STORE MANAGER:KEEP

TRACK OF INVENTORY

TEAM MEMBERS:
1.Dharshini.L-(team leader)
2.Hemavadhi.E
3.Kaviya.N
4.Arockiya Emima.R
5.Kalaivani.S

Introduction

The Inventory Management System is a web-based application designed to help businesses manage their stock efficiently. The system allows users to track, update, and organize products available in a store. It reduces manual errors, provides quick access to product details, and ensures smooth day-to-day operations.

Key Objectives:
- Maintain real-time inventory data.
- Provide a simple interface for adding, updating, and deleting products.
- Ensure transparency in stock management.
- Support scalability for future features like billing, analytics, and reporting.

Technology Stack:
- Frontend: React.js (with Create React App)
- Build Tools: Webpack, Babel (via react-scripts)
- Package Manager: npm
- Styling: CSS (optionally with Bootstrap/Tailwind if added)
- Optional Backend (if included): Node.js + Express + Database (MongoDB/MySQL)

Project Setup

Prerequisites:
- Install Node.js (LTS recommended).
- npm (comes bundled with Node).

Installation Steps:

1. Clone or download the project folder.git clone cd inventory

2. Install dependencies.npm install

3. Start the development server.npm start
Local: http://localhost:3000
Network: http://:3000

4. Create a production build.npm run build
This generates optimized static files in the build/ folder.

Project Structure

A typical structure looks like this:

inventory/
■■■ node_modules/ # Installed dependencies
■■■ public/ # Static files (index.html, images, icons)
■■■ src/ # Main source code
■ ■■■ App.js # Root React component
■ ■■■ index.js # Entry point for React DOM rendering
■ ■■■ components/ # Reusable UI components
■ ■ ■■■ Header.js # Navigation bar or header
■ ■ ■■■ Inventory.js # Main inventory management screen
■ ■ ■■■ ItemForm.js # Form for adding/editing products
■ ■■■ styles/ # CSS files
■ ■■■ utils/ # Helper functions
■■■ package.json # Project dependencies & scripts
■■■ README.md # Basic project info

Explanation of Key Files:
- App.js – Central component that controls navigation and layout.
- index.js – Renders the App component into index.html.
- Inventory.js – Displays the list of products, search bar, and stock details.
- ItemForm.js – Form for adding or editing product details.
- Header.js – Provides navigation or branding for the app.
- package.json – Lists dependencies, scripts (npm start, npm run build), and metadata.

Code Walkthrough

Example Flow: Adding an Item
1. User clicks Add Item → opens ItemForm.js.
2. User fills product details → stored in local React state (via useState).
3. On form submit → the new item is sent to parent component (Inventory.js).

4. Inventory.js updates its list of items using useState or useReducer.
5. The UI refreshes automatically to display the new item.

Example Flow: Editing an Item
1. User clicks Edit → pre-fills ItemForm.js with existing data.
2. User updates values and submits.
3. Updated data replaces old item in the list.

Example Flow: Deleting an Item
1. User clicks Delete on a product.
2. A confirmation prompt is shown.
3. On confirmation, the item is removed from the state array.

React Features Used:
- Functional Components
- React Hooks (useState, useEffect)
- Props (passing data between parent and child components)
- Conditional Rendering (showing messages when inventory is empty)

Dependency Notes

During setup, you may see warnings like:
- Deprecated libraries (rimraf, glob, babel-preset-react-app). - Security vulnerabilities (npm audit).

Fixes:
npm audit fix
npx update-browserslist-db@latest
npm install --save-dev @babel/plugin-proposal-private-property-in-object
Long-term Recommendation:
- Migrate from Create React App to Vite (faster, modern build tool).
- Alternatively, use Next.js for server-side rendering and routing.

Usage Guide

Adding an Item:
- Navigate to the Add Item page.
- Enter details (name, quantity, price). - Click Save to add it to inventory.

Editing an Item:
- Select an item from the list.
- Click Edit → update details. - Save changes.

Deleting an Item:
- Select an item.
- Click Delete → confirm action.

Searching Items:
- Use the search bar to filter items by name or category.

Future Improvements

- Authentication – Add user login for security.
- Database Integration – Connect to MySQL/MongoDB for persistent data storage.
- Reporting Module – Generate sales/inventory reports.
- Barcode/QR Scanner – For fast product entry and lookup.
- Role Management – Different permissions for admin, manager, and staff.
- UI Improvements – Use Material UI or TailwindCSS.

Conclusion

The Inventory Management System simplifies stock management by providing a user-friendly interface and reliable product tracking. While the current version is suitable for small to medium stores, the project is easily extensible for enterprise-level features.