**STORE MANAGER: KEEP TRACK OF INVENTORY**

**Project Documentation**

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

**Team ID: NM2025TMID37173**

**Team Leader: SHAHEERA BHANU S &** [**shaheerabhanu1507@gmail.com**](mailto:shaheerabhanu1507@gmail.com)

**Team Members: JEEVITHA K &** [**jeevikjeevitha8@gmail.com**](mailto:jeevikjeevitha8@gmail.com)

**Team Members: SHOBANA PATRA G &** [**shobanapatra02@gmail.com**](mailto:shobanapatra02@gmail.com)

**Team Members: POOJA N &** [**pooja24nagarajan@gmail.com**](mailto:pooja24nagarajan@gmail.com)

INTRODUCTION:

Inventory management is a crucial part of any retail or product-based business, regardless of its size. While large companies use complex ERP systems to manage their stock and sales, many small shops and businesses still rely on outdated methods such as handwritten notes, Excel sheets, or basic billing tools. These manual approaches are time-consuming, error-prone, and lack real-time updates or meaningful data tracking. Problems such as inaccurate stock levels, missed restocking opportunities, and loss of sales history are common in such systems. To address these challenges, there is a need for a simple, effective, and easy-to-use digital tool that helps small business owners manage their inventory and track sales without the overhead of backend servers or expensive software. The project titled **"Store Manager: Keep Track of Inventory"** is designed as a lightweight, frontend-based inventory management application that fulfills this exact need. Built using **ReactJS** and relying on **Chrome Local Storage** for data handling, the system provides an efficient and user-friendly platform to manage products, monitor stock, and record sales with minimal technical setup.

This project is divided into five main tabs: **Home, Cart, Sales, Inventory**,and**AddProduct**. The Home tab displays all available products with options to "Add to Cart" or "Remove from Cart" and includes a search bar for quick filtering. The Cart tab collects selected products and allows the user to either **Checkout**, which updates the stock and records the sale, or **Clear** the cart without processing. Completed transactions are recorded and displayed in the **Sales** tab, showing key details like product names, quantities, total sale value, and the date and time of the transaction. The **Inventory** tab provides features to update stock levels and set alert thresholds for low stock items. Products that fall below the set alert value are highlighted in red, and users can filter the view to show only depleted items. The **Add Product** tab allows for adding new products by entering details like name, image URL, price, stock quantity, and tags. All of this data is stored in the browser’s **Local Storage**, which means the application works without any backend database or server setup and can retain data across browser sessions.

The use of ReactJS makes the interface dynamic, modular, and responsive, allowing users to interact with the system in real time. Each component of the application is designed with clarity and simplicity, ensuring even users with little technical knowledge can operate it effectively. Moreover, the application can function offline since all operations are handled on the client side, making it ideal for small businesses that may not always have internet access. While the current version uses Local Storage for simplicity, the structure of the project allows for easy future upgrades to integrate with backend technologies like **MongoDB** or **Firebase** if needed. The goal of this project is not only to demonstrate technical skills in frontend development using ReactJS but also to provide a **practical, working solution** that addresses a real-world problem. By streamlining sales tracking and inventory control into a single, browser-based tool, the system enhances efficiency, reduces manual errors, and gives small business owners better visibility into their operations. This project showcases how modern frontend technologies can be used to build functional applications that make a tangible difference in day-to-day business tasks.

SOURCE CODE:

**Index.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8" />**

**<link rel="icon" href="%PUBLIC\_URL%/the pink store logo.png" />**

**<meta name="viewport" content="width=device-width, initial-scale=1" />**

**<meta name="theme-color" content="#000000" />**

**<meta name="description" content="Web site created using create-react-app" />**

**<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />**

**<!--**

**manifest.json provides metadata used when your web app is installed on a**

**user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/**

**-->**

**<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />**

**<!--**

**Notice the use of %PUBLIC\_URL% in the tags above.**

**It will be replaced with the URL of the `public` folder during the build.**

**Only files inside the `public` folder can be referenced from the HTML.**

**Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will**

**work correctly both with client-side routing and a non-root public URL.**

**Learn how to configure a non-root public URL by running `npm run build`.**

**-->**

**<title>THE PINK STORE</title>**

**</head>**

**<body style="background-color: rgb(243 244 246 / var(--tw-bg-opacity, 1));">**

**<noscript>You need to enable JavaScript to run this app.</noscript>**

**<div id="root"></div>**

**<!--**

**This HTML file is a template.**

**If you open it directly in the browser, you will see an empty page.**

**You can add webfonts, meta tags, or analytics to this file.**

**The build step will place the bundled scripts into the <body> tag.**

**To begin the development, run `npm start` or `yarn start`.**

**To create a production bundle, use `npm run build` or `yarn build`.**

**-->**

**</body>**

**</html>**

**AddProduct.jsx**

**import { useInventoryDispatch } from "../context/InventoryContext";**

**export default function AddProduct() {**

**const dispatchToInventory = useInventoryDispatch()**

**const onAddProduct = (e) => {**

**e.preventDefault()**

**const newProduct = {**

**productName: e.target.productName.value,**

**imageUrl: e.target.imageUrl.value,**

**price: parseFloat(e.target.price.value),**

**tags: e.target.tags.value.split(",").map((tag) => tag.trim()),**

**stock: e.target.stock.value,**

**}**

**dispatchToInventory({**

**type: 'NEW\_PRODUCT',**

**...newProduct,**

**})**

**e.target.reset()**

**alert("PRODUCT ADDED SUCCESSFULLY!")**

**};**

**return (**

**<div className="max-w-md mx-auto p-6 bg-pink-200 shadow-lg rounded-lg mt-6 mb-4">**

**<h1 className="font-bold text-4xl text-center mb-6 text-black-700">**

**Add new product**

**</h1>**

**<form onSubmit={onAddProduct} className="space-y-4">**

**<div>**

**<label**

**htmlFor="productName"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Name**

**</label>**

**<input**

**id="productName"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter product name"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="imageUrl"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Image URL**

**</label>**

**<input**

**id="imageUrl"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter image URL"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="price"**

**className="block text-sm font-medium text-black-700"**

**>**

**Price**

**</label>**

**<input**

**id="price"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter price"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="stock"**

**className="block text-sm font-medium text-black-700"**

**>**

**Stock**

**</label>**

**<input**

**id="stock"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter stock"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="tags"**

**className="block text-sm font-medium text-black-700"**

**>**

**Tags (comma-separated)**

**</label>**

**<input**

**id="tags"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter tags, separated by commas"**

**required**

**/>**

**</div>**

**<div className="text-center">**

**<button**

**type="submit"**

**className="w-full bg-pink-500 text-white font-bold py-2 px-4 rounded hover:bg-yellow-600 transition"**

**>**

**ADD PRODUCT!**

**</button>**

**</div>**

**</form>**

**</div>**

**);**

**}**

**NavBar.jsx**

**import { useInventoryDispatch } from "../context/InventoryContext";**

**export default function AddProduct() {**

**const dispatchToInventory = useInventoryDispatch()**

**const onAddProduct = (e) => {**

**e.preventDefault()**

**const newProduct = {**

**productName: e.target.productName.value,**

**imageUrl: e.target.imageUrl.value,**

**price: parseFloat(e.target.price.value),**

**tags: e.target.tags.value.split(",").map((tag) => tag.trim()),**

**stock: e.target.stock.value,**

**}**

**dispatchToInventory({**

**type: 'NEW\_PRODUCT',**

**...newProduct,**

**})**

**e.target.reset()**

**alert("PRODUCT ADDED SUCCESSFULLY!")**

**};**

**return (**

**<div className="max-w-md mx-auto p-6 bg-pink-200 shadow-lg rounded-lg mt-6 mb-4">**

**<h1 className="font-bold text-4xl text-center mb-6 text-black-700">**

**Add new product**

**</h1>**

**<form onSubmit={onAddProduct} className="space-y-4">**

**<div>**

**<label**

**htmlFor="productName"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Name**

**</label>**

**<input**

**id="productName"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter product name"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="imageUrl"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Image URL**

**</label>**

**<input**

**id="imageUrl"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter image URL"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="price"**

**className="block text-sm font-medium text-black-700"**

**>**

**Price**

**</label>**

**<input**

**id="price"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter price"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="stock"**

**className="block text-sm font-medium text-black-700"**

**>**

**Stock**

**</label>**

**<input**

**id="stock"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter stock"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="tags"**

**className="block text-sm font-medium text-black-700"**

**>**

**Tags (comma-separated)**

**</label>**

**<input**

**id="tags"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter tags, separated by commas"**

**required**

**/>**

**</div>**

**<div className="text-center">**

**<button**

**type="submit"**

**className="w-full bg-pink-500 text-white font-bold py-2 px-4 rounded hover:bg-yellow-600 transition"**

**>**

**ADD PRODUCT!**

**</button>**

**</div>**

**</form>**

**</div>**

**);**

**}**

**Cart.jsx**

**import { useInventoryDispatch } from "../context/InventoryContext";**

**export default function AddProduct() {**

**const dispatchToInventory = useInventoryDispatch()**

**const onAddProduct = (e) => {**

**e.preventDefault()**

**const newProduct = {**

**productName: e.target.productName.value,**

**imageUrl: e.target.imageUrl.value,**

**price: parseFloat(e.target.price.value),**

**tags: e.target.tags.value.split(",").map((tag) => tag.trim()),**

**stock: e.target.stock.value,**

**}**

**dispatchToInventory({**

**type: 'NEW\_PRODUCT',**

**...newProduct,**

**})**

**e.target.reset()**

**alert("PRODUCT ADDED SUCCESSFULLY!")**

**};**

**return (**

**<div className="max-w-md mx-auto p-6 bg-pink-200 shadow-lg rounded-lg mt-6 mb-4">**

**<h1 className="font-bold text-4xl text-center mb-6 text-black-700">**

**Add new product**

**</h1>**

**<form onSubmit={onAddProduct} className="space-y-4">**

**<div>**

**<label**

**htmlFor="productName"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Name**

**</label>**

**<input**

**id="productName"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter product name"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="imageUrl"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Image URL**

**</label>**

**<input**

**id="imageUrl"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter image URL"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="price"**

**className="block text-sm font-medium text-black-700"**

**>**

**Price**

**</label>**

**<input**

**id="price"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter price"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="stock"**

**className="block text-sm font-medium text-black-700"**

**>**

**Stock**

**</label>**

**<input**

**id="stock"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter stock"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="tags"**

**className="block text-sm font-medium text-black-700"**

**>**

**Tags (comma-separated)**

**</label>**

**<input**

**id="tags"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter tags, separated by commas"**

**required**

**/>**

**</div>**

**<div className="text-center">**

**<button**

**type="submit"**

**className="w-full bg-pink-500 text-white font-bold py-2 px-4 rounded hover:bg-yellow-600 transition"**

**>**

**ADD PRODUCT!**

**</button>**

**</div>**

**</form>**

**</div>**

**);**

**}**

**CartItem.jsx**

**import { useInventoryDispatch } from "../context/InventoryContext";**

**export default function AddProduct() {**

**const dispatchToInventory = useInventoryDispatch()**

**const onAddProduct = (e) => {**

**e.preventDefault()**

**const newProduct = {**

**productName: e.target.productName.value,**

**imageUrl: e.target.imageUrl.value,**

**price: parseFloat(e.target.price.value),**

**tags: e.target.tags.value.split(",").map((tag) => tag.trim()),**

**stock: e.target.stock.value,**

**}**

**dispatchToInventory({**

**type: 'NEW\_PRODUCT',**

**...newProduct,**

**})**

**e.target.reset()**

**alert("PRODUCT ADDED SUCCESSFULLY!")**

**};**

**return (**

**<div className="max-w-md mx-auto p-6 bg-pink-200 shadow-lg rounded-lg mt-6 mb-4">**

**<h1 className="font-bold text-4xl text-center mb-6 text-black-700">**

**Add new product**

**</h1>**

**<form onSubmit={onAddProduct} className="space-y-4">**

**<div>**

**<label**

**htmlFor="productName"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Name**

**</label>**

**<input**

**id="productName"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter product name"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="imageUrl"**

**className="block text-sm font-medium text-black-700"**

**>**

**Product Image URL**

**</label>**

**<input**

**id="imageUrl"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter image URL"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="price"**

**className="block text-sm font-medium text-black-700"**

**>**

**Price**

**</label>**

**<input**

**id="price"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter price"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="stock"**

**className="block text-sm font-medium text-black-700"**

**>**

**Stock**

**</label>**

**<input**

**id="stock"**

**type="number"**

**step="0.01"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter stock"**

**required**

**/>**

**</div>**

**<div>**

**<label**

**htmlFor="tags"**

**className="block text-sm font-medium text-black-700"**

**>**

**Tags (comma-separated)**

**</label>**

**<input**

**id="tags"**

**type="text"**

**className="w-full p-2 border rounded focus:ring focus:ring-pink-300 focus:outline-none"**

**placeholder="Enter tags, separated by commas"**

**required**

**/>**

**</div>**

**<div className="text-center">**

**<button**

**type="submit"**

**className="w-full bg-pink-500 text-white font-bold py-2 px-4 rounded hover:bg-yellow-600 transition"**

**>**

**ADD PRODUCT!**

**</button>**

**</div>**

**</form>**

**</div>**

**);**

**}**

**Inventory.jsx**

**import { useState } from "react";**

**import { useInventory } from "../../context/InventoryContext";**

**import Product from "./Product";**

**const Inventory = () => {**

**const inventory = useInventory()**

**const [alertValue, setAlertValue] = useState(10)**

**const [searchQuery, setSearchQuery] = useState("")**

**const [showOnlyDepleted, setShowOnlyDepleted] = useState(false)**

**const lowerCaseSearchQuery = searchQuery.toLowerCase()**

**const filteredInventory = inventory.filter((product) => {**

**const matchesSearchQuery = product.productName.toLowerCase().includes(lowerCaseSearchQuery);**

**const isDepleted = product.stock < alertValue;**

**return showOnlyDepleted ? (matchesSearchQuery && isDepleted) : matchesSearchQuery;**

**})**

**return (**

**<div className="m-2 mb-4 flex flex-col items-center bg-pink-200">**

**<h1 className="font-bold text-2xl mx-2">INVENTORY</h1>**

**<div className="m-2 mb-3 w-full max-w-md flex items-center gap-2">**

**<input**

**type="text"**

**placeholder="SEARCH?"**

**value={searchQuery}**

**onChange={(e) => {**

**setSearchQuery(e.target.value)**

**}}**

**className="border hover:border-pink-400 p-2 rounded flex-grow"**

**/>**

**<div className="flex items-center">**

**<label htmlFor="alert-value" className="mr-2 ml-8">ALERT VALUE</label>**

**<input**

**id="alert-value"**

**className="border p-2 hover:border-pink-400 rounded w-20"**

**value={alertValue}**

**type="number"**

**onChange={(e) => setAlertValue(e.target.value)}**

**/>**

**<label className="ml-8" htmlFor="show-only-depleted">SHOW ONLY DEPLETED</label>**

**<input**

**className="ml-3 w-[35px] h-[35px]"**

**id="show-only-depleted" type="checkbox" checked={showOnlyDepleted} onChange={() => {**

**setShowOnlyDepleted(!showOnlyDepleted)**

**}}**

**/>**

**</div>**

**</div>**

**{filteredInventory.length > 0 ? (**

**<div className="flex flex-wrap justify-center gap-4">**

**{filteredInventory.map((product) => (**

**<Product key={product.productName} product={product} alertValue={alertValue} />**

**))}**

**</div>**

**) : (**

**<p className="m-2 text-pink-500">NO INVENTORY FOUND</p>**

**)}**

**</div>**

**);**

**};**

**export default Inventory;**

**Product.jsx**

**import { useState } from "react";**

**import { useInventoryDispatch } from "../../context/InventoryContext";**

**const Product = ({ product, alertValue }) => {**

**const [addStock, setAddStock] = useState(0)**

**const inventoryDispatch = useInventoryDispatch()**

**return (**

**<div**

**className={`border-pink-400 px-3 py-2 sharp text-center flex flex-col items-center ${product.stock < alertValue ? "border-pink-800 bg-white-100 border-0" : "border-pink-400"**

**}`}**

**>**

**<h1 className="font text-xl">{product.productName}</h1>**

**<div className="w-[250px] h-[250px] overflow-hidden border border-pink-300 rounded mt-2">**

**<img**

**src={product.imageUrl}**

**alt={product.productName}**

**className="w-full h-full object-cover"**

**/>**

**</div>**

**<p className="text-lg mt-2">PRICE: ₹ {product.price.toFixed(2)}</p>**

**<div>**

**<p>Stock Available: {product.stock}</p>**

**</div>**

**<div className="mt-2">**

**Add Stock:{" "}**

**<input**

**onChange={(e) => {**

**setAddStock(e.target.value);**

**}}**

**className="border border-pink-400 rounded p-1"**

**value={addStock}**

**type="number"**

**name="new-stock-qty"**

**id="new-stock-qty"**

**/>**

**</div>**

**<div>**

**<button**

**onClick={() => {**

**setAddStock(0);**

**inventoryDispatch({**

**type: "STOCK\_ADDED",**

**productName: product.productName,**

**stock: addStock,**

**});**

**}}**

**className="bg-pink-500 hover:bg-red-600 rounded p-1 text-white mt-2"**

**>**

**Update Stock**

**</button>**

**</div>**

**</div>**

**);**

**};**

**export default Product;**

**Product.jsx**

**import { useCartDispatch } from "../../context/CartContext";**

**import { useCart } from "../../context/CartContext";**

**export default function Product({ product }) {**

**const cartItems = useCart();**

**const dispatchToCart = useCartDispatch();**

**const onCartToggle = () => {**

**if (cartItems.some((item) => item.productName === product.productName)) {**

**dispatchToCart({**

**type: "removed",**

**...product,**

**});**

**} else {**

**dispatchToCart({**

**type: "added",**

**...product,**

**});**

**}**

**};**

**return (**

**<div className="border border-red-800 p-4 sharp text-center flex flex-col items-center">**

**<h1 className="font text-xl">{product.productName}</h1>**

**<div className="w-[250px] h-[250px] overflow-hidden border border-red-800 rounded mt-2">**

**<img**

**src={product.imageUrl}**

**alt={product.productName}**

**className="w-full h-full object-cover"**

**/>**

**</div>**

**<p className="text-lg mt-2">PRICE: ₹ {product.price.toFixed(2)}</p>**

**<button**

**onClick={onCartToggle}**

**className={`mt-1 sharp p-2 transition ${**

**cartItems.some((item) => item.productName === product.productName)**

**? "bg-pink-500 hover:bg-red-600 text-white"**

**: "bg-pink-500 hover:bg-red-600 text-white"**

**}`}**

**>**

**{cartItems.some((item) => item.productName === product.productName)**

**? "REMOVE ME?"**

**: "ADD ME?"}**

**</button>**

**</div>**

**);**

**}**

**ProductCatalog.jsx**

**import { useState } from "react";**

**import ProductList from "./ProductList";**

**import { useInventory } from "../../context/InventoryContext";**

**export default function ProductCatalog() {**

**const inventory = useInventory();**

**const [searchQuery, setSearchQuery] = useState("")**

**const filteredProducts = inventory.filter(**

**(item) =>**

**item.stock > 0 &&**

**item.productName.toLowerCase().includes(searchQuery.toLowerCase())**

**);**

**return (**

**<div className="flex flex-col items-center p-1 mb-2 bg-pink-200">**

**<h1 className="mx-2 mt-1 font-bold text-2xl">PRODUCT LIST</h1>**

**<div className="m-2 w-full max-w-md">**

**<input**

**type="text"**

**placeholder="LOOKING FOR SOMETHING..."**

**value={searchQuery}**

**onChange={(e) => setSearchQuery(e.target.value)}**

**className="border-pink-200 hover:border-pink-400 p-2 rounded w-full"**

**/>**

**</div>**

**{filteredProducts.length > 0 ? (**

**<ProductList products={filteredProducts} />**

**) : (**

**<p className="m-2 text-pink-500">OOPS,THERE'S NO SUCH PRODUCT</p>**

**)}**

**</div>**

**);**

**}**

**ProductList.jsx**

**import Product from "./Product";**

**export default function ProductList({ products }) {**

**return (**

**<div className="flex flex-wrap justify-center gap-4 mt-4 bg-pink-200">**

**{products.map((product) => {**

**return <Product key={product.productName} product={product} />;**

**})}**

**</div>**

**);**

**}**

**SalesRecord.jsx**

**const SaleRecord = ({ sale, saleId }) => {**

**return (**

**<div className="border border-pink-300 shadow-sm rounded-lg p-4 bg-pink-200">**

**<div className="flex justify-between items-center mb-2">**

**<h2 className="text-lg font-semibold">Sale #{saleId + 1}</h2>**

**<span className="text-sm text-black-500">**

**{new Date(sale.datetime).toLocaleString()}**

**</span>**

**</div>**

**<p className="mb-2">**

**<strong>Total Sale Value:</strong> ₹{sale.saleValue.toFixed(2)}**

**</p>**

**<p className="mb-2">**

**<strong>Cart Details:</strong>**

**</p>**

**<ul className="list-disc list-inside pl-4 text-gray-700">**

**{sale.products.map((product) => (**

**<li key={product.productName}>**

**{product.productName} ({product.quantity}) - ₹**

**{(product.price \* product.quantity).toFixed(2)}**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

**};**

**export default SaleRecord;**

**Sales.jsx**

**import { useSales } from "../../context/SalesContext";**

**import SaleRecord from "./SaleRecord";**

**const Sales = () => {**

**const sales = useSales();**

**return (**

**<div className="m-4">**

**<h1 className="text-2xl font-bold mb-4">SALES RECORD</h1>**

**{sales.length > 0 ? (**

**<div className="grid gap-4">**

**{sales.map((sale, index) => (**

**<SaleRecord key={index} sale={sale} saleId={index} />**

**))}**

**</div>**

**) : (**

**<p className="text-blacK-600">NO SCALES RECORDED</p>**

**)}**

**</div>**

**);**

**};**

**export default Sales;**

**App.css**

**.App {**

**text-align: center;**

**}**

**.App-logo {**

**height: 40vmin;**

**pointer-events: none;**

**}**

**@media (prefers-reduced-motion: no-preference) {**

**.App-logo {**

**animation: App-logo-spin infinite 20s linear;**

**}**

**}**

**.App-header {**

**background-color: #282c34;**

**min-height: 100vh;**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**justify-content: center;**

**font-size: calc(10px + 2vmin);**

**color: white;**

**}**

**.App-link {**

**color: #61dafb;**

**font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;**

**}**

**@keyframes App-logo-spin {**

**from {**

**transform: rotate(0deg);**

**}**

**to {**

**transform: rotate(360deg);**

**}**

**}**

**App.js**

**import './App.css';**

**import {**

**createBrowserRouter,**

**RouterProvider,**

**} from "react-router-dom";**

**import AddProduct from './components/AddProduct';**

**import ProductCatalog from './components/Product/ProductCatalog';**

**import Layout from './Layout';**

**import Cart from './components/Cart/Cart';**

**import NotFound from './pages/NotFound';**

**import { CartProvider } from './context/CartContext';**

**import Inventory from './components/Inventory/Inventory';**

**import { InventoryProvider } from './context/InventoryContext';**

**import { SalesProvider } from './context/SalesContext';**

**import Sales from './components/Sales/Sales';**

**const router = createBrowserRouter([**

**{**

**element: (**

**<Layout />**

**),**

**children: [**

**{**

**path: "/",**

**element: <ProductCatalog />,**

**},**

**{**

**path: "/add-product",**

**element: <AddProduct />,**

**},**

**{**

**path: "/inventory",**

**element: <Inventory />,**

**},**

**{**

**path: "/cart",**

**element: <Cart />,**

**},**

**{**

**path: "/sales",**

**element: <Sales />,**

**},**

**{**

**path: "\*",**

**element: <NotFound />,**

**},**

**]**

**}**

**])**

**const App = () => {**

**return (**

**<InventoryProvider>**

**<CartProvider>**

**<SalesProvider>**

**<RouterProvider router={router} />**

**</SalesProvider>**

**</CartProvider>**

**</InventoryProvider>**

**);**

**}**

**export default App**

**App.test.js**

**import { render, screen } from '@testing-library/react';**

**import App from './App';**

**test('renders learn react link', () => {**

**render(<App />);**

**const linkElement = screen.getByText(/learn react/i);**

**expect(linkElement).toBeInTheDocument();**

**});**

**Index.css**

**@tailwind base;**

**@tailwind components;**

**@tailwind utilities;**

**body {**

**margin: 0;**

**font-family:cursive ;**

**-webkit-font-smoothing: antialiased;**

**-moz-osx-font-smoothing: grayscale;**

**}**

**code {**

**font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',**

**;**

**}**

**Index.js**

**import React from 'react';**

**import ReactDOM from 'react-dom/client';**

**import './index.css';**

**import App from './App';**

**import reportWebVitals from './reportWebVitals';**

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<React.StrictMode>**

**<App />**

**</React.StrictMode>**

**);**

**// If you want to start measuring performance in your app, pass a function**

**// to log results (for example: reportWebVitals(console.log))**

**// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals**

**reportWebVitals();**

**Layout.jsx**

**import React from 'react';**

**import ReactDOM from 'react-dom/client';**

**import './index.css';**

**import App from './App';**

**import reportWebVitals from './reportWebVitals';**

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<React.StrictMode>**

**<App />**

**</React.StrictMode>**

**);**

**// If you want to start measuring performance in your app, pass a function**

**// to log results (for example: reportWebVitals(console.log))**

**// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals**

**reportWebVitals();**

**reportWebVitals.js**

**const reportWebVitals = onPerfEntry => {**

**if (onPerfEntry && onPerfEntry instanceof Function) {**

**import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {**

**getCLS(onPerfEntry);**

**getFID(onPerfEntry);**

**getFCP(onPerfEntry);**

**getLCP(onPerfEntry);**

**getTTFB(onPerfEntry);**

**});**

**}**

**};**

**export default reportWebVitals;**

**setupTests.js**

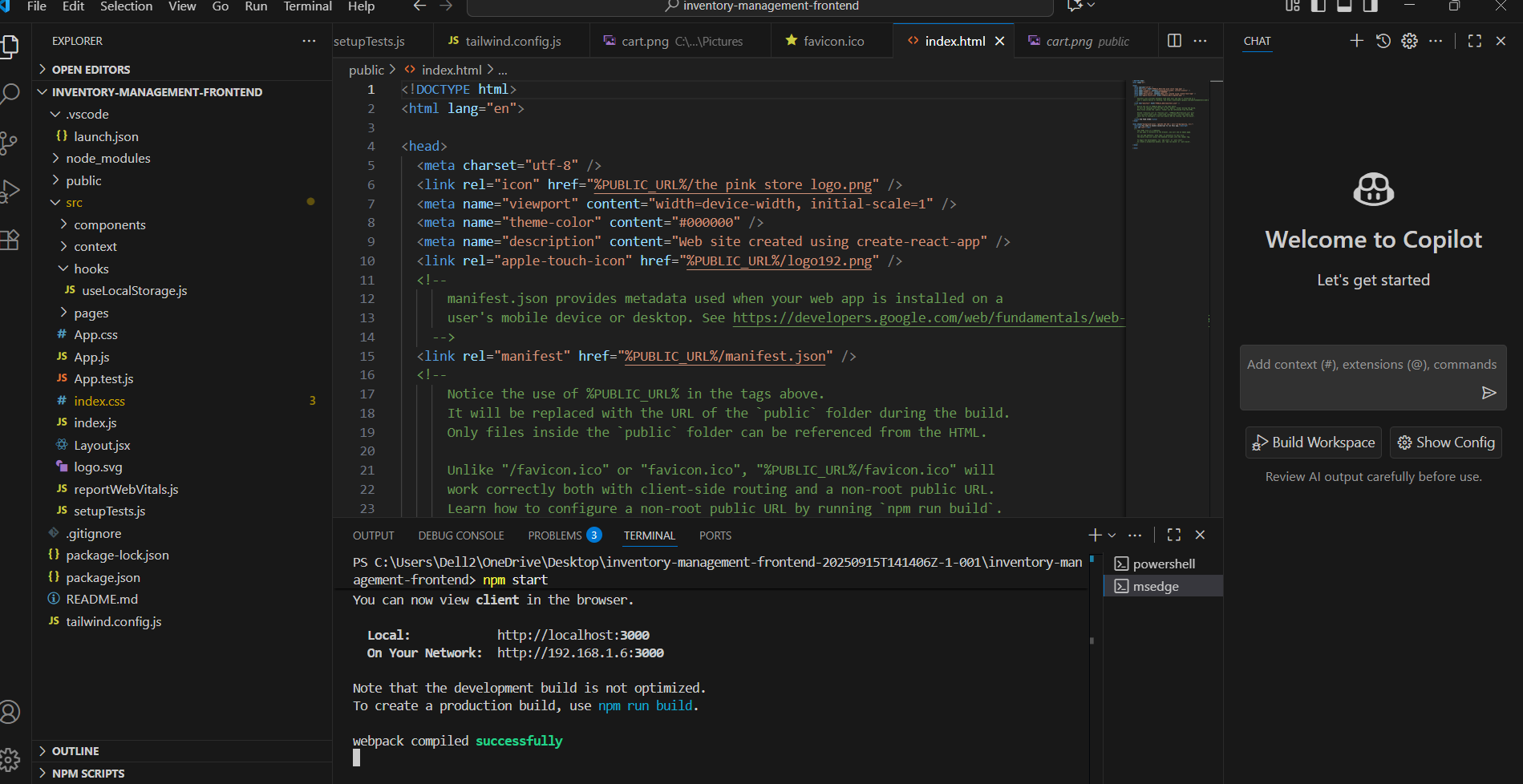
**// jest-dom adds custom jest matchers for asserting on DOM nodes.**

**// allows you to do things like:**

**// expect(element).toHaveTextContent(/react/i)**

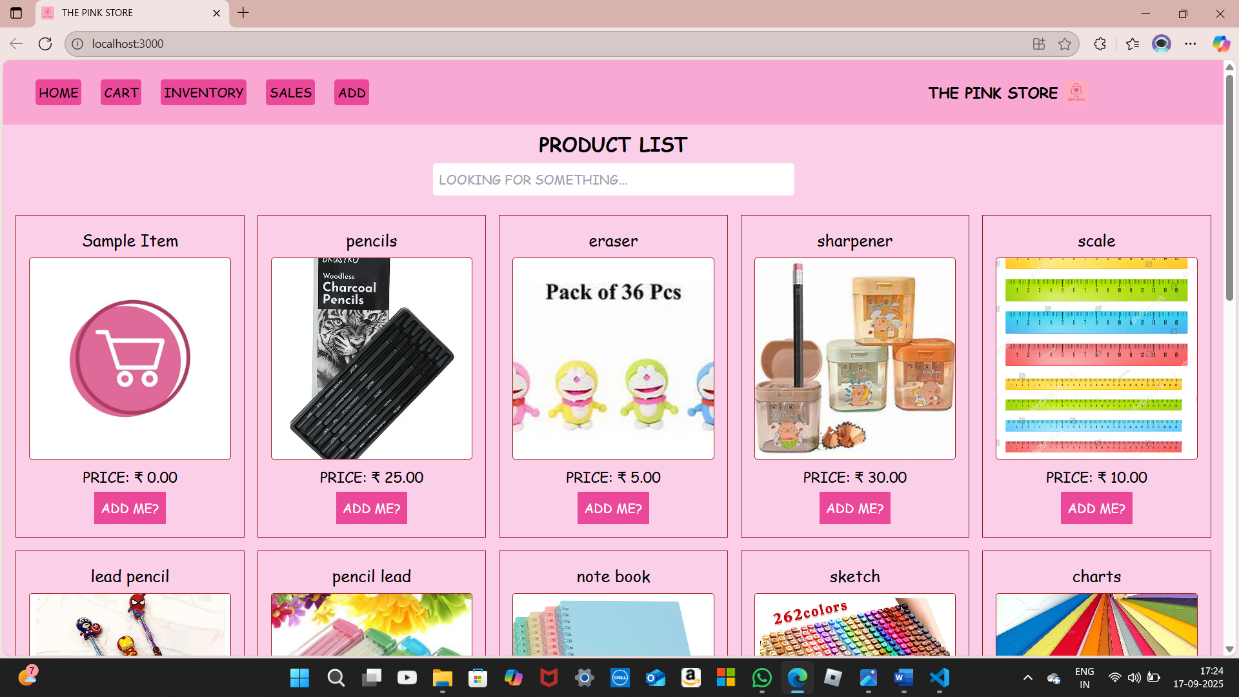
**// learn more: https://github.com/testing-library/jest-dom**

**import '@testing-library/jest-dom';**

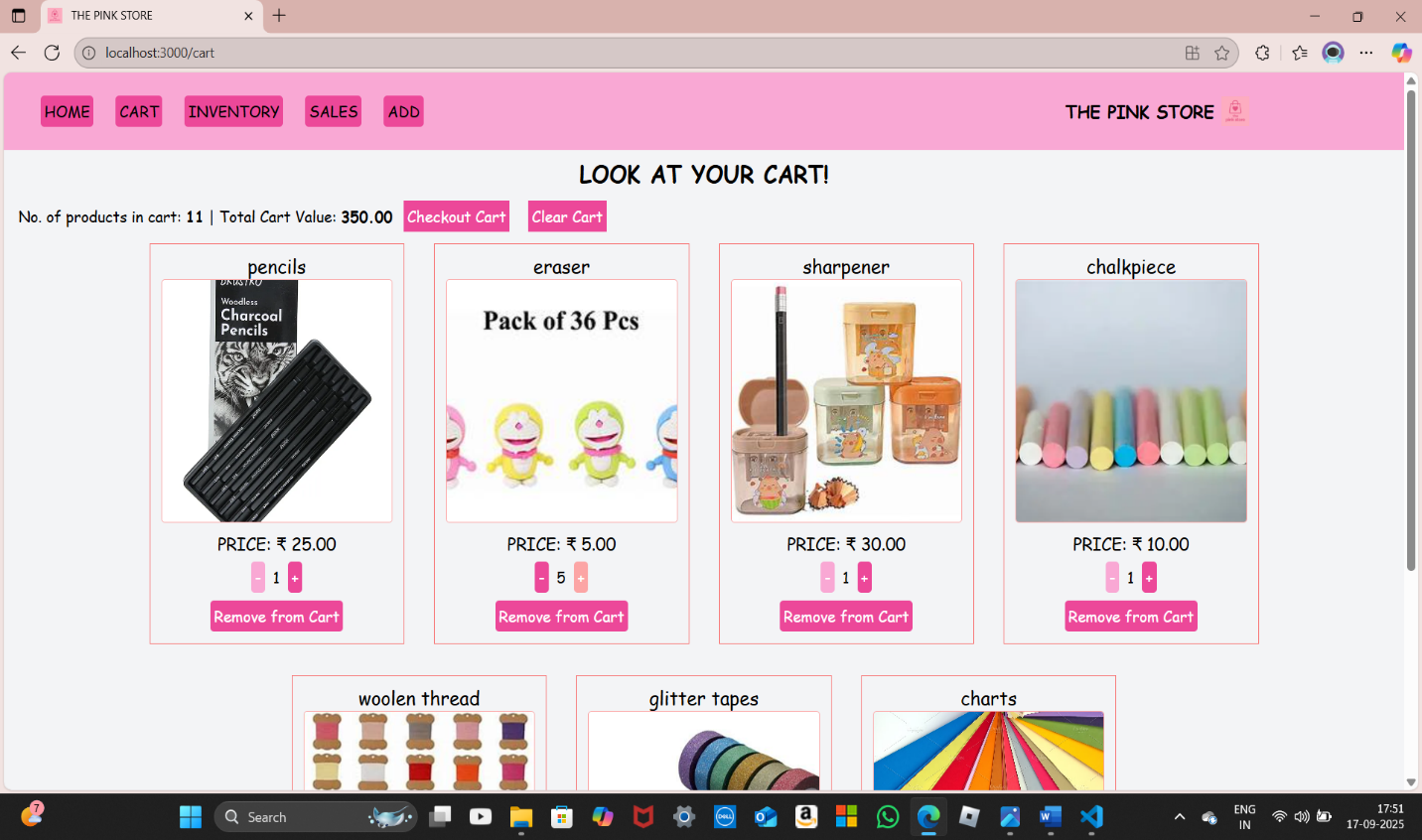
****

OUTPUT SCREENSHOT:

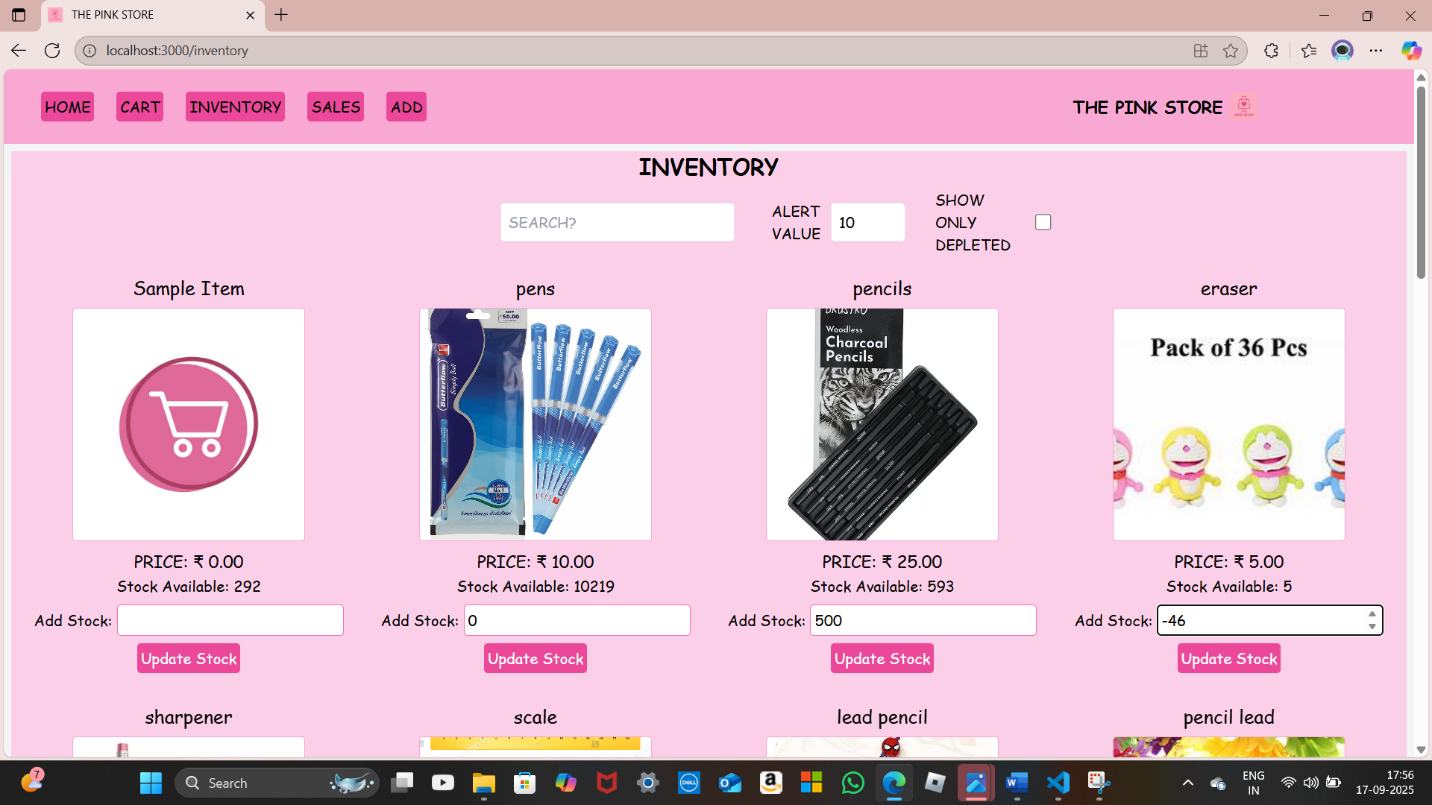
HOME PAGE:

****

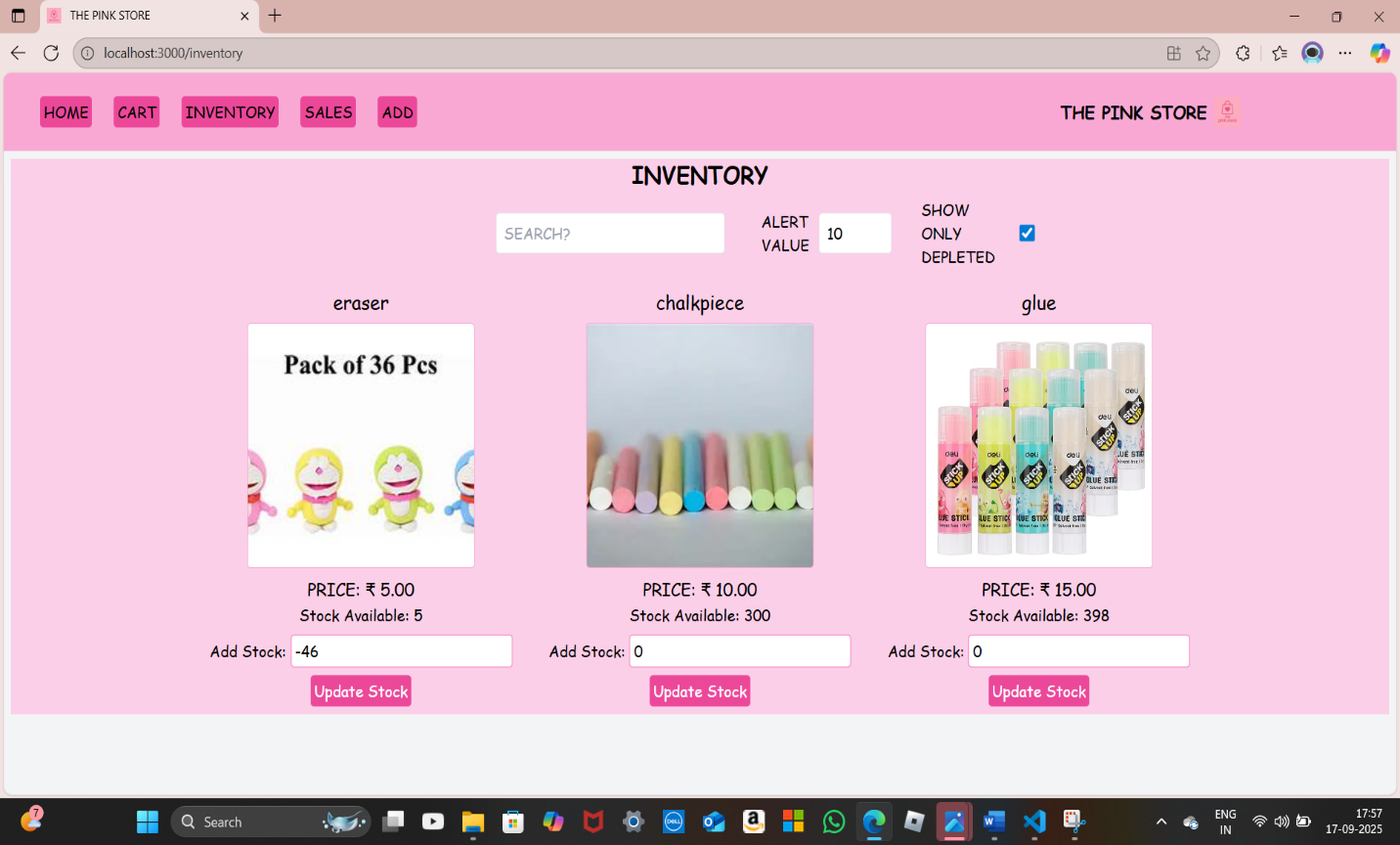
CART PAGE:



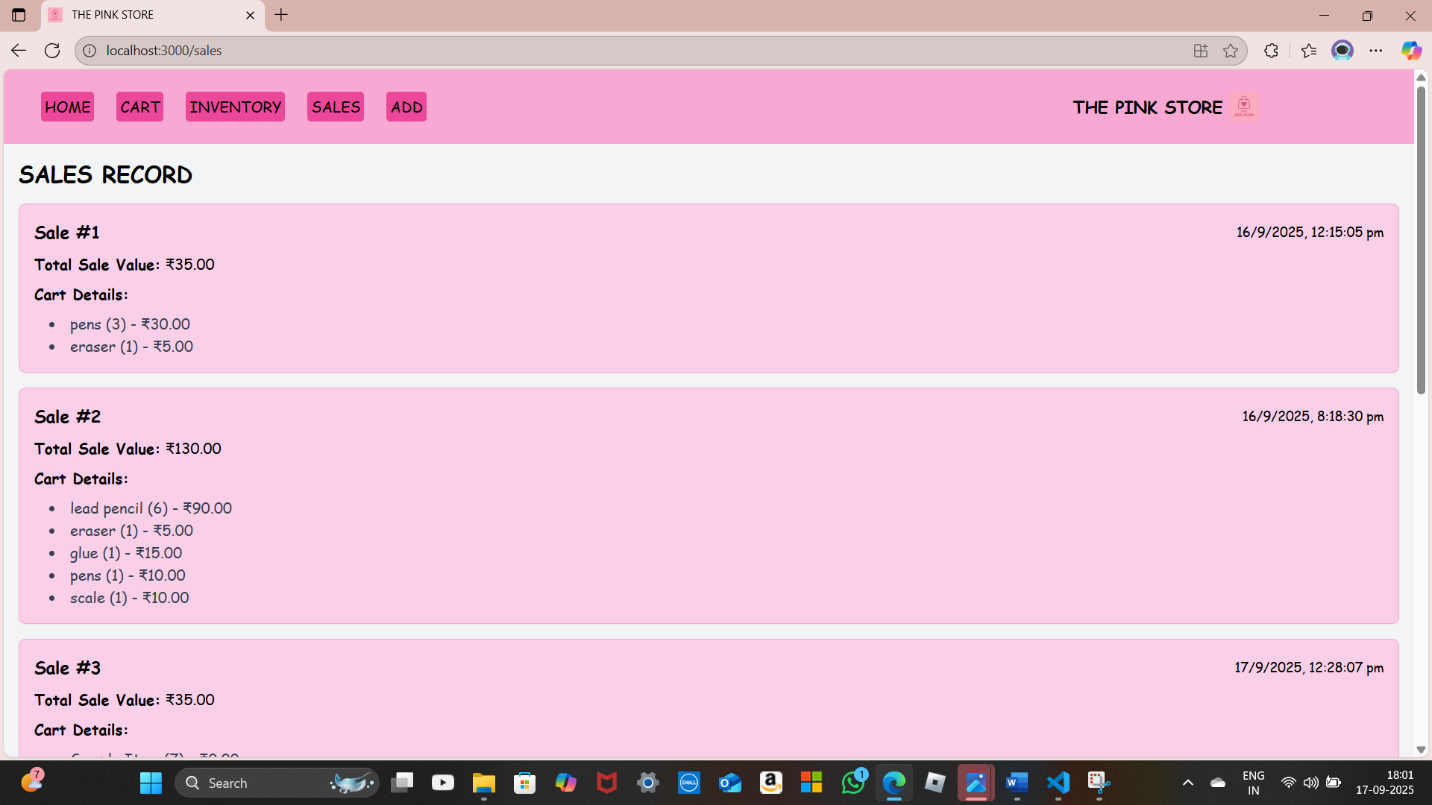
**INVENTORY PAGE:**

****

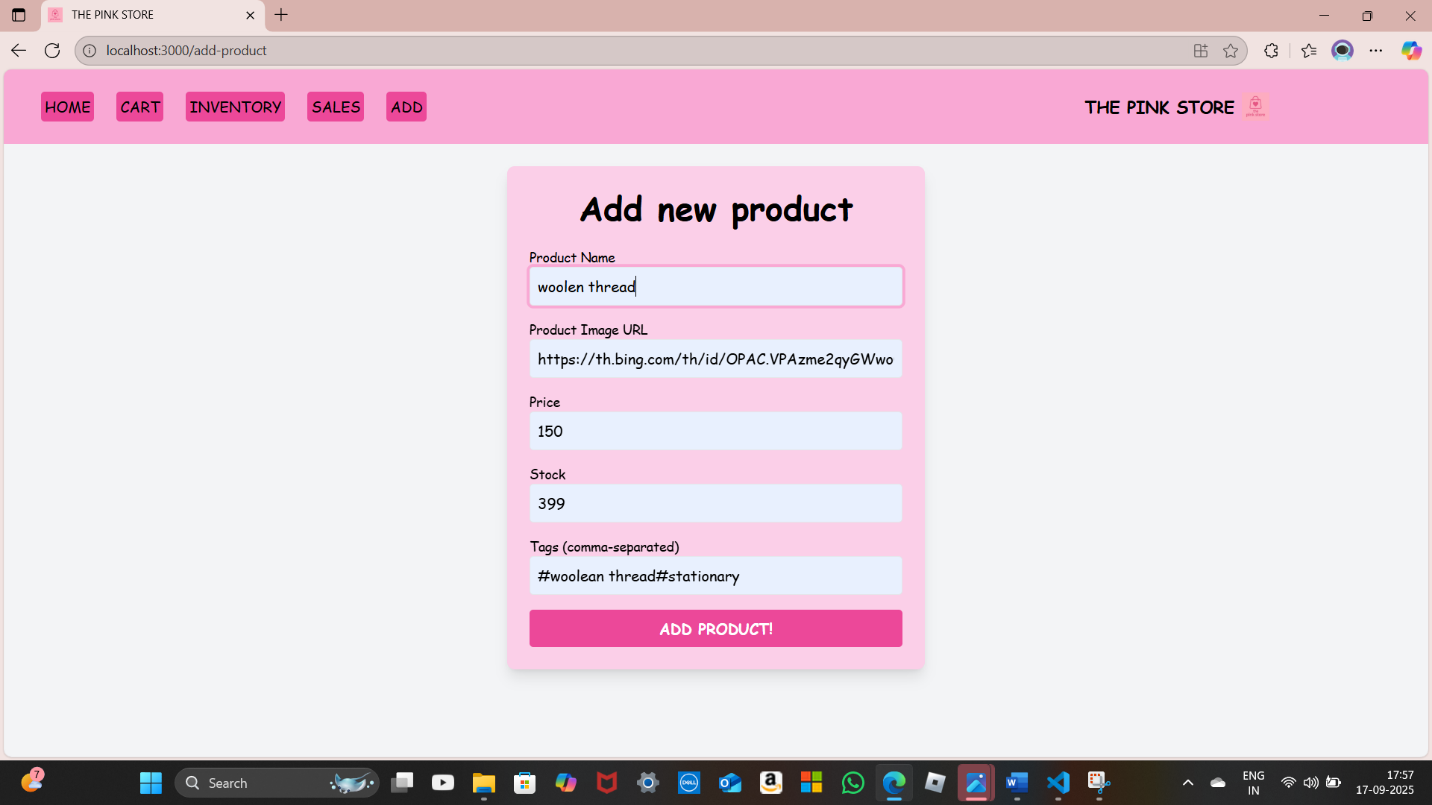
**DEPLETED PAGE:**

****

**SALES RECORD PAGE:**

****

**ADD NEW PRODUCT PAGE:**

****

DEMO LINK:

<https://drive.google.com/file/d/1MvWiFbz1667uuKHgSaWaBAS5AWme3Xez/view?usp=drivesdk>

GITHUB LINK:

<https://github.com/Shaheera15/SHAHEERA-BHANU-S.git>