

Capstone Project - BookMyConsultation

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

Table of content

1. Environment Setup	3
1.1. VPC Setup	3
1.2. EC2 Setup	3
1.3. RDS Setup	5
2. Steps to Deploy and Run (Without Docker)	6
3. Steps to Deploy and Run (With Docker)	7
4. Eureka Server	8
5. Doctor-Onboarding Service	9
5.1. Endpoint-1: Collect doctor information	9
5.2. Endpoint-2: Upload doctor documents.....	11
5.3. Endpoint-3: Approve doctor registration.....	12
5.4. Endpoint-4: Reject doctor registration	13
5.5. Endpoint-5: Return list of doctors.....	14
5.5.1. Scenario-1: The status is pending	14
5.5.2. Scenario-2: The status is Active.....	14
5.5.3. Scenario-2: Based on speciality.....	15
5.6. Endpoint-6: Return doctors based on doctor-ID.....	15
6. User-Onboarding Service	16
6.1. Endpoint-1: Collect user information.....	16
6.2. Endpoint-2: Fetch user information.....	18
6.3. Endpoint-3: Upload user documents	19
7. Appointment Service.....	20
7.1. Endpoint-1: Update availability of the doctors.	20
7.2. Endpoint-2: Fetch doctor's availability.....	21
7.3. Endpoint-3: Book Appointment	22
7.4. Endpoint-4: Fetch Appointment details.....	23
7.5. Endpoint-5: Fetch Appointments by userId	24
7.6. Endpoint-6: Send prescription	25

8. Payment Service	27
8.1. Endpoint-1: Make payment for appointment.....	27
9. Rating Service.....	28
9.1. Endpoint-1: Rate Doctors.....	28
10. Notification Service	29
10.1. Doctor Registration Approval.....	29
10.2. Doctor Registration Rejection	29
10.3. Appointment Confirmation.....	30
10.4. Prescription	30
11. API Gateway	31
12. Security.....	31
13. Future Enhancements	31

1. Environment Setup

1.1. VPC Setup

The screenshot shows the AWS VPC console for the VPC `vpc-04cab8452bd04d054`. The details are as follows:

Property	Value
VPC ID	vpc-04cab8452bd04d054
Tenancy	Default
Default VPC	No
Route 53 Resolver DNS Firewall rule groups	-
State	Available
DHCP options set	dopt-0a45ed86235d52d17
IPv4 CIDR	10.0.0.0/16
Owner ID	054244894115
DNS hostnames	Enabled
Main route table	rtb-0a7d250e0eb798614
IPv6 pool	-
DNS resolution	Enabled
Main network ACL	acl-026b0d7ecc13b55b3
IPv6 CIDR (Network border group)	-

1.2. 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.

The screenshot shows the AWS EC2 console. The top section displays a list of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
booking-kafka	i-0551aeb431b2d5bb6	Running	t2.medium	-	No alarms	us-east-1b	ec2-54-146-152-189.co...	54.146.1...
booking-docker-mongo	i-0a4c1ea37aadac099	Running	t2.micro	Initializing	No alarms	us-east-1b	ec2-3-239-160-1.comp...	3.239.16...

The bottom section shows the details for the instance `i-0a4c1ea37aadac099 (booking-docker-mongo)`:

Property	Value
Instance ID	i-0a4c1ea37aadac099 (booking-docker-mongo)
IPv6 address	-
Hostname type	IP name: ip-10-0-0-242.ec2.internal
Public IPv4 address	3.239.160.1 open address
Instance state	Running
Private IP DNS name (IPv4 only)	ip-10-0-0-242.ec2.internal
Private IPv4 addresses	10.0.0.242
Public IPv4 DNS	ec2-3-239-160-1.compute-1.amazonaws.com open address
Answer private resource DNS name	IPv4 (A)

EC2 (first instance) Security Group Inbound Rules -

Inbound rules Info						
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-015b6c84b7b022c46	Custom TCP ▼	TCP	27017	Custom ▼	165.225.124.231/32 ✕	Delete
sgr-04ff0c1f93ed04635	SSH ▼	TCP	22	Custom ▼	165.225.124.216/32 ✕	Delete
sgr-0f6759a8efef7674a	SSH ▼	TCP	22	Custom ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8080	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8081	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8082	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8083	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8084	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8085	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8086	Anywh... ▼	0.0.0.0/0 ✕	Delete
-	Custom TCP ▼	TCP	8761	Anywh... ▼	0.0.0.0/0 ✕	Delete

EC2 (second instance) Security Group Inbound Rules -

Inbound rules Info						
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-08313fbce3be82110	HTTP ▼	TCP	80	My IP ▼	165.225.124.220/32 ✕	Delete
sgr-0ba07b99f6e264549	SSH ▼	TCP	22	My IP ▼	165.225.124.220/32 ✕	Delete
sgr-003dabcb3e7cf5c31	Custom TCP ▼	TCP	9092	Custom ▼	0.0.0.0/0 ✕	Delete
sgr-0ad71e6c5687f8da2	HTTPS ▼	TCP	443	My IP ▼	165.225.124.220/32 ✕	Delete
sgr-008b97535b7f0f047	Custom TCP ▼	TCP	2181	Custom ▼	0.0.0.0/0 ✕	Delete

1.3. RDS Setup

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

The screenshot shows the AWS Management Console interface for an Amazon RDS instance named 'consultation'. The breadcrumb navigation is 'RDS > Databases > sweethome'. The instance is in the 'Available' state. The 'Summary' tab is selected, showing details like DB identifier, CPU usage, Status, Class, Role, Current activity, Engine, and Region & AZ. Below the summary, the 'Connectivity & security' tab is active, displaying 'Endpoint & port', 'Networking', and 'Security' sections. The 'Endpoint & port' section shows the endpoint 'consultation.ce1r6snv3ls.us-east-1.rds.amazonaws.com' and port '3306'. The 'Networking' section shows 'Availability Zone' as 'us-east-1d' and 'VPC' as 'vpc-0f3f387355ac0a7a0'. The 'Security' section shows 'VPC security groups' as 'default (sg-08578d19c03a8e642)' and 'Public accessibility' as 'Public accessibility'.

Summary			
DB identifier sweethome	CPU 6.23%	Status Available	Class db.t2.micro
Role Instance	Current activity 10 Connections	Engine MySQL Community	Region & AZ us-east-1d

Connectivity & security		
Endpoint & port Endpoint consultation.ce1r6snv3ls.us-east-1.rds.amazonaws.com Port 3306	Networking Availability Zone us-east-1d VPC vpc-0f3f387355ac0a7a0	Security VPC security groups default (sg-08578d19c03a8e642) Active Public accessibility

RDS Security Group Inbound Rules –

The screenshot shows the 'Edit inbound rules' page for a security group. The breadcrumb navigation is 'EC2 > Security Groups > sg-08578d19c03a8e642 - default > Edit inbound rules'. The page title is 'Edit inbound rules' with an 'Info' link. A subtitle states: 'Inbound rules control the incoming traffic that's allowed to reach the instance.' Below this, the 'Inbound rules' section is displayed with a table of rules. The table has columns: 'Security group rule ID', 'Type', 'Protocol', 'Port range', 'Source', and 'Description - optional'. There are three rules listed. The first rule has ID 'sgr-0b2fa81534f0feac5', Type 'MySQL/Aurora', Protocol 'TCP', Port range '3306', Source 'Custom', and Description '165.225.124.240/32'. The second rule has ID 'sgr-0e284a8e309cbb072', Type 'MySQL/Aurora', Protocol 'TCP', Port range '3306', Source 'My IP', and Description 'personal-pc'. The third rule has ID 'sgr-0009f7059264379fd', Type 'All traffic', Protocol 'All', Port range 'All', Source 'Custom', and Description 'sg-08578d19c03a8e642'. At the bottom of the table is an 'Add rule' button. At the bottom of the page are 'Cancel', 'Preview changes', and 'Save rules' buttons.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0b2fa81534f0feac5	MySQL/Aurora	TCP	3306	Custom	165.225.124.240/32
sgr-0e284a8e309cbb072	MySQL/Aurora	TCP	3306	My IP	personal-pc
sgr-0009f7059264379fd	All traffic	All	All	Custom	sg-08578d19c03a8e642

2. Steps to Deploy and Run (Without Docker)

- Start Kafka
 - o Start Kafka server and zookeeper on the second EC2 instance
- Start MongoDBs
 - o 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
```

3. Steps to Deploy and Run (With Docker)

- Start Kafka
 - o Start Kafka server and zookeeper on the second EC2 instance
- Start MongoDBs
 - o 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 -DskipTests
$ sudo docker build -t bookingaap/eurekasvc:1.0.0 .
```

```
$ cd BookMyConsultation/eureka
$ mvn clean install spring-boot:repackage -DskipTests
$ sudo docker build -t bookingaap/doctorsvc:1.0.0 .
```

And so on...

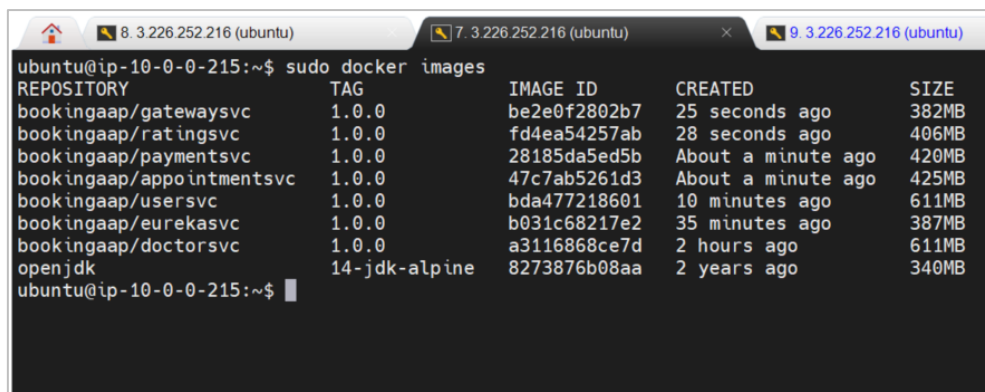
- Start services

```
$ 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 images and deploy all the services at once

```
$ sudo docker-compose up -d
```



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
bookingaap/gatewaysvc	1.0.0	be2e0f2802b7	25 seconds ago	382MB
bookingaap/ratingsvc	1.0.0	fd4ea54257ab	28 seconds ago	406MB
bookingaap/paymentsvc	1.0.0	28185da5ed5b	About a minute ago	420MB
bookingaap/appointmentsvc	1.0.0	47c7ab5261d3	About a minute ago	425MB
bookingaap/usersvc	1.0.0	bda477218601	10 minutes ago	611MB
bookingaap/eurekasvc	1.0.0	b031c68217e2	35 minutes ago	387MB
bookingaap/doctorsvc	1.0.0	a3116868ce7d	2 hours ago	611MB
openjdk	14-jdk-alpine	8273876b08aa	2 years ago	340MB


4. 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>:8761/> we can see the services are up –

 HOME LAST 1000 SINCE STARTUP

System Status

Environment	N/A	Current time	2022-02-21T17:14:12 +0530
Data center	N/A	Uptime	00:08
		Lease expiration enabled	true
		Renews threshold	11
		Renews (last min)	20

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:api-gateway:9191
APPOINTMENT-SERVICE	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:appointment-service:8083
DOCTOR-SERVICE	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:doctor-service:8081
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:payment-service:8084
RATING-SERVICE	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:rating-service:8085
USER-SERVICE	n/a (1)	(1)	UP (1) - QWE0QSMCOE.asiapacific.hpqcorp.net:user-service:8082

General Info

5. Doctor-Onboarding Service

5.1. Endpoint-1: Collect doctor information

POST localhost:8081/doctors

Content-Type application/json

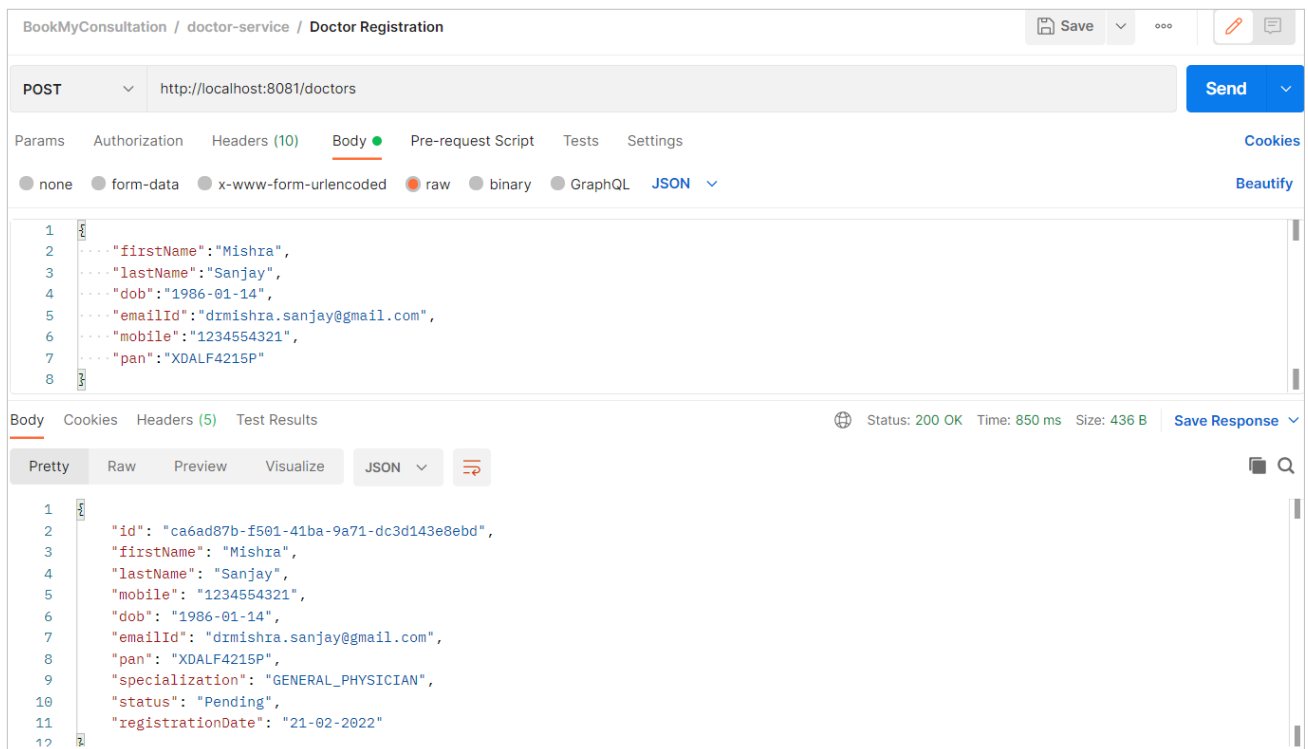
Request Body Ex -

```
{
  "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' -

The screenshot shows the MongoDB Compass interface for the 'upgrad.Doctor' collection. The top bar indicates 2 documents, 20.5KB storage size, 420B avg. size, 1 index, 20.5KB total size, and 20.5KB avg. size. The 'Documents' tab is active, showing a filter bar with the query '{ field: 'value' }'. Below the filter bar, there are buttons for 'ADD DATA', 'VIEW', and 'REFRESH'. The main area displays two documents in a JSON format.

```
{
  "_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": "Pending",
  "registrationDate": "19-02-2022",
  "approvedBy": "Pending",
  "approverComments": "Pending",
  "verificationDate": "Pending",
  "_class": "com.example.doctorservice.entity.Doctor"
}
```

```
{
  "_id": "4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",
  "firstName": "Mishra",
  "lastName": "Sanjay",
  "mobile": "1234554321",
  "dob": "1986-01-14"
}
```

Email received by the new doctor-

The screenshot shows a welcome email interface. The header includes the title 'Welcome Email' and a status 'Inbox x'. The sender is 'anubhavh3@gmail.com via amazonses.com' and the recipient is 'to me'. The email body contains the following text:

Hello Dr. Mishra

Welcome to BookMyConsultation!

Kindly upload the KYC documents.

Regards,

BookMyConsultation

At the bottom, there are three buttons for responding: 'Thanks a lot.', 'Awesome, thank you so much!', and 'Thank you, looking forward to it!'. Below these buttons are 'Reply' and 'Forward' buttons.

5.2. Endpoint-2: Upload doctor documents

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

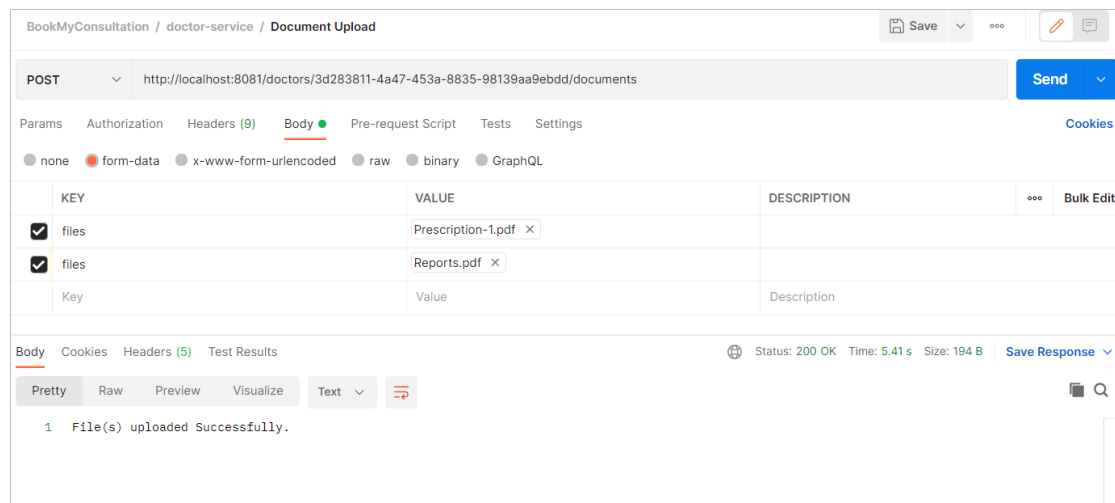
Request Body Ex -

	KEY	VALUE
<input checked="" type="checkbox"/>	files	Prescription-1.pdf ×
<input checked="" type="checkbox"/>	files	Reports.pdf ×

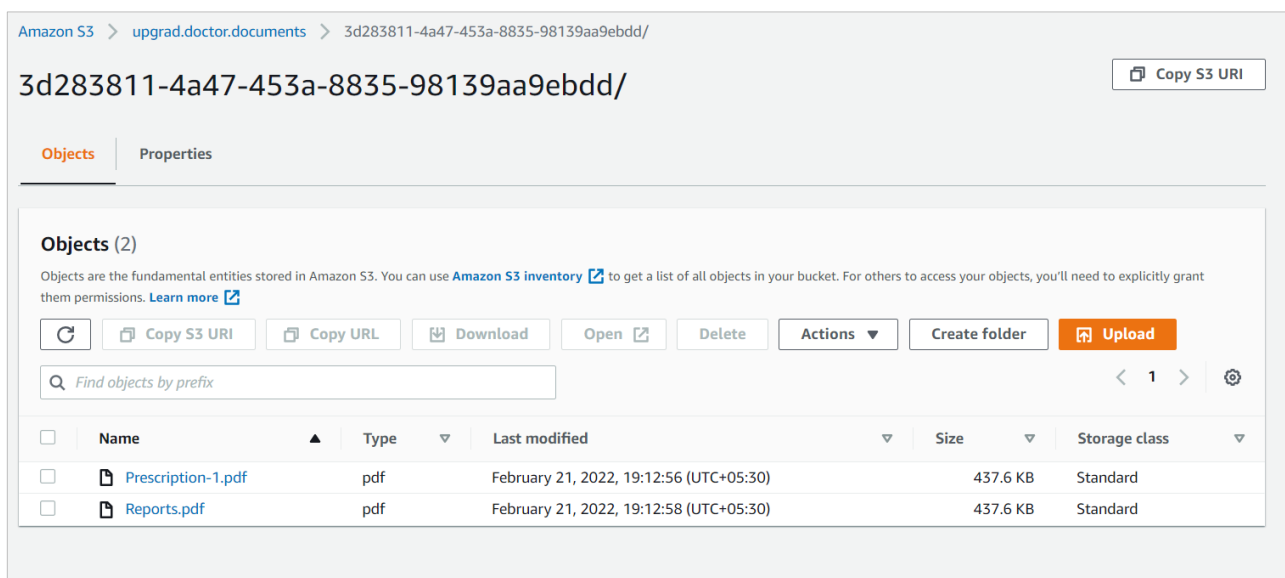
Response Body Ex -

File(s) uploaded Successfully.

Calling the endpoint from POSTMAN -



S3 bucket updated with the uploaded documents -



5.3. Endpoint-3: Approve doctor registration

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

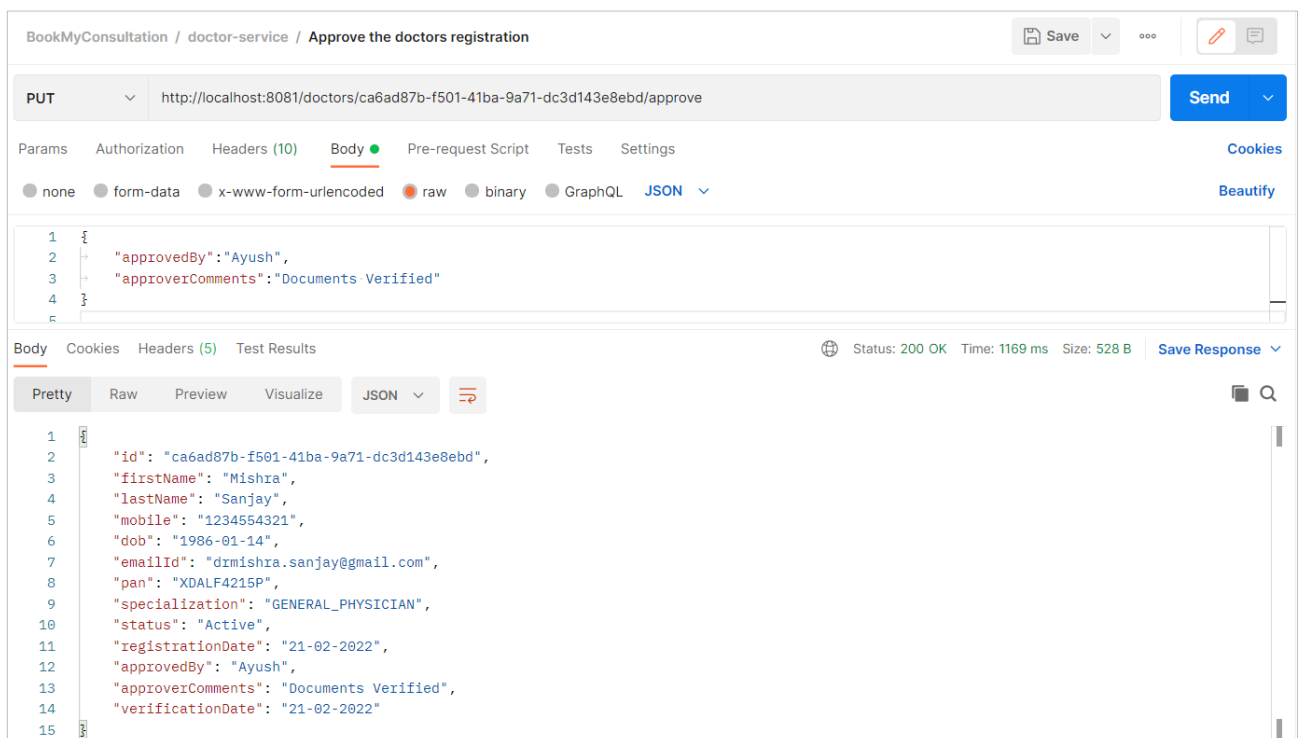
Request Body Ex –

```
{
  "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": "drnishra.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 –



5.4. 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 –

The screenshot shows the Postman interface for a PUT request to the endpoint `http://localhost:8081/doctors/d9b7db62-7b52-4982-92f6-821b7676a964/reject`. The request body is a JSON object with the following fields: `approvedBy` (Abhi), `approverComments` (Invalid documents). The response is a JSON object with the following fields: `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` (Abhinav), `approverComments` (Invalid documents), and `verificationDate` (21-02-2022). The status is 200 OK, time is 475 ms, and size is 525 B.

BookMyConsultation / doctor-service / Reject the doctors registration

PUT http://localhost:8081/doctors/d9b7db62-7b52-4982-92f6-821b7676a964/reject

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 2 3 4 5

```
1 "approvedBy": "Abhinav",
2 "approverComments": "Invalid documents"
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 "id": "d9b7db62-7b52-4982-92f6-821b7676a964",
2 "firstName": "Abhinav",
3 "lastName": "Jha",
4 "mobile": "1234554321",
5 "dob": "1986-01-14",
6 "emailId": "abhinav.jha@gmail.com",
7 "pan": "XDALF4215P",
8 "specialization": "GENERAL_PHYSICIAN",
9 "status": "Rejected",
10 "registrationDate": "19-02-2022",
11 "approvedBy": "Abhinav",
12 "approverComments": "Invalid documents",
13 "verificationDate": "21-02-2022"
```

Status: 200 OK Time: 475 ms Size: 525 B Save Response

5.5. Endpoint-5: Return list of doctors

5.5.1. Scenario-1: The status is pending

GET localhost:8081/doctors?status=Pending

Calling the endpoint from POSTMAN –

BookMyConsultation / doctor-service / Fetch the list of doctors based on status and speciality

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> status	Pending	
<input type="checkbox"/> speciality	GENERAL_PHYSICIAN	
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 248 ms Size: 511 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",
3   "firstName": "Kumar",
4   "lastName": "Shri",
5   "mobile": "1234554321",
6   "dob": "1986-01-14",
7   "emailId": "shri.kumar@gmail.com",
8   "pan": "XDALF4215P",
9   "specialization": "GENERAL_PHYSICIAN",
10  "status": "Pending",
11  "registrationDate": "19-02-2022",
12  "approvedBy": "Pending",
13  "approverComments": "Pending",
14  "verificationDate": "Pending"
15 }
```

5.5.2. Scenario-2: The status is Active

GET localhost:8081/doctors?status=Active

Calling the endpoint from POSTMAN –

BookMyConsultation / doctor-service / Fetch the list of doctors based on status and speciality

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

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> status	Active	
<input type="checkbox"/> speciality	GENERAL_PHYSICIAN	
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 269 ms Size: 530 B Save Response

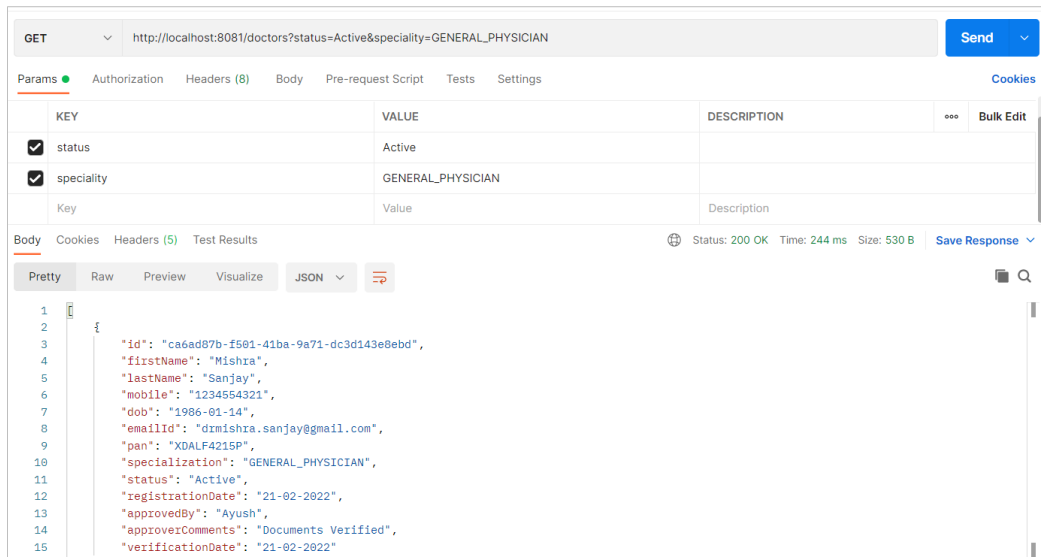
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
3   "firstName": "Mishra",
4   "lastName": "Sanjay",
5   "mobile": "1234554321",
6   "dob": "1986-01-14",
7   "emailId": "dr.mishra.sanjay@gmail.com",
8   "pan": "XDALF4215P",
9   "specialization": "GENERAL_PHYSICIAN",
10  "status": "Active",
11  "registrationDate": "21-02-2022",
12  "approvedBy": "Ayush",
13  "approverComments": "Documents Verified",
14  "verificationDate": "21-02-2022"
15 }
```

5.5.3. Scenario-2: Based on speciality

GET localhost:8081/status=Active&speciality=GENERAL_PHYSICIAN

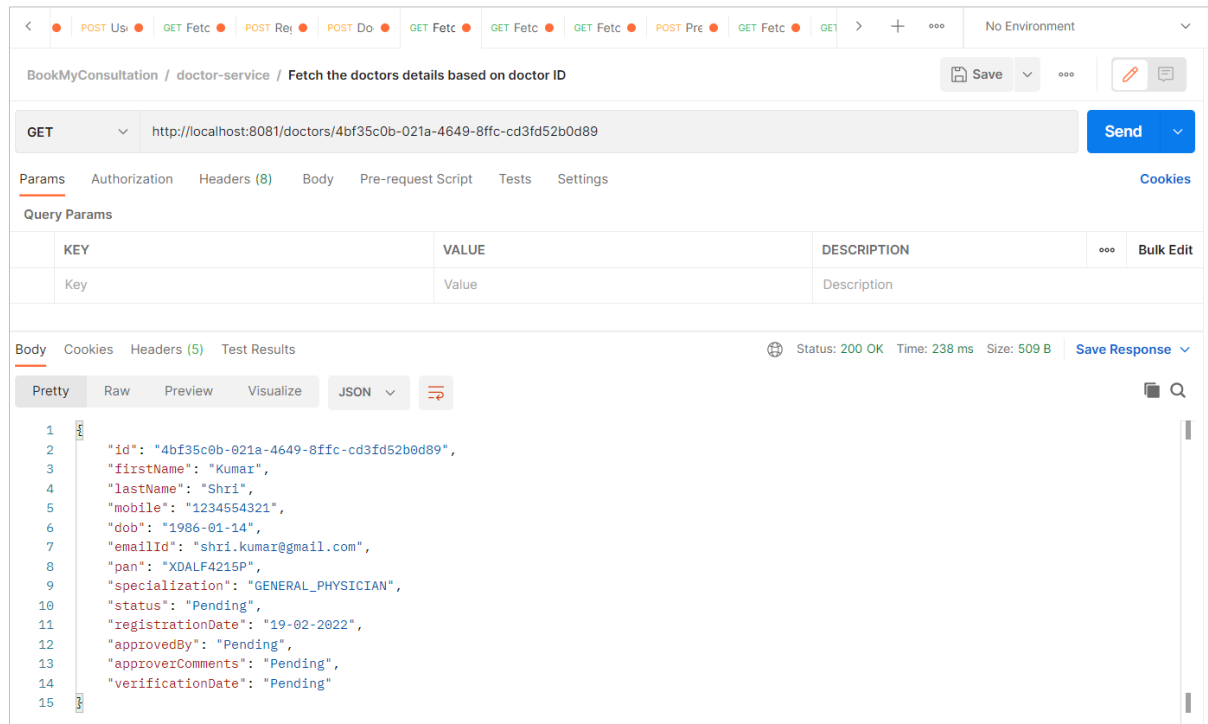
Calling the endpoint from POSTMAN –



5.6. Endpoint-6: Return doctors based on doctor-ID

GET localhost:8081/ /doctors/{doctorId}

Calling the endpoint from POSTMAN –



6. User-Onboarding Service

6.1. Endpoint-1: Collect user information

POST localhost:8082/users

Content-Type application/json

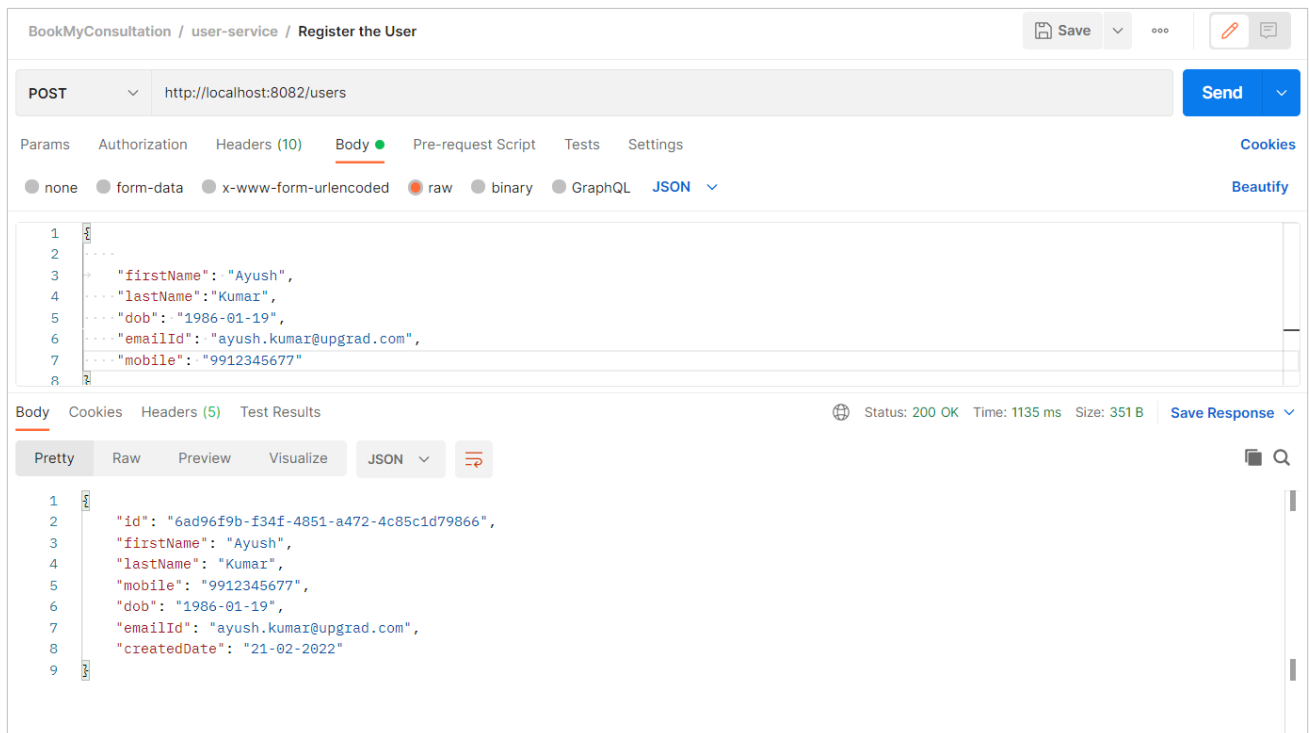
Request Body Ex -

```
{
  "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' -

upgrad.User

DOCUMENTS 4 STORAGE SIZE 20.5KB AVG. SIZE 247B INDEXES 1 TOTAL SIZE 36.9KB AVG. SIZE 36.9KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET ...

ADD DATA VIEW REFRESH

Displaying documents 1 - 4 of 4

```
{
  "_id": "6da5d31b-eed0-49f9-af9d-8912eae16725",
  "firstName": "Abhinav",
  "lastName": "Kumar",
  "mobile": "9912345677",
  "dob": "1986-01-19",
  "emailId": "ayush.kumar@upgrad.com",
  "createdAt": "18-02-2022",
  "_class": "com.example.userservice.entity.User"
}
```

```
{
  "_id": "d8af2910-f09f-4549-8d76-f003c8ead1a7",
  "firstName": "user1",
  "lastName": "Kumar",
  "mobile": "9912345677",
  "dob": "1986-01-19",
  "emailId": "ayush.kumar@upgrad.com",
  "createdAt": "19-02-2022",
  "_class": "com.example.userservice.entity.User"
}
```

Email received by the new user-

Welcome Email Inbox x

anubhavh3@gmail.com via amazonses.com
to me ▾

Hello Ayush

Welcome to BookMyConsultation!

Kindly upload the KYC documents.

Regards,

BookMyConsultation

[Thanks a lot.](#) [Awesome, thank you so much!](#) [Thank you, looking forward to it!](#)

[Reply](#) [Forward](#)

6.2. 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 -

The screenshot shows the Postman interface for a GET request. The URL is `http://localhost:8082/users/6ad96f9b-f34f-4851-a472-4c85c1d79866`. The response status is 200 OK, with a time of 328 ms and a size of 351 B. The response body is displayed in JSON format, showing user details for the specified ID.

BookMyConsultation / user-service / Fetch the details of the user based on userID

GET `http://localhost:8082/users/6ad96f9b-f34f-4851-a472-4c85c1d79866` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 328 ms Size: 351 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "6ad96f9b-f34f-4851-a472-4c85c1d79866",
3   "firstName": "Ayush",
4   "lastName": "Kumar",
5   "mobile": "9912345677",
6   "dob": "1986-01-19",
7   "emailId": "ayush.kumar@upgrad.com",
8   "createdDate": "21-02-2022"
9 }
```

6.3. Endpoint-3: Upload user documents

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

Request Body Ex -

KEY	VALUE
<input checked="" type="checkbox"/> files	Report-1.pdf <input type="button" value="x"/>
<input checked="" type="checkbox"/> files	XRay.pdf <input type="button" value="x"/>

Response Body Ex -

File(s) uploaded Successfully.

Calling the endpoint from POSTMAN -

BookMyConsultation / user-service / Upload the documents

POST http://localhost:8082/users/6ad96f9b-f34f-4851-a472-4c85c1d79866/documents

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> files	Report-1.pdf <input type="button" value="x"/>	
<input checked="" type="checkbox"/> files	XRay.pdf <input type="button" value="x"/>	
Key	Value	Description

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 4.36 s Size: 193 B Save Response

Pretty Raw Preview Visualize Text

1 File(s) uploaded Successfully

S3 bucket updated with the uploaded documents -

Amazon S3 > upgrad.user.documents > 6ad96f9b-f34f-4851-a472-4c85c1d79866/

6ad96f9b-f34f-4851-a472-4c85c1d79866/ Copy S3 URI

Objects Properties

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	Reports.pdf	pdf	February 21, 2022, 19:26:52 (UTC+05:30)	437.6 KB	Standard
<input type="checkbox"/>	XRay.pdf	pdf	February 21, 2022, 19:25:36 (UTC+05:30)	437.6 KB	Standard

7. Appointment Service

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

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

Content-Type application/json

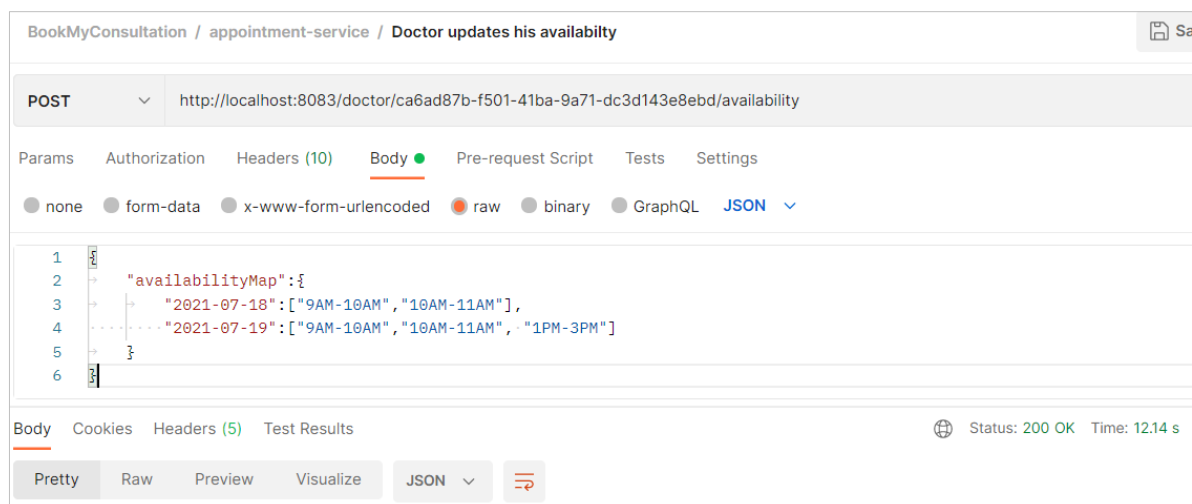
Request Body Ex -

```
{
  "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 -

The screenshot shows a SQL query result in RDS. The query is `select * from availability;`. The result is a table with 6 columns: `id`, `availability_date`, `doctor_id`, `is_booked`, and `timeslot`. The table contains 5 rows of data.

	id	availability_date	doctor_id	is_booked	timeslot
1	1	2021-07-18	ca6ad87b-f501-41ba-9a71-dc3d143e8ebd	0	9AM-10AM
2	2	2021-07-18	ca6ad87b-f501-41ba-9a71-dc3d143e8ebd	0	10AM-11AM
3	3	2021-07-19	ca6ad87b-f501-41ba-9a71-dc3d143e8ebd	0	9AM-10AM
4	4	2021-07-19	ca6ad87b-f501-41ba-9a71-dc3d143e8ebd	0	10AM-11AM
5	5	2021-07-19	ca6ad87b-f501-41ba-9a71-dc3d143e8ebd	0	1PM-3PM

7.2. 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 -

The screenshot shows a Postman interface for a GET request. The URL is `http://3.232.56.68:8083/doctor/ca6ad87b-f501-41ba-9a71-dc3d143e8ebd/availability`. The response status is 200 OK, with a time of 636 ms and a size of 321 B. The response body is displayed in JSON format, showing the doctor's ID and their availability for two dates: 2021-07-18 and 2021-07-19.

BookMyConsultation / appointment-service / Fetch the doctor's availability based on doctor ID

GET `http://3.232.56.68:8083/doctor/ca6ad87b-f501-41ba-9a71-dc3d143e8ebd/availability` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 636 ms Size: 321 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
3   "availabilityMap": {
4     "2021-07-18": [
5       "9AM-10AM",
6       "10AM-11AM"
7     ],
8     "2021-07-19": [
9       "9AM-10AM",
10      "10AM-11AM",
11      "1PM-3PM"
12    ]
13  }
14 }
```

7.3. Endpoint-3: Book Appointment

POST localhost:8083/appointments

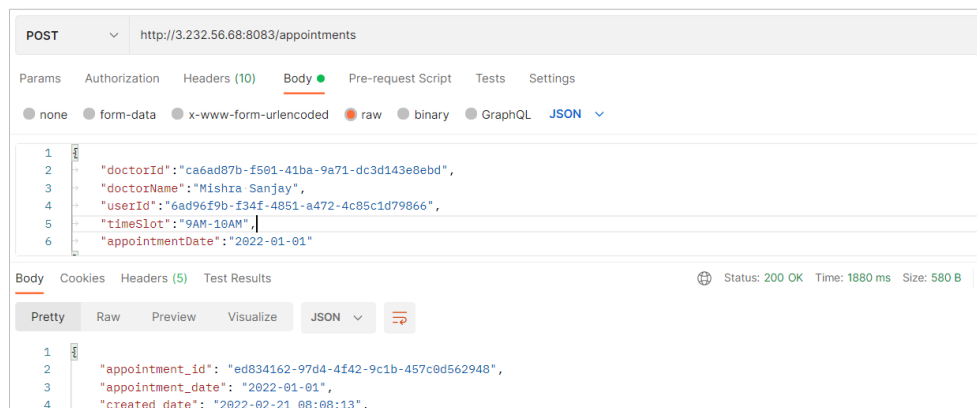
Request Body Ex -

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

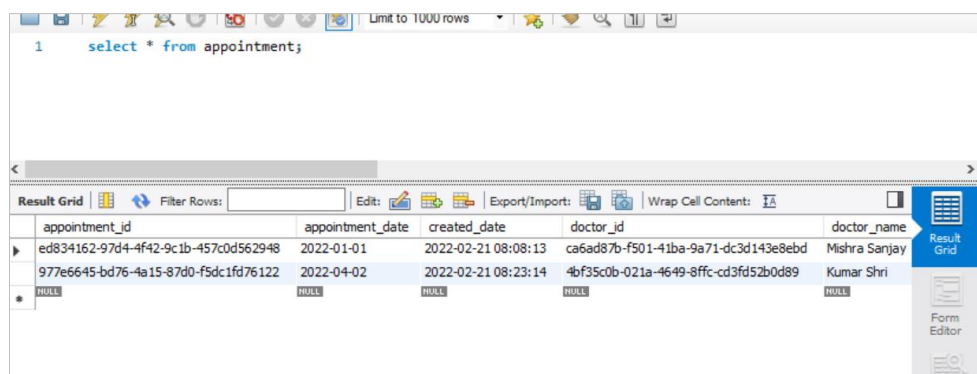
Response Ex -

```
{
  "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 -



7.4. 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 -

BookMyConsultation / appointment-service / Fetch the details of an appointment based on appointmentId

GET http://3.232.56.68:8083/appointments/ed834162-97d4-4f42-9c1b-457c0d562948

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1651 ms Size: 580 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "appointment_id": "ed834162-97d4-4f42-9c1b-457c0d562948",
3   "appointment_date": "2022-01-01",
4   "created_date": "2022-02-21 08:08:13",
5   "doctor_id": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
6   "prior_medical_history": null,
7   "status": "PAYMENT_PENDING",
8   "symptoms": null,
9   "timeslot": "9AM-10AM",
10  "userid": "6ad96f9b-f34f-4851-a472-4c85c1d79866",
11  "user_email_id": "ayush.kumar@upgrad.com",
12  "user_name": "Ayush Kumar",
13  "doctor_name": "Mishra Sanjay"
14 }
```

7.5. 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 -

BookMyConsultation / appointment-service / Fetch all the appointments of a user based on userId

GET http://3.232.56.68:8083/users/6ad96f9b-f34f-4851-a472-4c85c1d79866/appointments

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
-----	-------	-------------

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 480 ms Size: 636 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",
4     "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
5     "userId": "6ad96f9b-f34f-4851-a472-4c85c1d79866",
6     "timeSlot": "9AM-10AM",
7     "status": "PAYMENT_PENDING",
8     "appointmentDate": "2022-01-01"
9   },
10  {
11    "appointmentId": "977e6645-bd76-4a15-87d0-f5dc1fd76122",
12    "doctorId": "4bf35c0b-021a-4649-8ffc-cd3fd52b0d89",
13    "userId": "6ad96f9b-f34f-4851-a472-4c85c1d79866",
14    "timeSlot": "01PM-02PM",
15    "status": "PAYMENT_PENDING",
16    "appointmentDate": "2022-04-02"
17  }
18 ]
```


7.6. 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 -

BookMyConsultation / appointment-service / Prescription

POST http://localhost:8083/prescriptions

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",
3   "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
4   "doctorName": "Mishra Sanjay",
5   "userId": "2ed143af-1565-40b8-b322-8144024de9ee"
```

Body Cookies Headers (4) Test Results

Status: 400 Bad Request Time: 1635 ms Size: 202 B

Pretty Raw Preview Visualize Text

```
1 {\"statusCode\":400, \"message\": \"errorCode: PAYMENT_PENDING\"}
```

Calling the endpoint from POSTMAN when PAYMENT is confirmed –

BookMyConsultation / appointment-service / Prescription

POST http://3.232.56.68:8083/prescriptions

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
2 {
3   "appointmentId": "ed834162-97d4-4f42-9c1b-457c0d562948",
4   "doctorId": "ca6ad87b-f501-41ba-9a71-dc3d143e8ebd",
5   "doctorName": "Mishra Sanjay",
6   "userId": "2ed143af-1565-40b8-b322-8144024de9ee",
7   "diagnosis": "Teeth Cavity",
8   "medicineList": [
9     {
10      "name": "Calpol",
11      "type": "Tablet",
12      "dosage": "1 week",
13      "duration": "1 week",
14      "frequency": "3 times a day",
15      "remarks": "after food"
16    },
17     {
18      "name": "PainKill",
19      "type": "Syrup",
20      "dosage": "1 week",
21      "duration": "1 week",
22      "frequency": "3 times a day",
23      "remarks": "after food"
24    }
25   ]
26 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 1992 ms Size: 639 B

Pretty Raw Preview Visualize JSON

Mongo Collection 'Prescription' –

upgrad.Prescription

DOCUMENTS 3 20.5KB 561B INDEXES 1 36.9KB 36.9KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

ADD DATA VIEW

Displaying documents 1 - 4 of 4

```
{
  "_id": "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"
    }
  ]
}
```

8. Payment Service

8.1. 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",
  "createdAt": "21-02-2022 08:29:43"
}
```

Calling the endpoint from POSTMAN –

The screenshot shows the Postman interface for a GET request to the endpoint `http://3.232.56.68:8084/payments?appointmentId=ed834162-97d4-4f42-9c1b-457c0d562948`. The request is saved under the name "Payment endpoint". The "Query Params" tab is active, showing a single parameter `appointmentId` with the value `ed834162-97d4-4f42-9c1b-457c0d562948`. The "Body" tab is also active, showing the response in JSON format. The response status is 200 OK, with a time of 1916 ms and a size of 300 B. The response body is:

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

9. Rating Service

9.1. Endpoint-1: Rate Doctors.

POST localhost:8085/ratings

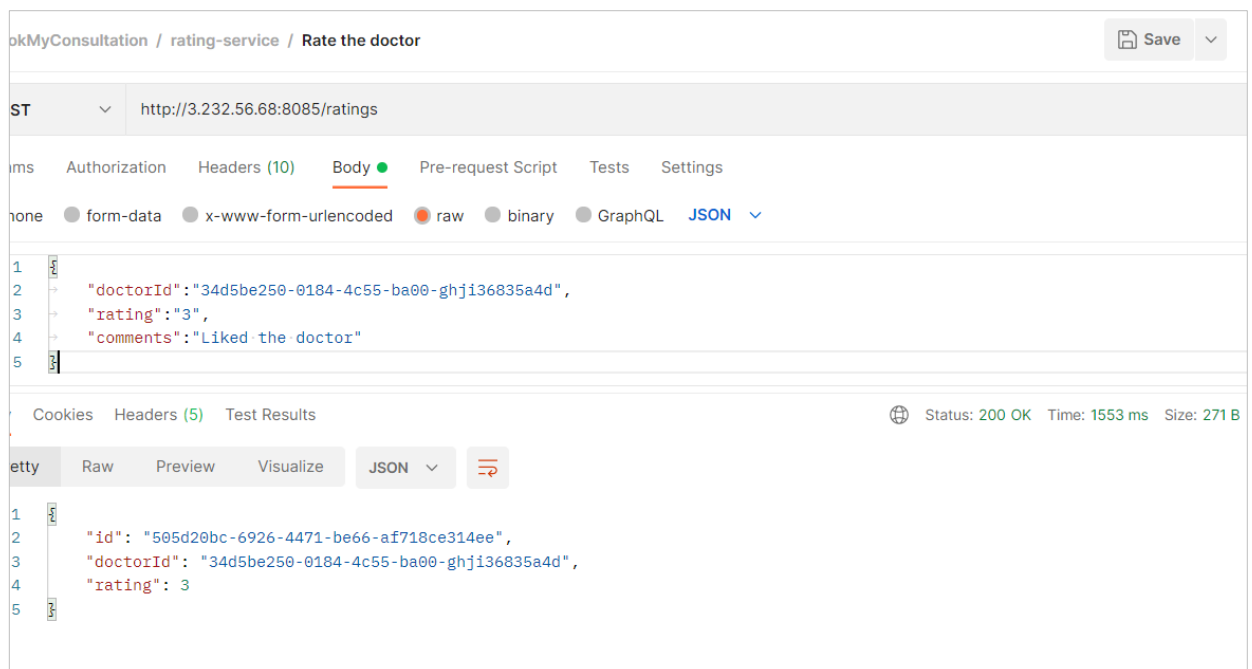
Request Body Ex –

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

Response Ex -

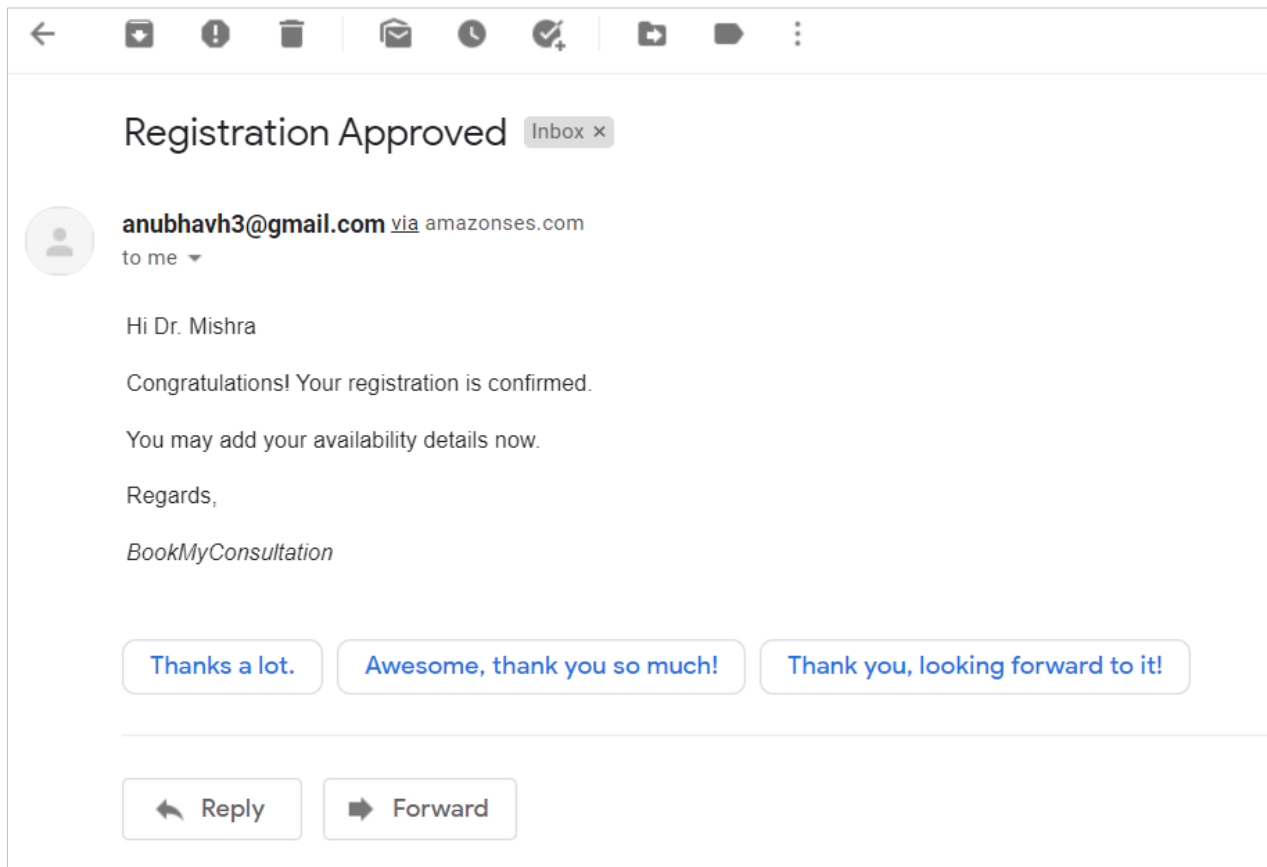
```
{
  "id": "505d20bc-6926-4471-be66-af718ce314ee",
  "doctorId": "34d5be250-0184-4c55-ba00-ghji36835a4d",
  "rating": 3
}
```

Calling the endpoint from POSTMAN –




10. Notification Service

10.1. Doctor Registration Approval



Registration Approved Inbox x

 **anubhavh3@gmail.com** [via amazonses.com](#)
to me ▾

Hi Dr. Mishra

Congratulations! Your registration is confirmed.

You may add your availability details now.

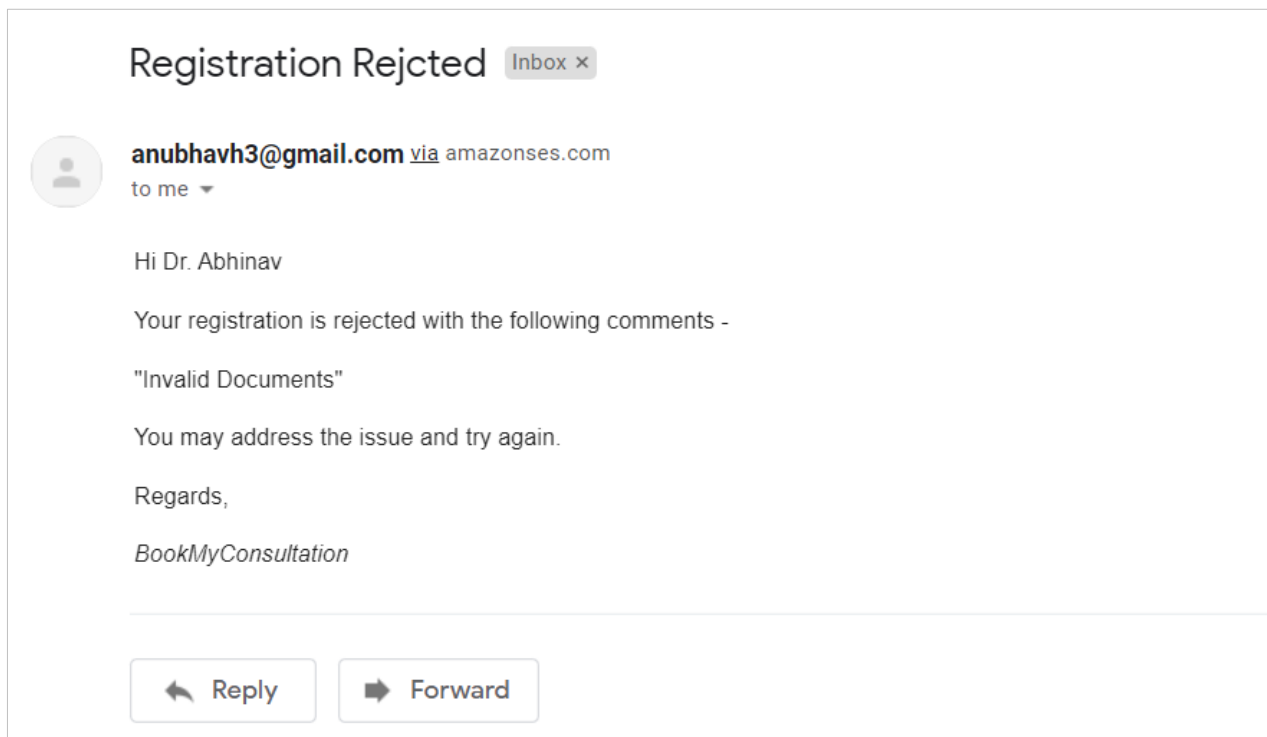
Regards,

BookMyConsultation


[Thanks a lot.](#) [Awesome, thank you so much!](#) [Thank you, looking forward to it!](#)

[↩ Reply](#) [➡ Forward](#)

10.2. Doctor Registration Rejection



Registration Rejected Inbox x

 **anubhavh3@gmail.com** [via amazonses.com](#)
to me ▾

Hi Dr. Abhinav

Your registration is rejected with the following comments -

"Invalid Documents"

You may address the issue and try again.

Regards,


BookMyConsultation

[↩ Reply](#) [➡ Forward](#)

10.3. Appointment Confirmation

Appointment Confirmation

Inbox x

**anubhavh3@gmail.com** via amazonses.com
to me ▾

Hello,

Your appointment has been successfully scheduled.


Doctor: Dr. Kumar Shri
Date: 2022-04-02
Timeslot: 01PM-02PM
Payment Status: Pending

Regards,

BookMyConsultation

↩ Reply➡ Forward

10.4. Prescription

**anubhavh3@gmail.com** via amazonses.com
to me ▾

Hello,

Please find attached the prescription for the appointment ID: "jd434162-u874-8f62-0h1t-457c0d562946" :

"diagnosis": "Cold and Fever",
"medicineList": [
 {
 "name": "Calpol",
 "type": "Tablet",
 "dosage": "1 week",
 "duration": "1 week",
 "frequency": "2 times a day",
 "remarks": "after food"
 }
]

Regards,

BookMyConsultation

↩ Reply➡ Forward

11. 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/ . . .
```

12. Security

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

13. Future Enhancements

- Add error and exception handling at all the possible points
- Implement a Configuration Server

*** END ***