Smart Inventory Management System

PROJECT REPORT

Submitted by

Shivam Kumar Tiwari (23BCS12755)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

TABLE OF CONTENTS

- Future Enhancements
- Conclusion
- Reference

ABSTRACT

The Smart Inventory Management System is a web-based application designed to simplify and automate the management of products, stock, and users in any business or warehouse environment. The system provides a centralized platform where administrators can add, update, and monitor products in real time while keeping track of low-stock alerts and sales trends.

Built using Spring Boot (Java) for the backend and React.js with Tailwind CSS for the frontend, this system ensures seamless communication between the client and server using RESTful APIs and MySQL as the relational database. It is scalable, user-friendly, and supports role-based access control for secure usage.CHAPTER

1.INTRODUCTION

Inventory management plays a vital role in the success of any organization. Manual tracking using spreadsheets or registers often leads to errors, stock mismanagement, and time inefficiency. To address these challenges, this project implements a modern, digital, and smart inventory management solution.

The system helps businesses to:

- Maintain accurate stock records
- Avoid product shortages or overstocking
- Improve operational efficiency
- Gain insights through data-driven analytics

3. Objectives

- To design a **full-stack web application** that allows easy tracking and updating of product details.
- To maintain a **centralized database** for products and stock.
- To provide **real-time updates** and a **user-friendly dashboard**.
- To include **low stock alerts** and **role-based access** (Admin/User).
- To ensure scalability, modularity, and data security.

---

4. Existing System & Problems

In traditional systems:

- Inventory is managed manually or through static spreadsheets.
- Tracking product stock, price updates, and low-stock alerts is cumbersome.
- Data consistency and multi-user access are limited.
- Time and labor costs are high due to repetitive manual operations.

---

5. Proposed System

The proposed Smart Inventory Management System automates the entire process.
It:

- Provides a **web-based dashboard** accessible from any device.
- Allows **admins** to manage products, stock, and users efficiently.
- Displays **real-time product quantity** and notifies when stock is low.
- Supports **CRUD operations** (Create, Read, Update, Delete) with validation.
- Uses a **modern UI** for easy navigation and dark mode support.
- Ensures data consistency through a **Spring Boot + MySQL backend**.

---

6. System Architecture

The system follows a **three-tier architecture**:

1. **Presentation Layer (Frontend):** Built using React.js, handles user interface and interactions.
2. **Application Layer (Backend):** Implemented using Spring Boot REST APIs for business logic.
3. **Data Layer (Database):** MySQL database stores all persistent data (products, users, etc.).

Data Flow:
User → React Frontend → Axios → Spring Boot REST API → MySQL Database → Response to Frontend

---

7. System Design

7.1 Use Case Diagram

Actors: Admin, User
Use Cases: Login, Add Product, Update Product, Delete Product, View Products, Check Low Stock, Manage Users

---

7.3 Database Schema

Table Name	Column Name	Data Type	Description
products	id	INT (PK, AUTO_INCREMENT)	Unique product ID
	name	VARCHAR(255)	Product name
	price	DECIMAL(10,2)	Product price
	quantity	INT	Current stock
users	id	INT (PK)	User ID
	username	VARCHAR(255)	Login username
	password	VARCHAR(255)	Encrypted password
	role	ENUM('ADMIN','USER')	User role

7.4 Flow of Control

1. User logs in → verified via backend API.
2. Dashboard displays all products.
3. Admin can add, update, or delete products.
4. Low stock products automatically appear in the “Low Stock” section.
5. Frontend state updates dynamically using React hooks and Axios API calls.

---

8. Module Description

8.1 Authentication Module

Handles login and role-based access.
Admins get access to product, user, and settings modules, while users can view

product details.

8.2 Product Management Module

Provides CRUD operations for products. Data validation ensures no empty or invalid entries.

8.3 Inventory Tracking Module

Displays current stock levels and product details in tabular form with sorting, pagination, and search filters.

8.4 Low Stock Alert Module

Automatically filters products below a threshold quantity and lists them separately for quick restocking.

8.5 User Management Module

Allows admin to view or remove users (future enhancement: add/edit users).

8.6 Settings & Theme Module

Provides light/dark mode toggling and preference persistence using local storage.

---

9. Technology Stack

Layer	Technology Used
Frontend	React.js, Tailwind CSS, Axios
Backend	Spring Boot (Java 22), Maven
Database	MySQL
Tools	IntelliJ IDEA, VS Code, Postman
Version Control	Git & GitHub
APIs	RESTful APIs
Build Tools	Maven for backend, Vite for frontend

---

10. Backend Implementation (Spring Boot)

- **Controllers:** Handle REST endpoints like /api/products.
- **Repositories:** Use JpaRepository for database operations.
- **Entities:** Product, User.
- **Configuration:** CORS enabled, connected to MySQL through application.properties.
- **Endpoints:**
  - GET /api/products
  - POST /api/products
  - PUT /api/products/{id}
  - DELETE /api/products/{id}
  - GET /api/products/page
  - GET /api/products/search

---

11. Frontend Implementation (React + Tailwind CSS)

- **Main Components:**
  - DashboardLayout.jsx → manages dark mode and sidebar.
  - Sidebar.jsx → navigation links (Admin/User).
  - Products.jsx → lists all products with sorting, pagination, and edit/delete buttons.
  - AddProduct.jsx → form to add new product.
  - UpdateProduct.jsx → edit product details (two-column layout with validation).
  - LowStock.jsx → low-stock list.
  - Settings.jsx → theme management.

---

12. Integration & API Endpoints

Axios is used to call backend APIs from React.
Each CRUD operation communicates via JSON.

Example API Call:

```
axios.get("http://localhost:8080/api/products")
    .then(res => setProducts(res.data))
    .catch(err => console.error(err));
```

---

13. Testing & Validation

- **Unit Testing:** Done for backend service layers using JUnit.
- **Integration Testing:** Verified API calls via Postman.
- **Frontend Testing:** Manual verification of all user flows.
- **Validation:** Client-side form checks for empty fields or invalid prices.

---

14. Screenshots (Placeholders)

1. Login Page
2. Dashboard View
3. Product Table
4. Add Product Form
5. Update Product Form
6. Low Stock Page
7. Dark Mode Dashboard

---

15. Advantages of the System

- Centralized and accurate stock management.
- User-friendly modern UI.
- Scalable and responsive web app.
- Easy data manipulation with search/sort/filter options.
- Reduces manual effort and human error.

---

16. Challenges Faced

- Synchronizing frontend and backend data flow.
- Handling asynchronous API calls.
- Managing CORS during local development.

- Mapping stock (quantity) between backend and frontend correctly.

---

17. Future Enhancements

- Implement **JWT-based authentication** and registration.
- Add **supplier and order management**.
- Enable **data analytics dashboard** with charts.
- Add **email/SMS notifications** for low-stock alerts.
- Support **barcode scanning** integration.
- Cloud deployment via AWS or Azure.

---

18. Conclusion

The Smart Inventory Management System successfully demonstrates how modern web technologies can transform traditional inventory handling into an efficient, automated process. By integrating a powerful backend (Spring Boot) with an interactive frontend (React + Tailwind), this project achieves real-time product management, improved user accessibility, and a scalable architecture for future growth.

---

19. References

1. Spring Boot Documentation – https://spring.io/projects/spring-boot
2. React Official Docs – https://react.dev
3. MySQL Documentation – https://dev.mysql.com/doc/
4. Tailwind CSS Docs – https://tailwindcss.com/docs
5. Axios GitHub Repository – https://github.com/axios/axios