Project Documentation

STORE MANAGER – KEEP TRACK OF INVENTORY

1.Introduction

System Name: Store Manager – Keep Track of Inventory

- **Objective:** Store Manager is designed to help businesses efficiently manage stock and reduce challenges such as overstocking, understocking, and manual record errors.
- **Functionality:** Provides real-time updates for inventory. Simplifies product management, sales, and purchase tracking. Automatically generates accurate sales records.
- **Benefits:** Minimizes manual errors. Improves decision-making through detailed reporting. Enhances planning and forecasting for store operations.

Team Details:

- **Team ID: NM2025TMID41764**
- **Team Leader:** SANDH**YA J (victorysandhya6@gmail.com)** - **Team Members and Their Roles:**
- **SANDHYA J– Lead (Frontend & State Management)**
- **SHERNIKA D– UI/UX Design & Navigation (sharnikadharmaraj@gmail.com)**
- **SHREE RANJANI S–Documentation & Release Management (shreeranjanisankar7@gmail.com)**
- **SONIYA JITH K– Testing & Record Management (soniyajithkathiresan@gmail.com)**

2.Project Overview

Purpose: Store Manager simplifies inventory management for all business sizes—from local shops to large retailers—by giving clear insights into stock levels, purchases, and sales.

Key Features:

1. Inventory Management
2. Stock Updates
3. Cart System
4. Checkout at Cart
5. Add New Products
6. Depleting Stock Alerts
7. Search Functionality
8. Sale Records

3.Architecture

- **Frontend:** React.js with Tailwind CSS (responsive & user-friendly)
- **Backend:** Node.js + Express.js (API and server-side logic)
- **Database:** JSON files (storing inventory, sales, and users)
- **Authentication:** JWT for secure login and role-based access

4.Setup Instructions

Prerequisites:

Node.js, It, React.js, Visual Studio Code **Steps:**

```
# Clone repository git clone
https://github.com/SANDHYAvictory/inventory/tree/main/inventory-management-frontend
 # Install client dependencies
cd      client
npm install
# Install server dependencies
cd ../server
npm install
#  Run  frontend
cd      client
npm start
#  Run  backend
cd ../server
npm start
```

Access application at: http://localhost:3000

5.Folder Structure

```
Client/
│── node_modules/
│── public/
│── src/
│     ├── components/
│     │     ├── Cart/
│     │     │     ├── Cart.jsx
│     │     │     └── CartItem.jsx
│     │     ├── Inventory/
│     │     │     ├── Inventory.jsx
│     │     │     └── Product.jsx
│     │     ├── Product/
│     │     │     ├── Product.jsx
│     │     │     ├── ProductCatalog.jsx
│     │     │     └── ProductList.jsx
│     │     ├── Sales/
│     │     │     ├── Sales.jsx
│     │     │     ├── SaleRecord.jsx
│     │     │     └── AddProduct.jsx
│     │     ├── NavBar.jsx
│
│     ├── context/
│     ├── hooks/
│     ├── pages/
│     ├── App.js
│     ├── App.css
│     ├── App.test.js
│     ├── index.js
│     ├── index.css
│     ├── Layout.jsx
│     ├── logo.svg
│     ├── reportWebVitals.js
│     └── setupTests.js
│
├── .gitignore
├── package.json
├── package-lock.json
├── README.md
└── tailwind.config.js
```

6.Running the Application

Store Manager can be run locally with **Node.js** and **React.js.**

Prerequisites:

- Node.js (LTS)
- Git **Steps:**

1.Clone repo:

- git **clone** https://github.com/SANDHYAvictory/inventory/tree/main/inventory-management-frontend
- **cd** Store_Manager

2.Install dependencies:

- **cd** client && npm install
- **cd** ../server && npm install

3.Start backend:

- **cd** server && npm start

4.Start frontend:

- **cd** client && npm start **Access:**
  - Frontend → http://localhost:3000
  - Backend API → http://localhost:5000

7.API Documentation

Inventory:

-POST /api/inventory/add – Add item
-GET /api/inventory/:id – Fetch item by ID
-PUT /api/inventory/update/:id – Update item
-DELETE /api/inventory/remove/:id – Remove item **Sales:**
-POST /api/sales/create – Create sales record **Reports:**
-GET /api/reports/summary – Fetch performance summary

8.Authentication

- Uses JWT for secure login.
- Middleware validates tokens on every protected route. - Ensures role-based access for admins and staff.

9.User Interface

- **Dashboard:** Displays stock levels and low-stock alerts.
- **Inventory Page:** Manage product details (price, stock, tags).
- **Sales & Purchases:** Record and review transactions.
- **Reports Page:** Charts and analytics for sales performance. - **Admin Panel:** Manage roles and system settings.

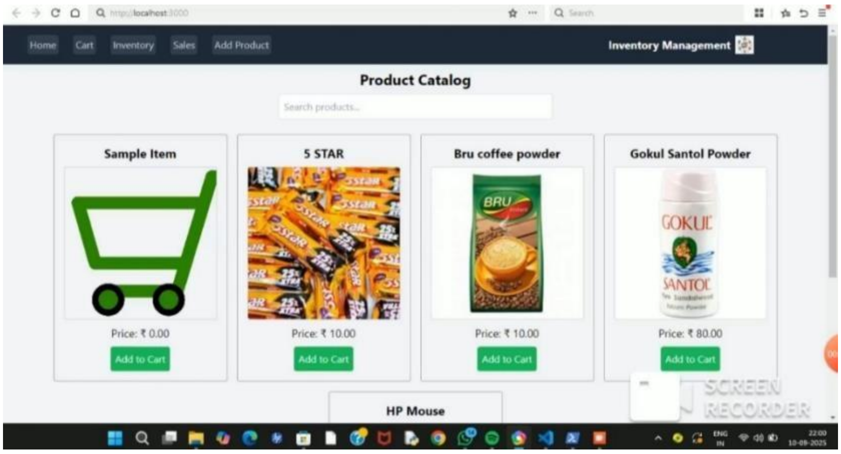
10.Testing

- **Manual Testing:** For milestone verification.
- **API Testing:** Postman used for backend validation.
- **Frontend Testing:** Chrome Developer Tools. - **Unit Testing:** For route validation (Jest).

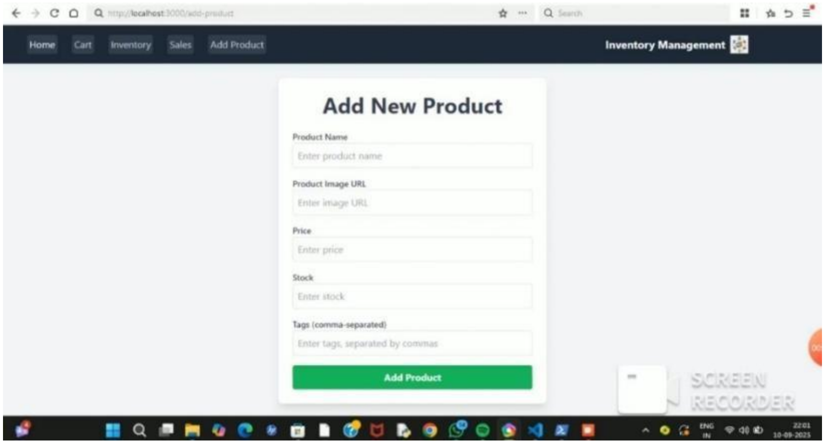
11.Screenshots or Demo

- The screenshots and demo highlight how the Store Manager system manages products, sales, and inventory effectively.

1. Product Catalog Page:



2.Add New Product Form:



Add New Product

Product Name: Enter product name

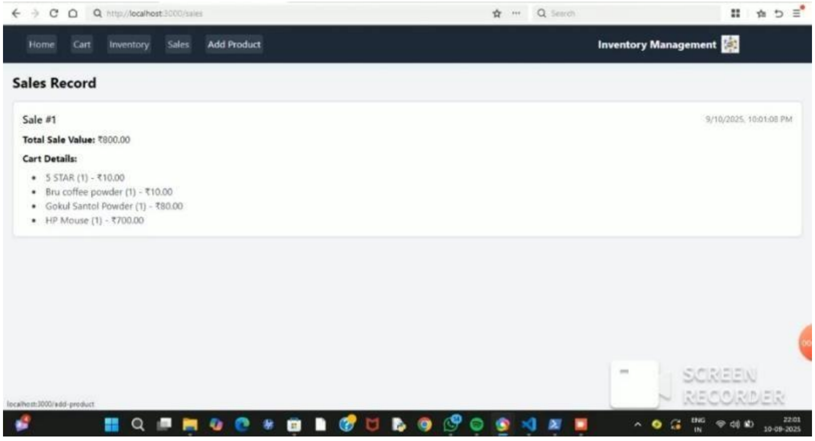
Product Image URL: Enter image URL

Price: Enter price

Stock: Enter stock

Tags (comma-separated): Enter tags, separated by commas

3.Sales Record Page:



Sales Record

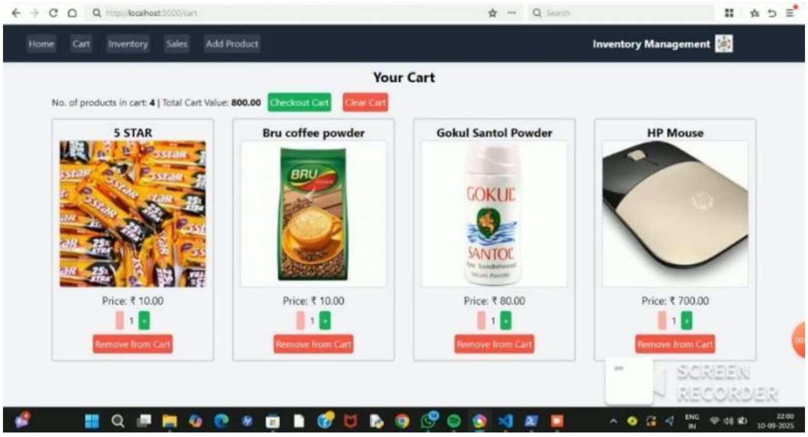
Sale #1 9/10/2025, 10:01:08 PM

Total Sale Value: ₹800.00

Cart Details:

- 5 STAR (1) - ₹10.00
- Bru coffee powder (1) - ₹10.00
- Gokul Santol Powder (1) - ₹80.00
- HP Mouse (1) - ₹700.00

4.Shopping Cart Page:



Your Cart

No. of products in cart: **4** | Total Cart Value: **800.00**

5 STAR	Bru coffee powder	Gokul Santol Powder	HP Mouse
Price: ₹ 10.00	Price: ₹ 10.00	Price: ₹ 80.00	Price: ₹ 700.00
1	1	1	1

12.Known Issues

- Limited input error handling.
- Reports only offer basic analytics.
- Currently web-only (no mobile version).

13.Future Enhancements

- Advanced predictive analytics for demand forecasting.

- Mobile application (Android & iOS).
- ERP/Accounting integration.
- AI-based stock recommendations. - Multi-language support.