Code of the project :-

WeatherApp.js :-

import "./Design.css";

import React, { useState } from 'react';

import SearchImg from "./images/search.png"

import ClearImg from "./images/clear.png"

import CloudsImg from "./images/clouds.png"

import DrizzleImg from "./images/drizzle.png"

import MistImg from "./images/mist.png"

import RainImg from "./images/rain.png"

import SnowImg from "./images/snow.png"

import ThunderstromImg from "./images/thunderstorm.png"

import SmokeImg from "./images/ash.png";

import HazeImg from "./images/fog.png"

import FogImg from "./images/fog.png";

import DustImg from "./images/sand.png";

import SandImg from "./images/sand.png";

import AshImg from "./images/ash.png";

import SquallImg from "./images/squall.png";

import TornadoImg from "./images/tornado.png"

import HumidityImg from "./images/humidity.png"

import WindImg from "./images/wind.png"

const WeatherApp = () => {

const apiKey = "c45e1ad83d9690f5a8e314eac1ff6b84";

const apiUrl = "https://api.openweathermap.org/data/2.5/weather?units=metric&q=";

// https:

const [place, setPlace] = useState("");

const [weatherData, setWeatherData] = useState(null);

const [error, setError] = useState(null);

const handleChange = (event) => {

setPlace(event.target.value);

};

const checkWeather = async () => {

try {

setError(null);

const response = await fetch(apiUrl + place + `&appid=${apiKey}`);

if (response.status === 404) {

setError("Invalid place name!");

setWeatherData(null);

} else {

const data = await response.json();

setWeatherData(data);

}

} catch (error) {

setError("Error fetching data!");

setWeatherData(null);

}

};

const weatherIcon = () => {

if (weatherData && weatherData.weather && weatherData.weather.length > 0) {

const weatherMain = weatherData.weather[0].main;

switch (weatherMain) {

case "Clouds":

return CloudsImg;

case "Clear":

return ClearImg;

case "Rain":

return RainImg;

case "Drizzle":

return DrizzleImg;

case "Mist":

return MistImg;

case "Snow":

return SnowImg;

case "Thunderstrom":

return ThunderstromImg;

case "Smoke":

return SmokeImg;

case "Haze":

return HazeImg;

case "Dust":

return DustImg;

case "Fog":

return FogImg;

case "Sand":

return SandImg;

case "Ash":

return AshImg;

case "Squall":

return SquallImg;

case "Tornado":

return TornadoImg;

default :

return null

}

}

return null;

};

return (

<div className="card">

<div className="search">

<input

type="text"

placeholder="enter place name"

spellCheck="false"

value={place}

onChange={handleChange}

/>

<button onClick={checkWeather}>

<img src={SearchImg} alt="Search" />

</button>

</div>

{error && <div className="error"><h6>`</h6><p>Error: {error}</p></div>}

{weatherData && (

<div className="weather">

<img src={weatherIcon()} className="weather-icon" alt='' />

<h1 className="temp">{Math.round(weatherData.main.temp)}°c</h1>

<h2 className="place">{weatherData.name}</h2>

<div className="details">

<div className="col">

<img src={HumidityImg} alt="Humidity Icon" />

<div>

<p className="humidity">{weatherData.main.humidity}%</p>

<p>Humidity</p>

</div>

</div>

<div className="col">

<img src={WindImg} alt="Wind Icon" />

<div>

<p className="wind">{weatherData.wind.speed} km/h</p>

<p>Wind Speed</p>

</div>

</div>

</div>

</div>

)}

</div>

);

};

export default WeatherApp

App.js :-

import WeatherApp from "./components/WeatherApp";

function App() {

return (

<div>

<WeatherApp/>

</div>

);

}

export default App;

<end\_of\_code>

***Questions :-***

***1. What is the project?***

***2. How was this project done?***

***3. What are the features?***

***4. What are the technologies used?***

***5. What concepts were used ?***

Ans)

1. \*\*Project Description\*\*:

The project is a weather forecast app built using ReactJS. It allows users to enter the name of a place, and based on the input, it fetches and displays the current weather data for that location, including temperature, humidity, and wind speed.

2. \*\*Project Implementation\*\*:

The project is implemented using ReactJS, a popular JavaScript library for building user interfaces. It fetches weather data from the OpenWeatherMap API using the user-entered place name and updates the UI with the received data.

Code Explanation :-

\*\*WeatherApp.js\*\*:

1. Importing CSS and Images:

- The code starts by importing CSS styles and images for various weather conditions (e.g., clouds, rain, snow) using `import` statements.

2. Functional Component:

- The `WeatherApp` is defined as a functional component using the `const` keyword and an arrow function.

3. State and API Key:

- The `apiKey` and `apiUrl` variables are declared using `const`. These variables store the API key for OpenWeatherMap and the API URL with the desired units (metric) for temperature.

4. State Hooks:

- The `useState` hook is used to define three state variables:

- `place`: Stores the user-entered place name.

- `weatherData`: Stores the weather data received from the API.

- `error`: Stores any error message that may occur during API calls or input validation.

5. Handling User Input:

- The `handleChange` function is called when the user enters text in the input field. It updates the `place` state with the entered value.

6. Fetching Weather Data:

- The `checkWeather` function is called when the user clicks the search button. It makes an asynchronous API call to OpenWeatherMap using the `fetch` function.

- If the API call is successful (status code 200), the weather data is extracted from the response using `await response.json()`, and `weatherData` state is updated.

- If the API call returns a 404 status code, it means the place name is invalid, and an error message is set in the `error` state.

- If there is an error during the API call, the catch block sets an error message in the `error` state.

7. Weather Icon:

- The `weatherIcon` function is defined to determine the weather icon based on the weather condition received from `weatherData`. It uses a switch statement to map the weather condition to the appropriate image.

8. JSX Rendering:

- The return statement contains JSX that defines the layout of the weather forecast app.

- It includes an input field where users can enter the place name and a search button to trigger the weather data fetch.

- If there is an error (non-null `error` state), an error message is displayed.

- If there is weather data (non-null `weatherData` state), the weather information is displayed, including temperature, place name, weather icon, humidity percentage, and wind speed.

\*\*App.js\*\*:

1. Importing WeatherApp:

- The code starts by importing the `WeatherApp` component.

2. Functional Component:

- The `App` component is defined as a functional component using the `const` keyword and an arrow function.

3. JSX Rendering:

- The return statement includes the `WeatherApp` component, rendering the weather forecast app on the main page.

Overall, the Weather Forecast App uses React's functional components and hooks (specifically, the `useState` hook) to manage state and user input. It fetches weather data from the OpenWeatherMap API based on the user-entered place name and updates the UI dynamically. The app displays weather information along with appropriate weather icons for different weather conditions, providing a simple and intuitive weather forecasting experience.

<end\_of\_code\_explanation>

3. \*\*Features\*\*:

- User Input: Users can enter the name of a place in the input field to check the weather forecast for that location.

- Weather Display: The app displays the current temperature in Celsius, the place name, and weather icons representing the weather condition.

- Weather Icons: The app dynamically selects weather icons based on the weather condition, such as cloudy, clear, rainy, drizzle, mist, snow, thunderstorm, etc.

- Humidity and Wind Speed: The app also shows the humidity percentage and wind speed in kilometers per hour for the selected location.

- Error Handling: The app handles errors gracefully, such as invalid place names or failed API calls, and displays appropriate error messages to the user.

4. \*\*Technologies Used\*\*:

- ReactJS: The core library used to build the user interface and manage state.

- HTML/CSS: For structuring and styling the app components and elements.

- OpenWeatherMap API: To fetch weather data for the specified location.

5. \*\*Concepts Used\*\*:

- React State: The app utilizes React's state to manage the user input, weather data, and error handling.

- Asynchronous JavaScript: As API calls are asynchronous, the app uses `async/await` to fetch weather data from the API.

- Conditional Rendering: The app uses conditional rendering to display weather data and error messages only when they are available.

- Handling Events: The app handles user input using `onChange` and `onClick` event handlers on the input field and the search button, respectively.

- Dynamic Weather Icons: The app maps the weather condition received from the API to specific weather icons, allowing dynamic display of icons based on weather conditions.

Overall, the weather forecast app is a simple and user-friendly ReactJS application that provides real-time weather information based on the user's input. It demonstrates the use of API integration, state management, and conditional rendering in a React project.