Mini Project -2 (MicroService Based Application)

Hotel Room Booking Application

Avtar Singh (20CSU241)

1. Context

2. Introduction

Welcome to our user-friendly hotel booking application! Crafted with care using Java and Spring Boot, this microservice-based project ensures a seamless experience. Behind the scenes, we've integrated MySQL for secure data storage. This project serves as a basic template for a hotel booking application. It includes two main entities: `Hotel` and `Booking`. The application provides RESTful APIs for managing hotels and bookings.

We are using MicroServices Architecture in which we are dividing our application into Three Different Spring Projects :

API-Gateway: Single Entry Point

- External Face: Serves as the public interface of our system, handling incoming requests.
- Router: Directs requests to the appropriate microservices within the application.
- Roles : Like a security, ensures smooth communication between external requests and internal services.

Booking Service: Handles Hotel and Booking

- Information Hub: Gathers and processes user details and booking preferences.
- Confirmation Provider: Manages the confirmation process once a booking is selected.
- Communication Hub: Sends out confirmation messages to users, akin to a hotel receptionist confirming reservations.

Booking Service: Handles Payments and Transactions

- Virtual Cashier: Although a "dummy" service, it plays a crucial role in the system.
- Triggered by Booking Service: Initiates the payment process after a booking is confirmed.
- Ensures Transaction: Similar to a cashier, it ensures the seamless execution of financial transactions within the booking system.

2. Getting Started

Lets Understand the working using the workflow diagram :

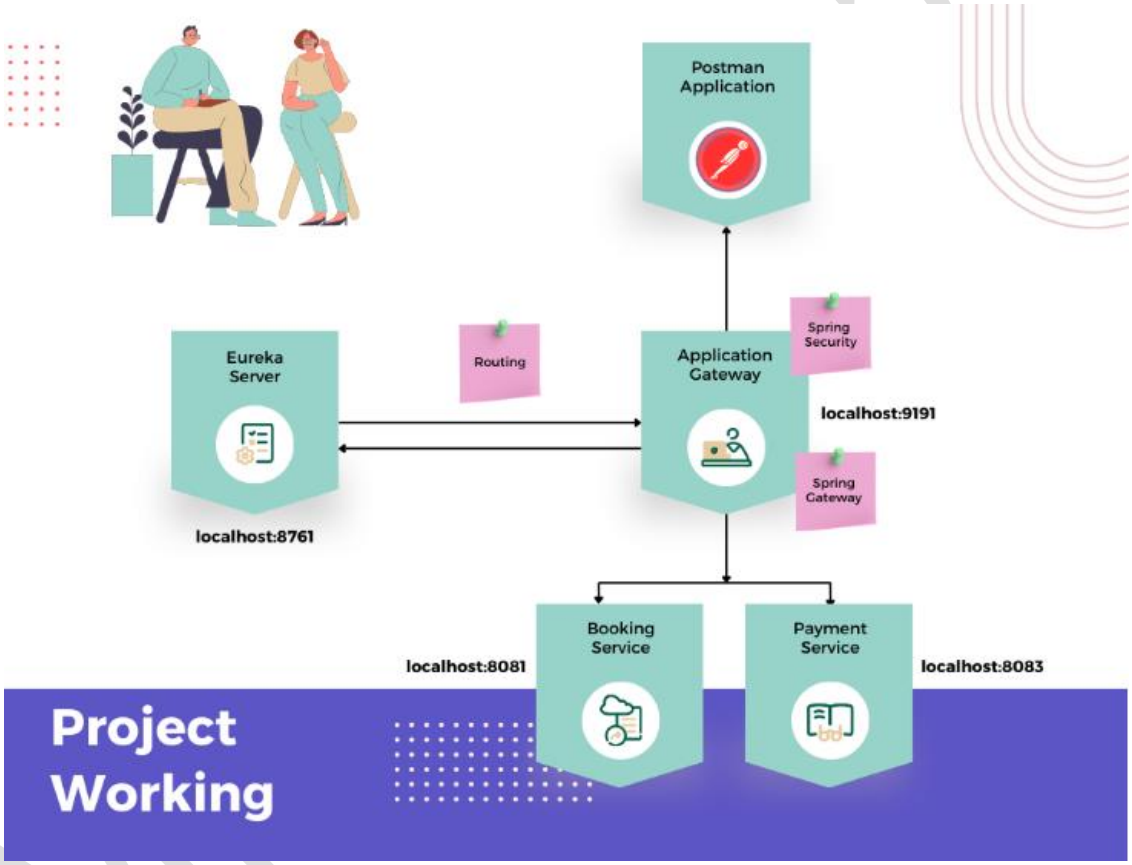


Figure 1

As Explained by this diagram :

- **Work via PostMan:**

API Gateway, Booking service, and Payment service will work and can we used by API Platform - PostMan , Curl etc.

- ## Service Registration:

API Gateway, Booking service, and Payment service register themselves on the Eureka server during the system initialization.

- ## User Interaction Flow:

All user interactions start with the API Gateway; users don't directly interact with the Booking or Payment services.
The 'Booking' service is triggered when a user initiates room booking, providing details like toDate, fromDate, aadharNumber, and numOfRooms.

- ## Booking Service Response:

'Booking' service responds with a list of available room numbers and their associated prices. User is prompted to enter payment details if they choose to proceed, and their provided information is stored in the 'Booking' service database.

- ## Payment Transaction Process:

If the user decides to proceed with the booking, they provide payment details (bookingMode, upiId/cardNumber) to the 'Booking' service.
'Booking' service forwards payment details to the 'Payment' service for a dummy transaction.
'Payment' service performs the transaction, returning the associated transaction Id to the 'Booking' service.
All transaction details are stored in the 'Payment' service database.

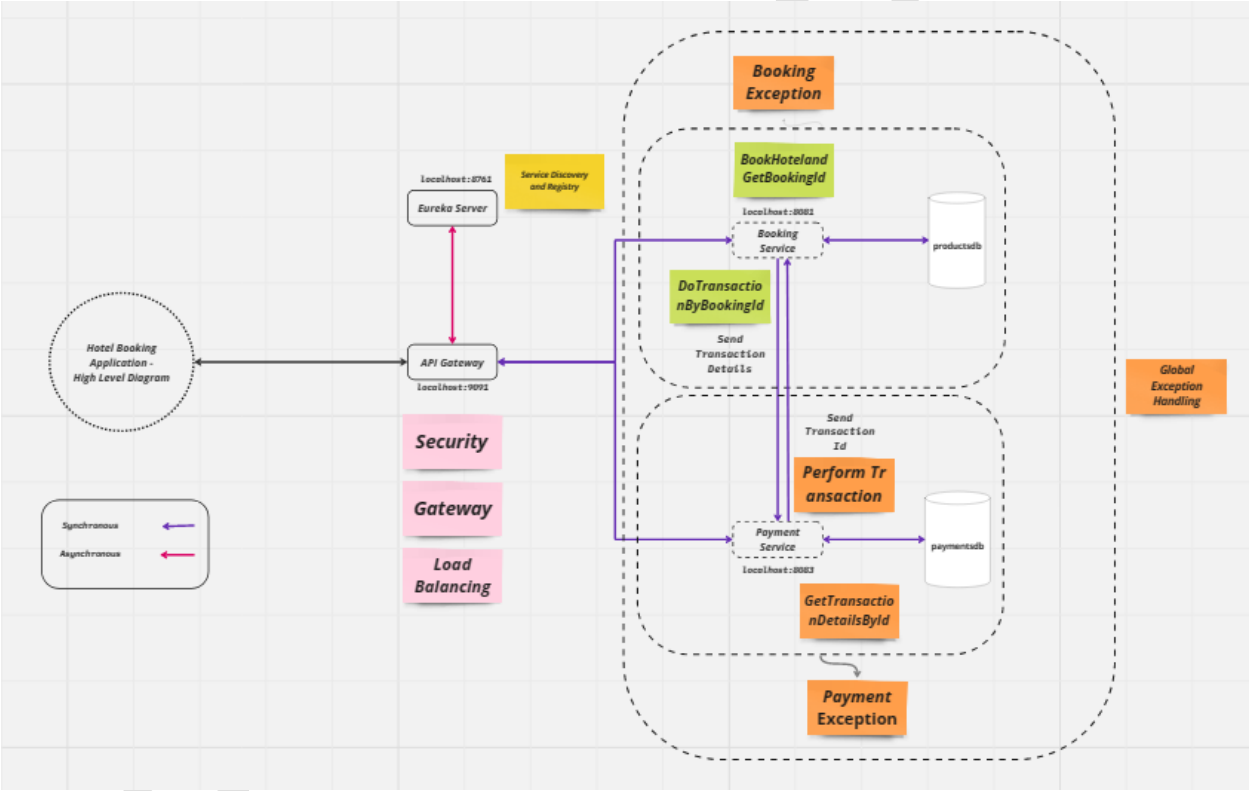
- ## Confirmation and Completion:

Once the transaction is completed, the 'Booking' service acknowledges the confirmed booking by sending a confirmation message to the console.

The entire process ensures a secure and streamlined room booking experience for the user, with clear communication between services and effective storage of relevant data in their respective databases.

3. High Level Diagram (Hotel Booking Application)

Board Link -> https://miro.com/app/board/uXjVNCqSVsQ=/?share_link_id=764323176558



4. System Configuration

Make sure you have the following installed on your system:

- Java 8 or higher
- Maven

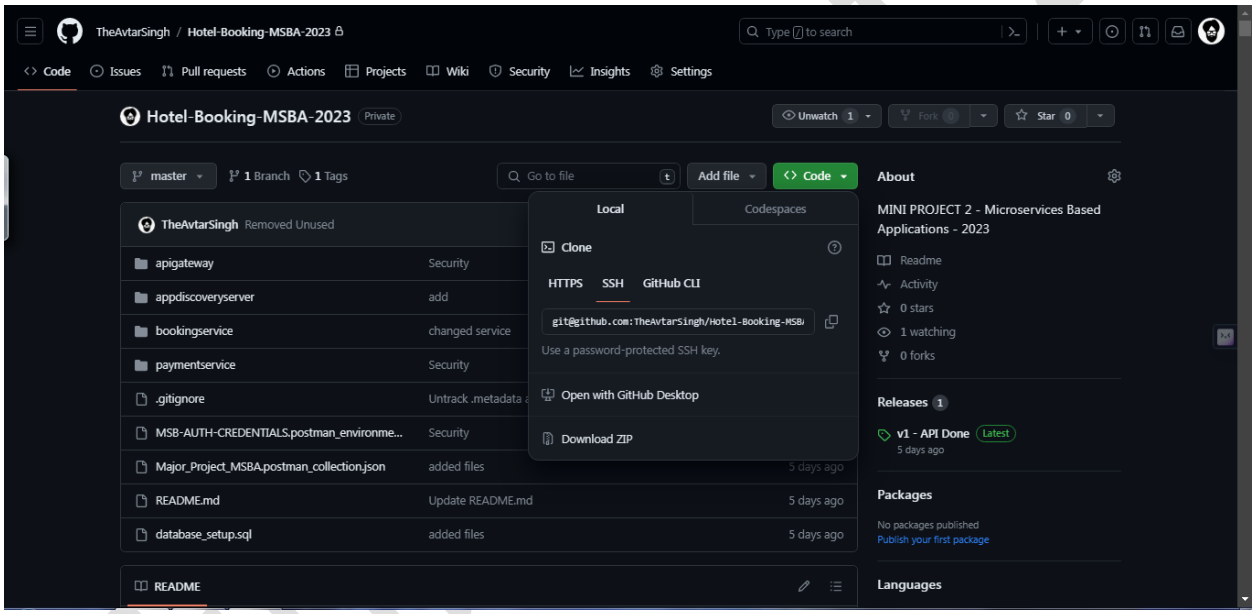
- Spring Boot
- MySQL or another compatible database (for data storage)

5. Installation

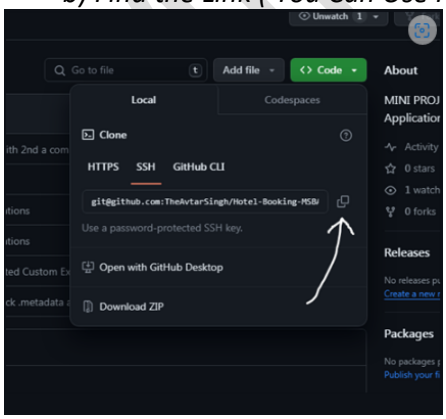
Follow Proper Procedure to run Smoothly

1. **Clone the repository**

a) Find the Link and Clone using Git Bash /Github Desktop (You Can Use Https/SSH)

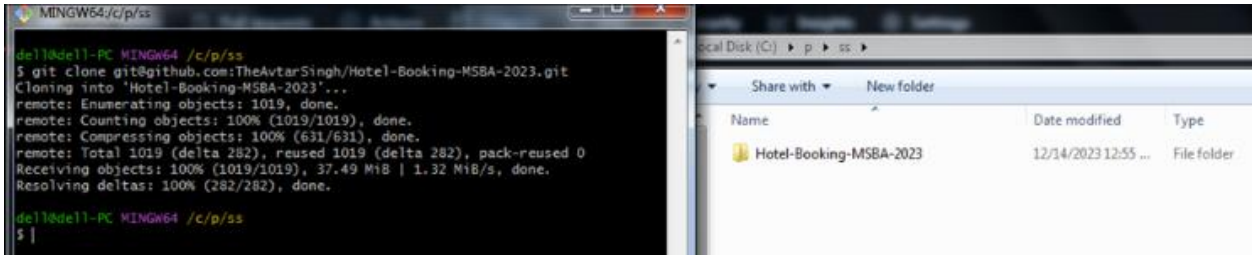


b) Find the Link (You Can Use Https/SSH)



2. **Clone the repository using**

git clone https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023.git



3. **Setup Database**

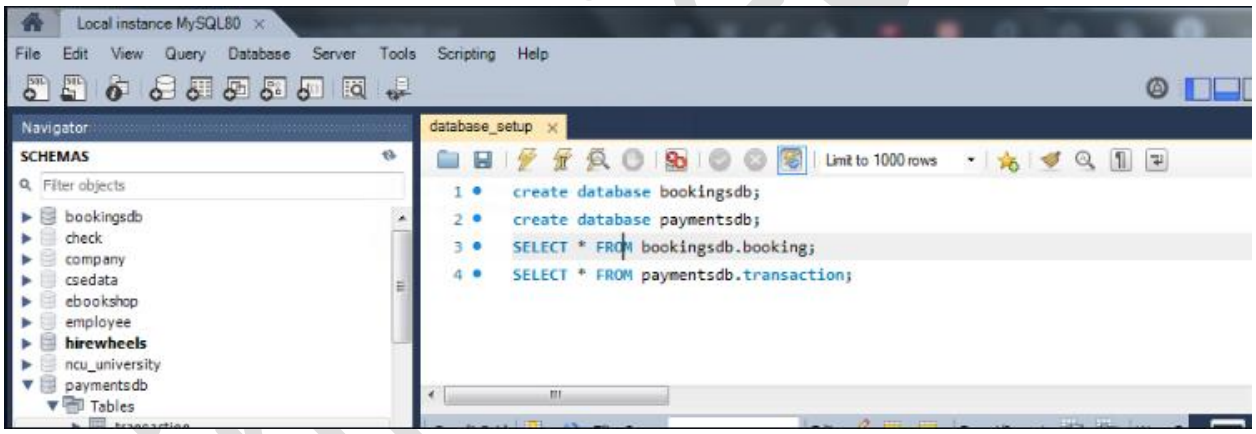
   Use Commands :

   create database bookingsdb;

   create database paymentsdb;

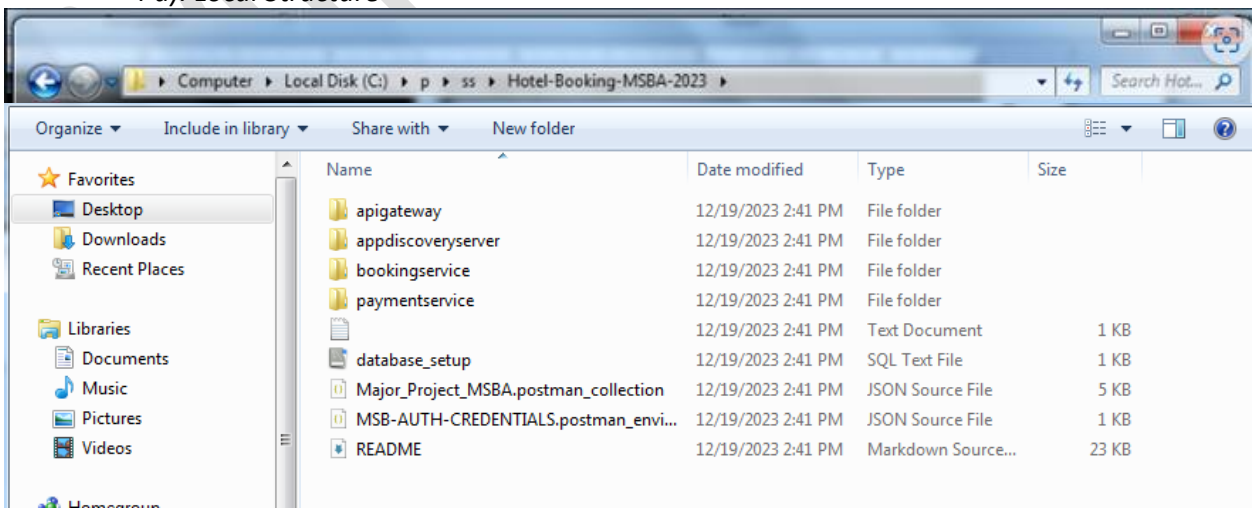
   *SELECT * FROM bookingsdb.booking;*

   *SELECT * FROM paymentsdb.transaction;*

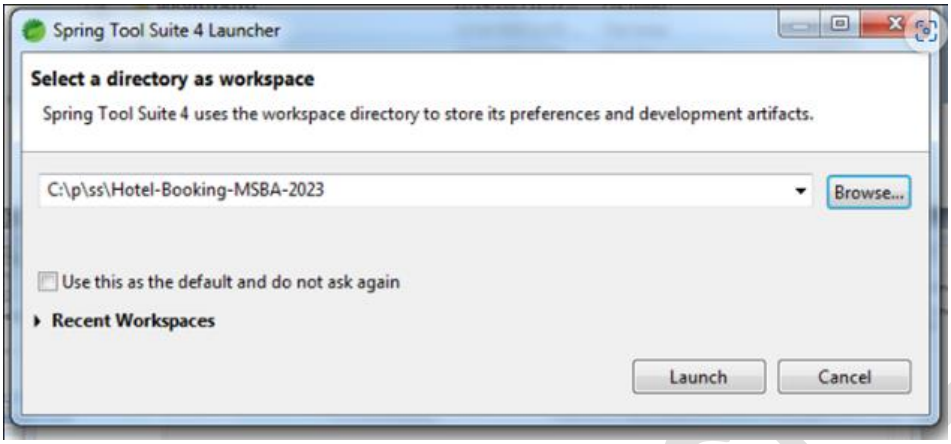


4. **Setup Your IDE (I am Using Spring Tool Suite) :**

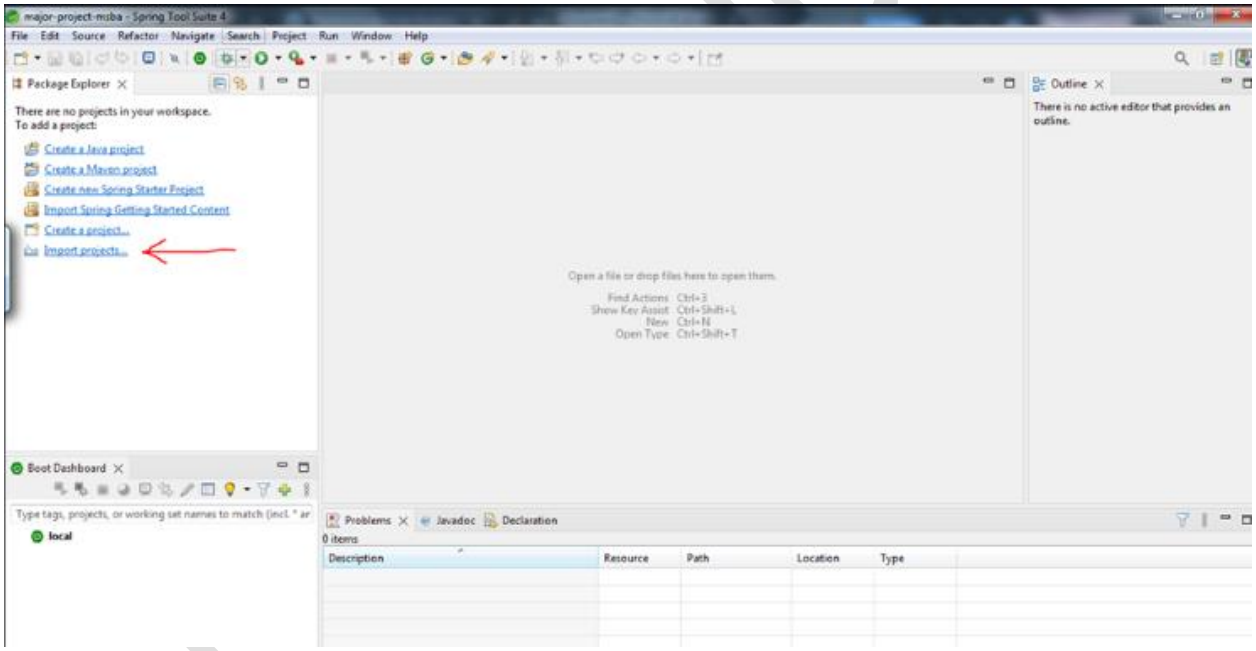
   4 a). Local Structure



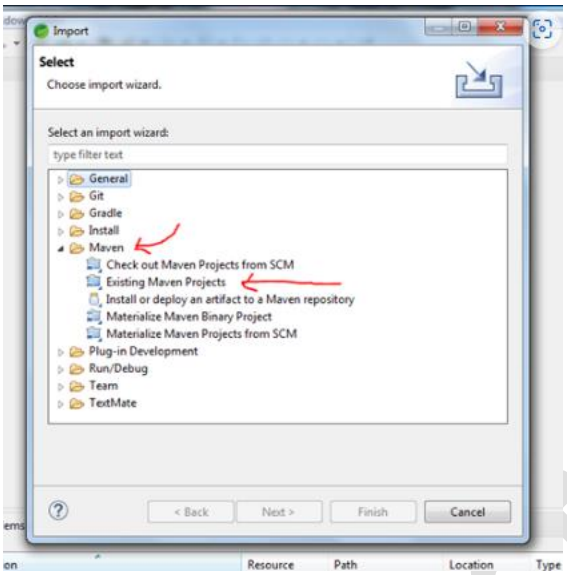
4 b). Open IDE



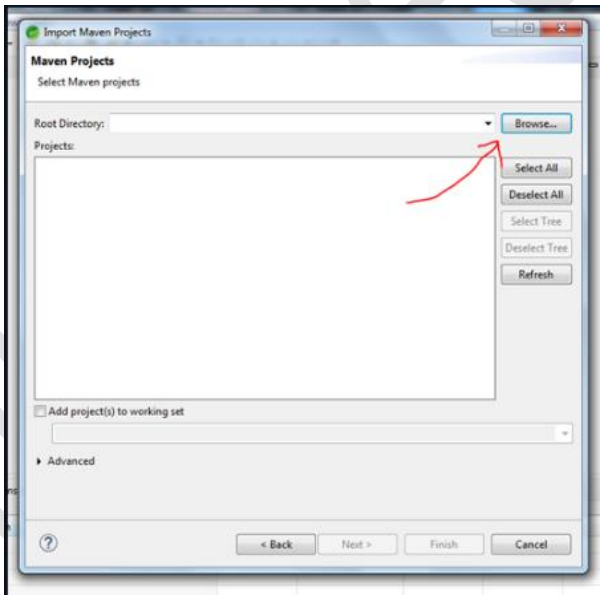
4 c). Import Projects



4 d). Import Maven Projects

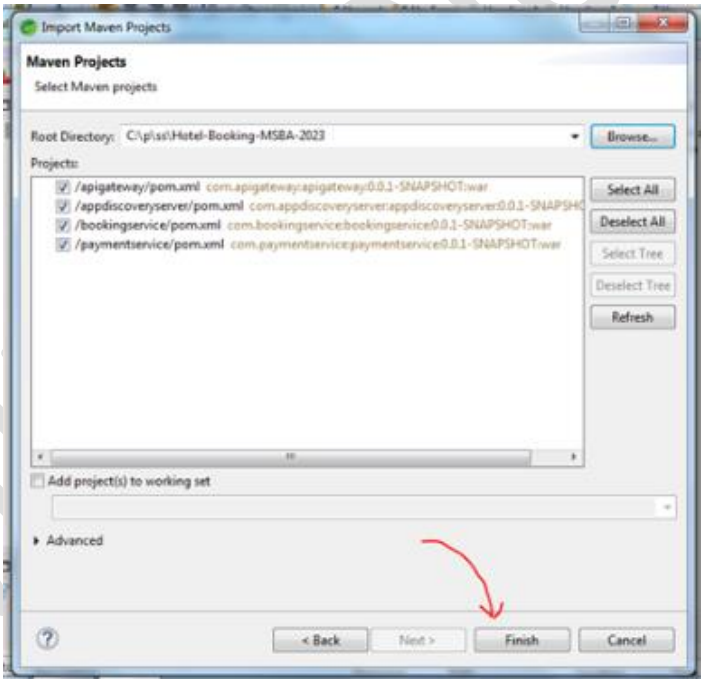


4 e). Choose the directory of Project

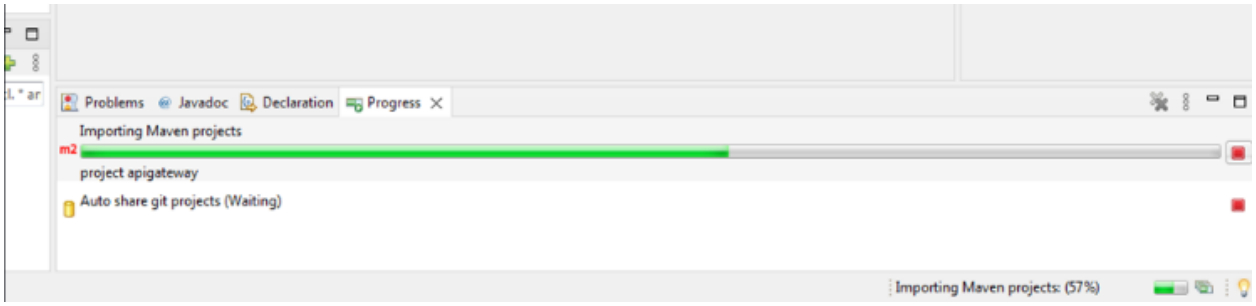


4 f). Choose the root directory of Project

4 g). Import All the Projects

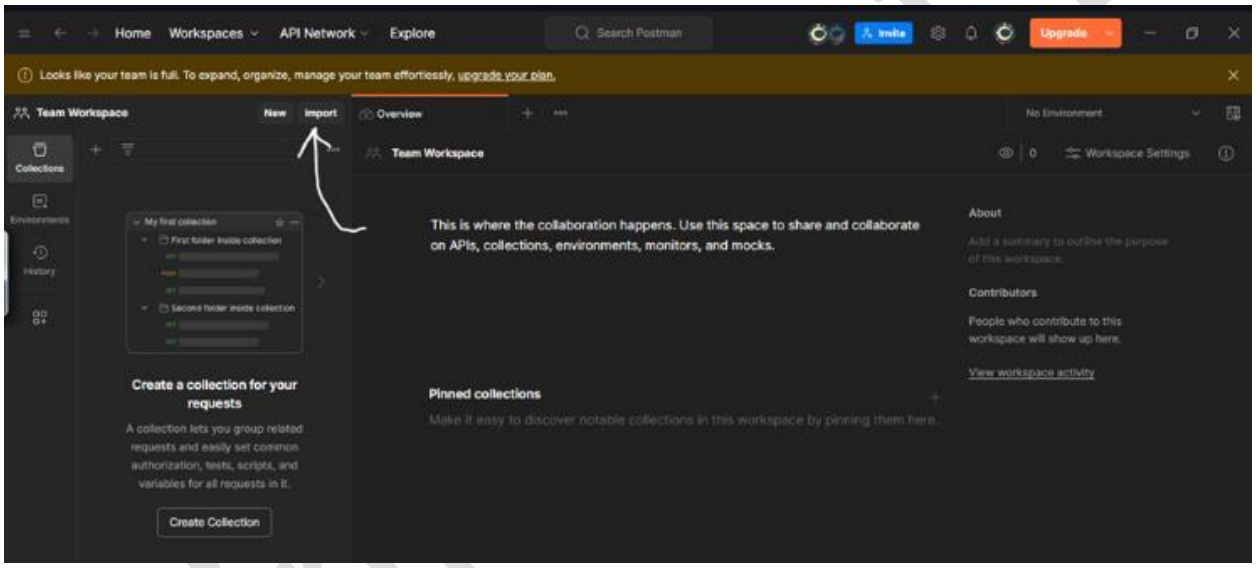


4 h). Importing and Building will be done by IDE

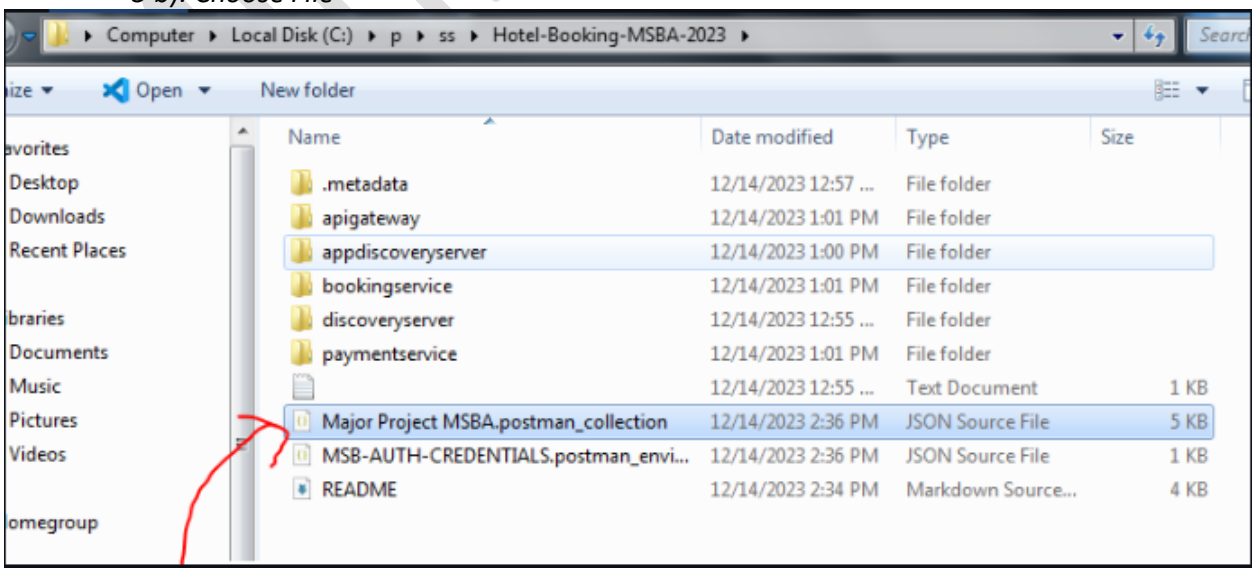


5. **Setup PostMan :**

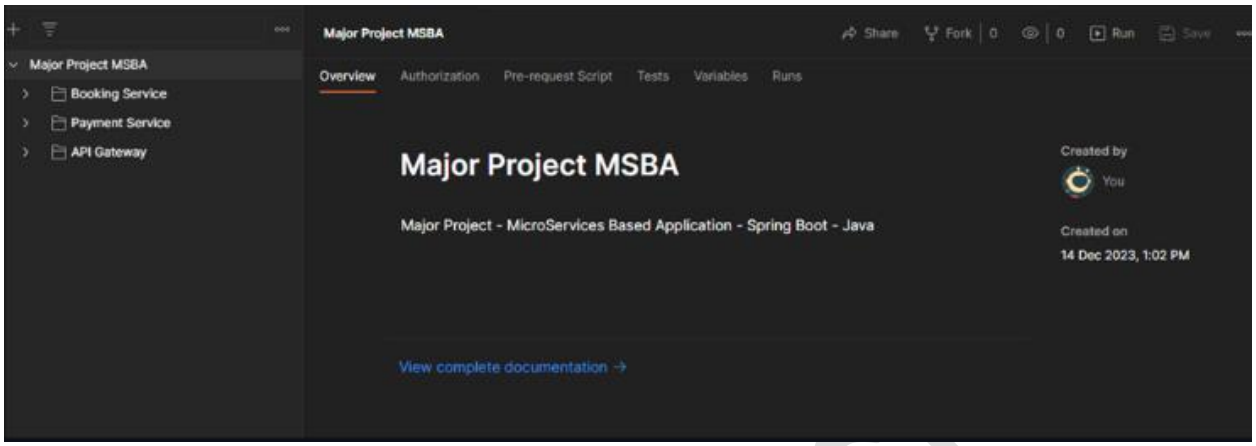
5 a). Import the Collection File - Present in ReadMe Files Section



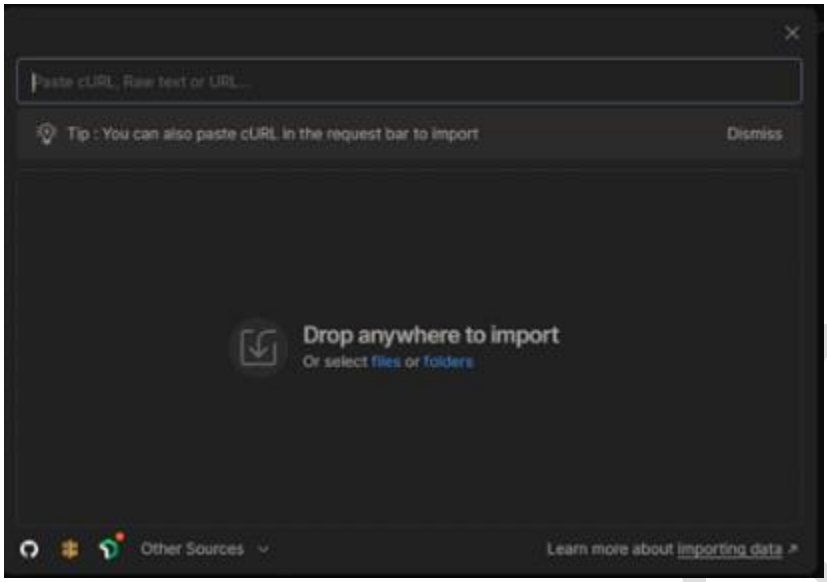
5 b). Choose File

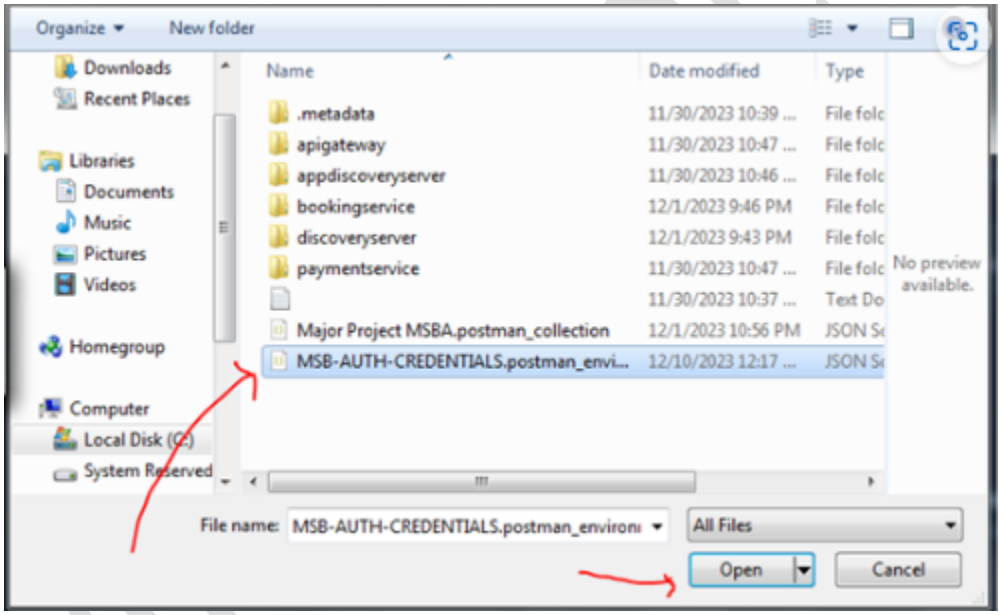


5 c). Result

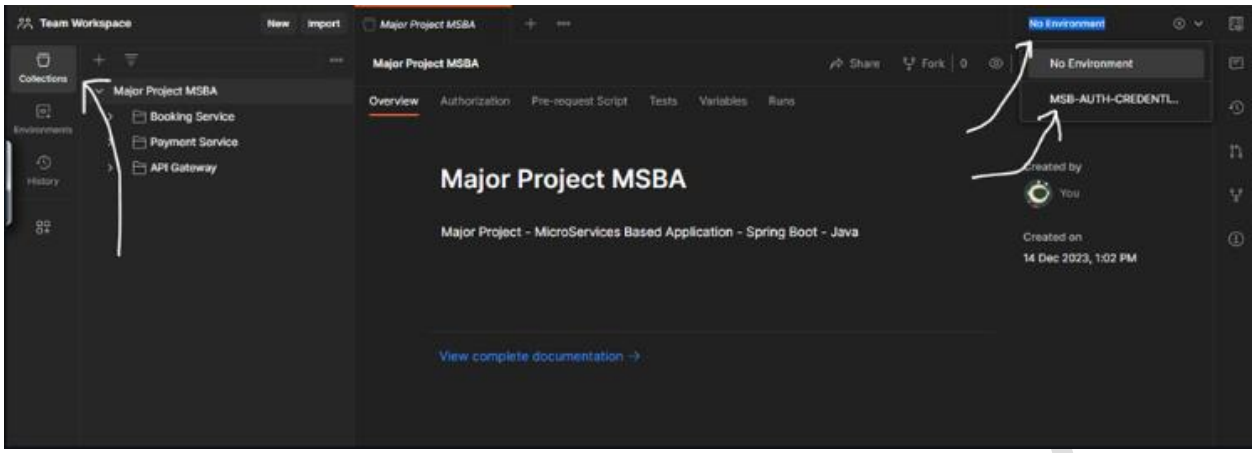


5 d). Import Security Credentails

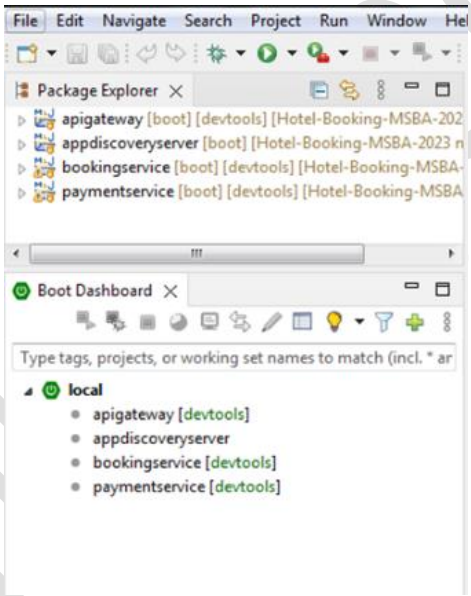




5 e). Choose Proper Environment

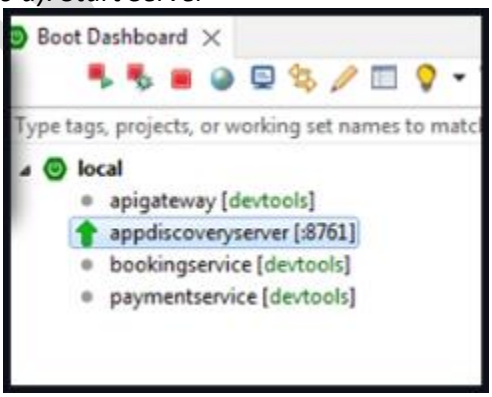


6. **Run the Application :**

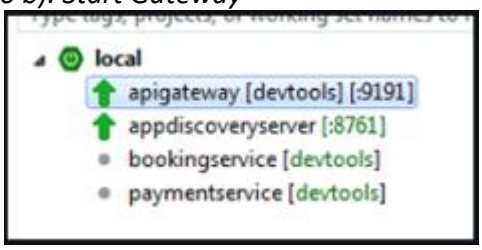


6 a). Order to Run the Application (Preventing Unknown Server Error)

6 a). Start Server

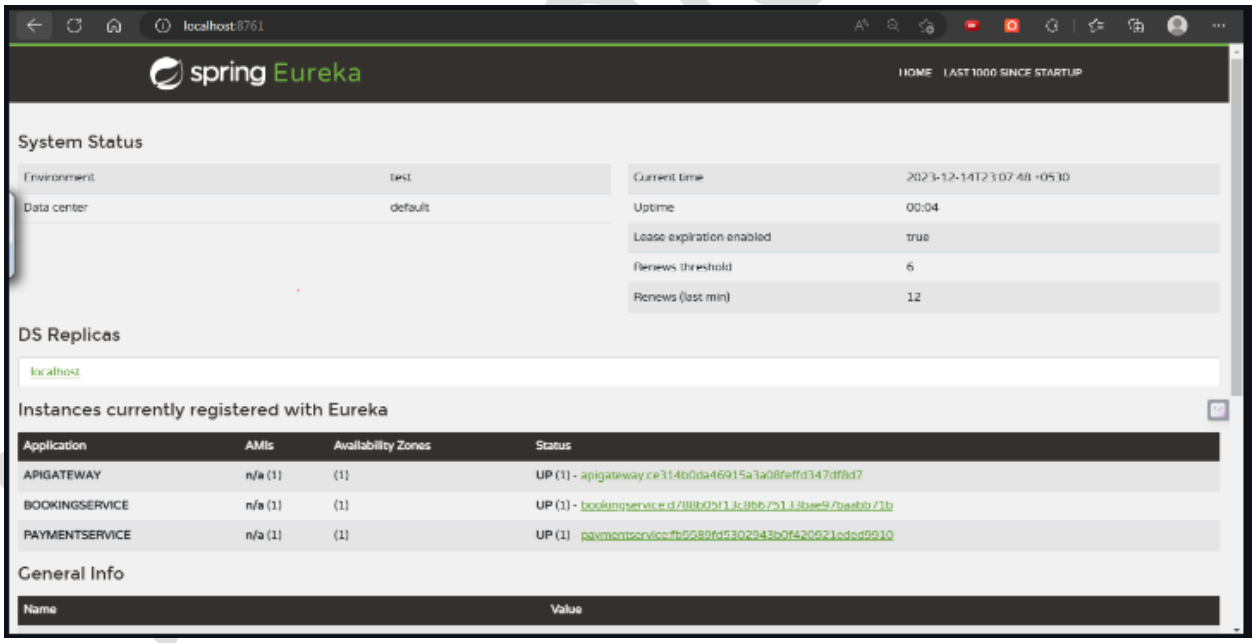


6 b). Start Gateway



6 a). Start Both Services

7. **Verify Succesful Start at *https://localhost:8761* :**



6. Usage

The application provides RESTful APIs for managing hotels and bookings. You can use tools like curl or Postman to interact with the APIs.

7. Endpoint

7. Booking Service : [Booking Service](bookingservice)

7.1. Logic Class (*To Calculate Price and Room Numbers****) -***

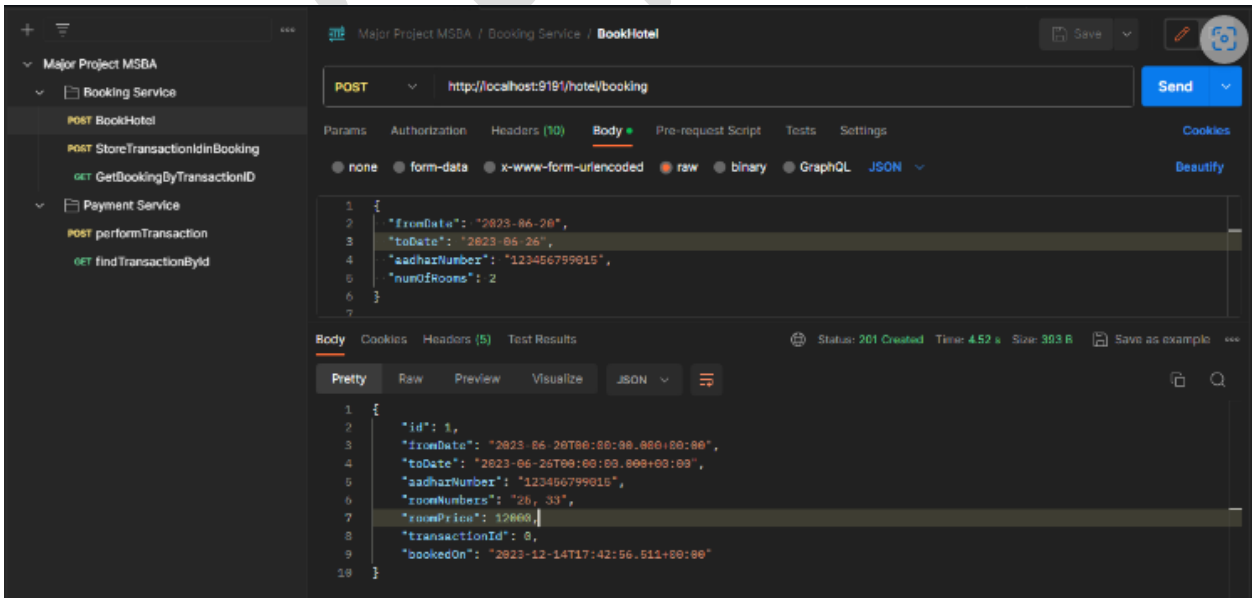
7.2. Model Classes *[BookingInfoEntity]*

7.3. Controller -

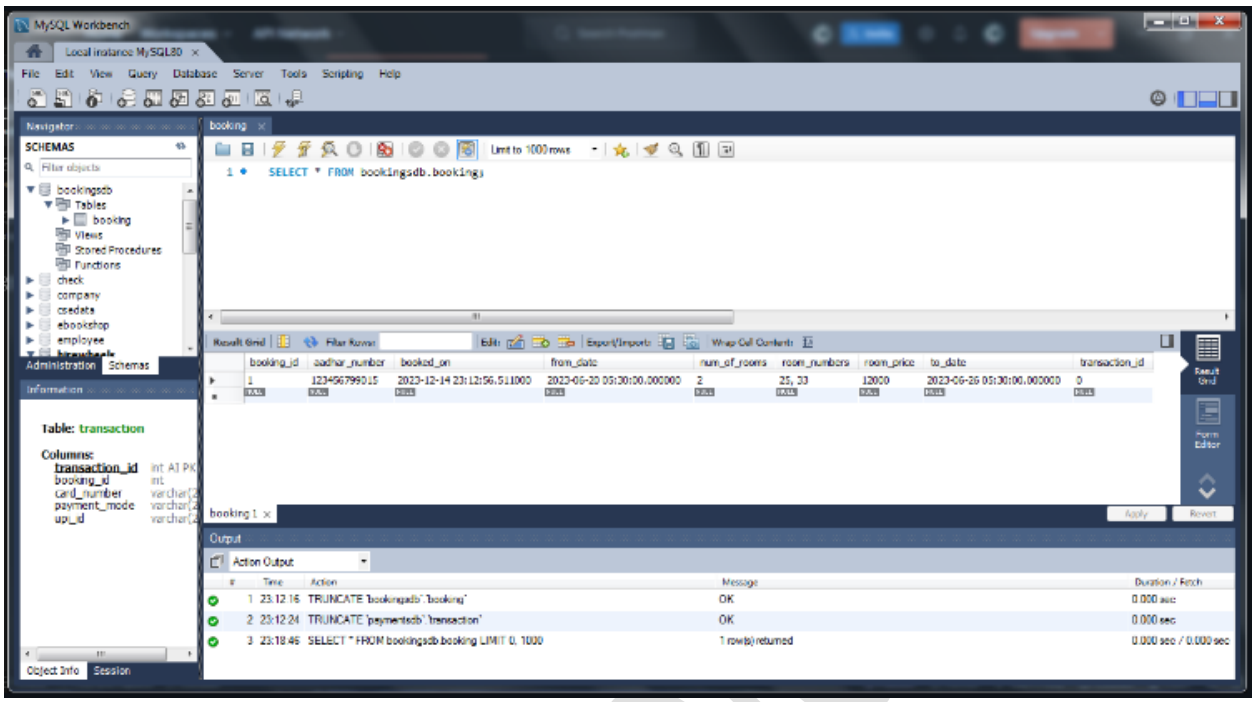
Endpoint 1: POST /booking

- ***URI:*** */booking*
- ***HTTP METHOD:*** *POST*
- ***RequestBody****: fromDate, toDate,aadharNumber,numOfRooms*
- ***Response Status****: Created*
- ***Response****: ResponseEntity<BookingInfoEntity>*

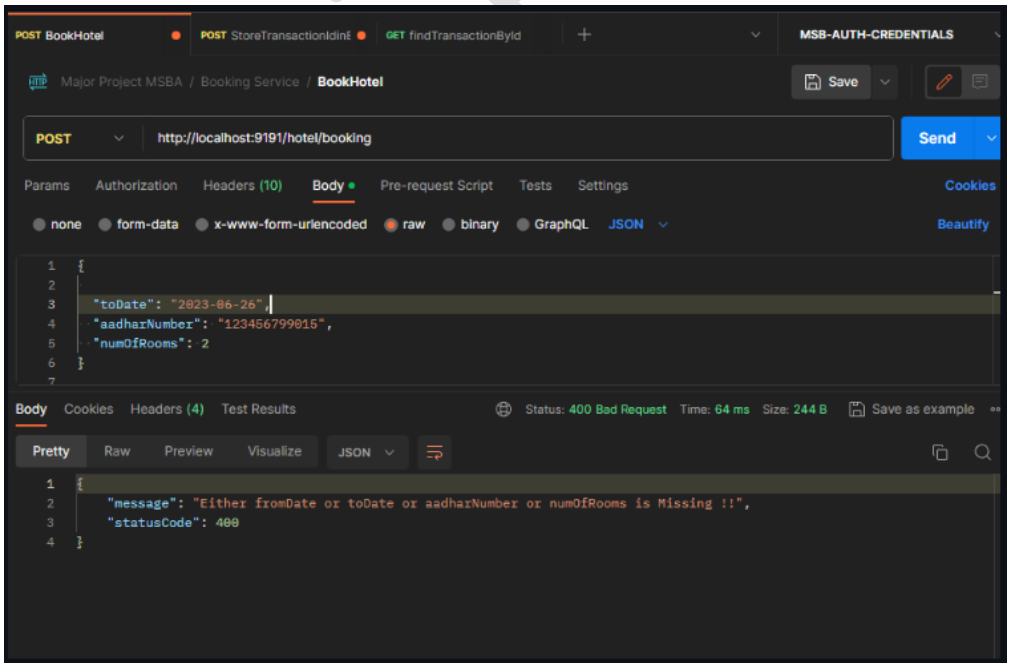
Working



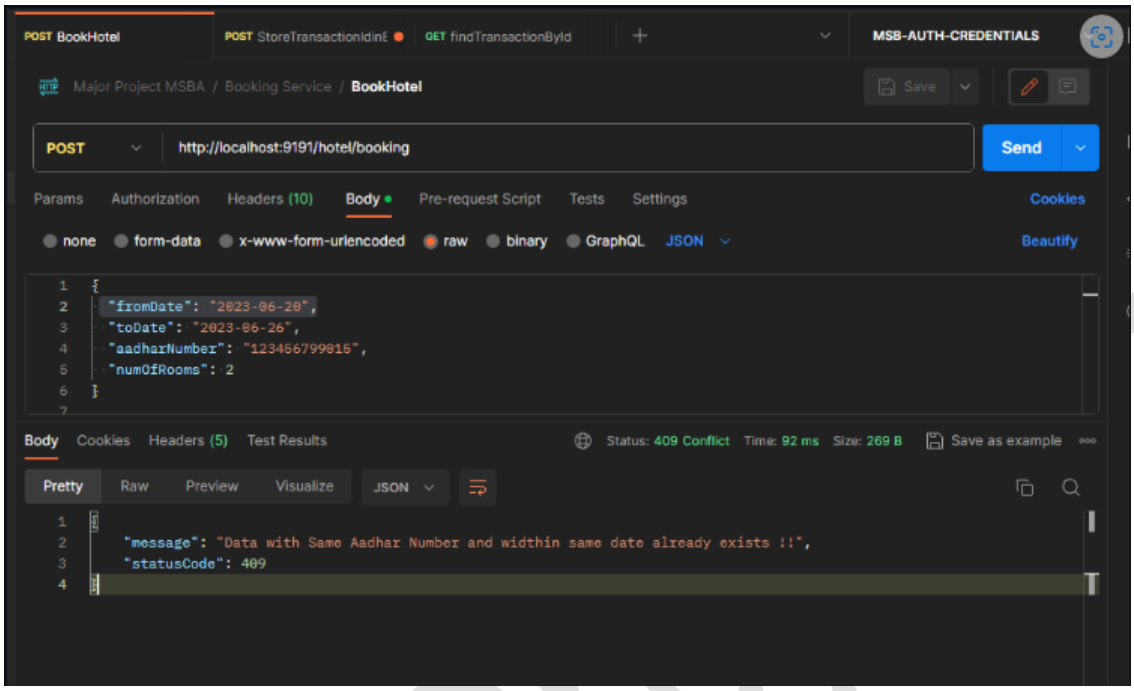
Database



Exception 1 - If Any Of One Field is Missing



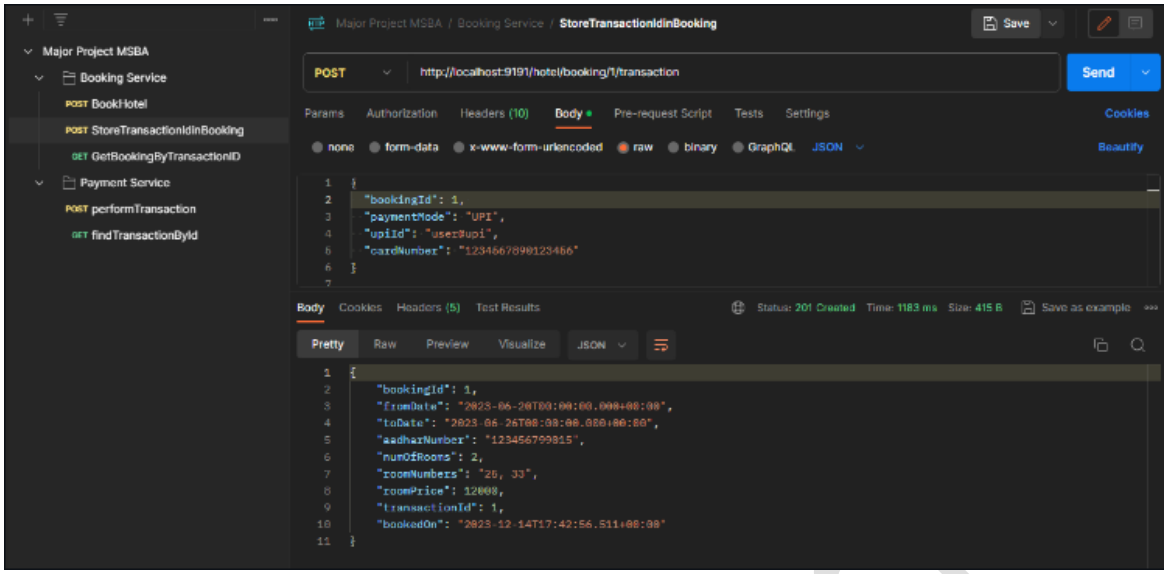
Exception 2 - If Data within Same Date and AAdhar Number is Present



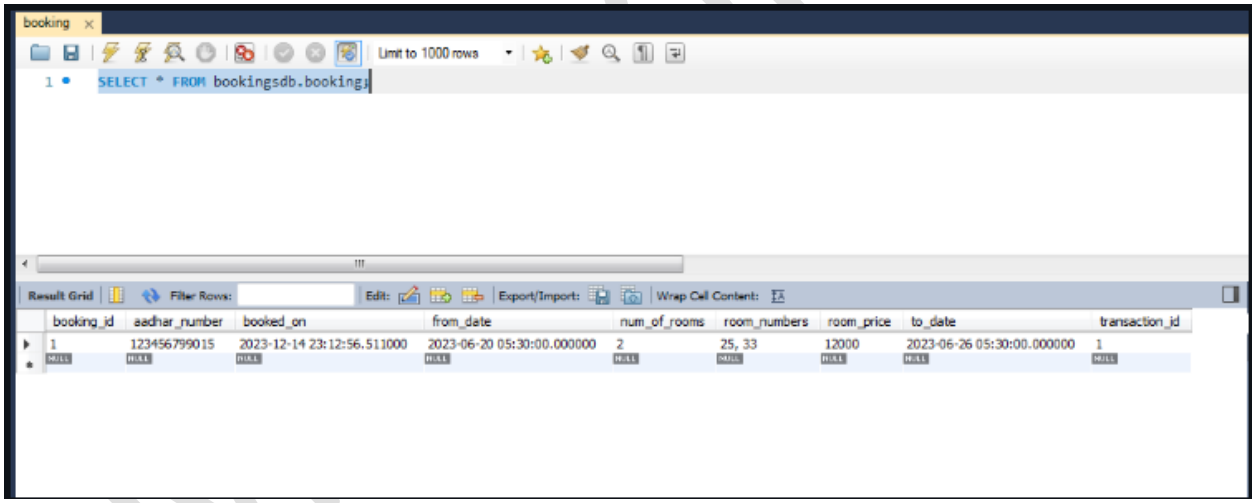
Endpoint 2: POST booking/{bookingId}/transaction

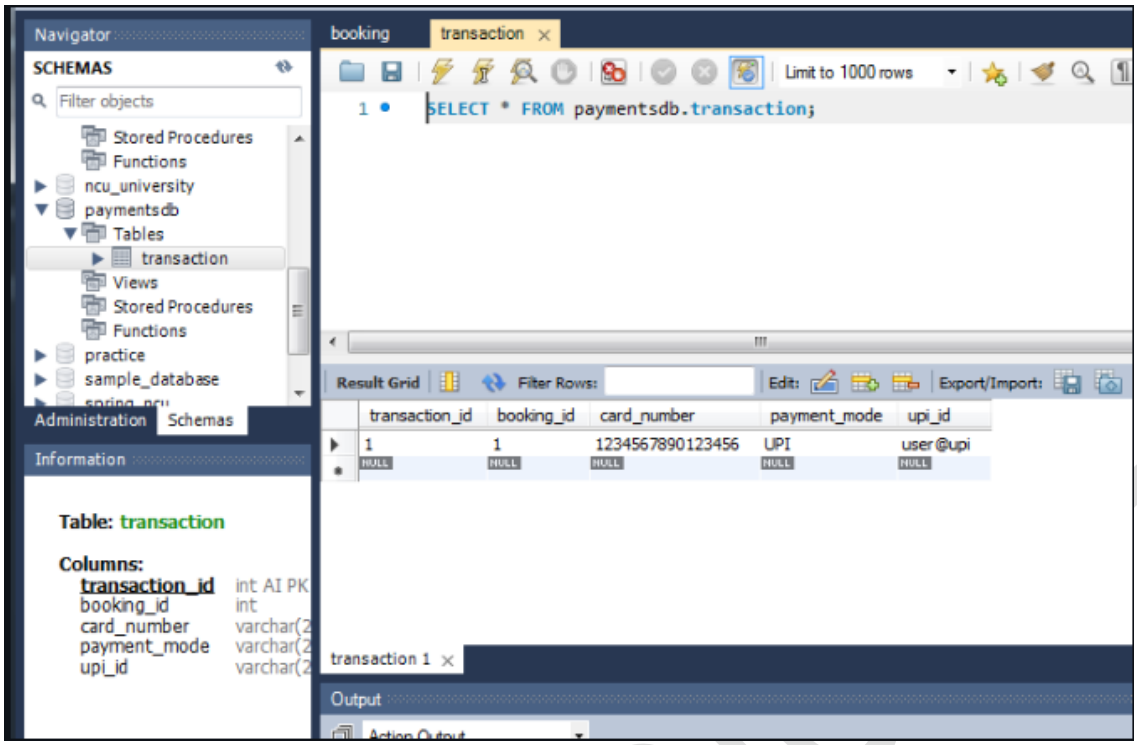
- ***URI:*** */booking/{bookingId}/transaction*
- ***HTTP METHOD****: POST*
- ***PathVariable*** *: int*
- ***RequestBody*** *: paymentMode, bookingId, upiId,cardNumber*
- ***Response****: ResponseEntity<BookingInfoEntity>*

Working

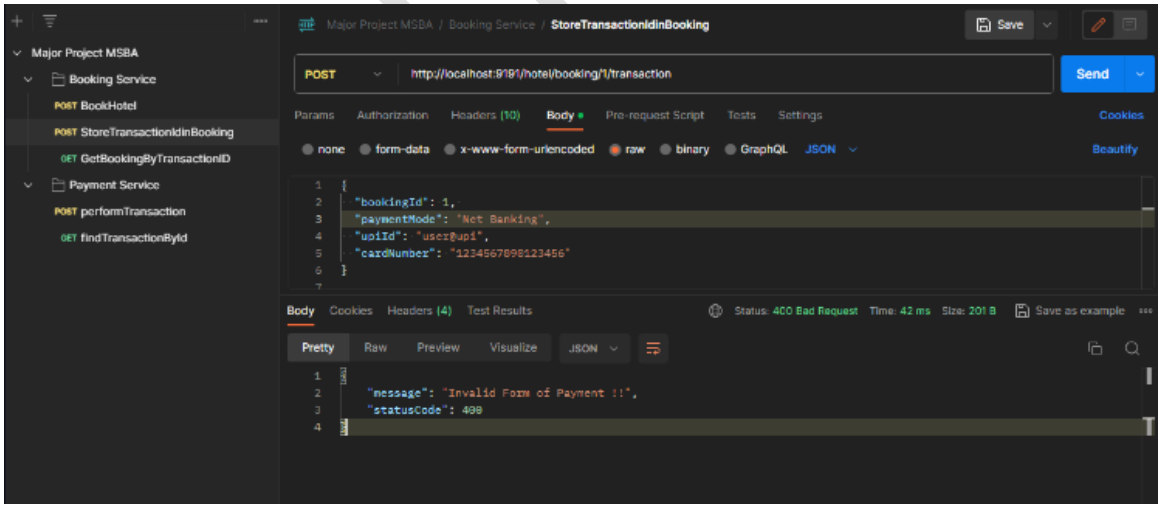

Major Project MSBA
Booking Service
POST BookHotel
POST StoreTransactionIdinBooking
GET GetBookingByTransactionID
Payment Service
POST performTransaction
GET findTransactionById
Major Project MSBA / Booking Service / StoreTransactionIdinBooking
Save
POST
http://localhost:9191/hotel/booking/1/transaction
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"bookingId": 1,
"paymentMode": "UPI",
"upiId": "user@upi",
"cardNumber": "1234567890123456"
}
Body Cookies Headers (5) Test Results
Status: 201 Created Time: 1183 ms Size: 415 B
Save as example
Pretty Raw Preview Visualize JSON
{
"bookingId": 1,
"fromDate": "2023-06-20T00:00:00.000+00:00",
"toDate": "2023-06-26T00:00:00.000+00:00",
"aadharNumber": "123456799015",
"numOfRooms": 2,
"roomNumbers": "25, 33",
"roomPrice": 12000,
"transactionId": 1,
"bookedOn": "2023-12-14T17:42:56.511+00:00"
}

Database

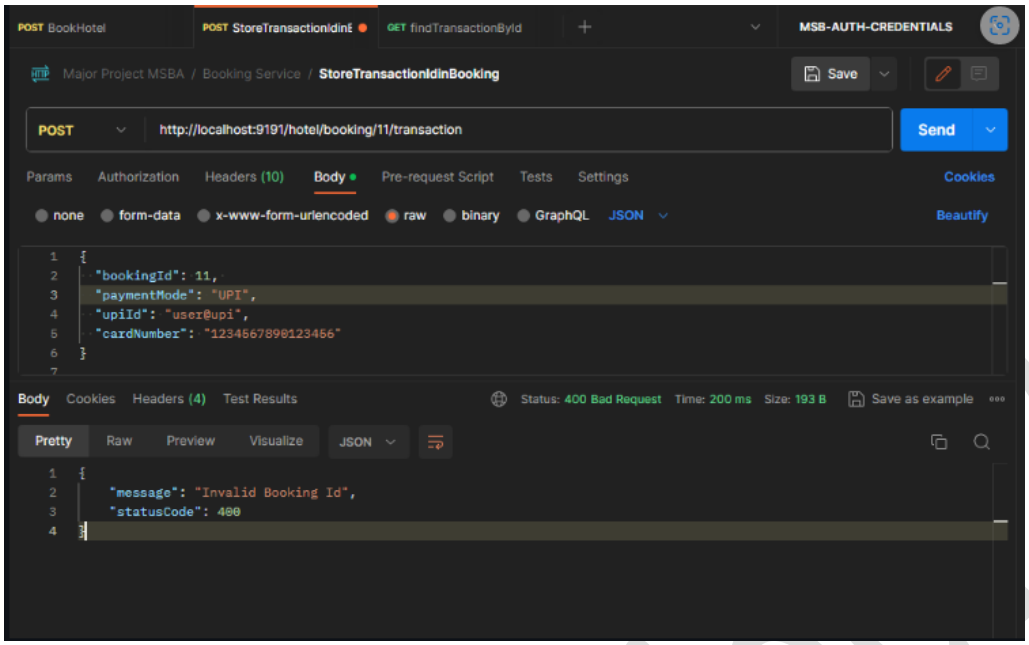

booking
Limit to 1000 rows
SELECT * FROM bookingsdb.booking;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
booking_id aadhar_number booked_on from_date num_of_rooms room_numbers room_price to_date transaction_id
1 123456799015 2023-12-14 23:12:56.511000 2023-06-20 05:30:00.000000 2 25, 33 12000 2023-06-26 05:30:00.000000 1



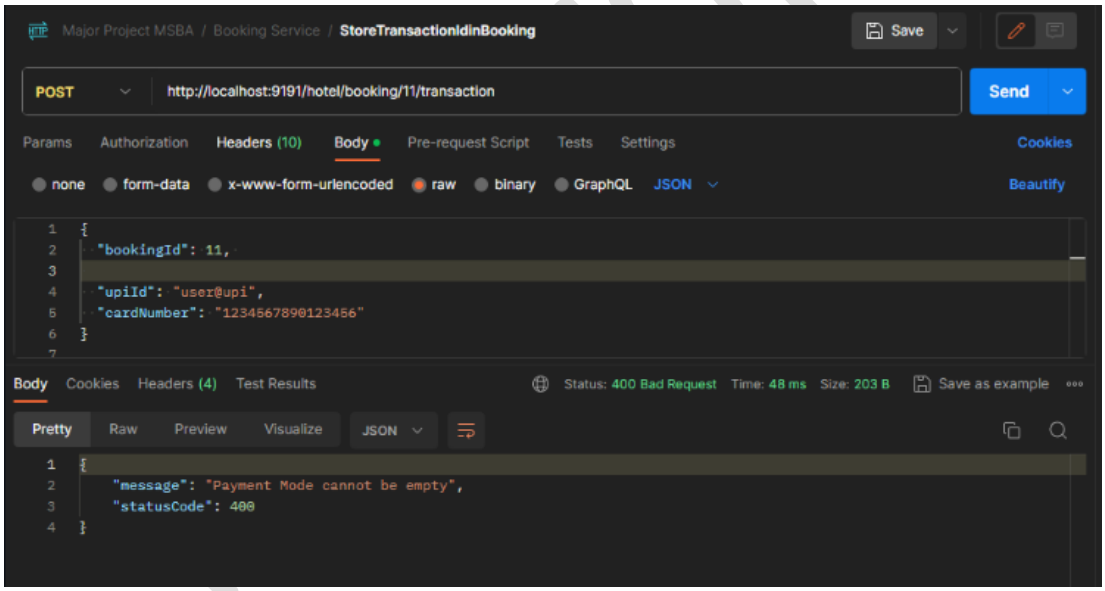
Exception 1 - If the user gives any other input apart from "UPI" or "CARD"



Exception 2 - If no transactions exist for the Booking Id passed to this endpoint



Exception 3 - If no Payment form is given



Exception 4 - If no Payment details is given for payment form

Major Project MSBA / Booking Service / **StoreTransactionIdinBooking**

Save

POST http://localhost:9191/hotel/booking/11/transaction Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
  "bookingId": 11,
  "paymentMode": "UPI",

  "cardNumber": "1234567890123456"
}
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 69 ms Size: 210 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "message": "UPI ID is required for UPI payments",
    "statusCode": 400
}
```

POST BookHotel | POST StoreTransactionIdinE | GET findTransactionById | MSB-AUTH-CREDENTIALS

Major Project MSBA / Booking Service / **StoreTransactionIdinBooking**

Save

POST http://localhost:9191/hotel/booking/11/transaction Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

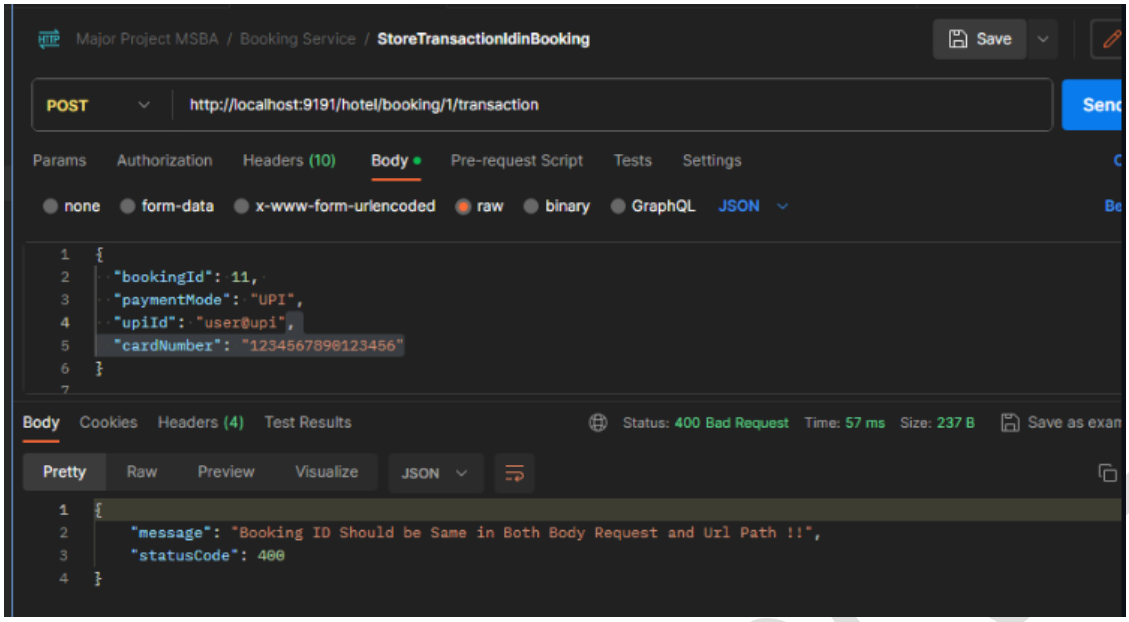
```
{
  "bookingId": 11,
  "paymentMode": "CARD",
  "upiId": "user@upi"
}
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 34 ms Size: 216 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "message": "Card Number is required for CARD payments",
    "statusCode": 400
}
```

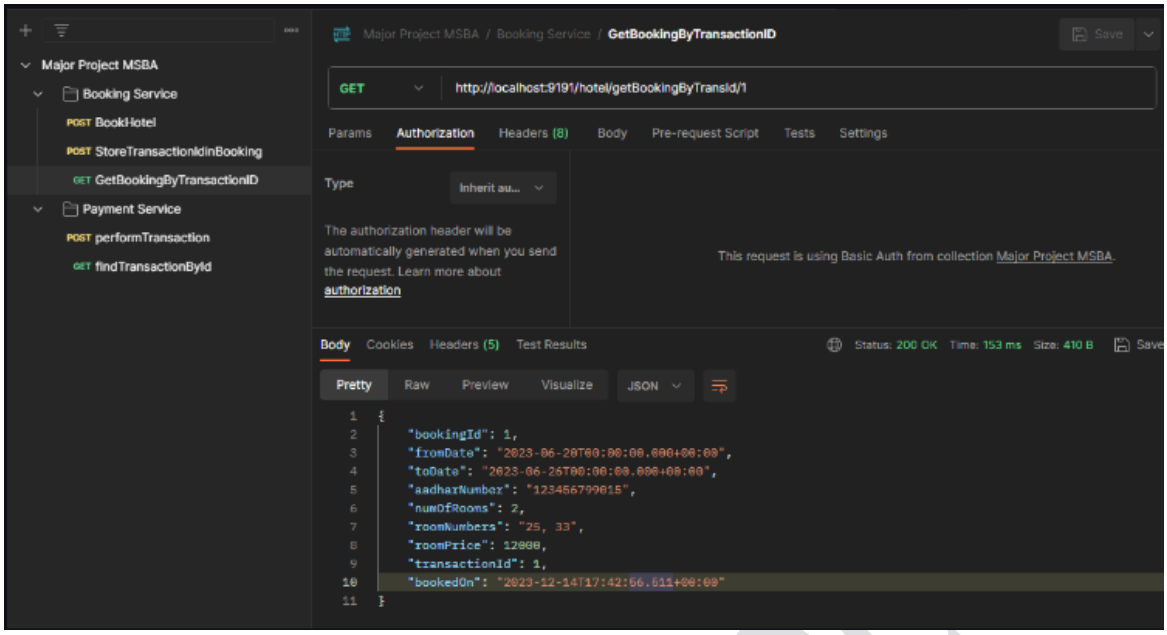
Exception 5 - If Booking Id is different in Path Variable and Request Body



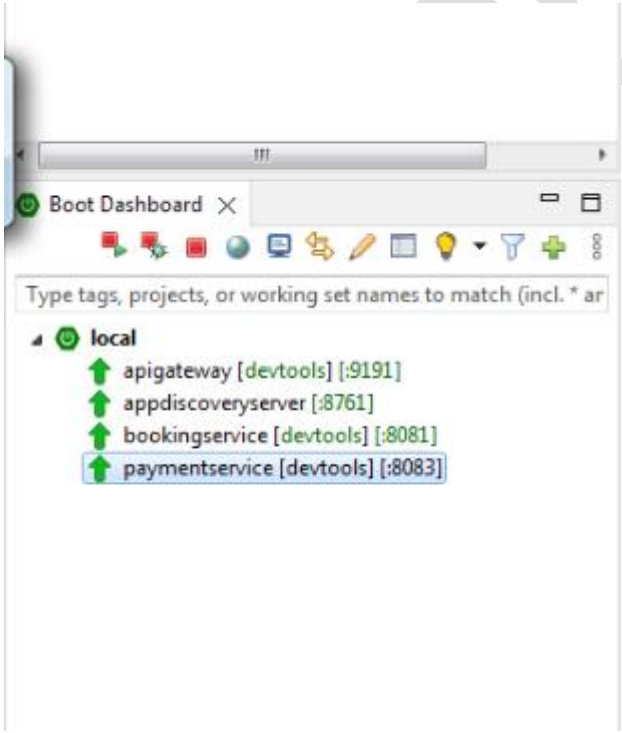
Endpoint 3: GET hotel/getBookingByTransId/{transactionId}

- ***URI:*** *hotel/getBookingByTransId/{transactionId}*
- ***HTTP METHOD:*** *GET*
- ***PathVariable*** *: int*
- ***Response****: ResponseEntity<BookingInfoEntity>*

Working



1.3 Configure this service to run on port number 8081.



1.4 Configure the hotel booking service as Eureka Client

[MainFile](bookingservice/src/main/java/com/bookingservice/BookingserviceApplication.java)

BookingserviceApplication.java

```java
package com.bookingservice;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
// Current Version Don't Need @EnableEurekaClient
public class BookingserviceApplication {

    public static void main(String[] args) {
        SpringApplication.run(BookingserviceApplication.class, args);
    }

    /*
     * @Bean public WebClient.Builder getWebClientBuilder() { return
     * WebClient.builder(); }
     */

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }

}
```

---

2. Payment Service : [Payment Service](paymentservice)

2.1. Model Classes -

[TransactionDetailsEntity](paymentservice/src/main/java/com/paymentservice/entity/TransactionDetailsEntity.java)

2.2. Controller -

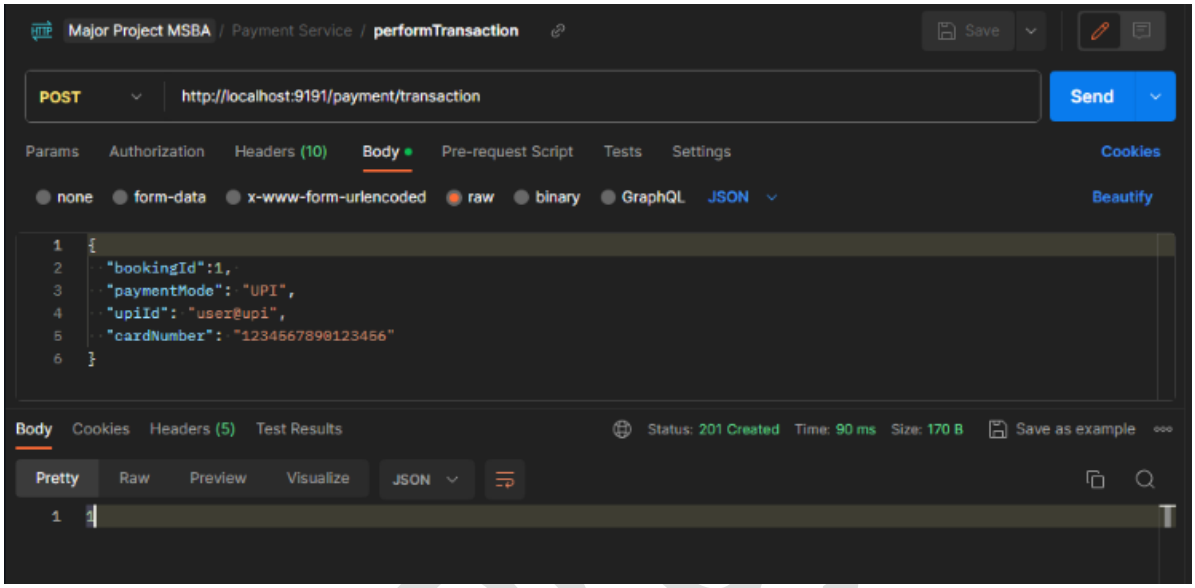
Note: This endpoint will be called by the 'endpoint 2' of the Booking service

Endpoint 1: POST /transaction

- ***URI***: */transaction*
- ***HTTP METHOD***: *POST*
- ***RequestBody***: *paymentMode, bookingId, upiId, cardNumber*

- ***Response Status**: Created*
- ***Response:** ResponseEntity<transactionId>*

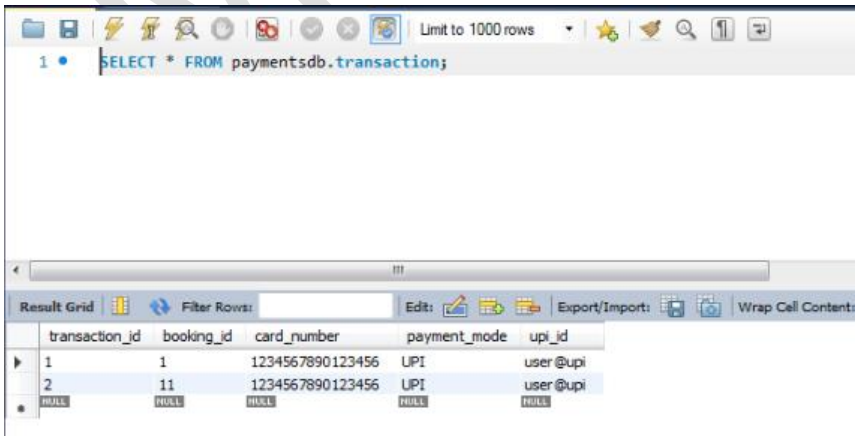
Working



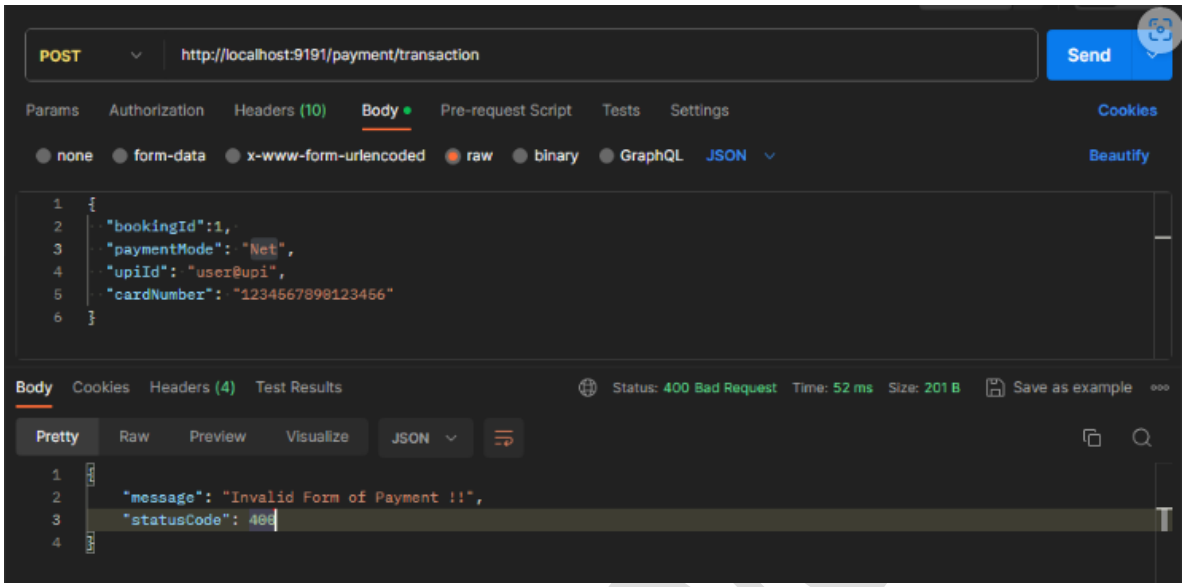
Console Message

```
Hibernate: select biel_0.booking_id,biel_0.aadhar_number,biel_0.booked_on,biel_0.from_date,biel_0.num_of_rooms,biel_0.room_numbers,biel_0.room_price,biel_0.to_date,biel
Hibernate: update booking set aadhar_number=?,booked_on=?,from_date=?,num_of_rooms=?,room_numbers=?,room_price=?,to_date=?,transaction_id=? where booking_id=?
Booking confirmed for user with aadhaar number: 123456799025 | Here are the booking details: BookingInfoEntity(bookingId=2, fromDate=2023-06-20 05:30:00.0, toDate=2023
```

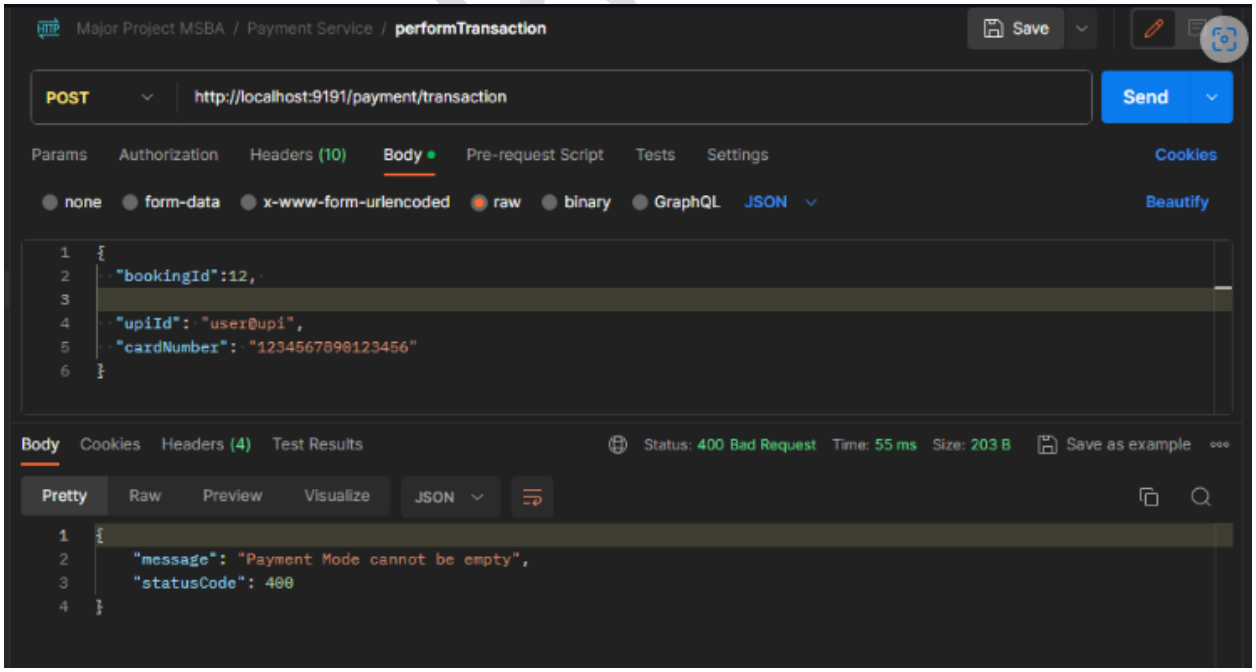
Database



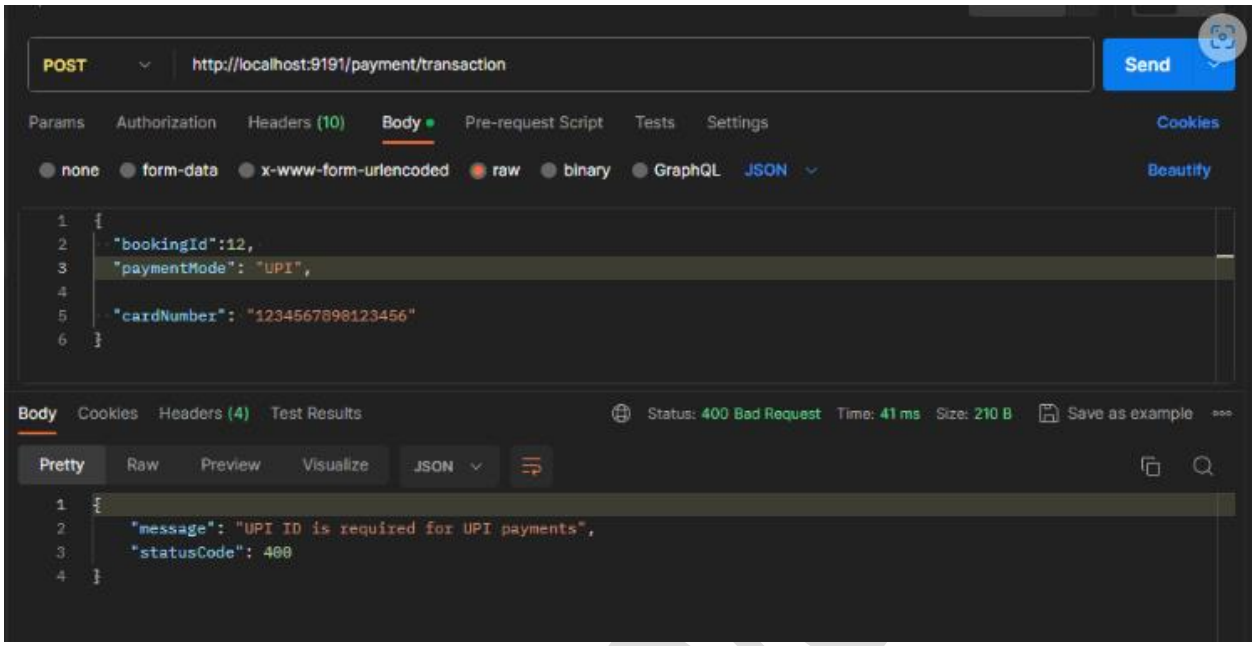
Exception 1 - If the user gives any other input apart from "UPI" or "CARD"



Exception 2 - If no Payment form is given



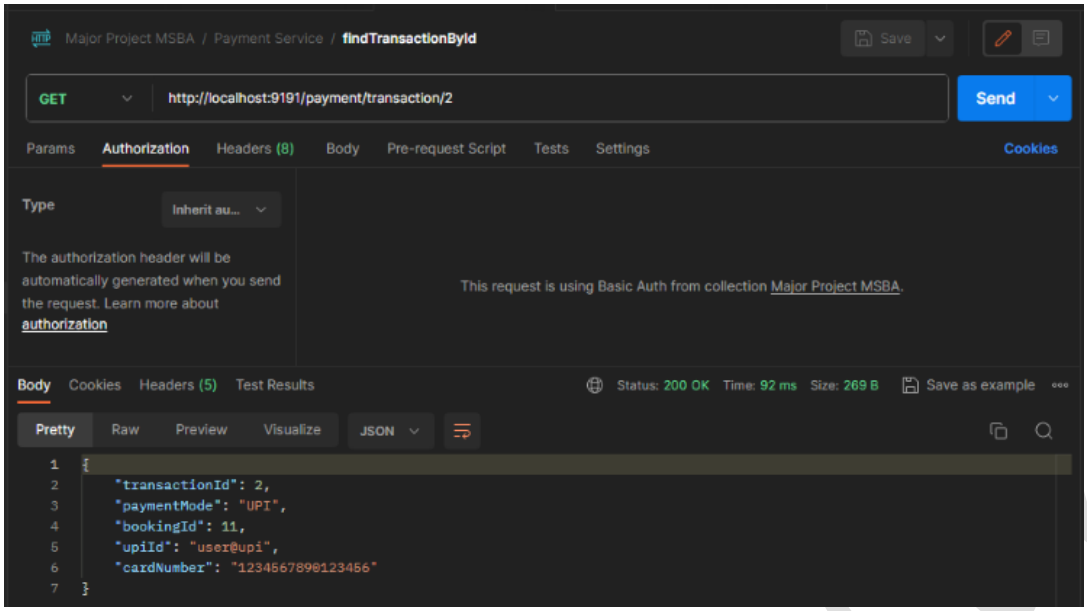
Exception 3 - If no Payment details is given for payment form



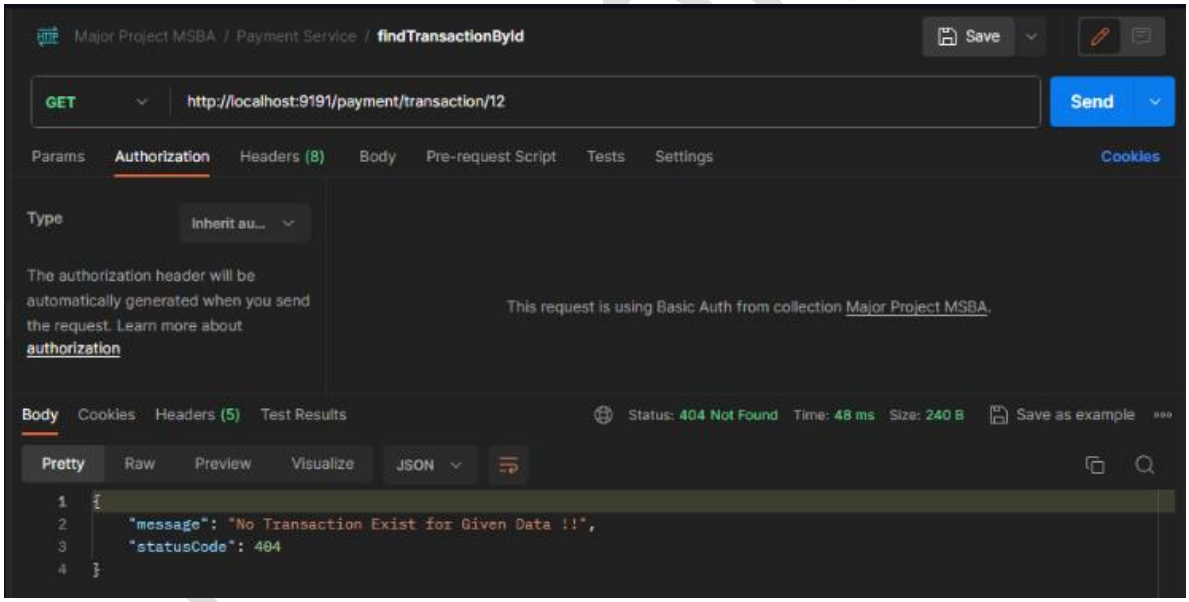
Endpoint 2: GET /transaction/{transactionId}

- ***URI:*** */transaction/{transactionId}*
- ***HTTP METHOD:*** *GET*
- ***Path Variable****: int*
- ***Response Status****: OK*
- ***Response:*** *ResponseEntity<TransactionDetailsEntity>*

Working



Exception 1 - If transaction Not Exists



2.3 Configure this service to run on port number 8083.

2.4 Configure the service as a Eureka client

```
TransactionCo...   RequestValidat...   BookingInfoCo...   RequestValidat...   Paymentservice...

1  package com.paymentservice;
2
3  import org.springframework.boot.SpringApplication;
7
8  @SpringBootApplication
9  // @EnableEurekaClient - No Need For Current Version
10 public class PaymentserviceApplication {
11
12     public static void main(String[] args) {
13         SpringApplication.run(PaymentserviceApplication.class, args);
14     }
15
16     @Bean
17     public WebClient.Builder getWebClientBuilder() {
18         return WebClient.builder();
19     }
20 }
21
```

---

8. Documentation Reference

8.1 -> All Documents are attached in ProblemStatement Folder

Name	Date modified	Type	Size
apigateway	12/19/2023 2:41 PM	File folder	
appdiscoveryserver	12/19/2023 2:41 PM	File folder	
bookingservice	12/19/2023 2:41 PM	File folder	
paymentservice	12/19/2023 2:41 PM	File folder	
ProblemStatement	12/19/2023 4:13 PM	File folder	
	12/19/2023 2:41 PM	Text Document	1 KB
database_setup	12/19/2023 2:41 PM	SQL Text File	1 KB
Major_Project_MSBA.postman_collection	12/19/2023 2:41 PM	JSON Source File	5 KB
MSB-AUTH-CREDENTIALS.postman_envi...	12/19/2023 2:41 PM	JSON Source File	1 KB
README	12/19/2023 2:41 PM	Markdown Source...	23 KB

Hotel+booking+schema.pdf

Service	Table Name	Columns	Datatypes	Constraints	Description
Booking Service	booking	bookingId	int (auto generated)	PRIMARY KEY	It refers to the "BookingId" of the user and is used to uniquely identify a booking.
		fromDate	Date	NULL	It refers to the date from which the user is looking for the room
		toDate	Date	NULL	It refers to the date untill when the user requires room
		aadharNumber	String	NULL	aadhar number of the user. It helps to uniquely identify the user.
		numOfRooms	int		It refers to the number of rooms required by user
		roomNumbers	String		It represents the list of room numbers allocated to the user
		roomPrice	int	NOT NULL	It refers to the total price of the allocated rooms for the requested days. Default value of a single room is Rs.1000. so if a user requests for 2 rooms for 2 days, then roomPrice will be Rs.4000.
		transactionId	int	DEFAULT=0	It refers to the transactionId which we get from the payment service.
		bookedOn	date	NULL	It refers to the current date
Payment Service	transaction	transactionId	int(auto generated)	PRIMARY KEY	It refers to the transaction Id and is used to uniquely identify a transaction.
		paymentMode	String		It refers to the user's mode of payment which can have values either as 'upi' or ' card'.
		booking Id	int	NOT NULL	It refers to the bookingId which we receive from the 'hotel booking service' when the payment service is called
		upi Id	String	NULL	If the user's mode of payment is 'upi', he has to provide the upiId and cardNumber must be null.
		cardNumber	String	NULL	If the user's mode of payment is 'card', he has to provide the cardNumber and upiId must be null.

NOTE:
1. NULL in the "Constraints" column mean that the variable can have null values and must be intialised to 'NULL' when there is no input from the user.

Microservices+Hotel+Room+Booking+App-1.pdf

Microservices Mini Project - 2

Hotel Room Booking Application

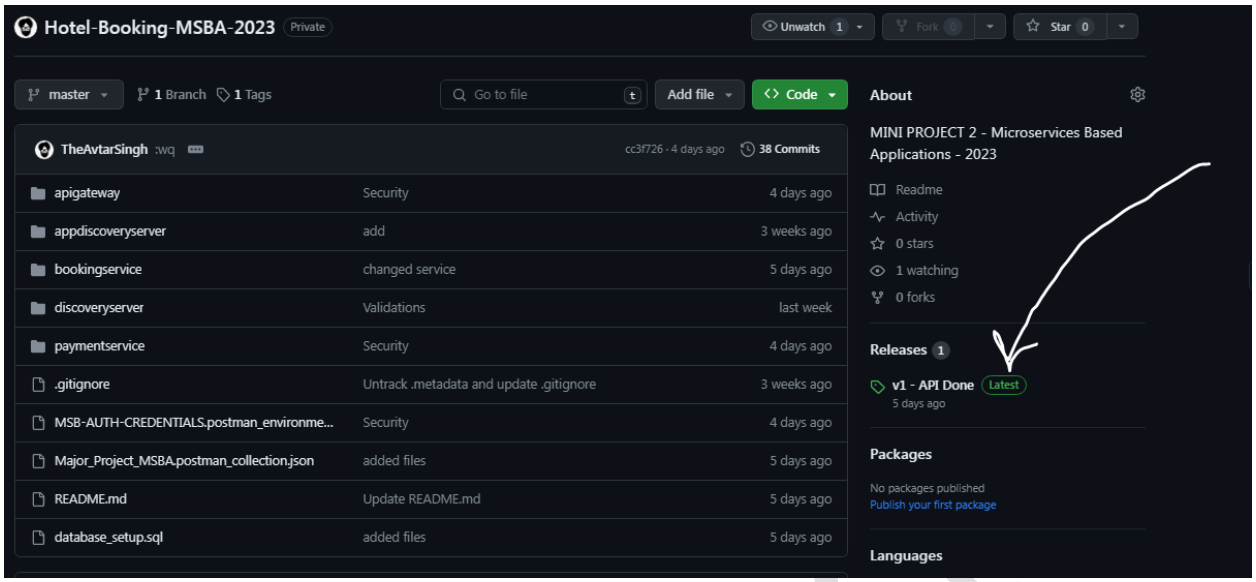
Project Overview

We are breaking the **Hotel room booking application** into three different microservices, which are as follows:

1. **API-Gateway** - This service is exposed to the outer world and is responsible for routing all requests to the microservices internally.
2. **Booking service** - This service is responsible for collecting all information related to user booking and sending a confirmation message once the booking is confirmed.
3. **Payment service** - This is a dummy payment service; this service is called by the booking service

9. ChangeLog

9.1 Current Version is in production only - v1



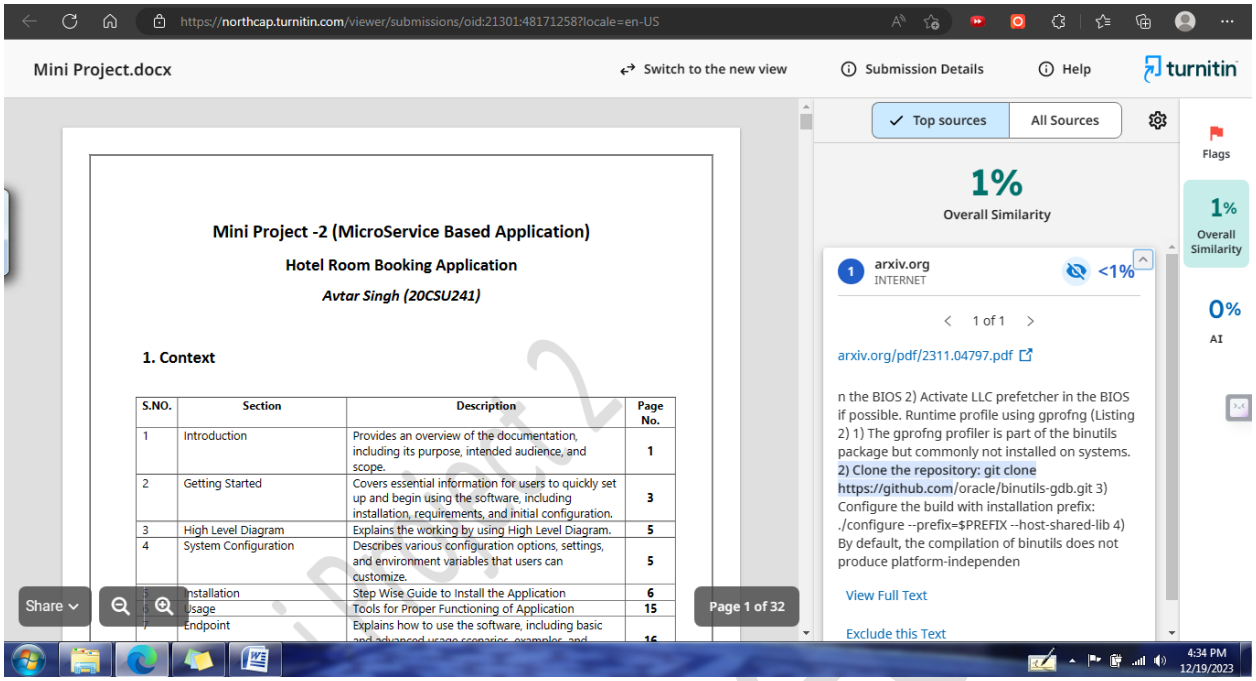
10. License

*This project is Copyrighted to **@TheAvtarSingh*** **[https://theavtarsingh.github.io]**.

11. References and Links

- **GITHUB - Hotel-Booking-MSBA-2023 -** https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023
- **POSTMAN ENVIROMENT FILE -** https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023/blob/master/Major_Project_MSBA.postman_collection.json
- **POSTMAN CREDENTAILS FILE -** https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023/blob/master/MSB-AUTH-CREDENTIALS.postman_environment.json
- **MYSQL DATA FILE -** https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023/blob/master/database_setup.sql
- **PROJECT README FILE -** https://github.com/TheAvtarSingh/Hotel-Booking-MSBA-2023/blob/master/README.md

12. Turnitin Plagiarism Report



https://northcap.turnitin.com/viewer/submissions/oid:21301:48171258?locale=en-US

Mini Project.docx | Switch to the new view | Submission Details | Help | turnitin

Mini Project -2 (MicroService Based Application)

Hotel Room Booking Application

Avtar Singh (20CSU241)

1. Context

S.NO.	Section	Description	Page No.
1	Introduction	Provides an overview of the documentation, including its purpose, intended audience, and scope.	1
2	Getting Started	Covers essential information for users to quickly set up and begin using the software, including installation, requirements, and initial configuration.	3
3	High Level Diagram	Explains the working by using High Level Diagram.	5
4	System Configuration	Describes various configuration options, settings, and environment variables that users can customize.	5
5	Installation	Step Wise Guide to Install the Application	6
6	Usage	Tools for Proper Functioning of Application	15
7	Endpoint	Explains how to use the software, including basic and advanced usage scenarios, examples, and	16

Share | Page 1 of 32

Top sources | All Sources

1%
Overall Similarity

1 arxiv.org
INTERNET
<1%

1 of 1

arxiv.org/pdf/2311.04797.pdf

n the BIOS 2) Activate LLC prefetcher in the BIOS if possible. Runtime profile using gprofng (Listing 2) 1) The gprofng profiler is part of the binutils package but commonly not installed on systems. 2) Clone the repository: git clone https://github.com/oracle/binutils-gdb.git 3) Configure the build with installation prefix: ./configure --prefix=$PREFIX --host-shared-lib 4) By default, the compilation of binutils does not produce platform-independen

View Full Text

Exclude this Text

Flags

1% Overall Similarity

0% AI

4:34 PM 12/19/2023

- ● -

End of Documentation

- ● -