HOSPITAL MANAGEMENT SYSTEM

Submitted by -
Mopidevi Deepthi

INTRODUCTION

- Hospital Management System brings together all the information and processes of a hospital, in a single platform.

- Hospital management system is the inevitable part of the lifecycle of the modern medical institution. It automates numerous daily operations and enables smooth interactions of the users.

- The system automatically generates a highly-efficient process and makes it quick.

PROJECT BRIEFING

- The application allows

  - Patient to book their appointments and can view.

  - Doctors look into the appointments they had appointed.

- Doctors posts and gets the prescription.

- Patients gets the prescription without any delay.

- One of the main requirement here is Secuity.

MICROSERVICE

- Single microservice
- Consists of two schemas

  - Appointment

  - Prescription
- Patient can book an appointment by passing required details.

  Ex : localhost:8083/patient/bookappointment
- Based on patient situation, a prescription can post by doctor.

  Ex : localhost:8083/prescription/save

- Doctor and patient can view their respective appointments.

  Ex : localhost:8083/doctor/appointments

  Ex : localhost:8083/patient/myappointments

- Finally, patient receives his prescription.

  Ex : localhost:8083/prescription/view

TOOLS USED

- Apache Maven
- Java
- Docker
- Spring Boot
- Mongo DB

APACHE MAVEN

- Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

- Maven uses Convention over Configuration, which means developers are not required to create build process themselves.

- Developers do not have to mention each and every configuration detail. Maven provides sensible default behavior for projects.

DOCKER

- *Docker* is an open source platform that enables developers to build, deploy, run, update and manage *containers*—standardized, executable components that combine application source code with the operating syste libraries and dependencies required to run that code in any environment.

- *Docker container* starts with a simple text file containing instructions for how to build the Docker container image.

- *Docker images* contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container.

- Some docker commands –

```
mvn install dockerfile:build
docker ps
docker images
docker-compose –f docker-compose-mongo.yml up –d
```

- Docker image –

Ex: deepthi007/hospital-management-system:0.0.1-SNAPSHOT

JAVA

- Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.

- A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support Java.

SPRING BOOT

- Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

- It is a Spring module that provides the *Rapid Application Development* feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.

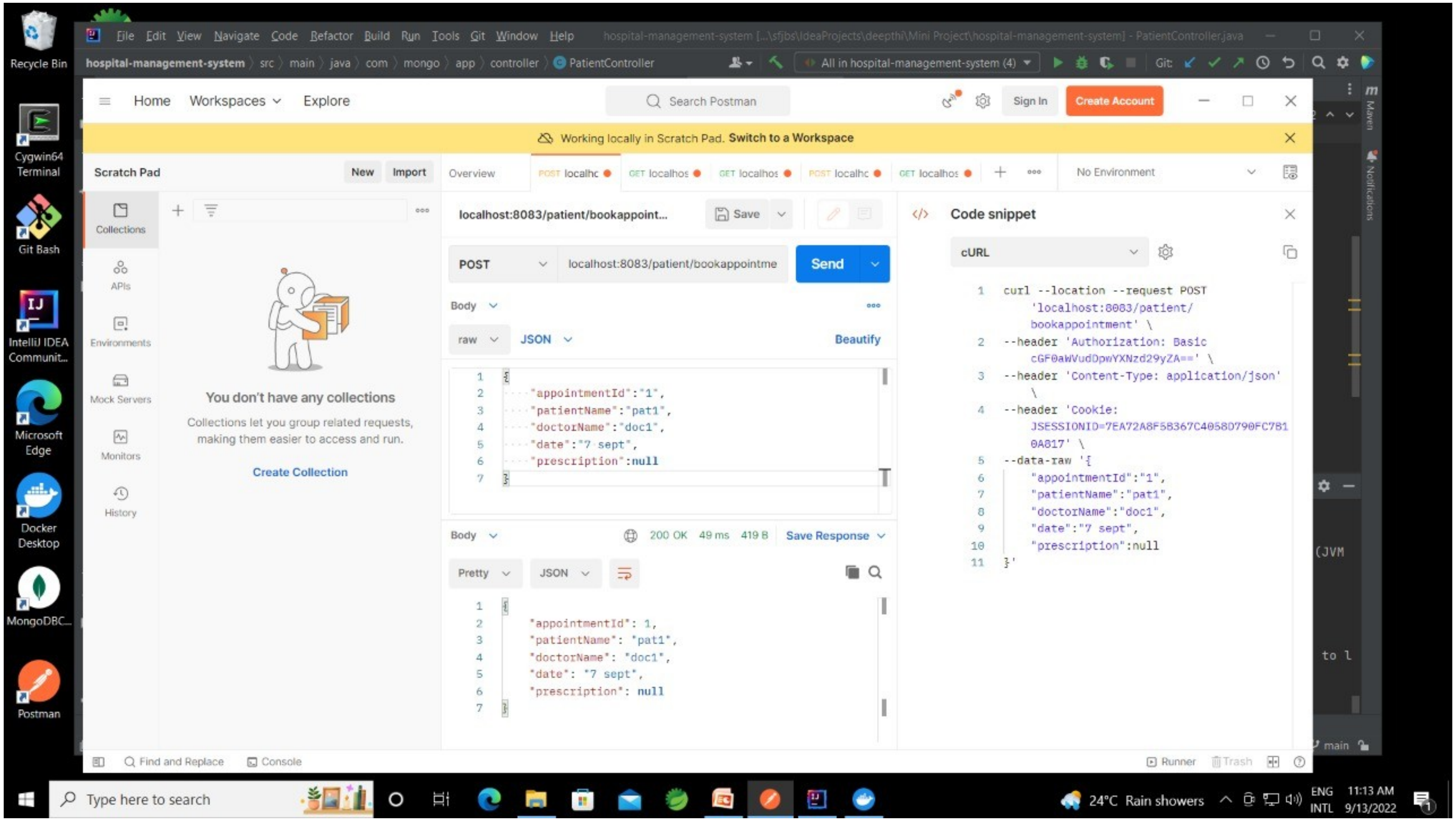
- In Spring Boot, there is no requirement for XML configuration (deployment descriptor). It uses convention over configuration software design paradigm that means it decreases the effort of the developer.

MONGO DB

- MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.
- Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema.
- A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

POSTMAN

- Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster. A comprehensive set of tools that help accelerate the API Lifecycle - from design, testing, documentation, and mocking to discovery.



localhost:8083/patient/bookappoint...

POST localhost:8083/patient/bookappointme

Body (raw, JSON):

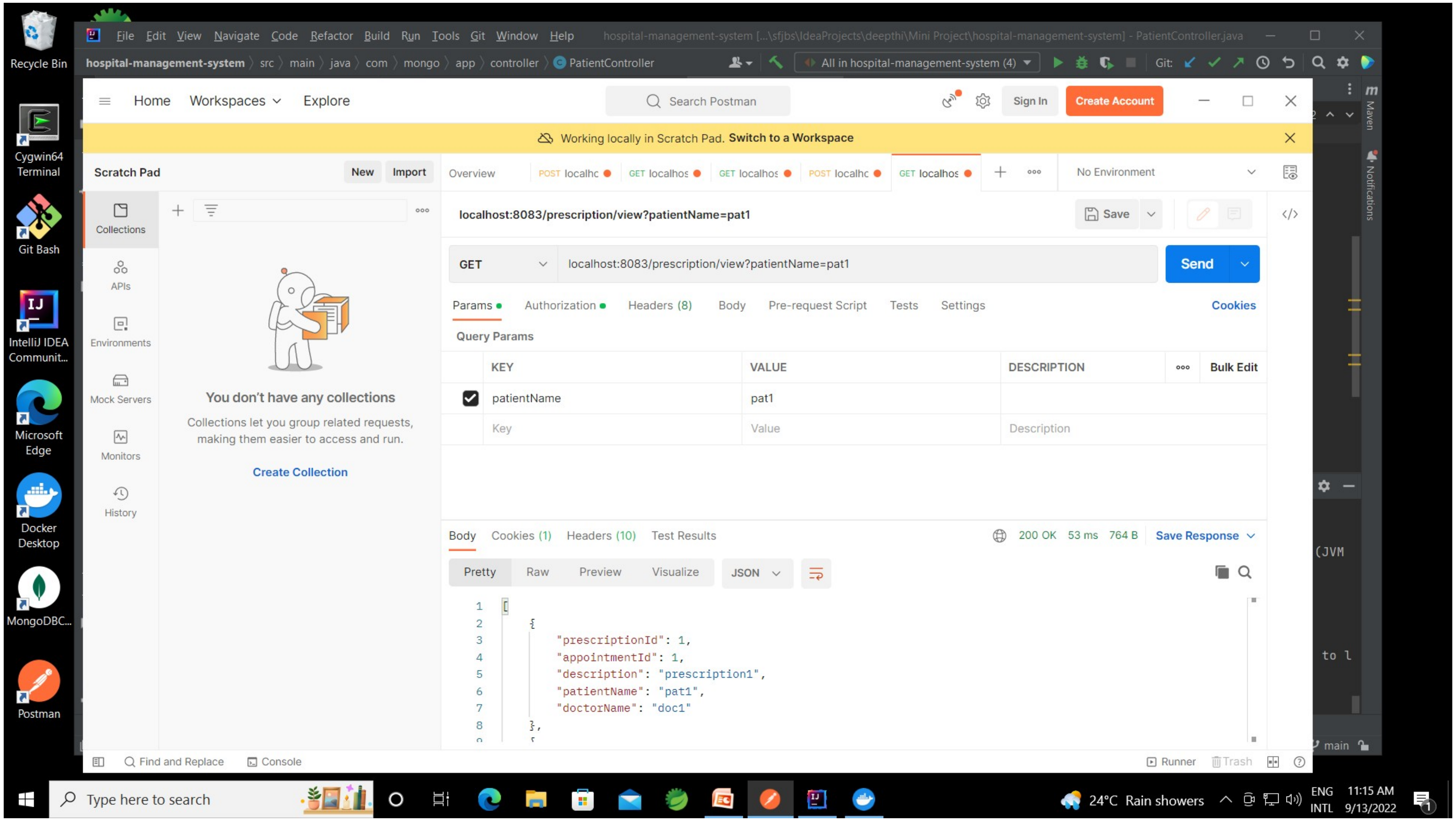
```
{
    "appointmentId":"1",
    "patientName":"pat1",
    "doctorName":"doc1",
    "date":"7 sept",
    "prescription":null
}
```

Response: 200 OK 49 ms 419 B

```
{
    "appointmentId": 1,
    "patientName": "pat1",
    "doctorName": "doc1",
    "date": "7 sept",
    "prescription": null
}
```

Code snippet (cURL):

```
curl --location --request POST
    'localhost:8083/patient/
    bookappointment' \
--header 'Authorization: Basic
    cGF0aWVudDpwYXNzd29yZA==' \
--header 'Content-Type: application/json'
    \
--header 'Cookie:
    JSESSIONID=7EA72A8F5B367C4058D790FC7B1
    0A817' \
--data-raw '{
    "appointmentId":"1",
    "patientName":"pat1",
    "doctorName":"doc1",
    "date":"7 sept",
    "prescription":null
}'
```



localhost:8083/prescription/view?patientName=pat1

GET localhost:8083/prescription/view?patientName=pat1

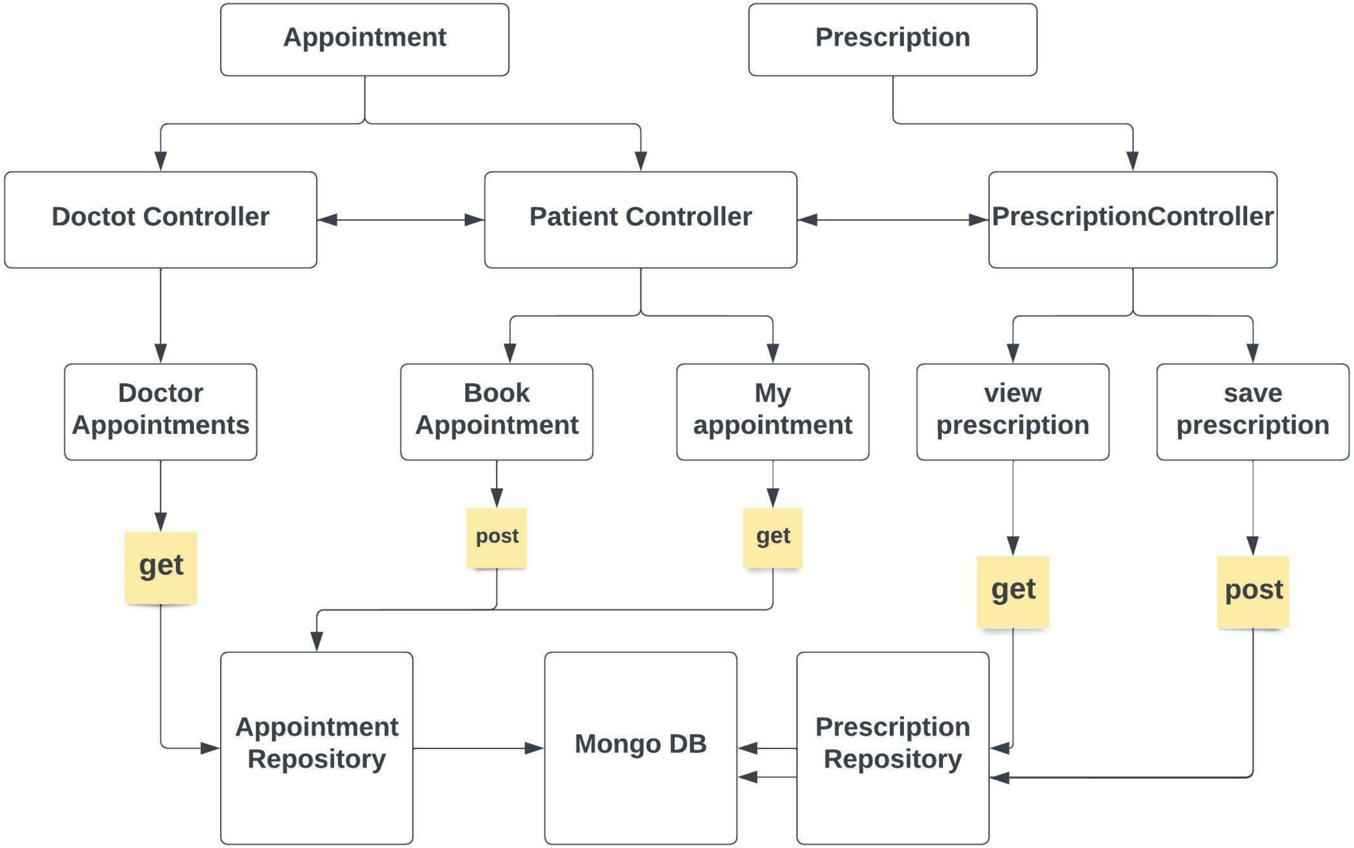
Query Params

	KEY	VALUE	DESCRIPTION
☑	patientName	pat1	
	Key	Value	Description

200 OK 53 ms 764 B

```
[
    {
        "prescriptionId": 1,
        "appointmentId": 1,
        "description": "prescription1",
        "patientName": "pat1",
        "doctorName": "doc1"
    },
```

ARCHITECTURE



USE CASE

Use case 1

- Patients to keep thier records of all appointments.
- Also useful for the patients who are unable to go to hospital.

Use case 2

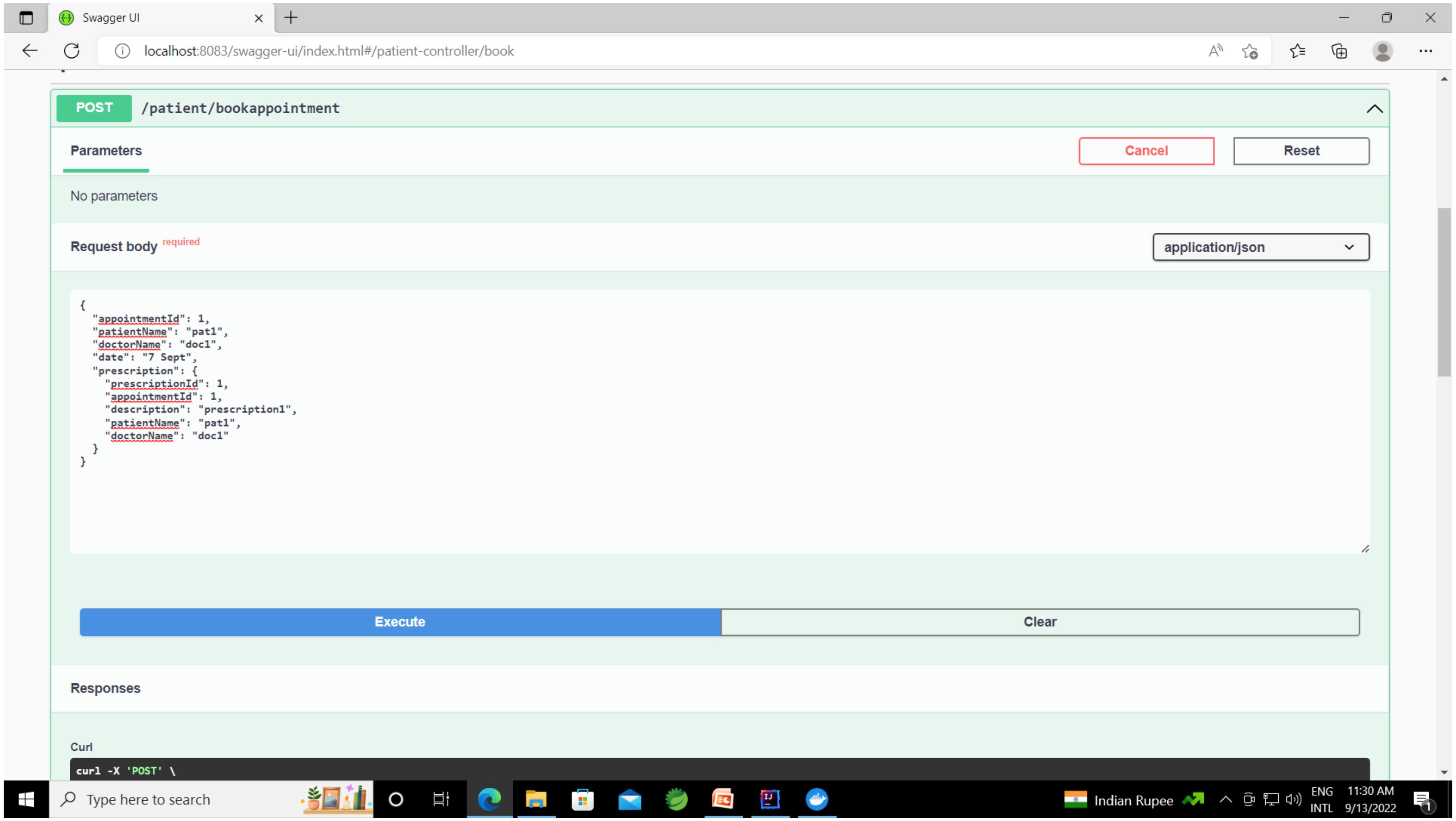
- Hospital management to have records of all the appointments as well as prescription given by doctor along with patient details.

TESTING

- The tool used for testing is SonarLint.
- The code coverage is-

Class	-	100%
Method	-	81%
Line	-	86%

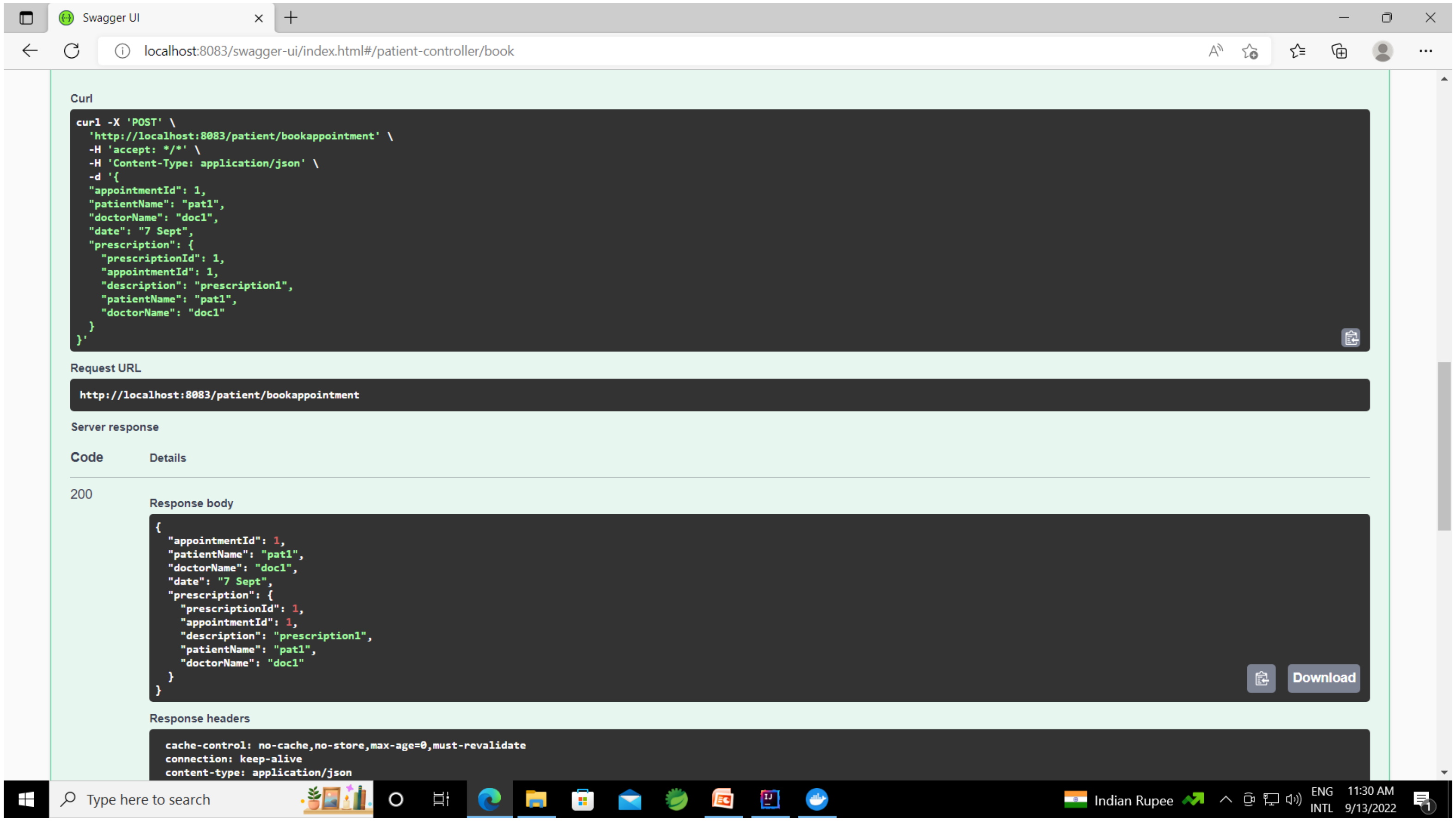
SWAGGER



localhost:8083/swagger-ui/index.html#/patient-controller/book

```
{
  "appointmentId": 1,
  "patientName": "pat1",
  "doctorName": "doc1",
  "date": "7 Sept",
  "prescription": {
    "prescriptionId": 1,
    "appointmentId": 1,
    "description": "prescription1",
    "patientName": "pat1",
    "doctorName": "doc1"
  }
}
```

```
curl -X 'POST' \
```



Curl

```
curl -X 'POST' \
  'http://localhost:8083/patient/bookappointment' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "appointmentId": 1,
  "patientName": "pat1",
  "doctorName": "doc1",
  "date": "7 Sept",
  "prescription": {
    "prescriptionId": 1,
    "appointmentId": 1,
    "description": "prescription1",
    "patientName": "pat1",
    "doctorName": "doc1"
  }
}'
```

Request URL

```
http://localhost:8083/patient/bookappointment
```

Server response

Code	Details
200	Response body

```
{
  "appointmentId": 1,
  "patientName": "pat1",
  "doctorName": "doc1",
  "date": "7 Sept",
  "prescription": {
    "prescriptionId": 1,
    "appointmentId": 1,
    "description": "prescription1",
    "patientName": "pat1",
    "doctorName": "doc1"
  }
}
```

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
```

Thank you!!