**📦 STORE MANAGER: KEEP TRACK OF INVENTORY**

**INTRODUCTION**

**🔹 Project Title:**

**Store Manager: Keep Track of Inventory**



**🔹 TEAM LEADER:**

* HARIPRIYA S – Frontend Development

Mail ID – 202400710@sigc.edu

**🔹 TEAM MEMBERS:**

* HEMALATHA P  –  UI/UX Designer

Mail ID – 202400849@sigc.edu

* ISHWARYA S – Backend Developer

Mail ID – 202400033@sigc.edu

* JAYASREE R  –  Tester

Mail ID – 202400725@sigc.edu

**PROJECT OVERVIEW**

**🔹 Purpose:**

The **Store Manager** application is a web-based inventory tracking system designed to help small to mid-sized businesses manage their stock, monitor inventory levels, and streamline their store operations.

**🔹 Goals:**

* Simplify inventory tracking
* Reduce human error in stock management
* Provide real-time updates and visualization of store items

**🔹 Key Features:**

* Add, update, and delete inventory items
* Real-time item listing
* User-friendly interface with responsive design
* Visual indicators for stock availability

**ARCHITECTURE**

**🔹 Component Structure:**

* App.js – Root component managing routing
* Navbar.js – Navigation bar
* InventoryList.js – Displays list of items
* InventoryItem.js – Single item card/component
* AddItemForm.js – Form to add or edit items

**🔹 Component Interaction:**

* App.js loads routes and passes props to pages
* InventoryList.js gets item data from state and maps it to InventoryItem components

**🔹 State Management:**

* **React Context API** is used for global state management
* Local state is used for form inputs

**ROUTING**

**🔹 React Router:**

* We used react-router-dom for routing.
* Defined Routes:
* / – Home/Inventory List
* /add-item – Add Item Page
* /edit/:id – Edit existing item

**🔹 Navigation:**

* Navbar contains links to all key pages
* Routes are wrapped in <BrowserRouter> for SPA behavior

**SETUP INSTRUCTIONS**

**🔹 Prerequisites:**

* Node.js (v14 or higher)
* npm (v6 or higher)
* Code Editor (e.g., VS Code)

**🔹 Installation Steps:**

# Clone the repo

git clone https://github.com/your-username/store-manager.git

# Navigate to the project

cd store-manager

# Install dependencies

npm install

# Start the development server

npm start

**FOLDER STRUCTURE**

store-manager/

├── public/

├── src/

│   ├── assets/

│   ├── components/

│   │   ├── Navbar.js

│   │   ├── InventoryItem.js

│   │   ├── InventoryList.js

│   │   └── AddItemForm.js

│   ├── context/

│   ├── pages/

│   ├── App.js

│   └── index.js

├── tailwind.config.js

├── package.json

**UTILITIES**

**🔹 Context:**

* InventoryContext.js provides a global state for inventory data
* Methods: addItem, deleteItem, updateItem, getItemById

**🔹 Custom Hooks:**

* useInventory.js – Access context functions and state

**RUNNING THE APPLICATION**

To run the application locally:

npm install

npm start 

**KEY COMPONENTS**

**✅ InventoryList**

* Maps through inventory and renders InventoryItem components.

**✅ AddItemForm**

* Takes user input to add or update items.
* Props: onSubmit, initialData

**✅ InventoryItem**

* Shows item details with buttons for edit and delete.
* Props: item, onEdit, onDelete

**REUSABLE COMPONENTS**

**🔁 Button.js**

* Configurable buttons used across the app.
* Props: label, onClick, type

**🔁 Modal.js *(Optional for future use)***

* Can be used for confirmation dialogs.

All reusable components are kept under /components/common/.

**STATE MANAGEMENT**

**🔹 Global State:**

* Managed using React Context API.
* Inventory list is available globally.
* Methods: Add, Edit, Delete handled in context.

**🔹 Local State:**

* Used within forms (useState) for inputs and error handling.
* Reset after form submission or cancel.

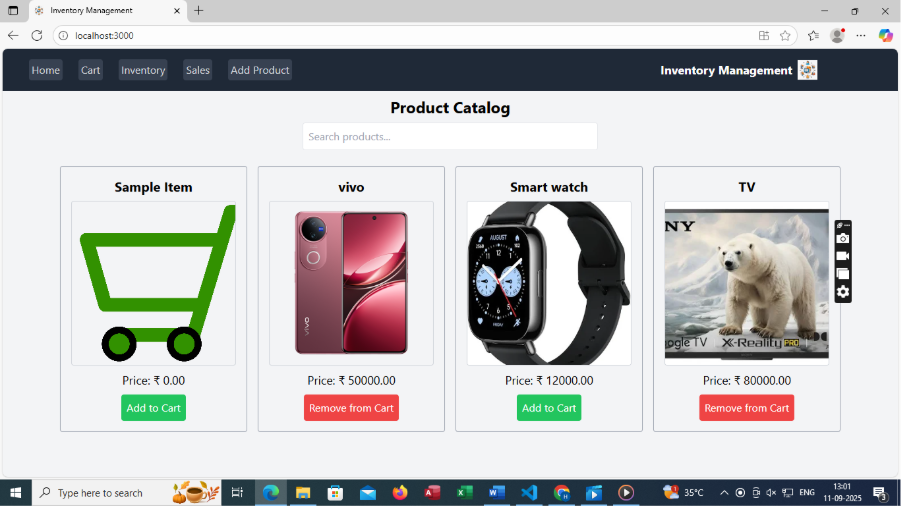
**USER INTERFACE**

**🔹 UI Overview:**

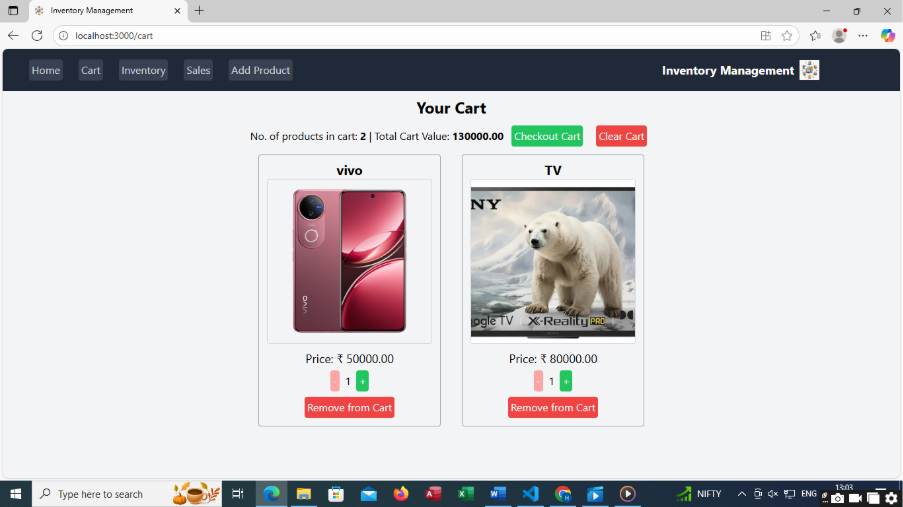
* Clean and modern interface using Tailwind CSS
* Responsive across devices (tested on mobile, tablet, desktop)

**🔹 Screenshots:**

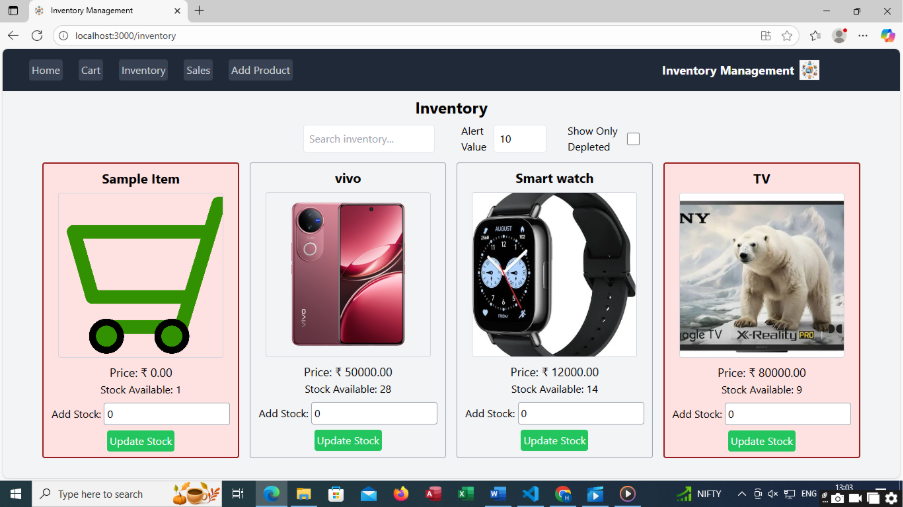
* **Home**



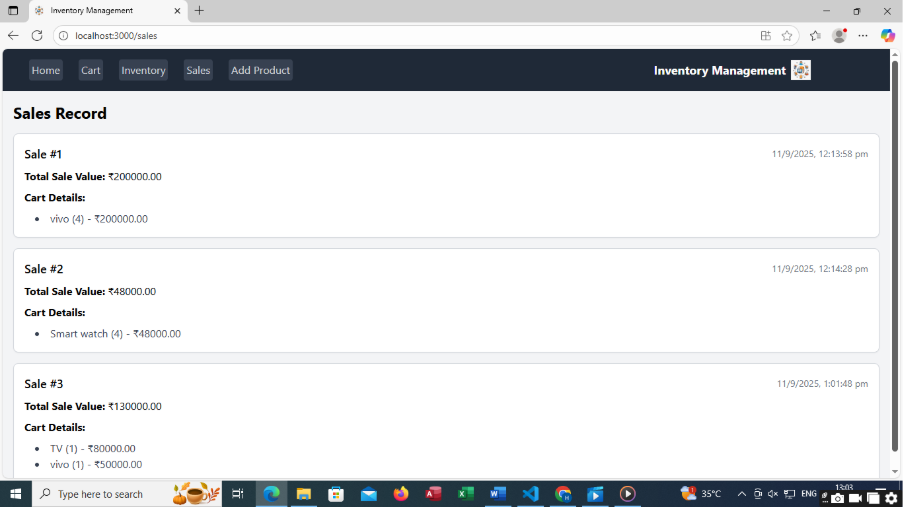
* **Cart**



* **Inventory**

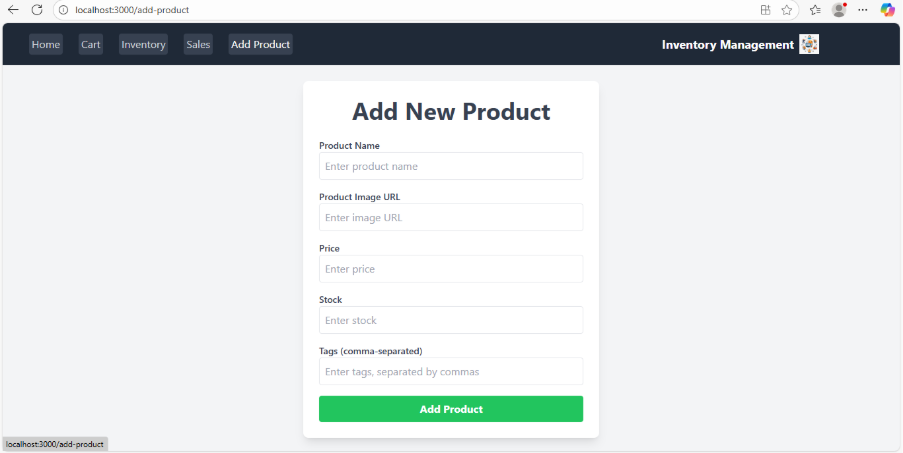


* **Sales**



* **Add Product**



**STYLING**

**🔹 Tailwind CSS:**

* Utility-first CSS framework used for layout and responsiveness.
* Custom colors defined in tailwind.config.js.

**🔹 Theming:**

* Primary and secondary color palette used for branding.
* Consistent spacing and typography via Tailwind classes.

**TESTING**

**🔹 Testing Strategy:**

* Unit Testing using **Jest**
* Component Testing using **React Testing Library**

**🔹 What We Test:**

* Rendering of components
* Add/Delete functions
* Route navigation

**🔹 Code Coverage:**

* npm test -- --coverage used to generate test coverage reports.

**KNOWN ISSUES & FUTURE ENHANCEMENTS**

**⚠️ Known Issues:**

* Deleting an item occasionally throws a console warning.
* Page refresh removes in-memory state (no backend used yet).

**🚀 Future Enhancements:**

* Integrate with a backend API (Node.js/Express + MongoDB)
* Add authentication (login/signup)
* Export inventory data to Excel or PDF
* Add pagination for large item lists
* Role-based access (Admin, Staff)