|  |
| --- |
| **Front End Engineering-II**        Project Report      Semester-IV (Batch-2022)            Weather-App    **Supervised By: Submitted By:**    Raveesh Samkaria Suraj adde  Roll Number: -2210990876    Group - 13            **Department of Computer Science and Engineering**  **Chitkara University Institute of Engineering & Technology,**  **Chitkara University, Punjab** |

# Abstract

The WeatherApp is a web-based application designed to provide users with real-time weather information and forecasts for cities worldwide. Utilizing the OpenWeatherMap and World Air Quality Index APIs, the WeatherApp fetches data such as temperature, humidity, wind speed, air quality index (AQI), and visibility based on user-selected cities. The application offers a user-friendly interface where users can toggle between Celsius and Fahrenheit temperature units, view current weather conditions, and access detailed weather forecasts for the upcoming days.

**Key features of the WeatherApp include:**

* Real-time weather data retrieval from OpenWeatherMap API.
* Air quality information sourced from World Air Quality Index API.
* Dynamic background image changes based on weather conditions.
* Ability to switch between Celsius and Fahrenheit temperature units.
* Detailed weather forecasts for the next several days.
* Dark and light mode themes for user preference.
* Interactive city selection for weather updates.

The WeatherApp aims to provide users with a comprehensive and visually appealing weather experience, making it easy to stay informed about weather conditions in their desired locations.

# Table of Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.no** |  | **Section** | **Page No.** |
|  | **1** | **Introduction** | 4 |
|  | **2** | **Problem Statement** | 5 |
|  | **3** | **Technical Details** | 6 |
|  | **5** | **File Structure** | 7 |
|  | **6** | **Result** | 8-25 |
|  | **7** | **References** | 26 |
|  |  |  |  |

|  |
| --- |
|  |

## Introduction

In today's fast-paced world, staying updated with weather conditions is crucial for planning daily activities, travel, and outdoor events. The WeatherApp project addresses this need by offering a user-friendly web application that provides real-time weather data and forecasts for cities worldwide. This introduction outlines the purpose, functionality, and significance of the WeatherApp.

The primary objective of the WeatherApp is to empower users with accurate and timely weather information, sourced from reliable APIs such as OpenWeatherMap and World Air Quality Index. By integrating these APIs, the WeatherApp retrieves essential weather parameters like temperature, humidity, wind speed, air quality index (AQI), visibility, and weather forecasts. Users can easily access this data by searching for their desired cities within the application.

One of the key features of the WeatherApp is its dynamic and interactive interface. Users can toggle between Celsius and Fahrenheit temperature units, allowing them to customize their experience based on personal preferences. Additionally, the application offers dark and light mode themes, enhancing usability and accessibility for users across different devices and environments.

The WeatherApp's intuitive design includes features such as background image changes based on weather conditions, detailed weather forecasts for multiple days, and interactive city selection for quick weather updates. These features collectively contribute to a seamless user experience, making it effortless for individuals to stay informed about weather patterns and make informed decisions accordingly.

Overall, the WeatherApp serves as a valuable tool for anyone seeking reliable weather information, whether for daily planning, travel, outdoor activities, or general curiosity. Its user-centric design, coupled with accurate data retrieval and interactive functionalities, positions the WeatherApp as a versatile and indispensable resource in today's digital landscape.

|  |
| --- |
| 4 |

## Problem Statement

Despite the availability of numerous weather forecast platforms and applications, many users still face challenges in accessing accurate, up-to-date, and user-friendly weather information. Existing weather apps may lack certain features, provide complex interfaces, or have limited coverage for specific locations. These issues create a demand for a comprehensive and intuitive weather application that caters to diverse user needs and preferences.

The problem statement for the WeatherApp project encompasses the following key challenges:

**1. Limited Accessibility:** Many weather applications may not provide coverage for all cities or regions worldwide, leading to gaps in weather information for users in specific locations.

**2. Complex Interfaces:**Some weather apps may have cluttered or confusing interfaces, making it challenging for users to navigate and access the desired weather data efficiently.

**3. Inaccurate or Outdated Data:** Users often encounter discrepancies or delays in weather information, impacting the reliability and timeliness of their decision-making based on weather forecasts.

**4. Lack of Customization:** Users may have limited options to customize their weather experience, such as choosing between different temperature units (Celsius or Fahrenheit) or adjusting the app's theme for better readability.

**5. Insufficient Weather Details:** While basic weather parameters like temperature and humidity are commonly available, users may require more detailed information such as air quality index (AQI), visibility, and comprehensive weather forecasts for multiple days.

Addressing these challenges is essential to develop a robust and user-centric weather application that meets the diverse needs of users across different locations, preferences, and usage scenarios. The WeatherApp aims to bridge these gaps by providing a seamless, accessible, and feature-rich platform for users to access accurate and comprehensive weather data effortlessly.

|  |
| --- |
| 5 |

FEE-II, 22CS0014

## Technical Details

**1. Frontend Development:** The WeatherApp's frontend is developed using HTML, CSS, and JavaScript. HTML provides the structure of the web page, CSS handles the styling and layout, and JavaScript adds interactivity and dynamic functionality to the user interface.

**2. Responsive Design:** The application is designed to be responsive, ensuring optimal viewing and usability across various devices and screen sizes. This is achieved using CSS media queries and responsive design principles.

**3. External APIs:** The WeatherApp integrates with external APIs to fetch weather data and air quality information. It utilizes the OpenWeatherMap API for current weather data and forecast, as well as the World Air Quality Index (WAQI) API for air quality information.

**4. API Key Management:**To access the external APIs securely, the application utilizes API keys provided by OpenWeatherMap and WAQI. These keys are stored securely and handled according to best practices to prevent unauthorized access.

**5. User Input Handling:** The application allows users to input their desired location (city) for weather information. It includes features such as a search input field and a submit button to initiate the weather data retrieval process based on user input.

**6. Temperature Units Conversion:** WeatherApp includes functionality to switch between temperature units (Celsius and Fahrenheit) based on user preference. This feature is implemented using JavaScript to perform the necessary temperature conversions.

**7. Dynamic Content Updates:** The user interface dynamically updates content such as temperature, weather descriptions, humidity, wind speed, and visibility based on the retrieved weather data from the API responses. This ensures that users receive real-time and accurate weather information.

**8. Background Image Updates:** WeatherApp enhances user experience by dynamically updating the background image of the application based on the current weather conditions (e.g., clear sky, clouds, snow). This feature adds visual appeal and context to the weather information displayed.

**9. Dark Mode Support:**The application includes a dark mode feature, allowing users to toggle between light and dark themes for improved readability and visual comfort. This functionality is implemented using CSS classes and JavaScript event handling.

**10. Error Handling:** WeatherApp incorporates robust error handling mechanisms to manage scenarios such as invalid user inputs, failed API requests, or data retrieval errors. It provides informative error messages and alerts to guide users and maintain application stability.

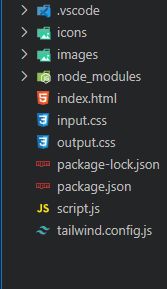
**11. Optimized Performance:**The application is optimized for performance, including efficient data fetching, minimal page reloads, and optimized asset loading. This ensures a smooth and responsive user experience even under varying network conditions.

Overall, WeatherApp combines frontend technologies, external API integration, user input handling, dynamic content updates, and enhanced features to deliver a reliable, user-friendly, and feature-rich weather application for users worldwide.

## File Structure: -

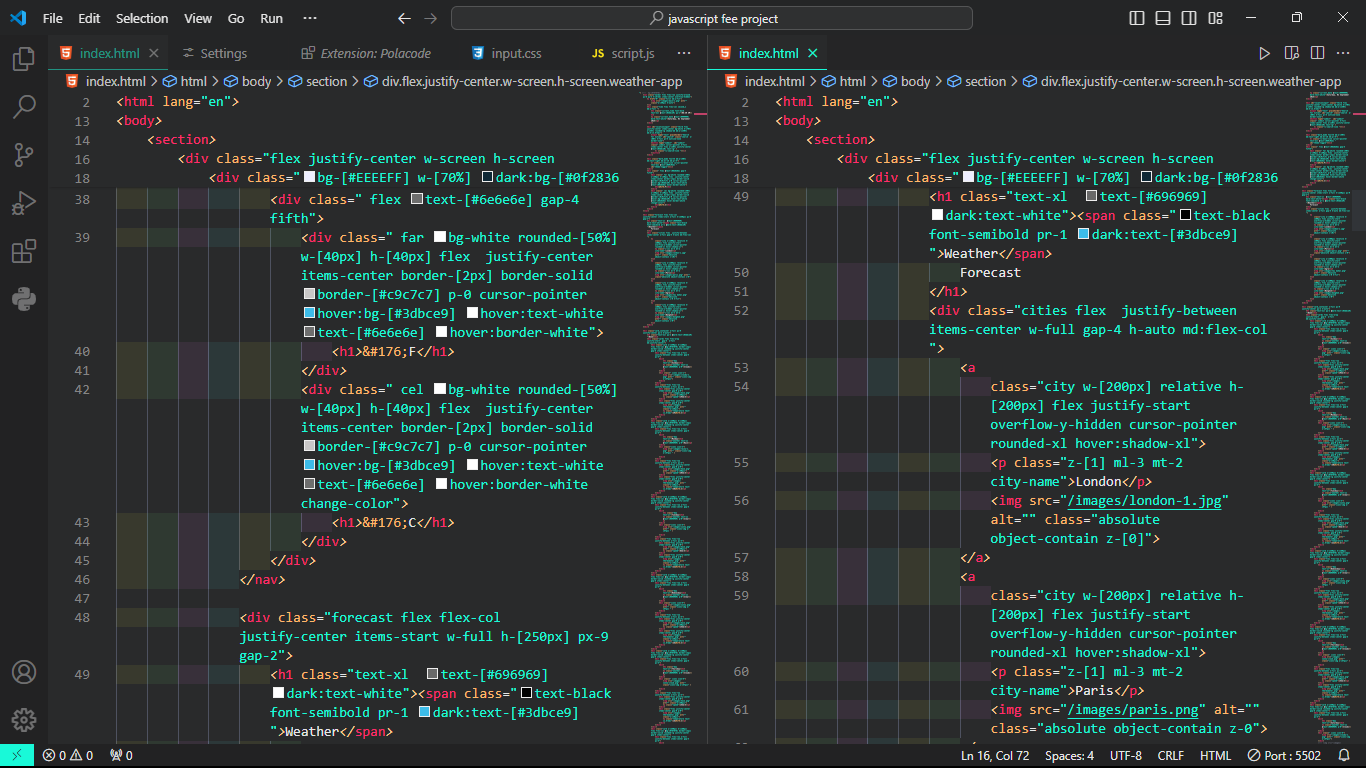
* **index.html:** This is the main HTML file that serves as the entry point for your WeatherApp. It contains the structure of the web page, including elements such as navigation bars, input fields, weather forecast cards, and highlights sections.
* **input.css:** This CSS file contains styles specifically related to user input elements, such as search boxes, buttons, and form elements. It helps to enhance the visual appearance and functionality of input components in the WeatherApp.
* **icons folder:**This folder contains various icons used throughout the application, such as search icons, weather icons (clouds, rain, sunny), temperature icons, and other visual elements. These icons are typically in image format (e.g., PNG, SVG) and are referenced in the HTML and CSS files.
* **images folder:** The images folder stores images used in the WeatherApp, such as background images for different weather conditions (clear sky, clouds, snow), city images for forecast cards, and other graphical assets. These images contribute to the overall visual presentation and context of weather information.
* **script.js:** This JavaScript file contains the client-side logic and functionality for the WeatherApp. It handles tasks such as fetching weather data from APIs, updating the user interface dynamically, managing user interactions (e.g., toggling between Celsius and Fahrenheit), handling API responses, and error handling.
* **output.css:**This CSS file contains styles related to the overall layout, design, and appearance of the WeatherApp. It includes styles for elements such as navigation bars, forecast cards, highlights sections, typography, color schemes, and responsive design for different screen sizes and devices.

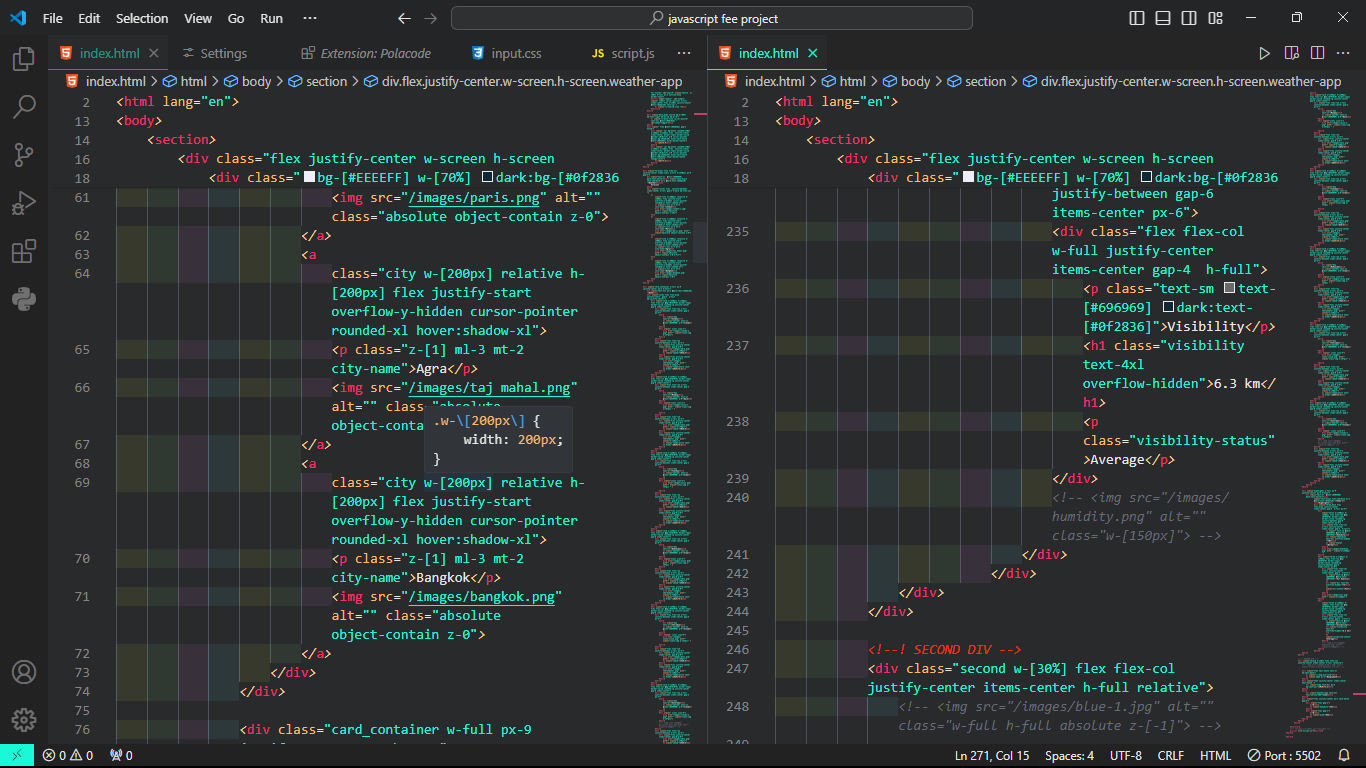
This file structure organizes the various components (HTML, CSS, JavaScript, images, icons) of my WeatherApp project, making it easier to manage, maintain, and update different aspects of the application.

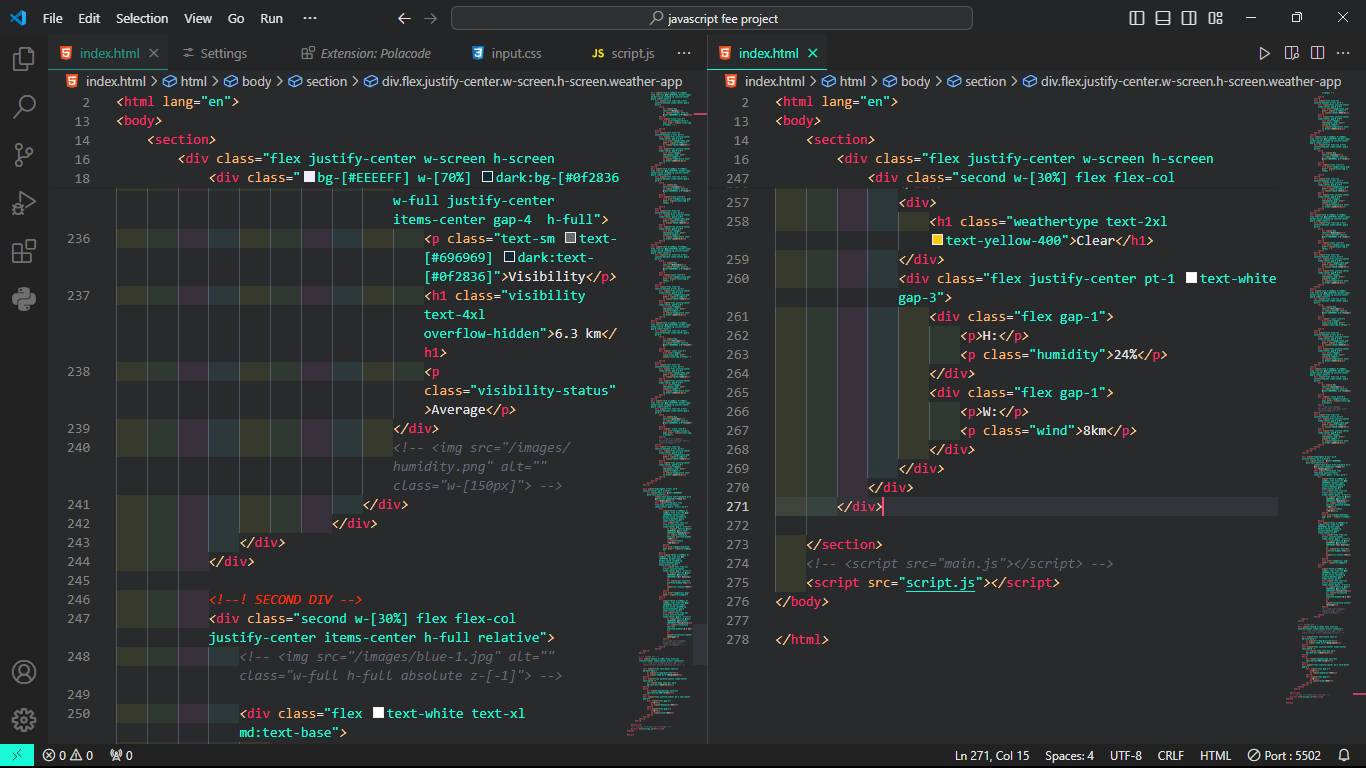


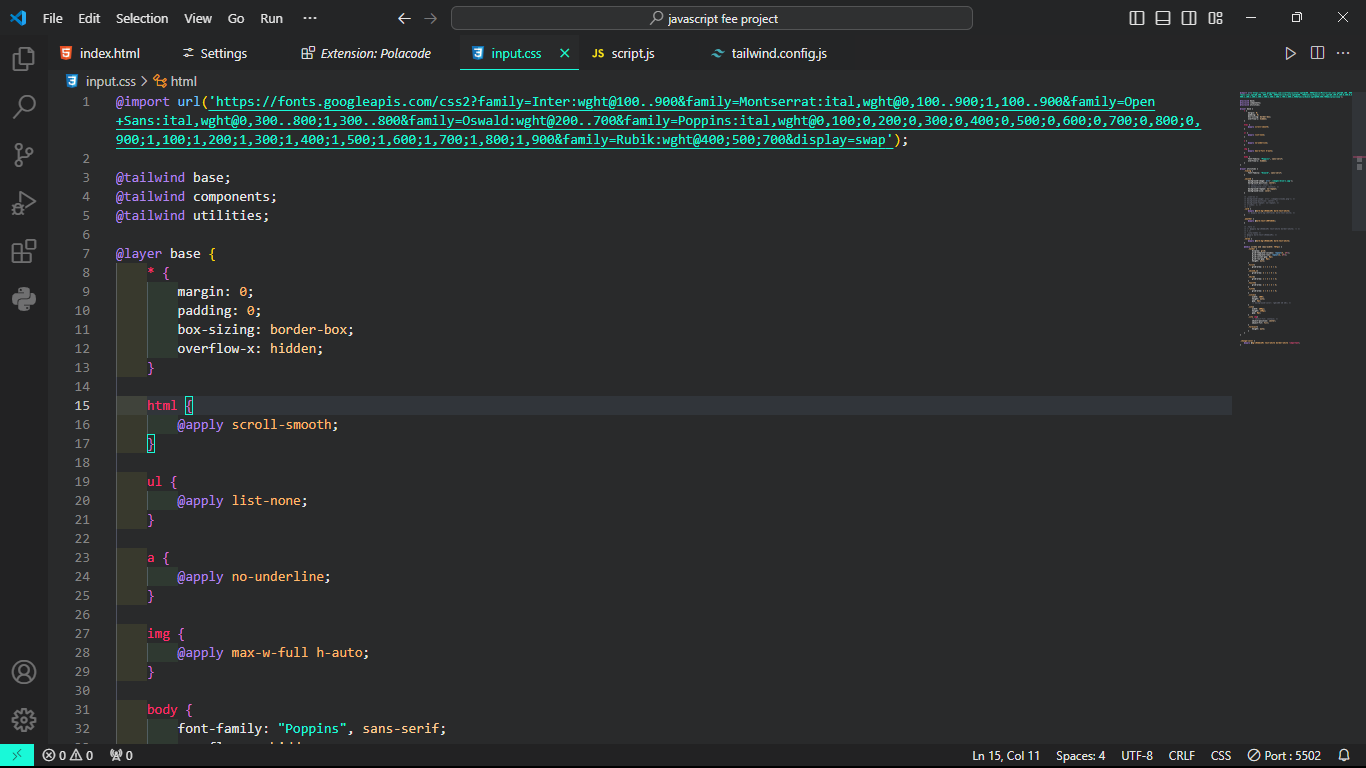
**Result**

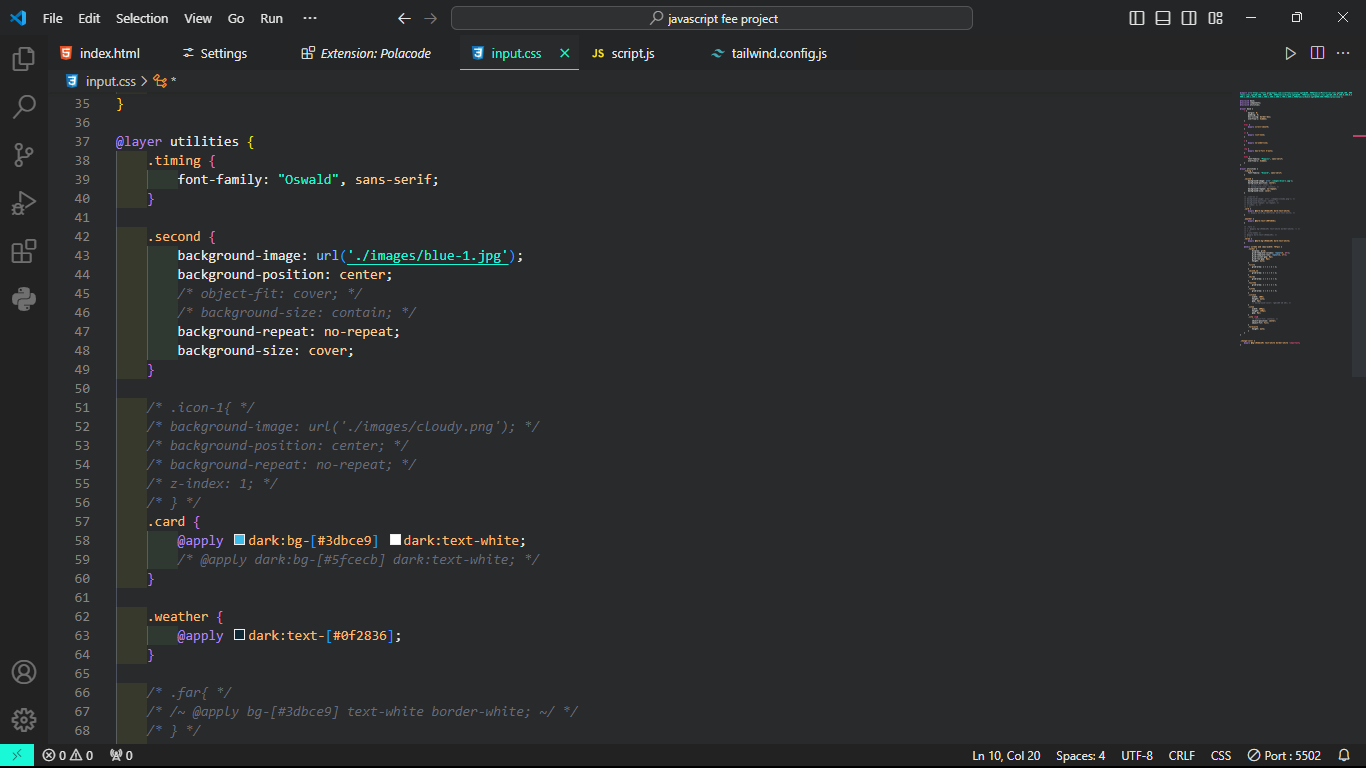


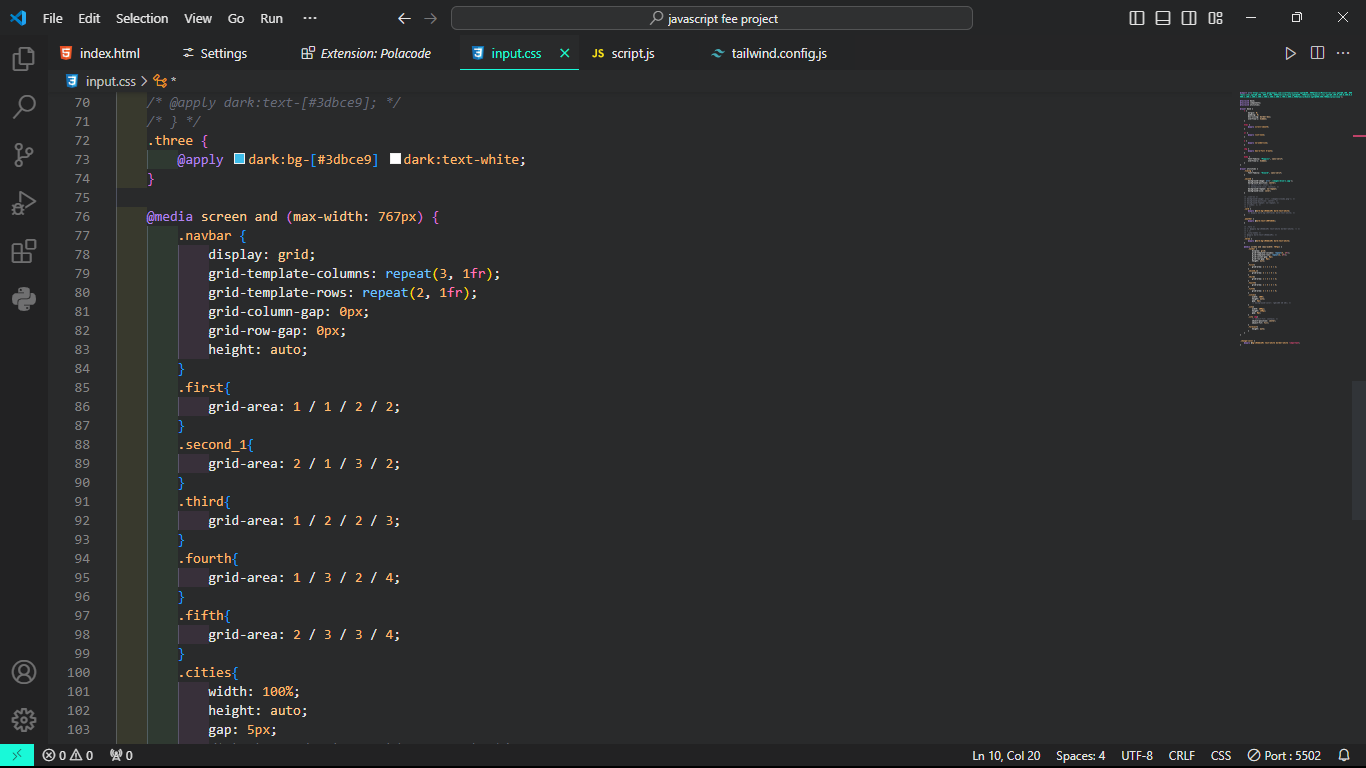


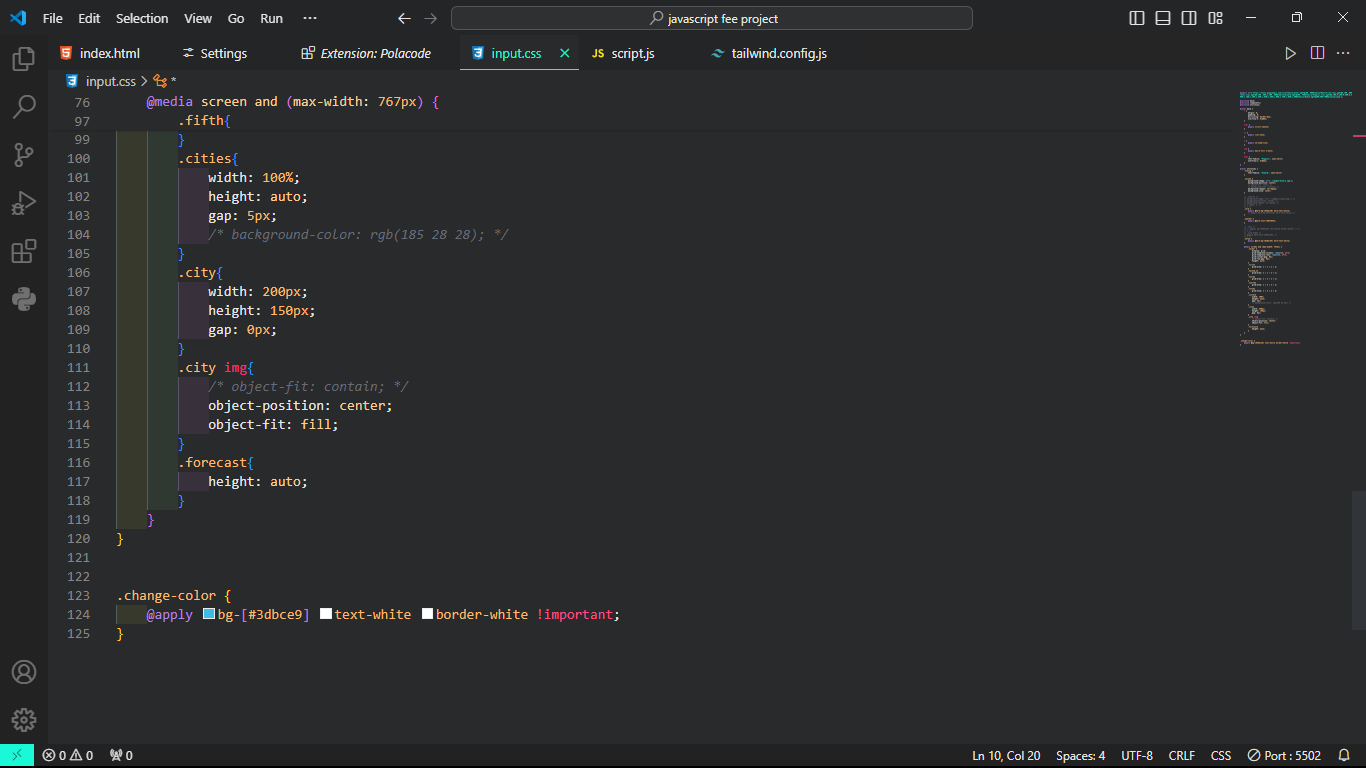


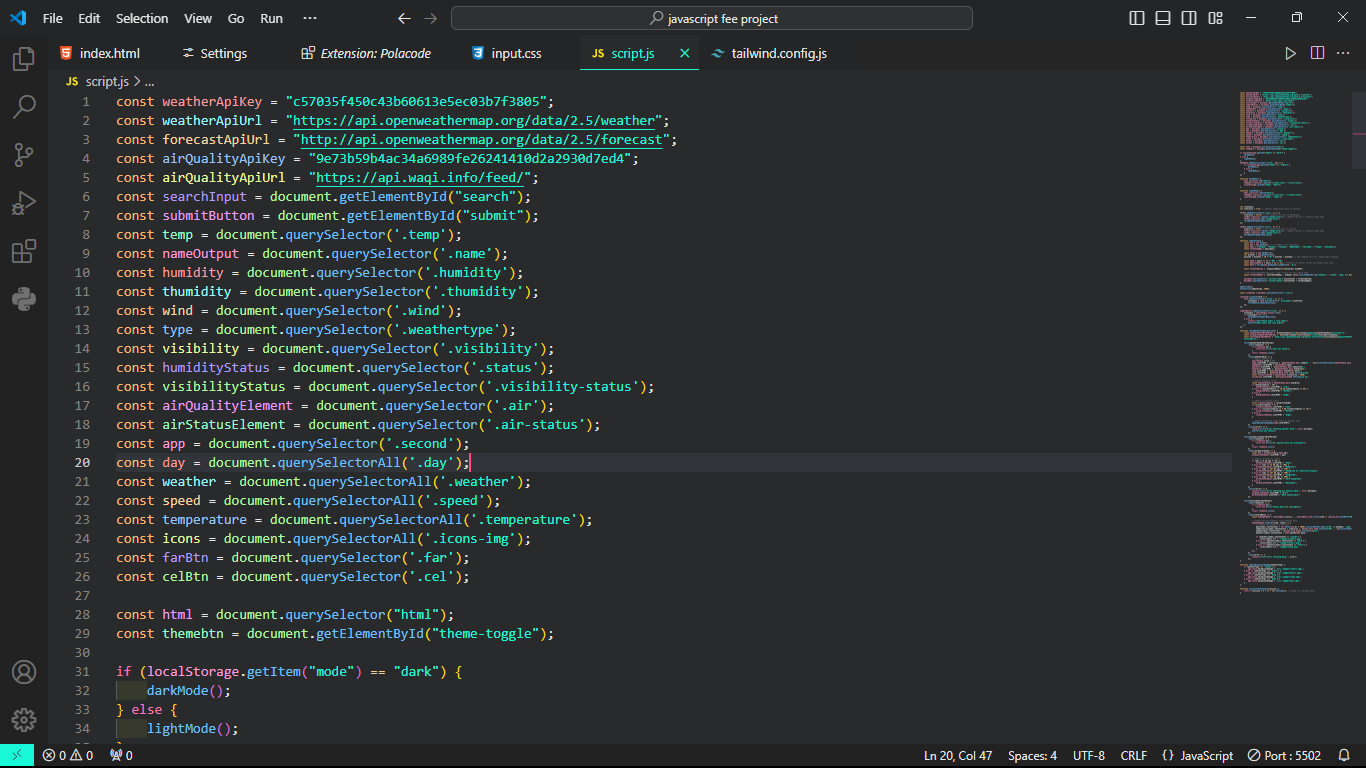


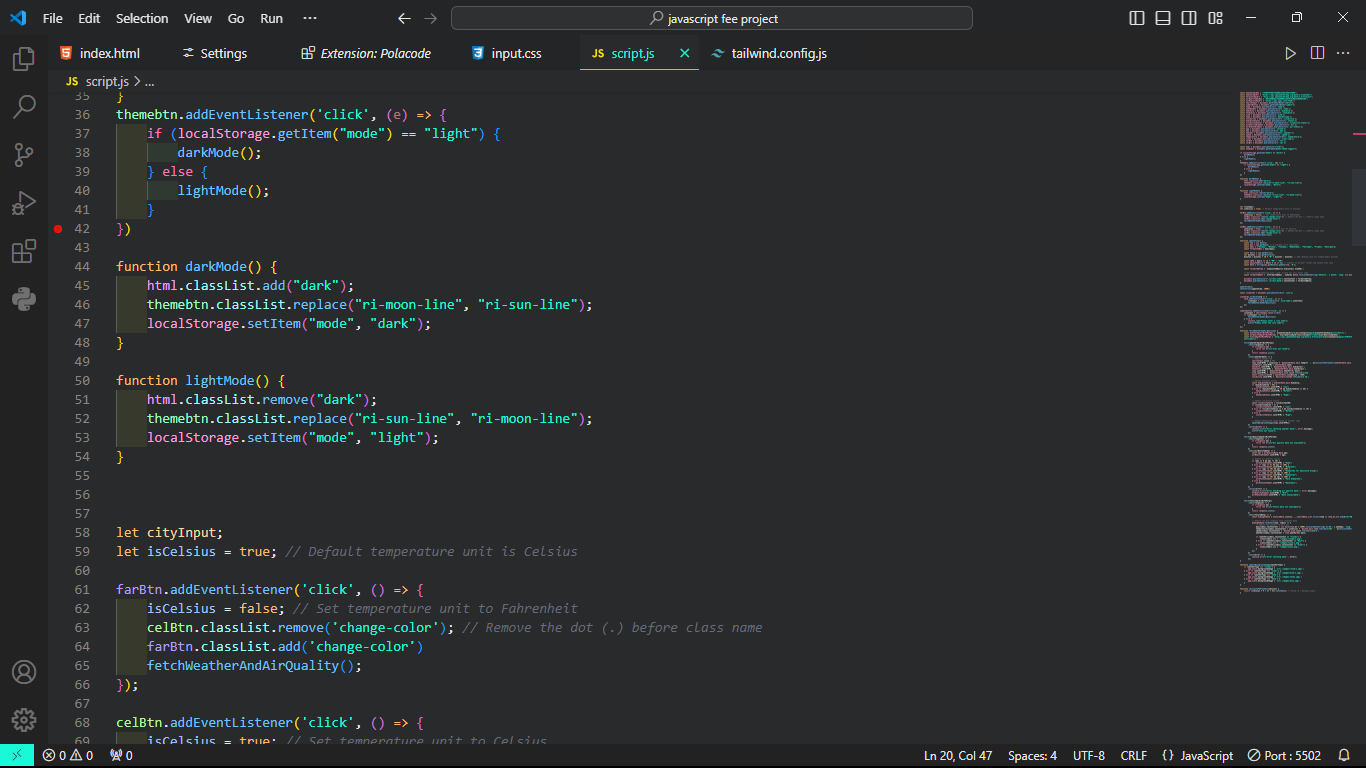


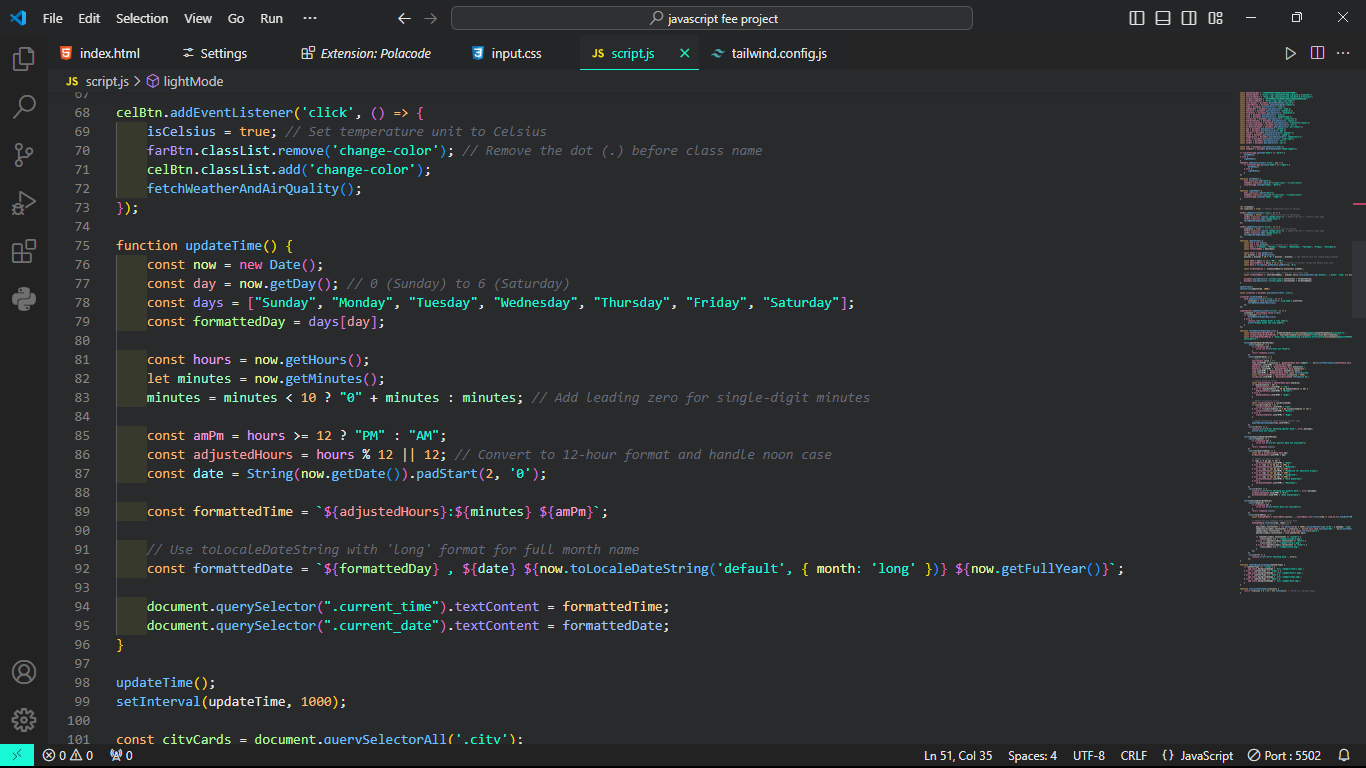


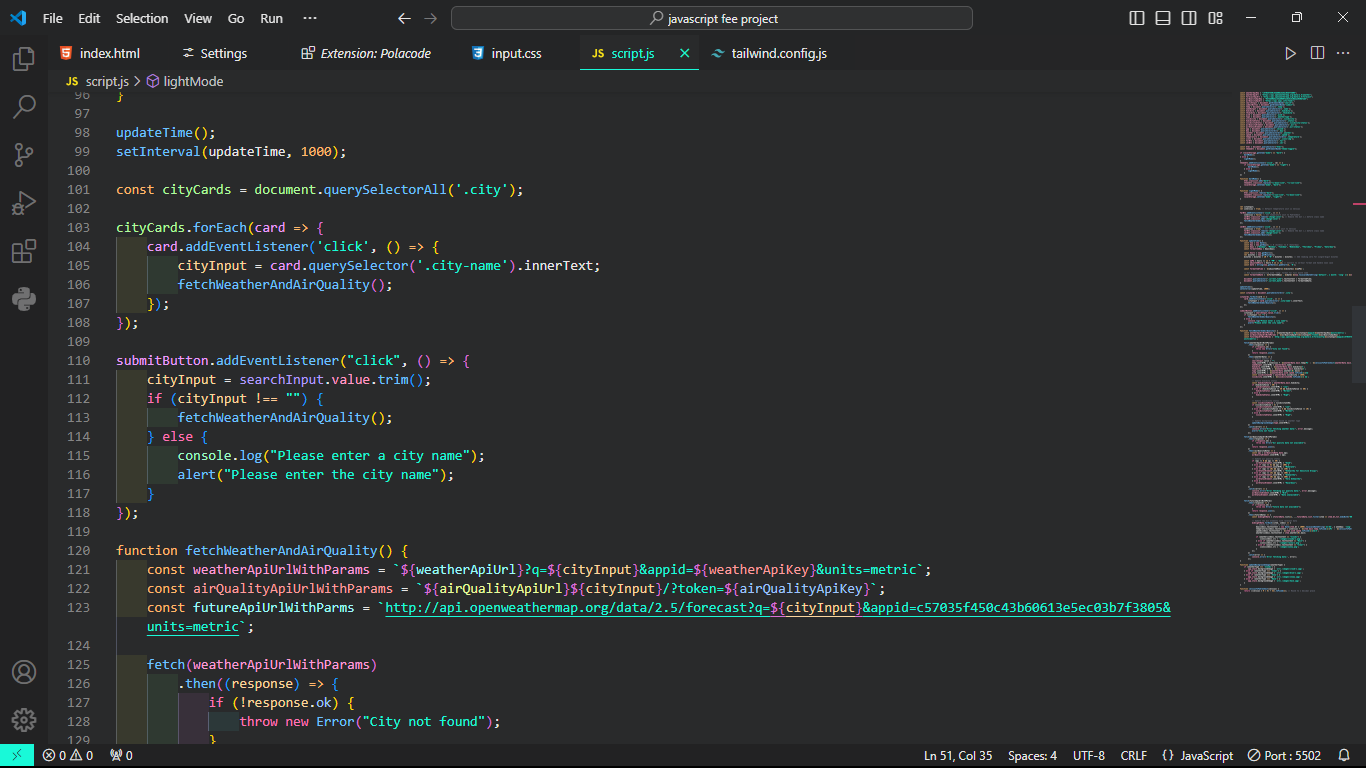


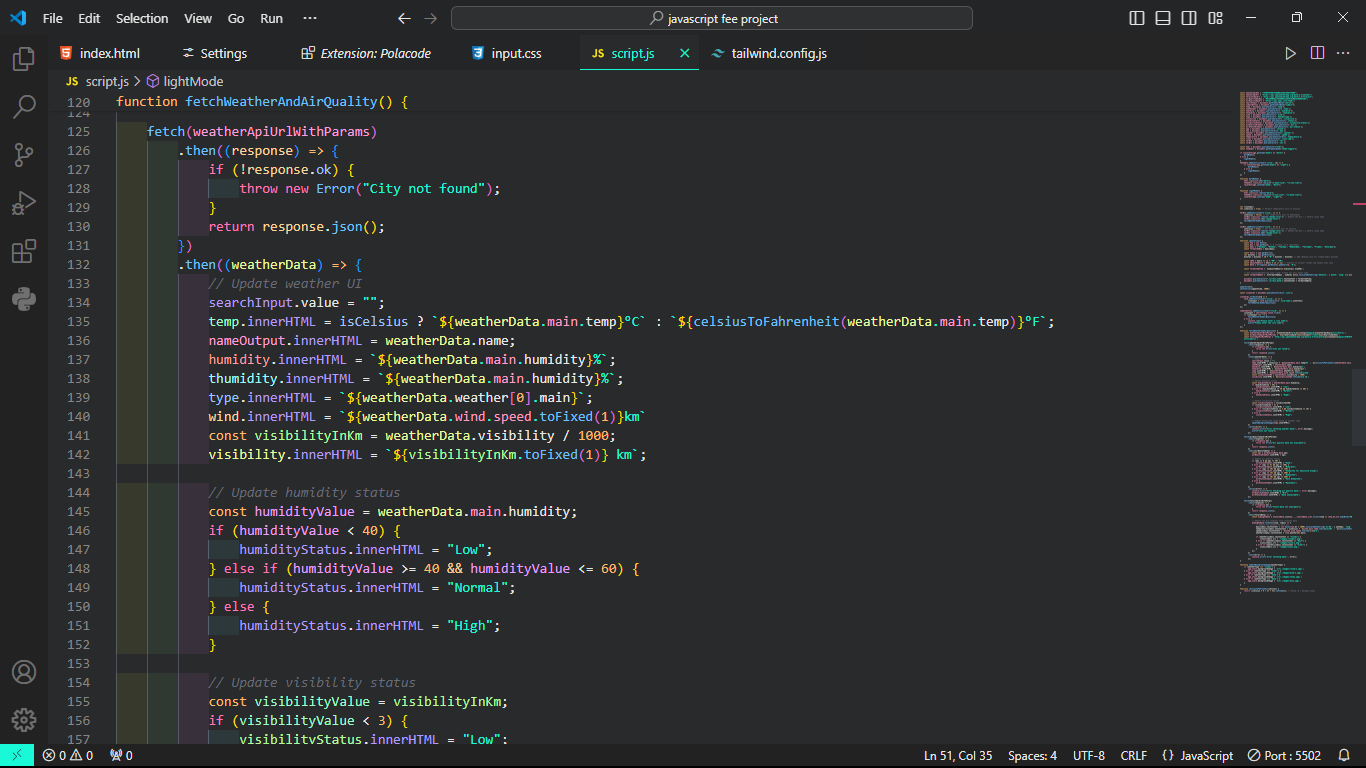


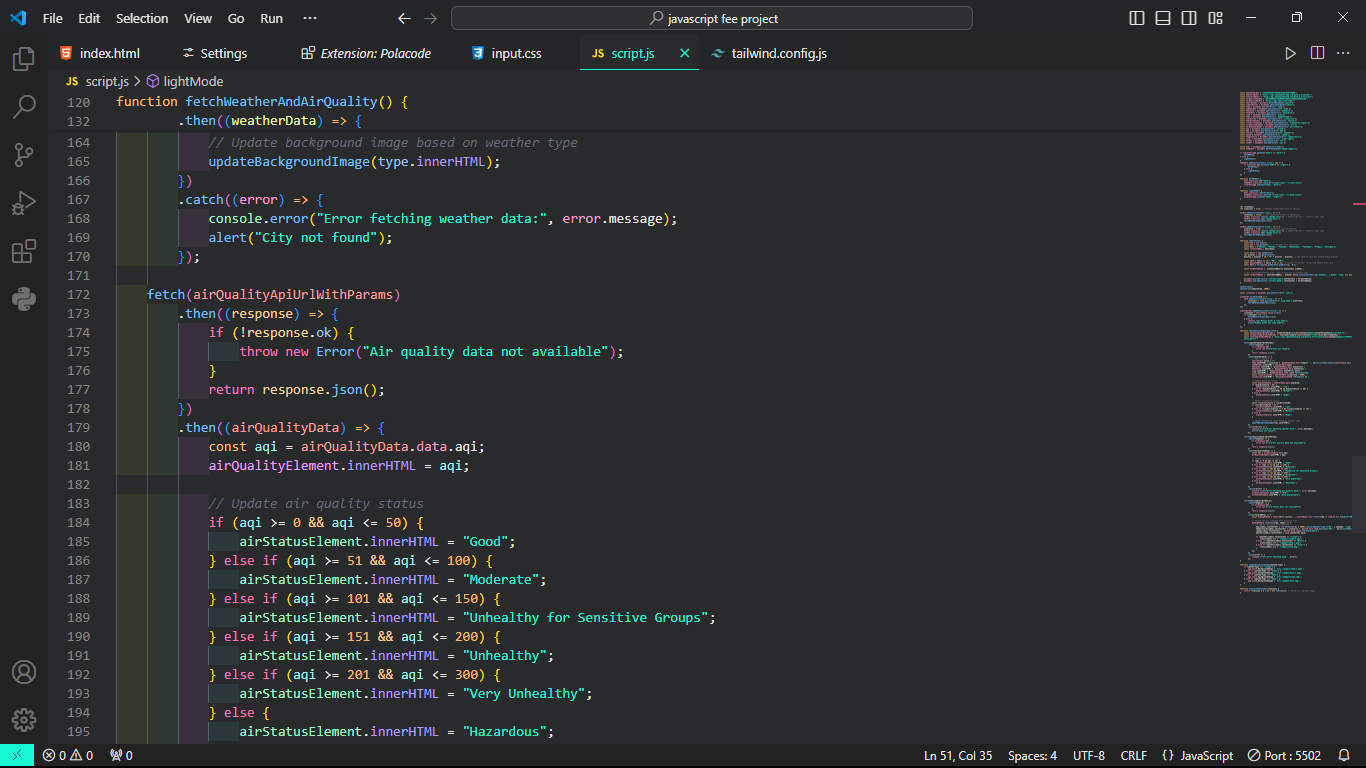


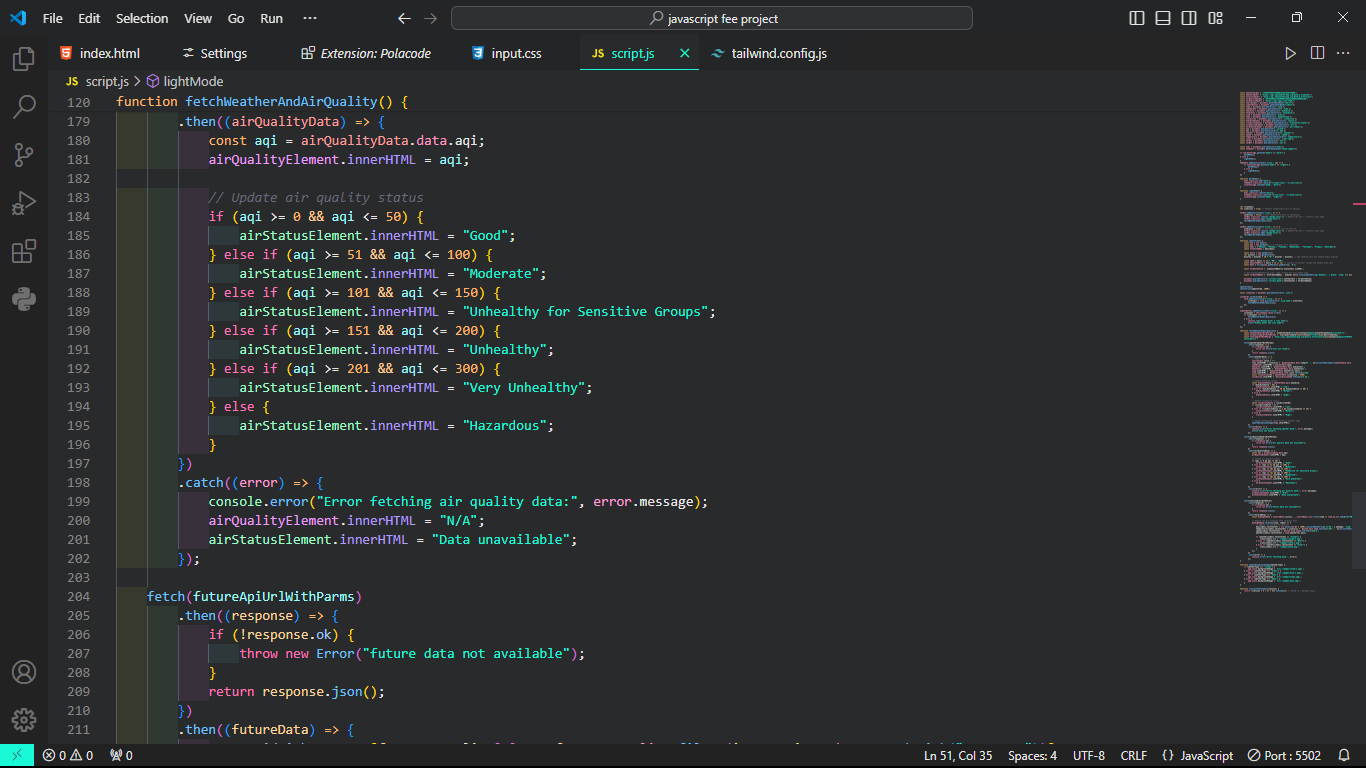


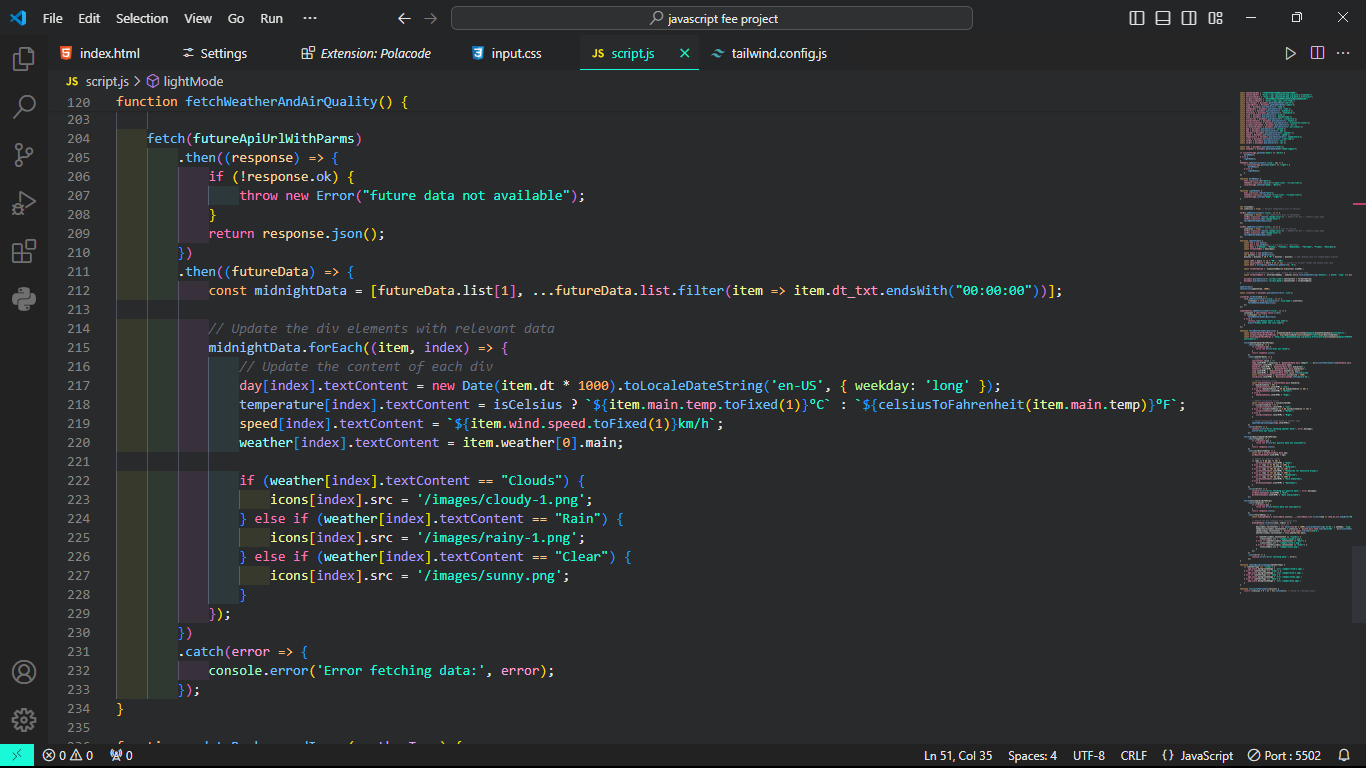


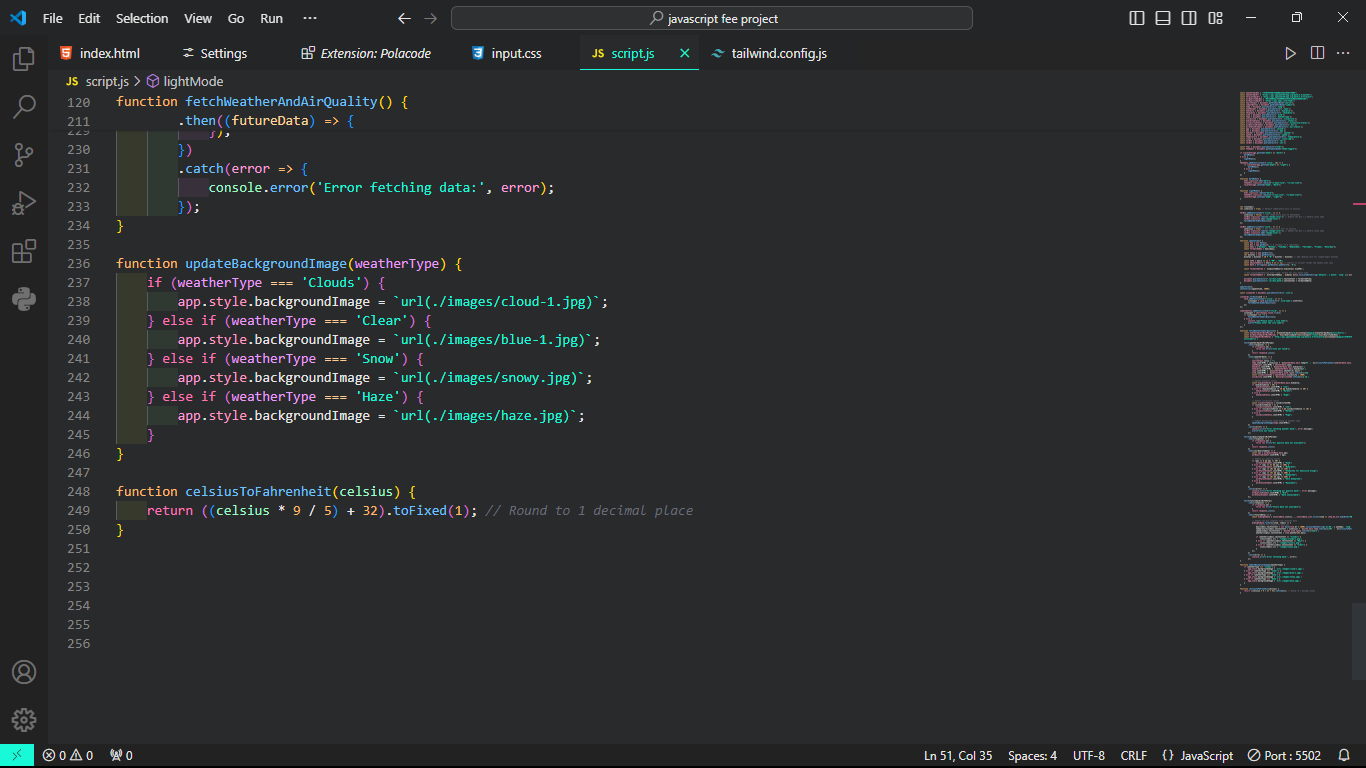




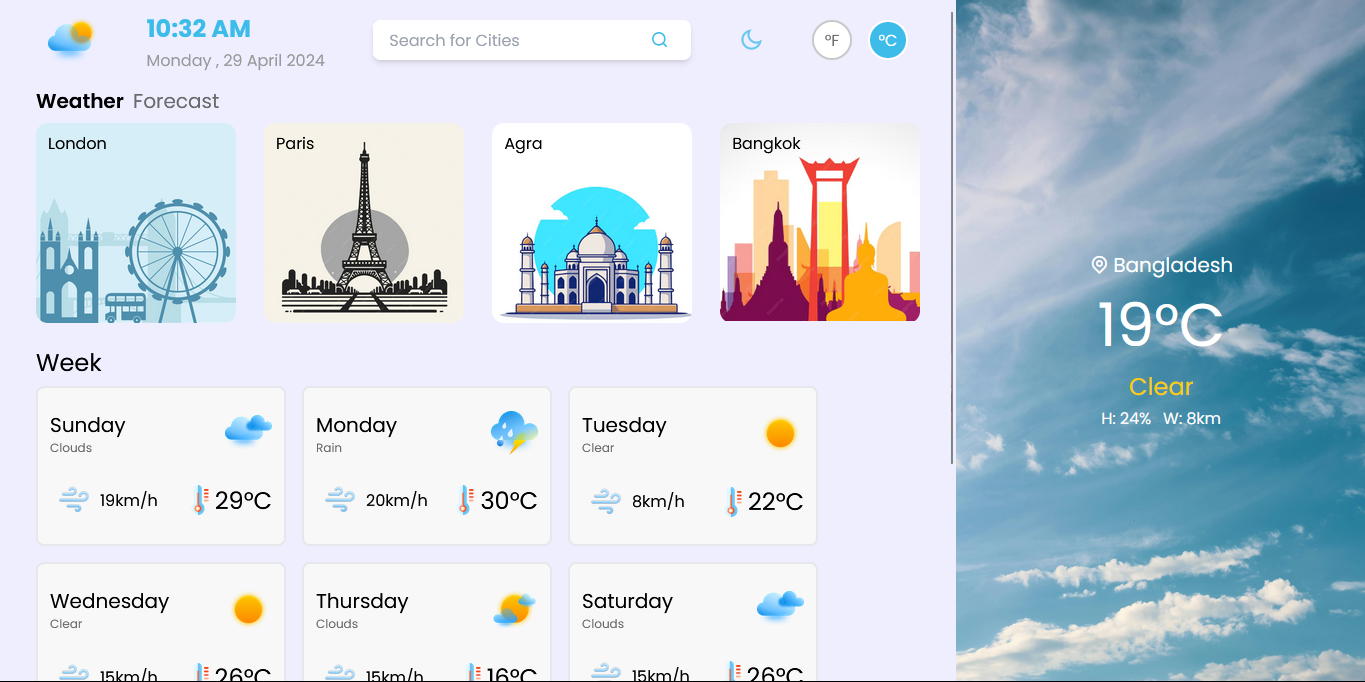


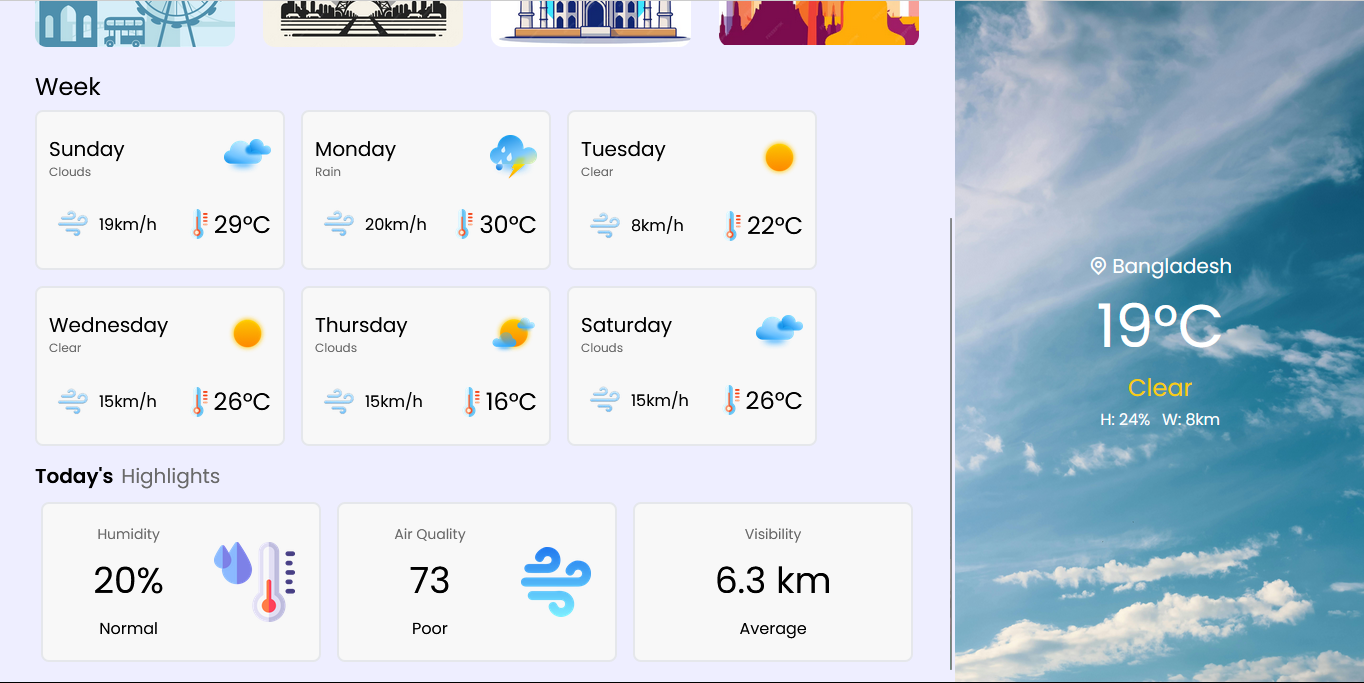






**Output:**

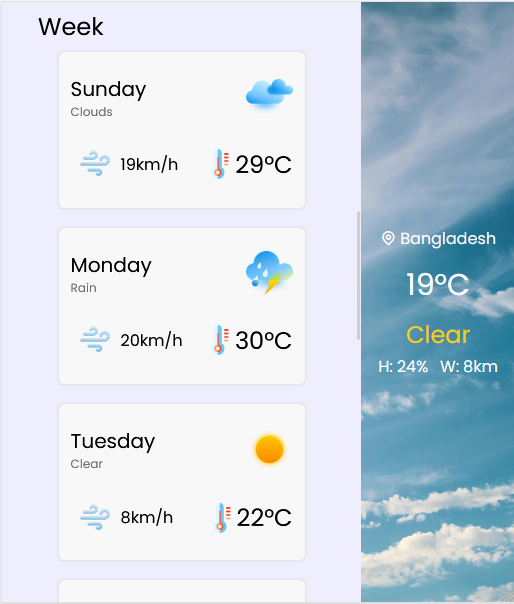
****

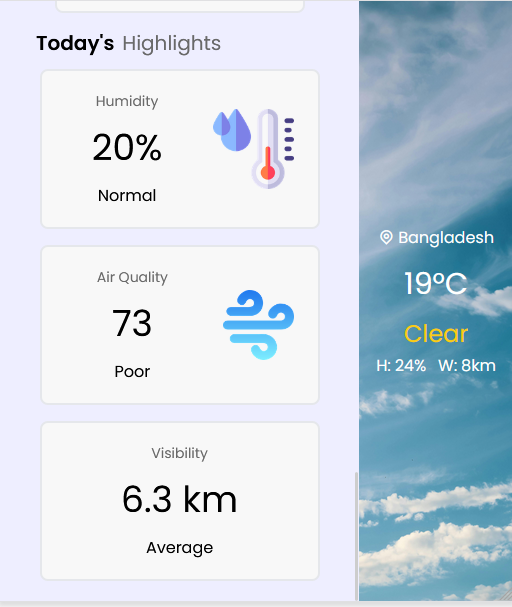
****

|  |
| --- |
|  |

**Mobile View:**

****

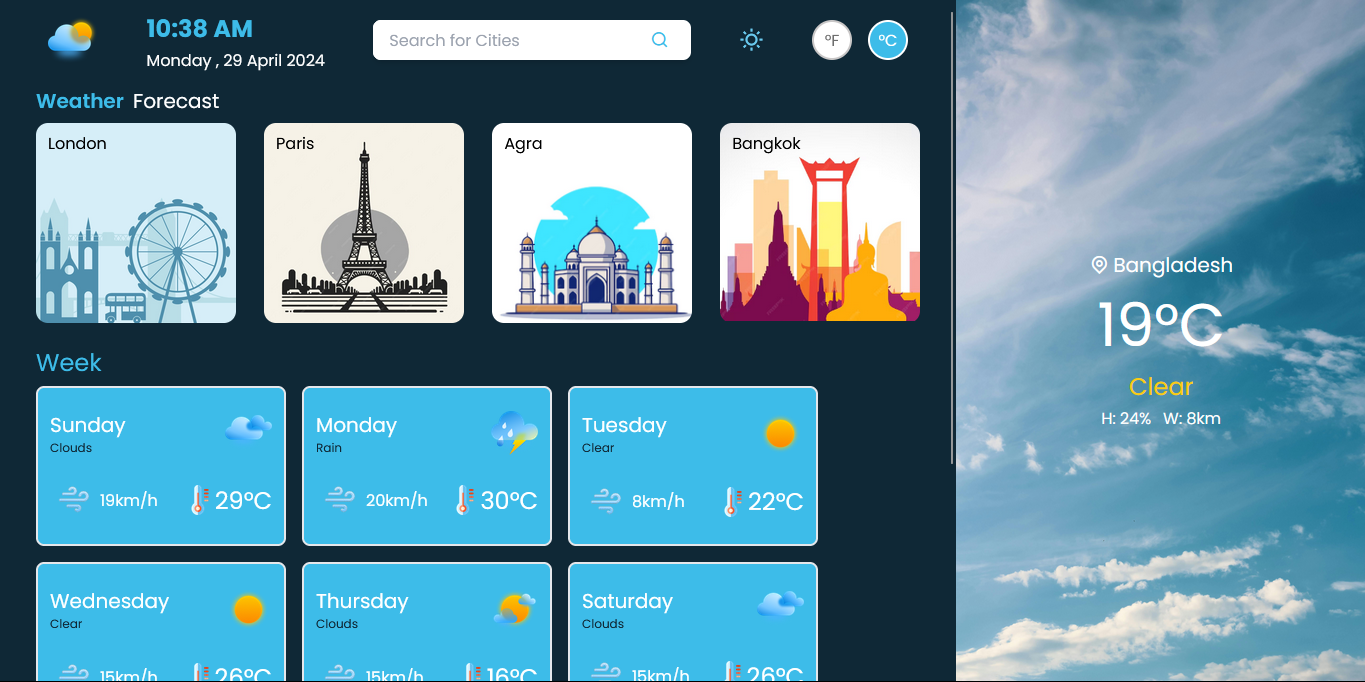
****

****

|  |
| --- |
|  |

|  |
| --- |
|  |

**Dark mode:**

****

|  |
| --- |
|  |



|  |
| --- |
|  |

# References

1. OpenWeatherMap API - [https://openweathermap.org/](%20https:/openweathermap.org/)
2. Tailwind Css.com - <https://tailwindcss.com/>
3. Dribble – <https://dribbble.com/following>
4. Remix-icon - <https://remixicon.com/>