Problem Statement

Developing a hospital management system in order to effectively manage most aspects of hospitals such as booking appointments, managing patient records and keeping medical history.

Overview

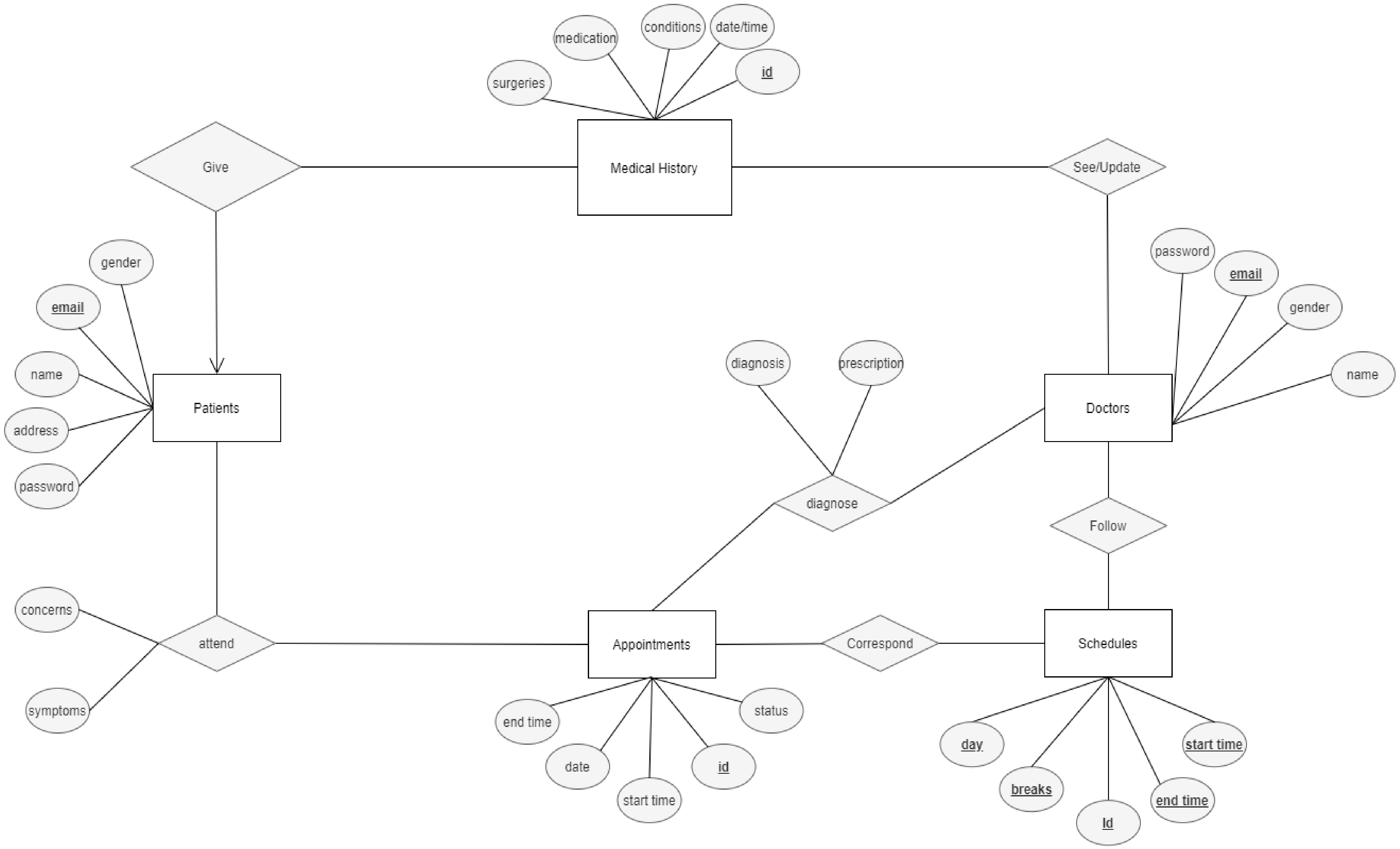
Organisations such as hospitals have to deal with a lot of patients regularly and hence a lot of data. Hence it is very important for a hospital to have a DBMS with a frontend that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

For this project I have chosen to build the frontend using ReactJS and JavaScript, backend in Node.js and the database used will be MySQL.

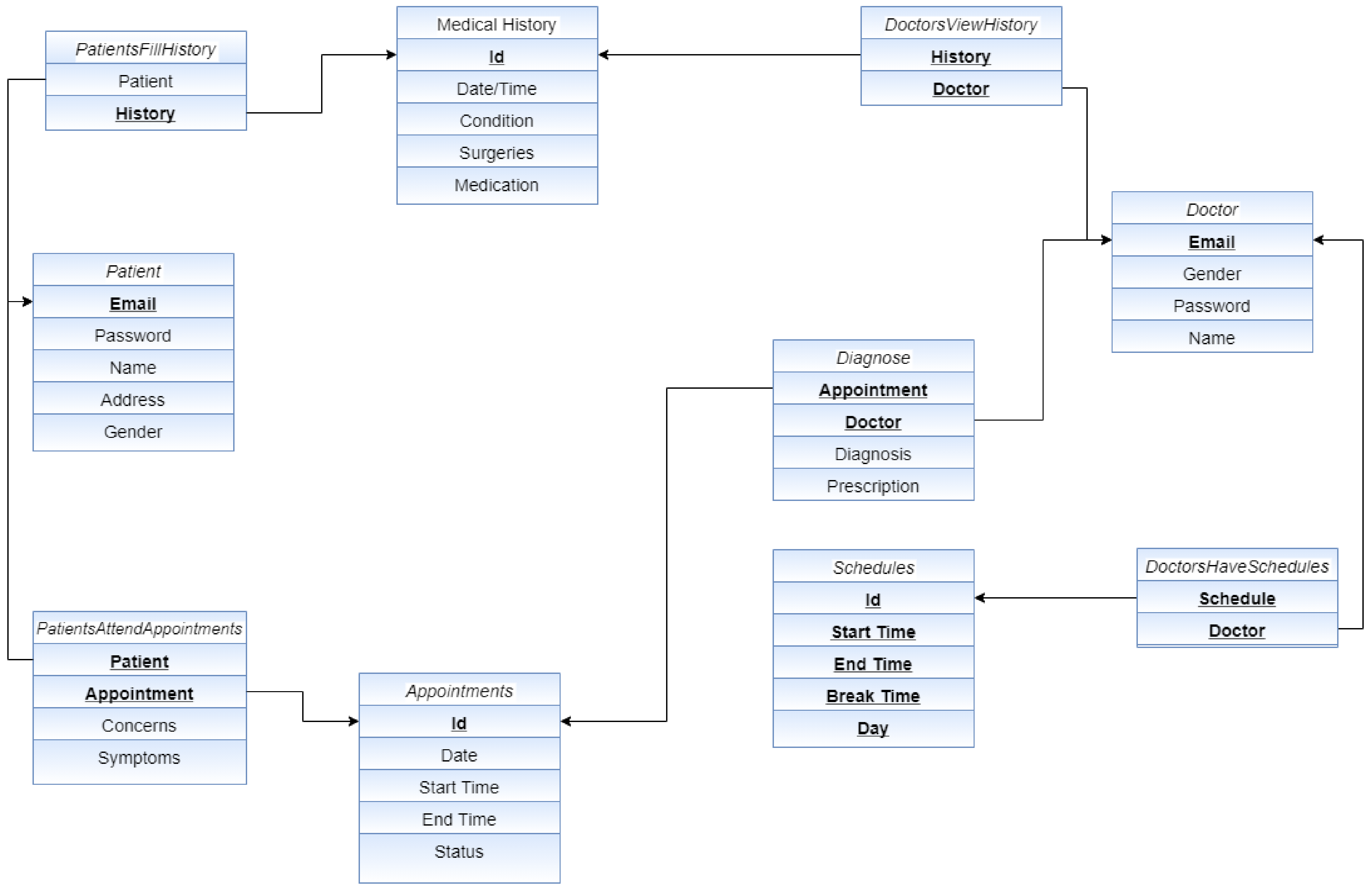
Functional Requirements

1. Separate interfaces for patients and doctors. Patients and doctors should have separate logins.
2. Allow patients to book appointments and give previous medical history.
3. Allow patients to view/update/cancel already booked appointments if necessary.
4. Allow doctors to cancel appointments.
5. Cancelled appointments should create free slots for other patients.
6. The system should avoid clash of appointments.
7. The system should take into consideration hospital and doctor schedules and allow appointments only when a doctor is not already busy or does not have a break.
8. Doctors should be able access patient history and profile, and add to patient history.
9. Doctors should be able to give diagnosis and prescriptions.
10. Patients should be able to see complete diagnosis, prescriptions and medical history.

Revised ER Diagram



Normalized Relational Schemas



Functional Dependencies and Normalisation

1. **Patient** :
   R = (**Email**, Password, Name, Address, Gender)
   FDs:
   a. Email -> Password
   b. Email -> Name
   c. Email -> Address
   d. Email -> Gender

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

2. **Medical History** :
   R = (**id**, Date, Conditions, Surgeries, Medication)
   FDs:
   a. id -> Password
   b. id -> Date
   c. id -> Conditions
   d. id -> Surgeries
   e. id -> Medication

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

3. **Doctor** :
   R = (**<u>email</u>**, gender, password, name)
   FDs:
   a. email -> gender
   b. email -> password
   c. email -> name

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

4. **Appointment:**
   R = (**<u>id</u>**, date, start time, end time, status)
   FDs:
   a. id -> date
   b. id -> start time
   c. id -> end time
   d. id -> status

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

5. **PatientsAttendAppointments:**
   R = (**<u>patient, appointment</u>**, concerns, symptoms)
   FDs:
   a. (patient, appointment) -> concerns
   b. (patient, appointment) -> symptoms

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

6. **Schedule:**
   R = (**<u>id, start time, end time, break time, day</u>**)

   Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.

   Hence it is in 3NF.

7. **PatientsFillHistory:**
   R = (Patient, **<u>History</u>**)
   FDs:
   a. History -> Patient

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

8. **Diagnose:**

   R = (**<u>appointment, doctor,</u>diagnosis, prescription**)

   FDs:

   a. (appointment, doctor) -> diagnosis

   b. (appointment, doctor) -> prescription

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

9. **DoctorsHaveSchedules:**

   R = (**<u>Schedule, Doctor</u>**)

   Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.

10. **DoctorViewsHistory:**

    R = (**<u>history, doctor</u>**)

    Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.