Experiment 3

Name : Astik Joshi **UID :** 23BCS10627
Branch: BE-CSE **Section/Group:** KRG-3B
Semester: 6th **Date of Performance:** -02-02-2026
Subject Name: Full Stack-II **Subject Code:** 23CSH-309

1. **Aim:** The aim of this implementation is 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..

2. Objective:-

- Configured a Redux store in a React application using Redux Toolkit
- Created and integrated Redux slices for managing application data
- Implemented asynchronous operations using Redux async thunks
- Managed loading, success, and error states during API calls
- Connected React components to Redux state using React-Redux hooks

3. Implementation/Code:

The **EcoTrack application** uses **Redux Toolkit** for centralized state management, ensuring predictable data flow and easier debugging. A global Redux store is configured using configureStore with feature-based slices like logsSlice, while asynchronous operations are handled using **Redux Async Thunks** to manage API request states. **React-Redux hooks** (useDispatch, useSelector) connect components to the store, enabling data fetching and state access. **Selectors** derive filtered views from the Redux state without mutation, and loading and error states are managed to improve performance and user experience.

- **/store/logSlice.js :-**

```
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) => {
                state.status = 'loading';
            })
            .addCase(fetchLogs.fulfilled, (state, action) => {
                state.status = 'succeeded';
                state.data = action.payload;
            })
            .addCase(fetchLogs.rejected, (state, action) => {
                state.status = 'failed';
                state.error = action.error.message;
            });
    },
});
export default logsSlice.reducer;
```

- **store.js :-**

```
import {configureStore} from '@reduxjs/toolkit';
import logsReducer from './logsSlice';
const store = configureStore({
    reducer: {
        logs: logsReducer,
    },
});
export default store;
```

- **LogSlice.jsx :-**

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

export const fetchLogs =
createAsyncThunk("logs/fetchLogs", async () => {
  return new Promise((resolve) =>
    setTimeout(
      () =>
        resolve([
          { id: 1, activity: "Commute", carbon: 2.5 },
          { id: 2, activity: "Lunch", carbon: 0.5 },
        ]),
      1000
    )
  );
});

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

```
export default logSlice.reducer;
```

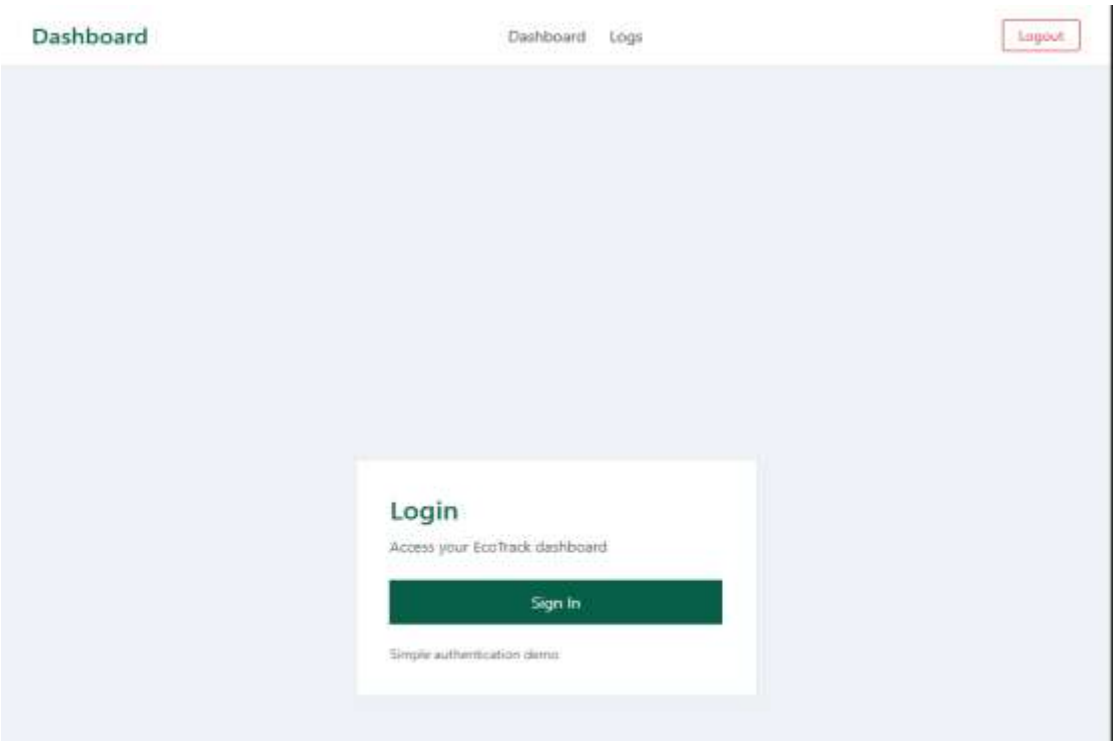
- **App.jsx:-**

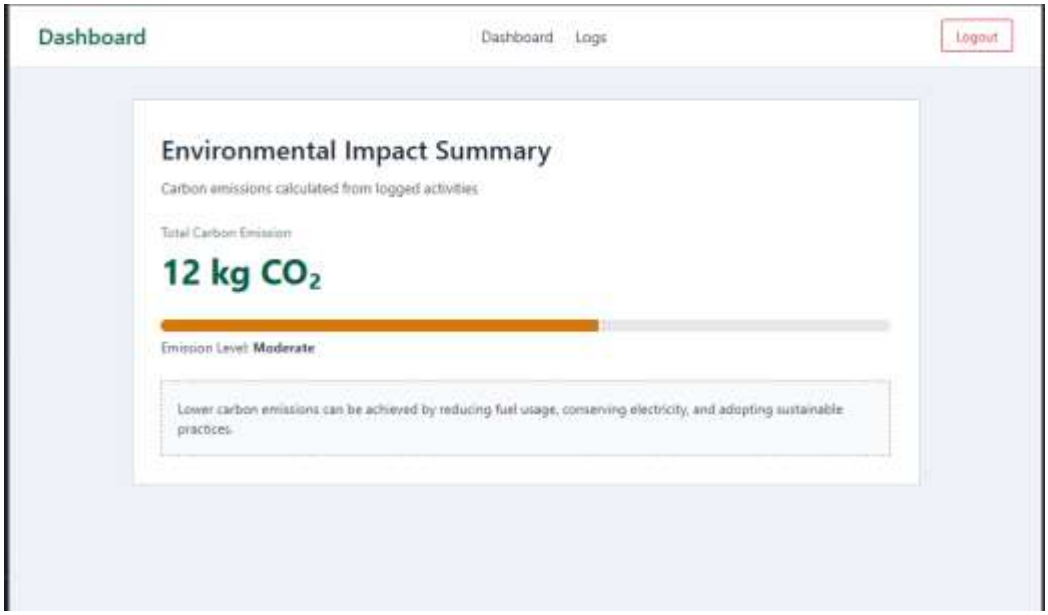
```jsx
import Dashboard from "./pages/dashboard.jsx";
import Log from "./pages/log.jsx";
import Header from "./components/Header.jsx";
import { Routes, Route, Link, BrowserRouter } from "react-router-dom";
import Login from "./pages/login.jsx";
import ProtectedRoute from "./Routes/ProtectedRoute.jsx";
import DashboardLayout from "./pages/dashboardLayout.jsx";

function App() {
  return (
    <>
      <BrowserRouter>
        <Header title="Dashboard" />
        <main>
          <Routes>
            <Route path="/login" element={<Login />} />
            <Route
              path="/dashboardLayout"
              element={
                <ProtectedRoute>
                  <DashboardLayout />
                </ProtectedRoute>
              }
            />
            <Route
              path="/"
              element={
                <ProtectedRoute>
                  <Dashboard />
                </ProtectedRoute>
              }
            />
            <Route
              path="/log"
              element={
                <ProtectedRoute>
                  <Log />
                </ProtectedRoute>
              }
            />
          </Routes>
        </main>
      </BrowserRouter>
    </>
  );
}

export default App;
```

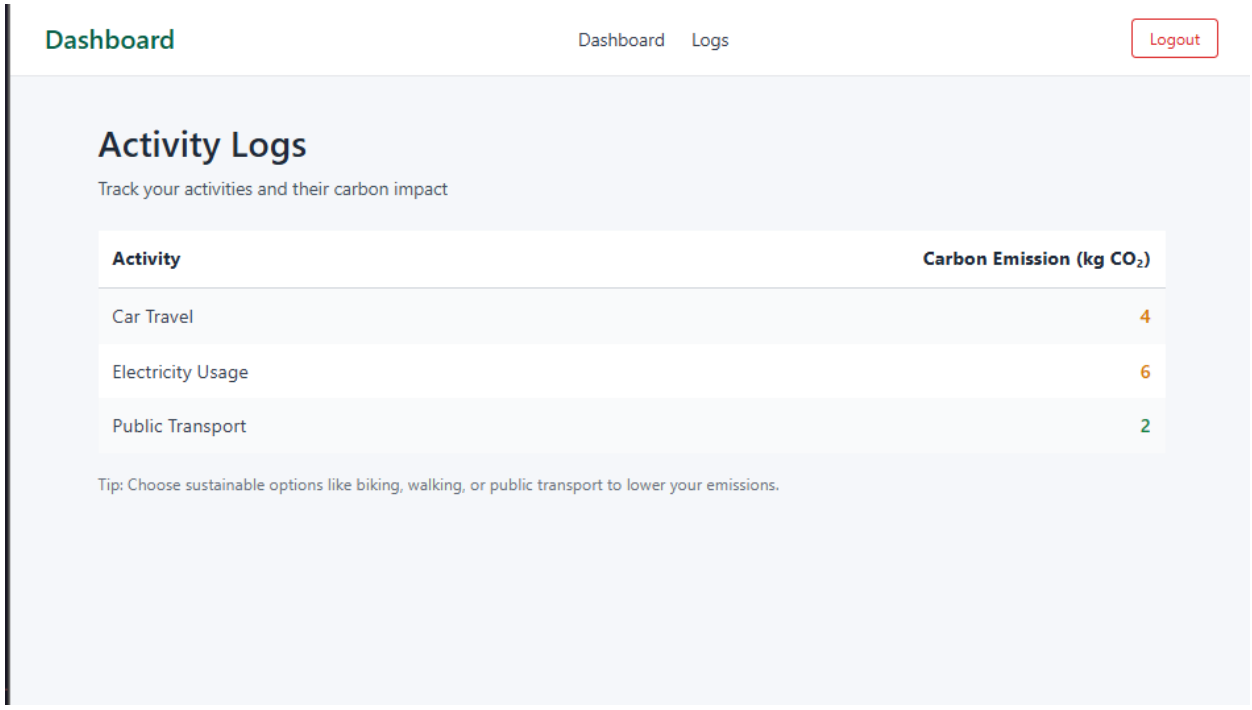
- ProtectRoute.jsx

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

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

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

4. Output




Dashboard
Dashboard
Logs
Logout
Environmental Impact Summary
Carbon emissions calculated from logged activities
Total Carbon Emission
12 kg CO_2
Emission Level: Moderate
Lower carbon emissions can be achieved by reducing fuel usage, conserving electricity, and adopting sustainable practices.


Dashboard
Dashboard
Logs
Logout
Activity Logs
Track your activities and their carbon impact
Activity
Carbon Emission (kg CO_2)
Car Travel
4
Electricity Usage
6
Public Transport
2
Tip: Choose sustainable options like biking, walking, or public transport to lower your emissions.

5. Learning Outcome :-

- Learned how to configure a Redux store and create feature-based slices
- Understood the use of async thunks for managing asynchronous operations
- Gained experience in handling loading, success, and error states in Redux
- Learned to derive filtered views using selectors without mutating the global state