**1. Introduction**

The system aims to automate the inventory management process and order placement to improve efficiency and service quality.

**2. Architecture Overview**

The system will follow a **microservices architecture** using Java and Spring Boot, enabling scalability and maintainability. The architecture will consist of the following components:

* **Controllers**: A web-based dashboard for garage owners to manage inventory and orders.
* **Inventory Service**: Manages part details, available quantities, and supplier information.
* **Order Service**: Handles the creation and tracking of orders.
* **Database**: A relational database (H2) to persist inventory and order information.
* **CSV Processing**: Functionality for importing part information via CSV files.

**2.1 Component Diagram**

| User Interface | <---> | Inventory Service |

|

|

v

| Order Service |

|

v

| Database |

**3. Components and Responsibilities**

**3.1 Inventory Service**

* **Responsibilities**:
  + Manage parts information, including threshold limits and available quantities.
  + Monitor stock levels and trigger automatic orders based on thresholds.
  + Expose REST APIs for adding, updating, and fetching part details.
* **Key Methods**:
  + List<Part> getAllParts()
  + Part addPart(Part part)
  + void updatePart(Part part)
  + List<Part> parseCSV(MultipartFile file)

**3.2 Order Service**

* **Responsibilities**:
  + Create and manage orders for parts based on inventory levels.
  + Ensure orders to Supplier-B are placed within the specified time window to get discounts.
  + Expose REST APIs for order management.
* **Key Methods**:
  + List<Order> getAllOrders()
  + Order createOrder(Order order)

**3.3 Database**

* **Database Schema**:
  + **Part Table**:
    - id (Long, PK)
    - name (String)
    - available\_qty (int)
    - threshold\_qty (int)
    - supplier (Enum: SUPPLIER\_A, SUPPLIER\_B)
  + **Order Table**:
    - id (Long, PK)
    - order\_time (Timestamp)
    - quantity (int)
    - supplier (Enum: SUPPLIER\_A, SUPPLIER\_B)
    - part\_id (Long, FK to Part)

**4. Data Flow**

1. **Part Addition/Update**:
   * The garage owner uploads a CSV file via the user interface.
   * The Inventory Service processes the CSV, creating or updating Part records in the database.
2. **Inventory Monitoring**:
   * Daily updates to available quantities are made by the garage owner through the user interface.
   * The Inventory Service checks parts against their threshold limits.
   * If a part's quantity falls below its threshold, the Order Service is triggered to create an order.
3. **Order Placement**:
   * Orders to Supplier-B are only placed between **12:00 AM and 01:00 AM** to secure discounts.
   * The Order Service places orders and logs them in the database.

**5. Error Handling**

* Validate CSV format and data integrity during the parsing process.
* Log errors for troubleshooting and provide user-friendly messages.

**6. Technology Stack**

* **Backend**: Java, Spring Boot
* **Database**: H2 (In-memory database for development; production might use PostgreSQL or MySQL)
* **Build Tool**: Maven

**7. Non-Functional Requirements**

* **Performance**: The system should handle at least 1000 parts and order transactions efficiently.
* **Scalability**: Microservices should allow easy scaling as the garage grows.
* **Security**: Basic authentication for the web interface, with role-based access control in future iterations.