Multi-Level Inventory & SCM System: Project Report

Built by Adarsh Milan

November 3, 2025

Abstract

The Multi-Level Inventory and Supply Chain Management (SCM) System is a web-based application designed to manage stock levels across multiple warehouses, automate the procurement (Purchase Order) lifecycle, and facilitate critical internal logistics like stock transfers. The system is built on a robust, role-based architecture using the Model-View-Controller (MVC) pattern to ensure data consistency and segregation of duties.

1 Technical Architecture and Stack

The system is developed using the classic Java web application model, emphasizing clean separation between data processing and presentation layers.

- **Architecture: MVC (Model-View-Controller)**.
- **Frontend/View: JSP (JavaServer Pages) and JSTL** for dynamic rendering and loop processing. A single layout container is used to provide a consistent SPA-like feel (full-page navigation).
- **Controller: Java Servlets** handle routing, validation, and action processing (`POServlet`, `InventoryServlet`, `TransferServlet`).
- **Data Layer: DAOs (Data Access Objects)** manage all SQL and execute business-critical transactions.
- **Database: MySQL/MariaDB** with strict Foreign Key constraints to enforce data integrity.

2 Core Architectural Principles

2.1 Data Management and Scopes

- **Session Scope Reliance:** User identity (`user`, `role`, `warehouseId`) is stored in the Session Scope upon successful login for global access and authorization checking.
- **Flash Messages:** Implemented via temporary Session storage, ensuring success or error messages display once after a `POST` operation and redirect, preventing URL clutter.
- **PRG Pattern:** The **Post/Redirect/Get** pattern is strictly followed for all data modification actions (`doPost`) to prevent form resubmission errors.

2.2 Database Transaction Discipline

Critical multi-step operations are executed as atomic database transactions (using `conn.setAutoCommit(f` and `conn.commit()/conn.rollback()`). This guarantees that complex updates—such as moving stock—either fully complete or entirely fail, preserving data integrity.

3 Key System Workflows

3.1 A. Procurement Lifecycle (Purchase Orders)

The system automates the procurement of goods from low stock alerts to final inventory entry.

1. **Create Request:** Initiated by the Warehouse Manager when stock falls below the reorder level. Inserts a PO into `PurchaseOrders` with status **Pending**.
2. **Admin Approval:** Admin views Pending requests and updates status to **Approved**.
3. **Supplier Confirmation:** Supplier logs in, views Approved orders, and confirms shipment. Status updates to **Sent**.
4. **Goods Receipt (Inventory Update):** Warehouse Manager confirms delivery. The system performs a **critical transaction** to read PO item quantity, **increase `Inventory` stock**, and set PO status to **Delivered**.

3.2 B. Internal Stock Transfer

This manages the movement of stock between different company warehouses, typically for supply mitigation.

1. **Request Transfer:** Source Manager submits a request to move stock to a destination warehouse. Status: **Requested**.
2. **Approval & Debit:** Source Manager approves and ships the request. This performs a **critical transaction** to atomically **decrement** source inventory and **increment** destination inventory. Status updates to **In Transit**.
3. **Final Receipt:** Destination Manager marks the transfer as received. Status updates to **Received**.

4 Role-Based Functionality Summary

- **Admin:** Manages system configuration (CRUD placeholder) and is the final approval authority for Pending Purchase Orders.
- **Warehouse Manager:** Responsible for initiating PO requests, managing local inventory status (low stock alerts), initiating stock transfers, and finalizing Goods Receipt.
- **Supplier:** Restricted access to view Approved Purchase Orders addressed to their `supplier_id` and update the status to Sent or Rejected.