PROJECT DOCUMENTATION

STORE MANAGER-KEEP TRACK ON INVENTORY

1.INTRODUCTION:
□ PROJECT TITLE :
STORE MANAGER-KEEP TRACK ON INVENTORY
□ TEAM ID:
NM2025TMID38644
 □ TEAM LEADER: • NAME: Harini B • MAIL ID: mgc8harini24@gmail.com □ TEAM MEMBERS: • NAME:Nandhini K
MAIL ID: mgc8nandhini24@gmail.com
NAME:Ramya devi S
MAIL ID: mgc8ramya24@gmail.com
NAME: N Oviya
MAIL ID: mgc8oviya24gmail.com
2 .PROJECT OVERVIEW:
□ PURPOSE:
Many small and medium businesses rely on spreadsheets or manual methods to track stock, which can lead to errors and stockouts. This project provides a streamlined digital solution that automates tracking, reduces manual effort, and helps maintain optimal inventory levels.
□ Features:

• Add, update, and delete product records.

• Monitor real-time stock quantities.

- Search and filter products by name, category, or availability.
- Generate reports such as total inventory value and low-stock alerts.

3. ARCHITECTURE:

□ FRONTEND:

- React.js with Bootstrap and Material UI
- Provides a responsive, component-based interface for managing inventory.
- Handles user interactions such as adding, editing, and deleting products.
- Communicates with the backend via RESTful API calls (Axios/Fetch).

∩ BACKEND:

- Node.js with Express.js
- Implements all server-side logic and API endpoints.
- Validates requests, applies business rules (e.g., low-stock checks), and formats responses as JSON.
- Handles authentication and authorization for admin users if required.

□ DATABASE:

- Stores product details, stock levels, user credentials, and transaction history.
- Uses a document-oriented schema for flexible inventory data.
- Supports indexing for fast search and filtering.

4. SETUP INSTRUCTIONS:

PREREQUISITES:

- Node.js (v16 or later).
- MongoDB (local or cloud instance such as MongoDB Atlas).
- Git (for version control and cloning the repository).
- React.js (via Create React App or Vite).
- Express.js (Node.js framework for backend APIs).
- Mongoose (MongoDB object modeling tool).
- Visual Studio Code (or any preferred code editor).

INSTALLATION STEPS:

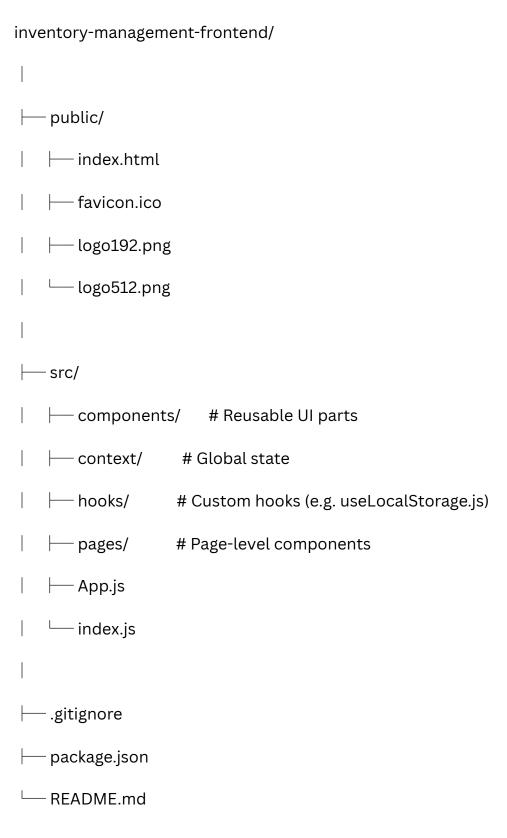
- #clone the repositary git clone
- #install client dependencies cd

Client npm install

• #install server dependencies cdcd

.. /server npm install

5. FOLDER STRUCTURE:



6. RUNNING THE APPLICATION:

□ BACKEND:

Open a terminal, navigate to the server folder, and start the Node.js server so it car handle API requests and database operations.
□ FRONTEND:
In a separate terminal, go to the client folder and start the React development server to serve the user interface.
□ ACCESS:
Once both servers are running, open a web browser and visit http://localhost:3000 to use the application.
7. API DOCUMENTATION:
□ PRODUCTS:
/api/products
/api/products/:id
/api/products/create
/api/products/update/:id
/api/products/delete/:id
□ CATEGORIES:
/api/categories
/api/categories/create
□ USERS:
/api/users/register
/api/users/login
□ REPORTS:
/api/reports/lowstock
/api/reports/summary

8. AUTHENTICATION:

- JWT-based authentication for secure admin login and session management.
- Middleware verifies tokens and protects all product management routes (create, update, delete) so only authorized users can access them.

9. USER INTERFACE:

- Landing Page entry point and overview.
- Inventory Dashboard real-time stock summary.
- Product Management Page add, edit, delete products.
- Reports & Analytics Page low-stock and inventory value reports.
- Admin Login Page secure access for authorized users.

10. TESTING:

- Manual testing performed at each development milestone.
- Tools: Postman for API endpoint verification, Chrome DevTools for frontend debugging and performance checks.