Instructions and Walk through of the project Sweet-Home

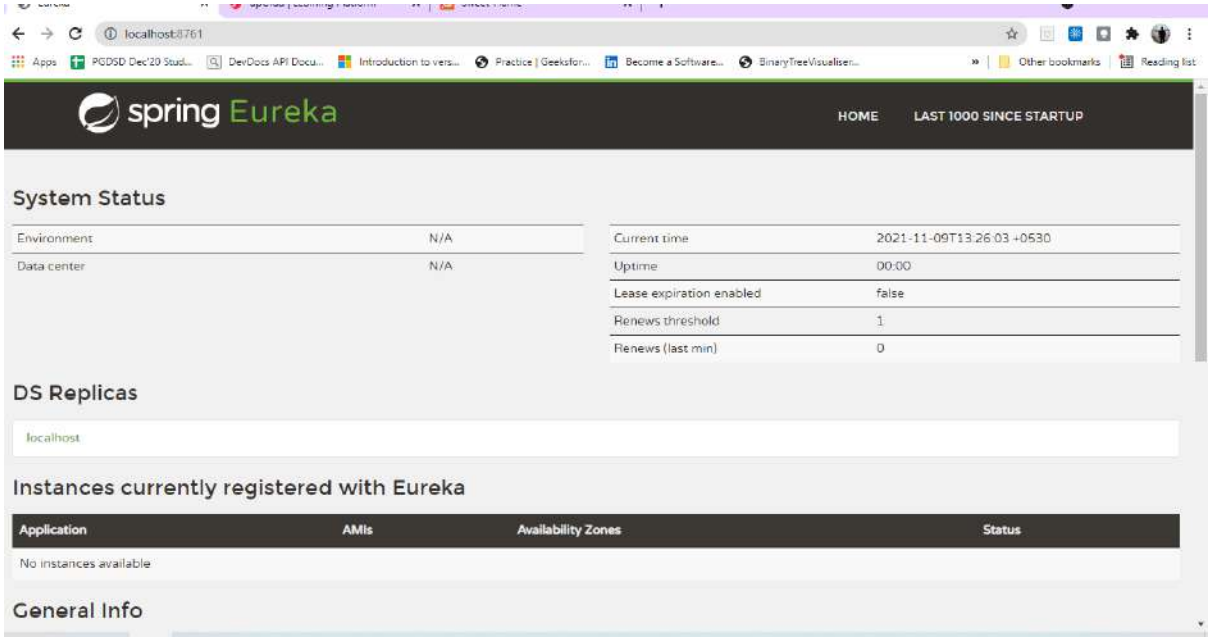
Open each of the projects present in the Sweet-Home folder in separate windows of the IDE

1. Run the Eureka server

EurekaServer → Run the Eureka Server application

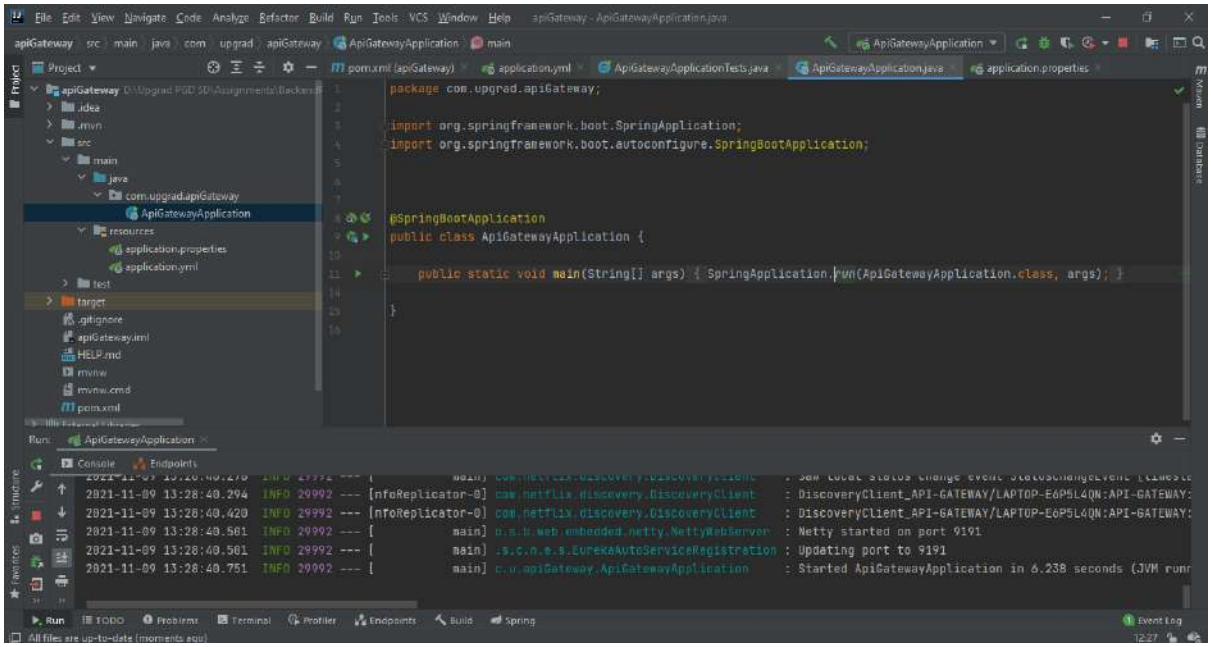
Open http://localhost:8761/

You should see the as below



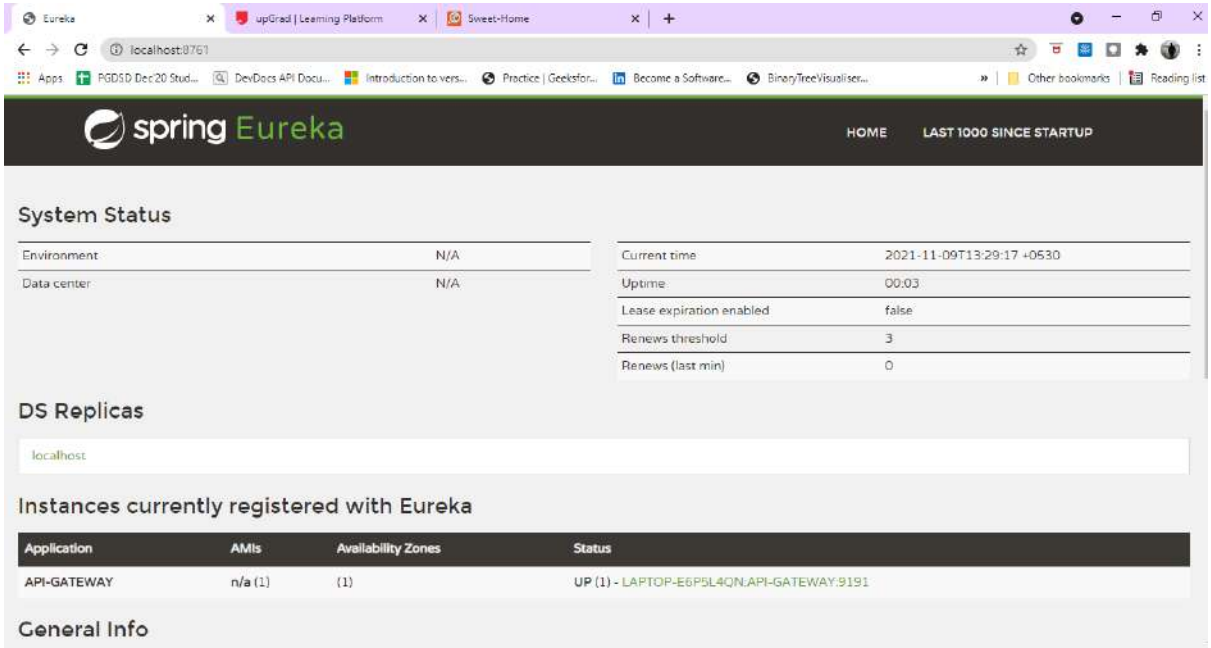
2.

apiGateway → Run the Api gateway application



Refresh the Eureka server URL http://localhost:8761/

You should see the API Gateway instance

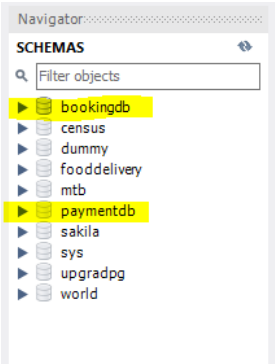


3.Create databases required for the bookingService and Payment service in the MYSQL workbench

Using Commands

create database bookingdb ;

create database paymentdb;



4.

BookingService → Open the proect folder and navigate to the

Path: **src/main/resources/application.properties**

And change the **application.properties** file with appropriate values of

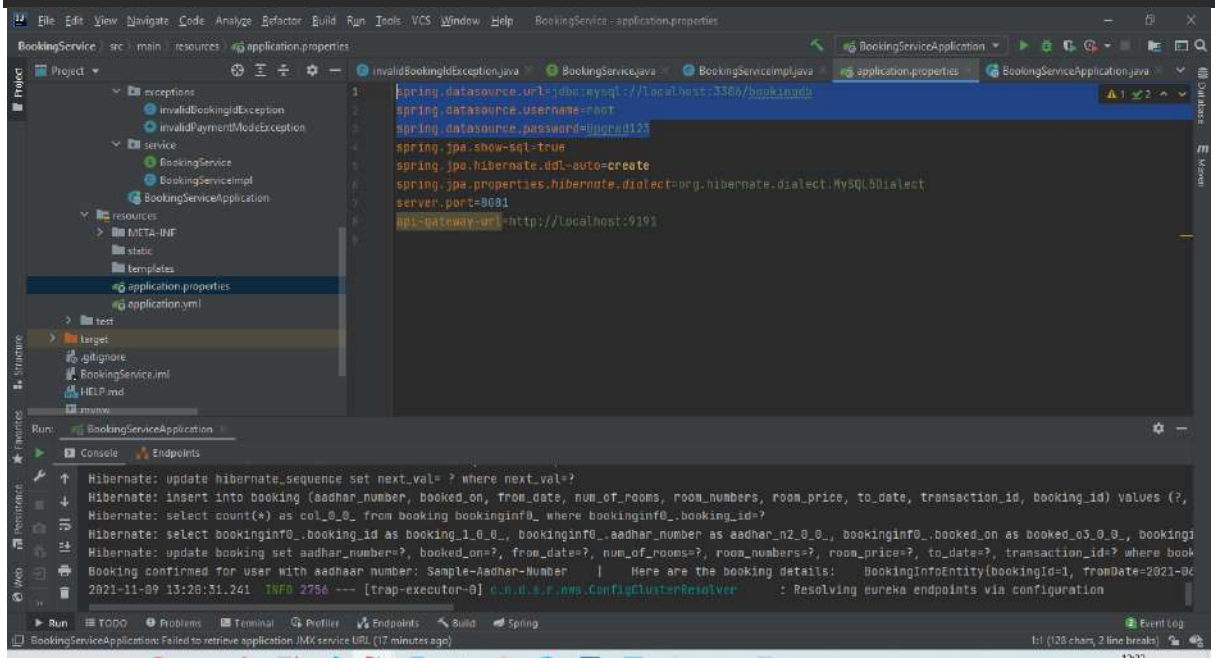
datasource url ,username and password.

In my case the db created was **bookingdb**

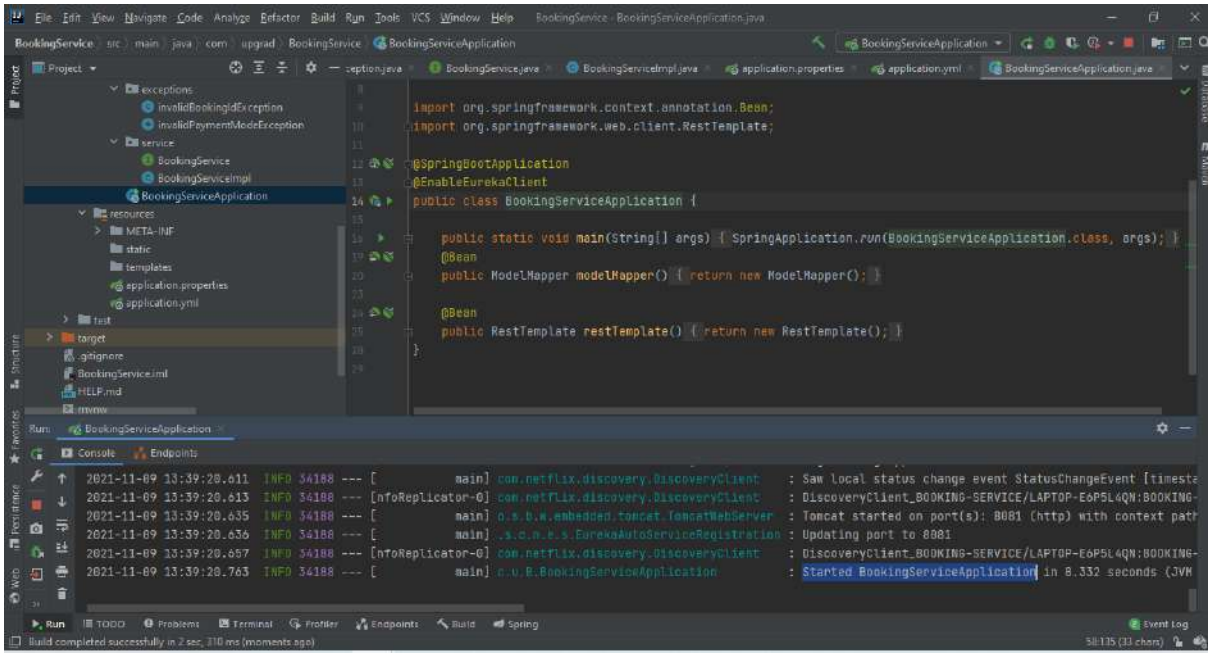
Username is **root**

Password is **Upgrad123**

```
spring.datasource.url=jdbc:mysql://localhost:3306/bookingdb
spring.datasource.username=root
spring.datasource.password=Upgrad123
```

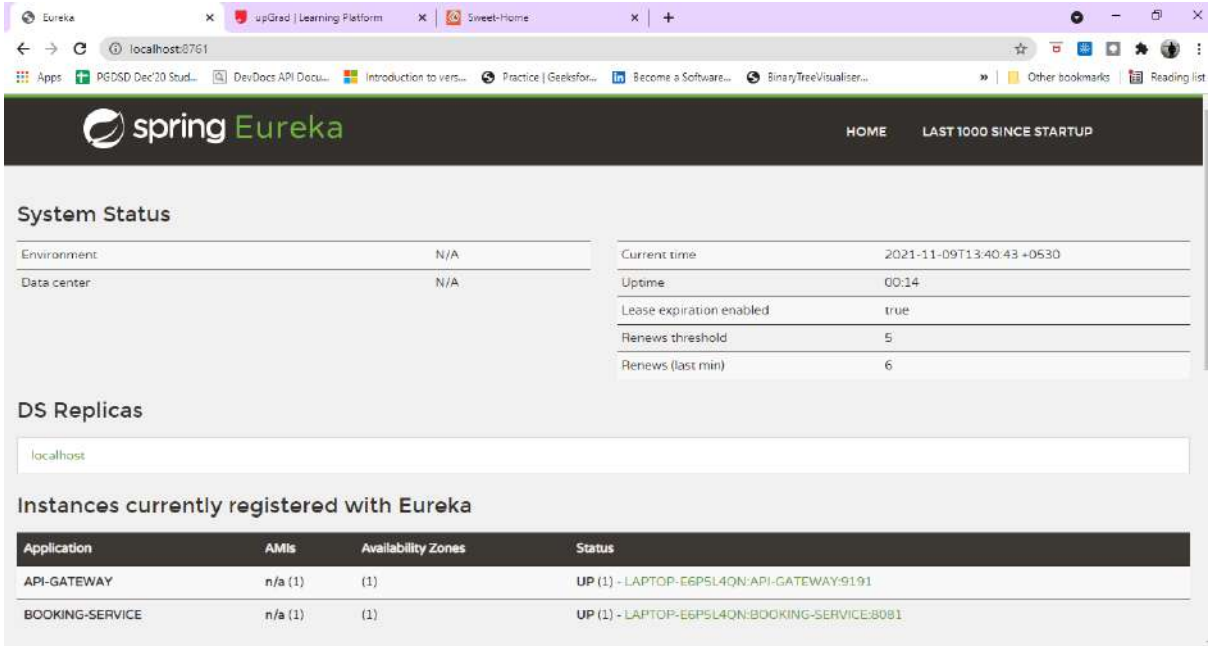


RUN the application



On refreshing the Eureka server URL http://localhost:8761/

We can see the BOOKING-SERVICE instance



5.PaymentService→ Open the proect folder and navigate to the

Path: **src/main/resources/application.properties**

And change the **application.properties** file with appropriate values of

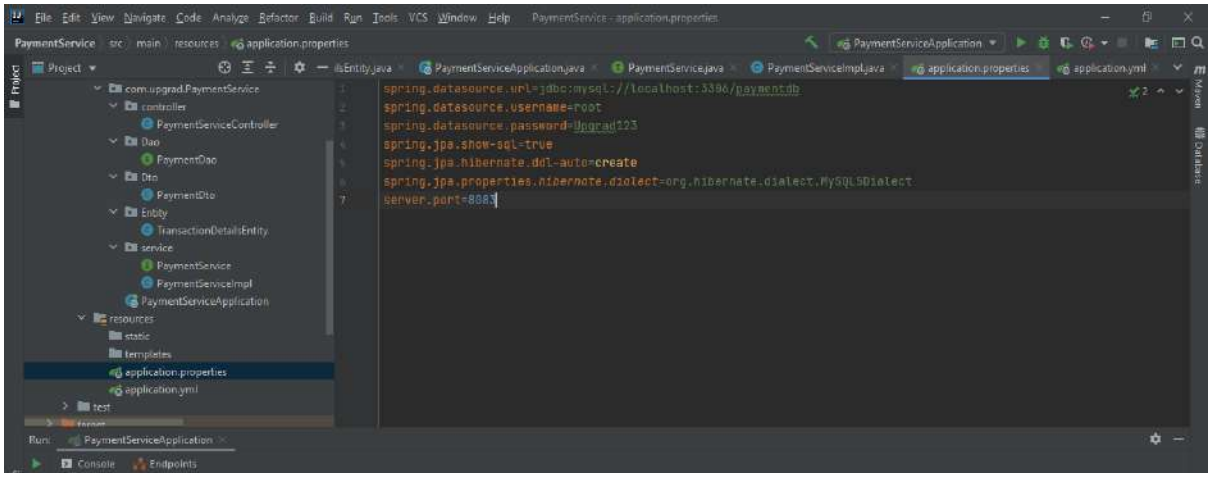
datasource url ,username and password.

In my case the db created was **paymentdb**

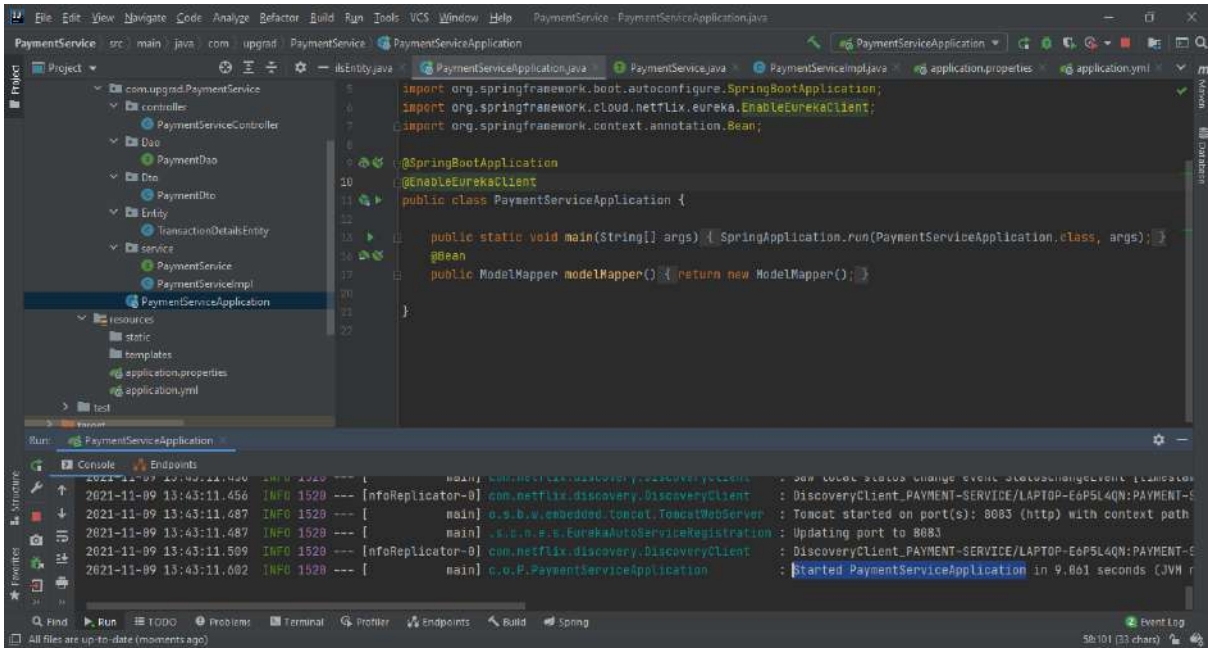
Username is **root**

Password is **Upgrad123**

```
spring.datasource.url=jdbc:mysql://localhost:3306/paymentdb
spring.datasource.username=root
spring.datasource.password=Upgrad123
```

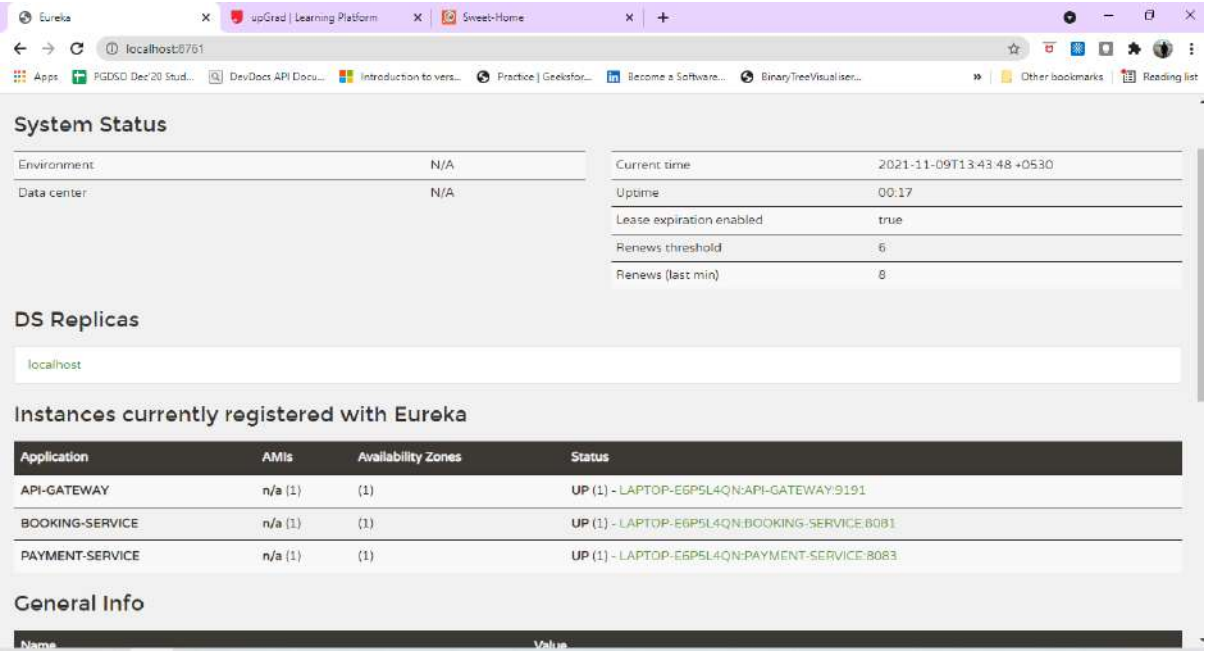


RUN the application



On refreshing the Eureka server URL http://localhost:8761/

We can see the PAYMENT-SERVICE instance

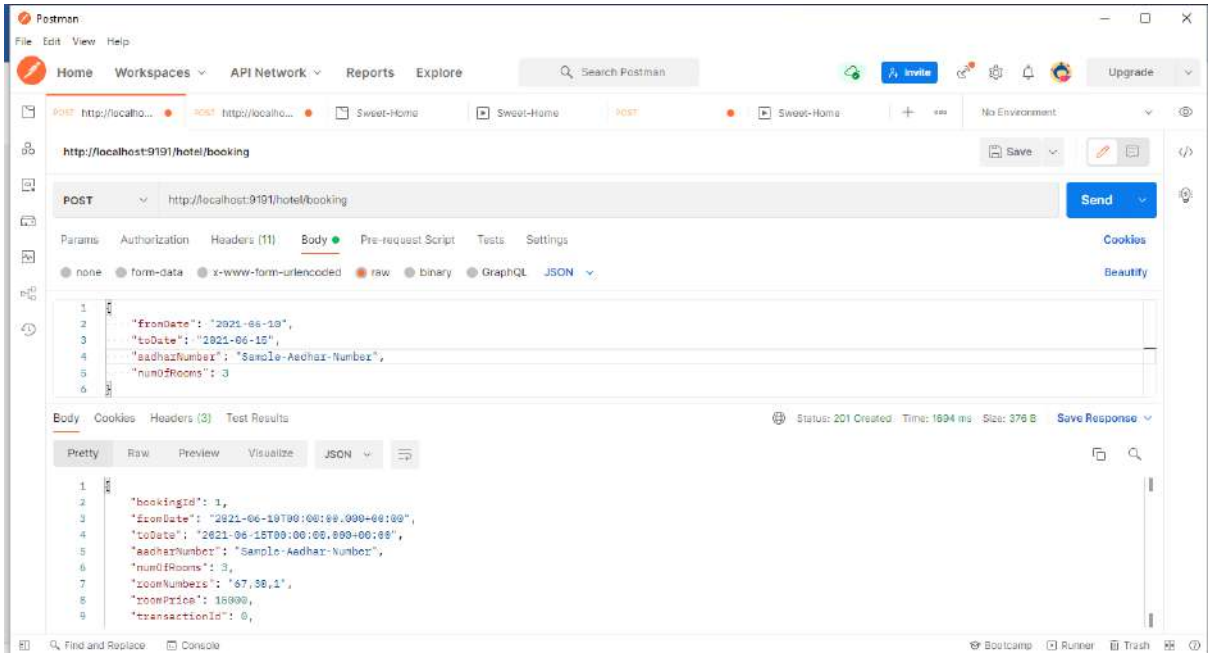


Now test the End points using POSTMAN

ENDPOINT 1: ***http://localhost:9191/hotel/booking***

```
/**
 * HTTP METHOD: POST to save Booking DATA
 * URL : http://localhost:9191/hotel/booking
 *
 * Request Body
 *
 *          {
 *          "fromDate": "2021-06-10",
 *          "toDate": "2021-06-15",
 *           "aadharNumber": "Sample-Aadhar-Number",
 *           "numOfRooms": 3
 *          }
 *
 *
 * Note that http://localhost:9191 is API GATE WAY URL
 * actual URI of the end point is
 *
 * http://localhost:8081/hotel/booking
 *
 */
```

You can see as below



You can see the Room numbers and the room price calculated

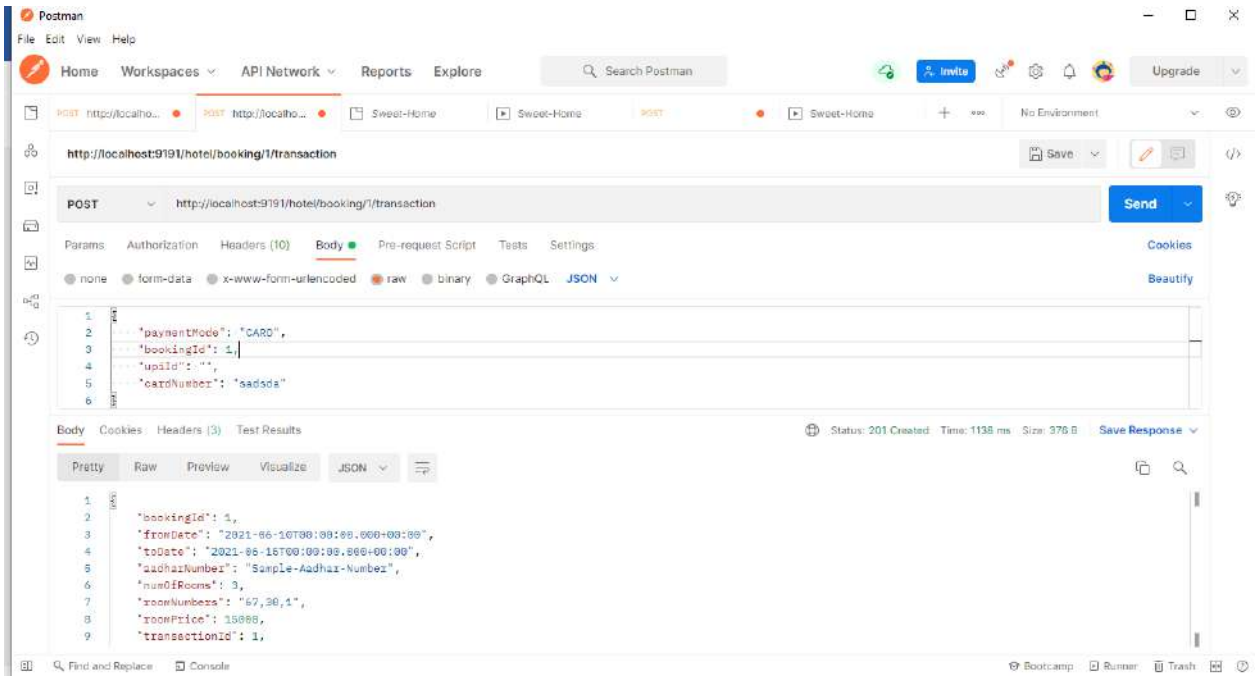
Also the transaction ID 0

ENDPOINT2:

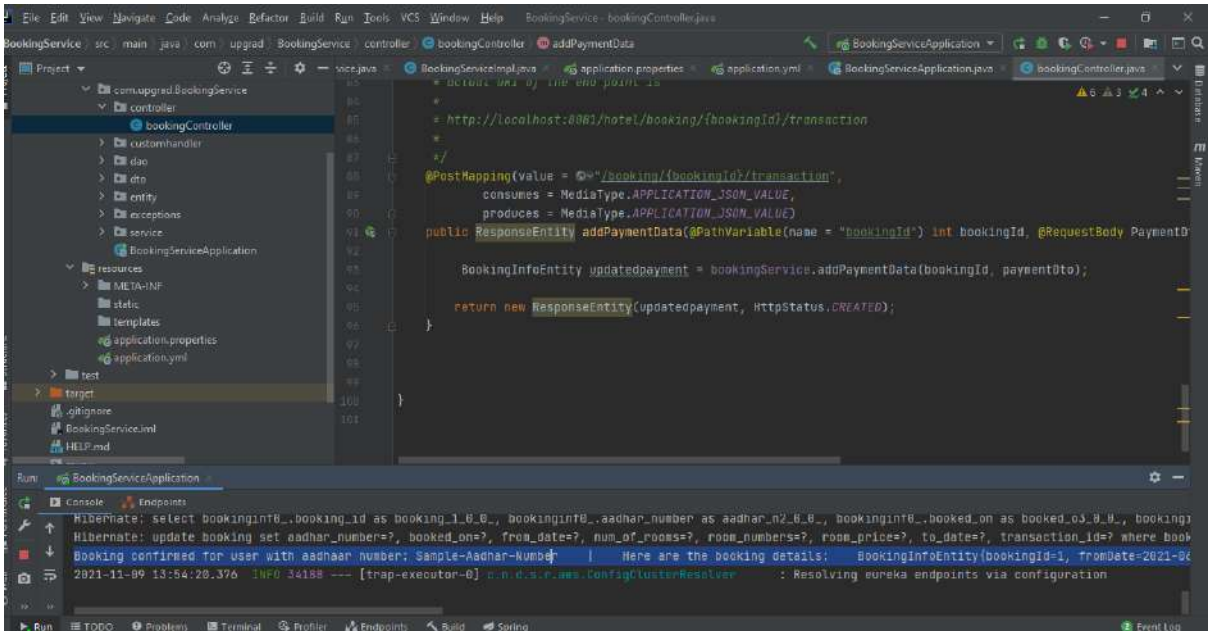
http://localhost:9191/hotel/booking/1/transaction

```
/**
 * HTTP METHOD: POST to call the Payment service
 * FROM Booking service using rest template
 * to get the transactionID
 *
 * URL: http://localhost:9191/hotel/booking/{bookingId}/transaction
 *
 * Request Body
 * {
 *     "paymentMode": "CARD",
 *     "bookingId": 1,
 *     "upiId": "",
 *     "cardNumber": "sadsda"
 * }
 *
 * Note that http://localhost:9191 is API GATE WAY URL
 * actual URI of the end point is
 *
 * http://localhost:8081/hotel/booking/{bookingId}/transaction
 *
 */
```

You can see all below with the transaction ID

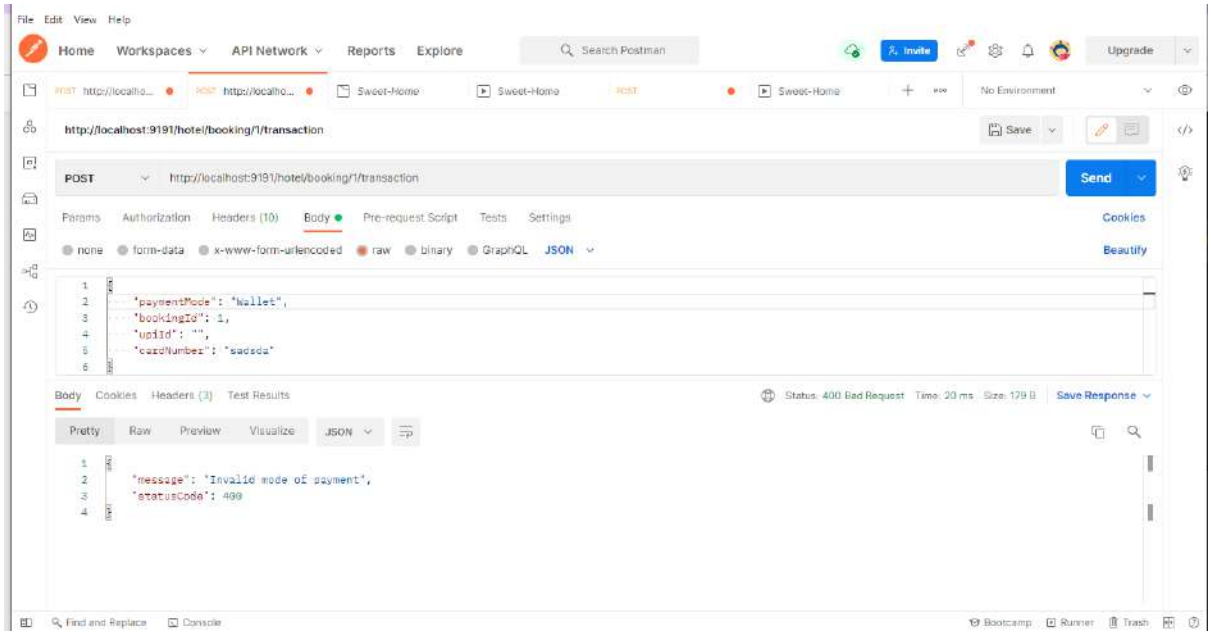


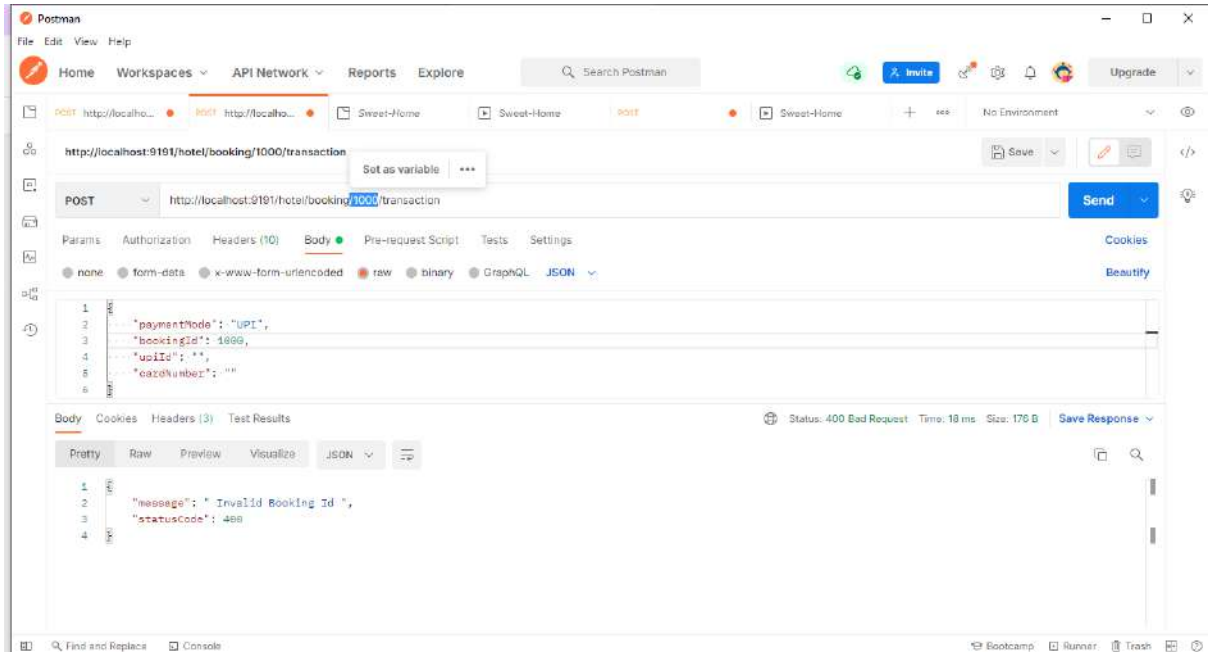
Also you can see the confirmation message on the console of BOOKING_SERVICE



Exceptions

Paymentmode and Wrong booking id were handled you can see as below if given wrong paymentMode and bookingid.





You can also run the given api

The code work flow is as follows

Created Eureka server and configured it to run on the port 8761

Created API Gateway and configured it to run on port 9191,made it to be discovered by the Eureka server and also gave the paths of BOOKING-SERVICE and PAYMENT -SERVICE

Created Booking and Payment service applications and also created Database bookingdb ,paymentdb in MySQL workbench respectively for the services.

Configured the databases respectively.Also the applications were configured with Eureka server and API GATEWAY.

Created Entities as per the sweethome schema document given.

Created DAO layer by extending with JPA repository

Created DTO classed

Implemented the service layer and applied logic as per requirement.

Code for calculating number of days from the given from and to date , price is done as below in **addbooking** method in BookingServiceImpl.java

```java
@Override
public BookingInfoEntity addBooking(BookingInfoEntity bookingInfoEntity) {

    int totalDays;
    int noOfRooms = bookingInfoEntity.getNumOfRooms();
    int roomPrice;

    /**
     *  calculating number of days and price based on given room price as Rs.1000 per room per day.
     */
    long difference = bookingInfoEntity.getToDate().getTime() - bookingInfoEntity.getFromDate().getTime();
    totalDays = (int) (difference / (1000 * 60 * 60 * 24));
    roomPrice = 1000 * noOfRooms * (totalDays);

    ArrayList<String> roomNumbers = getRandomNumbers(bookingInfoEntity.getNumOfRooms());
    String roomNumber = String.join( delimiter: ",", roomNumbers);

    bookingInfoEntity.setRoomNumbers(roomNumber);
    bookingInfoEntity.setRoomPrice(roomPrice);
    bookingInfoEntity.setBookedOn(new Date());


    bookingDao.save(bookingInfoEntity);

    return bookingInfoEntity;
```

The code for getting Transaction ID from PAYMENT-SERVICE using RestTemplate is highlighted

```java
public BookingInfoEntity addPaymentData(int bookingId, PaymentDto paymentDto) throws invalid
    String PaymentServiceURL = apiGateway + "/payment/transaction";
    if ("UPI".equals(paymentDto.getPaymentMode()) || "CARD".equals(paymentDto.getPaymentMode())) {
        if (!bookingDao.existsById(bookingId)) {
            throw new invalidBookingIdException();
        }
        // Getting Booking Data based on BookingID
        BookingInfoEntity bookingData = getBookingData(bookingId);
        // Getting transaction Id from Payment Service
        Integer transactionId = restTemplate.postForObject(PaymentServiceURL,
                paymentDto, Integer.class);
        //Updating the transaction ID received from Payment service in booking data
        bookingData.setTransactionId(transactionId);
        bookingDao.save(bookingData);
        String message = "Booking confirmed for user with aadhaar number: "
                + bookingData.getAadharNumber()
                + " | "
                + "Here are the booking details:    " + bookingData.toString();

        //Printing message to console
        System.out.println(message);

        return bookingData;

    } else {
        throw new invalidPaymentModeException();
    }
```

Created the controller classes and added end points required for the project

Also created Custom exception handler and handled different exceptions.

```java
import java.util.Map;

@ControllerAdvice
public class customBookingExceptionHandler {

    @ExceptionHandler
    public ResponseEntity PaymentModeExceptionHandler(invalidPaymentModeException e){
        Map<String, Object> errorResponse = new HashMap<>();
        errorResponse.put("message", "Invalid mode of payment");
        errorResponse.put("statusCode", 400);
        return new ResponseEntity(errorResponse, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler
    public ResponseEntity InvalidBookingIdExceptionHandler(invalidBookingIdException e){
        Map<String, Object> errorResponse = new HashMap<>();
        errorResponse.put("message", " Invalid Booking Id ");
        errorResponse.put("statusCode", 400);
        return new ResponseEntity(errorResponse, HttpStatus.BAD_REQUEST);
    }
}
```

-