

Weather App

Problem Statement

In today's digital age, accessing accurate weather information promptly is crucial for daily planning and decision-making. Existing weather apps often come with a plethora of features, some of which can be overwhelming or unnecessary for users who simply want quick access to basic weather information. There is a need for a streamlined, user-friendly weather application that provides essential weather details without unnecessary complexity.

The objective of this project is to develop a simple, intuitive weather application that displays key weather metrics such as temperature, humidity, and wind speed based on user input. By focusing on minimalistic design and ease of use, the app aims to enhance user experience by providing quick, reliable weather updates. This application uses JavaScript to handle user input, make API calls to fetch weather data, and dynamically update the interface to reflect the current weather conditions.

Techniques Used to Solve the Problem

1.HTML for Structure:

The HTML code provides the structural foundation of the weather app. It includes various sections like the search bar, error message, and weather details to organize the content logically. The use of semantic elements and descriptive classes enhances readability and accessibility.

2.CSS for Styling and Layout:

- The CSS code applies visual styling to the HTML elements to make the app aesthetically pleasing and responsive. Techniques like Flexbox are used for layout management, ensuring that elements are properly aligned and spaced. Media queries provide responsive design, adjusting the layout for different screen sizes to enhance the user experience on mobile devices.*

3.JavaScript for Interactivity and API Integration:

JavaScript is used to handle user interactions and dynamically update the app's content based on the API response.

API Integration: The code integrates with the Open Weather Map API using the Fetch API to retrieve real-time weather data. Asynchronous programming with async and await allows for non-blocking API calls, ensuring a smooth user experience

Error Handling: The JavaScript code includes error handling to manage cases where the city is not found (404 error). This is handled by displaying an error message and hiding the weather details.

Dynamic Content Updates: The weather information, such as temperature, humidity, and wind speed, is dynamically updated based on the API response. JavaScript manipulates the DOM to display the fetched data and update weather icons according to the current weather condition.

Event Listeners: The app uses event listeners to respond to user actions, like clicking the search button. This triggers the weather data fetching process, providing a seamless and interactive experience

Explanation of the Code

1. HTML Structure

A `<div class="card">` serves as the main container for the app content.

The search section allows users to enter a city name and search for its weather.

An error message `<div class="error">` is displayed if the entered city name is invalid.

The weather details section `<div class="weather">` is initially hidden and becomes visible once valid weather data is fetched.

2. CSS Styling:

Basic styling is applied to body elements, such as fonts, background image, and layout using Flexbox for centering.

card is styled to look like a weather card with a semi-transparent background, rounded corners, and a shadow.

search and its child elements are styled for a clean and functional input area.

error and .weather sections have display properties to control their visibility based on user actions and data validity.

Responsive styles adjust elements based on screen width to ensure readability and usability on all devices.

3.JavaScript Logic:

- API Key and URL:** Defined constants for the OpenWeatherMap API key and base URL.

- Element Selection:** Using `document.querySelector()` to select DOM elements for manipulation.

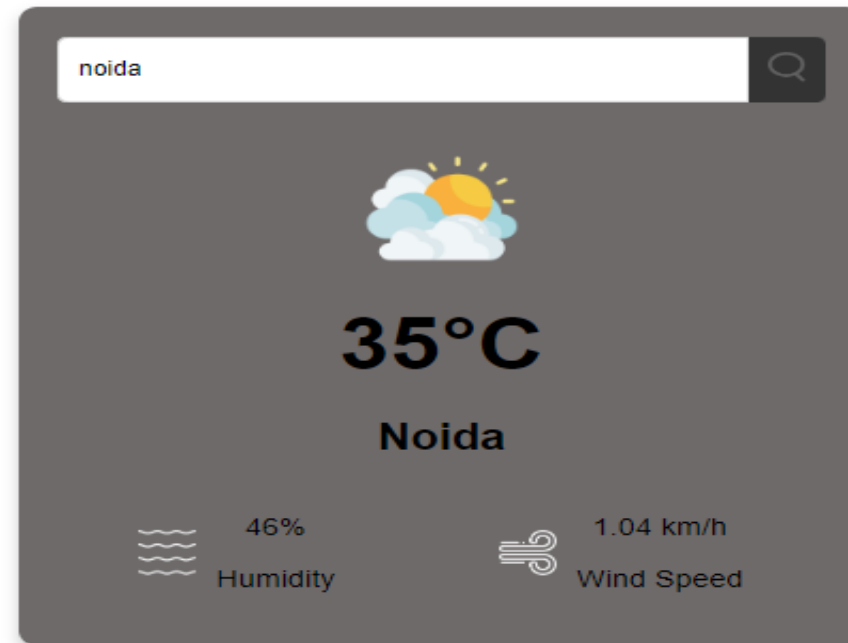
Weather Fetch Function: `checkWeather(city)` function fetches weather data using the Fetch API. It checks the response status to determine if the city name is valid and updates the DOM accordingly.

Error Handling: If the city is not found, the error message is displayed, and weather details are hidden.

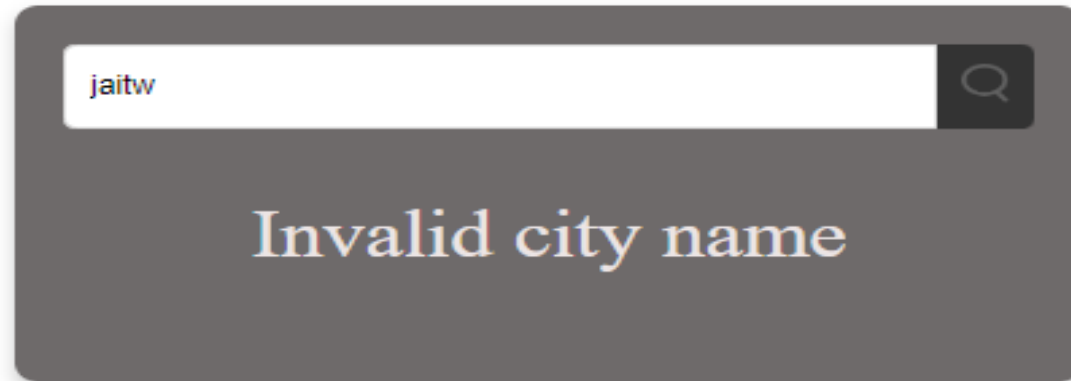
Updating Weather Details: Based on the API response, the weather details are populated in the DOM, including temperature, humidity, wind speed, and an appropriate weather icon.

Event Handling: An event listener on the search button triggers the weather data fetch when clicked.

Output with correct city name Noida



Output with wrong city name



jaitw

Invalid city name