Project: Warehouse Inventory Optimization System

1. Introduction

This document outlines the Low-Level Design (LLD) for a **Warehouse Inventory Optimization System**, which streamlines the management of stock levels, categorization, supplier coordination, and real-time tracking for warehouse operations.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

2.1 Stock Management Module

- Enables tracking and updating of current stock levels.
- Includes functionality for restocking and removal of items.

2.2 Warehouse Zones Management Module

- Organizes inventory into logical storage zones for efficient access.

2.3 Vendor Collaboration Module

- Facilitates the management of vendors supplying goods to the warehouse.

2.4 Performance Metrics Module

- Tracks warehouse KPIs, such as inventory turnover and space utilization.

2.5 Transaction Logging Module

- Records all stock movements, including arrivals and dispatches.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React.
- **Backend**: REST API-based architecture.
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server).

3.2 Component Interaction

- Frontend interacts with the backend through REST APIs for all operations.
- Backend connects to the relational database to handle data storage and retrieval.

4. Module-Wise Design

4.1 Stock Management Module

4.1.1 Features

- Track current stock levels for individual items.
- Enable automated notifications for low-stock alerts.

4.1.2 Data Flow

- User updates or queries stock levels via the frontend.
- Backend updates the database and sends feedback to the user interface.

4.1.3 Entities

- **StockItem**
    - ItemID
    - Name
    - CategoryID
    - Quantity
    - LocationZone

4.2 Warehouse Zones Management Module

4.2.1 Features

- Divide inventory storage into physical or logical zones.
- Allow visualization of available space per zone.

4.2.2 Entities

- **Zone**
    - ZoneID
    - Name
    - Capacity

4.3 Vendor Collaboration Module

4.3.1 Features

- Manage vendor details, contracts, and contact information.
- Track delivery schedules.

4.3.2 Entities

- **Vendor**
    - VendorID
    - Name
    - ContactDetails
    - GoodsSupplied

4.4 Performance Metrics Module

4.4.1 Features

- Generate reports on stock movement efficiency.
- Provide insights into inventory turnover rates.

4.4.2 Entities

- **Metrics**
    - MetricID
    - Type (e.g., Turnover, Space Utilization)
    - Value

4.5 Transaction Logging Module

4.5.1 Features

- Record incoming and outgoing stock transactions.
- Allow querying of historical transaction logs.

4.5.2 Entities

- **TransactionLog**
    - TransactionID
    - ItemID
    - Quantity
    - Type (Inbound/Outbound)
    - Timestamp

5. Deployment Strategy

5.1 Local Deployment

- Frontend and backend deployed on developer machines for initial testing.

5.2 Testing Environments

- Use containerized setups for staging environments, ensuring consistency with deployment.

6. Database Design

6.1 Tables and Relationships

- **StockItem**: Primary Key: ItemID, foreign key: ZoneID.
- **Zone**: Primary Key: ZoneID.
- **Vendor**: Primary Key: VendorID.
- **TransactionLog**: Primary Key: TransactionID, foreign keys: ItemID.

7. User Interface Design

7.1 Wireframes

- Dashboard: Displays stock levels and alerts.
- Zone Overview: Maps zones and current usage.
- Vendor Management: Lists active vendors and their profiles.

8. Non-Functional Requirements

8.1 Performance

- Capable of handling updates to 1,000 stock items within a minute.

8.2 Usability

- Interface designed for non-technical warehouse staff.

8.3 Security

- Implement access controls to restrict unauthorized users.

8.4 Scalability

- Ability to accommodate additional zones, vendors, and stock items without major redesigns.

9. Assumptions and Constraints

9.1 Assumptions

- The warehouse is digitally accessible with basic internet connectivity.

9.2 Constraints

- Limited to a single warehouse in the initial phase.