EXPERIMENT No 3

Aim : Manage complex state with **Redux or Context API**

Theory:

1. Introduction to State Management in React

In React, **state** refers to the dynamic data that determines how a component behaves and what it renders. For small applications, state can be managed within individual components using the useState or useReducer hooks.However, as the application grows and multiple components need to access or modify the same data, **prop drilling** (passing props through multiple levels) becomes inefficient and difficult to maintain. This is where **global state management** becomes essential.

2. What is Redux?

Redux is a predictable state container for JavaScript applications. It allows developers to store the entire application's state in a single object called the **store**.The store can be accessed from any component without the need for deep prop passing, ensuring a **single source of truth**.

Core Concepts of Redux:

1. **Store** – Holds the application's state.
2. **Actions** – Plain JavaScript objects describing what happened.
3. **Reducers** – Functions that take the current state and an action, and return a new state.
4. **Dispatch** – Method used to send actions to the store.
5. **Selectors** – Functions to retrieve specific pieces of state.

3. Redux Toolkit

Redux Toolkit (RTK) is the official, recommended way to write Redux logic. It simplifies the process of creating reducers, actions, and configuring the store.

Advantages of Redux Toolkit:

- Less boilerplate code.
- Built-in createSlice for reducers + actions in one place.
- Built-in configureStore with sensible defaults.
- Immutable state handling with **Immer** library.
- Developer-friendly features like Redux DevTools integration.

4. React Redux

React Redux is the official binding library for using Redux in React applications. It provides:

- **Provider**: Makes the Redux store available to the whole application.
- **useSelector**: Allows components to read data from the store.
- **useDispatch**: Allows components to dispatch actions to the store.

5. Application Flow with Redux Toolkit

1. Create a **slice** using createSlice which contains:
    - The initial state.
    - Reducer functions.
    - Automatically generated action creators.
2. Configure the **store** with configureStore.
3. Wrap the application in a Provider component and pass the store.
4. Use useDispatch to modify state and useSelector to read state.

6. Project Implementation (Base Redux Part)

The implemented application is a **Cart Management App** with the following features:

- **Add item by name**.
- **Remove individual items**.
- **Clear all items**.
- **Display item count**.

This covers the **core requirement of the lab**: managing a complex state using Redux.

7. 30% Extra Functionality Beyond Redux

To go beyond the lab requirements and add an **extra 30% concept**, the following features were implemented:

a) Local Storage Persistence

- The cart data is automatically saved to the browser's local storage.
- On page reload, data is restored, ensuring persistence.
- This feature is implemented using useEffect to watch for changes in the Redux store and sync them with local storage.

b) Price Assignment & Total Price Calculation

- Each item added has a randomly generated price (for demo purposes).
- The application calculates the total price of all items in the cart.
- This introduces an additional derived state concept, where the displayed value is calculated based on existing Redux state.

These additional functionalities go **beyond the basics of Redux** and demonstrate **real-world applicability** by combining persistent storage and computed state.

Code:

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSli

redux-demo > index.html > ...

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>

```

index.html | index.css | app.module.css | App.jsx | mair

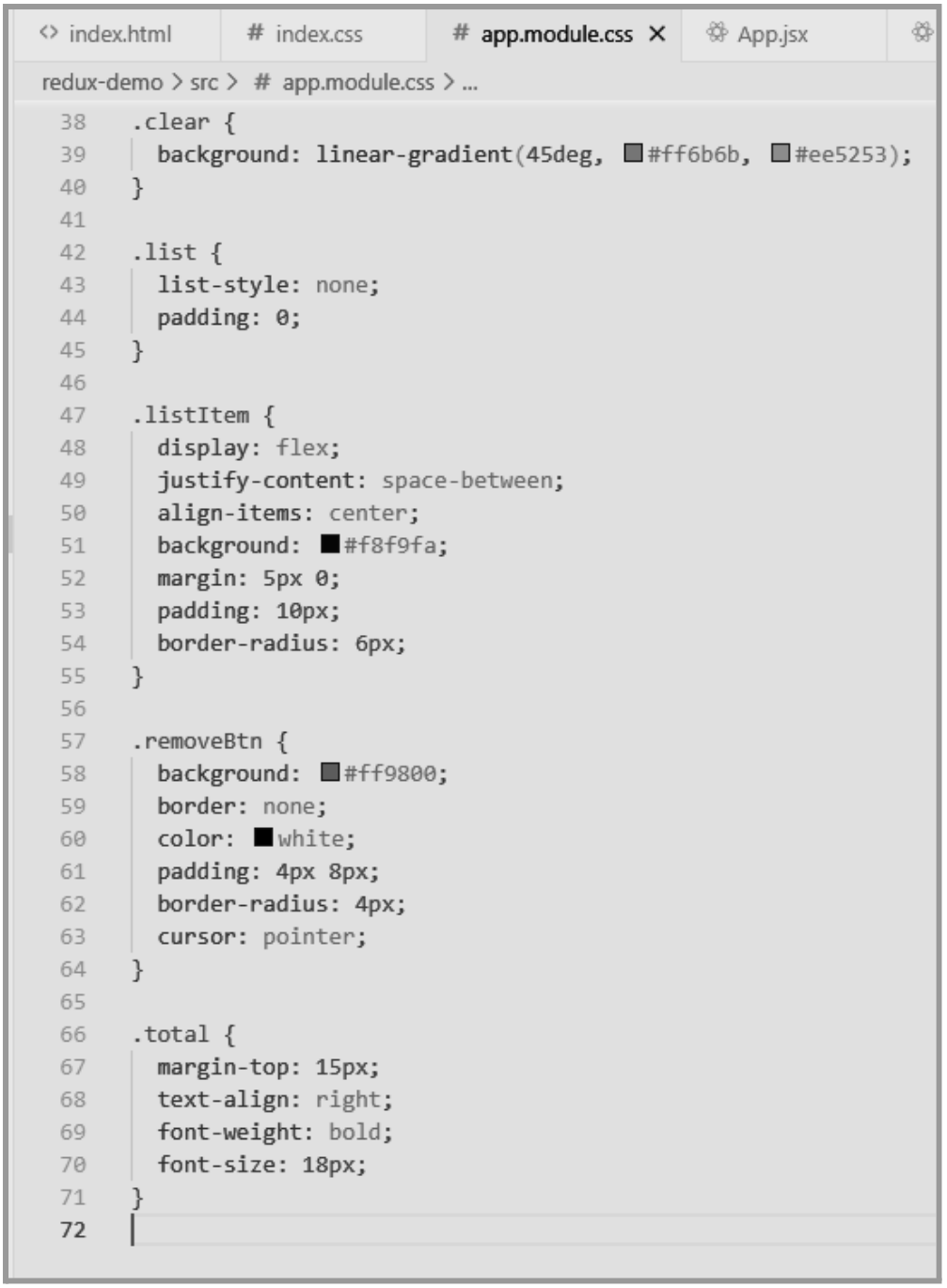
redux-demo > src > # index.css > ...

```
body {
  margin: 0;
  padding: 0;
  min-height: 100vh;
  background: linear-gradient(135deg, #f8f9fa, #e9ecef);
  display: flex;
  justify-content: center;
  align-items: flex-start;
  font-family: 'Segoe UI', sans-serif;
}

```

index.html | # index.css | # app.module.css | App.jsx

redux-demo > src > # app.module.css > ...

```css
.container {
  width: 100%;
  max-width: 600px;
  margin-top: 50px;
  background: #ffffff;
  padding: 30px;
  border-radius: 16px;
  box-shadow: 0 6px 30px rgba(0, 0, 0, 0.05);
}

.title {
  text-align: center;
  color: #2d3436;
  margin-bottom: 20px;
}

.buttons {
  display: flex;
  justify-content: center;
  gap: 10px;
  margin-bottom: 20px;
}

.button {
  background: linear-gradient(45deg, #6c63ff, #5f57d3);
  color: white;
  border: none;
  padding: 10px 18px;
  border-radius: 8px;
  cursor: pointer;
  transition: 0.3s;
}

.button:hover {
  transform: scale(1.05);
}

.clear {
  background: linear-gradient(45deg, #ff6b6b, #ee5253);
```


index.html
index.css
app.module.css
App.jsx
redux-demo > src > # app.module.css > ...
38 .clear {
39 background: linear-gradient(45deg, #ff6b6b, #ee5253);
40 }
41
42 .list {
43 list-style: none;
44 padding: 0;
45 }
46
47 .listItem {
48 display: flex;
49 justify-content: space-between;
50 align-items: center;
51 background: #f8f9fa;
52 margin: 5px 0;
53 padding: 10px;
54 border-radius: 6px;
55 }
56
57 .removeBtn {
58 background: #ff9800;
59 border: none;
60 color: white;
61 padding: 4px 8px;
62 border-radius: 4px;
63 cursor: pointer;
64 }
65
66 .total {
67 margin-top: 15px;
68 text-align: right;
69 font-weight: bold;
70 font-size: 18px;
71 }
72

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSlic

redux-demo > src > App.jsx > ...

```jsx
import { useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { addItem, removeItem, clearCart, applyDiscount } from "./store/cartSlice";
import styles from "./App.module.css";

export default function App() {
  const { items, discount } = useSelector(state => state.cart);
  const dispatch = useDispatch();
  const [itemName, setItemName] = useState("");
  const [itemPrice, setItemPrice] = useState("");
  const [code, setCode] = useState("");

  const handleAdd = () => {
    if (itemName.trim() === "" || !itemPrice) return;
    dispatch(addItem({
      id: Date.now(),
      name: itemName.trim(),
      price: parseFloat(itemPrice)
    }));
    setItemName("");
    setItemPrice("");
  };

  const handleDiscount = () => {
    if (code === "SAVE10") {
      dispatch(applyDiscount(10));
    } else if (code === "BIGSALE") {
      dispatch(applyDiscount(20));
    } else {
      dispatch(applyDiscount(0));
    }
  };

  const totalPrice = items.reduce((sum, i) => sum + i.price, 0);
  const discountedPrice = totalPrice - (totalPrice * discount / 100);

  return (
    <div className={styles.container}>
      <h1 className={styles.title}>🛒 Redux Cart with Extras</h1>
```

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSlice.js

redux-demo > src > App.jsx > ...

```
6   export default function App() {
39        <h1 className={styles.title}>🛒 Redux Cart with Extras</h1>
40
41        {/* Add Item Inputs */}
42        <div style={{ display: "flex", gap: "10px", marginBottom: "15px" }}>
43          <input
44            type="text"
45            value={itemName}
46            placeholder="Item name..."
47            onChange={(e) => setItemName(e.target.value)}
48            style={{
49              flex: 2,
50              padding: "10px",
51              borderRadius: "6px",
52              border: "1px solid #ccc",
53              fontSize: "16px"
54            }}
55          />
56          <input
57            type="number"
58            value={itemPrice}
59            placeholder="Price"
60            onChange={(e) => setItemPrice(e.target.value)}
61            style={{
62              flex: 1,
63              padding: "10px",
64              borderRadius: "6px",
65              border: "1px solid #ccc",
66              fontSize: "16px"
67            }}
68          />
69          <button className={styles.button} onClick={handleAdd}>➕ Add</button>
70        </div>
71
72        {/* Discount Code */}
73        <div style={{ display: "flex", gap: "10px", marginBottom: "20px" }}>
74          <input
75            type="text"
76            value={code}
```

index.html | index.css | app.module.css | App.jsx | main.jsx

redux-demo > src > App.jsx > ...

```
6   export default function App() {
71
72        {/* Discount Code */}
73        <div style={{ display: "flex", gap: "10px", marginBottom: "20px" }}>
74          <input
75            type="text"
76            value={code}
77            placeholder="Discount code..."
78            onChange={(e) => setCode(e.target.value)}
79            style={{
80              flex: 2,
81              padding: "10px",
82              borderRadius: "6px",
83              border: "1px solid #ccc",
84              fontSize: "16px"
85            }}
86          />
87          <button className={styles.button} onClick={handleDiscount}>
88            Apply
89          </button>
90        </div>
91
92        {/* Clear Cart Button */}
93        <div className={styles.buttons}>
94          <button
95            className={`${styles.button} ${styles.clear}`}
96            onClick={() => dispatch(clearCart())}
97          >
98            Clear Cart
99          </button>
100       </div>
101
102       {/* Cart List */}
103       <ul className={styles.list}>
104         {items.map(i => (
105           <li key={i.id} className={styles.listItem}>
106             {i.name} - ₹{i.price.toFixed(2)}
107             <button
```

index.html | index.css | app.module.css | App.jsx

redux-demo > src > App.jsx > ...

```jsx
export default function App() {
      </button>
    </div>

    {/* Cart List */}
    <ul className={styles.list}>
      {items.map(i => (
        <li key={i.id} className={styles.listItem}>
          {i.name} - ₹{i.price.toFixed(2)}
          <button
            className={styles.removeBtn}
            onClick={() => dispatch(removeItem(i.id))}
          >
            ✕
          </button>
        </li>
      ))}
    </ul>

    {/* Totals */}
    <p className={styles.total}>
      Subtotal: ₹{totalPrice.toFixed(2)}
    </p>
    {discount > 0 && (
      <p className={styles.total}>
        Discount: {discount}%
      </p>
    )}
    <p className={styles.total}>
      Final Total: ₹{discountedPrice.toFixed(2)}
    </p>
  </div>
  );
}
```

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSlice.js

redux-demo > src > main.jsx

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import { Provider } from "react-redux";
import { store } from "./store";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSlice.js

redux-demo > src > store > cartSlice.js > ...

```
import { createSlice } from "@reduxjs/toolkit";

const initialState = {
  items: JSON.parse(localStorage.getItem("cartItems")) || [],
  discount: 0
};

const cartSlice = createSlice({
  name: "cart",
  initialState,
  reducers: {
    addItem(state, action) {
      state.items.push(action.payload);
      localStorage.setItem("cartItems", JSON.stringify(state.items));
    },
    removeItem(state, action) {
      state.items = state.items.filter(i => i.id !== action.payload);
      localStorage.setItem("cartItems", JSON.stringify(state.items));
    },
    clearCart(state) {
      state.items = [];
      localStorage.removeItem("cartItems");
    },
    applyDiscount(state, action) {
      state.discount = action.payload; // percentage
    }
  },
});

export const { addItem, removeItem, clearCart, applyDiscount } = cartSlice.actions;
export default cartSlice.reducer;
```

index.html | index.css | app.module.css | App.jsx | main.jsx | cartSlice.js | index.js

redux-demo > src > store > index.js > ...

```
import { configureStore } from "@reduxjs/toolkit";
import cart from "./cartSlice";

export const store = configureStore({
  reducer: { cart },
});
```

Output:

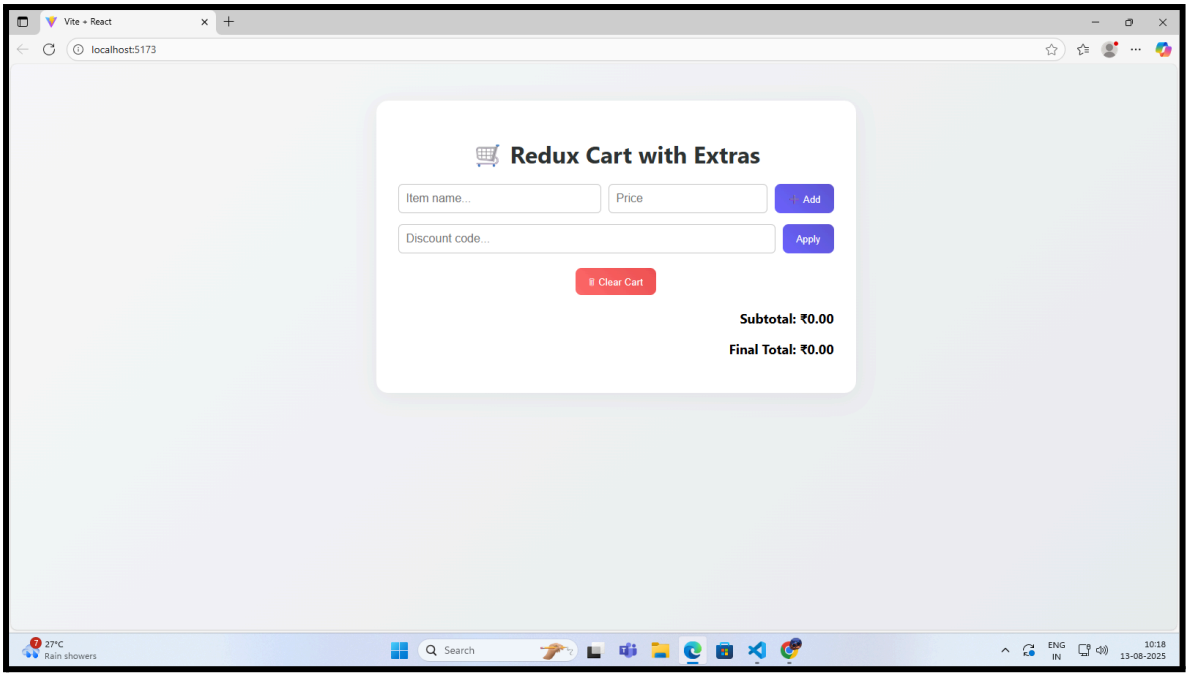


Figure 1.1 UI

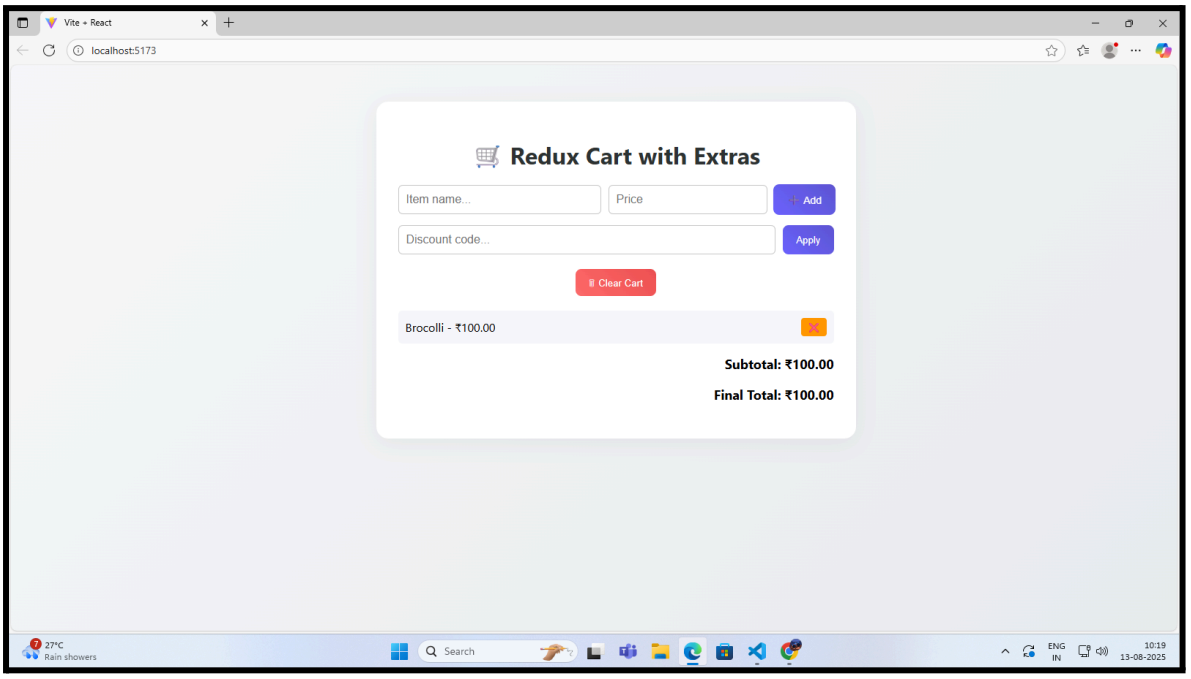


Figure 1.2 : Added Brocolli

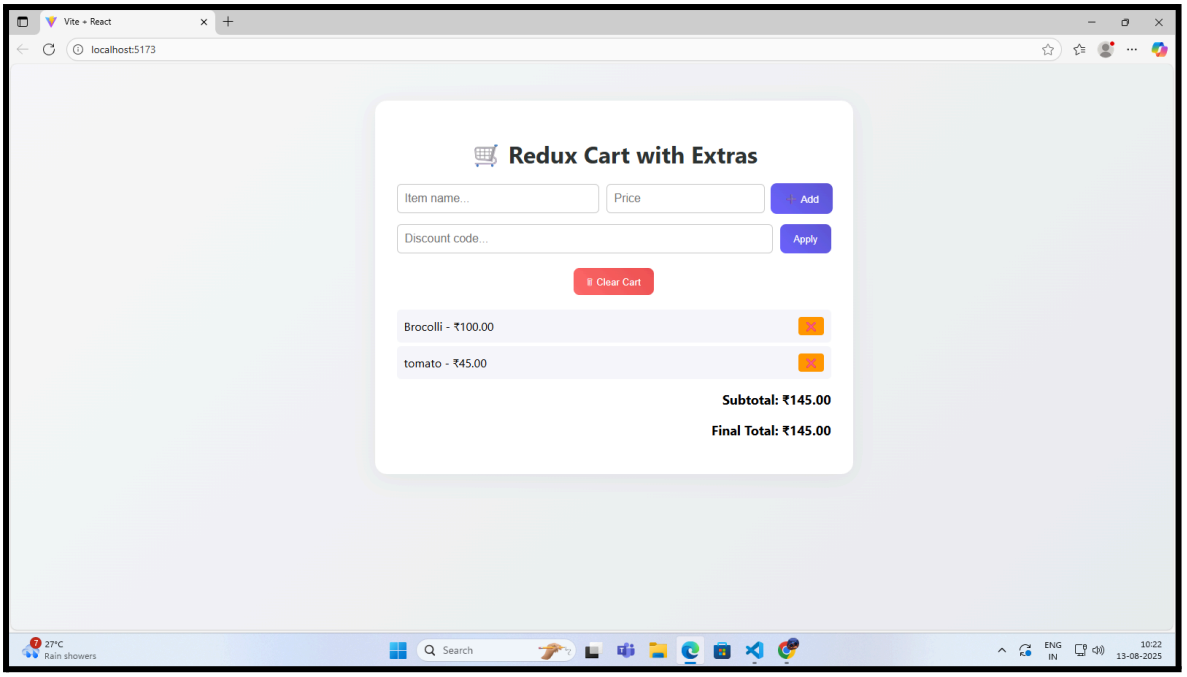


Figure 1.3 : Added Tomato

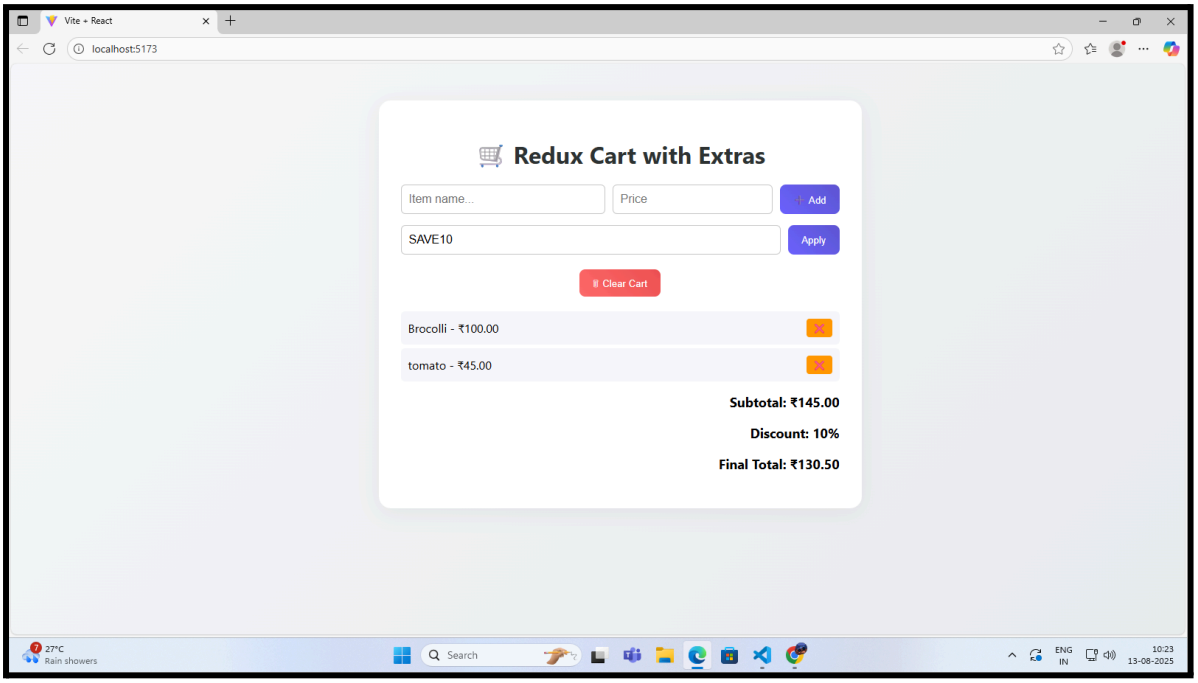


Figure 1.4 : Apply The Discount Code and Get the Output

Conclusion :

In this lab, we implemented **complex state management** using Redux Toolkit and React Redux, fulfilling the requirement to handle state centrally and make it accessible across multiple components without prop drilling.

We also added an **extra 30% functionality** that goes beyond Redux/Context API, namely:

- **Local storage persistence** to retain state after page reload.
- **Dynamic price generation and total price calculation** for cart items.

This enhancement showcases a real-world application scenario, improving user experience and demonstrating how Redux can be combined with browser APIs and derived state for advanced features.