Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2024), B.Sc. in CSE (Day)

Weather Forecasting Website

Course Title: Web Programming Lab
Course Code: CSE 302
Section: D17

Students Details

Name	ID
Shahariar Hossain Rakib	221902071

Submission Date: 10/06/2024
Course Teacher's Name: Mr. Montaser Abdul Quader

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The project is a dynamic Weather Forecasting Web Application built using HTML, CSS, and JavaScript. It allows users to input their location and retrieves real-time weather data using the OpenWeatherMap API. The application displays temperature, weather conditions, humidity, and wind speed, with responsive and interactive UI elements. It also handles errors by notifying the user if the location is not found.

1.2 Motivation

1.Real-time Information: Provide users with accurate and up-to-date weather data to help them plan their daily activities effectively.

2.User Convenience: Offer an easy-to-use interface where users can quickly check the weather conditions of any location with a simple search.

3.Skill Enhancement: Enhance proficiency in web development technologies like HTML, CSS, and JavaScript, and gain experience in working with APIs.

4.Problem-Solving: Address common user issues related to weather uncertainty and make information readily accessible to a broad audience.

5.Aesthetic Appeal: Design a visually appealing and responsive user interface that enhances user experience and engagement.

6.Educational Value: Serve as a practical project to learn about asynchronous programming and API integration.

7.Global Accessibility: Enable people from different regions to access weather data in a centralized and consistent manner.

8.Future Integration: Lay the foundation for integrating additional features such as weather forecasts, severe weather alerts, and personalized weather reports.

1.3 Problem Definition

1.3.1 Problem Statement

1.Access to Real-time Data: Users lack a convenient tool to access accurate, real-time weather information for their specific location.

2.User-Friendly Interface: Many existing weather apps are complex or cluttered, making it difficult for users to quickly find the information they need.

3.Error Handling: Users often encounter issues when a location is not found, and there is a need for clear and user-friendly error notifications.

4. Weather Detail Presentation: Users require a straightforward presentation of key weather details such as temperature, humidity, and wind speed to make informed decisions.

1.3.2 Complex Engineering Problem

The following table is completed according to my above discussion in detail

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	The project requires a solid understanding of web development fundamentals, including HTML, CSS, JavaScript, and API integration.
P2: Range of conflicting requirements	The project must balance user interface simplicity with comprehensive weather data presentation, ensuring both ease of use and detailed information availability.
P3: Depth of analysis required	The project necessitates thorough analysis of API data handling, user input validation, and effective error management to ensure accurate and reliable weather information delivery.
P4: Familiarity of issues	The project extensively applies web development best practices, including semantic HTML, responsive CSS design, and efficient JavaScript for dynamic content and API interaction.
P5: Extent of applicable codes	
P6: Extent of stakeholder involvement and conflicting requirements	The project involves balancing the needs and feedback of various stakeholders, such as end-users and developers, to address conflicting requirements like user-friendly design versus detailed data presentation.
P7: Interdependence	—-

1.4 Design Goals/Objectives

1. User-Centric Design: Prioritize an intuitive and visually appealing interface to enhance user experience and engagement.

2.Data Accessibility: Ensure easy access to accurate weather information with a minimal number of user interactions.

3.Error Handling: Implement clear and informative error messages to guide users in case of invalid inputs or API failures.

4. Responsiveness: Create a responsive design that adapts seamlessly across various devices and screen sizes for optimal usability.

5.Scalability: Design the application with modular components and efficient code to facilitate future updates and feature enhancements.

1.5 Application

The Weather Forecasting Web Application provides users with real-time weather updates by entering their location. It offers a user-friendly interface, displaying key weather details such as temperature, humidity, and wind speed. The application handles errors gracefully, notifying users if a location is not found. Built with HTML, CSS, and JavaScript, it ensures responsiveness across different devices for seamless accessibility.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The Weather Forecasting Web Application is designed to provide users with instant access to up-to-date weather information. By simply inputting their location, users can obtain detailed weather forecasts. This application aims to offer a seamless user experience through a clean and intuitive interface. Developed using HTML, CSS, and JavaScript, it serves as a reliable tool for users to plan their activities based on current weather conditions. [1].

2.2 Project Details

The project comprises HTML, CSS, and JavaScript code for a Weather Forecasting Web Application. It utilizes the OpenWeatherMap API to fetch real-time weather data based on user input. The application presents weather information such as temperature, humidity, and wind speed in a visually appealing manner. Error handling is implemented to notify users if their location is not found, ensuring a smooth user experience.

2.3 Implementation

The implementation involves integrating HTML, CSS, and JavaScript to create a responsive user interface and fetch weather data from the OpenWeatherMap API based on user input. Error handling is implemented to display informative messages in case of invalid locations or API failures, ensuring robust functionality.

2.3.1 Tools And Library

The project utilizes basic web development tools such as text editors (e.g., Visual Studio Code, Sublime Text) for coding and testing. Additionally, it incorporates the Font Awe-

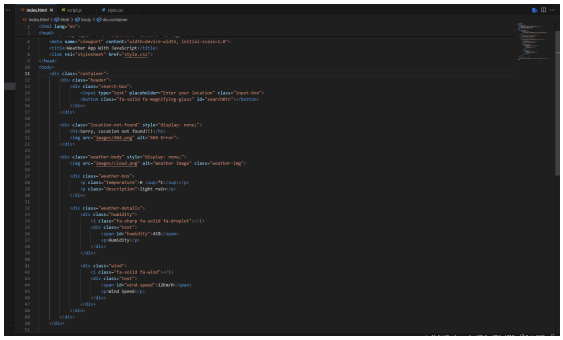
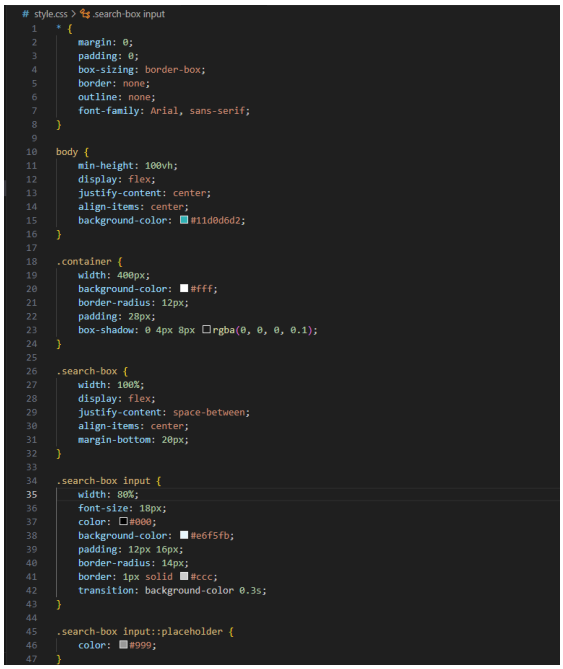


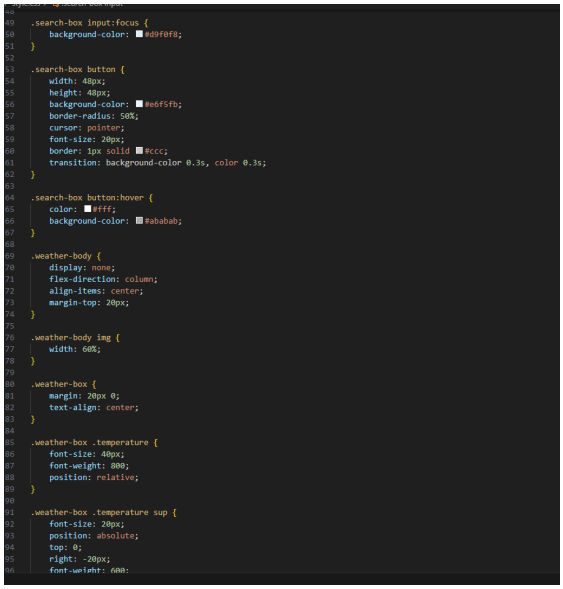
Figure 2.1: Html Code

Figure 2.2: Javascript code



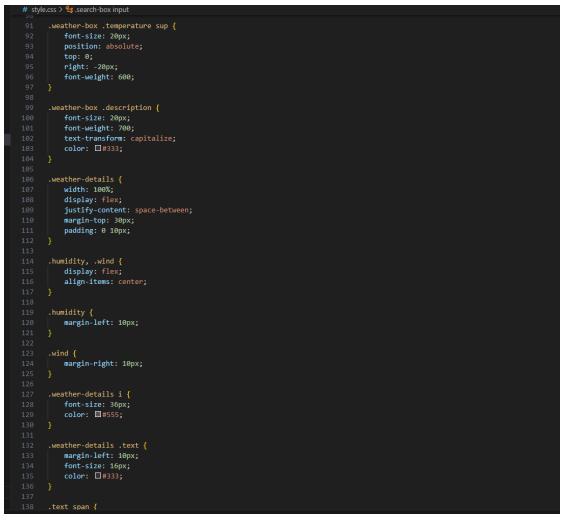
```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    border: none;
    outline: none;
    font-family: Arial, sans-serif;
}

body {
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #11d0d6d2;
}

.container {
    width: 400px;
    background-color: #fff;
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.search-box {
    width: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.search-box input {
    width: 80%;
    font-size: 18px;
    color: #000;
    background-color: #e6f5fb;
    padding: 12px 16px;
    border-radius: 14px;
    border: 1px solid #ccc;
    transition: background-color 0.3s;
}

.search-box input::placeholder {
    color: #999;
}
```

Figure 2.3: CSS Code



```
.search-box input:focus {
    background-color: #d9f0f8;
}

.search-box button {
    width: 48px;
    height: 48px;
    background-color: #e6f5fb;
    border-radius: 50%;
    cursor: pointer;
    font-size: 20px;
    border: 1px solid #ccc;
    transition: background-color 0.3s, color 0.3s;
}

.search-box button:hover {
    color: #fff;
    background-color: #ababab;
}

.weather-body {
    display: none;
    flex-direction: column;
    align-items: center;
    margin-top: 20px;
}

.weather-body img {
    width: 60%;
}

.weather-box {
    margin: 20px 0;
    text-align: center;
}

.weather-box .temperature {
    font-size: 40px;
    font-weight: 800;
    position: relative;
}

.weather-box .temperature sup {
    font-size: 20px;
    position: absolute;
    top: 0;
    right: -20px;
    font-weight: 600;
```

Figure 2.4: CSS Code



```
style.css > .search-box input
.weather-box .temperature sup {
    font-size: 20px;
    position: absolute;
    top: 0;
    right: -20px;
    font-weight: 600;
}

.weather-box .description {
    font-size: 20px;
    font-weight: 700;
    text-transform: capitalize;
    color: #333;
}

.weather-details {
    width: 100%;
    display: flex;
    justify-content: space-between;
    margin-top: 30px;
    padding: 0 10px;
}

.humidity, .wind {
    display: flex;
    align-items: center;
}

.humidity {
    margin-left: 10px;
}

.wind {
    margin-right: 10px;
}

.weather-details i {
    font-size: 36px;
    color: #555;
}

.weather-details .text {
    margin-left: 10px;
    font-size: 16px;
    color: #333;
}

.text span {
```

Figure 2.5: CSS Code

some library for icon integration, enhancing the visual presentation of the application. JavaScript fetch API is employed for making asynchronous requests to the OpenWeatherMap API to retrieve weather data.

2.4 Algorithms

2.4.1 User Input

1. User inputs their location into the search box

2.4.2 Decryption Algorithm

1. JavaScript code captures the user input.

2. It constructs a URL with the user's location and the API key for the OpenWeatherMap API.

3. A fetch request is made to the constructed URL to retrieve weather data.

2.4.3 Data Processing

1 Upon receiving the weather data, the JavaScript code parses the JSON response.

2. It checks for errors in the response, such as a "404" code indicating the location was not found.

3. If there are no errors, it extracts relevant weather information such as temperature, humidity, and wind speed.

2.4.4 Display

1. The extracted weather information is dynamically inserted into the HTML elements of the web page.

2. Icons corresponding to weather conditions are displayed based on the received data.

3. If there are errors, appropriate error messages are displayed to the user.

2.4.5 User Interaction

1. Users can repeat the process by entering new locations or refreshing the page to fetch updated weather data.

2.4.6 Error Handling

1. In case of errors during data retrieval or processing, the application displays user-friendly error messages to guide the user

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

The application operates within a web browser environment, where users interact with the interface to input their location and retrieve weather data. Simulation involves emulating user behavior by entering locations, triggering API requests, and observing the display of weather information and error messages.

3.2 Results

Sample Ouputs of the Program

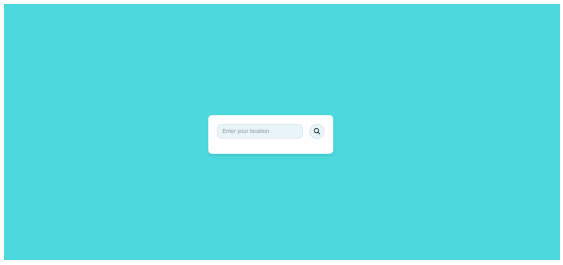


Figure 3.1: User Interface of Website.

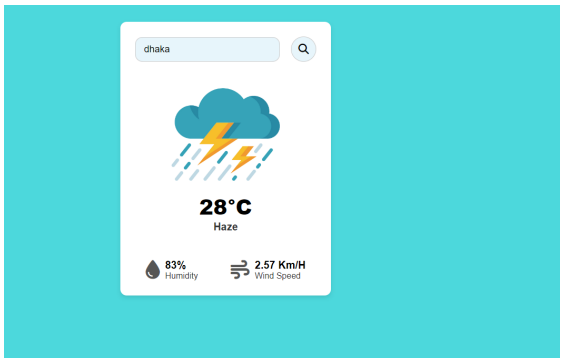


Figure 3.2: Temperature Of Dhaka

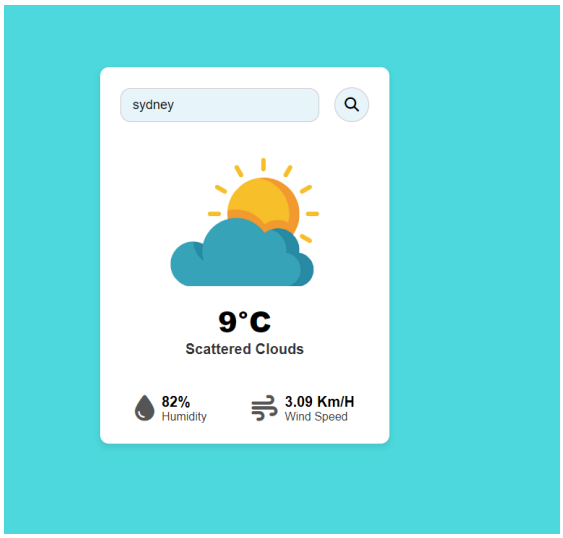


Figure 3.3: Temperature Of Sydney

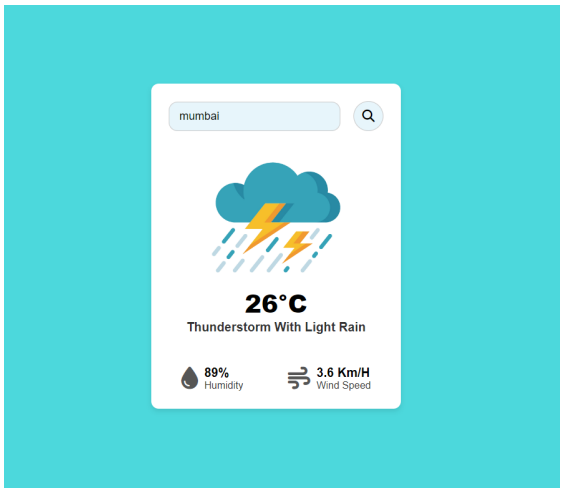


Figure 3.4: Temperature Of Mumbai

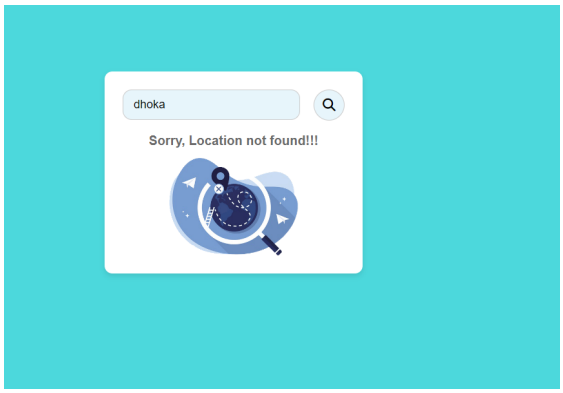


Figure 3.5: Invalid City Name

Chapter 4

Conclusion

4.1 Discussion

The Weather Forecasting Web Application offers a convenient solution for users seeking instant weather updates. Its user-friendly interface and seamless integration with the OpenWeatherMap API ensure accessibility and accuracy of weather data. The project's success hinges on effective error handling, ensuring a smooth user experience even in cases of invalid inputs or API failures. By employing responsive design principles, the application caters to a diverse range of devices, enhancing its usability and accessibility. Future enhancements could include additional features such as extended forecasts, location-based alerts, and customization options to further improve user engagement and satisfaction.

4.2 Limitations

Despite its functionality, the Weather Forecasting Web Application has several limitations. These include dependency on the availability and reliability of the OpenWeatherMap API, which may affect data retrieval. The application's accuracy is also contingent on the precision of the location input provided by users. Limited error handling capabilities may result in suboptimal user experience in scenarios where API requests fail or locations are not found. Additionally, the application's current version lacks advanced features such as extended forecasts, severe weather alerts, and localization options, limiting its utility for certain user needs. Continuous monitoring and updates are necessary to address these limitations and enhance the application's effectiveness and user satisfaction.

4.3 Scope of Future Work

Future iterations of the Weather Forecasting Web Application could focus on expanding its feature set to include extended weather forecasts, hourly updates, and customizable alert notifications for severe weather conditions. Localization options could be imple-

mented to provide weather information in multiple languages and units. Integration with additional weather APIs could enhance data accuracy and reliability. Improvements in error handling mechanisms and user feedback mechanisms could further enhance the application's usability and reliability. Finally, optimizing the application for performance and scalability to accommodate a growing user base and increased data demands would be beneficial for its long-term viability.

References

[1] Open Weather. https://openweathermap.org/api. Accessed Date: 2024-05-30.