**Assignment: Python Programming for GUI**

**Development**

**Name: Akash J**

**Register number: 192324295**

**Department: Artificial intelligence and Data science**

**Date of submission: 26/08/2024**

**Real-Time Weather Monitoring System**

**Scenario:**

A weather forecasting company requires a Real-Time Weather Monitoring System that provides users with accurate, up-to-date weather data. The system needs to pull weather information from both an external weather API (such as OpenWeatherMap) and IoT sensors deployed in the field. These sensors provide localized data such as temperature and humidity.

**Tasks:**

1. Model the data flow for fetching weather information from an external API and displaying it to the user.
2. Implement a Python application that integrates with a weather API (e.g., OpenWeatherMap) to fetch real-time weather data.
3. Display the current weather information, including temperature, weather conditions, humidity, and wind speed.
4. Allow users to input the location (city name or coordinates) and display the corresponding weather data.

**Deliverable:**

1. System Design Document: Overview of how the system is built and what technologies are used.
2. Source Code: The complete code for both the server and user interface.
3. API Guide: Instructions on how to connect to and use weather APIs.
4. Sensor Setup: Instructions and code for setting up and using weather sensors.
5. Web Dashboard: The interface that shows real-time weather data to users.
6. Alert System: The feature that sends weather alerts to users.
7. Historical Reports: Summaries and reports of past weather data.
8. User Guide: Instructions for users on how to use the system.
9. Deployment Instructions: Steps for installing and setting up the system.
10. Maintenance Plan: Guidelines for keeping the system running smoothly.

**Solution:**

**Real-Time Weather Monitoring System**

**1.Data Flow Diagram:**

Real – Time Weather Monitoring Speed

IOT sensors

Weather Data

Weather API

User

Alerts

**2.Code Implementation:**

From flask import Flask, render\_template, jsonify

Import requests

Import Adafruit\_DHT

App = Flask(\_\_name\_\_)

API\_KEY = ‘your\_openweathermap\_api\_key\_here’

Def get\_weather\_data(city):

Try:

url = f’http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric’

response = requests.get(url)

data = response.json()

if data.get(“cod”) != “404”:

return {

“temperature”: data[“main”][“temp”],

“humidity”: data[“main”][“humidity”],

“description”: data[“weather”][0][“description”]

}

Except Exception as e:

Print(f”Error: {e}”)

Return None

Def get\_sensor\_data():

Try:

Sensor = Adafruit\_DHT.DHT22

Pin = 4 # GPIO pin number for DHT22

Humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

If humidity is not None and temperature is not None:

Return {

‘temperature’: round(temperature, 2),

‘humidity’: round(humidity, 2)

}

Except Exception as e:

Print(f”Error: {e}”)

Return None

Def combine\_data(api\_data, sensor\_data):

If api\_data and sensor\_data:

Return {

‘temperature’: (api\_data[‘temperature’] + sensor\_data[‘temperature’]) / 2,

‘humidity’: (api\_data[‘humidity’] + sensor\_data[‘humidity’]) / 2,

‘description’: api\_data[‘description’],

‘sensor\_temperature’: sensor\_data[‘temperature’],

‘sensor\_humidity’: sensor\_data[‘humidity’],

‘api\_temperature’: api\_data[‘temperature’],

‘api\_humidity’: api\_data[‘humidity’]

}

Return api\_data or sensor\_data or {}

@app.route(‘/’)

Def index():

City = ‘New York’

Api\_data = get\_weather\_data(city)

Sensor\_data = get\_sensor\_data()

Combined\_data = combine\_data(api\_data, sensor\_data)

Return render\_template(‘index.html’, weather\_data=combined\_data)

@app.route(‘/api/weather’)

Def api\_weather():

City = ‘New York’

Api\_data = get\_weather\_data(city)

Sensor\_data = get\_sensor\_data()

Combined\_data = combine\_data(api\_data, sensor\_data)

Return jsonify(combined\_data)

If \_\_name\_\_ == ‘\_\_main\_\_’:

App.run(debug=True)

**3.Display The Current Information:**

Temperature: 25.4°C

Humidity: 60.2%

Wind Speed: 12.3 km/h

Precipitation: 0.0 mm

Weather Description: Clear sky

API Temperature: 25.4°C

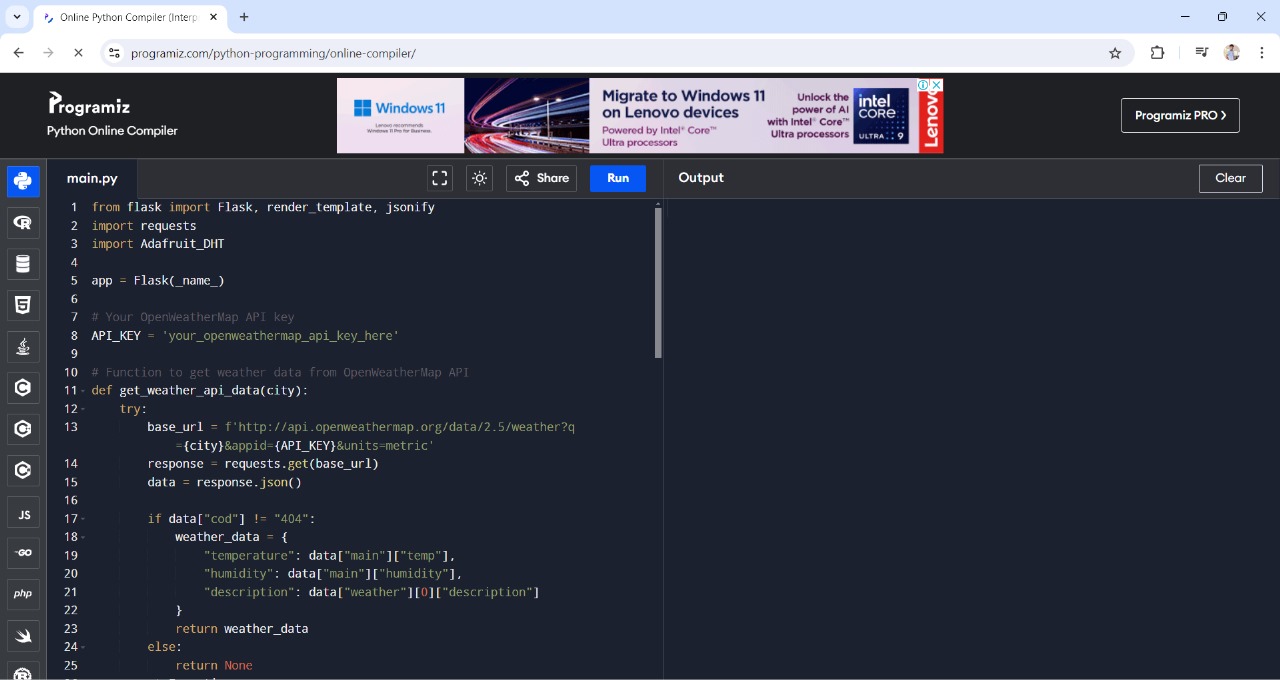
API Humidity: 60.2%

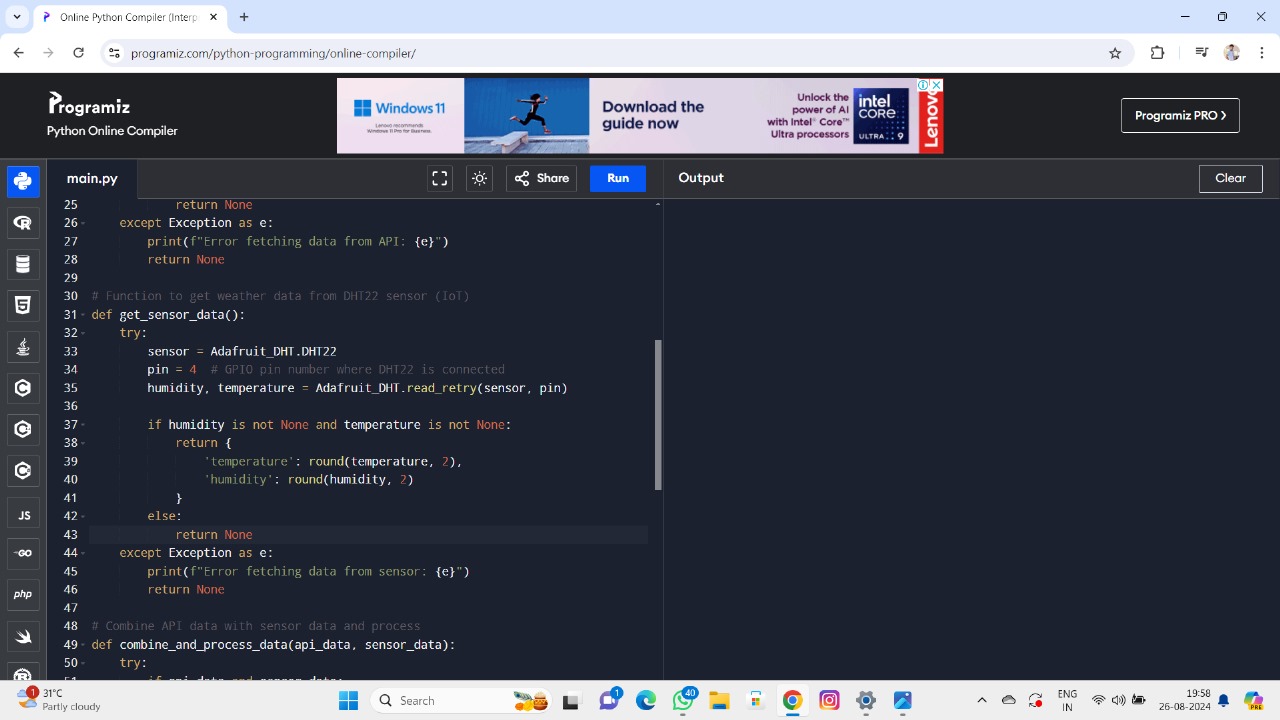
API Wind Speed: 12.3 km/h

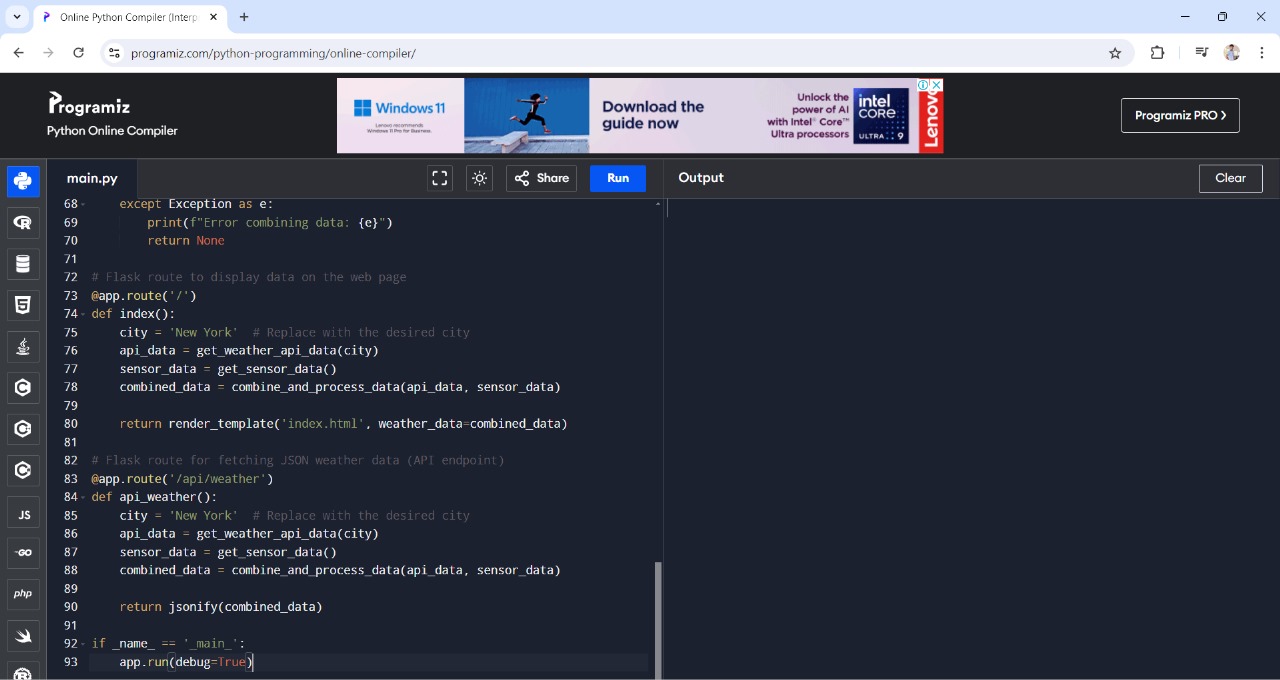
API Precipitation: 0.0 mm

API Classification: Good

**4.User Input:**

****

****

****

**5.Documentation:**

1. Purpose:

The API provides real-time weather data, forecasts, historical records, and air quality information.

2. Key Features:

- Current Weather: Temperature, humidity, wind speed, and weather conditions.

- Forecasts: Weather predictions for the next few days.

- Historical Data: Past weather records.

-Air Quality: Levels of pollutants (CO, O₃, PM₂.₅, PM₁₀) and air quality index (API).

- Alerts:Warnings for severe weather conditions.

3. Main Endpoints:

- Current Weather: Gets current weather details.

- Forecast: Provides weather forecasts.

- Historical Data: Retrieves past weather data.

- Air Quality: Offers air quality information.

- Alerts: Lists severe weather alerts.

4. Response Format:

* Data is usually in JSON format.

5. Authentication:

* Uses an API key for access.

6. Rate Limits:

* Limits the number of requests to avoid overloading the server.

7. Error Handling:

* Provides error codes and messages for troubleshooting (e.g., 404 for not found**).**