Project Documentation: Inventory Management Frontend

# Introduction

Project Title: Inventory Management Frontend **Store Manager: Keep Track of Inventory .**

Team Leader : THULASHI J

Team Members:

LEKHA SHREE R

NANDHINI R

SARIKA T

# Project Overview

Purpose: The Inventory Management Frontend is a React.js application designed to manage products, inventory, sales, and carts in a structured manner. It provides an intuitive user interface for small and medium businesses to track and manage stock efficiently.

Features:

• Product catalog with add, list, and detail views  
• Inventory tracking with context management  
• Sales record management  
• Shopping cart functionality  
• Responsive navigation bar  
• Error handling with NotFound page

# Architecture

Component Structure: The project is structured around reusable components located in the `components` folder, organized by feature modules such as Product, Inventory, Cart, and Sales. Layout and navigation components are separated for modularity.

State Management: The application uses the React Context API (`CartContext`, `SalesContext`, `InventoryContext`) to handle global state.

Routing: Implemented using `react-router-dom` for page navigation (e.g., product listing, cart, sales, inventory).

# Setup Instructions

Prerequisites:  
• Node.js (>=14)  
• npm or yarn package manager

Installation:  
1. Clone the repository  
2. Navigate to `inventory-management-frontend`  
3. Run `npm install` to install dependencies  
4. Configure environment variables if necessary

# Folder Structure

Client:  
- `src/components`: Contains all React components categorized by module (Product, Inventory, Cart, Sales)  
- `src/context`: Holds global state management files using React Context API  
- `src/hooks`: Custom hooks such as `useLocalStorage`  
- `src/pages`: Contains page-level components like `NotFound`  
- `src/assets`: Static files (images, icons, etc.)  
- `src/App.js`: Main application entry point  
- `src/index.js`: Renders the app into the DOM

# Running the Application

To start the application locally:  
  
`npm start` (inside the `inventory-management-frontend` directory)

# Component Documentation

Key Components:  
• AddProduct.jsx – Form to add new products  
• ProductList.jsx – Displays a list of products  
• Inventory.jsx – Manages product stock  
• Cart.jsx – Displays and manages items in the cart  
• Sales.jsx – Handles and displays sales records  
• NavBar.jsx – Provides navigation between pages

Reusable Components:  
• Product.jsx – Reusable product display component  
• CartItem.jsx – Reusable cart item component

# State Management

Global State:  
• CartContext – Manages cart items and operations  
• InventoryContext – Tracks product inventory  
• SalesContext – Manages sales records

Local State:  
• Managed within individual components using `useState` where necessary (e.g., input forms, toggles).

# User Interface

The application provides a clean, responsive UI built with React and TailwindCSS. It includes navigation, forms for adding products, tables for listing data, and modular pages for products, sales, and cart.

# Styling

CSS Frameworks/Libraries: TailwindCSS is used for utility-first styling.  
Theming: The app uses customizable Tailwind configuration for consistent color schemes and typography.

# Testing

Testing Strategy: Basic tests are included using Jest and React Testing Library (`App.test.js`, `setupTests.js`).  
Code Coverage: Coverage can be generated using Jest with the `--coverage` flag.

# Screenshots or Demo

Provide screenshots of key pages (Product List, Inventory, Cart, Sales) or deploy the app and include a demo link.

# Known Issues

• No backend integration is included, so data persistence is limited to local storage.  
• Authentication and role-based access control are not implemented.

# Future Enhancements

• Integration with a backend API for persistent storage  
• User authentication and authorization  
• Advanced reporting and analytics  
• Export functionality for sales and inventory reports  
• Improved test coverage and CI/CD integration