CHRIST (Deemed to be) UNIVERSITY

FRONT END UI/UX DESIGN FUNDAMENTALS

WEATHER FORECAST WEBSITE

Submitted by :

JON SUNIL THOMAS 2460384

TISHA PINHEIRO 2460465

NANDANA PADMASENAN 2462117

Submitted on : 26 September 2025

ABSTRACT

This project is a **live weather dashboard** website that provides real-time weather information and forecasts for a user-specified location. The key goal was to design and develop a **responsive and intuitive user interface** that fetches and displays data from a third-party weather API. The website was built using **HTML5, CSS3, and JavaScript**, focusing on asynchronous data fetching and dynamic content updates. The final outcome is a functional, visually appealing dashboard that allows users to quickly and easily check weather conditions, serving as an excellent demonstration of front-end development skills and API integration.

OBJECTIVE

The primary goals of this project were to:

- Design a modern and user-friendly interface for a weather dashboard.
- Develop a **fully responsive layout** that adapts to different screen sizes.
- Implement JavaScript to fetch and display **real-time weather data** from an external API.
- Ensure the website is accessible and provides a clear, readable display of weather information.

SCOPE

The scope of this project encompasses the complete design and development of a static, fully responsive website for weather forecast website. It focused on front-end development, including UI design and API integration. Uses a **third-party API** to fetch weather data, as opposed to a custom backend. This website is intended for desktop, tablet, and mobile viewports. It utilizes HTML, CSS, and vanilla JavaScript. No major front-end frameworks like React or Angular were used.

TECHNOLOGIES USED

Tool/Technology	Purpose
HTML5	Markup and content structure
CSS3	Styling and responsive layout management
JavaScript (ES6)	API calls, DOM manipulation, and interactivity
OpenWeather API	Source for real-time weather data
VS Code	Code editor
Chrome DevTools	Testing and debugging

HTML STRUCTURE OVERVIEW

 Used semantic tags: <header>, <main>, <section>, <footer>.

 Structured the page into distinct components: a search bar, a main weather display area, and a forecast section.

CSS STYLING STRATEGY

- Used an **external CSS file** (`style.css`).
- Organized styles with comments and sections for readability.
- Techniques used:
    - **Flexbox** and Grid for flexible layout of weather information.
    - **Media Queries** for responsiveness across devices.
    - CSS variables for theme customization (e.g., changing colors based on weather conditions).
    - Simple hover effects and transitions on elements like the search button and weather cards.

KEY FEATURES

Feature	Description
Search Functionality	A user can enter a city name to get current weather data.
Responsive Design	The layout adjusts seamlessly to different screen sizes, ensuring a good user experience on any device.
Dynamic Content	Weather data, icons, and background images change based on the current weather conditions.
Error Handling	The dashboard provides user-friendly error messages if a city is not found or the API fails to respond.
Real-time Data Display	Displays key weather metrics like temperature, humidity, wind speed, and a weather description.

CHALLENGES FACED AND SOLUTIONS

CHALLENGES FACED:

Asynchronous JavaScript and API fetching

Making a dynamic, responsive layout for varying content

Hiding API key credentials

SOLUTION:

Used async/await and try...catch blocks to handle API requests cleanly and manage potential errors.

Utilized **Flexbox** and **CSS Grid** to create a fluid layout that could adapt to different amounts of weather information without breaking.

Used a proxy server or an environment variable file to prevent the API key from being exposed in the client-side code, although for this project, it was a client-side implementation using a free, non-sensitive key

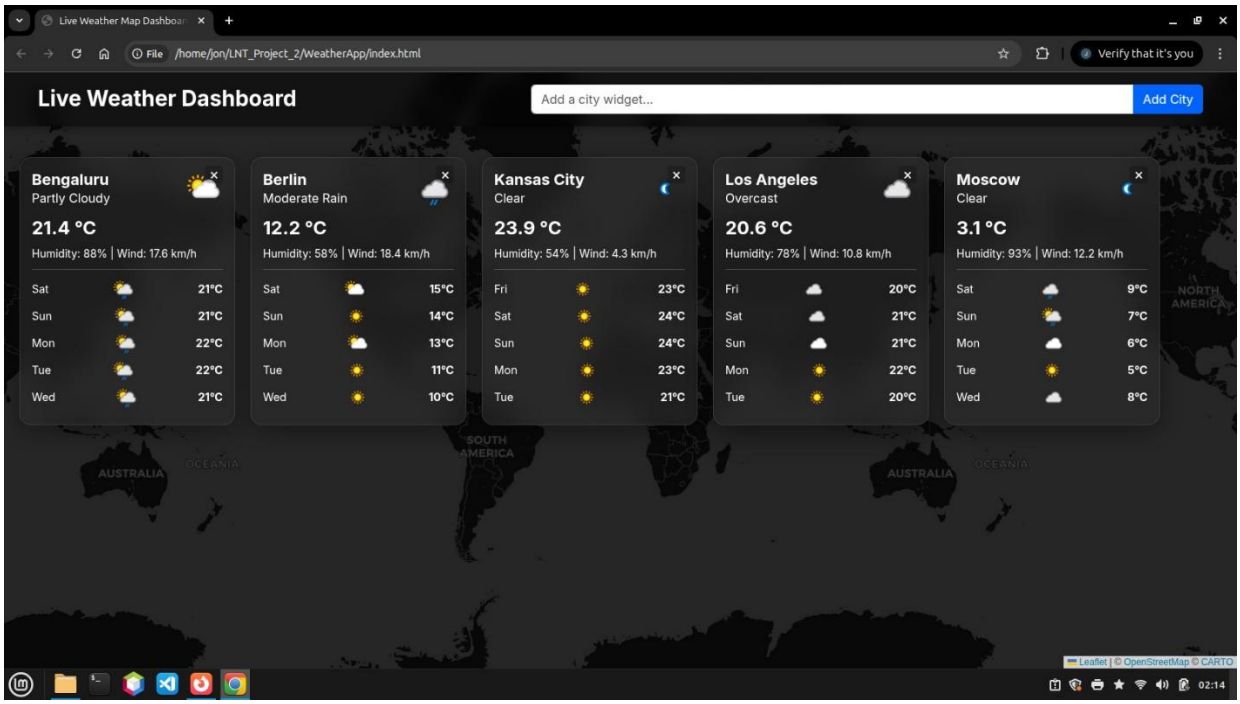
OUTCOMES

The project successfully delivered a clean, consistent, and visually engaging weather dashboard. All key components, including the search function and data display, work as intended. I gained significant experience in **API integration**, asynchronous JavaScript, and the importance of robust error handling in web applications.

FUTURE ENHANCEMENTS

1. Add a 5-day or hourly forecast display.
2. Allow users to switch between Celsius and Fahrenheit.
3. Implement a geolocation feature to automatically detect the user's location.
4. Integrate a different API for a radar map.

SCREENSHOT OF THE WEBSITE



CONCLUTION

This weather dashboard project successfully demonstrates the core principles of front-end web development, including UI design, responsive layout, and a practical application of JavaScript for API integration. This mini-project helped strengthen our ability to build dynamic web applications and provided practical insights into handling data from external sources. The hands-on implementation of design principles also enhanced our understanding of user-centric web design.