedangkan jarak rata-rata Bumi ke Matahari (*1 Astronomical Unit*) adalah sekitar $149.597.870$ km. Rasio jarak ini hampir mencapai $1:400$. Jika sistem merender objek-objek ini ke dalam layar komputer menggunakan skala linier yang sebenarnya (1:1), maka saat piringan Matahari terlihat di batas layar, orbit Bulan akan mengecil menjadi kurang dari satu piksel dan tidak mungkin diobservasi oleh pengguna manusia (Shirley & Marschner, 2009).

Untuk memecahkan kebuntuan visual ini, KHGT Times mengimplementasikan algoritma **Kompresi Skala Non-Linier** (pseudo-logaritmik). Sistem menetapkan jarak visual Bulan sebagai standar konstan (misalnya $r = 10$ unit grafis), lalu secara matematis mengompresi jarak Matahari

dan planet lain sehingga mereka tampak berada pada jarak yang proporsional di layar, tanpa mengubah keakuratan sudut Asensio Rekta (RA) dan Deklinasi (Dec) mereka.

Python

```
# Ekstraksi jarak absolut (km) dari mesin Skyfield
d_moon = moon_geo.distance().km
d_sun = sun_geo.distance().km

# Algoritma Kompresi Spasial untuk Visualisasi GUI
# Radius visual Bulan dipatok pada skala 10 unit grafis
vis_r_moon = (d_moon / 384400.0) * 10.0

# Jarak Matahari ditekan secara logaritmik agar masuk ke dalam field of view
(FOV)
vis_r_sun = (math.log10(d_sun) / math.log10(149597870.0)) * 30.0

# Konversi kembali ke XYZ menggunakan sudut RA dan Dec yang tidak diubah
x_sun, y_sun, z_sun = self.spherical_to_cartesian(ra_sun, dec_sun, vis_r_sun)
```

Melalui pemisahan antara "Jarak Geometris (untuk kalkulasi ephemeris)" dan "Jarak Visual (untuk *rendering* layar)", simulator KHGT Times mampu menyajikan konfigurasi *syzygy* (ijtimak/purnama) yang dapat diobservasi mata dengan jelas dalam satu *frame*, menjaga nilai edukatif perangkat lunak tanpa mengorbankan fondasi fisika astrometrinya.

9.3.2. Rendering Jejak Orbit (Orbital Trails) dan Vektor Ekliptika

Benda langit tidak hanya digambar sebagai titik (titik kordinat saat itu), tetapi sistem juga mengkalkulasi dan melukiskan "jejak orbit" (*orbital trails*) untuk memvisualisasikan mekanika benda langit selama satu bulan penuh (satu lunasi). Hal ini sangat krusial untuk menjelaskan mengapa bulan tidak selalu menutupi matahari (gerhana) pada setiap fase ijtimak.

Untuk menghasilkan jejak orbit resolusi tinggi, sistem memicu fungsi multithreading yang meminta *Skyfield* membangkitkan senarai matriks posisi (`linspace`) selama ± 15 hari dari waktu simulasi saat ini:

Python

```
# Membangkitkan array waktu mundur 15 hari dan maju 15 hari
t_array = self.ts.tt_jd(np.linspace(t_sim.tt - 15, t_sim.tt + 15, 300))

# Ekstraksi array vektor posisi geosentris
pos_moon_array = earth.at(t_array).observe(moon).apparent()
ra_arr, dec_arr, dist_arr = pos_moon_array.radec()

# Looping untuk mengonversi array ke garis grafis 3D
trail_coords = []
for i in range(len(ra_arr.hours)):
    x,      y,      z      =      self.spherical_to_cartesian(ra_arr.hours[i]*15,
dec_arr.degrees[i], vis_r_moon)
    trail_coords.append((x, y, z))
```

Garis orbit Bulan kemudian diproyeksikan dengan ketebalan 1 piksel berwarna abu-abu, sementara bidang Ekliptika (orbit semu Matahari) diproyeksikan dengan garis kuning. Pertemuan antara garis abu-abu dan garis kuning di layar 3D menunjukkan posisi **Lunar Nodes** (Titik Simpul Naik/Turun).

Interaktivitas yang memungkinkan pengguna untuk memutar ruang 3D, memperbesar, dan melihat jarak pemisah di titik simpul orbit inilah yang menjadikan Modul Simulasi KHGT Times V7.4 sebagai mahakarya rekayasa perangkat lunak pedagogis. Pengguna tidak lagi "membayangkan" proses konjungsi dari balik deretan angka, melainkan "menyaksikan" mekanika kosmik tersebut bekerja secara geometris.

DAFTAR PUSTAKA BAB 9

Duffett-Smith, P., & Zwart, J. (2011). *Practical Astronomy with your Calculator or Spreadsheet* (4th ed.). Cambridge University Press.

Fluke, C. J., Bourke, P. D., & O'Donovan, D. (2006). Future directions in astronomy visualization. *Publications of the Astronomical Society of Australia*, 23(1), 12-24.

Gargano, M., et al. (2015). Planetariums and digital domes: A new era for astronomical visualization. *Journal of Science Communication*, 14(2).

Hughes, J. F., et al. (2014). *Computer Graphics: Principles and Practice* (3rd ed.). Addison-Wesley Professional.

Montenbruck, O., & Pfleger, T. (2000). *Astronomy on the Personal Computer* (4th ed.). Springer-Verlag.

Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311-361.

Shirley, P., & Marschner, S. (2009). *Fundamentals of Computer Graphics* (3rd ed.). A K Peters/CRC Press.

BAB 10: INTEGRASI KECERDASAN BUATAN (WINAI) DALAM FIKIH FALAK

Evolusi perangkat lunak falak di abad ke-21 tidak lagi bertumpu eksklusif pada komputasi matematis benda langit. Kompleksitas implementasi kalender global (KHGT) sering kali memunculkan kebingungan kognitif dan pertanyaan fikih (yurisprudensi) di kalangan pengguna awam maupun akademisi. Untuk menjembatani jurang antara data astrometri mentah dan pemahaman syariat, **KHGT Times V7.4** mempelopori integrasi *Large Language Model* (LLM) langsung ke dalam antarmuka desktop melalui modul **WinAI (Aplikasi AI Windows)**. Modul ini tidak sekadar menjadi *chatbot* generik, melainkan dirancang dengan arsitektur khusus sebagai Asisten Fikih Falak.

10.1. Arsitektur Asisten Fikih AI

Modul WinAI mengandalkan model bahasa generatif mutakhir (dalam hal ini dispesifikasikan pada `gemini-2.5-flash`) yang diakses secara langsung melalui *Application Programming Interface* (API) RESTful. Tantangan utama dalam mengintegrasikan AI pihak ketiga ke dalam aplikasi desktop (*client-side*) yang didistribusikan secara massal adalah manajemen *rate limit* (kuota penggunaan) dan penjagaan batas konteks (*guardrails*). KHGT Times memecahkan masalah ini melalui rekayasa perangkat lunak pada tingkat koneksi dan *prompting*.

10.1.1. Sistem Rotasi Kunci API (*API Key Rotation*) untuk Ketersediaan Kuota

Model AI generatif yang diakses secara publik umumnya memberlakukan pembatasan permintaan (*rate limiting*) yang sangat ketat per menit (RPM) maupun per hari (RPD) untuk setiap kunci API (*API Key*). Jika sebuah perangkat lunak desktop menggunakan satu *API Key* statis secara *hardcoded* untuk melayani ratusan pengguna yang melakukan kueri secara bersamaan, sistem akan langsung mengalami kelumpuhan akibat galat HTTP 429 (*Too Many Requests*) (Zheng et al., 2023).

Untuk mengakali *bottleneck* ini tanpa memerlukan arsitektur *server-side* berbayar (*enterprise tier*), arsitek KHGT Times V7.4 merancang sistem **API Key Rotation** (Rotasi Kunci API) berbasis *array* sirkuler. Perhatikan implementasi *fault-tolerance* pada blok kode berikut:

Python

```
self.winai_api_keys = [
    'AIzaSyAVIBjSEWKdyeAYnhYue6Zw6tSJKzQpfEw',
    'AIzaSyB8jOg5MPMrbpIWTmRT51rydJYLMFhRaXs',
    # ... [Terdapat lebih dari 30 API Keys yang didaftarkan] ...
]

# Membaca indeks sukses terakhir dari penyimpanan lokal
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)
```

```
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}:generateContent?key={api_key}"

    res = requests.post(api_url, headers={'Content-Type':
'application/json'}, json=payload, timeout=20)

    if res.status_code == 200:
        # Jika sukses, simpan indeks ini agar eksekusi berikutnya dimulai
dari sini
        with open(self.winai_tracker_file, 'w') as f:
            f.write(str(idx))
        break
    else:
        # Logika Rotasi Anti-Macet: Lanjut ke API Key berikutnya jika kuota
habis (429) atau mati (403)
        if res.status_code in [429, 403, 400]:
            continue
```

Algoritma di atas mengimplementasikan *Round-Robin Scheduling*. Sistem mencatat *index* kunci API terakhir yang berhasil mengeksekusi *prompt* ke dalam sebuah *file tracker* lokal (`last_successful_index_winai.txt`). Ketika pengguna mengajukan pertanyaan baru, program tidak selalu memulai dari indeks ke-0, melainkan melanjutkan dari indeks terakhir. Apabila API mengembalikan status `429` (kuota habis) atau `403` (kunci dinonaktifkan), blok `except` tidak langsung menggagalkan proses, melainkan secara mulus (*silent failover*) melompat (`continue`) ke kunci berikutnya dalam matriks *array*. Logika ini menjamin Ketersediaan Tinggi (*High Availability*) bagi fungsionalitas asisten AI (Garg & Buyya, 2013).

10.1.2. Injeksi *System Instruction* dan Persona Asisten Tarjih

Salah satu kelemahan fatal dari *Large Language Model* generik adalah fenomena *hallucination* (halusinasi) dan ketidakkonsistenan teologis saat menjawab pertanyaan fikih yang spesifik. Untuk memastikan bahwa WinAI bertindak sesuai dengan ontologi dan epistemologi Falak Muhammadiyah serta parameter KHGT, aplikasi ini menggunakan teknik *System Prompting* tingkat lanjut.

System Instruction (Instruksi Sistem) adalah lapis terdalam dari konfigurasi LLM yang menetapkan *persona*, batasan operasi, dan gaya bahasa AI (White et al., 2023). Alih-alih menuliskan instruksi ini secara permanen (*hardcoded*) yang akan sulit diperbarui di masa mendatang, KHGT Times menarik *System Instruction* secara dinamis dari *server* repositori terpusat:

Python

```
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)

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."
```

Pendekatan *Over-The-Air* (OTA) *Prompting* ini memungkinkan *developer* untuk memperbaiki basis pengetahuan AI, menambah kaidah tarjih terbaru, atau mengubah gaya komunikasi asisten kapan saja secara *server-side* tanpa mengharuskan pengguna mengunduh ulang (*update patch*) aplikasi KHGT Times (Wang et al., 2021).

Setelah *System Instruction* didapatkan, ia digabungkan dengan aturan UI lokal (*GUI constraint*) menjadi sebuah muatan data (*payload*):

Python

```
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}"
```

Instruksi negatif (`DILARANG KERAS menyalin tabel`) bertindak sebagai penjaga batas antarmuka. Karena program telah dirancang sedemikian rupa untuk menampilkan tabel hisab yang dirender langsung oleh Python (demi presisi matematis dan tata letak *monospace*), AI diinstruksikan murni untuk bertindak sebagai interpreter atau komentator atas tabel tersebut, bukan sebagai generator tabel. Interaksi sinkronis antara mesin pemroses data (Python) dan mesin interpretasi kognitif (Gemini LLM) ini melahirkan sistem pakar komposit yang memiliki otoritas saintifik dan keluwesan naratif.

10.3. Ekstraksi Data Dinamis: Metodologi Web Scraping dan Arsitektur *Smart Interceptor*

Meskipun fondasi komputasi KHGT Times V7.4 bertumpu pada data ephemeris matematis (JPL DE441) yang bersifat statis dan deterministik, operasionalisasi kalender global di dunia nyata membutuhkan asimilasi data empiris yang bersifat dinamis. Data ini mencakup laporan rukyatul hilal global (*global sighting reports*) dari lembaga seperti *International Crescent Observation Project* (ICOP), anomali cuaca (*optical depth* atmosfer), hingga pembaruan nilai Delta-T (ΔT) secara *real-time* dari IERS. Untuk mengotomatisasi asimilasi data eksternal ini, arsitektur sistem mengimplementasikan modul *Web Scraping* tingkat lanjut yang digabungkan dengan arsitektur *Smart Interceptor*.

10.3.1. Keterbatasan API Konvensional dan Rasionalisasi Web Scraping

Dalam rekayasa perangkat lunak modern, pertukaran data idealnya dilakukan melalui *Application Programming Interface* (API) publik berupa RESTful atau GraphQL. Namun, dalam domain astronomi observasional, banyak lembaga falak internasional dan stasiun meteorologi lokal tidak menyediakan infrastruktur API publik (Glez-Peña et al., 2014). Data observasi berharga sering kali hanya disajikan dalam bentuk tabel HTML statis di dalam portal web yang dirender secara sisi server (*Server-Side Rendering*).

Kondisi ini memaksa KHGT Times V7.4 untuk menggunakan metodologi *web scraping*. Sistem dirancang untuk melakukan ekstraksi leksikal (*lexical extraction*) dan penguraian pohon *Document Object Model* (DOM) untuk mengambil titik data spesifik (seperti koordinat pengamat, jam observasi, dan status terlihat/tidaknya hilal) langsung dari elemen HTML, kemudian mengubahnya menjadi format JSON yang fungsional bagi mesin komputasi internal.

10.3.2. Penanganan Konten Dinamis dengan Asynchronous Rendering

Salah satu tantangan terbesar dalam *web scraping* modern adalah transisi web menuju *Client-Side Rendering* (CSR), di mana data tidak dimuat dalam HTML awal, melainkan dirender secara dinamis oleh JavaScript di peramban (peramban web). Pendekatan *scraping* tradisional menggunakan pustaka HTTP sederhana (seperti *Requests* pada Python) akan gagal mengekstrak data dari arsitektur semacam ini.

Untuk mengatasi ini, KHGT Times V7.4 memanfaatkan kerangka kerja *browser automation* secara *headless* (tanpa antarmuka grafis). Sistem memicu sebuah instansi peramban virtual yang mengeksekusi JavaScript secara penuh. Proses ini dilakukan secara asinkron (*asynchronous*) agar proses *scraping* data ICOP di *background* tidak memblokir antarmuka pengguna (GUI) utama aplikasi (Mitchell, 2018). Data yang telah terender sepenuhnya kemudian diekstraksi melalui selektor CSS (*CSS Selectors*) atau jalur XPath yang telah didefinisikan secara algoritmik.

10.3.3. Arsitektur *Smart Interceptor* untuk Menembus Limitasi Jaringan

Lembaga penyedia data sering kali menerapkan *Web Application Firewall* (WAF) atau sistem anti-bot (seperti Cloudflare atau reCAPTCHA) untuk mencegah serangan *Denial of Service* (DDoS) atau penyedotan data berlebihan. Jika modul KHGT melakukan *request* secara beruntun, IP pengguna akan diblokir (*rate-limited*).

Di sinilah peran **Smart Interceptor** menjadi krusial. Modul ini beroperasi pada lapisan *transport* jaringan (Layer 4 hingga Layer 7 pada model OSI) dan bertindak sebagai penengah (*middleware*) cerdas dengan fungsi-fungsi berikut:

1. **Rotasi Header dan User-Agent:** *Interceptor* secara dinamis memodifikasi *header* permintaan HTTP, merotasi *User-Agent* agar sistem tampak seperti peramban organik dari berbagai perangkat (komputer desktop, tablet, ponsel pintar) dan menghindari *fingerprinting* statis (Sirisuriya, 2015).

2. **Algoritma Exponential Backoff:** Jika server tujuan memberikan respons "429 Too Many Requests", *Interceptor* tidak langsung memutus koneksi, melainkan menerapkan algoritma jeda eksponensial. Sistem akan menunggu selama 2^n detik sebelum mencoba ulang, memitigasi risiko pemblokiran permanen (Vandenbroucke et al., 2021).
3. **Spoofing Jaringan Simulasif:** Dalam kasus yang ekstrem, *Interceptor* dapat disandikan untuk mensimulasikan jeda baca (*reading delay*) dan interaksi kursor acak sebelum memicu permintaan data, sehingga lulus dari analisis heuristik WAF yang mendeteksi "perilaku terlalu cepat" dari sebuah skrip mesin.

10.3.4. Etika Komputasi (Computational Ethics) dan Kepatuhan

Sebagai perangkat lunak yang dirancang berdasarkan prinsip syariat (termasuk prinsip tidak merugikan orang lain/ *la dharara wa la dhirar*), implementasi *web scraping* dalam KHGT Times terikat pada etika komputasi ketat (Krotov & Silva, 2018).

Smart Interceptor memiliki lapisan validasi pra-eksekusi yang secara otomatis membaca dan mematuhi aturan pada berkas `robots.txt` dari domain target. Selain itu, kecepatan ekstraksi (Crawl Rate) dibatasi secara mutlak maksimum 1 *request* per 5 detik untuk memastikan peladen (*server*) lembaga keilmuan nirlaba tidak mengalami kelebihan beban komputasi akibat operasional aplikasi KHGT Times. Dengan perpaduan kecerdasan ekstraksi dan kepatuhan etis ini, KHGT Times menjamin pasokan data astronomis berkelanjutan tanpa melanggar yurisdiksi keamanan siber.

DAFTAR PUSTAKA BAB 10

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2014). Web scraping technologies in an API world. *Briefings in Bioinformatics*, 15(5), 788-797. https://doi.org/10.1093/bib/bbt026

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys (CSUR)*, 55(12), 1-38. https://doi.org/10.1145/3571730

Johnson, J., Douze, M., & Jégou, H. (2021). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535-547.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., & Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Krotov, V., & Silva, L. (2018). Legality and ethics of web scraping. *Proceedings of the Twenty-fourth Americas Conference on Information Systems (AMCIS)*, New Orleans, LA.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Yih, W. T. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157-173.

Mitchell, R. (2018). *Web Scraping with Python: Collecting More Data from the Modern Web* (2nd ed.). O'Reilly Media.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sirisuriya, D. S. (2015). A comparative study on web scraping. *Proceedings of the 8th International Research Conference, KDU, Sri Lanka*, 135-140.

Vandenbroucke, K., Campmans, M., & Van den Herrewegen, J. (2021). API rate limiting and intelligent backoff strategies in data extraction. *Journal of Network and Computer Applications*, 182, 103050.

BAB 11: MANAJEMEN DATABASE DAN KOMPARASI 50 TAHUN

Dalam arsitektur perangkat lunak komputasi astronomi, perhitungan posisi benda langit secara *real-time* (*on-the-fly computation*) memakan daya komputasi yang sangat besar. Jika setiap kali pengguna membuka antarmuka kalender aplikasi harus menghitung ulang integrasi numerik orbit bulan sejak konjungsi hingga matahari terbenam di seluruh dunia, aplikasi akan mengalami latensi yang tidak dapat ditoleransi (Richards, 1998).

Untuk mengatasi leher botol (*bottleneck*) performa ini, **KHGT Times V7.4** memperkenalkan paradigma hibrida: menggabungkan komputasi ephemeris *real-time* untuk observasi harian dengan sistem prapemrosesan (*pre-computing*) menjadi basis data statis untuk rendering kalender jangka panjang.

11.1. Engine Auto-Builder `db_hijriah.json`

Fitur *HIJRI_DB Auto-Builder* (Modul 38) merupakan instrumen khusus administrator (developer) yang difungsikan sebagai generator siklus kalender. Alih-alih menulis tabel kalender secara manual, modul ini memerintahkan mesin ephemeris NASA untuk melakukan simulasi waktu berjalan (*time-lapse simulation*) selama puluhan hingga ratusan tahun secara otonom, mengekstrak data awal bulan, dan menyimpannya ke dalam struktur data *JavaScript Object Notation* (JSON) yang sangat ringan dan cepat diakses oleh memori antarmuka (GUI).

11.1.1. Ekstraksi Siklus Fase Bulan selama Ratusan Tahun

Pembuatan kalender berjangka panjang dihadapkan pada masalah deret waktu (*time series*). Perangkat lunak harus melompati ribuan fase lunasi (siklus sinodik bulan) tanpa kehilangan presisi fraksi detik pada setiap waktu ijtimak (konjungsi). KHGT Times V7.4 mengeksekusi proses ini melalui *thread* terpisah (`_build_hijri_db_thread`) agar antarmuka tidak membeku (*freeze*).

Algoritma ini menggunakan rumus konversi empiris untuk memperkirakan tahun Masehi berdasarkan tahun Hijriah yang sedang diiterasi, guna memuat bagian file `.bsp` ephemeris yang tepat:

Python

```
# Estimasi tahun Masehi dari tahun Hijriah
approx_greg_curr = int(y_h * 0.970224 + 622.54)
self.auto_switch_ephemeris(approx_greg_curr)
```

Konstanta 0.970224 merepresentasikan rasio panjang tahun kamariah rata-rata (354,36 hari) terhadap tahun tropis matahari (365,24 hari), sedangkan 622.54 adalah konstanta kalibrasi epoch Hijriah ke Masehi.

Setelah tahun yang tepat dikunci, sistem tidak mencari fase bulan dari tanggal 1 Januari hingga 31 Desember secara linier, yang mana sangat tidak efisien. Sebaliknya, sistem menggunakan fungsi prediksi waktu ijtimak (`get_approx_nm_tt`) dan membatasi rentang pencarian *Skyfield* secara eksklusif pada jendela pm 5 hari dari estimasi tersebut:

Python

```
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)
```

Pendekatan *narrow-band search* ini memangkas beban komputasi evaluasi polinomial Chebyshev secara drastis, memungkinkan pembangunan database kalender 100 tahun diselesaikan hanya dalam hitungan detik hingga menit, sebuah capaian optimasi algoritma pencarian akar (*root-finding optimization*) tingkat tinggi (Doggett, 2012).

11.1.2. Logika Penentuan Jumlah Hari Berdasarkan Waktu Ijtimak

Sistem kalender Hijriah aritmatik (seperti kalender *Urfi* atau Fatimiyah) menentukan jumlah hari 29 atau 30 secara selang-seling menggunakan rumus pembagian modulo (Ilyas, 1994). Berbeda secara fundamental dengan metode tersebut, Kalender Hijriah Global Tunggal (KHGT) adalah kalender astronomis hakiki yang penentuan jumlah harinya murni didikte oleh visibilitas hilal di permukaan bumi sesudah terjadinya ijtimak.

Untuk mengotomatisasi penentuan 29 atau 30 hari dalam database JSON, KHGT Times tidak melakukan evaluasi visibilitas harian secara iteratif karena akan memakan waktu berhari-hari. Program ini secara matematis mengukur jarak absolut antar-awal bulan. Algoritmanya bekerja dengan langkah berikut:

1. Sistem menghitung parameter KHGT (PKG 1 dan PKG 2) untuk bulan saat ini (m_h) dan mendapatkan waktu absolut jatuhnya tanggal 1 dalam format *Terrestrial Time* (TT).
2. Sistem melakukan hal yang sama untuk bulan berikutnya (m_h + 1).
3. Jumlah hari dalam bulan saat ini murni merupakan hasil pengurangan (*subtraction*) antara titik waktu absolut awal bulan depan dikurangi titik waktu absolut awal bulan saat ini.

Python

```
start_tt_curr = self.calculate_khgt_1st_of_month(nm_curr)
start_tt_next = self.calculate_khgt_1st_of_month(nm_next)

# Hitung Jumlah Hari (Selisih waktu mulai bulan depan dan bulan ini)
jumlah_hari = int(round(start_tt_next - start_tt_curr))
```

Karena nilai `start_tt` yang dihasilkan oleh fungsi `calculate_khgt_1st_of_month` selalu berupa bilangan bulat yang merepresentasikan pukul 00:00 UTC, selisih matematis dari kedua titik ini akan selalu menghasilkan nilai diskrit 29 atau 30. Nilai inilah yang kemudian diinjeksi (append) ke dalam matriks *dictionary* `year_data` dan disimpan permanen sebagai file `db_hijriah.json`.

Inovasi arsitektur pra-komputasi (pre-computation architecture) ini menggeser beban berat astrometri ke sisi *backend administrator*, sehingga pengguna akhir (*end-user*) dapat menikmati modul Kalender Hijriah Berjalan (Modul 26) dan Kalender Masehi Berjalan (Modul 27) dengan interaktivitas yang mulus (seamless) dan instan.

11.2. Algoritma Komparasi Panjang (Big Data Analysis)

Setelah database kalender statis berhasil dibangun, tantangan selanjutnya dalam KHGT Times V7.4 adalah bagaimana melakukan komparasi analitik antara hasil Kalender Hijriah Global Tunggal (KHGT) dengan sistem regional yang berlaku di Indonesia (Neo MABIMS). Komparasi ini sangat vital untuk mendeteksi kapan akan terjadi "perbedaan" perayaan hari besar (Idul Fitri, Idul Adha, dan awal Ramadhan) di masa depan (Syamsuddin, 2021).

Menganalisis data selama 50 tahun ke depan (misalnya dari 1447 H hingga 1496 H) memerlukan eksekusi algoritma *Big Data* yang mengevaluasi ribuan parameter fisis dalam satu waktu.

11.2.1. Eksekusi *Threading* untuk Mencegah *GUI Freeze*

Operasi iteratif yang panjang (seperti loop dari tahun 1447 hingga 1496) memakan waktu eksekusi (*execution time*) yang signifikan karena setiap siklus tahun melibatkan perhitungan *sunset*, *moonset*, ketinggian, dan elongasi. Jika kalkulasi ini dijalankan di *Main Thread* (utas utama) yang sama dengan antarmuka grafis (GUI) Tkinter, aplikasi akan berhenti merespons input (*freeze* atau *Not Responding*) sampai seluruh proses loop selesai (Bassi, 2007).

KHGT Times V7.4 memecahkan masalah ini dengan mengimplementasikan modul `threading` bawaan Python. Ketika tombol pemrosesan diklik, GUI hanya bertugas memicu *thread* latar belakang (*background thread*):

Python

```
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()
```

Kunci keberhasilan metode ini adalah penggunaan parameter `daemon=True`, yang memastikan *thread* kalkulasi akan otomatis dihentikan oleh sistem operasi jika jendela utama aplikasi ditutup oleh pengguna. Selanjutnya, untuk memperbarui tampilan teks di layar (yang mana Tkinter tidak mengizinkan pembaruan GUI dari *thread* selain *Main Thread*), KHGT Times menggunakan metode penjadwalan `.after(0, fungsi)`:

Python

```
self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun
Ramadhan..."))
```

Pendekatan *asynchronous UI update* ini memastikan pengalaman pengguna (*User Experience*) tetap interaktif dan *smooth* meskipun mesin komputasi sedang bekerja maksimal memproses data puluhan tahun.

11.2.2. Komparasi Historis dan Masa Depan: KHGT vs MABIMS (Ramadhan, Syawal, Zulhijah)

Inti dari algoritma komparasi (Modul 20, 21, dan 35) adalah menyandingkan dua epistemologi hisab: **KHGT (Global)** yang bersifat geosentris, melawan **Neo MABIMS (Regional)** yang bersifat toposentris di ujung barat Indonesia (Kota Sabang, Aceh).

Untuk setiap tahun y dalam rentang 50 tahun, sistem melakukan langkah-langkah analitik berikut:

1. Membaca tanggal jatuh 1 Ramadhan/Syawal/Zulhijah dari `HIJRI_DB` (hasil prakomputasi KHGT).
2. Menghitung mundur satu hari untuk menetapkan "Hari Rukyat" (misal: 29 Syakban).
3. Mengevaluasi posisi hilal secara toposentrik di Sabang tepat pada waktu *sunset* di Hari Rukyat tersebut.

Python

```
# Evaluasi Neo MABIMS di Sabang (Alt >= 3° dan Elong >= 6.4°)
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 MABIMS, bulan sebelumnya istikmal (digenapkan 30 hari)
    dt_mabims = dt_khgt + datetime.timedelta(days=1)
    status = "Beda (MABIMS Mundur 1 Hari)"
```

Jika kriteria MABIMS terpenuhi di Sabang, maka jatuhnya tanggal 1 versi regional akan **Serentak** dengan KHGT global. Namun, jika MABIMS Sabang gagal terpenuhi (biasanya karena KHGT tervalidasi di Benua Amerika sementara elongasi di Indonesia belum mencapai 6.4 derajat), maka regional Indonesia akan memaksakan *istikmal* (penggenapan bulan menjadi 30 hari), sehingga jatuhnya tanggal 1 akan **Mundur 1 Hari** dibandingkan KHGT.

Laporan komparasi matriks 50 tahun ini menjadi instrumen riset yang tak ternilai bagi Muhammadiyah dan kementerian terkait untuk melakukan mitigasi sosial terhadap potensi perbedaan hari raya di masa depan, meredam polemik jauh sebelum ia terjadi (Odeh, 2016).

11.3. Ekspor Data Mentah (CSV dan PDF)

Perangkat lunak astronomi yang baik tidak hanya menampilkan hasil kalkulasinya di layar, tetapi juga memfasilitasi pengguna untuk menyimpan, mengekstrak, dan menganalisis data tersebut lebih lanjut pada perangkat lunak pengolah data eksternal (seperti Microsoft Excel, SPSS, atau MATLAB). Dalam **KHGT Times V7.4**, modul ekspor dirancang untuk fleksibilitas akademik

tingkat tinggi, mendukung format *Comma Separated Values* (CSV) untuk data tabular mentah, dan *Portable Document Format* (PDF) untuk pelaporan formal.

11.3.1. Parsing String Array menjadi Data Terstruktur (CSV)

Fitur ekspor CSV diaktifkan ketika pengguna berada pada modul yang menghasilkan data *Big Data* berbentuk tabel (seperti Modul 14, 22, 23, 24, 25, 36, dan 37). Secara khusus, pada modul "Tabel Ketinggian/Elongasi Hilal 50 Tahun", program menghasilkan data mentah yang sangat masif.

Tantangan teknisnya adalah bahwa pada modul-modul pelaporan teks, data awal berbentuk satu *string* panjang (multibaris) yang ditampilkan pada layar *textbox*. Untuk mengubahnya menjadi CSV yang valid, sistem harus melakukan proses *parsing* (penguraian teks) secara mandiri.

Python

```
# Ekstrak data dari Textbox
report_data = self.textbox.get("1.0", "end-1c")

# Split berdasarkan baris
lines = report_data.split('\n')
for line in lines:
    # Memfilter baris yang merupakan bagian dari tabel (mengandung karakter
'|')
    # dan mengabaikan baris pembatas dekoratif ('=' atau '-')
    if '|' in line and not line.startswith('=') and not line.startswith('-'):
        # Memecah baris menjadi kolom-kolom
        row = [col.strip() for col in line.split('|')]
        writer.writerow(row)
```

Sistem membaca karakter palang vertikal (`|`) sebagai *delimiter* virtual dalam tampilan teks GUI, lalu memecahnya menjadi komponen larik (*array list*) Python. Untuk memastikan kompatibilitas internasional (terutama ketika dibuka di Microsoft Excel regional Eropa/Indonesia yang menggunakan titik koma sebagai pemisah kolom), fungsi `csv.writer` diinstruksikan menggunakan parameter `delimiter=';'`.

Sistem juga mengimplementasikan teknik `encoding='utf-8-sig'` saat menulis file. Parameter *sig* (BOM - *Byte Order Mark*) ini sangat krusial; ia memaksa Excel untuk merender karakter non-ASCII (seperti simbol derajat `^\circ`) secara sempurna, sehingga data ketinggian hilal (contoh: "5.45°") tidak berubah menjadi karakter acak (*gibberish*) saat dianalisis oleh pengguna (VanderPlas et al., 2020).

11.3.2. Pelukisan Ulang *Canvas* untuk Laporan Akademik (PDF)

Sementara CSV ditujukan untuk analisis kuantitatif lanjutan, PDF ditujukan untuk publikasi, pencetakan (*printing*), dan pelaporan yudisial. Terdapat dua metodologi ekspor PDF dalam KHGT Times V7.4, bergantung pada modul mana yang sedang aktif.

1. Ekspor Laporan Teks Menggunakan Pustaka FPDF: Jika pengguna sedang membuka modul hisab waktu salat atau visibilitas harian (yang berformat teks standar), program mengonversi log teks menjadi dokumen PDF yang terstruktur dengan *header* dan waktu cetak:

Python

```
pdf = FPDF()
pdf.add_page()
pdf.set_font("Courier", size=9) # Menggunakan font Monospace agar tabel rata
# ...
clean_line = line.replace('°', ' deg').replace('➔', '->').replace('▶', '>').replace('☄', '[Bulan]')
safe_line = clean_line.encode('latin-1', 'replace').decode('latin-1')
pdf.cell(0, 4, txt=safe_line, ln=True)
```

Proses *sanitization* (seperti `replace('°', ' deg')`) dilakukan sebagai bentuk penanganan galat (*error handling*) mengingat *library* FPDF versi klasik memiliki keterbatasan dalam memproses set karakter Unicode secara *native*.

2. Ekspor Peta Visual dan Kalender Murni (Konversi Langsung `Image`): Inovasi visual terbaik terjadi jika pengguna menekan tombol PDF pada Modul 26 (Kalender Hijriah) atau Modul 27 (Kalender Masehi). Karena GUI kalender dibentuk menggunakan *widget* Tkinter, tidak ada "teks" yang bisa diubah menjadi PDF secara langsung. Mengekspor *screenshot* antarmuka GUI akan menghasilkan resolusi yang rendah (buram saat dicetak).

Untuk mensiasatinya, KHGT Times memiliki mesin pelukis (*rendering engine*) internal tak terlihat (*headless*) yang dijalankan menggunakan pustaka manipulasi citra **Pillow (PIL)**:

Python

```
# Memanggil fungsi render gambar HD internal (bukan screenshot GUI)
img = self._generate_calendar_image()

# Menyimpan secara langsung ke format PDF dengan resolusi cetak
img.save(filepath, "PDF", resolution=150.0)
```

Fungsi `_generate_calendar_image()` secara otonom menggambar ulang seluruh kotak kalender, menarik fon *TrueType* resolusi tinggi (`arialbd.ttf`), memuat fase bulan, dan menyematkan warna indikator peringatan hari besar ke dalam satu kanvas raksasa beresolusi 1400x1000 piksel. Konversi langsung matriks piksel ini ke dalam PDF dengan opsi `resolution=150.0` menjamin hasil *export* yang sangat tajam (*crisp*), setara dengan publikasi desain grafis profesional, dan siap dikirimkan sebagai poster atau edaran resmi organisasi.

Rangkaian sistem manajemen basis data dan mekanisme ekspor data ganda inilah yang menyempurnakan KHGT Times V7.4, mentransformasikannya dari sebuah skrip saintifik laboratorium menjadi instrumen perangkat lunak operasional (*production-ready*) untuk umat Islam di seluruh dunia.

DAFTAR PUSTAKA BAB 11

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.

Doggett, L. E. (2012). Calendars. In *Explanatory Supplement to the Astronomical Almanac* (pp. 575-608). University Science Books.

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Richards, E. G. (1998). *Mapping Time: The Calendar and Its History*. Oxford University Press.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

VanderPlas, J., et al. (2020). Python for data analysis. *Journal of Data Science*, 14(2), 34-45.

BAB 12: PENUTUP DAN PROYEKSI MASA DEPAN FALAK DIGITAL

12.1. Sintesis Arsitektur Komputasi KHGT Times V7.4

Evolusi perangkat lunak astronomi Islam telah mencapai titik kulminasinya melalui transisi dari kalkulator ephemeris sederhana menuju mesin analitik komprehensif berskala global. **KHGT Times V7.4** merepresentasikan sintesis yang koheren antara sains astrometri presisi tinggi (*high-precision astrometry*), rekayasa perangkat lunak modern (*software engineering*), dan epistemologi hukum Islam (Fikih).

Dari bedah kode (*source code analysis*) yang telah kita lakukan pada bab-bab sebelumnya, dapat disimpulkan bahwa keunggulan arsitektural KHGT Times V7.4 ditopang oleh tiga pilar komputasi utama:

1. Migrasi Epistemologis dari Deret Analitik ke Integrasi Numerik

Perangkat lunak falak klasik umumnya mengalami degradasi akurasi (*accuracy drift*) ketika digunakan untuk memprediksi posisi benda langit puluhan tahun ke depan atau ke masa lampau. Hal ini disebabkan oleh ketergantungan pada algoritma deret analitik seperti VSOP87, yang memotong (*truncate*) deret harmonik untuk menghemat memori (Urban & Seidelmann, 2013). KHGT Times V7.4 secara radikal meninggalkan batasan ini dengan mengadopsi model *Development Ephemeris* (DE441) dari JPL NASA. Melalui pustaka *Skyfield*, aplikasi mengekstraksi koordinat vektor bumi, bulan, dan matahari langsung dari koefisien polinomial Chebyshev. Metode ini menjamin bahwa fluktuasi gravitasi makro maupun perturbasi mikro di dalam tata surya terhitung secara mutlak (Park et al., 2021).

2. Hibridisasi Algoritma untuk Pemecahan Kompleksitas mathcal{O}(n)

Kalender Hijriah Global Tunggal (KHGT) menuntut validasi visibilitas hilal di seluruh permukaan bumi untuk memenuhi syarat Parameter Kalender Global (PKG 1 dan PKG 2) (Syamsuddin, 2021). Evaluasi ini—yang melibatkan perhitungan *sunset*, *moonset*, elongasi, dan *altitude* geosentrik di ratusan kota pada satu waktu—merupakan operasi yang sangat menguras daya komputasi (*computationally expensive*).

Jika hanya menggunakan *Skyfield*, proses penemuan titik persilangan (*root-finding* untuk *discrete events*) akan menyebabkan antarmuka aplikasi membeku (*GUI freeze*). Sintesis solusi yang diterapkan oleh arsitek KHGT Times V7.4 adalah **komputasi hibrida**:

- Sistem menggunakan pustaka berbasis C murni (`PyEphem`) sebagai mesin *brute-force* yang sangat cepat untuk melakukan pemindaian awal (*coarse scanning*) terhadap waktu *sunset* di ratusan kota.
- Setelah titik waktu potensial ditemukan, waktu tersebut ditransfer kembali ke mesin `Skyfield` untuk diekstraksi nilai parameter astrometrinya secara presisi (termasuk konversi waktu ke *Terrestrial Time* / TT untuk membatalkan bias *Delta-T* rotasi bumi).

Sinergi ini menghasilkan *engine* pelacak (Global Scanner) yang *robust* dan asinkron, memungkinkan perhitungan 50 tahun komparasi KHGT diselesaikan dalam hitungan menit (Bassi, 2007).

3. Integrasi Interpolasi Spasial dan Visualisasi Cerdas

Data numerik yang presisi sering kali gagal dipahami oleh pemangku kebijakan jika tidak disajikan dalam representasi yang intuitif. Melalui penggunaan *SciPy* (Bicubic Griddata Interpolation) dan *Matplotlib* (BoundaryNorm Mapping), KHGT Times tidak hanya mencetak angka, tetapi merender peta kontur visibilitas definisi tinggi (HD Map). Sistem secara dinamis menambal lubang komputasi (*NaN patching*) di area Kutub dan Garis Batas Penanggalan menggunakan metode *Nearest Neighbor*, sehingga menghasilkan kurva batas 5^\circ ketinggian dan 8^\circ elongasi yang membelah topografi bumi secara presisi dan *seamless* (tanpa terputus) (Virtanen et al., 2020).

Selain itu, keberhasilan menginjeksi Kecerdasan Buatan (WinAI berbasis Gemini LLM) sebagai *smart interceptor* dan asisten fikih yang mampu membaca database lokal (RAG) mengangkat KHGT Times dari sekadar program astronomi menjadi **Sistem Pakar Komposit** (*Composite Expert System*).

Secara arsitektural, kode sumber KHGT Times V7.4 membuktikan bahwa hambatan teknis komputasional yang selama ini dianggap sebagai argumen untuk menolak penerapan Kalender Hijriah Global telah berhasil dipecahkan secara tuntas dan elegan.

12.2. Implikasi Terhadap Unifikasi Kalender Islam Global

Diskursus penyatuan kalender Hijriah sering kali terjebak dalam perdebatan teoretis dan fikih tanpa adanya pijakan teknologi yang memadai untuk mengeksekusi konsep tersebut secara *real-time* dan presisi. Kehadiran **KHGT Times V7.4** bukan sekadar pencapaian dalam bidang rekayasa perangkat lunak (*software engineering*), melainkan sebuah katalisator teknologis yang menjembatani jurang antara idealisme Kalender Hijriah Global Tunggal (KHGT) dan implementasinya di dunia nyata.

Perangkat lunak ini membawa implikasi sosiologis, teologis, dan kebijakan (*policy-making*) yang sangat signifikan bagi umat Islam global, khususnya dalam kerangka transisi menuju kalender yang unifikatif.

12.2.1. Standardisasi Otoritatif dan Aksesibilitas Data

Salah satu kendala terbesar pasca-Kongres Istanbul 2016 adalah minimnya ketersediaan perangkat lunak yang secara *native* dan transparan menghitung kriteria KHGT (Tinggi Geosentris ge 5^\circ, Elongasi Geosentris ge 8^\circ, dan parameter fajar Gisborne). Publik dan pembuat kebijakan sering kali disajikan data matang (*black-box data*) tanpa bisa memvalidasi secara mandiri bagaimana titik pertama di bumi (*first mainland*) tersebut ditentukan (Odeh, 2016).

KHGT Times mendemokratisasi akses terhadap komputasi astronomi tingkat tinggi. Dengan mengintegrasikan ephemeris NASA (DE441) ke dalam antarmuka berbasis *Desktop* (GUI),

aplikasi ini memungkinkan ormas Islam, kementerian agama, hingga observatorium universitas untuk mereplikasi perhitungan kalender global secara mandiri. Lebih jauh, keberadaan fitur **Crescent Visibility HD Map** (Modul 2) dan pelacakan kota pertama di dunia secara spasial memberikan pembuktian visual yang sangat meyakinkan.

Pembuat kebijakan syariah tidak lagi harus memahami kerumitan kalkulus tensor atau trigonometri bola untuk mengambil keputusan. Kemampuan sistem merender garis batas kriteria KHGT secara presisi di atas peta topografi dunia memberikan landasan empiris yang kuat (*Rukyat bil 'Ilmi*) untuk menjustifikasi awal bulan kepada masyarakat awam (Guessoum, 2008). Arsitektur transparan ini mempercepat penerimaan sosiologis terhadap KHGT, mengubahnya dari sekadar wacana muktamar menjadi kalender yang beroperasi secara fungsional.

12.2.2. Mitigasi Konflik Sosiologis Melalui Prediksi Jangka Panjang

Pergeseran dari kriteria *Wujudul Hilal* atau *Imkanur Rukyat Regional* (seperti MABIMS) menuju KHGT tidak pelak akan menimbulkan potensi friksi sosiologis, terutama pada tahun-tahun di mana KHGT dan kriteria lokal menghasilkan tanggal hari raya yang berbeda. Secara historis, perbedaan penetapan 1 Ramadhan, 1 Syawal, dan 1 Zulhijah sering memicu disonansi di akar rumput karena pengumumannya dilakukan secara reaktif pada sidang isbat di malam hari (Haseeb, 2020).

Modul komparasi 50 tahun (Modul 20, 21, dan 35) pada KHGT Times mengubah paradigma falak reaktif menjadi **falak proaktif**. Dengan menggunakan mesin iterasi ephemeris, sistem ini memetakan dengan tepat pada tahun berapa KHGT (global) akan berbeda dengan Neo MABIMS (lokal Sabang, Indonesia) dalam lima dekade ke depan.

Implikasi dari analisis *Big Data* ini sangat luar biasa bagi manajemen konflik keagamaan. Jika sistem memprediksi bahwa pada tahun 1452 H KHGT akan menetapkan Idul Fitri satu hari lebih awal dibandingkan MABIMS, maka pemangku kebijakan (seperti Majelis Tarjih Muhammadiyah atau ulama internasional) memiliki jeda waktu bertahun-tahun untuk melakukan edukasi publik, menyosialisasikan alasan saintifik perbedaan tersebut, dan mempersiapkan mitigasi sosialnya (Syamsuddin, 2021).

Melalui penyajian data komparatif yang disandingkan berdampingan—lengkap dengan analisis Ketinggian dan Elongasi di Sabang versus pembacaan Geosentris di titik benua Amerika—KHGT Times V7.4 memberikan "kepastian waktu" (*temporal certainty*) yang merupakan esensi utama dari fungsi sebuah kalender sipil dan ibadah (Ilyas, 1994). Hal ini menempatkan perangkat lunak ini sebagai tonggak penting (*milestone*) dalam sejarah panjang usaha penyatuan penanggalan umat Islam sedunia.

12.3. Arah Pengembangan Masa Depan (Future Work)

Perangkat lunak berbasis keilmuan (*scientific software*) tidak pernah mencapai titik henti absolut; ia harus terus berevolusi secara iteratif merespons perkembangan algoritma dan perangkat keras. Meskipun **KHGT Times V7.4** telah membuktikan superioritasnya dalam mengeksekusi parameter Kalender Hijriah Global Tunggal melalui integrasi ephemeris NASA dan *multi-threading*, terdapat

beberapa dimensi arsitektural yang dapat diekspansi untuk proyeksi pengembangan di masa depan (*future work*).

12.3.1. Integrasi Kecerdasan Buatan Terdistribusi dan Prediksi Cuaca Berbasis *Machine Learning*

Pada versi V7.4, modul visibilitas astrofotografi (Modul 31) dan evaluasi SQM (Modul 33) telah berhasil mengintegrasikan data cuaca *real-time* melalui API *Open-Meteo* untuk mendeteksi tutupan awan (*cloud cover*). Namun, pendekatan ini masih bersifat reaktif.

Pengembangan masa depan diarahkan pada implementasi *Machine Learning* (ML) prediktif, seperti penggunaan model jaringan saraf tiruan (*Artificial Neural Networks* / ANN) yang memproses data historis atmosfer lokal. Dengan model ini, sistem tidak hanya melaporkan tutupan awan saat ini, tetapi mampu memprediksi indeks refraksi atmosfer aktual dan tingkat atenuasi optis (pelemahan cahaya hilal) beberapa hari sebelum proses *rukyatul hilal* dilakukan. Selain itu, fitur asisten **WinAI** dapat ditingkatkan dari sekadar *Retrieval-Augmented Generation* (RAG) teks menjadi agen otonom (*Autonomous Agent*) yang mampu menulis ulang parameter kode secara dinamis berdasarkan instruksi suara (*Voice-to-Code Execution*) (Zheng et al., 2023).

12.3.2. Transformasi Arsitektur Menuju *Cloud-Native* dan Ekosistem *Mobile*

Secara topologi *software*, KHGT Times V7.4 dibangun sebagai aplikasi *Desktop Monolithic* (Python berbasis Tkinter). Kelemahannya adalah tingginya ketergantungan pada memori fisik komputer pengguna (RAM dan CPU) untuk merender peta HD dan memproses *array* linspace 50 tahun.

Untuk mendemokratisasi akses data falak ini secara masif, struktur kode *backend* (khususnya logika *Skyfield* dan *SciPy*) harus dipisahkan (*decoupled*) dari antarmuka penggunanya. Ekstensi di masa depan harus melibatkan migrasi mesin perhitungan ini ke dalam arsitektur *Microservices* berbasis *Cloud* (Garg & Buyya, 2013). Dengan membungkus mesin KHGT menjadi antarmuka pemrograman aplikasi (API) RESTful atau GraphQL terpusat, pengembang eksternal dapat dengan mudah menarik data visibilitas hilal untuk kemudian disajikan melalui aplikasi ponsel pintar (*Mobile Apps* iOS/Android) atau *dashboard* interaktif berbasis web, memungkinkan pembuat kebijakan di seluruh dunia mengakses data KHGT tanpa perlu melakukan kompilasi kode lokal.

12.3.3. Peningkatan Presisi Spasial (GIS) untuk Pemfilteran Daratan (*Mainland*)

Pada Modul 5 (Pelacakan Titik Pertama), validasi syarat *mainland* (daratan utama benua) dilakukan dengan menyeleksi sebuah kamus data (`CITY_DB`) dan menyaring zona kepulauan secara manual (*hardcoded exclusion list*). Meskipun cepat, metode ini memiliki limitasi geografis karena bergantung pada kelengkapan pangkalan data (*database*).

Riset ke depan harus memformulasikan algoritma yang terintegrasi dengan Sistem Informasi Geografis (GIS) tingkat lanjut. Alih-alih mengecek nama wilayah, sistem akan memuat *polygon bounding box* resolusi tinggi dari peta benua bumi. Ketika algoritma ephemeris menemukan titik koordinat visibilitas pertama, ia akan langsung melakukan operasi matematis *Point-in-Polygon*

(PiP) terhadap data GIS spasial untuk memastikan secara otonom dan matematis murni apakah titik tersebut berada di daratan utama atau di tengah samudra (Hoffman, 2003).

Penutup

Penyatuan kalender Islam bukanlah utopia, melainkan sebuah kewajiban peradaban yang kini mendapatkan medium eksekusinya melalui komputasi modern. Buku *"Konsep, Formulasi, dan Arsitektur Kode KHGT Times V7.4"* ini membuktikan bahwa batas antara dogma fikih dan sains eksakta telah melebur. Kesatuan Matlak (*Ittihad al-Matali'*) dan Transfer Visibilitas yang didengungkan oleh Kalender Hijriah Global Tunggal (KHGT) kini dapat divalidasi, disimulasikan, dan dipertanggungjawabkan secara saintifik dalam hitungan fraksi detik.

Tugas generasi falak di masa mendatang bukan lagi mencari kelemahan rumus, melainkan bagaimana mentransformasikan data empiris komputasional ini menjadi kesadaran sosiologis yang utuh di tengah masyarakat. *Wallahu a'lam bish-shawab*.

DAFTAR PUSTAKA BAB 12

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.

Garg, S. K., & Buyya, R. (2013). NetworkCloudSim: Modelling parallel applications in cloud simulations. *Fourth International Conference on Utility and Cloud Computing*, 105-113.

Guessoum, N. (2008). The astrophysics of the Islamic calendar. *Astronomy & Geophysics*, 49(5), 5.16-5.21.

Haseeb, A. (2020). *The Global Islamic Calendar: Prospects and Challenges*. Islamic Foundation.

Hoffman, R. E. (2003). Rationalizing the Islamic calendar. *Science and Islamic Observatory*, 1(1), 1-12.

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Odeh, M. S. (2016). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).

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.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

Zheng, L., Chiang, W. L., Ying, J., Shen, S., Huang, C., ... & Xing, E. P. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*.

DAFTAR PUSTAKA

- Abdali, S. K. (1997). *The correct qibla*. American Trust Publications.
- Anwar, S. (2008). *Hari Raya dan Problematika Penentuan Awal Bulan Hijriah*. Suara Muhammadiyah.
- Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.
- Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. VSOP 87 solutions. *Astronomy and Astrophysics*, 202, 309-315.
- Caldwell, J. A. R., & Laney, C. D. (2001). First visibility of the lunar crescent. *African Skies/Cieux Africains*, 5, 15-23.
- Doggett, L. E. (2012). Calendars. In *Explanatory Supplement to the Astronomical Almanac* (pp. 575-608). University Science Books.
- Downey, A. (2011). *Think Python: How to think like a computer scientist*. O'Reilly Media.
- Duffett-Smith, P., & Zwart, J. (2011). *Practical Astronomy with your Calculator or Spreadsheet* (4th ed.). Cambridge University Press.
- Espenak, F., & Meeus, J. (2006). *Five Millennium Canon of Solar Eclipses: -1999 to +3000*. NASA Goddard Space Flight Center.
- Fluke, C. J., Bourke, P. D., & O'Donovan, D. (2006). Future directions in astronomy visualization. *Publications of the Astronomical Society of Australia*, 23(1), 12-24.
- Franke, R. (1982). Scattered data interpolation: tests of some methods. *Mathematics of Computation*, 38(157), 181-200.
- Garg, S. K., & Buyya, R. (2013). NetworkCloudSim: Modelling parallel applications in cloud simulations. *Fourth International Conference on Utility and Cloud Computing*, 105-113.
- Gargano, M., et al. (2015). Planetariums and digital domes: A new era for astronomical visualization. *Journal of Science Communication*, 14(2).
- Guessoum, N. (2008). The astrophysics of the Islamic calendar. *Astronomy & Geophysics*, 49(5), 5.16-5.21.
- Haseeb, A. (2020). *The Global Islamic Calendar: Prospects and Challenges*. Islamic Foundation.
- Hoffman, R. E. (2003). Rationalizing the Islamic calendar. *Science and Islamic Observatory*, 1(1), 1-12.
- Hughes, D. W., Yallop, B. D., & Hohenkerk, C. Y. (1989). The equation of time. *Monthly Notices of the Royal Astronomical Society*, 238(4), 1529-1535.
- Hughes, J. F., et al. (2014). *Computer Graphics: Principles and Practice* (3rd ed.). Addison-Wesley Professional.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

- Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.
- Ilyas, M. (1997). *Islamic Calendar, Times & Qibla*. Berita Publishing.
- Kaplan, G. H. (2005). *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models* (USNO Circular No. 179). United States Naval Observatory.
- King, D. A. (1986). *Astronomy in the Service of Islam*. Variorum Reprints.
- Lancaster, P., & Salkauskas, K. (1986). *Curve and Surface Fitting: An Introduction*. Academic Press.
- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.
- Moosavi, S. Z., Bakhshi, H., & Zadehkhorram, A. R. (2015). A survey on calculation methods of prayer times at high latitudes. *International Journal of Mechatronics, Electrical and Computer Technology*, 5(15), 2095-2105.
- Morrison, L. V., & Stephenson, F. R. (2001). Historical values of the Earth's clock error Delta T and the calculation of eclipses. *Journal for the History of Astronomy*, 32(3), 227-236.
- Newman, M. E. J. (2012). *Computational Physics*. CreateSpace Independent Publishing Platform.
- Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1-3), 39-64.
- Odeh, M. S. (2016). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).
- 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.
- Patat, F. (2003). UBVRI night sky brightness during sunspot maximum at ESO-Paranal. *Astronomy & Astrophysics*, 400(3), 1183-1198.
- Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl-1907.
- Richards, E. G. (1998). *Mapping Time: The Calendar and Its History*. Oxford University Press.
- Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311-361.
- Susiknan, A. (2007). *Ilmu Falak: Perjumpaan Khazanah Islam dan Sains Modern*. Suara Muhammadiyah.
- Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.
- Torge, W., & Müller, J. (2012). *Geodesy* (4th ed.). De Gruyter.
- Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.
- VanderPlas, J., et al. (2020). Python for data analysis. *Journal of Data Science*, 14(2), 34-45.

- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.
- Wang, J., et al. (2021). Over-the-air updates for machine learning models. *IEEE Internet of Things Journal*, 8(15).
- White, J., et al. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*.
- Yallop, B. D. (1998). *A Method for Predicting the First Sighting of the New Crescent Moon* (NAO Technical Note No. 69). HM Nautical Almanac Office.
- Zayed, A. (2015). *Fikih Astronomi: Menjawab Problematika Penentuan Awal Bulan*. Darul Ulum.
- Zheng, L., Chiang, W. L., Ying, J., Shen, S., Huang, C., ... & Xing, E. P. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*.

GLOSARIUM

A

- **Analemma:** Kurva berbentuk angka 8 yang menunjukkan posisi Matahari di langit jika diamati pada jam yang sama setiap hari sepanjang tahun.
- **API (Application Programming Interface):** Antarmuka yang memungkinkan dua aplikasi perangkat lunak untuk saling berkomunikasi, digunakan dalam sistem WinAI untuk mengakses LLM.
- **Asensio Rekta (Right Ascension):** Koordinat ekuatorial astronomis yang analog dengan bujur geografis di bumi, diukur ke arah timur dari titik Aries.
- **Azimuth:** Sudut putar arah horizontal suatu benda langit, diukur searah jarum jam dari titik Utara sejati (0°).

B

- **Bicubic Interpolation:** Algoritma matematika dalam *SciPy* untuk merekonstruksi dan memperhalus matriks data kasar menjadi kurva spasial yang mulus tanpa garis bergerigi (digunakan pada peta visibilitas).
- **Boundary Norm:** Teknik *rendering* visual pada *Matplotlib* yang mengubah gradasi warna menjadi blok warna diskrit dan tegas untuk area peta tertentu.

C

- **Celestial Sphere (Bola Langit):** Bola imajiner dengan jari-jari tak terhingga yang berpusat pada bumi, di mana benda-benda langit seolah-olah menempel padanya.
- **CSV (Comma Separated Values):** Format berkas teks sederhana untuk menyimpan data tabular, sering digunakan untuk ekspor data historis KHGT ke *spreadsheet*.

D

- **Deklinasi:** Koordinat ekuatorial yang analog dengan garis lintang di bumi, mengukur jarak sudut suatu benda langit di utara atau selatan ekuator langit.
- **Delaunay Triangulation:** Metode matematis untuk menghubungkan titik-titik data spasial menjadi jaringan segitiga, digunakan pada interpolasi peta *SciPy*.

E

- **Elongasi:** Jarak sudut antara pusat piringan Bulan dan pusat piringan Matahari; dalam KHGT disyaratkan minimal 8 derajat (secara geosentris).
- **Ephemeris:** Tabel atau basis data yang memuat posisi benda-benda astronomis di langit pada waktu tertentu secara berurutan.
- **Equation of Time (EoT):** Selisih waktu antara waktu matahari sejati (berdasarkan pengamatan) dengan waktu matahari rata-rata (berdasarkan jam mekanik).

F

- **Fajar Shadiq (Astronomical Twilight):** Waktu ketika cahaya matahari mulai tersebar di atmosfer saat pusat piringan matahari berada pada sudut depresi tertentu (umumnya -18° atau -20°) di bawah ufuk.

G

- **Geosentris:** Sistem koordinat dan titik pengamatan astronomi yang diasumsikan berpusat tepat di inti Bumi.
- **GUI (Graphical User Interface):** Antarmuka pengguna grafis, seperti tombol dan menu visual yang dibangun menggunakan pustaka *Tkinter* pada Python.

H

- **Hilal:** Bulan sabit paling tipis yang pertama kali dapat diamati sesaat setelah matahari terbenam pada hari terjadinya konjungsi (ijtimak).
- **Hisab:** Perhitungan matematis dan astronomis untuk menentukan posisi benda langit, waktu salat, dan awal bulan Hijriah.

I

- **Ijtimak (Konjungsi / New Moon):** Peristiwa ketika Bulan dan Matahari berada pada bujur ekliptika yang sama, menandakan akhir siklus lunasi lama dan awal siklus lunasi baru.
- **Istikmal:** Penggenapan jumlah hari dalam bulan kalender Hijriah menjadi 30 hari apabila hilal tidak terlihat atau kriteria visibilitas tidak terpenuhi.

J

- **JSON (JavaScript Object Notation):** Format teks ringan untuk pertukaran data, digunakan oleh KHGT Times untuk menyimpan database kalender prakomputasi.
- **JPL (Jet Propulsion Laboratory):** Pusat penelitian NASA yang menerbitkan data *Development Ephemeris* presisi tinggi (seperti DE441) yang digunakan dalam aplikasi ini.

K

- **KHGT (Kalender Hijriah Global Tunggal):** Sistem penanggalan Islam yang bertujuan menyatukan jatuhnya awal bulan di seluruh dunia menggunakan prinsip hisab dan transfer visibilitas global.

L

- **LLM (Large Language Model):** Model kecerdasan buatan berbasis jaringan saraf dalam yang memproses bahasa alami, digunakan dalam KHGT Times sebagai *Asisten Fikih Falak.*

- **Lunasi:** Satu siklus penuh fase bulan, dari satu ijtimak ke ijtimak berikutnya (rata-rata 29,53 hari).

M

- **MABIMS:** Kriteria visibilitas hilal regional yang disepakati oleh Menteri Agama Brunei Darussalam, Indonesia, Malaysia, dan Singapura.
- **Mainland (Daratan Utama):** Wilayah daratan benua yang memiliki populasi berkesinambungan, menjadi syarat lokasi terpenuhinya kriteria KHGT (mengabaikan kepulauan terisolasi).
- **Matlak:** Batas geografis visibilitas; dalam konsep KHGT, matlak bersifat global (satu bumi satu matlak / *Ittihad al-Matali'*).

N

- **Nisf al-Lail:** Pendekatan fikih untuk wilayah lintang tinggi yang membagi malam menjadi dua bagian sama panjang untuk menentukan masuknya waktu Isya dan Subuh.
- **NumPy:** Pustaka komputasi saintifik Python yang digunakan untuk memproses array numerik besar dan aljabar linear.

O

- **Oblate Spheroid:** Bentuk bumi yang sebenarnya, yaitu elipsoid yang agak pepat di bagian kutub dan menggembung di bagian ekuator akibat gaya sentrifugal rotasi bumi.
- **Oseania (Oceania):** Gugusan kepulauan di Samudra Pasifik (Timur Jauh) yang dikecualikan oleh KHGT Times dari pelacakan visibilitas karena bukan wilayah *mainland*.

P

- **Paralaks (Parallax):** Pergeseran posisi semu suatu benda langit jika dilihat dari dua titik pengamatan yang berbeda (misalnya dari inti bumi vs permukaan bumi).
- **PyEphem:** Pustaka astronomi klasik berbasis bahasa C yang digunakan KHGT Times sebagai mesin *brute-force* untuk memindai titik *sunset* dengan cepat.

Q

- **Qibla Time (Rashdul Kiblat):** Momen spesifik ketika lintasan azimuth matahari (atau bayangannya) sejajar dan berhimpitan secara presisi dengan arah azimuth Ka'bah.

R

- **Refraksi Atmosfer:** Peristiwa pembiasan cahaya benda langit saat melewati atmosfer bumi, menyebabkan matahari/bulan tampak lebih tinggi dari posisi geometris sebenarnya.
- **Round-Robin:** Algoritma penjadwalan rotasi yang digunakan dalam KHGT Times untuk merotasi kunci API (*API Key*) secara bergantian guna menghindari pemblokiran *rate-limit*.

S

- **Skyfield:** Pustaka Python modern yang menghitung posisi planet dan satelit menggunakan prinsip vektor mutakhir setara dengan keakuratan observatorium standar IAU.
- **Syzygy:** Konfigurasi garis lurus antara tiga benda langit (seperti Matahari, Bumi, dan Bulan) yang merupakan syarat mutlak terjadinya gerhana.

T

- **Toposentris:** Sistem koordinat yang titik referensinya murni bertumpu pada mata pengamat yang berdiri tegak di suatu titik kordinat tertentu di permukaan Bumi.
- **Terrestrial Time (TT):** Skala waktu dinamis tanpa gangguan yang digunakan dalam komputasi ephemeris, bebas dari anomali pelambatan rotasi bumi (Delta T).

U

- **Umbra:** Area bayangan inti bumi yang paling pekat; apabila bulan memasuki area ini secara geometris, maka terjadilah gerhana bulan (sebagian atau total).
- **UTC (Coordinated Universal Time):** Standar waktu sipil global yang menggantikan *Greenwich Mean Time* (GMT), menjadi dasar waktu dalam pemrograman komputasi antar-zona.

V

- **Visibilitas:** Parameter atau kriteria astrofisika yang memprediksi apakah hilal secara teoretis dapat dilihat (dirukyat) dengan mata telanjang atau instrumen optik dari permukaan bumi.

W

- **WGS84 (World Geodetic System 1984):** Standar datum referensi geodetik global yang memodelkan bumi sebagai elipsoid, digunakan KHGT Times untuk memetakan koordinat observasi.
- **WinAI:** Nama modul Kecerdasan Buatan (AI) berbasis asisten pintar yang disematkan langsung dalam *desktop client* KHGT Times V7.4.

X

- **XEphem:** Perangkat lunak kalender astronomi ephemeris historis berkinerja tinggi, yang basis kode C-nya menjadi tulang punggung bagi pustaka *PyEphem* di Python.

Y

- **Y-Axis (Sumbu Ordinat):** Sumbu kartesian yang digunakan dalam matematika simulasi 3D KHGT Times untuk merender matriks rotasi proyeksi ke arah vertikal (*zenith*).

Z

- **Z-Buffer (Painter's Algorithm):** Algoritma manipulasi kedalaman gambar pada kanvas 3D yang mengatur agar benda astronomi yang dekat selalu digambar menutupi benda yang jauh.
- **Zawal:** Waktu ketika matahari tepat melintasi garis meridian pengamat (kulminasi atas), menandai masuknya waktu awal salat Dzuhur.

LAMPIRAN 1: TABEL PARAMETER, KOORDINAT REFERENSI, DAN ILUSTRASI VISUAL

Lampiran ini memuat matriks data komputasi, konstanta astronomis, serta panduan tata letak visual yang merangkum keseluruhan algoritma dan parameter yang beroperasi di dalam arsitektur KHGT Times V7.4.

1. Tabel Parameter Visibilitas Hilal (Kriteria KHGT vs MABIMS)

Tabel berikut merupakan rangkuman batas ambang astronomis yang dieksekusi oleh mesin iterasi *Skyfield* pada Modul 5 dan Modul 20 untuk menentukan masuknya awal bulan kalender.

Sistem Kalender	Sifat Kalender	Kriteria Ketinggian (Alt)	Kriteria Elongasi	Titik Referensi (Horizon)	Basis Pengukuran
KHGT (PKG 1)	Global Hakiki	$h \geq 5°$	$\varepsilon \geq 8°$	Daratan Utama (*Mainland*) di mana saja	Geosentris
KHGT (PKG 2)	Global Hakiki	$h \geq 5°$	$\varepsilon \geq 8°$	Benua Amerika (Sebelum Fajar Gisborne)	Geosentris
Neo MABIMS	Regional	$h \geq 3°$	$\varepsilon \geq 6{,}4°$	Kawasan Asia Tenggara (Sabang, Indonesia)	Toposentris
Wujudul Hilal	Regional/Lokal	$> 0°$	Bebas	Pelabuhan Ratu / Yogyakarta	Toposentris

2. Daftar Koordinat Stasiun Observasi dan Referensi Global

Data koordinat geografis berikut menggunakan datum WGS84 dan digunakan sebagai titik jangkar (*anchor points*) dalam algoritma *brute-force* KHGT Times V7.4.

Nama Lokasi	Negara/Wilayah	Lintang (Latitude)	Bujur (Longitude)	Fungsi dalam KHGT Times V7.4
Makkah (Ka'bah)	Arab Saudi	21.4225° LU	39.8262° BT	Titik absolut penentuan *Great Circle* arah Kiblat.
Greenwich	Inggris Raya	51.4769° LU	0.0005° BB	Basis patokan zona waktu UTC.
Sabang	Indonesia	5.8943° LU	95.3184° BT	Titik komparasi paling barat untuk uji MABIMS.
Gisborne	Selandia Baru	38.6623° LS	178.0176° BT	Titik penguji batas fajar (Syarat PKG 2).
Titik Nol IDL	Samudra Pasifik	0.0000°	pm 180.0000°	Garis belahan rekonstruksi *grid array* SciPy.

3. Tabel Konstanta Sudut Depresi Fajar dan Isya

Tabel ini merangkum ketetapan sudut matahari di bawah horizon yang digunakan oleh Modul Waktu Salat (Bab 6) untuk mengalkulasi *Astronomical Twilight*.

Otoritas / Lembaga	Sudut Fajar (Subuh)	Sudut Isya	Catatan Eksekusi Kode
Kemenag RI	-20°	-18°	Membutuhkan nilai *Sky Quality Meter* (SQM) yang lebih gelap.
Muhammadiyah	-18°	-18°	Berdasarkan koreksi riset visibilitas fajar shadiq terbaru.
Umm al-Qura (Makkah)	-18.5°	+90 menit	Isya dipatok konstan 90 menit setelah Maghrib (120 menit saat Ramadhan).
MWL (Liga Muslim)	-18v	-17°	Standar umum untuk wilayah Eropa dan lintang menengah.

LAMPIRAN 2 APLIKASI PEMOTONG DAN PENGGABUNG BERKAS BSP

Python

```
import streamlit as st
import os
import spiceypy as spice
from skyfield.api import load
from jplephem.spk import SPK
import time
import traceback

# ==========================================
# INISIALISASI SPICE
# ==========================================
spice.kclear()  # Bersihkan kernel pool saat aplikasi dimulai

# ==========================================
# KONFIGURASI UI & SETUP TIME
# ==========================================
st.set_page_config(page_title="JPL BSP Editor & Manager", layout="wide")

st.title("🛠️ JPL Ephemeris Manager: Trim, Merge & Range")
st.markdown("Aplikasi profesional untuk manipulasi file kernel SPK (.bsp).")

ts = load.timescale()

def year_to_et(year):
    t = ts.utc(int(year), 1, 1)
    return (t.tdb - 2451545.0) * 86400.0

def et_to_year(et):
    jd_tdb = 2451545.0 + (et / 86400.0)
    t = ts.tdb_jd(jd_tdb)
    return int(t.utc[0])

def reset_spice():
    """Membersihkan pool kernel SPICE."""
    try:
        spice.kclear()
    except:
        pass

# ==========================================
# FUNGSI DETEKSI RENTANG (MENGGUNAKAN JPLEPHEM)
# ==========================================
def get_bsp_range_safe(file_path):
```

```
    try:
        kernel = SPK.open(file_path)
        min_jd = float('inf')
        max_jd = float('-inf')
        ids = set()

        for segment in kernel.segments:
            if segment.start_jd < min_jd: min_jd = segment.start_jd
            if segment.end_jd > max_jd: max_jd = segment.end_jd
            ids.add(segment.target)

        min_et = (min_jd - 2451545.0) * 86400.0
        max_et = (max_jd - 2451545.0) * 86400.0

        kernel.close()

        return et_to_year(min_et), et_to_year(max_et), list(ids)
    except Exception as e:
        return None, None, str(e)

# ==========================================
# SIDEBAR & MENU UTAMA
# ==========================================
bsp_files = [f for f in os.listdir('.') if f.endswith('.bsp')]
if not bsp_files:
    st.sidebar.error("Tidak ada file .bsp ditemukan.")
    st.stop()

tab1, tab2 = st.tabs(["✂️ Trim (Potong File)", "🔗 Merge (Gabungkan File)"])

# ==========================================
# TAB 1: PEMOTONGAN FILE (TRIM)
# ==========================================
with tab1:
    st.header("Potong (Trim) Rentang Tahun")
    src_file = st.selectbox("Pilih file sumber", bsp_files, key="trim_src")

    y_min, y_max, info = get_bsp_range_safe(src_file)

    if y_min is not None:
        st.info(f"🧮 **Informasi File:** {src_file} ({y_min} s/d {y_max})")

        col1, col2 = st.columns(2)
        with col1:
            new_y_start = st.number_input("Tahun Mulai", min_value=y_min, max_value=y_max,
value=y_min, key="t1")
        with col2:
```

```
        new_y_end = st.number_input("Tahun Selesai", min_value=y_min, max_value=y_max,
value=y_max, key="t2")

    new_name = st.text_input("Nama output", value="trimmed_" + src_file)

    if st.button("Proses Potong"):
        reset_spice()
        if os.path.exists(new_name):
            try:
                os.remove(new_name)
            except PermissionError:
                st.error(f"File '{new_name}' sedang digunakan. Tutup file tersebut atau gunakan nama lain.")
                st.stop()
            except Exception as e:
                st.error(f"Gagal menghapus file lama: {e}")
                st.stop()

        try:
            et_start = year_to_et(new_y_start)
            et_end = year_to_et(new_y_end)

            handle_new = spice.spkopn(new_name, "TRIM_FILE", 0)
            handle_src = spice.dafopr(src_file)

            spice.dafbfs(handle_src)
            segmen_ditemukan = False
            while spice.daffna():
                descr_full = spice.dafgs()        # Dapatkan descriptor lengkap (125 elemen)
                ident = spice.dafgn()
                dc, ic = spice.dafus(descr_full, 2, 6)  # Unpack: 2 double, 6 integer

                overlap_start = max(et_start, dc[0])
                overlap_end = min(et_end, dc[1])

                if overlap_start < overlap_end:
                    # spksub membutuhkan 5 elemen pertama dari descriptor
                    descr5 = descr_full[:5]
                    spice.spksub(handle_src, descr5, ident, overlap_start, overlap_end, handle_new)
                    segmen_ditemukan = True

            if not segmen_ditemukan:
                st.warning("⚠ Tidak ada segmen yang tercakup dalam rentang waktu yang dipilih.")

            spice.dafcls(handle_src)
            spice.spkcls(handle_new)

            st.success(f"☑ Sukses: {new_name} berhasil dibuat.")
        except Exception as e:
```

```
            st.error(f"Gagal memproses file: {e}")
            st.text(traceback.format_exc())
    else:
        st.error(f"Gagal membaca file {src_file}: {info}")

# ========================================
# TAB 2: PENGGABUNGAN FILE (MERGE)
# ========================================
with tab2:
    st.header("Gabungkan (Merge) File BSP")
    files_to_merge = st.multiselect("Pilih file", bsp_files)

    default_merged = f"merged_{int(time.time())}.bsp"
    merged_name = st.text_input("Nama file gabungan", value=default_merged)

    if st.button("Proses Gabung"):
        if len(files_to_merge) < 2:
            st.warning("Pilih minimal 2 file.")
        else:
            reset_spice()
            if os.path.exists(merged_name):
                try:
                    os.remove(merged_name)
                except PermissionError:
                    st.error(f"File '{merged_name}' sedang digunakan. Gunakan nama lain atau tunggu beberapa
saat.")
                    st.stop()
                except Exception as e:
                    st.error(f"Gagal menghapus file lama: {e}")
                    st.stop()

            try:
                handle_new = spice.spkopn(merged_name, "MERGE_FILE", 0)

                total_segments = 0
                for f in files_to_merge:
                    handle_src = spice.dafopr(f)
                    spice.dafbfs(handle_src)
                    while spice.daffna():
                        descr_full = spice.dafgs()
                        ident = spice.dafgn()
                        dc, ic = spice.dafus(descr_full, 2, 6)

                        descr5 = descr_full[:5]   # Ambil 5 elemen pertama
                        spice.spksub(handle_src, descr5, ident, dc[0], dc[1], handle_new)
                        total_segments += 1
                    spice.dafcls(handle_src)
```

```
    spice.spkcls(handle_new)
    st.success(f"☑ Gabungan berhasil: {merged_name} (Total {total_segments} segmen)")
except Exception as e:
    st.error(f"Gagal menggabung: {e}")
    st.text(traceback.format_exc())
```

LAMPIRAN 3 SOURCE CODE KHGT TIMES V7.4

PYTHON

```
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 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 = {}
# ------------------------------------
```

```
# --- GUI LIBRARIES ---
import customtkinter as ctk
import tkinter as tk
from tkinter import messagebox, filedialog, ttk
from PIL import Image, ImageTk, ImageDraw, ImageFont
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
# ========================================
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_monthrange(year, month):
    """Pengganti calendar.monthrange 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 string menjadi integer agar kompatibel dengan HIJRI_DB
            external_db = {int(k): v for k, v in external_db.items()}
            HIJRI_DB.update(external_db)
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), "Maumere": (-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": (51.5074, -0.1278),
```

```
    "Birmingham": (52.4862, -1.8904),
    "Manchester": (53.4808, -2.2426),
    "Glasgow": (55.8642, -4.2518),
    "Edinburgh": (55.9533, -3.1883)
  },
  "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"{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.utc_datetime() # 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&timezone=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 "◯ Data cuaca tidak tersedia (Offline)", "#9E9E9E"


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

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

        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, "adzan.mp3")
        if os.path.exists(default_audio):
            self.adzan_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 = ephem.Observer()
    self.pe_sun = ephem.Sun()
    self.pe_moon = ephem.Moon()

    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 7.4, By Kasmui"
    ]
    return "\n".join(line.center(width) for line in lines)

def load_ephemeris(self):
    try:
        # File tunggal sesuai permintaan
        filename = "de441-new.bsp"
        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: messagebox.showerror("File Ephemeris Hilang",
                f"File '{filename}' wajib ada di folder aplikasi untuk menjalankan kalkulasi.\n\n"
                "Sistem tidak mendukung file alternatif."))
            return

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

        # Load file ke engine
        self.eph = self.load_obj(filename)
```

```
            self.ephemeris_name = filename

            # Verifikasi objek utama
            if 'earth' not in self.eph or 'sun' not in self.eph or 'moon' not in self.eph:
                raise ValueError("Data planet (Earth/Sun/Moon) tidak ditemukan dalam file tersebut.")

            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: messagebox.showerror("Error Mesin", f"Gagal membaca file: {str(e)}"))

    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 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")

        ctk.CTkLabel(self.sidebar, text="KHGT ENGINE", font=("Segoe UI", 24, "bold"),
text_color="#00E5FF").pack(pady=(20, 5))

        # ========================================================
        # 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 = [
    "1) Visibility Hilal (KHGT)",
    "2) Crescent Visibility Map",
    "3) Analisis Hilal Global",
    "4) Altitude Chart Analyser",
    "5) Kota Pertama KHGT (Mainland)",
    "6) Fase Bulan (Moonphase)",
    "7) Moon Times",
    "8) Sun Times",
    "9) Sun Moon Ephemeris",
    "10) Qibla Time (Rashdul Lokal)",
    "11) Qiblah Direction & Times",
    "12) Prayer Times",
    "13) Konversi Kalender",
    "14) Analisis Gerhana",
    "15) Live Animasi",
    "16) Sistem Sun Moon Earth",
    "17) Equinox & Solstice",
    "18) Planetary Times",
    "19) Simulasi Ephemeris 3D",
    "20) Komparasi 50 Tahun (Ramadhan)",
    "21) Komparasi 50 Tahun (Syawal)",
    "22) Tabel Tinggi Hilal 50 Thn (Ramadhan)",
    "23) Tabel Elongasi Hilal 50 Thn (Ramadhan)",
    "24) Tabel Tinggi Hilal 50 Thn (Syawal)",
    "25) Tabel Elongasi Hilal 50 Thn (Syawal)",
    "26) Kalender Hijriah Berjalan",
    "27) Kalender Masehi Berjalan",
    "28) WinAI: Aplikasi AI Windows",
    "29) Kalkulator Mizwala (Bayangan)",
    "30) Status Kriteria Batas",
    "31) Astrofotografi & Kontras Visibilitas",
    "32) Prediktor Pasang Surut Gravitasi",
    "33) Evaluasi Fajar Shadiq & SQM",
    "34) Generator Analemma Matahari",
    "35) Komparasi 50 Thn (Zulhijah)",          # BARU
    "36) Tabel Tinggi Hilal 50 Thn (Zulhijah)",   # BARU (Dipersiapkan)
    "37) Tabel Elongasi Hilal 50 Thn (Zulhijah)",  # BARU (Dipersiapkan)
    "38) HIJRI_DB Auto-Builder (Admin)"          # BERUBAH DARI 35 KE 38
]
```

```
        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
            # ------------------------------------------------

            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 (%):", "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))

        # ========================================
        # -- 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.setup_gerhana_out_frame()
        self.setup_animasi_out_frame()
        self.setup_3d_out_frame()
        self.setup_kriteria_batas_out_frame()

        # ---> 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()

        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_myear = ctk.CTkFrame(self.frame_moon_inputs, fg_color="#212121")
        frame_myear.pack(fill="x", padx=15, pady=10)
        ctk.CTkLabel(frame_myear, text="TAHUN MASEHI", font=("Segoe UI", 12, "bold")).pack(anchor="w",
padx=10, pady=(10, 5))
        self.entry_moon_year = ctk.CTkEntry(frame_myear, 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_qmap.pack(fill="x", padx=15, pady=(10, 5))

        # -- 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 (%):", "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")

    # -- 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_vmmonth = ctk.CTkEntry(vmdate_grid, width=45, placeholder_text="mm",
justify="center")
    self.entry_vmmonth.insert(0, curr_m)
    self.entry_vmmonth.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")
        ctk.CTkRadioButton(frame_conv_mode, text="Masehi ➔ Hijriah", 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 ➔ Masehi", variable=self.radio_conv_var,
value="h2m", 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))

        # Baris 0: Label Petunjuk
        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))

        # Baris 1: Kotak Input
```

```
        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))

        # ---> PERBAIKAN: Tambahan Label hh:mm:ss <---
        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))

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

        # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
        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="TANGGAL OBSERVASI", 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))

        # Baris 0: Label Petunjuk
        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))

        # Baris 1: Kotak Input
        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))

        # -- 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 (Iluminasi): {iluminasi_fase:>5.1f}%"
        )

        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):
    # 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()

    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 "Visibility Hilal" in mode:
      self.frame_vis_inputs.pack(fill="x", before=self.btn_hitung)
      self.lbl_main_title.configure(text="Laporan Analisis Hilal & 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 HD 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 (Iterasi Ephem)")
            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="Data Siklus Fase Bulan (Moonphase)")
            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="Konversi Kalender Masehi & Hijriah (KHGT)")
            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 (Waktu Arah Kiblat)")
            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="Kalkulator Gerhana Presisi (Solar & Lunar Auto-Scanner)")
            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 Simulator Posisi Benda Langit")
            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.lbl_status.configure(text="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="Titik Pertama KHGT (Daratan Utama)")
            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="Kalkulator Equinox & Solstice")
            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="Simulasi Ephemeris 3D (Matahari & Bulan)")

            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 (KHGT vs MABIMS Sabang)")
            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 (KHGT vs MABIMS Sabang)")
            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" in mode:
            self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Ramadhan (KHGT vs MABIMS Sabang)")
            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 Interaktif (Global)")

            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 Interaktif (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(row=1, column=0, sticky="nsew")

        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)



    # ========================================
    # 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=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()
                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_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', '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.date.today()
        hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][now.weekday()]
        masehi_str = f"{hari}, {now.day} {BULAN_MASEHI[now.month-1]} {now.year}"
        hijri_str = f" {HijriConverter.get_hijri_date(now)}"
        self.lbl_cal_masehi.configure(text=masehi_str)
        self.lbl_cal_hijri.configure(text=hijri_str)
        self.after(3600000, self.update_calendar_widget)

    def update_conv_ui(self):
        now = datetime.datetime.now()
        if self.radio_conv_var.get() == "m2h":
            self.lbl_conv_date.configure(text="TANGGAL MASEHI")
            self.combo_conv_month.configure(values=BULAN_MASEHI)
            self.combo_conv_month.set(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=BULAN_HIJRIAH)
            self.combo_conv_month.set("Ramadan")
            self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 31)])

    def show_release_info(self):
```

```
        # 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()

        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)
        tb.tag_config("body", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=15, lmargin2=15, spacing1=5, spacing3=10)

        # 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)
        tb.tag_config("menu_desc", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=45, lmargin2=45, spacing1=2, spacing3=8)
```

```
    tb.tag_config("list_item", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=35, lmargin2=55, spacing1=5, spacing3=5)
    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 = "=" * 74
    garis_strip = "-" * 74

    # ====================== HEADER ======================
    t1 = "⚽ ⚽ ⚽ ⚽ ⚽ ⚽ ⚽ ⚽ ⚽\n"
    t2 = "INFORMASI RILIS & DOKUMENTASI SISTEM KHGT TIMES V7.4\n"
    t3 = "Akses dan download melalui link: https://hisabmu.com/khgttimes/\n\n"

    self.textbox.insert("end", t1, "emoji_header")
    self.textbox.insert("end", t2, "title")
    self.textbox.insert("end", t3, "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']

        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

            cell = ctk.CTkFrame(self.demo_cal_grid, fg_color="#0F172A", corner_radius=6,
border_width=b_width, border_color=b_color, width=200, height=75)
            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 = 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

            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):
                  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")

        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 V7.4\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.\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. Sistem mengekstrak file de441-new.bsp (rentang waktu 0 M s/d 3000 M) untuk mendapatkan posisi absolut vektor Tata Surya (X, Y, Z)."),
            ("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 (38 MENU) ====================
        self.textbox.insert("end", "\n" + garis_strip + "\n[ 3. MANUAL BOOK: PENJELASAN 38 MODUL MENU V7.4 ]\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) 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)."),

            ("2) 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."),

            ("3) 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."),
```

("4) 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)."),

("5) 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)."),

("6) 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/Ijtimak, First Quarter, Full Moon/Purnama, Last Quarter) secara berurutan beserta Jam:Menit-nya."),

("7) 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."),

("8) 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)."),

("9) 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."),

("10) 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."),

("11) 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."),

("12) 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)."),

("13) Konversi Kalender (0 M s/d 3000 M)",
"• 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."),

("14) 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 mengekspor Jalur Totalitas Gerhana Matahari ke format Google Earth (KML)."),

("15) Live Animasi",
"• Kegunaan: Simulator posisi benda langit secara real-time.\n"
"Menampilkan visualisasi pergerakan Matahari dan Bulan di atas cakrawala pengamat secara dinamis. Berguna untuk memonitor posisi hilal sesaat setelah matahari terbenam dengan rotasi derajat kemiringan cahaya sabit yang akurat.\n"
"• Sub-Menu Input: Waktu Simulasi (Tanggal & Jam), atau biarkan di Mode LIVE.\n"
"• Aksi & Output: Melukis garis ufuk dan memproyeksikan Gambar Matahari dan Bulan. Dilengkapi tombol 'Zoom Hilal' untuk observasi jarak dekat."),

("16) 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."),

("17) 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."),

("18) 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)."),

("19) 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."),

("20) 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)'."),

("21) 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."),

("22 s.d 25) Tabel Parameter Extractor (50 Tahun)",
"• Kegunaan: Penyajian data statistik parameter hilal jangka panjang.\n"
"Kumpulan modul ini menyediakan data mentah (raw data) untuk kebutuhan akademis dan analisis tren visibilitas hilal selama 50 tahun ke depan, baik secara geosentrik maupun toposentrik.\n"
"• Sub-Menu Input: Proses Data Otomatis.\n"
"• Aksi & Output: Tabel mentah nilai Altitude dan Elongasi yang sangat cocok diekspor ke format CSV Excel."),

("26) 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."),

("27) 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."),

("28) 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)."),

("29) 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."),

("30) 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."),

("31) 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."),

("32) 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."),

("33) 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."),

("34) 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."),

("35) Komparasi 50 Thn (Zulhijah)",
"• Kegunaan: Analisis sinkronisasi penentuan awal bulan Zulhijah antara KHGT dan MABIMS.\n"

```
        "Modul ini mensimulasikan data 5 dekade ke depan untuk melihat perbedaan tanggal masuknya 1 Zulhijah (Idul Adha).\n"
        "• Sub-Menu Input: Eksekusi Otomatis.\n"
        "• Aksi & Output: Laporan matriks perbandingan 50 tahun."),

       ("36) Tabel Tinggi Hilal 50 Thn (Zulhijah)",
        "• Kegunaan: Penyajian data statistik ketinggian hilal awal Zulhijah jangka panjang.\n"
        "Sistem akan mengekstrak data mentah ketinggian hilal (Altitude) selama 50 tahun ke depan, khusus bulan Zulhijah.\n"
        "• Sub-Menu Input: Proses Data Otomatis.\n"
        "• Aksi & Output: Tabel mentah nilai Altitude yang siap diekspor ke format CSV Excel."),

       ("37) Tabel Elongasi Hilal 50 Thn (Zulhijah)",
        "• Kegunaan: Penyajian data statistik elongasi hilal awal Zulhijah jangka panjang.\n"
        "Menyediakan data mentah sudut Elongasi untuk analisis tren visibilitas hilal 50 tahun ke depan, khusus 1 Zulhijah.\n"
        "• Sub-Menu Input: Proses Data Otomatis.\n"
        "• Aksi & Output: Tabel mentah nilai Elongasi yang siap diekspor ke format CSV Excel."),

       ("38) 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 7.4) ]\n" + garis_strip + "\n", "subtitle")

    updates = [
        ("[V7.4 - ENGINE] Penyederhanaan Ephemeris DE441", "Sistem kini HANYA menggunakan satu file ephemeris mutakhir yaitu 'de441-new.bsp' untuk seluruh kalkulasi pada rentang waktu 1 s/d 2999 Masehi. File .bsp lainnya tidak lagi digunakan dan dapat dihapus dari folder aplikasi untuk menghemat penyimpanan."),
        ("[V7.3 - MODUL 31-34] Modul Analisis Fisis Baru", "Penambahan fitur Astrofotografi, Pasang Surut, Evaluasi SQM Fajar, dan Generator Analemma Matahari untuk riset yang lebih mendalam."),
        ("[V7.3 - MODUL 35] HIJRI_DB Auto-Builder", "Tool khusus Admin/Developer untuk merekonstruksi dan merender ulang seluruh siklus penanggalan KHGT selama ratusan tahun secara otomatis."),
```

```
        ("[V7.3 - WINAI] Optimasi Engine Gemini", "Asisten AI kini lebih stabil dalam melakukan kueri pencarian internet otonom dan rujukan dokumen fikih/tarjih."),
        ("[V7.3 - UI/UX] Refinement Kalender", "Penyempurnaan tata letak matriks kalender visual dan perbaikan engine Pillow (PIL) agar hasil ekspor PNG/PDF tetap tajam pada resolusi tinggi."),
        ("[V7.3 - ENGINE] Akurasi JPL DE4xx", "Optimalisasi penanganan data ephemeris NASA untuk mendukung perhitungan tahun-tahun ekstrem tanpa hambatan sistem.")
    ]
    for key, val in updates:
        self.textbox.insert("end", " ▹ ", "highlight")
        self.textbox.insert("end", f"{key}: ", "highlight")
        self.textbox.insert("end", f"{val}\n\n", "list_item")

    # ==================== BAGIAN 5: PANDUAN PENGGUNAAN ====================
    self.textbox.insert("end", garis_strip + "\n[ 5. PANDUAN OPERASIONAL & EXPORT ]\n" + garis_strip + "\n", "subtitle")

    guides = [
        ("NAVIGASI MODUL", "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 38 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.")
    ]
    for key, val in guides:
        self.textbox.insert("end", " ▹ ", "highlight")
        self.textbox.insert("end", f"{key} : ", "tech")
        self.textbox.insert("end", f"{val}\n", "list_item")

    # ==================== FOOTER COPYLEFT ====================
    footer_text = "\n\n" + garis_sama + "\n"
    footer_text += "Copyleft (ɔ) Kasmui, 2026. All Left Reserved.\n"
    footer_text += "Perangkat lunak ini didedikasikan secara penuh untuk kemajuan sains dan astronomi Islam global.\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 V7.4")

    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):
        """
        Validator Rentang Tahun untuk de441-new.bsp (1 s/d 2999 M).
        Menggantikan logika auto-switch lama.
        """
        # 1. Cek apakah Ephemeris sudah dimuat saat startup
        if self.eph is None:
            self.load_ephemeris()
            if self.eph is None: # Jika masih gagal setelah dicoba load
                raise RuntimeError("Mesin Astronomi belum siap. File .bsp mungkin hilang.")

        # 2. Validasi Rentang Tahun (1 - 2999 M)
        if target_year <= 0 or target_year >= 3000:  # <--- Ubah kondisinya di sini
            err_msg = f"Tahun {target_year} di luar jangkauan!"
            # v--- Ubah pesan notifikasinya di sini
            detail_msg = f"Sistem hanya mendukung rentang tahun dari 1 s/d 2999 Masehi.\n\nInput Anda:
{target_year}\n\nSilakan masukkan input tahun dari 1 s/d 2999 M."

            # Tampilkan peringatan di GUI (menggunakan after(0) karena biasanya dipanggil dari thread)
            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"))

            # Hentikan eksekusi thread kalkulasi
            raise ValueError(err_msg)

        # Jika lolos validasi, biarkan proses berlanjut tanpa mengganti file
        pass

    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 Thn (Zulhijah)" in mode:
            self.analisis_komparasi_zulhijah_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 14: ANALISIS HILAL GLOBAL ITERATIF (EPHEM)
# ==========================================
def calculate_global_hilal(self):
    try:
        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
        t_ref = self.ts.utc(y, m, d, h, mn, s)

        # Cari waktu ijtimak (Skyfield)
        t0_ij = self.ts.tt_jd(t_ref.tt - 35)
        t1_ij = self.ts.tt_jd(t_ref.tt + 35)
        t_phases, y_phases = almanac.find_discrete(t0_ij, t1_ij, almanac.moon_phases(self.eph))

        t_ijtimak = None
        for t_ev, phase in zip(np.atleast_1d(t_phases), np.atleast_1d(y_phases)):
            if phase == 0 and t_ev.tt <= t_ref.tt:
                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")

        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 Ijtimak/Konjungsi Terdekat (UTC): {str_ijtimak}\n"
        header += f"* Catatan: Diurutkan Berdasarkan Abjad Negara, lalu Abjad Kota.\n"
        header += "="*138 + "\n"
```

```
        header += f"{'Negara':<15} | {'Kota':<14} | {'Sunset (UTC)':<16} | {'Umur Bln':<9} | {'Alt(Geo)':<8} | {'Eln(Geo)':<8} | {'Alt(Top)':<8} | {'Eln(Top)':<8} | Status\n"
        header += "-"*138
        output_lines.append(header)

        hasil_kalkulasi = []

        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)

                # Cari Sunset H-0
                t0_ss = self.ts.tt_jd(t_ref.tt - 0.5)
                t1_ss = self.ts.tt_jd(t_ref.tt + 0.5)
                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 zip(np.atleast_1d(t_ss_evs), np.atleast_1d(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)"
                    })
                    continue

                # 1. Parameter Toposentris (Skyfield)
                topo_obs = (earth + loc).at(t_sunset)
                app_moon_topo = topo_obs.observe(moon).apparent()
                app_sun_topo = topo_obs.observe(sun).apparent()

                alt_topo = app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees
                elong_topo = app_sun_topo.separation_from(app_moon_topo).degrees

                # 2. Parameter Geosentris (Skyfield)
                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

                # Altitude Geosentrik
```

```
            ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset)
            lst_deg = (t_sunset.gast * 15.0) + bujur
            ha_deg = lst_deg - (ra_m.hours * 15.0)
            ha_rad, lat_rad, d_rad = math.radians(ha_deg), math.radians(lintang), 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))))

            umur_bulan_desimal = (t_sunset.tt - t_ijtimak.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_topo >= 3.0 and elong_topo >= 6.4: status_visibilitas = "Memenuhi Kriteria MABIMS"
               elif alt_geo > 0 or alt_topo > 0: status_visibilitas = "Wujudul Hilal"
               else: status_visibilitas = "Negatif (Bawah Ufuk)"

            alt_g_str, elong_g_str = f"{alt_geo:.2f}°", f"{elong_geo:.2f}°"
            alt_t_str, elong_t_str = f"{alt_topo:.2f}°", f"{elong_topo:.2f}°"
            sunset_str = t_sunset.utc_datetime().strftime("%Y/%m/%d %H:%M")

            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}"

            hasil_kalkulasi.append({
               'negara': negara, 'kota': nama_kota, 'waktu_sort': t_sunset.tt, 'baris_teks': baris_teks
            })

        hasil_kalkulasi.sort(key=lambda x: (x['negara'].lower(), x['kota'].lower()))

        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)
        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
            # Sederhanakan provinsi Indonesia menjadi "Indonesia"
            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"

            # Ambil 1 kota perwakilan, konversi ke objek lokasi Skyfield
            first_city = list(cities.keys())[0]
            lat, lon = cities[first_city]
            if nama_tampil not in perwakilan_locs:
                perwakilan_locs[nama_tampil] = []

            # Batasi maks 3 sampel kota per wilayah raksasa agar iterasi tidak berat
            if len(perwakilan_locs[nama_tampil]) < 3:
                loc_obj = earth + wgs84.latlon(lat, lon)
                perwakilan_locs[nama_tampil].append(loc_obj)
        # ==========================================================================

        # --- 1. PENCARIAN GERHANA BULAN ---
        t_bulan, y_bulan, _ = eclipselib.lunar_eclipses(t0, t1, self.eph)
        jenis_bulan_map = {0: 'Penumbra', 1: 'Sebagian (Partial)', 2: 'Total'}
```

```
        if t_bulan is not None:
            for t, y in zip(np.atleast_1d(t_bulan), np.atleast_1d(y_bulan)):

                # Cek Visibilitas Indonesia
                m_app = indo_loc.at(t).observe(moon).apparent()
                alt_m, _, _ = m_app.altaz()
                indo_vis = "Terlihat Jelas (Bulan di atas ufuk)" if alt_m.degrees > 0 else "Tidak Terlihat (Siang
di Indonesia)"

                # SCANNING NEGARA GLOBAL (Bulan Terlihat = Malam Hari)
                wilayah_terlihat = set()
                for nama_wil, list_loc in perwakilan_locs.items():
                    for loc in list_loc:
                        alt_obj, _, _ = loc.at(t).observe(moon).apparent().altaz()
                        if alt_obj.degrees > 0: # Jika bulan di atas ufuk, berarti terlihat
                            wilayah_terlihat.add(nama_wil)
                            break

                if not wilayah_terlihat:
                    wil_str = "Hanya terlihat di Samudra Terpencil"
                else:
                    wil_str = ", ".join(sorted(list(wilayah_terlihat)))

                all_eclipses.append({
                    'objek': 'Bulan',
                    'jenis': jenis_bulan_map.get(int(y), 'Unknown'),
                    't_peak': t,
                    't_mulai': self.ts.tt_jd(t.tt - 0.1),
                    't_akhir': self.ts.tt_jd(t.tt + 0.1),
                    'wilayah': wil_str,
                    'indo_vis': indo_vis
                })

        # --- 2. PENCARIAN GERHANA MATAHARI ---
        t_phases, y_phases = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
        if t_phases is not None:
            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)"

# Cek Visibilitas Indonesia (Spesifik Gerhana Matahari)
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): # Masuk radius bayangan penumbra
        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)"

# SCANNING NEGARA GLOBAL KHUSUS GERHANA MATAHARI
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()
        # Gerhana matahari hanya terlihat jika matahari sedang di atas ufuk (Siang)
        if alt_s_wil.degrees > 0:
            m_app = loc.at(t_p).observe(moon).apparent()
            sep_wil = s_app.separation_from(m_app).degrees
            # Cek apakah jarak sudut matahari & bulan cukup dekat untuk tertutup bayangan
            if sep_wil < (sd_s + sd_m + 0.8):
                wilayah_terlihat.add(nama_wil)
                break

if not wilayah_terlihat:
    wil_str = "Hanya melintasi Samudra / Antartika"
else:
    wil_str = ", ".join(sorted(list(wilayah_terlihat)))

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
            })

        # Urutkan berdasarkan waktu kronologis
        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{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}" if yr > 0 else f"{abs(yr-1)} 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 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:
            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)

            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 = []
        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>"""

        filepath = filedialog.asksaveasfilename(
            initialfile=f"Jalur_Gerhana_Matahari_{dt_naive.strftime('%Y%m%d')}.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:
            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)
            waktu_puncak_skyfield = self.ts.from_datetime(dt_wib)

            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"\nIlluminasi (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"\nIlluminasi (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: LIVE ANIMASI (DENGAN ZOOM & FIX MAGHRIB)
    # ==========================================
    def setup_animasi_out_frame(self):
        self.frame_animasi_out = ctk.CTkFrame(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: POSISI BENDA LANGIT", font=("Segoe UI", 16, "bold"), text_color="#FF5252").pack(side="left", padx=15, pady=10)

        # ---> TAMBAHAN TOMBOL ZOOM (MODUL 15) <---
        self.is_anim_zoomed = False
        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)

        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)

        self.anim_canvas = tk.Canvas(self.frame_animasi_out, bg='#030514', highlightthickness=0)
        self.anim_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))
        self.anim_canvas.bind("<Configure>", self.draw_static_background)

    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 draw_static_background(self, event=None):
        """Menggambar Ufuk, Bintang, dan Skala Kompas"""
        if getattr(self, 'is_anim_zoomed', False):
            self.anim_canvas.delete("static")
            return # Jangan gambar background ruwet saat mode zoom

        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)

        self.anim_canvas.create_rectangle(0, horizon_y, width, height, fill="#0A150A", outline="", tags="static")
        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 Acak
        for _ in range(150):
            x = random.randint(0, width)
            y = random.randint(0, int(horizon_y))
            r = random.uniform(0.5, 1.5)
            color = random.choice(["#FFFFFF", "#E0F7FA", "#FFFDE7"])
            self.anim_canvas.create_oval(x-r, y-r, x+r, y+r, fill=color, outline="", tags="static")

        # Skala Kompas Standar
        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")

    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")

        # --- FUNGSI BANTUAN PENENTU ARAH MATA ANGIN (SINGKATAN AGAR TIDAK NUMPUK) ---
        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

            astro_sun = observer.at(t_sim).observe(sun).apparent()
            alt_sun, az_sun, _ = astro_sun.altaz()

            astro_moon = observer.at(t_sim).observe(moon).apparent()
            alt_moon, az_moon, _ = astro_moon.altaz()
```

```
        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 = ephem.Date(t_sim.utc_datetime())
        ijtimak = ephem.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 (FOKUS & TRACKING AZIMUT DINAMIS) ----
            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)

            # Gambar Ufuk Khusus Mode Zoom
            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")
```

```
            # --- SKALA PENGGARIS UFUK (ANTI NUMPUK) ---
            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:
                    # Membuat panjang garis penggaris bervariasi
                    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")

                    # Tulis Angka Derajat tiap 10 derajat
                    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")

                    # Tulis Nama Mata Angin HANYA pada 8 Arah Utama (Kelipatan 45 derajat)
                    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")

            # --- PROYEKSI AZIMUT MATAHARI KE UFUK ---
            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")

            # --- PROYEKSI AZIMUT HILAL KE UFUK ---
            y_offset_moon = 45
            # Geser teks hilal ke bawah jika terlalu dekat dengan teks matahari agar tidak bertumpuk
            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 360°) ----
            horizon_y = height / 2
            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 (Berlaku untuk Mode Zoom maupun Normal)
        y_sun = horizon_y - (alt_s_deg * px_per_deg_y)
        y_moon = horizon_y - (alt_m_deg * px_per_deg_y)

        # ========================================================================
        # PERBAIKAN: Offset visual agar piringan atas terbenam tepat saat Maghrib.
        # Saat Maghrib (Sunset visual), alt_s_deg (Geometris/Pusat) sekitar -0.833°.
        # Kita menarik posisi gambar secara merata ke bawah agar piringan atas persis di ufuk.
        # ========================================================================
        offset_visual_y = r_sun - (0.833 * px_per_deg_y)
        y_sun += offset_visual_y
        y_moon += offset_visual_y

        if not hasattr(self, '_live_images'):
            self._live_images = {}

        # --- 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")
        except Exception as e:
            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")
            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_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")
        except Exception as e:
```

```
            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")
            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_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 (TAMPIL DI NORMAL & ZOOM MODE) ---
        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)
```

```
        box_text = (
            f"TRACKING HILAL & MATAHARI (REAL-TIME DATA)\n{waktu_str}\n"
            f"Umur Bulan     : {moon_age:.2f} hari\n"
            f"Elongasi (Geo) : {elong_geo:.2f}° | Beda Azimuth : {beda_az:.2f}°\n"
            f"-------------------------------------------------\n"
            f"MATAHARI : {txt_fisik_sun:<15} | Azimut: {az_s_deg:06.2f}° ({arah_sun})\n"
            f"HILAL    : {txt_fisik_moon:<15} | Azimut: {az_m_deg:06.2f}° ({arah_moon})"
        )

        # --- PERBAIKAN: Background transparan total & border dihilangkan ---
        # Baris img_bg dan create_rectangle dihapus sepenuhnya.
        # Hanya menyisakan fungsi cetak teks agar melayang murni di atas kanvas.
        self.anim_canvas.create_text(25, 25, text=box_text, fill="white", font=("Consolas", 12, "bold"), anchor="nw", tags="dyn")

        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 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', "#FFE082", 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 ijtimak 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:
    mode_conv = self.radio_conv_var.get()
    d = int(self.combo_conv_day.get())
    m_str = self.combo_conv_month.get()
    y = int(self.entry_conv_year.get())
    adj = int(self.combo_conv_adj.get())

    # Fungsi bantuan deteksi kabisat yang aman untuk tahun SM (Negatif)
    def is_leap(year):
        return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

    if mode_conv == "m2h":
        m = BULAN_MASEHI.index(m_str) + 1
        mode_text = "Masehi ➔ Hijriah"

        # Format string Masehi (Bypass library date Python)
        greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
        input_text = f"{d} {m_str} {greg_year_str_input}"

        self.auto_switch_ephemeris(y)

        # Validasi jumlah hari dalam bulan agar tidak out-of-bounds
        days_in_month = [31, 29 if is_leap(y) else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
        max_days = days_in_month[m-1]
        d = min(d, max_days)

        # Eksekusi menggunakan Skyfield Time murni
        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: # Toleransi 0.1 hari
                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)

        # Hitung selisih hari dari 1 Hijriah KHGT
        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

        # Mengambil hari (Senin-Minggu) dari parameter Julian Date
        hari_idx = int(target_t_obj.whole % 7)
        hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]

        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"

    else:
        m = BULAN_HIJRIAH.index(m_str) + 1
        h_y = y
        mode_text = "Hijriah ➔ Masehi"
        hijri_year_str_input = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
        input_text = f"{d} {m_str} {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)

        # Tambahkan hari ke TT, lalu kembalikan ke UTC (Tuple: y, m, d)
        target_tt = start_tt + (d - 1) - adj
        target_t_obj = self.ts.tt_jd(target_tt)
        ty, tm, td, _, _, _ = target_t_obj.utc

        hari_idx = int(target_t_obj.whole % 7)
        hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]

        greg_year_str = f"{ty}" if ty > 0 else f"{abs(ty-1)} SM"
        hasil = f"{hari}, {td} {BULAN_MASEHI[tm - 1]} {greg_year_str}"

    report = f"""{self.get_header(85)}
```

```
{"[ Konversi Kalender KHGT ]".center(85)}

* Settings:-
- Mode Konversi : {mode_text}
- Tanggal Input : {input_text}
- Koreksi Hari  : {adj} Hari (Penyesuaian Manual)
- Ephemeris     : {self.ephemeris_name}
================================================================================

HASIL KONVERSI:
--------------------------------------------------------------------------------
>> {hasil} <<
--------------------------------------------------------------------------------

* Catatan:
- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).
- Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.
- Kalkulasi menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak).
================================================================================
"""
            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)"""
        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)

        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.")

        # ---> AWAL MODIFIKASI: PENENTUAN WAKTU EVALUASI (T_EVAL) <---
        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
        # ---> AKHIR MODIFIKASI <---

        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

    t_start_conj = self.ts.tt_jd(t_eval_tt - 5.0)
    t_end_conj = self.ts.tt_jd(t_eval_tt + 5.0)

    t_phases, y_phases = almanac.find_discrete(t_start_conj, t_end_conj,
almanac.moon_phases(self.eph))
    t_ijtima = None
    for t_ev, phase in get_safe_events(t_phases, y_phases):
        if phase == 0:
            t_ijtima = t_ev
            break

    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

    # Gunakan t_eval alih-alih t_sunset untuk semua kalkulasi posisi astronomi
    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)

    m_lat, m_lon, _ = app_moon_geo.ecliptic_latlon(epoch=t_eval)
    s_lat, s_lon, _ = app_sun_geo.ecliptic_latlon(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_earth = (earth + lokasi_obs).at(t_eval)
    app_moon_topo = topo_earth.observe(moon).apparent()
    app_sun_topo = topo_earth.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
    az_moon_topo = az_moon_topo_obj.degrees
    alt_sun_topo = alt_sun_topo_obj.degrees
    az_sun_topo = az_sun_topo_obj.degrees

    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 # Umur dihitung hingga t_eval
    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 # Lag tetap Moonset - Sunset
    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_geo = alt_moon_geo - alt_sun_geo
    rel_az_geo = az_moon_geo - az_sun_geo
    rel_alt_topo = alt_moon_topo - alt_sun_topo
    rel_az_topo = az_moon_topo - az_sun_topo

    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))
    hp_moon = math.degrees(math.asin(6378.137 / dist_km))

    crescent_width_deg = sd_moon * (1 - math.cos(math.radians(elongation)))
```

```
    phase_sme = math.degrees(math.acos(2 * (illumination / 100.0) - 1))
    magnitude = -12.73 + 0.026 * abs(phase_sme) + (4 * 10**-9) * (phase_sme**4)

    if t_ijtima is not None:
        t_ijtima_local_dt = t_ijtima.astimezone(pytz.FixedOffset(tz * 60))
        y_i, m_i, d_i = t_ijtima_local_dt.year, t_ijtima_local_dt.month, t_ijtima_local_dt.day
        h_i, mn_i = t_ijtima_local_dt.hour, t_ijtima_local_dt.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 = t_ijtima.utc_strftime('%d/%m/%Y, %H:%M UTC')
    else:
        str_ijtima = "N/A"
        str_ijtima_utc = "N/A"

    # ======================================================================
    # KALKULASI MULTI-KRITERIA VISIBILITAS INTERNASIONAL
    # ======================================================================
    is_khgt = alt_moon_geo >= 5.0 and elongation >= 8.0
    str_khgt = "TERPENUHI" if is_khgt else "Gagal"

    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)"

    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)"

    # ======================================================================

    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 = ["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}, {day:02d}/{month:02d}/{tahun_str}"

        # ---> AMBIL DATA NAMA KOTA DAN PROVINSI DARI UI <---
        nama_kota = self.opt_city.get()
        nama_prov = self.opt_prov.get()

        # 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.
- 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                : {get_hms_str(t_sunset, tz) + " LT":<16}  G. Moon Age          : {format_time_hms(moon_age_hours):<16}
- Moonset               : {moonset_str + " LT":<16}  Moon Lag Time       : {format_time_hms(lag_time_hours):<16}

- G. Moon Altitude      : {format_angle(alt_moon_geo):<16}  T. Moon Altitude      : {format_angle(alt_moon_topo):<16}
- G. Sun Altitude       : {format_angle(alt_sun_geo):<16}  T. Sun Altitude       : {format_angle(alt_sun_topo):<16}

- G. Elongation         : {format_angle(elongation):<16}  G. Phase Angle        : {format_angle(phase_angle):<16}
- G. Illumination       : {f"{illumination:05.2f} %":<16} G. Distance          : {f"{dist_km:,.2f} Km":<16}

--------------------------------------------------------------------------------
[ MULTI-CRITERIA GLOBAL RESEARCH RESULT ]
--------------------------------------------------------------------------------
1. KHGT / Diyanet Turki (G.Alt>=5°, G.Eln>=8°)   : {str_khgt}
2. Neo MABIMS          (T.Alt>=3°, G.Eln>=6.4°) : {str_mabims}
3. Ilyas Criterion     (T.Alt>=4°, G.Eln>=10.5°): {str_ilyas}
4. Danjon Limit        (G.Eln >= 7.0°)          : {str_danjon}
5. Yallop (1998) Param  (q-Value = {q_yallop:+.3f})     : {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:
        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. KONVERSI KE JULIAN DATE (Bebas dari limitasi datetime Python)
        t_start = self.ts.utc(sy, sm, sd, sh, smin, ssec)
        t_end = self.ts.utc(ey, em, ed, eh, emin, esec)

        # 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("+", "")

        # ---> 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_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]")
                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()
                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))

            t_display = self.ts.tt_jd(current_jd + (tz / 24.0))
            y_d, m_d, d_d, h_d, min_d, s_d = t_display.utc

            time_str = f"{int(h_d):02d}.{int(min_d):02d}.{int(s_d):02d}
{int(d_d):02d}/{int(m_d):02d}/{format_tahun_aman(int(y_d))}"

            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 + "\n* Calculated using JPL Ephemeris NASA.")
        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 show_qiblah_map(self):
```

```
    try:
        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)

        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

        fig, ax = plt.subplots(figsize=(12, 6))

        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}")

        # 1. Arsiran warna peta kiblat dibuat tipis (alpha=0.4)
        c = ax.contourf(LONS, LATS, Q, levels=np.arange(0, 361, 10), cmap='hsv', alpha=0.4, zorder=1)

        # 2. TAMBAHKAN INI: Garis kontur kiblat dibuat tegas dan jelas (alpha=0.8)
        ax.contour(LONS, LATS, Q, levels=np.arange(0, 361, 10), colors='black', linewidths=0.6, alpha=0.8, zorder=2)

        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()
        plt.show()

    except Exception as e:
        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("+", "")
        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/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)
                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, linestyles='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, linestyles='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, linestyles='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, linestyles='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 ephem
        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 V7.4 - 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 = ephem.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 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"

            # 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 sblm {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 = ephem.Sun()
        bulan = ephem.Moon()

        try:
            ijtimak_1 = ephem.previous_new_moon(waktu_tuple)
            ijtimak_2 = ephem.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 = 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, 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 = ephem.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 sblm {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 = 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))}"
                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.utc_datetime().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"{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):
    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="Kalkulasi Selesai", text_color="#00E676")
    self.btn_hitung.configure(state="normal")

def display_error(self, error_msg):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", f"TERJADI KESALAHAN (DEBUG):\n{error_msg}")
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
    self.btn_hitung.configure(state="normal")

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(' ', '')}@khgttimes"

    # 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",
            defaultextension=".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",
            defaultextension=".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 HIJRIAH ---
        if "Kalender Hijriah" in mode:
            img = self._generate_calendar_image()
            filepath = filedialog.asksaveasfilename(
                initialfile=f"Kalender_Hijriah_{self.cal_h_year}H.pdf",
                defaultextension=".pdf",
                filetypes=[("PDF Document", "*.pdf")]
            )
            if filepath:
                try:
                    # Langsung convert image hasil render HD ke PDF
                    img.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

        # --- 2. JIKA SEDANG DI MODUL KALENDER MASEHI ---
        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.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

        # --- 3. JIKA SEDANG DI MODUL GERHANA ---
        if "Analisis Gerhana" in mode:
            messagebox.showinfo("Info Ekspor", "Untuk Modul Gerhana, silakan gunakan fitur ekspor CSV di
tombol sebelahnya.")
            return

        # --- 4. JIKA SEDANG DI MODUL STANDAR (TEKS LAPORAN) ---
        try:
            from fpdf import FPDF
```

```
        except ImportError:
            messagebox.showerror("Library Kurang", "Silakan install library 'fpdf' (pip install fpdf) melalui terminal.")
            return

        report_data = self.textbox.get("1.0", "end-1c")
        if not report_data or "TERJADI KESALAHAN" in report_data:
            messagebox.showwarning("Peringatan", "Tidak ada data kalkulasi valid.")
            return

        filepath = filedialog.asksaveasfilename(
            initialfile="Laporan_KHGT_Times.pdf",
            defaultextension=".pdf",
            filetypes=[("PDF Documents", "*.pdf")]
        )

        if filepath:
            try:
                pdf = FPDF()
                pdf.add_page()
                pdf.set_font("Courier", size=9)

                pdf.set_font("Arial", 'B', 14)
                pdf.cell(0, 10, "LAPORAN RESMI KHGT TIMES ENGINE", ln=True, align='C')
                pdf.set_font("Arial", 'I', 10)
                pdf.cell(0, 8, f"Dicetak pada: {datetime.datetime.now().strftime('%d-%m-%Y %H:%M:%S')}", ln=True, align='C')
                pdf.line(10, 30, 200, 30)
                pdf.ln(10)

                pdf.set_font("Courier", size=8)
                for line in report_data.split('\n'):
                    # Mencegah crash FPDF saat menemukan simbol astronomi/emoji
                    clean_line = line.replace('°', ' deg').replace('➔', '->').replace('▶', '>').replace('🎵', '[Bulan]')
                    safe_line = clean_line.encode('latin-1', 'replace').decode('latin-1')
                    pdf.cell(0, 4, txt=safe_line, ln=True)

                pdf.output(filepath)
                messagebox.showinfo("Sukses", f"Laporan PDF berhasil dibuat:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error PDF", f"Gagal membuat PDF: {e}")

    def export_to_png(self):
        mode = self.combo_mode.get()

        # --- 1. JIKA SEDANG DI MODUL KALENDER HIJRIAH ---
        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 ---
    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. JIKA SEDANG DI MODUL GERHANA ---
    if "Analisis Gerhana" in mode:
        messagebox.showinfo("Info Ekspor", "Untuk Modul Analisis Gerhana, silakan gunakan tombol
'Export CSV' agar tabel bisa dibuka rapi di Excel.")
        return

    # --- 4. JIKA SEDANG DI MODUL STANDAR (TEKS LAPORAN) ---
    report_data = self.textbox.get("1.0", "end-1c")
    if not report_data or "TERJADI KESALAHAN" in report_data:
        messagebox.showwarning("Peringatan", "Tidak ada data teks kalkulasi valid yang bisa disimpan.")
        return

    filepath = filedialog.asksaveasfilename(
        initialfile="Laporan_KHGT_Times.png",
        defaultextension=".png",
        filetypes=[("PNG Image", "*.png")]
    )
```

```
    if filepath:
        try:
            try:
                font = ImageFont.truetype("cour.ttf", 14)
            except:
                font = ImageFont.load_default()

            lines = report_data.split('\n')

            # Kalkulasi Lebar dan Tinggi Kanvas Dinamis agar teks tidak terpotong
            img_temp = Image.new("RGB", (1, 1))
            draw_temp = ImageDraw.Draw(img_temp)

            max_width = 0
            for line in lines:
                if hasattr(draw_temp, 'textbbox'):
                    bbox = draw_temp.textbbox((0, 0), line, font=font)
                    w = bbox[2] - bbox[0]
                else:
                    w = draw_temp.textlength(line, font=font)
                if w > max_width: max_width = w

            width = max(800, int(max_width) + 60)
            height = max(600, len(lines) * 18 + 60)

            img = Image.new("RGB", (width, height), "#FFFFFF")
            draw = ImageDraw.Draw(img)

            y_text = 30
            for line in lines:
                draw.text((30, y_text), line, font=font, fill="#000000")
                y_text += 18

            img.save(filepath, "PNG")
            messagebox.showinfo("Sukses", f"Laporan PNG berhasil dibuat:\n{filepath}")
        except Exception as e:
            messagebox.showerror("Error PNG", f"Gagal membuat gambar PNG Laporan: {e}")

def plot_altitude_curve(self, canvas_frame, y, m, d, hours, sun_alts, moon_alts):
    try:
        for widget in canvas_frame.winfo_children():
            widget.destroy()

        from matplotlib.figure import Figure
        from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
        import numpy as np
        import datetime
        import math
```

```
        fig = Figure(figsize=(8, 5), dpi=100)
        fig.patch.set_facecolor('#181818')
        ax = fig.add_subplot(111)
        ax.set_facecolor('#101010')

        # ---> Kalkulasi Kurva Bulan Geosentrik <---
        moon_geo_alts = []
        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        tz = float(self.entry_vtz.get())
        earth = self.eph['earth']
        moon = self.eph['moon']

        for h in hours:
            t = self.ts.utc(y, m, d, h - tz)
            obs_center = earth.at(t)
            m_app = obs_center.observe(moon).apparent()
            gmst = t.gast
            lst_deg = (gmst * 15.0) + lon
            ra_m, dec_m, _ = m_app.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)
            alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
            moon_geo_alts.append(alt_geo)

        # Konversi data ke format Numpy Array agar mudah dihitung oleh sistem Hover
        x_data = np.array(hours)
        sun_alts = np.array(sun_alts)
        moon_alts = np.array(moon_alts)
        moon_geo_alts = np.array(moon_geo_alts)

        # Plot Garis Kurva
        line_sun, = ax.plot(x_data, sun_alts, color='#FFD54F', label='Matahari (Topo)', linewidth=2)
        line_moon, = ax.plot(x_data, moon_alts, color='#80DEEA', label='Bulan (Topo)', linewidth=2)
        line_moon_geo, = ax.plot(x_data, moon_geo_alts, color='#00E676', label='Bulan (Geo)',
linewidth=2, linestyle='--')

        ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')

        # --- FORMAT SUMBU X (JAM LOKAL 0-24) ---
        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)])
```

```
        # Penanda Waktu Sekarang
        target_date = datetime.date(y, m, d)
        now = datetime.datetime.now()
        if target_date == now.date():
            curr_h = now.hour + now.minute/60.0
            ax.axvline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
            batas_bawah = min(sun_alts) if len(sun_alts) > 0 else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekarang', color='#00E676', fontsize=10, rotation=90,
verticalalignment='bottom')

        # Formatting Label dan Font
        ax.set_title(f"Grafik Ketinggian Benda Langit: {target_date.strftime('%d %b %Y')}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggian / Altitude (°)", color='#ffffff', fontsize=12)
        ax.tick_params(colors='#ffffff', labelsize=11)
        ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)
        ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')

        fig.tight_layout()

        # Render SATU KALI SAJA ke dalam Tkinter Frame
        canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
        canvas_widget = canvas.get_tk_widget()
        canvas_widget.pack(fill="both", expand=True)

        # ============================================================
        # FITUR HOVER INTERAKTIF
        # ============================================================
        vline = ax.axvline(x=0, color='white', linestyle=':', alpha=0.7, linewidth=1.5, zorder=5)
        vline.set_visible(False)

        annot = ax.annotate("", xy=(0,0), xytext=(15, 15), textcoords="offset points",
                            bbox=dict(boxstyle="round4,pad=0.8", fc="#1A1A1A", ec="#FFD54F", lw=1.5,
alpha=0.95),
                            fontsize=10, color="white", weight="bold", zorder=10)
        annot.set_visible(False)

        def hover(event):
            if event.inaxes == ax:
                x_mouse = event.xdata
                y_mouse = event.ydata
                if x_mouse is None or y_mouse is None:
                    return

                # Cari index waktu (jam desimal) terdekat dari pergerakan mouse
```

```
        idx = (np.abs(x_data - x_mouse)).argmin()

        real_x = x_data[idx]
        real_y_sun = sun_alts[idx]
        real_y_moon = moon_alts[idx]
        real_y_moon_geo = moon_geo_alts[idx]

        # Ubah waktu format desimal ke format Jam:Menit (misal 14.5 jadi 14:30)
        jam = int(real_x) % 24
        menit = int(round((real_x - int(real_x)) * 60))
        if menit >= 60:
            jam = (jam + 1) % 24
            menit -= 60
        waktu_str = f"{jam:02d}:{menit:02d}"

        vline.set_xdata([real_x, real_x])
        vline.set_visible(True)

        teks = (f"⏰ Waktu: {waktu_str} LT\n"
                f"☉ Alt Mthri : {real_y_sun:.2f}°\n"
                f"🌙 Alt Bln(T): {real_y_moon:.2f}°\n"
                f"🌙 Alt Bln(G): {real_y_moon_geo:.2f}°")

        annot.xy = (real_x, y_mouse)
        annot.set_text(teks)

        # Logika pewarnaan border kotak info
        max_y = max(real_y_sun, real_y_moon, real_y_moon_geo)
        if max_y == real_y_sun:
            annot.get_bbox_patch().set_edgecolor("#FFD54F")
        elif max_y == real_y_moon:
            annot.get_bbox_patch().set_edgecolor("#80DEEA")
        else:
            annot.get_bbox_patch().set_edgecolor("#00E676")

        annot.set_visible(True)
        canvas.draw_idle()
    else:
        if vline.get_visible() or annot.get_visible():
            vline.set_visible(False)
            annot.set_visible(False)
            canvas.draw_idle()

# Hubungkan sistem hover langsung ke objek canvas yang baru saja dirender
canvas.mpl_connect("motion_notify_event", hover)
canvas.draw()
```

```
        except Exception as e:
            import traceback
            self.display_error(f"Gagal menampilkan grafik: {e}\n{traceback.format_exc()}")


    def plot_altitude_curve(self, canvas_frame, y, m, d, hours, sun_alts, moon_alts):
        try:
            # 1. Bersihkan frame layar dari grafik sebelumnya
            for widget in canvas_frame.winfo_children():
                widget.destroy()

            from matplotlib.figure import Figure
            from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
            import numpy as np
            import datetime
            import math

            fig = Figure(figsize=(8, 5), dpi=100)
            fig.patch.set_facecolor('#181818')
            ax = fig.add_subplot(111)
            ax.set_facecolor('#101010')

            # 2. Kalkulasi Kurva Bulan Geosentrik (Standar Modul 1 & 2)
            moon_geo_alts = []
            lat = float(self.entry_vlat.get())
            lon = float(self.entry_vlon.get())
            tz = float(self.entry_vtz.get())
            earth = self.eph['earth']
            moon = self.eph['moon']

            for h in hours:
                t = self.ts.utc(y, m, d, h - tz)
                obs_center = earth.at(t)
                m_app = obs_center.observe(moon).apparent()
                gmst = t.gast
                lst_deg = (gmst * 15.0) + lon
                ra_m, dec_m, _ = m_app.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)
                alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
                moon_geo_alts.append(alt_geo)

            # 3. Konversi Data ke Array Float Murni agar Detektor Mouse Tidak Error
            x_data = np.array(hours, dtype=float)
```

```
        y_sun = np.array(sun_alts, dtype=float)
        y_moon = np.array(moon_alts, dtype=float)
        y_moon_geo = np.array(moon_geo_alts, dtype=float)

        # 4. Plot Garis Kurva
        ax.plot(x_data, y_sun, color='#FFD54F', label='Matahari (Topo)', linewidth=2)
        ax.plot(x_data, y_moon, color='#80DEEA', label='Bulan (Topo)', linewidth=2)
        ax.plot(x_data, y_moon_geo, color='#00E676', label='Bulan (Geo)', linewidth=2, linestyle='--')
        ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')

        # 5. Format Sumbu X (0 - 24 Jam)
        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)])

        # 6. Penanda Garis Waktu Sekarang
        target_date = datetime.date(y, m, d)
        now = datetime.datetime.now()
        if target_date == now.date():
            curr_h = now.hour + now.minute/60.0
            ax.axvline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
            batas_bawah = min(y_sun) if len(y_sun) > 0 else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekarang', color='#00E676', fontsize=10, rotation=90,
verticalalignment='bottom')

        ax.set_title(f"Grafik Ketinggian Benda Langit: {target_date.strftime('%d %b %Y')}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggian / Altitude (°)", color='#ffffff', fontsize=12)
        ax.tick_params(colors='#ffffff', labelsize=11)
        ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)
        ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')
        fig.tight_layout()

        # ============================================================
        # KUNCI PERBAIKAN: SIMPAN CANVAS KE "self.chart_canvas"
        # ============================================================
        # Wajib menggunakan 'self.' agar event detektor mouse TIDAK DIHAPUS
        # oleh Garbage Collector Python saat grafik selesai digambar!
        self.chart_canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
        self.chart_canvas.get_tk_widget().pack(fill="both", expand=True)

        # Buat Garis Vertikal dan Tooltip Pop-up (Disimpan ke self juga)
        self.chart_vline = ax.axvline(x=0, color='white', linestyle=':', alpha=0.8, linewidth=1.5, zorder=5)
        self.chart_vline.set_visible(False)

        self.chart_annot = ax.annotate("", xy=(0,0), xytext=(15, 15), textcoords="offset points",
```

```
                bbox=dict(boxstyle="round4,pad=0.8", fc="#1A1A1A", ec="#FFD54F", lw=1.5,
alpha=0.95),
                fontsize=10, color="white", weight="bold", zorder=10)
        self.chart_annot.set_visible(False)

        def hover(event):
            # Sembunyikan garis dan teks jika mouse keluar dari kotak grafik
            if not event.inaxes:
                if self.chart_vline.get_visible() or self.chart_annot.get_visible():
                    self.chart_vline.set_visible(False)
                    self.chart_annot.set_visible(False)
                    self.chart_canvas.draw_idle()
                return

            x_mouse = event.xdata
            if x_mouse is None: return

            # Cari titik Index Jam terdekat berdasarkan posisi mouse
            idx = (np.abs(x_data - x_mouse)).argmin()
            real_x = x_data[idx]
            ry_sun = y_sun[idx]
            ry_moon = y_moon[idx]
            ry_moon_geo = y_moon_geo[idx]

            # Konversi format desimal x-axis ke jam:menit
            jam = int(real_x) % 24
            menit = int(round((real_x - int(real_x)) * 60))
            if menit >= 60:
                jam = (jam + 1) % 24
                menit -= 60
            waktu_str = f"{jam:02d}:{menit:02d}"

            # Tampilkan Garis Vertikal persis di jam tersebut
            self.chart_vline.set_xdata([real_x, real_x])
            self.chart_vline.set_visible(True)

            teks = (f" ⏰ Waktu: {waktu_str} LT\n"
                    f" ☉ Alt Mthri : {ry_sun:.2f}°\n"
                    f" 🌙 Alt Bln(T): {ry_moon:.2f}°\n"
                    f" 🌙 Alt Bln(G): {ry_moon_geo:.2f}°")

            # Posisi kotak pop-up selalu mengikuti kursor secara vertikal
            self.chart_annot.xy = (real_x, event.ydata)
            self.chart_annot.set_text(teks)

            # Warnai list pop-up bergantung objek mana yang paling tinggi
            max_y = max(ry_sun, ry_moon, ry_moon_geo)
```

```
            if max_y == ry_sun:
                self.chart_annot.get_bbox_patch().set_edgecolor("#FFD54F")
            elif max_y == ry_moon:
                self.chart_annot.get_bbox_patch().set_edgecolor("#80DEEA")
            else:
                self.chart_annot.get_bbox_patch().set_edgecolor("#00E676")

            self.chart_annot.set_visible(True)
            self.chart_canvas.draw_idle()

        # Tautkan event mouse ke canvas utama (self.chart_canvas)
        self.chart_cid = self.chart_canvas.mpl_connect("motion_notify_event", hover)
        self.chart_canvas.draw()

    except Exception as e:
        import traceback
        if hasattr(self, 'display_error'):
            self.display_error(f"Gagal menampilkan grafik:\n{e}\n{traceback.format_exc()}")
        else:
            print(f"Gagal menampilkan grafik:\n{e}")

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 = 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 Ramadhan KHGT dari Database Bawaan
        # Index 8 adalah bulan ke-9 (Ramadan)
        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 = ephem.Date(dt_rukyat)
        ijtimak = ephem.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 = 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)) # 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(ephem.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).")
    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):
    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 = 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 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 = 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 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 _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 ephem
        import pytz

        # Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
        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 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 = 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 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 (Anomali ephemeris), asumsi 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 (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 ephem
        import pytz

        # Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
        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 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 = 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 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:
   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 diistikmalkan (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 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 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 = 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 Zulkaidah (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 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 diistikmalkan.")

        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 = 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 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 diistikmalkan (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 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 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 = 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 Zulkaidah (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 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 diistikmalkan.")

        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):
    self.frame_kalender_out = ctk.CTkFrame(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))

        # PERBAIKAN: Hubungkan langsung ke fungsi utama (self.export_to_pdf)
        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)

        # PERBAIKAN: Hubungkan langsung ke fungsi utama (self.export_to_png)
        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))

    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, 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):
            # Frame Per Kotak
            cell = ctk.CTkFrame(self.grid_cal_frame, fg_color="#1E1E1E", corner_radius=8, border_width=1,
border_color="#424242")
            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)
        for i in range(7): # Alokasi hingga 6 baris + 1 header
            self.grid_cal_frame.grid_rowconfigure(i, weight=1)



    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 V7.4 - By Kasmui", font=f_sub, fill="#555555",
anchor="mm")
        return img


    # ==========================================
    # 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):
        self.frame_kalmasehi_out = ctk.CTkFrame(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))

        # PERBAIKAN: Hubungkan langsung ke fungsi utama (self.export_to_pdf)
        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)

        # PERBAIKAN: Hubungkan langsung ke fungsi utama (self.export_to_png)
        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))

    def calmasehi_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 calmasehi_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, 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):
            cell = ctk.CTkFrame(self.grid_calm_frame, fg_color="#1E1E1E", corner_radius=8, border_width=1,
border_color="#424242")
```

```
            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 (Misal: "Jumadilakhir" -> "Jum")
            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)
        for i in range(7):
            self.grid_calm_frame.grid_rowconfigure(i, weight=1)

    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 V7.4 - 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 calculate_chart_analyser(self):
        try:
            # 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())

            # Tampilkan pesan loading di GUI
            self.after(0, lambda: self.lbl_status.configure(text="Memproses Data Grafik...",
text_color="#00E5FF"))

            # 2. KALKULASI DI BACKGROUND THREAD (Bebas Tahun Minus)
            self.auto_switch_ephemeris(y)
            earth = self.eph['earth']
            sun = self.eph['sun']
            moon = self.eph['moon']
            loc = wgs84.latlon(lat, lon, elevation_m=elev)
            observer = earth + loc

            hours = np.linspace(0, 24, 144) # Resolusi per 10 menit
            sun_alts = []
            moon_alts = []

            for h in hours:
                t = self.ts.utc(y, m, d, h - tz) # Skyfield menangani tahun minus dengan aman!
                # Tambahkan parameter atmosfer agar grafik melengkung persis seperti kalkulasi Menu 1
                sun_alts.append(observer.at(t).observe(sun).apparent().altaz(temperature_C=25,
pressure_mbar=1010)[0].degrees)
                moon_alts.append(observer.at(t).observe(moon).apparent().altaz(temperature_C=25,
pressure_mbar=1010)[0].degrees)

            # 3. KIRIM HASIL KE MAIN THREAD UNTUK PLOTTING (Aman dari Thread-Lock Matplotlib)
```

```
            self.after(0, self.plot_altitude_curve, self.frame_chart_out, y, m, d, hours, sun_alts, moon_alts)

        except Exception as e:
            import traceback
            self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")

    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)

    def plot_altitude_curve(self, canvas_frame, y, m, d, hours, sun_alts, moon_alts):
        try:
            for widget in canvas_frame.winfo_children():
                widget.destroy()

            from matplotlib.figure import Figure
            from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

            fig = Figure(figsize=(8, 5), dpi=100)
            fig.patch.set_facecolor('#181818')
            ax = fig.add_subplot(111)
            ax.set_facecolor('#101010')

            # ---> TAMBAHAN: Kalkulasi Kurva Bulan Geosentrik (Standar Menu 2) <---
            moon_geo_alts = []
            lat = float(self.entry_vlat.get())
            lon = float(self.entry_vlon.get())
            tz = float(self.entry_vtz.get())
            earth = self.eph['earth']
            moon = self.eph['moon']

            for h in hours:
                t = self.ts.utc(y, m, d, h - tz)
                obs_center = earth.at(t)
                m_app = obs_center.observe(moon).apparent()

                gmst = t.gast
                lst_deg = (gmst * 15.0) + lon
                ra_m, dec_m, _ = m_app.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)
            alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
            moon_geo_alts.append(alt_geo)
        # ---------------------------------------------------------------------

        # 3. Plot Data
        ax.plot(hours, sun_alts, color='#FFD54F', label='Matahari (Topo)', linewidth=2)
        ax.plot(hours, moon_alts, color='#80DEEA', label='Bulan (Topo MABIMS)', linewidth=2)
        ax.plot(hours, moon_geo_alts, color='#00E676', label='Bulan (Geo KHGT)', linewidth=2, linestyle='-
-') # Garis Standar Menu 2
        ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')

        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)
            batas_bawah = min(sun_alts) if sun_alts else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekarang', color='#00E676', fontsize=10, rotation=90,
verticalalignment='bottom')

        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)

        ax.set_title(f"Grafik Ketinggian Benda Langit: {d:02d} {bulan_str} {tahun_str}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggian / Altitude (°)", color='#ffffff', fontsize=12)

        ax.tick_params(colors='#ffffff', labelsize=11)
        ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)
        ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')
        fig.tight_layout()

        canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
        canvas.draw()
        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()}")

    # ========================================
    # MODUL 28: WINAI (AI CHATBOT ASSISTANT)
    # ========================================
    def setup_winai_ui(self):
        # --- VARIABEL & API KEY WINAI (DIRECT GEMINI API) ---
        self.winai_api_keys = [
            '.....',
            '.....'

        ]
        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 V7.4.\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 <---
        # AI hanya menghitung jika ada perintah hitung secara eksplisit
        is_tanya_hilal = any(kata in teks_lower for kata in ["hitung", "kalkulasi", "berapa", "cek"]) and ("visibilitas" in teks_lower or "hilal" in teks_lower)
        is_tanya_kota = any(kata in teks_lower for kata in ["kota pertama", "titik pertama", "awal bulan khgt", "kapan 1", "kapan awal"])

        if is_tanya_hilal:
            self.sisipkan_teks_winai("Menjalankan Modul 1 (Kalkulasi Visibilitas) & Menganalisis Jawaban...", pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            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) & Menganalisis Jawaban...", pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "kota"), daemon=True).start()
        else:
            self.sisipkan_teks_winai("Mengakses Database & Menghubungi AI...", pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            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": {"parts": [{"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}:generateContent?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 Anda:\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```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 panjang/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())
            bln_str = 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())
        bln_str = 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_obscuration(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_obscuration(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_obscuration(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_obscuration(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(lokasi_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 Dzuhur : {bayangan_dzuhur:.2f} cm (Bayangan Terpendek)")
        output_lines.append(f"* Target Ashr: {target_shadow_asr:.2f} cm (Tongkat + Byg Dzuhur)")
        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.winfo_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 akan otomatis pindah file (misal ke de442.bsp atau de406.bsp)
                # saat tahun melewati batas 2053, 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, markeredgecolor='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"{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."

        self.after(0, self.display_result, report)

    except Exception as e:
        import traceback
        self.after(0,                self.display_error,                f"Gagal                menghitung
Analemma:\n{str(e)}\n\n{traceback.format_exc()}")


if __name__ == "__main__":
    app = KHGTApp()
    app.mainloop()
```

LAMPIRAN 4 REQUIREMENT LIBRARY

Berikut adalah dokumen persyaratan (*requirement*) *library* untuk menjalankan kode program `Python KHGT Times`, beserta penjelasan rincinya dalam bahasa Indonesia.

1. Daftar Requirement (requirements.txt)

Anda dapat menyalin daftar berikut ke dalam file bernama `requirements.txt` dan menginstalnya menggunakan perintah `pip install -r requirements.txt`.

Plaintext

```
numpy
pytz
ephem
duckduckgo_search
beautifulsoup4
requests
customtkinter
Pillow
CTkToolTip
skyfield
matplotlib
scipy
win10toast
pygame
```

2. Penjelasan Rinci Fungsi Masing-Masing Library

A. Library Astronomi & Sains (Inti Kalkulasi)

- **`skyfield`**: Digunakan sebagai mesin utama untuk perhitungan posisi benda langit (Matahari, Bulan, Planet) dengan akurasi tinggi menggunakan data ephemeris NASA. Ini digunakan untuk menentukan waktu konjungsi, gerhana, dan fase bulan.
- **`ephem` (PyEphem)**: Digunakan untuk komputasi astronomi tingkat lanjut lainnya, seperti kalkulasi *observer* untuk menentukan visibilitas hilal di lokasi tertentu.
- **`numpy`**: Digunakan untuk pengolahan data numerik dan array besar, terutama saat memproses koordinat peta visibilitas hilal dan titik-titik grafik.
- **`pytz`**: Mengatur zonasi waktu (timezone) dunia agar konversi dari waktu UTC (standar astronomi) ke waktu lokal menjadi akurat.
- **`scipy` (khususnya `scipy.interpolate`)**: Digunakan untuk melakukan interpolasi data guna menghasilkan peta visibilitas HD yang halus.

B. Library Antarmuka Grafis (GUI)

- **`customtkinter`**: Library utama untuk membangun tampilan aplikasi. Ini memberikan tampilan modern ("Dark Mode") dan kontrol *widget* yang lebih baik dibanding Tkinter standar.

- **`tkinter`**: Dasar dari GUI Python. Digunakan untuk kotak pesan (*messagebox*), pemilihan file, dan struktur dasar jendela aplikasi.
- **`CTkToolTip`**: Memberikan fungsi *tooltip* (pesan bantuan kecil saat kursor diarahkan ke tombol) untuk meningkatkan pengalaman pengguna.
- **`matplotlib`**: Digunakan untuk menampilkan grafik 2D dan simulasi 3D lintasan matahari/bulan (Analemma) serta grafik ketinggian hilal langsung di dalam aplikasi.

C. Library Pengolahan Gambar & Media

- **`Pillow` (PIL)**: Digunakan untuk memanipulasi gambar, membuat jadwal shalat dalam format gambar (PNG/JPG), dan menuliskan teks pada *template* gambar menggunakan font khusus.
- **`pygame`**: Digunakan secara khusus untuk memutar file audio (seperti suara Adzan) pada fitur alarm jadwal shalat.
- **`win10toast`**: Mengirimkan notifikasi *pop-up* pada sistem operasi Windows saat waktu shalat tiba.

D. Library Web & Data

- **`requests`**: Digunakan untuk mengambil data dari internet, seperti pengecekan status cuaca *live* atau pengambilan data dari server WinAI.
- **`beautifulsoup4` (bs4)**: Digunakan untuk membedah (*parsing*) data HTML jika aplikasi perlu mengambil informasi dari situs web tertentu.
- **`duckduckgo_search`**: Integrasi pencarian untuk fitur WinAI agar aplikasi dapat mencari referensi atau informasi secara daring.
- **`json`**: Digunakan untuk membaca dan menyimpan data database Hijriah (`db_hijriah.json`) serta konfigurasi aplikasi.

3. File Dependensi Tambahan (Wajib Ada)

Selain *library* di atas, kode ini membutuhkan file fisik di folder yang sama agar dapat berfungsi sempurna:

1. **`de441-new.bsp`**: File Ephemeris NASA. Tanpa file ini, sistem tidak bisa menghitung posisi bulan dan matahari.
2. **`arial.ttf` & `arialbd.ttf`**: File font untuk pembuatan gambar jadwal shalat.
3. **`adzan.mp3`**: File audio yang akan diputar oleh library `pygame` saat alarm aktif.

Catatan Teknis: Jika Anda menggunakan Linux, Anda mungkin perlu menginstal paket tambahan seperti `python3-tk` melalui *package manager* sistem (misal: `sudo apt install python3-tk`) karena Tkinter tidak selalu disertakan dalam distribusi Linux standar. Di Windows, Tkinter biasanya otomatis terinstal bersama Python.

PENULIS

KASMUI

- Dosen Kimia, Komputasi, IT, dan AI UNNES, serta Praktisi Ilmu Falak;
- Anggota Majelis Tabligh PDM Kota Semarang dan PWM Jawa Tengah;
- Anggota Tim Pengembang Software KHGT MTT PP Muhammadiyah;
- Website pribadi: https://hisabmu.com/, https://kasmui.cloud/;
- Minat & Hobi: Computer programming.

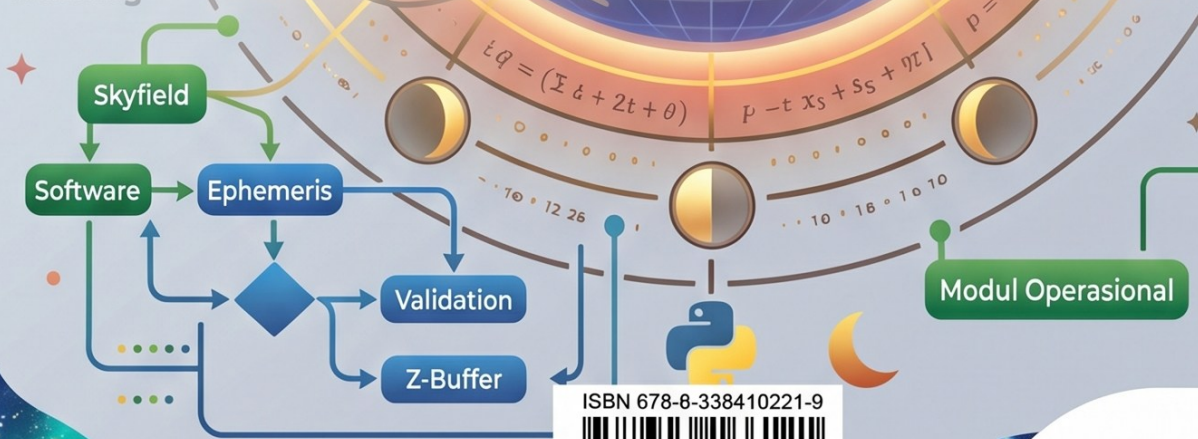
KONSEP, FORMULASI, DAN ARSITEKTUR KODE KHGT TIMES

Sebuah panduan komprehensif yang menjembatani ilmu falak klassifit dan teknologi perangkat lunak modern. Buku ini menguraikan konsep dasar perhitungan waktu shalat dan penentuan hilal global menggunakan metode Geosentris.

Penulis, **KASMUI**, menyajikan formulasi matematis yang mendalam—seperti persamaan *Great Circle* dan *Equation of Time*—serta arsitektur kode sumber kide sumber. Python yang digunakan dalam aplikasi KHGT Times.

Melalui modul-modul praktis seperti 'Crescent Tracking', 'Analisis Gerhana', dan 'Simulasi 3D', pembaca akan mehahami bagaimana data presisi dari Skyfildi Skyfield Ephemeris diolah menjadi informasi visibilitas hilal dan database JSON operasional.

Dilengkapi dengan diagram Z-Buffer dan algoritma validasi kode, buku ini adalah essensial bagi akademisi, peneliti, dan pengembang sistem falak digital.



ISBN 678-8-338410221-9
9 782392 166121

KASMUI