Air Quality Monitoring Dashboard

Final Project Report

Course: Web Application Programming (CB2001105-062)

Professor: Aich Satyabrata

Name: Byambasuren Tuvshinjargal

Student ID: 202355673

1.Objective

The **Air Quality Monitoring Dashboard** is designed to provide real-time weather and air quality data in a simple and interactive way. It helps users make informed decisions by combining real-time data, visualizations, and responsive design. The dashboard aims to raise environmental awareness and offer an easy-to-use tool for monitoring key metrics like temperature, humidity, and air pollutants.

2. Technologies Used

2.1. Frontend

- **HTML5**: Structured the web pages.
- **CSS3**: Styled the interface, ensuring responsiveness and modern layouts.
- **JavaScript**: Enabled interactivity and dynamic updates.

2.2. Libraries and Tools

- **Chart.js**: Created interactive visualizations such as line, bar, and doughnut charts.
- **Moment.js**: Formatted sunrise, sunset, and other time-based data.

2.3. APIs

- **OpenWeatherMap API**:
    - *Current Weather API*: Real-time weather metrics.
    - *Air Pollution API*: AQI levels and pollutant details.
    - *Geocoding API*: Converts city names to geographic coordinates.
    - *5-Day Forecast API*: Provides weather predictions.

Hosting

- **GitHub Pages**: Used to deploy the project for public access.

3. Features

Weather Monitoring

- Real-time metrics for:
    - Temperature (Celsius and "feels-like").
    - Humidity (%), wind speed (m/s), and visibility (km).
- Displays sunrise and sunset times in the local timezone.

Air Quality Monitoring

- Tracks pollutant levels:
    - PM2.5, PM10, SO2, CO, NOx, NH3, CO2.
- Color-coded AQI levels:
    - Green: Good, Yellow: Fair, Orange: Moderate, Red: Poor, Maroon: Very Poor.

Data Visualization

- **Line Chart**: Humidity and temperature at real time
- **Bar Chart**: Comparison of pollutant levels.
- **Doughnut Chart**: Percentage contributions of pollutants.

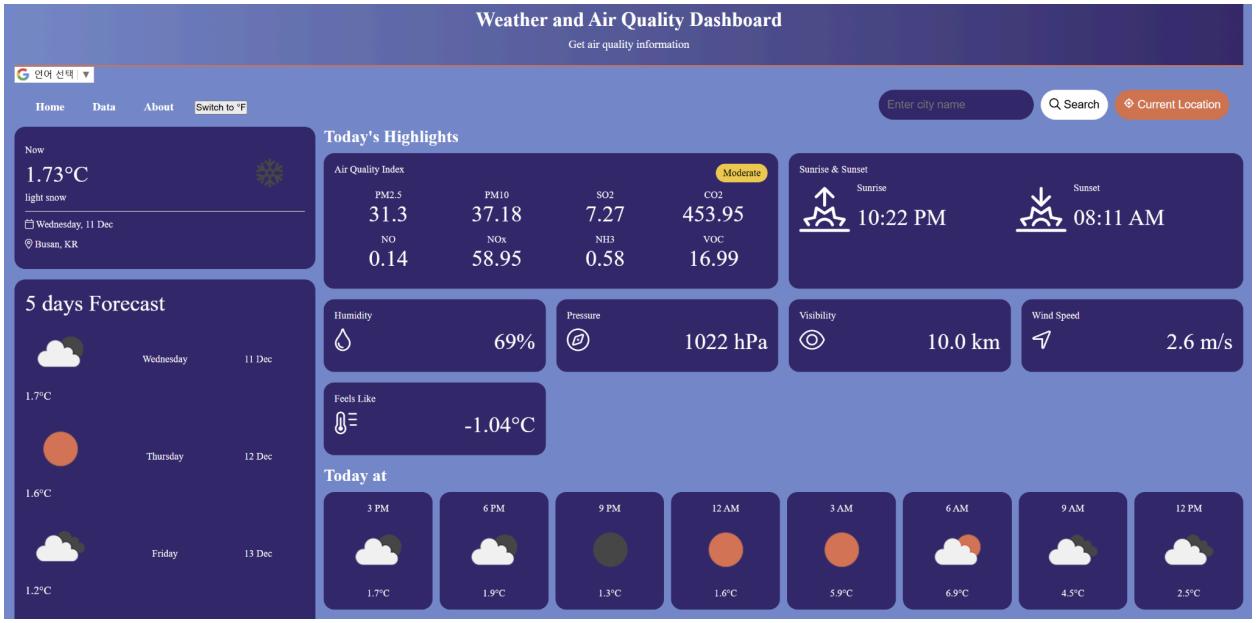
Search and Location Detection

- Search by city name or fetch location-based data using geolocation.

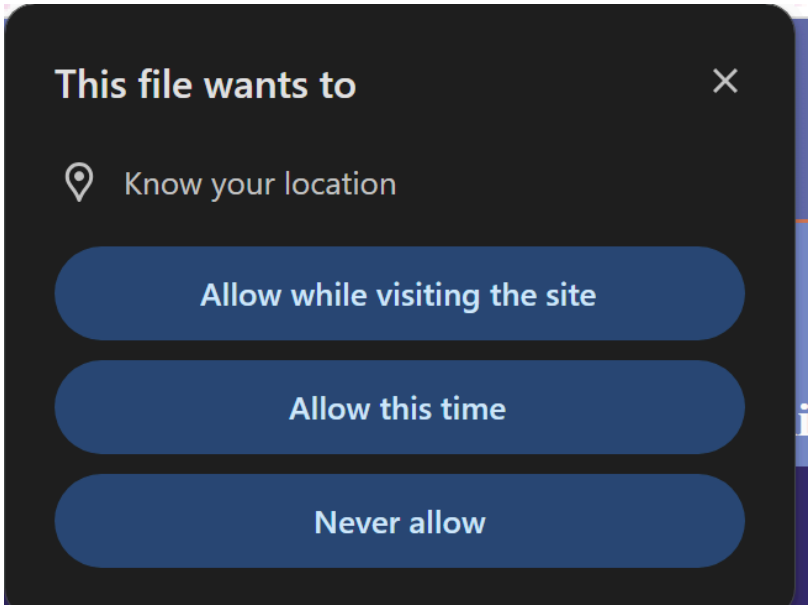
Responsive Design

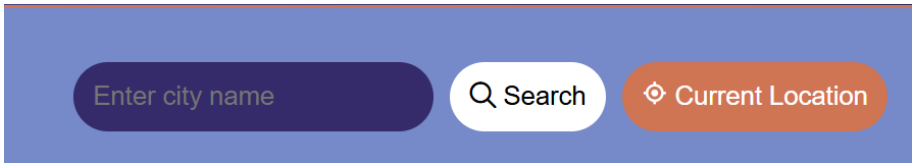
- Optimized layouts for devices of all sizes.

<u>INITIAL SCREEN</u>

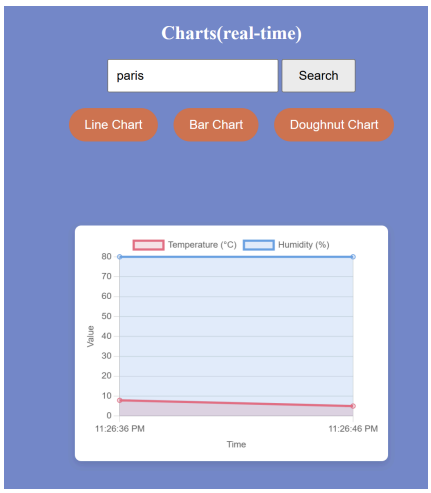


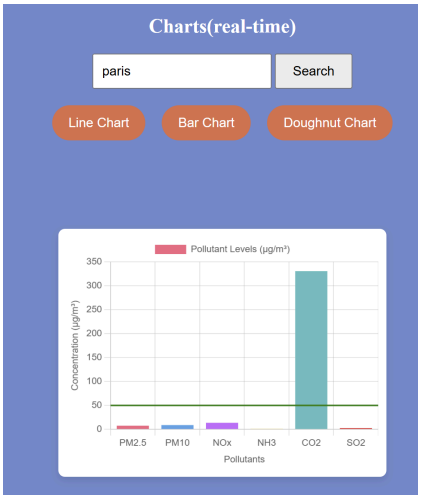
CURRENT LOCATION

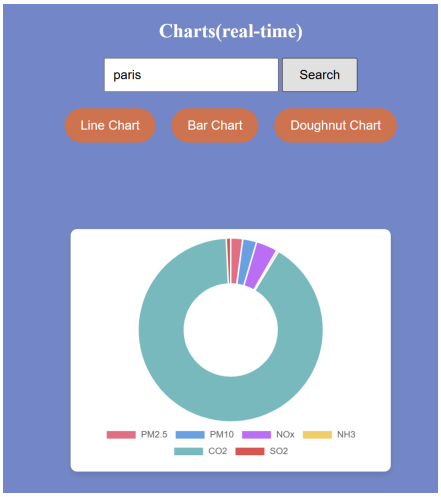




CHARTS

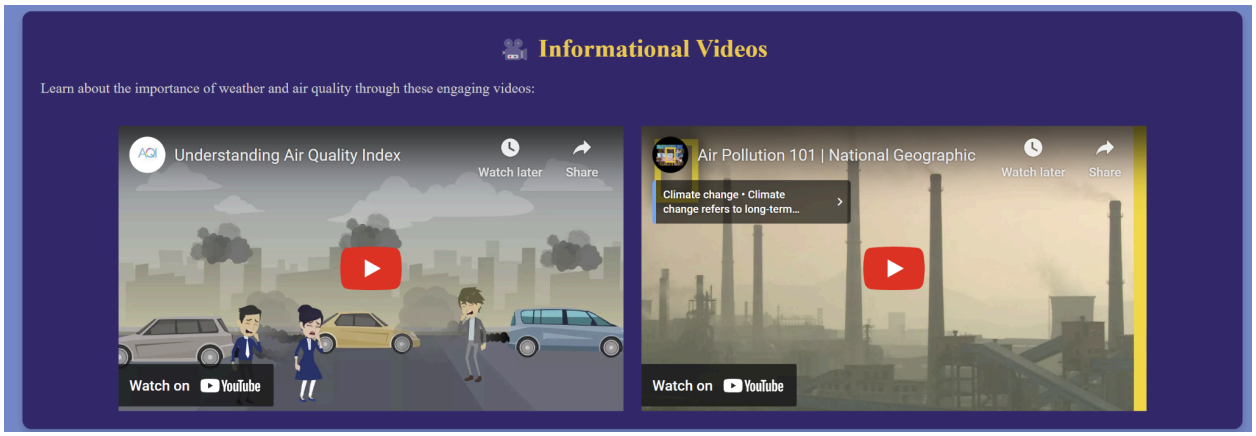






FOOTER

ABOUT



4. Challenges and Solutions

4.1. API Integration

- **Challenge**: Managing multiple endpoints for weather and AQI data.
- **Solution**: Modularized API calls to streamline fetching and processing.

4.2. Error Handling

- **Challenge**: Incomplete or missing API responses.
- **Solution**: Added fallback messages and a safeAccess function to handle nested properties.

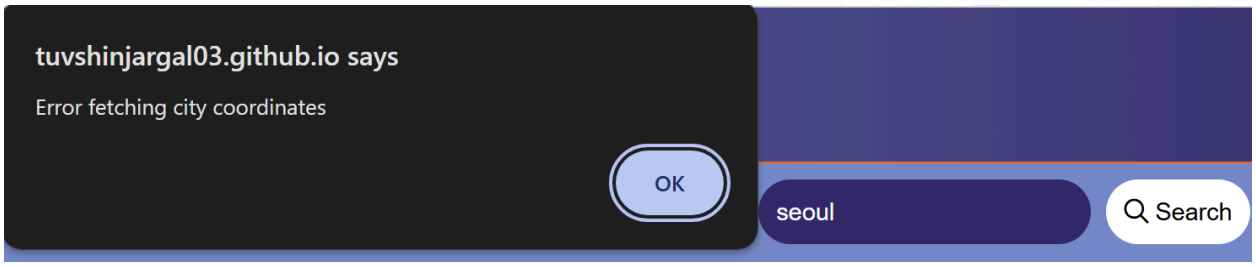
4.3 Translation feature

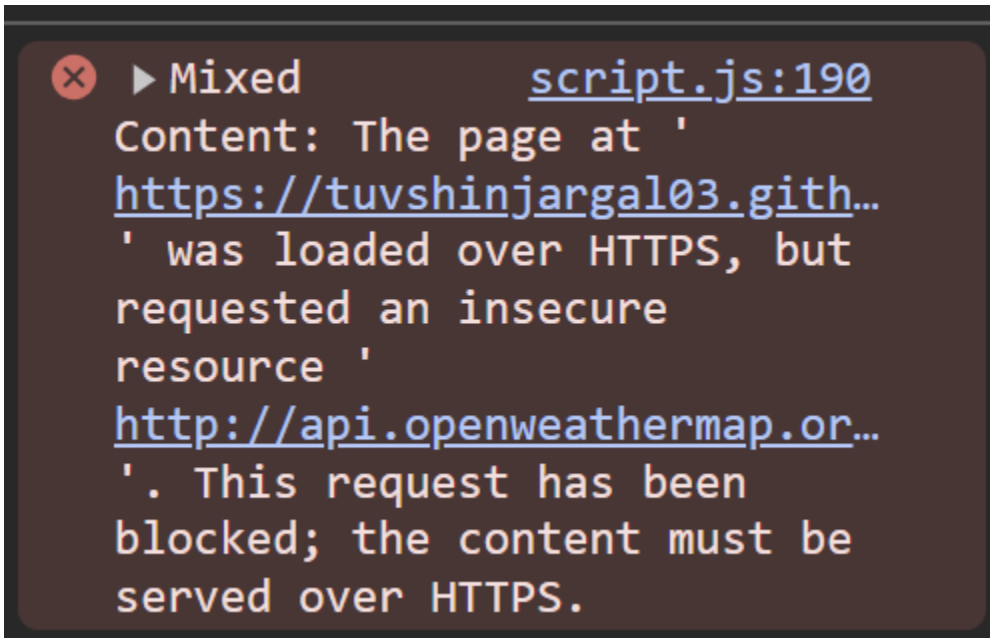
I initially tried to create a custom translation feature, but it didn't work as planned. Instead, I used Google Translate, which provided an easy and effective solution for making the content accessible in multiple languages.


```
<script type="text/javascript">
    function googleTranslateElementInit() {
        new google.translate.TranslateElement({
            pageLanguage: 'en',
            includedLanguages: 'en,fr,de,es,it,ja,ko,zh-CN,ru,ar,hi',
            layout: google.translate.TranslateElement.InlineLayout.SIMPLE,
            autoDisplay: false,
        }, 'google_translate_element');
    }
    // Load Google Translate API
    function loadGoogleTranslate() {
        var script = document.createElement('script');
        script.src = 'https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit';
        script.async = true;
        document.getElementsByTagName('head')[0].appendChild(script);
    }
    // Trigger Google Translate on document ready
    $(document).ready(function() {
        loadGoogleTranslate();
    });
</script>
```

4.4 Search Feature

When I first implemented the search feature, it didn't work because I used "http" instead of "https," making the connection insecure. After identifying the issue, I corrected it by switching to "https," which resolved the problem and allowed the feature to function properly






```
const GEOCODING_API_URL = `http://api.openweathermap.org/geo/1.0/direct?q=${cityName}&limit=1&appid=${api_key}`;
fetch(GEOCODING_API_URL)
    .then(res => res.json())
    .then(data => {
        if (data.length > 0) {
            const { name, lat, lon, country } = data[0];
            getWeatherDetails(name, lat, lon, country);
        } else {
            handleError(`City not found: ${cityName}`);
        }
    })
    .catch(() => handleError('Error fetching city coordinates'));
```

4.5 About Section

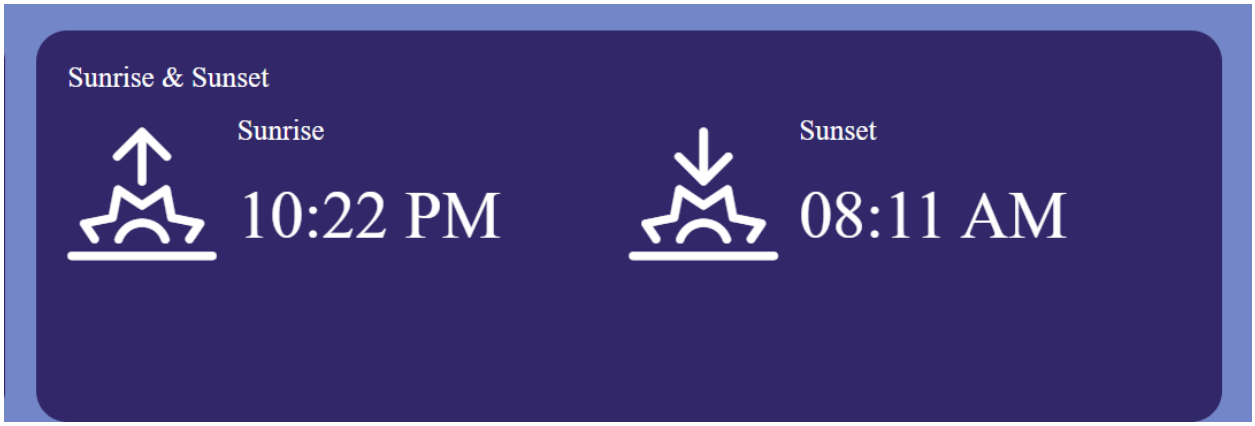
Initially, the "About" section focused solely on details about my project. I later updated it to include information related to air quality and other useful resources, making it more relevant and informative for users.

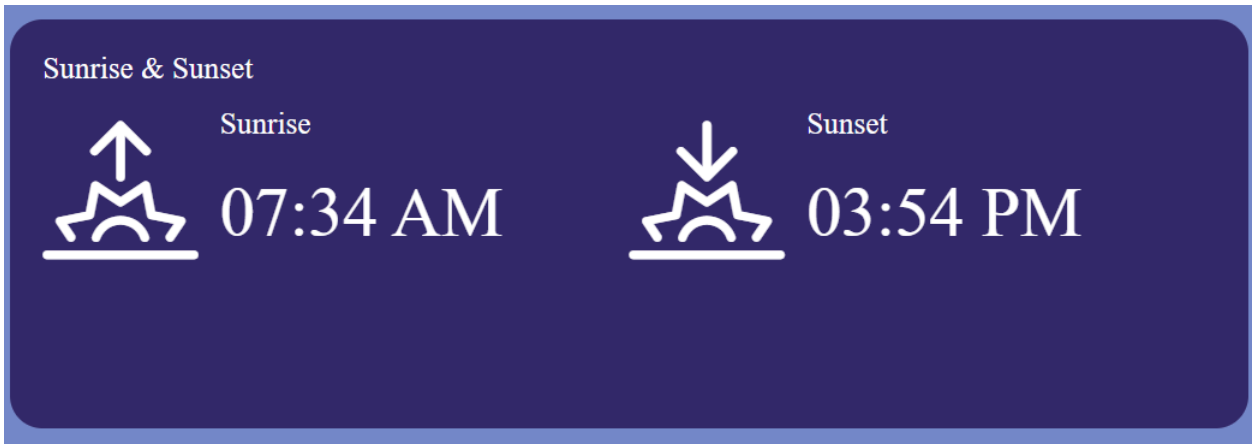
5. Successes

1. Real-time data integration with OpenWeatherMap APIs.
2. Intuitive and visually engaging charts using Chart.js.
3. A responsive design that adapts seamlessly to all devices.
4. Dynamic search and location-based data retrieval.

6. Limitations

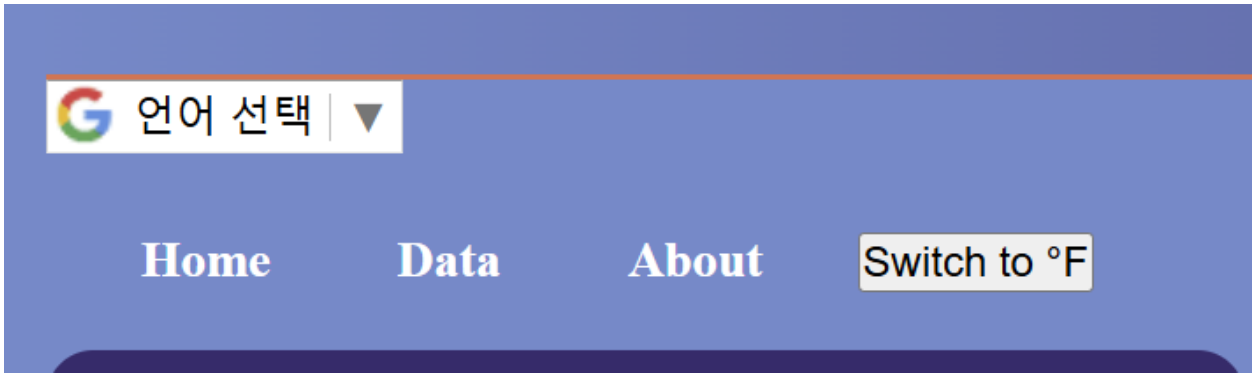
1. The issue with sunrise and sunset times not always working correctly may be due to incorrect handling of timezone offsets or inconsistent API data. My attempts to fix it failed.





2. Toggle button for unit conversion

   The toggle button for unit conversion was intended to allow users to switch between Celsius and Fahrenheit. However, I faced challenges in implementing real-time updates using fetched API data. While the button design and basic functionality were implemented, the dynamic updating of temperature values in real-time proved difficult, highlighting areas for future improvement in API integration and data handling.



7. Conclusion

The Air Quality Monitoring Dashboard has been a significant stepping stone in my journey as a student. It successfully combines real-time environmental data with interactive visualizations and a responsive design, providing valuable insights into air quality and weather conditions.Through this project, I gained a deeper understanding of key web development concepts, including front-end design, API integration, and dynamic data visualization. Despite some challenges, the experience strengthened my problem-solving skills and adaptability. Overall, this project not only showcases the potential of web technologies for public benefit but also lays a strong foundation for future improvements, making it a pivotal learning experience for me.

8. References

1. OpenWeatherMap API Documentation: https://openweathermap.org/api
2. Chart.js Documentation: https://www.chartjs.org/docs
3. Moment.js Documentation: https://momentjs.com

- **GitHub Repository Link:** https://github.com/Tuvshinjargal03/Web-Application-Programming-project
- **Final Demo Link:** https://tuvshinjargal03.github.io/Web-Application-Programming-project/