Weather App Project Documentation

Abstract

This project is a comprehensive weather application that enables users to obtain real-time weather data by simply inputting a city name. Using the OpenWeatherMap API, the app fetches and displays detailed weather information including temperature, humidity, wind speed, and weather conditions (e.g., sunny, cloudy, rain). The primary interface of the app is built using HTML, CSS, and JavaScript, ensuring a user-friendly and responsive design.

The application operates by allowing users to input a city name into a search field, which triggers an API request to OpenWeatherMap. The data is retrieved in JSON format and parsed to display relevant weather information. The app includes visual indicators, such as icons, to represent different weather conditions (like sun, clouds, or rain) based on the current weather in the selected city.

In addition to the web version, this project is also designed as an Android application, built using Java and XML in Android Studio. The Android version provides a similar experience, allowing users to enter a city and view real-time weather updates. This cross-platform approach enhances accessibility, allowing users to check weather conditions on both desktop and mobile devices.

This document provides an in-depth overview of the project's modules, functionality, and overall workflow, along with the code implementation. It highlights key features, such as error handling for incorrect city names, dynamic weather icon updates, and asynchronous data fetching from the API.

Modules Breakdown

1. User Interface (UI) Module:

- Responsible for rendering the input field, button, and display area for weather details.
- Uses HTML for structure, CSS for styling, and JavaScript for interactivity.

2. API Request Module:

- Handles the API key and URL setup for OpenWeatherMap.
- Makes asynchronous API requests to fetch weather data based on user input.

3. Data Processing and Display Module:

- Parses the JSON data received from the API to extract relevant details like temperature, humidity, and conditions.
- Dynamically updates the UI with this data, including selecting weather icons based on conditions (e.g., sunny, cloudy).

4. Weather Details Reset/Clear Module:

- Clears previous weather details from the UI when a new search is initiated.
- Ensures a clean interface, providing a smooth transition between searches.

5. Initialization and Event Listener Module:

- Automatically fetches weather data for a default city ('Chennai') on initial load.
- Adds an event listener to the search button to handle user input and trigger the API request.

Overall Flow

1. Initial Setup and UI Rendering:

- Loads HTML and CSS to display the interface, which includes an input field, button, and display area.

- JavaScript initializes key elements and fetches weather data for a default city (Chennai) upon load.

2. User Interaction:

- User enters a city name into the input field and clicks the search button to initiate a weather request.

3. Fetching Weather Data:

- Clears any previously displayed weather data from the UI.

- Constructs an API request URL using the OpenWeatherMap endpoint, city name, and API key.

- Fetches data asynchronously, with error handling in case of issues (e.g., city not found).

4. Processing and Displaying Data:

- Parses the JSON response to extract city name, temperature, humidity, wind speed, and weather conditions.

- Selects and displays appropriate weather icons based on current conditions.

- Updates UI elements with the fetched data for easy viewing.

5. Error Handling and Feedback:

- If the API fails or the city name is invalid, displays an error message ("City not found") to the user.

6. Repeat or End:

- Allows user to enter another city name and retrieve updated weather information.
- Repeats the process each time the user searches for a new city.