Weather Reporter – Technical Implementation Document

Overview

The Weather Reporter is a lightweight, browser-based weather information dashboard developed using HTML, CSS (via Bootstrap), and vanilla JavaScript. It fetches real-time weather data from an external API (WeatherApi.com)..

Technology Stack

- Frontend: HTML5, CSS3, Bootstrap 5, Vanilla JavaScript
- Hosting: Vercel
- API Integration: WeatherAPI
- Serverless Function: Node.js (via Vercel Serverless Functions)
- Security: Environment variable handling for API key protection

Environment Variable & API Integration

JavaScript running in the browser cannot securely access `.env` variables directly. To solve this securely and efficiently:

1. Vercel's serverless function (`api/weather.js`) acts as a middleware layer.
2. This function accesses the `WEATHER_API_KEY` stored securely in the environment.
3. The client (`index.js`) sends a request to this serverless function with the city name.
4. The serverless function then fetches real-time data from WeatherAPI and returns it to the frontend.

Why this method?

- Avoids exposing sensitive API keys in the code.
- Keeps the codebase simple by using serverless architecture over setting up a full backend.

Reason for Choosing Vanilla JS Instead of Frameworks

While modern frameworks like React or Vue offer many benefits, for a small to medium-scale utility app like this, using them can be overkill. This project intentionally uses vanilla JavaScript to:

- Keep the bundle size minimal for faster load times.
- Maintain simplicity and avoid unnecessary abstraction.
- Focus more on raw JavaScript fundamentals and environmental challenges (e.g., managing `.env` access).

Additionally, building this from scratch helped solidify understanding of:

- DOM manipulation without libraries.
- Native fetch-based API integration.

- Bootstrap-based responsive design.
- Environment variable limitations in frontend JS and how to work around them securely.

Functional Features

The application provides the following core features:
- User local time display (based on system time).
- Selected city local time display, derived from the city's `tz_id` using `Intl.DateTimeFormat`.
- Real-time weather data display for selected city:
  - Temperature
  - Condition text and icon
  - Feels like temperature
  - Wind speed
  - Humidity
  - Pressure
  - Precipitation
  - Visibility
  - Cloud coverage
  - UV index
  - Last updated timestamp
- Dynamic background image that changes based on local time and weather conditions.
- Responsive design using Bootstrap.
- Loading spinner while waiting for the API response.
- Error handling for invalid or empty input with fallback to 'Colombo'.
- Weather icons displayed directly from API responses.

UI & UX Considerations

- Loading State: Bootstrap spinner is displayed while waiting for API response.
- Search UX:
  - On pressing Enter or clicking the search button, weather is fetched.
  - If the input is invalid or blank, a default fallback is triggered with an alert.
- Weather Icons: Pulled directly from WeatherAPI’s icon URLs.
- Responsive Layout: Built fully with Bootstrap 5 classes.

Asset Management

- Images (backgrounds) are stored locally in the `/assets` folder.
- Since Vercel treats static assets as CDN-optimized by default, storing images locally still ensures fast, distributed delivery.
- Bootstrap and icon libraries are loaded via CDN for better caching and geographic distribution.

Optimization & Best Practices

- Minimal external dependencies to reduce bundle size.
- Environment-based security through serverless functions.

- Modular code organization.
- Graceful fallback and retry on error.

Auto Refresh & Clock Sync

- Local system clock updates every second using `setInterval(updateClock, 1000)`.
- City time clock updates every second using `Intl.DateTimeFormat`.
- Weather data auto-refreshes every 10 minutes using `setInterval(fetchWeather, 600000)`.

Deployment

Deployed on Vercel using:
- Vercel CLI: `vercel`
- Environment variable `WEATHER_API_KEY` configured in project settings
- Automatic HTTPS, CDN distribution, and serverless function hosting