Real-time/Field-Based Research Project Report

On

PyWeather

A dissertation submitted to the Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirement for real-time project

BACHELOR OF TECHNOLOGY
IN
CSE-(DATA SCIENCE)

Submitted by

K.NITHIN REDDY	**(225U1A6723)**
K.SANDEEP	**(225U1A6724)**
K.ROHITH	**(225U1A6725)**
K.PAVANADITHYA	**(225U1A6727)**

SUBMITTED TO:

B.MAJULA
Project Co-Ordinator (CSE-DATA SCIENCE)

Department of CSE- (DATA SCIENCE)
AVN INSTITUTE OF ENGINEERING AND TECHNOLOGY
Ramdaspally, Ibrahimpatanam
2022-2026

AVN INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF CSE-(DATA SCIENCE)

CERTIFICATE

This is to certify that the project work entitled "**PyWeather**" is being submitted by **K. NITHIN REDDY (225U1A6723), K. SANDEEP (225U1A6724) K. ROHITH (225U1A6725), K. PAVANADITHYA (225U1A6727)** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering,** during the academic year 2023-2024.

B. MANJULA Madam
Project Coordinator

D.REVATHI Madam
HOD

DECLARATION

We hereby declare that this project report titled "**PyWeather**" submitted to the Department of Computer Science and Engineering (Data Science), **AVN Institute of Engineering and Technology**, is a record of original work done by us. The information and data given in the report is authentic to the best of our knowledge. This RFP report is not submitted to any other university or institution for the award of any degree or diploma or published at any time before.

K. NITHIN REDDY	**(225U1A6723)**
K. SANDEEP	**(225U1A6724)**
K. ROHITH	**(225U1A6725)**
K. PAVANADITHYA	**(225U1A6727)**

ABSTRACT

PyWeather is a dynamic web application designed to provide users with real-time weather information for any city in the world. Leveraging the OpenWeatherMap API, PyWeather fetches and displays current weather conditions including temperature, pressure, humidity, and descriptive weather phenomena such as thunderstorms, drizzle, rain, snow, and atmospheric conditions. The application's interface adapts visually to various weather conditions, providing users with an intuitive and engaging experience. By integrating these features, PyWeather aims to enhance user interaction through a visually appealing, easy-to-use platform that makes accessing weather data both informative and enjoyable. This project highlights the power of modern web technologies and APIs in creating responsive, user-centric applications that cater to everyday needs.

INTRODUCTION

In today's fast-paced world, timely and accurate weather information is crucial for daily planning and decision-making. Whether preparing for a commute, planning outdoor activities, or ensuring safety during extreme weather events, access to reliable weather data is essential. The PyWeather app is developed to address this need by providing users with up-to-date weather information in a visually engaging and easily accessible format.

PyWeather utilizes the OpenWeatherMap API, a comprehensive and widely-used weather data provider, to gather real-time weather data. This data includes essential metrics such as temperature, atmospheric pressure, humidity, and a detailed description of the weather conditions. The app's user interface is designed to be intuitive and responsive, ensuring that users can quickly obtain the information they need with minimal effort.

One of the standout features of PyWeather is its dynamic background color changes, which reflect the current weather conditions. This visual feedback enhances the user experience by providing an immediate, at-a-glance understanding of the weather. For example, a blue background indicates clear skies, while shades of gray and blue represent cloudy and rainy conditions, respectively. This feature not only makes the app visually appealing but also helps users quickly interpret the weather conditions without delving into numerical data.

The development of PyWeather involved the integration of modern web development technologies, including HTML, CSS, and JavaScript. The backend logic ensures seamless communication with the OpenWeatherMap API, processing and rendering the data in real-time. This project demonstrates the effective use of APIs in building functional and user-friendly applications, highlighting the importance of responsive design in enhancing user experience.

In conclusion, PyWeather is a testament to the capabilities of contemporary web technologies in creating practical, user-centric applications. By providing real-time weather information in a visually engaging format, PyWeather aims to simplify the way users interact with weather data, making it an indispensable tool for daily life.

REQUIREMENTS

Software Requirements

Backend

1. **Python (Version 3.6+)**
   - The core programming language for backend development.
2. **Flask**
   - A lightweight WSGI web application framework for Python.
   - Install via pip: `pip install Flask`
3. **Requests**
   - A simple HTTP library for Python to handle API requests.
   - Install via pip: `pip install requests`
4. **Jinja2**
   - The template engine for rendering HTML.
   - It comes bundled with Flask.

Frontend

1. **HTML5**
   - For structuring the web page.
2. **CSS3**
   - For styling the web page.
3. **JavaScript (optional)**
   - For additional interactivity (e.g., form validation).

API

1. **OpenWeatherMap API Key**
   - To fetch real-time weather data.
   - Sign up and get your API key from OpenWeatherMap.

Hardware Requirements

1. **Development Machine**
   - A computer with at least:
     - 4GB RAM
     - 2 GHz dual-core processor
     - 10GB free disk space
   - Operating System: Windows, macOS, or Linux.

Purpose

The primary purpose of the PyWeather app is to provide users with real-time, accurate weather information for any city worldwide. By leveraging the OpenWeatherMap API, the app fetches current weather conditions, including temperature, pressure, humidity, and descriptive weather phenomena. The app aims to offer an intuitive and engaging user experience by dynamically adapting the visual presentation based on different weather conditions.

Use Cases

1. Daily Planning

- Users can check the weather forecast to plan their day, including commuting, outdoor activities, and travel.

2. Safety and Preparedness

- The app provides critical information during severe weather conditions like thunderstorms, heavy rain, or snow, helping users prepare accordingly.

3.Travel Planning

- Travelers can use the app to check the weather at their destination and make informed decisions about their trips.

4. Event Planning

- Organizers can ensure the success of outdoor events by monitoring weather conditions in advance.

5. General Awareness

- The app helps users stay informed about the weather in their location and other cities of interest.

Advantages

1. Real-Time Weather Information

- PyWeather provides up-to-date weather data, ensuring users always have access to the latest information.

2.Dynamic Visual Feedback

- The app's background color changes based on the weather conditions, offering a quick visual cue for the current weather. This feature enhances user experience and makes the app more engaging.

3. User-Friendly Interface

- The simple and intuitive design ensures that users can easily navigate the app and access the information they need without any hassle.

4.Wide Coverage

- By leveraging the OpenWeatherMap API, PyWeather can provide weather data for virtually any city in the world, making it useful for a global audience.

5.Easy Integration and Deployment

- The app is built using popular web technologies like Python, Flask, HTML, and CSS, making it easy to deploy and integrate with other systems.

6. Educational Tool

- PyWeather can serve as an educational resource for learning about weather patterns and meteorology, making it useful for students and weather enthusiasts.

7. Customization

- The codebase allows for easy customization and scaling. Developers can add new features, modify the user interface, and integrate additional APIs as needed.

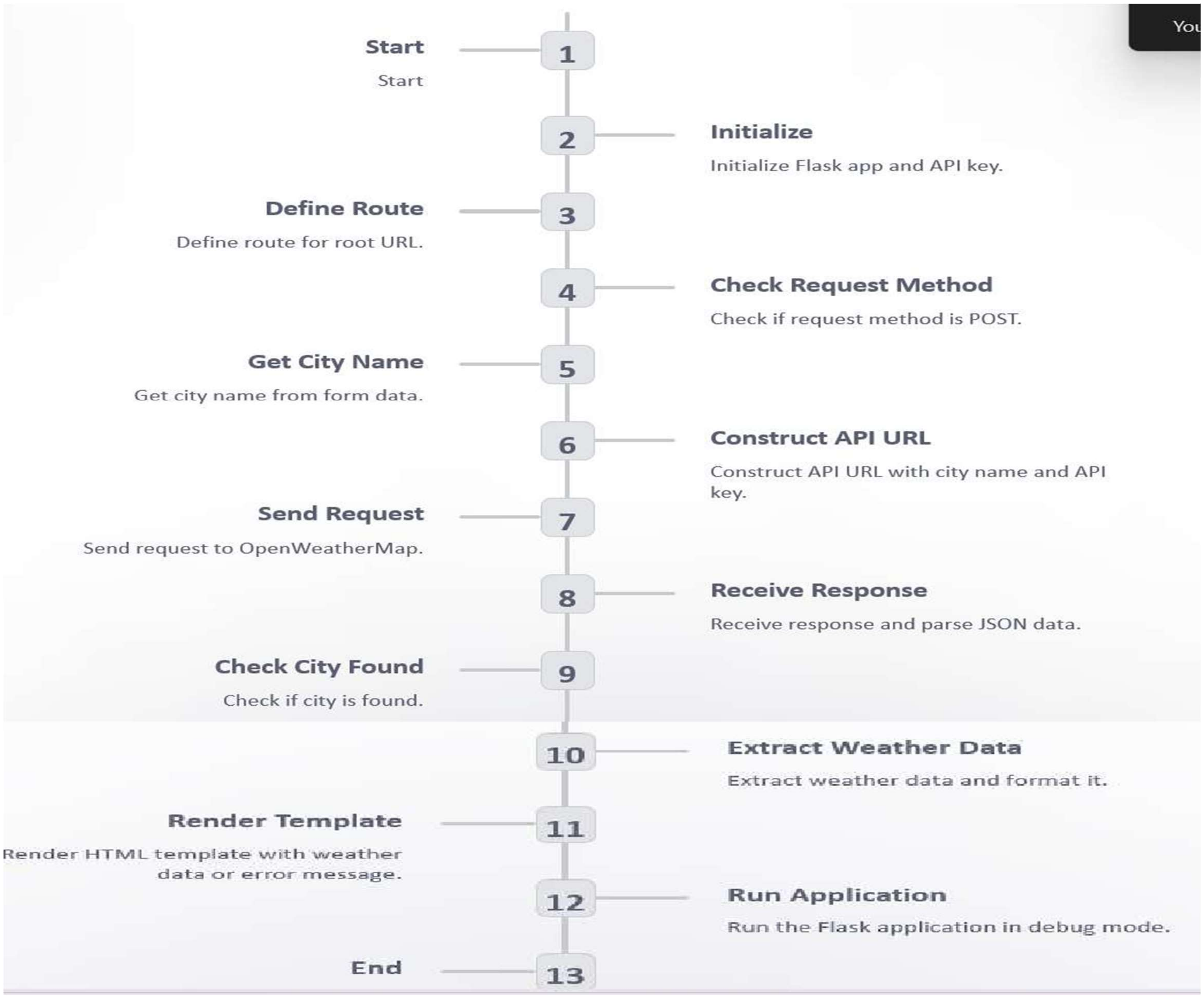
8. Cost-Effective

- Using the OpenWeatherMap free tier for basic weather data makes PyWeather a cost-effective solution for accessing weather information.

9. Accessibility

- The app is accessible from any device with a web browser, making it convenient for users to check the weather on the go.

ALGORITHM



1 **Start**
Start

2 **Initialize**
Initialize Flask app and API key.

3 **Define Route**
Define route for root URL.

4 **Check Request Method**
Check if request method is POST.

5 **Get City Name**
Get city name from form data.

6 **Construct API URL**
Construct API URL with city name and API key.

7 **Send Request**
Send request to OpenWeatherMap.

8 **Receive Response**
Receive response and parse JSON data.

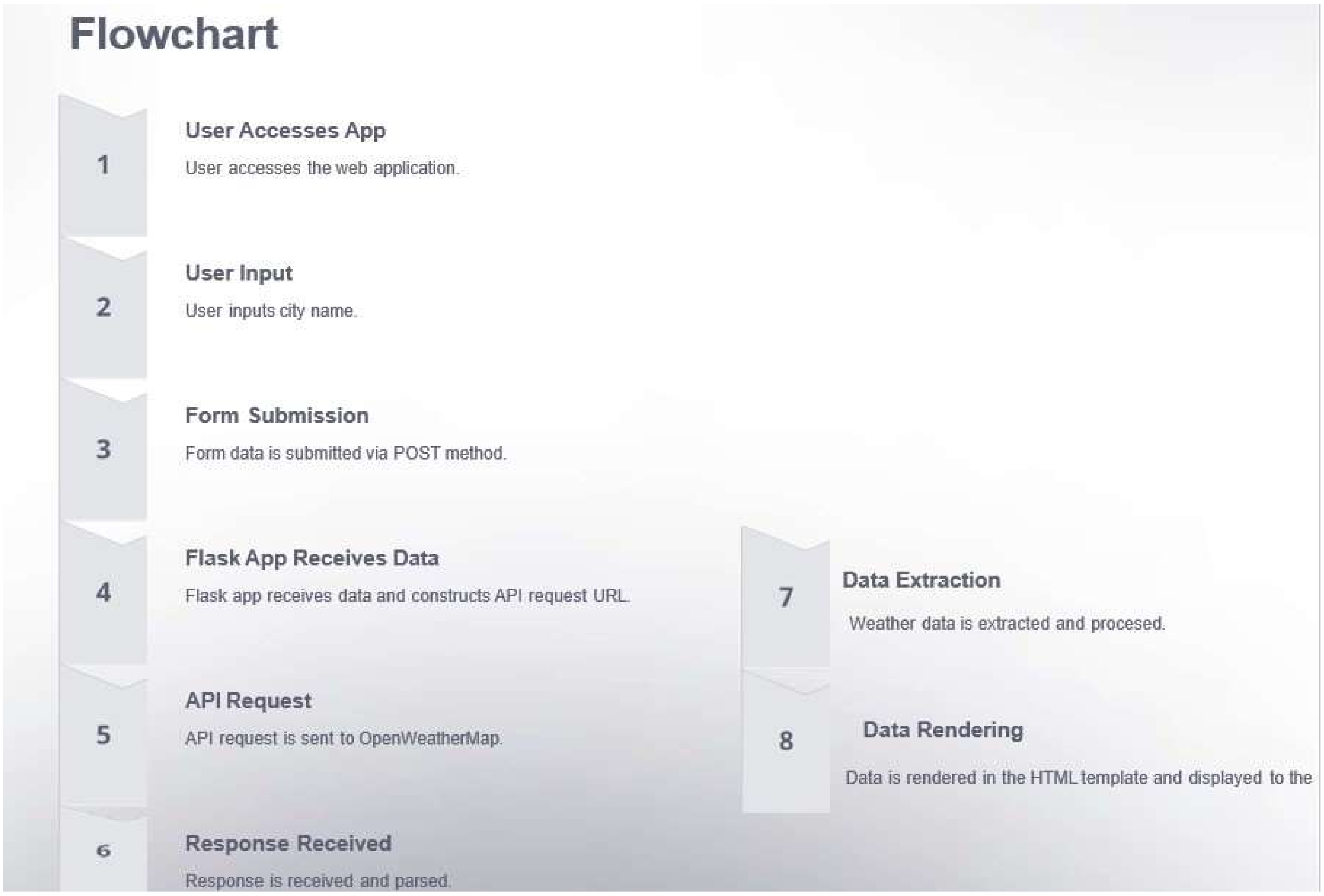
9 **Check City Found**
Check if city is found.

10 **Extract Weather Data**
Extract weather data and format it.

11 **Render Template**
Render HTML template with weather data or error message.

12 **Run Application**
Run the Flask application in debug mode.

13 **End**


Flowchart
1
User Accesses App
User accesses the web application.
2
User Input
User inputs city name.
3
Form Submission
Form data is submitted via POST method.
4
Flask App Receives Data
Flask app receives data and constructs API request URL.
5
API Request
API request is sent to OpenWeatherMap.
6
Response Received
Response is received and parsed.
7
Data Extraction
Weather data is extracted and procesed.
8
Data Rendering
Data is rendered in the HTML template and displayed to the

1. **User Accesses App**: The user opens the web application.
2. **User Input**: The user enters the name of a city.
3. **Form Submission**: The entered city name is submitted via a POST method.
4. **Flask App Receives Data**: The Flask application receives the submitted data and constructs an API request URL.
5. **API Request**: The Flask app sends an API request to OpenWeatherMap (or another weather API service).
6. **Response Received**: The application receives the response from the weather API and parses the data.
7. **Data Extraction**: The weather data is extracted and processed from the received response.
8. **Data Rendering**: The processed data is rendered in the HTML template and displayed to the user.

Code

1.app.py

```
from flask import Flask, render_template, request
import requests

app = Flask(__name__)

api_key = "30d4741c779ba94c470ca1f63045390a"
base_url = "http://api.openweathermap.org/data/2.5/weather?"

@app.route('/', methods=['GET', 'POST'])
def index():
weather_data = None
if request.method == 'POST':
city_name = request.form['city']
complete_url = f"{base_url}appid={api_key}&q={city_name}"
response = requests.get(complete_url)
data = response.json()
if data.get("cod") != "404":
     main_data = data["main"]
     weather = data["weather"][0]
weather_data = {
"temperature": f"{main_data['temp']} ",
"pressure": f"{main_data['pressure']}",
"humidity": f"{main_data['humidity']}",
"description": weather["description"].upper(),
"icon": weather["icon"],
"city": city_name.upper()
  }
else:
 weather_data = {"error": "CITY NOT FOUND"}  # Set error message when city is not found

return render_template('index.html', weather=weather_data)

if __name__ == "__main__":
  app.run(debug=True)
```

2.index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pyweather</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }

        .container {
            max-width: 500px;
            margin: 50px auto;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            {% if weather %}
                {% if weather.error %}
                    background-color: #f4f4f4; /* Default background color */
                {% elif weather.code in [200, 201, 202, 210, 211, 212, 221, 230, 231, 232] %}
                    background-color: #778899; /* Light Slate Gray for Thunderstorm */
                {% elif weather.code in [300, 301, 302, 310, 311, 312, 313, 314, 321] %}
                    background-color: #b0c4de; /* Light Blue for Drizzle */
                {% elif weather.code in [500, 501, 502, 503, 504, 511, 520, 521, 522, 531] %}
                    background-color: #4682b4; /* Steel Blue for Rain */
                {% elif weather.code in [600, 601, 602, 611, 612, 613, 615, 616, 620, 621, 622]
%}
                    background-color: #fffafa; /* Snow color */
                {% elif weather.code == 701 %}
                    background-color: #e0e0e0; /* Light gray for Mist */
                {% elif weather.code == 711 %}
                    background-color: #696969; /* Dim gray for Smoke */
                {% elif weather.code == 721 %}
                    background-color: #dcdcdc; /* Gainsboro for Haze */
                {% elif weather.code in [731, 751] %}
                    background-color: #edc9af; /* Desert sand for Sand/Dust whirls */
                {% elif weather.code == 741 %}
                    background-color: #c0c0c0; /* Silver for Fog */
                {% elif weather.code in [761, 762] %}
                    background-color: #bdb76b; /* Dark khaki for Volcanic ash */
```

```
{% elif weather.code == 771 %}
                    background-color: #778899; /* Light slate gray for Squalls */
                {% elif weather.code == 781 %}
                    background-color: #808080; /* Gray for Tornado */
                {% elif weather.code == 800 %}
                    background-color: #87ceeb; /* Sky Blue for Clear sky */
                {% elif weather.code == 801 %}
                    background-color: #f0f8ff; /* Alice Blue for Few clouds */
                {% elif weather.code == 802 %}
                    background-color: #b0c4de; /* Light Blue for Scattered clouds */
                {% elif weather.code == 803 %}
                    background-color: #d3d3d3; /* Light Gray for Broken clouds */
                {% elif weather.code == 804 %}
                    background-color: #a9a9a9; /* Dark Gray for Overcast clouds */
                {% else %}
                    background-color: #f4f4f4; /* Default background color */
                {% endif %}
            {% else %}
                background-color: #f4f4f4; /* Default background color */
            {% endif %}
            color: white; /* Adjust text color for better visibility */
            padding: 20px;
        }

        h1 {
            margin-top: 0;
            font-size: 28px;
            color: #333;
        }

        form {
            margin-bottom: 20px;
        }

        input[type="text"] {
            width: calc(100% - 22px);
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        input[type="submit"] {
            width: 100%;
            padding: 10px;
            background-color: #007bff;
            color: #fff;
```

```
             border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .weather-info {
            margin-top: 20px;
            text-align: left;
        }

        .weather-info p {
            margin: 10px 0;
            font-size: 18px;
            color: #333;
        }

        .error {
            color: red;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>PyWeather</h1>
        <form method="POST">
            <input type="text" name="city" placeholder="Enter city name"
required>
            <input type="submit" value="Get Weather">
        </form>
        {% if weather %}
            {% if weather.error %}
                <p class="error">{{ weather.error }}</p>
            {% else %}
                <div class="weather-info">
                    <p>City: {{ weather.city | upper }}</p>
                    <p>Temperature: {{ weather.temperature }} °K</p>
                    <p>Pressure: {{ weather.pressure }} hPa</p>
                    <p>Humidity: {{ weather.humidity }} %</p>
                    <p>Description: {{ weather.description | upper }}</p>
                </div>
            {% endif %}
        {% endif %}
    </div>
</body>
</html>
```

OUTPUT

CONCLUSION

The PyWeather app stands as a testament to the seamless integration of modern web technologies with practical, everyday applications. By harnessing the power of the OpenWeatherMap API, PyWeather delivers real-time, accurate weather information through a user-friendly and visually engaging interface. One of the standout features of PyWeather is its dynamic background color adaptation based on current weather conditions, which not only enhances the user experience but also provides a quick and intuitive way for users to grasp weather scenarios at a glance.

Through this Request for Proposal (RFP), we have outlined the purpose, use cases, advantages, and detailed requirements of the PyWeather app. The app's ability to cater to a global audience, its ease of integration, and its cost-effectiveness make it a valuable tool for anyone needing reliable weather updates. The dynamic color-changing feature is particularly noteworthy, as it visually represents different weather conditions—such as blue for clear skies, gray for clouds, and darker shades for thunderstorms—ensuring that users can instantly understand the weather at a glance.

Whether for daily planning, safety preparedness, travel, or educational purposes, PyWeather offers a comprehensive solution that combines functionality with aesthetics. Its development exemplifies the potential of leveraging open APIs and modern web development practices to create applications that are not only useful but also enjoyable to use.

In conclusion, PyWeather is more than just a weather forecasting tool; it is an innovative application designed to simplify and enhance the way users interact with weather data. Its dynamic color-changing ability makes it both informative and visually appealing, ensuring that users have a pleasant and efficient experience. We believe that PyWeather will significantly benefit its users by providing timely and accurate weather information in a beautifully crafted format, making it an indispensable part of their daily lives.