**Assignment: Python Programming for GUI Development**

Name :K.Mathivadhani

Register Number :192311424

Department :C.S.E

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.

**INTRODUCTION:**

The Real-Time Weather Monitoring System continuously collects and analyzes meteorological data to provide up-to-date weather information and alerts. It supports decision-making and safety by delivering real-time insights into current and forecasted weather conditions

**Features**

* Weather forecasting
* Alerts and notifications for severe weather

**Sensors and Instruments**

* **Weather Stations**, **Radar Systems**, **Satellite Systems Buoys ,Remote Sensors:**

Real-Time Weather Monitoring System

# 1.Data Flow Diagram



DEFAUL

PARA

F

RESET

PARAMETER

T

COLLECT

INCREMENT

MIDDLING

DISPLAY

INCREMENT

MIDDLING

DISPLAY THE

X-MR

CHART

HISTORIAL

DATA

STORAGE

HI HISTORY

DATA

ALARM

C CURRENT

DATA

ST

LIMIT OF

COLLECT

F F

T T T

ALARM

S STOP

F

T

# 2. Implementation

import requests

api\_key="0b563bad1d7970f0a7d1a2d766f968f2"

user\_input=input("enter the city name:")

weather\_data=requests.get(

   f"https://api.openweathermap.org/data/2.5/weather?q={user\_input}&units=imperial&APPID={api\_key}")

if weather\_data.json()["cod"]=="404":

    print("city not found")

else:

    #print(weather\_data.json())

    weather=weather\_data.json()["weather"][0]["main"]

    temp=round(weather\_data.json()["main"]["temp"])

    humidity=weather\_data.json()["main"]["humidity"]

    windspeed=weather\_data.json()["wind"]["speed"]

    country=weather\_data.json()["sys"]["country"]

    print(f"the weather in {user\_input} is {weather}")

    print(f"the temperture in {user\_input} is {temp}F")

    print(f"the humidity in {user\_input} is {humidity}")

    print(f"the windspeed in {user\_input} is {windspeed}kmph")

    print(f"the {user\_input} is in {country}")

# 3.Display the Current weather information

enter the city:bengaluru

 Temperature (in kelvin unit) = 295.37

 atmospheric pressure (in hPa unit) = 1002

 humidity (in percentage) =80

 description = overcast clouds

# 4.User Input



**5.Documentation**

**ALGORITHM:**

**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**

**Identify and configure weather sensors to monitor parameters like temperature and humidity, and set up data transmission protocols.**

**Collect real-time weather data from these sensors at regular intervals, including timestamps.**

**Transmit the collected data to a central processing unit.**

**Integrate and analyze the data to generate forecasts and detect anomalies.**

**Update user interfaces with real-time weather information and issue alerts for severe conditions.**

**Continuously monitor system performance, perform maintenance, and calibrate sensors as needed.**

# HOW HISTORICAL DATA INFLUENCES DECISION ON REAL-TIME WEATHER MONITORING SYSTEM

# Impact of Historical Data: Historical weather data helps enhance real-time weather monitoring by providing context for current conditions and improving forecast accuracy (e.g., refining hurricane predictions based on past storms). By comparing historical data with current trends, the system can better identify unusual weather patterns and predict extreme events (e.g., analyzing past heatwaves to forecast current temperature anomalies). Historical data also supports calibration of sensors and models, ensuring accurate real-time data (e.g., adjusting sensors based on historical temperature records). Additionally, it informs emergency response strategies by evaluating past weather impacts and improving preparedness for future events (e.g., using data from previous floods to enhance flood risk assessments).

**CONCLUSION:**

**In conclusion, historical weather data is essential for optimizing real-time weather monitoring systems. By providing valuable context and improving forecast accuracy, it allows for better predictions of extreme weather events, such as hurricanes and heatwaves. Historical data enhances the identification of unusual weather patterns and supports accurate calibration of sensors and models, ensuring reliable real-time information. Additionally, it informs emergency response strategies by evaluating past weather impacts, helping to improve preparedness and mitigate risks. Overall, integrating historical data into real-time monitoring systems enhances decision-making, public safety, and the effectiveness of weather-related interventions.**

# 

# Top of Form

# Bottom of Form