**Store Manager: Keep Track of Inventory**

**1. Introduction**

**Team Members**

* **MAHESH KRISHNA M (TEAM LEADER)**
* **GUHAN KARTHICK S A**
* **HARIHARAN E**
* **AJAY R**
* **MANOJ S K**

**2. Project Overview**

**Purpose:**  
Store Manager is a web application designed to help store owners and managers keep track of their inventory efficiently. It allows users to manage products, view stock levels, track sales, receive low-stock alerts, and generate inventory reports. The goal is to simplify store operations and reduce manual effort.

**Features:**

* Add, edit, and delete products
* Track product quantity, price, and supplier details
* Low-stock alerts and notifications
* Search and filter products by name, category, or stock level
* Generate inventory and sales reports
* Responsive design for mobile and desktop

**3. Architecture**

**Component Structure**

* **App.js** – Main entry point that includes routing and layout components
* **Header** – Navigation bar with links to Home, Inventory, Reports, and Profile
* **ProductList** – Displays all products with stock, price, and category info
* **ProductCard** – Individual product card showing details and actions
* **AddProductForm** – Allows users to add or edit product information
* **InventoryDashboard** – Displays analytics such as total stock, low-stock items, etc.
* **Reports** – Generates inventory and sales reports
* **SearchBar** – Allows users to search and filter products
* **UserPreferences** – Manages user profile and app settings

**State Management**

* Uses **Context API** to handle global state (products, user preferences, and app theme).
* Local component states are managed using **useState** for form inputs and search queries.

**Routing**

* Implemented using **React Router v6** with the following routes:
  + / – Home page with inventory dashboard
  + /inventory – Displays all products and allows managing inventory
  + /reports – Displays inventory/sales reports
  + /profile – User settings and preferences

**4. Setup Instructions**

**Prerequisites**

* Node.js version 18 or above
* npm version 9 or above

**Installation**

1. Clone the repository:

git clone <https://github.com/Maheshsona/inventory.git>

1. Navigate to the client directory:
2. cd store-manager/client
3. Install dependencies:
4. npm install
5. Create a .env file with any required environment variables (if using APIs for sales tracking, etc.)
6. Start the development server:
7. npm start

**5. Folder Structure**

client/

├── public/

├── src/

│ ├── components/

│ │ ├── ProductCard.js

│ │ ├── ProductList.js

│ │ ├── AddProductForm.js

│ │ ├── Header.js

│ │ ├── InventoryDashboard.js

│ │ ├── Reports.js

│ │ └── SearchBar.js

│ ├── context/

│ │ └── AppContext.js

│ ├── pages/

│ │ ├── Home.js

│ │ ├── Inventory.js

│ │ ├── ReportsPage.js

│ │ └── Profile.js

│ ├── utils/

│ │ └── localStorage.js

│ ├── App.js

│ ├── index.js

├── .env

├── package.json

└── README.md

**Utilities**

* **localStorage.js** – Handles storing and retrieving inventory data from local storage
* **custom hooks** – useInventory for inventory operations and useTheme for theme preferences

**6. Running the Application**

Run the following command in the client directory:

npm start

This will launch the application locally at [**http://localhost:3000/**](http://localhost:3000/).

**7. Component Documentation**

**Key Components**

* **ProductList**  
  Displays all products in the inventory  
  Props: products, onDelete, onEdit
* **ProductCard**  
  Displays individual product details (name, price, stock, category)  
  Props: product object
* **AddProductForm**  
  Allows adding and editing product information  
  Props: onSubmit, existingProduct
* **InventoryDashboard**  
  Shows inventory stats like total products, low stock count, etc.  
  Props: products
* **Reports**  
  Generates downloadable inventory and sales reports  
  Props: products

**Reusable Components**

* **SearchBar**  
  Handles keyword search input  
  Props: onSearch callback
* **Header**  
  Navigation layout used across pages  
  Props: none

**8. State Management**

**Global State**

* Managed using **Context API**
* Stores product list, user preferences, and theme data

**Local State**

* Managed using **useState** inside form and search components
* Used for inputs like product details and search queries

**9. User Interface**

**Screenshots**

* **Home Page:** Shows a quick inventory summary (total items, stock levels)
* **Inventory Page:** Lists all products with edit and delete actions
* **Reports Page:** Generates and displays inventory/sales reports
* **Profile Page:** Manages user preferences and app theme