INFRASTRUKTUR DAN PLATFORM SAINS DATA

Implementasi Streaming Data Berbasis API untuk Pemantauan Harga Emas secara Real-Time

Disusun oleh :

Faiz Arfian Ilhami	**L200220102**
Labib Awwam Husnayya	**L200220147**
Tyar Berliana Umi Kalsum	**L200220178**
Nazwa Diajeng Istika R	**L200220251**
Azza Belva Darindro	**L200220278**

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

TAHUN 2024

A. Abstract

Dalam era digital yang serba cepat, kebutuhan akan informasi *real-time* menjadi sangat penting, terutama dalam industri keuangan dan investasi. Implementasi sistem **Streaming Data berbasis API** untuk pemantauan harga emas secara *real-time* menghadirkan solusi yang efisien dan andal bagi pengguna yang membutuhkan data terkini untuk pengambilan keputusan. Sistem ini memungkinkan data harga emas dari berbagai sumber terpercaya untuk dikumpulkan, diproses, dan disajikan secara langsung tanpa jeda waktu.

Kami mengidentifikasi beberapa tantangan utama dalam pengembangan sistem ini, termasuk masalah integrasi API, keandalan koneksi, dan pengelolaan data yang terus-menerus masuk dalam volume besar. Untuk mengatasi tantangan tersebut, kami mengusulkan solusi berupa pengelolaan koneksi API yang optimal, penggunaan mekanisme buffering untuk mencegah kehilangan data, serta penerapan arsitektur yang mampu menangani skalabilitas dan latensi rendah.

Selain itu, penelitian ini juga mengeksplorasi penggunaan teknologi terbaru, seperti **message brokers** (misalnya, Kafka) dan protokol komunikasi ringan seperti **WebSocket**, untuk meningkatkan efisiensi dan kecepatan streaming data. Studi ini memberikan panduan praktis bagi organisasi yang ingin mengembangkan atau meningkatkan sistem pemantauan data harga secara real-time berbasis API.

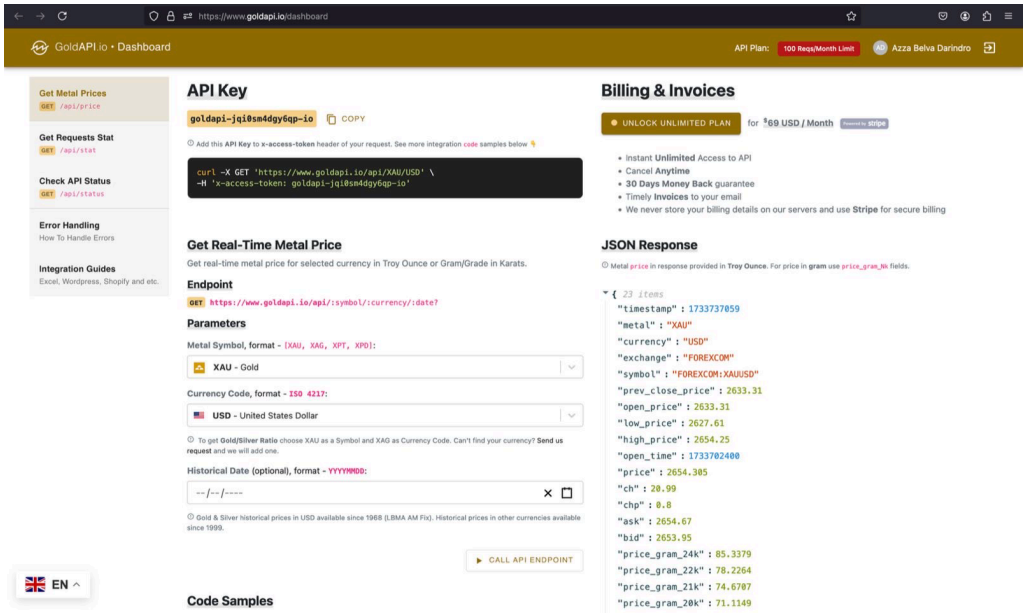
Hasil yang diperoleh diharapkan tidak hanya memberikan wawasan teknis mengenai implementasi streaming data, tetapi juga membuka peluang bagi aplikasi serupa dalam pemantauan aset keuangan lainnya. Dengan demikian, sistem ini berpotensi meningkatkan akurasi dan kecepatan dalam mendukung strategi investasi pengguna.

B. Menggunakan API

1. Menentukan API yang akan digunakan

   Antarmuka Pemrograman Aplikasi atau *Application Programming Interface (API)* adalah serangkaian aturan, protokol, dan alat yang memungkinkan berbagai aplikasi atau sistem untuk saling berkomunikasi. API berfungsi sebagai penghubung yang memungkinkan pengembang untuk

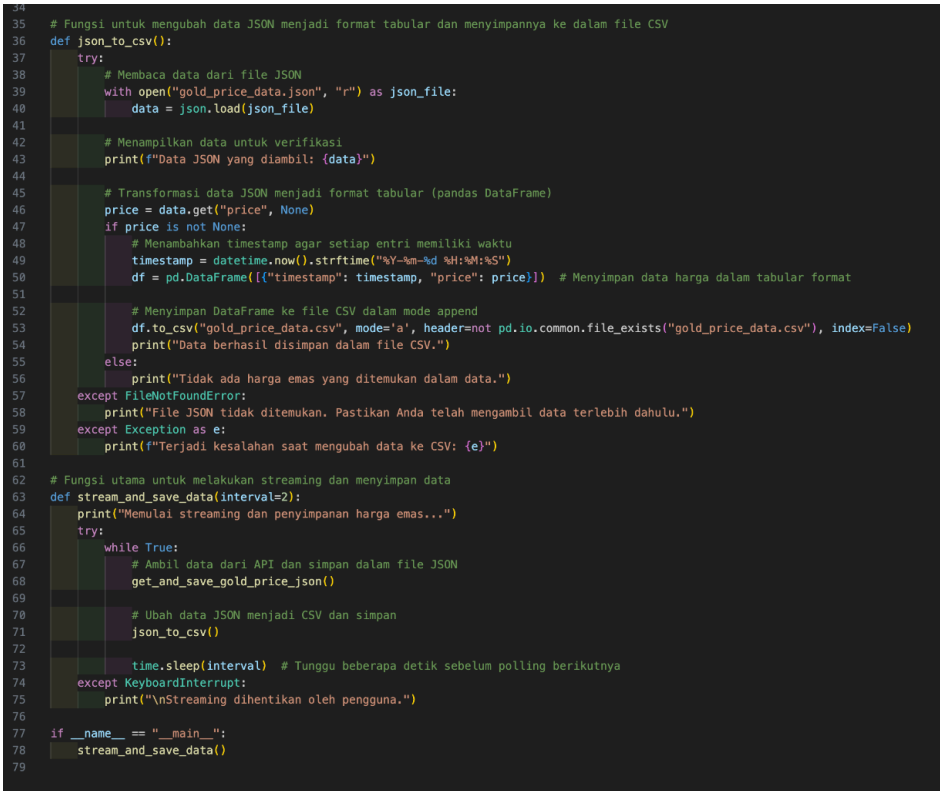
mengakses fitur atau data dari aplikasi lain tanpa harus memahami seluruh detail implementasinya.



API dapat diakses melalui link berikut https://www.goldapi.io/dashboard, menggunakan API tersebut untuk mendapatkan data harga emas secara *real-time.*

2. Memasukkan API ke dalam kode program

```python
import requests
import json
import pandas as pd
import time
from datetime import datetime

# Masukkan API Key Anda di sini
API_KEY = "goldapi-dv5sm4dh1bdw-io"

# Endpoint untuk mendapatkan harga XAU/USD
API_URL = "https://www.goldapi.io/api/XAU/USD"

# Header yang menyertakan API Key untuk autentikasi
HEADERS = {
    'x-access-token': API_KEY
}

# Fungsi untuk mengambil harga emas dari API dan menyimpannya dalam file JSON
def get_and_save_gold_price_json():
    try:
        response = requests.get(API_URL, headers=HEADERS, timeout=10)  # Timeout untuk menghin
        response.raise_for_status()  # Periksa apakah HTTP request berhasil
        data = response.json()

        # Menyimpan data dalam file JSON
        with open("gold_price_data.json", "w") as json_file:
            json.dump(data, json_file, indent=4)

        print("Data berhasil disimpan dalam file JSON.")
    except requests.exceptions.Timeout:
        print("Error: Permintaan ke API timeout. Periksa koneksi Anda.")
    except requests.exceptions.RequestException as e:
        print(f"Error saat mengambil data: {e}")
```

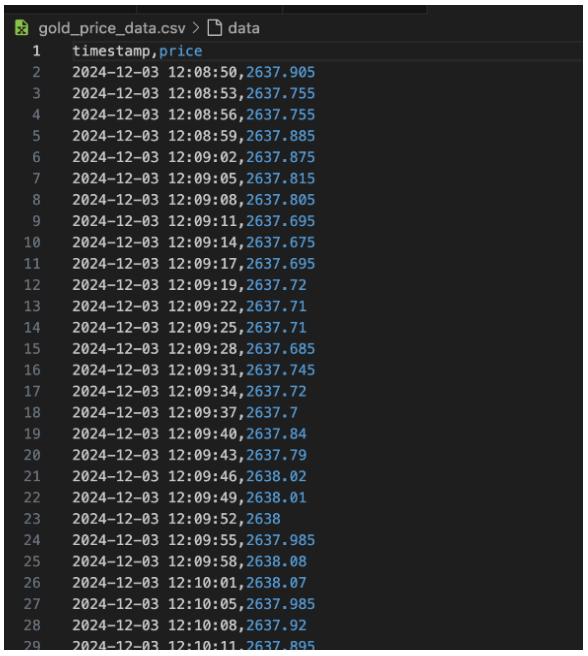


```python
# Fungsi untuk mengubah data JSON menjadi format tabular dan menyimpannya ke dalam file CSV
def json_to_csv():
    try:
        # Membaca data dari file JSON
        with open("gold_price_data.json", "r") as json_file:
            data = json.load(json_file)

        # Menampilkan data untuk verifikasi
        print(f"Data JSON yang diambil: {data}")

        # Transformasi data JSON menjadi format tabular (pandas DataFrame)
        price = data.get("price", None)
        if price is not None:
            # Menambahkan timestamp agar setiap entri memiliki waktu
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            df = pd.DataFrame([{"timestamp": timestamp, "price": price}])  # Menyimpan data harga dalam tabular format

            # Menyimpan DataFrame ke file CSV dalam mode append
            df.to_csv("gold_price_data.csv", mode='a', header=not pd.io.common.file_exists("gold_price_data.csv"), index=False)
            print("Data berhasil disimpan dalam file CSV.")
        else:
            print("Tidak ada harga emas yang ditemukan dalam data.")
    except FileNotFoundError:
        print("File JSON tidak ditemukan. Pastikan Anda telah mengambil data terlebih dahulu.")
    except Exception as e:
        print(f"Terjadi kesalahan saat mengubah data ke CSV: {e}")

# Fungsi utama untuk melakukan streaming dan menyimpan data
def stream_and_save_data(interval=2):
    print("Memulai streaming dan penyimpanan harga emas...")
    try:
        while True:
            # Ambil data dari API dan simpan dalam file JSON
            get_and_save_gold_price_json()

            # Ubah data JSON menjadi CSV dan simpan
            json_to_csv()

            time.sleep(interval)  # Tunggu beberapa detik sebelum polling berikutnya
    except KeyboardInterrupt:
        print("\nStreaming dihentikan oleh pengguna.")

if __name__ == "__main__":
    stream_and_save_data()
```

Hasil dari kode program tersebut menghasilkan timestamp (waktu), metal(simbol emas), *currency* (mata uang dalam USD), *exchange* (bursa atau platform yang diambil), symbol (simbol pasangan logam/mata uang), prev_close_price (harga penutupan emas sebelumnya), *open_price* (harga pembukaan emas saat perdagangan dimulai), low_price (harga terendah yang dicapai emas pada sesi perdagangan), *high_price* (harga tertinggi yang dicapai emas pada sesi perdagangan), *open_time* (waktu pembukaan perdagangan dalam Unix timestamp), *price* (harga emas terkini dalam USD per-ons), ch (Perubahan harga dibandingkan harga sebelumnya), chp (Persentase perubahan harga dibandingkan harga sebelumnya), *ask* (harga di mana penjual bersedia menjual emas)), bid (harga di mana pembeli bersedia membeli emas), *price_gram_24k* (harga emas per gram untuk emas 24 karat), *price_gram_22k* (harga emas per gram untuk emas 22 karat), *price_gram_20k* (harga emas per gram untuk emas 20 karat), *price_gram_18k* (harga emas per gram untuk emas 18 karat), *price_gram_16k* (harga emas per gram untuk emas 16 karat), *price_gram_14k* (harga emas per gram untuk emas 14 karat), *price_gram_10k* (harga emas per gram untuk emas 10 karat). Setiap 2 detik akan menyimpan

data json ditransfer menjadi data dengan format csv. Mengambil data berupa waktu, dan emas per-ons.

Data dalam format json



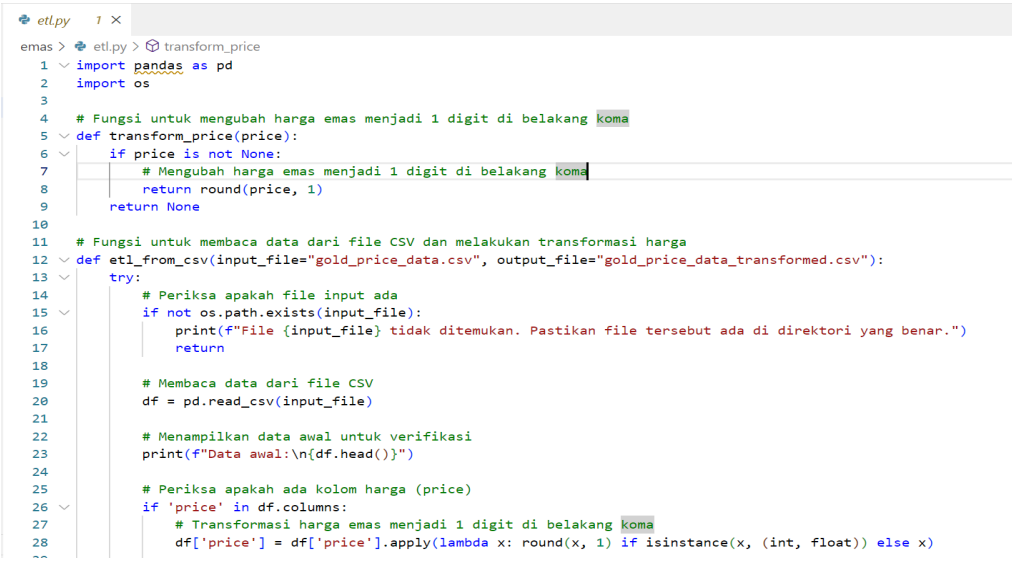
Data dalam format csv

gold_price_data_transformed.csv

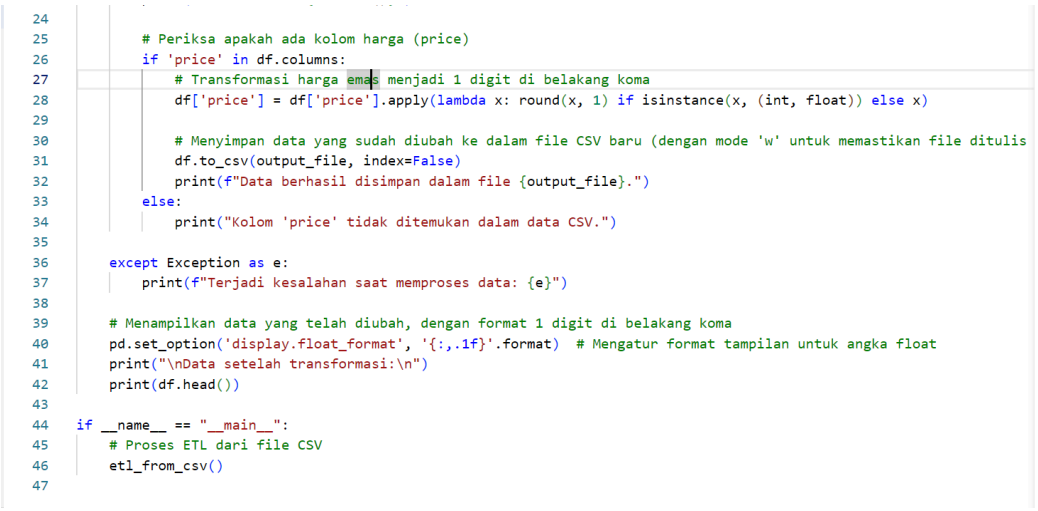
```
timestamp,price
2024-12-03 12:08:50,2637.9
2024-12-03 12:08:53,2637.8
2024-12-03 12:08:56,2637.8
2024-12-03 12:08:59,2637.9
2024-12-03 12:09:02,2637.9
2024-12-03 12:09:05,2637.8
2024-12-03 12:09:08,2637.8
2024-12-03 12:09:11,2637.7
2024-12-03 12:09:14,2637.7
2024-12-03 12:09:17,2637.7
2024-12-03 12:09:19,2637.7
2024-12-03 12:09:22,2637.7
2024-12-03 12:09:25,2637.7
2024-12-03 12:09:28,2637.7
2024-12-03 12:09:31,2637.7
2024-12-03 12:09:34,2637.7
2024-12-03 12:09:37,2637.7
2024-12-03 12:09:40,2637.8
2024-12-03 12:09:43,2637.8
2024-12-03 12:09:46,2638.0
2024-12-03 12:09:49,2638.0
2024-12-03 12:09:52,2638.0
2024-12-03 12:09:55,2638.0
2024-12-03 12:09:58,2638.1
2024-12-03 12:10:01,2638.1
2024-12-03 12:10:05,2638.0
2024-12-03 12:10:08,2637.9
2024-12-03 12:10:11,2637.9
```

Gold_price data transformed

C. Proses ETL



```python
import pandas as pd
import os

# Fungsi untuk mengubah harga emas menjadi 1 digit di belakang koma
def transform_price(price):
    if price is not None:
        # Mengubah harga emas menjadi 1 digit di belakang koma
        return round(price, 1)
    return None

# Fungsi untuk membaca data dari file CSV dan melakukan transformasi harga
def etl_from_csv(input_file="gold_price_data.csv", output_file="gold_price_data_transformed.csv"):
    try:
        # Periksa apakah file input ada
        if not os.path.exists(input_file):
            print(f"File {input_file} tidak ditemukan. Pastikan file tersebut ada di direktori yang benar.")
            return

        # Membaca data dari file CSV
        df = pd.read_csv(input_file)

        # Menampilkan data awal untuk verifikasi
        print(f"Data awal:\n{df.head()}")

        # Periksa apakah ada kolom harga (price)
        if 'price' in df.columns:
            # Transformasi harga emas menjadi 1 digit di belakang koma
            df['price'] = df['price'].apply(lambda x: round(x, 1) if isinstance(x, (int, float)) else x)
```



```python
        # Periksa apakah ada kolom harga (price)
        if 'price' in df.columns:
            # Transformasi harga emas menjadi 1 digit di belakang koma
            df['price'] = df['price'].apply(lambda x: round(x, 1) if isinstance(x, (int, float)) else x)

            # Menyimpan data yang sudah diubah ke dalam file CSV baru (dengan mode 'w' untuk memastikan file ditulis
            df.to_csv(output_file, index=False)
            print(f"Data berhasil disimpan dalam file {output_file}.")
        else:
            print("Kolom 'price' tidak ditemukan dalam data CSV.")

    except Exception as e:
        print(f"Terjadi kesalahan saat memproses data: {e}")

    # Menampilkan data yang telah diubah, dengan format 1 digit di belakang koma
    pd.set_option('display.float_format', '{:,.1f}'.format)  # Mengatur format tampilan untuk angka float
    print("\nData setelah transformasi:\n")
    print(df.head())

if __name__ == "__main__":
    # Proses ETL dari file CSV
    etl_from_csv()
```

Kode tersebut merupakan program Python untuk melakukan proses **ETL (Extract, Transform, Load)** sederhana dari file CSV yang berisi data harga emas. Fungsi **transform_price** digunakan untuk mengubah harga emas menjadi format dengan satu digit di belakang koma menggunakan fungsi **round()**. Fungsi utama, **etl_from_csv,** pertama-tama memeriksa apakah file input (**gold_price_data.csv**) ada di direktori yang benar. Jika file ditemukan, data dibaca menggunakan **pandas** dan divisualisasikan untuk verifikasi. Selanjutnya, kolom price pada DataFrame diperiksa, dan jika ada, harga emas dalam kolom tersebut diubah menggunakan **apply()** untuk memastikan setiap nilai harga memiliki satu tempat desimal. Hasil transformasi disimpan dalam file CSV baru (**gold_price_data_transformed.csv**). Jika kolom price tidak ditemukan atau terjadi kesalahan saat pemrosesan, program akan menampilkan

pesan yang sesuai. Setelah itu, data yang telah diubah ditampilkan dengan format satu tempat desimal. Proses ini bertujuan untuk memastikan bahwa data harga emas tersusun dengan benar dan siap untuk analisis lebih lanjut. Kode ini digunakan juga untuk membersihkan dan merapikan data harga emas dalam file CSV dengan format yang lebih seragam (1 digit di belakang koma).

D. **Proses Visualisasi**

1. visual1.py

```python
import requests
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from datetime import datetime
import time

# Masukkan API Key Anda di sini
API_KEY = "goldapi-15f6ksm4gyhn8v-id"

# Endpoint untuk mendapatkan harga XAU/USD
API_URL = "https://www.goldapi.io/api/XAU/USD"

# Header yang menyertakan API Key untuk autentikasi
HEADERS = {
    'x-access-token': API_KEY
}

# Data awal untuk grafik
times = []
prices = []

# Fungsi untuk mengambil harga emas dari API
def get_gold_price():
    try:
        response = requests.get(API_URL, headers=HEADERS, timeout=10)  # Timeout untuk menghindari hang
        response.raise_for_status()  # Periksa apakah HTTP request berhasil
        data = response.json()
        return data.get('price', None)
```

```python
def get_gold_price():
    try:
    except requests.exceptions.RequestException as e:
        print(f"Error saat mengambil data: {e}")
        return None

# Fungsi untuk memperbarui grafik
def update(frame):
    # Ambil data harga emas terbaru
    price = get_gold_price()
    if price is not None:
        # Menambahkan data waktu dan harga ke list
        times.append(datetime.now().strftime('%H:%M:%S'))
        prices.append(price)

        # Membatasi jumlah data yang ditampilkan (misalnya hanya 30 titik terakhir)
        if len(times) > 30:
            times.pop(0)
            prices.pop(0)

        # Clear grafik sebelumnya dan plot data terbaru
        ax.clear()
        ax.plot(times, prices, marker='o', color='gold', linestyle='-', linewidth=2, markersize=5)
        ax.set_title('Harga Emas (XAU/USD) Seiring Waktu', fontsize=14)
        ax.set_xlabel('Waktu', fontsize=12)
        ax.set_ylabel('Harga Emas (USD)', fontsize=12)
        plt.xticks(rotation=45)
        ax.grid(True)
```

```
# Membuat figure dan axis untuk plot
fig, ax = plt.subplots(figsize=(10, 6))

# Membuat animasi untuk memperbarui grafik setiap 5 detik
ani = FuncAnimation(fig, update, interval=5000)  # interval dalam milidetik (5000ms = 5 detik)

# Menampilkan grafik secara real-time
plt.tight_layout()
plt.show()
```

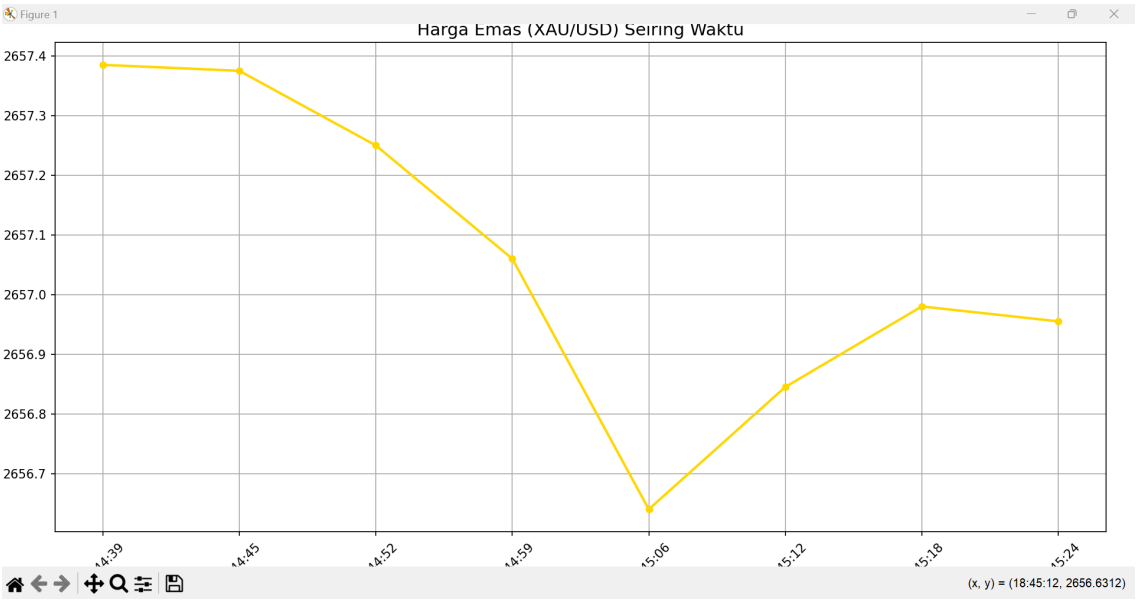
Kode tersebut adalah skrip Python yang melakukan visualisasi data harga emas dengan menggunakan data dari **GoldAPI.** Dengan menggunakan beberapa library seperti **requests, matplotlib**, dan **datetime**, kode ini membuat grafik yang diperbarui secara berkala untuk menampilkan fluktuasi harga emas.

Fungsi utama dalam kode ini adalah **get_gold_price()**, yang mengambil data harga emas terkini dari API menggunakan permintaan **HTTP GET**. Data yang diterima dalam format JSON diolah untuk mengambil nilai harga emas. Fungsi ini mengatasi potensi kesalahan saat permintaan data dengan penanganan pengecualian, memastikan program tidak terhenti saat terjadi kesalahan jaringan atau masalah API lainnya.

Grafik harga emas diperbarui setiap 5 detik dengan menggunakan **matplotlib.animation.FuncAnimation**. Setiap pembaruan, fungsi **update()** menambahkan waktu dan harga baru ke dalam *list* ***times*** dan ***prices*****,** dan hanya menampilkan 30 titik data terbaru untuk menjaga grafik tetap rapi. Grafik tersebut dibuat dengan garis berwarna emas, yang menampilkan harga emas seiring waktu, sehingga memudahkan pengguna untuk memantau pergerakan harga secara visual.

Kode ini cocok digunakan dalam aplikasi yang memerlukan pemantauan harga emas secara langsung, seperti dalam platform investasi atau untuk analisis pasar. Penting untuk memastikan bahwa API key yang digunakan valid dan perhatikan batasan penggunaan API agar tidak terblokir. Dengan pemanfaatan kode ini, pengguna dapat mengakses data harga emas terkini dalam format yang mudah dipahami dan visualisasi yang menarik.

Hasil:



2. visual2.py

```python
import pandas as pd
import matplotlib.pyplot as plt

# Fungsi untuk membaca data dari file CSV
def load_data_from_csv(csv_file="gold_price_data_transformed.csv"):
    try:
        # Membaca data dari file CSV
        df = pd.read_csv(csv_file)

        # Pastikan kolom 'timestamp' dan 'price' ada dalam data
        if 'timestamp' in df.columns and 'price' in df.columns:
            return df['timestamp'], df['price']
        else:
            print("File CSV tidak memiliki kolom yang diperlukan.")
            return pd.Series(), pd.Series()  # Mengembalikan Series kosong
    except Exception as e:
        print(f"Error saat membaca file CSV: {e}")
        return pd.Series(), pd.Series()  # Mengembalikan Series kosong

# Fungsi untuk memperbarui grafik
def update_graph(times, prices):
    # Clear grafik sebelumnya dan plot data terbaru
    ax.clear()
    ax.plot(times, prices, marker='o', color='gold', linestyle='-', linewidth=2, markersize=5)
    ax.set_title('Harga Emas (XAU/USD) Seiring Waktu (per ounce)', fontsize=14)
    ax.set_xlabel('Waktu', fontsize=12)
    ax.set_ylabel('Harga Emas (USD)', fontsize=12)
    plt.xticks(rotation=45)
    ax.grid(True)

# Membaca data dari file CSV
times, prices = load_data_from_csv("gold_price_data_transformed.csv")

# Memeriksa apakah data ada
if not times.empty and not prices.empty:
    # Membuat figure dan axis untuk plot
    fig, ax = plt.subplots(figsize=(10, 6))

    # Menampilkan grafik dengan data yang sudah ada
    update_graph(times, prices)

    # Menampilkan grafik
    plt.tight_layout()
    plt.show()
else:
    print("Tidak ada data untuk ditampilkan.")
```

Kode digunakan untuk memvisualisasikan harga emas dari file CSV yang sudah diproses. Fungsi load_data_from_csv digunakan untuk membaca

data dari file CSV (**gold_price_data_transformed.csv**). Fungsi ini memastikan bahwa file tersebut mengandung kolom timestamp dan price, yang merupakan data waktu dan harga emas. Jika kolom yang diperlukan ada, fungsi ini mengembalikan dua Series berisi waktu dan harga emas, yang akan digunakan untuk membuat grafik. Fungsi **update_graph** bertanggung jawab untuk memperbarui grafik dengan data yang baru. Grafik akan menampilkan harga emas seiring waktu, dengan waktu pada sumbu x dan harga emas pada sumbu y. Grafik ini dibuat dengan menggunakan matplotlib dan diatur dengan label sumbu, judul, serta rotasi pada label waktu agar lebih mudah dibaca. Setelah data dibaca dan diverifikasi, grafik akan ditampilkan menggunakan **plt.show().** Jika file CSV tidak ada atau tidak mengandung data yang diperlukan, pesan error akan ditampilkan. Kode ini memberikan cara yang efektif untuk memvisualisasikan data harga emas secara dinamis.

Hasil :