****

**Superior University Lahore**

***Lab Task # 7 & 8***

**Name:** Muhammad Abdullah Kashif

**Roll No**: SU92-BSAIM-F23-070

**Instructor**: Sir Rasikh Ali

# Course: Programming for Artificial Intelligence (Lab)

**Weather Web App Using Flask and OpenWeatherMap API**

**Step-by-Step Explanation:**

**1. Objective**

This project builds a simple weather application using Flask. Users can enter a city name to get real-time weather information including temperature, wind speed, and description using the OpenWeatherMap API.

**2. Virtual Environment Setup (venv)**

To isolate dependencies:

**Step 1: Create a Virtual Environment**

python -m venv weather\_env

**Step 2: Activate the Virtual Environment**

weather\_env\Scripts\activate

**Step 3: Install Required Packages**

pip install flask requests

**3. Import Required Libraries**

from flask import Flask, render\_template, request

import requests

* flask: To create the web app.
* render\_template: To render HTML pages.
* request: To handle form data.
* requests: To call the external weather API.

**4. Create the Flask App**

app = Flask(\_\_name\_\_)

Initializes the Flask application instance.

**5. OpenWeatherMap API Key**

API\_KEY = "ca6be27f7171af261db33fe5d49fa9fb"

* You need to register on <https://openweathermap.org> to get this key.
* This key is used to authenticate requests to the weather API.

**6. Define the Route and Logic**

@app.route('/', methods=['GET', 'POST'])

def weather():

* Sets up a route (/) that supports both GET and POST methods.
* Handles form submission and displays weather data.

**7. Initialize Variables**

weather\_data = None

error = None

Used to store the result and any error message.

**8. Handle POST Request and Call API**

if request.method == 'POST':

city = request.form.get('city')

url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric"

response = requests.get(url)

* Takes the city name from the form.
* Constructs the API URL with city name, API key, and metric units.
* Sends a GET request to the API.

**9. Debug Prints**

print(f"[DEBUG] API Response Code: {response.status\_code}")

print(f"[DEBUG] API Response Body: {response.text}")

* Helps in debugging if the API call fails or returns unexpected data.

**10. Parse the Response**

if response.ok:

data = response.json()

weather\_data = {

"city": data['name'],

"temperature": f"{data['main']['temp']} °C",

"wind": f"{data['wind']['speed']} m/s",

"description": data['weather'][0]['description'].title()

}

* Parses the JSON response and extracts:
  + City name
  + Temperature (°C)
  + Wind speed (m/s)
  + Weather description

**11. Handle API Errors**

else:

error = f"Error: {response.json().get('message', 'Something went wrong.')}"

* If response is not OK, extract error message from the API and show it on the page.

**12. Render the HTML Page**

return render\_template('index.html', weather=weather\_data, error=error)

* Sends weather\_data and error to the index.html page for display.

**13. Run the Flask App**

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* Starts the development server with debug mode enabled.

**14. index.html Example (Template File)**

<!DOCTYPE html>

<html>

<head>

<title>Weather App</title>

</head>

<body>

<h1>Check Weather</h1>

<form method="post">

<input type="text" name="city" placeholder="Enter city name" required>

<button type="submit">Get Weather</button>

</form>

{% if weather %}

<h2>Weather in {{ weather.city }}</h2>

<p>Temperature: {{ weather.temperature }}</p>

<p>Wind: {{ weather.wind }}</p>

<p>Description: {{ weather.description }}</p>

{% endif %}

{% if error %}

<p style="color: red;">{{ error }}</p>

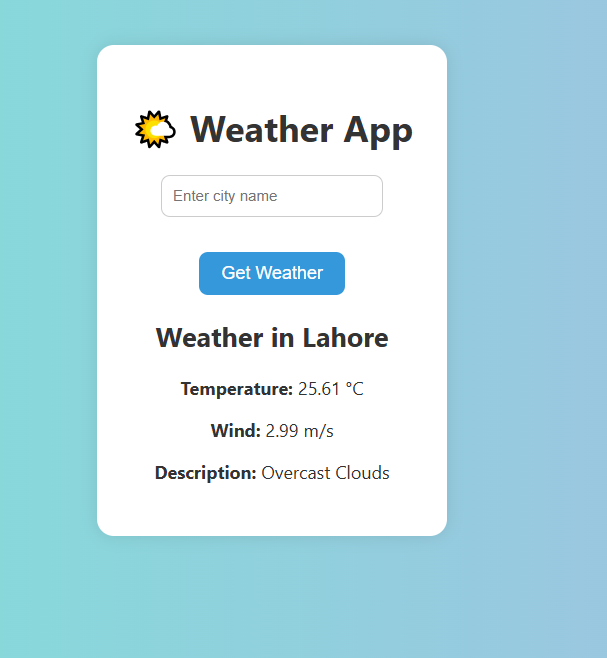
{% endif %}

</body>

</html>

**Output:**

****

****