Experiment 2

Student Name: Nitin Upadhyay **UID: 23BCS12270**
Branch: CSE **Section/Group: KRG 3-A**
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:

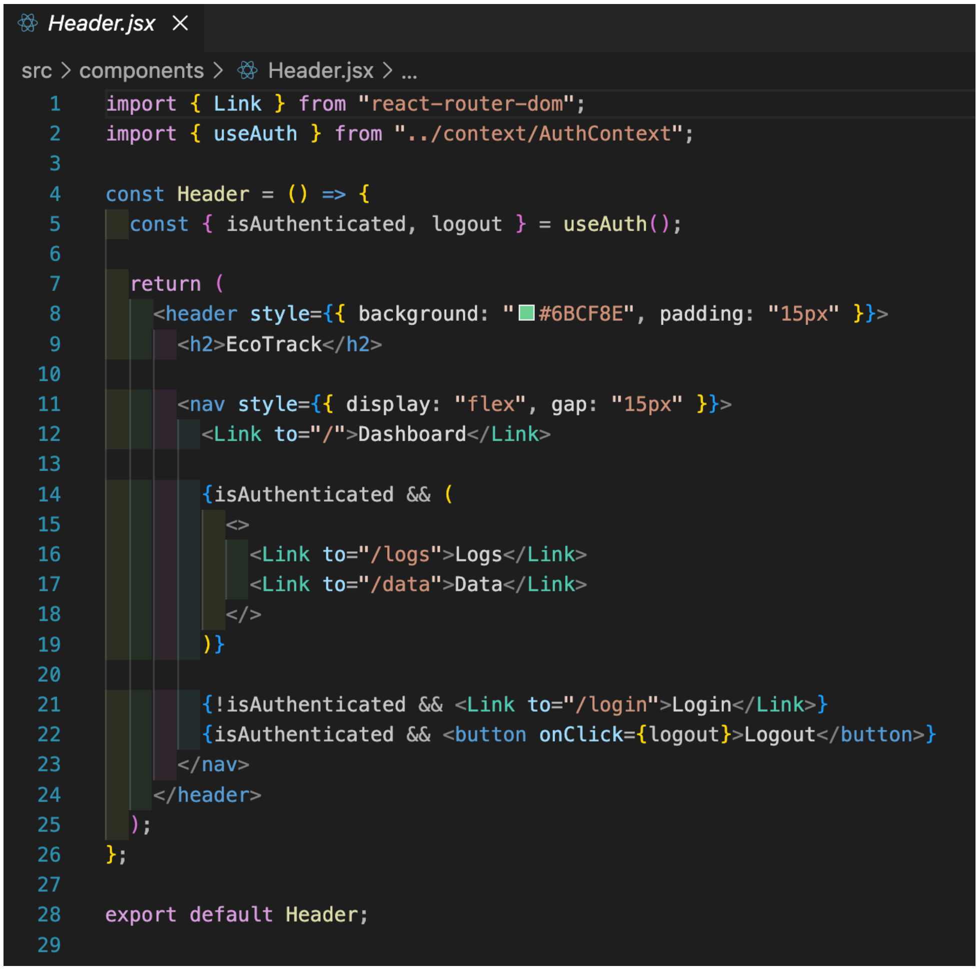
- AWS Free Tier Account
- Web Browser (Google Chrome / Firefox)
- Amazon EC2 Service
- RDP Client (Microsoft Remote Desktop)
- Internet-enabled Laptop/Desktop

Implementation Description:

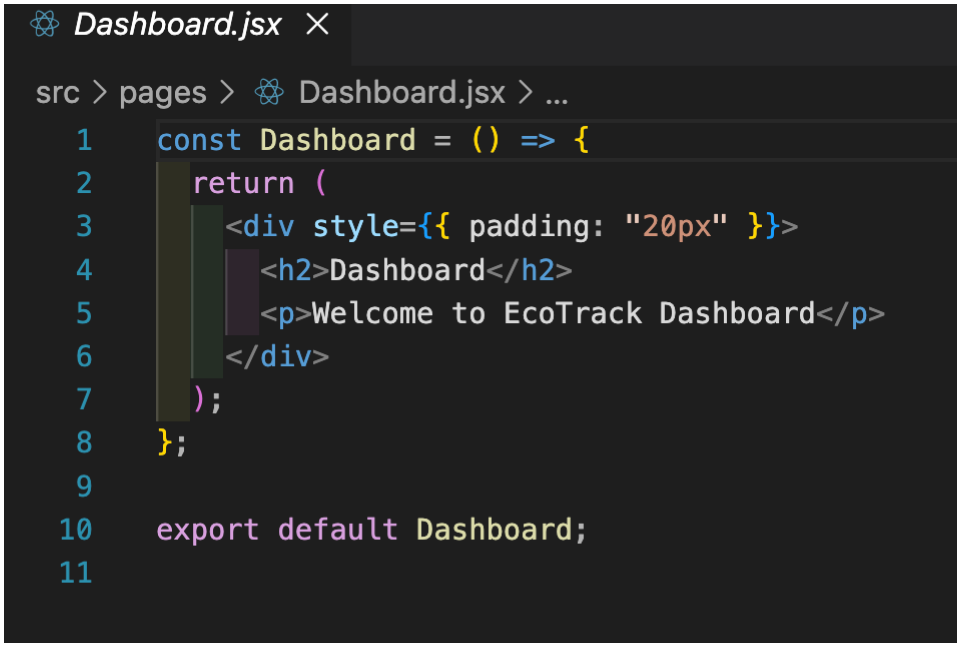
- The EcoTrack application is enhanced by implementing client-side routing using React Router, enabling seamless navigation between different pages without full page reloads.
- An authentication system is implemented using React Context API, which stores and manages the authentication state (isAuthenticated) 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.

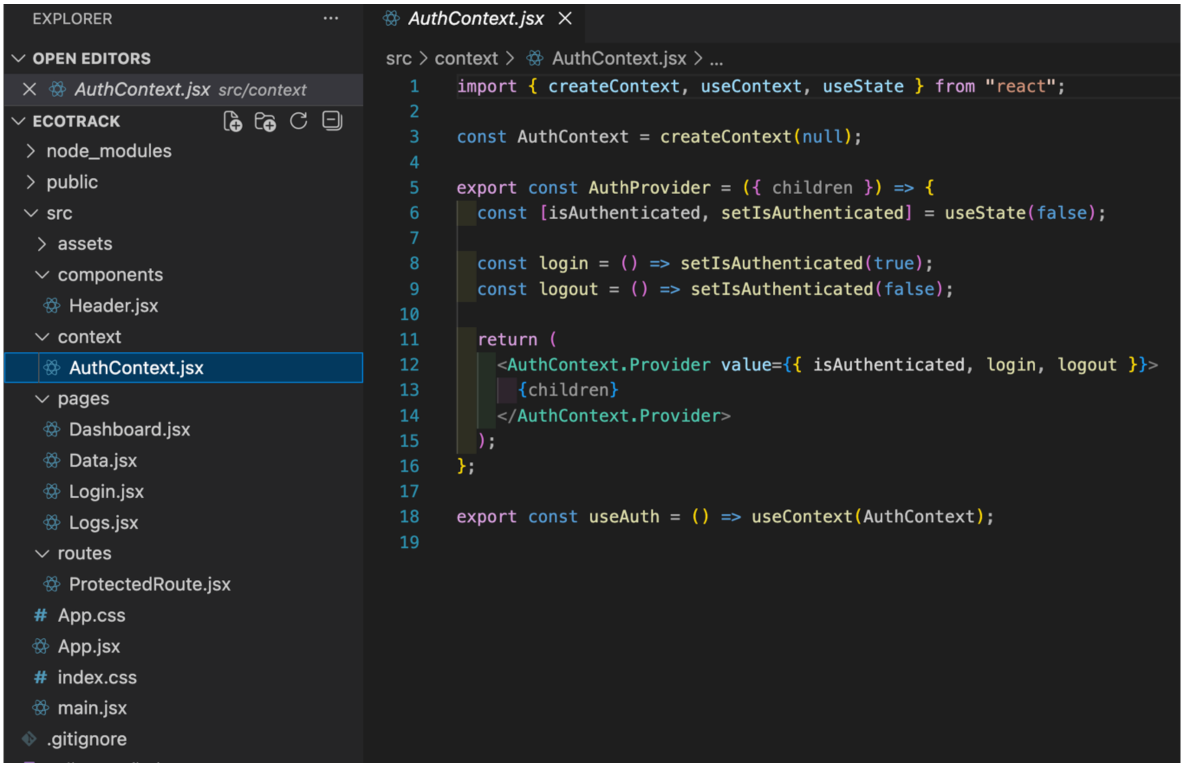
Sample Code Snippet:



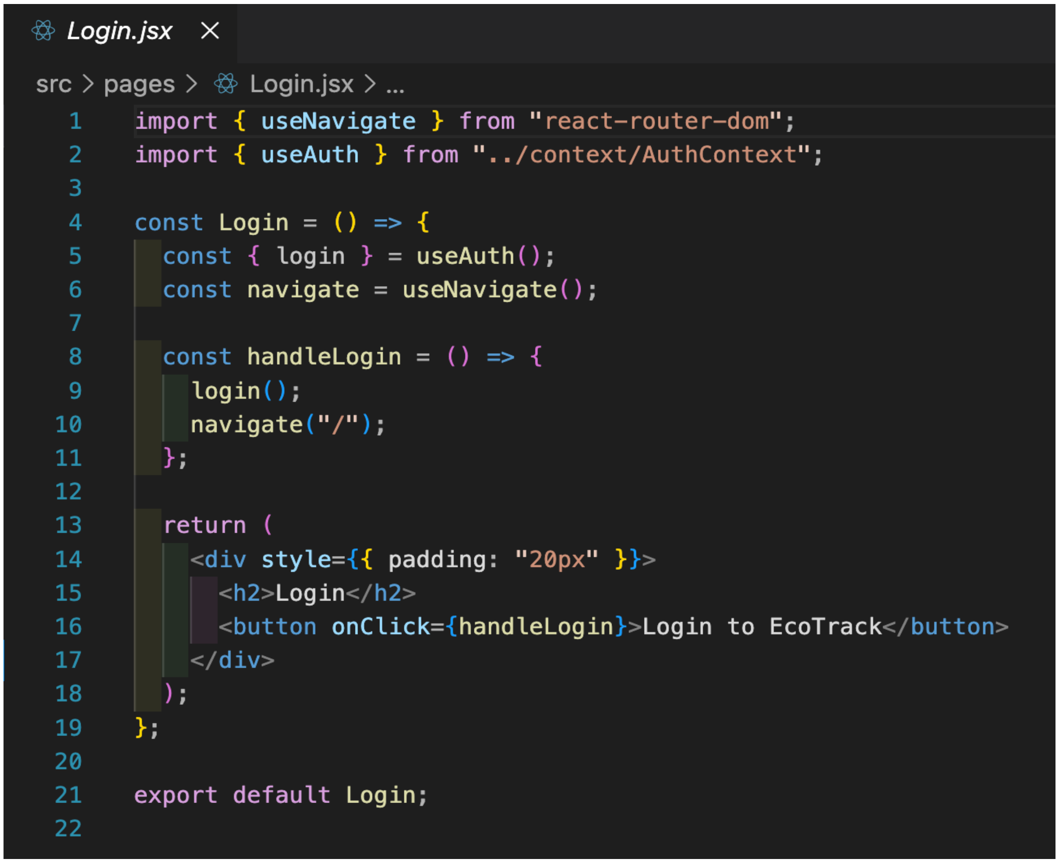
```jsx
import { Link } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

const Header = () => {
  const { isAuthenticated, logout } = useAuth();

  return (
    <header style={{ background: "🟩#6BCF8E", padding: "15px" }}>
      <h2>EcoTrack</h2>

      <nav style={{ display: "flex", gap: "15px" }}>
        <Link to="/">Dashboard</Link>

        {isAuthenticated && (
          <>
            <Link to="/logs">Logs</Link>
            <Link to="/data">Data</Link>
          </>
        )}

        {!isAuthenticated && <Link to="/login">Login</Link>}
        {isAuthenticated && <button onClick={logout}>Logout</button>}
      </nav>
    </header>
  );
};

export default Header;
```



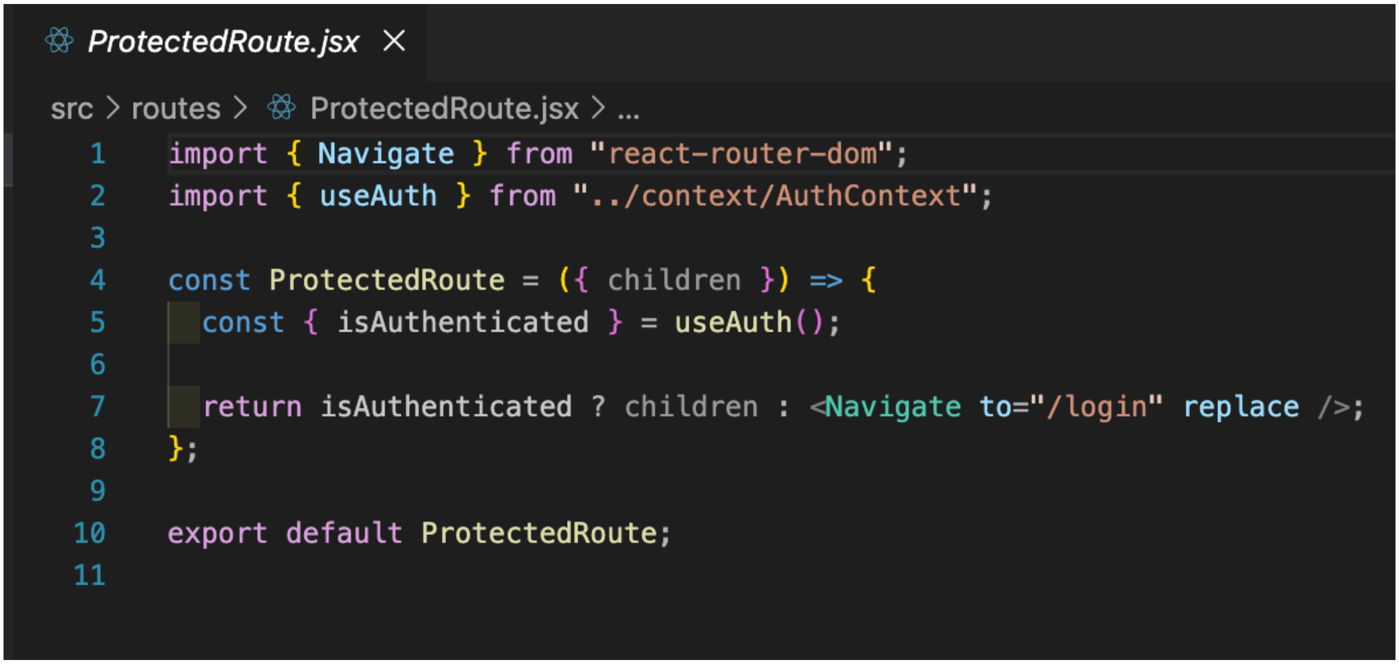
```jsx
const Dashboard = () => {
  return (
    <div style={{ padding: "20px" }}>
      <h2>Dashboard</h2>
      <p>Welcome to EcoTrack Dashboard</p>
    </div>
  );
};

export default Dashboard;
```



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

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const login = () => setIsAuthenticated(true);
  const logout = () => setIsAuthenticated(false);

  return (
    <AuthContext.Provider value={{ isAuthenticated, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => useContext(AuthContext);
```



```jsx
import { useNavigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

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

  const handleLogin = () => {
    login();
    navigate("/");
  };

  return (
    <div style={{ padding: "20px" }}>
      <h2>Login</h2>
      <button onClick={handleLogin}>Login to EcoTrack</button>
    </div>
  );
};

export default Login;
```



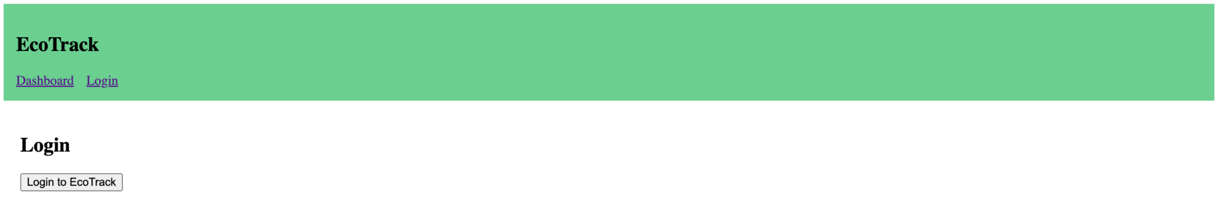
ProtectedRoute.jsx

src > routes > ProtectedRoute.jsx > ...

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

const ProtectedRoute = ({ children }) => {
  const { isAuthenticated } = useAuth();

  return isAuthenticated ? children : <Navigate to="/login" replace />;
};

export default ProtectedRoute;
```

4. Output:

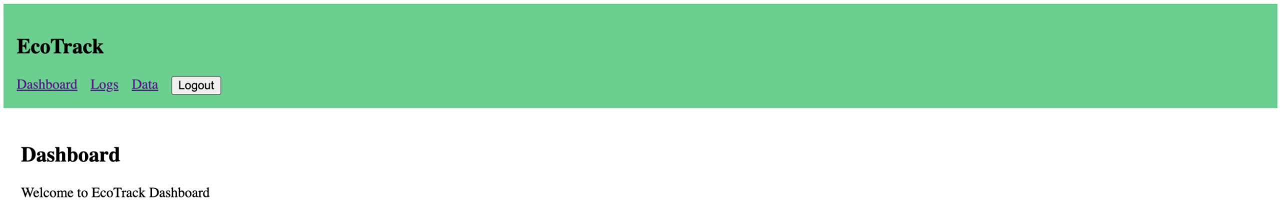
- The EcoTrack application successfully implements SPA routing
- Navigation occurs without full page reloads
- Unauthorized users are redirected to the login page
- Authenticated users can access Dashboard, Logs, and Data pages
- System logs and environmental data are displayed dynamically
- Logout functionality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



EcoTrack

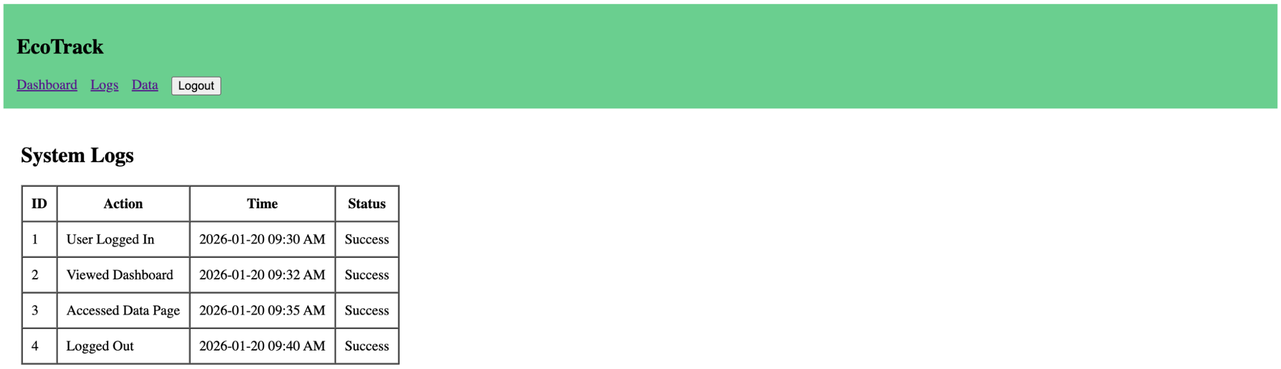
Dashboard Login

Login



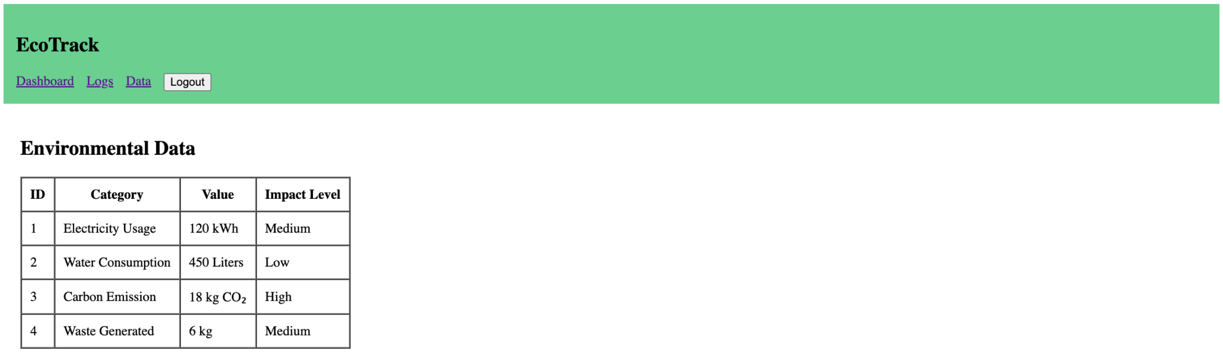
Dashboard

Welcome to EcoTrack Dashboard



System Logs

ID	Action	Time	Status
1	User Logged In	2026-01-20 09:30 AM	Success
2	Viewed Dashboard	2026-01-20 09:32 AM	Success
3	Accessed Data Page	2026-01-20 09:35 AM	Success
4	Logged Out	2026-01-20 09:40 AM	Success



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