Frontend Development with React.js

Project Documentation

1. Introduction

Project Title: Store Manager – Keep Track of Inventory

Team Leader: R. Sandhya

Team Members: R. Saritha, K. Savithri

2. Project Overview

Purpose: The Store Manager project is designed to help manage store inventory efficiently. It maintains healthy stock levels, updates stock automatically during sales, and provides features for product management, checkout, and sales record keeping.

Features:

Inventory Management – Helps maintain healthy stock levels.

Stock Updates – Stock updates automatically on sales and when adding new stock.

Cart – Products can be added with specific quantities for a sale.

Checkout – Clears the cart, updates stock, and records sales.

Add Products – Add new products with name, image URL, price, stock, and tags.

Alert View – Highlights low-stock products with alerts.

Search Functionality – Search inventory and product catalog.

Sale Records – Stores records with sale value, products, and datetime.

3. Architecture

Component Structure: The project includes components such as Navbar, InventoryList, ProductCard, Cart, Checkout, AddProductForm, StockAlert, and SalesRecords. These interact to provide a smooth user experience.

State Management: Global state is managed with Context API/Redux, and local state is managed with React hooks (useState/useReducer).

Routing: Implemented using React Router with routes for Inventory, Cart, Add Product, and Sales Records.

4. Setup Instructions

Prerequisites: Node.js, npm/yarn, and MongoDB (local or Atlas).

Installation:

Clone the repository: git clone

Navigate into the folder: cd store-manager

Install dependencies: npm install

Set up environment variables in .env (API URL).

Run the application: npm start

5. Folder Structure

Client folder contains React application organized as follows:

store-manager/ ■■■ public/ ■■■ src/ ■ ■■■ assets/ # Images, icons ■ ■■■ components/ # Navbar, ProductCard, StockAlert ■ ■■■ pages/ # InventoryList, Cart, AddProductForm, SalesRecords ■ ■■■ context/ # Context API for global state ■ ■■■ hooks/ # Custom hooks for API/data fetching ■ ■■■ utils/ # Helper functions ■ ■■■ App.js ■ ■■■ index.js ■■■ package.json

6. Running the Application

Frontend: Run the following command in the client directory:

npm start

7. Component Documentation

Key Components:

InventoryList – Displays all products with stock and search functionality.

ProductCard – Shows product details (name, price, stock, image URL).

Cart – Manages selected products for checkout.

Checkout – Finalizes purchase, clears cart, updates inventory.

AddProductForm – Allows adding new products.

StockAlert – Highlights products with low stock.

SalesRecords – Displays past sales records.

8. State Management

Global State: Maintains inventory data, sales records, and cart items across the application.

Local State: Manages temporary states such as form inputs and cart item counts.

9. User Interface

The user interface consists of the following pages: Inventory Page, Cart Page, Add Product Page, Sales Records Page, and Stock Alerts. Products are displayed in a grid view with search functionality.

10. Styling

CSS Frameworks/Libraries: Tailwind CSS is used for styling.

Theming: Green color is used for available stock, red for low stock, with consistent design system.

11. Testing

Testing Strategy: Unit testing using React Testing Library, integration testing with Jest, and mock API testing using MSW.

Code Coverage: Istanbul (via Jest) is used for test coverage reports.

12. Screenshots or Demo

Provide UI screenshots or link to deployed demo (if available).

13. Known Issues

Known issues include lack of authentication system, possible performance issues with large datasets, and limited analytics features.

14. Future Enhancements

Add authentication (Admin/User roles).

Introduce product categories and filtering.

Add sales analytics dashboard with charts.

Barcode scanner integration.

Export sales reports to PDF/Excel.