1. Booking Service:

Added dependent libraries

- Eureka Discovery Client
- Spring Data JPA
- MySQL Driver
- Spring Web
- Kafka client

Added profile management, to select the database based on the selected profile (DEV, DEPLOY)

- DEV – this will enable H2 db (uses application-DEV.properties)
- DEPLOY – this will enable MySQL db hosted on AWS RDS (uses application-DEPLOY.properties)

Code currently submitted with DEPLOY profile.

To update the end point of MySQL RDS, uses application-DEPLOY.properties to be updated

Implemented the service with multi layered design consisting of Controller, Service, DAO, Exception, Model, Entity, DTO, Util layers

Exposed 2 Rest EndPoints

I. http://localhost:8080/booking
   - This is a post call to persist the booking information and return the booking details
   - Service layer implements the business logic of
     i. Getting available rooms
     ii. Calculating the room price based on no. of rooms, no. of days and room price
   - Dao layer persists the entity into the DB
   - Exceptions handled and logged as error
   - Controller advice written to calculate the time taken for the service to be processed

II. http://localhost:8080/booking/{{bookingId}}/transaction
   - This is a post call to update the transaction details into the booking info by connecting to Payment-Service
   - Service layer implements the business logic of
     i. Fetching the Booking based on Booking Id by calling DAO class
     ii. call payment service using RestTemplate to get the transaction id for the transaction info
     iii. use DAO layer to update the booking information based on the response from above step
     iv. publish the notification on to kafka topic “message”
   - Dao layer persists the entity into the DB
   - Exceptions handled and logged as error

- Exception handler will respond with ErrorModel for exception cases
- Controller advice written to calculate the time taken for the service to be processed

2. Payment Service

Added dependent libraries

- Eureka Discovery Client
- Spring Data JPA
- MySQL Driver
- Spring Web

Added profile management, to select the database based on the selected profile (DEV, DEPLOY)

- DEV – this will enable H2 db (uses application-DEV.properties)
- DEPLOY – this will enable MySQL db hosted on AWS RDS (uses application-DEPLOY.properties)

Code currently submitted with DEPLOY profile.

To update the end point of MySQL RDS, uses application-DEPLOY.properties to be updated

Implemented the service with multi layered design consisting of Controller, Service, DAO, Exception, Model, Entity, DTO, Util layers

Exposed 2 Rest EndPoints

I. http://localhost:8083/transaction
   - This is a post call to persist the booking information and return the booking details
   - Service layer implements the business logic of
     i. Map dto into entity and enrich the entity
     ii. Calculating the room price based on no. of rooms, no. of days and room price
   - Dao layer persists the entity into the DB
   - Exception handler will respond with ErrorModel for exception cases
   - Controller advice written to calculate the time taken for the service to be processed

II. http://localhost:8083/transaction/{{transactionId}}
   - This is a get call to get the transaction details
   - Service layer implements the business logic of
     i. Fetching the transaction based on transaction Id by calling DAO class
   - Dao layer persists the entity into the DB
   - Exception handler will respond with ErrorModel for exception cases
   - Controller advice written to calculate the time taken for the service to be processed

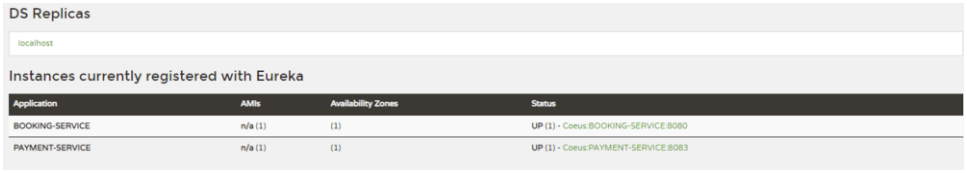
3. **Eureka Server**
   Run on port 8761
   Payment service registers as PAYMENT-SERVICE
   Booking service registers as BOOKING-SERVICE

4. **Notification Service**
   Simple java project with a main class written to consume messages from kafka topic "message" which is hosted on AWS EC2 instance

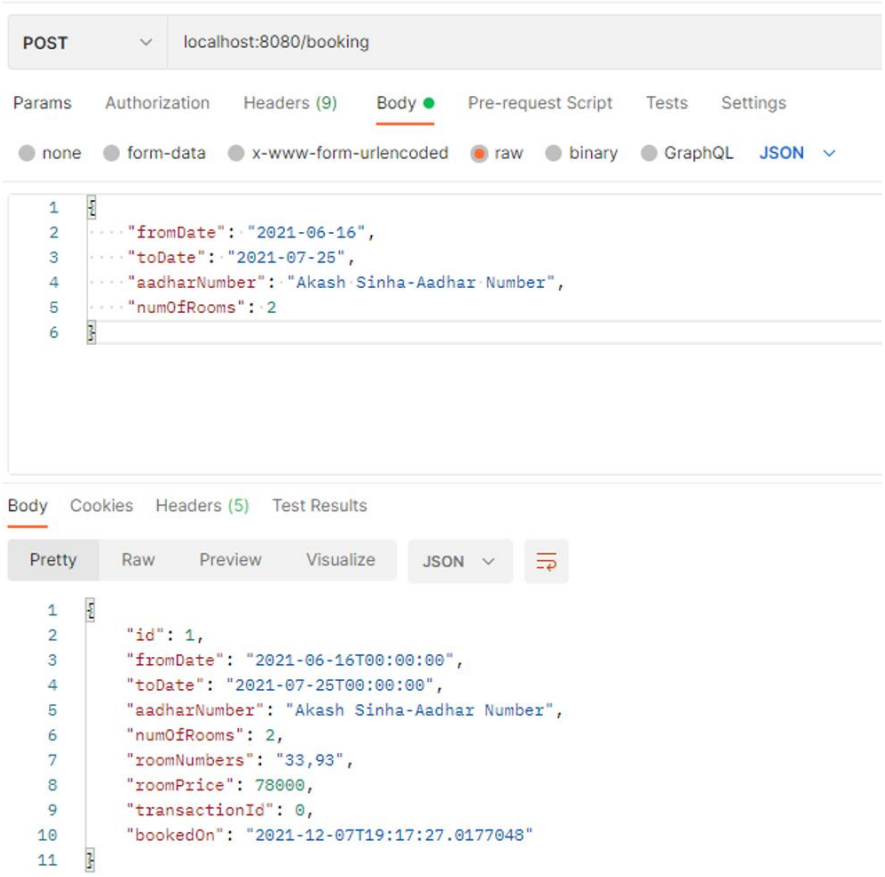
5. **AWS EC2 created to host Kafka.**
   After installing java and kafka, server.properties updated with IP of the EC2 elastic IP.
   Zookeeper server to be started
   Kafka server to be started
   Kafka topic created with replication factor 1 and partition 1
6. **AWS RDS created**
   Schema created: sweethome
   - create schema sweethome
     Tables automatically created with help of spring jpa

ScreenShots:

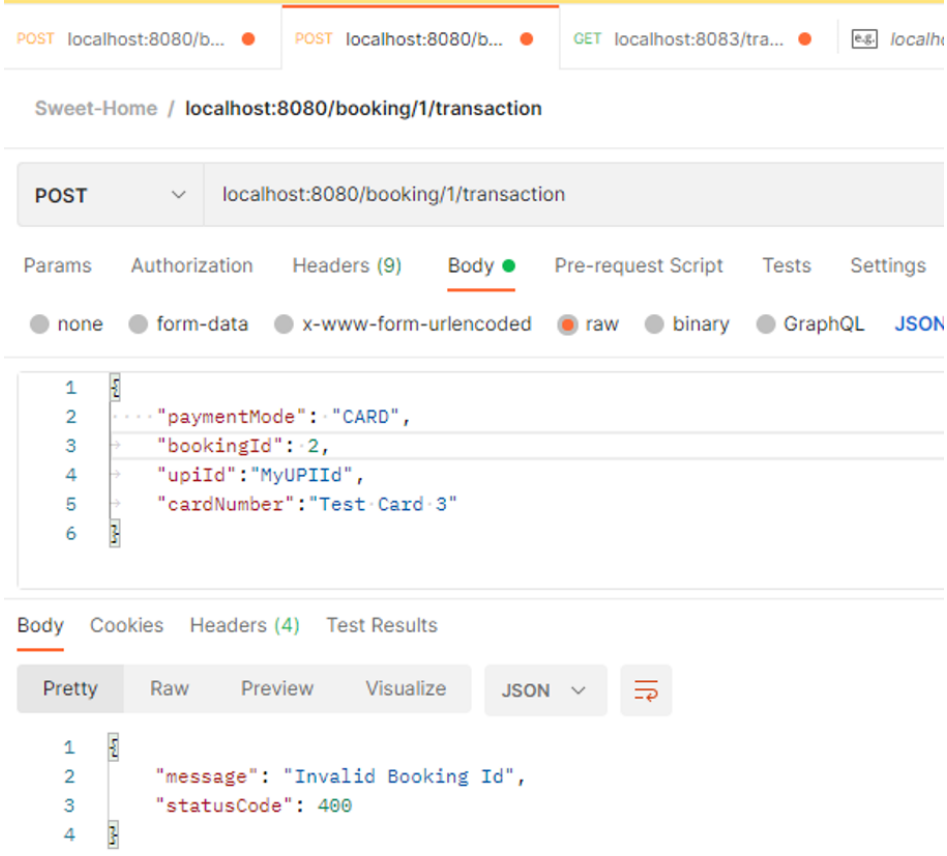
Eureka – service registry



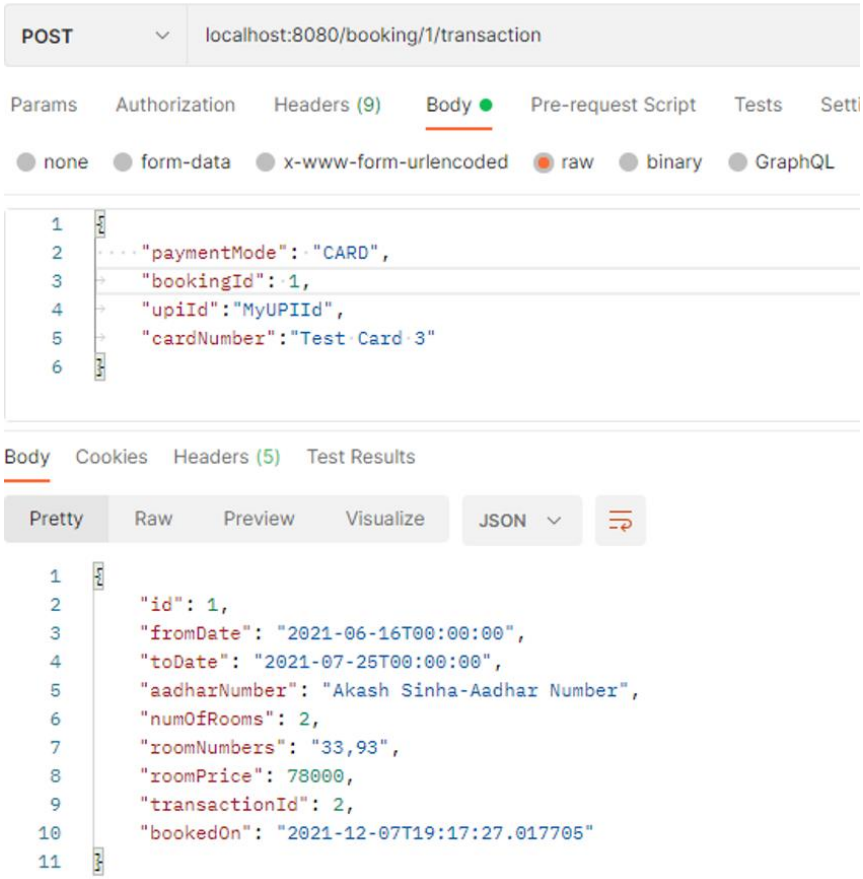
Post on booking



7. Transaction with invalid booking id – exception handled



8. Transaction success



9. Notification in Kafka