Hospital Management System (Mini - Project)

By – Darsh Jotangia

Hospital-management-system

The Project uses:

1. Springboot
2. Maven
3. MongoDB
4. Swagger
5. Junit
6. Docker
7. GitHub
8. Postman

Dependencies

Dependencies used are:

1. spring-boot-starter-web
2. springdoc-openapi-ui
3. Junit
4. spring-boot-starter-test
5. spring-boot-starter-security
6. spring-security-test
7. spring-boot-starter-data-mongodb
8. commons-io

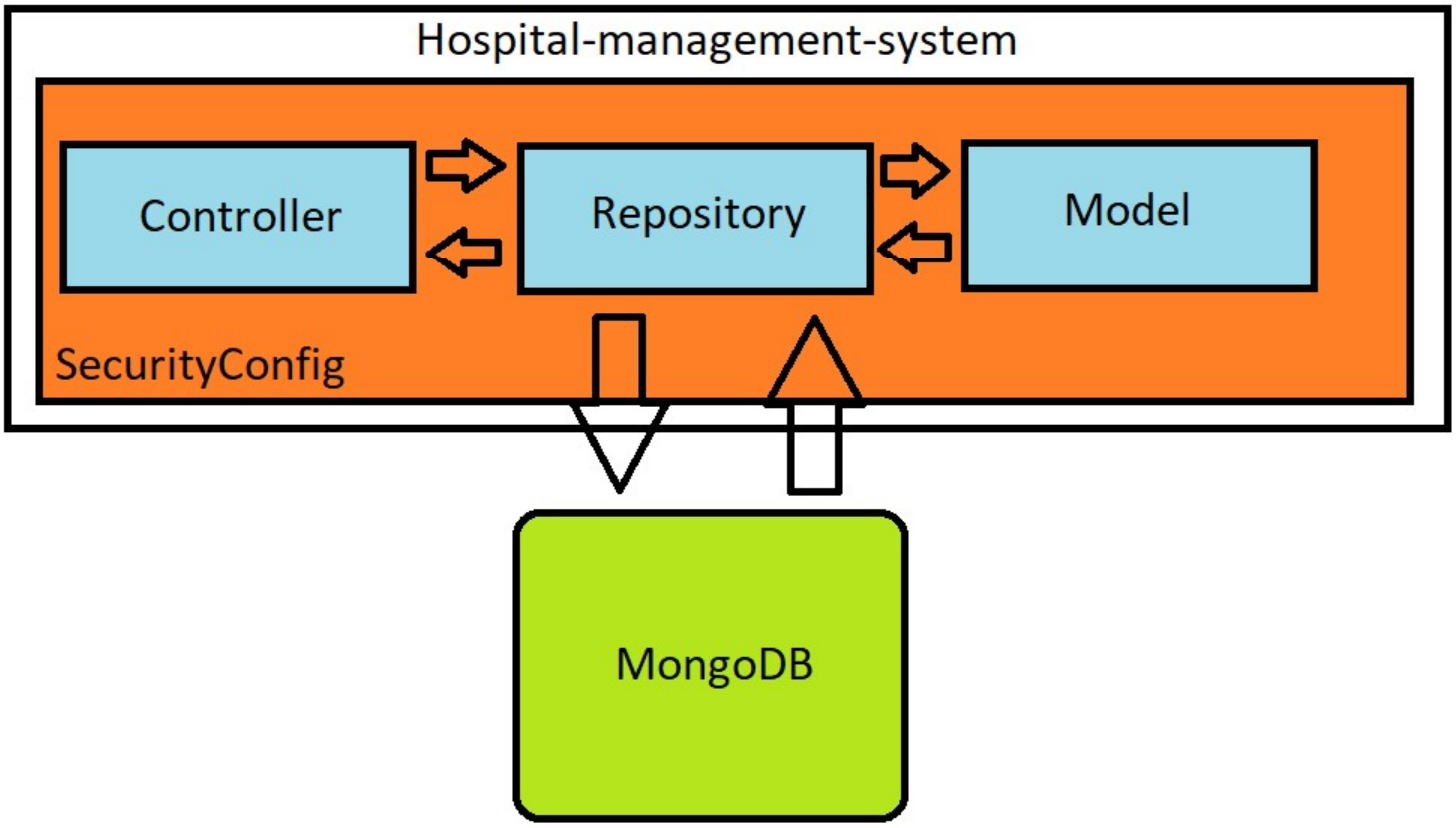
Architecture

The microservice has 3 controllers , DoctorController, PatientController and PrescriptionController.

These controllers communicate with 2 repositories namely AppointmentRepository and PrescriptionRepository.

The repositories further use Models - Appointment and Prescription to store data in MongoDB Database.

Architecture



Data Model / Classes used

Two classes were used , which are:

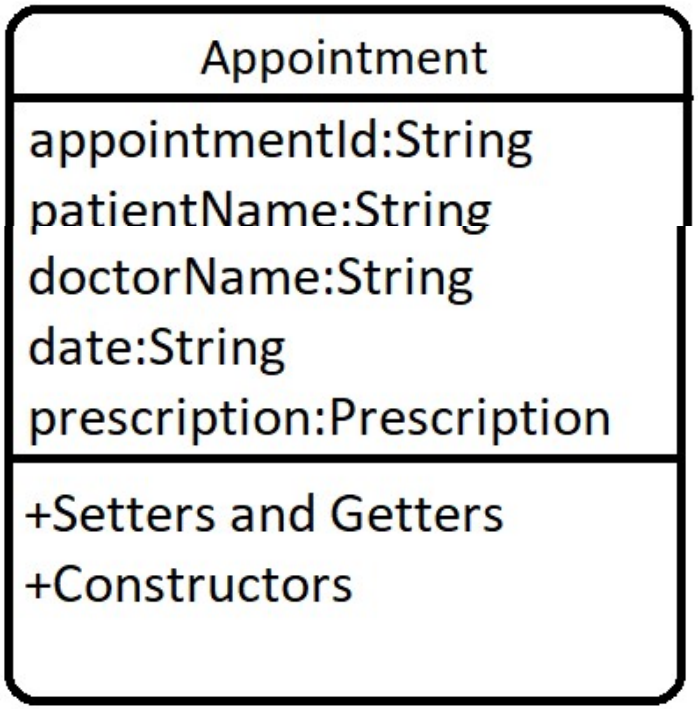
Appointment

Prescription

Appointment - Class

Attributes:

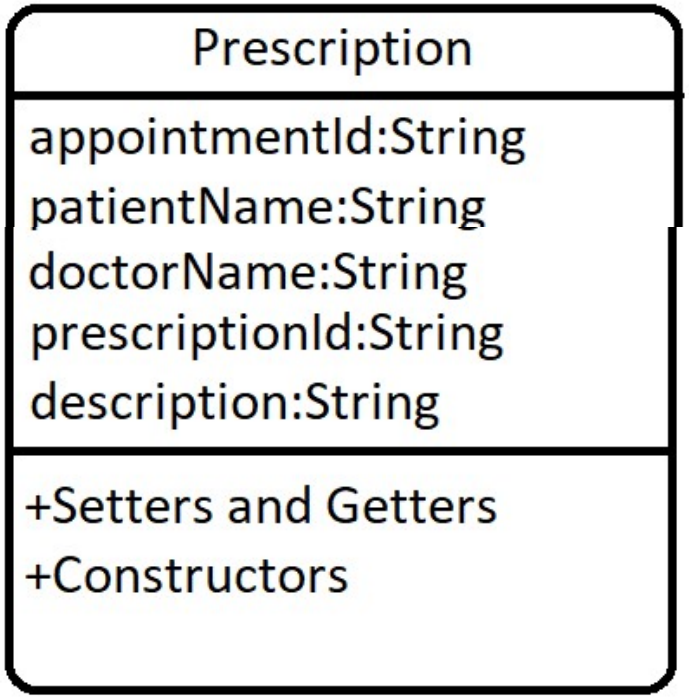
appointmentId
patientName
doctorName
date
prescription



Prescription - Class

Attributes:

prescriptionId
appointmentId
description
patientName
doctorName



Repository

This projects uses 2 repositories , AppointmentRepository and PrescriptionRepository.

Each repository extends MongoRepository which provides all the necessary methods which help to create CRUD application.

Controllers

In total 3 controllers were used, namely:

- DoctorController
- PatientController
- PrescriptionController

Each controller has role-based access restriction, based on the SecurityConfig.java file.

DoctorController can be accessed by doctors only, similarly PatientController can be accessed by patients only and PrescriptionController can be accessed by users of both roles.

DoctorController

DoctorController is mapped with "/doctor" and has the following end points:

- "/doctorappointment" - which will receive GET requests along with request parameters. For ex-" http://localhost:8081/doctor/doctorappointment?doctorName=doctor1".

- "/save" – which will receive POST requests along with Appointment object in JSON format as body.For ex – "http://localhost:8081/doctor/save" .

PatientController

PatientController is mapped with "/patient" and has the following end points:

- "/myappointment" - which will receive GET requests along with request parameters. For ex-" http://localhost:8081/patient/patientappointment?patientName=patient1".

- "/save" – which will receive POST requests along with Appointment object in JSON format as body.For ex – "http://localhost:8081/patient/save" .

PrescriptionController

PrescriptionController has the following end points:

- "/viewprescription" - which will receive GET requests along with request parameters. For ex-" http://localhost:8081/saveprescription?patientName= patient1".

- "/saveprescription" – which will receive POST requests along with Prescription object in JSON format as body.For ex – "http://localhost:8081/saveprescription" .

Security

Configuration for security was added allowing only authorized users with appropriate roles to have access.

WebSecurityConfigerAdapter is extended and roles to endpoints and users were created.

The Roles are as follows:

- DOCTOR - doctor1 PASSWORD - password
- PATIENT - patient1 PASSWORD - password
- DOCTOR,PATIENT - user1 PASSWORD - password

Testing and Code Coverage

Test Cases were made for each mapping using Mockito and Junit and with code coverage:

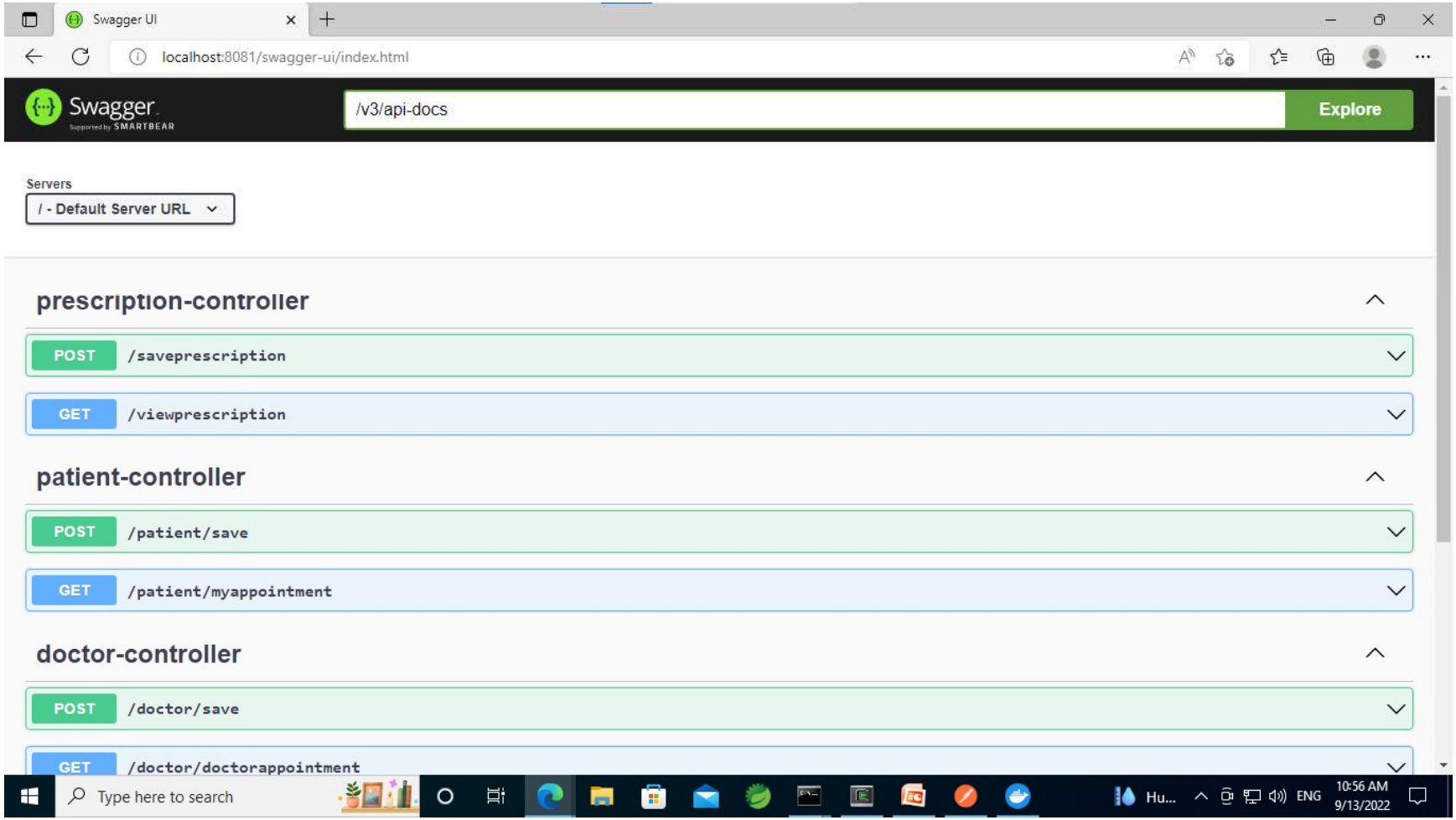
Classes	-	100%
Methods	-	93%
Lines	-	88%

Documentation

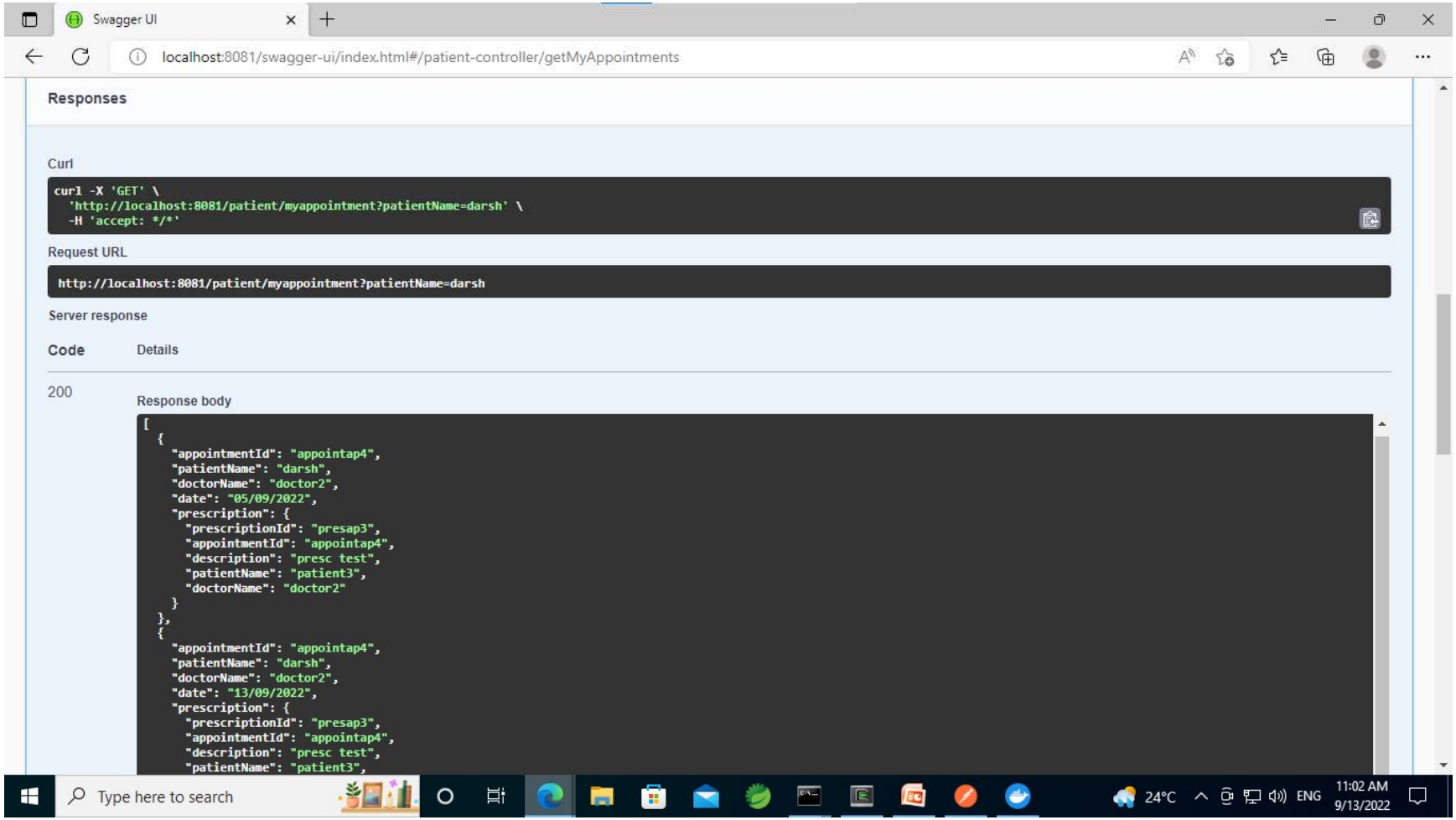
Swagger is used for interactive API documentation of the microservice and its architecture.

It involves description of the classes and mappings used in the controllers along with execution and response monitoring of the mappings.

Swagger Screenshot



Swagger Screenshot



Dockerization

Dockerization is the process of packing, deploying and running applications using Docker containers. Docker is a tool that ships the application with all the necessary functionalities as one package.

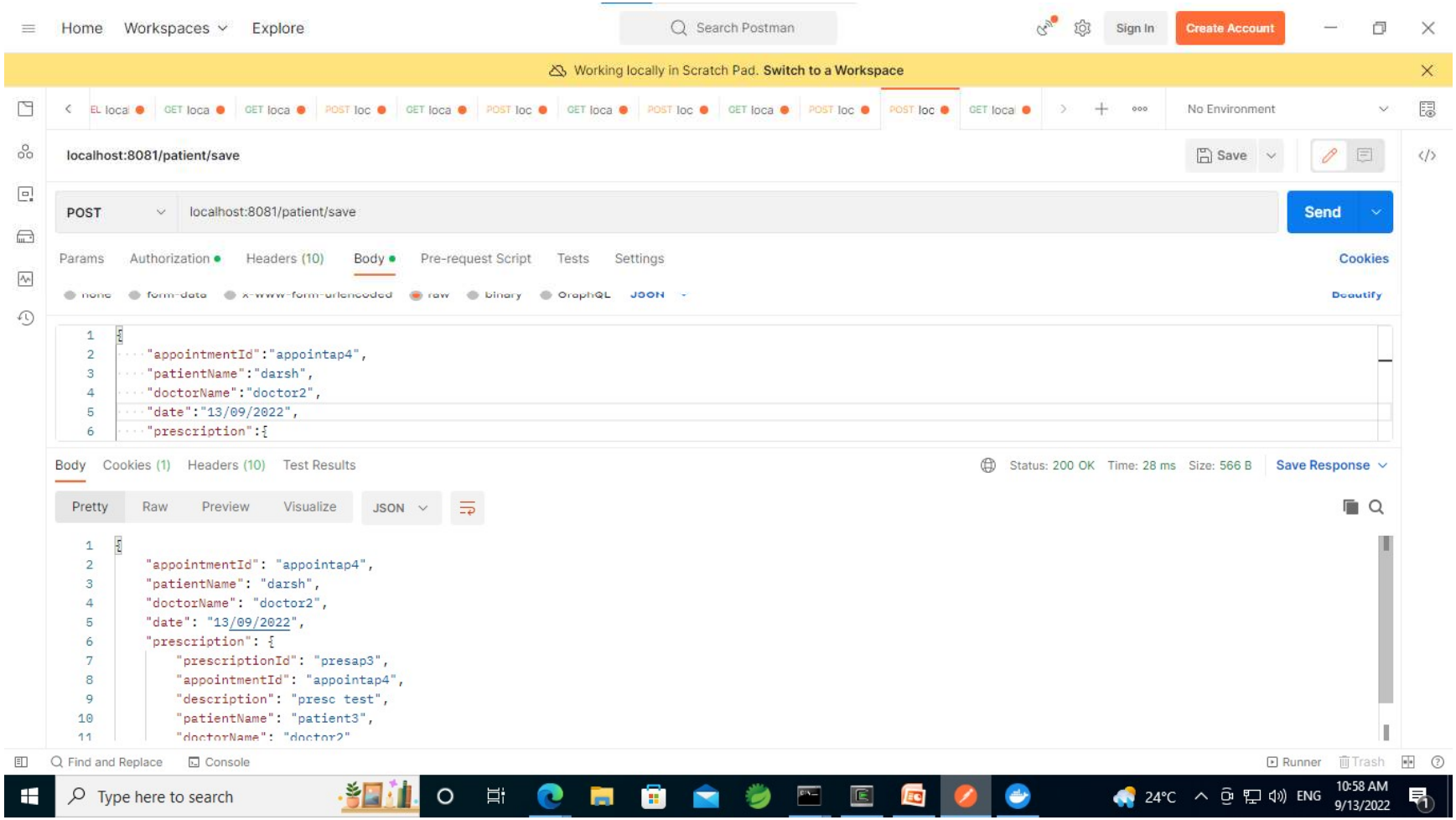
This is acheived by making a docker image of our springboot appplication and then by the help of docker compose, we run our containers(i.e. Application container and MongoDB container) on the same network.

Postman

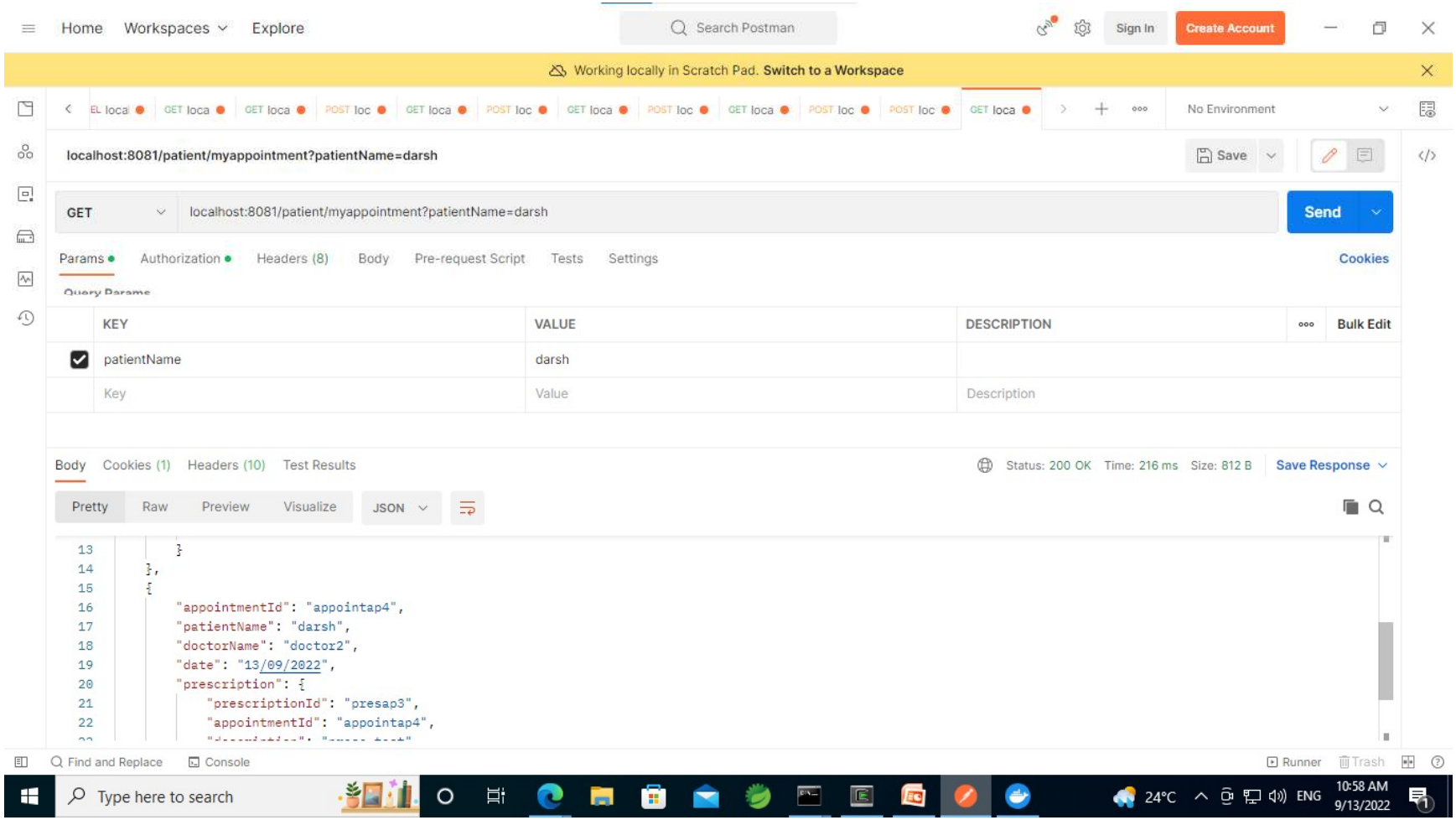
Postman makes it easier to create, share, test and document APIs. With this we can create and save simple and complex HTTP/s requests and their responses.

We have also used the curl commands generated from Postman to test the functionality of the project through command line tools such as Cygwin and Command Prompt.

Postman Screenshot - POST



Postman Screenshot - GET



Thank You