CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Experiment 4

Student Name: Anuj Yadav
Branch: BE-AIT-CSE
Semester: 6th
Subject Name: Full Stack II

UID: 23BIA50011
Section/Group: 23AML_KRG-G2
Date of Performance: 7 Feb 2026
Subject Code: 23CSH-382

1. Aim:

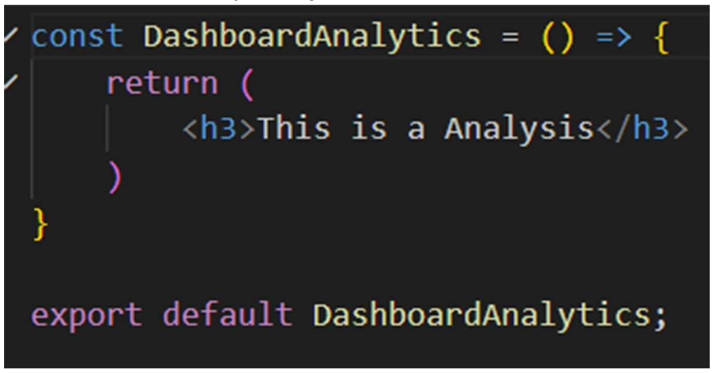
To optimize the performance of the EcoTrack React application using memoization techniques and code splitting, and to enhance the user interface using enterprise-grade Material UI components.

2. Objective:

After completing this experiment, the student will be able to:

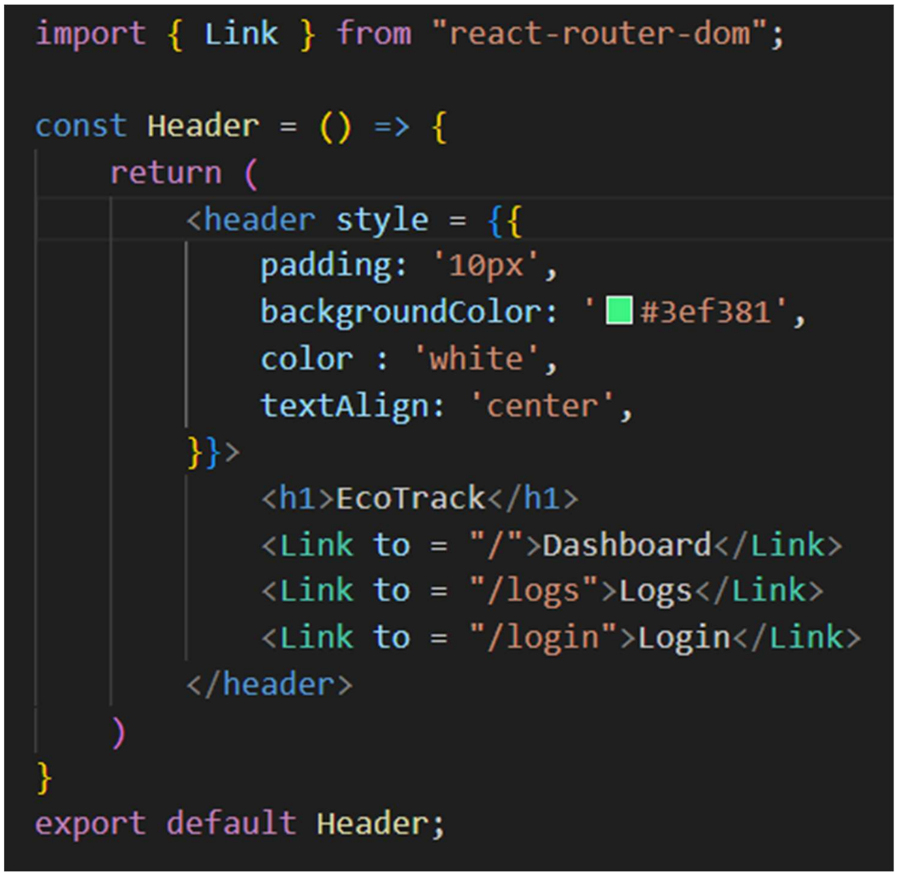
- **1.Understand the causes of unnecessary re-renders in React applications**
- **2.Optimize React components using React.memo to prevent avoidable re-renders**
- **3.Apply useMemo to efficiently compute derived data and avoid redundant calculations**
- **4.Use useCallback to memoize event handler functions and improve component performance**
- **5.Implement lazy loading of components and routes using React.lazy and Suspense**
- **6.Reduce initial bundle size and improve application load performance through code splitting**
- **7.Enhance the visual appearance and usability of the EcoTrack application using Material UI components**
- **8.Design a clean, consistent, and responsive user interface using Material UI layouts and typography**
- **To implement centralized state management in the EcoTrack application using Redux Toolkit and to handle asynchronous data operations using Redux async thunks with proper loading and error states.**

3. Implementation/Code:
DashboardAnalytics.jsx:



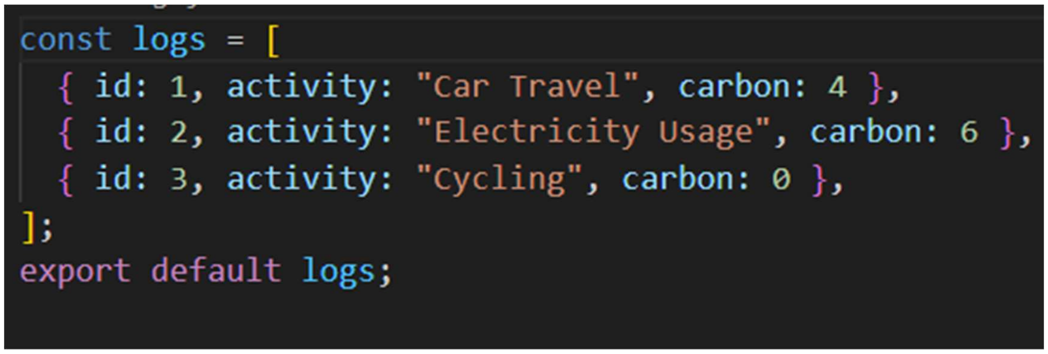
```jsx
const DashboardAnalytics = () => {
    return (
        <h3>This is a Analysis</h3>
    )
}

export default DashboardAnalytics;
```

Header.jsx:



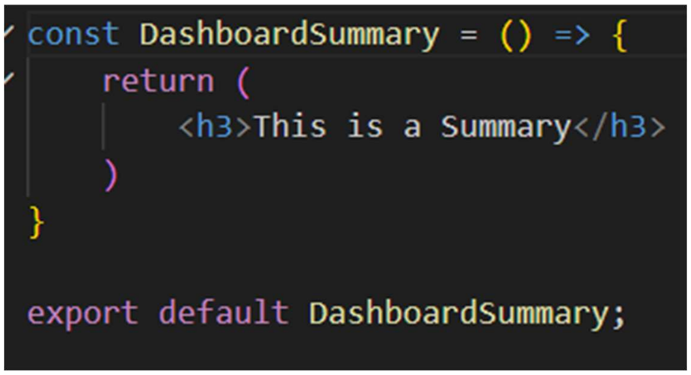
```jsx
import { Link } from "react-router-dom";

const Header = () => {
    return (
        <header style = {{
            padding: '10px',
            backgroundColor: '🟩#3ef381',
            color : 'white',
            textAlign: 'center',
        }}>
            <h1>EcoTrack</h1>
            <Link to = "/">Dashboard</Link>
            <Link to = "/logs">Logs</Link>
            <Link to = "/login">Login</Link>
        </header>
    )
}
export default Header;
```

Logs.js:



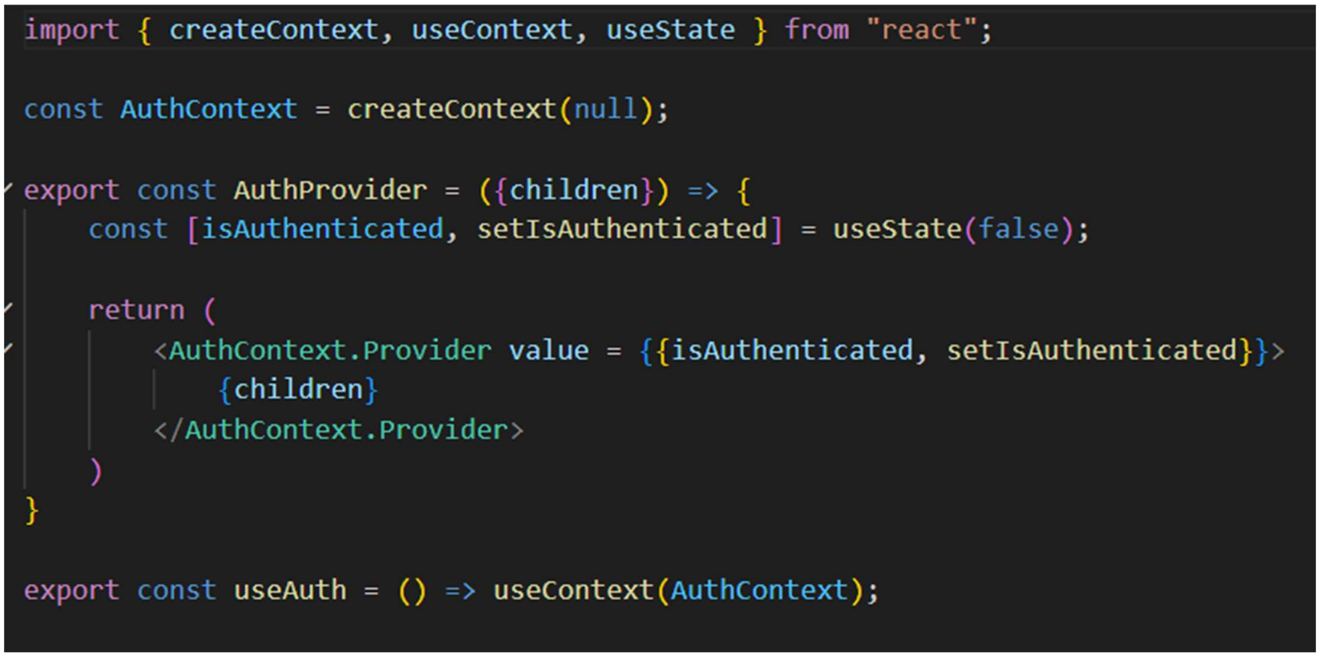
```js
const logs = [
  { id: 1, activity: "Car Travel", carbon: 4 },
  { id: 2, activity: "Electricity Usage", carbon: 6 },
  { id: 3, activity: "Cycling", carbon: 0 },
];
export default logs;
```

DashboardSummary.jsx:



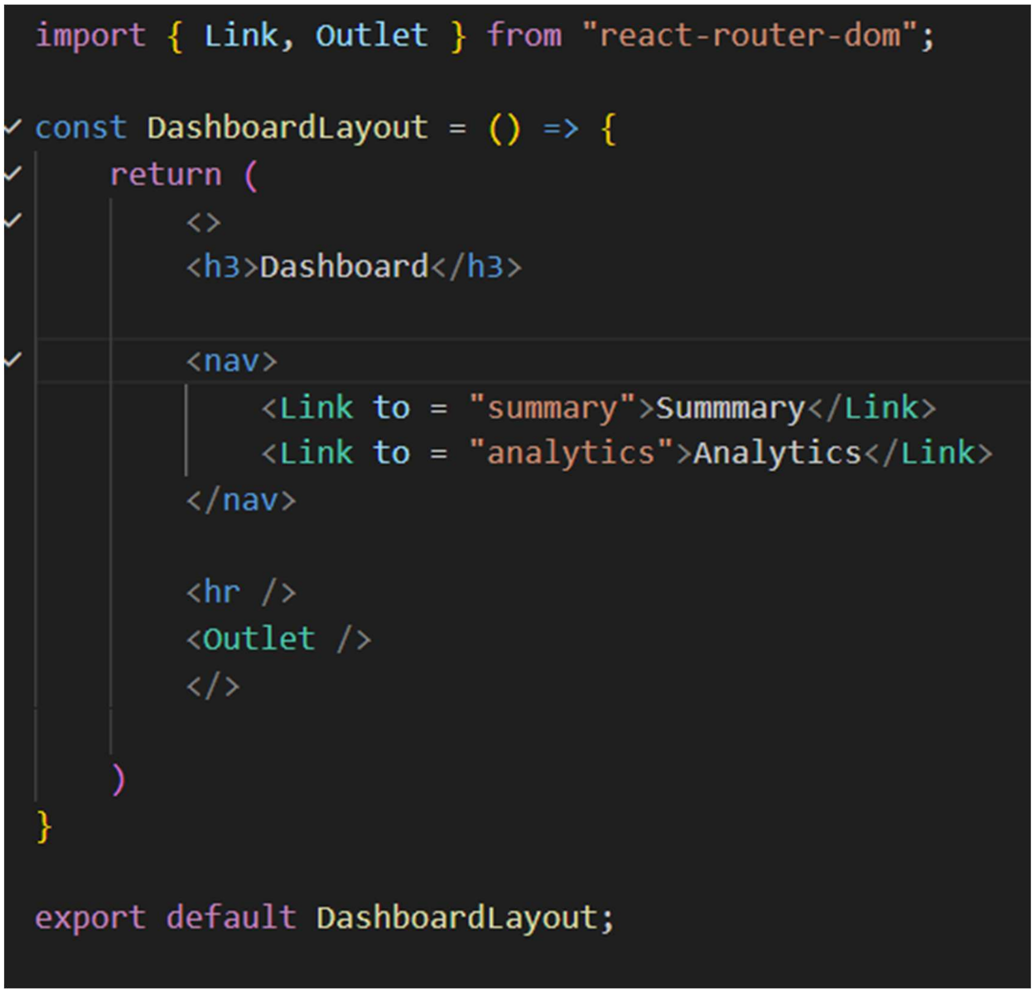
```jsx
const DashboardSummary = () => {
    return (
        <h3>This is a Summary</h3>
    )
}

export default DashboardSummary;
```

AuthContext.jsx:



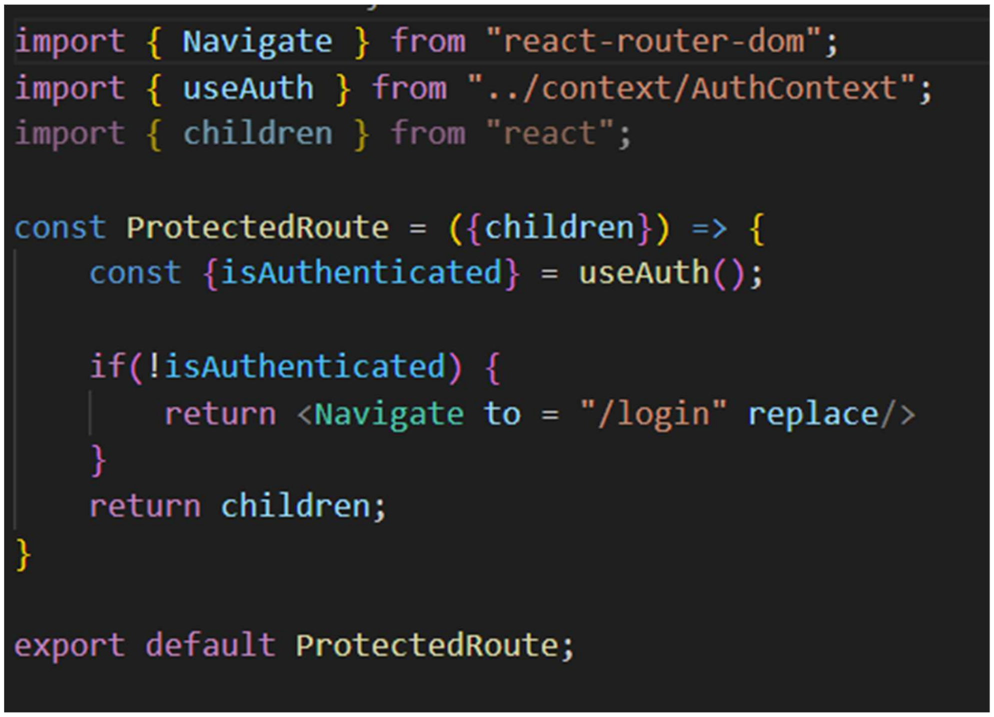
```jsx
import { createContext, useContext, useState } from "react";

const AuthContext = createContext(null);

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

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

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

DashboardLayout.jsx:



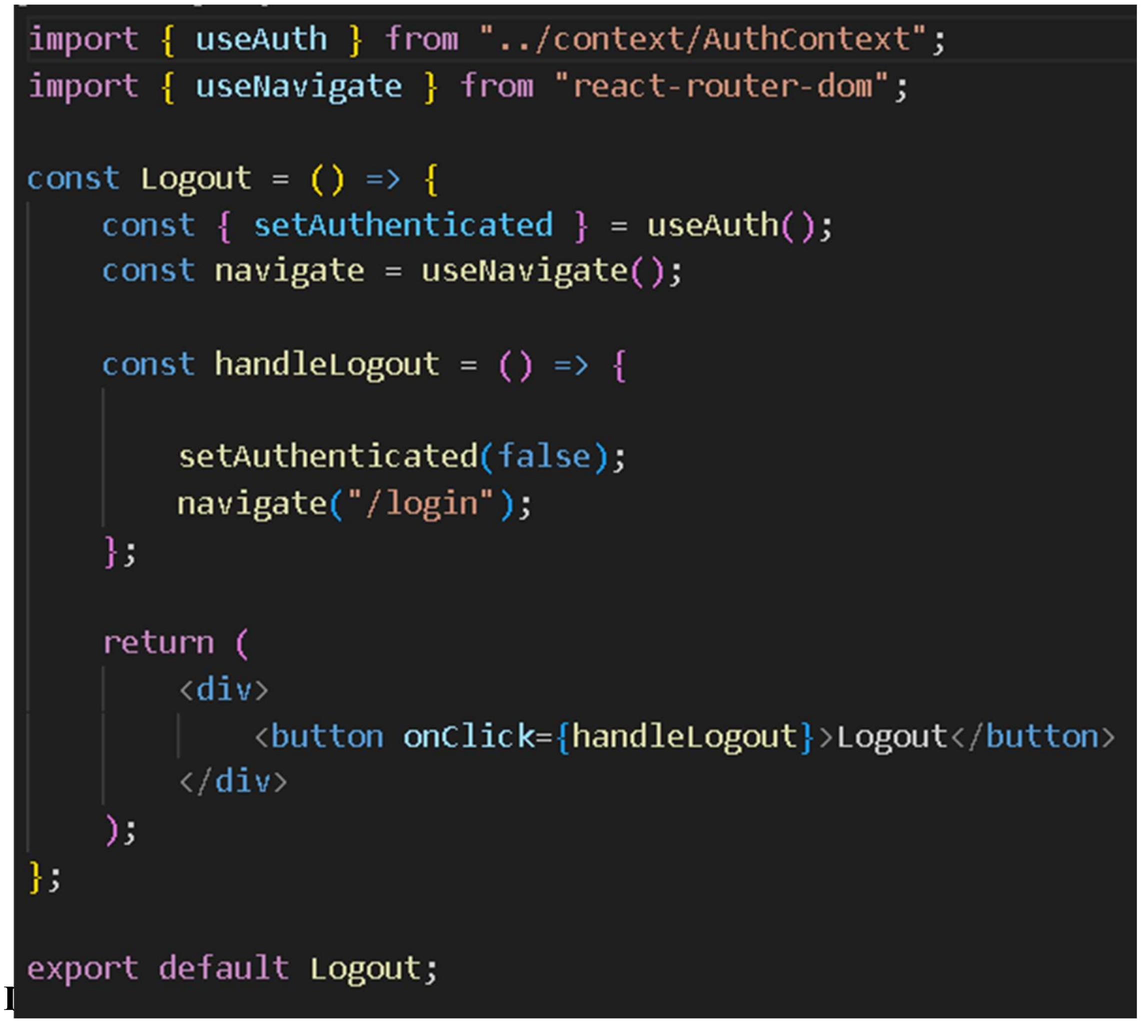
```jsx
import { Link, Outlet } from "react-router-dom";

const DashboardLayout = () => {
    return (
        <>
        <h3>Dashboard</h3>

        <nav>
            <Link to = "summary">Summmary</Link>
            <Link to = "analytics">Analytics</Link>
        </nav>

        <hr />
        <Outlet />
        </>

    )
}

export default DashboardLayout;
```

ProtectedRoute.jsx:



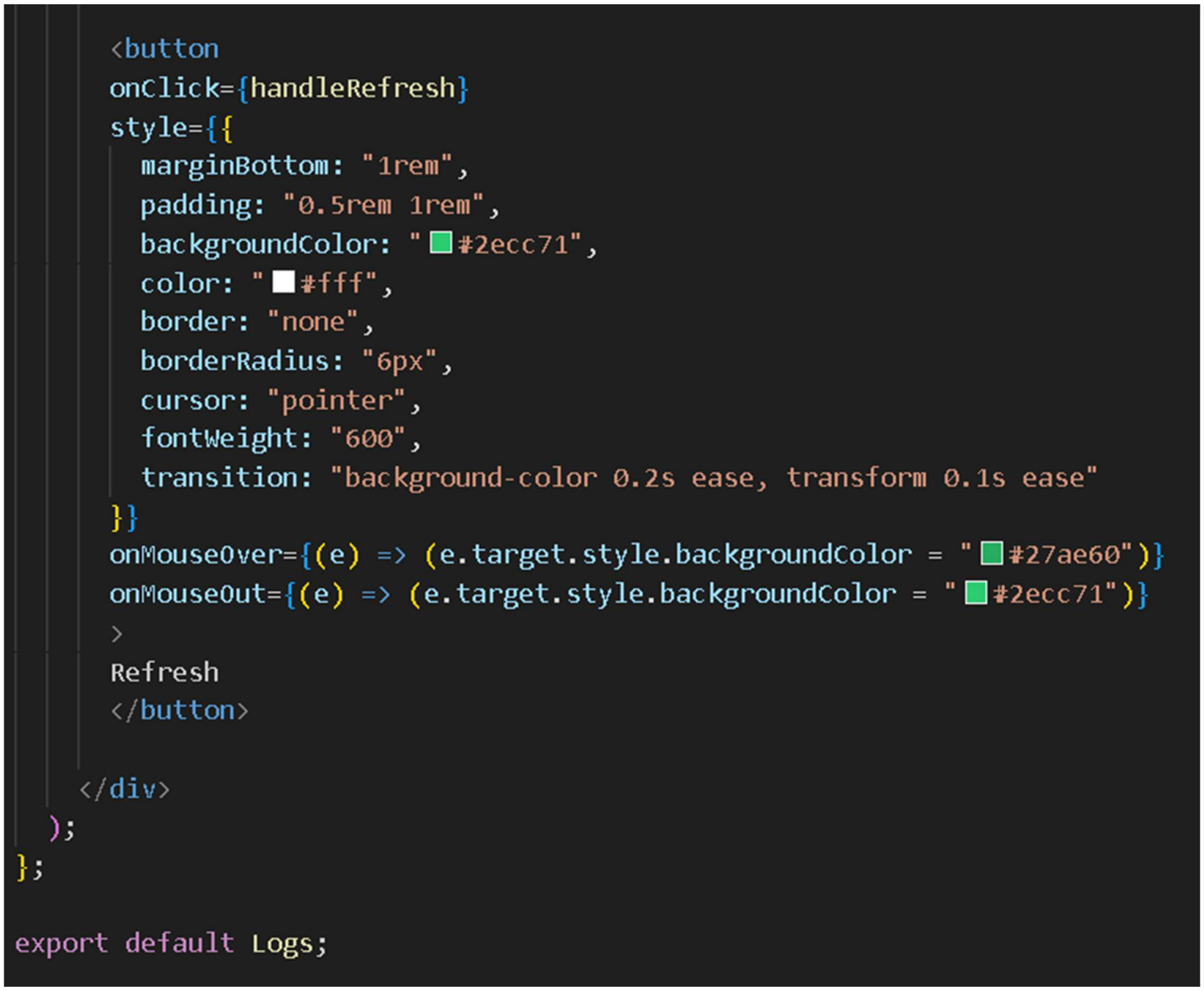
```jsx
import { Navigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import { children } from "react";

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

    if(!isAuthenticated) {
        return <Navigate to = "/login" replace/>
    }
    return children;
}

export default ProtectedRoute;
```

Logout.jsx:



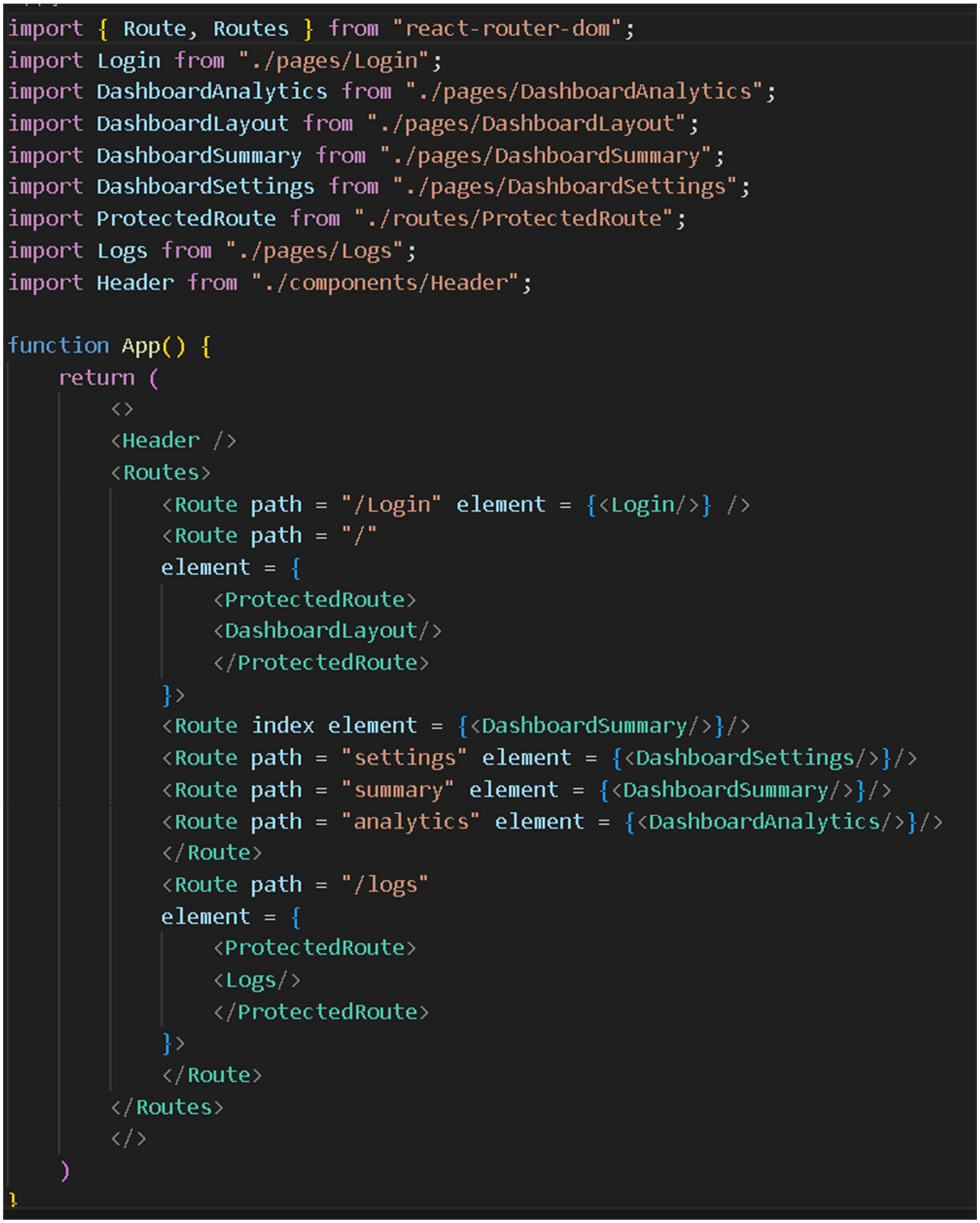
```
import { useAuth } from "../context/AuthContext";
import { useNavigate } from "react-router-dom";

const Logout = () => {
    const { setAuthenticated } = useAuth();
    const navigate = useNavigate();

    const handleLogout = () => {

        setAuthenticated(false);
        navigate("/login");
    };

    return (
        <div>
            <button onClick={handleLogout}>Logout</button>
        </div>
    );
};

export default Logout;
```

Logs.jsx:

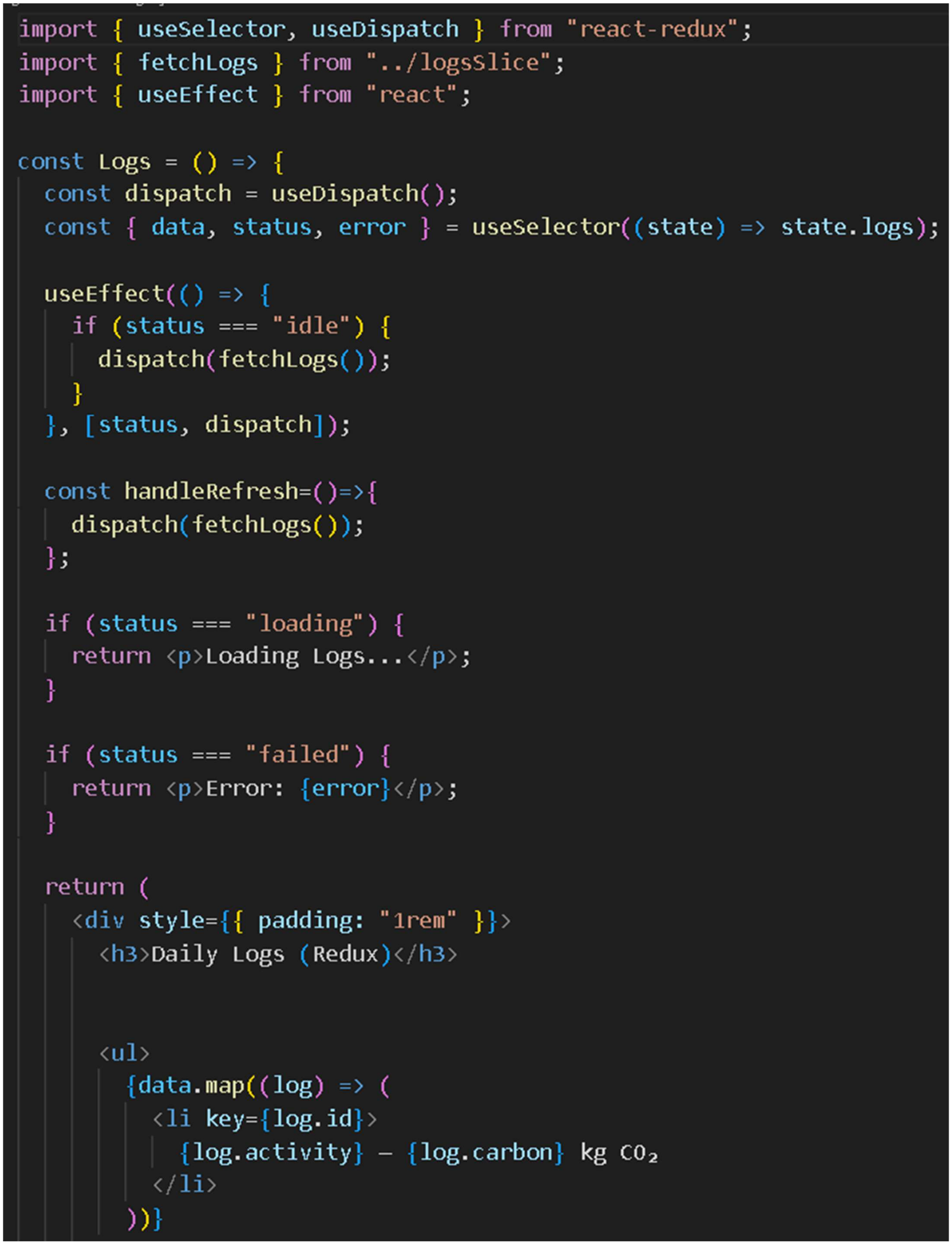


```jsx
      <button
      onClick={handleRefresh}
      style={{
        marginBottom: "1rem",
        padding: "0.5rem 1rem",
        backgroundColor: "#2ecc71",
        color: "#fff",
        border: "none",
        borderRadius: "6px",
        cursor: "pointer",
        fontWeight: "600",
        transition: "background-color 0.2s ease, transform 0.1s ease"
      }}
      onMouseOver={(e) => (e.target.style.backgroundColor = "#27ae60")}
      onMouseOut={(e) => (e.target.style.backgroundColor = "#2ecc71")}
      >
      Refresh
      </button>

    </div>
  );
};

export default Logs;
```

App.jsx:

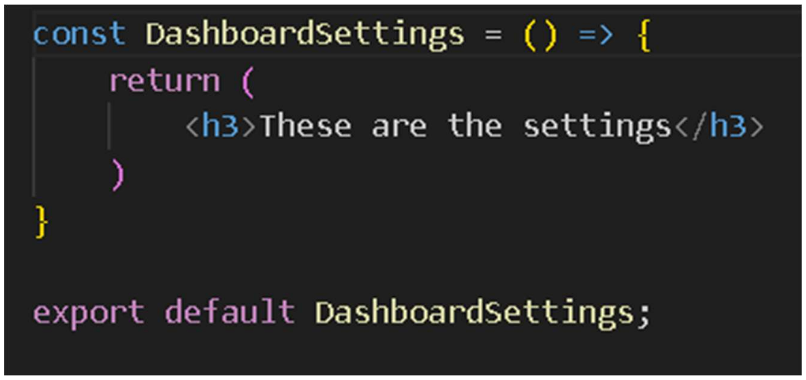


```jsx
import { Route, Routes } from "react-router-dom";
import Login from "./pages/Login";
import DashboardAnalytics from "./pages/DashboardAnalytics";
import DashboardLayout from "./pages/DashboardLayout";
import DashboardSummary from "./pages/DashboardSummary";
import DashboardSettings from "./pages/DashboardSettings";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

function App() {
    return (
        <>
        <Header />
        <Routes>
            <Route path = "/Login" element = {<Login/>} />
            <Route path = "/"
            element = {
                <ProtectedRoute>
                <DashboardLayout/>
                </ProtectedRoute>
            }>
            <Route index element = {<DashboardSummary/>}/>
            <Route path = "settings" element = {<DashboardSettings/>}/>
            <Route path = "summary" element = {<DashboardSummary/>}/>
            <Route path = "analytics" element = {<DashboardAnalytics/>}/>
            </Route>
            <Route path = "/logs"
            element = {
                <ProtectedRoute>
                <Logs/>
                </ProtectedRoute>
            }>
            </Route>
        </Routes>
        </>
    )
}
```



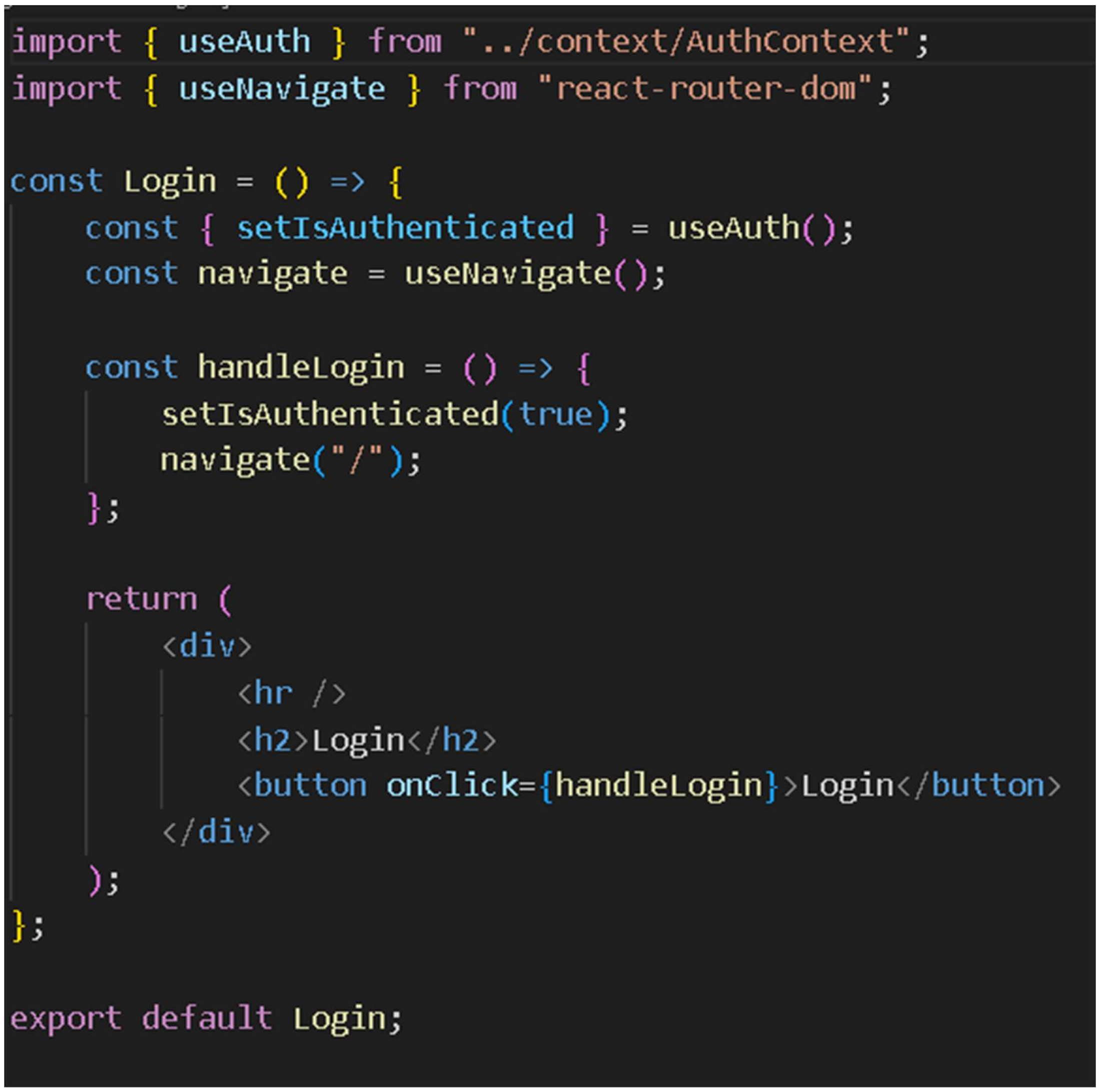
```
import { useSelector, useDispatch } from "react-redux";
import { fetchLogs } from "../logsSlice";
import { useEffect } from "react";

const Logs = () => {
  const dispatch = useDispatch();
  const { data, status, error } = useSelector((state) => state.logs);

  useEffect(() => {
    if (status === "idle") {
      dispatch(fetchLogs());
    }
  }, [status, dispatch]);

  const handleRefresh=()=>{
    dispatch(fetchLogs());
  };

  if (status === "loading") {
    return <p>Loading Logs...</p>;
  }

  if (status === "failed") {
    return <p>Error: {error}</p>;
  }

  return (
    <div style={{ padding: "1rem" }}>
      <h3>Daily Logs (Redux)</h3>


      <ul>
        {data.map((log) => (
          <li key={log.id}>
            {log.activity} – {log.carbon} kg CO₂
          </li>
        ))}
```

DashboardSettings.jsx:



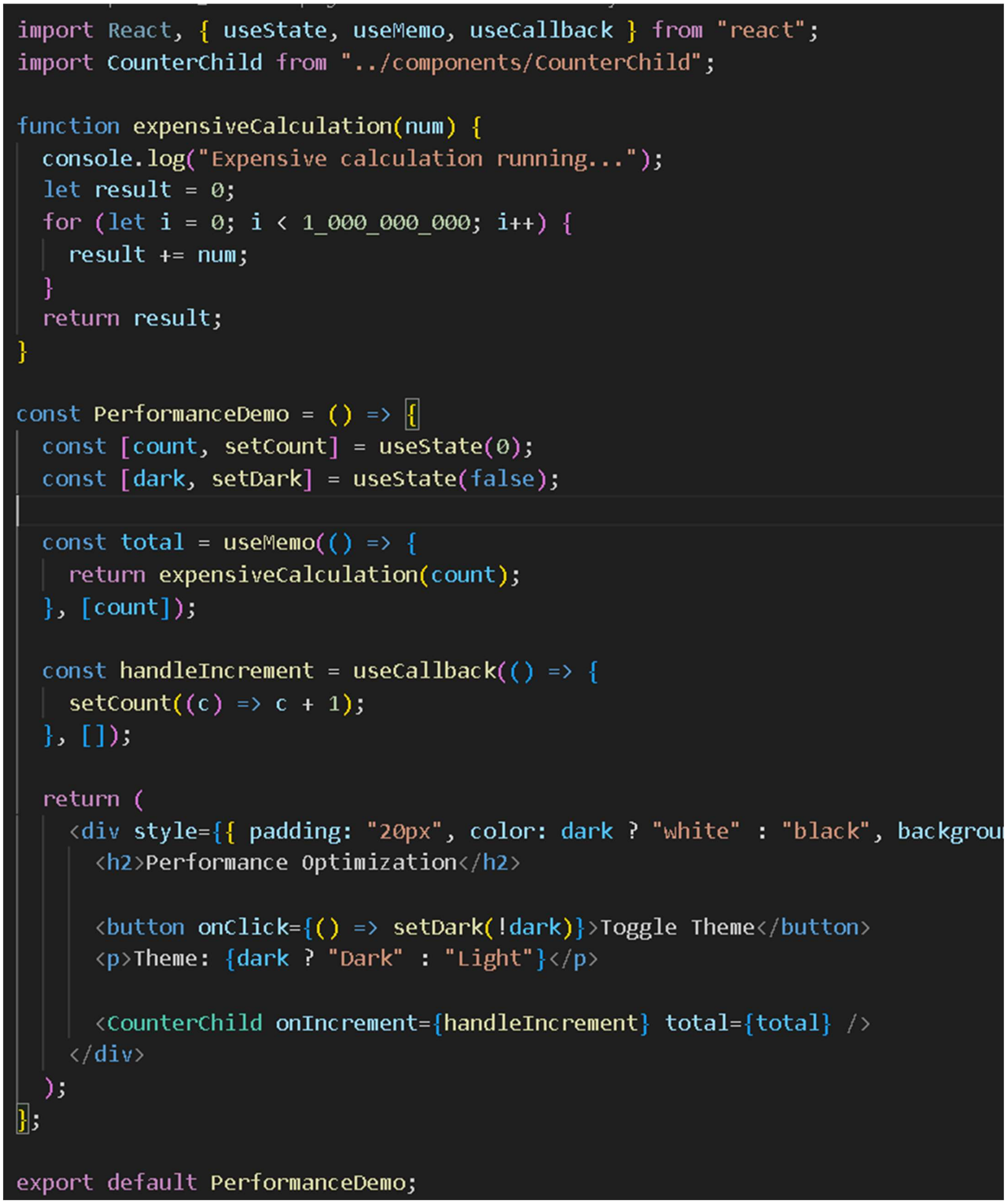
```jsx
const DashboardSettings = () => {
    return (
        <h3>These are the settings</h3>
    )
}

export default DashboardSettings;
```

Login.jsx:



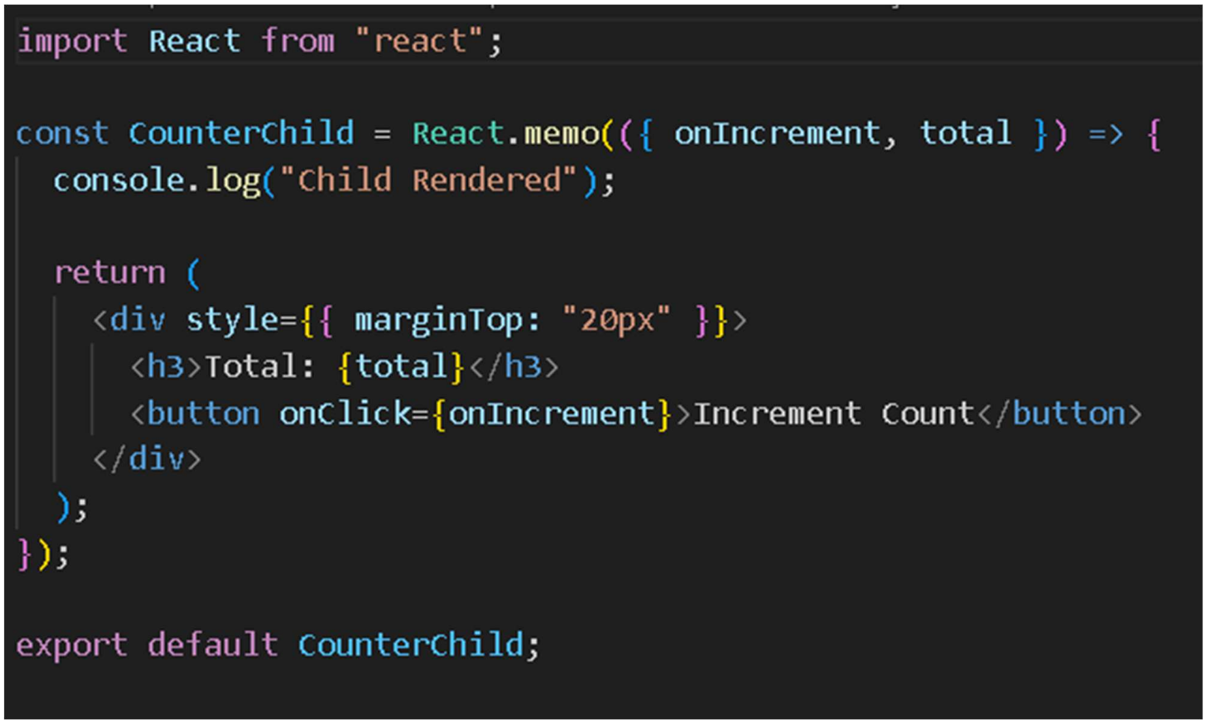
```jsx
import { useAuth } from "../context/AuthContext";
import { useNavigate } from "react-router-dom";

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

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

    return (
        <div>
            <hr />
            <h2>Login</h2>
            <button onClick={handleLogin}>Login</button>
        </div>
    );
};

export default Login;
```

PerformenceDemo.jsx:



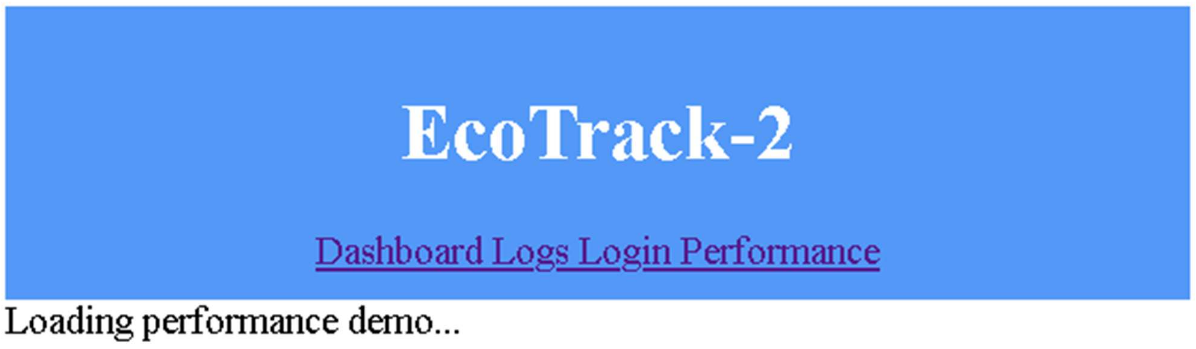
```jsx
import React, { useState, useMemo, useCallback } from "react";
import CounterChild from "../components/CounterChild";

function expensiveCalculation(num) {
  console.log("Expensive calculation running...");
  let result = 0;
  for (let i = 0; i < 1_000_000_000; i++) {
    result += num;
  }
  return result;
}

const PerformanceDemo = () => {
  const [count, setCount] = useState(0);
  const [dark, setDark] = useState(false);

  const total = useMemo(() => {
    return expensiveCalculation(count);
  }, [count]);

  const handleIncrement = useCallback(() => {
    setCount((c) => c + 1);
  }, []);

  return (
    <div style={{ padding: "20px", color: dark ? "white" : "black", backgrou
      <h2>Performance Optimization</h2>

      <button onClick={() => setDark(!dark)}>Toggle Theme</button>
      <p>Theme: {dark ? "Dark" : "Light"}</p>

      <CounterChild onIncrement={handleIncrement} total={total} />
    </div>
  );
};

export default PerformanceDemo;
```

CounterChild.jsx:



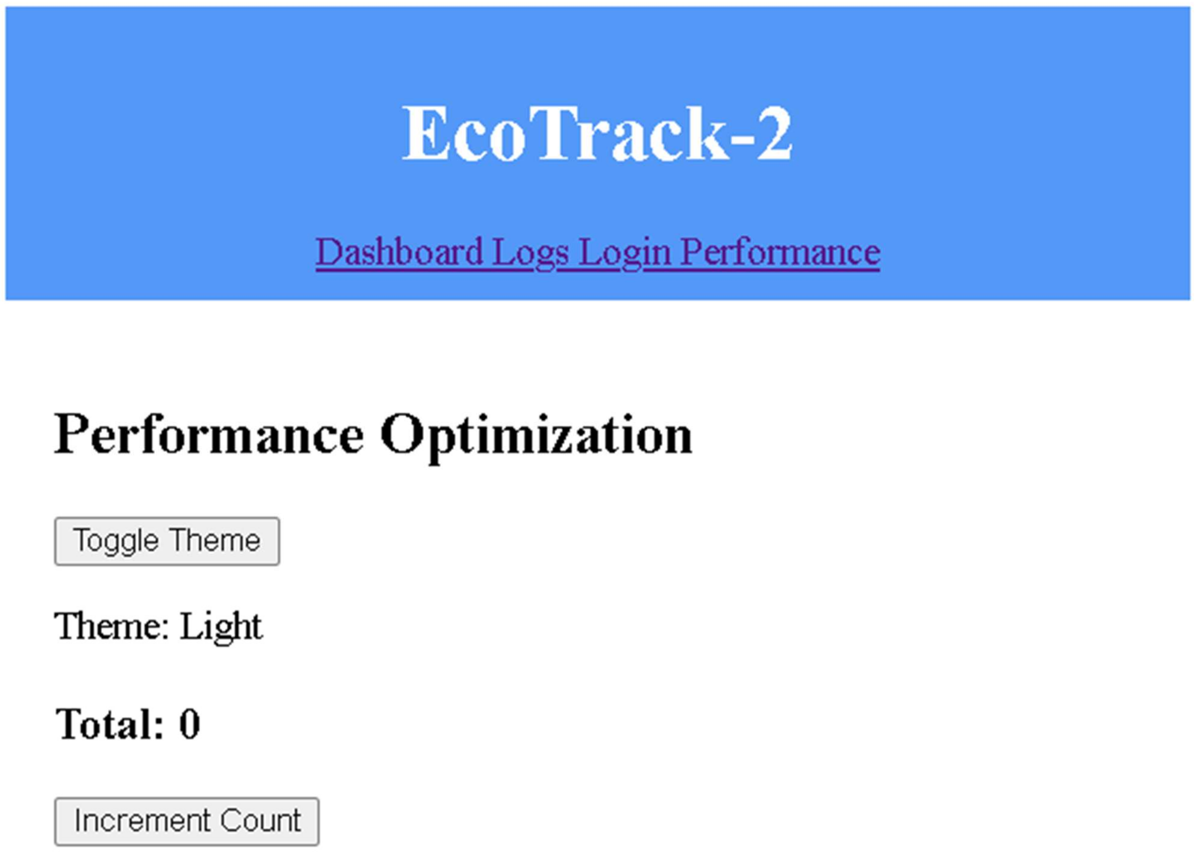
```jsx
import React from "react";

const CounterChild = React.memo(({ onIncrement, total }) => {
  console.log("Child Rendered");

  return (
    <div style={{ marginTop: "20px" }}>
      <h3>Total: {total}</h3>
      <button onClick={onIncrement}>Increment Count</button>
    </div>
  );
});

export default CounterChild;
```

4. Output:


EcoTrack-2

Dashboard Logs Login Performance

Loading performance demo...


EcoTrack-2

Dashboard Logs Login Performance

Performance Optimization

Theme: Light

Total: 0

5. Learning Outcome:

- **Configure and integrate a Redux store in a React application using Redux Toolkit.**
- **Create and manage Redux slices for centralized application state handling.**
- **Implement asynchronous operations using Redux async thunks.**
- **Handle loading, success, and error states during asynchronous data fetching.**
- **Connect React components to Redux state using React-Redux hooks.**
- **Improve user experience by managing refresh actions and responsive async UI feedback.**