**STORE MANAGER KEEP TRACK OF INVENTORY**

****

Submitted by:-

* Team id: NM2025TMID37866
* Team Leader: S. Arthi & [arthi01102006@gmail.com](mailto:arthi01102006@gmail.com)
* Team Members:
* A.K. Kokilavani & [kokianbu25@gmail.com](mailto:kokianbu25@gmail.com)
* R.Ishwarya & [ishwaryashanthi20@gmail.com](mailto:ishwaryashanthi20@gmail.com)
* S.Kanchana Priya & [priyadhachu18@gmail.com](mailto:priyadhachu18@gmail.com)

Index

|  |  |  |
| --- | --- | --- |
| **S.no** | **Content** | **pg.no** |
| 1. | Introduction | 3 |
| 2. | Project Overview:   * Purpose * Features | 3 |
| 3. | Architecture | 4 |
| 4. | Start Management | 4 |
| 5. | Setup Instructions | 5 |
| 6. | Installation Steps | 6 |
| 7. | Folder Structure | 6 |
| 8. | Running the Application | 7 |
| 9. | API Documentation | 8 |
| 10. | Authentication | 9 |
| 11. | User Interface | 10 |
| 12. | Testing | 11 |
| 13. | Screenshots or Demo | 12 |
| 14. | Known Issues | 15 |
| 15. | Future Enhancement | 15 |

**INTRODUCTION:**

Inventory tracking is the process store managers use to monitor, record, and control all products within a retail environment, making sure the right items are available at the right time and location

Effective inventory tracking ensures stock levels are accurate, prevents overstocking and stock outs, supports timely reorder decisions, reduces loss, and improves customer satisfaction. For store managers, it streamlines workflows, aligns sales and purchasing, and supports informed decision-making through reliable inventory data.

**PROJECT OVERVIEW:**

* **PURPOSE:**
* improve inventory accuracy: Reduced error and discrepancies in inventory tracking.
* Enchance inventory control: Provide real-time Visiliablity into inventory: level and movement.
* Stock level: Enable business to maintain stock level, reducing stock out and overstocking
* **FEATURE:**
* Inventory Tracking: Track inventory levels, movements, and    locations.
* Real-time Updates: Provide real-time updates on inventory levels and movements.
* Alerts and Notifications: Send alerts and notifications for low stock levels, stock outs, or other inventory-related issues.
* Reporting and Analytics: Generate reports and analytics on inventory levels, movements, and trends

**ARCHITECTURE:**

** Frontend:**

* User Interface (UI) where the store manager interacts with the system.
* React.js with Bootstrap and Material UI.
* Responsive web interface for managing inventory, viewing items, adding new products, and updating stock levels.

** Backend:**

* API that connects the frontend with the database. The backend handles the logic, data processing, and interaction with the database.
* Business logic for CRUD operations (Create, Read, Update, Delete), inventory management, stock alerts, etc.
* Node.js and Express.js managing server logic and API endpoints

 **Database**:

* A relational or NoSQL database to store product information, stock levels, transaction history, and user data.

**START MANAGEMENT:**

* Inventory Initialization: Initialize inventory levels, product information, and other relevant data
* Inventory Updates: Update inventory levels based on sales, purchases, and other inventory movements.

* Inventory Tracking: Track inventory levels, movements, and  locations

* Notification System: Sends alerts and notifications for inventory-related issues.

**ROUTING:**

1. Inventory Routes: Define routes for inventory management, such as:

* Display inventory levels and movements.
* Add new inventory items.
* Update existing inventory items.

2. User Routes: Define routes for user management, such as:

* Display user information.
* Add new users.
* Update existing user information.

**SETUP INSTRUCTIONS:**

**Pre-requisites:**

* **Node.js**

Install Node.js to run JavaScript code outside the browser.

It is required for building and running the backend server.

* **MongoDB**

Set up MongoDB to store and manage application data.

It provides a NoSQL database solution for your project.

* **Git**

Use Git for version control and project collaboration.

It helps you track changes and manage code efficiently.

* **React.js**

React.js is used for building the frontend user interface.

It makes the UI dynamic, reusable, and responsive.

* **Express.js**

Express.js is a framework for Node.js to handle APIs and routing.

It simplifies backend development and server management.

* **Mongoose**

Mongoose is used to connect Node.js with MongoDB.

It provides a schema-based solution for modeling application data.

* **Visual Studio Code**

VS Code is the recommended code editor for development.

It offers useful extensions and debugging tools.

**INSTALLATION STEPS:**

* Clone the Repository git clone:

**git Clone <repository-url>**

* Install client dependencies :

**cd client**

**npm install**

* Install server dependencies:/server

**cd client**

**npm start**

**FOLDER STRUCTURE:**

*STORE-MANAGER INVENTORY:*

* *client/ # React frontend components*
* *src/*
* *components/ # Reusable UI parts*
  + *Navbar.jsx*
  + *Sidebar.jsx*
  + *InventoryTable.jsx*
  + *ProductCard.jsx*
* *pages/ # Main pages*
* *Dashboard.jsx # Stock overview*
* *Products.jsx # Product list*
* *AddProduct.jsx # Add new product form*
* *Sales.jsx # Track sales/orders*
* *Reports.jsx # Stock and sales reports*
* *App.js*
* *index.js*
* *package.json*
* *server/ # Node.js backend*
* *routes/ # API routes*
* *productRoutes.js # Product CRUD*
* *salesRoutes.js # Sales/orders*
* *userRoutes.js # Auth/login*
* *models/ # MongoDB schemas*
* *Product.js # Product schema*
* *Sale.js # Sales schema*
* *User.js # User schema*
* *controllers/ # Business logic*
* *productController.js # Handle product actions*
* *salesController.js # Handle sales/orders*
* *userController.js # Handle authentication*
* *server.js # Main Express app*
* *package.json*
* *README.md*
* **Client:**

Inventory Management Client: A web or mobile application that allows users to:

* + View inventory levels and movements
  + Add, edit, and delete inventory items
  + Track inventory levels and receive alerts
  + Generate reports and analytics
* **Utilization:**
* Inventory Tracking: Utilize the system to track inventory levels, movements, and locations.

* Reporting and Analytics: Generate reports and analytics on inventory levels, movements, and trends.

* Alerts and Notifications: Set up alerts and notifications for low stock levels, stock outs, or other inventory-related issues.

* Integration: Integrate the system with other business applications, such as accounting or e-commerce platforms.

**RUNNING THE APPLICATION:**

* On your **computer/phone** → you “run” an app by opening it.
* For a **web app** → running means starting the code (backend + frontend) so users can access it in a browser.

**Frontend**

This is the **part of the application the user sees and interacts with**.

* Examples: buttons, menus, product list, search bar.
* Built with: **HTML, CSS, JavaScript** (and frameworks like React, Angular, Vue).
* It’s like the **storefront of a shop** — customers see and use it.

**Backend**

This is the **engine behind the scenes** that processes data and makes things work.

* Examples: handling login, saving inventory data, calculating stock levels.
* Built with: **Node.js, Python (Django/Flask), Java (Spring), PHP, etc.**
* Uses **databases** like MySQL, PostgreSQL, or MongoDB to store inventory.
* It’s like the **storage room and staff** — not visible to customers, but essential.
* Example: This will start the frontend server, and you can access the application at http://localhost:3000 (or the specified port).

**API DOCUMENTATION:**

API documentation for a store manager to keep track of inventory would typically cover a range of functionalities, enabling the management of products, stock levels, and related data. Key sections and features commonly found in such documentation include:

**1. Product Management:**

* **Create Product:** Allows adding new products with details like SKU, name, description, price, and initial stock quantity.
* **Retrieve Product:** Enables fetching product information by ID, SKU, or other identifiers.
* **Update Product:** Permits modifying existing product details, including pricing, description, or other attributes.
* **Delete Product:** Provides the functionality to remove a product from the inventory.

**2. Inventory Management**:

* **Update Stock Levels:**

Adjusts the quantity of a specific product, either by increasing (e.g., for new shipments) or decreasing (e.g., for sales or returns).

* **Get Stock Levels:**

Retrieves the current stock quantity for a given product or across multiple locations.

* **Track Product Movement:**

Records and retrieves information on the movement of products between warehouses, stores, or other locations.

* **Manage Variants and Bundles:**

Handles inventory for product variations (e.g., size, color) or bundled items, ensuring accurate stock updates for all components.

**3. Location Management (for multi-location stores):**

* **Create Location:** Adds new storage or sales locations (e.g., warehouses, retail stores).
* **Retrieve Location Inventory:** Fetches the inventory details specific to a particular location.
* **Transfer Stock:** Facilitates the movement of products between different locations.

**4. Reporting and Analytics**:

* **Generate Inventory Reports:**

Provides data on stock levels, low-stock alerts, sales trends, and other relevant metrics.

* **Out-of-Stock Alerts:**

Configures and retrieves notifications for products nearing or reaching zero stock.

**5. API Details and Usage:**

* **Authentication and Authorization:**

Explains how to securely access the API, including required tokens or credentials.

* **Endpoint URLs:**

Lists the specific URLs for each API operation.

* **Request and Response Formats:**

Describes the expected data structure for sending requests (e.g., JSON, XML) and receiving responses.

* **Error Codes and Handling:**

Details potential errors and how to interpret and handle them in client applications.

* **Rate Limits:**

Specifies any limitations on the number of API requests allowed within a given timeframe.

**AUTHENTICATION:**

Authentication ensures that only authorized users (like store managers or staff) can access the inventory system. It protects sensitive store data such as stock levels, product details, and sales records.

**Key Points:**

* + User Login – Store managers and staff must log in using a username/email and password.
  + Role-Based Access – Different roles (Admin, Manager, Staff) can have different levels of access to features.
  + Secure Passwords – Passwords are stored in the database in encrypted form (e.g., bcrypt hashing).
  + Session or Token – After login, a session (cookie-based) or token (JWT – JSON Web Token) is generated to verify the user for further actions.
  + Logout Function – Users can log out to end their session securely.

**Example Flow:**

* Manager opens the app → enters login credentials
* Backend verifies credentials against database
* If correct → system issues a JWT/session token
* Manager now has access to inventory dashboard
* Unauthorized users cannot access without valid credentials

**USER INTERFACE:**

* Dashboard: A central dashboard that provides an overview of the inventory, including low stock alerts and other important information.
* Inventory Table: A table view of inventory items, with columns for name, quantity, location, and other relevant information.
* Form Fields: Form fields for adding and editing inventory items, including fields for name, quantity, location, and other relevant information.
* Buttons: Buttons for adding, removing, and editing inventory items, as well as for updating quantities.

* Alerts/Notifications: Alerts and notifications for low stock levels, inventory discrepancies, and other important.

**TESTING:**

**Testing for Store Manager Inventory System:**

**1. Authentication & Access:**

* Test login with valid and invalid credentials.
* Ensure only authorized managers can access the system.
* Verify session timeout and logout functionality.

**2. Add Inventory:**

* Add new products with correct details (name, quantity, price, category).
* Test adding duplicate products (should update or show error).
* Check mandatory fields validation.

**3. Update Inventory:**

* Edit product details (price, quantity, category).
* Verify real-time stock update after sales or restock.
* Ensure negative stock values are not allowed.

**4. Delete Inventory:**

* Delete an item and check if it is removed from records.
* Test accidental deletion with confirmation prompt.

**5. Search & View Inventory**

* Search by product name, ID, or category.
* Sort and filter items (low stock, high price, etc.).
* Check pagination for large inventory lists.

**6. Stock Alerts**

* Low stock notification when item drops below threshold.
* Out-of-stock warning before sale.

**7. Reports & Analytics**

* Generate daily/weekly/monthly stock reports.
* Test export to PDF/Excel.
* Verify correct calculation of total inventory value.

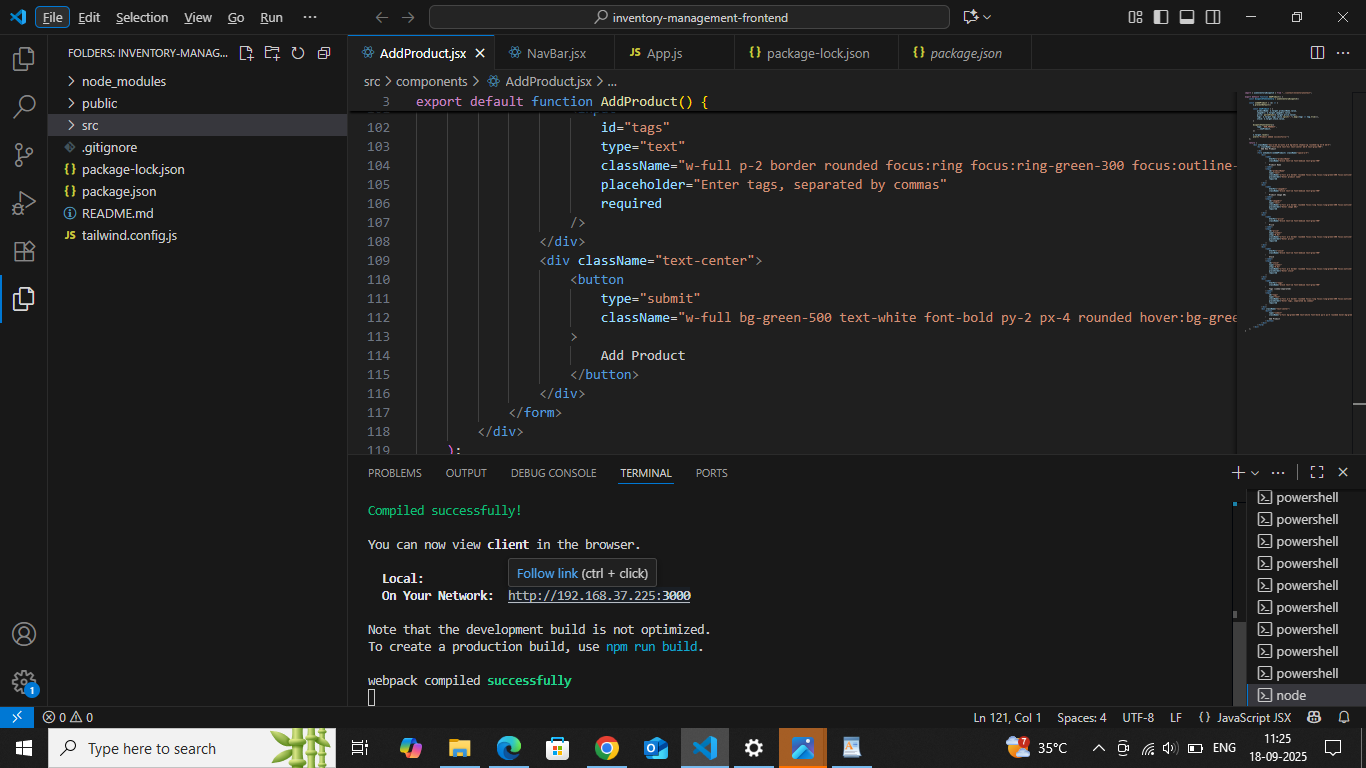
**8. Performance & Security**

* Test system with large inventory data (scalability).
* Verify secure password storage (hashing).
* Check for SQL injection or invalid input handling.

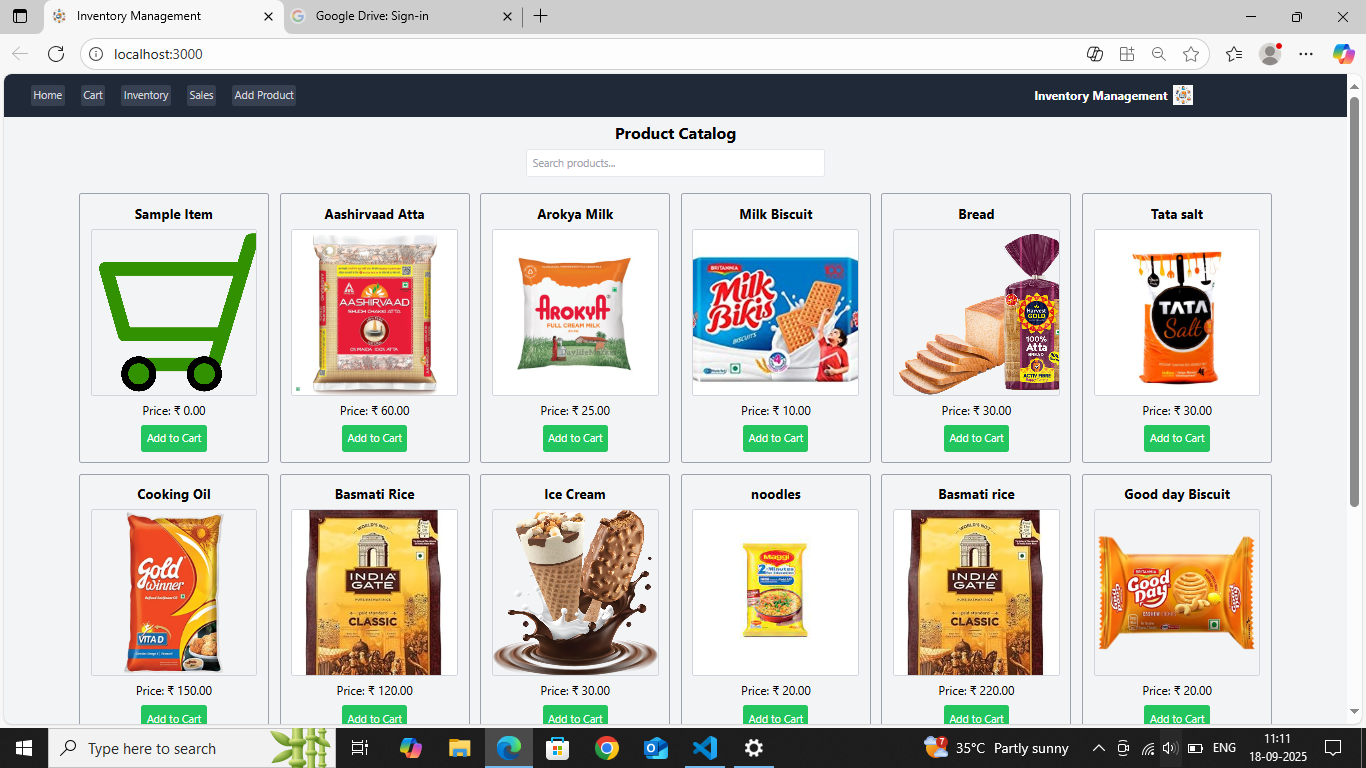
**DEMO LINK:**

<https://drive.google.com/file/d/1ZzZUSCJa613tFxQJwa-3DByjnhwW3P4l/view?usp=drive_link>

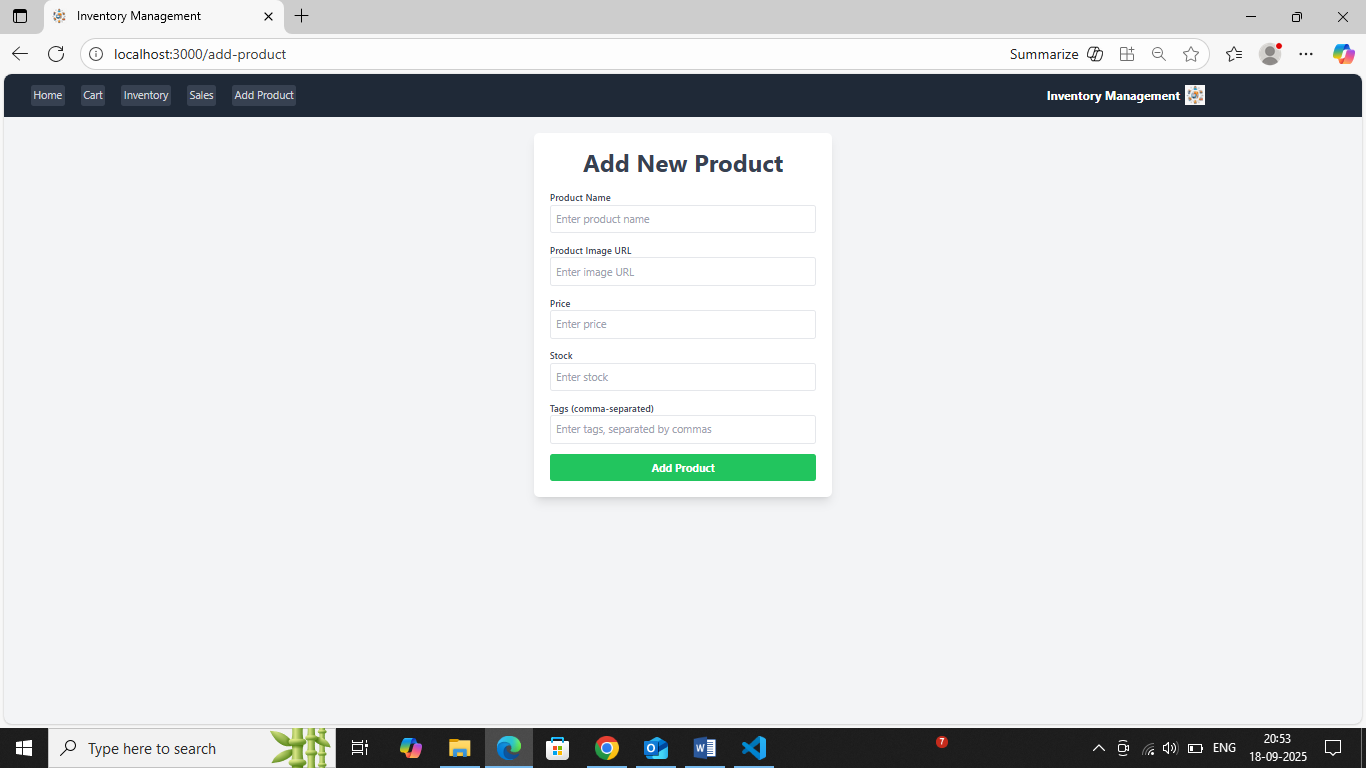
**SCREENSHOTS:**



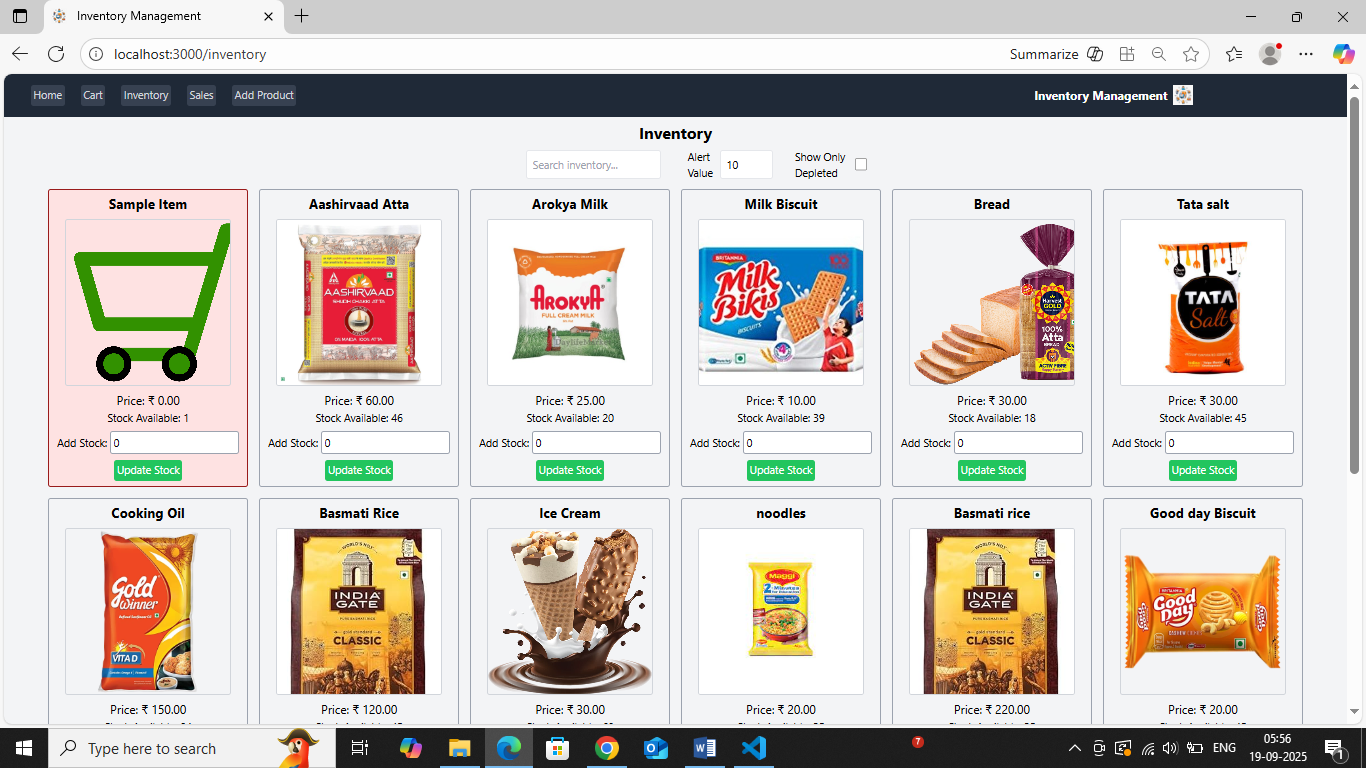
**PRODUCT CATALOG:**

****

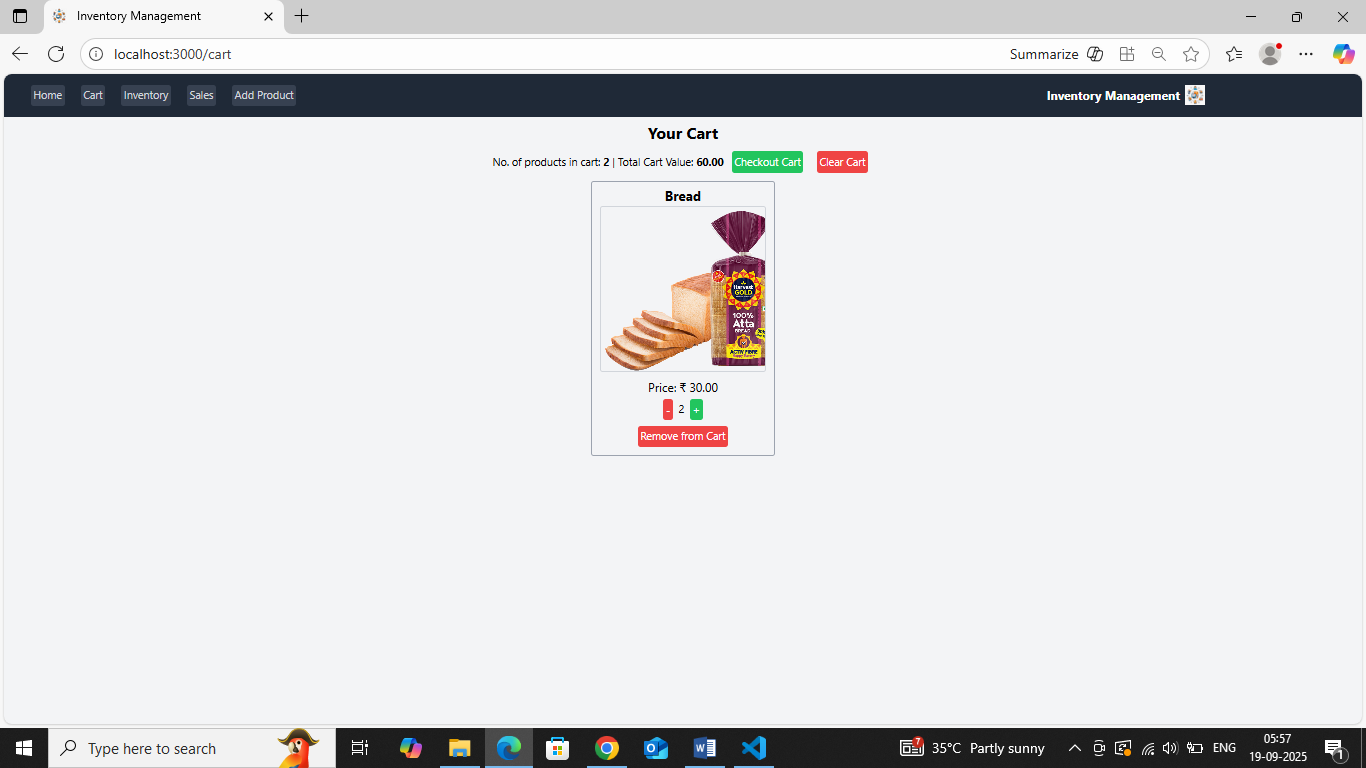
**ADD PRODUCT:**

****

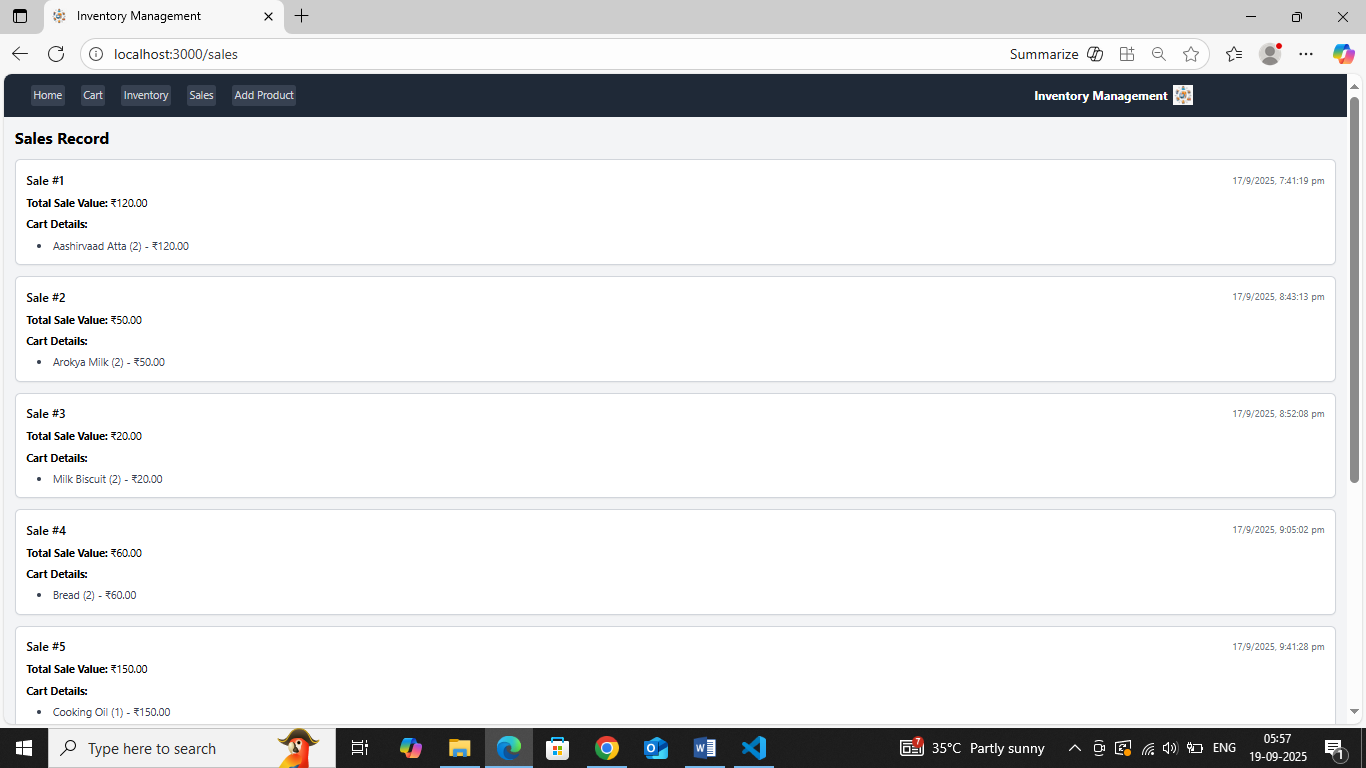
**INVENTORY:**

****

**CART**:



**SALES:**

****

**KNOWN ISSUES:**

* Sometimes duplicate products are added instead of updating existing ones.
* Negative stock values are allowed when updating items.
* Deleted items still show up in some reports.
* Search is slow when many products are stored.
* Low-stock alerts do not always trigger on time.
* Reports sometimes give wrong total value if data is missing.
* System becomes slow when handling very large inventory.

**FUTURE ENCHANCEMENTS:**

1. Barcode/QR Code Scanning – Quickly add or update items using a scanner.

2. Mobile App Access – Manage inventory from phone or tablet.

3. Cloud Backup – Automatic backup of all inventory data.

4. Multi-Store Support – Handle stock across different branches.

5. Smart Stock Alerts – AI-based predictions for low-stock before it happens.

6. Supplier Integration – Directly place orders with suppliers when stock is low.

7. Role-Based Access – Different permissions for admin, staff, and manager.

8. Voice Commands – Add or update stock using simple voice input.

9. Offline Mode – Work without internet, sync later when online.

10. Advanced Analytics – Charts, graphs, and sales trends for better decision-making.

**CONCLUSION:**

The store manager inventory system helps keep track of products easily, saves time, and makes store management better.

*THANK YOU*