Sweet-Home Application

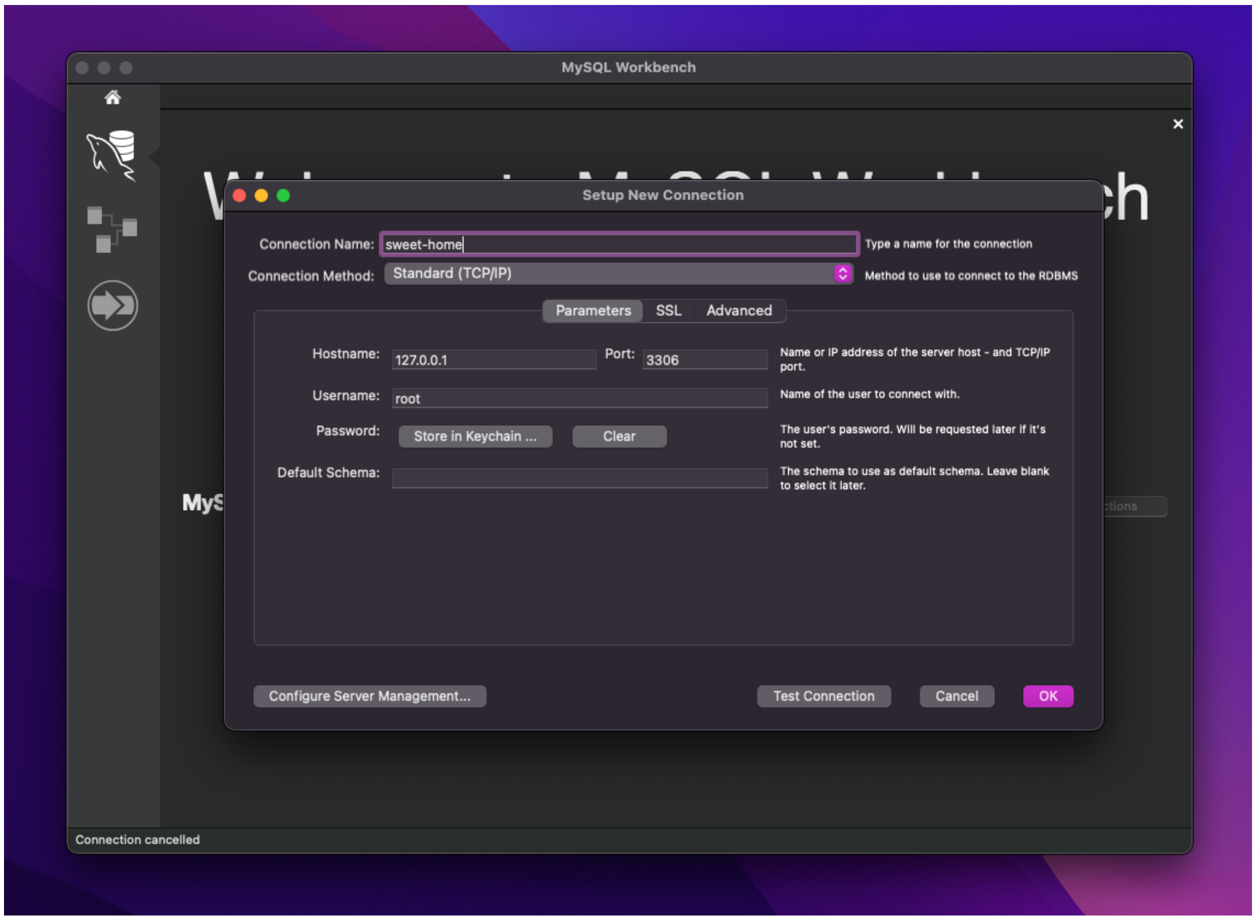
The Sweet-Home app has 3 services: **booking**, **payment**, and **eureka-server**. This app helps people book hotel rooms and pay for their bookings. The goal was to create model java microservices and follow the SOLID coding principles. The app was made using Spring Boot and Eureka, and MySQL was picked as the database.

Now, let's walk through the steps to run the project on your computer.

To keep things simple, we'll use **MySQL Workbench** to set up our database.

Steps to run project -

1. Open Workbench, make a connection named **'sweet-home'**, and provide any **username** and **password**.



Setup New Connection

Connection Name: sweet-home — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - and TCP/IP port.

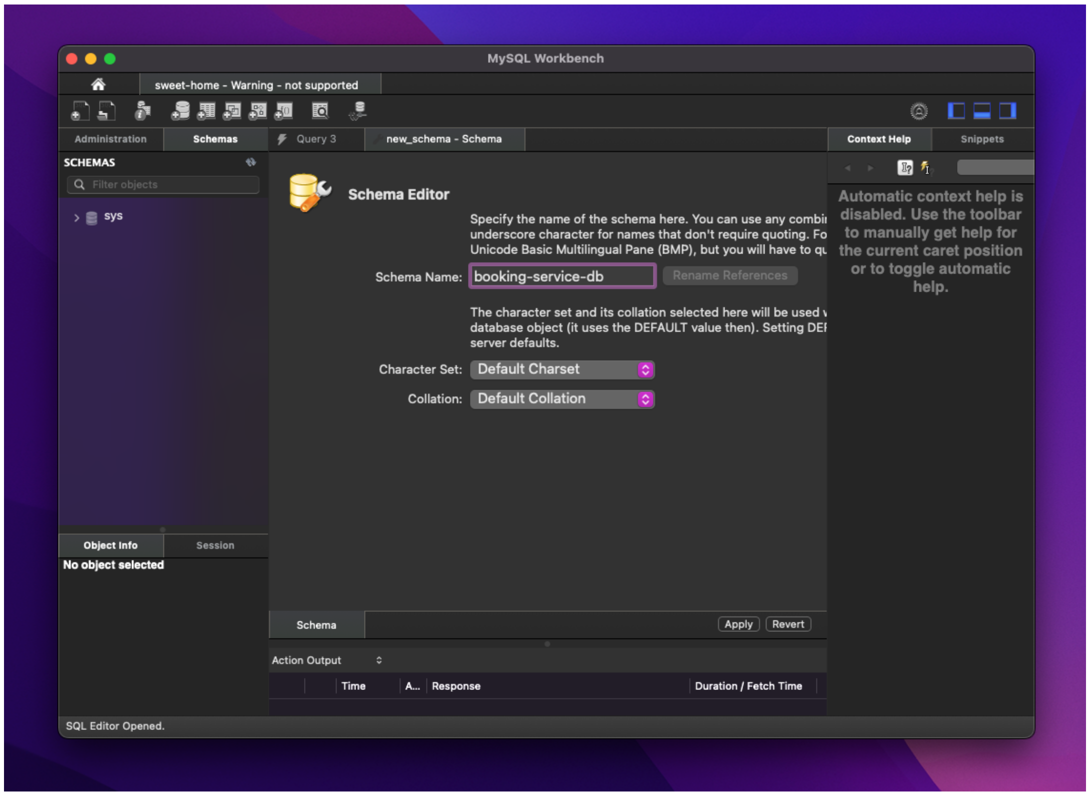
Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

Connection cancelled

2. Navigate to **"Add New Schema"** and create 2 schemas: **"booking-service-db"** and **"payment-service-db"**.



Specify the name of the schema here. You can use any combi
underscore character for names that don't require quoting. Fo
Unicode Basic Multilingual Pane (BMP), but you will have to qu

Schema Name: booking-service-db

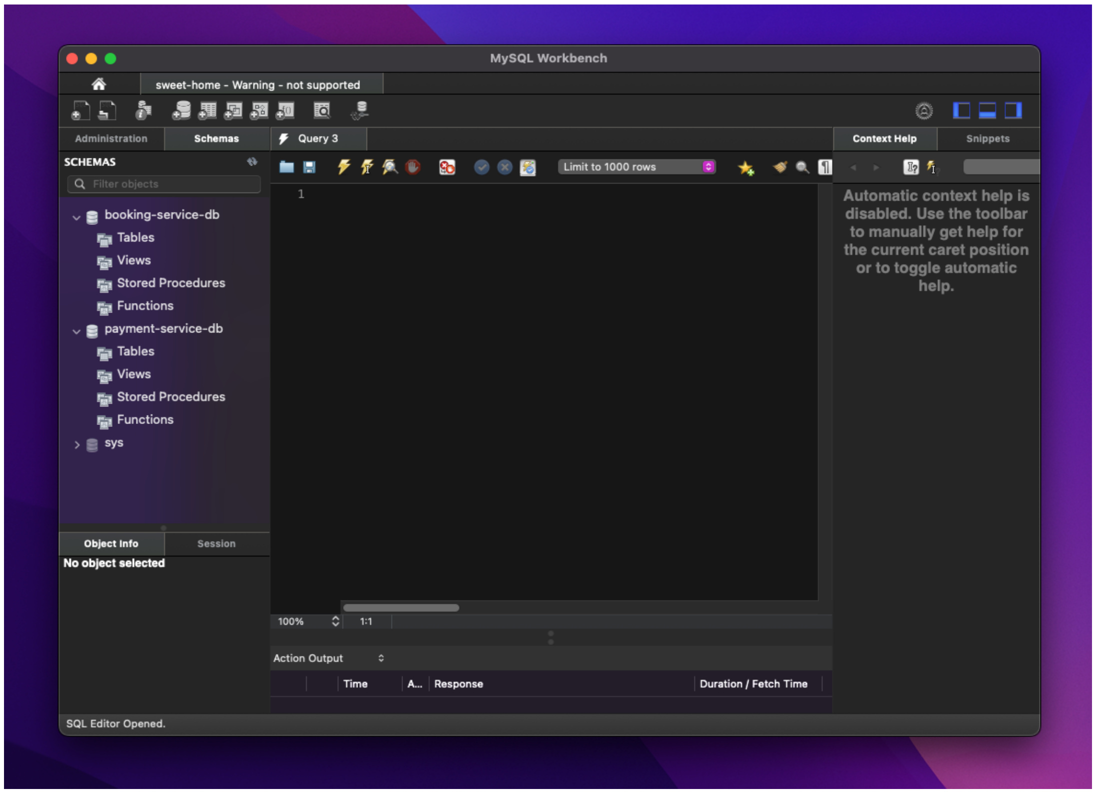
Rename References

The character set and its collation selected here will be used v
database object (it uses the DEFAULT value then). Setting DEI
server defaults.

Character Set: Default Charset

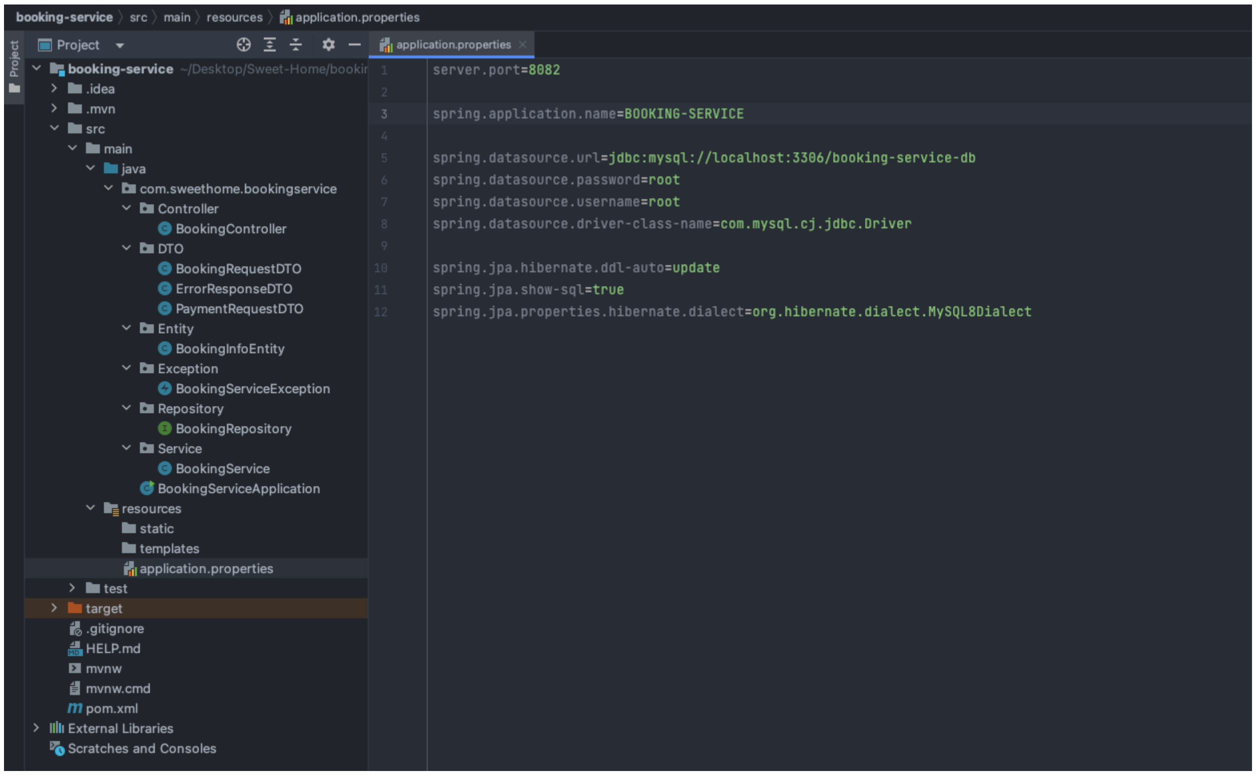
Collation: Default Collation

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.



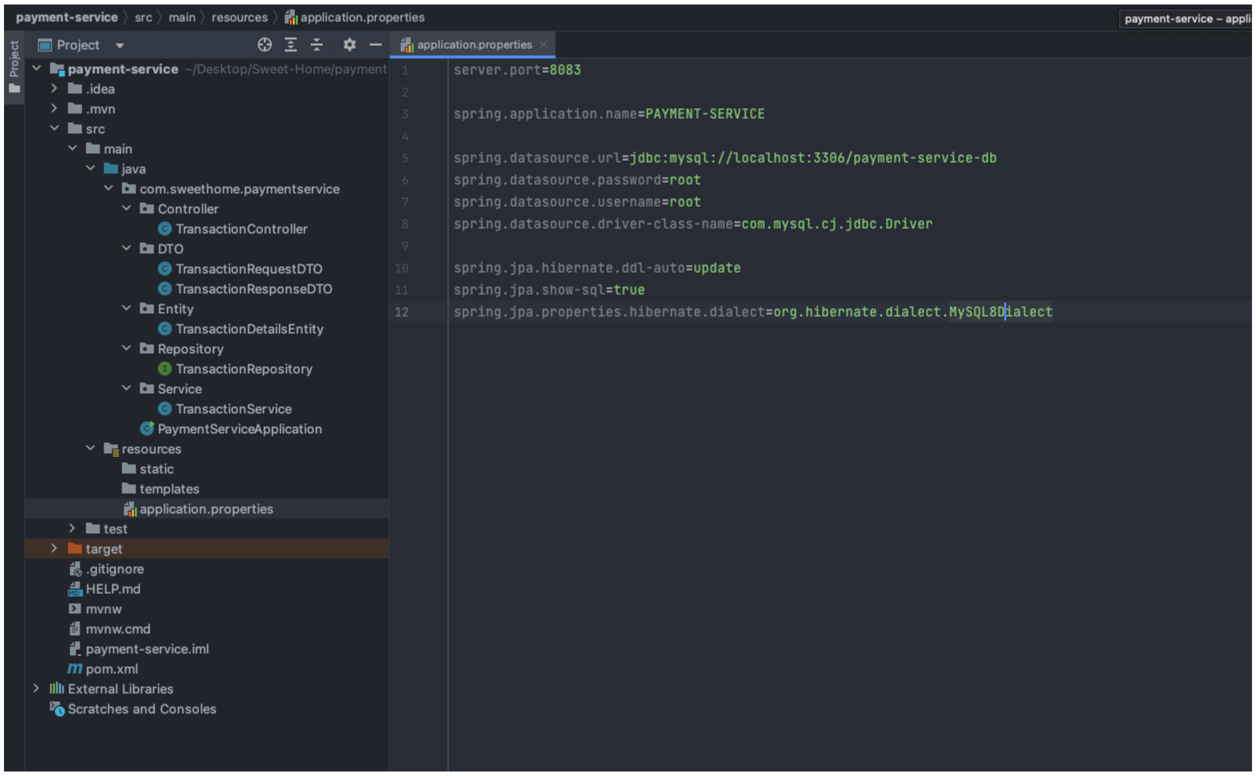
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

3. Go to **"src/main/resources/application.properties"** (on both booking & payment services) and enter your database **URL**, **username**, and **password**.


booking-service 〉 src 〉 main 〉 resources 〉 application.properties
Project
application.properties

```
booking-service ~/Desktop/Sweet-Home/bookir
  .idea
  .mvn
  src
    main
      java
        com.sweethome.bookingservice
          Controller
            BookingController
          DTO
            BookingRequestDTO
            ErrorResponseDTO
            PaymentRequestDTO
          Entity
            BookingInfoEntity
          Exception
            BookingServiceException
          Repository
            BookingRepository
          Service
            BookingService
          BookingServiceApplication
      resources
        static
        templates
        application.properties
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

```
server.port=8082

spring.application.name=BOOKING-SERVICE

spring.datasource.url=jdbc:mysql://localhost:3306/booking-service-db
spring.datasource.password=root
spring.datasource.username=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```


payment-service 〉 src 〉 main 〉 resources 〉 application.properties
payment-service – appli
Project
application.properties

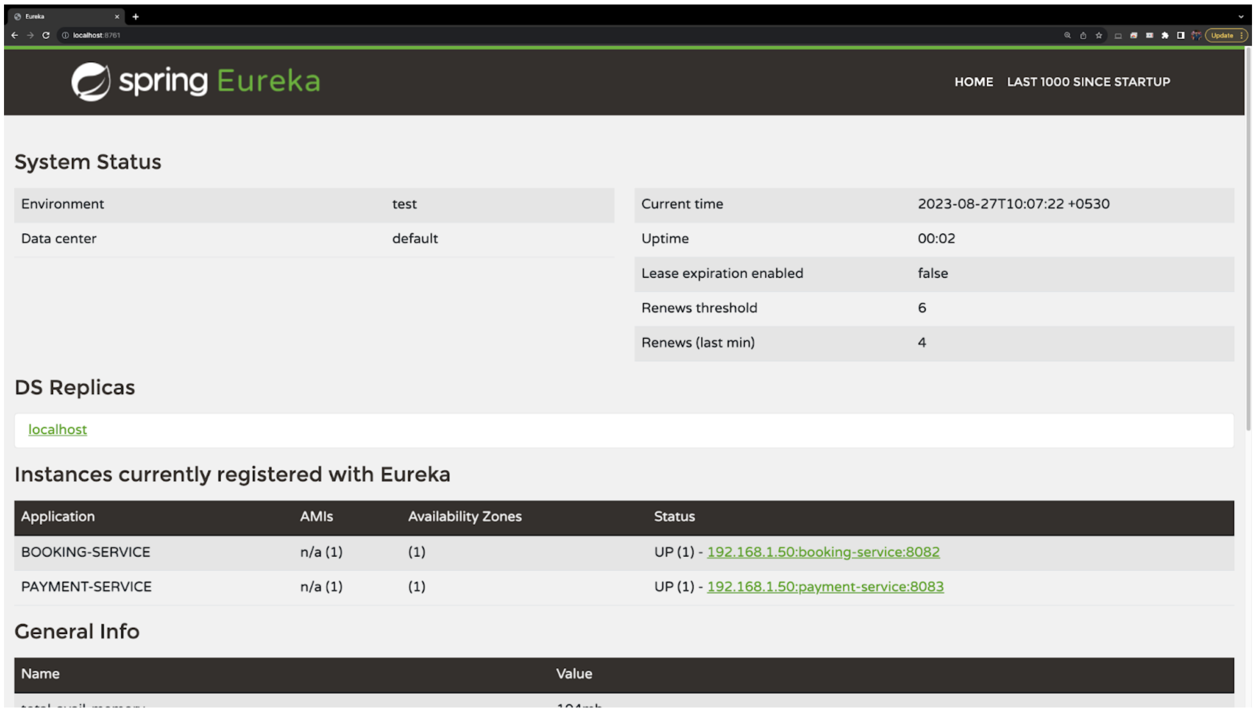
```
payment-service ~/Desktop/Sweet-Home/payment
  .idea
  .mvn
  src
    main
      java
        com.sweethome.paymentservice
          Controller
            TransactionController
          DTO
            TransactionRequestDTO
            TransactionResponseDTO
          Entity
            TransactionDetailsEntity
          Repository
            TransactionRepository
          Service
            TransactionService
          PaymentServiceApplication
      resources
        static
        templates
        application.properties
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  payment-service.iml
  pom.xml
External Libraries
Scratches and Consoles
```

```
server.port=8083

spring.application.name=PAYMENT-SERVICE

spring.datasource.url=jdbc:mysql://localhost:3306/payment-service-db
spring.datasource.password=root
spring.datasource.username=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

4. Go to the root directory of the Eureka server and run the command: **'mvn spring-boot:run'**.

5. Now do the same for both **payment** and **booking** service.

6. Open your web browser and go to **http://localhost:8761**. Here, you should be able to see the records for both the booking service and payment service.



System Status

Environment	test	Current time	2023-08-27T10:07:22 +0530
Data center	default	Uptime	00:02
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	4

DS Replicas

localhost

Instances currently registered with Eureka

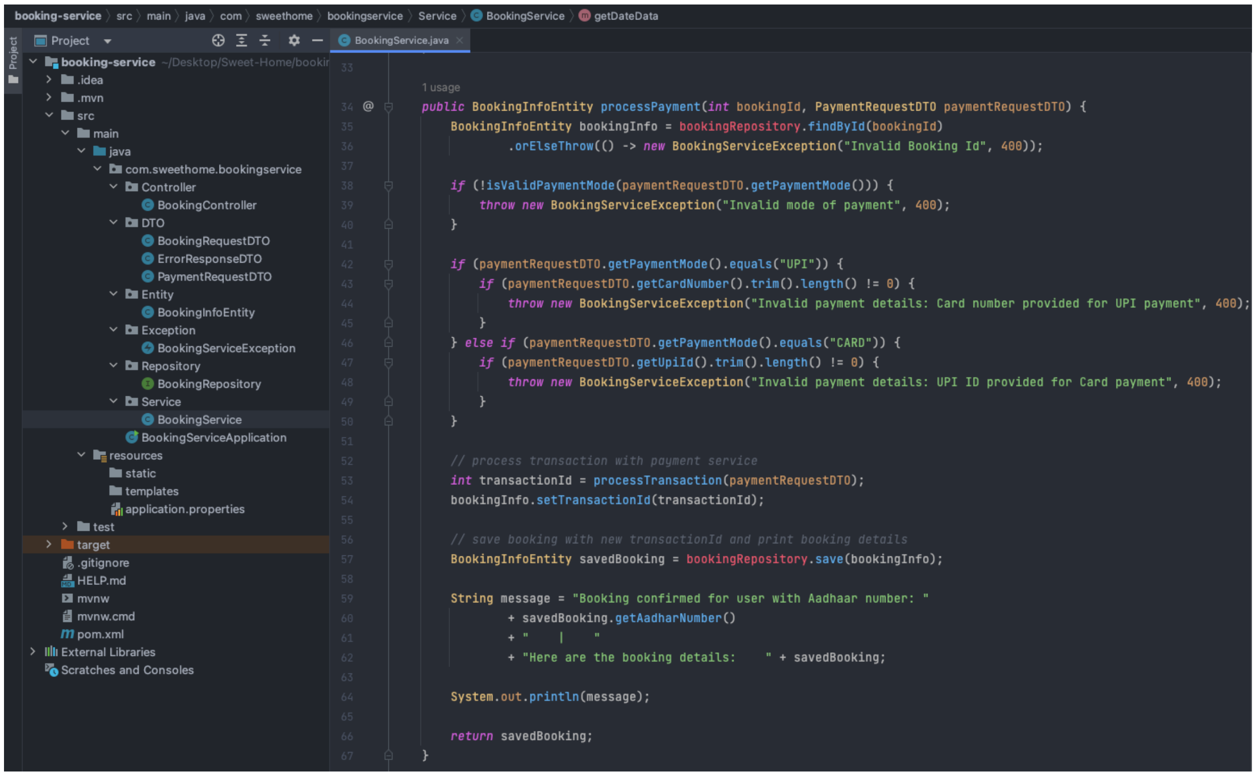
Application	AMIs	Availability Zones	Status
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.50:booking-service:8082
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.50:payment-service:8083

General Info

Name	Value

Code Logic -

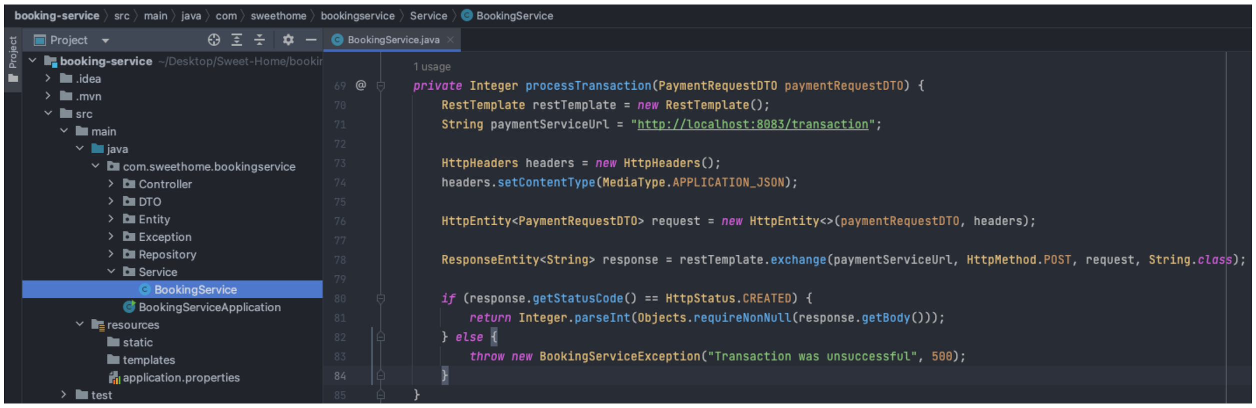
1. src/main/java/com/sweethome/bookingservice/Service/BookingService.java/processPayment()



```java
public BookingInfoEntity processPayment(int bookingId, PaymentRequestDTO paymentRequestDTO) {
    BookingInfoEntity bookingInfo = bookingRepository.findById(bookingId)
            .orElseThrow(() -> new BookingServiceException("Invalid Booking Id", 400));

    if (!isValidPaymentMode(paymentRequestDTO.getPaymentMode())) {
        throw new BookingServiceException("Invalid mode of payment", 400);
    }

    if (paymentRequestDTO.getPaymentMode().equals("UPI")) {
        if (paymentRequestDTO.getCardNumber().trim().length() != 0) {
            throw new BookingServiceException("Invalid payment details: Card number provided for UPI payment", 400);
        }
    } else if (paymentRequestDTO.getPaymentMode().equals("CARD")) {
        if (paymentRequestDTO.getUpiId().trim().length() != 0) {
            throw new BookingServiceException("Invalid payment details: UPI ID provided for Card payment", 400);
        }
    }

    // process transaction with payment service
    int transactionId = processTransaction(paymentRequestDTO);
    bookingInfo.setTransactionId(transactionId);

    // save booking with new transactionId and print booking details
    BookingInfoEntity savedBooking = bookingRepository.save(bookingInfo);

    String message = "Booking confirmed for user with Aadhaar number: "
            + savedBooking.getAadharNumber()
            + "    |    "
            + "Here are the booking details:    " + savedBooking;

    System.out.println(message);

    return savedBooking;
}
```

This code snippet defines a method named **processPayment** that takes a **booking ID** and a **PaymentRequestDTO** object as parameters.
It retrieves booking information from a repository using the provided booking ID. If the payment mode in the DTO is invalid, it throws an **exception**. Depending on the payment mode ("**UPI**" or "**CARD**"), it checks for valid payment details and throws exceptions if inconsistencies are found.
The method then processes a transaction using a payment service and updates the **transaction ID** in the booking information.
The updated booking information is saved and a confirmation message, along with the booking details, is printed. Finally, the updated booking information is returned.
This code essentially handles payment processing and validation for booking transactions.

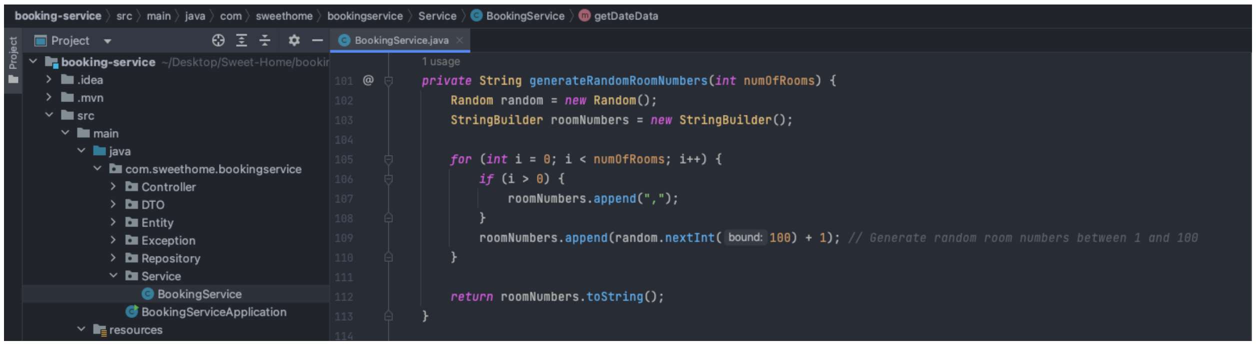
2. src/main/java/com/sweethome/bookingservice/Service/BookingService.java/ processTransaction()


booking-service 〉 src 〉 main 〉 java 〉 com 〉 sweethome 〉 bookingservice 〉 Service 〉 BookingService
Project
BookingService.java
booking-service ~/Desktop/Sweet-Home/bookir
.idea
.mvn
src
main
java
com.sweethome.bookingservice
Controller
DTO
Entity
Exception
Repository
Service
BookingService
BookingServiceApplication
resources
static
templates
application.properties
test
1 usage

```java
private Integer processTransaction(PaymentRequestDTO paymentRequestDTO) {
    RestTemplate restTemplate = new RestTemplate();
    String paymentServiceUrl = "http://localhost:8083/transaction";

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    HttpEntity<PaymentRequestDTO> request = new HttpEntity<>(paymentRequestDTO, headers);

    ResponseEntity<String> response = restTemplate.exchange(paymentServiceUrl, HttpMethod.POST, request, String.class);

    if (response.getStatusCode() == HttpStatus.CREATED) {
        return Integer.parseInt(Objects.requireNonNull(response.getBody()));
    } else {
        throw new BookingServiceException("Transaction was unsuccessful", 500);
    }
}
```

This code snippet defines a private method named **processTransaction** that takes a **PaymentRequestDTO** object as a parameter and returns an Integer representing a **transaction ID.**
Inside the method, a **RestTemplate** is created to make an HTTP POST request to a **payment service's URL** (http://localhost:8083/transaction).
Headers are set to indicate that the content type is JSON. The PaymentRequestDTO object is included in the HTTP request's body. The method then sends the HTTP request and receives a response in the form of a **ResponseEntity<String>**.
If the response status code indicates a successful creation (**HTTP 201**), the transaction ID from the response body is extracted and returned as an integer. If the response status code is not as expected, an exception is thrown, indicating that the transaction was unsuccessful.
This code is responsible for sending payment information to a payment service and handling the response.

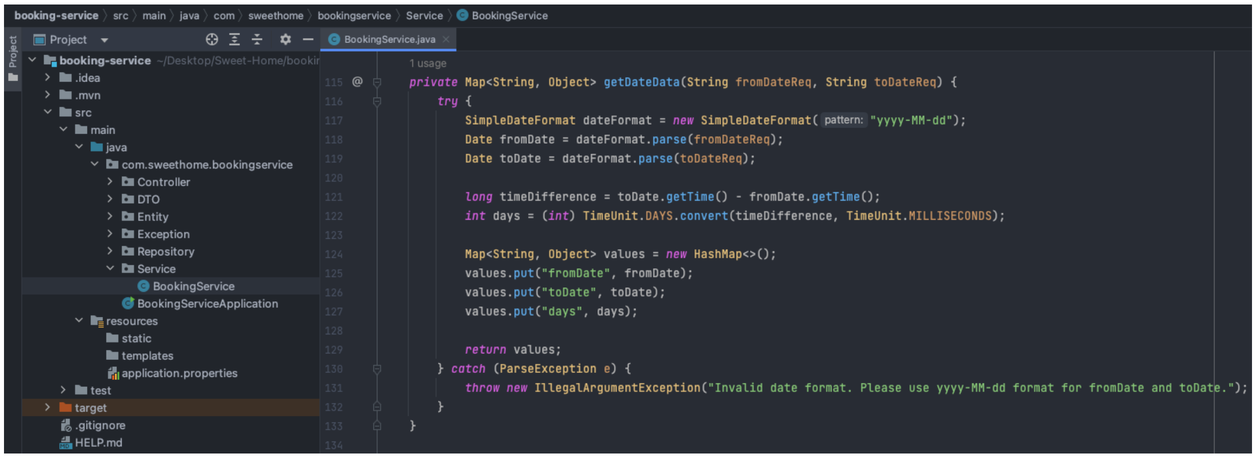
3. src/main/java/com/sweethome/bookingservice/Service/BookingService.java/ generateRandomRoomNumbers()


booking-service 〉 src 〉 main 〉 java 〉 com 〉 sweethome 〉 bookingservice 〉 Service 〉 BookingService 〉 getDateData
Project
BookingService.java
booking-service ~/Desktop/Sweet-Home/bookir
.idea
.mvn
src
main
java
com.sweethome.bookingservice
Controller
DTO
Entity
Exception
Repository
Service
BookingService
BookingServiceApplication
resources
1 usage

```java
private String generateRandomRoomNumbers(int numOfRooms) {
    Random random = new Random();
    StringBuilder roomNumbers = new StringBuilder();

    for (int i = 0; i < numOfRooms; i++) {
        if (i > 0) {
            roomNumbers.append(",");
        }
        roomNumbers.append(random.nextInt( bound: 100) + 1); // Generate random room numbers between 1 and 100
    }

    return roomNumbers.toString();
}
```

This code snippet defines a private method named **generateRandomRoomNumbers** that takes an integer **numOfRooms** as a parameter and returns a string representing a **list of randomly generated room numbers**.
Inside the method, a Random instance is created to generate random numbers. A StringBuilder named roomNumbers is used to build the resulting string. The method then loops numOfRooms times, generating a random number between 1 and 100 for each iteration.
These numbers are appended to the **roomNumbers** string, separated by commas. The generated string of room numbers is then returned.
This code is responsible for generating a specified number of random room numbers for use, such as in a hotel room allocation system.

4. src/main/java/com/sweethome/bookingservice/Service/BookingService.java/ generateRandomRoomNumbers()

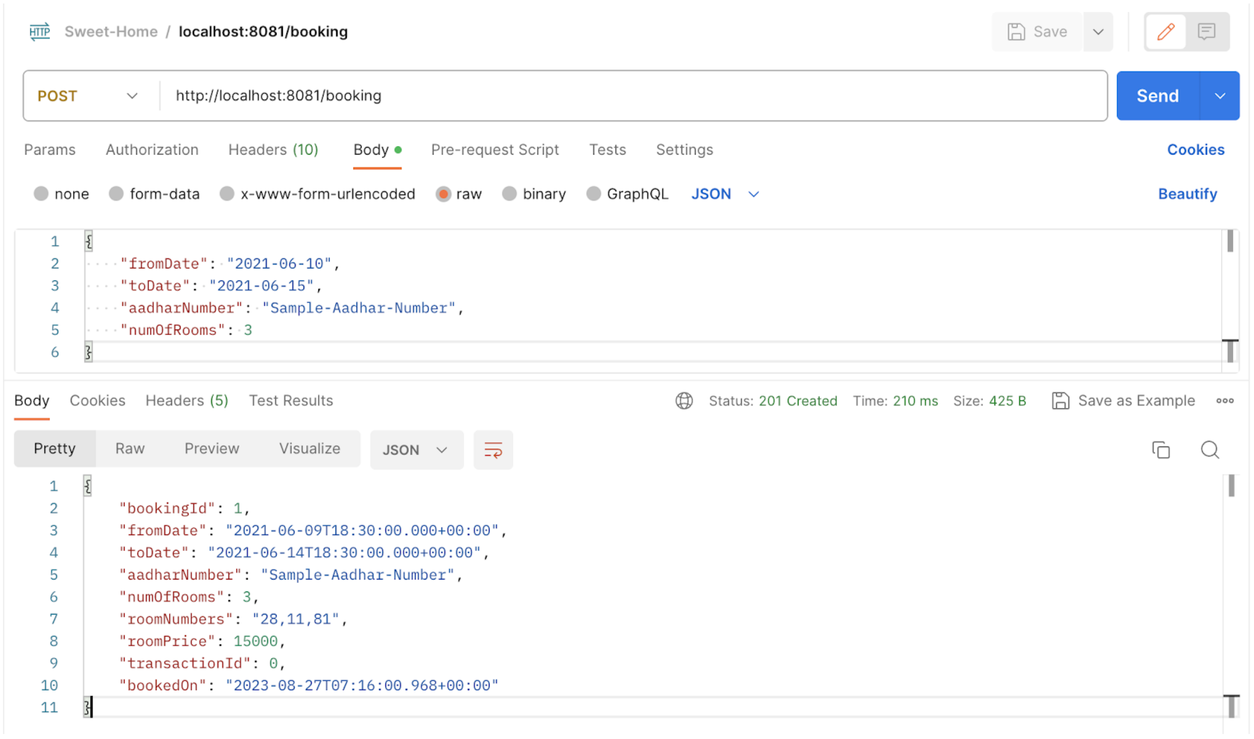

booking-service 〉 src 〉 main 〉 java 〉 com 〉 sweethome 〉 bookingservice 〉 Service 〉 BookingService
Project
booking-service ~/Desktop/Sweet-Home/bookir
.idea
.mvn
src
main
java
com.sweethome.bookingservice
Controller
DTO
Entity
Exception
Repository
Service
BookingService
BookingServiceApplication
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
BookingService.java
1 usage

```java
private Map<String, Object> getDateData(String fromDateReq, String toDateReq) {
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
        Date fromDate = dateFormat.parse(fromDateReq);
        Date toDate = dateFormat.parse(toDateReq);

        long timeDifference = toDate.getTime() - fromDate.getTime();
        int days = (int) TimeUnit.DAYS.convert(timeDifference, TimeUnit.MILLISECONDS);

        Map<String, Object> values = new HashMap<>();
        values.put("fromDate", fromDate);
        values.put("toDate", toDate);
        values.put("days", days);

        return values;
    } catch (ParseException e) {
        throw new IllegalArgumentException("Invalid date format. Please use yyyy-MM-dd format for fromDate and toDate.");
    }
}
```

This code snippet defines a private method named **getDateData** that takes two date strings (**fromDateReq** and **toDateReq**) as parameters and **returns a Map** containing relevant date-related information. Inside the method, it attempts to parse the input date strings using the "yyyy-MM-dd" format. If the parsing is successful, it calculates the time difference in milliseconds between the two parsed dates and then converts that difference into the **number of days** using the TimeUnit.DAYS conversion.
A Map named values is created to store the parsed dates and calculated days. The parsed **fromDate, toDate, and days** are put into the values map with corresponding keys. If there's a parsing error (if the date strings are not in the expected format), an exception is caught and an IllegalArgumentException is thrown, indicating that the date format is invalid.
This code is responsible for parsing and processing date strings to calculate the time difference between them and provide the result in terms of days along with the parsed dates.

Endpoint postman tests -

1. localhost:8081/booking



Sweet-Home / localhost:8081/booking
Save
Send
Params Authorization Headers (10) 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 as Example
Pretty Raw Preview Visualize JSON

POST http://localhost:8081/booking

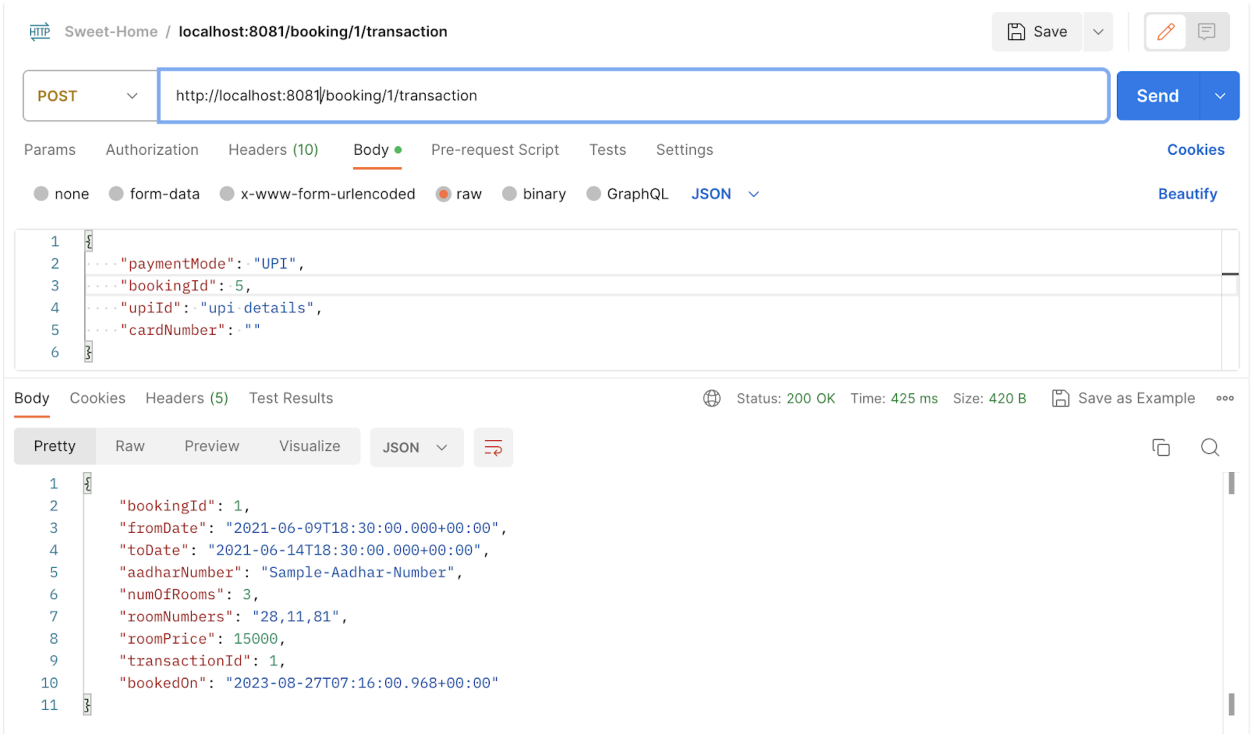
Body: raw, JSON

```
{
    "fromDate": "2021-06-10",
    "toDate": "2021-06-15",
    "aadharNumber": "Sample-Aadhar-Number",
    "numOfRooms": 3
}
```

Status: 201 Created Time: 210 ms Size: 425 B

```
{
    "bookingId": 1,
    "fromDate": "2021-06-09T18:30:00.000+00:00",
    "toDate": "2021-06-14T18:30:00.000+00:00",
    "aadharNumber": "Sample-Aadhar-Number",
    "numOfRooms": 3,
    "roomNumbers": "28,11,81",
    "roomPrice": 15000,
    "transactionId": 0,
    "bookedOn": "2023-08-27T07:16:00.968+00:00"
}
```

2. localhost:8081/booking/1/transaction



Sweet-Home / localhost:8081/booking/1/transaction
Save
Send
Params Authorization Headers (10) 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 as Example
Pretty Raw Preview Visualize JSON

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

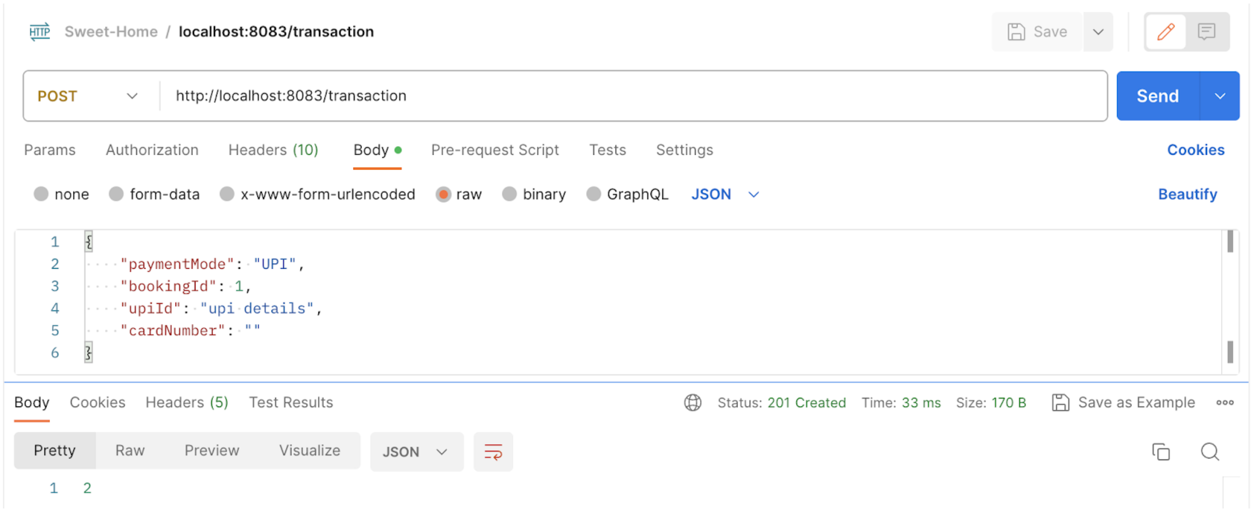
Body: raw, JSON

```
{
    "paymentMode": "UPI",
    "bookingId": 5,
    "upiId": "upi details",
    "cardNumber": ""
}
```

Status: 200 OK Time: 425 ms Size: 420 B

```
{
    "bookingId": 1,
    "fromDate": "2021-06-09T18:30:00.000+00:00",
    "toDate": "2021-06-14T18:30:00.000+00:00",
    "aadharNumber": "Sample-Aadhar-Number",
    "numOfRooms": 3,
    "roomNumbers": "28,11,81",
    "roomPrice": 15000,
    "transactionId": 1,
    "bookedOn": "2023-08-27T07:16:00.968+00:00"
}
```

3. localhost:8083/transaction



Sweet-Home / localhost:8083/transaction
Save
POST
Send
Params Authorization Headers (10) 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 as Example
Pretty Raw Preview Visualize JSON

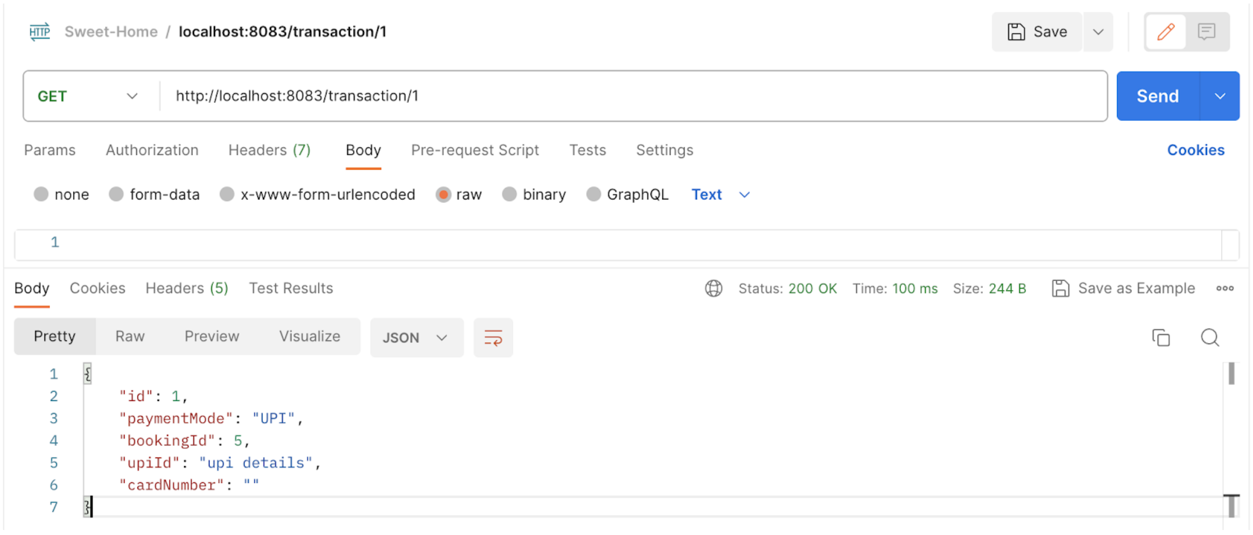
POST http://localhost:8083/transaction

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

Status: 201 Created Time: 33 ms Size: 170 B

```
2
```

4. localhost:8083/transaction/1



Sweet-Home / localhost:8083/transaction/1
Save
GET
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL Text
Body Cookies Headers (5) Test Results
Save as Example
Pretty Raw Preview Visualize JSON

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

Status: 200 OK Time: 100 ms Size: 244 B

```
{
    "id": 1,
    "paymentMode": "UPI",
    "bookingId": 5,
    "upiId": "upi details",
    "cardNumber": ""
}
```