RKDF UNIVERSITY

Ranchi , Jharkhand

INTERNSHIP REPORT

Submitted by

Kashish Dangil (001CSL23GT012)

in partial fulfillment for the award of the degree

of

Bachelor of Engineering

in

Computer Science and Engineering

RKDF University

Ranchi , Jharkhand

An Autonomous Instituution

MAY 2025

BONAFIDE CERTIFICATE

Certified that this internship report

“ Weather System Website “

is the bonafide work of

Kashish Dangil(001CSL23GT012)

who carried out the project work under my supervision

SIGNATURE OF HOD	SIGNATURE OF SUPERVISIOR
Shubhangni Dey	**Abhishek Kumar Singh**
HEAD OF DEPARTMENT	SUPERVISOR
Computer science and Engineering	Computer science and Engineering
RKDF University , Ranchi , Jharkhand	**RKDF University , Ranchi , Jharkhand**

Submitted for the Autonomous End Semester Examination Internship Project

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Weather System Website is a client-side web application designed to deliver real-time weather updates for any city across the globe. Developed using core front-end technologies—HTML, CSS, and JavaScript—this project demonstrates the integration of web development practices with external APIs to create a dynamic and interactive user experience. The primary goal of the project is to allow users to input the name of a city and instantly retrieve up-to-date weather information including temperature, weather conditions, humidity, wind speed, and corresponding weather icons.

From a design perspective, the application features a clean and responsive user interface. CSS is used extensively to enhance the visual appeal and usability of the site across different devices and screen sizes. The layout ensures that content is easy to read and access, while media queries allow for smooth adaptation to smartphones, tablets, and desktop screens.

Functionality such as error handling (for invalid city names or network issues) is also implemented, making the application more robust and user-friendly. While the system is limited to current weather data and requires an internet connection for API access, it lays the groundwork for more advanced features such as GPS-based location detection, forecast data, and multilingual support.

In conclusion, this project serves as a practical example of modern front-end web development and API integration. It highlights the power of JavaScript in handling asynchronous operations and demonstrates how web applications can be enriched with real-time data services to solve real-world problems. The Weather System Website not only fulfills its primary function effectively but also provides a scalable framework for further development and enhancements.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep gratitude to all those who supported me throughout the development of my project, *Weather System Website*. This project has been a significant part of my academic journey, and its successful completion would not have been possible without the valuable support, guidance, and encouragement of several individuals.

First and foremost, I would like to express my sincere thanks to my project guide, **Rohit Kumar**, , for his invaluable guidance, constant support, and encouragement throughout the duration of this project. Their expert advice, technical knowledge, and timely feedback were instrumental in shaping this project and overcoming the various challenges I encountered along the way.

I am also thankful to the faculty members of the Department of **Computer Science and Engineering** at **RKDF University , Ranchi** for providing a strong foundation in web development and programming principles, which greatly helped in the practical implementation of this project.

I would like to extend my gratitude to my classmates and friends who assisted me by providing constructive feedback and testing the website to ensure better functionality and user experience. Their suggestions contributed meaningfully to refining the features and user interface of the application.

I also acknowledge the role of the OpenWeatherMap API, which served as the primary data source for this project. The accessibility and reliability of their API made it possible to integrate real-time weather data into the application effectively.

Finally, I would like to thank my family for their unwavering support, motivation, and patience. Their continuous encouragement gave me the strength to work with dedication and complete this project successfully.

In conclusion, I am deeply grateful to all the individuals mentioned above for their contribution to the successful completion of the *Weather System Website*. Their involvement and support have been truly meaningful and appreciated.

Kashish Dangil

(001CSL23GT012)

B.TECH (C.S.E) [LATERAL]

Semester-VI

TABLE OF CONTENTS

1. INTRODUCTION

In the modern digital age, access to real-time information is not just a luxury but a necessity. Weather information, in particular, plays a crucial role in our daily lives—impacting everything from personal travel plans and event scheduling to agriculture, aviation, and emergency response planning. In this context, web-based weather systems have become increasingly popular and essential tools for individuals and businesses alike.

The *Weather System Website* project is designed as a simple yet functional web application that provides users with up-to-date weather information for any city around the world. This system is built using core front-end technologies: HTML, CSS, and JavaScript, making it lightweight, responsive, and easy to deploy across various platforms and devices.

The main functionality of the website revolves around real-time data fetching from an external weather API, such as OpenWeatherMap, which provides comprehensive and accurate weather data in a developer-friendly format. Users simply enter the name of a city in a search field, and the website fetches and displays the current temperature, weather conditions, humidity, wind speed, and an appropriate weather icon that visually represents the climate.

From a technical perspective, the project emphasizes several foundational concepts in web development:

- HTML is used to structure the content of the web pages.
- CSS is employed to style the user interface, ensuring that the website is visually appealing and responsive.
- JavaScript is used to implement logic, handle API requests, process responses, and update the DOM dynamically based on the data received.

This project not only demonstrates the practical use of front-end development skills but also introduces the concept of asynchronous programming through the use of fetch() and Promises in JavaScript. Moreover, it helps build a deeper understanding of how modern web applications communicate with third-party services to enrich user experiences.

The *Weather System Website* is particularly useful for learning how to integrate real-world data into a web application and how to present that data in a clear and user-friendly way. It is a beginner-to-intermediate level project that can serve as a strong foundation for further advancements, such as integrating forecast data, geolocation-based weather detection, or voice search functionality.

2. OBJECTIVE

The primary goal of the *Weather System Website* project is to develop a dynamic, responsive, and user-friendly web application that provides real-time weather information for cities around the world. This project is not only focused on delivering functional weather data but also aims to enhance the developer's understanding of front-end web technologies and real-time API integration. The following are the detailed objectives of the project:

(i). To Design and Develop a Web-Based Weather Information System

The project aims to create a fully functional web application using HTML, CSS, and JavaScript. It involves designing an intuitive user interface and implementing functionality that allows users to search for and retrieve weather data seamlessly.

(ii). To Integrate Real-Time Weather Data Using a Public API

A major objective of the project is to connect the website with an external weather service provider, such as OpenWeatherMap, to fetch live data based on the user's input. This teaches how to work with RESTful APIs, handle JSON data, and perform asynchronous operations using JavaScript.

(iii). To Implement a Clean and Responsive User Interface

The project aims to provide a responsive layout that works well on all device types, including desktops, tablets, and mobile phones. The use of CSS media queries and flexible design components ensures the site is visually consistent and accessible across screen sizes.

(iv). To Enable User Interaction and Dynamic Content Rendering

The website allows users to enter a city name to receive weather updates. The objective is to capture user input, validate it, and dynamically update the web page with weather data without refreshing the entire page—demonstrating JavaScript's power for creating interactive content.

(v). To Display Key Weather Parameters Effectively

An important goal is to present essential weather parameters clearly and concisely. These include:

- Current temperature
- Weather condition (e.g., clear, cloudy, rainy)
- Humidity level
- Wind speed
- Location name
- Weather icon representing the current condition

(v). To Incorporate Basic Error Handling and User Feedback

To improve user experience, the website should handle common errors gracefully—such as invalid city names, API errors, or network failures—and provide informative feedback to guide users in correcting their input.

3. TOOLS AND TECHNOLOGIES

The development of the *Weather System Website* involved a variety of tools and technologies that together enabled the creation of a responsive, interactive, and data-driven web application. Each component played a specific role in the design, development, and deployment of the project. Below is a detailed overview:

(i). HTML5 (HyperText Markup Language)

- Purpose: Structure and layout of web pages.
- Usage: Defined the content and elements of the website such as headings, input fields, buttons, and result display sections.
- Benefits: Provides a semantic structure to the page, improving both accessibility and SEO.

(ii). CSS3 (Cascading Style Sheets)

- Purpose: Styling and visual design.
- Usage: Used for layout control, color schemes, responsive design with media queries, and interactive elements such as hover effects.
- Benefits: Enhances user experience by making the site visually appealing and adaptable to various screen sizes.

(iii). JavaScript (ES6+)

- Purpose: Functionality and interactivity.
- Usage: Implemented client-side logic including event handling, API calls, DOM manipulation, error handling, and updating the interface dynamically.
- Benefits: Enables asynchronous communication with the weather API and interactive user experiences without reloading the page.

(Iv). OpenWeatherMap API

- Purpose: Source of real-time weather data.
- Usage: Fetches current weather details based on user input (city name).
- Features: Returns data in JSON format including temperature, humidity, wind speed, and weather conditions.
- Benefits: Provides reliable and globally accessible weather data through a free-tier plan.

(v). Visual Studio Code (VS Code)

- Purpose: Code editor and development environment.
- Usage: Used for writing, editing, and debugging HTML, CSS, and JavaScript files.
- Benefits: Offers built-in extensions, syntax highlighting, IntelliSense, and Git integration for efficient development.

(Vi). Git & GitHub (Optional)

- Purpose: Version control and project hosting.
- Usage: Git is used to track changes and manage the project history. GitHub is used to host and share the project repository online.
- Benefits: Facilitates collaboration, backup, and easy access to project code from anywhere.

(vii). Web Browser (Chrome / Firefox / Edge)

- Purpose: Testing and running the web application.
- Usage: Used to test responsiveness, inspect code, debug JavaScript, and monitor API requests through developer tools.
- Benefits: Helps in performance testing, cross-browser compatibility checks, and real-time debugging.

(viii). Internet Connection

- Purpose: Required for fetching real-time data from the weather API.
- Usage: Enables communication between the website and the external API.
- Limitations: The application depends on a working internet connection to retrieve weather data.

Summary Table

Tool/Technology	Purpose
HTML5	Page structure and content
CSS3	Styling and layout
JavaScript (ES6+)	Logic, interactivity, and API integration
Visual Studio Code	Code development and editing
Web Browser	Testing and debugging
Internet Connection	Required for live API communication

4. SYSTEM DESIGN

The *Weather System Website* is a client-side web application that follows a straightforward architecture. The system is designed to be interactive, responsive, and efficient in delivering real-time weather information using an external API. The design focuses on clear separation of structure (HTML), presentation (CSS), and behavior (JavaScript).

This section covers both the logical architecture and functional workflow of the system.

4.1. System Architecture

The system can be divided into the following major components:

1. User Interface Layer (Frontend)

- Technologies Used: HTML, CSS, JavaScript
- Function: Provides input forms and displays output data. Users interact through this layer by entering a city name to request weather data.

2. API Communication Layer

- Technology Used: JavaScript (Fetch API)
- Function: Sends HTTP requests to the OpenWeatherMap API, processes responses, handles errors, and extracts relevant data.

3. Data Presentation Layer

- Function: Dynamically updates the content of the webpage (DOM) based on the API response, displaying data like temperature, humidity, weather conditions, and icons.

4.2. Functional Modules

(i). Input Module

- A simple text input field where the user enters the city name.
- Triggered by a search button or "Enter" key.

(ii). API Request Module

- Constructs the API URL using the entered city name and API key.
- Sends a fetch() request to the OpenWeatherMap API.
- Handles success and error responses.

(iii). Data Processing Module

- Extracts key weather attributes from the JSON response:
    - City Name
    - Temperature
    - Weather Description
    - Humidity
    - Wind Speed
    - Weather Icon Code

(iv). Display Module

- Dynamically updates the DOM using JavaScript.
- Renders the fetched weather data in a clean, styled format.
- Updates weather icons using image URLs based on icon codes from the API.

(v). Error Handling Module

- Detects and manages incorrect or empty city input.
- Handles API errors (e.g., 404 - City Not Found).

4.3. Design Considerations

- Responsiveness: Implemented using CSS media queries to ensure usability on both desktop and mobile devices.
- User Experience: Clean and minimal design, intuitive input, and immediate visual feedback.
- Scalability: The modular JavaScript design allows for easy integration of additional features (e.g., forecast data, map view).
- Performance: Lightweight, client-side-only application ensures fast load times and smooth interaction.

5. FEATURES

The *Weather System Website* is designed to offer a user-friendly interface with essential functionality for retrieving and displaying real-time weather information. Below is a comprehensive list of its key features:

(i). Real-Time Weather Updates

- The website fetches and displays live weather data from an external API.
- Users can instantly get up-to-date information for any city across the globe.

(ii) City-Based Weather Search

- Allows users to search by city name.
- Supports cities from all over the world, enhancing the global usability of the application.

(iii). Display of Key Weather Parameters

The application displays several important weather metrics, including:

- City Name (location verification)
- Temperature (in °C or °F)
- Weather Condition (e.g., Clear, Clouds, Rain)
- Humidity Level (in %)
- Wind Speed (in km/h or m/s)
- Weather Icon (graphical representation of current condition)

(iv) Responsive and Adaptive UI

- The website is mobile-friendly and adapts to different screen sizes (desktop, tablet, mobile).
- Achieved using CSS and media queries for a seamless user experience across devices.

(v) Interactive and Dynamic Interface

- No page reloads: The weather data is updated dynamically using JavaScript and DOM manipulation.
- Provides a smooth and interactive user experience.

(vi) . Error Handling and Validation

- Displays user-friendly messages for:
  - Invalid city names
  - Empty input fields
  - API errors or connection issues
- Ensures the application handles exceptions gracefully.

(vii) . Lightweight and Fast

- Built using only HTML, CSS, and JavaScript, with no heavy frameworks or dependencies.
- Loads quickly and performs efficiently on most modern web browsers.

(viii) . Clean and Minimalist Design

- Aesthetic layout with well-organized content.
- Focus on readability and usability, making the information easy to understand at a glance.

(ix). Extensible Code Structure

- Modular JavaScript code makes it easy to:
  - Add new features (like 5-day forecasts)
  - Integrate maps or geolocation features
  - Switch between units (Celsius/Fahrenheit)

6. OUTPUT AND CODE

Code:

Index.html

```
<!DOCTYPE html>


<head>
   <link rel="stylesheet" href="style.css">
   <link rel="stylesheet" href=
"https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">
   <link rel="stylesheet" href=
"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css">
   <link rel="stylesheet" href=
"https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=s
wap">
   <title>Weather App</title>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>


<body>
   <div class="container">
      <div class="weather-card">
         <h1 style="color: green;">
            Weather App
            </h1>
            <h3>
                Your Daily Weather Companion !
            </h3>
            <input type="text" id="city-input"
                placeholder="Enter city name">
            <button id="city-input-btn"
                    onclick="weatherFn($('#city-input').val())">
                    Get Weather
            </button>
            <div id="weather-info"
                class="animate__animated animate__fadeIn">
                <h3 id="city-name"></h3>
                <p id="date"></p>

                <p id="temperature"></p>
                <p id="description"></p>
                <p id="wind-speed"></p>
            </div>
        </div>
    </div>
    <script src=
"https://momentjs.com/downloads/moment.min.js">
```

```
    </script>
    <script src="script.js"></script>
</body>

</html>
```

Style.css

```
body {
    margin: 0;
    font-family: 'Montserrat', sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(to right, #4CAF50, #2196F3);
}

.container {
    text-align: center;
}
```

```
.weather-card {
    background-color: rgba(255, 255, 255, 0.95);
    border-radius: 20px;
    padding: 20px;
    box-shadow: 0 0 30px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s ease-in-out;
    width: 450px;
}

.weather-card:hover {
    transform: scale(1.05);
}

#city-input {
    padding: 15px;
    margin: 10px 0;
    width: 70%;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}

#city-input:focus {
    outline: none;
    border-color: #2196F3;
}

#city-input::placeholder {
    color: #aaa;
}

#city-input-btn {
    padding: 10px;
    background-color: #2196F3;
    color: #fff;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
}

#city-input-btn:hover {
    background-color: #1565C0;
}

#weather-info {
    display: none;
}

#weather-icon {
    width: 100px;
    height: 100px;
}

#temperature {
    font-size: 24px;
    font-weight: bold;
    margin: 8px 0;
}

#description {
    font-size: 18px;
    margin-bottom: 10px;
}

#wind-speed {
    font-size: 16px;
    color: rgb(255, 0, 0);
}
```

```
#date {
    font-size: 14px;
    color: rgb(255, 0, 0);
}
```

Script.js

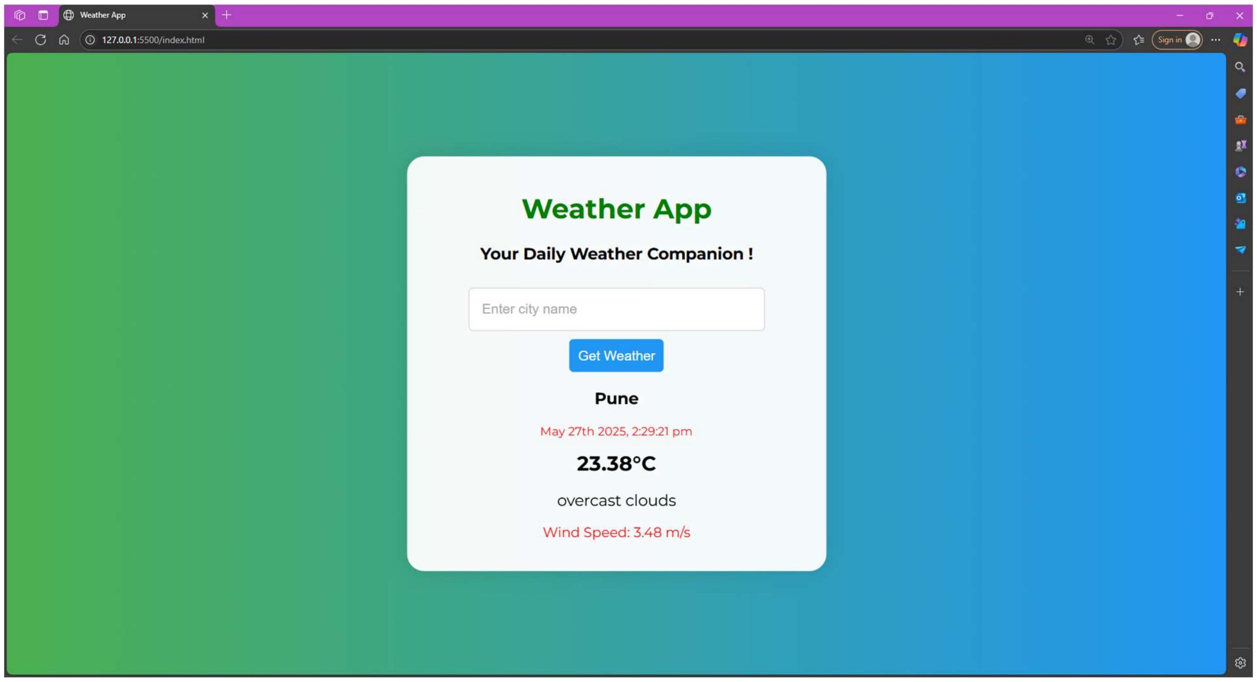
```
const url =
    'https://api.openweathermap.org/data/2.5/weather';
const apiKey =
    'f00c38e0279b7bc85480c3fe775d518c';

$(document).ready(function () {
    weatherFn('Pune');
});

async function weatherFn(cName) {
    const temp =
        `${url}?q=${cName}&appid=${apiKey}&units=metric`;
    try {
        const res = await fetch(temp);
        const data = await res.json();
        if (res.ok) {
            weatherShowFn(data);
        } else {
            alert('City not found. Please try again.');
        }
    } catch (error) {
        console.error('Error fetching weather data:', error);
    }
}

function weatherShowFn(data) {
    $('#city-name').text(data.name);
    $('#date').text(moment().
        format('MMMM Do YYYY, h:mm:ss a'));
    $('#temperature').
        html(`${data.main.temp}°C`);
    $('#description').
        text(data.weather[0].description);
    $('#wind-speed').
        html(`Wind Speed: ${data.wind.speed} m/s`);
    $('#weather-icon').
        attr('src',
        `...`);
    $('#weather-info').fadeIn();
}
```

OUTPUT:


Weather App

Your Daily Weather Companion !

Enter city name

Get Weather

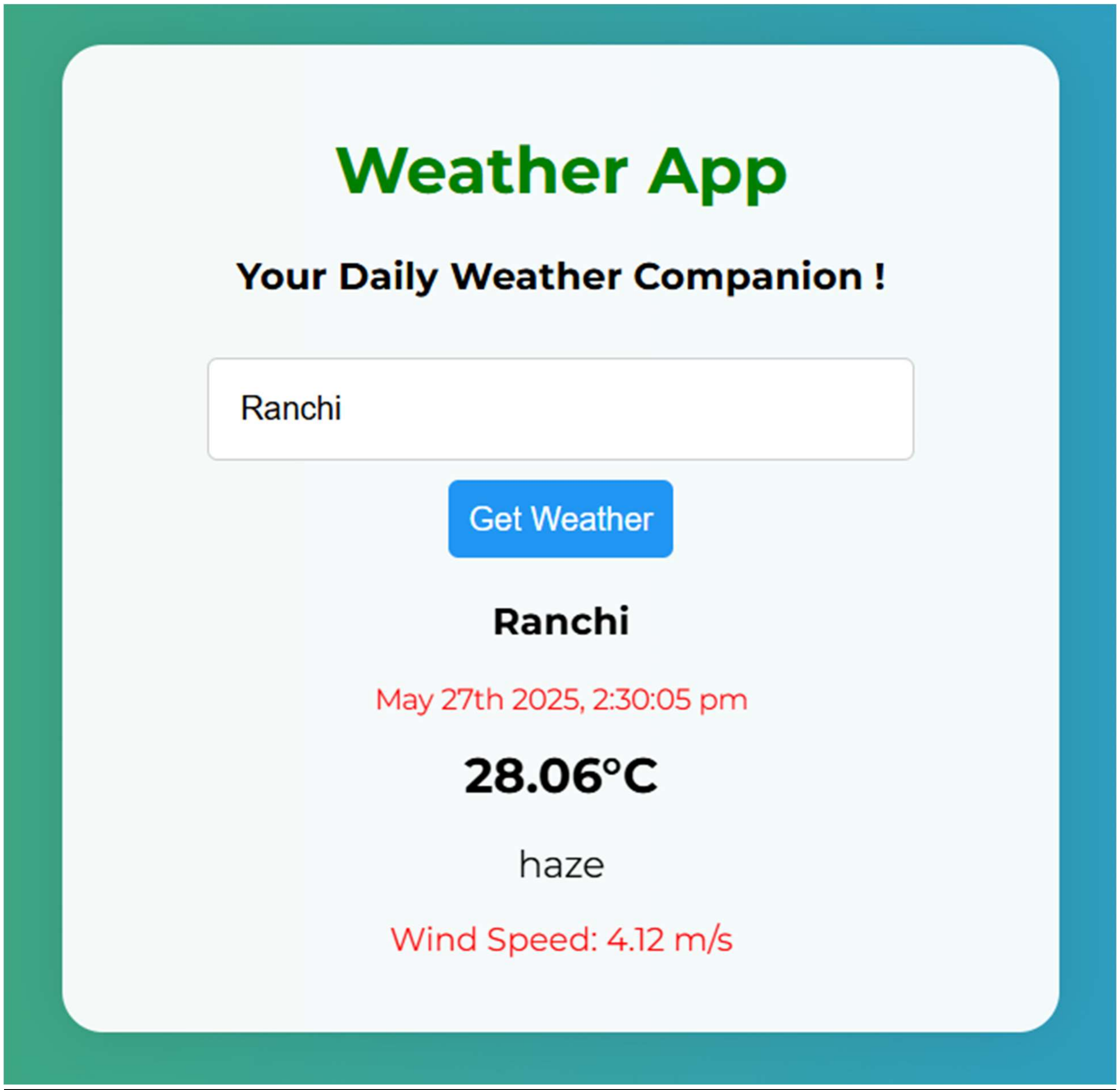
Pune

May 27th 2025, 2:29:21 pm

23.38°C

overcast clouds

Wind Speed: 3.48 m/s


Weather App

Your Daily Weather Companion !

Ranchi

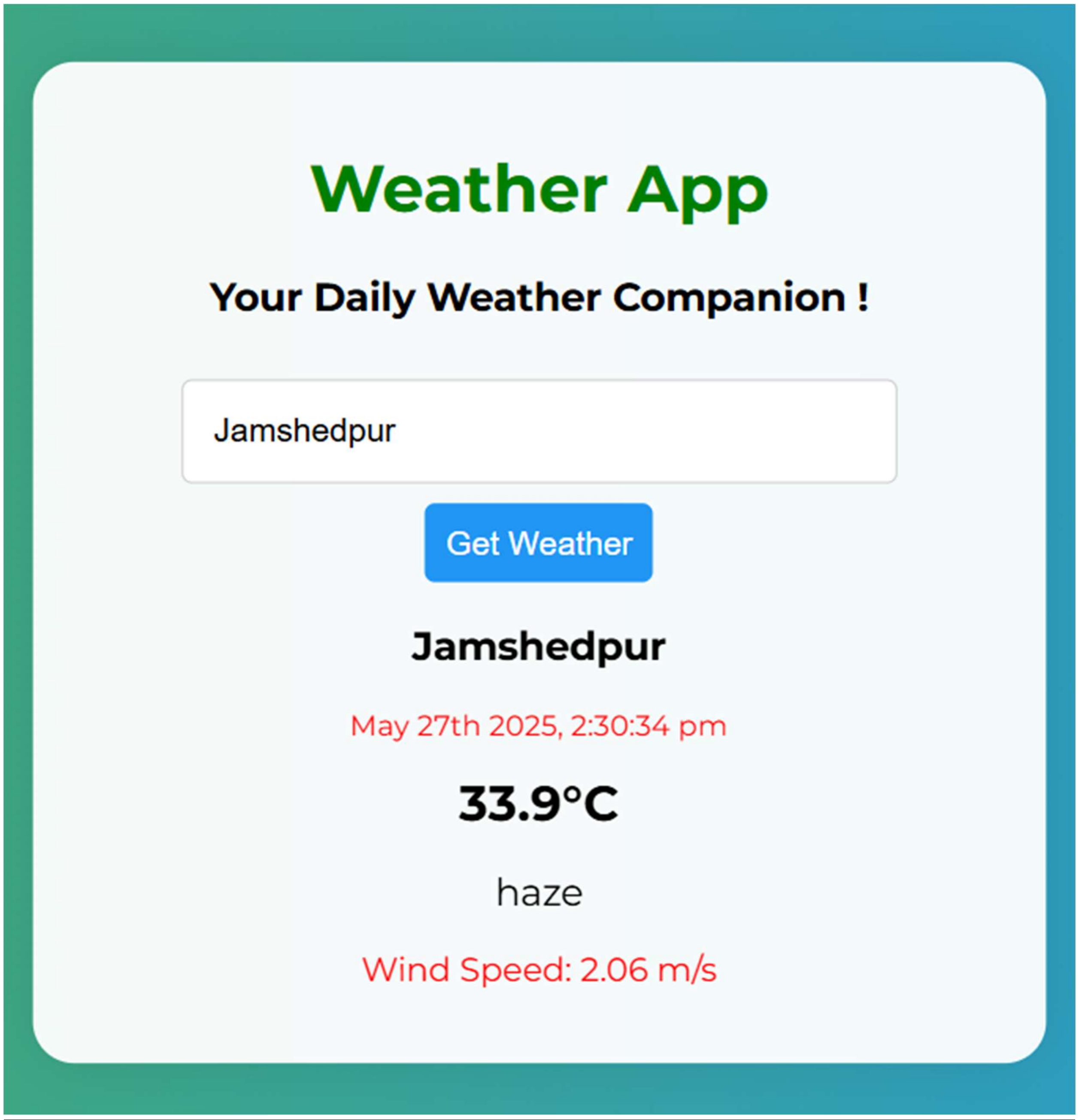
Ranchi

May 27th 2025, 2:30:05 pm

28.06°C

haze

Wind Speed: 4.12 m/s


Weather App
Your Daily Weather Companion !
Jamshedpur
Get Weather
Jamshedpur
May 27th 2025, 2:30:34 pm
33.9°C
haze
Wind Speed: 2.06 m/s

7.CONCLUSION

The *Weather System Website* project successfully demonstrates the implementation of a real-time weather information portal using core web development technologies—HTML, CSS, and JavaScript—alongside API integration. The system is designed to be user-friendly, responsive, and efficient, providing instant weather updates for cities around the world.

Through this project, a deeper understanding was gained of front-end web development principles, such as structuring content with HTML, styling interfaces with CSS, and handling interactivity and data manipulation using JavaScript.

The application meets its objectives by enabling users to:

- Input a city name and retrieve weather data in real time.
- View essential weather details such as temperature, humidity, wind speed, and conditions.
- Experience a clean and intuitive interface that adapts across device types.

Furthermore, this project laid a strong foundation for learning about modern web development challenges and solutions, such as handling API errors, managing user input validation, and ensuring a consistent user experience across devices. It also highlights the importance of separating concerns in code—structuring logic, presentation, and content in a maintainable and scalable way.

Key Takeaways:

- Real-world application of front-end technologies.
- Practical experience in using public APIs.
- Hands-on exposure to asynchronous programming.
- Importance of responsive design and user experience.

Scope for Future Enhancements:

While the current version of the application is basic and functional, it has ample scope for future development. Features such as geolocation-based weather retrieval, multi-day forecasts, dark/light themes, unit conversion (Celsius/Fahrenheit), and even voice input can be integrated to enhance user engagement and functionality.

In conclusion, the *Weather System Website* project not only fulfilled its technical and functional goals but also served as a strong learning platform. It demonstrates how even a simple tool, when well-designed and thoughtfully implemented, can address real-world needs and provide meaningful value to users.

8. REFERENCES

(i) OpenWeatherMap API Documentation
https://openweathermap.org/api
Used for retrieving real-time weather data including temperature, humidity, weather conditions, and more.

(ii) Mozilla Developer Network (MDN Web Docs)
https://developer.mozilla.org/
Primary resource for understanding HTML, CSS, JavaScript, and the Fetch API.

(iii) W3Schools – Web Development Tutorials
https://www.w3schools.com/
Referenced for syntax guides, examples, and explanations of HTML, CSS, and JavaScript concepts.

(iv) Stack Overflow
https://stackoverflow.com/
Used for debugging and resolving development-related issues and understanding common implementation patterns.

(v) CSS-Tricks
https://css-tricks.com/
Consulted for responsive design techniques, media queries, and layout optimization using CSS.

(vi) Visual Studio Code – Official Site
https://code.visualstudio.com/
Information about the code editor used for development and its useful extensions for web development.

(vii) GitHub Documentation
https://github.com/Kasshish29/Weather-system-website.git