**🌦️ Weather App – Project Summary**

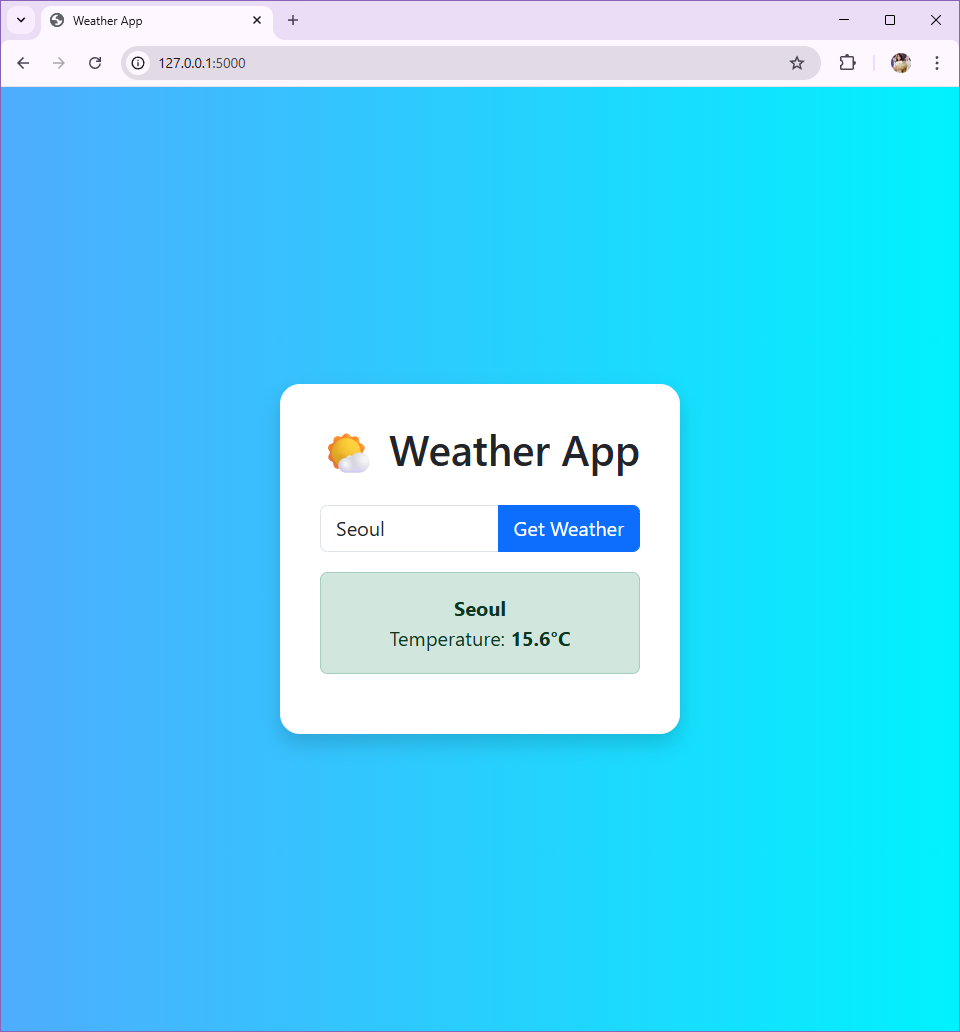
This project is a simple, mobile-friendly **web-based weather application** built using Python and Flask. The app allows users to input a city name and, upon submission, fetches real-time weather data—specifically, the current temperature—for that location using the **Open-Meteo API**. The interface is styled with **Bootstrap 5**, making it responsive and visually appealing on both desktop and mobile devices.

**🚀 Features Demonstrated**

The app's main features include:

* A clean, single-page user interface with a form to input the city name.
* Real-time retrieval of **current weather data** using Open-Meteo's free APIs.
* Error handling for invalid or unknown cities.
* Dynamic display of weather results with styled success and error messages.
* Mobile-first responsive layout using Bootstrap.

Here’s a screenshot of the app in action:



**🧠 Use of AI in Development**

AI played a valuable role throughout this project. I used **ChatGPT** to:

* Design the project’s **folder structure** following Flask best practices.
* Write and refine the get\_weather\_data() function for interacting with the API.
* Fix a specific import error and a jinja2.exceptions.TemplateNotFound issue.
* Style the frontend with Bootstrap and improve the user experience.
* Write this project summary for submission.

Using AI helped me debug faster, avoid unnecessary trial-and-error, and focus on learning how all parts of the app work together.

**💡 Reflections**

This project taught me how to integrate multiple key components of a full web app: frontend form handling, backend API calls, and dynamic rendering of templates. I gained hands-on experience with **Flask routing, blueprints, templates, and error handling**. One of the challenges I faced was figuring out why Flask couldn’t locate my index.html file—it turned out to be a mismatch in the folder structure, which I resolved with AI help.

**🌟 What I’m Proud Of**

I’m most proud of getting the **entire app working end-to-end**—from a city name input to a clean weather result—using real API data, and wrapping it in a mobile-friendly interface with minimal setup.

**🔧 What I’d Improve**

If I had more time, I would add:

* **Weather icons** and descriptions (e.g., sunny, cloudy).
* A **5-day forecast** view using Open-Meteo’s extended API options.
* A **city autocomplete feature** for better usability.
* **Unit toggling** (°C/°F) and geolocation support to auto-detect user location.