

## Experiment 2

**Student Name:** Ansh Sangal

**Branch:** BE CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** Full Stack Development

**UID:** 23BCS12003

**Section/Group:** KRG 3A

**Date of Performance:** 19/01/26

**Subject Code:** 23CSH-309

### Aim:

To implement Single Page Application (SPA) navigation in the EcoTrack application using React Router, secure application routes using context-based authentication, and extend nested dashboard routing through follow-up enhancements.

### Objective:

After completing this experiment and its follow-up tasks, the students will be able to:

- Configure client-side routing in a React application using React Router
- Implement SPA navigation without full page reloads
- Design and apply protected routes using route-guard patterns
- Manage shared authentication state using React Context API
- Implement nested routing to build dashboard-style layouts
- Extend existing nested routes by adding new dashboard sections
- Implement logout functionality by updating shared context state
- Analyze route access behavior and explain redirection logic
- Understand the role of Context API in shared state management and its comparison with Redux at an introductory level

### Implementation/Code:

#### Dashboard.jsx:

```
import { logs } from "../data/logs"; import { Link,  
useNavigate } from "react-router-dom"; import {  
useAuth } from "../context/AuthContext";
```

```
const Dashboard = () => {  const totalCarbon = logs.reduce((total,  
log) => total + log.carbon, 0); const highCarbonActivities =  
logs.filter(log => log.carbon > 4);
```

```
const lowCarbonActivities = logs.filter(log => log.carbon <= 4);
```

```
const { logout } = useAuth();  
const navigate = useNavigate();
```

```
const handleLogout = () => {  
  logout();  
  navigate("/login");  
};
```

```
return (  
  <div>  
    <h2 style={{ color: "green" }}>Dashboard</h2>
```

```
    <nav>  
      <Link style={{ color: "blue", marginRight: "10px" }} to="/overview">
```

Overview

```
      </Link>  
      <Link style={{ color: "blue", marginRight: "10px" }} to="/reports">
```

Reports

```
      </Link>
```

```
<button  
  onClick={handleLogout}  
  style={{ color: "red", marginLeft: "10px" }}  
>  
  Logout  
</button>  
</nav>
```

```
<p style={{ color: "purple" }}>  
  Total Carbon Footprint: {totalCarbon} Kg  
</p>
```

```
<h3>All Activities</h3>
```

```
<ul>
```

```
  {logs.map(log => (  
<li key={log.id}>  
    {log.activity}: {log.carbon} Kg
```

</li>

)))

</ul>

<h3 style={{ color: "red" }}>High Carbon Activities (> 4 Kg)</h3>

<ul>

{highCarbonActivities.map(log => (

<li key={log.id}>

{log.activity}: {log.carbon} Kg

</li>

)))

</ul>

<h3 style={{ color: "green" }}>Low Carbon Activities (≤ 4 Kg)</h3>

<ul>

{lowCarbonActivities.map(log => (

<li key={log.id}>

{log.activity}: {log.carbon} Kg

</li>

)))

</ul>

</div>

);

};

export default Dashboard;

**Login.jsx:** import { useAuth } from

"../context/AuthContext";

import { useNavigate } from "react-router-dom";

const Login = () => { const

{ login } = useAuth();

const navigate = useNavigate();

const handleLogin = () => {

login();

navigate("/dashboard");

```
};  
return (  
  <div>  
    <h2>Login</h2>  
    <button onClick={handleLogin}>Login to EcoTrack</button>  
  </div>  
);  
};
```

```
export default Login;
```

**ProtectedRoute.jsx:** import { Navigate } from  
"react-router-dom"; import { useAuth } from  
"../context/AuthContext";

```
const ProtectedRoute = ({ children }) => {  
  const { isAuthenticated } = useAuth();  
  
  return isAuthenticated ? children : <Navigate to="/login" />;  
};
```

```
export default ProtectedRoute;
```

## **Output:**



## Login

Login to EcoTrack

## Reports Page

This is the Analytics page of EcoTrack.

## Overview Page

This is the Overview page of EcoTrack.

### Learning Outcome:

- Implement SPA navigation in React using React Router without full page reloads.
- Design and apply protected routes using authentication state with Context API.
- Manage shared authentication state and implement login/logout functionality.
- Build and extend nested dashboard routes for modular layouts.
- Filter, display, and analyze data dynamically while understanding route access behavior.