DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment No: 2

Student Name: Rashid Khan
Branch: B.E./C.S.E.
Semester: 6th
Subject Name: Full Stack II
Subject Code: 23CSH-309

UID: 23BCS12551
Section/Group: KRG-3B
Date of Performance: 20/01/2026

1. 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.

2. Objective:

- 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

3. Implementation / Code:

// App.jsx

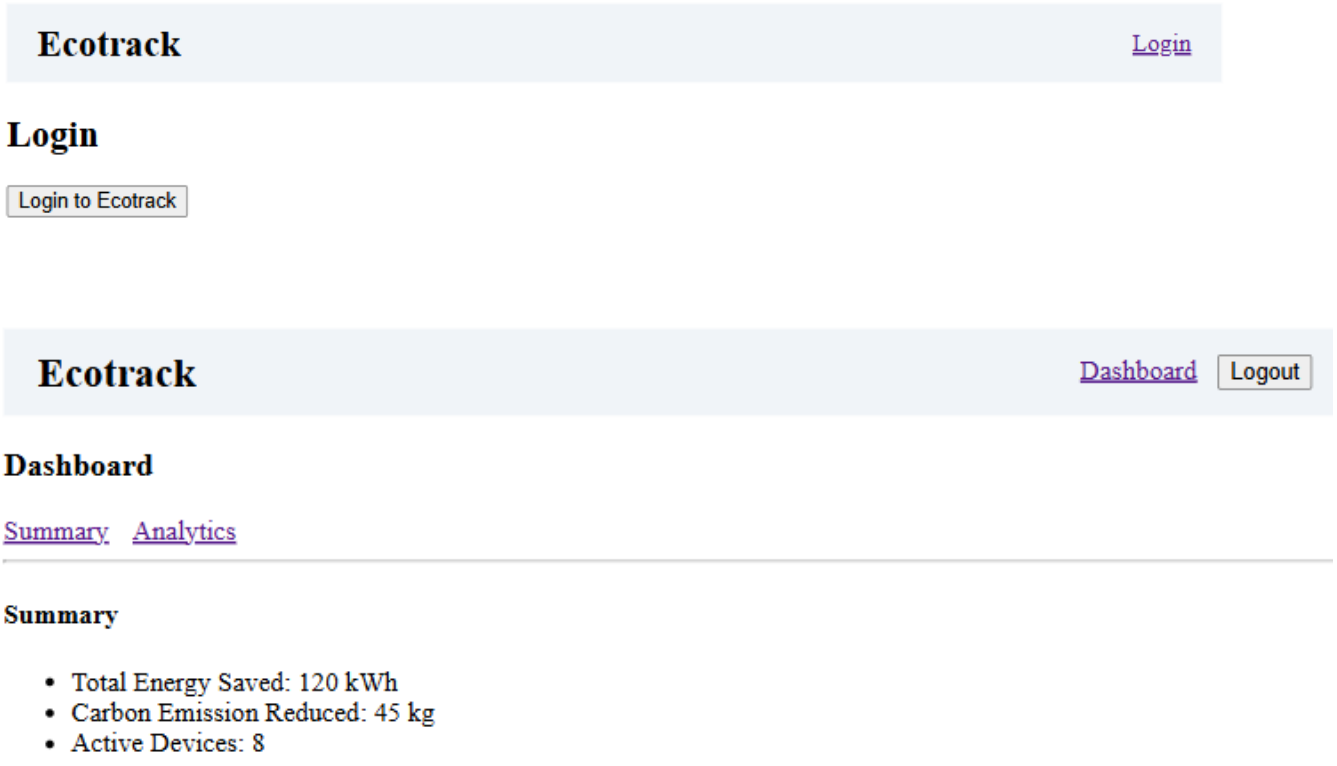
```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Header from "./components/Header";
import Login from "./pages/Login";
import DashboardLayout from "./pages/DashboardLayout";
import DashboardSummary from "./pages/DashboardSummary";
import DashboardAnalytics from "./pages/DashboardAnalytics";
import ProtectedRoutes from "./routes/ProtectedRoutes";
```

```
function App() {
  return (
    <BrowserRouter>
      <Header />
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route
          path="/"
          element={
            <ProtectedRoutes>
              <DashboardLayout />
            </ProtectedRoutes>
          }
        >
          <Route index element={<DashboardSummary />} />
          <Route path="summary" element={<DashboardSummary />} />
          <Route path="analytics" element={<DashboardAnalytics />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
export default App;

// components/Header.jsx
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

const Header = () => {
  const { isAuth, setIsAuth } = useAuth();
  const navigate = useNavigate();

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

```
  return (
    <div
      style={{
        backgroundColor: "#f0f4f8",
        padding: "12px 20px",
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
      }}
    >
      <h2 style={{ margin: 0 }}>Ecotrack</h2>

      <nav style={{ display: "flex", gap: "12px" }}>
        {isAuth && <Link to="/">Dashboard</Link>}
        {isAuth ? (
          <button onClick={handleLogout}>Logout</button>
        ) : (
          <Link to="/login">Login</Link>
        )}
      </nav>
    </div>
  );
};
export default Header;

// pages/DashboardLayout.jsx
import { Link, Outlet } from "react-router-dom";
const DashboardLayout = () => {
  return (
    <div>
      <h3>Dashboard</h3>
      <nav>
        <div style={{display:"flex", gap:"14px"}}>
          <Link to="summary" >Summary</Link>
          <Link to="analytics">Analytics</Link>
        </div>
      </nav>
```

```
      <hr />
      <Outlet />
    </div>
  );
};
export default DashboardLayout;

// routes/ProtectedRoutes.jsx
import { Navigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

const ProtectedRoutes = ({ children }) => {
  const { isAuth } = useAuth();
  if (!isAuth) {
    return <Navigate to="/login" replace />;
  }
  return children;
};
export default ProtectedRoutes;
```

4. Output :



5. Learning Outcomes:

- Understood React Router v6 routing (BrowserRouter, Routes, Route, Outlet)
- Learned protected routes using authentication state
- Used Context API for global auth state (AuthContext)
- Implemented login and logout flow
- Conditional rendering based on authentication status
- Used nested routes with a dashboard layout
- Handled programmatic navigation using useNavigate