**THE SUPERIOR UNIVERSITY LAHORE** 

**LAB#7**

**Semester: 4th Se~~ctio~~n: AI (B)**

**Faculty of Computer Science and Information Technology Deadline:**

**Subject: PAI LAB Total Marks:**

**Name: M.Hassan Shahid**

**Roll No: 059**

***Instructions:***

* Copying of the assignment willresult in failure.
* Assignment should be submitted in word or pdf.

# Building the Backend of a Weather App with Flask

## Introduction­

Weather apps are a part of our everyday lives, providing quick insights into the current weather conditions so we can plan our day accordingly. In this project, we built the backend of a Weather App using Flask—a lightweight and easy-to-use Python framework. The backend fetches real-time weather information by communicating with the OpenWeatherMap API based on the city provided by the user.

## Tools and Technologies

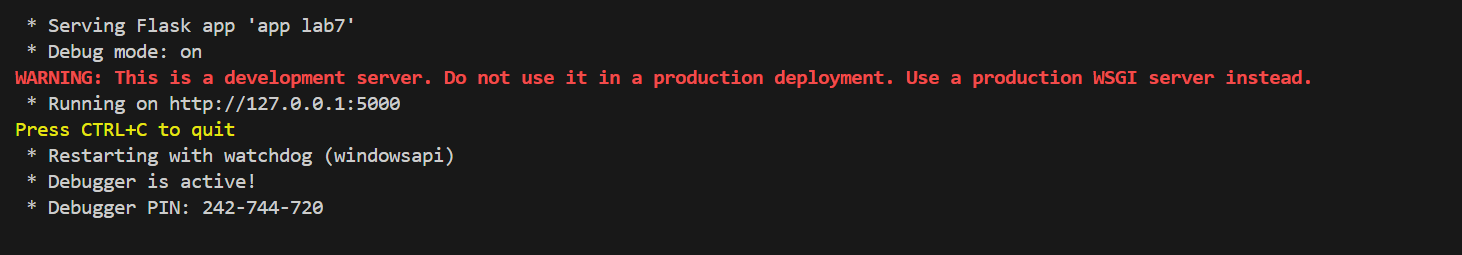
* **Flask**: A micro web framework for Python that helps you build web applications quickly.
* **Python**: The programming language used to create the backend, known for its readability and ease of use.
* **OpenWeatherMap API**: A service that offers current weather data, which our app accesses using a free API key.
* **Requests Library**: A Python package that makes it simple to send HTTP requests and receive responses.

## Full Backend Code

Below is the complete code for the backend of the Weather App. This code is organized into separate code blocks without inline comments.

## Code Explanation

* **Library Imports and App Initialization:**  
  The code begins by importing necessary libraries, setting up the Flask application, and defining the API key and the base URL for the OpenWeatherMap API.
* **Home Route:**  
  The root route (/) is defined to render an HTML template (index.html). Although this is part of the front-end, it ensures that the application has a landing page.
* **Weather Endpoint:**  
  The /weather endpoint is designed to accept a city name as a query parameter. If the city is not provided, the server returns an error. Once the city is received, the backend constructs a request to the OpenWeatherMap API using the provided city name, the API key, and metric units.
* **Response Handling:**  
  The backend checks the API response status. If the request is unsuccessful, an error message is returned. Otherwise, it parses the JSON response, extracts key pieces of weather data (such as the city name, temperature, weather description, and humidity), and returns this information in JSON format.
* **Running the Server:**  
  Finally, the server is started in debug mode, which makes it easier to troubleshoot and see real-time changes during development.



## Conclusion

In this project, we built a simple and effective backend for a Weather App using Flask. The backend efficiently fetches real-time weather data from the OpenWeatherMap API and returns it in a structured JSON format. This approach offers a solid foundation for further enhancements, such as expanding functionality, refining error handling, or developing a more interactive front-end interface.

This report demonstrates how to use Flask and Python to build a robust backend that serves as a crucial part of a web application. Enjoy building and expanding on your Weather App!