Store Manager: Keep Track of Inventory

1. Introduction

* **Project Title:** **Store Manager: Keep Track of Inventory**
* **Team ID: NM2025TMID41733**
* **Team Leader**: Hema Ramsan V
* **Team Members**:

1. Jaithun Safrin A
2. Janigaa k
3. Juhikha R

1. Project Overview

* **Purpose:** The Store Manager Inventory System helps businesses maintain healthy stock levels, manage sales, and track inventory efficiently. It ensures accurate real-time stock updates, prevents stockouts, and provides a clear record of all sales transactions.

* **Features:**
* Inventory management
* Automatic stock updates
* Cart system for sales
* Checkout with sales record
* Add new products (with name, image URL, price, stock, tags)
* Stock depletion alerts
* Product search functionality
* Sales records with value, items, and datetime

1. Architecture

**Component Structure:**   
The application is structured with modular React components such as Inventory, Product, Cart, Sales, and NavBar. Each module handles a specific responsibility and interacts via props and context.

**State Management**:

* Global State: Managed through Context API (CartContext, InventoryContext, SalesContext)
* Persistence: Achieved with a custom hook (useLocalStorage)
* Local State: Controlled using useState and useEffect

**Routing:**   
Basic routing is implemented using React Router, including a NotFound page for invalid routes.

1. Setup Instructions

**Prerequisites:**

* Node.js
* npm

**Installation Steps:**

1. Clone the repository:
2. git clone https://github.com/jaithunsafrin70-eng/inventory-.git
3. Navigate to the project folder:
4. cd inventory
5. Install dependencies:
6. npm install
7. Start the development server:
8. npm start
9. Folder Structure

**Client (React App):**

* **src/components** – Reusable UI components
* Cart/ → Cart & CartItem
* Inventory/ → Inventory, Product
* Product/ → Product, ProductCatalog, ProductList
* Sales/ → Sales, SaleRecord
* NavBar.jsx – Navigation component
* **src/pages** – Contains pages like NotFound.jsx
* **src/context** – Includes global contexts (CartContext, InventoryContext, SalesContext)
* **src/hooks** – Includes custom hook useLocalStorage.js
* **public/** – Static assets and base HTML file
* **tailwind.config.js** – Tailwind CSS configuratio

1. Running the Application

* **Frontend:** To run the frontend locally
* **Backend:** npm start
* **Access:** The app will be available at [http://localhost:3000](http://localhost:3000/)

1. Component Documentation

**Key Components:**

* **Inventory** – Manages product inventory
* **ProductCatalog & ProductList –** Displays product listings
* **Cart & CartItem –** Handles shopping cart functionality
* **Sales & SaleRecord** – Displays sales data
* **NavBar –** Handles navigation
* **NotFound –** Error handling page

1. State Management

**Global State:** Managed using **Context API** (Cart, Inventory, Sales).

**Local State:** Managed using React hooks (useState, useEffect).

**Persistence:** Implemented with useLocalStorage.

1. User Interface

The UI is **responsive and styled with TailwindCSS**. Screens include:

* Dashboard (inventory overview)
* Cart page (with checkout option)
* Add product form
* Sales records page
* Alerts for low stock

1. Styling

**CSS Framework:** TailwindCSS   
**Theming:** Custom Tailwind configurations for colors, spacing, and layout.

1. Testing

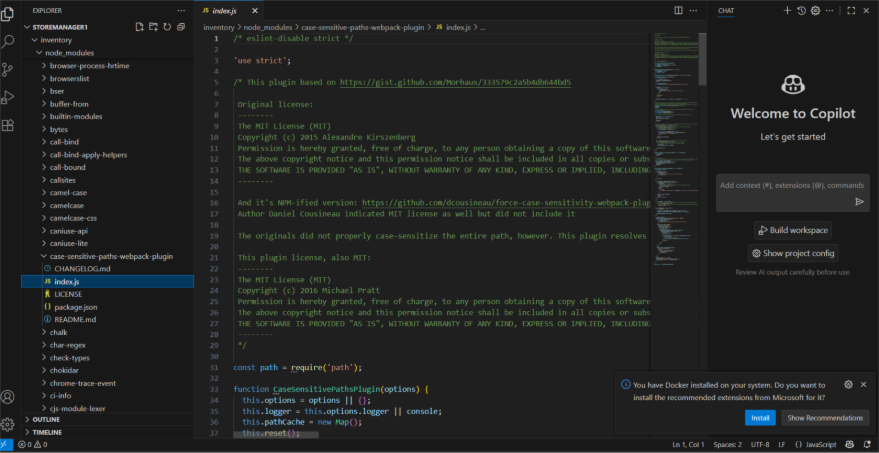
**Testing Strategy:** Jest and React Testing Library are used for unit and integration tests.

**Code Coverage:** Default CRA setup ensures test coverage for major components.

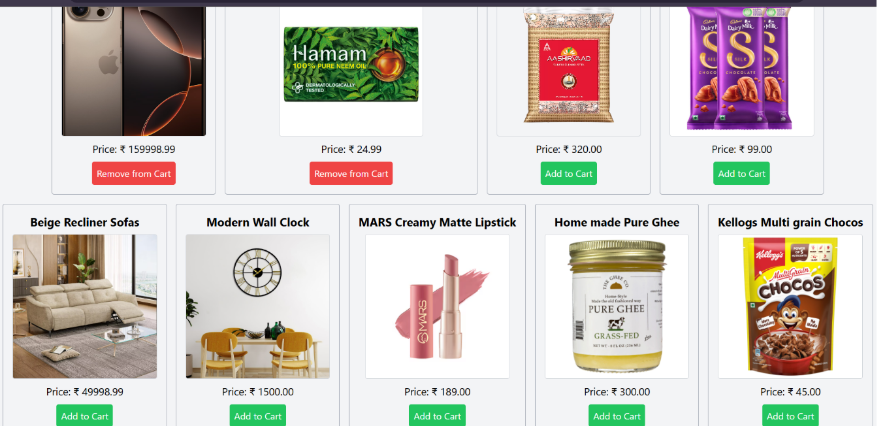
1. Screenshots or Demo

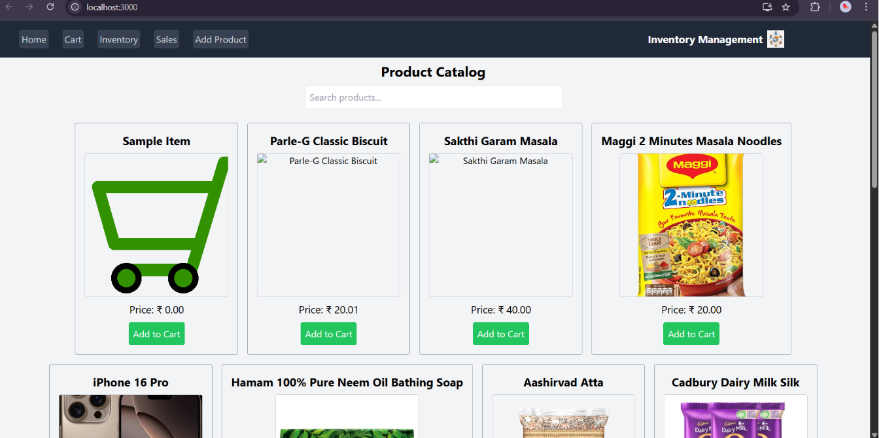
(Add screenshots of dashboard, cart, sales record, etc.)

Coding:



Output:





1. Known Issues

* Data persistence is limited to local storage.
* No backend integration for real-time data.
* Authentication features are minimal.

1. Future Enhancements

* User authentication with role-based access
* Sales reports and analytics dashboard
* Barcode/QR code integration for products
* Supplier management with auto purchase requests
* Cloud sync and mobile app support
* AI-based demand forecasting and smart restocking