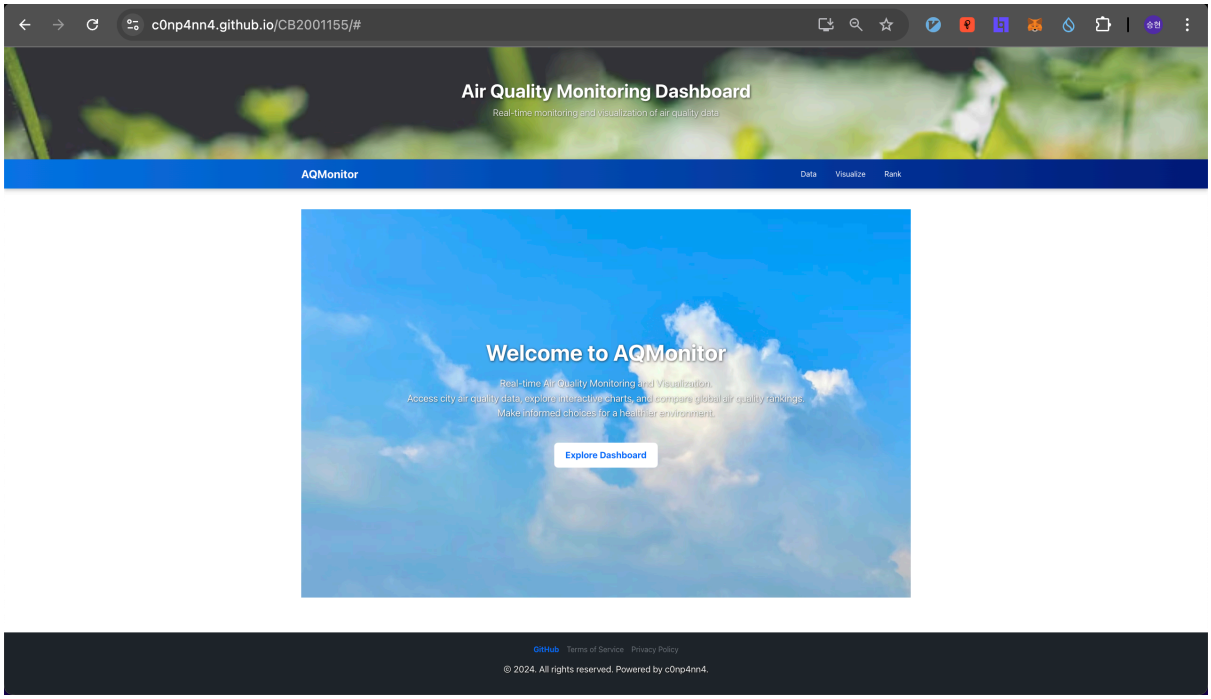
Web Application Programming (CB2001105-062)

Final Project Report

Name: 조승현 (Seunghyun Cho)
Student ID: 201824590

Introduction

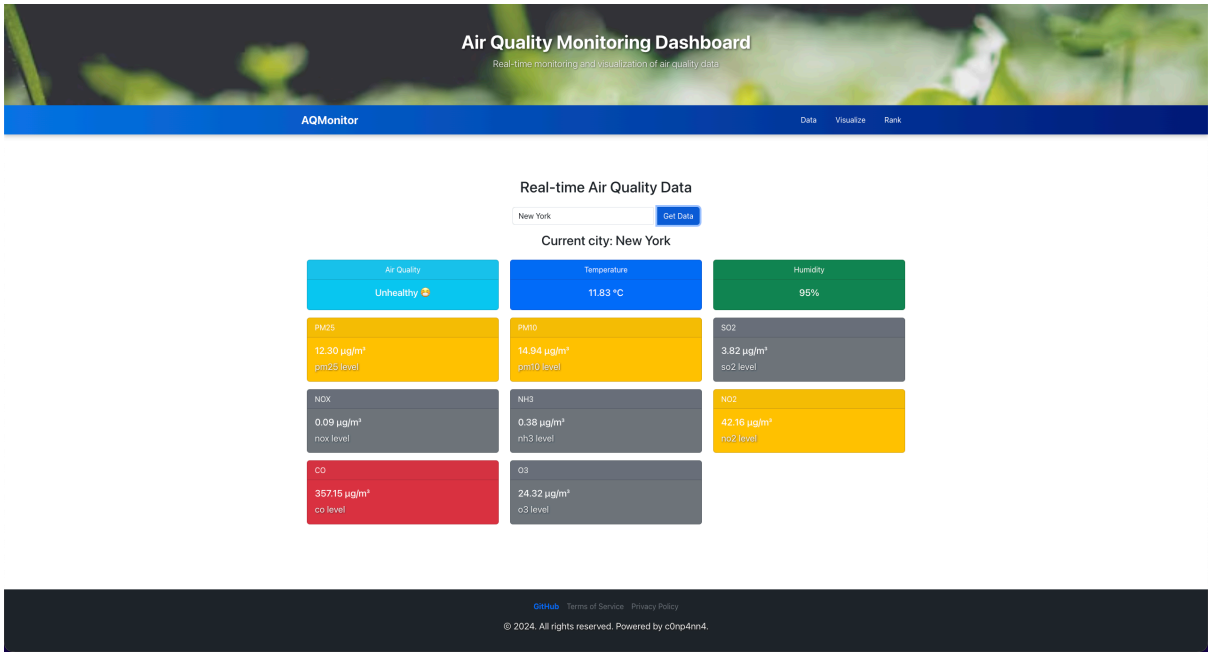
AQMonitor is a comprehensive web application designed to provide users with real-time air quality data for cities worldwide. It features interactive visualizations of key environmental metrics, global air quality rankings, and actionable insights to promote a healthier and more informed approach to environmental awareness. AQMonitor empowers individuals and organizations with user-friendly tools to monitor air quality and make informed decisions.



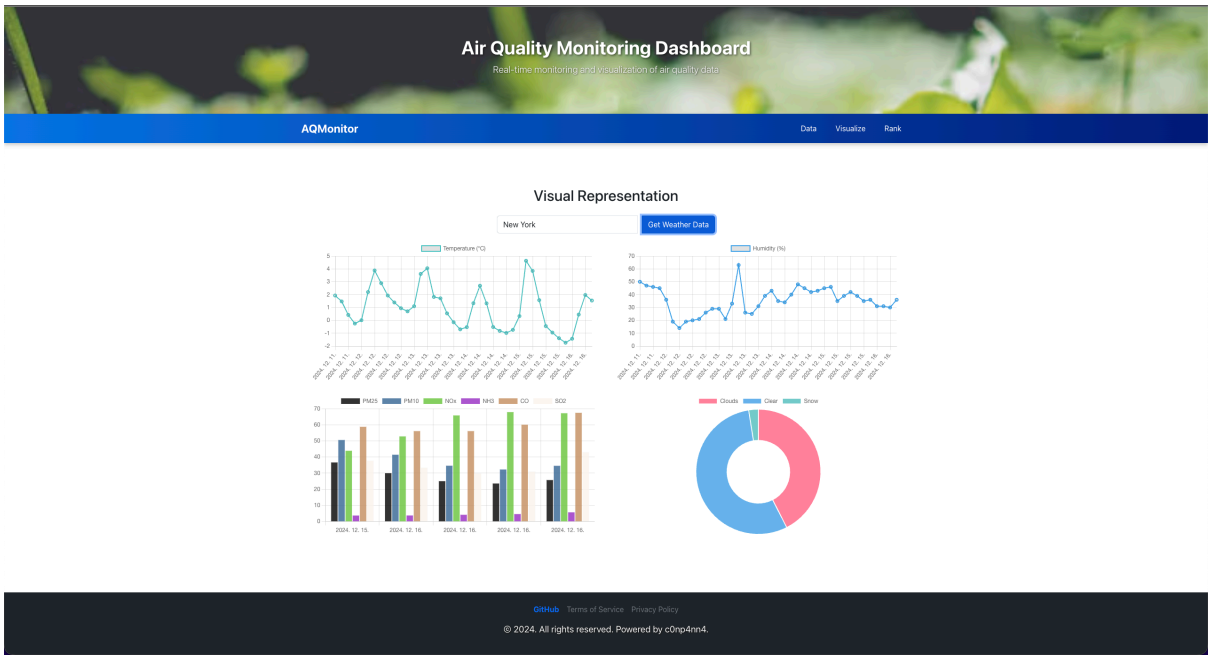
Features

The platform includes the following core features:

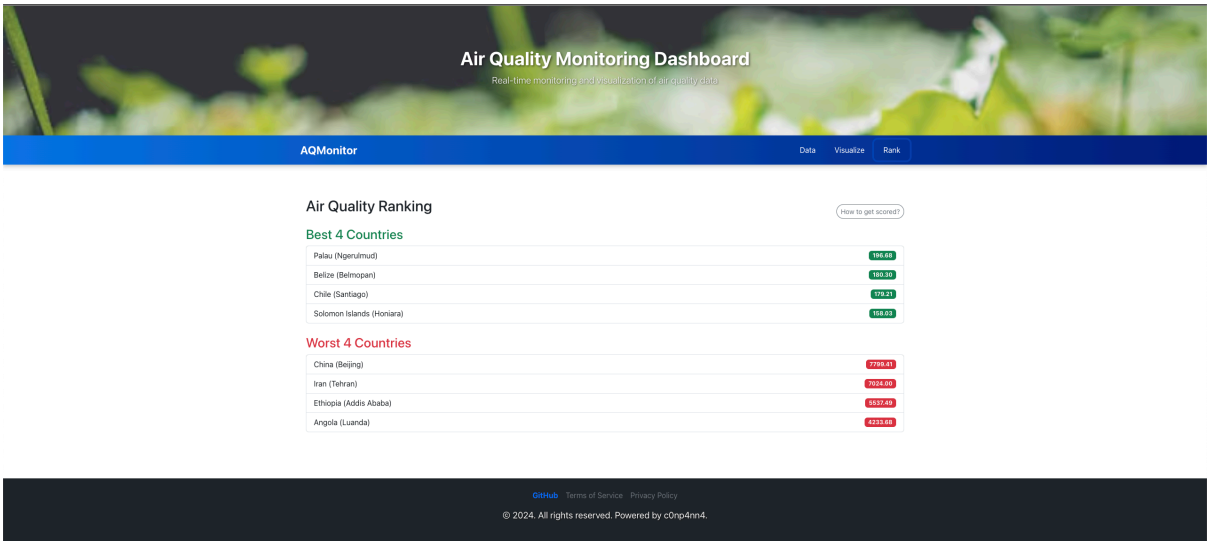
- Landing Page: A welcoming page that introduces the purpose and functionality of the platform.
- City Data Section: Allows users to fetch and view real-time air quality data for specific cities.



- **Visualize Section**: Provides interactive charts and graphs for analyzing air quality trends over time.



- **Ranking Section**: Displays global air quality rankings, enabling city-to-city comparisons.



The application is built with a responsive layout, ensuring usability across devices and delivering a modern, intuitive user experience.

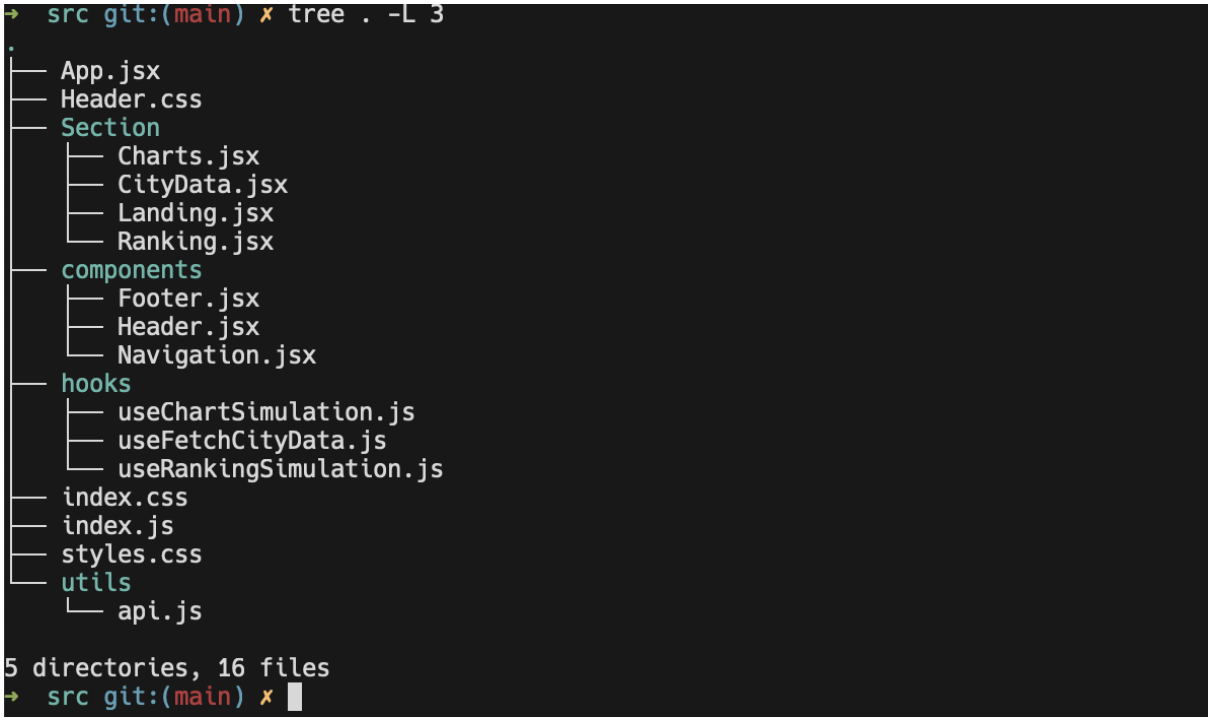
Technologies Used

AQMonitor was built using modern web development tools and libraries:

- **React.js**: For building a modular, interactive user interface.
- **Bootstrap**: To create responsive and visually appealing components.
- **Chart.js**: For rendering interactive charts and visualizations.
- **OpenWeatherMap API**: Integrated to fetch real-time weather and air quality data.
- **GitHub**: Used for version control and repository management.

Project Structure

The project structure was designed for maintainability and scalability:



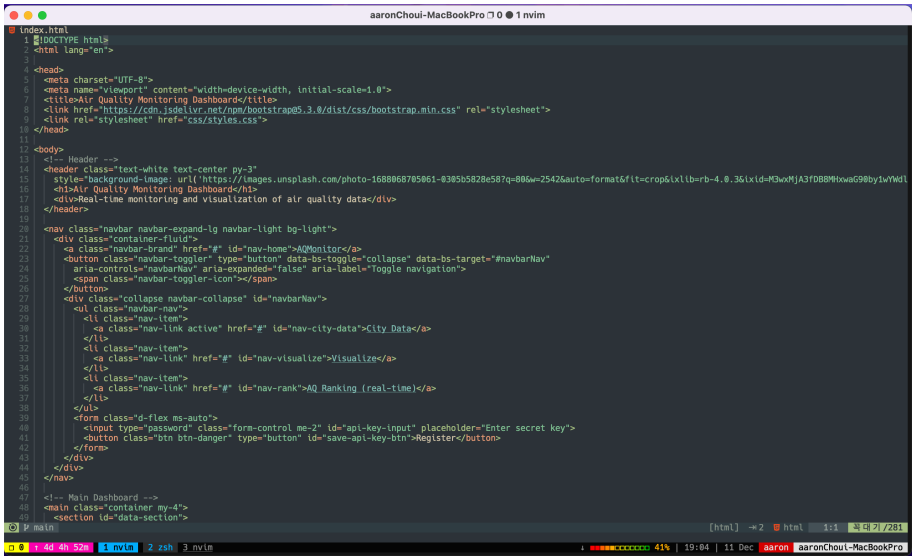
- App.jsx: The main application logic, handling navigation between sections.
- Section Folder: Contains components for individual sections:
  - Landing.jsx
  - CityData.jsx
  - Charts.jsx
  - Ranking.jsx
- Components Folder: Includes reusable elements:
  - Header.jsx
  - Navigation.jsx
  - Footer.jsx
- Hooks Folder: Modularizes logic for data fetching and chart rendering with hooks:
  - useChartSimulation.js
  - useFetchCityData.js
  - useRankingSimulation.js
- Utils Folder: Stores utility functions, such as API configuration in api.js.

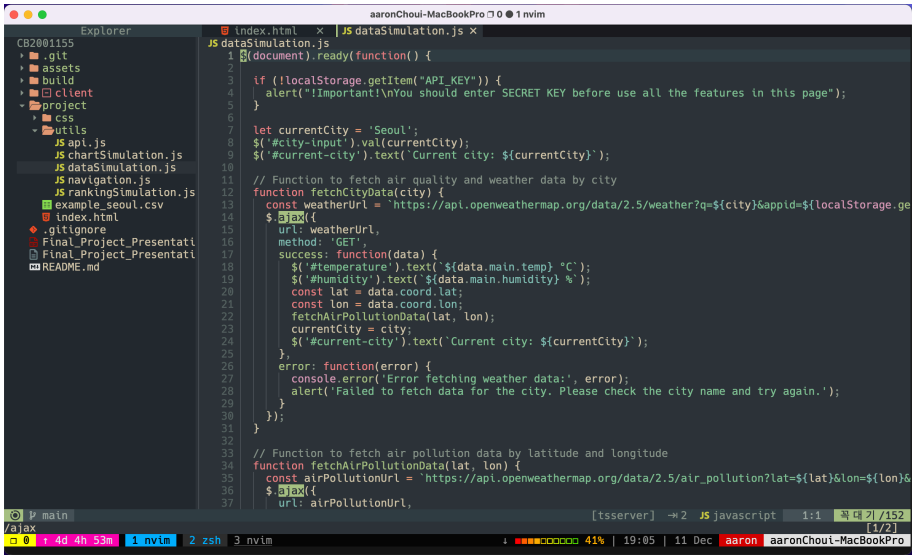
Development Process

1. Initial Development: HTML, CSS, and Vanilla JS

The project began as a simple prototype using HTML, CSS, and vanilla JavaScript. jQuery and AJAX were used to fetch data from the

OpenWeatherMap API, enabling basic functionality such as displaying air quality data for a city.





Tools and Techniques:

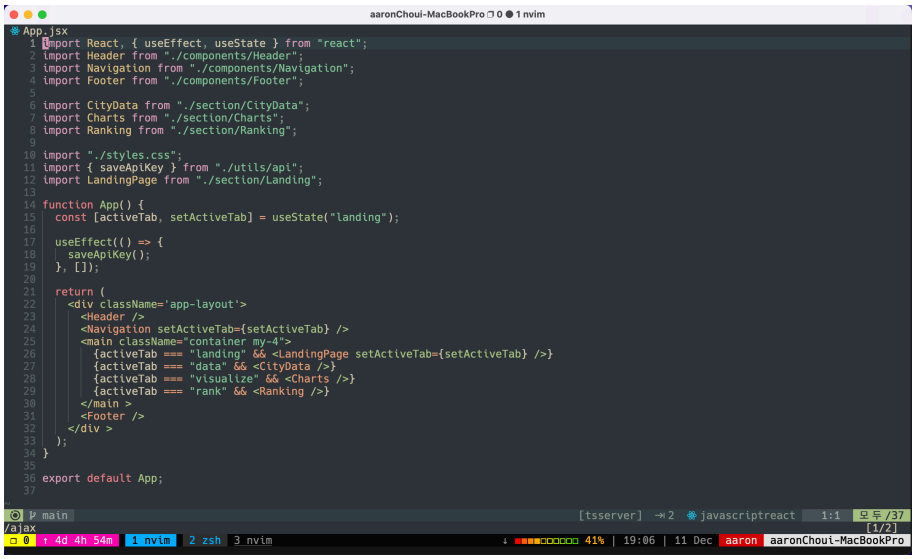
- HTML & CSS: For structuring and styling the application.
- JavaScript: For implementing dynamic functionality.
- jQuery: For efficient DOM manipulation and API calls.
- OpenWeatherMap API: To fetch real-time air quality and weather data.

2. Transition to React and Axios

As the project evolved, React was adopted to handle increasing complexity and ensure scalability. **Axios** replaced *jQuery* for API requests, offering a more

streamlined, promise-based approach. This transition introduced a component-based architecture, enabling modularity and reusability. State management and dynamic data updates were implemented using React's *useState* and *useEffect* hooks.

While integrating APIs, alternative weather-related APIs, such as **AccuWeather** and data from the **Korean Meteorological Administration**, were considered. However, the functionality provided by the **OpenWeatherMap** API was deemed sufficient for the current scope of the project. Thus, additional API integrations were not implemented, though their potential use for future expansions was acknowledged.



Tools and Techniques:

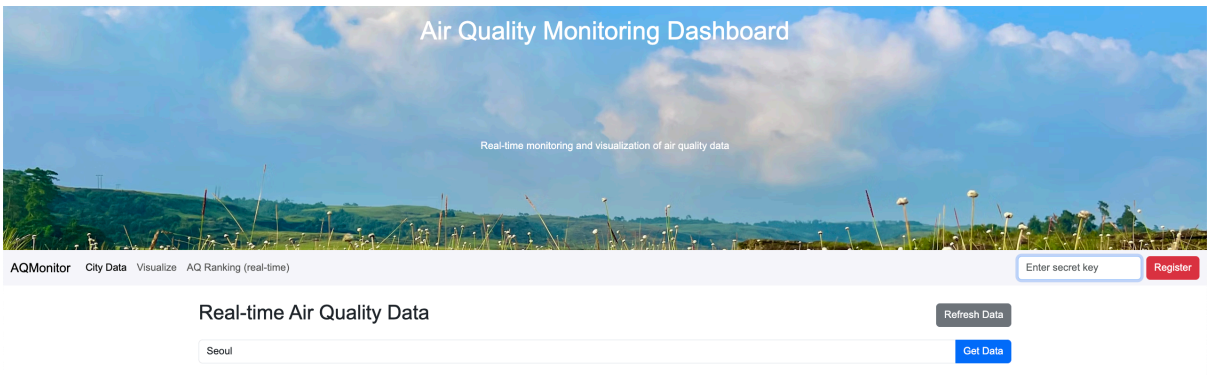
- React: For creating a modular, scalable, and responsive user interface.
- Axios: For efficient API integration and error handling.
- Component-Based Architecture: For improved code maintainability and reusability.
- CSS Enhancements: For modern animations, styling, and responsive design.

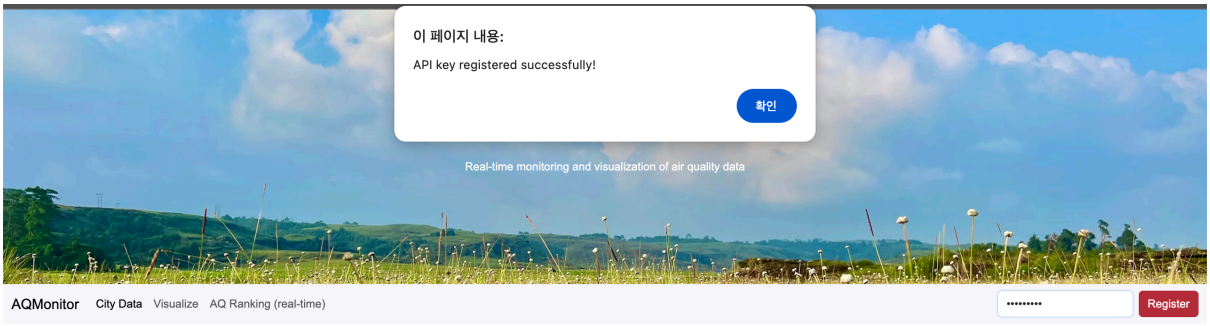
Challenges and Solutions

1. File Structure Refactoring
   Challenge: As the project grew, maintaining a clean and scalable file structure became difficult.
   Solution: Components, hooks, and utility functions were organized into separate folders, improving modularity and readability.

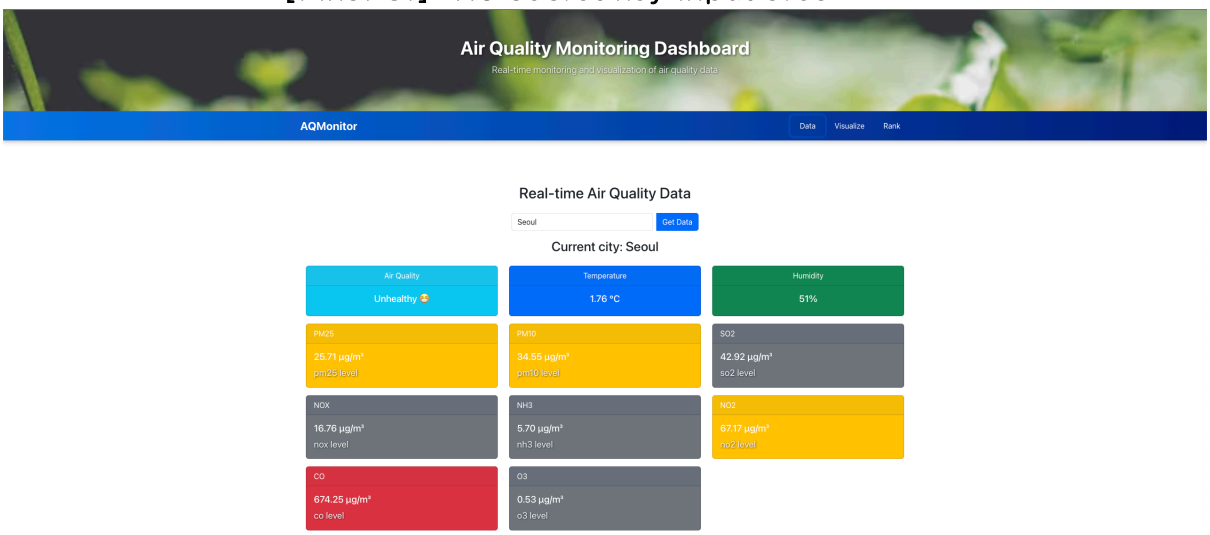
2. API_KEY Security vs. UX
   Challenge: Initially, AES encryption was used to secure the API_KEY, requiring users to handle a shared symmetric key. This approach ensured security but heavily impacted user experience (UX).
   Solution: The API_KEY was openly integrated into the project, relying on OpenWeatherMap's built-in API rate limits and security features. This decision prioritized UX by simplifying the user workflow.

[Prototype UI]: include secret key input area





[Final UI]: No secret key input area



3. **Responsive Design and UX Improvements**
   Challenge: Designing for various devices and screen sizes was challenging. Additionally, the Ranking Section's client-side data processing caused noticeable delays during data loading.
   Solution: CSS Flexbox and Grid were utilized to create responsive layouts, while media queries were added to ensure visual consistency across devices. For the Ranking Section, Skeleton Code was introduced to simulate loading states. This approach, inspired by platforms like YouTube and Twitter, improved UX by visually indicating that the data was still loading.

Achievements and Lessons Learned

AQMonitor successfully delivered a simple yet functional web application for real-time air quality monitoring. The project demonstrated the importance of structuring a frontend using modern frameworks like React, transitioning from

basic HTML, CSS, and JavaScript. Key features, such as modular components and state management with hooks, improved both scalability and maintainability.

By integrating the OpenWeatherMap API and using Axios for efficient data communication, the project emphasized the practical challenges of handling real-time external data. Designing responsive and user-friendly interfaces was another significant focus, achieved through CSS techniques and the addition of Skeleton Code to enhance the user experience during data loading.

Conclusion

AQMonitor is a straightforward platform that makes real-time air quality data accessible to users. The project highlights how even small applications can benefit from modern tools like React, efficient API integration, and thoughtful interface design. It serves as a stepping stone for developing more robust and scalable solutions while reinforcing the fundamentals of frontend development.