**Web Application Project Report**

**Author: Tannous Paul**

**Table of Contents**

1. Introduction
2. Code Structure  
   2.1 HTML File Structure  
   2.2 CSS File Organization  
   2.3 JavaScript Functionality
3. Conclusion

**1. Introduction**

This **Web Application Project** is designed to provide an interactive and visually engaging interface displaying weather information, analytics, and user-customizable options.

**Technologies Used**:

* **Visual Studio Code** with the Live Server extension for streamlined development.
* **HTML** for page structure.
* **CSS** for styling and layout.
* **JavaScript** for dynamic logic and user interactions.
* **OpenWeather API** for real-time weather and air quality data retrieval.

The application is organized into four primary pages:

1. **Dashboard** (index.html)
2. **Favorites** (favorite.html)
3. **Analytics** (analytics.html)
4. **Settings** (settings.html)

Each page is accessible via a shared sidebar navigation component.

**2. Code Structure**

**2.1 HTML File Structure**

The application consists of four main HTML files:

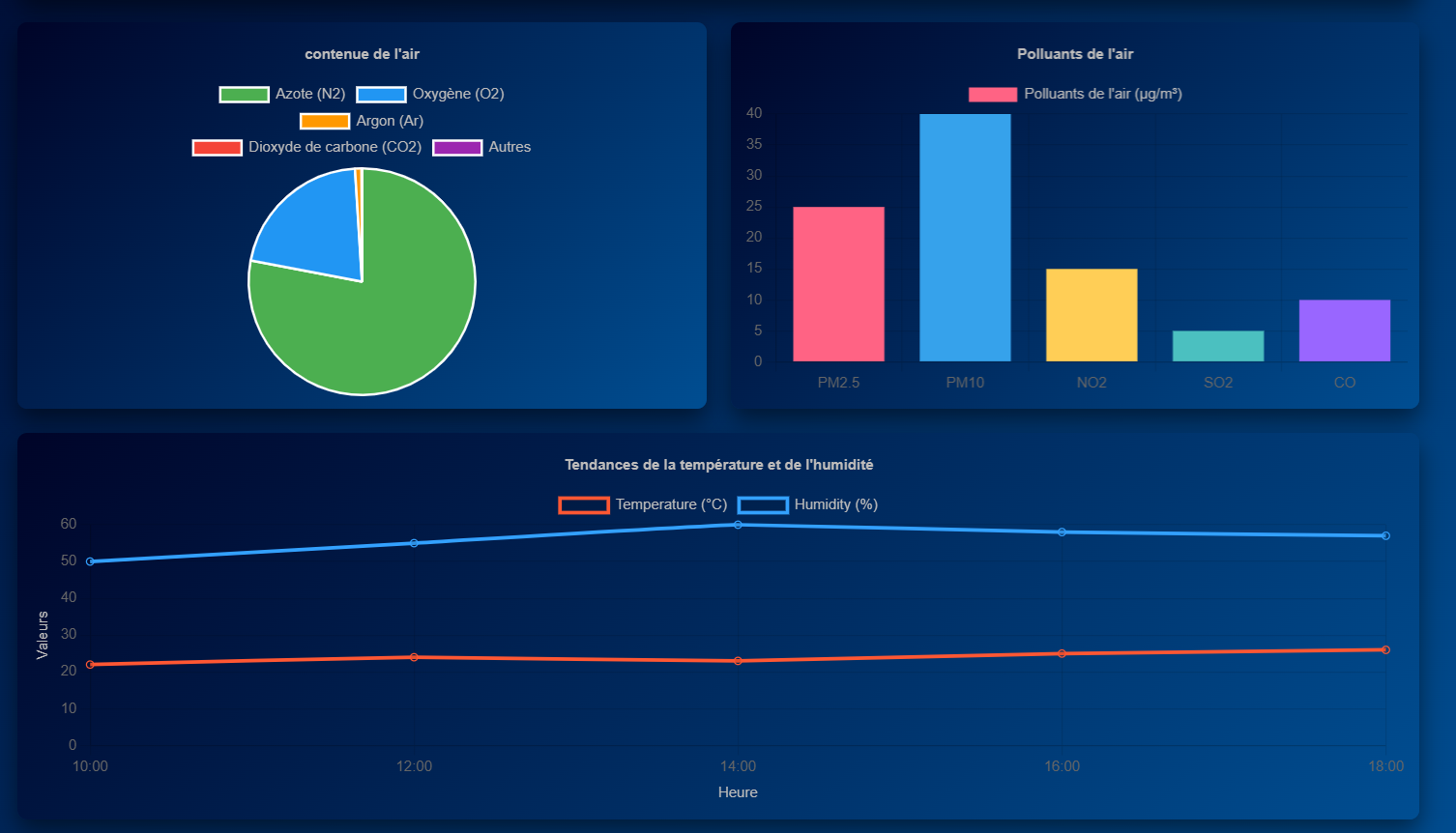
* **index.html**: The main dashboard displaying most interactive elements and data.
* **favorite.html**: Displays liked or saved elements chosen by the user.
* **analytics.html**: Intended to showcase analytical charts (currently under development).
* **settings.html**: Provides customization and configuration options.

**Common File Structure**:

* **Lines 1–12**: Import required files, including CSS and JavaScript.
* **Lines 14–60**: Sidebar content, enabling navigation across pages.
* **Line 60 to End**: Content specific to the individual page.

**Analytics.html :**

I didn’t succeed to capped the data from the api but I spent time on the design, so I decided to let it with fake data.



**Explanation of the Features of This Website**

One of the distinctive goals of this website was to implement a **"like" or "favorite" system** for weather parameters. Users can interact with individual cells on the dashboard to mark them as favorites, which would then be saved and displayed on the **Favorites** page.

**Behavior of the Favorite System:**

* Clicking on a cell changes its color to indicate its state:
  + **Blue**: Not liked (visible only on the dashboard).
  + **Purple**: Liked (visible on both the dashboard and the Favorites page).

 🡪Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

While this feature was conceptually appealing, it is not yet functional due to technical challenges in transferring information between different files and complications in handling data with localStorage.

This limitation highlights areas for future development, such as optimizing data persistence across pages and refining the communication between components.

**2.2 CSS File Organization**

Two CSS files manage the visual design and layout of the application:

1. **styles\_dashboard.css**:
   * Defines the sidebar design and interactivity.
2. **style\_grid.css**:
   * Manages the layout and display of data using a grid system.

**Example Grid Definition:**

css

Copy code

.grid-container {

display: grid;

grid-template-columns: repeat(8, 1fr); /\* 8 columns \*/

grid-auto-rows: 150px; /\* Default row height \*/

gap: 20px; /\* Spacing between cells \*/

}

**Color Theme**:  
A consistent color theme is achieved through the use of a **linear gradient**:

css

Copy code

linear-gradient(135deg, #001f3f, #007bff);

This design is applied across multiple CSS functions, creating a visually unified website.

**2.3 JavaScript Functionality**

The application uses three main JavaScript files to handle user interactions and data management:

1. **script.js**:
   * Manages sidebar dynamics, such as highlighting active sections.
2. **script\_weather.js**:
   * Handles interactions with the OpenWeather API to retrieve weather and air quality data.
   * **API Key**:

javascript

Copy code

const apiKey = "8ab4431e3edf2971b5cb1ceec8cb1e0a";

*Note: This key should be replaced with a private key for security purposes.*

* + Data is fetched from two key endpoints:

javascript

Copy code

const weatherBaseUrl = "https://api.openweathermap.org/data/2.5/weather";

const airQualityBaseUrl = "https://api.openweathermap.org/data/2.5/air\_pollution";

* + Main Functions:
    - fetchWeatherData(city): Fetches weather data for a specified city.
    - fetchAirQualityData(city): Retrieves air quality indices for a city.
    - displayWeatherData(data) and displayAirQualityComponents(components): Display weather and air quality data, respectively.

1. **analytics.js**:
   * Designed to provide analytical charts using libraries like Chart.js.
   * Currently, this functionality is not operational.

**3. Conclusion**

This project represents aweb application, offering a clean user interface with interactive features powered by the OpenWeather API. While some functionalities, like the analytics page, are still under development, the overall architecture is coherent.

**Next Steps:**

* Finalize the analytics page functionality.
* Optimize JavaScript code for better performance.
* Finish the Favorite System:
* Make customizable features in the setting file.