Weather Terminal Advanced
Project Report

Prepared by: Samad Mehboob
samadmehboob940@gmail.com

Date: August 12, 2025

A comprehensive overview of the Weather Terminal Advanced application, detailing its features, implementation, and reported issues.

Contents

1 Introduction

The Weather Terminal Advanced is a web-based application designed to provide real-time weather information and forecasts using the OpenWeather API. The application features a cyberpunk-inspired user interface with a neon-green aesthetic, responsive design, and advanced functionalities like geolocation and temperature trend visualization. This report outlines the application's features, implementation details, a reported geolocation issue, and potential improvements.

2 Features

The Weather Terminal Advanced offers a range of features to enhance user experience and provide comprehensive weather data.

2.1 Current Weather Display

The application fetches and displays current weather data, including:

- Temperature (in Celsius or Fahrenheit)
- Weather condition (e.g., Clear, Rain)
- Humidity
- Wind speed
- Air quality index (AQI)

2.2 24-Hour Forecast

A 24-hour weather forecast is provided, showing temperature, weather conditions, and wind speed at 3-hour intervals for the next 8 time slots.

2.3 Geolocation Support

Users can retrieve weather data for their current location using the browser's Geolocation API, with a fallback to manual city input if geolocation fails.

2.4 Temperature Trend Visualization

A line chart, powered by Chart.js, visualizes temperature trends over the forecast period, enhancing data interpretability.

2.5 Unit Conversion

Users can toggle between metric (Celsius, meters/second) and imperial (Fahrenheit, miles/hour) units for temperature and wind speed.

2.6 Cyberpunk UI

The interface features a neon-green, glitch-effect design with responsive layouts for mobile, tablet, and desktop devices.

3 Implementation

The application is built using HTML, CSS, and JavaScript, leveraging modern web technologies and the OpenWeather API.

3.1 Technology Stack

- **HTML/CSS**: For structure and styling, with responsive design using `clamp()` for fluid sizing and media queries.
- **JavaScript**: For dynamic functionality, including API calls and DOM manipulation.
- **Chart.js**: For rendering temperature trend charts.
- **OpenWeather API**: Provides weather (`2.5/weather`), forecast (`2.5/forecast`), and air quality (`2.5/air_pollution`) data.

3.2 Code Structure

The application is a single-page web app with the following key components:

- **HTML**: Defines the UI structure with input fields, buttons, and weather display sections.
- **CSS**: Implements a cyberpunk aesthetic with neon-green colors, glitch animations, and responsive layouts.
- **JavaScript**: Handles API calls, geolocation, unit toggling, and chart rendering.

Here is a sample of the geolocation function:

```
function getGeolocation() {
    if (!navigator.geolocation) {
        document.getElementById('location').textContent = '[GEO_UNSUPPORTED]';
        document.getElementById('condition').textContent = '[BROWSER_NO_GEO_SUPPORT]';
        console.error('[GEO_ERROR]: Browser does not support geolocation');
        alert('[GEO_UNSUPPORTED]: Your browser does not support location services. Please enter a city manually.');
        return;
    }
    container.classList.add('loading');
    navigator.geolocation.getCurrentPosition(
        pos => {
            const { latitude, longitude } = pos.coords;
            console.log(`[GEO_SUCCESS]: Lat=${latitude}, Lon=${longitude}`);
            document.getElementById('city-input').value = '';
            fetchWeather(latitude, longitude);
        },
        err => {
            console.error('[GEO_ERROR]:', err.message, 'Code:', err.code);
            document.getElementById(’location’).textContent = ’[GEO_ERROR]’;
            document.getElementById(’condition’).textContent = `[ERROR: ${err.message}]`;
            container.classList.remove(’loading’);
            if (err.code === 1) {
                alert(’[GEO_DENIED]: Please allow location access in browser settings and try again.’);
            } else if (err.code === 2) {
                alert(’[GEO_UNAVAILABLE]: Location services unavailable. Please check GPS, network, or enter a city manually.’);
            } else {
                alert(’[GEO_ERROR]: Unable to fetch location. Try entering a city manually.’);
            }
        },
        { timeout: 10000, maximumAge: 60000, enableHighAccuracy: true }
    );
}
```

3.3 API Integration

The application uses the OpenWeather API endpoints:

- `2.5/weather`: Fetches current weather data by city name or coordinates.
- `2.5/forecast`: Provides 3-hourly forecast data for the next 24 hours.
- `2.5/air_pollution`: Retrieves air quality index (AQI) based on coordinates.

API calls are made asynchronously using `fetch`, with robust error handling to display issues in the UI.

4 Reported Geolocation Issue

A reported error, `[GEO_UNAVAILABLE]: Location services unavailable. Check GPS or network.`, indicates that the Geolocation API failed to retrieve the user’s location. This corresponds to error code 2 from `navigator.geolocation.getCurrentPosition`.

4.1 Possible Causes

- **GPS/Location Services Disabled**: The device’s location services may be turned off, preventing the browser from accessing location data.
- **Network Issues**: Lack of a stable internet connection, as geolocation relies on Wi-Fi triangulation or IP-based location services.
- **Non-Secure Context**: The application must run on HTTPS or localhost, as Geolocation API does not work on HTTP or file:// protocols.

- **Browser Compatibility**: Older browsers or configurations may not support geolocation properly.

4.2 Solutions Implemented

The code includes:

- Detailed error handling with specific messages for permission denial (code 1), position unavailable (code 2), and timeout errors.
- UI feedback displaying errors in the `location` and `condition` elements.
- A fallback to manual city input when geolocation fails.

Users are advised to:

- Enable device location services.
- Ensure a stable internet connection.
- Run the application on HTTPS or localhost.
- Check browser settings for location permissions.

5 Future Improvements

To enhance the Weather Terminal Advanced, the following improvements are suggested:

- **One Call API Integration**: Use OpenWeather's One Call API for more efficient data retrieval, including hourly and daily forecasts, and weather alerts.
- **Favorite Cities**: Implement LocalStorage to save user-preferred cities.
- **PWA Support**: Add service workers for offline access and installability.
- **Multi-Language Support**: Detect browser language and translate UI labels (e.g., English, Urdu).
- **Additional Data**: Include UV index, sunrise/sunset times, and detailed air pollution metrics.

6 Conclusion

The Weather Terminal Advanced is a robust and visually appealing weather application with a cyberpunk-inspired interface. Despite the reported geolocation issue, the application provides reliable weather data and forecasts, with a fallback to manual input. Future enhancements can make it even more user-friendly and feature-rich, aligning with professional weather applications.