Weather App Master

Overview

This project is a weather forecasting web application that allows users to search for a city and get current weather conditions and a 5-day weather forecast. The app is built using Python and Flask on the back-end and uses HTML, CSS, and Jinja templating for the front-end. The OpenWeatherMap API provides the weather data.

Features:

- **Search for a City:** Users can search for a city and get weather information.
- **Current Weather:** Displays the current temperature, weather conditions, wind speed, and minimum/maximum temperatures.
- **5-Day Forecast:** Provides temperature and weather conditions for the next five days.
- **Error Handling:** Displays an error message if the city is not found.

Project Structure

css
Copy code

```
.weather-web-app/
│
├── .idea/
├── screenshots/
├── static/
│   ├── assets/
│   └── css/
│       └── main.css
├── templates/
│   ├── city.html
│   ├── error.html
│   └── index.html
├── main.py
├── Procfile
├── README.md
└── requirements.txt
```

1. Main Python Application (`main.py`)

This is the core back-end file of the application that handles requests, connects to the OpenWeather API, processes data, and renders the respective templates.

Key Sections:

- **Home Page (`@app.route("/")`):**
  - Displays the home page where users can search for a city.
  - If a city is submitted, the form sends a POST request, which redirects to the weather page for the searched city.
- **Get Weather (`@app.route("/<city>")`):**
  - Fetches current weather data using OpenWeatherMap API based on the city entered by the user.
  - Extracts and displays key weather data such as temperature, weather condition, wind speed, etc.
  - Fetches a 5-day forecast using the OpenWeatherMap forecast API.
- **Error Handling (`@app.route("/error")`):**
  - Renders the error page if the city is not found or the API fails to return valid coordinates.

2. HTML Templates

- `index.html` (Home Page):
  - The user enters the name of a city they want to search.
  - It contains a search bar with a city search form.
- `city.html` (City Weather Page):
  - Displays the current weather for the city, including:
    - City name and current date.
    - Current temperature, weather condition, wind speed, min/max temperature.
  - A 5-day weather forecast is also shown with corresponding weather icons.
- `error.html` (Error Page):
  - Displays an error message and allows the user to navigate back to the home page to search for a new city.

3. CSS (`main.css`)

The styling file manages the layout and appearance of the application across devices.

Key Sections:

- **Reset Styles:** Resets browser default styles.
- **General Styles:** Defines typography and layout for elements such as `h1`, `h2`, `p`, and `a`.

- **Layout Styles:**
  - **Background:** Handles background images and styles.
  - **Search Bar:** Styles for the city search form.
  - **Weather Sections:** Layouts for displaying temperature, weather icons, and forecast information.
- **Media Queries:**
  - Optimises the layout for different screen sizes (laptops, tablets, phones).

4. Assets

- **Images:**
  - Various weather icons such as sunny, cloudy, rainy, etc., are stored in the `assets` folder.
  - Additional images are used for the background of the home and error pages.

5. External Dependencies

The `requirements.txt` contains all the necessary dependencies for the project. Key packages include:

- **Flask**: The web framework.
- **Requests**: To interact with the OpenWeatherMap API.
- **Gunicorn**: Used for deploying the application.

6. API Integration

The application integrates with the OpenWeatherMap API, which provides:

- **Current Weather Data**: `https://api.openweathermap.org/data/2.5/weather`
- **5-Day Forecast Data**: `https://api.openweathermap.org/data/2.5/forecast`

To connect with the API, the application requires an API key, which is stored in the `.env` file and accessed using the `python-dotenv` library.

Key API Parameters:

- `appid`: API key.
- `units`: Set to metric for temperature in Celsius.
- `q`: City name (for geocoding).
- `lat` and `lon`: Latitude and longitude (for fetching weather).

7. Error Handling

If the city name is invalid or not found, the app gracefully redirects to an error page. The template `error.html` contains a message indicating that the city does not exist, allowing the user to search again.

8. Procfile

The `Procfile` is used for deploying the application on platforms like Heroku. It specifies the web process:

makefile
Copy code
```
web: gunicorn main:app
```

9. How to Run the Application

To run the application on your local machine:

Install the required dependencies:
bash
Copy code
```
pip install -r requirements.txt
```

Create a `.env` file and add your OpenWeatherMap API key:
makefile
Copy code
```
OWM_API_KEY=your_api_key_here
```

Run the Flask app:
bash
Copy code
```
flask run
```

Access the application at `http://127.0.0.1:5000/`.

Future Enhancements:

- **Add More Weather Information:** Include humidity, sunrise/sunset times, and pressure data.
- **User Authentication:** Allow users to save their favorite cities.
- **Dark Mode Support:** Introduce a toggle for dark mode to improve accessibility.

Class Diagram (Simplified UML)