Week6Day5

--

Notification processing part.
    security check.

LIVE MCQ

Project in Resume

Summary

=======

Stripe notification

Security HMACSHa256 - Stripe library

https://docs.stripe.com/checkout/fulfillment

checkout.session.completed
checkout.session.async_payment_succeeded
checkout.session.async_payment_failed

Event Object
https://docs.stripe.com/api/events/object

Every event has the possibility to send different event.data.object field.

In future we might have to listen to more events for process.
Definitely dynamic object structure handling is needed (based on the event that we receive).

Gson
string to object
object to string

JsonObject

First find the type of event
based on type, we can create more java object structures

CheckoutSession COmpleted:

status complete
payment_status paid

Update Payment Status to Success
Inform ClientApp, that its success (Async ActiveMQ)

8 weeks
7 weeks LIVE + 1 week for demo & task complete

10 days - common training

3 Sprints (2 weeks)
Sprint1
Week3 - Week4

Sprint2
Week5 & Week6

Sprint3
Week7 & Week8 (28Mar)

LWD - 28Mar
29-30Mar - Issue Internship Completion Letter

Either
By 28Mar EOD
5 Demos - 70% or more attendance
7 Demos - Less than 70%

By default recording access if upto 4months from start of internship
3Feb (2months Feb-Mar)
+ 2months
Apr - May2025

Total access for 1years

Google Review

===

Have total 4internship
    Feb-Mar

7Apr - 2months

Extend by 2months
Start date: 3Feb
End date - 30May

2k

Week7:
    ActiveMQ
    AWS - SecretsManager, RDS...
    UnitTesting
    Logging

======

how to present during interview....

Model 1. End-to-end Stripe integration
Model 2. Stripe Integration + Core Payment System

Optional - ActiveMQ, Security, Notification

=====

Mandatory / Required----------

Stripe Integration
    Analysis of the API
    Working with DashBoard
    API integration with Stripe (R) - 1/5
    Spring Boot & Microservice (R) - 1/5

Basics of AWS (R) - 1/5
Error handling system - 1/5

========

1. Analysis of the Stripe API
    - Reading documentation
    - exploring how these apis work
    - setting up in postman & testing
    - relating to how we will implement in our project

2. Working with Stripe DashBoard
    - Test account vs LIVE account
    - API Keys - for making all API calls
    - Rolling the keys feature visible
    - Webhook details are visible in dashboar
    - You were having access to test dashboard
    - Events & logs for debugging is available in dashboard

3. Core payment processing system (R) - 1/5
    - 2 apis in processing service & how validation calls it
        - Contrats to make payment possible end-to-end
    - DB design & how to manage with process DB
    - payment statuses & how each status is updated in flow of payment
    - How Stripe API interacts in entire flow
    - how success / error responses are handled
    - Recon system aswell.

4. Payment Status Management - 1/5
    - define why status tracking needed
    - what 5 statues we have
    - when is each status triggered
    - designing for coding system for status tracking
        factory, handlers, updates
    - What are final status - Success & failure
    - Communicating final status to client app.
    - if status is pending for long, then recon should work
    - all status should finally end it either success/failure

5. API integration with Stripe (R) - 1/5

- API documentation reading
- postman setup
- dashboard API keys for api calls
- Rest standards
- create, get, expire api call
- how these apis work with core payment system
- developing stripe-provider-service for end-to-end
- how you communicate with stripe PSP (restclient)
- security - when you call APIs. Basic Auth
- How to interpret success / failure.

6. Spring Boot & Microservice (R) - 1/5

- rapid application dev
- Autoconfiguration
- Starter
- Embeded tomcat
- sensible defaults
- All our services are microservices
- 9 attributes defined my martin fowler. we tried to implement max as possible
- via service, domain, decentralized DB, Rest call, ALB & API Gateway.. for service communice RestCLient,
- circuitbreaker
- Infra will auto scale & descale.
- CI-CD

Basics of AWS (R) - 1/5

mobaxterm
java -jar
Linux
build & deploy

7. Error handling system - 1/5

- FOr every error give meaning errorcode & errorMessage
- standard response structure form service
- defined Enum to represent code & message

- throw custom exception from code, where you identify error
- Have http status code along with it for communicating in response
- Have globalexceptionhandler for processing exceptions
- Spring boot calls global handle when exception is triggered
- custom vs generic exception
- how the error received form stripe PSP should be processed & returned to invoker. stripe-provder error back to validation service
- when we add try-catch we need to be careful

8. Spring Boot JDBC with MySQL - 1/5
- build on top of core jdbc
- internal NameedParameter object for working with DB
- internal connection pool
- working with DB.
- DDL / DML
- git repo for db tracking
- Txn tabe that we have, and how to reflect CRUD operations from java layer
- why not orm & choosing this approach. when to go for ORM
- version of MySQL. tools to connect.
- methods of namedParatmer vs using jdbcTemplate
- When exception happens while working with DB, then how you handle it.

9. RestAPI
- communication between 2 systems happen with API
- communication in web works, web api, or webservice
- when API are developed as per Rest Standard, its rest api
- build around concept of resource
- something which can be uniquely identified & manipulated on net is a resource
- 2 types of operations possible on resource
    CRUD / Actions
- Using HTTP method & naming of your endpoint, defines the functionality
- POST /students - create it in db
- versioning of apis
- all key annotations used to make Rest API in Spring boot.. @RestCOntroller, @requestmapping, @postmapping,.. @RestHeader, @RequestBody, @Pathvariable
- Post & get & put & deltate relate..
    ID from post is passed as the path variable of other APIs..
- HttpStatusCode handling

- all our services we are developing using Rest
- Stripe is doing rest based communication
- we are using JSON body as request & response.
- Stripe has form url encoded in request & response is JSON.

10. Design Patterns
- standard solutions for commonly reoccurring problems
- some best practices to build good systems
- We are using factory dP & builder DP in our system
- Explain factory pattern
    1 main class, multiple child
    define a method which returns type of parent, & creats objects of child depending on some parameter.
- BuilderDP - for object creating in optimized manner
- Lombok @Builder

MyCLass.builder().build
- CircuitBreaker

- Singleton DP
- Chain of responsibility DB

11. Debugging & problem solving.
- using logs in system. no SOP
- logs to tell the flow of application, with needed data.
- sensitive data should not be logged
- for every request, you relate log statements with sourcecode to understand what is happening.
- whenever some problem occurs, you reproduce the problem to carefully check the flow.
- for more precise checking, you use IDE debugging with breakpoints.
- any exception triggered in application, you think logically why its coming by reasing caused by in exception trace.
- you go to internet & AI tools to make sense of the errors.
- Whatever internet solutions, you read it patiently and experient 1-by-1 whichever is relavent for you.
- the problem that you are facing, might definitely be based on other professionals of the wordls.. which we can find on internet.. and save time to speed up in our analysis.
- I am confident that my debugging skills are growing with more coding I do.

==========

12. stripe notification process - 1/5

- when customer is sitting on stripe hosted page. the activity of what happened for that payment, is known only to stripe PSP system.
- they inform us as part of notification
- they expect that we expose an Rest API, which they will invoke for informing us about success/failure
- they are sending us multiple events, but we need to focus only on 3 for interpreting the state of our payment
- checkout session completed, async success, asycn failed
- we expose APi in stripe-provider service which will be invoked form StripePSP
- internally for valid status we invoke processing service where our system payment status is update.
- for success cases, we update as SUCCESS & inform client app aysn
- for failed cases, we update as FAILED & infor to client app
- if we don't get notification, we keep the status as pending. later recon process will take it up.
- we apply security check before processing notification
- stripe psp has defined hmacsha256 based on security modle
- we use stripe library (maven dependency), to check whether its valid or not.

13. security layer -

APi call

notification hmacsha256

- any APIs which we expose to outer work has to be secured
- our APIs are exposed from validation service(clientapp) & from stripe-provider (notification)
- We are working as part of process & stripe-provider, currently we have not finalized on approach to use in validation service for security.
- however spring boot security module will be used in validation service for security.
- while making api call to stripe, stripe PSP has basic uath (apikey), so it ensures that request is coming only from us.
- while receiving notifications from stripe,we have hmacsha256
- we used stripe lirary for processing it.
- in infra, we will enable WAF for security.
- currently withing our system, the api can be directly access.. not sure, if in future we will

safeguard that or not.

- only valid requests will be processed.

ActiveMQ

========