Front-End UI/UX Mini Project

WEATHER DASHBOARD

Date: 25/09/25

Course: Front-End UI/UX Design Fundamentals.

Instructor Name: Ms. Nagaveena

Institution: Christ (Deemed to be Univeristy) Kengeri

Submitted by:

Adithyan CR – 2462306

[adithyan.cr@btech.christuniversity.in](mailto:adithyan.cr@btech.christuniversity.in)

Aveepsa Hatua – 2462319

[aveepsa.hatua@btech.christuniveristy.in](mailto:aveepsa.hatua@btech.christuniveristy.in)

Axilia Jennifer – 2462320

[axilia.jennifer@btech.christuniversity.in](mailto:axilia.jennifer@btech.christuniversity.in)

Abstract

The Weather Dashboard is a web-based application developed to provide real-time weather information in a simple and accessible manner. With the growing demand for instant updates on climatic conditions, this project demonstrates how modern web technologies can be integrated with external APIs to deliver meaningful data to users. The application allows users to input any city name and receive immediate information on the current weather, including temperature, humidity, and overall conditions, along with a 5-day forecast for better planning. It utilizes the OpenWeatherMap API as the data source, ensuring reliability and global coverage. Built with HTML, CSS, JavaScript, Bootstrap, and jQuery, the dashboard emphasizes a responsive, user-friendly design that adapts seamlessly across devices. By incorporating real-time data handling, error management, and dynamic UI updates, this project showcases the practical application of API integration in front-end development while providing a useful tool for everyday users.

Objectives

* To design a responsive and user-friendly weather dashboard.
* To fetch real-time weather data using an external API.
* To display both current weather and a 5-day forecast.
* To practice integrating Bootstrap and jQuery for enhanced design and interactivity.
* To handle errors gracefully, such as invalid city searches.

Scope of the Project

The scope of this project is limited to retrieving and displaying weather information based on city names provided by the user. It supports global cities as per the OpenWeatherMap API database. The project focuses on frontend design, real-time API integration, and responsive layouts. While it does not include backend storage or user accounts, the foundation is extendable for future enhancements.

Tools and Technology Used:

| **Tool/Technology** | **Purpose** |
| --- | --- |
| **HTML5** | To structure the web page and define content. |
| **CSS3** | To style the web page and enhance user experience. |
| **JavaScript (ES6)** | To handle API requests, parse JSON data, and update the DOM dynamically. |
| **jQuery** | To simplify AJAX requests and DOM manipulation. |
| **Bootstrap 5** | To ensure responsive design and prebuilt styling components. |
| **OpenWeatherMap API** | To fetch real-time weather and forecast data. |

Structural Overview of Coding Languages Used

* **HTML**: Provides the structure of the dashboard including input fields, cards, and layout.
* **CSS**: Defines custom styles for background gradients, card designs, and text formatting.
* **JavaScript**: Implements the core logic for API calls, data processing, and dynamic updates.
* **jQuery**: Used to simplify JavaScript code for event handling and AJAX requests.
* **Bootstrap**: Provides grid layout, responsive utilities, and styled components.

CSS Styling Overview

* A gradient background applied to the body to give a weather-themed appearance.
* Bootstrap cards customized with rounded corners and shadows for a modern look.
* Text styling and responsive spacing to ensure readability on different devices.
* Separate CSS file (style.css) to allow modular and clean styling.

Key Features

| **Feature** | **Description** |
| --- | --- |
| **Current Weather** | Displays city name, country, temperature, humidity, condition, and an icon. |
| **5-Day Forecast** | Shows daily weather summary at 12:00 with temperature and condition. |
| **Responsive Layout** | Works across desktop, tablet, and mobile devices. |
| **Error Handling** | Alerts the user if the entered city is not found. |

Challenges Faced and Solutions

| **Challenge** | **Solution** |
| --- | --- |
| Integrating API responses in JSON | Used JavaScript and jQuery to parse data and extract required fields. |
| Handling invalid city inputs | Implemented error handling using .fail() in jQuery’s $.get() method. |
| Displaying only one forecast per day | Filtered API data to include only entries at 12:00:00. |
| Making design mobile-friendly | Used Bootstrap grid system and responsive classes. |

Outcome

The Weather Dashboard successfully retrieves and displays accurate weather information and forecasts. It demonstrates seamless API integration and provides a responsive, user-friendly experience. The interface adapts well to different devices, and the project achieves its objectives.

Future Enhancements

* Add geolocation support to auto-detect and show weather for the user’s current location.
* Store previously searched cities for quick access.
* Provide more weather details such as wind speed, pressure, and sunrise/sunset times.
* Add multi-language support for wider accessibility.
* Develop a backend to store user preferences and history.

Sample Code:

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Weather Dashboard</title>

  <!-- Bootstrap -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

  <!-- jQuery -->

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <!-- Custom CSS -->

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <div class="container mt-4">

    <h2 class="text-center mb-4">🌤 Weather Dashboard</h2>

    <!-- Search Form -->

    <div class="row mb-4">

      <div class="col-md-8 offset-md-2">

        <div class="input-group">

          <input type="text" id="cityInput" class="form-control" placeholder="Enter city name">

          <button id="searchBtn" class="btn btn-primary">Search</button>

        </div>

      </div>

    </div>

    <!-- Current Weather -->

    <div id="currentWeather" class="card p-3 shadow mb-4 d-none">

      <h4>Current Weather</h4>

      <div id="weatherDetails"></div>

    </div>

    <!-- Forecast -->

    <h4 class="text-center mb-3 d-none" id="forecastTitle">5-Day Forecast</h4>

    <div id="forecast" class="row"></div>

  </div>

  <!-- Bootstrap JS -->

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

  <!-- Custom JS -->

  <script src="script.js"></script>

</body>

</html>

CSS:

body {

*background*: linear-gradient(to right, #83a4d4, #b6fbff);

*font-family*: Arial, sans-serif;

}

.card {

*border-radius*: 15px;

}

#forecast .card {

*margin*: 10px;

*background-color*: #ffffffdd;

}

h2 {

*font-weight*: bold;

}

Jscript:

*const* apiKey = "8fc82dec76f220f173f463bc1a04012c"; // Replace with your OpenWeatherMap API key

// Search button click

$("#searchBtn").click(*function* () {

*let* city = $("#cityInput").val();

  if (city !== "") {

    getWeather(city);

  }

});

// Fetch weather data

*function* getWeather(*city*) {

*const* currentWeatherURL = `https://api.openweathermap.org/data/2.5/weather?q=${*city*}&appid=${apiKey}&units=metric`;

*const* forecastURL = `https://api.openweathermap.org/data/2.5/forecast?q=${*city*}&appid=${apiKey}&units=metric`;

  // Current weather

  $.get(currentWeatherURL, *function* (*data*) {

    $("#currentWeather").removeClass("d-none");

*let* content = `

      <h5>${*data*.name}, ${*data*.sys.country}</h5>

      <p><b>Temperature:</b> ${*data*.main.temp}°C</p>

      <p><b>Humidity:</b> ${*data*.main.humidity}%</p>

      <p><b>Condition:</b> ${*data*.weather[0].main} <img src="https://openweathermap.org/img/wn/${*data*.weather[0].icon}.png"></p>

    `;

    $("#weatherDetails").html(content);

  }).fail(*function* () {

    alert("City not found. Please try again.");

  });

  // Forecast

  $.get(forecastURL, *function* (*data*) {

    $("#forecastTitle").removeClass("d-none");

    $("#forecast").html("");

    // Show one forecast per day at 12:00

*let* forecastList = *data*.list.filter(*item* *=>* *item*.dt\_txt.includes("12:00:00"));

    forecastList.forEach(*day* *=>* {

*let* date = new *Date*(*day*.dt\_txt).toLocaleDateString();

*let* card = `

        <div class="col-md-2 col-sm-4 col-6">

          <div class="card text-center p-2 shadow">

            <h6>${date}</h6>

            <img src="https://openweathermap.org/img/wn/${*day*.weather[0].icon}.png" alt="icon">

            <p><b>${*day*.main.temp}°C</b></p>

            <p>${day.weather[0].main}</p>

          </div>

        </div>

      `;

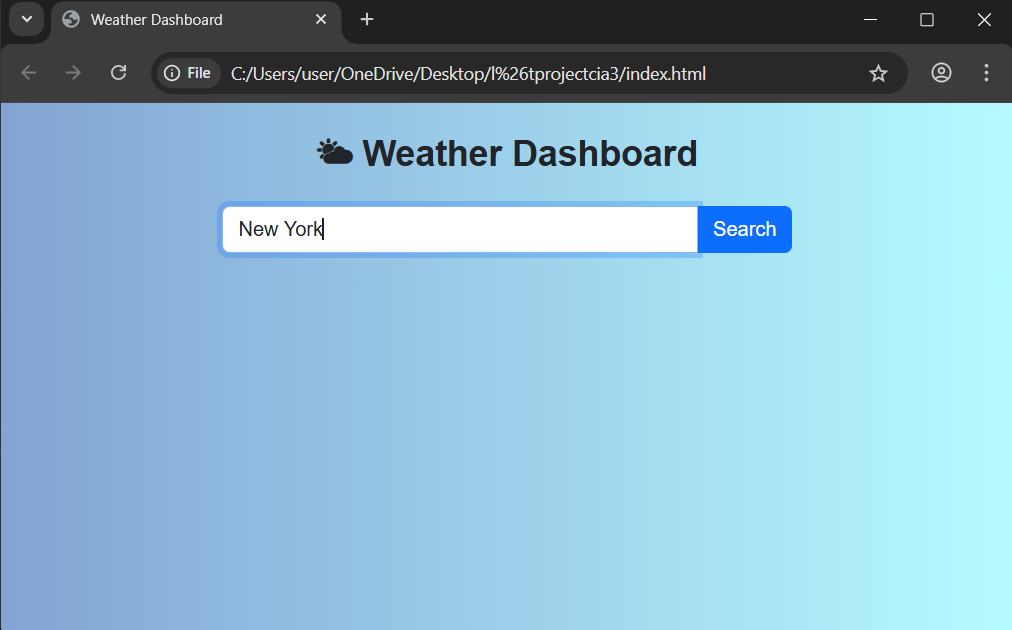
      $("#forecast").append(card);

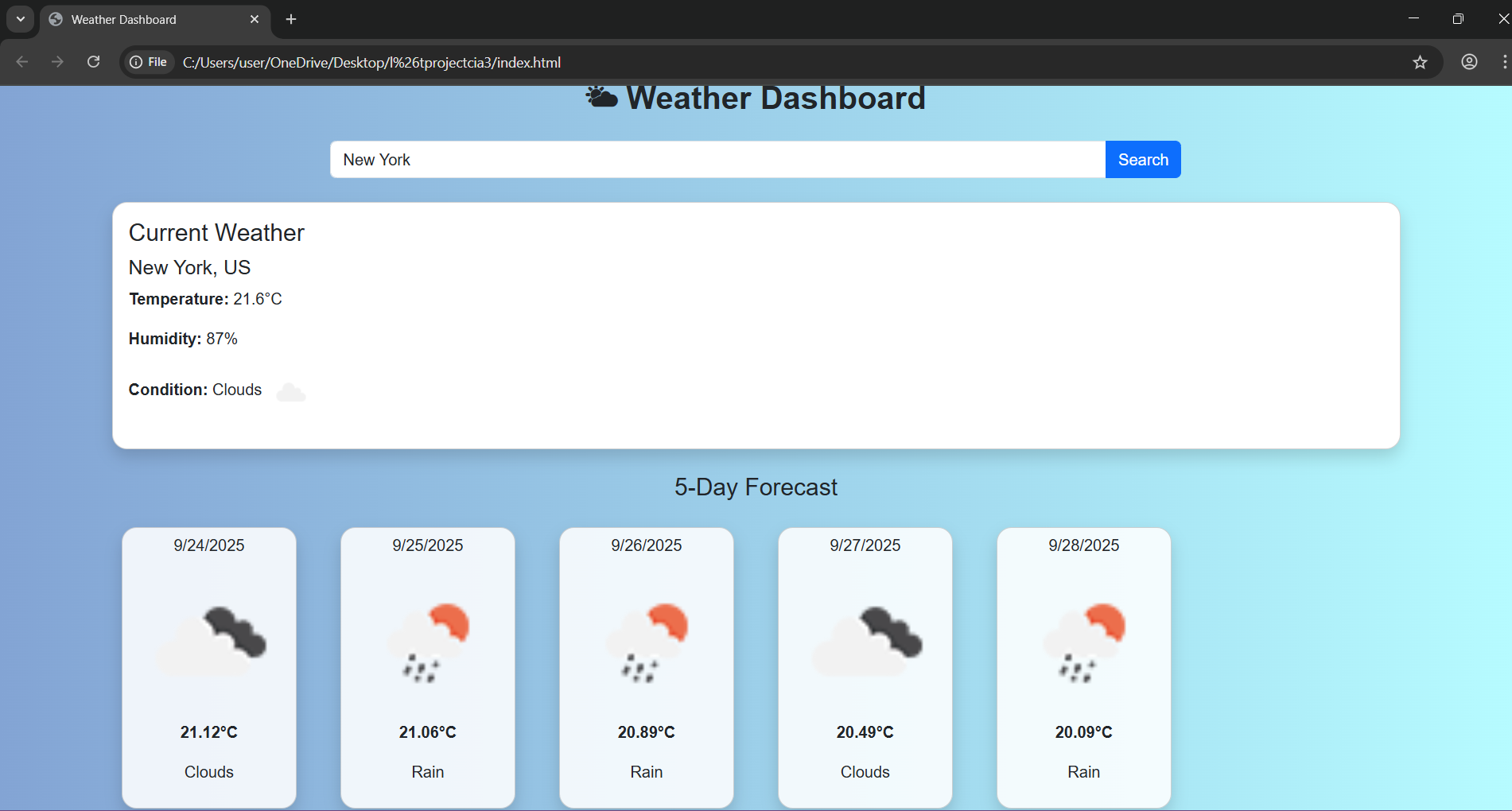
    });

  });

}

Output





Conclusion

The Weather Dashboard project showcases the integration of front-end technologies with an external API to create a real-time application. It highlights responsive design, API handling, and interactive features. While the current scope focuses on essential weather functionalities, the project can be extended with more advanced features, making it a solid foundation for a comprehensive weather application.

References

* L&T LMS: https://learn.lntedutech.com/Landinig/MyCourse