

Sweet-Home documentation

Eureka server and client setup

Booking and Payment service are registered here at port no 8761

<http://localhost:8761/>

The screenshot shows the Spring Eureka web interface. The top navigation bar includes the Spring Eureka logo and links for HOME and LAST 1000 SINCE STARTUP. The main content area is divided into three sections: System Status, DS Replicas, and Instances currently registered with Eureka.

System Status

Environment	N/A	Current time	2022-07-10T16:32:40 +0530
Data center	N/A	Uptime	00:44
		Lease expiration enabled	true
		Renews threshold	5
		Renews (last min)	8

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-4QBKNEGT:booking-service:9191
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - LAPTOP-4QBKNEGT:payment-service:8080

General Info

Name	Value
------	-------

Eureka Service registry project structure

The screenshot shows the IntelliJ IDEA IDE with the project structure on the left and the code for the ServiceRegistryApplication in the main editor.

Project Structure:

- service_registry (src) main java com upgrad service registry ServiceRegistryApplication
 - resources
 - application.properties
 - application.yml

Code for ServiceRegistryApplication.java:

```
package com.upgrad.service.registry;

import ...

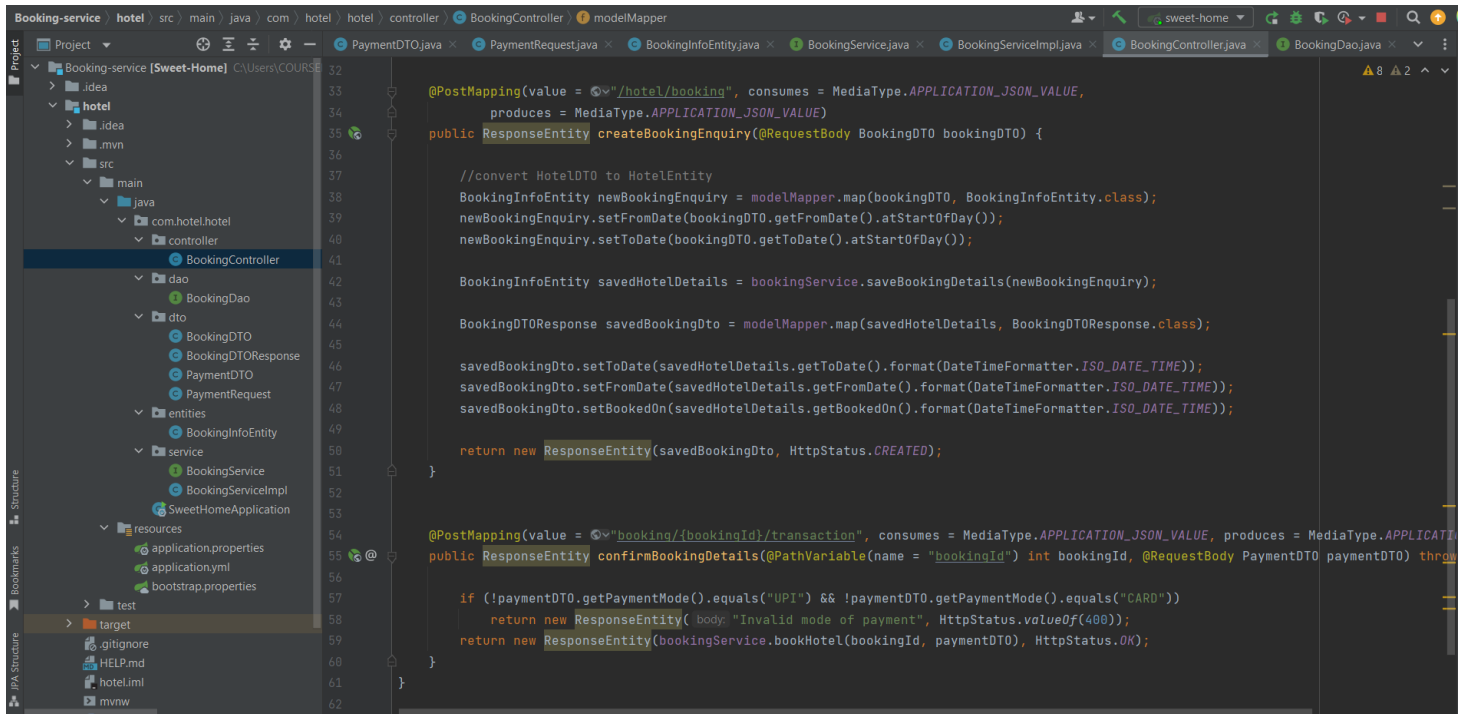
@SpringBootApplication
@EnableEurekaServer
public class ServiceRegistryApplication {

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

}
```

Booking-Service

Project structure



BookingController -

POST <http://localhost:9191/hotel/booking> url will be called to save data in the database using the method

```
public ResponseEntity createBookingEnquiry(@RequestBody BookingDTO bookingDTO)
which has its implementation in the service Class BookingServiceImpl (interface used BookingService ) in the below method.
```

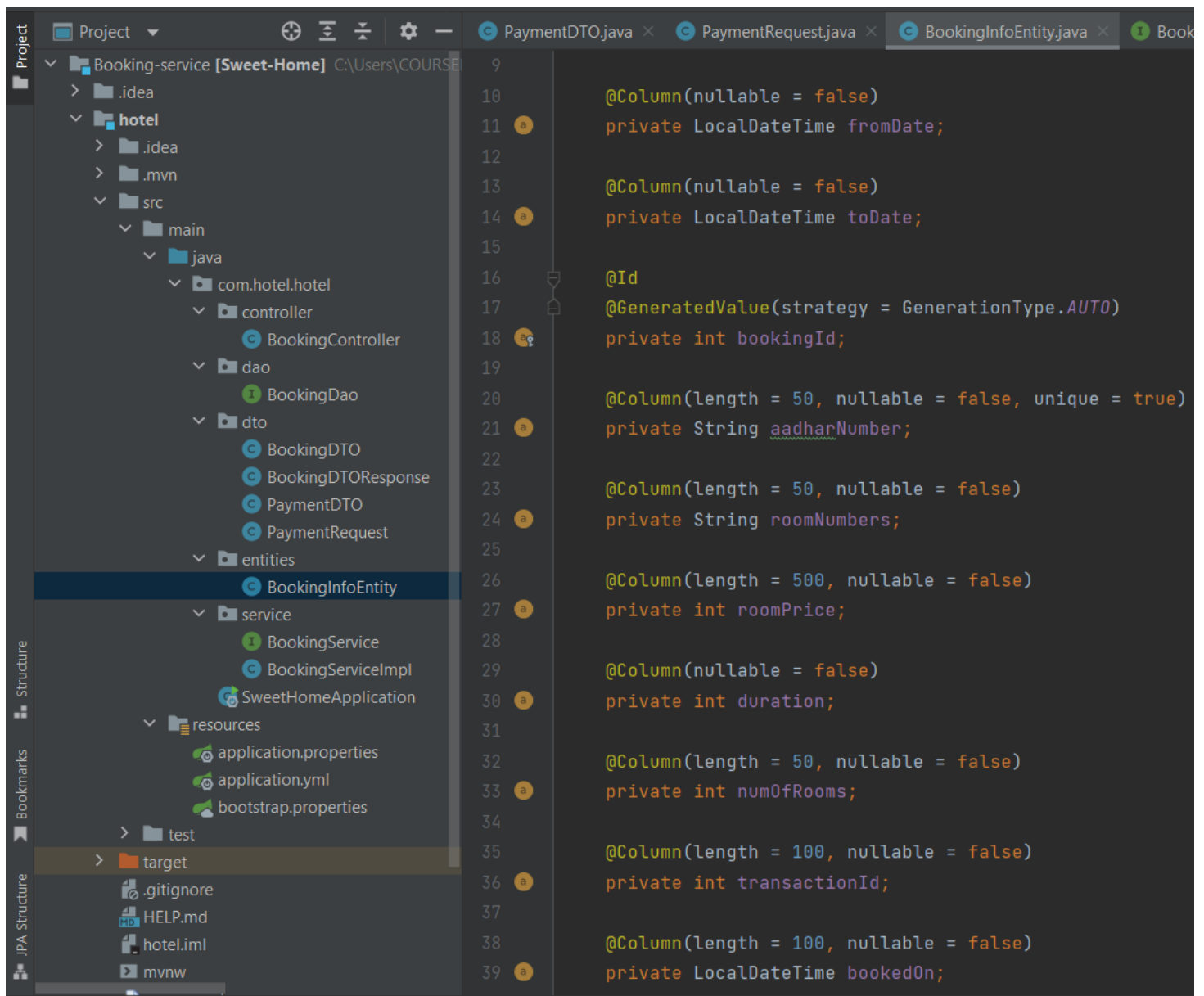
```
public BookingInfoEntity saveBookingDetails(BookingInfoEntity bookingInfo);
```

POST <http://localhost:9191/booking/{bookingId}/transaction> API call is made to confirm the booking using the method

```
public ResponseEntity confirmBookingDetails(@PathVariable(name = "bookingId") int
bookingId, @RequestBody PaymentDTO paymentDTO) throws Exception
which has its implementation in the service Class BookingServiceImpl (interface used BookingService ) in the below method.
```

```
public BookingInfoEntity bookHotel(int bookingId, PaymentDTO paymentDTO) throws Exception;
```

Entity class with the field mapping



API calls

1. When the user enquires about the booking in the hotel , call done using postman , below is the request and response JSON

<http://localhost:9191/hotel/booking>

Postman interface showing a POST request to `http://localhost:9191/hotel/booking`. The request body is a JSON object with the following fields:

```
{  "fromDate": "2021-06-10",  "toDate": "2021-06-15",  "aadharNumber": "1234-4567-0000123",  "numOfRooms": 5}
```

The response body is a JSON object with the following fields:

```
{  "fromDate": "2021-06-10T00:00:00",  "toDate": "2021-06-15T00:00:00",  "bookingId": 1,  "aadharNumber": "1234-4567-0000123",  "roomPrice": 25000,  "duration": 0,  "numOfRooms": 5,  "transactionId": 0,  "bookedOn": "2022-07-10T16:33:38.2145545",  "roomNumbers": "[77, 98, 12, 86, 6]"}
```

Data saved in the database

Database management tool interface showing the schema of the `booking_info_entity` table and the data saved in it.

Table: booking_info_entity

Columns:

- `booking_id` int PK
- `aadhar_number` varchar(50)
- `booked_on` int
- `duration` int
- `from_date` datetime
- `num_of_rooms` int
- `room_numbers` varchar(50)
- `room_price` int
- `to_date` datetime
- `transaction_id` int

Result Grid:

booking_id	aadhar_number	booked_on	duration	from_date	num_of_rooms	room_numbers	room_price	to_date	transaction_id
1	1234-4567-0000123	2022-07-1...	0	2021-06-10 00:00:00	5	[77, 98, 12, 86, 6]	25000	2021-06-15 00:00:00	0

- Request with the booking id and fetch details from the payment service and save it in the booking table and display it in the response.

<http://localhost:9191/booking/1/transaction>

Postman interface showing a POST request to `http://localhost:9191/booking/1/transaction`. The request body is a JSON object:

```
{  "paymentMode": "CARD",  "bookingId": 1,  "upiId": "12324-34323-00091",  "cardNumber": ""}
```

The response body is a JSON object:

```
{  "fromDate": "2021-06-10T00:00:00",  "toDate": "2021-06-15T00:00:00",  "bookingId": 1,  "aadharNumber": "1234-4567-0000123",  "roomNumbers": "[77, 98, 12, 86, 6]",  "roomPrice": 25000,  "duration": 0,  "numOfRooms": 5,  "transactionId": 2,  "bookedOn": "2022-07-10T16:33:38"}
```

Record updated the booking table

DBeaver interface showing the `booking_info_entity` table in the `upgradpg` database. The table contains one record:

booking_id	aadhar_number	booked_on	duration	from_date	num_of_rooms	room_numbers	room_price	to_date	transaction_id
1	1234-4567-0000123	2022-07-10	0	2021-06-10 00:00:00	5	[77, 98, 12, 86, 6]	25000	2021-06-15 00:00:00	2

Record updated in the payment table

The screenshot shows a database management interface. On the left, a 'SCHEMAS' pane lists various tables under the 'upgradpg' schema, including 'transaction_details_entity'. The main area displays a SQL query: `SELECT * FROM upgradpg.transaction_details_entity;`. Below the query, a 'Result Grid' shows a single record with the following values:

transaction_id	booking_id	card_number	from_date	payment_mode	upi_id
1	1	12324-34323-00091	NULL	CARD	NULL

Below the result grid, the 'Columns' for 'transaction_details_entity' are listed:

- transaction_id: int PK
- booking_id: int
- card_number: varchar(50)
- from_date: date
- payment_mode: varchar(50)
- upi_id: varchar(50)

At the bottom, an 'Output' pane shows a message: 'There are pending changes. Please'.

If the payment mode is not UPI or CARD , it throws bad request 400

The screenshot shows a REST client interface. The selected request is a POST to `http://localhost:9191/hotel/booking/1/transaction`. The request body is a JSON object:

```
{  "paymentMode": "CARDId",  "bookingId": 1,  "upiId": "12324-34323-00091",  "cardNumber": ""}
```

The response status is '400 Bad Request' with a message: 'Invalid mode of payment'.

If the booking id is not present in the database , it throws bad request 400

<http://localhost:9191/booking/3/transaction>

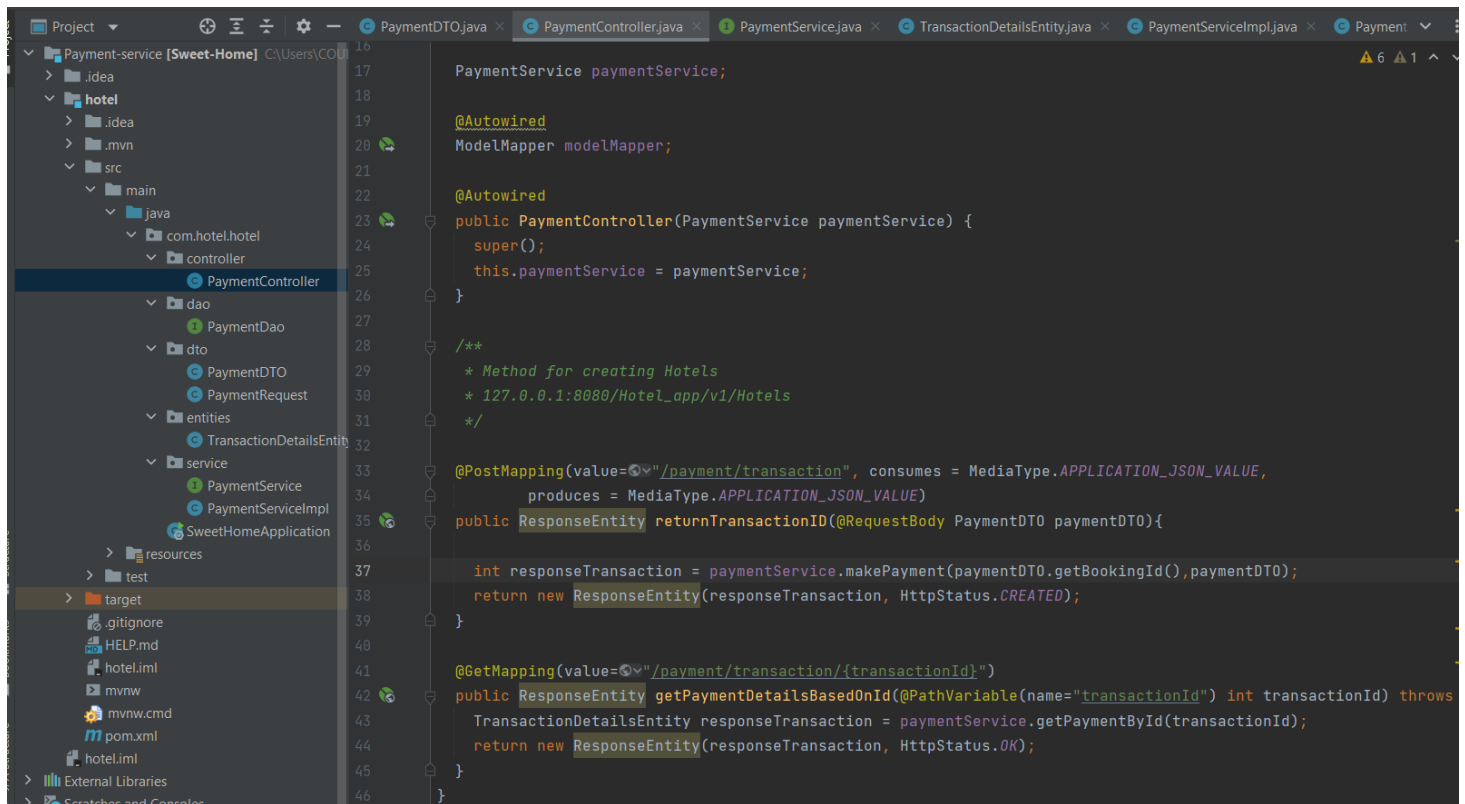
The screenshot displays the Postman API client interface. On the left sidebar, the 'My Workspace' section is active, showing a collection of requests. The main panel shows a POST request to `http://localhost:9191/booking/3/transaction`. The request body is a JSON object with the following fields:

```
1 {
2   "paymentMode": "CARD",
3   "bookingId": 3,
4   "upiId": "12324-34323-00091",
5   "cardNumber": ""
6 }
```

The response status is `400 Bad Request` with a time of 23 ms and a size of 154 B. The response body is `Invalid Booking Id`.

Payment-Service

Project structure



BookingController -

POST <http://localhost:8080/payment/transaction> url will be called to save data in the database using the method

which has its implementation in the service Class **PaymentServiceImpl** (interface used **PaymentService**) in the below method.

```
public BookingInfoEntity saveBookingDetails(BookingInfoEntity bookingInfo);
```

GET <http://localhost:8080/payment/transaction/{{transactionId}}> API call is made to fetch the payment details based on the transaction id using the method

```
public ResponseEntity getPaymentDetailsBasedOnId(@PathVariable(name="transactionId") int transactionId) throws Exception
```

which has its implementation in the service Class **PaymentServiceImpl** (interface used **PaymentService**) in the below method.

```
public TransactionDetailsEntity getPaymentById(int transactionId) throws Exception
```


Entity class with the field mapping

The screenshot shows an IDE with the following components:

- Project View (Left):** Displays the project structure. The path `Payment-service > hotel > src > main > java > com > hotel > hotel > entities` is expanded, and `TransactionDetailsEntity` is selected.
- Code Editor (Right):** Shows the `TransactionDetailsEntity` class with the following code:

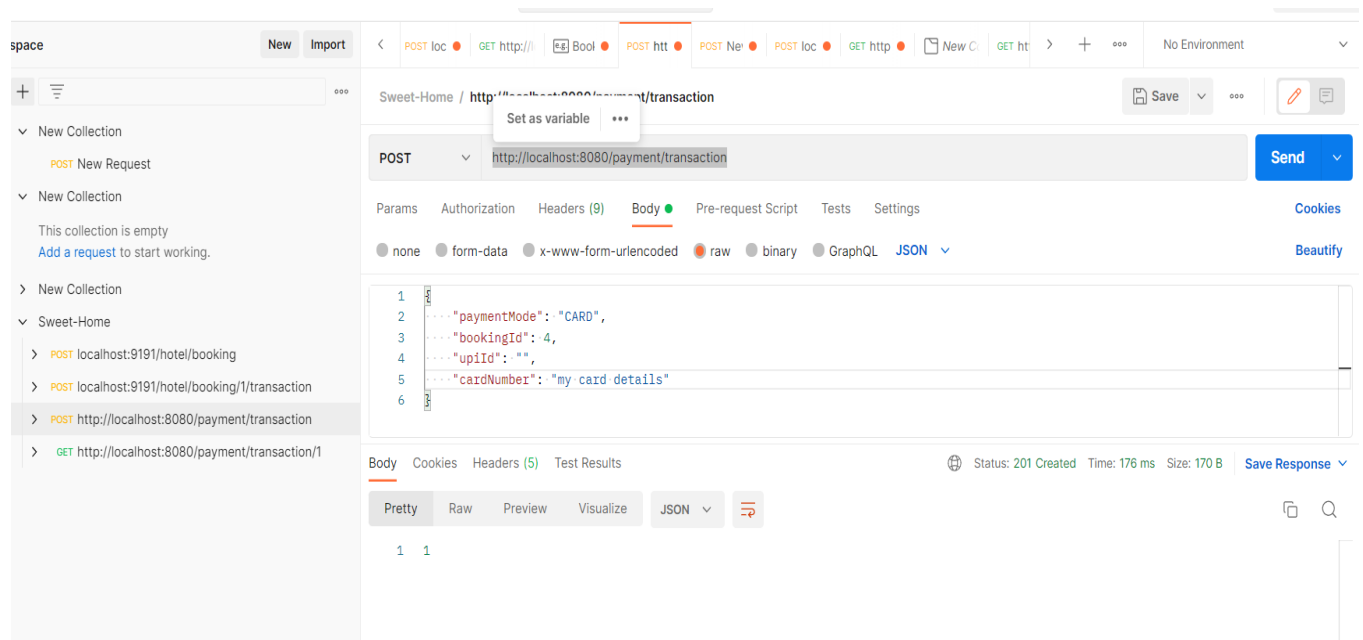
```
1 package com.hotel.hotel.entities;  
2  
3 import ...  
4  
5  
6  
7  
8  
9 @Entity  
10 public class TransactionDetailsEntity {  
11  
12     @Id  
13     @GeneratedValue(strategy = GenerationType.AUTO)  
14     private int transactionId;  
15  
16     @Column(length=50, nullable = false)  
17     private String paymentMode;  
18  
19     @Column(length = 50, nullable = false)  
20     private int bookingId;  
21  
22     @Column(length = 50)  
23     private String upiId;  
24  
25     @Column(length = 50)  
26     private String cardNumber;  
27  
28     private LocalDate fromDate;  
29 }
```

API calls

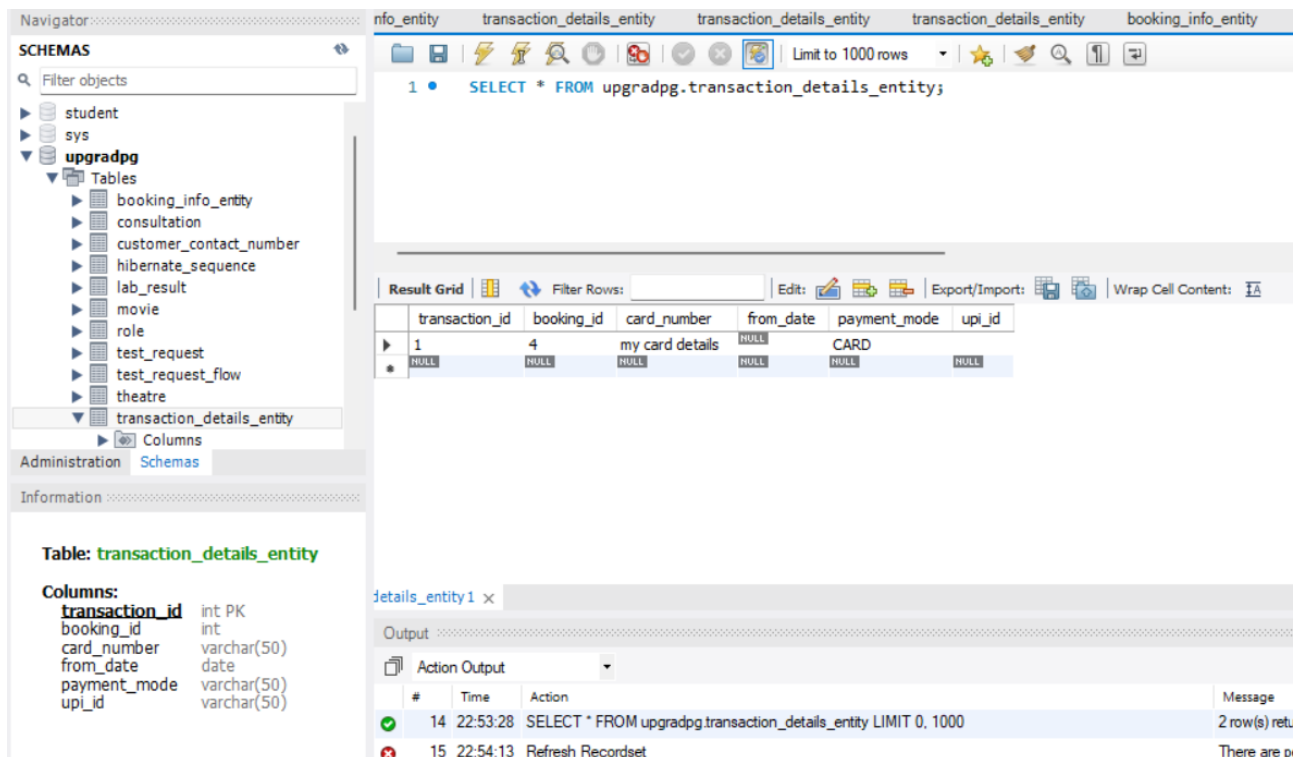
(please use port 8080 to make the API calls for this service)

1. Using this API call the payment details is saved the database and the transaction id is returned in the response, call done using postman, below is the request and response JSON in the screenshot.

<http://localhost:8080/payment/transaction>



Data saved in the database



2. Request with the transaction id and get the payment details from the database and display it in the response.

<http://localhost:8080/payment/transaction/1>

The screenshot shows a REST client interface with a sidebar on the left containing a collection named 'Sweet-Home'. The main panel displays a GET request to the URL `http://localhost:8080/payment/transaction/1`. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response status is 200 OK, with a time of 36 ms and a size of 276 B.

```
1
2
3
4
5
6
7
8
{
  "transactionId": 1,
  "paymentMode": "CARD",
  "bookingId": 4,
  "upiId": "",
  "cardNumber": "my card details",
  "fromDate": null
}
```

(Note : database details used is as provided in the assignment details)