**WEATHER WEB APP**

**DOCUMENTATION**

[**WeatherApp Link**](https://github.com/divyanshkataria16/WeatherApp)

Submitted by

|  |
| --- |
| **Divyansh Kataria**  **(2010993646)** |

1. **Project Title**

Weather Web App

1. **Abstract**

A Weather web application developed using HTML, CSS, JavaScript, and the OpenWeather API. The application provides real-time weather data including temperature, humidity, wind speed, and forecasts for users based on their location. Key features include current weather display, location-based search, interactive UI elements, and responsive design. This project showcases the effective integration of modern web technologies and APIs to deliver accurate weather information to users. Future improvements may include enhanced data visualization and user customization options.

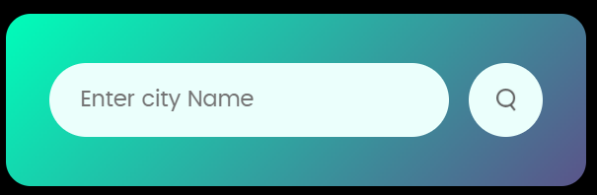
1. **Introduction**

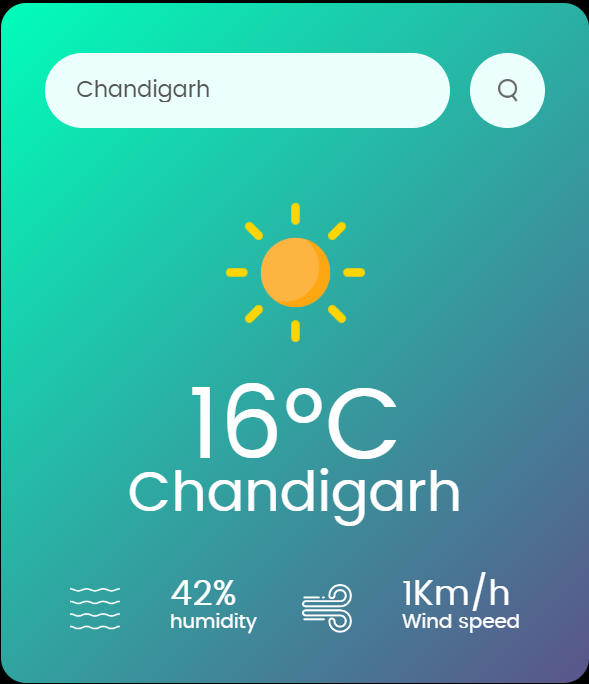
Welcome to the documentation of our Weather Web Application. This application provide user with real time weather information for the locations across the world. It is build using Html, CSS for the Frontend and integrated with the OpenWeatherMap API to fetch weather data.

1. **Functional Overview**

Functional Overview of the Weather Web Application:

1. Current Weather Display: Shows temperature, humidity, wind speed, and weather icon for the user's location.
2. Location-Based Search: Allows users to search weather information by city name or coordinates.
3. Interactive UI Elements: Includes dropdown menus, search bars, and buttons for easy navigation.
4. Responsive Design: Adapts layout and content based on device screen size and orientation.
5. **Key Features**
6. **Current Weather Display:** Provides real-time information about temperature, humidity, wind speed, and weather conditions for a specific location.
7. **Location-Based Search:** Allows users to search for weather data for specific locations by entering city names or ZIP codes.
8. **Responsive Design:** Ensures that the application adapts its layout and functionality seamlessly across different devices and screen sizes.
9. **Deploy Application**: Upload the application files to a web server or hosting platform that supports HTML, CSS, JavaScript, and server-side technologies if applicable. Ensure that the server environment meets the application's requirements.
10. **Test the Application**: Once the application is deployed, access it through a web browser to test its functionality. Enter a location to fetch weather data and verify that the application displays accurate and up-to-date weather information.

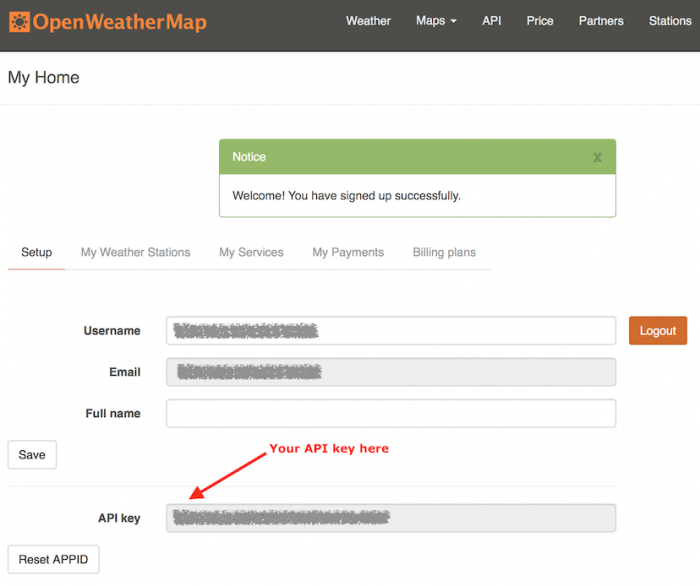
****

****

1. **Implementation**

**Install Dependencies:** Before developing our project, we made sure that all the necessary dependencies required for the node are installed. Using command **npm install**.

**Get OpenWeather API Key**: Sign up for a free account on OpenWeatherMap and generate your API key.

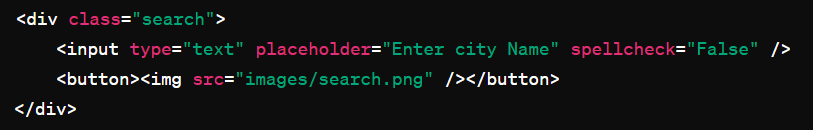


Once you have the API key, Create the file in the root directory and add the following in it

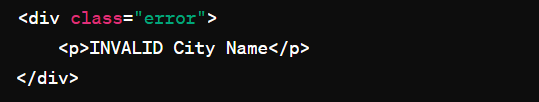
REACT\_APP\_API\_KEY=api\_key\_generated

**Configure API Key:** Replace the placeholder API key in the application's code with your actual API key.

1. **Application Functions**



The <div class="search"> section includes an input field for users to enter a city name, complemented by a search button containing an image icon sourced from "images/search.png". This interactive feature allows users to input their desired location and initiate a search for weather information.

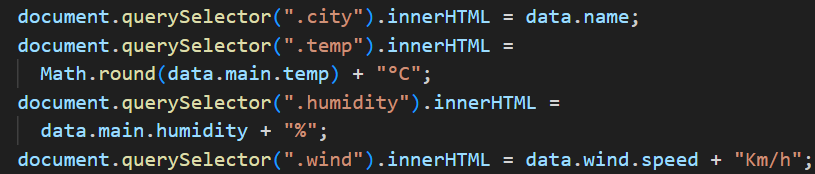


The <div class="error"> container is designed to display an error message, specifically "INVALID City Name", if the entered city name is not recognized or invalid, providing feedback to users about their input.



This line includes an external JavaScript file named "index.js" to add interactivity and functionality to the webpage.

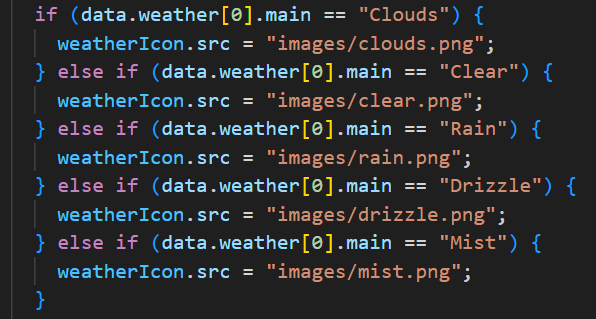
The provided functions in the "index.js" file work together to create a functional weather application that interacts with an API (e.g., OpenWeather API) to fetch weather data and update the user interface (UI) accordingly.



The Update Weather UI Function (function updateWeatherUI(data)) manipulates DOM elements to display weather-related data such as the city name, temperature, humidity, and wind speed.

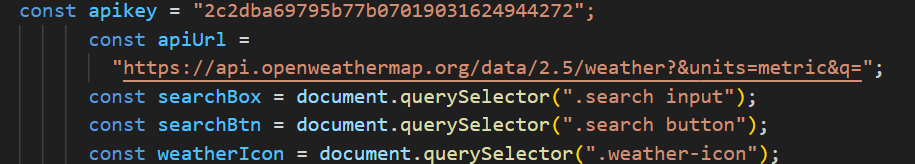


The Show Error Function (function showError(message)) handles the display of error messages on the webpage when there are issues with fetching weather data or if the entered city name is invalid.



The Update Weather Icon Function (functionupdateWeatherIcon(weatherCondition)) dynamically changes the weather icon displayed on the webpage based on the weather condition received from the API response. It uses a switch statement to set the source (src) of the weather icon image (weatherIcon) to the appropriate weather condition icon image file depending on the weather condition.

**API Used**

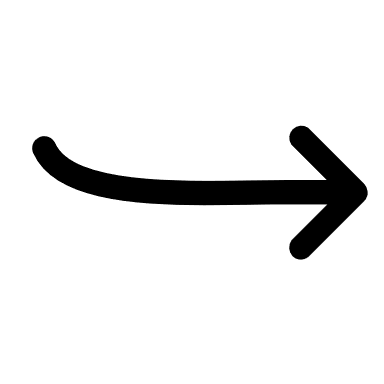


This app uses the OpenWeatherApp API to fetch weather data. It is a REST API.

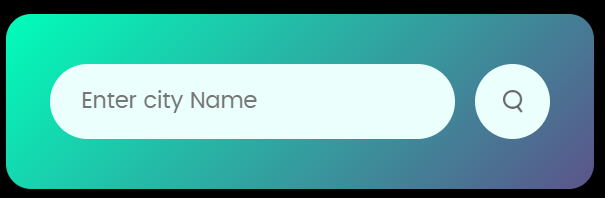
For more information, visit its official website.

**Starting the WebApp**

Clone the repository from GitHub

 <https://github.com/divyanshkataria16/WeatherApp>

The frontend application design and implementation of the project is:



1. **Test Cases**

UI Testing:

- Verify that the app interface displays correctly on different screen sizes and orientations.

- Check if all elements like buttons, icons, and text are properly aligned and visible.

Search Functionality:

-Test the search functionality by entering valid city names and checking if the weather data is displayed correctly.

- Test the accuracy of current weather data displayed.

Performance Testing:

- Test the app's loading time for weather data.

- Evaluate the app's response time when switching between different views or locations

Cross-Browser Compatibility:

- Test the app on different devices, operating systems, and browsers if applicable.

API Key Validation:

Test how the application handles invalid or missing API keys

Offline Testing:

- Verify if the app provides cached weather data when the device is offline.

- Check if the app shows the last updated weather information until a new update is available.

Security Testing:

- Ensure that the app does not expose user data or location information.

- Check if data transmission is secure and encrypted.

1. **Challenges Faced**

-API Integration: Integrating and working with external APIs, such as the OpenWeather API, can be challenging

-Asynchronous Programming: Working with asynchronous JavaScript, especially when making API calls or handling user interactions, can be complex.

-Performance Optimization: Optimizing the application's performance, such as reducing load times, minimizing network requests, caching data, and optimizing images, is crucial for a smooth user experience.

1. **References**

**OpenWeather API Documentation:**

-Website: OpenWeather API Documentation

-Access to real-time weather data, key for fetching and displaying weather information in the application.

**MDN Web Docs - JavaScript:**

-Website: MDN Web Docs - JavaScript

-Essential resource for JavaScript concepts, syntax, and best practices, utilized for handling API requests and asynchronous programming.

**CSS-Tricks**:

-Website: CSS-Tricks

-Valuable tutorials and articles on CSS, responsive design techniques, and modern front-end development practices, aiding in creating a visually appealing and responsive UI.

**Stack Overflow:**

-Website: Stack Overflow

-Community-driven platform for troubleshooting coding issues, finding solutions to challenges, and gaining insights from experienced developers, valuable for resolving technical issues encountered during development.