**Store Manager: Keep Track of Inventory**

1. **Introduction**

**Project Title :** **Store Manager: Keep Track of Inventory**  
**Team Members:**

1. **Gokulnathan GS** – Lead Developer (Frontend & State Management)

**2. H. Nicholas Sharon Merin** – UI/UX Designer & Frontend Developer

**3. Gokul P** – Testing & Quality Assurance

**4. Jawahar V** – Documentation & Deployment Support

1. **Project Overview**

**Purpose:**  
The Store Manager application is designed to streamline **inventory, cart, and sales management** for small businesses. It provides a clean and responsive interface to add, track, and manage store products, while offering essential cart and sales functionalities.

**Features:**

* Inventory Management (add, update, delete products)
* Product Catalog and Listings
* Cart Management with persistence
* Sales Records and Tracking
* Navigation and Routing
* Error Page for invalid routes

1. **Architecture**

**Component Structure:**  
The application is structured with modular React components such as Inventory, Product, Cart, Sales, and NavBar. Each module handles a specific responsibility and interacts via props and context.

**State Management:**

* **Global State**: Managed through **Context API** (CartContext, InventoryContext, SalesContext)
* **Persistence**: Achieved with a custom hook (useLocalStorage)
* **Local State**: Controlled using useState and useEffect

**Routing:**  
Basic routing is implemented using **React Router**, including a NotFound page for invalid routes.

1. **Setup Instructions**

**Prerequisites:**

* Node.js
* npm

**Installation Steps:**

1. Clone the repository:
2. git clone https://github.com/GreatGokul/Gokulnathan-store-manager.git
3. Navigate to the project folder:
4. cd Gokulnathan-store-manager
5. Install dependencies:
6. npm install
7. Start the development server:
8. npm start
9. **Folder Structure**

**Client (React App):**

* **src/components** – Reusable UI components
  + Cart/ → Cart & CartItem
  + Inventory/ → Inventory, Product
  + Product/ → Product, ProductCatalog, ProductList
  + Sales/ → Sales, SaleRecord
  + NavBar.jsx – Navigation component
* **src/pages** – Contains pages like NotFound.jsx
* **src/context** – Includes global contexts (CartContext, InventoryContext, SalesContext)
* **src/hooks** – Includes custom hook useLocalStorage.js
* **public/** – Static assets and base HTML file
* **tailwind.config.js** – Tailwind CSS configuration

1. **Running the Application**

To run the frontend locally:

npm start

The app will be available at [**http://localhost:3000**](http://localhost:3000/)

1. **Component Documentation**

**Key Components:**

* Inventory – Manages product inventory
* ProductCatalog & ProductList – Displays product listings
* Cart & CartItem – Handles shopping cart functionality
* Sales & SaleRecord – Displays sales data
* NavBar – Handles navigation
* NotFound – Error handling page

**Reusable Components:**

* Product, ProductList, and CartItem are reusable across different contexts.

1. **State Management**

* **Global State:** Managed using **Context API** (Cart, Inventory, Sales).
* **Local State:** Managed using React hooks (useState, useEffect).
* **Persistence:** Implemented with useLocalStorage.

1. **User Interface**

The UI is **responsive and styled with TailwindCSS**. Screens include:

* Product catalog
* Inventory management
* Cart view
* Sales records

1. **Styling**

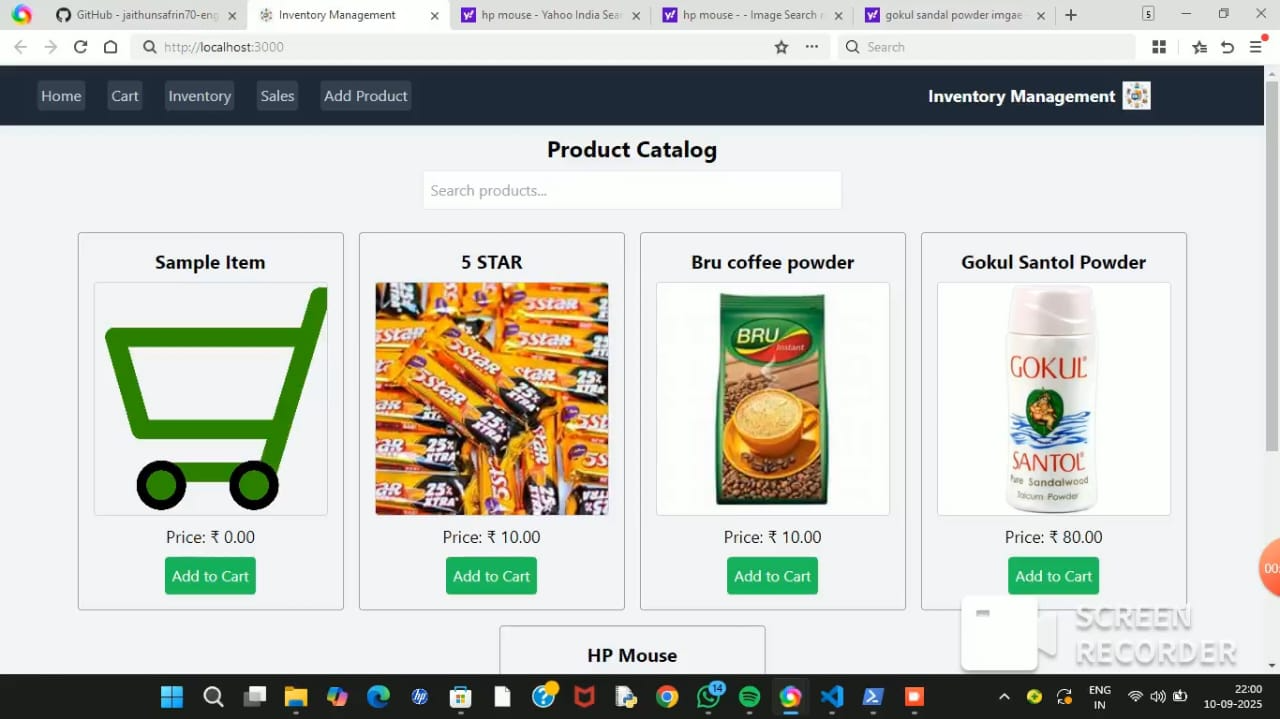
**CSS Framework:** TailwindCSS  
**Theming:** Custom Tailwind configurations for colors, spacing, and layout.

1. **Testing**

* **Testing Strategy:** Jest and React Testing Library are used for unit and integration tests.
* **Code Coverage:** Default CRA setup ensures test coverage for major components.

1. **Screenshots or Demo**

*(screenshots of the UI — product catalog, cart page, sales view, etc.)*



1. **Known Issues**

* Data persistence is limited to local storage.
* No backend integration for real-time data.
* Authentication features are minimal.

1. **Future Enhancements**

* Add backend API integration for real-time product & sales data.
* Implement authentication & user roles.
* Enhance dashboard with analytics and charts.
* Expand test coverage with end-to-end testing.
* Add export/import functionality for sales data.