INTERNSHIP REPORT

WEEK 7 DAY 3

Submitted to:	**Ali Hyder**	Submission Date:	**7th August, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

React.js Advanced

Topics: React Custom Hooks (Basic Intro)

Objective

To understand the concept of **custom hooks** in React, explore why and when to use them, and create a basic reusable hook to demonstrate clean separation of logic and reusable patterns across components.

Tasks Completed:

1. **Learned the Purpose of Custom Hooks:**

- Hooks allow sharing logic between components.
- Helps avoid code repetition by extracting reusable logic.

2. **Created First Custom Hook – useFetch:**

- Created a hook that fetches data from an API.
- Hook returns data, loading, and error.

3. **Used the Custom Hook in a Component:**

- Called useFetch inside UserList.js to fetch and display users.
- Used the returned values to display loading, error, and results.

4. **Practiced Clean Code Principles:**

- Separated concerns between UI and logic.
- Kept components focused only on rendering UI.

Key Learnings:

- **Custom hooks** are JavaScript functions starting with use (e.g., useFetch, useCounter).
- They can use **other hooks like useState, useEffect** inside them.
- Helps improve **code reusability**, readability, and maintainability.
- Encouraged to use them when the same logic is repeated in multiple components.

Code Summary:

useFetch.js

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



function useFetch(url) {

 const [data, setData] = useState(null);

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

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



 useEffect(() => {

  fetch(url)

   .then((res) => {

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

    return res.json();

   })

   .then((data) => {

    setData(data);

    setLoading(false);
```

```
    })
    .catch((err) => {
      setError(err.message);
      setLoading(false);
    });
  }, [url]);

  return { data, loading, error };
}

export default useFetch;
```

UserList.js

```
import React from "react";
import useFetch from "../hooks/useFetch";

export default function UserList() {
  const { data: users, loading, error } =
useFetch("https://jsonplaceholder.typicode.com/users");

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

```
  return (
    <div>
      <h2>Fetched Users</h2>
      <ul>
        {users.map((user) => (
          <li key={user.id}>
            <strong>{user.name}</strong> - {user.email}
          </li>
        ))}
      </ul>
    </div>
  );
}
```

About.js

```
import React from "react";


export default function About() {
  return <h1>This is the About Page</h1>;
}
```

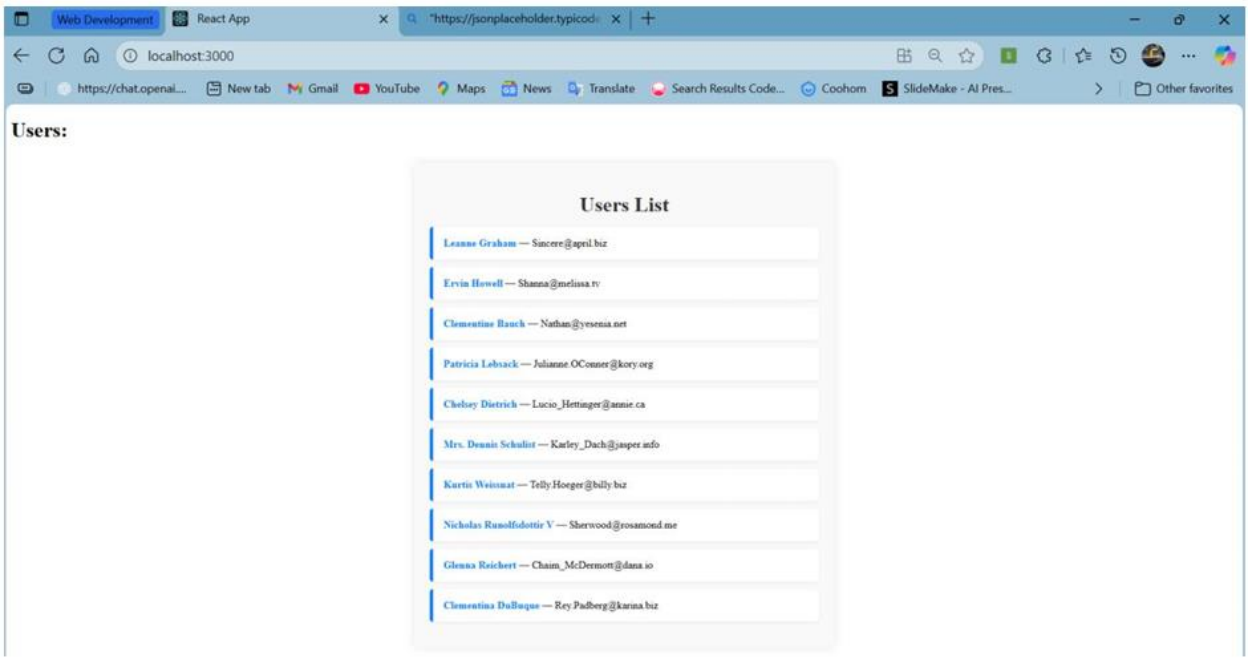
NotFound.js

```
import React from "react";
```

```
export default function NotFound() {

  return <h1>404 - Page Not Found</h1>;

}
```



CONCUSLION:

Today, I successfully created and implemented my **first custom hook** in React. This helped me understand how to **separate reusable logic** from the component UI. By doing so, I improved the maintainability and scalability of the application. This was an important step forward in writing clean, professional React code and has prepared me to handle more complex logic with confidence.