**Description**:  
This project will be a weather dashboard that allows users to search for weather information by city or their current location. The app will display the current weather conditions as well as a 5-day forecast, including details like temperature, humidity, and wind speed. The dashboard will be a simple, user-friendly web application that uses the **Open-Meteo API** for real-time data.

**Stack**:

* **Back-end**: Python with Flask (for handling routes, API calls, and rendering templates)
* **Database**: PostgreSQL with SQLAlchemy (to store user information, login details, and possibly search history)
* **Front-end**: HTML, CSS, JavaScript (for the user interface and interaction)
* **API**: Open-Meteo (for retrieving weather data)

**Focus**:  
The project will have a **full-stack focus**, with Python/Flask managing the back-end and RESTful API integration, while the front-end will use JavaScript for interactivity and dynamic weather data display.

**Type**:  
This will be a **traditional website**, accessible via desktop, using HTML/CSS for the UI and Flask to serve the pages.

**Goal**:  
The main goal of the app is to provide users with real-time weather information for a specific location (via search or location-based search). Users can view current conditions and plan their week with a 5-day forecast. There will also be a **simple login system** to personalize the user experience and save favorite cities.

**Users**:  
The target users for the app are **anyone** who wants quick access to weather information. This could include students, professionals, travelers, or general users who are looking to plan their day or week based on weather conditions.

**Data**:

* **Open-Meteo API** will be used to fetch real-time weather data.
* The data will include current conditions (temperature, weather state, etc.) and a 5-day forecast.
* **PostgreSQL** will be used to manage user accounts, login details, and potentially saved searches or favorite cities.

### **Approach Outline:**

1. **Database Schema**:
   * A simple PostgreSQL schema will include a users table (with fields like username, password, email) and a favorites table (to store user-specific favorite cities).
   * **SQLAlchemy** will be used to handle database interactions within Flask.
2. **Potential API Issues**:
   * Error handling will be implemented for invalid city names or failed API requests (e.g., displaying a message to the user).
   * API responses may require formatting before sending them to the front-end for display.
3. **Sensitive Information**:
   * User login credentials (passwords) will need to be securely hashed using a method like **Bcrypt**.
   * Since Open-Meteo doesn’t require an API key, no sensitive tokens need to be secured.
4. **Functionality**:
   * **City search**: Users can search for weather by entering a city name.
   * **Login system**: Users can create accounts, log in, and potentially save favorite cities for easy access.
   * **Weather display**: Shows current weather conditions and a 5-day forecast.
   * **User customization**: Logged-in users can save favorite cities to quickly access weather data for those locations in future sessions.
5. **User Flow**:
   * Users visit the homepage, where they can log in or search for weather information.
   * After entering a city or allowing location access, weather data is fetched and displayed (current and forecast).
   * Logged-in users can save favorite cities, which will be stored in the PostgreSQL database.
   * Users can perform additional searches for different cities.
6. **Beyond CRUD**:
   * The project involves more than basic CRUD by fetching and displaying data from an external API and providing user customization (favorites).
   * **Stretch goals**: If time allows, features like a dark mode or search history can be added.