# Lab Activity: Redux and Authentication in a Todo App

Course: Software Systems Development

## Objective

The goal of this lab is to implement **Redux state management** and **Authentication** in a Todo application. By the end of this activity, you will:

- Use Redux for global state management of todos and authentication status.
- Implement authentication (signin/signout) using JWT and cookies.
- Connect authenticated users with their own todos.

## Prerequisites

Students are expected to already have:

- A working React Todo application (add, delete, list todos).
- Basic knowledge of Redux Toolkit.
- Basic knowledge of Express.js backend and JWT authentication.

## Tasks

Complete the following tasks. **Do not change the API endpoints or state structure** as these will be tested automatically.

### Task 1: Setup Redux Store (20 marks)

1. Create a Redux store using `@reduxjs/toolkit`.
2. Add two slices:
   - **authSlice** with state `{ token: string`null, user: object—null —
   - **todoSlice** with state `{ todos: [] }`
3. Wrap the App component with `<Provider store={store}>`.

### Task 2: Authentication (30 marks)

1. Implement a `signin` component that:
   - Sends a POST request to `/signin` with email and password.
   - On success, stores the JWT in cookies (`httpOnly`) and updates Redux `authSlice`.
2. Implement a `logout` button that:
   - Calls `/logout` endpoint.
   - Clears Redux `authSlice`.

### Task 3: Protected Todo Features (30 marks)

1. Modify the `TodoList` component so it:
   - Fetches todos from `/todos` (server must check JWT from cookies).
   - Saves the todos in Redux `todoSlice`.
2. Implement `add todo` using POST `/todos`.
3. Implement `delete todo` using DELETE `/todos/:id`.

### Task 4: UI Integration (20 marks)

1. If the user is not authenticated, show only the `signin` page.
2. Once authenticated, show the todo app linked to their account.

## Submission Instructions

- Submit your project as a zipped folder named `rollnumber_todoapp.zip`.
- The backend must expose the following routes:

```
POST   /signin   -> { token }
POST   /logout   -> {}
GET    /todos    -> [ { id, title } ]
POST   /todos    -> { id, title }
DELETE /todos/:id -> {}
```

- The frontend Redux state shape must strictly match the given slices.

## Marking Scheme (100 Marks Total)

| Task | Marks |
|---|---|
| Redux store + slices implemented correctly | 20 |
| Signin + Logout flow working with JWT cookies | 30 |
| Fetching, adding, deleting todos with authentication | 30 |
| Conditional UI rendering (signin vs todos) | 20 |

## Automated Testing Notes (Instructor Only)

- A test script will:
  1. Run `npm start` on student frontend and `npm run dev` on backend.
  2. Perform signin with test credentials, check if Redux state updates with token.
  3. Create 2 todos, delete 1, and fetch final list.
  4. Verify Redux `todoSlice.todos` matches backend response.
  5. Verify logout clears `authSlice`.
- Students who change API routes or Redux state shape will fail tests automatically.