

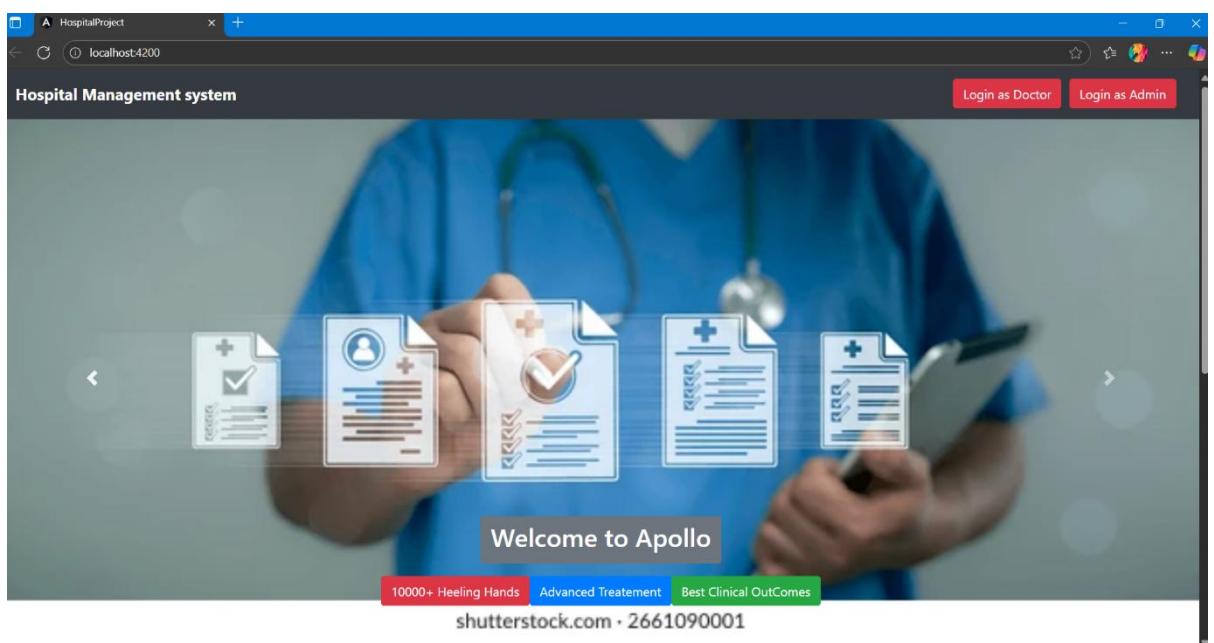
Hospital Management System – Project Report Template

- 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

OS Replicas

Instances 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

The screenshot shows a Java IDE (IntelliJ IDEA) with several tabs open. One tab contains Java code for an OAuth controller:

```

14 public class OAuthController {
15
16     private String jwtId;
17
18     @PostMapping("token")
19     public ResponseEntity<Object> checkAuthHeader(@RequestHeader("Authorization") String authHeader) {
20         if (authHeader == null || !authHeader.startsWith("Bearer ")) {
21             return ResponseEntity.badRequest().body("invalid Authorization header");
22         }
23
24         String token = authHeader.substring(7);
25         Map<String, Object> response = new HashMap<>();
26
27         try {
28             response.put("email", jwtId.substring(token));
29             response.put("name", jwtId.substring(token));
30             response.put("role", jwtId.substring(token));
31             return ResponseEntity.ok(response);
32         } catch (Exception e) {
33             response.put("error", e.getMessage());
34         }
35         return ResponseEntity.badRequest().body(response);
36     }
37 }

```

The terminal window below shows log output from a Spring Boot application:

```

2025-09-02T11:40:07.668+05:30 --- [User-AUTHENTICATION-SERVICE] | restartedMain | java.lang.IllegalArgumentException: No active profile set, falling back to default profiles
2025-09-02T11:40:07.679+05:30 --- [User-AUTHENTICATION-SERVICE] | restartedMain | java.lang.IllegalArgumentException: No active profile set, falling back to default profiles
2025-09-02T11:40:07.754+05:30 --- [User-AUTHENTICATION-SERVICE] | restartedMain |

```



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

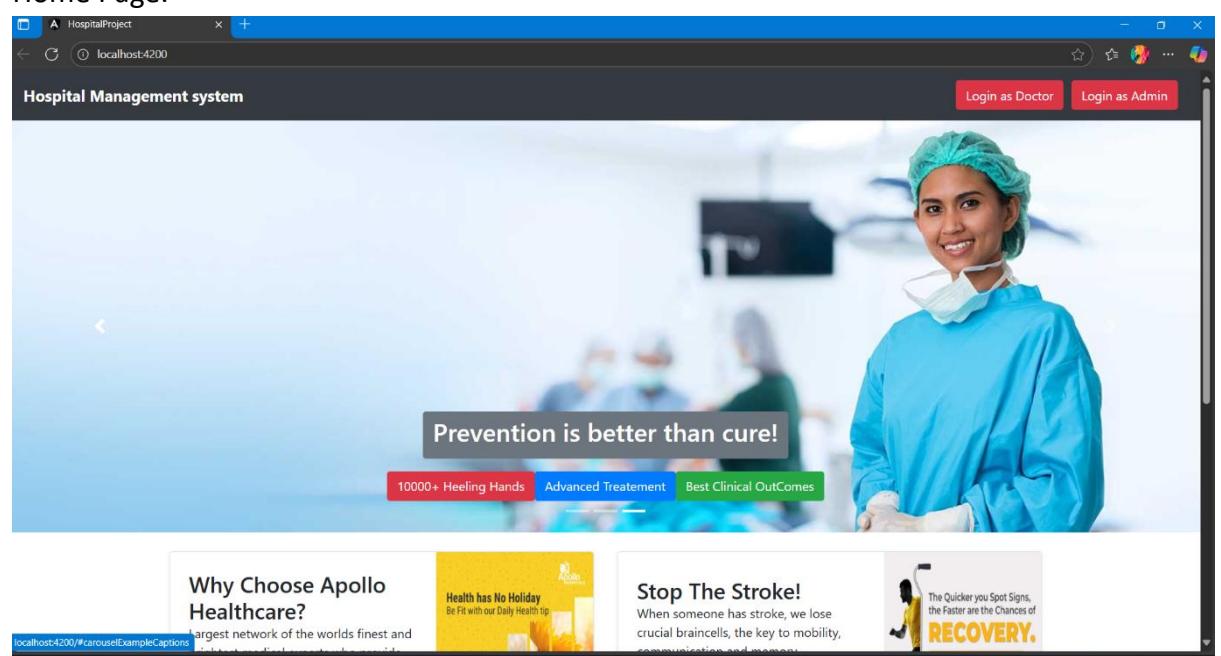
Users (Patient / Doctor / Admin) → API Gateway →
→ JWT Security →
→ Microservices (Patient, Doctor, Appointment, Billing, Notification) →
→ MySQL + MongoDB →
→ Kafka Events + SMS/Email Alerts

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 Statement

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

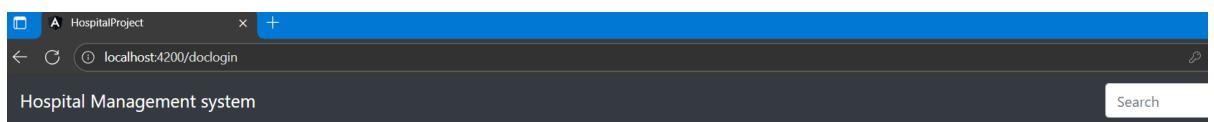
(Core Workflow)



Doctor Workflow

- Login via secure dashboard

Doctor Login Page:-



Login As Doctor

Please enter your Username & Password

Username

Password

Login **Cancel**

- View appointments:-

Dashboard

Welcome to Dashboard. Please find the patient list available below!!.. Click Add Patient to Add patients, C

Search

Current Patient List in your Queue

ID	NAME	AGE	BLOOD TYPE	URGENCY	ACTION
1	Nesamani	33	b+	No	<button>Update</button> <button>Delete</button> <button>View</button>
2	Ekaambaram	37	Ab-	No	<button>Update</button> <button>Delete</button> <button>View</button>
3	Kaipulla	38	b+	Yes	<button>Update</button> <button>Delete</button> <button>View</button>
4	Pulikesi	24	O+	No	<button>Update</button> <button>Delete</button> <button>View</button>

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

Home Workspaces API Network Search Postman Ctrl K Invite Share Upgrade

POST http://localhost:8899/ GET http://localhost:8899/api/v3/appointments

http://localhost:8899/api/v3/appointments

GET http://localhost:8899/api/v3/appointments

Params Authorization Headers Body Scripts Tests Settings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers Test Results

{ } JSON ▾ Preview Visualize

```

1  [
2   {
3     "id": 0,
4     "name": "Priya Sharma",
5     "age": "45",
6     "symptoms": "Back pain",
7     "number": "9123456789"
8   }
9 ]

```

200 OK 22 ms 341 B

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash

- Update diagnoses and prescriptions

Dashboard

Role 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	<button>Update</button> <button>Delete</button> <button>View</button>

BackEnd:-

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (empty), 'Environments' (empty), 'Flows' (empty), and 'History'. The main area has a header with 'POST http://localhost:8890/' and 'GET http://localhost:8090/api/v2/medicines/2'. Below this, a 'GET' method is selected with the URL 'http://localhost:8090/api/v2/medicines/2'. The 'Body' tab is active, showing 'none' selected. The response section shows a status of '200 OK' with a response time of '43 ms' and a size of '299 B'. The response body is displayed as JSON:

```
1 [ { 2 "id": 2, 3 "drugName": "Amoxicillin", 4 "stock": "75" 5 } ]
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, it says "Local instance MySQL80 x". Below the menu bar, there are several icons for different tools. The left sidebar shows the "SCHEMAS" section with "Patients" selected, revealing tables like "medical_record", "medicine", "patient", and "patients". The main area displays a SQL query: "SELECT * FROM patients_db.medical_record;". The results grid shows the following data:

record_id	diagnosis	visit_date	patient_id
1	Fever and cold	2024-08-01	1
2	Diabetes Type II	2024-08-15	2
3	Back pain	2024-08-20	3
4	Allergy - pollen	2024-09-02	4
5	Hypertension	2024-09-10	5
*	HULL	HULL	HULL

- Access patient history

Patient Diagnosis Details

Name: Nesamani

Age: 33

Blood Type: b+

Dosage 5 mg

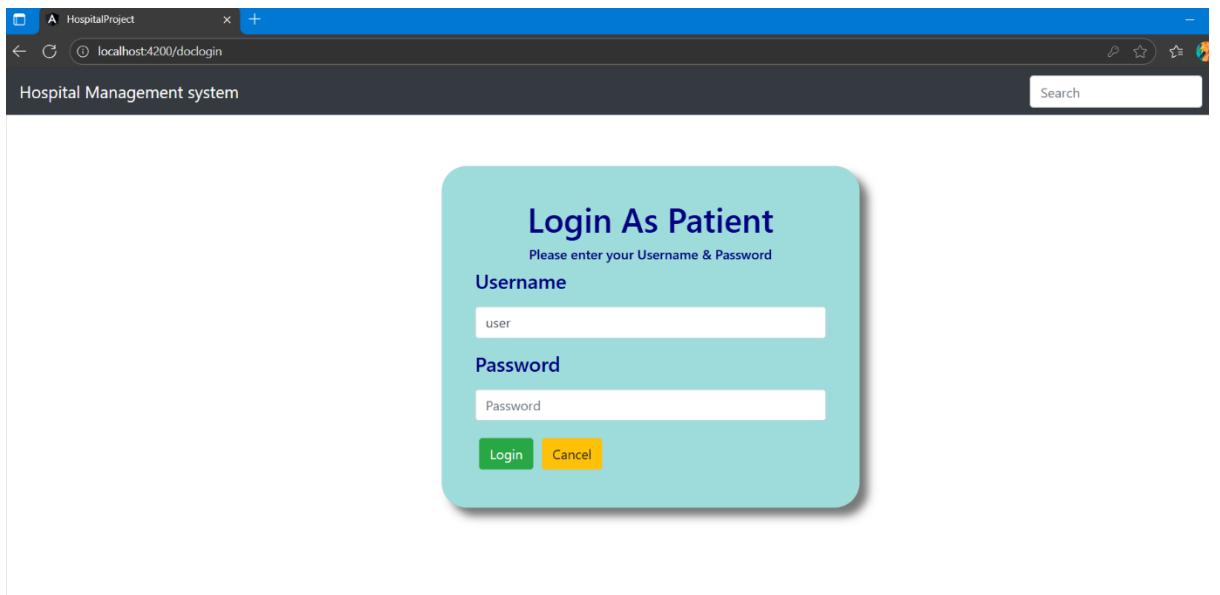
Urgency No

Detailed Diagnosis & Prescription: Name : Nesamani Dob : 30-06-1979 Nesamani is 43 years old, who presents on 1-08-2022 with the complaint of "Cold" for 2 weeks. Patient admitted to the general ward for taking test. Patient got discharged.



Patient Workflow

- Register and authenticate (JWT)



- Book appointments:-

A screenshot of a web form titled "Book an Appointment" in blue font at the top center. The form consists of five text input fields stacked vertically. The first field is labeled "Name:" and contains a placeholder text box. The second field is labeled "Age:" and also contains a placeholder text box. The third field is labeled "Symptoms:" and contains a placeholder text box. The fourth field is labeled "Phone Number:" and contains a placeholder text box. Below these four fields is a large green rectangular button with the word "Submit" in white.

-

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8090/api/v2/medicines/2`. The response status is 200 OK, with a response time of 43 ms and a size of 299 B. The response body is a JSON object:

```
{ } JSON ▾ 1 { 2   "id": 2, 3   "drugName": "Amoxicillin", 4   "stock": 75 5 }
```

A screenshot of a browser displaying a grid of data. The grid has the following structure:

	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:

Amount (₹):

Payment Method:

Status:

Submit Payment

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8090/api/v2/medicines`. The method is `POST`. The body contains the following JSON payload:

```

1 {
2   "drugName": "Vitamin C",
3   "stock": "300"
4 }
5
    
```

The response status is `200 OK` with a response time of `12 ms` and a size of `298 B`. The response body is identical to the sent body.

- View medical records

Medicine List

Welcome to Medicine list page. Click Add Medicines to Add medicines

ID	DRUG NAME	STOCK	ACTION
1	Paracetamol	1050	<button>Update</button> <button>Delete</button>
2	Dolo 650	2500	<button>Update</button> <button>Delete</button>
3	Amaxolin	4000	<button>Update</button> <button>Delete</button>
4	Hydro codone	500	<button>Update</button> <button>Delete</button>
5	Metformin	835	<button>Update</button> <button>Delete</button>
6	Gabapentin	380	<button>Update</button> <button>Delete</button>

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8090/api/v2/medicines`. The response body is displayed as JSON:

```
1  [
2   {
3     "id": 1,
4     "drugName": "Paracetamol",
5     "stock": "159"
6   },
7   {
8     "id": 2,
9     "drugName": "Amoxicillin",
10    "stock": "75"
11  },
12  {
13    "id": 3,
14    "drugName": "Ibuprofen",
15    "stock": "209"
16  },
17  {
18    "id": 4,
19    "drugName": "Cetirizine".
```

Admin Workflow

- Handle billing and insurance

Admin Dashboard

Welcome to Admin Dashboard. Please find the patient list below!! Click Appointments to view and update

Search

Current Patient List in the Queue

ID	NAME	AGE	FEES	URGENCY	ACTION
1	Nesamani	33	2200	No	<button>Delete</button>
2	Ekaambaram	37	1500	No	<button>Delete</button>
3	Kaipulla	38	2000	Yes	<button>Delete</button>
4	Pulikesi	24	1300	No	<button>Delete</button>

- 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

```
ail | 1 spring.application.name=Email-Service
| 2
| 3 server.port=8082
| 4
| 5 # Kafka config
| 6 spring.kafka.bootstrap-servers=localhost:9092
| 7 spring.kafka.consumer.group-id=email-group
| 8 spring.kafka.consumer.auto-offset-reset=earliest
| 9 spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer
| 10 spring.kafka.consumer.value-deserializer=org.springframework.kafka.support.serializer.JsonDeserializer
| 11 spring.kafka.consumer.properties.spring.json.trusted.packages=*
| 12
| 13 # Email config (for Gmail)
| 14 spring.mail.host=smtp.gmail.com
| 15 spring.mail.port=587
| 16 spring.mail.username=avulapraveenkumar922@gmail.com
| 17 spring.mail.password=zbjk ytah qbww pdij
| 18 spring.mail.properties.mail.smtp.auth=true
| 19 spring.mail.properties.mail.smtp.starttls.enable=true
| 20
```



Security

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

```
52 spring.cloud.gateway.server.webflux.routes[1].id=doctor-service-route
53 spring.cloud.gateway.server.webflux.routes[1].uri=lb://DOCTOR-SERVICE
54 spring.cloud.gateway.server.webflux.routes[1].predicates[0]=Path=/api/v1/doctors/**
55
56 # -----
57 # Route for APPOINTMENT -SERVICE
58 #
59 spring.cloud.gateway.server.webflux.routes[2].id=appointment-service-route
60 spring.cloud.gateway.server.webflux.routes[2].uri=lb://APPOINTMENT-SERVICE
61 spring.cloud.gateway.server.webflux.routes[2].predicates[0]=Path=/api/v1/appointments/**
62
63 # -----
64 # Route for MEDICAL -RECORD -SERVICE
65 #
66 spring.cloud.gateway.server.webflux.routes[3].id=medical-record-service-route
67 spring.cloud.gateway.server.webflux.routes[3].uri=lb://MEDICAL-RECORD-SERVICE
68 spring.cloud.gateway.server.webflux.routes[3].predicates[0]=Path=/api/v1/records/**
69
70 # -----
71 # Route for PAYMENT -SERVICE
72 #
73 spring.cloud.gateway.server.webflux.routes[4].id=payment-service-route
74 spring.cloud.gateway.server.webflux.routes[4].uri=lb://PAYMENT-SERVICE
75 spring.cloud.gateway.server.webflux.routes[4].predicates[0]=Path=/api/v1/payments/**
76
77 # Distributed Tracing Configuration
78 management.tracing.sampling.probability=1.0
79 management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans
80 # Log pattern to include traceId and spanId for Spring Boot 3.x
81 logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} [%X{traceId:-},%X{spanId:-}] - %msg
82
83 # Additional tracing configuration
84 management.tracing.enabled=true
85 management.zipkin.tracing.enabled=true
86
87 # Ensure proper service name propagation
88 management.tracing.baggage.remote-fields=service-name
89 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

