**Weather Dashboard Application Documentation**

**Introduction**

This application provides real-time weather updates for various locations, along with a user-friendly interface for easy access to weather information.

**Setup Instructions**

Prerequisites

- Python

- Flask

- PostgreSQL

- OpenWeatherMap API Key

**Steps**

1. Clone the repository from GitHub: **git clone https://github.com/vishwajittidke/real-time-weather-app**

2. Install dependencies: **pip install -r requirements.txt**

3. Set up a virtual environment : **python -m venv venv**

4. Activate the virtual environment: **venv\scripts\activate**

5. Configure the PostgreSQL database:

   - Create a database named weather_db.

   - Update the PostgreSQL database credentials in app.py file.

6. Set environment variables in .env file:

   - API_KEY = OpenWeatherMapAPI Key

7. Run the application:

   - **python app.py**
     OR
   - flask run

**API Documentation**

Endpoints

-`/weather_dashboard`: Endpoint to fetch weather information from OpenWeatherMap API.

- `/login`: Endpoint to authenticate users using session-based authentication.

- `/register`: Endpoint to register users using user authentication.

**Example Requests**

- GET /weather_dashboard

- POST /weather_dashboard

- Get /login

- POST /login

- POST /register

- GET /register

**Example Response (JSON)**

Json response:


{

“temperature”: “34.99”,

“tempMin”: “31.94”,

“tempMax”: “34.99”,

“humid”: “55”,

“windspeed”: “6.17”

}


**WebSocket Integration**

The application uses long polling for real-time updates. When a user opens the dashboard Implemented long polling to update weather data every 10 minutes.

*Example Code*:

```
setInterval(() => updateWeather(), 6000);
```

**User Authentication**

User authentication is implemented using session-based authentication. Users need to log in with their username and password to access the dashboard. If you are new to website first you have to register

**How to Authenticate**

- Send a POST request to /login with username and password.

- Receive a session token in the response header (Set-Cookie).

- Use the session token for subsequent requests.

**Additional Details**

- The frontend is built with HTML, CSS, JavaScript, and Bootstrap.

- The backend is built with Flask and Python.

- The database schema includes a table user_details with columns id, fullname, email, username, and password.