Empowering Health Systems: Advancing Healthcare through Online Medical Consultation

Project Team 4: Anjali Haryani, Jaamie Maarsh, Jessica James, Kalyan Kumar, Shubham Gaur

Database Purpose:

Online medical services can somewhat lower the risk of infection in this day and age. The database's principal function is to preserve the information needed to create and assist individuals receiving initial medical services online. It will primarily serve the information needs of individuals and physicians and fulfill their requests for appointment scheduling, online diagnosis, medical record administration, patient and physician information management, feedback management, and other auxiliary services.

Business Problems Addressed:

- Enable doctors and patients to initiate the medical process seamlessly without physical contact.
- Facilitate access to valuable information for doctors, patients, or administrators, such as assisting patients in selecting appropriate doctors through DoctorSchedule.
- Support the evaluation of doctors' performance, enabling service managers to fine-tune online medical services based on performance assessments.
- Provide comprehensive medical records from appointment booking to feedback, streamlining information tracking for administrators.
- Empower doctors to analyze their patients' conditions by accessing all treatment information, offering insights for effective treatment.
- Enable patients to directly order prescribed medications through the platform.

Business Rules:

1. **Appointment Scheduling:**

- Each department may have one or more doctors.
- Each doctor must belong to only one department and no less.
- Each patient may have zero or more bookings.
- Each doctor may have zero or more doctor schedules.
- Each doctor can only treat one patient at a time in a scheduled time slot.
- Each booking may have only one case associated with it.

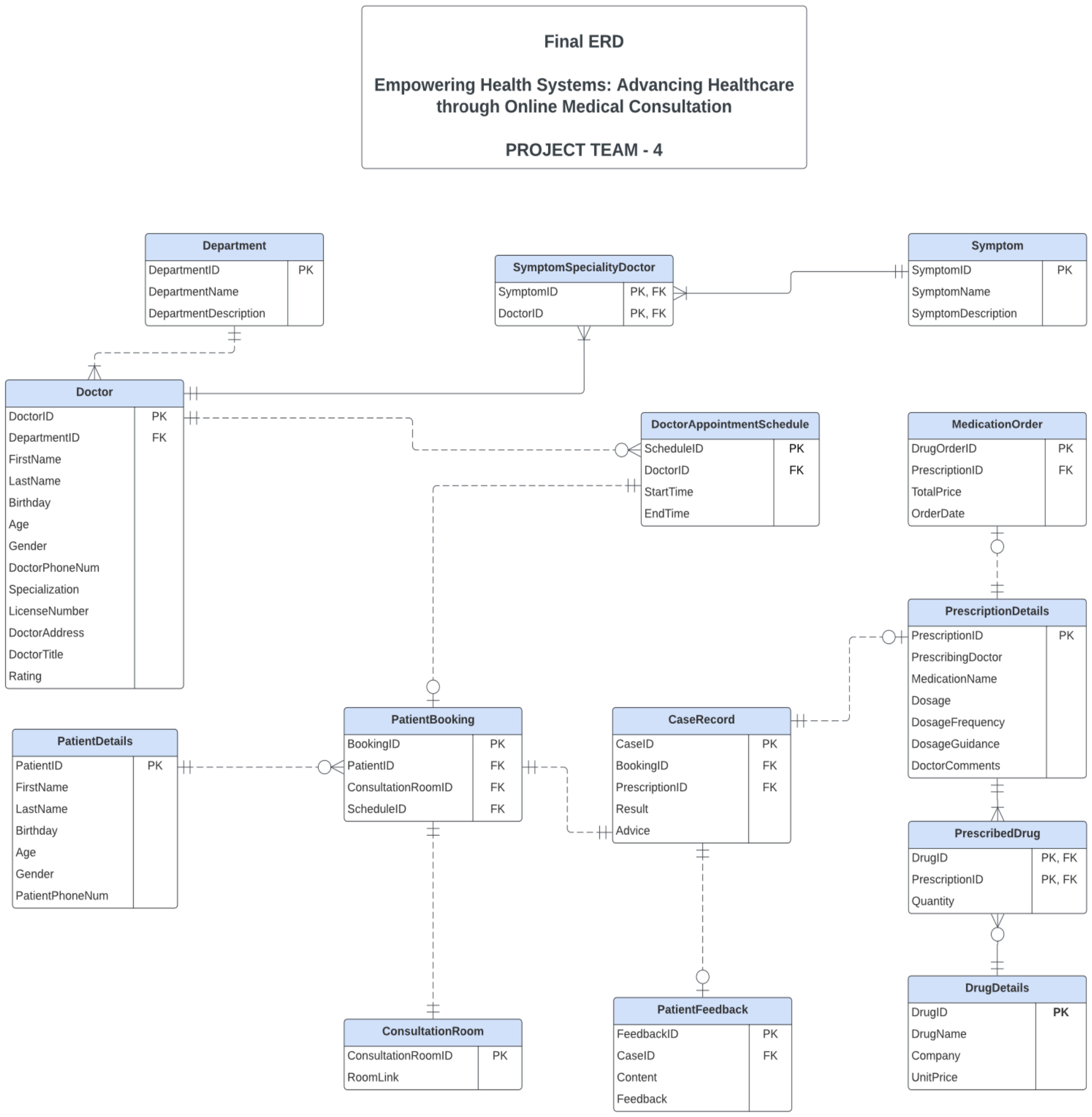
2. **Medical Record Administration & Feedback Management:**

- Each doctor can handle many symptoms.
- Each symptom can be handled by many doctors.
- Each booking may have only one consultant room associated.
- Each booking may be associated with at most one scheduled time slot.
- Each case may be associated with at most one feedback.

3. **Medication Ordering:**

- Each case may be associated with at most one electronic prescription.
- Each patient should have only one prescription under their name.
- Each electronic prescription may have one or more drugs prescribed.
- Each electronic prescription may have zero or one drug order.

Entity Relationship Diagram:



Database Design:

Entity Name	Reason for Entity	Entity relationship
Doctor:	In a medical service platform, it's necessary to store information about doctors. They should be registered within specific departments and possess titles that reflect their experience and position within the medical field.	The Doctor Entity is associated with the ‘Department’, ‘DoctorAppointmentSchedule’, and ‘SymptomSpecialityDoctor’ entities. Its primary identifier is DoctorID. Due to a many-to-many relationship, it is connected to the ‘SymptomSpecialityDoctor’ entity through an intermediary entity. It's typical for a doctor to manage multiple symptoms, and likewise, one symptom may be addressed by multiple doctors. The primary key also facilitates the association with the ‘DoctorAppointmentSchedule’ entity, allowing one doctor to have either no time slot or multiple slots. Regarding the relationship with the Department entity, the DepartmentID serves as foreign key.
Symptom:	The Symptom entity holds significant importance for both patients and doctors. Patients typically rely on symptoms to find the appropriate medical services, while doctors specialize in treating specific symptoms. Effectively organizing and storing symptom information facilitates patients in quickly locating a suitable doctor tailored to their needs.	The Symptom Entity connects to the ‘Doctor’ entity via an intermediary entity called ‘SymptomSpecialityDoctor’, reflecting many-to-many relationships. SymptomID serves as the primary key, linking this entity to the ‘Doctor’ entity. Each symptom may be addressed by multiple doctors, and conversely, each doctor may specialize in treating multiple symptoms.
SymptomSpecialityDoctor:	DoctorSymptomSpecialty functions as an intermediary entity, storing symptoms that doctors are capable of managing, thereby establishing a connection between the Symptom Entity and the Doctor Entity.	The associative entity (SymptomSpecialtyDoctor) establishes connections with both the ‘Symptom’ entity and the ‘Doctor’ entity, effectively addressing the many-to-many relationship between them. Primary keys and foreign keys, namely SymptomID and DoctorID, are utilized to uphold these relationships.
Department:	The Department entity serves as a foundational component within a medical service platform, capturing	The Department Entity maintains a relationship with the Doctor Entity, utilizing ‘DepartmentID’ as its

	essential information about departments and categorizing doctors into distinct groups. This classification enhances the hospital's efficiency in managing doctors, while also enabling patients to easily locate the appropriate doctor based on departmental affiliation.	primary key to establish connections with doctors. Each department encompasses one or more doctors, and each doctor is assigned to a specific department.
PatientDetails:	Patients are integral entities within a medical service platform. Storing patient information like name, gender, address, phone number etc. in ‘PatientDetails’ Entity	The ‘PatientDetails’ Entity has associations with the ‘PatientBooking’ Entity. Its primary key, PatientID, facilitates its connection with the ‘PatientBooking’ Entity, where a patient can have zero to multiple bookings.
DoctorAppointmentSchedule:	When a patient seeks to schedule an appointment, ensuring the availability of both the patient and the doctor is paramount. Utilizing the doctor's schedule aids the system in determining the doctor's availability status.	The ‘DoctorAppointmentSchedule’ entity maintains relationships with both the ‘Doctor’ entity and the ‘PatientBooking’ entity, linking to the ‘Doctor’ entity through its foreign key, DoctorID. Its primary key, ScheduleID, uniquely identifies each schedule. This ScheduleID is Foreign key in ‘PatientBooking’ Entity. A doctor can have zero or multiple schedules.
PatientBooking:	Booking marks the initiation of preliminary medical services online, facilitating appointments between patients and doctors.	The ‘PatientBooking’ entity establishes connections with the ‘DoctorAppointmentSchedule’, the ‘CaseRecord’, the ‘PatientDetails’, and the ‘ConsultationRoom’ entities. It utilizes the primary key BookingID to relate to the ‘CaseRecord’ entity and the foreign key ‘ScheduleID’,’PatientID’, ‘ConsultationRoomID’ to relate to the ‘DoctorAppointmentSchedule’ entity, ‘PatientDetails’, and the ‘ConsultationRoom’ entities respectively. Patients may have zero or multiple bookings.
ConsultationRoom:	The Consultation Room serves as the virtual space where patients and doctors interact, akin to a Zoom room. Consequently, both the doctor and the patient can access this room via the provided link at the scheduled start time.	The ‘ConsultationRoom’ Entity is linked to the ‘PatientBooking’ Entity through its primary key, ConsultationRoomID. Each booking is associated with only one consultation room.

CaseRecord:	The Case document holds significant importance within the online medical service, documenting the patient's condition and the doctor's recommendations. Additionally, doctors provide electronic prescriptions to aid in the patient's recovery. A CaseID uniquely identifies each case for record-keeping purposes. Upon attending a consultation, patients are assigned a case by the attending doctor.	The “CaseRecord" is uniquely linked to a single case through its primary key, prescriptionID. However, not all cases are accompanied by an electronic prescription, as doctors prescribe medication based on the severity of the patient's condition. In cases of mild conditions, prescriptions may not be necessary. Regarding the relationship with the ‘PatientBooking’ Entity, there exists a one-to-one relationship: after a patient completes an online treatment, a unique case is created and directly linked to the ‘PatientBooking’ Entity through the foreign key BookingID. Feedback is associated with individual cases through the foreign key CaseID, yet not every case receives feedback. Reasons for this variance may include instances where patients forget to provide feedback or choose not to do so.
PatientFeedback:	Feedback serves as the final stage in online medical services, offering insights into a doctor's popularity and patient satisfaction levels. However, it's important to note that feedback is optional, and the medical service process can be considered complete without it.	The “PatientFeedback” entity is Zero or at most one case is linked using the primary key CaseID. Various reasons may contribute to this, such as patients forgetting to provide feedback or being unwilling to do so. However, once feedback is submitted, it must be associated with a specific case.
PrescriptionDetails:	The PrescriptionDetails stems from the doctor's diagnosis. Its creation aims to retain comprehensive information regarding each patient's case, encompassing PrescriptionID and the doctor's recommendations. This facilitates seamless querying and updating of Electronic Prescription details during patient case processing.	The “PrescriptionDetails Entity” is linked to the PrescribedDrug Entity via its primary key, PrescriptionID. Each tuple in the PrescriptionDetails Entity corresponds to multiple tuples in the PrescribedDrug Entity. Additionally, all tuples in the PrescriptionDetails Entity must correspond to at least one tuple in the PrescribedDrug Entity (mandatory). Furthermore, the PrescriptionDetails Entity's primary key, PrescriptionID, establishes a

		relationship with the MedicationOrderID Entity. Each tuple in the PrescriptionDetails Entity corresponds to exactly one tuple in the MedicationOrderID Entity. However, it's important to note that not all tuples in the PrescriptionDetails Entity are necessarily associated with a tuple in the MedicationOrderID Entity (optional).
DrugDetails:	Effective DrugDetails management constitutes a crucial aspect of online medical services. The creation of the DrugDetails Entity aims to preserve essential drug information, encompassing drugID, drug name, drug company, and drug price. This facilitates seamless querying and updating of drug information during drug shipments and prescription processing.	The “DrugDetails” Entity is linked to the PrescribedDrug Entity via its primary key, DrugID. Each tuple in the DrugDetails Entity corresponds to multiple tuples in the PrescribedDrug Entity. However, it's important to note that not all tuples in the DrugDetails Entity are necessarily associated with tuples in the PrescribedDrug Entity.
PrescribedDrug:	The PrescribedDrug entity serves as an intermediary between the Drug Details entity and the PrescriptionDetails entity, addressing the many-to-many relationship between them. Specifically, the PrescribedDrug entity is responsible for preserving detailed prescription information for each drug within the electronic prescription, which includes DrugID, PrescriptionID, and DrugQuantity.	The foreign key of the “PrescribedDrug” Entity, DrugID, establishes a relationship with the Drug Details Entity. Each tuple in the PrescribedDrug Entity corresponds to exactly one tuple in the Drug Details Entity, with this association being mandatory. Similarly, the foreign key PrescriptionID in the PrescribedDrug Entity links it to the PrescriptionDetails Entity. Each tuple in the PrescribedDrug Entity is linked to precisely one tuple in the PrescriptionDetails Entity, with this connection being mandatory.
MedicationOrderID:	The aim of establishing the MedicationOrderID entity is to manage medication order details, encompassing DrugOrderID, PrescriptionID, TotalPrice, OrderDate, and tracking number. This facilitates seamless querying, updating, and tracking of drug order information and shipment statuses.	The “MedicationOrderID” Entity's foreign key, PrescriptionID, establishes a relationship with the PrescriptionDetails Entity. Each tuple in the MedicationOrderID Entity corresponds to exactly one tuple in the PrescriptionDetails Entity. Furthermore, all tuples in the MedicationOrderID Entity are required to correspond to one tuple in the PrescriptionDetails Entity (mandatory).