Project Documentation - Store Manager

1. Introduction

- Project Title: Inventory Management System

Team Members:

Dhatchayani.R

Gayathiri.J

Gopika.T

Jeevitha.G

2. Project Overview

Purpose: The Store Manager application is designed to help store owners efficiently manage inventory, sales, billing, and customer data. It simplifies stock tracking, billing, and reporting.

Features:

Add, update, and delete productsManage inventory and stock levelsGenerate bills aninvoicesCustomer managementSales and revenue reports

3. Architecture

Component Structure:

Product Component add/edit/view products

Customer Componentmanage customer details

Billing Componentgenerate and view invoices

Dashboard Component display sales, stock, and analytics

State Management:

Use Context API/Redux for handling global state like product list, cart, and reports.

Routing:

React Router for navigation between pages (Products, Customers, Billing, Reports).

4. Setup Instructions

Prerequisites: Node.js, npm, React, Database (MongoDB/MySQL).

Installation:

a. Clone repository: git clone [repo link]

b. Install dependencies: npm install

c. Configure.env file with DB credentials.

d. Run the app: npm start

5. Folder Structure

Client:/src/components for Ul components, /src/pages for main screens.

Utilities: Helper functions (e.g., validation, calculations).

6. Running the Application

Frontend: npm start inside client directory.

Backend: npm run server (if backend is separate).

7. Component Documentation

Key Components:

Product form (add/edit products)

CustomerForm (manage customers)

BillGenerator (generate invoice)

Report Chart (sales reports)

Reusable Components:

Button, InputField, Navbar, Sidebar

8. State Management

Global State: Manage product list, cart, and user data globally using Redux/Context API.

Local State: Form inputs and modal states managed locally inside components.

9. User Interface

Screenshots of:

Dashboard with analytics.

Product management page

Billing page

Reports page

10. Styling

CSS Frameworks/Libraries: Tailwind CSS / Bootstrap.

Theming: Custom color palette for store branding.

11. Testing

Testing Strategy: Component testing using Jest + React Testing Library.

Code Coverage: Ensure product CRUD and billing workflows are fully tested.

12. Screenshots or Demo

Provide Ul screenshots (Product Page, Billing Page, Dashboard).

13. Known Issues

Limited offline support.

Reports may take longer to load with large data.

14. Future Enhancements

Add barcode scanner integration.

Enable multi-store support.

Al-based demand forecasting for products.

Mobile app version for on-the-go management.