Weather Dashboard Documentation (Frontend Intern Task)

Submitted By

Shanker Pangeni

Submitted to

Yeti Bytes

Date: August 31, 2025

1. Project Overview

The Weather Dashboard is a React-based web application that fetches real-time weather data from the OpenWeatherMap API. The app allows users to enter a city name and get current weather details including temperature (Celsius/Fahrenheit), weather condition, humidity, and country information. Humidity is displayed with a custom icon for better visual representation.

Purpose:
To create a responsive, user-friendly interface that shows accurate weather information and allows toggling between Celsius and Fahrenheit.

---

2. Problem Approach

When approaching the problem, I focused on:

1. **User Input Handling**
   - Accept city names and handle invalid inputs gracefully.
   - Display error messages when the city is not found.
2. **API Integration**
   - Used OpenWeatherMap's API for real-time weather data.
   - Extracted key information: temperature, weather description, humidity, and country.
3. **State Management**
   - Used React's `useState` to manage city input, weather data, errors, and UI states (like Celsius/Fahrenheit toggle).
4. **Responsive Design**
   - Built the layout with Tailwind CSS to ensure it looks good on all screen sizes.
   - Used flexbox for clean alignment of temperature, humidity, and weather icons.
5. **Temperature Conversion**
   - Stored temperature in Kelvin from API and converted to Celsius and Fahrenheit using formulas.
   - Added a toggle button to switch units dynamically.
6. **Error Handling**
   - Implemented try/catch blocks for API errors.
   - Displayed a temporary alert when a city is not found.
7. **Iconography**
   - Weather condition images were mapped based on the weather description.
   - Added a dedicated humidity icon for visual clarity.

3. Key Decisions

- **Static vs. API Icons:**
  Initially, static images were used. Later, OpenWeatherMap icons could be used for dynamic weather representation.
- **Temperature Toggle:**
  Decided to allow switching between Celsius and Fahrenheit to cater to different user preferences.
- **Error Display:**
  Used a dismissible alert at the top instead of inline messages for better visibility.
- **Component Structure:**
  Decided to keep everything in a single `Weather` component for simplicity, as the project is small and straightforward.
- **Responsive Layout:**
  Used Tailwind CSS classes to ensure mobile-first responsive design.
- **Humidity Feature:**
  Added a humidity section below the temperature with a custom icon and percentage value to provide a complete weather snapshot.

4. Thought Process

1. **Start with Input & Fetch Logic:**
   - Ensure city input is captured correctly.
   - Fetch weather data from API and parse JSON.
2. **Manage UI State:**
   - Use `useState` to store data and trigger re-renders.
3. **Display Weather Info:**
   - Map weather status to appropriate icons.
   - Show temperature with toggle functionality.
   - Display humidity with icon for better UX.
4. **Error Handling & Feedback:**
   - Display errors prominently without breaking the UI.
   - Auto-hide error alerts to improve flow.
5. **Responsive Design:**
   - Use Tailwind to make all sections (input, temperature, humidity, city/country) responsive.
   - Test on multiple screen sizes for layout consistency.

5. Future Improvements

- Add **forecast for multiple days**.
- Use **OpenWeatherMap icons dynamically** instead of static images.
- Implement **geolocation** to fetch weather automatically for user’s location.
- Add **animations** for weather icons (rain, snow, clouds) for better UX.