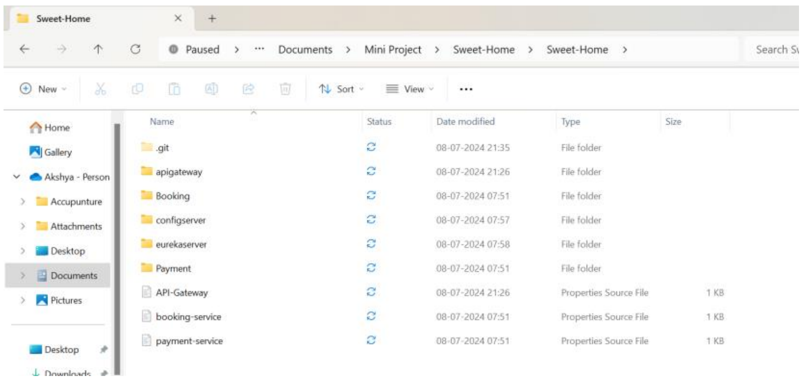
Hotel Booking Application

Created a Hotel Booking Application with 2 different microservices Booking & Payment services registered with Eureka server and using API-Gateway for handling multiple API's.

Project Folder Structure:

Swee-Home (Main folder)



Sweet-Home

Paused > ··· Documents > Mini Project > Sweet-Home > Sweet-Home >

Name	Status	Date modified	Type	Size
.git	⟳	08-07-2024 21:35	File folder	
apigateway	⟳	08-07-2024 21:26	File folder	
Booking	⟳	08-07-2024 07:51	File folder	
configserver	⟳	08-07-2024 07:57	File folder	
eurekaserver	⟳	08-07-2024 07:58	File folder	
Payment	⟳	08-07-2024 07:51	File folder	
API-Gateway	⟳	08-07-2024 21:26	Properties Source File	1 KB
booking-service	⟳	08-07-2024 07:51	Properties Source File	1 KB
payment-service	⟳	08-07-2024 07:51	Properties Source File	1 KB

Sub folders:

- ➔ configserver
- ➔ eurekaserver
- ➔ api-gateway
- ➔ Booking (microservice application)
- ➔ Payment (microservice application)

Let's see the Overview of each application,

Configserver:

In this service we will define the Port information, URI of files in Git repository.

Eurekaserver:

In Eureka server application we will define port number where Eureka server will be running.

API – Gateway:

In Api-Gateway application we will be importing Config server properties.

Booking Application:

Controller layer: Booking Controller

Created 3 methods in Controller:

Create method:

Creates a new booking. Maps the incoming BookingDTO to a BookingInfoEntity, processes it using the booking service, maps the saved entity back to a BookingDTO, and returns it with HTTP status CREATED.

getBookingDetails method:

Retrieves booking details based on the booking ID.Maps the retrieved BookingInfoEntity to a BookingDTO and returns it with HTTP status OK.

createTransaction method:

Describes the process of creating a new transaction for a specific booking, including sending payment details to the payment service, retrieving the transaction ID, updating the booking with the transaction ID, handling exceptions, and returning the updated booking details with HTTP status CREATED.

DAO:

Extends JPA repository interface with BookingInfoEntity class.

DTO:

Has declaration of parameters like: bookingId, fromDate, toDate, aadharNumber, numOfRooms, roomPrice, transactionId, bookedOn.

Entities:

Creating table "booking" with Above parameters(Mentioned in DTO) as per Schema.

Service layer: BookingServiceImpl

Service layer has 4 methods they are:

acceptBooking: Describes the process of accepting and processing a booking, including generating random room numbers, validating the booking dates, calculating the room price, setting the booking date, and saving the booking to the database.

getBookingDetails: Describes the process of retrieving booking details based on the booking ID, including handling invalid booking IDs.

getRandomNumbers: Describes the process of generating a list of random room numbers.

calculateRoomPrice: Describes the process of calculating the room price based on the number of rooms and the number of days.

Exception Handling:

CustomExceptionhandler:

Handles exception when there is Invalid Argument passed in request.

Main class – BookingApplication class:

Starter of the Booking microservice and defines bean for ModelMapper and RestTemplate.

<u>Payment Application:</u>

Controller layer: PaymentController

Payment Controller consists of 2 methods they are:

createTransaction:

Describes the process of creating a new transaction, including converting the incoming PaymentDTO to TransactionDetailsEntity, processing it using the payment service, and returning the transaction ID with HTTP status CREATED.

getTransaction:

Describes the process of retrieving transaction details based on the transaction ID, including converting the retrieved TransactionDetailsEntity to PaymentDTO, handling invalid transaction IDs, and returning the payment details with HTTP status OK.

DTO:

BookingDTO:

Same as the one in Booking Application.

PaymentDTO:

Has declaration of parameters like transactionId, paymentMode, bookingId,upiId,cardNumber

DAO:

Extends JPA repository interface with TransactionDetailsEntity class.

Entity:

TransactionDetailsEntity:

Creating table “transaction” with above parameters in PaymentDTO as per Schema given.

Exception Handling:

CustomExceptionhandler:

Handles exception when there is Invalid Argument passed in request.

Service layer: PaymentServiceImpl

Service layer consists 2 methods they are:

acceptPaymentDetails:

Describes the purpose and process of accepting and processing payment details.

getTransactionDetails:

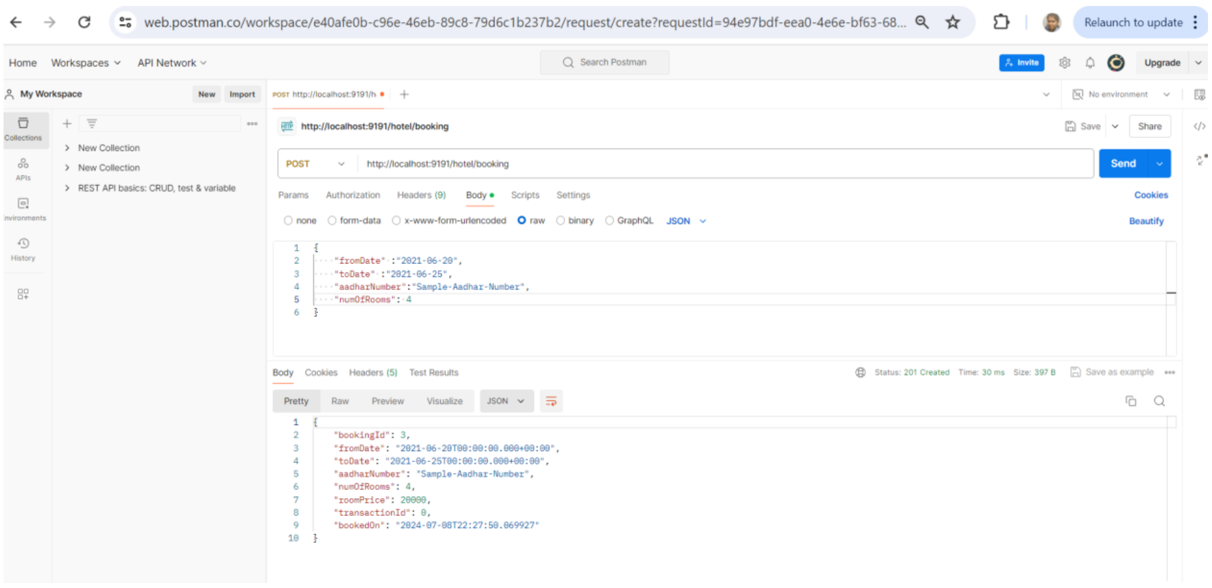
Describes the purpose and process of retrieving transaction details by ID, including handling invalid IDs.

Main Class – PaymentApplication class:

Starter of the Payment microservice and defines bean for ModelMapper and RestTemplate.

Screenshots of API calls tested:

Booking Application:



web.postman.co/workspace/e40afe0b-c96e-46eb-89c8-79d6c1b237b2/request/create?requestId=94e97bdf-eea0-4e6e-bf63-68...

Relaunch to update

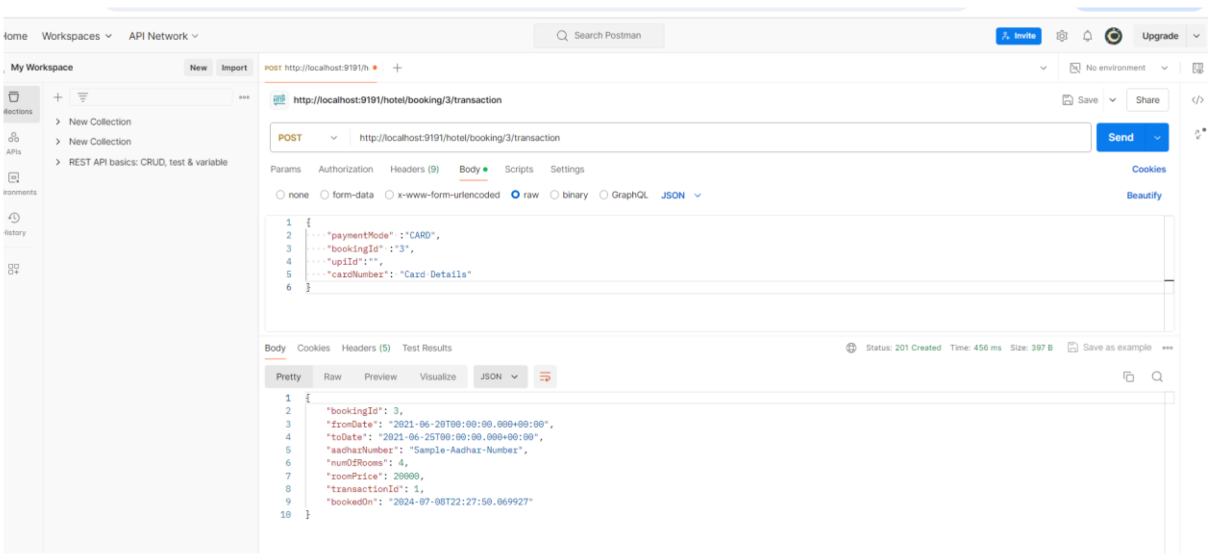
POST http://localhost:9191/hotel/booking

Body: raw JSON

```
{
    "fromDate" :"2021-06-20",
    "toDate" :"2021-06-25",
    "aadharNumber":"Sample-Aadhar-Number",
    "numOfRooms": 4
}
```

Status: 201 Created Time: 30 ms Size: 397 B

```
{
    "bookingId": 3,
    "fromDate": "2021-06-20T00:00:00.000+00:00",
    "toDate": "2021-06-25T00:00:00.000+00:00",
    "aadharNumber": "Sample-Aadhar-Number",
    "numOfRooms": 4,
    "roomPrice": 20000,
    "transactionId": 0,
    "bookedOn": "2024-07-08T22:27:50.069927"
}
```



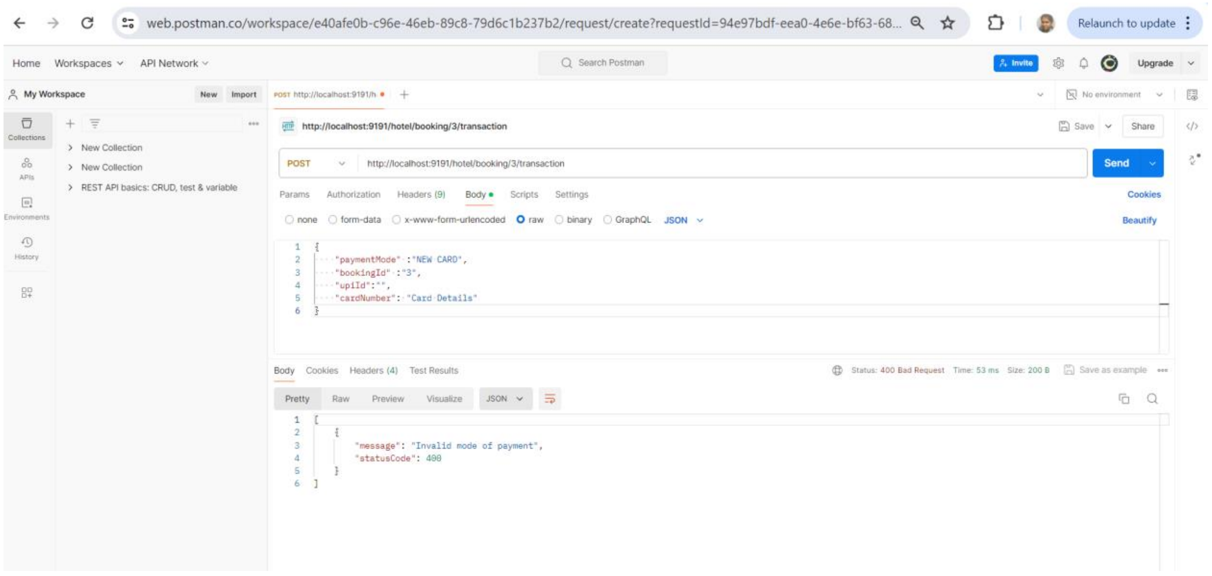
POST http://localhost:9191/hotel/booking/3/transaction

Body: raw JSON

```
{
    "paymentMode" :"CARD",
    "bookingId" :"3",
    "upiId":"",
    "cardNumber": "Card Details"
}
```

Status: 201 Created Time: 456 ms Size: 397 B

```
{
    "bookingId": 3,
    "fromDate": "2021-06-20T00:00:00.000+00:00",
    "toDate": "2021-06-25T00:00:00.000+00:00",
    "aadharNumber": "Sample-Aadhar-Number",
    "numOfRooms": 4,
    "roomPrice": 20000,
    "transactionId": 1,
    "bookedOn": "2024-07-08T22:27:50.069927"
}
```



web.postman.co/workspace/e40afe0b-c96e-46eb-89c8-79d6c1b237b2/request/create?requestId=94e97bdf-eea0-4e6e-bf63-68...

Relaunch to update

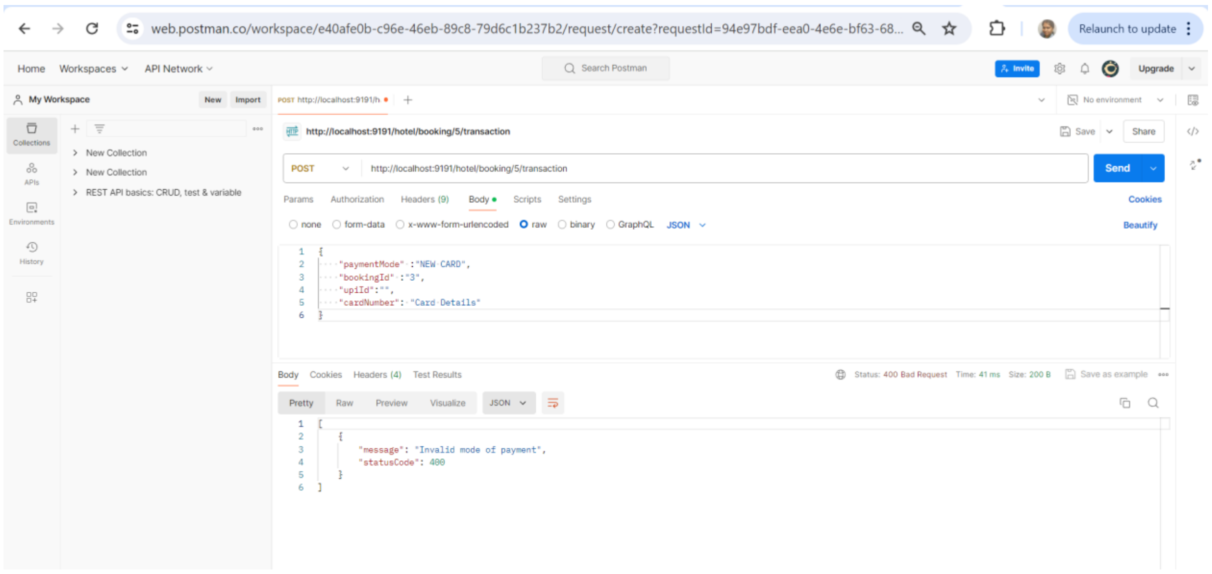
POST http://localhost:9191/hotel/booking/3/transaction

Body: raw JSON

```
{
    "paymentMode" :"NEW CARD",
    "bookingId" :"3",
    "upiId":"",
    "cardNumber": "Card Details"
}
```

Status: 400 Bad Request Time: 53 ms Size: 200 B

```
[
    {
        "message": "Invalid mode of payment",
        "statusCode": 400
    }
]
```


web.postman.co/workspace/e40afe0b-c96e-46eb-89c8-79d6c1b237b2/request/create?requestId=94e97bdf-eea0-4e6e-bf63-68...

POST http://localhost:9191/hotel/booking/5/transaction

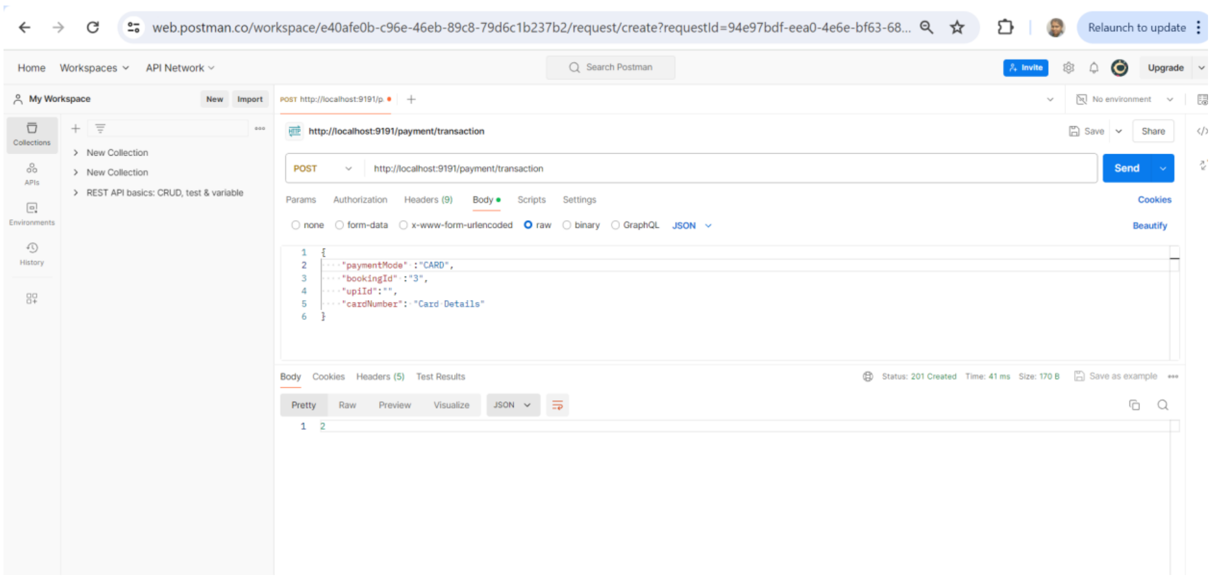
Body: raw JSON

```
{
    "paymentMode" :"NEW CARD",
    "bookingId" :"3",
    "upiId":"",
    "cardNumber": "Card Details"
}
```

Status: 400 Bad Request Time: 41 ms Size: 200 B

```
[
    {
        "message": "Invalid mode of payment",
        "statusCode": 400
    }
]
```

Payment Application:


web.postman.co/workspace/e40afe0b-c96e-46eb-89c8-79d6c1b237b2/request/create?requestId=94e97bdf-eea0-4e6e-bf63-68...

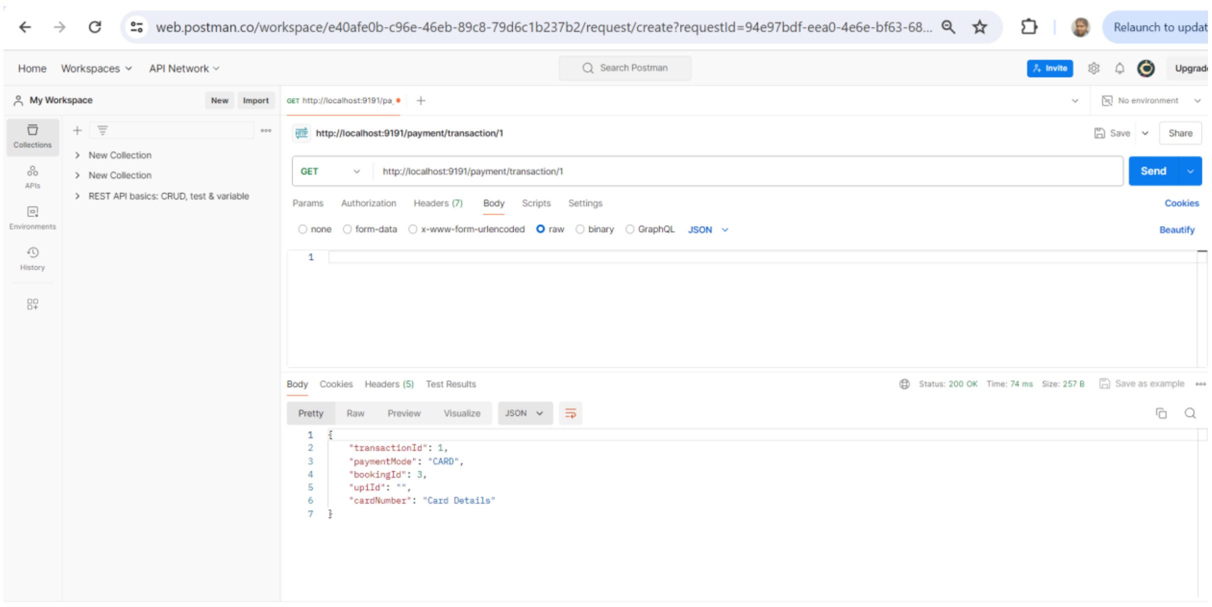
POST http://localhost:9191/payment/transaction

Body: raw JSON

```
{
    "paymentMode" :"CARD",
    "bookingId" :"3",
    "upiId":"",
    "cardNumber": "Card Details"
}
```

Status: 201 Created Time: 41 ms Size: 170 B

```
2
```


web.postman.co/workspace/e40afe0b-c96e-46eb-89c8-79d6c1b237b2/request/create?requestId=94e97bdf-eea0-4e6e-bf63-68...

GET http://localhost:9191/payment/transaction/1

Status: 200 OK Time: 74 ms Size: 257 B

```
{
    "transactionId": 1,
    "paymentMode": "CARD",
    "bookingId": 3,
    "upiId": "",
    "cardNumber": "Card Details"
}
```