Name: Yash Jayeshkumar Pandya
Email : yashp614929@gmail.com
Project : Course Project: Application Deployment Using Docker

---

Overview

The project `**Book My Consultation**` is an application that will register doctors. Users will be able to register themselves and then book appointments based on their requirements and the availability status of the doctors. After consulting the available doctor, users will be able to rate their experience. In this project, I have created a microservices based backend application using spring boot framework. The deployment using docker mainly has the same project code structure and addition of deployment files. Ie. **Dockerfile** and **docker-compose** file.

The project is divided into eight different services, which are as follows:

Doctor Service

> This service defines multiple API endpoints, this service will be used to register the doctors. Once the doctor submits their registration, the BMC employee will verify their details and approve or reject the registration. If the registration is approved, the status of the doctor will change from Pending to Active. If it is rejected, the status will change to Rejected.

User Service

> This service defines multiple API endpoints, this service will be used to register the users on the application. The users can also upload documents related to their past medical history.

Appointment Service

> This service defines multiple API endpoints, this service will be used to schedule the availability of doctors and book an appointment. The users can also upload documents related to their past medical history.

Payment Service

> This service defines single API endpoints, this service will be used to make the payment, and the status of the appointment will change from PendingPayment to Confirmed.

Rating Service

> This service defines single API endpoints, this service will be used to rate the user’s experience with the doctor.

Notification Service

This service will listen to the Kafka topics to which the other services will send the messages. This service will be responsible for sending emails for the following events:

- Doctor's approval
- Doctors' rejection
- Appointment confirmation
- Prescription

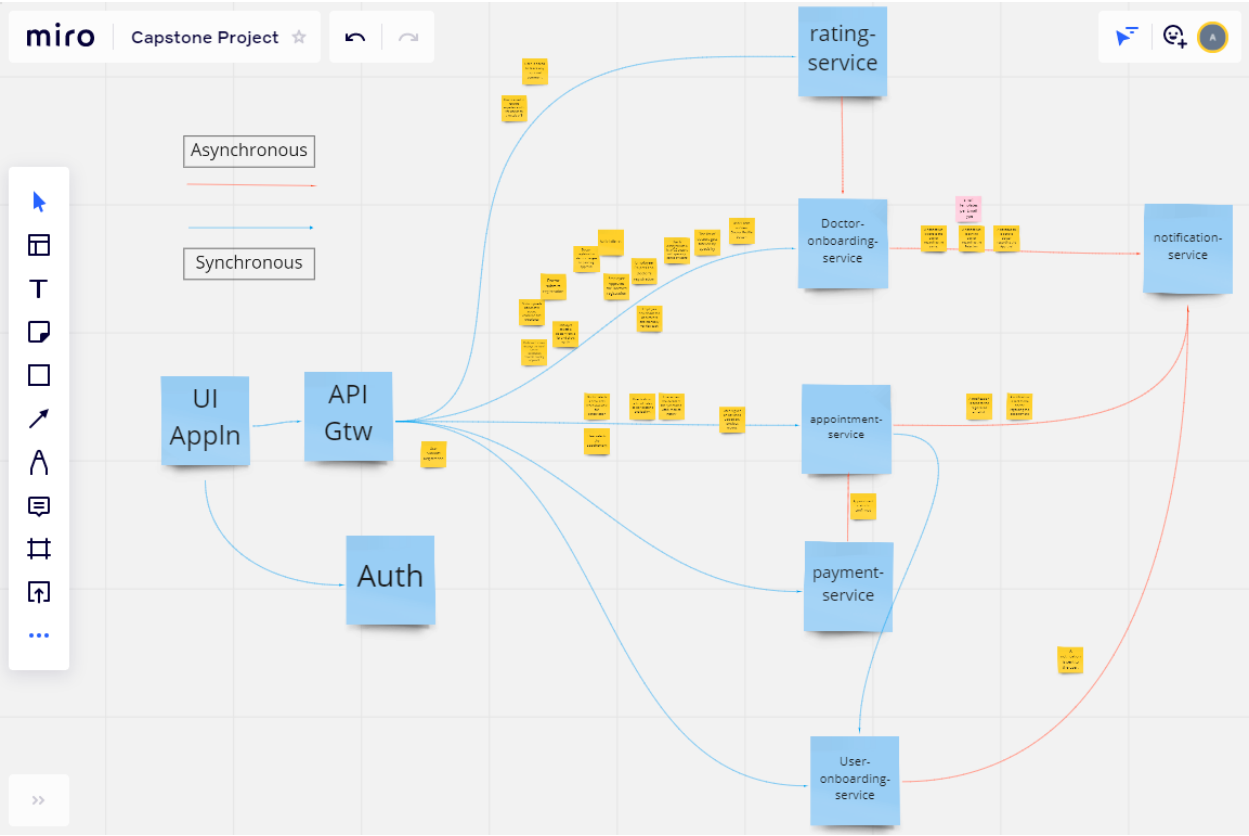
Security Provider

This service defines an endpoint which is used to generate the token and will be used to generate the JWT token. The token validation should be part of each microservice

BMC Gateway

This service is used to hide the actual API URLs from the UI application. The API gateway will intercept all the requests and route them to the correct service.

Application Workflow



Solution Architecture:

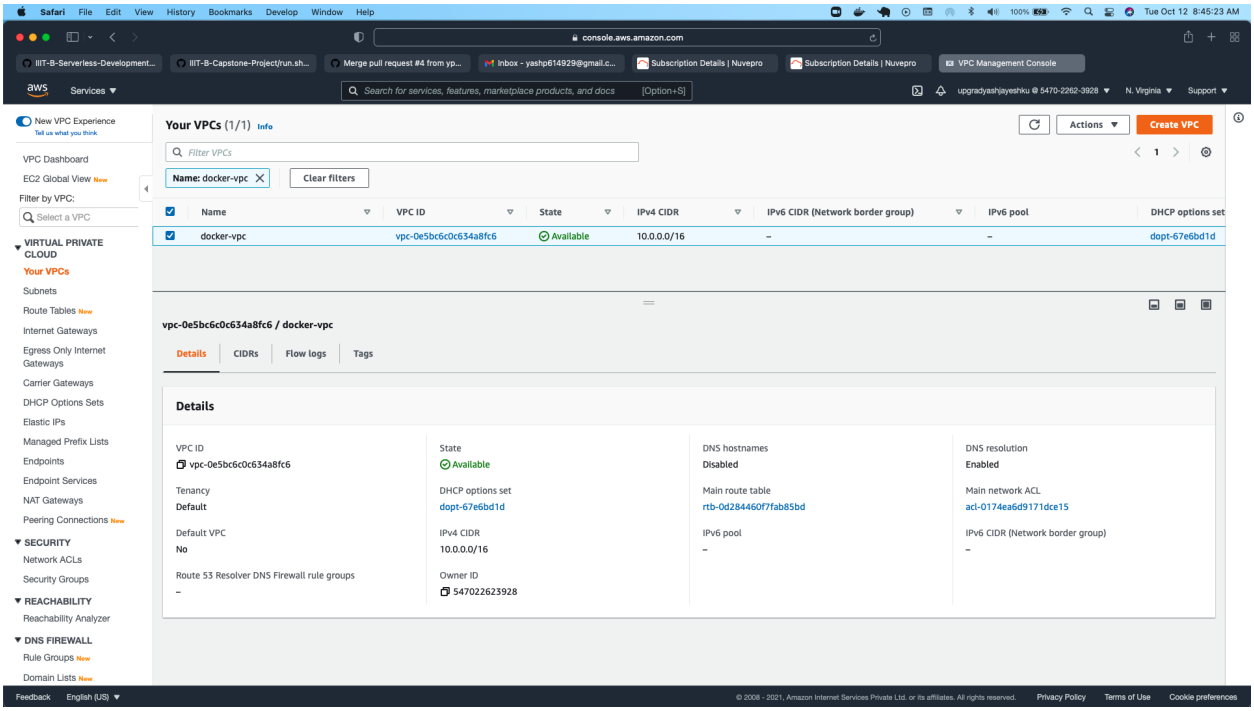
All Services are combined into a single directory called **'BookMyConsultation'**. All the defined services are a type of spring boot application.

Deployment cycle contains the following resources.

- AWS VPC
- AWS RDS
- AWS EC2
- AWS Elastic IPs
- AWS Security Groups
- Apache kafka
- AWS S3
- AWS SES
- MySQLWorkbench
- Postman

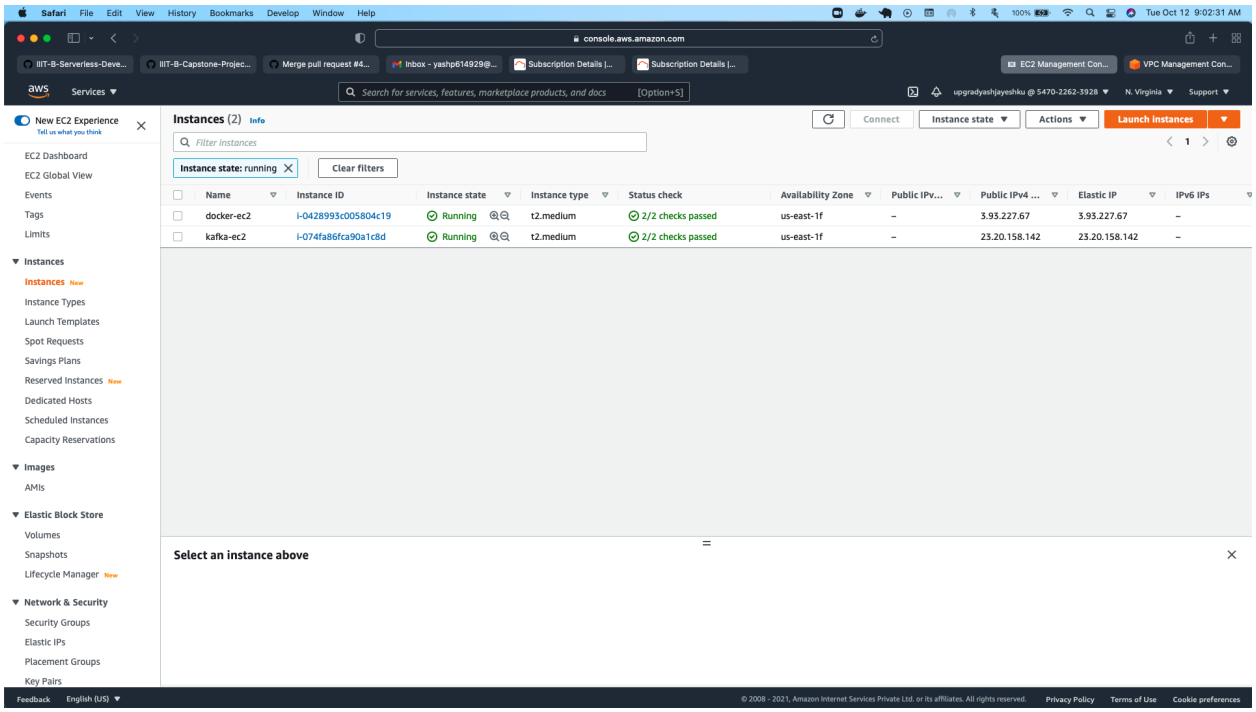
Solution approach is to **use a separate EC2 instance to install the apache kafka** and **do not containerize the kafka using the docker-compose**. Ie. **externalized kafka**. The connection to kafka is done using the public IP of the EC2 instance which is associated with the Elastic IP.

Firstly, a new VPC is created to launch resources that are being used while setting up the project.



Two EC2 instances are created with the type t2.medium.

- **docker-ec2** instance is to deploy the application code using docker.
- **Kafka-ec2** instance is to install apache kafka for asynchronous communication for microservices.



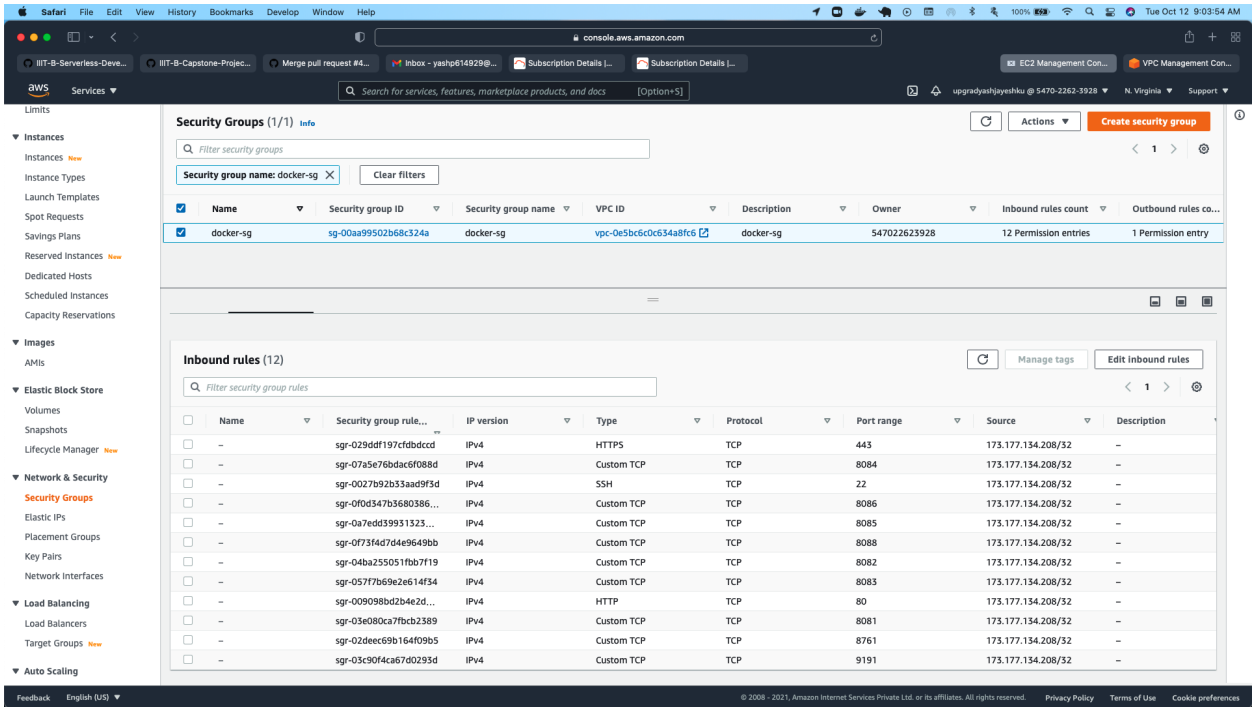
New security group **docker-sg** is created and below inbound rules are added under this security group. This security group is associated with the **docker-ec2** where the application code is being deployed.

Note that all inbound rules are restricted to allow access only within developer/student's IP.

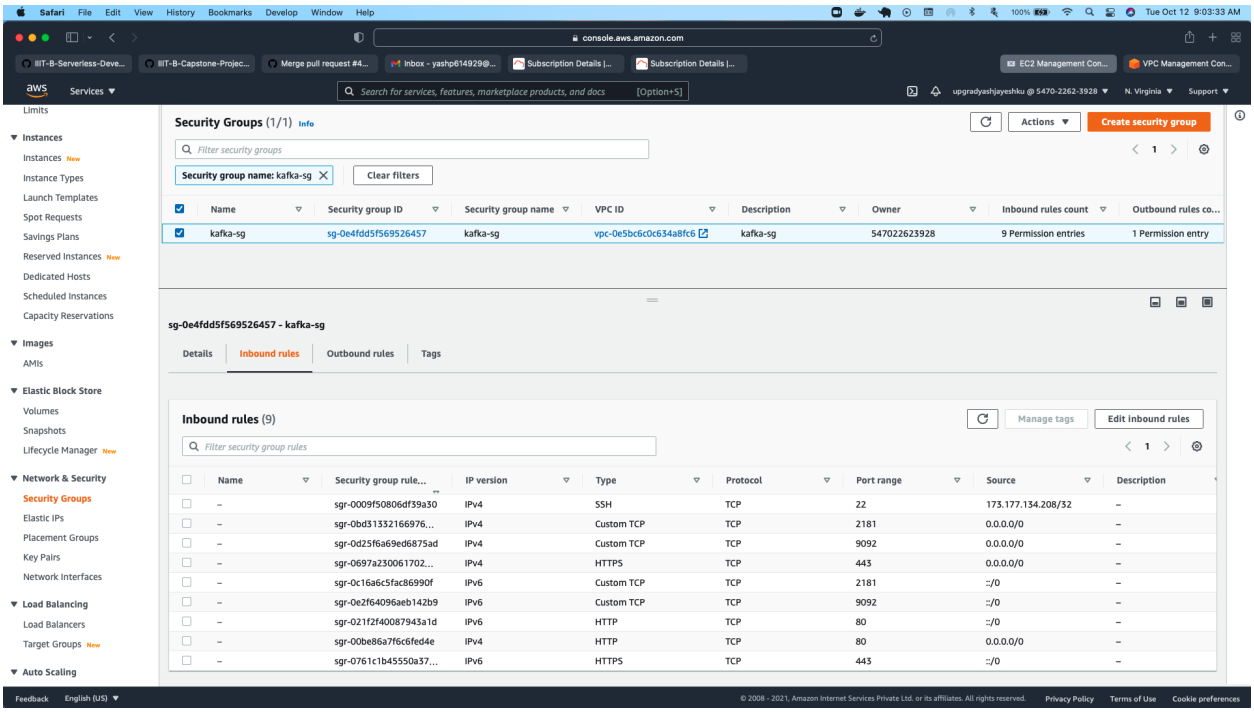
<u>Ports</u>

- **22** is open to gain access to **docker-ec2** instances.
- **80** is open for all HTTP requests.
- **443** is open for all HTTPS requests.
- **27017** is open for all MongoDB.
- **8761** is open to access the Service Registry(Eureka).
- **9191** is open to access api gateway (BMC-GATEWAY).
- **8081** is open to access doctor service (DOCTOR-SERVICE).
- **8082** is open to access payment service (APPOINTMENT-SERVICE).
- **8083** is open to access user service (USER-SERVICE).

- **8084** is open to access rating service (RATING-SERVICE).
- **8085** is open to access payment service (NOTIFICATION-SERVICE).
- **8086** is open to access payment service (PAYMENT-SERVICE).
- **8088** is open to access payment service (SECURITY-PROVIDER).



Another security group **kafka-sg** is created and below inbound rules are added under this security group. This security group is associated with the **kafka-ec2** where the apache kafka is being deployed.



Note that all inbound rules are publicly opened.

Ports

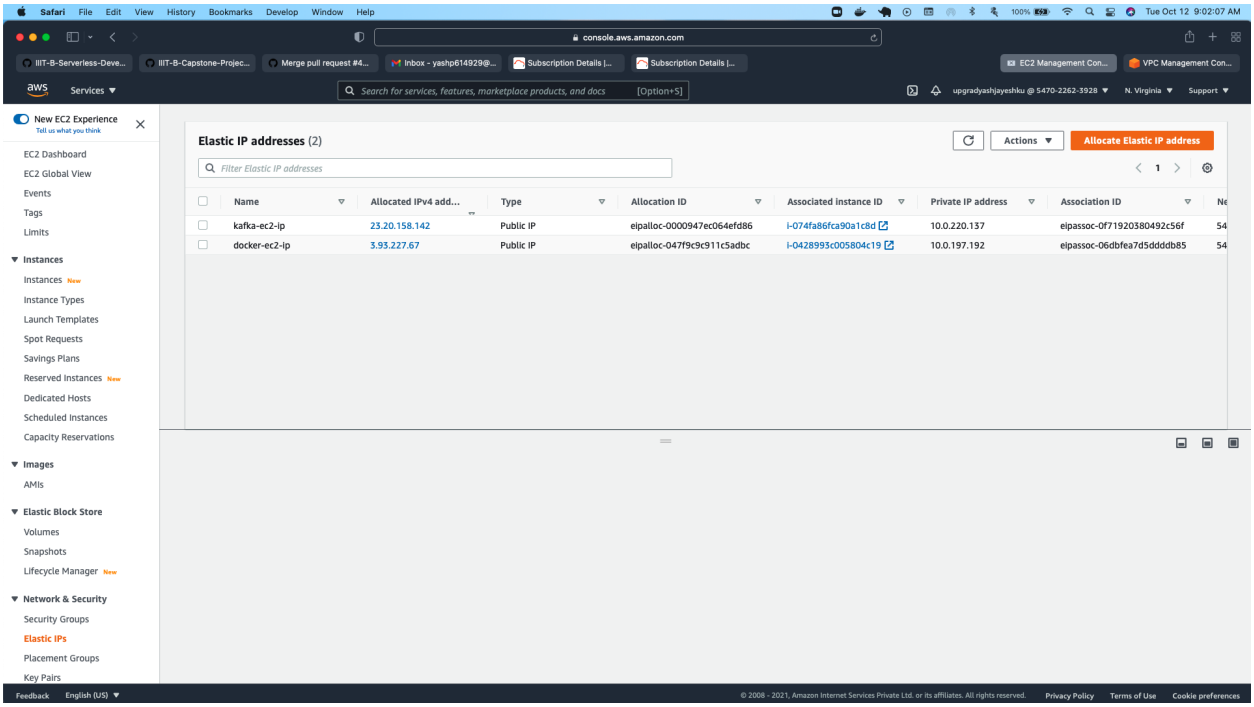
- **22** is open to gain access to **kafka-ec2** instance.
- **80** is open for all HTTP requests.
- **443** is open for all HTTPS requests.
- **2181** is open for apache kafka.
- **9092** is open for netcat.

EC2 instance connection

The single pem file is used to connect with both the ec2 instances. docker-ec2 and kafka-ec2.

ssh -i "docker-dep.pem" ubuntu@3.93.227.67
ssh -i "docker-dep.pem" ubuntu@23.20.158.142

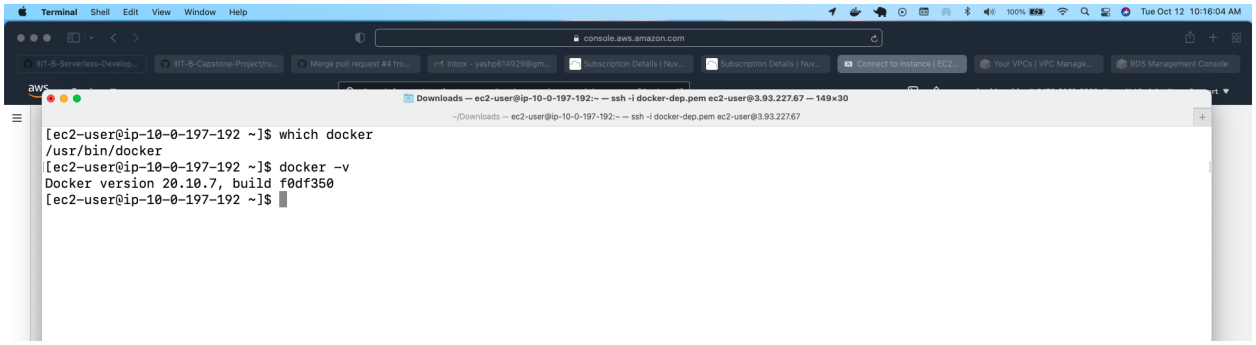
The public IPs are used to connect with EC2 instances using the ssh command. However, the IP gets changed when the EC2 instance is stopped. In-order to make a common IP until the project deployment is completed, Elastic IPs are used. Each EC2 instance is associated with the elastic IP.



Docker and Docker-Compose Installation

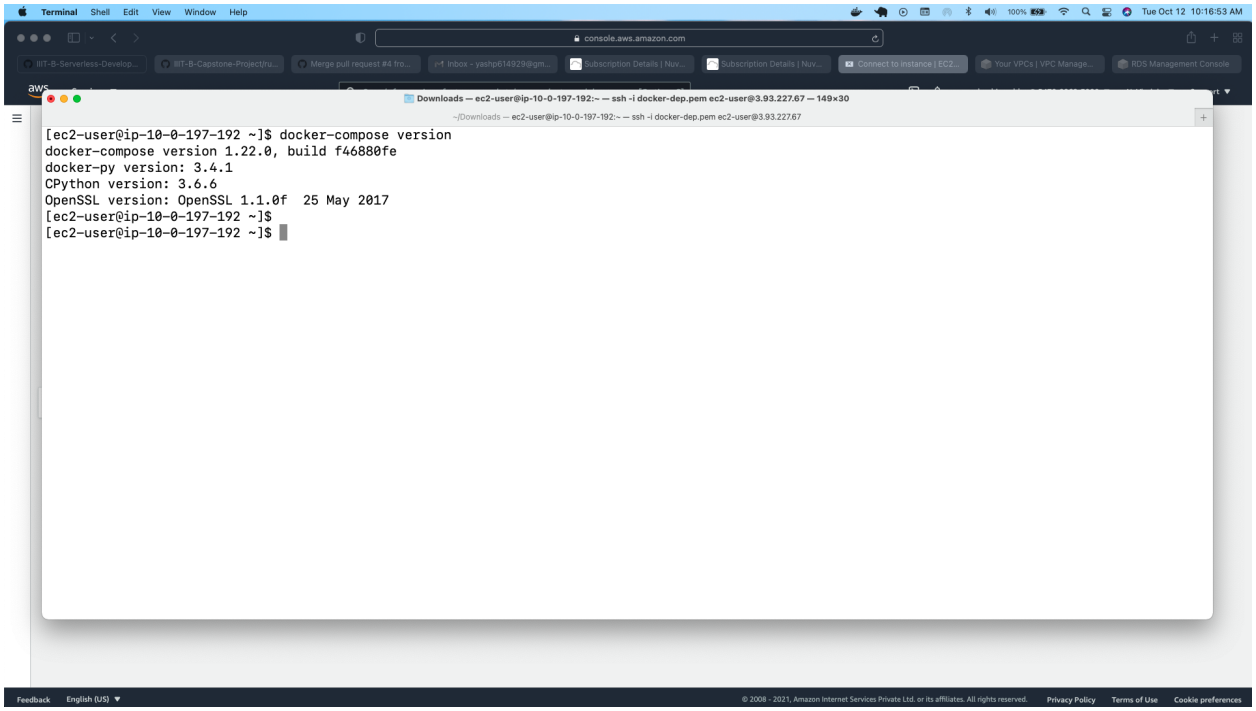
docker is installed on the **docker-ec2** instance using the docker installation process and verifies the installation using the below command.

```
which docker
docker -v
```



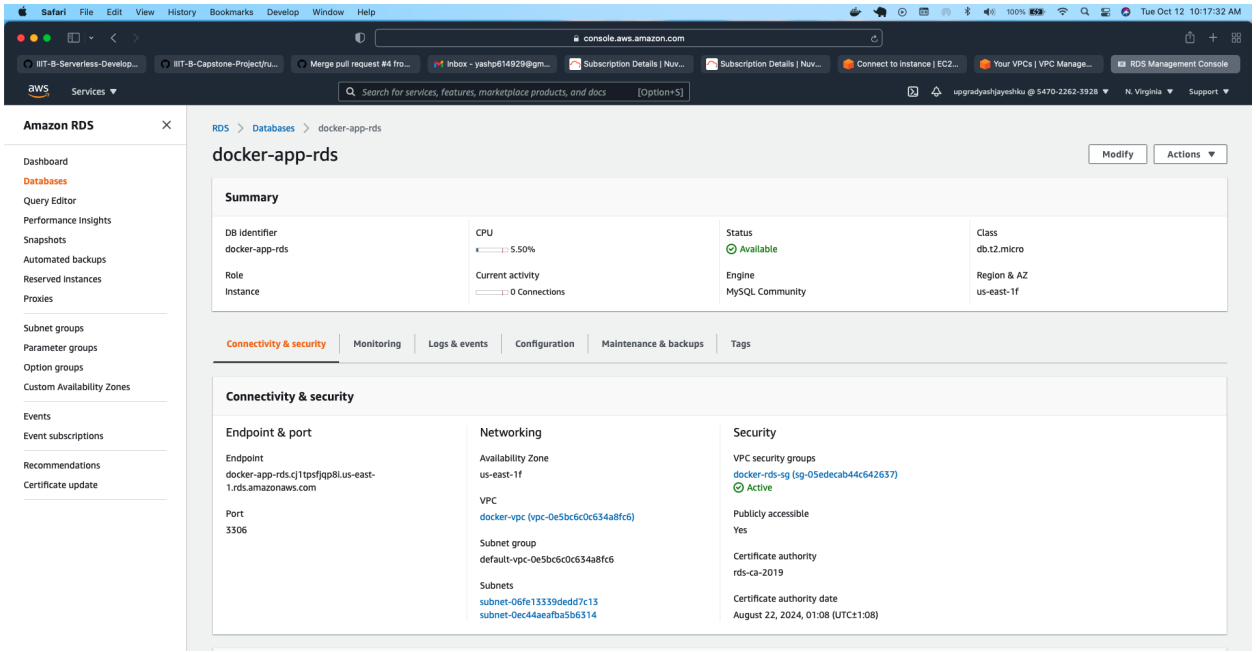
docker-compose is installed on the **docker-ec2** instance using the docker-compose installation process and verifies the installation using the below command.

```
docker-compose --version
```



RDS

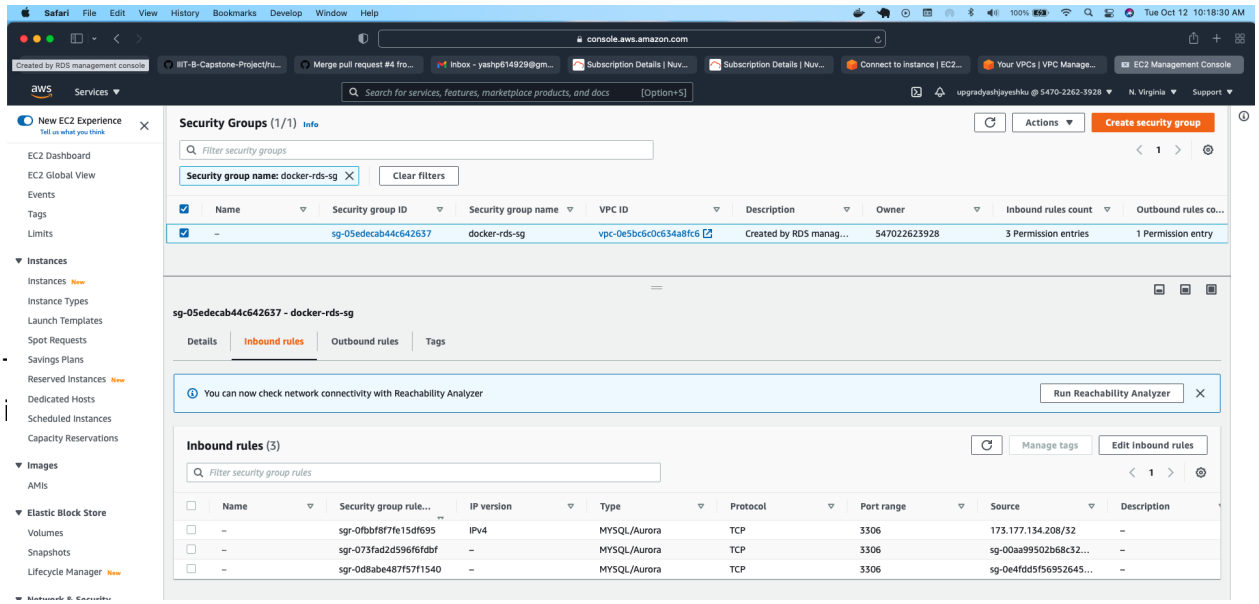
The RDS database instance named **docker-app-rds** is created using the MySQL Engine. The RDS is created within the **docker-vpc**.



The security group **docker-rds-sg** is created with below inbound rules and associated with **docker-app-rds** RDS instance.

Ports

- **3306** is open to connect RDS from developer/student's IP.
- **3306** is open to connect from the docker-sg security group.



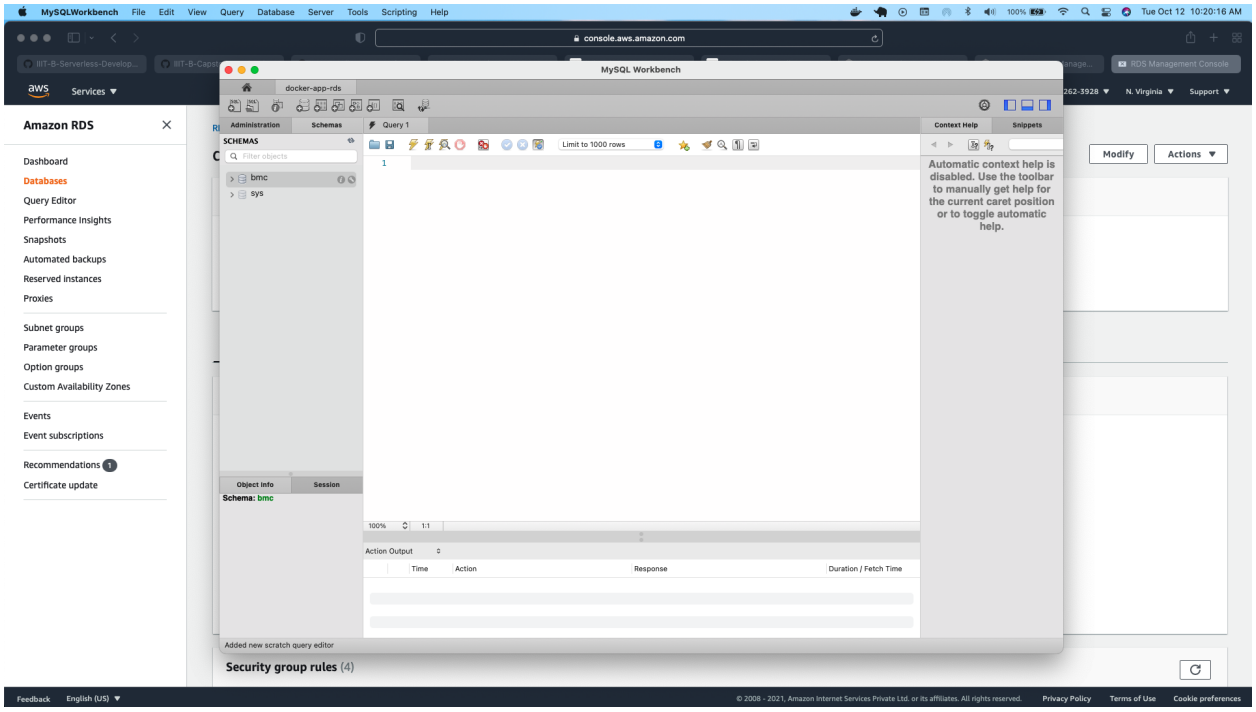
Appointment service uses the below database for availability and appointment information storage.

- bmc

All the services use the below database to validate authorization tokens (JWT).

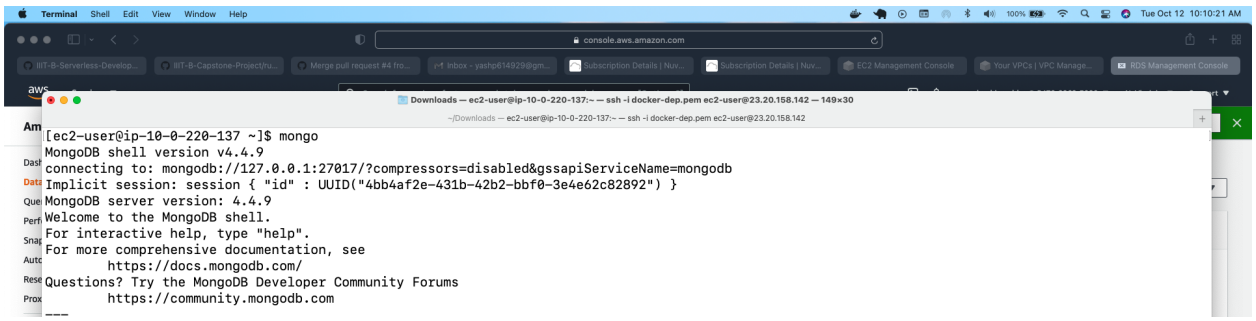
- bmc

RDS is created with username as admin and password as upgrad123



MONGO DB

The mongo database is also installed and running on **kafka-ec2** instance.

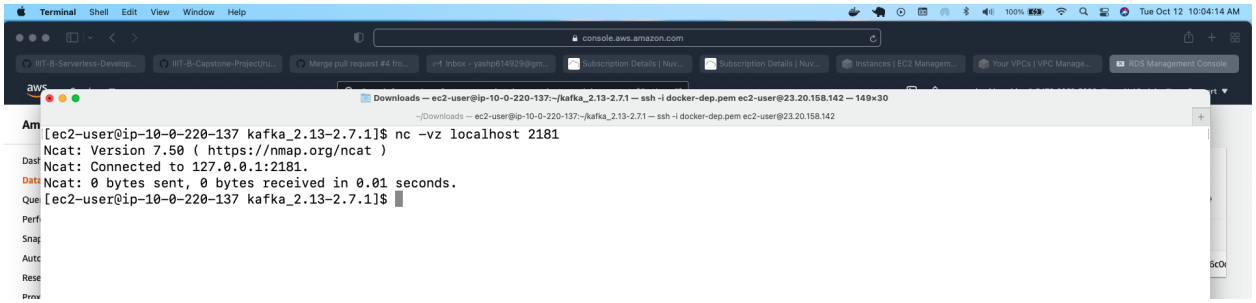


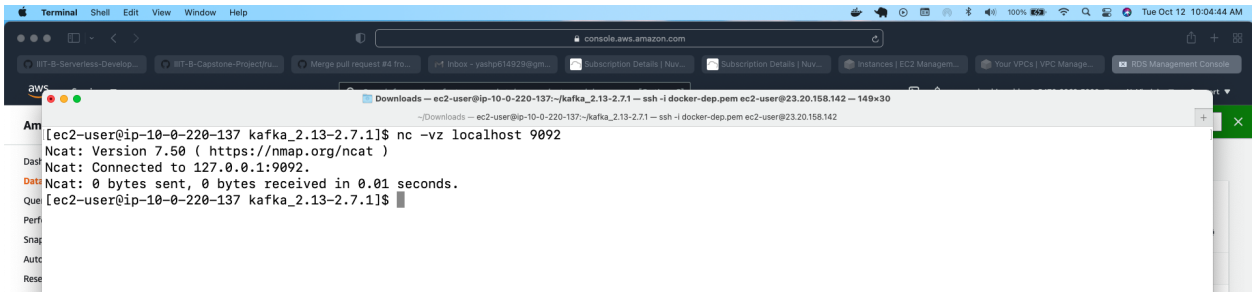
Kafka Installation

`bin/zookeeper-server-start.sh config/zookeeper.properties`

As described above **kafka-ec2** instance is used to install the apache kafka and run the kafka asynchronous communication for the notification service

Kafka and Netcat are installed using the installation process. The properties file is configured with the elastic IP of **kafka-ec2** instance. Both kafka and zookeeper are running and verified using netcat as shown below.





Application Deployment and Run instructions

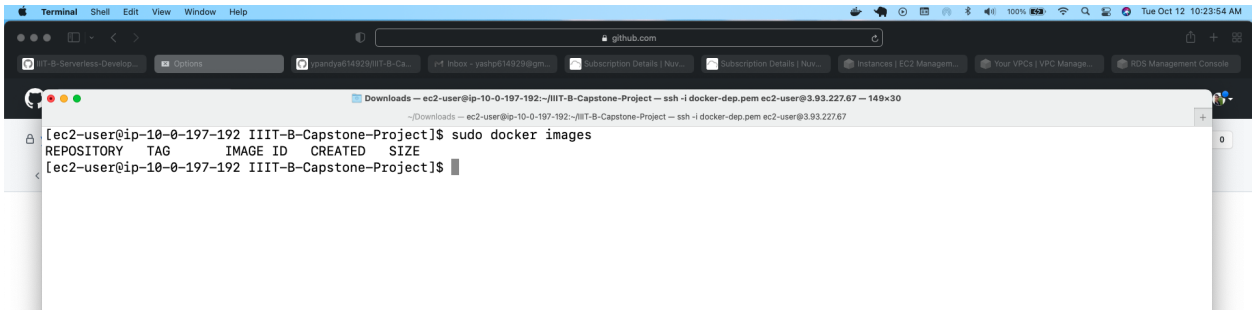
The application code including the Dockerfiles and docker-compose file are grouped together and pushed to the github private repository. The codebase is pulled on to the docker-ec2 instance using the below command.

`git clone`
`https://github.com/ypandya614929/IIIT-B-Capstone-Project.git`

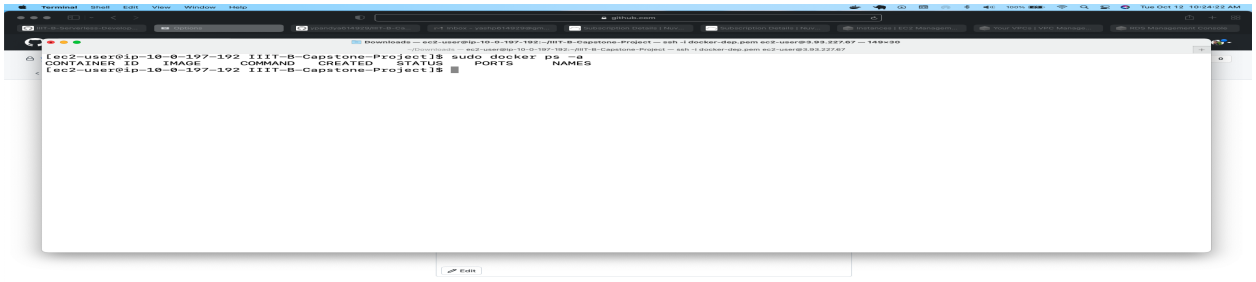
Please note that this is the private repository created under the developer/student's github account in order to simplify the process.

There are no docker images present initially and there are no containers running on the docker-ec2 instance. Both checks are verified using below commands.

sudo docker images



sudo docker ps -a



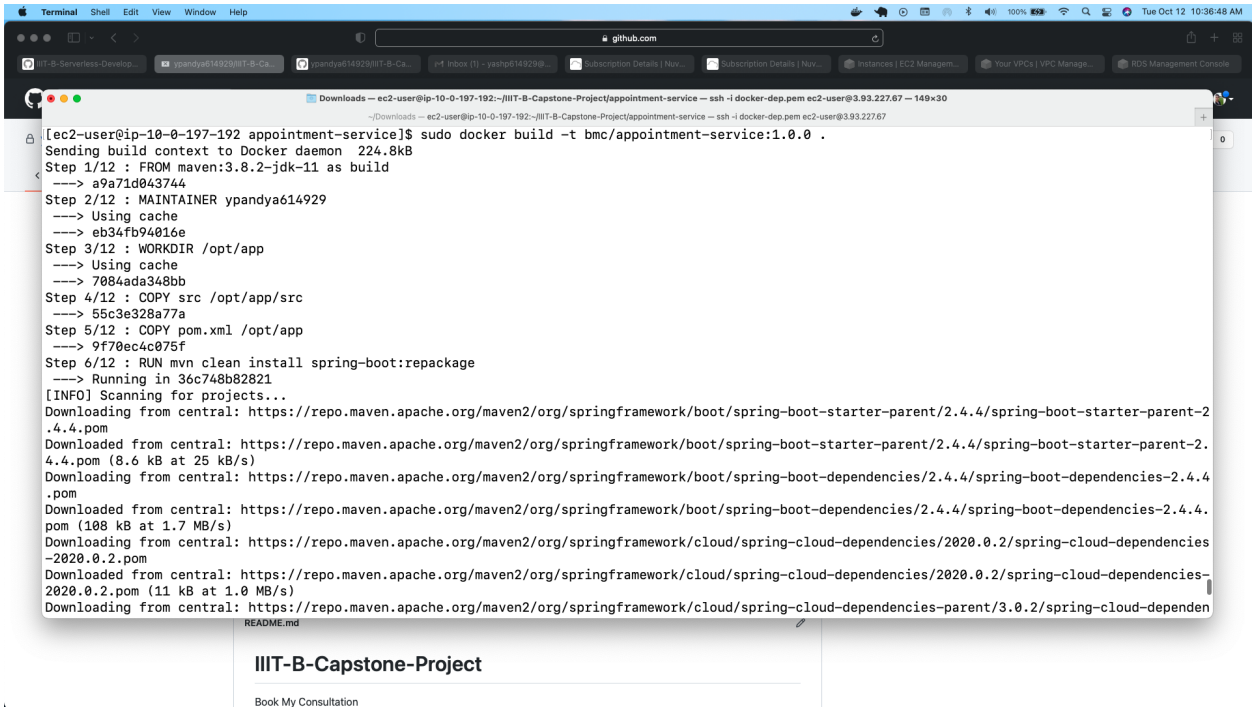
Appointment-Service Image

The below commands are present in the Appointment service Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/appointment-service.jar
/opt/app/appointment-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/appointment-service.jar"]
```

The docker image of the appointment service is built using the following command.

sudo docker build -t bmc/appointment-service:1.0.0 .



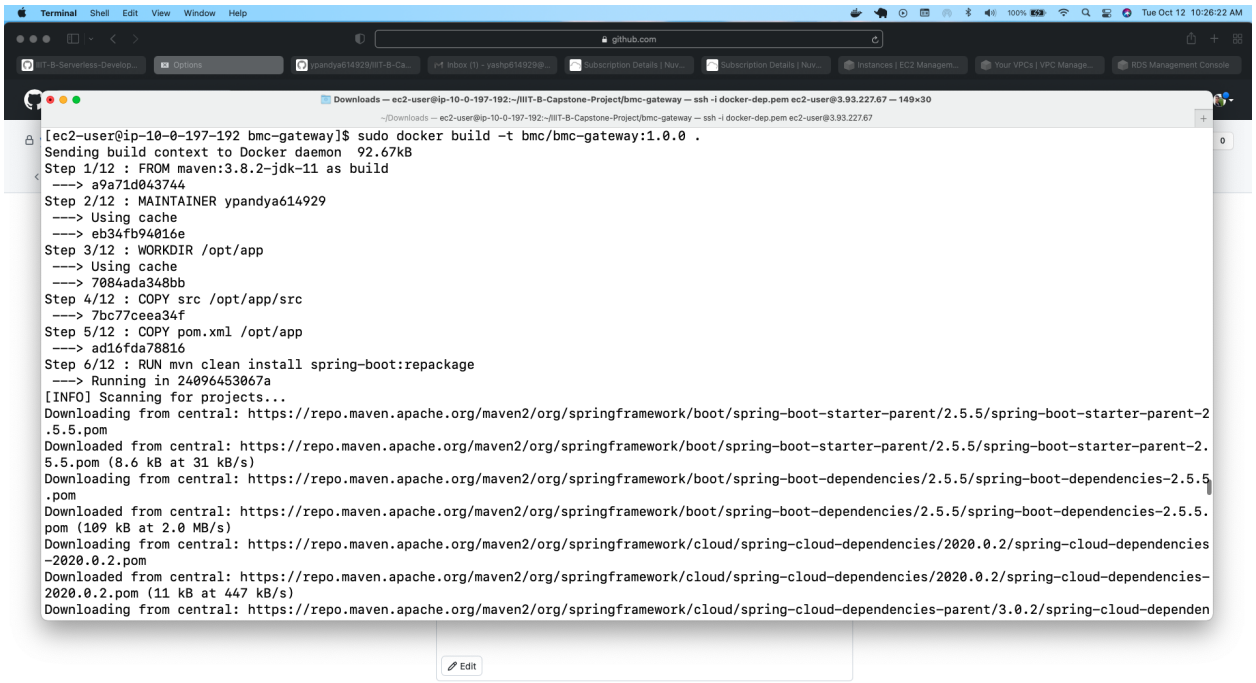
Bmc-Gateway Image

The below commands are present in the bmc-gateway Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/bmc-gateway.jar
/opt/app/bmc-gateway.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/bmc-gateway.jar"]
```

The docker image of the bmc-gateway is built using the following command.

sudo docker build -t bmc/bmc-gateway:1.0.0 .



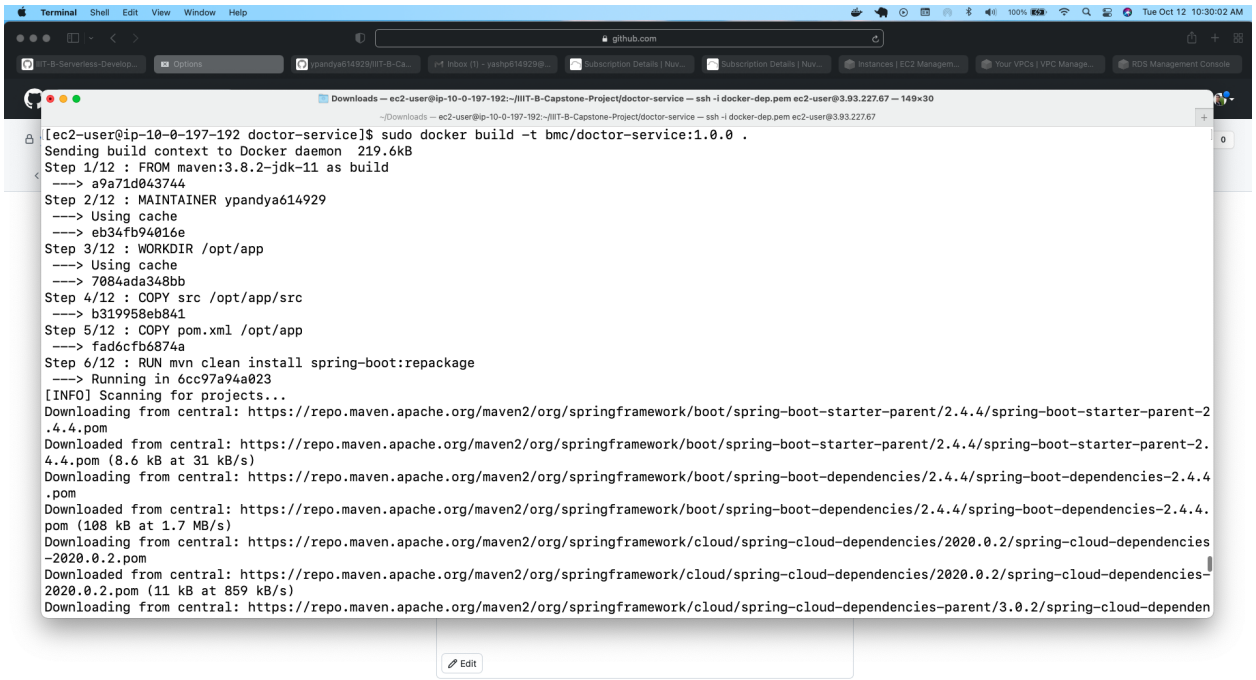
Doctor-Service Image

The below commands are present in the Doctor service Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/doctor-service.jar
/opt/app/doctor-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/doctor-service.jar"]
```

The docker image of the doctor service is built using the following command.

sudo docker build -t bmc/doctor-service:1.0.0 .



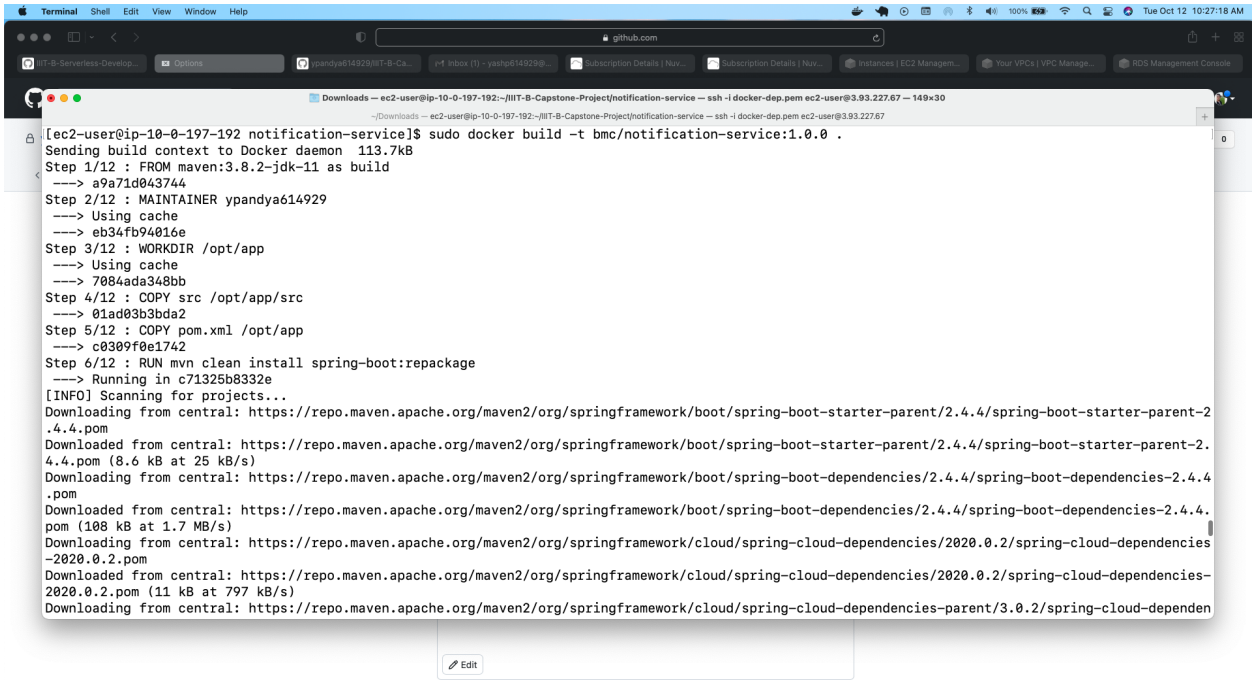
Notification-Service Image

The below commands are present in the Notification service Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/notification-service.jar
/opt/app/notification-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/notification-service.jar"]
```

The docker image of the notification service is built using the following command.

sudo docker build -t bmc/notification-service:1.0.0 .



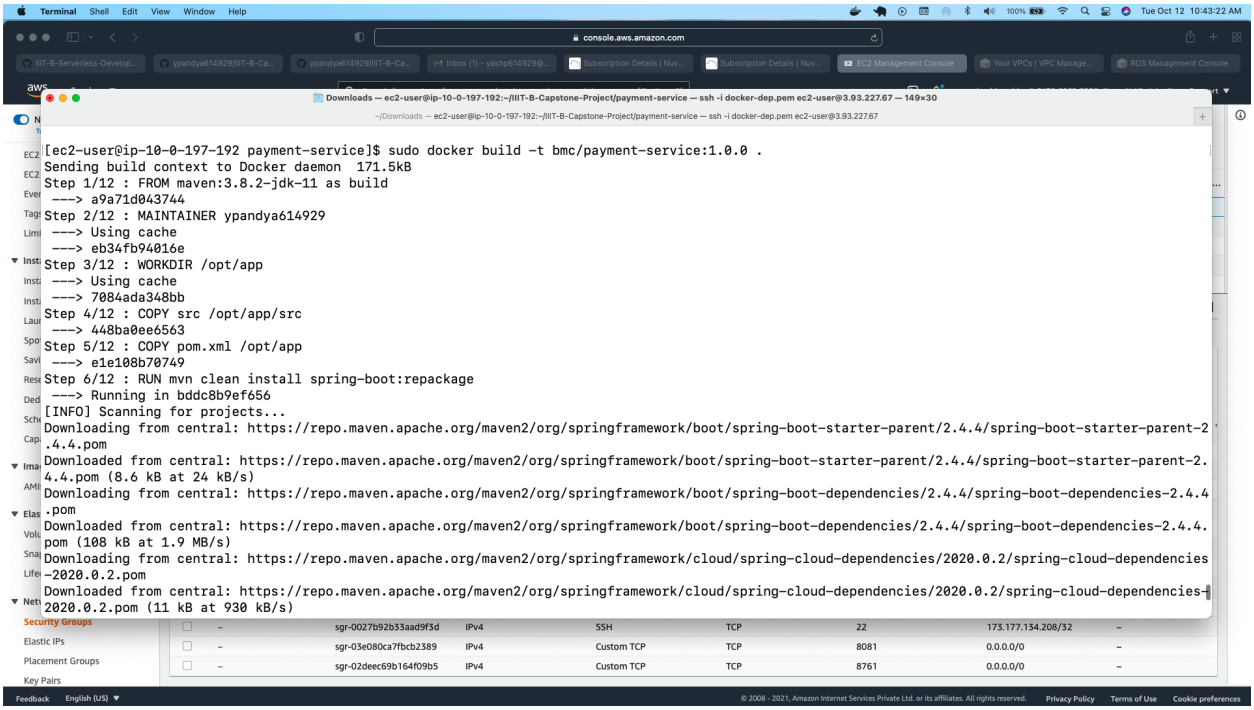
Payment-Service Image

The below commands are present in the Payment service Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/payment-service.jar
/opt/app/payment-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/payment-service.jar"]
```

The docker image of the payment service is built using the following command.

sudo docker build -t bmc/payment-service:1.0.0 .



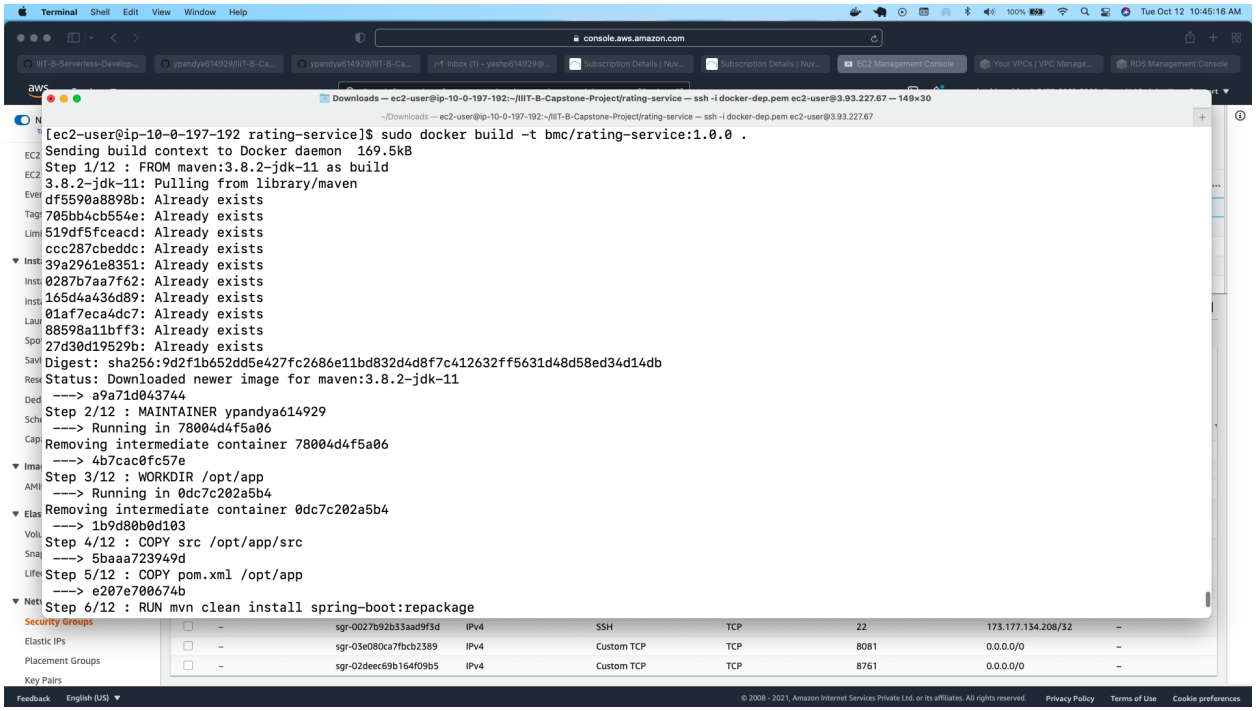
Rating-Service Image

The below commands are present in the Rating service Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/rating-service.jar
/opt/app/rating-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/rating-service.jar"]
```

The docker image of the rating service is built using the following command.

```
sudo docker build -t bmc/rating-service:1.0.0 .
```



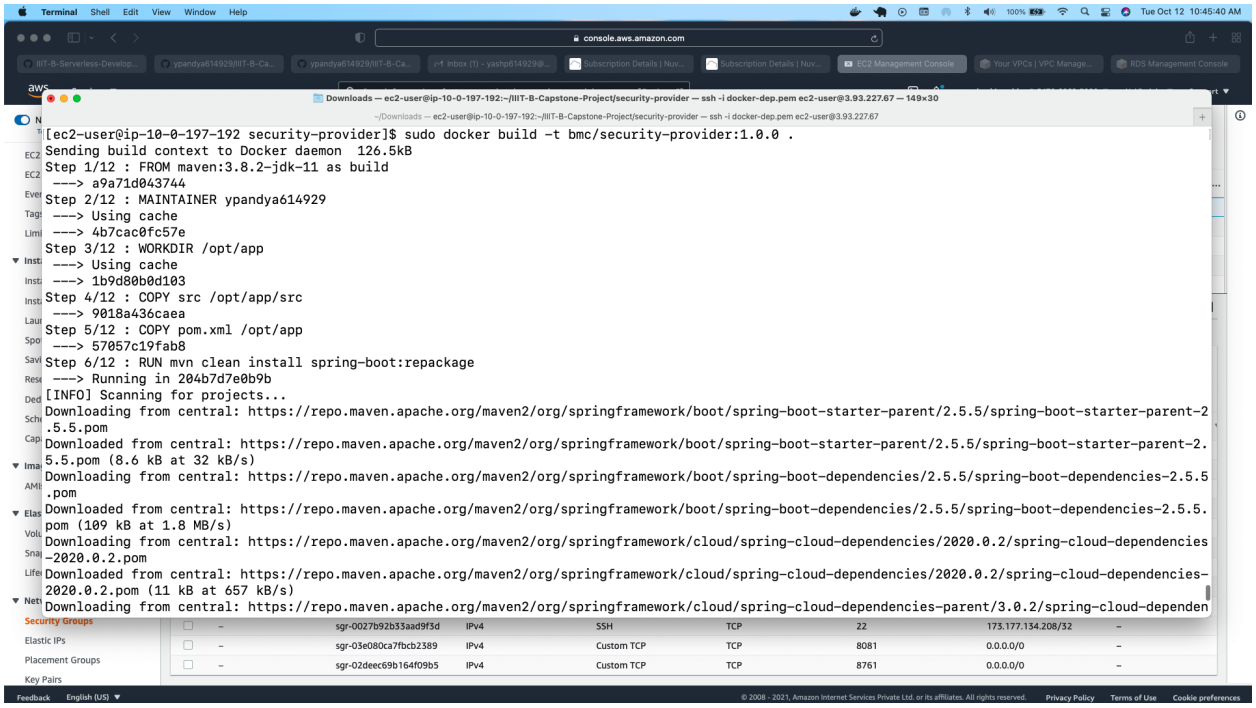
Security-Provider Image

The below commands are present in the security provider server Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/security-provider.jar
/opt/app/security-provider.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/security-provider.jar"]
```

The docker image of the security provider is built using the following command.

sudo docker build -t bmc/security-provider:1.0.0 .



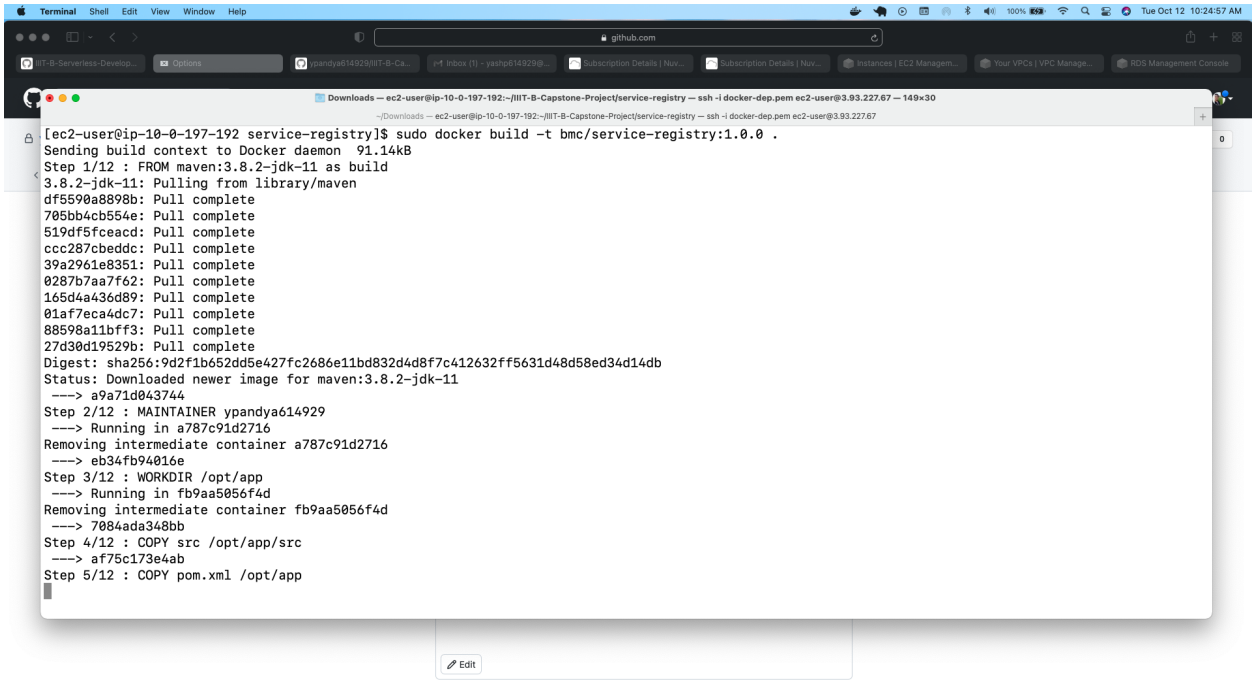
Service-Registry Image

The below commands are present in the service registry server Dockerfile.

```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/service-registry.jar
/opt/app/service-registry.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/service-registry.jar"]
```

The docker image of the service registry is built using the following command.

sudo docker build -t bmc/service-registry:1.0.0 .



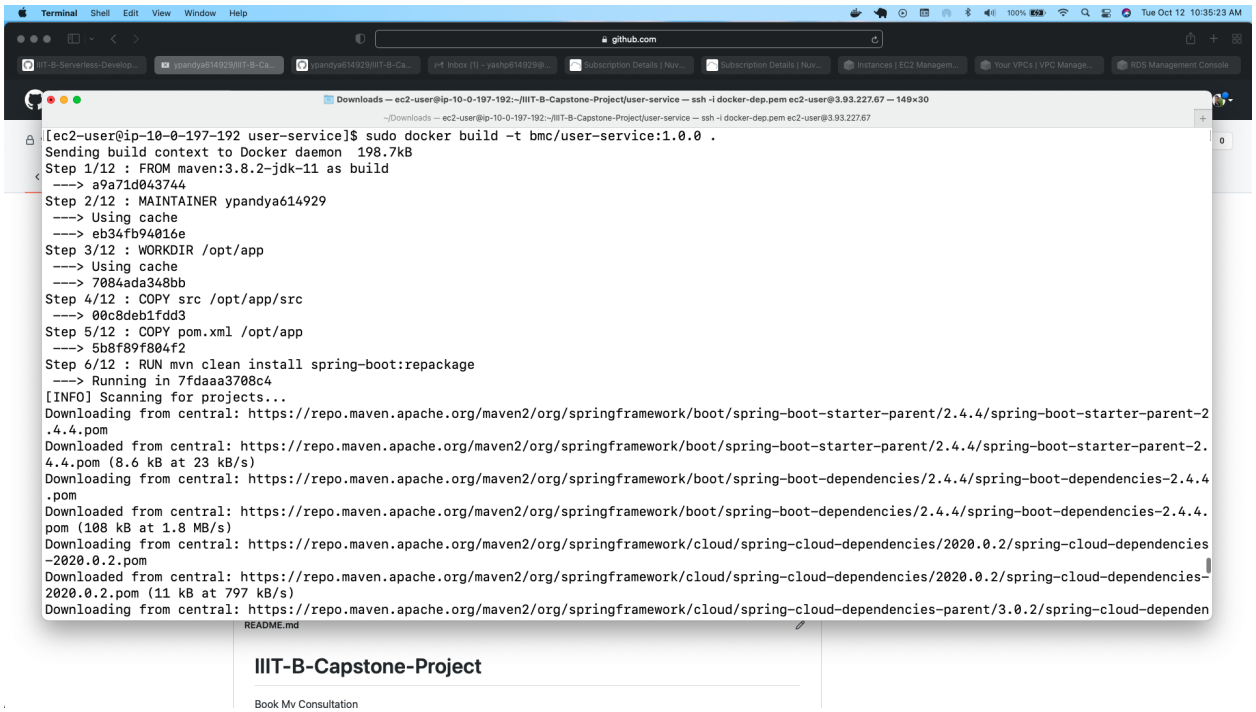
User-Service Image

The below commands are present in the security provider server Dockerfile.

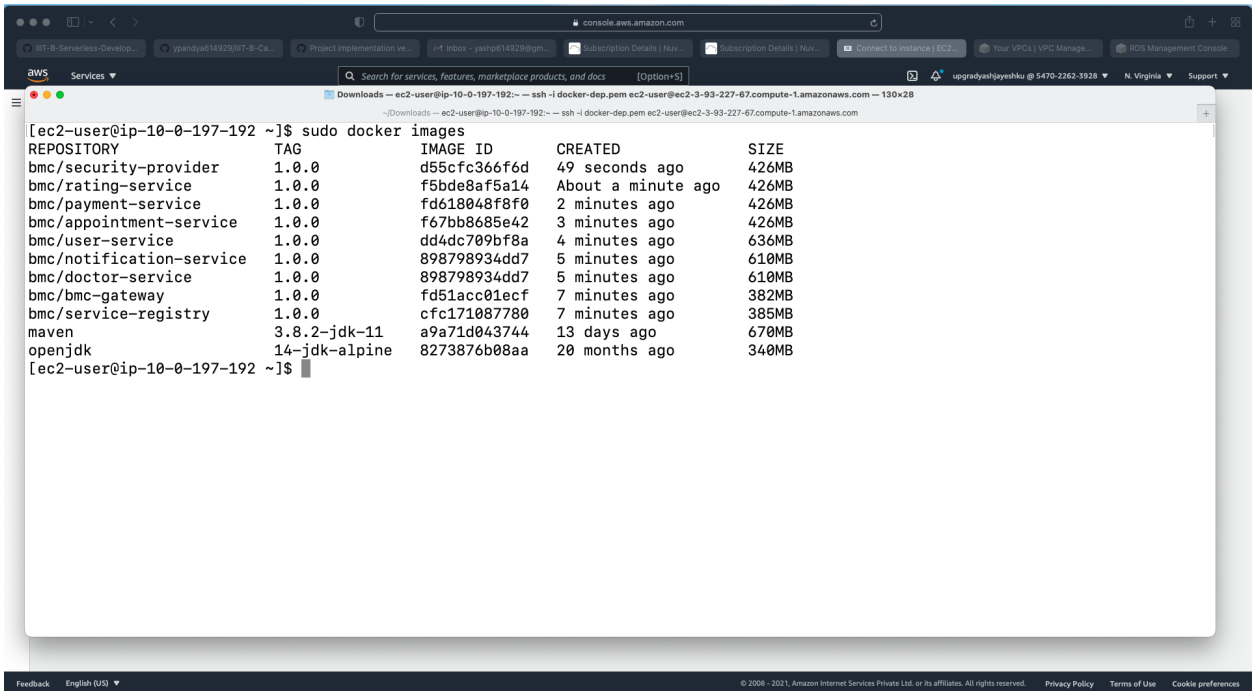
```
# 1
FROM maven:3.8.2-jdk-11 as build
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY src /opt/app/src
COPY pom.xml /opt/app
RUN mvn clean install spring-boot:repackage

# 2
FROM openjdk:14-jdk-alpine
MAINTAINER ypandya614929
WORKDIR /opt/app
COPY --from=build /opt/app/target/user-service.jar
/opt/app/user-service.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/app/user-service.jar"]
```

The docker image of the user service is built using the following command.

```
sudo docker build -t bmc/user-service:1.0.0 .
```



All the necessary images are built and can be verified as below.



Further, after building all the images of the micro services, the deployment is carried out by running the docker-compose file.

docker-compose file contains all the specifications related to containers and which image need to use to start/run the container. All networks and volumes can also be added into the file in-order to create based on the necessity.

The sequence of container creation can be specified via depend_on. Environment classified as a collection of variables that can be used while starting the application from the container. Port mapping is necessary in which one has to specify the port of the container and the mapping of that port with the host machine.

In our docker-compose file we have added `service-registry, bmc-gateway, notification-service,rating-service,user-service,doctor-service, appointment-service, payment-service, and security-provider.`

The **docker-compose file** is built using the below content.

```
version: '3'

services:
 service-registry:
   build: service-registry
   container_name: serviceregistry
   image: bmc/service-registry:1.0.0
   ports:
     - "8761:8761"
   networks:
     - book-my-consultation-nw
   environment:
     REGISTRY_HOST_NAME: 127.0.0.1
     REGISTRY_PORT: 8761

 bmc-gateway:
   build: bmc-gateway
   container_name: bmcgateway
   image: bmc/bmc-gateway:1.0.0
   ports:
     - "9191:9191"
   networks:
     - book-my-consultation-nw
   environment:
     REGISTRY_HOST_NAME: service-registry
     REGISTRY_PORT: 8761
   depends_on:
     - service-registry
```

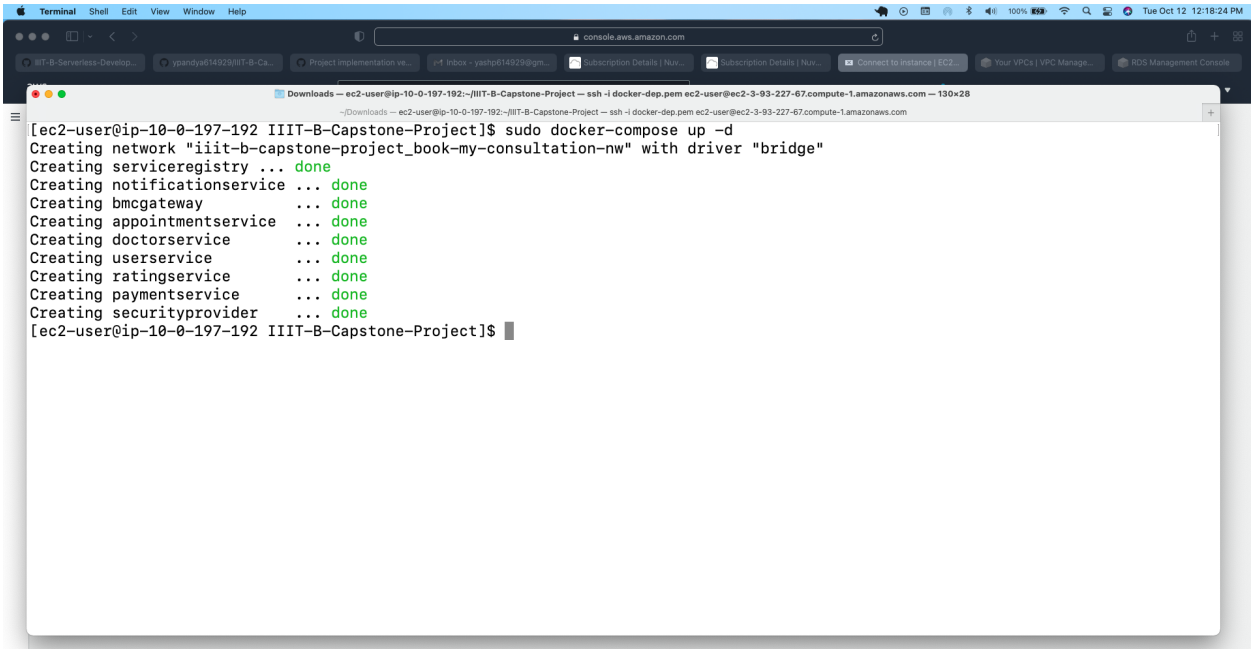
```yaml
notification-service:
  build: notification-service
  container_name: notificationservice
  image: bmc/notification-service:1.0.0
  ports:
    - "8085:8085"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    NOTIFICATION_PORT: 8085
  depends_on:
    - service-registry

rating-service:
  build: rating-service
  container_name: ratingservice
  image: bmc/rating-service:1.0.0
  ports:
    - "8084:8084"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    RATING_SVC_PORT: 8084
  depends_on:
    - service-registry
    - bmc-gateway
    - doctor-service

user-service:
  build: user-service
  container_name: userservice
  image: bmc/user-service:1.0.0
  ports:
    - "8083:8083"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    USER_SVC_PORT: 8083
  depends_on:
    - service-registry
    - bmc-gateway
    - notification-service
```

```
doctor-service:
  build: doctor-service
  container_name: doctorservice
  image: bmc/doctor-service:1.0.0
  ports:
    - "8081:8081"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    DOCTOR_SVC_PORT: 8081
  depends_on:
    - service-registry
    - bmc-gateway
    - notification-service

appointment-service:
  build: appointment-service
  container_name: appointmentservice
  image: bmc/appointment-service:1.0.0
  ports:
    - "8082:8082"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    APPOINTMENT_SVC_PORT: 8082
  depends_on:
    - service-registry
    - bmc-gateway
    - notification-service

payment-service:
  build: payment-service
  container_name: paymentservice
  image: bmc/payment-service:1.0.0
  ports:
    - "8086:8086"
  networks:
    - book-my-consultation-nw
  environment:
    REGISTRY_HOST_NAME: service-registry
    REGISTRY_PORT: 8761
    PAYMENT_SVC_PORT: 8086
  depends_on:
    - service-registry
```

```
      - bmc-gateway
      - appointment-service

  security-provider:
    build: security-provider
    container_name: securityprovider
    image: bmc/security-provider:1.0.0
    ports:
      - "8088:8088"
    networks:
      - book-my-consultation-nw
    environment:
      REGISTRY_HOST_NAME: service-registry
      REGISTRY_PORT: 8761
      SECURITY_PROVIDER_SVC_PORT: 8088
    depends_on:
      - service-registry
      - bmc-gateway
      - doctor-service
      - user-service
      - appointment-service
      - payment-service
      - rating-service


networks:
  book-my-consultation-nw:
    driver: bridge
```

To start the containers based on definition from the docker-compose can be done using the below command.

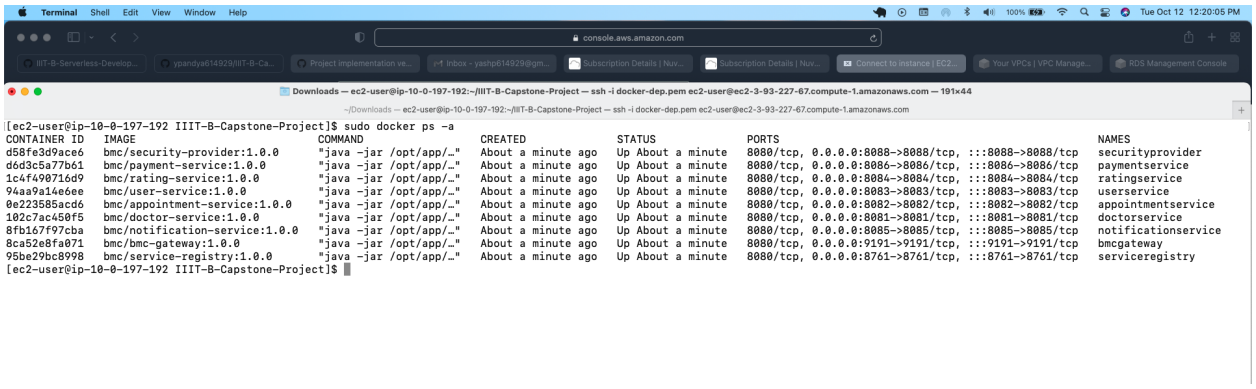
sudo docker-compose up -d



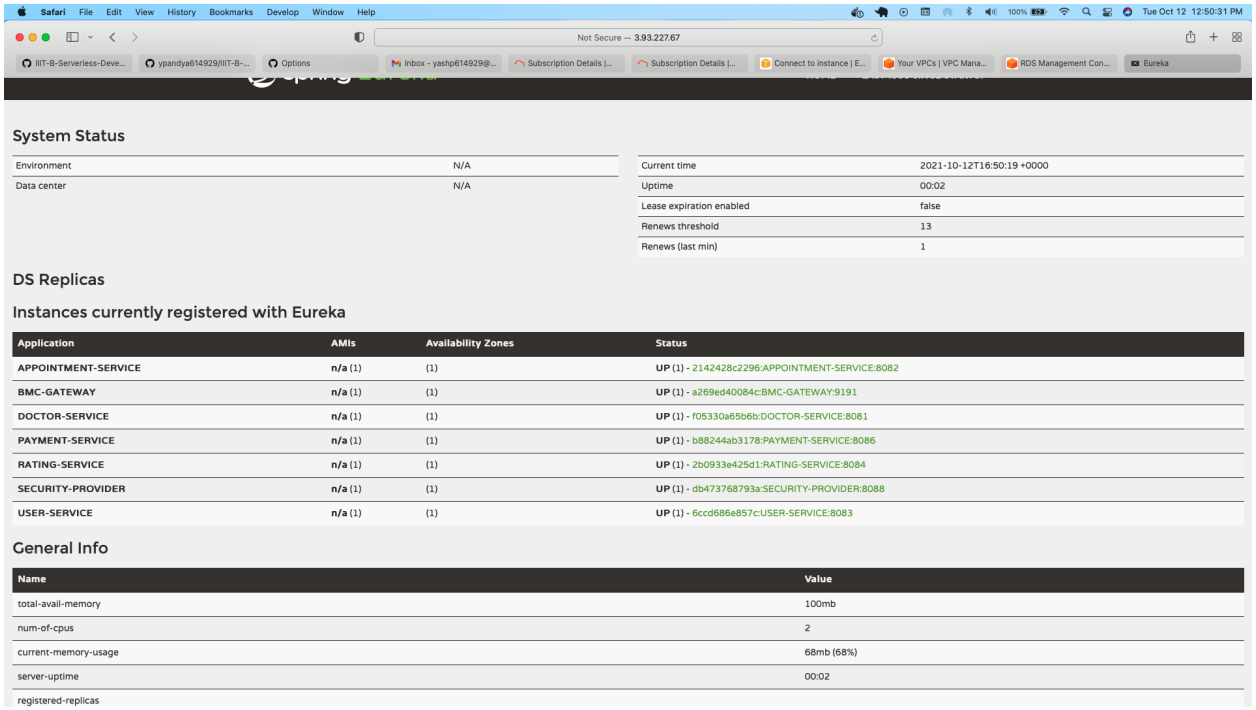
Here, a bridge type of network named `book-my-consultation-nw` is created as the docker-compose start building architecture.

All the services are up and running on specified ports and can be verified using below docker command.

`sudo docker ps -a`



Eureka can be viewed from the browser by accessing with the **docker-ec2** public IP which is Elastic IP **docker-ec2-ip** and the eureka port **8761**



Storage and Approach

All the documents from the Doctor and User services are being stored at **AWS S3**.

All micro services test cases are attached as below and data validation is also performed and verified using the MySQLWorkbench and/or MongoDB shell in order to validate data consistency.

- Notification service is connected with kafka and service notification is being sent and printed on the console once the other services write the log into the kafka topic and the notification service asynchronously connects and reads the data from the kafka topic.

- Appointment service also reads data from `**payment**` kafka topic once payment is being made, based on async call, Appointment service reads the appointment id from the kafka topic message and updates the status to Confirmed from PendingPayment.

- Doctor service also reads data from `**doctor**` kafka topic once rating is being given, based on async call, Doctor service reads the doctor id and average rating from the kafka topic message and updates the average rating of the doctor

Doctor creation, User creation, Doctor approval, Doctor rejection, Appointment confirmation and Prescription issue emails are being sent to respective users by using the **AWS SES service**.

To preview notification service logs, one has to preview the logs of the notification service container.

Docker has a built-in command to analyse the container logs. The command to preview the logs from the notification container is as shown below.

```
sudo docker logs notificationservice
```

Instructions To Run

- The RDS database URL needs to be mapped in all the services properties and username and password too (if changes).
- EC2 Elastic IP of **kafka-ec2** instance needs to be mapped in the KafkaConfig and Consumer classes.

- Kafka Consumer classes settings are mapped in application.properties as those consumers have producer and consumer both types of intrims
- Eureka/Service-Registry ie. **docker-ec2** Elastic IP needs to be mapped in the docker-compose file as **REGISTRY_HOST_NAME** within Environment section of **service-registry**

Once all the settings changed, including application properties for MongDB and RDS (MySQL) and above mentioned IP settings, one need to run build images command mentioned above for each services and then docker build command to start the container and all services are running on specified ports can be verified from the eureka URL.

Note

- Kafka is externalized using the two EC2 instances approach.

Verification

All the micro services are running on a specified port, all micro services are verified using the postman.

- User service endpoints are defined as below and running on port 8083

  POST /users
  GET /users/{userId}
  POST /users/{id}/documents
  GET /users/{id}/documents/{documentName}
  GET /users/{id}/documents/metadata

- Doctor service endpoints are defined as below and running on port 8081

  POST /doctors
  POST /doctors/{doctorId}/documents
  PUT /doctors/{doctorId}/approve
  PUT /doctors/{doctorId}/reject
  GET /doctors
  GET /doctors/{doctorId}
  GET /doctors/{doctorId}/documents/metadata
  GET /doctors/{doctorId}/documents/{documentName}

- Appointment service endpoints are defined as below and running on port 8082

  POST /doctor/{doctorId}/availability
  GET /doctor/{doctorId}/availability
  POST /appointments
  GET /appointments/{appointmentId}
  GET /users/{userId}/appointments
  POST /prescriptions

- Payment service endpoints are defined as below and running on port 8086

  POST /payments

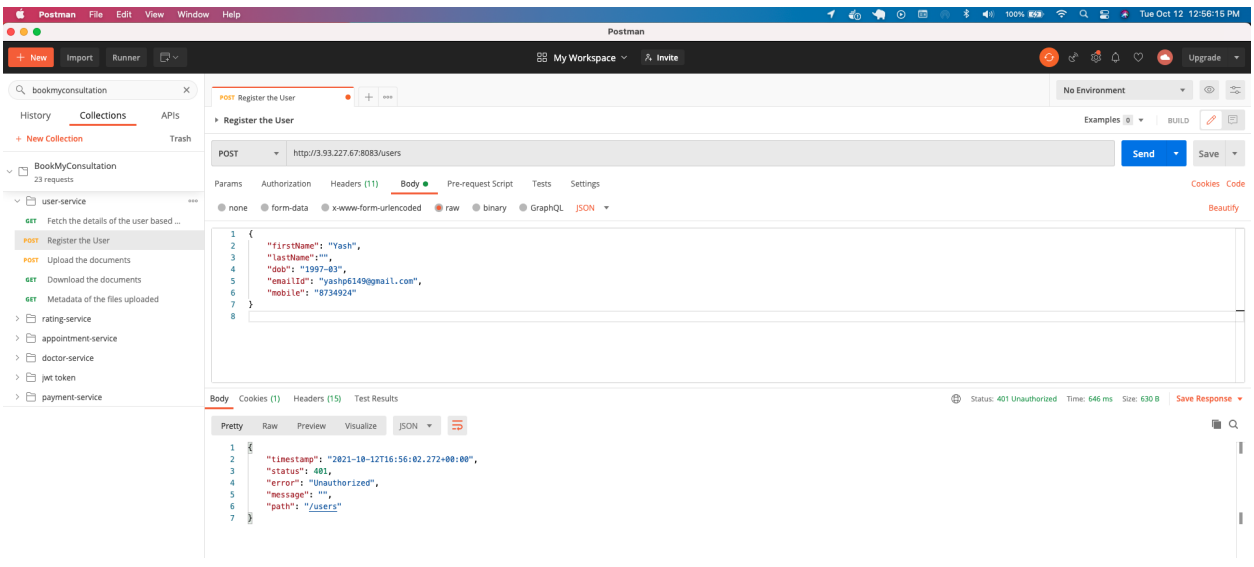
- Rating service endpoints are defined as below and running on port 8084

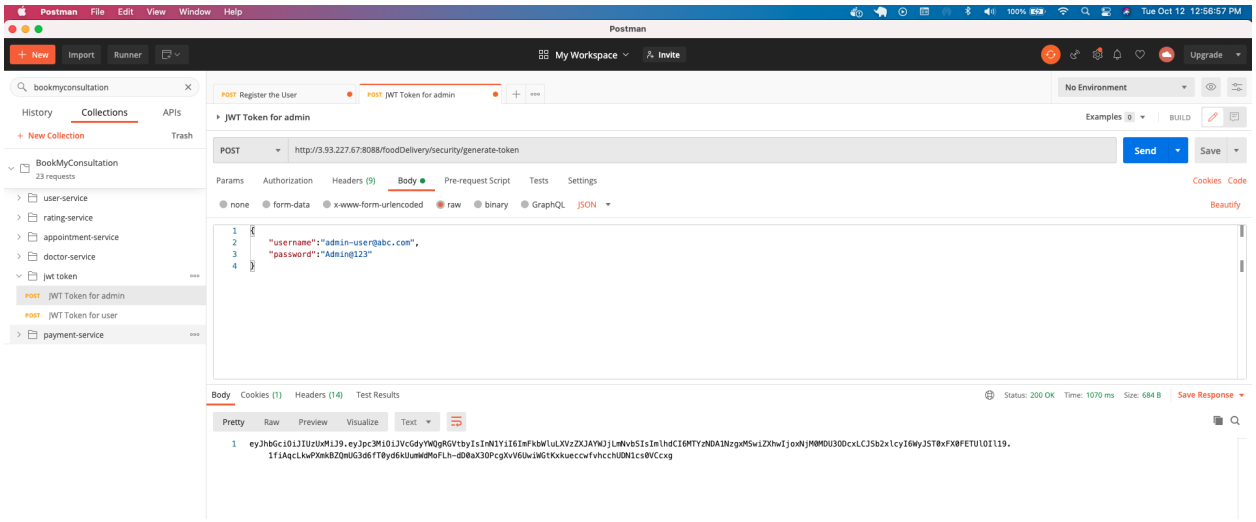
  POST /ratings
- Security-Provider service endpoints are defined as below and running on port 8088

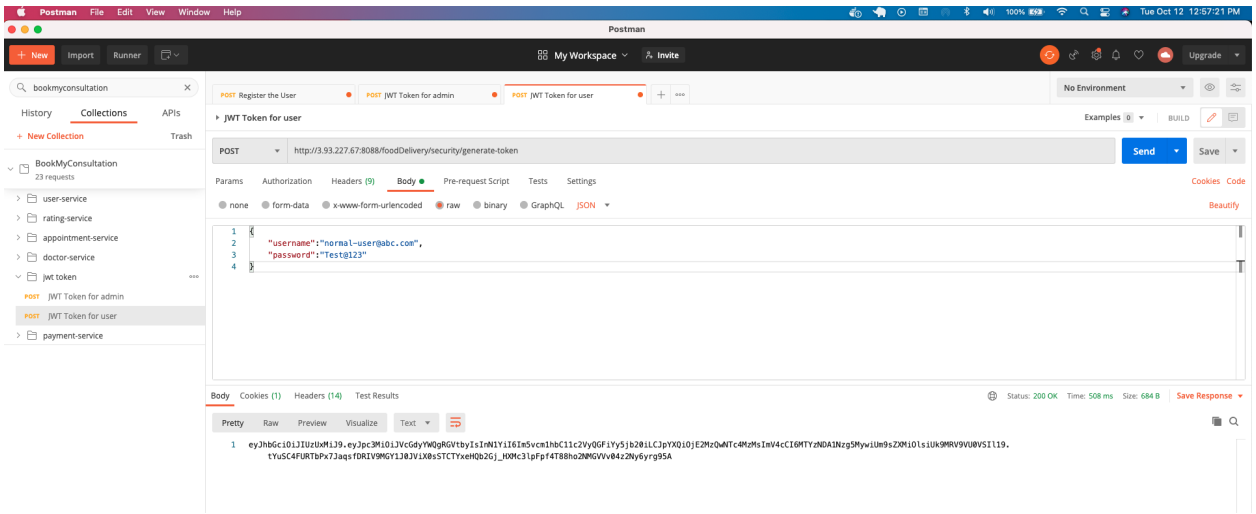
  POST /generate-token
  GET /user/all

- API Gateway running on port 9191 and redirects the traffic based on the matching path

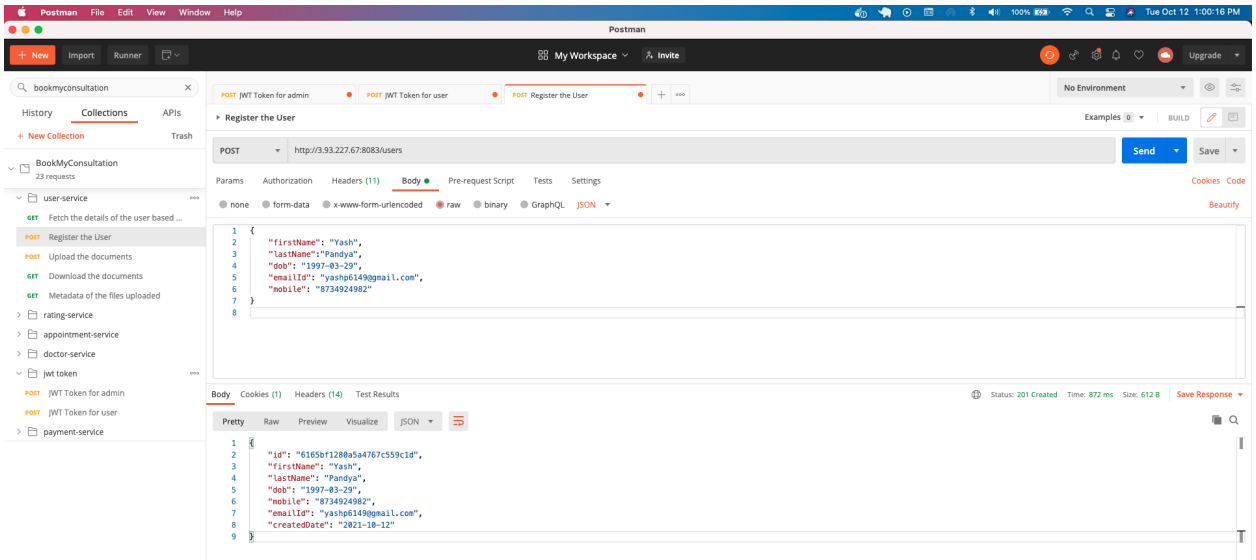
Test Results

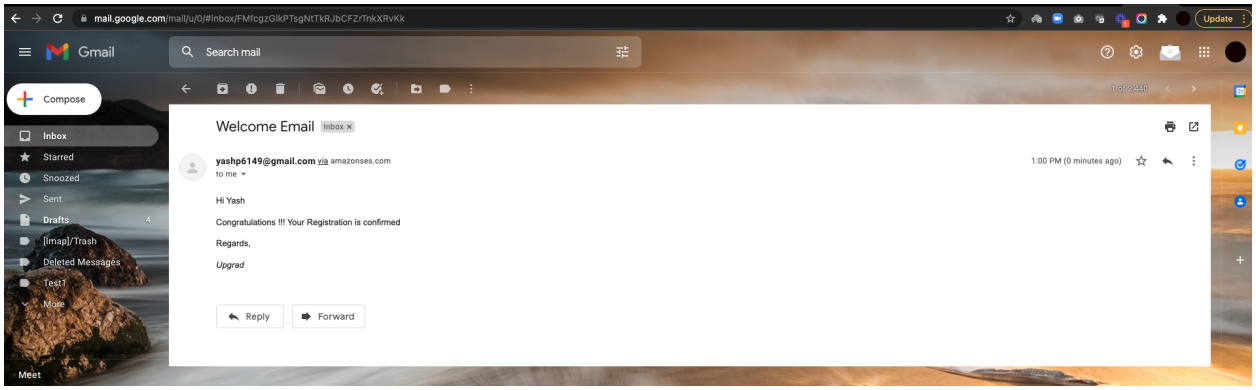


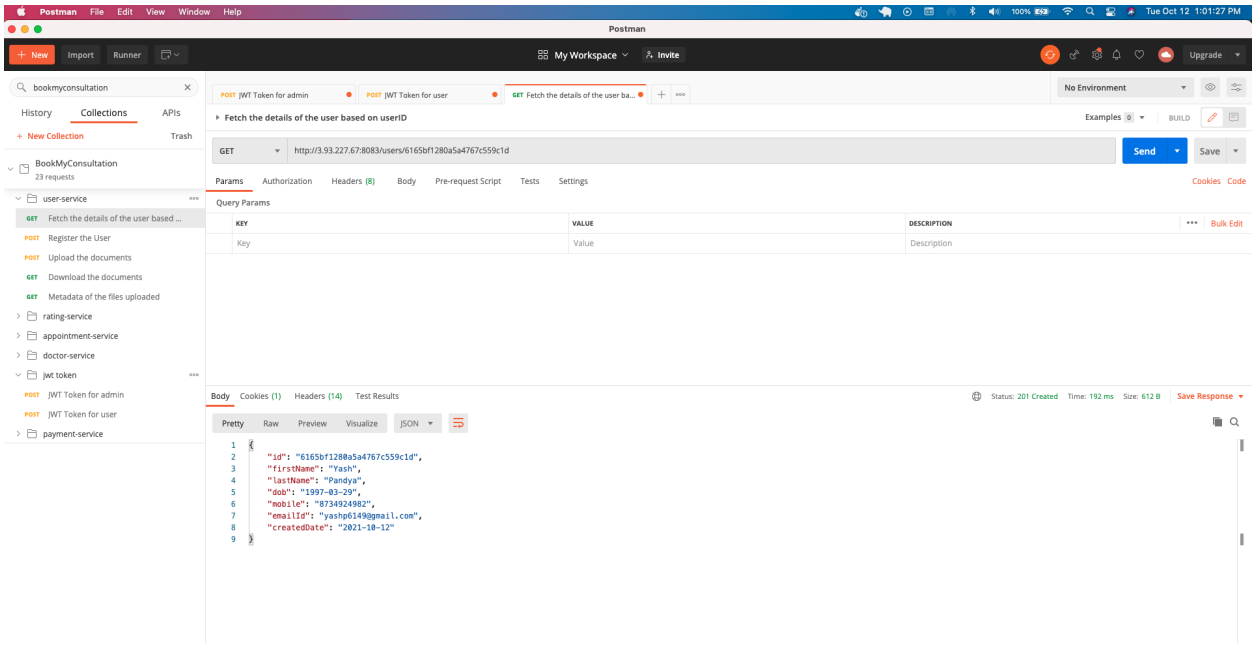


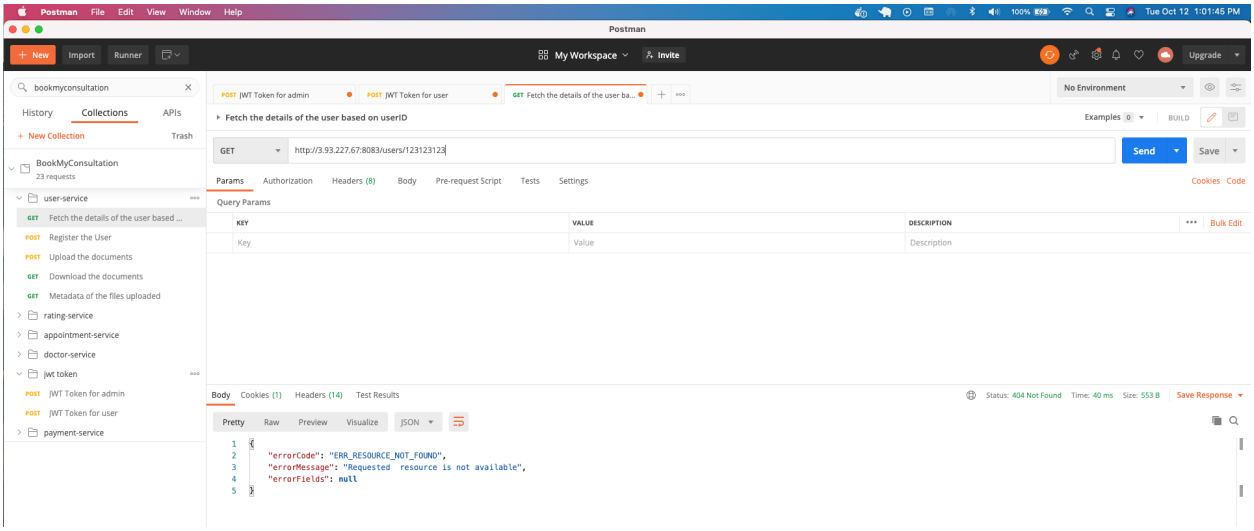


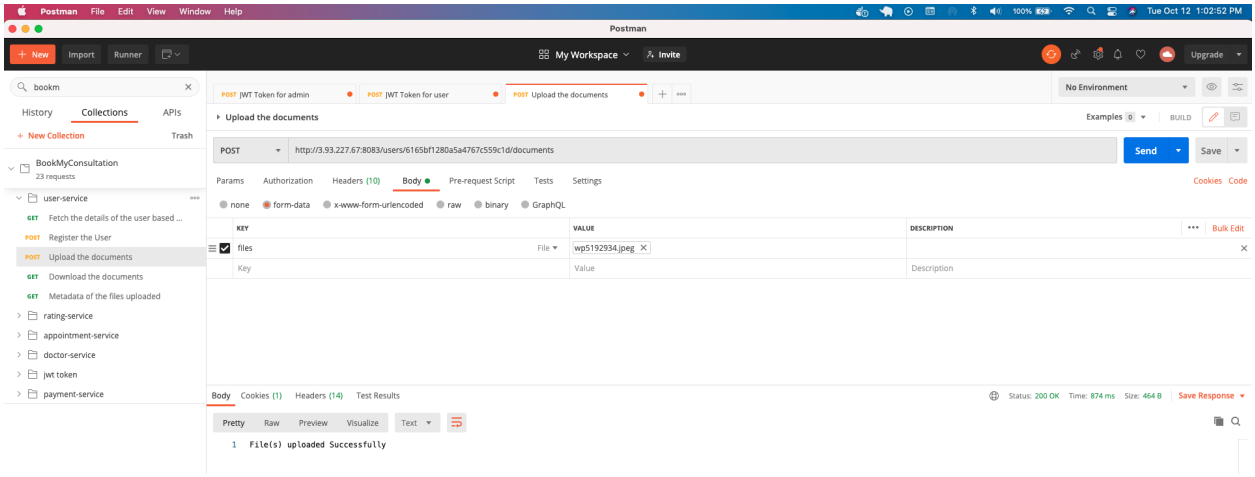
Please note that Token service only generates the token whereas all other services can validate the token separately.

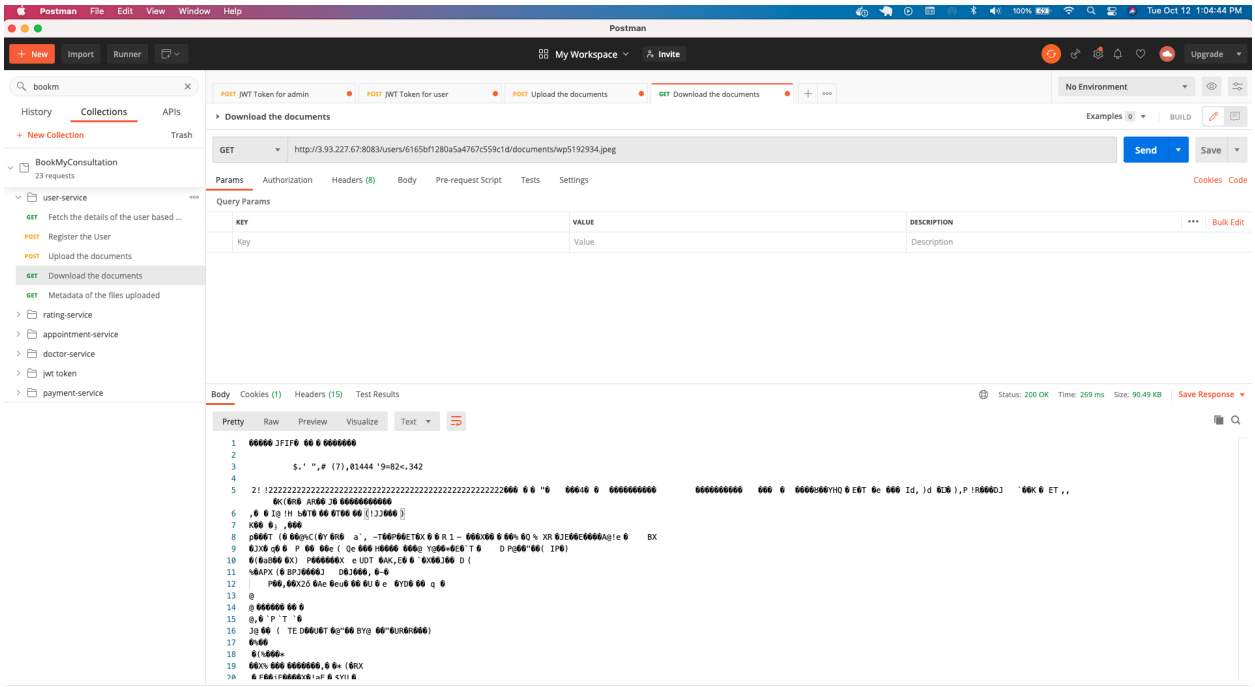


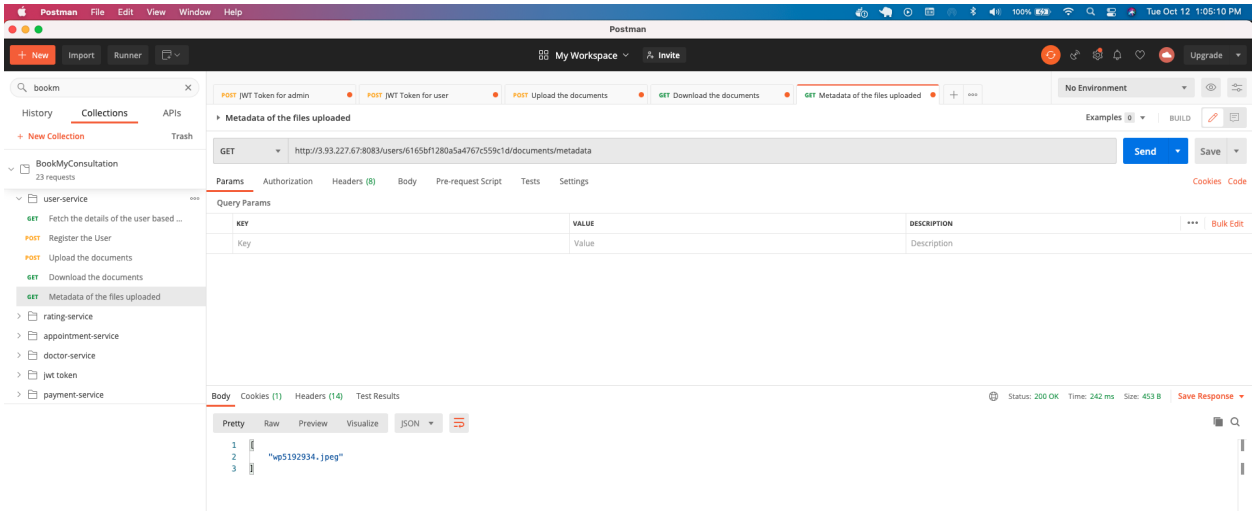

mail.google.com/mail/u/0/#inbox/FMfcgzGlkPTsgNtTkRJbCFZrTnkXRvKk
Gmail
Search mail
Compose
Inbox
Starred
Snoozed
Sent
Drafts
[Imap]/Trash
Deleted Messages
Test1
More
Meet
Welcome Email
Inbox ×
yashp6149@gmail.com via amazonses.com
to me
1:00 PM (0 minutes ago)
Hi Yash
Congratulations !!! Your Registration is confirmed
Regards,
Upgrad
Reply
Forward

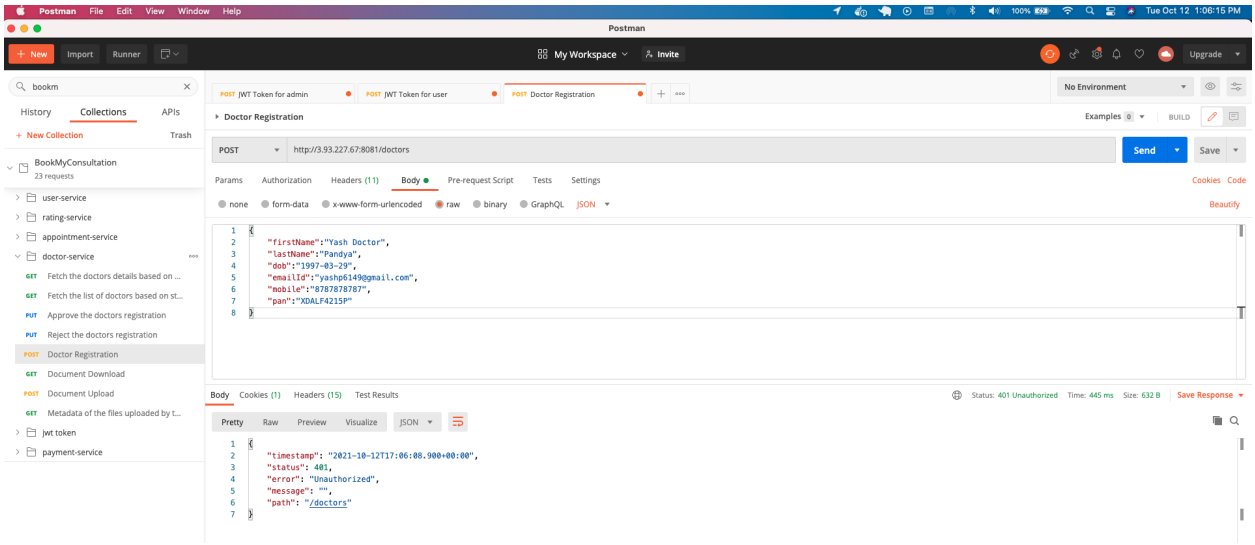

Postman
My Workspace
Invite
Upgrade
bookmyconsultation
History
Collections
APIs
New Collection
Trash
BookMyConsultation
23 requests
user-service
Fetch the details of the user based ...
Register the User
Upload the documents
Download the documents
Metadata of the files uploaded
rating-service
appointment-service
doctor-service
jwt token
JWT Token for admin
JWT Token for user
payment-service
Fetch the details of the user based on userID
GET
http://3.93.227.67:8083/users/6165bf1280a5a4767c559c1d
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies (1)
Headers (14)
Test Results
Status: 201 Created
Time: 192 ms
Size: 612 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": "6165bf1280a5a4767c559c1d",
"firstName": "Yash",
"lastName": "Pandya",
"dob": "1997-03-29",
"mobile": "8734924982",
"emailId": "yashp6149@gmail.com",
"createdDate": "2021-10-12"
}

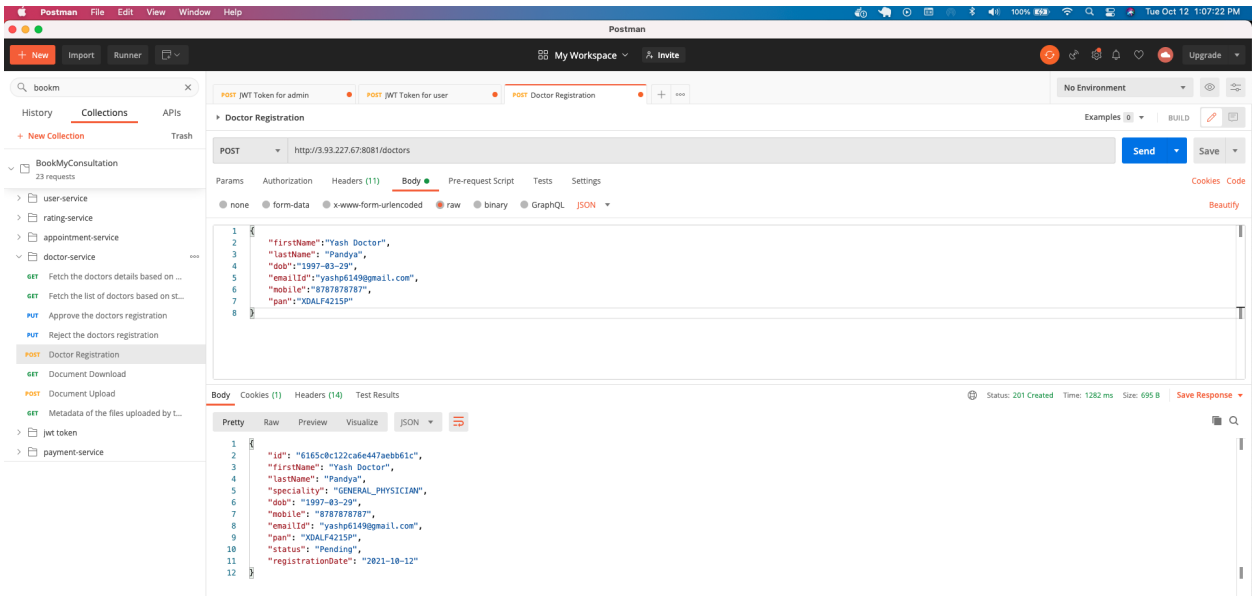

Postman
My Workspace
Invite
Upgrade
bookmyconsultation
History
Collections
APIs
New Collection
Trash
BookMyConsultation
23 requests
user-service
Fetch the details of the user based ...
Register the User
Upload the documents
Download the documents
Metadata of the files uploaded
rating-service
appointment-service
doctor-service
jwt token
JWT Token for admin
JWT Token for user
payment-service
Fetch the details of the user based on userID
GET
http://3.93.227.67:8083/users/123123123
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies (1)
Headers (14)
Test Results
Status: 404 Not Found
Time: 40 ms
Size: 553 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"errorCode": "ERR_RESOURCE_NOT_FOUND",
"errorMessage": "Requested resource is not available",
"errorFields": null
}

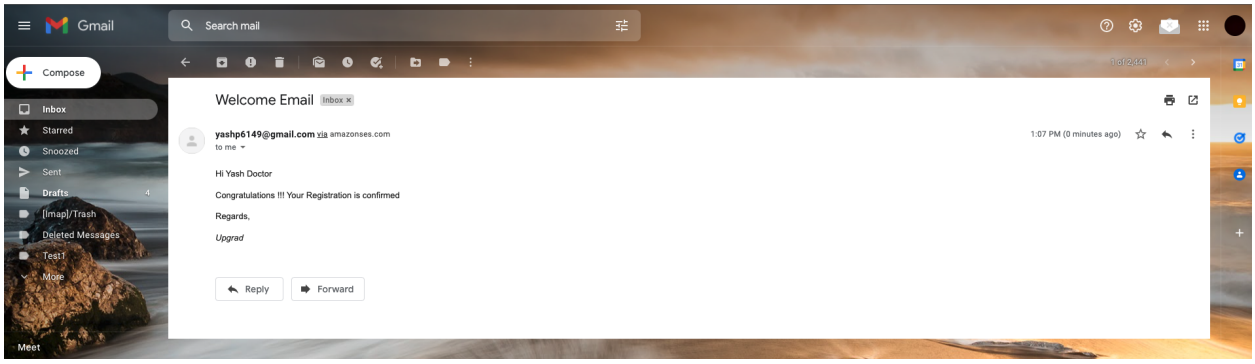

Postman
POST JWT Token for admin
POST JWT Token for user
POST Upload the documents
Upload the documents
POST
http://3.93.227.67:8083/users/6165bf1280a5a4767c559c1d/documents
Send
Save
Params Authorization Headers (10) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL
KEY VALUE DESCRIPTION
files File wp5192934.jpeg
Body Cookies (1) Headers (14) Test Results
Status: 200 OK Time: 874 ms Size: 464 B
Pretty Raw Preview Visualize Text
1 File(s) uploaded Successfully

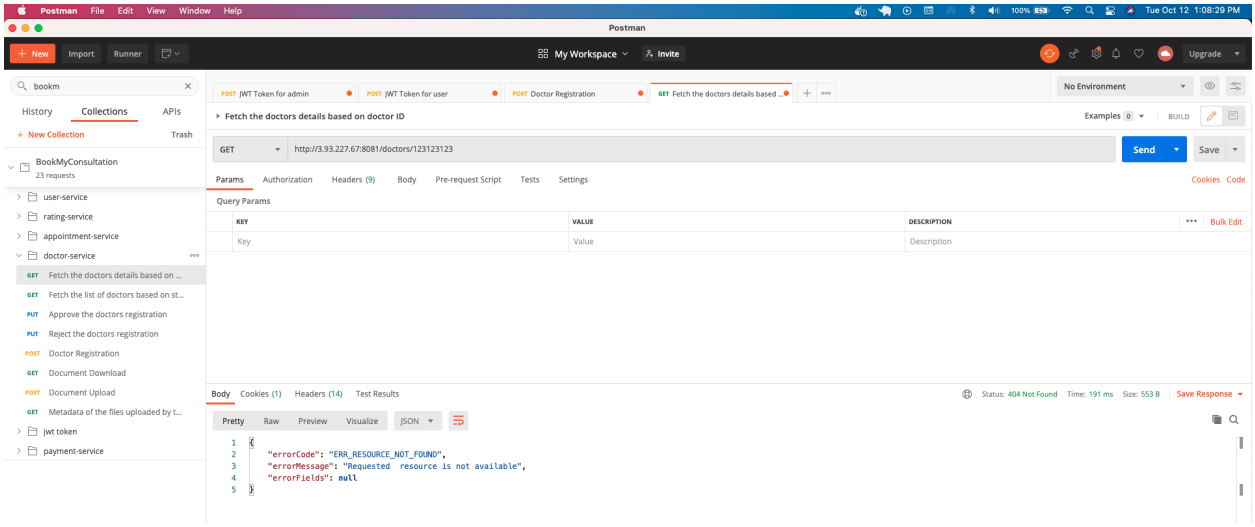

Postman
POST JWT Token for admin
POST JWT Token for user
POST Upload the documents
GET Download the documents
Download the documents
GET
http://3.93.227.67:8083/users/6165bf1280a5a4767c559c1d/documents/wp5192934.jpeg
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Query Params
KEY VALUE DESCRIPTION
Body Cookies (1) Headers (15) Test Results
Status: 200 OK Time: 269 ms Size: 90.49 KB
Pretty Raw Preview Visualize Text

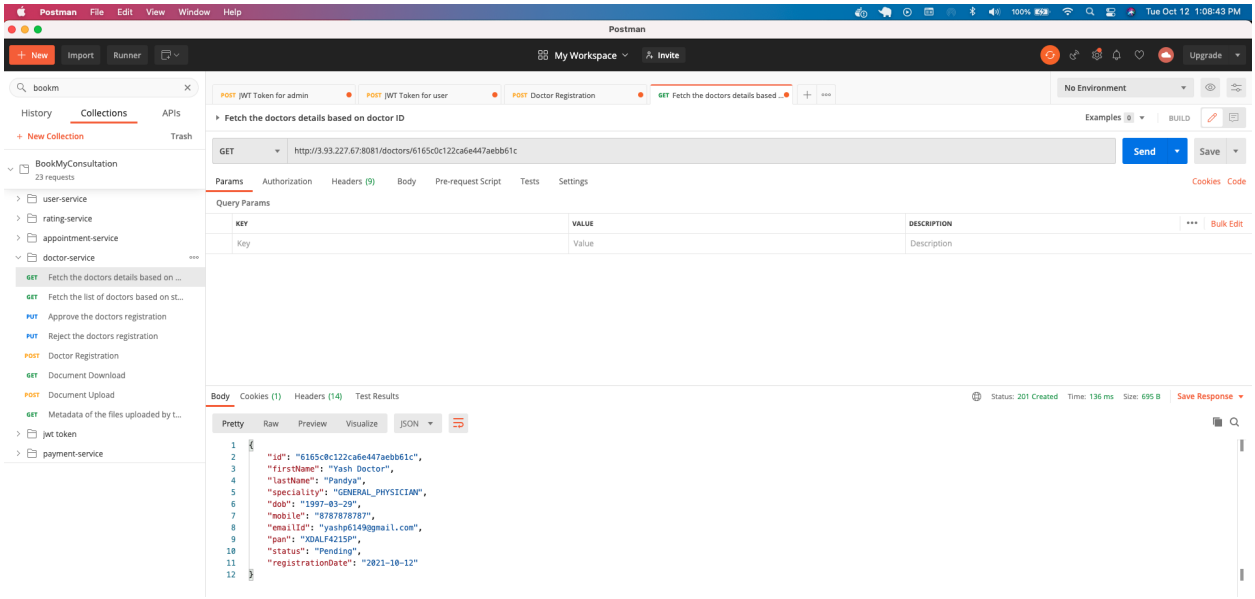

Postman
POST JWT Token for admin
POST JWT Token for user
POST Upload the documents
GET Download the documents
GET Metadata of the files uploaded
Metadata of the files uploaded
GET
http://3.93.227.67:8083/users/6165bf1280a5a4767c559c1d/documents/metadata
Send
Save
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Query Params
KEY VALUE DESCRIPTION
Body Cookies (1) Headers (14) Test Results
Status: 200 OK Time: 242 ms Size: 453 B
Pretty Raw Preview Visualize JSON
{
"wp5192934.jpeg"
}

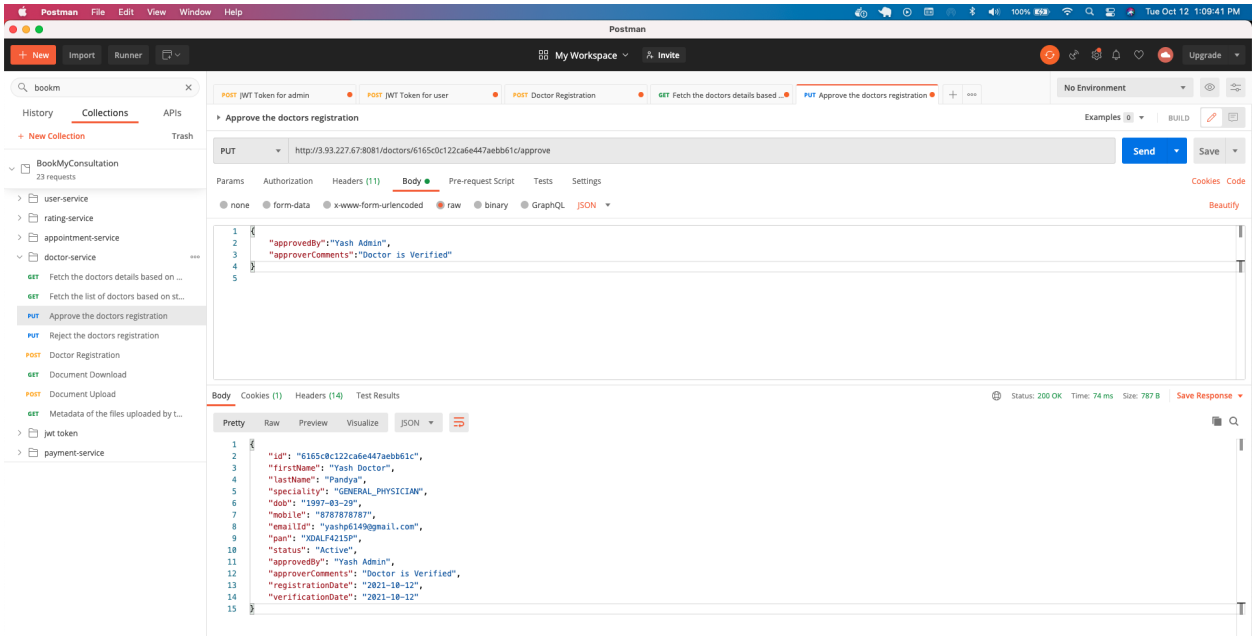

Postman
Doctor Registration
POST
http://3.93.227.67:8081/doctors
Send
Save
{
"firstName":"Yash Doctor",
"lastName":"Pandya",
"dob":"1997-03-29",
"emailId":"yashp6149@gmail.com",
"mobile":"8787878787",
"pan":"XDALF4215P"
}
Status: 401 Unauthorized
Time: 445 ms
Size: 632 B
{
"timestamp": "2021-10-12T17:06:08.900+00:00",
"status": 401,
"error": "Unauthorized",
"message": "",
"path": "/doctors"
}

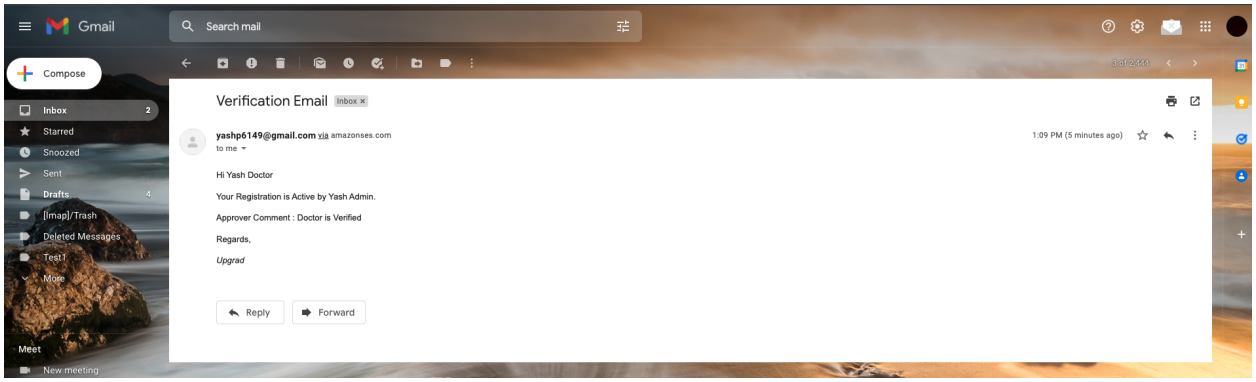

Postman
Doctor Registration
POST
http://3.93.227.67:8081/doctors
Send
Save
{
"firstName":"Yash Doctor",
"lastName": "Pandya",
"dob":"1997-03-29",
"emailId":"yashp6149@gmail.com",
"mobile":"8787878787",
"pan":"XDALF4215P"
}
Status: 201 Created
Time: 1282 ms
Size: 695 B
{
"id": "6165c0c122ca6e447aebb61c",
"firstName": "Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "8787878787",
"emailId": "yashp6149@gmail.com",
"pan": "XDALF4215P",
"status": "Pending",
"registrationDate": "2021-10-12"
}

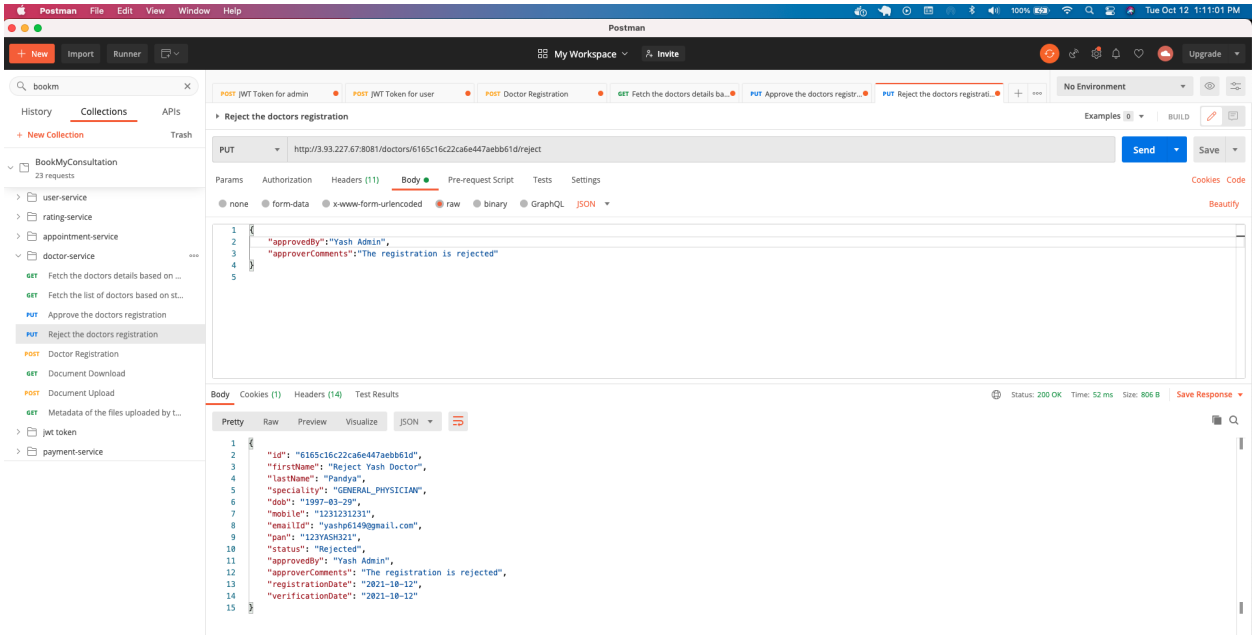

Gmail
Search mail
Compose
Inbox
Starred
Snoozed
Sent
Drafts
[Imap]/Trash
Deleted Messages
Test1
More
Welcome Email
Inbox
yashp6149@gmail.com via amazonses.com
to me
1:07 PM (0 minutes ago)
Hi Yash Doctor
Congratulations !!! Your Registration is confirmed
Regards,
Upgrad
Reply
Forward
Meet

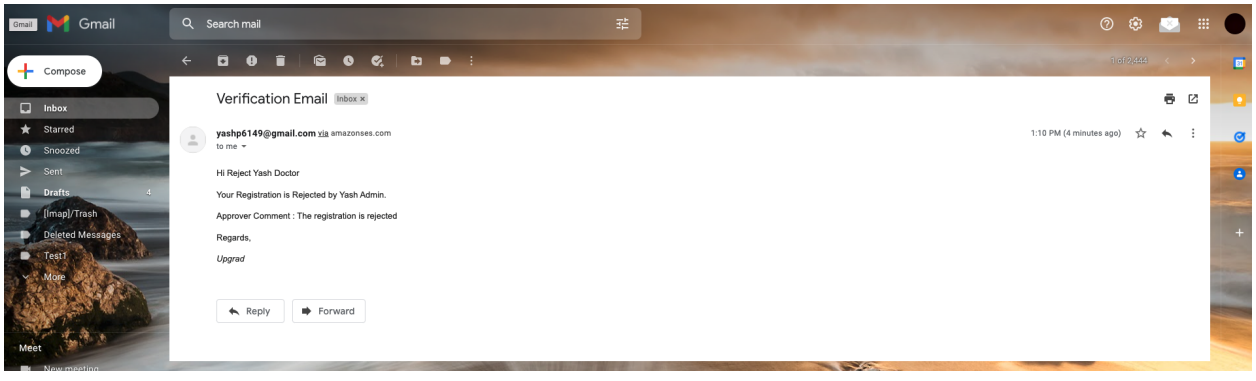

Fetch the doctors details based on doctor ID
GET
http://3.93.227.67:8081/doctors/123123123
Send
Save
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Query Params
KEY VALUE DESCRIPTION
Body Cookies (1) Headers (14) Test Results
Status: 404 Not Found Time: 191 ms Size: 553 B
Pretty Raw Preview Visualize JSON
{
"errorCode": "ERR_RESOURCE_NOT_FOUND",
"errorMessage": "Requested resource is not available",
"errorFields": null
}

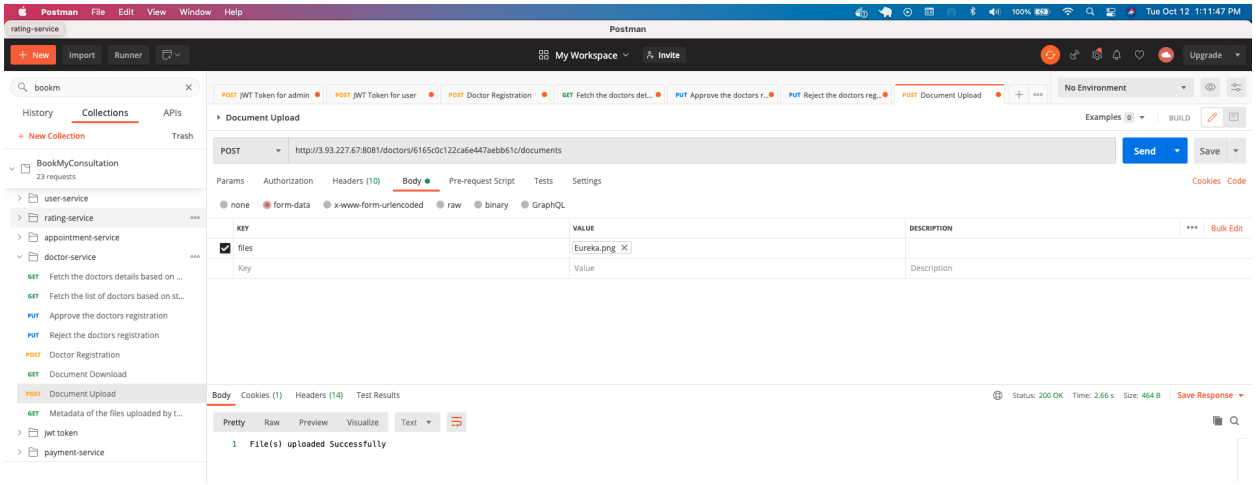

Fetch the doctors details based on doctor ID
GET
http://3.93.227.67:8081/doctors/6165c0c122ca6e447aebb61c
Send
Save
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Query Params
KEY VALUE DESCRIPTION
Body Cookies (1) Headers (14) Test Results
Status: 201 Created Time: 136 ms Size: 695 B
Pretty Raw Preview Visualize JSON
{
"id": "6165c0c122ca6e447aebb61c",
"firstName": "Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "8787878787",
"emailId": "yashp6149@gmail.com",
"pan": "XDALF4215P",
"status": "Pending",
"registrationDate": "2021-10-12"
}

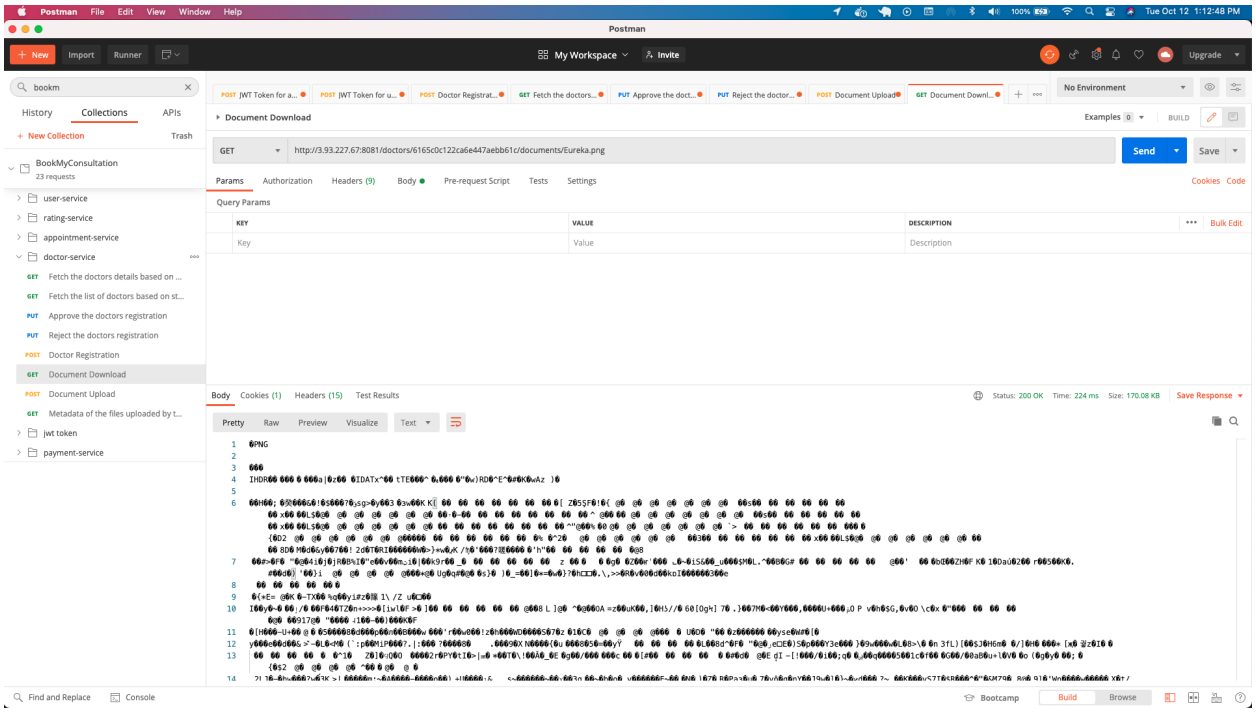

Approve the doctors registration
PUT
http://3.93.227.67:8081/doctors/6165c0c122ca6e447aebb61c/approve
Send
Save
Params Authorization Headers (11) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"approvedBy":"Yash Admin",
"approverComments":"Doctor is Verified"
}
Body Cookies (1) Headers (14) Test Results
Status: 200 OK Time: 74 ms Size: 787 B
Pretty Raw Preview Visualize JSON
{
"id": "6165c0c122ca6e447aebb61c",
"firstName": "Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "8787878787",
"emailId": "yashp6149@gmail.com",
"pan": "XDALF4215P",
"status": "Active",
"approvedBy": "Yash Admin",
"approverComments": "Doctor is Verified",
"registrationDate": "2021-10-12",
"verificationDate": "2021-10-12"
}

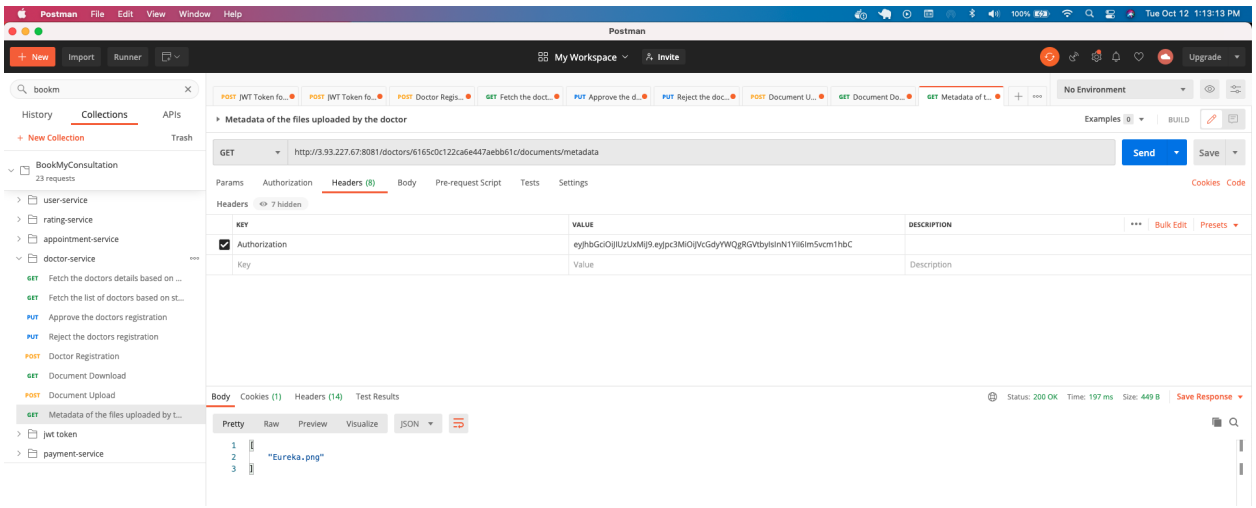

Gmail
Search mail
Compose
Inbox 2
Starred
Snoozed
Sent
Drafts 4
[Imap]/Trash
Deleted Messages
Test1
More
Meet
New meeting
Verification Email
Inbox ×
yashp6149@gmail.com via amazonses.com
to me
1:09 PM (5 minutes ago)
Hi Yash Doctor
Your Registration is Active by Yash Admin.
Approver Comment : Doctor is Verified
Regards,
Upgrad
Reply
Forward

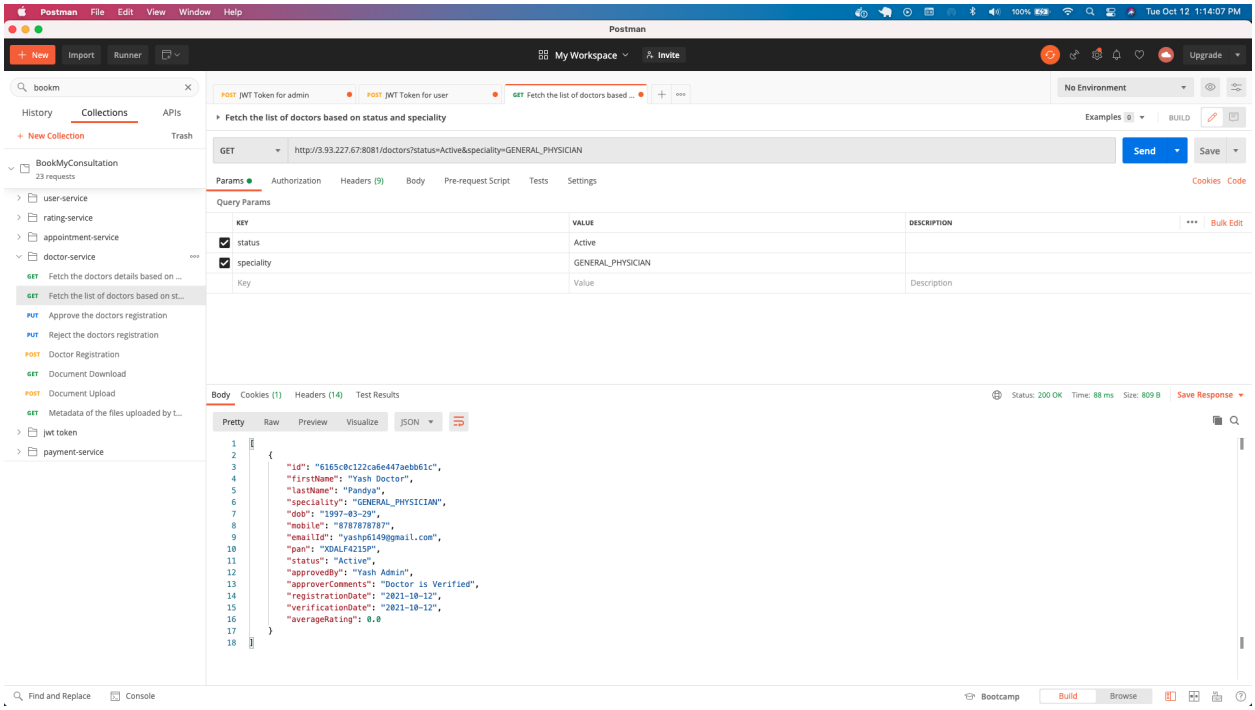

Postman
File Edit View Window Help
Tue Oct 12 1:11:01 PM
New
Import
Runner
My Workspace
Invite
Upgrade
bookm
History
Collections
APIs
New Collection
Trash
BookMyConsultation
23 requests
user-service
rating-service
appointment-service
doctor-service
Fetch the doctors details based on ...
Fetch the list of doctors based on st...
Approve the doctors registration
Reject the doctors registration
Doctor Registration
Document Download
Document Upload
Metadata of the files uploaded by t...
jwt token
payment-service
JWT Token for admin
JWT Token for user
Doctor Registration
Fetch the doctors details ba...
Approve the doctors registr...
Reject the doctors registrati...
No Environment
Reject the doctors registration
Examples 0
BUILD
PUT
http://3.93.227.67:8081/doctors/6165c16c22ca6e447aebb61d/reject
Send
Save
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
Cookies
Code
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"approvedBy":"Yash Admin",
"approverComments":"The registration is rejected"
}
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 52 ms
Size: 806 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": "6165c16c22ca6e447aebb61d",
"firstName": "Reject Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "1231231231",
"emailId": "yashp6149@gmail.com",
"pan": "123YASH321",
"status": "Rejected",
"approvedBy": "Yash Admin",
"approverComments": "The registration is rejected",
"registrationDate": "2021-10-12",
"verificationDate": "2021-10-12"
}

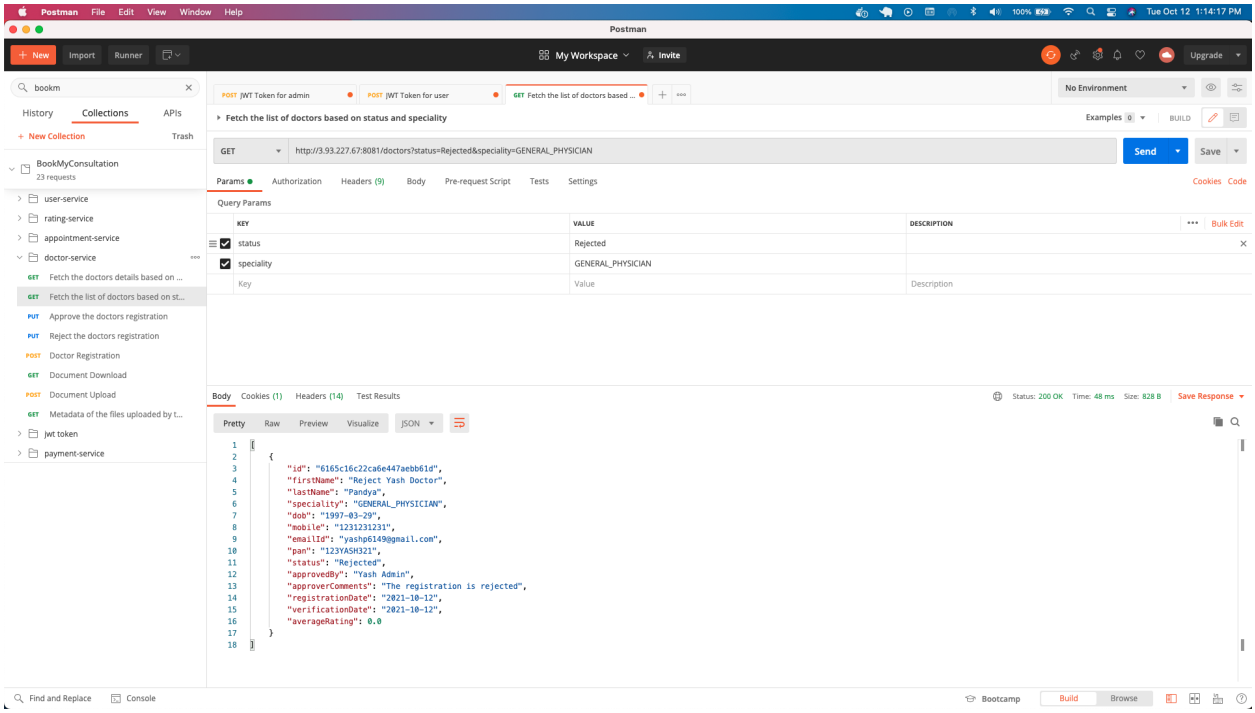

Gmail
Search mail
Compose
Inbox
Starred
Snoozed
Sent
Drafts 4
[Imap]/Trash
Deleted Messages
Test1
More
Meet
New meeting
Verification Email
Inbox ×
yashp6149@gmail.com via amazonses.com
to me
1:10 PM (4 minutes ago)
Hi Reject Yash Doctor
Your Registration is Rejected by Yash Admin.
Approver Comment : The registration is rejected
Regards,
Upgrad
Reply
Forward

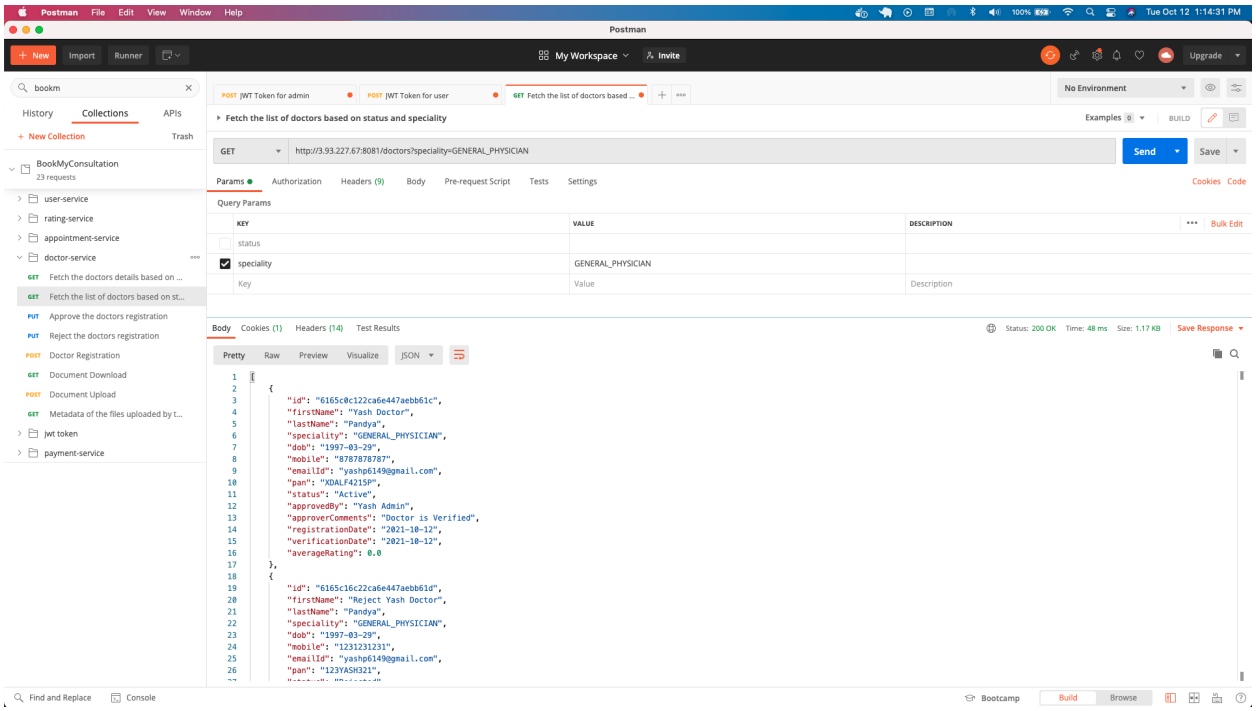

Postman
rating-service
POST Document Upload
http://3.93.227.67:8081/doctors/6165c0c122ca6e447aebb61c/documents
Body
form-data
files
Eureka.png
Status: 200 OK
Time: 2.66 s
Size: 464 B
File(s) uploaded Successfully

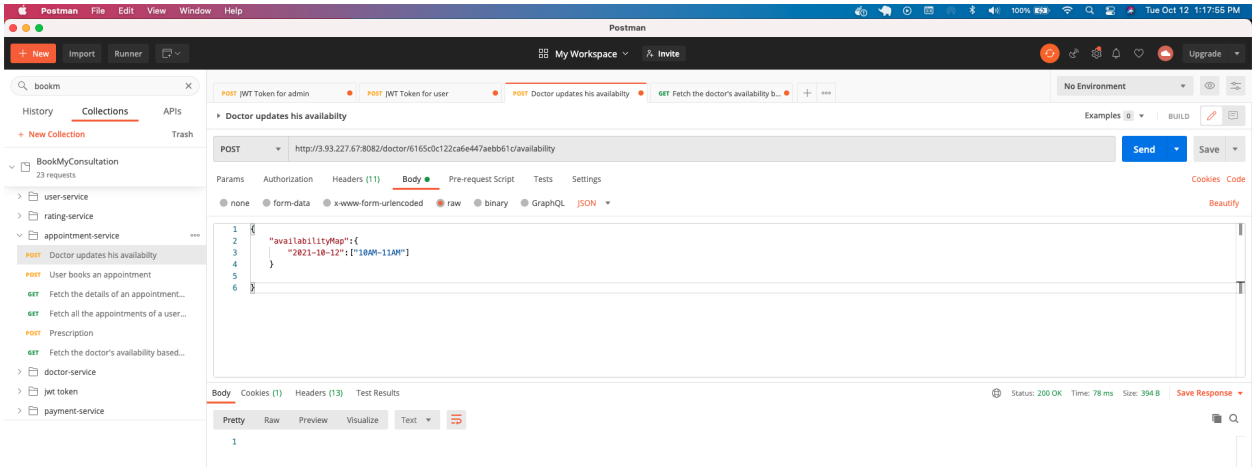

Postman
GET Document Download
http://3.93.227.67:8081/doctors/6165c0c122ca6e447aebb61c/documents/Eureka.png
Status: 200 OK
Time: 224 ms
Size: 170.08 KB

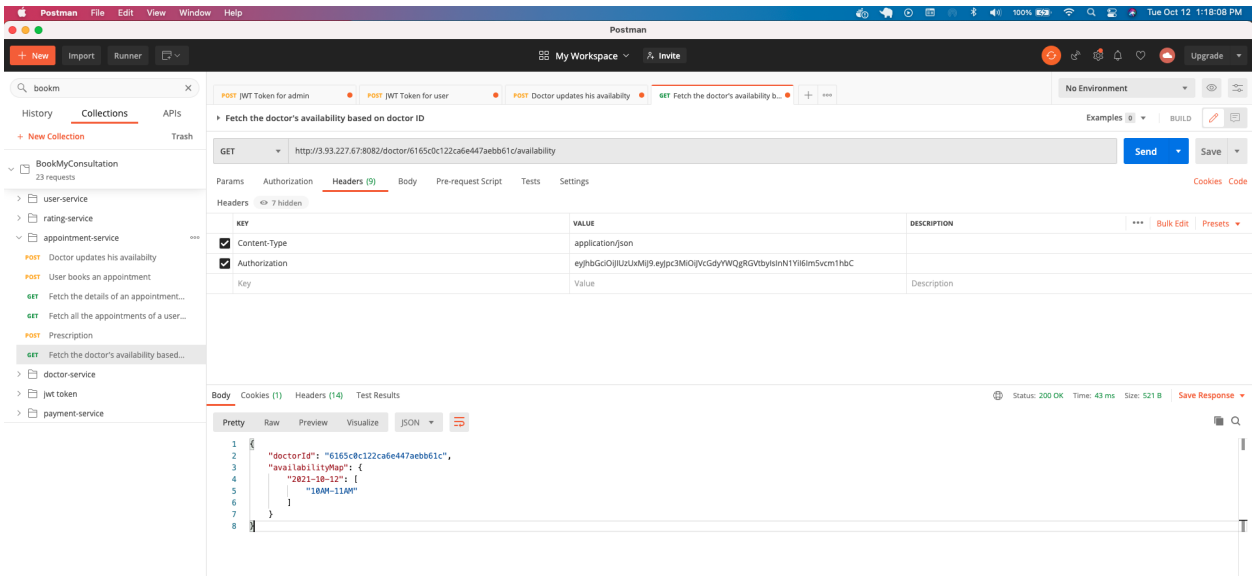

Postman
GET Metadata of the files uploaded by the doctor
http://3.93.227.67:8081/doctors/6165c0c122ca6e447aebb61c/documents/metadata
Authorization
eyJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJlVcGdyYWQgRGVtbylsInN1YiI6Im5vcm1hbC
Status: 200 OK
Time: 197 ms
Size: 449 B
{
"Eureka.png"
}

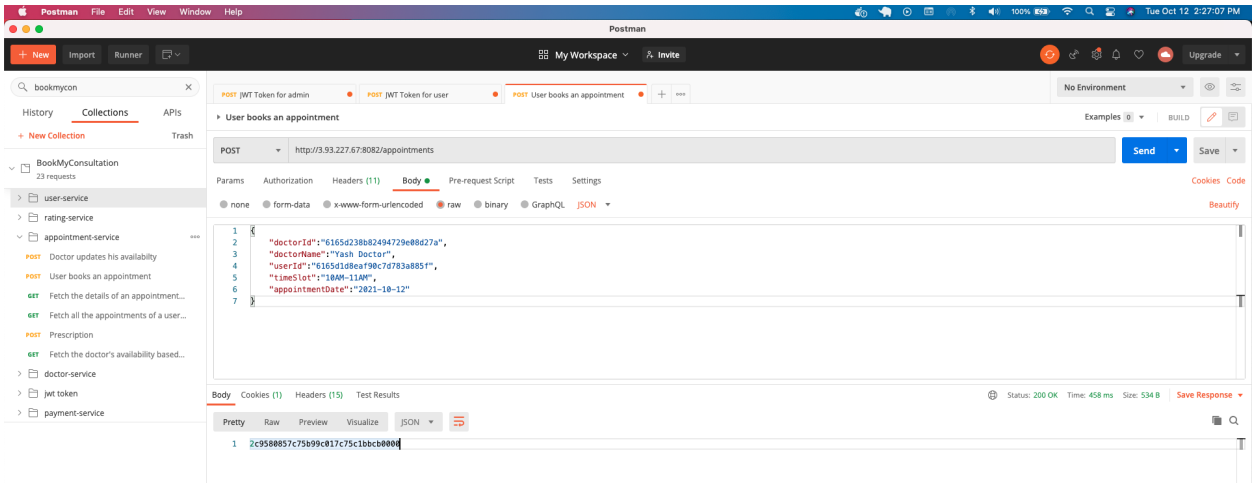

Postman
GET http://3.93.227.67:8081/doctors?status=Active&speciality=GENERAL_PHYSICIAN
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
status
Active
speciality
GENERAL_PHYSICIAN
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 88 ms
Size: 809 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": "6165c0c122ca6e447aebb61c",
"firstName": "Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "8787878787",
"emailId": "yashp6149@gmail.com",
"pan": "XDALF4215P",
"status": "Active",
"approvedBy": "Yash Admin",
"approverComments": "Doctor is Verified",
"registrationDate": "2021-10-12",
"verificationDate": "2021-10-12",
"averageRating": 0.0
}
]

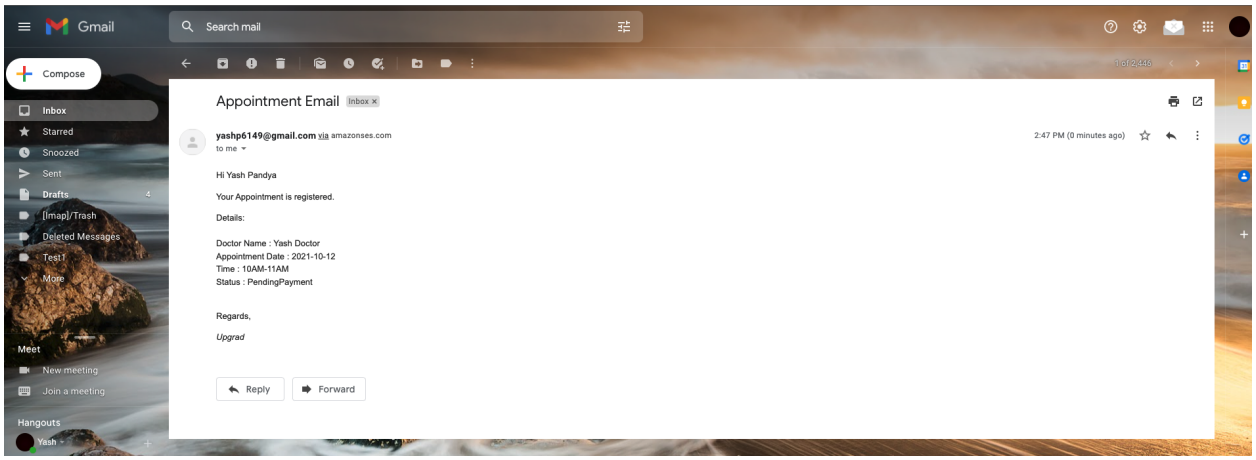

Postman
GET http://3.93.227.67:8081/doctors?status=Rejected&speciality=GENERAL_PHYSICIAN
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
status
Rejected
speciality
GENERAL_PHYSICIAN
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 48 ms
Size: 828 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": "6165c16c22ca6e447aebb61d",
"firstName": "Reject Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "1231231231",
"emailId": "yashp6149@gmail.com",
"pan": "123YASH321",
"status": "Rejected",
"approvedBy": "Yash Admin",
"approverComments": "The registration is rejected",
"registrationDate": "2021-10-12",
"verificationDate": "2021-10-12",
"averageRating": 0.0
}
]

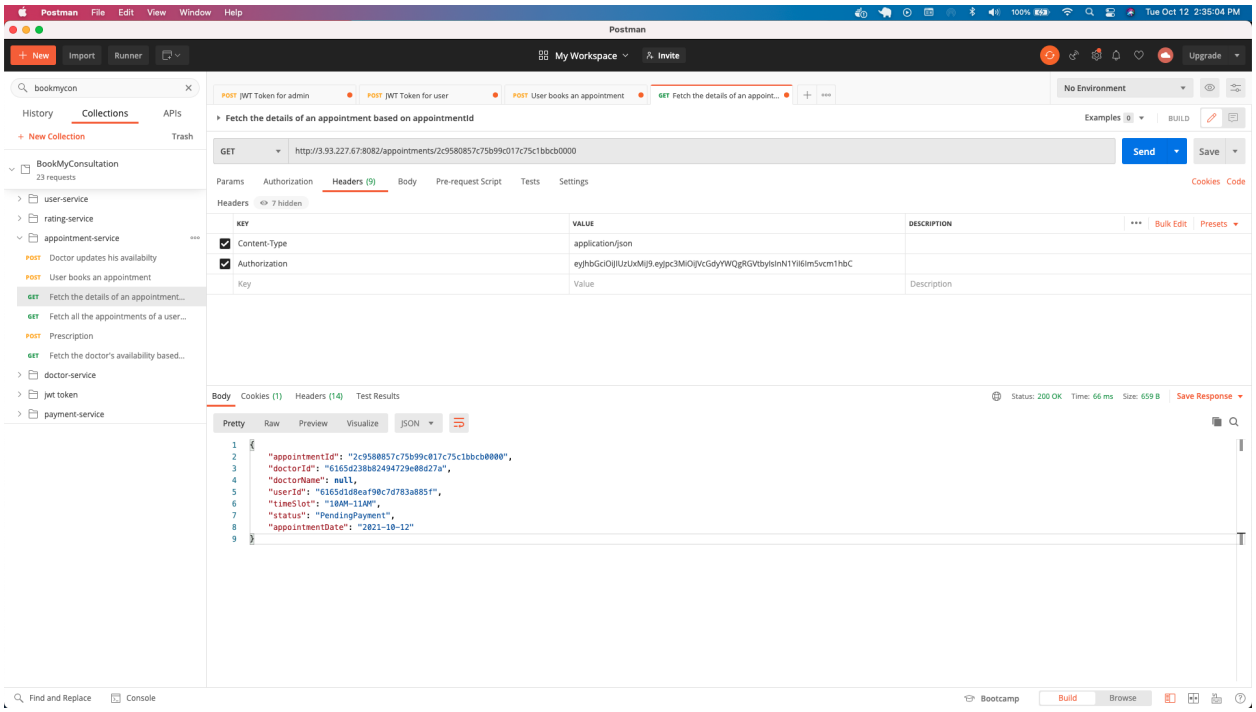

Postman
Fetch the list of doctors based on status and speciality
GET
http://3.93.227.67:8081/doctors?speciality=GENERAL_PHYSICIAN
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
status
speciality
GENERAL_PHYSICIAN
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 48 ms
Size: 1.17 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": "6165c0c122ca6e447aebb61c",
"firstName": "Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "8787878787",
"emailId": "yashp6149@gmail.com",
"pan": "XDALF4215P",
"status": "Active",
"approvedBy": "Yash Admin",
"approverComments": "Doctor is Verified",
"registrationDate": "2021-10-12",
"verificationDate": "2021-10-12",
"averageRating": 0.0
},
{
"id": "6165c16c22ca6e447aebb61d",
"firstName": "Reject Yash Doctor",
"lastName": "Pandya",
"speciality": "GENERAL_PHYSICIAN",
"dob": "1997-03-29",
"mobile": "1231231231",
"emailId": "yashp6149@gmail.com",
"pan": "123YASH321",

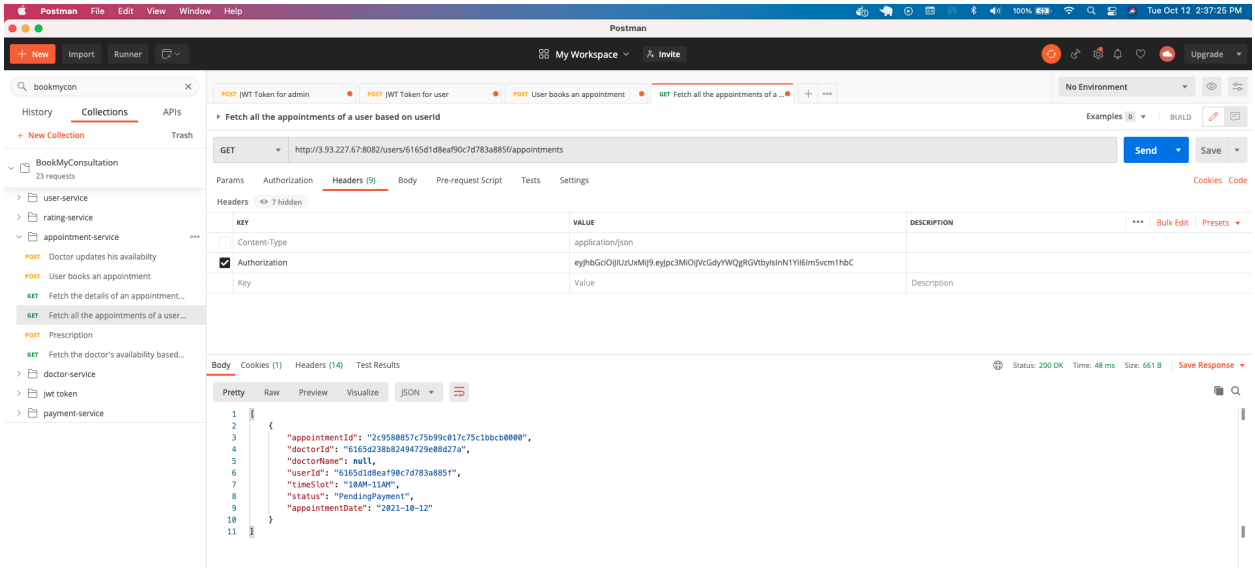

Postman
Doctor updates his availabilty
POST
http://3.93.227.67:8082/doctor/6165c0c122ca6e447aebb61c/availability
Send
Save
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"availabilityMap":{
"2021-10-12":["10AM-11AM"]
}
}
Body
Cookies (1)
Headers (13)
Test Results
Status: 200 OK
Time: 78 ms
Size: 394 B
Save Response
Pretty
Raw
Preview
Visualize
Text

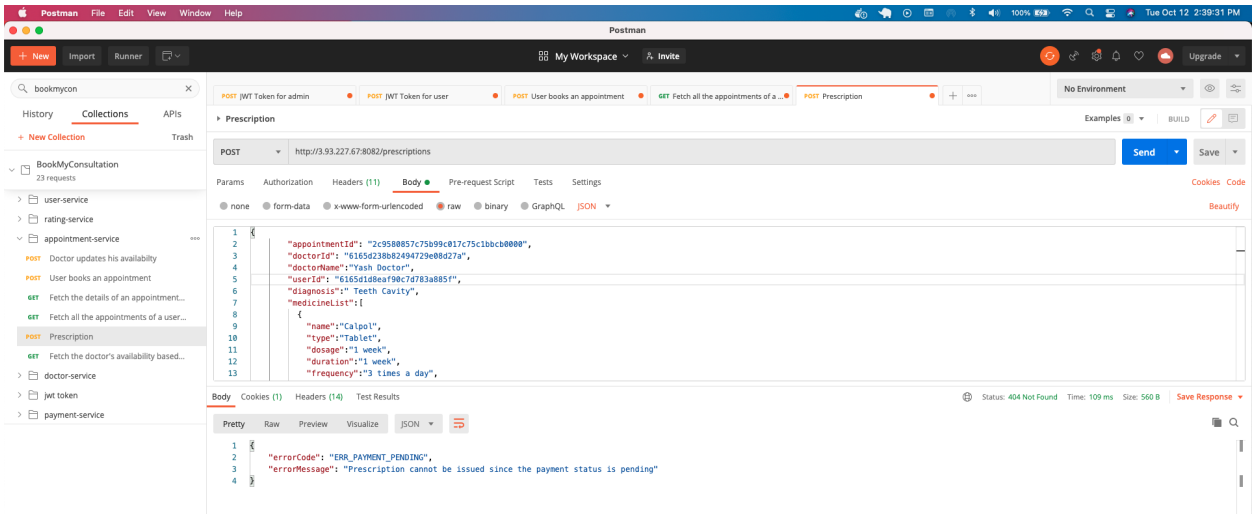

Postman
Fetch the doctor's availability based on doctor ID
GET
http://3.93.227.67:8082/doctor/6165c0c122ca6e447aebb61c/availability
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Headers
7 hidden
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
Content-Type
application/json
Authorization
eyJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJVcGdyYWQgRGVtbyIsInN1YiI6Im5vcm1hbC
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 43 ms
Size: 521 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"doctorId": "6165c0c122ca6e447aebb61c",
"availabilityMap": {
"2021-10-12": [
"10AM-11AM"
]
}
}

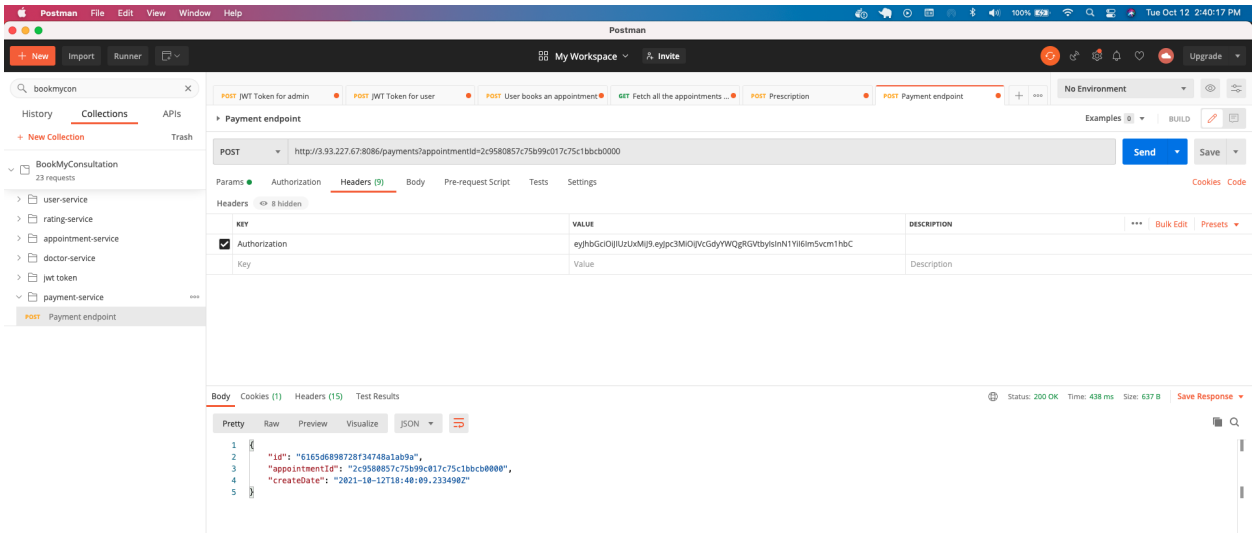

Postman
POST JWT Token for admin
POST JWT Token for user
POST User books an appointment
User books an appointment
POST http://3.93.227.67:8082/appointments
Send
Save
Body raw
{
"doctorId":"6165d238b82494729e08d27a",
"doctorName":"Yash Doctor",
"userId":"6165d1d8eaf90c7d783a885f",
"timeSlot":"10AM-11AM",
"appointmentDate":"2021-10-12"
}
Status: 200 OK Time: 458 ms Size: 534 B
2c9580857c75b99c017c75c1bbcb0000

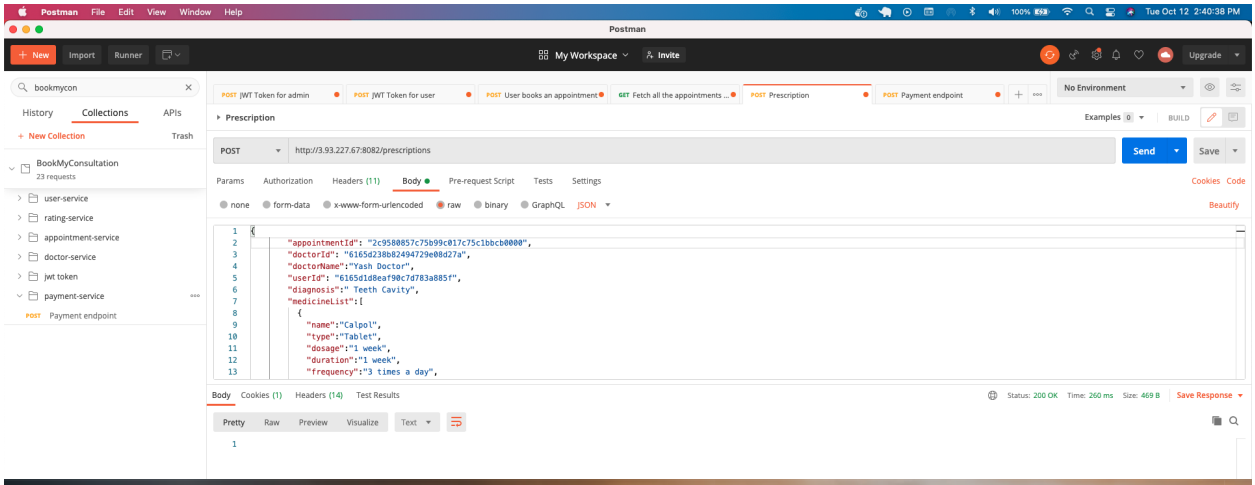

Gmail
Search mail
Compose
Inbox
Starred
Snoozed
Sent
Drafts
[Imap]/Trash
Deleted Messages
Test1
More
Meet
New meeting
Join a meeting
Hangouts
Appointment Email
yashp6149@gmail.com via amazonses.com
to me
2:47 PM (0 minutes ago)
Hi Yash Pandya
Your Appointment is registered.
Details:
Doctor Name : Yash Doctor
Appointment Date : 2021-10-12
Time : 10AM-11AM
Status : PendingPayment
Regards,
Upgrad
Reply
Forward

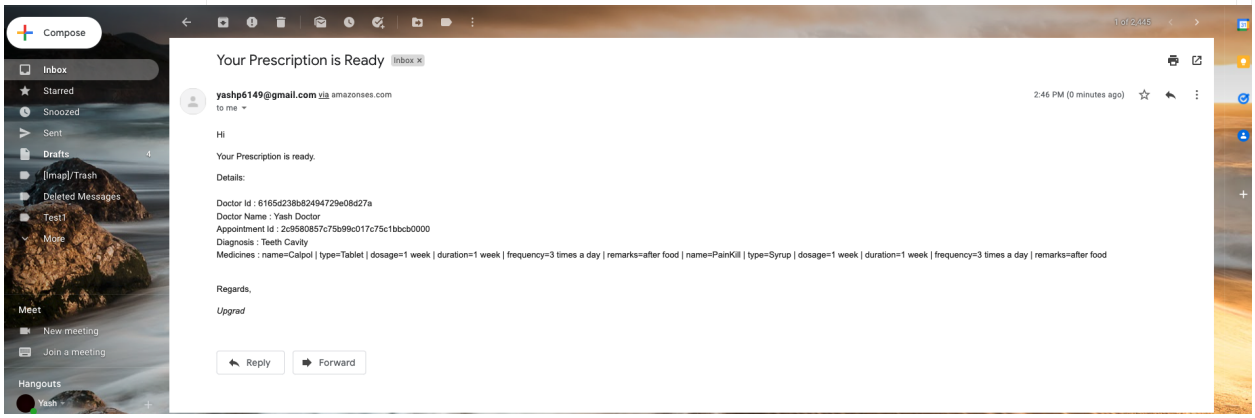

Postman
GET Fetch the details of an appoint...
Fetch the details of an appointment based on appointmentId
GET http://3.93.227.67:8082/appointments/2c9580857c75b99c017c75c1bbcb0000
Send
Save
Headers (9)
Content-Type application/json
Authorization eyJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJiVcGdyYWQgRGVtbyIsInN1YiI6Im5vcm1hbC
Body Cookies (1) Headers (14) Test Results
Status: 200 OK Time: 66 ms Size: 659 B
{
"appointmentId": "2c9580857c75b99c017c75c1bbcb0000",
"doctorId": "6165d238b82494729e08d27a",
"doctorName": null,
"userId": "6165d1d8eaf90c7d783a885f",
"timeSlot": "10AM-11AM",
"status": "PendingPayment",
"appointmentDate": "2021-10-12"
}

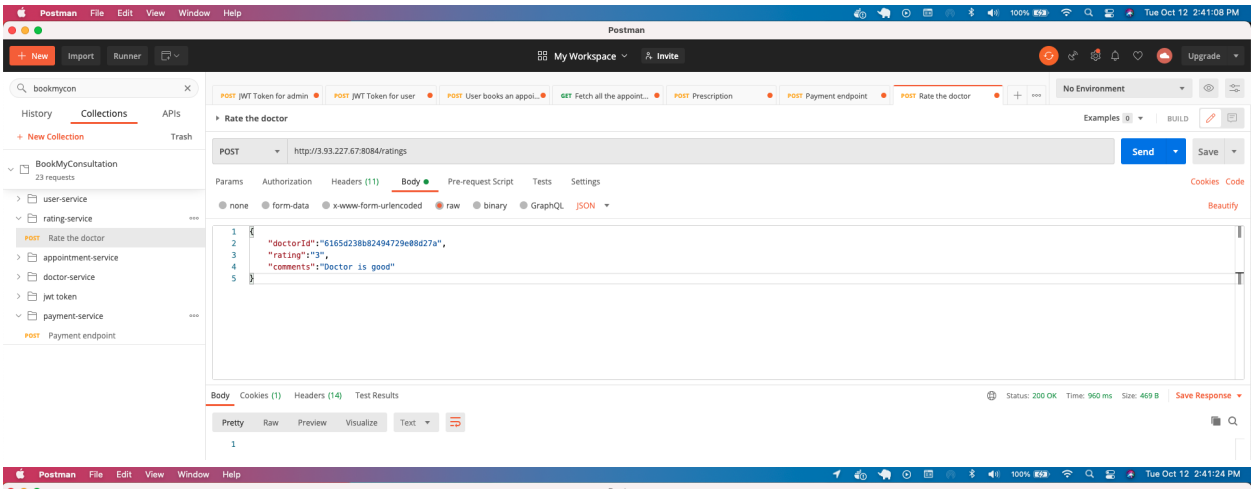

Postman
GET Fetch all the appointments of a user based on userId
http://3.93.227.67:8082/users/6165d1d8eaf90c7d783a885f/appointments
Send
Save
Authorization
eyJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJVcGdyYWQgRGVtbyIsInN1YiI6ImSvcm1hbC
Status: 200 OK Time: 48 ms Size: 661 B
{
"appointmentId": "2c9580857c75b99c017c75c1bbcb0000",
"doctorId": "6165d238b82494729e08d27a",
"doctorName": null,
"userId": "6165d1d8eaf90c7d783a885f",
"timeSlot": "10AM-11AM",
"status": "PendingPayment",
"appointmentDate": "2021-10-12"
}

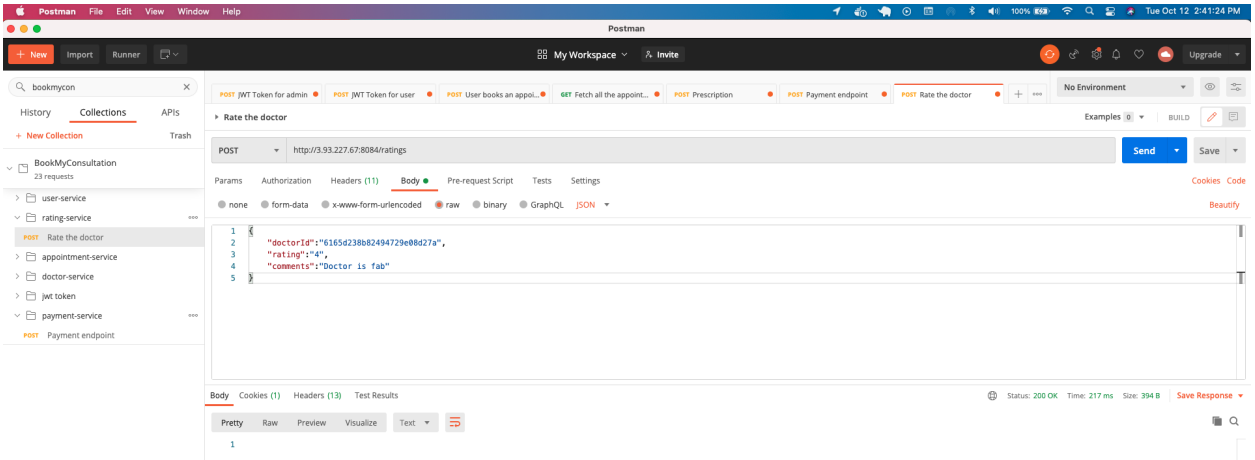

Postman
Prescription
POST
http://3.93.227.67:8082/prescriptions
Send
Save
Body
raw
JSON
{
"appointmentId": "2c9580857c75b99c017c75c1bbcb0000",
"doctorId": "6165d238b82494729e08d27a",
"doctorName":"Yash Doctor",
"userId": "6165d1d8eaf90c7d783a885f",
"diagnosis":" Teeth Cavity",
"medicineList":[
{
"name":"Calpol",
"type":"Tablet",
"dosage":"1 week",
"duration":"1 week",
"frequency":"3 times a day",
Status: 404 Not Found Time: 109 ms Size: 560 B
{
"errorCode": "ERR_PAYMENT_PENDING",
"errorMessage": "Prescription cannot be issued since the payment status is pending"
}

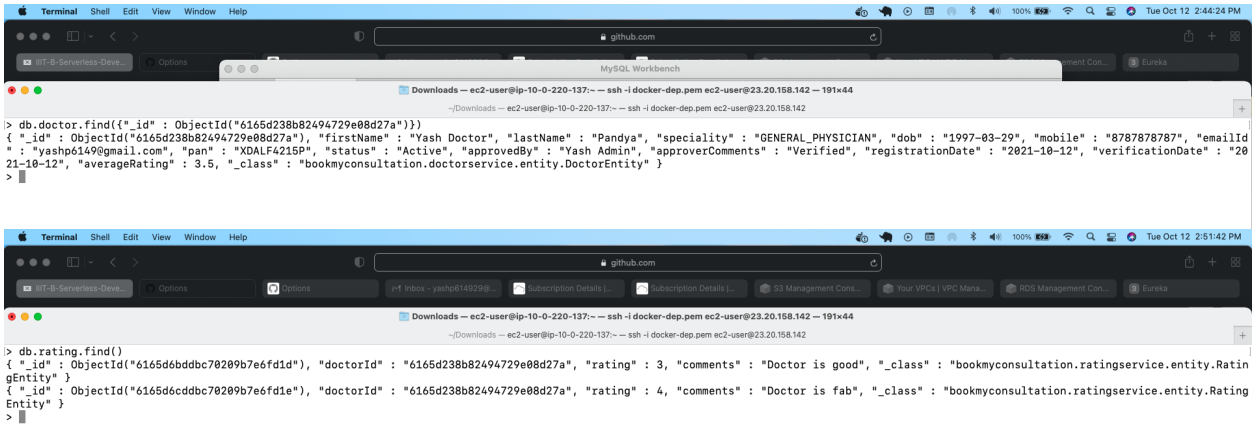

Postman
Payment endpoint
POST
http://3.93.227.67:8086/payments?appointmentId=2c9580857c75b99c017c75c1bbcb0000
Send
Save
Authorization
eyJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJVcGdyYWQgRGVtbyIsInN1YiI6ImSvcm1hbC
Status: 200 OK Time: 438 ms Size: 637 B
{
"id": "6165d6898728f34748a1ab9a",
"appointmentId": "2c9580857c75b99c017c75c1bbcb0000",
"createDate": "2021-10-12T18:40:09.233490Z"
}


Postman
POST http://3.93.227.67:8082/prescriptions
Send
Save
{
"appointmentId": "2c9580857c75b99c017c75c1bbcb0000",
"doctorId": "6165d238b82494729e08d27a",
"doctorName":"Yash Doctor",
"userId": "6165d1d8eaf90c7d783a885f",
"diagnosis":" Teeth Cavity",
"medicineList":[
{
"name":"Calpol",
"type":"Tablet",
"dosage":"1 week",
"duration":"1 week",
"frequency":"3 times a day",


Your Prescription is Ready
yashp6149@gmail.com via amazonses.com
Hi
Your Prescription is ready.
Details:
Doctor Id : 6165d238b82494729e08d27a
Doctor Name : Yash Doctor
Appointment Id : 2c9580857c75b99c017c75c1bbcb0000
Diagnosis : Teeth Cavity
Medicines : 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
Regards,
Upgrad
Reply
Forward


Postman
POST http://3.93.227.67:8084/ratings
Send
Save
{
"doctorId":"6165d238b82494729e08d27a",
"rating":"3",
"comments":"Doctor is good"
}


Postman
POST http://3.93.227.67:8084/ratings
Send
Save
{
"doctorId":"6165d238b82494729e08d27a",
"rating":"4",
"comments":"Doctor is fab"
}



- Ratings are updated by doctor service via rating service using kafka to send doctor Id and rating in the topic and doctor services fetches it and updates.
- Similarly, once a doctor registration is approved or rejected, doctor service sends notification to notification service using kafka and notification service sends the email.
- When payment is being made, payment service sends a message to appointment service via kafka topic and appointment services changes status to Confirmed from PendingPayment and once Payment is completed and appointment status is Confirmed then only Prescription can be registered and sent via email.