Project Report: Weather Visualization Web App

1. Project Overview

This project is a browser-based weather visualization tool that provides:

- Daily weather forecasts for the current and following days. Tidal 8-day forecasts.
- Today , tomorrow hourly temperature
- Live air quality index (AQI)
- Live air UV index
- Other details: windspeed, sunrise, sunset, presipitation
- Settleligh images from for interactive maps
- Historical climate data visualization from 1940-2024 (last 80+ years) - graph
- City search functionality to view data worldwide

2. Tools and APIs Used

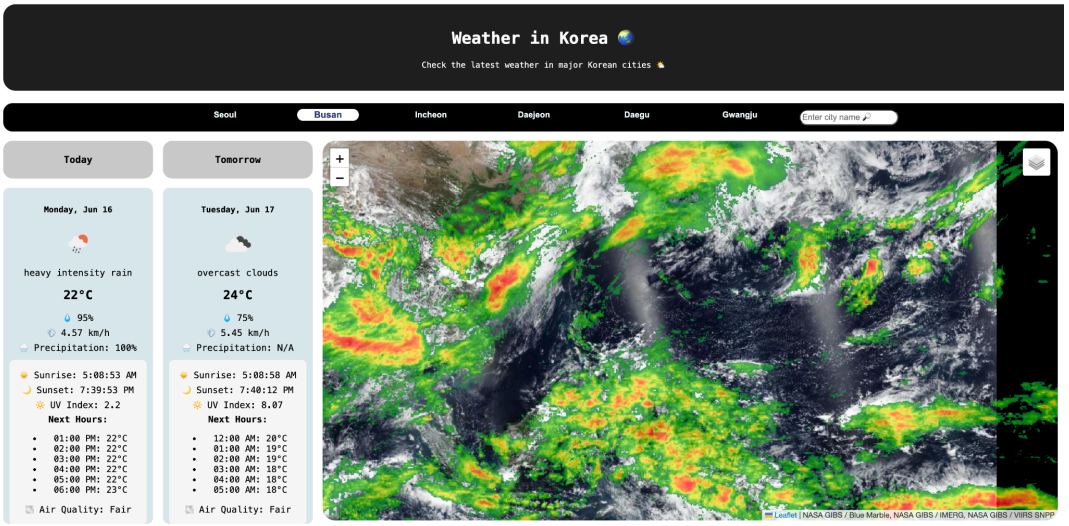
The application is written entirely in JavaScript, HTML, and CSS

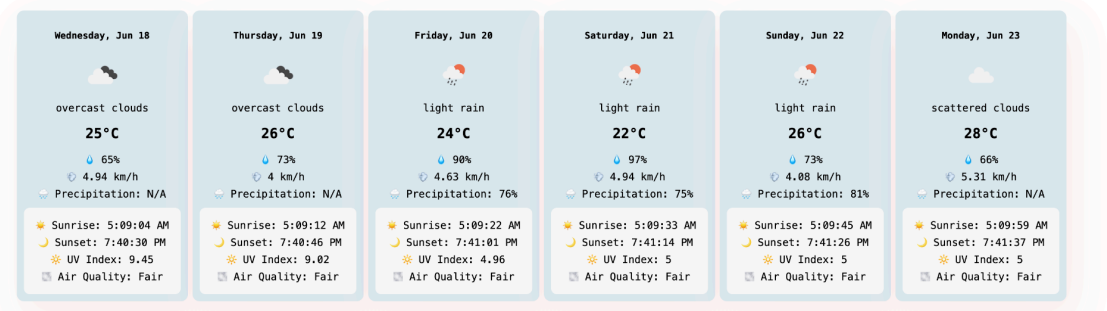
Tool/API	Purpose
OpenWeatherMap API (3.0) and (2.5)	Current, hourly, and daily weather forecasts
OpenWeatherMap Air Pollution	Provides AQI data by geographic coordinates
OpenWeatherMap Geocoding API	Converts city names to coordinates for further queries
NASA GIBS (WMTS)	Supplies satellite imagery overlays for maps
Open -Meteo API	Historica weather data from 1940-2024
Leaflet.js	Handles interactive maps, layers, and overlay toggling
Chart.js	Renders historical temperature data charts

---

3. Implementation Summary

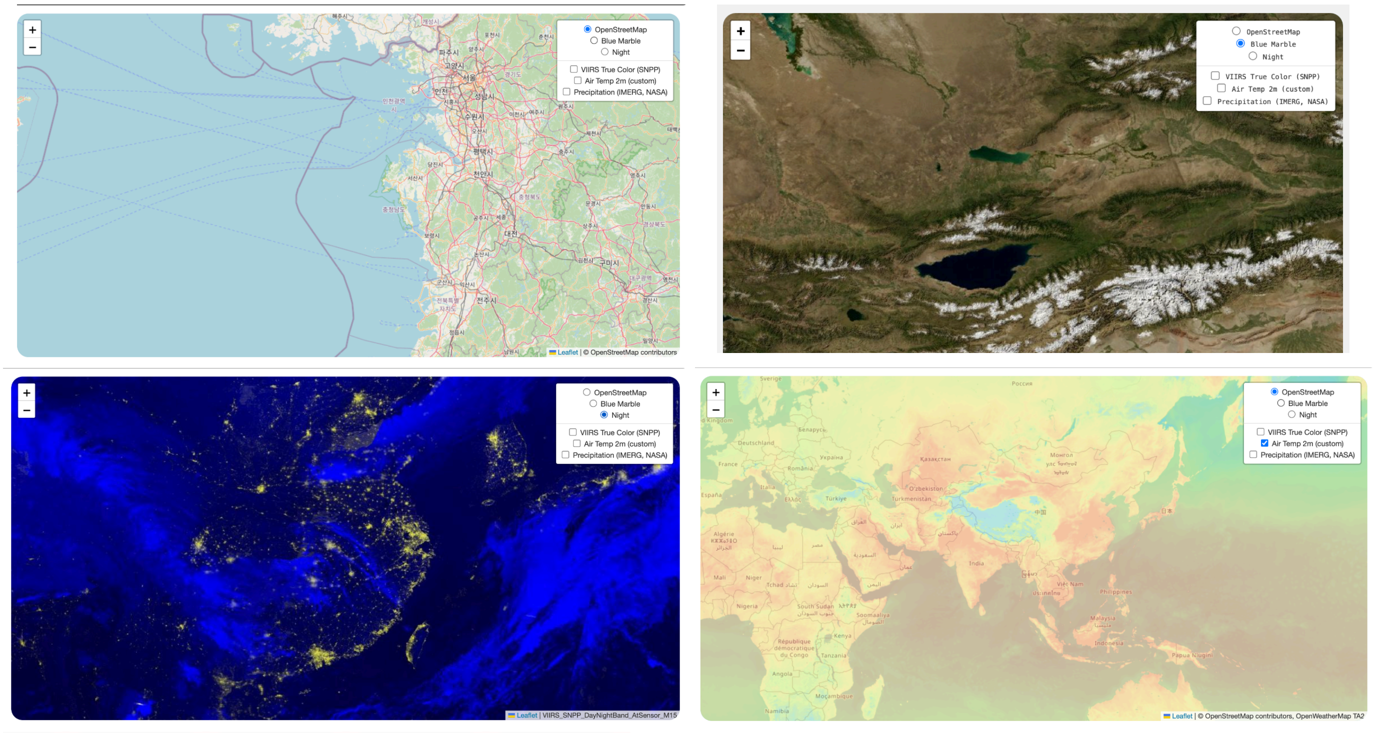
- Weather forecast cards are generated dynamically using data from OpenWeatherMap's OneCall API 3.0 and 2.5
- The user interface is segmented into 6 main parts:
    - Title
    - City selection and search buttons
    - Today's and Tomorrow's forecast (detailed)
    - Map
    - Extended 6-day forecast
    - Historical chart





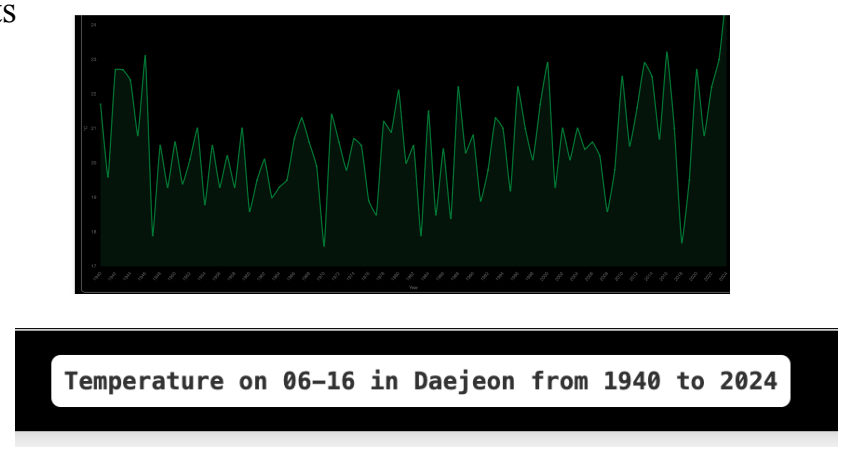
- Each forecast card includes:
    - Weather condition icon and description
    - Daily average temperature, humidity, wind speed, UV index
    - Sunrise and sunset times
    - Precipitation probability
    - AQI (air quality index), fetched separately and added after card generation
    - Hourly temperature breakdown for the current and next day (first two cards only)

- Map initialization is handled via Leaflet.js with three selectable
  - **Base layers:**
    - OpenStreetMap - Names of countries and cities (static image)
    - NASA Blue Marble - Day view relief (static image)
    - NASA Black Marble - Night View (static image)
  - **Overlays can be toggled on the map:**
    - VIIRS True Color - Clouds conditions(updates evry 24 hours)
    - Air Temperature 2m (updates evry 24 hours)
    - Precipitation Rate (IMERG) (updates evry 24 hours)



- City search is powered by the OpenWeatherMap geocoding API and supports autocomplete

- Historical data visualization:
  - Uses Open-Meteo's archive API
  - Filters daily mean temperatures from 1940 to the previous full year for the current calendar date
  - Data is visualized using a responsive line chart with labeled axes and interactive points



4. Functionalities

- Dynamic creation of weather cards based on selected city
- AQI data integration for each forecast card
- Smooth user experience for switching between predefined cities and searched cities
- Interactive map with switchable base layers and togglable overlays
- Hourly previews for today and tomorrow, properly matched to forecast date
- Reliable integration of historical data from 1940 through the last complete year
- Real-time search with suggestions and dropdown list for city selection
- Dynamic chart rendering with clear labeling and styled visuals

---

6. Future Improvements

- Add Ocean weather and data for tides
- Expand historical visualizations to include humidity, wind speed, and precipitation trends
- Emergency alarms
- Agricultural data

7. Notes

If you cannot see the **precipitation layer** you are probably too close to the map. You need to **zoom out** until you can see the layer.

8. Conclusion

This web application showcases the integration of multiple weather and geospatial APIs to deliver an interactive and informative weather exploration experience. By combining current forecast data, live air quality readings, map overlays, and long-term historical climate records, it enables users to engage with weather trends in both real-time and over a historical timeline. The project serves as an example of how public APIs and client-side visualization tools can be used to build accessible and data-rich web applications.