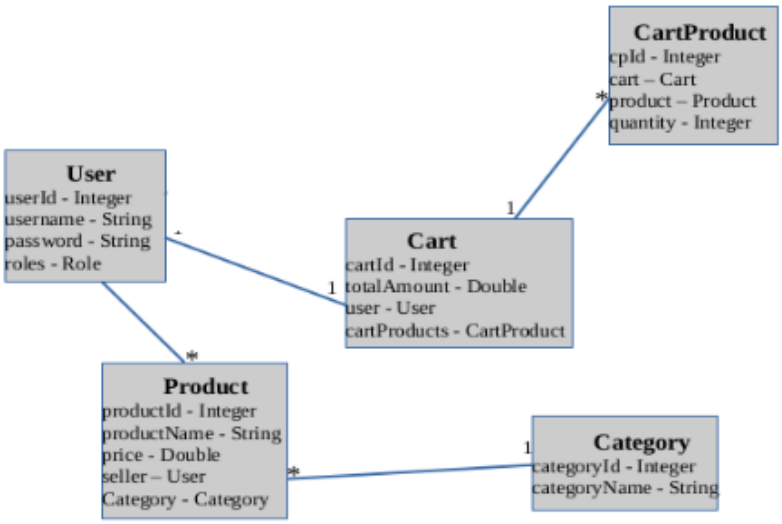
McDiffyStore is a vendor who sells various products ranging across different categories. They hired you to develop a distributed e-commerce platform to move their business online. As an initial MVP you are required to develop a restful API backend application in spring boot.

Here is the requirement for the application.

Database models are already created and initialized as below:

User is saved as UserInfo in DB



DB initialized with following default data:

The Password has been saved in Encrypted form in DB

Categories:	**Role**	**UserInfo**	**Cart**
categoryId,categoryName 'Fashion' 'Electronics' 'Books' 'Groceries' 'Medicines'	**role** 'CONSUMER' 'SELLER'	**userId, username, password, roles** 1,'jack', 'pass_word',"CONSUMER" 2, 'bob', 'pass_word',"CONSUMER" 3,'apple', 'pass_word',"SELLER" 4, 'glaxo', 'pass_word,"SELLER"	**totalAmount, userId** 20, 1 0, 2

Product	**CartProduct**
price, productName, categoryId, sellerId 29190, 'Apple iPad 10.2 8th Gen WiFi iOS Tablet', 2,3 10, 'Crocin pain relief tablet', 5, 4	**cartId, productId, quantity** 1, 2, 2

Note:

- Your job is to create the following APIs, use JWT authentication with roles to protect consumer and seller specific endpoints.
- All authentication and authorization processes should be implemented using **JWT** Token.
- JWT token should be sent as a Bearer token in Authorization request header.
  For example: Authorization value would be "**Bearer <SPACE> <JWT TOKEN>"**.
- Consumers can search, add, update and delete items in cart.
- Sellers can add, update and delete products to the database.

Endpoints:

• APIs preceding with /api/public are public APIs and can be accessed by anyone.

• APIs preceding with /api/auth/consumer are authenticated and consumer APIs.

• APIs preceding with /api/auth/seller are authenticated and seller APIs

• if authenticated endpoints are accessed without JWT, return 401.

• if a consumer endpoint is accessed with seller JWT or vice versa, return 403.

Below are public endpoints:

1. Get - /api/public/product/search

- This endpoint takes a query parameter **'keyword'** and returns all the matching products containing the keyword either in productName or categoryName with status of 200

Request: **localhost:8000/api/public/product/search?keyword="tablet"**

Response:

[{"productId":1,"productName":"Apple iPad 10.2 8th Gen WiFi iOS Tablet","price":29190.0,"category":
{"categoryName":"Electronics"}},{"productId":2,"productName":"Crocin pain relief tablet","price":10.0,"category":
{"categoryName":"Medicines"}}]

Error Response: If any error occurs while fetching the data, return the status of 400.

2. POST- /api/public/login

- It takes username and password in json body, authenticates and returns JWT token with status code as 200, which should authenticate both consumer and seller respectively.

Request: **localhost:8000/api/public/login**

```
{
    "username":"bob",
    "password":"pass_word"
}
```

Response:


```
{
"accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJib2IiLCJpYXQiOjE3MzMxMjA3MTQsImV4cCI6MTczNDAyMDcxNH0.
    Dc0mqKw2fCo9wyEdXKXWSxQoAczztwhhZIlmeW7s6Wc",
"status": 200
}
```


Error Response:

- If the credentials are wrong or any error occurs return the status code of 401.

Below are all authenticated endpoints: Handled by CONSUMER

3. GET - /api/auth/consumer/cart

- It should return the consumer's cart.

Request: localhost:8000/api/auth/consumer/cart

Response:

{"cartId":1,"totalAmount":20.0,"cartProducts":[{"cpId":1,"product":{"productId":2,"productName":"Crocin pain relief tablet","price":10.0,"category":{"categoryName":"Medicines"}},"quantity":2}]}

4. POST - /api/auth/consumer/cart

- It should take a Product json in request body and adds it to the consumer's cart.

Request: localhost:8000/api/auth/consumer/cart

{"productId":3, "category":

{"categoryName":"Electronics","categoryId":"2"}, "price":"98000.0",
"productName":"iPhone 12"}

Response: It should return the status code of 201.

Note: If the consumer tried to add the existing product in cart again, return the status code of 409.

5. PUT - /api/auth/consumer/cart

- It takes a CartProduct json in request body and updates the quantity of the product in cart.
- If the product is not in the cart, add the product to cart with supplied quantity. If the quantity of the product is zero, then delete the product from the cart.

Request: localhost:8000/api/auth/consumer/cart
{"product": {"productId":3,"category":
{"categoryName":"Electronics","categoryId":"2"},"price":"98000.0","productName":

"iPhone 12"},"quantity":3}

Response : Return the status code of **200**

6. DELETE - /api/auth/consumer/cart

- It takes the Product json in request body and removes the product from the cart.

Request: /api/auth/consumer/cart
{"productId":3,"category":
{"categoryName":"Electronics","categoryId":"2"},"price":"98000.0","productName":
"iPhone 12"}

Response: returns the status code of 200.

Below are all authenticated endpoints: Handled by SELLER

7. GET- /api/auth/seller/product/{productId}

- It should return all the products owned by the seller using the productId.
- It returns the product identified by the supplied path parameter productId only.

Request: /api/auth/seller/product/1

Response:

{"productId":1,"productName":"Apple iPad 10.2 8th Gen WiFi iOS Tablet","price":29190.0,"category":
{"categoryName":"Electronics"}}

8. POST- /api/auth/seller/product

- It takes the product json in request body and saves it to database.

Request:

{"productId":3,"category":
{"categoryName":"Electronics","categoryId":"2"},"price":"98000.0","productName":

"iPhone 12 Pro Max"}

Response:

- It should return the status code of 201.
- And redirect url (created URI) - http://localhost/api/auth/seller/product/3

9. GET - /api/auth/seller/product

- It should return all the products owned by the seller.

Request: /api/auth/seller/product

Response:

{"productId":1,"productName":"Apple iPad 10.2 8th Gen WiFi iOS Tablet","price":29190.0,"category":
{"categoryName":"Electronics"}}

10. PUT - /api/auth/seller/product

- It takes a product json in request body with mandatory product id and updates the product in the database.

Request: localhost:8000/api/auth/seller/product

{"productId":3,"category":

{"categoryName":"Electronics","categoryId":"2"},"price":"98000.0","productName":

"iPhone 12 Pro Max"}

Response: It should return the status code of 200

11. DELETE - /api/auth/seller/product/{productId}

- It takes a productId path parameter and deletes the product from the database.
- If the product is not owned by the seller, then return the status code 404.

Request: /api/auth/seller/product/1

Response: It returns the status code of 200.

Error Response:

- It should return the status code of 404.
- If the request is in this format: **/api/auth/seller/product/2**

Take a look at the testcases to understand more on how the validation works.

Good Luck and Start Coding!