Hotel Room Booking Application

Overview:

The project ‘Hotel Room Booking Application’ is used to book rooms for a single hotel. In this project, I have created a microservices based backend application using spring boot framework.

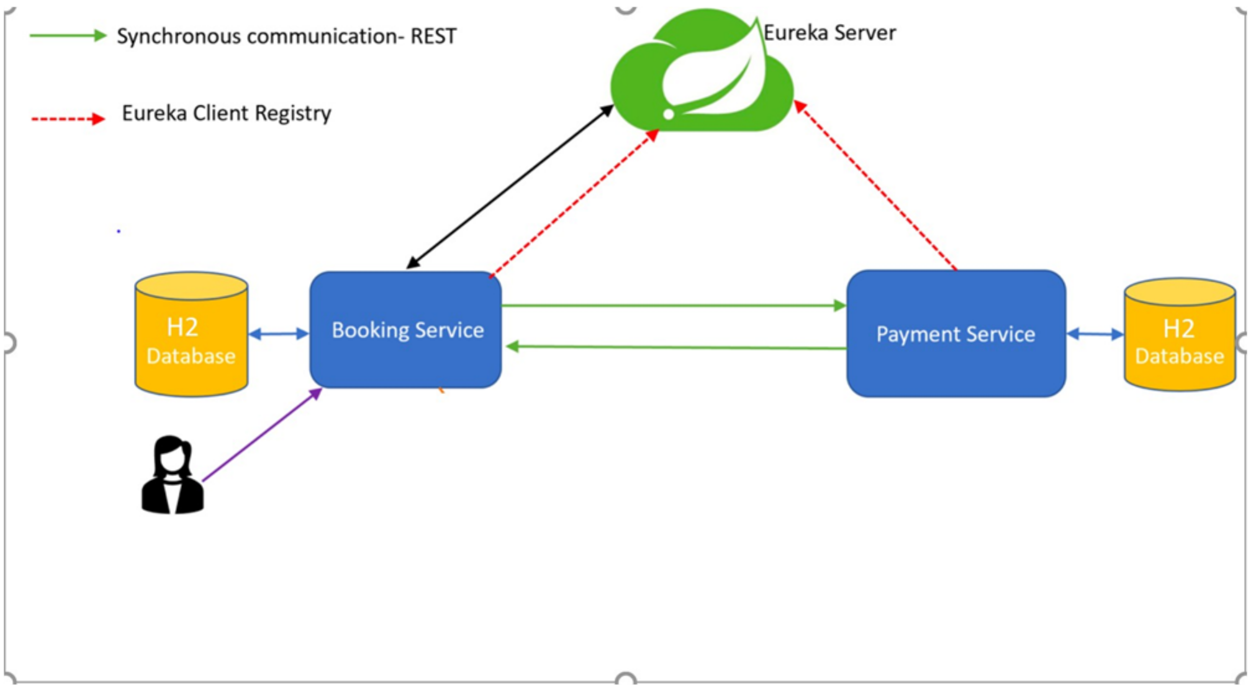
The project is divided into two different microservices, which are as follows:

Booking service
This service is exposed to the outer world and is responsible for collecting all information related to user booking.

Payment service
This is a dummy payment service; this service is called by the booking service for initiating payment after confirming rooms.

Application Workflow:

Initially, the Booking service and Payment service register themselves on the Eureka server.
The user initiates room booking using the 'Booking' service by providing information such as toDate, fromDate, aadharNumber and number of rooms required (number of Rooms).



The 'Booking' service returns back the list of room numbers and price associated and prompts the user to enter the payment details if they wish to continue ahead with the booking. It also stores the details provided by the user in its database.

If the user wants to go ahead with the booking, then they can simply provide the payment related details like bookingMode, upiId/ cardNumber to the 'Booking' service which will be further sent to the 'Payment' service. Based on this data, the payment service will perform a dummy transaction and return back the transaction Id associated with the transaction to the Booking service. All the information related to transactions is saved in the H2 database of the payment service.

Architecture:

All Services are combined into a single directory called 'Sweet-Home'. Booking service, Payment service and Eureka are a type of Spring boot application.

MySQL & H2 is used for database operations and services for Booking and Payment Service respectively. **Before running please update the username and password for database configuration.**

- Based on Schema definition, Entity class is built.
- Data Access Layer (DAO), a mediator between Database and Service Layer, through Data Transfer Objects (DTO), DAO is defined as a repository into the project.
- Service Layer acts as an interface between Controller and DAO.
- Controller layer interacts with presentation layer, through HTTP Requests

Details of Services and APIs:

Booking Service:

This service is responsible for taking input from users like- toDate, fromDate, aadharNumber and the number of rooms required (numOfRooms) and save it in its RDS database. This service also generates a random list of room numbers depending on 'numOfRooms' requested by the user and returns the room number list (roomNumbers) and total roomPrice to the user.

If the user wishes to go ahead with the booking, they can provide the payment-related details like paymentMode, upiId / cardNumber, which will be further sent to the payment service to retrieve the transactionId.

API 1:

This endpoint is responsible for collecting information like fromDate, toDate, aadharNumber, numOfRooms from the user and save it in its database.

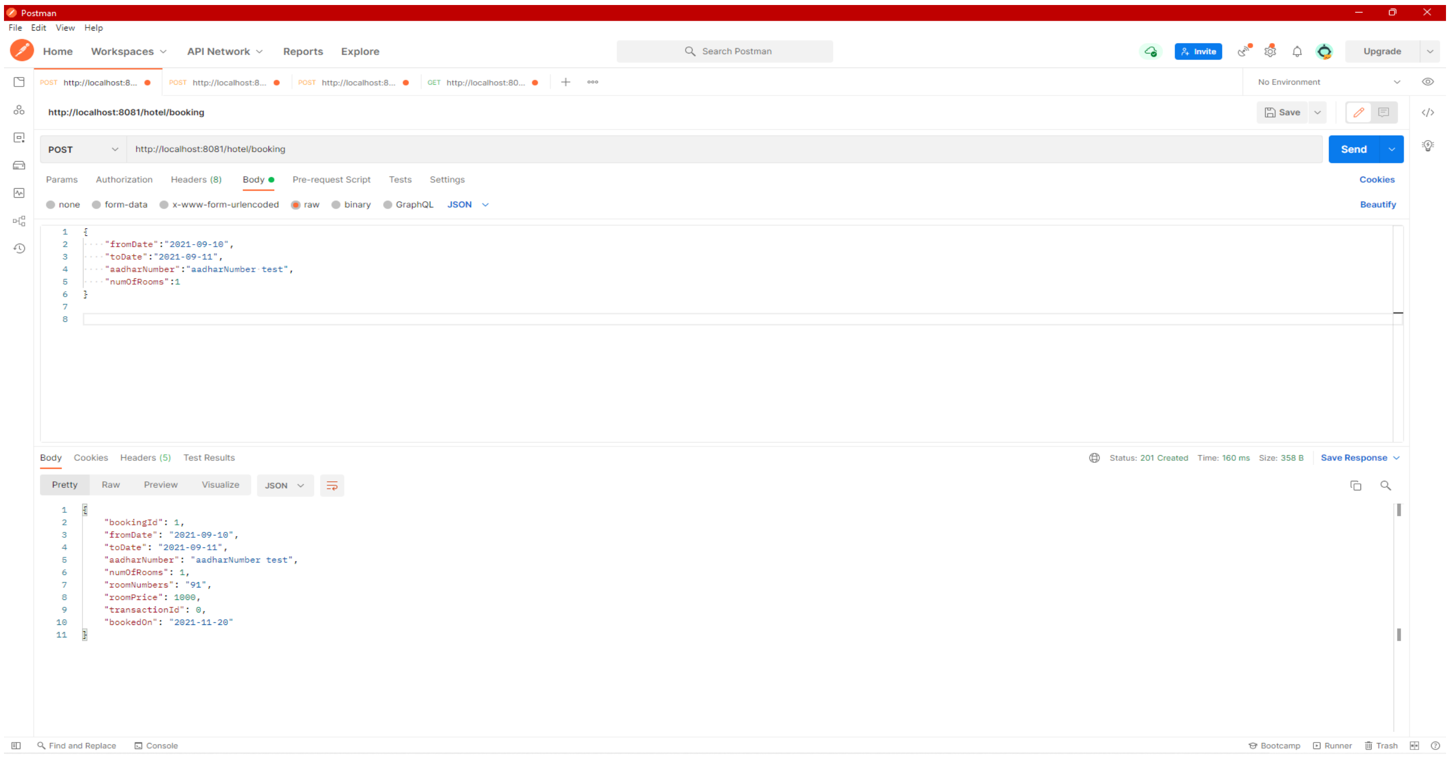
Endpoint/URLt: http://localhost:8081/hotel/booking

HTTP METHOD: POST

RequestBody: fromDate, toDate,aadharNumber,numOfRooms

Response Status: Created

Response: ResponseEntity<BookingInfoEntity>



POST http://localhost:8081/hotel/booking

```
{
    "fromDate":"2021-09-10",
    "toDate":"2021-09-11",
    "aadharNumber":"aadharNumber test",
    "numOfRooms":1
}
```

Status: 201 Created Time: 160 ms Size: 358 B

```
{
    "bookingId": 1,
    "fromDate": "2021-09-10",
    "toDate": "2021-09-11",
    "aadharNumber": "aadharNumber test",
    "numOfRooms": 1,
    "roomNumbers": "91",
    "roomPrice": 1000,
    "transactionId": 0,
    "bookedOn": "2021-11-20"
}
```

API 2:

This endpoint is responsible for taking the payment-related details from the user and sending it to the payment service. It gets the transactionId from the Payment service in response and saves it in the booking

table.

Endpoint/URL: http://localhost:8081/hotel/booking/1/transaction

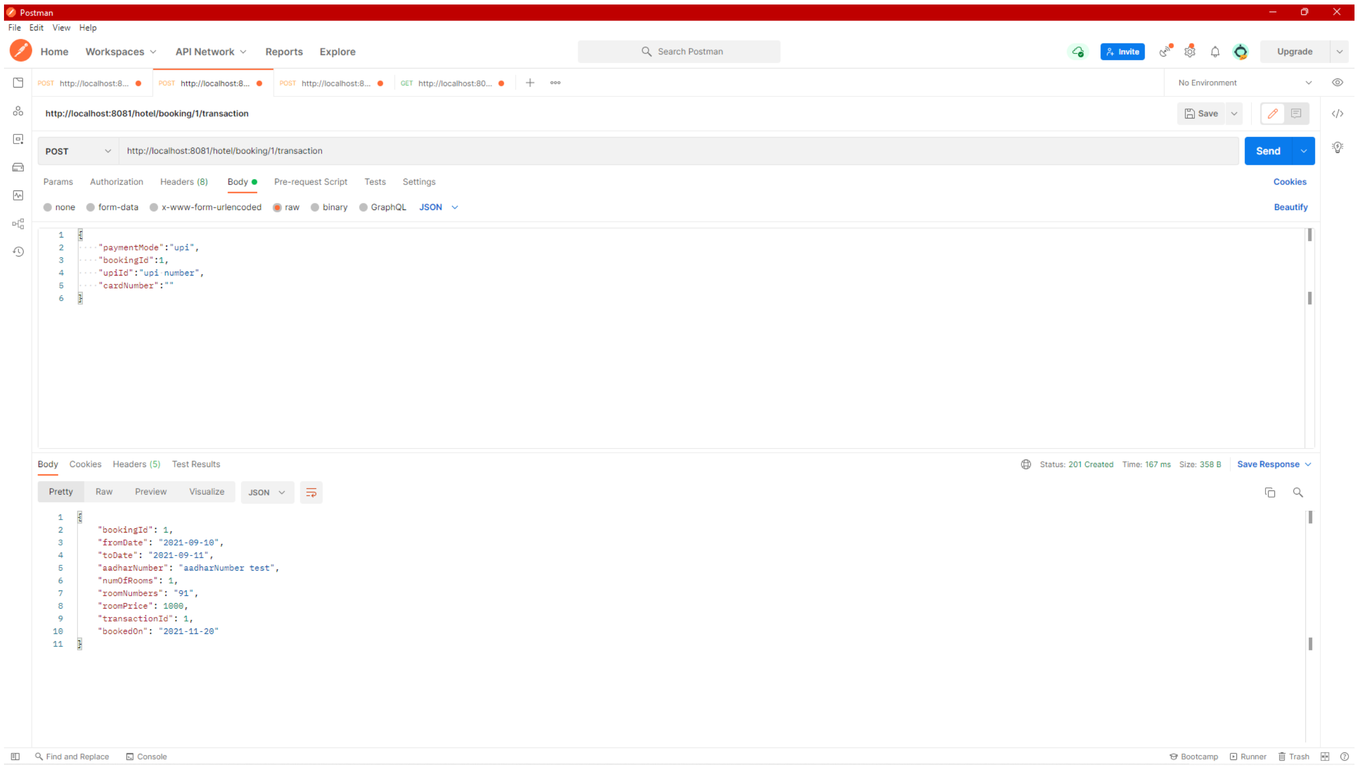
HTTP METHOD: POST

PathVaraiable: int

RequestBody: paymentMode, bookingId,upiId,cardNumber

Response Status: Created

Response: ResponseEntity<BookingInfoEntity>


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite Upgrade
No Environment
Save
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (5) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console
Bootcamp Runner Trash

http://localhost:8081/hotel/booking/1/transaction

POST http://localhost:8081/hotel/booking/1/transaction

```
{
    "paymentMode":"upi",
    "bookingId":1,
    "upiId":"upi number",
    "cardNumber":""
}
```

Status: 201 Created Time: 167 ms Size: 358 B

```
{
    "bookingId": 1,
    "fromDate": "2021-09-10",
    "toDate": "2021-09-11",
    "aadharNumber": "aadharNumber test",
    "numOfRooms": 1,
    "roomNumbers": "91",
    "roomPrice": 1000,
    "transactionId": 1,
    "bookedOn": "2021-11-20"
}
```

Payment Service:

This service is responsible for taking payment-related information- paymentMode, upiId or cardNumber,

bookingId and returns a unique transactionId to the booking service. It saves the data in its RDS database

and returns the transactionId as a response.

API 1:

This endpoint is used to imitate performing a transaction for a particular booking. It takes details such as bookingId, paymentMode, upiId or cardNumber and returns the transactionId automatically generated while storing the details in the 'transaction' table.

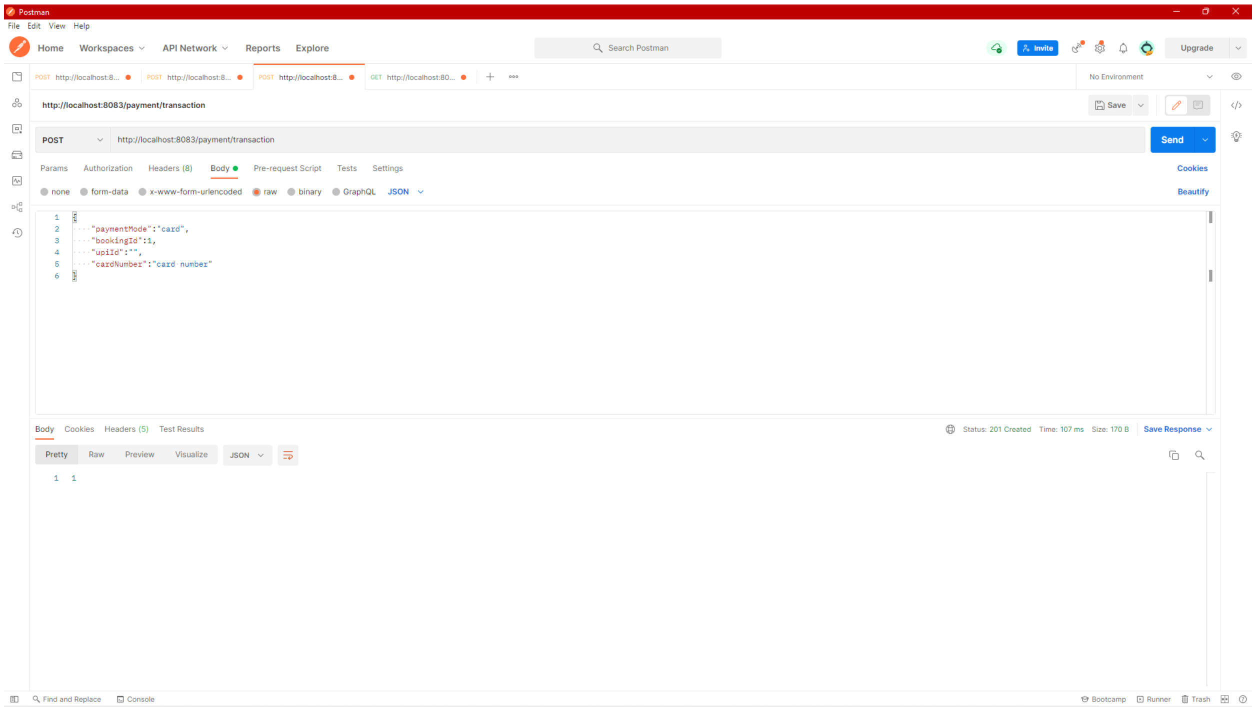
Endpoint/URL : http://localhost:8083/payment/transaction

HTTP METHOD: POST

RequestBody: bookingId, paymentMode, upiId, cardNumber

Response Status: Created

Response: ResponseEntity<transactionId>


POST http://localhost:8083/payment/transaction

```
{
    "paymentMode":"card",
    "bookingId":1,
    "upiId":"",
    "cardNumber":"card number"
}
```

Status: 201 Created Time: 107 ms Size: 170 B

```
1
```

API 2:

This endpoint presents the transaction details to the user based on the transactionId provided by the user.

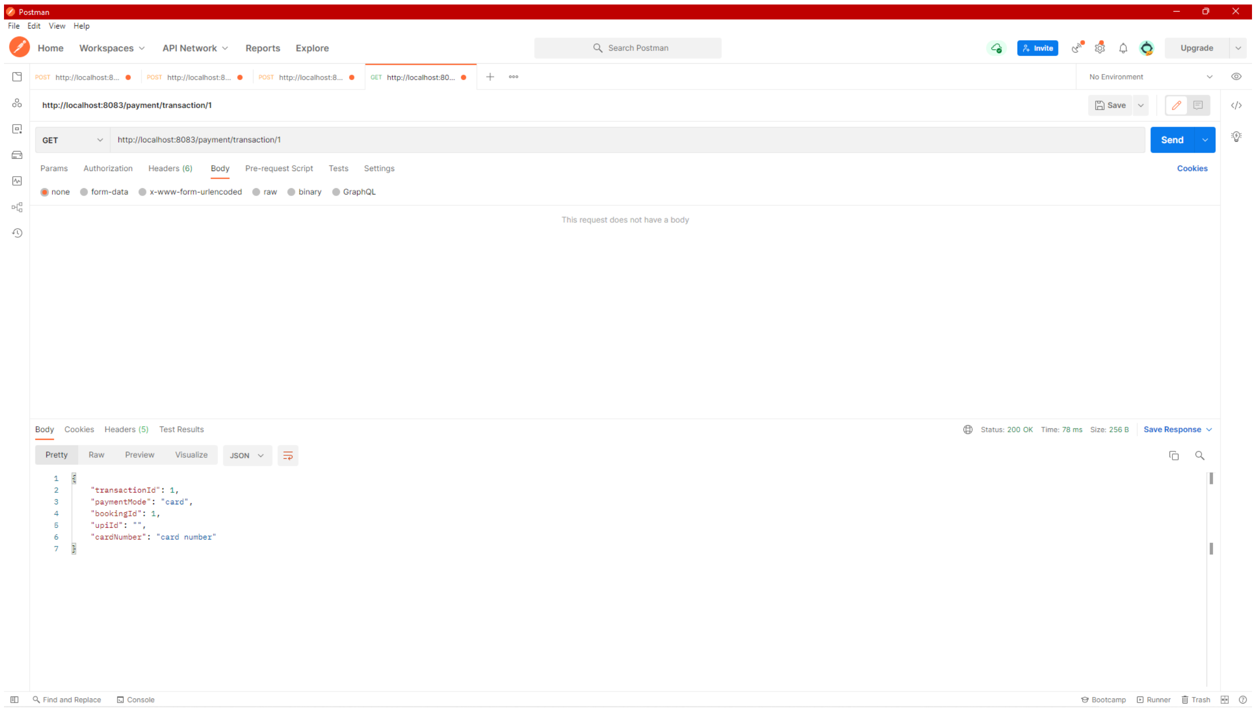
Endpoint/URL : http://localhost:8083/payment/transaction/1

HTTP METHOD: GET

RequestBody: (PathVaraiable) int

Response Status: OK

Response: ResponseEntity<TransactionDetailsEntity>



GET http://localhost:8083/payment/transaction/1

Body: none — This request does not have a body

Status: 200 OK Time: 78 ms Size: 256 B

```
{
    "transactionId": 1,
    "paymentMode": "card",
    "bookingId": 1,
    "upiId": "",
    "cardNumber": "card number"
}
```