Experiment 2

Student Name: Aditya Kapoor **UID: 23BCS12101**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 6th **Date of Performance:27/01/2026**
Subject Name: Full Stack Development – II **Subject Code: 23CSH-309**

1. **Aim**: To implement Single Page Application (SPA) routing in the EcoTrack application using React Router, secure application routes using protected routing with Context API–based authentication, and manage shared authentication state across components.

2. **Objective**:

- To configure client-side routing using React Router
- To implement SPA navigation without page reloads
- To protect routes using authentication-based route guards
- To manage shared authentication state using React Context API
- To implement login and logout functionality
- To restrict unauthorized access to protected pages
- To understand redirection logic in protected routing
- To analyze the role of Context API in state management

3. **Implementation / Code:**

Tools & Technologies Used:

- AWS Free Tier Account
- Web Browser (Google Chrome / Firefox)
- Amazon EC2 Service
- RDP Client (Microsoft Remote Desktop)
- Internet-enabled Laptop/Desktop

Implementation Description:

- The EcoTrack application is enhanced by implementing client-side routing using React Router, enabling seamless navigation between different pages without full page reloads.
- An authentication system is implemented using React Context API, which stores and manages the authentication state (isAuthenticated) across the entire application.
- A ProtectedRoute component is created to restrict access to sensitive pages such as Dashboard, Logs, and Data. If the user is not authenticated, they are automatically redirected to the Login page.
- Login functionality updates the authentication state using context, while logout functionality resets the authentication state and redirects the user back to the login page.

- This approach ensures secure navigation, centralized state management, and a smooth SPA user experience.

Sample Code Snippet:

Header.js U eco-track\src\components\Header.js\ Header

```jsx
import {Link} from 'react-router-dom';
import { UserAuth } from '../context/AuthContext';

const Header = () => {
    const { isAuthenticated, setIsAuthenticated } = UserAuth();

        return (
            <header style = {{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', padding: '10px 20px', backgroundColor: '
                <h1>EcoTrack</h1>
                <nav>
                    <Link to="/" Style = {{ color: 'white', marginRight: '1rem' }} > Dashboard </Link>
                    <Link to="/" Style = {{ color: 'white', marginRight: '1rem' }} > Logs </Link>
                    <Link to="/" Style = {{ color: 'white', marginRight: '1rem' }} > Login </Link>
                </nav>
            </header>
        );
};

export default Header;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        node - eco-track

To create a production build, use npm run build.

webpack compiled successfully
```

EXPLORER

```
CREAT...
eco-track
  node_modules
  public
  src
    compone...
      Header.js U
    context
      AuthCon... U
    data
    pages
      dashboar... U
      Dashboa... U
      Dashboa... U
      Dashboa... U
      Dashboa... U
      Dashboa... U
      login.js U
      logs.js U
    routes
    store
      logsSlice.js U
      store.js U
```

DashboardLayout.js U eco-track\src\pages\DashboardLayout.js\ DashboardLayout

```jsx
import {Link, Outlet} from 'react-router-dom';

const DashboardLayout = () => {
    return (
        <div style = {{padding: "1rem"}}>
            <h3>
                Dashboard
            </h3>

            <nav>
                <Link to="summary">Summary</Link> | {" "}
                <Link to="analytics">Analytics</Link>

            </nav>

            <hr/>
            <Outlet />

        </div>
    );
};

export default DashboardLayout;
```

AuthContext.js U eco-track\src\context\AuthContext.js\ AuthProvider

```
1  import { createContext, useContext, useState } from "react";
2
3  const AuthContext = createContext(null);
4
5  export const AuthProvider = ({ children }) => {
6    const [isAuthenticated, setIsAuthenticated] = useState(false);
7
8      return (
9          <AuthContext.Provider value = {{ isAuthenticated, setIsAuthenticated }} >
10             {children}
11         </AuthContext.Provider>
12     );
13 };
14
15
16 export const UserAuth = () => useContext(AuthContext);
17
```

login.js U eco-track\src\pages\login.js\...

```
1   import React from "react";
2   import { UserAuth } from "../context/AuthContext";
3   import { useNavigate } from "react-router-dom";
4
5   const Login = () => {
6
7       const {setIsAuthenticated} = UserAuth();
8       const navigate = useNavigate();
9
10      const handleLogin = () => {
11          // Simulate authentication process
12          setIsAuthenticated(true);
13          navigate("/");
14      };
15
16      return (
17          <div>
18              <h2>Login</h2>
19              <button onClick={handleLogin}>Login</button>
20          </div>
21      );
22  };
23
24
25
26  export default Login;
27
28
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
To create a production build, use npm run build.

webpack compiled successfully
```

ProtectedRoute.js U eco-track\src\routes\ProtectedRoute.js\ Protec

```
1   import { Navigate } from "react-router-dom";
2   import { UserAuth } from "../context/AuthContext";
3
4   const ProtectedRoute = ({ children }) => {
5       const { isAuthenticated } = UserAuth();
6
7       if (!isAuthenticated) {
8           return <Navigate to="/login" replace/>;
9       }
10
11      return children;
12  };
13
14  export default ProtectedRoute;
```

4. Output:

- The EcoTrack application successfully implements SPA routing
- Navigation occurs without full page reloads
- Unauthorized users are redirected to the login page
- Authenticated users can access Dashboard, Logs, and Data pages
- System logs and environmental data are displayed dynamically
- Logout functionality securely ends the session
- Proper route protection is verified using ProtectedRoute

5. Learning Outcomes (What I Have Learnt)

After completing this experiment, the student is able to:

- Implement SPA routing using React Router
- Secure application routes using protected routing
- Manage shared authentication state using Context API
- Implement login and logout functionality
- Understand route redirection logic

Compare Context API with Redux at an introductory level
Build scalable and secure React applications