Inventra – Intelligent Inventory Management System

Team 2

Christina J
Krishna Moorthy Anbalagan
Rajeswari R
Shrisha Singha
Ganji Sudharsan Balaji

INTRODUCTION

1. Inventory management is a critical function for organizations dealing with products and stock.

2. Manual or poorly designed inventory systems often lead to issues such as duplicate products, incorrect stock levels, and lack of transaction tracking.

3. With multiple users accessing the system simultaneously, problems like data inconsistency and stock mismatch can occur.

4. Inventra – Intelligent Inventory Management System is designed to solve these challenges by providing a secure, reliable, and automated inventory solution.

5. The system uses modern web technologies and Java-based backend frameworks to ensure data consistency, security, and scalability.

PROBLEM STATEMENT

1. Many organizations rely on **manual or semi-automated inventory systems**, which are prone to human errors.

2. The same product may be stored multiple times with different names or identifiers, leading to **duplicate inventory records**.

3. Lack of proper **transaction tracking** results in incorrect stock levels and poor accountability.

4. Concurrent access by multiple users can cause **data inconsistency and negative stock values**.

5. Traditional systems often lack **security, role-based access control, and real-time alerts**, increasing operational risk.

PROPOSED SYSTEM

- **Modules of Inventra:**

- **Authentication**
  - Login, token generation, role validation
- **Product / Inventory Management**
  - Add, update, delete items
  - Track stock levels
  - Store supplier details, purchase history
- **Inventory Alerts**
  - Low stock alerts
  - Expiry-near alerts
  - Fast and slow-moving items detection

- **Transaction Management**
  - Record stock-in and stock-out operations
  - Maintain complete transaction history
  - Ensure data consistency and auditing

- **Reports & Analytics**
  - Charts and insights
  - Predictions
  - Incoming & outgoing stock summary

TECHNOLOGY STACK

Development Tool

- IntelliJ IDEA

Frontend

- HTML
- CSS
- JavaScript

Backend

- Java
- Spring Boot
- REST API

Security

- JWT
- BCrypt
- Role-Based Access Control (RBAC)

Database

- MySQL

SYSTEM ARCHITECTURE

- The system follows a **Controller – Service – Repository** architecture.
  Controller Layer
- Handles client (UI) requests
- Exposes REST APIs
  Service Layer
- Contains business logic
- Manages transactions and validations
  Repository Layer
- Handles database operations
- Interacts with MySQL using JPA
- Ensures **clear separation of responsibilities**, **maintainability**, and **scalability**

Module 1: Authentication Module

Module Description
This module ensures only authorized users can access the system,. It manages login, logout, roles, and security tokens.

Classes

- **User:** Stores user details in the DB
- **AuthController:** Handles API endpoints (/login, /register)
- **AuthService:** Authentication logic & error handling
- **JwtProvider:** Creates secure tokens
- **UserRepository:** Interacts with database for CRUD operations

Workflow of Module 1 – Authentication

1. User/Admin submits credentials
2. AuthController receives request
3. AuthService validates and encrypts password
4. User data stored in database
5. User logs in with credentials
6. System verifies user
7. JWT token is generated
8. Token validated for every request
9. Role-based access enforced

Database Schema (Authentication Module)

Field Name	Data Type	Description
user_id	BIGINT (Primary Key)	Unique user identifier
username	VARCHAR(100)	Unique username
email	VARCHAR(150)	User email address
password	VARCHAR(255)	Encrypted password (BCrypt)
role	VARCHAR(50)	Admin / Manager / Staff
is_active	BOOLEAN	Account status

MODULE 2 – PRODUCT / INVENTORY MANAGEMENT

- **MODULE 2 – PURPOSE**

1. **Store product details in the system**
2. **Maintain accurate inventory records**
3. **Handle stock received in multiple shipments**
4. **Display current available stock at any time**

Classes Involved

- Product
- Category
- Supplier
- Inventory
- ProductController
- ProductService
- ProductRepository
- TransactionService

Inventory / Product Management – Workflow

1. User logs into the system with valid credentials.
2. User sends a request to add a new product or update stock.
3. Request reaches the **ProductController**.
4. Controller forwards the request to **ProductService**.
5. ProductService checks for duplicate product (using SKU).
6. Stock is increased or decreased based on the operation.
7. Product data is saved using **ProductRepository**.
8. Transaction details are logged in **TransactionService**.
9. Low stock is checked and alert is triggered if required.
10. Updated inventory status is sent back to the user.

Database Schema (Inventory / Product Management)

Field Name	Data Type	Description
product_id	BIGINT (Primary Key)	Unique product identifier
sku	VARCHAR(100)	Unique product code
name	VARCHAR(150)	Product name
category_id	BIGINT (Foreign Key)	Category reference
supplier_id	BIGINT (Foreign Key)	Supplier reference
unit_price	DECIMAL	Price per unit
stock_quantity	INT	Avaliable stock
min_stock_level	INT	Minimum stock threshold

MODULE 3 – ALERTS & MONITORING

- **MODULE 3 – PURPOSE**

1. **Monitor system conditions continuously**
2. **Detect warning and critical situations**
3. **Generate alerts to notify users**
4. **Support timely decision-making**

Classes Involved

- **Alert**
- **AlertService**
- **NotificationService**

Inventory Alerts– Workflow

1. Stock quantity is updated in the inventory module.
2. The **ProductService** calls the **AlertService** after stock change.
3. AlertService compares current stock with minimum stock level.
4. If stock is below the minimum level, an alert is generated.
5. Alert information is sent to the **NotificationService**.
6. NotificationService displays alert on dashboard or sends email/SMS.
7. User is notified about low stock condition.

Database Schema (Inventory Alerts)

Field Name	Data Type	Description
alert_id	BIGINT (Primary Key)	Unique alert identifier
product_id	BIGINT (Foreign Key)	Related product
current_stock	INT	Current stock quantity
min_stock_level	INT	Minimum allowed stock
alert_message	VARCHAR(255)	Alert description
alert_status	VARCHAR(50)	ACTIVE / RESOLVED
created_at	TIMESTAMP	Alert creation time

MODULE 4 – Transaction Management

MODULE 4 – PURPOSE

- To maintain a **complete record of all inventory transactions**
- To track **stock-in and stock-out activities**
- To ensure **data consistency and accountability**
- To support **audit and verification** of inventory operations
- To help in **report generation and analysis**

Classes Involved

- **Transaction**
- **TransactionService**
- **TransactionRepository**

Transaction Management– Workflow

1. User performs a stock-in or stock-out operation.
2. Request is processed by **ProductService**.
3. After stock update, **TransactionService** is invoked.
4. TransactionService creates a new transaction record.
5. Transaction details (type, quantity, user, date) are stored.
6. Data is saved using **TransactionRepository**.
7. Transaction history is available for reports and audits.

Database Schema (Transaction Management)

Field Name	Data Type	Description
transaction_id	BIGINT (Primary Key)	Unique transaction ID
product_id	BIGINT (Foreign Key)	Related product
transaction_type	VARCHAR(50)	STOCK_IN / STOCK_OUT
quantity	INT	Quantity changed
transaction_date	TIMESTAMP	Date and time of transaction
performed_by	BIGINT (Foreign Key)	User who performed action
performed_by	TIMESTAMP	User who performed action

Key Features & Advantages

1. **Secure authentication with role-based access control**
2. **Accurate and real-time inventory management**
3. **Automatic low-stock alert generation**
4. **Complete transaction history for auditing**
5. **Improved data consistency and reliability**
6. **Reduced manual errors and better decision making**

Conclusion

1. Inventra provides a **secure and efficient inventory management system**.
2. The system ensures **accurate stock tracking, automated alerts, and complete transaction history**.
3. Role-based authentication improves **system security and access control**.
4. The modular architecture makes the system **scalable and easy to maintain**.

THANK YOU