**Lists , Hooks , Localstorage , Api Project**

**1.Lists**

**1 What is a List in React?**

**A List in React refers to rendering a collection of items (like an array) as multiple React elements using .map() method.**

**It helps display dynamic data like user profiles, products, todos, etc.**

import React from "react";

import FruitList from "./FruitList";

import UserList from "./UserList";

function App() {

return (

<div>

<FruitList />

<UserList />

</div>

);

}

export default App;

// FruitList.js

import React from "react";

const FruitList = () => {

const fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"];

return (

<div>

<h2>Fruit List</h2>

<ul>

{fruits.map((fruit, index) => (

<li key={index}>{fruit}</li>

))}

</ul>

</div>

);

};

export default FruitList;

// UserList.js

import React from "react";

const UserList = () => {

const users = [

{ id: 1, name: "Vansh" },

{ id: 2, name: "Aman" },

{ id: 3, name: "Riya" },

{ id: 4, name: "Kiran" },

];

return (

<div>

<h2>User List</h2>

<ul>

{users.map((user) => (

<li key={user.id}>{user.name}</li>

))}

</ul>

</div>

);

};

export default UserList;

**Hooks**

Hooks are built-in functions in React that let you use state and other React features in functional components.

**🔧 1. useState Example:**

jsx

CopyEdit

import { useState } from "react";

function Counter() {

const [count, setCount] = useState(0);

return (

<>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increment</button>

</>

);

}

✅ useState(0) — creates state variable count with default 0.

**🔁 2. useEffect Example:**

jsx

CopyEdit

import { useEffect, useState } from "react";

function Timer() {

const [seconds, setSeconds] = useState(0);

useEffect(() => {

const interval = setInterval(() => setSeconds(s => s + 1), 1000);

return () => clearInterval(interval); // cleanup

}, []);

return <p>Timer: {seconds}s</p>;

}

useEffect runs after the component mounts — perfect for API calls, timers, etc.

**Rules of Hooks:**

1. ✅ **Only call hooks inside functional components or custom hooks**
2. ✅ **Always call hooks at the top level (not inside loops or conditions)**

**📦 Custom Hooks:**

You can create your own custom hook:

jsx

CopyEdit

function useTitle(title) {

useEffect(() => {

document.title = title;

}, [title]);

}

**Summary:**

* Hooks add power to functional components
* No need to use class components anymore
* useState & useEffect are the most commonly used
* Improve readability, reusability, and performance

**Api Project**

Create a simple web app in React that fetches a list of users from a public API and displays them in a clean, card-style layout.

**🛠️ Technologies:**

* React (with Hooks)
* fetch or Axios (for API calls)
* Functional Components
* (Optional: TailwindCSS or Bootstrap for styling)

**🌐 API to Use:**

url

CopyEdit

https://jsonplaceholder.typicode.com/users

This will return an array of fake users (name, email, address, phone, etc.).

**📁 Project Folder Structure:**

css

CopyEdit

src/

│

├── components/

│ ├── UserCard.jsx

│

├── App.jsx

└── index.js

**✨ Features to Implement:**

1. ✅ Fetch users from API when the app loads.
2. ✅ Display each user in a styled card (name, email, city).
3. ✅ Show a loading spinner while fetching.
4. ✅ Show an error message if the API call fails.
5. ✅ Bonus: Add a search filter by name or city.

**💻 Code Walkthrough:**

**🔸 App.jsx**

jsx

CopyEdit

import React, { useState, useEffect } from 'react';

import UserCard from './components/UserCard';

function App() {

const [users, setUsers] = useState([]);

const [loading, setLoading] = useState(true);

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

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => {

if (!res.ok) throw new Error("Failed to fetch data");

return res.json();

})

.then(data => setUsers(data))

.catch(err => setError(err.message))

.finally(() => setLoading(false));

}, []);

if (loading) return <p>Loading...</p>;

if (error) return <p>Error: {error}</p>;

return (

<div className="user-list">

<h2>User Directory</h2>

{users.map(user => (

<UserCard key={user.id} user={user} />

))}

</div>

);

}

export default App;

**🔸 UserCard.jsx**

jsx

CopyEdit

import React from 'react';

function UserCard({ user }) {

return (

<div className="user-card">

<h3>{user.name}</h3>

<p><strong>Email:</strong> {user.email}</p>

<p><strong>City:</strong> {user.address.city}</p>

</div>

);

}

export default UserCard;

**🎨 (Optional) Basic Styling:**

You can use TailwindCSS, Bootstrap, or plain CSS like:

css

CopyEdit

.user-card {

border: 1px solid #ccc;

padding: 15px;

margin: 10px;

border-radius: 8px;

background-color: #f9f9f9;

}

**🧪 Bonus Ideas to Extend:**

* 🔍 Add search filter input
* ✅ Add pagination
* 🔁 Add refresh button
* 📋 Show user’s company or phone number
* 🌙 Add dark mode toggle

**🎯 Summary:**

* You learned how to build an API project in React
* Used useEffect for API calls
* Used useState for loading, error, and data
* Separated components for clean structure

**Localstorage**

**Limitations of LocalStorage:**

* **Only stores strings (you must use JSON.stringify / JSON.parse for objects/arrays).**
* **Data is not encrypted — not safe for passwords or sensitive info.**
* **Storage is synchronous (can block the main thread for large data).**

**🧪 JSON Example:**

**js**

**CopyEdit**

**const user = { name: "Sagar", age: 22 };**

**// Save object**

**localStorage.setItem("user", JSON.stringify(user));**

**// Get object**

**const storedUser = JSON.parse(localStorage.getItem("user"));**

**console.log(storedUser.name); // Sagar**

**🎯 Summary:**

* **LocalStorage lets you store small amounts of data permanently in the browser.**
* **In React, use useEffect and useState to read/write it.**
* **Ideal for things like saved preferences, caching, and simple offline storage.**

**What is LocalStorage?**

**LocalStorage is a built-in Web API that allows you to store key-value data in the browser, even after the page is reloaded or the browser is closed.**

**🧠 It stores data as strings and can hold up to ~5MB.**

**🔧 Basic JavaScript Usage:**

**js**

**CopyEdit**

**// ✅ Set Item**

**localStorage.setItem('username', 'Sagar');**

**// ✅ Get Item**

**const user = localStorage.getItem('username'); // "Sagar"**

**// ✅ Remove Item**

**localStorage.removeItem('username');**

**// ✅ Clear all data**

**localStorage.clear();**

**🟡 All keys and values are stored as strings.**

**⚛️ Using LocalStorage in React**

**We typically use useEffect + useState to read/write from/to LocalStorage.**

**📌 Example: Save Counter in LocalStorage**

**jsx**

**CopyEdit**

**import { useState, useEffect } from 'react';**

**function Counter() {**

**const [count, setCount] = useState(() => {**

**// Load from localStorage when component mounts**

**return Number(localStorage.getItem("count")) || 0;**

**});**

**useEffect(() => {**

**// Save to localStorage when count changes**

**localStorage.setItem("count", count);**

**}, [count]);**

**return (**

**<>**

**<p>Count: {count}</p>**

**<button onClick={() => setCount(count + 1)}>+</button>**

**</>**

**);**

**}**