DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Report File FULL STACK

Student Name: Sukhjinder Singh

UID: 23BAI70078

Branch: BE-AIT-CSE

Section/Group: 23AIT-KRG-G2

Semester: 5th

Subject Code: 23CSP-339

Subject Name: Full Stack

Aim:

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.

Objectives:

After completing this experiment and its follow-up task, the student will be able to:

1. Configure a Redux store in a React application using Redux Toolkit 2.
Create and integrate Redux slices for managing application data
3.Implement asynchronous actions using Redux async thunks.
4. Manage loading, success, and error states during asynchronous operations
5. Connect React components to Redux state using React-Redux hooks
6. Trigger asynchronous data fetching through Redux actions from UI components
7. Use Redux state to derive filtered views without modifying the global store
8. Enhance user experience by handling refresh actions and improving async UI feedback

Hardware Requirements:

- Processor: Intel i5/Ryzen 5 or higher
- RAM: 8GB minimum
- Display: 1920x1080 resolution Software Requirements:
- Node.js v18+
- React.js v18+
- VS Code with ES7+ extensions
- JSON Server (for mock API)

Code Implementation:

App.js

```
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>
      </>
   )
}  export default App;
```

logsSlice.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 },
      { id: 4, activity: "Public Transport", carbon: 12 },
      { id: 5, activity: "Meat Consumption", carbon: 5 },
      { id: 6, activity: "Plant-based Meal", carbon: 2 },
      { id: 7, activity: "Air Travel", carbon: 1 },
  ];
 }
); 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;
```

store.jsx

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

Expected Output:

- Learn’t about reduxjs.
- Learn’t about its implications.
- Learn’t how to implement the redux.