FA-2

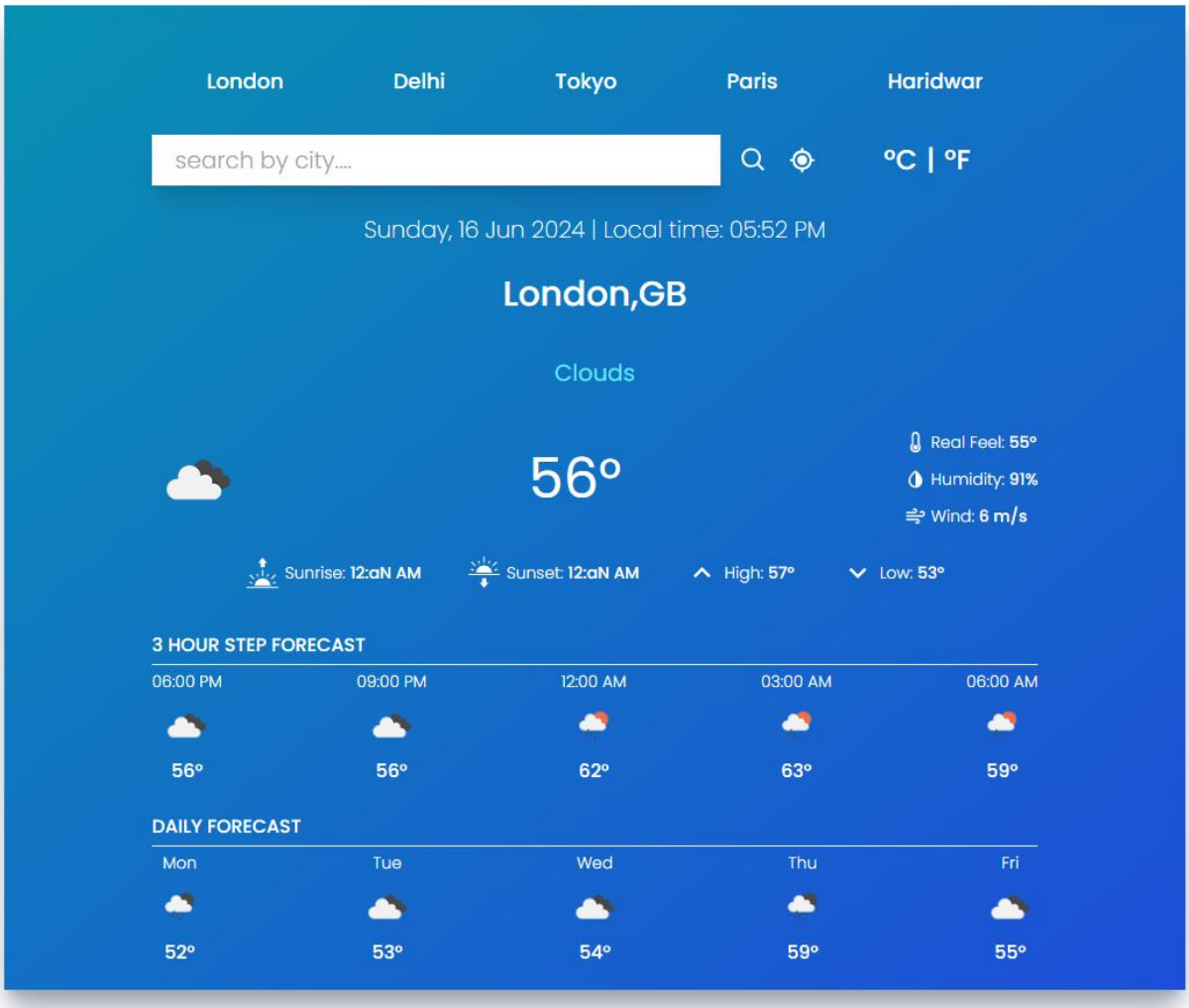
Topic :weather forecast

Roll no :39

Name rawat vijay

compb

Image:



Code of app.jsx:

```
import React, { useEffect, useState } from "react";
import TopButtons from "./components/TopButtons";
import Inputs from "./components/Inputs";
```

```
import TimeAndLocation from "./components/TimeAndLocation";
import TempAndDetails from "./components/TempAndDetails";
import Forecast from "./components/Forcast";
import getFormattedWeatherData from "./services/weatherservice";

const App = () => {
  const [query, setQuery] = useState({ q: 'Ahmedabad' });
  const [units, setUnits] = useState('metric');
  const [weather, setWeather] = useState(null);

  useEffect(() => {
    const getWeather = async () => {
      try {
        const data = await getFormattedWeatherData({ ...query, units });
        setWeather(data);
        console.log(data);
      } catch (error) {
        console.error('Error fetching weather data:', error);
      }
    };

    getWeather();
  }, [query, units]);

  const formatBackground = () => {
    if (!weather) return "from-cyan-600 to-blue-700";
    const threshold = units === 'metric' ? 20 : 60;
    if (weather.temp <= threshold) return 'from-cyan-600 to-blue-700';
    return 'from-yellow-600 to-orange-700';
  };

  return (
    <div className={`mx-auto max-w-screen-lg mt-4 py-5 px-32 bg-gradient-to-br
shadow-xl shadow-gray-400 ${formatBackground()}`}>
      <TopButtons setQuery={setQuery} />
      <Inputs setQuery={setQuery} setUnits={setUnits} />
      {weather && (
        <>
          <TimeAndLocation weather={weather} />
          <TempAndDetails weather={weather} units={units} />
          <Forecast title="3 Hour Step Forecast" data={weather.hourly} />
          <Forecast title="Daily Forecast" data={weather.daily} />
        </>
```

```
      )}
    </div>
  );
};

export default App;
```

Description: Still working on the project cuz its fetching but displaying wrong date and time,, The Weather Application is a modern, responsive web application designed to provide users with current weather conditions, detailed weather information, and forecasts for various locations. This project leverages React for building the user interface and integrates with a weather API service to fetch and display real-time weather data.

Key Features

1. **Real-Time Weather Data**: The application fetches current weather data for the user's specified location using a weather API.
2. **Location-Based Weather Updates**: Users can search for weather information by city name, ensuring they receive relevant data for their area of interest.
3. **Temperature Units**: Users have the option to toggle between metric (Celsius) and imperial (Fahrenheit) units for temperature display.
4. **Dynamic Backgrounds**: The application's background changes dynamically based on the current temperature, enhancing the user experience with visual feedback.
5. **Detailed Weather Information**: Displays comprehensive weather details including temperature, humidity, wind speed, and more.
6. **Hourly and Daily Forecasts**: Provides forecasts with detailed weather predictions for the upcoming hours and days.

Project Structure

The application is structured into several components, each responsible for a specific part of the UI or functionality:

7. **App Component**: This is the root component that manages the overall state and layout of the application.
8. **TopButtons Component**: Provides quick access buttons for pre-defined cities, allowing users to quickly fetch weather data for these locations.
9. **Inputs Component**: Contains input fields for users to enter city names and toggle between temperature units.
10. **TimeAndLocation Component**: Displays the current date, time, and location information based on the fetched weather data.
11. **TempAndDetails Component**: Shows detailed weather information such as current temperature, humidity, wind speed, etc.
12. **Forecast Component**: Used twice to show both the hourly and daily weather forecasts.

Technical Details

- **React**: Utilized for building the user interface due to its component-based architecture, which allows for reusable and manageable code.
- **State Management**: The `useState` hook manages the state of the query (location) and units (metric/imperial). The `useEffect` hook is used to fetch weather data whenever these states change.
- **Weather Service**: A custom service (`getFormattedWeatherData`) fetches data from a weather API, processes it, and formats it for use in the application.
- **Conditional Rendering**: Components are conditionally rendered based on the availability of weather data to ensure a smooth user experience.
- **Dynamic Styling**: The background color of the application changes based on the current temperature to provide a visual indication of the weather.