



Experiment 2

Student Name: Saurabh Kumar

UID: 23BCS10675

Branch: CSE

Section/Group: KRG 3-B

Semester: 6th

Date of Performance: 20/01/2026

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:

```
import { createContext,useContext,useState } from "react";

const AuthContext = createContext(null); //Creating Default value of the authContext

export const AuthProvider=({children})=>{
  const [isAuthenticated, setIsAuthenticated]=useState(false); //Creating Uestate with passing the inital value
}

return (
  //Pasing the context to all the children who are consuming it
  <AuthContext.Provider value={{isAuthenticated,setIsAuthenticated}}>
    {children}
  </AuthContext.Provider>
)

export const useAuth=()=> useContext(AuthContext); //Custom hook which is the medium of passing
```

```
JS login.js > default
import { useAuth } from "../context/AuthContext";
import { useNavigate } from "react-router-dom";

const Login = () => {
  const { setIsAuthenticated } = useAuth();
  const navigate = useNavigate();

  const handleLogin = () => {
    setIsAuthenticated(true);
    navigate("/", { replace: true });
  };

  return (
    <>
      <h3>Login</h3>
      <button onClick={handleLogin}>Login</button>
    </>
  );
};

export default Login;
```

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

const Header=()=>{

  return(
    <>
    <h2>Ecotrack</h2>
    <nav>
      <Link to="/">Dashboard</Link>
      <Link to="/">Logs</Link>
      <Link to="/">Login</Link>
      <Link to="/">DashboardAnalysis</Link>
    </nav>
    </>
  )
}

export default Header;
```

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
import { AuthProvider } from './context/AuthContext.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <AuthProvider>
      <App />
    </AuthProvider>
  </StrictMode>,
)
```

```
App.js > App
import { BrowserRouter, Routes, Route } from "react-router-dom";

import Header from "../Header";
import Login from "../login";
import DashboardLayout from "../DashboardLayout";
import ProtectedRoute from "../ProtectedRoute";
import DashboardSummary from "../DashboardSummary";
import DashboardAnalytics from "../DashboardAnalytics";

function App() {
  return (
    <BrowserRouter>
      <Header />
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route element={<ProtectedRoute />} />
        <Route element={<DashboardLayout />} />
        <Route path="/dashboard" element={<DashboardSummary />} />
        <Route path="/dashboard/analytics" element={<DashboardAnalytics />} />
      </Route>
    </Routes>
  </BrowserRouter>
);
}

export default App;
```

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

EcoTrack

[Dashboard](#) [Login](#)

Login

Login to EcoTrack



EcoTrack

[Dashboard](#) [Logs](#) [Data](#) [Logout](#)

Environmental Data

ID	Category	Value	Impact Level
1	Electricity Usage	120 kWh	Medium
2	Water Consumption	450 Liters	Low
3	Carbon Emission	18 kg CO ₂	High
4	Waste Generated	6 kg	Medium

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