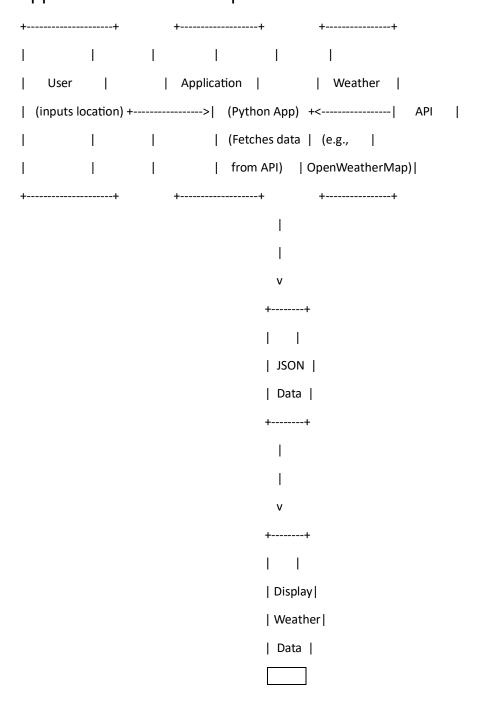
ASSIGNMENT 1

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



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:
```

```
# 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: