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

**1. Data Flow Diagram:**

**Here’s the data flow for the weather monitoring system:**

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

**| User | | Python Program | | Weather API |**

**| (City/Location) |<------->| Fetch Weather |<------->| (OpenWeatherMap) |**

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

**|**

**v**

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

**| Display Data |**

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

1. **User Input: The user inputs the location (city name or coordinates).**
2. **Python Program: The program sends the location to the weather API.**
3. **Weather API: The API returns weather data for the specified location.**
4. **Display Data: The program processes and displays the weather data to the user.**

**2. Pseudocode:**

**BEGIN**

**Prompt user to enter city name or coordinates (latitude, longitude)**

**IF user input is valid THEN**

**Send request to the weather API with the user-provided location**

**Receive response with weather data (temperature, humidity, wind speed, weather conditions)**

**IF response is successful THEN**

**Parse and extract weather data**

**Display the weather information to the user**

**ELSE**

**Display error message to the user (e.g., invalid city name or server error)**

**END IF**

**ELSE**

**Display invalid input message to the user**

**END IF**

**END**

**3. Detailed Explanation of the Code:**

1. **Fetch Data from OpenWeatherMap:** The Python script uses the **requests** module to fetch weather data by sending a GET request to the OpenWeatherMap API.
2. **Input Validation:** The system ensures that the user input (city name or coordinates) is valid before making the API request.
3. **Error Handling:** If the API call fails (e.g., invalid city, API key issues), the program gracefully handles the error by displaying a message to the user.
4. **Data Display:** The weather data, including temperature, weather conditions, humidity, and wind speed, is displayed to the user.

**4. Python Implementation:**

Make sure to install the necessary package by running:

pip install requests

Here's the Python implementation:

import requests  
import matplotlib.pyplot as plt  
from tkinter import Tk, Label, OptionMenu, StringVar, Button, Frame  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from datetime import datetime  
from PIL import Image, ImageTk  
  
# Define the API endpoint and your API key  
url = "https://weatherapi-com.p.rapidapi.com/current.json"  
headers = {  
 "x-rapidapi-key": "634ef24f07mshc0598f2465d3939p16058cjsn92985c8a46ea",  
 "x-rapidapi-host": "weatherapi-com.p.rapidapi.com"  
}  
  
# Initialize lists to store weather data  
times = []  
temperatures = []  
humidities = []  
rainfalls = []  
colors = []  
  
# Season images paths  
season\_images = {  
 'summer': "C:\\Users\\Sasi Kumar\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093204.png",  
 'winter': "C:\\Users\\Sasi Kumar\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093226.png",  
 'monsoon': "C:\\Users\\Sasi Kumar\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093822.png",  
 'autumn': "C:\\Users\\Sasi Kumar\\OneDrive\\Pictures\\Screenshots\\Screenshot 2024-08-17 093213.png"  
}  
  
# Function to fetch weather data  
def fetch\_weather(city):  
 querystring = {"q": city}  
 try:  
 response = requests.get(url, headers=headers, params=querystring)  
 response.raise\_for\_status()  
 data = response.json()  
  
 current = data['current']  
 temperature = current['temp\_c']  
 humidity = current['humidity']  
 rainfall = current.get('precip\_mm', 0)  
 condition = current['condition']['text'].lower()  
  
 # Determine graph color based on weather condition  
 if 'rain' in condition:  
 color = 'blue'  
 elif 'cloud' in condition:  
 color = 'gray'  
 else:  
 color = 'orange'  
  
 return temperature, humidity, rainfall, color  
  
 except requests.exceptions.RequestException as e:  
 print(f"API request error: {e}")  
 return None, None, None, 'black'  
 except ValueError as e:  
 print(f"Value error: {e}")  
 return None, None, None, 'black'  
  
# Function to determine the season  
def get\_season(temperature):  
 if temperature >= 30:  
 return 'summer'  
 elif temperature <= 10:  
 return 'winter'  
 elif 10 < temperature < 30:  
 if temperature < 20:  
 return 'autumn'  
 else:  
 return 'monsoon'  
 return 'autumn'  
  
# Function to update the plot and weather information  
def update\_plot():  
 city = city\_var.get()  
 temperature, humidity, rainfall, color = fetch\_weather(city)  
  
 if temperature is not None:  
 times.append(datetime.now().strftime('%H:%M:%S'))  
 temperatures.append(temperature)  
 humidities.append(humidity)  
 rainfalls.append(rainfall)  
 colors.append(color)  
  
 ax.clear()  
 ax.plot(times, temperatures, color='red', label='Temperature (°C)')  
 ax.plot(times, humidities, color='green', label='Humidity (%)')  
 ax.plot(times, rainfalls, color='blue', label='Rainfall (mm)')  
  
 ax.set\_xlabel('Time')  
 ax.set\_ylabel('Value')  
 ax.legend(loc='upper left')  
 ax.set\_title(f'Weather Data for {city}')  
 plt.tight\_layout()  
  
 # Add semi-opaque watermark  
 plt.text(0.5, 0.5, 'Naash Weather Tracking', fontsize=40, color='gray', alpha=0.3,  
 ha='center', va='center', transform=ax.transAxes)  
  
 # Update the text labels with the latest values  
 current\_temperature.set(f"Temperature: {temperature} °C")  
 current\_humidity.set(f"Humidity: {humidity} %")  
 current\_rainfall.set(f"Rainfall: {rainfall} mm")  
  
 # Determine the season  
 season = get\_season(temperature)  
 season\_image\_path = season\_images.get(season, "C:\\Users\\Sasi Kumar\\Downloads\\Design.jpeg")  
  
 # Load the image and display it  
 image = Image.open(season\_image\_path)  
 season\_photo = ImageTk.PhotoImage(image)  
 season\_label.config(image=season\_photo)  
 season\_label.image = season\_photo # Keep a reference to avoid garbage collection  
  
 # Redraw the canvas  
 canvas.draw()  
  
 root.after(60000, update\_plot)  
  
# Set up the GUI  
def setup\_gui():  
 global city\_var, root, canvas, fig, ax  
 global current\_temperature, current\_humidity, current\_rainfall, season\_label  
  
 root = Tk()  
 root.title("Naash Weather Tracker")  
  
 control\_frame = Frame(root, bg='white')  
 control\_frame.pack(side='top', fill='x')  
  
 Label(control\_frame, text="Select City:", bg='white').pack(side='left')  
  
 city\_var = StringVar(root)  
 city\_var.set("Mumbai")  
  
 city\_dropdown = OptionMenu(control\_frame, city\_var, "Mumbai", "Delhi", "Bangalore", "Kolkata", "Chennai",  
 "Hyderabad", "Lucknow", "Amritsar", "Jaipur", "Shimla")  
 city\_dropdown.pack(side='left')  
  
 refresh\_button = Button(control\_frame, text="Refresh", command=update\_plot)  
 refresh\_button.pack(side='left')  
  
 info\_frame = Frame(root, bg='white')  
 info\_frame.pack(side='top', fill='x')  
  
 current\_temperature = StringVar()  
 current\_humidity = StringVar()  
 current\_rainfall = StringVar()  
  
 Label(info\_frame, textvariable=current\_temperature, bg='white', font=('Helvetica', 16)).pack(side='left')  
 Label(info\_frame, textvariable=current\_humidity, bg='white', font=('Helvetica', 16)).pack(side='left')  
 Label(info\_frame, textvariable=current\_rainfall, bg='white', font=('Helvetica', 16)).pack(side='left')  
  
 season\_label = Label(root)  
 season\_label.pack(side='bottom', fill='x')  
  
 fig, ax = plt.subplots(figsize=(10, 6))  
  
 canvas = FigureCanvasTkAgg(fig, master=root)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack(side='bottom', fill='both', expand=True)  
  
 update\_plot()  
 root.mainloop()  
  
# Run the GUI setup  
setup\_gui()

**5. Documentation of API Integration:**

* **API Used:** OpenWeatherMap API (**http://api.openweathermap.org/data/2.5/weather**)
* **API Key:** Requires an API key from OpenWeatherMap. You can get one by signing up at [OpenWeatherMap](https://openweathermap.org/).
* **Endpoint:** The script uses the **/weather** endpoint.
* **Parameters:**
  + **q**: City name (location)
  + **appid**: API key for authentication
  + **units**: Metric system (to return temperature in Celsius)
* **Response Data:** The API returns data in JSON format, which includes the current temperature, weather conditions, humidity, and wind speed.

**6. Assumptions Made:**

* The user will enter a valid city name in English.
* The OpenWeatherMap API will be available and responsive during the request.
* Temperature is displayed in Celsius.

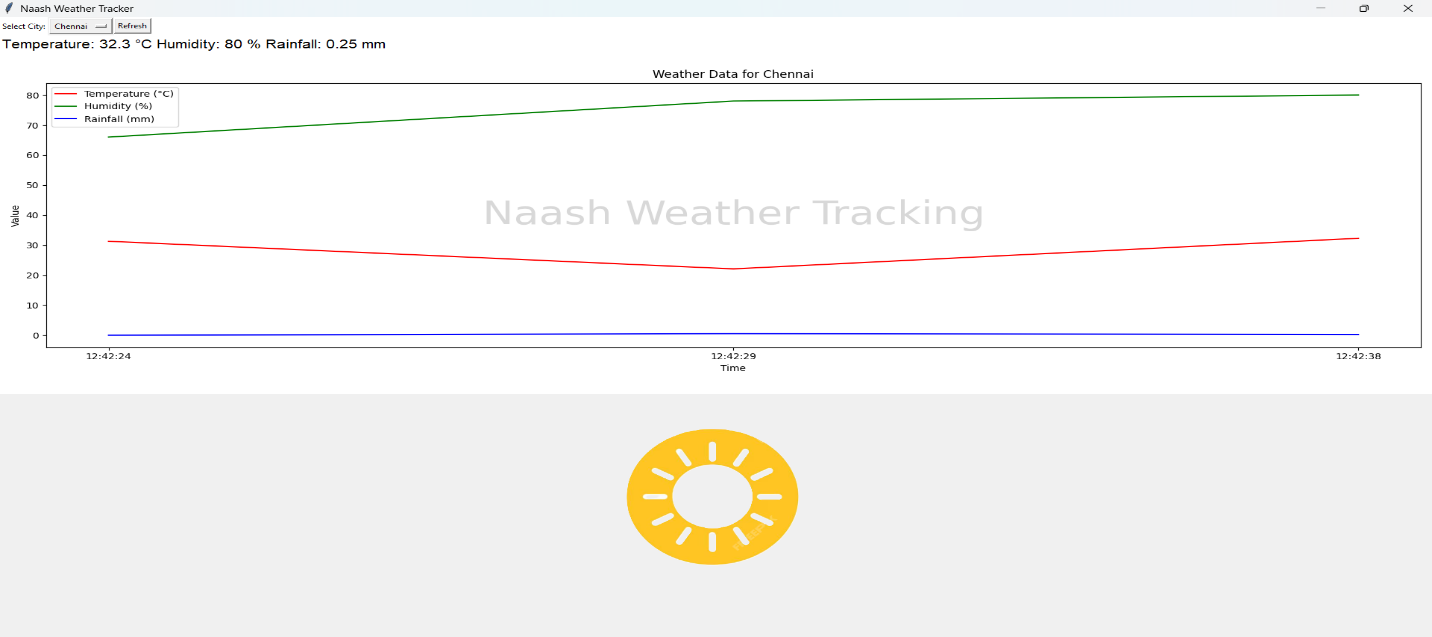
**7. Limitations:**

* The script only supports fetching weather data by city name (no coordinates yet).
* Error handling could be enhanced (e.g., catching network issues, handling API key errors).
* It requires an active internet connection.

**8. Potential Improvements:**

* Extend functionality to allow input of geographic coordinates (latitude/longitude).
* Add a graphical user interface (GUI) for a better user experience.
* Include caching to minimize redundant API requests

**9. Code Sample Output:**



Example:

Enter the city name: Chennai

Temperature: 32.3°C

Weather Condition: Rainfall

Humidity: 80%

Wind Speed: 3.5 m/s

**Conclusion:**

This real-time weather monitoring system fetches weather data from OpenWeatherMap, processes it, and displays current weather information based on the user input. This basic implementation can be extended for more advanced use cases, such as forecast data, multi-location queries, or additional weather parameters.