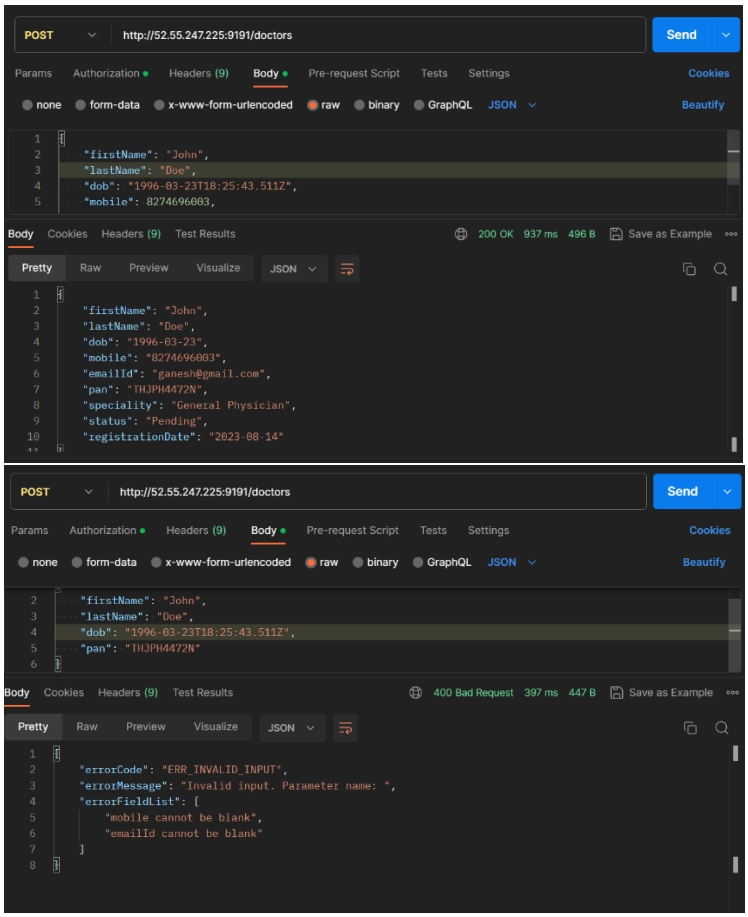
Code Logic

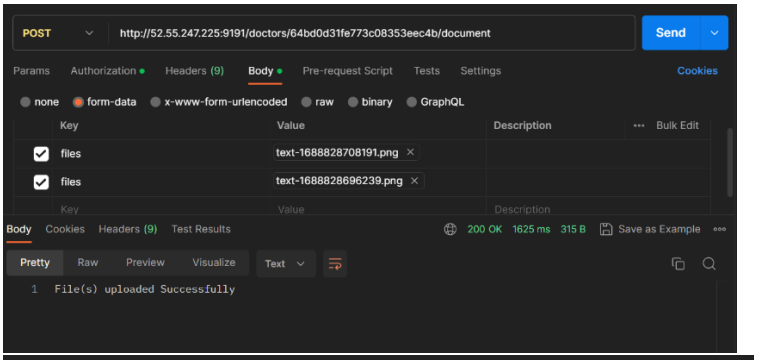
Following microservices are created for implementation of BookMyConsultation:

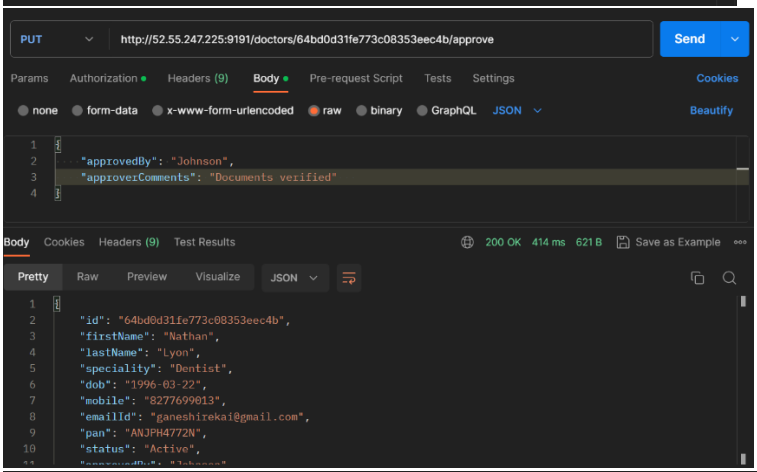
1. Doctor Service

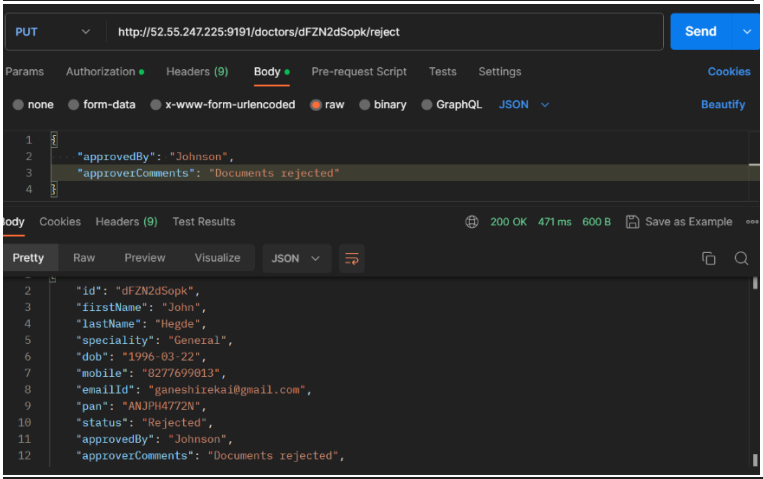
   Spring boot project is created including dependencies like Web, security, MongoDB, Eureka client etc. MongoRepository is created for connection with mongo DB and operations. This service is registered as a eureka client. Security is implemented using Roles/Users based authentication for different endpoints. Endpoints are implemented using RestController concepts. Exception handling is implemented using controller advice concepts. Similarly, Kafka is used for sending notifications to users for different operations. This service takes care of saving new doctor details, uploading documents of a doctor, approving and rejecting doctor details, listing of doctors etc.

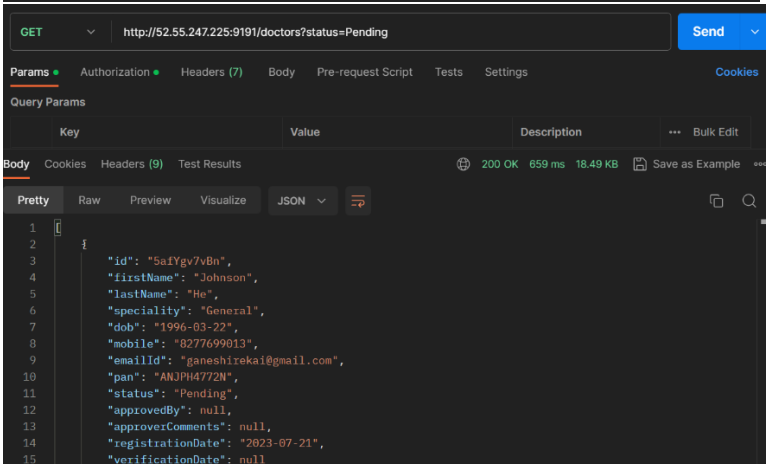
   Below are the screenshots of endpoints:

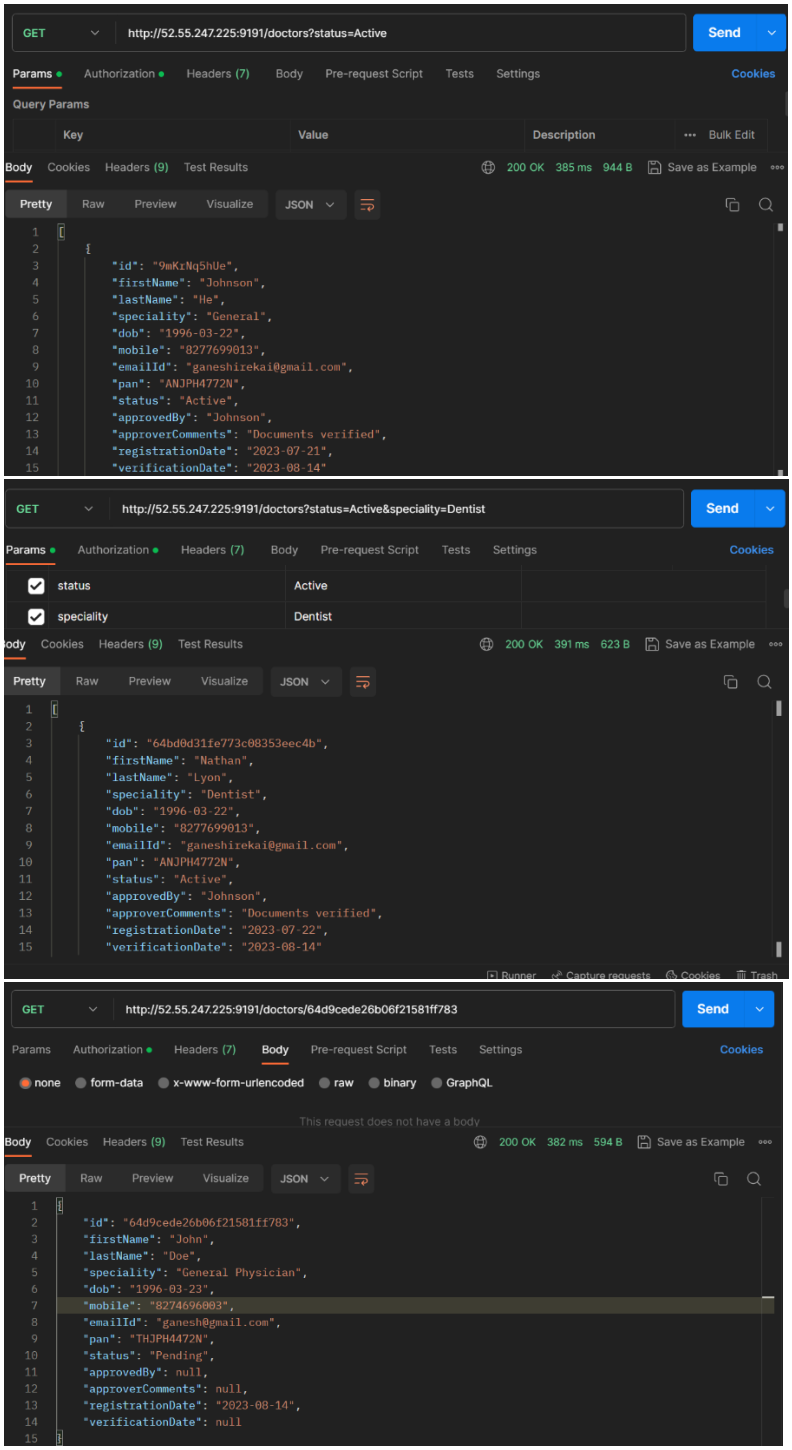



POST
http://52.55.247.225:9191/doctors/64bd0d31fe773c08353eec4b/document
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Key
Value
Description
Bulk Edit
files
text-1688828708191.png
files
text-1688828696239.png
Body
Cookies
Headers (9)
Test Results
200 OK
1625 ms
315 B
Save as Example
Pretty
Raw
Preview
Visualize
Text
1 File(s) uploaded Successfully


PUT
http://52.55.247.225:9191/doctors/64bd0d31fe773c08353eec4b/approve
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"approvedBy": "Johnson",
"approverComments": "Documents verified"
}
Body
Cookies
Headers (9)
Test Results
200 OK
414 ms
621 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"id": "64bd0d31fe773c08353eec4b",
"firstName": "Nathan",
"lastName": "Lyon",
"speciality": "Dentist",
"dob": "1996-03-22",
"mobile": "8277699013",
"emailId": "ganeshirekai@gmail.com",
"pan": "ANJPH4772N",
"status": "Active",


PUT
http://52.55.247.225:9191/doctors/dFZN2dSopk/reject
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"approvedBy": "Johnson",
"approverComments": "Documents rejected"
}
Cookies
Headers (9)
Test Results
200 OK
471 ms
600 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
"id": "dFZN2dSopk",
"firstName": "John",
"lastName": "Hegde",
"speciality": "General",
"dob": "1996-03-22",
"mobile": "8277699013",
"emailId": "ganeshirekai@gmail.com",
"pan": "ANJPH4772N",
"status": "Rejected",
"approvedBy": "Johnson",
"approverComments": "Documents rejected",


GET
http://52.55.247.225:9191/doctors?status=Pending
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (9)
Test Results
200 OK
659 ms
18.49 KB
Save as Example
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": "5afYgv7vBn",
"firstName": "Johnson",
"lastName": "He",
"speciality": "General",
"dob": "1996-03-22",
"mobile": "8277699013",
"emailId": "ganeshirekai@gmail.com",
"pan": "ANJPH4772N",
"status": "Pending",
"approvedBy": null,
"approverComments": null,
"registrationDate": "2023-07-21",
"verificationDate": null

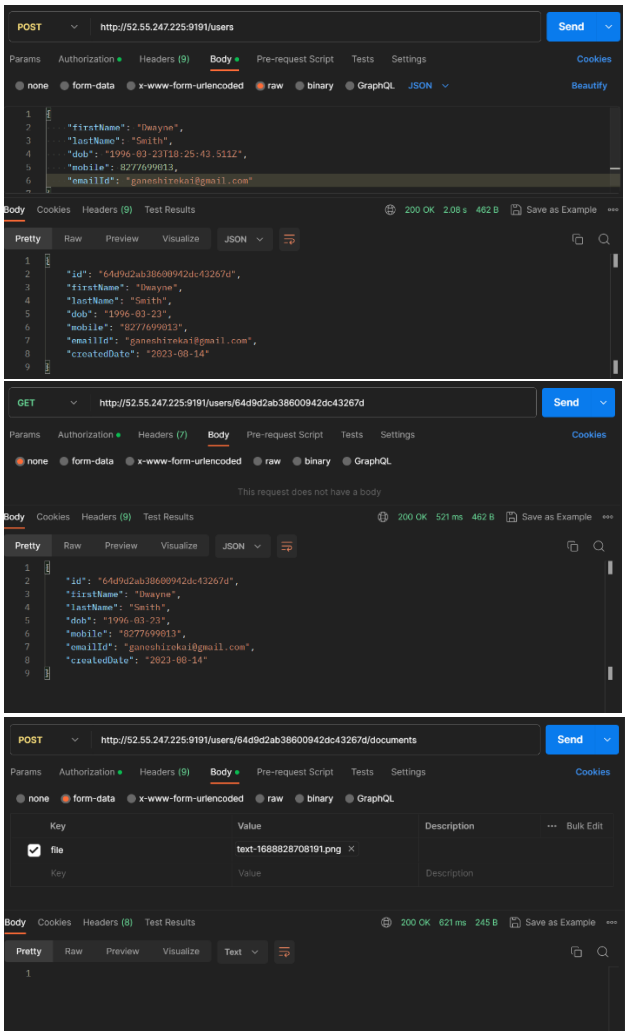


2. <u>User Service</u>

   User service is implemented similar to doctor service including spring boot dependencies like MongoDB, web, security etc.

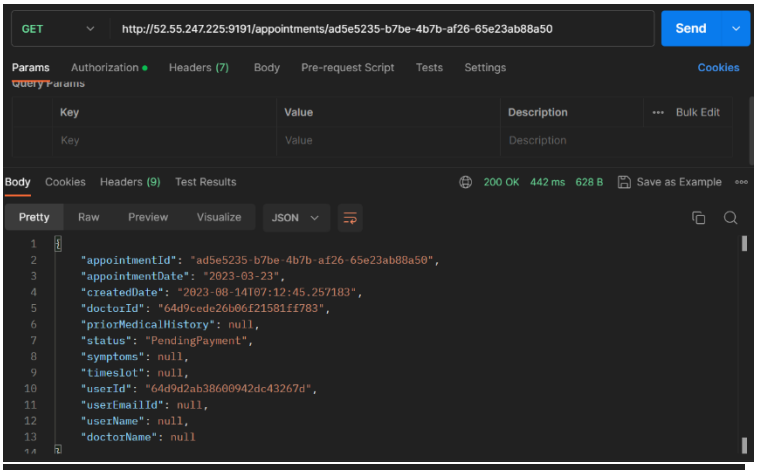
   Security, exception handling, kafka producer, mongo repository, rest controller concepts are used for implementation. This service is registered as a eureka client. This service takes care of saving new user details, uploading documents for a user, getting details of a user etc.

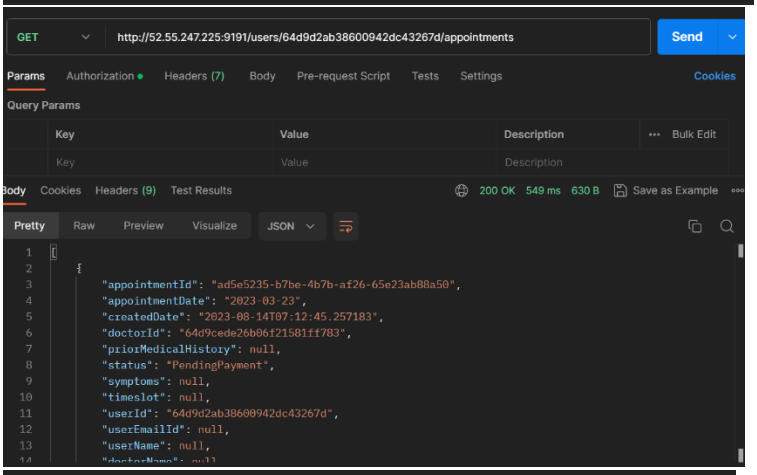
Below are the screenshots of endpoints:

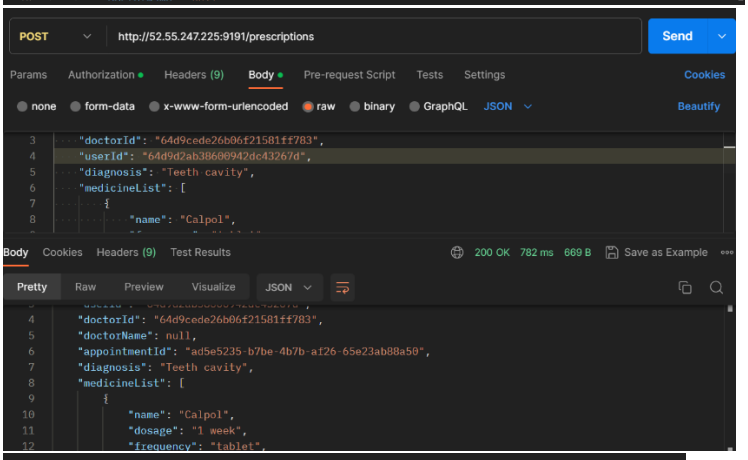


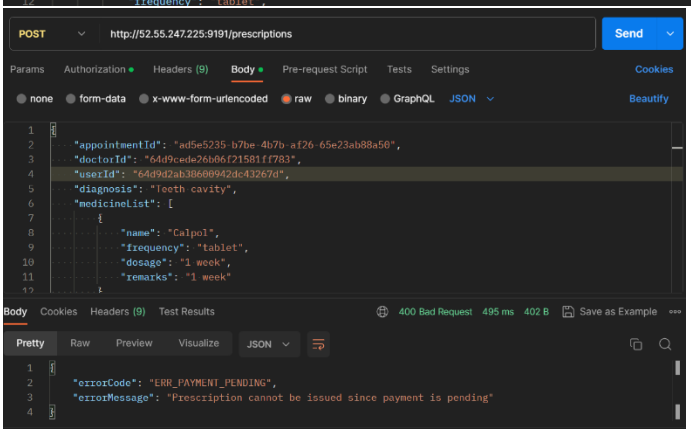
3. <u>Appointment Service</u>

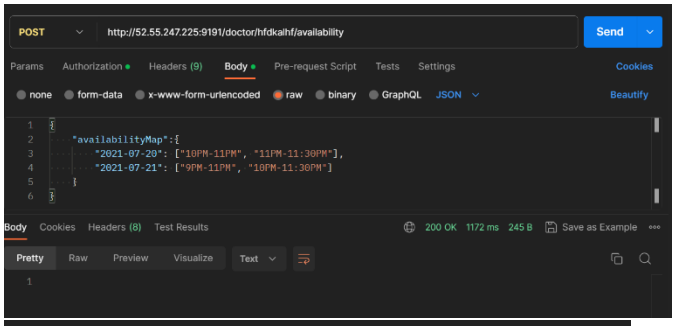
   Appointment service is registered as eureka client like other services. Similar to other services this service is a Spring boot project including dependencies like Web, JPA, mongoDB, Security, Kafka clients, etc. This service takes care of handling appointment related operations like saving doctor availability, booking an appointment, getting details about appointment, and uploading prescription etc. Below are the screenshots of appointment service endpoints:

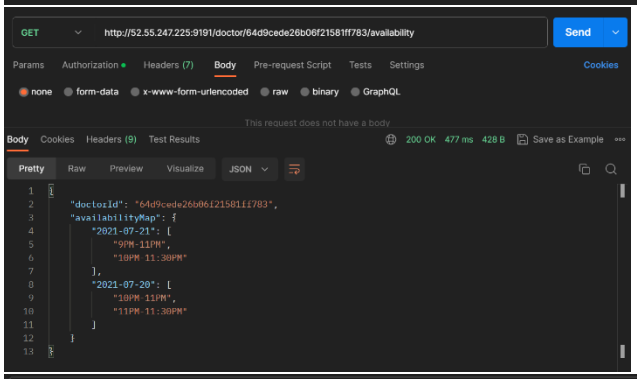

GET
http://52.55.247.225:9191/appointments/ad5e5235-b7be-4b7b-af26-65e23ab88a50
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies
Key Value Description Bulk Edit
Body Cookies Headers (9) Test Results 200 OK 442 ms 628 B Save as Example
Pretty Raw Preview Visualize JSON
{
"appointmentId": "ad5e5235-b7be-4b7b-af26-65e23ab88a50",
"appointmentDate": "2023-03-23",
"createdDate": "2023-08-14T07:12:45.257183",
"doctorId": "64d9cede26b06f21581ff783",
"priorMedicalHistory": null,
"status": "PendingPayment",
"symptoms": null,
"timeslot": null,
"userId": "64d9d2ab38600942dc43267d",
"userEmailId": null,
"userName": null,
"doctorName": null
}

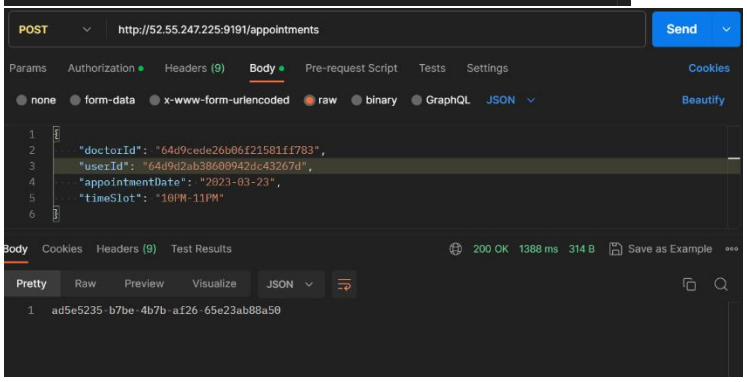

GET
http://52.55.247.225:9191/users/64d9d2ab38600942dc43267d/appointments
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies
Query Params
Key Value Description Bulk Edit
Body Cookies Headers (9) Test Results 200 OK 549 ms 630 B Save as Example
Pretty Raw Preview Visualize JSON
[
{
"appointmentId": "ad5e5235-b7be-4b7b-af26-65e23ab88a50",
"appointmentDate": "2023-03-23",
"createdDate": "2023-08-14T07:12:45.257183",
"doctorId": "64d9cede26b06f21581ff783",
"priorMedicalHistory": null,
"status": "PendingPayment",
"symptoms": null,
"timeslot": null,
"userId": "64d9d2ab38600942dc43267d",
"userEmailId": null,
"userName": null,


POST
http://52.55.247.225:9191/prescriptions
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
"doctorId": "64d9cede26b06f21581ff783",
"userId": "64d9d2ab38600942dc43267d",
"diagnosis": "Teeth cavity",
"medicineList": [
{
"name": "Calpol",
Body Cookies Headers (9) Test Results 200 OK 782 ms 669 B Save as Example
Pretty Raw Preview Visualize JSON
"doctorId": "64d9cede26b06f21581ff783",
"doctorName": null,
"appointmentId": "ad5e5235-b7be-4b7b-af26-65e23ab88a50",
"diagnosis": "Teeth cavity",
"medicineList": [
{
"name": "Calpol",
"dosage": "1 week",
"frequency": "tablet",


POST
http://52.55.247.225:9191/prescriptions
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
{
"appointmentId": "ad5e5235-b7be-4b7b-af26-65e23ab88a50",
"doctorId": "64d9cede26b06f21581ff783",
"userId": "64d9d2ab38600942dc43267d",
"diagnosis": "Teeth cavity",
"medicineList": [
{
"name": "Calpol",
"frequency": "tablet",
"dosage": "1 week",
"remarks": "1 week"
Body Cookies Headers (9) Test Results 400 Bad Request 495 ms 402 B Save as Example
Pretty Raw Preview Visualize JSON
{
"errorCode": "ERR_PAYMENT_PENDING",
"errorMessage": "Prescription cannot be issued since payment is pending"
}



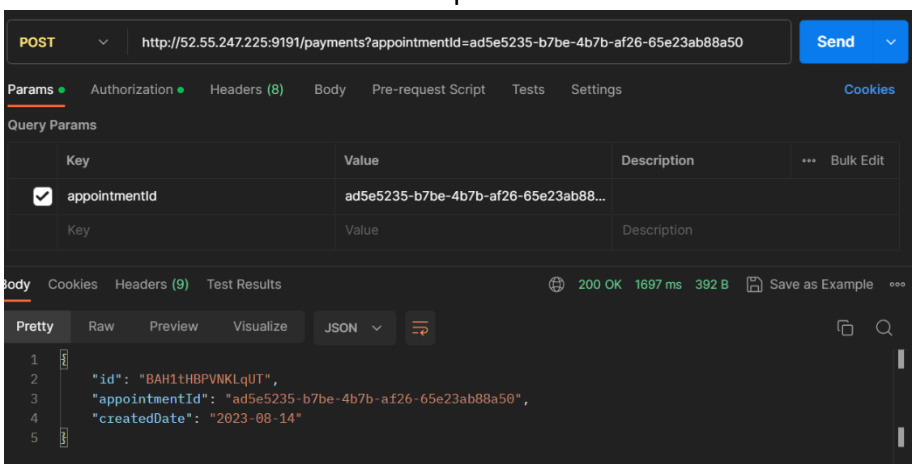




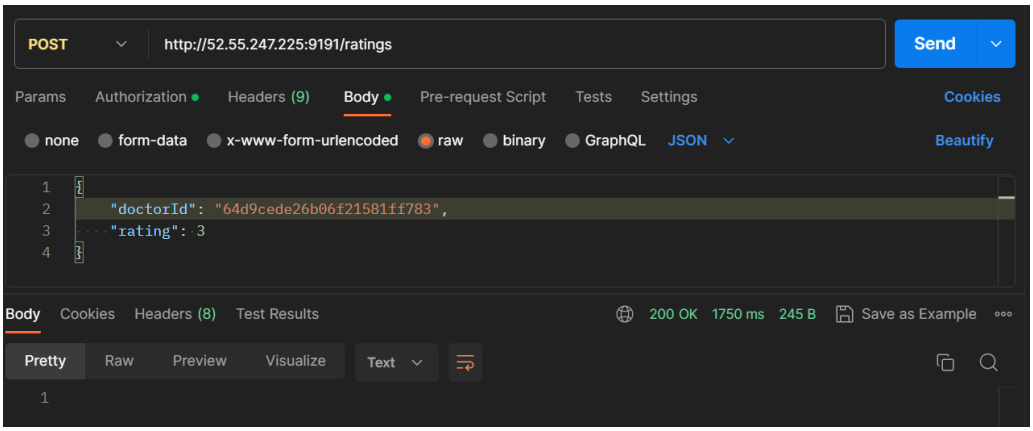
4. <u>Payment Service</u>

   Payment service takes care of making a dummy payment for a particular appointment id. It has an endpoint which generates a payment id on for a valid appointment id. This is implemented using Spring boot project with dependencies like Web, Security, JPA etc

   Below is the screenshot of the endpoint:



5. Rating Service
   This service takes care of storing rating for a particular doctor when doctor ID is passed to it. Similar to other services, this is a spring boot project containing dependencies like Spring security, Web, MongoDB etc.
   Below is the screenshot of the endpoint:

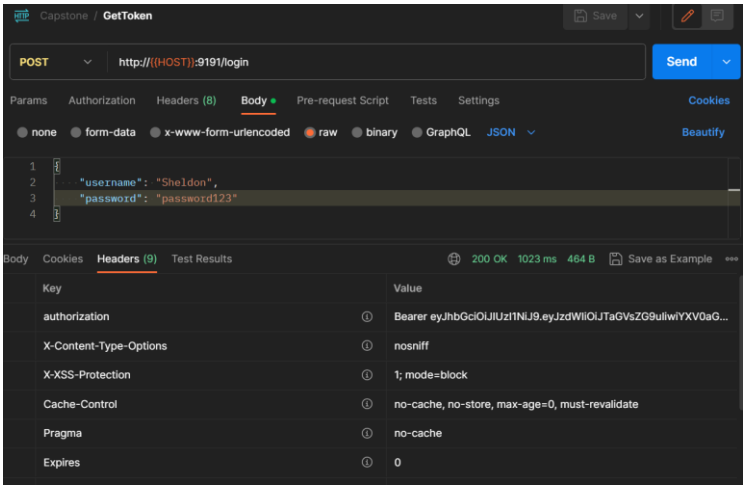


6. Eureka Server
   Eureka server is implemented for service discovery using Spring Boot Eureka server concepts. This server gets itself registered at port number 8761 and waits for other services to be registered with it.

7. BMC Gateway
   This service acts like an API gateway for BMC application which listens on port number 9191 for different API requests and routes them to appropriate microservices. This is implemented using Spring Cloud Starter Gateway

8. Notification Service
   Notification service acts like a consumer of Kafka messages produced by different microservices and sends mail email IDs from the kafka messages. It listens to doctor-service, user-service, appointment-service topics. This is implemented using Kafka Consumer concepts and Email service is implemented using AWS SQS concepts.

9. JWT Generator
   JWT generator implements token generation of JWT of different users with different roles.
   Below is the screenshot of the endpoint to get JWT Token:



Deployment of BMC:

Dockerfile is created for each of the above microservices. OpenJDK 17 is taken as base image for each of the microservices and corresponding binaries are added after packaging. Docker compose file is created in the BookMyConsultation folder which is responsible for building docker images, creating the containers and running them. Config.env file is created for all the dynamic values like AWS instance credentials, IP addresses, URLs etc.

Instructions to run the application:

Prerequisites:

- Docker is installed and is running.
- AWS account, Access key and secret key for AWS account
- An AWS RDS instance
- An EC2 instance running MongoDB
- An EC2 instance running Kafka server

Steps:

- Navigate to BookMyConsultation directory.
- Modify the config.env file with appropriate values.
- Run the following command:
  sudo docker compose up -d
- BMC application is ready to take API requests!
- BMC is configured with below user credentials and roles to get started:
  Username – Leonard, password – password, Role - USER

Username – Sheldon, password – password123, Role – USER
Username – Rajesh, password – password12, Role – ADMIN

- BMC API gateway is configured to run on port number 9191, users can use this port for all the endpoints.
- To get the JWT token for above users, send an API request to /login endpoint with username and password fields in the request body