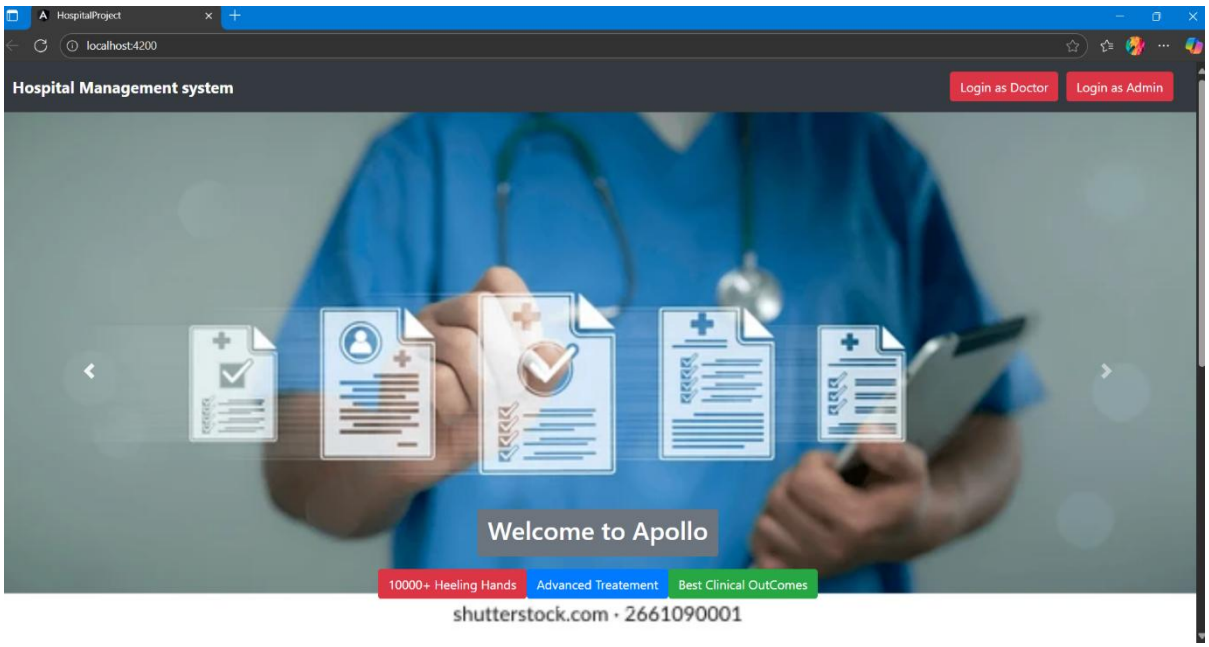
🏥 Hospital Management System – Project Report Template

- Project Title: *Hospital Management System*
- Name: Avula Praveen Kumar

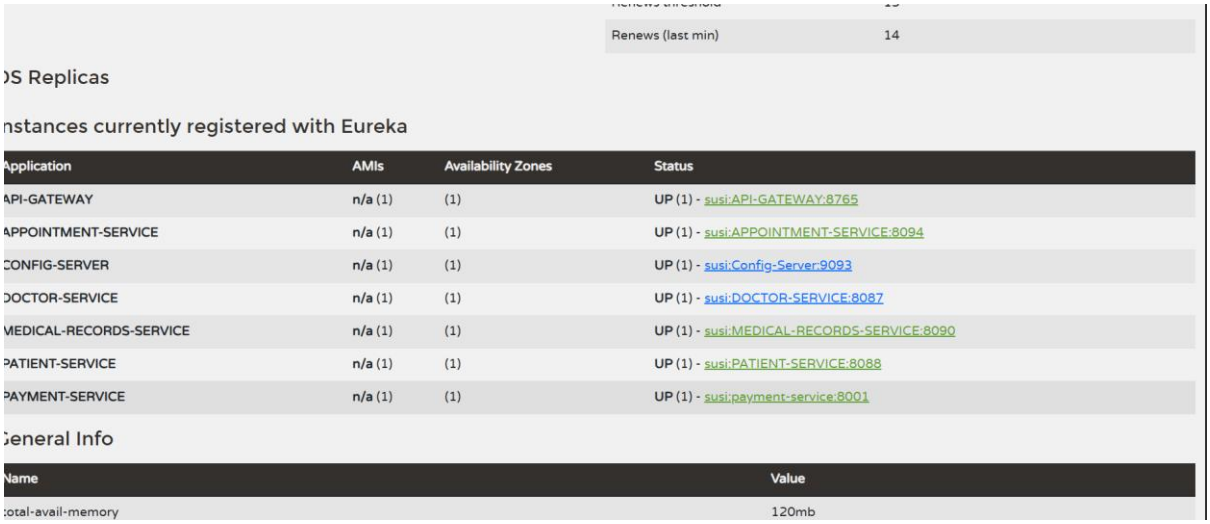
🎨 Frontend

- Angular + Bootstrap for responsive UI



🔧 Backend Microservices

- **Patient Service**
- **Doctor Service**
- **Appointment Service**
- **Billing Service**
- **Notification Service**
- **Audit/Logs Service**

- 
Renews (last min) 14

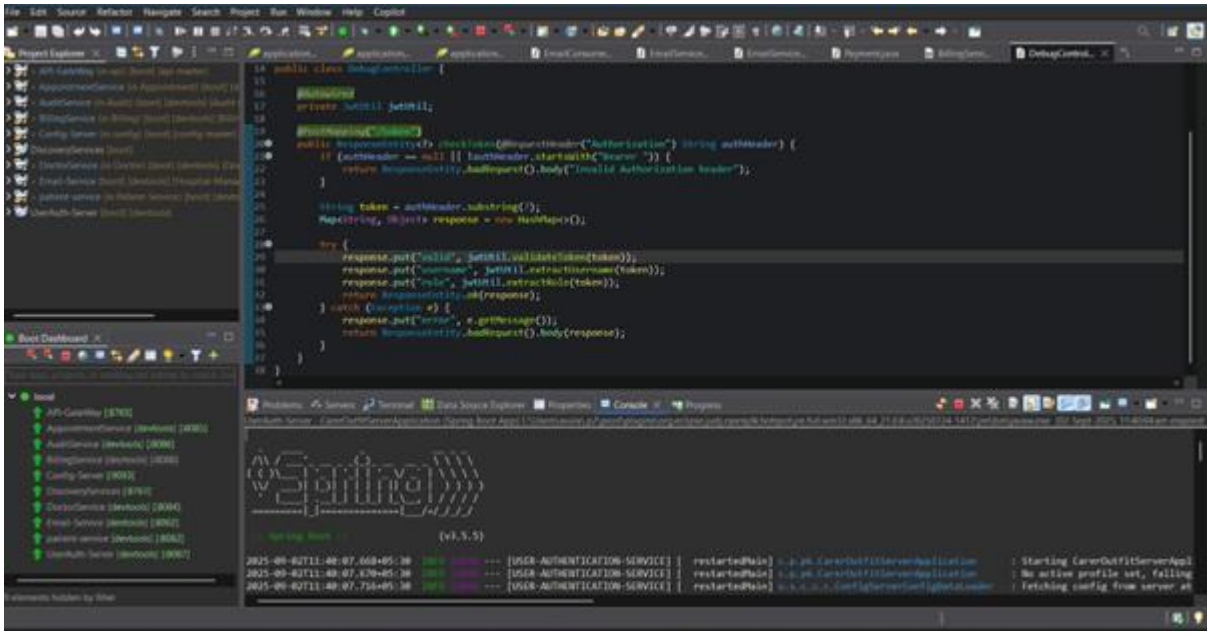
DS Replicas

nstances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - susi:API-GATEWAY:8765
APPOINTMENT-SERVICE	n/a (1)	(1)	UP (1) - susi:APPOINTMENT-SERVICE:8094
CONFIG-SERVER	n/a (1)	(1)	UP (1) - susi:Config-Server:9093
DOCTOR-SERVICE	n/a (1)	(1)	UP (1) - susi:DOCTOR-SERVICE:8087
MEDICAL-RECORDS-SERVICE	n/a (1)	(1)	UP (1) - susi:MEDICAL-RECORDS-SERVICE:8090
PATIENT-SERVICE	n/a (1)	(1)	UP (1) - susi:PATIENT-SERVICE:8088
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - susi:payment-service:8001

General Info

Name	Value
total-avail-memory	120mb



📈 Best Practices

- Pagination for large datasets (e.g., billing history)
- AOP logging for sensitive actions
- Kafka for async communication

- Audit readiness for HIPAA/GDPR compliance

📊 System Flow Diagram

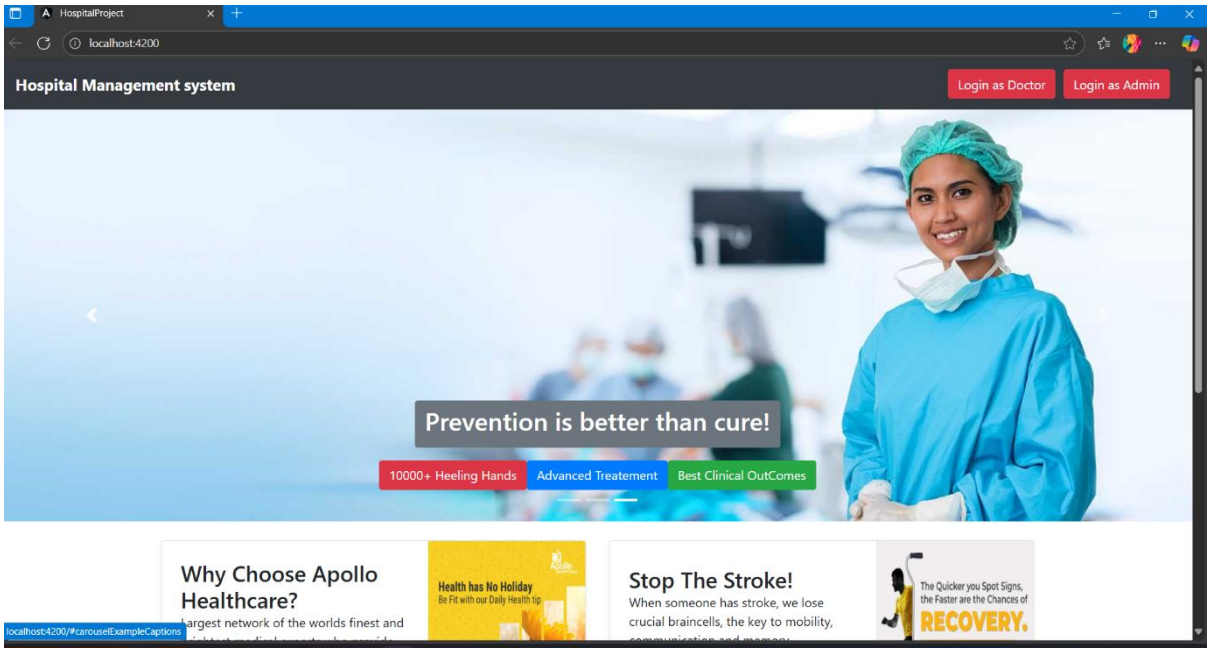
Users (Patient / Doctor / Admin) → **API Gateway** →
→ **JWT Security** →
→ Microservices (Patient, Doctor, Appointment, Billing, Notification) →
→ **MySQL + MongoDB** →
→ **Kafka Events + SMS/Email Alerts**

Home Page:- *Hospital Management System*

- ## Abstract

  The Hospital Management System (HMS) is a full-stack web application designed to streamline hospital operations. It enables patients to book appointments online, allows doctors to manage schedules and medical records, and provides admins with tools for billing, reporting, and audit logging. The system is built using Spring Boot microservices, Angular frontend, and integrates Kafka for event-driven communication.

  Home Page:-



🎯 Problem Statemen

Hospitals face challenges with manual paperwork, fragmented systems, and inefficient workflows. This project aims to:

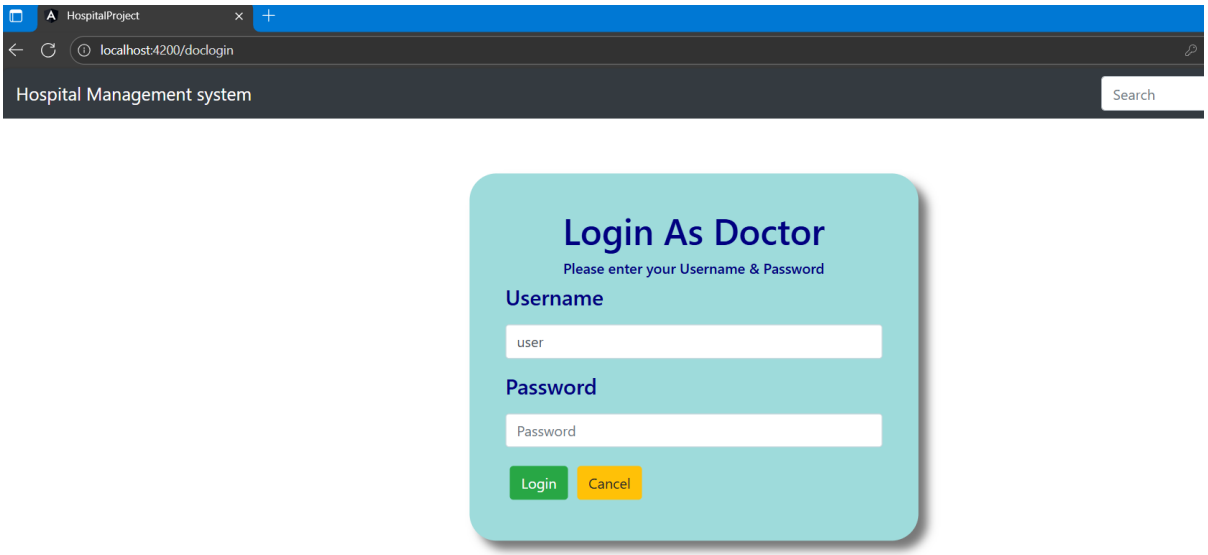
- Digitize patient-doctor interactions
- Securely manage medical records
- Automate billing and notifications
- Support concurrent operations across departments

🔄 Core Workflow

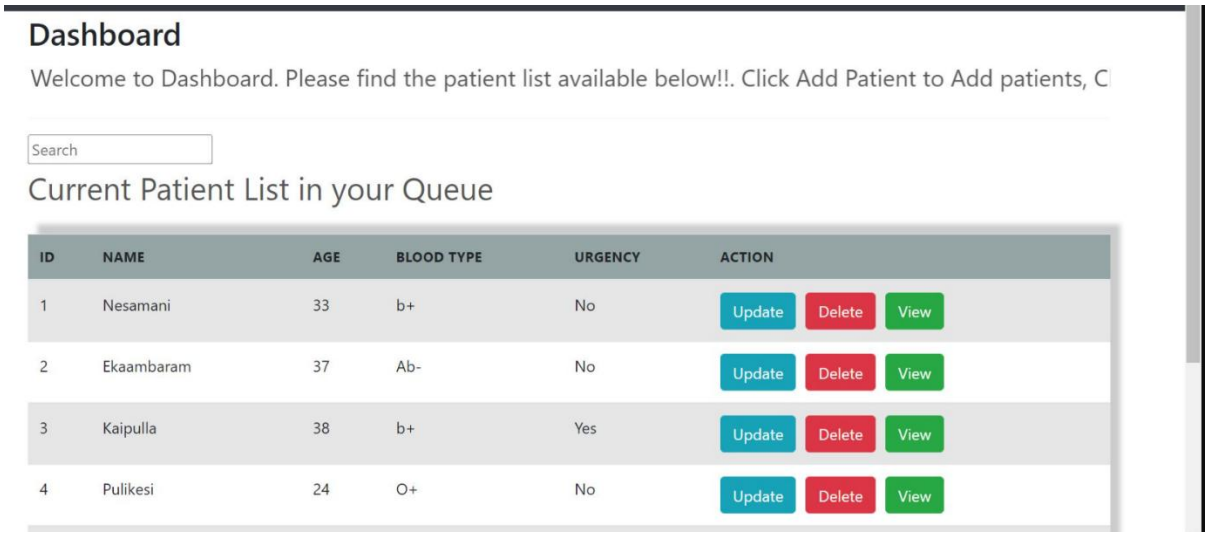
👨‍⚕️ Doctor Workflow

- Login via secure dashboard

  Doctor Login Page:-

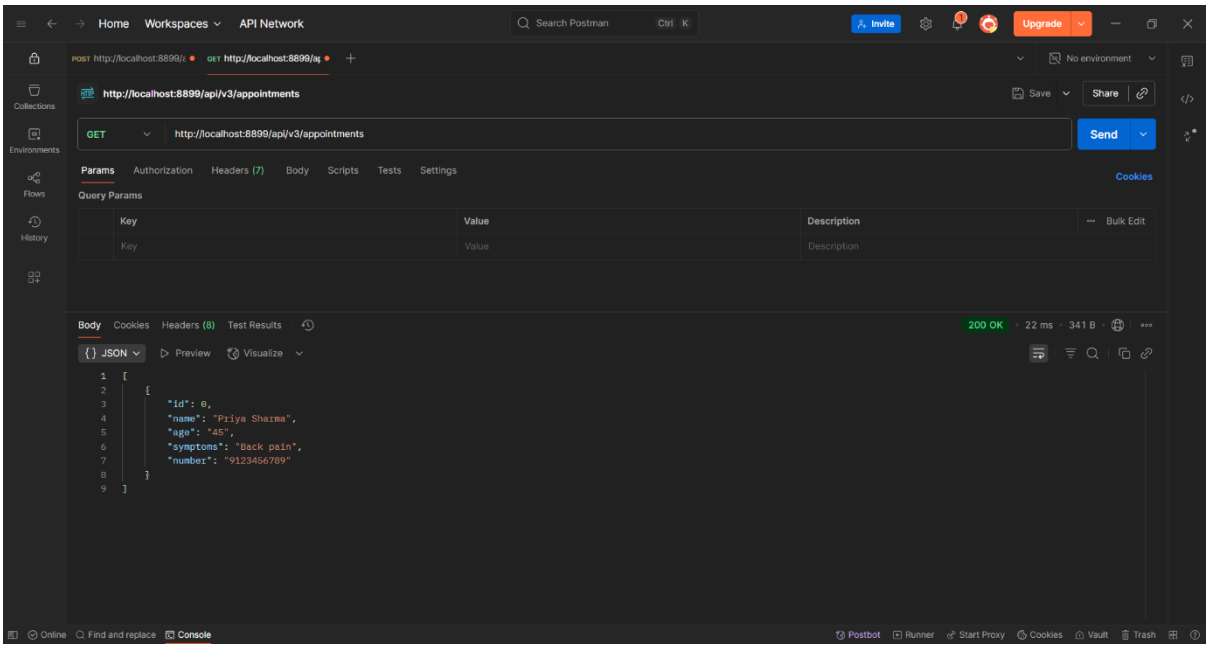


- View appointments:-

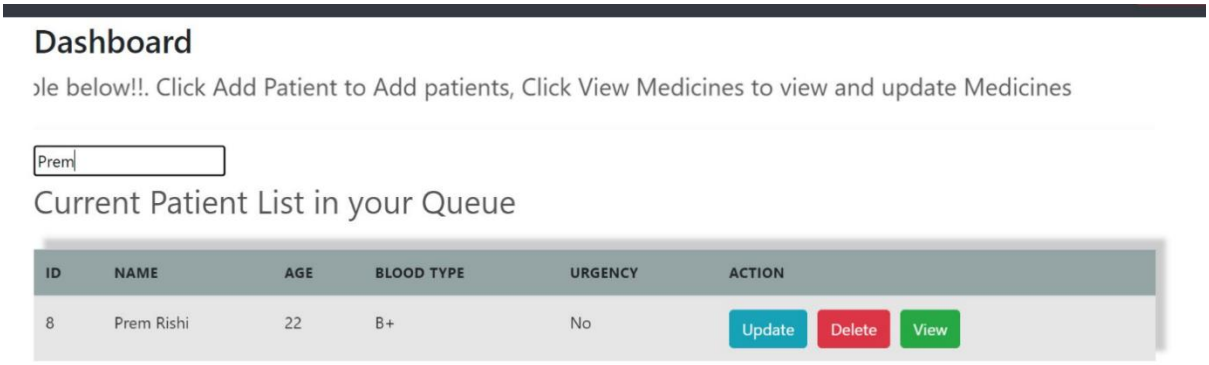


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

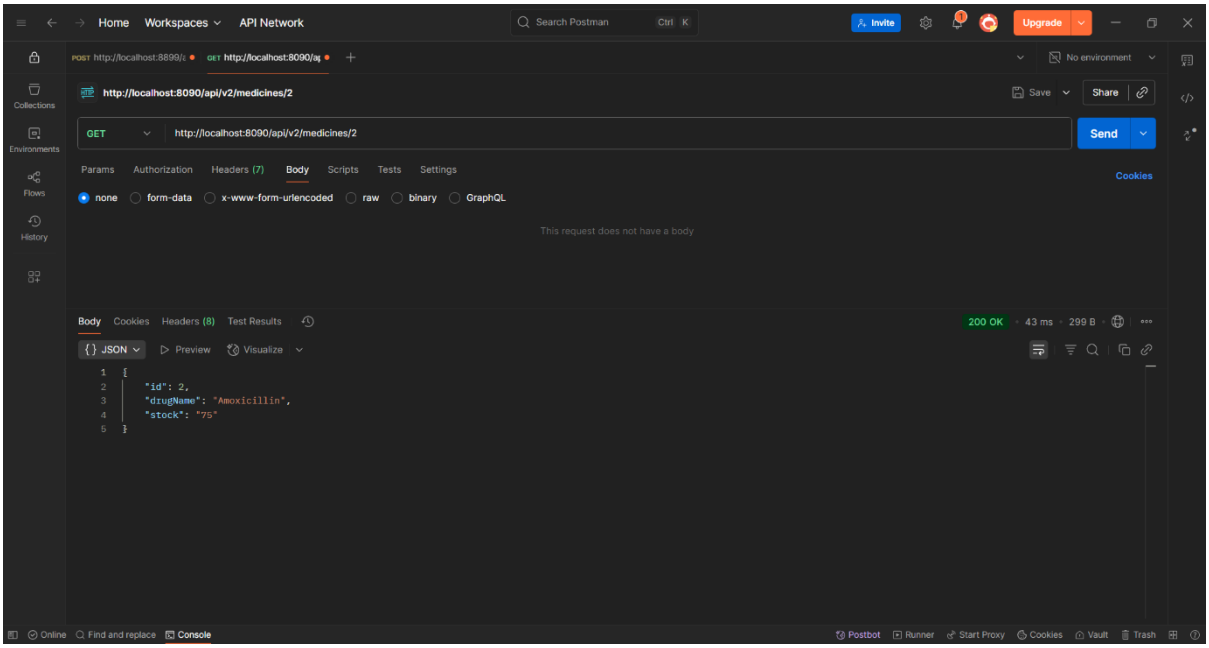
id	name	specialization	email	phone_number	experience_years
1	Dr. Asha Mehta	Cardiology	asha.mehta@hospital.com	9876543210	12
2	Dr. Rajiv Menon	Neurology	rajiv.menon@hospital.com	9123456780	15
3	Dr. Sneha Kapoor	Pediatrics	sneha.kapoor@hospital.com	9988776655	8
4	Dr. Arjun Rao	Orthopedics	arjun.rao@hospital.com	9090909090	10
5	Dr. Kavita Sharma	Dermatology	kavita.sharma@hospital.com	9191919191	6
NULL	NULL	NULL	NULL	NULL	NULL

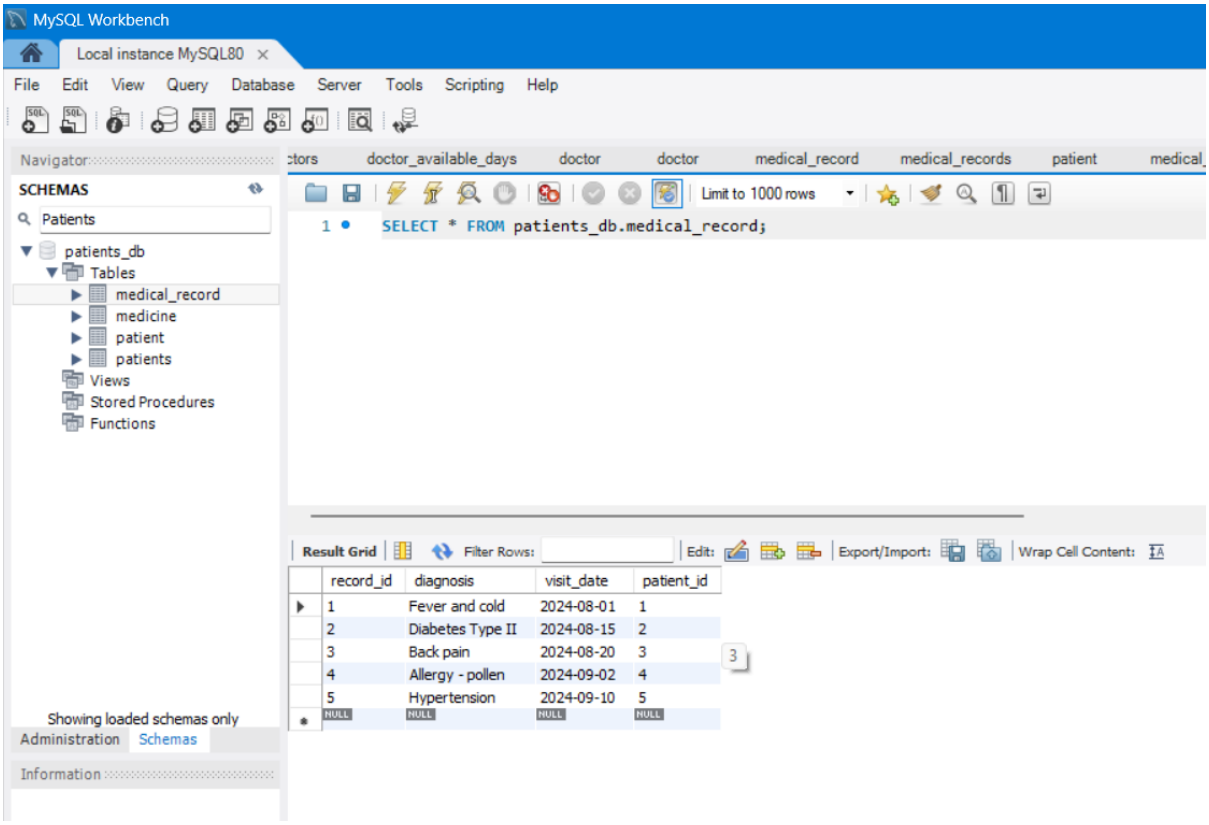


- Update diagnoses and prescriptions

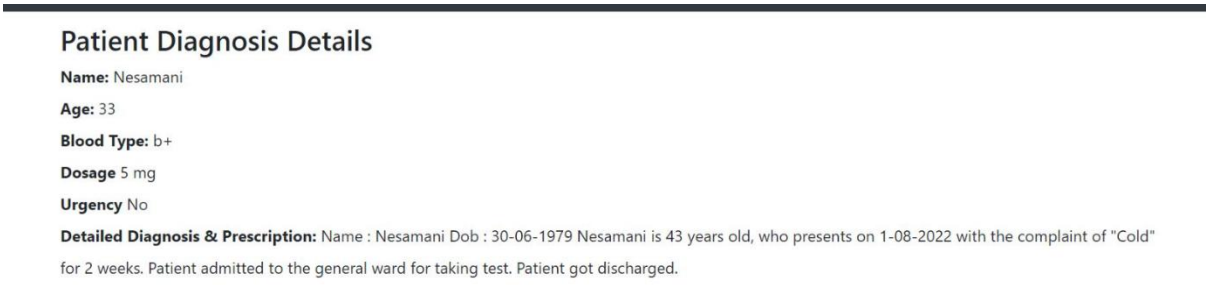

Dashboard
ole below!!. Click Add Patient to Add patients, Click View Medicines to view and update Medicines
Prem
Current Patient List in your Queue
ID
NAME
AGE
BLOOD TYPE
URGENCY
ACTION
8
Prem Rishi
22
B+
No
Update
Delete
View

BackEnd:-


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No environment
http://localhost:8090/api/v2/medicines/2
Save
Share
GET
http://localhost:8090/api/v2/medicines/2
Send
Params
Authorization
Headers (7)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (8)
Test Results
200 OK
43 ms
299 B
JSON
Preview
Visualize
{
"id": 2,
"drugName": "Amoxicillin",
"stock": "75"
}
Collections
Environments
Flows
History
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash

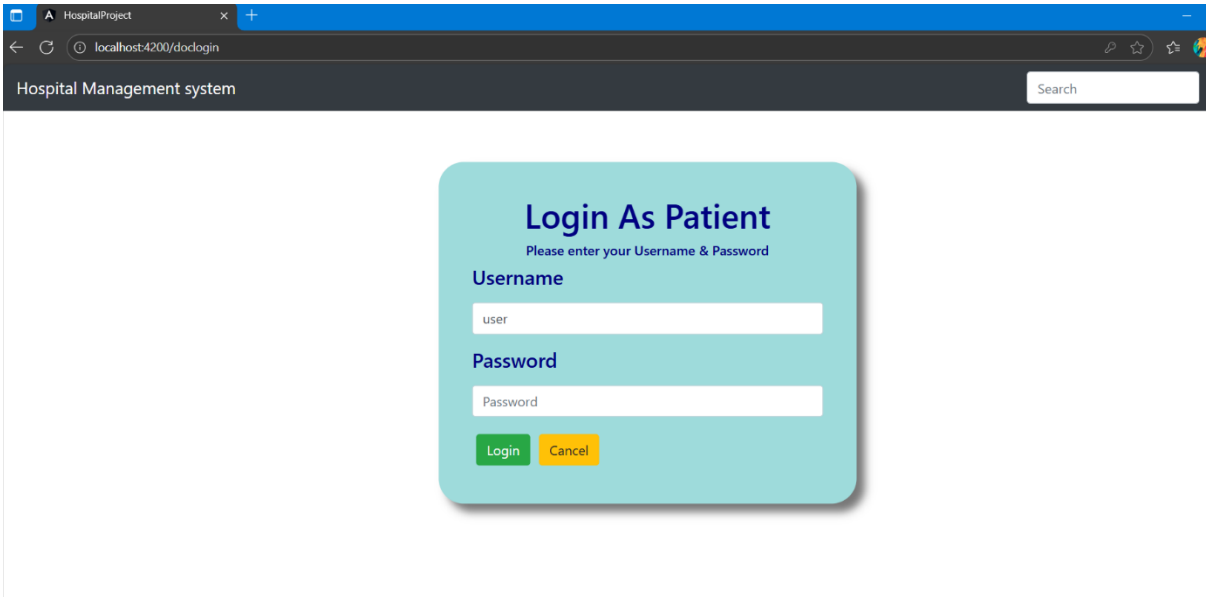


- Access patient history

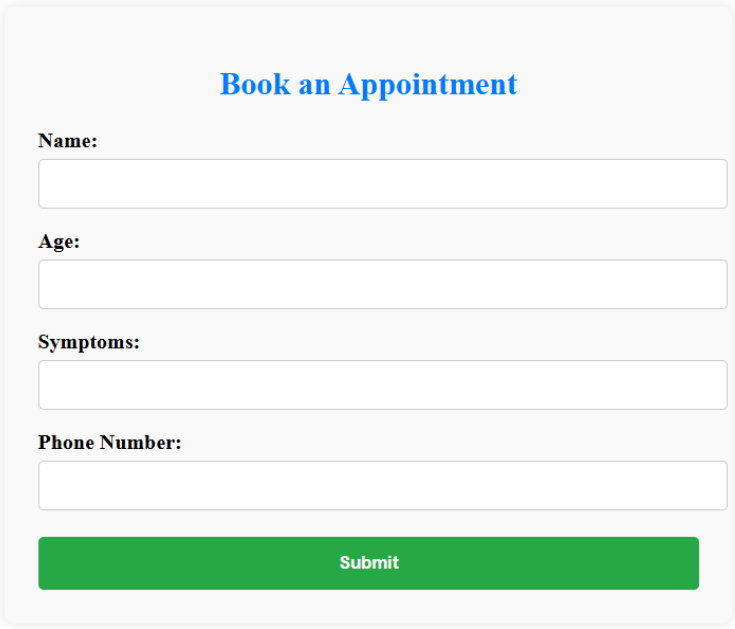


👨 Patient Workflow

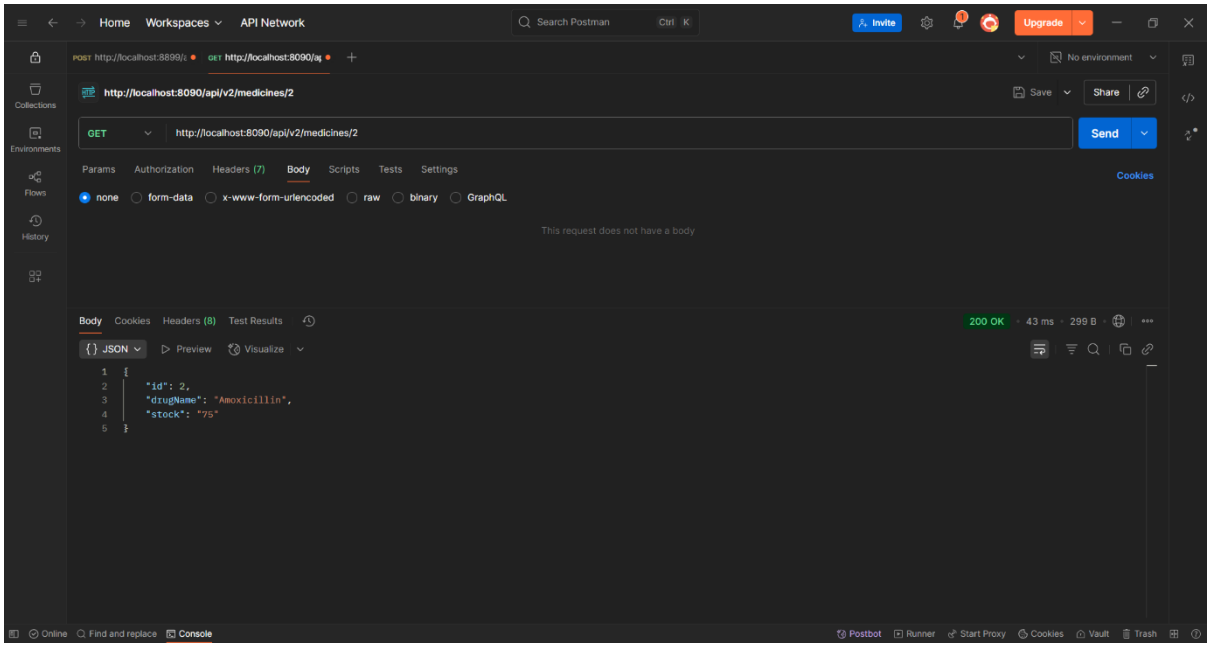
- Register and authenticate (JWT)

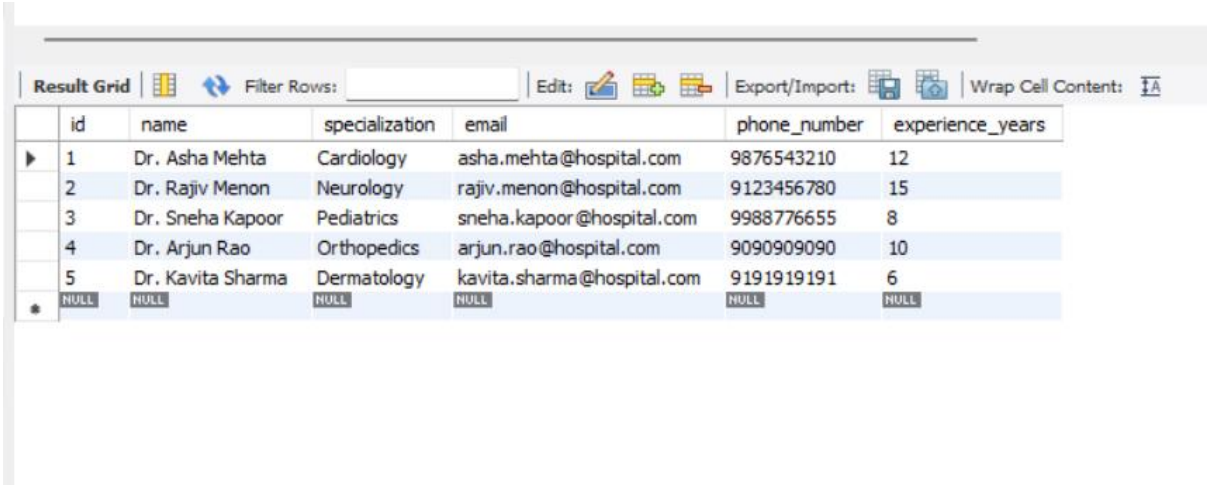


- Bookappointments:-



-





Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	specialization	email	phone_number	experience_years
1	Dr. Asha Mehta	Cardiology	asha.mehta@hospital.com	9876543210	12
2	Dr. Rajiv Menon	Neurology	rajiv.menon@hospital.com	9123456780	15
3	Dr. Sneha Kapoor	Pediatrics	sneha.kapoor@hospital.com	9988776655	8
4	Dr. Arjun Rao	Orthopedics	arjun.rao@hospital.com	9090909090	10
5	Dr. Kavita Sharma	Dermatology	kavita.sharma@hospital.com	9191919191	6
NULL	NULL	NULL	NULL	NULL	NULL

- Receive SMS/email notifications

- Pay bills online/offline

Pay Your Hospital Bill

Patient ID:

02458

Amount (₹):

5500

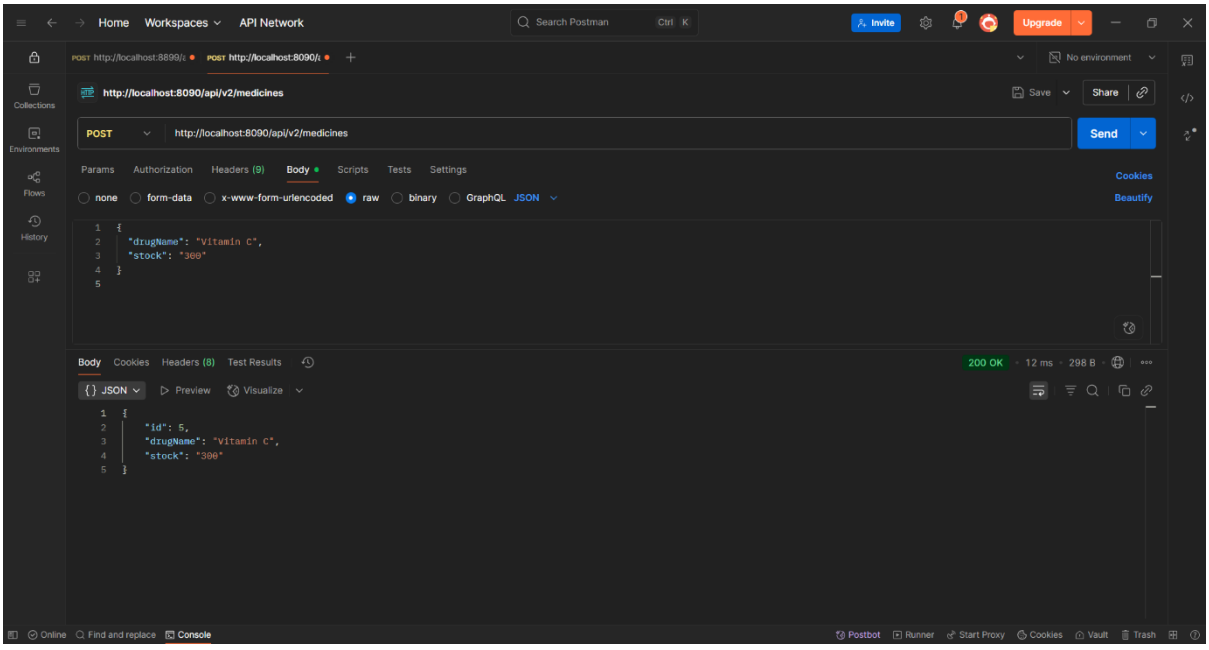
Payment Method:

Online

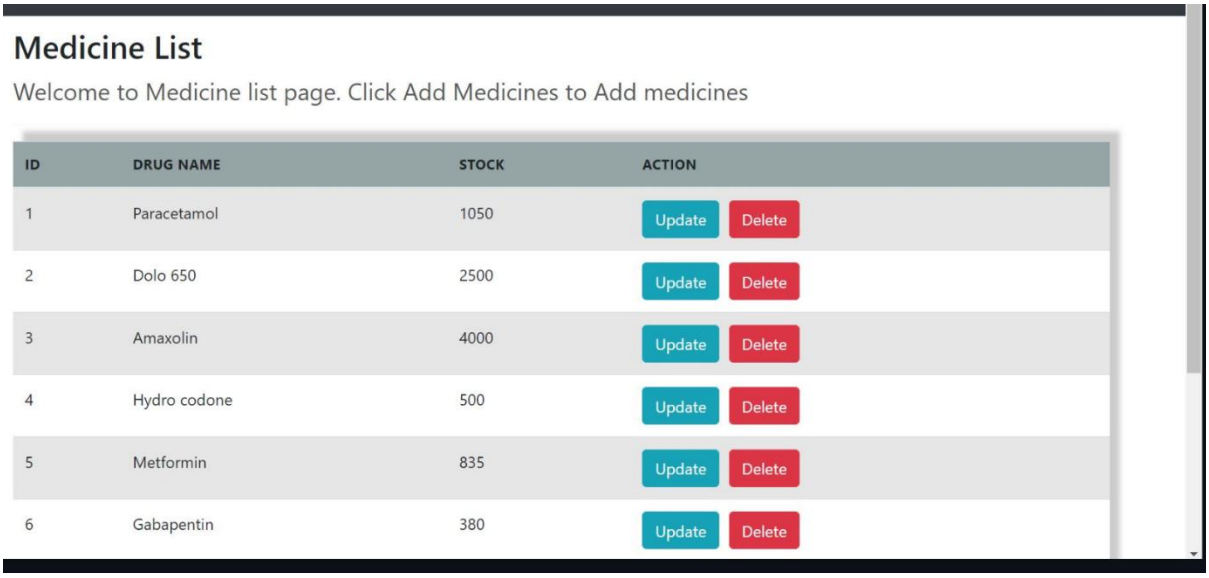
Status:

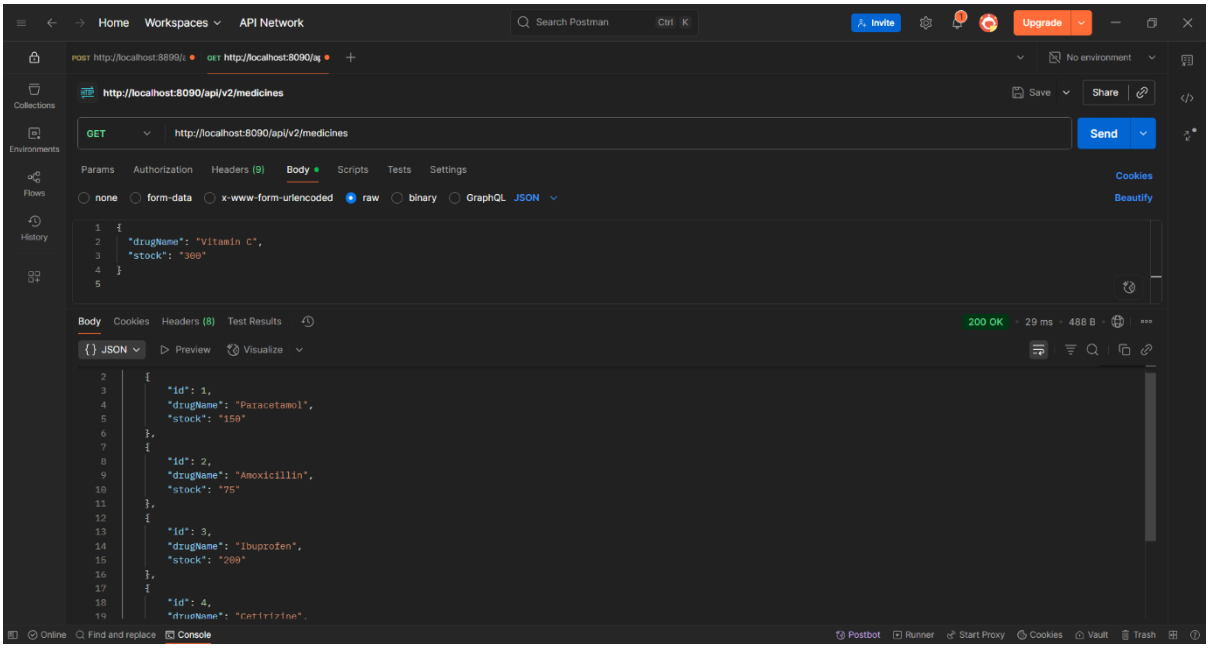
Paid

Submit Payment



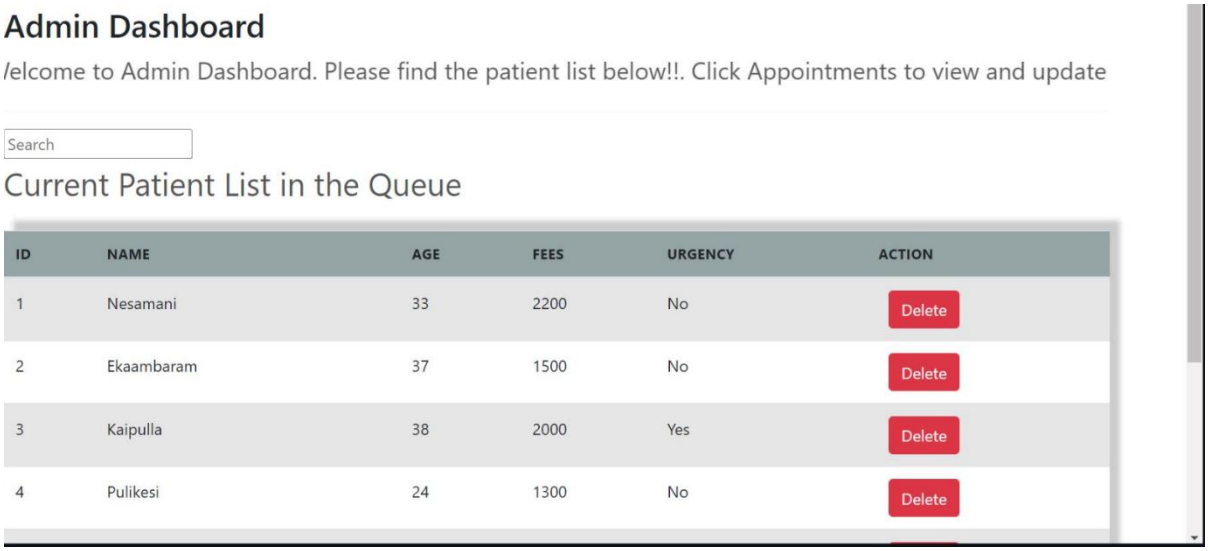
- View medical records





Admin Workflow

- Handle billing and insurance



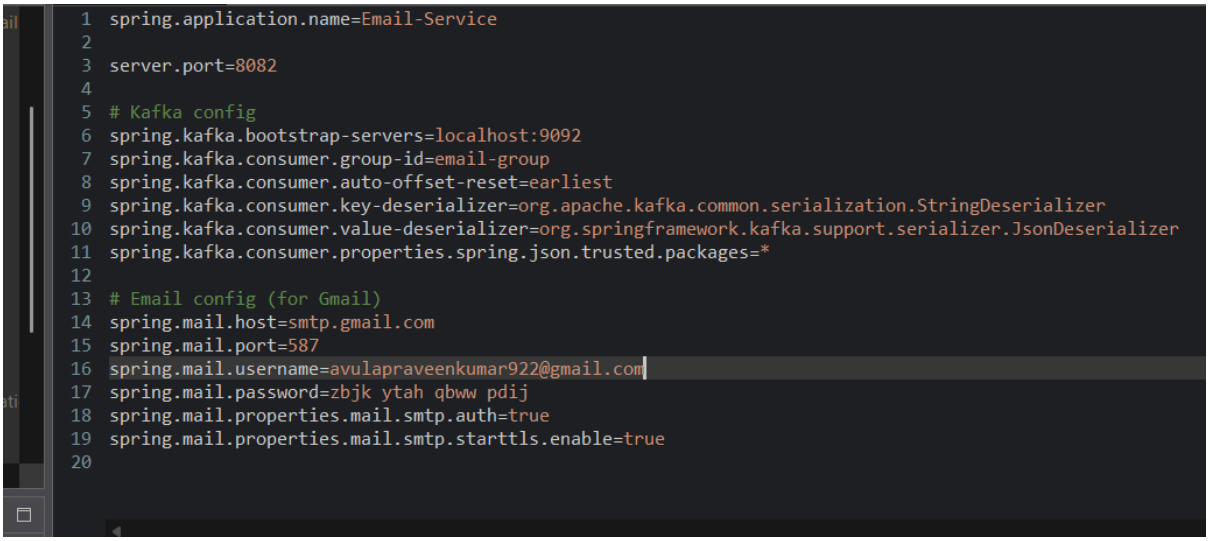
- Generate audit logs and reports

Audit Logs

Timestamp	Actor	Action	Details
2025-09-04T14:45:12	Admin	Delete Patient	Removed patient record: ID 3 - Kaipulla
2025-09-04T14:46:30	Doctor	Update Diagnosis	Updated prescription for Nesamani: 5mg dosage for cold symptoms
2025-09-04T14:47:55	Patient	Book Appointment	Appointment booked with Dr. Neha Mehra for 2025-09-06
2025-09-04T14:49:10	Admin	Bill Payment	Processed ₹2200 payment for patient ID 1 - Nesamani via Offline
2025-09-04T14:50:42	System	Send Notification	SMS sent to 9159821775 confirming appointment for Prem Rishi

Messaging

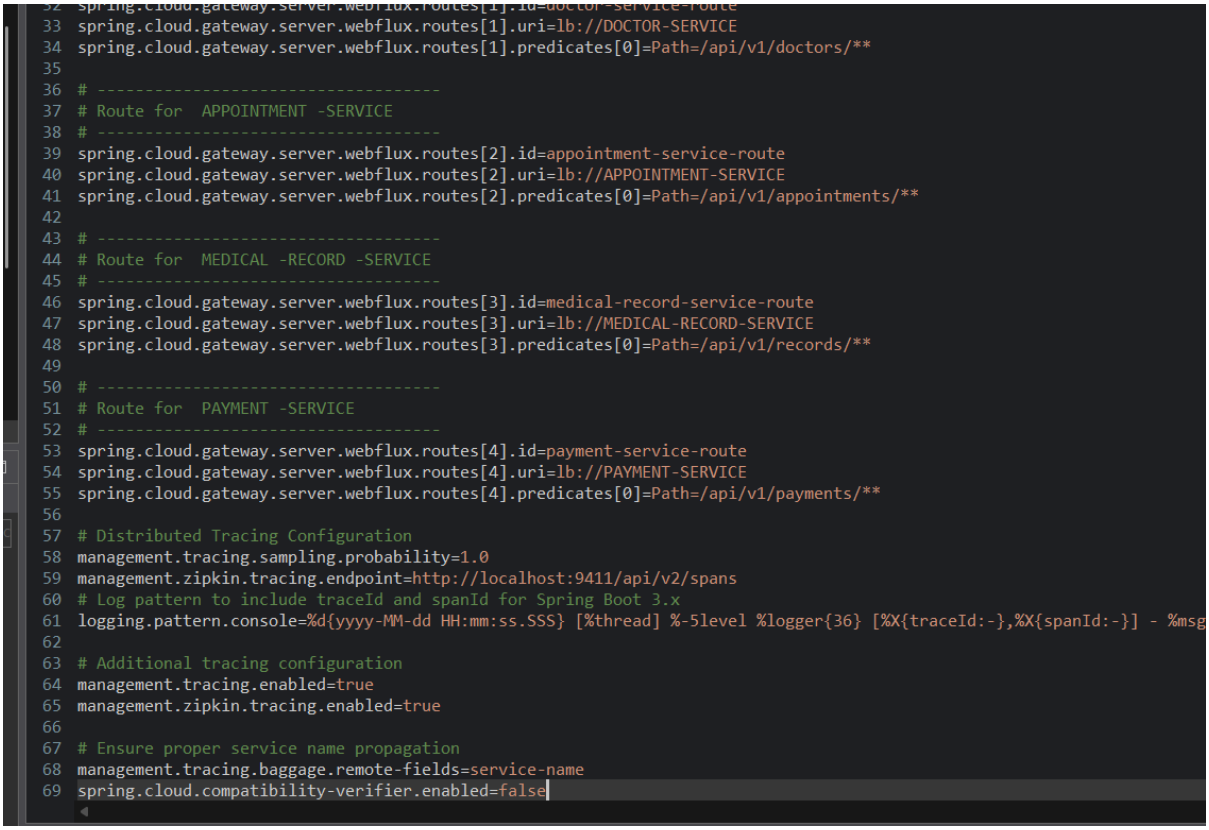
- **Apache Kafka**: Event-driven updates and notifications



```
spring.application.name=Email-Service

server.port=8082

# Kafka config
spring.kafka.bootstrap-servers=localhost:9092
spring.kafka.consumer.group-id=email-group
spring.kafka.consumer.auto-offset-reset=earliest
spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.consumer.value-deserializer=org.springframework.kafka.support.serializer.JsonDeserializer
spring.kafka.consumer.properties.spring.json.trusted.packages=*

# Email config (for Gmail)
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=avulapraveenkumar922@gmail.com
spring.mail.password=zbjk ytah qbww pdij
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

🔐 Security

- JWT/OAuth2 for role-based access (Doctor, Patient, Admin)



```
spring.cloud.gateway.server.webflux.routes[1].uri=lb://DOCTOR-SERVICE
spring.cloud.gateway.server.webflux.routes[1].predicates[0]=Path=/api/v1/doctors/**

# ------------------------------------
# Route for  APPOINTMENT -SERVICE
# ------------------------------------
spring.cloud.gateway.server.webflux.routes[2].id=appointment-service-route
spring.cloud.gateway.server.webflux.routes[2].uri=lb://APPOINTMENT-SERVICE
spring.cloud.gateway.server.webflux.routes[2].predicates[0]=Path=/api/v1/appointments/**

# ------------------------------------
# Route for  MEDICAL -RECORD -SERVICE
# ------------------------------------
spring.cloud.gateway.server.webflux.routes[3].id=medical-record-service-route
spring.cloud.gateway.server.webflux.routes[3].uri=lb://MEDICAL-RECORD-SERVICE
spring.cloud.gateway.server.webflux.routes[3].predicates[0]=Path=/api/v1/records/**

# ------------------------------------
# Route for  PAYMENT -SERVICE
# ------------------------------------
spring.cloud.gateway.server.webflux.routes[4].id=payment-service-route
spring.cloud.gateway.server.webflux.routes[4].uri=lb://PAYMENT-SERVICE
spring.cloud.gateway.server.webflux.routes[4].predicates[0]=Path=/api/v1/payments/**

# Distributed Tracing Configuration
management.tracing.sampling.probability=1.0
management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans
# Log pattern to include traceId and spanId for Spring Boot 3.x
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} [%X{traceId:-},%X{spanId:-}] - %msg

# Additional tracing configuration
management.tracing.enabled=true
management.zipkin.tracing.enabled=true

# Ensure proper service name propagation
management.tracing.baggage.remote-fields=service-name
spring.cloud.compatibility-verifier.enabled=false
```

✅ Outcome

By the end of this project, the HMS delivers:

- Secure login and role-based dashboards
- Online appointment booking
- Billing and payment history
- Medical record access
- Real-time notifications
- Admin reporting and audit logs