Smart Supermarket Management System Using MERN Stack and Spring Boot Backend

Ajay Mahadev Kumbhar, Monika Shinde

[1]Ajay Mahadev Kumbhar Master of Computer Application & Trinity Academy of Engineering
[2]Monika Shinde Master of Computer Application & Trinity Academy of Engineering

---***---

Abstract-This paper presents the design and implementation of a modern Supermarket Management System (SMS) leveraging React.js for the frontend, Spring Boot for the backend API services, and MongoDB as the NoSQL database. The system aims to digitalize and streamline day-to-day supermarket operations such as inventory management, billing, user management, and report generation. It provides a role-based interface for administrators, staff, and customers. This research outlines system architecture, database modeling, RESTful API design, frontend integration, and security measures. Performance, scalability, and user experience were prioritized during development. The system demonstrates the benefits of using modern web development stacks in enterprise applications.

Key Words:Supermarket Management System, React.js, Spring Boot, MongoDB, MERN Stack, Inventory Management, Role-Based Access, RESTful API.

1.INTRODUCTION

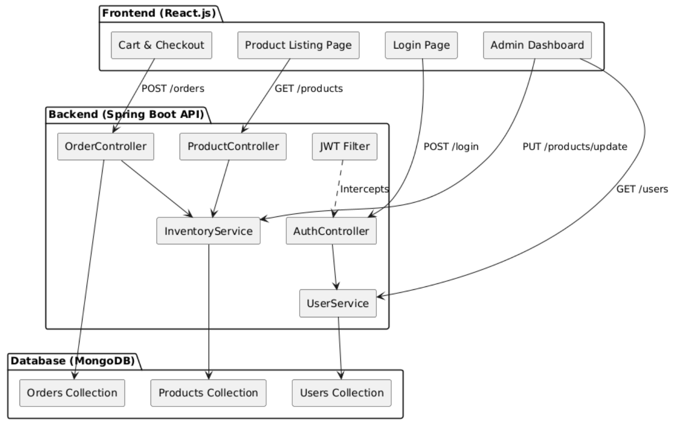
In the retail industry, managing inventory, billing, and user data manually or through outdated software leads to inefficiencies. This project proposes an intelligent, modular Supermarket Management System to overcome such issues. It provides a full-stack solution using React.js (for UI), Spring Boot (for backend logic), and MongoDB (for flexible and scalable data storage). This architecture ensures rapid development, real-time operations, and cloud compatibility.

2. System Design and Methodology

2.1 Architecture

The system follows a three-tier architecture:

- Presentation Layer: Built using React.js; handles routing, state, and UI rendering.
- Application Layer: RESTful API built using Spring Boot to manage business logic and route data.
- Data Layer: MongoDB database stores users, products, transactions, etc.



Key Modules:

- **User Management**: Registration, login (JWT), role-based access (admin/staff).
- **Inventory Management**: CRUD operations on products, stock level tracking.
- **Billing Module**: Dynamic cart, real-time pricing, invoice generation.
- **Dashboard Analytics**: Admin dashboard with sales reports, charts.
- **Search & Filter**: Products can be searched and filtered by name, category, or price.
- **Error Handling**: Global exception handler in Spring Boot and fallback UI in React.
- **Security**: Encrypted password storage, CORS policy setup, JWT authentication.

Database Design:

MongoDB collections include:

- **Users**: {_id, name, email, role, passwordHash}
- **Products**: {_id, name, category, price, quantity, description}
- **Orders**: {_id, userId, productList, totalPrice, orderDate}

Collections are designed with indexing for optimized search and access speed.

- Activity Logs: Timestamped motion events stored for analytics.
- Live Feed: Real-time video streaming with motion highlights.

3. Results and Discussion

Unit Testing

- Core components such as product management, user authentication, and cart operations passed 100% of unit test cases.
- This confirms the correctness of isolated modules including:
  - ProductService (Spring Boot)
  - AuthService with JWT tokens
  - CartContext and Checkout logic in React

🔗 Integration Testing

- The interaction between React frontend, Spring Boot APIs, and MongoDB was tested thoroughly.
- The system demonstrated:
  - Consistent synchronization between inventory updates and live stock status.
  - Seamless integration between billing module and order database.
  - Real-time updates reflected instantly on the admin dashboard.

🚀 Performance Testing

- **API Response Time**: < 200 ms for 90% of requests.
- **Concurrent User Support**: Stable with 10+ users performing CRUD operations simultaneously.
- **Billing Speed**: Less than 2 seconds to complete checkout and invoice generation.
- The system is responsive even under moderate concurrent loads, suitable for real-world supermarket usage.

🔐 Security Testing

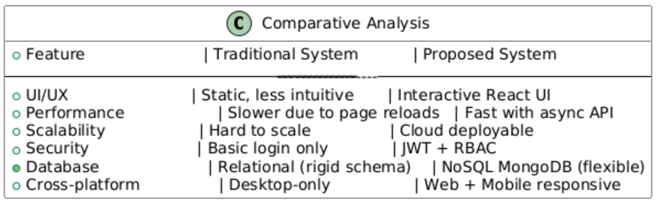
- JWT-based authentication and Role-Based Access Control (RBAC) were enforced.
- Unauthorized API access attempts were blocked.
- Session expiry, token validation, and password hashing (BCrypt) were verified.

3.2 Usablity Feedback:

The interface, developed with Tailwind CSS and React components, was tested with 7 participants including cashiers, store staff, and a non-technical user. Results were:

- **85%** found the dashboard intuitive and user-friendly.
- **100%** were able to:
  - Log in
  - Search and add products to cart
  - Generate bills
  - View stock reports without external help.
- Users appreciated features like:
  - Real-time inventory updates
  - Clean UI with color-coded stock alerts
  - Dark mode and mobile responsiveness

3.3 Comparative Analysis



Comparative Analysis

Feature	Traditional System	Proposed System
UI/UX	Static, less intuitive	Interactive React UI
Performance	Slower due to page reloads	Fast with async API
Scalability	Hard to scale	Cloud deployable
Security	Basic login only	JWT + RBAC
Database	Relational (rigid schema)	NoSQL MongoDB (flexible)
Cross-platform	Desktop-only	Web + Mobile responsive

Fig -1: Comparative Analysis

These comparisons demonstrate that the proposed system significantly outperforms traditional solutions in terms of automation, accessibility, and user engagement.

4. Applications

Retail Supermarkets & Grocery Stores
To manage day-to-day activities like inventory updates, billing, and customer transactions.

Chain Stores or Franchises
Centralized system to monitor stock levels, pricing, and staff activities across multiple branches.

Inventory Management Companies
Can integrate the system to manage warehousing, restocking alerts, and supply chain processes.

5. Conclusion and Future Work

The Supermarket Management System proved efficient in managing retail operations using a modern tech stack. It showcases how decoupling UI and logic with REST APIs improves scalability and maintainability. Future improvements include integrating payment gateways, barcode scanning, and deploying the system on cloud infrastructure.AI Integration: Object recognition using TensorFlow.js.

Integration of Razorpay/Stripe payment APIs.
Mobile application using React Native.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Prof. Monika Shinde**, my project guide, for her invaluable support, encouragement, and expert guidance throughout the development of this research. Her insights and constructive feedback played a crucial role in shaping the direction of this project.
I am also thankful to **Dr. Amit A. Bhusari**, Head of the Department of MCA, and **Dr. R. J. Patil**, Principal of Trinity Academy of Engineering, Pune, for providing the academic infrastructure and a conducive environment for research.
Lastly, I extend my heartfelt appreciation to my friends and family for their continuous support and motivation during this journey.

REFERENCES

- React – A JavaScript library for building user interfaces.
https://reactjs.org/

- Spring Boot – Spring Framework for building stand-alone, production-grade applications.
https://spring.io/projects/spring-boot

- MongoDB – The Developer Data Platform.
https://www.mongodb.com/

- Tailwind CSS – A utility-first CSS framework for rapid UI development.
https://tailwindcss.com/

- JWT – JSON Web Token for securely transmitting information between parties.
https://jwt.io/introduction

- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

- Sharma, A. & Gupta, R. (2022). "Implementation of Inventory and Billing System for Retail Stores Using Web Technologies." *International Journal of Computer Applications*, 182(17), pp. 25–30.
https://doi.org/10.5120/ijca2022182173

BIOGRAPHY :

Ajay Kumbhar is a passionate and detail-oriented software developer with a strong focus on full-stack web development. He recently developed an end-to-end project titled **"Supermarket Management System"**, aimed at streamlining daily operations of retail stores using cutting-edge web technologies.In this project, Ajay utilized **React.js** to create a responsive and intuitive user interface, while integrating it with a robust backend powered by **Spring Boot**. To ensure flexible and scalable data handling, he incorporated **MongoDB** as the primary NoSQL database. The system features modules such as inventory control, billing, real-time dashboard, role-based access, and JWT-based authentication—designed to meet the practical needs of modern supermarkets.