**PROJECT DOCUMENTATION**

**Store Manager:Keep Track of Inventory**

**1. Introduction**

**Project Title**

**Store Manager:Keep Track of Inventory**

**Team Leader –** Nandhini R **–** [nandhiniramu10@gmail.com](https://d.docs.live.net/7565DA48565B3CBE/Documents/nandhiniramu10@gmail.com)

**Team Members:**

* Roshini K – [roshini2007ro@gmail.com](https://d.docs.live.net/7565DA48565B3CBE/Documents/roshini2007ro@gmail.com)
* Santhini K **–** [santhini1422007@gmail.com](https://d.docs.live.net/7565DA48565B3CBE/Documents/santhini1422007@gmail.com)
* Santhiya C **–** [c.santhiya7311@gmail.com](https://d.docs.live.net/7565DA48565B3CBE/Documents/c.santhiya7311@gmail.com)
* Santhiya C **–** [ranjithya80@gmail.com](https://d.docs.live.net/7565DA48565B3CBE/Documents/ranjithya80@gmail.com)

**Project Description**

The **Inventory Management System** is a **React.js-based web application** designed to streamline inventory operations. It enables businesses to **manage products, track sales, and monitor stock** with a clean and responsive interface powered by **TailwindCSS**.

With integrated modules for **Products, Cart, Inventory, and Sales**, the application ensures businesses can efficiently handle day-to-day transactions while keeping data consistent through global state management using **React Context API**.

**2. Project Overview**

* **Purpose**  
  To provide businesses with an intuitive tool to manage their **products, sales, and inventory records** in real-time.
* **Goals**
  + Enable product catalog management (add, list, view).
  + Provide cart and sales tracking functionality.
  + Implement inventory stock tracking.
  + Deliver a responsive UI using TailwindCSS.
* **Key Features**
  + **Product Management**: Add, edit, and list products.
  + **Inventory Tracking**: Monitor stock levels.
  + **Sales Management**: Track sales and generate records.
  + **Cart Management**: Add/remove items in the cart.
  + **Persistent Storage**: Data retained via custom hooks (localStorage).

**3. Architecture**

**Component Structure**

* **App.js / Layout.jsx** → Root components defining app layout.
* **NavBar.jsx** → Global navigation bar.

**Product Module**

* Product.jsx → Product details.
* ProductList.jsx → Displays all products.
* ProductCatalog.jsx → Categorized product listing.
* AddProduct.jsx → Form to add a new product.

**Inventory Module**

* Inventory.jsx → Handles stock overview.
* Inventory/Product.jsx → Inventory-specific product component.

**Cart Module**

* Cart.jsx → Displays user’s shopping cart.
* CartItem.jsx → Individual item in cart.

**Sales Module**

* Sales.jsx → Sales management dashboard.
* SaleRecord.jsx → Displays individual sales records.

**State Management**

* **CartContext.js** → Manages cart operations (add/remove items).
* **InventoryContext.js** → Stores and updates inventory stock.
* **SalesContext.js** → Manages sales data and history.

**Custom Hooks**

* **useLocalStorage.js** → Persists state in localStorage to keep data after refresh.

**Routing**

* Routes are handled in App.js with a fallback:
  + / → ProductCatalog / Home
  + /inventory → Inventory.jsx
  + /cart → Cart.jsx
  + /sales → Sales.jsx
  + \* → NotFound.jsx

**4. Setup Instructions**

**Prerequisites**

* Node.js (>= 16)
* npm or yarn
* Git
* React.js knowledge (hooks, context, routing)

**Installation**

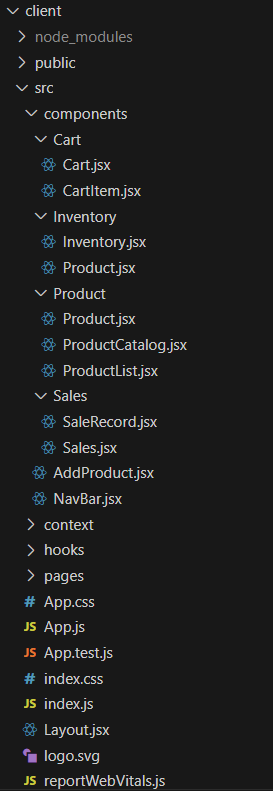
git clone https://github.com/<org>/inventory-management-frontend.git

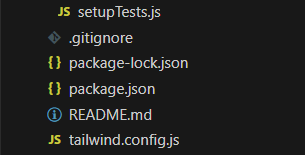
cd inventory-management-frontend

npm install

npm start

**5. Folder Structure**

****

****

**6. Running the Application**

* Start development server:
* npm start
* Build for production:
* npm run build
* Run tests:
* npm test

**7. Component Documentation**

**NavBar.jsx**

* Provides navigation to Products, Inventory, Cart, and Sales.

**Product Components**

* **Product.jsx** → Displays individual product.
* **ProductList.jsx** → Renders product collection.
* **ProductCatalog.jsx** → Structured view of product categories.
* **AddProduct.jsx** → Form for adding products to catalog.

**Cart Components**

* **Cart.jsx** → Shows items added to cart.
* **CartItem.jsx** → Displays individual cart item with quantity and price.

**Sales Components**

* **Sales.jsx** → Dashboard for all sales records.
* **SaleRecord.jsx** → Shows individual sale details.

**Inventory Components**

* **Inventory.jsx** → Displays all stock.
* **Inventory/Product.jsx** → Item-level inventory tracking.

**8. State Management**

* **CartContext.js** →
  + Functions: addToCart, removeFromCart, clearCart.
  + State: cartItems.
* **InventoryContext.js** →
  + Tracks available products and stock levels.
  + Updates on product addition/removal.
* **SalesContext.js** →
  + Stores sales history and calculates totals.
* **useLocalStorage.js** →
  + Persists cart, inventory, and sales data in browser localStorage.

**9. User Interface**

* **Home/Product Catalog** — Browse available products.
* **Inventory Page** — Check stock details.
* **Cart Page** — Manage selected products.
* **Sales Page** — Review transaction history.
* **Responsive Design** — Styled with TailwindCSS.

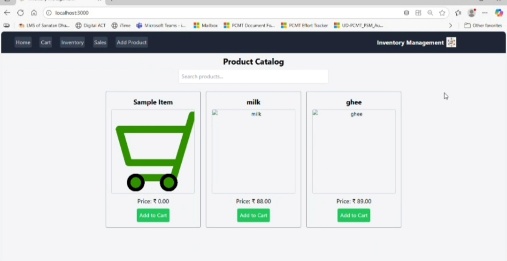
**10. Styling**

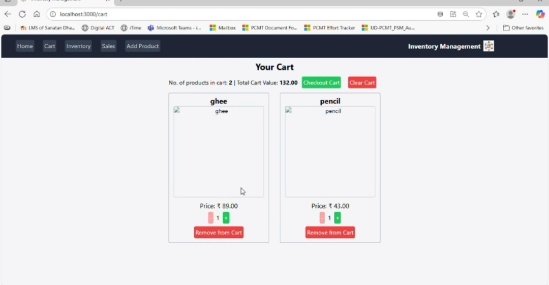
* **Framework:** TailwindCSS
* **Approach:** Utility-first styling for responsive layouts.
* **Theme:** Clean, minimal, business-friendly UI.

**11. Testing**

* **Tools:** Jest, React Testing Library.
* **Unit Tests:** Example in App.test.js.
* **Setup:** setupTests.js.

**12. Screenshots / Demo**





[**https://drive.google.com/file/d/15qQRb-CY0MbAMxv\_Xf0SmNsGZi\_SggW6/view?usp=drive\_link**](https://drive.google.com/file/d/15qQRb-CY0MbAMxv_Xf0SmNsGZi_SggW6/view?usp=drive_link)

**13. Known Issues**

* Data persistence limited to **localStorage** (no backend).
* No authentication for multiple users.

**14. Future Enhancements**

* Integrate **backend API** for persistent data.
* Add **authentication & role-based access** (admin, staff).
* Export **sales reports** to PDF/Excel.
* Add **search & filtering** in inventory and products.
* Implement **real-time updates** via WebSockets.