# DB Foundations - Hackathon

# Hackathon Level-1

## Level 1: Ideation & Solution Documentation

**Use Case Title**: Hospital Appointment Management System

**Student Name**: Thara m

**Register Number**:asanm31431423ucsc036

**Institution**: Government Arts And Science College – Tirukkoilur

**Department**: Bachelor of computer science

**Date of Submission**:16.08.2025

---

### 1. Understanding the Use Case and Assumptions

**Challenges:**

- Manual appointment scheduling causes delays, errors, and double bookings.
- Patients often need to visit the hospital to book or change appointments.
- No centralized database to check doctor availability across departments.
- Difficulty generating reports for hospital administration.

**Goals:**

- Create a centralized digital system for booking and managing appointments.
- Allow patients and hospital staff to check availability in real-time.
- Prevent double bookings and ensure efficient use of doctor schedules.
- Enable quick generation of daily, weekly, or monthly reports.

## Assumptions:

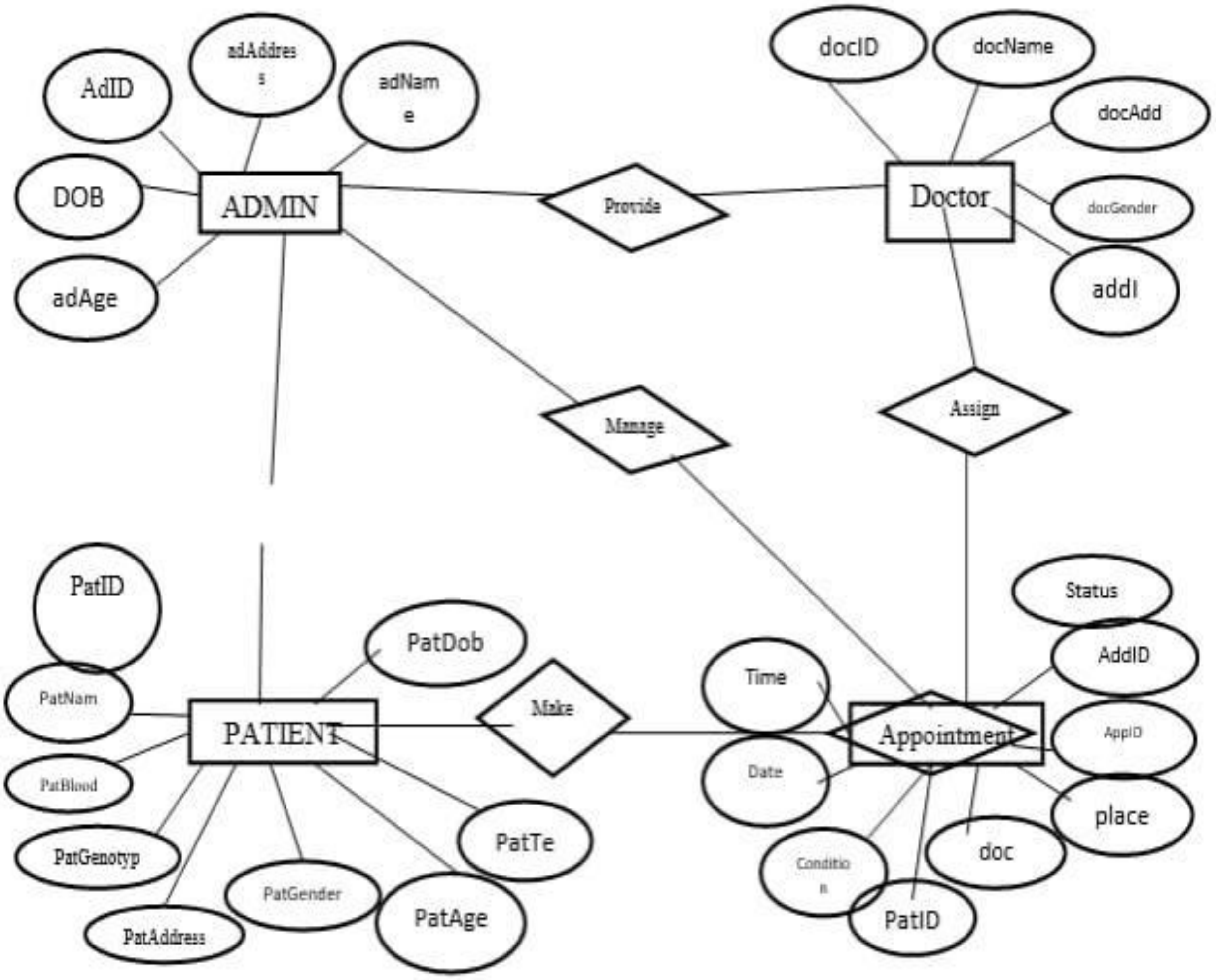
- Each doctor belongs to only one department.
- Each department operates within specific working hours.
- Appointments are booked in predefined time slots (e.g., 30 minutes).
- Patients must be registered before booking.
- Doctors' schedules are maintained and updated in the system.

## 2. Entity Identification and ER Diagram **Entities**:

- **Patient** (Patient_ID, Name, DOB, Gender, Contact)
- **Doctor** (Doctor_ID, Name, Specialization, Department_ID, Contact)
- **Department** (Department_ID, Name, Location)
- **Appointment** (Appointment_ID, Patient_ID, Doctor_ID, Date, TimeSlot_ID, Status)
- **Branch** (Branch_ID, Name, Address, Contact)
- **Time Slot** (TimeSlot_ID, StartTime, EndTime)

**Relationships**:

- A patient can book **many appointments.**
- A doctor can have **many appointments.**
- Each doctor belongs to **one department.**
- Each appointment is linked to **one time slot.**
- Departments belong to a **branch**.



## 3. Relational Schema Design

| Table Name | Column Name | Data Type | Constraints |
|---|---|---|---|
| Patient | Patient_ID | INT | PK, AUTO_INCREMENT |
| | Name | VARCHAR(100) | NOT NULL |
| | DOB | DATE | NOT NULL |
| | Gender | CHAR(1) | CHECK (Gender IN ('M','F','O')) |
| | Contact | VARCHAR(15) | UNIQUE |

| | | | |
|---|---|---|---|
| Doctor | Doctor_ID | INT | PK, AUTO_INCREMENT |
| | Name | VARCHAR(100) | NOT NULL |
| | Specialization | VARCHAR(100) | NOT NULL |
| | Department_ID | INT | FK → Department.Department_ID |
| | Contact | VARCHAR(15) | UNIQUE |
| Department | Department_ID | INT | PK, AUTO_INCREMENT |
| | Name | VARCHAR(100) | NOT NULL |
| | Location | VARCHAR(255) | NOT NULL |
| | Branch_ID | INT | FK → Branch.Branch_ID |
| Branch | Branch_ID | INT | PK, AUTO_INCREMENT |
| | Name | VARCHAR(100) | NOT NULL |
| | Address | VARCHAR(255) | NOT NULL |
| | Contact | VARCHAR(15) | UNIQUE |
| TimeSlot | TimeSlot_ID | INT | PK, AUTO_INCREMENT |
| | StartTime | TIME | NOT NULL |
| | EndTime | TIME | NOT NULL |
| Appointment | Appointment_ID | INT | PK, AUTO_INCREMENT |
| | Patient_ID | INT | FK → Patient.Patient_ID |
| | Doctor_ID | INT | FK → Doctor.Doctor_ID |
| | Date | DATE | NOT NULL |
| | TimeSlot_ID | INT | FK → TimeSlot.TimeSlot_ID |
| | Status | VARCHAR(20) | DEFAULT 'Booked' |

---

## 4. Appointment Booking Flow Logic

**1. Patient Registration** – New patients are registered with basic details.

**2. Check Availability** – The system searches for available doctors by date, department, and time slot.

**3. Conflict Prevention**:

Query checks if the doctor already has an appointment in the selected time slot.

If booked, the time slot is disabled for selection.

**4. Booking Confirmation** – Appointment is inserted into the Appointment table with status 'Booked'.

5. **Cancellation/Rescheduling** – Appointment status updated to 'Cancelled' or new slot assigned.

6. **Automatic Updates** – Doctor availability is refreshed after each booking or cancellation.

## 5. Planned SQL Operations for Level-2

- Insert new patient record.
- Insert appointment (with conflict check).
- Update appointment status (cancel/reschedule).
- Retrieve available time slots for a doctor.
- Generate report: total appointments by date/department/doctor.
- Delete old records after a retention period.

## 6. Platform Compatibility Strategy

- Use **ANSI-standard SQL** for maximum compatibility.
- For MySQL: AUTO_INCREMENT for primary keys.
- For Oracle: Use CREATE SEQUENCE + TRIGGER instead of AUTO_INCREMENT.
- Avoid MySQL-specific functions like NOW()—use CURRENT_DATE instead.
- Ensure data type compatibility:
- MySQL VARCHAR ↔ Oracle VARCHAR2
- MySQL AUTO_INCREMENT ↔ Oracle SEQUENCE

Test queries in both environments before deployment.