Product Requirements Document (PRD): Weather Application

Purpose and Background

The goal of this weather application is to deliver real-time weather data to users upon loading the website, automatically showing local weather if location permission is granted, and allowing manual search by city if desired. The application is designed for simplicity, reliability, and responsiveness, supporting both successful data fetching and clear error handling if the API or location lookup fails.

Stakeholders

- End Users (general public seeking weather information)
- Product Owner/Developer
- UI/UX Designers
- QA/Test Engineers

Functional Requirements

Initial Load and Location-based Weather

- On first visit (or reload), the app attempts to access the device's current location using the browser's Geolocation API.
- If permission is granted:
    - The app queries Open-Meteo for current, hourly, and weekly weather based on the user's latitude and longitude.
    - If Geolocation fails or is denied, the app falls back to approximately determine location by IP using a public API (e.g., ipwho.is).
    - If both location approaches fail, an appropriate error message is displayed with a prompt for manual search.

City Search

- The user can enter a city name in the search bar.
- On submitting a city, the app uses Open-Meteo's geocoding service to fetch matching coordinates and names.
- If a city is found:
    - The app requests weather data for the city's coordinates and updates all displayed weather components.
- If not found, an error message "City not found" (or similar) is shown to the user.

Weather Information Display

- Current Weather: Displays temperature (°C/°F), humidity, wind speed, pressure, precipitation, weather condition description, and icon.
- Hourly Forecast: Shows temperature, descriptions, and icons for each hour (at least the next 30 hours).
- 7-Day Forecast: Shows each day's max/min temperature (°C/°F), weather code and description, and an icon.
- Weather condition codes are translated to human-readable descriptions and emoji-style icons.

Error Handling

- Any problem with network, API responses, missing data, or denied permissions triggers an error component that explains the issue and prompts a retry or alternative action.
- The app shows a loader/spinner indicator while data is being fetched.

Non-Functional Requirements

Usability

- The interface should be mobile-friendly and performant.
- Inputs are debounced and properly handled for fast or repeated requests.
- The UI must be visually clear, attractive, and offer visual feedback for errors, loading, and successful states.

Accessibility

- All inputs, notifications, and displayed data must be accessible to assistive technologies (ARIA roles for error, status, and key landmarks).

Technical/Integration

- Written with React and Vite for fast local and production builds.
- State managed via React hooks.
- Uses Axios for HTTP requests.
- Relies on Open-Meteo for both weather and geocoding APIs.
- All sensitive operations (API calls, location) handled client-side only.
- Error cases and alternate flows (like permission denied, city not found, API down) gracefully degrade to user-friendly explanations.

Dependencies and Constraints

- Internet connection required.

- Relies on browser APIs for location.
- Open-Meteo service must be available and responsive.
- Geolocation and IP-based location may not always yield fully accurate results, so manual city search is essential as a fallback.

Acceptance Criteria

- On application start, weather for the current location is shown if permissions are granted.
- If permission is denied or not supported, fallback mechanisms are triggered and clearly communicated.
- City search updates the weather display with accurate information or a clear error if the city doesn't exist.
- Up-to-date weather data is displayed whenever possible, with user interface feedback for all loading and error conditions.
- UI is responsive, visually polished, and user-focused.
- All functions gracefully handle unavailable APIs or network errors.

Out-of-Scope Features

- User authentication, account system
- Persistent user preferences (beyond current session)
- Notifications or weather alerts (push)
- Historical weather queries or analytics

This PRD covers all high-level requirements and details derived from the codebase and stated objectives, ensuring clear guidance for any stakeholder or implementer of the weather application.Below is a detailed Product Requirements Document (PRD) for your Vite+React weather app using Open-Meteo, based on your source files and project goals.

Product Requirements Document (PRD)

Weather App (Vite + React + Open-Meteo)

---

1. Purpose

This application allows users to view real-time weather data for their current location (with permission) or for any searchable city, providing current, hourly, and weekly forecasts. Usability, responsiveness, and clear error handling are key priorities.

---

2. Features/Requirements

A. Geolocation Weather Detection

- On page load, attempt to access browser geolocation.
    - If granted, fetch weather for these coordinates.
    - If denied or unavailable, attempt to guess location via IP-based fallback.
    - If both fail, prompt user to search for city manually.
- Use reverse geocoding to get human-readable location names from coordinates.

B. City Search Functionality

- Search bar for city names.
    - On search, geocode input to get coordinates.
    - If city not found, display error ("City not found").
    - If network/geocoding error, display relevant error.
- On successful city choice, fetch and display weather for that location.

C. Weather Data Display

- Current conditions: Temperature (°C/°F), humidity, wind speed, pressure, precipitation, weather code/description, and emotion-style icon.
- Hourly forecast: Temperatures and weather icons for at least next 30 hours.
- 7-day forecast: Daily high/low, weather code/description, and icon for each day.
- Update all displays when user changes location (auto or via search).

D. Feedback & Error Handling

- Loader/spinner shown during data fetches.
- All errors (location, API, city not found) are displayed in a prominent, user-friendly way.
- Edge cases (missing or malformed data, denied permissions, network failure) handled gracefully and do not crash UI.

E. UI/UX

- Responsive layout; mobile and desktop friendly.
- Visually clear, modern, and accessible (contrast, font, keyboard support).

---

3. Technical Requirements

- React (functional components with hooks)
- Vite build system
- Axios for HTTP requests
- State held at top level (App.jsx), with children as pure components
- All API and service URLs client-side (no backend)
- Weather and geocoding provided by Open-Meteo API
- Utility code for translating weather codes to icons/descriptions

---

4. Constraints

- Internet and Open-Meteo API availability are required.
- Browser must support Geolocation API or fallback to IP-based mechanism.
- App does not persist user preferences between reloads (stateless).

---

5. Out of Scope

- User accounts or login
- Push notifications/alerts
- Custom user settings

---

6. Success Criteria / Acceptance

- App always shows weather for either current location or entered city, with graceful fallback and error messages.
- City search works as expected and errors are clearly shown if city is not found.
- No blank or broken states; all loading and error situations are visually distinguished.
- Current, hourly, and weekly weather are accurate per Open-Meteo API, with proper labeling/icons.