Final Project

Project 1: FINANCE - MasterCard - MDS2.0

Scenario: When you put your mastercard in apple pay, that card will have an attached TOKEN, you go to a store and checkout, you put your Iphone on the device, the device will recognize the TOKEN and pass it to mastercard application for validation. This application will use that TOKEN to get the account status, there could be many status, the application will convert the status to final two status (active and inactive) and send it back to device.

Goals:

API:
register
update
delete(不删)

- Create a Spring Boot application **AccountManagementSystem**
  - To support Register account(generate new token), validate token status, update account for token status, delete account(delete account will only change status to DELETED, not really remove it from database).
  - Store token status inside Database. Different status (like ACTIVE, DELETED, DEACTIVATED, SUSPENDED etc)
  - Create 1 consumer to receive the token and then call database to get the account and store the status to another table called POS_STATUS (the status should be either ACTIVE or INACTIVE)
  - Create 1 producer, once receives the token and then load from database to read the POST_STATUS table, send the status back to the **MasterCardApp** device (you can send to another kafka topic or just print them out)

- Create another Spring Boot application **MasterCardApp**
  - When customer is shopping, it will get Token from the card, and send it to **AccountManagementSystem** via kafka producer.
  - It also can hear back Token status from **AccountManagementSystem** via kafka queue, and then returns the result to the customer(Print it out).

Tips:

- Using Enum for different token status
- Using topic to identify different kafka flows
- Please add about 20 – 30 dummy data in each tables if needed.

Basic Level: (Points 50)

- Proper Design of Entity and table. (Point 10)

- Register new account, and generate one account id for it, and generate one unique token for it. And data can be stored in database. (Point 10)
- Get Token status API can work properly. (Point 10)
- Token status can be changed to DEACTIVATED via API. (Point 5)
- Token status can be changed to DELETED via API. (Point 5)
- Proper logs, proper class, proper annotations. (Point 5)
- Proper unit tests covered. (Point 5)

Advanced Level: (Points 50)

- Created SpringBoot **MasterCardApp** and can communicate successfully with **AccountManagementSystem** App. (Point 10)
- One kafka flow from **MasterCardApp** (producer) to send the token to **AccountManagementSystem** (consumer) works. (Point 20)
- The other kafka flow from **AccountManagementSystem** (producer)to return the Token status back to **MasterCardApp** (consumer), and then the result can be printed out. (Point 20)

Project 2: TRAVEL – CWT

Scenario: **CWT** is working with 3rd party agent **CONCUR** to retrieve customer data
The system will take input from kafka stream, the stream data is customer email, firstname and last name. Based on this stream data, you will visit **CWT** database to fetch the token that matches the input. Token usually has expiration on it. Once you fetch the token, you will use this token to call the **CONCUR** api to fetch detailed user information, which includes the user detail data and payments. This detailed user data will be send out via kafka producer so another service can consume.

Goals:

- Create a SpringBoot application **CustomerSystem**
  - API can receive user email, first name, last name.
  - Kafka flow 1 producer to send user email to **CWT**
  - Kafka flow 2 consumer to shows user infos from **CONCUR**. And print it out.
- Create another SpringBoot application **CWT**
  - Kafka flow 1 consumer to receive the customer email, first name, last name, and based on it to fetch the token.

    - The token should stores in database table (column: email, token, expiration_timestamp).
    - If token is valid, use it. Otherwise, if token is expired or will expire in 2 minutes, create a new token and stored in database, and then returned.
    - Encode the user email and expiration_timestamp inside the token.
- Create another SpringBoot application **CONCUR**
    - Manually using the token returned by **CWT**, and call API to fetch user details using token.
    - Decode the token to check if the token has been expired. If yes, reject the request, and you need to call **CWT** to generate a new one. If no, decode the email address from the token and then get data from database table.
    - Use this as **CONCUR** data simulation (https://jsonplaceholder.typicode.com/users)
    - APIs to update or delete customer infos inside **CONCUR** server.
    - Once get the user details from database, send it as the producer of Kafka flow2 back to **CustomerSystem**.

Tips:

- Please add about 20 – 30 dummy data in each tables if needed.
- Please create different roles for your database for different services, to mock different permissions of tables.

Basic Level: (Point 50)

- Proper Design of Entity and table. (Point 10)
- Create **CONCUR** application, and the Get customer infos API works. (Point 10)
- Token decode works. (Point 10)
- Update API works. (Point 5)
- Delete API works. (Point 5)
- Proper logs, proper class, proper annotations. (Point 5)
- Proper unit tests covered. (Point 5)

Advanced Level: (Point 50)

- Created **CustomerSystem** and **CWT** applications, and all three can integrated with each other (Point 10).
- Kafka flow 1 from **CustomerSystem** to CWT works.
- Kafka flow 2 from **CONCUR** to **CustomerSystem** works.