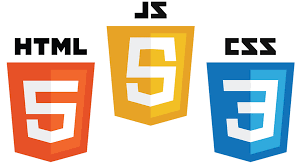
# Une image contenant Graphique, symbole, logo, conception Description générée automatiquementProjet Report

## Introduction

The WeatherInsights project is an interactive web application that provides real-time weather information and air quality data. Its main goal is to allow users easy access to weather forecasts, explore atmospheric pollution levels, and leverage interactive charts to visualize meteorological data, such as changes in humidity and temperature over time. This platform combines modern technologies and APIs to deliver a seamless and informative user experience.

## Languages Used

Several technologies were employed to design and develop this project, including:



1. **HTML (Hypertext Markup Language)**

HTML was used to structure the webpage. The main elements of the application, such as the sidebar, header, search fields, and weather information containers, were defined using HTML. It forms the foundation of the user interface.

1. **CSS (Cascading Style Sheets)**

CSS was used to style and customize the appearance of the application. Through CSS, we:

Designed a visually appealing layout with responsive elements.

Applied color themes and animations to enhance the interface's aesthetics.

Adapted the layout for different devices (desktops, tablets, smartphones).**3. JavaScript (JS)**

JavaScript enabled the interactivity of the application. Key functionalities developed with JavaScript include:

Event handling, such as city-based searches and automatic location detection.

API integration to fetch weather and pollution data.

Dynamic chart management via the Chart.js library.

Real-time updates of displayed data without page reloads.

## APIs and External Libraries Used

This project relies on APIs and external libraries to provide accurate and up-to-date information. Below is a detailed list of the external resources used:

1. **OpenWeatherMap API**

Weather Data API: Provides detailed information about current weather conditions (temperature, humidity, wind speed, description).

Air Pollution API: Fetches data on pollution levels (PM2.5, PM10, CO, NO2, etc.) and the Air Quality Index (AQI).

Geocoding API: Translates city names into geographical coordinates (latitude and longitude) to locate weather data.

Reverse Geocoding API: Used to retrieve a place's name from GPS coordinates provided by the user's browser.

1. **Chart.js**

This JavaScript library enables the creation of interactive and visually appealing charts. It is used to:

Display hourly and daily forecasts in chart form.

Represent variations in pollution levels or other meteorological parameters.

Facilitate the visualization of complex data for users.

**3. Moment.js**

Moment.js was also used to manipulate and display the sunrise and sunset times in the application. These events are often provided in UTC, and it's essential to convert them to local time to offer an accurate and consistent experience for users, regardless of their location.

In addition, Moment.js was used in this project primarily to ensure that the times displayed on the graphs were adjusted to the local time of each city. Weather data is often provided in UTC (Coordinated Universal Time), which requires conversion to the local time zone to ensure users see the correct time for their location.

In the context of the graphs, using Moment.js allows the conversion of UTC time to local time by considering the specific time zone of the city. This enables plotting temperatures on a time axis that reflects the local time, which is essential for understanding and interpreting the forecasts. Without this conversion, the displayed times could be incorrect, which would compromise the accuracy of the graphs and the user experience.

**4. Navigator.geolocation**

A native functionality of modern browsers that retrieves the user's GPS coordinates to automatically display weather information for their current location.

## Failure

Several challenges were encountered during the development of this project:

1. **Map**

An attempt was made to integrate an interactive map displaying weather conditions via a specialized JavaScript library. However, this feature could not be completed due to technical and time constraints. This highlights the importance of thoroughly planning complex features and ensuring that chosen tools meet project requirements.

1. **Humidity Chart**

Displaying humidity charts using Chart.js also posed challenges, leading to recurring errors during initialization. Despite multiple adjustments, this issue persists, underscoring the need for a deeper understanding of the library's specific configurations.

These experiences provide valuable lessons for future developments and emphasize the importance of rigorous tool management and planned functionalities.

1. **Git Hub**

The code I implemented for fetching air quality data works locally, but unfortunately, it doesn't function properly when deployed on GitHub. Despite my efforts, I was unable to resolve the issue. The problem seems to stem from how the external API is accessed in the GitHub Pages environment. Unlike in the local setup, GitHub Pages might have restrictions on cross-origin requests (CORS) or handling environment variables, causing the API fetch to fail. Additionally, external images and dynamic data are not loading as expected, possibly due to path issues or permission restrictions. While the code works fine on a local server, I have not been able to find a solution for making it fully functional when deployed on GitHub Pages.

## Conclusion

The WeatherInsights project represents a synthesis of several web development skills. It demonstrates how to use modern front-end technologies to create a functional and user-friendly application. The integration of APIs enables the provision of dynamic and accurate data, while tools like Chart.js enrich the user experience. This application could serve as a foundation for more advanced projects, integrating long-term forecasts or personalized features for users.

Live demo link : <https://exall29.github.io/Web-final-project/>