Shopping App

Design and develop a shopping web app using Java and Spring Framework in which users can perform the following actions:

- Enter the quantity of products the user wants to order
- Check if the product is available in the desired quantity
- Check available coupons and apply one, if valid. A coupon can only be applied once per user
- Make payment for the order
- View the status of all orders

For the sake of simplicity, you can assume the app has only one type of product in a fixed quantity that can be fetched from a config file during startup. For every user action, the response should be a JSON.

Application Requirements -

Create a REST API based on MVC architecture to simulate the following::

1. **GET {{url}}/inventory**
Status Code: 200
On Startup
```
{
  "ordered": 0,
  "price": 100,
  "available": 100
}
```
When we have ordered 50 items
```
{
  "ordered": 50,
  "price": 100,
  "available": 50
}
```

2. **GET {{url}}/fetchCoupons**
Status Code: 200
```
{
        "OFF5": 5, //discount in percentage
        "OFF10": 10
}
```

3. **POST {{url}}/{userId}/order?qty=10&coupon="OFF5"**
Status Code: 200
```
{
        "orderId": 100,
        "userId": 1,
        "quantity": 10,
        “amount”: 950 //if the price of 10 quantity is 1000, user has to pay 950 after applying 5%
off coupon
        "coupon": "OFF5"
}
```

Status Code: 404

```
{
        "description" : "Invalid quantity" //Quantity is either less than 1 or more than the
maximum quantity of product available
}
```

Status Code: 404

```
{
        "description": "Invalid coupon" //User has already used the coupon before, or coupon
does not exist
}
```

4. **POST {{url}}/{userId}/{orderId}/pay?amount=950**
//Mock the payment API to randomly return any of the following status codes
Status Code: 200

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100001,
        "status": "successful"
}
```

Status Code: 400

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100002,
        "status": "failed",
        "description": "Payment Failed as amount is invalid"
}
```

Status Code: 400

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100003,
        "status": "failed",
        "description": "Payment Failed from bank"
}
```

Status Code: 400

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100004,
        "status": "failed",
        "description": "Payment Failed due to invalid order id"
}
```

Status Code: 504

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100005,
        "status": "failed",
        "description": "No response from payment server"
}
```

Status Code: 405

```
{
        "userId": 1,
        "orderId": 100,
        "transactionId": tran010100006,
        "status": "failed",
        "description": "Order is already paid for"
}
```

5. **GET {{url}}/{userId}/orders**
Status Code: 200

```
[
        {"orderId": 100, "amount": 950, "date": "25-11-2021", "coupon": "OFF5"},
        {"orderId": 101, "amount": 950, "date": "25-11-2021", "coupon": "OFF5"}
]
```

6. **Get {{url}}/{userId}/orders/{orderId}**
Status Code: 200

```
[
        {"orderId": 100, "amount": 950, "date": "25-11-2021", "coupon": "OFF5", "transactionId":
tran010100001, "status": "successful"},
```

{"orderId": 100, "amount": 950, "date": "25-11-2021", "coupon": "OFF5", "transactionId": tran010100002, "status": "failed"}

]

Status Code: 404

```
{
	"orderId": orderId,
	"description": "Order not found"
}
```

Guidelines:

- Build an executable Java application project and share steps to run the project
- Follow good coding practices and use comments wherever necessary
- List down any assumptions you make while designing
- Perform error handling
- Share project by uploading code in a git repository
- Maintain an appropriate database to keep track of all users, orders, coupons and transactions. You are free to use any DB of your choice.
- Share ER and class diagram of your design