Work in Progress Document

Project Name: DocHub

Group: 56

Team Name: Panchayat

Ishaan Arora (2018041) • Shashwat Aggarwal (2018097) • Abhinay Gupta (2018209) • Rishabh (2018255) • Bhunesh (2018280)

https://github.com/Bhunesh2000/dochub

Assumptions:

- The start and end times entered will be logical i.e. start time < end time
- The test results will be populated by a lab which is not a part of our system or by the doctor.
- The pharmacy will bill the patient for medicines in its separate system.
- QR code will be scanned by an external device

WEEK 1

APPLICATION

Dochub is a clinic/doctor appointment system along with prescription and patient history.

Our application will allow clinics to join in which patients can book an appointment with the doctors in the clinic during their available hours. The patient can submit their medical history which will be made available to the doctor after booking the appointment. The patient can also view the medicines prescribed by the doctor and show them to the pharmacy in the form of a code to buy the necessary medicines.

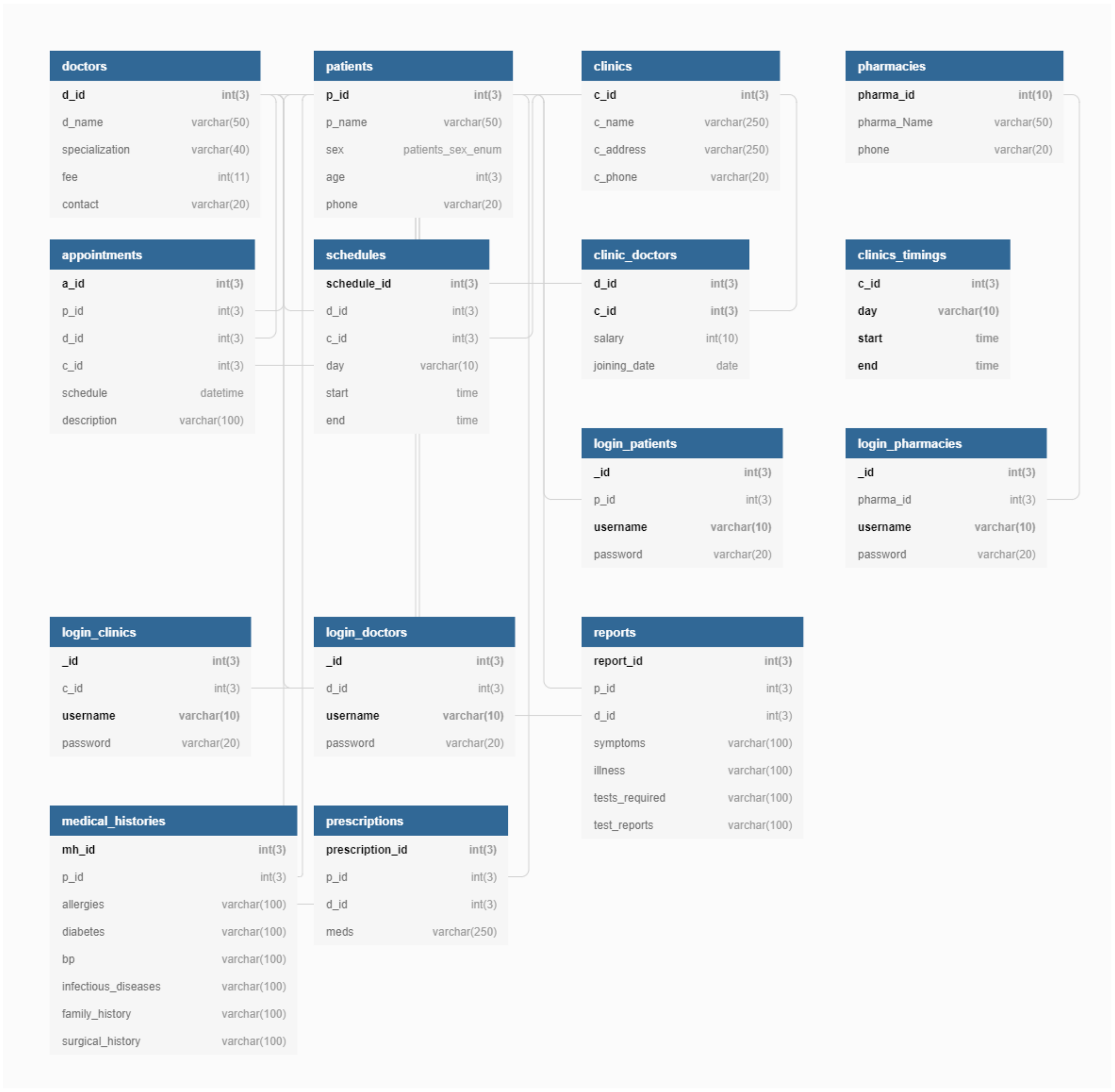
STAKEHOLDERS

1. **Clinic:** The clinic will be responsible for adding doctors to the clinic working in it. The clinic will also be responsible for removing any doctors that leave the clinic. Also, it will set the charges for various doctors.
2. **Doctors:** Each doctor will be able to set their visiting hours. They will be able to login into their account and view/manage their upcoming appointments. They might choose to cancel some appointments they are not able to cover. They can also view the medical history of the patients who have booked an appointment while treating them. They will also give prescriptions and assign lab tests. The medical history of the patient will also be visible to the doctor and can be updated by them too. Doctors will also have an option to leave the clinic which will be executed by the clinic admin as mentioned above.

3. **Patients:** They can log in to their account and view their upcoming appointments. If they choose to, they can cancel any appointment. Patients can book appointments from the list of doctors by specialization and make an appointment during the available times. The patients will also be able to view their prescriptions given by the doctor. They can show the code of the prescriptions to the pharmacy to purchase the required medicines.
4. **Pharmacy:** They can scan the code shown to them by the patients to get the list of prescribed medicines given to the patient and then bill them for the patient.

WEEK 2

Schema Diagram



Description

Clinic: It will be performing actions such as :

- Adding doctors to the clinic
- Setting schedules of doctors
- Fix salaries of doctors
- Managing the appointment of the patients with the doctors

For implementing these functionalities for the clinic we will be using data entities such as -

Table for doctors at clinic -

- Doctor ID (int)
- Doctor Name (varchar)
- Contact No. (int)
- Specialization (varchar)
- Timings (datetime)
- Fee (int)

Doctors: Another stakeholder of the system are doctors. They are working at the clinic/hospital. They will be doing the following functionalities -

- Set their visiting hours at the clinic
- View and manage their appointments
- View the medical history of the patient
- Also, they can view past appointments of the patients
- They will give the prescription which will be added to the database of the patient

- They might also cancel some appointments in any emergency case.

Doctors will be performing many functions, thus they will be using -

Table containing their information -

- DoctorID (int)
- Username (varchar)
- Password (varchar)

Table of appointments -

- PatientID (int)
- DoctorID (int)
- Date (date)
- Timings (time)
- Fee (int)
- Description (varchar)
- AppointmentID (int)

Access to a particular row of the patient medical history table containing info such as -

- PatientID (int)
- BloodGroup (varchar)
- Medical History (varchar)
- Allergies (varchar)

Patients: One of the major stakeholders of the database system is the patients. They are the users who are most benefited from this system. Their medical information will be stored in the database, which will be very convenient for them as they don't have to worry about medical reports and slips anymore. Some of the major actions they will be performing while using the system are -

- Login to their account
- View clinics and doctors.
- Search for doctors/clinics based on various categories such as speciality, location, name etc.
- Schedule their meeting with a doctor
- View their past reports and prescription.
- Buy medicines from the pharmacy using the QR code facility of the system, which stores all the data of patient including prescriptions

Patients will be using the following data entries -

Table with their data in it -

- P_ID (int)
- username (varchar)
- Password (varchar)

Access to particular row of the patient's medical history table containing info such as -

- PatientID (int)
- BloodGroup (varchar)
- Medical History (varchar)

- Allergies (varchar)

View the prescription for himself/herself from the prescription table -

- PatientID (int)
- DoctorID (int)
- Illness (varchar)
- Prescription (varchar)
- Date (date)
- Next Appointment (date)
- QRCode (varchar)
- AppointmentID (int)

For viewing and editing his appointment he/she will access the appointment table -

- PatientID (int)
- DoctorID (int)
- Date (date)
- Timings (time)
- Fee (int)
- Description (varchar)
- AppointmentID (int)

For finding and purchasing from a pharmacy he/she will see the pharmacy table-

- Name (varchar)
- PharmacyID (int)
- Contact No. (int)
- Address (varchar)

Patient will find his/her bill in the pharmacy bill table -

- Patient ID (int)
- Medicines (varchar)
- Billing amount (int)
- PharmacyID (int)
- DoctorID (int)
- QR code (varchar)

Pharmacy: Another user of the system will be the people at the pharmacy. They will be provided medicines to the patients based on the information of the patient stored in the system. Their major role will be to -

- Scan the information of the patient using the QR code
- Provide the right medicine to the patient
- Bill the patient
- Store the information about purchase of the medicine in system

Pharmacy will be having access to following tables for its functions -

Table for it’s information -

- _ID (*INT)*

- *Username (varchar)*
- PASSWORD (*varchar)*

Table for prescription of the patient -

- PatientID (int)
- DoctorID (int)
- Illness (varchar)
- Prescription (varchar)
- Date (date)
- Next Appointment (date)
- QRCode (varchar)
- AppointmentID (int)

It will bill the patient and store it in the pharmacy bill table -

- Patient ID (int)
- Medicines (varchar)
- Billing amount (int)
- PharmacyID (int)
- DoctorID (int)
- QR code (varchar)

WEEK 3

The following Tables would be required by our system. Schema can be viewed in schema.sql in SQL folder.

1. **CLINIC**
    - C_ID *int(3)*
    - C_NAME *varchar(250)*
    - C_ADDRESS *varchar(250)*
    - C_PHONE *varchar(20)*
    - Primary Key (C_ID)
2. **CLINIC_TIMINGS**
    - C_ID *int(3)*
    - DAY *varchar(10)*
    - START *time*
    - END *time*
    - Primary Key (C_ID, DAY, START, END)
3. **CLINIC_DOCTORS**
    - D_ID *int(3) Foreign Key References* DOCTORS(D_ID)
    - C_ID *int(3) Foreign Key References* CLINICS(C_ID)
    - SALARY int(10)
    - JOINING *date*
    - Primary Key (D_ID, C_ID)

4. **DOCTORS**
    - D_ID *int(3)*
    - D_NAME *varchar(50)*
    - SPECIALIZATION *varchar(40)*
    - FEE *int(11)*
    - CONTACT *varchar(20)*
    - Primary Key (D_ID)
5. **PATIENTS**
    - P_ID *int(3)*
    - P_NAME *varchar(50)*
    - SEX *char (1) enum (‘m’,’f’)*
    - AGE *int(3)*
    - PHONE *varchar(20)*
    - Primary Key (P_ID)
6. **PHARMACY**
    - PHARMA_ID *int(3)*
    - PHARMA_NAME *varchar(50)*
    - PHONE *varchar(20)*
    - Primary Key (Ph_ID)
7. **LOGIN_CLINIC**
    - _ID *int(3)*
    - USERNAME *varchar(10)*
    - PASSWORD *varchar(15)*
    - C_ID *int(3) Foreign Key References* CLINICS(C_ID)

- Primary Key (_ID)

8. **LOGIN_DOCTORS**
    - _ID *int(3)*
    - USERNAME *varchar(10)*
    - PASSWORD *varchar(15)*
    - D_ID *int(3) Foreign Key References* DOCTORS(D_ID)
    - Primary Key (_ID)
9. **LOGIN_PATIENTS**
    - _ID *int(3)*
    - USERNAME *varchar(10)*
    - PASSWORD *varchar(15)*
    - P_ID *int(3) Foreign Key References* DOCTORS(D_ID)
    - Primary Key (_ID)
10. **LOGIN_PHARMACY**
    - _ID *int(3)*
    - USERNAME *varchar(10)*
    - PASSWORD *varchar(15)*
    - Ph_ID *int(3) Foreign Key References* PHARMACY(Ph_ID)
    - Primary Key (_ID)
11. **APPOINTMENTS**
    - A_ID *int(3)*
    - P_ID *int(3) Foreign Key References PATIENTS(P_ID)*

- D_ID *int(3) Foreign Key References* DOCTORS(D_ID)
- C_ID *int(3) Foreign Key References* CLINICS(C_ID)
- SCHEDULE *datetime*
- DESCRIPTION *varchar(1000)*
- Primary Key (A_ID, P_ID, D_ID, C_ID, SCHEDULE)

12. **MEDICAL_HISTORY**
    - Med_ID *int(3)*
    - P_ID *int(3) Foreign Key References* PATIENTS(P_ID)
    - ALLERGIES varchar(100)
    - Diabetes *varchar(100)*
    - BP *varchar(100)*
    - Infectious_diseases *varchar(100)*
    - Family_history *varchar(100)*
    - Surgical_history *varchar(100)*
    - Primary Key (Med_ID)
13. **PRESCRIPTION**
    - PRESCRIPTION_ID *int(3)*
    - P_ID *int(3) Foreign Key References* PATIENTS(P_ID)
    - D_ID *int(3) Foreign Key References* DOCTORS(D_ID)
    - Meds *varchar(250)*
    - Primary Key (PRESCRIPTION_ID)
14. **SCHEDULE**
    - Schedule_ID *int(3)*
    - D_ID *int(3) Foreign Key References* DOCTORS(D_ID)

- C_ID *int(3) Foreign Key References* CLINIC(C_ID)
- DAY *varchar(10)*
- START *time*
- END *time*
- Primary Key (Schedule_ID, D_ID, C_ID)

15. REPORT

- REPORT_ID *int(3)*
- P_ID *int(3) Foreign Key References* PATIENTS
- D_ID *int(3) Foreign Key References* DOCTORS
- SYMPTOMS varchar(100)
- ILLNESS varchar(100)
- TEST_REQUIRED varchar(100)
- TEST_REPORTS varchar(100)
- Primary Key (REPORT_ID)

WEEK 6

The first column is the Search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

Therefore, The reason for using primary key or candidate key is that indexing needs a unique key that's why we always use primary or foreign or candidate keys for indexing.

For minimizing access time and managing the tradeoff between storage taken by indexing and access time, we observe those keys that are used in updation and selection. Hence, these are the following indices created.

Single Indexing: - Make primary and foreign key the basic index for every table

1. Clinic :- Index(C_ID)
2. Doctors :- Index(D_ID)
3. Clinic Doctors:- Index(C_ID,D_ID,Schedule_ID)
4. Clinic Timings:- Index(D_ID,Day Start, End)
5. Patients:- Index(P_ID)
6. Pharmacy:- Index(Ph_ID)
7. Login Clinic:- Index(C_ID,_ID)
8. Login Doctors:- Index(D_ID,_ID)
9. Login Patients:- Index(D_ID,_ID)
10. Login Pharmacy:- Index(*Ph_ID,_ID*)
11. Appointments:- Index(A_ID, C_ID,D_ID)
12. Medical History:- Index(Med_ID,P_ID)
13. Prescription:- Index(Prescription_ID)
14. Schedule:- Index(C_ID,D_ID,Schedule_ID)

15. Report:- Index(Report_ID,P_ID,D_ID)

Multi Indexing:-

Clinic, Clinic Timings, Clinic Doctors, Doctors, Schedule: - Index(C_ID,D_ID,Schedule_ID,Day Start, End)

Clinic, Login Clinic:- Index(C_ID,_ID)

Doctors, Login Doctors:- Index(D_ID,_ID)

Doctors, Login Patients:- Index(D_ID,_ID)

Pharmacy, Login Pharmacy:- Index(Ph_ID,_ID)

Appointments, Patients, Doctors, Clinic:-Index (A_ID, C_ID,D_ID,P_ID)

Medical History, Patients;- Index(P_ID,Med_ID)

Prescription, Patients, Doctors:- Index(Prescription_ID,P_ID,C_ID)

Report, Patients, Doctors:- Index(Report_ID,P_ID,D_ID)

11 queries involving various relational algebraic operations

1. Show doctor's all appointments with doctor id = id:

   SELECT * FROM APPOINTMENTS WHERE D_ID = id
   $\sigma_{D\_ID = id}(APPOINTMENTS)$

2. List of all doctors:

   SELECT D_NAME FROM DOCTORS

   $\Pi_{D\_NAME}\ (DOCTORS)$

3. List of all doctors who visit clinic with c_id = id

   SELECT D_NAME FROM DOCTORS

$\Pi_{D\_NAME}\ (DOCTORS)$

4. Validate Login

   Select {x}_id, {x}_name from {user_type} natural join login_{user_type} where username='{username}' and password='{password}'

   $\Pi_{\text{\{x\}\_id, \{x\}\_name}}(\boldsymbol{\sigma}_{\text{username'\{username\}' and password='\{password\}'}}(\text{user\_type} \bowtie \text{user\_type}))$

5. View Doctor’s Upcoming Appointments

   "select a_id, p_id, p_name, c_name, schedule, description from appointments natural join clinics natural join patients where d_id = {d_id} and c_id = {c_id} and schedule >= now()"

   $\Pi_{\text{a\_id, p\_id, p\_name, c\_name, schedule, description}}(\boldsymbol{\sigma}_{\text{d\_id = \{d\_id\}} \wedge \text{c\_id = \{c\_id\}} \wedge \text{ schedule >= now()"}}(\text{clinics} \bowtie \text{patients}))$

6. Create_user

   "insert into {login_table} values(NULL, {x_id}, '{username}', '{password}');"

   $\text{r1 <- r}_{\text{NULL, \{x\_id\}, '\{username\}', '\{password\}'}}(\text{login\_table})$

7. view_doc_past_appointments

   "select a_id, p_id, p_name, c_name, schedule, description from appointments natural join clinics natural join patients where d_id = {d_id} and c_id = {c_id} and schedule < now()"

   $\Pi_{\text{a\_id, p\_id, p\_name, c\_name, schedule, description}}(\boldsymbol{\sigma}_{\text{d\_id = \{d\_id\}} \wedge \text{c\_id = \{c\_id\}} \wedge \text{ schedule < now()"}}(\text{clinics} \bowtie \text{patients}))$

8. Clinic_remove_doc

   DELETE FROM clinic_doctors WHERE c_id = {c_id} and d_id = {d_id}

   $\text{Clinic\_doctors <- clinic\_doctors - } \boldsymbol{\sigma}_{\text{c\_id = \{c\_id\}} \wedge \text{ d\_id = \{d\_id\}}}$

9. view_patient_upcoming_appointments

Select a_id, c_name, d_name, schedule, description from appointments natural join doctors natural join clinics where p_id = {p_id} and schedule < now()

$\Pi_{\text{a\_id, c\_name, d\_name, schedule, description}}(\boldsymbol{\sigma}_{\text{p\_id = \{p\_id\}} \land \text{ schedule < now()"}}(\text{appointments} \bowtie \text{clinics}))$

10. Show Schedule

    Select * from schedules where d_id = {d_id}

    $\boldsymbol{\sigma}_{\text{d\_id = \{d\_id\}}}(\text{schedules})$

11. Patient ManageProfile by changing password

    Update login_patients set password='{new}' where p_id={p_id}

    $\text{Login\_patients} \leftarrow \Pi_{\text{password='\{new\}'}}(\boldsymbol{\sigma}_{\text{p\_id=\{p\_id\}}}(\text{Login\_patients}))$

WEEK 7

Embedded queries are written in python. We have used the mysql-connector for python to connect to our database. This gives us a cursor to our database through which we execute all SQL queries. This can be seen in the queries.py file.

Following are some advanced aggregation queries:

1. Count the number of patients of each doctor.

   Select count(P_ID) as number_of_patients

   from appointment

   group by doctors

   having count(P_ID>=0)

   order by number_of_patients ;

2. Count the number of doctors whose working days are the same.

   Select count(D_ID)as number_of_doctors

   From schedule as T, schedule as S

   Where T.Days=S.Days

3. Find the number of doctors of each specialization

   Select count(D_ID) as number_of_doctors

   From doctors

   Group by specialization

   Order by number_of_doctors

4. Find the name of doctors who have the most no of patients today.

   Select MAX (number_of_patients) , D_ID

   From (Select count(P_ID) as nummber_of_patients , D_ID

   From Appointments

   Where SCHEDULE = “NOW()”

   Group by D_ID);

5. Find the number of doctors who have more than average of patients daily and specialize in a cardiologist.

   Select AVG(number_of_patients) as average, D_ID

   From (Select count(P_ID) as nummber_of_patients , D_ID

   From Appointments

   Group by D_ID)

   Where specialization= Cardiologist

   Having number_of_patients>=average

   ORDER by number_of_patients

6. To find another doctor of the same specialisation for the same appointment time.

   Select D_ID ,specialization

   From doctors as T, doctors as S, appointment as A,appointment as B

   Where T.specialisation=S.specialisation and A.A_ID=B.A_ID

```
Group by (D_ID,specialization)
```

7. Find the number of patients that the doctor with a particular D_ID has to see today.

```
Select count(P_ID) as No_of_patients

From Appointment

Where D_ID = 12345 ;
```

8. Count the number of appointments of a particular patient

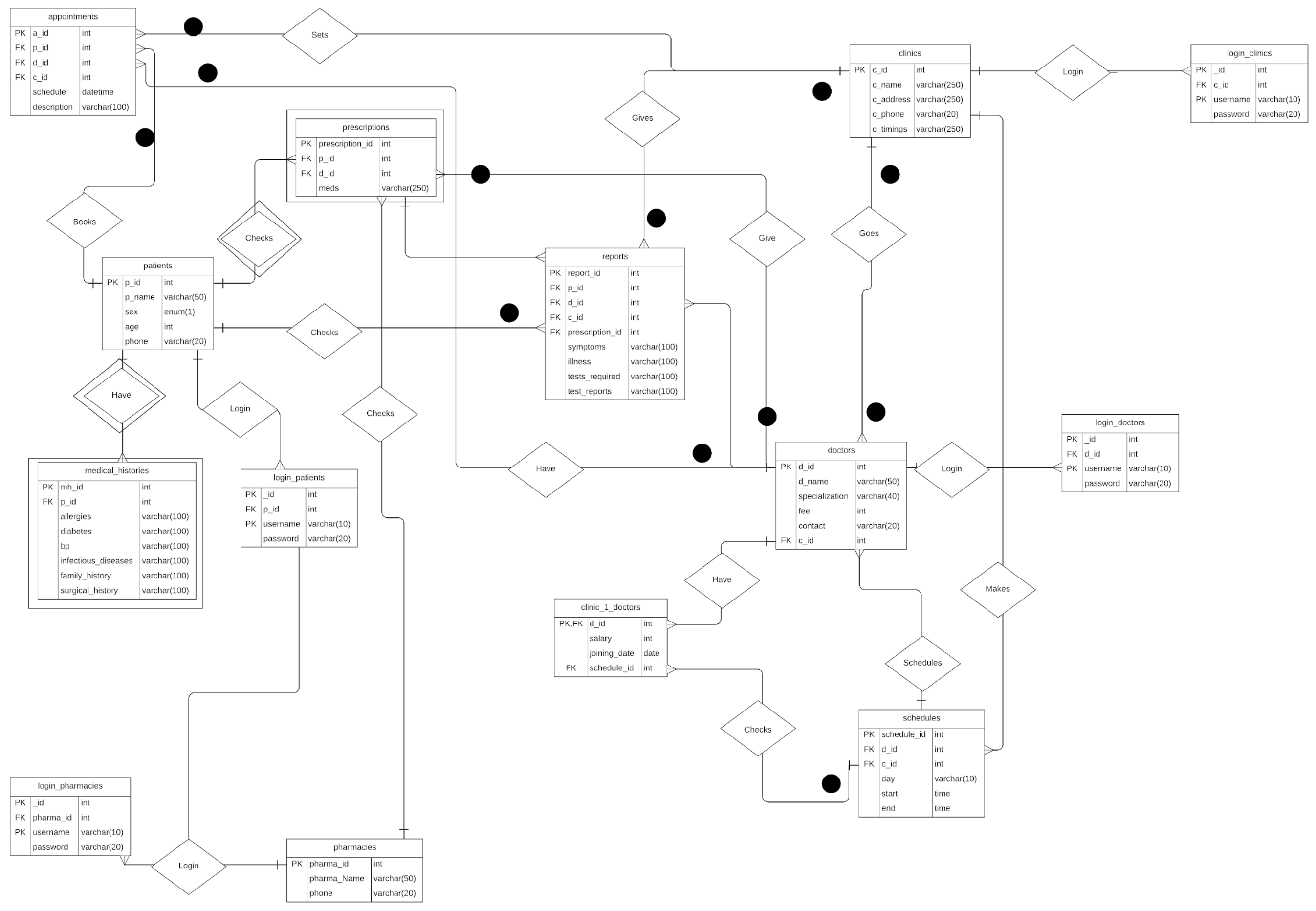
```
Select count(A_ID)

From Appointment

Where P_ID = 980;
```

WEEK 8

ER Diagram



appointments

PK	a_id	int
FK	p_id	int
FK	d_id	int
FK	c_id	int
	schedule	datetime
	description	varchar(100)

clinics

PK	c_id	int
	c_name	varchar(250)
	c_address	varchar(250)
	c_phone	varchar(20)
	c_timings	varchar(250)

login_clinics

PK	_id	int
FK	c_id	int
PK	username	varchar(10)
	password	varchar(20)

prescriptions

PK	prescription_id	int
FK	p_id	int
FK	d_id	int
	meds	varchar(250)

reports

PK	report_id	int
FK	p_id	int
FK	d_id	int
FK	c_id	int
FK	prescription_id	int
	symptoms	varchar(100)
	illness	varchar(100)
	tests_required	varchar(100)
	test_reports	varchar(100)

patients

PK	p_id	int
	p_name	varchar(50)
	sex	enum(1)
	age	int
	phone	varchar(20)

medical_histories

PK	mh_id	int
FK	p_id	int
	allergies	varchar(100)
	diabetes	varchar(100)
	bp	varchar(100)
	infectious_diseases	varchar(100)
	family_history	varchar(100)
	surgical_history	varchar(100)

login_patients

PK	_id	int
FK	p_id	int
PK	username	varchar(10)
	password	varchar(20)

doctors

PK	d_id	int
	d_name	varchar(50)
	specialization	varchar(40)
	fee	int
	contact	varchar(20)
FK	c_id	int

login_doctors

PK	_id	int
FK	d_id	int
PK	username	varchar(10)
	password	varchar(20)

clinic_1_doctors

PK,FK	d_id	int
	salary	int
	joining_date	date
FK	schedule_id	int

schedules

PK	schedule_id	int
FK	d_id	int
FK	c_id	int
	day	varchar(10)
	start	time
	end	time

login_pharmacies

PK	_id	int
FK	pharma_id	int
PK	username	varchar(10)
	password	varchar(20)

pharmacies

PK	pharma_id	int
	pharma_Name	varchar(50)
	phone	varchar(20)

Bonus Implementation

Public Question and Answer system (QnA)

Considering the spread of the COVID-19 pandemic having a general awareness of the basic and most important essential practices are more important than ever and should be made available to all. People have questions regarding their health and better living. To ensure everyone gets access to these QnAs we have a QnA button on the home page itself from which people can view the answers to the questions people have asked. The public will be allowed to ask questions to the doctors on the platform freely which can be answered by a doctor if he/she knows the answer the same. This helps in general public awareness and reduces the barrier of lack of information about common practices and precautions needed to adhere too. The answers can only be given by the doctors on our platform as they are verified doctors and hence, people don't get any wrong information.

Work Distribution (post midsem evaluation)

1. **Ishaan Arora (2018041):** ER Diagram and SQL queries for login and update profile. Updated SQL database schema with changes.
2. **Shashwat Aggarwal (2018097):** Wrote all embedded SQL queries for all operations besides update and login (done by Ishaan). Integrated the back end with the front end to fetch data and send data from the website to the database and vice-versa using Flask and the created APIs (queries.py). Beautified (CSS) of all web pages. Created the QnA bonus and implemented the same including its HTML.

3. **Abhinay Gupta (2018209):** Week 6 relational algebra and Week 7 advanced aggregate queries written above. Populated the tables for the database.
4. **Rishabh (2018255):** Week 6 indexing implementation. Week 6 relational algebra and Week 7 advanced aggregate queries written above. Populated the tables for the database.
5. **Bhunesh (2018280):** Created all webpages in the website with Flask and HTML.