**PROJECT DOCUMENTATION**

**Store Manager – Keep Track of Inventory.**

**1. Introduction**

**• Project Title:Store Manager – Keep Track of Inventory.**

**• Team ID: NM2025TMID31172**

**• Team Leader:DHARANI P-pandiselvamdharani@gmail.com**

**• Team Members:**

**o DHARSHINI C-m60982235@gmail.com**

**o GOMATHI A– gomathiias2007@gmail.com**

**o HINDUJA M – hindujakuberan@gmail.com**

**o JAYASRI B–dharshusri12@gmail.com**

**2. Project Overview**

* **Purpose:**

**The Store Manager project is designed to keep track of inventory in an efficient way.It allows adding, editing, and removing products with essential details. Users can search, filter, and sort items for quick access.   
 Low stock alerts help prevent shortages and maintain smooth operations. Overall, it simplifies inventory management and improves store efficiency.**

* **Goals:** 
  + 1. **To provide a simple and efficient system for managing store products and inventory.**
    2. **To implement cart and sales management for easy tracking of transactions.**
    3. **To enable quick addition of new products with essential details.**
    4. **To ensure smooth stock management with alerts for low or depleted items.**
    5. **To improve the overall accuracy and efficiency of store operations.**
* **Key Features:**
  + 1. **Product Catalog** – Displays available products with price, image, and option to add to cart.
    2. **Cart Management** – Allows users to add items to a cart and view selected products.
    3. **Inventory Tracking** – Shows stock availability and gives low-stock alerts with highlighted warning.
    4. **Sales Records** – Keeps track of sales history for future analysis.
    5. **Add New Product** – Admin can easily add products with name, image, price, and stock.
    6. **Search & Filter** – Quick product search in catalog and inventory.
    7. **Stock Update** – Option to increase stock quantities directly from the inventory page.
    8. **Responsive UI** – Clean layout that works well across different devices.

**3. Architecture**

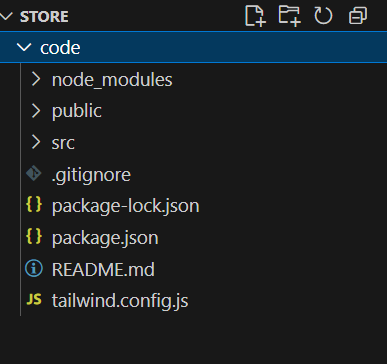
* **Component Structure**
  + **App.js** — Root component, sets up routes and layout.
  + **Layout.jsx** — Defines common layout with Navbar and page content.
  + **NavBar.jsx** — Provides navigation across pages (Home, Cart, Inventory, Sales, Add Product).
  + **ProductCatalog.jsx** — Main page displaying all products.
  + **ProductList.jsx** — Renders list/grid of products.
  + **Product.jsx** — Shows a single product card with image, price, and Add to Cart button.
  + **Cart.jsx** — Displays all items currently in the cart.
  + **CartItem.jsx** — Renders a single cart item with details.
  + **Inventory.jsx** — Manages stock, search bar, and low-stock alerts.
  + **Product.jsx** — Reused for showing product with stock details.
  + **Sales.jsx** — Displays sales overview.
  + **SaleRecord.jsx** — Lists recorded sales transactions.
  + **AddProduct.jsx** — Form to add new items (name, image, price, stock, tags).
* **State Management** 
  + **Local State: Managed using React useState and useEffect.**
  + **API Integration: Axios used for fetching data from Keep Track of Inventory. API & YouTube API.**
* **Routing** 
  + **Library: react-router-dom**
  + **Routes:** 
    - **/ → ProductCatalog.jsx (Home page showing all products)**
    - **/cart → Cart.jsx (Shopping cart with CartItems)**
    - **/inventory → Inventory.jsx (Stock management with low-stock alerts)**
    - **/sales → Sales.jsx (Sales overview and records)**
    - **/sales/records → SaleRecord.jsx (Detailed sales history)**
    - **/add-product → AddProduct.jsx (Form to add a new product)**

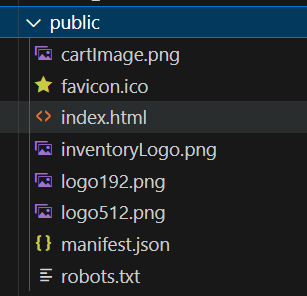
**4. Setup Instructions**

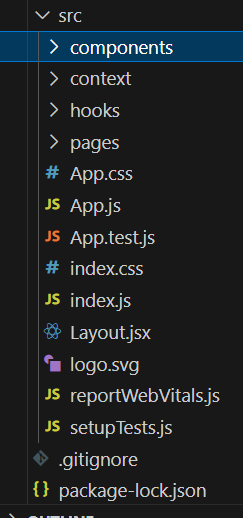
**Prerequisites**

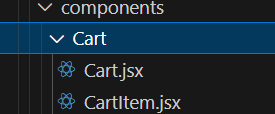
* **Node.js & npm** 
  + **Node.js is required to run React applications.**
  + **npm (Node Package Manager) is used to install dependencies.**
  + **Download Node.js**
* **React.js**
  + **React is the main JavaScript library used to build this project.**
  + **If you don’t have an existing React app, create one using:**
  + **npx create-react-app my-app**
  + **cd my-app**
  + **npm start**
  + **In SB Fitzz, the React app is already created, so you just need to install dependencies (npm install).**
* **Git** 
  + **Used for cloning and version control.**
  + **Download Git**
* **Code Editor**
  + **Recommended: Visual Studio Code (VS Code)**
  + **Download VS Code**
* **Basic Knowledge** 
  + **HTML, CSS, JavaScript**
  + **React concepts (components, props, hooks, state, routing)**
* **Installation**
  + **Get the code:**
    - **Download the code from the drive link given below:**
      * + <https://drive.google.com/drive/folders/1BPYYmXEAEmpDlvCQnz5HDGmhXloYzJ5c>
* **Install Dependencies:** 
  + **Navigate into the cloned repository directory and install libraries:**
    - **Navigate into the cloned repository directory and install libraries:** 
      * **cd CODE**
      * **npm install**
  + **Start the Development Server:** 
    - **To start the development server, execute the following command:** 
      * **npm start**
* **Access the App:**
  + **Open your web browser and navigate to http://localhost:3000.**
  + **You should see the application's homepage, indicating that the installation and setup were successful.**
* **Environment Variables** 
  + **Create a .env file with:**
    - **REACT\_APP\_API\_URL=<https://exercisedb.p.rapidapi.com/exercises/equipmentList>**
    - **REACT\_APP\_YOUTUBE\_API\_KEY=<33cf3a7616msh4c3b1e3204f24e2p1294b3jsne16a7323d732>**

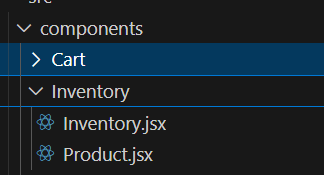
**5. Folder Structure**

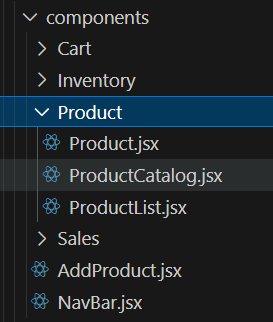


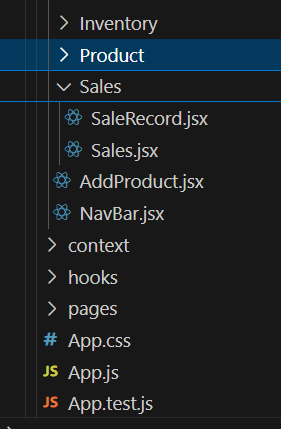












**6.Running the Application**

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

**7. Component Documentation**

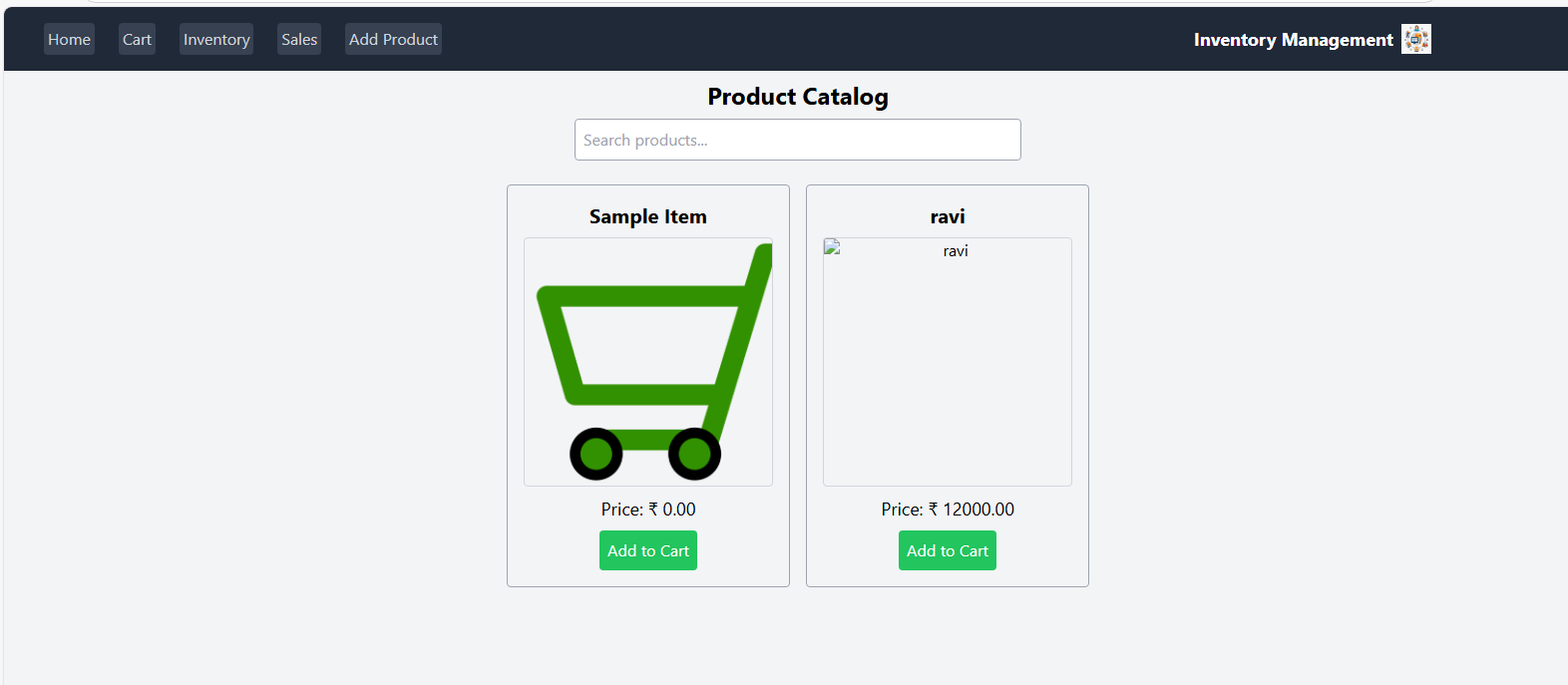
* **index.js** → Renders <App /> as the root component.
* **App.js** → Wraps NavBar, defines routes, and includes layout.
* **NavBar.jsx** → Provides navigation (Home, Cart, Inventory, Sales, Add Product).
* **ProductCatalog.jsx** → Homepage showing all available products.
* **Cart.jsx** → Displays items added to the cart.
* **Inventory.jsx** → Manages stock with search and low-stock alerts.
* **AddProduct.jsx** → Form to add new products to the inventory.

**8. State Management**

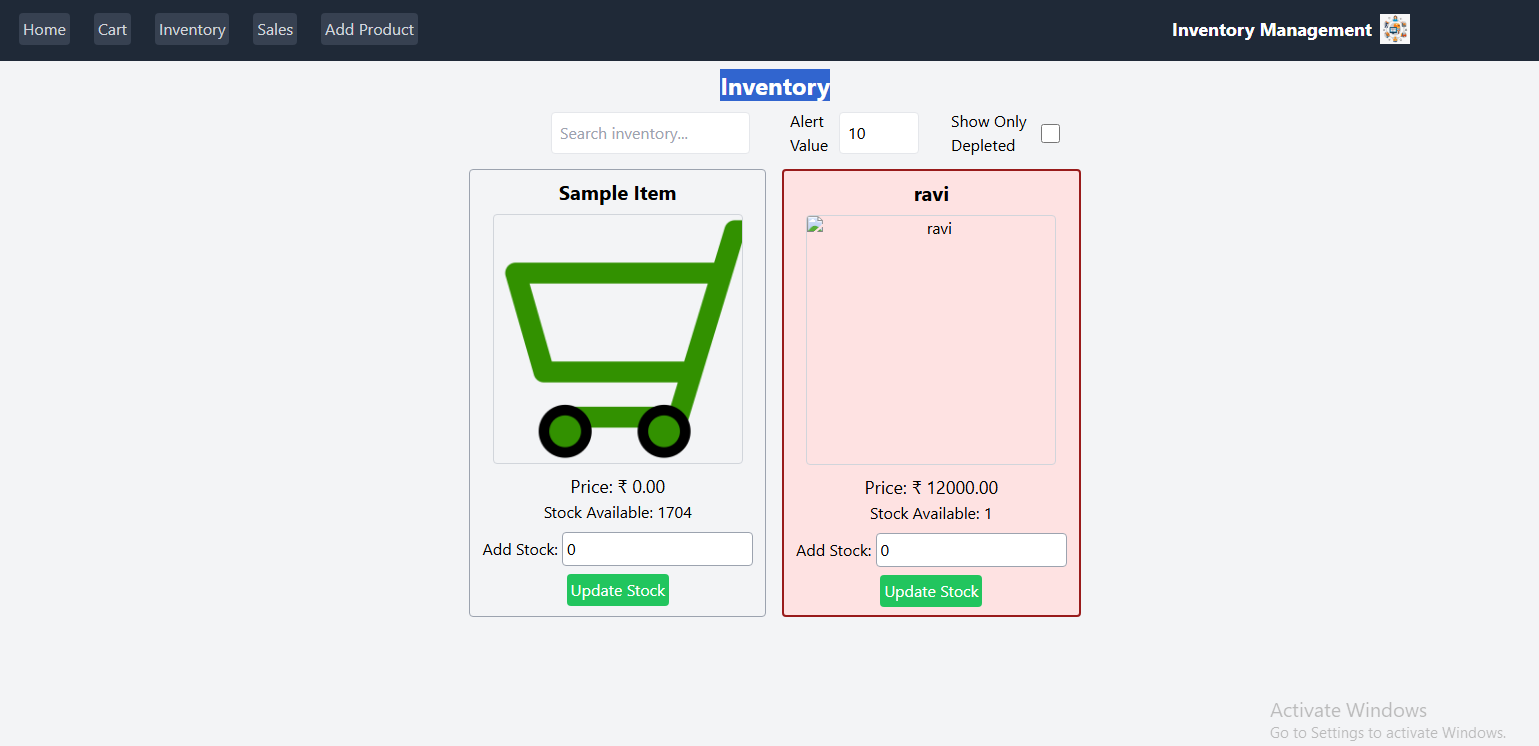
* **Local State:** 
  + **Search queries stored in HomeSearch.**
  + **API data fetched and stored per-page.**
* **Global State:**
  + **Not implemented — app uses component-level state.**

**9. User Interface**

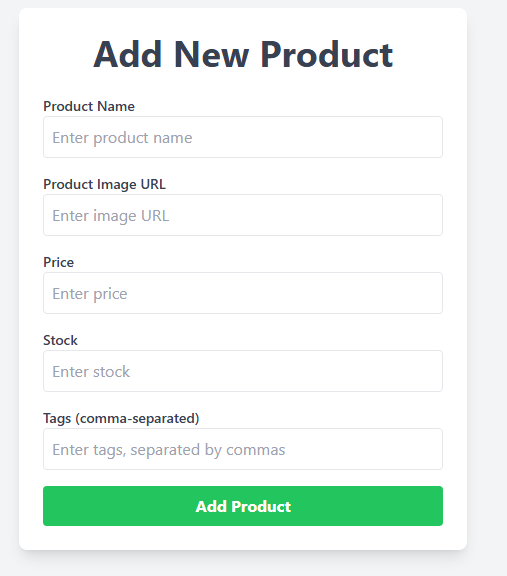
* **Pages include:**



* **Inventory**



**Add New Product**



**10. Styling**

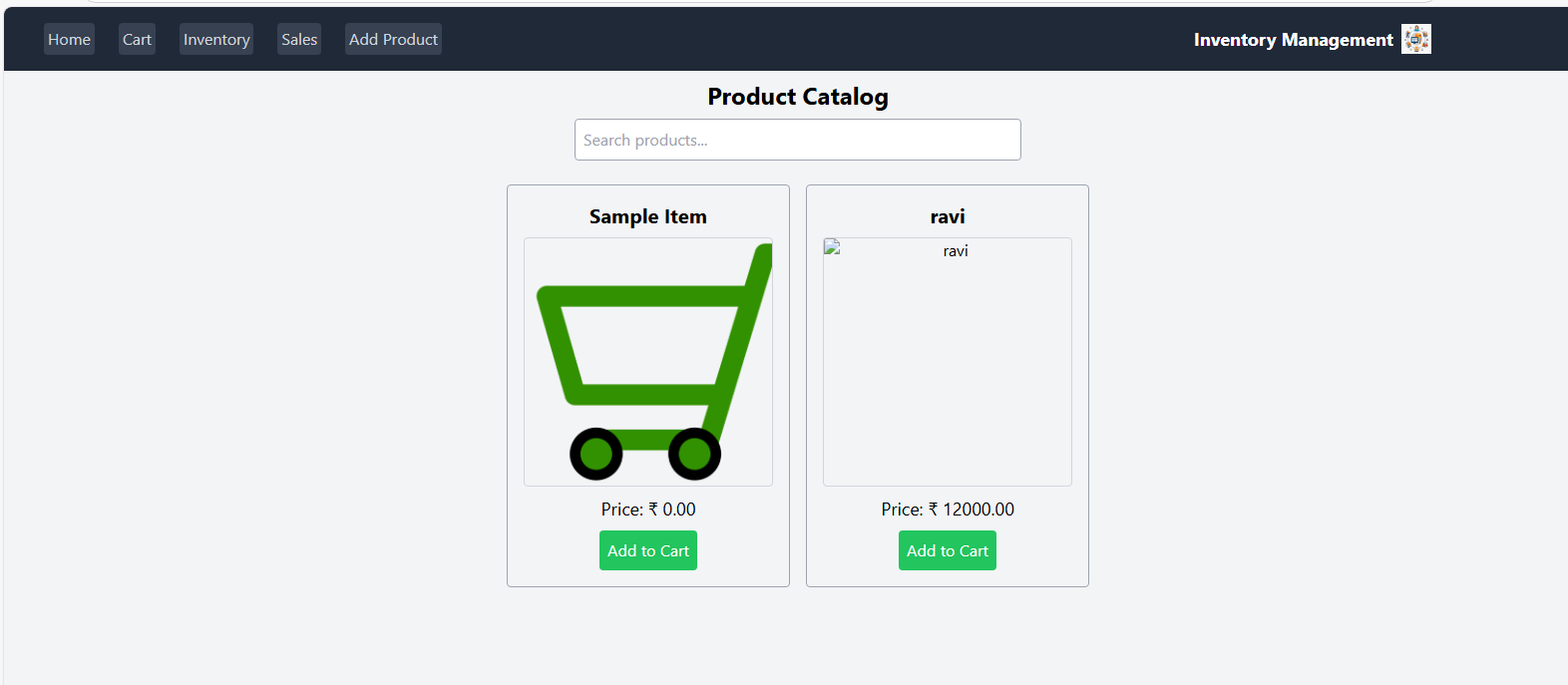
* **Frameworks Used: Tailwind CSS / Bootstrap.**
* **Custom CSS: Stored in src/styles/.**
* **Each page/component has a dedicated CSS file for modularity.**

**11. Testing**

* **Libraries Used: Jest, React Testing Library.**
* **Unit Tests: Written in App.test.js.**
* **Setup: Configured with setupTests.js.**

**12. Screenshots / Demo**

* **Demo Link:**
  + [**VID-20250914-WA0002.mp4**](https://drive.google.com/file/d/1nrQFQHyOrRHz_CZaIvFZkd_1089RFGpk/view?usp=drivesdk)
* **Screenshot:**



**13. Known Issues**

* + **API rate-limit may cause some exercises not to load.**
  + **YouTube API sometimes fails to fetch related videos.**

**14. Future Enhancements**

* **User Authentication** – Allow admins and staff to log in with different access levels.
* **Supplier Management** – Track suppliers, purchase orders, and restocking history.
* **Advanced Search & Filters** – Search products by category, price range, or stock status.
* **Reports & Analytics** – Generate sales reports, profit analysis, and inventory trends.
* **Notifications & Alerts** – Email or SMS alerts for low stock and important updates.
* **Multi-Store Support** – Manage inventory across multiple store locations.
* **Cloud Database Integration** – Connect with a backend (e.g., MongoDB/Firebase) for real-time data storage.

.