Project Documentation

1. Introduction

Project Title: Store Manager – Inventory Management System

Team ID: NM2025TMID47829

Team Leader: Balaji M - balajimani264@gmail.com

Team Members:

- Sakthi P - palanisamyp55895@gmail.com
- Sakthivel G - sakthivelmegala820@gmail.com
- Sanjay K - sanjaykumat7427@gmail.com
- Santhosh Kumar S - sanjaykumat7427@gmail.com

---

2. Project Overview

Purpose:

The Store Manager is a React-based frontend application that helps a manager keep track of inventory, manage products, handle customer carts, and record sales. It simulates a small shop's inventory management system where the Store Manager is the main user.

Features:

Manager Functionalities

- **Add New Products** – name, image URL, price, stock, tags.
- **Update Stock on Arrival** – includes negative stock prevention.
- **View Inventory & Stock** – searchable inventory list with stock alerts.
- **Check Depleting Stock** – monitor products reaching custom stock thresholds.

- **Cart Management** – add, update, remove items, and calculate total.
- **Checkout Process** – clears cart, reduces stock, logs sale.
- **Sales Records** – logs date/time, items sold, and total value (latest first).

---

3. Architecture

- **Frontend:** React.js (via Create React App), TailwindCSS
- **Backend:** Not applicable – frontend-only simulation
- **Database/Storage:** LocalStorage for persistence

---

4. Setup Instructions

Prerequisites:

- **Node.js + npm** – https://nodejs.org/
- **Git** – Install Git for version control: https://git-scm.com/downloads
- **Code Editor** – Use a code editor like:
    - VS Code: https://code.visualstudio.com/
- **Basic React Knowledge**

Installation Steps:

- Clone the repository
    - git clone https://github.com/balafromtn/NM-Project-Store-Management
- Move into project directory: cd store-manager
- Install dependencies: npm install
- Run the app: npm start
- Access App: http://localhost:3000

5. Folder Structure

```
store-manager/
├─ public/
│   └─ index.html
├─ src/
│   ├─ components/
│   │   ├─ CartDrawer.js
│   │   ├─ Filters.js
│   │   ├─ Header.js
│   │   ├─ ProductCart.js
│   │   └─ ProductForm.js
│   ├─ context/
│   │   ├─ CartContext.js
│   │   ├─ InventoryContext.js
│   │   ├─ SalesContext.js
│   │   └─ reducers.js
│   ├─ hooks/
│   │   └─ useLocalStorage.js
│   ├─ pages/
│   │   ├─ CartPage.js
│   │   ├─ InventoryPage.js
│   │   └─ SalesPage.js
│   ├─ utils.js
│   ├─ App.js
│   ├─ index.js
│   └─ index.css
├─ package.json
├─ package-lock.json
├─ tailwind.config.js
└─ postcss.config.js
```

6. State & Technical Flow

State Management: React Context API + Reducers

Reducers:

- **Inventory Reducer** → add/update stock, handle sales.
- **Cart Reducer** → manage cart operations.
- **Sales Reducer** → log completed sales.

Persistence: State is stored in localStorage to maintain cart and inventory across sessions.

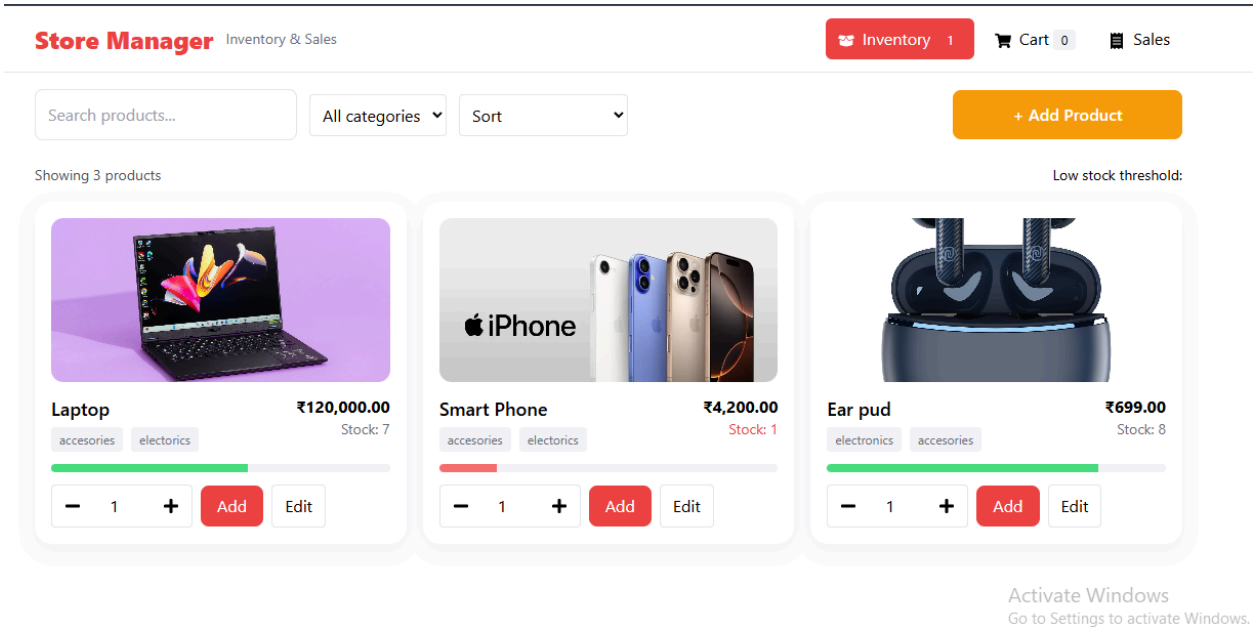
Event Flow:

1. User adds products to inventory.
2. Products are added to the cart via ProductCard.
3. Checkout reduces inventory and logs the sale.
4. CartDrawer is controlled via a button in the Header component.

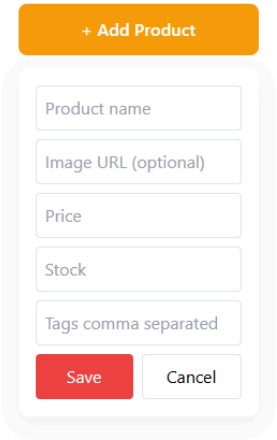
---

7. User Interface

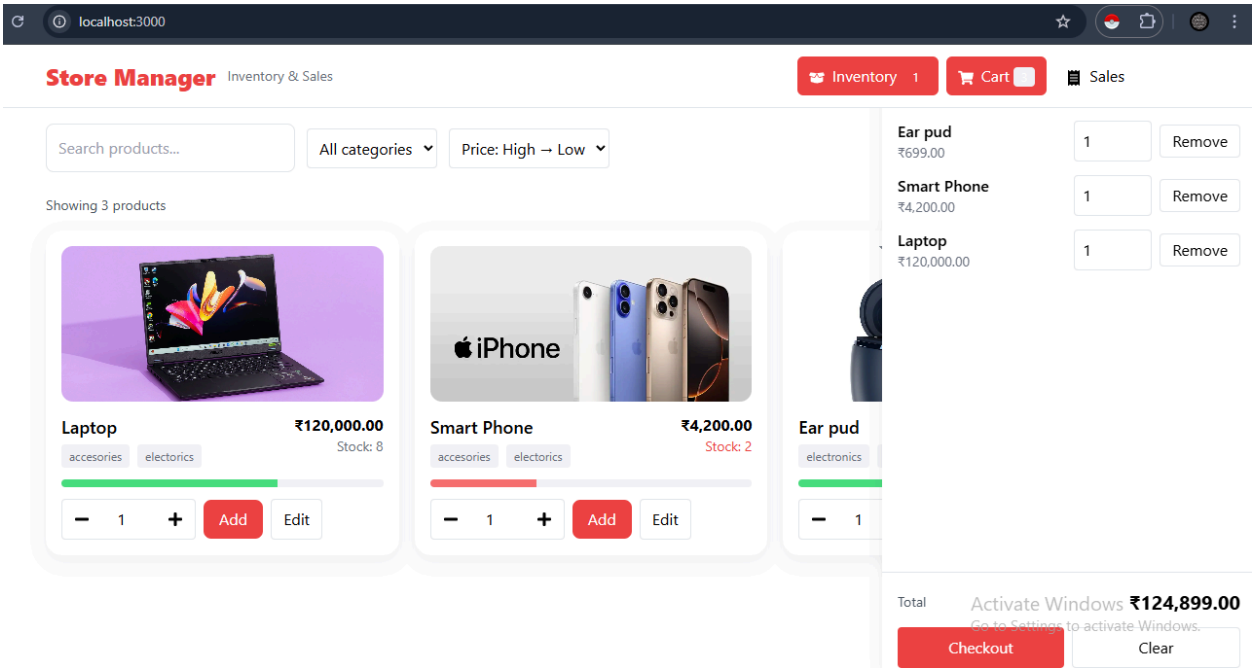
- **Inventory Page:** Displays product list, search bar, stock alerts, low-stock highlights.



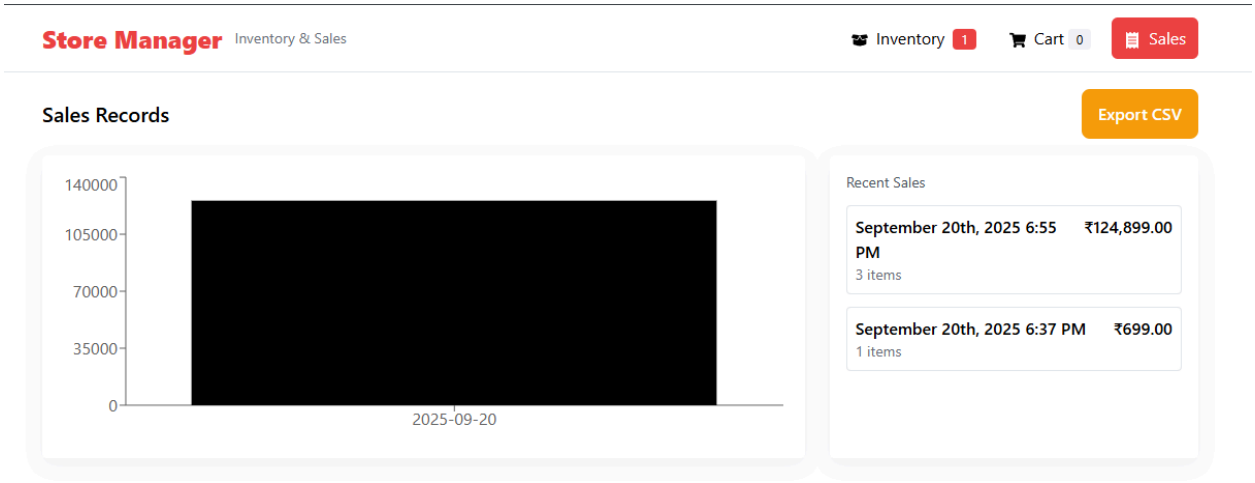
- **Product Card:** Shows image, price, stock, and action buttons (add to cart, edit, remove).



- **Cart Drawer:** Shows items in the cart, quantity controls, total calculation, and checkout/clear options.



- **Sales Page:** Lists all sales with date, items sold, and total amount in descending order.



8. Utility Features

- **Currency formatting** → consistent display of prices.
- **Sorting & Filtering** → search products by name.
- **Input Validation** → prevents negative stock, ensures valid cart quantities.
- **Error Handling** → toast notifications for invalid actions like empty cart checkout
- **CSV Export** → Sales records can be exported as CSV for reporting.

---

9. Styling

- **TailwindCSS** → modern utility-based styling.
- **Responsive Design** → mobile and desktop-friendly layouts.
- **Interactive UI Elements:** hover effects, cart drawer slide animation.
- **Low Stock Alerts** → visually highlighted.
- **Sales List** → clean, readable table/card format.

---

10. Testing & Debugging

- **Component-Level Testing:** Manual testing of ProductCard, CartDrawer, and Header.
- **State Verification:** Ensure context + reducer updates work correctly.
- **UI/UX Testing:** Verify responsiveness, drawer animation, cart behavior, and route changes.
- **Debugging:** Console logs and toast notifications used to trace errors and user actions.

---

11. Resources

- **Demo:** Watch Demo
- **Source Code:** Access Code

---

12. Known Issues

- CartDrawer on mobile sometimes overlaps page content on smaller screens.
- Sales data is only stored in memory/localStorage; no backend persistence.
- No unit tests implemented yet.

---

13. Future Enhancements

- Add backend API for persistent storage of inventory and sales.
- Implement user authentication and role-based access (manager vs cashier).
- Add product categories, filtering, and sorting.
- Include automated tests for components and reducers.
- Improve mobile responsiveness and drawer accessibility.