Patient Appointment Booking System - Project Planning & Architecture (Java Spring Boot)

1. 📄Project Overview

A web-based patient appointment booking system where patients can book appointments with doctors, and the system ensures time slot uniqueness, real-time scheduling, and timely notifications. The backend is built in Java Spring Boot and deployed on a completely free environment.

---

2. 💸Core Features

- Patient registration and login
- Doctor registration and schedule management
- Appointment booking (mutex locking per time slot)
- Queue system to handle booking requests
- Scheduled reminders before appointments (e.g., 5 mins)
- Cron jobs to auto-create time slots for available doctors
- Admin panel for managing users and viewing analytics

---

3. ⚙️Technologies & Tools

Backend:

- Java 17+
- Spring Boot
- Spring Data JPA (Hibernate)
- Spring Security (JWT authentication)
- Spring Scheduler (for cron jobs and reminders)

Database:

- PostgreSQL (use free tier via Railway or Supabase)

Queue System:

- RabbitMQ (via CloudAMQP - free tier) OR Redis Streams (via Upstash - free tier)

Locking:

- Redisson for distributed Redis locking (Upstash Redis free tier)

Notifications:

- JavaMailSender (for email)
- Twilio Free Tier or Firebase Cloud Messaging (for SMS/push)

Deployment (Free):

- Render / Railway / Fly.io (Free Spring Boot app hosting)
- GitHub Actions (CI/CD)
- GitHub (Repo management)

---

4. 🕰️Project Modules Breakdown

4.1 Authentication & Authorization

- Register/Login for Patients and Doctors
- Use JWT for secure authentication

4.2 Doctor Schedule Management

- Doctors define their available days and time ranges
- Stored in DoctorSchedule table

4.3 Slot Generation (Cron Job)

- Runs every night to generate appointment slots for next X days
- Respects doctor's availability stored in DB
- Uses `@Scheduled(cron = "0 0 0 * * ?")`

4.4 Appointment Booking

- Patients book available slots
- Booking goes through:
- Queue (RabbitMQ/Redis Stream)
- Mutex Locking using Redis (Redisson)
- Slot verification & reservation

4.5 Reminder Notification Scheduler

- Cron job or scheduled task checks every minute
- Sends reminder if appointment is exactly 5 minutes away
- Email or SMS via API

---

5. 📊Database Design (Simplified)

Users

id	name	email	password	role

DoctorSchedule

| id | doctor_id | day_of_week | start_time | end_time |

AppointmentSlot

| id | doctor_id | datetime | is_booked |

Appointments

| id | doctor_id | patient_id | datetime | status |

NotificationQueue (Optional)

| id | appointment_id | notify_at | status |

---

6. 🗜️ Development Roadmap

Phase 1: Setup & Auth

- Setup Spring Boot project
- Configure PostgreSQL + JPA
- Setup User (Patient/Doctor) registration/login
- JWT-based authentication

Phase 2: Doctor Schedule & Slot Generation

- Create Doctor Schedule APIs
- Implement cron job to auto-generate slots

Phase 3: Appointment Booking with Queue + Locking

- Setup Redis (Upstash)
- Add booking service with queue & distributed locking

Phase 4: Reminder System

- Implement scheduler to check and send notifications 5 mins before appointment
- Configure JavaMailSender or Twilio

Phase 5: Frontend & Admin Dashboard (Optional)

- Build basic UI using React or Thymeleaf (if keeping full stack in Java)
- Add analytics for admin panel (number of bookings, etc.)

Phase 6: Deployment

- Host backend on Render/Railway/Fly.io
- Use Railway for free PostgreSQL
- Use Upstash for Redis
- Use GitHub Actions for CI/CD

---

7. 🪮 Free Hosting Recommendations

Component	Free Provider	Notes
Spring Boot App	Railway / Render	Easy deployment with GitHub
PostgreSQL DB	Railway / Supabase	Free tier available
Redis	Upstash	Serverless Redis + Redisson compatible
RabbitMQ	CloudAMQP	Free shared plan
Email/SMS	Mailtrap / Twilio / FCM	Free dev plans

---

8. ✉️ Final Notes

- Keep environment variables secure (use Railway secrets or GitHub secrets)
- Keep logic modular and service-based for easy testing and deployment
- Use DTOs and validation for clean API inputs

Would you like sample code or starter templates next?