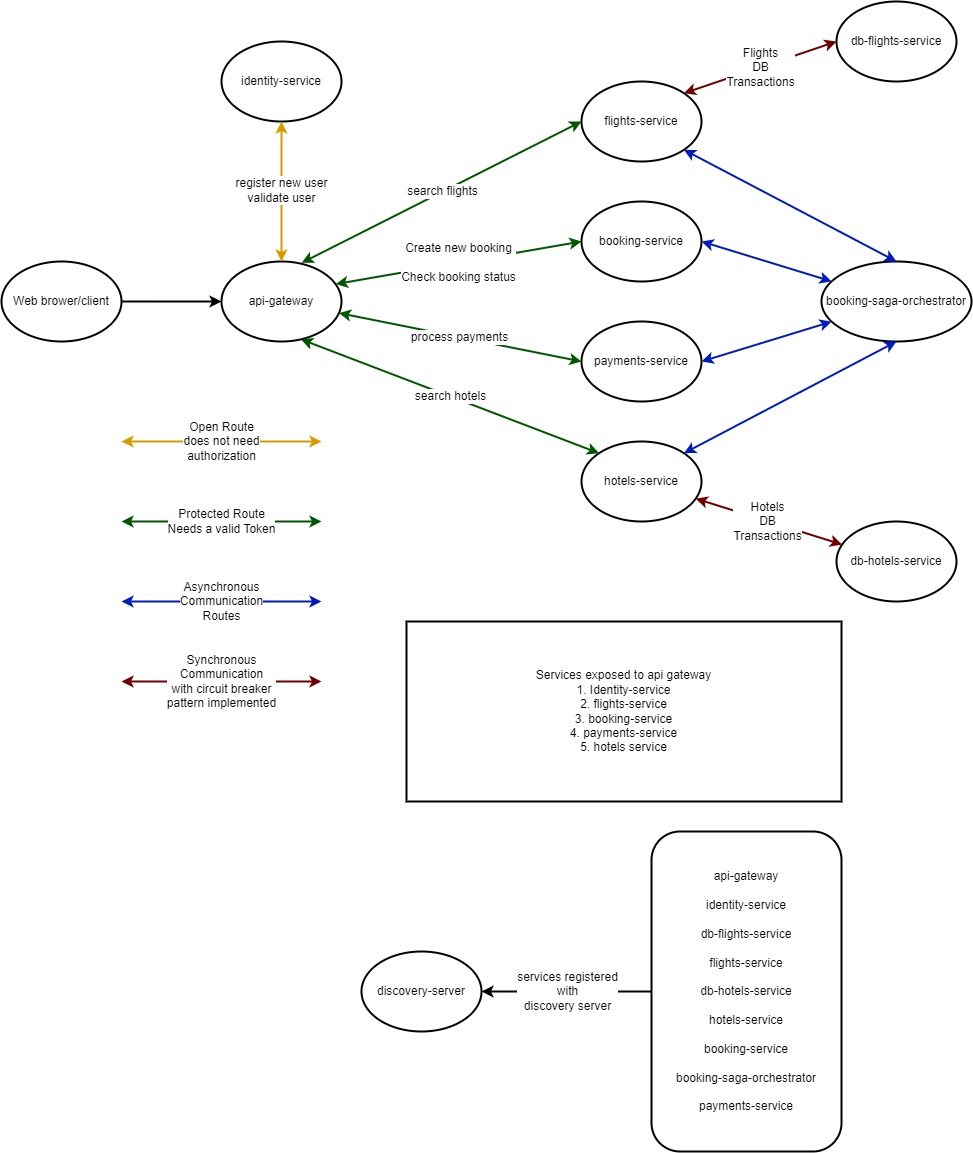
**Microservices assignment 2023**

**Booking System**

**Table of contents**

|  |  |
| --- | --- |
| Topic Name | Page Number |
| Booking System High Level Design | 1 |
| Identified Microservices with explanation | 2 |
| API endpoints with explanation | 3 |
| Successful and Failure Flow during Flight booking | 13 |
|  |  |
|  |  |
|  |  |

**Booking System High Level Design**

****

**Identified Microservices with explanation.**

1. **Identity Service** (identity-service)This deals with user related operations such as registering a new user, log in the user and provide a valid authentication token. The user details along with a valid authentication token is required to create a successful booking.
2. **Flights Database Service** (db-flights-service)This service is created to act as a database that holds flights data. This service provides the capability to read flights in the database. This service updates the flights in the database according to bookings created.
3. **Flights Service** (flights-service)This service is created to provide abstraction to Flights Database Service. As Database can have confidential data related to flights so it cannot be exposed to the public. Therefore, Flights Service deals with user operations and provides the user only the data that is important to the user.

It allows authenticated user to search flights by defining options like source location, destination location, departure date, flightId.

1. **Hotels Database Service** (db-hotels-service)

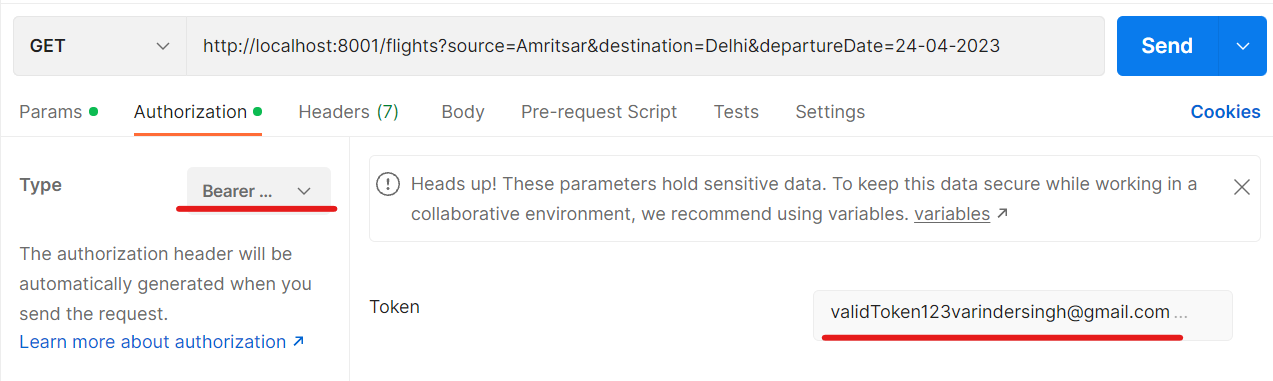
This service is created to act as a database that holds Hotels data. This service provides the capability to read hotels in the database. This service updates the hotels in the database according to bookings created.

1. **Hotels Service** (hotels-service)This service is created to provide abstraction to Hotels Database Service. As Database can have confidential data related to hotels so it cannot be exposed to the public. Therefore, Hotels Service deals with user operations and provides the user only the data that is important to the user.

It allows authenticated user to search hotels by defining options like hotel name, city, hotel address.

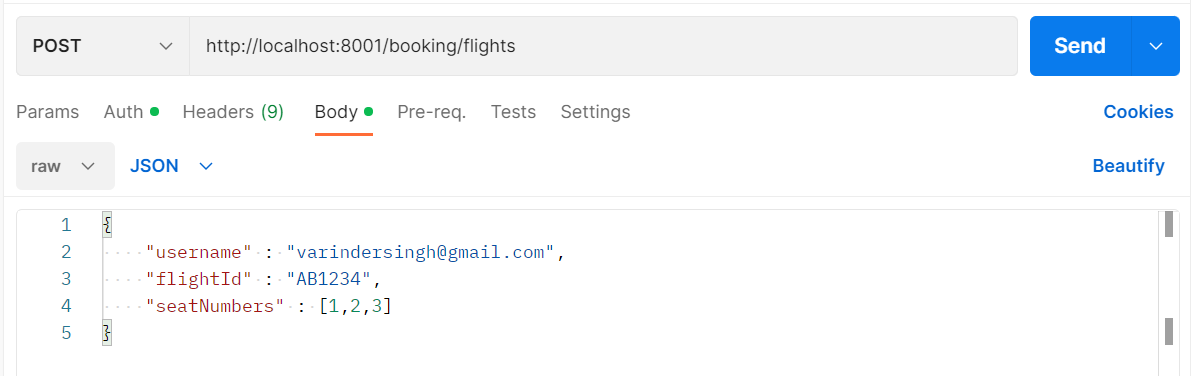
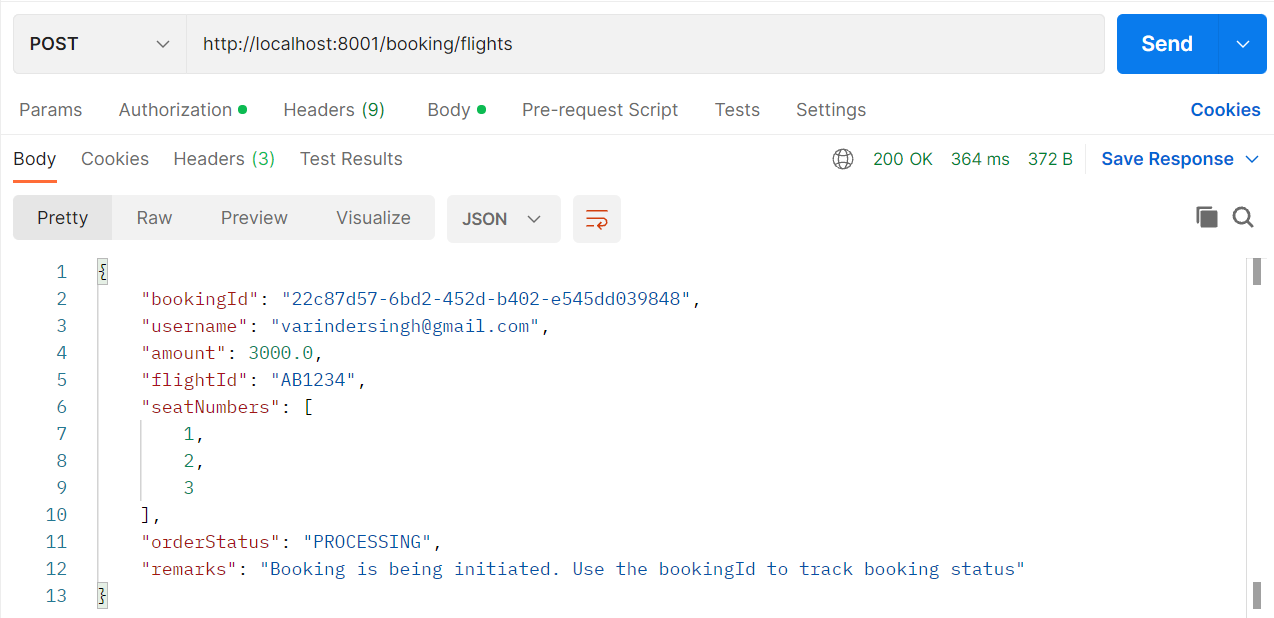
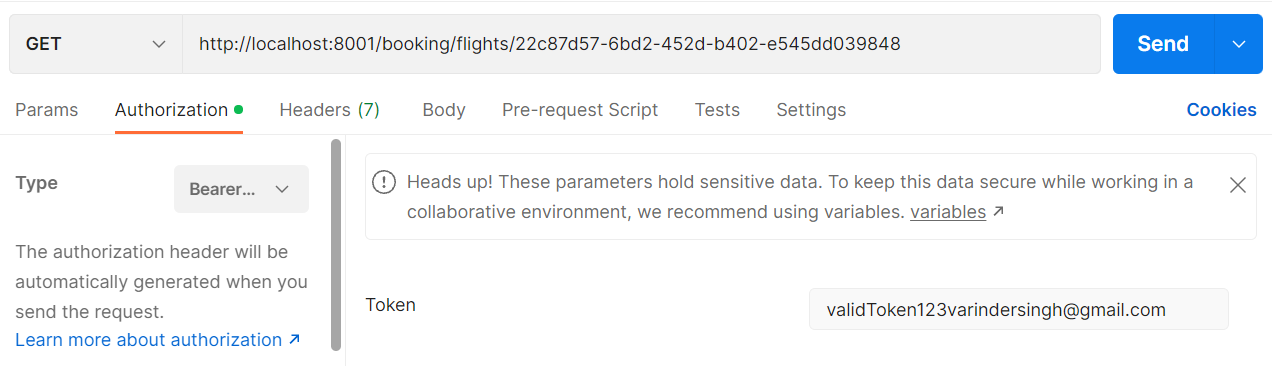
1. **Booking Service** (booking-service)  
   This service is created to create bookings related to flights and hotels. This service keeps tracks of all the transactions related to particular booking and updates it in the booking order stored in it.  
   This service allows authenticated user to create a new booking and view the booking order details.
2. **Payments Service** (payments-service)  
   This service is used to mock user payment requests. This service provides functionality to mock a successful payment request and also to mock a failed payment request. These two scenarios are important to showcase fault tolerance features of the system.
3. **Booking Saga Orchestrator** (booking-saga-orchestrator)  
   This service is based on saga orchestrator pattern in microservices. This service manages all the asynchronous communication that is happening between different microservices. This service coordinates the events in a systematic manner according to transactional flow required for the booking system.
4. **Discovery Server** (discovery-server)  
   This service is a Eureka Discovery Server which helps the various microservices to communicate with each other. This service removes the necessity for a microservice to know the actual address of other microservice to communicate with it. All the microservices that are meant to communicate with each other are registered to discovery server as discovery clients.

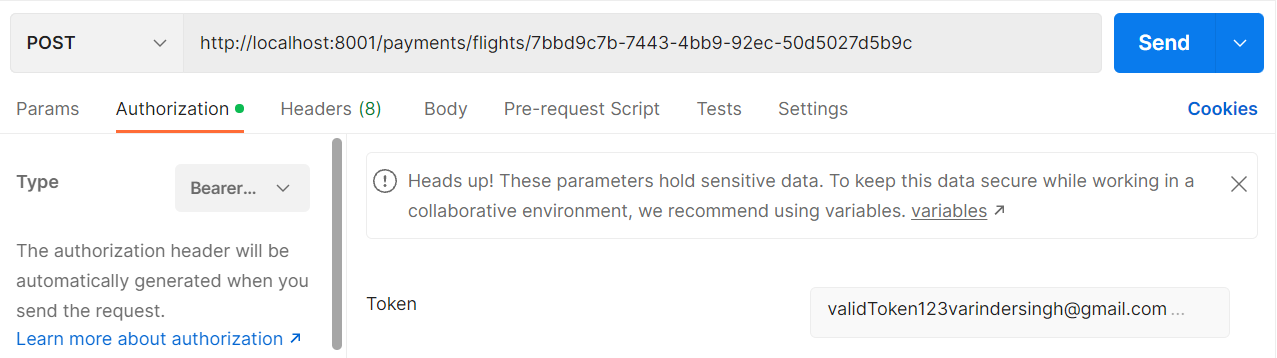
**API endpoints with explanation**

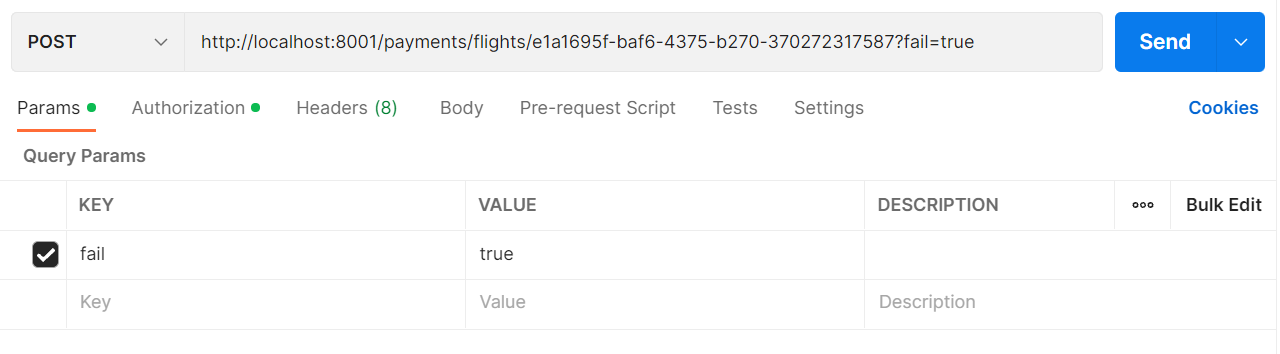
1. **Api gateway endpoints:**This service is used to map endpoints of different microservices under a single domain.  
     
   Base Uri: <http://localhost:8001>  
   Various microservices are mapped to api gateway as show below  
   1. **Register a new user ( identity-service)**  
      Uri : <http://localhost:8001/users>  
      Method : Post  
      All fields are required  
      Request Body : JSON  
      Result :  
      
   2. **Login user (identity service)**  
      Uri : <http://localhost:8001/users/login>   
      Method : Post  
      All fields are required  
      Request Body : JSON  
      Result :  
      
   3. **Search Flight (filghts-service)**Uri : http://localhost:8001/flights?source=Amritsar&destination=Delhi&departureDate=24-04-2023  
      Method : Get  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Query params (all query params are optional, if not provided all flights will be shown)  
      1. source = name of the source city  
      2. destination = name of the destination city  
      3. departureDate = date of travel must be in DD-MM-YYY format  
        
        
        
        
        
        
        
        
        
      Request  
        
        
      Result  
      Graphical user interface, text, application, email

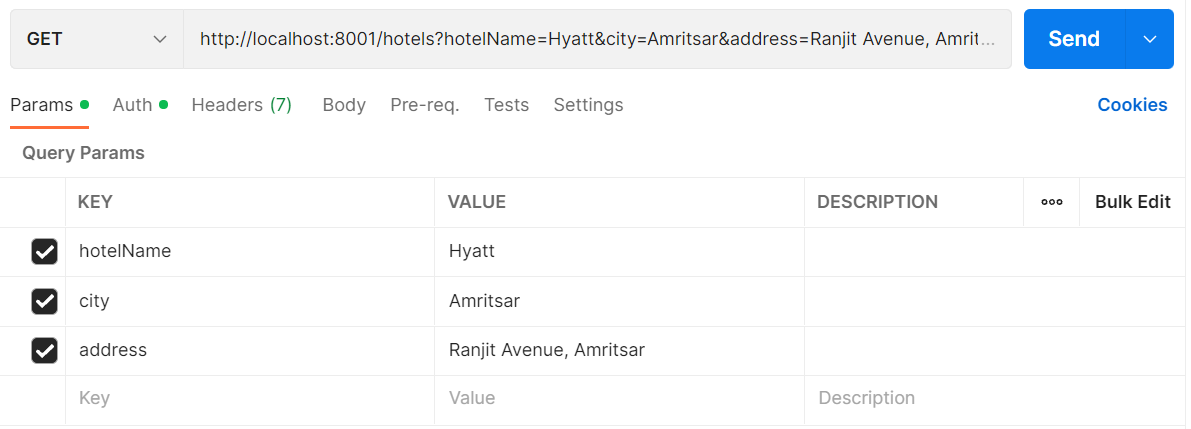
      Description automatically generated
   4. **Search a specific flight (flights-service)**  
      Uri : [http://localhost:8001/flights/{flightd}](http://localhost:8001/flights/%7bflightd%7d)  
      Method : Get  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Path Variable (required)  
      1. flightId = Flight Id of a flight.  
        
      Request  
      Graphical user interface, text, application

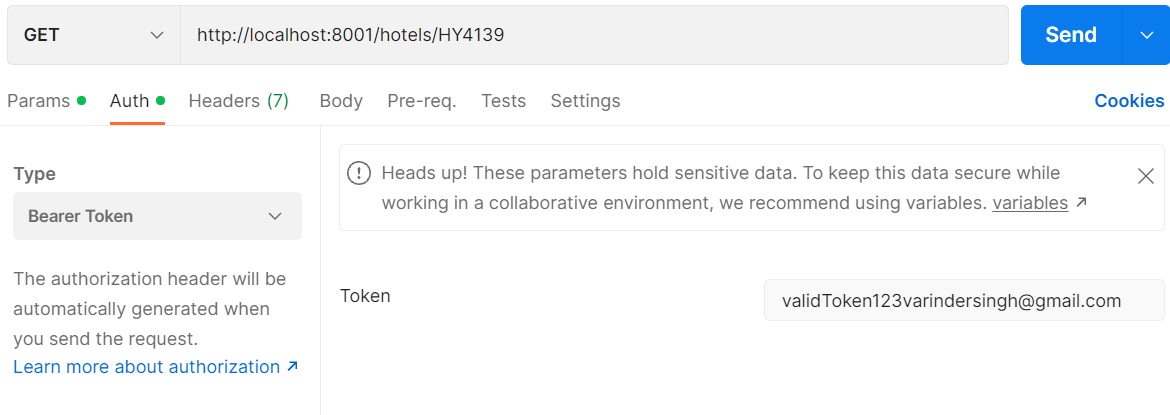
      Description automatically generated  
        
      Response  
      Graphical user interface, text, application, email

      Description automatically generated
   5. **Book a Flight (booking-service)**Uri : http://localhost:8001/booking/flights  
      Method : Post  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Request Body : Json (All fields required)  
      1. username = username of the user booking flight  
      2. flightId = Flight Id of the flight to be booked  
      3. seatNumbers = Array of seat numbers user wish to book. (seats available 1 to 30)   
        
        
      Request: ****Response:  
      
   6. **Check Flight Booking status (booking-service)**Uri : http://localhost:8001/booking/flights/{bookingId}  
      Method : Get  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Path Variable (required)  
      1. bookingId = bookingId is unique id to specify the booking order.  
        
      Request:  
        
        
      Response:  
      Graphical user interface, text, application

      Description automatically generated
   7. **Send Payment (payment-service)**It is payment mocking service and can be used to perform two tasks  
      1. To mock payment success  
      2. To mock payment failure  
        
      **I. Flights Payment Success Mocking**Uri : http://localhost:8001/payments/flights/{bookingId}   
      Method : Post  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Path Variable (required)  
      1. bookingId = bookingId is unique id to specify the booking order.  
        
      Request:  
        
        
      Response:  
      **Graphical user interface, text, application

      Description automatically generated**  
        
        
      2. **Flights Payment Failure Mocking**Uri : http://localhost:8001/payments/flights/{bookingId}?fail=true   
      Method : Post  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Path Variable (required)  
      1. bookingId = bookingId is unique id to specify the booking order.  
        
      To make the payment fail add the following query parameter in the URI.  
      Query Param (Required)  
      1. fail=true  
        
      Request:  
        
        
        
        
      Response:  
      Graphical user interface, text, application

      Description automatically generated
   8. **Search Hotels (hotels-service)**Uri: http://localhost:8001/hotels?hotelName=Hyatt&city=Amritsar&address=Ranjit Avenue, Amritsar  
      Method: Get  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
        
      Query params (all query params are optional, if not provided all flights will be shown)  
      1. hotelName = name of the hotel   
      2. city = name of the hotel city  
      3. address = complete address of a hotel  
        
      Request:  
        
        
      Result:  
      Graphical user interface, text, application

      Description automatically generated
   9. **Search a specific hotel (hotels-service)**  
      Uri : http://localhost:8001/hotels/HY4139  
      Method : Get  
      Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
      Path Variable (required)  
      1. hotelId = Hotel Id of a hotel.  
        
      Request  
        
        
      Response  
      Graphical user interface, text, application

      Description automatically generated
   10. **Book a Hotel (booking-service)**Uri : http://localhost:8001/booking/hotels  
       Method : Post  
       Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
         
       Request Body : Json (All fields required)  
       1. username = username of the user booking flight  
       2. hotelId = Hotel Id of the hotel to be booked  
       3. checkInDate = Check In Date (required date format is DD-MM-YYYY)  
       4. checkOutDate = Check Out Date (required date format is DD-MM-YYYY)  
       5. roomsRequired = Number of rooms to be booked  
       Request:  
       **Graphical user interface, text, application, email

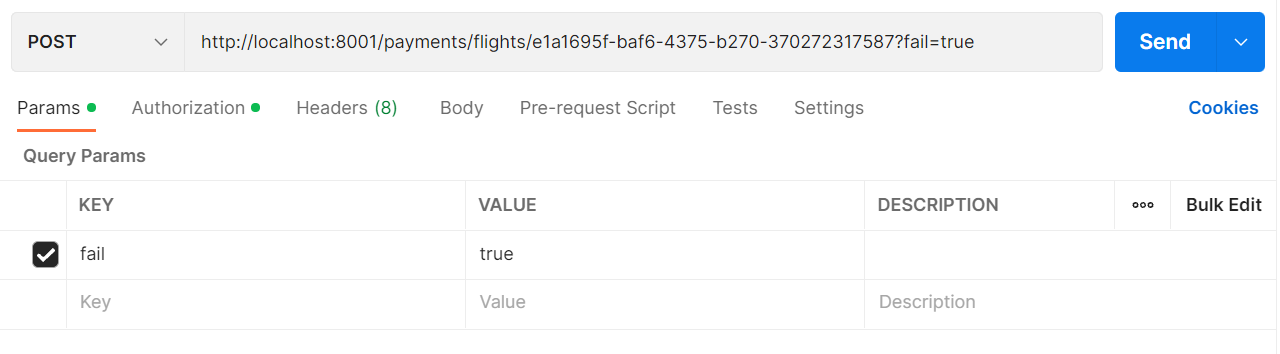
       Description automatically generated**Response :  
       Graphical user interface, text, application

       Description automatically generated
   11. **Check Hotel Booking status (booking-service)**Uri : http://localhost:8001/booking/hotels/{bookingId}  
       Method : Get  
       Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
         
       Path Variable (required)  
       1. bookingId = bookingId is unique id to specify the booking order.  
         
       Request:  
       Graphical user interface, text, application, email

       Description automatically generated  
         
         
         
         
         
         
       Response:  
       Graphical user interface, text, application

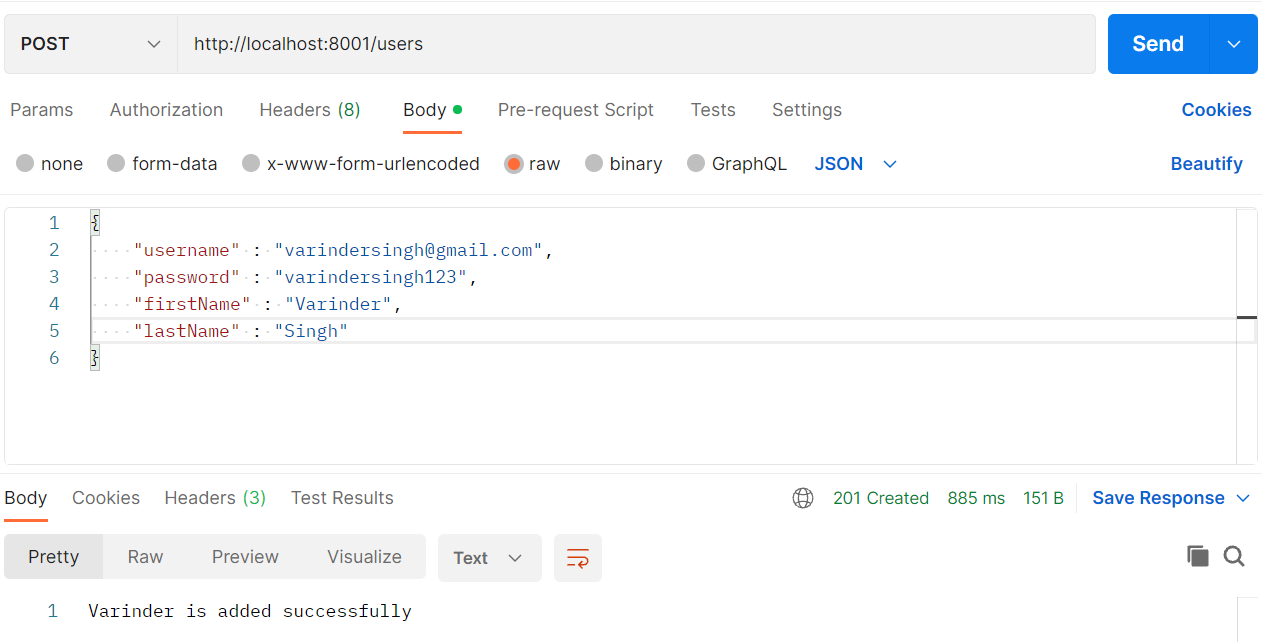
       Description automatically generated
   12. **Send Payment (payment-service)**It is payment mocking service and can be used to perform two tasks  
       1. To mock payment success  
       2. To mock payment failure  
         
       **I. Hotels Payment Success Mocking**Uri : http://localhost:8001/payments/hotels/{bookingId}   
       Method : Post  
       Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
         
       Path Variable (required)  
       1. bookingId = bookingId is unique id to specify the booking order.  
         
       Request:  
       Graphical user interface, text, application, email

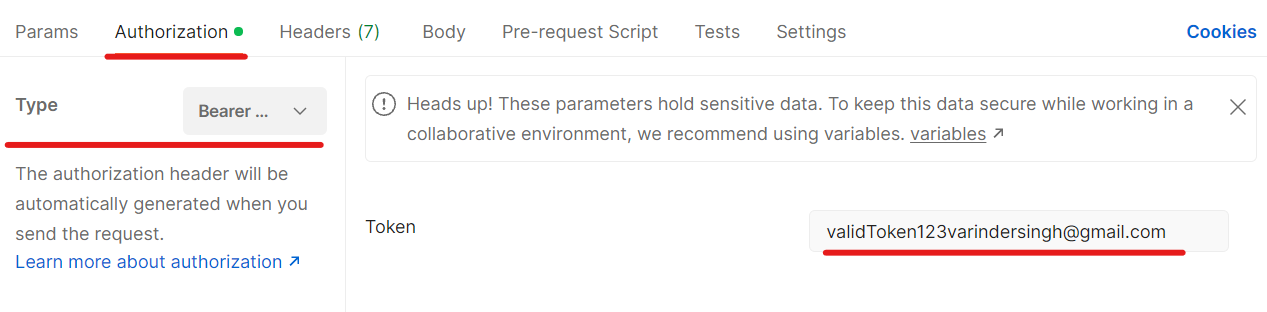
       Description automatically generated  
         
       Response:  
       **Graphical user interface, text, application

       Description automatically generated**  
         
         
         
       2. **Hotels Payment Failure Mocking**Uri : http://localhost:8001/payments/hotels/{bookingId}?fail=true   
       Method : Post  
       Authentication is Required, kindly provide valid token under Bearer Token in Postman.  
         
       Path Variable (required)  
       1. bookingId = bookingId is unique id to specify the booking order.  
         
       To make the payment fail add the following query parameter in the URI.  
       Query Param (Required)  
       1. fail=true  
         
       Request:  
         
         
         
         
       Response:  
       Graphical user interface, text, application

       Description automatically generated

**Successful and Failure Flow during Flight booking**

1. **Successful flow flight booking**  
   Follow the following steps to create a successful booking.
2. Register a new user.  
   
3. Login and get a valid token to carry out further requests.  
   Graphical user interface, text, application, email

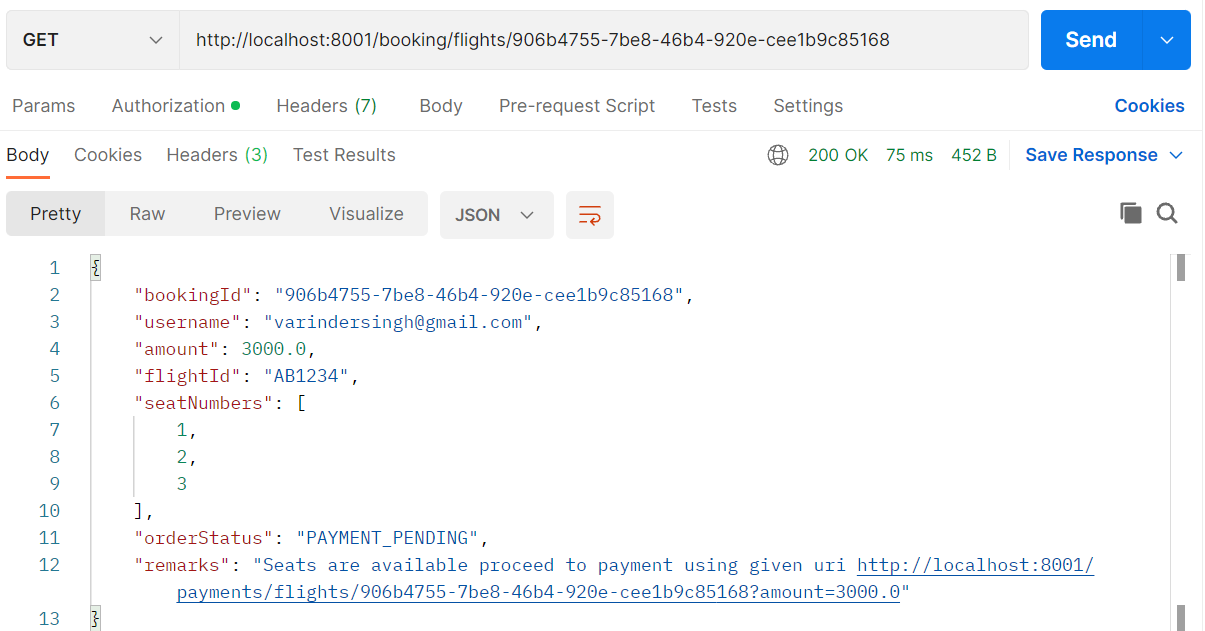
   Description automatically generated
4. Place the valid token under   
   Authorization => Type => Bearer Token field in Postman application for the below mentioned requests as these are protected routes.  
   
5. Search Flights (for example lets search for flights from Amritsar to Delhi)  
   Graphical user interface, text, application

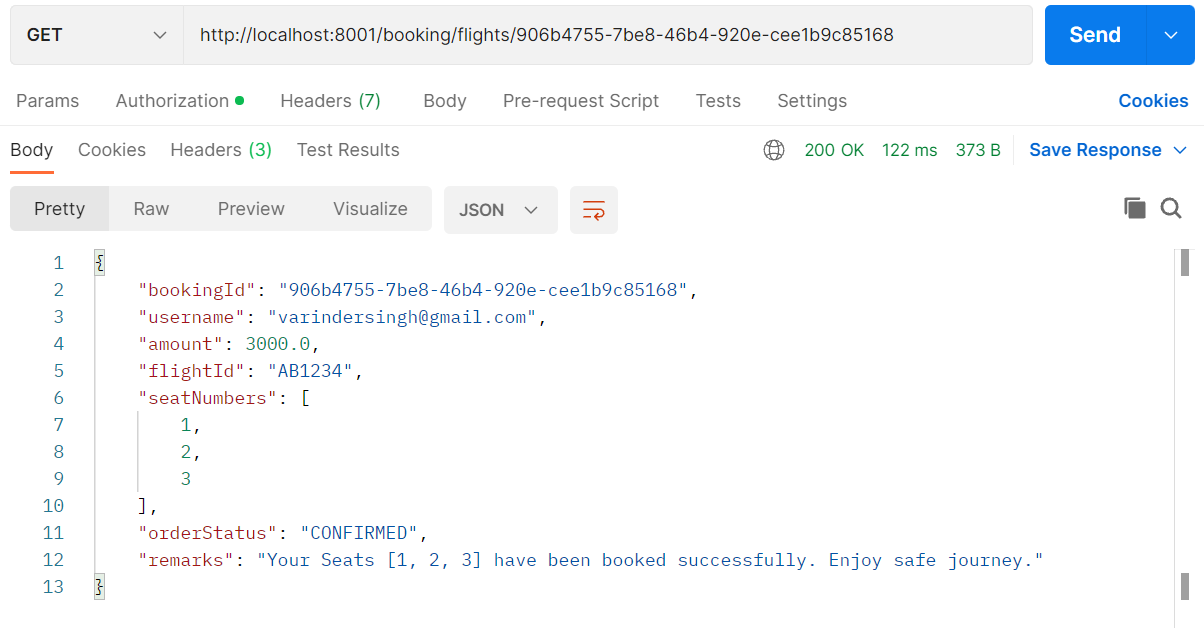
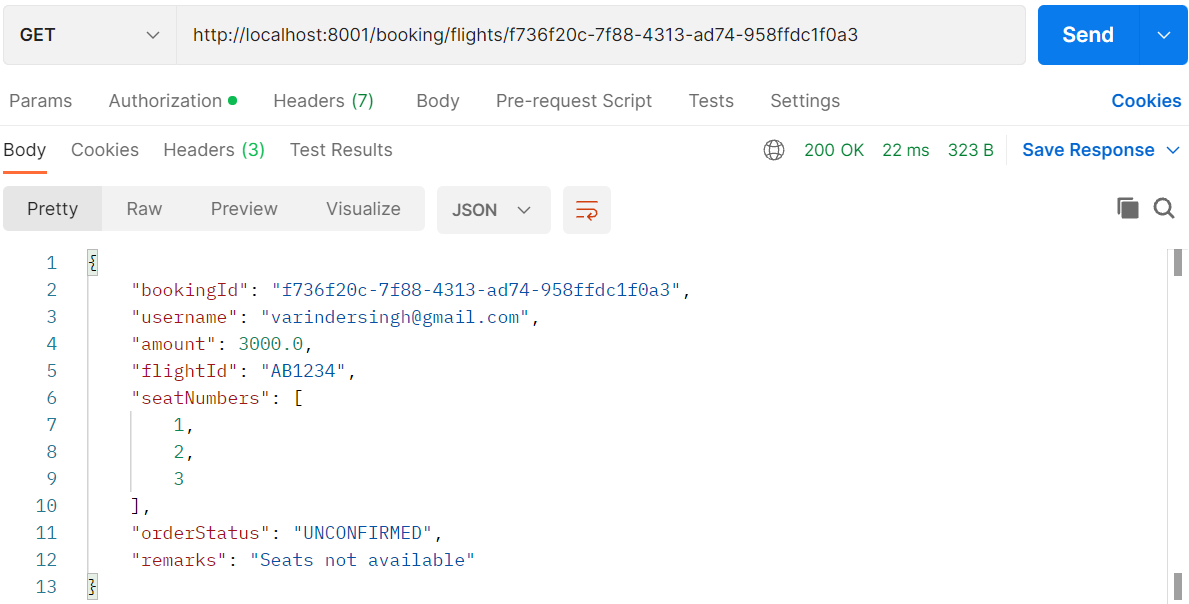
   Description automatically generated  
     
   Graphical user interface, text, application

   Description automatically generated
6. Copy the flightId of the flight whose complete details you wish to find i.e. which seats are available.  
   let’s use the flightId = AB1234 for further transactions.
7. Using flightId get details of a flight.  
   Graphical user interface, text, application

   Description automatically generated
8. See which seats are available and choose the ones to book.
9. Request to create a booking as shown below.  
   Let’s book seats [1,2,3]  
     
   You will get your booking details in response where the initial status of the booking will be PROCESSING.   
   Graphical user interface, text, application, email

   Description automatically generated  
     
   Graphical user interface, text, application, email

   Description automatically generated  
   Copy the bookingId to get further updates on your booking order.
10. Check booking order status using bookingId.  
      
    Here you will see the PAYMENT\_PENDING status.
11. Proceed to payment using the same bookingId or copy the payment url shown in booking.  
    Graphical user interface, text, application, email

    Description automatically generated  
    On successful payment, you will see payment is successful in response and your booking status will be updated accordingly.
12. Now after successful payment the application will again confirm whether the required seats are available or not. If seats are available then the seats will be marked booked and booking status will then be updated to Confirmed.  
      
    Check booking once again.  
      
    This how a successful flight booking flow will work.
13. **Negative/Failure flow flight booking.**There can be two scenarios leading to failure of booking when all the services are working i.e. below mentioned failures does not occurs when some of the services are down.  
      
    1. User try to book seats which are already booked before making payments.  
      
      
    2. A case when two users have created booking for same seats in a flight but have not proceed with payments. In this case the bookings will be created for both users.  
    But say user1 proceeds with payment first, then user1 will be able to book successfully.  
    But now when user2 proceeds with payments, the payments will be successful but then the application will notice that the seats requested by user2 have been booked already. So in this case the seats can’t be booked therefore a payment refund event will be generated for user2 and user2’s booking will be marked unconfirmed with suitable remarks.  
      
    Graphical user interface, text, application, email

    Description automatically generated

**Assumptions made for the above identified services.**

1. identity-service :  
   This service will perform users related tasks like  
   a. registering new user  
   b. logging in the user and providing a valid token
2. db-flights-service :  
   This service will act as a database for all the flights details. It will provide the functionality to  
   a. read flights data  
   b. update flights data
3. flights-service :  
   This service is created to consume the db-flights-service such that the db-flights-service is not exposed publicly. Because the user may be able to directly modify the flights data in database if db-flights-service is exposed to api-gateway/publicly.  
   This service uses circuit breaker pattern to consume db-flights-service and activemq to communicate asynchronously.  
   This service provide functionality like  
   a. view flights where we can filter flights using source, destination, departure date.  
   b. view complete data for a specific flight
4. booking-service :  
   This service is created to create a booking order for a user. This service will get a post request for creating a booking. Then it will initiate the booking and will dispatch various events like check seat availability, payment request etc. to keep the booking order up to date.
5. booking-saga-orchestrator :  
   This service is based on the saga pattern used in microservices to carry out the asynchronous requests in a systematic order.  
   This service manages the events between booking-service, payments-service, flights-service.
6. payments-service :  
   This service is created to mock the payment requests in the booking transactions.

**To Be Continued…**