# Project Title: Weather Forecast Application Development in JavaScript:

This project creates a weather application called **MJ-Weather**, which provides current weather information and a 4-day forecast for a given city or the user's current location. This uses **OpenWeather API** to fetch weather data and **TailwindCSS** for styling. The primary features include:

* Search functionality
* Time and date
* Location based weather
* Recent searches
* Weather display(temperature, rain, humidy and wind)
* Responsive.

## HTML and CSS(tailwindCSS):

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>MJWeather</title>

  <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">

  <script src="app.js" defer></script>

</head>

* <html lang="en">: Sets the page's main language to English.
* <meta charset="UTF-8">: Makes sure characters display correctly using UTF-8 encoding.
* <meta name="viewport">: Helps the page adjust to fit all screen sizes for better responsiveness.
* Title: Sets the page title as "MJWeather."
* TailwindCSS: Adds a link to the TailwindCSS framework for styling.
* <script src="app.js" defer>: Loads the app's JavaScript (app.js) and runs it only after the HTML is fully loaded.

Header Section:

<header class="bg-blue-500 text-white py-6">

    <div class="container mx-auto text-center">

      <h1 class="text-4xl font-bold">MJ-Weather</h1>

      <p class="text-lg mt-2">A trusted weather companion</p>

    </div>

  </header>

* It is used for title and tagline
* bg-blue-500: Blue background color.
* text-white: White text color.
* py-6: Vertical padding of 6 units.
* text-center: Centers the text horizontally.
* text-4xl: Sets the font size for the title.
* font-bold: Makes the title bold.
*  text-lg and mt-2: Medium-sized text with top margin for the tagline.

Top section:

 <!--Date/Time, Search, Current Location -->

    <div class="flex flex-col lg:flex-row items-center justify-between gap-6 mb-6">

      <!-- Time and Date -->

      <div id="time-date" class="text-left text-lg font-medium text-white">

        <!-- Dynamic Time and Date -->

      </div>

      <!-- Search Section -->

      <div class="relative flex items-center mb-2 gap-1">

        <!-- Search Input -->

        <input id="search-input" type="text" placeholder="Enter city..."

               class="p-3 border border-blue-500 rounded-l-md focus:ring-2 focus:ring-blue-400" autocomplete="off">

        <!-- Search Button -->

        <button id="search-btn"

                class="bg-blue-500 text-white px-5 py-3 rounded-r-md hover:bg-blue-600">

          Search

        </button>

        <!-- Dropdown List -->

        <ul id="city-dropdown" class="absolute top-full left-0 w-full bg-white shadow-md mt-1 rounded-lg hidden z-10 max-h-48 overflow-y-auto">

          <!-- City List will be inserted here dynamically -->

        </ul>

      </div>

      <!-- Current Location Button -->

      <button id="current-location-btn"

              class="bg-blue-500 text-white px-5 py-3 rounded-md hover:bg-blue-800 mt-2 lg:mt-0">

        Use Current Location

      </button>

    </div>

* Top section consists of date/time, search section, search button, dropdowns and users location.
* I used only 4 colors (black for background, white for input field , blue-500 for buttons and blue-600 for hovering effect)
* In this section time will be displayed dynamically when the user opens the page.
* Recent searched cities will be present in the dropdown.
* User can also know his locations weather.

Main Section:

<!-- Weather Display -->

    <div class="flex flex-col lg:flex-row gap-6 justify-center">

      <!-- Today's Weather -->

      <div id="today-weather" class="flex-1 bg-blue-500 text-white rounded-lg shadow-lg p-8 hidden flex flex-col items-center text-center">

        <!-- Today's Weather Details -->

      </div>

      <!-- 4-Day Forecast -->

      <div id="forecast" class="flex-1 bg-black grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-2 gap-4 hidden justify-center">

        <!-- Forecast Cards -->

      </div>

    </div>

  </main>

</body>

</html>

* In main section the weather of a particular city will be displayed.
* Half of the main section will be consist of today weather of a particular place.
* Remaining half will be consisting of next 4 days weather in 2/2 manner and I used grid method for that.
* The 5 cards background color will be blue-500 which gives a good look.

## JavaScript:

API key and elements:

const apiKey = 'ddf32ed4ba6a24adad46657a4e4acdcb'; // Replace with your API key

const searchInput = document.getElementById('search-input');

const cityDropdown = document.getElementById('city-dropdown');

const searchBtn = document.getElementById('search-btn');

const currentLocationBtn = document.getElementById('current-location-btn');

const todayWeatherContainer = document.getElementById('today-weather');

const forecastContainer = document.getElementById('forecast');

const timeDateContainer = document.getElementById('time-date');

* API Key: The API key used to authenticate requests to OpenWeatherMap.
* **DOM Elements**: References to HTML elements for interacting with the search bar, buttons, dropdown, and weather display sections.

Function:

1. To update time:

function updateTimeAndDate() {

  const now = new Date();

  const options = { year: 'numeric', month: 'long', day: 'numeric', hour: '2-digit', minute: '2-digit' };

  timeDateContainer.textContent = now.toLocaleString('en-US', options);

}

setInterval(updateTimeAndDate, 60000);

updateTimeAndDate();

* Updates the current date and time every 60 seconds using setInterval.
* toLocaleString: Formats the date/time in a human-readable format.

2.Recent Searches:

function showCityDropdown() {

  const query = searchInput.value.trim().toLowerCase();

  if (query.length > 0) {

    const filteredCities = recentCities.filter(city => city.toLowerCase().includes(query));

    cityDropdown.innerHTML = '';

    filteredCities.forEach(city => {

      const li = document.createElement('li');

      li.textContent = city;

      li.classList.add('p-2', 'cursor-pointer', 'hover:bg-blue-100');

      li.addEventListener('click', () => {

        searchInput.value = city;

        cityDropdown.classList.add('hidden');

        fetchWeatherByCity(city);

      });

      cityDropdown.appendChild(li);

    });

    cityDropdown.classList.toggle('hidden', filteredCities.length === 0);

  } else {

    cityDropdown.innerHTML = '';

    recentCities.forEach(city => {

      const li = document.createElement('li');

      li.textContent = city;

      li.classList.add('p-2', 'cursor-pointer', 'hover:bg-blue-500');

      li.addEventListener('click', () => {

        searchInput.value = city;

        cityDropdown.classList.add('hidden');

        fetchWeatherByCity(city);

      });

      cityDropdown.appendChild(li);

    });

    cityDropdown.classList.remove('hidden');

  }

}

* Dynamically shows a dropdown list of recent cities or filtered suggestions based on user input.
* Adds click functionality to each dropdown item to search for the selected city.
* Adds the city to the recent Cities list, limiting it to the 3 most recent cities.
* Stores the list in local storage for persistence.

3. Fetch by city name:

async function fetchWeatherByCity(city) {

  if (!city) {

    showError('Please select a city.');

    return;

  }

  try {

    const [weather, forecastData] = await Promise.all([

      fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`).then(res => res.json()),

      fetch(`https://api.openweathermap.org/data/2.5/forecast?q=${city}&appid=${apiKey}&units=metric`).then(res => res.json())

    ]);

    if (weather.cod === 200 && forecastData.cod === "200") {

      displayTodayWeather(weather);

      displayForecast(forecastData);

      addRecentCity(city);

    } else {

      showError('City not found.');

    }

  } catch {

    showError('Failed to fetch weather data.');

  }

}

* Fetches current weather and a 4-day forecast from OpenWeatherMap's API for the specified city using Promise.all.
* If successful, it displays the data using displayTodayWeather and displayForecast.

4. Fetch by coordinates:

async function fetchWeatherByCoordinates(lat, lon) {

  try {

    const [weather, forecastData] = await Promise.all([

      fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${apiKey}&units=metric`).then(res => res.json()),

      fetch(`https://api.openweathermap.org/data/2.5/forecast?lat=${lat}&lon=${lon}&appid=${apiKey}&units=metric`).then(res => res.json())

    ]);

    if (weather.cod === 200 && forecastData.cod === "200") {

      displayTodayWeather(weather);

      displayForecast(forecastData);

    } else {

      showError('Unable to fetch location-based weather.');

    }

  } catch {

    showError('Failed to fetch weather data.');

  }

}

* Fetches weather and forecast based on user coordinates (latitude, longitude).

5.Function to display weather data :

function displayTodayWeather(data) {

  const { name, main, weather, wind } = data;

  const iconUrl = `https://openweathermap.org/img/wn/${weather[0].icon}@4x.png`;

  const dayOfWeek = new Date().toLocaleString('en-us', { weekday: 'long' });

  todayWeatherContainer.innerHTML = `

    <h2 class="text-3xl font-bold text-white">${name} - ${dayOfWeek}</h2>

    <img src="${iconUrl}" alt="${weather[0].description}" class="my-4 mx-auto">

    <p class="text-white">${weather[0].description}</p>

    <p class="text-4xl font-bold text-white">${main.temp}°C</p>

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

    <p>Wind: ${wind.speed} m/s</p>

  `;

  todayWeatherContainer.classList.remove('hidden');

}

* Displays details about today’s weather, including temperature, humidity, and wind speed.

6. Function to forecast:

function displayForecast(data) {

  forecastContainer.innerHTML = '';

  const forecastList = data.list.filter((\_, index) => index % 8 === 0).slice(1, 5);

  forecastList.forEach(day => {

    const iconUrl = `https://openweathermap.org/img/wn/${day.weather[0].icon}@2x.png`;

    const dayOfWeek = new Date(day.dt \* 1000).toLocaleString('en-us', { weekday: 'long' });

    forecastContainer.innerHTML += `

      <div class="bg-blue-500 p-6 rounded-lg shadow-md flex flex-col items-center text-center text-white">

        <p class="text-xl font-semibold">${dayOfWeek}</p>

        <p>${new Date(day.dt \* 1000).toLocaleDateString()}</p>

        <img src="${iconUrl}" alt="${day.weather[0].description}" class="w-16 h-16">

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

        <p>${day.main.temp}°C</p>

        <p>Humidity: ${day.main.humidity}%</p>

        <p>Wind: ${day.wind.speed} m/s</p>

      </div>

    `;

  });

  forecastContainer.classList.remove('hidden');

}

* Displays details about today’s weather, including temperature, humidity, and wind speed.
* Filters and displays forecast data for the next 4 days.

7. Error handling:

function showError(message) {

  todayWeatherContainer.innerHTML = `<p class="text-red-500">${message}</p>`;

  todayWeatherContainer.classList.remove('hidden');

  forecastContainer.innerHTML = '';

}

// Hide dropdown if clicked outside

document.addEventListener('click', (event) => {

  if (!event.target.closest('#search-input') && !event.target.closest('#city-dropdown')) {

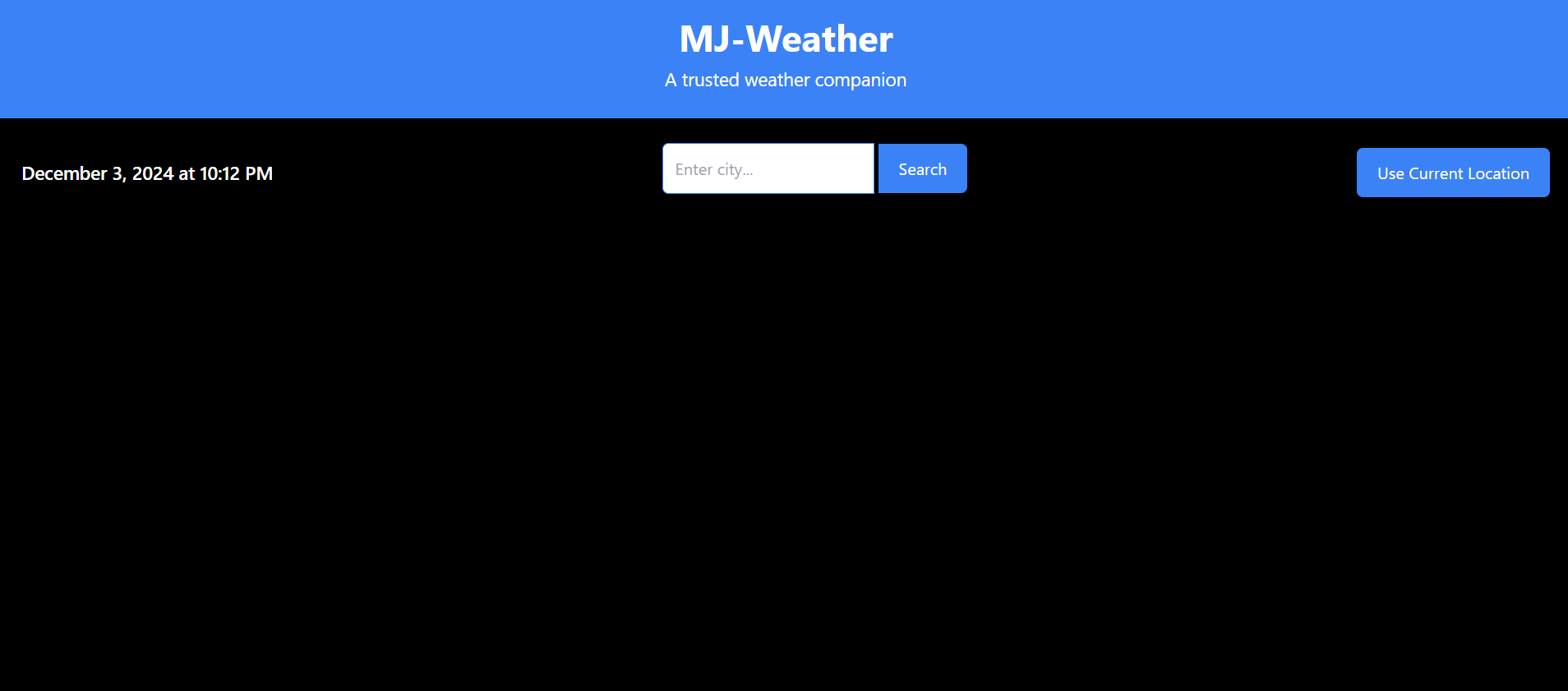
    cityDropdown.classList.add('hidden');

  }

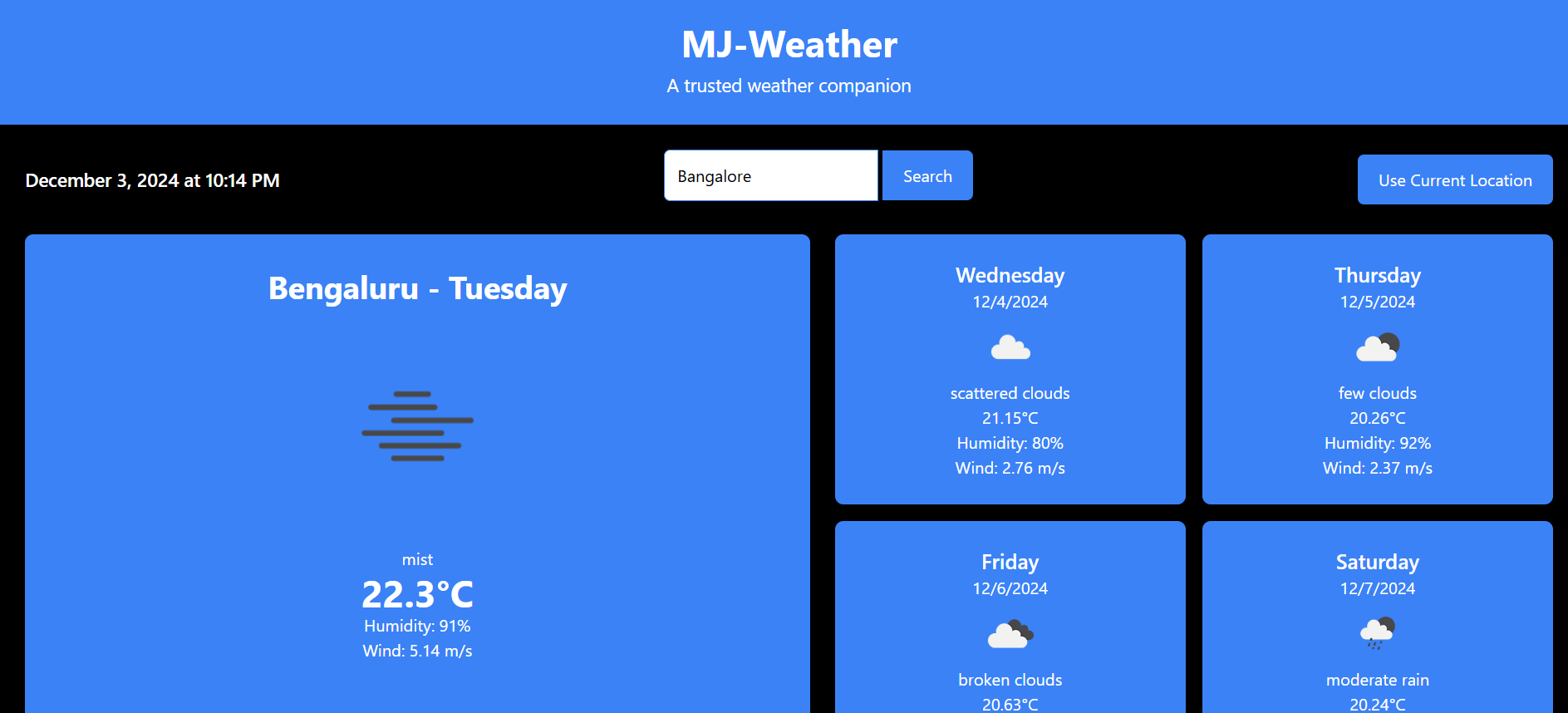
});

* Displays an error message if the city is not enter please enter city.
* Displays city not found when city is not found.

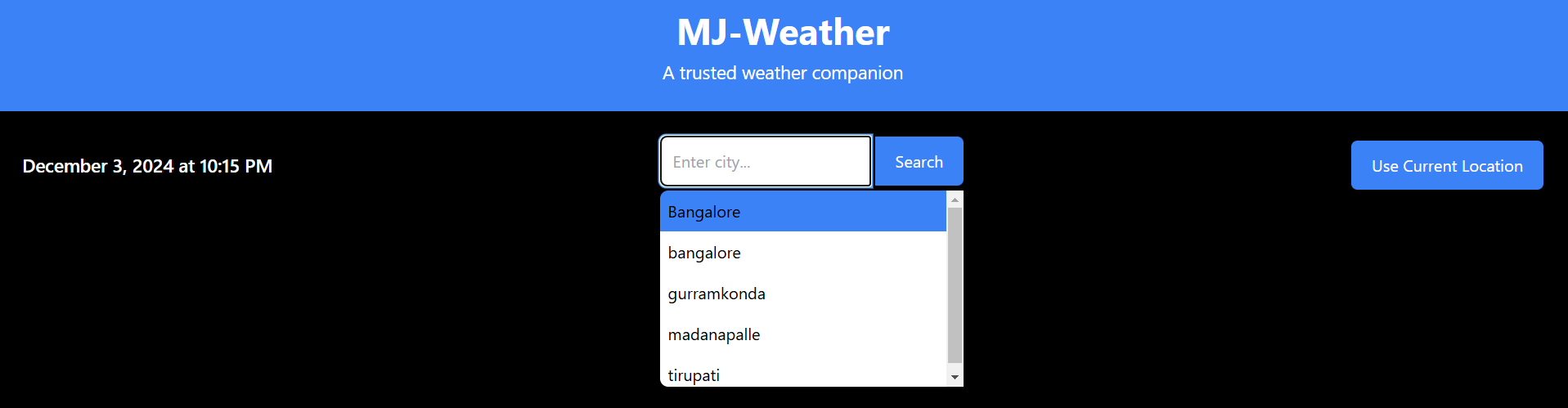
## OUTPUT:



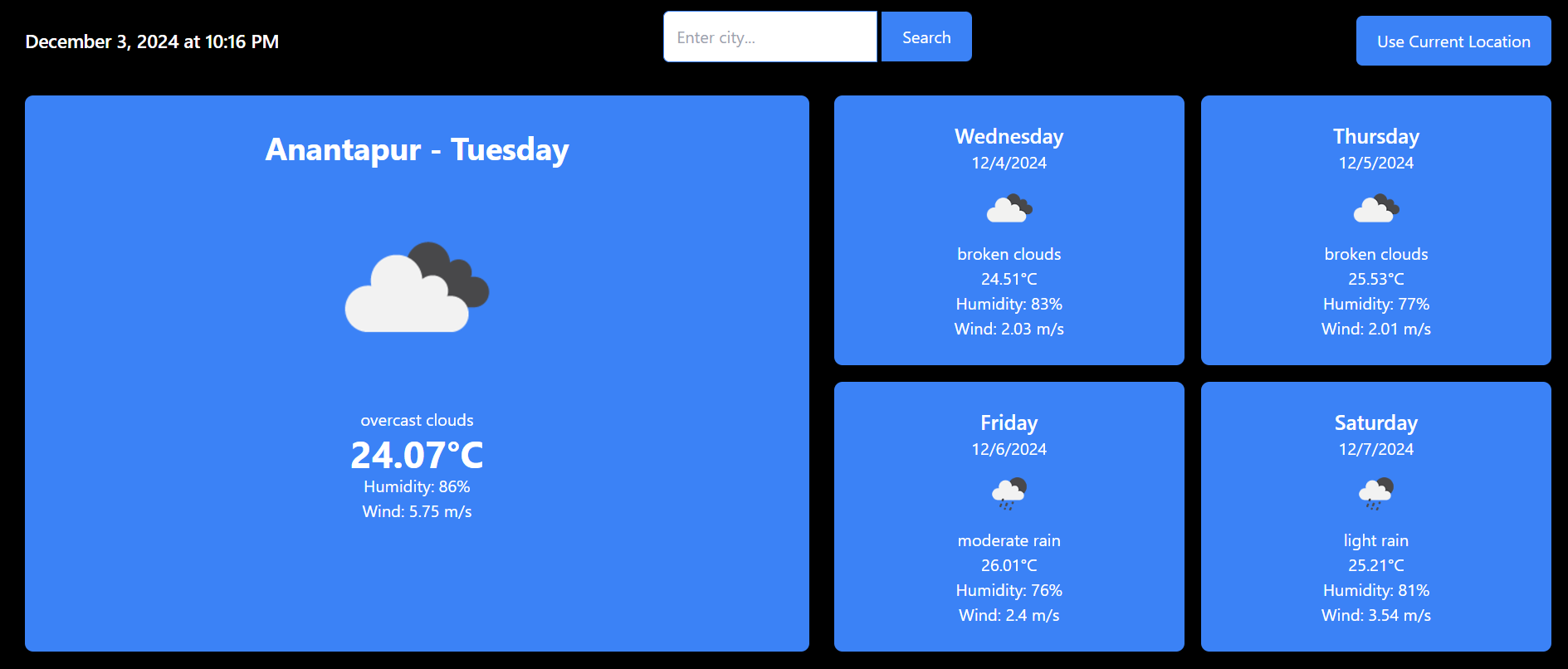
This is my home page.



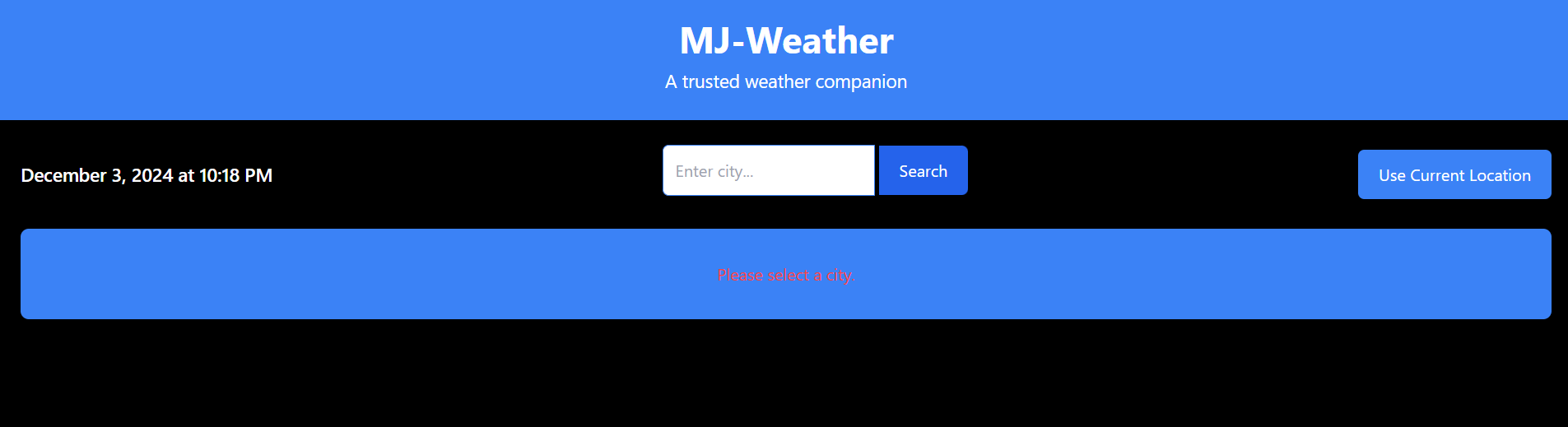
My webpage when a city is searched.

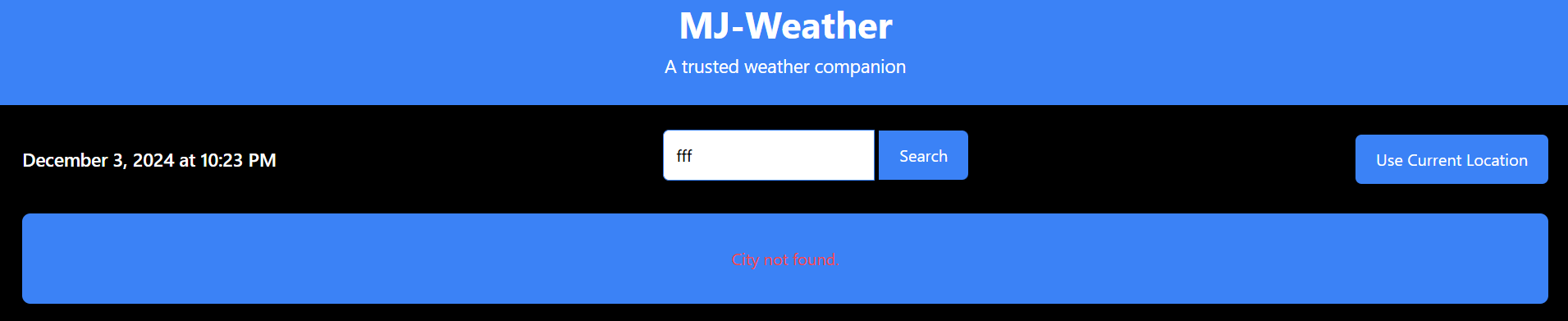


My webpage of recent searches.

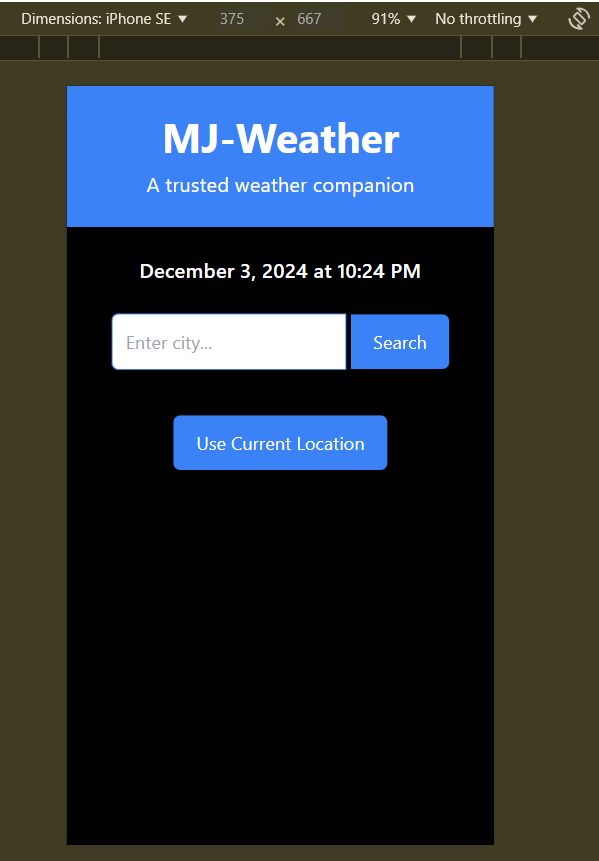


My webpage of my current location.

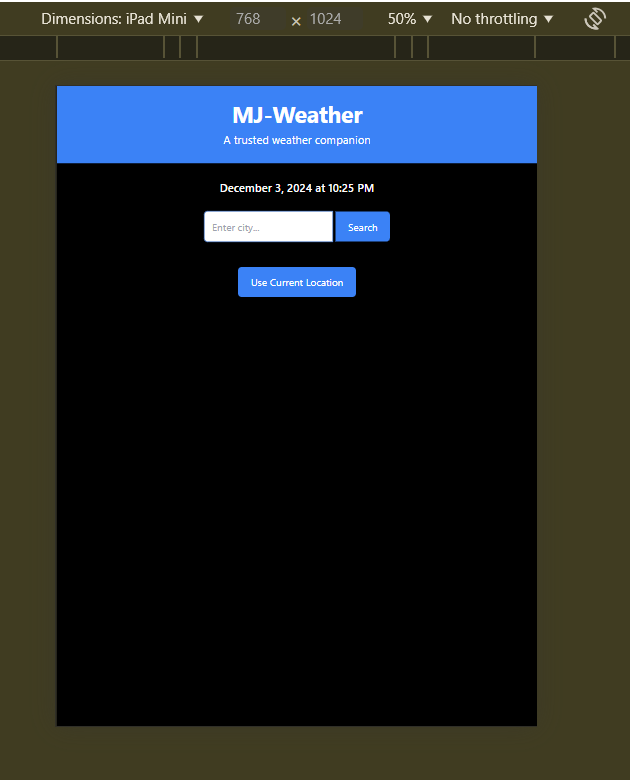
  
 My webpage when searched without selecting city.

  
 My webpage when city is not found.

Responsiveness:



My webpage on iphone SE



My webpage on ipad MINI.

MY GITHUB LINK:

<https://github.com/MohammedJunaidKhan18/Weather-Forecast-Application-.git>