STORE MANAGER: KEEP TRACK OF INVENTORY

Project documentation

1. INTRODUCTION:

PROJECT TITLE: Store Manager – Keep Track of Inventory

TEAM ID: NM2020TMID38006

TEAM LEADER: SANDHIYA.B – manimegalaib246@gmail.com

TEAM MEMBERS:

• PRIYADHARSHINI.T – priyasarala13@gmail.com

• SUWAATHIGA.V – suwaathigav@gmail.com

• NARMADHA.R – narmatharangathan4@gamil.com

2. PROJECT OVERVIEW

PURPOSE:
Store Manager is an inventory management system designed to help businesses maintain healthy stock levels, update inventory in real time, and track sales records efficiently. It provides tools for managing products, checkout processes, and stock alerts.

FEATURES:

- ✓ Real-time stock updates on product sales and new entries
- ✓ Add new products with details (name, image, price, stock, tags)
- ✓ Cart & checkout system with automatic inventory update
- ✓ Alert view for depleting stock (red highlight)
- ✓ Search functionality for products
- ✓ Sale records with value, products, and datetime
- ✓ Admin control panel

3. ARCHITECTURE

• Frontend: React.js with Tailwind CSS / Material UI

• Backend: Node.js and Express.js handling server logic and APIs

• Database: MongoDB (for products, sales, users, alerts)

• Optional AI Layer (Future): Demand prediction & auto-restocking

4. SETUP INSTRUCTIONS

Prerequisites:

• Node.js

• MongoDB

• Git

• React.js

• Express.js

• Visual Studio Code

5. INSTALLATION STEPS:

```
#Clone the repository
 git clone <repository-link>

#Install client dependencies
 cd client
 npm install

#Install server dependencies
 cd ../server
 npm install
```

6. FOLDER STRUCTURE

```
Store-Manager/
├-- client/         # React frontend
│   ├-- components/
│   ├-- pages/
│
├-- server/         # Node.js backend
│   ├-- routes/
│   ├-- models/
│   ├-- controllers/
│
├-- database/       # MongoDB schemas & configuration
```

7. RUNNING THE APPLICATION

FRONTEND:
cd client
npm start

BACKEND:
cd server
npm start

ACCESS: Visit http://localhost:3000

8. API – DOCUMENTATION

USER:
POST /api/user/register
POST /api/user/login

PRODUCTS:
GET /api/products
POST /api/products/add
PUT /api/products/:id
DELETE /api/products/:id

CART & CHECKOUT:
POST /api/cart/add
POST /api/cart/checkout

SALES:
GET /api/sales

ADMIN:
GET /api/admin/low-stock

9. AUTHENTICATION

✓ JWT-based authentication for secure login

✓ Middleware to protect user dashboards and admin routes

10. USER INTERFACE

• Landing page (product catalog & search)

• Personalized dashboard (inventory overview + alerts)

• Cart page (add/remove products & checkout)

• Admin panel (manage inventory & view reports)

• Sale records page (list of all transactions)

Testing:
- Manual testing: functional testing at milestones
- Tools: Postman, Chrome DevTools, Jest (unit testing)

11. SCREENSHOTS OR DEMO

(To be attached later with UI pages and workflow)

12. KNOWN ISSUES

✓ Inventory may lag under heavy load

✓ Manual refresh sometimes required after checkout

✓ Stock alert threshold not always precise

13. FUTURE ENHANCEMENTS

✓ Mobile application (Android/iOS)

✓ Barcode scanner integration

✓ AI-based predictive restock suggestions

✓ Multi-store support with centralized management