INTERNSHIP REPORT

WEEK 7 DAY 2

Submitted to:	**Ali Hyder**	Submission Date:	**5th August, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

React.js Advanced

Topics: React useEffect for API Calls using fetch

Objective

To learn and apply the useEffect hook in React to make asynchronous API calls using the fetch() method, and display fetched data in a component once it's loaded.

Tasks Completed:

1. **Understood the useEffect Hook:**
    - Learned that useEffect() runs side effects like API calls after the component mounts.
2. **Created a Component to Fetch API Data:**
    - Made a UserList.js component that fetches data from a public API (https://jsonplaceholder.typicode.com/users).
3. **Used useState to Store Fetched Data:**
    - Used useState([]) to store user data and display it using .map().
4. **Handled Loading & Error States:**
    - Added loading and error states to show appropriate UI feedback.
5. **Displayed Fetched Data in a List:**
    - Rendered a list of user names and emails from the fetched API data.

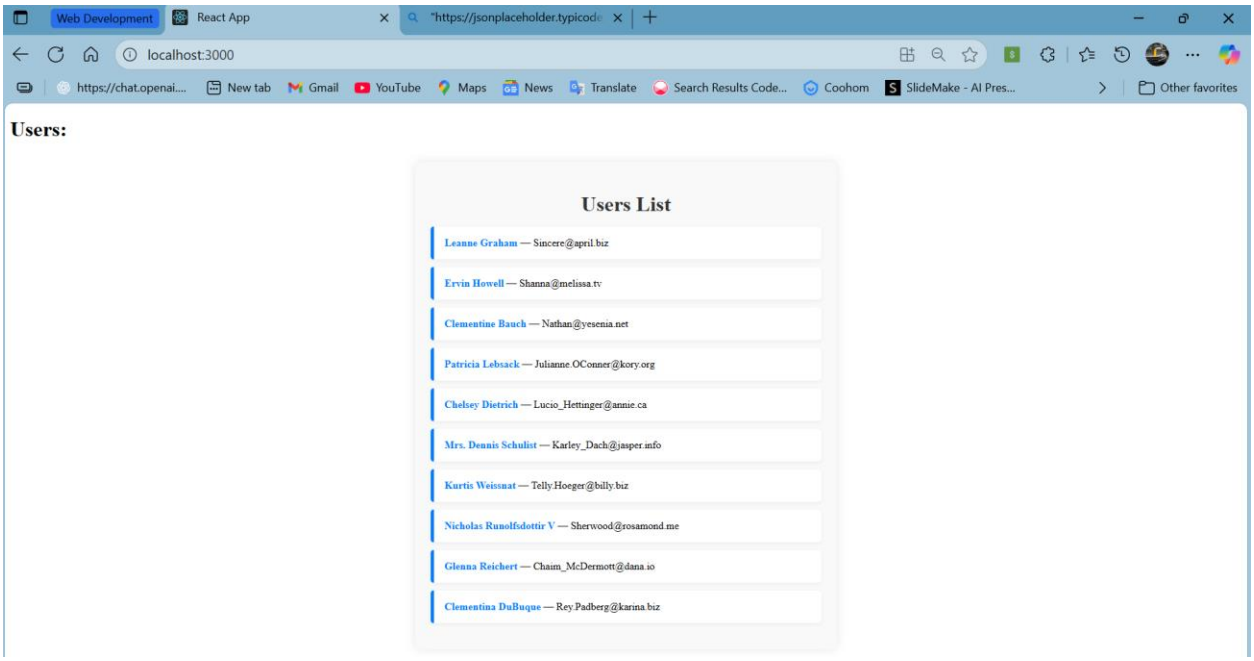
Key Learnings:

- useEffect allows side effects like fetching data on component mount.
- Using useState with useEffect helps manage data flow and re-rendering.
- Always add error handling when working with real APIs.
- The fetch call must be inside useEffect to avoid infinite loops.

Code Sample – UserList.js:

```
import React, { useEffect, useState } from "react";

export default function UserList() {

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

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

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



 useEffect(() => {

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

   .then((res) => {

    if (!res.ok) throw new Error("API Error");

    return res.json();

   })

   .then((data) => {

    setUsers(data);

    setLoading(false);

   })

   .catch((err) => {

    setError(err.message);
```

```
      setLoading(false);
    });
  }, []);

  if (loading) return <p>Loading users...</p>;
  if (error) return <p>Error: {error}</p>;

  return (
    <ul>
      {users.map((user) => (
        <li key={user.id}>
          <strong>{user.name}</strong> — {user.email}
        </li>
      ))}
    </ul>
  );
}
```



Code Sample Used in App.js:

```
import React from "react";
import UserList from "./pages/UserList";

function App() {
 return (
  <div>
   <h1>Users:</h1>
   <UserList />
  </div>
 );
}

export default App;
```

Concept	Description
useEffect	Executes code after render (side effects)
fetch()	Calls external API
useState	Stores users, loading, and error data
map()	Loops through array to display users

CONCUSLION:

Today's task deepened my understanding of how to handle data fetching in React using the useEffect hook. I practiced clean coding, asynchronous operations, state management, and user feedback through loading/error messages.