

## Experiment 3

**Student Name:** Nitin

**Branch:** BE CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** Full Stack Development

**UID:** 23BCS122270

**Section/Group:** KRG 3A

**Date of Performance:** 27/01/26

**Subject Code:** 23CSH-309

### 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.

### Objective:

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

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

### Implementation/Code:

#### logsSlice.js:

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { logs as logsData } from "../data/logs";
```

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



# CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

```
);

const logsSlice = createSlice({
  name: "logs",
  initialState: {
    data: [],
    loading: false,
    error: null,
  },
  reducers: { },
  extraReducers: builder => {
    builder
      .addCase(fetchLogs.pending, state => {
        state.loading = true;
        state.error = null;
      })
      .addCase(fetchLogs.fulfilled, (state, action) => {
        state.loading = false;
        state.data = action.payload;
      })
      .addCase(fetchLogs.rejected, state => {
        state.loading = false;
        state.error = "Failed to fetch logs";
      });
  },
});

export default logsSlice.reducer;
```

### store.js:

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

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

## Main.jsx:

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import { AuthProvider } from "./context/AuthContext";
import { Provider } from "react-redux";
import { store } from "./redux/store";
import "./index.css";
```

```
ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <Provider store={store}>
      <BrowserRouter>
        <AuthProvider>
          <App />
        </AuthProvider>
      </BrowserRouter>
    </Provider>
  </React.StrictMode>
);
```

## Output:



**Dashboard**

Home Overview Reports Logout

---

**Total Activities**

- Car Travel: 4 Kg
- Electricity Usage: 6 Kg
- Cycling: 0 Kg

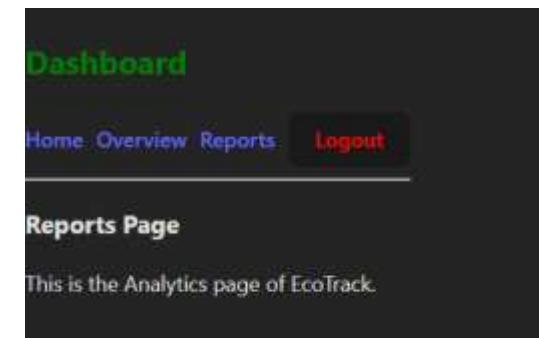
**High Carbon (> 4 Kg)**

- Electricity Usage

**Low Carbon ( $\leq 4$  Kg)**

- Car Travel
- Cycling

Refresh Logs



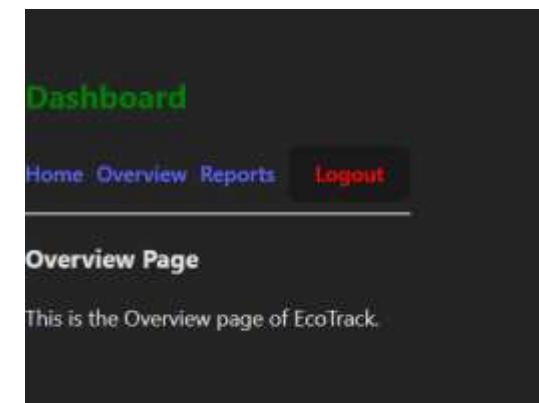
**Dashboard**

Home Overview Reports Logout

---

**Reports Page**

This is the Analytics page of EcoTrack.



**Dashboard**

Home Overview Reports Logout

---

**Overview Page**

This is the Overview page of EcoTrack.



## **Learning Outcome:**

- Configured and integrated a Redux store in a React application using Redux Toolkit.
- Created Redux slices to manage centralized application state efficiently.
- Implemented asynchronous data fetching using Redux createAsyncThunk.
- Handled loading and error states to improve user experience during async operations.
- Connected React components to Redux state using React-Redux Provider and hooks.