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

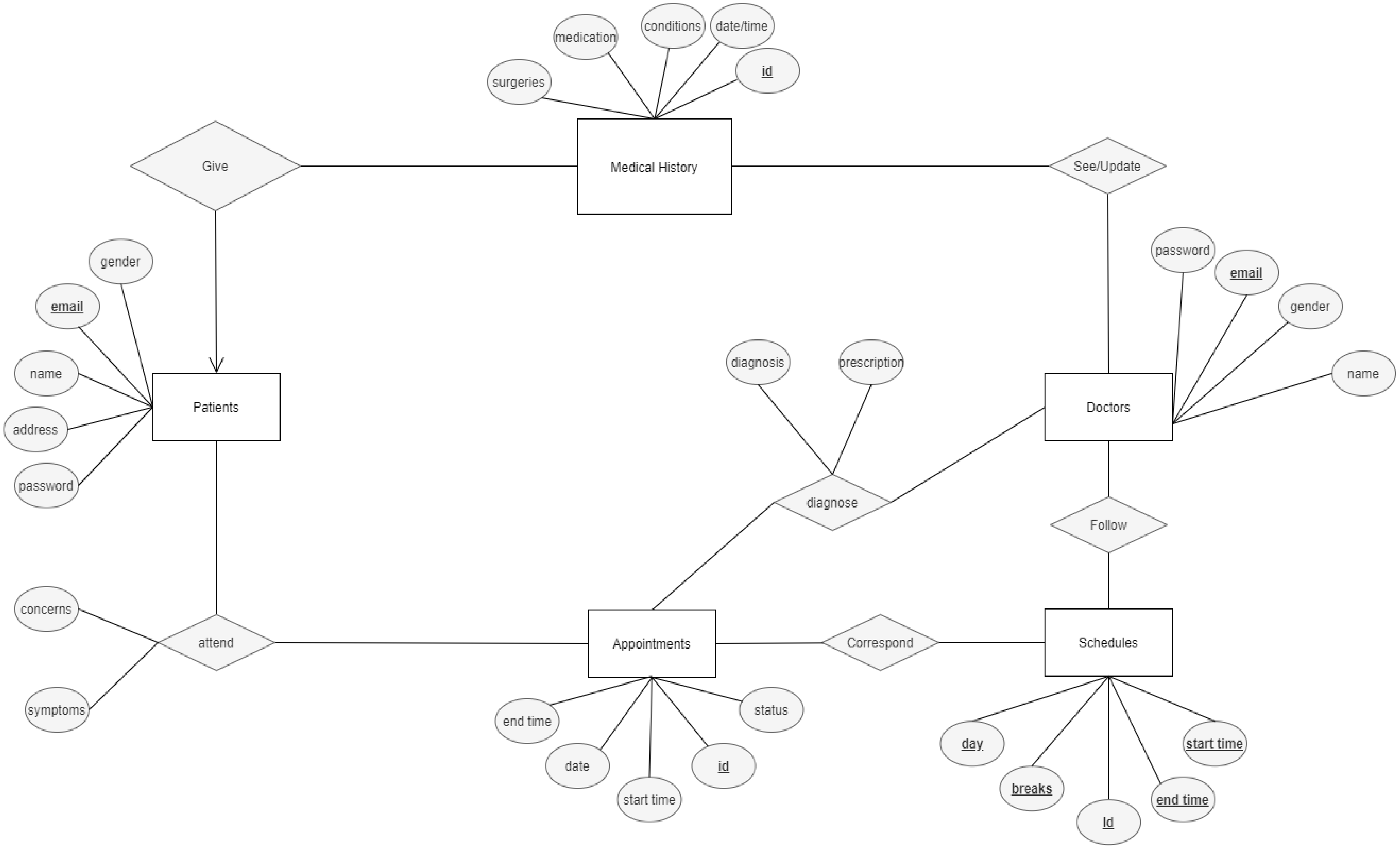
Organisations such as hospitals have to deal with a lot of patients regularly and hence a lot of data. Hence it is very important for a hospital to have a DBMS with a frontend that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

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.

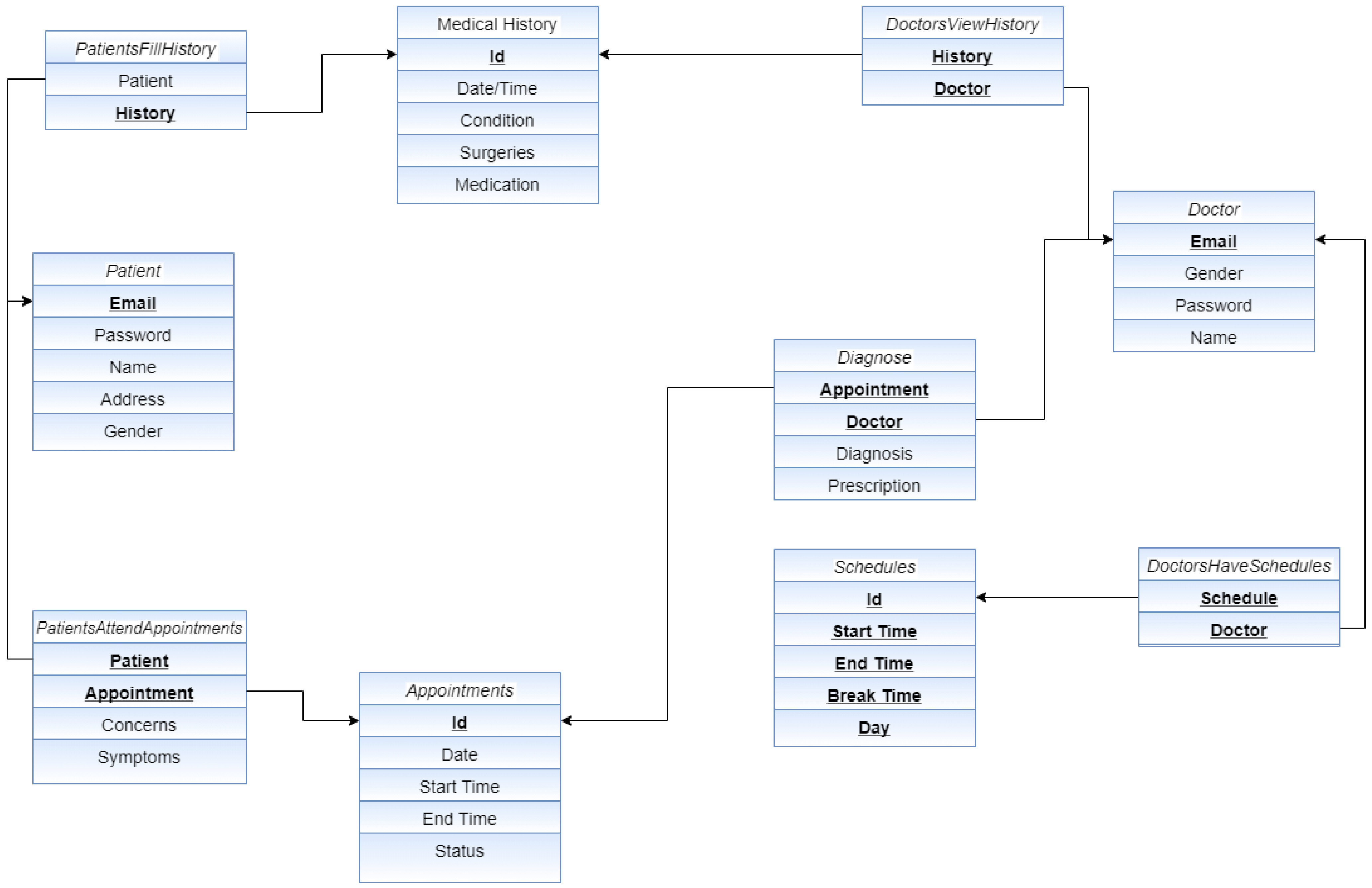
Functional Requirements

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.

<u>SQL CODE</u>

<u>DDL</u>

```
CREATE DATABASE HMS;
USE HMS;

CREATE TABLE Patient(
email varchar(50) PRIMARY KEY,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL,
address varchar(60) NOT NULL,
gender VARCHAR(20) NOT NULL
);

CREATE TABLE MedicalHistory(
id int PRIMARY KEY,
date DATE NOT NULL,
conditions VARCHAR(100) NOT NULL,
surgeries VARCHAR(100) NOT NULL,
medication VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Doctor(
email varchar(50) PRIMARY KEY,
gender varchar(20) NOT NULL,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL
);

CREATE TABLE Appointment(
id int PRIMARY KEY,
date DATE NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
status varchar(15) NOT NULL
);

CREATE TABLE PatientsAttendAppointments(
patient varchar(50) NOT NULL,
appt int NOT NULL,
concerns varchar(40) NOT NULL,
symptoms varchar(40) NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
PRIMARY KEY (patient, appt)
);

CREATE TABLE Schedule(
id int NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
breaktime TIME NOT NULL,
day varchar(20) NOT NULL,
PRIMARY KEY (id, starttime, endtime, breaktime, day)
);

CREATE TABLE PatientsFillHistory(
patient varchar(50) NOT NULL,
history int NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE CASCADE,
PRIMARY KEY (history)
);

CREATE TABLE Diagnose(
appt int NOT NULL,
doctor varchar(50) NOT NULL,
diagnosis varchar(40) NOT NULL,
prescription varchar(50) NOT NULL,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (appt, doctor)
);

CREATE TABLE DocsHaveSchedules(
sched int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (sched) REFERENCES Schedule (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (sched, doctor)
);

CREATE TABLE DoctorViewsHistory(
history int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE CASCADE,
PRIMARY KEY (history, doctor)
);
```

DML

```
INSERT INTO Patient(email,password,name,address,gender)
VALUES
('ramesh@gmail.com','hrishikesh13','Ramesh','Tamil Nadu', 'male'),
('suresh@gmail.com','hrishikesh13','Suresh','Karnataka', 'male'),
('rakesh@gmail.com','hrishikesh13','Rakesh','Gujarat', 'male')
;
```

```
INSERT INTO MedicalHistory(id,date,conditions,surgeries,medication)
VALUES
(1,'19-01-14','Pain in abdomen','Heart Surgery','Crocin'),
(2,'19-01-14','Frequent Indigestion','none','none'),
(3,'19-01-14','Body Pain','none','Iodex')
;

INSERT INTO Doctor(email, gender, password, name)
VALUES
('hathalye7@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye'),
('hathalye8@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye')
;

INSERT INTO Appointment(id,date,starttime,endtime,status)
VALUES
(1, '19-01-15', '09:00', '10:00', 'Done'),
(2, '19-01-16', '10:00', '11:00', 'Done'),
(3, '19-01-18', '14:00', '15:00', 'Done')
```

```
;

INSERT INTO PatientsAttendAppointments(patient,appt,concerns,symptoms)
VALUES
('ramesh@gmail.com',1, 'none', 'itchy throat'),
('suresh@gmail.com',2, 'infection', 'fever'),
('rakesh@gmail.com',3, 'nausea', 'fever')
;

INSERT INTO Schedule(id,starttime,endtime,breaktime,day)
VALUES
(001,'09:00','17:00','12:00','Tuesday'),
(001,'09:00','17:00','12:00','Friday'),
(001,'09:00','17:00','12:00','Saturday'),
(001,'09:00','17:00','12:00','Sunday'),
(002,'09:00','17:00','12:00','Wednesday'),
(002,'09:00','17:00','12:00','Friday')
;
```

```
INSERT INTO PatientsFillHistory(patient,history)
VALUES
('ramesh@gmail.com', 1),
('suresh@gmail.com', 2),
('rakesh@gmail.com', 3)
;

INSERT INTO Diagnose(appt,doctor,diagnosis,prescription)
VALUES
(1,'hathalye7@gmail.com', 'Bloating', 'Ibuprofen as needed'),
(2,'hathalye8@gmail.com', 'Muscle soreness', 'Stretch morning/night'),
(3,'hathalye8@gmail.com', 'Vitamin Deficiency', 'Good Diet')
;

INSERT INTO DocsHaveSchedules(sched,doctor)
VALUES
(001,'hathalye7@gmail.com'),
(002,'hathalye8@gmail.com')
;
```

```
INSERT INTO DoctorViewsHistory(history,doctor)
VALUES
(1,'hathalye7@gmail.com'),
(2,'hathalye8@gmail.com'),
(3,'hathalye8@gmail.com')
;
```

Patient Side Features :

1. There is a seperate interface for patients. Patients have a seperate login.
2. Patients can book appointments.
3. Patients can give previous medical history
4. Patients can view/update/cancel already booked appointments if necessary.
5. Cancelled appointments create free slots for other patients.
6. The system avoids clash of appointments with other patients. Each patient is therefore ensured his/her slot.
7. Patients are able to see complete diagnosis, prescriptions and medical history.
8. Patient medical history is only available to the doctor with whom the appointment is booked to ensure privacy.

Doctor Side Features :

1. There is a seperate interface for doctors. Doctors have a seperate login.

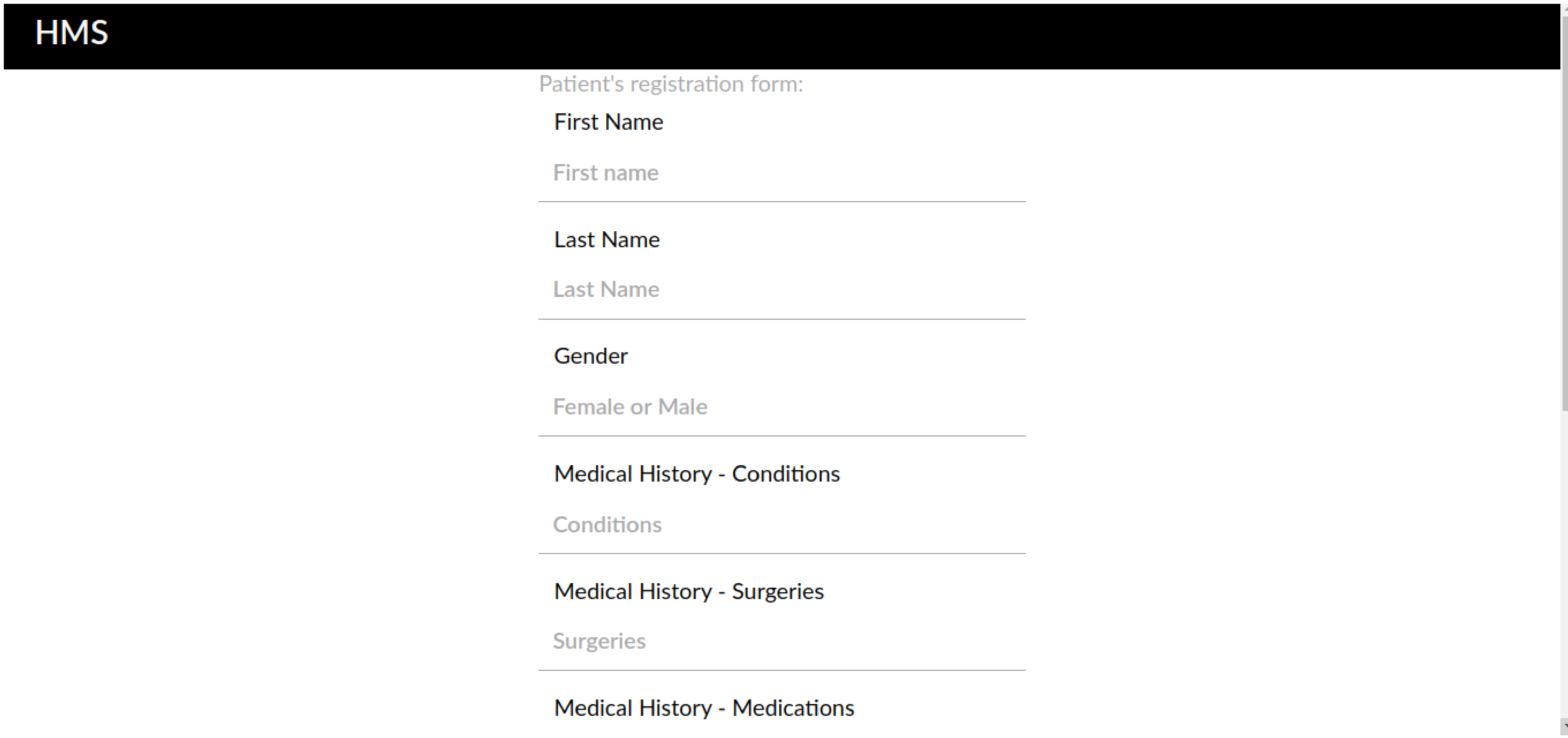
2. The system takes into consideration doctor schedules and does not allow appointments when a doctor is already busy or has a break.

3. Doctors are able to access patient history and profile, and add to patient history.

4. Doctors are able to give diagnosis and prescriptions.

5. Doctors are able to modify diagnosis and prescriptions.

Web Application:



HMS

Patient's registration form:

First Name

First name

Last Name

Last Name

Gender

Female or Male

Medical History - Conditions

Conditions

Medical History - Surgeries

Surgeries

Medical History - Medications

Conclusion

We have built a web application for hospital management including the login page and a functional database successfully.