Full Flow

```
////////////////////////// IMPORTS //////////////////////////
import React from 'react';
import { createRoot } from 'react-dom/client';
import { Provider, useDispatch, useSelector } from 'react-redux';
import { configureStore, createSlice } from '@reduxjs/toolkit';

////////////////////////////////////////////////////////////
// 🧠 REDUX SLICE — A Mini-Warehouse for One Feature (cart)
////////////////////////////////////////////////////////////
const cartSlice = createSlice({
  name: 'cart',
  initialState: {
    cartItems: [],
    totalQuantity: 0,
  },
  reducers: {
    // ➕ Adds item to cart or increments quantity if it already exists
    addItem: (state, action) => {
      const item = state.cartItems.find(i => i.id === action.payload.id);
      if (item) {
        item.quantity += 1;
      } else {
        state.cartItems.push({ ...action.payload, quantity: 1 });
      }
      state.totalQuantity += 1;
    },

    // ❌ Removes item from cart completely
    removeItem: (state, action) => {
      const item = state.cartItems.find(i => i.id === action.payload.id);
      if (item) {
        state.totalQuantity -= item.quantity;
        state.cartItems = state.cartItems.filter(i => i.id !== action.payload.id);
      }
    },

    // 🧹 Clears entire cart
    clearCart: (state) => {
      state.cartItems = [];
      state.totalQuantity = 0;
    },

    // ⬆️ Increases quantity of an item
    incrementQuantity: (state, action) => {
      const item = state.cartItems.find(i => i.id === action.payload.id);
      if (item) {
        item.quantity += 1;
        state.totalQuantity += 1;
```

```
      }
    },

    // ⬇️ Decreases quantity or removes if it's 1
    decrementQuantity: (state, action) => {
      const item = state.cartItems.find(i => i.id === action.payload.id);
      if (item && item.quantity > 1) {
        item.quantity -= 1;
        state.totalQuantity -= 1;
      } else if (item && item.quantity === 1) {
        state.cartItems = state.cartItems.filter(i => i.id !== action.payload.id);
        state.totalQuantity -= 1;
      }
    },
  },
});

////////////////////////// EXPORT ACTIONS & REDUCER //////////////////////////
export const {
  addItem,
  removeItem,
  clearCart,
  incrementQuantity,
  decrementQuantity,
} = cartSlice.actions;

export default cartSlice.reducer;

/////////////////////////////////////////////////////////////
// 🏢 APP STORE — The Central Warehouse
/////////////////////////////////////////////////////////////
import cartReducer from './cartSlice'; // 👈 This would be your slice file

const store = configureStore({
  reducer: {
    cart: cartReducer, // Registers the `cart` slice in the global store
  },
});

export default store;

/////////////////////////////////////////////////////////////
// 🛍️ PRODUCT COMPONENT — UI + Dispatching Actions
/////////////////////////////////////////////////////////////
function Product({ product }) {
  const dispatch = useDispatch(); // To trigger actions like add/remove

  const handleAdd = () => dispatch(addItem(product));
  const handleRemove = () => dispatch(removeItem(product));

  return (
    <div style={{ margin: '1rem' }}>
      <h4>{product.name}</h4>
      <button onClick={handleAdd}>➕ Add to Cart</button>
```

```
        <button onClick={handleRemove} style={{ marginLeft: '10px' }}>
          ❌ Remove
        </button>
      </div>
    );
}

////////////////////////////////////////////////////////////
// 🛒 CART SUMMARY – Reading & Dispatching Redux State
////////////////////////////////////////////////////////////
function CartSummary() {
  const dispatch = useDispatch();

  // ☑️ Scoped selectors used only inside this component
  const selectCartItems = state => state.cart.cartItems;
  const selectTotalQuantity = state => state.cart.totalQuantity;

  // 👇 Reading Redux state using useSelector + selectors
  const cartItems = useSelector(selectCartItems);
  const totalQuantity = useSelector(selectTotalQuantity);

  // 🧠 Dispatching actions to manipulate cart
  const handleClearCart = () => dispatch(clearCart());
  const handleIncrement = item => dispatch(incrementQuantity(item));
  const handleDecrement = item => dispatch(decrementQuantity(item));
  const handleRemove = item => dispatch(removeItem(item));

  return (
    <div style={{ padding: '1rem', border: '1px solid gray' }}>
      <h3>🛒 Cart Summary</h3>
      <p>Total Items: {totalQuantity}</p>
      <ul>
        {cartItems.map(item => (
          <li key={item.id}>
            {item.name} – Qty: {item.quantity}
            <button onClick={() => handleIncrement(item)}>➕</button>
            <button onClick={() => handleDecrement(item)}>➖</button>
            <button onClick={() => handleRemove(item)}>❌</button>
          </li>
        ))}
      </ul>
      {cartItems.length > 0 && (
        <button onClick={handleClearCart}>🧹 Clear Cart</button>
      )}
    </div>
  );
}

////////////////////////////////////////////////////////////
// 🌐 MAIN APP COMPONENT – Renders UI
////////////////////////////////////////////////////////////
function App() {
  const products = [
    { id: 1, name: '🍏 Apple' },
```

```
    { id: 2, name: '🥕 Carrot' },
    { id: 3, name: '🥛 Milk' },
  ];

  return (
    <div style={{ padding: '2rem' }}>
      <h1>🛍️ Redux Cart (Scoped Selectors)</h1>
      <CartSummary />
      {products.map(product => (
        <Product key={product.id} product={product} />
      ))}
    </div>
  );
}

////////////////////////////////////////////////////////////
// 🧩 ENTRY POINT — React DOM + Redux Store Binding
////////////////////////////////////////////////////////////
const container = document.getElementById('root');
const root = createRoot(container);
root.render(
  // Wrap App with Provider to give Redux store to all components
  <Provider store={store}>
    <App />
  </Provider>
);
```
