PROJECT REPORT

on

" WEATHER FORECASTING APPLICATION "

(CSE V Sem mini project)
2024-2025

Submitted to:
Miss. Manika Manwal

Submitted by:
Navneet Bahuguna

Section: E2

Roll No.:39

Uni Roll No.: 2219138

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNIVERSITY DEHRADUN

CERTIFICATE

This is to certify that **Navneet Bahuguna**, a student of B.Tech in Computer Science and Engineering (CSE), 5th Semester at Graphic Era Hill University Dehradun, has successfully developed a mini-project titled **"WEATHER FORECASTING APPLICATION"**. The project was carried out independently to the best of the student's knowledge and skills, showcasing a strong understanding of technical and practical concepts. We acknowledge the student's hard work, dedication, and efforts in completing this project.

Miss. Manika Manwal
Class Coordinator
CSE E2, 5th Semester
(CSE Department)
Graphic Era Hill University, Dehradun

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Graphic Era Hill University, Dehradun, for providing me with the opportunity to undertake this mini-project titled “WEATHER FORECASTING APPLICATION” as part of my academic curriculum in the 5th Semester of B.Tech in Computer Science and Engineering (CSE).

I extend my heartfelt thanks to Miss. Manika Manwal, Our Class Coordinator, for her constant support, encouragement, and valuable guidance throughout the development of this project. Her insights and expertise have been instrumental in successfully completing this work.

I am also thankful to my classmates and peers for their cooperation and constructive feedback during the project. Finally, I am deeply grateful to my family and friends for their unwavering support and motivation, which inspired me to give my best efforts to this endeavor.

Navneet Bahuguna
B.Tech CSE, 5th Semester
Graphic Era Hill University, Dehradun

Introduction

The **Weather Forecasting Application** is an interactive web-based application designed to provide users with accurate and real-time weather updates for any specified location. With weather being an integral part of daily life and planning, this application simplifies weather monitoring by offering a user-friendly interface that presents key weather details such as temperature, humidity, wind speed, and weather conditions in a visually appealing format.
This project aims to demonstrate the practical application of web technologies and API integration in developing functional and user-centric applications, showcasing the technical skills of the developer.

Objective

The project has the following objectives:

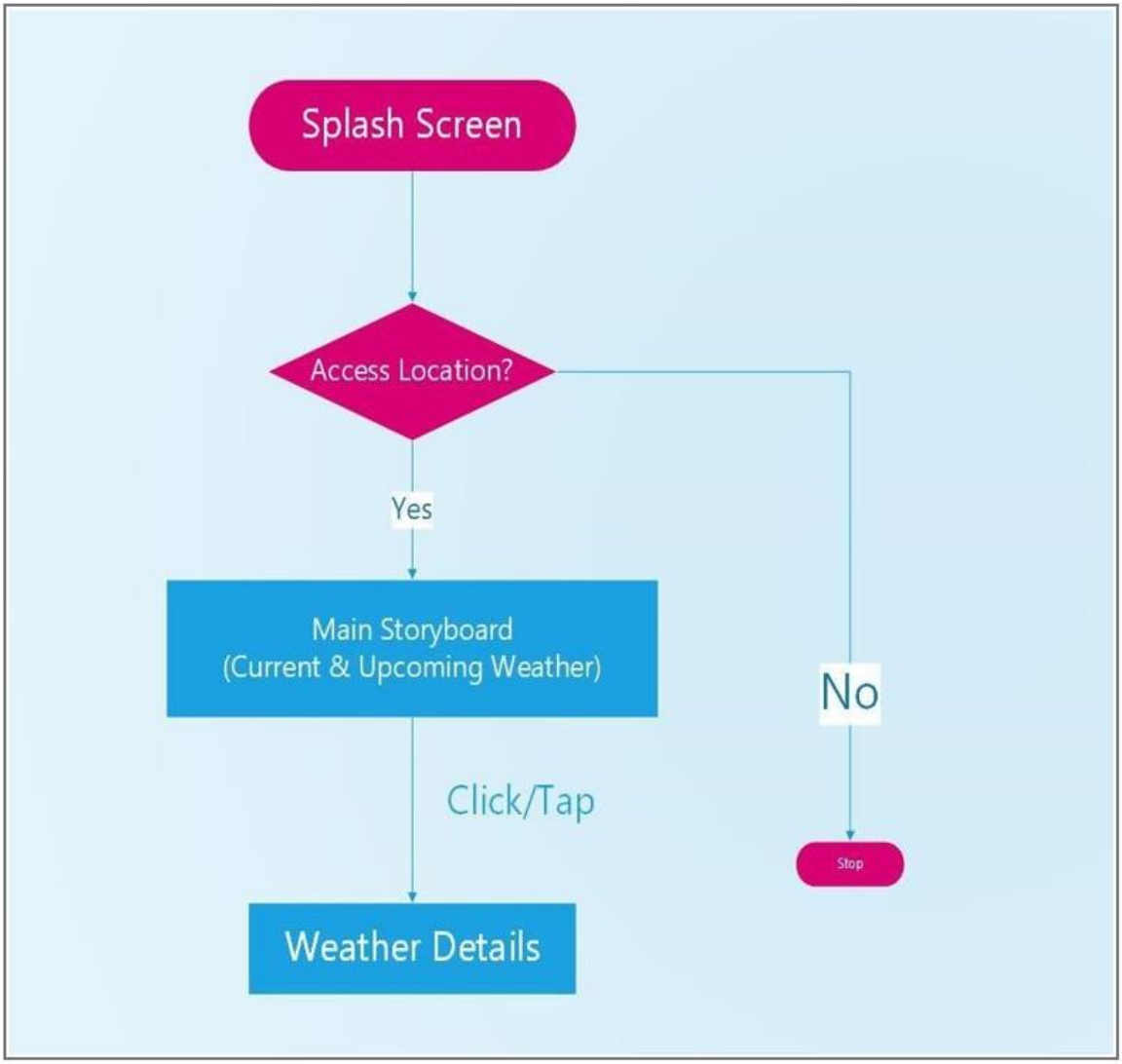
- To create a responsive web application that retrieves and displays real-time weather data.
- To enhance understanding of front-end technologies such as HTML, CSS, and JavaScript.
- To implement API integration for fetching weather data from reliable sources.
- To handle user inputs and potential errors effectively for an optimal user experience.

Technology Stack

The project utilizes the following technologies and tools:

- **HTML5**: For structuring the web application's content.
- **CSS3**: For designing and styling the user interface, ensuring it is visually appealing and user-friendly.
- **JavaScript (ES6)**: For implementing interactivity, fetching data from APIs, and dynamically updating the UI.
- **Weather API**: OpenWeatherMap API (or similar) to retrieve accurate and up-to-date weather information.
- **Visual Studio Code**: As the primary code editor for development.
- **Browser DevTools**: For debugging and testing the application across multiple devices and screen sizes.

weather forecast application design flow diagram



Features

The Weather Forecasting System provides the following features:

1. **User Input for Location**: Users can input the name of any city or location to get weather data.
2. **Real-Time Data Retrieval**: The application fetches weather data from the API and displays it in real time.
3. **Weather Metrics**:
    - Current Temperature (in Celsius/Fahrenheit).
    - Humidity Percentage.
    - Wind Speed (in km/h or mph).
    - General Weather Condition (e.g., Sunny, Rainy, Cloudy).
4. **Responsive Design**: The application is optimized for various devices, including mobile, tablet, and desktop screens.
5. **Error Handling**: Alerts users when an invalid city name or location is entered or if there is a network issue.
6. **Dynamic Updates**: Weather data and visuals are dynamically updated based on user input.

Methodology

1. **Planning**: Defining the scope of the project, selecting appropriate technologies, and identifying required functionalities.
2. **Design**: Creating a simple and user-friendly interface using HTML and CSS.
3. **Development**:
    - Writing JavaScript code to handle user input and API calls.
    - Integrating the Weather API to fetch real-time data for the specified location.
    - Processing and dynamically displaying the data in a structured format.
4. **Testing**:
    - Debugging JavaScript code for errors.
    - Verifying the application's performance on multiple devices.
5. **Deployment**: Ensuring the application is accessible and functions as expected in real-world scenarios.

Implementation

The Weather Forecasting System was implemented using the following steps:

- **HTML Structure**:
  Designed input fields, buttons, and placeholders for displaying weather data.
- **CSS Styling**:
  Created a responsive and visually appealing layout, ensuring usability across devices.
- **JavaScript Functionality**:
  - Implemented asynchronous functions to fetch data from the Weather API.
  - Processed JSON responses and extracted relevant weather details.
  - Handled errors such as incorrect user input or network issues.

Limitation

1. **Internet Dependency**: Requires an internet connection; offline mode could be added.
2. **Limited Data**: Only basic weather metrics are provided; more features can be added.
3. **No Geolocation**: Does not detect user location automatically.
4. **Error Handling**: Needs better handling for edge cases and API issues.
5. **Accuracy**: Dependent on one API; using multiple sources could improve reliability.
6. **UI Design**: Simple design; could be improved for better visuals and mobile optimization.
7. **Localization**: Currently only in English; multi-language support would enhance accessibility.

Source code:

HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Weather App Project JavaScript | Navneet</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <script src="script.js" defer></script>
  </head>
  <body>
    <h1>Weather Dashboard</h1>
    <div class="container">
      <div class="weather-input">
        <h3>Enter a City Name</h3>
        <input class="city-input" type="text" placeholder="E.g., New York,
London, Tokyo">
        <button class="search-btn">Search</button>
        <div class="separator"></div>
        <button class="location-btn">Use Current Location</button>
      </div>
      <div class="weather-data">
        <div class="current-weather">
          <div class="details">
            <h2>_______ ( ______ )</h2>
            <h6>Temperature: __°C</h6>
            <h6>Wind: __ M/S</h6>
            <h6>Humidity: __%</h6>
          </div>
        </div>
        <div class="days-forecast">
          <h2>5-Day Forecast</h2>
          <ul class="weather-cards">
            <li class="card">
              <h3>( ______ )</h3>
              <h6>Temp: __C</h6>
```

```
        <h6>Wind: __ M/S</h6>
        <h6>Humidity: __%</h6>
      </li>
      <li class="card">
        <h3>( ______ )</h3>
        <h6>Temp: __C</h6>
        <h6>Wind: __ M/S</h6>
        <h6>Humidity: __%</h6>
      </li>
      <li class="card">
        <h3>( ______ )</h3>
        <h6>Temp: __C</h6>
        <h6>Wind: __ M/S</h6>
        <h6>Humidity: __%</h6>
      </li>
      <li class="card">
        <h3>( ______ )</h3>
        <h6>Temp: __C</h6>
        <h6>Wind: __ M/S</h6>
        <h6>Humidity: __%</h6>
      </li>
      <li class="card">
        <h3>( ______ )</h3>
        <h6>Temp: __C</h6>
        <h6>Wind: __ M/S</h6>
        <h6>Humidity: __%</h6>
      </li>
    </ul>
   </div>
  </div>
 </div>

</body>
</html>
```

JavaScript:

```
const cityInput = document.querySelector(".city-input");
const searchButton = document.querySelector(".search-btn");
const locationButton = document.querySelector(".location-btn");
const currentWeatherDiv = document.querySelector(".current-weather");
const weatherCardsDiv = document.querySelector(".weather-cards");

const API_KEY = "YOUR-API-KEY-HERE"; // API key for OpenWeatherMap
API

const createWeatherCard = (cityName, weatherItem, index) => {
   if(index === 0) { // HTML for the main weather card
      return `<div class="details">
                  <h2>${cityName} (${weatherItem.dt_txt.split(" ")[0]})</h2>
                  <h6>Temperature: ${(weatherItem.main.temp -
273.15).toFixed(2)}°C</h6>
                  <h6>Wind: ${weatherItem.wind.speed} M/S</h6>
                  <h6>Humidity: ${weatherItem.main.humidity}%</h6>
               </div>
               <div class="icon">
                  <img
src="https://openweathermap.org/img/wn/${weatherItem.weather[0].ic
on}@4x.png" alt="weather-icon">
                  <h6>${weatherItem.weather[0].description}</h6>
               </div>`;
   } else { // HTML for the other five day forecast card
      return `<li class="card">
                  <h3>(${weatherItem.dt_txt.split(" ")[0]})</h3>
                  <img
src="https://openweathermap.org/img/wn/${weatherItem.weather[0].ic
on}@4x.png" alt="weather-icon">
                  <h6>Temp: ${(weatherItem.main.temp -
273.15).toFixed(2)}°C</h6>
                  <h6>Wind: ${weatherItem.wind.speed} M/S</h6>
                  <h6>Humidity: ${weatherItem.main.humidity}%</h6>
               </li>`;
   }
}
```

```
const getWeatherDetails = (cityName, latitude, longitude) => {
    const WEATHER_API_URL =
https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}&lon=
${longitude}&appid=${API_KEY};

    fetch(WEATHER_API_URL).then(response => response.json()).then(data
=> {
        // Filter the forecasts to get only one forecast per day
        const uniqueForecastDays = [];
        const fiveDaysForecast = data.list.filter(forecast => {
            const forecastDate = new Date(forecast.dt_txt).getDate();
            if (!uniqueForecastDays.includes(forecastDate)) {
                return uniqueForecastDays.push(forecastDate);
            }
        });

        // Clearing previous weather data
        cityInput.value = "";
        currentWeatherDiv.innerHTML = "";
        weatherCardsDiv.innerHTML = "";

        // Creating weather cards and adding them to the DOM
        fiveDaysForecast.forEach((weatherItem, index) => {
            const html = createWeatherCard(cityName, weatherItem, index);
            if (index === 0) {
                currentWeatherDiv.insertAdjacentHTML("beforeend", html);
            } else {
                weatherCardsDiv.insertAdjacentHTML("beforeend", html);
            }
        });
    }).catch(() => {
        alert("An error occurred while fetching the weather forecast!");
    });
}

const getCityCoordinates = () => {
    const cityName = cityInput.value.trim();
    if (cityName === "") return;
```

```
    const API_URL =
https://api.openweathermap.org/geo/1.0/direct?q=${cityName}&limit=1
&appid=${API_KEY};

    // Get entered city coordinates (latitude, longitude, and name) from
the API response
    fetch(API_URL).then(response => response.json()).then(data => {
        if (!data.length) return alert(No coordinates found for ${cityName});
        const { lat, lon, name } = data[0];
        getWeatherDetails(name, lat, lon);
    }).catch(() => {
        alert("An error occurred while fetching the coordinates!");
    });
}

const getUserCoordinates = () => {
    navigator.geolocation.getCurrentPosition(
        position => {
            const { latitude, longitude } = position.coords; // Get coordinates
of user location
            // Get city name from coordinates using reverse geocoding API
            const API_URL =
https://api.openweathermap.org/geo/1.0/reverse?lat=${latitude}&lon=$
{longitude}&limit=1&appid=${API_KEY};
            fetch(API_URL).then(response => response.json()).then(data => {
                const { name } = data[0];
                getWeatherDetails(name, latitude, longitude);
            }).catch(() => {
                alert("An error occurred while fetching the city name!");
            });
        },
        error => { // Show alert if user denied the location permission
            if (error.code === error.PERMISSION_DENIED) {
                alert("Geolocation request denied. Please reset location
permission to grant access again.");
            } else {
                alert("Geolocation request error. Please reset location
permission.");
            }
        });
```

```
}

locationButton.addEventListener("click", getUserCoordinates);
searchButton.addEventListener("click", getCityCoordinates);
cityInput.addEventListener("keyup", e => e.key === "Enter" &&
getCityCoordinates());
```

CSS:

```
@import
url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;50
0;600;700&display=swap');
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Open Sans', sans-serif;
}
body {
  background: #E3F2FD;
}
h1 {
  background: #5372F0;
  font-size: 1.75rem;
  text-align: center;
  padding: 18px 0;
  color: #fff;
}
.container {
  display: flex;
  gap: 35px;
  padding: 30px;
}
.weather-input {
  width: 550px;
}
.weather-input input {
  height: 46px;
  width: 100%;
  outline: none;
  font-size: 1.07rem;
  padding: 0 17px;
  margin: 10px 0 20px 0;
  border-radius: 4px;
  border: 1px solid #ccc;
}
.weather-input input:focus {
```

```
  padding: 0 16px;
  border: 2px solid #5372F0;
}
.weather-input .separator {
  height: 1px;
  width: 100%;
  margin: 25px 0;
  background: #BBBBBB;
  display: flex;
  align-items: center;
  justify-content: center;
}
.weather-input .separator::before{
  content: "or";
  color: #6C757D;
  font-size: 1.18rem;
  padding: 0 15px;
  margin-top: -4px;
  background: #E3F2FD;
}
.weather-input button {
  width: 100%;
  padding: 10px 0;
  cursor: pointer;
  outline: none;
  border: none;
  border-radius: 4px;
  font-size: 1rem;
  color: #fff;
  background: #5372F0;
  transition: 0.2s ease;
}
.weather-input .search-btn:hover {
  background: #2c52ed;
}
.weather-input .location-btn {
  background: #6C757D;
}
.weather-input .location-btn:hover {
  background: #5c636a;
```

```
}
.weather-data {
  width: 100%;
}
.weather-data .current-weather {
  color: #fff;
  background: #5372F0;
  border-radius: 5px;
  padding: 20px 70px 20px 20px;
  display: flex;
  justify-content: space-between;
}
.current-weather h2 {
  font-weight: 700;
  font-size: 1.7rem;
}
.weather-data h6 {
  margin-top: 12px;
  font-size: 1rem;
  font-weight: 500;
}
.current-weather .icon {
  text-align: center;
}
.current-weather .icon img {
  max-width: 120px;
  margin-top: -15px;
}
.current-weather .icon h6 {
  margin-top: -10px;
  text-transform: capitalize;
}
.days-forecast h2 {
  margin: 20px 0;
  font-size: 1.5rem;
}
.days-forecast .weather-cards {
  display: flex;
  gap: 20px;
}
```

```
.weather-cards .card {
  color: #fff;
  padding: 18px 16px;
  list-style: none;
  width: calc(100% / 5);
  background: #6C757D;
  border-radius: 5px;
}
.weather-cards .card h3 {
  font-size: 1.3rem;
  font-weight: 600;
}
.weather-cards .card img {
  max-width: 70px;
  margin: 5px 0 -12px 0;
}

@media (max-width: 1400px) {
  .weather-data .current-weather {
    padding: 20px;
  }
  .weather-cards {
    flex-wrap: wrap;
  }
  .weather-cards .card {
    width: calc(100% / 4 - 15px);
  }
}
@media (max-width: 1200px) {
  .weather-cards .card {
    width: calc(100% / 3 - 15px);
  }
}
@media (max-width: 950px) {
  .weather-input {
    width: 450px;
  }
  .weather-cards .card {
    width: calc(100% / 2 - 10px);
  }
```

```
}
@media (max-width: 750px) {
  h1 {
    font-size: 1.45rem;
    padding: 16px 0;
  }
  .container {
    flex-wrap: wrap;
    padding: 15px;
  }
  .weather-input {
    width: 100%;
  }
  .weather-data h2 {
    font-size: 1.35rem;
  }
}
```

Outcome

The Weather Forecasting Application successfully achieves its goal of providing an interactive and functional weather application. It efficiently displays real-time data, such as temperature, humidity, wind speed, and conditions, through a user-friendly interface.
The project demonstrates proficiency in HTML, CSS, and JavaScript, along with seamless integration of external APIs for real-time data retrieval. Responsive design principles ensure accessibility across devices, while problem-solving and debugging skills were honed during development. This project highlights readiness for real-world software tasks and practical application of theoretical knowledge.

Conclusion

The Weather Forecasting Application effectively applies web development skills to solve a real-world problem. It showcases expertise in front-end technologies, API integration, and responsive design while improving problem-solving, debugging, and user-focused development skills.

This project emphasizes creating reliable and efficient web applications and offers a strong foundation for future software development endeavors. Opportunities for enhancement include adding advanced features, improving performance, and expanding functionality. Overall, it marks significant progress in mastering modern web technologies.

References

1.Coding Nepal. "Weather App Project using HTML, CSS & JavaScript."
Available at: https://www.codingnepalweb.com/weather-app-project-html-javascript/

2.OpenWeatherMap API Documentation.
Available at: https://openweathermap.org/api

3.Mozilla Developer Network (MDN). "HTML, CSS, and JavaScript Documentation."
Available at: https://developer.mozilla.org/

4.W3Schools. "JavaScript Fetch API Tutorial."
Available at: https://www.w3schools.com/js/js_api_fetch.asp

5.Bootstrap Documentation (if you used it for responsive design).
Available at: https://getbootstrap.com/

6.Stack Overflow. "Common JavaScript Issues and Their Solutions."
Available at: https://stackoverflow.com/