App.py

from flask import Flask, render\_template, request

import requests

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

API\_KEY = "5f315e6583b9c250c7676ea21faa42b8"

def get\_background\_class(condition):

    """Return a CSS class based on the weather condition."""

    condition = condition.lower()

    if "clear" in condition:

        return "clear-sky"

    elif "rain" in condition:

        return "rainy"

    elif "snow" in condition:

        return "snowy"

    elif "cloud" in condition:

        return "cloudy"

    else:

        return "default-weather"

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

def index():

    weather\_data = None

    error\_message = None  # Variable for error messages

    background\_class = "default-weather"  # Default background class

    if request.method == "POST":

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

        if city:

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

            response = requests.get(url)

            if response.status\_code == 200:

                data = response.json()

                weather\_data = {

                    "city": data["name"],

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

                    "condition": data["weather"][0]["description"].capitalize(),

                    "humidity": f"{data['main']['humidity']}%",

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

                }

                background\_class = get\_background\_class(data["weather"][0]["description"])

            else:

                error\_message = "City not found. Please try again."

        else:

            error\_message = "Please enter a city name."

    return render\_template(

        "index.html",

        weather=weather\_data,

        error=error\_message,

        background\_class=background\_class

    )

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

    app.run(debug=True)

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="/static/styles.css">

    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css" rel="stylesheet">

    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet">

    <title>Weather App</title>

</head>

<body class="{{ background\_class }}">

    <div class="container">

        <aside class="sidebar">

            <h1 class="brand">

                <i class="fas fa-cloud-sun"></i> Weather Pro

            </h1>

            <nav>

                <ul>

                    <li><a href="#"><i class="fas fa-home"></i> Home</a></li>

                    <li><a href="#"><i class="fas fa-info-circle"></i> About</a></li>

                    <li><a href="#"><i class="fas fa-question-circle"></i> Help</a></li>

                    <li><a href="#"><i class="fas fa-phone-alt"></i> Contact</a></li>

                </ul>

            </nav>

            <footer>

                <p>Powered by OpenWeatherMap</p>

                <p>© 2025 WeatherPro</p>

            </footer>

        </aside>

        <main class="content">

            <h2>Check Your City Weather</h2>

            <p>Enter the name of a city to get the latest weather details.</p>

            <form action="/" method="POST">

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

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

            </form>

            <!-- Display Weather Data -->

            {% if weather %}

            <div class="weather-result">

                <h3>Weather in {{ weather.city }}</h3>

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

                <p><strong>Condition:</strong> {{ weather.condition }}</p>

                <p><strong>Humidity:</strong> {{ weather.humidity }}</p>

                <p><strong>Wind Speed:</strong> {{ weather.wind\_speed }}</p>

            </div>

            {% elif error %}

            <div class="error-message">

                <p>{{ error }}</p>

            </div>

            {% endif %}

        </main>

    </div>

</body>

</html>

Style.css

/\* General Styles \*/

body {

    font-family: 'Roboto', sans-serif;

    margin: 0;

    padding: 0;

    background: #f5f7fa;

    color: #333;

    transition: background 0.3s ease-in-out;

}

body.clear-sky {

    background: url('/static/clear-sky.jpg') no-repeat center center fixed;

    background-size: cover;

}

body.rainy {

    background: url('/static/rainy.jpg') no-repeat center center fixed;

    background-size: cover;

}

body.cloudy {

    background: url('/static/cloudy.jpg') no-repeat center center fixed;

    background-size: cover;

}

body.snowy {

    background: url('/static/snowy.jpg') no-repeat center center fixed;

    background-size: cover;

}

body.default-weather {

    background: url('/static/default.jpg') no-repeat center center fixed;

    background-size: cover;

}

.container {

    display: flex;

    height: 100vh;

}

/\* Sidebar Styles \*/

.sidebar {

    background-color: #2b3a3f;

    color: #f5f7fa;

    width: 260px;

    padding: 20px;

    display: flex;

    flex-direction: column;

    justify-content: space-between;

}

.brand {

    display: flex;

    align-items: center;

    gap: 10px;

    font-size: 1.6rem;

    font-weight: 700;

}

.brand-icon {

    width: 40px;

    height: 40px;

}

.sidebar nav ul {

    list-style: none;

    padding: 0;

}

.sidebar nav ul li {

    margin-bottom: 10px;

}

.nav-icon {

    width: 20px;

    height: 20px;

    margin-right: 10px;

    vertical-align: middle;

}

.sidebar nav ul li a {

    color: #a8c4c7;

    text-decoration: none;

    font-size: 1rem;

    gap: 5px;

    padding: 10px;

    display: flex;

    align-items: center;

    border-radius: 5px;

    transition: background 0.3s, color 0.3s;

}

.sidebar nav ul li a:hover,

.sidebar nav ul li a.active {

    background: #3d565a;

    color: #f5f7fa;

}

/\* Footer Styles \*/

.sidebar footer {

    font-size: 0.9rem;

    color: #a8c4c7;

    text-align: center;

}

/\* Main Content Styles \*/

.content {

    flex: 1;

    padding: 40px;

    background: rgba(255, 255, 255, 0.9);

    border-radius: 15px;

    margin: auto;

    width: 90%;

    max-width: 600px;

    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

.content h2 {

    font-size: 2rem;

    margin-bottom: 10px;

}

.content p {

    margin-bottom: 20px;

    font-size: 1rem;

    color: #555;

}

form {

    display: flex;

    gap: 10px;

}

form input {

    flex: 1;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

    font-size: 1rem;

}

form button {

    background-color: #ffa726;

    color: #fff;

    border: none;

    padding: 10px 20px;

    font-size: 1rem;

    border-radius: 5px;

    cursor: pointer;

    transition: background 0.3s;

}

form button:hover {

    background-color: #f57c00;

}

/\* Weather Result Styles \*/

.weather-result {

    margin-top: 20px;

    background: #ffffff;

    padding: 20px;

    border-radius: 8px;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.weather-result h3 {

    margin-bottom: 10px;

    font-size: 1.5rem;

}

.weather-result p {

    margin-bottom: 5px;

    font-size: 1rem;

}

.error-message {

    margin-top: 20px;

    color: #d9534f;

    font-size: 1rem;

}

**1. Enhanced User Experience**

* **Autocomplete for City Names:** Implement an autocomplete feature using APIs like Google Places or OpenWeatherMap's geocoding API.
* **Error Handling:** Display more user-friendly error messages when an invalid city is entered or if the API limit is exceeded.
* **Location Detection:** Add functionality to detect the user’s current location and fetch the weather automatically.

**2. Weather Forecast**

* **5-Day Forecast:** Display a 5-day or hourly weather forecast using OpenWeatherMap’s forecast API.
* **Graphical Representation:** Add graphs to display temperature, humidity, and wind speed trends over time.

**3. Data Visualization**

* **Charts and Graphs:** Use libraries like Chart.js or D3.js to visually represent historical weather data.
* **Climate Comparisons:** Allow users to compare weather conditions between multiple cities.

**4. Additional Features**

* **Save Favorite Cities:** Let users save a list of their favorite cities and quickly check their weather.
* **Weather Alerts:** Notify users of severe weather conditions like storms or heatwaves.
* **Multi-Language Support:** Add options for multiple languages for global accessibility.

**5. Integration with Other APIs**

* **Air Quality Index (AQI):** Include air quality data alongside weather information.
* **Map Integration:** Embed an interactive map showing weather patterns using Mapbox or Leaflet.
* **News API Integration:** Show weather-related news or alerts.

**6. Performance and Scalability**

* **Caching:** Implement caching for API responses to reduce API calls and improve performance.
* **PWA (Progressive Web App):** Make your app installable on mobile devices for an app-like experience.
* **Server-Side Rendering (SSR):** Optimize the app for better SEO and faster loading times.