Weather App – Summary

Overview

A simple yet modern weather application built with **Vite + React + Tailwind CSS**, that fetches real-time weather data using a **public weather API**. It provides **current weather** details and a **5-day forecast**, with smooth UI transitions and error handling.

Features

- **Search Weather**: Users can search for weather in any city.
- **Upcoming Forecast**: Displays upcoming weather (e.g., next 5 days).
- **Gradient UI**: Dynamic background gradient transitions based on weather conditions (sunny, cloudy, rainy, etc.).
- **Loading & Error States**:
  - Shows a loading spinner while fetching API data.
  - Displays a friendly error message if the city is not found or API fails.
- **Responsive Design**: Works on desktop and mobile devices.

Tech Stack

- **Frontend Framework**: React (Vite for fast build & dev)
- **Styling**: Tailwind CSS with gradient transitions
- **API**: OpenWeatherMap (or similar weather API)
- **State Management**: React hooks (`useState`, `useEffect`)

Problems Faced & Solutions

1. **CORS Issues** → Solved by configuring proxy in `vite.config.js`.
2. **API Key Exposure** → Used `.env` file to store API key securely.
3. **Error Handling** → Added `try...catch` to handle invalid city inputs and API errors.
4. **Loading State Flicker** → Introduced a smooth loader before data renders.

SearchWeather Component – Detailed Explanation

1. Purpose

- Provides a **search box** where users can type a city name.
- Fetches **current weather data** from the OpenWeather API.
- Displays **temperature, weather condition, humidity, and wind speed**.
- Passes the city name to <UpcomingWeather /> for forecasts.

Key Code Snippets:

```
 const [forecast, setForecast] = useState(null);
 const [loading, setLoading] = useState(false);
 const [error, setError] = useState("");
Fetch Function:
 try {
    const res = await fetch(
     `https://api.openweathermap.org/data/2.5/forecast?q=${city}&appid=${API_KEY}&units=me
tric`
    );
    console.log("res",res);

    if (!res.ok) throw new Error("Unable to fetch forecast");
    const data = await res.json();
    console.log("upcoming_weater",data);

    setForecast(data.list.slice(0, 5)); // next 5 intervals
   } catch (err) {
    setError(err.message);
   }
 };
```

Learning Outcomes

1. **React + Vite Setup**
    - Learned how to create a modern React app using **Vite** for faster builds and development.
2. **Tailwind CSS Styling**
    - Applied **utility-first CSS** for responsive layouts and animated gradient backgrounds.
3. **State Management in React**
    - Used useState and useEffect to handle user input, API calls, and conditional rendering.
4. **Fetching Data from APIs**

    - Integrated **OpenWeather API**, handled `fetch`, parsed JSON, and displayed real-time weather data.
5. **Error & Loading Handling**
    - Implemented error messages for invalid cities, network issues, and API limits.
    - Added **loading states** for better UX.
6. **Component-Based Architecture**
    - Built reusable components (`SearchWeather`, `UpcomingWeather`) for modular development.
7. **Responsive UI Design**
    - Created layouts with **grids and flexbox** using Tailwind, ensuring usability on desktop and mobile.
8. **Troubleshooting Skills**
    - Faced issues like API key exposure, wrong city input, and unit conversion.
    - Learned to debug with `console.log`, proper error handling, and environment variables.