Modul: Automasi pengolahan database pengeboran

Developed by: Hamba ALLAH

For further colaborations, feel free to reach me out with Hyperlink above

Sekapur Sirih

Modul ini diharapkan dapat membantu dalam pembuatan automasi database dengan meng-update data hariian secara automatis (contoh kasus dalam modul ini menggunakan data pengeboran yang akan terus diupdate berdasarkan data titik bor yang finish). Semoga bermanfaat dan membawa kebaikan.

Panduan Step by Steps

Langkah 1: Menyiapkan Environment Tools

1. **Install Python:**
   - Unduh dan install Python 3.11 atau lebih baru dari situs resmi Python.
   - Saat instalasi, pastikan opsi "Tambahkan Python ke PATH" dicentang untuk akses baris perintah.
   - Verifikasi instalasi dengan membuka Command Prompt (Windows) atau Terminal (Mac/Linux) dan ketik `python --version`.
2. **Buat Folder Proyek:**
   - Buat folder bernama `DrillHole_Automation` di komputer Anda (misalnya, `C:\Users\NamaPenggunaAnda\DrillHole_Automation`).
   - Di dalamnya, buat subfolder `Daily_Data` untuk file data harian yang akan ditambahkan ke database.
3. **Install Library yang Diperlukan:**
   - Buka Command Prompt, navigasikan ke `DrillHole_Automation` (misalnya, `cd C:\Users\NamaPenggunaAnda\DrillHole_Automation`), dan jalankan perintah dibawah:

     ```
     pip install flask watchdog openpyxl pandas matplotlib
     ```

   - Perintah diatas menginstal library Flask (server web), Watchdog (pemantauan file), OpenPyXL (penanganan Excel), Pandas (manipulasi data), dan Matplotlib (visualisasi).
4. **Siapkan File database Anda:**
   - Siapkan format database, e.g: `drilling_database.xlsx` dalam folder `DrillHole_Automation` dengan format disesuaikan dengan format database yang dibutuhkan (misalnya, `Date Logging`, `Hole ID`, `From`, dll.).
   - Pastikan file dapat diedit dan tidak dibuka di Excel selama eksekusi skrip.
5. **Siapkan File Data Harian:**

- Letakkan file Excel harian sampel (misalnya, `daily-data.xlsx`) di `Daily_Data`. File ini harus mengikuti format konsisten dengan metadata (ID Lubang di B2, Tanggal di J3) dan data dimulai pada baris 9.

Langkah 2: Buat Skrip Utama (`data_updater.py`)

Buat file bernama `data_updater.py` di `DrillHole_Automation`. Script kode pada bagian ini akan kami bagi menjadi beberapa bagian yang akan membantu memudahkan dalam memahami format kode.

Bagian 1: Impor dan Konfigurasi

Tujuan: Menyiapkan library yang diperlukan.

```
import matplotlib
matplotlib.use('Agg')  # Gunakan backend non-GUI untuk Matplotlib untuk
menghindari masalah threading

import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
from flask import Flask, request, render_template_string
import openpyxl
import threading
import os
import pandas as pd
import logging
from datetime import datetime
import matplotlib.pyplot as plt
from io import BytesIO
import base64

# Siapkan logging untuk debugging dan pelacakan
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

# Konfigurasi
daily_folder = r'Daily_Data'
database_file =
r'C:\Users\NamaPenggunaAnda\DrillHole_Automation\drilling_database.xlsx'
```

- **Penjelasan:**
  - `matplotlib.use('Agg')` mencegah error terkait GUI (misalnya, konflik dengan Tkinter) dengan menggunakan backend non-interaktif.
  - Flask digunakan untuk menampilkan data bebasis dashboard, pemantauan file (Watchdog), manipulasi Excel (OpenPyXL), analisis data (Pandas), dan visualisasi (Matplotlib).
  - Logging melacak operasi untuk debugging.

- `daily_folder` dan `database_file` menentukan jalur file (sesuaikan `NamaPenggunaAnda` dengan nama pengguna Windows Anda atau gunakan `os.path.expanduser` untuk kompatibilitas lintas platform, misalnya, `os.path.expanduser('~') + '/DrillHole_Automation/drilling_database.xlsx'`).

Potensi eror:

- **Masalah penempatan folder:** Jika jalur `daily_folder` atau `database_file` tidak sesuai, Anda akan mendapatkan `FileNotFoundError`. Verifikasi jalur di File Explorer dan sesuaikan.
- **Error Instalasi Library:** Potensi `pip install` gagal, pastikan ada akses internet atau periksa kompatibilitas versi Python.

Bagian 2: Pemantauan Sistem File dengan Watchdog

Tujuan: Mendeteksi folder `Daily_Data` untuk mendeteksi file Excel baru dan melakukan pemrosesan.

```
class Watcher:
    def __init__(self):
        self.observer = Observer()

    def run(self):
        event_handler = Handler()
        self.observer.schedule(event_handler, daily_folder, recursive=False)
        self.observer.start()
        try:
            while True:
                time.sleep(1)
        except KeyboardInterrupt:
            self.observer.stop()
        self.observer.join()

class Handler(FileSystemEventHandler):
    def on_created(self, event):
        if event.is_directory:
            return
        if event.src_path.endswith('.xls') atau event.src_path.endswith('.xlsx'):
            logger.info(f"File baru terdeteksi: {event.src_path}")
            try:
                process_new_file(event.src_path)
                logger.info(f"Berhasil memproses file: {event.src_path}")
            except Exception as e:
                logger.error(f"Error memproses file {event.src_path}: {str(e)}")
```

- **Penjelasan:**
  - `Watcher` membuat `Observer` untuk memantau `daily_folder` secara continue.
  - `Handler` memantau pembuatan file, memfilter file `.xls` atau `.xlsx`, dan menjalankan `process_new_file` untuk mengolah file baru.
  - `time.sleep(1)` mencegah overload CPU; `KeyboardInterrupt` menutup sistem ketika mengalami overload or stuck.

Kemungkinan Eror:

- **Izin Ditolak:** Jika `Daily_Data` tidak memiliki izin baca/tulis, `Watchdog` gagal. Periksa izin folder di Windows Explorer.
- **Penguncian File:** Jika file dibuka di Excel, `Watchdog` mungkin memicu sebelum file sepenuhnya diedit, menyebabkan `PermissionError`. Tambahkan `time.sleep(2)` di `process_new_file` untuk menunda pemrosesan.

Bagian 3: Fungsi Pemrosesan Data

Tujuan: Menganalisis, memindahkan, dan memperbarui data harian ke dalam database.

```
def process_new_file(file_path):
    time.sleep(2)  # Pastikan file sepenuhnya ditulis
    transformed_data = transform_data(file_path)
    append_to_database(transformed_data)

def transform_data(file_path):
    try:
        wb = openpyxl.load_workbook(file_path, data_only=True)
        sheet = wb.active

        raw_hole_id = sheet['B2'].value
        raw_date_logging = sheet['J3'].value
        logger.info(f"Metadata mentah dari {os.path.basename(file_path)} - B2:
{raw_hole_id}, J3: {raw_date_logging}")

        hole_id = str(raw_hole_id).strip() if raw_hole_id else None
        date_logging = raw_date_logging
        if isinstance(date_logging, datetime):
            date_logging = date_logging.date()
        elif isinstance(date_logging, str):
            try:
                date_logging = datetime.strptime(date_logging, '%Y-%m-%d
%H:%M:%S').date()
            except ValueError:
                date_logging = None

        if not hole_id or not date_logging:
            raise ValueError(f"Missing or invalid Hole ID ({hole_id}) or Date
Logging ({date_logging}) in metadata")

        daily_data = pd.read_excel(file_path, skiprows=7, header=0, usecols='B:V')
        logger.info(f"Kolom di {os.path.basename(file_path)}:
{list(daily_data.columns)}")

        # Data kolom disesuaikan dengan format database yang dibuat atau dimiliki.
        required_columns = ['FROM', 'TO', 'INTERVAL (M)', 'ACT CORE (M)',
'RECOVERY (%)',
                            'GENERAL LITHOLOGY', 'SUB GEN LITHOLOGY', 'ROCK CODE',
'GRAIN SIZE',
```

```
                            'WEATHERING', 'COLOUR', 'PRIMARY.1', 'SECONDARY.1',
'TERTIARY']
        missing_columns = [col for col in required_columns if col not in
daily_data.columns]
        if missing_columns:
            logger.warning(f"Kolom yang diharapkan hilang: {missing_columns}.
Kembali ke indeks.")
            daily_data = pd.read_excel(file_path, skiprows=8, header=None,
usecols='B:V')
            transformed_rows = []
            for _, row in daily_data.iterrows():
                if pd.isna(row[0]):
                    continue
                transformed_row = {
                    'Date Logging': date_logging,
                    'Hole ID': hole_id,
                    'From': row[0],
                    'To': row[1],
                    'Length': row[2],
                    'Actual Core': row[3],
                    'Recovery pecentage': row[6] / 100 if pd.notna(row[6]) else
1.0,
                    'Material Code': str(row[7]).lower() if pd.notna(row[7]) else
'',
                    'Layer Code': str(row[8]).lower() if pd.notna(row[8]) else '',
                    'Rock Code': str(row[9]).lower() if pd.notna(row[9]) else '',
                    'Grain': str(row[10]).lower() if pd.notna(row[10]) else '',
                    'Weath': row[12] if pd.notna(row[12]) else None,
                    'Colour': str(row[13]).lower() if pd.notna(row[13]) else '',
                    'Minerals Pri': str(row[16]).lower() if pd.notna(row[16]) else
'',
                    'Minerals Sec': str(row[17]).lower() if pd.notna(row[17]) else
'',
                    'Minerals Ter': str(row[18]).lower() if pd.notna(row[18]) else
'',
                    'Bolder leght (m)': None
                }
                transformed_rows.append(transformed_row)
        else:
            transformed_rows = []
            for _, row in daily_data.iterrows():
                if pd.isna(row['FROM']):
                    continue
                transformed_row = {
                    'Date Logging': date_logging,
                    'Hole ID': hole_id,
                    'From': row['FROM'],
                    'To': row['TO'],
                    'Length': row['INTERVAL (M)'],
                    'Actual Core': row['ACT CORE (M)'],
                    'Recovery pecentage': row['RECOVERY (%)'] / 100 if
pd.notna(row['RECOVERY (%)']) else 1.0,
                    'Material Code': str(row['GENERAL LITHOLOGY']).lower() if
pd.notna(row['GENERAL LITHOLOGY']) else '',
```

```
                    'Layer Code': str(row['SUB GEN LITHOLOGY']).lower() if
pd.notna(row['SUB GEN LITHOLOGY']) else '',
                    'Rock Code': str(row['ROCK CODE']).lower() if
pd.notna(row['ROCK CODE']) else '',
                    'Grain': str(row['GRAIN SIZE']).lower() if pd.notna(row['GRAIN
SIZE']) else '',
                    'Weath': row['WEATHERING'] if pd.notna(row['WEATHERING']) else
None,
                    'Colour': str(row['COLOUR']).lower() if
pd.notna(row['COLOUR']) else '',
                    'Minerals Pri': str(row['PRIMARY.1']).lower() if
pd.notna(row['PRIMARY.1']) else '',
                    'Minerals Sec': str(row['SECONDARY.1']).lower() if
pd.notna(row['SECONDARY.1']) else '',
                    'Minerals Ter': str(row['TERTIARY']).lower() if
pd.notna(row['TERTIARY']) else '',
                    'Bolder leght (m)': None
                }
                transformed_rows.append(transformed_row)
        return transformed_rows
    except Exception as e:
        logger.error(f"Error di transform_data: {str(e)}")
        raise

def append_to_database(transformed_rows):
    try:
        db_path = os.path.normpath(database_file)
        logger.info(f"Mencoba memperbarui database di: {db_path}")

        wb = openpyxl.load_workbook(db_path)
        sheet = wb.active
        last_row = sheet.max_row

        if last_row == 1 and sheet.cell(row=1, column=1).value is None:
            headers = ['Date Logging', 'Hole ID', 'From', 'To', 'Length', 'Actual
Core', 'Recovery pecentage',
                       'Material Code', 'Layer Code', 'Rock Code', 'Grain',
'Weath', 'Colour',
                       'Minerals Pri', 'Minerals Sec', 'Minerals Ter', 'Bolder
leght (m)']
            for col, header in enumerate(headers, 1):
                sheet.cell(row=1, column=col).value = header
            last_row = 1

        for row_data in transformed_rows:
            last_row += 1
            sheet.cell(row=last_row, column=1).value = row_data['Date Logging']
            sheet.cell(row=last_row, column=2).value = row_data['Hole ID']
            sheet.cell(row=last_row, column=3).value = row_data['From']
            sheet.cell(row=last_row, column=4).value = row_data['To']
            sheet.cell(row=last_row, column=5).value = row_data['Length']
            sheet.cell(row=last_row, column=6).value = row_data['Actual Core']
            sheet.cell(row=last_row, column=7).value = row_data['Recovery
pecentage']
```

```
            sheet.cell(row=last_row, column=8).value = row_data['Material Code']
            sheet.cell(row=last_row, column=9).value = row_data['Layer Code']
            sheet.cell(row=last_row, column=10).value = row_data['Rock Code']
            sheet.cell(row=last_row, column=11).value = row_data['Grain']
            sheet.cell(row=last_row, column=12).value = row_data['Weath']
            sheet.cell(row=last_row, column=13).value = row_data['Colour']
            sheet.cell(row=last_row, column=14).value = row_data['Minerals Pri']
            sheet.cell(row=last_row, column=15).value = row_data['Minerals Sec']
            sheet.cell(row=last_row, column=16).value = row_data['Minerals Ter']
            sheet.cell(row=last_row, column=17).value = row_data['Bolder leght
(m)']

        wb.save(db_path)
        logger.info(f"Berhasil menambahkan {len(transformed_rows)} baris ke
{db_path}")
    except Exception as e:
        logger.error(f"Error menambahkan ke database: {str(e)}")
        raise
```

- **Penjelasan:**
  - `process_new_file`: Mengkoordini pengolahan data dan memastikan ketersediaan file.
  - `transform_data`: Membaca metadata (ID Lubang, Tanggal), memvalidasinya, dan mentransformasi data harian ke format database, menangani ketidaksesuaian header dengan indeks sebagai cadangan.
  - `append_to_database`: Memperbarui `drilling_database.xlsx`, menambahkan header jika diperlukan, memastikan konsistensi data.

Kemungkinan Eror:

- **Error Metadata:** Jika cell excel `B2` atau `J3` kosong, gunakan log untuk debugging dan buat ulang file jika rusak.
- **Ketidaksesuaian Kolom:** Jika header berbeda, log akan menunjukkan nama kolom; sesuaikan `transform_data` untuk indeks atau header.
- **Izin File:** Pastikan akses tulis ke `drilling_database.xlsx`, periksa melalui properti Windows Explorer.

Bagian 4: Menampilakan data dengan dashboard sederhana dengan `Flask`

Tujuan: Menyediakan dashboard di internal komputer untuk visualisasi dan pembaruan manual.

```
app = Flask(__name__)

def load_database():
    return pd.read_excel(database_file)

def generate_recovery_plot(df):
    depth_avg = df.groupby('From')['Recovery pecentage'].mean().reset_index()
    plt.figure(figsize=(10, 6))
    plt.plot(depth_avg['From'], depth_avg['Recovery pecentage'], marker='o',
linestyle='-', color='biru')
```

```
    plt.title('Persentase Pemulihan Rata-rata vs. Kedalaman Rata-rata')
    plt.xlabel('Kedalaman Rata-rata (m) - Dari')
    plt.ylabel('Persentase Pemulihan Rata-rata')
    plt.grid(True)
    img = BytesIO()
    plt.savefig(img, format='png', bbox_inches='tight')
    plt.close()
    return base64.b64encode(img.getvalue()).decode('utf-8')

def generate_material_distribution_plot(df):
    material_counts = df['Material Code'].value_counts()
    plt.figure(figsize=(10, 6))
    material_counts.plot(kind='bar', color='ungu')
    plt.title('Distribusi Kode Material')
    plt.xlabel('Kode Material')
    plt.ylabel('Jumlah')
    plt.xticks(rotation=45)
    plt.grid(axis='y')
    img = BytesIO()
    plt.savefig(img, format='png', bbox_inches='tight')
    plt.close()
    return base64.b64encode(img.getvalue()).decode('utf-8')

DASHBOARD_TEMPLATE = """
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard Analitik DrillHole</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 0; background-color:
#f5f7fa; color: #333; }
        .container { max-width: 1200px; margin: 0 auto; padding: 20px; }
        .sidebar { width: 200px; float: left; background-color: #e9ecef; padding:
20px; height: 100vh; position: fixed; }
        .content { margin-left: 220px; }
        h1 { text-align: center; color: #2c3e50; margin-bottom: 20px; }
        .stats-grid { display: grid; grid-template-columns: repeat(4, 1fr); gap:
20px; margin-bottom: 20px; }
        .stat-card { background-color: white; border-radius: 8px; padding: 15px;
text-align: center; box-shadow: 0 2px 4px rgba(0,0,0,0.1); }
        .stat-card h3 { margin: 0 0 5px 0; font-size: 14px; color: #7f8c8d; }
        .stat-card p { margin: 0; font-size: 18px; color: #2c3e50; }
        .plot-section { background-color: white; border-radius: 8px; padding:
20px; margin-bottom: 20px; box-shadow: 0 2px 4px rgba(0,0,0,0.1); }
        .plot { text-align: center; }
        img { max-width: 100%; height: auto; }
    </style>
</head>
<body>
    <div class="sidebar">
        <h2>Navigasi</h2>
        <ul style="list-style-type: none; padding: 0;">
            <li><a href="/" style="color: #2c3e50; text-decoration:
none;">Dashboard</a></li>
```

```
            <li><a href="/update" style="color: #2c3e50; text-decoration:
none;">Perbarui Data</a></li>
            <li><a href="/database-status" style="color: #2c3e50; text-decoration:
none;">Status database</a></li>
        </ul>
    </div>
    <div class="content">
        <div class="container">
            <h1>Dashboard Analitik DrillHole</h1>
            <div class="stats-grid">
                <div class="stat-card">
                    <h3>Total Lubang Pengeboran</h3>
                    <p>{{ total_holes }}</p>
                </div>
                <div class="stat-card">
                    <h3>Kedalaman Rata-rata</h3>
                    <p>{{ avg_depth|round(2) }} m</p>
                </div>
                <div class="stat-card">
                    <h3>Kedalaman Terdalam</h3>
                    <p>{{ deepest_depth|round(2) }} m</p>
                </div>
                <div class="stat-card">
                    <h3>Kedalaman Terpendek</h3>
                    <p>{{ shallowest_depth|round(2) }} m</p>
                </div>
            </div>
            <div class="plot-section">
                <h2>Persentase Pemulihan Rata-rata vs. Kedalaman Rata-rata</h2>
                <div class="plot">
                    {% if recovery_plot %}
                        <img src="{{ recovery_plot }}"
alt="Plot Pemulihan">
                    {% else %}
                        <p>Tidak ada data yang tersedia.</p>
                    {% endif %}
                </div>
            </div>
            <div class="plot-section">
                <h2>Distribusi Kode Material</h2>
                <div class="plot">
                    <img src="{{ material_plot }}"
alt="Distribusi Material">
                </div>
            </div>
        </div>
    </div>
</body>
</html>
"""

@app.route('/', methods=['GET'])
def dashboard():
    try:
```

```
        df = load_database()
        total_holes = len(df['Hole ID'].unique()) if not df.empty else 0
        avg_depth = df['Length'].mean() if not df.empty else 0
        deepest_depth = df['To'].max() if not df.empty else 0
        shallowest_depth = df['From'].min() if not df.empty else 0
        recovery_plot = generate_recovery_plot(df)
        material_plot = generate_material_distribution_plot(df)
        return render_template_string(DASHBOARD_TEMPLATE, total_holes=total_holes,
avg_depth=avg_depth,
                                      deepest_depth=deepest_depth,
shallowest_depth=shallowest_depth,
                                      recovery_plot=recovery_plot,
material_plot=material_plot)
    except Exception as e:
        logger.error(f"Error merender dashboard: {str(e)}")
        return f"Error: {str(e)}", 500

@app.route('/update', methods=['GET', 'POST'])
def trigger_update():
    try:
        processed_files = []
        for file in os.scandir(daily_folder):
            if file.is_file() and (file.name.endswith('.xls') or
file.name.endswith('.xlsx')):
                process_new_file(file.path)
                processed_files.append(file.name)
        return {'status': 'success', 'processed_files': processed_files,
'message': 'Pembaruan manual dipicu'}
    except Exception as e:
        return {'status': 'error', 'message': str(e)}, 500

@app.route('/database-status', methods=['GET'])
def database_status():
    try:
        wb = openpyxl.load_workbook(database_file)
        sheet = wb.active
        return {'status': 'success', 'last_row': sheet.max_row}, 200
    except Exception as e:
        return {'status': 'error', 'message': str(e)}, 500
```

- **Penjelasan:**
  - `load_database`: Memuat database untuk analisis.
  - `generate_recovery_plot`, `generate_material_distribution_plot`: Membuat visualisasi, dideskripsikan sebagai base64 untuk dashboard.
  - `DASHBOARD_TEMPLATE`: HTML dengan CSS untuk menampilkan dashboard, meliputi statistik dan plot.
  - Rute menangani tampilan dashboard, pembaruan manual, dan pemeriksaan status database.

Kemungkinan Eror:

- **Masalah Backend Matplotlib:** Jika `matplotlib.use('Agg')` dilewatkan, expect error threading. Pastikan ada di awal skrip.

- **Error Memuat database:** Jika `drilling_database.xlsx` hilang atau salah format, `pd.read_excel` gagal. Verifikasi keberadaan dan format file.
- **Error Rute Flask:** Pastikan metode HTTP benar; sesuaikan jika terjadi error 405, seperti yang terlihat dengan `/update`.

Bagian 5: Menjalankan Sistem

Tujuan: Jalankan skrip untuk memantau dan memperbarui data secara terus-menerus.

```
if __name__ == '__main__':
    watcher_thread = threading.Thread(target=lambda: Watcher().run())
    watcher_thread.daemon = True
    watcher_thread.start()
    app.run(host='127.0.0.1', port=5000)
```

- **Penjelasan:**
  - Menjalankan `Watcher` untuk pemantauan file pada folder yang telah ditentukan sebelumnya.
  - Meluncurkan Flask di `localhost:5000`, dapat diakses melalui browser untuk dashboard.

Kemungkinan Eror:

- **Konflik Port:** Jika anda sedang menjalankan 5000 sebagai lokal host untuk projek lain maka Flask tidak dapat mengunakan host tersebut. Ubah ke `app.run(host='127.0.0.1', port=5001)` atau hapus poject yang bersangkutan.
- **Masalah Threading:** Jika `Watcher` atau Flask crash, periksa log untuk pengecualian, memastikan izin dan akses file yang tepat.

Langkah 3: Ujicoba dan Penanganan Masalah

1. **Jalankan Skrip:**
   - Buka Command Prompt di `DrillHole_Automation`, ketik `python data_updater.py`, dan tekan Enter.
   - Periksa log untuk konfirmasi (misalnya, `File baru terdeteksi`).
2. **Tambahkan File Uji:**
   - Salin `daily-data.xlsx` ke `Daily_Data` dan verifikasi pembaruan database, memeriksa log untuk keberhasilan atau error.
3. **Akses Dashboard:**
   - Buka browser di `http://127.0.0.1:5000` untuk melihat statistik dan plot.
   - Gunakan tautan sidebar untuk menguji `/update` dan `/database-status`.
4. **Tangani Kesalahan:**

- **Masalah Metadata:** Jika `Missing or invalid metadata`, buka `daily-data.xlsx`, pastikan `B2` dan `J3` memiliki data, dan log nilai mentah untuk debugging.
- **Ketidaksesuaian Kolom:** Jika `KeyError`, log `daily_data.columns` dan sesuaikan `transform_data` untuk indeks atau header.
- **Error Matplotlib:** Jika `RuntimeError`, pastikan `matplotlib.use('Agg')` ada di awal skrip.
- **Error 405:** Jika `/update` gagal, verifikasi eror tersebut dengan melihat GET di dashboard.

Penanganan Kesalahan Rinci dan Solusi

Kesalahan	Penyebab	Solusi
`FileNotFoundError` untuk `daily_folder`	Jalur folder atau file salah	Verifikasi jalur, gunakan `os.path.exists()` untuk memeriksa
`PermissionError` untuk akses file	File terkunci atau tidak ada izin	Tutup Excel, periksa izin Windows
`KeyError: 'FROM'`	Header hilang atau salah nama di file harian	Log kolom, gunakan indeks, pastikan format file
`ValueError: Missing or invalid metadata`	Sel `B2` atau `J3` kosong atau salah format	Buka file, verifikasi data, log nilai mentah, buat ulang
`RuntimeError: main thread not in main loop`	Konflik backend Matplotlib Tkinter	Gunakan `matplotlib.use('Agg')` di awal skrip
`405 Method Not Allowed` untuk `/update`	Permintaan GET ke rute POST-only	Tambah `methods=['GET', 'POST']` ke rute `/update`
`Port Already in Use` untuk Flask	Port 5000 sedang digunakan	Ubah ke port lain (misalnya, 5001)

Kesimpulan

Panduan ini diharapkan membantu Anda membuat, menguji, dan memelihara sistem automasi data pengeboran di lokal komputer, menangani kesalahan melalui logging dan penyesuaian dari format data harian ke format database. Sistem ini mengotomatisasi pembaruan, menyediakan dashboard untuk memantau data, dan beroperasi secara offline, ideal untuk lingkungan kerja lapangan yang terkendala akses internet.

Peningkatan Potensial

- Tambahkan validasi data untuk penanganan error yang lebih kuat.
- Gunakan Plotly untuk visualisasi interaktif di dashboard.
- Implementasikan cadangan terjadwal untuk `drilling_database.xlsx`.

Referensi

- Dokumentasi Resmi Flask
- Library Watchdog Python

- Dokumentasi OpenPyXL
- Dokumentasi Pandas
- Dokumentasi Matplotlib