Weather API Mini Project – Sky Scan

Learner Details

- **Name: Shrayanth S**
- **Enrollment Number: SU625MR011**
- **Batch / Class: June 2025**
- **Assignment: Weather API Assignment**
- **Date of Submission: 27/07/2025**

Problem Solving Activity 1.1

1. Program Statement

Create a weather predictor app by giving city name as an input in the search bar and predict its weather report in real-time using weather API

---

2. Algorithm

1. Start
   - Load the HTML page (Sky Scan®).
   - Display header with logo and title.
   - Show search bar, search button, and empty weather display section.

2. User Input
   - User enters City Name in the input field (#inputValue).
   - User clicks Search button.

3. Validate Input
   - If input is empty:
     - Show alert:
       "Don't keep the search bar empty, Please do enter a city name!!!!"
     - Stop execution.

4. API Call Setup

- Prepare API endpoint using OpenWeatherMap:

bash

https://api.openweathermap.org/data/2.5/weather?q={city}&units=metric&appid=API_KEY

- Insert user-entered city and API key into URL.

5. Fetch Weather Data

- Use fetch() to send a GET request to the weather API.
- If response NOT OK:
    - Throw error message:
      "Hi, This is Your Sky Scan Weather Assistant... Please make sure you enter the correct city name..."
    - Show alert(error.message).

6. Process API Response

- Convert API response to JSON (data).
- Extract:
    - data.name → City Name
    - data.main.temp → Temperature in °C
    - data.weather[0].main → Weather Type (Clear, Clouds, Rain, etc.)
    - data.weather[0].description → Weather Description

7. Update UI with Weather Info

- Display:
    - City Name → #name
    - Temperature → #temp
    - Description + Message → #desc
    - Weather Icon → #weathericon (show it)

8. Display Weather Icon, Message & Safety Tip

- Check weatherMain:
    - If Clear → Show sunny icon + tip:
      "Wear sunglasses and stay hydrated."
    - If Rain → Show rain icon + tip:
      "Carry an umbrella."
    - If Thunderstorm → Tip:
      "Stay indoors."
- If weather type not found in iconMap, hide icon and tip.

9. Change Background Dynamically

- Based on weatherMain:
    - Clear → Blue sky background
    - Clouds → Cloudy background
    - Rain → Rainy background
    - Snow → Snow background
    - Mist → Foggy background
- Apply image using document.body.style.background.

10. Error Handling

- If API call fails or city not found:
    - Clear all displayed data.
    - Hide icon and tip.
    - Reset background color to default.
    - Show alert with error message.

11. End

- Wait for next user search.

---

3. Pseudocode

START

DISPLAY header with title and logo

DISPLAY input field and search button

DISPLAY empty weather info section

FUNCTION getWeather():

city ← value from input field

IF city is empty THEN

SHOW alert "Don't keep the search bar empty, Please do enter a city name!!!!"

RETURN

ENDIF

TRY:

apiURL ← "https://api.openweathermap.org/data/2.5/weather?q=" + city + "&units=metric&appid=API_KEY"

response ← fetch(apiURL)

IF response not OK THEN

THROW error with message

ENDIF

data ← response.json()

name ← data.name

temperature ← data.main.temp

weatherMain ← data.weather[0].main

description ← data.weather[0].description

SET #name = name

SET #temp = temperature + "°C"

```
    IF weatherMain in iconMap THEN
        icon ← iconMap[weatherMain].icon
        text ← iconMap[weatherMain].text
        tip ← iconMap[weatherMain].tip

        DISPLAY icon
        SET #desc = description + text
        SET #tip = tip
    ELSE
        HIDE icon
        SET #desc = description
        CLEAR #tip
    ENDIF

    SWITCH weatherMain:
        CASE "Clear": background = sunny image
        CASE "Clouds": background = cloudy image
        CASE "Rain": background = rainy image
        CASE "Snow": background = snowy image
        CASE "Drizzle": background = drizzle image
        CASE "Thunderstorm": background = thunderstorm image
        CASE "Mist": background = mist image
        DEFAULT: background = default image
    END SWITCH

    APPLY background to document body

CATCH error:
```

CLEAR #name, #temp, #desc, #tip

HIDE icon

SET background to default color

SHOW alert with error message

END TRY

END FUNCTION

ON search button click:

CALL getWeather()

END

---

4. Program Code

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sky Sacn ®</title>
    <link rel="stylesheet" href="style.css">
    <link rel="icon" type="image/x-icon" href="sky scan.jpg">
</head>
<body>
    <header class="header">
        <h1 class="header-title">Sky Scan &reg;</h1>
        <img src="sky scan.jpg"
            alt="Sky Scan Logo" class="header-logo">
```

```
    </header>

    <div class="container">
        <div class="input">
            <input type="text" id="inputValue" placeholder="Enter the City Name" required>
            <button id="button" class="button">Search 🔍</button>
        </div>
        <div class="displayWeather">
            <h2 id="name"></h2>
            <p id="temp"></p>
            <p id="desc"></p>
            <img id="weathericon" style="width: 90px; height: 90px; display: none;">
            <p id="tip" style="font-weight:bold; color:#007acc;"></p>
        </div>
    </div>

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


CSS CODE

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: rgb(224, 203, 120);
    display: flex;
    justify-content: center;
    align-items: center;
```

```
    height: 100vh;
    transition: background 0.8s ease-in-out;
}

.container {
        position: fixed;
    background: #ffffff;
    padding: 70px 30px;
    border-radius: 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    width: 300px;
    text-align: center;
}

.input {
    margin-bottom: 15px;
}

input[type="text"] {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 20px;
    font-size: 16px;
    outline: none;
    transition: border 0.3s;
}

input[type="text"]:focus {
```

```
    border-color: #007acc;
}

.button {
    margin-top: 10px;
    width: 100%;
    padding: 10px;
    background-color: #007acc;
    color: #fff;
    border: none;
    border-radius: 15px;
    font-size: 17px;
    cursor: pointer;
    transition: background 0.3s;
}

.button:hover {
    background-color: #27b722;
}

.displayWeather h2 {
    margin: 15px 0 10px;
    font-size: 25px;
    color: #6028a9;
}

.displayWeather p {
    margin: 5px 0;
    font-size: 14px;
```

```
    color: #333;
}

#weathericon {
    margin-top: 10px;
}

#tip {
    margin-top: 10px;
    font-weight: bold;
    font-size: 14px;
    color: #007acc;
}

.header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    background: #007acc;
    color: white;
    text-align: center;
    padding: 14px ;
    font-size: 24px;
    font-weight: bold;
    font-family: Arial, sans-serif;
    z-index: 1000;
}
```

```
body {
    margin-top: 80px;
}


.header {
    position: fixed;
    top: 0;
    left: 0;
    width: 1570px;
    height: 75px;
    background: #007acc;
    color: white;
    display: flex;
    justify-content:space-evenly;
    align-items:center;
    padding: 20px 20px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
    z-index: 1000;
}

.header-title {
    font-size: 35px;
    font-family: Arial, sans-serif;
    margin: 0;
    color: black;
}

.header-logo {
    width: 75px;
```

```
    height: 75px;
    border-radius: 98%;
}

body {
    margin-top: 60px;
    background: rgb(224, 203, 120);
    font-family: Arial, sans-serif;
}
```

JAVASCRIPT CODE

```
const getWeather = async () => {
    const city = document.getElementById('inputValue').value.trim();
    const nameVal = document.getElementById('name');
    const temp = document.getElementById('temp');
    const desc = document.getElementById('desc');
    const tipEl = document.getElementById('tip');
    const weatherIcon = document.getElementById('weathericon');

    if (!city) {
        alert("Don't keep the search bar empty, Please do enter a city name!!!!");
        return;
    }

    try {
        const response = await
fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=a2c2f6
f87d64ff29fa0bf08ad41b5cc4`);

        if (!response.ok) {
```

```
        throw new Error("Hi, This is Your Sky Scan Weather
Assistant                    Please do check for spelling mistake, Please make sure you
enter the correct city name in the search
bar.
          Thanking You, Sky Scan Weather Assistant!!");
    }


    const data = await response.json();



    nameVal.innerText = data.name;
    temp.innerText = `${data.main.temp}°C`;


    const weatherMain = data.weather[0].main;
    const description = data.weather[0].description;


    const iconMap = {
      Clear: {
        icon: "https://cdn-icons-png.flaticon.com/512/869/869869.png",
        text: "It's a bright and sunny day! 🌞",
        tip: "Wear sunglasses and stay hydrated. Use sunscreen to protect your skin."
      },
      Clouds: {
        icon: "https://cdn-icons-png.flaticon.com/512/1163/1163624.png",
        text: "Looks like it's cloudy outside. ☁",
        tip: "Great weather for outdoor walks, but keep an umbrella handy just in case."
      },
      Rain: {
        icon: "https://cdn-icons-png.flaticon.com/512/1163/1163657.png",
```

```
    text: "It's raining! Don't forget your umbrella. 🌧️",
    tip: "Carry an umbrella or raincoat. Watch out for slippery roads."
  },
  Snow: {
    icon: "https://cdn-icons-png.flaticon.com/512/642/642102.png",
    text: "Snow is falling! Stay warm. ❄️",
    tip: "Wear warm clothes and non-slip boots. Drive slowly on icy roads."
  },
  Drizzle: {
    icon: "https://cdn-icons-png.flaticon.com/512/4834/4834551.png",
    text: "Light drizzle is on the way. ☔",
    tip: "Keep an umbrella nearby. Roads might still be slippery."
  },
  Thunderstorm: {
    icon: "https://cdn-icons-png.flaticon.com/512/1146/1146869.png",
    text: "Thunderstorms! Stay safe indoors. ⛈️",
    tip: "Avoid going outside. Stay away from trees, poles, and electrical wires."
  },
  Mist: {
    icon: "https://cdn-icons-png.flaticon.com/512/4005/4005901.png",
    text: "Misty weather ahead. Drive carefully! 🌫️",
    tip: "Use fog lights while driving and maintain a slow speed."
  }
};

if (iconMap[weatherMain]) {
  weatherIcon.src = iconMap[weatherMain].icon;
  weatherIcon.style.display = "inline-block";
```

```
        desc.innerText = `${description}\n${iconMap[weatherMain].text}`;
        tipEl.innerText = iconMap[weatherMain].tip;
    } else {
        weatherIcon.style.display = "none";
        desc.innerText = description;
        tipEl.innerText = "";
    }

    let bgImage = "";
    switch (weatherMain) {
        case "Clear":
            bgImage = "url('https://media.istockphoto.com/id/476263042/photo/fluffy-clouds-in-the-sky.jpg?s=612x612&w=0&k=20&c=XcNQ8dq2FhC518Ef1hkOwrblUz5GxVhBo-AYuHV6fFI=')";
            break;
        case "Clouds":
            bgImage = "url('https://t4.ftcdn.net/jpg/05/13/26/73/360_F_513267391_QEmNGeOFLLqrILTnoq21dReUPp5UsoNr.jpg')";
            break;
        case "Rain":
            bgImage = "url('https://img.freepik.com/free-photo/weather-effects-composition_23-2149853295.jpg?semt=ais_hybrid&w=740&q=80')";
            break;
        case "Snow":
            bgImage = "url('https://wjla.com/resources/media2/16x9/full/1015/center/80/be94f27f-c70a-4e6c-b3cc-9a448da929b8-large16x9_SnowfallsinEllicottCityVeronicaJohnson.JPG')";
            break;
        case "Drizzle":
            bgImage = "url('https://www.shutterstock.com/image-photo/drizzle-on-windshield-evening-600nw-667415725.jpg')";
            break;
```

```
        case "Thunderstorm":
            bgImage = "url('https://ambientweather.com/media/blog/pexels-alexandre-bringer-3637060-_1_.jpg')";
            break;
        case "Mist":
            bgImage = "url('https://www.shutterstock.com/image-photo/landscape-heavy-foggy-road-winter-260nw-1594521517.jpg')";
            break;
        default:
            bgImage = "url('https://media.istockphoto.com/id/157592559/photo/green-landscape.jpg?s=612x612&w=0&k=20&c=dfieVqumfZZ2F3Y3cCi1w2LnkuWIZabjsjT3KcL32_0=')";
    }

    document.body.style.background = `${bgImage} no-repeat center center fixed`;
    document.body.style.backgroundSize = "cover";

  } catch (error) {
    nameVal.innerText = "";
    temp.innerText = "";
    desc.innerText = "";
    tipEl.innerText = "";
    weatherIcon.style.display = "none";
    document.body.style.background = "rgb(224, 203, 120)";

    alert(error.message);
  }
};

document.getElementById('button').addEventListener('click', getWeather);
```

5. Screenshots of Output

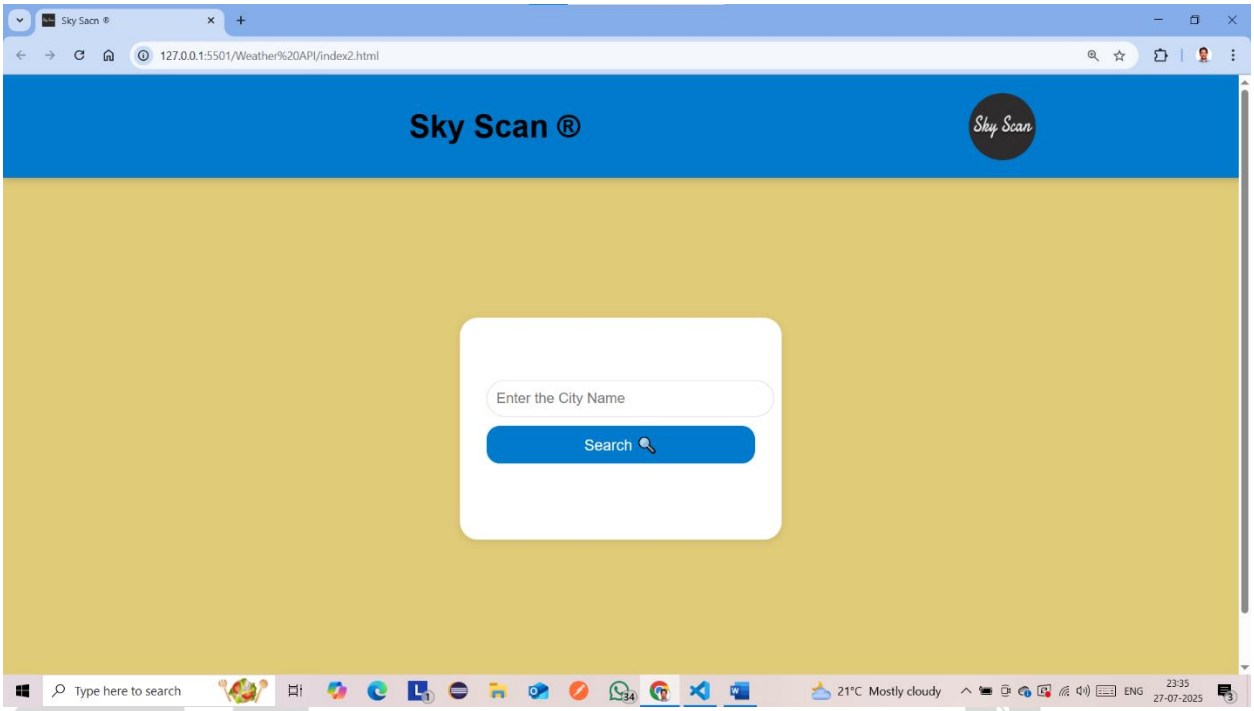

Figure 1 : Sky Scan Webpage Home Page

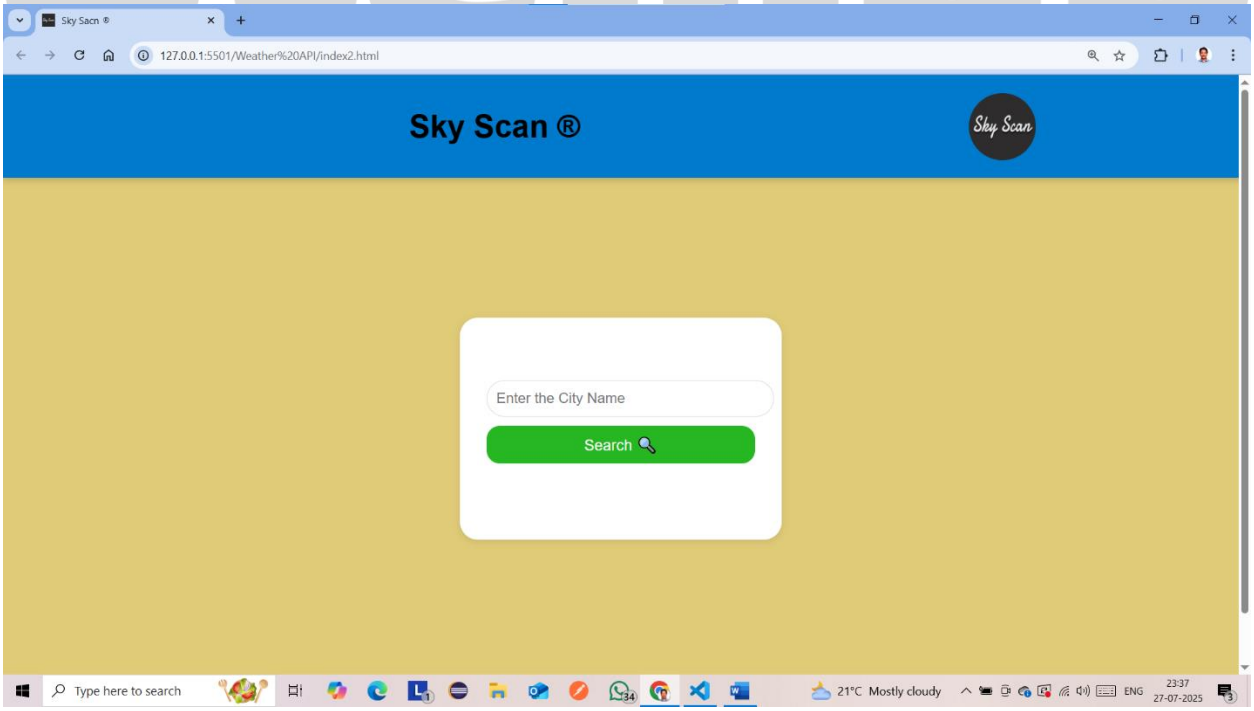

Figure 2 : Color of the search bar changes when mouse is hovered on it

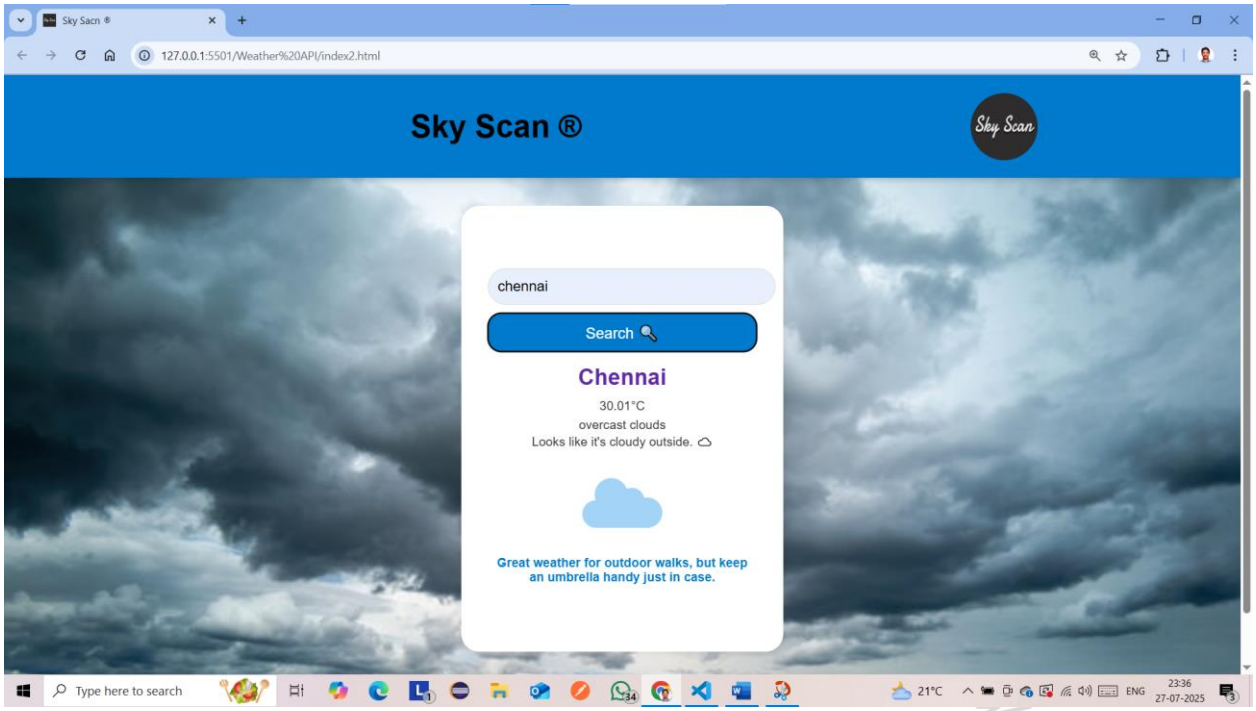

Figure 3 : Output when searched for Chennai's Weather – it is cloudy

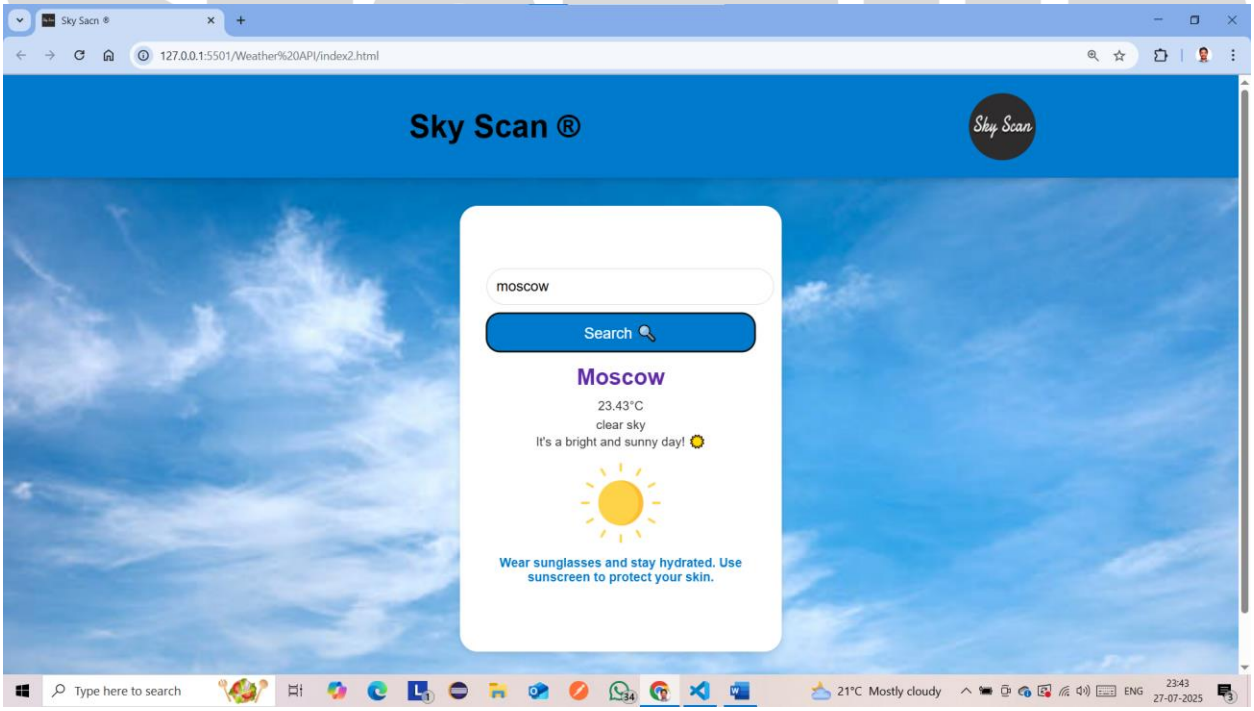

Figure 4 : Output when searched for Moscow Weather – it is clear sky

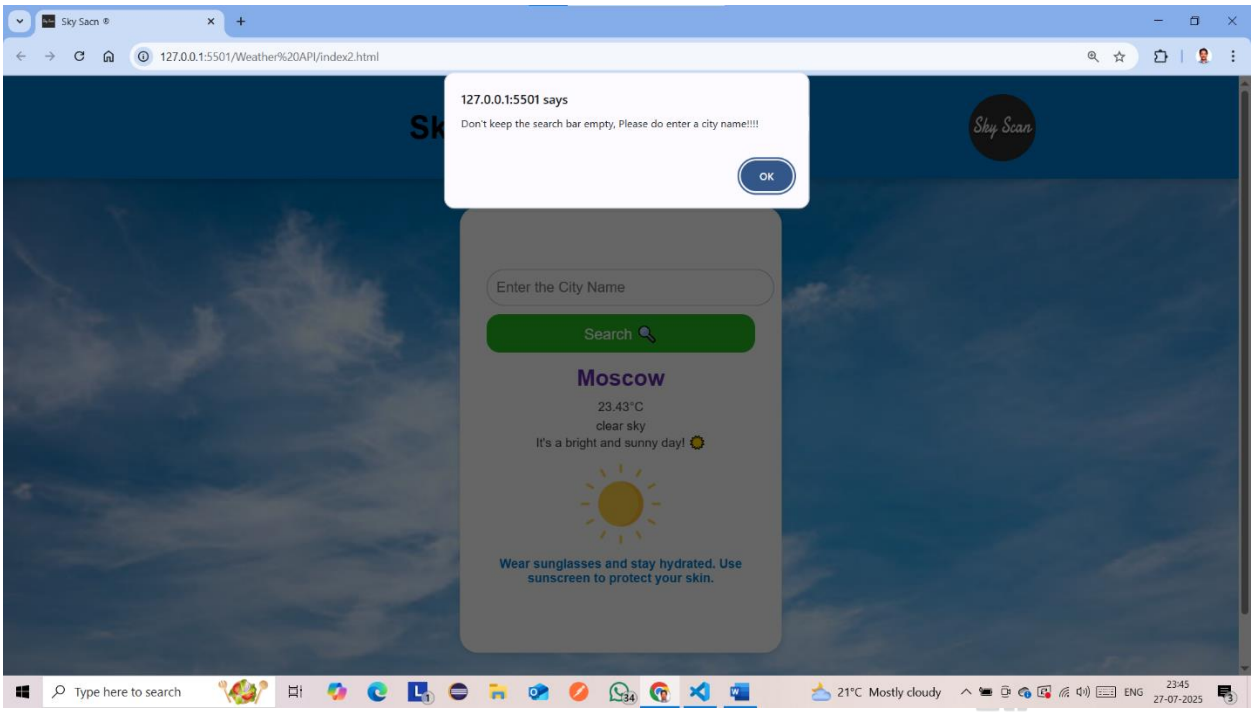

Figure 5 : Alert Message appears when you press search bar without entering any city name

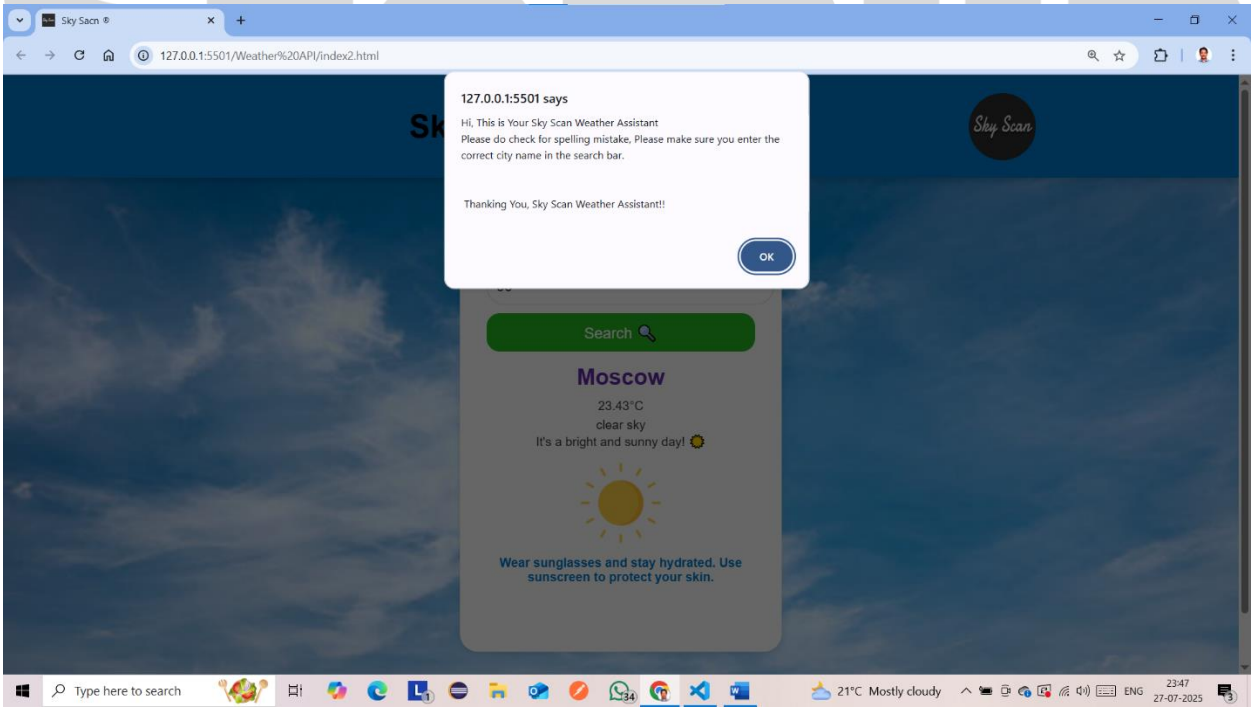

Figure 6 : If user enters an invalid city name it throws an alert message to do a spell check

6. Observation / Reflection

While working on this assignment, one of the main challenges I faced was integrating the OpenWeatherMap API and making sure the app handled errors correctly when users entered invalid city names. Another tricky part was dynamically updating the background images and icons based on different weather conditions, as I wanted the app to look visually appealing and responsive.

Through this task, I learned how to work with real-time data using the Fetch API and async/await in JavaScript. I also became more confident in manipulating the DOM dynamically and applying conditional logic to create a more interactive and user-friendly experience.

If I were to improve this project in the future, I would add a loading animation while fetching data to make the app feel smoother, and allow users to press Enter to search for convenience. I'd also like to make the background transitions more seamless with CSS animations and display error messages inside the page instead of using alerts. Adding features like recent searches and a cleaner UI would also make the app more professional and enjoyable to use.