**Low-Level Technical Design Document for B-Garage Inventory and Order Management System**

1. **Overview**

The goal of this system is to provide an automated inventory and order management system for B-Garage. The system needs to manage two books, Inventory and Orders, and automate the ordering process based on predefined thresholds. The system will support placing orders for parts from two suppliers (local and international), ensuring proper inventory management and cost savings through optimized order times.

**2. Microservice Architecture**

The system will be divided into three microservices:

Inventory Service: Manages the parts inventory.

Order Service: Manages orders placed with the suppliers.

Scheduler Service: Ensures orders to Supplier-B are placed between 12:00 AM and 1:00 AM for discount benefits.

Each service will communicate over REST APIs. We will use Spring Boot to develop each microservice and H2 in-memory database for persistence in the initial stages. The architecture is designed to scale with a move to persistent storage (such as MySQL) in the future.

**3. Functional Requirements**

Inventory Management:

Add/Modify/Delete parts.

Track available quantity of each part.

Automated checking of parts threshold limits and triggering an order when stock falls below the threshold.

Order Management:

Automatically place orders when inventory is low.

Order to Supplier-B should only be placed between 12:00 AM and 1:00 AM.

Order to Supplier-A can be placed any time.

Auditing:

Keep a record of all parts and order transactions for audit purposes.

**4. Non-Functional Requirements**

Persistence: Use H2 in-memory database initially for development, with the possibility to move to an RDBMS like MySQL.

Modularity: Each service is independent, deployable, and scalable.

Reliability: Orders should never fail due to timing issues or supplier constraints.

Maintainability: Code should be modular, adhering to SOLID principles.

Scalability: Services should be designed to scale easily with traffic, particularly the Inventory Service.

**5. Entity Relationship Diagram**

+---------------------+ +------------------+

| Inventory | | Order |

+---------------------+ +------------------+

| part\_id (PK) |<------------>| order\_id (PK) |

| part\_name | | part\_id (FK) |

| available\_quantity | | supplier |

| threshold\_quantity | | order\_quantity |

| minimum\_order\_qty | | order\_date |

| supplier (enum) | | is\_discounted |

+---------------------+ +------------------+

**6. Class Structure and Design**

**src**

**└── main**

**├── java**

**│ └── com**

**│ └── bg**

**│ ├── inventory**

**│ │ ├── controller**

**│ │ │ └── InventoryController.java**

**│ │ ├── model**

**│ │ │ ├── Part.java**

**│ │ │ └── Supplier.java**

**│ │ ├── repository**

**│ │ │ └── PartRepository.java**

**│ │ └── service**

**│ │ └── InventoryService.java**

**│ └── order**

**│ ├── controller**

**│ │ └── OrderController.java**

**│ ├── model**

**│ │ └── Order.java**

**│ ├── repository**

**│ │ └── OrderRepository.java**

**│ └── service**

**│ └── OrderService.java**

**└── resources**

**└── application.properties**

**7. Endpoints**

**Inventory Service:**

GET /api/inventory/parts: Get all parts in the inventory.

POST /api/inventory/parts: Add a new part.

PUT /api/inventory/parts/{id}/quantity: Update available quantity of a part.

**Order Service:**

POST /api/orders/place: Place an order for a part.

**8. Git repo**

<https://github.com/gayatroct22/BgarageInventory/tree/main>