



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 4

Student Name: ASHISH

UID: 23BAI70108

Branch: BE CSE AIML

Section/Group: 23AIT-KRG G2

Semester: 6th

Date of Performance: 4 February 2026

Subject Name: Full Stack

Subject Code: 23CSH-382

1. Title:

Redux Toolkit & Asynchronous State Management in EcoTrack

2. 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.

3. Objective:

After completing this experiment and its follow-up tasks, 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

4. Implementation/Code:

- logSlice.jsx:

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";

export const fetchLogs = createAsyncThunk(
  "logs/fetchLogs",
  async() => {
    await new Promise((resolve) => setTimeout(resolve, 1000));

    return [
      { id: 1, activity: "Car Travel", carbon: 4 },
      { id: 2, activity: "Electricity Usage", carbon: 6 },
      { id: 3, activity: "Cycling", carbon: 0 },
    ]
  }
)

const logsSlice = createSlice({
  name : "logs",
  initialState : {
    data : [],
    status : "idle",
    error : null,
  },
  reducers : {},
  extraReducers : (builder) => {
    builder
      .addCase(fetchLogs.pending, (state, action) =>{
        state.status = "loading";
      })
      .addCase(fetchLogs.fulfilled, (state, action) =>{
        state.status = "success";
        state.data = action.payload;
      })
      .addCase(fetchLogs.rejected, (state, action) =>{
        state.status = "failed";
        state.error = action.error.message;
      })
  }
})

export default logsSlice.reducer;
```

- Logs.jsx:

```
import { useEffect, useMemo, useCallback } from "react";
import { useDispatch, useSelector } from "react-redux";
import { fetchLogs } from "../store/logSlice";

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

  const handleRefresh = useCallback(() => {
    dispatch(fetchLogs());
  }, [dispatch]);

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

  const totalCarbon = useMemo(() => {
    return data.reduce((acc, log) => acc + log.carbon, 0);
  }, [data]);

  if(status === "loading"){
    return <p>Loading Logs...</p>;
  }
  if (status === "failed"){
    return <p>Error: {error}</p>;
  }
  return (
    <div>
      <h1>Logs</h1>
      <ol>
        {data.map((log) => (
          <li key={log.id}>
            {log.activity}: {log.carbon} kg CO2
          </li>
        ))}
      </ol>
      <h3>Total Carbon: {totalCarbon} kg CO2</h3>

      <button onClick={handleRefresh}>Refresh</button>
    </div>
  );
};

export default Logs;
```

- Store.jsx:

```
import { configureStore } from "@reduxjs/toolkit";
import logsReducer from "../logSlice"

const store = configureStore({
  reducer: {
    logs: logsReducer,
  },
});

export default store;
```

- AuthContext.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);
```

- ProtectedRoute.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;
```

- Login.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 (
    <>
      <h3>Login</h3>
      <button onClick={handleLogin}>Login</button>
    </>
  )
}

export default Login;
```

- Logout.jsx:

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

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

  useEffect(() => {
    setIsAuthenticated(false);
    navigate("/login");
  }, [setIsAuthenticated, navigate]);

  return <p>Logging you out...</p>;
}

export default Logout;
```

- DashboardSettings, DashboardSummary, DashboardAnalytics:

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

export default DashboardSettings;

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

export default DashboardSummary;

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

export default DashboardAnalytics;
```

- DashboardLayout.jsx:

```
import { Link, Outlet } from "react-router-dom";
import { Container, Button, Stack, Typography } from "@mui/material";

const DashboardLayout = () => {
  return (
    <Container>
      <Typography variant="h4" gutterBottom>
        Dashboard
      </Typography>

      <Stack direction="row" spacing={2} marginBottom={2}>
        <Button variant="contained" component={Link} to="settings">
          Settings
        </Button>

        <Button variant="contained" component={Link} to="summary">
          Summary
        </Button>

        <Button variant="contained" component={Link} to="analytics">
          Analytics
        </Button>
      </Stack>

      <Outlet />
    </Container>
  );
};

export default DashboardLayout;
```

- Header.jsx:

```
import React from "react";
import { AppBar, Toolbar, Typography, Button } from "@mui/material";
import { Link } from "react-router-dom";
import { useAuth } from "../context/AuthContext";

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

  return (
    <AppBar position="static">
      <Toolbar>
        <Typography variant="h6" sx={{ flexGrow: 1 }}>
          EcoTrack
        </Typography>

        <Button color="inherit" component={Link} to="/">
          Dashboard
        </Button>

        <Button color="inherit" component={Link} to="/logs">
          Logs
        </Button>

        {isAuthenticated ? (
          <Button color="inherit" component={Link} to="/logout">
            Logout
          </Button>
        ) : (
          <Button color="inherit" component={Link} to="/login">
            Login
          </Button>
        )}
      </Toolbar>
    </AppBar>
  );
};

export default React.memo(Header);
```

- App.jsx:

```
import { Route, Routes } from "react-router-dom";
import { Suspense, lazy } from "react";
import ProtectedRoute from "../routes/ProtectedRoute";
import Header from "../components/Header";

const Login = lazy(() => import("../pages/Login"));
const Logout = lazy(() => import("../pages/Logout"));
const DashboardLayout = lazy(() => import("../pages/DashboardLayout"));
const DashboardSummary = lazy(() => import("../pages/DashboardSummary"));
const DashboardSettings = lazy(() => import("../pages/DashboardSettings"));
const DashboardAnalytics = lazy(() => import("../pages/DashboardAnalytics"));
const Logs = lazy(() => import("../pages/Logs"));

function App() {
  return (
    <>
      <Header />
      <Suspense fallback={<h2>Loading...</h2>}>
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/logout" element={<Logout />} />
          <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 />} />
        </Routes>
        <Route
          path="/logs"
          element={
            <ProtectedRoute>
              <Logs />
            </ProtectedRoute>
          }
        />
      </Suspense>
    </>
  );
}

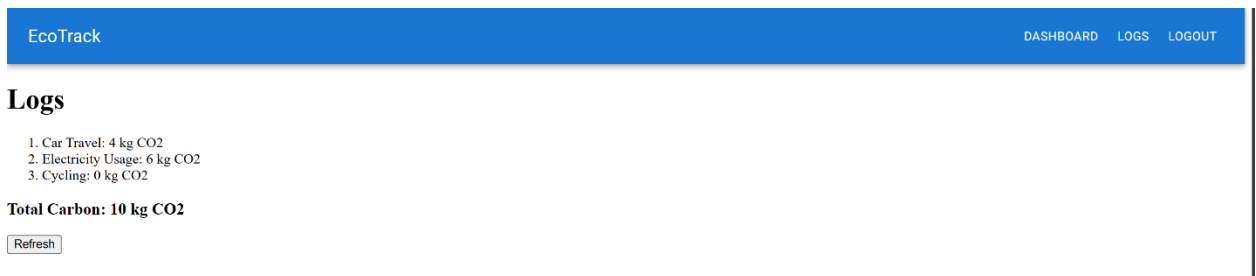
export default App;
```


5. Output



Login

Login



6. Learning Outcome

- We learnt about React Apps and how to create them.
- We learnt about redux and it's components.
- We learnt about the use of thunks and Slices.
- We learnt about Authentication and index pages
- We learnt the use of MaterialUI and useSelector.
- We learnt about the flow of a React project.