



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment 2

Student Name: Jayanaath S  
Branch: AIT-CSE  
Semester: 6  
Subject Name: Full Stack II

UID: 23BCC70022  
Section: 23AIT\_KRG-1 A  
Date of Performance: 23/01/2026  
Subject Code: 23CSH-382

- **Aim:**

The aim of this experiment is to advance the "Eco-Track" application by implementing a secure navigation system, integrating a centralized authentication state using the React Context API, and organizing complex view structures through nested routing and layouts.

- **Objectives:**

The main objectives of this experiment are as follows:

1. Implementing a robust client-side routing architecture using react-router-dom.
2. Managing global application state (authentication) via the React Context API to control user access.
3. Developing a "Protected Route" pattern to safeguard sensitive dashboard information from unauthenticated users.
4. Utilizing nested routes and layouts to create a multi-view dashboard interface.
5. Applying programmatic navigation to handle user flows like login, logout, and redirection.

- **Implementation:**

The development of the second iteration of Eco-Track followed these technical phases:

1. The existing React environment was updated to include react-router-dom for managing multiple URL paths.
2. An AuthContext.jsx file was created to provide a centralized isLoggedIn state and a setter function to all application components.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH  
UNIVERSITY

Discover. Learn. Empower.

3. A custom useAuth hook was implemented to allow components to easily access and modify the authentication status.
4. A ProtectedRoute component was developed to check the isLoggedIn status; it renders the requested content using <Outlet/> if authenticated, or redirects the user to the login page using <Navigate/> if not.
5. DashboardLayout component was created to provide a consistent side-navigation or sub-menu for dashboard-specific views.
6. Specific sub-pages for Summary, Analytics, and Settings were developed to display categorized system information.
7. The Login.jsx component was updated to trigger the setIsLoggedIn(true) state change and programmatically navigate the user to the home dashboard upon success.
8. The App.jsx entry point was reconfigured to wrap the application in a BrowserRouter and define a hierarchy of public and private routes.
9. The index route and various path-based routes (e.g., /summary, /analytics) were nested within the protected dashboard layout to ensure consistent UI.
10. The Logs.jsx component was integrated into the new routing system to maintain access to filtered high-carbon activity logs.
11. All modules were connected to ensure a seamless flow from authentication to data visualization across different protected views.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- **Output:**

The screenshot shows a web browser window with the URL `localhost:5173/login`. The title bar says "Eco-Track". Below it, there is a navigation menu with the text "Carbon Footprint Tracker", "Dashboard | Logs | Login | Logout". The main content area is titled "Login" and contains a single button labeled "Login". The browser's address bar and various tabs are visible at the top.

The screenshot shows a web browser window with the URL `localhost:5173`. The title bar says "Eco-Track". Below it, there is a navigation menu with the text "Carbon Footprint Tracker", "Dashboard | Logs | Login | Logout". The main content area is titled "Dashboard" and contains three links: "Summary | Analytics | Settings". Below these links, the text "This is Dashboard Summary Page." is displayed. The browser's address bar and various tabs are visible at the top.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- **Results:**

The Eco-Track application was successfully enhanced with an advanced navigation and security architecture. The integration of the React Context API enabled centralized authentication handling across all application pages. The implementation of protected and nested routes ensured that sensitive content remained accessible only to authenticated users while preserving a structured and modular interface. Programmatic navigation and layout-based routing demonstrated effective separation between authentication logic, navigation flow, and data presentation.

- **Learning Outcomes:**

After completing this experiment, I have learnt to:

1. Implement advanced navigation systems using modern routing libraries such as React Router.
2. Create and manage global application state using the React Context API.
3. Secure frontend routes through reusable authentication protection mechanisms.
4. Design modular layouts that support nested routing and structured navigation.
5. Control user navigation and redirection programmatically within a single-page application.
6. Organize large React projects by separating authentication logic, contexts, and layout structures effectively.