DBMS Project

UCS 310

17th May 2022

Hospital Management System

Prakhar Pathak 102016007

Mirang Puri 102016008

Dev Maheshwari 102016031

2CS9

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

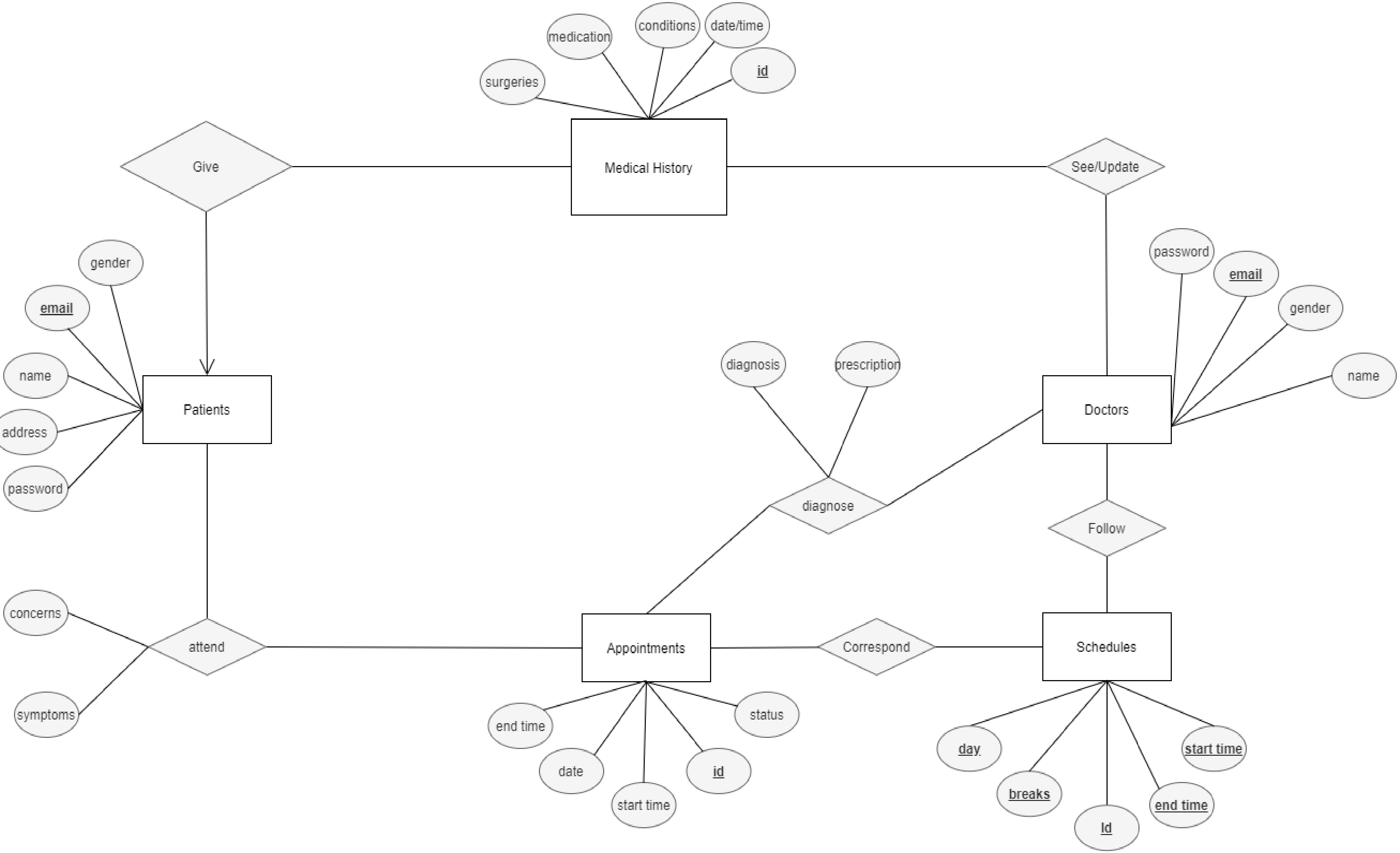
Organisations such as hospitals have to deal with a lot of patients regularly and hence a lot of data. Hence it is very important for a hospital to have a DBMS with a frontend that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

For this project we have chosen to build the database used will be MySQL and PLSQL.

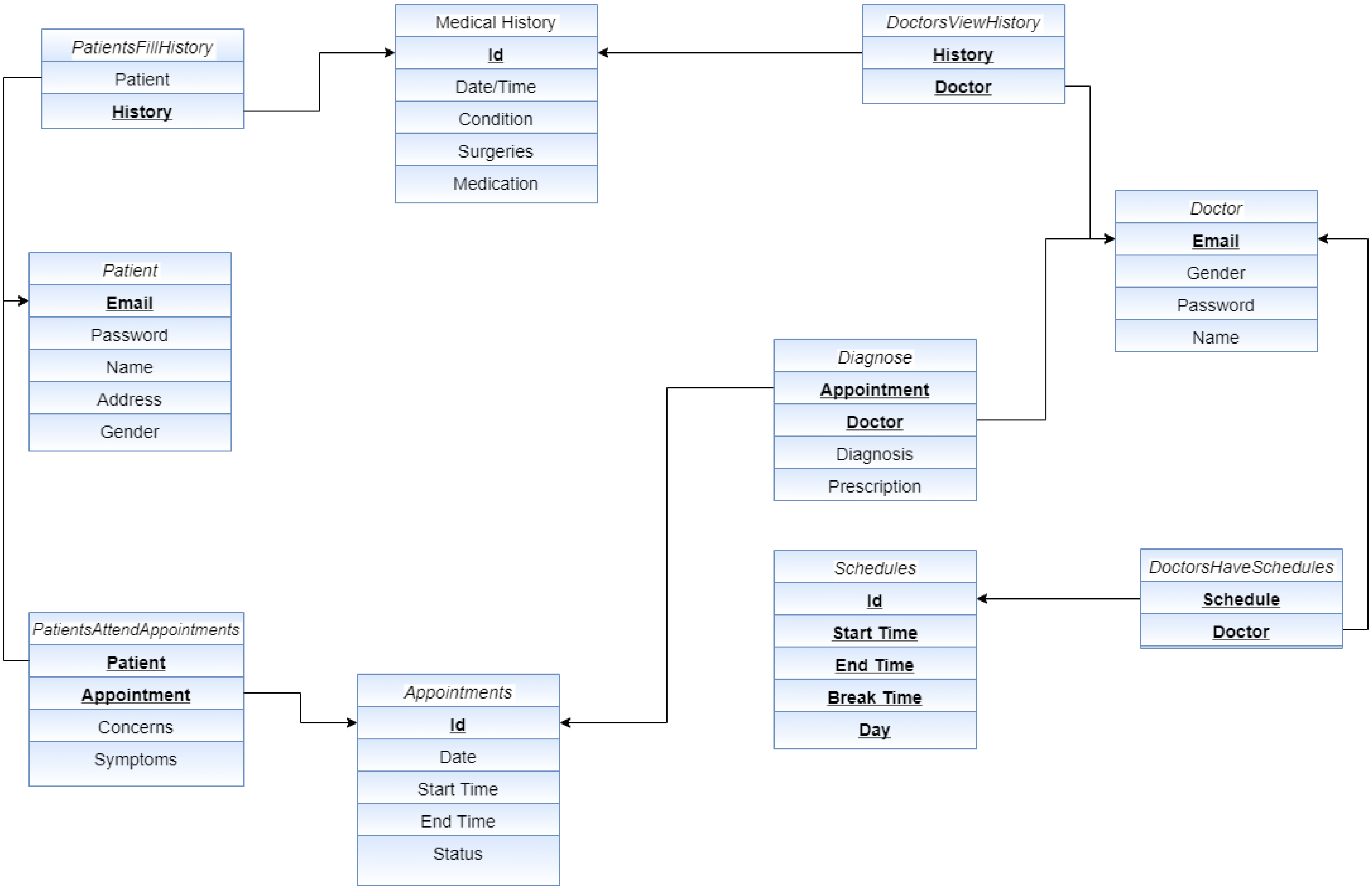
Functional Requirements

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.

ER Diagram


medication
conditions
date/time
surgeries
id
Give
Medical History
See/Update
password
email
gender
gender
email
name
diagnosis
prescription
name
Patients
Doctors
address
password
diagnose
Follow
concerns
attend
Appointments
Correspond
Schedules
symptoms
end time
status
date
id
start time
day
start time
breaks
end time
Id

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 = (**patient, appointment**, 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 = (**id, start time, end time, break time, day**)

   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, **History**)
   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 = (**appointment, doctor,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 = (**Schedule, Doctor**)

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 = (**history, doctor**)

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

SQL/PLSQL Code :

Creating Tables :

1. **Creating table Patient**

```sql
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

);
```

2. Creating table MedicalHistory

```
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

);
```

3. Creating Table Doctor

```
CREATE TABLE Doctor(

email varchar(50) PRIMARY KEY,

gender varchar(20) NOT NULL,

password varchar(30) NOT NULL,

name varchar(50) NOT NULL

);
```

4. Creating Table Appointment

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

5. Creating Table PatientsAttendAppointments

```sql
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)
);
```

6. Creating Table Schedule

```sql
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)
);
```

7. Creating Patient Fill History

```sql
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)
);
```

8. Creating Diagnose

```sql
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)

);
```

9. Creating Docs Schedule

```sql
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)

);
```

10. Creating Doctor History

```sql
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)

);
```

Inserting into Tables :

1. Inserting the values into the Patient

```sql
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')
```

```
;
```

12. Inserting the values into the Medical History

```
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')

;
```

13. Inserting the values into the Doctor

```
INSERT INTO Doctor(email, gender, password, name)

VALUES

('hathalye7@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye'),

('hathalye8@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye')

;
```

4. Inserting the values into the appointment table

```
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')

;
```

5. Inserting the values into the patients at appointment table

```
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')

;
```

6. Inserting the values into the schedule table

```
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')

;
```

7.Inserting the values into patients fill history

```
INSERT INTO PatientsFillHistory(patient,history)

VALUES

('ramesh@gmail.com', 1),

('suresh@gmail.com', 2),

('rakesh@gmail.com', 3)

;
```

8. Insert into diagnose doctor

```
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')

;
```

9.Inserting into docs have schedules table

```
INSERT INTO DocsHaveSchedules(sched,doctor)

VALUES

(001,'hathalye7@gmail.com'),

(002,'hathalye8@gmail.com')

;
```

10. Insert into doctor view history table

```
INSERT INTO DoctorViewsHistory(history,doctor)

VALUES

(1,'hathalye7@gmail.com'),

(2,'hathalye8@gmail.com'),

(3,'hathalye8@gmail.com')

;
```

PLSQL Code :

PROCEDURES :

Procedure to select all Patients :

```
sqlite> CREATE PROCEDURE SelectAllPatients @user_type varchar(1)

...> AS

 ...> SELECT * FROM Patients

 ...> GO;
```

Procedureto select all Appointments :

```
sqlite> CREATE PROCEDURE SelectAppointment @class_type varchar(1), @day

smallint

...> AS

...> SELECT * FROM Schedule WHERE id = @id AND day

= @day

...> GO;
```

Triggers :

Creating new table audit

```
CREATE TABLE audit (
post_id int NOT NULL,
entry_text text NOT NULL
);
```

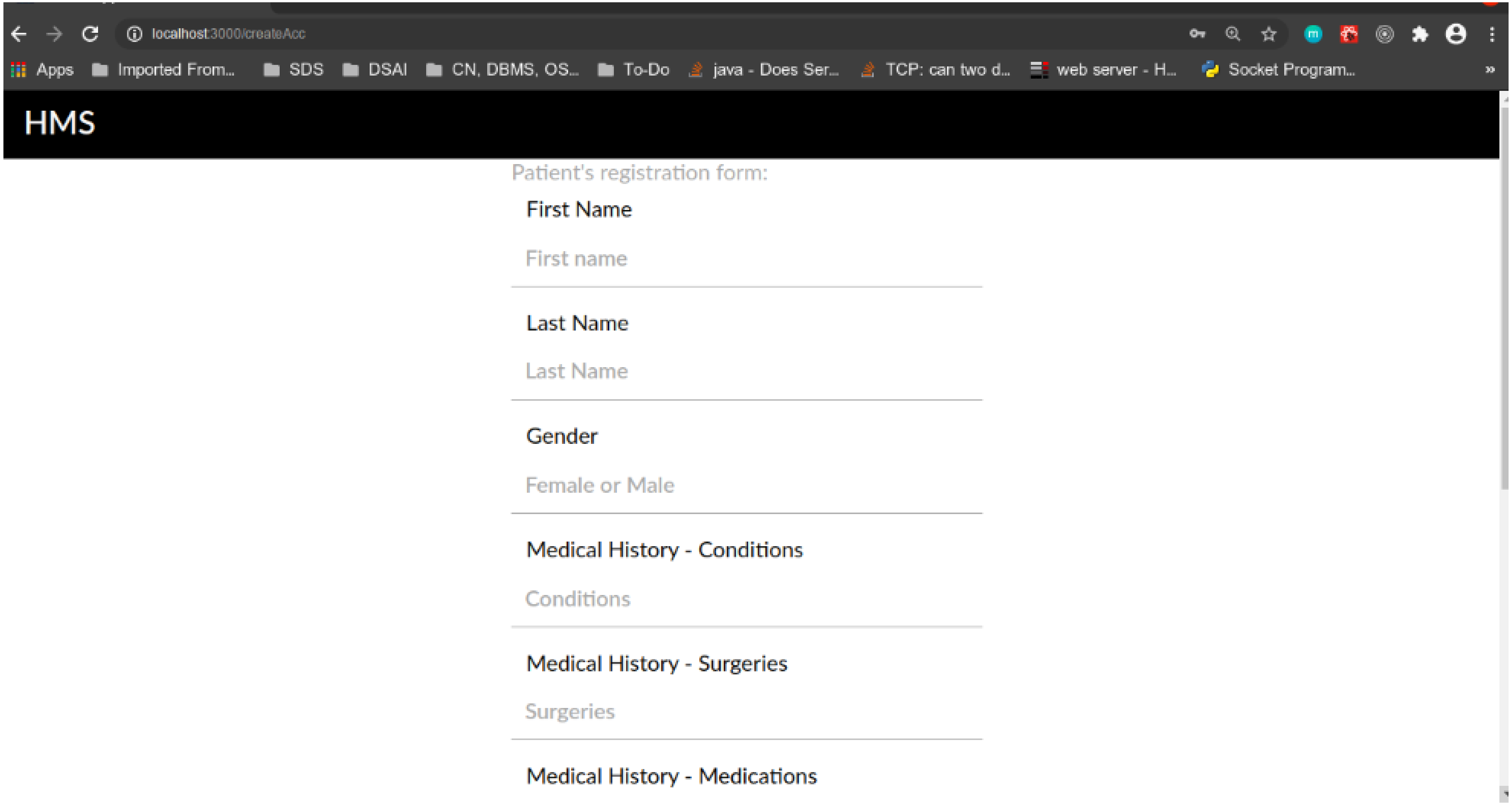
Creating Trigger

```
sqlite> CREATE OR REPLACE TRIGGER audit_log
AFTER INSERT OR DELETE OR UPDATE
...> ON forum_post
...> BEGIN
...> INSERT INTO audit(post_id, entry_date) VALUES (new.ID,
datetime('now'));
...> END;
```

Trigger being called when Creating a new post

```
sqlite> INSERT INTO forum_post VALUES (
...> 1, "Can someone help me solve this?", "", "20120618 10:34:09 AM",
1, 10
...> );
sqlite> SELECT * FROM audit;
1|2022-05-16 19:10:21
```

Screenshots of the Working Project :



HMS

Patient's registration form:

First Name

First name

Last Name

Last Name

Gender

Female or Male

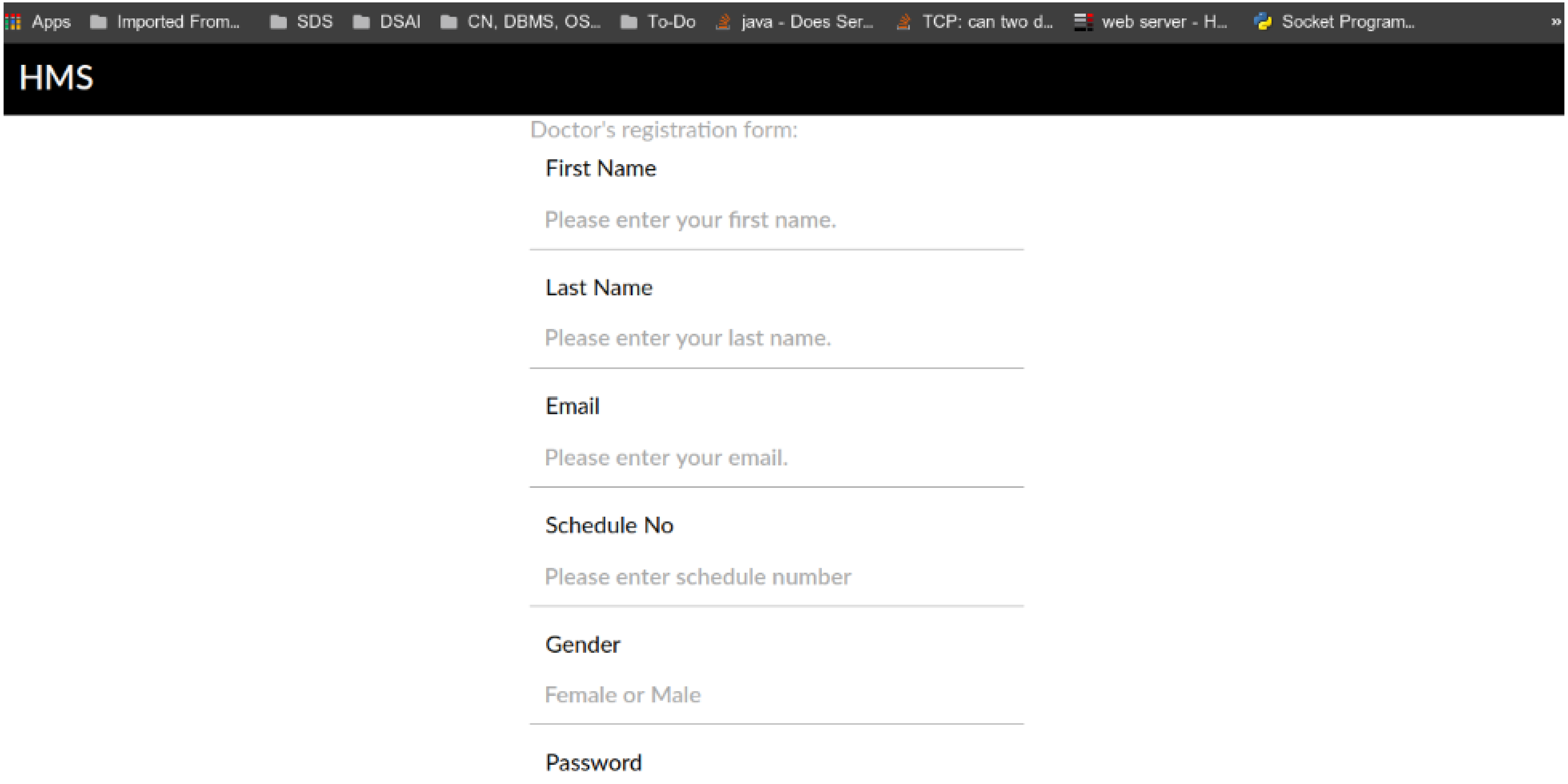
Medical History - Conditions

Conditions

Medical History - Surgeries

Surgeries

Medical History - Medications



Doctor's registration form:

First Name

Please enter your first name.

Last Name

Please enter your last name.

Email

Please enter your email.

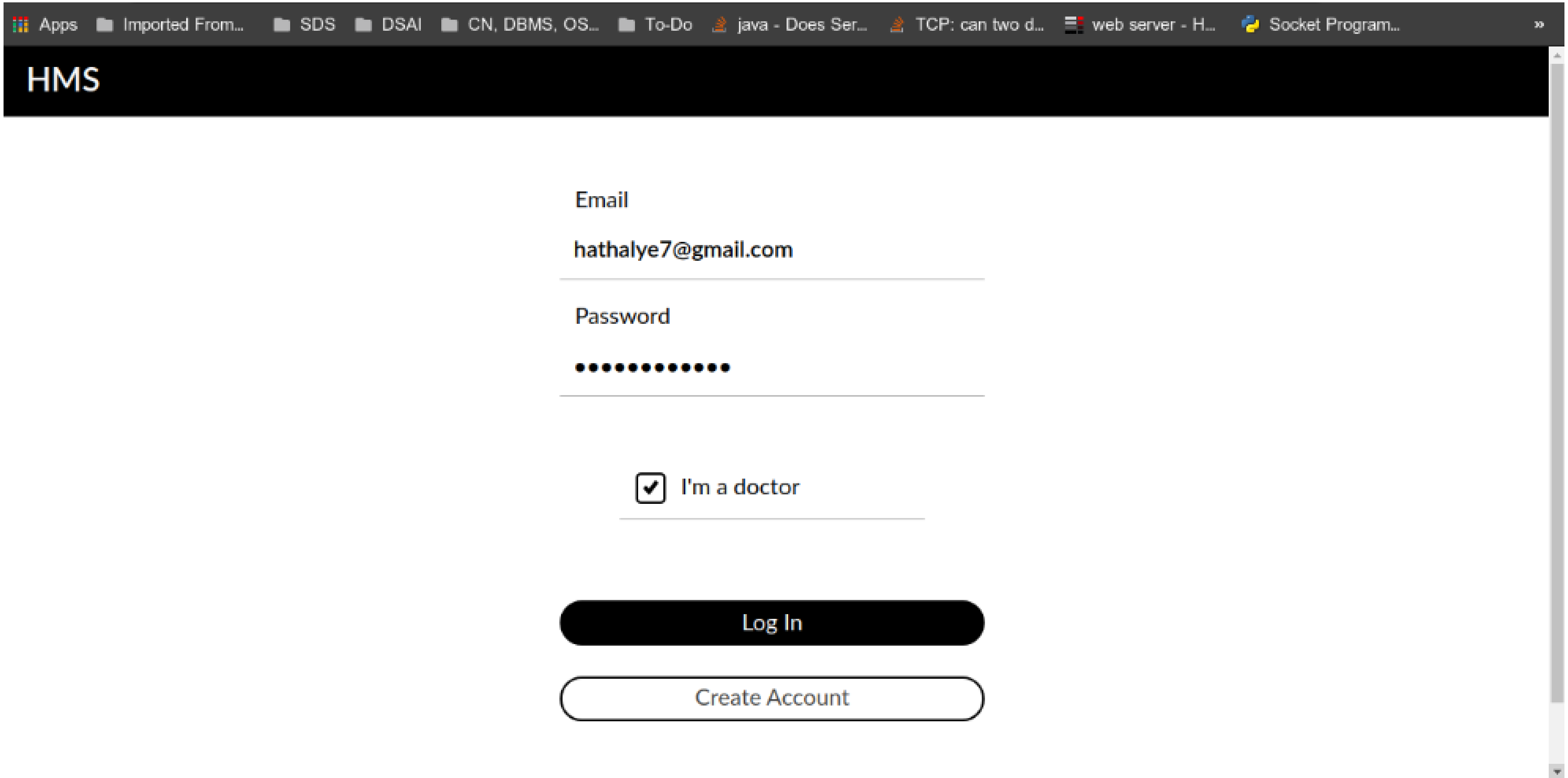
Schedule No

Please enter schedule number

Gender

Female or Male

Password



HMS

Email

hathalye7@gmail.com

Password

•••••••••••••

☑ I'm a doctor