Documentation task

Lab 6

Java bootcamp

November 11, 2025

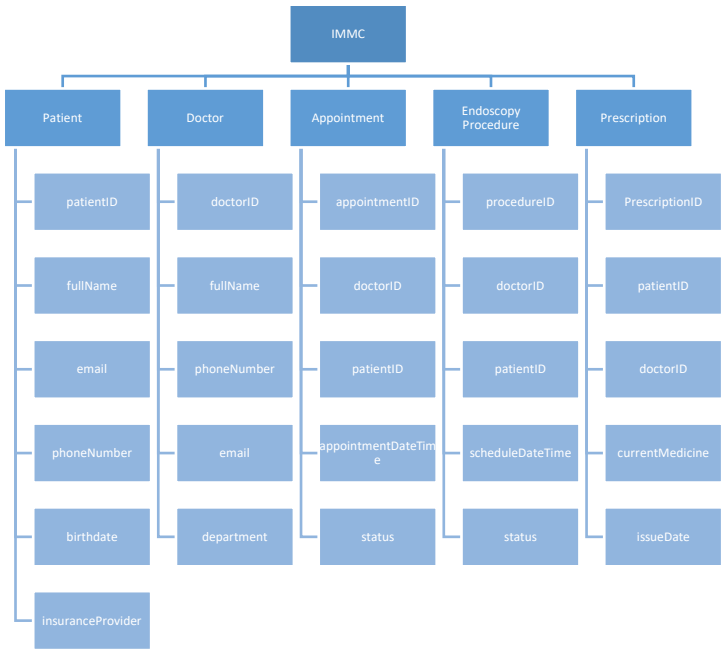
Rand Abalkhail

Table of Contents

System Title: IMCC Hospital Management System

Brief description:

This system supports IMCC's clinical operations — including patient registration, doctor scheduling, endoscopic procedures, prescriptions, and follow-up care for gastrointestinal and liver diseases. It ensures accurate data capture, safe workflows, compliance with healthcare standards, and efficient service delivery.



Model 1: Doctor

Variable 1: doctorID

- Cannot be null.
- Must be exactly 10 numeric digits.

Variable 2: fullName

- Cannot be null or empty.
- Must be only letters and spaces; minimum length of 5 characters.

Variable 3: phoneNumber

- Cannot be null or empty.
- Must match Saudi mobile format: starts “05” plus 8 more digits (10 digits total).

Variable 4: email

- Cannot be null or empty.
- Must be a valid email format.

Variable 5: department

- Rule: Cannot be null or empty.
- Rule: Must match one of IMCC’s approved departments (“Gastroenterology”, “Hepatology”, “Endoscopy”, “Internal Medicine”)

```java
public class Doctor {
    @NotEmpty(message = "Doctor ID cannot be null or empty")
    @Pattern(regexp = "^\\d{10}$", message = "Doctor ID must be exactly 10 numeric digits")
    private String doctorID;

    @NotEmpty(message = "full name cannot be null or empty")
    @Size(min = 5, message = "full name must be at least 5 characters long")
    @Pattern(regexp = "^[A-Za-z\\s]+$", message = "Full name must contain only letters and spaces")
    private String fullName;

    @NotEmpty(message = "phone number cannot be null or empty")
    @Pattern(regexp = "^05\\d{8}$", message = "Phone number must start with 05 and be 10 digits long")
    private String phoneNumber;

    @NotEmpty(message = "email cannot be null or empty")
    @Email(message = "must be a valid email format")
    private String email;

    @NotEmpty(message = "department cannot be null or empty")
    @Pattern(
            regexp = "^(Gastroenterology|Hepatology|Endoscopy|Internal Medicine)$",
            message = "Department must be one of: Gastroenterology, Hepatology, Endoscopy, or Internal Medicine"
    )
    private String department;
}
```

Model 2: Patient

Variable 1: patientID

- Rule: Cannot be null.
- Rule: Exactly 10 numeric digits.

Variable 2: fullName

- Cannot be null or empty.
- Minimum length of 3 characters, letters and spaces only.

Variable 3: phoneNumber

- Cannot be null or empty.
- Format “05XXXXXXXX” (Saudi mobile).

Variable 4: email

- If provided must be valid format. (Could be optional)

Variable 5: birthDate

- Cannot be null.
- Must be a date in the past (not today or future).

Variable 6: insuranceProvider

- If provided, must match accepted insurers list (“Bupa”, “Tawuniya”, etc (I will only put these two).

```
public class Patient {
    @NotEmpty(message = "Patient ID cannot be null or empty")
    @Pattern(regexp = "^\\d{10}$", message = "Patient ID must be exactly 10 numeric digits")
    private String patientID;

    @NotEmpty(message = "Full name cannot be null or empty")
    @Size(min = 3, message = "Full name must be at least 3 characters long")
    @Pattern(regexp = "^[A-Za-z\\s]+$", message = "Full name must contain only letters and spaces")
    private String fullName;

    @NotEmpty(message = "Phone number cannot be null or empty")
    @Pattern(regexp = "^05\\d{8}$", message = "Phone number must start with 05 and be 10 digits long")
    private String phoneNumber;

    // optional so no @NotEmpty
    @Email(message = "Must be a valid email format")
    private String email;

    @NotNull(message = "Birth date cannot be null")
    @Past(message = "Birth date must be in the past")
    private LocalDate birthDate;

    @Pattern(regexp = "^(Bupa|Tawuniya)?$",
            message = "Insurance provider must be one of: Bupa or Tawuniya")
    private String insuranceProvider;
}
```

Model 3: Appointment

Variable 1: appointmentID

- Cannot be null or empty.
- Must be 8 alphanumeric characters (it's according to system specification but I assumed this).

Variable 2: doctorID

- Cannot be null.
- Must be an existing doctorID.

Variable 3: patientID

- Cannot be null.
- Must be an existing patientID.

Variable 4: appointmentDateTime

- Cannot be null.
- Must be today or future (cannot schedule in the past).

Variable 5: status

- Cannot be null or empty.
- Must be one of defined statuses ("Scheduled", "Completed", "Cancelled").

```
@Data  no usages
@AllArgsConstructor
public class Appointment {
    @NotEmpty(message = "Appointment ID cannot be null or empty")
    @Pattern(regexp = "^[A-Za-z0-9]{8}$", message = "Appointment ID must be 8 alphanumeric characters")
    private String appointmentID;

    @NotEmpty(message = "Doctor ID cannot be null or empty")
    private String doctorID;

    @NotEmpty(message = "Patient ID cannot be null or empty")
    private String patientID;

    @NotNull(message = "Appointment date and time cannot be null")
    @FutureOrPresent(message = "Appointment date and time must be today or in the future")
    private LocalDateTime appointmentDateTime;

    @NotEmpty(message = "Status cannot be null or empty")
    @Pattern(regexp = "^(Scheduled|Completed|Cancelled)$",
            message = "Status must be one of: Scheduled, Completed, or Cancelled")
    private String status;
}
```

Model 4: EndoscopyProcedure

(Since IMCC specializes in GI/endoscopy, I included it as a specific model)

Variable 1: procedureID

- Cannot be null.
- Must be 8 alphanumeric characters (it's according to system specification but I assumed this).

Variable 2: patientID

- Cannot be null.
- Must be an existing patientID.

Variable 3: doctorID

- Cannot be null.
- Must be an existing doctorID.

Variable 4: scheduledDateTime

- Cannot be null.
- Must be today or future (cannot schedule in the past).

Variable 5: status

- Cannot be null or empty.
- Rule: Must be one of defined statuses ("Scheduled", "Completed", "Cancelled").

Variable 6: findingsSummary

I am not sure if it will be possible to add logic in the Model with the validation but if it possible I want to do this:

- If status is "Completed", findingsSummary cannot be null or empty.

```java
@Data  no usages
@AllArgsConstructor
public class EndoscopyProcedure {
    @NotEmpty(message = "Procedure ID cannot be null or empty")
    @Pattern(regexp = "^[A-Za-z0-9]{8}$", message = "Procedure ID must be 8 alphanumeric characters")
    private String procedureID;

    @NotEmpty(message = "Patient ID cannot be null or empty")
    private String patientID;

    @NotEmpty(message = "Doctor ID cannot be null or empty")
    private String doctorID;

    @NotNull(message = "Scheduled date and time cannot be null")
    @FutureOrPresent(message = "Scheduled date must be today or in the future")
    private LocalDateTime scheduledDateTime;

    @NotEmpty(message = "Status cannot be null or empty")
    @Pattern(regexp = "^(Scheduled|Completed|Cancelled)$",
            message = "Status must be one of: Scheduled, Completed, or Cancelled")
    private String status;

    private String findingsSummary;
}
```

Model 5: Prescription

Variable 1: prescriptionID

- Cannot be null.
- Must be 8 alphanumeric characters (it's according to system specification but I assumed this).

Variable 2: patientID

- Cannot be null.
- Must be existing patient.

Variable 3: doctorID

- Cannot be null.
- Must be existing doctor.

Variable 4: medicationList

(I know you said not to make lists but I couldn't help myself, it for my convenience you can ignore it)

- Cannot be null.
- Must be existing medicine.
- Consequence: Missing meds, pharmacy errors, incomplete treatment.

Variable 5: issueDate

- Cannot be null.
- Must be today or in the past (cannot be future).

```java
@Data  no usages
@AllArgsConstructor
public class Prescription {
    @NotEmpty(message = "Prescription ID cannot be null or empty")
    @Pattern(regexp = "^[A-Za-z0-9]{8}$", message = "Prescription ID must be 8 alphanumeric characters")
    private String prescriptionID;

    // I want to have a method that checks existing
    @NotEmpty(message = "Patient ID cannot be null or empty")
    private String patientID;

    // I want to have a method that checks existing
    @NotEmpty(message = "Doctor ID cannot be null or empty")
    private String doctorID;

    //@NotEmpty(message = "Medication list cannot be empty")
    //private List<Medication> medicationList;

    @NotNull(message = "Issue date cannot be null")
    @PastOrPresent(message = "Issue date cannot be in the future")
    private LocalDate issueDate;
}
```