🛒 **Shopping Cart in React – Implementation Report**

👨‍💻 **Developer: Dhanish Kumar**

🎓 **B.Tech Computer Science, Galgotias University**

🧰 **Tech Stack: React, Vite, Tailwind CSS, React Context, Reducers, Portals, Refs**

✅ Objective

The objective of this project was to implement a fully functional **shopping cart** using **React**. The cart supports adding, removing, and updating food items dynamically, and includes reusable components, context-based global state, reducers for cart operations, and modals using portals. The app is responsive and styled using Tailwind CSS.

🛠️ Implementation Process

1. Project Setup

- Used **Vite** for fast React project initialization.
- Installed Tailwind CSS for utility-first styling.
- Folder structure was organized into /components, /components/UI, /store, and root files.

2. Component Creation

- **Header.jsx**: Displays the app title and a Cart button with an item count badge.
- **Meals.jsx**: Renders a list of available meals using hardcoded DUMMY_MEALS.
- **MealItem.jsx**: Each meal item includes name, price, description, image, and an "Add to Cart" form.
- **Cart.jsx**: Modal-based component showing added items with dynamic updates.
- **CartItem.jsx**: Displays individual cart items with an image, details, quantity, and remove button.

3. State Management

- Used **React Context API** in CartContext.jsx to manage global cart state.
- Implemented a **reducer** in cart-reducer.js to handle cart operations like adding/removing items.

- Used useReducer for scalable and clean state updates.

4. React Portal & Modal

- Modal.jsx uses **React Portals** to render the cart over existing content for better UX.
- Backdrop and ModalOverlay were implemented for modal behavior.

5. Form Handling with Refs

- Input.jsx is a reusable component using forwardRef for input access in MealItemForm.
- Enabled item quantity selection with min/max validations.

🎨 UI Styling and Responsiveness

- Used **Tailwind CSS** for consistent, responsive design.
- Ensured responsiveness with flex/grid layout, proper spacing, and scrollable modal content on smaller screens.
- Images were added to each meal and cart item for a polished UI.
- Buttons, hover effects, and accessibility (alt tags, button titles) were considered.

Sample Styling Highlights:

- Cart modal appears centered on all screen sizes.
- Responsive meal grid.
- Clear typography and spacing using Tailwind utility classes.

⚠️ Challenges Faced

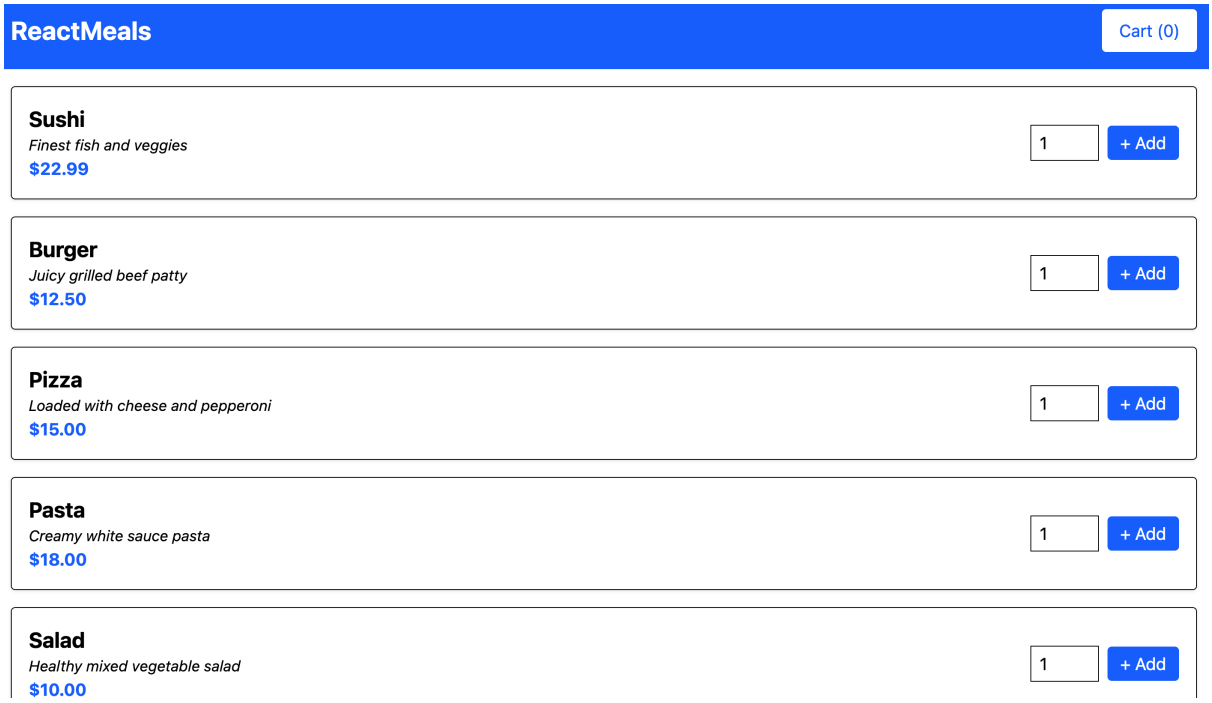
Challenge	Solution
Modal appearing behind content	Used z-index and ReactDOM.createPortal correctly.
Syncing cart item quantities	Managed with reducer logic and consistent id usage.
Form validation	Added input limits and default values using refs.

Challenge	Solution
Tailwind styles not applying initially	Fixed by checking PostCSS setup and ensuring Tailwind CSS was correctly configured in vite.config.js.

📦 Future Improvements

- Add **localStorage** to persist cart across reloads.
- Integrate with **Firebase or a backend** for real data.
- Add **checkout form** and payment simulation.
- Add **+ / – quantity** buttons inside the cart.
- Dark mode support using Tailwind's theming.

📸 Screenshots



Your Cart

Sushi

Finest fish and veggies

$22.99 × 1

Burger

Juicy grilled beef patty

$12.50 × 1

Pizza

Loaded with cheese and pepperoni

$15.00 × 1

Pasta

Creamy white sauce pasta

$18.00 × 1

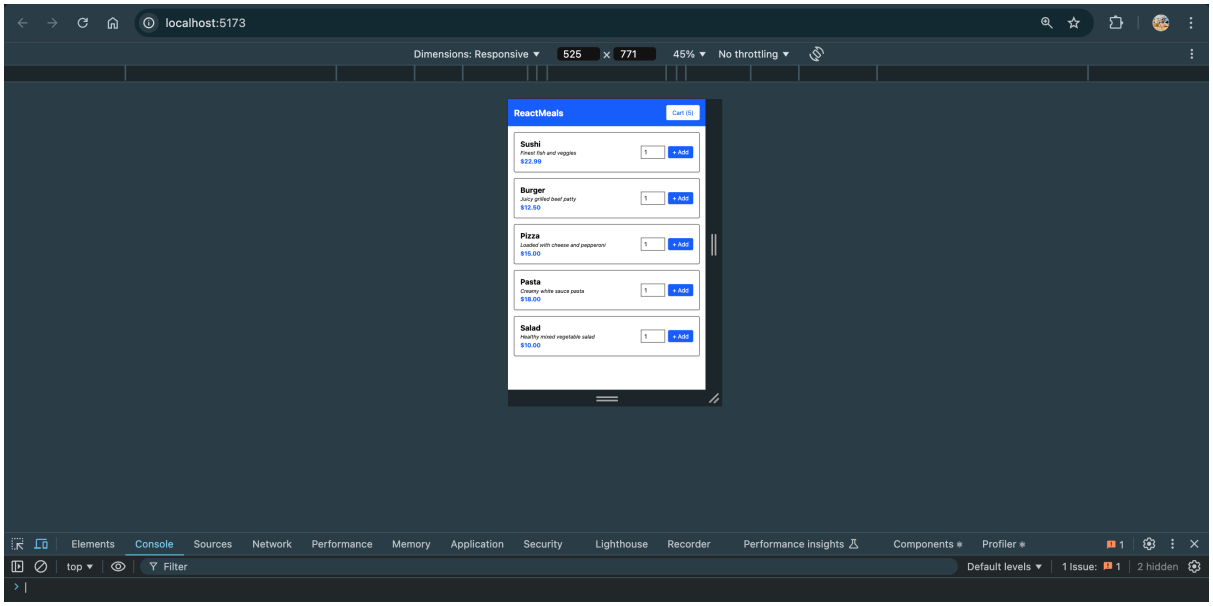
Salad

Healthy mixed vegetable salad

$10.00 × 1

Total: $78.49

Close


localhost:5173
Dimensions: Responsive
525
771
45%
No throttling
ReactMeals
Cart (5)
Sushi
Finest fish and veggies
$22.99
Burger
Juicy grilled beef patty
$12.50
Pizza
Loaded with cheese and pepperoni
$15.00
Pasta
Creamy white sauce pasta
$18.00
Salad
Healthy mixed vegetable salad
$10.00
+ Add
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Components
Profiler
top
Filter
Default levels
1 Issue:
2 hidden