ASSIGNMENT 1

QUESTION : 1

**. Data Flow Diagram**

The data flow diagram (DFD) for the weather monitoring system outlines the interaction between the user, the weather API, and the application. Here's a simplified version

+---------------------+ +-------------------+ +----------------+

| | | | | |

| User | | Application | | Weather |

| (inputs location) +----------------->| (Python App) +<-----------------| API |

| | | | (Fetches data | (e.g., |

| | | | from API) | OpenWeatherMap)|

+---------------------+ +-------------------+ +----------------+

|

|

v

+--------+

| |

| JSON |

| Data |

+--------+

|

|

v

+--------+

| |

| Display|

| Weather|

| Data |

|  |
| --- |
|  |

# Pseudocode for Real-Time Weather Monitoring System

# Import necessary libraries

import requests

# Function to fetch weather data

def fetch\_weather\_data(location):

# Define the base URL and API key

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

api\_key = "your\_api\_key\_here" # Replace with your API key

# Construct the request URL

params = {

'q': location,

'appid': api\_key,

'units': 'metric' # Use 'imperial' for Fahrenheit

}

# Make the API request

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

# Check if the request was successful

if response.status\_code == 200:

# Parse the JSON data

weather\_data = response.json()

return weather\_data

else:

return None

# Function to display weather data

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

if weather\_data:

# Extract relevant information

temperature = weather\_data['main']['temp']

weather\_conditions = weather\_data['weather'][0]['description']

humidity = weather\_data['main']['humidity']

wind\_speed = weather\_data['wind']['speed']

# Display the information

print(f"Temperature: {temperature}°C")

print(f"Weather: {weather\_conditions}")

print(f"Humidity: {humidity}%")

print(f"Wind Speed: {wind\_speed} m/s")

else:

print("Could not retrieve weather data. Please check the location.")

# Main function to run the application

def main():

# Get user input for location

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

# Fetch weather data

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

# Display the weather data

display\_weather\_data(weather\_data)

# Run the main function

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

main()

CODE :

import requests

def fetch\_weather\_data(location):

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

api\_key = "your\_api\_key\_here" # Replace with your actual OpenWeatherMap API key

params = {

'q': location,

'appid': api\_key,

'units': 'metric'

}

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

if response.status\_code == 200:

weather\_data = response.json()

return weather\_data

else:

return None

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

if weather\_data:

temperature = weather\_data['main']['temp']

weather\_conditions = weather\_data['weather'][0]['description']

humidity = weather\_data['main']['humidity']

wind\_speed = weather\_data['wind']['speed']

print(f"Temperature: {temperature}°C")

print(f"Weather: {weather\_conditions}")

print(f"Humidity: {humidity}%")

print(f"Wind Speed: {wind\_speed} m/s")

else:

print("Could not retrieve weather data. Please check the location.")

def main():

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

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

display\_weather\_data(weather\_data)

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

main()

**4. Documentation and Explanation**

**API Integration:**

* **API Used:** The application integrates with the OpenWeatherMap API, which provides real-time weather data for any location.
* **Request Parameters:** The **q** parameter specifies the location (city name or coordinates), and the **appid** parameter is your unique API key. The **units** parameter is set to 'metric' for temperature in Celsius.
* **Response Handling:** The API returns data in JSON format, which is parsed by the application to extract temperature, weather conditions, humidity, and wind speed.

**Assumptions Made:**

* The user will input a valid city name or coordinates.
* The API key is valid and has the necessary permissions for API requests.

**Potential Improvements:**

* **Error Handling:** More robust error handling could be implemented to manage invalid inputs, API errors, or network issues.
* **UI Enhancement:** The application could be enhanced with a graphical user interface (GUI) for better user experience.
* **Extended Features:** Additional features like weather forecasting, alerts for extreme weather conditions, and historical data could be added.

This implementation provides a basic yet functional real-time weather monitoring system, with room for further enhancements and scalability.

OUT PUT :