Java Full Stack Program

Personal Project - Online Shopping Application

Project Overview

- The retailer tycoon, Super Duper Mart™, requires your assistance in setting up their online shopping website. The following is the requirement they sent over. Based on these requirements, please come up with a RESTful application that is equipped with the necessary endpoints to return the needed information for each of their requirements.

Part 1 - Backend

Overall Requirements (Backend)

- Use Spring Boot, Hibernate (HQL, Criteria), MySQL, **Spring Security + JWT**, **Spring AOP**, **Spring Validation,** to develop the backend.
  - Please do not use native SQL query, JDBC/JdbcTemplate or Spring Boot JpaRepository/CrudRepository
  - At least one of the DAO methods need to be implemented with Criteria.
- Use Github as version control system to manage your project codebase
  - please make your repository PRIVATE, and share with aiden.gong@beaconfireinc.com, kevin.qi@beaconfireinc.com, will.wang@beaconfireinc.com.
- RESTful application required following a layered architecture
  - @RestController, @Service, @Repository.
- Use Postman to present your project by calling your RESTful endpoints.

ERD

- Please design the ERD by yourself based on the requirements.

User

A customer(buyer) of the app

- Register
- Log in
- Home Page
- Purchasing
- Watchlist
- Statistics

Registration

- Before being able to purchase products, a user has to first register.
  - Your application should prevent registration using the same username and email.
  - Only **username**, **email** and **password** are required to register an account.
- Can only register users (not admins) via the registration endpoint.

Login

- If the user has entered the correct credentials, they may proceed to the corresponding page based on their authorities.
- If the user has entered incorrect credentials, a custom named exception ‘InvalidCredentialsException’ should be thrown and handled by the Exception handler. The message the user will get is: “Incorrect credentials, please try again.”

Home Page

- (**GET** product list - user) The user is able to view all of the products. An ***out of stock product*** should **NOT** be shown to the user.
- (**GET** product detail) When a user clicks on one product, the user should be redirected to the detail page of that product, including the description and price (retail_price) of the product. (The user should **NOT** be able to see the ***actual quantity*** of any items).
- After purchasing the product, the user should be able to view order details including, order placement time and order status which is ***Processing***, ***Completed*** or ***Canceled***.

Purchasing

- (**POST**: Place an order)The user should be able to purchase listing items with a specified quantity by creating a “Processing” order. After a user places an order, [the item’s stock should be deducted accordingly].
  - The user should be able to purchase multiple different items within a single order.
  - If the quantity of an item that the user is purchasing is greater than the item’s stock, throw a custom exception named ***NotEnoughInventoryException*** using Exception Handler and the order should not be placed.
- (**PATCH**: Cancel the order) The user should be able to cancel an order by updating the status from ***Processing*** to ***Canceled***.
  - If so, [the item’s stock should be incremented accordingly] to offset the auto-deduction that took place when the order is first placed. However, a “Completed” order cannot be changed to “Canceled”.

Product Watchlist

- The user can add/remove products to/from their watchlist.
- The user can view all in stock products within their watchlist.

- (Please design these API above by yourself)

Summary

- (GET all orders by that user) The user should be able to view all their orders.
  - Note that the wholesale_price and retail_price of a product can be adjusted by the seller, implement something to prevent the adjustments from affecting previous orders.
- (GET order detail) The user can then click and look into any one specific order created by them, completed with the items included in that order.
- (GET recently bought products) The user can also view their top **x** most recently purchased items (excluding canceled orders, use item ID as the tiebreaker).

Admin

The seller

- Home Page
- Listing (product management)
- Selling
- Order
- Statistics

Home Page

- The seller should be able to view a dashboard, consisting of the following:
  - **GET Order information**, with details of order placed time, users who placed the order and the order status (Processing, Completed, Canceled).
  - (Bonus) Proper pagination: A page should only have 5 orders and each page should be fetched from DB only when requested.
- (GET Product List - admin) Listing information, the current products that are listed to sell.
  - (GET product detail - admin) When the seller clicks on one product, the seller should be redirected to the detail page of that product, including the description, wholesale_price, retail_price and stock's quantity of the product; the seller should be able to modify the wholesale_price, retail_price, description and quantity of a product.

Listing

- (POST add product) The seller should be able to add products. A product has fields including description, wholesale_price, retail_price and quantity in stock.
  - The wholesale price is the price which the seller paid for the product.
  - The retail price is the price which customers pay for the product.
- When one product is sold, the quantity of that product should be deducted accordingly. And such quantity should be reflected on the dashboard.

Order

- (PATCH) The seller should be able to complete a “Processing” order by updating its status to “Completed”.
- (PATCH) The seller should also be able to cancel an order
  - for some reasons, such as that the product is sold out locally, by updating the order status to “Canceled”.
  - If so, [the item’s stock should be incremented accordingly] to offset the auto-deduction that took place when the order is first placed.
  - However, a “Canceled” order cannot be completed, nor can a “Completed” order be canceled.

Summary

- (GET) The seller can see the total number of successfully sold items.
- (GET) The seller can see which 3 products are the most popular/sold (excluding canceled and ongoing order).
- (GET) The seller can see which product brings the most profit.
    - The profit is calculated as (retail price - wholesale price).
    - Note: This should address situations where the seller alters either the wholesale_price or retail_price, causing a discrepancy when comparing between the past orders and the current updated product details.
    - This should not include “Processing” and “Canceled” orders.

Additional Features

- Data Transfer Objects
- AOP
- Exception Handling
- Logging
- Security (not required at this checkpoint)