Introduction

Project Title: Store Manager

Team Members:

- **Sathish V** – Team Leader
- **Saurab Kumar S** – Team Member
- **Siva V** – Team Member
- **Venkatachalapathy M** – Team Member

Project Overview

Purpose:
Store Manager is a React.js-based frontend application designed to streamline inventory management, cart handling, and sales tracking for retail businesses. The application allows users to view product catalogs, manage inventory, process sales, and track customer purchases efficiently.

Features:

- Product catalog browsing and filtering
- Inventory management with add/update functionality
- Cart management with real-time updates
- Sales tracking and record management
- Persistent state using local storage
- Responsive user interface powered by Tailwind CSS

Architecture

Component Structure

The application is built using functional React components organized into categories:

- **Inventory components:** `Inventory.jsx`, `Product.jsx`
- **Sales components:** `Sales.jsx`, `SaleRecord.jsx`

- **Cart components:** `Cart.jsx`, `CartItem.jsx`
- **Product components:** `ProductCatalog.jsx`, `ProductList.jsx`, `Product.jsx`
- **Navigation and layout:** `NavBar.jsx`, `Layout.jsx`
- **Utility components:** `AddProduct.jsx`

Components interact via context providers and hooks to maintain state and share data seamlessly across the app.

State Management

The app uses the Context API with three primary contexts:

- **InventoryContext:** Manages product data and inventory states.
- **CartContext:** Handles cart items, quantities, and total cost.
- **SalesContext:** Tracks sales records and transaction history.

A custom hook `useLocalStorage.js` is used to persist state across sessions.

Routing

React Router is implied by the presence of `NotFound.jsx`, which handles undefined routes. Other routing configurations are likely implemented within `App.js` or `Layout.jsx`.

Setup Instructions

Prerequisites

- Node.js (v14 or above recommended)
- npm (comes with Node.js)
- Git (optional, for cloning the repository)

Installation Steps

1. Clone the repository:
2. `git clone [repository-url]`
3. Navigate to the project folder:
4. `cd inventory-management-frontend`
5. Install dependencies:
6. `npm install`

7. Run the development server:
8. `npm start`
9. Open `http://localhost:3000` in your browser to view the application.

Folder Structure

```
inventory-management-frontend/
├── public/
│   ├── favicon.ico
│   ├── index.html
│   ├── manifest.json
│   ├── logo192.png
│   ├── logo512.png
│   └── assets (cartImage.png, inventoryLogo.png)
├── src/
│   ├── components/
│   │   ├── AddProduct.jsx
│   │   ├── Cart/
│   │   │   ├── Cart.jsx
│   │   │   └── CartItem.jsx
│   │   ├── Inventory/
│   │   │   ├── Inventory.jsx
│   │   │   └── Product.jsx
│   │   ├── Product/
│   │   │   ├── ProductCatalog.jsx
│   │   │   ├── Product.jsx
│   │   │   └── ProductList.jsx
│   │   ├── Sales/
│   │   │   ├── Sales.jsx
│   │   │   └── SaleRecord.jsx
│   │   └── NavBar.jsx
│   ├── context/
│   │   ├── CartContext.js
│   │   ├── InventoryContext.js
│   │   └── SalesContext.js
│   ├── hooks/
│   │   └── useLocalStorage.js
│   ├── pages/
│   │   └── NotFound.jsx
│   ├── App.js
│   ├── Layout.jsx
│   ├── index.js
│   ├── index.css
│   └── other configuration files
```

Utilities

- `useLocalStorage.js`: Custom hook to persist state in local storage for better UX.

Running the Application

To run the app locally:

```
npm start
```

This command starts the React development server, and you can access the application via `http://localhost:3000`.

Component Documentation

Key Components

- **NavBar.jsx:** Navigation component allowing users to switch between inventory, products, and sales pages.
- **Inventory.jsx / Product.jsx:** Interfaces for managing stock levels and product details.
- **Cart.jsx / CartItem.jsx:** Cart functionality with add/remove operations and quantity adjustments.
- **Sales.jsx / SaleRecord.jsx:** Displays and manages sales transactions.
- **ProductCatalog.jsx / ProductList.jsx:** Organized display of products with filtering and selection options.
- **AddProduct.jsx:** Form to add new products to the inventory.

Reusable Components

- **CartItem.jsx, Product.jsx, SaleRecord.jsx:** Designed to be reusable across pages with configurable props.

State Management

Global State

The application leverages React’ s Context API to manage global state for inventory, cart, and sales, providing shared state access to multiple components.

Local State

Components like `AddProduct.jsx` and form inputs maintain local state for temporary data handling before submission.

User Interface

The UI is built using Tailwind CSS, ensuring responsiveness and a modern look. Key screens include:

- Product catalog with filters and pagination
- Inventory management with product addition/editing
- Cart summary with interactive adjustments
- Sales tracking dashboard

Screenshots and GIFs can be added here once available.

Styling

CSS Framework

The project uses Tailwind CSS for styling, as evident from `tailwind.config.js`.

Theming

The styling approach is utility-first, allowing rapid UI development with reusable class patterns.

Testing

Testing Strategy

Basic tests are included in `App.test.js`, possibly using Jest and React Testing Library.

Code Coverage

Further test coverage can be implemented for critical user interactions like adding products, handling sales, and managing the cart.

Screenshots or Demo

Screenshots can be captured from the running application once setup. A live demo link can be provided if hosted.

Known Issues

- No formal bug report yet, but edge cases like invalid inputs, empty carts, or failed local storage operations may need testing.
- Routing beyond the defined pages may lead to the NotFound component.

Future Enhancements

- Implement user authentication and role-based access
- Add backend API integration for persistent data storage
- Expand testing coverage for edge cases
- Include animations and transitions for smoother interactions
- Optimize performance for large product catalogs

Let me know if you want this document exported as a