

120A3051

Shreya Idate

Batch: E3

Experiment 11

Aim: Design Weather app using Flask

Theory:

Flask is a lightweight framework written in Python. It is lightweight because it does not require particular tools or libraries and allow rapid web development. we will create a weather app using flask as a web framework. this weather web app will provide current weather updates of cities searched.

Basic setup :

- Create a file and name it as weather.py
- Create html file as weather.html

Get API Key

Goto https://home.openweathermap.org/users/sign_up and sign up for a free plan if you haven't already. Then visit the API key section https://home.openweathermap.org/api_keys to get your key. You can use the default key or generate a new one as you wish. You may have to wait for a while to get your key verified.

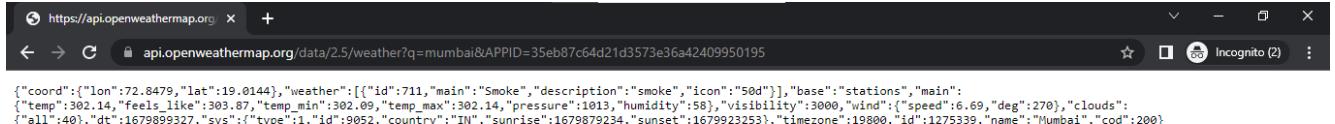
The screenshot shows the OpenWeatherMap API keys management interface. At the top, there's a navigation bar with links like Weather in your city, Guide, API, Dashboard, Marketplace, Pricing, Maps, Our Initiatives, Partners, Blog, For Business, shrey..., and Support. Below the navigation, a green message box says: "We have sent the confirmation link to shreya.idateit120@siesgst.ac.in. Please check your email." A horizontal menu bar below the message includes New Products, Services, API keys (which is highlighted in red), Billing plans, Payments, Block logs, My orders, My profile, and Ask a question. A sub-section message box states: "You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them." Below these sections is a table titled "API keys" with columns: Key, Name, Status, Actions, and a "Create key" button. One row is visible in the table, showing a key value (35eb87c64d21d3573e36a42409950195), a Name field containing "Default", a Status field showing "Active", and two small circular icons under Actions. To the right of the table are input fields for "API key name" and a "Generate" button.

Making API Calls

Going through the documentation, the simplest way to make requests is using the city name. The url to make request will look like this

http://api.openweathermap.org/data/2.5/weather?q=city_name&APPID=your_api_key

You can copy paste the url and replace the city_name and your_api_key with appropriate value to see the result.



The screenshot shows a browser window with the URL <https://api.openweathermap.org/data/2.5/weather?q=mumbai&APPID=35eb87c64d21d3573e36a42409950195>. The page displays a JSON object representing the weather in Mumbai. Key fields include coordinates, weather conditions (main: Smoke, icon: 50d), base: stations, main temperature (temp: 302.14), feels like (feels_like: 303.87), temp_min (temp_min: 302.09), temp_max (temp_max: 302.14), pressure (pressure: 1013), humidity (humidity: 58), visibility (visibility: 3000), wind (wind: speed: 6.69, deg: 270), clouds (all: 40), dt (dt: 1679899327), sys (type: 1, id: 9052, country: IN, sunrise: 1679879234, sunset: 1679923253), timezone (timezone: 19800), id (id: 1275339), name (name: Mumbai), and cod (cod: 200).

```
# Importing essential libraries
from flask import Flask, render_template, request
import requests
app = Flask(__name__)
def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
    :param: city_name
    :return: temperature, humidity
    """
    #api_key = config.weather_api_key
    api_key = "852a3b91d73420cf518eb8c685a509c0"
    base_url = "http://api.openweathermap.org/data/2.5/weather?"
    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]

        temperature = round((y["temp"] - 273.15), 2)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        return None

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/weather', methods=['POST'])
def weather():
    if request.method == 'POST':
        city_name = request.form['city']
        if weather_fetch(city_name) != None:
            temperature, humidity = weather_fetch(city_name)
            return render_template('index.html', weather="The temperature and Humidity in "+city_name+" is "+str(temperature)+" and "+str(humidity))
if __name__ == '__main__':
    app.run(debug=True)
```

Create weather.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Weather Details</title>
    <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='CSS/1.css')}}" />
  </head>
  <body>
    <h2>Weather Details</h2>
    <form action="{{ url_for('weather') }}" method="POST">
      <input type="text" name="city" placeholder="Enter the city name"><br><br>
      <input type="submit" value="Go">
    </form>
    <br><br>
    <div>
      <h2 style="color: orange;"> {{ weather }} </h2>
    </div>
  </body>
```

Run Flask Application using command python weather.py

Output:



Weather Details



Weather Details

The temperature and Humidity of the Pune is 30.32 celsius and 14%

Conclusion: Successfully designed a weather app using Flask.