WEB APP DEVELOPMENT WITH REACTJS (INT252)

Lecture 20: Redux Toolkit & React Integration (Basics)

Unit V – State Management & Advanced React

Syllabus Mapping:

- Redux Toolkit
- Store configuration
- Slices, actions, reducers
- React–Redux integration

Target Learner: Beginner → Intermediate

Why Redux Toolkit?

Classic Redux problems:

- Too much boilerplate
- Hard for beginners

Redux Toolkit (RTK):

- Official
- Simplified
- Recommended by Redux team

What Redux Toolkit Provides

- `configureStore`
- `createSlice`
- Built-in best practices

Less code, fewer bugs ✅

Libraries Needed

```
npm install @reduxjs/toolkit react-redux
```

Redux Toolkit Architecture

```
Slice → Reducer → Store → React App
```

Step 1: Create a Slice

A **slice** contains:

- State
- Reducers
- Actions

Creating a Counter Slice

```
// features/counterSlice.js
import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({
  name: 'counter',
  initialState: { value: 0 },
  reducers: {
    increment: (state) => {
      state.value += 1;
    },
    decrement: (state) => {
      state.value -= 1;
    }
  }
});

export const { increment, decrement } = counterSlice.actions;
export default counterSlice.reducer;
```

Why This Looks Like Mutation

Redux Toolkit uses **Immer**:

- Allows safe mutation syntax
- Internally creates immutable updates

Step 2: Create the Store

```
// store.js
import { configureStore } from '@reduxjs/toolkit';
import counterReducer from './features/counterSlice';

export const store = configureStore({
  reducer: {
    counter: counterReducer
  }
});
```

Step 3: Provide Store to React

```
// main.jsx
import { Provider } from 'react-redux';
import { store } from './store';

<Provider store={store}>
  <App />
</Provider>
```

Step 4: Access State (useSelector)

```
import { useSelector } from 'react-redux';

const count = useSelector(state => state.counter.value);
```

Step 5: Dispatch Actions (useDispatch)

```
import { useDispatch } from 'react-redux';
import { increment } from './features/counterSlice';

const dispatch = useDispatch();
```

Full Counter Component

```
function Counter() {
  const count = useSelector(state => state.counter.value);
  const dispatch = useDispatch();

  return (
    <>
      <h1>{count}</h1>
      <button onClick={() => dispatch(increment())}>+</button>
    </>
  );
}
```

Redux Toolkit Data Flow

```
UI → dispatch(action)
   ↓
Reducer updates store
   ↓
useSelector reads state
   ↓
UI updates
```

Common Beginner Mistakes

❌ Forgetting Provider
❌ Wrong state path
❌ Mutating state outside slice

When to Use Redux Toolkit

✅ Medium to large apps
✅ Complex global state

Practice Exercises

1. Create todo slice
2. Add addTodo action
3. Read todos using useSelector

✅ Answers – Practice (Conceptual)

- Slice manages todos
- Action adds todo
- Selector reads todos

Key Takeaways

- Redux Toolkit simplifies Redux
- Slices manage state logic
- React connects via Provider

Next Lecture

Lecture 21: Debugging, Error Handling & Best Practices

Unit VI – Debugging, Testing & Deployment