NAAN MUDHALVAN

PROJECT TITLE:STORE MANAGER KEEP TRACK OF INVENTORY

1. Introduction:

TEAM LEADER: P. Priyadharshini (demo video maker)

E-MAIL ID : deepakpriya222007@gmail.com

TEAM ID : NM2025TMID47173

Team Members:

Name	Mail-ID	Role
1. M. Gopika	- murugankumaran33@gmail.com	(document maker)
2. V.AISHWARYA	- aishwaryavelu2006@gmail.com	(code maker)
3. M. Deepika	- smadhappan050@gmail.com	(document maker)

2. Overview

Purpose:

- The Store Manager project is a web-based application designed to simplify inventory, sales, and product management for retail stores.
- It allows store managers to keep track of stock levels, manage product categories, process sales
- It is generate reports through a user-friendly interface.

Features:

Dashboard for quick store insight Product management (add, edit, delete products)Inventory tracking with stock alerts Sales management and history logs. User authentication and role-based access (admin/manager). Responsive design for desktop and mobile

3.Architecture

This architecture supports a scalable, modern web application that provides interactive recipe guidance, meal planning, and pantry management through an intuitive interface.

Frontend: React.js + Bootstrap + Material UI

Role:

The user interface that delivers a smooth, responsive, and interactive experience.

Technologies Used:

- React.js: Component-based structure for dynamic UI.

- Bootstrap: Layout grid system, responsiveness, and basic styling.

- Material UI: Modern, sleek UI components (buttons, cards, modals, etc.).

Backend: Node.js + Express.js

4. Setup Instructions

Prerequisites:

Node.js

Npm

Github

Installation:

1. Clone the repository :
   Git clone https://github.com/your-repo/store-manager.git
   Cd store-manager/client
2. Install dependencies: Npm install
3. Configure environment variables in .env:

REACT_APP_API_URL=http://localhost:5000/api

5.Folder Structure

```
Store management keep track inverory
 │
 ├── public/              # Static files (not processed by build tools)
 │   ├── index.html        # Main HTML file
 │   ├── favicon.ico       # Site icon
 │   └── assets/           # Images, fonts, videos (static)
 │
 ├── src/                 # Main source code
 │   ├── assets/           # Processed assets (images, fonts, svgs, icons)
 │   │   ├── images/
 │   │   ├── fonts/
 │   │   └── styles/       # Global styles (CSS/SCSS)
 │
 │   ├── components/       # Reusable UI components
 │   │   ├── Button.jsx
 │   │   └── Navbar.jsx
 □
 │   ├── pages/            # Page-level components (Home, About, etc.)
 │   │   ├── Home.jsx
 │   │   └── About.jsx
 □
 │   ├── layouts/          # Layouts for wrapping pages (Header/Footer)
 │   │   └── MainLayout.jsx
 □
 │   ├── hooks/            # Custom React hooks (if using React)
 │   │   └── useAuth.js
 □
 │   ├── services/         # API calls or external services
 │   │   └── api.js
 □
 │   ├── context/          # Context API or global state (React/Vue)
 │   │   └── AuthContext.jsx
 □
 │   ├── utils/            # Helper functions
 │   │   └── formatDate.js
 │   ├── App.js            # Root component
 │   ├── index.js          # Entry point
 │   └── routes.js         # Route definitions (if needed)
 │
 ├── .gitignore            # Files ignored by Git
 ├── package.json          # Dependencies & scripts
 ├── README.md             # Project documentation
 └── vite.config.js / webpack.config.js / next.config.js (depending on framework)
```

6.Running the Application

Start the development server:

Cd client

Npm start

Acess: http://localhost:3000

Component Documentation

Key Components:

o Navbar – Navigation links, user profile dropdown. o DashboardCard

– Reusable info card for sales/inventory.

o ProductTable – Displays products with search, sort, and actions.

- ➢ Reusable Components:
  - ○ Button – Custom styled button supporting variants (primary, secondary).
- ➢ Modal – For confirmations (delete product, update stock).
  - ○ FormInput – Reusable input with validation support.

8. State Management

○ **Global State:**

- ✓ AutoZone → Stores user authentication, roles, and JWT tokens.
- ✓ StoreContext → Manages product list, sales history, and stock data.

o Local State:

- ✓ Handled via useState in forms, search filters, and modals.

9. User Interface

- ➢ (Screenshots to be inserted after UI implementation) Dashboard with metrics cards.
  Product management page with search and CRUD options.
  Sales entry page with invoice generator.
- ➢ Login page with authentication.

10. Styling

CSS Frameworks/Libraries:

1. Tailwind CSS for utility-first styling.
2. React Icons for iconography.
3. Theming:
4. Custom color palette defined in tailwind.config.js.
5. Dark mode toggle under consideration.

11.Testing

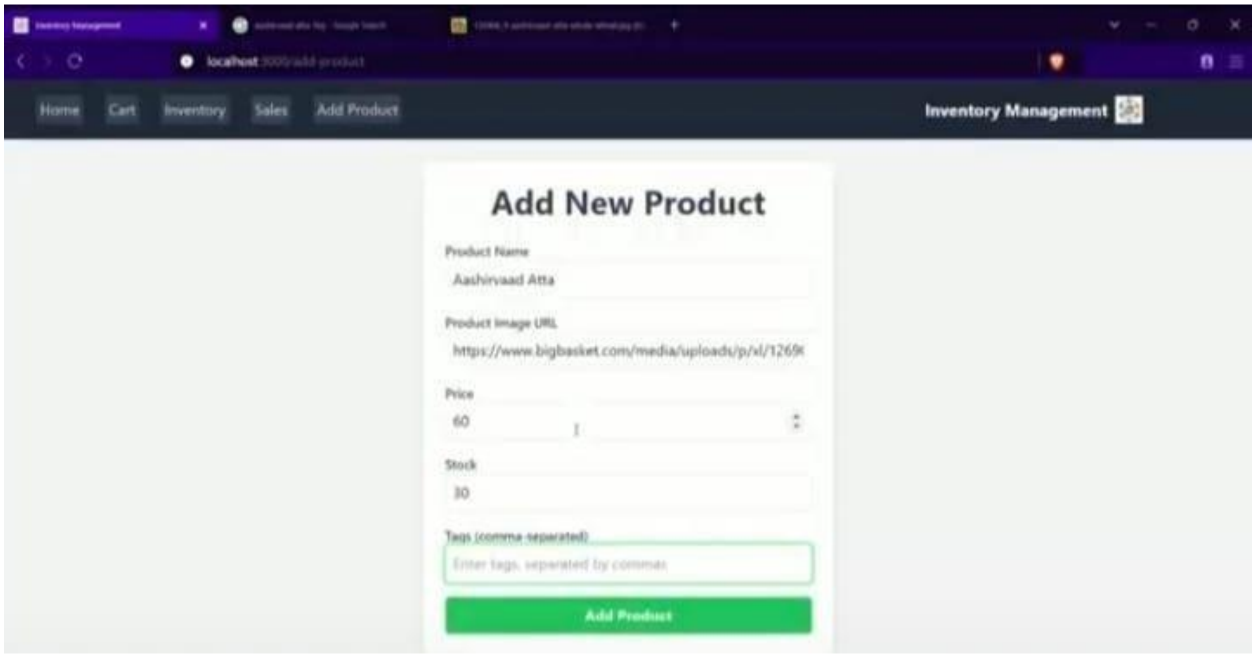
Testing Strategy:

- ✓ Unit tests with Jest & React Testing Library for components.
- ✓ Integration tests for API calls and context.
- ✓ E2E tests planned using Cypress.

Code Coverage:

- ✓ Configured via Jest with coverage reports generated on build.

12.Screenshots or Demo



Add New Product

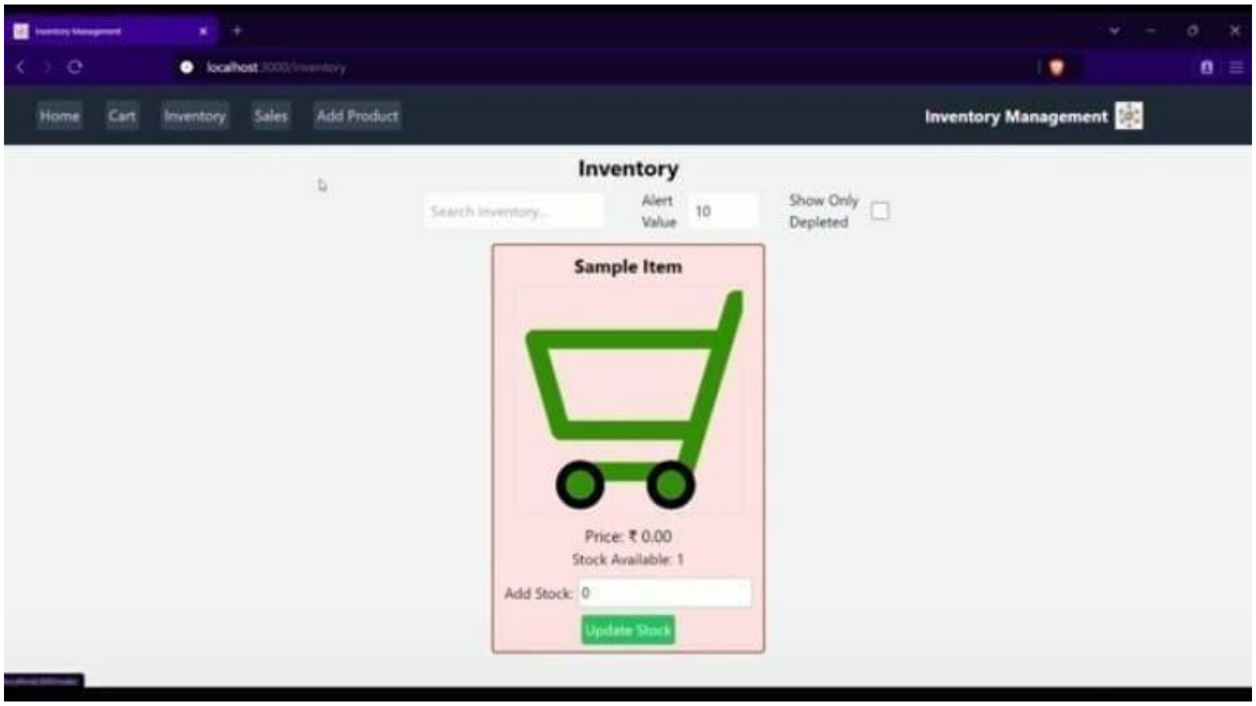
Product Name
Aashirvaad Atta

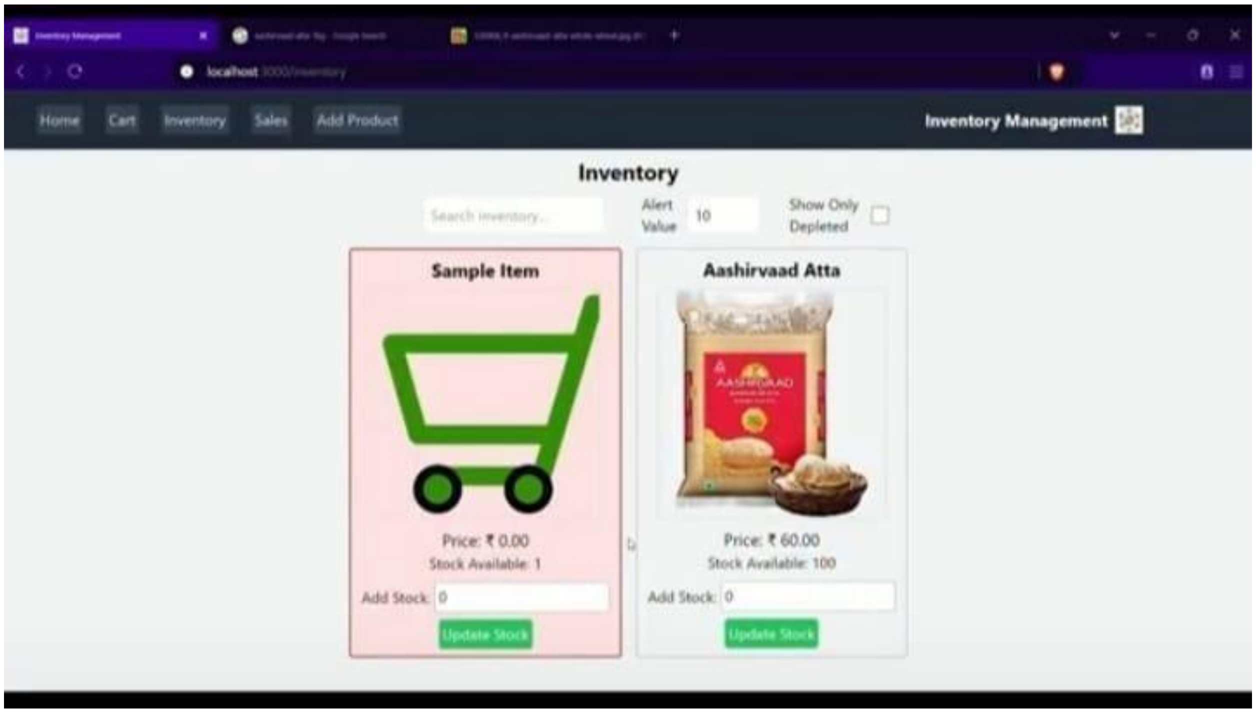
Product Image URL
https://www.bigbasket.com/media/uploads/p/xl/1269

Price
60

Stock
30

Tags (comma separated)
Enter tags, separated by commas


Home
Cart
Inventory
Sales
Add Product
Inventory Management
Inventory
Search inventory...
Alert Value
10
Show Only Depleted
Sample Item
Price: ₹ 0.00
Stock Available: 1
Add Stock:
0
Update Stock


Home
Cart
Inventory
Sales
Add Product
Inventory Management
Inventory
Search inventory...
Alert Value
10
Show Only Depleted
Sample Item
Price: ₹ 0.00
Stock Available: 1
Add Stock:
0
Update Stock
Aashirvaad Atta
Price: ₹ 60.00
Stock Available: 100
Add Stock:
0
Update Stock

13. Known Issues

- ✓ Product search filter needs optimization for large datasets.
- ✓ Sales reporting page still under development.
- ✓ Mobile responsiveness for tables requires improvement.

14. Future Enhancements

- ✓ Add barcode scanner support for product entry.
- ✓ Implement dark mode theme.
- ✓ Advanced analytics and chart visualizations.
- ✓ Offline mode with PWA support.