Chapter 15: State Management with Redux

1 Introduction

Is chapter mein humne Redux Toolkit se state management seekhi, jo **EduCampusHustle** MVP ke liye cart aur user data manage karne ke liye zaroori hai.

2 15.1 Why State Management?

- Local state limits jab multiple components data share karte hain.
- Redux ka fayda: Centralized state, predictable updates.
- Example: Cart items, user login state.

3 15.2 Redux Basics

- **Store**: Global state ka centralized place.
- **Actions**: State change instructions.
- **Reducers**: State update logic.
- **Dispatch**: Actions ko store mein bhejta hai.

4 15.3 Setting Up Redux Toolkit

4.1 Code Example: Counter with Redux Toolkit

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Counter with Redux Toolkit</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script
     src="https://cdn.jsdelivr.net/npm/@reduxjs/toolkit@1.9.5/dist/redux-too
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    const { configureStore, createSlice } = ReduxToolkit;

    const counterSlice = createSlice({
      name: 'counter',
```

```
      initialState: { value: 0 },
      reducers: {
        increment: (state) => {
          state.value += 1;
        },
        decrement: (state) => {
          state.value -= 1;
        }
      }
    });

    const { increment, decrement } = counterSlice.actions;
    const store = configureStore({ reducer: counterSlice.reducer });

    function Counter() {
      const [count, setCount] = React.useState(store.getState().value);

      React.useEffect(() => {
        const unsubscribe = store.subscribe(() => {
          setCount(store.getState().value);
        });
        return unsubscribe;
      }, []);

      const handleIncrement = () => store.dispatch(increment());
      const handleDecrement = () => store.dispatch(decrement());

      return (
        <div className="max-w-md mx-auto mt-10 p-6 bg-white rounded-lg shadow-md text-center">
          <h2 className="text-2xl font-bold text-blue-600 mb-4">Counter</h2>
          <p className="text-4xl mb-4">{count}</p>
          <button
            onClick={handleIncrement}
            className="bg-green-500 text-white p-2 rounded mr-2 hover:bg-green-600"
          >
            Increment
          </button>
          <button
            onClick={handleDecrement}
            className="bg-red-500 text-white p-2 rounded hover:bg-red-600"
          >
            Decrement
          </button>
        </div>
      );
```

```
    }

    const root =
       ReactDOM.createRoot(document.getElementById('root'));
    root.render(<Counter />);
  </script>
</body>
</html>
```

4.2 Output

- Centered box, "Counter" heading (blue).
- Large number (initially 0).
- "Increment" (green) aur "Decrement" (red) buttons.
- **Behavior**: Buttons pe click karne pe count badhta ya ghatta hai, Redux store se update hota hai.

5 15.4 Real-World Example: EduCampusHustle Cart

5.1 Code Example: Cart with Redux

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>EduCampusHustle Cart</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script
     src="https://cdn.jsdelivr.net/npm/@reduxjs/toolkit@1.9.5/dist/redux-too
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    const { configureStore, createSlice } = ReduxToolkit;

    // Cart Slice
    const cartSlice = createSlice({
      name: 'cart',
      initialState: { items: [], total: 0 },
      reducers: {
        addToCart: (state, action) => {
          const item = action.payload;
          state.items.push(item);
```

```
25            state.total += item.price;
26          },
27          removeFromCart: (state, action) => {
28            const itemId = action.payload;
29            const item = state.items.find(item => item.id ===
                 itemId);
30            if (item) {
31              state.items = state.items.filter(item => item.id !==
                   itemId);
32              state.total -= item.price;
33            }
34          }
35        }
36      });
37
38      const { addToCart, removeFromCart } = cartSlice.actions;
39      const store = configureStore({ reducer: cartSlice.reducer });
40
41      function CartItem({ item, onRemove }) {
42        return (
43          <div className="flex justify-between items-center p-2
                 border-b">
44            <span>{item.name} - Rs {item.price}</span>
45            <button
46              onClick={() => onRemove(item.id)}
47              className="bg-red-500 text-white p-1 rounded
                   hover:bg-red-600"
48            >
49              Remove
50            </button>
51          </div>
52        );
53      }
54
55      function Cart() {
56        const [cart, setCart] = React.useState(store.getState());
57
58        React.useEffect(() => {
59          const unsubscribe = store.subscribe(() => {
60            setCart(store.getState());
61          });
62          return unsubscribe;
63        }, []);
64
65        const handleAddToCart = () => {
66          const newItem = { id: Date.now(), name: "Web Dev 101",
                 price: 500 };
67          store.dispatch(addToCart(newItem));
68        };
69
70        const handleRemoveFromCart = (id) => {
```

```
71          store.dispatch(removeFromCart(id));
72        };
73
74        return (
75          <div className="max-w-md mx-auto mt-10 p-6
               bg-gradient-to-br from-blue-500 to-purple-600
               rounded-lg shadow-lg text-white">
76            <h2 className="text-2xl font-bold mb-4">Your Cart</h2>
77            {cart.items.length === 0 ? (
78              <p className="text-center">Your Cart is Empty! Start
                   Hustling!</p>
79            ) : (
80              <div>
81                {cart.items.map((item) => (
82                  <CartItem key={item.id} item={item}
                       onRemove={handleRemoveFromCart} />
83                ))}
84                <div className="mt-4 text-xl font-bold">Total: Rs
                     {cart.total}</div>
85              </div>
86            )}
87            <button
88              onClick={handleAddToCart}
89              className="mt-4 bg-green-500 text-white p-2 rounded
                   hover:bg-green-600"
90            >
91              Add to Cart
92            </button>
93          </div>
94        );
95      }
96
97      const root =
           ReactDOM.createRoot(document.getElementById('root'));
98      root.render(<Cart />);
99    </script>
100 </body>
101 </html>
```

5.2 Output

- Centered box with gradient (blue to purple).
- Heading: "Your Cart" (white, bold).
- Initially: "Your Cart is Empty! Start Hustling!" (centered).
- **Behavior**: - "Add to Cart" button pe click karne pe "Web Dev 101 - Rs 500" add hota hai. - "Remove" button pe click karne pe item delete hota hai. - Total price update hota hai (e.g., Rs 500, Rs 1000 agar multiple items).

6 15.5 Common Pitfalls

- Overusing Redux: Sirf complex state ke liye use karo, simple state ke liye useState enough hai.
- Improper Store Setup: Sare reducers ek saath combine karna bhoolna.
- Performance: Unnecessary re-renders avoid karne ke liye memoization.

7 15.6 Interview Tips

- **Redux kyun use karte ho?** - Global state manage karne ke liye, predictable updates ke liye.
- **Redux Toolkit kya hai?** - Redux ka simplified version, createSlice se asaan setup.
- **State optimization?** - useSelector aur memoization ke saath.

8 Assignment: Practice Time

8.1 Task 1: User Authentication State

- Ek Redux slice banao jisme user login state ho (e.g., isLoggedIn: true/false, user: name, email).
- Actions: `loginUser` (user data add kare) aur `logoutUser` (state reset kare).
- UI: Ek button "Login" (login pe "Logout" ho jaye) aur user name dikhaye.

8.2 Task 2: Course Filters

- Ek Redux slice banao jisme course filters ho (e.g., category: "Web Dev", priceRange: "0-500").
- Actions: `setCategory` aur `setPriceRange`.
- UI: Dropdowns se filters change ho, aur filtered courses (mock data) dikhaye.

8.3 Task 3: Enhance Cart

- Tumhare `CartWithRedux.html` ko enhance karo:
- Add quantity field har item ke liye (increment/decrement buttons).
- Update total price accordingly.
- Add "Checkout" button (alert with total).