DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Experiment 3

Student Name: Alisha Ameer UID: 23BCC70017

Branch: AIT-CSE Section: 23AIT_KRG-2 A

Semester: 6 Date of Performance: 30/01/2026

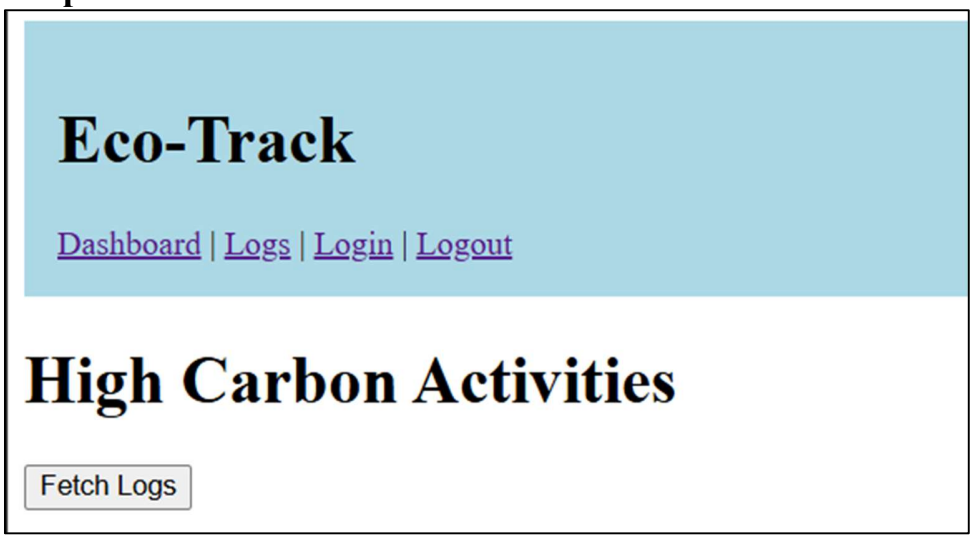
Subject Name: Full Stack II Subject Code: 23CSH-382

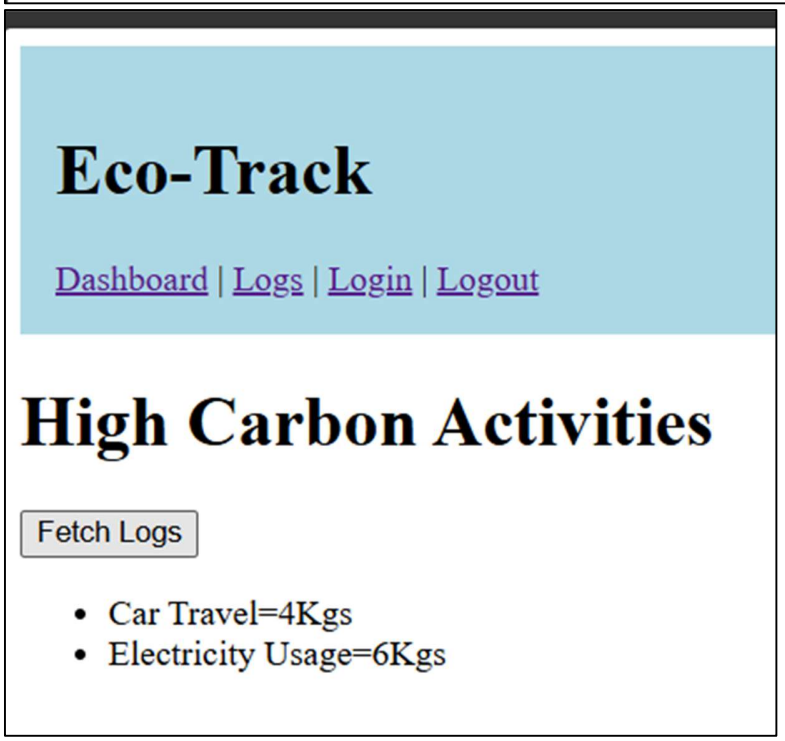
- **Aim:**
  The aim of this experiment was to upgrade the existing EcoTrack application by replacing local component state with Redux Toolkit-based centralized state management and implementing asynchronous data fetching for carbon activity logs.
- **Objectives:**
  The main objectives of this experiment are as follows:
  1. To understand the need for centralized state management in larger React applications.
  2. To learn how to create and manage state using Redux Toolkit.
  3. To gain practical experience in handling asynchronous operations in Redux.
  4. To understand different API request states (loading, success, failure) and how to manage them in the UI.
  5. To practice connecting React components with a Redux store using hooks.

- **Implementation:**
  The following key steps were followed to implement Redux Toolkit in the EcoTrack application:
  1. Redux Toolkit was installed and integrated into the existing React project to replace local state management.
  2. A logSlice.js file was created to define the initial state, including logs data, request status, and error handling.
  3. An asynchronous thunk fetchLogs was implemented using createAsyncThunk to simulate fetching activity data with a network delay.
  4. The slice was configured with extraReducers to handle pending, fulfilled, and rejected states of the async request.
  5. A centralized Redux store was created using configureStore, and the application was wrapped with <Provider> in main.jsx.

6. The Logs.jsx component used useSelector to read data from the Redux store and useDispatch to trigger data fetching.
7. Conditional rendering was added to display loading messages or errors, and data was filtered to show only high-carbon activities (≥ 4 kg).

- **Output:**





- **Results:**
  Redux Toolkit was successfully integrated into the EcoTrack application for managing activity log data. Asynchronous data fetching functioned correctly with clear handling of loading and error states. The Redux store provided a structured and predictable way to manage shared state, improving the organization and maintainability of the application.

- **Learning Outcomes:**
  After completing this experiment, I have learnt to:
  1. Replace local state with Redux Toolkit for better state management.
  2. Create slices, actions, and reducers using createSlice.
  3. Handle asynchronous operations using createAsyncThunk.
  4. Manage different request states using extraReducers.
  5. Connect React components to Redux using useSelector and useDispatch.
  6. Design a more scalable and maintainable frontend architecture