

Project Documentation

Store Manager: Keep track of inventory

1.Introduction.

Project Title: Store Manager: Keep Track of Inventory

Team ID : NM2025TMID42360

Team Leader :S. KANISHKA & Mail ID:kanishka357sk@gmail.com

Team member :

C.KALAIYARASI & Mail ID:kalaiyarasi728@gmail.com

S.KALPANA & Mail ID:kalpanasettu2006@gmail.com

S.Karishma & Mail ID:karishsaiyath2007@gmail.com

2. Project Overview

- PurposeThe Store Manager – Keep Track of Inventory project is designed to help shop owners and staff maintain accurate, real-time records of products and stock levels. Its main goal is to reduce manual errors, prevent stockouts or over-ordering, and give the business a clear picture of inventory at any moment.
- Features:
 - Project posting and bidding
 - Secure chat system
 - Feedback and review system
 - Admin control panel

3. Architecture

- a. Frontend: HTML/CSS/JavaScript (or react) user interface for viewing and managing products.
- b. Backend: Node.js with Express handles API requests and business logic.
- c. Database: MongoDB/MySQL stores product details, stock levels, and transactions.

4. Setup Instructions

a. Prerequisites:

- Install Node.js LTS from nodejs.org and verify with `node -v / npm -v`.
 - Windows only: In PowerShell (Admin) run `Set-ExecutionPolicy Unrestricted` and confirm.
 - Install Visual Studio Code or another code editor.
 - Download and extract the Store Manager project code.
- Installation Steps:
 - Install Node.js LTS from nodejs.org and verify with `node -v / npm -v`.
 - (Windows only) In PowerShell (Admin) run `Set-ExecutionPolicy`

Unrestricted→ press Y.

- Download & Extract the Store Manager project folder.
- Open the folder in VS Code or terminal: `cd store-manager`.
- Run `npm install` to install dependencies.
- Start the app with `npm start` and open `http://localhost:3000` in your browser.

5. Folder Structure

store-manager/

- ├─ package.json # Project info & dependencies
- ├─ README.md # Documentation
- ├─ public/ # Static assets (index.html, images, css)
- ├─ src/ # Frontend code (components, styles)
- ├─ server/ # Backend (app.js, routes, models, controllers)
- └─ database/ # Seed or migration scripts

6. Running the Application

- Install Node.js & MongoDB.
- `npm install`
- Create `.env` with DB URL & POR
- `npm start`
- Open `http://localhost:3000` in browser.

7. API Documentation

a. User:

- `/api/user/register`
- `/api/user/login`

b. Projects:

- `/api/projects/create`
- `/api/projects/:id`

c. Applications:

- `/api/apply`

d. Chats:

- `/api/chat/send`
- `/api/chat/:userId`

8. Authentication

- Register/Login Users using /auth/register and /auth/login endpoints with hashed passwords.
- Generate JWT Token on login for secure access.
- Protect Inventory Routes using middleware that verifies the JWT before allowing CRUD operations.

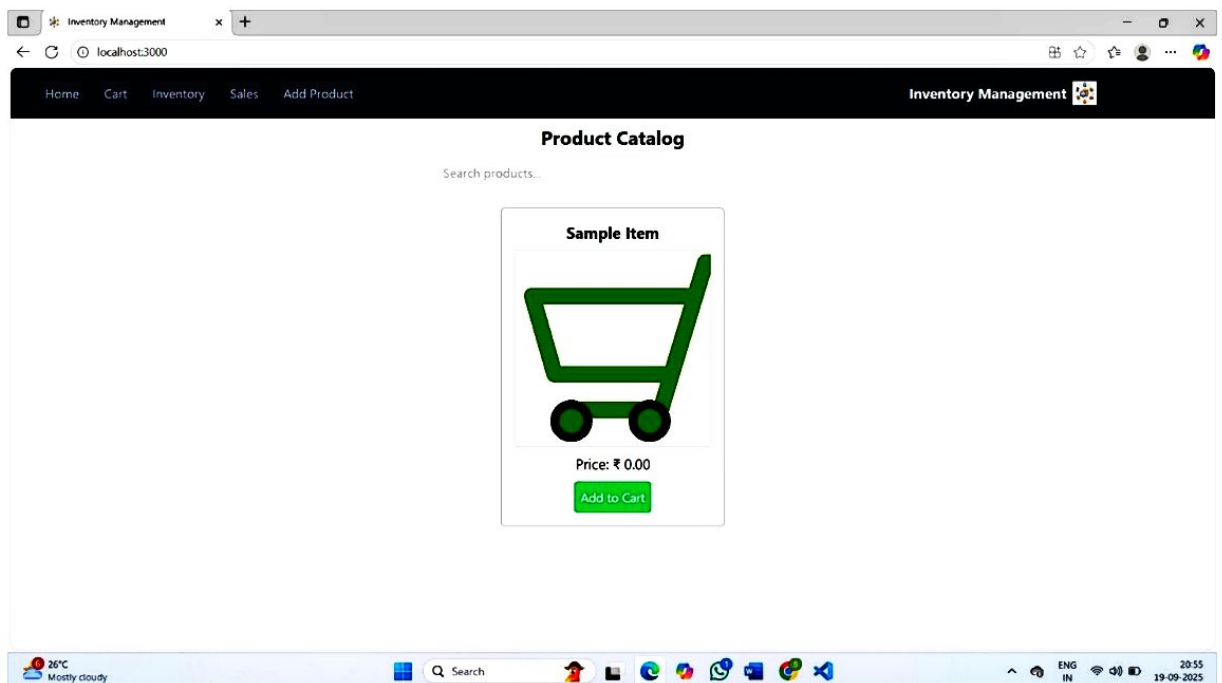
9. User Interface

- LoginPage: Username, Password, Login button.
- Dashboard: Shows total items, low stock alerts, navigation menu.
- Inventory Page: Table of items with Edit/Delete, Add Item button.
- Add/Edit Item Page: Form for name, quantity, price.
- Notifications: Success/failure messages.

10. Testing

- UnitTests: Test functions like add, update, delete items.
- APITests: Verify endpoints, responses, and JWT auth using Postman.
- UITests: Check forms, buttons, and dashboard functionality manually or with Cypress.

11. Screenshots or Demo



12.Known Issues

- No role-based access control; all users have full permissions.
- Limited input validation; negative quantities or prices may be accepted.

13.Future Enhancements

- Add role-based access (admin/staff).
- Enable real-time stock alerts.
- Generate inventory reports.
- Make UI mobile-friendly.
- Support barcode/QR code scanning.