**Weather Forecast Web Application**

**Code -:**

**App.jsx**

import { useState } from "react";

import CityAndTime from "./components/CityAndTime";

import NavBar from "./components/NavBar";

import { ToastContainer } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

const App = () => {

const [cityName, setCityName] = useState('');

const [lat, setLat] = useState(null);

const [lon, setLon] = useState(null);

const handleCitySearch = (city) => {

setCityName(city);

setLat(null);

setLon(null);

};

const handleLocationFetch = (latitude, longitude) => {

setLat(latitude);

setLon(longitude);

setCityName('');

};

return (

<div className="container mx-auto">

<ToastContainer />

<div className="w-full h-full">

<NavBar onCitySearch={handleCitySearch} onLocationFetch={handleLocationFetch} />

</div>

<div>

<CityAndTime cityName={cityName} lat={lat} lon={lon} setLat={setLat} setLon={setLon} />

</div>

</div>

);

};

export default App;

**NavBar.jsx**

import logo from '../assets/LogoChristmas.png';

import search from '../assets/search.png';

import location from '../assets/locationicon.png';

import { useState } from 'react';

import { toast } from 'react-toastify';

const NavBar = ({ onCitySearch, onLocationFetch }) => {

const [searchQuery, setSearchQuery] = useState('');

const handleSearchQuery = (e) => setSearchQuery(e.target.value);

const handleSearchSubmit = (e) => {

e.preventDefault();

if (searchQuery) {

onCitySearch(searchQuery);

setSearchQuery('');

}

};

const handleLocationClick = () => {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition((pos) => {

const { latitude, longitude } = pos.coords;

onLocationFetch(latitude, longitude);

setSearchQuery('');

}, (error) => {

console.log(error);

toast.error("Geolocation failed");

});

}

};

return (

<div className="m-4">

<div className="flex flex-col items-center justify-between gap-4 lg:flex-row">

<img src={logo} alt="logo" className="w-48 select-none" />

<form onSubmit={handleSearchSubmit} className="relative flex items-center w-full max-w-md bg-white rounded-lg shadow-md">

<img src={search} alt="search" className="absolute left-3 w-4 h-4 select-none" />

<input

type="text"

value={searchQuery}

onChange={handleSearchQuery}

placeholder="Search city..."

className="w-full py-2 pl-10 pr-4 text-sm text-gray-700 placeholder-gray-400 border-none rounded-lg outline-none"

/>

<button type="submit" className="bg-[#05e1fde] text-white px-5 py-2">Search</button>

</form>

<div onClick={handleLocationClick} className="flex items-center gap-3 px-4 text-sm font-medium text-white bg-green-500 rounded cursor-pointer">

<img src={location} alt="location" />

<p>Current Location</p>

</div>

</div>

</div>

);

};

export default NavBar;

**Clock.jsx**

import { useEffect, useState } from "react";

const Clock = () => {

const [currentTime, setCurrentTime] = useState(new Date());

useEffect(() => {

const timer = setInterval(() => {

setCurrentTime(new Date());

}, 1000);

return () => clearInterval(timer);

}, []);

return (

<div className="flex flex-col items-center">

<h1 className="text-5xl md:text-7xl font-bold">{currentTime.toLocaleTimeString()}</h1>

<p className="text-sm md:text-md font-medium">{currentTime.toLocaleDateString()}</p>

</div>

);

};

export default Clock;

**ForeCast.jsx**

/\* const ForeCast = ({ forecast }) => {

const dailyForecast = forecast.reduce((acc, item) => {

const date = new Date(item.dt \* 1000).toLocaleDateString();

if (!acc.find(f => f.date === date)) {

acc.push({

temperature: `${item.main.temp}°C`,

day: new Date(item.dt \* 1000).toLocaleDateString("en-EN", { weekday: 'short' }),

date,

icon: `https://openweathermap.org/img/wn/${item.weather[0].icon}@2x.png`

});

}

return acc;

}, []).slice(0, 5);

const hourlyForeCast = forecast.slice(0, 5).map(item => ({

time: new Date(item.dt \* 1000).toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' }),

icon: `https://openweathermap.org/img/wn/${item.weather[0].icon}@2x.png`,

degree: `${item.main.temp}°C`,

windSpeed: `${item.wind.speed} km/h`

}));

return (

<div className="flex">

<div className="xl:w-96 w-full mt-4">

<h2 className="text-2xl font-bold text-white">5 Days Forecast</h2>

<div className="flex flex-row justify-between items-center p-1">

{dailyForecast.map((cast, index) => (

<div key={index} className="text-center text-white">

<p>{cast.day}</p>

<p className="font-bold text-xl">{cast.temperature}</p>

<p>{cast.date}</p>

</div>

))}

</div>

<div className="bg-[#05aefc] shadow-2xl mt-4 mx-10 rounded-lg text-white">

<h1 className="text-2xl font-bold text-center">Hourly Forecast</h1>

<div className="flex flex-row items-center justify-center gap-5 h-54 mt-4">

{hourlyForeCast.map((hourcast, index) => (

<div key={index} className="flex flex-col items-center gap-5 bg-[#f2c938] rounded-xl p-4 w-28 text-center shadow-md text-black">

<p>{hourcast.time}</p>

<img src={hourcast.icon} alt="hourcastIcon" className="w-16 h-16 select-none" />

<p>{hourcast.degree}</p>

<p>{hourcast.windSpeed}</p>

</div>

))}

</div>

</div>

</div>

</div>

);

};

export default ForeCast;

**CityAndTime.jsx**

import { useEffect, useState } from 'react';

import { toast } from 'react-toastify';

import axios from 'axios';

import sunImg from '../assets/sun.png';

import sunrise from '../assets/sunrise-white.png';

import sunset from '../assets/sunset-white.png';

import humidity from '../assets/humidity.png';

import pressure from '../assets/pressure.png';

import uvIcon from '../assets/uv.png';

import windIcon from '../assets/wind.png';

import Clock from './Clock';

import ForeCast from './ForeCast';

/\* --------------------------------------------------

CONFIG: Put your API key in an .env file for security

Vite: import.meta.env.VITE\_OPENWEATHER\_KEY

CRA: process.env.REACT\_APP\_OPENWEATHER\_KEY

--------------------------------------------------- \*/

const API\_KEY = import.meta?.env?.VITE\_OPENWEATHER\_KEY || process.env.REACT\_APP\_OPENWEATHER\_KEY || 'YOUR\_OPENWEATHER\_API\_KEY\_HERE';

const CityAndTime = ({ cityName, lat, lon, setLat, setLon }) => {

const [weather, setWeather] = useState(null); // full current weather object

const [forecast, setForecast] = useState(null); // forecast list (array)

const [uvIndex, setUVIndex] = useState(null); // number

/\* ---------------- Fetch All (by coords) ---------------- \*/

const fetchAll = async (latitude, longitude) => {

try {

// current weather

const { data: current } = await axios.get(

`https://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lon=${longitude}&units=metric&appid=${API\_KEY}`

);

setWeather(current);

setLat?.(current.coord.lat);

setLon?.(current.coord.lon);

// 5-day / 3-hour forecast

const { data: fc } = await axios.get(

`https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}&lon=${longitude}&units=metric&appid=${API\_KEY}`

);

setForecast(fc.list);

// UV index (legacy endpoint still works)

const { data: uv } = await axios.get(

`https://api.openweathermap.org/data/2.5/uvi?lat=${latitude}&lon=${longitude}&appid=${API\_KEY}`

);

setUVIndex(uv.value);

} catch (err) {

console.error('fetchAll error:', err);

toast.error('Unable to load weather data.');

}

};

/\* -------- Fetch by City then call fetchAll -------- \*/

const fetchByCity = async (city) => {

if (!city) return;

try {

const enc = encodeURIComponent(city);

const { data } = await axios.get(

`https://api.openweathermap.org/data/2.5/weather?q=${enc}&units=metric&appid=${API\_KEY}`

);

// now fetch using coords returned

fetchAll(data.coord.lat, data.coord.lon);

} catch (err) {

console.error('fetchByCity error:', err);

toast.error('City not found.');

}

};

/\* -------------- Effect: respond to props -------------- \*/

useEffect(() => {

if (cityName) {

fetchByCity(cityName);

} else if (lat != null && lon != null) {

fetchAll(lat, lon);

} else {

// fallback: get browser location

if ('geolocation' in navigator) {

navigator.geolocation.getCurrentPosition(

({ coords }) => fetchAll(coords.latitude, coords.longitude),

() => toast.error('Enable location or search a city.')

);

} else {

toast.error('No geolocation support. Please enter a city.');

}

}

// eslint-disable-next-line react-hooks/exhaustive-deps

}, [cityName, lat, lon]);

/\* -------------- Loading state -------------- \*/

if (!weather || !forecast) {

return (

<div className="flex justify-center items-center text-white text-3xl h-40">

Loading…

</div>

);

}

/\* -------------- Extract data -------------- \*/

const { main, sys, wind, weather: wArr, name } = weather;

const iconCode = wArr?.[0]?.icon;

const description = wArr?.[0]?.description;

return (

<>

{/\* main panel \*/}

<div className="flex flex-col xl:flex-row gap-4 mt-4 text-white">

{/\* city / time \*/}

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 w-full xl:w-1/3 flex flex-col items-center">

<h1 className="text-2xl font-bold">{name}</h1>

<img src={sunImg} alt="icon" className="w-24" />

<Clock />

</div>

{/\* temp + sunrise/sunset \*/}

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 flex-grow flex flex-col md:flex-row md:items-center gap-4">

<h2 className="text-5xl font-bold">{Math.round(main.temp)}°C</h2>

<p>

Feels like{' '}

<span className="font-bold">{Math.round(main.feels\_like)}°C</span>

</p>

<div className="flex md:flex-col gap-4 md:ml-auto">

<div className="flex items-center gap-2">

<img src={sunrise} alt="sunrise" className="h-8" />

<div>

<h6>Sunrise</h6>

<p>{new Date(sys.sunrise \* 1000).toLocaleTimeString()}</p>

</div>

</div>

<div className="flex items-center gap-2">

<img src={sunset} alt="sunset" className="h-8" />

<div>

<h6>Sunset</h6>

<p>{new Date(sys.sunset \* 1000).toLocaleTimeString()}</p>

</div>

</div>

</div>

</div>

{/\* icon & desc \*/}

<div className="flex flex-col items-center justify-center">

<img

src={`https://openweathermap.org/img/wn/${iconCode}@2x.png`}

alt="weather"

className="w-32 h-32"

/>

<p className="text-xl font-bold capitalize">{description}</p>

</div>

</div>

{/\* extra metrics \*/}

<div className="grid grid-cols-2 md:grid-cols-4 gap-4 mt-4 text-white">

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 flex flex-col items-center">

<img src={humidity} alt="Humidity" className="h-8 mb-1" />

<p className="text-lg font-bold">{main.humidity}%</p>

<h6>Humidity</h6>

</div>

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 flex flex-col items-center">

<img src={windIcon} alt="Wind" className="h-8 mb-1" />

<p className="text-lg font-bold">{wind.speed} km/h</p>

<h6>Wind Speed</h6>

</div>

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 flex flex-col items-center">

<img src={pressure} alt="Pressure" className="h-8 mb-1" />

<p className="text-lg font-bold">{main.pressure} hPa</p>

<h6>Pressure</h6>

</div>

<div className="bg-[#050e1fde] shadow-lg rounded-lg p-4 flex flex-col items-center">

<img src={uvIcon} alt="UV" className="h-8 mb-1" />

<p className="text-lg font-bold">{uvIndex ?? 'N/A'}</p>

<h6>UV</h6>

</div>

</div>

{/\* forecast \*/}

<ForeCast forecast={forecast} />

</>

);

};

export default CityAndTime;**index.css**

@@tailwind base;

@tailwind components;

@tailwind utilities;

body {

background-color: #09162e;

}