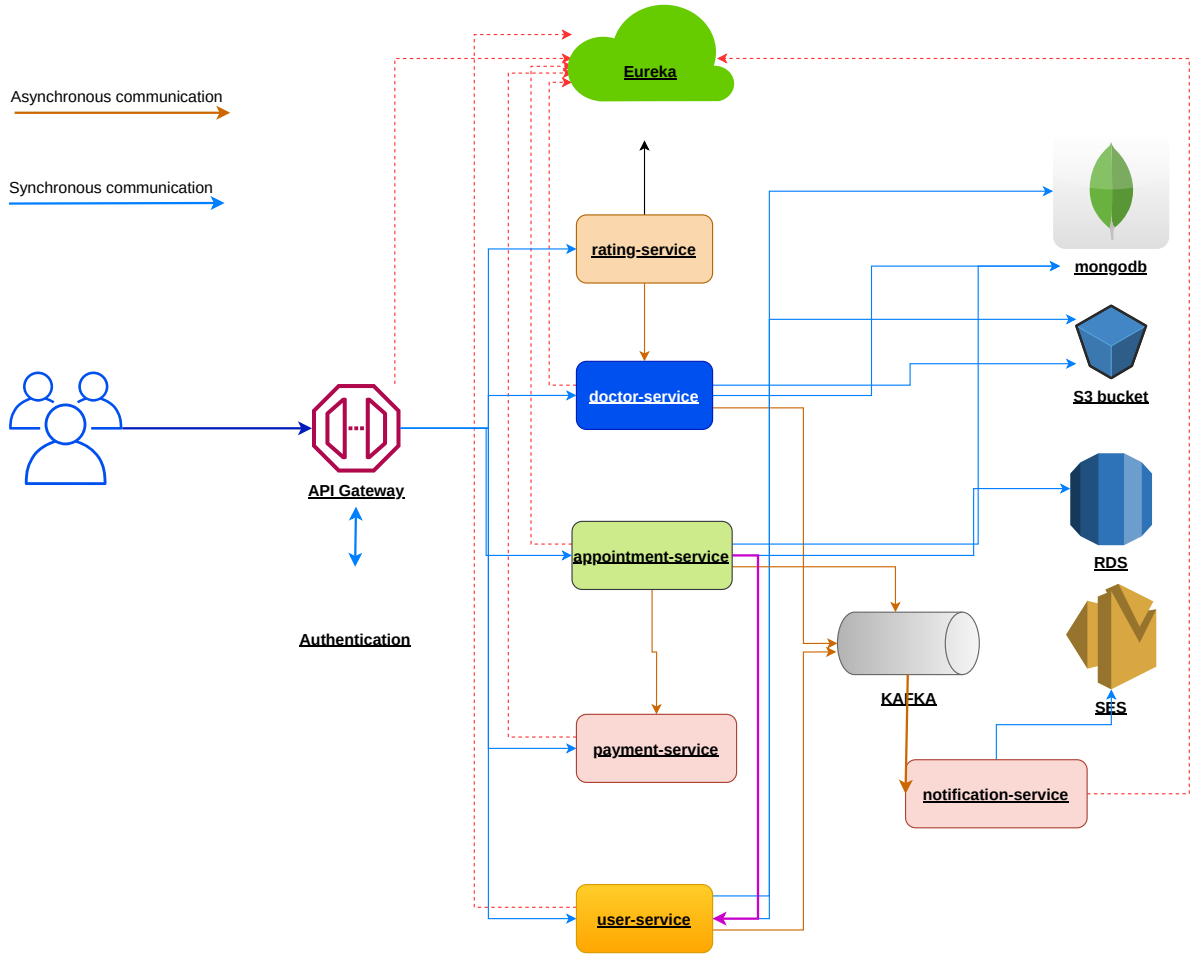
CodeLogic.pdf

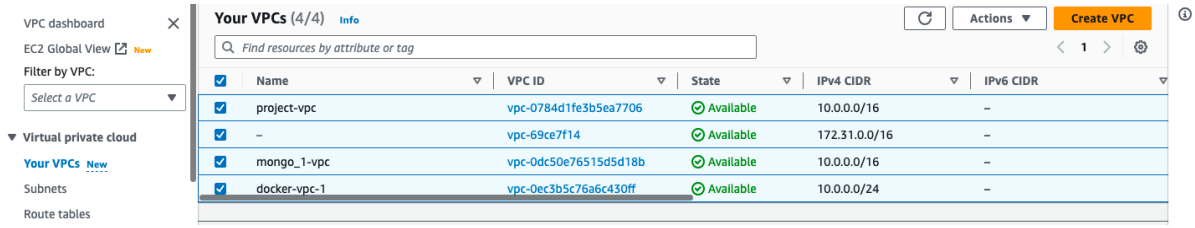
Architecture



Setup Required to run application

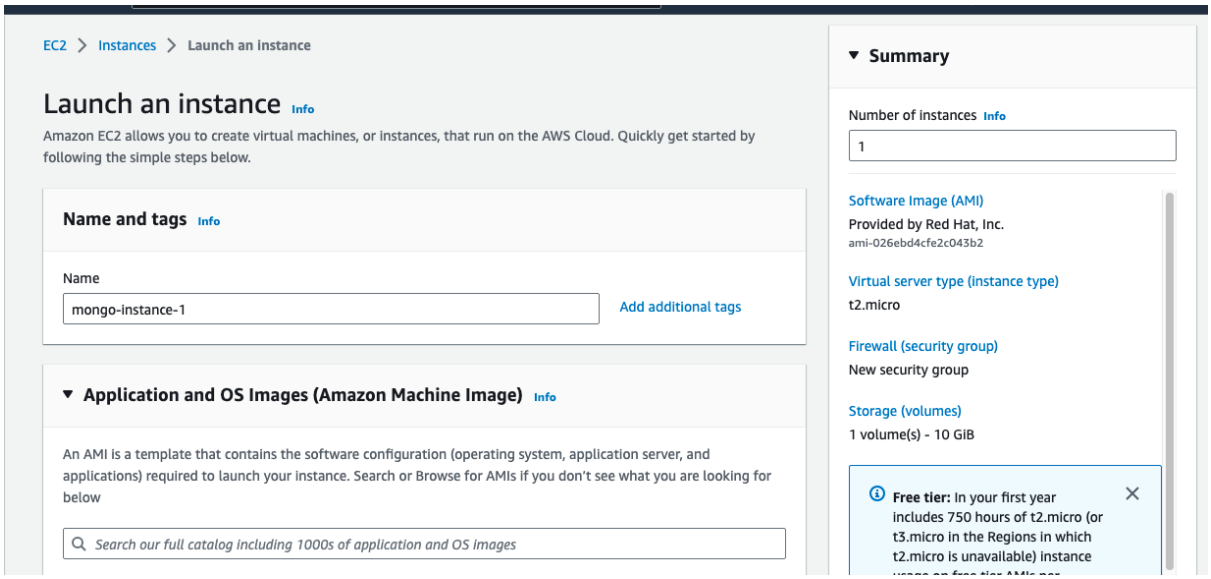
VPC setup:

- create an `vpc` first before creating the resources in aws so we can use same vpc for all the resources.



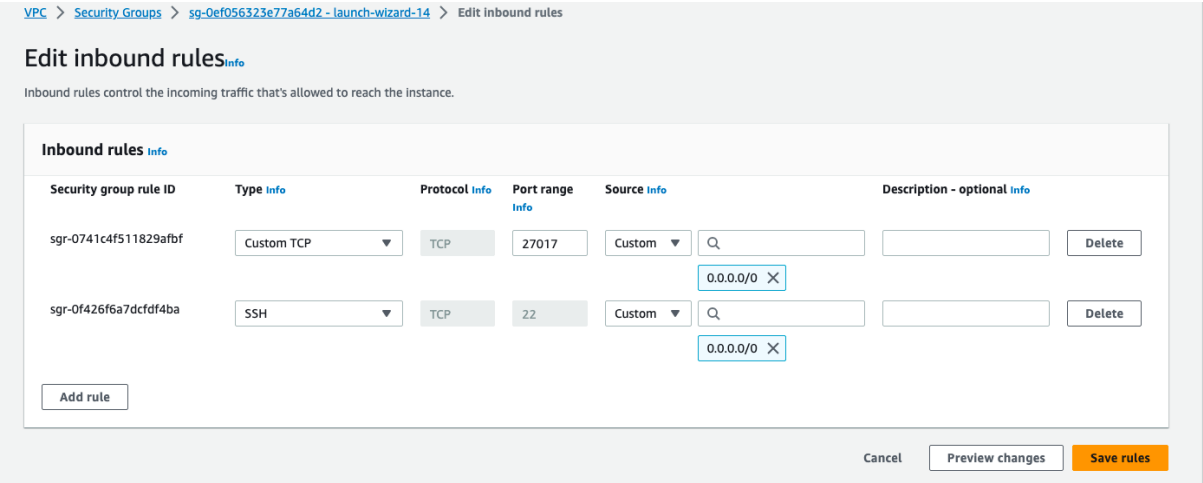
MongoDb setup:

- we create an `ec2` instance of `t2-micro` and install mongodb in that



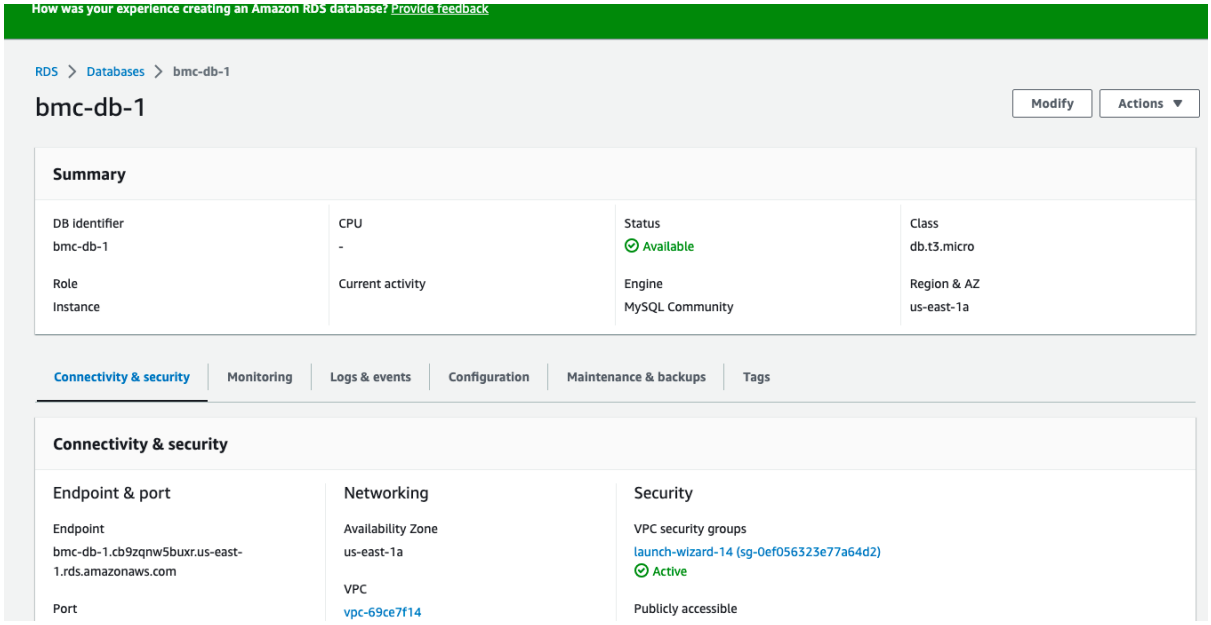
```
[ec2-user@ip-172-31-33-151 ~]$ systemctl status mongod
● mongod.service - MongoDB Database Server
     Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; preset: disabled)
     Active: active (running) since Wed 2023-08-16 03:56:46 UTC; 2h 28min ago
       Docs: https://docs.mongodb.org/manual
    Process: 927 ExecStartPre=/usr/bin/mkdir -p /var/run/mongodb (code=exited, status=0/SUCCESS)
    Process: 930 ExecStartPre=/usr/bin/chown mongod:mongod /var/run/mongodb (code=exited, status=0/SUCCESS)
    Process: 931 ExecStartPre=/usr/bin/chmod 0755 /var/run/mongodb (code=exited, status=0/SUCCESS)
    Process: 935 ExecStart=/usr/bin/mongod $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 965 (mongod)
     Memory: 153.4M
        CPU: 27.907s
     CGroup: /system.slice/mongod.service
             └─965 /usr/bin/mongod -f /etc/mongod.conf

Aug 16 03:56:43 ip-172-31-33-151.ec2.internal systemd[1]: Starting MongoDB Database Server...
Aug 16 03:56:43 ip-172-31-33-151.ec2.internal mongod[935]: about to fork child process, waiting until server is ready fo
Aug 16 03:56:43 ip-172-31-33-151.ec2.internal mongod[965]: forked process: 965
Aug 16 03:56:46 ip-172-31-33-151.ec2.internal mongod[935]: child process started successfully, parent exiting
Aug 16 03:56:46 ip-172-31-33-151.ec2.internal systemd[1]: Started MongoDB Database Server.
lines 1-19/19 (END)
```

Change security group to accept external request for mongo port `27017`

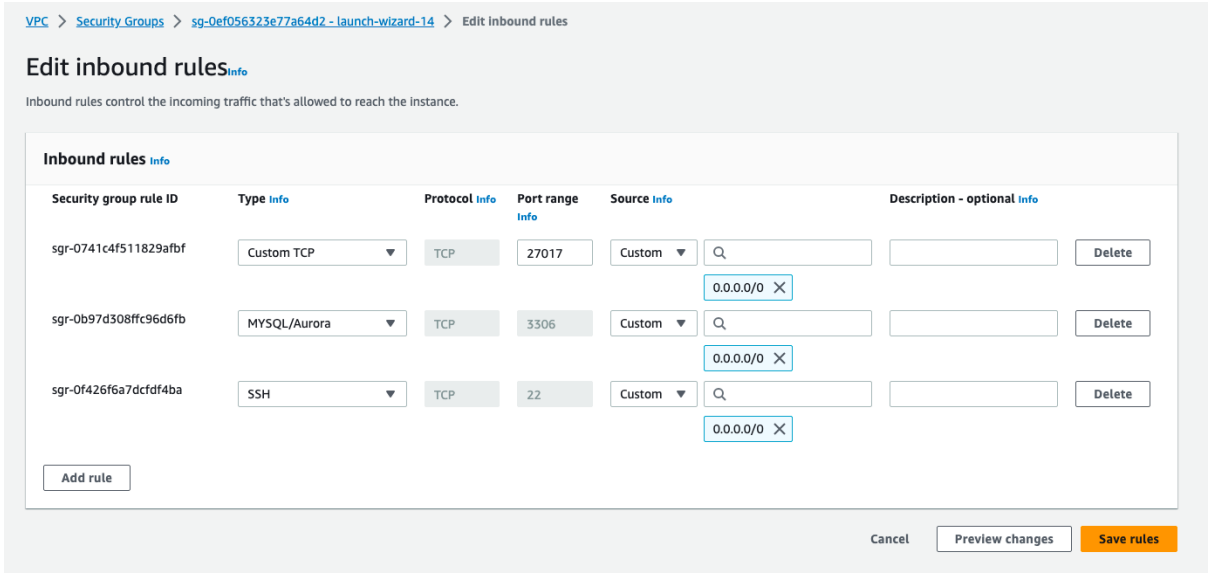


RDS setup:

- we required relational database for running our application so set we setup `DynamoDb` `rds` instance

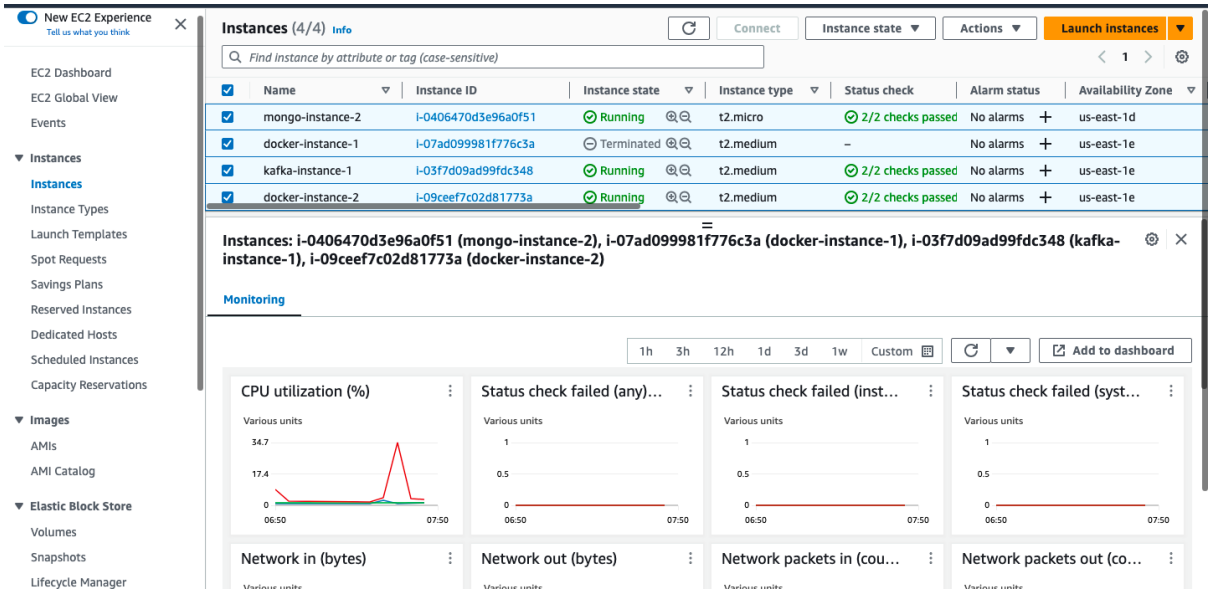


Change security group to accept external request for `rds` port `3306`



Kafka Setup

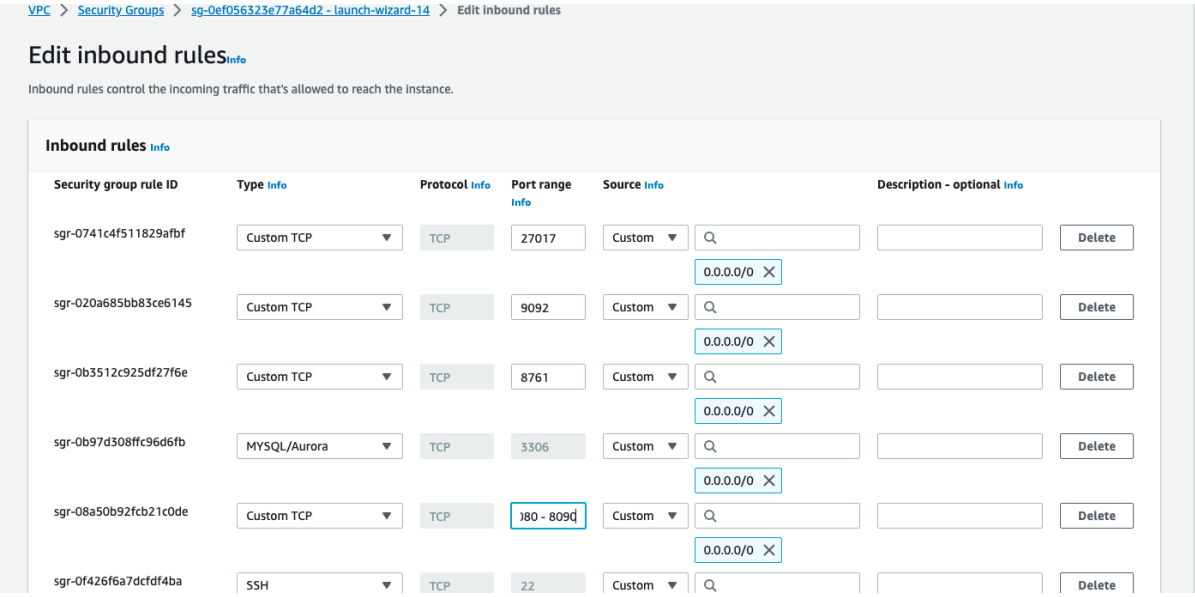
- we are using external kafka server for our application.
- so created one more `ec2` instance of `t2-medium` and installed kafka there



start kafka from binaries

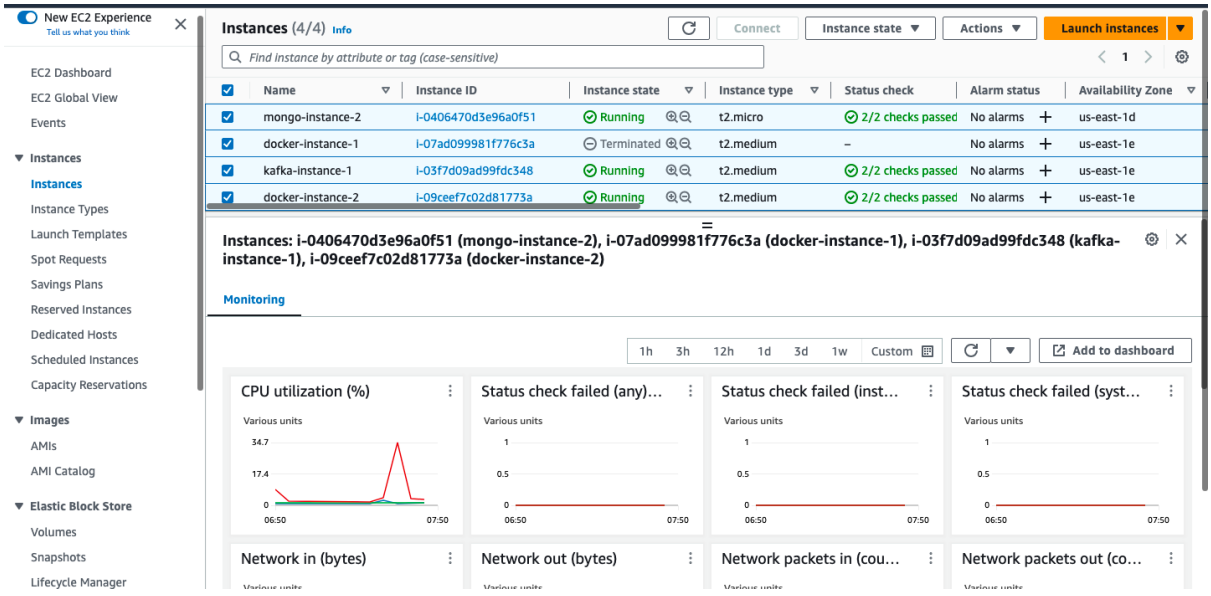
```
[ec2-user@ip-172-31-48-24 kafka_2.12-3.5.1]$ ./bin/zookeeper-server-start.sh ./config/zookeeper.properties ^C
[ec2-user@ip-172-31-48-24 kafka_2.12-3.5.1]$ ./bin/kafka-server-start.sh ./config/server.properties
```

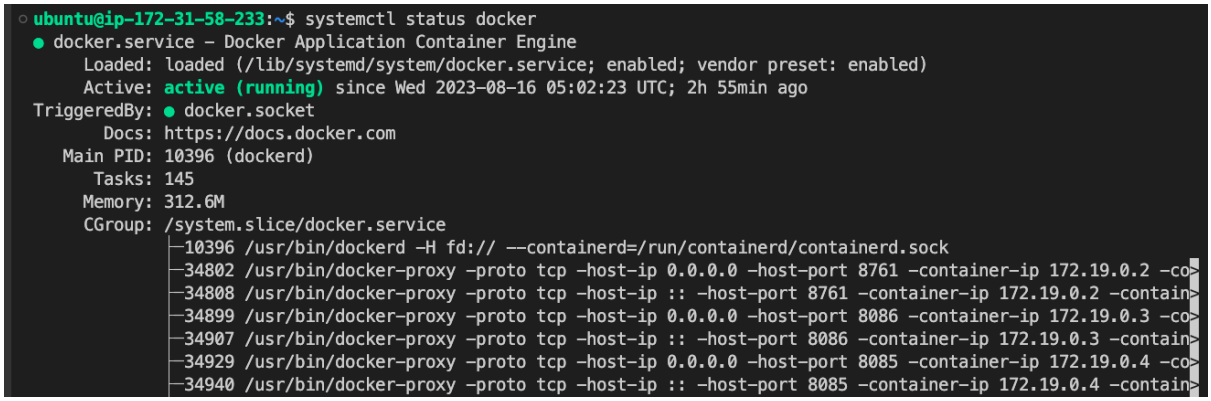
edit security group to access kafka from external open port `9092`



Docker setup

- to setup docker we created a one more instance with `t2-medium`
- then installed docker

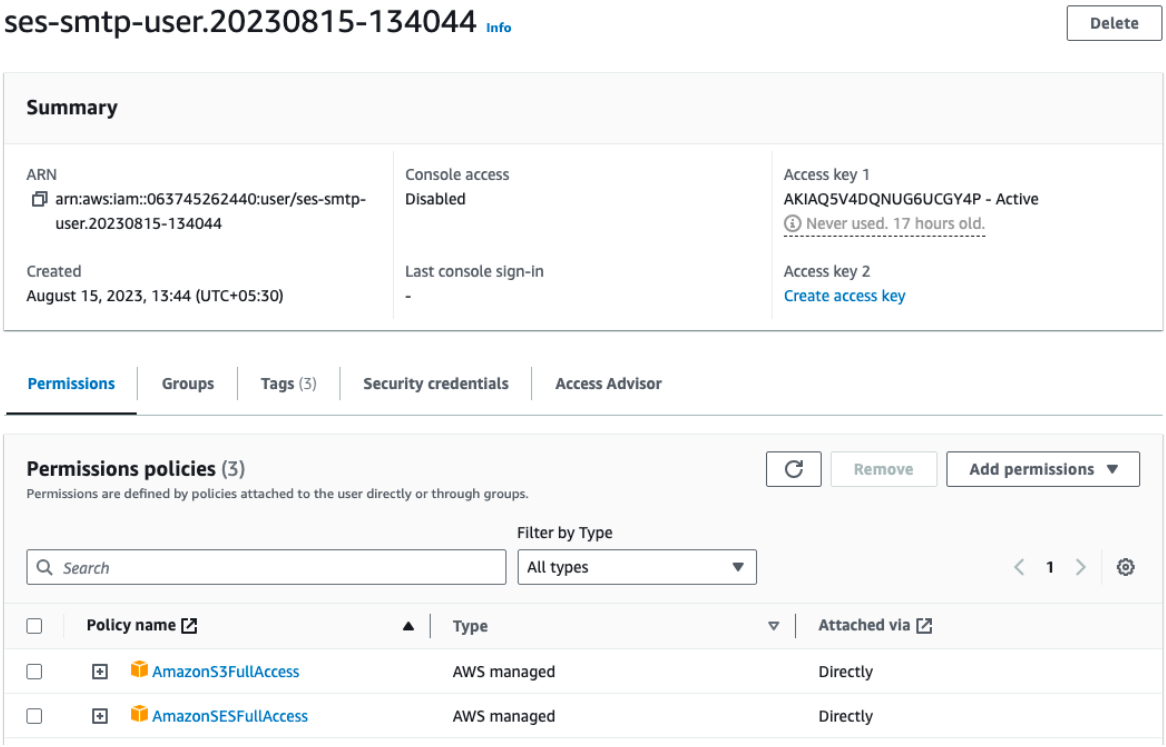




- make sure `rds` , `mongodb` and `kafka` services must be accessible from `docker instance`

IAM user setup for SES and S3 access

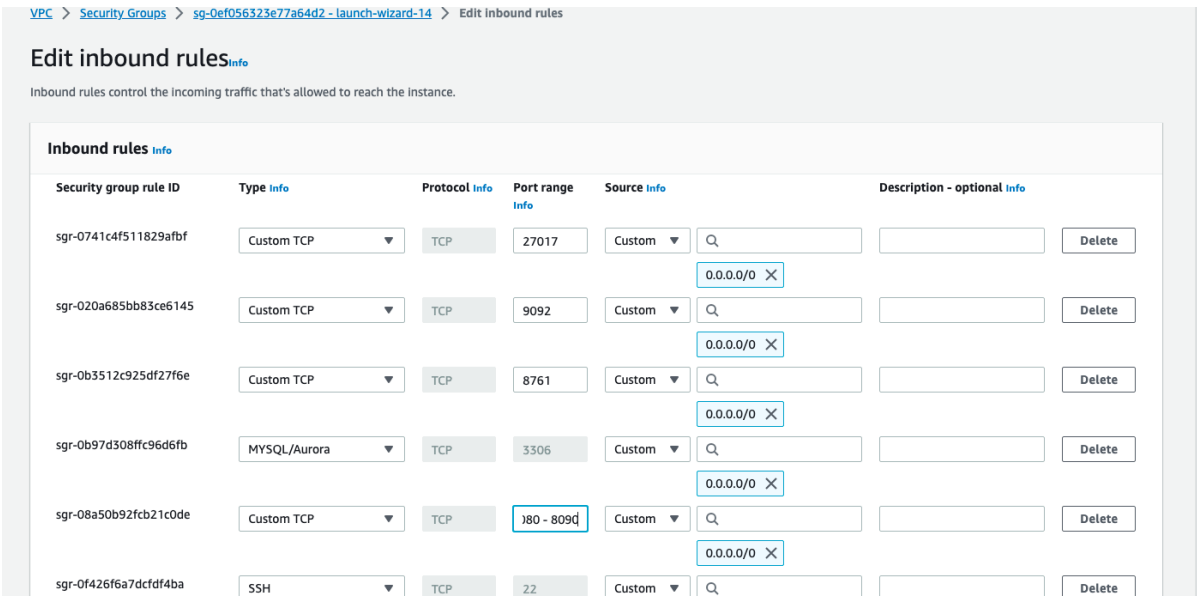
- create an `IAM` user with full access to `SES` and `S3`
- create a `Access Key` for that user and save the `Access Key` and `Password`
- This is required to be passed to the application while running



open ports for external access to access the apis

→ ports `8761` for eureka service to run

→ ports `8080` to `8090` for applications to expose the services



Running an application

steps to run the application:

1. upload the `BookMyConsultation.zip` to the `docker instance`
2. unzip the file and navigate to the file
3. before running need to give parameter to the application to start

   Only `docker-compose.yml` file need to be changed

   1. `mongo host` and `port` changes

      → replace `MONGO_HOST` value with `mongo running instance id`

      → replace `MONGO_PORT` value with port where mongo is running in mongo host

      → replace `MONGO_DATABASE` with mongo db name

   2. `MYSQL RDS` changes.

      → replace `MYSQL_HOST` value with `rds public IP`

      → replace `MYSQL_HOST_PORT` value with mysql port set by you

      → replace `MYSQL_USER` value with admin user details set for rds

      → replace `MYSQL_PASSWORD` value with admin user password set for rds

      → replace `MySQL_DATABASE` value with database name

   3. `KAFKA` changes

      → replace `KAFKA_HOST` value with kafka instance ip where kafka can be accessed from.

      → replace `KAFKA_HOST_PORT` value with kafka port value set by you

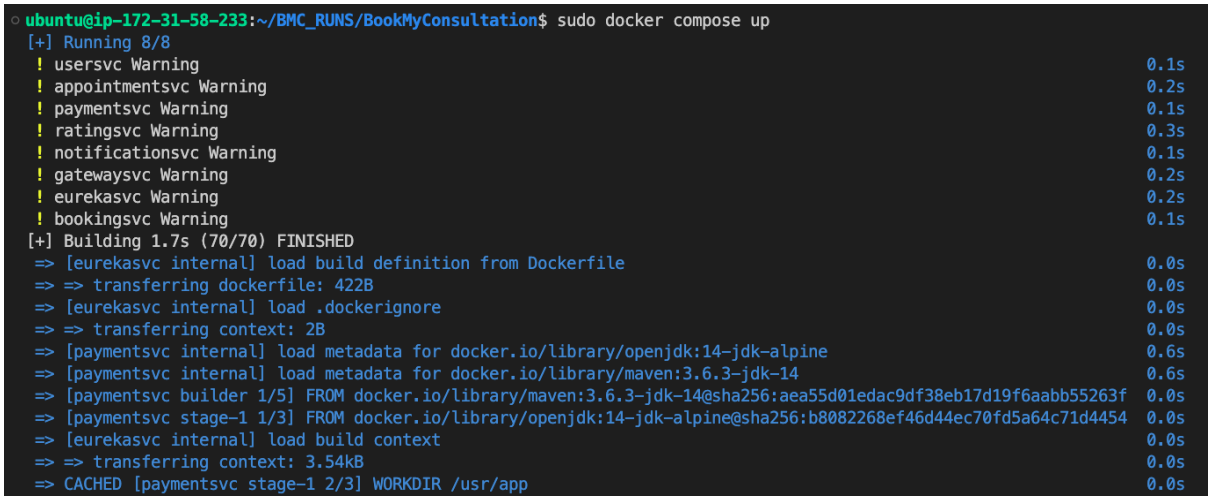
   4. `SES` and `S3` `access key` and `secret key`

      → replace values of `SES_ACCESSKEY` and `S3_ACCESS_KEY` with the token access token name generated

→ replace values of `SES_SECRETKEY` and `S3_SECRET_KEY` with the password generated for the access token

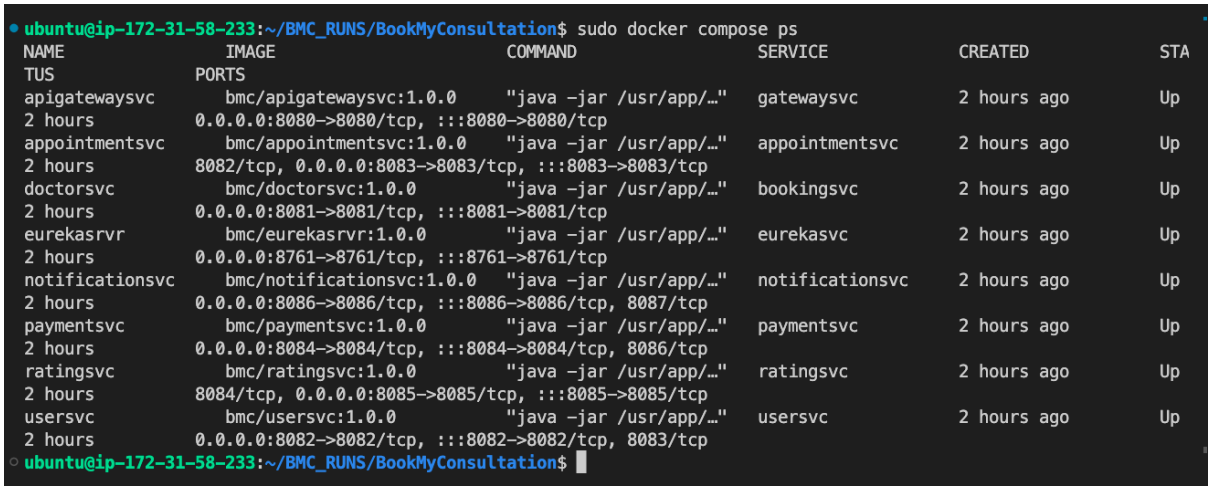
→ replace the `SES_FROM_EMAIL` with the email from which emails need to be sent this should be verified email.

4. After properly replacing all the values we are ready to run the application.
5. execute command `sudo docker compose up`
6. it will references the `Dockerfile` present in each services home folder and build the image one-by one and start the containers.



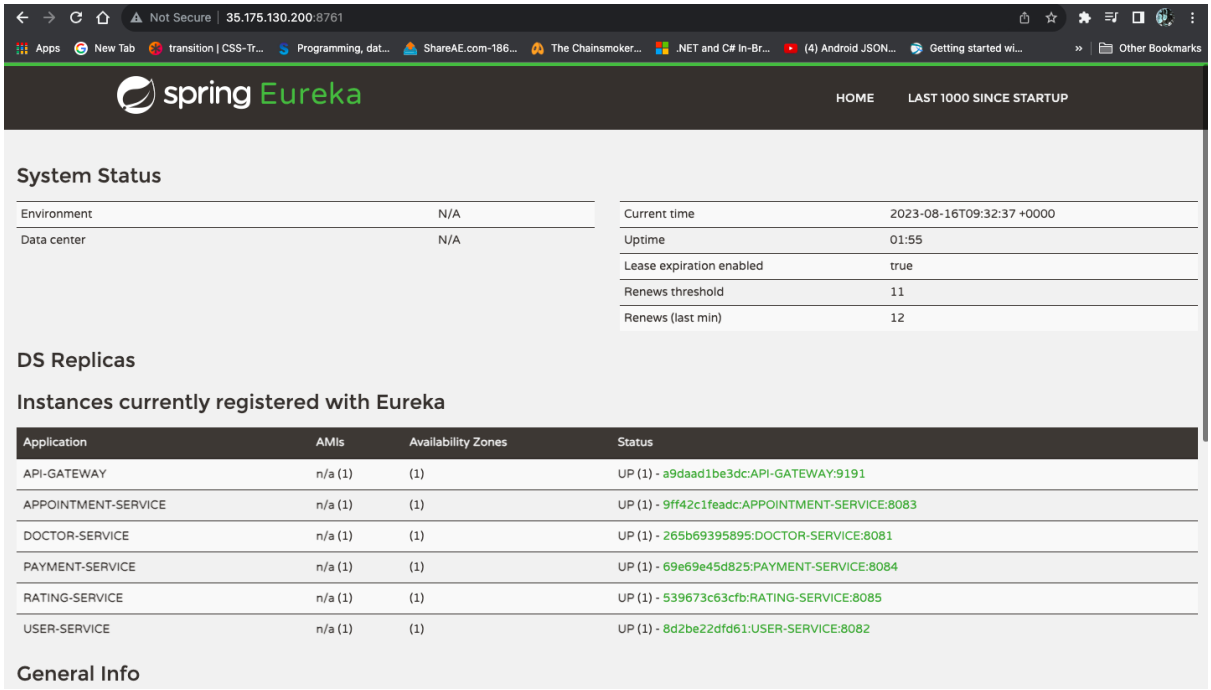
you can see applications coming up one by one

once all applications are up we can see the containers running as shown below



now we can see the services in `eureka service discovery`

visit eureka by following url `http://<docker-ec2-public-IP>:8761` replace `docker-ec2-public-IP` with proper ip address of where you are running docker compose



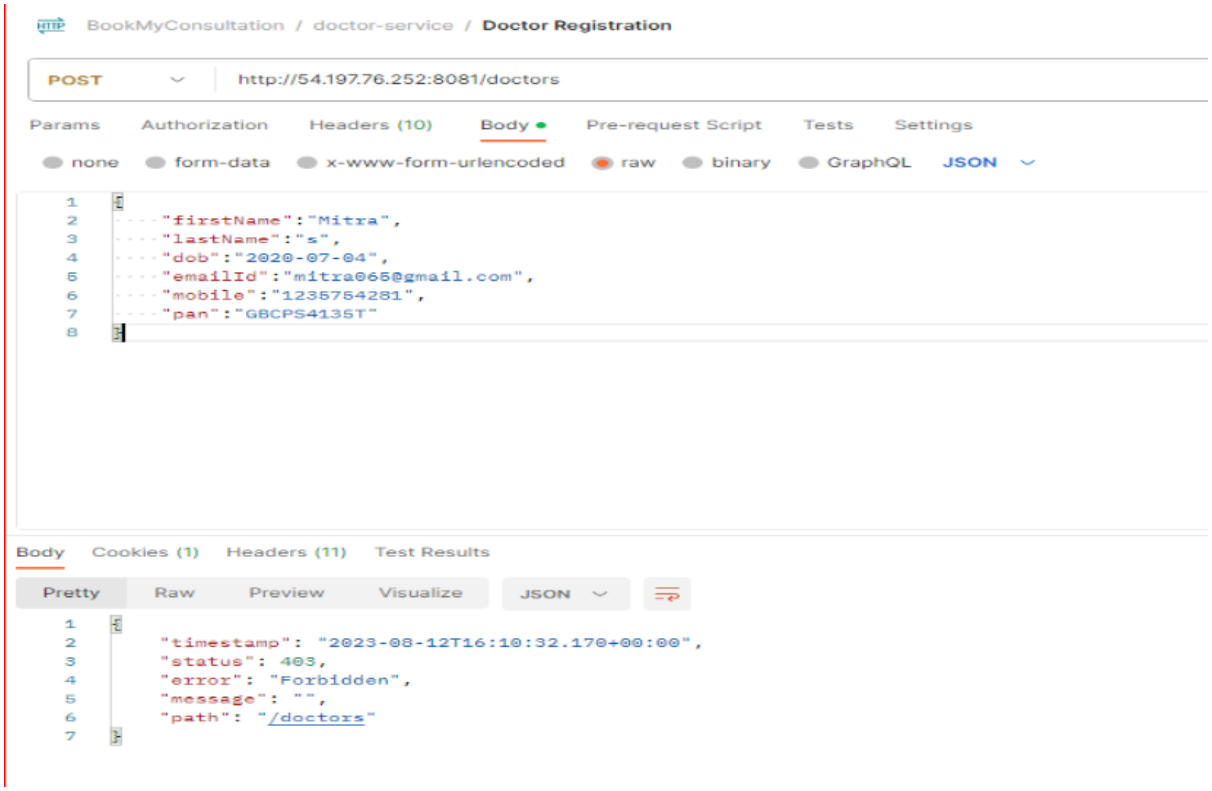
to bring down the services `sudo docker compose down`

API test screen shots

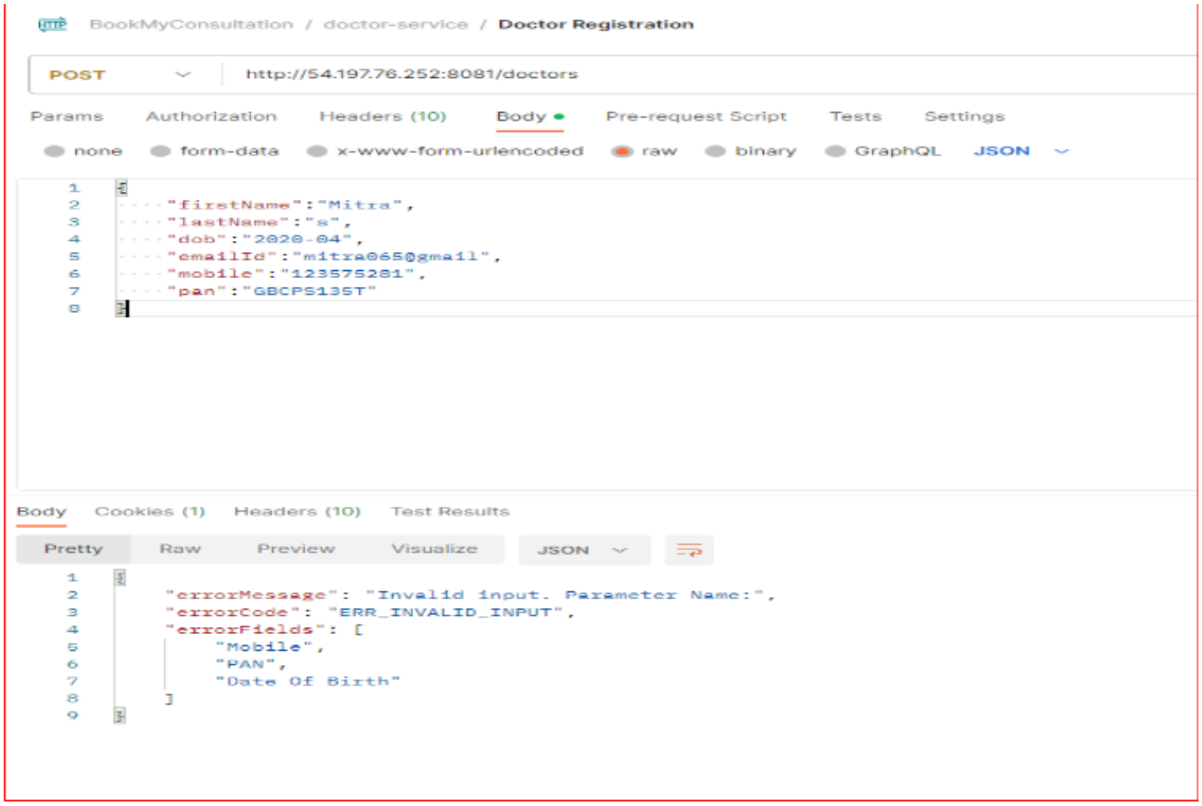
1. Doctor-Onboarding Service

Endpoint 1: URI: `/doctors`
This endpoint is responsible for collecting information about the doctor.

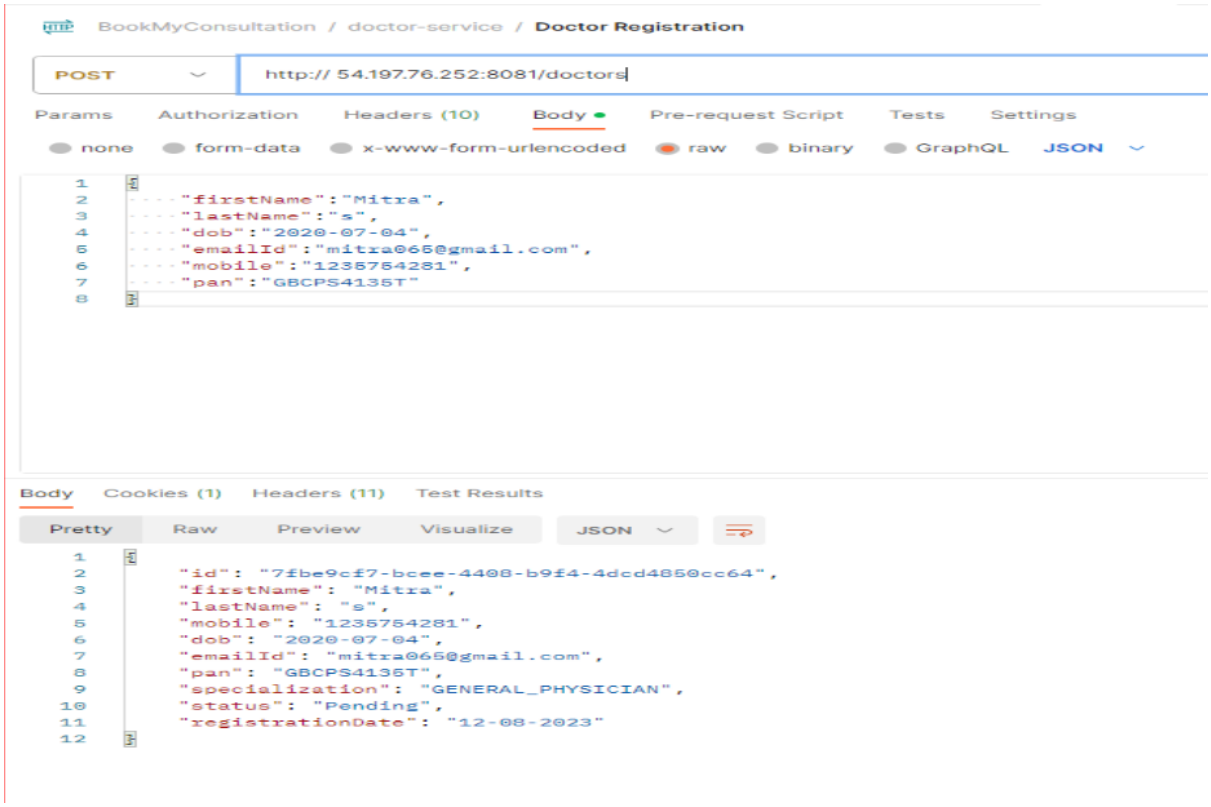
- Endpoint is forbidden, for roles other than User, Admin



- Endpoint is forbidden, for roles other than User, Admin



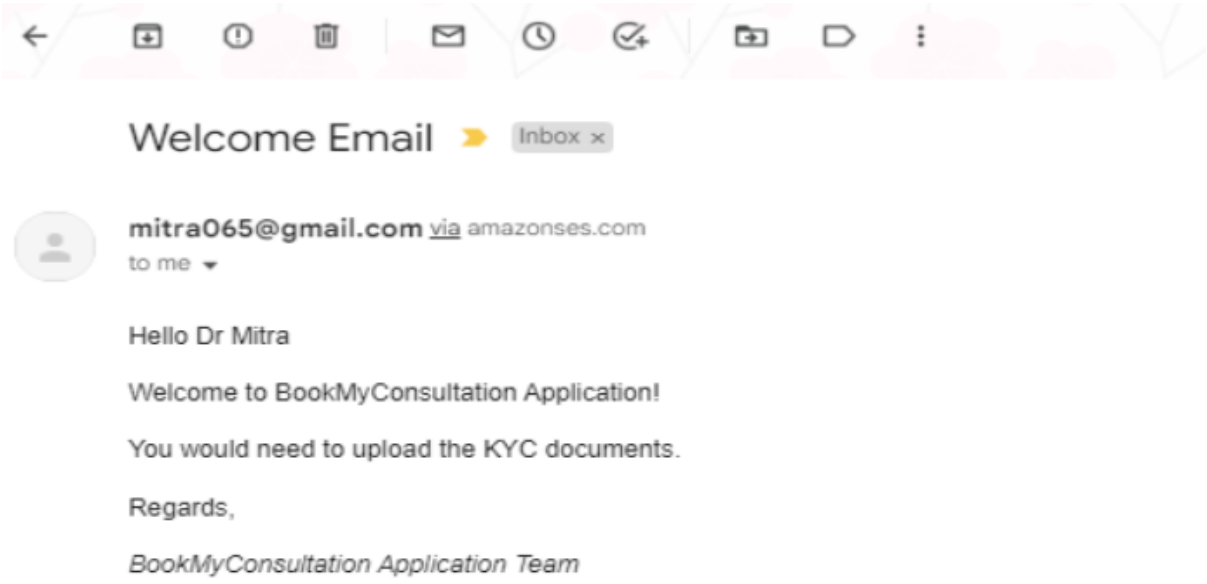
- Success Scenario



Kafka consumer logs


```
{
  "id" : "7fbe9cf7-bcee-4408-b9f4-4dcd4850cc64",
  "firstName" : "Mitra",
  "lastName" : "s",
  "mobile" : "1235754281",
  "dob" : "2020-07-04",
  "emailId" : "mitra065@gmail.com",
  "pan" : "GBCPS4135T",
  "specialization" : "GENERAL_PHYSICIAN",
  "status" : "Pending",
  "registrationDate" : "12-08-2023",
  "approvedBy" : "Pending",
  "approverComments" : "Pending",
  "verificationDate" : "Pending",
  "rating" : 0
}
```

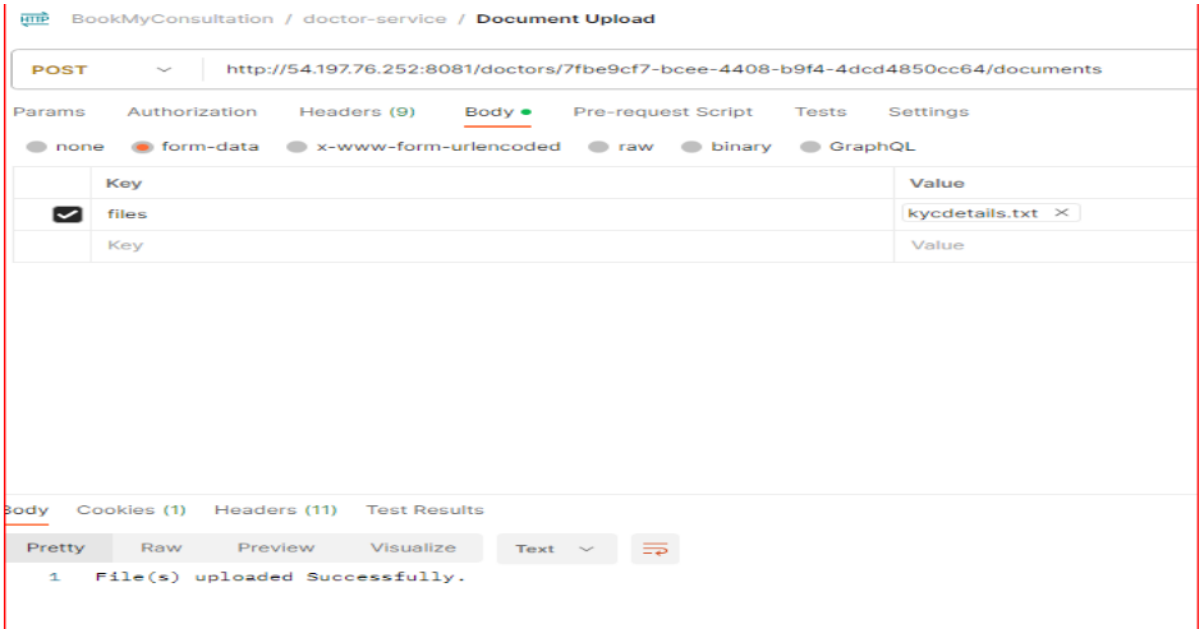
- Verification mail Sent to Doctor



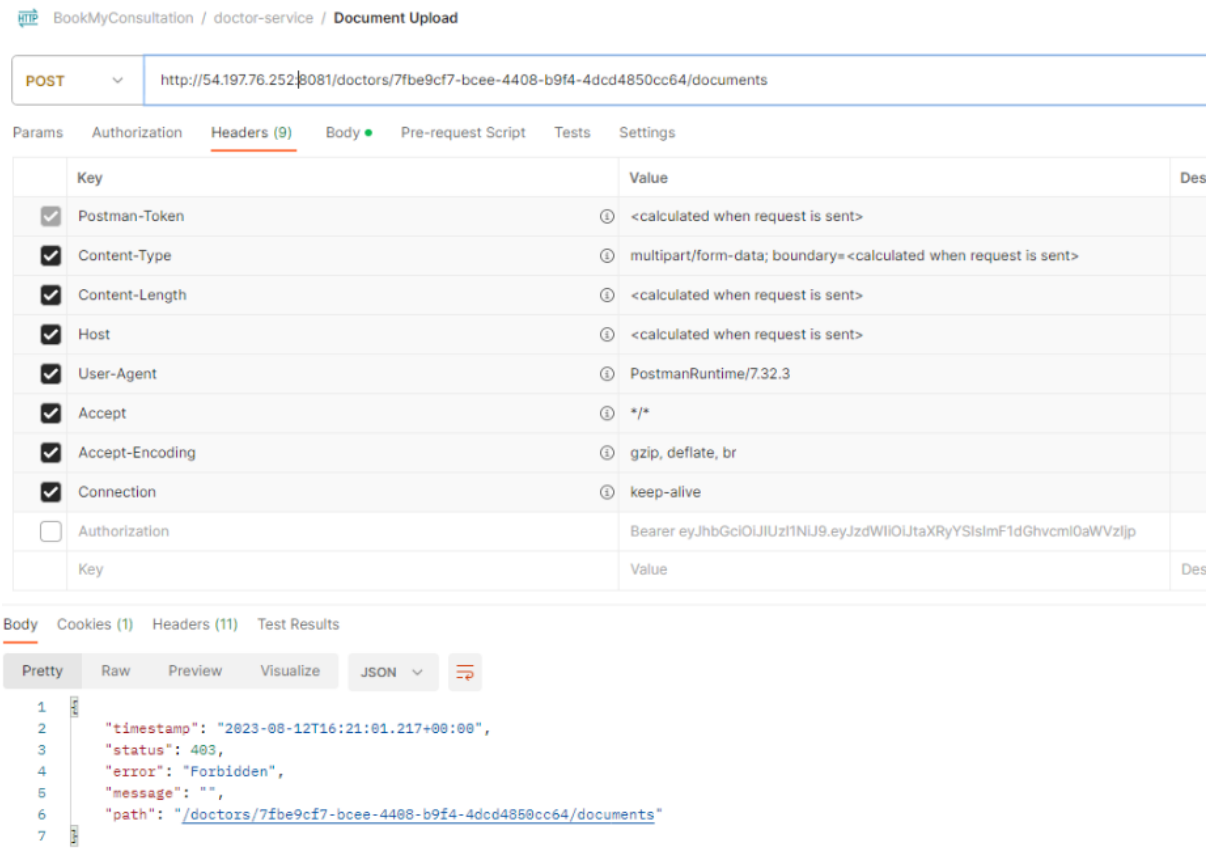
Endpoint 2: URI: `/doctors/{doctorId}/document`

This endpoint is responsible for uploading the documents to an S3 bucket by the doctor.

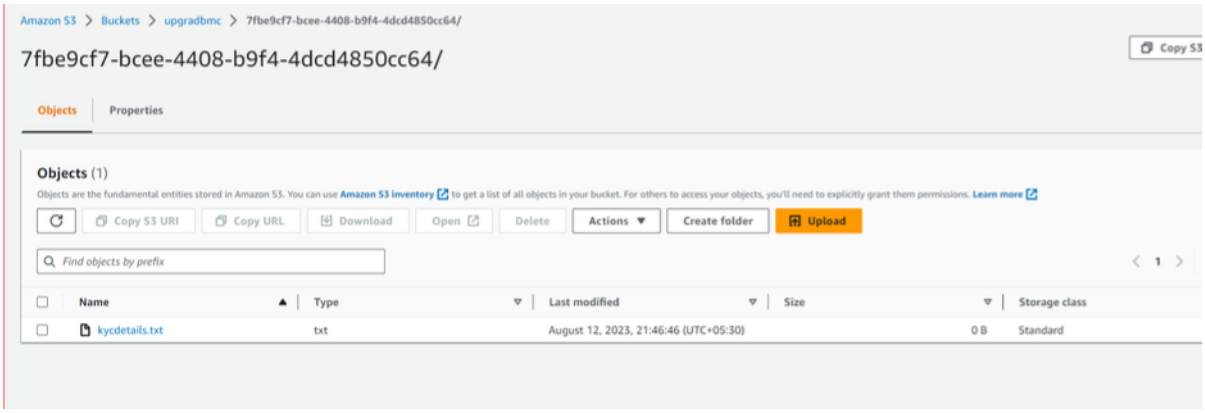
- Success Scenario – Document Uploaded Successfully



- Endpoint Forbidden with authorisation header

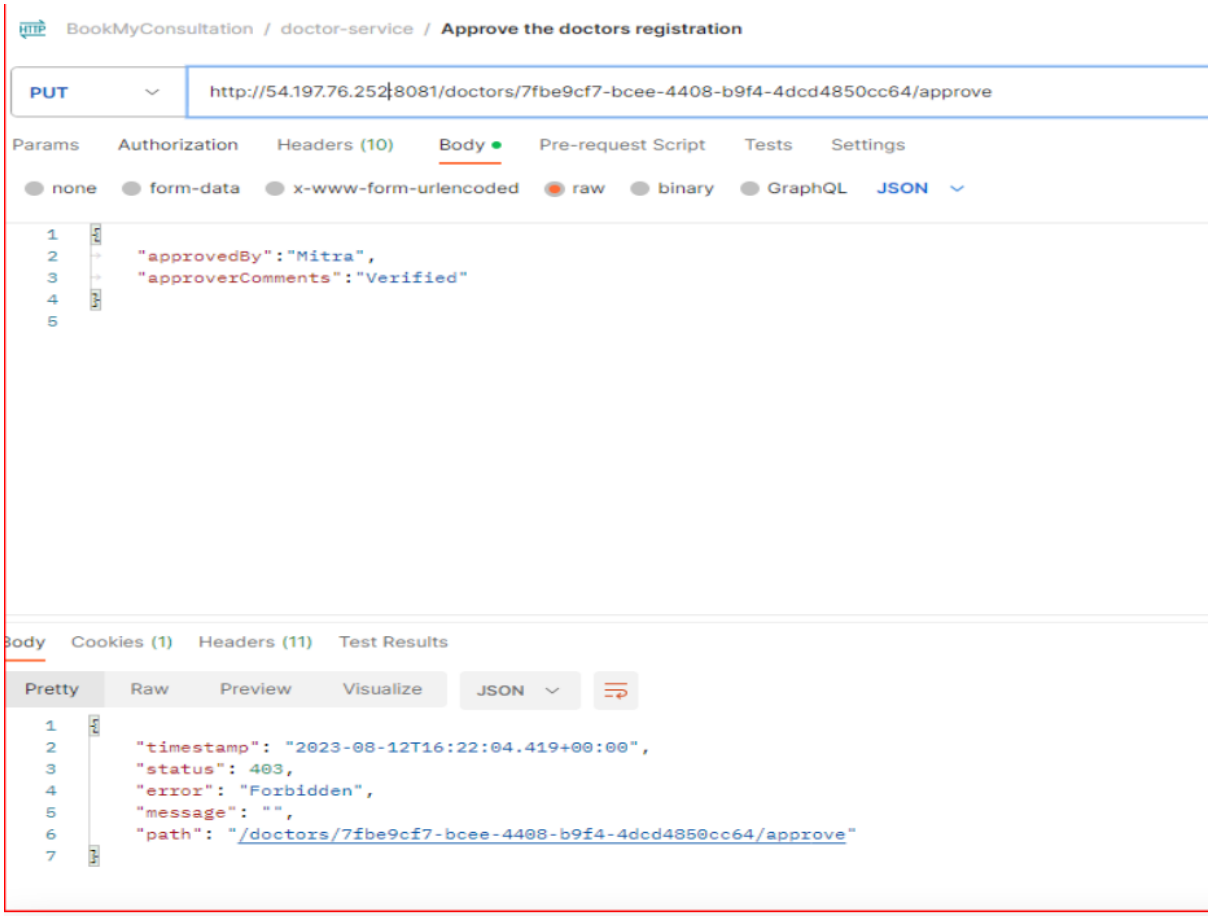


- Files Uploaded in S3 bucket

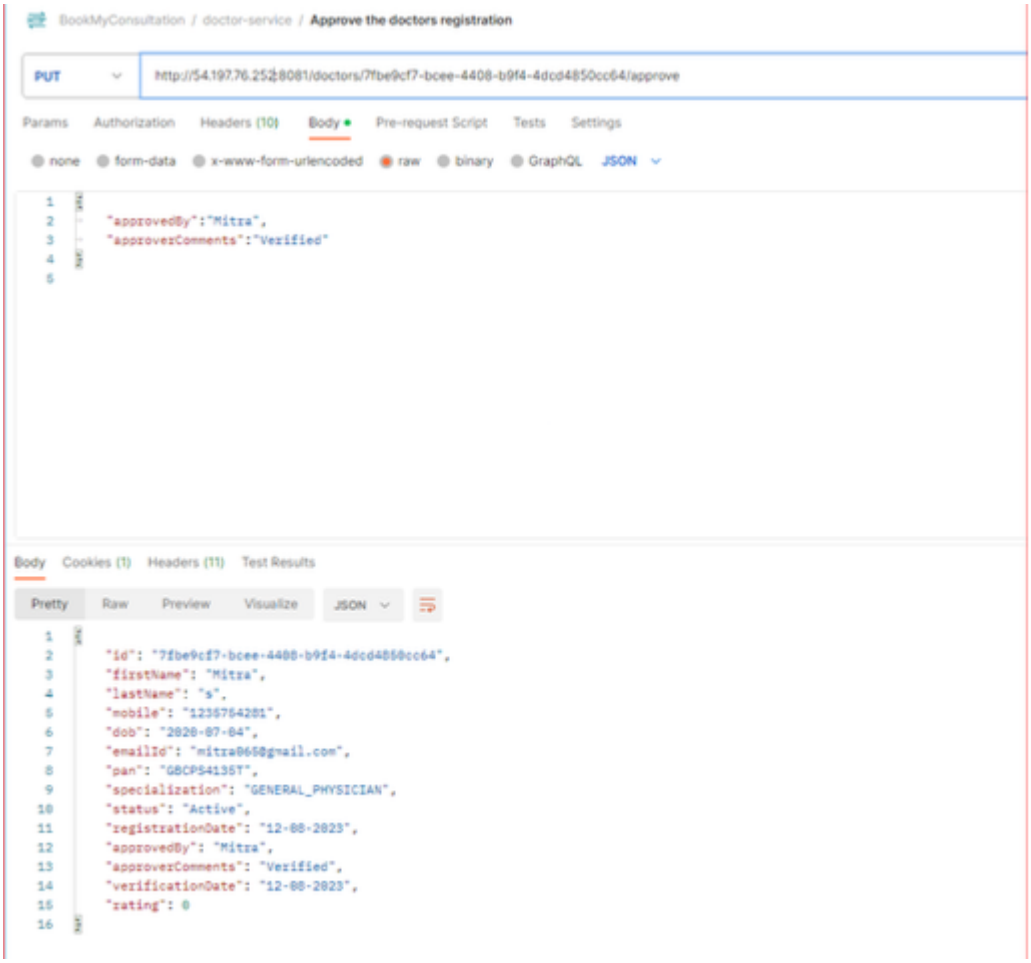


Endpoint 3: URI: `/doctors/{doctorId}/approve`

Endpoint Forbidden for unauthorised users



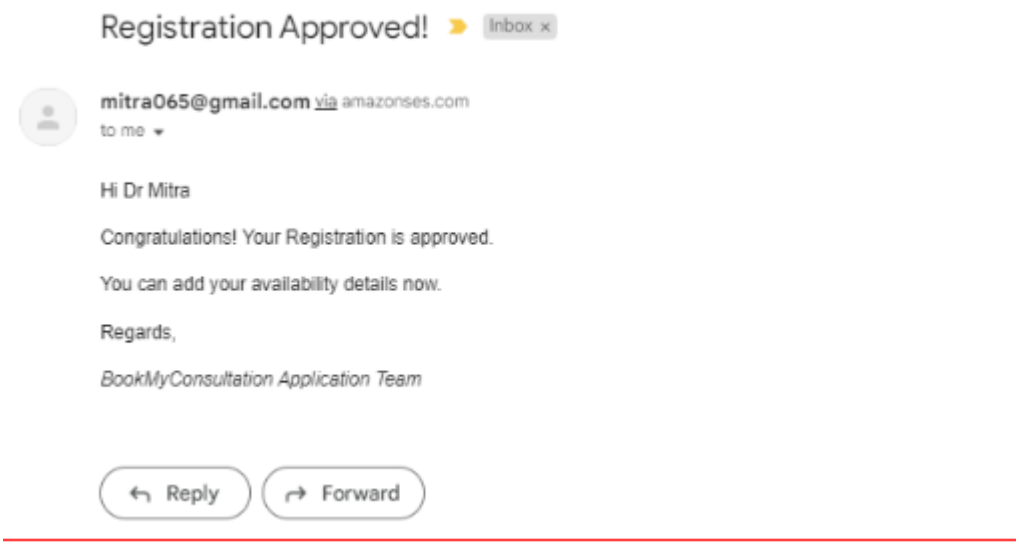
- Successful Scenario – Approve Request



- Kafka Consumer Logs

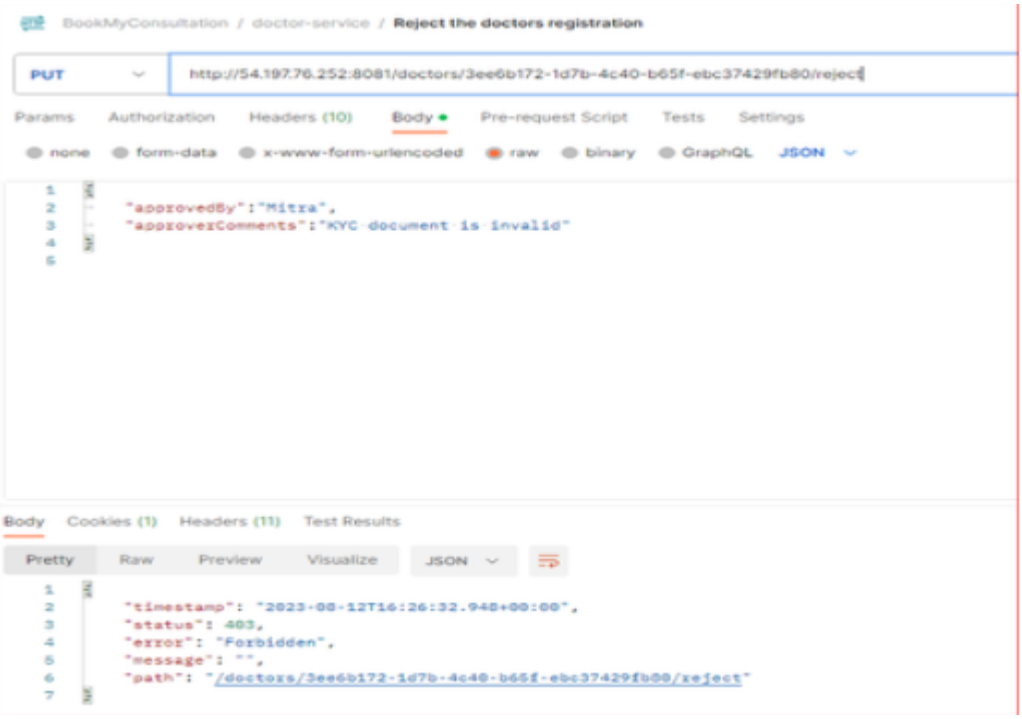

```
{
  "id" : "7fbe9cf7-bcee-4408-b9f4-4dcd4850cc64",
  "firstName" : "Mitra",
  "lastName" : "s",
  "mobile" : "1235754281",
  "dob" : "2020-07-04",
  "emailId" : "mitra065@gmail.com",
  "pan" : "GBCPS4135T",
  "specialization" : "GENERAL_PHYSICIAN",
  "status" : "Active",
  "registrationDate" : "12-08-2023",
  "approvedBy" : "Mitra",
  "approverComments" : "Verified",
  "verificationDate" : "12-08-2023",
  "rating" : 0
}
```

- Registration Approved Mail Screenshot

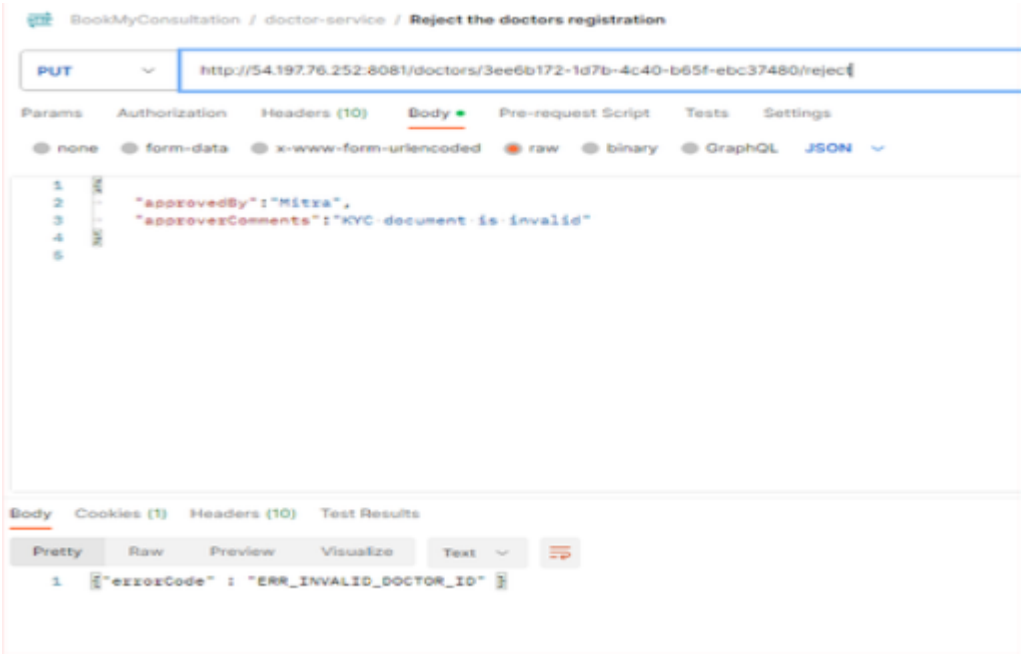


Endpoint 4: URI: `/doctors/{doctorId}/reject`
This endpoint is responsible for rejecting the registration of the doctor

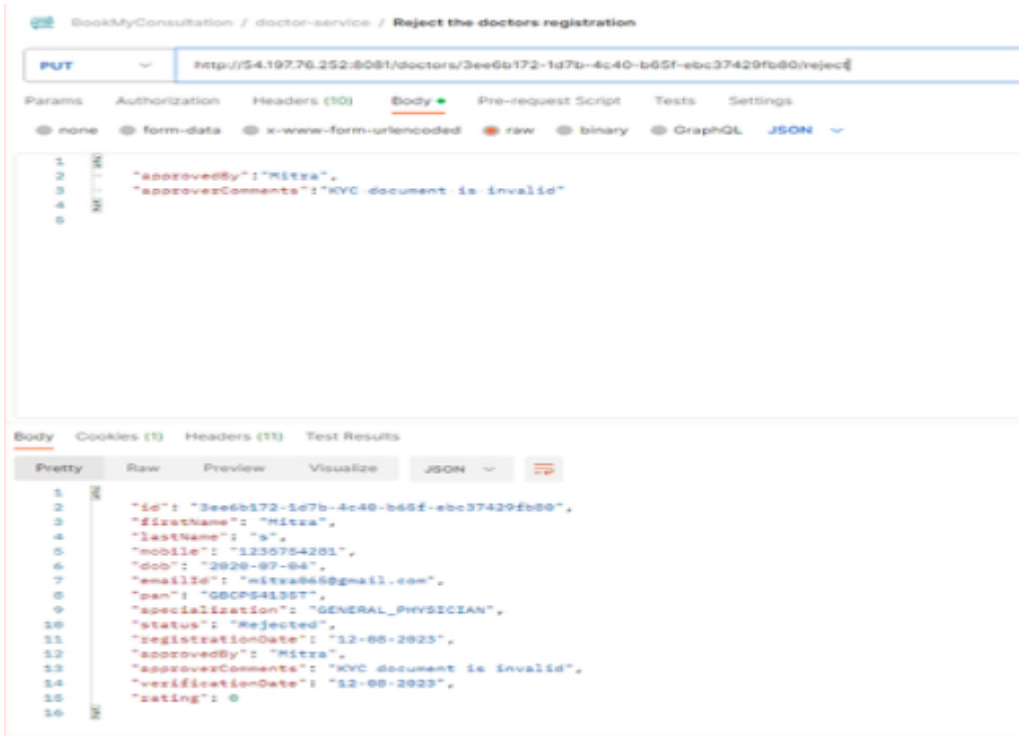
- Endpoint Forbidden for unauthorised users



- Validation Error when Invalid Doctor Id is passed



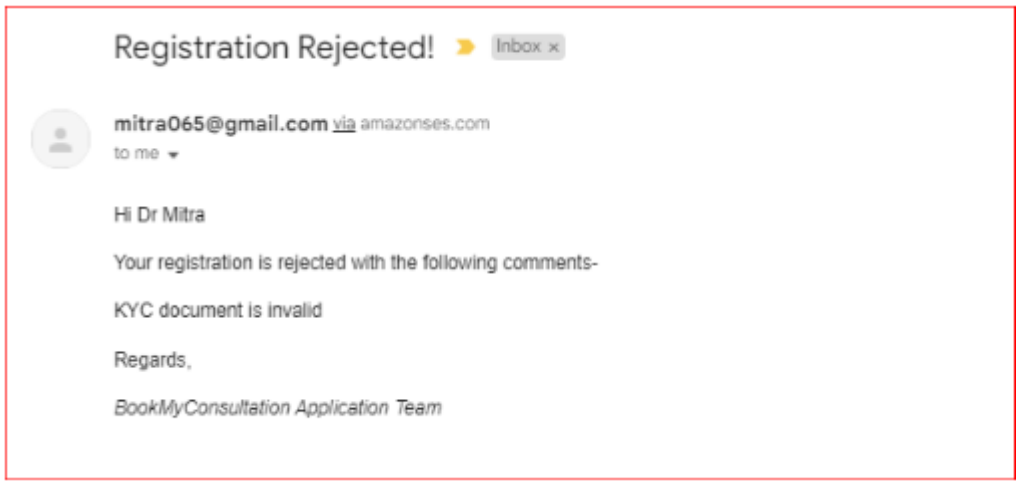
- Success Scenario - Rejection



- Kafka Consumer Logs

```
{
  "id" : "3ee6b172-1d7b-4c40-b65f-ebc37429fb80",
  "firstName" : "Mitra",
  "lastName" : "s",
  "mobile" : "1235754281",
  "dob" : "2020-07-04",
  "emailId" : "mitra065@gmail.com",
  "pan" : "GBCPS4135T",
  "specialization" : "GENERAL_PHYSICIAN",
  "status" : "Rejected",
  "registrationDate" : "12-08-2023",
  "approvedBy" : "Mitra",
  "approverComments" : "KYC document is invalid",
  "verificationDate" : "12-08-2023",
  "rating" : 0
}
```

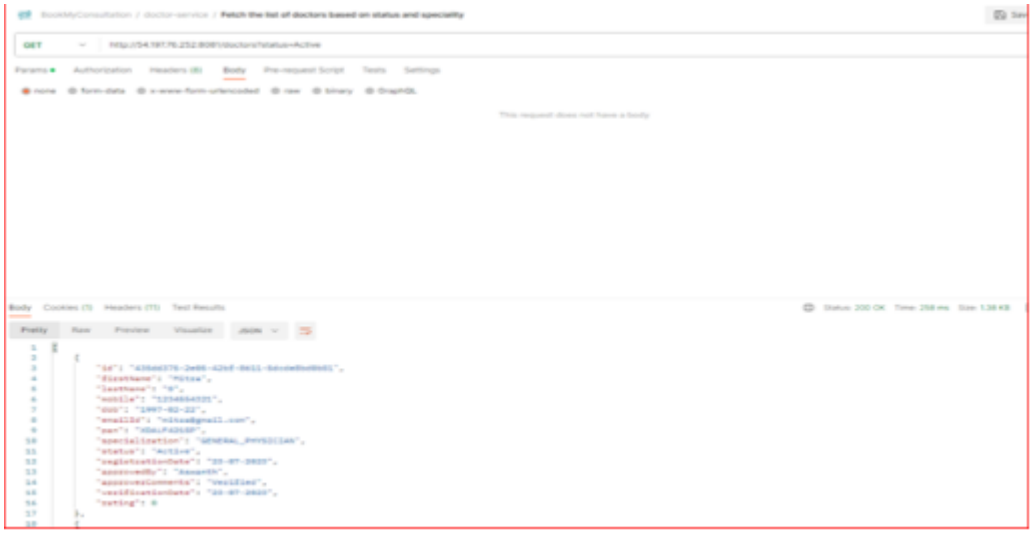

- Registration Rejection Mail Received Screenshot



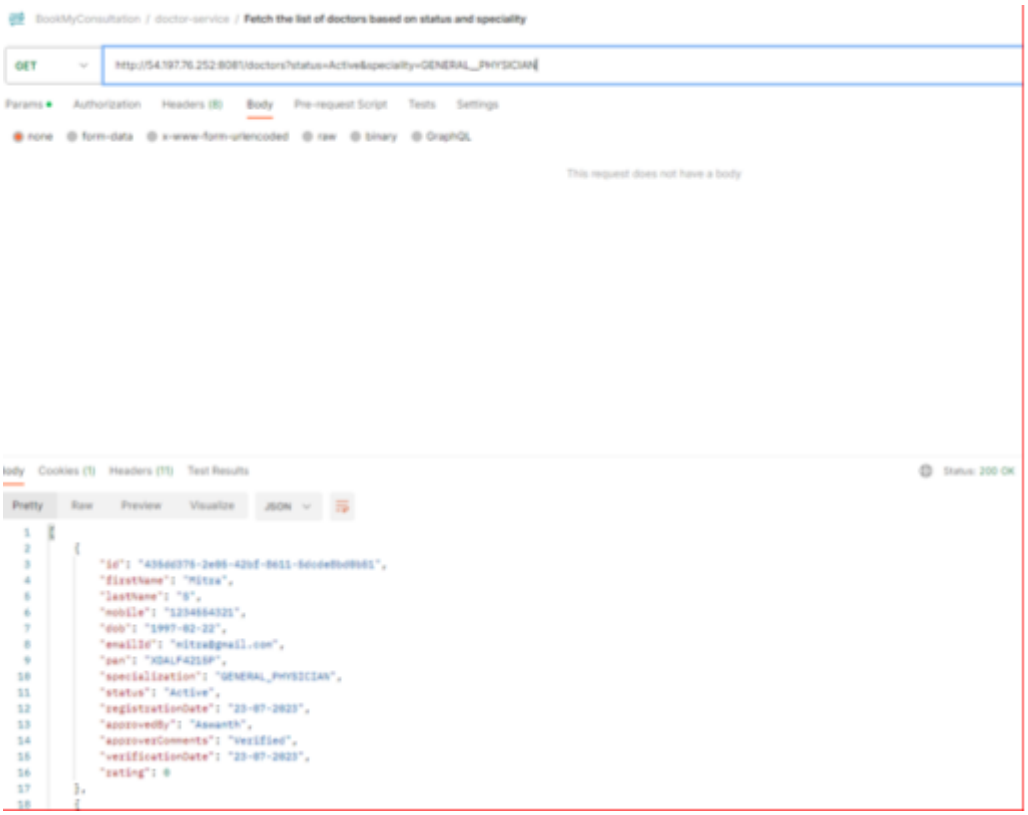
Endpoint 5: URI: `/doctors?<Required Status & Speciality Filter Condition>`

This endpoint is responsible for returning the list of 20 doctors sorted by ratings.

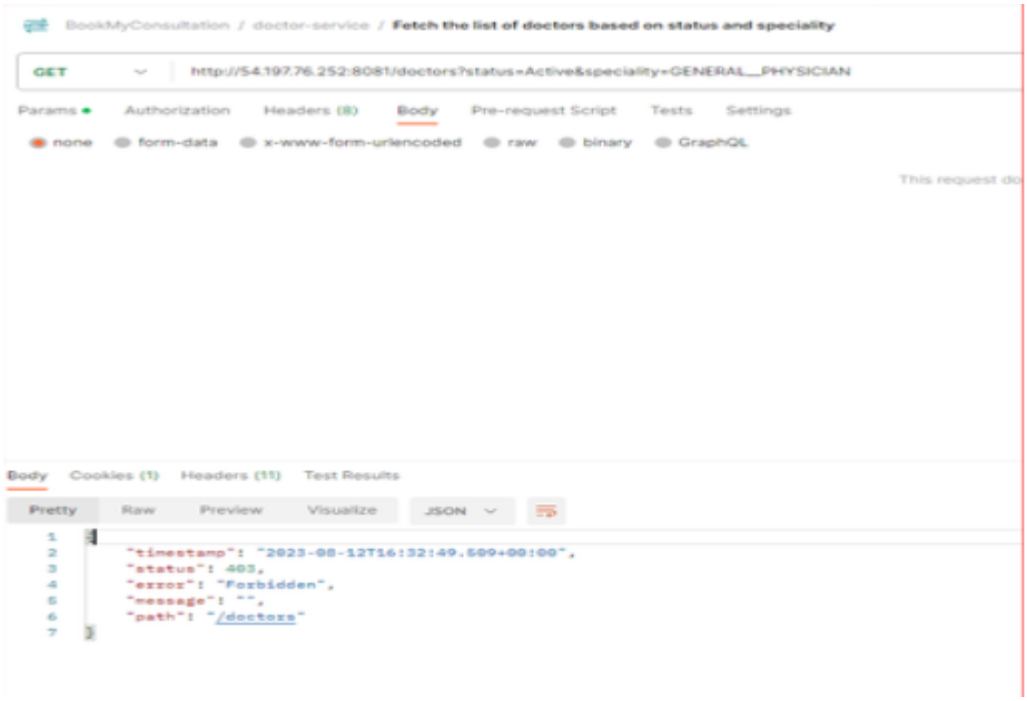
- With Status Filter



- With Status & Specialty Filter



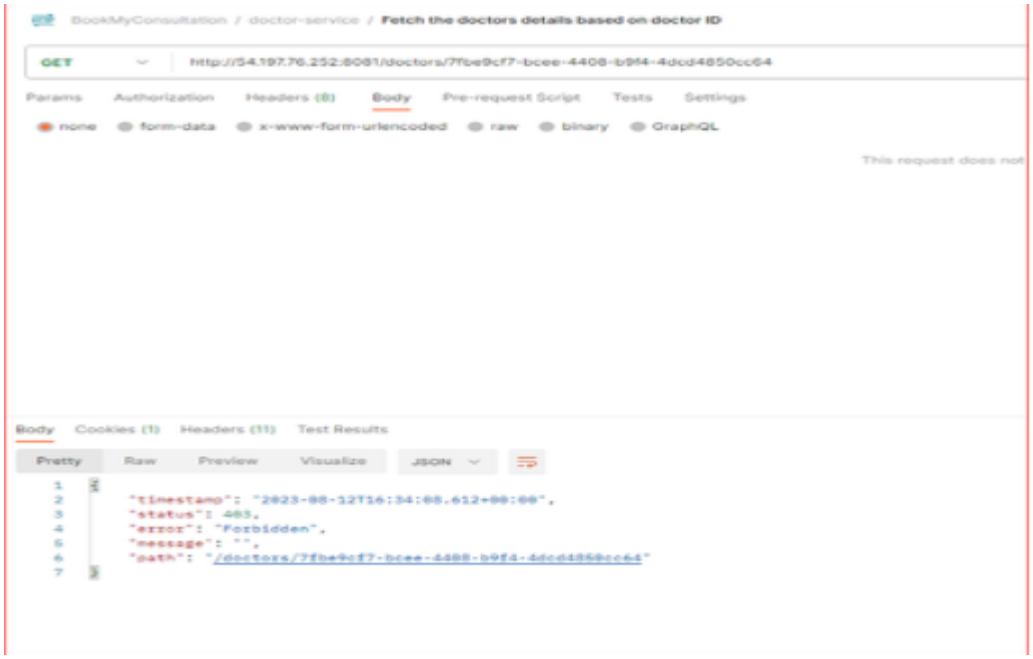
- Forbidden Error for Unauthorised error



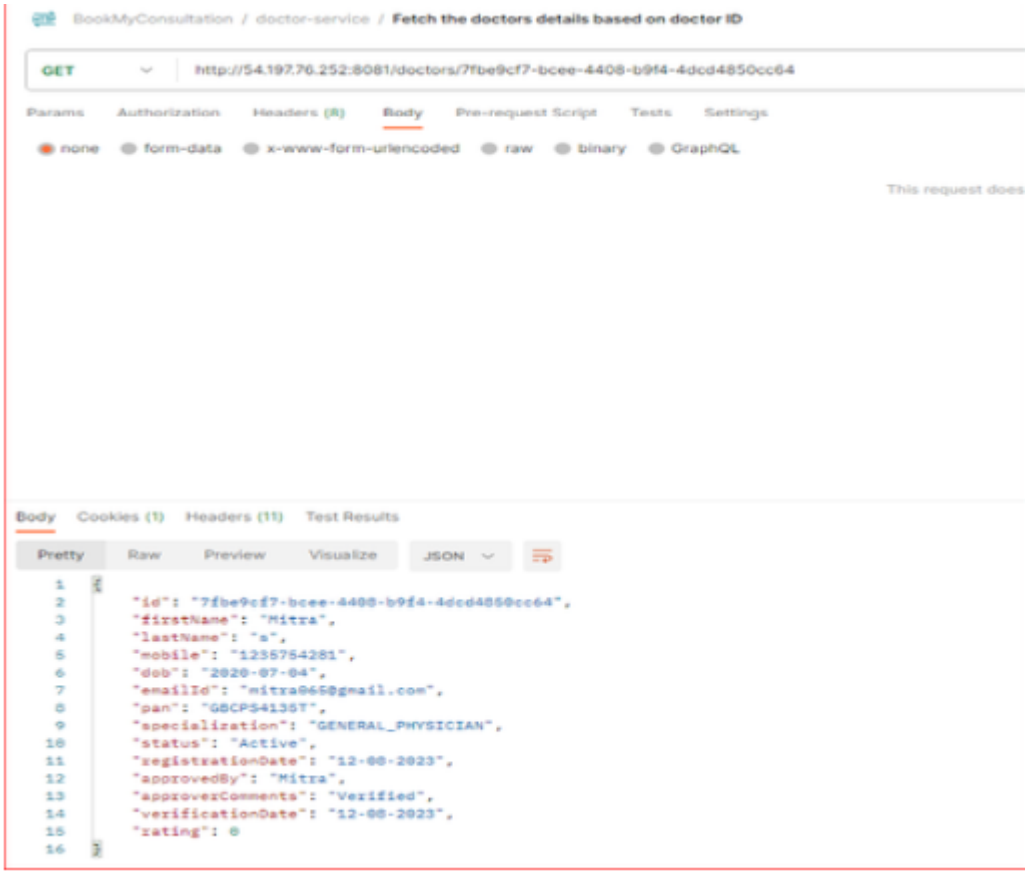
Endpoint 6: URI: `/doctors/{doctorId}`

This endpoint is responsible for returning the details of the doctor based on the doctor ID.

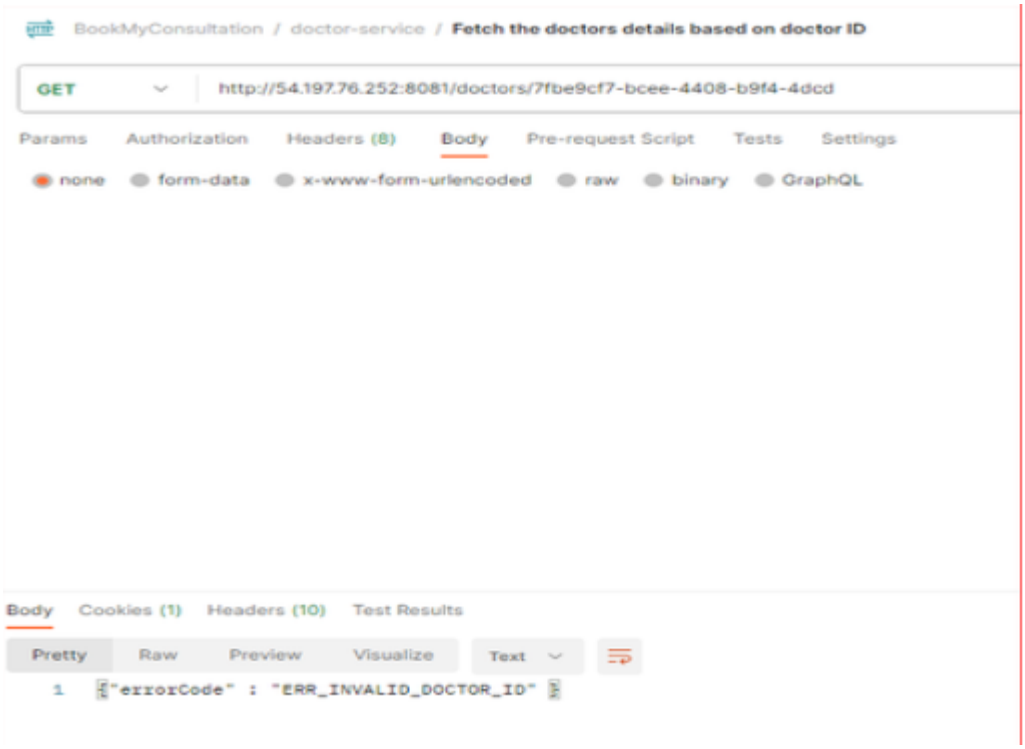
- Forbidden Error for Unauthorised error



- Success Scenario



- Validation Error – When Invalid Doctor Id passed

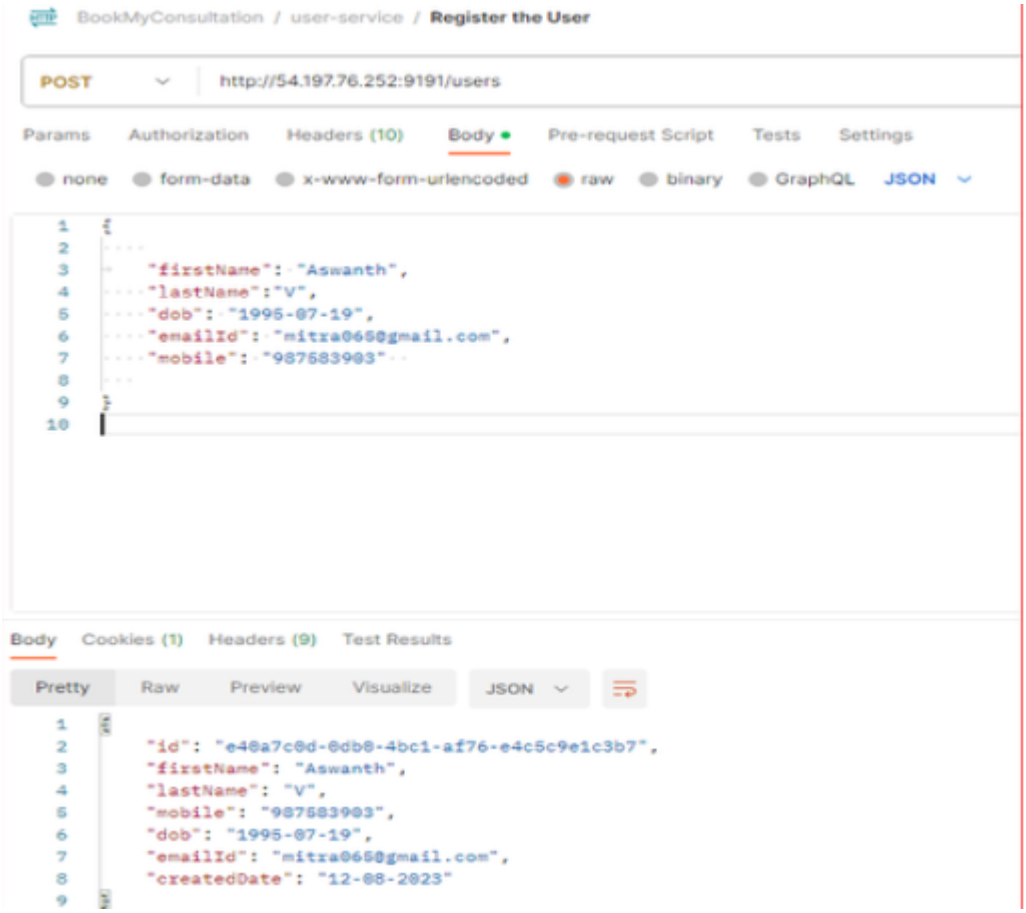


User -Onboarding Service

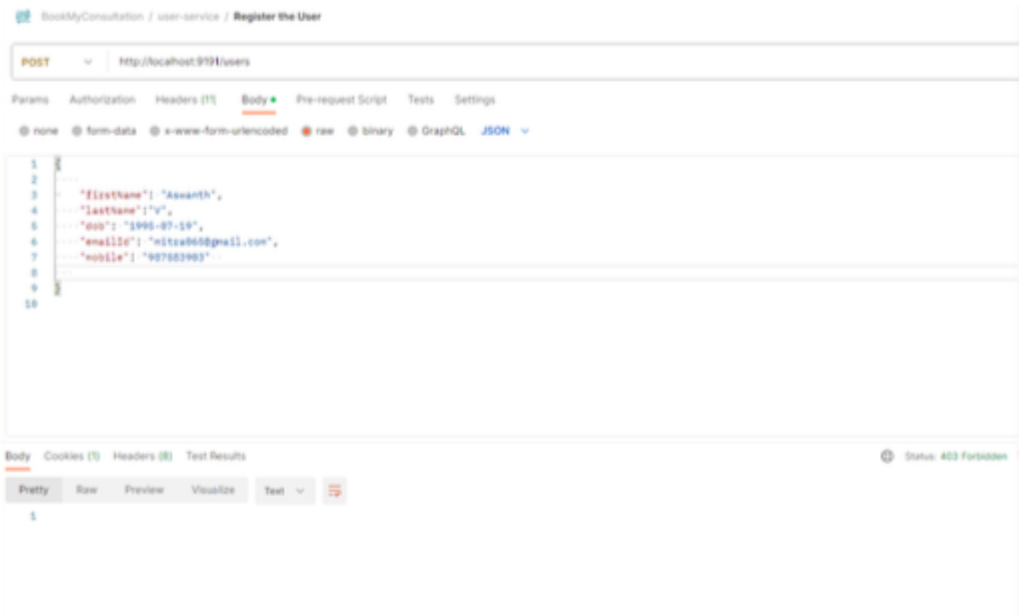
Endpoint 1: URI: `/users`

This endpoint is responsible for collecting information about the user.

- Successful Scenario



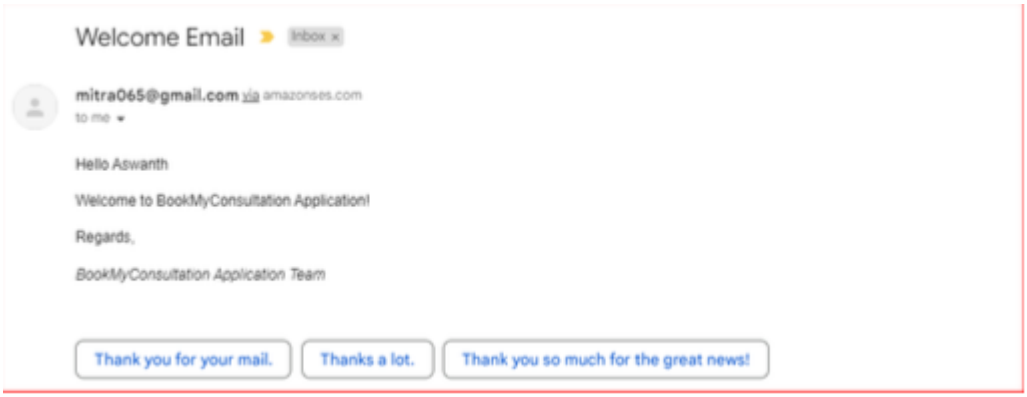
- Forbidden Error for unauthorised error



- Kafka Consumer logs

```
{
  "id" : "e40a7c0d-0db0-4bc1-af76-e4c5c9e1c3b7",
  "firstName" : "Aswanth",
  "lastName" : "V",
  "mobile" : "987583903",
  "dob" : "1995-07-19",
  "emailId" : "mitra065@gmail.com",
  "createdDate" : "12-08-2023"
}
```

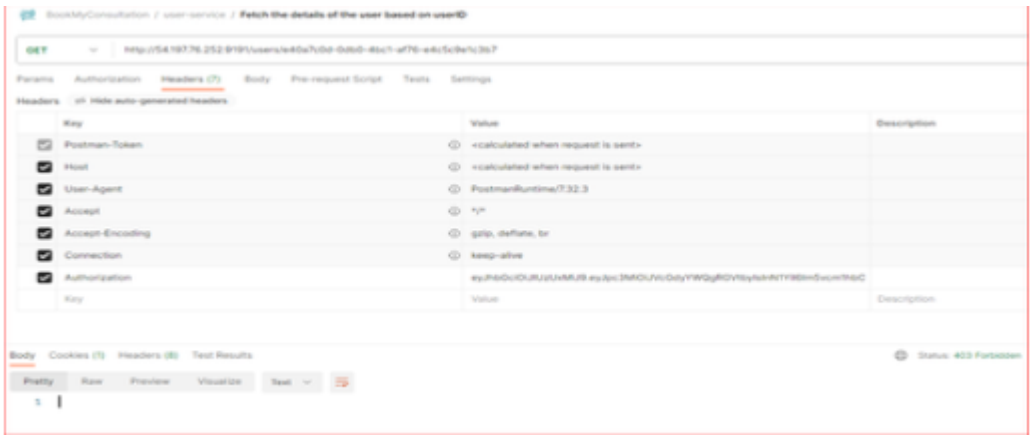

- Verification mail received after user registration



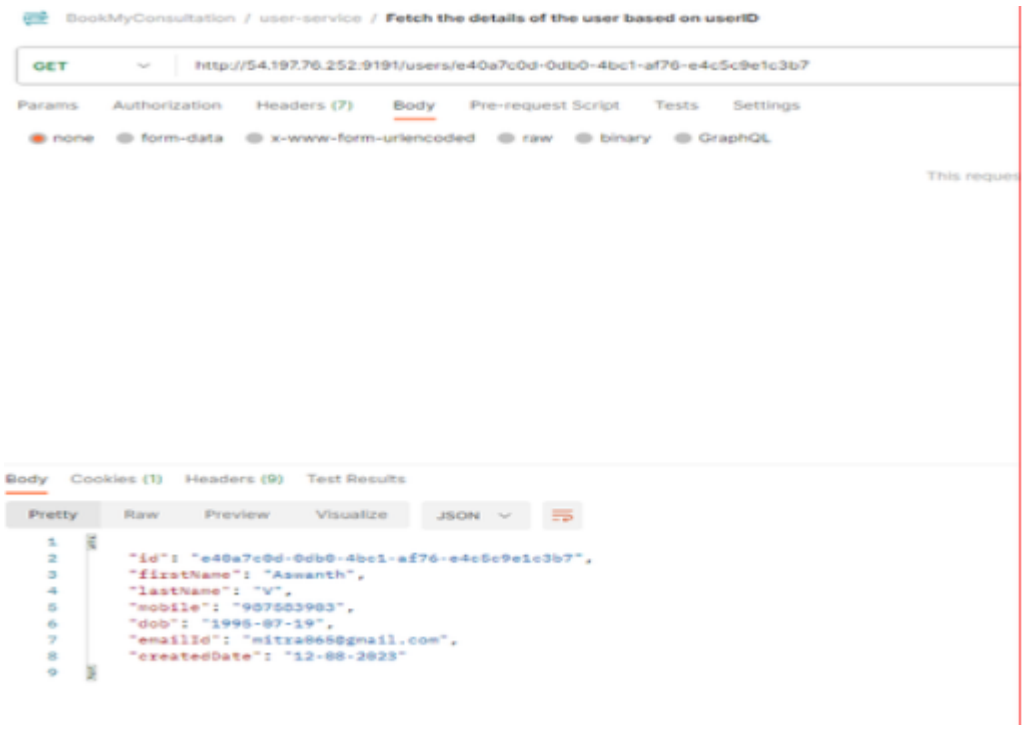
Endpoint 2: URI: `/users/{userID}`

This endpoint is responsible for collecting information about the user.

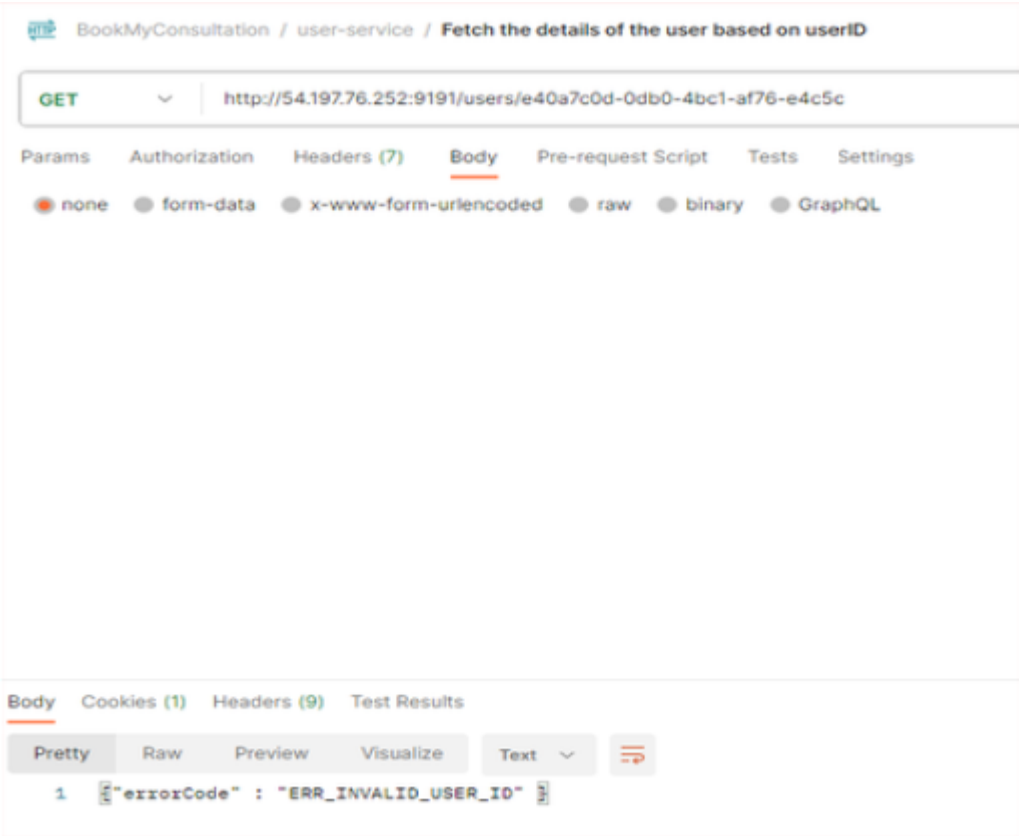
- Forbidden Error for unauthorised error



- Successful Scenario



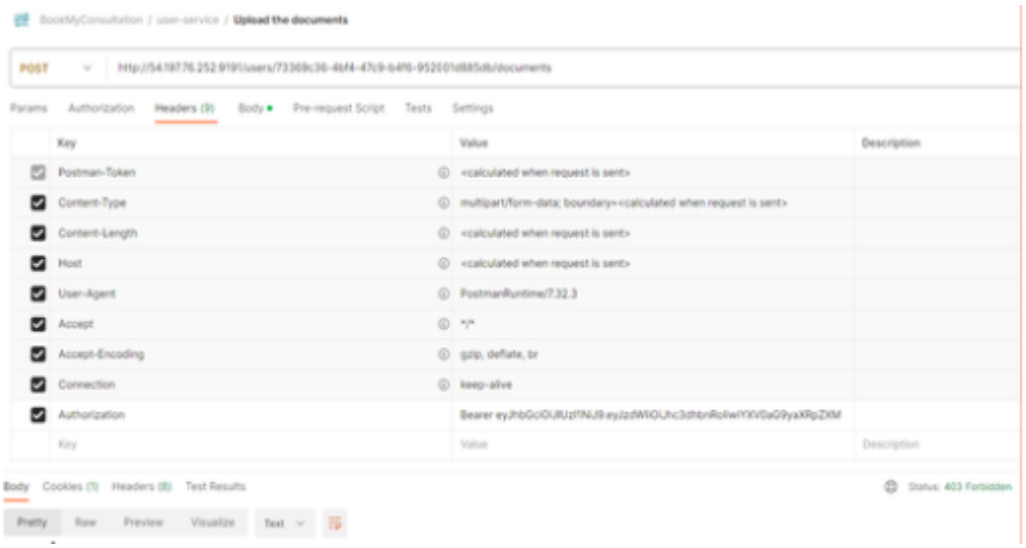
- Validation Error - Invalid User Id Passed



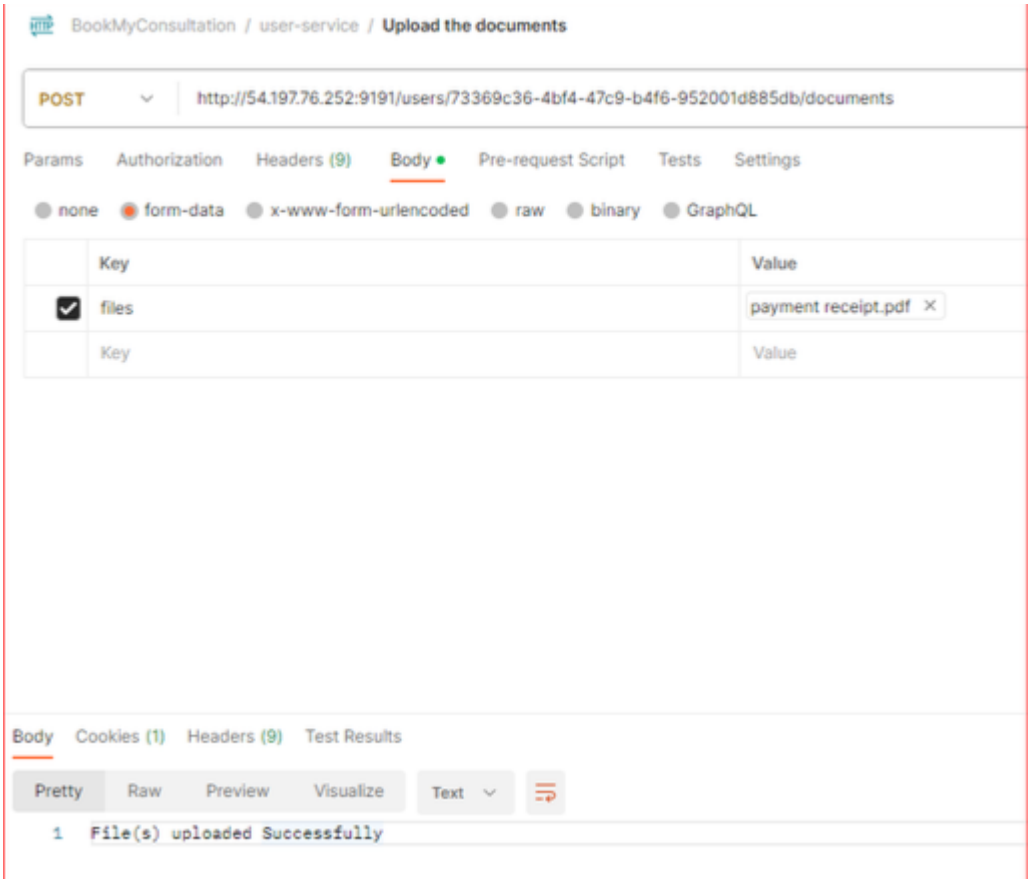
Endpoint 3: URI: `/users/{id}/documents.`

A POST request will be sent to this endpoint to upload the files.

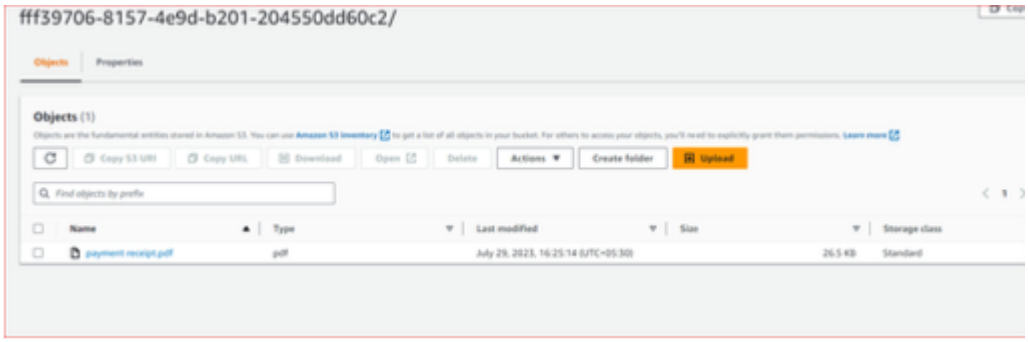
- Forbidden Error for unauthorised error



- File Uploaded Successfully



- File Uploaded in S3 bucket



Appointment Service

Endpoint 1: POST URI: `/doctor/{doctorId}/availability`

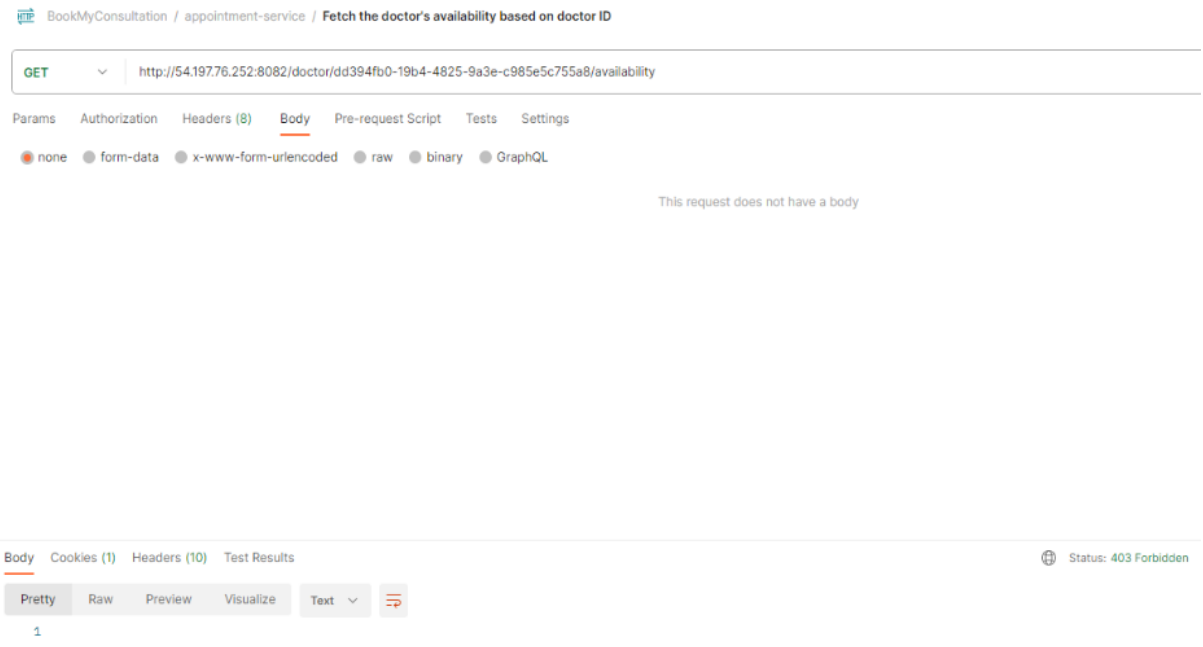
This endpoint is responsible for updating the availability of the doctors.

- Successful Scenario

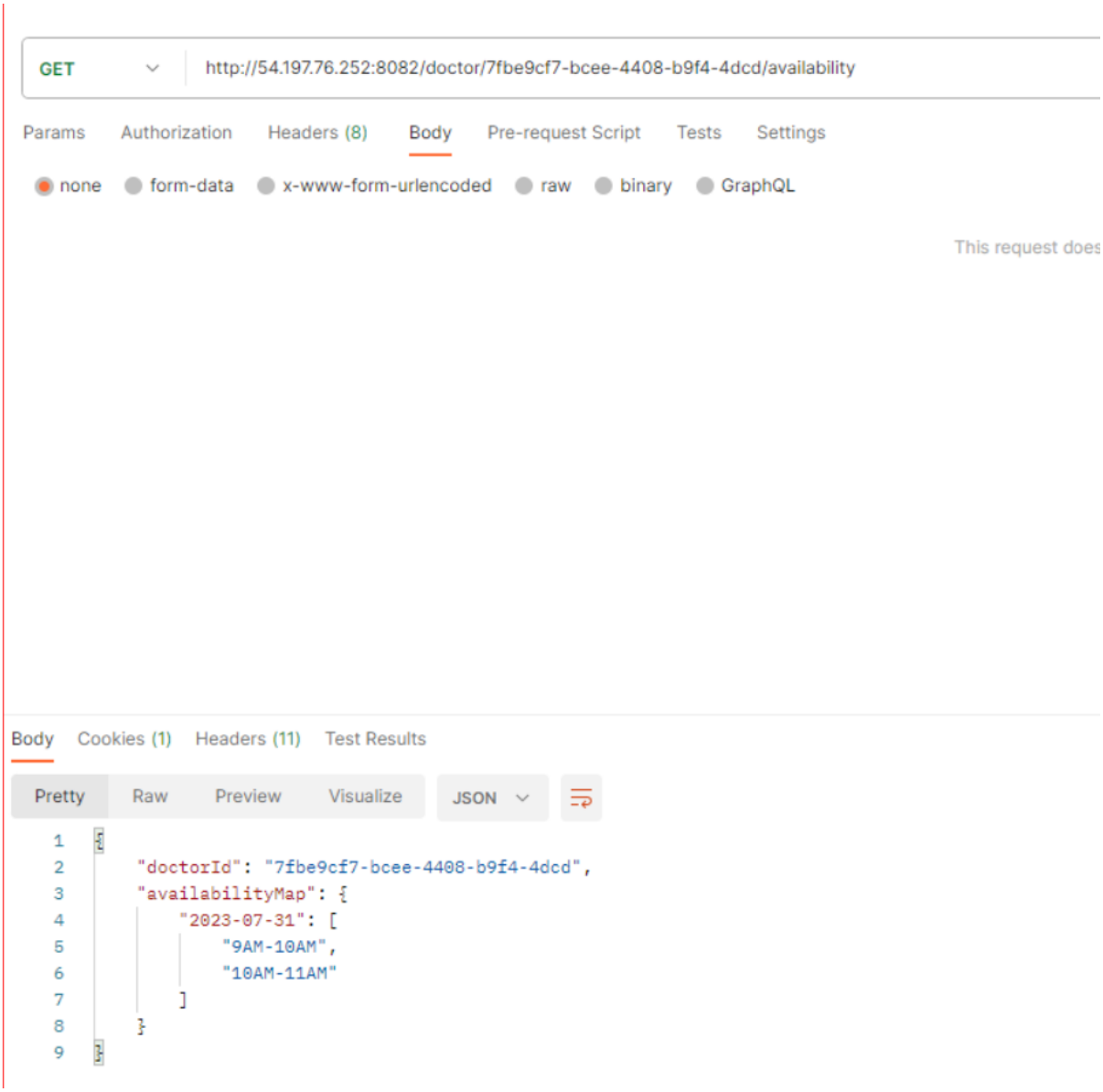
Endpoint 2: GET URI: `/doctor/{doctorId}/availability`

This endpoint is responsible for returning the availability of the doctors.

- Forbidden Error for Unauthorised Users



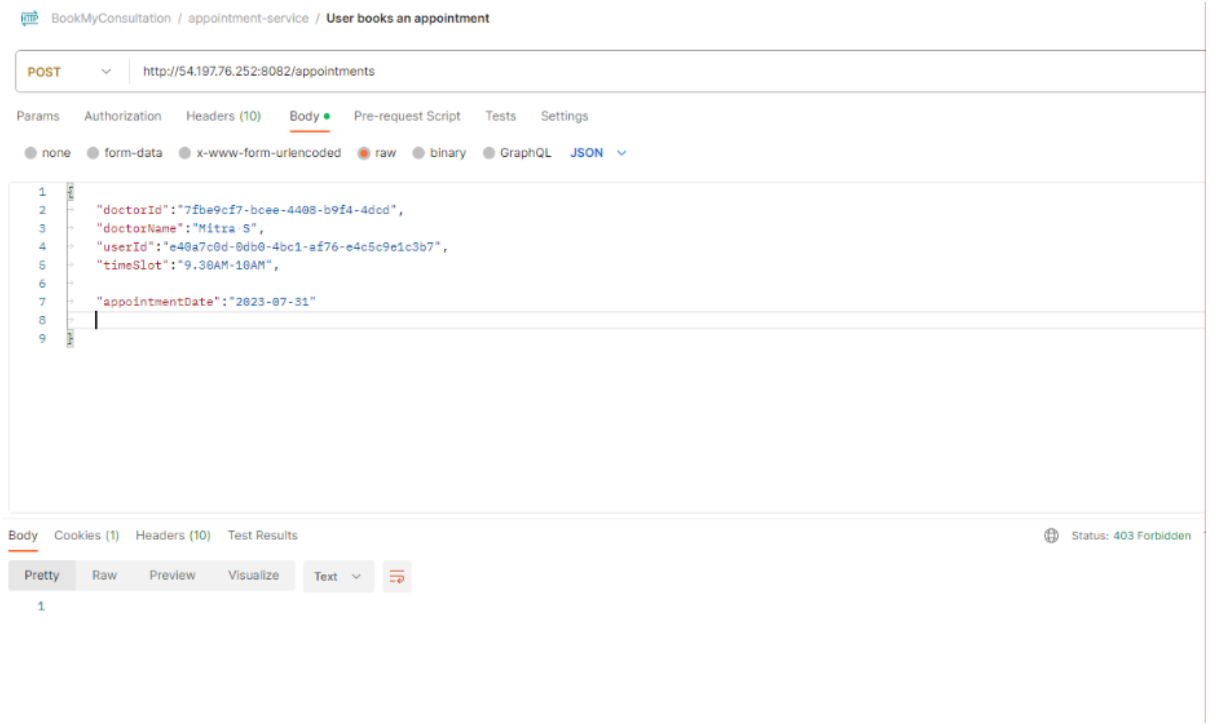
- Successful Scenario



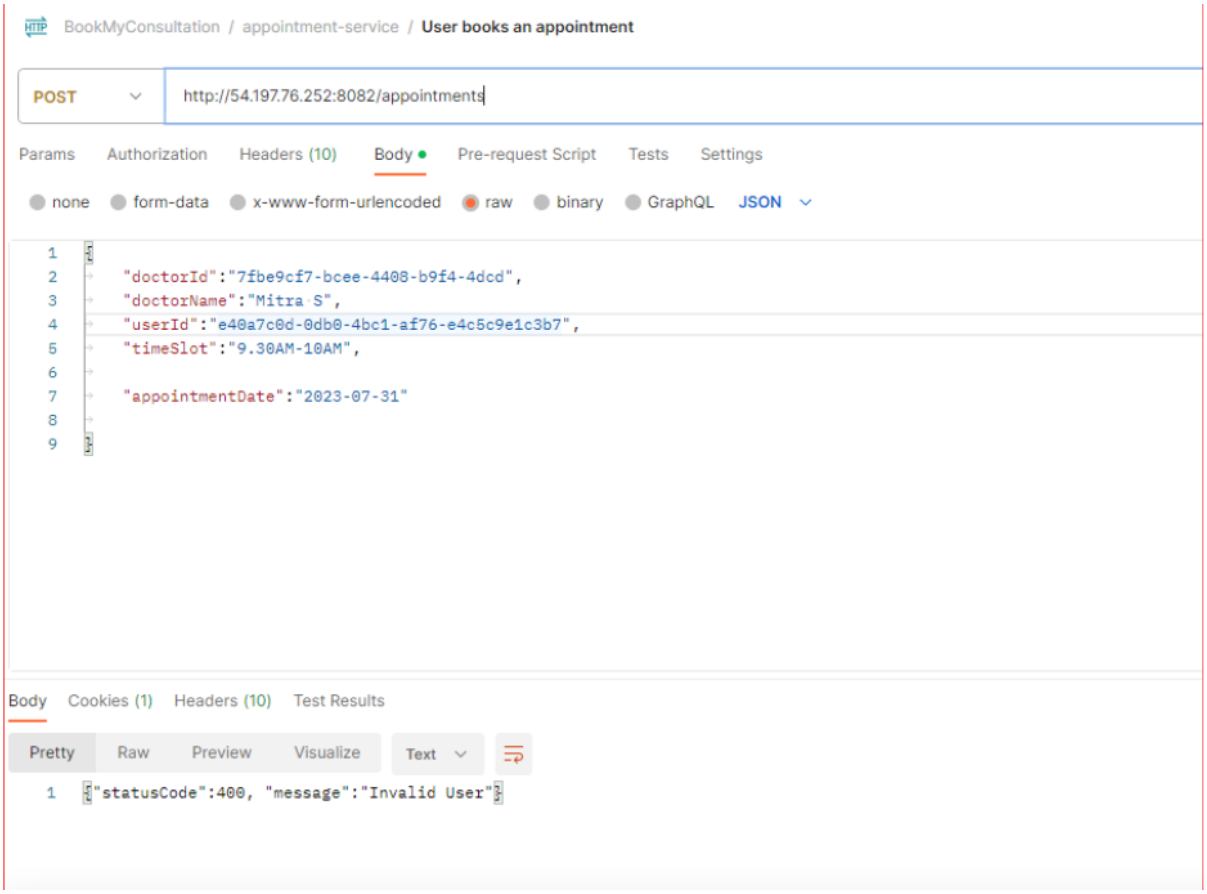
Endpoint 3: URI: `/appointments`

This endpoint is responsible for booking an appointment.

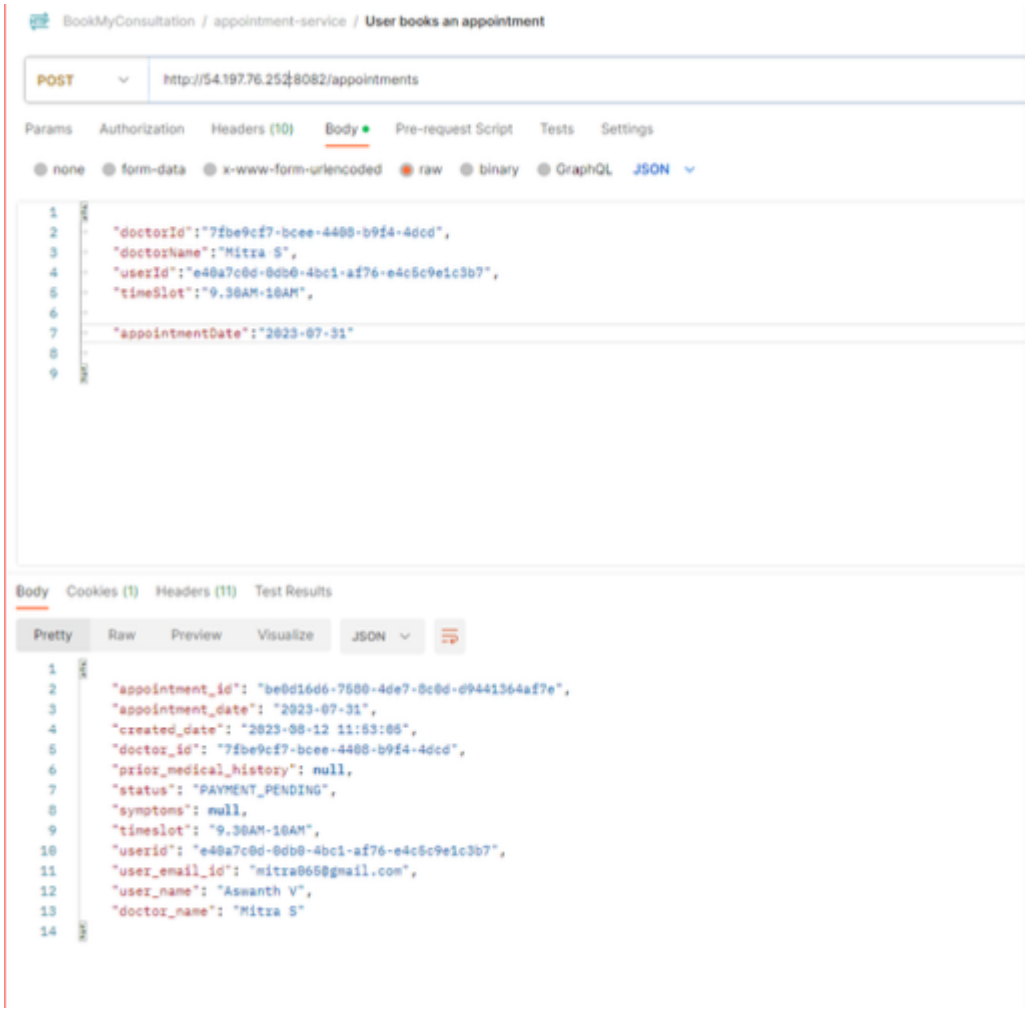
- Forbidden Error for Unauthorised Users



- Validation Error When Invalid User Id is Passed



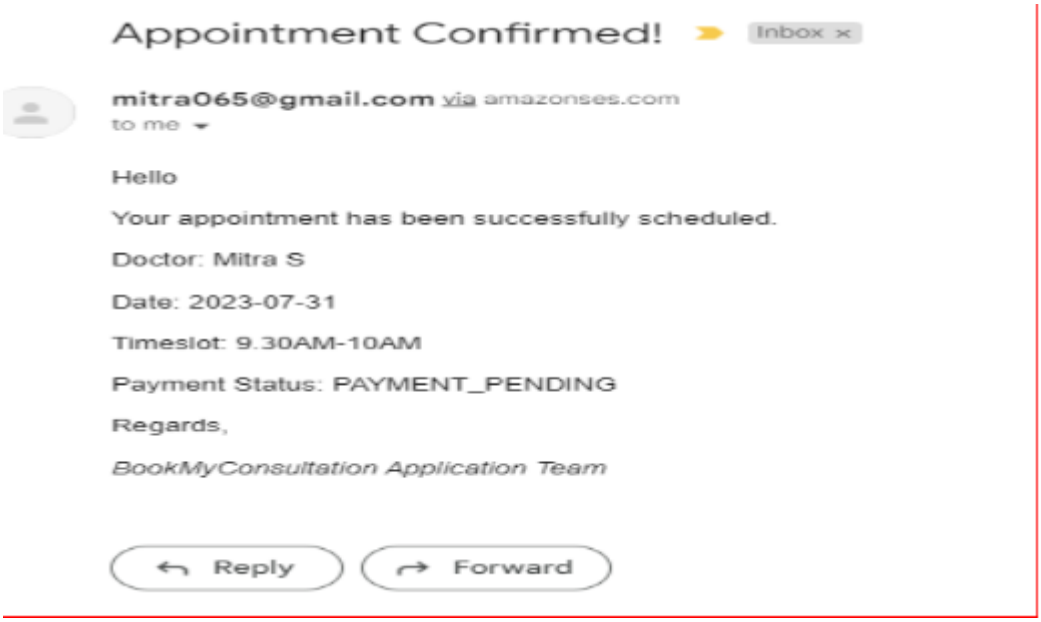
- Successful Scenario



- Kafka Consumer logs

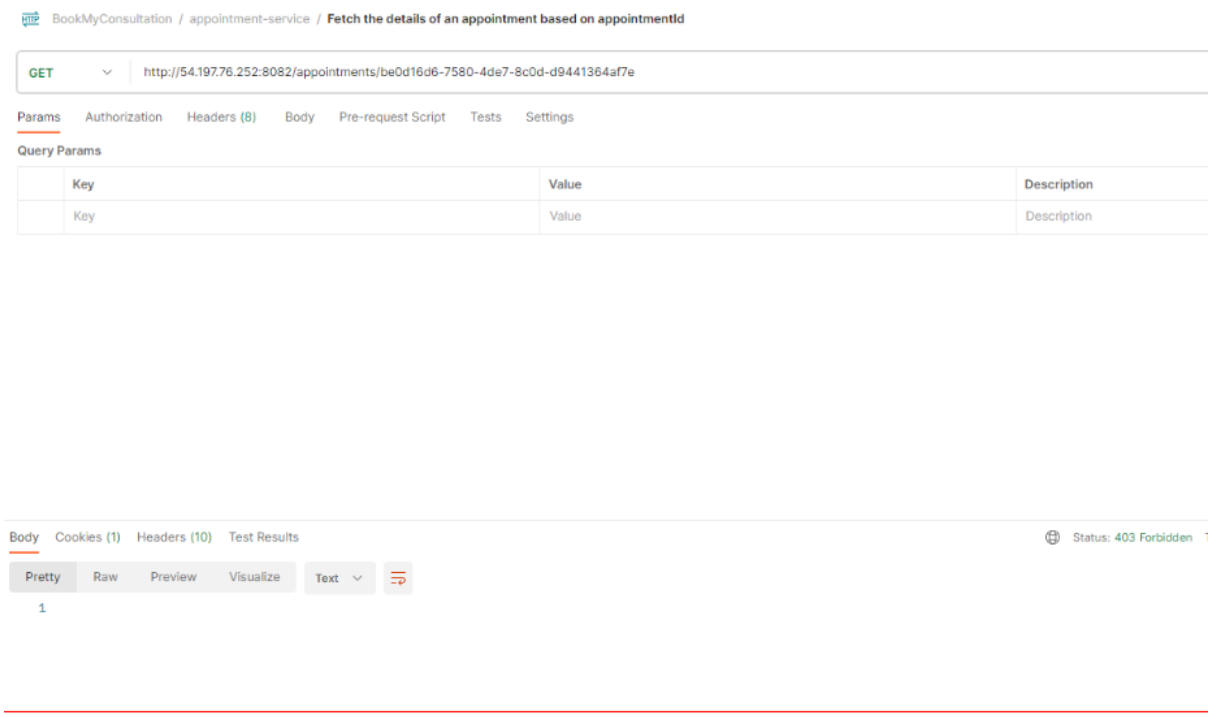
```
{
  "appointment_id" : "be0d16d6-7580-4de7-8c0d-d9441364af7e",
  "appointment_date" : "2023-07-31",
  "created_date" : "2023-08-12 11:53:05",
  "doctor_id" : "7fbe9cf7-bcee-4408-b9f4-4dcd",
  "prior_medical_history" : null,
  "status" : "PAYMENT_PENDING",
  "symptoms" : null,
  "timeslot" : "9.30AM-10AM",
  "userid" : "e40a7c0d-0db0-4bc1-af76-e4c5c9e1c3b7",
  "user_email_id" : "mitra065@gmail.com",
  "user_name" : "Aswanth V",
  "doctor_name" : "Mitra S"
}
```


- Appointment Confirmation Mail

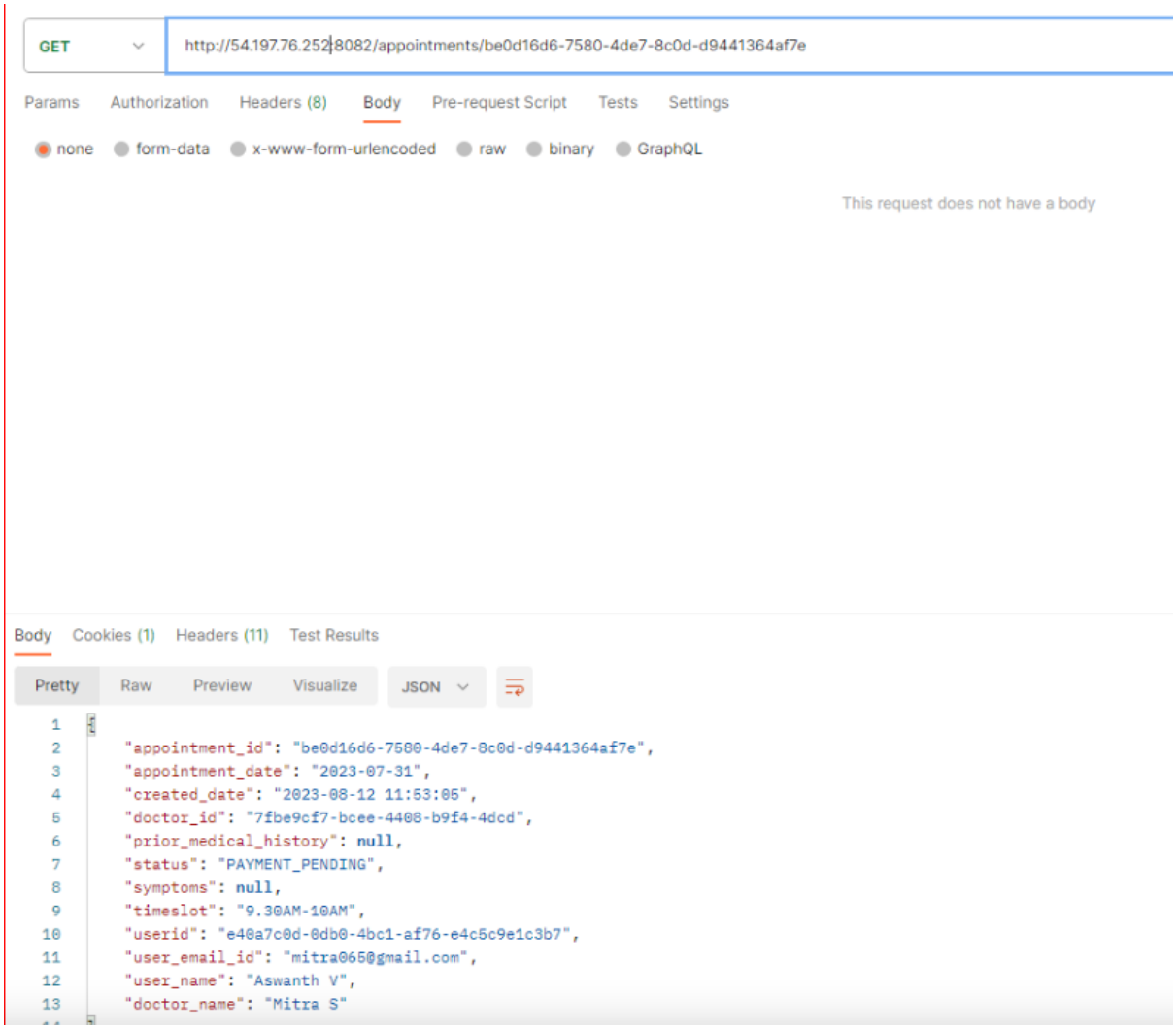


Endpoint 4: URI: `/appointments/{appointmentId}`
This endpoint is responsible for retrieving the details of an appointment.

- Forbidden Error for Unauthorised Users

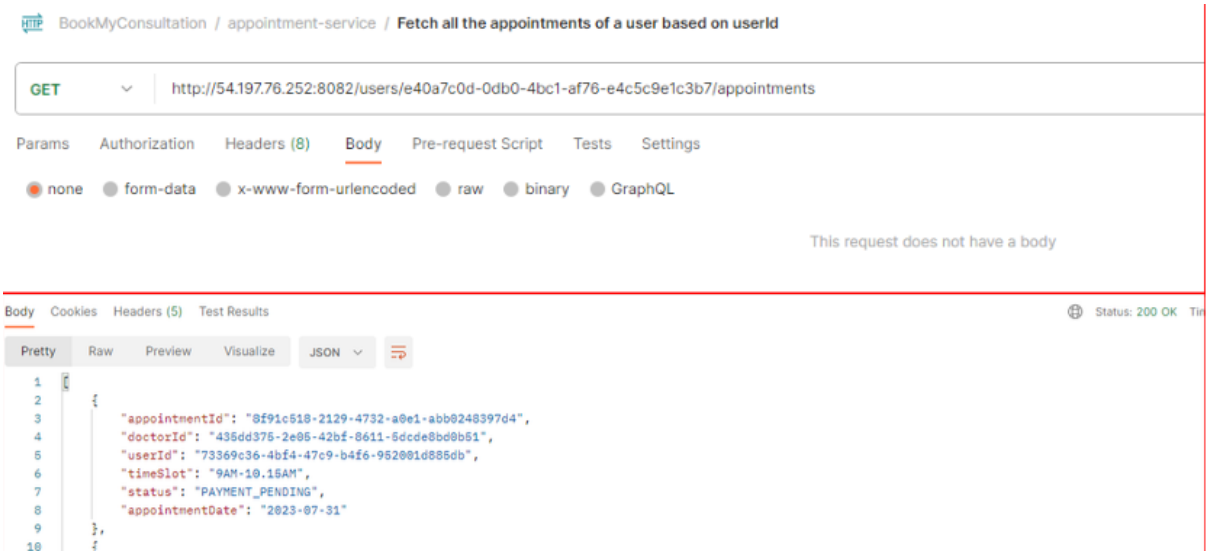


- Successful Scenario

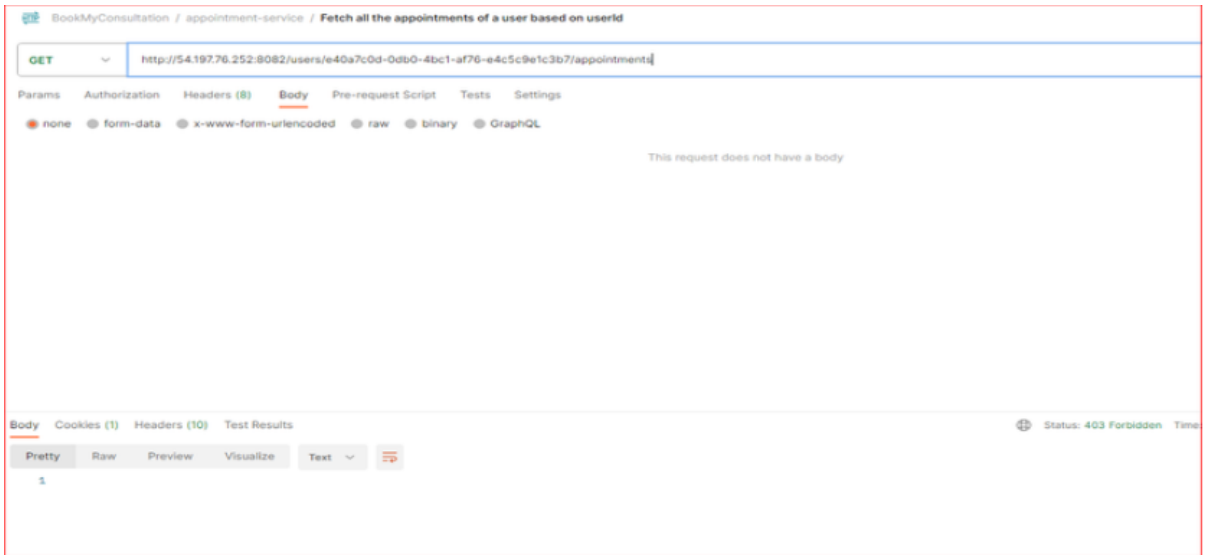


Endpoint 5: URI: `/users/{userId}/appointments`
This endpoint is responsible for retrieving the details of all the appointments corresponding to a userId.

- Successful Scenario



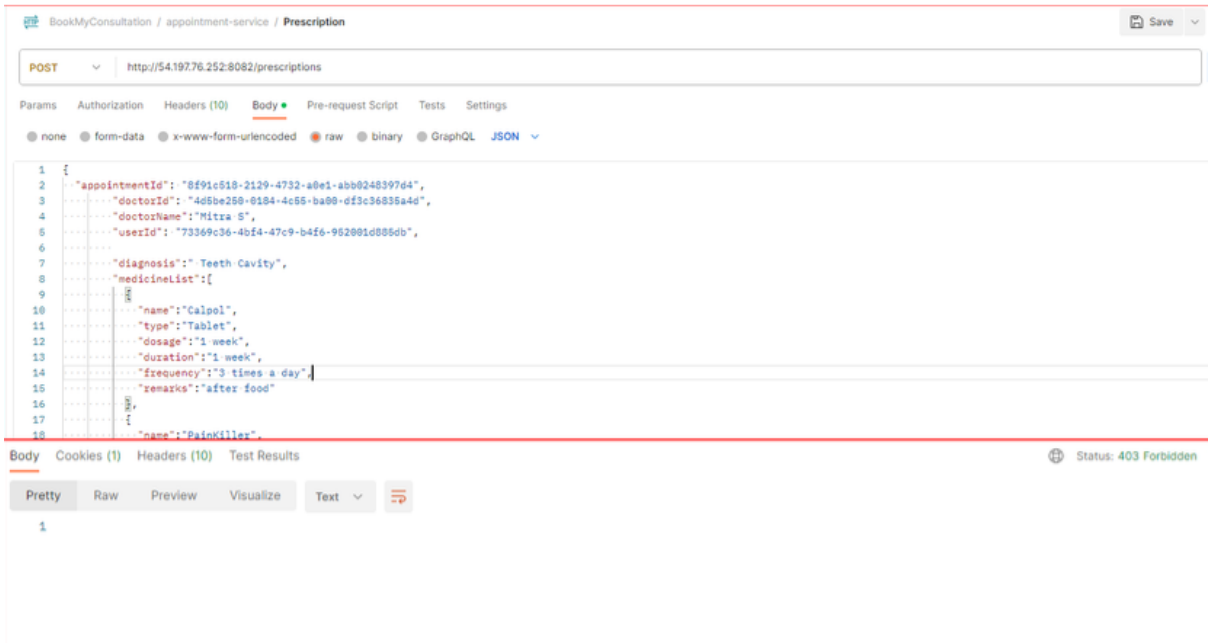
- Forbidden Error for Unauthorised Users



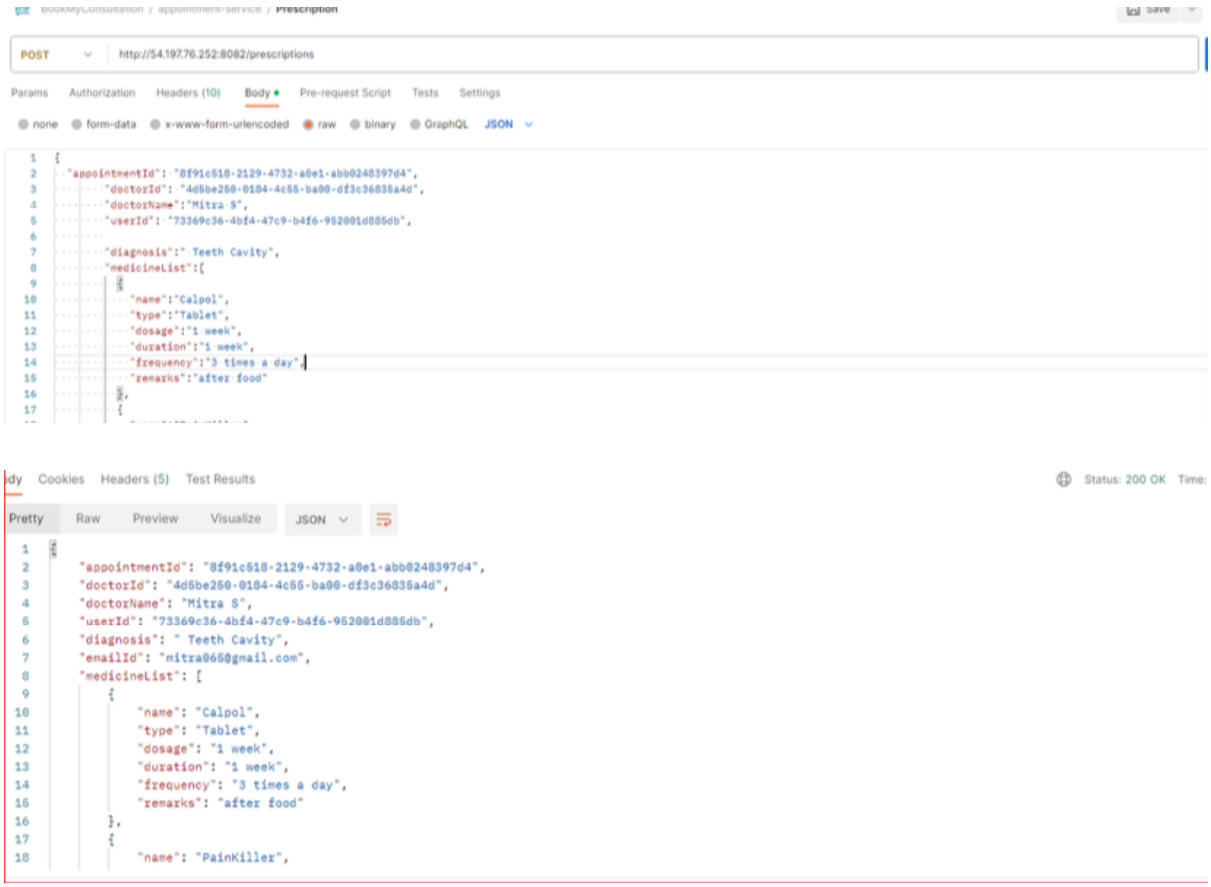
Endpoint 6: URI: `/prescriptions`

This endpoint is responsible for sending the prescriptions for the appointment.

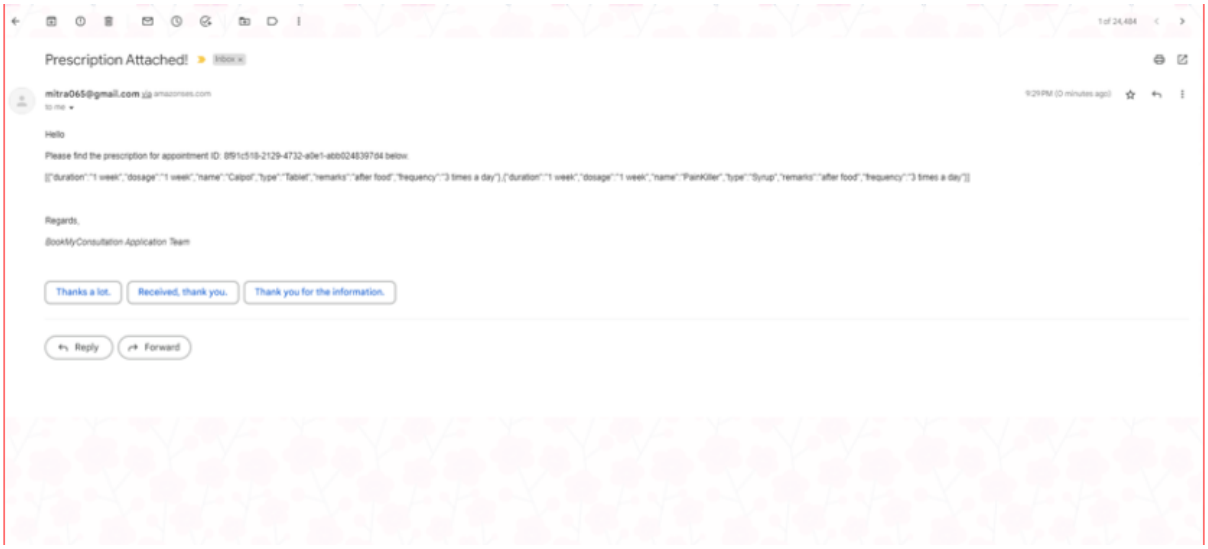
- Forbidden Error for Unauthorised Users



- Successful Scenario



- Prescription Mail Received

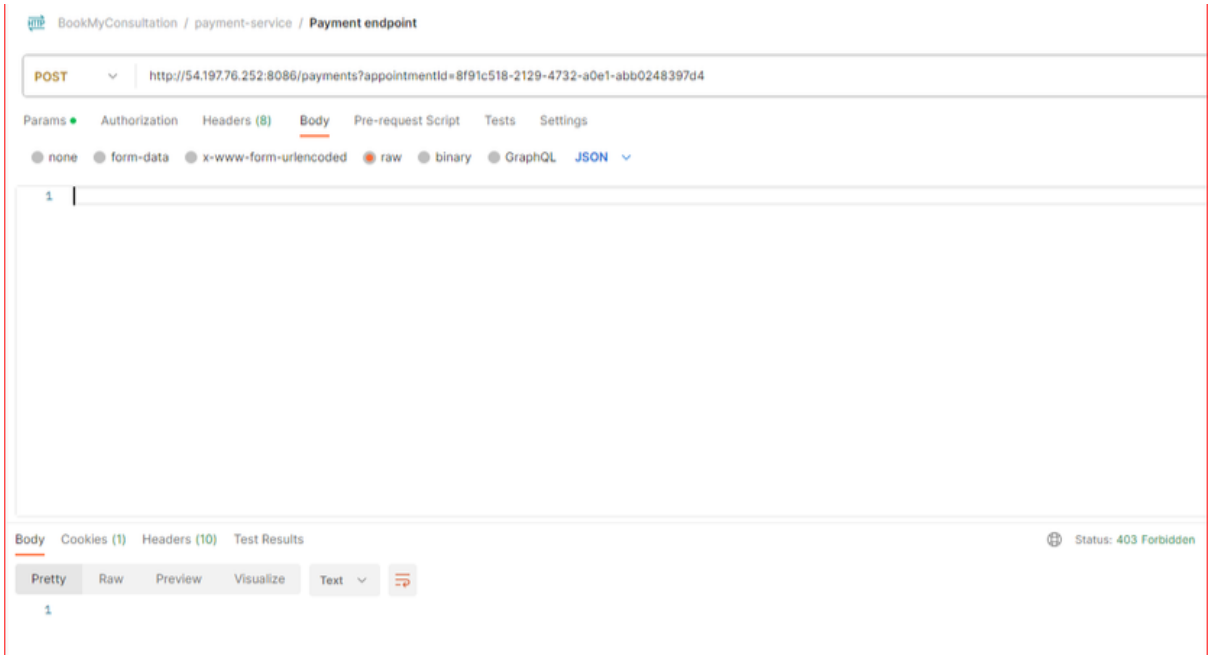


Payment Service

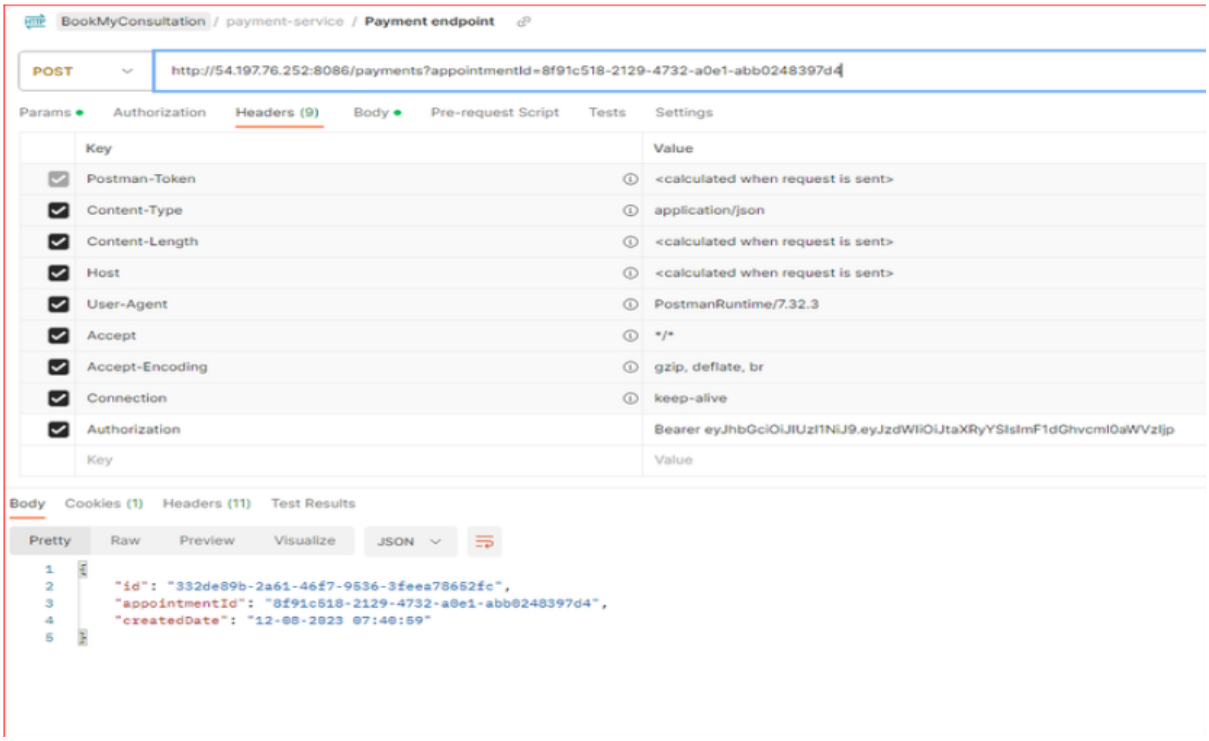
Endpoint 1: URI: `/payments?appointmentId=<the appointmentId for which you want to make apayment>`

This endpoint is responsible for making payments.

- Forbidden Error for Unauthorised Users



- Successful Scenario



Rating Service

Endpoint 1: URI: `/ratings`

This endpoint is used by the users to submit the ratings of their experience with the doctor with whom
they had an appointment.

- Successful Scenario

- Kafka Logs Screenshot

Notification Service

The notification service uses Kafka to receive notifications sent from each service. For particular
services, we receive confirmations to send emails using Amazon SES. The functionality has been
implemented as shown in each of the required APIs.

The ***CreateDoctor*** and ***CreateUser*** API sends verification email using SES to the respective email ID to
confirm that the email exists.

The ***approveDoctor***, ***rejectDoctor***, ***setAppointment*** and ***setPrescription*** APIs uses SES to send custom
messages to the respective email ID of the Doctor or the User.

Other services send kafka notifications to print by themselves, as shown in each API