UseCaseTitle:[Whether Forecast]
RegisterNumber[20223031506130]
Institution:[Government Arts&Sciencecollege konam ,Nagerkoil]
StudentName:[Kanniammal S]
Department:[Computer Science]
DateofSubmission:[18.03.2025]

Problem Statement**

Travelers, commuters, and general users often need **real-time weather updates** to plan their activities effectively. While many weather apps exist, they can be cluttered, slow, or require subscriptions.

This project aims to create a **simple, fast, and user-friendly** web application that provides real-time weather updates by fetching data from OpenWeatherAPI.

Key Features:

- **Search for any city's weather** in real-time.

- **Display key weather details**: Temperature, humidity, and weather description.

- **Use weather icons** to visually represent conditions.

- **Ensure a lightweight and responsive UI** for a seamless experience.

- **Deploy on a free hosting platform** for easy accessibility.

Proposed Solution**

The Weather Dashboard will be a **web-based application** that allows users to check real-time weather conditions quickly.

How it Works:

- Users **enter a city name** to get weather details.

- The app **fetches real-time data** from OpenWeatherAPI.

- Weather details such as **temperature, humidity, and conditions** are displayed in a clean UI.

- Icons provide a **visual representation** of weather conditions (e.g., sun for clear weather, clouds for cloudy conditions).

- The application is **responsive** for both desktop and mobile users.

- If an **invalid city name** is entered, a user-friendly **error message** is displayed.

Technologies & Tools Considered**

Frontend Development:

- **HTML, CSS** – For structuring and styling the UI.

- **JavaScript** or **React.js** – For dynamic content and API integration.

Weather Data API:

- **OpenWeatherAPI** – Fetches real-time weather details.

Deployment Platforms:

- **Render.com, Railway.app, or Netlify** – Free-tier hosting solutions.

Solution Architecture & Workflow**

High-Level Architecture:

1. The **frontend (HTML, CSS, JS/React.js)** handles user input and displays weather data.

2. The **backend (OpenWeatherAPI)** fetches real-time weather data.

3. The **UI updates dynamically** to show results or error messages.

Workflow:

✅ **Step 1**: User enters a city name. ✅ **Step 2**: The app sends a request to OpenWeatherAPI. ✅ **Step 3**: API returns weather details in JSON format. ✅ **Step 4**: The frontend extracts and displays relevant data. ✅ **Step 5**: If the city name is invalid, an error message appears.

A simple flowchart can be created using Draw.io or Figma to illustrate this visually.

5. Feasibility & Challenges

Feasibility:

- **Simple to Implement**: Uses standard web technologies (HTML, CSS, JS).

- **Free & Open Source**: OpenWeatherAPI offers free-tier access.

- **Fast & Lightweight**: No heavy dependencies, ensuring a smooth user experience.

Expected Outcome & Impact**

Expected Benefits:

- **Fast and efficient** weather updates for travelers and daily users.

- **Mobile-friendly UI**, accessible from any browser.

- **Visually appealing weather icons** for easy interpretation.

- **User-friendly design** with simple interactions.

- **Open-source solution** that others can improve upon.

Who Benefits?

- **Travelers & Commuters** needing real-time weather updates.

- **Outdoor Enthusiasts** planning trips, hikes, or sports.

Future Enhancements**

Possible Upgrades:

- **5-day weather forecast** instead of just the current weather.

- **Multi-city comparisons** to check weather conditions in different places.

- **Unit preference storage** (allow users to switch between °C and °F).

- **Dark mode support** for better UI experience.

- **Geolocation-based weather** to automatically detect the user's location.

- **Alternative API integration** in case OpenWeatherAPI has limitations.

https://kannniammal.github.io/Kanni
ammal/