



### Experiment 3

**Student Name: Vishal Saini**

**Branch: CSE**

**Semester: 6th**

**Subject Name: Full Stack Development – II**

**UID: 23BCS10163**

**Section/Group: KRG 3-B**

**Date of Performance: 02/02/2026**

**Subject Code: 23CSH-309**

**1. Aim:** To implement **global state management** in the EcoTrack application using **React Redux (Redux Toolkit)** for managing daily environmental activity logs, including fetching data asynchronously, adding new logs, and removing existing logs.

### **2. Objective:**

- To understand the concept of **state management using Redux**
- To implement **Redux Toolkit** for simplified Redux configuration
- To manage application-wide state using a **centralized Redux store**
- To implement **async data fetching** using `createAsyncThunk`
- To add and delete data from the Redux store using reducers
- To integrate Redux with React using `useDispatch` and `useSelector`
- To simulate real-time environmental activity logging
- To understand the separation of UI logic and state logic

### **3. Implementation / Code:**

#### **Tools & Technologies Used:-**

- React.js
- Redux Toolkit
- React Redux
- JavaScript (ES6)
- VS Code
- Web Browser (Google Chrome / Firefox)


#### **Implementation Description:-**

- 1) The EcoTrack application uses Redux Toolkit to manage daily carbon emission logs globally.
- 2) A Redux store is configured using `configureStore`, and a logs slice is created using `createSlice`.

- 3) Asynchronous data fetching is implemented using createAsyncThunk to simulate an API call that loads initial activity logs.
- 4) The Logs component uses:
- 5) useSelector to access logs data and loading status from the Redux store
- 6) useDispatch to dispatch actions such as fetching logs, adding logs, and removing logs
- 7) Users can:
- 8) Add new activities with carbon emission values
- 9) View the list of logged activities
- 10) Delete any activity from the list
- 11) This approach ensures predictable state updates, better scalability, and cleaner code organization.

### Sample Code Snippet:-

 App.jsx U X

experiment-3-redux > ecotrack > src >  App.jsx > [🔍] default

```
1  import Logs from './pages/Logs';
2
3
4  function App() {
5    return (
6      <div>
7        <h1>EcoTrack (Experiment 3)</h1>
8        <Logs />
9      </div>
10   );
11 }
12
13 export default App;
14
```

JS store.js U X

experiment-3-redux > ecotrack > src > store > JS store.js > ...

```
1 import { configureStore } from '@reduxjs/toolkit';
2 import logsReducer from '../logsSlice';
3
4 export const store = configureStore({
5   reducer: {
6     logs: logsReducer,
7   },
8 });
9
```

Logs.jsx U X

experiment-3-redux > ecotrack > src > pages > Logs.jsx > Logs

```
1 import { useEffect, useState } from 'react';
2 import { useDispatch, useSelector } from 'react-redux';
3 import { fetchLogs, addLog, removeLog } from '../store/logsSlice';
4
5 const Logs = () => {
6   const dispatch = useDispatch();
7
8   const { data: logs, status, error } = useSelector(
9     (state) => state.logs
10  );
11
12   const [activity, setActivity] = useState('');
13   const [carbon, setCarbon] = useState('');
14
15   useEffect(() => {
16     if (status === 'idle') {
17       dispatch(fetchLogs());
18     }
19   }, [status, dispatch]);
20
21   const handleSubmit = (e) => {
22     e.preventDefault();
23     if (!activity || !carbon) return;
24
25     dispatch(
26       addLog({
27         id: Date.now(),
28         activity,
29         carbon: Number(carbon),
```

```
JS logsSlice.js U X
experiment-3-redux > ecotrack > src > store > JS logsSlice.js > ...
1  import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
2
3  /**
4   * ASYNC THUNK
5   */
6  export const fetchLogs = createAsyncThunk(
7    'logs/fetchLogs',
8    async () => {
9      // simulate API call
10     await new Promise((resolve) => setTimeout(resolve, 500));
11
12     return [
13       { id: 1, activity: 'Car Travel', carbon: 4 },
14       { id: 2, activity: 'Electricity Usage', carbon: 6 },
15       { id: 3, activity: 'Cycling', carbon: 0 },
16     ];
17   }
18 );
19
20 const initialState = {
21   data: [],
22   status: 'idle', // idle | loading | succeeded | failed
23   error: null,
24 };
25
26 const logsSlice = createSlice({
27   name: 'logs',
28   initialState,
29   reducers: {
30     addLog(state, action) {
31       state.data.push(action.payload);
32     },
33     removeLog(state, action) {
34       state.data = state.data.filter(
35         (log) => log.id !== action.payload
36       );
37     },
38   },
39   extraReducers: (builder) => {
40     builder
41       .addCase(fetchLogs.pending, (state) => {
42         state.status = 'loading';
43       })
44       .addCase(fetchLogs.fulfilled, (state, action) => {
45         state.status = 'succeeded';
```

## 4. Output:

- The EcoTrack application successfully loads daily activity logs using Redux
- Logs are fetched asynchronously and displayed dynamically
- Users can add new environmental activities with carbon values



- Users can remove existing logs instantly
- State updates occur without page reload
- Redux ensures centralized and predictable state management
- UI remains responsive and synchronized with the store

## EcoTrack (Experiment 3)

### Daily Logs (Redux)

Activity	Carbon (kg CO <sub>2</sub> )	Add Log
----------	------------------------------	---------

- Car Travel — 4 kg CO<sub>2</sub>
- Electricity Usage — 6 kg CO<sub>2</sub>
- Cycling — 0 kg CO<sub>2</sub>

## 5. Learning Outcomes (What I Have Learnt):

- Understand Redux architecture and data flow
- Implement Redux Toolkit for efficient state management
- Use createAsyncThunk for asynchronous operations
- Manage global state using Redux store and slices
- Integrate Redux with React using hooks
- Perform add and delete operations on centralized state
- Build scalable and maintainable React applications
- Differentiate between Context API and Redux usage