**Project Documentation**

**Project Title: Store Manager – Keep Track of Inventory**

**1. Introduction**

**Project Title**: Store Manager – Keep Track of Inventory

**Team ID:** NM2025TMID30065

**Team Leader:** VARSHINI.S – 202400500@sigc.edu

**Team Members:** BHARANI SRI.D.J - 202400406@sigc.edu

PRIYA DHARSHINI.S – 202400771@sigc.edu

RITHIKA.G - 202400206@sigc.edu

**2. Project Overview**

**PURPOSE:**

The Store Manager application helps businesses efficiently manage their inventory by keeping track of stock levels, sales, purchases, and suppliers in real-time.

**FEATURES:**

* Product management (add, update, delete products)
* Track stock levels and auto alerts for low inventory
* Supplier and customer management
* Sales and purchase records
* Dashboard with analytics & reports
* Secure user authentication (Admin & Staff roles)

**3. Architecture**

* **Frontend:** React.js with Bootstrap / Material UI
* **Backend:** Node.js + Express.js (REST APIs)
* **Database:** MongoDB (stores product data, suppliers, sales, and purchase history)

**4. Setup Instructions**

**Prerequisites:**

* Node.js
* MongoDB
* Git
* React.js
* Express.js – Mongoose – Visual Studio Code

**Installation Steps:**

* Clone the repository

git clone <https://github.com/RithikaGovindharaju/Store-Manager.git>

* Install client dependencies

cd client

* npm install

Install server dependencies

cd ../server

npm install

**5. Folder Structure**

Store-Manager/

|-- client/ # React frontend

| |-- components/

| |-- pages/

|

|-- server/ # Node.js backend

| |-- routes/

| |-- models/

| |-- controllers/

**6. Running the Application**

**Frontend:**

* cd client
* npm start

**Backend:**

* cd server
* npm start

Access: Visit <http://localhost:3000>

**7. API Documentation**

**User:**

* POST /api/user/register
* POST /api/user/login

**Products:**

* POST /api/products/create
* GET /api/products/:id
* PUT /api/products/:id
* DELETE /api/products/:id

**Inventory:**

* GET /api/inventory
* PUT /api/inventory/update/:id

**Sales:**

* POST /api/sales/add
* GET /api/sales/report

**8. Authentication**

* JWT-based authentication for secure login
* Middleware protects private routes (only admins can add/remove products)

**9. User Interface**

* Admin Dashboard
* Staff Dashboard
* Product Management Page
* Inventory Overview Page
* Sales & Reports Page

**10. Testing**

* Manual testing during milestones
* **Tools:** Postman, Chrome Dev Tools

**11. Screenshot Demo**

[**https://drive.google.com/file/d/1zL9N0M8iwtqcU7nj137Rz1SgFSTsVV3O/view?usp=drivesdk**](https://drive.google.com/file/d/1zL9N0M8iwtqcU7nj137Rz1SgFSTsVV3O/view?usp=drivesdk)

**12. Known Issues**

* Minor delays may occur when updating stock levels in real-time.
* Limited error handling for incorrect product data entries.
* Currently supports only manual product entry (no barcode/QR scanning yet).
* User interface may need further optimization for mobile screens.

**13. Future Enhancements**

* Barcode/QR code scanning for products
* Automated purchase orders when stock is low
* Mobile app integration
* Advanced analytics with graphs & AI-based demand prediction