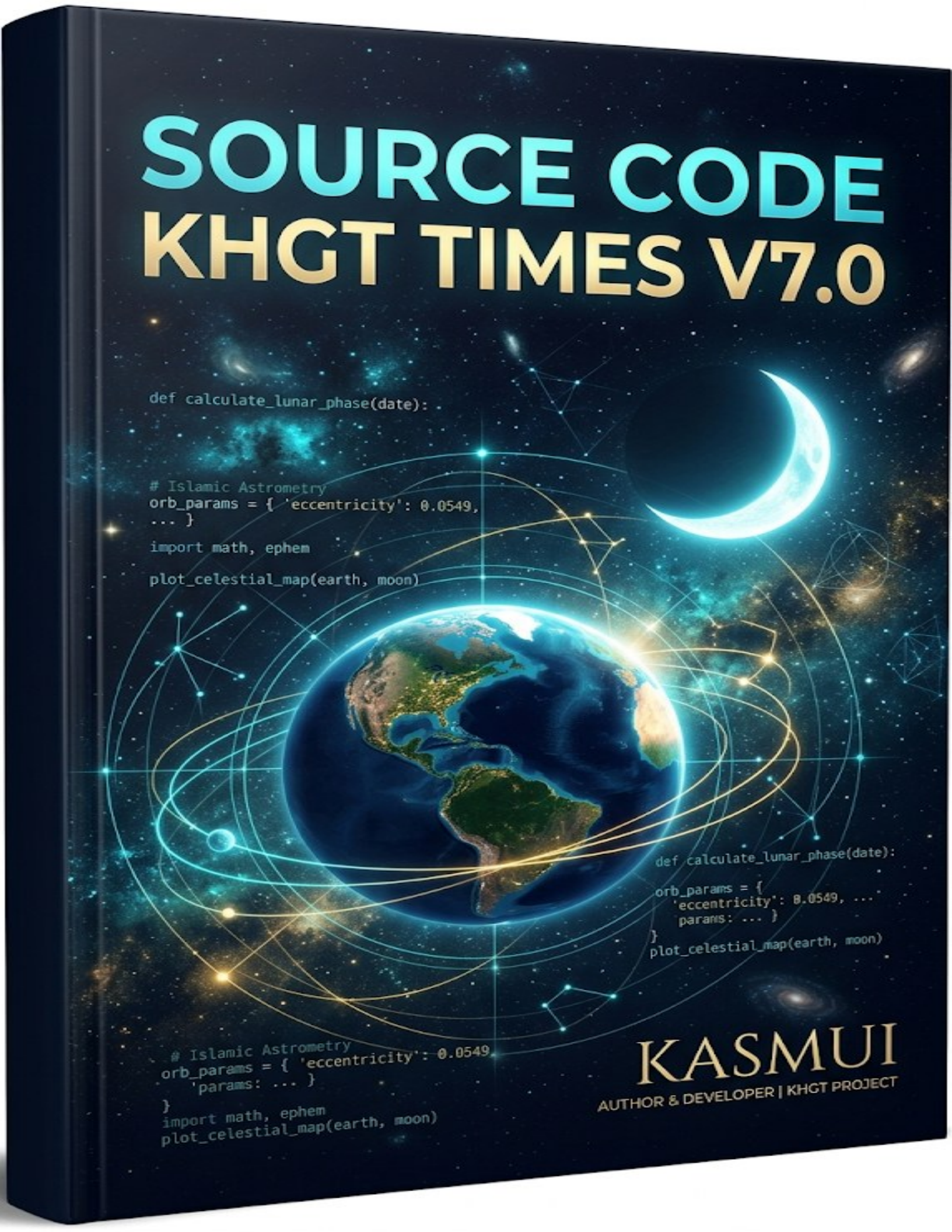
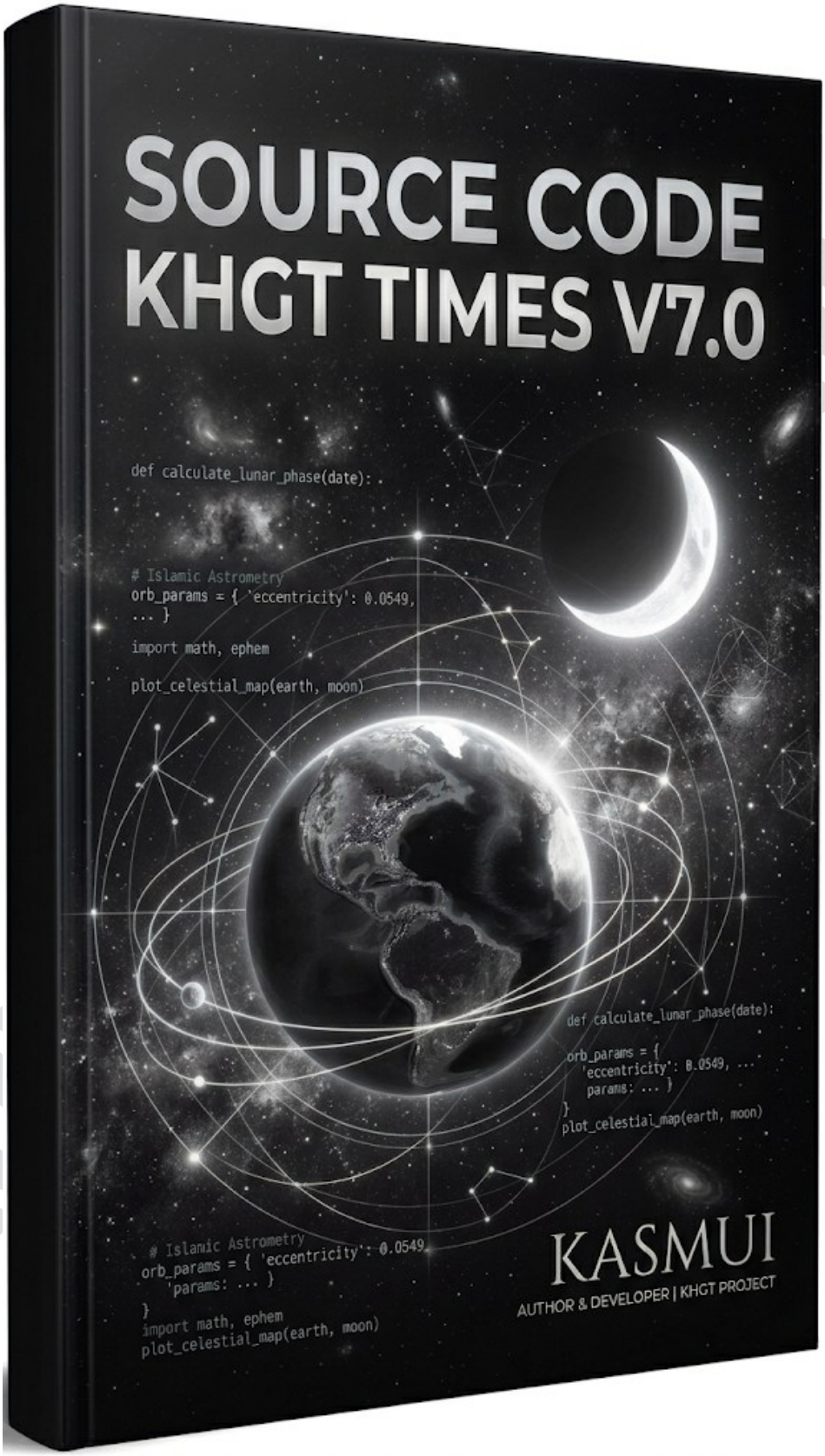
SOURCE CODE KHGT TIMES V7.0



KASMUI
AUTHOR & DEVELOPER | KHGT PROJECT


SOURCE CODE
KHGT TIMES V7.0
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 = {
'eccentricity': 0.0549, ...
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 Geosentris 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 V7.0"** 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 V7.0, sebuah ekosistem 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 menyelami anatomi dari 28 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 pengungkapan arsitektur WinAI, sebuah lompatan integrasi di mana model Kecerdasan Buatan (AI) dibekali akses 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, peneliti, astronom, dan pengambil kebijakan di berbagai lembaga ketarjihan pada era modern ini.

Akhir kata, semoga 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

ALGORITHM FLOW) *SOURCE CODE* KHGT TIMES V7.0

Berikut adalah alur algoritma (Algorithm Flow) dari *source code* KHGT Times V7.0. Karena kode ini adalah aplikasi GUI (*Graphical User Interface*) skala besar dengan 28 modul, alur algoritmanya dirancang secara modular dan berbasis *Event-Driven* dengan eksekusi *Multi-Threading*.

Secara garis besar, siklus hidup aplikasi (*Application Lifecycle*) berjalan melalui 7 fase utama:

Fase 1: Booting & Inisialisasi Sistem (Startup)

1. **Membangun GUI:** Menjalankan `KHGTApp` yang mewarisi `customtkinter.CTk`. Membangun tata letak (*layout*) Sidebar dan Area Utama.
2. **Memuat Dependensi Astronomi:** Menginisialisasi objek `Loader` dari library Skyfield untuk memuat *Timescale* (`self.ts`).
3. **Threading Ephemeris:** Memulai *background thread* `load_ephemeris()` untuk memeriksa dan mengunduh (jika belum ada) file *Development Ephemeris* (misal: `de421.bsp`) secara otomatis tanpa membekukan layar.
4. **Menjalankan Loop Latar Belakang:** * Mengaktifkan `alarm_tick()` yang berjalan setiap 1 detik untuk memantau jadwal salat dan *countdown*.
    - Mengaktifkan `eph3d_live_update_loop()` untuk memantau waktu secara *real-time* bagi kanvas 3D.
    - Menyiapkan koneksi lokal ke database WinAI (`load_winai_databases()`).

Fase 2: Interaksi & Routing Antarmuka (User Input)

1. **Pemilihan Modul:** Pengguna memilih salah satu dari 28 fitur melalui *dropdown* `combo_mode`.
2. **Switching Mode:** Fungsi `switch_mode(mode)` dipanggil. Sistem akan menyembunyikan semua parameter input dan kanvas *output* yang tidak relevan, lalu hanya menampilkan antarmuka yang spesifik untuk modul terpilih (misal: input untuk "Visibilitas Hilal").
3. **Pengisian Parameter:** Pengguna memasukkan data seperti Tahun, Bulan, Tanggal, Koordinat (Lat, Lon, Elevasi), dan Zona Waktu. Fungsi *Auto-Detect* dapat dipanggil untuk mencari titik GPS dan mengatur Zona Waktu secara dinamis (`get_tz_from_lon`).

Fase 3: Pemicu Kalkulasi (Triggering)

1. Pengguna menekan tombol biru **"▶ PROSES DATA"**.
2. Fungsi `run_calculation()` dijalankan.
3. Tombol dinonaktifkan sementara (*disabled*) dan teks status berubah menjadi "Menghitung..." untuk mencegah *spam click*.

4. Sistem membaca mode yang sedang aktif, lalu memutar (*routing*) perintah ke fungsi spesifik menggunakan **Multi-Threading**. Contoh:
    - Jika mode = "Visibility Hilal", jalankan `threading.Thread(target=self.calculate_visibility)`.
    - Jika mode = "Analisis Gerhana", jalankan `threading.Thread(target=self._calc_gerhana_thread)`.

Fase 4: Engine Astrometri & Manajemen Ephemeris

Di dalam setiap *thread* kalkulasi, sistem melakukan pengkondisian engine:

1. **Auto-Switch Ephemeris:** Memanggil `auto_switch_ephemeris(year)`. Algoritma akan memeriksa tahun *input*. Jika tahun jauh di masa lalu (misal -3000 SM) atau masa depan jauh, sistem memuat `de406.bsp`. Jika era modern, memuat `de421.bsp` atau `de442.bsp`.
2. **Penentuan Objek Waktu:** Mengubah input tanggal dan zona waktu pengguna menjadi objek `Time` Skyfield berformat Julian Date (JD) atau Terrestrial Time (TT) agar kebal terhadap *error* tahun minus/BCE.
3. **Instansiasi Objek:** Mengambil data planet dari *file* ephemeris (misal: `self.eph['earth']`, `self.eph['sun']`, `self.eph['moon']`).

Fase 5: Komputasi Matematika Inti (Core Logic)

Bagian ini bergantung pada modul yang dipanggil. Sebagai representasi, berikut adalah alur logika untuk **Modul 1 (Visibilitas Hilal / KHGT)**:

1. **Mencari Sunset:** Algoritma mencari waktu kapan Matahari terbenam (*Sunset*) pada tanggal dan koordinat yang diinput pengguna.
2. **Kalkulasi Geosentris (Untuk Syarat KHGT):**
    - Menempatkan pengamat di pusat Bumi secara matematis.
    - Menghitung Deklinasi, *Right Ascension*, dan Sudut Jam (*Hour Angle*) dari Matahari dan Bulan.
    - Menghasilkan **Altitude Geosentris** dan **Elongasi Geosentris**.
3. **Kalkulasi Toposentris (Untuk Syarat Rukyat / MABIMS):**
    - Menempatkan pengamat di permukaan Bumi (sesuai lat, lon, elevasi).
    - Memasukkan koreksi refraksi atmosfer (suhu dan tekanan).
    - Menghasilkan **Altitude Toposentris** dan **Elongasi Toposentris**.
4. **Evaluasi Kriteria Syariah:**
    - Jika *Alt Geo* >= 5° dan *Elong Geo* >= 8°, maka status = "Memenuhi KHGT".
    - Jika *Alt Topo* >= 3° dan *Elong Topo* >= 6.4°, maka status = "Memenuhi Neo MABIMS".

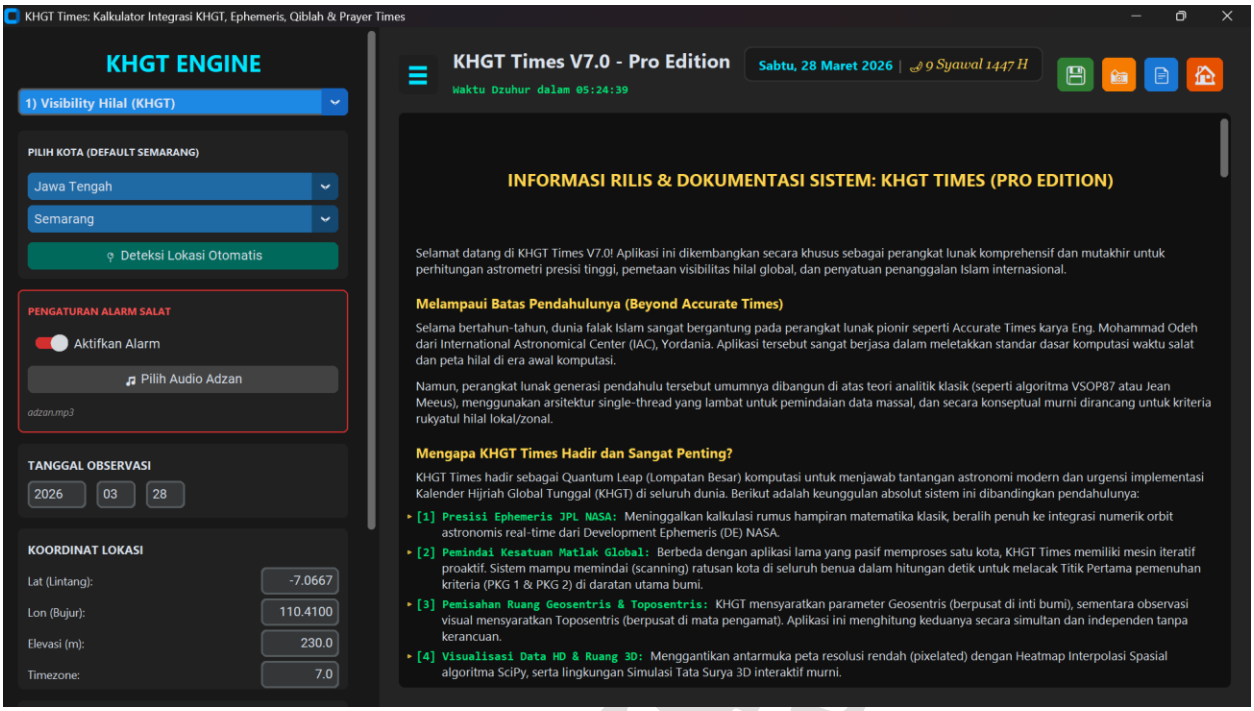
Fase 6: Render Output & Kembali ke Main Thread

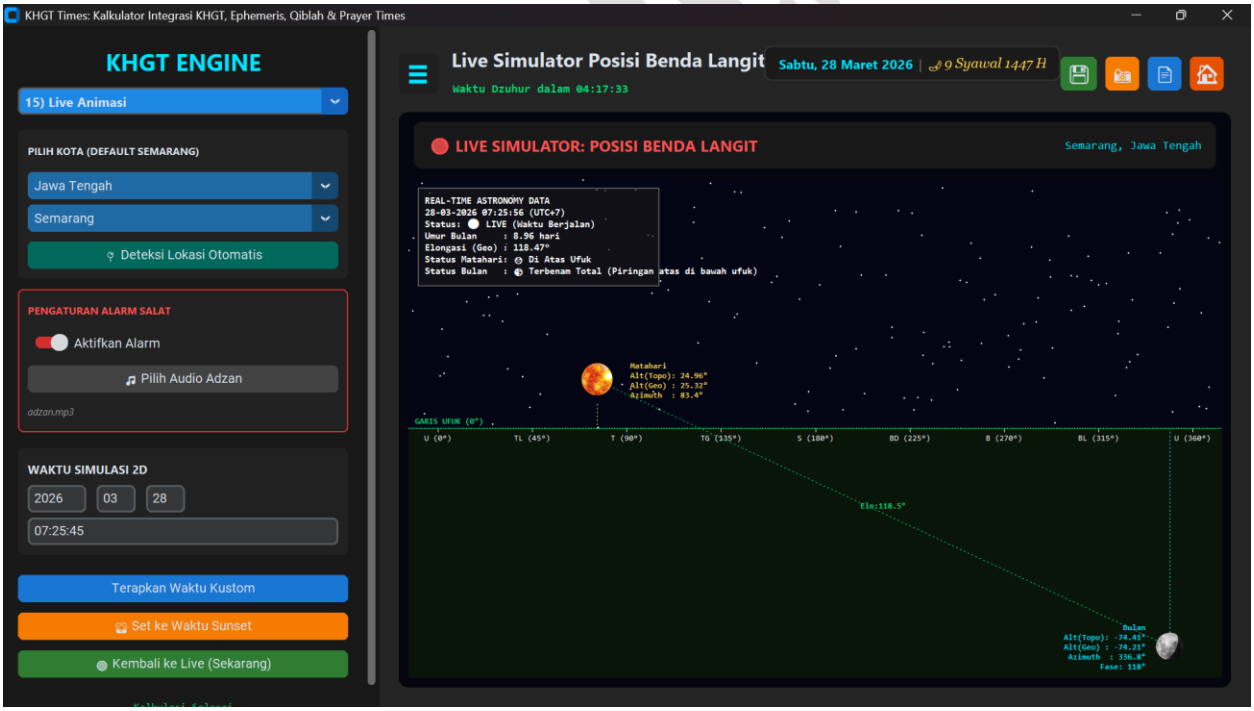
1. **Penyusunan Teks/Grafik:** Sistem merangkai hasil angka-angka astronomi (*float*) menjadi teks (*string*) laporan yang rapi (mengatur spasi, derajat, konversi waktu lokal). Jika berupa modul Peta/Grafik, sistem menyusun array NumPy dan merendernya di Matplotlib.
2. **Thread Saftey (`self.after`):** Karena proses dilakukan di *background thread*, aplikasi memanggil fungsi `self.after(0, ...)` untuk melempar hasil kembali ke GUI *Main Thread* tanpa mengalami *crash*.
3. Hasil ditampilkan di dalam `self.textbox` atau `FigureCanvasTkAgg` (untuk visual 3D/Peta). Tombol "PROSES DATA" diaktifkan kembali.

Fase 7: Ekspor & Pasca-Pemrosesan (Opsional)

Pengguna menggunakan panel tombol di kanan atas untuk menyimpan hasil:

- Fungsi `export_to_pdf()` mengubah teks ke PDF menggunakan `FPDF`.
- Fungsi `export_to_png()` merender teks ke dalam kanvas gambar murni menggunakan `Pillow` (PIL).
- Fungsi `export_kml_solar()` (pada gerhana) merender jalur umbra ke format vektor KML untuk Google Earth.


KHGT Times: Kalkulator Integrasi KHGT, Ephemeris, Qiblah & Prayer Times
KHGT ENGINE
1) Visibility Hilal (KHGT)
PILIH KOTA (DEFAULT SEMARANG)
Jawa Tengah
Semarang
Deteksi Lokasi Otomatis
PENGATURAN ALARM SALAT
Aktifkan Alarm
Pilih Audio Adzan
adzan.mp3
TANGGAL OBSERVASI
2026
03
28
KOORDINAT LOKASI
Lat (Lintang):
-7.0667
Lon (Bujur):
110.4100
Elevasi (m):
230.0
Timezone:
7.0
KHGT Times V7.0 - Pro Edition
Sabtu, 28 Maret 2026 | 9 Syawal 1447 H
Waktu Dzuhur dalam 05:24:39
INFORMASI RILIS & DOKUMENTASI SISTEM: KHGT TIMES (PRO EDITION)
Selamat datang di KHGT Times V7.0! Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak komprehensif dan mutakhir untuk perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal global, dan penyatuan penanggalan Islam internasional.
Melampaui Batas Pendahulunya (Beyond Accurate Times)
Selama bertahun-tahun, dunia falak Islam sangat bergantung pada perangkat lunak pionir seperti Accurate Times karya Eng. Mohammad Odeh dari International Astronomical Center (IAC), Yordania. Aplikasi tersebut sangat berjasa dalam meletakkan standar dasar komputasi waktu salat dan peta hilal di era awal komputasi.
Namun, perangkat lunak generasi pendahulu tersebut umumnya dibangun di atas teori analitik klasik (seperti algoritma VSOP87 atau Jean Meeus), menggunakan arsitektur single-thread yang lambat untuk pemindaian data massal, dan secara konseptual murni dirancang untuk kriteria rukyatul hilal lokal/zonal.
Mengapa KHGT Times Hadir dan Sangat Penting?
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. Berikut adalah keunggulan absolut sistem ini dibandingkan pendahulunya:
[1] Presisi Ephemeris JPL NASA: Meninggalkan kalkulasi rumus hampiran matematika klasik, beralih penuh ke integrasi numerik orbit astronomis real-time dari Development Ephemeris (DE) NASA.
[2] Pemindai Kesatuan Matlak Global: Berbeda dengan aplikasi lama yang pasif memproses satu kota, KHGT Times memiliki mesin iteratif proaktif. Sistem mampu memindai (scanning) ratusan kota di seluruh benua dalam hitungan detik untuk melacak Titik Pertama pemenuhan kriteria (PKG 1 & PKG 2) di daratan utama bumi.
[3] Pemisahan Ruang Geosentris & Toposentris: KHGT mensyaratkan parameter Geosentris (berpusat di inti bumi), sementara observasi visual mensyaratkan Toposentris (berpusat di mata pengamat). Aplikasi ini menghitung keduanya secara simultan dan independen tanpa kerancuan.
[4] Visualisasi Data HD & Ruang 3D: Menggantikan antarmuka peta resolusi rendah (pixelated) dengan Heatmap Interpolasi Spasial algoritma SciPy, serta lingkungan Simulasi Tata Surya 3D interaktif murni.


KHGT Times: Kalkulator Integrasi KHGT, Ephemeris, Qiblah & Prayer Times
KHGT ENGINE
15) Live Animasi
PILIH KOTA (DEFAULT SEMARANG)
Jawa Tengah
Semarang
Deteksi Lokasi Otomatis
PENGATURAN ALARM SALAT
Aktifkan Alarm
Pilih Audio Adzan
adzan.mp3
WAKTU SIMULASI 2D
2026
03
28
07:25:45
Terapkan Waktu Kustom
Set ke Waktu Sunset
Kembali ke Live (Sekarang)
Live Simulator Posisi Benda Langit
Sabtu, 28 Maret 2026 | 9 Syawal 1447 H
Waktu Dzuhur dalam 04:17:33
LIVE SIMULATOR: POSISI BENDA LANGIT
Semarang, Jawa Tengah
REAL-TIME ASTRONOMY DATA
28-03-2026 07:25:56 (UTC+7)
Status: LIVE (Waktu Berjalan)
Umur Bulan : 8.96 hari
Elongasi (Geo) : 118.47°
Status Matahari: Di Atas Ufuk
Status Bulan : Terbenam Total (Piringan atas di bawah ufuk)
Matahari
Alt(Topo): 24.96°
Alt(Geo) : 25.32°
Azimuth : 83.4°
GARIS UFUK (0°)
U (0°)
TL (45°)
T (90°)
TG (135°)
S (180°)
BD (225°)
B (270°)
BL (315°)
U (360°)
Ele:118.5°
Bulan
Alt(Topo): -74.41°
Alt(Geo) : -74.21°
Azimuth : 336.8°
Fase: 118°

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

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

import numpy as np
import pytz
import ephem

import warnings
warnings.filterwarnings("ignore")

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

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


# --- GUI LIBRARIES ---
import customtkinter as ctk
```

```
import tkinter as tk
from tkinter import messagebox, filedialog, ttk
from PIL import Image, ImageTk, ImageDraw, ImageFont
from CTkToolTip import * # --- SKYFIELD LIBRARIES ---
if not hasattr(CTkToolTip, 'block_update_dimensions_event'):
    setattr(CTkToolTip, 'block_update_dimensions_event', lambda self: None)
if not hasattr(CTkToolTip, 'unblock_update_dimensions_event'):
    setattr(CTkToolTip, 'unblock_update_dimensions_event', lambda self: None)
# -------------------------------------------------------
from skyfield.api import Loader, wgs84
from skyfield import almanac, eclipselib
from skyfield.positionlib import Geocentric

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

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

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

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

# ========================================
# PERBAIKAN CRASH GUI & SSL UNTUK EXE
# ========================================
if sys.stdout is None: sys.stdout = io.StringIO()
```

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

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

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

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

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

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

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

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

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

    bulan_angka = "03"
```

```
tahun = "2026"
koordinat_str = ""

import re
for line in lines:
    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)

    if "Long:" in line and "Lat:" in line:
        start_idx = line.find("Long:")
        end_idx = line.find(", Ele:")
        if end_idx != -1:
            koordinat_str = line[start_idx:end_idx].strip()
        else:
            koordinat_str = line[start_idx:].strip()

    if re.match(r'^\s*\d{2}/\d{2}/\d{4}', line):
        parts = [p for p in line.strip().split(' ') if p]

        if len(parts) >= 9:
            tanggal_full = parts[0]
            hari = str(int(tanggal_full.split('/')[0]))

            subuh = parts[1]
            terbit = parts[2]
            duha = parts[3]
            zuhur = parts[4]
            asar = parts[5]
            magrib = parts[6]
            isya = parts[7][:5]
            midnight = parts[8]

            data.append({
                'hari': hari,
                'subuh': subuh,
                'terbit': terbit,
                'duha': duha,
                'zuhur': zuhur,
                'asar': asar,
                'magrib': magrib,
                'isya': isya,
                'midnight': midnight
            })
```

```
    nama_bulan_upper = _util_get_nama_bulan(bulan_angka)

    # PERBAIKAN: Mengembalikan nilai bulan_angka agar bisa dipakai di fungsi generate
    return data, nama_bulan_upper, bulan_angka, tahun, koordinat_str

def _util_generate_image(raw_text, template_path, output_path):
    # PERBAIKAN: Menangkap parameter bulan_angka dari fungsi parse di atas
    jadwal_data, bulan_upper, bulan_angka, tahun, koordinat_str = _util_parse_jadwal_text(raw_text)

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

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

    draw = ImageDraw.Draw(img)

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

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

    start_y_judul = 120
    row_spacing_judul = 60
```

```
    start_y_header = 330
    row_spacing_header = 45
    start_y_data = 430
    row_spacing_data = 40

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

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

    if koordinat_str:
        teks_wilayah = f"UNTUK WILAYAH SEKITAR KOORDINAT {koordinat_str.upper()}"
    else:
        teks_wilayah = "UNTUK WILAYAH SEKITAR KOORDINAT (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)

        # --- PERBAIKAN FORMAT TANGGAL DI SINI ---
        # Sekarang menggunakan format DD/MM/YYYY murni tanpa nama teks bulan
        tanggal_str = f"{tanggal_hari}/{bulan_angka}/{tahun}"
        # ----------------------------------------

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

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

    img.save(output_path)
    return output_path

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

HIJRI_DB = {
    1446: [["Muharam", "Ahad Kliwon", "07-Jul-2024", 29], ["Safar", "Senin", "05-Aug-2024", 30],
["Rabiulawal", "Rabu", "04-Sep-2024", 30], ["Rabiulakhir", "Jumat", "04-Oct-2024", 30], ["Jumadilawal",
"Ahad", "03-Nov-2024", 29], ["Jumadilakhir", "Senin", "02-Dec-2024", 30], ["Rajab", "Rabu", "01-Jan-
2025", 30], ["Syakban", "Jumat", "31-Jan-2025", 29], ["Ramadan", "Sabtu", "01-Mar-2025", 30],
```

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

# (Memasukkan sisa HIJRI_DB tahun 1449-1497 agar converter tetap berjalan sempurna sesuai kode master Anda)
HIJRI_DB.update({
  1449: [["Muharam", "Ahad Wage", "06-Jun-2027", 29], ["Safar", "Senin", "05-Jul-2027", 29], ["Rabiulawal", "Selasa", "03-Aug-2027", 30], ["Rabiulakhir", "Kamis", "02-Sep-2027", 29], ["Jumadilawal", "Jumat", "01-Oct-2027", 30], ["Jumadilakhir", "Ahad", "31-Oct-2027", 29], ["Rajab", "Senin", "29-Nov-2027", 30], ["Syakban", "Rabu", "29-Dec-2027", 29], ["Ramadan", "Jumat", "28-Jan-2028", 30], ["Syawal", "Sabtu", "26-Feb-2028", 30], ["Zulkaidah", "Senin", "27-Mar-2028", 30], ["Zulhijah", "Rabu", "26-Apr-2028", 29]],
  1450: [["Muharam", "Kamis", "25-May-2028", 30], ["Safar", "Sabtu", "24-Jun-2028", 29], ["Rabiulawal", "Ahad", "23-Jul-2028", 29], ["Rabiulakhir", "Senin", "21-Aug-2028", 30], ["Jumadilawal", "Rabu", "20-Sep-2028", 29], ["Jumadilakhir", "Kamis", "19-Oct-2028", 30], ["Rajab", "Sabtu", "18-Nov-2028", 29], ["Syakban", "Ahad", "17-Dec-2028", 30], ["Ramadan", "Selasa", "16-Jan-2029", 29], ["Syawal", "Rabu", "14-Feb-2029", 30], ["Zulkaidah", "Jumat", "16-Mar-2029", 30], ["Zulhijah", "Ahad", "15-Apr-2029", 29]],
  1451: [["Muharam", "Senin", "14-May-2029", 30], ["Safar", "Rabu", "13-Jun-2029", 29], ["Rabiulawal", "Kamis", "12-Jul-2029", 30], ["Rabiulakhir", "Sabtu", "11-Aug-2029", 29], ["Jumadilawal", "Ahad", "09-Sep-2029", 30], ["Jumadilakhir", "Selasa", "09-Oct-2029", 29], ["Rajab", "Rabu", "07-Nov-2029", 30], ["Syakban", "Jumat", "07-Dec-2029", 29], ["Ramadan", "Sabtu", "05-Jan-2030", 30], ["Syawal", "Senin", "04-Feb-2030", 29], ["Zulkaidah", "Selasa", "05-Mar-2030", 30], ["Zulhijah", "Kamis", "04-Apr-2030", 29]],
  1452: [["Muharam", "Jumat", "03-May-2030", 30], ["Safar", "Ahad", "02-Jun-2030", 30], ["Rabiulawal", "Selasa", "02-Jul-2030", 30], ["Rabiulakhir", "Kamis", "01-Aug-2030", 29], ["Jumadilawal", "Jumat", "30-Aug-2030", 29], ["Jumadilakhir", "Sabtu", "28-Sep-2030", 30], ["Rajab", "Senin", "28-Oct-2030", 29], ["Syakban", "Selasa", "26-Nov-2030", 30], ["Ramadan", "Kamis", "26-Dec-2030", 29], ["Syawal", "Jumat", "24-Jan-2031", 29], ["Zulkaidah", "Sabtu", "22-Feb-2031", 30], ["Zulhijah", "Senin", "24-Mar-2031", 30]],
  1453: [["Muharam", "Rabu", "23-Apr-2031", 29], ["Safar", "Kamis", "22-May-2031", 30], ["Rabiulawal", "Sabtu", "21-Jun-2031", 30], ["Rabiulakhir", "Senin", "21-Jul-2031", 29], ["Jumadilawal", "Selasa", "19-Aug-2031", 30], ["Jumadilakhir", "Kamis", "18-Sep-2031", 29], ["Rajab", "Jumat", "17-Oct-2031", 30],

["Syakban", "Ahad", "16-Nov-2031", 30], ["Ramadan", "Senin", "15-Dec-2031", 29], ["Syawal", "Rabu", "14-Jan-2032", 29], ["Zulkaidah", "Kamis", "12-Feb-2032", 29], ["Zulhijah", "Jumat", "12-Mar-2032", 30]],

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

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

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

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

1458: [["Muharam", "Kamis", "28-Feb-2036", 30], ["Safar", "Sabtu", "29-Mar-2036", 29], ["Rabiulawal", "Ahad", "27-Apr-2036", 30], ["Rabiulakhir", "Selasa", "27-May-2036", 29], ["Jumadilawal", "Rabu", "25-Jun-2036", 29], ["Jumadilakhir", "Kamis", "24-Jul-2036", 30], ["Rajab", "Sabtu", "23-Aug-2036", 29], ["Syakban", "Ahad", "21-Sep-2036", 29], ["Ramadan", "Senin", "20-Oct-2036", 30], ["Syawal", "Rabu", "19-Nov-2036", 30], ["Zulkaidah", "Jumat", "19-Dec-2036", 30], ["Zulhijah", "Ahad", "18-Jan-2037", 29]],

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

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

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

1462: [["Muharam", "Ahad", "15-Jan-2040", 29], ["Safar", "Senin", "13-Feb-2040", 30], ["Rabiulawal", "Rabu", "14-Mar-2040", 29], ["Rabiulakhir", "Kamis", "12-Apr-2040", 30], ["Jumadilawal", "Sabtu", "12-May-2040", 29], ["Jumadilakhir", "Ahad", "10-Jun-2040", 30], ["Rajab", "Selasa", "10-Jul-2040", 30], ["Syakban", "Kamis", "09-Aug-2040", 29], ["Ramadan", "Jumat", "07-Sep-2040", 30], ["Syawal", "Ahad", "07-Oct-2040", 30], ["Zulkaidah", "Selasa", "06-Nov-2040", 29], ["Zulhijah", "Rabu", "05-Dec-2040", 30]],
1463: [["Muharam", "Jumat", "04-Jan-2041", 29], ["Safar", "Sabtu", "02-Feb-2041", 29], ["Rabiulawal", "Ahad", "03-Mar-2041", 30], ["Rabiulakhir", "Selasa", "02-Apr-2041", 29], ["Jumadilawal", "Rabu", "01-May-2041", 30], ["Jumadilakhir", "Jumat", "31-May-2041", 29], ["Rajab", "Sabtu", "29-Jun-2041", 30], ["Syakban", "Senin", "29-Jul-2041", 29], ["Ramadan", "Selasa", "27-Aug-2041", 30], ["Syawal", "Kamis", "26-Sep-2041", 30], ["Zulkaidah", "Sabtu", "26-Oct-2041", 30], ["Zulhijah", "Senin", "25-Nov-2041", 29]],
1464: [["Muharam", "Selasa", "24-Dec-2041", 30], ["Safar", "Kamis", "23-Jan-2042", 29], ["Rabiulawal", "Jumat", "21-Feb-2042", 30], ["Rabiulakhir", "Ahad", "23-Mar-2042", 29], ["Jumadilawal", "Senin", "21-Apr-2042", 29], ["Jumadilakhir", "Selasa", "20-May-2042", 30], ["Rajab", "Kamis", "19-Jun-2042", 29], ["Syakban", "Jumat", "18-Jul-2042", 30], ["Ramadan", "Ahad", "17-Aug-2042", 29], ["Syawal", "Senin", "16-Sep-2042", 30], ["Zulkaidah", "Rabu", "15-Oct-2042", 30], ["Zulhijah", "Jumat", "14-Nov-2042", 30]],
1465: [["Muharam", "Ahad", "14-Dec-2042", 29], ["Safar", "Senin", "12-Jan-2043", 30], ["Rabiulawal", "Rabu", "11-Feb-2043", 29], ["Rabiulakhir", "Kamis", "12-Mar-2043", 30], ["Jumadilawal", "Sabtu", "11-Apr-2043", 29], ["Jumadilakhir", "Ahad", "10-May-2043", 30], ["Rajab", "Senin", "08-Jun-2043", 29], ["Syakban", "Selasa", "07-Jul-2043", 30], ["Ramadan", "Kamis", "06-Aug-2043", 29], ["Syawal", "Jumat", "04-Sep-2043", 30], ["Zulkaidah", "Ahad", "04-Oct-2043", 30], ["Zulhijah", "Selasa", "03-Nov-2043", 30]],
1466: [["Muharam", "Kamis", "03-Dec-2043", 29], ["Safar", "Jumat", "01-Jan-2044", 30], ["Rabiulawal", "Ahad", "31-Jan-2044", 30], ["Rabiulakhir", "Selasa", "01-Mar-2044", 29], ["Jumadilawal", "Rabu", "30-Mar-2044", 30], ["Jumadilakhir", "Jumat", "29-Apr-2044", 29], ["Rajab", "Sabtu", "28-May-2044", 29], ["Syakban", "Ahad", "26-Jun-2044", 29], ["Ramadan", "Senin", "25-Jul-2044", 30], ["Syawal", "Rabu", "24-Aug-2044", 29], ["Zulkaidah", "Kamis", "22-Sep-2044", 30], ["Zulhijah", "Sabtu", "22-Oct-2044", 30]],
1467: [["Muharam", "Senin", "21-Nov-2044", 29], ["Safar", "Selasa", "20-Dec-2044", 30], ["Rabiulawal", "Kamis", "19-Jan-2045", 30], ["Rabiulakhir", "Sabtu", "18-Feb-2045", 30], ["Jumadilawal", "Senin", "20-Mar-2045", 29], ["Jumadilakhir", "Selasa", "18-Apr-2045", 30], ["Rajab", "Kamis", "18-May-2045", 29], ["Syakban", "Jumat", "16-Jun-2045", 29], ["Ramadan", "Sabtu", "15-Jul-2045", 29], ["Syawal", "Ahad", "13-Aug-2045", 30], ["Zulkaidah", "Selasa", "12-Sep-2045", 29], ["Zulhijah", "Rabu", "11-Oct-2045", 30]],
1468: [["Muharam", "Jumat", "10-Nov-2045", 30], ["Safar", "Ahad", "10-Dec-2045", 29], ["Rabiulawal", "Senin", "08-Jan-2046", 30], ["Rabiulakhir", "Rabu", "07-Feb-2046", 30], ["Jumadilawal", "Jumat", "09-Mar-2046", 29], ["Jumadilakhir", "Sabtu", "07-Apr-2046", 30], ["Rajab", "Senin", "07-May-2046", 29], ["Syakban", "Selasa", "05-Jun-2046", 30], ["Ramadan", "Kamis", "05-Jul-2046", 29], ["Syawal", "Jumat", "03-Aug-2046", 30], ["Zulkaidah", "Ahad", "02-Sep-2046", 29], ["Zulhijah", "Senin", "01-Oct-2046", 29]],
1469: [["Muharam", "Selasa", "30-Oct-2046", 30], ["Safar", "Kamis", "29-Nov-2046", 29], ["Rabiulawal", "Sabtu", "29-Dec-2046", 30], ["Rabiulakhir", "Ahad", "27-Jan-2047", 29], ["Jumadilawal", "Selasa", "26-Feb-2047", 30], ["Jumadilakhir", "Kamis", "28-Mar-2047", 29], ["Rajab", "Jumat", "26-Apr-2047", 30], ["Syakban", "Ahad", "26-May-2047", 29], ["Ramadan", "Selasa", "25-Jun-2047", 30], ["Syawal", "Rabu", "24-Jul-2047", 29], ["Zulkaidah", "Kamis", "22-Aug-2047", 30], ["Zulhijah", "Sabtu", "21-Sep-2047", 29]],
1470: [["Muharam", "Ahad", "20-Oct-2047", 30], ["Safar", "Selasa", "19-Nov-2047", 29], ["Rabiulawal", "Rabu", "18-Dec-2047", 29], ["Rabiulakhir", "Kamis", "16-Jan-2048", 30], ["Jumadilawal", "Sabtu", "15-Feb-2048", 29], ["Jumadilakhir", "Ahad", "15-Mar-2048", 30], ["Rajab", "Selasa", "14-Apr-2048", 30],

["Syakban", "Kamis", "14-May-2048", 30], ["Ramadan", "Sabtu", "13-Jun-2048", 29], ["Syawal", "Ahad", "12-Jul-2048", 30], ["Zulkaidah", "Selasa", "11-Aug-2048", 29], ["Zulhijah", "Rabu", "09-Sep-2048", 30]],
1471: [["Muharam", "Jumat", "09-Oct-2048", 29], ["Safar", "Sabtu", "07-Nov-2048", 30], ["Rabiulawal", "Senin", "07-Dec-2048", 29], ["Rabiulakhir", "Selasa", "05-Jan-2049", 29], ["Jumadilawal", "Rabu", "03-Feb-2049", 30], ["Jumadilakhir", "Jumat", "05-Mar-2049", 29], ["Rajab", "Sabtu", "03-Apr-2049", 30], ["Syakban", "Senin", "03-May-2049", 30], ["Ramadan", "Rabu", "02-Jun-2049", 29], ["Syawal", "Kamis", "01-Jul-2049", 30], ["Zulkaidah", "Sabtu", "31-Jul-2049", 30], ["Zulhijah", "Senin", "30-Aug-2049", 29]],
1472: [["Muharam", "Selasa", "28-Sep-2049", 30], ["Safar", "Kamis", "28-Oct-2049", 29], ["Rabiulawal", "Jumat", "26-Nov-2049", 30], ["Rabiulakhir", "Ahad", "26-Dec-2049", 29], ["Jumadilawal", "Senin", "24-Jan-2050", 29], ["Jumadilakhir", "Selasa", "22-Feb-2050", 30], ["Rajab", "Kamis", "24-Mar-2050", 29], ["Syakban", "Jumat", "22-Apr-2050", 30], ["Ramadan", "Ahad", "22-May-2050", 29], ["Syawal", "Senin", "20-Jun-2050", 30], ["Zulkaidah", "Rabu", "20-Jul-2050", 30], ["Zulhijah", "Jumat", "19-Aug-2050", 29]],
1473: [["Muharam", "Sabtu", "17-Sep-2050", 30], ["Safar", "Senin", "17-Oct-2050", 30], ["Rabiulawal", "Rabu", "16-Nov-2050", 29], ["Rabiulakhir", "Kamis", "15-Dec-2050", 30], ["Jumadilawal", "Sabtu", "14-Jan-2051", 29], ["Jumadilakhir", "Ahad", "12-Feb-2051", 30], ["Rajab", "Selasa", "14-Mar-2051", 29], ["Syakban", "Rabu", "12-Apr-2051", 29], ["Ramadan", "Kamis", "11-May-2051", 30], ["Syawal", "Sabtu", "10-Jun-2051", 29], ["Zulkaidah", "Ahad", "09-Jul-2051", 30], ["Zulhijah", "Selasa", "08-Aug-2051", 29]],
1474: [["Muharam", "Rabu", "06-Sep-2051", 30], ["Safar", "Jumat", "06-Oct-2051", 30], ["Rabiulawal", "Ahad", "05-Nov-2051", 29], ["Rabiulakhir", "Senin", "04-Dec-2051", 30], ["Jumadilawal", "Rabu", "03-Jan-2052", 30], ["Jumadilakhir", "Jumat", "02-Feb-2052", 29], ["Rajab", "Sabtu", "02-Mar-2052", 30], ["Syakban", "Senin", "01-Apr-2052", 29], ["Ramadan", "Selasa", "30-Apr-2052", 29], ["Syawal", "Rabu", "29-May-2052", 30], ["Zulkaidah", "Jumat", "28-Jun-2052", 29], ["Zulhijah", "Sabtu", "27-Jul-2052", 30]],
1475: [["Muharam", "Senin", "26-Aug-2052", 29], ["Safar", "Selasa", "24-Sep-2052", 30], ["Rabiulawal", "Kamis", "24-Oct-2052", 29], ["Rabiulakhir", "Jumat", "22-Nov-2052", 30], ["Jumadilawal", "Ahad", "22-Dec-2052", 30], ["Jumadilakhir", "Selasa", "21-Jan-2053", 29], ["Rajab", "Kamis", "20-Feb-2053", 29], ["Syakban", "Jumat", "21-Mar-2053", 30], ["Ramadan", "Ahad", "20-Apr-2053", 29], ["Syawal", "Senin", "19-May-2053", 29], ["Zulkaidah", "Selasa", "17-Jun-2053", 30], ["Zulhijah", "Kamis", "17-Jul-2053", 29]],
1476: [["Muharam", "Jumat", "15-Aug-2053", 29], ["Safar", "Sabtu", "13-Sep-2053", 30], ["Rabiulawal", "Senin", "13-Oct-2053", 29], ["Rabiulakhir", "Selasa", "11-Nov-2053", 30], ["Jumadilawal", "Kamis", "11-Dec-2053", 30], ["Jumadilakhir", "Sabtu", "10-Jan-2054", 30], ["Rajab", "Senin", "09-Feb-2054", 29], ["Syakban", "Rabu", "11-Mar-2054", 29], ["Ramadan", "Kamis", "09-Apr-2054", 30], ["Syawal", "Sabtu", "09-May-2054", 29], ["Zulkaidah", "Ahad", "07-Jun-2054", 29], ["Zulhijah", "Senin", "06-Jul-2054", 30]],
1477: [["Muharam", "Rabu", "05-Aug-2054", 29], ["Safar", "Kamis", "03-Sep-2054", 29], ["Rabiulawal", "Jumat", "02-Oct-2054", 30], ["Rabiulakhir", "Ahad", "01-Nov-2054", 29], ["Jumadilawal", "Senin", "30-Nov-2054", 30], ["Jumadilakhir", "Rabu", "30-Dec-2054", 30], ["Rajab", "Jumat", "29-Jan-2055", 30], ["Syakban", "Ahad", "28-Feb-2055", 29], ["Ramadan", "Senin", "29-Mar-2055", 30], ["Syawal", "Rabu", "28-Apr-2055", 30], ["Zulkaidah", "Jumat", "28-May-2055", 29], ["Zulhijah", "Sabtu", "26-Jun-2055", 29]],
1478: [["Muharam", "Ahad", "25-Jul-2055", 30], ["Safar", "Selasa", "24-Aug-2055", 29], ["Rabiulawal", "Rabu", "22-Sep-2055", 29], ["Rabiulakhir", "Kamis", "21-Oct-2055", 30], ["Jumadilawal", "Sabtu", "20-Nov-2055", 29], ["Jumadilakhir", "Ahad", "19-Dec-2055", 30], ["Rajab", "Selasa", "18-Jan-2056", 30], ["Syakban", "Kamis", "17-Feb-2056", 29], ["Ramadan", "Jumat", "17-Mar-2056", 30], ["Syawal", "Ahad", "16-Apr-2056", 30], ["Zulkaidah", "Selasa", "16-May-2056", 29], ["Zulhijah", "Rabu", "14-Jun-2056", 30]],

1479: [["Muharam", "Jumat", "14-Jul-2056", 29], ["Safar", "Sabtu", "12-Aug-2056", 30], ["Rabiulawal", "Senin", "11-Sep-2056", 29], ["Rabiulakhir", "Selasa", "10-Oct-2056", 30], ["Jumadilawal", "Rabu", "08-Nov-2056", 30], ["Jumadilakhir", "Jumat", "08-Dec-2056", 29], ["Rajab", "Sabtu", "06-Jan-2057", 30], ["Syakban", "Senin", "05-Feb-2057", 29], ["Ramadan", "Selasa", "06-Mar-2057", 30], ["Syawal", "Kamis", "05-Apr-2057", 30], ["Zulkaidah", "Sabtu", "05-May-2057", 29], ["Zulhijah", "Ahad", "03-Jun-2057", 30]],
1480: [["Muharam", "Selasa", "03-Jul-2057", 30], ["Safar", "Kamis", "02-Aug-2057", 29], ["Rabiulawal", "Jumat", "31-Aug-2057", 30], ["Rabiulakhir", "Sabtu", "29-Sep-2057", 30], ["Jumadilawal", "Senin", "29-Oct-2057", 30], ["Jumadilakhir", "Rabu", "28-Nov-2057", 29], ["Rajab", "Kamis", "27-Dec-2057", 29], ["Syakban", "Jumat", "25-Jan-2058", 30], ["Ramadan", "Ahad", "24-Feb-2058", 29], ["Syawal", "Senin", "25-Mar-2058", 30], ["Zulkaidah", "Rabu", "24-Apr-2058", 29], ["Zulhijah", "Kamis", "23-May-2058", 30]],
1481: [["Muharam", "Sabtu", "22-Jun-2058", 29], ["Safar", "Ahad", "21-Jul-2058", 30], ["Rabiulawal", "Selasa", "20-Aug-2058", 30], ["Rabiulakhir", "Kamis", "19-Sep-2058", 30], ["Jumadilawal", "Sabtu", "19-Oct-2058", 29], ["Jumadilakhir", "Ahad", "17-Nov-2058", 30], ["Rajab", "Selasa", "17-Dec-2058", 29], ["Syakban", "Rabu", "15-Jan-2059", 29], ["Ramadan", "Kamis", "13-Feb-2059", 30], ["Syawal", "Sabtu", "15-Mar-2059", 29], ["Zulkaidah", "Ahad", "13-Apr-2059", 30], ["Zulhijah", "Selasa", "13-May-2059", 29]],
1482: [["Muharam", "Rabu", "11-Jun-2059", 30], ["Safar", "Jumat", "11-Jul-2059", 29], ["Rabiulawal", "Sabtu", "09-Aug-2059", 30], ["Rabiulakhir", "Senin", "08-Sep-2059", 30], ["Jumadilawal", "Rabu", "08-Oct-2059", 30], ["Jumadilakhir", "Jumat", "07-Nov-2059", 29], ["Rajab", "Sabtu", "06-Dec-2059", 30], ["Syakban", "Senin", "05-Jan-2060", 29], ["Ramadan", "Selasa", "03-Feb-2060", 30], ["Syawal", "Kamis", "04-Mar-2060", 29], ["Zulkaidah", "Jumat", "02-Apr-2060", 29], ["Zulhijah", "Sabtu", "01-May-2060", 30]],
1483: [["Muharam", "Senin", "31-May-2060", 29], ["Safar", "Selasa", "29-Jun-2060", 29], ["Rabiulawal", "Rabu", "28-Jul-2060", 30], ["Rabiulakhir", "Jumat", "27-Aug-2060", 30], ["Jumadilawal", "Ahad", "26-Sep-2060", 30], ["Jumadilakhir", "Selasa", "26-Oct-2060", 29], ["Rajab", "Rabu", "24-Nov-2060", 30], ["Syakban", "Jumat", "24-Dec-2060", 30], ["Ramadan", "Ahad", "23-Jan-2061", 29], ["Syawal", "Senin", "21-Feb-2061", 30], ["Zulkaidah", "Rabu", "23-Mar-2061", 29], ["Zulhijah", "Kamis", "21-Apr-2061", 29]],
1484: [["Muharam", "Jumat", "20-May-2061", 29], ["Safar", "Sabtu", "18-Jun-2061", 30], ["Rabiulawal", "Senin", "18-Jul-2061", 29], ["Rabiulakhir", "Selasa", "16-Aug-2061", 30], ["Jumadilawal", "Kamis", "15-Sep-2061", 30], ["Jumadilakhir", "Sabtu", "15-Oct-2061", 29], ["Rajab", "Ahad", "13-Nov-2061", 30], ["Syakban", "Selasa", "13-Dec-2061", 30], ["Ramadan", "Kamis", "12-Jan-2062", 30], ["Syawal", "Sabtu", "11-Feb-2062", 29], ["Zulkaidah", "Ahad", "12-Mar-2062", 30], ["Zulhijah", "Selasa", "11-Apr-2062", 29]],
1485: [["Muharam", "Rabu", "10-May-2062", 29], ["Safar", "Kamis", "08-Jun-2062", 29], ["Rabiulawal", "Jumat", "07-Jul-2062", 30], ["Rabiulakhir", "Ahad", "06-Aug-2062", 29], ["Jumadilawal", "Senin", "04-Sep-2062", 30], ["Jumadilakhir", "Rabu", "04-Oct-2062", 29], ["Rajab", "Kamis", "02-Nov-2062", 30], ["Syakban", "Sabtu", "02-Dec-2062", 30], ["Ramadan", "Senin", "01-Jan-2063", 30], ["Syawal", "Rabu", "31-Jan-2063", 29], ["Zulkaidah", "Kamis", "01-Mar-2063", 30], ["Zulhijah", "Sabtu", "31-Mar-2063", 29]],
1486: [["Muharam", "Ahad", "29-Apr-2063", 30], ["Safar", "Selasa", "29-May-2063", 29], ["Rabiulawal", "Rabu", "27-Jun-2063", 30], ["Rabiulakhir", "Jumat", "27-Jul-2063", 29], ["Jumadilawal", "Sabtu", "25-Aug-2063", 29], ["Jumadilakhir", "Ahad", "23-Sep-2063", 30], ["Rajab", "Selasa", "23-Oct-2063", 29], ["Syakban", "Rabu", "21-Nov-2063", 30], ["Ramadan", "Jumat", "21-Dec-2063", 30], ["Syawal", "Ahad", "20-Jan-2064", 29], ["Zulkaidah", "Senin", "18-Feb-2064", 30], ["Zulhijah", "Rabu", "19-Mar-2064", 30]],

1487: [["Muharam", "Jumat", "18-Apr-2064", 29], ["Safar", "Sabtu", "17-May-2064", 30], ["Rabiulawal", "Senin", "16-Jun-2064", 29], ["Rabiulakhir", "Selasa", "15-Jul-2064", 30], ["Jumadilawal", "Kamis", "14-Aug-2064", 29], ["Jumadilakhir", "Jumat", "12-Sep-2064", 29], ["Rajab", "Sabtu", "11-Oct-2064", 30], ["Syakban", "Senin", "10-Nov-2064", 29], ["Ramadan", "Selasa", "09-Dec-2064", 30], ["Syawal", "Kamis", "08-Jan-2065", 29], ["Zulkaidah", "Jumat", "06-Feb-2065", 30], ["Zulhijah", "Ahad", "08-Mar-2065", 30]],
1488: [["Muharam", "Selasa", "07-Apr-2065", 29], ["Safar", "Rabu", "06-May-2065", 30], ["Rabiulawal", "Jumat", "05-Jun-2065", 30], ["Rabiulakhir", "Ahad", "05-Jul-2065", 29], ["Jumadilawal", "Senin", "03-Aug-2065", 30], ["Jumadilakhir", "Rabu", "02-Sep-2065", 29], ["Rajab", "Kamis", "01-Oct-2065", 29], ["Syakban", "Jumat", "30-Oct-2065", 30], ["Ramadan", "Ahad", "29-Nov-2065", 29], ["Syawal", "Senin", "28-Dec-2065", 30], ["Zulkaidah", "Rabu", "27-Jan-2066", 29], ["Zulhijah", "Kamis", "25-Feb-2066", 30]],
1489: [["Muharam", "Sabtu", "27-Mar-2066", 29], ["Safar", "Ahad", "25-Apr-2066", 30], ["Rabiulawal", "Selasa", "25-May-2066", 30], ["Rabiulakhir", "Kamis", "24-Jun-2066", 30], ["Jumadilawal", "Sabtu", "24-Jul-2066", 29], ["Jumadilakhir", "Ahad", "22-Aug-2066", 30], ["Rajab", "Selasa", "21-Sep-2066", 29], ["Syakban", "Rabu", "20-Oct-2066", 30], ["Ramadan", "Jumat", "19-Nov-2066", 29], ["Syawal", "Sabtu", "18-Dec-2066", 29], ["Zulkaidah", "Ahad", "16-Jan-2067", 30], ["Zulhijah", "Selasa", "15-Feb-2067", 29]],
1490: [["Muharam", "Rabu", "16-Mar-2067", 30], ["Safar", "Jumat", "15-Apr-2067", 29], ["Rabiulawal", "Sabtu", "14-May-2067", 30], ["Rabiulakhir", "Senin", "13-Jun-2067", 30], ["Jumadilawal", "Rabu", "13-Jul-2067", 29], ["Jumadilakhir", "Kamis", "11-Aug-2067", 30], ["Rajab", "Sabtu", "10-Sep-2067", 30], ["Syakban", "Ahad", "10-Oct-2067", 29], ["Ramadan", "Selasa", "08-Nov-2067", 30], ["Syawal", "Kamis", "08-Dec-2067", 29], ["Zulkaidah", "Jumat", "06-Jan-2068", 29], ["Zulhijah", "Sabtu", "04-Feb-2068", 30]],
1491: [["Muharam", "Senin", "05-Mar-2068", 29], ["Safar", "Selasa", "03-Apr-2068", 30], ["Rabiulawal", "Kamis", "03-May-2068", 29], ["Rabiulakhir", "Jumat", "01-Jun-2068", 30], ["Jumadilawal", "Ahad", "01-Jul-2068", 29], ["Jumadilakhir", "Senin", "30-Jul-2068", 30], ["Rajab", "Rabu", "29-Aug-2068", 30], ["Syakban", "Jumat", "28-Sep-2068", 29], ["Ramadan", "Sabtu", "27-Oct-2068", 30], ["Syawal", "Senin", "26-Nov-2068", 30], ["Zulkaidah", "Rabu", "26-Dec-2068", 29], ["Zulhijah", "Kamis", "24-Jan-2069", 29]],
1492: [["Muharam", "Jumat", "22-Feb-2069", 30], ["Safar", "Ahad", "24-Mar-2069", 29], ["Rabiulawal", "Senin", "22-Apr-2069", 30], ["Rabiulakhir", "Rabu", "22-May-2069", 29], ["Jumadilawal", "Kamis", "20-Jun-2069", 29], ["Jumadilakhir", "Jumat", "19-Jul-2069", 30], ["Rajab", "Ahad", "18-Aug-2069", 30], ["Syakban", "Selasa", "17-Sep-2069", 29], ["Ramadan", "Rabu", "16-Oct-2069", 30], ["Syawal", "Jumat", "15-Nov-2069", 30], ["Zulkaidah", "Ahad", "15-Dec-2069", 29], ["Zulhijah", "Senin", "13-Jan-2070", 30]],
1493: [["Muharam", "Rabu", "12-Feb-2070", 29], ["Safar", "Jumat", "14-Mar-2070", 29], ["Rabiulawal", "Sabtu", "12-Apr-2070", 29], ["Rabiulakhir", "Ahad", "11-May-2070", 30], ["Jumadilawal", "Selasa", "10-Jun-2070", 29], ["Jumadilakhir", "Rabu", "09-Jul-2070", 29], ["Rajab", "Kamis", "07-Aug-2070", 30], ["Syakban", "Sabtu", "06-Sep-2070", 29], ["Ramadan", "Ahad", "05-Oct-2070", 30], ["Syawal", "Selasa", "04-Nov-2070", 30], ["Zulkaidah", "Kamis", "04-Dec-2070", 29], ["Zulhijah", "Jumat", "02-Jan-2071", 30]],
1494: [["Muharam", "Ahad", "01-Feb-2071", 30], ["Safar", "Selasa", "03-Mar-2071", 30], ["Rabiulawal", "Kamis", "02-Apr-2071", 29], ["Rabiulakhir", "Jumat", "01-May-2071", 29], ["Jumadilawal", "Sabtu", "30-May-2071", 30], ["Jumadilakhir", "Senin", "29-Jun-2071", 29], ["Rajab", "Selasa", "28-Jul-2071", 29], ["Syakban", "Rabu", "26-Aug-2071", 30], ["Ramadan", "Jumat", "25-Sep-2071", 29], ["Syawal", "Sabtu", "24-Oct-2071", 30], ["Zulkaidah", "Senin", "23-Nov-2071", 29], ["Zulhijah", "Selasa", "22-Dec-2071", 30]],
1495: [["Muharam", "Kamis", "21-Jan-2072", 30], ["Safar", "Sabtu", "20-Feb-2072", 30], ["Rabiulawal", "Senin", "21-Mar-2072", 29], ["Rabiulakhir", "Selasa", "19-Apr-2072", 30], ["Jumadilawal", "Kamis", "19-

```
May-2072", 29], ["Jumadilakhir", "Jumat", "17-Jun-2072", 30], ["Rajab", "Ahad", "17-Jul-2072", 29],
["Syakban", "Senin", "15-Aug-2072", 29], ["Ramadan", "Selasa", "13-Sep-2072", 30], ["Syawal", "Kamis",
"13-Oct-2072", 29], ["Zulkaidah", "Jumat", "11-Nov-2072", 30], ["Zulhijah", "Ahad", "11-Dec-2072", 29]],
    1496: [["Muharam", "Senin", "09-Jan-2073", 30], ["Safar", "Rabu", "08-Feb-2073", 30], ["Rabiulawal",
"Jumat", "10-Mar-2073", 30], ["Rabiulakhir", "Ahad", "09-Apr-2073", 29], ["Jumadilawal", "Senin", "08-
May-2073", 30], ["Jumadilakhir", "Rabu", "07-Jun-2073", 29], ["Rajab", "Kamis", "06-Jul-2073", 30],
["Syakban", "Sabtu", "05-Aug-2073", 29], ["Ramadan", "Ahad", "03-Sep-2073", 29], ["Syawal", "Senin",
"02-Oct-2073", 30], ["Zulkaidah", "Rabu", "01-Nov-2073", 29], ["Zulhijah", "Kamis", "30-Nov-2073", 30]],
    1497: [["Muharam", "Sabtu", "30-Dec-2073", 29], ["Safar", "Ahad", "28-Jan-2074", 30], ["Rabiulawal",
"Selasa", "27-Feb-2074", 30], ["Rabiulakhir", "Kamis", "29-Mar-2074", 29], ["Jumadilawal", "Jumat", "27-
Apr-2074", 30], ["Jumadilakhir", "Ahad", "27-May-2074", 30], ["Rajab", "Selasa", "26-Jun-2074", 29],
["Syakban", "Rabu", "25-Jul-2074", 30], ["Ramadan", "Jumat", "24-Aug-2074", 29], ["Syawal", "Sabtu",
"22-Sep-2074", 29], ["Zulkaidah", "Ahad", "21-Oct-2074", 30], ["Zulhijah", "Selasa", "20-Nov-2074", 30]]
})

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

# ========================================
# DATABASE KOTA (Diambil dari master Anda)
# ========================================
CITY_DB = {
    "Aceh": {"Banda Aceh": (5.5483, 95.3238), "Lhokseumawe": (5.1801, 97.1507), "Langsa": (4.4725,
97.9658), "Meulaboh": (4.1438, 96.1265), "Sabang": (5.8942, 95.3184), "Subulussalam": (2.6312,
98.0012), "Takengon": (4.6212, 96.8412), "Kutacane": (3.4812, 97.8112), "Singkil": (2.2851, 97.7942),
"Calang": (4.6312, 95.5812), "Blangpidie": (3.7412, 96.8412), "Karang Baru": (4.3112, 98.0512),
"Blangkejeren": (3.9912, 97.3312), "Sigli": (5.3812, 95.9612), "Meureudu": (5.2412, 96.2512), "Bireuen":
(5.2031, 96.7011), "Simpang Tiga Redelong": (4.7212, 96.8512), "Suka Makmue": (4.1512, 96.3312),
"Sinabang": (2.4812, 96.3712)},
    "Bali": {"Denpasar": (-8.6705, 115.2126), "Singaraja": (-8.1120, 115.0882), "Mangupura": (-8.5812,
115.1712), "Gianyar": (-8.5412, 115.3212), "Tabanan": (-8.5312, 115.1212), "Semarapura": (-8.5312,
```

115.4012), "Amlapura": (-8.4412, 115.6112), "Bangli": (-8.4512, 115.3512), "Negara": (-8.3512, 114.6212)},
"Bangka Belitung": {"Pangkal Pinang": (-2.1300, 106.1100), "Sungailiat": (-1.8512, 106.1112), "Tanjung Pandan": (-2.7312, 107.6312)},
"Banten": {"Serang": (-6.1104, 106.1601), "Cilegon": (-6.0234, 106.0152), "Tangerang": (-6.1702, 106.6403), "Tangerang Selatan": (-6.2886, 106.7179), "Tigaraksa": (-6.2712, 106.4712), "Pandeglang": (-6.3112, 106.1012), "Rangkasbitung": (-6.3512, 106.2412)},
"Bengkulu": {"Bengkulu": (-3.8004, 102.2655), "Curup": (-3.4612, 102.5212), "Manna": (-4.4712, 102.9012), "Argamakmur": (-3.4212, 102.1912), "Mukomuko": (-2.5812, 101.1212)},
"DI Yogyakarta": {"Yogyakarta": (-7.7956, 110.3695), "Sleman": (-7.7212, 110.3644), "Bantul": (-7.8922, 110.3322), "Wates": (-7.8512, 110.1512), "Wonosari": (-7.9612, 110.6012)},
"DKI Jakarta": {"Jakarta Pusat": (-6.1865, 106.8270), "Jakarta Selatan": (-6.2615, 106.8106), "Jakarta Barat": (-6.1674, 106.7637), "Jakarta Timur": (-6.2250, 106.9004), "Jakarta Utara": (-6.1214, 106.8745), "Kepulauan Seribu": (-5.7412, 106.6112)},
"Gorontalo": {"Gorontalo": (0.5435, 123.0568), "Limboto": (0.6212, 122.9812), "Marisa": (0.4512, 121.9412)},
"Jambi": {"Jambi": (-1.6099, 103.6057), "Sungai Penuh": (-2.0722, 101.3789), "Muara Bulian": (-1.7212, 103.2712), "Bangko": (-2.0712, 102.2612), "Muara Bungo": (-1.4812, 102.1212)},
"Jawa Barat": {"Bandung": (-6.9175, 107.6191), "Bekasi": (-6.2383, 106.9756), "Depok": (-6.4025, 106.7942), "Bogor": (-6.5971, 106.7986), "Cimahi": (-6.8722, 107.5422), "Tasikmalaya": (-7.3274, 108.2207), "Garut": (-7.2272, 107.9086), "Ciamis": (-7.3211, 108.3512), "Banjar": (-7.3676, 108.5364), "Cirebon": (-6.7063, 108.5570), "Indramayu": (-6.3271, 108.3201), "Majalengka": (-6.8312, 108.2212), "Kuningan": (-6.9712, 108.4812), "Sukabumi": (-6.9242, 106.9350), "Cianjur": (-6.8201, 107.1394), "Karawang": (-6.3012, 107.3011), "Subang": (-6.5712, 107.7612), "Purwakarta": (-6.5512, 107.4411), "Sumedang": (-6.8412, 107.9211)},
"Jawa Tengah": {"Semarang": (-7.0667, 110.4100), "Demak": (-6.8906, 110.6389), "Kudus": (-6.8048, 110.8400), "Pati": (-6.7509, 111.0384), "Rembang": (-6.7058, 111.3414), "Jepara": (-6.5888, 110.6675), "Blora": (-6.9697, 111.4184), "Grobogan": (-7.0264, 110.9187), "Surakarta": (-7.5703, 110.8297), "Sukoharjo": (-7.6833, 110.8333), "Wonogiri": (-7.8122, 110.9253), "Karanganyar": (-7.5961, 110.9511), "Sragen": (-7.4267, 111.0211), "Boyolali": (-7.5317, 110.5961), "Klaten": (-7.7031, 110.6025), "Magelang": (-7.4706, 110.2178), "Temanggung": (-7.3167, 110.1667), "Wonosobo": (-7.3597, 109.9111), "Purworejo": (-7.7122, 110.0078), "Kebumen": (-7.6697, 109.6522), "Purwokerto": (-7.4244, 109.2300), "Cilacap": (-7.7277, 109.0159), "Purbalingga": (-7.3875, 109.3675), "Banjarnegara": (-7.3961, 109.6967), "Pekalongan": (-6.8886, 109.6753), "Batang": (-6.9111, 109.7289), "Pemalang": (-6.8919, 109.3814), "Tegal": (-6.8676, 109.1371), "Brebes": (-6.8722, 109.0436), "Kendal": (-6.9189, 110.2039), "Salatiga": (-7.3305, 110.5084)},
"Jawa Timur": {"Surabaya": (-7.2575, 112.7521), "Sidoarjo": (-7.4412, 112.7112), "Gresik": (-7.1612, 112.6511), "Malang": (-7.9797, 112.6304), "Batu": (-7.8671, 112.5239), "Kediri": (-7.8172, 112.0116), "Blitar": (-8.0983, 112.1681), "Tulungagung": (-8.0612, 111.9012), "Nganjuk": (-7.6012, 111.9012), "Trenggalek": (-8.0512, 111.7112), "Madiun": (-7.6298, 111.5239), "Ngawi": (-7.4012, 111.4411), "Magetan": (-7.6512, 111.3312), "Ponorogo": (-7.8712, 111.4612), "Pacitan": (-8.2012, 111.0912), "Jember": (-8.1712, 113.7011), "Banyuwangi": (-8.2112, 114.3611), "Bondowoso": (-7.9112, 113.8212), "Situbondo": (-7.7012, 114.0012), "Probolinggo": (-7.7554, 113.2162), "Lumajang": (-8.1312, 113.2212), "Bojonegoro": (-7.1512, 111.8811), "Tuban": (-6.9012, 112.0511), "Lamongan": (-7.1212, 112.4111), "Bangkalan": (-7.0412, 112.7411), "Sampang": (-7.1812, 113.2411), "Pamekasan": (-7.1512, 113.4711),

"Sumenep": (-7.0112, 113.8611), "Pasuruan": (-7.6449, 112.9061), "Mojokerto": (-7.4712, 112.4311), "Jombang": (-7.5512, 112.2312)},
"Kalimantan Barat": {"Pontianak": (-0.0263, 109.3425), "Singkawang": (0.9023, 108.9711), "Mempawah": (0.3512, 108.9612), "Sambas": (1.3512, 109.3012), "Sintang": (0.0712, 111.4912), "Ketapang": (-1.8412, 109.9712)},
"Kalimantan Selatan": {"Banjarmasin": (-3.3167, 114.5900), "Banjarbaru": (-3.4431, 114.8286), "Pelaihari": (-3.7912, 114.7712), "Kotabaru": (-3.2412, 116.2212), "Tanjung": (-2.1812, 115.3912)},
"Kalimantan Tengah": {"Palangkaraya": (-2.2107, 113.9213), "Sampit": (-2.5312, 112.9512), "Pangkalan Bun": (-2.6812, 111.6212), "Muara Teweh": (-0.9512, 114.8912), "Kapuas": (-3.0112, 114.3912)},
"Kalimantan Timur": {"Samarinda": (-0.4949, 117.1492), "Balikpapan": (-1.2654, 116.8312), "Bontang": (0.1328, 117.4728), "Tenggarong": (-0.4312, 116.9812), "Sangatta": (0.5012, 117.5512)},
"Kalimantan Utara": {"Tanjung Selor": (2.8333, 117.3667), "Tarakan": (3.3033, 117.5878), "Nunukan": (4.1312, 117.6512), "Malinau": (3.5812, 116.6312)},
"Kepulauan Riau": {"Tanjung Pinang": (0.9167, 104.4500), "Batam": (1.1301, 104.0520), "Karimun": (0.9912, 103.4212), "Natuna": (3.9412, 108.3812)},
"Lampung": {"Bandar Lampung": (-5.4292, 105.2611), "Metro": (-5.1133, 105.3067), "Kalianda": (-5.7412, 105.5912), "Kotabumi": (-4.8212, 104.8912)},
"Maluku": {"Ambon": (-3.6954, 128.1814), "Tual": (-5.6322, 132.7389), "Masohi": (-3.3012, 128.9512), "Langgur": (-5.6512, 132.7112), "Saumlaki": (-7.9512, 131.3012)},
"Maluku Utara": {"Ternate": (0.7901, 127.3821), "Tidore": (0.6457, 127.4422), "Sofifi": (0.7312, 127.5812), "Tobelo": (1.7212, 128.0112)},
"NTB": {"Mataram": (-8.5833, 116.1167), "Praya": (-8.7112, 116.2712), "Selong": (-8.6512, 116.5312), "Bima": (-8.4605, 118.7265), "Sumbawa Besar": (-8.5012, 117.4212), "Dompu": (-8.5312, 118.4612)},
"NTT": {"Kupang": (-10.1772, 123.6070), "Soe": (-9.8612, 124.2812), "Kefamenanu": (-9.4512, 124.4812), "Waingapu": (-9.6512, 120.2612), "Labuan Bajo": (-8.5012, 119.8812), "Ruteng": (-8.6112, 120.4712), "Ende": (-8.8412, 121.6512), "Maumere": (-8.6254, 122.2132), "Larantuka": (-8.3412, 122.9812)},
"Papua": {"Jayapura": (-2.5916, 140.6690), "Biak": (-1.1712, 136.0812), "Serui": (-1.8812, 136.2312)},
"Papua Barat": {"Manokwari": (-0.8614, 134.0620), "Fakfak": (-3.2212, 132.2912), "Kaimana": (-3.6612, 133.7712)},
"Papua Barat Daya": {"Sorong": (-0.8765, 131.2558), "Waisai": (-0.4112, 130.8112)},
"Papua Selatan": {"Merauke": (-8.4912, 140.4012), "Agats": (-5.5412, 138.1312)},
"Papua Tengah": {"Nabire": (-3.3612, 135.5012), "Timika": (-4.5412, 136.8812)},
"Papua Pegunungan": {"Wamena": (-4.0612, 138.9412)},
"Riau": {"Pekanbaru": (0.5071, 101.4478), "Dumai": (1.6712, 101.4455), "Bangkinang": (0.3312, 101.0312), "Siak": (0.7912, 102.0412)},
"Sulawesi Barat": {"Mamuju": (-2.6712, 118.8812), "Majene": (-3.5412, 118.9712), "Polewali": (-3.4312, 119.3212)},
"Sulawesi Selatan": {"Makassar": (-5.1476, 119.4327), "Parepare": (-4.0135, 119.6247), "Palopo": (-2.9926, 120.1916), "Gowa": (-5.2012, 119.4512), "Bone": (-4.5412, 120.3212)},
"Sulawesi Tengah": {"Palu": (-0.8917, 119.8707), "Luwuk": (-0.9512, 122.7812), "Poso": (-1.3912, 120.7512), "Donggala": (-0.6712, 119.7412)},
"Sulawesi Tenggara": {"Kendari": (-3.9722, 122.5149), "Baubau": (-5.4612, 122.6112), "Kolaka": (-4.0512, 121.5912)},

```
    "Sulawesi Utara": {"Manado": (1.4748, 124.8404), "Bitung": (1.4412, 125.1212), "Tomohon": (1.3212, 124.8312), "Kotamobagu": (0.7212, 124.3112)},
    "Sumatera Barat": {"Padang": (-0.9471, 100.4172), "Bukittinggi": (-0.3041, 100.3695), "Payakumbuh": (-0.2241, 100.6358), "Pariaman": (-0.6171, 100.1239), "Solok": (-0.7937, 100.6599)},
    "Sumatera Selatan": {"Palembang": (-2.9761, 104.7754), "Lubuklinggau": (-3.2952, 102.8611), "Prabumulih": (-3.4308, 104.2255), "Lahat": (-3.7812, 103.5412), "Pagar Alam": (-4.0112, 103.2712)},
    "Sumatera Utara": {"Medan": (3.5952, 98.6722), "Binjai": (3.6010, 98.4854), "Pematangsiantar": (2.9620, 99.0664), "Tebing Tinggi": (3.3364, 99.1625), "Sibolga": (1.7392, 98.7776)},

    # --- PENAMBAHAN 10 KOTA BENUA AMERIKA (MAINLAND) ---
    "Amerika Serikat": {
        # --- Daratan Utama (Mainland USA) ---
        "New York City": (40.7128, -74.0060),
        "Los Angeles": (34.0522, -118.2437),
        "Chicago": (41.8781, -87.6298),
        "Houston": (29.7604, -95.3698),
        "Phoenix": (33.4484, -112.0740),
        "Seattle": (47.6062, -122.3321),
        "Miami": (25.7617, -80.1918),
        "Washington D.C.": (38.9072, -77.0369),

        # --- Alaska & Semenanjung Alaska (Mainland) ---
        "Anchorage": (61.2181, -149.9003),     # Pusat Kota Alaska
        "Fairbanks": (64.8378, -147.7164),     # Pedalaman Daratan Alaska
        "King Salmon": (58.6833, -156.6667),    # Semenanjung Alaska (Alaska Peninsula)
        "Port Heiden": (56.9, -158.6),
        "Aleutians East Borough": (56.807956, -158.944580),
        "Cold Bay": (55.2000, -162.7167),      # Ujung Semenanjung Alaska
        "Homer": (59.6425, -151.5483),         # Semenanjung Kenai, Daratan Alaska
        "Valdez": (61.1308, -146.3483)         # Pesisir Daratan Alaska
    },
    "Kanada": {
        # --- Kanada (Mainland Amerika Utara) ---
        "Toronto": (43.6510, -79.3470),
        "Vancouver": (49.2827, -123.1207),
        "Montreal": (45.5017, -73.5673)
    },
    "Meksiko": {
        # --- Meksiko (Mainland Amerika Utara) ---
        "Mexico City": (19.4326, -99.1332),
        "Monterrey": (25.6866, -100.3161),
        "Guadalajara": (20.6597, -103.3496)
    },
    "Brasil": {
        "São Paulo": (-23.5505, -46.6333)
    },
```

```
"Argentina": {
   "Buenos Aires": (-34.6037, -58.3816)
},
"Kolombia": {
   "Bogota": (4.7110, -74.0721)
},
"Peru": {
   "Lima": (-12.0464, -77.0428)
},
"Chili": {
   "Santiago": (-33.4489, -70.6693)
},

# Saudi Arabia
"Arab Saudi": {
   # --- Lokasi Utama Rukyatul Hilal & Observatorium ---
   "Tumair": (25.7039, 45.8678),     # Pusat rukyah hilal paling terkenal di KSA
   "Majmaah": (25.8970, 45.3372),    # Observatorium Universitas Majmaah (Sudair)
   "Shaqra": (25.2524, 45.2536),     # Sering menjadi titik laporan awal hilal

   # --- Kota Suci & Pusat Administratif ---
   "Makkah": (21.4225, 39.8262),     # Titik referensi Kalender Umm al-Qura
   "Madinah": (24.4672, 39.6112),
   "Riyadh": (24.7136, 46.6753),
   "Jeddah": (21.4858, 39.1925),

   # --- Kota-Kota Besar di Berbagai Penjuru ---
   "Taif": (21.2703, 40.4158),       # Dataran tinggi, sering digunakan untuk rukyah
   "Tabuk": (28.3835, 36.5662),      # Wilayah Utara
   "Dammam": (26.4207, 50.0888),     # Pesisir Timur (Teluk Persia)
   "Abha": (18.2164, 42.5053),       # Dataran tinggi Asir (Selatan)
   "Jazan": (16.8892, 42.5511),      # Pesisir Barat Daya
   "Buraidah": (26.3260, 43.9750),   # Pusat wilayah Al-Qassim
   "Hail": (27.5219, 41.6907),
   "Najran": (17.4933, 44.1277),     # Perbatasan Selatan
   "Al-Baha": (20.0129, 41.4677),
   "Sakaka": (29.9697, 40.2064),     # Wilayah Al Jawf
   "Arar": (30.9833, 40.0333),       # Perbatasan Utara
   "Yanbu": (24.0891, 38.0637),      # Pesisir Barat Laut
   "Al Ahsa": (25.3795, 49.5858)     # Wilayah Timur (Hofuf)
},

"Mesir": {
   # --- Observatorium & Kota Utama ---
   "Kairo": (30.0444, 31.2357),         # Pusat Dar al-Ifta (dekat Obs. Helwan)
   "Alexandria": (31.2001, 29.9187),    # Pesisir Utara (Laut Tengah)
```

```
    "Aswan": (24.0889, 32.8998),          # Wilayah Selatan (Sering cerah)

    # --- Titik Ekstrem & Geografis Beragam ---
    "Sallum": (31.5658, 25.1481),         # Ujung Barat Laut (Batas Libya, sunset paling akhir)
    "Abu Simbel": (22.3372, 31.6258),     # Ujung Selatan
    "Kharga": (25.4390, 30.5586),         # Dataran gurun (New Valley), minim polusi cahaya
    "Sohag": (26.5591, 31.6957),          # Mesir Tengah
    "Faiyum": (29.3084, 30.8428),
    "Qena": (26.1551, 32.7160),
    "Sharm El-Sheikh": (27.9158, 34.3299)  # Ujung Timur Laut (Semenanjung Sinai)
},
"Maroko": {
    # --- Kota Utama & Pusat Rukyat ---
    "Rabat": (34.0209, -6.8416),          # Pusat pemerintahan & pelaporan resmi
    "Casablanca": (33.5731, -7.5898),
    "Marrakech": (31.6295, -7.9811),      # Dataran tinggi pedalaman
    "Fez": (34.0331, -5.0003),

    # --- Titik Ekstrem Barat & Selatan (Krusial untuk Hilal Global) ---
    "Dakhla": (23.6848, -15.9579),        # Ujung Selatan-Barat (Sangat penting untuk elongasi hilal)
    "Laayoune": (27.1253, -13.1625),      # Wilayah Barat Daya
    "Agadir": (30.4278, -9.5981),         # Pesisir Barat
    "Tangier": (35.7595, -5.8340),        # Ujung Utara (Selat Gibraltar)
    "Oujda": (34.6814, -1.9086),          # Ujung Timur (Batas Aljazair)
    "Ouarzazate": (30.9189, -6.8934)      # Gerbang Gurun Sahara (Langit sangat bersih)
},

"Kepulauan Pasifik (Timur Jauh)": {
    # --- Garis Batas Penanggalan Internasional (Garis Awal KHGT) ---
    "Kiritimati": (1.8709, -157.3962),    # Kiribati (UTC+14, titik paling awal terbenam matahari di bumi)
    "Apia": (-13.8333, -171.7667),        # Samoa (UTC+13)
    "Nuku'alofa": (-21.1393, -175.2018),  # Tonga (UTC+13)
    "Suva": (-18.1248, 178.4501)          # Fiji (UTC+12)
},
"Selandia Baru": {
    # --- Kawasan Daratan Paling Timur ---
    "Waitangi": (-43.9510, -176.5595),    # Kepulauan Chatham (Waktu maghrib sangat awal)
    "Auckland": (-36.8485, 174.7633),     # Pulau Utara
    "Wellington": (-41.2865, 174.7762),
    "Christchurch": (-43.5321, 172.6362), # Pulau Selatan
    "Dunedin": (-45.8788, 170.5028)
},
"Australia": {
    # --- Sebaran Ufuk Timur ke Barat Australia ---
    "Sydney": (-33.8688, 151.2093),       # Pesisir Timur
    "Brisbane": (-27.4698, 153.0251),
```

```
    "Melbourne": (-37.8136, 144.9631),
    "Hobart": (-42.8821, 147.3272),       # Ujung Selatan (Tasmania)
    "Adelaide": (-34.9285, 138.6007),     # Australia Selatan
    "Alice Springs": (-23.6980, 133.8807), # Pedalaman Tengah (Gurun, langit cerah)
    "Darwin": (-12.4634, 130.8456),       # Ujung Utara
    "Perth": (-31.9505, 115.8605)         # Pesisir Barat (Krusial sebelum masuk bujur Asia)
},
"Jepang": {
    # --- Ufuk Timur Belahan Bumi Utara ---
    "Tokyo": (35.6762, 139.6503),         # Kota metropolitan timur
    "Osaka": (34.6937, 135.5023),
    "Sapporo": (43.0618, 141.3545),       # Ujung Utara (Hokkaido)
    "Naha": (26.2124, 127.6809)           # Ujung Selatan (Okinawa)
},

"Turki": {
    # --- Titik Penting Kriteria Diyanet & Observasi ---
    "Istanbul": (41.0082, 28.9784),       # Penghubung Eropa-Asia
    "Ankara": (39.9334, 32.8597),         # Pusat penentuan kriteria Turki
    "Izmir": (38.4192, 27.1287),          # Pesisir Barat (Laut Aegea)
    "Konya": (37.8746, 32.4932),          # Dataran Tinggi Anatolia
    "Erzurum": (39.9043, 41.2679),        # Wilayah Timur (Elevasi tinggi)
    "Antalya": (36.8969, 30.7133)         # Pesisir Selatan
},
"Inggris Raya": {
    # --- Mewakili Ujian Lintang Tinggi (High Latitudes) ---
    "London": (51.5074, -0.1278),         # Titik nol bujur bumi (Greenwich)
    "Birmingham": (52.4862, -1.8904),
    "Manchester": (53.4808, -2.2426),
    "Glasgow": (55.8642, -4.2518),        # Skotlandia (Lintang sangat tinggi, siang/malam ekstrem)
    "Edinburgh": (55.9533, -3.1883)
},
"Eropa Daratan": {
    # --- Titik Krusial di Berbagai Negara Eropa ---
    "Paris": (48.8566, 2.3522),           # Prancis
    "Berlin": (52.5200, 13.4050),         # Jerman
    "Roma": (41.9028, 12.4964),           # Italia
    "Madrid": (40.4168, -3.7038),         # Spanyol (Semenanjung Iberia)
    "Cordoba": (37.8882, -4.7794),        # Spanyol Selatan (Andalusia, titik historis)
    "Sarajevo": (43.8563, 18.4131)        # Bosnia dan Herzegovina
},
"Afrika Selatan": {
    # --- Ufuk Selatan Benua Afrika (Krusial untuk Hilal Global) ---
    "Cape Town": (-33.9249, 18.4241),     # Titik paling barat daya (Sering jadi rujukan rukyat awal)
    "Johannesburg": (-26.2041, 28.0473),  # Dataran tinggi pedalaman
    "Durban": (-29.8587, 31.0218),        # Pesisir Timur
```

```
    "Pretoria": (-25.7479, 28.2293),
    "Bloemfontein": (-29.1141, 26.2206),
    "Port Elizabeth": (-33.9608, 25.6022)  # Pesisir Selatan
  },
  "Rusia": {
    # --- Kaukasus & Rusia Selatan (Tepat di utara Jazirah Arab & Timur Tengah) ---
    "Makhachkala": (42.9831, 47.5046),    # Dagestan (Pesisir Kaspia, Mayoritas muslim, ufuk strategis)
    "Grozny": (43.3194, 45.6949),         # Chechnya (Pusat populasi muslim Kaukasus Utara)
    "Sochi": (43.5853, 39.7203),          # Pesisir Laut Hitam
    "Astrakhan": (46.3497, 48.0408),      # Delta Sungai Volga dekat Laut Kaspia
    "Rostov-na-Donu": (47.2357, 39.7015),  # Pusat administratif Distrik Federal Selatan
    "Volgograd": (48.7000, 44.5167),      # Persimpangan penting Rusia Selatan

    # --- Wilayah Volga & Ural (Pusat populasi Muslim signifikan & kajian falak) ---
    "Kazan": (55.7963, 49.1088),          # Tatarstan (Pusat historis kajian astronomi Islam di Rusia)
    "Ufa": (54.7388, 55.9721),            # Bashkortostan
    "Samara": (53.2415, 50.2212),         # Pertemuan Sungai Volga dan Samara
    "Nizhny Novgorod": (56.3269, 44.0059), # Pusat ekonomi di Distrik Federal Volga

    # --- Kota-Kota Utama (Referensi Zona Waktu Nasional) ---
    "Moskow": (55.7558, 37.6173),         # Ibukota, referensi utama waktu standar Rusia (MSK)

    # --- Lintang Ekstrem Utara (Krusial untuk anomali waktu shalat & puasa) ---
    "Sankt-Peterburg": (59.9343, 30.3351), # Lintang tinggi (Sering mengalami fenomena White Nights)
    "Murmansk": (68.9585, 33.0827),       # Di atas Lingkar Arktik (Matahari tidak terbenam/terbit di
musim tertentu)
    "Arkhangelsk": (64.5399, 40.5167),    # Pelabuhan utara dengan durasi siang ekstrem saat musim
panas

    # --- Siberia & Timur Jauh (Membentang multi-zona waktu ke arah Timur) ---
    "Yekaterinburg": (56.8389, 60.6057),  # Titik batas geografis Eropa - Asia
    "Chelyabinsk": (55.1644, 61.4368),    # Gerbang menuju Siberia
    "Omsk": (54.9885, 73.3242),           # Siberia Barat
    "Novosibirsk": (55.0084, 82.9357),    # Kota terbesar di Siberia (Pusat sains dan observatorium)
    "Krasnoyarsk": (56.0153, 92.8932),    # Siberia Tengah
    "Irkutsk": (52.2978, 104.2964),       # Siberia Timur (Dekat Danau Baikal)
    "Khabarovsk": (48.4814, 135.0721),    # Timur Jauh Rusia
    "Vladivostok": (43.1198, 131.8869)    # Pesisir Pasifik (Ufuk timur paling ujung Rusia)
  },
  "Mesir": {
    # --- Timur Laut Afrika & Pusat Observasi Dar al-Ifta ---
    "Kairo": (30.0444, 31.2357),          # Pusat peradaban Islam dan referensi hisab/rukyat utama
    "Iskandariyah": (31.2001, 29.9187),   # Pesisir Laut Tengah (Utara)
    "Aswan": (24.0889, 32.8998),          # Mesir Hulu (Selatan, dekat Garis Balik Utara)
    "Sharm El-Sheikh": (27.9158, 34.3299), # Ujung Semenanjung Sinai (Timur)
    "Marsa Matruh": (31.3525, 27.2453)    # Ujung barat pesisir utara Mesir
```

```
  },

  "Aljazair": {
    # --- Afrika Utara Bagian Tengah ---
    "Aljir": (36.7538, 3.0588),          # Ibukota, pesisir Laut Tengah
    "Oran": (35.6987, -0.6308),          # Pesisir Barat Laut
    "Constantine": (36.3650, 6.6147),    # Wilayah Timur pedalaman
    "Tamanrasset": (22.7850, 5.5228),    # Wilayah Gurun Sahara (Selatan, kondisi langit sangat jernih untuk observasi)
    "Tindouf": (27.6711, -8.1474)        # Wilayah paling barat Aljazair
  },

  "Maroko": {
    # --- Ufuk Barat Dunia Islam (Sangat krusial untuk Hilal Global/KHGT) ---
    "Rabat": (34.0209, -6.8416),         # Ibukota
    "Casablanca": (33.5731, -7.5898),    # Ufuk Barat menghadap Samudra Atlantik langsung
    "Marrakesh": (31.6295, -7.9811),     # Pedalaman tengah
    "Tangier": (35.7595, -5.8340),       # Ujung utara dekat Selat Gibraltar
    "Dakhla": (23.6848, -15.9580)        # Ujung barat daya (Titik rukyat paling ujung barat di Benua Afrika)
  },

  "Tunisia": {
    # --- Pesisir Utara Afrika ---
    "Tunis": (36.8065, 10.1815),         # Ibukota (Utara)
    "Sfax": (34.7406, 10.7603),          # Pesisir timur
    "Tozeur": (33.9197, 8.1335)          # Pedalaman barat/selatan
  },

  "Libya": {
    # --- Penghubung Mesir dan Maghreb ---
    "Tripoli": (32.8872, 13.1913),       # Ibukota, pesisir barat laut
    "Benghazi": (32.1167, 20.0667),      # Kota utama di timur (Cyrenaica)
    "Sebha": (27.0377, 14.4283)          # Pusat wilayah selatan (Fezzan)
  },

  "Amerika Selatan": {
    # --- Bagian Barat (Pesisir Pasifik & Pegunungan Andes) ---
    "Lima": (-12.0464, -77.0428),        # Ibukota Peru, pusat ekonomi di pesisir barat
    "Santiago": (-33.4489, -70.6693),    # Ibukota Chile, terletak di lembah tengah Andes
    "Bogota": (4.7110, -74.0721),        # Ibukota Kolombia, dataran tinggi Andes utara
    "Quito": (-0.1807, -78.4678),        # Ibukota Ekuador, kota tinggi dekat khatulistiwa
    "La Paz": (-16.4897, -68.1193),      # Pusat pemerintahan Bolivia di pegunungan tinggi
    "Guayaquil": (-2.1894, -79.8891),    # Kota pelabuhan utama di pantai barat Ekuador
    "Valparaiso": (-33.0472, -71.6127),  # Pelabuhan bersejarah dan pusat budaya di Chile
    "Arequipa": (-16.4090, -71.5375),    # Kota besar di selatan Peru, dikelilingi gunung api
```

```
    "Cali": (3.4516, -76.5320),          # Pusat industri di bagian barat daya Kolombia
    "Antofagasta": (-23.6500, -70.4000),   # Kota pelabuhan penting di wilayah utara Chile

    # --- Bagian Timur (Pesisir Atlantik & Dataran Rendah) ---
    "Sao Paulo": (-23.5505, -46.6333),     # Megacity Brasil, pusat finansial terbesar di Amerika Selatan
    "Rio de Janeiro": (-22.9068, -43.1729),# Ikon pariwisata dan budaya pesisir timur Brasil
    "Buenos Aires": (-34.6037, -58.3816),  # Ibukota Argentina, pelabuhan utama di Rio de la Plata
    "Montevideo": (-34.9011, -56.1645),    # Ibukota Uruguay, pusat ekonomi pesisir tenggara
    "Brasilia": (-15.8267, -47.9218),      # Ibukota terencana Brasil di pedalaman timur/tengah
    "Salvador": (-12.9714, -38.5014),      # Kota bersejarah di pesisir timur laut Brasil
    "Fortaleza": (-3.7319, -38.5267),      # Kota besar strategis di pesisir utara/timur laut Brasil
    "Rosario": (-32.9468, -60.6393),       # Pusat ekspor pertanian di sebelah timur Argentina
    "Asuncion": (-25.2637, -57.5759),      # Ibukota Paraguay, akses sungai menuju pesisir timur
    "Caracas": (10.4806, -66.9036)         # Ibukota Venezuela, pusat utara/timur menghadap Karibia
  },

  "Afrika Barat": {
    # --- Pusat Ekonomi & Titik Strategis ---
    "Lagos": (6.5244, 3.3792),           # Pusat ekonomi terbesar di Nigeria dan Afrika Barat
    "Dakar": (14.7167, -17.4677)         # Ibukota Senegal, titik daratan paling barat di benua Afrika
  },

  "Amerika Selatan Tambahan": {
    # --- Bagian Barat (Andes & Pesisir Pasifik) ---
    "Medellin": (6.2442, -75.5812),        # Pusat industri dan inovasi di lembah pegunungan Kolombia
    "Cartagena": (10.3910, -75.4794),      # Kota pelabuhan utama di bagian barat laut Kolombia
    "Cuenca": (-2.9001, -79.0059),         # Kota dataran tinggi bersejarah di selatan Ekuador
    "Manta": (-0.9621, -80.7127),          # Pelabuhan komersial strategis di pesisir barat Ekuador
    "Trujillo": (-8.1159, -79.0282),       # Kota pesisir utara Peru, pusat ekonomi regional
    "Piura": (-5.1945, -80.6328),          # Kota penting di wilayah pesisir barat laut Peru
    "Cusco": (-13.5226, -71.9673),         # Kota di ketinggian Andes Peru, bekas pusat Inca
    "Arica": (-18.4783, -70.3126),         # Pelabuhan paling utara Chile, dekat perbatasan Peru
    "Iquique": (-20.2208, -70.1431),       # Zona perdagangan bebas di pesisir utara Chile
    "Concepcion": (-36.8201, -73.0444),    # Pusat industri dan pelabuhan di tengah-selatan Chile
    "Temuco": (-38.7359, -72.5904),        # Kota utama dan pusat agrikultur di selatan tengah Chile

    # --- Bagian Tengah (Pedalaman, Dataran Tinggi & Lembah Amazon) ---
    "Manaus": (-3.1190, -60.0217),         # Kota pelabuhan besar di tengah lembah sungai Amazon, Brasil
    "Cuiaba": (-15.6014, -56.0979),        # Kota yang diklaim sebagai titik pusat geografis Amerika Selatan
(Brasil)
    "Iquitos": (-3.7491, -73.2538),        # Kota pedalaman Peru, gerbang masuk ke Amazon barat
    "Santa Cruz de la Sierra": (-17.7833, -63.1821), # Pusat komersial dan populasi di dataran rendah
tengah Bolivia
    "Cochabamba": (-17.3895, -66.1568),    # Kota lembah agrikultur di wilayah tengah Bolivia
    "Sucre": (-19.0333, -65.2627),         # Ibukota konstitusional Bolivia di dataran tinggi tengah-selatan
    "Potosi": (-19.5836, -65.7531),        # Kota tambang di dataran sangat tinggi (Andes), Bolivia
```

```
        "Cordoba": (-31.4201, -64.1888),      # Pusat industri dan pendidikan di wilayah tengah Argentina
        "Mendoza": (-32.8908, -68.8272)       # Pusat pertanian di bagian barat-tengah Argentina, kaki
gunung Andes
    },

    "Afrika Barat dan Tengah": {
        # --- Bagian Barat (Pusat Perdagangan & Pesisir Teluk Guinea) ---
        "Accra": (5.5560, -0.1969),          # Ibukota Ghana, pusat komersial di pesisir pesisir selatan
        "Abidjan": (5.3600, -4.0083),        # Kota terbesar di Pantai Gading, pusat ekonomi utama Afrika
Barat
        "Bamako": (12.6392, -8.0029),        # Ibukota Mali, kota perdagangan penting di tepi Sungai Niger
        "Kano": (12.0022, 8.5920),           # Pusat ekonomi dan sejarah Islam di bagian utara Nigeria
        "Port Harcourt": (4.8156, 7.0498),    # Jantung industri minyak dan gas di Delta Niger, Nigeria
        "Conakry": (9.5092, -13.7122),       # Ibukota Guinea, pelabuhan utama di Samudra Atlantik
        "Freetown": (8.4657, -13.2317),       # Ibukota Sierra Leone, pelabuhan alam strategis
        "Monrovia": (6.3156, -10.8074),       # Ibukota Liberia, pusat pemerintahan di pesisir barat
        "Ouagadougou": (12.3714, -1.5197),    # Ibukota Burkina Faso, pusat administratif di pedalaman
        "Niamey": (13.5116, 2.1254),         # Ibukota Niger, terletak strategis di jalur Sungai Niger

        # --- Bagian Tengah (Lembah Kongo & Perbatasan Gurun) ---
        "Kinshasa": (-4.4419, 15.2663),       # Megacity pesisir Sungai Kongo, ibukota RD Kongo
        "Brazzaville": (-4.2634, 15.2429),    # Ibukota Republik Kongo, berseberangan langsung dengan
Kinshasa
        "Yaounde": (3.8480, 11.5021),         # Ibukota Kamerun, terletak di perbukitan pedalaman
        "Douala": (4.0511, 9.7679),          # Kota terbesar dan pelabuhan paling sibuk di Kamerun
        "Libreville": (0.4162, 9.4673),       # Ibukota Gabon, pusat perdagangan di pesisir khatulistiwa
        "Bangui": (4.3947, 18.5582),         # Ibukota Republik Afrika Tengah di tepi Sungai Ubangi
        "N'Djamena": (12.1348, 15.0557),      # Ibukota Chad, pusat perdagangan di utara dekat Danau Chad
        "Lubumbashi": (-11.6609, 27.4794),    # Pusat industri pertambangan tembaga di tenggara RD
Kongo
        "Malabo": (3.7504, 8.7860),          # Ibukota Guinea Khatulistiwa, terletak di Pulau Bioko
        "Pointe-Noire": (-4.7692, 11.8664)    # Pusat industri minyak dan pelabuhan utama Republik Kongo
    },

"China": {
    # --- Bagian Utara & Timur Laut (Pusat Pemerintahan & Industri Berat) ---
    "Beijing": (39.9042, 116.4074),       # Ibukota negara, pusat politik dan budaya
    "Tianjin": (39.0842, 117.2010),       # Kota pelabuhan utama di utara
    "Shijiazhuang": (38.0428, 114.5149),  # Ibukota provinsi Hebei, pusat transportasi
    "Taiyuan": (37.8706, 112.5489),       # Pusat industri batubara di Shanxi
    "Hohhot": (40.8423, 111.6708),        # Ibukota Mongolia Dalam
    "Shenyang": (41.8057, 123.4315),      # Pusat industri dan kota terbesar di timur laut
    "Dalian": (38.9140, 121.6147),        # Pelabuhan utama dan pusat finansial di Liaoning
    "Changchun": (43.8171, 125.3235),     # Ibukota Jilin, dikenal dengan industri otomotif
    "Harbin": (45.8038, 126.5350),        # Ibukota Heilongjiang, terkenal dengan festival es
    "Tangshan": (39.6309, 118.1802),      # Kota industri baja besar di Hebei
```

```
"Baoding": (38.8739, 115.4648),       # Kota bersejarah dekat Beijing
"Handan": (36.6256, 114.5395),        # Kota industri tua di selatan Hebei
"Qinhuangdao": (39.9354, 119.5865),   # Kota pelabuhan timur Hebei
"Datong": (40.0768, 113.2914),        # Kota tambang bersejarah di Shanxi
"Baotou": (40.6522, 109.8222),        # Pusat industri logam tanah jarang di Mongolia Dalam
"Anshan": (41.1078, 122.9945),        # "Ibukota Baja" China di Liaoning
"Jilin": (43.8378, 126.5494),         # Kota industri di provinsi Jilin
"Qiqihar": (47.3543, 123.9182),       # Kota penting di barat laut Heilongjiang
"Chengde": (40.9515, 117.9338),       # Terkenal dengan resor musim panas kekaisaran
"Zhangjiakou": (40.8244, 114.8797),   # Tuan rumah bersama Olimpiade Musim Dingin 2022

# --- Bagian Timur (Pesisir Pantai, Pusat Finansial & Teknologi) ---
"Shanghai": (31.2304, 121.4737),      # Megacity, pusat finansial global
"Nanjing": (32.0603, 118.7969),       # Bekas ibukota kuno, pusat budaya dan pendidikan
"Hangzhou": (30.2741, 120.1551),      # Pusat teknologi (markas Alibaba) dan pariwisata
"Jinan": (36.6512, 117.1201),         # Ibukota Shandong, kota mata air
"Qingdao": (36.0671, 120.3826),       # Kota pelabuhan besar dan pusat pembuatan bir
"Fuzhou": (26.0745, 119.2965),        # Ibukota pesisir Fujian
"Xiamen": (24.4798, 118.0894),        # Kota pulau dan zona ekonomi khusus di Fujian
"Hefei": (31.8206, 117.2272),         # Pusat riset sains dan teknologi yang berkembang pesat
"Nanchang": (28.6820, 115.8579),      # Ibukota Jiangxi
"Suzhou": (31.2989, 120.5853),        # Kota bersejarah yang kaya dengan industri modern
"Wuxi": (31.4912, 120.3119),          # Pusat manufaktur di pesisir Danau Tai
"Ningbo": (29.8683, 121.5439),        # Salah satu pelabuhan tersibuk di dunia
"Wenzhou": (27.9938, 120.6994),       # Pusat perdagangan dan manufaktur swasta
"Quanzhou": (24.8739, 118.6758),      # Titik awal bersejarah Jalur Sutra Maritim
"Xuzhou": (34.2646, 117.1859),        # Hub transportasi penting di utara Jiangsu
"Yantai": (37.4638, 121.4479),        # Pelabuhan utama di semenanjung Shandong
"Weifang": (36.7068, 119.1618),       # Pusat industri layang-layang dunia di Shandong
"Jinhua": (29.0781, 119.6495),        # Pusat e-commerce dan perdagangan di Zhejiang
"Shaoxing": (30.0024, 120.5763),      # Kota bersejarah tekstil di Zhejiang
"Nantong": (32.0147, 120.8646),       # Kota pelabuhan sungai di sebelah utara Shanghai

# --- Bagian Selatan (Pusat Manufaktur, Perdagangan Global & Tropis) ---
"Guangzhou": (23.1291, 113.2644),     # Pusat perdagangan utama dan ibukota Guangdong
"Shenzhen": (22.5431, 114.0579),      # Lembah Silikon-nya China, pusat teknologi raksasa
"Dongguan": (23.0205, 113.7518),      # Basis manufaktur utama dunia
"Foshan": (23.0215, 113.1214),        # Pusat manufaktur furnitur dan elektronik
"Nanning": (22.8146, 108.3200),       # Hub utama perdagangan China dengan ASEAN
"Haikou": (20.0174, 110.3212),        # Ibukota provinsi pulau Hainan
"Sanya": (18.2528, 109.5119),         # Destinasi wisata tropis utama di ujung selatan
"Zhuhai": (22.2707, 113.5767),        # Zona ekonomi khusus berbatasan dengan Makau
"Zhongshan": (22.5176, 113.3928),     # Kota industri di Pearl River Delta
"Shantou": (23.3541, 116.6820),       # Kota pesisir pelabuhan di timur Guangdong
"Zhanjiang": (21.2707, 110.3590),     # Pangkalan angkatan laut dan pelabuhan selatan
"Guilin": (25.2536, 110.2902),        # Terkenal dengan pemandangan pegunungan karst
```

```
    "Liuzhou": (24.3146, 109.4160),       # Pusat industri otomotif di Guangxi
    "Beihai": (21.4812, 109.1194),        # Pelabuhan wisata dan perdagangan pesisir selatan
    "Huizhou": (23.1118, 114.4162),       # Pusat manufaktur elektronik di Pearl River Delta
    "Jiangmen": (22.5787, 113.0816),      # Kampung halaman bagi banyak diaspora Tionghoa
    "Zhaoqing": (23.0515, 112.4651),      # Kota pariwisata di barat Pearl River Delta
    "Chaozhou": (23.6570, 116.6226),      # Pusat budaya Chaoshan dan keramik
    "Maoming": (21.6620, 110.9255),       # Pusat industri petrokimia di Guangdong
    "Yangjiang": (21.8565, 111.9826),     # Dikenal luas karena produksi pisau dan gunting

    # --- Bagian Tengah (Pertanian, Transportasi & Hub Logistik) ---
    "Wuhan": (30.5928, 114.3055),         # Pusat transportasi dan industri utama di lembah Yangtze
    "Zhengzhou": (34.7466, 113.6253),     # Hub kereta api terbesar dan ibukota Henan
    "Changsha": (28.2282, 112.9388),      # Ibukota Hunan, pusat hiburan dan manufaktur alat berat
    "Luoyang": (34.6198, 112.4539),       # Salah satu dari empat ibukota kuno besar
    "Yichang": (30.6919, 111.2865),       # Titik akses ke Bendungan Tiga Ngarai (Three Gorges Dam)
    "Xiangyang": (32.0084, 112.1224),     # Kota bersejarah dan pusat industri otomotif
    "Jingzhou": (30.3349, 112.2397),      # Kota bersejarah di tepi Sungai Yangtze
    "Yueyang": (29.3730, 113.1287),       # Kota pelabuhan sungai dekat Danau Dongting
    "Zhuzhou": (27.8274, 113.1330),       # Pusat manufaktur lokomotif kereta api
    "Xiangtan": (27.8296, 112.9332),      # Pusat industri berat di Hunan
    "Hengyang": (26.8954, 112.5719),      # Pusat transportasi di selatan provinsi Hunan
    "Kaifeng": (34.7969, 114.3076),       # Ibukota kuno Dinasti Song di Henan
    "Xinxiang": (35.3031, 113.9268),      # Kota industri tekstil di utara Henan
    "Anyang": (36.0967, 114.3920),        # Kota asal tulisan tulang oracle Tiongkok awal
    "Xuchang": (34.0253, 113.8223),       # Terkenal dengan produksi rambut palsu dunia
    "Pingdingshan": (33.7371, 113.3005),  # Kota industri batubara di Henan
    "Nanyang": (32.9908, 112.5283),       # Kota berpopulasi padat dengan sejarah medis tradisional
    "Xinyang": (32.1458, 114.0847),       # Terkenal dengan produksi Teh Maojian
    "Shangqiu": (34.4150, 115.6562),      # Kota bersejarah dan jalur transportasi persimpangan
    "Zhumadian": (32.9773, 114.0254),     # Pusat pertanian besar di Henan

    # --- Bagian Barat (Pedalaman, Pegunungan & Jalur Sutra Baru) ---
    "Chengdu": (30.5728, 104.0668),       # Ibukota Sichuan, pusat teknologi, budaya, dan panda
    "Chongqing": (29.5630, 106.5515),     # Megacity pegunungan, pusat manufaktur dan pelabuhan
sungai
    "Xi'an": (34.3416, 108.9398),         # Ibukota Shaanxi, titik awal Jalur Sutra (Prajurit Terakota)
    "Kunming": (25.0453, 102.7100),       # "Kota Musim Semi", pusat gerbang ke Asia Tenggara
    "Guiyang": (26.6470, 106.6302),       # Pusat Big Data yang berkembang pesat di pegunungan
    "Urumqi": (43.8256, 87.6168),         # Ibukota Xinjiang, pusat perdagangan Asia Tengah
    "Lanzhou": (36.0611, 103.8343),       # Pusat transportasi utama di Sungai Kuning
    "Xining": (36.6171, 101.7782),        # Ibukota Qinghai, gerbang menuju Dataran Tinggi Tibet
    "Yinchuan": (38.4872, 106.2309),      # Ibukota Ningxia, pusat budaya suku Hui
    "Lhasa": (29.6500, 91.1000),          # Ibukota Tibet, pusat spiritual dan wilayah dataran tertinggi
    "Mianyang": (31.4675, 104.6796),      # Kota sains dan teknologi utama di Sichuan
    "Nanchong": (30.8378, 106.0849),      # Pusat produksi sutra dan pertanian di Sichuan
    "Zunyi": (27.6868, 106.9272),         # Kota bersejarah revolusi di Guizhou
```

```
    "Qujing": (25.4900, 103.7978),       # Pusat industri terbesar kedua di Yunnan
    "Dali": (25.5916, 100.2227),         # Destinasi wisata sejarah dan pemandangan alam
    "Lijiang": (26.8778, 100.2277),      # Kota kuno warisan UNESCO di pegunungan
    "Kashgar": (39.4677, 75.9938),       # Oasis penting di jalur sutra, ujung barat Tiongkok
    "Turpan": (42.9452, 89.1835),        # Kota oase gurun dengan iklim ekstrim
    "Jiuquan": (39.7289, 98.5042),       # Terkenal dengan pusat peluncuran satelit luar angkasa
    "Baoji": (34.3619, 107.1448)         # Kota industri strategis di bagian barat Shaanxi
},

"Asia Selatan dan Timur Tengah": {
    # --- India (20 Kota) ---
    "New Delhi": (28.6139, 77.2090),     # Ibukota India, pusat pemerintahan di utara
    "Mumbai": (19.0760, 72.8777),        # Pusat finansial dan hiburan di pesisir barat
    "Bengaluru": (12.9716, 77.5946),     # Lembah Silikon India di wilayah selatan
    "Chennai": (13.0827, 80.2707),       # Pusat manufaktur dan budaya di pesisir timur
    "Kolkata": (22.5726, 88.3639),       # Kota bersejarah dan pusat komersial di timur
    "Hyderabad": (17.3850, 78.4867),     # Pusat IT dan kota bersejarah di selatan-tengah
    "Ahmedabad": (23.0225, 72.5714),     # Pusat industri tekstil di Gujarat
    "Pune": (18.5204, 73.8567),          # Pusat pendidikan dan IT di Maharashtra
    "Jaipur": (26.9124, 75.7873),        # "Pink City", pusat budaya di Rajasthan
    "Surat": (21.1702, 72.8311),         # Pusat industri berlian dan tekstil
    "Lucknow": (26.8467, 80.9462),       # Ibukota Uttar Pradesh, kota budaya
    "Kanpur": (26.4499, 80.3319),        # Pusat industri kulit dan tekstil di utara
    "Nagpur": (21.1458, 79.0882),        # Titik pusat geografis India
    "Indore": (22.7196, 75.8577),        # Pusat komersial di Madhya Pradesh
    "Bhopal": (23.2599, 77.4126),        # "Kota Danau" di India bagian tengah
    "Patna": (25.5941, 85.1376),         # Kota kuno di tepi Sungai Gangga
    "Vadodara": (22.3072, 73.1812),      # Pusat budaya dan pendidikan di Gujarat
    "Agra": (27.1767, 78.0081),          # Rumah bagi Taj Mahal
    "Varanasi": (25.3176, 82.9739),      # Kota suci kuno di pesisir Gangga
    "Ludhiana": (30.9010, 75.8573),      # Pusat industri di Punjab

    # --- Pakistan (20 Kota) ---
    "Karachi": (24.8607, 67.0011),       # Megacity pesisir, pusat ekonomi utama Pakistan
    "Lahore": (31.5204, 74.3587),        # Pusat budaya dan sejarah di Punjab
    "Islamabad": (33.6844, 73.0479),     # Ibukota negara yang terencana di utara
    "Rawalpindi": (33.5984, 73.0441),    # Kota militer dan satelit bagi Islamabad
    "Faisalabad": (31.4504, 73.1350),    # "Manchester Pakistan", pusat tekstil
    "Multan": (30.1575, 71.5249),        # "Kota Orang Suci" di Punjab selatan
    "Peshawar": (34.0151, 71.5249),      # Gerbang ke Khyber Pass di barat laut
    "Quetta": (30.1798, 66.9750),        # Ibukota Balochistan di pegunungan
    "Gujranwala": (32.1617, 74.1883),    # Kota industri dan agrikultur di Punjab
    "Sialkot": (32.4945, 74.5229),       # Produsen alat olahraga dan medis kelas dunia
    "Hyderabad_PK": (25.3960, 68.3578),  # Kota tua terbesar kedua di Sindh
    "Bahawalpur": (29.3957, 71.6833),    # Kota pangeran di perbatasan Gurun Cholistan
    "Sargodha": (32.0740, 72.6861),      # Pusat agrikultur penghasil jeruk
```

```
"Sukkur": (27.7052, 68.8574),         # Pusat komersial di tepi Sungai Indus
"Larkana": (27.5570, 68.2028),        # Dekat situs kuno Mohenjo-Daro
"Sheikhupura": (31.7167, 73.9850),    # Kota industri dekat Lahore
"Rahim Yar Khan": (28.4212, 70.2989), # Pusat pertanian dan perdagangan di Punjab selatan
"Jhang": (31.2681, 72.3181),          # Terletak di pertemuan Sungai Chenab dan Jhelum
"Dera Ghazi Khan": (30.0489, 70.6317), # Pusat persimpangan regional
"Gujrat": (32.5736, 74.0741),         # Kota penghasil furnitur dan keramik

# --- Iran (20 Kota) ---
"Tehran": (35.6892, 51.3890),         # Ibukota negara, megacity di kaki Gunung Alborz
"Mashhad": (36.2972, 59.6067),        # Kota suci Islam Syiah di timur laut
"Isfahan": (32.6539, 51.6660),        # Kota mahakarya arsitektur Persia
"Karaj": (35.8327, 50.9915),          # Kota satelit dan industri dekat Tehran
"Shiraz": (29.5918, 52.5836),         # Kota puisi, literatur, dan dekat situs Persepolis
"Tabriz": (38.0773, 46.2919),         # Pusat historis dan ekonomi di barat laut (Azerbaijan Iran)
"Qom": (34.6416, 50.8746),            # Pusat pendidikan agama penting di Iran
"Ahvaz": (31.3183, 48.6706),          # Pusat industri minyak di barat daya
"Kermanshah": (34.3142, 47.0650),     # Pusat budaya Kurdi terbesar di Iran
"Urmia": (37.5527, 45.0761),          # Terletak dekat Danau Urmia di barat laut
"Rasht": (37.2799, 49.5886),          # Kota terbesar di pesisir Laut Kaspia
"Zahedan": (29.4963, 60.8629),        # Ibukota Sistan dan Baluchestan di tenggara
"Hamadan": (34.7982, 48.5146),        # Salah satu kota tertua di dunia
"Kerman": (30.2839, 57.0834),         # Pusat tenun karpet dan eksportir pistachio
"Yazd": (31.8974, 54.3569),           # Kota arsitektur gurun dan pusat Zoroastrianisme
"Ardabil": (38.2514, 48.2973),        # Terkenal dengan perdagangan sutra dan karpet
"Bandar Abbas": (27.1832, 56.2666),   # Pelabuhan utama di Selat Hormuz
"Arak": (34.0954, 49.6909),           # Pusat industri berat
"Zanjan": (36.6736, 48.4787),         # Terkenal dengan kerajinan pisau dan sejarahnya
"Sanandaj": (35.3144, 46.9923),       # Ibukota provinsi Kurdistan di Iran

# --- Afghanistan (20 Kota) ---
"Kabul": (34.5553, 69.2075),          # Ibukota dan kota terbesar di lembah tinggi
"Kandahar": (31.6200, 65.7158),       # Pusat ekonomi utama di selatan
"Herat": (34.3529, 62.2040),          # Kota bersejarah di jalur sutra barat
"Mazar-i-Sharif": (36.7000, 67.1167), # Pusat utama di utara, dekat perbatasan Uzbekistan
"Jalalabad": (34.4265, 70.4515),      # Pusat perdagangan strategis menuju Pakistan
"Kunduz": (36.7286, 68.8681),         # Pusat agrikultur utama di utara
"Taloqan": (36.7361, 69.5345),        # Kota di provinsi Takhar
"Puli Khumri": (35.9446, 68.7151),    # Kota industri dan energi di utara-tengah
"Sheberghan": (36.6676, 65.7529),     # Kaya akan sumber daya alam (gas alam) di utara
"Zaranj": (30.9596, 61.8604),         # Kota perbatasan penting di barat daya
"Maymana": (35.9214, 64.7836),        # Ibukota provinsi Faryab
"Ghazni": (33.5539, 68.4203),         # Kota kuno di poros jalan Kabul-Kandahar
"Khost": (33.3338, 69.9172),          # Terletak di wilayah pegunungan dekat Pakistan
"Shir Khan Bandar": (37.1704, 68.5833),# Pelabuhan kering di perbatasan Tajikistan
"Charikar": (35.0136, 69.1715),       # Pintu gerbang utama ke Lembah Panjshir
```

```
        "Lashkargah": (31.5938, 64.3716),     # Terletak strategis di antara sungai Helmand
        "Farah": (32.3745, 62.1164),          # Kota agrikultur di wilayah barat
        "Asadabad": (34.8731, 71.1469),       # Kota pegunungan di timur laut
        "Gardez": (33.5856, 69.2259),         # Pusat wilayah Loya Paktia di tenggara
        "Bamyan": (34.8216, 67.8242),         # Terkenal dengan situs bersejarah Buddha di pegunungan
tengah

        # --- Irak (20 Kota) ---
        "Baghdad": (33.3152, 44.3661),        # Ibukota negara, dibelah oleh Sungai Tigris
        "Basra": (30.5081, 47.7835),          # Pelabuhan dan pusat industri minyak utama di selatan
        "Mosul": (36.3400, 43.1300),          # Kota utara terbesar, kaya akan peninggalan sejarah
        "Erbil": (36.1901, 44.0090),          # Ibukota wilayah Kurdistan Irak
        "Kirkuk": (35.4670, 44.3831),         # Pusat ladang minyak kuno di utara
        "Najaf": (31.9961, 44.3147),          # Kota suci Islam Syiah di selatan
        "Karbala": (32.6160, 44.0249),        # Situs ziarah Syiah paling penting di dunia
        "Sulaymaniyah": (35.5618, 45.4328),   # Pusat budaya Kurdi di pegunungan
        "Nasiriyah": (31.0580, 46.2573),      # Terletak dekat reruntuhan kota kuno Ur
        "Amarah": (31.8413, 47.1436),         # Kota komersial di wilayah rawa Mesopotamia
        "Diwaniyah": (31.9868, 44.9215),      # Terletak di koridor pertanian yang subur
        "Kut": (32.5115, 45.8247),            # Kota penting di sepanjang Sungai Tigris
        "Hilla": (32.4800, 44.4336),          # Kota yang dibangun di dekat puing-puing Babilonia
        "Ramadi": (33.4243, 43.2989),         # Ibukota provinsi Al Anbar di tepi Efrat
        "Fallujah": (33.3491, 43.7828),       # Kota dengan sejarah modern yang kompleks ("Kota Masjid")
        "Samarra": (34.1979, 43.8906),        # Kota dengan situs Islam bersejarah yang dilindungi UNESCO
        "Baqubah": (33.7466, 44.6247),        # Dikenal luas akan perkebunan jeruknya
        "Halabja": (35.1778, 45.9861),        # Kota perbatasan penting di Kurdistan
        "Zakho": (37.1415, 42.6848),          # Pintu perlintasan ekonomi vital menuju Turki
        "Duhok": (36.8679, 42.9884)           # Kota lembah yang dikelilingi pegunungan Kurdistan
    }
}

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

# ========================================
# FUNGSI-FUNGSI FORMATTING & SAFETY
# ========================================
```

```
def get_safe_events(t_obj, y_obj):
    y_arr = np.atleast_1d(y_obj)
    if len(y_arr) == 0: return []
    events = []
    for k in range(len(y_arr)):
        try: t_val = t_obj[k]
        except Exception: t_val = t_obj
        events.append((t_val, int(y_arr[k])))
    return events

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

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

```
        d = int(deg)
        m = int((deg - d) * 60)
        s = int(round((deg - d - m/60.0) * 3600))
        if s >= 60: s -= 60; m += 1
        if m >= 60: m -= 60; d += 1
        if is_sd: return f"{d:02d}:{m:02d}:{s:02d}"
        if not is_sd and d >= 100: return f"{d:03d}:{m:02d}:{s:02d}"
        return f"{sign}{d:02d}:{m:02d}:{s:02d}"

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

def get_hms_str(time_obj, tz_offset):
    if time_obj is None: return "--.--"

    # BYPASS: Mengekstrak tuple mentah agar kebal terhadap tahun minus (SM/BCE)
    y, m, d, h, mn, s = time_obj.utc

    # Kalkulasi manual untuk menambah zona waktu (tz_offset)
    total_minutes = int(h * 60 + mn + tz_offset * 60 + round(s / 60.0))

    # Modulo untuk memastikan jam tetap dalam format 0-23
    local_hour = (total_minutes // 60) % 24
    local_minute = total_minutes % 60

    return f"{local_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 e:
        print("Pygame mixer init failed:", e)

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

self.setup_ui()

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

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

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

```
    def get_header(self, width):
        lines = [
            "By the Name of Allah",
            "KALENDER HIJRIAH GLOBAL TUNGGAL",
            "KHGT Times 7.0, 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="Mengunduh ephemeris (de421.bsp)...", text_color="#00E5FF")
                download_custom_bsp("de421.bsp", "https://hisabmu.com/aifikih/de421.bsp")

            # Panggil engine deteksi cerdas 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))

        # Mengganti CTkOptionMenu menjadi CTkComboBox agar bisa di-scroll rapi
        # Mengganti CTkOptionMenu menjadi CTkComboBox agar bisa di-scroll rapi
        self.combo_mode = ctk.CTkComboBox(
            self.sidebar,
            values=[
                "1) Visibility Hilal (KHGT)",
                "2) Crescent Visibility Map",
                "3) Analisis Hilal Global",
                "4) Altitude Chart Analyser",
                "5) Kota Pertama KHGT (Mainland)",
                "6) Fase Bulan (Moonphase)",
                "7) Moon Times",
                "8) Sun Times",
                "9) Sun Moon Ephemeris",
                "10) Qibla Time (Rashdul Lokal)",
                "11) Qiblah Direction & Times",
                "12) Prayer Times",
                "13) Konversi Kalender",
                "14) Analisis Gerhana",
                "15) Live Animasi",
                "16) Sistem Sun Moon Earth",
                "17) Equinox & Solstice",
                "18) Planetary Times",
                "19) Simulasi Ephemeris 3D",
                "20) Komparasi 50 Tahun (Ramadhan)",
                "21) Komparasi 50 Tahun (Syawal)",
```

```
            "22) Tabel Tinggi Hilal 50 Thn (Ramadhan)",
            "23) Tabel Elongasi Hilal 50 Thn (Ramadhan)",
            "24) Tabel Tinggi Hilal 50 Thn (Syawal)",
            "25) Tabel Elongasi Hilal 50 Thn (Syawal)",
            "26) Kalender Hijriah Berjalan",
            "27) Kalender Masehi Berjalan",
            "28) WinAI: Aplikasi AI Windows"
        ],
        command=self.switch_mode,
        font=("Segoe UI", 13, "bold"),
        dropdown_font=("Segoe UI", 13),
        fg_color="#1E88E5", button_color="#1565C0", button_hover_color="#0D47A1",
        state="readonly"
    )

    # ---> TAMBAHKAN 1 BARIS INI AGAR MENU PERTAMA JADI DEFAULT TERLIHAT <---
    self.combo_mode.set("1) Visibility Hilal (KHGT)")

    self.combo_mode.pack(fill="x", padx=15, pady=(5, 10))

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

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

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

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

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

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

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

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

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

        # --- CONTAINER 1: VISIBILITY INPUTS ---
        self.frame_vis_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_vdate = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
        frame_vdate.pack(fill="x", padx=15, pady=5)
        ctk.CTkLabel(frame_vdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
        vdate_inputs = ctk.CTkFrame(frame_vdate, fg_color="transparent")
        vdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_vyear = ctk.CTkEntry(vdate_inputs, width=60, placeholder_text="Thn")
        self.entry_vyear.insert(0, curr_y)
        self.entry_vyear.pack(side="left", padx=(0, 5))
        self.entry_vmonth = ctk.CTkEntry(vdate_inputs, width=40, placeholder_text="Bln")
        self.entry_vmonth.insert(0, curr_m)
        self.entry_vmonth.pack(side="left", padx=5)
        self.entry_vday = ctk.CTkEntry(vdate_inputs, width=40, placeholder_text="Tgl")
        self.entry_vday.insert(0, curr_d)
        self.entry_vday.pack(side="left", padx=(5, 0))

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

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

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

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

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

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

        self.var_eph3d_tahun, self.lbl_eph3d_tahun = self.create_eph3d_slider(frame_eph3d_waktu,
"Tahun", -3000, 3000, int(curr_y), 1)
        self.var_eph3d_bulan, self.lbl_eph3d_bulan = self.create_eph3d_slider(frame_eph3d_waktu,
"Bulan", 1, 12, int(curr_m_np), 1)
        self.var_eph3d_hari, self.lbl_eph3d_hari = self.create_eph3d_slider(frame_eph3d_waktu,
"Tanggal", 1, 31, int(curr_d_np), 1)
        self.var_eph3d_jam, self.lbl_eph3d_jam = self.create_eph3d_slider(frame_eph3d_waktu, "Jam
(Lokal)", 0, 23.99, 12.0, 0.01)
        self.var_eph3d_elev, self.lbl_eph3d_elev = self.create_eph3d_slider(frame_eph3d_waktu, "Elevasi
(m)", 0, 5000, 230, 1)

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

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

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

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

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

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

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

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

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

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

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

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

        self.f_cal = ctk.CTkFrame(header_frame, fg_color="#1A1A1A", corner_radius=8, border_width=1,
border_color="#333")
```

```
 self.f_cal.grid(row=0, column=1, sticky="n")

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

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

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

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

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

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

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

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

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

      self.setup_gerhana_out_frame()
      self.setup_animasi_out_frame()
```

```
        self.setup_3d_out_frame()

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

        self.setup_gerhana_out_frame()
        self.setup_animasi_out_frame()
        self.setup_3d_out_frame()

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

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

        # -- KALENDER HIJRIAH INPUTS (MENU 26) --
        self.frame_kalender_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        # ... (lanjutkan kodenya seperti yang sudah ada) ...

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

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

        # -- KALENDER HIJRIAH INPUTS (MENU 26) --
        self.frame_kalender_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        kal_info = ctk.CTkFrame(self.frame_kalender_inputs, fg_color="#212121")
        kal_info.pack(fill="x", padx=15, pady=10)
        ctk.CTkLabel(kal_info, text="PENGATURAN KALENDER", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
        ctk.CTkLabel(kal_info, text="Gunakan kontrol di layar utama\nuntuk berpindah bulan.",
font=("Segoe UI", 11), text_color="#B0BEC5", justify="left").pack(anchor="w", padx=10, pady=(0, 10))
        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=10)

        # -- KALENDER MASEHI INPUTS (MENU 27) --
        self.frame_kalmasehi_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
```

```
        kalm_info = ctk.CTkFrame(self.frame_kalmasehi_inputs, fg_color="#212121")
        kalm_info.pack(fill="x", padx=15, pady=10)
        ctk.CTkLabel(kalm_info, text="PENGATURAN KAL. MASEHI", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(8, 2))
        ctk.CTkLabel(kalm_info, text="Gunakan kontrol di layar utama\nuntuk berpindah bulan.",
font=("Segoe UI", 11), text_color="#B0BEC5", justify="left").pack(anchor="w", padx=10, pady=(0, 10))
        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=10)

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

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

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

        # 2. BARU PANGGIL SWITCH MODE & STARTUP LAINNYA DI SINI
        self.switch_mode("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=(0, 15))
```

```
        # -- EPHEMERIS --
        self.frame_eph_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_eph_cfg = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
        frame_eph_cfg.pack(fill="x", padx=15, pady=5)
        row_obj = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
        row_obj.pack(fill="x", padx=10, pady=2)
        ctk.CTkLabel(row_obj, text="Object:", font=("Segoe UI", 12)).pack(side="left")
        self.combo_eph_obj = ctk.CTkOptionMenu(row_obj, values=["Sun", "Moon"], width=100)
        self.combo_eph_obj.pack(side="right")
        row_tref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
        row_tref.pack(fill="x", padx=10, pady=2)
        ctk.CTkLabel(row_tref, text="Time Ref:", font=("Segoe UI", 12)).pack(side="left")
        self.combo_eph_tref = ctk.CTkOptionMenu(row_tref, values=["Local (UT1)", "Local (TDT)"],
width=100)
        self.combo_eph_tref.pack(side="right")
        row_lref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
        row_lref.pack(fill="x", padx=10, pady=2)
        ctk.CTkLabel(row_lref, text="Location:", font=("Segoe UI", 12)).pack(side="left")
        self.combo_eph_lref = ctk.CTkOptionMenu(row_lref, values=["Topocentric", "Geocentric"],
width=100)
        self.combo_eph_lref.pack(side="right")
        frame_eph_start = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
        frame_eph_start.pack(fill="x", padx=15, pady=5)
        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)
        self.entry_eph_sh, self.entry_eph_smin, self.entry_eph_ssec =
self.create_hms_row(frame_eph_start, "22", "0", "0")
        frame_eph_end = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
        frame_eph_end.pack(fill="x", padx=15, pady=5)
        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)
        self.entry_eph_eh, self.entry_eph_emin, self.entry_eph_esec =
self.create_hms_row(frame_eph_end, "22", "0", "0")
        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=2)
        self.entry_eph_step = ctk.CTkEntry(row_inc, width=50)
        self.entry_eph_step.insert(0, "1")
        self.entry_eph_step.pack(side="left")
        self.combo_eph_step = ctk.CTkOptionMenu(row_inc, values=["Day", "Hour", "Minute", "Second"],
width=80)
        self.combo_eph_step.pack(side="left", padx=5)
        self.entry_eph_lat = self.create_input_row(frame_eph_step, "Lat:", "-7.0667")
        self.entry_eph_lon = self.create_input_row(frame_eph_step, "Lon:", "110.4100")
        self.entry_eph_elev = self.create_input_row(frame_eph_step, "Elev:", "230.0")
```

```
        self.entry_eph_tz = self.create_input_row(frame_eph_step, "TZ:", "7.0")

        # -- QIBLAH DIR --
        self.frame_qiblah_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_qdate = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
        frame_qdate.pack(fill="x", padx=15, pady=5)
        qdate_inputs = ctk.CTkFrame(frame_qdate, fg_color="transparent")
        qdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_qyear = ctk.CTkEntry(qdate_inputs, width=60, placeholder_text="Thn")
        self.entry_qyear.insert(0, curr_y)
        self.entry_qyear.pack(side="left", padx=(0, 5))
        self.entry_qmonth = ctk.CTkEntry(qdate_inputs, width=40, placeholder_text="Bln")
        self.entry_qmonth.insert(0, curr_m)
        self.entry_qmonth.pack(side="left", padx=5)
        self.entry_qday = ctk.CTkEntry(qdate_inputs, width=40, placeholder_text="Tgl")
        self.entry_qday.insert(0, curr_d)
        self.entry_qday.pack(side="left", padx=(5, 0))
        frame_qloc = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
        frame_qloc.pack(fill="x", padx=15, pady=5)
        self.entry_qlat = self.create_input_row(frame_qloc, "Lat (Lintang):", "-7.0667")
        self.entry_qlon = self.create_input_row(frame_qloc, "Lon (Bujur):", "110.4100")
        self.entry_qtz = self.create_input_row(frame_qloc, "Timezone:", "7.0")
        frame_qmode = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
        frame_qmode.pack(fill="x", padx=15, pady=5)
        self.radio_qiblah_var = ctk.StringVar(value="sun")
        ctk.CTkRadioButton(frame_qmode, text="Sun is at Qiblah direction", variable=self.radio_qiblah_var,
value="sun").pack(anchor="w", padx=15, pady=5)
        ctk.CTkRadioButton(frame_qmode, text="Sun's shadow is at Qiblah",
variable=self.radio_qiblah_var, value="shadow").pack(anchor="w", padx=15, pady=(5, 10))
        self.btn_qmap = ctk.CTkButton(self.frame_qiblah_inputs, text="🗺 Show Qiblah Map",
fg_color="#00695C", hover_color="#004D40", command=self.show_qiblah_map)
        self.btn_qmap.pack(fill="x", padx=15, pady=(10, 5))

        # -- MOON TIMES --
        self.frame_mt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_mtdate = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
        frame_mtdate.pack(fill="x", padx=15, pady=5)
        mtdate_inputs = ctk.CTkFrame(frame_mtdate, fg_color="transparent")
        mtdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_mt_year = ctk.CTkEntry(mtdate_inputs, width=60, placeholder_text="Thn")
        self.entry_mt_year.insert(0, curr_y)
        self.entry_mt_year.pack(side="left", padx=(0, 5))
        self.entry_mt_month = ctk.CTkEntry(mtdate_inputs, width=40, placeholder_text="Bln")
        self.entry_mt_month.insert(0, curr_m)
        self.entry_mt_month.pack(side="left", padx=5)
        frame_mtloc = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
```

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

        # -- SUN TIMES --
        self.frame_st_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_stdate = ctk.CTkFrame(self.frame_st_inputs, fg_color="#212121")
        frame_stdate.pack(fill="x", padx=15, pady=5)
        stdate_inputs = ctk.CTkFrame(frame_stdate, fg_color="transparent")
        stdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_st_year = ctk.CTkEntry(stdate_inputs, width=60, placeholder_text="Thn")
        self.entry_st_year.insert(0, curr_y)
        self.entry_st_year.pack(side="left", padx=(0, 5))
        self.entry_st_month = ctk.CTkEntry(stdate_inputs, width=40, placeholder_text="Bln")
        self.entry_st_month.insert(0, curr_m)
        self.entry_st_month.pack(side="left", padx=5)
        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)
        ptdate_inputs = ctk.CTkFrame(frame_ptdate, fg_color="transparent")
        ptdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_pt_year = ctk.CTkEntry(ptdate_inputs, width=60, placeholder_text="Thn")
        self.entry_pt_year.insert(0, curr_y)
        self.entry_pt_year.pack(side="left", padx=(0, 5))
        self.entry_pt_month = ctk.CTkEntry(ptdate_inputs, width=40, placeholder_text="Bln")
        self.entry_pt_month.insert(0, curr_m)
        self.entry_pt_month.pack(side="left", padx=5)
        self.entry_pt_day = ctk.CTkEntry(ptdate_inputs, width=40, placeholder_text="Tgl")
        self.entry_pt_day.insert(0, curr_d)
        self.entry_pt_day.pack(side="left", padx=(5, 0))
        frame_ptloc = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
```

```
        frame_ptloc.pack(fill="x", padx=15, pady=5)
        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))
        frame_pt_opt = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
        frame_pt_opt.pack(fill="x", padx=15, pady=5)
        self.entry_pt_ikhtiyat = self.create_input_row(frame_pt_opt, "Ikhtiyat (Detik):", "16")
        row_fmt = ctk.CTkFrame(frame_pt_opt, fg_color="transparent")
        row_fmt.pack(fill="x", padx=10, pady=2)
        self.combo_pt_fmt = ctk.CTkOptionMenu(row_fmt, values=["HH:MM:SS (Jam:Menit:Detik)",
"HH:MM (Jam:Menit)"])
        self.combo_pt_fmt.pack(fill="x")

        # -- VISMAP --
        self.frame_vismap_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_vmdate = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
        frame_vmdate.pack(fill="x", padx=15, pady=5)
        vmdate_inputs = ctk.CTkFrame(frame_vmdate, fg_color="transparent")
        vmdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_vmday = ctk.CTkEntry(vmdate_inputs, width=35, placeholder_text="D")
        self.entry_vmday.insert(0, curr_d)
        self.entry_vmday.pack(side="left", padx=2)
        self.entry_vmmonth = ctk.CTkEntry(vmdate_inputs, width=35, placeholder_text="M")
        self.entry_vmmonth.insert(0, curr_m)
        self.entry_vmmonth.pack(side="left", padx=2)
        self.entry_vmyear = ctk.CTkEntry(vmdate_inputs, width=50, placeholder_text="Y")
        self.entry_vmyear.insert(0, curr_y)
        self.entry_vmyear.pack(side="left", padx=2)
        frame_vmtoggle = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
        frame_vmtoggle.pack(fill="x", padx=15, pady=5)
        self.radio_vmphase = ctk.StringVar(value="new")
```

```
        ctk.CTkRadioButton(frame_vmtoggle, text="New Crescent (Evening)", variable=self.radio_vmphase,
value="new").pack(anchor="w", padx=15, pady=5)
        ctk.CTkRadioButton(frame_vmtoggle, text="Old Crescent (Morning)", variable=self.radio_vmphase,
value="old").pack(anchor="w", padx=15, pady=(0, 8))
        frame_vmtime = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
        frame_vmtime.pack(fill="x", padx=15, pady=5)
        self.radio_vmtime = ctk.StringVar(value="sunset")
        row_t1 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
        row_t1.pack(fill="x", padx=5)
        ctk.CTkRadioButton(row_t1, text="Sunset", variable=self.radio_vmtime, value="sunset",
width=80).pack(side="left", padx=5)
        ctk.CTkRadioButton(row_t1, text="Moonset", variable=self.radio_vmtime, value="moonset",
width=80).pack(side="left", padx=5)
        row_t2 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
        row_t2.pack(fill="x", padx=5, pady=5)
        ctk.CTkRadioButton(row_t2, text="Best Time", variable=self.radio_vmtime,
value="best").pack(side="left", padx=5)
        frame_vmcrit = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
        frame_vmcrit.pack(fill="x", padx=15, pady=5)
        self.combo_vmcrit = ctk.CTkOptionMenu(frame_vmcrit, values=["KHGT Criterion (Alt>=5,
Elong>=8)", "Odeh Criterion"])
        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)

        # Tambahkan "Interseksi" di akhir daftar (List) values
        self.combo_vmparam = ctk.CTkOptionMenu(frame_vmparam, values=["Kriteria Visibilitas",
"Altitude Bulan", "Elongasi", "Iluminasi", "Interseksi"])
        self.combo_vmparam.pack(fill="x", padx=10, pady=(0, 10))

        # -- KONVERSI --
        self.frame_conv_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_conv_mode = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
        frame_conv_mode.pack(fill="x", padx=15, pady=5)
        self.radio_conv_var = ctk.StringVar(value="m2h")
        ctk.CTkRadioButton(frame_conv_mode, text="Masehi ➔ Hijriah", variable=self.radio_conv_var,
value="m2h", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=5)
        ctk.CTkRadioButton(frame_conv_mode, text="Hijriah ➔ Masehi", variable=self.radio_conv_var,
value="h2m", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=(5, 10))
        frame_conv_date = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
        frame_conv_date.pack(fill="x", padx=15, pady=5)
        self.lbl_conv_date = ctk.CTkLabel(frame_conv_date, text="TANGGAL MASEHI", font=("Segoe UI",
12, "bold"))
        self.lbl_conv_date.pack(anchor="w", padx=10, pady=(8, 2))
        convdate_inputs = ctk.CTkFrame(frame_conv_date, fg_color="transparent")
        convdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
```

```
        days = [str(d).zfill(2) for d in range(1, 32)]
        self.combo_conv_day = ctk.CTkComboBox(convdate_inputs, values=days, width=60)
        self.combo_conv_day.set(curr_d)
        self.combo_conv_day.pack(side="left", padx=2)
        self.combo_conv_month = ctk.CTkComboBox(convdate_inputs, values=BULAN_MASEHI,
width=120)
        self.combo_conv_month.set(BULAN_MASEHI[datetime.datetime.now().month - 1])
        self.combo_conv_month.pack(side="left", padx=2)
        self.entry_conv_year = ctk.CTkEntry(convdate_inputs, width=60, placeholder_text="Thn")
        self.entry_conv_year.insert(0, curr_y)
        self.entry_conv_year.pack(side="left", padx=2)
        frame_conv_adj = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
        frame_conv_adj.pack(fill="x", padx=15, pady=5)
        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)
        qtdate_inputs = ctk.CTkFrame(frame_qtdate, fg_color="transparent")
        qtdate_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_qtyear = ctk.CTkEntry(qtdate_inputs, width=60, placeholder_text="Thn")
        self.entry_qtyear.insert(0, curr_y)
        self.entry_qtyear.pack(side="left", padx=(0, 5))
        self.entry_qtmonth = ctk.CTkEntry(qtdate_inputs, width=40, placeholder_text="Bln")
        self.entry_qtmonth.insert(0, curr_m)
        self.entry_qtmonth.pack(side="left", padx=5)
        self.entry_qtday = ctk.CTkEntry(qtdate_inputs, width=40, placeholder_text="Tgl")
        self.entry_qtday.insert(0, curr_d)
        self.entry_qtday.pack(side="left", padx=(5, 0))
        frame_qtloc = ctk.CTkFrame(self.frame_qt_inputs, fg_color="#212121")
        frame_qtloc.pack(fill="x", padx=15, pady=5)
        self.entry_qtlat = self.create_input_row(frame_qtloc, "Lat (Lintang):", "-7.0667")
        self.entry_qtlon = self.create_input_row(frame_qtloc, "Lon (Bujur):", "110.4100")
        self.entry_qttz = self.create_input_row(frame_qtloc, "Timezone:", "7.0")

        # -- GERHANA --
        self.frame_gerhana_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_gdate = ctk.CTkFrame(self.frame_gerhana_inputs, fg_color="#212121")
        frame_gdate.pack(fill="x", padx=15, pady=10)
        ctk.CTkLabel(frame_gdate, text="PILIH TAHUN GERHANA", font=("Segoe UI", 12,
"bold")).pack(anchor="w", padx=10, pady=(10, 5))
        self.combo_tahun_gerhana = ctk.CTkComboBox(frame_gdate, values=[str(y) for y in range(-3000,
3000)], justify="center")
```

```
        self.combo_tahun_gerhana.set(curr_y)
        self.combo_tahun_gerhana.pack(padx=10, pady=(0, 15))

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

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

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

        self.entry_anim_year = ctk.CTkEntry(anim_date_inputs, width=60, placeholder_text="Thn")
        self.entry_anim_year.insert(0, curr_y)
        self.entry_anim_year.pack(side="left", padx=(0, 5))

        self.entry_anim_month = ctk.CTkEntry(anim_date_inputs, width=40, placeholder_text="Bln")
        self.entry_anim_month.insert(0, curr_m)
        self.entry_anim_month.pack(side="left", padx=5)

        self.entry_anim_day = ctk.CTkEntry(anim_date_inputs, width=40, placeholder_text="Tgl")
        self.entry_anim_day.insert(0, curr_d)
        self.entry_anim_day.pack(side="left", padx=(5, 0))

        # PERBAIKAN: Langsung isi dengan waktu berjalan saat ini
        self.entry_anim_time = ctk.CTkEntry(frame_anim_date, placeholder_text="HH:MM:SS")
        self.entry_anim_time.insert(0, datetime.datetime.now().strftime("%H:%M:%S"))
        self.entry_anim_time.pack(fill="x", padx=10, pady=(0, 10))

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

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

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

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

```
        self.anim_is_live = True
        self.anim_custom_time = None

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

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

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

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

        # Dropdown Tahun - Ditambahkan command
        years_range = [str(i) for i in range(-2999, 3001)]
        self.combo_3d_y = ctk.CTkComboBox(f3d_row, values=years_range, width=85,
command=self.on_3d_date_change)
        self.combo_3d_y.set(curr_y)
        self.combo_3d_y.pack(side="left", padx=2)

        # Info navigasi
        ctk.CTkLabel(self.frame_3d_inputs, text="Klik & Drag layar untuk rotasi.\nScroll/Mousewheel untuk
Zoom.",
            font=("Segoe UI", 11, "italic"), text_color="#FFD54F").pack(pady=10)

        # -- GHA --
        self.frame_gha_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_gha_date = ctk.CTkFrame(self.frame_gha_inputs, fg_color="#212121")
        frame_gha_date.pack(fill="x", padx=15, pady=5)
        gha_date_inputs = ctk.CTkFrame(frame_gha_date, fg_color="transparent")
        gha_date_inputs.pack(fill="x", padx=10, pady=(0, 5))
        self.entry_gha_year = ctk.CTkEntry(gha_date_inputs, width=60, placeholder_text="Thn")
        self.entry_gha_year.insert(0, curr_y)
```

```
        self.entry_gha_year.pack(side="left", padx=(0, 5))
        self.entry_gha_month = ctk.CTkEntry(gha_date_inputs, width=40, placeholder_text="Bln")
        self.entry_gha_month.insert(0, curr_m)
        self.entry_gha_month.pack(side="left", padx=5)
        self.entry_gha_day = ctk.CTkEntry(gha_date_inputs, width=40, placeholder_text="Tgl")
        self.entry_gha_day.insert(0, curr_d)
        self.entry_gha_day.pack(side="left", padx=(5, 0))
        self.entry_gha_time = ctk.CTkEntry(frame_gha_date, placeholder_text="HH:MM:SS (Default
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")
        c_date = ctk.CTkFrame(self.frame_chart_inputs, fg_color="#212121")
        c_date.pack(fill="x", padx=15, pady=10)
        self.entry_chart_y, self.entry_chart_m, self.entry_chart_d = self.create_ymd_row(c_date, curr_y,
curr_m_np, curr_d_np)

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

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

        fp_inputs_row = ctk.CTkFrame(fp_date, fg_color="transparent")
        fp_inputs_row.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 2000 s.d 2100

        # Input Tanggal (Dropdown)
        self.entry_fp_d = ctk.CTkComboBox(fp_inputs_row, values=days_list, width=65)
        self.entry_fp_d.set(curr_d) # Otomatis set ke tanggal hari ini
        self.entry_fp_d.pack(side="left", padx=2)

        # Input Bulan (Dropdown)
        self.entry_fp_m = ctk.CTkComboBox(fp_inputs_row, values=months_list, width=65)
        self.entry_fp_m.set(curr_m) # Otomatis set ke bulan hari ini
        self.entry_fp_m.pack(side="left", padx=2)

        # Input Tahun (Dropdown)
```

```
        self.entry_fp_y = ctk.CTkComboBox(fp_inputs_row, values=years_list, width=80)
        self.entry_fp_y.set(curr_y) # Otomatis set ke tahun hari ini
        self.entry_fp_y.pack(side="left", padx=2)

        # -- SEASON --
        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)
        self.entry_season_year = ctk.CTkEntry(frame_s_year, justify="center")
        self.entry_season_year.insert(0, curr_y)
        self.entry_season_year.pack(pady=(0, 10))

        # -- PLANETARY --
        self.frame_planet_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
        frame_pl_date = ctk.CTkFrame(self.frame_planet_inputs, fg_color="#212121")
        frame_pl_date.pack(fill="x", padx=15, pady=5)
        pl_date_inputs = ctk.CTkFrame(frame_pl_date, fg_color="transparent")
        pl_date_inputs.pack(fill="x", padx=10, pady=(0, 10))
        self.entry_pl_year = ctk.CTkEntry(pl_date_inputs, width=60)
        self.entry_pl_year.insert(0, curr_y)
        self.entry_pl_year.pack(side="left", padx=2)
        self.entry_pl_month = ctk.CTkEntry(pl_date_inputs, width=40)
        self.entry_pl_month.insert(0, curr_m)
        self.entry_pl_month.pack(side="left", padx=2)
        self.combo_pl_target = ctk.CTkOptionMenu(frame_pl_date, values=["Mercury", "Venus", "Mars",
"Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"])
        self.combo_pl_target.pack(fill="x", padx=10, pady=(0, 10))

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


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

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

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

        parts = time_str.split(":")
        jam = int(parts[0])
        menit = int(parts[1]) if len(parts) > 1 else 0
        detik = int(parts[2]) if len(parts) > 2 else 0

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

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

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



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

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

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

    var = ctk.DoubleVar(value=valinit)

    def on_slide(val):
        fmt = "{:.0f}" if res >= 1 else "{:.2f}"
        val_label.configure(text=fmt.format(val))
```

```
            if not getattr(self, 'eph3d_updating', False):
                self.eph3d_is_live = False
                self.eph3d_update_plot()

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

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

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

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

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

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

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

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

        self.ax_eph3d.text(0, 11, 0, 'Utara', color='white', ha='center', fontsize=12)
```

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

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

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

        self.canvas_eph3d.draw()

    def eph3d_r_ke_xyz(self, alt, az, r=10):
        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 Bujur (Longitude) untuk menghitung Zona Waktu dinamis
            lon_val = float(self.entry_eph3d_lon.get())
            self.pe_obs.lat = str(self.entry_eph3d_lat.get())
            self.pe_obs.lon = str(lon_val)
            self.pe_obs.elevation = self.var_eph3d_elev.get()

            # 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)

        # ========================================================
        # 1. 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))

        # ========================================================
        # 2. PERHITUNGAN GEOSENTRIK (Berdasarkan pusat Bumi - KHGT)
        # ========================================================
        # Tinggi Geosentrik Matahari
        HA_sun = self.pe_obs.sidereal_time() - self.pe_sun.g_ra
        sin_alt_geo_sun = math.sin(self.pe_obs.lat) * math.sin(self.pe_sun.g_dec) +
math.cos(self.pe_obs.lat) * math.cos(self.pe_sun.g_dec) * math.cos(HA_sun)
        alt_geo_sun = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_sun))))

        # Tinggi Geosentrik Bulan
        HA_moon = self.pe_obs.sidereal_time() - self.pe_moon.g_ra
        sin_alt_geo_moon = math.sin(self.pe_obs.lat) * math.sin(self.pe_moon.g_dec) +
math.cos(self.pe_obs.lat) * math.cos(self.pe_moon.g_dec) * math.cos(HA_moon)
        alt_geo_moon = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_moon))))

        # Elongasi Geosentrik Bulan & Matahari
        cos_elong_geo = math.sin(self.pe_sun.g_dec) * math.sin(self.pe_moon.g_dec) +
math.cos(self.pe_sun.g_dec) * math.cos(self.pe_moon.g_dec) * math.cos(self.pe_sun.g_ra -
self.pe_moon.g_ra)
        elong_geo = math.degrees(math.acos(max(-1.0, min(1.0, cos_elong_geo))))

        # ========================================================

        dip_deg = (1.76 * math.sqrt(max(0, self.pe_obs.elevation))) / 60.0

        # Plotting garis dan titik di ruang 3D (menggunakan visual toposentrik)
        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}"

        # Format UI Teks di sisi kiri atas Matplotlib
        teks = (
          f"LOKASI AKTIF:\n"
          f"Lat: {self.pe_obs.lat} | Lon: {self.pe_obs.lon} | Elevasi: {self.pe_obs.elevation}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:
        print(f"Error 3D Plot: {e}")

    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 di 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 = 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_start_live(self):
        self.eph3d_is_live = True

    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 limitasi tahun 1-9999 pada datetime
            now_utc = datetime.datetime.now(datetime.timezone.utc).replace(tzinfo=None)
            now_target = now_utc + datetime.timedelta(hours=tz_offset)

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

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

            # Update label teks pendamping slider
            self.lbl_eph3d_tahun.configure(text=str(now_target.year))
            self.lbl_eph3d_bulan.configure(text=str(now_target.month))
            self.lbl_eph3d_hari.configure(text=str(now_target.day))
            self.lbl_eph3d_jam.configure(text=f"{jam_des:.2f}")

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

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

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

    # 2. Sembunyikan semua input module di sidebar
    # ---> PERBAIKAN: Masukkan menu 26 dan 27 ke dalam list ini agar ikut di-hidden <---
    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]: # <--- TAMBAHKAN self.frame_winai_inputs
        f.pack_forget()

    # 3. Sembunyikan semua output module di ruang kanan secara aman
    # ---> PERBAIKAN: Indentasi digeser keluar dari loop 'for' di atas <---
    self.textbox.grid_remove()
    if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
    if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
    if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
    if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
    if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
    if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
    if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()
    if hasattr(self, 'frame_winai_out'): self.frame_winai_out.grid_remove() # <--- TAMBAHKAN INI

    self.anim_running = False
    self.is_viewing_3d = False

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

    # 4. LOGIKA PERGANTIAN MENU
    if "Visibility Hilal" in mode:
        self.frame_vis_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Laporan Analisis Hilal & KHGT")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
    elif "Visibility Map" in mode:
        self.frame_vismap_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Crescent Visibility HD Map Scanner")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", "Silakan klik 'PROSES DATA' untuk melakukan pemindaian peta global
HD. \nProses ini menghitung data spasial dan melakukan interpolasi bicubic. Harap tunggu...")
        self.textbox.configure(state="disabled")
    elif "Analisis Hilal Global" in mode:
        self.frame_gha_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Global Hilal Visibility Analyzer (Iterasi Ephem)")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E676")
```

```
elif "Moonphase" in mode:
        self.frame_moon_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Data Siklus Fase Bulan (Moonphase)")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E5FF")
    elif "Sun Moon Ephemeris" in mode:
        self.frame_eph_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Sun and Moon Ephemeris Data")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
    elif "Qiblah Direction" in mode:
        self.frame_qiblah_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Qiblah Direction & Alignment Times")
        self.textbox.grid(row=1, column=0, sticky="nsew") # Menampilkan box teks laporan
        self.textbox.configure(text_color="#FFD54F")
    elif "Moon Times" in mode:
        self.frame_mt_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Moon Rise, Transit, and Set Times")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
    elif "Sun Times" in mode:
        self.frame_st_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Sun Rise, Transit, and Set Times")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
    elif "Prayer Times" in mode:
        self.frame_pt_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Islamic Prayer Times & Twilight")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#E0E0E0")
    elif "Konversi" in mode:
        self.frame_conv_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Konversi Kalender Masehi & Hijriah (KHGT)")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#00E676")
    elif "Qibla Time" in mode:
        self.frame_qt_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Rashdul Qiblah Lokal (Waktu Arah Kiblat)")
        self.textbox.grid(row=1, column=0, sticky="nsew")
        self.textbox.configure(text_color="#FFD54F")
    elif "Analisis Gerhana" in mode:
        self.frame_gerhana_inputs.pack(fill="x", before=self.btn_hitung)
        self.lbl_main_title.configure(text="Kalkulator Gerhana Presisi (Solar & Lunar Auto-Scanner)")
        self.frame_gerhana_out.grid(row=1, column=0, sticky="nsew")
    elif "Live Animasi" in mode:
        self.frame_animasi_inputs.pack(fill="x", before=self.btn_hitung)
```

```
            self.lbl_main_title.configure(text="Live Simulator Posisi Benda Langit")
            self.frame_animasi_out.grid(row=1, column=0, sticky="nsew")
            self.btn_hitung.pack_forget() # Sembunyikan tombol proses

            # Reset ke waktu berjalan
            self.anim_start_live()

            self.anim_running = True
            self.draw_static_background()
            self.update_animation()
        elif "Sistem Sun Moon" in mode:
            self.frame_3d_inputs.pack(fill="x", before=self.btn_hitung)
            self.lbl_main_title.configure(text="3D Geocentric Space System")
            self.frame_3d_out.grid(row=1, column=0, sticky="nsew")
            self.btn_hitung.pack_forget()
            self.is_viewing_3d = True
            self.update_3d_animation()
        elif "Altitude Chart" in mode:
            self.frame_chart_inputs.pack(fill="x", before=self.btn_hitung)
            self.lbl_main_title.configure(text="Celestial Altitude Graphics")
            self.textbox.grid_remove()
            self.frame_chart_out.grid(row=1, column=0, sticky="nsew")
            self.lbl_status.configure(text="Klik PROSES DATA untuk render grafik", text_color="#00E5FF")
        elif "Kota Pertama" in mode:
            self.frame_first_point_inputs.pack(fill="x", before=self.btn_hitung)
            self.lbl_main_title.configure(text="Titik Pertama KHGT (Daratan Utama)")
            self.textbox.grid(row=1, column=0, sticky="nsew")
            self.textbox.configure(state="normal")
            self.textbox.delete("1.0", "end")
            self.textbox.insert("1.0", "Klik 'PROSES DATA' untuk memulai scanning global mencari titik
pertama di daratan utama...")
            self.textbox.configure(state="disabled")
        elif "Equinox" in mode:
            self.frame_season_inputs.pack(fill="x", before=self.btn_hitung)
            self.lbl_main_title.configure(text="Kalkulator Equinox & Solstice")
            self.textbox.grid(row=1, column=0, sticky="nsew")
            self.textbox.configure(text_color="#FFD54F")
        elif "Planetary Times" in mode:
            self.frame_planet_inputs.pack(fill="x", before=self.btn_hitung)
            self.lbl_main_title.configure(text="Rise, Transit & Set Planet")
            self.textbox.grid(row=1, column=0, sticky="nsew")
            self.textbox.configure(text_color="#00E5FF")

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

```
            self.lbl_main_title.configure(text="Simulasi Ephemeris 3D (Matahari & Bulan)")

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

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

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

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

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

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

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

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

            self.textbox.grid_remove()
```

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

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

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

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

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

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

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



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

```
            except: pass

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

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

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

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

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

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

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

            method_val = self.combo_pt_method.get()
            if "Kemenag" in method_val:
```

```
    fajr_angle, isha_angle = -20.0, -18.0
elif "Muslim World League" in method_val:
    fajr_angle, isha_angle = -18.0, -17.0
elif "ISNA" in method_val:
    fajr_angle, isha_angle = -15.0, -15.0
elif "Egypt" in method_val:
    fajr_angle, isha_angle = -19.5, -17.5
else:
    fajr_angle, isha_angle = -18.0, -18.0

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

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

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

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

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

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

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

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

def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
    diffs = alt_arr - target_alt
```

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

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

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

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

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

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

    except Exception as e:
        print(f"Error Silent Schedule: {e}")
        self.after(0, lambda e=e: self.lbl_countdown.configure(text="Gagal Kalkulasi Alarm!",
text_color="#FF5252"))
    finally:
        self._is_calculating_alarm = False

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

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

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

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

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

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

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

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

                    if diff.total_seconds() <= 1 and self.last_triggered_prayer != next_prayer:
                        self.trigger_alarm(next_prayer, next_time)
                else:
                    # Langsung beralih ke jadwal besok agar countdown ke Subuh berjalan
                    self.lbl_countdown.configure(text="Beralih ke jadwal esok hari...", text_color="#FFAB40")
                    if not getattr(self, '_is_calculating_alarm', False):
                        self._force_tomorrow = True
                        self._is_calculating_alarm = True
                        threading.Thread(target=self.update_silent_schedule, daemon=True).start()
```

```
            elif not getattr(self, '_is_calculating_alarm', False):
                self.lbl_countdown.configure(text="Menunggu Data...", text_color="#FFAB40")
        elif not self.alarm_enabled.get():
            self.lbl_countdown.configure(text="Alarm Salat: Dinonaktifkan", text_color="#FF5252")

        self.after(1000, self.alarm_tick)

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

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

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

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

        self.show_alarm_popup(prayer_name, msg)

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

        ctk.CTkLabel(popup, text=f"WAKTU {prayer_name.upper()}", font=("Segoe UI", 24, "bold"),
text_color="#00E5FF").pack(pady=(20, 10))
        ctk.CTkLabel(popup, text=msg, font=("Segoe UI", 12), wraplength=350,
justify="center").pack(pady=10)

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

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

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

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

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

    def on_city_change(self, city):
        prov = self.opt_prov.get()
        lat, lon = CITY_DB[prov][city]

        # Tambahkan self.entry_eph3d_lat & lon ke dalam daftar update otomatis
        for ent in [self.entry_vlat, self.entry_eph_lat, self.entry_qlat, self.entry_mt_lat, self.entry_st_lat,
self.entry_pt_lat, self.entry_qtlat, self.entry_eph3d_lat]:
            ent.delete(0, 'end')
            ent.insert(0, str(lat))

        for ent in [self.entry_vlon, self.entry_eph_lon, self.entry_qlon, self.entry_mt_lon, self.entry_st_lon,
self.entry_pt_lon, self.entry_qtlon, self.entry_eph3d_lon]:
            ent.delete(0, 'end')
            ent.insert(0, str(lon))
```

```
        # [TAMBAHAN BARU] Update form timezone otomatis berdasarkan zona WIB/WITA/WIT
        tz_val = self.get_tz_from_lon(lon)
        for ent in [self.entry_vtz, self.entry_eph_tz, self.entry_qtz, self.entry_mt_tz, self.entry_st_tz,
self.entry_pt_tz, self.entry_qttz]:
            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", "Minggu"][now.weekday()]
        masehi_str = f"{hari}, {now.day} {BULAN_MASEHI[now.month-1]} {now.year}"
        hijri_str = f" {HijriConverter.get_hijri_date(now)}"
        self.lbl_cal_masehi.configure(text=masehi_str)
        self.lbl_cal_hijri.configure(text=hijri_str)
        self.after(3600000, self.update_calendar_widget)

    def update_conv_ui(self):
        now = datetime.datetime.now()
        if self.radio_conv_var.get() == "m2h":
            self.lbl_conv_date.configure(text="TANGGAL MASEHI")
            self.combo_conv_month.configure(values=BULAN_MASEHI)
            self.combo_conv_month.set(BULAN_MASEHI[now.month - 1])
            self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 32)])
            self.combo_conv_day.set(f"{now.day:02d}")
            self.entry_conv_year.delete(0, 'end')
            self.entry_conv_year.insert(0, str(now.year))
        else:
            self.lbl_conv_date.configure(text="TANGGAL HIJRIAH")
            self.combo_conv_month.configure(values=BULAN_HIJRIAH)
            self.combo_conv_month.set("Ramadan")
            self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 31)])

    def show_release_info(self):
        # 1. Reset state dan sembunyikan frame lain (Dibuat kebal error / Anti-Crash)
        self.textbox.grid(row=1, column=0, sticky="nsew")
        if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
        if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
        if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
        if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
        if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
```

```
        if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
        if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()

        self.anim_running = False
        self.is_viewing_3d = False
        if hasattr(self, 'eph3d_is_live'): self.eph3d_is_live = False

        # 2. Pengaturan Dasar Textbox
        self.textbox.configure(state="normal", font=("Segoe UI", 14), wrap="word")
        self.textbox.delete("1.0", "end")

        # 3. Konfigurasi Tag Warna & Tipografi
        tb = self.textbox._textbox
        tb.tag_config("title", font=("Segoe UI", 20, "bold"), foreground="#FFD54F", justify="center",
spacing1=10, spacing3=15)
        tb.tag_config("subtitle", font=("Segoe UI", 18, "bold"), foreground="#00E5FF", justify="center",
spacing1=20, spacing3=10)
        tb.tag_config("h4", font=("Segoe UI", 16, "bold"), foreground="#FFD54F", lmargin1=15,
lmargin2=15, spacing1=15, spacing3=5)
        tb.tag_config("body", font=("Segoe UI", 14), foreground="#E2E8F0", lmargin1=15, lmargin2=15,
spacing1=5, spacing3=10)

        # Format Khusus untuk Daftar Menu
        tb.tag_config("menu_title", font=("Segoe UI", 15, "bold"), foreground="#00E676", lmargin1=25,
lmargin2=25, spacing1=12, spacing3=2)
        tb.tag_config("menu_desc", font=("Segoe UI", 14), foreground="#E2E8F0", lmargin1=45,
lmargin2=45, spacing1=2, spacing3=8)

        tb.tag_config("list_item", font=("Segoe UI", 14), foreground="#E2E8F0", lmargin1=35, lmargin2=55,
spacing1=5, spacing3=5)
        tb.tag_config("tech", font=("Consolas", 14, "bold"), foreground="#00E676")
        tb.tag_config("highlight", font=("Segoe UI", 14, "bold"), foreground="#FFD54F")
        tb.tag_config("box_info", font=("Segoe UI", 14), foreground="#00E5FF", lmargin1=40, lmargin2=60,
spacing1=5, spacing3=5)
        tb.tag_config("box_warn", font=("Segoe UI", 14), foreground="#FFD54F", lmargin1=40,
lmargin2=60, spacing1=5, spacing3=5)
        tb.tag_config("footer", font=("Segoe UI", 12, "italic"), foreground="#ffffff", justify="center")

        # ==================== HEADER ====================
        t1 = "\n"
        t2 = "INFORMASI RILIS & DOKUMENTASI SISTEM: KHGT TIMES (PRO EDITION)\n"
        t3 = "\n"
        self.textbox.insert("end", t1+t2+t3, "title")

        # ==================== INTRO ====================
```

```
        intro_1 = ("Selamat datang di KHGT Times V7.0! Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak "
                   "komprehensif dan mutakhir untuk perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal global, "
                   "dan penyatuan penanggalan Islam internasional.\n")
        self.textbox.insert("end", intro_1, "body")

        self.textbox.insert("end", "Melampaui Batas Pendahulunya (Beyond Accurate Times)\n", "h4")
        intro_2 = ("Selama bertahun-tahun, dunia falak Islam sangat bergantung pada perangkat lunak pionir seperti Accurate Times karya Eng. Mohammad Odeh dari International Astronomical Center (IAC), Yordania. Aplikasi tersebut sangat berjasa dalam meletakkan standar dasar komputasi waktu salat dan peta hilal di era awal komputasi.\n"
                   "Namun, perangkat lunak generasi pendahulu tersebut umumnya dibangun di atas teori analitik klasik (seperti algoritma VSOP87 atau Jean Meeus), menggunakan arsitektur single-thread yang lambat untuk pemindaian data massal, dan secara konseptual murni dirancang untuk kriteria rukyatul hilal lokal/zonal.\n")
        self.textbox.insert("end", intro_2, "body")

        self.textbox.insert("end", "Mengapa KHGT Times Hadir dan Sangat Penting?\n", "h4")
        intro_3 = ("KHGT Times hadir sebagai Quantum Leap (Lompatan Besar) komputasi untuk menjawab tantangan astronomi modern dan urgensi implementasi Kalender Hijriah Global Tunggal (KHGT) di seluruh dunia. Berikut adalah keunggulan absolut sistem ini dibandingkan pendahulunya:\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", "Berbeda dengan aplikasi lama yang pasif memproses satu kota, KHGT Times memiliki mesin iteratif proaktif. Sistem mampu memindai (scanning) ratusan kota di seluruh benua dalam hitungan detik untuk melacak Titik Pertama pemenuhan kriteria (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", "KHGT mensyaratkan parameter Geosentris (berpusat di inti bumi), sementara observasi visual mensyaratkan Toposentris (berpusat di mata pengamat). Aplikasi ini menghitung keduanya secara simultan dan independen tanpa kerancuan.\n", "list_item")

        self.textbox.insert("end", " ▸ ", "highlight")
        self.textbox.insert("end", "[4] Visualisasi Data HD & Ruang 3D: ", "tech")
```

```
        self.textbox.insert("end", "Menggantikan antarmuka peta resolusi rendah (pixelated) dengan Heatmap Interpolasi Spasial algoritma SciPy, serta lingkungan Simulasi Tata Surya 3D interaktif murni.\n", "list_item")

        intro_4 = ("\nDengan arsitektur ini, KHGT Times bukan lagi sekadar kalkulator waktu salat biasa, melainkan sebuah karya kontemporer dan modern 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 ====================
        s1 = "-----------------------------------------------------------------\n"
        s2 = "[ 1. TEKNOLOGI & ENGINE ASTROMETRI TERINTEGRASI ]\n"
        s3 = "-----------------------------------------------------------------\n"
        self.textbox.insert("end", "\n" + s1+s2+s3, "subtitle")

        intro_tech = ("Aplikasi KHGT Times V7.0 dibangun di atas arsitektur komputasi multi-engine. Sistem ini menggabungkan berbagai pustaka (library) saintifik standar industri dan akademik untuk mencapai keseimbangan sempurna antara akurasi tingkat tinggi (High Precision) dan kecepatan pemrosesan (High Performance).\n")
        self.textbox.insert("end", intro_tech, "body")

        techs = [
            ("Skyfield API", "Engine astrometri utama berbasis Python modern. Menghitung posisi benda langit (Geocentric & Topocentric Apparent) dengan akurasi setara observatorium angkatan laut AS (USNO), lengkap dengan koreksi aberasi cahaya, defleksi gravitasi, dan nutasi."),
            ("JPL Ephemeris", "Basis data posisi planet dan bulan resmi dari NASA Jet Propulsion Laboratory (DE421, DE440, DE442, dll). Sistem memiliki auto-switch cerdas untuk memilih akurasi file .bsp terbaik berdasarkan rentang tahun yang diinput."),
            ("PyEphem (ephem)", "Engine sekunder berbahasa C (berbasis library XEphem). Sangat ringan dan dieksekusi dengan kecepatan luar biasa. Dialokasikan khusus untuk komputasi brutal seperti iterasi pelacakan hilal global ke 300+ kota di seluruh dunia dalam hitungan detik."),
            ("SciPy & NumPy", "Pustaka komputasi numerik tingkat tinggi. NumPy menangani operasi matriks array data raksasa, sementara SciPy digunakan untuk interpolasi spasial matematis, menyulap titik-titik data kasar menjadi Peta Visibilitas Hilal (Heatmap) beresolusi tinggi (HD)."),
            ("Matplotlib", "Engine rendering visual saintifik. Berfungsi merender grafik kurva Altitude harian, mencetak kontur peta spasial, serta membangun lingkungan Simulasi Ephemeris 3D interaktif."),
            ("CustomTkinter & Tk", "Kerangka kerja antarmuka pengguna (GUI) modern dengan desain Dark-Mode yang elegan, responsif, dan nyaman di mata."),
            ("PyTZ", "Pustaka database zona waktu dunia (Olson database) untuk konversi presisi dari waktu Universal (UTC) ke Waktu Lokal (LT)."),
            ("Pillow (PIL)", "Engine pemrosesan gambar digital tingkat tinggi. Bertugas me-render tekstur visual, mask bulan real-time, hingga menjadi mesin cetak (renderer) kalender Ultra-HD secara mandiri di belakang layar."),
            ("Pygame & Win10Toast", "Modul asinkron untuk menangani sistem alarm. Memutar audio adzan jernih tanpa menghentikan (freeze) kalkulasi program, didukung notifikasi Pop-Up Windows."),
```

```
        ("FPDF & TextWrap", "Generator laporan otonom yang mengekspor data komputasi mentah ke format dokumen PDF standar akademik.")
    ]

    for name, desc in techs:
        self.textbox.insert("end", " ▸ ", "highlight")
        self.textbox.insert("end", f"{name} : ", "tech")
        self.textbox.insert("end", f"{desc}\n", "list_item")

    # ==================== BAGIAN 2: KONSEP DASAR KHGT ====================
    self.textbox.insert("end", "\n" + s1 + "[ 2. KONSEP DASAR, PRINSIP & FORMULASI KHGT ]\n" + s3, "subtitle")

    b3_intro = ("Kalender Hijriah Global Tunggal (KHGT) merupakan tonggak peradaban sistem penanggalan Islam unifikatif hasil Muktamar Internasional Turki 2016. Berbeda dengan sistem lokal/zonal (Rukyatul Hilal lokal), KHGT bertujuan menyatukan umat Islam di bawah satu sistem waktu global.\n")
    self.textbox.insert("end", b3_intro, "body")

    self.textbox.insert("end", "3 Prinsip Utama KHGT:\n", "h4")
    self.textbox.insert("end", " ▸ Kesatuan Matlak (Ittihad al-Matali' Global): ", "highlight")
    self.textbox.insert("end", "Seluruh bumi dianggap sebagai satu zona wilayah hukum penanggalan.\n", "list_item")
    self.textbox.insert("end", " ▸ Kaidah Transfer Visibilitas (Naql al-Rukyat): ", "highlight")
    self.textbox.insert("end", "Jika hilal secara astronomis mungkin dirukyat di SATU titik daratan di bumi, maka visibilitas tersebut sah dan diberlakukan untuk SELURUH dunia.\n", "list_item")
    self.textbox.insert("end", " ▸ Keterpaduan Hari: ", "highlight")
    self.textbox.insert("end", "Satu Hari Satu Tanggal. Bulan Hijriah dimulai secara serentak di seluruh dunia tanpa ada negara yang berbeda tanggal.\n", "list_item")

    self.textbox.insert("end", "Parameter Visibilitas (Imkanur Rukyat Istanbul 2016):\n", "h4")
    self.textbox.insert("end", "Agar bulan baru dapat dimulai, hasil kalkulasi ephemeris HARUS memenuhi dua syarat mutlak secara bersamaan pada saat Matahari terbenam (Sunset) di suatu daratan utama:\n", "body")
    self.textbox.insert("end", "1. Tinggi Hilal Geosentris (Geocentric Altitude) minimal 5 derajat (Alt >= 5°).\n2. Sudut Elongasi Geosentris (Geocentric Elongation) minimal 8 derajat (Elong >= 8°).\n", "box_info")

    self.textbox.insert("end", "Mekanisme Eksekusi (Parameter Kriteria Global / PKG):\n", "h4")
    self.textbox.insert("end", "Sistem algoritma KHGT Times mengevaluasi parameter di atas melalui dua lapis pengecekan (PKG 1 dan PKG 2) untuk menjaga sinkronisasi waktu Islam terhadap Garis Batas Penanggalan Internasional (International Date Line):\n", "body")

    self.textbox.insert("end", "[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 syarat ini terpenuhi, maka besok adalah awal bulan baru secara global.\n\n", "box_info")

        self.textbox.insert("end", "[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 mengaktifkan PKG 2 dengan membandingkan waktu Ijtimak (Konjungsi) terhadap Terbit Fajar di Gisborne, Selandia Baru.\n", "box_warn")
        self.textbox.insert("end", "• Jika Ijtimak TERJADI SEBELUM fajar di Gisborne: Maka besok tetap masuk bulan baru.\n• Jika Ijtimak TERJADI SETELAH fajar di Gisborne: Maka masuk bulan baru DITUNDA lusa (bulan berjalan digenapkan 30 hari). Penundaan ini wajib dilakukan agar wilayah Timur Bumi tidak memulai puasa sebelum konjungsi terjadi.\n", "box_warn")

        # ==================== BAGIAN 3: DAFTAR FITUR LENGKAP (DIRINCI) ====================
        self.textbox.insert("end", "\n" + s1 + "[ 3. DAFTAR 27 MODUL FITUR LENGKAP & KOMPREHENSIF ]\n" + s3, "subtitle")
        self.textbox.insert("end", "Berikut adalah spesifikasi detail dari ke-27 modul komputasi utama yang tersedia di sistem KHGT Times:\n", "body")

        detailed_features = [
            ("1) Visibility Hilal (KHGT)", "Modul analisis hilal tingkat lanjut pada saat matahari terbenam (sunset) di lokasi tertentu. Modul ini menghitung dan menyandingkan secara komprehensif nilai-nilai Geosentris (sebagai rujukan KHGT Global) dengan nilai Toposentris (sebagai rujukan Rukyat lokal). Meliputi umur bulan, beda azimut, ketebalan sabit, rasio iluminasi fase, hingga vonis keterpenuhan kriteria visibilitas."),
            ("2) Crescent Visibility Map", "Mesin pemindai global yang merender peta (Heatmap) visibilitas hilal berkualitas Ultra-HD menggunakan interpolasi spasial SciPy (Bicubic). Memiliki fungsionalitas rendering kurva warna untuk Altitude, Elongasi, Fraksi Iluminasi, serta menyediakan layer khusus 'Interseksi' untuk melihat arsir area persilangan yang memenuhi syarat KHGT."),
            ("3) Analisis Hilal Global", "Modul kalkulasi masif dan proaktif yang memanfaatkan library PyEphem C-Core. Mampu memindai iteratif lebih dari 300 kota dan ibukota dunia secara bersamaan (looping) dalam hitungan detik untuk melacak secara instan wilayah mana saja di bumi yang Hilalnya sudah positif (memenuhi MABIMS atau KHGT)."),
            ("4) Altitude Chart Analyser", "Menyediakan rendering grafik kurva (Line Chart) harian dari pergerakan ketinggian (Altitude) Matahari dan Bulan dalam format 2D menggunakan Matplotlib. Grafik dilengkapi dengan garis batas Ufuk (Horizon), titik persilangan terbit/terbenam, dan penanda garis vertikal interaktif yang menunjukkan posisi lintasan waktu saat ini (Real-Time Marker)."),
            ("5) Kota Pertama KHGT (Mainland)", "Sistem pelacak absolut (Auto-Tracker) yang memindai siklus ijtimak dan menyisir seluruh database daratan utama bumi (Mainland) untuk mendeteksi secara presisi titik lokasi (kota) pertama di dunia yang memvalidasi kriteria masuknya awal bulan baru KHGT melalui uji berjenjang PKG 1 dan PKG 2 (Kompensasi Fajar Gisborne)."),
            ("6) Fase Bulan (Moonphase)", "Kalkulator astronomis spesifik yang menghitung dan mencetak data presisi tingkat detik kapan terjadinya empat fase utama bulan (New Moon/Ijtimak, First Quarter, Full Moon/Purnama, Last Quarter) sepanjang 1 tahun berjalan."),
```

("7) Moon Times", "Pembangkit tabel waktu kalender sebulan penuh khusus untuk fenomena siklus Bulan. Menghitung dan menyajikan secara rapi waktu Terbit Bulan (Moonrise), Transit (Kulminasi Atas di Meridian), dan Terbenam Bulan (Moonset), dikalkulasi berdasarkan refraksi atmosfer dan ketinggian observer."),

("8) Sun Times", "Pembangkit tabel waktu kalender sebulan penuh khusus untuk Matahari. Menyajikan jadwal presisi Terbit Matahari (Sunrise), Waktu Transit/Zawal, dan Terbenam Matahari (Sunset), sangat bermanfaat untuk landasan referensi waktu sipil dan syariah."),

("9) Sun Moon Ephemeris", "Engine pembuat data mentah astronomi riil (Ephemeris). Menghasilkan log pergerakan posisi benda langit yang mencakup Right Ascension, Declination, Altitude, Azimuth, Longitude, Latitude, hingga Jarak dan Semi-Diameter secara bertahap (increment) per hari, per jam, atau bahkan per detik."),

("10) Qibla Time (Rashdul Lokal)", "Modul kalkulator bayangan kiblat lokal. Menentukan waktu spesifik pada tanggal yang dipilih kapan titik Matahari atau bayangan tegak lurus benda di suatu tempat berada tepat memotong garis Azimut (Arah) menuju Ka'bah di Makkah."),

("11) Qiblah Direction & Times", "Kalkulator Arah Kiblat geografis berakurasi ekstrem yang menggunakan model bumi Ellipsoid WGS84. Selain menghitung derajat sudut azimut kiblat, modul ini juga me-render posisi titik lokasi pengguna di atas Peta Topografi Dunia beserta rute terpendeknya ke Makkah."),

("12) Prayer Times", "Modul komputasi komprehensif Jadwal Salat Fardu (Harian & Bulanan Sebulan Penuh) yang mensimulasikan waktu fajar dan isya berdasarkan batas lengkung Astronomis Twilight. Menyediakan kustomisasi tinggi: Opsi Multi-Metode (Muhammadiyah, Kemenag, MWL, ISNA, Egypt), Opsi Mazhab Ashar (Hanafi/Syafi'i), input Suhu/Tekanan Udara, hingga kalkulasi sepertiga malam terakhir."),

("13) Konversi Kalender", "Mesin konverter tanggal dua arah (Masehi ke Hijriah dan sebaliknya). Konversi tidak didasarkan pada rumus aritmatika konvensional, melainkan kalkulasi riil dari algoritma pelacakan fase Bulan Baru (Ijtimak) global dari mesin JPL Ephemeris, menjamin keselarasan dengan ketentuan KHGT."),

("14) Analisis Gerhana", "Algoritma sistem cerdas (Syzygy scanner) yang mendeteksi kedudukan node Matahari dan Bulan untuk melacak secara otonom kapan terjadinya Gerhana Matahari dan Bulan dalam rentang 1 tahun. Modul ini merinci klasifikasi gerhana (Total/Cincin/Sebagian), visibilitasnya di Indonesia, hingga kapabilitas mutakhir mengekspor garis Jalur Sentral Umbra Gerhana ke format file KML untuk disimulasikan di Google Earth."),

("15) Live Animasi", "Simulasi interaktif grafis 2D yang memvisualisasikan kedudukan riil Matahari dan Bulan terhadap Garis Ufuk (Horizon) tempat pengamat berdiri. Dilengkapi Engine Deteksi Waktu Dinamis yang mampu memutar waktu ke format 'Live' maupun 'Kustom'. Terdapat auto-kalkulasi Offset Zona Waktu (UTC) berbasis titik bujur (longitude)."),

("16) Sistem Sun Moon Earth", "Simulator spasial tata surya 3D murni berbasis grafis geometri kanvas. Modul ini menghadirkan interaksi kamera di mana pengguna dapat menggeser/merotasi layar (drag) dan melakukan pembesaran (Scroll/Zoom) untuk melihat rasio jarak serta garis proyeksi orbit."),

("17) Equinox & Solstice", "Kalkulator fisika astronomis presisi ultra yang melacak detik-detik terjadinya deklinasi 0 derajat dan titik ekstrem balik Matahari (Equinox Musim Semi/Gugur dan Solstice Musim Panas/Dingin). Referensi wajib penanggalan sipil tropis."),

("18) Planetary Times", "Sama seperti modul Sun/Moon Times, namun dioptimalkan untuk memproses dan menelusuri data risings (terbit) dan settings (terbenam) untuk benda-benda tata surya lainnya seperti Merkurius, Venus, Mars, Jupiter, Saturnus, Uranus, Neptunus, hingga Pluto."),

("19) Simulasi Ephemeris 3D", "Engine visualisasi 3D alternatif berkinerja tinggi yang ditenagai oleh library Matplotlib. Modul ini mengkombinasikan fungsi slider interaktif waktu dan kalkulasi multi-parameter secara live. Mampu merender Wireframe grid bumi dan menggambar titik lintasan Matahari-Bulan dalam proyeksi bola langit."),

("20) Komparasi 50 Tahun (Ramadhan)", "Mesin analitik super-komputasi (Batch Processor) yang menelusuri sejarah ke depan selama 50 Tahun. Mengkomparasi jatuhnya 1 Ramadhan secara otomatis antara hitungan KHGT (Matlak Global Geo) disandingkan dengan hitungan Neo MABIMS (Toposentrik di titik kritis Ufuk Barat Indonesia/Sabang)."),

("21) Komparasi 50 Tahun (Syawal)", "Serupa dengan modul 20, namun difokuskan secara khusus untuk penelusuran dan komparasi jatuhnya 1 Syawal (Idul Fitri) selama rentang 50 Tahun ke depan, memperlihatkan potensi perbedaan (istikmal) antara KHGT dan MABIMS lokal."),

("22) Tabel Tinggi Hilal 50 Thn (Ramadhan)", "Modul ekstraksi data tingkat lanjut yang menghasilkan tabel 50 tahun (1447 H - 1496 H) khusus untuk memuat parameter Ketinggian (Altitude) Hilal di awal 1 Ramadhan secara terstruktur."),

("23) Tabel Elongasi Hilal 50 Thn (Ramadhan)", "Modul ekstraksi data pemfokusan parameter sudut Elongasi bulan-matahari selama 50 tahun berturut-turut untuk mengevaluasi posisi awal bulan 1 Ramadhan secara terpadu."),

("24) Tabel Tinggi Hilal 50 Thn (Syawal)", "Modul komputasi ekstraksi data untuk melacak dan merangkum Ketinggian (Altitude) Hilal selama 50 tahun ke depan khusus pada penentuan awal 1 Syawal (Idul Fitri)."),

("25) Tabel Elongasi Hilal 50 Thn (Syawal)", "Modul pemrosesan data panjang yang dikhususkan untuk melacak parameter Elongasi Geosentris dan Toposentris selama 50 tahun berturut-turut pada akhir Ramadhan/awal 1 Syawal."),

("26) Kalender Hijriah Berjalan", "Modul interaktif visual kalender satu layar penuh. Menampilkan penanggalan Hijriah (hasil kalkulasi KHGT) sebagai entitas utama di tengah kotak, berdampingan dengan tanggal Masehi pendamping. Menyediakan fitur cetak ke format PDF & PNG beresolusi tinggi yang ditenagai algoritma gambar independen."),

("27) Kalender Masehi Berjalan", "Modul kebalikan dari menu 26. Memproyeksikan penanggalan sipil (Masehi/Gregorian) secara dominan dengan hari libur Ahad berwarna merah, didampingi data konversi tanggal Hijriah di sudutnya. Turut dibekali kapabilitas ekspor Ultra-HD yang tangguh."),

("28) WinAI: Aplikasi AI Windows", "Akses aplikasi Windows yang terintegrasi dengan AI Gemini melalui internet. Dalam versi 7.0 ini WinAI sudah terintegrasi dengan kode program KHGT Times, sehingga hasil hisab bisa ditanyakan langsung di WinAI.")

]

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" + s1 + "[4. HIGHLIGHT PEMBARUAN EKSKLUSIF (VERSI 7.0)]\n" + s3, "subtitle")

updates = [

--- PEMBARUAN VERSI 7.0 ---

```
        ("[V7.0 - MODUL 26 & 27] Dual-Calendar Suite Interaktif", "Ekspansi fitur besar dengan menambahkan dua modul kalender visual berjalan (Hijriah & Masehi). Keduanya didesain dengan antarmuka elegan, offset tata letak matriks otomatis (Senin-Ahad sinkron), serta membaca data langsung dari pustaka KHGT."),
        ("[V7.0 - EXPORT HD] Rendering Kalender Independen", "Sistem cetak untuk Modul 26 & 27 tidak mengambil 'screenshot' layar yang rawan blur. Dibangun menggunakan engine Pillow (PIL) murni pada kanvas 1400x1000 pixel, sehingga hasil cetak PDF/PNG tetap tajam (Ultra-HD) dan bebas distorsi."),
        ("[V7.0 - ALARM NOTIFIKASI] Auto-Active Background System", "Fungsi notifikasi waktu salat dan pemutar Audio Adzan kini diatur menyala (Aktif) secara default sesaat setelah aplikasi dijalankan. Anda tidak perlu lagi mengaktifkannya secara manual tiap kali membuka aplikasi."),
        # --- PEMBARUAN VERSI 7.0 ---
        ("[V7.0 - VISMAP] Layer Peta Interseksi KHGT", "Penambahan layer baru 'Interseksi' pada modul Crescent Visibility Map. Fitur ini secara presisi mengarsir area hijau di atas peta dunia yang memenuhi kedua kriteria KHGT secara simultan (Altitude >= 5° & Elongasi >= 8°), lengkap dengan penggambaran garis batas (contour lines) secara independen."),
        ("[V7.0 - VISMAP] Resolusi Interpolasi Ultra-HD (0.1°)", "Rapatan grid interpolasi SciPy ditingkatkan tajam dari 0.25° menjadi 0.1°. Hasil render garis lengkung (curve) batas visibilitas hilal dan heatmap iluminasi kini sangat halus (smooth) dan jauh lebih akurat terhadap topografi geografis."),
        ("[V7.0 - MODUL 19-25] Ekspansi Simulasi 3D & Super-Komputasi", "Penambahan 7 modul baru termasuk visualisasi Ephemeris Matplotlib 3D, serta mesin komparasi multi-dekade untuk Ramadhan dan Syawal secara terpisah (ketinggian dan elongasi)."),
        ("[V7.0 - ALARM NOTIFIKASI] Threading Audio Adzan", "Pembaruan arsitektur Background Threading untuk menyertakan Notifikasi Windows (Toast) & Pemutar Audio Adzan (Pygame) tanpa membekukan layar GUI."),
        ("[V7.0 - TIMEZONE ENGINE] Dynamic UTC Offset", "Mesin waktu pada fitur Live Animasi (Modul 15) dan 3D sekarang mendeteksi zona waktu (UTC) secara dinamis berdasarkan garis Bujur (Longitude) kota yang dipilih, menggantikan sistem statis WIB."),
        ("[V7.0 - ENGINE/MAPS] Eliminasi Cartopy", "Peta HD sepenuhnya di-render ulang menggunakan SciPy murni (Griddata) demi kompatibilitas penuh (aman 100%) jika program dicompile menjadi berkas instalasi .EXE."),
        ("[V7.0 - EXPORT] Integrasi Standar Akademik", "Peningkatan fitur ekspor lintas modul: Mulai dari Kalender .ICS (lengkap dengan trigger pengingat 10 menit), cetak Laporan FPDF, ekspor Gambar .PNG, hingga ekspor jalur vektor Gerhana format .KML untuk Google Earth.")
    ]
    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", s1 + "[ 5. PANDUAN OPERASIONAL & EXPORT ]\n" + s3, "subtitle")

    guides = [
        ("NAVIGASI MODUL", "Gunakan menu dropdown 'KHGT ENGINE' di sidebar kiri untuk beralih antar 27 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan input yang tidak perlu agar layar selalu rapi."),
```

```
        ("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 mendeteksi titik GPS dan elevasi Anda saat ini."),
        ("EKSEKUSI DATA", "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter input lalu klik tombol biru besar '▶ PROSES DATA' di panel bawah. Tombol ini otomatis akan bersembunyi jika Anda sedang memasuki modul Live/Real-time atau modul Kalender Interaktif."),
        ("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 dan Zoom Out."),
        ("AUDIO ADZAN", "Secara default sistem membaca file 'adzan.mp3' di direktori utama dan langsung aktif berkat Auto-Alarm. Anda dapat menggantinya kapan pun menggunakan tombol ' 🎵 Pilih Audio Adzan' di bagian Pengaturan Alarm Salat."),
        ("SISTEM EXPORT", "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di dalam kanvas): \n  [💾 TXT] Menyimpan teks hasil komputasi laporan saat ini ke file Notepad.\n  [📸 PNG] Menyimpan dokumen laporan analitik dalam format gambar PNG.\n  [📄 PDF] Mencetak dokumen laporan analitik atau Kalender HD murni.\n  [🏠 RILIS] Tombol reset darurat untuk membersihkan layar dan 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================================================================================
===============\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 +=
"================================================================================
===========\n"
    self.textbox.insert("end", footer_text, "footer")

    # 4. Kunci Textbox dan update judul layar utama
    self.textbox.configure(state="disabled")
    self.lbl_main_title.configure(text="KHGT Times V7.0 - Pro Edition")

  def create_input_row(self, parent, label_text, default_val):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)
    ctk.CTkLabel(row, text=label_text, font=("Segoe UI", 12)).pack(side="left")
    entry = ctk.CTkEntry(row, width=80, justify="right")
    entry.delete(0, 'end')
    entry.insert(0, default_val)
```

```
    entry.pack(side="right")
    return entry

def create_ymd_row(self, parent, dy, dm, dd):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)
    y = ctk.CTkEntry(row, width=45, placeholder_text="Y")
    y.delete(0, 'end')
    y.insert(0, dy)
    y.pack(side="left", padx=2)
    m = ctk.CTkEntry(row, width=35, placeholder_text="M")
    m.delete(0, 'end')
    m.insert(0, dm)
    m.pack(side="left", padx=2)
    d = ctk.CTkEntry(row, width=35, placeholder_text="D")
    d.delete(0, 'end')
    d.insert(0, dd)
    d.pack(side="left", padx=2)
    return y, m, d

def create_hms_row(self, parent, dh, dm, ds):
    row = ctk.CTkFrame(parent, fg_color="transparent")
    row.pack(fill="x", padx=10, pady=2)
    h = ctk.CTkEntry(row, width=35, placeholder_text="h")
    h.delete(0, 'end')
    h.insert(0, dh)
    h.pack(side="left", padx=2)
    m = ctk.CTkEntry(row, width=35, placeholder_text="m")
    m.delete(0, 'end')
    m.insert(0, dm)
    m.pack(side="left", padx=2)
    s = ctk.CTkEntry(row, width=35, placeholder_text="s")
    s.delete(0, 'end')
    s.insert(0, ds)
    s.pack(side="left", padx=2)
    return h, m, s

def auto_switch_ephemeris(self, target_year):
    # Daftar urutan prioritas file beserta rentang tahun amannya (min_year, max_year)
    ephemeris_priority = [
        ("de421.bsp", 1900, 2050),      # Paling ringan & optimal untuk masa kini
        ("de442.bsp", 1550, 2650),      # Alternatif resolusi tinggi menengah
        ("de406.bsp", -3000, 3000)      # 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 ephemeris {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 "Altitude Chart" in mode:
            threading.Thread(target=self.calculate_chart_analyser, daemon=True).start()
            return

        if "Visibility Map" not in mode and "Analisis Gerhana" not in mode and "Live Animasi" not in mode
and "Sistem Sun Moon" not in mode and "Simulasi Ephemeris" not in mode:
            self.textbox.configure(state="normal")
            self.textbox.delete("1.0", "end")
            self.textbox.insert("1.0", "Memproses data astrometri...\n")
            self.textbox.configure(state="disabled")

        if "Visibility Hilal" in mode:
            threading.Thread(target=self.calculate_visibility, daemon=True).start()
        elif "Visibility Map" in mode:
            threading.Thread(target=self.calculate_visibility_map, daemon=True).start()
        elif "Analisis Hilal Global" in mode:
            threading.Thread(target=self.calculate_global_hilal, daemon=True).start()
        elif "Moonphase" in mode:
            threading.Thread(target=self.calculate_moonphase, daemon=True).start()
        elif "Sun Moon Ephemeris" in mode:
            threading.Thread(target=self.calculate_ephemeris, daemon=True).start()
        elif "Qiblah Direction" in mode:
            threading.Thread(target=self.calculate_qiblah, daemon=True).start()
        elif "Moon Times" in mode:
            threading.Thread(target=self.calculate_moontimes, daemon=True).start()
        elif "Sun Times" in mode:
            threading.Thread(target=self.calculate_suntimes, daemon=True).start()
        elif "Prayer Times" in mode:
            threading.Thread(target=self.calculate_prayertimes, daemon=True).start()
        elif "Konversi" in mode:
            threading.Thread(target=self.calculate_conversion, daemon=True).start()
        elif "Qibla Time" in mode:
            threading.Thread(target=self.calculate_qiblatime, daemon=True).start()
        elif "Analisis Gerhana" in mode:
            tahun = int(self.combo_tahun_gerhana.get())
            threading.Thread(target=self._calc_gerhana_thread, args=(tahun,), daemon=True).start()
        elif "Kota Pertama" in mode:
            threading.Thread(target=self.calculate_first_point, daemon=True).start()
        elif "Equinox" in mode:
            threading.Thread(target=self.calculate_seasons, daemon=True).start()
        elif "Planetary Times" in mode:
```

```
        threading.Thread(target=self.calculate_planetary_times, daemon=True).start()
    # ---> URUTAN YANG BENAR DI TOMBOL PROSES (SYAWAL HARUS DI ATAS) <---
    elif "Komparasi 50 Tahun (Syawal)" in mode:
        self.analisis_komparasi_syawal_50_tahun()

    elif "Komparasi 50 Tahun" in mode:
        self.analisis_komparasi_ramadhan_50_tahun()

    elif "Tabel Tinggi Hilal 50 Thn (Syawal)" in mode:
        threading.Thread(target=self._proses_tabel_ketinggian_syawal_50_tahun, daemon=True).start()

    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()


# 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} {time_str}"

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

        waktu_ijtimak = ephem.previous_new_moon(tanggal_referensi_utc)
        str_ijtimak = str(ephem.Date(waktu_ijtimak))

        output_lines = []
        header = f"{self.get_header(138)}\n"
        header += f"{'[ Analisis Hilal Global (KHGT, MABIMS, Wujudul Hilal) - Ephem ]'.center(138)}\n\n"
        header += f"* Tanggal Referensi Pencarian (UTC): {tanggal_referensi_utc}\n"
        header += f"* Waktu Ijtimak/Konjungsi Terdekat (UTC): {str_ijtimak}\n"
```

```
        header += f"* Catatan: KHGT dinilai dari parameter Geo, MABIMS dinilai dari parameter Topo.\n"
        header += "="*138 + "\n"
        header += f"{'Negara':<15} | {'Kota':<14} | {'Sunset (UTC)':<16} | {'Umur Bln':<9} | {'Alt(Geo)':<8}
| {'Eln(Geo)':<8} | {'Alt(Top)':<8} | {'Eln(Top)':<8} | Status\n"
        header += "-"*138
        output_lines.append(header)

        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)

                try:
                    pengamat.date = ephem.Date(ephem.Date(tanggal_referensi_utc) - 0.5)
                    waktu_sunset = pengamat.next_setting(matahari)
                except (ephem.AlwaysUpError, ephem.NeverUpError):
                    output_lines.append(f"{negara[:15]:<15} | {nama_kota[:14]:<14} | {'Anomali Ekstrem':<16}
| {'-':<9} | {'-':<8} | {'-':<8} | {'-':<8} | {'-':<8} | Lintang Tinggi (Midnight Sun)")
                    continue

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

                # 1. Parameter Toposentris (Bawaan Ephem)
                alt_topo = math.degrees(bulan.alt)
                elong_topo = math.degrees(ephem.separation(matahari, bulan))

                # 2. Parameter Geosentris (Kalkulasi dari Apparent Geocentric)
                HA_moon = pengamat.sidereal_time() - bulan.g_ra
                sin_alt_geo = math.sin(pengamat.lat) * math.sin(bulan.g_dec) + math.cos(pengamat.lat) *
math.cos(bulan.g_dec) * math.cos(HA_moon)
                alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

                cos_elong_geo = math.sin(matahari.g_dec) * math.sin(bulan.g_dec) +
math.cos(matahari.g_dec) * math.cos(bulan.g_dec) * math.cos(matahari.g_ra - bulan.g_ra)
                elong_geo = math.degrees(math.acos(max(-1.0, min(1.0, cos_elong_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 (Bawah Ufuk)"

            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))

            output_lines.append(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}")

        output_lines.append("="*138)
        self.after(0, self.display_result, "\n".join(output_lines))

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

# ========================================
# MODUL TAMBAHAN 11: ANALISIS GERHANA ALGORITMIK
# ========================================
def setup_gerhana_out_frame(self):
    self.frame_gerhana_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")

    style = ttk.Style(self)
    style.theme_use("clam")

    # 1. Mengubah font Judul Kolom (Heading) menjadi ukuran 13
    style.configure("Gerhana.Treeview.Heading", font=('Segoe UI', 13, 'bold'), background="#2b5797",
foreground="white")

    # 2. Mengubah font Isi Tabel menjadi ukuran 13 dan tinggi baris (rowheight) menjadi 32
```

```
    style.configure("Gerhana.Treeview", font=('Segoe UI', 13), rowheight=32, background="#1e1e1e",
foreground="white", fieldbackground="#1e1e1e")

    style.map('Gerhana.Treeview', background=[('selected', '#0078D7')], foreground=[('selected',
'white')])

    frame_tabel = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
    frame_tabel.pack(fill="both", expand=True, pady=(0, 10))

    kolom_gerhana = ("objek", "jenis", "mulai", "puncak", "akhir", "wilayah_global", "indo_vis")
    self.tabel_gerhana = ttk.Treeview(frame_tabel, columns=kolom_gerhana, show="headings",
style="Gerhana.Treeview")

    self.tabel_gerhana.heading("objek", text="Objek")
    self.tabel_gerhana.heading("jenis", text="Jenis Gerhana")
    self.tabel_gerhana.heading("mulai", text="Kontak Awal (WIB)")
    self.tabel_gerhana.heading("puncak", text="Puncak (WIB)")
    self.tabel_gerhana.heading("akhir", text="Kontak Akhir (WIB)")
    self.tabel_gerhana.heading("wilayah_global", text="Karakteristik & Wilayah")
    self.tabel_gerhana.heading("indo_vis", text="Visibilitas (WIB Siang/Malam)")

    self.tabel_gerhana.column("objek", width=80, anchor="center")
    self.tabel_gerhana.column("jenis", width=120, anchor="center")
    self.tabel_gerhana.column("mulai", width=130, anchor="center")
    self.tabel_gerhana.column("puncak", width=130, anchor="center")
    self.tabel_gerhana.column("akhir", width=130, anchor="center")
    self.tabel_gerhana.column("wilayah_global", width=220, anchor="w")
    self.tabel_gerhana.column("indo_vis", width=260, anchor="w")

    self.tabel_gerhana.tag_configure('ganjil', background='#2b2b2b')
    self.tabel_gerhana.tag_configure('genap', background='#1e1e1e')

    sb_gerhana = ttk.Scrollbar(frame_tabel, orient="vertical", command=self.tabel_gerhana.yview)
    self.tabel_gerhana.configure(yscroll=sb_gerhana.set)

    self.tabel_gerhana.pack(side="left", fill="both", expand=True)
    sb_gerhana.pack(side="right", fill="y")

    frame_btn = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
    frame_btn.pack(pady=10)

    btn_detail = ctk.CTkButton(frame_btn, text="🔭 Lihat Detail Lokal Saat Puncak (Berdasarkan GPS
Observer)", font=("Segoe UI", 12, "bold"), command=self.tampilkan_detail_lokal_gerhana)
    btn_detail.pack(side="left", padx=10)
```

```
        self.btn_kml = ctk.CTkButton(frame_btn, text="🗺️ Export Jalur Totalitas/Anularitas ke KML",
font=("Segoe UI", 12, "bold"), fg_color="#F57C00", hover_color="#EF6C00",
command=self.export_kml_solar)
        self.btn_kml.pack(side="left", padx=10)

    def insert_gerhana_wrapped_row(self, values, tags):
        char_limits = [10, 16, 18, 18, 18, 30, 36]
        wrapped_cols = []
        max_lines = 1

        for i, val in enumerate(values):
            wrapped = textwrap.wrap(str(val), width=char_limits[i])
            if not wrapped: wrapped = [""]
            wrapped_cols.append(wrapped)
            if len(wrapped) > max_lines: max_lines = len(wrapped)

        for line_idx in range(max_lines):
            row_data = [col[line_idx] if line_idx < len(col) else "" for col in wrapped_cols]
            self.tabel_gerhana.insert("", "end", values=row_data, tags=tags)

        self.tabel_gerhana.insert("", "end", values=["", "", "", "", "", "", ""], tags=tags)

    def analisis_visibilitas_indonesia(self, jam_utc):
        if 10 <= jam_utc <= 14: return "Terlihat Jelas: Papua, Maluku, Sulawesi. (Sumatra/Jawa saat terbit)"
        elif 15 <= jam_utc <= 19: return "Terlihat Jelas Seluruh Indonesia (Tengah Malam)"
        elif 20 <= jam_utc <= 22: return "Terlihat Jelas: Sumatera, Jawa, Kalimantan. (Indonesia Timur saat
terbenam)"
        else: return "Tidak Terlihat (Terjadi Siang Hari di Indonesia)"

    def _calc_gerhana_thread(self, tahun):
        try:
            # AUTO-SWITCH: Memastikan de406 dimuat jika tahun ekstrem
            self.auto_switch_ephemeris(tahun)

            # PROTEKSI: Skyfield utc() mendukung tahun negatif secara native
            t0 = self.ts.utc(tahun, 1, 1)
            t1 = self.ts.utc(tahun, 12, 31, 23, 59, 59)

            all_eclipses = []

            # --- 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:
                # Handle jika hasil return skalar (hanya 1 gerhana) atau array
```

```
            t_bulan_arr = np.atleast_1d(t_bulan)
            y_bulan_arr = np.atleast_1d(y_bulan)
            for t, y in zip(t_bulan_arr, y_bulan_arr):
                all_eclipses.append({
                    'objek': 'Bulan',
                    'jenis': jenis_bulan_map.get(int(y), 'Unknown'),
                    't_peak': t,
                    't_mulai': self.ts.tt_jd(t.tt - 0.08), # Estimasi -2 jam
                    't_akhir': self.ts.tt_jd(t.tt + 0.08), # Estimasi +2 jam
                    'wilayah': 'Global (Sisi Malam)'
                })

        # --- 2. PENCARIAN GERHANA MATAHARI ---
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
        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:
                # Scan 12 jam di sekitar konjungsi
                tt_array = np.linspace(t_nm.tt - 0.25, t_nm.tt + 0.25, 500)
                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)"

                    all_eclipses.append({
                        'objek': 'Matahari',
                        'jenis': jenis,
                        't_peak': t_p,
                        't_mulai': self.ts.tt_jd(t_p.tt - 0.1),
                        't_akhir': self.ts.tt_jd(t_p.tt + 0.1),
                        'wilayah': f"Sep: {seps[min_idx]:.2f}°"
                    })
```

```
            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

        for ev in all_eclipses:
            # Konversi ke WIB (+7) manual lewat Julian Date untuk keamanan tahun minus
            t_local = self.ts.tt_jd(ev['t_peak'].tt + 7.0/24.0)
            y, m, d, h, mn, s = t_local.utc

            # Format Tahun: Jika y <= 0, maka y-1 adalah tahun SM (BCE)
            yr = int(y)
            txt_yr = f"{yr}" if yr > 0 else f"{abs(yr-1)} SM"
            waktu_str = f"{int(d):02d}-{int(m):02d}-{txt_yr} {int(h):02d}:{int(mn):02d}"

            tag = 'ganjil' if count % 2 == 0 else 'genap'
            self.insert_gerhana_wrapped_row((
                ev['objek'], ev['jenis'], "---", waktu_str, "---", ev['wilayah'], "Cek Detail"
            ), (tag,))
            count += 1

        self.lbl_status.configure(text=f"Analisis Gerhana {tahun} Selesai", text_color="#00E676")
        self.btn_hitung.configure(state="normal")

    def export_kml_solar(self):
        selected_item = self.tabel_gerhana.selection()
        if not selected_item:
            messagebox.showwarning("Peringatan", "Silakan pilih jadwal Gerhana Matahari di tabel terlebih
dahulu.")
            return

        vals = self.tabel_gerhana.item(selected_item[0])['values']
        if "Matahari" not in str(vals[0]):
            messagebox.showinfo("Info", "Pilih data Gerhana Matahari, bukan Bulan.")
            return
        if "Sebagian" in str(vals[1]):
            messagebox.showinfo("Info", "Gerhana Sebagian (Partial) tidak memiliki Jalur Sentral
Umbra/Antumbra (Totality/Annular).")
            return
```

```
        waktu_puncak_str = str(vals[3]).strip()
        jenis = str(vals[1]).strip()

        try:
            tz_wib = pytz.timezone('Asia/Jakarta')
            dt_naive = datetime.datetime.strptime(waktu_puncak_str, "%d-%m-%Y %H:%M")
            dt_wib = tz_wib.localize(dt_naive)
            t_peak = self.ts.from_datetime(dt_wib)

            t_start = self.ts.tt_jd(t_peak.tt - 4.0/24.0)
            t_end = self.ts.tt_jd(t_peak.tt + 4.0/24.0)
            t_arr = self.ts.tt_jd(np.linspace(t_start.tt, t_end.tt, 1000))

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

            e_pos = earth.at(t_arr)
            M = e_pos.observe(moon).position.km
            S = e_pos.observe(sun).position.km

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

            M_dot_v = np.sum(M * v_hat, axis=0)
            M_sq = np.sum(M**2, axis=0)

            R_E = 6371.0
            b = 2.0 * M_dot_v
            c = M_sq - R_E**2
            delta = b**2 - 4.0 * c

            valid_idx = delta >= 0
            if not np.any(valid_idx):
                messagebox.showwarning("Peringatan", "Jalur sentral meleset dari permukaan Bumi (tidak ada
daratan yang dilewati sumbu bayangan pusat).")
                return

            b_val = b[valid_idx]
            delta_val = delta[valid_idx]
            M_val = M[:, valid_idx]
            v_hat_val = v_hat[:, valid_idx]
            t_valid = t_arr.tt[valid_idx]

            k = (-b_val - np.sqrt(delta_val)) / 2.0
            P_km = M_val + v_hat_val * k
```

```
        P_au = P_km / 149597870.7

        coords_list = []
        for i in range(len(t_valid)):
            geo = Geocentric(position_au=P_au[:, i], t=self.ts.tt_jd(t_valid[i]))
            subpt = wgs84.subpoint(geo)
            coords_list.append(f"{subpt.longitude.degrees},{subpt.latitude.degrees},0")

        kml_coords = "\n              ".join(coords_list)

        kml_content = f"""<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Jalur Gerhana Matahari {waktu_puncak_str}</name>
    <description>Tipe: {jenis}</description>
    <Style id="pathStyle">
      <LineStyle>
        <color>7f0000ff</color> <width>5</width>
      </LineStyle>
    </Style>
    <Placemark>
      <name>Path of Totality / Annularity (Center Line)</name>
      <styleUrl>#pathStyle</styleUrl>
      <LineString>
        <tessellate>1</tessellate>
        <coordinates>
          {kml_coords}
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>"""

        filepath = filedialog.asksaveasfilename(
            initialfile=f"Jalur_Gerhana_Matahari_{dt_naive.strftime('%Y%m%d')}.kml",
            defaultextension=".kml",
            filetypes=[("KML Files", "*.kml")]
        )

        if filepath:
            with open(filepath, 'w', encoding='utf-8') as f:
                f.write(kml_content)
            messagebox.showinfo("Sukses", f"File Jalur Gerhana KML berhasil
diekspor:\n{filepath}\n\nSilakan buka file ini menggunakan Google Earth untuk melihat simulasi
jalurnya.")
```

```
        except Exception as e:
            import traceback
            traceback.print_exc()
            messagebox.showerror("Error", f"Gagal export KML: {e}")

    def tampilkan_detail_lokal_gerhana(self):
        selected_item = self.tabel_gerhana.selection()
        if not selected_item:
            messagebox.showwarning("Peringatan", "Silakan klik/pilih salah satu jadwal gerhana di tabel
terlebih dahulu.")
            return

        item_values = self.tabel_gerhana.item(selected_item[0])['values']
        if not item_values or item_values[0] == "": return

        objek = str(item_values[0]).strip()
        waktu_puncak_str = str(item_values[3]).strip()
        if not waktu_puncak_str: return

        try:
            tz_wib = pytz.timezone('Asia/Jakarta')
            dt_naive = datetime.datetime.strptime(waktu_puncak_str, "%d-%m-%Y %H:%M")
            dt_wib = tz_wib.localize(dt_naive)
            waktu_puncak_skyfield = self.ts.from_datetime(dt_wib)

            earth = self.eph['earth']
            target_objek = self.eph['moon'] if "Bulan" in objek else self.eph['sun']

            try:
                lat = float(self.entry_vlat.get())
                lon = float(self.entry_vlon.get())
            except:
                lat, lon = -7.0667, 110.4100

            lokasi_observasi = earth + wgs84.latlon(lat, lon)
            astrometric = lokasi_observasi.at(waktu_puncak_skyfield).observe(target_objek)
            alt, az, distance = astrometric.apparent().altaz()

            extra_info = ""
            if "Bulan" in objek:
                iluminasi = almanac.fraction_illuminated(self.eph, 'moon', waktu_puncak_skyfield) * 100.0
                dt_utc_peak = waktu_puncak_skyfield.utc_datetime()
                t0 = self.ts.utc(dt_utc_peak - datetime.timedelta(days=35))
                t1 = waktu_puncak_skyfield
```

```
            fase_t, fase_y = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
            waktu_new_moon = [t for t, y in zip(fase_t, fase_y) if y == 0]

            if waktu_new_moon:
                last_new_moon = waktu_new_moon[-1]
                moon_age_actual_days = waktu_puncak_skyfield.tt - last_new_moon.tt

                app_moon = earth.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
                app_sun = earth.at(waktu_puncak_skyfield).observe(self.eph['sun']).apparent()
                _, m_lon, _ = app_moon.ecliptic_latlon()
                _, s_lon, _ = app_sun.ecliptic_latlon()

                phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
                moon_age_phase_days = (phase_angle / 360.0) * 29.530588861

                extra_info = f"\nIlluminasi (Fase): {iluminasi:.2f}%\nUmur Aktual (True Time Elapsed):
{moon_age_actual_days:.4f} Hari\nUmur Fase Sudut (Siklik/Elongasi): {moon_age_phase_days:.4f} Hari"
            else:
                extra_info = f"\nIlluminasi (Fase): {iluminasi:.2f}%\nUmur Bulan: N/A"
        else:
            app_moon = lokasi_observasi.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
            m_alt, m_az, _ = app_moon.altaz()
            elong = astrometric.separation_from(app_moon).degrees
            extra_info = f"\nAltitude Bulan: {m_alt.degrees:.2f}°\nElongasi Sudut (Toposentrik):
{elong:.2f}°\n"

            if alt.degrees > 0:
                extra_info += ">> MATAHARI DI ATAS UFUK: Gerhana mungkin teramati dari lokasi ini jika
elongasi mendekati 0°."
            else:
                extra_info += ">> MATAHARI DI BAWAH UFUK: Gerhana terjadi saat malam di lokasi ini."

        pesan = (
            f"LOKASI OBSERVASI (GPS)\n"
            f"{self.lokasi_nama.get()}\n"
            f"{'-'*40}\n"
            f"Waktu Puncak Global (WIB): {waktu_puncak_str}\n"
            f"Waktu Puncak Global (UTC): {waktu_puncak_skyfield.utc_strftime('%Y-%m-%d
%H:%M:%S')}\n"
            f"{'-'*40}\n"
            f"PARAMETER VISUAL {objek.upper()} (TOPOSENTRIK)\n"
            f"Altitudo (Tinggi ufuk): {alt.degrees:.2f}°\n"
            f"Azimuth (Arah kompas): {az.degrees:.2f}°\n"
            f"Jarak Bumi-{objek}: {distance.km:,.0f} km\n"
            f"{extra_info}"
        )
```

```
            win_detail = ctk.CTkToplevel(self)
            win_detail.title(f"Detail Parameter Lokal - {objek}")
            win_detail.geometry("550x450")
            win_detail.attributes("-topmost", True)

            lbl_title = ctk.CTkLabel(win_detail, text=f"Detail Visual {objek} Saat Puncak", font=("Segoe UI", 16,
"bold"), text_color="#00E5FF")
            lbl_title.pack(pady=(15, 5))

            textbox = ctk.CTkTextbox(win_detail, width=500, height=350, font=("Consolas", 13),
wrap="word", fg_color="#1e1e1e")
            textbox.pack(padx=20, pady=10, fill="both", expand=True)
            textbox.insert("1.0", pesan)
            textbox.configure(state="disabled")

        except Exception as e:
            messagebox.showerror("Error", f"Terjadi kesalahan saat menghitung detail parameter: {e}")

    # ==========================================
    # MODUL TAMBAHAN 12: LIVE ANIMASI
    # ==========================================
    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)
        self.lbl_anim_lokasi = ctk.CTkLabel(frame_atas, textvariable=self.lokasi_nama, font=("Consolas",
12), text_color="#00E5FF")
        self.lbl_anim_lokasi.pack(side="right", padx=15)

        self.anim_canvas = tk.Canvas(self.frame_animasi_out, bg='#030514', highlightthickness=0)
        self.anim_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))
        self.anim_canvas.bind("<Configure>", self.draw_static_background)

    def draw_static_background(self, event=None):
        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  # <-- Padding aman di kiri & kanan layar
```

```
        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")

        # Gambar Skala Kompas Azimuth (0-360)
        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  # <-- Koordinat X disesuaikan dengan Padding
            self.anim_canvas.create_text(x, horizon_y + 15, text=label, fill="white", font=("Consolas", 9),
tags="static")
            self.anim_canvas.create_line(x, horizon_y, x, horizon_y + 6, fill="white", tags="static")

    def update_animation(self):
        if not getattr(self, 'anim_running', False): return

        width = self.anim_canvas.winfo_width()
        height = self.anim_canvas.winfo_height()
        if width < 10 or height < 10:
            self.after(500, self.update_animation)
            return

        self.anim_canvas.delete("dyn")

        try:
            # Ambil timezone berdasarkan koordinat
            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))

            # --- PERBAIKAN MESIN WAKTU ---
            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

        # --- 1. KALKULASI TOPOSENTRIK (Posisi Visual Aktual bagi Pengamat) ---
        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()

        # --- 2. KALKULASI GEOSENTRIK (Posisi dari Pusat Bumi untuk KHGT) ---
        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()
        ra_m, dec_m, _ = geo_moon.radec()

        gast = t_sim.gast
        lst_deg = (gast * 15.0) + lon
        lat_rad = math.radians(lat)

        def calc_geo_alt(ra, dec):
            ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
            dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians
            ha_rad = math.radians(lst_deg - (ra_h * 15.0))
            sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) *
math.cos(ha_rad)
            return math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

        alt_sun_geo = calc_geo_alt(ra_s, dec_s)
        alt_moon_geo = calc_geo_alt(ra_m, dec_m)
```

```
        elong_geo = geo_sun.separation_from(geo_moon).degrees

        _, m_lon, _ = astro_moon.ecliptic_latlon()
        _, s_lon, _ = astro_sun.ecliptic_latlon()
        phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
        moon_idx = int(phase_angle) % 360

        waktu_sekarang = ephem.Date(t_sim.utc_datetime())
        ijtimak = ephem.previous_new_moon(waktu_sekarang)
        moon_age = waktu_sekarang - ijtimak

        # ==========================================================================
        # PROYEKSI PROPORSIONAL JARAK & ELONGASI (PERBAIKAN AZIMUTH)
        # ==========================================================================
        horizon_y = height / 2
        px_per_deg_y = horizon_y / 90.0
        px_per_deg_x = width / 360.0  # Skala horizontal mutlak 0-360 derajat

        alt_s_deg = alt_sun.degrees
        az_s_deg = az_sun.degrees
        alt_m_deg = alt_moon.degrees
        az_m_deg = az_moon.degrees

        # 1. Hitung Radius UI Visual
        r_sun = max(36, int(3.6 * px_per_deg_y))
        r_moon = max(22, int(2.2 * px_per_deg_y))

        # 2. TOP LIMB ALIGNMENT (PENYELARASAN PIRINGAN ATAS)
        alt_top_sun = alt_s_deg + 0.833
        alt_top_moon = alt_m_deg + 0.833

        # 3. Posisikan Y (Altitude)
        y_sun = (horizon_y - (alt_top_sun * px_per_deg_y)) + r_sun
        y_moon = (horizon_y - (alt_top_moon * px_per_deg_y)) + r_moon

        # 4. Posisikan X (Azimuth) secara proporsional sesuai kompas latar belakang
        x_sun = az_s_deg * px_per_deg_x
        x_moon = az_m_deg * px_per_deg_x

        if not hasattr(self, '_live_images'):
            self._live_images = {}

        # --- GAMBAR GARIS ELONGASI ---
        # Garis dikembalikan seperti semula agar selalu muncul menghubungkan titik Matahari dan Bulan
        self.anim_canvas.create_line(x_sun, y_sun, x_moon, y_moon, fill="#00E676", dash=(4, 2),
tags="dyn")
```

```
        mid_x = (x_sun + x_moon) / 2
        mid_y = (y_sun + y_moon) / 2

        # Label teks diletakkan di tengah garis
        self.anim_canvas.create_text(mid_x, mid_y - 12, text=f"Eln:{elong_geo:.1f}°", fill="#00E676",
font=("Consolas", 10, "bold"), tags="dyn")

        # --- RENDER MATAHARI ---
        try:
            sun_path = os.path.join(BASE_DIR, "matahari.png")
            sun_img = Image.open(sun_path).convert("RGBA")
            sun_img = sun_img.resize((r_sun*2, r_sun*2), Image.Resampling.LANCZOS)
            self._live_images['sun'] = ImageTk.PhotoImage(sun_img)
            self.anim_canvas.create_image(x_sun, y_sun, image=self._live_images['sun'], tags="dyn")
        except Exception as e:
            self.anim_canvas.create_oval(x_sun-r_sun-4, y_sun-r_sun-4, x_sun+r_sun+4, y_sun+r_sun+4,
fill="#FF6D00", outline="",stipple="gray25", tags="dyn")
            self.anim_canvas.create_oval(x_sun-r_sun, y_sun-r_sun, x_sun+r_sun, y_sun+r_sun,
fill="#FFD54F", outline="#FFF59D", width=2, tags="dyn")

        # [INFO TEKS MATAHARI]
        sun_text = f"Matahari\nAlt(Topo): {alt_s_deg:.2f}°\nAlt(Geo) : {alt_sun_geo:.2f}°\nAzimuth  :
{az_s_deg:.1f}°"
        if x_sun > width - 120:
            self.anim_canvas.create_text(x_sun - r_sun - 15, y_sun, text=sun_text, fill="#FFD54F",
font=("Consolas", 10, "bold"), justify="right", anchor="e", tags="dyn")
        else:
            self.anim_canvas.create_text(x_sun + r_sun + 15, y_sun, text=sun_text, fill="#FFD54F",
font=("Consolas", 10, "bold"), justify="left", anchor="w", tags="dyn")

        # --- RENDER BULAN & FASE MASK (DENGAN ROTASI KEMIRINGAN) ---
        try:
            moon_filename = f"m{moon_idx:03d}.png"
            moon_path = os.path.join(BASE_DIR, "moon", moon_filename)
            moon_img = Image.open(moon_path).convert("RGBA")
            target_size = r_moon * 2
            moon_img = moon_img.resize((target_size, target_size), Image.Resampling.LANCZOS)

            # ------------------------------------------------------------
            # LOGIKA ROTASI MENGHADAP MATAHARI
            # Hitung sudut vektor arah dari Bulan ke Matahari di atas Kanvas
            # Karena sumbu Y di Tkinter terbalik (nilai Y makin besar ke bawah),
            # rotasi dihitung menggunakan math.atan2, dan dirotasi secara berlawanan arah
            dx = x_sun - x_moon
            dy = y_sun - y_moon
```

```
            sudut_kemiringan = math.degrees(math.atan2(dy, dx))

            # Pillow memutar gambar berlawanan arah jarum jam untuk sudut positif.
            # Kita rotasi gambar agar bagian yang tadinya menghadap kanan (0 derajat)
            # menatap lurus sejajar dengan garis elongasi menuju Matahari.
            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")

        # [INFO TEKS BULAN]
        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 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 (DUA ARAH) ---
        # Jika di atas ufuk, tarik garis ke bawah. Jika di bawah ufuk, tarik garis ke atas.
        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")

        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")

        # --- STATUS TEXT LOGIC BERDASARKAN TOP LIMB ---
        is_sun_set_visually = alt_top_sun <= 0
        is_moon_set_visually = alt_top_moon <= 0

        txt_fisik_sun = "🌑 Terbenam Total (Piringan atas di bawah ufuk)" if is_sun_set_visually else "🌞
Di Atas Ufuk"
        txt_fisik_moon = "🌑 Terbenam Total (Piringan atas di bawah ufuk)" if is_moon_set_visually else
"🌙 Di Atas Ufuk"

        box_text = (
            f"REAL-TIME ASTRONOMY DATA\n{waktu_str}\n"
            f"Umur Bulan     : {moon_age:.2f} hari\n"
            f"Elongasi (Geo) : {elong_geo:.2f}°\n"
            f"Status Matahari: {txt_fisik_sun}\n"
            f"Status Bulan   : {txt_fisik_moon}"
        )

        # ======================================================================
        # [KOTAK INFORMASI DATA DIPERBARUI (DIBUAT SEMI-TRANSPARAN)]
        # ======================================================================
        box_width, box_height = 370, 150

        # Buat background gambar RGBA dengan Alpha (150 dari 255) untuk efek kaca transparan
        img_bg = Image.new("RGBA", (box_width, box_height), (24, 24, 24, 160))
        self._live_images['info_bg'] = ImageTk.PhotoImage(img_bg)

        # Tempelkan gambar sebagai background transparan
        self.anim_canvas.create_image(15, 15, image=self._live_images['info_bg'], anchor="nw",
tags="dyn")
        # Gambar garis tepi (outline) tanpa fill (isi)
        self.anim_canvas.create_rectangle(15, 15, 15 + box_width, 15 + box_height, fill="",
outline="#555", width=1.5, tags="dyn")
        # Cetak teks di atasnya
        self.anim_canvas.create_text(25, 25, text=box_text, fill="white", font=("Consolas", 11, "bold"),
anchor="nw", tags="dyn")

        # ======================================================
        # TAMBAHAN INFO KOTAK EVALUASI KHGT DAN NEO MABIMS
        # ======================================================

        elong_topo = astro_sun.separation_from(astro_moon).degrees

        # Hanya tampilkan jika animasi TIDAK dalam mode Live (dipicu oleh tombol Set ke Waktu Sunset)
```

```
        if not getattr(self, 'anim_is_live', True):

            # 1) Evaluasi Kriteria KHGT (Pojok Kiri Bawah)
            if alt_moon_geo >= 5.0 and elong_geo >= 8.0:
                box_w_khgt, box_h_khgt = 380, 40
                y1_khgt = height - 55

                # Buat transparan juga agar konsisten
                img_khgt = Image.new("RGBA", (box_w_khgt, box_h_khgt), (26, 26, 26, 160))
                self._live_images['khgt_bg'] = ImageTk.PhotoImage(img_khgt)

                self.anim_canvas.create_image(15, y1_khgt, image=self._live_images['khgt_bg'],
anchor="nw", tags="dyn")
                self.anim_canvas.create_rectangle(15, y1_khgt, 15 + box_w_khgt, y1_khgt + box_h_khgt,
fill="", outline="#00E676", width=1.5, tags="dyn")
                self.anim_canvas.create_text(25, y1_khgt + 20, text="Memenuhi kriteria KHGT",
fill="#00E676", font=("Consolas", 20, "bold"), anchor="w", tags="dyn")

            # 2) Evaluasi Kriteria Neo MABIMS (Pojok Kanan Bawah)
            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

                # Buat transparan juga agar konsisten
                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)

    # ========================================
    # MODUL TAMBAHAN 13 & 14: 3D SPACE SYSTEM W/ ZOOM
    # ========================================
    def setup_3d_out_frame(self):
        self.frame_3d_out = ctk.CTkFrame(self.main_frame, fg_color="#020205")
```

```
        header = ctk.CTkFrame(self.frame_3d_out, fg_color="#101018", corner_radius=8)
        header.pack(fill="x", padx=15, pady=10)

        ctk.CTkLabel(header, text="🛰 3D GEOCENTRIC VIEW", font=("Montserrat", 16, "bold"),
text_color="#00E5FF").pack(side="left", padx=20)

        self.btn_peak_eclipse = ctk.CTkButton(header, text="🌑 ECLIPSE 2026", width=140,
fg_color="#D32F2F", hover_color="#B71C1C", command=self.set_eclipse_time_3d)
        self.btn_peak_eclipse.pack(side="right", padx=(10, 15), pady=10)

        # --- TOMBOL ZOOM ---

        self.anim_3d_canvas = tk.Canvas(self.frame_3d_out, bg='#010105', highlightthickness=0)
        self.anim_3d_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))

        # Inisialisasi Kamera & Skala
        self.cam_angle_x = 0.8
        self.cam_angle_y = 0.3
        self.cam_zoom = 1.0

        # Binding Interaksi Mouse (Rotasi & Scroll Zoom Cross-Platform)
        self.anim_3d_canvas.bind("<B1-Motion>", self.rotate_3d_view)
        self.anim_3d_canvas.bind("<MouseWheel>", self.on_mouse_wheel_3d) # Windows/Mac
        self.anim_3d_canvas.bind("<Button-4>", self.on_mouse_wheel_3d)   # Linux Scroll Up
        self.anim_3d_canvas.bind("<Button-5>", self.on_mouse_wheel_3d)   # Linux Scroll Down

    def zoom_in(self):
        if hasattr(self, 'cam_zoom'):
            self.cam_zoom *= 1.2
            if self.cam_zoom > 12.0: self.cam_zoom = 12.0 # Batas maksimal perbesaran

    def zoom_out(self):
        if hasattr(self, 'cam_zoom'):
            self.cam_zoom *= 0.8
            if self.cam_zoom < 0.1: self.cam_zoom = 0.1 # Batas maksimal pengecilan

    def on_mouse_wheel_3d(self, event):
        # Logika scroll menangkap pergerakan roda mouse dengan aman
        if str(event.type) == 'MouseWheel':
            if event.delta > 0: self.zoom_in()
            else: self.zoom_out()
        else:
            if getattr(event, 'num', 0) == 4: self.zoom_in()
            elif getattr(event, 'num', 0) == 5: self.zoom_out()
```

```
def set_eclipse_time_3d(self):
    # Update dropdown Menu 16
    self.combo_3d_y.set("2026")
    self.combo_3d_m.set("03")
    self.combo_3d_d.set("03")

    # Update juga entry general (untuk sinkronisasi antar menu)
    self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, "2026")
    self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, "03")
    self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, "03")

    messagebox.showinfo("Eclipse Mode", "Simulasi 3D disetel ke Puncak Gerhana 3 Maret 2026.")
    # Perbaikan: Panggil update_3d_animation agar visual langsung berubah
    self.update_3d_animation()


def get_shared_date_components(self):
    try:
        # Jika Menu 16 aktif, ambil dari dropdown khusus 3D
        if "Sistem Sun Moon" in self.combo_mode.get():
            return int(self.combo_3d_y.get()), int(self.combo_3d_m.get()), int(self.combo_3d_d.get())
        # Fallback ke input general Menu 1
        return int(self.entry_vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
    except:
        now = datetime.datetime.now()
        return now.year, now.month, now.day

def 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_raw(self, x, y, z):
    # Murni melakukan rotasi sumbu 3D dan kalkulasi jarak kamera tanpa terpengaruh Zoom
    x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
    z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
    y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
    z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)

    camera_dist = 1000.0
    factor = camera_dist / (camera_dist + z2) if (camera_dist + z2) != 0 else 1

    return x1 * factor, -y2 * factor, z2
```

```
def update_3d_animation(self):
    if not self.is_viewing_3d: return

    canvas = self.anim_3d_canvas
    w = canvas.winfo_width()
    h = canvas.winfo_height()

    if w <= 1:
        self.after(200, self.update_3d_animation)
        return

    canvas.delete("all")
    cx, cy = w / 2, h / 2
    zoom = getattr(self, 'cam_zoom', 1.0)

    try:
        y, m, d = self.get_shared_date_components()

        # Pastikan ephemeris yang tepat dimuat (Switch de406 untuk tahun negatif)
        self.auto_switch_ephemeris(y)

        # Jam diset 12:00 agar posisi planet stabil saat ganti tanggal
        t_sim = self.ts.utc(y, m, d, 12, 0, 0)

        earth_obj = self.eph['earth']
        celestial_bodies = [
            ('moon', "#ECEFF1", 12, 140, "BULAN"),
            ('mercury', "#B0BEC5", 8, 200, "MERKURIUS"),
            ('venus', "#FFCC80", 14, 260, "VENUS"),
            ('sun', "#FFD600", 35, 330, "MATAHARI"),
            ('mars', "#EF5350", 12, 420, "MARS"),
            ('jupiter', "#FFB74D", 26, 560, "JUPITER"),
            ('saturn', "#FFE082", 22, 700, "SATURNUS")
        ]

        draw_list = []
        # Proyeksi Bumi
        raw_ex, raw_ey, z_earth = self.project_3d_raw(0, 0, 0)
        p_earth = (raw_ex * zoom + cx, raw_ey * zoom + cy, z_earth)
        draw_list.append(('earth', p_earth, "#2196F3", 20 * zoom, "BUMI"))

        for name, color, base_size, vis_dist, label in celestial_bodies:
            try:
                target_key = f'{name} barycenter' if name not in ['sun', 'moon'] else name
                target_obj = self.eph[target_key] if target_key in self.eph else self.eph[name]
```

```
                pos_km = earth_obj.at(t_sim).observe(target_obj).position.km
                dist_km = np.linalg.norm(pos_km)
                px, py, pz = (pos_km / dist_km) * vis_dist
                rx, ry, rz = self.project_3d_raw(px, py, pz)
                final_p = (rx * zoom + cx, ry * zoom + cy, rz)
                draw_list.append((name, final_p, color, base_size * zoom, label))
            except: continue

        draw_list.sort(key=lambda item: item[1][2], reverse=True)
        for tag, p, color, size, label in draw_list:
            if size < 0.5: continue
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill=color, outline="white" if
tag=='earth' else "")
            if label and size > 6:
                canvas.create_text(p[0], p[1]+size+10, text=label, fill="white", font=("Consolas", 9, "bold"))

        # Label status di pojok layar
        tahun_display = f"{y}" if y > 0 else f"{abs(y-1)} SM"
        canvas.create_text(35, 35, text=f"3D SYSTEM - DATE: {d}/{m}/{tahun_display}\nEph:
{self.ephemeris_name}",
                           fill="#00E5FF", font=("Consolas", 10), anchor="nw")

    except Exception as e:
        canvas.create_text(cx, cy, text=f"Updating Data... {e}", fill="gray")

    # After diperlambat ke 100ms karena data tanggal bersifat statis
    self.after(100, self.update_3d_animation)

def adjust_zoom(self, factor):
    self.cam_zoom *= factor
    # Membatasi skala zoom agar tidak terlalu jauh atau tembus ke dalam
    if self.cam_zoom < 0.1: self.cam_zoom = 0.1
    if self.cam_zoom > 8.0: self.cam_zoom = 8.0

def on_mouse_wheel_3d(self, event):
    # Logika menangkap pergerakan roda mouse
    if event.num == 4 or getattr(event, 'delta', 0) > 0:
        self.adjust_zoom(1.1)
    elif event.num == 5 or getattr(event, 'delta', 0) < 0:
        self.adjust_zoom(0.9)

def set_eclipse_time_3d(self):
    # Update dropdown Menu 16
    self.combo_3d_y.set("2026")
    self.combo_3d_m.set("03")
    self.combo_3d_d.set("03")
```

```
    # Update juga entry general (untuk sinkronisasi antar menu)
    self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, "2026")
    self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, "03")
    self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, "03")

    messagebox.showinfo("Eclipse Mode", "Simulasi 3D disetel ke Puncak Gerhana 3 Maret 2026.")

def get_shared_date_components(self):
    try: return int(self.entry_vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
    except:
        now = datetime.datetime.now()
        return now.year, now.month, now.day

def rotate_3d_view(self, event):
    w = self.anim_3d_canvas.winfo_width()
    h = self.anim_3d_canvas.winfo_height()
    if w > 0 and h > 0:
        self.cam_angle_x = (event.x / w) * 2 * math.pi
        self.cam_angle_y = (event.y / h) * math.pi

def project_3d(self, x, y, z, width, height):
    x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
    z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
    y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
    z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)

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

def update_3d_animation(self):
    if not self.is_viewing_3d: return

    canvas = self.anim_3d_canvas
    w = canvas.winfo_width()
    h = canvas.winfo_height()

    if w <= 1:
        self.after(200, self.update_3d_animation)
        return

    canvas.delete("all")

    # Gambar background bintang (statis)
```

```
random.seed(42)
for _ in range(100):
    rx, ry = random.randint(0, w), random.randint(0, h)
    canvas.create_oval(rx, ry, rx+1, ry+1, fill="#555555")

try:
    y, m, d = self.get_shared_date_components()
    now = datetime.datetime.now()
    t_sim = self.ts.utc(y, m, d, now.hour, now.minute, now.second)

    earth_obj = self.eph['earth']

    # Daftar benda langit yang divisualisasikan
    celestial_bodies = [
        ('moon', "#ECEFF1", 15, 180, "BULAN"),
        ('mercury', "#B0BEC5", 10, 230, "MERKURIUS"),
        ('venus', "#FFCC80", 18, 290, "VENUS"),
        ('sun', "#FFD600", 50, 350, "MATAHARI"),
        ('mars', "#EF5350", 16, 450, "MARS"),
        ('jupiter', "#FFB74D", 35, 600, "JUPITER"),
        ('saturn', "#FFE082", 30, 750, "SATURNUS"),
        ('uranus', "#81D4FA", 25, 900, "URANUS"),
        ('neptune', "#5C6BC0", 24, 1050, "NEPTUNUS"),
        ('pluto', "#CFD8DC", 8, 1200, "PLUTO")
    ]

    draw_list = []

    # 1. Menggambar Bumi di Pusat (Geocentric)
    p_earth = self.project_3d(0, 0, 0, w, h)
    draw_list.append(('earth', p_earth, "#2196F3", 30 * self.cam_zoom, "BUMI"))

    # 2. Menggambar Umbra (Bayangan Gelap Bumi)
    try:
        pos_sun = earth_obj.at(t_sim).observe(self.eph['sun']).position.km
        dist_s = np.linalg.norm(pos_sun)
        # Jarak visual umbra diskalakan dengan zoom
        ux, uy, uz = -(pos_sun / dist_s) * (180 * self.cam_zoom)
        p_umbra = self.project_3d(ux, uy, uz, w, h)
        draw_list.append(('umbra', p_umbra, "#1A1A1A", 22 * self.cam_zoom, ""))
    except:
        pass

    # 3. Kalkulasi dan Skala Zoom untuk Semua Objek
    for name, color, size, vis_dist, label in celestial_bodies:
        try:
```

```
            target_key = f'{name} barycenter' if name not in ['sun', 'moon'] else name
            try:
               target_obj = self.eph[target_key]
            except KeyError:
               target_obj = self.eph[name]

            pos_km = earth_obj.at(t_sim).observe(target_obj).position.km
            dist_km = np.linalg.norm(pos_km)

            # Kalikan jarak visual dengan faktor zoom kamera saat ini
            px, py, pz = (pos_km / dist_km) * (vis_dist * self.cam_zoom)

            proj_p = self.project_3d(px, py, pz, w, h)
            # Kalikan juga ukuran objek (size) dengan zoom
            draw_list.append((name, proj_p, color, size * self.cam_zoom, label))

         except Exception as e:
            pass

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

      # 4. Merender objek ke Kanvas
      for tag, p, color, size, label in draw_list:
         if tag == 'umbra':
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill="", outline="#333",
dash=(4,4))
         else:
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill=color, outline="white" if
tag=='earth' else "")
            # Tampilkan Label Nama Planet (Sembunyikan jika di zoom out terlalu jauh agar tidak
numpuk)
            if label and size > 4:
               font_size = max(7, int(9 * min(self.cam_zoom, 1.5)))
               canvas.create_text(p[0], p[1]+size+12, text=label, fill="white", font=("Consolas", font_size,
"bold"))

      # Menggambar Garis Orbit Panduan (Ikut ter-zoom)
      r_moon = 180 * self.cam_zoom
      r_sun = 350 * self.cam_zoom
      r_out = 600 * self.cam_zoom
      canvas.create_oval(w/2-r_moon, h/2-r_moon, w/2+r_moon, h/2+r_moon, outline="#222")
      canvas.create_oval(w/2-r_sun, h/2-r_sun, w/2+r_sun, h/2+r_sun, outline="#222")
      canvas.create_oval(w/2-r_out, h/2-r_out, w/2+r_out, h/2+r_out, outline="#111")

      info = f"3D GEOCENTRIC SOLAR SYSTEM (LIVE)\n" \
```

```
            f"Date: {y}-{m:02d}-{d:02d}\n" \
            f"Zoom Level: {self.cam_zoom:.1f}x\n" \
            f"* Klik Kiri & Geser Mouse untuk Rotasi\n" \
            f"* Scroll Mouse / Tombol UI untuk Zoom"

        canvas.create_rectangle(20, 20, 360, 110, fill="#050510", outline="#00E5FF", width=1)
        canvas.create_text(35, 35, text=info, fill="#00E5FF", font=("Consolas", 10), anchor="nw")

    except Exception as e:
        canvas.create_text(w/2, h/2, text=f"System Updating Data... {e}", fill="gray")

    self.after(50, self.update_3d_animation)

def set_eclipse_time_3d(self):
    self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, "2026")
    self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, "03")
    self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, "03")
    messagebox.showinfo("Eclipse Mode", "Waktu diset ke 3 Maret 2026.")

def get_shared_date_components(self):
    try: return int(self.entry_vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
    except:
        now = datetime.datetime.now()
        return now.year, now.month, now.day

def rotate_3d_view(self, event):
    w = self.anim_3d_canvas.winfo_width()
    h = self.anim_3d_canvas.winfo_height()
    if w > 0 and h > 0:
        self.cam_angle_x = (event.x / w) * 2 * math.pi
        self.cam_angle_y = (event.y / h) * math.pi

def project_3d(self, x, y, z, width, height):
    x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
    z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
    y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
    z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)

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

def update_3d_animation(self):
    if not self.is_viewing_3d: return
```

```
canvas = self.anim_3d_canvas
w = canvas.winfo_width()
h = canvas.winfo_height()

if w <= 1:
    self.after(200, self.update_3d_animation)
    return

canvas.delete("all")

# Gambar bintang background
random.seed(42)
for _ in range(100):
    rx, ry = random.randint(0, w), random.randint(0, h)
    canvas.create_oval(rx, ry, rx+1, ry+1, fill="#555555")

try:
    y, m, d = self.get_shared_date_components()
    now = datetime.datetime.now()
    t_sim = self.ts.utc(y, m, d, now.hour, now.minute, now.second)

    earth_obj = self.eph['earth']

    # Daftar benda langit: (Kata Kunci Skyfield, Warna, Ukuran Pixel, Jarak Visual Kanvas, Label)
    # Skala jarak dibuat artifisial agar muat di layar
    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. Masukkan Bumi (Pusat / Geocentric)
    p_earth = self.project_3d(0, 0, 0, w, h)
    draw_list.append(('earth', p_earth, "#2196F3", 30, "BUMI"))

    # 2. Kalkulasi Umbra (Bayangan Bumi)
    try:
```

```
            pos_sun = earth_obj.at(t_sim).observe(self.eph['sun']).position.km
            dist_s = np.linalg.norm(pos_sun)
            ux, uy, uz = -(pos_sun / dist_s) * 180
            p_umbra = self.project_3d(ux, uy, uz, w, h)
            draw_list.append(('umbra', p_umbra, "#1A1A1A", 22, ""))
        except:
            pass

        # 3. Kalkulasi Semua Planet & Benda Langit
        for name, color, size, vis_dist, label in celestial_bodies:
            try:
                # Coba ambil data dari file ephemeris (.bsp)
                # File JPL biasanya menggunakan tambahan 'barycenter' untuk planet
                target_key = f'{name} barycenter' if name not in ['sun', 'moon'] else name

                try:
                    target_obj = self.eph[target_key]
                except KeyError:
                    target_obj = self.eph[name] # Fallback jika barycenter tidak ada

                pos_km = earth_obj.at(t_sim).observe(target_obj).position.km
                dist_km = np.linalg.norm(pos_km)

                # Normalisasi vektor arah, lalu kalikan dengan jarak visual kanvas
                px, py, pz = (pos_km / dist_km) * vis_dist

                # Proyeksi ke 2D Layar
                proj_p = self.project_3d(px, py, pz, w, h)
                draw_list.append((name, proj_p, color, size, label))

            except Exception as e:
                # Abaikan jika ada planet yang tidak didukung oleh file .bsp saat ini
                pass

        # Urutkan berdasarkan kedalaman Z (Z-Buffer sederhana) agar objek di depan menutupi yang di
belakang
        draw_list.sort(key=lambda x: x[1][2], reverse=True)

        # 4. Gambar Objek ke Canvas
        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:
                # Gambar Lingkaran Planet
```

```
            canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill=color, outline="white" if
tag=='earth' else "")
            # Gambar Teks Nama Planet
            if label:
               canvas.create_text(p[0], p[1]+size+12, text=label, fill="white", font=("Consolas", 9, "bold"))

        # Garis orbit referensi artifisial (hanya pemanis visual)
        canvas.create_oval(w/2-180, h/2-180, w/2+180, h/2+180, outline="#222") # Orbit Bulan
        canvas.create_oval(w/2-350, h/2-350, w/2+350, h/2+350, outline="#222") # Orbit Matahari
        canvas.create_oval(w/2-600, h/2-600, w/2+600, h/2+600, outline="#111") # Orbit Luar

        info = f"3D GEOCENTRIC SOLAR SYSTEM (LIVE)\n" \
             f"Date: {y}-{m:02d}-{d:02d}\n" \
             f"Displaying: Earth, Moon, Sun, & Planets\n" \
             f"* Jarak diskala untuk visualisasi UI"

        canvas.create_rectangle(20, 20, 360, 110, fill="#050510", outline="#00E5FF", width=1)
        canvas.create_text(35, 35, text=info, fill="#00E5FF", font=("Consolas", 10), anchor="nw")

      except Exception as e:
        canvas.create_text(w/2, h/2, text=f"System Updating Data... {e}", fill="gray")

      self.after(50, self.update_3d_animation)

  # ========================================
  # FUNGSI KONVERSI KHGT
  # ========================================
  def get_new_moons_in_range(self, start_tt_float, end_tt_float):
    t0 = self.ts.tt_jd(start_tt_float)
    t1 = self.ts.tt_jd(end_tt_float)
    t_obj, y_obj = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
    if t_obj is None: return []
    if getattr(t_obj, 'shape', ()) == ():
      tt_list = [float(t_obj.tt)]
      y_list = [int(y_obj)]
    else:
      tt_list = t_obj.tt.tolist()
      y_list = y_obj.tolist()
    new_moons_tt = []
    for i in range(len(y_list)):
      if y_list[i] == 0:
        new_moons_tt.append(tt_list[i])
    return new_moons_tt

  def get_hijri_month_from_tt(self, tt_float):
    delta_months = round((tt_float - 2451550.0) / 29.530588853)
```

```
    abs_month = 17038 + delta_months
    y = (abs_month - 1) // 12 + 1
    m = (abs_month - 1) % 12 + 1
    return int(y), int(m)

def get_approx_nm_tt(self, y, m):
    abs_month = (y - 1) * 12 + m
    delta_months = abs_month - 17038
    return 2451550.0 + delta_months * 29.530588853

def calculate_khgt_1st_of_month(self, nm_tt_float):
    # Menggunakan Julian Date (Float TT) agar kebal dari error tahun minus Python
    t_nm = self.ts.tt_jd(nm_tt_float)
    y, m, d, h, mi, s = t_nm.utc

    # Cari titik waktu 00:00 UTC pada hari konjungsi
    t_midnight = self.ts.utc(y, m, d)

    # Aturan KHGT: Jika ijtimak sebelum jam 15:00 UTC (Fajar Selandia Baru),
    # bulan baru masuk besok (+1 hari). Jika tidak, lusa (+2 hari).
    if h < 15:
        return t_midnight.tt + 1.0
    else:
        return t_midnight.tt + 2.0

def calculate_conversion(self):
    try:
        mode_conv = self.radio_conv_var.get()
        d = int(self.combo_conv_day.get())
        m_str = self.combo_conv_month.get()
        y = int(self.entry_conv_year.get())
        adj = int(self.combo_conv_adj.get())

        # Fungsi bantuan deteksi kabisat yang aman untuk tahun SM (Negatif)
        def is_leap(year):
            return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

        if mode_conv == "m2h":
            m = BULAN_MASEHI.index(m_str) + 1
            mode_text = "Masehi ➔ Hijriah"

            # Format string Masehi (Bypass library date Python)
            greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
            input_text = f"{d} {m_str} {greg_year_str_input}"

            self.auto_switch_ephemeris(y)
```

```
# Validasi jumlah hari dalam bulan agar tidak out-of-bounds
days_in_month = [31, 29 if is_leap(y) else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
max_days = days_in_month[m-1]
d = min(d, max_days)

# Eksekusi menggunakan Skyfield Time murni
target_t_obj = self.ts.utc(y, m, d)
target_tt = target_t_obj.tt

new_moons_tt_list = self.get_new_moons_in_range(target_tt - 35.0, target_tt + 2.0)
current_nm_tt = None
for nm_tt in reversed(new_moons_tt_list):
    start_tt = self.calculate_khgt_1st_of_month(nm_tt)
    if target_tt >= start_tt - 0.1: # Toleransi 0.1 hari
        current_nm_tt = nm_tt
        break

if current_nm_tt is None:
    raise ValueError(f"Tanggal target di luar jangkauan Ephemeris ({self.ephemeris_name}).")

h_y, h_m = self.get_hijri_month_from_tt(current_nm_tt)
start_tt = self.calculate_khgt_1st_of_month(current_nm_tt)

# Hitung selisih hari dari 1 Hijriah KHGT
h_d = int(round(target_tt - start_tt)) + 1
h_d += adj

while h_d <= 0:
    h_m -= 1
    if h_m <= 0:
        h_m, h_y = 12, h_y - 1
    h_d += 30
while h_d > 30:
    h_d -= 30
    h_m += 1
    if h_m > 12:
        h_m, h_y = 1, h_y + 1

# Mengambil hari (Senin-Minggu) dari parameter Julian Date
hari_idx = int(target_t_obj.whole % 7)
hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]

hijri_year_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
hasil = f"{hari}, {h_d} {BULAN_HIJRIAH[h_m - 1]} {hijri_year_str} H"
```

```
        else:
            m = BULAN_HIJRIAH.index(m_str) + 1
            h_y = y
            mode_text = "Hijriah ➔ Masehi"
            hijri_year_str_input = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
            input_text = f"{d} {m_str} {hijri_year_str_input} H"

            approx_greg_year = int(h_y * 0.970224 + 622.54)
            self.auto_switch_ephemeris(approx_greg_year)

            approx_tt = self.get_approx_nm_tt(h_y, m)
            new_moons_tt_list = self.get_new_moons_in_range(approx_tt - 5.0, approx_tt + 5.0)
            if not new_moons_tt_list:
                raise ValueError(f"Tanggal tersebut di luar jangkauan Ephemeris ({self.ephemeris_name}).")

            exact_nm_tt = new_moons_tt_list[0]
            start_tt = self.calculate_khgt_1st_of_month(exact_nm_tt)

            # Tambahkan hari ke TT, lalu kembalikan ke UTC (Tuple: y, m, d)
            target_tt = start_tt + (d - 1) - adj
            target_t_obj = self.ts.tt_jd(target_tt)
            ty, tm, td, _, _, _ = target_t_obj.utc

            hari_idx = int(target_t_obj.whole % 7)
            hari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]

            greg_year_str = f"{ty}" if ty > 0 else f"{abs(ty-1)} SM"
            hasil = f"{hari}, {td} {BULAN_MASEHI[tm - 1]} {greg_year_str}"

        report = f"""{self.get_header(85)}
{"[ Konversi Kalender KHGT ]".center(85)}

* Settings:-
- Mode Konversi : {mode_text}
- Tanggal Input : {input_text}
- Koreksi Hari  : {adj} Hari (Penyesuaian Manual)
- Ephemeris     : {self.ephemeris_name}
================================================================================

HASIL KONVERSI:
--------------------------------------------------------------------------------
>>  {hasil}  <<
--------------------------------------------------------------------------------

* Catatan:
- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).
```

```
- Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.
- Kalkulasi menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak).
================================================================================
"""
            self.after(0, self.display_result, report)

        except Exception as e:
            import traceback
            trace_err = traceback.format_exc()
            self.after(0, self.display_error, f"{str(e)}\n\n(Details):\n{trace_err}")

    def generate_visibility_report(self):
        """Fungsi inti kalkulasi hilal yang me-return string report murni (Bisa dipanggil oleh WinAI)"""
        year = int(self.entry_vyear.get())
        month = int(self.entry_vmonth.get())
        day = int(self.entry_vday.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        elev = float(self.entry_velev.get())
        tz = float(self.entry_vtz.get())

        temp = float(self.entry_vtemp.get())
        pres = float(self.entry_vpres.get())
        hum = float(self.entry_vhum.get())

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

        t0 = self.ts.utc(year, month, day, 4 - int(tz))
        t1 = self.ts.utc(year, month, day, 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("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_latlon(epoch=t)
      _, mlon, _ = e.observe(moon).apparent().ecliptic_latlon(epoch=t)
      s_deg = slon.degrees.item() if hasattr(slon.degrees, 'item') else slon.degrees
      m_deg = mlon.degrees.item() if hasattr(mlon.degrees, 'item') else mlon.degrees
      return (m_deg - s_deg) % 360.0

    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, dist_moon = app_moon_geo.radec()
    ra_sun, dec_sun, dist_sun = app_sun_geo.radec()

    m_lat, m_lon, _ = app_moon_geo.ecliptic_latlon(epoch=t_sunset)
    s_lat, s_lon, _ = app_sun_geo.ecliptic_latlon(epoch=t_sunset)

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

    def get_geo_altaz(ra, dec):
      ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
```

```
        dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians
        ha_deg = lst_deg - (ra_h * 15.0)
        lat_rad, dec_rad, ha_rad = math.radians(lat), dec_r, math.radians(ha_deg)
        sin_alt = math.sin(dec_rad) * math.sin(lat_rad) + math.cos(dec_rad) * math.cos(lat_rad) *
math.cos(ha_rad)
        sin_alt = max(-1.0, min(1.0, sin_alt))
        alt = math.degrees(math.asin(sin_alt))
        y = -math.sin(ha_rad)
        x = math.tan(dec_rad) * math.cos(lat_rad) - math.sin(lat_rad) * math.cos(ha_rad)
        az = (math.degrees(math.atan2(y, x)) + 360) % 360
        return alt, az

    alt_moon_geo, az_moon_geo = get_geo_altaz(ra_moon, dec_moon)
    alt_sun_geo, az_sun_geo = get_geo_altaz(ra_sun, dec_sun)

    topo_earth = (earth + lokasi_obs).at(t_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 - ts_tt) * 24.0
    else:
        lag_time_hours = 0.0

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

    rel_alt_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 = self.ts.ut1_jd(t_ijtima.ut1 + (tz / 24.0))
        y_i, m_i, d_i, h_i, mn_i, s_i = t_ijtima_local.utc
        y_str = f"{y_i} CE" if y_i > 0 else f"{abs(y_i-1)} BCE"
        str_ijtima = f"{d_i:02d}/{m_i:02d}/{y_str}, {int(h_i):02d}.{int(mn_i):02d}"
    else:
        str_ijtima = "N/A"

    khgt_alt, khgt_elong = 5.0, 8.0
    is_khgt_visible = alt_moon_geo >= khgt_alt and elongation >= khgt_elong

    if is_khgt_visible:
        khgt_status_str = "Fulfilled"
        khgt_note = "KHGT criteria are fulfilled. Crescent position has reached the minimum limits."
    else:
        khgt_status_str = "Not Fulfilled"
        if lag_time_hours < 0:
            khgt_note = "Impossible to meet the criteria because the Moon sets before the Sun."
        else:
            khgt_note = "Crescent altitude or elongation has not reached the minimum limits (Alt >= 5° &
Elong >= 8°)."

    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}"
```

```
        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.
- CUSTOM LOCATION, Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz}
================================================================================

- G. Conjunction Time: {str_ijtima} LT
- Julian Date at Time of Calculations: {ts_tt:.5f}

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

- G. Moon Right Ascension: {format_angle(ra_moon.degrees, is_ra=True):<16} G. Moon Declination    :
{format_angle(dec_moon.degrees):<16}
- G. Sun Right Ascension : {format_angle(ra_sun.degrees, is_ra=True):<16} G. Sun Declination     :
{format_angle(dec_sun.degrees):<16}

- G. Moon Altitude      : {format_angle(alt_moon_geo):<16} G. Moon Azimuth       :
{format_angle(az_moon_geo):<16}
- T. Moon Altitude      : {format_angle(alt_moon_topo):<16} T. Moon Azimuth       :
{format_angle(az_moon_topo):<16}

- G. Sun Altitude       : {format_angle(alt_sun_geo):<16} G. Sun Azimuth        :
{format_angle(az_sun_geo):<16}
- T. Sun Altitude       : {format_angle(alt_sun_topo):<16} T. Sun Azimuth        :
{format_angle(az_sun_topo):<16}

- G. Elongation         : {format_angle(elongation):<16} G. Phase Angle        :
{format_angle(phase_angle):<16}
- G. Illumination       : {f"{illumination:05.2f} %":<16} G. Distance          : {f"{dist_km:.2f} Km":<16}

- According to KHGT Criteria, using the following values at Sunset Time:
  * Geocentric Altitude = {format_angle(alt_moon_geo)} ({alt_moon_geo:.2f}°)
  * Geocentric Elongation = {format_angle(elongation)} ({elongation:.2f}°)

  * Global Calendar Parameter: {khgt_status_str}
- Note: {khgt_note}
================================================================================
"""
        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, almanac.moon_phases(self.eph))

        tz_offset = 7.0
        rows = []
        current_row = ["", "", "", ""]

        # Pastikan t_phases tidak kosong
        if t_phases is not None:
            for t_obj, phase_idx in get_safe_events(t_phases, y_phases):
                t_lokal = self.ts.tt_jd(t_obj.tt + (tz_offset / 24.0))
                y_l, m_l, d_l, h_l, min_l, s_l = t_lokal.utc

                tahun_str = f"{int(y_l)} M" if y_l > 0 else f"{abs(int(y_l)-1)} SM"
                date_str = f"{int(d_l):02d}/{int(m_l):02d}/{tahun_str}  {int(h_l):02d}.{int(min_l):02d}"

                if current_row[phase_idx] != "":
                    rows.append(current_row)
                    current_row = ["", "", "", ""]
```

```
                current_row[phase_idx] = date_str

            if any(current_row):
                rows.append(current_row)

            tahun_header = f"{tahun} M" if tahun > 0 else f"{abs(tahun-1)} SM"
            info_batas = " (Mulai 30 Jan)" if tahun == -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.ephemeris_name}).
"""
            output_lines.append(footer)
            self.after(0, self.display_result, "\n".join(output_lines))

        except Exception as e:
            import traceback
            self.after(0, self.display_error, f"Kesalahan Rentang Data: {str(e)}\n\n(Catatan: File ephemeris
Anda hanya mendukung hingga 29 Jan -3000 SM)")

    def calculate_ephemeris(self):
        try:
```

```
        obj_target = self.combo_eph_obj.get()
        time_ref = self.combo_eph_tref.get()
        loc_ref = self.combo_eph_lref.get()

        # 1. Ambil Input (Aman untuk tahun minus)
        sy, sm, sd = int(self.entry_eph_sy.get()), int(self.entry_eph_smon.get()),
int(self.entry_eph_sd.get())
        sh, smin, ssec = int(self.entry_eph_sh.get()), int(self.entry_eph_smin.get()),
int(self.entry_eph_ssec.get())

        ey, em, ed = int(self.entry_eph_ey.get()), int(self.entry_eph_emon.get()),
int(self.entry_eph_ed.get())
        eh, emin, esec = int(self.entry_eph_eh.get()), int(self.entry_eph_emin.get()),
int(self.entry_eph_esec.get())

        step_val = float(self.entry_eph_step.get())
        step_unit = self.combo_eph_step.get()

        lat = float(self.entry_eph_lat.get())
        lon = float(self.entry_eph_lon.get())
        elev = float(self.entry_eph_elev.get())
        tz = float(self.entry_eph_tz.get())

        # Auto-Switch Ephemeris berdasarkan tahun awal
        self.auto_switch_ephemeris(sy)
        earth = self.eph['earth']

        if obj_target == "Sun":
           target = self.eph['sun']
           radius_km = 696000.0
        else:
           target = self.eph['moon']
           radius_km = 1737.4

        loc = wgs84.latlon(lat, lon, elevation_m=elev)

        # 2. KONVERSI KE JULIAN DATE (Bebas dari limitasi datetime Python)
        # Kita hitung waktu mulai dan akhir dalam format TT (Terrestrial Time)
        t_start = self.ts.utc(sy, sm, sd, sh, smin, ssec)
        t_end = self.ts.utc(ey, em, ed, eh, emin, esec)

        # Pengaman batas bawah ephemeris de406 (-3000-01-29)
        jd_limit = self.ts.utc(-3000, 1, 29, 12).tt
        current_jd = max(t_start.tt, jd_limit)
        end_jd = t_end.tt
```

```
        # Tentukan besaran kenaikan (Step) dalam satuan hari (Julian Date)
        if step_unit == "Day": step_jd = step_val
        elif step_unit == "Hour": step_jd = step_val / 24.0
        elif step_unit == "Minute": step_jd = step_val / 1440.0
        else: step_jd = step_val / 86400.0

        output_lines = []
        lat_str = format_angle(lat).replace("+", "").replace("-", "-")
        lon_str = format_angle(lon).replace("+", "")

        header = f"""{self.get_header(128)}
{"[ Sun & Moon Ephemeris ]".center(128)}

* Settings:-
- {obj_target} Ephemeris
- LOKASI: Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f}
- 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

            # Buat objek waktu Skyfield dari Julian Date aktif
            t = self.ts.tt_jd(current_jd)

            # Kalkulasi Data Astronomi
            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()
                # Hitung Alt-Az Geosentrik Manual
                gast = t.gast
                lst_deg = (gast * 15.0) + lon
                ha_rad = math.radians(lst_deg - (ra.hours * 15.0))
                dec_r = dec.radians
                lat_r = math.radians(lat)
                sin_alt = math.sin(dec_r)*math.sin(lat_r) +
math.cos(dec_r)*math.cos(lat_r)*math.cos(ha_rad)
                alt_deg = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
                az_deg = (math.degrees(math.atan2(-math.sin(ha_rad), math.tan(dec_r)*math.cos(lat_r)-
math.sin(lat_r)*math.cos(ha_rad))) + 360) % 360

            lat_ecl, lon_ecl, _ = apparent.ecliptic_latlon(epoch=t)
            sd_deg = math.degrees(math.asin(radius_km / distance.km))

            # --- FORMAT WAKTU LOKAL (ANTI-CRASH) ---
            # Untuk tampilan, kita geser JD ke waktu lokal pengguna
            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))}"

            # --- FORMAT ANGKA ---
            ra_str = format_eph_angle(ra.hours * 15.0, is_ra=True)
            dec_str = format_eph_angle(dec.degrees)
            alt_str = format_eph_angle(alt_deg)
            az_str = format_eph_angle(az_deg)
            lat_e_str = format_eph_angle(lat_ecl.degrees)
            lon_e_str = format_eph_angle(lon_ecl.degrees)
            dist_str = f"{distance.km:.1f}".replace(".", ",")
            sd_str = format_eph_angle(sd_deg, is_sd=True)
            hp_str = format_eph_angle(hp_deg, is_sd=True)
            dt_str = f"{t.delta_t:.1f}".replace(".", ",")

            line = f"{time_str:<21} {ra_str}  {dec_str:>9}  {alt_str:>9}  {az_str:>9}  {lat_e_str:>9}
{lon_e_str:>9}  {dist_str:>11}  {sd_str}  {hp_str}  {dt_str:>4}"
            output_lines.append(line)

            if loop_count % 4 == 0: output_lines.append("")

            # Naikkan JD berdasarkan interval yang dipilih
            current_jd += step_jd
```

```
            output_lines.append("=" * 128 + "\n* Calculated using JPL Ephemeris NASA.")
            self.after(0, self.display_result, "\n".join(output_lines))

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

    def calculate_qiblah(self):
        try:
            year = int(self.entry_qyear.get())
            month = int(self.entry_qmonth.get())
            day = int(self.entry_qday.get())
            target_date = datetime.date(year, month, day)

            self.auto_switch_ephemeris(year)

            lat = float(self.entry_qlat.get())
            lon = float(self.entry_qlon.get())
            tz = float(self.entry_qtz.get())
            mode_qiblah = self.radio_qiblah_var.get()

            phi_k = math.radians(21.4225)
            lam_k = math.radians(39.8262)
            phi = math.radians(lat)
            lam = math.radians(lon)

            y_val = math.sin(lam_k - lam)
            x_val = math.cos(phi) * math.tan(phi_k) - math.sin(phi) * math.cos(lam_k - lam)
            qiblah_angle = math.degrees(math.atan2(y_val, x_val))
            qiblah_angle = (qiblah_angle + 360.0) % 360.0

            target_azimuth = qiblah_angle
            if mode_qiblah == "shadow":
                target_azimuth = (qiblah_angle + 180.0) % 360.0

            earth = self.eph['earth']
            sun = self.eph['sun']
            loc = wgs84.latlon(lat, lon, elevation_m=0.0)

            t0 = self.ts.utc(target_date.year, target_date.month, target_date.day, -int(tz))
            t1 = self.ts.utc(target_date.year, target_date.month, target_date.day, 24 - int(tz))

            tt_array = np.linspace(t0.tt, t1.tt, 1440)
            t_search = self.ts.tt_jd(tt_array)
```

```
        astrometric = (earth + loc).at(t_search).observe(sun)
        apparent = astrometric.apparent()
        alt_arr, az_arr, _ = apparent.altaz()

        az_degrees = az_arr.degrees
        alt_degrees = alt_arr.degrees

        diffs = (az_degrees - target_azimuth + 180.0) % 360.0 - 180.0

        crossings = []
        for i in range(len(diffs) - 1):
            if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
                if abs(diffs[i] - diffs[i+1]) < 180.0:
                    fraction = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
                    t_cross_tt = tt_array[i] + fraction * (tt_array[i+1] - tt_array[i])
                    alt_cross = alt_degrees[i] + fraction * (alt_degrees[i+1] - alt_degrees[i])

                    if alt_cross > 0:
                        crossings.append(self.ts.tt_jd(t_cross_tt))

        dt_val = t0.delta_t.item() if hasattr(t0.delta_t, 'item') else t0.delta_t

        lat_str = format_angle(lat).replace("+", "").replace("-", "-")
        lon_str = format_angle(lon).replace("+", "")

        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')}
- CUSTOM LOCATION, Long: {lon_str.replace(":", ":").replace('"', ',0')}, Lat: {lat_str.replace(":",
":").replace('"', ',0')}, Zone:{tz:.2f}
- No Summer Time.
- Delta T: {dt_val:.1f} Second(s)
================================================================================

  Date          Qiblah Time                  Qiblah Time
                First Time                 Second Time
"""
```

```
        time_1 = "----"
        time_2 = "----"

        if len(crossings) > 0:
            dt1 = crossings[0].utc_datetime() + datetime.timedelta(hours=tz)
            time_1 = dt1.strftime("%H:%M:%S")
        if len(crossings) > 1:
            dt2 = crossings[1].utc_datetime() + datetime.timedelta(hours=tz)
            time_2 = dt2.strftime("%H:%M:%S")

        data_row = f"\n{target_date.strftime('%d/%m/%Y')}       {time_1:^10}                {time_2:^10}
\n"

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

* Remarks:-
- Date format: dd/mm/yyyy.
- The Symbol '*' before the date, refers to Summer Time.
- '----' Means that the Sun / Sun's Shadow does not reach the Qiblah Direction on that day.
- Kindly notice that at certain locations and on certain days, the Sun / Sun's Shadow reaches
  Qiblah direction twice on the same day!
"""
        report = header + data_row + footer
        self.after(0, self.display_result, report)

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

  def show_qiblah_map(self):
    try:
        map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
        map_file = "map_topografi_qiblah.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=0.6, zorder=0)
            except Exception as e:
                print("Gagal memuat gambar peta via Pillow:", e)
        else:
            messagebox.showwarning("Peringatan", f"File gambar {map_file} gagal diunduh atau tidak
ditemukan di {full_map_path}")

        c = ax.contourf(LONS, LATS, Q, levels=np.arange(0, 361, 10), cmap='hsv', alpha=0.4, zorder=1)
        ax.scatter(39.8262, 21.4225, color='black', marker='*', s=200, zorder=10, label='Makkah (Kaaba)')

        ax.set_title("Accurate Times: Qiblah World Map", fontweight='bold')
        ax.set_xlabel("Longitude")
        ax.set_ylabel("Latitude")

        ax.set_xticks(np.arange(-180, 181, 30))
        ax.set_yticks(np.arange(-80, 81, 20))
        ax.grid(True, linestyle='-', color='gray', linewidth=0.5, zorder=2)
        ax.legend(loc='upper right')

        fig.tight_layout()
        plt.show()

    except Exception as e:
        messagebox.showerror("Error Map", f"Gagal memuat peta Kiblat: {e}")

  def calculate_moontimes(self):
    try:
        year = int(self.entry_mt_year.get())
        month = int(self.entry_mt_month.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_mt_lat.get())
```

```
lon = float(self.entry_mt_lon.get())
elev = float(self.entry_mt_elev.get())
tz = float(self.entry_mt_tz.get())

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

# Gunakan fungsi safe_monthrange agar kebal tahun minus
_, num_days = safe_monthrange(year, month)

# --- PENGAMAN RENTANG DATA ---
# Menghindari error batas ephemeris -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))

f_rs = almanac.risings_and_settings(self.eph, moon, loc)
t_rs, y_rs = almanac.find_discrete(t0_month, t1_month, f_rs)

# Inisialisasi dictionary untuk menampung kejadian per tanggal
# Kita gunakan string "Tahun-Bulan-Hari" sebagai kunci (Key) agar aman
events_by_date = defaultdict(lambda: {'rise': '----', 'transit': '----', 'set': '----'})

# 1. PROSES TERBIT & TERBENAM
if t_rs is not None:
    for t_val, y_val in get_safe_events(t_rs, y_rs):
        # Geser Julian Date secara manual untuk mendapatkan waktu Lokal
        t_lokal = self.ts.tt_jd(t_val.tt + (tz / 24.0))
        yl, ml, dl, hl, minl, sl = t_lokal.utc

        date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
        time_str = f"{int(hl):02d}:{int(minl):02d}"

        if y_val == 1: # Rising
            events_by_date[date_key]['rise'] = time_str
        else: # Setting
            events_by_date[date_key]['set'] = time_str

# 2. PROSES TRANSIT (KULMINASI) - Menggunakan metode pencarian titik tertinggi
for d_idx in range(safe_day_start, num_days + 1):
    t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
    # Ambil 1440 sampel (per menit) dalam 24 jam untuk mencari puncak altitude
    tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
```

```
            t_search = self.ts.tt_jd(tt_array)

            alt_arr, _, _ = (earth + loc).at(t_search).observe(moon).apparent().altaz()
            max_idx = np.argmax(alt_arr.degrees)

            t_max = self.ts.tt_jd(tt_array[max_idx])
            # Konversi ke lokal
            t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
            yl, ml, dl, hl, minl, sl = t_max_lokal.utc

            date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
            events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"

        # --- PEMBUATAN LAPORAN ---
        output_lines = []
        lat_str = format_angle(lat).replace("+", "").replace("-", "-")
        lon_str = format_angle(lon).replace("+", "")
        tahun_header = format_tahun_aman(year)

        header = f"""{self.get_header(84)}
{"[ Moon Times ]".center(84)}

* Settings:-
- Moon Times for:  {month:02d}/{tahun_header}
- LOKASI: Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f}
- Refraction is included.
================================================================================

  Date          Moonrise         Transit          Moonset
"""
        output_lines.append(header)

        for d in range(1, num_days + 1):
            date_key = f"{year}-{month}-{d}"
            ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})

            # Format tanggal display (dd/mm/yyyy)
            date_display = f"{d:02d}/{month:02d}/{year:04d}"
            if year <= 0:
                date_display = f"{d:02d}/{month:02d}/{abs(year-1):04d} SM"

            line = f"{date_display:<14}  {ev['rise']:^12}       {ev['transit']:^12}       {ev['set']:^12}"
            output_lines.append(line)
```

```
        footer =
"""================================================================================
=
* Remarks:-
- Date format: dd/mm/yyyy.
- '----' Means that the event does not occur on that day.
- Calculated using JPL Ephemeris Engine."""
        output_lines.append(footer)

        self.after(0, self.display_result, "\n".join(output_lines))

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

  def calculate_suntimes(self):
    try:
      year = int(self.entry_st_year.get())
      month = int(self.entry_st_month.get())

      self.auto_switch_ephemeris(year)

      lat = float(self.entry_st_lat.get())
      lon = float(self.entry_st_lon.get())
      elev = float(self.entry_st_elev.get())
      tz = float(self.entry_st_tz.get())

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

      # Gunakan fungsi safe_monthrange agar kebal tahun minus/BCE
      _, num_days = safe_monthrange(year, month)

      # --- PENGAMAN RENTANG DATA ---
      # Menghindari error batas ephemeris NASA de406 (-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 (ZAWAL) - Mencari titik kulminasi atas
        for d_idx in range(safe_day_start, num_days + 1):
            t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
            # Sampel 1440 titik (per menit) untuk akurasi tinggi
            tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
            t_search = self.ts.tt_jd(tt_array)

            alt_arr, _, _ = (earth + loc).at(t_search).observe(target_obj).apparent().altaz()
            max_idx = np.argmax(alt_arr.degrees)

            t_max = self.ts.tt_jd(tt_array[max_idx])
            # Konversi ke waktu lokal
            t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
            yl, ml, dl, hl, minl, sl = t_max_lokal.utc

            date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
            events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"

        # --- GENERATE LAPORAN TEKS ---
        output_lines = []
        lat_str = format_angle(lat).replace("+", "").replace("-", "-")
        lon_str = format_angle(lon).replace("+", "")
        tahun_header = format_tahun_aman(year)

        header = f"""{self.get_header(84)}
{"[ Sun Times ]".center(84)}
```

```
* Settings:-
- Sun Times for:  {month:02d}/{tahun_header}
- LOKASI: Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f}
- Refraction is included.
================================================================================

  Date          Sunrise          Transit          Sunset
"""
        output_lines.append(header)

        for d in range(1, num_days + 1):
            date_key = f"{year}-{month}-{d}"
            ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})

            # Format tampilan tanggal (Bebas crash tahun minus)
            date_display = f"{d:02d}/{month:02d}/{year:04d}"
            if year <= 0:
                # Contoh: -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 ASLI (DIPERTAHANKAN) ---
mazhab_val = self.combo_pt_mazhab.get()
method_val = self.combo_pt_method.get()

if "Hanafi" in mazhab_val:
   asr_factor = 2.0
   mazhab_name = "Hanafi"
else:
   asr_factor = 1.0
   mazhab_name = "Standard (Syafi'i, Maliki, Hanbali)"

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

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

# --- PERBAIKAN: GUNAKAN safe_monthrange UNTUK TAHUN MINUS ---
_, num_days = safe_monthrange(year, month)

# Pengaman batas bawah ephemeris de406 (-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("+", "")
        nama_kota_terpilih = self.opt_city.get()

        # --- FORMAT HEADER TABEL ASLI (DIPERTAHANKAN) ---
        if "HH:MM:SS" in fmt_pt:
            pad = 8
            hdr_line1 = "  Date       Fajer     Shuroq     Dhuha     Dhohur     Aser     Maghreb     Isha
Midnight"
            hdr_line2 = "            B. Twi.   Sunrise    +4.5°     Transit    -----     Sunset     E. Twi.   (Mid/Last
1/3)"
            sep =
"=============================================================================================
======================="
        else:
            pad = 5
            hdr_line1 = "  Date       Fajer    Shuroq    Dhuha     Dhohur    Aser     Maghreb    Isha
Midnight"
            hdr_line2 = "          B. Twi.   Sunrise   +4.5°    Transit    -----    Sunset    E. Twi.  (Mid/Last 1/3)"
            sep =
"=============================================================================================
=============="

        report_lines = []
        header_text = f"""{self.get_header(len(sep))}
{"[ Islamic Prayer Times ]".center(len(sep))}

* Settings:-
- Prayer times for:  {date_info}
- LOKASI: Long: {lon_str.replace(":", ":").replace('"', ',0')}, Lat: {lat_str.replace(":", ":").replace('"', ',0')}
({nama_kota_terpilih}), Ele:{elev}, Zone:{tz:.2f}
- 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)

        # --- FUNGSI LOKAL find_crossing (DIPERTAHANKAN) ---
        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

        # --- PERBAIKAN: fmt_time VERSI ANTI-CRASH (Bypass datetime) ---
        def fmt_time(tt_val, is_shuroq=False):
            if tt_val is None: return "----".center(pad)
            offset_i = (ikhtiyat_sec / 86400.0)
            if is_shuroq:
                t_res = self.ts.tt_jd(tt_val + (tz / 24.0) - offset_i)
            else:
                t_res = self.ts.tt_jd(tt_val + (tz / 24.0) + offset_i)

            _, _, _, h_f, m_f, s_f = t_res.utc # Ekstrak tuple angka mentah

            if "HH:MM:SS" in fmt_pt:
                return f"{int(h_f):02d}:{int(m_f):02d}:{int(s_f):02d}"
            else:
                return f"{int(h_f):02d}.{int(m_f):02d}"

        # --- LOOPING UTAMA (LOGIKA HISAB UTUH) ---
        for d in days_to_calc:
            t0 = self.ts.utc(year, month, d, -int(tz))
            t1 = self.ts.utc(year, month, d, 24 - int(tz))
            tt_array = np.linspace(t0.tt, t1.tt, 2880)
            t_search = self.ts.tt_jd(tt_array)

            # Perhitungan Toposentrik dengan Koreksi Atmosfer
```

```
        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))

        # Mencari Waktu Salat (Crossing)
        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')

        # --- PERBAIKAN: KALKULASI MIDNIGHT (Bypass datetime) ---
        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)

        # Format Tanggal Baris (dd/mm/yyyy)
        date_str = f"{d:02d}/{month:02d}/{format_tahun_aman(year)}"

        # --- MEMPERTAHANKAN FORMAT JARAK SPASI ASLI ---
        if "HH:MM:SS" in fmt_pt:
```

```
            line = f"{date_str:<15}  {str_fajr:^{pad}}  {str_shuroq:^{pad}}  {str_dhuha:^{pad}}
{str_dhohur:^{pad}}  {str_asr:^{pad}}  {str_maghreb:^{pad}}  {str_isha:^{pad}}  {str_midnight:^15}"
        else:
            line = f"{date_str:<12}  {str_fajr:^{pad}}  {str_shuroq:^{pad}}  {str_dhuha:^{pad}}
{str_dhohur:^{pad}}  {str_asr:^{pad}}  {str_maghreb:^{pad}}  {str_isha:^{pad}}  {str_midnight:^15}"

        report_lines.append(line)

    report_lines.append(f"{sep}")
    report_lines.append("""
* Remarks:-
- Date format: dd/mm/yyyy.
- Fajer: Beginning of Astronomical Twilight. | Shuroq: Sunrise.
- Dhohur: Transit of Sun. | Maghreb: Sunset. | Isha: End of Astronomical Twilight.
- Calculated using JPL NASA Engine. Supported Year -3000 to 3000.""")

    final_report = "\n".join(report_lines)

    # --- 1. TAMPILKAN TEKS KE LAYAR GUI (DIPERTAHANKAN) ---
    self.after(0, self.display_result, final_report)

    # --- 2. GENERATE GAMBAR OTOMATIS & BUKA (DIKEMBALIKAN UTUH) ---
    if "Bulanan" in mode_pt:
        self.after(0, lambda: self.lbl_status.configure(text="Sedang membuat gambar jadwal...",
text_color="#FFAB40"))

        template_path = os.path.join(BASE_DIR, "template_kosong.jpg")
        # Format file output agar unik per kota dan bulan
        output_filename = f"Jadwal_Shalat_{nama_kota_terpilih.replace(' ',
'_')}_{month:02d}_{year}.jpg"
        output_path = os.path.join(BASE_DIR, output_filename)

        if os.path.exists(template_path):
            try:
                # Jalankan fungsi utilitas pembuat gambar
                _util_generate_image(final_report, template_path, output_path)

                # Trigger Buka Gambar Otomatis berdasarkan Sistem Operasi
                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:\n{e}"))
        else:
            self.after(0, lambda: self.lbl_status.configure(text="Kalkulasi selesai (template_kosong.jpg tidak ditemukan).", text_color="#FFD54F"))

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

def calculate_visibility_map(self):
    # PASTIKAN CARTOPY TIDAK DIPAKAI LAGI AGAR AMAN DI-BUILD KE .EXE
    if not HAS_MAP_TOOLS:
        self.after(0, self.display_error, "Fitur Peta HD dinonaktifkan.\nLibrary 'scipy' tidak ditemukan.")
        return

    try:
        year = int(self.entry_vmyear.get())
        month = int(self.entry_vmmonth.get())
        day = int(self.entry_vmday.get())

        crit = self.combo_vmcrit.get()
        param_target = self.combo_vmparam.get()

        self.auto_switch_ephemeris(year)
        earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']

        t_noon_utc = self.ts.utc(year, month, day, 12)
        sun_geo = earth.at(t_noon_utc).observe(sun).apparent()
        _, dec_sun, _ = sun_geo.radec()
        dec_rad = dec_sun.radians

        lons_coarse = []
        lats_coarse = []
        alt_data = []
        elong_data = []
        arcv_data = []
        illum_data = []

        # [PERBAIKAN] Grid titik awal dirapatkan dari 10 menjadi 5 derajat
        grid_step = 5

        for lat in range(-90, 90, grid_step):
```

```
        lat_rad = math.radians(lat)

        for lon in range(-180, 180, grid_step):
            noon_utc_hour = 12.0 - (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

            # KODE PENGGANTI YANG AMAN UNTUK TAHUN SEBELUM MASEHI
            t_sunset = self.ts.utc(year, month, day, sunset_utc_hour)

            obs = earth.at(t_sunset)
            s_app = obs.observe(sun).apparent()
            m_app = obs.observe(moon).apparent()

            gmst = t_sunset.gmst
            lst_deg = (gmst * 15.0) + lon

            def quick_alt(app_obj, l_rad):
                ra, dec_o, _ = app_obj.radec()
                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))))

            m_alt = quick_alt(m_app, lat_rad)
            s_alt = quick_alt(s_app, lat_rad)
            elong = s_app.separation_from(m_app).degrees
            arcv = m_alt - s_alt

            illum = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
            illum_val = illum.item() if hasattr(illum, 'item') else illum

            lons_coarse.append(lon)
            lats_coarse.append(lat)
            alt_data.append(m_alt)
            elong_data.append(elong)
            arcv_data.append(arcv)
            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']))

            # Resolusi grid interpolasi dirapatkan menjadi 0.1
            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')

            # Menutup area NaN (tepi) dengan titik terdekat agar heatmap penuh
            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 (Matplotlib Standalone)")
        win_plot.geometry("1100x750")
        win_plot.attributes("-topmost", True)

        fig, ax = plt.subplots(figsize=(10, 6), dpi=120)

        # LOAD GAMBAR PETA TOPOGRAFI SEBAGAI BACKGROUND
        map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
        map_file = "map_topografi_qiblah.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=0.6, zorder=0)
                ax.set_ylim(-90, 90)
            except Exception as e:
                print("Gagal memuat gambar peta:", e)

        cf = None

        # --- 1. KRITERIA VISIBILITAS ---
        if param_target == "Kriteria Visibilitas":
            grid_score = np.zeros_like(grid_alt)
            if "KHGT" 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, cmap=cmap, norm=norm, alpha=0.55,
zorder=2, antialiased=True)

                p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi Kriteria KHGT')
                p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
                p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible (Bulan terbenam lebih dulu)')
                ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18),
fontsize='small')
            else:
                grid_score[:] = 1
                grid_score[(grid_arcv < 0)] = 0
```

```
            grid_score[(grid_arcv > 6.0) & (grid_elong > 8.0)] = 2
            grid_score[(grid_arcv > 8.0) & (grid_elong > 10.0)] = 3
            grid_score[(grid_arcv > 10.4) & (grid_elong > 12.0)] = 4

            cmap = ListedColormap(['#FF5252', '#FFFFFF', '#2979FF', '#E040FB', '#00E676'])
            bounds = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5]
            norm = BoundaryNorm(bounds, cmap.N)

            cf = ax.contourf(LONS_fine, LATS_fine, grid_score, cmap=cmap, norm=norm, alpha=0.55,
zorder=2, antialiased=True)

            p1 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
            p2 = mpatches.Patch(color='#FFFFFF', label='White: Not Possible')
            p3 = mpatches.Patch(color='#2979FF', label='Blue: Need Optical Aid')
            p4 = mpatches.Patch(color='#E040FB', label='Magenta: Seen by Naked Eye (Perfect Cond)')
            p5 = mpatches.Patch(color='#00E676', label='Green: Easily Visible by Naked Eye')
            ax.legend(handles=[p1, p2, p3, p4, p5], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -
0.22), fontsize='small')

        # --- 2. INTERSEKSI (AREA HIJAU KHGT & GARIS KONTUR) ---
        elif param_target == "Interseksi":
            grid_score = np.zeros_like(grid_alt)
            grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1

            # Setup Colormap transparan untuk area 0, dan Hijau untuk area 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)

            # Render arsiran hijau
            cf = ax.contourf(LONS_fine, LATS_fine, grid_score, cmap=cmap_inter, norm=norm_inter,
alpha=0.6, zorder=2, antialiased=True)

            # Render ulang garis kontur batas secara spesifik agar tertindih di atas arsiran
            cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0], colors=['#FFD54F'],
linewidths=2.5, linestyles='solid', zorder=3)
            ax.clabel(cs_alt, inline=True, fontsize=11, fmt='Alt 5°')

            cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0], colors=['#00E5FF'],
linewidths=2.5, linestyles='dashed', zorder=3)
            ax.clabel(cs_elong, inline=True, fontsize=11, fmt='Elong 8°')

            # Legend kustom
            p_interseksi = mpatches.Patch(color='#00E676', alpha=0.6, label='Area Hijau: Pemenuhan
KHGT (Alt >= 5° & Elong >= 8°)')
```

```
            line_alt = plt.Line2D([0], [0], color='#FFD54F', lw=2.5, linestyle='solid', label='Garis Batas Altitude 5°')
            line_elong = plt.Line2D([0], [0], color='#00E5FF', lw=2.5, linestyle='dashed', label='Garis Batas Elongasi 8°')
            ax.legend(handles=[p_interseksi, line_alt, line_elong], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.20), fontsize='small')

        # --- 3. PARAMETER HEATMAP REGULER ---
        else:
            if param_target == "Altitude Bulan":
                target_grid = grid_alt
                cmap_name = 'plasma'
                lbl = "Altitude Bulan (°)"
            elif param_target == "Elongasi":
                target_grid = grid_elong
                cmap_name = 'inferno'
                lbl = "Sudut Elongasi (°)"
            else:
                target_grid = grid_illum
                cmap_name = 'magma'
                lbl = "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')

        ax.set_title(f"High-Res Visibility Scanner (Interpolasi Halus 0.2°)\n{crit_name} - {date_str} - Layer: {param_target}", fontweight='bold', pad=10)
        ax.set_xlabel("Longitude")
        ax.set_ylabel("Latitude")
        ax.set_xticks(np.arange(-180, 181, 30))
        ax.set_yticks(np.arange(-60, 61, 20))
        ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=4)

        fig.tight_layout()

        frame_canvas = ctk.CTkFrame(win_plot)
        frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)

        canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
        canvas_plot.draw()
        canvas_plot.get_tk_widget().pack(fill="both", expand=True)

        toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
```

```
        toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))

        toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
        toolbar.update()

        def simpan_gambar_kustom():
            filepath = filedialog.asksaveasfilename(
                initialfile=f"HD_VisMap_{param_target.replace(' ', '')}.png",
                defaultextension=".png",
                filetypes=[("PNG Image", "*.png"), ("SVG Vector", "*.svg")]
            )
            if filepath:
                fig.savefig(filepath, dpi=300, bbox_inches='tight')
                messagebox.showinfo("Sukses", f"Peta HD berhasil diekspor ke:\n{filepath}")

        btn_export = ctk.CTkButton(toolbar_frame, text="📸 Export Gambar HD", font=("Segoe UI", 12,
"bold"), fg_color="#E65100", hover_color="#BF360C", command=simpan_gambar_kustom)
        btn_export.pack(side="right", padx=10)

        self.lbl_status.configure(text=f"Render Peta HD Interpolasi {param_target} 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 Grid
Data (5°) & Interpolasi Halus (0.2°) Selesai.\nPeta Visualisasi Kualitas Tinggi (HD) sedang ditampilkan
pada Window baru...\n\nLayer Heatmap  : {param_target}\nLibrary Engine : SciPy & Matplotlib (Tanpa
Cartopy, Aman 100% untuk EXE)")
        self.textbox.configure(state="disabled")

    except Exception as e:
        import traceback
        self.display_error(f"Gagal menampilkan HD Peta Interpolasi: {e}\n\n{traceback.format_exc()}")

  def calculate_qiblatime(self):
    try:
        # 1. Ambil Input (Aman untuk tahun minus)
        year = int(self.entry_qtyear.get())
        month = int(self.entry_qtmonth.get())
        day = int(self.entry_qtday.get())

        self.auto_switch_ephemeris(year)

        lat = float(self.entry_qtlat.get())
        lon = float(self.entry_qtlon.get())
```

```
tz = float(self.entry_qttz.get())

# 2. Hitung Azimut Kiblat (Rumus Spherical Trigonometry)
phi_k, lam_k = math.radians(21.4225), math.radians(39.8262) # Ka'bah
phi, lam = math.radians(lat), math.radians(lon)

y_q = math.sin(lam_k - lam)
x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
q_az = math.degrees(math.atan2(y_q, x_q)) % 360
shadow_az = (q_az + 180) % 360

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

# 3. Tentukan Rentang Pencarian (00:00 s.d 24:00 waktu lokal dalam UTC)
t0 = self.ts.utc(year, month, day, -int(tz))
t1 = self.ts.utc(year, month, day, 24 - int(tz))

# Gunakan resolusi tinggi (setiap menit) untuk mencari persilangan azimut
tt_array = np.linspace(t0.tt, t1.tt, 1440)
t_search = self.ts.tt_jd(tt_array)

obs = (earth + loc).at(t_search).observe(sun).apparent()
alt_arr, az_arr, _ = obs.altaz()
az_deg_list = az_arr.degrees
alt_deg_list = alt_arr.degrees

def find_crossing_times(target_az):
    # Hitung selisih sudut (handling 360 wrap around)
    diffs = (az_deg_list - target_az + 180) % 360 - 180
    results = []
    for i in range(len(diffs)-1):
        # Jika ada perubahan tanda, berarti ada persilangan
        if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
            # Pastikan matahari di atas ufuk saat itu terjadi
            if alt_deg_list[i] > 0:
                frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
                tt_res = tt_array[i] + frac * (tt_array[i+1] - tt_array[i])

                # Konversi ke waktu lokal
                t_final = self.ts.tt_jd(tt_res + (tz / 24.0))
                _, _, _, h, mn, s = t_final.utc
                results.append(f"{int(h):02d}:{int(mn):02d}:{int(s):02d}")
    return results

res_sun = ", ".join(find_crossing_times(q_az)) if find_crossing_times(q_az) else "----"
```

```
        res_shd = ", ".join(find_crossing_times(shadow_az)) if find_crossing_times(shadow_az) else "----"

        # Format Tanggal untuk Laporan (Anti Crash)
        tgl_str = f"{day:02d}/{month:02d}/{format_tahun_aman(year)}"

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

* Settings:-
- Tanggal         : {tgl_str}
- Lokasi          : Lat {lat}, Lon {lon}, TZ {tz:.2f}
- Azimut Kiblat    : {q_az:.2f}° (dari Utara ke Timur)
- Azimut Bayangan  : {shadow_az:.2f}°
================================================================================

1. WAKTU MATAHARI DI ARAH KIBLAT:
  (Saat matahari terlihat tepat di arah Ka'bah)
  >> {res_sun} LT

2. WAKTU BAYANGAN DI ARAH KIBLAT:
  (Saat bayangan benda tegak lurus mengarah ke Ka'bah)
  >> {res_shd} LT

* Catatan:
- "----" berarti fenomena tidak terjadi di lokasi pada tanggal tersebut.
- Rashdul Qiblah hanya terjadi saat matahari berada di atas ufuk (>0°).
- Kalkulasi berbasis JPL Ephemeris NASA ({self.ephemeris_name}).
================================================================================
"""
        self.after(0, self.display_result, report)

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

  # ========================================
  # FUNGSI GENERAL GUI
  # ========================================
  def display_result(self, report_text):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", report_text)
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Kalkulasi Selesai", text_color="#00E676")
    self.btn_hitung.configure(state="normal")
```

```
    def display_error(self, error_msg):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", f"TERJADI KESALAHAN (DEBUG):\n{error_msg}")
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
        self.btn_hitung.configure(state="normal")

    def auto_detect_location(self):
        self.lbl_status.configure(text="Mendeteksi lokasi...", text_color="#00E5FF")
        def fetch_location():
            try:
                import json
                import urllib.request
                # Menggunakan API gratis dari ip-api.com
                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 = data['lat']
                    lon = data['lon']
                    city = data['city']

                    # Update semua entry lokasi di GUI (harus di Main Thread)
                    self.after(0, self._update_all_loc_entries, lat, lon, city)
                else:
                    self.after(0, lambda: messagebox.showerror("Gagal", "Tidak dapat mendeteksi lokasi."))
            except Exception as e:
                self.after(0, lambda: messagebox.showerror("Error", f"Koneksi Geolocation Gagal: {e}"))
            finally:
                self.after(0, lambda: self.lbl_status.configure(text="Sistem Siap", text_color="#00E676"))

        threading.Thread(target=fetch_location, daemon=True).start()

    def _update_all_loc_entries(self, lat, lon, city):
        for ent in [self.entry_vlat, self.entry_eph_lat, self.entry_qlat, self.entry_mt_lat, self.entry_st_lat,
self.entry_pt_lat, self.entry_qtlat, self.entry_eph3d_lat]:
            ent.delete(0, 'end')
            ent.insert(0, str(lat))

        for ent in [self.entry_vlon, self.entry_eph_lon, self.entry_qlon, self.entry_mt_lon, self.entry_st_lon,
self.entry_pt_lon, self.entry_qtlon, self.entry_eph3d_lon]:
            ent.delete(0, 'end')
            ent.insert(0, str(lon))
```

```
        # ---> PERBAIKAN: Ubah teks dropdown di sidebar agar sesuai hasil deteksi <---
        self.opt_prov.set("Lokasi Otomatis (GPS)")
        self.opt_city.set(city)
        # ------------------------------------------------------------------------

        self.lokasi_nama.set(f"{city} (Auto-Detected)")
        messagebox.showinfo("Lokasi Ditemukan", f"Lokasi berhasil diset ke {city}\nLat: {lat}, Lon: {lon}")

    def export_to_ics(self):
        # Cek apakah jadwal salat sudah ada di memori
        if not self.daily_prayer_schedule:
            messagebox.showwarning("Peringatan", "Jadwal Salat belum tersedia di memori!\nSilakan
aktifkan 'Alarm Salat' di sidebar agar sistem menghitung jadwal hari ini terlebih dahulu.")
            return

        # Mengambil nilai zona waktu dari input untuk penyesuaian UTC
        try:
            tz_offset = float(self.entry_pt_tz.get())
        except ValueError:
            tz_offset = 7.0 # Default ke WIB (UTC+7) jika kosong

        # Membuat struktur Header file iCalendar murni
        ics_content = [
            "BEGIN:VCALENDAR",
            "VERSION:2.0",
            "PRODID:-//KHGT Times By Kasmui//ID",
            "CALSCALE:GREGORIAN"
        ]

        # Waktu pembuatan file (Stamp) dalam format UTC
        now_utc_str = datetime.datetime.now(datetime.timezone.utc).strftime('%Y%m%dT%H%M%SZ')

        # Looping setiap jadwal salat yang ada di memori
        for prayer_name, dt_local in self.daily_prayer_schedule.items():
            # Konversi waktu lokal (naive) kembali ke UTC karena standar ICS wajib UTC
            dt_utc = dt_local - datetime.timedelta(hours=tz_offset)

            # Format waktu sesuai standar ICS: YYYYMMDDThhmmssZ
            dt_start_str = dt_utc.strftime('%Y%m%dT%H%M%SZ')

            # Asumsi durasi event salat di kalender adalah 15 menit
            dt_end_str = (dt_utc + datetime.timedelta(minutes=15)).strftime('%Y%m%dT%H%M%SZ')

            # Membuat Unique ID untuk kalender
            uid = f"{dt_start_str}-{prayer_name.replace('/', '').replace(' ', '')}@khgttimes"
```

```
            # Memasukkan data event salat
            ics_content.extend([
                "BEGIN:VEVENT",
                f"UID:{uid}",
                f"DTSTAMP:{now_utc_str}",
                f"SUMMARY:Waktu Salat {prayer_name}",
                f"DTSTART:{dt_start_str}",
                f"DTEND:{dt_end_str}",
                "BEGIN:VALARM",
                "ACTION:DISPLAY",
                f"DESCRIPTION:Telah masuk waktu {prayer_name}",
                "TRIGGER:-PT10M",  # Setel pengingat (notifikasi) 10 menit sebelum waktu salat
                "END:VALARM",
                "END:VEVENT"
            ])

        ics_content.append("END:VCALENDAR")
        final_ics = "\n".join(ics_content)

        # Dialog penyimpanan file
        filepath = filedialog.asksaveasfilename(
            initialfile=f"Jadwal_Salat_{datetime.datetime.now().strftime('%Y%m%d')}.ics",
            defaultextension=".ics",
            filetypes=[("iCalendar Files", "*.ics")]
        )

        # Eksekusi penyimpanan ke format .ics
        if filepath:
            try:
                with open(filepath, 'w', encoding='utf-8') as f:
                    f.write(final_ics)
                messagebox.showinfo("Sukses", f"File Kalender ICS berhasil dibuat murni tanpa library eksternal!\nDisimpan di:\n{filepath}\n\nFile ini siap diimpor ke Google Calendar, Apple Calendar, atau Outlook.")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan file ICS: {e}")

    def export_to_pdf(self):
        try:
            from fpdf import FPDF
        except ImportError:
            messagebox.showerror("Library Kurang", "Silakan install library 'fpdf' (pip install fpdf).")
            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) # Menggunakan Courier agar format tabel TXT tetap lurus (Monospaced)

                # Header Surat
                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)

                # Isi Laporan
                pdf.set_font("Courier", size=8)
                for line in report_data.split('\n'):
                    # Escape karakter khusus jika perlu
                    safe_line = line.encode('latin-1', 'replace').decode('latin-1')
                    pdf.cell(0, 4, txt=safe_line, ln=True)

                pdf.output(filepath)
                messagebox.showinfo("Sukses", f"Laporan PDF berhasil dibuat:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error PDF", f"Gagal membuat PDF: {e}")

    def export_to_png(self):
        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 monospaced 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)

            # Render 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 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())

        # 2. Bypass string, gunakan Tuple PyEphem!
        waktu_tuple = (y, m, d, 12, 0, 0)
```

```
        # Update UI via main thread
        self.after(0, lambda: self.lbl_status.configure(text="Melacak Titik Pertama (Iterasi Multi-Hari)...",
text_color="#00E5FF"))

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

        try:
            ijtimak_1 = ephem.previous_new_moon(waktu_tuple)
            ijtimak_2 = ephem.next_new_moon(waktu_tuple)
        except Exception:
            self.after(0, self.display_error, "Format tanggal tidak valid. Harap periksa input.")
            return

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

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

            # Cek dari H+0 (Hari Ijtimak) sampai maksimal H+3
            for offset_hari in range(4):
                kandidat_hari_ini = []
                waktu_pencarian = ephem.Date(w_ijtimak + offset_hari)

                for negara, kota_dict in CITY_DB.items():
                    if negara in zona_dikecualikan:
                        continue

                    for nama_kota, koordinat in kota_dict.items():
                        lintang, bujur = koordinat
                        pengamat = ephem.Observer()
                        pengamat.lat = str(lintang)
                        pengamat.lon = str(bujur)
                        pengamat.elevation = 0

                        pengamat.date = waktu_pencarian
                        try:
                            waktu_sunset = pengamat.next_setting(matahari)
                        except (ephem.AlwaysUpError, ephem.NeverUpError):
                            continue # Lewati jika di kutub / tidak ada sunset

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

```
                    bulan.compute(pengamat)

                    # Kalkulasi Geo (Syarat KHGT)
                    HA_moon = pengamat.sidereal_time() - bulan.g_ra
                    sin_alt_geo = math.sin(pengamat.lat) * math.sin(bulan.g_dec) + math.cos(pengamat.lat)
* math.cos(bulan.g_dec) * math.cos(HA_moon)
                    alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

                    cos_elong_geo = math.sin(matahari.g_dec) * math.sin(bulan.g_dec) +
math.cos(matahari.g_dec) * math.cos(bulan.g_dec) * math.cos(matahari.g_ra - bulan.g_ra)
                    elong_geo = math.degrees(math.acos(max(-1.0, min(1.0, cos_elong_geo))))

                    if alt_geo >= 5.0 and elong_geo >= 8.0:
                        kandidat_hari_ini.append({
                            'kota': nama_kota,
                            'negara': negara,
                            'lat': lintang,
                            'lon': bujur,
                            'sunset_utc': waktu_sunset,
                            'alt_geo': alt_geo,
                            'elong_geo': elong_geo,
                            'ijtimak': w_ijtimak,
                            'label': label_siklus
                        })

                # Jika di hari ini sudah ada minimal 1 kota yang memenuhi kriteria,
                # Berhenti mencari di hari berikutnya! Ambil yang paling pertama terbenam.
                if kandidat_hari_ini:
                    kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
                    titik_hasil.append(kandidat_hari_ini[0])
                    break

            if not titik_hasil:
                self.after(0, self.display_error, "Tidak ada daratan utama yang memenuhi kriteria KHGT pada 2
siklus ini.")
                return

            self.after(0, self.show_first_point_map, titik_hasil)

        except Exception as e:
            import traceback
            self.after(0, self.display_error, f"Gagal menghitung Titik
Pertama:\n{str(e)}\n\nTraceback:\n{traceback.format_exc()}")

    def show_first_point_map(self, list_titik):
        try:
```

```
        win_plot = ctk.CTkToplevel(self)
        win_plot.title("Peta Titik Pertama Visibilitas KHGT (2 Siklus) & Analisis Gisborne")
        win_plot.geometry("1100x750")
        win_plot.attributes("-topmost", True)

        fig, ax = plt.subplots(figsize=(11, 6.5), dpi=100)

        # Load Peta Latar Belakang
        map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
        map_file = "map_topografi_qiblah.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=0.7, zorder=0)
                ax.set_ylim(-90, 90)
            except Exception as e:
                print("Gagal memuat gambar peta:", e)

        # Plot semua kota dari database sebagai noktah merah
        all_lats = []
        all_lons = []
        for negara, kota_dict in CITY_DB.items():
            for nama_kota, koordinat in kota_dict.items():
                all_lats.append(koordinat[0])
                all_lons.append(koordinat[1])

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

        # Konfigurasi visual marker
        warna_bintang = ['#FFD54F', '#00E676']
        warna_tepi = ['red', 'white']

        text_summary = "Pencarian Multi-Siklus Selesai.\n\nTitik daratan utama pertama di dunia yang
masuk bulan baru KHGT:\n\n"

        for idx, pt in enumerate(list_titik):
            c_star = warna_bintang[idx % len(warna_bintang)]
            c_edge = warna_tepi[idx % len(warna_tepi)]

            # Plot Bintang Titik Pertama
            ax.scatter(pt['lon'], pt['lat'], color=c_star, marker='*', s=600, edgecolor=c_edge, linewidth=1.5,
zorder=5)
```

```
            tz_approx = int(self.get_tz_from_lon(pt['lon']))
            tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"

            # KALKULASI WAKTU LOKAL (Bypass Python Datetime, pakai Ephem Math)
            # 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\n(Gagal Menghitung Fajar)\n»
Kesimpulan: LUSA masuk awal bulan"
         # ========================================================

         info_text = (
            f"TITIK KHGT ({pt['label'].upper()})\n"
            f"{'-'*45}\n"
            f"Lokasi  : {pt['kota']}, {pt['negara']}\n"
            f"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
            f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
            f"{'-'*45}\n"
            f"Parameter Waktu (UTC):\n"
            f" Waktu Maghrib: {ss_jam} UTC\n"
            f" Batas PKG 1  : {bt_jam} UTC\n"
            f" Ijtimak      : {ij_jam} UTC\n"
```

```
            f"{fajar_info}"
            f"{'-'*45}\n"
            f"{status_ijtimak}"
        )

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

        # --- 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})\nStatus: {status_ijtimak.replace(chr(10), ' ')}\n\n"
```

```
        ax.set_title("Titik Awal Pemenuhan Kriteria KHGT & Komparasi Fajar Gisborne",
fontweight='bold', pad=10)
        ax.set_xlabel("Longitude")
        ax.set_ylabel("Latitude")
        ax.set_xticks(np.arange(-180, 181, 30))
        ax.set_yticks(np.arange(-60, 61, 20))
        ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=1)

        fig.tight_layout()

        frame_canvas = ctk.CTkFrame(win_plot)
        frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)

        canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
        canvas_plot.draw()
        canvas_plot.get_tk_widget().pack(fill="both", expand=True)

        toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
        toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
        toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
        toolbar.update()

        self.lbl_status.configure(text=f"Pencarian Titik Pertama & Analisis Selesai",
text_color="#00E676")
        self.btn_hitung.configure(state="normal")

        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        text_summary += "Detail visual komparasi telah ditambahkan pada peta (Text Box melayang)."
        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")

        # --- PASTIKAN NAMA VARIABELNYA ADALAH 'self.textbox' BUKAN 'self.tex' ---
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        text_summary += "Detail visual komparasi telah ditambahkan pada peta (Text Box melayang)."
        self.textbox.insert("1.0", text_summary)
        self.textbox.configure(state="disabled")

    except Exception as e:
```

```
        import traceback
        self.display_error(f"Gagal menampilkan peta Titik Pertama: {e}\n\n{traceback.format_exc()}")
        tbox.insert("1.0", text_summary)
        self.textbox.configure(state="disabled")

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

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

            tz_wib = pytz.timezone('Asia/Jakarta')
            musim_nama = ["Vernal Equinox (Musim Semi Utara)",
                          "Summer Solstice (Musim Panas Utara)",
                          "Autumnal Equinox (Musim Gugur Utara)",
                          "Winter Solstice (Musim Dingin Utara)"]

            report = f"{self.get_header(85)}\n"
            report += f"{'[ Equinox & Solstice Calculator ]'.center(85)}\n\n"
            report += f"* Tahun Astronomis: {year} CE\n"
            report += f"* Zona Waktu: WIB (UTC+7)\n"
            report += "="*85 + "\n\n"

            for t_ev, y_ev in zip(t_seasons, y_seasons):
                dt_wib = t_ev.utc_datetime().replace(tzinfo=pytz.utc).astimezone(tz_wib)
                waktu_str = dt_wib.strftime('%d %B %Y - %H:%M:%S WIB')
                nama = musim_nama[y_ev]
                report += f" >> {nama:<38} : {waktu_str}\n"

            report += "\n" + "="*85
            self.after(0, self.display_result, report)
        except Exception as e:
            self.after(0, self.display_error, str(e))

    def calculate_planetary_times(self):
        try:
            year = int(self.entry_pl_year.get())
            month = int(self.entry_pl_month.get())
            target_name = self.combo_pl_target.get().lower()

            self.auto_switch_ephemeris(year)

            # Khusus planet, JPL Ephemeris menggunakan 'barycenter'
```

```
        target_obj = self.eph[f'{target_name} barycenter']
        earth = self.eph['earth']

        # Pinjam lokasi dari form general (visibility)
        lat, lon = float(self.entry_vlat.get()), float(self.entry_vlon.get())
        tz, elev = float(self.entry_vtz.get()), float(self.entry_velev.get())
        loc = wgs84.latlon(lat, lon, elevation_m=elev)

        _, num_days = calendar.monthrange(year, month)
        t0 = self.ts.utc(year, month, 1, -int(tz))
        t1 = self.ts.utc(year, month, num_days, 24 - int(tz))

        f_rs = almanac.risings_and_settings(self.eph, target_obj, loc)
        t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)

        report = f"{self.get_header(75)}\n"
        report += f"{f'[ Planetary Times: {target_name.upper()} ]'.center(75)}\n\n"
        report += f" Bulan/Tahun : {month:02d}/{year}\n"
        report += f" Lokasi      : Lat {lat}, Lon {lon}, TZ UTC+{tz}\n"
        report += "="*75 + "\n"
        report += " Tanggal       Terbit (Rise)      Terbenam (Set)\n"
        report += "-"*75 + "\n"

        events = defaultdict(lambda: {'rise': '----', 'set': '----'})
        for t_val, y_val in get_safe_events(t_rs, y_rs):
            dt = t_val.utc_datetime() + datetime.timedelta(hours=tz)
            if y_val == 1: events[dt.date()]['rise'] = dt.strftime("%H:%M")
            else: events[dt.date()]['set'] = dt.strftime("%H:%M")

        for d in range(1, num_days + 1):
            dt_key = datetime.date(year, month, d)
            ev = events[dt_key]
            report += f" {dt_key.strftime('%d/%m/%Y')}     {ev['rise']:^13}      {ev['set']:^13}\n"

        report += "="*75
        self.after(0, self.display_result, report)
    except Exception as e:
        self.after(0, self.display_error, str(e))

def display_result(self, report_text):
    self.textbox.configure(state="normal")
    self.textbox.delete("1.0", "end")
    self.textbox.insert("1.0", report_text)
    self.textbox.configure(state="disabled")
    self.lbl_status.configure(text="Kalkulasi Selesai", text_color="#00E676")
    self.btn_hitung.configure(state="normal")
```

```
    def display_error(self, error_msg):
        self.textbox.configure(state="normal")
        self.textbox.delete("1.0", "end")
        self.textbox.insert("1.0", f"TERJADI KESALAHAN (DEBUG):\n{error_msg}")
        self.textbox.configure(state="disabled")
        self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
        self.btn_hitung.configure(state="normal")

    def auto_detect_location(self):
        self.lbl_status.configure(text="Mendeteksi lokasi...", text_color="#00E5FF")
        def fetch_location():
            try:
                import json
                import urllib.request
                req = urllib.request.Request("http://ip-api.com/json/", headers={'User-Agent': 'Mozilla/5.0'})
                with urllib.request.urlopen(req, timeout=5) as response:
                    data = json.loads(response.read().decode())
                if data['status'] == 'success':
                    lat, lon, city = data['lat'], data['lon'], data['city']
                    self.after(0, self._update_all_loc_entries, lat, lon, city)
                else:
                    self.after(0, lambda: messagebox.showerror("Gagal", "Tidak dapat mendeteksi lokasi."))
            except Exception as e:
                self.after(0, lambda: messagebox.showerror("Error", f"Koneksi Geolocation Gagal: {e}"))
            finally:
                self.after(0, lambda: self.lbl_status.configure(text="Sistem Siap", text_color="#00E676"))
        threading.Thread(target=fetch_location, daemon=True).start()

    def _update_all_loc_entries(self, lat, lon, city):
        # 1. Update form koordinat dengan koordinat asli dari GPS
        for ent in [self.entry_vlat, self.entry_eph_lat, self.entry_qlat, self.entry_mt_lat, self.entry_st_lat,
self.entry_pt_lat, self.entry_qtlat, self.entry_eph3d_lat]:
            ent.delete(0, 'end')
            ent.insert(0, str(lat))

        for ent in [self.entry_vlon, self.entry_eph_lon, self.entry_qlon, self.entry_mt_lon, self.entry_st_lon,
self.entry_pt_lon, self.entry_qtlon, self.entry_eph3d_lon]:
            ent.delete(0, 'end')
            ent.insert(0, str(lon))

        # 2. Cari Provinsi dari CITY_DB berdasarkan nama kota hasil deteksi
        ditemukan_prov = None
        ditemukan_kota = city

        for prov_name, cities_dict in CITY_DB.items():
```

```
            for db_city in cities_dict.keys():
                # Pencarian fleksibel (misal GPS mendeteksi "Kota Semarang", cocok dengan "Semarang")
                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:
            # Jika kota dikenali di database, set provinsi dan list kotanya
            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 TIDAK ada di database (misal di desa terpencil), 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 Timezone secara otomatis berdasarkan Bujur
        try:
            tz_val = self.get_tz_from_lon(lon)
            for ent in [self.entry_vtz, self.entry_eph_tz, self.entry_qtz, self.entry_mt_tz, self.entry_st_tz,
self.entry_pt_tz, self.entry_qttz]:
                ent.delete(0, 'end')
                ent.insert(0, str(tz_val))
        except:
            pass

        # 5. Update Label atas dan pop-up sukses
        self.lokasi_nama.set(f"{ditemukan_kota} (Auto-Detected)")
        messagebox.showinfo("Lokasi Ditemukan", f"Lokasi berhasil diset ke {ditemukan_kota}\nLat: {lat},
Lon: {lon}")

    def export_to_ics(self):
        # Cek apakah jadwal salat sudah ada di memori
        if not self.daily_prayer_schedule:
```

```
        messagebox.showwarning("Peringatan", "Jadwal Salat belum tersedia di memori!\nSilakan aktifkan 'Alarm Salat' di sidebar agar sistem menghitung jadwal hari ini terlebih dahulu.")
        return

    # Mengambil nilai zona waktu dari input untuk penyesuaian UTC
    try:
        tz_offset = float(self.entry_pt_tz.get())
    except ValueError:
        tz_offset = 7.0 # Default ke WIB (UTC+7) jika kosong

    # Membuat struktur Header file iCalendar murni
    ics_content = [
        "BEGIN:VCALENDAR",
        "VERSION:2.0",
        "PRODID:-//KHGT Times By Kasmui//ID",
        "CALSCALE:GREGORIAN"
    ]

    # Waktu pembuatan file (Stamp) dalam format UTC
    now_utc_str = datetime.datetime.now(datetime.timezone.utc).strftime('%Y%m%dT%H%M%SZ')

    # Looping setiap jadwal salat yang ada di memori
    for prayer_name, dt_local in self.daily_prayer_schedule.items():
        # Konversi waktu lokal (naive) kembali ke UTC karena standar ICS wajib UTC
        dt_utc = dt_local - datetime.timedelta(hours=tz_offset)

        # Format waktu sesuai standar ICS: YYYYMMDDThhmmssZ
        dt_start_str = dt_utc.strftime('%Y%m%dT%H%M%SZ')

        # Asumsi durasi event salat di kalender adalah 15 menit
        dt_end_str = (dt_utc + datetime.timedelta(minutes=15)).strftime('%Y%m%dT%H%M%SZ')

        # Membuat Unique ID untuk kalender
        uid = f"{dt_start_str}-{prayer_name.replace('/', '').replace(' ', '')}@khgttimes"

        # Memasukkan data event salat
        ics_content.extend([
            "BEGIN:VEVENT",
            f"UID:{uid}",
            f"DTSTAMP:{now_utc_str}",
            f"SUMMARY:Waktu Salat {prayer_name}",
            f"DTSTART:{dt_start_str}",
            f"DTEND:{dt_end_str}",
            "BEGIN:VALARM",
            "ACTION:DISPLAY",
            f"DESCRIPTION:Telah masuk waktu {prayer_name}",
```

```
            "TRIGGER:-PT10M",  # Setel pengingat (notifikasi) 10 menit sebelum waktu salat
            "END:VALARM",
            "END:VEVENT"
        ])

    ics_content.append("END:VCALENDAR")
    final_ics = "\n".join(ics_content)

    # Dialog penyimpanan file
    filepath = filedialog.asksaveasfilename(
        initialfile=f"Jadwal_Salat_{datetime.datetime.now().strftime('%Y%m%d')}.ics",
        defaultextension=".ics",
        filetypes=[("iCalendar Files", "*.ics")]
    )

    # Eksekusi penyimpanan ke format .ics
    if filepath:
        try:
            with open(filepath, 'w', encoding='utf-8') as f:
                f.write(final_ics)
            messagebox.showinfo("Sukses", f"File Kalender ICS berhasil dibuat murni tanpa library eksternal!\nDisimpan di:\n{filepath}\n\nFile ini siap diimpor ke Google Calendar, Apple Calendar, atau Outlook.")
        except Exception as e:
            messagebox.showerror("Error", f"Gagal menyimpan file ICS: {e}")

def export_to_pdf(self):
    try:
        from fpdf import FPDF
    except ImportError:
        messagebox.showerror("Library Kurang", "Silakan install library 'fpdf' (pip install fpdf).")
        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) # Menggunakan Courier agar format tabel TXT tetap lurus
(Monospaced)

            # Header Surat
            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)

            # Isi Laporan
            pdf.set_font("Courier", size=8)
            for line in report_data.split('\n'):
               # Escape karakter khusus jika perlu
               safe_line = 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 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 hari ini)
        now = datetime.datetime.now()
        if target_date == now.date():
            curr_h = now.hour + now.minute/60.0
            ax.axvline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
            batas_bawah = min(sun_alts) if sun_alts else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekarang', color='#00E676', fontsize=10, rotation=90,
verticalalignment='bottom')

        # Formatting Label dan Font
        ax.set_title(f"Grafik Ketinggian Benda Langit: {target_date.strftime('%d %b %Y')}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggian / Altitude (°)", color='#ffffff', fontsize=12)

        ax.tick_params(colors='#ffffff', labelsize=11)
        ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)

        # Legend (Keterangan Garis)
        ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')
        fig.tight_layout()

        # Render murni ke dalam Tkinter Frame
        canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)

    def save_to_txt(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 yang bisa disimpan.")
            return

        mode = self.combo_mode.get()
        default_filename = "khgt_report.txt"

        filepath = filedialog.asksaveasfilename(
            initialfile=default_filename,
            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 analisis_komparasi_ramadhan_50_tahun(self):
        # Jalankan di thread terpisah agar GUI tidak Not Responding (Freeze)
        threading.Thread(target=self._proses_komparasi_ramadhan, daemon=True).start()

    def _proses_komparasi_ramadhan(self):
        # Update UI Status
        self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Ramadhan...",
text_color="#00E5FF"))
        self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
        self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
        self.after(0, lambda: self.textbox.delete("1.0", "end"))
        self.after(0, lambda: self.textbox.insert("end", "Memulai mesin Ephemeris untuk 50 Tahun (1447 H -
1496 H)...\nMohon tunggu...\n\n"))

        output_lines = []
        output_lines.append(self.get_header(125))
        output_lines.append("[ REKAPITULASI KOMPARASI AWAL RAMADHAN (1447 H - 1496 H)
]".center(125))
        output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°,
Eln>=6.4°)".center(125))
        output_lines.append("=" * 125)
```

```
    output_lines.append(f"{'Tahun':<8} | {'Waktu Ijtimak (UTC)':<19} | {'1 Ramadhan (KHGT)':<20} | {'1 Ramadhan (MABIMS Sabang)':<28} | {'Status / Selisih'}")
    output_lines.append("-" * 125)

    # Setup Observer Sabang untuk Neo MABIMS
    sabang = ephem.Observer()
    sabang.lat = math.radians(5.8942)
    sabang.lon = math.radians(95.3184)
    sabang.elevation = 0
    sabang.pressure = 1010
    sabang.temp = 25

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

    # Iterasi dari 1447 H sampai 1496 H
    for thn_h in range(1447, 1497):
      try:
        # 1. Ambil data 1 Ramadhan KHGT dari Database Bawaan
        # Index 8 adalah bulan ke-9 (Ramadan)
        data_ramadan = HIJRI_DB[thn_h][8]
        tgl_khgt_str = data_ramadan[2] # Format: DD-MMM-YYYY
        dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")

        # 2. Tanggal Rukyat (29 Syakban) adalah 1 hari sebelum 1 Ramadhan KHGT
        dt_rukyat = dt_khgt - datetime.timedelta(days=1)

        # 3. Cari Waktu Ijtimak di sekitar tanggal Rukyat
        waktu_pencarian = ephem.Date(dt_rukyat)
        ijtimak = ephem.previous_new_moon(waktu_pencarian + 5) # +5 hari sbg buffer pencarian
        dt_ijtimak_utc = ijtimak.datetime()
        str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")

        # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
        sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
        try:
          waktu_sunset = sabang.next_setting(matahari)
        except:
          waktu_sunset = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0)) # Fallback perkiraan UTC Sunset Sabang

        # 5. Hitung Posisi Bulan saat Sunset Sabang (Toposentrik)
        sabang.date = waktu_sunset
        matahari.compute(sabang)
        bulan.compute(sabang)
```

```
            alt_topo = math.degrees(bulan.alt)
            elong_topo = math.degrees(ephem.separation(matahari, bulan))

            # 6. Evaluasi Neo MABIMS (Alt >= 3° dan Elong >= 6.4°)
            # Serta pastikan ijtimak sudah terjadi sebelum sunset (umur bulan positif)
            umur_bulan = waktu_sunset - ijtimak

            if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
                dt_mabims = dt_khgt
                status = "Serentak (Sama)"
            else:
                # Jika gagal memenuhi MABIMS di Sabang, bulan Syakban digenapkan 30 hari
                dt_mabims = dt_khgt + datetime.timedelta(days=1)
                status = "Beda (MABIMS Mundur 1 Hari)"

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

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

            # Sisipkan pemisah per 10 Tahun (Dekade)
            if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
                output_lines.append("-" * 125)

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

    output_lines.append("=" * 125)
    output_lines.append("* Catatan Metodologi:")
    output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari struktur HIJRI_DB.")
    output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia (Sabang, Aceh).")
    output_lines.append("- Perbedaan biasanya terjadi jika KHGT tervalidasi di Benua Amerika, namun Sabang belum memenuhi syarat MABIMS.")

    laporan_final = "\n".join(output_lines)

    # Kembalikan ke UI (Main Thread)
    self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
    self.after(0, lambda: self.textbox.configure(state="disabled"))
    self.after(0, lambda: self.lbl_status.configure(text="Rekapitulasi 50 Tahun Selesai", text_color="#00E676"))
    self.after(0, lambda: self.btn_hitung.configure(state="normal"))
```

```
# ========================================
# FUNGSI PEMBERSIH SAAT APLIKASI DITUTUP
# ========================================
def on_closing(self):
    self.anim_running = False
    self.is_viewing_3d = False
    self.eph3d_is_live = False
    self.alarm_enabled.set(False)

    if HAS_PYGAME:
        try:
            import pygame
            pygame.mixer.music.stop()
            pygame.quit()
        except:
            pass

    self.destroy()
    import os
    os._exit(0)

def get_tz_from_lon(self, lon_val):
    """Menentukan Zona Waktu Administratif (Indonesia & Override Kustom) atau Geografis Global."""

    # 1. ZONA WAKTU INDONESIA (WIB, WITA, WIT)
    if 94.0 <= lon_val < 115.0:
        return 7.0
    elif 115.0 <= lon_val < 126.0:
        return 8.0
    elif 126.0 <= lon_val <= 141.0:
        return 9.0

    # 2. OVERRIDE ZONA WAKTU ALASKA -> UTC-11
    # Memotong 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 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'); self.entry_anim_time.insert(0, now.strftime("%H:%M:%S"))

    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 = datetime.datetime.now().strftime("%H:%M:%S")
                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

            try:
                lon_val = float(self.entry_vlon.get())
            except:
                lon_val = 110.4100

            tz_offset = int(self.get_tz_from_lon(lon_val))

            # PERBAIKAN: Gunakan konstruktor utc() dari Skyfield secara langsung
            # untuk menghindari limitasi tahun 1-9999 pada datetime.datetime
            # Jam lokal dikonversi ke UTC dengan mengurangi offset
            self.anim_custom_time = self.ts.utc(y, m, d, jam - tz_offset, menit, detik)

            self.lbl_status.configure(text=f"Waktu Kustom Diterapkan: {y}", text_color="#FFAB40")
        except Exception as e:
            messagebox.showerror("Error Input", f"Format tidak valid!\nError: {e}")
            self.anim_start_live()

    def analisis_komparasi_syawal_50_tahun(self):
        threading.Thread(target=self._proses_komparasi_syawal, daemon=True).start()

    def _proses_komparasi_syawal(self):
        # Update UI Status
        self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Syawal...",
text_color="#00E5FF"))
```

```
        self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
        self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
        self.after(0, lambda: self.textbox.delete("1.0", "end"))
        self.after(0, lambda: self.textbox.insert("end", "Memulai komputasi Ephemeris untuk penentuan 1 SYAWAL (1447 H - 1496 H)...\nMohon tunggu...\n\n"))

        output_lines = []
        output_lines.append(self.get_header(125))
        output_lines.append("[ REKAPITULASI KOMPARASI AWAL SYAWAL / IDUL FITRI (1447 H - 1496 H) ]".center(125))
        output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°, Eln>=6.4°)".center(125))
        output_lines.append("=" * 125)
        output_lines.append(f"{'Tahun':<8} | {'Waktu Ijtimak (UTC)':<19} | {'1 Syawal (KHGT)':<20} | {'1 Syawal (MABIMS Sabang)':<28} | {'Status / Selisih'}")
        output_lines.append("-" * 125)

        # Setup Observer Sabang untuk Neo MABIMS
        sabang = ephem.Observer()
        sabang.lat = math.radians(5.8942)
        sabang.lon = math.radians(95.3184)
        sabang.elevation = 0
        sabang.pressure = 1010
        sabang.temp = 25

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

        # Iterasi dari 1447 H sampai 1496 H
        for thn_h in range(1447, 1497):
            try:
                # 1. Ambil data 1 Syawal KHGT dari Database Bawaan (Index 9 adalah Syawal)
                data_syawal = HIJRI_DB[thn_h][9]
                tgl_khgt_str = data_syawal[2] # Format: DD-MMM-YYYY
                dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")

                # 2. Tanggal Rukyat (29 Ramadan) adalah 1 hari sebelum 1 Syawal KHGT
                dt_rukyat = dt_khgt - datetime.timedelta(days=1)

                # 3. Cari Waktu Ijtimak Akhir Ramadan
                waktu_pencarian = ephem.Date(dt_rukyat)
                ijtimak = ephem.previous_new_moon(waktu_pencarian + 5)
                dt_ijtimak_utc = ijtimak.datetime()
                str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")

                # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
```

```
            sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
            try:
                waktu_sunset = sabang.next_setting(matahari)
            except:
                waktu_sunset = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0))

            # 5. Hitung Posisi Bulan saat Sunset Sabang
            sabang.date = waktu_sunset
            matahari.compute(sabang)
            bulan.compute(sabang)

            alt_topo = math.degrees(bulan.alt)
            elong_topo = math.degrees(ephem.separation(matahari, bulan))

            # 6. Evaluasi Neo MABIMS
            umur_bulan = waktu_sunset - ijtimak

            if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
                dt_mabims = dt_khgt
                status = "Serentak (Sama)"
            else:
                # Jika MABIMS Sabang belum tembus, maka Idul Fitri mundur 1 hari (Ramadan istikmal 30
hari)
                dt_mabims = dt_khgt + datetime.timedelta(days=1)
                status = "Beda (MABIMS Mundur 1 Hari)"

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

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

            # Sisipkan pemisah per 10 Tahun
            if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
                output_lines.append("-" * 125)

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

    output_lines.append("=" * 125)
    output_lines.append("* Catatan Metodologi Penentuan 1 Syawal:")
    output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari
struktur HIJRI_DB.")
    output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia
(Sabang, Aceh).")
```

```
        output_lines.append("- Jika status 'Beda (MABIMS Mundur 1 Hari)', artinya KHGT sudah merayakan Idul Fitri, sedangkan wilayah MABIMS masih berpuasa (istikmal).")

        laporan_final = "\n".join(output_lines)

        # Lempar ke Main Thread
        self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
        self.after(0, lambda: self.textbox.configure(state="disabled"))
        self.after(0, lambda: self.lbl_status.configure(text="Komparasi 50 Tahun Syawal Selesai", text_color="#00E676"))
        self.after(0, lambda: self.btn_hitung.configure(state="normal"))

    def _proses_tabel_elongasi_50_tahun(self):
        # 1. Fungsi Pembantu untuk update GUI (Thread-Safe / Anti-Kosong)
        def _set_ui_loading():
            self.lbl_status.configure(text="Menganalisis Elongasi 50 Tahun...", text_color="#00E5FF")
            self.textbox.configure(state="normal", wrap="none")
            self.textbox.delete("1.0", "end")
            self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi)...\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 ke layar
        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], 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")

                # Khusus menu 23, Eln kita letakkan di depan
```

```
            str_khgt = "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 + 2)
                    str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

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

```
            # Evaluasi MABIMS di 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="Menganalisis 50 Tahun 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 (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, 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")

        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_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_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: 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, 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 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)"

                # Format output Eln lebih dulu
                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()}"))

# ========================================
# MODUL TAMBAHAN 26: KALENDER HIJRIAH BERJALAN
# ========================================
def reset_kalender(self, update_ui=True):
    today = datetime.date.today()
    found_y, found_m = 1447, 0 # Default aman

    # Deteksi otomatis bulan hijriah saat ini
    for y, months in HIJRI_DB.items():
        for m_idx, m_data in enumerate(months):
            start = HijriConverter.parse_date(m_data[2])
            if start and start <= today < start + datetime.timedelta(days=m_data[3]):
                found_y, found_m = y, m_idx
                break
    self.cal_h_year = found_y
    self.cal_h_month = found_m

    if update_ui:
        self.render_kalender()

def setup_kalender_out_frame(self):
    self.frame_kalender_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")

    # --- Header Navigasi & Cetak ---
    header_cal = ctk.CTkFrame(self.frame_kalender_out, fg_color="#101018", corner_radius=8)
    header_cal.pack(fill="x", padx=15, pady=(0, 10))

    btn_prev = ctk.CTkButton(header_cal, text="◀ Sebelumnya", command=self.cal_prev_month,
fg_color="#1F1F1F", hover_color="#333333", width=120)
    btn_prev.pack(side="left", padx=15, pady=10)

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

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

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

        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 batas
                self.cal_h_month = 0
                messagebox.showinfo("Batas Data", "Telah mencapai batas awal database.")
                return
        self.render_kalender()

    def cal_next_month(self):
        self.cal_h_month += 1
        if self.cal_h_month > 11:
            self.cal_h_month = 0
            self.cal_h_year += 1
            if self.cal_h_year not in HIJRI_DB:
                self.cal_h_year -= 1
                self.cal_h_month = 11
                messagebox.showinfo("Batas Data", "Telah mencapai batas akhir database.")
                return
        self.render_kalender()

    def render_kalender(self):
```

```
        # Bersihkan grid lama
        for widget in self.grid_cal_frame.winfo_children():
            widget.destroy()

        y = self.cal_h_year
        m = self.cal_h_month

        if y not in HIJRI_DB: return

        m_data = HIJRI_DB[y][m]
        nama_bulan, _, start_date_str, jumlah_hari = m_data

        # Konversi string Masehi (e.g. 26-Jun-2025) ke datetime
        bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
        parts = start_date_str.split('-')
        start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
        end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

        # Update Judul Header
        hijri_title = f"{nama_bulan} {y} H"
        greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
        if start_date.month != end_date.month:
            greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"

        self.lbl_cal_title.configure(text=f"{hijri_title}\n({greg_title})")

        # Kolom Header Hari
        days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
        for col, day_name in enumerate(days_header):
            lbl = ctk.CTkLabel(self.grid_cal_frame, text=day_name, font=("Segoe UI", 14, "bold"),
fg_color="#1E88E5", corner_radius=5)
            lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")

        # Cari titik awal kolom (Hari Masehi Senin=0, Ahad=6. Kita ubah Ahad=0, Senin=1)
        offset = (start_date.weekday() + 1) % 7

        row, col = 1, offset
        current_g_date = start_date

        # Konstruksi Grid Hari
        for h_day in range(1, jumlah_hari + 1):
            # Frame Per Kotak
            cell = ctk.CTkFrame(self.grid_cal_frame, fg_color="#1E1E1E", corner_radius=8, border_width=1,
border_color="#424242")
            cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")
```

```
        # Angka Hijriah (Besar di Tengah)
        lbl_h = ctk.CTkLabel(cell, text=str(h_day), font=("Consolas", 32, "bold"), text_color="#FFD54F")
        lbl_h.pack(expand=True)

        # Angka Masehi (Kecil di Bawah Kanan)
        m_str = f"{current_g_date.day} {BULAN_MASEHI[current_g_date.month-1][:3]}
{current_g_date.year}"

        # Tandai Jumat warna sedikit beda
        color_masehi = "#00E676" if col == 5 else "#B0BEC5"
        lbl_g = ctk.CTkLabel(cell, text=m_str, font=("Segoe UI", 11), text_color=color_masehi)
        lbl_g.pack(side="bottom", anchor="e", padx=8, pady=3)

        # Update Posisi
        col += 1
        if col > 6:
            col = 0
            row += 1
        current_g_date += datetime.timedelta(days=1)

    # Proporsi Seragam Grid Matriks
    for i in range(7):
        self.grid_cal_frame.grid_columnconfigure(i, weight=1)
    for i in range(7): # Alokasi hingga 6 baris + 1 header
        self.grid_cal_frame.grid_rowconfigure(i, weight=1)

  def _generate_calendar_image(self):
    """Merender kanvas elegan murni lewat Pillow untuk resolusi Ultra HD"""
    img = Image.new("RGB", (1400, 1000), "#0A0A10")
    draw = ImageDraw.Draw(img)

    try:
        f_title = ImageFont.truetype("arialbd.ttf", 46)
        f_sub = ImageFont.truetype("arial.ttf", 26)
        f_day = ImageFont.truetype("arialbd.ttf", 22)
        f_h = ImageFont.truetype("arialbd.ttf", 64)
        f_m = ImageFont.truetype("arial.ttf", 16)
    except Exception:
        # Fallback jika komputer tidak memiliki ttf
        f_title = f_sub = f_day = f_h = f_m = ImageFont.load_default()

    y = self.cal_h_year
    m = self.cal_h_month
    m_data = HIJRI_DB[y][m]
    nama_bulan, _, start_date_str, jumlah_hari = m_data
```

```
    bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9,
"Oct":10, "Nov":11, "Dec":12}
    parts = start_date_str.split('-')
    start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
    end_date = start_date + datetime.timedelta(days=jumlah_hari-1)

    # Tulis Judul
    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")

    # Rendering Grid (Kotak-kotak)
    start_x, start_y = 50, 180
    cell_w, cell_h = 185, 120
    days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]

    # 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
      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")

    # Gambar Isi Tanggal
    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

      # Background Box
      draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)

      # Angka Hijriah
```

```
        draw.text(((x0+x1)/2, y0 + 40), str(h_day), font=f_h, fill="#FFD54F", 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 V7.0 - By Kasmui", font=f_sub, fill="#555555",
anchor="mm")
    return img

  def export_kalender(self, export_format):
    img = self._generate_calendar_image()

    y = self.cal_h_year
    m_data = HIJRI_DB[y][self.cal_h_month]
    nama_bulan = m_data[0]
    default_file = f"Kalender_{nama_bulan}_{y}H"

    if export_format == "png":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG
di:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")

    elif export_format == "pdf":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.pdf",
            defaultextension=".pdf",
            filetypes=[("PDF Document", "*.pdf")]
        )
        if filepath:
            try:
```

```
            # Mengkonversi Image PIL langsung menjadi PDF beresolusi penuh
            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 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-1]} {y}"

        # Ekstrak bulan dan tahun Hijriah (misal: "1 Muharam 1446 H" -> "Muharam 1446")
        try:
            h1_parts = h_start.split(" ")
            h2_parts = h_end.split(" ")
            h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
            h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
            if h1_m_y != h2_m_y:
                hijri_title = f"{h1_m_y} - {h2_m_y} H"
```

```
            else:
                hijri_title = f"{h1_m_y} H"
        except:
            hijri_title = "Data Hijriah KHGT"

        self.lbl_calm_title.configure(text=f"{greg_title}\n({hijri_title})")

        # Header Hari Masehi (Senin di awal, Ahad di akhir)
        days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
        for col, day_name in enumerate(days_header):
            color_bg = "#D84315" if col == 6 else "#E65100" # Ahad warnanya lebih gelap
            lbl = ctk.CTkLabel(self.grid_calm_frame, text=day_name, font=("Segoe UI", 14, "bold"),
fg_color=color_bg, corner_radius=5)
            lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")

        # Offset kalender Masehi (Monday=0, Sunday=6)
        offset = start_date.weekday()
        row, col = 1, offset

        curr_g = start_date
        for m_day in range(1, jumlah_hari + 1):
            cell = ctk.CTkFrame(self.grid_calm_frame, fg_color="#1E1E1E", corner_radius=8, border_width=1,
border_color="#424242")
            cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")

            # Angka Masehi Besar (Warna merah khusus untuk Ahad)
            color_masehi = "#FF5252" if col == 6 else "#00E5FF"
            lbl_m = ctk.CTkLabel(cell, text=str(m_day), font=("Consolas", 32, "bold"),
text_color=color_masehi)
            lbl_m.pack(expand=True)

            # Angka Hijriah Kecil di bawah
            h_str = HijriConverter.get_hijri_date(curr_g)
            if h_str == "N/A": h_str = "-"

            # Singkat nama bulan agar muat (Misal: "Jumadilakhir" -> "Jum")
            parts = h_str.split()
            if len(parts) >= 4:
                h_str_short = f"{parts[0]} {parts[1][:3]} {parts[2]} H"
            else:
                h_str_short = h_str

            lbl_h = ctk.CTkLabel(cell, text=h_str_short, font=("Segoe UI", 11), text_color="#FFD54F")
            lbl_h.pack(side="bottom", anchor="e", padx=8, pady=3)

            col += 1
```

```
            if col > 6:
                col = 0
                row += 1
            curr_g += datetime.timedelta(days=1)

        # Proporsi Seragam
        for i in range(7):
            self.grid_calm_frame.grid_columnconfigure(i, weight=1)
        for i in range(7):
            self.grid_calm_frame.grid_rowconfigure(i, weight=1)

    def _generate_calmasehi_image(self):
        """Merender kanvas kalender masehi lewat Pillow untuk ekspor PDF & PNG HD"""
        img = Image.new("RGB", (1400, 1000), "#0A0A10")
        draw = ImageDraw.Draw(img)

        try:
            f_title = ImageFont.truetype("arialbd.ttf", 46)
            f_sub = ImageFont.truetype("arial.ttf", 26)
            f_day = ImageFont.truetype("arialbd.ttf", 22)
            f_m = ImageFont.truetype("arialbd.ttf", 64)
            f_h = ImageFont.truetype("arial.ttf", 16)
        except Exception:
            f_title = f_sub = f_day = f_m = f_h = ImageFont.load_default()

        y = self.cal_m_year
        m = self.cal_m_month

        _, jumlah_hari = calendar.monthrange(y, m)
        start_date = datetime.date(y, m, 1)
        end_date = datetime.date(y, m, jumlah_hari)

        h_start = HijriConverter.get_hijri_date(start_date)
        h_end = HijriConverter.get_hijri_date(end_date)

        greg_title = f"Kalender Masehi: {BULAN_MASEHI[m-1]} {y}"

        try:
            h1_parts = h_start.split(" ")
            h2_parts = h_end.split(" ")
            h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
            h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
            if h1_m_y != h2_m_y:
                hijri_title = f"Data KHGT: {h1_m_y} - {h2_m_y} H"
            else:
                hijri_title = f"Data KHGT: {h1_m_y} H"
```

```
except:
    hijri_title = "Data Hijriah KHGT"

# Cetak Header Text
draw.text((700, 60), greg_title, font=f_title, fill="#00E5FF", anchor="mm")
draw.text((700, 110), hijri_title, font=f_sub, fill="#FFD54F", anchor="mm")

# Konfigurasi Dimensi Grid
start_x, start_y = 50, 180
cell_w, cell_h = 185, 120
days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]

# Gambar Header Hari
for i, d_name in enumerate(days_header):
    x0 = start_x + i * cell_w
    y0 = start_y
    x1 = x0 + cell_w - 10
    y1 = y0 + 45
    color_bg = "#D84315" if i == 6 else "#E65100"
    draw.rectangle([x0, y0, x1, y1], fill=color_bg, outline="#E65100", width=2)
    draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")

# Gambar Isi Tanggal
offset = start_date.weekday()
row, col = 0, offset
curr_g = start_date
grid_y_start = start_y + 60

for m_day in range(1, jumlah_hari + 1):
    x0 = start_x + col * cell_w
    y0 = grid_y_start + row * cell_h
    x1 = x0 + cell_w - 10
    y1 = y0 + cell_h - 10

    draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)

    # 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

        draw.text((x1 - 10, y1 - 15), h_str_short, font=f_h, fill="#FFD54F", 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 V7.0 - By Kasmui", font=f_sub, fill="#555555",
anchor="mm")
    return img

  def export_kalmasehi(self, export_format):
    img = self._generate_calmasehi_image()

    y = self.cal_m_year
    nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
    default_file = f"Kalender_Masehi_{nama_bulan}_{y}"

    if export_format == "png":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.png",
            defaultextension=".png",
            filetypes=[("PNG Image", "*.png")]
        )
        if filepath:
            try:
                img.save(filepath, "PNG")
                messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG
di:\n{filepath}")
            except Exception as e:
                messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")

    elif export_format == "pdf":
        filepath = filedialog.asksaveasfilename(
            initialfile=f"{default_file}.pdf",
            defaultextension=".pdf",
            filetypes=[("PDF Document", "*.pdf")]
        )
        if filepath:
            try:
```

```
            # Konversi otomatis Image PIL menjadi PDF resolusi tinggi
            img.save(filepath, "PDF", resolution=150.0)
            messagebox.showinfo("Sukses", f"Kalender HD berhasil diekspor menjadi PDF
di:\n{filepath}")
        except Exception as e:
            messagebox.showerror("Error", f"Gagal menyimpan PDF: {e}")

    def anim_cari_sunset(self):
        self.anim_is_live = False
        try:
            # 1. Ambil data input tanggal
            y = int(self.entry_anim_year.get())
            m = int(self.entry_anim_month.get())
            d = int(self.entry_anim_day.get())

            # 2. Ambil Lokasi
            try:
                lat_val = float(self.entry_vlat.get())
                lon_val = float(self.entry_vlon.get())
                elev_val = float(self.entry_velev.get())
            except Exception:
                lat_val, lon_val, elev_val = -7.0667, 110.4100, 230.0

            tz_offset = int(self.get_tz_from_lon(lon_val))

            # 3. MENGGUNAKAN PYEPHEM MURNI (Bukan Skyfield)
            # PyEphem memproses waktu secara skalar, bebas dari error "single Time / array"
            import ephem
            pengamat = ephem.Observer()
            pengamat.lat = str(lat_val)
            pengamat.lon = str(lon_val)
            pengamat.elevation = elev_val
            pengamat.pressure = 1010  # Standar atmosfer
            pengamat.temp = 25        # Standar suhu
            matahari = ephem.Sun()

            # Mulai pencarian dari jam 12:00 siang Waktu Lokal (diubah ke UTC)
            waktu_mulai_cari = datetime.datetime(y, m, d, 12, 0, 0) - datetime.timedelta(hours=tz_offset)
            pengamat.date = ephem.Date(waktu_mulai_cari)

            # Cari waktu sunset (Terbenam)
            try:
                # Menghasilkan datetime UTC murni
                waktu_sunset_utc = pengamat.next_setting(matahari).datetime()

                # Konversi ke waktu lokal untuk ditampilkan di form GUI
```

```
            dt_lokal = waktu_sunset_utc + datetime.timedelta(hours=tz_offset)

            # Konversi ke format Skyfield Time HANYA untuk kebutuhan Animasi
            self.anim_custom_time = self.ts.from_datetime(waktu_sunset_utc.replace(tzinfo=pytz.utc))

            # 4. Update Angka di Form UI
            self.entry_anim_year.delete(0, 'end')
            self.entry_anim_year.insert(0, str(dt_lokal.year))

            self.entry_anim_month.delete(0, 'end')
            self.entry_anim_month.insert(0, f"{dt_lokal.month:02d}")

            self.entry_anim_day.delete(0, 'end')
            self.entry_anim_day.insert(0, f"{dt_lokal.day:02d}")

            self.entry_anim_time.delete(0, 'end')
            self.entry_anim_time.insert(0, dt_lokal.strftime("%H:%M:%S"))

        except (ephem.AlwaysUpError, ephem.NeverUpError):
            messagebox.showwarning("Peringatan", "Matahari tidak terbenam pada hari/lokasi tersebut
(Anomali Kutub).")
            self.anim_start_live()

    except Exception as e:
        import traceback
        # Memunculkan log lengkap jika gagal, agar mudah dilacak
        messagebox.showerror("Error", f"Gagal mencari waktu sunset:\n{e}\n\nDetail
Error:\n{traceback.format_exc()}")
        self.anim_start_live()

  # ========================================
  # MODUL 04: ALTITUDE CHART ANALYSER (FINAL & ANTI-FREEZE)
  # ========================================
  def calculate_chart_analyser(self):
    try:
        # 1. Ambil input UI
        y = int(self.entry_chart_y.get())
        m = int(self.entry_chart_m.get())
        d = int(self.entry_chart_d.get())

        lat = float(self.entry_vlat.get())
        lon = float(self.entry_vlon.get())
        tz = float(self.entry_vtz.get())
        elev = float(self.entry_velev.get())

        # Tampilkan pesan loading di GUI
```

```
        self.after(0, lambda: self.lbl_status.configure(text="Memproses Data Grafik...",
text_color="#00E5FF"))

        # 2. KALKULASI DI BACKGROUND THREAD (Bebas Tahun Minus)
        self.auto_switch_ephemeris(y)
        earth = self.eph['earth']
        sun = self.eph['sun']
        moon = self.eph['moon']
        loc = wgs84.latlon(lat, lon, elevation_m=elev)
        observer = earth + loc

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

        for h in hours:
            t = self.ts.utc(y, m, d, h - tz) # Skyfield menangani tahun minus dengan aman!
            sun_alts.append(observer.at(t).observe(sun).apparent().altaz()[0].degrees)
            moon_alts.append(observer.at(t).observe(moon).apparent().altaz()[0].degrees)

        # 3. KIRIM HASIL KE MAIN THREAD UNTUK PLOTTING (Aman dari Thread-Lock Matplotlib)
        self.after(0, self.plot_altitude_curve, self.frame_chart_out, y, m, d, hours, sun_alts, moon_alts)

    except Exception as e:
        import traceback
        self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")

  def _force_show_error(self, err_msg):
    """Memaksa textbox muncul kembali jika terjadi error di mode grafik"""
    if hasattr(self, 'frame_chart_out'):
        self.frame_chart_out.grid_remove()
    self.textbox.grid(row=1, column=0, sticky="nsew")
    self.display_error(err_msg)

  def plot_altitude_curve(self, canvas_frame, y, m, d, hours, sun_alts, moon_alts):
    try:
        # 1. Bersihkan Canvas Lama
        for widget in canvas_frame.winfo_children():
            widget.destroy()

        # 2. Gunakan OOP Figure Matplotlib (Lebih stabil untuk Tkinter)
        from matplotlib.figure import Figure
        from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

        fig = Figure(figsize=(8, 5), dpi=100)
        fig.patch.set_facecolor('#181818')
```

```
        ax = fig.add_subplot(111)
        ax.set_facecolor('#101010')

        # 3. Plot Data
        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)')

        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 (Hanya muncul jika grafiknya untuk hari ini)
        now = datetime.datetime.now()
        if y == now.year and m == now.month and d == now.day:
            curr_h = now.hour + now.minute/60.0
            ax.axvline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
            batas_bawah = min(sun_alts) if sun_alts else 0
            ax.text(curr_h + 0.2, batas_bawah, 'Waktu Sekarang', color='#00E676', fontsize=10,
rotation=90, verticalalignment='bottom')

        # Format Nama Bulan dan Tahun (Aman dari Limitasi Python Datetime)
        tahun_str = f"{y} M" if y > 0 else f"{abs(y-1)} SM"
        nama_bulan = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov",
"Des"]
        bulan_str = nama_bulan[m] if 1 <= m <= 12 else str(m)

        ax.set_title(f"Grafik Ketinggian Benda Langit: {d:02d} {bulan_str} {tahun_str}", color='white',
fontsize=14, fontweight='bold', pad=15)
        ax.set_xlabel("Waktu Lokal (Jam)", color='#ffffff', fontsize=12)
        ax.set_ylabel("Ketinggian / Altitude (°)", color='#ffffff', fontsize=12)

        ax.tick_params(colors='#ffffff', labelsize=11)
        ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)
        ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='upper
right')
        fig.tight_layout()

        # 4. Render ke Layar GUI Tkinter
        canvas = FigureCanvasTkAgg(fig, master=canvas_frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)

        # Update Status Sukses
        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 = [
            'AIzaSyBAPn3uZO2P-7j5KugUFRA83Z61nIjNGX4',
'AIzaSyDI9NDqgzd3XdROOkcK9nmuxGAzscsNoJU',
            'AIzaSyBfW_dBySVHLo74-f0Tvg2grTi3K3K3U1Q',
'AIzaSyCS_HjbzrEN5ddevYNjSvbv0K4oONBOHuY',
            'AIzaSyBnC4vNv-43Kwz3XMnYLKSHCdxUdVgU35M',
'AIzaSyA7Tyz5OAN95sgUbUCOkOGvUKwzijYIrY4',
            'AIzaSyBR3VtwvfDNLzi7s1YvZPGBb0yEOIRReLk', 'AIzaSyBabaqecoW-
bVXD8AwVKc4roEMTZiyrsTM',
            'AIzaSyARUKFKy8dUsiRJRlI4kV2OyiAnTeN6MSM', 'AIzaSyCChatfstuiHaiTxR5SmyHq7PSTMjcHoRg',
            'AIzaSyAFZCjVd9ehgfmFu5mY1pAVnm_EtqCKD20',
'AIzaSyD5e_bbZ9gPE2uafYnWLDXc3hzET5zVI3k',
            'AIzaSyAt5nF52Il_8nQnmfUgihjVUjONd6BxcMQ',
'AIzaSyDnvTKjTAf8HjcmM1ExupBqT7uPeibpmbg',
            'AIzaSyC5_ye8d4SZD4GOHe2aDPyrmqRbk7uwGHQ',
'AIzaSyBJerwj4clbNjFuFs6Q_QYoMZnkOEerNcI',
            'AIzaSyBMh0M2eKXWLHHdZQatkhuUkngblgVqbUI', 'AIzaSyA-
DAdY0EXs1gvc7wcqLCdswxLCJIj2_W8',
            'AIzaSyDylXHRTDYk0xtmYCSRQGmCnIce6v8DI10',
'AIzaSyDamciIrHuZo7EXpNgUKu3Fvd3_EXmNhxU',
            'AIzaSyBn0HuPpJCoX4Rr1tNpGVyFdirG1ep80g4', 'AIzaSyBMXyRo-
dVVbKAs_daQIwyOpq6VkJKos7Y',
            'AIzaSyAcfGW4K0J8HTUba_f4rzPp6IrDaYpPLHY', 'AIzaSyAfMPp3Wfu-2Ksq_iCGxkS1XO6V_IxaO6Y',
            'AIzaSyARhHCJ6vfh_fKI4yza40uxhezsFpF8TAA', 'AIzaSyAH5y4SnbKfy6jFSnCgUmyr7DsP4HXwV18',
            'AIzaSyBVYaHGaQOpshl5E5spFRcCIA4x2stzN8I', 'AIzaSyB2dQEsl8pFUl2FlfU2s6Xty2zl1D0IHHE',
            'AIzaSyCHfn_Y2DzFRhSjWNnPUj5s5Msz4QNp6SY',
'AIzaSyCDQLyQQxs59xKEBfngQRxIa6iFQbf0R44',
            'AIzaSyCyVKZoUvfRpashCoAXO8iLr5HAfBesGyY', 'AIzaSyBjbJVno098Ruq-41zlX3hNVsEtO5SQ-_0',
            'AIzaSyBIw9L26U9BP_RA7wBYSM0liO81acfCZkU',
'AIzaSyAFWmbuloYzfUdaWZT9329HA1T1Q8PWh44',
            'AIzaSyBGRz7pausR71hjj_WyVh2FxsqVZedj3dc'
        ]
        self.winai_primary_model = 'gemini-2.5-flash'
        self.winai_tracker_file = 'last_successful_index_winai.txt'
        self.winai_db_items = []
```

```
        self.winai_db_loaded = False

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

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

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

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

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

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

        self.winai_input_entry = ctk.CTkEntry(winai_info, placeholder_text="Ketik pertanyaan...",
font=("Segoe UI", 13), height=40)
        self.winai_input_entry.pack(fill="x", padx=10, pady=10)
        self.winai_input_entry.bind("<Return>", lambda e: self.proses_pertanyaan_winai())

        self.btn_winai_kirim = ctk.CTkButton(winai_info, text="Kirim Pertanyaan", font=("Segoe UI", 12,
"bold"), fg_color="#1565C0", hover_color="#0D47A1", height=35,
command=self.proses_pertanyaan_winai)
        self.btn_winai_kirim.pack(fill="x", padx=10, pady=(0, 5))

        self.btn_winai_reset = ctk.CTkButton(winai_info, text="🔄 Reset Memori", font=("Segoe UI", 12,
"bold"), fg_color="#dc3545", hover_color="#c82333", height=35, command=self.reset_memori_winai)
        self.btn_winai_reset.pack(fill="x", padx=10, pady=(5, 10))

        # ================= MAIN FRAME OUTPUT (KANAN) =================
        self.frame_winai_out = ctk.CTkFrame(self.main_frame, fg_color="#050510", corner_radius=10)

        header_winai = ctk.CTkFrame(self.frame_winai_out, fg_color="#181818", corner_radius=8)
        header_winai.pack(fill="x", padx=15, pady=10)
```

```
    ctk.CTkLabel(header_winai, text="🤖 Ruang Interaksi AI", font=("Segoe UI", 16, "bold"),
text_color="#00E5FF").pack(side="left", padx=15, pady=10)

    self.btn_winai_salin = ctk.CTkButton(header_winai, text="📋 Salin", fg_color="#28a745",
hover_color="#218838", width=80, command=self.salin_teks_winai)
    self.btn_winai_salin.pack(side="right", padx=(5, 15), pady=10)
    self.btn_winai_berbagi = ctk.CTkButton(header_winai, text="🔗 Share", fg_color="#17a2b8",
hover_color="#138496", width=80, command=self.berbagi_teks_winai)
    self.btn_winai_berbagi.pack(side="right", padx=5, pady=10)
    self.btn_winai_buka = ctk.CTkButton(header_winai, text="📄 Log Teks", fg_color="#6c757d",
hover_color="#5a6268", width=80, command=self.buka_output_winai)
    self.btn_winai_buka.pack(side="right", padx=5, pady=10)

    # Ruang Chat - Background Putih & Teks Hitam Standar Dokumen
    self.winai_output_box = ctk.CTkTextbox(self.frame_winai_out, wrap="word", fg_color="#FFFFFF",
text_color="#000000", font=("Segoe UI", 16))
    self.winai_output_box.pack(fill="both", expand=True, padx=15, pady=(0, 15))

    font_nama = "Segoe UI"
    ukuran_normal = 16

    tb = self.winai_output_box._textbox
    tb.tag_configure("info", foreground="#856404", background="#FFF3CD", font=(font_nama, 14,
"italic"), spacing1=5, spacing3=10, justify="center")
    tb.tag_configure("info_internet", foreground="#0056b3", background="#e7f1ff", font=(font_nama,
14), lmargin1=15, lmargin2=15, spacing1=5, spacing3=10, borderwidth=1, relief="solid")
    tb.tag_configure("pembuka", foreground="#000000", font=(font_nama, 18, "bold"), spacing1=0,
spacing2=0, spacing3=5, justify="left")
    tb.tag_configure("user_header", foreground="#0056b3", font=(font_nama, 16, "bold"),
spacing1=15)
    tb.tag_configure("user_teks", background="#E3F2FD", foreground="#000000", font=(font_nama,
ukuran_normal), lmargin1=10, lmargin2=10, spacing1=5, spacing3=10)
    tb.tag_configure("ai_header", foreground="#28A745", font=(font_nama, 17, "bold"), spacing1=15)
    tb.tag_configure("normal", font=(font_nama, ukuran_normal), foreground="#000000",
justify="left")
    tb.tag_configure("bold", font=(font_nama, ukuran_normal, "bold"), foreground="#000000",
justify="left")
    tb.tag_configure("paragraf", spacing1=5, spacing3=10, lmargin1=10, lmargin2=10, justify="left")
    tb.tag_configure("kutipan", font=(font_nama, ukuran_normal, "italic"), lmargin1=30, lmargin2=30,
foreground="#333333", background="#F8F9FA")
    tb.tag_configure("kode", font=("Consolas", 14), foreground="#000000", background="#E8F5E9",
spacing1=2, spacing3=2, lmargin1=15, lmargin2=15)

    pesan_pembuka = (
        "Selamat datang di WinAI: Aplikasi AI Windows.\n"
        "Silakan ubah tanggal di panel kiri atau ketik pertanyaan Anda...\n"
```

```
        "[!] Anda dapat meminta AI untuk menghitung Visibilitas Hilal atau mencari Kota Pertama KHGT sesuai tanggal yang Anda set.\n\n"
    )
    self.winai_output_box.insert("0.0", pesan_pembuka, "pembuka")
    self.winai_output_box.configure(state="disabled")

  def load_winai_databases(self):
    try:
      base_url = "https://hisabmu.com/aifikih/db_fikih_tarjih"
      res = requests.get(f"{base_url}.json", timeout=10)
      if res.status_code == 200: self.winai_db_items.extend(res.json())
      for i in range(2, 21):
        try:
          res = requests.get(f"{base_url}_{i}.json", timeout=5)
          if res.status_code == 200: self.winai_db_items.extend(res.json())
          else: break
        except: break
      res_txt = requests.get("https://hisabmu.com/aifikih/tanyajawabagama.txt", timeout=10)
      if res_txt.status_code == 200: self.winai_db_items.extend(res_txt.json())
      self.winai_db_loaded = True
    except:
      pass

  def retrieve_winai_context(self, user_question):
    if not self.winai_db_loaded: return "Database belum siap."
    clean_text = re.sub(r'[^a-zA-Z0-9 ]', '', user_question.lower())
    words = clean_text.split()
    keywords = [w for w in words if len(w) > 3]

    scored_items = []
    for item in getattr(self, 'winai_db_items', []):
      judul = item.get('judul', '')
      isi = item.get('isi_konten', '')
      text = f"Q: {judul}\nA: {isi}\n\n"
      text_lower = text.lower()
      score = sum(1 for kw in keywords if kw in text_lower)
      if score > 0: scored_items.append({'score': score, 'text': text})

    scored_items.sort(key=lambda x: x['score'], reverse=True)
    internal_db_text = ""
    for si in scored_items:
      if len(internal_db_text) + len(si['text']) > 80000: break
      internal_db_text += si['text']
    return internal_db_text if internal_db_text.strip() else "Tidak ada referensi lokal yang relevan."

  def sisipkan_teks_winai(self, teks, pengirim="AI"):
```

```
        self.winai_output_box.configure(state="normal")
        if pengirim == "Anda":
            self.winai_pertanyaan_terakhir = teks
            self.winai_output_box.insert("end", "🗣️ Anda:\n", "user_header")
            self.winai_output_box.insert("end", f"  {teks}  \n\n", "user_teks")
        elif pengirim == "Sistem":
            self.winai_output_box.insert("end", " ⏳ " + teks + "\n\n", ("info", "status_loading"))
        else:
            self.winai_output_box.insert("end", "🤖 WinAI:\n", "ai_header")
            self.winai_jawaban_terakhir = teks

            try:
                with open("output_winai.txt", "a", encoding="utf-8") as file_out:
                    waktu_sekarang = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    file_out.write(f"=== TANGGAL: {waktu_sekarang} ===\nPERTANYAAN:
{self.winai_pertanyaan_terakhir}\nJAWABAN:\n{teks}\n{'='*50}\n\n")
            except Exception: pass

            # --- PEMBERSIHAN DAN FORMAT DOKUMEN STANDAR ---
            text_cleaned = re.sub(r'(?i)Asisten Fikih ini.*', '', teks, flags=re.DOTALL).strip()

            lines = text_cleaned.split('\n')
            for i, line in enumerate(lines):
                # Format khusus blok kode (kalkulasi)
                if line.startswith("   ") and "|" in line or "===" in line or "---" in line:
                    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, "bold")
                else:
                    # Deteksi teks tebal (**)
                    parts = line.split('**')
                    for j, part in enumerate(parts):
                        if j % 2 == 1:
                            self.winai_output_box.insert("end", part, "bold")
                        else:
                            # Hapus simbol aneh, biarkan teks normal
                            part_clean = part.replace('*', '')
                            self.winai_output_box.insert("end", part_clean, "normal")

                # Baris baru
                if i < len(lines) - 1:
                    self.winai_output_box.insert("end", "\n", "normal")

            self.winai_output_box.insert("end", "\n\n" + "="*80 + "\n\n", "normal")
```

```
        self.winai_output_box.see("end")
        self.winai_output_box.configure(state="disabled")

    def tampilkan_jejak_internet(self, teks):
        self.winai_output_box.configure(state="normal")
        self.winai_output_box.insert("end", "🌐 BERHASIL MENDAPATKAN DATA INTERNET
TERBARU:\n\n" + teks + "\n", "info_internet")
        self.winai_output_box.see("end")
        self.winai_output_box.configure(state="disabled")

    def hapus_status_loading_winai(self):
        self.winai_output_box.configure(state="normal")
        ranges = self.winai_output_box.tag_ranges("status_loading")
        if ranges:
            self.winai_output_box.delete(ranges[0], ranges[-1])
        self.winai_output_box.configure(state="disabled")

    def proses_pertanyaan_winai(self):
        pertanyaan = self.winai_input_entry.get().strip()
        if not pertanyaan: return

        self.winai_input_entry.delete(0, "end")
        self.sisipkan_teks_winai(pertanyaan, pengirim="Anda")

        teks_lower = pertanyaan.lower()

        # ---> SMART INTERCEPTOR <---
        # AI hanya menghitung jika ada perintah hitung secara eksplisit
        is_tanya_hilal = any(kata in teks_lower for kata in ["hitung", "kalkulasi", "berapa", "cek"]) and
("visibilitas" in teks_lower or "hilal" in teks_lower)
        is_tanya_kota = any(kata in teks_lower for kata in ["kota pertama", "titik pertama", "awal bulan
khgt", "kapan 1", "kapan awal"])

        if is_tanya_hilal:
            self.sisipkan_teks_winai("Menjalankan Modul 1 (Kalkulasi Visibilitas) & Menganalisis Jawaban...",
pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "hilal"),
daemon=True).start()
        elif is_tanya_kota:
            self.sisipkan_teks_winai("Menjalankan Modul 5 (Pelacakan Kota Pertama) & Menganalisis
Jawaban...", pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "kota"),
daemon=True).start()
```

```
        else:
            self.sisipkan_teks_winai("Mengakses Database & Menghubungi AI...", pengirim="Sistem")
            self.btn_winai_kirim.configure(state="disabled")
            threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()

    def hitung_dan_tanya_ai(self, pertanyaan, mode):
        """Fungsi yang memproses hitungan terlebih dahulu, lalu mengirim datanya ke AI untuk diinterpretasi"""
        report_data = ""
        if mode == "hilal":
            try:
                y, m, d = self.winai_entry_y.get(), self.winai_entry_m.get(), self.winai_entry_d.get()
                # Sinkronkan dengan menu 1
                self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, y)
                self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, m)
                self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, d)
                report_data = self.generate_visibility_report()
            except Exception as e:
                report_data = f"Gagal menghitung visibilitas: {e}"
        elif mode == "kota":
            try:
                y, m, d = int(self.winai_entry_y.get()), int(self.winai_entry_m.get()), int(self.winai_entry_d.get())
                report_data = self.generate_first_point_report(y, m, d)
            except Exception as e:
                report_data = f"Gagal melacak kota pertama: {e}"

        # Teruskan ke AI dengan membawa data hasil hitungan
        self.tanya_ai_thread(pertanyaan, report_tambahan=report_data)

    def tanya_ai_thread(self, teks_pertanyaan, report_tambahan=None):
        self.after(0, lambda: self.sisipkan_teks_winai("Membaca instruksi & Database Internal...", pengirim="Sistem"))

        try:
            headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"}
            resp_sys = requests.get("https://hisabmu.com/aifikih/system_prompt.php", headers=headers, timeout=15)
            resp_sys.encoding = 'utf-8'
            if resp_sys.status_code == 200 and "$systemPersona" not in resp_sys.text:
                self.winai_instruksi = resp_sys.text.strip()
            else:
                self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447 H."
        except:
            self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447 H."
```

```
    if not self.winai_data_kal:
        try:
            resp_cal = requests.get(CALENDAR_DATA_URL, headers=headers, timeout=10)
            if resp_cal.status_code == 200: self.winai_data_kal = resp_cal.text.strip()
        except: pass

    teks_bersih = teks_pertanyaan.strip()
    gabungan_internet = ""

    if not teks_bersih.startswith("/"):
        self.after(0, lambda: self.sisipkan_teks_winai("Menelusuri informasi pendukung di internet...",
pengirim="Sistem"))
        hasil_google = ""
        try:
            kata_kunci = urllib.parse.quote(teks_bersih)
            url_g =
f"https://www.google.com/search?&lr=lang_id&hl=id&nem=143&udm=50&q={kata_kunci}"
            resp_g = requests.get(url_g, headers=headers, cookies=GOOGLE_SESSION_COOKIES,
timeout=15)
            if resp_g.status_code == 200:
                soup = BeautifulSoup(resp_g.text, 'html.parser')
                snippets = soup.find_all(['div', 'span'], class_=['BNeawe', 'VwiC3b', 's3v9rd', 'HwtZe'])
                kumpulan = []
                for s in snippets:
                    t = s.get_text(strip=True)
                    if t and len(t) > 60 and t not in kumpulan: kumpulan.append(t)
                if kumpulan: hasil_google = "[Data Referensi Google Search]:\n" + "\n".join(f"- {txt}" for txt in
kumpulan[:3]) + "\n"
        except: pass

        hasil_ddg = ""
        try:
            with DDGS() as ddgs:
                results = list(ddgs.text(teks_bersih, max_results=2))
                if results: hasil_ddg = "[Data Referensi DuckDuckGo]:\n" + "\n".join(f"- {r['body']}" for r in
results) + "\n"
        except: pass

        if hasil_google or hasil_ddg:
            gabungan_internet = f"{hasil_google}\n{hasil_ddg}".strip()
            self.after(0, self.tampilkan_jejak_internet, gabungan_internet)

    # RAG / Local DB
    konteks_lokal = self.retrieve_winai_context(teks_bersih)
```

```
        self.after(0, lambda: self.sisipkan_teks_winai("Menganalisis pertanyaan dan memanggil Model AI Gemini...", pengirim="Sistem"))

        loc_prov = self.opt_prov.get()
        loc_city = self.opt_city.get()
        win_y = self.winai_entry_y.get()
        win_m = self.winai_entry_m.get()
        win_d = self.winai_entry_d.get()

        prompt_pintar = f"=== KONTEKS PENGGUNA ===\nLokasi: {loc_city}, {loc_prov}.\nTanggal Tinjauan: {win_d}/{win_m}/{win_y}.\nWaktu Aktual: {datetime.datetime.now().strftime('%A, %d %B %Y %H:%M:%S')}\n\n"
        if self.winai_data_kal: prompt_pintar += f"=== DATA KALENDER KHGT ===\n{self.winai_data_kal}\n\n"
        if gabungan_internet: prompt_pintar += f"=== REFERENSI INTERNET ===\n{gabungan_internet}\n\n"
        if self.winai_memori: prompt_pintar += f"=== RIWAYAT OBROLAN LALU ===\n{self.winai_memori}\n\n"

        # Injeksi Data Report Sistem (Bila Ada Perintah Hitung)
        if report_tambahan:
            prompt_pintar += f"=== HASIL KALKULASI SISTEM SAAT INI ===\n{report_tambahan}\n\n"

        prompt_pintar += f"=== PERTANYAAN USER ===\n{teks_bersih}\n\n"

        # Aturan menjawab dirancang agar natural
        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 WAJIB menjelaskan hasilnya secara ringkas di awal, LALU melampirkan teks/tabel aslinya secara utuh di bagian akhir jawaban Anda."

        final_instruction = self.winai_instruksi + f"\n\n====================\nREFERENSI DATABASE INTERNAL LOKAL:\n\"\"\"{konteks_lokal}\"\"\"\n====================\n{aturan}"

        payload = {
            "systemInstruction": {"parts": [{"text": final_instruction}]},
            "contents": [{"role": "user", "parts": [{"text": prompt_pintar}]}],
            "generationConfig": {"temperature": 0.2, "topP": 0.8}
        }

        start_index = 0
        if os.path.exists(self.winai_tracker_file):
            with open(self.winai_tracker_file, 'r') as f:
                start_index = int(f.read().strip() or 0)
```

```
        total_keys = len(self.winai_api_keys)
        jawaban = ""
        sukses_dijawab = False
        last_error = ""

        for i in range(total_keys):
            idx = (start_index + i) % total_keys
            api_key = self.winai_api_keys[idx]
            api_url =
f"https://generativelanguage.googleapis.com/v1beta/models/{self.winai_primary_model}:generateCont
ent?key={api_key}"

            try:
                res = requests.post(api_url, headers={'Content-Type': 'application/json'}, json=payload,
timeout=20)
                if res.status_code == 200:
                    data = res.json()
                    jawaban = data['candidates'][0]['content']['parts'][0]['text']
                    with open(self.winai_tracker_file, 'w') as f:
                        f.write(str(idx))
                    sukses_dijawab = True
                    break
                else:
                    err = res.json()
                    last_error = err.get('error', {}).get('message', f'HTTP {res.status_code}')
                    if res.status_code in [400, 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}"

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

            if len(self.winai_memori) > 15000:
                self.winai_memori = self.winai_memori[-15000:]

        self.after(0, self.selesai_memproses_winai, jawaban)

    def selesai_memproses_winai(self, jawaban):
```

```
        self.hapus_status_loading_winai()
        self.sisipkan_teks_winai(jawaban, pengirim="AI")
        self.btn_winai_kirim.configure(state="normal")

    def salin_teks_winai(self):
        teks_salin = self.winai_output_box.get("1.0", "end-1c").strip()
        if not teks_salin:
            messagebox.showwarning("Peringatan", "Tidak ada teks untuk disalin.")
            return

        self.clipboard_clear()
        self.clipboard_append(teks_salin)
        messagebox.showinfo("Sukses", "Seluruh teks di layar berhasil disalin ke Clipboard!")

    def berbagi_teks_winai(self):
        teks_share = self.winai_output_box.get("1.0", "end-1c").strip()
        if not teks_share: return

        self.btn_winai_berbagi.configure(text=" ⏳ Memproses...", state="disabled")
        threading.Thread(target=self.proses_berbagi_thread, args=(teks_share,), daemon=True).start()

    def proses_berbagi_thread(self, teks_share):
        payload = {'action': 'share_content', 'text': teks_share}
        try:
            # Karena PHP Server WinAI tetap butuh endpoint untuk berbagi URL
            response = requests.post("https://hisabmu.com/aifikih/system_php.php", data=payload,
timeout=15)
            if response.status_code == 200:
                data = response.json()
                if data.get('status') == 'success':
                    self.after(0, self.berbagi_sukses, data.get('url'))
                else:
                    self.after(0, self.berbagi_gagal, data.get('message', 'Gagal membuat link.'))
            else:
                self.after(0, self.berbagi_gagal, "Error koneksi.")
        except Exception as e:
            self.after(0, self.berbagi_gagal, str(e))

    def berbagi_sukses(self, url):
        self.btn_winai_berbagi.configure(text=" 🔗 Share", state="normal")
        webbrowser.open(url)
        self.clipboard_clear()
        self.clipboard_append(url)
        messagebox.showinfo("Sukses", f"Link berhasil dibuat!\n\n{url}")

    def berbagi_gagal(self, pesan_error):
```

```
        self.btn_winai_berbagi.configure(text="🔗 Share", state="normal")
        messagebox.showerror("Gagal", f"Gagal membagikan teks:\n{pesan_error}")

    def buka_output_winai(self):
        nama_file = "output_winai.txt"
        if not os.path.exists(nama_file):
            with open(nama_file, "w", encoding="utf-8") as f:
                f.write("=== LOG OUTPUT WinAI ===\n\n")
        try: os.startfile(nama_file)
        except Exception as e: messagebox.showerror("Error", f"Gagal membuka:\n{e}")

    def reset_memori_winai(self):
        self.winai_memori = ""
        messagebox.showinfo("Reset Berhasil", "Ingatan dikosongkan. AI siap!")

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

        try:
            ijtimak_1 = ephem.previous_new_moon(waktu_tuple)
            ijtimak_2 = ephem.next_new_moon(waktu_tuple)
        except Exception:
            return "Format tanggal tidak valid. Harap periksa input."

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

        for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "Bulan Berjalan/Depan")]:
            for offset_hari in range(4):
                kandidat_hari_ini = []
                waktu_pencarian = ephem.Date(w_ijtimak + offset_hari)

                for negara, kota_dict in CITY_DB.items():
                    if negara in zona_dikecualikan: continue
                    for nama_kota, koordinat in kota_dict.items():
                        lintang, bujur = koordinat
                        pengamat = ephem.Observer()
                        pengamat.lat, pengamat.lon = str(lintang), str(bujur)
                        pengamat.elevation = 0
                        pengamat.date = waktu_pencarian

                        try: waktu_sunset = pengamat.next_setting(matahari)
```

```
                except: continue

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

                HA_moon = pengamat.sidereal_time() - bulan.g_ra
                sin_alt_geo = math.sin(pengamat.lat) * math.sin(bulan.g_dec) + math.cos(pengamat.lat) *
math.cos(bulan.g_dec) * math.cos(HA_moon)
                alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))

                cos_elong_geo = math.sin(matahari.g_dec) * math.sin(bulan.g_dec) +
math.cos(matahari.g_dec) * math.cos(bulan.g_dec) * math.cos(matahari.g_ra - bulan.g_ra)
                elong_geo = math.degrees(math.acos(max(-1.0, min(1.0, cos_elong_geo))))

                if alt_geo >= 5.0 and elong_geo >= 8.0:
                    kandidat_hari_ini.append({
                        'kota': nama_kota, 'negara': negara, 'lat': lintang, 'lon': bujur,
                        'sunset_utc': waktu_sunset, 'alt_geo': alt_geo, 'elong_geo': elong_geo,
                        'ijtimak': w_ijtimak, 'label': label_siklus
                    })

            if kandidat_hari_ini:
                kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
                titik_hasil.append(kandidat_hari_ini[0])
                break

        if not titik_hasil:
            return "Tidak ada daratan utama (Mainland) yang memenuhi kriteria KHGT pada 2 siklus ini."

        report = "Pencarian Multi-Siklus Selesai.\nTitik daratan utama pertama di dunia yang masuk bulan
baru KHGT:\n\n"

        for pt in titik_hasil:
            tz_approx = int(self.get_tz_from_lon(pt['lon']))
            tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"

            dt_local_ephem = ephem.Date(pt['sunset_utc'] + (tz_approx / 24.0))
            y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()
            nama_bulan = ["", "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}"

        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 (Titik 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 Tidak Terpenuhi ({alasan})\n» Kesimpulan: LUSA masuk
awal bulan"
            except:
                fajar_info = " Fajar Gisb.  : Error/N/A\n"
                status_ijtimak = "Kriteria PKG 2 Tidak 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"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
            f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
            f"{'-'*45}\n"
            f"Parameter Waktu (UTC):\n"
            f" Waktu Maghrib: {ss_jam} UTC\n"
            f" Batas PKG 1  : {bt_jam} UTC\n"
            f" Ijtimak      : {ij_jam} UTC\n"
            f"{fajar_info}"
            f"{'-'*45}\n"
            f"Status: {status_ijtimak}\n"
            f"{'='*45}\n\n"
        )
    return report.strip()

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 Kongres Internasional Penyatuan Kalender Hijriah di Istanbul 2016*. (Rujukan utama syarat KHGT: Alt ≥ 5° dan Elongasi ≥ 8°).
- Majelis Tarjih dan Tajdid PP Muhammadiyah. (n.d.). *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 Astronomi Fundamental

Referensi yang mendasari formula pergerakan 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* seperti `de421.bsp`, `de406.bsp`).
- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell. (Rujukan algoritma gerhana, 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 Fikih AI Terintegrasi).

GLOSARIUM A–Z: KHGT TIMES V7.0

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 (Ultra-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

- **Declination (Deklinasi):** Salah satu dari dua koordinat ekuatorial. Mengukur jarak sudut benda langit di sebelah utara atau selatan ekuator langit (setara dengan Lintang pada peta bumi).
- **Development Ephemeris (DE):** Model numerik terkomputerisasi dari tata surya yang diterbitkan oleh NASA Jet Propulsion Laboratory (JPL), seperti file `de421.bsp` atau `de406.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 Penanggalan 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 HD Map*) 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 persilangan (irisan) geografis pada peta visibilitas di mana kedua syarat visibilitas (Altitude $\geq 5°$ dan Elongasi $\geq 8°$) terpenuhi secara bersamaan.

J

- **Julian Date (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 kontras langit dan topografi, digunakan sebagai opsi pembanding di dalam program.

P

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

Q

- **Qiblah (Kiblat):** Arah menuju Ka'bah di Makkah. Modul program menggunakan rumus trigonometri bola (*Spherical Trigonometry*) untuk menghitung azimut pasti dari titik 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 V7.0 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 elipsoida Bumi yang digunakan program untuk mengkalkulasi jarak dan presisi pengamat secara akurat.
- **WinAI:** Modul terintegrasi di dalam KHGT Times V7.0 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

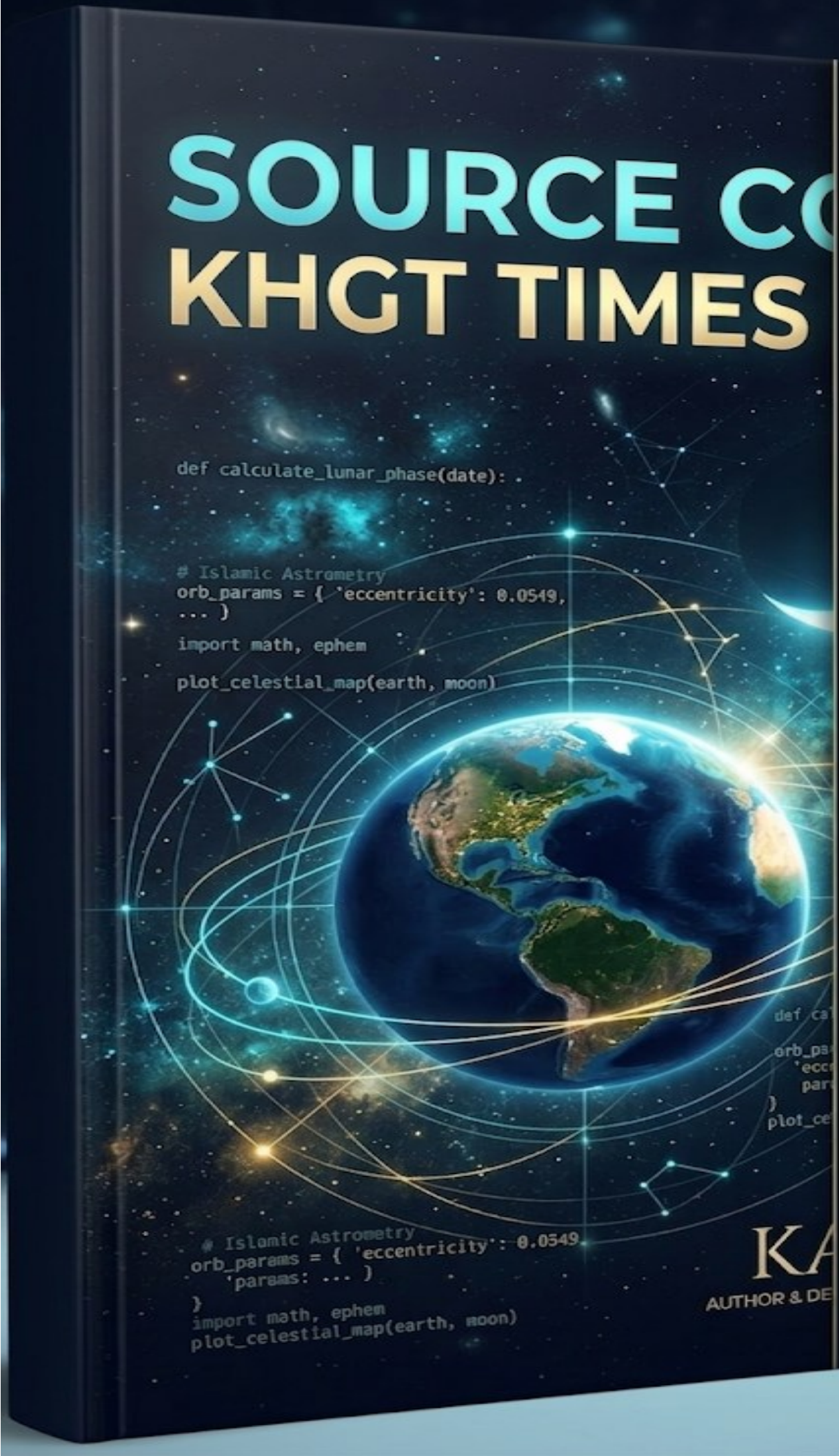
- **Y-Axis (Sumbu Y):** Sumbu vertikal pada antarmuka *Altitude Chart Analyser* Matplotlib yang difungsikan untuk merepresentasikan nilai derajat Ketinggian (Altitude) objek.

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 V7.0, perangkat lunak astrometri presisi tinggi generasi terbaru.

Buku ini adalah *Quantum Leap* komputasi astronomi Islam, 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 hilal HD dan lingkungan simulasi 3D interaktif.

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

KHGT PROJECT