**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.

**Data Flow Diagram:**

**+----------------+ +-------------------+ +---------------------+**

**| User Interface | <--> |Python Application | <--> | Weather API (e.g., |**

**| (User Input) | | (Backend) | | OpenWeatherMap)|**

**+----------------+ +-------------------+ +---------------------+**

**|**

**V**

**+---------------------+**

**| Weather Data Display |**

**| (User Interface) |**

**+---------------------+**

**Pseudocode:**

# Function to fetch weather data from OpenWeatherMap API

def fetch\_weather\_data(location):

api\_key = "YOUR\_OPENWEATHERMAP\_API\_KEY"

api\_url = f"http://api.openweathermap.org/data/2.5/weather?q={location}&appid={api\_key}"

response = requests.get(api\_url)

if response.status\_code == 200:

data = response.json()

return data

else:

return None

# Function to display weather data to the user

def display\_weather\_data(data):

if data:

print(f"Current Weather in {data['name']}:")

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

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

print(f"Humidity: {data['main']['humidity']}%")

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

else:

print("Error fetching weather data.")

# Main function to handle user input and display weather data

def main():

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

data = fetch\_weather\_data(location)

display\_weather\_data(data)

# Run the main function

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

main()

**System Architecture**

The system architecture revolves around a Python application that interacts with the

OpenWeatherMap API. It consists of two main components:

1. **WeatherAPI Class:**

o Manages HTTP requests to the OpenWeatherMap API.

o Provides methods to fetch weather data by city name

(get\_weather\_by\_city\_name) or coordinates

(get\_weather\_by\_coordinates).

2. **WeatherApp Class:**

o User-facing interface managing user input and displaying weather data.

o Integrates with the WeatherAPI class to fetch and display weather information

based on user input.

**Assumptions Made and Potential Improvements**

• **Assumptions:**

o Users will input either a valid city name or coordinates in the format

latitude,longitude.

o Only basic weather information (temperature, humidity, wind speed, weather

description) is displayed

.• **Potential Improvements:**

o Error handling for API requests (e.g., network errors, invalid responses).

o Adding more detailed weather information (e.g., sunrise/sunset times,

pressure).

o Enhancing user interface for better user experience (e.g., graphical interface).

o Support for multiple languages and units (currently metric units for

temperature).

This system provides a basic framework for fetching and displaying weather data from an

API, focusing on simplicity and functionality

**Code**:

import requests

class WeatherAPI:

def \_\_init\_\_(self, api\_key):

self.api\_key = api\_key

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

def get\_weather\_by\_city\_name(self, city\_name):

params = {'q': city\_name, 'appid': self.api\_key, 'units': 'metric'}

return self.\_get\_weather(params)

def get\_weather\_by\_coordinates(self, lat, lon):

params = {'lat': lat, 'lon': lon, 'appid': self.api\_key, 'units': 'metric'}

return self.\_get\_weather(params)

def \_get\_weather(self, params):

try:

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

response.raise\_for\_status()

return response.json()

except requests.exceptions.RequestException as e:

print(f"Error fetching data: {e}")

return None

class WeatherApp:

def \_\_init\_\_(self, api\_key):

self.api = WeatherAPI(api\_key)

def run(self):

print("Welcome to the Real-Time Weather Monitoring System!")

print("Enter 'quit' at any time to exit.\n")

while True:

location = input("Enter city name or coordinates (latitude,longitude): ").strip()

if location.lower() == 'quit':

break

weather\_data = self.fetch\_weather(location)

if weather\_data:

self.display\_weather(weather\_data)

else:

print("Failed to fetch weather data. Please try again.\n")

def fetch\_weather(self, location):

if self.\_is\_valid\_coordinates(location):

lat, lon = location.split(',')

return self.api.get\_weather\_by\_coordinates(lat.strip(), lon.strip())

else:

return self.api.get\_weather\_by\_city\_name(location)

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

print(f"Weather in {weather\_data['name']} ({weather\_data['sys']['country']}):")

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

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

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

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

def \_is\_valid\_coordinates(self, location):

parts = location.split(',')

if len(parts) == 2:

try:

lat = float(parts[0])

lon = float(parts[1])

return True

except ValueError:

pass

return False

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

API\_KEY = 'fe1fb635f6a319191b58ab07034c74c9'

app = WeatherApp(API\_KEY)

app.run()