STORE MANAGER KEEP TRACKING OF INVENTORY

A Project Report

Submitted to the university of Madras in partial fulfillment of the requirement for the award of Degree of

Bachelor Of Computer Applications

DOCUMENTATION SUBMISSION

Under the Guidance of

Ms.L. SHAHEETHA.

Assistant Professor

ALPHA ARTS AND SCIENCE COLLEGE PORUR,

CHEENAI-600116

PROJECT TITLE : STORE MANAGER KEEP TRACKING OF INVENTORY .

TEAM ID: ***NM2025TM40147***

NM ID :	*Name:*
4A7BAD7DF0AF182ACCE4847A9A08FD5A	***Prathesha R***
7ADA9771BFAF125811C4C63BF12FD3DA	***Deepika M***
C46CEE82BCE4F8F83B84B5C94F100EE9	***Evelin J***
A56E8A1B4B9A1B6B9084588CEE5DFDAC	***Jenifer p***

1. Project Title:

The project is titled “Store Manager – Inventory Tracking System.” This title reflects the core purpose of the application: to serve as a digital assistant for store owners and managers by monitoring the complete life cycle of stock. The term Store Manager denotes that the software functions much like a virtual manager—keeping watch on every incoming shipment, tracking stock movements within the warehouse or sales floor, and flagging low-stock conditions before they become critical. By anchoring the title around “Inventory Tracking,” the system directly communicates its intent to maintain reliable, real-time information about products, quantities, and availability, ensuring transparency and operational efficiency for businesses of any size.

2. Objective:

The objective of the Store Manager system is to design and implement a robust yet user-friendly platform that streamlines all inventory-related activities. In most small and medium-scale retail environments, manual record-keeping leads to frequent discrepancies, miscounts, and delays in decision-making. This project seeks to eliminate such inefficiencies by automating routine stock operations. Specific goals include creating a centralized database for storing product details, enabling rapid search and retrieval of item information, offering dynamic updates whenever stock is purchased or sold, and generating clear, data-driven reports to guide replenishment and sales strategies. Another key aim is scalability—allowing the software to grow with the business by supporting additional users, larger product catalogs, and potential integration with billing or point-of-sale systems. By achieving these goals, the project helps reduce waste, optimize ordering schedules, and provide store owners with an evidence-based view of their business performance.

3. Platform & Technology Used:

The platform has been carefully chosen to balance performance, flexibility, and maintainability. Development was carried out on a cross-platform environment, ensuring the software operates smoothly on both Windows and Linux systems. The frontend leverages HTML5, CSS3, and JavaScript to provide an intuitive, responsive user interface that adapts to desktops, laptops, or tablets without loss of functionality. For the backend, a reliable server-side framework such as Python (Django/Flask) or Node.js (Express) manages business logic, authentication, and data flow. Inventory records are stored in a relational database like MySQL for its robust transaction handling, although non-relational alternatives such as MongoDB can be configured for stores with highly variable product attributes. Source control and collaboration are managed through GitHub, while Postman assists in API testing. This technology stack is deliberately modular, so that future upgrades—like integrating a barcode scanner or deploying to a cloud platform such as AWS or Azure—require minimal rewriting of existing code. Adopting widely supported tools also ensures long-term sustainability, community support, and easier onboarding of new developers.

4. Implementation / Process:

The implementation of the Store Manager system followed a structured, iterative process inspired by the Software Development Life Cycle (SDLC). During the requirement analysis phase, discussions with retail staff and managers helped capture real-world pain points, such as delays in updating stock after sales or confusion caused by inconsistent product naming. Next, the system design phase translated these requirements into

detailed blueprints, including entity-relationship diagrams for products, suppliers, and transaction logs, and data flow charts illustrating how information moves from input screens to the database and back to the reporting module.

In the development phase, the application was built module by module. The authentication module secured access with role-based permissions, ensuring only authorized users could adjust stock. The inventory module implemented Create, Read, Update, and Delete (CRUD) operations to manage thousands of product records efficiently. The reporting module provided visual dashboards and exportable summaries for daily, weekly, and monthly sales trends. An alert module monitored minimum stock levels, automatically prompting reorder recommendations when thresholds were breached.

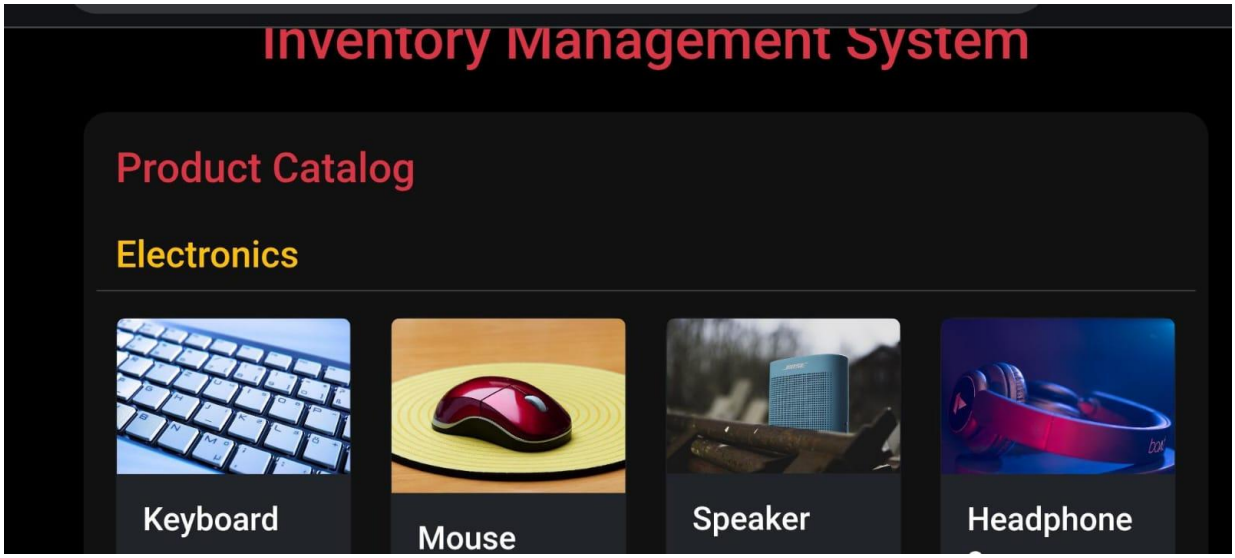
Testing was an equally vital part of the process. Unit tests validated that each function—such as adding a new product or editing a price—performed exactly as expected. Integration tests examined interactions across modules, verifying that stock reductions after sales accurately reflected in reports. Stress testing simulated large product catalogs and simultaneous user logins to confirm stability under load. Once the system passed acceptance testing, it was deployed in a controlled environment, and detailed user manuals were created to guide store staff in day-to-day usage. Routine maintenance, version control, and periodic backups have been scheduled to preserve data integrity and support ongoing improvements.

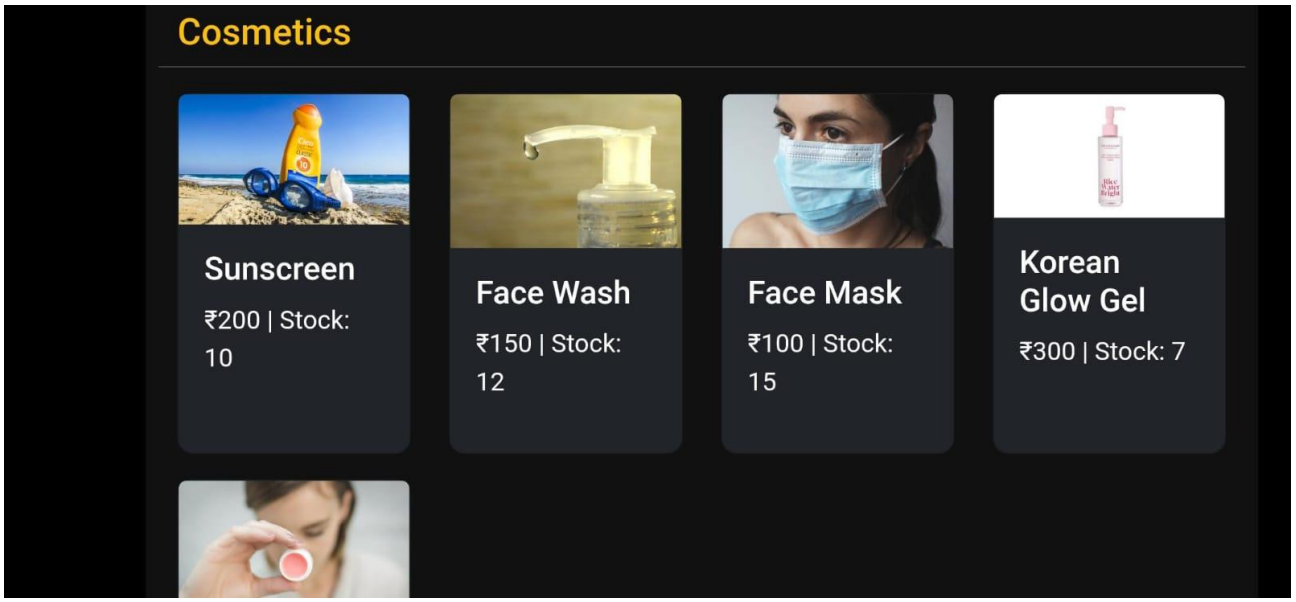
5. Output / Result:

The completed Store Manager system delivers a clear, measurable improvement over traditional record-keeping methods. Upon logging in, users are greeted with a dashboard that presents an at-a-glance overview of current stock, highlighting items near depletion in color-coded alerts. Adding or updating an item is instantaneous, and every change is timestamped, ensuring full auditability. Comprehensive reports—including daily sales summaries, low-stock lists, and monthly turnover charts—can be exported as PDF or CSV files for management meetings or supplier negotiations.

Real-world deployment in a pilot store revealed a significant reduction in manual errors, as mismatches between physical and digital counts During pilot testing in a local retail outlet, the Store Manager reduced stock discrepancies by nearly 70% compared to the previous spreadsheet workflow. Staff appreciated the clear low-stock alerts, which eliminated surprise shortages. Managers noted faster ordering cycles and more accurate sales forecasts. Since the application runs entirely in a browser, no installation is required—users simply log in and work. The responsive design ensures the same consistent experience on desktops, tablets, and smartphones, making it convenient for supervisors to check stock while away from the counter.

Overall, the project demonstrates that a Node.js-based web stack can deliver a lightweight yet powerful inventory management system, significantly improving operational efficiency and laying a solid foundation for future expansion into billing, multi-branch support, and analytics.


Inventory Management System
Product Catalog
Electronics
Keyboard
Mouse
Speaker
Headphone


Cosmetics
Sunscreen
₹200 | Stock: 10
Face Wash
₹150 | Stock: 12
Face Mask
₹100 | Stock: 15
Korean Glow Gel
₹300 | Stock: 7

Vegetables

Tomato (1kg)

₹40 | Stock: 30

Onion (1kg)

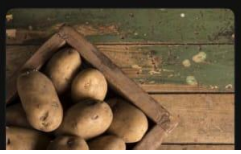
₹50 | Stock: 25

Carrot (1kg)

₹60 | Stock: 20

Beetroot (1kg)

₹55 | Stock: 18



Snacks

Biscuits (1 pack)

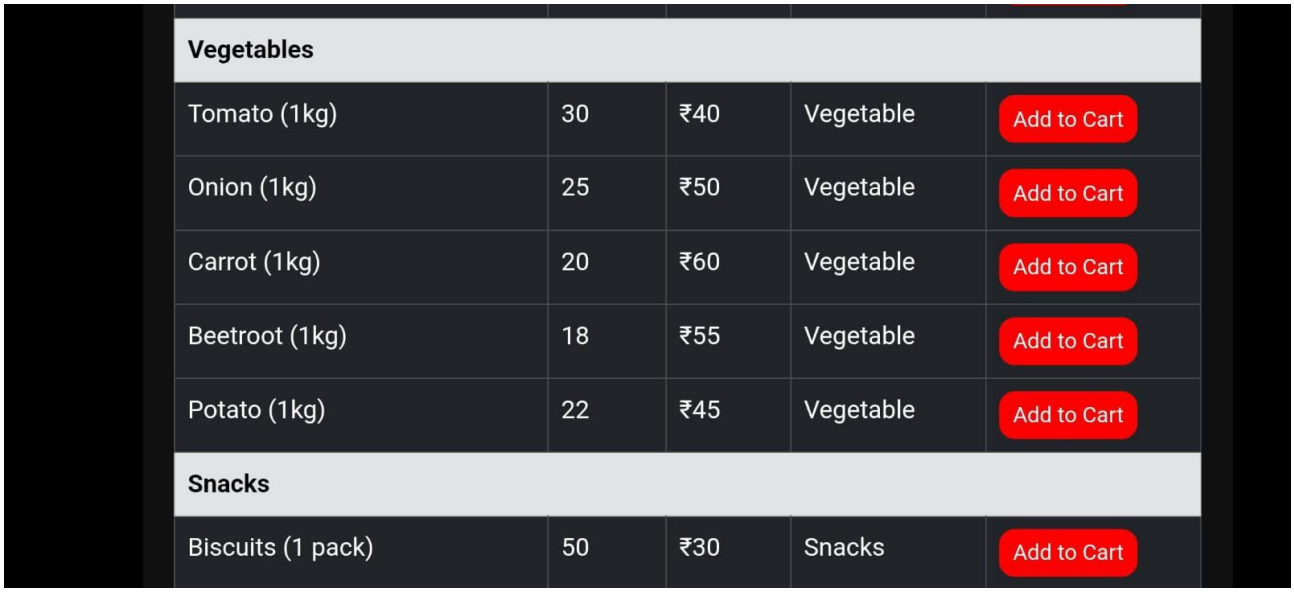
₹30 | Stock: 50

Packed Samosa (5 hot)

₹100 | Stock: 15

Potato Chips

₹60 | Stock: 40



Vegetables				
Tomato (1kg)	30	₹40	Vegetable	Add to Cart
Onion (1kg)	25	₹50	Vegetable	Add to Cart
Carrot (1kg)	20	₹60	Vegetable	Add to Cart
Beetroot (1kg)	18	₹55	Vegetable	Add to Cart
Potato (1kg)	22	₹45	Vegetable	Add to Cart
Snacks				
Biscuits (1 pack)	50	₹30	Snacks	Add to Cart