

Weather App

Python Project

Sneha | Weather App | 18th Sept

Here’s a simplified plan:

**Technologies We'll Use:**

* **HTML**: Structure of the web page.
* **CSS**: Styling the web page.
* **Python**: Fetching weather data and handling logic.
* **SQLite**: Storing user search history (optional).

We’ll start with simple goals:

1. **Create a basic webpage** with HTML.
2. **Use Python** to fetch real-time weather data from a free weather API.
3. **Connect the webpage to Python** so users can enter a city name and see the weather forecast.

**Step 1: Setting Up Your Environment**

**Installing Python**

* First, make sure you have **Python** installed on your computer. You can download it from the official [Python website](https://www.python.org/downloads/).

**Installing the Required Libraries**

We’ll need a couple of Python libraries:

1. **Requests**: To fetch weather data from the OpenWeatherMap API.
2. **Flask**: To create a web server and handle the user input.

Once Python is installed, open a terminal (or command prompt) and install these libraries:

**pip install flask requests**

**Step 2: Create the Project Folder**

Let’s set up a simple folder structure for the project:

**mkdir weather\_app**

**cd weather\_app**

Inside this folder, we’ll create:

1. **Python script** (app.py) to handle the logic.
2. **HTML file** (index.html) to display the form and results.
3. **CSS file** (style.css) for styling the webpage.

**Step 3: Writing the Python Script**

This script will handle getting the weather data. We'll use the **OpenWeatherMap API** for this. First, you’ll need an API key, which is free.

**Get Your API Key:**

1. Go to OpenWeatherMap and sign up.
2. c
3. After registering, you’ll get an API key.

**Python Script (app.py)**

Now, let's create the **app.py** file in your project folder. This will:

* Accept the city name from the user.
* Fetch weather data from the API.
* Send that data back to the web page.

Python code

from flask import Flask, render\_template, request

import requests

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

# Replace 'your\_api\_key' with your actual OpenWeatherMap API key

API\_KEY = 'your\_api\_key'

# Route for the homepage

@app.route('/')

def index():

return render\_template('index.html')

# Route to handle the form submission and get the weather

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

def get\_weather():

city = request.form['city'] # Get the city from the form input

weather = fetch\_weather\_data(city)

if weather:

return render\_template('result.html', weather=weather)

else:

return render\_template('index.html', error="City not found!")

def fetch\_weather\_data(city):

# OpenWeatherMap API URL

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()

return {

'city': data['name'],

'temperature': data['main']['temp'],

'description': data['weather'][0]['description'],

'humidity': data['main']['humidity'],

'wind\_speed': data['wind']['speed']

}

else:

return None

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

app.run(debug=True)

**Step 4: Create the HTML and CSS Files**

We need a basic webpage where users can enter the city name. Let’s create that now.

**HTML Form (templates/index.html)**

This will display a simple form for the user to input a city name.

**Html code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Weather Forecast</title>

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

</head>

<body>

<div class="container">

<h1>Weather Forecast</h1>

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

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

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

</form>

{% if error %}

<p class="error">{{ error }}</p>

{% endif %}

</div>

</body>

</html>

**Display the Result (templates/result.html)**

This page will show the weather data after the user submits the city name.

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Weather Result</title>

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

</head>

<body>

<div class="container">

<h1>Weather in {{ weather['city'] }}</h1>

<p>Temperature: {{ weather['temperature'] }}°C</p>

<p>Condition: {{ weather['description'] }}</p>

<p>Humidity: {{ weather['humidity'] }}%</p>

<p>Wind Speed: {{ weather['wind\_speed'] }} m/s</p>

<a href="/">Search another city</a>

</div>

</body>

</html>

**CSS Styling (static/style.css)**

Let’s add some basic styling. Create a style.css file inside the static folder.

**Css code**

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

text-align: center;

margin: 0;

padding: 0;

}

.container {

margin-top: 50px;

}

input[type="text"] {

padding: 10px;

width: 250px;

}

button {

padding: 10px;

background-color: #28a745;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

.error {

color: red;

margin-top: 20px;

}

**Step 5: Run the Application**

Now, we’ll run the app. In your terminal, go to the weather\_app folder and type:

**python app.py**

Open a browser and go to http://127.0.0.1:5000/. You should see the form, and when you submit a city name, it will display the weather information.

**Next Steps**

* **Deployment**: Once you're happy with the app, we can deploy it online using free platforms like Heroku.
* **Error Handling**: Add some more error handling for invalid city names or network issues.