Redux Toolkit Shopping Cart Code

### 1. package.json Dependencies

{

"name": "shopping-cart-app",

"version": "1.0.0",

"private": true,

"dependencies": {

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-scripts": "5.0.1",

"@reduxjs/toolkit": "^1.9.5",

"react-redux": "^8.1.1"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

}

}

### 2. src/index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import store from './app/store';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Provider store={store}>

<App />

</Provider>

);

### 3. src/app/store.js

import { configureStore } from '@reduxjs/toolkit';

import cartReducer from '../features/cart/cartSlice';

export const store = configureStore({

reducer: {

cart: cartReducer,

},

});

export default store;

### 4. src/features/cart/cartSlice.js

import { createSlice } from '@reduxjs/toolkit';

const initialState = {

items: [],

};

const cartSlice = createSlice({

name: 'cart',

initialState,

reducers: {

addItem: (state, action) => {

const existing = state.items.find(item => item.id === action.payload.id);

if (existing) {

existing.quantity += 1;

} else {

state.items.push({ ...action.payload, quantity: 1 });

}

},

removeItem: (state, action) => {

state.items = state.items.filter(item => item.id !== action.payload);

},

incrementQuantity: (state, action) => {

const item = state.items.find(item => item.id === action.payload);

if (item) item.quantity += 1;

},

decrementQuantity: (state, action) => {

const item = state.items.find(item => item.id === action.payload);

if (item && item.quantity > 1) item.quantity -= 1;

},

},

});

export const { addItem, removeItem, incrementQuantity, decrementQuantity } = cartSlice.actions;

export default cartSlice.reducer;

### 5. src/components/ProductList.js

import React from 'react';

import { useDispatch } from 'react-redux';

import { addItem } from '../features/cart/cartSlice';

const products = [

{ id: 1, name: 'Product A', price: 100 },

{ id: 2, name: 'Product B', price: 200 },

{ id: 3, name: 'Product C', price: 300 },

];

const ProductList = () => {

const dispatch = useDispatch();

return (

<div>

<h2>Products</h2>

{products.map(product => (

<div key={product.id} style={{ border: '1px solid black', margin: '10px', padding: '10px' }}>

<h3>{product.name}</h3>

<p>Price: ${product.price}</p>

<button onClick={() => dispatch(addItem(product))}>Add to Cart</button>

</div>

))}

</div>

);

};

export default ProductList;

### 6. src/components/Cart.js

import React from 'react';

import { useSelector, useDispatch } from 'react-redux';

import { removeItem, incrementQuantity, decrementQuantity } from '../features/cart/cartSlice';

const Cart = () => {

const items = useSelector(state => state.cart.items);

const dispatch = useDispatch();

const totalPrice = items.reduce((total, item) => total + item.price \* item.quantity, 0);

return (

<div>

<h2>Shopping Cart</h2>

{items.length === 0 && <p>Cart is empty</p>}

{items.map(item => (

<div key={item.id} style={{ border: '1px solid gray', margin: '5px', padding: '5px' }}>

<h4>{item.name}</h4>

<p>Price: ${item.price}</p>

<p>

Quantity:

<button onClick={() => dispatch(decrementQuantity(item.id))}>-</button>

{item.quantity}

<button onClick={() => dispatch(incrementQuantity(item.id))}>+</button>

</p>

<p>Total: ${item.price \* item.quantity}</p>

<button onClick={() => dispatch(removeItem(item.id))}>Remove</button>

</div>

))}

<h3>Total Price: ${totalPrice}</h3>

</div>

);

};

export default Cart;

### 7. src/App.js

import React from 'react';

import ProductList from './components/ProductList';

import Cart from './components/Cart';

function App() {

return (

<div style={{ display: 'flex', gap: '50px', padding: '20px' }}>

<ProductList />

<Cart />

</div>

);

}

export default App;

### 8. Basic Styling (Optional)

body { font-family: Arial, sans-serif; }

button { margin: 0 5px; padding: 5px 10px; }