Microservices Identification with HLD --------------------------------------------------------------------

API Endpoints for each Microservice ----------------------------------------------------------------

Success & Failure Scenarios --------------------------------------------------------------------------

Discovery Server Implementation --------------------------------------------------------------------

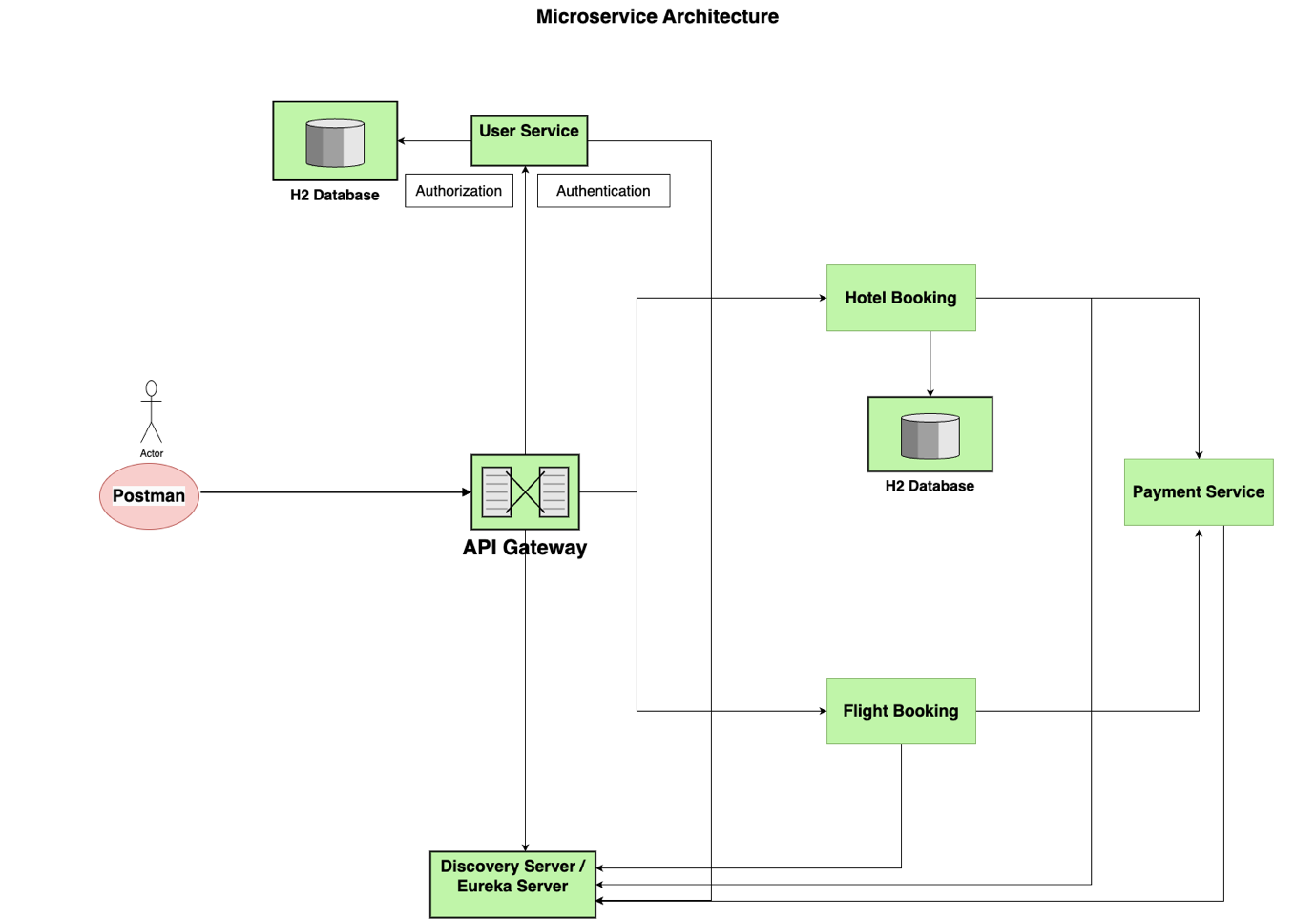
Interservice Communication Approach --------------------------------------------------------------

Cross Cutting Concerns --------------------------------------------------------------------------------

Deployment ----------------------------------------------------------------------------------------------

Technologies & Tools used ----------------------------------------------------------------------------

**Travel Portal MSA**

****

**High Level Diagram**

Services Identified:

1. User: User service is responsible for the following functionalities:

* Registering a user.
* Logging a user.
* Authentication and authorization of users.
* Logger is implemented so log all the details.
* Validates all the request parameters and then proceed with the request else throw exception.

2. Discovery Server: Discovery Server is responsible for the following functionalities:

* Registers all the services.
* Displays all the available services.
* Allows services to find and communicate with each other without hard coding the hostname and port.

3. API Gateway: API gateway is responsible for the following functionalities:

* All services are not directly exposed to end-user, so we get all the requests in gateway only.
* Only allows two requests to hit directly without token i.e., signup & login.
* Validates all the requests and then route to services.
* Provides ease of access for users that they don’t need to hit multiple ports for multiple services.

4. Hotel Booking: Hotel Booking is responsible for the following functionalities:

* Responsible for searching hotels according to availability.
* Maintains inventories for all the hotels.
* User can book flight for multiple rooms in a single request only.
* This service hits payment service internally and make payment or issue refund in case of cancellation.
* Circuit breaker and logging is enabled.
* Hotel Details and customer details both are logged when the hotel is confirmed.
* Validates all the request parameters and then proceed with the request else throw exception.

5. Flight Booking: Flight Booking is responsible for the following functionalities:

* Responsible for searching flights according to availability.
* Maintains inventories for all the flights.
* User can book flight for multiple people in a single request only.
* User can also book round trip for multiple people in a request.
* This service hits payment service internally and make payment or issue refund.
* Circuit breaker and logging is enabled.
* Flight Details and customer details both are logged when the flight is confirmed.
* Validates all the request parameters and then proceed with the request else throw exception.

6. Payment Service: Payment service is responsible for the following functionalities:

* Responsible for accepting payments.
* Responsible for issuing refund according to transaction Id.
* Logger is implemented so log all the details.
* Validates all the request parameters and then proceed with the request else throw exception.

Assumptions:

* User can hit only through API gateway cannot hit microservices directly so cannot do anything without logging in.

Detailed API Explanation:

User Service

* Signup

This API is used for registering users.

Method Type: - POST

Request URL: - /user/signup

Request Body: -

{

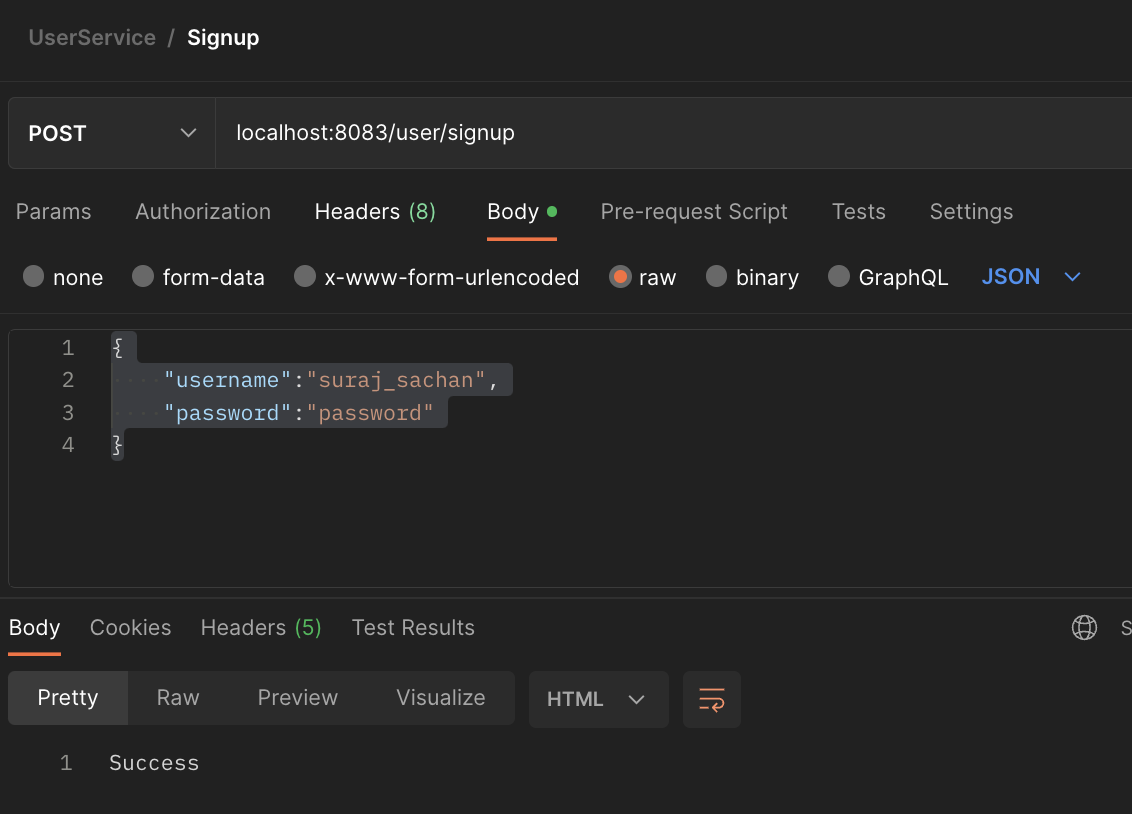
"username":"user-name",

"password":"password"

}

Response: -

Success



* Login API

This API is used for login users and generating token for further requests, and we can pass that token in other requests.

Method Type: - POST

Request URL: - /user/login

Request Body: -

{

"username":"user-name",

"password":"password"

}

Response: -

{

"token": "eyJhbGciOiJIUzUxMiJ9.eyJyb2xlIjpbXSwic3ViIjoic3VyYWpfc2FjaGFuIiwiaWF0IjoxNjc5NTk4NTgzLCJleHAiOjE2Nzk2MjczODN9.xfaq3uvhQ5GEkyAg9ClWFrGjRE5BxmHa5-pMl26GZU8Z6U\_JNuIFrvahxurLOA2hCgfoK\_FQT63pO33rBdbANg"

}



* Verify API

This API is used for verifying and identifying users using bearer token that we get from login api.

Method Type: - POST

Request URL: - /user/verify

Request Body: -

{

"username":"user-name",

"password":"password"

}

Response: -

{{

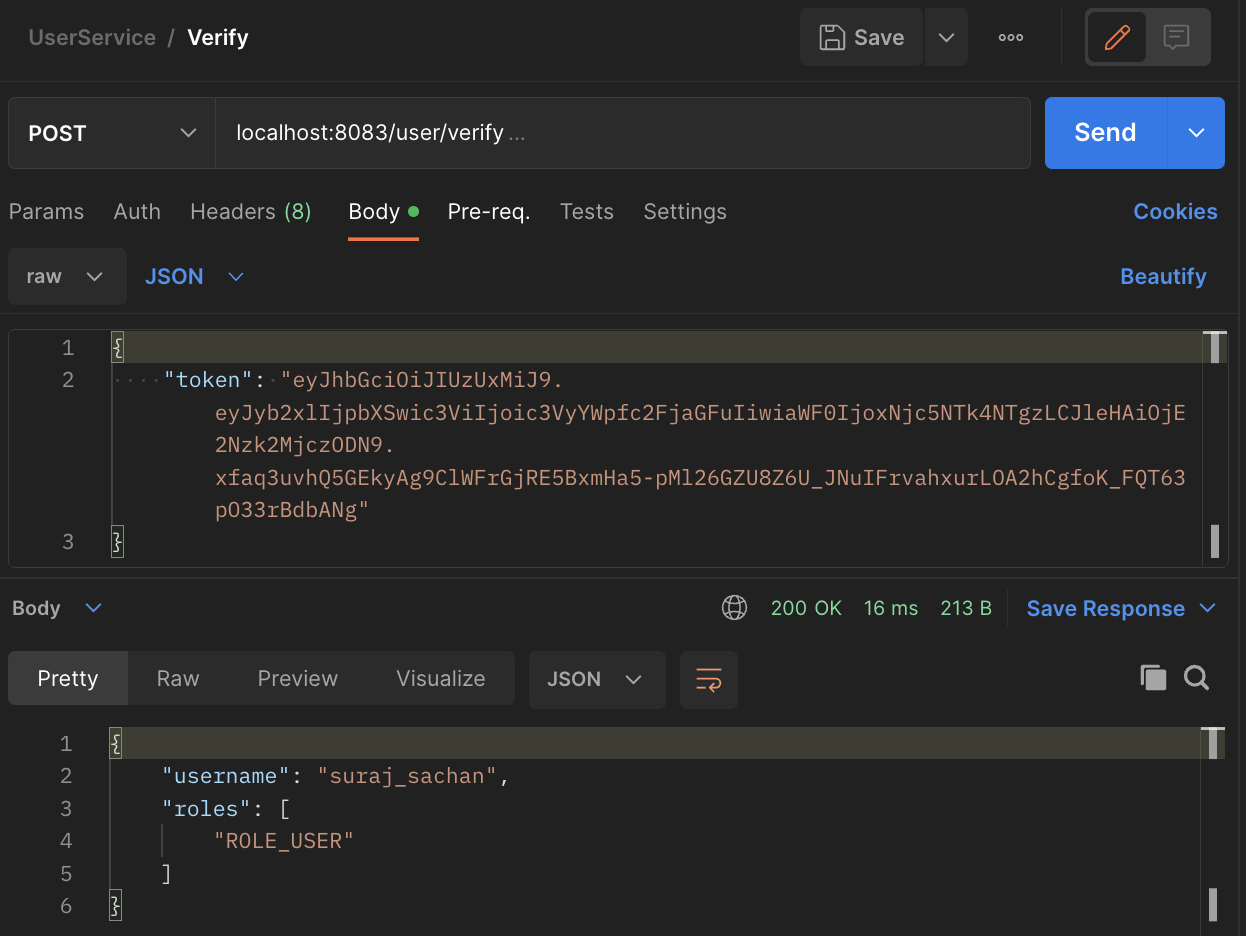
"username": "suraj\_sachan",

"roles": [

"ROLE\_USER"

]

}



Hotel Booking

* Search all available hotels

This API is used for searching all the available hotels in inventory and return all the available hotel list with location.

Method Type: - GET

Request URL: - /search/

Response: -

[

{

"hotelName": "mahagun",

"hotelId": "992c51bb-830f-4f49-bef0-61bfed0ae2fd",

"locationDTO": [

{

"locationId": "87856eb1-1ccd-4345-913b-091008934c47",

"city": "NEW DELHI",

"capacity": 80,

"price": 5000

},

{

"locationId": "a6da16fa-f034-402b-b049-5e53b76ade39",

"city": "BANGALORE",

"capacity": 100,

"price": 5000

},

{

"locationId": "e9db7f2a-f788-4cb7-9530-eaace683bf6a",

"city": "MUMBAI",

"capacity": 60,

"price": 5000

},

{

"locationId": "ecb7e466-5e22-40e0-980b-1a3ddd57661f",

"city": "GOA",

"capacity": 25,

"price": 4000

}

]

},

{

"hotelName": "le meridien",

"hotelId": "6d21a7ef-20e4-4020-9488-72c094ed245d",

"locationDTO": [

{

"locationId": "5eb9b906-9979-4bb3-9d9d-afa2ce789e83",

"city": "NEW DELHI",

"capacity": 80,

"price": 7500

},

{

"locationId": "73de0bf4-8d43-43e4-ab1c-91a6ff64c6fe",

"city": "MUMBAI",

"capacity": 60,

"price": 8500

}

]

},

{

"hotelName": "hotel royal park",

"hotelId": "7777a7f9-3623-4f99-86a7-c764dc5af68d",

"locationDTO": [

{

"locationId": "0e60f2e2-5586-4faf-a4c0-54a00abb13c8",

"city": "NEW DELHI",

"capacity": 40,

"price": 10000

},

{

"locationId": "a4ee1c15-d228-4fa9-808e-5eb44f260eba",

"city": "GOA",

"capacity": 35,

"price": 6000

},

{

"locationId": "acfddcee-d9c9-42b2-931f-8ac776df509f",

"city": "MUMBAI",

"capacity": 40,

"price": 10000

},

{

"locationId": "dbc367cd-a5f5-4886-a139-fd9c41a88af5",

"city": "BANGALORE",

"capacity": 50,

"price": 9000

}

]

},

{

"hotelName": "country inn",

"hotelId": "40bd62ca-9552-494d-8727-9d1cd49a25be",

"locationDTO": [

{

"locationId": "3eaeac1f-3e0e-43d1-bfb2-ea43af4adf5d",

"city": "BANGALORE",

"capacity": 50,

"price": 9000

},

{

"locationId": "89c2c16f-a269-463b-b981-b715b92b7e1d",

"city": "GOA",

"capacity": 35,

"price": 6000

},

{

"locationId": "c4d7b257-aaf4-43c5-9e87-c1fbb19c51fc",

"city": "NEW DELHI",

"capacity": 40,

"price": 10000

}

]

},

{

"hotelName": "hotel the grand palace",

"hotelId": "f964f703-5a7c-4602-8036-8a632b544c50",

"locationDTO": [

{

"locationId": "7ec3f21c-475a-4795-b190-ca3de61c3336",

"city": "NEW DELHI",

"capacity": 40,

"price": 9000

}

]

}

]

Text

Description automatically generated

* Get hotel by ID

This API is used for searching hotel with hotelId and it will return all the available locations for that hotelId.

Method Type: - GET

Request URL: - /search/id/{hotelID}

Response: -

{

"hotelName": "mahagun",

"hotelId": "992c51bb-830f-4f49-bef0-61bfed0ae2fd",

"locationDTO": [

{

"locationId": "87856eb1-1ccd-4345-913b-091008934c47",

"city": "NEW DELHI",

"capacity": 80,

"price": 5000

},

{

"locationId": "a6da16fa-f034-402b-b049-5e53b76ade39",

"city": "BANGALORE",

"capacity": 100,

"price": 5000

},

{

"locationId": "e9db7f2a-f788-4cb7-9530-eaace683bf6a",

"city": "MUMBAI",

"capacity": 60,

"price": 5000

},

{

"locationId": "ecb7e466-5e22-40e0-980b-1a3ddd57661f",

"city": "GOA",

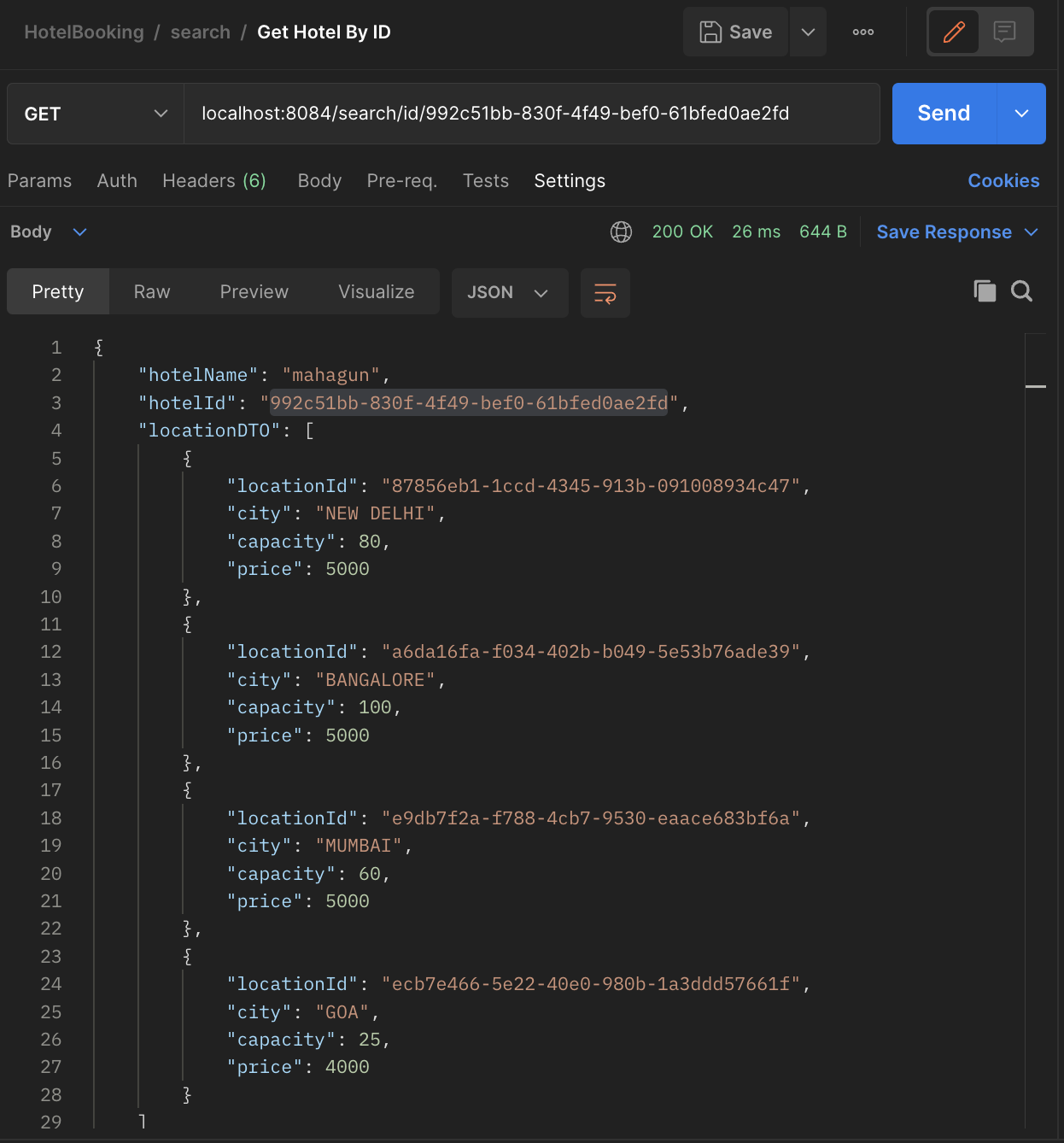
"capacity": 25,

"price": 4000

}

]

}



* Get hotel by Name

This API is used for searching hotel with name and it will return all the available locations for that hotel name.

Method Type: - GET

Request URL: - /search/id/{ name }

Response: -

{

"hotelName": "mahagun",

"hotelId": "992c51bb-830f-4f49-bef0-61bfed0ae2fd",

"locationDTO": [

{

"locationId": "87856eb1-1ccd-4345-913b-091008934c47",

"city": "NEW DELHI",

"capacity": 80,

"price": 5000

},

{

"locationId": "a6da16fa-f034-402b-b049-5e53b76ade39",

"city": "BANGALORE",

"capacity": 100,

"price": 5000

},

{

"locationId": "e9db7f2a-f788-4cb7-9530-eaace683bf6a",

"city": "MUMBAI",

"capacity": 60,

"price": 5000

},

{

"locationId": "ecb7e466-5e22-40e0-980b-1a3ddd57661f",

"city": "GOA",

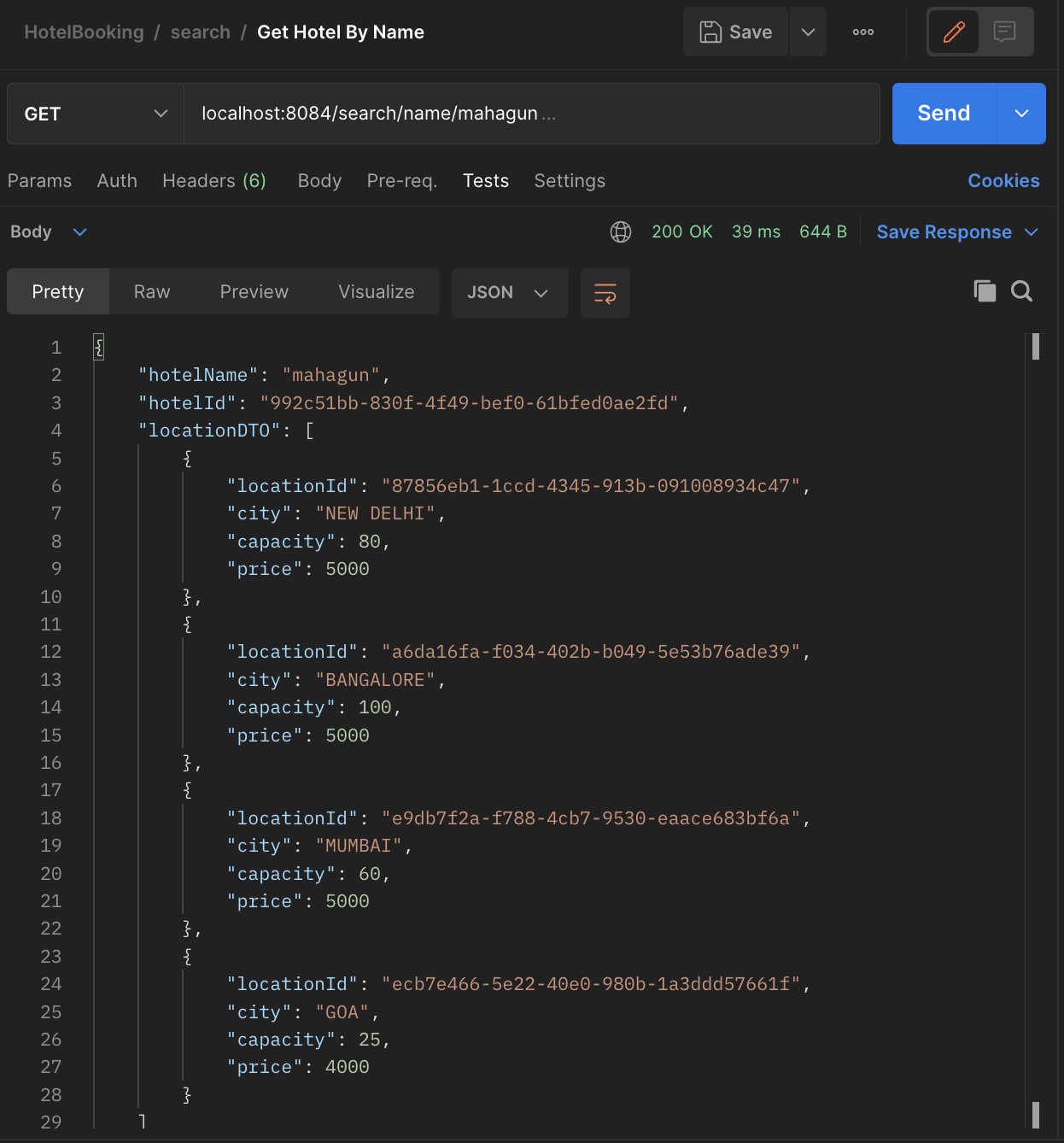
"capacity": 25,

"price": 4000

}

]

}



* Get hotel by City

This API is used for searching hotels and it will return all the hotels available in that city.

Method Type: - GET

Request URL: - /search/id/{ name }

Response: -

[

{

"hotelName": "mahagun",

"locationId": "e9db7f2a-f788-4cb7-9530-eaace683bf6a",

"capacity": "60",

"city": "MUMBAI",

"price": "5000",

"hotelId": "992c51bb-830f-4f49-bef0-61bfed0ae2fd"

},

{

"hotelName": "le meridien",

"locationId": "73de0bf4-8d43-43e4-ab1c-91a6ff64c6fe",

"capacity": "60",

"city": "MUMBAI",

"price": "8500",

"hotelId": "6d21a7ef-20e4-4020-9488-72c094ed245d"

},

{

"hotelName": "hotel royal park",

"locationId": "acfddcee-d9c9-42b2-931f-8ac776df509f",

"capacity": "40",

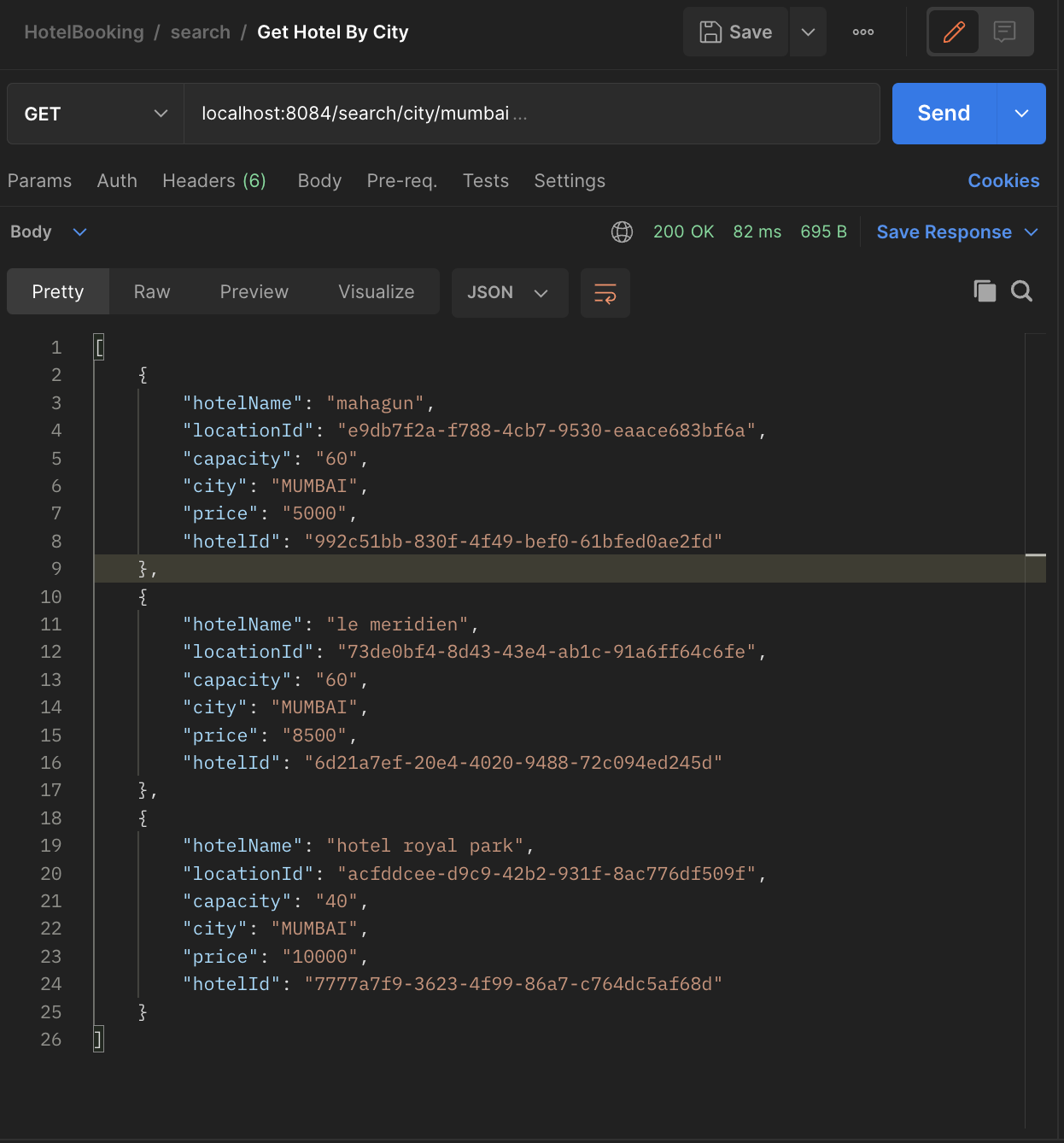
"city": "MUMBAI",

"price": "10000",

"hotelId": "7777a7f9-3623-4f99-86a7-c764dc5af68d"

}

]



* Add hotel booking

This API is used for creating a booking of hotel using location id and then you confirm the hotel by making payment within 15 min else it will get cancelled.

User can book multiple rooms by hitting a single api.

Method Type: - POST

Request URL: - /booking/add

Request Header:-

X-Authenticated-User:{token}

Request Body:-

{

"locationId": "87856eb1-1ccd-4345-913b-091008934c47",

"date": "03/28/2023",

"rooms": 2

}

Response: -

{

"roomIDs": [

3,

4

],

"bookingID": "27e2746e-3a8c-48e0-a214-a027390d2064",

"amount": 10000,

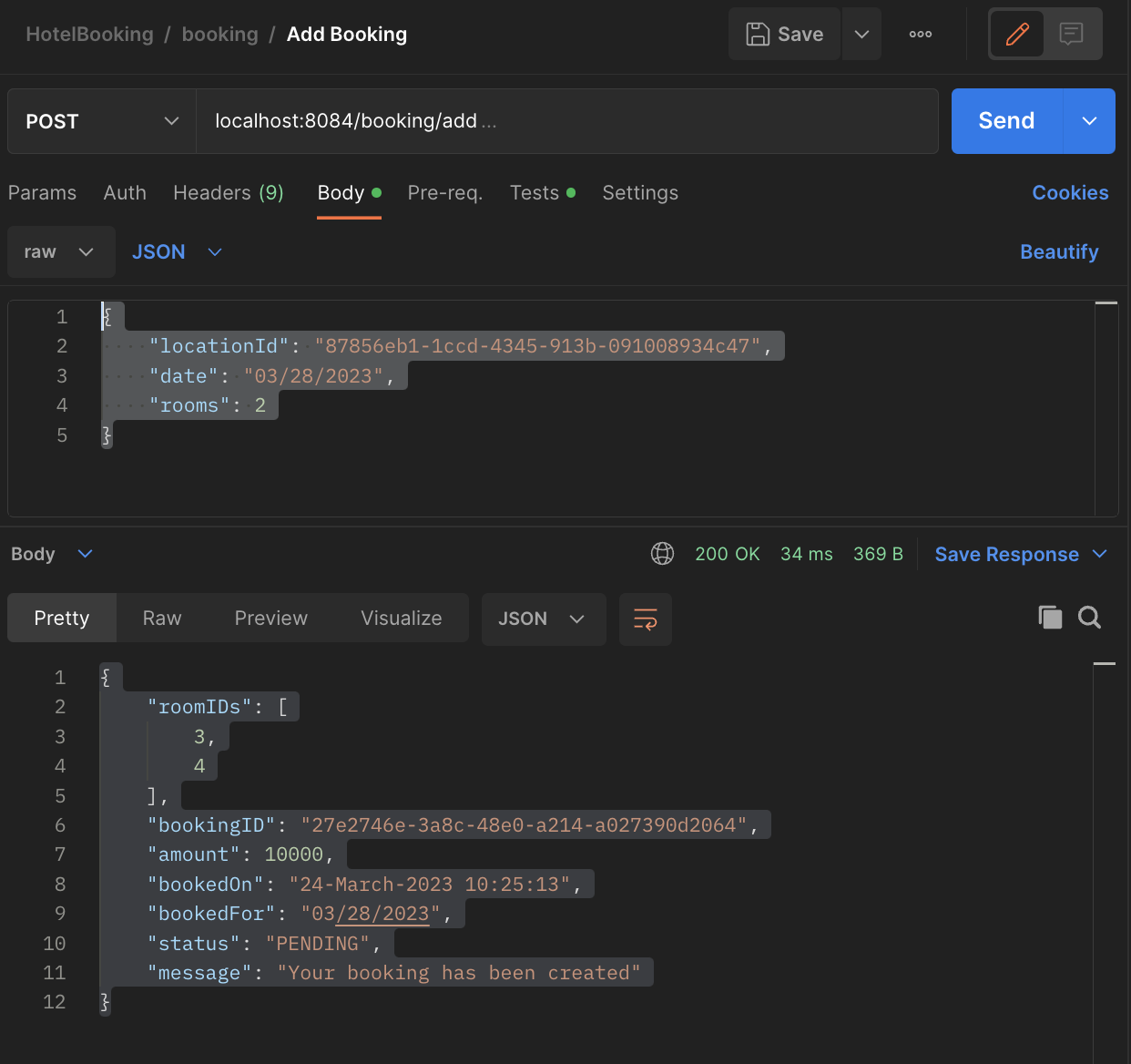
"bookedOn": "24-March-2023 10:25:13",

"bookedFor": "03/28/2023",

"status": "PENDING",

"message": "Your booking has been created"

}



* Confirm hotel booking

This API is used for confirming a booking of hotel using booking id and payment details.

Method Type: - POST

Request URL: - /booking/confirm

Request Header: -

X-Authenticated-User:{token}

Request Body: -

{

"bookingId": "*{{bookingID}}*",

"paymentDetails": {

"cardNumber": "1234",

"cvv": "123"

}

}

Response: -

{

"roomIDs": [

1,

2,

3

],

"bookingID": "0ac68b64-7c22-424d-b9e9-fd2d3ce47f82",

"amount": 15000,

"bookedOn": "24-March-2023 10:52:43",

"bookedFor": "03/30/2023",

"confirmedOn": "24-March-2023 10:54:14",

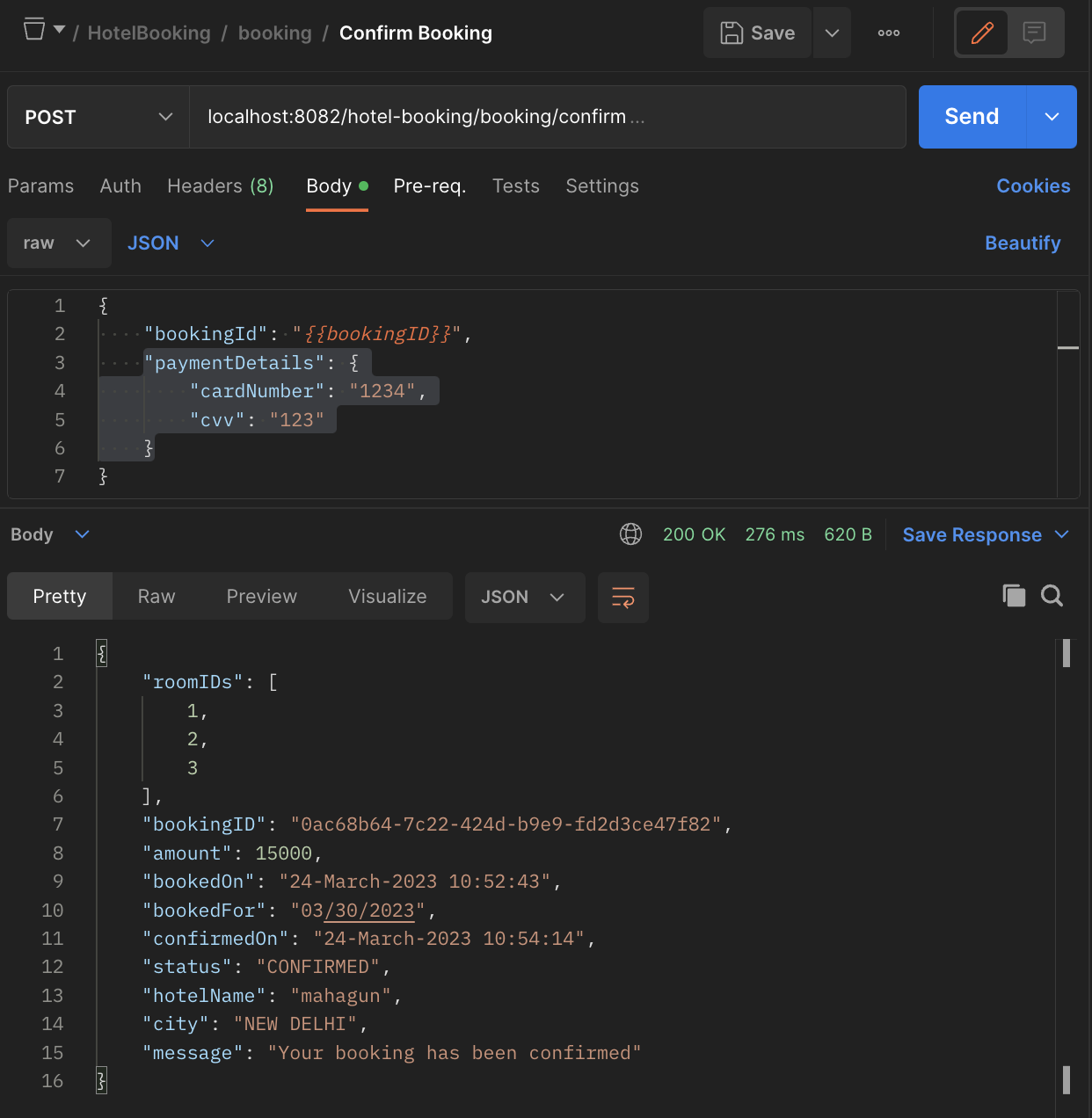
"status": "CONFIRMED",

"hotelName": "mahagun",

"city": "NEW DELHI",

"message": "Your booking has been confirmed"

}



* Cancel hotel booking

This API is used for cancelling a booking of hotel using booking id.

Method Type: - POST

Request URL: - /booking/cancel

Request Header: -

X-Authenticated-User:{token}

Request Body: -

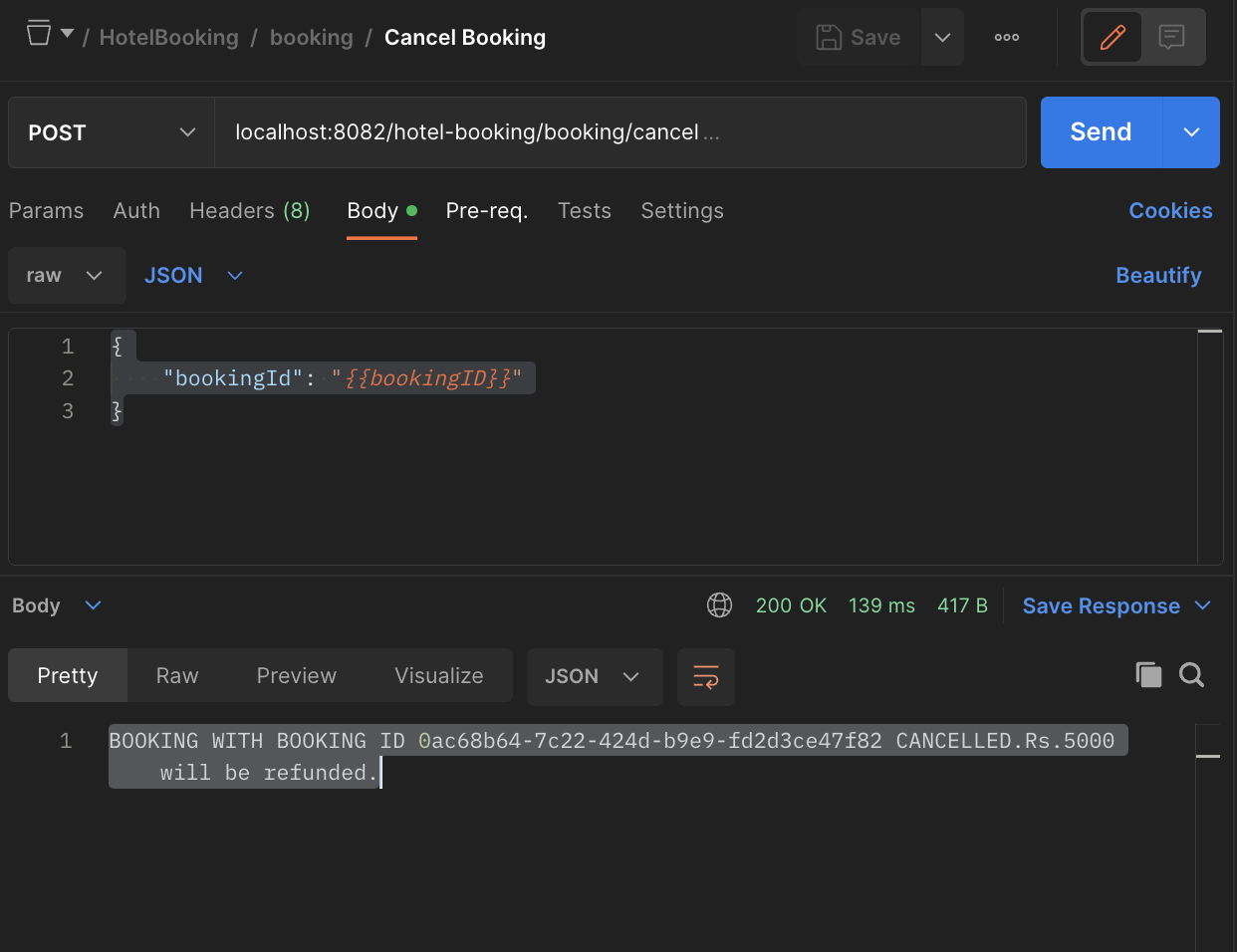
{

"bookingId": "*{{bookingID}}*"

}

Response: -

BOOKING WITH BOOKING ID 0ac68b64-7c22-424d-b9e9-fd2d3ce47f82 CANCELLED.Rs.5000 will be refunded.



* Get hotel availability by city on given date

This API is used for searching hotels available on given date.

Method Type: - POST

Request URL: - /booking/getByCity

Request Body: -

{

"city": "NEW DELHI",

"date": "03/30/2023"

}

Response: -

[

{

"availability": "80",

"hotelName": "mahagun",

"city": "NEW DELHI"

},

{

"availability": "80",

"hotelName": "le meridien",

"city": "NEW DELHI"

},

{

"availability": "40",

"hotelName": "hotel royal park",

"city": "NEW DELHI"

},

{

"availability": "40",

"hotelName": "country inn",

"city": "NEW DELHI"

},

{

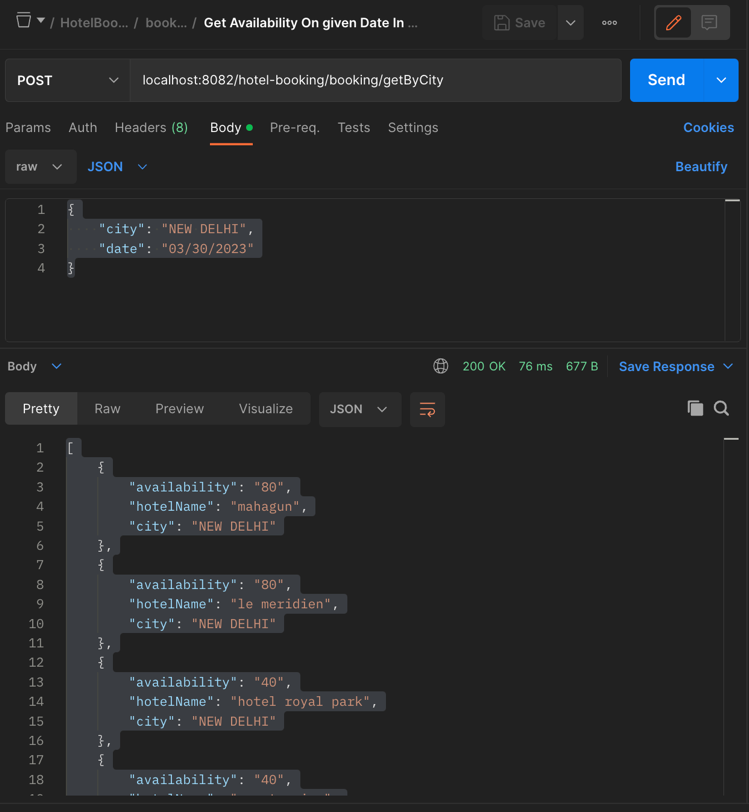
"availability": "40",

"hotelName": "hotel the grand palace",

"city": "NEW DELHI"

}

]



* Get booking details with booking id

This API is used for getting booking details using booking id.

Method Type: - GET

Request URL: - /booking/id/{{bookingID}}

Response: -

{

"bookingId": "0ac68b64-7c22-424d-b9e9-fd2d3ce47f82",

"locationId": "87856eb1-1ccd-4345-913b-091008934c47",

"userID": "suraj",

"roomIds": [

1,

2,

3

],

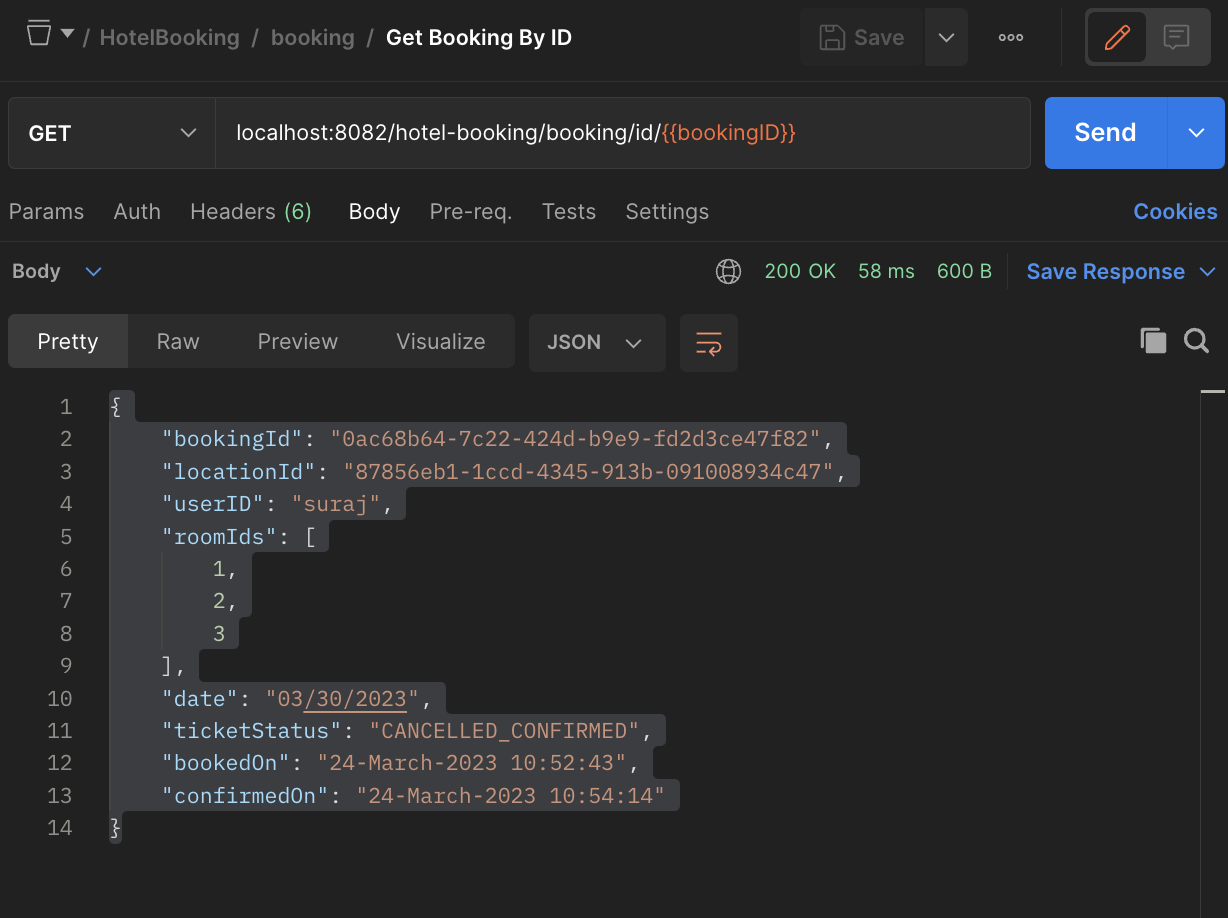
"date": "03/30/2023",

"ticketStatus": "CANCELLED\_CONFIRMED",

"bookedOn": "24-March-2023 10:52:43",

"confirmedOn": "24-March-2023 10:54:14"

}



Flight Booking

* Search all available flights

This API is used for searching all the available flights on given date and return all the available flight list with time.

Method Type: - POST

Request URL: - search/flights

Request Header: -

X-Authenticated-User:{token}

Request Body: -

{

"flightClass": "Both",

"arrivalLocation": "GOA",

"departureLocation": "NEW DELHI",

"departureDate": "2023-03-14",

"returnDate": "2023-03-15",

"searchType": "RoundTrip"

}

Response: -

[

[

{

"flightName": "Vistara",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "02B",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Bussiness",

"amount": 4000.0

},

{

"flightName": "Indigo",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "01B",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Bussiness",

"amount": 2000.0

},

{

"flightName": "Air India",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "03E",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Economy",

"amount": 5500.0

},

{

"flightName": "Vistara",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "02E",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Economy",

"amount": 2500.0

},

{

"flightName": "Indigo",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "01E",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Economy",

"amount": 3000.0

},

{

"flightName": "Air India",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

"flightId": "03B",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Bussiness",

"amount": 5000.0

}

],

[

{

"flightName": "Indigo",

"depLocation": "GOA",

"arrLocation": "NEW DELHI",

"flightId": "10B",

"depTime": "2023-03-15T12:45:30",

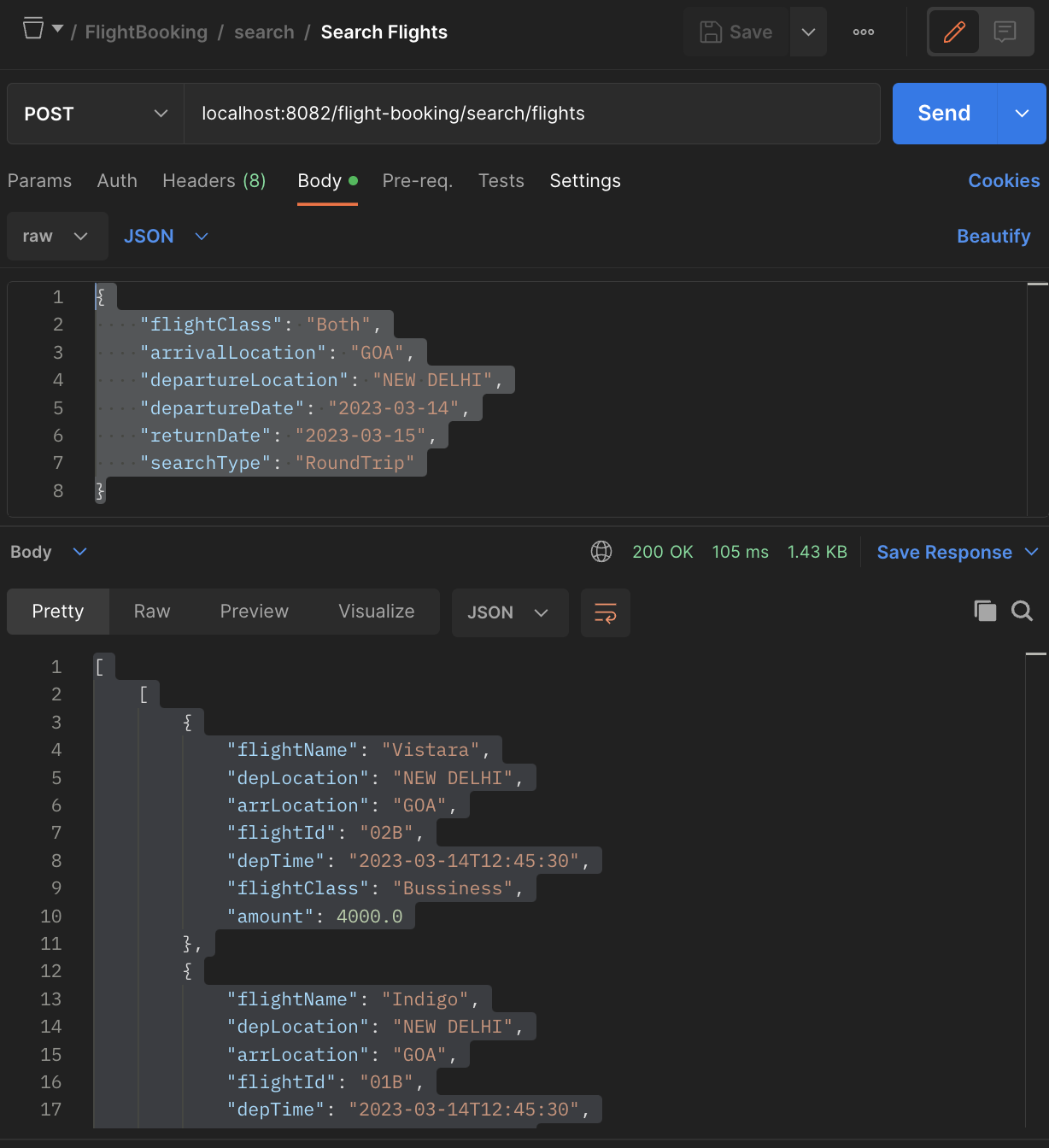
"flightClass": "Bussiness",

"amount": 4500.0

}

]

]



* Check Flight Availability

This API is used for checking availability before blocking seat or processing payment.

Method Type: - POST

Request URL: - search/checkAvailability

Request Body: -

{

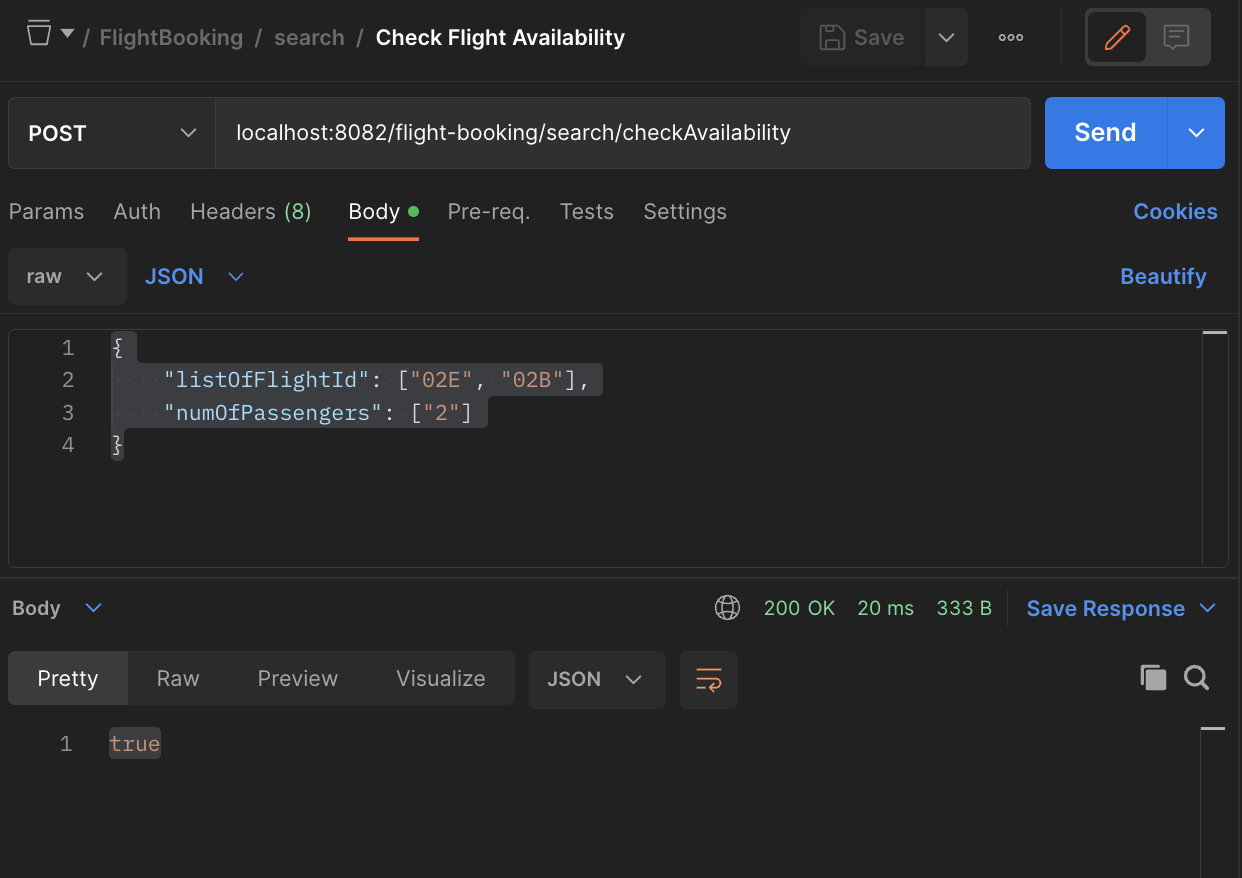
"listOfFlightId": ["02E", "02B"],

"numOfPassengers": ["2"]

}

Response: -

true



* Search flight details with Id

This API is used for searching flight details with flight id.

Method Type: - POST

Request URL: - search/flightInfo?flightId={{flightID}}

Response: -

{

"flightName": "Indigo",

"depLocation": "NEW DELHI",

"arrLocation": "GOA",

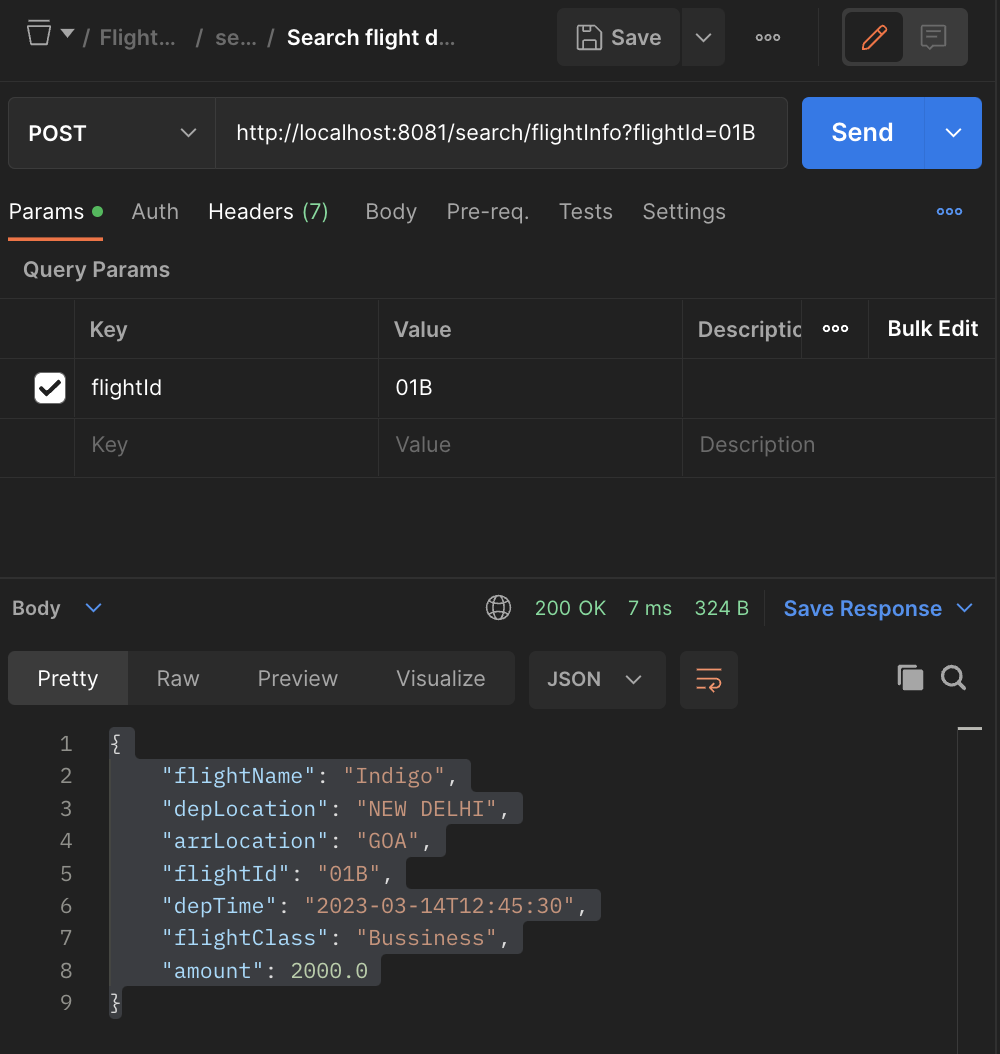
"flightId": "01B",

"depTime": "2023-03-14T12:45:30",

"flightClass": "Bussiness",

"amount": 2000.0

}



* Create flight booking

This API is used for booking flight using flight and you can book for multiple people, one-way and round-trip.

Method Type: - POST

Request URL: - booking/createFlightBooking

Request Body: -

{

"userId": "suraj",

"flightBookingInfo":[{

"customerInfo":[{

"custName": "abc",

"custAddress": "xyz",

"custContactNumber": "9870221585"

}],

"flightId": "01B"

},

{

"customerInfo":[{

"custName": "abc",

"custAddress": "xyz",

"custContactNumber": "9870221585"

}],

"flightId": "02E"

}],

"paymentDetails": {

"cardNumber": "1234567890123456",

"cvv": "123"

}

}

Response: -

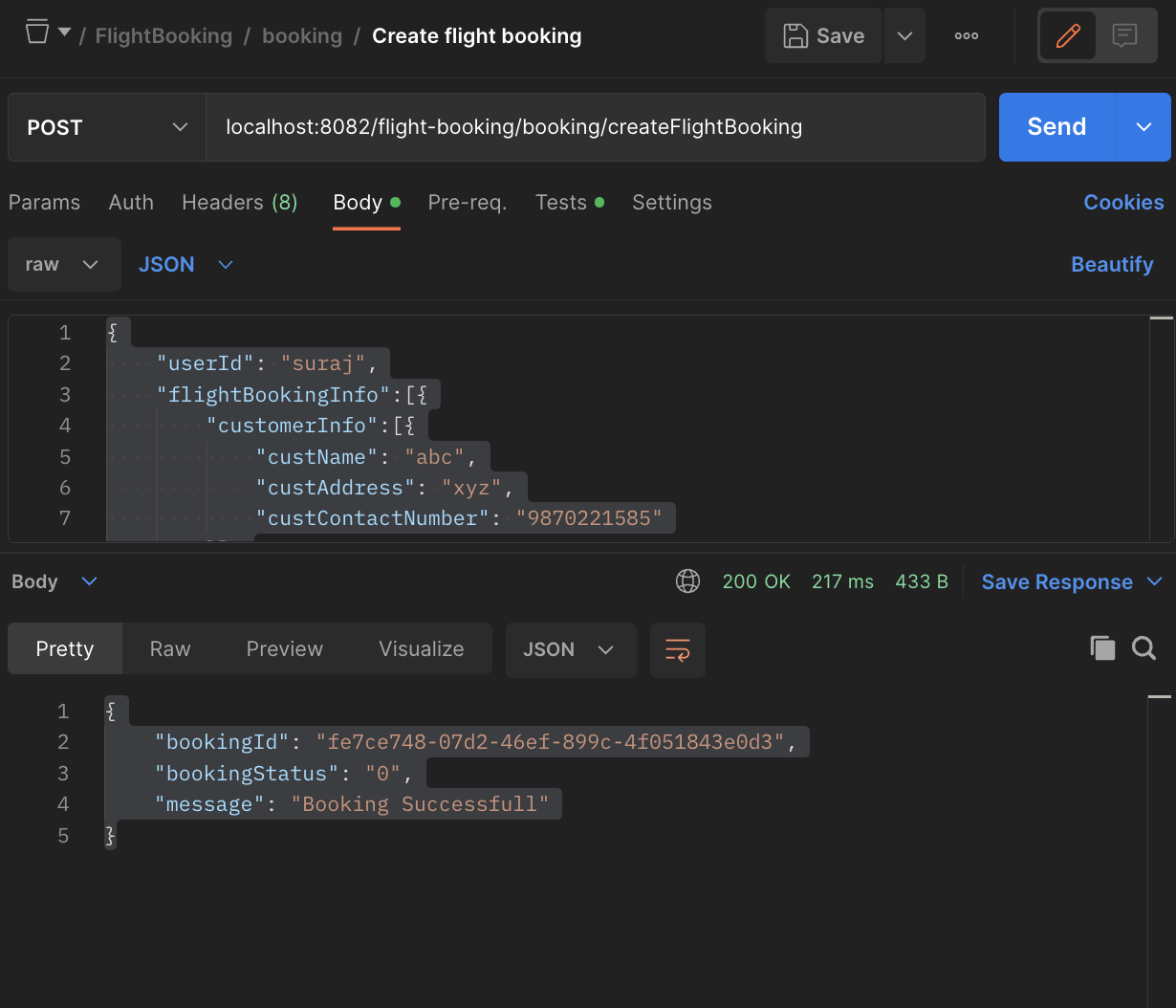
{

"bookingId": "fe7ce748-07d2-46ef-899c-4f051843e0d3",

"bookingStatus": "0",

"message": "Booking Successfull"

}



* Cancel flight booking

This API is used for cancelling flight using booking id and get refund on source account.

Method Type: - POST

Request URL: - booking/cancelFlightBooking?bookingId={{bookingID}}

Response: -

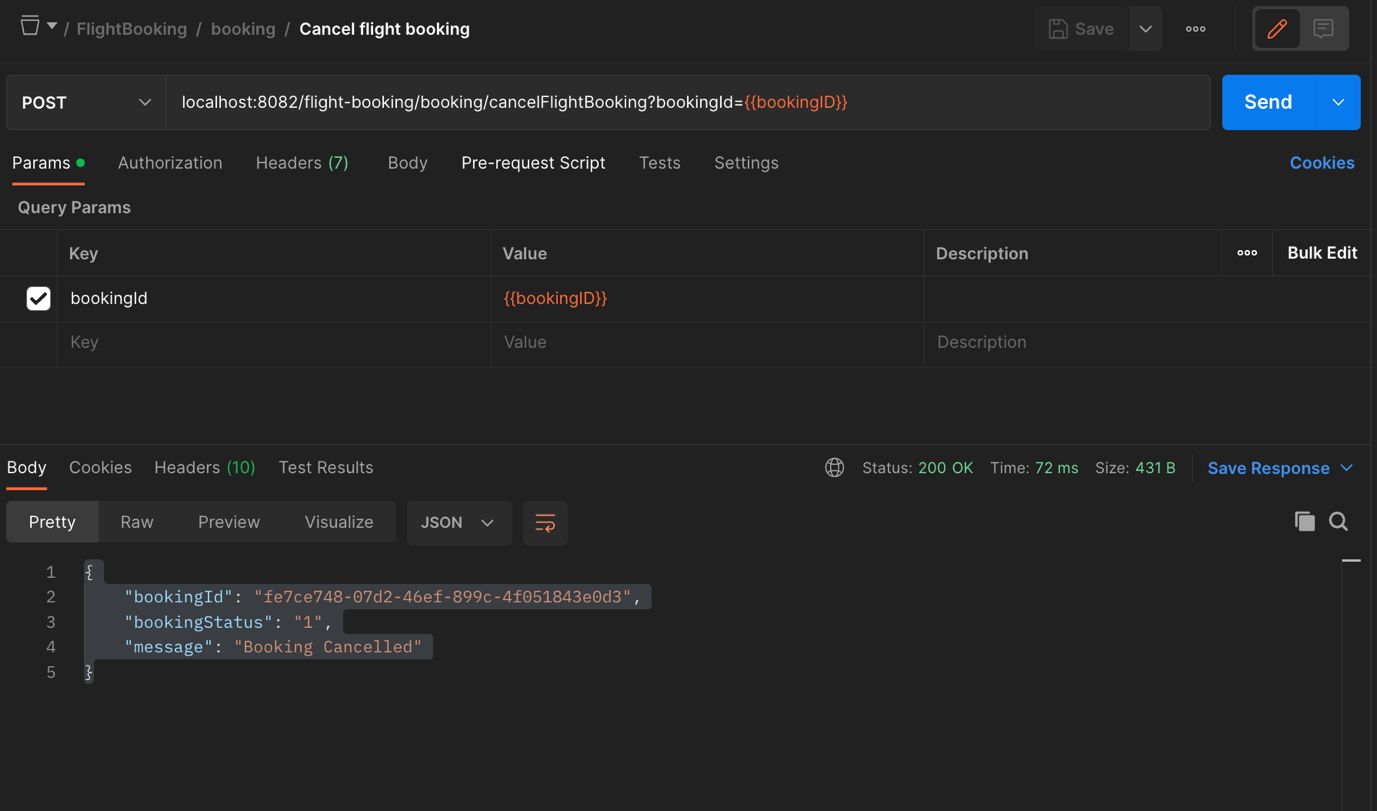
{

"bookingId": "fe7ce748-07d2-46ef-899c-4f051843e0d3",

"bookingStatus": "1",

"message": "Booking Cancelled"

}



Payment Service

This service is requested by flight-booking and hotel-booking service for payment purpose only.

* Make payment API

This API is not directly exposed to user, so this is just for example in this we accept the payment through card details and amount and then return a transaction id.

Method Type: - POST

Request URL: - /makePayment

Request Body: -

{

"cardNumber":"1234567890123456",

"cvv":"123",

"amount":"4500"

}

Response: -

{

"transactionId": "dd9219f4-7de9-4a49-b1e6-82561950e33d",

"message": "Payment processed successfully"

}

A screenshot of a computer

Description automatically generated

* Make refund API

This API is not directly exposed to user, so this is just for example in this we refund the amount with the transaction id.

Method Type: - POST

Request URL: - /refund

Request Body: -

{

"transactionId":"*{{transactionID}}*"

}

Response: -

{

"transactionId": "dd9219f4-7de9-4a49-b1e6-82561950e33d",

"message": "Refund processed successfully"

}

A screenshot of a computer

Description automatically generated

**Success flow and failure flow along with explanation**

Hotel

Success flow  
  
Signup Api > Login Api > Search Hotel > Add Hotel > Confirm Hotel

In this user will first create his account using user service then user will login to his account using his credentials then search for his required hotel and select the hotel then will add hotel using location id and then hotel room will be blocked for next 15 min so user will need to confirm the booking within 15 min using confirm hotel api and in this he need to provide booking id and payment details then this api will hit payment service internally and return the response after performing all the validations.

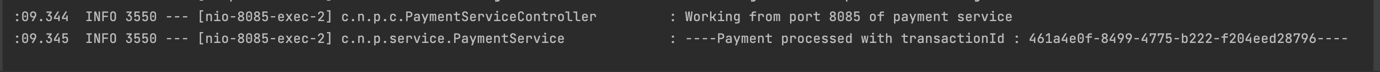
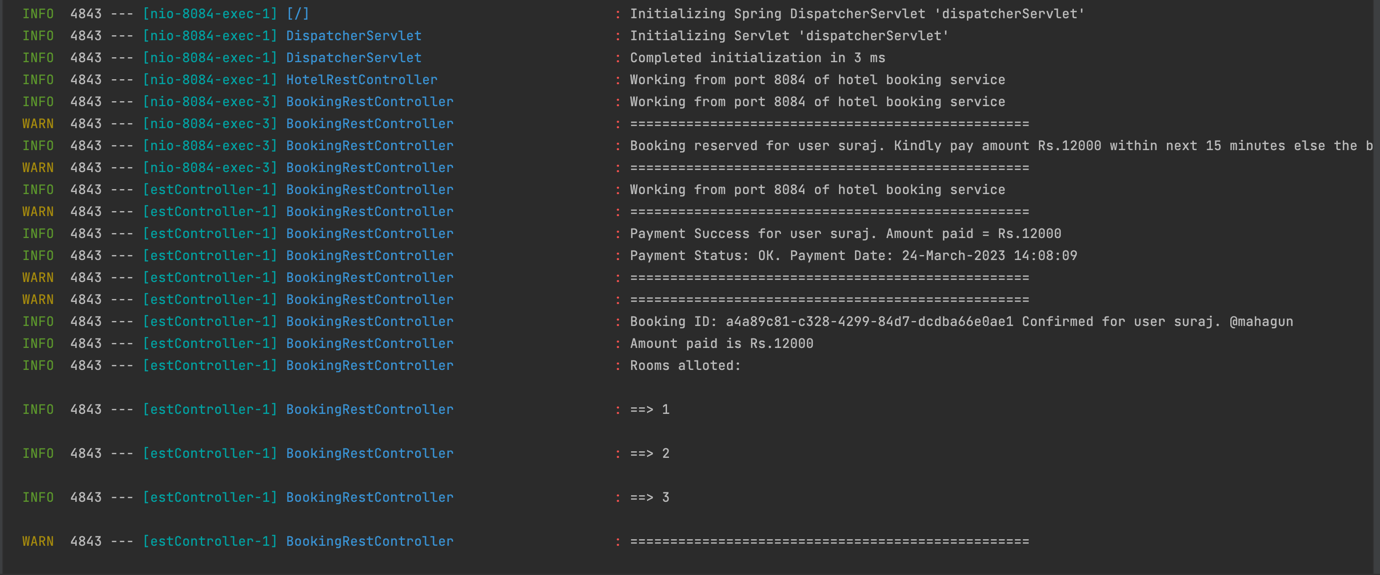
In this I have implemented a scheduler so when a user will hit add hotel api then it will block the tickets for 15 min and then if user will not confirm within 15 min then scheduler will automatically cancel the api.

Text

Description automatically generated

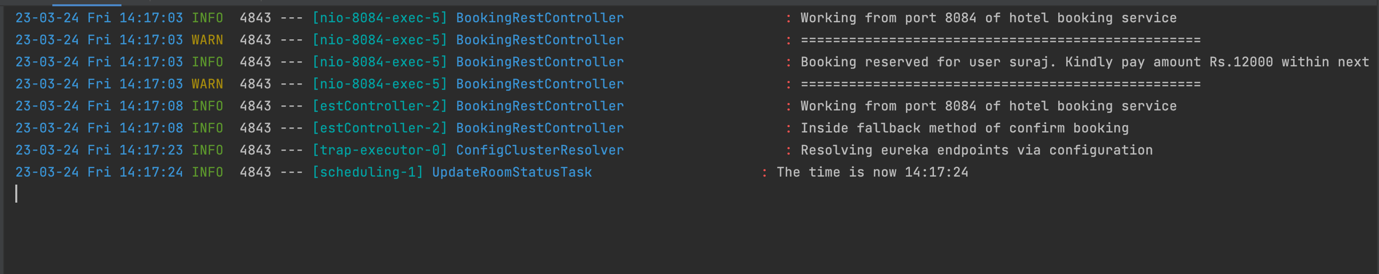
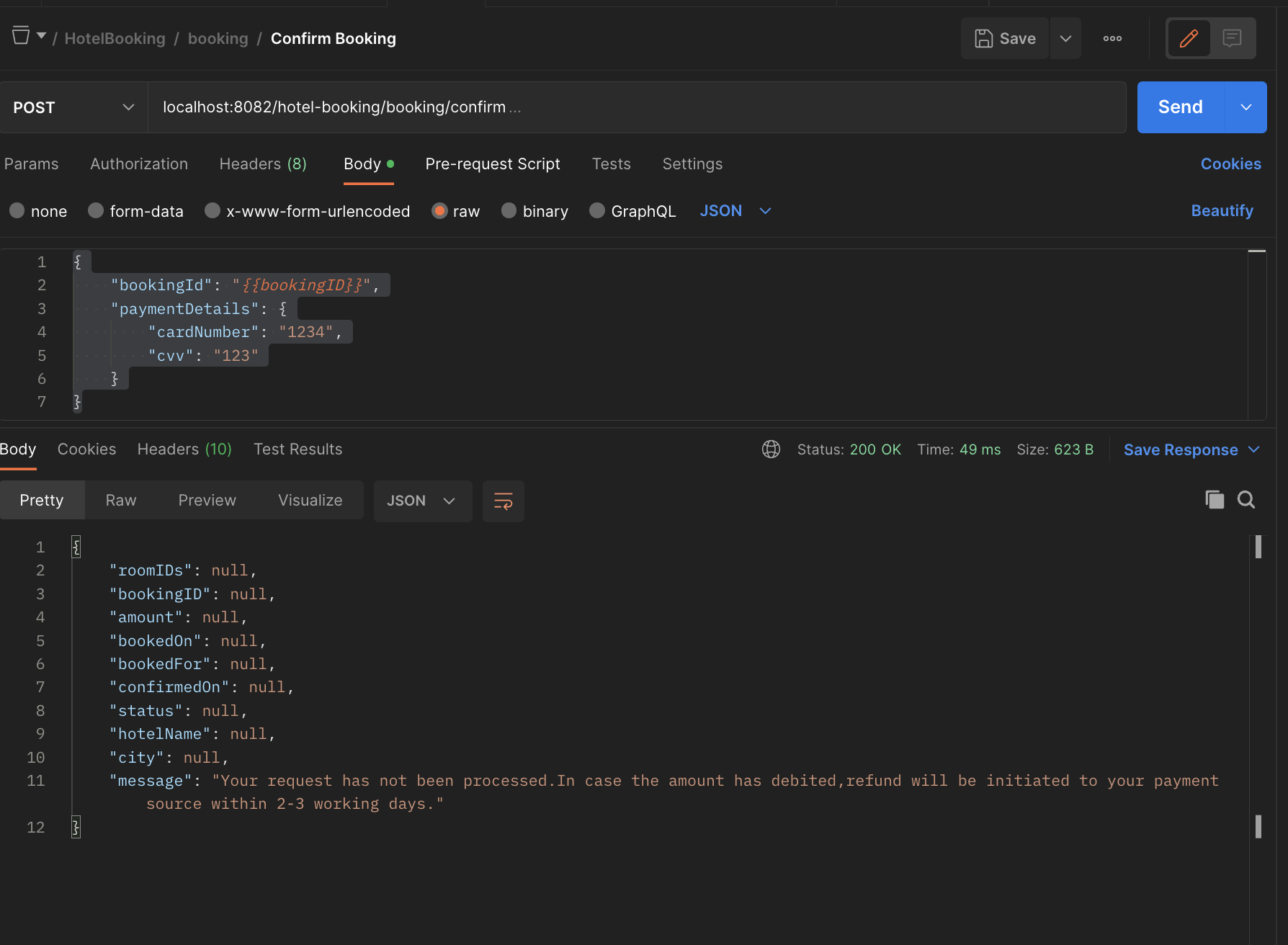
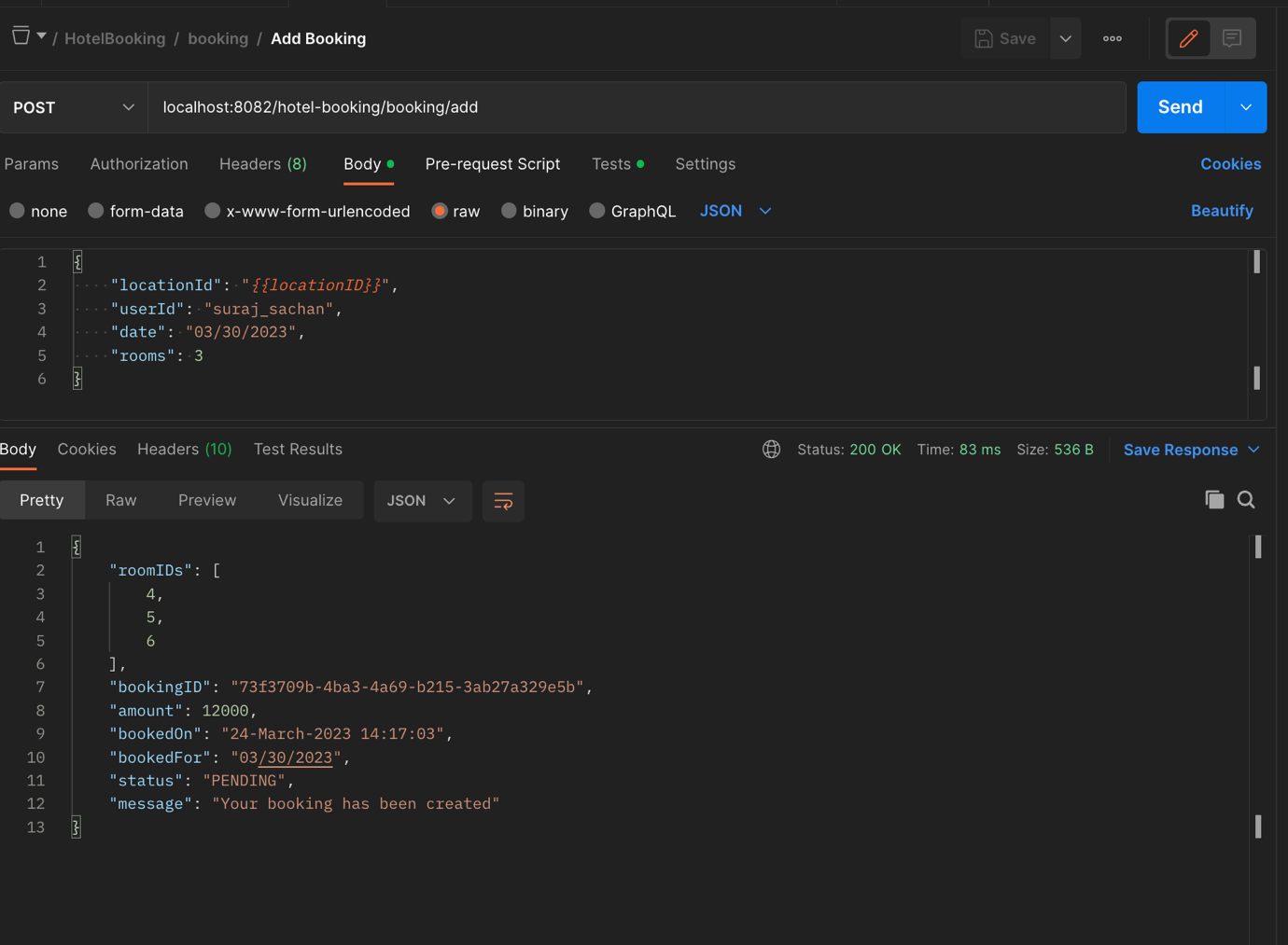
A screenshot of a computer

Description automatically generated

  
  
Failure Booking Flow

Signup Api > Login Api > Search Hotel > Add Hotel > Confirm Hotel

Failure Scenario 1: - Payment Service is down so ticket get cancelled.



Failure scenario 2: - Payment service is up but session timeout happened so refund issued.

In this I will stop the request in between using debugger so that session timeout will happen, and it will do to fallback method and process the refund in case payment is already happened.

Text

Description automatically generated

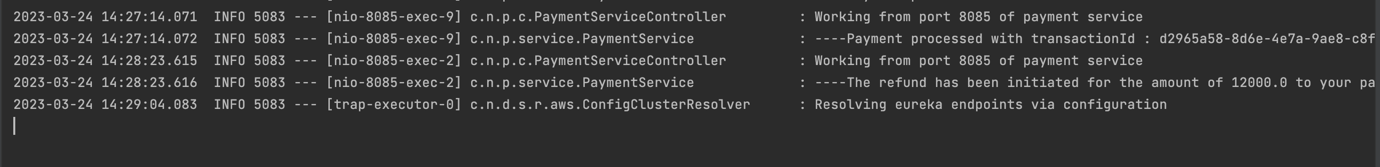
Text

Description automatically generatedText

Description automatically generated

Graphical user interface, text

Description automatically generated

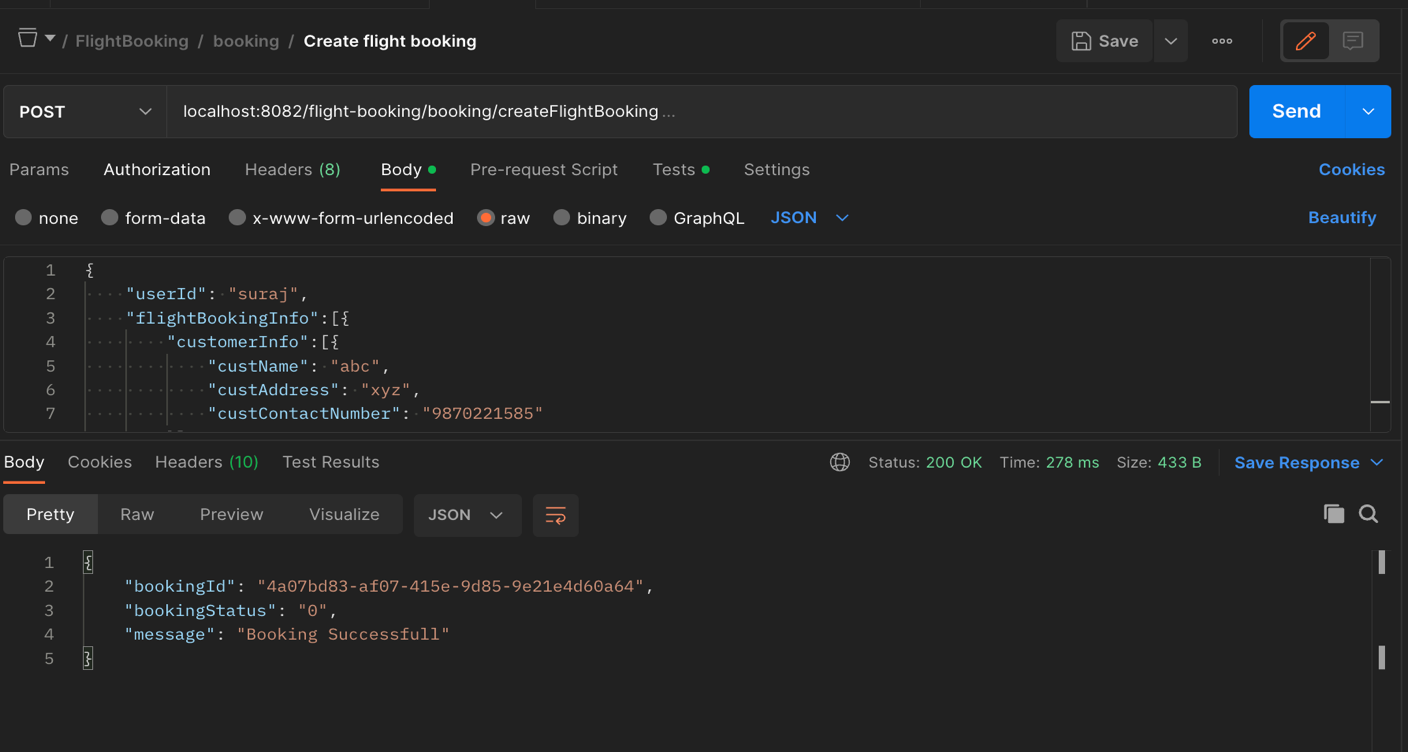


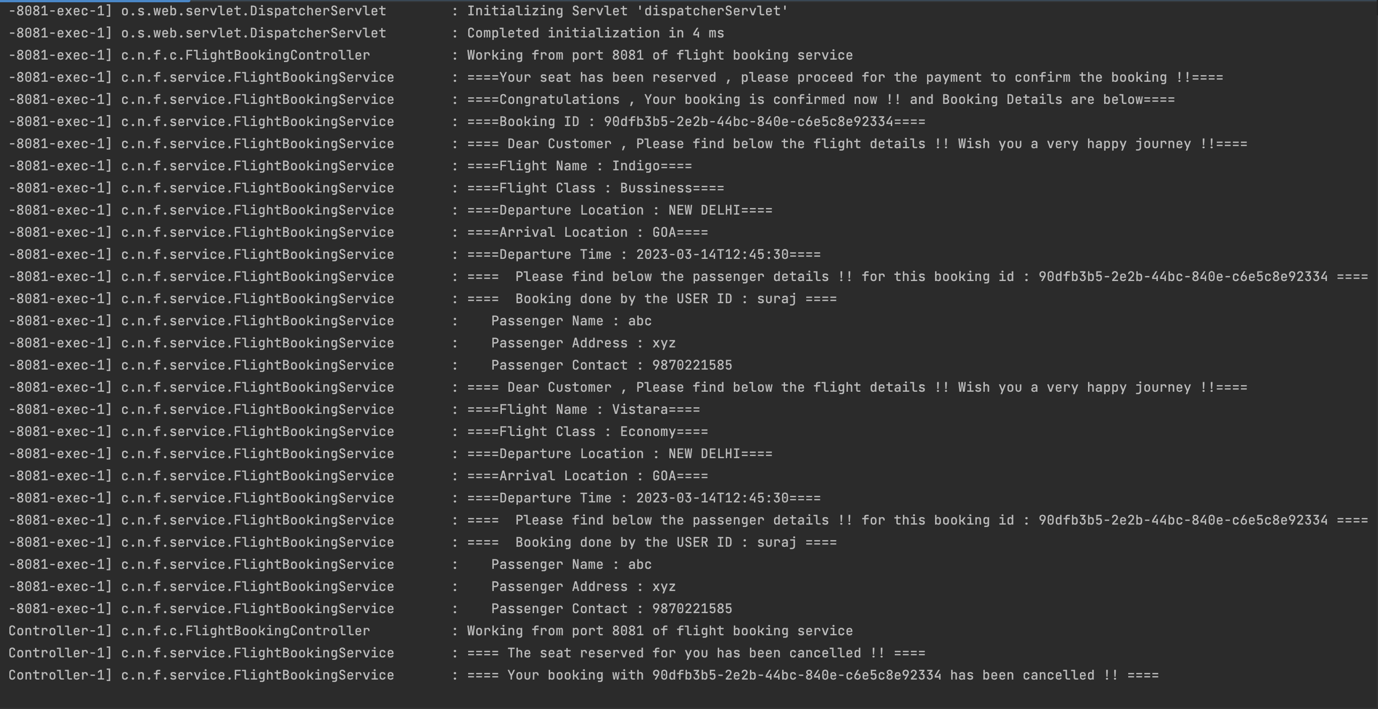
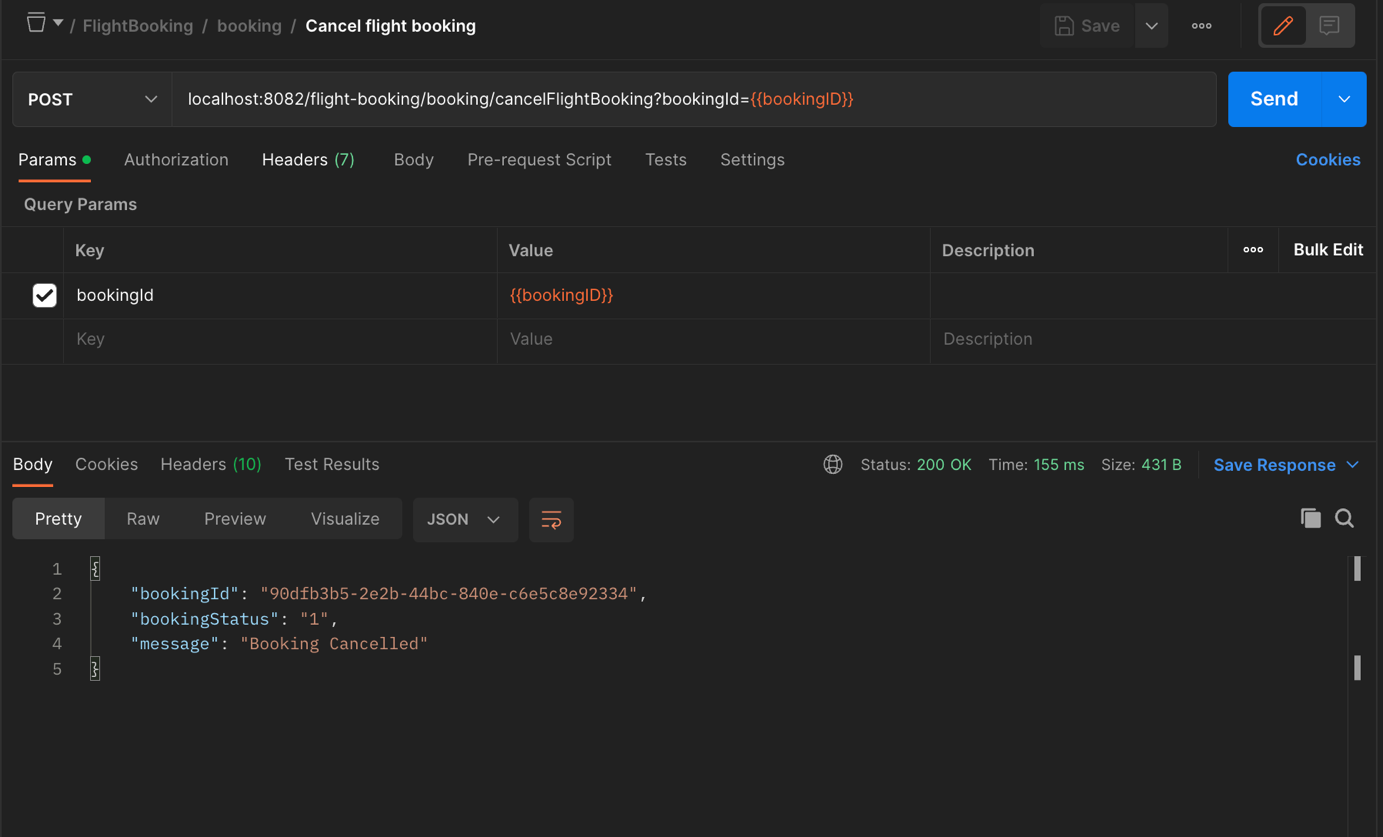
Flight

Success Flow

Signup > Login > Search Flight > Create Flight Booking/Cancel Flight Booking

In this user will first create his account using user service then user will login to his account using his credentials then search for his required flight and select the flight then will hit create flight booking and in this he need to provide flight id and payment details then this api will hit payment service internally and return the response after performing all the validations.

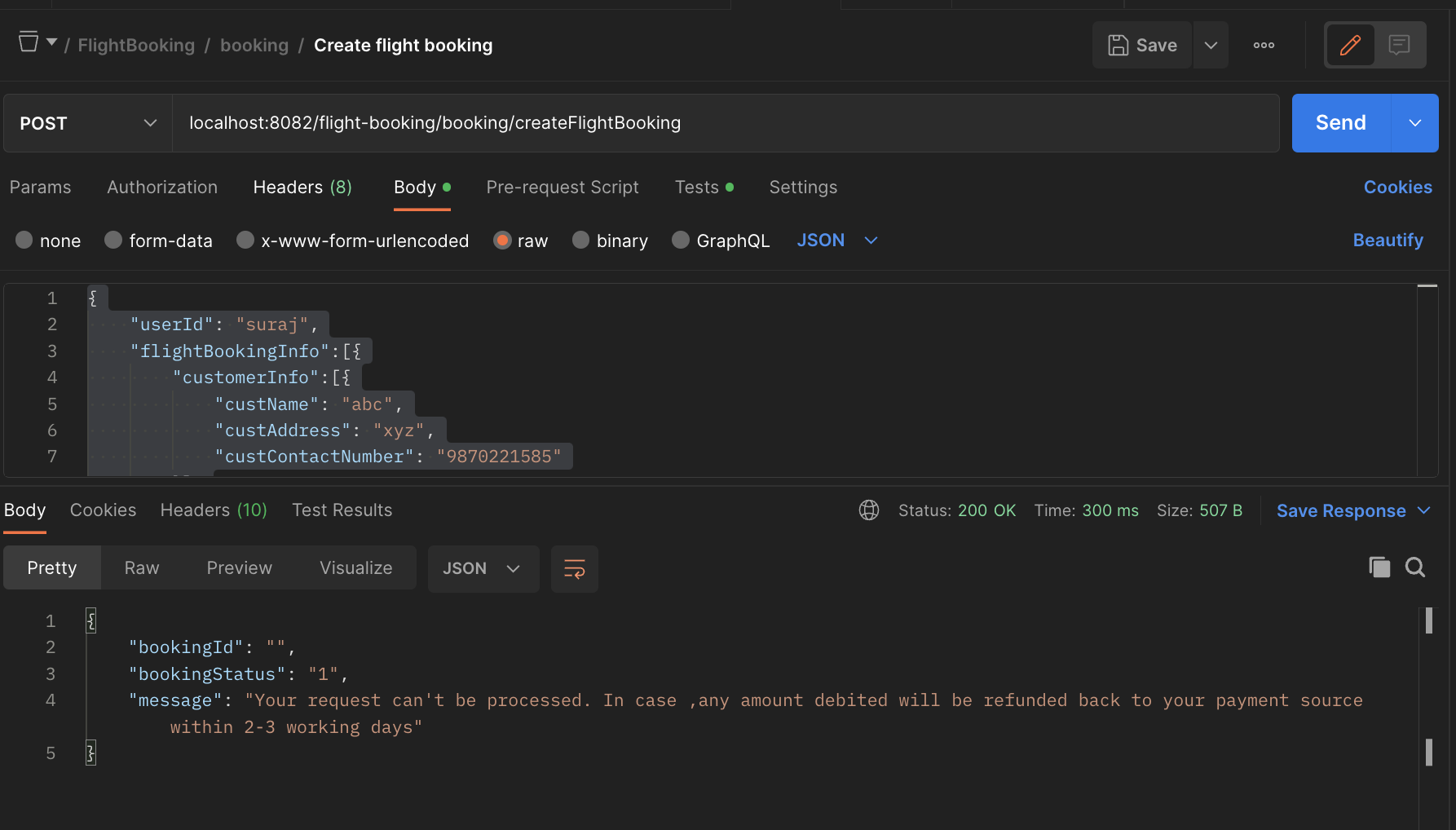


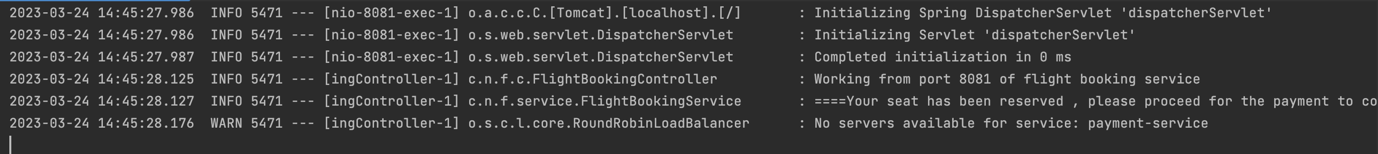


Failure Booking Flow

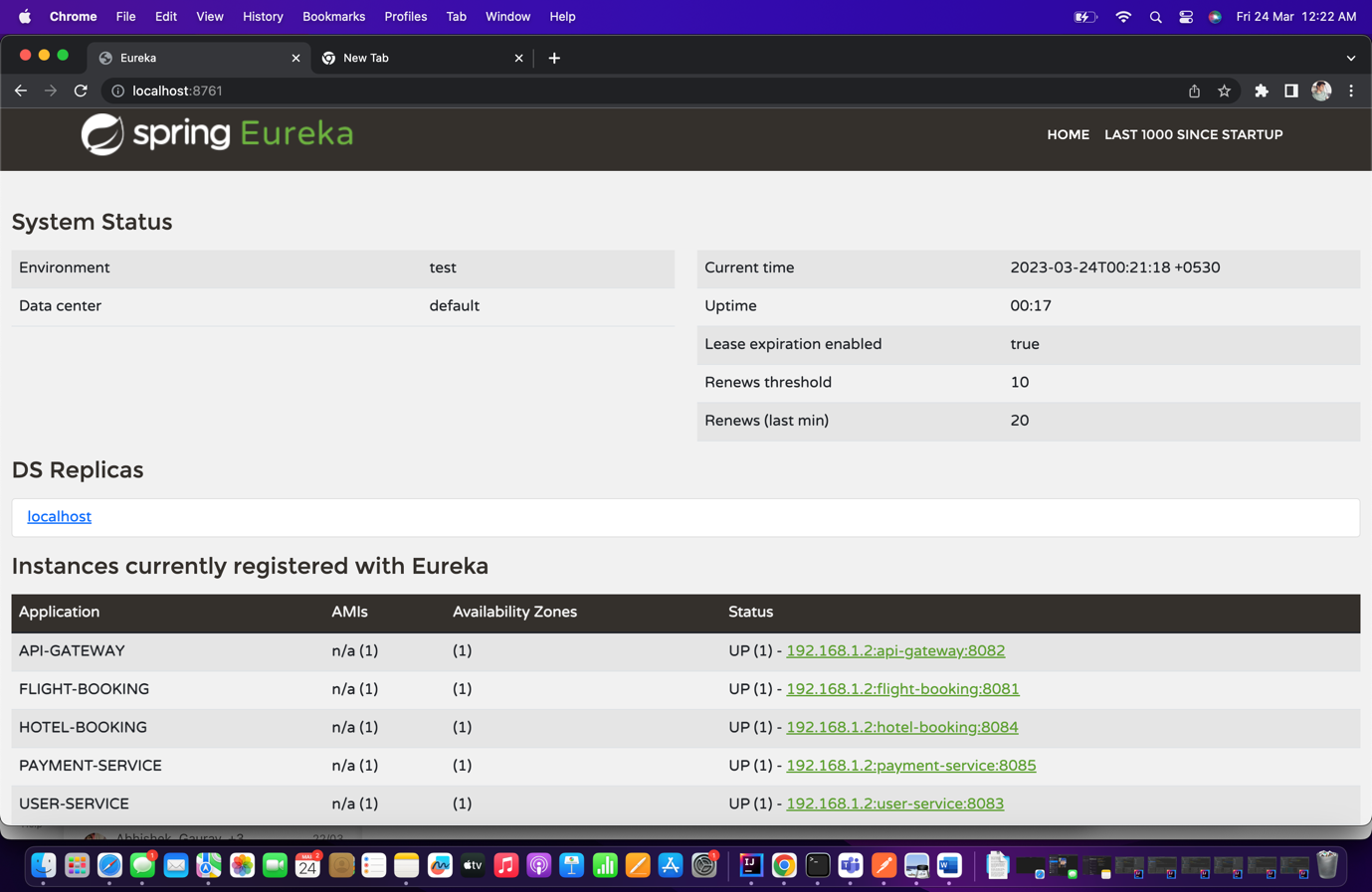
Signup > Login > Search Flight > Create Flight Booking

Failure Scenario 1: - Payment Service is down so ticket get cancelled and user is not able to book tickets.





**Discovery Server Implementation**

****

**Inter-communication approach**

The interactions between various microservices is happening using Rest Template and Web Client where the necessary URL and associated details are fetched using the discovery server.

The API Gateway microservice sits between the user/postman and the microservices for load balancing and to keep the paths dynamic, we have added a discovery-server 'name of service', to which all the discovery clients registers as soon as they boot up.    
  
From client-server perspective, the first interaction is between the API gateway and the user-service for signup and login.

For all the subsequent requests, The API gateway authenticates and authorize using the token and role resp.

The Flight Booking service and Hotel Booking Service for all the payment related functionalities interacts with the Payment Details service.

**Cross-Cutting Concerns**

* Used Slf4j logger for logging.
* Logger is implemented throughout all the microservices also attached some screenshots in above success and failure scenarios.
* Hystrix is used for circuit-breaker pattern.
* Circuit breaker handles all the cases like if any service is down so directly sends the method to its fallback method and returns the response instead of throwing error.

Deployment:-

Docker Images :-

1. surajsachansudo/discovery-server:latest
2. surajsachansudo/api-gateway:latest
3. surajsachansudo/hotel-booking:latest
4. surajsachansudo/flight-booking:latest
5. surajsachansudo/user-service:latest
6. surajsachansudo/payment-service:latest

For deployment attached docker compose inside Deployment and you can run all the containers using command – docker compose up.

**Note:-**

* Overlapping issues are handled in hotel-booking by allocating particular room one by one by checking whether they are available or not.
* For security we are using token verification in user-service with the help of api-gateway.
* Attached postman collection for each microservice inside their repo and also attached a common collection inside api-gateway.
* For notification loggers are implemented for all the required details.

Use Cases:-

* Multiple rooms can be booked using add and confirm API in hotel-booking.
* Room numbers are also allocated after booking confirmation.
* Round trip flight tickets can be booked using create flight booking API.
* Cancellation functionality is also given on both the services hotel & flight.
* In case payment is deducted and still tickets are not booked so refund will be issued in both the cases.

Technologies Used:-

Java – 11

Spring Boot

Docker

Maven

Tools: -

IntelliJ Idea  
Docker Desktop

Postman