**Capstone Project - BookMyConsultation**

Author: Anubhav Apurva  
Email ID: anubhavh3@gmail.com  
Batch: SD C25 Dec'20

**Table of content**

[**1.** **Environment Setup** 3](#_Toc96523128)

[1.1. VPC Setup 3](#_Toc96523129)

[1.2. EC2 Setup 3](#_Toc96523130)

[1.3. RDS Setup 5](#_Toc96523131)

[**2.** **Steps to Deploy and Run (Without Docker)** 6](#_Toc96523133)

[**3.** **Steps to Deploy and Run (With Docker)** 7](#_Toc96523134)

[**4.** **Eureka Server** 8](#_Toc96523135)

[**5.** **Doctor-Onboarding Service** 9](#_Toc96523136)

[5.1. Endpoint-1: Collect doctor information 9](#_Toc96523137)

[5.2. Endpoint-2: Upload doctor documents 11](#_Toc96523138)

[5.3. Endpoint-3: Approve doctor registration 12](#_Toc96523139)

[5.4. Endpoint-4: Reject doctor registration 13](#_Toc96523140)

[5.5. Endpoint-5: Return list of doctors 14](#_Toc96523141)

[5.5.1. Scenario-1: The status is pending 14](#_Toc96523142)

[5.5.2. Scenario-2: The status is Active 14](#_Toc96523143)

[5.5.3. Scenario-2: Based on speciality 15](#_Toc96523144)

[5.6. Endpoint-6: Return doctors based on doctor-ID 15](#_Toc96523145)

[**6.** **User-Onboarding Service** 16](#_Toc96523146)

[6.1. Endpoint-1: Collect user information 16](#_Toc96523147)

[6.2. Endpoint-2: Fetch user information 18](#_Toc96523148)

[6.3. Endpoint-3: Upload user documents 19](#_Toc96523149)

[**7.** **Appointment Service** 20](#_Toc96523150)

[7.1. Endpoint-1: Update availability of the doctors. 20](#_Toc96523151)

[7.2. Endpoint-2: Fetch doctor's availability. 21](#_Toc96523152)

[7.3. Endpoint-3: Book Appointment 22](#_Toc96523153)

[7.4. Endpoint-4: Fetch Appointment details 23](#_Toc96523154)

[7.5. Endpoint-5: Fetch Appointments by userId 24](#_Toc96523155)

[7.6. Endpoint-6: Send prescription 25](#_Toc96523156)

[**8.** **Payment Service** 27](#_Toc96523157)

[8.1. Endpoint-1: Make payment for appointment. 27](#_Toc96523158)

[**9.** **Rating Service** 28](#_Toc96523159)

[9.1. Endpoint-1: Rate Doctors. 28](#_Toc96523160)

[**10.** **Notification Service** 29](#_Toc96523161)

[10.1. Doctor Registration Approval 29](#_Toc96523162)

[10.2. Doctor Registration Rejection 29](#_Toc96523163)

[10.3. Appointment Confirmation 30](#_Toc96523164)

[10.4. Prescription 30](#_Toc96523165)

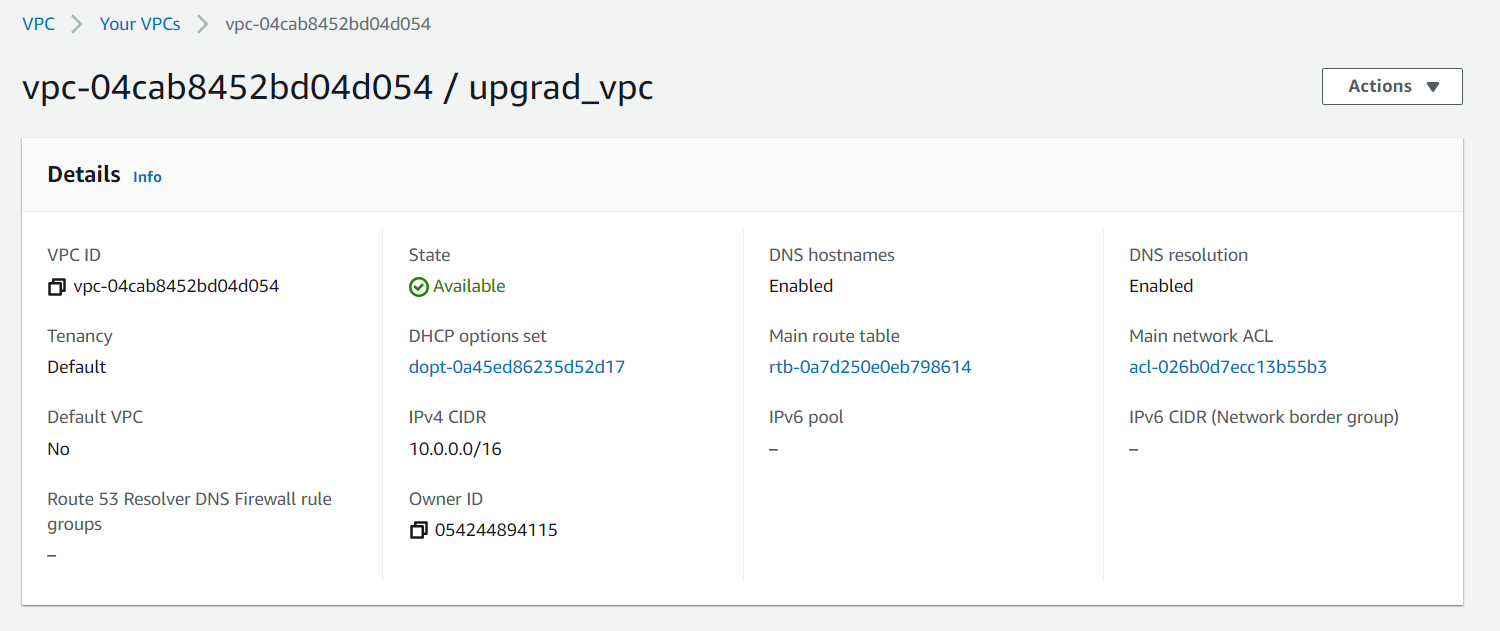
[**11.** **API Gateway** 31](#_Toc96523166)

[**12.** **Security** 31](#_Toc96523167)

[**13.** **Future Enhancements** 31](#_Toc96523168)

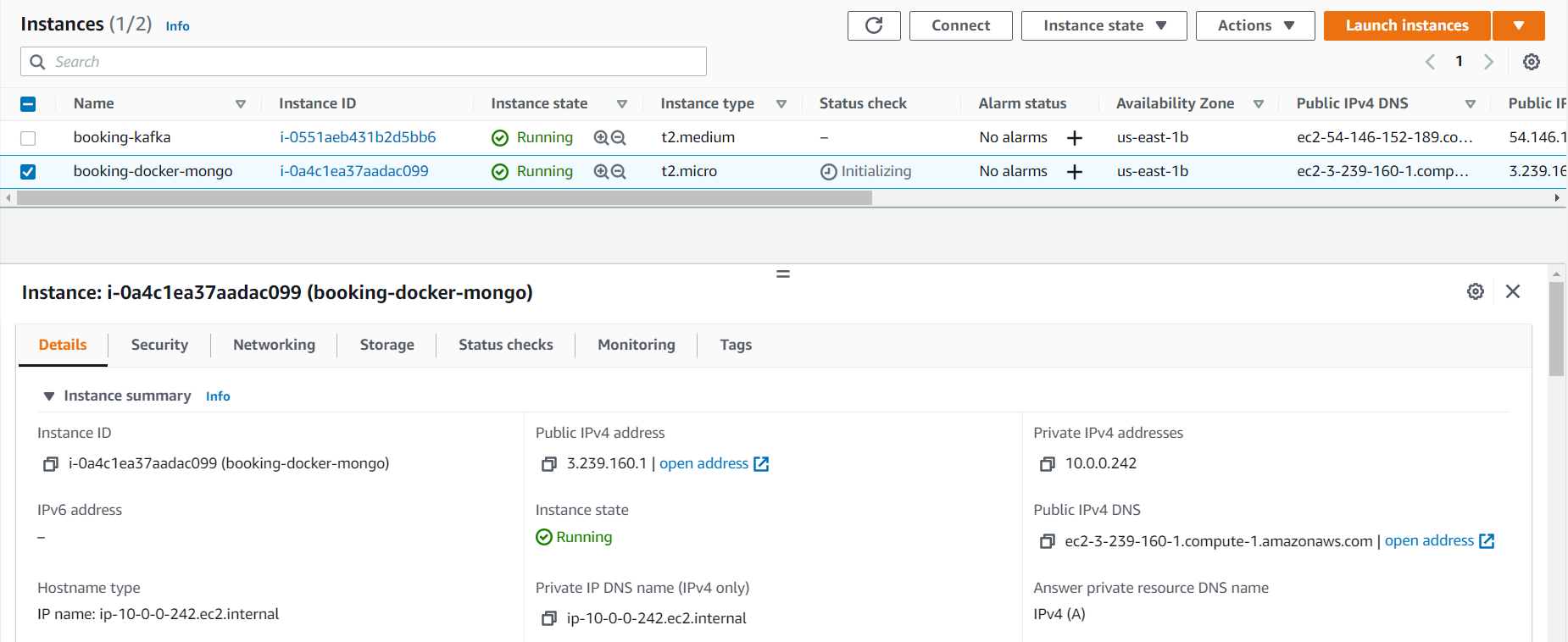
# **Environment Setup**

## VPC Setup



## EC2 Setup

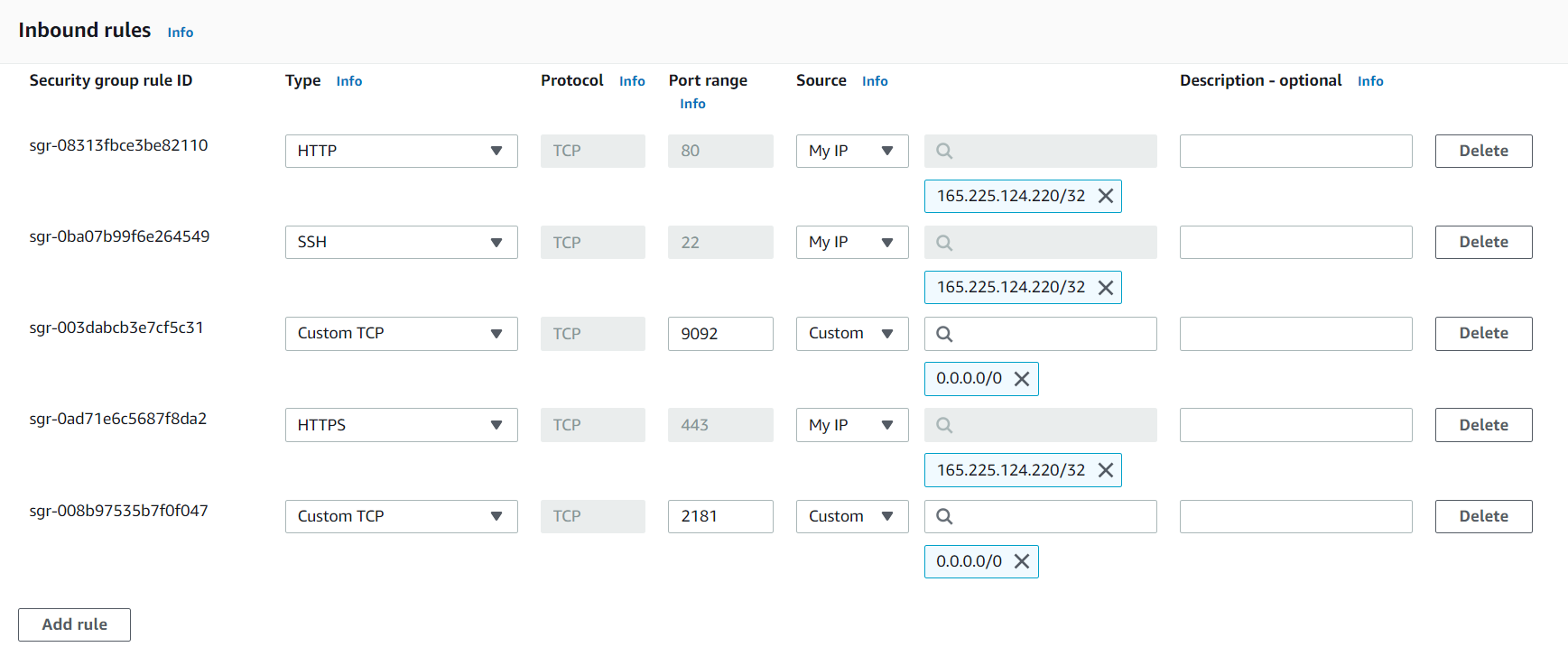
In this project we need two EC2 instances. The first instance (booking-docker-mongo in this screenshot) will be used to run MongoDB and deploy the developed microservices. The second instance (booking-kafka in this screenshot) will be used to run Kafka.



EC2 (first instance) Security Group Inbound Rules -

|  |
| --- |
|  |

EC2 (second instance) Security Group Inbound Rules -

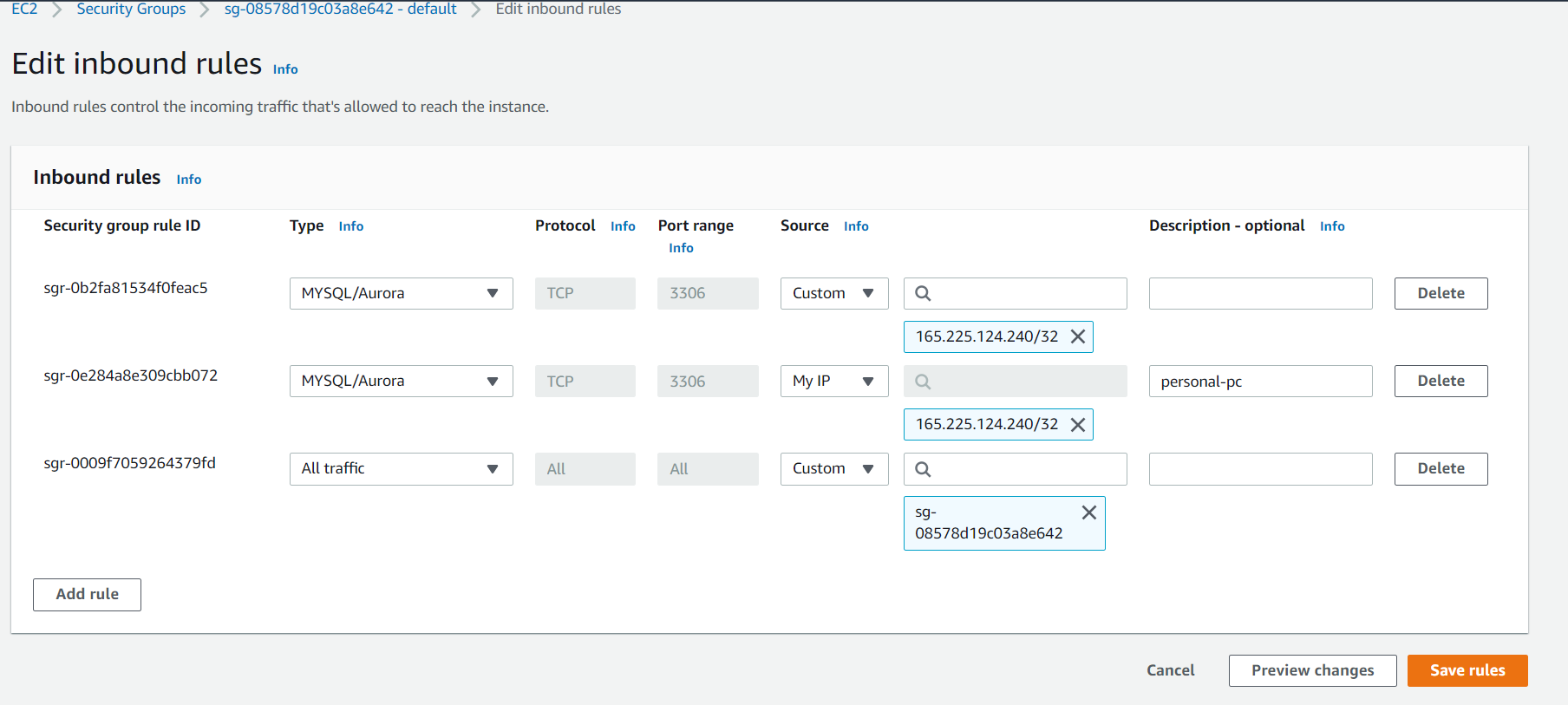


## RDS Setup

Create a RDS MySQL instance and create a database (e.g. upgrad) in it.

## 

RDS Security Group Inbound Rules –



# **Steps to Deploy and Run (Without Docker)**

* Start Kafka
  + Start Kafka server and zookeeper on the second EC2 instance
* Start MongoDBs
  + Start MongoDB either on the first EC2 instance or on the second EC2 instance.
* FTP BookMyConsultation.zip to the first EC2 instance
* Unzip Sweet-Home.zip  
    
  $ unzip BookMyConsultation.zip
* Configure ENV variables

Set all the ENV variables in the env file in the root directory of the project. Export all the variables  
  
$ . ./env

* cd to each microservice directory and bring up the service

$ cd BookMyConsultation/eureka

$ mvn spring-boot:run

$ cd BookMyConsultation/doctor-service

$ mvn spring-boot:run

$ cd BookMyConsultation/user-service

$ mvn spring-boot:run

$ cd BookMyConsultation/appointment-service

$ mvn spring-boot:run

$ cd BookMyConsultation/payment-service

$ mvn spring-boot:run

$ cd BookMyConsultation/rating-service

$ mvn spring-boot:run

$ cd BookMyConsultation/notification-service

$ mvn spring-boot:run

$ cd BookMyConsultation/bmc-gateway

$ mvn spring-boot:run

# **Steps to Deploy and Run (With Docker)**

* Start Kafka
  + Start Kafka server and zookeeper on the second EC2 instance
* Start MongoDBs
  + Start MongoDB either on the first EC2 instance or on the second EC2 instance.
* FTP BookMyConsultation.zip to the first EC2 instance
* Unzip Sweet-Home.zip  
    
  $ unzip BookMyConsultation.zip
* Configure ENV variables

Set all the ENV variables or edit them in docker-compose.yaml file in the root directory

* Create docker bridge network

$ sudo docker network create microservicesnet

* Create JAR of each service and build the docker image

$ cd BookMyConsultation/eureka

$ mvn clean install spring-boot:repackage

$ sudo docker build -t bookingaap/eurekasvc:1.0.0 .

$ cd BookMyConsultation/eureka

$ mvn clean install spring-boot:repackage

$ sudo docker build -t bookingaap/doctorsvc:1.0.0 .

And so on…

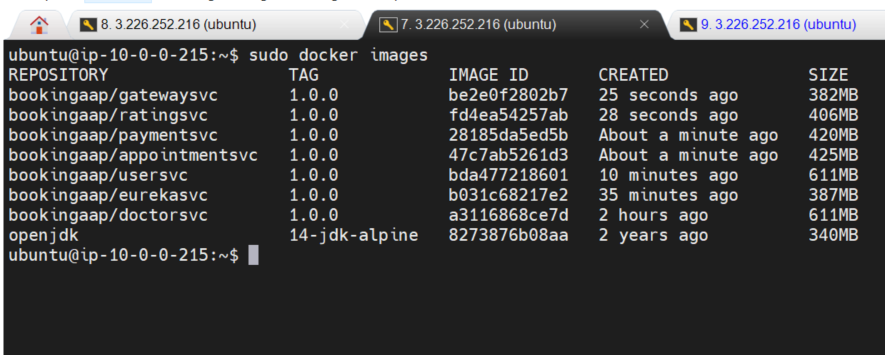
* Start services

$ sudo docker run -it --name=serviceregistry -d -p8761:8761 -e EUREKA\_HOST\_NAME=54.87.134.192 --net=microservicesnet bookingaap/eurekasvc:1.0.0

Similarly, repeat above steps for each service.

Or use the docker-compose file to build and deploy all the services at once

$ docker-compose up -d



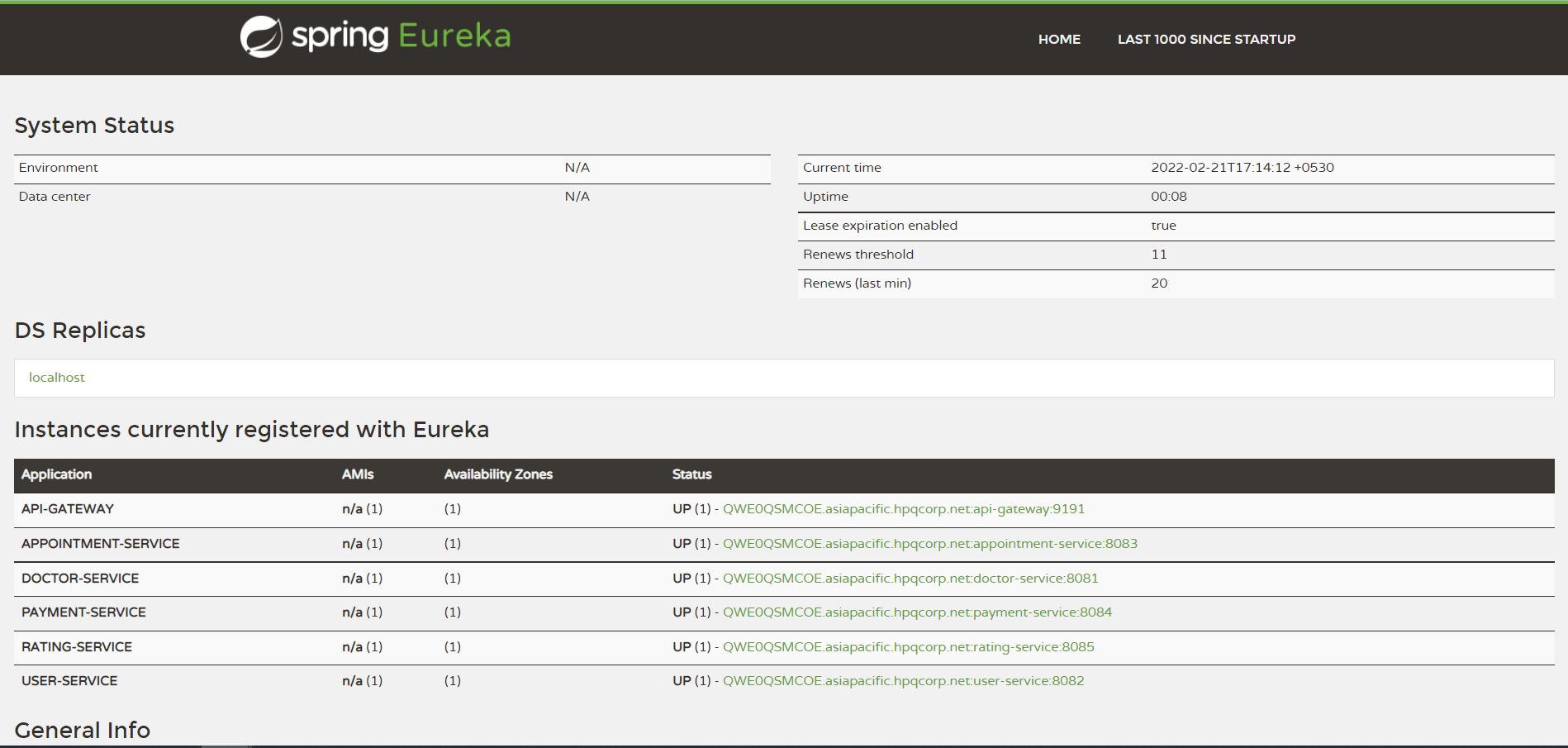
# **Eureka Server**

Eureka Server is an application that holds the information about all client-service applications. It knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

Every micro-service in this project including the API gateway registers themselves into the Eureka server.

The Eureka Server is started on port 8761.

On the browser if we go to <http://localhost:8761/> or http://<ec2-host-ip>:8671/ we can see the services are up –



# **Doctor-Onboarding Service**

## Endpoint-1: Collect doctor information

**POST** localhost:8081/doctors

Content-Type application/json

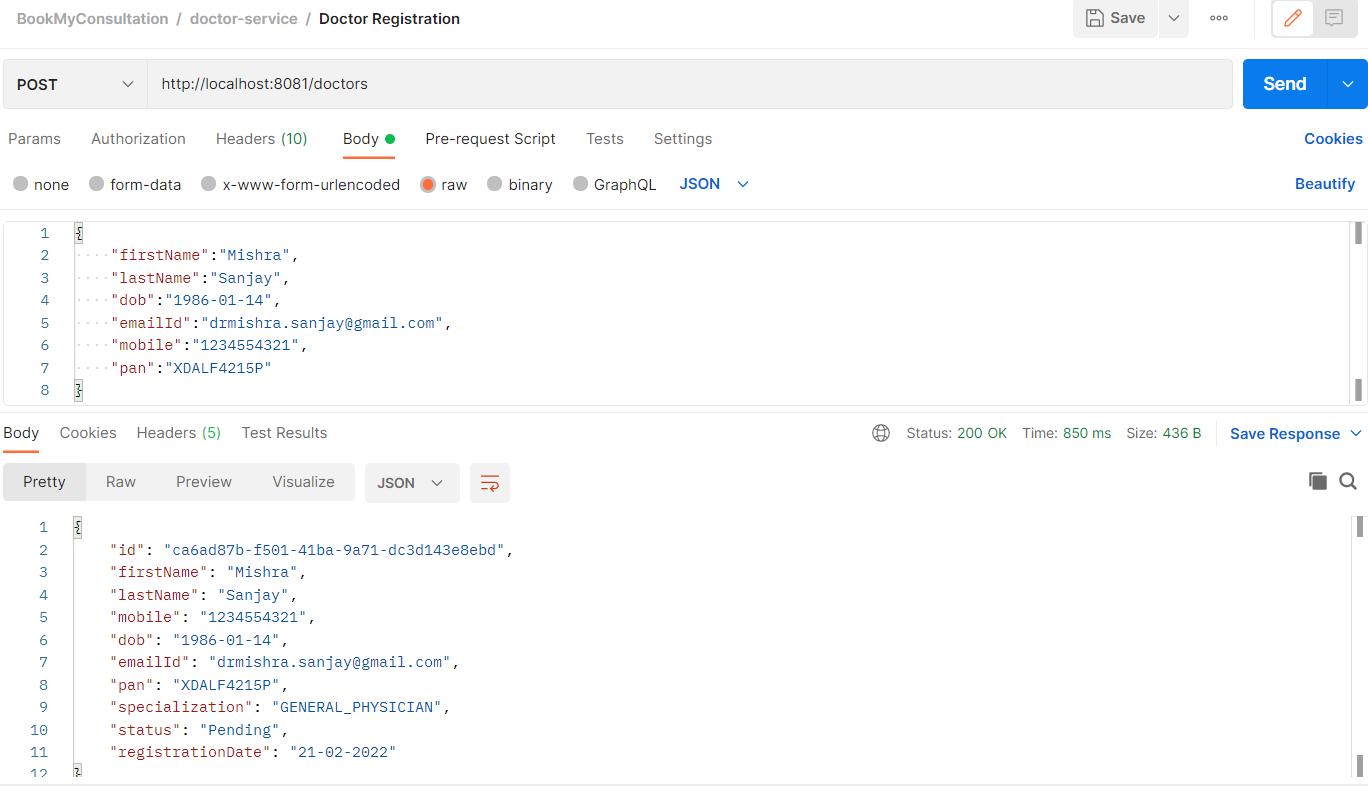
Request Body Ex –

|  |
| --- |
| {      "firstName":"Mishra",      "lastName":"Sanjay",      "dob":"1986-01-14",      "emailId":"drmishra.sanjay@gmail.com",      "mobile":"1234554321",      "pan":"XDALF4215P"  } |

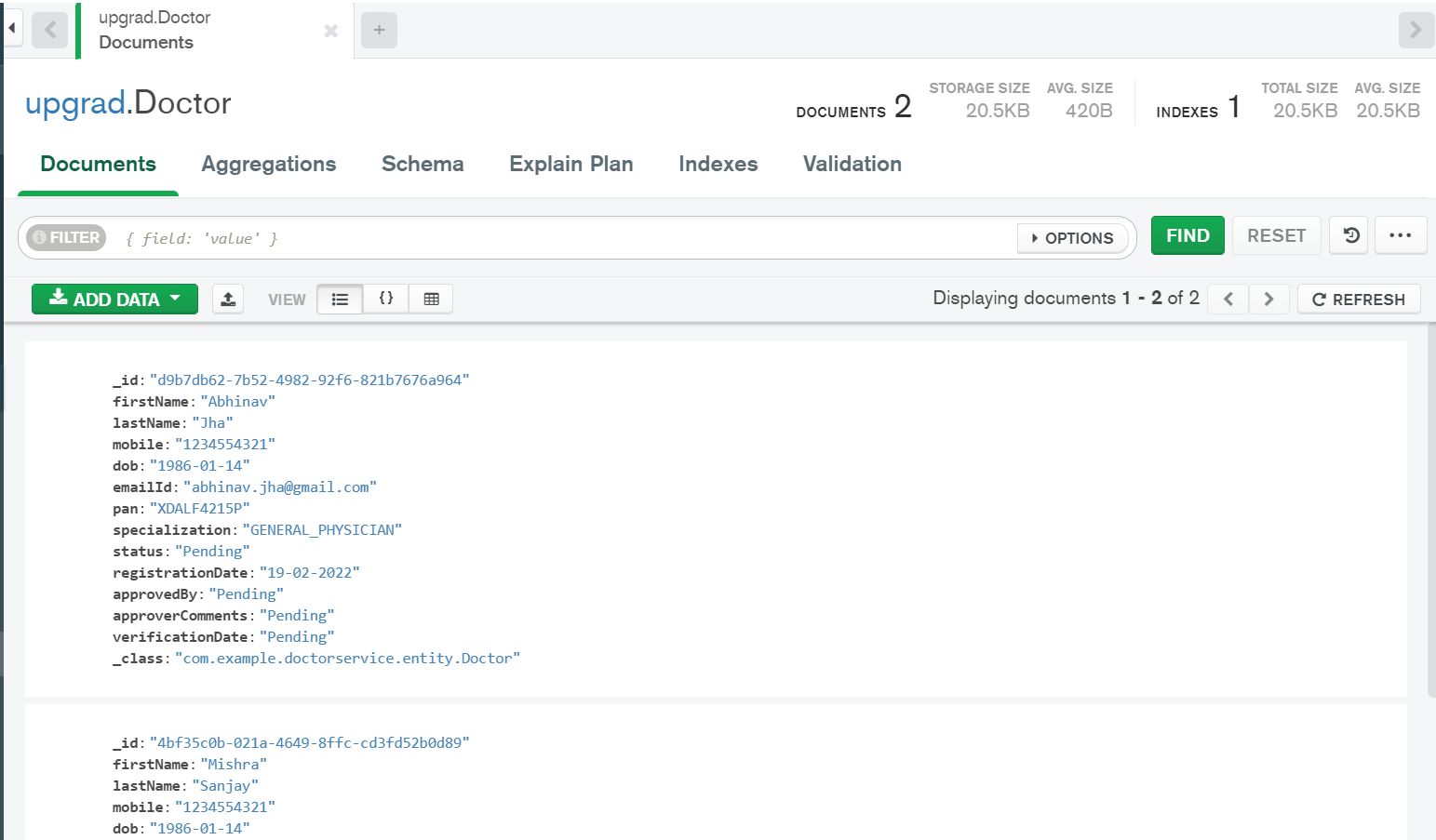
Response Body Ex –

|  |
| --- |
| {      "id": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",      "firstName": "Mishra",      "lastName": "Sanjay",      "mobile": "1234554321",      "dob": "1986-01-14",      "emailId": "drmishra.sanjay@gmail.com",      "pan": "XDALF4215P",      "specialization": "GENERAL\_PHYSICIAN",      "status": "Pending",      "registrationDate": "21-02-2022"  } |

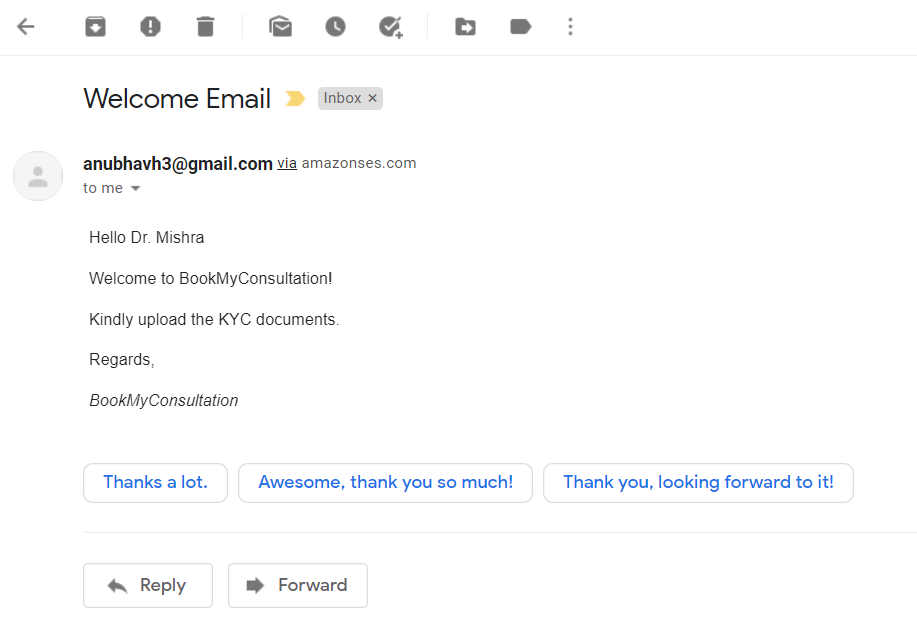
Calling the endpoint from POSTMAN -



Mongo Collection 'Doctor' -



Email received by the new doctor-



## Endpoint-2: Upload doctor documents

**POST** localhost:8081/doctors/{doctorId}/document

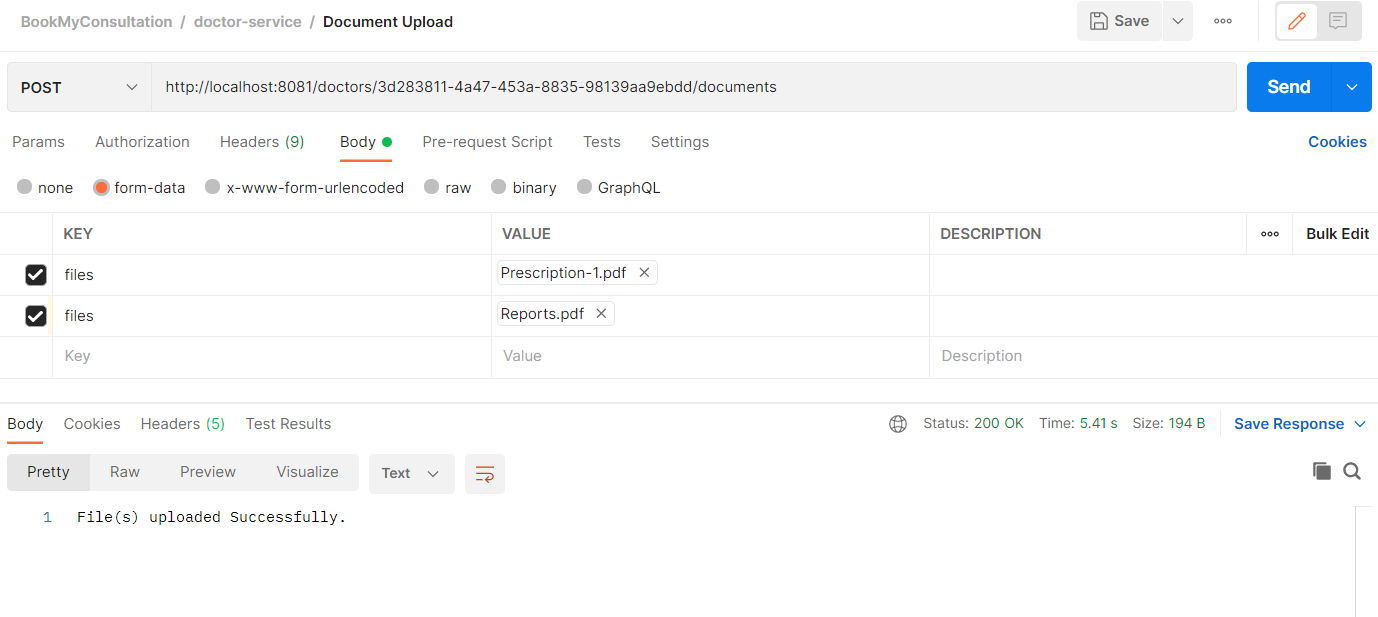
Request Body Ex –

|  |
| --- |
|  |

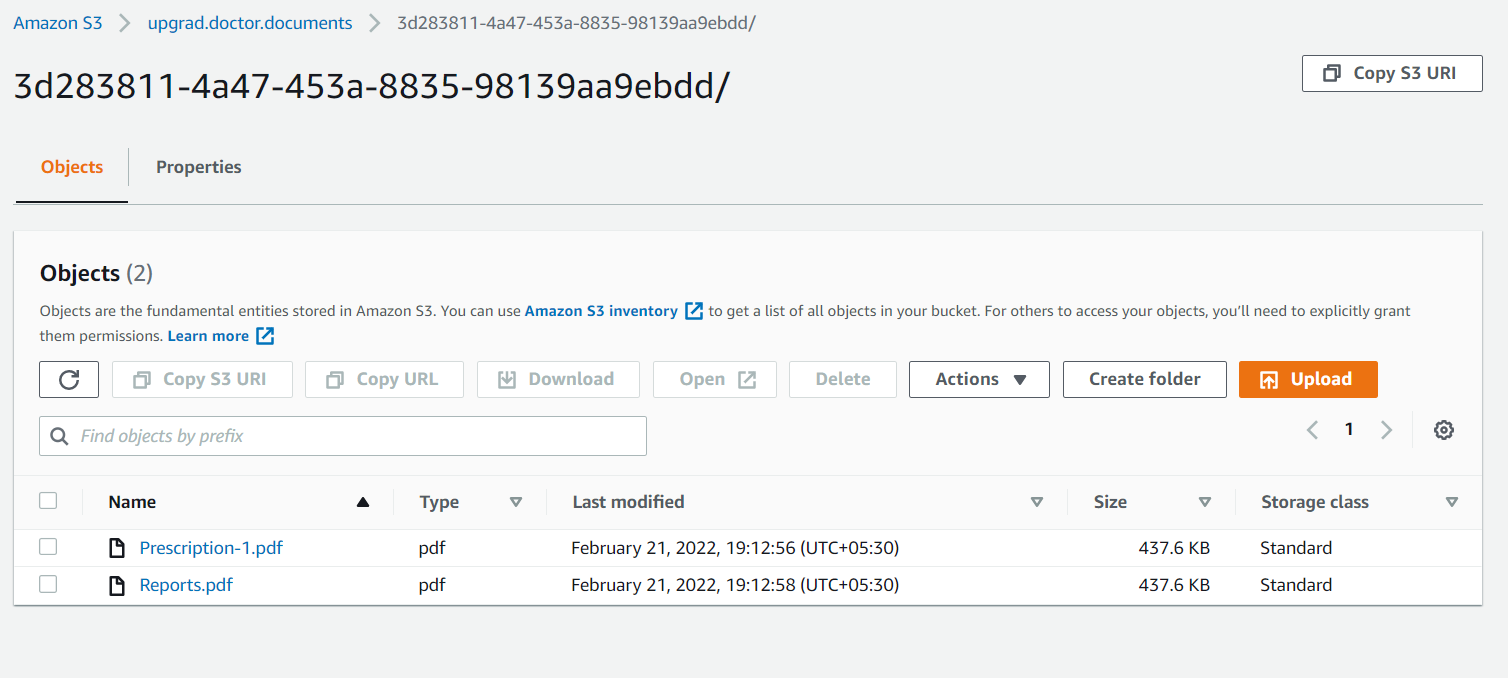
Response Body Ex –

|  |
| --- |
| File(s) uploaded Successfully. |

Calling the endpoint from POSTMAN -



S3 bucket updated with the uploaded documents -



## Endpoint-3: Approve doctor registration

**PUT** localhost:8081/doctors/{doctorId}/approve

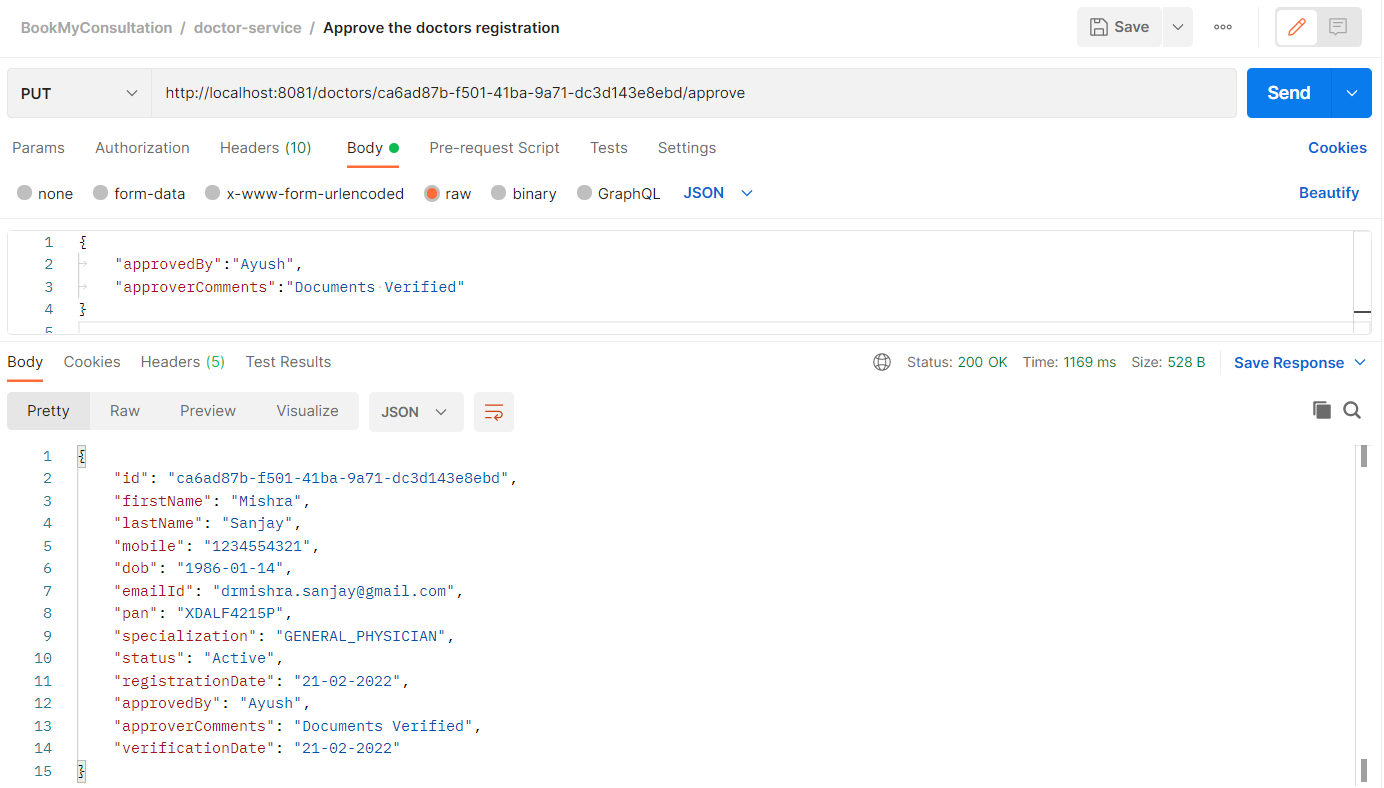
Request Body Ex –

|  |
| --- |
| {      "approvedBy":"Ayush",      "approverComments":"Documents Verified"  } |

Response Body Ex –

|  |
| --- |
| {      "id": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",      "firstName": "Mishra",      "lastName": "Sanjay",      "mobile": "1234554321",      "dob": "1986-01-14",      "emailId": "drmishra.sanjay@gmail.com",      "pan": "XDALF4215P",      "specialization": "GENERAL\_PHYSICIAN",      "status": "Active",      "registrationDate": "21-02-2022",      "approvedBy": "Ayush",      "approverComments": "Documents Verified",      "verificationDate": "21-02-2022"  } |

Calling the endpoint from POSTMAN –



## Endpoint-4: Reject doctor registration

**PUT** localhost:8081/doctors/{doctorId}/reject

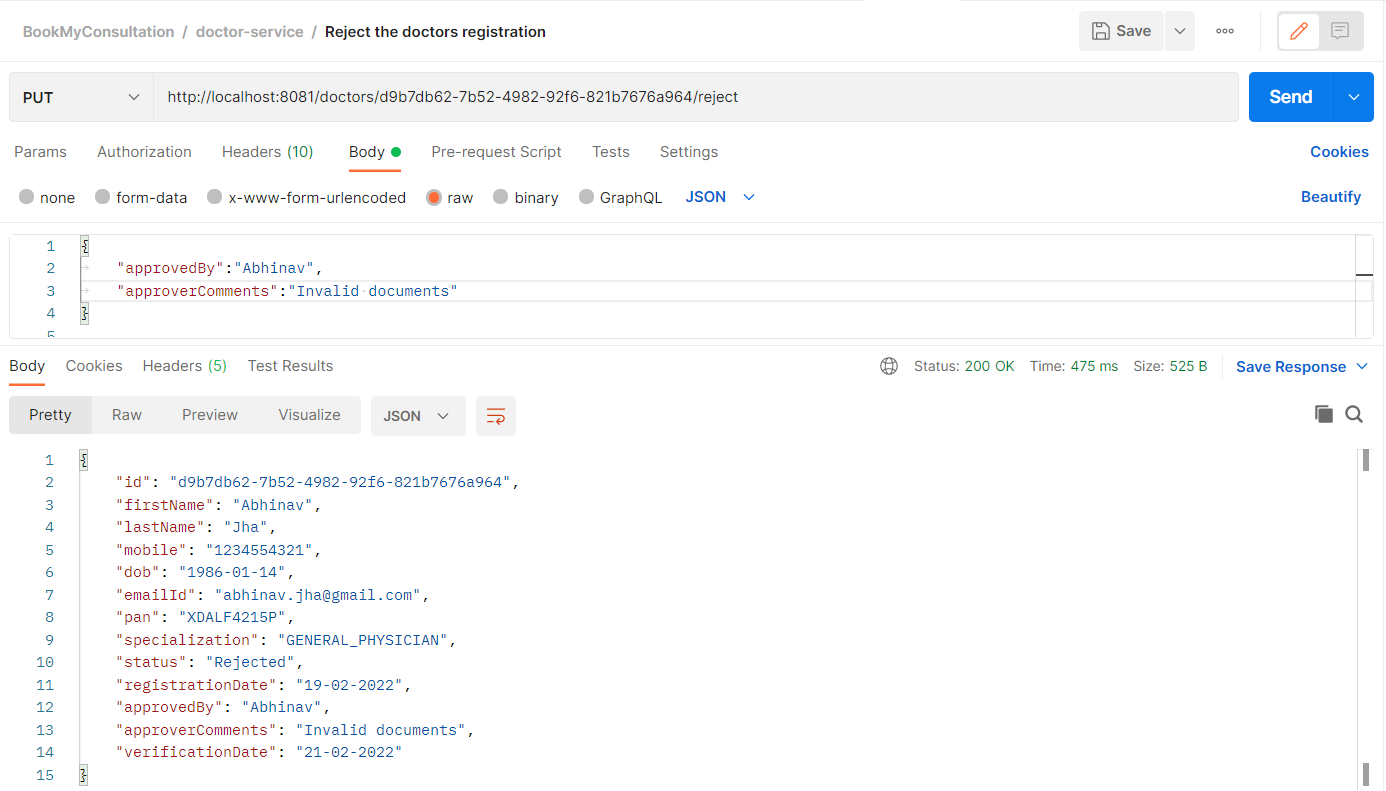
Request Body Ex –

|  |
| --- |
| {      "approvedBy":"Abhi",      "approverComments":"Invalid documents"  } |

Response Body Ex –

|  |
| --- |
| {      "id": "d9b7db62-7b52-4982-92f6-821b7676a964",      "firstName": "Abhinav",      "lastName": "Jha",      "mobile": "1234554321",      "dob": "1986-01-14",      "emailId": "abhinav.jha@gmail.com",      "pan": "XDALF4215P",      "specialization": "GENERAL\_PHYSICIAN",      "status": "Rejected",      "registrationDate": "19-02-2022",      "approvedBy": "Abhi",      "approverComments": "Invalid documents",      "verificationDate": "21-02-2022"  } |

Calling the endpoint from POSTMAN –

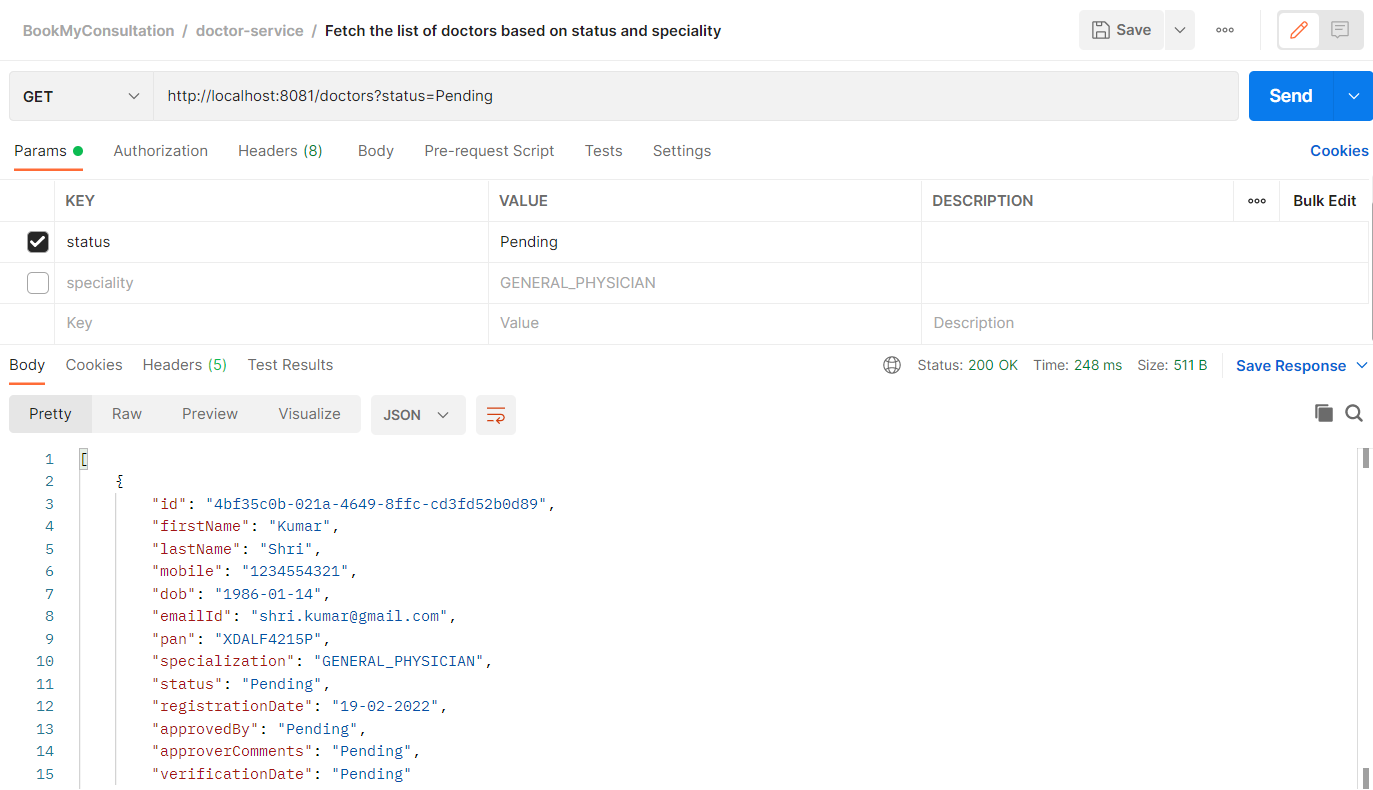


## Endpoint-5: Return list of doctors

## Scenario-1: The status is pending

**GET** localhost:8081/doctors?status=Pending

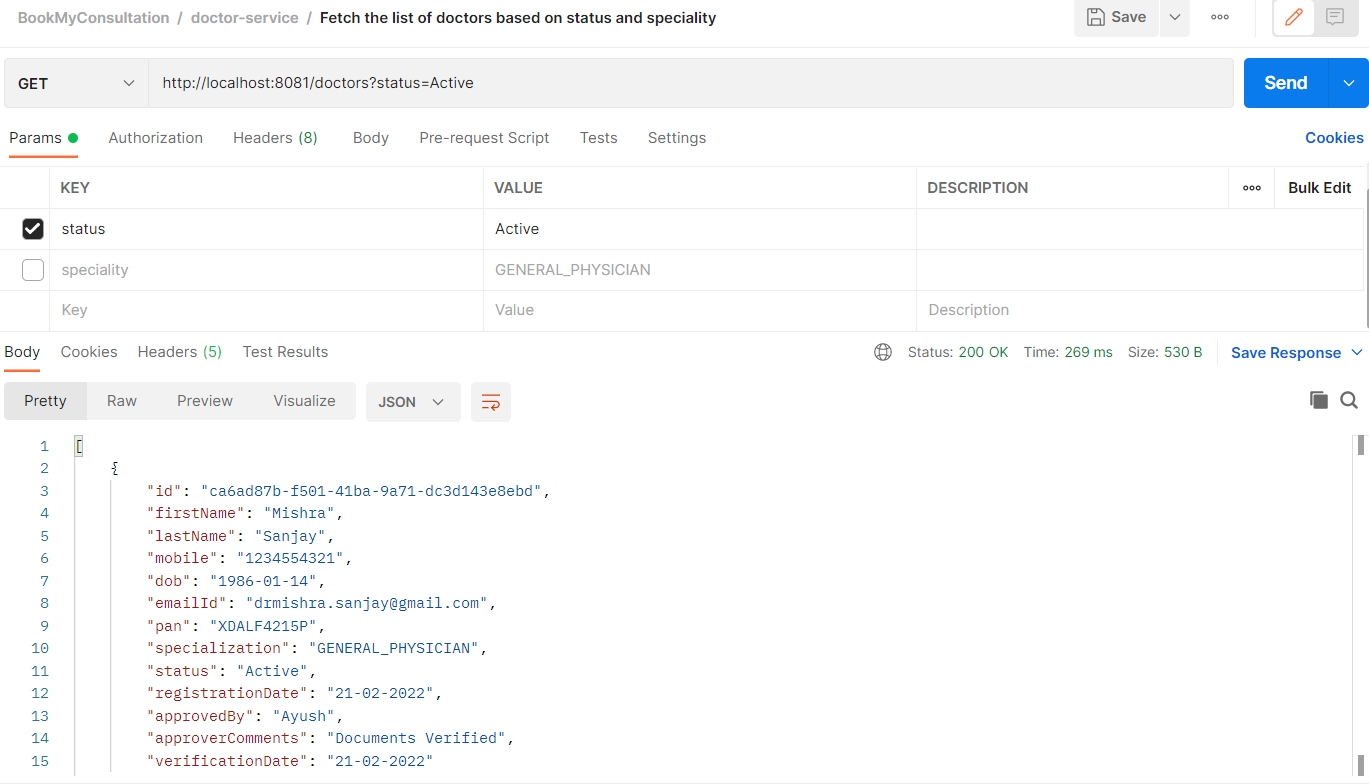
Calling the endpoint from POSTMAN –



## Scenario-2: The status is Active

**GET** localhost:8081/doctors?status=Active

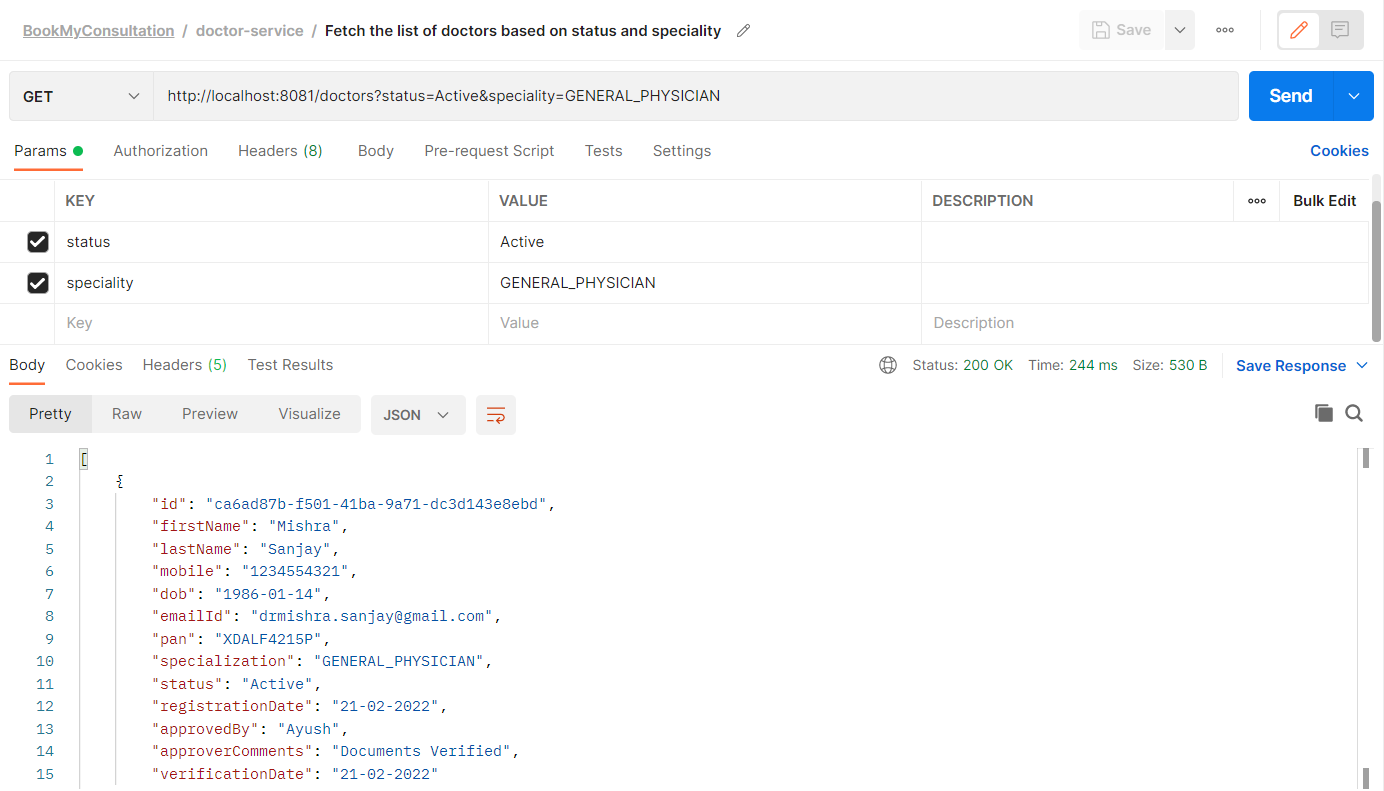
Calling the endpoint from POSTMAN –



## Scenario-2: Based on speciality

**GET** localhost:8081/status=Active&speciality=GENERAL\_PHYSICIAN

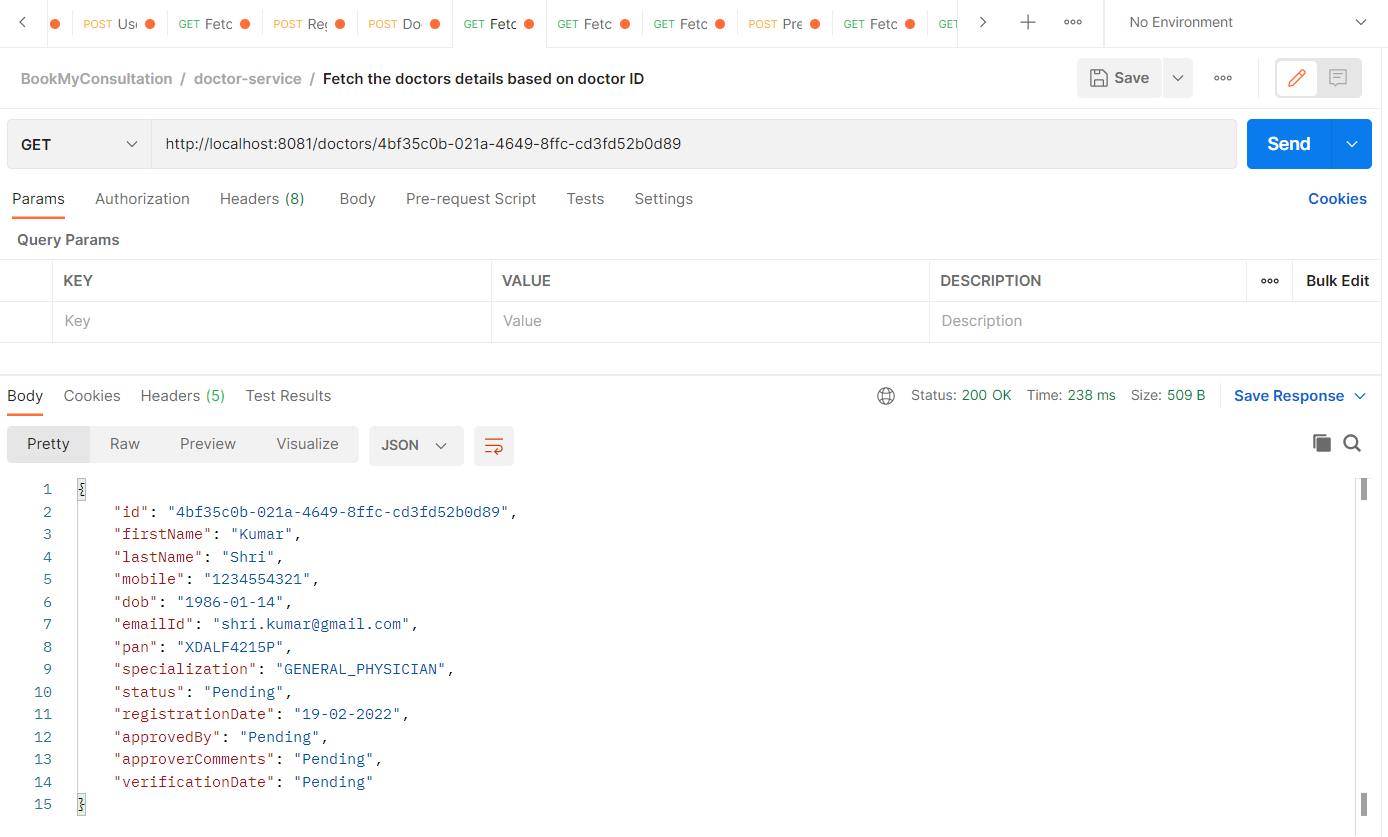
Calling the endpoint from POSTMAN –



## Endpoint-6: Return doctors based on doctor-ID

**GET** localhost:8081/ /doctors/{doctorId}

Calling the endpoint from POSTMAN –



# **User-Onboarding Service**

## Endpoint-1: Collect user information

**POST** localhost:8082/users

Content-Type application/json

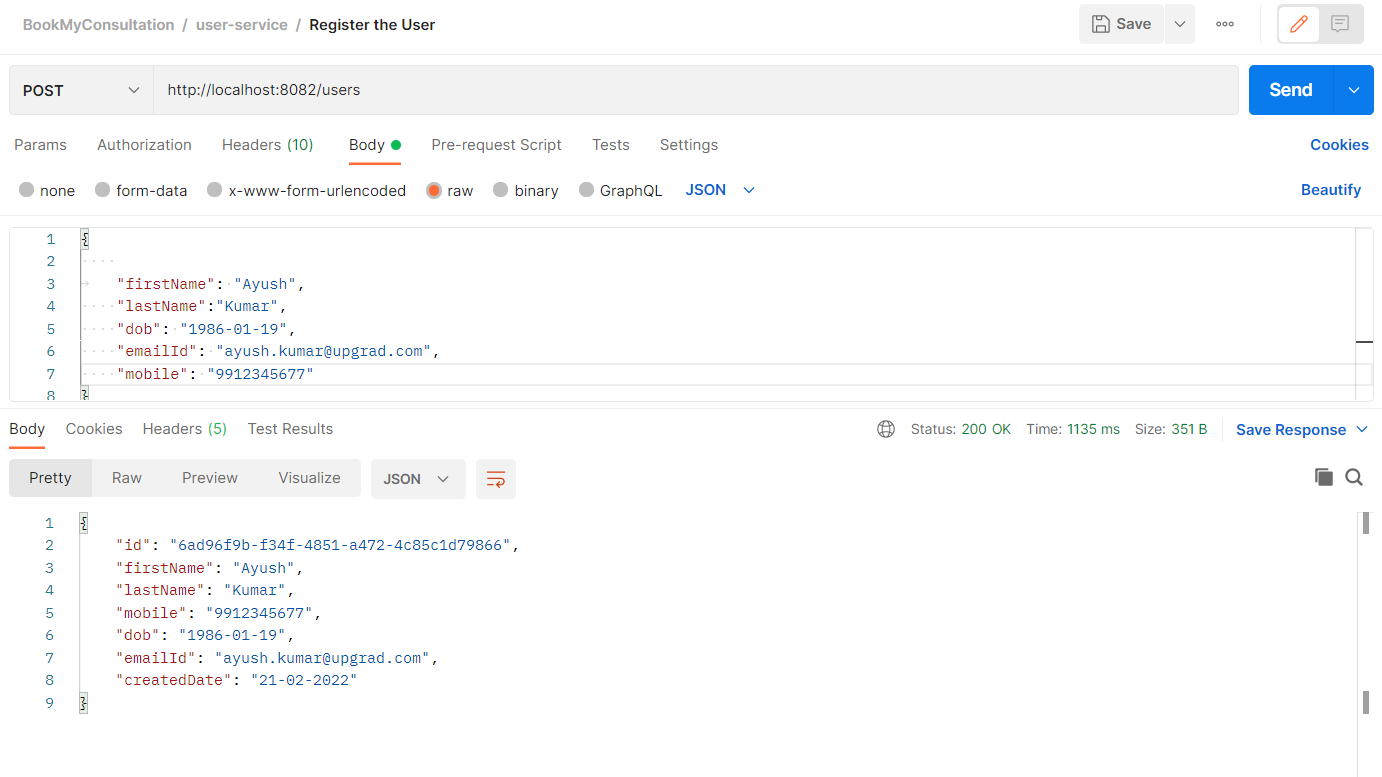
Request Body Ex –

|  |
| --- |
| {      "firstName": "Ayush",      "lastName":"Kumar",      "dob": "1986-01-19",      "emailId": "ayush.kumar@upgrad.com",      "mobile": "9912345677"  } |

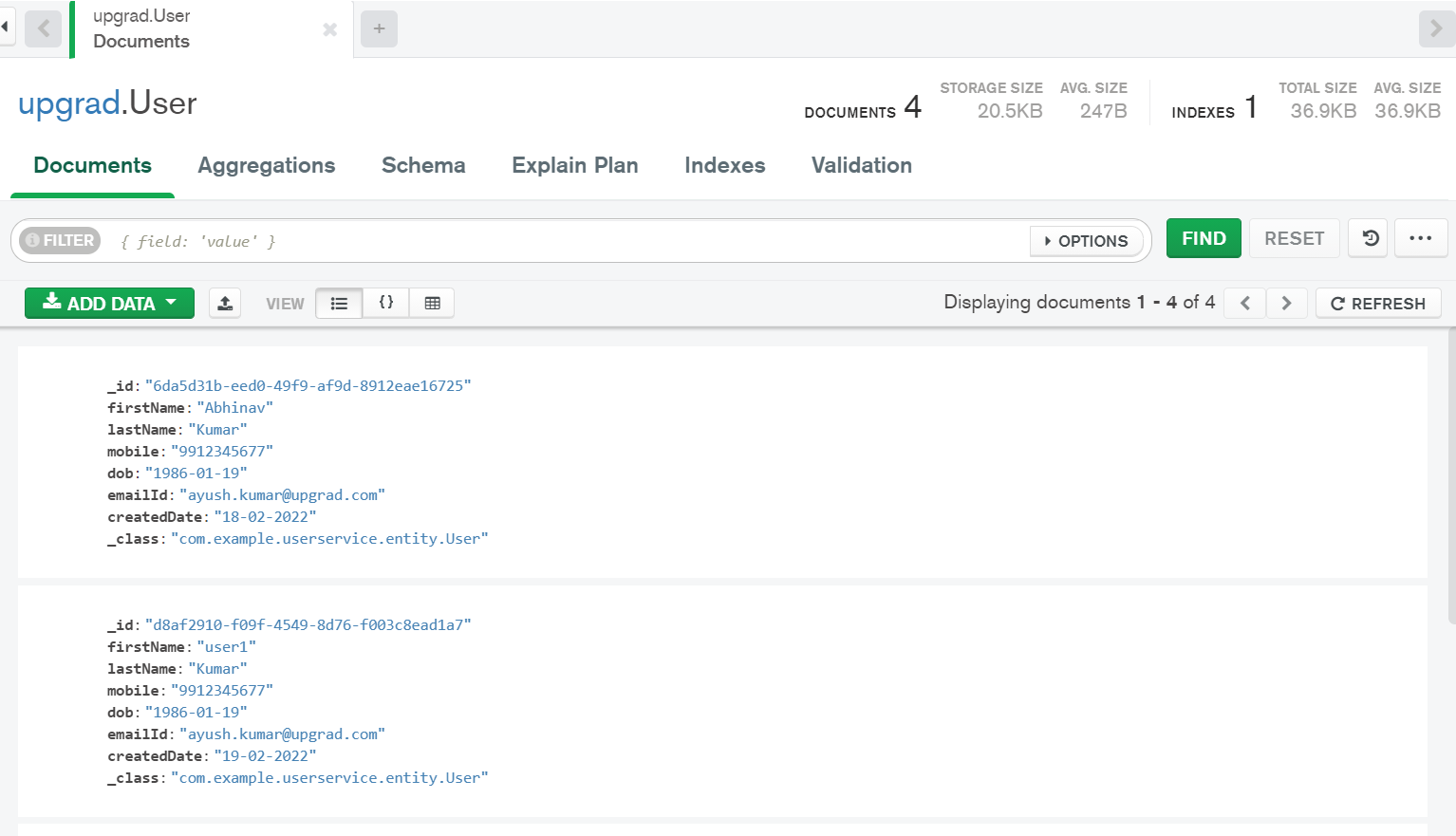
Response Body Ex –

|  |
| --- |
| {      "id": "6ad96f9b-f34f-4851-a472-4c85c1d79866",      "firstName": "Ayush",      "lastName": "Kumar",      "mobile": "9912345677",      "dob": "1986-01-19",      "emailId": "ayush.kumar@upgrad.com",      "createdDate": "21-02-2022"  } |

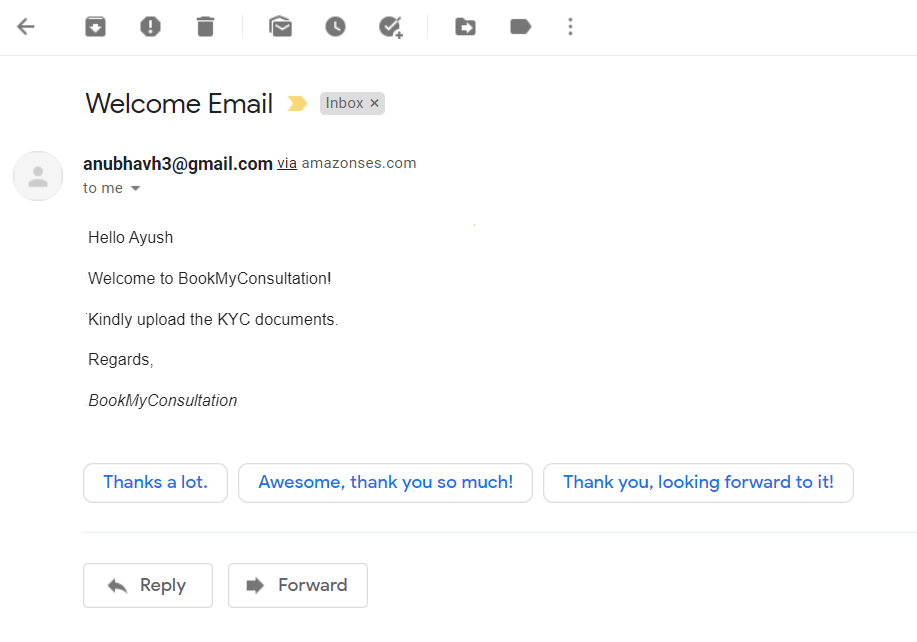
Calling the endpoint from POSTMAN -



Mongo Collection 'User' -



Email received by the new user-



## Endpoint-2: Fetch user information

**POST** localhost:8082/users/{userID}

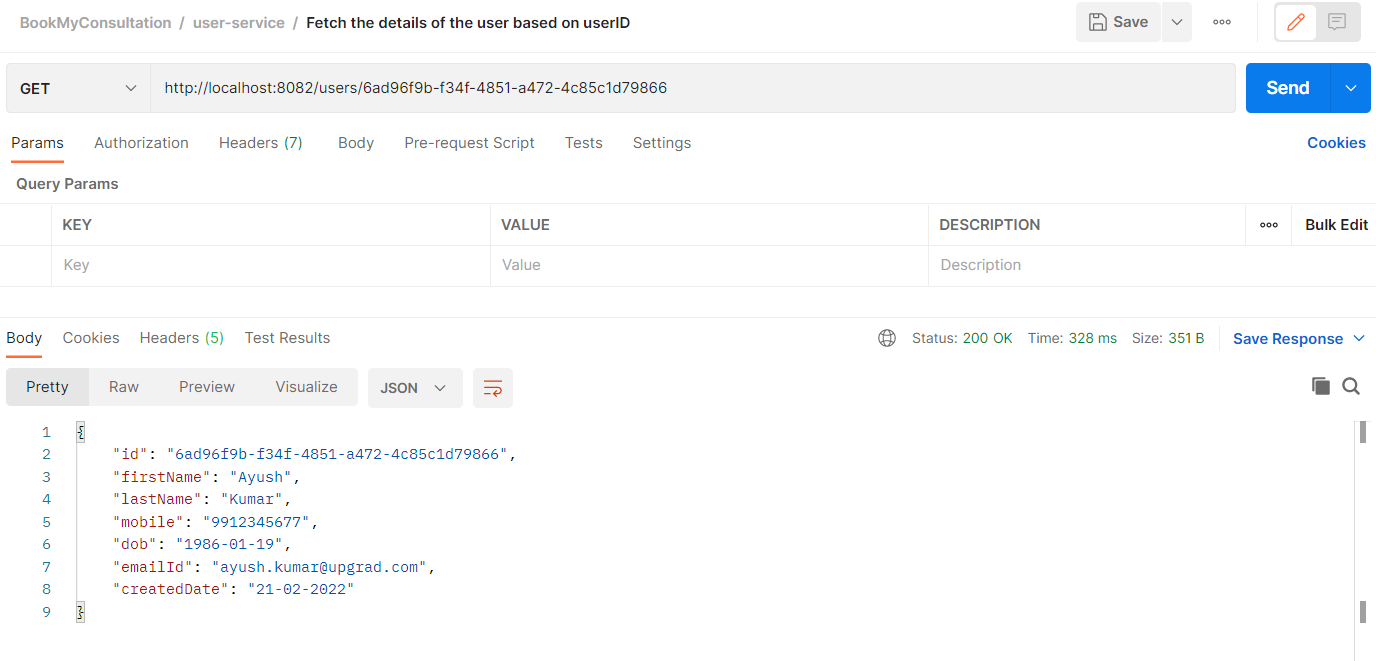
Request Ex –

|  |
| --- |
| http://localhost:8082/users/6da5d31b-eed0-49f9-af9d-8912eae16725 |

Response Body Ex –

|  |
| --- |
| {      "id": "6da5d31b-eed0-49f9-af9d-8912eae16725",      "firstName": "Abhinav",      "lastName": "Kumar",      "mobile": "9912345677",      "dob": "1986-01-19",      "emailId": "ayush.kumar@upgrad.com",      "createdDate": "18-02-2022"  } |

Calling the endpoint from POSTMAN -



## Endpoint-3: Upload user documents

**POST** localhost:8082/users/{userId}/document

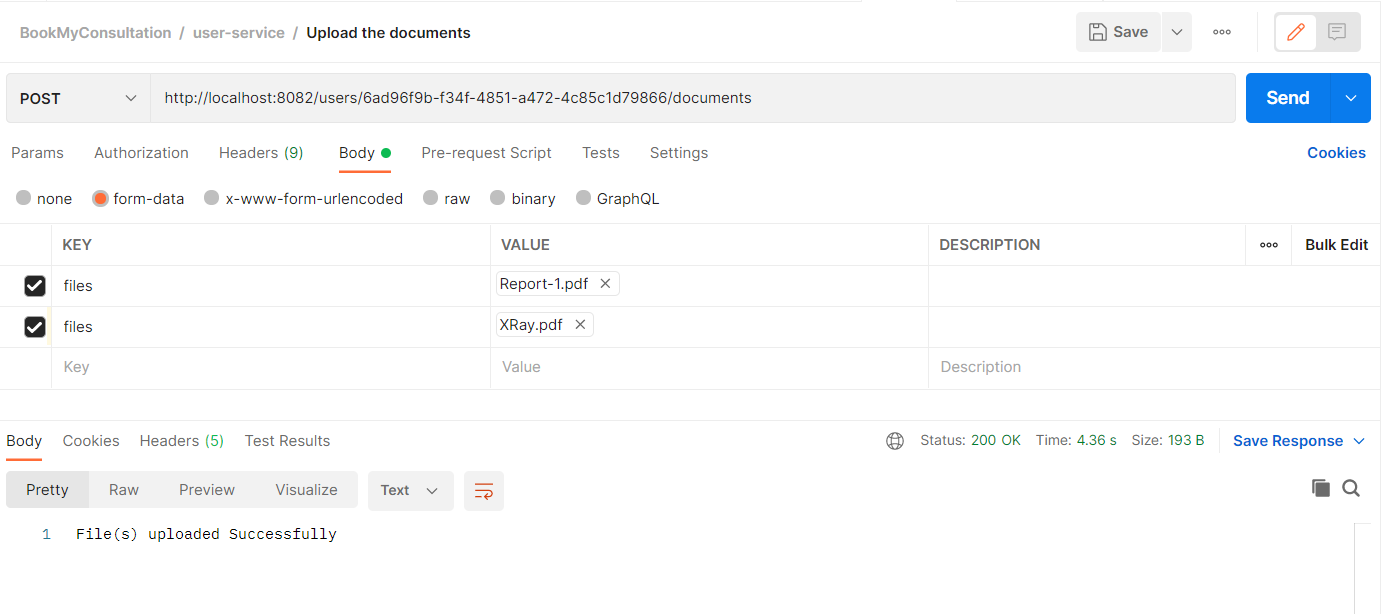
Request Body Ex –

|  |
| --- |
|  |

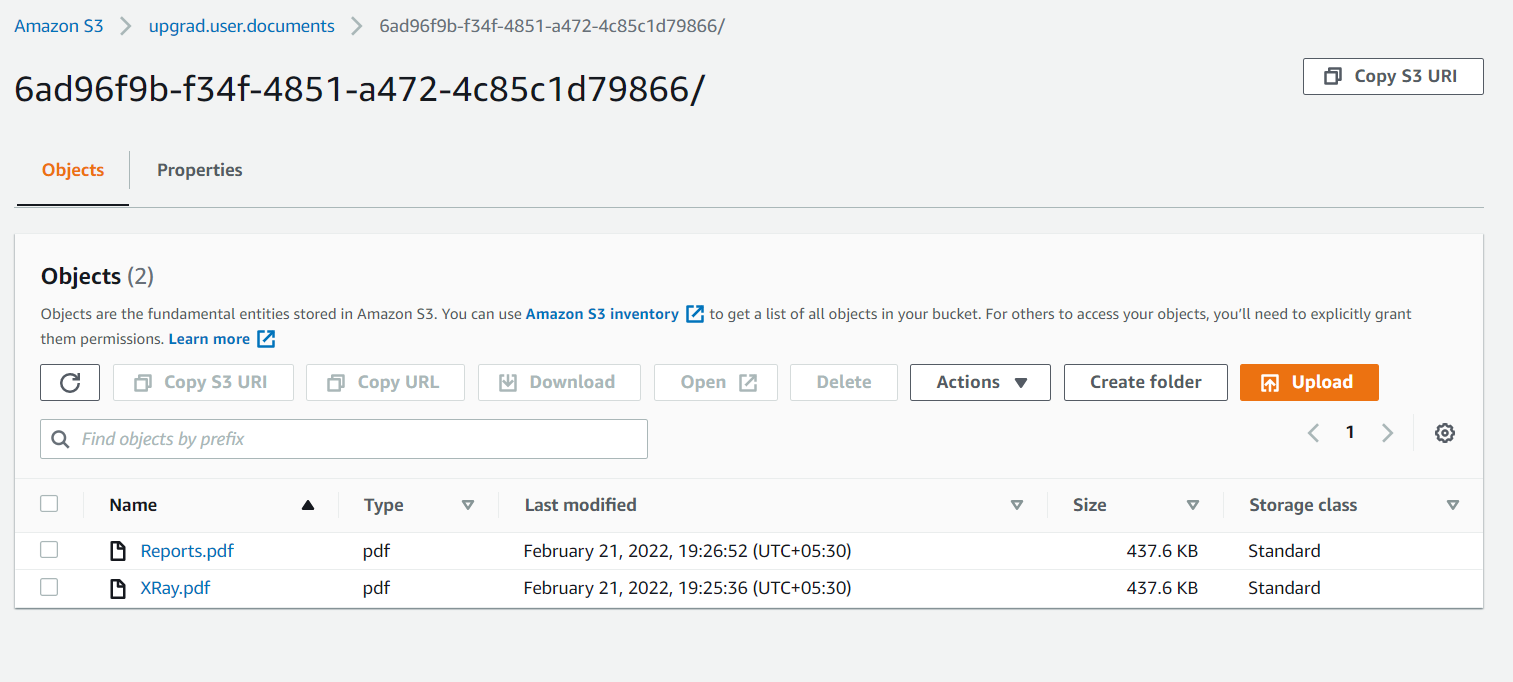
Response Body Ex –

|  |
| --- |
| File(s) uploaded Successfully. |

Calling the endpoint from POSTMAN -



S3 bucket updated with the uploaded documents -



# **Appointment Service**

## Endpoint-1: Update availability of the doctors.

**POST** localhost:8083/doctor/{doctorId}/availability

Content-Type application/json

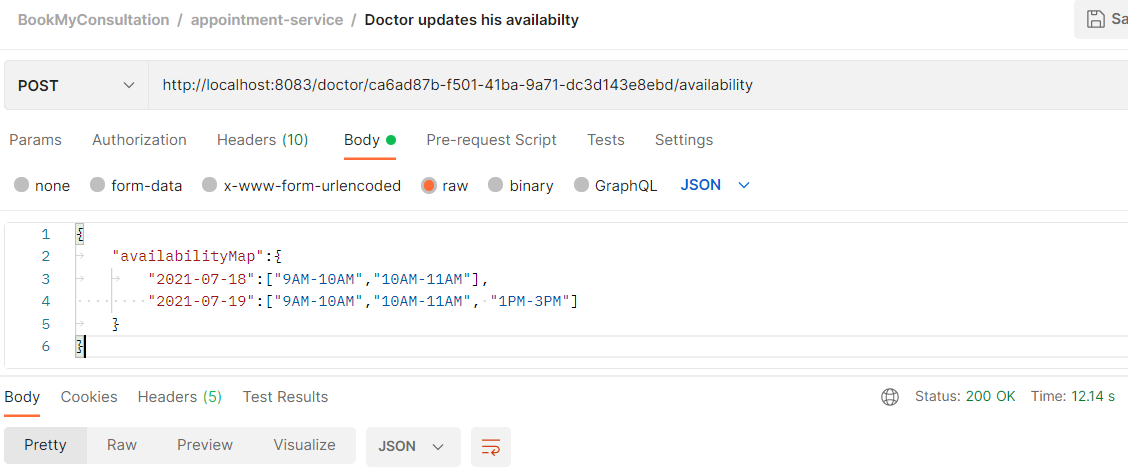
Request Body Ex –

|  |
| --- |
| {      "availabilityMap":{          "2021-07-18":["9AM-10AM","10AM-11AM"],          "2021-07-19":["9AM-10AM","10AM-11AM", "1PM-3PM"]      }  } |

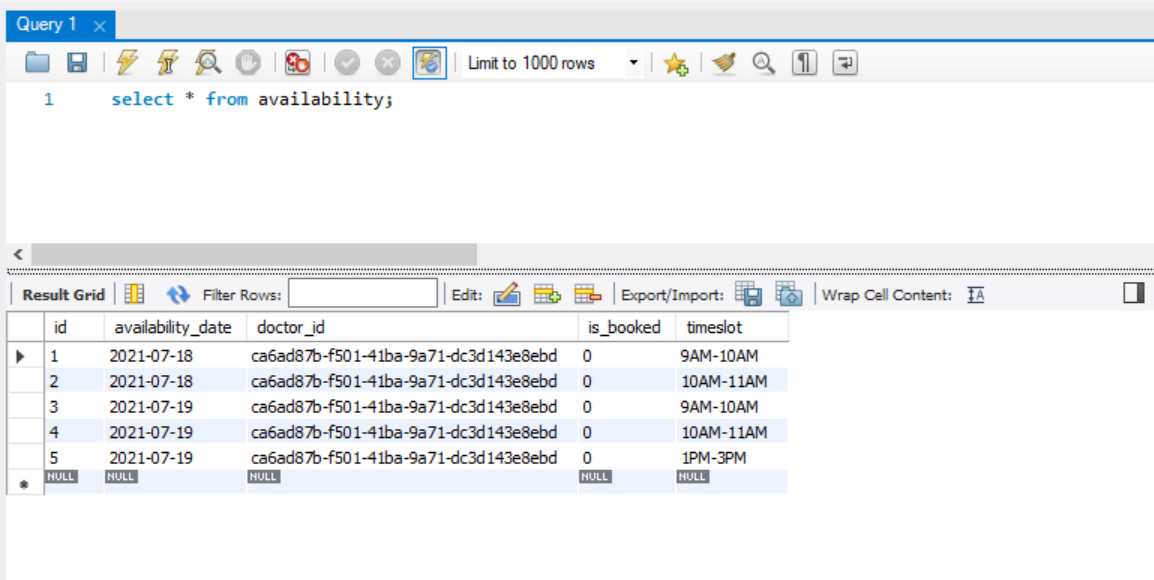
Response -

|  |
| --- |
| HTTP Status 200 OK |

Calling the endpoint from POSTMAN -



Records stored in RDS –



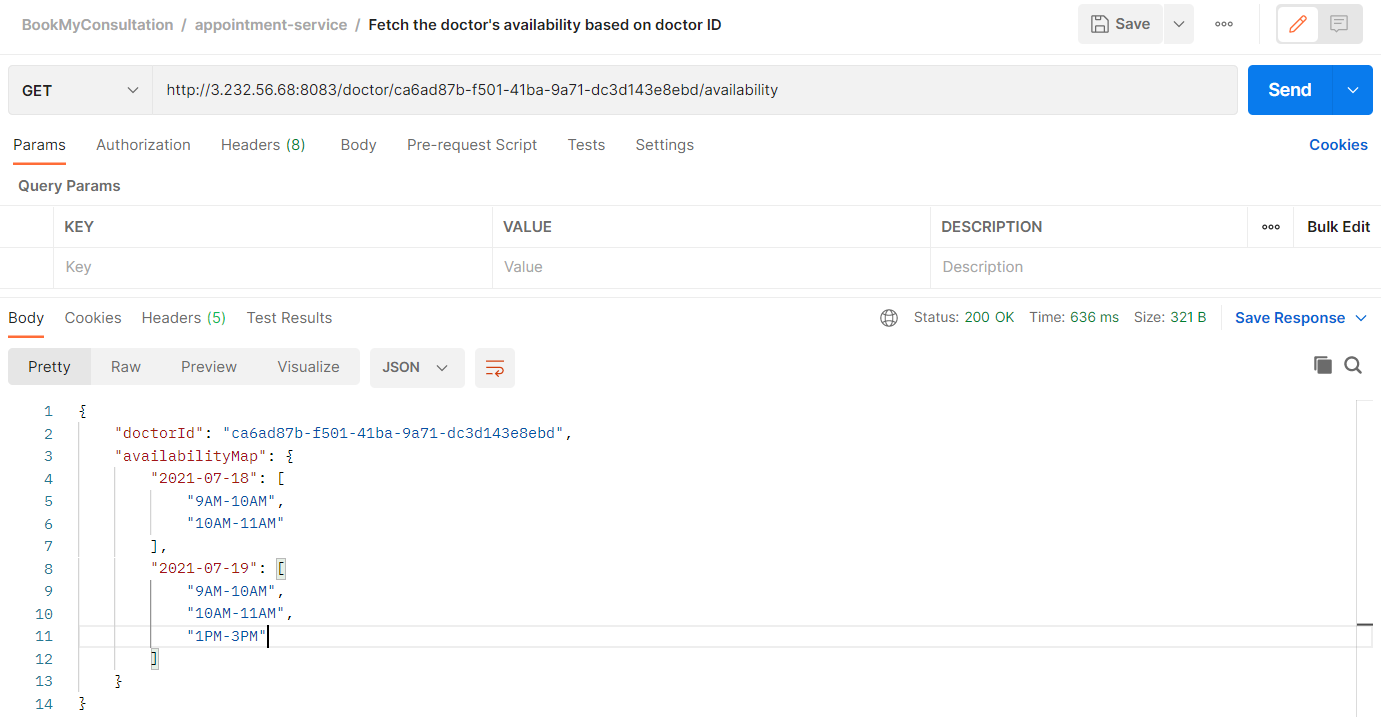
## Endpoint-2: Fetch doctor's availability.

**GET** localhost:8083/doctor/{doctorId}/availability

Response Ex -

|  |
| --- |
| {      "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",      "availabilityMap": {          "2021-07-18": [              "9AM-10AM",              "10AM-11AM"          ],          "2021-07-19": [              "9AM-10AM",              "10AM-11AM",              "1PM-3PM"          ]      }  } |

Calling the endpoint from POSTMAN -



## Endpoint-3: Book Appointment

**POST** localhost:8083/appointments

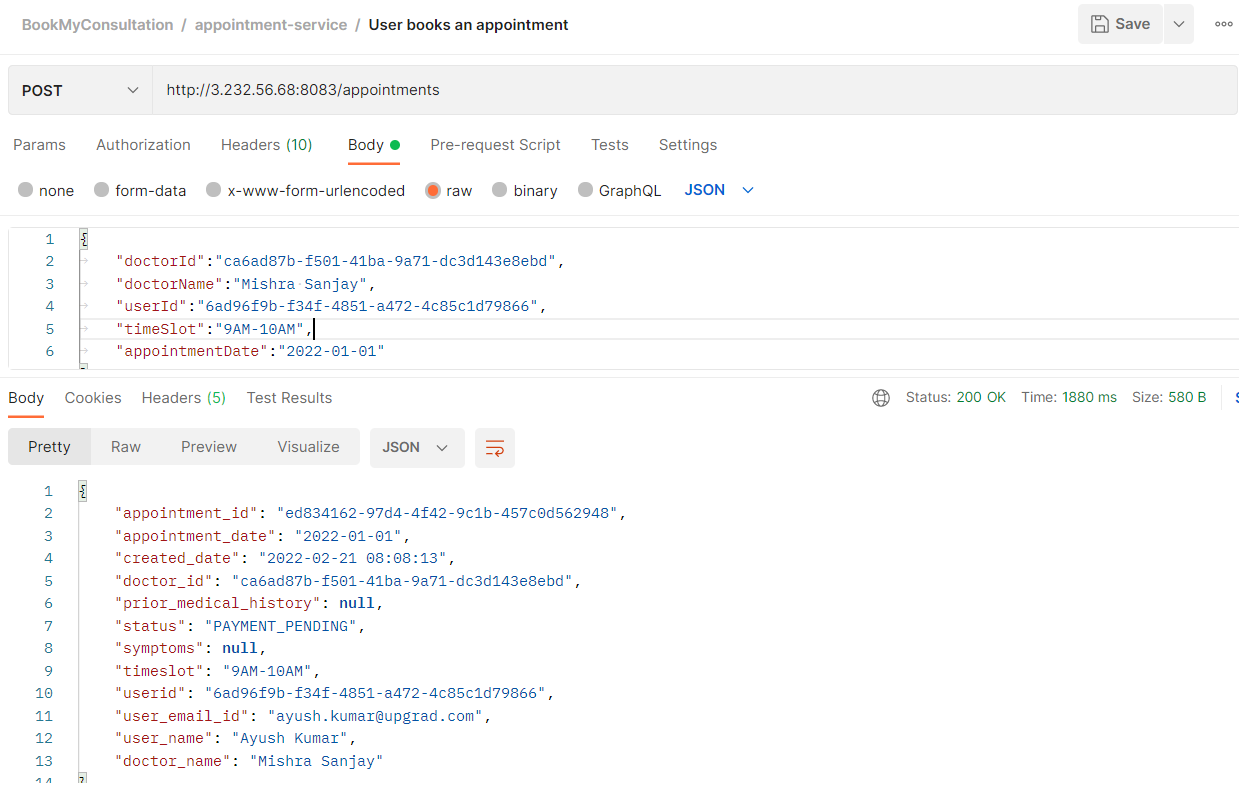
Request Body Ex –

|  |
| --- |
| {      "doctorId":"4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",      "doctorName":"Kumar Shri",      "userId":"6ad96f9b-f34f-4851-a472-4c85c1d79866",      "timeSlot":"01PM-02PM",      "appointmentDate":"2022-04-02"  } |

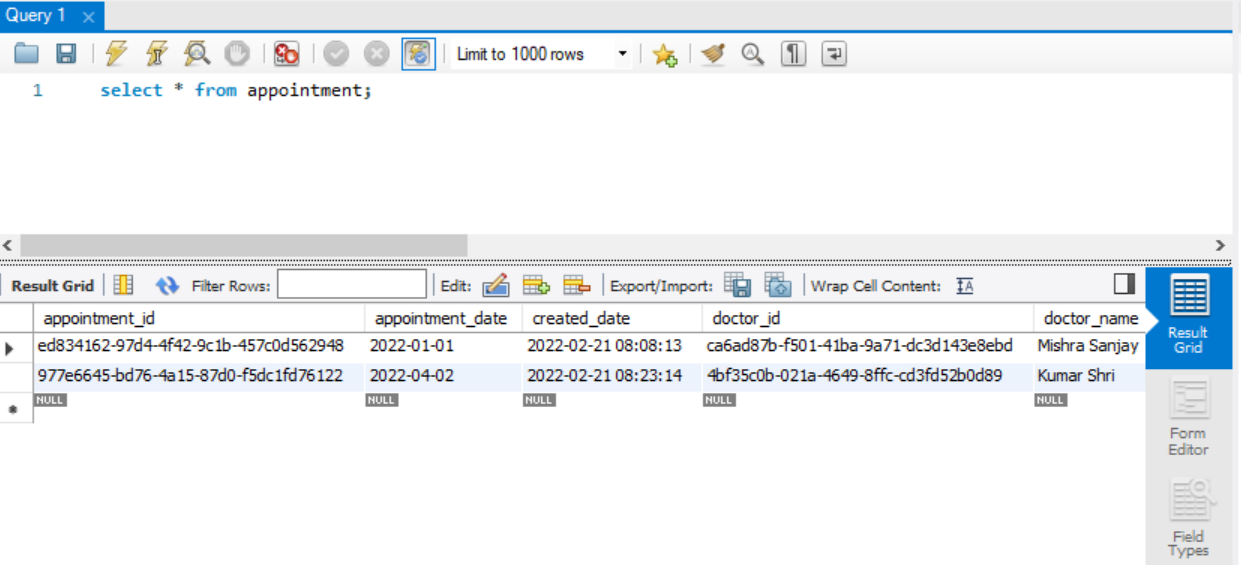
Response Ex -

|  |
| --- |
| {      "appointment\_id": "977e6645-bd76-4a15-87d0-f5dc1fd76122",      "appointment\_date": "2022-04-02",      "created\_date": "2022-02-21 08:23:14",      "doctor\_id": "4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",      "prior\_medical\_history": **null**,      "status": "PAYMENT\_PENDING",      "symptoms": **null**,      "timeslot": "01PM-02PM",      "userid": "6ad96f9b-f34f-4851-a472-4c85c1d79866",      "user\_email\_id": "ayush.kumar@upgrad.com",      "user\_name": "Ayush Kumar",      "doctor\_name": "Kumar Shri"  } |

Calling the endpoint from POSTMAN -



Records stored in RDS –



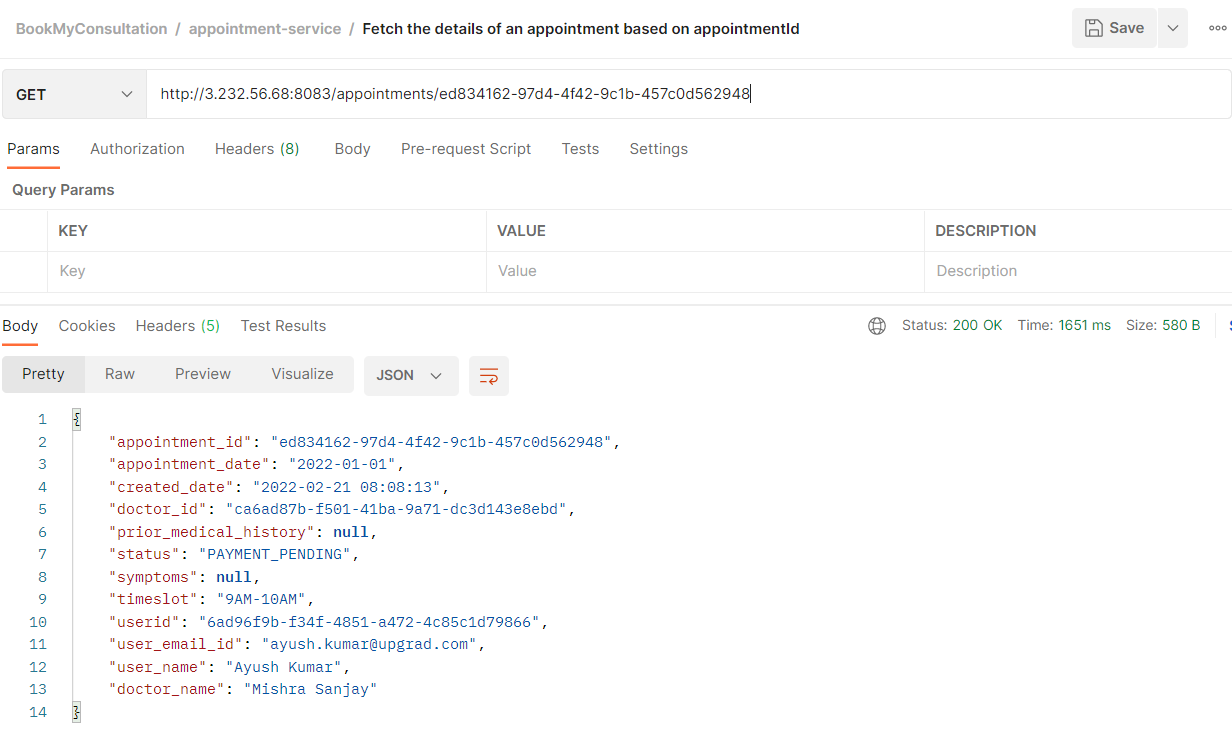
## Endpoint-4: Fetch Appointment details

**GET** localhost:8083 /appointments/{appointmentId}

Response Ex -

|  |
| --- |
| {      "appointment\_id": "ed834162-97d4-4f42-9c1b-457c0d562948",      "appointment\_date": "2022-01-01",      "created\_date": "2022-02-21 08:08:13",      "doctor\_id": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",      "prior\_medical\_history": **null**,      "status": "PAYMENT\_PENDING",      "symptoms": **null**,      "timeslot": "9AM-10AM",      "userid": "6ad96f9b-f34f-4851-a472-4c85c1d79866",      "user\_email\_id": "ayush.kumar@upgrad.com",      "user\_name": "Ayush Kumar",      "doctor\_name": "Mishra Sanjay"  } |

Calling the endpoint from POSTMAN -



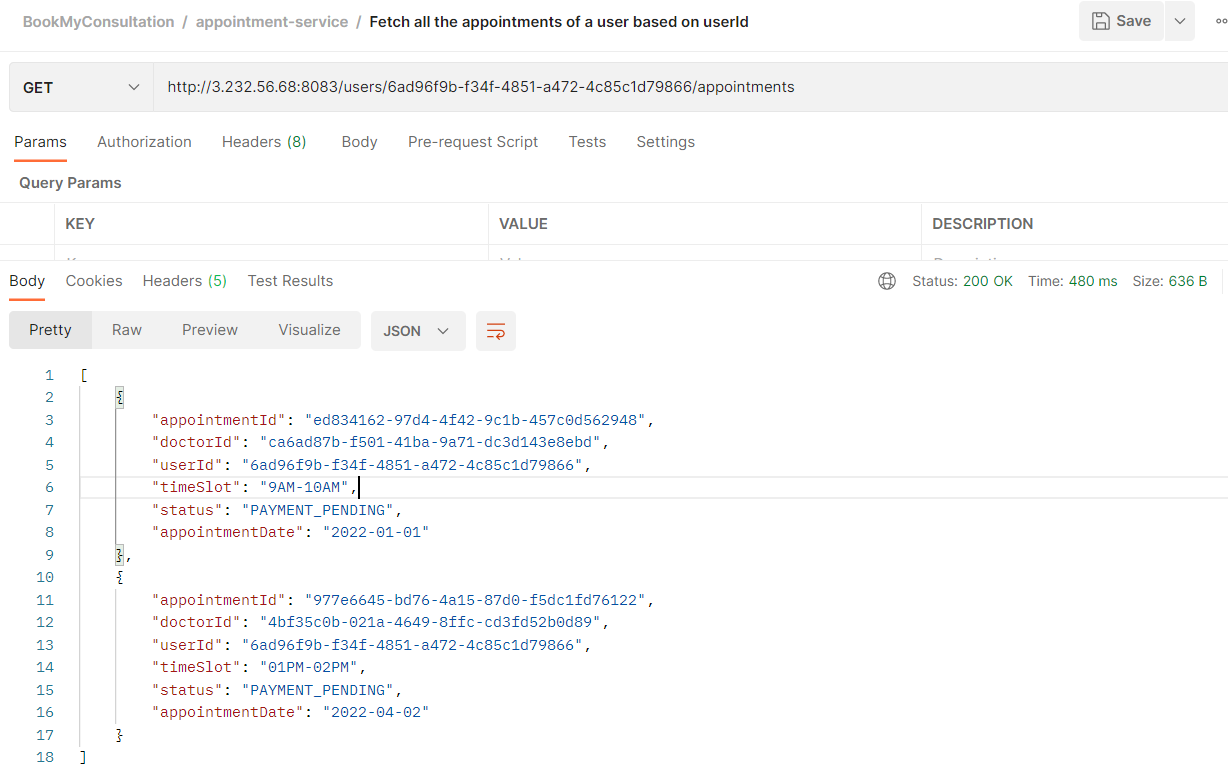
## Endpoint-5: Fetch Appointments by userId

**GET** localhost:8083 /users/{userId}/appointments

Response Ex -

|  |
| --- |
| [      {          "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",          "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",          "userId": "6ad96f9b-f34f-4851-a472-4c85c1d79866",          "timeSlot": "9AM-10AM",          "status": "PAYMENT\_PENDING",          "appointmentDate": "2022-01-01"      },      {          "appointmentId": "977e6645-bd76-4a15-87d0-f5dc1fd76122",          "doctorId": "4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",          "userId": "6ad96f9b-f34f-4851-a472-4c85c1d79866",          "timeSlot": "01PM-02PM",          "status": "PAYMENT\_PENDING",          "appointmentDate": "2022-04-02"      }  ] |

Calling the endpoint from POSTMAN -



## Endpoint-6: Send prescription

**GET** localhost:8083 /users/{userId}/appointments

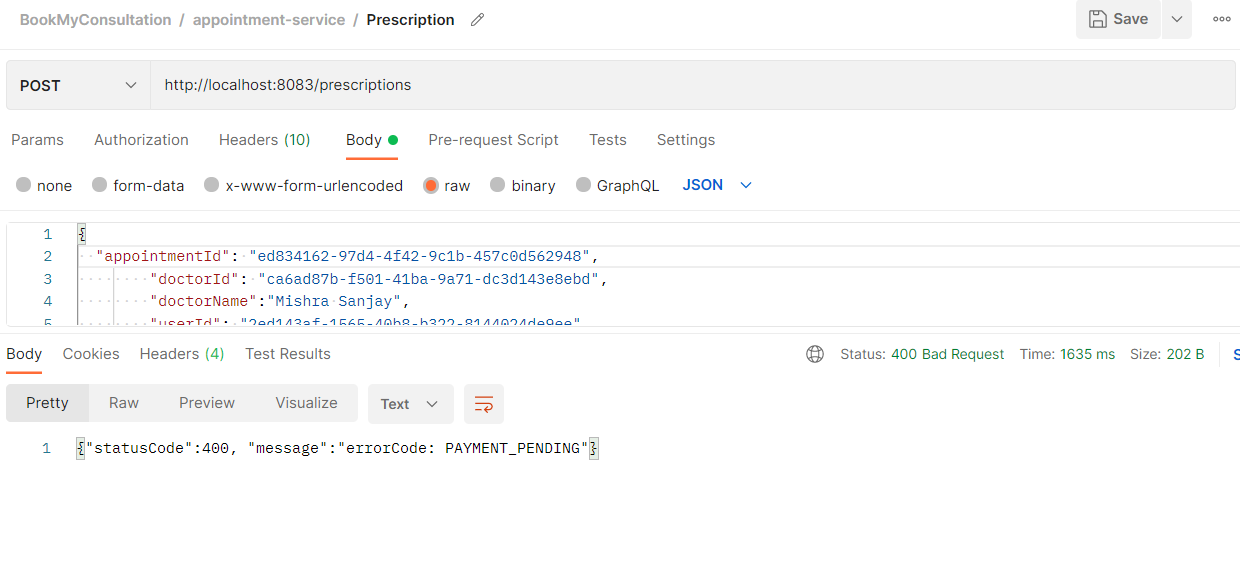
Request Body Ex -

|  |
| --- |
| {    "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",          "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",          "doctorName":"Mishra Sanjay",          "userId": "2ed143af-1565-40b8-b322-8144024de9ee",          "diagnosis":" Teeth Cavity",          "medicineList":[            {              "name":"Calpol",              "type":"Tablet",              "dosage":"1 week",              "duration":"1 week",              "frequency":"3 times a day",              "remarks":"after food"            },            {              "name":"PainKill",              "type":"Syrup",              "dosage":"1 week",              "duration":"1 week",              "frequency":"3 times a day",              "remarks":"after food"            }            ]  } |

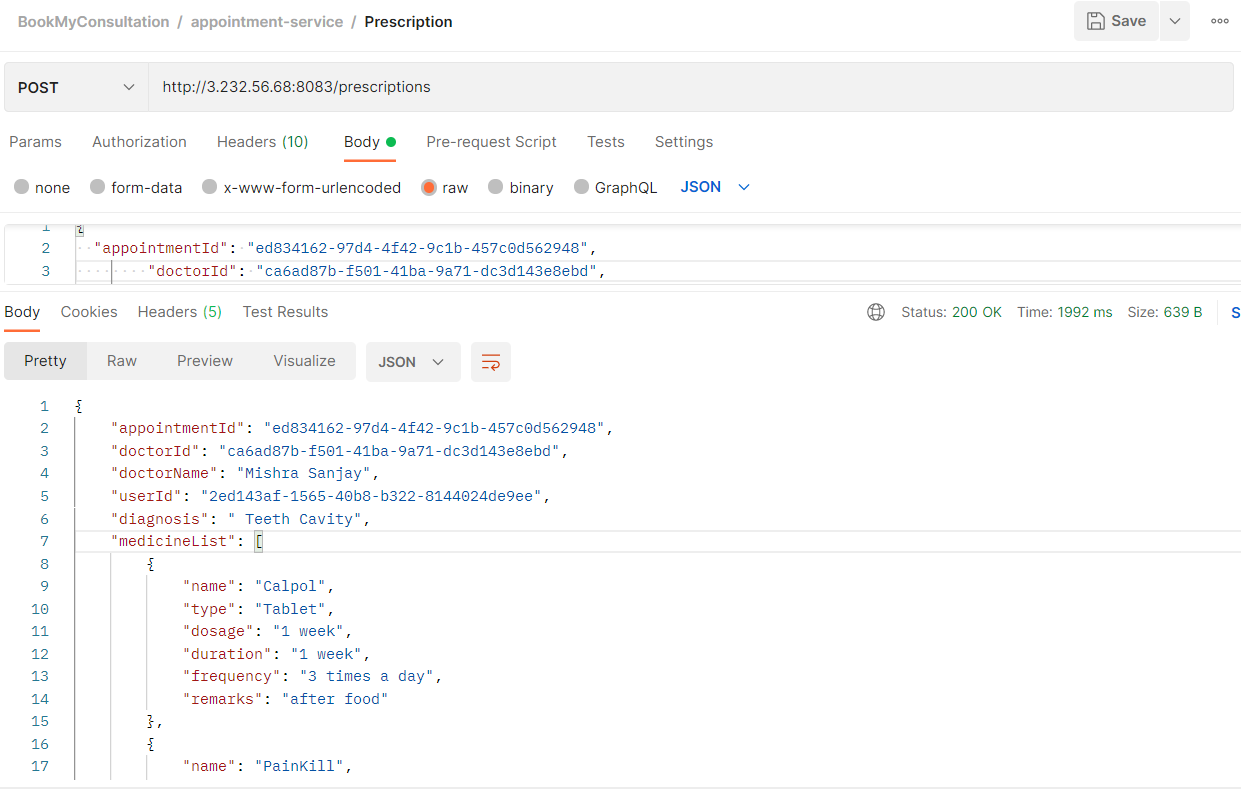
Response Ex -

|  |
| --- |
| {      "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",      "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",      "doctorName": "Mishra Sanjay",      "userId": "2ed143af-1565-40b8-b322-8144024de9ee",      "diagnosis": " Teeth Cavity",      "medicineList": [          {              "name": "Calpol",              "type": "Tablet",              "dosage": "1 week",              "duration": "1 week",              "frequency": "3 times a day",              "remarks": "after food"          },          {              "name": "PainKill",              "type": "Syrup",              "dosage": "1 week",              "duration": "1 week",              "frequency": "3 times a day",              "remarks": "after food"          }      ]  } |

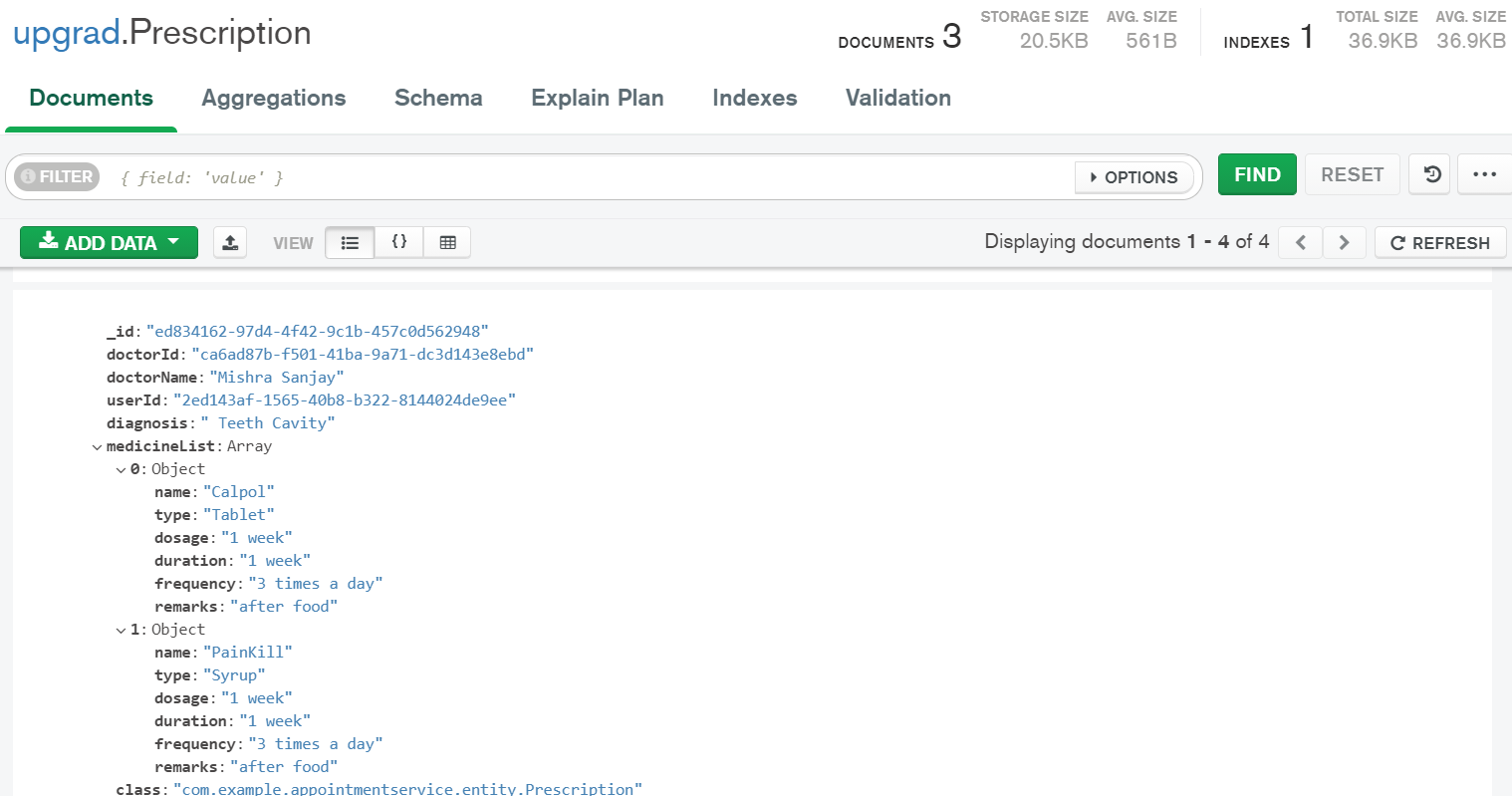
Calling the endpoint from POSTMAN when PAYMENT is pending -



Calling the endpoint from POSTMAN when PAYMENT is confirmed –



Mongo Collection 'Prescription' –



# **Payment Service**

## Endpoint-1: Make payment for appointment.

**POST** localhost:8084/payments?appointmentId=<appointment-id>

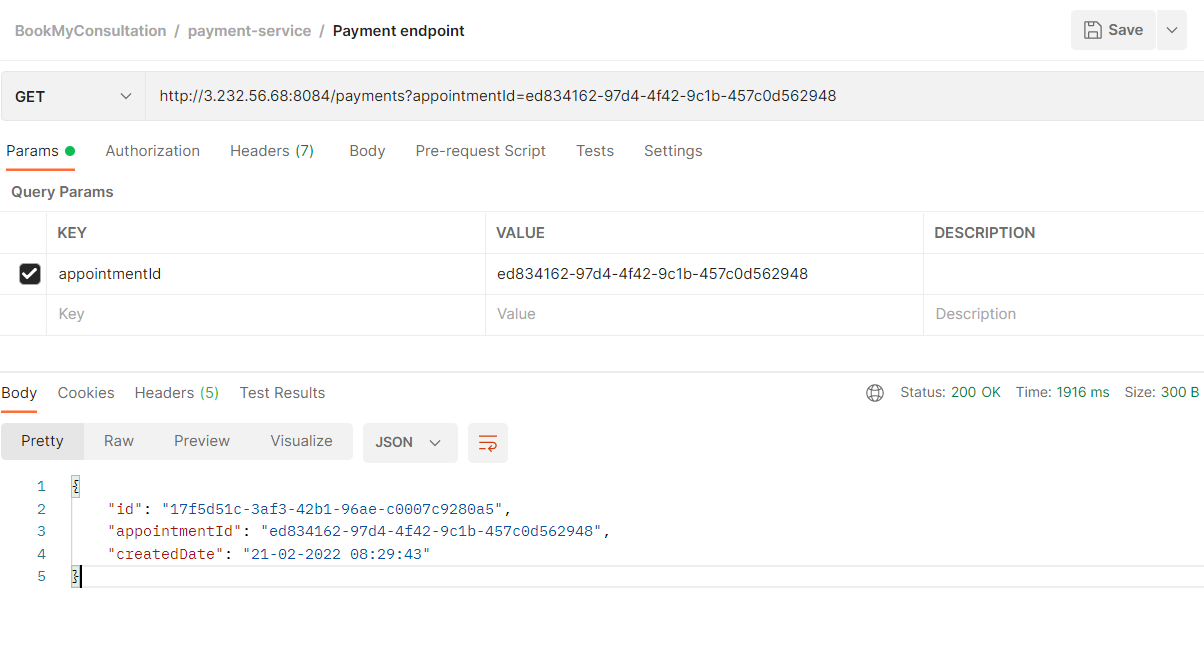
Request Body Ex –

|  |
| --- |
| http://3.232.56.68:8084/payments?appointmentId=ed834162-97d4-4f42-9c1b-457c0d562948 |

Response Ex -

|  |
| --- |
| {      "id": "17f5d51c-3af3-42b1-96ae-c0007c9280a5",      "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",      "createdDate": "21-02-2022 08:29:43"  } |

Calling the endpoint from POSTMAN –



# **Rating Service**

## Endpoint-1: Rate Doctors.

**POST** localhost:8085/ratings

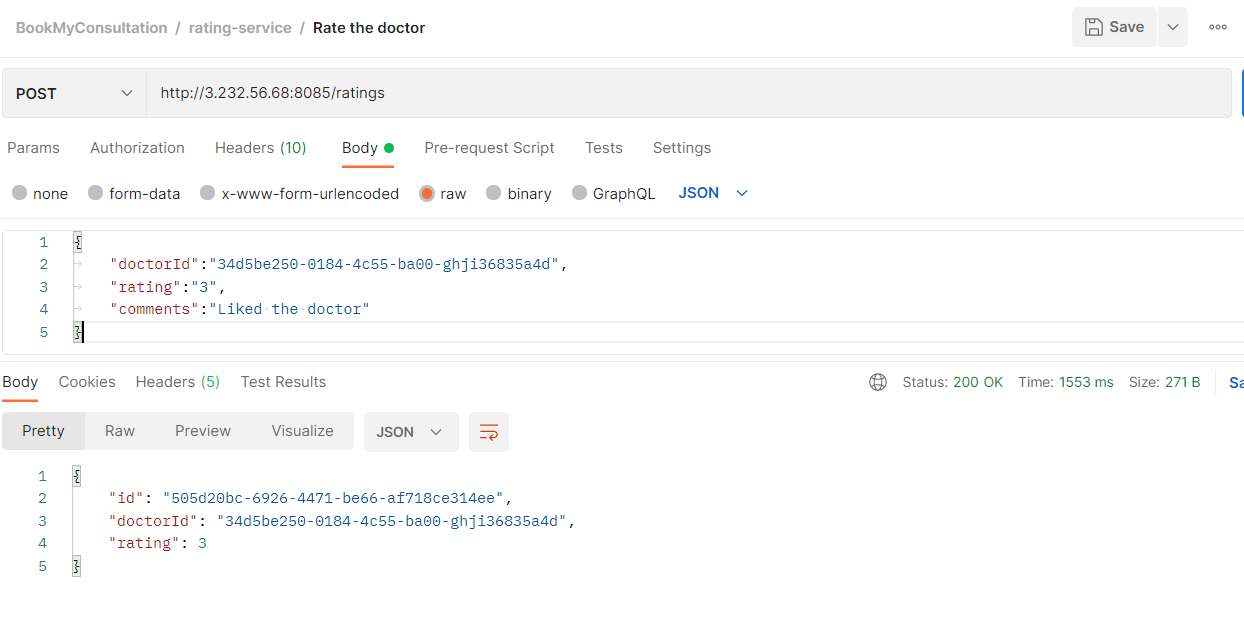
Request Body Ex –

|  |
| --- |
| {      "doctorId":"34d5be250-0184-4c55-ba00-ghji36835a4d",      "rating":"3",      "comments":"Liked the doctor"  } |

Response Ex -

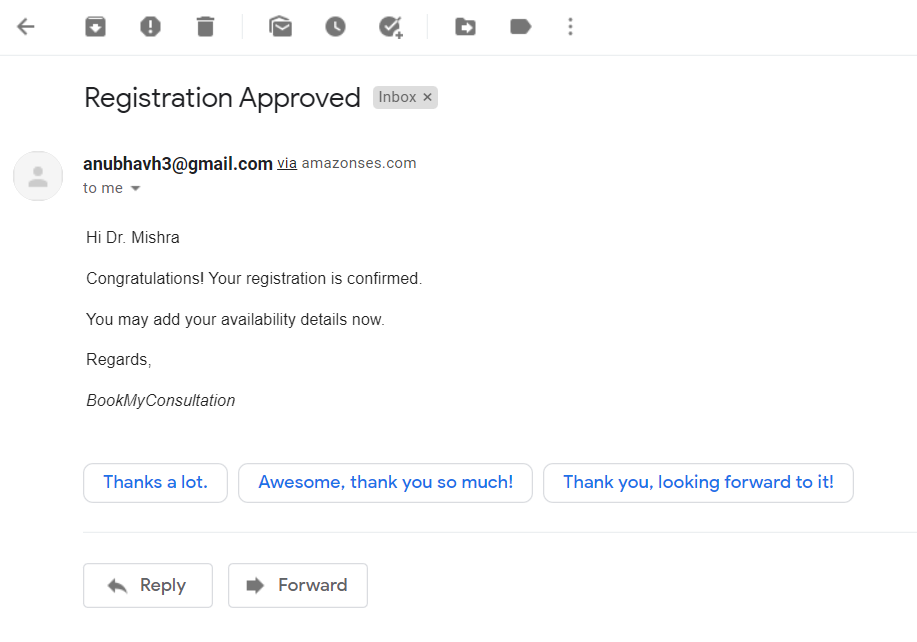
|  |
| --- |
| {      "id": "505d20bc-6926-4471-be66-af718ce314ee",      "doctorId": "34d5be250-0184-4c55-ba00-ghji36835a4d",      "rating": 3  } |

Calling the endpoint from POSTMAN –

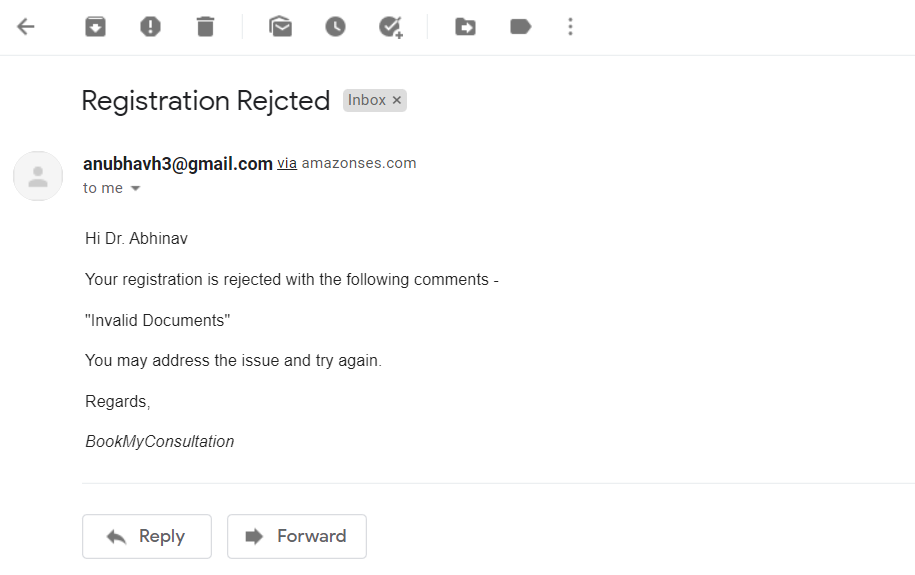


# **Notification Service**

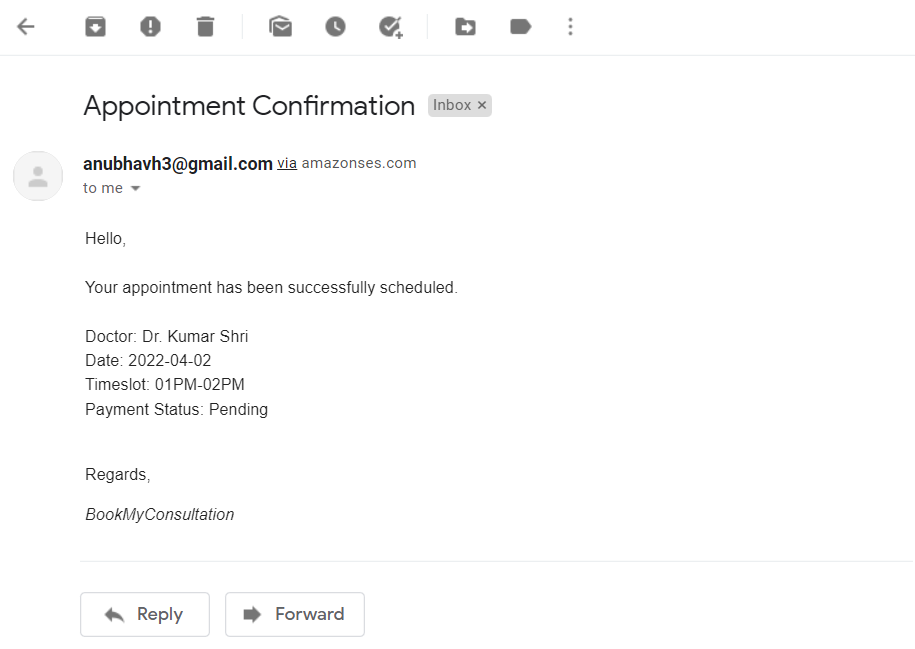
## Doctor Registration Approval



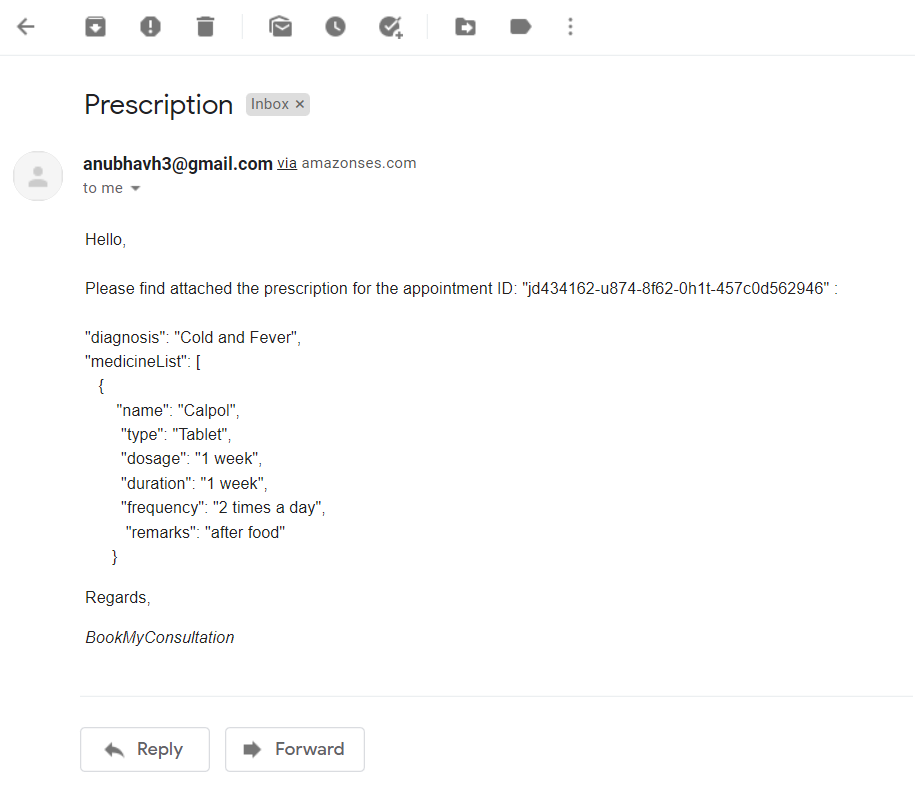
## Doctor Registration Rejection



## Appointment Confirmation



## Prescription



# **API Gateway**

An API Gateway has been implemented to act as reverse proxy. Instead of calling the individual API's, one can call the API Gateway like following-

http://<EC2host|localhost>:8080/doctorsvc/ . . .

http://<EC2host|localhost>:8080/usersvc/ . . .

http://<EC2host|localhost>:8080/appointmentsvc/ . . .

http://<EC2host|localhost>:8080/paymentsvc/ . . .

http://<EC2host|localhost>:8080/ratingsvc/ . . .

# **Security**

Token generation and validation has not been implemented in this submission.

# **Future Enhancements**

- Add error and exception handling at all the possible points

- Implement a Configuration Server

\*\*\* END \*\*\*