Java Spring Boot Machine Test

Healthcare Management System

Objective

Build a backend Healthcare Management System using Spring Boot to manage doctors, patients, appointments, and basic medical records. The goal is to evaluate practical backend skills including API design, data modelling, validation, and basic security.

Duration: 3–5 Hours

Tech Stack:

• Java 17+
• Spring Boot 3+
• Spring Data JPA / Hibernate
• H2 / MySQL (H2 for tests)
• Spring Security / JWT
• RESTful APIs
• (Optional: Swagger for API documentation)

Problem Statement

Create a backend system to manage doctors, patients, appointments and simple medical records. The system should allow patients to search for doctors, book/cancel appointments, and allow doctors to view their schedule and add simple prescriptions/notes.

Entities

Doctor Entity

Field	Type	Description
id	Long	Primary key
name	String	Doctor's full name
specialization	String	Medical specialization (e.g., Cardiologist)
email	String	Unique email for login
phone	String	Contact number
availableSlots	List<LocalDateTime>	Available times for appointments (simple model)

Patient Entity

Field	Type	Description
id	Long	Primary key
name	String	Patient's full name
email	String	Unique email for login
phone	String	Contact number
dob	LocalDate	Date of birth
medicalHistory	String	Short medical history notes (optional)

Appointment Entity

Field	Type	Description
id	Long	Primary key
doctorId	Long	FK to Doctor
patientId	Long	FK to Patient
appointmentTime	LocalDateTime	Scheduled appointment time
status	ENUM (SCHEDULED, CANCELLED, COMPLETED)	Appointment status
notes	String	Optional notes or prescription text

Features to Implement

A. User Management (Patients & Doctors)

- POST /api/auth/register/patient — Register a new patient (name, email, phone, dob, password).
- POST /api/auth/register/doctor — Register a new doctor (name, specialization, email, phone, password).
- POST /api/auth/login — Authenticate user (email + password) and return JWT.

B. Doctor Search & Availability

- GET /api/doctors?specialization={spec}&page;={}&size;={} — Search doctors by specialization with pagination.
- GET /api/doctors/{id}/availability — Get available slots for a doctor.

C. Appointment Management

- POST /api/appointments/book — Book an appointment (patientId, doctorId, appointmentTime). Validate slot availability.
- POST /api/appointments/cancel — Cancel an appointment (appointmentId) — only patient or doctor can cancel.
- GET /api/appointments/patient/{patientId} — List patient appointments.
- GET /api/appointments/doctor/{doctorId} — List doctor's schedule.

D. Medical Notes / Prescription

- POST /api/appointments/{id}/prescription — Doctor can add prescription/notes to a completed appointment.
- GET /api/medical-records/patient/{patientId} — Return appointments with notes/prescriptions.

E. Validation & Business Rules

• Prevent double-booking of the same doctor for the same slot.
• Only authenticated users can access protected endpoints.
• Only doctors can add prescriptions/notes for their appointments.
• Appointments can only be booked at future times.

F. Security

- Implement JWT-based authentication for protected endpoints.
- Store passwords hashed (BCrypt).

G. Bonus (Optional Enhancements)

• Add Swagger UI documentation.
• Add role-based access control (ROLE_DOCTOR, ROLE_PATIENT).
• Write unit & integration tests using JUnit and Mockito.
• Add email notifications (simulated) upon booking/cancellation.
• Dockerize the application.
• Implement file upload for simple reports (PDF) stored on disk.

Evaluation Criteria

Area	Weight	Description
Code Quality	25%	Clean, readable, modular code and meaningful names
API Design	20%	RESTful standards, error handling, status codes
Security	15%	JWT, password hashing, endpoint protection
Data Modeling	15%	Efficient schema and constraints
Functionality	15%	Features working as specified
Bonus Features	10%	Swagger, tests, Docker, notifications

Expected Deliverables

- Source code (GitHub link or zip)
- README.md with setup steps, API documentation (examples), and sample data
- Postman collection (optional)
- Instructions to run using Docker (if provided)

Good luck! Provide the GitHub link or zip when ready for evaluation.