**Low Level Design of Automobile Garage Application**

**Requirements:**

1. Inventory and Order should be persisted for audit requirement.

2. There are only two suppliers of parts (Local - named as Supplier-A /International - as Supplier-B)

3. Every part must have

* + Threshold limit (if stock goes below this threshold, then automated order with minimum Qty should be placed)
  + Supplier (Supplier-A or Supplier-B)
  + Available Qty
  + Minimum Order Qty

4. To get the discount benefit for orders to Supplier-B, order should be placed only between 12:00AM to 01:00 AM

5. There is no discount benefit for order to Supplier-A, hence order to Supplier-A can be placed atany time.

6. Provide feature to add new parts, modify existing parts, parts supplier, and available qty.

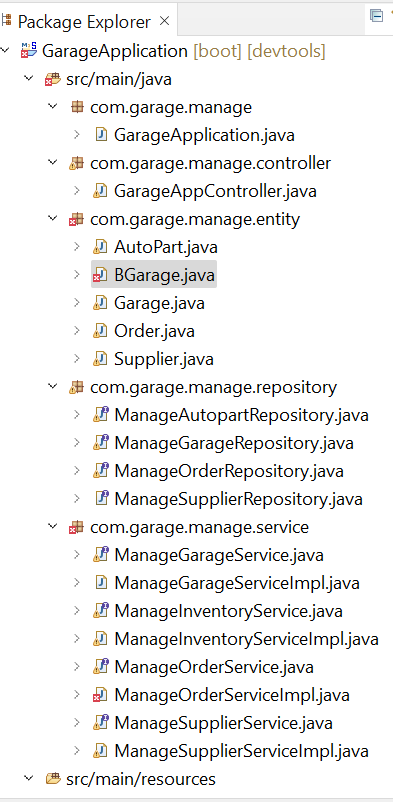
**Architecture**

Frontend- can be developed with Reactjs

Backend- SpringBoot Application

Database- MySQL or Postgres(currently used h2)

**Components**

****

Service Interface and Implementation to process CRUD operations for different entities along with auto generation of orders for Local and International Suppliers

Repositories extending JPA Repository for CRUD operations using default and custom queries

Entity Objects for processing data in db using JPA

Controller exposes the APIs to add/fetch/update/delete autoparts ,suppliers,garage and orders

SpringBoot Main class

**Garage (BGarage)**

Attributes: name, location, startDate, supplierList

Operations:createGarage,updateGarage,fetchGarage,deleteGarage

**Inventory/AutoPart**

Attributes: id,autoPartName,manufacturingCompany,manufacturingDate,price,thresholdQuantity,minOrderQuantity,availableQuantity,supplier,garageName,vehicleType

Operations: List<AutoPart> addAutoParts(List<AutoPart> parts);

List<AutoPart> fetchAutoParts();

AutoPart updateAutoPart(AutoPart parts) **throws** Exception;

**void** deleteAutoPart(AutoPart parts);

**Order**

Attributes: orderId,orderQuantity,autopartName,autopartManufacturingCompany,vehicleType,supplierName,deliveryLocation,orderDate

Operations: Order createOrder(Order order);

List<Order> fetchOrders();

Order updateOrder(Order order);

**void** deleteOrder(Order order);

**Supplier**

Attributes: id,supplierName,supplierAddress,supplierContact,supplierBaseUrl,discountTimeApplicable,supplierType

Operations: Supplier addSupplier(Supplier supplier);

List<Supplier> fetchSupplierList();

Supplier updateSupplier(Supplier supplier);

**boolean** deleteSupplier(Supplier supplier);

**🡺For Auto Order Creation**

Spring Scheduler with cron configuration is used to trigger the call every 5 days for local and International(cron time in discount window) suppliers which fetches all the Autoparts from database

Checks if the availableQty is less than threshold Quantity .If yes adds it to a list of orders with minOrderQuantity and triggers the order to supplier on his website

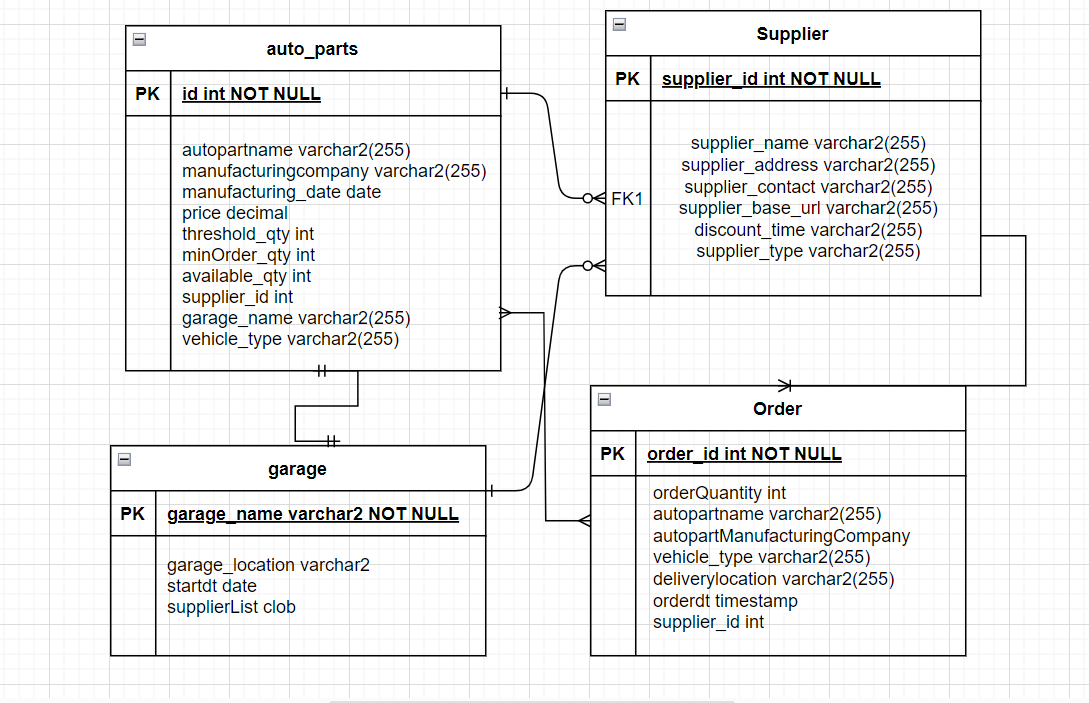
🡺**Java Persistence API** has been used for all CRUD operations like add,fetch,update and delete for entities like AutoParts, Suppliers, Garage and Orders(for manual order creation).

Endpoints for the same have been exposed through controller.

**🡺Depending on requirement** the application can be extended to multiple garages in single location as well as multiple locations with a plethora of suppliers both Local and International and multiple vehicle types(2,3,4,6,8 wheelers)

🡺**Depending on volume** the application can be enhanced for automated scaling,real time load balancing, distributed caching, reactive api leverage for low latency and high throughput, Database sharding etc

**Entity Relationship Diagram**

****