Weather App

Description

This project focuses on creating an interactive web application that uses weather APIs to display real-time weather data. Users can select their desired location, view temperatures, wind conditions, atmospheric conditions and navigate through hourly or daily data. The project's goal is to apply knowledge from the lessons we had and to build a modern, responsive and user-oriented application.

Goal

- Using APIs for obtaining weather data:
  - WeatherAPI: https://www.weatherapi.com
  - Open-Meteo: https://open-meteo.com
  - Open Weather API: https://openweathermap.org/api
- Consolidating JavaScript skills by building real functionalities inspired by professional weather applications.
- Developing a web application focused on interactivity, performance, and best practices.

Minimum Features

- Sign up and log in pages – user authentication with data stored in localStorage (username, email, password etc.)
- User details page – automatic detection of user's city using browser geolocation, displaying user profile data and basic current weather information.
- Daily forecast – display weather forecast for each day (minimum and maximum temperatures, weather conditions, icons etc.)
- Search functionality – search bar allowing users to find weather information by city, postal code etc.
- Favorites system – save searched cities to a list stored in localStorage with add/remove functionality; view all favorites in a list where clicking any item opens a dedicated page with detailed weather information.

Other Features (optional)

- Weather alerts notifications – use WebSockets to receive real-time severe weather alerts and display them as popup notifications.
- Compare weather feature – select two cities from favorites and display their weather data side-by-side for easy comparison.
- Simple Chart - Use a lightweight JS chart (like Chart.js) to show temperature trend for the day or week.
- Installable PWA – the app can be added to a mobile device like a native app.

Software Principles

- Separation of concerns – API logic, DOM manipulation, and styles are separated into different modules.
- Reusability & modularity – functions for fetch, display, and data processing are reusable.
- Error handling – handling API errors, lack of connection, or incomplete data.
- Best Practices – applying ES6+, async/await, const/let and arrow functions.