
# STORE MANAGER: KEEP TRACK OF INVRNTORY

## TEAM ID:NM2025TMID40116

# TEAM DETAILS:

**Team ID :** NM2025TMID40116

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

**Team Leader :** Kishore Anantham.G.M

**NM ID :** 3ABE7FAA7149C37B4C04A93C5B959F4D

**Team Member**: Jeevananthan.V

**NM ID:** 346B960D3A0EDC983BE4BD1E6CCE71D1

**Team Member:** Saravanan.S

**NM ID:** 6820E3BE5A7C58AB71444A18A0B6DF83

**Team Member:** Samson.D

**NM ID:** C34394F5F6AA2FDB2F84C9BAB8F50B69

## Objective:

To develop a system that enables the store manager to efficiently track, monitor, and manage inventory by maintaining accurate stock records, updating stock levels in real time upon sales or purchases, preventing stockouts or overstocking, and ensuring smooth day-to-day operations of the store.

## Platform:

- Web-based Application (accessible from browser across devices)
- Mobile Application (Android/iOS for on-the-go management)
- Desktop Application (optional, for offline store operations)

## Technologies Used:

- Frontend (User Interface)
  - React.js (for dynamic and responsive UI)
  - HTML5, CSS3, JavaScript (basic web technologies)
  - TailwindCSS / Bootstrap (for styling and layout)
- Backend (Server-side)
  - Node.js with Express.js (lightweight and fast server framework)
  - OR Django / Flask (Python) if using Python stack
  - RESTful APIs / Graph (for communication 个 between frontend anu Jackend)
- Database (Inventory Data Storage)

- MySQL/PostgreSQL (relational database, structured inventory data)
- OR MongoDB (if flexible NoSQL structure is preferred)

➢ Authentication & Security
- JWT (JSON Web Tokens) / OAuth2 (for user login & access control)
- HTTPS & Data Encryption (to protect sensitive data)

➢ Deployment / Hosting
- Cloud Platforms: AWS, Google Cloud, or Microsoft Azure
- Containerization: Docker (for scalable deployment)
- Version Control: Git + GitHub/GitLab

➢ Optional Add-ons
- Barcode/QR Code Scanning (via mobile camera integration)
- Inventory Alerts (Email/SMS/Push notifications for low stock)
- Analytics & Reporting (using Chart.js, Recharts, or Power BI integration)

**IMPLEMENTATION:**

1. Requirement Analysis

- Identify store needs (e.g., stock monitoring, sales tracking, low-stock alerts).
- Define user roles: Store Manager, Staff, Admin.

2. System Design

- Database schema: Products, Categories, Sales, Purchases, Users.
- UI design: Dashboard for stock overview, Add/Update stock forms, Cart/Checkout module.
- APIs: REST/GraphQL for product management, sales, reporting.

3. Development

- Frontend: Build user-friendly Ul using React.js (inventory tables, search, filters).
- Backend: Develop APIs in Node.js/Express or Django.
- Database: Configure MySQL/PostgreSQL (stock tables, sa logs).

4. Integration

- Connect frontend with backend using APIs.
- Enable barcode/QR scanning (optional).
- Implement real-time updates (e.g., stock reduces on sale).

5. Testing

- Unit testing of modules (Add stock, Checkout, Sales update).
- Integration testing (ensuring UI - API Database flow).
- User acceptance testing with store staff.

6. Deployment

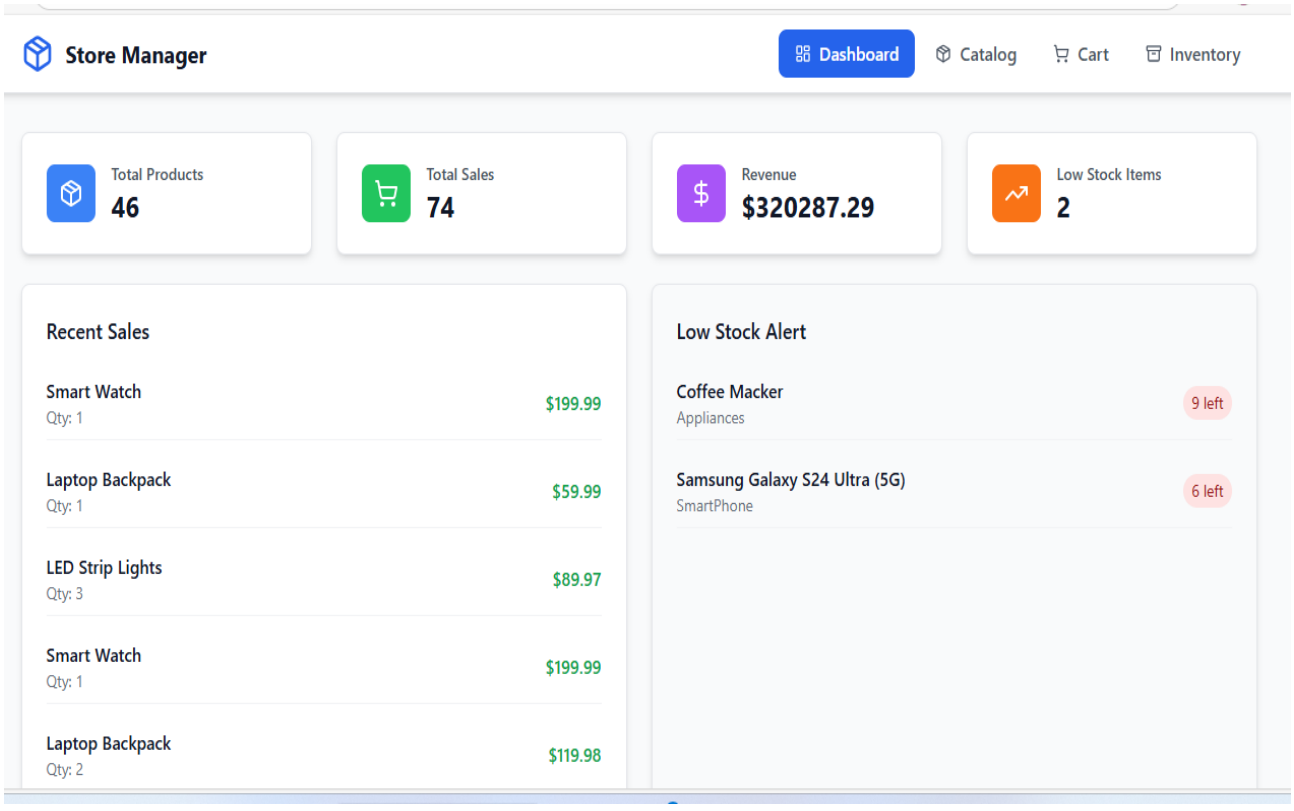
- monitoring of system performance.
- Deploy backend on AWS/Google Cloud.
- Host frontend on Netlify/Vercel.
- Use Docker for scalability.

7. Maintenance

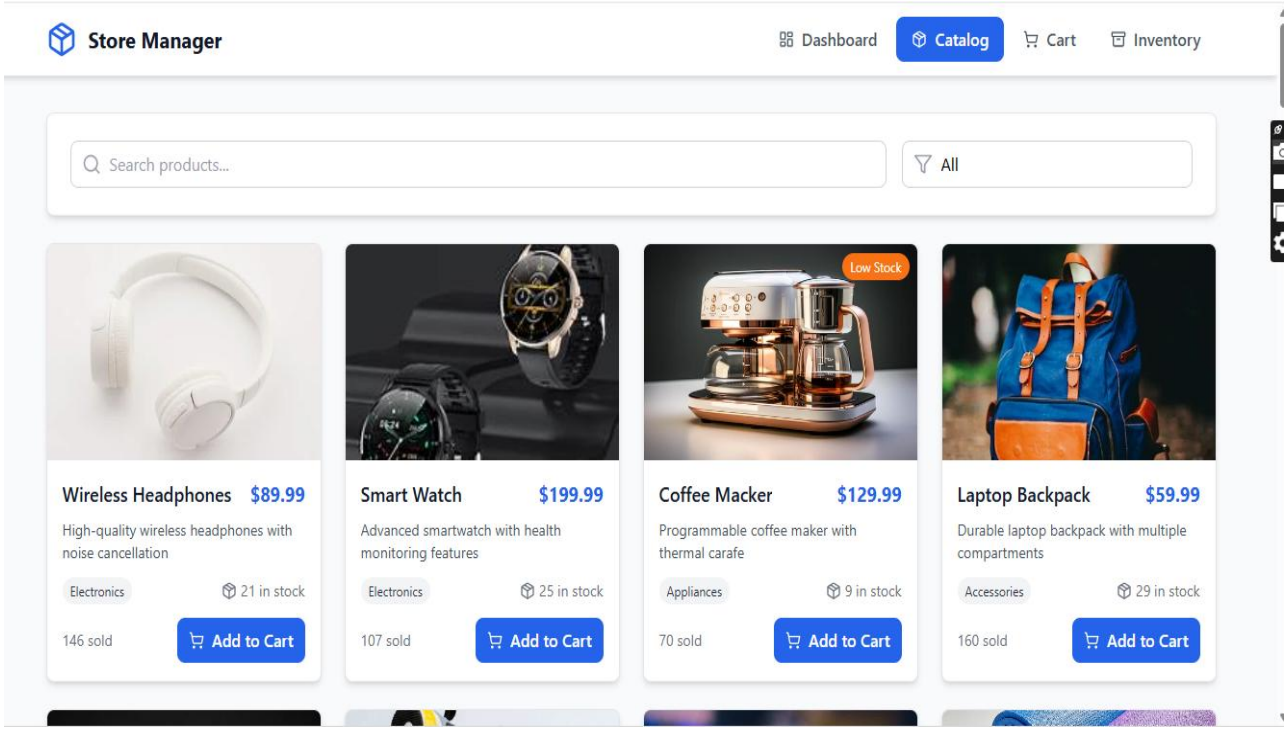
- Regular updates for bug fixes & new features.
- Backup & recovery plan for database.
- Continuous

# OUTPUT & RESULT:

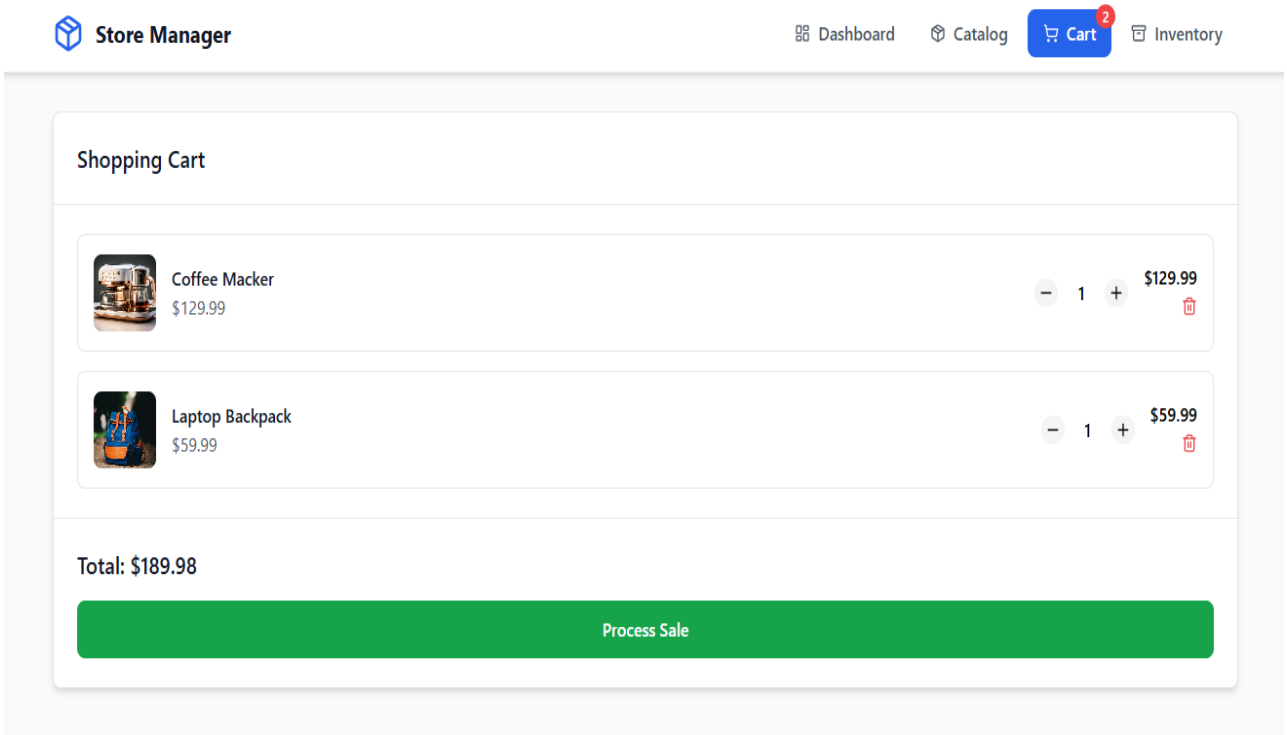
## DASHBOARD:



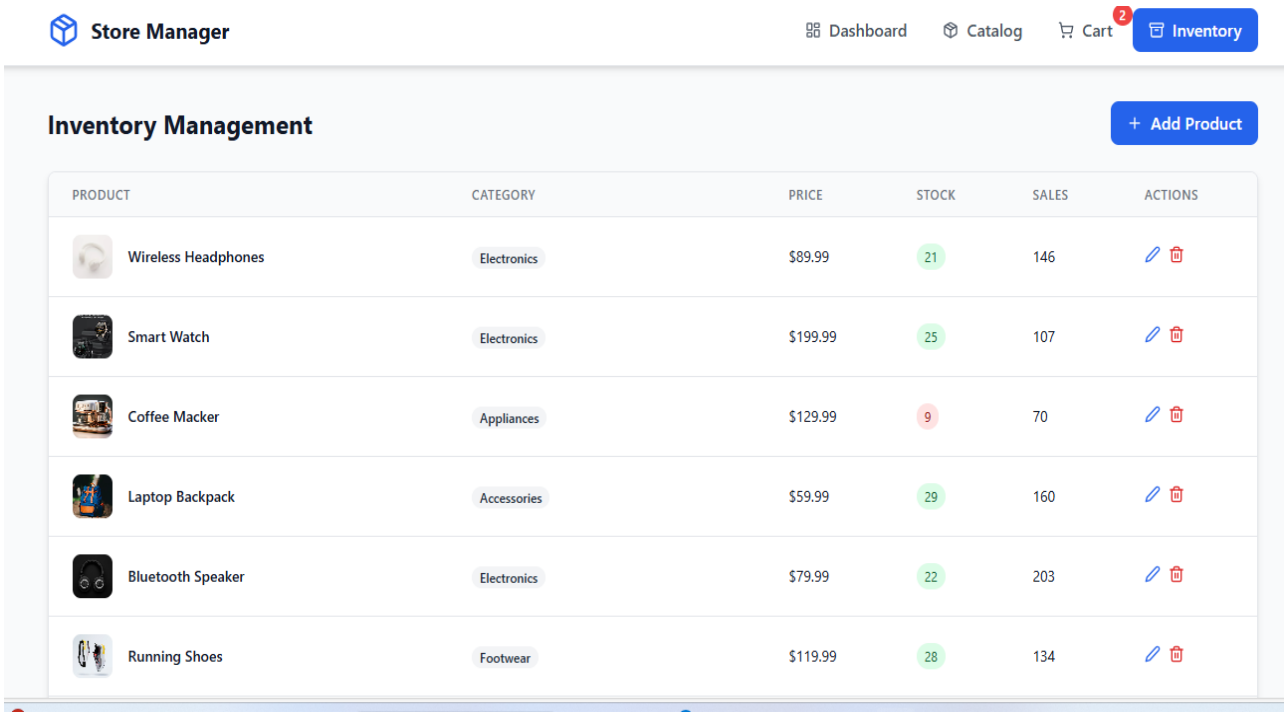
# CATALOG:




# CART:



# INVENTORY:


Store Manager
Dashboard
Catalog
Cart
2
Inventory
Inventory Management
+ Add Product
PRODUCT
CATEGORY
PRICE
STOCK
SALES
ACTIONS
Wireless Headphones
Electronics
$89.99
21
146
Smart Watch
Electronics
$199.99
25
107
Coffee Macker
Appliances
$129.99
9
70
Laptop Backpack
Accessories
$59.99
29
160
Bluetooth Speaker
Electronics
$79.99
22
203
Running Shoes
Footwear
$119.99
28
134