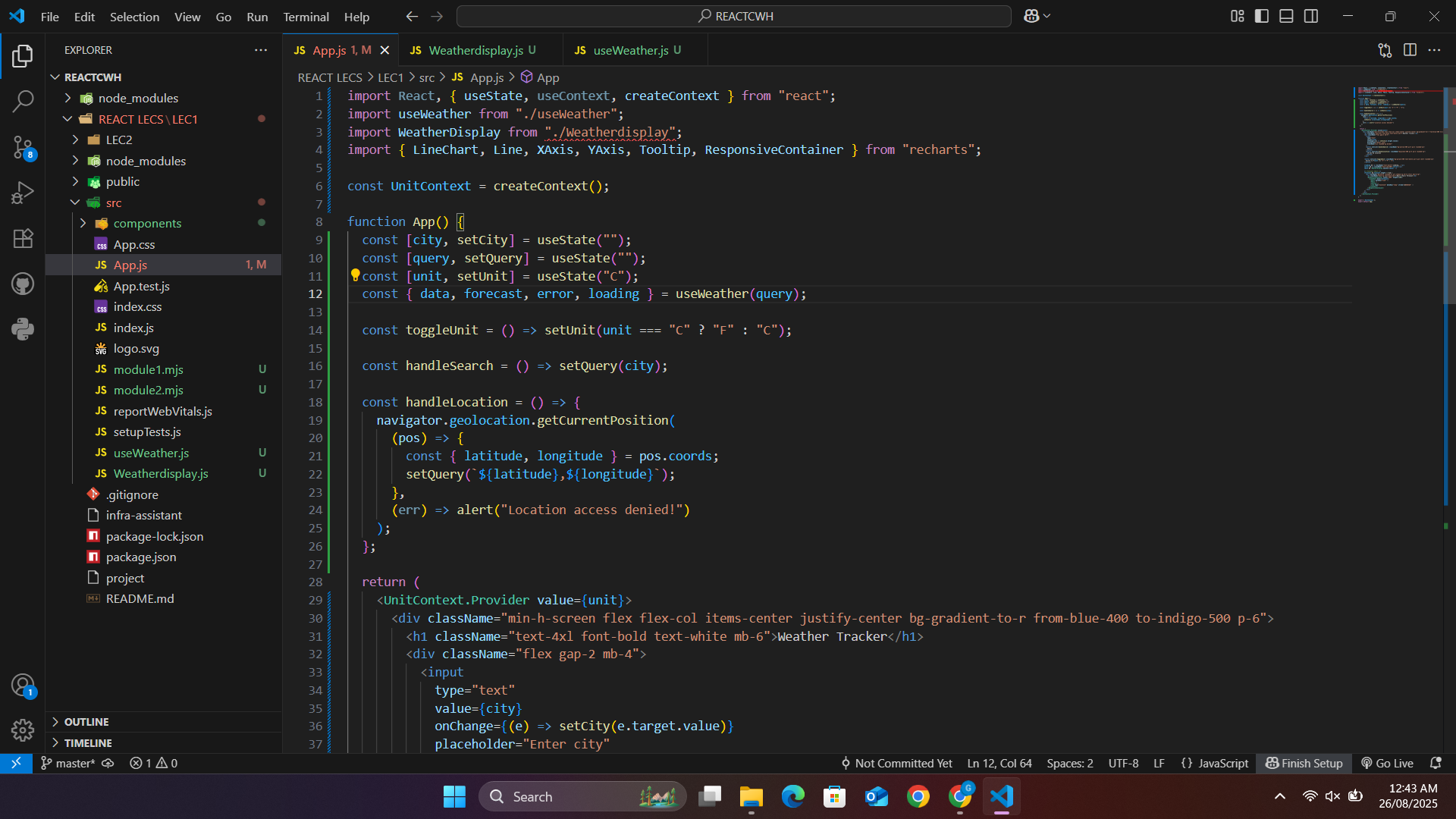
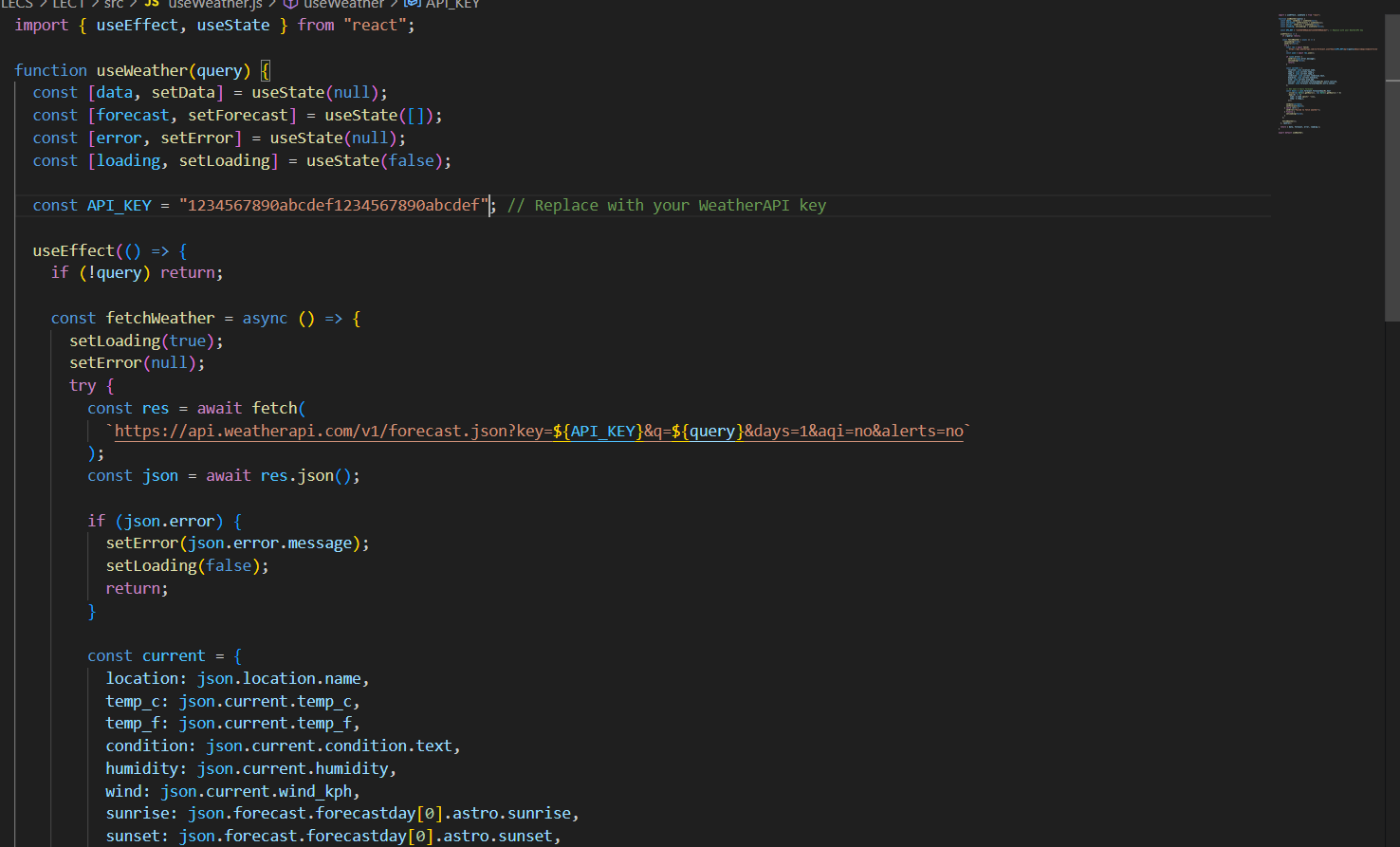
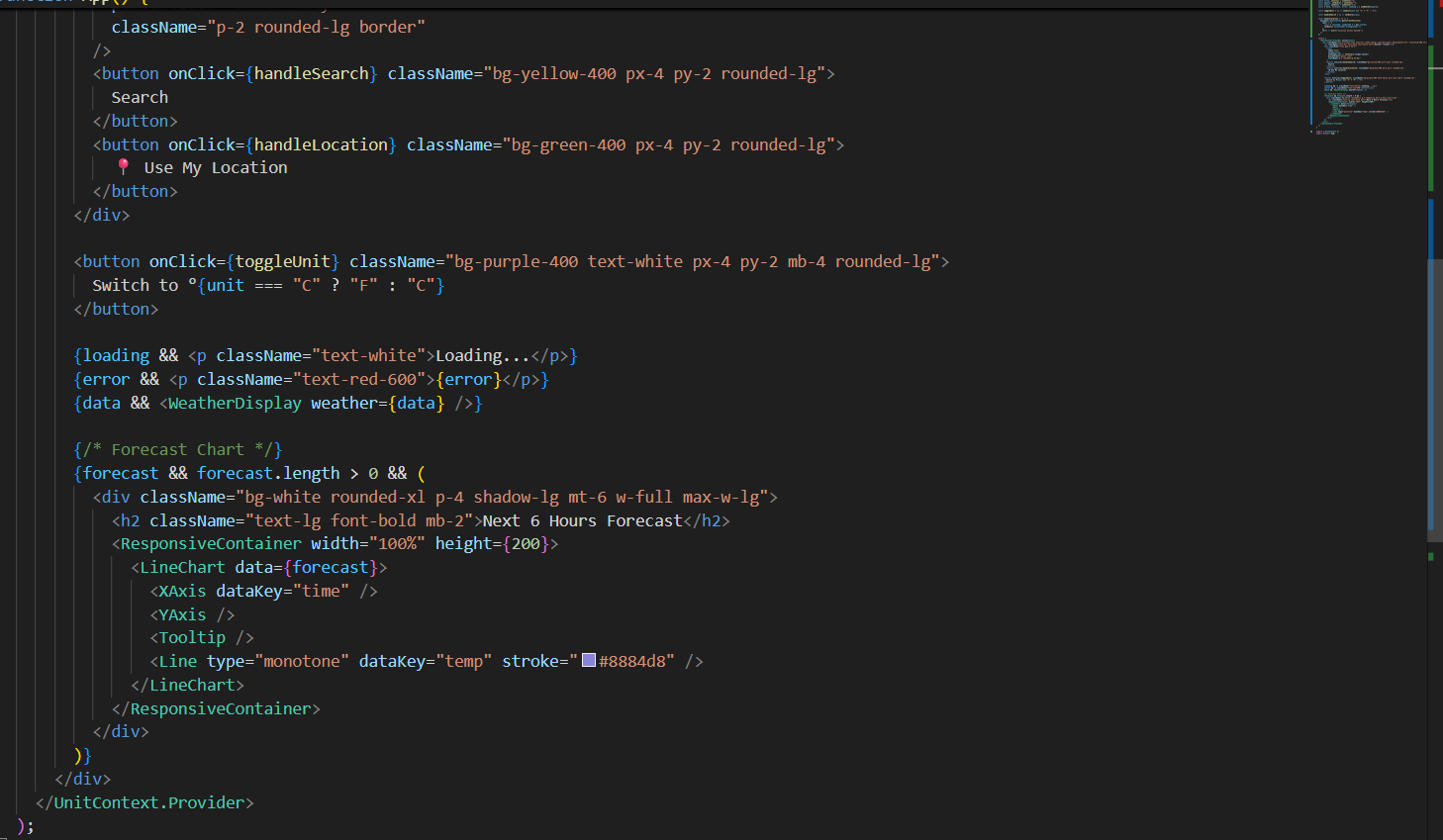
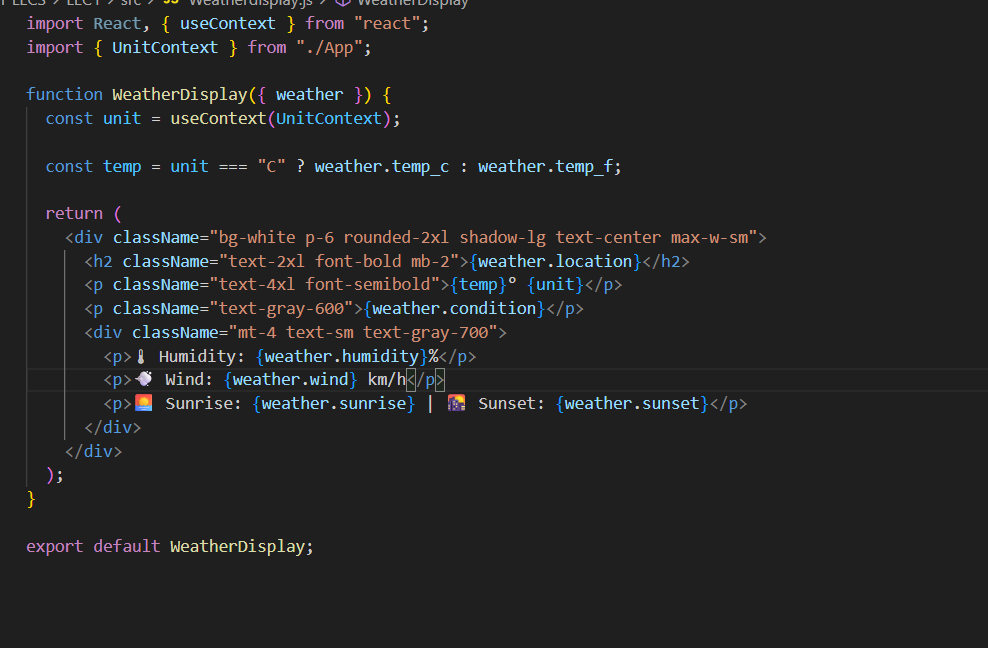
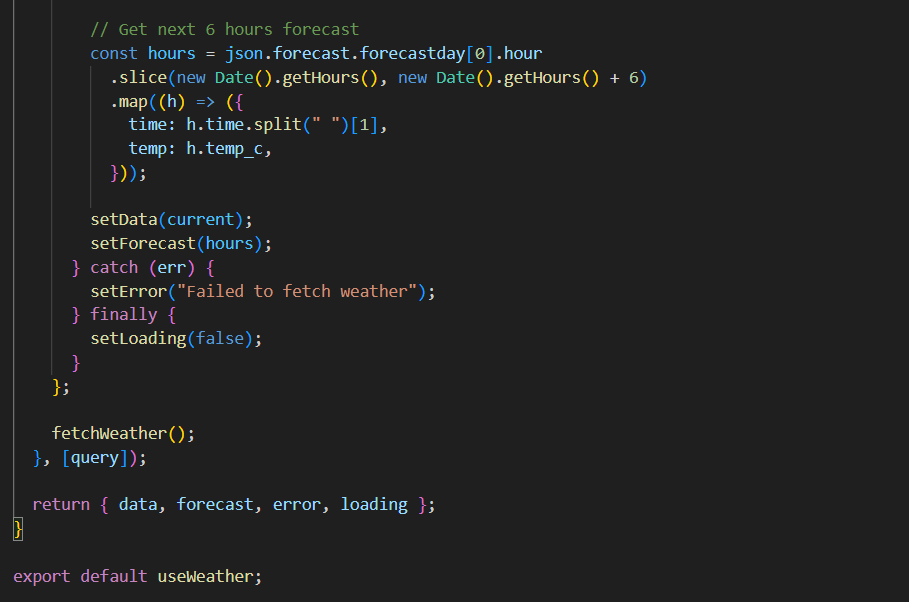
**EXPERIMENT-2**

**SOURCE CODE :**

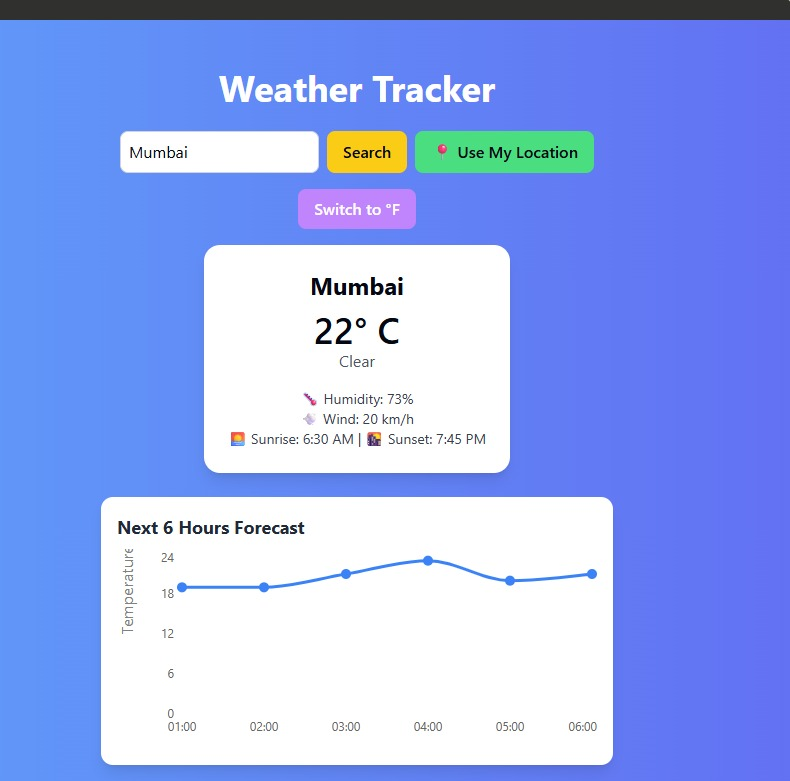
   1)App.js

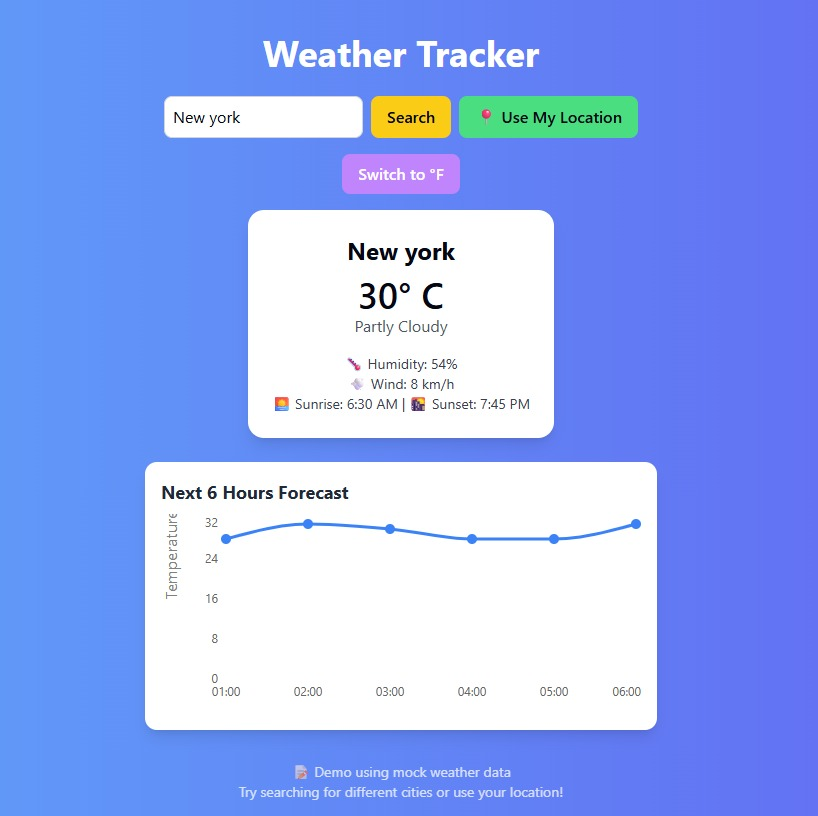
2)useweather.js  


3)Weatherdisplay.js

Output:

Output



**Output:**

* The app allows users to enter a city name and fetch weather data.
* It supports dark/light theme toggle using useContext.
* It uses a custom hook (useWeather) to fetch data and manage state.
* Weather data such as temperature, condition, humidity, and wind speed is displayed.

## **Additional Features Implemented (30% Extra)**

1. Location Detection (Geolocation API)  
   * Automatically detects the user’s current location and displays local weather.
   * Improves usability without requiring manual input every time.
2. Unit Conversion (°C ⇆ °F)  
   * Added a toggle button to switch between Celsius and Fahrenheit.
   * Gives flexibility to users based on preference.
3. Forecast Visualization  
   * Integrated a chart (using Recharts library) to display upcoming temperature trends.
   * Provides a clear visual understanding of how weather will change in the next few hours.
4. Sunrise & Sunset Timings  
   * Displays additional insights like sunrise and sunset.
   * Makes the weather report more detailed and informative.

### 

### **Conclusion:**

The React application successfully demonstrates the practical use of useEffect, useContext, and custom hooks for efficient state and side-effect management. By integrating these hooks, the app is able to fetch and display real-time weather data, manage global theme preferences without prop drilling, and encapsulate reusable logic in a custom hook. The experiment highlights how React Hooks simplify development, improve code reusability, and enhance user experience by providing features like weather data visualization and dark/light theme support.