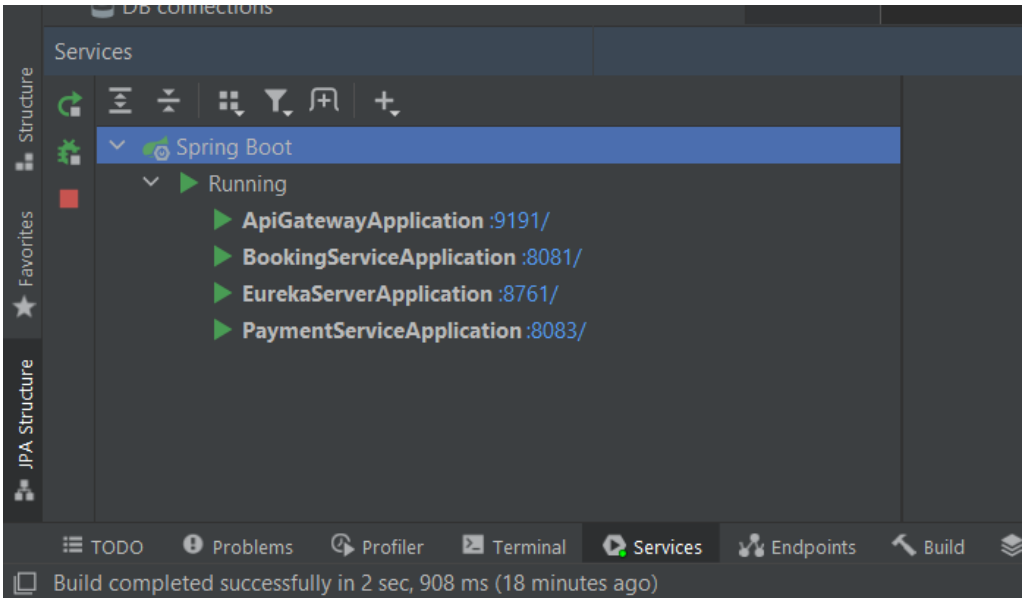
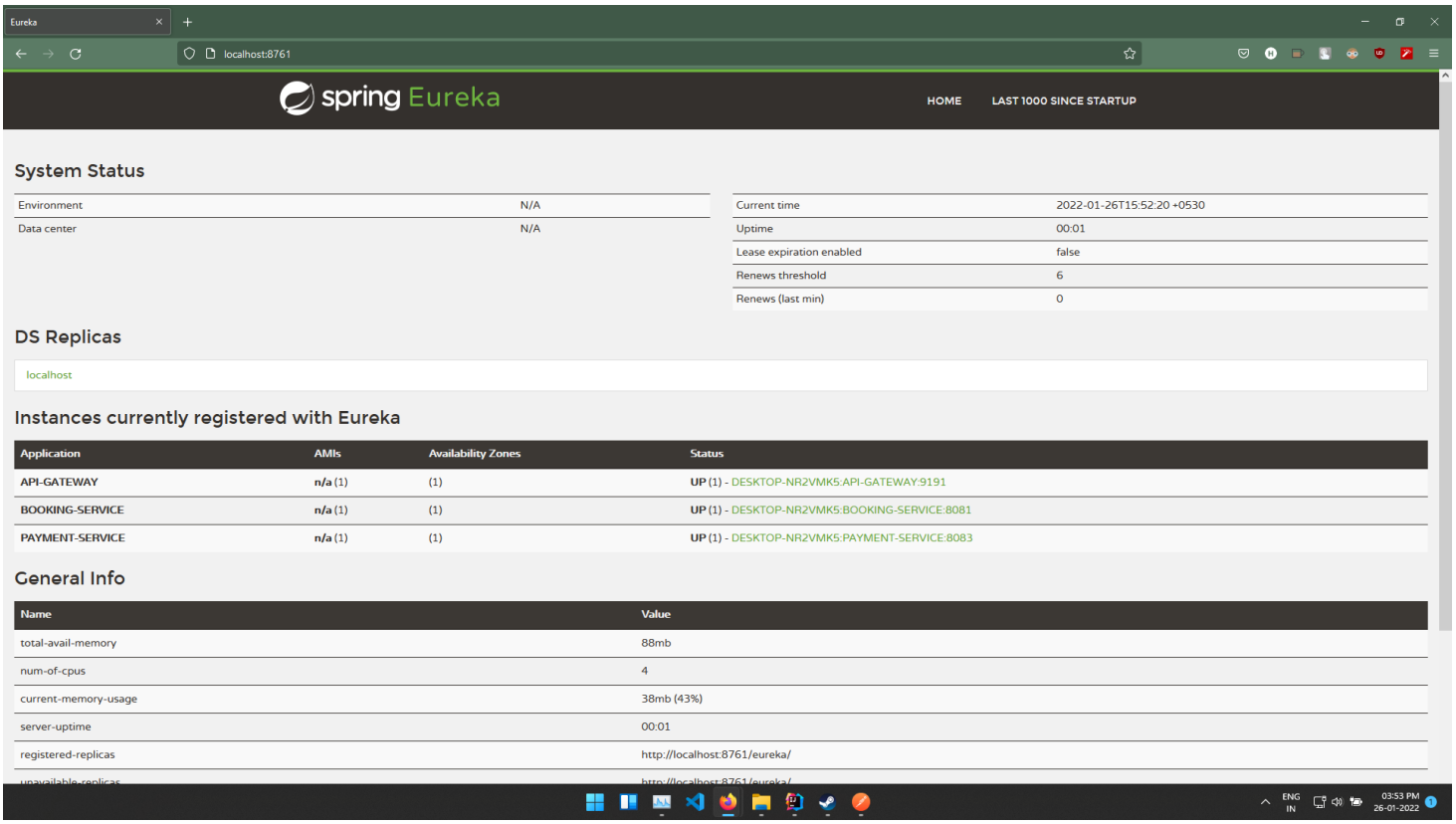
Harjot Singh
harjotsrance@gmail.com
PGDSD Mar'21

Ports -

Booking Service : 8081
Payment Service : 8083
API Gateway : 9191
Eureka Server : 8761





Eureka Server after running all the services in the project

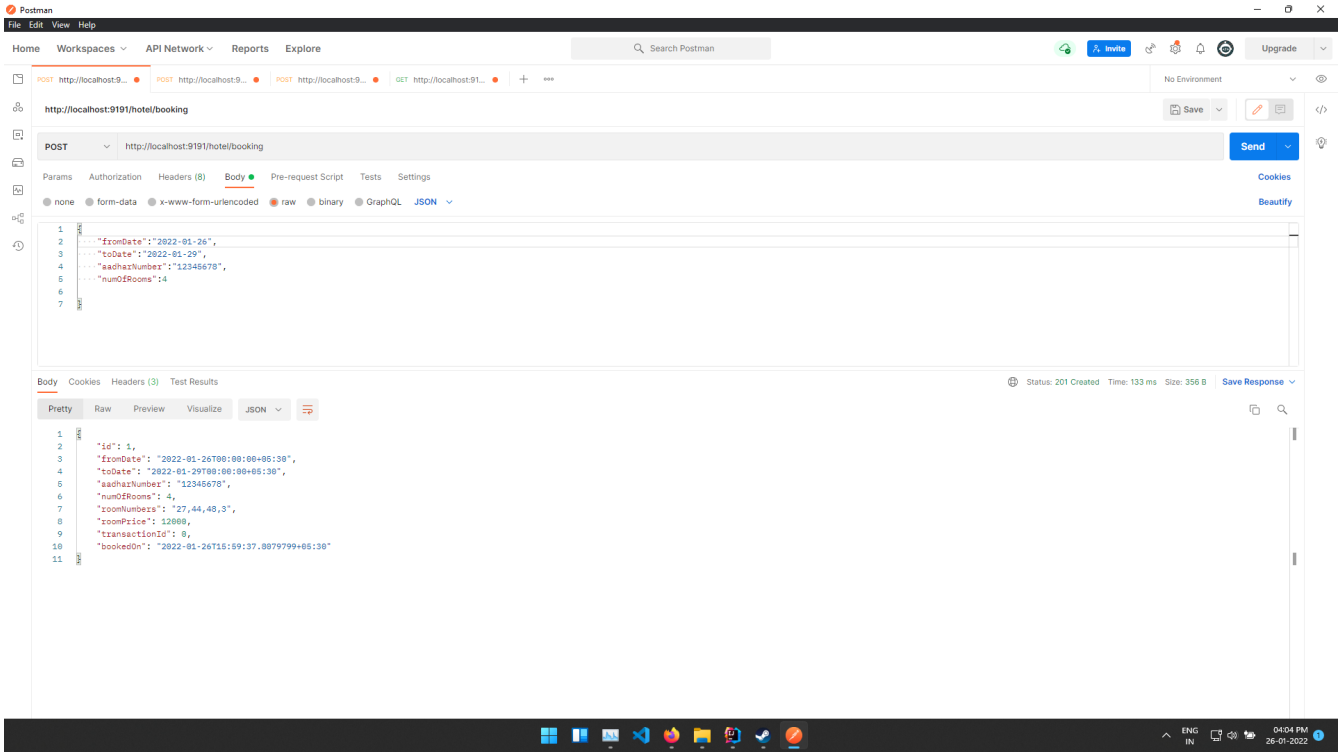
Application Properties

```
rvice)  .gitignore  service_registry.iml  BookingServiceImpl.java  application.properties
spring.application.name=BOOKING-SERVICE

spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.userName=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
logging.level.root=DEBUG
spring.h2.console.enabled=true

server.port=8081

paymentService.url=http://localhost:9191/payment/transaction
```

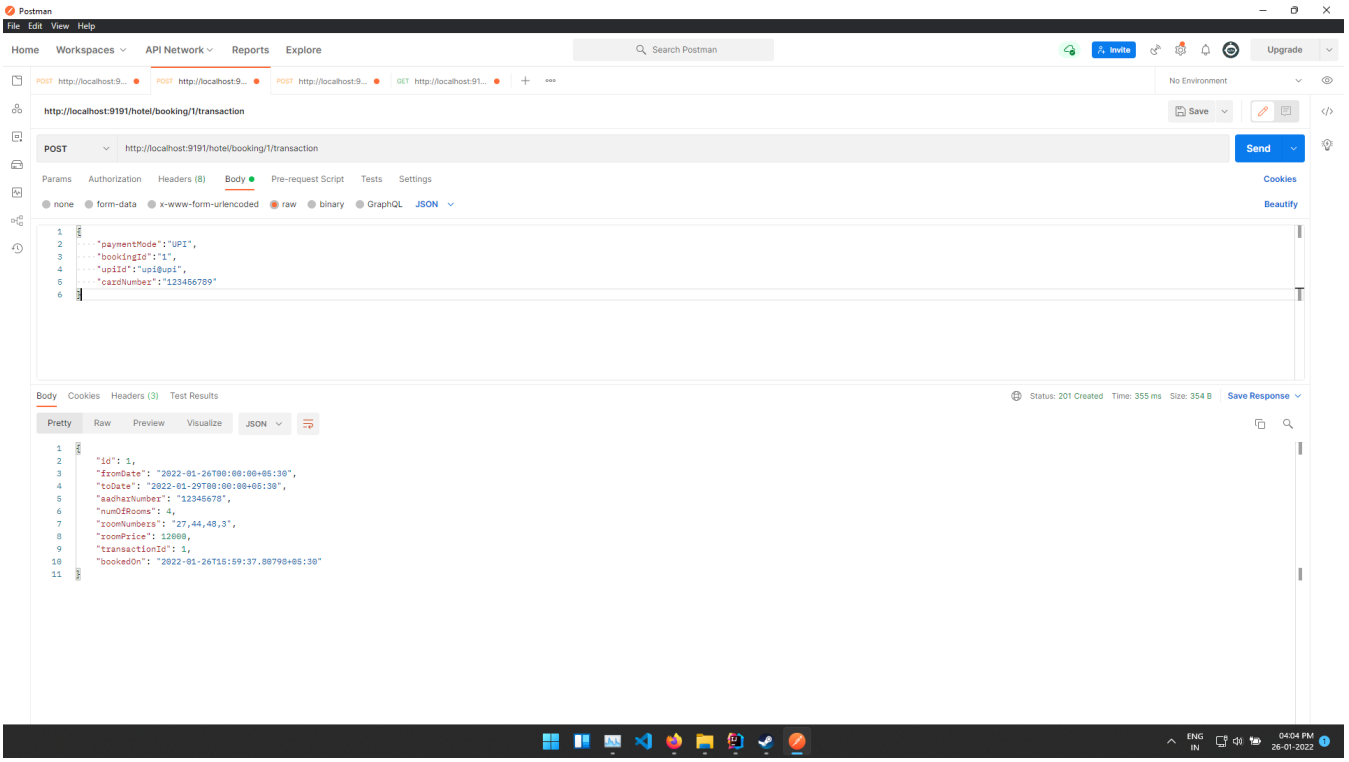
```
registry.iml  PaymentService\...\application.properties  BookingServiceIm
spring.application.name = PAYMENT-SERVICE
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.userName=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create

logging.level.root=DEBUG

spring.h2.console.enabled=true

server.port=8083
```

Postman API Responses

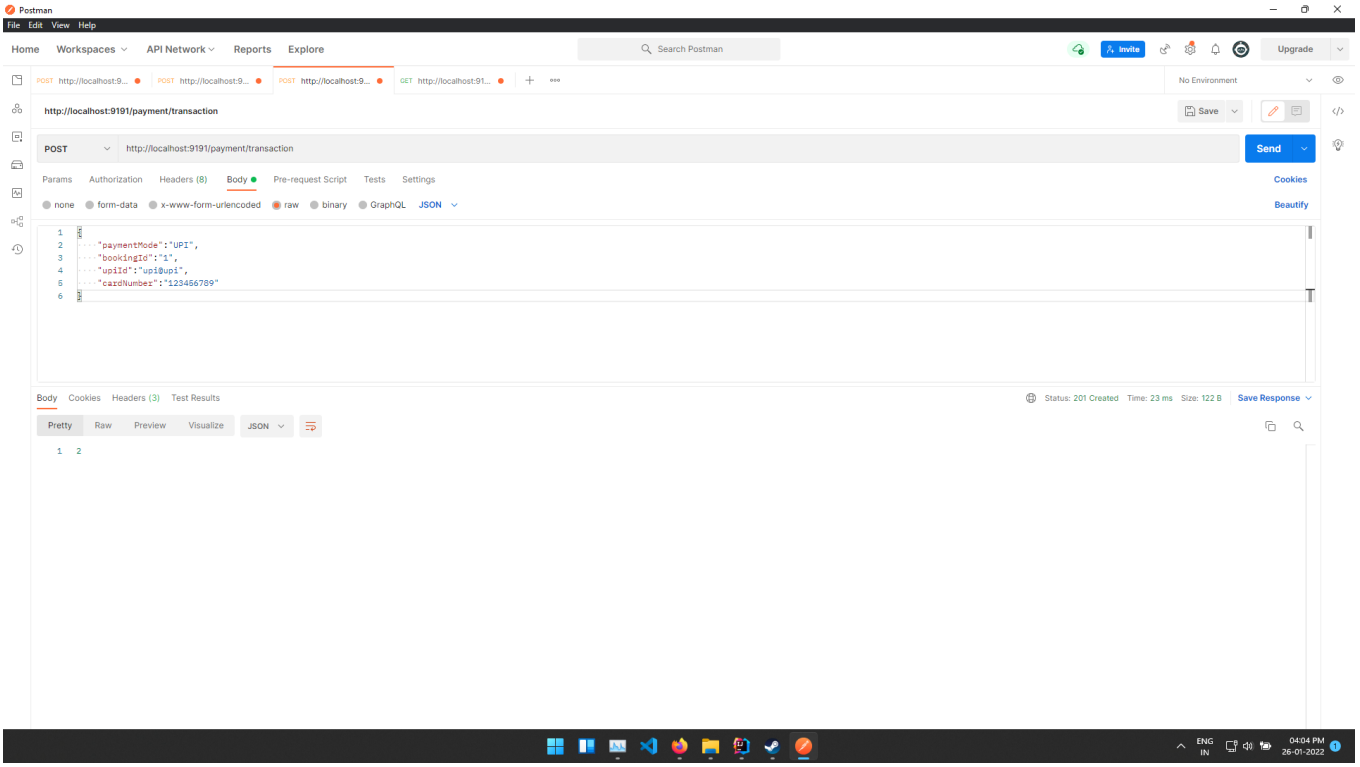
1



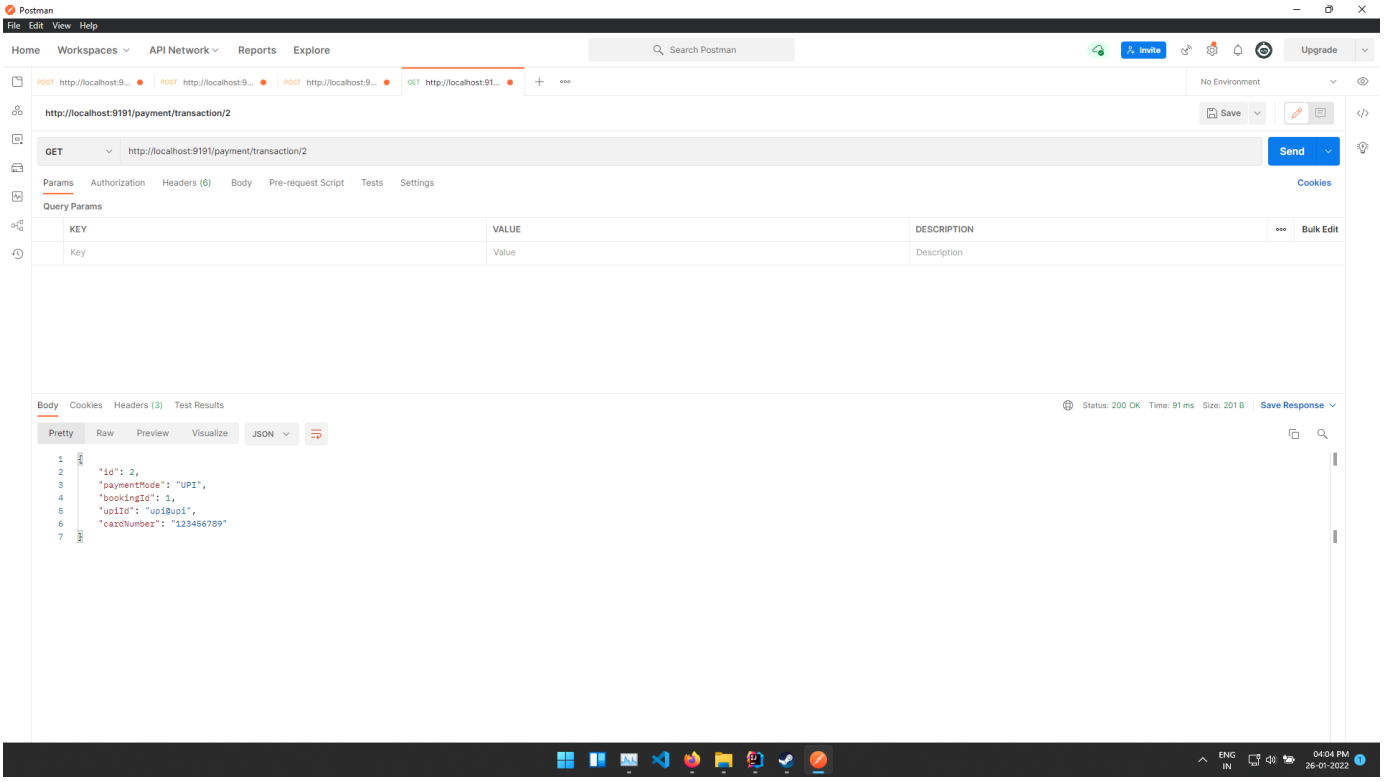
2



3


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
No Environment
http://localhost:9191/payment/transaction
Save
POST
http://localhost:9191/payment/transaction
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"paymentMode":"UPI",
"bookingId":"1",
"upiId":"upi@upi",
"cardNumber":"123456789"
}
Body Cookies Headers (3) Test Results
Status: 201 Created Time: 23 ms Size: 122 B
Save Response
Pretty Raw Preview Visualize JSON
2
ENG IN
04:04 PM
26-01-2022

4


Postman
File Edit View Help
Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
No Environment
http://localhost:9191/payment/transaction/2
Save
GET
http://localhost:9191/payment/transaction/2
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (3) Test Results
Status: 200 OK Time: 91 ms Size: 201 B
Save Response
Pretty Raw Preview Visualize JSON
{
"id": 2,
"paymentMode": "UPI",
"bookingId": 1,
"upiId": "upi@upi",
"cardNumber": "123456789"
}
ENG IN
04:04 PM
26-01-2022

Database Config

By default, Spring Boot configures the application to connect to an in-memory store with the username sa and an empty password. The in-memory database is volatile, and data will be lost when we restart the application.

Configuration in Application.properties file

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.userName=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
```

Eureka Configuration

For Eureka Clients (BookingService, PaymentService, API Gateway in application.yml file) eureka:

```
    client:
        register-with-eureka: true fetch-registry:
        true service-url:

            defaultZone: http://localhost:8761/eureka/

    instance:
                hostname: localhost
```

Eureka Server (in application.yml file)

```
server:
    port: 8761

eureka:
    client:
        register-with-eureka: false fetch-registry:
        false
```

API Gateway Configuration

```yaml
server:
  port: 9191

spring:
  application:
    name: API-GATEWAY
  cloud:
    gateway:
      routes:
        - id: BOOKING-SERVICE
          uri: lb://BOOKING-SERVICE
          predicates:
            - Path=/hotel/**

        - id: PAYMENT-SERVICE
          uri: lb://PAYMENT-SERVICE
          predicates:
            - Path=/payment/**

    discovery:
      enabled: true
```

Structure of Booking Service

Controller @RequestMapping(value = "/hotel")

Has the following dependencies

```
@Autowired
private BookingService bookingService;
```

This class has two methods for the two endpoints of Booking Service.

1. **bookingDetails** with @PostMapping(value = "/booking")
   invokes bookingService.acceptBookingDetails(DTO)
2. **bookingConfirmation** with @PostMapping(value = "/booking/{id}/transaction")
   invokes bookingService.acceptPaymentDetails(DTO)

The second Controller method handles the following two exceptions -

1. "Invalid mode of payment" : checks if payment mode = "UPI" || "CARD", if true, only then invokes the bookingService. Otherwise sends back appropriate ResponseEntity

2. "Invalid Booking Id" : catches **BookingIdNotPresentException** thrown by bookingService and sends back appropriate ResponseEntity

BookingServiceImpl

Has the following dependencies

```
@Autowired
private BookingInfoDao bookingInfoDao;

@Autowired
private RestTemplate restTemplate;

@Value("${paymentService.url}") //in applications.properties private String
paymentServiceUrl;
```

This class contains the following two utility methods

1. **stringToLocalDate** for converting String to LocalDate
2. **getRandomNumbers** to generate room numbers based on count

This class contains the following two service methods

1. **acceptBookingDetails** fromDate, toDate, aadharNumber, numOfRooms from DTO. Calculates Room Price, Gets random room numbers from getRandomNumbers, sets bookedOn to now(), transactionId is set to 0 by default.

2. **acceptPaymentDetails** uses bookingInfoDao to find the bookingInfoEntity stored in the database. Throws BookingIdNotPresentException if not found. Uses restTemplate to call Payment Service through API Gateway. Sets the transactionId. Prints booking confirmation message on console.

DTO Classes

- Booking DTO : To map request body of endpoint 1

none form-data x-www-form-urlencoded raw

```
{
    "fromDate":"2022-01-26",
    "toDate":"2022-01-29",
    "aadharNumber":"12345678",
    "numOfRooms":4

}
```

Payment DTO : To map request body of endpoint 2

none form-data x-www-form-urlencoded raw

```
{
    "paymentMode":"UPI",
    "bookingId":"1",
    "upiId":"upi@upi",
    "cardNumber":"123456789"
}
```

- Exception DTO : To send back appropriate ResponseEntity in case of Exceptions

Pretty Raw Preview Visualize JSON

```
{
    "message": "Invalid Booking Id",
    "statusCode": 400
}
```

Structure of Payment Service

Controller @RequestMapping(value = "/payment")

Has the following dependency
@Autowired
private PaymentService paymentService;

This class has two methods for the two endpoints of Payment Service.

1. **transactionConfirmation** with @PostMapping(value = "/transaction")
   invokes paymentService.acceptPaymentDetails(DTO)
2. **transactionDetails** with @GetMapping(value = "/transaction/{id}")
   invokes paymentService.getTransactionDetails(id)
   Catches Exception and sends appropriate message if transaction not found.

PaymentServiceImpl

Has the following dependencies -
@Autowired
private TransactionDetailsDao transactionDetailsDao;

This class contains the following two service methods

1. **acceptPaymentDetails** uses DTO to create a new transactionDetailsEntity and saves it in the database using transactionDetailsDao.

2. **getTransactionDetails** uses transactionDetailsDao to find the transactionDetailsEntity.Throws exception if transaction not found.

DTO Class

Payment DTO : To map request body of endpoint 1