Coding Challenge - Inventory Management

Introduction

Design an inventory management service for a grocery store to help staff manage inventory and cashier terminals.

Requirements

The service should support features such as create/update/read/delete information on existing products such as name, ID, UPC (barcode number), price, and inventory count. In addition, the service should also provide support to cashier terminals. Cashier terminals should be able to create new orders and add products purchased by customer by UPC number and quantity to orders. The service would them be able to calculate total price of orders including applicable taxes (13% of total price). The service should provide HTTP REST API endpoints to fulfill its functionality.

Guidelines

Please note that you **do not** have to strictly follow everything in this guideline. The guideline only suggests the minimal features required to accomplish the task. Feel free to improvise as you see fit. However, your design is required to provide, at least, all functionality described in the *Requirements* and *Endpoints* sections.

Domain Entities

- **Product**

  Represents a single product carried by the store. The object should include at least product name, ID, UPC number, and inventory count at this store.

- **Cashier**

  Represents a cashier terminal in the store, identified by an unique ID.

- **Order**

  Represents an order placed by a customer. The order may consist at least 1 product purchased from current store and a cashier ID. Customer may purchase any product at any quantity as long as the store has enough inventory to fulfill the order.

- **Payment**

  Represent payments made by customers for their orders. Customers may choose to pay for their orders using multiple payment methods, such as cash, credit cards, debit cards, and gift cards. You are only required to store type of the payment, amount, and order ID. However, orders do have to be paid in full before it can be completed.

Endpoints

- List all products
- Find a product by its ID
- Find a product by UPC number
- Edit a product
- Create a cashier terminal
- List all cashier terminals
- Find a cashier terminal by ID
- Create an order
- List all orders
- Find an order by ID
- Find all products purchased and quantities of each product in an order
- Add a product to an order by UPC and quantity
- *(Optional)* Remove products from an order
- Get total price of an order
- Add a payment method to an order
- Close an order when payments are made in full
- List all payments made for an order

Considerations

- Payment must be made in full before an order can be closed.
- Cashier should not be able to add products to an order without enough inventory.
- Error handling
- Multiple instances of the service may be deployed to balance the load horizontally.
- The service maybe queried frequently during rush hours. It might not be a good idea to run all tasks synchronously on API calls.

Additional Tasks

Promotion

Products may go on sale for a predetermined period of time at a discounted price(in percentage to the total price of applicable products). However, customer may be required to purchase 2 or 3 of the same product to qualify for the discounted price*(e.g. buy one get one for half price)*. Alternatively, we may also ask customers to purchase different items to qualify for the discounted price. *(e.g. A game console and an additional controller bundle)*. The service should automatically recognize all applicable promotions in orders and apply discounts when calculating total price. A product should not exist in multiple promotions at any time. However, they may be added to promotions with different time ranges.

Furthermore, if you choose to implement this feature, your order object should also include all promotions applied.

Sales Report

Provide hourly and daily report on sales of each product, including popularity ranking of each product. Reports may be fetched frequently; it might be a good idea to save generated reports in database.

Optional Tasks

You do not have to complete all or any optional tasks. However, please specify which optional tasks you choose to challenge in your endpoint documentation.

Recall

Sold products may be recalled by their manufactures for various reasons. Add an endpoint to search for all orders placed by customers with specific product.

Inventory Alert

Allow manager to set a threshold on inventory of specific products. Once reached, an alert should be generated and stored.

Security

API invocations must be authenticated. Each cashier terminal is assigned a unique token in order to authenticate their requests with the API service.

Submission

Your completed task should include

- A microservice project, preferably written in Java.
- Database initialization script if necessary.
- Instructions on starting your microservice.
- Brief documentation for each endpoint you implement.