Final Project

Goal:

Design an online shopping website that which at least contains following services

- Item Service (including inventory service) mongoDB
- Order Service cassandra
- Payment Service
- Account Service

On your website, customers should be able to create an account, select an item, add the item to cart, check out (create an order), update order, and cancel order.

General requirements:

1. You are responsible for only the backend part of this project, you are recommended to implement some sample web pages for demo purposes if you are able to do so.
2. Create one or multiple repositories for these services, dockerize services and their dependencies, make sure your applications are “one-click” runnable.
3. You’re encouraged to team up, up to 3 members per team, and you’re supposed to deliver a more deliberated, comprehensive solution than solo players.
4. Please share your git repo link and grant access to following Github userIds once you started the project:
    a. CTYue
    b. Liam-Zhou
    c. CharlesCCC
5. Tech Stack:
   Spring boot, Maven, Spring Security, Junit/Mockito/PowerMock, Jacoco, Swagger, Hibernate, Spring JPA, Spring Cloud and everything else we covered in class.
6. Use Spring Cloud OpenFeign and/or RestTemplate to implement inter-service synchronous communications.
7. You are required to use all following databases: MySQL/PostgreSQL, MongoDB, Cassandra.
8. You are required to use Kafka or RabbitMQ for event-driven asynchronous communication.
9. You are recommended to use Spring Cloud modules such as Eureka, Hystrix to establish microservice components.

10. Unit testing coverage: at least 30% coverage for each service layer. Covering all possible cases.
11. An **authentication server** is required, apart from the 4 business services we mentioned (*you can also implement the authentication server as part of Account service*), this server helps to secure services by providing authentication and authorization facility, e.g. once a user logs into its account successfully, the server generates a token, with this token as part of the request header, requests sent to order service can be accepted.

Item Service:

The item service provides item related information including unit price, item name, item picture urls, UPC (universal product code), item id.

Item Service stores everything about an item, we usually call such data as metadata, you may use MongoDB to realize it as it's difficult to finalize a schema for such data.

Item service is also responsible for inventory lookup and update, by returning remaining available units of a product.

Order Service:

The order service supports both synchronous and asynchronous communications, it produces Kafka messages and also consumes kafka messages.

Order Flow:

- Create Order (right after user clicks the submit button, an order will be created)
- Cancel Order
- Update Order

Order Service APIs:

Create Order
Cancel Order
Update Order
Order Lookup

While RESTful APIs remain stateless, each order itself has a state, for example.

Order Create, Order Paid, Order Completed, Order Cancel

Consider using Cassandra database for order service, and design a reasonable schema for order information.

Please note that above APIs provide a synchronous approach, order service should publish order information to some other services…

Payment Service:

integrates with order service, provides REST APIs, and publish payment transaction results to consumers.

- Submit Payment
- Update Payment
- Reverse Payment: Refund
- Payment Lookup: lookup a payment, return its status

Idempotency should be guaranteed throughout the payment flow, as we don't want to double-charge customers or double-refund them.

Account Service:

- Create Account
- Update Account
- Account Lookup
- …

Users should be able to create/update their account with following necessary information:

- User Email
- User name
- Password
- Shipping Address
- Billing Address
- Payment Method

You may use MySQL or PostgreSQL for data persistence.