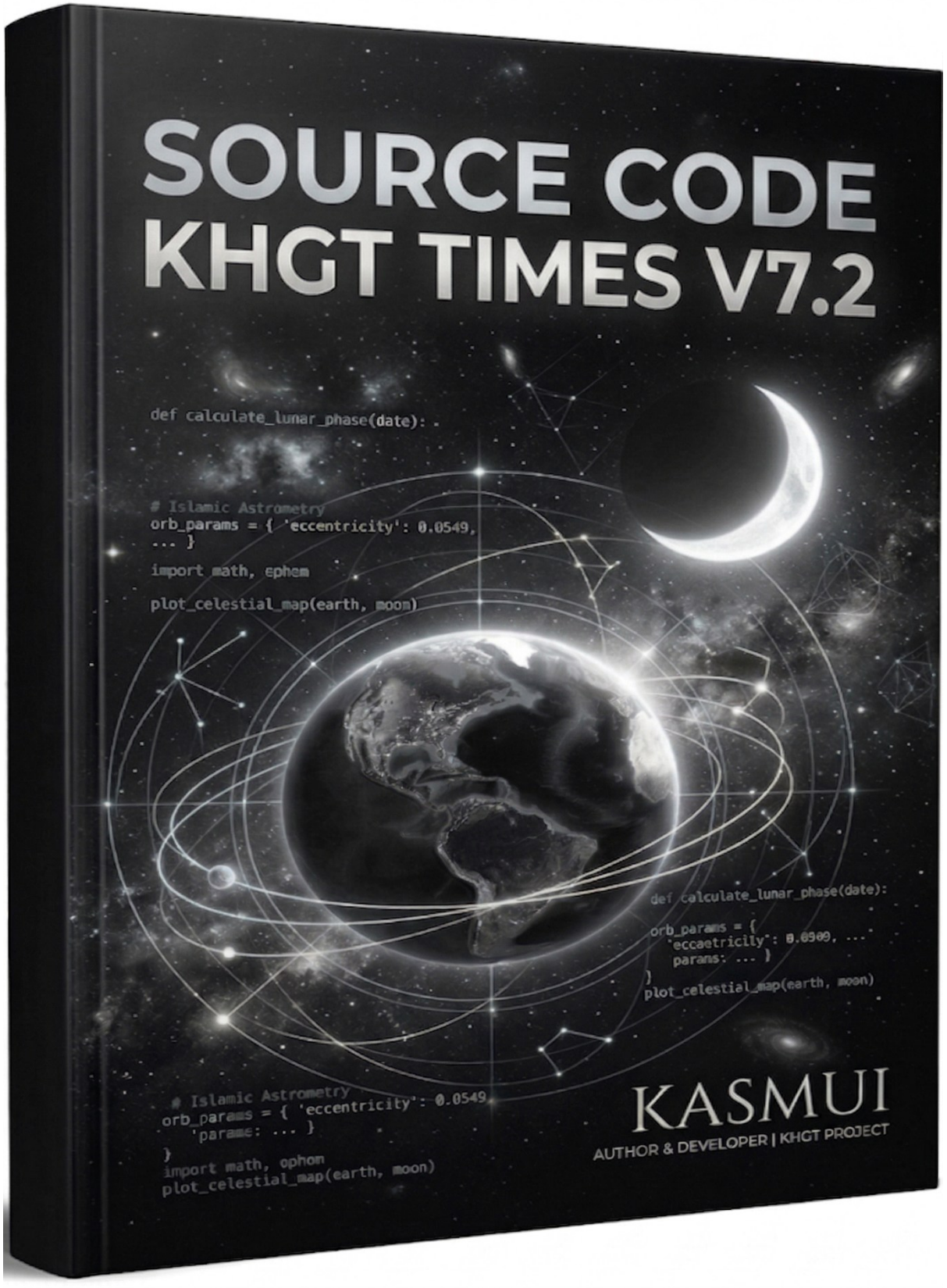
SOURCE CODE KHGT TIMES V7.2

KASMUI

AUTHOR & DEVELOPER | KHGT PROJECT


SOURCE CODE
KHGT TIMES V7.2
def calculate_lunar_phase(date):
# Islamic Astrometry
orb_params = { 'eccentricity': 0.0549,
... }
import math, ephem
plot_celestial_map(earth, moon)
def calculate_lunar_phase(date):
orb_params = {
'eccaetricily': 0.0909, ...
params: ... }
}
plot_celestial_map(earth, moon)
# Islamic Astrometry
orb_params = { 'eccentricity': 0.0549,
'params: ... }
}
import math, ephem
plot_celestial_map(earth, moon)
KASMUI
AUTHOR & DEVELOPER | KHGT PROJECT

KATA PENGANTAR

Bismillaahirrahmanirrahim.

Diskursus mengenai unifikasi kalender Islam internasional telah mencapai momentum historisnya melalui gagasan Kalender Hijriah Global Tunggal (KHGT). Implementasi dari konsep *ittihad al-matali'* (kesatuan matlak) global ini menuntut hadirnya infrastruktur teknologi dan perangkat komputasi astronomi (ilmu falak) yang melampaui standar konvensional. Di sinilah urgensi perumusan ulang algoritma hisab menjadi sangat krusial, di mana presisi matematis harus mampu berjalan presisi dan selaras dengan parameter syariat secara *real-time*.

Selama beberapa dekade terakhir, observatorium dan lembaga hisab di dunia Islam banyak bergantung pada perangkat lunak generasi awal yang dibangun di atas teori analitik klasik (seperti formulasi Jean Meeus atau VSOP87). Perangkat-perangkat tersebut, meski sangat berjasa pada masanya, memiliki keterbatasan (*gap* teknologi) untuk menjawab tantangan modern. Mayoritas beroperasi menggunakan arsitektur *single-thread* yang lambat untuk pemrosesan massal, resolusi pemetaan spasial yang rendah, dan secara konseptual dirancang murni untuk mengakomodasi kriteria rukyat lokal/zonal. Terdapat kekosongan sistem yang secara khusus, proaktif, dan iteratif mampu mengakomodasi prasyarat KHGT—yakni pemisahan tegas antara elevasi Toposentris dan Toposentris, serta algoritma pemindai Parameter Kriteria Global (PKG) 1 dan 2 yang memverifikasi sinkronisasi waktu terhadap fajar di Gisborne, Selandia Baru.

Buku **"Source Code KHGT Times V7.2"** ini lahir untuk menjembatani jurang pemisah tersebut, sekaligus menawarkan kebaruan ide (*novelty*) berupa *quantum leap* (lompatan besar) dalam rekayasa falak digital. Buku ini membedah arsitektur kode sumber dari KHGT Times V7.2 sebagai sistem komputasi berbasis Python yang mengeliminasi ketergantungan pada rumus hampiran matematika klasik dengan mengintegrasikan *Development Ephemeris* (DE) standar NASA Jet Propulsion Laboratory (JPL) secara langsung.

Di dalam karya ini, pembaca akan dibawa menjelajahi 21 modul komputasi tingkat lanjut. Barisan kode yang diuraikan mendemonstrasikan bagaimana sistem ini menelusuri rentang waktu 50 tahun komparasi dalam hitungan detik, mengeksekusi interpolasi spasial *bicubic* melalui library `SciPy` untuk menghasilkan peta visibilitas (*crescent heatmap*) beresolusi *Ultra* HD, hingga merender ruang tata surya secara 3D. Terobosan yang tak kalah penting dalam buku ini adalah pengenalan arsitektur WinAI, sebuah lompatan integrasi di mana model Kecerdasan Buatan (AI) diskalakan untuk membaca data astrometri internal guna berfungsi sebagai asisten fikih yang interaktif dan solutif.

Karya ini disusun bukan sekadar sebagai dokumentasi teknis *engineering*, melainkan sebagai manifesto akademik yang otoritatif untuk mengawal implementasi KHGT. Penulis berharap buku ini dapat menjadi rujukan fundamental bagi para ahli falak, pemikir, astronom, dan pengambil kebijakan di berbagai lembaga ketarjihan pada era modern ini.

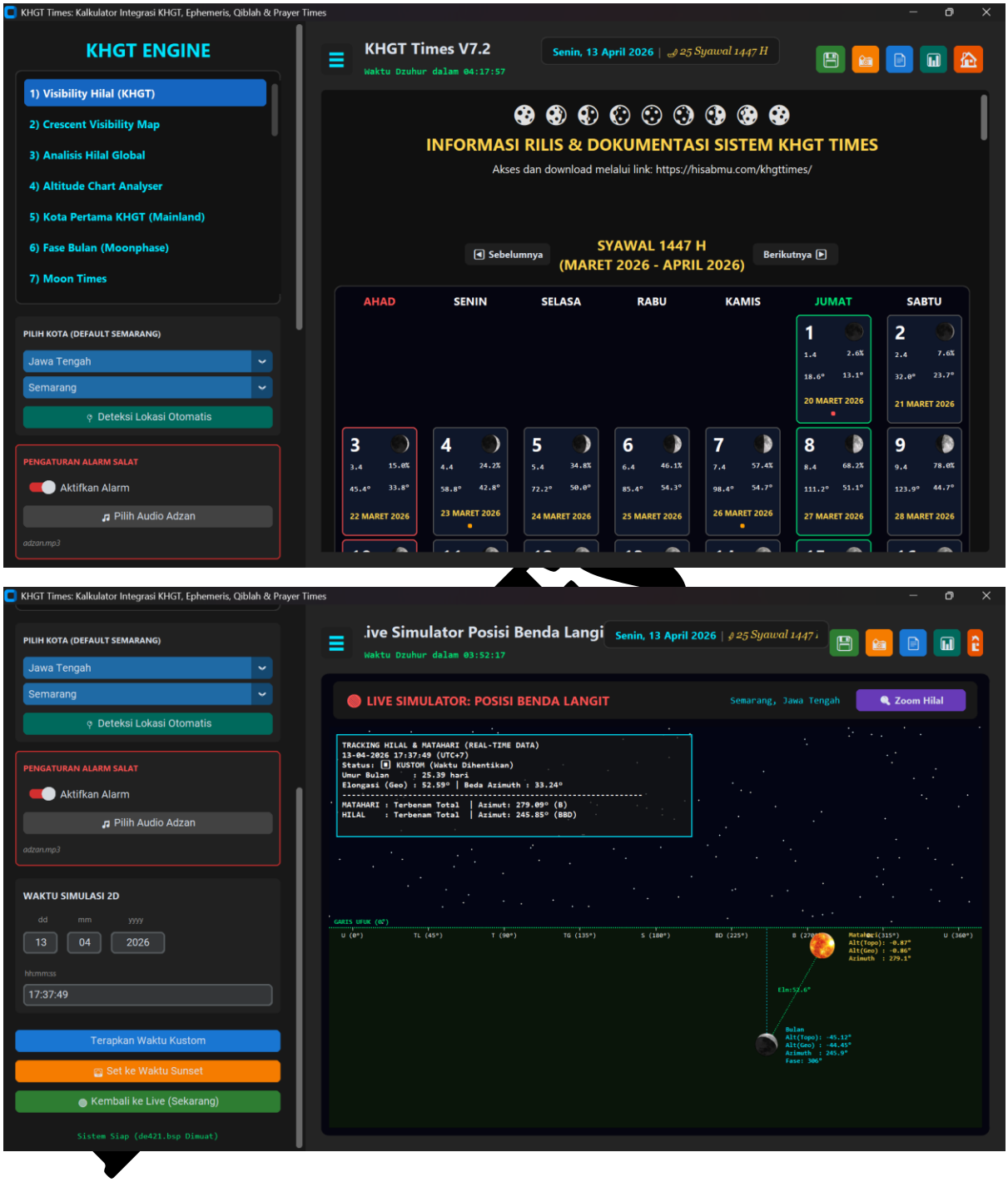
Akhir kata, hanya kepada Allah SWT senantiasa melimpahkan taufik-Nya, dan menjadikan karya ini sebagai bagian dari amal jariah yang memperkokoh peradaban keilmuan Islam global.

Semarang, Maret 2026

Kasmui
Penulis & Pengembang KHGT Times

DAFTAR ISI

FRONTPAGE KHGT TIMES V7.2



TAMPILAN KALENDER & DATA LENGKAP

Kalender Hijriah KHGT: SYAWAL 1447 H

Maret 2026 - April 2026

Ahad	Senin	Selasa	Rabu	Kamis	Jumat	Sabtu
					1 1.4 2.6% 18.6° 13.1° 20 MAR 2026	2 2.4 7.6% 32.0° 23.7° 21 MAR 2026
3 3.4 15.0% 45.4° 33.8° 22 MAR 2026	4 4.4 24.2% 58.8° 42.8° 23 MAR 2026	5 5.4 34.8% 72.2° 50.0° 24 MAR 2026	6 6.4 46.1% 85.4° 54.3° 25 MAR 2026	7 7.4 57.4% 98.4° 54.7° 26 MAR 2026	8 8.4 68.2% 111.2° 51.1° 27 MAR 2026	9 9.4 78.0% 123.9° 44.7° 28 MAR 2026
10 10.4 86.2% 136.4° 36.6° 29 MAR 2026	11 11.4 92.7% 148.6° 27.6° 30 MAR 2026	12 12.4 97.2% 160.6° 18.1° 31 MAR 2026	13 13.4 99.5% 172.1° 8.5° 1 APR 2026	14 14.4 99.8% 174.8° -1.1° 2 APR 2026	15 15.4 98.1% 164.0° -10.5° 3 APR 2026	16 16.4 94.5% 152.8° -19.7° 4 APR 2026
17 17.4 89.3% 141.8° -28.5° 5 APR 2026	18 18.4 82.8% 130.9° -36.8° 6 APR 2026	19 19.4 75.1% 120.1° -44.3° 7 APR 2026	20 20.4 66.6% 109.2° -50.7° 8 APR 2026	21 21.4 57.4% 98.3° -55.3° 9 APR 2026	22 22.4 47.8% 87.3° -57.2° 10 APR 2026	23 23.4 38.1% 76.1° -55.9° 11 APR 2026
24 24.4 28.6% 64.5° -51.4° 12 APR 2026	25 25.4 19.7% 52.6° -44.5° 13 APR 2026	26 26.4 11.9% 40.3° -35.7° 14 APR 2026	27 27.4 5.7% 27.5° -25.7° 15 APR 2026	28 28.4 1.6% 14.6° -14.8° 16 APR 2026	29 29.4 0.1% 3.9° -3.3° 17 APR 2026	

LEGENDA WARNA:

- Hari Haram Puasa / Hari Raya
- Puasa Sunnah Utama
- Puasa Ayyamul Bidh
- Puasa Senin Kamis
- Ramadhan
- Hari Besar Islam

* Kiri: Umur Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas), Tinggi Hilal Geo (Bawah). Dihitung saat Maghrib.

Kota: Semarang, Jawa Tengah | Koordinat: (Lat: -7.0667, Lon: 110.41)

KHGT Times Engine V7.2 - By Kasmui

INFORMASI RILIS & DOKUMENTASI SISTEM: KHGT TIMES V7.2

Perangkat Lunak Astrometri Presisi Tinggi & Penyatuan Kalender Islam Global

Informasi	Detail
Versi Aplikasi	7.2 (Stable Release)
Pengembang	Kasmui
Tujuan	Kalkulasi Astronomi Islam, Pemetaan Hilal Global, & Implementasi KHGT
Lisensi	Copyleft (c) 2026 - Untuk Kemajuan Sains Islam

1. PENDAHULUAN (Apa itu KHGT Times?)

KHGT Times adalah aplikasi desktop berbasis Python yang dirancang untuk melakukan perhitungan astrometri benda langit dengan tingkat presisi milidetik. Berbeda dengan aplikasi falak konvensional, sistem ini dibangun di atas mesin **JPL Ephemeris NASA** untuk mendukung implementasi **Kalender Hijriah Global Tunggal (KHGT)**.

Sistem ini mengintegrasikan parameter **Geosentrik** (Pusat Bumi) untuk validasi kalender internasional dan **Toposentrik** (Permukaan Bumi) untuk kebutuhan lokal seperti waktu salat dan arah kiblat.

2. PERSYARATAN SISTEM (Library Requirements)

Untuk menjalankan kode program ini di lingkungan Python Anda harus memenuhi dependensi berikut:

Library	Kegunaan Utama
`skyfield`	Engine utama kalkulasi posisi benda langit standar NASA/USNO.
`ephem`	Akselerator pencarian iteratif (Sunset, Ijtimak, Gerhana).
`numpy`	Manipulasi matriks data dan array komputasi 50 tahun.
`scipy`	Interpolasi Bicubic untuk perenderan Peta Hilal HD.
`customtkinter`	Framework antarmuka pengguna (GUI) modern & dark mode.
`matplotlib`	Visualisasi grafik 2D, 3D, dan peta proyeksi dunia.
`Pillow (PIL)`	Engine grafis untuk ekspor jadwal shalat dan kalender ke gambar.
`requests & bs4`	Integrasi WinAI untuk pencarian data internet real-time.
`pygame`	Sistem pemutar audio Adzan otomatis.

3. DAFTAR FITUR & MODUL (31 Menu Utama)

Aplikasi ini dibagi menjadi 31 modul fungsional yang dapat diakses melalui Sidebar interaktif:

1. **Visibility Hilal (KHGT):** Analisis parameter hilal (Alt/Eln) saat Maghrib di lokasi spesifik.
2. **Crescent Visibility Map:** Render peta dunia HD yang menunjukkan area visibilitas hilal.
3. **Analisis Hilal Global:** Tabel pemindaian kondisi hilal di ratusan kota besar dunia.
4. **Altitude Chart Analyser:** Grafik kurva pergerakan harian Matahari dan Bulan.

5. **Kota Pertama KHGT:** Pelacak otomatis titik daratan pertama yang memenuhi syarat bulan baru.
6. **Fase Bulan (Moonphase):** Kalender siklus fase bulan (Ijtimak hingga Bulan Mati).
7. **Moon Times:** Jadwal terbit, transit, dan terbenam bulan selama sebulan.
8. **Sun Times:** Jadwal terbit, transit (zawal), dan terbenam matahari.
9. **Sun Moon Ephemeris:** Ekstraktor data mentah (R.A, Deklinasi, Jarak) untuk riset.
10. **Qibla Time (Rashdul Lokal):** Mencari jam berapa bayangan benda mengarah ke Ka'bah.
11. **Qiblah Direction & Times:** Kalkulator sudut kiblat dengan visualisasi peta bola bumi.
12. **Prayer Times:** Jadwal salat 5 waktu dengan berbagai metode (Kemenag, Muhammadiyah, dll).
13. **Konversi Kalender:** Pengubah tanggal Masehi ke Hijriah KHGT sec[illegible] akurat.
14. **Analisis Gerhana:** Scanner gerhana Matahari & Bulan (mendukung tahun [illegible]M).
15. **Live Animasi:** Simulator 2D posisi benda langit di ufuk seca[illegible]eal-time.
16. **Sistem Sun Moon Earth:** Visualisasi 3D interaktif tata su[illegible] be[illegible]asis ru[illegible].
17. **Equinox & Solstice:** Penentu titik balik musim dan per[illegible]tian ta[illegible] astronomis.
18. **Planetary Times:** Jadwal terbit dan terbenam untuk [illegible]lanet tata s[illegible].
19. **Simulasi Ephemeris 3D:** Plotting koordinat benda [illegible] menggunakan proyeksi Matplotlib.
20. **Komparasi 50 Thn (Ramadhan):** Analis[illegible] potensi perbedaan awal puasa antara KHGT vs MABIMS.
21. **Komparasi 50 Thn (Syawal):** Analis[illegible]tensi perbe[illegible]an Idul Fitri hingga 50 tahun ke depan.
22. **Tabel Tinggi Hilal (Ramadhan):** [illegible]rak[illegible]ta angka ketinggian hilal selama 50 tahun.
23. **Tabel Elongasi Hilal (Ra[illegible]han):** [illegible]strak[illegible] data sudut elongasi selama 50 tahun.
24. **Tabel Tinggi Hilal (Sy[illegible]wal):** [illegible]a te[illegible]nis ketinggian hilal menjelang Idul Fitri.
25. **Tabel Elongasi Hila[illegible]wal):** [illegible] teknis elongasi hilal menjelang Idul Fitri.
26. **Kalender Hijriah [illegible]erjal[illegible]:** Kalen[illegible]er Islam interaktif dengan penanda puasa sunnah.
27. **Kalender Mas[illegible]Berjalan[illegible]:** Kalender sipil yang terintegrasi dengan penanggalan KHGT.
28. **WinAI:** A[illegible]sisten Kece[illegible]an Buatan (Gemini) untuk konsultasi fikih dan astronomi.
29. **Kalkulat[illegible]Mi[illegible]la:** Simulasi panjang bayangan tongkat istiwa untuk waktu salat.
30. **HIJ[illegible]DB[illegible]to[illegible]lder:** Tool admin untuk membuat ulang basis data kalender r[illegible]an[illegible]un.
31. **Sta[illegible]Ki[illegible]ia Batas:** Simulator interaktif untuk memahami ambang batas 5° Alt dan 8° El[illegible].

4. TU[illegible]AL OPERASIONAL (User Manual)

A. Pengaturan Lokasi

1. Buka Sidebar dengan menekan tombol ☰.
2. Gunakan dropdown **Provinsi** dan **Kota** untuk memilih lokasi (Default: Semarang).
3. Atau, tekan tombol ⚲ **Deteksi Lokasi Otomatis** untuk menggunakan koordinat GPS asli Anda.
4. Aplikasi akan otomatis memperbarui Timezone dan Elevasi.

B. Menghitung Visibilitas Hilal (Awal Bulan)

1. Pilih menu **"1) Visibility Hilal (KHGT)"**.
2. Masukkan tanggal rencana rukyat (misal: 29 Ramadhan).
3. Klik tombol ▶ **PROSES DATA**.
4. Laporan akan muncul di layar utama mencakup nilai **Geosentrik** dan status kelulusan kriteria internasional.

C. Menggunakan WinAI (Asisten Cerdas)

1. Pilih menu **"28) WinAI"**.
2. Ketik pertanyaan Anda secara natural, misalnya: *"Kapan awal Ramadhan 2027 menurut KHGT?"*.
3. AI akan secara otomatis memanggil modul kalkulasi internal, menghitung data astronomisnya, dan memberikan jawaban beserta kesimpulan fikih.

D. Ekspor Hasil

- **Format Teks:** Klik ikon 💾 untuk menyimpan laporan ke file `.txt`.
- **Format Gambar:** Klik ikon 📷 untuk menyimpan visual (Jadwal Salat/Kalender) ke `.png`.
- **Format Dokumen:** Klik ikon 📄 untuk mencetak laporan resmi ke `.pdf`.
- **Format Spreadsheet:** Klik ikon 📊 untuk mengekspor data komparasi 50 tahun ke `.csv` (Excel).

5. CATATAN TEKNIS & REFERENSI (Scientific Reference)

Aplikasi ini menggunakan standar **Muktamar Turki 2016** untuk KHGT:

1. **Ketinggian Hilal (Geosentrik):** Minimal 5° saat matahari terbenam.
2. **Elongasi (Geosentrik):** Minimal 8° saat matahari terbenam.
3. **Transfer Visibilitas:** Jika satu titik di daratan bumi memenuhi syarat sebelum pukul 00:00 UTC, maka seluruh dunia memulai bulan baru di hari yang sama.

KHGT Times Engine V7.2 - *Bringing Precision to Islamic Civilization.*

SOURCE CODE 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_PHP_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, NavigationToolbar2
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: ARAH KIBLAT & JADWAL SHOLAT ---
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 SHOLAT", 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 Wage", "16-Jul-2026", 30],
["Rabiulawal", "Kamis", "13-Aug-2026", 30], ["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", 29], ["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 - 1507 secara otomatis, 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", "02-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", 29], ["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", 30], ["Safar", "Senin", "09-Apr-2035", 30], ["Rabiulawal", "Rabu", "09-May-2035", 29], ["Rabiulakhir", "Kamis", "07-Jun-2035", 29], ["Jumadilawal", "Jumat", "06-Jul-2035", 30], ["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", 29], ["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", "Kamis", "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", 30], ["Jumadilawal", "Rabu", "01-May-2041", 30], ["Jumadilakhir", "Jumat", "31-May-2041", 30], ["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", 30], ["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", "16-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", 29]],

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", 30], ["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", "Ahad", "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", "2[illegible]-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", "[illegible]-Jul-2050", 30], ["Zulhijah", "Jumat", "19-Aug-2050", 29]],

1473: [["Muharam", "Sabtu", "17-Sep-2050", 30], ["Safar", "Senin", "[illegible]ct-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", "[illegible]-Apr-2051", 29], ["Ramadan", "Kamis", "11-May-2051", 30], ["Syawal", "Sabtu", "10-Jun-2051", "Zulkaidah", "Ahad", "09-Jul-2051", 30], ["Zulhijah", "Selasa", "08-Aug-2051", 29]],

1474: [["Muharam", "Rabu", "06-Sep-2051", "Safar", "Jumat", "06-Oct-2051", 30], ["Rabiulawal", "Ahad", "05-Nov-2051", 29], ["Rabiulakhir", "Senin", "04-Dec-2051", 30], ["Jumadilawal", "Rabu", "03-Jan-2052", "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", "[illegible]-May-2052", 30], ["Zulkaidah", "Jumat", "28-Jun-2052", 29], ["Zulhijah", "Sabtu", "27-Jul-2052", 30]],

1475: [["Muharam", "Selasa", "26-Aug-2052", 29], ["Safar", "Selasa", "24-Sep-2052", 30], ["Rabiulawal", "Kamis", "24-Oct-2052", 29], ["Rabiulakhir", "Jumat", "22-Nov-2052", 30], ["Jumadilawal", "Ahad", "23-Dec-2052", 30], ["Jumadilakhir", "Selasa", "21-Jan-2053", 29], ["Rajab", "Kamis", "20-Feb-2053", "Syakban", "Jumat", "21-Mar-2053", 30], ["Ramadan", "Ahad", "20-Apr-2053", "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", "[illegible]-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", [illegible]], ["Jumadilawal", "Senin", "29-Oct-2057", 30], ["Jumadilakhir", "Rabu", "28-Nov-205[illegible]", 2[illegible], "Rajab", "Kamis", "27-Dec-2057", 29], ["Syakban", "Jumat", "25-Jan-2058", 30], ["Ramadan", "Ahad", "[illegible]-Feb-2058", 29], ["Syawal", "Senin", "25-Mar-2058", 30], ["Zulkaidah", "Rabu", "[illegible]-Apr-2058", [illegible]], ["Zulhijah", "Kamis", "23-May-2058", 30]],

1481: [["Muharam", "Sabtu", "22-Jun-2058", 29], ["Safar", "Ahad", "[illegible]-Jul-2058", 30], ["Rabiulawal", "Selasa", "20-Aug-2058", 30], ["Rabiulakhir", "Kamis", "[illegible]-Sep-2058", 30], ["Jumadilawal", "Sabtu", "19-Oct-2058", 29], ["Jumadilakhir", "Ahad", "[illegible]-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", "Kamis", "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", "[illegible]-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", "[illegible]-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", [illegible]], ["Syakban", "Jumat", "24-Dec-2060", 30], ["Ramadan", "Ahad", "23-Jan-2061", 29], ["Syawal", "Senin", "[illegible]-Feb-2061", 30], ["Zulkaidah", "Rabu", "23-Mar-2061", 29], ["Zulhijah", "Kamis", "21-Apr-2061", 30]],

1484: [["Muharam", "Sabtu", "21-May-2061", 29], ["Safar", "Sabtu", "18-Jun-2061", 30], ["Rabiulawal", "Senin", "18-Jul-2061", 29], ["Rabiulakhir", "Selasa", "16-Aug-2061", 30], ["Jumadilawal", "Kamis", "[illegible]-Sep-2061", 30], ["Jumadilakhir", "Sabtu", "15-Oct-2061", 29], ["Rajab", "Ahad", "13-[illegible]-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-Ja[illegible]66", 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-Ju[illegible]2066", 30], ["Jumadilawal", "Sabtu", "24-Jul-2066", 29], ["Jumadilakhir", "Ahad", "22[illegible]g2066", 30], [illegible]ajab", "Selasa", "21-Sep-2066", 29], ["Syakban", "Rabu", "20-Oct-2066", 30], [illegible]amadan", "Ju[illegible]t", "19-Nov-2066", 29], ["Syawal", "Sabtu", "18-Dec-2066", 29], ["Zulkaidah", "Ahad", "[illegible]an-2067", 30], ["Zulhijah", "Selasa", "15-Feb-2067", 29]],

1490: [["Muharam", "Rabu", "16-Mar-2067", 30], ["Safar", "Jumat", "[illegible]pr-2067", 29], ["Rabiulawal", "Sabtu", "14-May-2067", 30], ["Rabiulakhir", "Senin", "13-Ju[illegible]2067", 30], ["Jumadilawal", "Rabu", "13-Jul-2067", 29], ["Jumadilakhir", "Kamis", "11-Aug-2067", 30], ["Rajab", "Sabtu", "10-Sep-2067", 30], ["Syakban", "Ahad", "[illegible]-Oct-20[illegible]", 29], ["Ramadan", "Selasa", "08-Nov-2067", 30], ["Syawal", "Kamis", "08-Dec-206[illegible]", "Zulkaidah", "Jumat", "06-Jan-2068", 29], ["Zulhijah", "Sabtu", "04-Feb-2068", 30]],

1491: [["Muharam", "Senin", "05-Mar-20[illegible]", "Sa[illegible]", "Selasa", "03-Apr-2068", 30], ["Rabiulawal", "Kamis", "03-May-2068", 29], [illegible]", "Jumat", "01-Jun-2068", 30], ["Jumadilawal", "Ahad", "01-Jul-2068", [illegible]", "Jumadila[illegible]r", "Senin", "30-Jul-2068", 30], ["Rajab", "Rabu", "29-Aug-2068", 30], ["Sya[illegible]ban", "[illegible]at", "28-Sep-2068", 29], ["Ramadan", "Sabtu", "27-Oct-2068", 30], ["Syawal", "Senin", "[illegible]v-2068", [illegible]", "Zulkaidah", "Rabu", "26-Dec-2068", 29], ["Zulhijah", "Kamis", "24-Ja[illegible]069", [illegible]

1492: [["Muharam", "[illegible]", "22-Feb[illegible]9", 30], ["Safar", "Ahad", "24-Mar-2069", 29], ["Rabiulawal", "Senin", "22-A[illegible]069", 30], ["Rabiulakhir", "Rabu", "22-May-2069", 29], ["Jumadilawal", "K[illegible]mis", "20-Jul[illegible]69", 29], ["Jumadilakhir", "Jumat", "19-Jul-2069", 30], ["Rajab", "Ahad", "18-Aug-2[illegible]", [illegible]"Syakban", "Selasa", "17-Sep-2069", 29], ["Ramadan", "Rabu", "16-Oct-2069", 30], [illegible]awal", [illegible]"15-Nov-2069", 30], ["Zulkaidah", "Ahad", "15-Dec-2069", 29], ["Zulhijah", "Se[illegible]", "1[illegible]070", 30]],

1493: [[illegible]ra[illegible]", "Rabu", "12-Feb-2070", 29], ["Safar", "Jumat", "14-Mar-2070", 29], ["R[illegible]awal", [illegible]tu[illegible]12-Apr-2070", 29], ["Rabiulakhir", "Ahad", "11-May-2070", 30], [illegible]al", [illegible]sa", "10-Jun-2070", 29], ["Jumadilakhir", "Rabu", "09-Jul-2070", 29], ["Rajab", "Kam[illegible]2070", 30], ["Syakban", "Sabtu", "06-Sep-2070", 29], ["Ramadan", "Ahad", "05-Oct-2070", [illegible]"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", ["Jumadilawal", "Jumat", "27-Apr-2074", 30], ["Jumadilakhir", "Ahad", "27-May-20[illegible], ["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", "[illegible]-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

],

[

"R[illegible]

"Aha

18-Ma[illegible]75

2

],

[

"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-N
  30
]
]
[
  "
  "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-207
  30
],
[
  "Ramad
  "
  2
],
[
  "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-207",
      29
    ],
    [
      "Jum
      Rabu",
      Mar-2
    ],
    [
      "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
        ]
    ]

})

# ---> ZAMAN DAHULU: 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=int(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, 96.1288), "Sabang": (5.8942, 95.3184), "Subulussalam": (2.6312,
98.0012), "Takengon": (4.6212, 96.8412), "Tapaktuan": (3.4812, 97.8112), "Singkil": (2.2851, 97.7942),
"Calang": (4.6312, 95.5812), "Blangpidie": (3.8312, 96.8412), "Karang Baru": (4.3112, 98.0512),
"Blangkejeren": (3.9912, 97.3512), "Simeulue": (2.7812, 95.9612), "Meureudu": (5.2412, 96.2512),
"Bireuen": (5.2031, 96.7112), "Simpang Kanan": "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": (-8.4512, 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,
106.6312), "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), "Je (-6.5888, 110.6675), "Blora": (-6.9697, 111.4184), "Grobogan": (-7.0264, 110.9187), "Suraka 5703, 110.8297), "Sukoharjo": (-7.6833, 110.8333), "Wonogiri": (-7.8122, 110.9253), "Karangan r": 7.5961, 110.9511), "Sragen": (-7.4267, 111.0211), "Boyolali": (-7.5317, 110 061), "Klaten 031, 110.6025), "Magelang": (-7.4706, 110.2178), "Temanggung": (-7.3167, 1 67), "Wonos o": (-7.3597, 109.9111), "Purworejo": (-7.7122, 110.0078), "Kebumen": (-7 7, 1 6522), "Purwokerto": (-7.4244, 109.2300), "Cilacap": (-7.7277, 109.0159), "Pu alingga 7.3875, 109.3675), "Banjarnegara": (-7.3961, 109.6967), "Pekalongan": (6, 109.675 "Batang": (-6.9111, 109.7289), "Pemalang": (-6.8919, 109.3814), "Tegal": (-6.867 1371), "Brebes": (-6.8722, 109.0436), "Kendal": (-6.9189, 110.2039), "Salati (-7.3305, 11 84)},
"Jawa Timur": {"Surabaya": (-7.2575, 112.7521), "Sido (-7.4412, 112.7112), "Gresik": (-7.1612, 112.6511), "Malang": (-7.9797, 112.6304), "Batu": (112.5239), "Kediri": (-7.8172, 112.0116), "Blitar": (-8.0983, 112.1681), "Tulung 0612, 1 1.9012), "Nganjuk": (-7.6012, 111.9012), "Trenggalek": (-8.0512, 111.7112), diu 7.6 98, 111.5239), "Ngawi": (-7.4012, 111.4411), "Magetan": (-7.6512, 111.3312 o": 12, 111.4612), "Pacitan": (-8.2012, 111.0912), "Jember": (-8.1712, 113.701 ": (8.2112, 114.3611), "Bondowoso": (-7.9112, 113.8212), "Situbondo": (-7. 114. 01), "Probolinggo": (-7.7554, 113.2162), "Lumajang": (-8.1312, 113.2212) ojone ": (7.1512, 111.8811), "Tuban": (-6.9012, 112.0511), "Lamongan": (-7.1212, 112.41 ngkala 7.0412, 112.7411), "Sampang": (-7.1812, 113.2411), "Pamekasan": (12, 1 711), "Sumenep": (-7.0112, 113.8611), "Pasuruan": (-7.6449, 112.9061), "Moj ": (-7.47 2.4311), "Jombang": (-7.5512, 112.2312)},
"Kalimantan Barat": {"Pont ": (-0.0263, 109.3425), "Singkawang": (0.9023, 108.9711), "Mempawah": (0. 12, 108.961 Sambas": (1.3512, 109.3012), "Sintang": (0.0712, 111.4912), "Ketapang": (-1.84 10 12)},
"Kalimant lat "B masin": (-3.3167, 114.5900), "Banjarbaru": (-3.4431, 114.8286), "Pelaihar 3. 11 12), "Kotabaru": (-3.2412, 116.2212), "Tanjung": (-2.1812, 115.3912)},
"Kal en ": {"Palangkaraya": (-2.2107, 113.9213), "Sampit": (-2.5312, 112.9512), "Pa kalan B 2. 12, 111.6212), "Muara Teweh": (-0.9512, 114.8912), "Kapuas": (-3.0112, 11),
"K mur": {"Samarinda": (-0.4949, 117.1492), "Balikpapan": (-1.2654, 116.8312), "Bontan 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, 130.811[illegible],
"Papua Selatan": {"Merauke": (-8.4912, 140.4012), "Agats": (-5.5412, 138.1312)},
"Papua Tengah": {"Nabire": (-3.3612, 135.5012), "Timika": (-4.5412, 136.8[illegible]12)},
"Papua Pegunungan": {"Wamena": (-4.0612, 138.9412)},
"Riau": {"Pekanbaru": (0.5071, 101.4478), "Dumai": (1.6712, 101.44[illegible], "Bat[illegible]kinang": (0.3312, 101.0312), "Siak": (0.7912, 102.0412)},
"Sulawesi Barat": {"Mamuju": (-2.6712, 118.8812), "Majene": ([illegible]12, 118.971[illegible]), "Polewali": (-3.4312, 119.3212)},
"Sulawesi Selatan": {"Makassar": (-5.1476, 119.4327), "[illegible]ppare": (-4.01[illegible], 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": ([illegible]2, 122.7812), "Poso": (-1.3912, 120.7512), "Donggala": (-0.6712, 119.7412)},
"Sulawesi Tenggara": {"Kendari": (-3.9722, 1[illegible]145), "[illegible]aubau": (-5.4612, 122.6112), "Kolaka": (-4.0512, 121.5912)},
"Sulawesi Utara": {"Manado": (1.474[illegible], 12[illegible]), "[illegible]ung": (1.4412, 125.1212), "Tomohon": (1.3212, 124.8312), "Kotamobagu": ([illegible], 12[illegible]312)},
"Sumatera Barat": {"Padang": ([illegible]9471, [illegible]41[illegible]), "Bukittinggi": (-0.3041, 100.3695), "Payakumbuh": (-0.2241, 100.[illegible]Pariam[illegible]-0.6171, 100.1239), "Solok": (-0.7937, 100.6599)},
"Sumatera Selatan": {"Pa[illegible]bang": (-2.9761, 104.7754), "Lubuklinggau": (-3.2952, 102.8611), "Prabumulih": (-3.4308, [illegible]155), "Lah[illegible]": (-3.7812, 103.5412), "Pagar Alam": (-4.0112, 103.2712)},
"Sumatera Utara": {"Medan": [illegible].5952, 98.6722), "Binjai": (3.6010, 98.4854), "Pematangsiantar": (2.9620, 99.0664), "Tebing Tinggi": [illegible].8364, 99.1625), "Sibolga": (1.7392, 98.7776)},

"Amerika [illegible]at":
"New [illegible]rk": (4[illegible], -74.0060),
"[illegible]s A[illegible]s": [illegible].0522, -118.2437),
"[illegible]icago": [illegible]8[illegible], -87.6298),
"[illegible]n": ([illegible]604, -95.3698),
"[illegible]": [illegible].3.4484, -112.0740),
"Se[illegible]": (47.6062, -122.3321),
"Mia[illegible]": (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, -43.1
    "Buenos Aires": (-34.6037, -5
    "Montevideo": (-34.9011, -5
    "Brasilia": (-15.8267, -4
    "Salvador": (-12.971
    "Fortaleza": (-3.7
    "Rosario": (-3
    "Asuncion": (-
    "Caraca
    "Me
    "Carta
    "Cuenca
    "
    "
    "Pi
    "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, 4[illegible]77),
    "Al-Baha": (20.0129, 41.4677), "Sakaka": (29.9697, 40.2064), "Arar": (30.9833, [illegible]),
    "Yanbu": (24.0891, 38.0637), "Al Ahsa": (25.3795, 49.5858)
},
# MENGGABUNGKAN KUNCI "Mesir" YANG SEBELUMNYA DITULIS 2 KA
"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, 34.32[illegible]9),
    "Marsa Matruh": (31.3525, 27.2[illegible]
},
# MENGGABUNGKAN KUNCI "M[illegible]ko" YA[illegible]SEBELUMNYA DITULIS 2 KALI
"Maroko": {
    "Rabat": (34.0209, [illegible]),
    "Casablanca": (3[illegible]5731, [illegible]98),
    "Marrakech": (31.6295, -7.9[illegible]
    "Fez": (34.033[illegible]5.[illegible]
    "Dakhla": (23.68[illegible]15[illegible]0),
    "Laa[illegible]ne [illegible]7.12[illegible]3.1625),
    "[illegible]gadi[illegible]0.4[illegible]-9.5981),
    "T[illegible]ngier [illegible]75[illegible], -5.8340),
    "[illegible]": (34.[illegible]24, -1.9086),
    "[illegible]": (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": (-43.9510, 176.5595),
```

```
    "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": (24.4[illegible]118.0894),
"Hefei": (31.8206, 117.2272), "Nanchang": (28.6820, 115.8579), "Suzhou": (31.[illegible]20.5853),
"Wuxi": (31.4912, 120.3119), "Ningbo": (29.8683, 121.5439), "Wenzhou": (27.9938, 120.6[illegible]94),
"Quanzhou": (24.8739, 118.6758), "Xuzhou": (34.2646, 117.1859), "Yan[illegible]i": (37.4638, 121.4479),
"Weifang": (36.7068, 119.1618), "Jinhua": (29.0781, 119.6495), "Sha[illegible]xing": (30.00[illegible] 120.5763),
"Nantong": (32.0147, 120.8646), "Guangzhou": (23.1291, 11[illegible]14), "Shenzhen": (22.5431, 114.0579),
"Dongguan": (23.0205, 113.7518), "Foshan": (23.021[illegible]13.1214), "Na[illegible]ng": (22.8146, 108.3200),
"Haikou": (20.0174, 110.3212), "Sanya": (18.25[illegible], 109.[illegible]Zhuhai": (22.2707, 113.5767),
"Zhongshan": (22.5176, 113.3928), "Shanto[illegible]39[illegible]1, 116.6[illegible]20), "Zhanjiang": (21.2707, 110.3590),
"Guilin": (25.2536, 110.2902), "Liuzho[illegible](2[illegible]6, 1[illegible]60), "Beihai": (21.4812, 109.1194),
"Huizhou": (23.1118, 114.4162), "Jiang[illegible](2[illegible]87, 113.0816), "Zhaoqing": (23.0515, 112.4651),
"Chaozhou": (23.6570, 116.6[illegible]6), "M[illegible]ing[illegible](21.6620, 110.9255), "Yangjiang": (21.8565, 111.9826),
"Wuhan": (30.5928, 11[illegible]3055), "[illegible]ngzhou": (34.7466, 113.6253), "Changsha": (28.2282, 112.9388),
"Luoyang": (34.6[illegible]98, 112[illegible]9), "Yichang": (30.6919, 111.2865), "Xiangyang": (32.0084, 112.1224),
"Jingzhou": (3[illegible]24[illegible]2.239[illegible]), "Yueyang": (29.3730, 113.1287), "Zhuzhou": (27.8274, 113.1330),
"Xia[illegible]n[illegible]825[illegible]2.9332), "Hengyang": (26.8954, 112.5719), "Kaifeng": (34.7969, 114.3076),
"Xinxiang[illegible]5.3[illegible]1, 113.9268), "Anyang": (36.0967, 114.3920), "Xuchang": (34.0253, 11[illegible]
[illegible]n": (33.7371, 113.3005), "Nanyang": (32.9908, 112.5283), "Xinyang": (32.1458, 114.084[illegible]
"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": (24.8[illegible]67.0011),
    "Lahore": (31.5204, 74.3587), "Islamabad": (33.6844, 73.0479), "Rawalpindi": ([illegible]
73.0441),
    "Faisalabad": (31.4504, 73.1350), "Multan": (30.1575, 71.5249), "Peshawar": (34.015[illegible]
71.5249),
    "Quetta": (30.1798, 66.9750), "Gujranwala": (32.1617, 74.1883), "Sialkot": (32.494[illegible]74.5229),
    "Hyderabad_PK": (25.3960, 68.3578), "Bahawalpur": (29.395[illegible]71.6833), "Sargodha": (32.0740,
72.6861),
    "Sukkur": (27.7052, 68.8574), "Larkana": (27.5570, 68.2028), "Shei[illegible]ura": (31.7167, 73.9850),
    "Rahim Yar Khan": (28.4212, 70.2989), "Jhang": (31.2[illegible]72.3181), "D[illegible] Ghazi Khan":
(30.0489, 70.6317),
    "Gujrat": (32.5736, 74.0741), "Tehran": (35.689[illegible]51.38[illegible]shhad": (36.2972, 59.6067),
    "Isfahan": (32.6539, 51.6660), "Karaj": (35.83[illegible]9[illegible]15), "Shiraz": (29.5918, 52.5836),
    "Tabriz": (38.0773, 46.2919), "Qom": (34.[illegible]50[illegible]6), "Ahvaz": (31.3183, 48.6706),
    "Kermanshah": (34.3142, 47.0650), "[illegible]552[illegible]0761), "Rasht": (37.2799, 49.5886),
    "Zahedan": (29.4963, 60.8629), "H[illegible]34[illegible]32[illegible]8.5146), "Kerman": (30.2839, 57.0834),
    "Yazd": (31.8974, 54.3569), "Ardab[illegible](38.[illegible]48.[illegible]973), "Bandar Abbas": (27.1832, 56.2666),
    "Arak": (34.0954, 49.6909), "[illegible]673[illegible]48.4787), "Sanandaj": (35.3144, 46.9923),
    "Kabul": (34.5553, 69.207[illegible]dahar[illegible]6200, 65.7158), "Herat": (34.3529, 62.2040),
    "Mazar-i-Sharif": (36.7[illegible]67.1[illegible]"Jalalabad": (34.4265, 70.4515), "Kunduz": (36.7286,
68.8681),
    "Taloqan": (36.7[illegible]61, 69.[illegible]), "Puli Khumri": (35.9446, 68.7151), "Sheberghan": (36.6676,
65.7529),
    "Zaranj": (30.9[illegible]504), "Maymana": (35.9214, 64.7836), "Ghazni": (33.5539, 68.4203),
    "Khost[illegible]33[illegible]"Shir Khan Bandar": (37.1704, 68.5833), "Charikar": (35.0136,
69.1715),
    "[illegible]5938, 64.3716), "Farah": (32.3745, 62.1164), "Asadabad": (34.8731, 71.1469),
    "Gardez[illegible]58[illegible], 69.2259), "Bamyan": (34.8216, 67.8242), "Baghdad": (33.3152, 44.3661),
    "[illegible]": (30.[illegible]1, 47.7835), "Mosul": (36.3400, 43.1300), "Erbil": (36.1901, 44.0090),
    "[illegible]4670, 44.3831), "Najaf": (31.9961, 44.3147), "Karbala": (32.6160, 44.0249),
    "Su[illegible]aniyah": (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): retur
    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 -
        s = int(round(
        if s == 60:
        if m ==
        retu
    else
        d = int(de
        m = int((deg
        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 "--:

    # BYPASS: Mengekstrak t   me   gar k bal terhadap tahun minus (SM/BCE)
    y, m, d, h, mn, s = tim     tc

    # Kalkulasi manu l untuk men     h zona waktu (tz_offset)
    total_minutes =  (h     mn  tz_offset * 60 + round(s / 60.0))

    # Mod     un     em     n jam tetap dalam format 0-23
    loca  ho     ot     inutes // 60) % 24
    l ca  minut     ota  inutes % 60

    re     {hour:02d}.{local_minute:02d}"

# ========================================
# 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
                print("Pygame
                it failed
        # --- Variabel
        mulasi 3D M
        lib ---
        self.eph3d_up
        lse
        self.eph
        Fal
        self.
        erver()
        self.pe_
        m.Sun()
        self.pe_m
        em.Moon()
        )

        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 = [
            "By the Name of Allah",
            "KALENDER HIJRIAH GLOBAL TUNGGAL",
            "KHGT Times 7.2, By Kasmui"
        ]
        return "\n".join(line.center(width) for line in lines)

    def load_ephemeris(self):
        try:
            # Gunakan logika auto-switch berdasarkan tahun saat ini saat startup
            current_year = datetime.datetime.now().year

            # Daftar prioritas dasar jika tidak ada ephemeris sama sekali
            priority_files = ["de421.bsp", "de440s.bsp", "de440.bsp", "de442.bsp", "de406.bsp",
"de441.bsp"]
            selected_bsp = None

            # Cek apakah ada satupun file bsp yang tersedia
            for filename in priority_files:
                if os.path.exists(os.path.join(BASE_DIR, filename)):
                    selected_bsp = filename
                    break

            if not selected_bsp:
                self.lbl_status.configure(text="M[illegible] (de421.bsp)...",
text_color="#00E5FF")
                download_custom_bsp("de421.bsp", "https://hisabmu.com/aifikih/de421.bsp")

            # Panggil engine deteksi [illegible] yang baru kita buat untuk startup
            self.auto_switch_ephemeris(current_year)

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

        except Exception as e:
            self.lbl_status.configure(text="Gagal memuat ephemeris!", text_color="#FF1744")
            messagebox.showerror("Error", f"Gagal memuat file ephemeris.\nDetail: {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) [illegible] 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) HIJRI_DB Auto-Builder (Admin)",
        "31) Status Kriteria Batas"
    ]

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

    def on_menu_click(mode_name):
        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 halaman utama
        self.switch_mode(mode_name)

    # Generate daftar menu ke dalam kotak scroll
    for item in menu_items:
        btn = ctk.CTkButton(
            self.menu_scroll_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=(5, 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="#EF5252").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))

        # Lanjut ke frame_vloc dst) ...

        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, "Lintang):", "-7.0667")
        self.entry_eph3d_lon = self.create_input_row(frame_eph3d_loc, "Lo Bujur):", "110.4100")

        frame_eph3d_waktu = ctk.CTkFrame(self.fram eph3d_ fg_color="#212121")
        frame_eph3d_waktu.pack(fill="x", padx=15, pa 5)
        ctk.CTkLabel(frame_eph3d_waktu, text=" GA N WAKTU & ELEVASI", font=("Segoe UI",
12, "bold")).pack(anchor="w", padx=10, p (8

        self.var_eph3d_tahun, self.lbl_eph tahun = self create_eph3d_slider(frame_eph3d_waktu,
"Tahun", -3000, 3000, int(curr_y)
        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_ = self.create_eph3d_slider(frame_eph3d_waktu,
"Tanggal", 1, 31, int(curr_d_np)
        self.var_eph3d_jam, self.lbl_eph3d_jam = self.create_eph3d_slider(frame_eph3d_waktu, "Jam
(Lokal)", 0, 23.99,
        self.var_eph3d_elev, self.lbl_eph3d_elev = self.create_eph3d_slider(frame_eph3d_waktu,
"Elevasi (
        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):", "-7.9667")
        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 MULAI <---
        self.entry_miz_waktu = self.create_input_row(frame_miz_opt, "Waktu (HH:MM):", "LIVE")

        # Tombol Pemicu Simulasi Mizwala
        self.btn_simulasi_miz = ctk.CTkBu
            self.frame_mizwala_input
            text="◎ Buka Simulasi Fis
            font=("Segoe UI", 12, "bold"),
            fg_color="#F57C0
            hover_color="#E65100
            command=self.buka_simu
        )
        self.btn_simulasi_miz.pack(fill="x", padx=15, pady=(10, 5))
        self.entry_interval = self.create_input_row(frame_miz_opt, "Interval (Menit):", "10")

        # -- AUTO BUILD 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):", "200")
        self.entry_kb_tz = self.create_input_row(frame_kb_loc, "Time Zone:", "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="SIMULASI 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, 0.5)

        # 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())

        # --- 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 Hisab & 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))

        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", padx=5, pady=4)

        self.lbl_cal_hijri = ctk.CTkLabel(self.f_cal, text="", font=("Georgia", 14, "italic"),
text_color="#FFD700")
        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 U    18), width=3
height=35, fg_color="#E65100", hover_color="#BF360C", command=sel     ov  release_ini  )
        self.btn_home.pack(side="left", padx=(0, 8))
        CTkToolTip(self.btn_home, delay=0.5, message="Kembali ke  eranda / Info    ")

        # 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="ns    ")
        self.textbox.insert("1.0", "Menunggu input dari  egun...")
        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_    e()

        # ---> TAMBAHKAN B    INI (RUM    KHUSUS UNTUK GRAFIK CHART) <---
        self.frame_char    t = ct    Frame(self.main_frame, fg_color="transparent")

        # ========== ==========================================
        # 1. PIND    AN    AL    SI KALENDER KE SINI
        # (W    dib    sebel    switch_mode dipanggil!)
        #    ==========================================

    -- ENDE    HIJRIAH INPUTS (MENU 26) --
        s    am    alender_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        kal    = 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="#121212")
        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 (Mas
        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))

        # 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



  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=10)

    # -- 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=5, 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=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=(5, 10))
        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.CTkComboBox(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 ---
        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, "Lintang):", "-7.0667")
        self.entry_qlon = self.create_input_row(frame_qloc, "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="#065F46", hover_color="#004D40", command=self.show_qiblah_map)
        self.btn_qmap.pack(fill="x", padx=15, pady=(10, 5))

    # --- MENTION 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, "Zona:", "7.0")

        # -- SUN TIMES --
        self.frame_st_inputs = ctk.CTkFrame(self.frame_sun_times, 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=15, pady=(5,

        # --- 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 Label
        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_ihtiyat = 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")

        # ...
        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=60, 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 a
        frame_vmtoggle = ctk.CTkFrame(s
        frame_vmtoggle.pack(fill="x", p
        self.radio_vmphase = ctk.Str
        ctk.CTkRadioButton(frame_vm
variable=self.radio_vmpha
        ctk.CTkRadioButton(
variable=self.radio_vmphase,
        frame_vmtim
        frame_vmtim
        self.radi
        row_
        r
        ct
w
        c
width=8
        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, 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_mode).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_mode).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))

        # --- BARIS BARU: 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")

        # -- 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, 5))

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

        self.combo_tahun_gerhana = ctk.CTkComboBox(gerhana_grid, values=[str(y) for y in range(-
3000, 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 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=5, 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=5, 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}" f      if 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(-2999, 3001)]
        self.combo_3d_y = ctk.CTkComboBox(f3d_row, values=      nge, width=85,
command=self.on_3d_date_change)
        self.combo_3d_y.set(curr_y)
        self.combo_3d_y.pack(side="left", pa    2)

        # ---> TAMBAHAN: TOMBOL PRO     HUS S     EN    16 <---
        self.btn_apply_3d = ctk.CTk     on(se     me_3d_inputs, text="▶ TERAPKAN TANGGAL",
font=("Segoe UI", 14, "bold"),      0, fg_     "#1565C0", hover_color="#0D47A1",
command=self.force_upda     d)
        self.btn_apply_3d.p     "x", pad     , pady=(5, 10))

        # Info navigas
        ctk.CTkLabel(s     fra     3d_in     uts, text="Klik & Drag layar untuk rotasi.\nScroll/Mousewheel
untuk Zoom
            fo     ("S     UI"     italic"), text_color="#FFD54F").pack(pady=5)

        # --- GHA -
        f     e_gh     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=(0, 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: Header 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"), 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 PENGAMATAN", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 

        # --- PERBAIKAN: Frame Tanggal dengan 
        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)]
        years_list = [str(y) for y in range(-3000, 3000)] # Range tahun -3000 s.d 3000

        # Baris Label
        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, pla      der_text="yyyy",
justify="center")
        self.entry_season_year.insert(0, curr_y)
        self.entry_season_year.grid(row=1, column=0,  adx=(0,

        # -- PLANETARY --
        self.frame_planet_inputs = ctk.CTkFra      (se      eba      color="transparent")
        frame_pl_date = ctk.CTkFrame(self.fra            nputs, fg_color="#212121")
        frame_pl_date.pack(fill="x", padx=      pady=5)
        ctk.CTkLabel(frame_pl_date,      xt="B      L OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", pad=1      ady=(8

        # --- PERBAIKAN: Fra      nggal de      Sistem Grid ---
        planet_grid = ctk.CTkFra      ame_pl_date, fg_color="transparent")
        planet_grid.pack(fill="x", pa      0, pady=(0, 10))

        # Baris      nju
        ctk.C      ab      anet_      d, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
colum      =0,      =(0
        ct      CTkLa      lan      _grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0,
co      n=      ady=      2))

        # Ba      Kotak Input
        self.e      y_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

            # Hitung 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=("C[illegible]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=(5,5))
    val_label.pack(side="right")

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

  def setup_eph3d_output(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 ELEMEN 3D")

    # Panel Info (Ditempatkan 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, az, r=10):
    return r * np.sin(az) * np.cos(alt), r * np.cos(az) * np.cos(alt), r * np.sin(alt)

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

        self.pe_obs.lat = str(lat_val)
        self.pe_obs.lon = str(lon_val)
        self.pe_obs.elevation = elev_val

        # Perhitungan Offset Zona Waktu (Longitude / 15)
        tz_offset = int(self.get_tz_from_lon(lon_val))

        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)

        jam = int(jam_desimal)
        menit = int((jam_desimal - jam) * 60)
        detik = min(59, max(0, int((((jam_desimal - jam) * 60) - menit) * 60)))

        waktu_lokal = datetime.datetime(y, m, d, jam, menit, detik)

        # Kurangi dengan offset dinamis agar dikonversi ke UTC untuk PyEphem
        self.pe_obs.date = waktu_lokal - datetime.timedelta(hours=tz_offset)

        self.pe_sun.compute(self.pe_obs)
        self.pe_moon.compute(self.pe_obs)

        # ==========================================================
        # 2. PERHITUNGAN TOPOSENTRIK (Berdasarkan lokasi pengamat)
        # ==========================================================
        alt_topo_sun = math.degrees(float(self.pe_sun.alt))
        az_topo_sun = math.degrees(float(self.pe_sun.az))
        alt_topo_moon = math.degrees(float(self.pe_moon.alt))
        az_topo_moon = math.degrees(float(self.pe_moon.az))
        elong_topo = math.degrees(ephem.separation(self.pe_sun, self.pe_moon))

        # ==========================================================
        # 3. PERHITUNGAN GEOSENTRIK (Berdasarkan pusat Bumi via Skyfield)
        # ==========================================================
        waktu_utc = waktu_lokal - datetime.timedelta(hours=tz_offset)
        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']

        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

        # Memperbarui posisi koordinat titik di ruang 3D
        sx, sy, sz = self.eph3d_r_ke_xyz(float(self.pe_sun.alt), float(self.pe_sun.az))
        mx, my, mz = self.eph3d_r_ke_xyz(float(self.pe_moon.alt), float(self.pe_moon.az))

        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 (Update Otomatis)" if getattr(self, 'eph3d_is_live', False)
else "⏸ WAKTU KUSTOM (Manual)"
        tz_str = f"UTC+{tz_offset}" if tz_offset >= 0 else f"UTC{tz_offset}"

        # ---> PERBAIKAN: MENGAMBIL NAMA KOTA & PROVINSI <---
        try:
            nama_kota = self.get_city_name()
            nama_prov = self.var_prov.get()
            lokasi_text = f"{nama_kota}, {nama_prov}"
        except:
            lokasi_text = "CUSTOM LOCATION"

        # Format Output Teks Info di sisi kiri atas Matplotlib
        info_text = (
            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): {self.pe_moon.phase:>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:
        # Ambil koordinat dan hitung offset zona waktu
        lat_val = str(self.entry_eph3d_lat.get())
        lon_val = float(self.entry_eph3d_lon.get())

        self.pe_obs.lat = lat_val
        self.pe_obs.lon = str(lon_val)
        tz_offset = int(self.get_tz_from_lon(lon_val))

        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])

        # Set titik awal pencarian ke jam 12 siang LOKAL (dikurangi offset agar jadi UTC)
        self.pe_obs.date = datetime.datetime(y, m, d, 12, 0, 0) - datetime.timedelta(hours=tz_offset)

        # Waktu sunset dari PyEphem adalah UTC murni. Kita kembalikan ke waktu lokal menggunakan tz_offset
        waktu_sunset_lokal = ephem.Date(self.pe_obs.next_setting(self.pe_sun).datetime() + datetime.timedelta(hours=tz_offset)

        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()
```

```
    except Exception as e:
        print(f"Gagal mencari sunset: {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:
            # Ambil koordinat untuk hitung offset zona waktu
            try:
                lon_val = float(self.entry_eph3d_lon.get())
            except:
                lon_val = 110.4100

            tz_offset = int(self.get_tz_from_lon(lon_val))

            # Gunakan UTC murni untuk menghindari bug 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 secara damping 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]:
            f.pack_forget()

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

        if hasattr(self, 'frame_gerhana_out'): self.frame_gerha
        if hasattr(self, 'frame_animasi_out'): self.fra
        if hasattr(self, 'frame_3d_out'): self.frame
        if hasattr(self, 'frame_chart_out'): sel
        if hasattr(self, 'frame_eph3d_out'):
        if hasattr(self, 'frame_kalender_
        if hasattr(self, 'frame_kalmas
        if hasattr(self, 'frame_win
        if hasattr(self, 'frame_
rid_remove()
i_out.grid_remove()
ve()
d_remove()
d_out.grid_remove()
self.frame_kalender_out.grid_remove()
self.frame_kalmasehi_out.grid_remove()
self.frame_winai_out.grid_remove()
out'): self.frame_kriteria_batas_out.grid_remove()

        self.anim_runnin
        self.is_viewin
= False
= False

        # Bersih
tidak menumpuk
        if ha
r(s
rt_out'):
            for w
 in self.frame_chart_out.winfo_children():
                widg
()

        #
ERGANTIAN MENU
        if "
ility Hilal" in mode:
            self
rame_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=sel        hitung)
        self.lbl_main_title.configure(text="Qiblah Di      ction &      ent Times")
        self.textbox.grid(row=1, column=0, sticky="      ")      Menampilkan box teks laporan
        self.textbox.configure(text_color="#FFD      ")
    elif "Moon Times" in mode:
        self.frame_mt_inputs.pack(fill="      , b      se      n_hitung)
        self.lbl_main_title.configure(te      Moo      Rise, T      nsit, and Set Times")
        self.textbox.grid(row=1, co      mn=0,      ky="      sew")
        self.textbox.configure(t      t_      r="#E      0")
    elif "Sun Times" in mo      :
        self.frame_st_inp      ts      k(fill="x"      ore=self.btn_hitung)
        self.lbl_main_      tle.con      e(text="Sun Rise, Transit, and Set Times")
        self.textbox      rid(row=1, c      n=0, sticky="nsew")
        self.textbox      nfig      text_color="#E0E0E0")
    elif "Pra      Tim      m
        se      am      inp      ack(fill="x", before=self.btn_hitung)
        self.      an      e.configure(text="Islamic Prayer Times & Twilight")
        self.tex      gr      ow=1, column=0, sticky="nsew")
        s      extbo      nfigure(text_color="#E0E0E0")
        e      "      in mode:
        s      ame_conv_inputs.pack(fill="x", before=self.btn_hitung)
        self.      bl_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("0.0", "Klik 'PROSES DATA' untuk memulai scanning global mencari titik pertama di daratan utama...")
            self.textbox.configure(state="disabled")
        elif "Equinox" in mode:
            self.frame_equinox_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()

        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()

        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()

    # 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_input.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Kalender Hijriah Interaktif (Global)")

        self.textbox.grid_remove()
        self.frame_kalender_output.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 visibilita 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 "b agar hitam p t dan tebal <---
        self.winai_output_box.insert("end", f" Sistem: n_konteks}\n\n", "bold")
        self.winai_output_box.see("end")
        self.winai_output_box.configure(state="

    elif "Mizwala" in mode:
        self.frame_mizwala_inputs.pac ll= or .btn_hitung)
        self.lbl_main_title.configure(te lkul tor Bayangan Tongkat Istiwa (Mizwala)")
        self.textbox.grid(row=1, c mn=0, ew")
        self.textbox.configure(t r="#FF

    elif "Auto-Builder" i
        self.frame_au build_in pack(fill="x", before=self.btn_hitung)
        self.lbl_mai title.c figu t="HIJRI_DB Auto-Builder (Generator Siklus KHGT)")
        self.text ox column=0, sticky="nsew")
        self.te .c re, t_color="#FFD54F", state="normal")
        se tb lete 0", "end")
        self.te x. t("1.0", " TOOL DEVELOPER / ADMIN \n\nKlik ' PROSES DATA'
u e-run gine Ephemeris NASA.\nSistem akan merekonstruksi ulang penanggalan
Ka ah Global Tunggal (KHGT) untuk rentang tahun yang Anda input, kemudian
menu ecara otomatis ke dalam file 'db_hijriah.json'.\n\n(Peringatan: Pemrosesan 100
Tahun ak emakan 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


    # ========================================
    # 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 hari ini dan 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.g      e 1.0
try: ikhtiyat_sec = int(self.entry_pt_ikhtiyat.g
except: ikhtiyat_sec = 16

earth = self.eph['earth']
sun = self.eph['sun']
loc = wgs84.latlon(lat, lon  evatio    =ele

# --- PERBAIKAN 2: B   wak    dan  z yang presisi ---
t0_dt = datetime.  me(year,  h, day, 0, 0, 0, tzinfo=tz_info)
t1_dt = t0_dt + datetim  medelta(days=1)

t0 = self.ts.f    ime(t  dt)
t1 = s    fr    ate    (t1_dt)
# ----------------------------------------------------------

t_arra    .li  pace(t0.tt, t1.tt, 2880)
t    rch =    .ts.tt_jd(tt_array)

a    = (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 saat Isya, hitung durasi malam
        if val_maghreb is not None and val_shuroq is not None:
            # Durasi siang (Julian Date: 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._is_tomorrow = True
                        self._is_calculating_alarm = True
                        threading.Thread(target=self.update_silent_schedule, daemon=True).start()
                    if 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"W[illegible]U {[illegible]me.upper()}", font=("Segoe UI", 24, "bold"),
text_color="#00E5FF").pack(pa[illegible]10))
        ctk.CTkLabel(popup, te[illegible]msg, f[illegible]("Segoe UI", 12), wraplength=350,
justify="center").pack(p[illegible])

        btn_frame = c[illegible].CTkFrame(p[illegible], fg_color="transparent")
        btn_frame.pa[illegible])

        def [illegible]
            if HA[illegible]G:
                try: p[illegible]ixer.music.stop()
                [illegible]ept: p[illegible]
            [illegible]troy()

        def se[illegible]_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()
    lat, lon = CITY_DB[prov][city]
    tz_val = self.get_tz_from_lon(lon)

    # Daftar semua nama variabel input lokasi dari seluruh modul
    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']
    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']
    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']

    # Menggunakan hasattr agar aman walaupun ada menu yang belum ter-load
    for name in lat_entries:
      if hasattr(self, name):
        ent = getattr(self, name)
        ent.delete(0, 'end')
        ent.insert(0, str(lat))

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

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

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

    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("Ramadhan")
            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 (Dibuat kebal error / Anti-Crash)
        self.textbox.grid(row=0, 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.is_animasi_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 UNTUK CENTER WINDOW <---
        tb.tag_config("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\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():
        # PERBAIKAN: Gunakan enumerate() karena months adalah tipe data lis
        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 di 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 change_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))

    # ========== INTEGRASI WIDGET KALENDER HIJRIAH INTERAKTIF ==========
    # State Kalender Preview
    self.demo_h_year = 1447
    self.demo_h_month = 9 # Ramadhan awal

    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=(10, 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=20, pady=(0, 20))

        # --- 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)
            ctk.CTkFrame(container, width=12, height=12, corner_radius=6, fg_color=color).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") # <--- Tambahkan
baris ini
        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():
            # --- PENAMBAHAN BARU: TUNGGU EPHEMERIS SELESAI DIMUAT DARI BACKGROUND THREAD ---
            if getattr(self, 'demo_h', 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 PADA 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 = -6.9667, 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_koordinat.configure(text=teks_info_bawah)
        # ----------------------------------------------------------------

        days_header = ["AHAD", "SENIN", "SELASA", "RABU", "KAMIS", "JUMAT", "SABTU"]
        for col, day_name in enumerate(days_header):
            color = "#FF5252" if col == 0 else ("#00E676" if col == 5 else "#FFFFFF")
            lbl = CTkLabel(self.demo_cal_grid, text=day_name, font=("Segoe UI", 14, "bold"),
text_color=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()) # Mengecek apakah tanggal di kotak sama dengan hari ini

            if is_today:
                b_color = "#FFEA00"  # <--- BAGIAN INI: Mengatur warna border menjadi Kuning Terang
                b_width = 2          # <--- BAGIAN INI: Mengatur ketebalan border menjadi 2 pixel
            else:
                b_color = "#FF5252" if col_cal == 0 else ("#0277BD" if col_cal == 5 else "#555555")
                b_width = 1.0 if (col_cal == 0 or col_cal == 5) else 1

            cell = ctk.CTkFrame(self.dem_cal_grid, fg_color="#0F172A", corner_radius=6,
border_width=b_width, border_color=b_color, width=200, height=75)
            cell.grid(row=row_ca, column=col_cal, padx=10, pady=2, sticky="nsew")
            cell.grid_propagate(False)

            frame_top = CTkFrame(cell, fg_color="transparent")
            frame_top.pack(fill="x", padx=10, pady=(5, 0))

            # --- MENGHITUNG DATA ASTRONOMI SAAT MAGHRIB (SUNSET) LOKAL ---
            # Mencari sunset lokal dari tengah hari agar tidak meleset
            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 # Fallback untuk daerah kutub

            # Konversi hasil pelacakan Sunset ke Mesin Skyfield (Sesuai Menu 1)
            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()

        # 1. Elongasi Geocentric
        sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
        elongasi = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg

        # 2. Tinggi Hilal Geocentric
        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))))

        # 3. Umur Bulan & Fraksi Iluminasi
        try:
            prev_nm = ephem.previous_new_moon(waktu_sunset_ephem)
            umur_bulan = waktu_sunset_eph - prev_nm
        except:
            umur_bulan = 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

        # --- Kolom Kiri (Angka Hijriah, Umur Bulan & Elongasi) ---
        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))
```

```
            # --- KOTAK KANAN (Gambar Bulan, Iluminasi & Tinggi Hilal) ---
            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,

            lbl_ilu = ctk.CTkLabel(box_kanan, text=f"Ilu: {ilu:.1f}%", font=("Consolas", 9, "bold"),
text_color="#FFFFFF")
            lbl_ilu.pack(anchor="e", pady=(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))

            # --- BAGIAN BAWAH (Tanggal Masehi & Tanda Event) ---
            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 ==========
# Memecah teks sapaan dan teks deskripsi
welcome_text = "Selamat datang di KHGT  es  n"
intro_1 = ("Aplikasi ini dikembangkan  an  us  ai perangkat lunak "
           "komprehensif dan mutakhir  nt  n astrometri presisi tinggi, pemetaan
visibilitas hilal global, "
           "dan penyatuan penanggalan I  internasional.\n")

# Memasukkan teks de  tag  berbeda
self.textbox.insert("  welcome  , "welcome_title") # Menggunakan tag baru
self.textbox.inse ("end",  _1, "body")           # Menggunakan tag standar

self.textbox.in  t("  "Melampaui Batas Pendahulunya (Beyond Accurate Times)\n", "h4")
intro_2 = ("Sela  ert  -tahun, dunia falak Islam sangat bergantung pada perangkat lunak
pionir pe  hu  plik  sebut sangat berjasa meletakkan standar dasar komputasi waktu salat
dan p a h  e  al komputasi.\n"
           "Na  g  erasi pendahulu tersebut umumnya dibangun di atas teori analitik klasik
(  t(  /SOPa  menggunakan arsitektur single-thread yang lambat, dan dikonsep murni untuk
ruky  e  l/zonal, bukan untuk integrasi kalender global.\n")
sel  box.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 disarankan MABIMS.\n", "list_item")

intro_4 = ("\nDengan arsitektur ini, KHGT Times adalah sebuah karya konten explorer 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 + "[REQUIREMENTS LIBRARY & ENGINE ASTROMETRI]\n" + garis_strip + "\n", "subtitle")

intro_tech = ("Sistem ini menggabungkan pemrograman pustaka (library) saintifik standar industri untuk mencapai keseimbangan antara presisi 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 [illegible]", "Bukan library kode, melainkan Basis Data fisik (.bsp) dari NASA. Menyediakan posisi absolut vektor X,Y,Z Tata Surya. Aplikasi otomatis melakukan 'Auto-Switch' antara DE421, DE430, DE442, atau DE406 berdasarkan rentang tahun yang diinput pengguna."),
("PyEphem (ephem)", "Library engine C-based turunan XEphem. Dipakai khusus sebagai akselerator kalkulasi brutal (Brute-Force), seperti iterasi mencari Waktu Terbenam (Sunset) dan Ijtima' Ratusan kota pada Modul 3 dan Modul 5 secara instan."),
("[illegible] (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 ... 2016. Berbeda dengan sistem lokal/zonal, KHGT bertujuan menyatukan umat Islam ... sebuah satu sistem waktu global.\n\n")
    self.textbox.insert("end", b2_intro, "body")

    # --- Prinsip Utama ---
    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 ... dianggap sebagai satu kesatuan zona wilayah.\n", "list_item")

    self.textbox.insert("end", " ▸ ", "highlight")
    self.textbox.insert("end", "... Faidah Transfer Visibilitas (Naql al-Rukyat): ", "tech")
    self.textbox.insert("end", "Jika hilal secara astronomis mungkin dirukyat di SATU titik daratan di bumi, maka ... 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")

    # --- Parameter Kriteria ---
    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")

    # --- Mekanisme Eksekusi ---
```

```
        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 [illegible] masuk bulan baru.\n", "list_item")
        self.textbox.insert("end", "• Jika Ijtimak SETELAH fajar Gisborne: Maka [illegible] masuk bulan baru DITUNDA lusa (Lihat hasil hitung pada tanggal = ijtimak + 1) agar wilayah [illegible] bumi tidak [illegible] berpuasa sebelum ijtimak terjadi.\n", "list_item")

        # ==================== BAGIAN 3: DAFTAR FITUR LENGKAP (DIRINCI) ====================
        self.textbox.insert("end", "\n" + garis_strip + "\n[ 3. MANUAL BOOK: PENJELASAN 31 MODUL MENU ]\n" + garis_strip + "\n", "subtitle")
        self.textbox.insert("end", "Gunakan Dropdown Menu di [illegible] Kiri untuk berpindah antar modul. Berikut rinciannya:\n", "body")

        detailed_features = [
            ("1) Visibility Hilal (KHGT)",
             "• Kegunaan: Menganalisis parameter hilal di suatu lokasi tepat saat Matahari Terbenam (Sunset).\n"
             "• Sub-Menu Input: Tanggal Observasi (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 waktu Geosentris dan Toposentris. Mengevaluasi langsung terhadap 5 kriteria sekaligus (KHGT, MABIMS, Danjon, Yallop).\n"),

            ("2) Global Visibilitas Map",
             "• Kegunaan: Membuat gambar Peta Global High-Definition (Heatmap) wilayah mana saja yang dapat melihat hilal.\n"
             "• Sub-Menu Input: Tanggal Observasi, Mode Fase Bulan, Target Pemetaan (Sunset, Moonset, Fajar Time).\n"
             "• Sub-Menu Mode Layer: Pengguna memilih apakah ingin merender peta Ketinggian Bulan, Elongasi, Kriteria MABIMS, atau Peta Arsiran Interseksi Seamless KHGT 2D.\n"
             "• Aksi & Output: Membuka Jendela Matplotlib baru. Menampilkan peta dunia dengan arsiran warna. Memiliki opsi Toolbar untuk Zoom dan tombol Ekspor ke format PNG/SVG Resolusi Tinggi.\n"),

            ("3) Analisis Hilal Global",
             "• Kegunaan: Pemindai massal (Iterasi Cepat) keadaan langit di seluruh Benua.\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.\n"),
```

("4) Altitude Chart Analyser",
"• Kegunaan: Membuat kurva 2D untuk memvisualisasikan pergerakan naik-turun Ketinggian Matahari dan Bulan.\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: Sistem Auto-Tracker (Pendeteksi Otomatis) untuk mencari di titik mana bagian mana hilal PERTAMA KALI lolos kriteria KHGT (Alt 5, Eln 8).\n"
"• Sub-Menu Input: Tanggal Pencarian Awal.\n"
"• Aksi & Output: Aplikasi membuang wilayah pulau kecil terpencil di Pasifik, lalu melakukan komputasi 2 Siklus Bulan untuk menemukan titik daratan utama (Mainland) pertama. Hasilnya dicetak dalam Map Interaktif yang mendeskripsikan apakah parameter waktu lolos PKG 1 (Batas 00:00 UTC) atau masuk evaluasi PKG 2 (Komparasi Fajar Gisborne NZ).

("6) Fase Bulan (Moonphase)",
"• Kegunaan: Kalkulator siklus bulan setahun penuh."
"• Sub-Menu Input: Tahun (Mendukung tahun Masehi hingga -3000 SM).\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: Menyajikan jadwal harian pergerakan Bulan di satu lokasi spesifik selama sebulan penuh.\n"
"• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
"• Aksi & Output: Mencetak tabel Waktu Terbit (Moonrise), Titik Puncak/Kulminasi (Transit), dan Waktu Terbenam (Moonset). (Waktu dihitung dengan efek refraksi atmosfer lokal).",

("8) Sun Times",
"• Kegunaan: Menyajikan jurnal perjalanan harian Matahari selama sebulan penuh.\n"
"• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
"• Aksi & Output: Tabel Waktu Sunrise, Transit Zawal (Waktu Dzuhur), dan Sunset. Sangat krusial sebagai fondasi utama penentuan jadwal puasa dan jadwal shalat lokal.",

("9) Sun & Moon Ephemeris",
"• Kegunaan: Modul Ekstraktor Data Mentah (Raw Data) astronomis untuk keperluan penelitian.\n"
"• Sub-Menu Input: Target (Sun/Moon), Ref Waktu (UT1/TDT), Lokasi (Geosentris/Toposentris), dan Set Waktu Interval (Misal dilacak per 1 Jam atau per 1 Hari).\n"
"• Aksi & Output: Tabel raksasa data saintifik murni. Berisi parameter R.A (Right Ascension), Deklinasi, Altitude, Azimuth, Jarak Aktual (Km), Semi-Diameter, dan Delta-T.",

("10) Qibla Time (Rashdul Lokal)",
"• Kegunaan: Mencari waktu bayangan harian di mana matahari tepat memotong sudut kiblat.\n"
"• Sub-Menu Input: Tanggal Observasi, Titik Koordinat.\n"

"• Aksi & Output: Mencetak Waktu Pertama dan Waktu Kedua (jika matahari memotong dua kali). Pengguna memanfaatkan waktu tersebut untuk mencocokkan arah Ka'bah hanya dengan melihat bayangan tiang lurus di halaman rumah."),

("11) Qiblah Direction & Times",
"• Kegunaan: Penghitung Azimut derajat arah Kiblat secara presisi berdasarkan model Bumi Ellipsoid WGS84.\n"
"• Sub-Menu Input: Titik Koordinat.\n"
"• Aksi & Output: Menampilkan derajat sudut Ka'bah dari Utara Sejati. Memiliki fitur Tombol Spesial 'Show Qiblah Map' yang memproyeksikan Garis Lengkung Great Circle di peta topografi Dunia dari titik Anda menuju Mekkah."),

("12) Prayer Times",
"• Kegunaan: Kalkulator Waktu Shalat 5 Waktu internasional.\n"
"• Sub-Menu Input: Tanggal (Harian/Bulanan), Koordinat Lokasi, Parameter Suhu dan Ketinggian tempat.\n"
"• Sub-Menu Fikih: Pengaturan Mazhab Ashar (Hanafi/Syafii), Kemilihan Metode Lembaga (Kemenag 20°, Muhammadiyah 18°, MWL, ISNA), Nilai Ikhtiyat (Kehati-hatian), dan Metode Lintang Tinggi (Aqrab Al-Balad, dll).\n"
"• Aksi & Output: Menghasilkan Tabel Jadwal Fajr, Syuroq, Dhuha (4.5°), Dhohur, Asr, Maghreb, Isha, dan Midnight. Mendukung ekspor ke Gambar (Jadwal Dinding)."),

("13) Konversi Kalender",
"• Kegunaan: Alat pengubah tanggal lintas sistem Masehi ke Hijriah KHGT dan sebaliknya).\n"
"• Sub-Menu Input: Mode Konversi (Hijriah), Input Tanggal, Koreksi Hari Ikhtiyat (-2 hingga +2).\n"
"• Aksi & Output: Menghitung secara astronomis jatuhnya fase bulan baru (bukan rumus pembagian hari manual) dan menghasilkan hasil konversi tanggal ke layar utama."),

("14) Analisis Gerhana",
"• Kegunaan: Mesin Pencari Otonom fenomena Syzygy (Gerhana) Solar maupun Lunar.\n"
"• Sub-Menu Input: Tahun Kalkulasi (Mendukung -3000 SM hingga Masa Depan).\n"
"• Aksi & Output: Kalkulasi interaktif Waktu Puncak Gerhana dan Wilayah Terlihat. Di bawah tabel, terdapat tombol aksi untuk Membuka Jendela Rincian Lokal, mengekspor Jalur Totalitas Gerhana Matahari ke format Google Earth (KML), dan Membuka Simulator Visual Telescope View."),

("15) Lintasan Hilal",
"• Kegunaan: Simulator grafis 2D letak kedudukan Matahari dan Bulan di Ufuk Pengamat secara Real-time.\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. Mengalkulasi rotasi derajat kemiringan cahaya Sabit Bulan agar selalu akurat menghadap pusat Matahari. Dilengkapi tombol 'Zoom Hilal' untuk observasi jarak dekat (Tracking Azimut)."),

("16) Sistem Sun Moon Earth",
"• Kegunaan: Simulator Spasial 3D Tata Surya Geosentris.\n"
"• Sub-Menu Input: Set Tanggal Simulasi 3D.\n"
"• Aksi & Output: Rendering Objek Bumi di tengah, memproyeksikan kerucut Hitam (Umbra Bayangan Bumi), serta mengorbitkan Bulan, Matahari, dan Planet-planet lain (Venus, Mars, Jupiter,

dsb). Bisa di-Rotasi menggunakan Kiri Mouse, dan di-Zoom (Perbesar) menggunakan Scroll Wheel Mouse."),

("17) Equinox & Solstice",
"• Kegunaan: Pencari Titik Balik Musim Astronomis Matahari.\n"
"• Sub-Menu Input: Tahun.\n"
"• Aksi & Output: Menghitung waktu absolut terjadinya Vernal Equinox (Semi), Summer Solstice (Panas), Autumnal Equinox (Gugur), dan Winter Solstice (Dingin)."),

("18) Planetary Times",
"• Kegunaan: Sama seperti Moon/Sun Times, namun dikhususkan untuk objek Planet Tata Surya.\n"
"• Sub-Menu Input: Pemilihan Target Planet (Merkurius, Venus, Mars, Jupiter, Saturnus, Uranus, Neptunus, Pluto), Bulan, dan Tahun.\n"
"• Aksi & Output: Tabel Jadwal Terbit (Rise) dan Terbenam (Set) harian selama 1 bulan penuh."),

("19) Simulasi Ephemeris 3D",
"• Kegunaan: Engine Alternatif Visualisasi 3D menggunakan Library Matplotlib murni.\n"
"• Sub-Menu Input: Slider Kontrol Waktu (Tahun, Bulan, Tgl, Jam) dan Tombol Cari Sunset Otomatis.\n"
"• Aksi & Output: Kanvas grid Bumi hijau. Melukis titik Matahari dan Bulan pada koordinat lintang/bujurnya secara Toposentrik. Sangat berguna untuk presentasi saintifik akademik."),

("20) Komparasi 50 Tahun (Ramadhan)",
"• Kegunaan: Super-Komputer analisis Awal dan Hari Raya umat Islam masa depan.\n"
"• Sub-Menu Input: Hanya diklik untuk sekali jalan (Menelusuri dari tahun 1447 H hingga 1496 H).\n"
"• Aksi & Output: Memunculkan tabel komparasi dua kolom. Kolom pertama hasil keputusan 1 Ramadhan KHGT (Geo Global), kolom kedua hasil keputusan Neo MABIMS (di titik rawan Ufuk Barat Indonesia / Sabang). Menarik vonis status 'Serentak (Sama)' atau 'Beda (Mundur 1 Hari)'."),

("21) Komparasi 50 Tahun (Syawal)",
"• Kegunaan: Algoritma sangat identik dengan Modul 20, namun dikhususkan sepenuhnya untuk melacak tanggal 1 Syawal (Hari Raya Idul Fitri) di masa depan."),

("22) Global Parameter Extractor (50 Tahun)",
"• Kegunaan: Modul pendamping fitur Komparasi (Modul 20-21).\n"
"• Aksi & Output: Mempreteli dan mengekstrak khusus angka mentah (Nilai Altitude atau Nilai Elongasi) hasil komputasi 50 tahun. Mencetaknya dalam wujud Tabel Mentah yang sangat cocok diekspor ke format CSV Excel (tombol Toolbar) untuk bahan Skripsi / Analisis Data Falak mahasiswa."),

("26) Kalender Hijriah Berjalan",
"• Kegunaan: Mesin Render Grafis Kalender Penanggalan Islam.\n"
"• Sub-Menu Input: Pilih Bulan dan Tahun Hijriah.\n"
"• Aksi & Output: Menggambar matriks kalender dengan penanggalan Hijriah sebagai angka utama (Tengah) dan kalender Masehi sebagai penopang (Bawah). Titik warna di kalender menandakan jadwal peringatan (Haram Puasa, Ayyamul Bidh, Senin-Kamis, dsb). Dilengkapi tombol cetak PDF / PNG Resolusi Tinggi."),

("27) Kalender Masehi Berjalan",
"• Kegunaan: Mesin Render Grafis Kalender Penanggalan Sipil Internasional.\n"
"• Sub-Menu Input: Pilih Bulan dan Tahun Masehi.\n"
"• Aksi & Output: Memproyeksikan hari Ahad dengan warna Merah pada kolom matriks. Penanggalan Gregorian berada di tengah, dan penanggalan KHGT disematkan sebagai pelengkap di sudut layar."),

("28) WinAI: Aplikasi AI Windows",
"• Kegunaan: Ruang Asisten Kecerdasan Buatan (Google Gemini) yang ditanamkan langsung ke dalam Aplikasi.\n"
"• Sub-Menu Input: Kotak Chat untuk mengetik Prompt (Pertanyaan) kepada [illegible]"
"• Aksi & Output: AI mampu membaca Konteks Tanggal dan Lokasi Sistem Anda. Terhubung dengan Database Offline Dokumen Tarjih Muhammadiyah (RAG System). AI dibekali fitur 'Math Interceptor', jika Anda mengetik 'Hitung Visibilitas Hilal Sekarang', AI akan menyuruh modul aplikasi untuk menghitungnya terlebih dahulu secara tersembunyi, lalu AI akan menerjemahkan laporan tabel sistem tersebut ke dalam bahasa manusia."),

("29) Kalkulator Mizwala (Bayangan)",
"• Kegunaan: Instrumen simulasi kompas panjang bayangan Tongkat Istiwa penentu waktu shalat awal.\n"
"• Sub-Menu Input: Tinggi Tongkat (cm), dan Waktu Simulasi (Bisa diketik Jam spesifik atau dibiarkan 'LIVE').\n"
"• Aksi & Output: Tabel harian pergerakan arah bayangan (Derajat Azimut Matahari) dan Panjang Bayangan (cm). Fitur 'Buka Simulasi Visual' akan merender Kompas Polar Matplotlib di mana garis bayangan akan berputar selaras dengan [illegible]ra Real-Time."),

("30) HIJRI_DB Auto-Builder (Admin)",
"• Kegunaan: Fitur khusus Developer/Pengembang untuk merakit fondasi Kalender Aplikasi (Siklus Pembuatan Data JSON) [illegible]
"• Sub-Menu Input: Tahun Awal Hijriah s/d Tahun Akhir Hijriah.\n"
"• Aksi & Output: Memaksa mesin NASA Ephemeris melakukan iterasi deteksi pergantian bulan berdasarkan kriteria KHGT secara brutal. Menulis seluruh hasilnya menjadi File 'db_hijriah.json' baru. Sistem otomatis memuat ulang kalender baru tersebut ke dalam memori aplikasi secara Live."),

("3[illegible] Kriteria [illegible]as",
"• Kegunaan: Simulator Anatomi Kriteria Imkanur Rukyat (KHGT).\n"
"• Sub-Menu Input: Tanggal, Lokasi, dan dua buah Slider Interaktif (Parameter Elongasi & [illegible]\n"
"• Aksi & Output: Sistem akan menaruh titik Bintang hasil kalkulasi astrometri aktual pada grafik Kartesian 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.")
]

# 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 7.2) ]\n" + garis_strip + "\n", "subtitle")

        updates = [
            ("[V7.2 - MODUL 29] Kalkulator Mizwala (Bayangan Tongkat Istiwa)", "Penambahan instrumen komputasi bayangan matahari interaktif. Meliputi tabel arah dan panjang bayangan per menit, penanda otomatis masuknya waktu Dhuha, Zawal (Dzuhur), dan Ashar, serta visualisasi kompas Polar 2D secara Live/Kustom."),
            ("[V7.2 - MODUL 30] HIJRI_DB Auto-Builder", "Tool khusus Admin/Developer untuk merekonstruksi dan merender ulang seluruh siklus penanggalan KHGT selama ratusan tahun. Sistem melakukan iterasi Ephemeris NASA dan mengekspor hasilnya langsung menjadi file JSON yang memutakhirkan memori kalender aplikasi secara real-time."),
            ("[V7.2 - MODUL 31] Status Kriteria Batas (Interactive Simulator)", "Engine visualisasi Matplotlib 2D interaktif baru untuk memetakan secara detail batasan fisik dan optis parameter elongasi-ketinggian hilal berdasarkan kriteria KHGT 5-8 derajat."),
            ("[V7.2 - WINAI] Integrasi AI Gemini Cerdas", "Asisten kecerdasan buatan terpadu. WinAI kini memiliki kapabilitas membaca konteks UI pengguna, melakukan pencarian internet secara otonom, dan merujuk pada ratusan dokumen Tarjih/Fikih secara offline (RAG). AI juga bisa dipicu untuk menjalankan modul perhitungan secara background."),
            ("[V7.2 - DUAL-CALENDAR SUITE] Rendering & Navigasi", "Sistem kalender visual (Modul 26 & 27) telah disempurnakan dengan offset tata letak matriks otomatis, integrasi file JSON dinamis, dan perbaikan engine Pillow (PIL) murni pada kanvas [4000x3000] pixel agar ekspor PDF/PNG tidak pecah/blur."),
            ("[V7.2 - ENGINE] Peningkatan Stabilitas & Penanganan BCE", "Penyempurnaan logika Auto-Switching Ephemeris untuk file JPL (DE406, DE440), perbaikan error bypass limitasi datetime Python untuk tahun-tahun Sebelum Masehi (BCE / SM), dan optimalisasi UI/UX dengan CustomTkinter.")
        ]
        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", "\n" + garis_strip + "\n[ 5. PANDUAN OPERASIONAL & EXPORT ]\n" + garis_strip + "\n", "subtitle")

        updates = [
            ("NAVIGASI MODUL", "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 31 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 Time... 7.2")

  def create_input_row(self, parent, label_tex... fau...):
    row = ctk.CTkFrame(parent, fg_color... ...")
    row.pack(fill="x", padx=10, pady=...
    ctk.CTkLabel(row, text=label_t..., ...("Segoe UI", 12)).pack(side="left")
    entry = ctk.CTkEntry(row, wi...=80, ju...="right")
    entry.delete(0, 'end')
    entry.insert(0, default...)
    entry.pack(side="ri...
    return entry

  def create_...nd_...(se..., parent, dy, dm, dd):
    # --- PE... ...AN... ...gg...akan sistem Grid terpusat ---
    grid_... ...e ... CTk...me(parent, fg_color="transparent")
    g...d_fra... ...pac... ..."x", padx=10, pady=(0, 10))

    ... Lab... Petunjuk
    ct... ...kL... (grid_frame, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=... ...dy=(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.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):
    # Daftar urutan prioritas beserta rentang tahun amannya (min_year, max_year)
    ephemeris_priority = [
        ("de421.bsp", 1900, 2050),      # Paling ringan & optimal untuk masa kini
        ("de440s.bsp", 1849, 2150),     # Alternatif resolusi tinggi menengah
        ("de406.bsp", -3000, 6000)      # Jangkauan luas 6000 tahun
    ]
    best_bsp = None

    # 1. Cari file pertama yang rentang tahunnya cocok DAN filenya ada di komputer
    for filename, min_yr, max_yr in ephemeris_priority:
        if min_yr <= target_year <= max_yr and os.path.exists(os.path.join(BASE_DIR, filename)):
            best_bsp = filename
            break

    # 2. Fallback Darurat: Jika tidak ada yang cocok dengan tahun target,
    # ambil file apa saja yang tersedia agar aplikasi tidak blank
    if best_bsp is None:
        for filename, _, _ in ephemeris_priority:
            if os.path.exists(os.path.join(BASE_DIR, filename)):
```

```
                best_bsp = filename
                break

        if best_bsp is None:
            best_bsp = "de421.bsp"

        # 3. Ganti ephemeris dan Verifikasi kelengkapan objek benda langitnya
        if self.ephemeris_name != best_bsp:
            try:
                # Load ke variabel sementara dulu (agar tidak merusak memori jika file rusak)
                temp_eph = self.load_obj(best_bsp)

                # --- SISTEM VERIFIKASI KEAMANAN FILE BSP ---
                missing_targets = []
                for target in ['earth', 'sun', 'moon']:
                    if target not in temp_eph:
                        missing_targets.append(target.upper())

                if missing_targets:
                    raise ValueError(f"File '{best_bsp}' tidak lengkap/corrupt!\nFile ini tidak memiliki data
untuk: {', '.join(missing_targets)}.\nPastikan Anda memakai file yang benar.")
                # -------------------------------------------

                self.eph = temp_eph
                self.ephemeris_name = best_bsp
                print(f"[Engine] Berhasil beralih ke {best_bsp} untuk tahun {target_year}")

            except Exception as e:
                print(f"[Error] Gagal memuat {best_bsp}: {e}")
                raise RuntimeError(str(e))

    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 "Grafik 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_kota_pertama, 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()

    elif "Komparasi 50 Tahun (Ramadhan)" 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()

    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()

    # 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()

  # MODUL TAMBAHAN 14: ANALISIS HILAL GLOBAL ITERATIF (EPHEM)
  # ==========================================
  def calculate_global_hilal(self):
    try:
      y = self.entry_gha_year.get()
      m = self.entry_gha_month.get()
      d = self.entry_gha_day.get()
      time_str = self.entry_gha_time.get()
      if not time_str: time_str = "12:00:00"

      tanggal_referensi_utc = f"{y}/{m}/{d} {t      str}

      matahari = ephem.Sun()
      bulan = ephem.Moon()

      waktu_ijtimak = ephem      s_new      n(tanggal_referensi_utc)
      str_ijtimak = str(eph      Date(      ijtimak))

      output_lines = []
      header = f"{self.get_head      8)}\n"
      header += f"      Hilal Global (KHGT, MABIMS, Wujudul Hilal) - Ephem
]'.center(13      \n"
      he      r +=      Referensi Pencarian (UTC): {tanggal_referensi_utc}\n"
      head      = f"      aktu Ijtimak/Konjungsi Terdekat (UTC): {str_ijtimak}\n"
      # PERB      N:      ks header diubah sesuai logika sorting baru
      h      er +=      Catatan: Diurutkan Berdasarkan Abjad Negara, lalu Abjad Kota.\n"
      h      ="*138 + "\n"
      h      += 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)

      # Buat list penampung untuk di-sorting nantinya
      hasil_kalkulasi = []

      for negara, kota_dict in CITY_DB.items():
        for nama_kota, koordinat in kota_dict.items():
          lintang, bujur = koordinat
```

```
            pengamat = ephem.Observer()
            pengamat.lat = math.radians(lintang)
            pengamat.lon = math.radians(bujur)
            pengamat.elevation = 0 # Normalisasi elevasi ke 0 agar adil secara global
            # WAJIB di-set agar PyEphem menerapkan rumus refraksi optik Toposentrik
            pengamat.pressure = 1010
            pengamat.temp = 25

            try:
               pengamat.date = ephem.Date(ephem.Date(tanggal_referensi_utc) - 0.5)
               waktu_sunset = pengamat.next_setting(matahari)
            except (ephem.AlwaysUpError, ephem.NeverUpError):
               hasil_kalkulasi.append({
                  'negara': negara,      # <--- TAMBAHAN
                  'kota': nama_kota,     # <--- TAMBAHAN
                  'waktu_sort': 999999.0, # Nilai ekstrem agar anomali ditaruh di paling bawah tabel
(jika disort via waktu)
                  'baris_teks': f"{negara[:15]:<15} | {nama_kota[:1                  4} | {'Anoma  Ekstrem':<16} |
{'-':<9} | {'-':<8} | {'-':<8} | {'-':<8} | {'-':<8} | Lintang Tinggi (Midnight S
               })
               continue

            pengamat.date = waktu_sunset
            matahari.compute(pengamat)
            bulan.compute(pengamat)

            # 1. Parameter Toposentris Bawaan Ephem
            alt_topo = math.degre  bulan
            elong_topo = math  eg    (ephe    aration(matahari, bulan))

            # 2. Paramet  sentris (D    ELARASKAN DENGAN MENU 2 - SKYFIELD)
            # Konvers  waktu  hem ke waktu Skyfield
            import pytz
            t_sunse  ky    f.ts.fr m_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))
            e    en    se    ['earth'].at(t_sunset_sky)
            ap    y =    enter.observe(self.eph['sun']).apparent()
            m_ap_    = obs_center.observe(self.eph['moon']).apparent()

            # Elong  Geosentrik (Standar Menu 2)
            e    geo = s_app_sky.separation_from(m_app_sky).degrees

            # Altitude Geosentrik (Standar Menu 2 - Memakai GAST)
            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))))


            umur_bulan_desimal = (waktu_sunset - waktu_ijtimak) * 24

            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"

               # --- EVALUASI STATUS BERJENJANG (KHGT -> MABIMS -> WUJUDUL HILAL) ---
               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 (Bulan Tua)"

            alt_g_str = f"{alt_geo:.2f}°"
            elong_g_str = f"{elong_geo:.2f}°"
            alt_t_str = f"{alt_topo:.2f}°"
            elong_t_str = f"{elong_topo:.2f}°"
            sunset_str = str(ephem.Date(waktu_sunset))

            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,      # <--- TAMBAHAN
               'kota': nama_kota,     # <--- TAMBAHAN
               'waktu_sort': waktu_sunset, # Objek ephem.Date (float) bisa langsung disort
               'baris_teks': baris_teks

         # SORTING: Urutkan abjad Negara dulu, lalu urut abjad Kota
         hasil_kalkulasi.sort(key=lambda x: (x['negara'].lower(), x['kota'].lower()))

         # Masukkan baris yang sudah terurut ke dalam output
         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', 1[illegible]old'),
background="#2b5797", foreground="white")

        # 2. Mengubah font Isi Tabel menjadi ukuran 13 dan tinggi baris ([illegible]ight) menjadi 32
        style.configure("Gerhana.Treeview", font=('Segoe UI[illegible]), rowheight=
background="#1e1e1e", foreground="white", fieldbackg[illegible]d="#1e1e1e")

        style.map('Gerhana.Treeview', background=[('[illegible]t', '#0078[illegible]7')], foreground=[('selected',
'white')])

        frame_tabel = ctk.CTkFrame(self.fr[illegible]t_g_color="transparent")
        frame_tabel.pack(fill="both", ex[illegible]True, p[illegible]y=[illegible], 10))

        kolom_gerhana = ("objek", "[illegible]", "mul[illegible]", "puncak", "akhir", "wilayah_global", "indo_vis")
        self.tabel_gerhana = tt[illegible]reevie[illegible]me_tabel, columns=kolom_gerhana, show="headings",
style="Gerhana.Treevie[illegible]

        self.tabel_ger[illegible]ana.heading([illegible]k", text="Objek")
        self.tabel_ger[illegible]a.h[illegible]ng("je[illegible]is", text="Jenis Gerhana")
        self.tab[illegible]rha[illegible]ea[illegible]"mulai", text="Kontak Awal (WIB)")
        self.t[illegible]_g[illegible]na.[illegible]ng("puncak", text="Puncak (WIB)")
        s[illegible].tab[illegible]er[illegible]_heading("akhir", text="Kontak Akhir (WIB)")
        s[illegible].tabel_[illegible]har[illegible]heading("wilayah_global", text="Karakteristik & Wilayah")
        [illegible]_tabel_ge[illegible]na.heading("indo_vis", text="Visibilitas (WIB Siang/Malam)")

        sel[illegible]_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 Totalit /Anularitas k
font=("Segoe UI", 12, "bold"), fg_color="#F57C00", hover_color="#EF6C
command=self.export_kml_solar)
        self.btn_kml.pack(side="left", padx=10)

        self.btn_simulasi_gerhana = ctk.CTkButton(frame_btn, text="🌍 Si si Visual Gerhana",
font=("Segoe UI", 12, "bold"), fg_color="#673AB7", hove color="#512DA8
command=self.buka_simulator_gerhana)
        self.btn_simulasi_gerhana.pack(side="left", p =10)

    def insert_gerhana_wrapped_row(self, valu gs):
        char_limits = [10, 16, 18, 18, 18, 30,
        wrapped_cols = []
        max_lines = 1

        for i, val in enumerate(val s):
            wrapped = textwra rap(str( width=char_limits[i])
            if not wrapped: w = [""]
            wrapped_col end(w ed)
            if len(wrapp d) > m _line ax_lines = len(wrapped)

        for line_ ra max mes):
            ro ta l[line_idx] if line_idx < len(col) else "" for col in wrapped_cols]
            self.t ge insert("", "end", values=row_data, tags=tags)

        ge ana.insert("", "end", values=["", "", "", "", "", "", ""], tags=tags)

    def an 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 maksimal 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 MATAHA
                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 ...tahari sedang di atas ufuk (Siang)
                    if alt_s_wil.degrees > 0:
                      m_app = loc.at(t_p).observe(m...).apparent()
                      sep_wil = s_app.separatio...om(...pp).degrees
                      # Cek apakah jarak su...ma...ri &... cukup dekat untuk tertutup
bayangan
                      if sep_wil < (sd_s...m + 0.8):
                        wilayah_te...hat.ad...ma_wil)
                        break

                if not wilay...lihat:
                  wil_s... = "Han...elintasi Samudra / Antartika"
                else:
                  wil... = ...in(sorted(list(wilayah_terlihat)))

                al...ipse...end({
                  'ob...': 'Matahari',
                  'j...enis,
                  't_...ak': 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.showwarni  Peri n", "Jalur sentral meleset dari permukaan Bumi (tidak
ada daratan yang dilewati sum  ngan p  ")
            return

        b_val = b[valid idx]
        delta_val = elta[valid_idx
        M_val = M[ lid
        v_hat  = v  :, v  idx]
        t  d =  .tt[  dx]

   = (-b_  np.sqrt(delta_val)) / 2.0
   = M_  + v_hat_val * k

   _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_{nama_file_time}(%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 - 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 matahari 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_s[illegible]e('%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[illegible]1°\n"
            f"Jarak Bumi-{objek}: {distance.km[illegible]0f} km"
            f"{extra_info}"
         )

         win_detail = ctk.CTkTop[illegible]lf)
         win_detail.title(f"Detail Parame[illegible] Lokasi - {objek}")
         win_detail.geome[illegible]50x450")
         win_detail.att[illegible]most", True)

         lbl_title = ct[illegible]k[illegible](win_detail, text=f"Detail Visual {objek} Saat Puncak", font=("Segoe UI",
16, "bold"), [illegible]co[illegible]#00[illegible]F")
         lbl[illegible]le.p[illegible](pad[illegible], 5))

         [illegible]extbox[illegible]k.C[illegible]Textbox(win_detail, width=500, height=350, font=("Consolas", 13),
w[illegible]", fg_[illegible]or="#1e1e1e")
         [illegible]ck(padx=20, pady=10, fill="both", expand=True)
         te[illegible]x.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=10, 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

        # Panggil 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 + 20, 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(50, self.update_animation)
            return

        self.anim_canvas.delete("dyn")

        # --- FUNGSI PEMBANTU 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 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 DI[illegible]) ----
    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_d[illegible]) /

    delta_az = abs(diff_az)
    delta_alt = abs(alt_m_d[illegible] - alt_s[illegible])

    if delta_az < 2.0: d[illegible]_az =
    if delta_alt < 2.[illegible] alt = 2.

    pad_zoom_x = 120
    pad_zoom[illegible] =
    ava[illegible]_w[illegible]ax([illegible] width - (2 * pad_zoom_x))
    [illegible]lab[illegible] = m[illegible]0, height - (2 * pad_zoom_y))

    scale_[illegible]ava[illegible]able_w / delta_az
    [illegible]le_y [illegible]ailable_h / delta_alt
    [illegible]x_[illegible]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="#00E67[illegible] wi[illegible]=3,
tags="dyn")
        self.anim_canvas.create_text(10, horizon_y - 12, text="GARIS UFUK (0°)", fill="#0[illegible],
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 pe[illegible]an[illegible]varia
                tick_h = 5
                if az_mark % 5 == 0: tick[illegible]10
                if az_mark % 10 == 0: tick_h[illegible]

                self.anim_can[illegible]as.create[illegible]e(mx, horizon_y, mx, horizon_y + tick_h, fill="white",
width=1.5 if tick_h > 5 el[illegible] tags="dy[illegible]

                # Tul[illegible] Angka Deraj[illegible]n 10 derajat
                if az[illegible]k[illegible]== 0:
                    [illegible]lf.[illegible]ca[illegible].create_text(mx, horizon_y + 28, text=f"{real_az}°", fill="white",
font=("Co[illegible]las[illegible]0), t[illegible]"dyn")

                # [illegible]is [illegible]ma Mata Angin HANYA pada 8 Arah Utama (Kelipatan 45 derajat)
                if re[illegible]az % 45 == 0:
                    [illegible]ah_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 + 15, 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)

         # ==========================================================================
         # PENYESUAIAN: Offset visual agar piringan atas terbenam tepat saat Maghrib.
         # Saat Maghrib (sunset visual), alt_s_deg (Geometris/Pusat) sekitar -0.833°.
         # Kita menambahkan 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="gray50", 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_sun_topo:.2f}° (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 & PHASE 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 JIKA TERTAHAN (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})"
        )

        box_width, box_height = 680, 195
        img_bg = Image.new("RGBA", (box_width, box_height), (24, 24, 24, 180))
        self._live_images['info_bg'] = ImageTk.PhotoImage(img_bg)
        self.anim_canvas.create_image(15, 15, image=self._live_images['info_bg'], anchor="nw",
tags="dyn")
        self.anim_canvas.create_rectangle(15, 15, 15 + box_width, 15 + box_height, fill="",
outline="#00E5FF", width=1.5, tags="dyn")
        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(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 = 450, 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="#00FF76", width=1.5, tags="dyn")
              self.anim_canvas.create_text(25, y1_khgt + 20, text="Memenuhi kriteria KHGT",
fill="#00FF76", 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, 1[illegible]100, [illegible]0

            tz_offset = int(self.get_tz_from_[illegible]

            # 3. MENGGUNAKAN PYEPHEM M[illegible] (Bukan Skyfield)
            # PyEphem memproses [illegible]ecara [illegible], bebas dari error "single Time / array"
            import ephem
            pengamat = ephe[illegible]erver()
            pengamat.lat = str(lat_[illegible]
            pengamat.lon = str(lon_va[illegible]
            pengamat.e[illegible]elev_val
            peng[illegible]pr[illegible]0  # Standar atmosfer
            pe[illegible]mp[illegible]  # Standar suhu
            mata[illegible]m.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 GEOCENTRIC 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 maks[illegible]ecilan

def on_mouse_wheel_3d(self, event):
    # Logika scroll menangkap pergerakan roda mouse [illegible]an aman
    if str(event.type) == 'MouseWheel':
        if event.delta > 0: self.zoom_in()
        else: self.zoom_out()
    else:
        if getattr(event, 'num', 0) == 4: s[illegible]
        elif getattr(event, 'num', 0) == [illegible]

def get_shared_date_comp[illegible]elf):
    """Memastikan tanggal[illegible]mbil [illegible]Menu 16 jika sedang aktif (Anti Macet)"""
    try:
        mode_aktif = [illegible]ode.get()
        if "Sistem S[illegible]Moon" in [illegible]aktif:
            return int([illegible]_3d_y.get()), int(self.combo_3d_m.get()), int(self.combo_3d_d.get())
        else:
            [illegible]vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
    except:
        now = [illegible]datetime.now()
        [illegible]now.year, now.month, now.day

def fo[illegible]pdate_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 [illegible]es disetel ke [illegible]isi Gerhana Matahari 3 Maret 2026.")

    def project_3d_raw(self, x, y, z):
        # Murni melakukan rotasi sumbu 3D dan [illegible]lasi [illegible]k kamera tanpa terpengaruh Zoom
        x1 = x * math.cos(self.cam_angle_x) - [illegible]n(self.cam_angle_x)
        z1 = x * math.sin(self.cam_angle_x) + z [illegible]self.cam_angle_x)
        y2 = y * math.cos(self.cam_angle_y) - z1 * [illegible]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 [illegible]) if (camera_dist + z2) != 0 else 1

        return x1 * factor, -y2 * factor, z2


    def adjust_zoom(self, factor):
        self.cam_zoom *= factor
        # Membatasi agar 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_anim[illegible])
        return

    canvas.delete("all")

    # Gambar background [illegible]ang ([illegible])
    random.seed(42)
    for _ in range(10[illegible]):
        rx, ry = rand[illegible]m.randint(0, [illegible]ndom.randint(0, h)
        canvas.crea[illegible]ov[illegible]ry, rx+1, ry+1, fill="#555555")

    try:
        [illegible], m, [illegible]_shared_date_components()
        [illegible]ow = [illegible]im[illegible]datetime.now()
        [illegible] = se[illegible]utc(y, m, d, now.hour, now.minute, now.second)

        e[illegible]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.c    om, ""))
        except:
            pass

        # 3. Kalkulasi dan Skala Zoom untuk Semua C
        for name, color, size, vis_dist, label in c    ial_    es:
            try:
                target_key = f'{name} baryc nter    t ir ['sun', 'moon'] else name
                try:
                    target_obj = self.eph[ arget_
                except KeyError:
                    target_obj = s    .eph[na

                pos_km = e    rth_o    (t_sim).observe(target_obj).position.km
                dist_km    = np.linalg.n    os_km)

                #    kan    k vi    dengan faktor zoom kamera saat ini
                x, p    =    m / dist_km) * (vis_dist * self.cam_zoom)

                pro    = s    .project_3d(px, py, pz, w, h)
                # Kali    juga ukuran objek (size) dengan zoom
                d    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 R
            f"* Scroll Mouse / Tombol UI u

        canvas.create_rectangle(20, 20,
        canvas.create_text(35, 35,

      except Exception as e:
        canvas.create_te
        h/2, text=
        stem Updating Data... {e}", fill="gray")

      self.after(50, self.update_3d
      mation)

  def set_e
    self.
    self.en
    self.entry_
    e
    ebox.
    winfo("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) aga[illegible] 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 [illegible]
    t_midnight = self.ts.utc(y, [illegible]

    # Aturan KHGT: Jika [illegible] sebelum [illegible] 15:00 UTC (Fajar Selandia Baru),
    # bulan baru ma[illegible] hari). Jika tidak, lusa (+2 hari).
    if h < 15:
        return t_mi[illegible].0
    else:
        re[illegible] t_[illegible].0

def calculat[illegible]nversion(self):
    [illegible]
        [illegible] = self.radio_conv_var.get()
        d[illegible](self.combo_conv_day.get())
        m_[illegible] = 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(ta      t - 35.0, tar    t_tt + 2.0)
current_nm_tt = None
for nm_tt in reversed(new_moons_tt_list):
    start_tt = self.calculate_khgt_1st_of_month(
    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 t     t di luar jangkauan Ephemeris ({self.ephemeris_name}).")

h_y, h_m = self.get_h     th_fro     (current_nm_tt)
start_tt = self.calc     ate_khg     of_month(current_nm_tt)

# Hitung sel     h hari d     Hijriah KHGT
h_d = int(     ound(t     get_     art_tt)) + 1
h_d += ad

    le h     = 0:
        h     -= 
    if h     = 
        h_m     y = 12, h_y - 1
    h     = 30
    h     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, approx_tt + 5)
            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)}

* Setting:
- Mode Konversi : {mode_text}
- Tanggal Input : {input_text}
- Koreksi Hijri : {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 WebAI)"""
    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)) # Mulai dari jam 00:00 waktu lokal
    t1 = self.ts.utc(year, month, day, 23 - int(tz)) # Sampai jam 23:00 waktu lokal

    t_sunset = None
    t_evs, y_evs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, lokasi_obs))
    for t_ev, is_sunset in get_safe_events(t_evs, y_evs):
        if not is_sunset:
            continue
        t_sunset = t_ev
        break

    if t_sunset is None:
        raise ValueError("Sunset (Matahari Terbenam) tidak ditemukan pada tanggal/koordinat
tersebut.")

    ts_tt = t_sunset.tt.item() if hasattr(t_sunset.tt, 'item') else t_sunset.tt

    t_bound_moon = self.ts.tt_jd(ts_tt + 1.5)
    t_moonset = None
    t_mevs, y_mevs = almanac.find_discrete(t0, t_bound_moon,
almanac.risings_and_settings(self.eph, moon, lokasi_obs))
```

```
    for t_ev, is_moonrise in get_safe_events(t_mevs, y_mevs):
        tm_tt = t_ev.tt.item() if hasattr(t_ev.tt, 'item') else t_ev.tt
        if not is_moonrise and tm_tt > (ts_tt - 0.5):
            t_moonset = t_ev
            break

    t_start_conj = self.ts.tt_jd(ts_tt - 5.0)
    t_end_conj = self.ts.tt_jd(ts_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_lat[illegible]och=t)
        _, mlon, _ = e.observe(moon).apparent().ecli[illegible]h=t)
        s_deg = slon.degrees.item() if hasattr(slon.d[illegible]item') else slon.degrees
        m_deg = mlon.degrees.item() if hasattr[illegible].d[illegible]s, 'item') else mlon.degrees
        return (m_deg - s_deg) % 360.0

    geo_earth = earth.at(t_sunset)
    app_moon_geo = geo_earth.observe([illegible]on).apparent()
    app_sun_geo = geo_earth.ob[illegible](sun).[illegible]rent()

    ra_moon, dec_moo[illegible], moon = [illegible]moon_geo.radec(epoch=t_sunset)
    ra_sun, dec_sun, dist_sun = [illegible]pp_sun_geo.radec(epoch=t_sunset)

    m_lat, m_lon, [illegible]on_geo.ecliptic_latlon(epoch=t_sunset)
    s_lat, s_[illegible] = [illegible]o.ecliptic_latlon(epoch=t_sunset)

    g[illegible]t = [illegible]
    lst_deg = [illegible].0) + lon

    [illegible]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_sunset)
    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 = (ts_tt - ti_tt) * 24.0
    else:
        moon_age_hours = 0.0

    if t_moonset is not None:
        tm_tt = t_moonset.tt.item() if hasattr(t_moonset.tt, 'item') else t_moonset.tt
        lag_time_hours = (tm_tt - ti_tt) * 24.0
    else:
        lag_time_hours = 0.0

    sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
    elongation = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg

    rel_alt_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_sunset)
    illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
    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 >= 10.3
    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 - 1/r * 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 = "E (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 Sunset Time at:  {get_hms_str(t_sunset, tz)} LT
- 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: {ts_tt:.5f}

- Sunset               : {get_hms_str(t_sunset, tz) + " LT":<16} G. Mo                :
{format_time_hms(moon_age_hours):<16}
- Moonset              : {moonset_str + " LT":<16} Moon Lag T                :
{format_time_hms(lag_time_hours):<16}

- G. Moon Altitude     : {format_angle(alt_moon_g          16} T. Moon Altitude     :
{format_angle(alt_moon_topo):<16}
- G. Sun Altitude      : {format_angle(alt_s          6} T          Altitude      :
{format_angle(alt_sun_topo):<16}

- G. Elongation        : {format_an          long          n):<16} G. Phase Angle       :
{format_angle(phase_angle):<16}
- G. Illumination      : {f"{ill          ation          2f} %":<16} G. Distance          : {f"{dist_km:,.2f} Km":<16}

------------------------------------------------------------------------------------------
[ MULTI-CRITERIA GLOBAL RESEA          RESULT ]
------------------------------------------------------------------------------------------
1. KHGT / Di          t Tu          G.A          5°, G.Eln>=8°)   : {str_khgt}
2. Neo M          MS          (          3°, G.Eln>=6.4°) : {str_mabims}
3. Ilyas Crit          Alt>=4°, G.Eln>=10.5°): {str_ilyas}
4. Danjon Limi          (          Eln >= 7.0°)         : {str_danjon}
5.          998) F          m  (q-Value = {q_yallop:+.3f})     : {yallop_stat}

==========================================================================================
===
"""
        return report

    def calculate_visibility(self):
        try:
            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 -3000) ---
            # Jika tahun -3000, mulai dari akhir Januari sesuai batas DE406
            if tahun == -3000:
                t0 = self.ts.utc(-3000, 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, al[illegible]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 zip(t_phases, y_phases):
                    t_lokal = self.ts.[illegible](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"{abs(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 == -3000 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: -3000-01-29 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.current_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 mulai 29 Jan -3000 WSM)")

   def calculate_ephemeris(self):
      try:
        obj_t = self.combo_eph_obj.get()
        tipe_ref = self.combo_eph_tref.get()
        loc_ref = self.combo_eph_lref.get()

        # Ambil input (Aman untuk tahun minus)
        sy, smon, 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 de (-3
        jd_limit = self.ts.utc(-3000, 1, 29, 12
        current_jd = max(t_start.tt, jd_limit)
        end_jd = t_end.tt

        if step_unit == "Day": st
        elif step_unit == "Ho
        elif step_unit == "
        else: step_jd = step_va

        output_line
        lat_st
        lo

            ---> A
            t
            ota = self.opt_city.get()
            _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_deg = (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:>
{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 u  JPL Ephem  NASA.")
        self.after(0, self.display_result, "\n".join(output_l

    except Exception as e:
        import traceback
        self.after(0, self.display_error, f"{st  \n{  ormat_exc()}")

  def calculate_qiblah(self):
    try:
        year = int(self.entry_qy  .g  ))
        month = int(self.ent  month  ()))
        day = int(self.ent  .get())
        target_date = d  e(year, month, day)

        self.auto_sw  _  meris(year)

        la  loa  f.en  at.get())
        lon =  (s  ntry_qlon.get())
        z = flo  lf.e  y_qtz.get())
        q  qibl  = self.radio_qiblah_var.get()

        p  = 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_clean = f"{abs(lat):.6f}".replace("+", "").replace("-", "-")
        lon_clean = f"{abs(lon):.6f}".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() + date[illegible]me.timedelta(hours=tz)
           time_1 = dt1.strftime("%H:%M:%S")
        if len(crossings) > 1:
           dt2 = crossings[1].utc_datetime() + [illegible]me.timedelta(hours=tz)
           time_2 = dt2.strftime("%H:%M:%S")

        data_row = f"\n{target_[illegible]rftime([illegible]/%m/%Y')}        {time_1:^10}
{time_2:^10}          \n"

        footer =
"""================================================================================
=====

* Remark
- Date form[illegible]d/mm/yyyy.
- The Symbol [illegible] for the date, refers to Summer Time.
- [illegible] that the Sun / Sun's Shadow does not reach the Qiblah Direction on that day.
- Kind [illegible]t at certain locations and on certain days, the Sun / Sun's Shadow reaches
  Qiblah [illegible]tion 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 dengan Pillow:", e)
        else:
            messagebox.showwarning("Peringatan", f"File gambar {map_file} gagal diunduh atau tidak
ditemukan di {full_map_path}")

        # 1. Arsiran warna 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. GARIS KONTUR 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 bug minus
        _, num_days = safe_monthrange(year, month)

        # --- PENGAMAN RENTANG DATA ---
        # Menghindari error batas ephemeris -3000-01-
        safe_day_start = 1
        if year == -3000 and month == 1:
            safe_day_start = 3

        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
        # Menggunakan string "Tahun-Bulan-Hari" sebagai kunci (Key) agar aman
        moon_data = 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"{in hl):02d} inl):02d}"

        # --- PEMBUATAN LAPORAN ---
        output_lines = []
        lat_str = format_angle(lat).repla ("-")
        lon_str = format_angle(lon).re "+", "")
        tahun_header = format_ta n_am ear)

        # ---> MENGAMBIL N A KO PROV NSI DARI SIDEBAR GUI <---
        try:
            nama_kota self.op .get()
            nama_pr = self pt_p et()
            lokasi_te f"{ a_kot}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone:
UTC{'+' if tz  else z})
        ex t:
            lo ex "CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+'
if t >= else )"

        "{self.get_header(84)}
{"[ Moo es ]".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.after(0, self.show_loading_ephemeris(year)

        lat = float(self.entry_st_lat.get())
        lon = float(self.entry_st_lon.get())
        elev = float(self.entry_st_elev.get())
        tz = 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 (-3000-01-29)
        safe_day_start = 1
        if year == -3000 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 (Z[illegible] Menc[illegible]tik 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 = (self.eph['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 (handle kasus tahun minus)
            date_display = f"{d:02d}/{month:02d}/{year:04d}"
            if year <= 0:
                # Contoh: -3000 -> 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 (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 (-3000-01-29)
        start_d_iter = 1
        if year == -3000 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} ({lon_clean}, Lat: {lat_clean}, Ele:{elev}m,
Zone: UTC{'+' if tz >= 0 else ''}{tz})"
        except:
            lokasi_text = f"CUSTOM LOCATION ({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 "  --:--:--  "
            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)

            y, m, d, h_f, m_f, s_f = t_res.utc

            if "H" in fmt_pt:
                return f"{int(h_f):02d}:{int(m_f):02d}:{int(s_f):02d}"
            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_a...ar)}"

            line = f"  {date_str:<15}  {str_fajr:^{pa...  {str_shuroq:^{pad}}  {str_dhuha:^{pad}}
{str_dhohur:^{pad}}  {str_asr:^{pad}}  {str_...^{pad}}  {str_isha:^{pad}}  {str_midnight:^15}"

            report_lines.append(line)

        report_lines.append(f"{...
        report_lines.append(...
* Remarks:-
- Date format: dd/mm/yyyy.
- Fajer: Beginning of Astronomical ...ilight. | Shuroq: Sunrise.
- Dhohur: Transit ...ghreb: Sunset. | Isha: End of Astronomical Twilight.
- Calculated ... JPL ... SA ...ne. Supported Year -3000 to 3000."""")

        final_...ort = "\n".join(report_lines)

        # --- TAMPILKAN TEKS KE LAYAR GUI ---
        ..., 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"{e}\n\n{traceback.format_exc()}")

def calculate_visibility_map(self):
    if not HAS_MATPLOTLIB:
        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 Besok (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, ...)
            # Menggunakan Native Skyfield agar efek Parallax & Refraksi dihitung 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(eph, 'moon', t_sunset)
            illum_val = illum.item() if hasattr(illum, 'item') else illum

            lons_coarse.append(real_lon)
            lats_coarse.append(lat)

            # Pilih Toposentrik dan Geosentrik berdasarkan mode peta
            if "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 (GARIS UNTUK PETA HALUS/SEAMLESS)
        # ==========================================================
        if is_khgt_map:
            grid_score = np.zeros_like(grid_a
            grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1

            cmap_inter = LinearSegmentedColormap([(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=0.8)
          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 (Kriteria Mode: Altitude (Geo), Elongasi
(Geo)", fontweight='bold', pad=15)

          # Mengubah Sumbu X menjadi Format
          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, = ax.plot([0], [0], color='#1A237E', lw=1.5, linestyle='solid', label='Batas Altitude
5° (G)')
          line_elong, = ax.plot([0], [0], color='#D32F2F', lw=1.5, linestyle='solid', label='Batas
Elongasi (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 U
            ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_a    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', '#00
            bounds = [-0.5, 0.5, 1.5, 2.5]
            norm = BoundaryNorm(bounds, cmap.N)
            cf = ax.contourf(LONS_fine, LATS_fine, grid_
norm=norm, alpha=0.2, zorder=2, antialiased=True)

            p1 = mpatches.Patch(color='#00
Alt>=3°, Geo Eln>=6.4°)')
            p2 = mpatches.Patch(colo
            p3 = mpatches.Patch(co
            ax.legend(handles=[
0.18), fontsize='small')

        elif "Ilyas" in
            grid_sc
            grid_
            cmap
            nd
            nc
            a
norm=
Eln>=10

            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) - 0.1 18
(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', '#                '#E040F                '#FFD54F', '#00E676'])
            bounds = [-0.5, 0.5, 1.5, 2.5, 3.5,
            norm = BoundaryNorm(boun                cm
            cf = ax.contourf(LONS_fin                LA                e, grid_score, levels=bounds, cmap=cmap,
norm=norm, alpha=0.2, zorder=2, an                ed=True)

            p1 = mpatches.Pa                cl                or='#F                ', label='F: Not Visible (Topo q-Value)')
            p2 = mpatche                tch(co                '#297                FF', label='D: Only with Optical Aid (Topo q)')
            p3 = mpat                tch(colo                040FB', label='C: Need Optical Aid First (Topo q)')
            p4 = mp                tches.P                (color='#FFD54F', label='B: Visible / Perfect Cond. (Topo q)')
            p5 =                patches.Pat                lor='#00E676', label='A: Easily Visible / Naked Eye (Topo q)')
            ax.le                d(h                es=[p                , p2, p3, p4, p5], loc='lower center', ncol=3,
bbox_to_an                =(0.                22),                ntsize='small')

        elif "Odeh" in crit_name:
            g                sco                [:] = 1
            grid_                ore[(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=
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_s      levels=boun      nter,
cmap=cmap_inter, norm=norm_inter, alpha=0.2, zorde      tialiased=True)
        cs_alt = ax.contour(LONS_fine, LATS_fin grid_a      =[5.0], colors=['#FFD54F'],
linewidths=2.5, linestyles='solid', zorder=3)
        ax.clabel(cs_alt, inline=True, fontsi      1, f      Geo Alt 5°')
        cs_elong = ax.contour(LONS_fi      LA      e, g      long, levels=[8.0], colors=['#00E5FF'],
linewidths=2.5, linestyles='dashed', zor      er=
        ax.clabel(cs_elong, inline=      fon      ize=11      mt='Geo Eln 8°')

        p_interseksi = mpat      h      tch(co      '#00E676', alpha=0.2, label='Area Pemenuhan KHGT
(Geo)')
        line_alt = plt.      ([0], [0],      ='#FFD54F', lw=2.5, linestyle='solid', label='Batas
Altitude 5° (Geo)')
        line_elong = plt.Line2      ], [0], color='#00E5FF', lw=2.5, linestyle='dashed', label='Batas
Elongasi 8° (Geo)')
        a      end      dle      _interseksi, line_alt, line_elong], loc='lower center', ncol=3,
bbox_to      ho      5,      fontsize='small')

      else:
        if para      target == "Altitude Bulan":
          et_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=4[illegible] color="transparent")
        toolbar_frame.pack(fill="x", side="bottom", [illegible]padx=10, [illegible]0, 10))

        toolbar = NavigationToolbar2Tk(canvas[illegible], to[illegible]r_frame)
        toolbar.update()

        def simpan_gambar_kustom():
            filepath = filedialog.asks[illegible]easfile[illegible]e(
                initialfile=f"HD_VisMap_KHGT_Se[illegible]ss.png" if is_khgt_map else
f"HD_VisMap_{param_targ[illegible].replace[illegible]')}.png",
                defaultexten[illegible]png",
                filetypes=[("PNG I[illegible]", "*.png"), ("SVG Vector", "*.svg")]
            )
            if filepath:
                fi[illegible]ef[illegible]pat[illegible]pi=300, bbox_inches='tight')
                [illegible]box.[illegible]nfo("Sukses", f"Peta HD berhasil diekspor ke:\n{filepath}")

        btn_exp[illegible] = c[illegible].CTkButton(toolbar_frame, text="🖼 Export Gambar HD", font=("Segoe UI",
12, [illegible]fg_co[illegible]="#E65100", hover_color="#BF360C", command=simpan_gambar_kustom)
        [illegible].pack(side="right", padx=10)

        self.[illegible]bl_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)
        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, 0 - 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(sh    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    Lon {lon}, TZ {tz:.2f})"
        except:
            lokasi_text = f"CUSTOM LOCATION (Lat    on}, TZ    z:.2f})"

        report = f"""{self.get_header(85)}
{"[ Rashdul Qiblah Lokal / Qibla Time ]"    em

* Settings:-
- Tanggal        : {tgl_str}
- LOKASI         : {lokasi_text
- Azimut Kiblat    : {q_az:    ari Utara    mur)
- Azimut Bayangan  : {shadow_    2f}°
================================================================================
===

1. WAKTU MATAHARI BERADA DI KIBLAT:
   (Saat matahari    t tepat di arah Ka'bah)
   >> {res_sun}

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()}")

    # ========================================
    # FUNGSI GENERAL GUI
    # ========================================
    def export_to_png(self):
        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.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )

        if filepath:
            try:
                # Setup Font (Gunakan Courier monospace agar tabel rapi seperti PDF/TXT)
                try:
                    font = ImageFont.truetype("cour.ttf", 14)
                except:
                    font = ImageFont.load_default()

                lines = report_data.split('\n')

                # Menghitung dimensi gambar yang dibutuhkan berdasarkan jumlah baris dan teks terpanjang
                max_line_len = max(len(line) for line in lines) if lines else 0
                char_w, char_h = 8, 18  # Estimasi pixel per karakter
                width = max(800, max_line_len * char_w + 60)
                height = max(600, len(lines) * char_h + 60)

                # Membuat Canvas dengan Background Putih (Light Mode)
                img = Image.new("RGB", (int(width), int(height)), "#FFFFFF")
                draw = ImageDraw.Draw(img)

                # Mencetak Teks per baris dengan Font Hitam
                y_text = 30
                for line in lines:
                    draw.text((30, y_text), line, font=font, fill="#000000")
                    y_text += char_h

                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: {e}")
```

```
  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", [illegible])  # Angka Hijriyah
      f_astro = ImageFont.truetype("courbd.ttf", 15)  # Data Astronomi
      f_greg = ImageFont.truetype("arialbd.ttf", [illegible])  # Masehi Bawah
      f_leg = ImageFont.truetype("arialbd.ttf", [illegible])  # Legenda

      # ---> FONT KHUSUS KETERANGAN & LOKASI AGAR PAS DI KANVAS <---
      f_ket = ImageFont.truetype("arial.ttf", [illegible])  # 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 berdasarkan lokasi
    y = str(self.cal_year.get())
    m = self.cal_month.get()
    m_data = [illegible][y][m]
    [illegible], 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="#FF[illegible]F", anchor="mm")
draw.text((700, 140), greg_title, font=f_sub, fill="#00E5[illegible]hor="mm")

# 4. Konfigurasi Grid Portrait
start_x = 85
start_y = 230
cell_w = 175
cell_h = 190
days_header = ["Ahad", "S[illegible]Selasa[illegible]bu", "Kamis", "Jumat", "Sabtu"]

for i, d_name in enu[illegible](days_h[illegible]r):
    x0 = start_x + i* cell_w
    y0 = start_y
    x1 = x0 + ce[illegible]-
    y1 = y[illegible]5
    co[illegible]bg[illegible]FF5[illegible]i == 0 else ("#00E676" if i == 5 else "#1E88E5")
    draw[illegible]an[illegible][x0, y0, x1, y1], fill=color_bg, outline=color_bg, width=2)
    raw.te[illegible](x0+1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")

[illegible]t_date.weekday() + 1) % 7
row[illegible]= 0, offset
curr_g[illegible]= 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) % 360.0

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)) % 360.0)
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_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="w[illegible]chor="l[illegible]")
      ly += 40
      if i == 2:
        ly = legend_y + 40
        lx = start_x + 450

    # ---> BAGIAN KETERANGA[illegible] LOKA[illegible]ETAK LENGKAP <---
    # Kita tempatkan elem[illegible]ooter [illegible]yesuaikan akhir dari letak legenda (dinamis)
    footer_y = legend_y[illegible]

    # 1. Teks Kete[illegible]ngan Param[illegible] (Teks Lengkap)
    teks_ket = "* [illegible]U[illegible]Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas),
Tinggi Hilal [illegible]Bawah[illegible] Di[illegible]g saat Maghrib."
    draw[illegible]t(([illegible]fo[illegible]), teks_ket, font=f_ket, fill="#FFFFFF", anchor="mm")

    # [illegible] Teks [illegible]Lok[illegible] (Kota dan Koordinat)
    t[illegible]si = [illegible] Kota: {nama_kota}, {nama_prov} | Koordinat: (Lat: {lat_val}, Lon: {lon_val})"
    d[illegible]te[illegible]00, footer_y + 45), ket_lokasi, font=f_loc, fill="#00E5FF", anchor="mm")

    # 3. Copyright / Watermark
    draw.text((700, footer_y + 105), "KHGT Times Engine V7.2 - 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 (N
        zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "
"Maluku Utara", "NTT"]

        # Loop untuk 2 siklus ijtimak
        for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "
Berjalan/Depan")]:

            # Cek dari H+0 (Hari Ijtimak) sampai maksimal H
            for offset_hari in range(4):
                kandidat_hari_ini = []
                waktu_pencarian = ephem.Date(w_
                for negara, kota_dict in CITY_D
                    if negara in zona_dikecual
                        continue

                    for nama_kota, k
                        lintang, bu
                        pengam
                        peng
                        pe
                        pe
                        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_sunse
                        'alt_geo': alt_geo,
                        'elong_geo': elong_eo,
                        'ijtimak': w_ijtima
                        'label': label_
                    })

            # Jika di hari i
            # Berhent
            if kandi
                kand
                h
                br

            if not ti
                self.after(0, self.display_error, "Tidak ada daratan utama yang memenuhi kriteria KHGT
pada

            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 mera
        all_lats = []
        all_lons = []
        for negara, kota_dict in CITY_DB.ite():
            for nama_kota, koordinat in k_d.ite():
                all_lats.append(koordinat[0])
                all_lons.append(koord[1])

        ax.scatter(all_lons, all_lats, co='red', marker='o', s=8, alpha=0.6, zorder=3)

        # Konfigurasi visual ma
        warna_bintang = ['#FFD54F', '#00E676']
        warna_tepi = ['re', 'white']

        te = "Perhitungan Multi-Siklus Selesai.\n\nTitik daratan utama pertama di dunia
yang masuk dalam KHGT:\n\n"

        for idx, pt in enumerate(list_titik):
            warna_bintang[idx % len(warna_bintang)]
            dge = 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)
            # Menambahkan jam = menambahkan hari pecahan (jam / 24.0)
```

```
        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 (KHGT GLOBAL) - ANTI CRASH TAHUN MINUS
        # ==========================================================

        # 1. Tentukan Batas 00:00 UTC berdasarkan hari lokal Sunset
        # Batas 00:00 UTC adalah: Jam 00 hari esoknya dari dt_local
        batas_00_utc_ephem = ephem.Date((int(y_l), int(m_l), int(d_l), 0, 0, 0)) + 1.0
        bt_jam = "00:00:00"
        fajar_info = ""

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

        # EVALUASI PKG 1 (Sunset < 00:00 UTC)
        if pt['sunset_utc'] < batas_00_utc_ephem:
            status_ijtimak = f"Kriteria PKG 1 Terpenuhi\n(Terpenuhi sblm {bt_jam} UTC)\n»
Kesimpulan: BESOK masuk bulan baru"

        # EVALUASI PKG 2 (Jika PKG 1 gagal)
        else:
            # Hitung Fajar Gisborne
            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()

            # Waktu Midnight NZ adalah UTC+12.
            # Maka kita set pengamat di Gisborne ke batas 00 UTC dikurangi 12 Jam (12/24 hari)
            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"

            # Syarat PKG 2: Titik di Amerika DAN Ijtimak sebelum Fajar Gisborne
            if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
              status_ijtimak = f"Kriteria PKG 2 Terpenuhi\n(Titik Amerika & Ijtima < Fajar)\n»
Kesimpulan: BESOK masuk bulan baru"
            else:
              if not is_amerika:
                alasan = "Titik bukan di Benua Amerika"
              else:
                alasan = f"Ijtima > Terbit Fajar"
              status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi\n({alasan})\n» Kesimpulan: LUSA
masuk awal bulan"

          except Exception as e:
            fj_jam = "N/A"
            fajar_info = f" Fajar Gisb.  : Error/N/A\n"
            status_ijtimak = "Kriteria PKG 2 Tidak Terpenuhi (Gagal Menghitung Fajar)\n»
Kesimpulan: LUSA masuk awal bulan"
        # ============================================================

        info_text = (
          f"TITIK KHGT ({pt['label']} {pt['type']})\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" Fajar Gisb.   : {fj_jam} UTC\n"
          f" Ijtimak       : {ijtimak_jam} UTC\n"
          f"Fajar Gisb."
          f"{'-'*45}\n"
          f"{status_ijtimak}"
        )

        props = dict(boxstyle='round,pad=0.6', facecolor='#1A1A1A', alpha=0.85, edgecolor=c_star,
linewidth=1.5)

        # --- Anchoring ke sudut BAWAH layar kanvas (Axes Fraction) ---
        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}) ns: {sta itimak.replace(chr(10), ' ')}\n\n"

        ax.set_title("Titik emenuh teria KHGT & Komparasi Fajar Gisborne",
fontweight='bold', pa=10)
        ax.set_xlab("Longitude")
        ax.set_ylab ")
        ax.set ks( 180, 181, 30))
        ax. yt (np. (-60, 61, 20))
        ax.g ue, style='--', color='gray', linewidth=0.5, zorder=1)

        f ht_la t()

        fr canvas = ctk.CTkFrame(win_plot)
        fra e_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()
```

```
        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 P ama: {e}\n\n{ ceback.format_exc()}")
        self.textbox.insert("1.0", text_summary)
        self.textbox.configure(state="disabled")

    def calculate_seasons(self):
      try:
        year = int(self.entry_season_yea get
        self.auto_switch_ephemeris(ye

        t0 = self.ts.utc(year, 1, 1
        t1 = self.ts.utc(year, 12, 31)
        t_seasons, y_seasons = almanac. discrete(t0, t1, almanac.seasons(self.eph))

        tz_wib = pytz.timezone('A karta')
        musim_nama = ["V al Equinox (Musim Semi Utara)",
                      "Summer Solstice (Musim Panas Utara)",
                      "Autumn 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 aman untuk 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)

            # --- BAGIAN BARU: 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 m
            date_display = f"{d:02d}/{month:02d}/{format_
            # Tambahkan baris data ke laporan
            line = f" {date_display:<18} {found_ri
            report += line

        report += "="*85
        report += "\n* Remarks: '-
        report += f"* Calculated
        # Tampilkan ke ru
        self.after(0, se

    except Except
        impo
        se
def display_
    l
    s
    sel
    self.t
    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 Location 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):
        # 1. Gunakan nama variabel string dan pengaman agar kebal dari error (anti-crash)
        lat_entries = ['entry_vlat', 'entry_epoch_lat', 'entry_qlat', 'entry_mt_lat', 'entry_st_lat',
'entry_pt_lat', 'entry_qtlat', 'entry_epoch3d_lat', 'entry_miz_lat', 'entry_kb_lat']
        lon_entries = ['entry_vlon', 'entry_epoch_lon', 'entry_qlon', 'entry_mt_lon', 'entry_st_lon',
'entry_pt_lon', 'entry_qtlon', 'entry_epoch3d_lon', 'entry_miz_lon', 'entry_kb_lon']
        tz_entries = ['entry_vtz', 'entry_epoch_tz', 'entry_qtz', 'entry_mt_tz', 'entry_st_tz', 'entry_pt_tz',
'entry_qttz', 'entry_miz_tz', 'entry_kb_tz']

        # Update Latitude
        for name in lat_entries:
            if hasattr(self, name):
                ent = getattr(self, name)
                ent.delete(0, 'end')
                ent.insert(0, str(lat))

        # Update Longitude
        for name in lon_entries:
            if hasattr(self, name):
                ent = getattr(self, name)
                ent.delete(0, 'end')
                ent.insert(0, str(lon))

        # Update Timezone
        try:
            tz_val = self.get_tz_from_lon(lon)
            for name in tz_entries:
                if hasattr(self, name):
```

```
            ent = getattr(self, name)
            ent.delete(0, 'end')
            ent.insert(0, str(tz_val))
      except Exception:
         tz_val = 7.0 # Default fallback jika gagal

      # 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 iCalendar lebih aman pakai 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",
        "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\n"
            headers = [self.tabel_gerhana.heading(col)["text"] for col 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_Hisab_Rukyat_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 (Tree View)
                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 yang tersedia.")
            return

        # --- 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_image(self):
        mode = self.combo_mode.get()

        # --- TAMBAHKAN INI 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 sebagai
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 LAIN (TANDA TEKS LAPORAN) ---
    report_data = self.text_box.get("1.0", "end-1c")
    if not report_data or "HASIL KESIMPULAN" 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, target_date, lat, lon, tz, elev):
    # Bersihkan widget Matplotlib sebelumnya di frame tersebut agar tidak menumpuk
    for widget in canvas_frame.winfo_children():
        widget.destroy()

    # Inisialisasi Figure Matplotlib
    fig, ax = plt.subplots(figsize=(8, 5), dpi=100)
    fig.patch.set_facecolor('#181818')
    ax.set_facecolor('#101010')

    # Generate data waktu (0-24 jam)
    hours = np.linspace(0, 24, 144) # Resolusi per 10 menit
    sun_alts = []
    moon_alts = []

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

    # Kalkulasi Altitude via Skyfield
    for h in hours:
```

```
            t = self.ts.utc(target_date.year, target_date.month, target_date.day, h - tz)
            sun_alts.append(observer.at(t).observe(sun).apparent().altaz()[0].degrees)
            moon_alts.append(observer.at(t).observe(moon).apparent().altaz()[0].degrees)

        # Plot Garis Kurva
        ax.plot(hours, sun_alts, color='#FFD54F', label='Matahari', linewidth=2)
        ax.plot(hours, moon_alts, color='#80DEEA', label='Bulan', linewidth=2)
        ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')

        # --- SUMBU X (JAM LOKAL) ---
        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 (Aktif jika tanggal yang dipilih adalah ha
        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, a  8)
            batas_bawah = min(sun_alts) if sun_alts else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekara  olor='#00E676', fontsize=10,
rotation=90, verticalalignment='bottom')

        # Formatting Label dan Font
        ax.set_title(f"Grafik Ketinggian Benda  gi  get_  .strftime('%d %b %Y')}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)  or='  ', fontsize=12)
        ax.set_ylabel("Ketinggian / A  tude (  olor='#ffffff', fontsize=12)

        ax.tick_params(colors='#  ', la  ize=1 )
        ax.grid(True, color='  ', linestyle  alpha=0.3)

        # Legend (Ket  angan Garis)
        ax.legend(fon  =  facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')
        fig.ti  la  ()

        # Render  i k dalam Tkinter Frame
        c  = Figu  anvasTkAgg(fig, master=canvas_frame)
        c  .  ()
        can  get_tk_widget().pack(fill="both", expand=True)


    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 = KHGT_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")

                # 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 >
            dt_mabims = dt_khgt
            status = "Serentak (Sama)"
        else:
            # Jika gagal memenuhi MABIMS di Sabang, bulan
            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 (Administrasi Indonesia & Override Kustom) atau Geografis
Global."""

        # 1. ZONA WAKTU INDONESIA (WIB, WITA, WIT)
        if 94.0 <= lon_val < 115.0:
            return 7.0
        elif 115.0 <= lon_val < 132.0:
            return 8.0
        elif 132.0 <= lon_val <= 141.0:
            return 9.0

        # 2. KHUSUS ZONA WAKTU ALASKA -> UTC-11
        # Mencakup seluruh rentang bujur Alaska (dari Kanada barat -140.0 hingga batas penanggalan -180.0)
        elif -180.0 <= lon_val <= -140.0:
            return -11.0

        # 3. 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 / HARI RAYA IDUL FITRI (1447 H - 1496
H) ]".center(125))
        output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokasi 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(95.318
        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(da
               status = "Beda (MABIMS Mundur 1 Hari)"

            # 7. Format Output Tabel
            str_khgt = dt_khgt.strftime("%d %b
            str_mabims = dt_mabims.strftime("%d %b %Y")

            baris = f"{thn_h} H | {ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
            output_lines.append(ba

            # Si
            if thn_h % 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 _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="#00B5FF")
            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)...\nMohon tunggu...\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 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 sementara
        self.after(0, _set_ui_loading)

        # 2. Proses Komputasi Utama
        try:
            output_lines = []
            output_lines.append(self.get_header(130))
            output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (RAMADHAN 1447H - 1496H)
]".center(130))
            output_lines.append("Metode: Akselerasi Database KHGT vs Toposentrik
Sabang".center(130))
            output_lines.append("=" * 130)
            output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<16} | {'KHGT: Parameter Global
(Geo)':<40} | {'MABIMS: Sabang (Topo)':<30} | {'Status'}")
            output_lines.append("-" * 130)

            # Dictionary untuk konversi nama bulan 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

# Setup Observer untuk Sabang
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

      data_ramadan = HIJRI_DB[thn_h][8]

      parts = data_ramadan[2].split('
      d = int(parts[0])
      m = bulan_map.get(parts[1]
      y = int(parts[2])

      dt_khgt = datetime.datetime(y, m, d)
      dt_rukyat = dt_khgt - datetime.timedelta(days=1)

      t_ref = ephem.Date(dt_rukyat)
      ijtimak = ephem.previous_new_moon(t_ref + 2)
      str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

      # Kriteria Danjon 23, Eln kita letakkan di depan
      str_kriteria = "Terpenuhi (Eln >= 8°, Alt >= 5°)"

      # Hitung MABIMS Sabang
      engine.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
      try:
         ss_sabang = engine.next_setting(sun)
         engine.date = ss_sabang
         sun.compute(engine)
         moon.compute(engine)

         alt_s = math.degrees(moon.alt)
         eln_s = math.degrees(ephem.separation(sun, moon))
         umur_s = (ss_sabang - ijtimak) * 24
```

```
                if alt_s >= 3.0 and eln_s >= 6.4 and umur_s > 0:
                    status_s = "Lolos"
                    kesimpulan = "Serentak"
                else:
                    status_s = "Belum"
                    kesimpulan = "Beda (MABIMS Mundur)"

                # Khusus menu 23, format dititikberatkan pada Elongasi
                str_sabang = f"Eln:{eln_s:5.2f}°, Alt:{alt_s:5.2f}° [{status_s}]"
            except Exception:
                str_sabang = "Anomali Sunset Sabang"
                kesimpulan = "Cek Manual"

            output_lines.append(f"{thn_h} H | {str_ijtimak:<16} | {str_khgt:<40} | {str_sabang:<30} |
{kesimpulan}")

            if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
                output_lines.append("-" * 130)

        except Exception as err_baris:
            output_lines.append(f"{thn_h} H | Error komputasi: {str(err_baris)}")

    output_lines.append("=" * 130)

    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):
  try:
    output_lines = []
    output_lines.append(self.get_header(130))
    output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (RAMADHAN 1447H - 1496H)
]".center(130))
    output_lines.append("Metode: Akselerasi Database KHGT vs Toposentrik
Sabang".center(130))
    output_lines.append("=" * 130)
    output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<16} | {'KHGT: Parameter Global
(Geo)':<40} | {'MABIMS: Sabang (Topo)':<30} | {'Status'}")
    output_lines.append("-" * 130)

    # 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}

    # Setup Observer untuk Sabang
    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

    data_ramadan = HIJRI_DB[thn_h][8]

    # Parsing tanggal
    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)
    dt_rukyat = dt_khgt - datetime.timedelta(days=1)

    t_ref = ephem.Date(dt_rukyat)
    ijtimak = ephem.previous_new_moon(t_ref)
    str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

    str_khgt = "Terpenuhi (Alt≥3° Elong≥6.4°)"

    # Evaluasi MABIMS Sabang
    engine.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
    try:
      ss_sabang = engine.next_setting(sun)
      engine.date = ss_sabang
      sun.compute(engine)
      moon.compute(engine)

      alt_s = math.degrees(moon.alt)
      eln_s = math.degrees(ephem.separation(sun, moon))
      umur_s = (ss_sabang - ijtimak) * 24

      if alt_s >= 3.0 and eln_s >= 6.4 and umur_s > 0:
        status_s = "Lolos"
        kesimpulan = "Serentak"
      else:
        status_s = "Belum"
        kesimpulan = "Beda (MABIMS Mundur)"

      str_sabang = f"Alt:{alt_s:5.2f}°, Eln:{eln_s:5.2f}° [{status_s}]"
    except Exception:
      str_sabang = "Anomali Sunset Sabang"
      kesimpulan = "Cek Manual"
```

```
            output_lines.append(f"{thn_h} H | {str_ijtimak:<16} | {str_khgt:<40} | {str_sabang:<30} |
{kesimpulan}")

            # Garis pembatas per dekade
            if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
               output_lines.append("-" * 130)

          except Exception as err_baris:
            output_lines.append(f"{thn_h} H | Error baris: {str(err_baris)}")

        output_lines.append("=" * 130)

        # --- MENGIRIM TEKS KE LAYAR ---
        final_text = "\n".join(output_lines)
        self.after(0, self.display_result, final_text)

      except Exception as e:
        import traceback
        self.after(0, self.display_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="Menghitung Tabel Syawal...", text_color="#00E5FF")
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Awal
Syawal)...\nMohon tunggu...\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 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(130))
        output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (SYAWAL 1447H - 1496H)
]".center(130))
        output_lines.append("Metode: Akselerasi Database KHGT vs Toposentrik
Sabang".center(130))
        output_lines.append("=" * 130)
        output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<16} | {'KHGT: Parameter Global
(Geo)':<40} | {'MABIMS: Sabang (Topo)':<30} | {'Status'}")
        output_lines.append("-" * 130)

        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

        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, 1[illegible]):
           try:
              if thn_h not in H[illegible]_DB:
                 continue

              # INDEX 9 ADALAH B[illegible] SYAWAL
              data_sy[illegible]al = [illegible]RI_DB[thn_h][9]

              [illegible]parts = [illegible]data_[illegible]al[2].split('-')
              [illegible]t(parts[0])
              m = bulan_map.get(parts[1], 1)
              [illegible] = int(parts[2])

              dt_khgt = datetime.datetime(y, m, d)
              # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal)
              dt_rukyat = dt_khgt - datetime.timedelta(days=1)

              t_ref = ephem.Date(dt_rukyat)
              ijtimak = ephem.previous_new_moon(t_ref + 2)
              str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

              str_khgt = "Terpenuhi (Alt >= 5°, Eln >= 8°)"

              # Evaluasi MABIMS di Sabang pada akhir Ramadhan
```

```
            engine.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
            try:
                ss_sabang = engine.next_setting(sun)
                engine.date = ss_sabang
                sun.compute(engine)
                moon.compute(engine)

                alt_s = math.degrees(moon.alt)
                eln_s = math.degrees(ephem.separation(sun, moon))
                umur_s = (ss_sabang - ijtimak) * 24

                if alt_s >= 3.0 and eln_s >= 6.4 and umur_s > 0:
                    status_s = "Lolos"
                    kesimpulan = "Serentak (Idul Fitri Sama)"
                else:
                    status_s = "Belum"
                    kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"

                str_sabang = f"Alt:{alt_s:5.2f}°, Eln:{eln_s:5.2f}° [{status_
            except Exception:
                str_sabang = "Anomali Sunset Sabang"
                kesimpulan = "Cek Manual"

            output_lines.append(f"{thn_h} H |          ijti      <16 | {str_khgt:<40} | {str_sabang:<30} |
{kesimpulan}")

            if (thn_h - 1446) % 10 == 0 a      thn_      != 149
                output_lines.appen      * 1

        except Exception a      r_bar
            output_lines      d(f"{thn_      | Error komputasi: {str(err_baris)}")

    output_line      append("=      2)

    final      = "      in(      ut_lines)
    se      fter(      mb      et_ui_done(final_text))

ex      ept Ex      ion      e:
    i      t tra      ack
    ,      lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))

def _pro      es_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...\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(130))
            output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (SYAWAL 1447H - 1496H)
]".center(130))
            output_lines.append("Metode: Aksara Pas Hisab KHGT vs Toposentrik
Sabang".center(130))
            output_lines.append("=" * 130)
            output_lines.append(f"{'Tahun':<7} | {'Waktu Ijtimak (UTC)':<16} | {'KHGT: Parameter Global
(Geo)':<40} | {'MABIMS: Sabang (Toposentrik)':<30} | {'Status'}")
            output_lines.append("-" * 130)

            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

            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]

            parts = data_syawal[2].split('-')
            d = int(parts[0])
            m = bulan_map.get(parts[1], 1)
            y = int(parts[2])

            dt_khgt = datetime.datetime(y, m, d)
            # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal)
            dt_rukyat = dt_khgt - datetime.timedelta(days=1)

            t_ref = ephem.Date(dt_rukyat)
            ijtimak = ephem.previous_new_moon(t_ref + 2)
            str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

            # Penulisan Eln di-utamakan (di depan)
            str_khgt = "Terpenuhi (Eln >= 8°, Alt >= 5°)"

            # Evaluasi MABIMS di Sabang pada akhir Ram
            engine.date = ephem.Date(dt_rukyat.re ace(ho  inute=0, second=0))
            try:
                ss_sabang = engine.next_setting(
                engine.date = ss_sabang
                sun.compute(engine)
                moon.compute(engine)

                alt_s = math.deg  es  on.alt)
                eln_s = math  egrees(e  m.sep ration(sun, moon))
                umur_s = ( s_  ng - ijti  * 24

                if alt  >= 3.0 and e  >= 6.4 and umur_s > 0:
                    sta  s  los"
                    esi  an  rentak (Idul Fitri Sama)"
                el
                    ta  s = "Belum"
                    im  lan = "Beda (MABIMS Istikmal 30 Hr)"

                rmat output Eln lebih dulu
                tr_sabang = f"Eln:{eln_s:5.2f}°, Alt:{alt_s:5.2f}° [{status_s}]"
            except Exception:
                str_sabang = "Anomali Sunset Sabang"
                kesimpulan = "Cek Manual"

            output_lines.append(f"{thn_h} H | {str_ijtimak:<16} | {str_khgt:<40} | {str_sabang:<30} |
{kesimpulan}")

            if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
                output_lines.append("-" * 130)
```

```
            except Exception as err_baris:
                output_lines.append(f"{thn_h} H | Error komputasi: {str(err_baris)}")

        output_lines.append("=" * 130)

        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[
            if start and start <= today < start + da      me.t      delt(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_kalen

def setup_kalen      er_out_fram      f):
    self.frame_ka      le      = ctk.CTkFrame(self.main_frame, fg_color="transparent")

    # ---      de      viga      etak ---
    h      ader      = c      TkFrame(self.frame_kalender_out, fg_color="#101018", corner_radius=8)
    he      der_ca      ck(      ="x", padx=15, pady=(0, 10))

    b      te      tk.CTkButton(header_cal, text="◀ Sebelumnya", command=self.cal_prev_month,
fg_colo      5      1F1F", hover_color="#333333", width=120)
    btn_p      ev.pack(side="left", padx=15, pady=10)

    self.lbl_cal_title = ctk.CTkLabel(header_cal, text="", font=("Segoe UI", 22, "bold"),
text_color="#00E5FF", justify="center")
    self.lbl_cal_title.pack(side="left", expand=True)

    btn_next = ctk.CTkButton(header_cal, text="Berikutnya ▶", command=self.cal_next_month,
fg_color="#1F1F1F", hover_color="#333333", width=120)
    btn_next.pack(side="right", padx=15, pady=10)
```

```
        frame_export = ctk.CTkFrame(self.frame_kalender_out, fg_color="transparent")
        frame_export.pack(fill="x", padx=15, pady=(0, 10))

        btn_pdf = ctk.CTkButton(frame_export, text="📄 Cetak PDF", font=("Segoe UI", 12, "bold"),
command=lambda: self.export_kalender("pdf"), fg_color="#D32F2F", hover_color="#B71C1C")
        btn_pdf.pack(side="right", padx=5)

        btn_png = ctk.CTkButton(frame_export, text="🖼 Simpan PNG", font=("Segoe UI", 12, "bold"),
command=lambda: self.export_kalender("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 bata
                self.cal_h_month = 0
                messagebox.showinfo("Batas Da  "T  nen  batas awal database.")
                return
        self.render_kalender()

    def cal_next_month(self):
        self.cal_h_month += 1
        if self.cal_h_month  1
            self.cal_h_mo  h = 0
            self.cal_h_y  ar += 1
            if self.ca  h_  r  HIJRI_DB:
                sel  .h_y  = 1
                ca  .mont  11
                me  eb  howinfo("Batas Data", "Telah mencapai batas akhir database.")
                retur
            er_k  ender()

    def re  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, Selasa=1, ... Ahad=6) -> 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 = ctk.CTkLabel(cell, text=str(h_day), font=("Consolas", 32, "bold"),
text_color="#4FC54F")
            lbl.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 Pi[illegible]ntuk [illegible]lusi Ultra HD"""
        img = Image.new("RGB", (1400, 1000), [illegible]
        draw = ImageDraw.Draw(img)

        try:
            f_title = ImageFont.truetype([illegible]bd.ttf", 46)
            f_sub = ImageFont.truetype("ari[illegible], 26)
            f_day = ImageFont.truetype("arialbd.ttf", 22)
            f_h = ImageFont.truetype([illegible]bd.ttf", 64)
            f_m = ImageFont.truetype("arial.ttf", 16)
        except [illegible]
            f_[illegible] f_h = f_m = ImageFont.load_default()

        y = self.ca[illegible]ye
        [illegible]f.cal_[illegible]onth
        [illegible]RI_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 BERDASARKAN 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+5, 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 V7.2 - 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))

    btn_pdf = ctk.CTkButton(frame_export, text="📄 Cetak PDF", font=("Segoe UI", 12, "bold"),
command=lambda: self.export_kalmasehi("pdf"), fg_color="#D32F2F", hover_color="#B71C1C")
    btn_pdf.pack(side="right", padx=5)

    btn_png = ctk.CTkButton(frame_export, text="📸 Simpan PNG", font=("Segoe UI", 12, "bold"),
command=lambda: self.export_kalmasehi("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]} {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
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 (M
        parts = h_str.split()
        if len(parts) >= 4:
            h_str_short = f"{parts[0]} {parts[
        else:
            h_str_short = h_st

        lbl_h = ctk.CTk
        lbl_h.pack(s

        col +=
        if
            co
            row +
        c
    # Si
    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_str(start_date)
    h_end = HijriConverter.get_hijri_date_str(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} H"
        else:
            hijri_title = f"Data KHGT: {h1_m_y} - {h2_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(curr_g.day + 2, 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 V7.2 - By [illegible]", font=f_sub, fill="#555555",
anchor="mm")
        return img

    def export_kalmasehi(self, export_form[illegible]
        img = self._generate_calmasehi_im[illegible]

        y = self.cal_m_year
        nama_bulan = BULAN_MA[illegible]lf.cal_[illegible]onth - 1]
        default_file = f"Kalend[illegible]lasehi[illegible]ama_bulan}_{y}"

        if export_format == "png":
            filepath = fi[illegible]dialog.asksa[illegible]filename(
                initialfile=[illegible]file}.png",
                def[illegible]xte[illegible]=[illegible]",
                f[illegible]yp[illegible]("PN[illegible]ge", "*.png")]

            if filepa[illegible]

                [illegible]ve(filepath, "PNG")
                [illegible]essagebox.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 GU
            self.after(0, lambda: self.lbl_st  onf  ur  te  = "Memproses Data Grafik...",
text_color="#00E5FF"))

            # 2. KALKULASI DI BA  GROU  THREAD (Bebas Tahun Minus)
            self.auto_switch_  eris(y)
            earth = self.ep ['earth
            sun = self.e  ['sun']
            moon = self  h['  ']
            loc = wgs  .la  (lat,  , elevation_m=elev)
            ob  ver  rth

            ours =  ins  ce(0, 24, 144) # Resolusi per 10 menit
            s  alts =
              = []

            for  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, m[illegible]lts):
        try:
            for widget in canvas_frame.winfo_children():
                widget.destroy()

            from matplotlib.figure import Figure
            from matplotlib.backends.backend_tkagg import [illegible]eCanvasTkAgg

            fig = Figure(figsize=(8, 5), dpi=100)
            fig.patch.set_facecolor('#181818')
            ax = fig.add_subplot(111)
            ax.set_facecolor('#101010')

            # ---> TAMBAHAN: Kalkula[illegible] Kurva [illegible]n Geosentrik (Standar Menu 2) <---
            moon_geo_alts = []
            lat = float(self.entry_[illegible].get())
            lon = float(self.en[illegible]n.get())
            tz = float(self.entry_vtz[illegible]())
            earth = self[illegible]ph['earth']
            moon = self[illegible]['[illegible]']

            fo[illegible]n h[illegible]:
                t = [illegible]ts.[illegible], m, d, h - tz)
                obs_[illegible]ar = [illegible]arth.at(t)
                [illegible]app = [illegible]s_center.observe(moon).apparent()

                [illegible]t = 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, [illegible]=0.8)
            batas_bawah = min(sun_alts) if sun_alts else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sek[illegible]', color='#00E676', fontsize=10,
rotation=90, verticalalignment='bottom')

        tahun_str = f"{y} M" if y > 0 else f"{abs(y[illegible] SM
        nama_bulan = ["", "Jan", "Feb", "M[illegible]", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]
        bulan_str = nama_bulan[m] if [illegible] <= 12 [illegible](m)

        ax.set_title(f"Grafik Ket[illegible] Benda [illegible]: {d:02d} {bulan_str} {tahun_str}", color='white',
fontsize=14, fontweight='b[illegible], pad=[illegible]
        ax.set_xlabel("W[illegible]kal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggi[illegible] Altitude (°)", color='#ffffff', fontsize=12)

        ax.tick_para[illegible](colors='#ffffff', labelsize=11)
        ax.gri[illegible], color='#[illegible]', linestyle=':', alpha=0.3)
        ax[illegible]ent[illegible]tsize=[illegible], facecolor='#181818', edgecolor='#333333', labelcolor='white',
loc='u[illegible]per [illegible])
        [illegible].tight_[illegible]ou[illegible]

        [illegible]gureCanvasTkAgg(fig, master=canvas_frame)
        ca[illegible].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 = [
            'AIzaSyAVIBjSEWKdyeAYnhYue6Zw6tSJKzQpfEw',
            'AIzaSyB8jOg5MPMrbpIWTmRT51rydJYLMFhRaXs',
            'AIzaSyD2UOrD7VaYU21FI6z8Ps1N4oEh86IrvJ4',
'AIzaSyCS_HjbzrEN5ddevYNjSvbv0K4oONBOHuY',
            'AIzaSyBnC4vNv-43Kwz3XMnYLKSHCdxUdVgU35M',
'AIzaSyA7Tyz5OAN95sgUbUCOkOGvUKwzijYIrY4',
            'AIzaSyBR3VtwvfDNLzi7s1YvZPGBb0yEOIRReLk', 'AIzaSyBabaqecoW-
bVXD8AwVKc4roEMTZiyrsTM',
            'AIzaSyARUKFKy8dUsiRJRlI4kV2OyiAnTeN6MSM',
'AIzaSyCChatfstuiHaiTxR5SmyHq7PSTMjcHoRg',
            'AIzaSyAFZCjVd9ehgfmFu5mY1pAVnm_EtqCKD20',
'AIzaSyD5e_bbZ9gPE2uafYnWLDXc3hzET5zVI3k',
            'AIzaSyAt5nF52Il_8nQnmfUgihjVUjONd6BxcMQ',
'AIzaSyDnvTKjTAf8HjcmM1ExupBqT7uPeibpmbg',
            'AIzaSyC5_ye8d4SZD4GOHe2aDPyrmqRbk7uwGH
'AIzaSyBJerwj4clbNjFuFs6Q_QYoMZnkOEerNcI',
            'AIzaSyBMh0M2eKXWLHHdZQatkhuUkngbg
DAdY0EXs1gvc7wcqLCdswxLCJIj2_W8',
            'AIzaSyDylXHRTDYk0xtmYCSRQGm
'AIzaSyDamciIrHuZo7EXpNgUKu3Fvd3
            'AIzaSyBn0HuPpJCoX4Rr1tNp
dVVbKAs_daQIwyOpq6VkJKos7Y'
            'AIzaSyAcfGW4K0J8HTU
2Ksq_iCGxkS1XO6V_IxaO6Y
            'AIzaSyARhHCJ6v
'AIzaSyAH5y4SnbKfy
            'AIzaSyBVY
            'AIzaSyCHfn
'AIzaSyCDQL
            'AI
_0',
            AIzaSy
'A
            z7pausR71hjj_WyVh2FxsqVZedj3dc'
        ]
        self.winai_primary_model = 'gemini-2.5-flash'
        self.winai_tracker_file = 'last_successful_index_winai.txt'
        self.winai_db_items = []
        self.winai_db_loaded = False

        self.winai_jawaban_terakhir = ""
        self.winai_pertanyaan_terakhir = ""
        self.winai_memori = ""
        self.winai_instruksi = ""
        self.winai_data_kal = ""
```

```
        # Muat database lokal di background
        threading.Thread(target=self.load_winai_databases, daemon=True).start()

        # ================ SIDEBAR INPUT (KIRI) ================
        self.frame_winai_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

        winai_info = ctk.CTkFrame(self.frame_winai_inputs, fg_color="#212121")
        winai_info.pack(fill="x", padx=15, pady=10)
        ctk.CTkLabel(winai_info, text="WINAI ASSISTANT", font=("Segoe UI", 12, "bold"),
text_color="#00E5FF").pack(anchor="w", padx=10, pady=(8, 2))

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

        now = datetime.datetime.now()
        self.winai_entry_y, self.winai_entry_m, self.winai_entry_d =
self.create_ymd_row(winai_date_frame, str(now.year), f"{now.m    :02d}", f"{n    .day:02d}")

        self.winai_input_entry = ctk.CTkEntry(winai_info, pla    older_text="    k pertanyaan...",
font=("Segoe UI", 13), height=40)
        self.winai_input_entry.pack(fill="x", padx=10, p    dy=10)
        self.winai_input_entry.bind("<Return>", lam    se.proses_pertanyaan_winai())

        self.btn_winai_kirim = ctk.CTkButton(    ai,     text="    im Pertanyaan", font=("Segoe UI", 12,
"bold"), fg_color="#1565C0", hover_co    r="    A    eight=35,
command=self.proses_pertanyaan_wi
        self.btn_winai_kirim.pack(fil    x", pa    0, p    dy=(0, 5))

        self.btn_winai_reset = ctk.CTkBu    (wina    info, text="🔄 Reset Memori", font=("Segoe UI",
12, "bold"), fg_color="#    ", hover_    r="#c82333", height=35,
command=self.reset_memori_    ai)
        self.btn_wina    reset.p    ck(fi    ", padx=10, pady=(5, 10))

        # ====    ==    =    N FRAME OUTPUT (KANAN) ================
        self.f    e_    i_o    tk.CTkFrame(self.main_frame, fg_color="#050510", corner_radius=10)

        he    der_w    = c    .CTkFrame(self.frame_winai_out, fg_color="#181818", corner_radius=8)
        .    wina    ack(fill="x", padx=15, pady=10)

        ctk.C    bel(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="#2E7D32", 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 V7.2.\n"
            "Terhubung 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']) > 3000: 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 * yang tersisa dan cetak 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 ini memproses hitungan terlebih dahulu, lalu mengirim datanya ke AI untuk
diinterpretasikan."""
        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 Interna...",
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" 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_konteks = ""

        if 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("Mengolah 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_ke
            api_key = self.winai_api_k[idx]
            api_url =
f"https://generativelangua.google.com/v1beta/models/{self.winai_primary_model}:generateContent?key={api_key}"

            try:
                res = requ.p(api_u, headers={'Content-Type': 'application/json'}, json=payload, timeout=20)
                if s. _co == 200:
                    = .json()
                    jaw = data['candidates'][0]['content']['parts'][0]['text']
                    with o (self.winai_tracker_file, 'w') as f:
                        te(str(idx))
                    ukses_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 m[illegible]an kalkulasi untuk
Anda:\n\n```text\n{report_tambahan}\n```"

        if sukses_dijawab:
            # --- INJEKSI OTOMATIS LAPORAN MURNI DI BELAKANG JAWAB[illegible]---
            if report_tambahan:
                jawaban += f"\n\n[Laporan Asli Sistem]:\n```tex[illegible]port_tambahan}\n```"

            if len(jawaban) < 2000:
                self.winai_memori += f"User: {teks_b[illegible]}\n[illegible]awaban}\n\n"
            else:
                self.winai_memori += f"User: {teks_[illegible]}\n[illegible][Telah memberikan jawaban
panjang/dokumen]\n\n"

            if len(self.winai_memor[illegible]0:
                self.winai_memor[illegible]self.wi[illegible]memori[-15000:]

        self.after(0, self.selesai_[illegible]proses_winai, jawaban)

    def selesai_mem[illegible]ses_winai(self, jawaban):
        self.hap[illegible]att[illegible]adi_winai()
        self.[illegible]kan_teks_w[illegible]jawaban, pengirim="AI")
        self.bt[illegible]nai_[illegible].configure(state="normal")

    def teks_winai(self):
        t[illegible]self.winai_output_box.get("1.0", "end-1c").strip()
        if no[illegible]s_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 generate_first_point_report(self, y, m, d):
        """Fungsi ekstraksi teks murni dari Menu 5 tanpa memunculkan peta GUI"""
        waktu_tuple = (y, m, d, 12, 0, 0)

        # --- PERBAIKAN: Pastikan Ephemeris dimuat sebelum diakses oleh Skyfield ---
        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", "Ma", "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 = 1
            pengamat.temp = 2
            pengamat.date = wa

            try: waktu_s = peng.next_setting(matahari)
            except: continue

            peng = waktu_sunset
            tah (pengamat)
            bu (pengamat)

            # Geo (DISELELARASKAN DENGAN MENU 2 - SKYFIELD)
            imp pytz
            nset_sky =
self.ts.fr 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! 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 = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]
         tgl_lokal = f"{int(d_l):02d} {nama_bulan[int(m_l)]} {format_tahun_aman(int(y_l))}"
         jam_lokal = 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()
         ijtimak_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}"

         batas_00_utc_ephem = 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
```

```
        if pt['sunset_utc'] < batas_00_utc_ephem:
            status_ijtimak = f"Kriteria PKG 1 Terpenuhi (Terpenuhi sblm {bt_jam} UTC)\n» Kesimpulan:
BESOK masuk bulan baru"
        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:
                    status_ijtimak = "Kriteria PKG 2 Terpenuhi (Amerika & Ijtima < Fajar Gisb.)\n»
Kesimpulan: BESOK masuk bulan baru"
                else:
                    alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima > Terbit Fajar"
                    status_ijtimak = f"Kriteria PKG 2 Terpenuhi ({alasan})\n» Kesimpulan: LUSA
masuk awal bulan"
            except:
                fajar_info = " Fajar Gisb.  : Error (N/A)\n"
                status_ijtimak = "Kriteria PKG 2 Terpenuhi (Gagal Menghitung Fajar)\n»
Kesimpulan: LUSA masuk awal bulan"

        report += (
            f"[{pt['label'].upper()}]\n"
            f"{'-'*45}\n"
            f"Lokasi : {pt['kota']}, {pt['negara']}\n"
            f"Tanggal : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
            f"Koordinat : {pt['lat_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
            f"{'-'*45}\n"
            f"Parameter Waktu (UTC):\n"
            f"Sunset 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 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, h ght=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=1 mand=self._update_sim_frame)
        self.sim_slider.pack(fill="x", pady=5)
        self.sim_slider.set(0)

        btn_play = ctk.CTkButton(ctrl_fra t, te ause Animasi", fg_color="#1565C0",
hover_color="#0D47A1", command=se oggl sim_ ay)
        btn_play.pack(pady=5)

        self.sim_data = []
        self.sim_is_playing

        # Mulai Kalkul si di Ba kgro thread (Kirim wilayah_global juga)
        threading.Thr (ta self._kalkulasi_data_simulator, args=(waktu_puncak_str, objek,
wilayah_glo daem Tr ).start()

    def kalk data simulator(self, waktu_puncak_str, objek, wilayah_global):
        try:
            f skyfie api import wgs84
            W ytz.timezone('Asia/Jakarta')
            dt e = datetime.datetime.strptime(waktu_puncak_str, "%d-%m-%Y %H:%M")
            dt_ ib = 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 = 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 = 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['sep'] < 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 mana pun dia terjadi di bumi.
        if not valid_frames:
            sim_frames = generate_frames(Nama_Lokasi_Pusat_Bumi, 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 simulasi.", text_color="#FF5252"))
            return

        # Ambil frame dengan obs tertinggi
        peak_frame = max(valid_frames, key=lambda x: x['obs'])

        # Ambil 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':
            scale = 120
            sun_px = frame['sd_sun'] * scale
            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[
2f}%\nAlt
Matahari: {frame['alt_matahari']:.1f}°"
        canvas.create_text(15, 20, text=info_teks, fill="#00E676", font=("Consolas", 12, "b
d"),
anchor="nw")

    else:
        # RENDER GERHANA BULAN
        scale = 130 / frame['r_umbra'] # Jari-jari Umbra dipatok
        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, 5
        canvas.create_line(0, cy, 5
        # Gambar Penumbra (
        canvas.create_o
fill="#15151A", outli
        canvas.crea
font=("Consolas",

        ca
fill="#
        canvas.
fo

        #
        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 animasi


    def _get_islamic_event(self, h_day, h_month, weekday):
        """
        Mendeteksi hari libur Islam dan jadwal puasa sunnah.
        h_month: 1-12 (Muharram - 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 Mu
        if h_day in [13, 14, 15] and h_month != 9:
            events.append("Puasa Ayyamul Bidh
            if not color: color = "#00B0FF" # B  M  Cya

        # Cek Puasa Senin & Kamis
        if weekday in [0, 3] and h  onth !  # 0 = Senin, 3 = Kamis
            nama_hari = "Senin" i  day ==  Kamis"
            events.append(f"P asa Sun  {nama_hari}")
            if not color: col  FFA000"  nge

    # Set warna p ingatan bias  tidak ada puasa/haram
    if events and  co
        color  504  #

    t ltip  =  join(events) if events else ""
    re urn co  po  _text

de  bi  izwala(self):
    try:
        yea  = 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 Hari Ini
        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 : {tgl:02d}/{bln:02d}/{format_tahun_aman(year)}")
        output_lines.append(f"* Lokasi : {lokasi_text}")
        output_lines.append(f"* Tinggi 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 Asar : {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 = tz_aris.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 [illegible] dhuha_marked:
                  marker = "  << MASUK DHUHA"
                  dhuha_marked = True

               # 2. Dzuhur
               elif curr_dt >= noon_dt and no[illegible]uhur_marked:
                  marker = "  << ZA[illegible]DZUHUR"
                  dzuhur_marked = True

               # 3. Ashar
               elif curr_dt > noon_dt and shadow_len >= target_shadow_asr and not ashar_marked:
                  mark[illegible]"  MASUK ASHAR"
                  ar_[illegible]ed[illegible]ue

               [illegible].append(f"{waktu_str:<12} | {alt_str:<14} | {arah_str:<15} |
{panjang_str}{marker}")

               [illegible] = 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 = 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_live_mode = is_live_mode

        def toggle_live():
            self.miz_live_mode = 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 merusak kalender
                jam_safe = min(23, max(0, jam))

                base_local = datetime.date(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.degrees:.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="#[illegible]", 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="#FFD700",
        button_color="#FFD700"
    )
    slider.pack(fill="x", padx=20, pady=(10, 20))

    # ---> ANIMASI REAL-TIME (TICK PER DETIK) <---
    def update_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 menekan tombol "LIVE" atau membiarkannya kosong
        now = datetime.now()
        jam_sekarang_desimal = now.hour + (now.minute / 60.0) + (now.second / 3600.0)
        if (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 da un
{start_year} H hingga {end_year} H...\nProses ini memerlukan waktu pemrosesan C U, h on
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", "  ", "Jul", "Aug", "Sep", "Oct",
"Nov", "Dec"]

        try:
            for y_h in range(start_year, end_year + 1):
                # ==========
                # PERBAIKAN: Hitung estimasi tahun  ehi  terasi (bukan di luar loop).
                # Auto-Switch akan otomatis pind  fil  al k  42.bsp atau de406.bsp)
                # saat tahun melewati batas 2 53,  ga  ak terjadi EphemerisRangeError.
                # ==========
                approx_greg_curr = int(y_h  0.9  24 + 622.54)
                self.auto_switch_eph  approx  _curr)

                year_data = []
                for m_h in r ge(1, 1
                    # 1. Ca  TT Ijtimak bu  ini
                    approx  cu  elf.ge  approx_nm_tt(y_h, m_h)
                    n  t_  s  t_new_moons_in_range(approx_tt_curr - 5.0, approx_tt_curr + 5.0)
                    no  lis  raise ValueError(f"Fase Bulan Baru tidak ditemukan untuk {y_h}-
{m_h}")
                    nm  r = n_list_curr[0]
                    tart_  urr = 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 Secara 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 terupdate secara Live."))
        self.after(0, lambda: self.lbl_status.configure(text="Auto-Builder Selesai",
text_color="#00FF66"))

      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(Sec... Fisis)', ha='center', va='center', fontsize=12,
fontweight='bold', color='#FF5252')
        self.ax_kb.text(8.5, 2.5, 'TIDAK TER... (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, 'RELATIF BIS... MI', 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_hilal = 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=1

        # Status Box di bawah grafik
        status_frame = ctk.CTkFrame(self.frame_kriteria_batas_out, fg_color="
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="STA
"bold"))
        self.kb_status_label.pack(pady=(10, 5))

        self.kb_alasan_label = ctk.CTkLabel(status_fra
justify="center", wraplength=700)
        self.kb_alasan_label.pack(pady=(0, 10), pa

    def update_kriteria_batas_plot(self):
        """Memperbarui grafik dan pen
        if not hasattr(self, 'ax_kb'): r

        x = self.var_kb_elonga
        y = self.var_kb_keti

        self.titik_hilal_
        self.titik_hilal_

        # --- MODIFIKASI: Logika Isi Angka & Posisi Kotak Info <---
        # Logika
        # Teks dita
        offset_y = -1
        self.
        self.
        # ----------------------------------------------------------

        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 kemalakteraturan
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 Pasti untak!)"
            color_msg = '#00E676'
            penjelasan_msg = ("ALASAN: Memenuhi kriteria global Istanbul 2016. Pada posisi ini, hilal
memiliki elongasi\n"
                              "yang cukup untuk membentuk sabit tipis dan ketinggian yang cukup untuk
mengalahkan bias\n"
                              "cahaya senja. Secara sains, ini adalah (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_baru(self):
        """Menghitung ketinggian dan elongasi real-time berdasarkan input di Menu 31"""
        try:
            y = int(self.entry_kb_tahun.get())
            m = int(self.entry_kb_bulan.get())
            d = int(self.entry_kb_tanggal.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_moon_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()}")

if __name__ == "__main__":
    app = KHGTApp()
    app.mainloop()
```

DAFTAR PUSTAKA

A. Literatur Fikih, KHGT, dan Astronomi Islam

Buku dan jurnal di bawah ini menjadi landasan teoretis dari algoritma visibilitas hilal, parameter KHGT, MABIMS, serta kalkulasi waktu salat dan arah kiblat.

- Anwar, S. (2022). *Fikih Kalender Hijriah Global: Membangun Peradaban Waktu Umat Islam*. Suara Muhammadiyah. (Landasan teoretis untuk implementasi parameter PKG 1 dan PKG 2).
- Djamaluddin, T. (2011). *Astronomi Islam dan Astronomi Observasional*. Departemen Astronomi ITB. (Rujukan parameter visibilitas hilal Neo MABIMS: Alt ≥ 3° Elongasi ≥ 6.4°).
- Ilyas, M. (1994). *Islamic Astronomy and Science Development: Glorious Past, Challenging Future*. Pelanduk Publications.
- International Astronomical Center (IAC). (2016). *Resolusi Konferensi Internasional Penyatuan Kalender Hijriah di Istanbul 2016*. (Rujukan utama syarat KHGT: Alt ≥ 5° dan Elongasi ≥ 8°).
- Majelis Tarjih dan Tajdid PP Muhammadiyah (Ed.). *Pedoman Hisab Muhammadiyah*. (Referensi metode perhitungan fajar 18° dan isya 18°, serta rashdul kiblat).
- Odeh, M. S. (2006). *New Criterion for Lunar Crescent Visibility*. Experimental Astronomy, 18(1-3), 39-64. (Referensi untuk `odeh_criterion` pada *Crescent Visibility HD Map Scanner*).

B. Ephemeris, Geodesi, dan Algoritma Komputasi Fundamental

Referensi yang mendasari formula perhitungan benda langit, jarak, refraksi atmosfer, dan model bumi WGS84.

- Folkner, W. M., Williams, J. G., Boggs, D. H., Park, R. S., & Kuchynka, P. (2014). *The Planetary and Lunar Ephemerides DE430 and DE431*. Interplanetary Network Progress Report 42-196. NASA Jet Propulsion Laboratory. (Basis data *Development Ephemeris* versi `de421.bsp`, `de406.bsp`).
- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell. (Rujukan algoritma Ardhana, interpolasi, dan fase bulan yang diimplementasikan di balik library PyEphem).
- National Imagery and Mapping Agency (NIMA). (2000). *Department of Defense World Geodetic System 1984 (WGS 84): Its Definition and Relationships with Local Geodetic Systems*. NIMA TR8350.2. (Digunakan dalam objek `wgs84.latlon` pada kalkulasi toposentrik).
- Seidelmann, P. K. (Ed.). (1992). *Explanatory Supplement to the Astronomical Almanac*. University Science Books.

C. Sains Data & Pustaka Perangkat Lunak (Python Libraries)

Dokumentasi teknis untuk eksekusi komputasi matriks, rendering 3D, GUI, dan kecerdasan buatan (WinAI).

- Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90-95. (Digunakan untuk *Altitude Chart Analyser* dan *Simulasi Ephemeris 3D*).
- Rhodes, B. (n.d.). *Skyfield: Elegant Astronomy for Python*. Diakses dari https://rhodesmill.org/skyfield/ (Mesin utama kalkulasi *apparent topocentric* dan *geocentric*).
- Rhodes, B. (n.d.). *PyEphem: Astronomical Ephemeris for Python*. Diakses dari https://rhodesmill.org/pyephem/ (Mesin sekunder berkinerja tinggi untuk iterasi *Global Hilal Analyzer*).
- Virtanen, P., et al. (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, 261-272. (Digunakan khusus untuk modul `scipy.interpolate.griddata` pada peta visibilitas *bicubic*).
- Harris, C. R., et al. (2020). *Array Programming with NumPy*. Nature, 585, 357-362. (Penanganan data *array* pada kalkulasi spasial HD).
- Clark, A. (2015). *Pillow (PIL Fork) Documentation*. Diakses dari https://python-pillow.org/ (Engine pembuat gambar kalender Masehi/Hijriah resolusi tinggi).
- Schlingensiepen, T. (n.d.). *CustomTkinter: A modern and customizable python UI-library based on Tkinter*. (Infrastruktur antarmuka GUI *Dark Mode*).
- Google. (n.d.). *Gemini API Documentation*. Diakses dari https://ai.google.dev/ (Infrastruktur untuk fitur WinAI - Asisten Filsafah AI Terintegrasi).

GLOSARIUM A–Z: KHGT TIMES V7.2

A

- **Altitude (Ketinggian):** Jarak sudut suatu benda langit diukur dari ufuk (horizon) pengamat. Bernilai positif jika di atas ufuk, dan negatif jika di bawah ufuk.
- **Astrometri:** Cabang astronomi yang fokus pada pengukuran presisi posisi dan pergerakan benda-benda langit, yang menjadi dasar logika library *Skyfield* dan *PyEphem*.
- **Azimuth (Azimut):** Arah sudut horizontal benda langit yang diukur searah jarum jam dari titik Utara sejati (0°) menyusuri ufuk.

B

- **Barycenter:** Titik pusat massa dari sebuah sistem benda langit (misal: tata surya atau sistem Bumi-Bulan) yang saling mengorbit. JPL Ephemeris menggunakan referensi ini untuk kalkulasi orbit presisi.
- **Bicubic Interpolation:** Algoritma matematika dari library `SciPy` (metode `cubic`) yang digunakan program untuk mengubah titik-titik data kasar (grid 5°) menjadi peta visibilitas hilal lengkung beresolusi tinggi (Full-HD).

C

- **CustomTkinter:** Pustaka (library) antarmuka pengguna grafis (GUI) pada Python yang digunakan untuk membangun *dashboard* KHGT Times dengan gaya modern dan mode gelap (*dark-mode*).
- **Crescent (Hilal):** Fase bulan sabit pertama yang terlihat setelah terjadinya ijtimak (konjungsi).

D

- **Deklinasi (Deklinasi):** Salah satu dari dua koordinat ekuatorial. Mengukur jarak sudut benda langit di sebelah utara atau selatan ekuator langit (setara dengan Lintang pada bola bumi).
- **Development Ephemeris (DE):** Model numerik terkomputerisasi dari tata surya yang dikembangkan oleh NASA Jet Propulsion Laboratory (JPL), seperti file `de421.bsp` atau `de440.bsp` yang dibaca oleh sistem.

E

- **Elongasi (Elongation):** Jarak sudut busur antara pusat piringan Matahari dan pusat piringan Bulan jika diamati dari Bumi. KHGT mensyaratkan elongasi geosentris minimal 8°.
- **Ephemeris:** Tabel, *database*, atau file biner berisi sekumpulan nilai kalkulasi matematis yang memberikan posisi lintasan benda-benda langit pada waktu-waktu tertentu secara sangat presisi.

F

- **Fajar Astronomis (*Astronomical Twilight*):** Batas waktu saat pusat geometri Matahari berada 18° hingga 20° (tergantung kriteria) di bawah ufuk timur, yang menjadi penanda masuknya waktu salat Subuh.

G

- **Geosentris (*Geocentric*):** Sistem referensi koordinat astronomi yang menempatkan pusat massa Bumi sebagai titik pusat pengamatan. Ini adalah parameter wajib (mutlak) untuk pengujian kriteria KHGT.
- **Gisborne:** Kota di Selandia Baru yang berada dekat dengan garis Batas Perubahan Tanggalan Internasional (IDL). Fajar di kota ini digunakan sebagai titik uji krusial pada algoritma PKG 2 KHGT.

H

- **Heatmap:** Representasi visual dua dimensi di mana nilai data (seperti elevasi dan elongasi pada modul *Crescent Visibility 3D*) direpresentasikan menggunakan gradasi warna (seperti *Plasma* atau *Inferno*).

I

- **Ijtimak (Konjungsi):** Peristiwa ketika bujur ekliptika Bulan dan Matahari sama jika diukur dari Bumi. Ini adalah titik nol (awal) dari siklus fase bulan.
- **Interseksi:** Area pertemuan (irisan) geografis pada peta visibilitas di mana kedua syarat visibilitas (ketinggian ≥ 3° dan elongasi ≥ 8°) terpenuhi secara bersamaan.

J

- **Julian Day (JD):** Format waktu standar astronomi yang menghitung jumlah hari dan pecahan hari yang telah berlalu sejak 1 Januari 4713 SM pukul 12:00 UT. Sangat krusial untuk menghindari *bug* (crash) komputasi pada Python saat menghitung tahun Sebelum Masehi.

K

- **KHGT (*Kalender Hijriah Global Tunggal*):** Konsep penyatuan kalender Islam internasional berlandaskan kesepakatan Muktamar Turki 2016 dengan prinsip satu hari satu tanggal di seluruh belahan bumi.
- **Kulminasi (Transit / Zawal):** Peristiwa ketika sebuah benda langit (seperti Matahari) melintasi meridian lokal pengamat dan mencapai titik ketinggian maksimumnya (penanda masuknya waktu Zuhur).

L

- **Lintang (*Latitude*):** Garis khayal horizontal yang mengukur jarak sudut suatu tempat di utara atau selatan garis khatulistiwa Bumi.

M

- **MABIMS:** Singkatan dari Menteri Agama Brunei, Indonesia, Malaysia, dan Singapura. "Neo MABIMS" adalah kriteria visibilitas regional (Toposentris) dengan syarat Altitude ≥ 3° dan Elongasi ≥ 6.4°.
- **Matplotlib:** Library Python terkemuka yang digunakan dalam kode ini untuk merender kurva grafik harian (*Altitude Chart*) dan simulasi tata surya 3D.

N

- **Nadir:** Titik di bola langit yang berada tepat di bawah pengamat, berlawanan dengan titik Zenit.
- **NumPy:** Pustaka dasar Python untuk komputasi ilmiah yang memproses array multi-dimensi secara masif, sangat digunakan dalam iterasi *brute force* pada modul 50 Tahun.

O

- **Odeh Criterion:** Kriteria visibilitas hilal astronomis modern (dikembangkan oleh M. Shawkat Odeh) yang memperhitungkan faktor cahaya langit dan topografi, digunakan sebagai opsi pembanding di dalam program.

P

- **PKG (*Parameter Kriteria Ka'bah*):** Mekanisme algoritma evaluasi KHGT. PKG 1 menguji apakah *sunset* di daerah yang memenuhi kriteria terjadi sebelum pukul 00:00 UTC. PKG 2 menguji kriteria terpenuhi parameter di Benua Amerika terhadap batas Fajar Gisborne.
- **PyEphem:** Pustaka astronomi *wrapper* berbasis C (*XEphem*) di dalam program yang difungsikan khususnya sebagai akselerator untuk pelacakan hilal di ratusan kota dalam hitungan detik.

Q

- **Qiblat (Kiblat):** Arah menuju Ka'bah di Makkah. Modul program menggunakan rumus trigonometri bola (*Spherical Trigonometry*) untuk menghitung azimut pasti dari mana pun di Bumi.

R

- **Rashdul Qiblah:** Peristiwa astronomis lokal atau global ketika bayangan benda yang tegak lurus mengarah persis ke arah kiblat, biasanya dihitung saat Matahari berada di lintasan azimut kiblat tersebut.
- **Refraksi Atmosfer:** Pembelokan (pembiasan) cahaya benda langit oleh atmosfer Bumi yang menyebabkan posisi semu (*apparent*) benda langit terlihat lebih tinggi dari posisi sebenarnya secara geometris.

S

- **Skyfield:** Engine (mesin) astronomi Python presisi tinggi yang digunakan dalam KHGT Times V7.2 untuk menghasilkan tingkat akurasi komputasi setara dengan US Naval Observatory.
- **Solstice (Titik Balik):** Momen ketika Matahari mencapai titik paling utara atau paling selatan relatif terhadap ekuator angkasa, penanda awal musim panas atau musim dingin.

T

- **Toposentris (*Topocentric*):** Sistem referensi koordinat astronomi yang pusat pengamatannya berada murni di titik pengamat berdiri di permukaan Bumi (mempertimbangkan garis lintang, bujur, dan elevasi). Ini adalah referensi utama untuk kriteria Rukyat (MABIMS).

U

- **Ufuk (Horizon):** Garis batas nyata atau matematis tempat bertemunya proyeksi langit dan bumi (0°).
- **Umbra:** Bagian bayangan terdalam dan tergelap yang dilewati oleh pengamat saat terjadi peristiwa gerhana matahari total.

V

- **Visibilitas (*Visibility*):** Status tingkat probabilitas (kemungkinan) keterlihatan hilal jika diamati oleh mata manusia atau alat optik, diuji menggunakan komparasi matematis antara posisi Matahari dan Bulan.

W

- **WGS84 (*World Geodetic System 1984*):** Standar sistem koordinat, datum permukaan, dan model referensi ellipsoida Bumi yang digunakan program untuk mengkalkulasi jarak dan resiprokal pengamat secara akurat.
- **Wi[illegible]** Model terintegrasi di dalam KHGT Times V7.2 yang menanamkan teknologi *Artificial Intelligence* (AI) berbasis Gemini sebagai asisten analisis data *real-time* dan mesin tanya jawab fikih.

X

- **X/Y/Z Axes (Sumbu X/Y/Z):** Vektor koordinat tiga dimensi. Sumbu ini dikalkulasikan menggunakan fungsi transformasi matriks (trigonometri) untuk memproyeksikan orbit tata surya 3D Geocentric ke dalam kanvas 2D layar monitor komputer.

Y

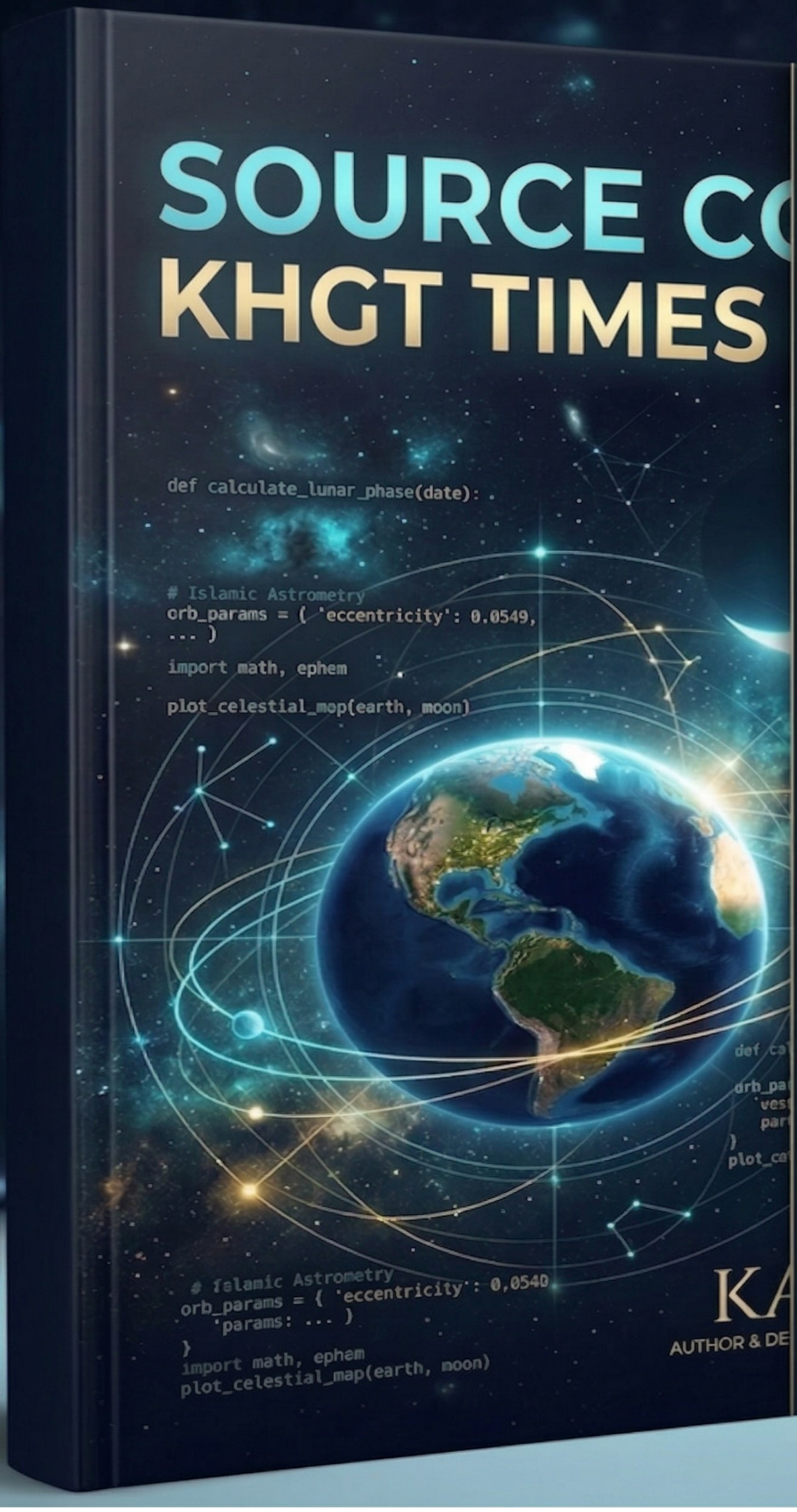
- **Y-Axis (Sumbu Y):** Sumbu vertikal pada antarmuka *Altitude Chart Analyser* Matplotlib yang difungsikan untuk merepresentasikan nilai derajat Ketinggian (Altitude) objek.

Z

- **Zawal:** Waktu ketika Matahari tergelincir atau melewati titik kulminasi atas (Meridian Transit). Momen ini adalah batas awal masuknya waktu salat Zuhur.
- **Zenith (Zenit):** Titik di bola langit yang berada tepat dan vertikal (90°) di atas kepala pengamat.


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.

SOURCE CO
KHGT TIMES



KA

AUTHOR & DE

SINOPSIS BUKU

Selamat datang di panduan mendalam tentang *Source Code* KHGT Times V7.2, perangkat lunak astrometri presisi tinggi generasi terbaru.

Buku ini adalah *Quantum Leap* komputasi astronomi lalam, menjembatani teori analitik klasik dengan integrasi numerik orbit real-time dari Development Ephemeris JPL NASA.

Pelajari teknik-teknik canggih seperti:

[1] **Integrasi Presisi Ephemeris NASA:** Menggunakan data orbit numerik real-time untuk akurasi tertinggi.

[2] **Pemindai Matlak Global:** Mesin Iteratif proaktif untuk memindai ratusan kota dalam hitungan detik.

[3] **Pemisahan Ruang:** Kalkulasi simultan geosentris dan toposentris untuk visibilitas hilal yang akurat.

[4] **Visualisasi Data HD:** Visualisasi visibilitas hilai HD dan lingkungan simulasi 3D interaktif.

Essential bagi pengambang, astronom, dan peneliti yang ingin melampaui standar pendahulu seperti Accurate Times. Buku ini adalah kunci untuk masi depan penanggaian Islam yang dipandu oleh teknologi modern.

KHGT PROJECT