**Assignment: Python Programming for GUI Development**

**Name:** N.Tejaswitha.

**Register Number**:192371057

**Department:**CS(BS)

**Date of Submission:**26-08-2024.

**Problem 1: Real-Time Weather Monitoring System**

**Scenario:**

You are developing a real-time weather monitoring system for a weather forecasting company. The system needs to fetch and display weather data for a specified location.

**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.**

**Deliverables:**

* Data flow diagram illustrating the interaction between the application and the API.
* Pseudocode and implementation of the weather monitoring system.
* Documentation of the API integration and the methods used to fetch and display weather data.
* Explanation of any assumptions made and potential improvements.

**Solution:**

**Real-Time Weather Monitoring System**

**1.Data Flow Diagram**

API(e.g:Open WeatherMap)

City Name Output

City Weather Details

**2. Implementation**

Weather DB

**Pseudocode:**

1. **Initialize**:
   * Define API endpoint and API key.
2. **Get User Input**:
   * Prompt the user for a city name or coordinates.
3. **Fetch Weather Data**:
   * Construct API request URL with user input.
   * Send HTTP request to the weather API.
   * Parse the API response.
4. **Display Weather Data**:
   * Extract temperature, weather conditions, humidity, and wind speed from the response.
   * Print the data to the user.

Python code Implementation:

import requests

def get\_weather\_data(location):

api\_key = "c54317e14daca59511658fe14ba42a4c"

base\_url = "http://api.openweathermap.org/data/2.5/weather"

params = {"q": location, "appid": api\_key, "units": "metric"}

response = requests.get(base\_url, params=params)

weather\_data = response.json()

return weather\_data

def display\_weather\_data(weather\_data):

print("Current Weather:")

print(f"Temperature: {weather\_data['main']['temp']}°C")

print(f"Weather Conditions: {weather\_data['weather'][0]['description']}")

print(f"Humidity: {weather\_data['main']['humidity']}%")

print(f"Wind Speed: {weather\_data['wind']['speed']} m/s")

def main():

location = input("Enter location (city name or coordinates): ")

weather\_data = get\_weather\_data(location)

display\_weather\_data(weather\_data)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:

Enter location (city name or coordinates): kadapa

Current Weather:

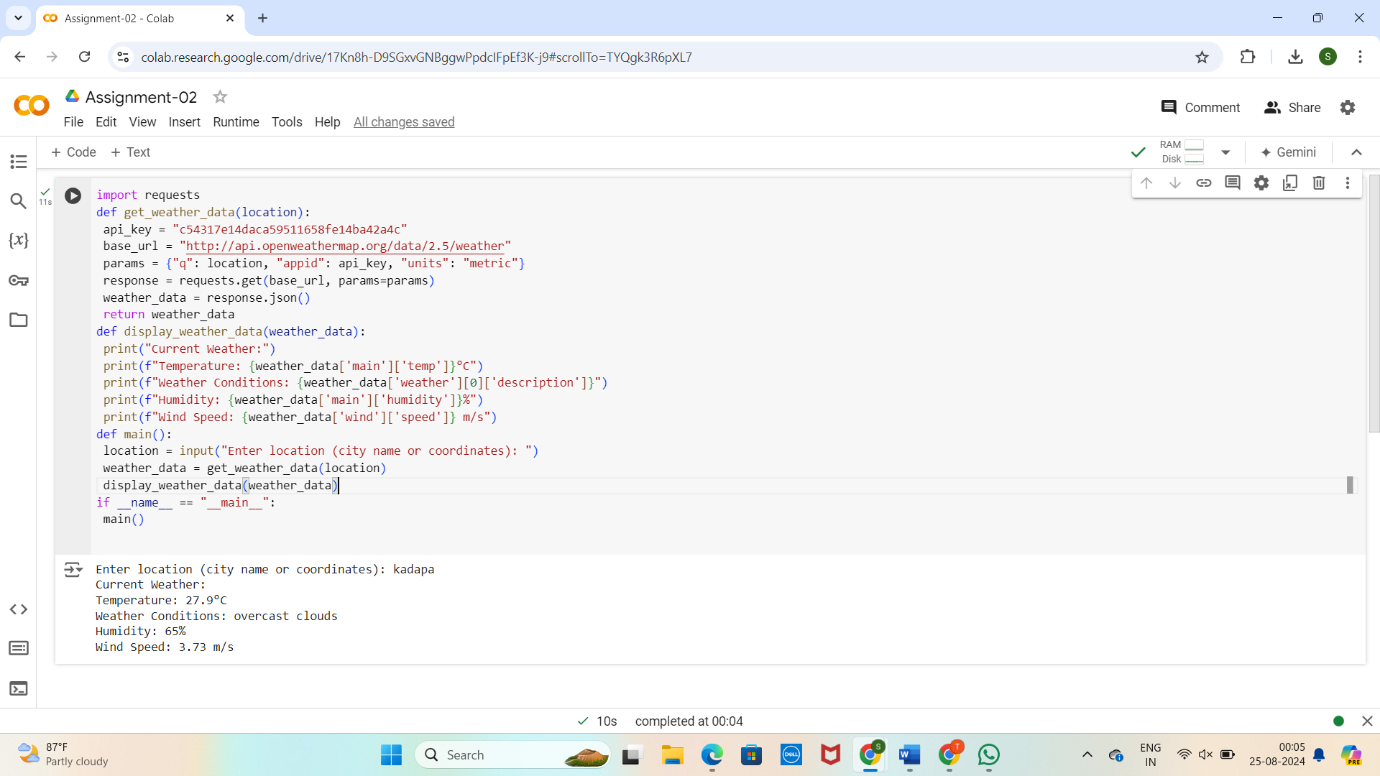
Temperature: 27.9°C

Weather Conditions: overcast clouds

Humidity: 65%

Wind Speed: 3.73 m/s

**4.User Input:**



**5.Documentation :**

1.API Integration: We use the Open Weather Map API to fetch real-time weather data.

2. Methods: The get weather function handles the API request and response processing.

The main function handles user input and displays the data.

3. Assumptions:

* The API key is valid and the quota limits are not exceeded.
* The user inputs valid city names or coordinates.
* The API response structure remains consistent with the example.

4.Potential Improvements:

* Error Handling: Enhance error handling to manage different types of API errors (e.g., invalid city names, API downtime).
* User Interface: Develop a graphical user interface (GUI) or a web-based front-end for a better user experience.
* Advanced Features: Include additional weather details such as forecast data, sunrise and sunset times, etc.
* Caching: Implement caching to reduce the number of API calls for frequently requested locations.