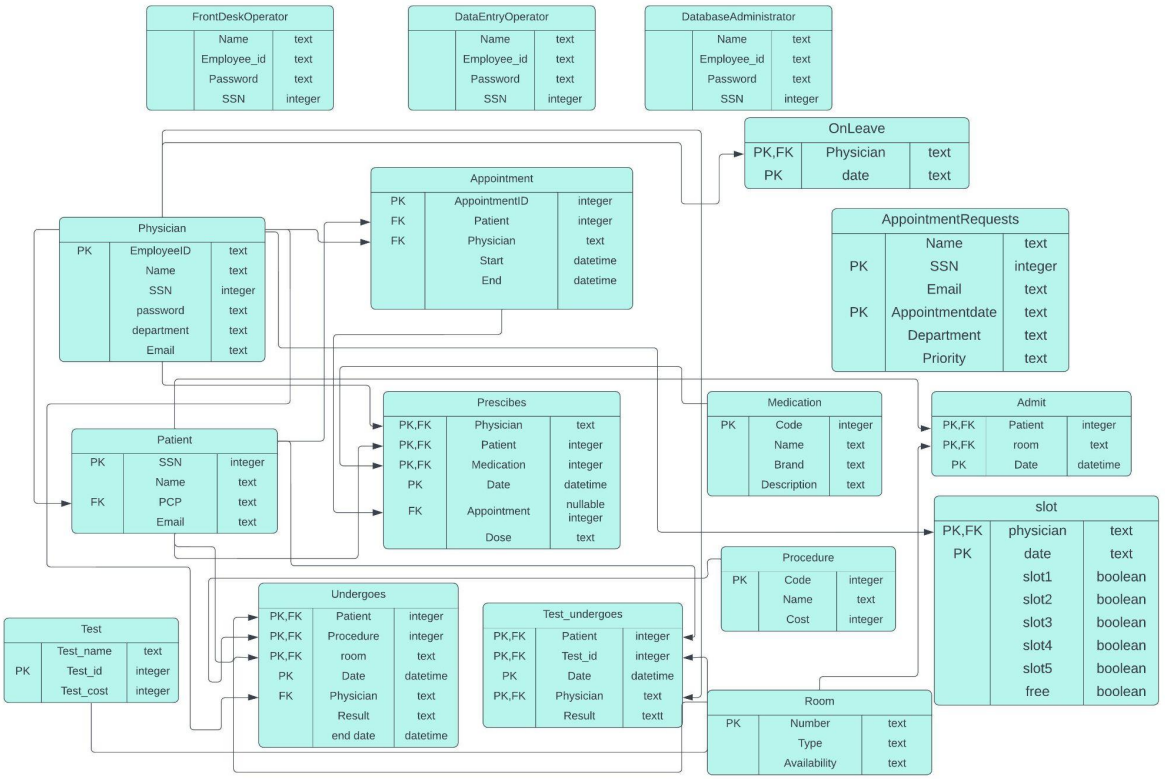
Hospital Management System

Group Members:

Jaswanth Kumar Chukka(20CS10021)
Sushanth Marada(20CS10034)
Bhanu Teja Siripuram(20CS10059)
Sathvik Reddy Garlapati(20CS10028)
Sai Sahan Talabattula(20CS30055)

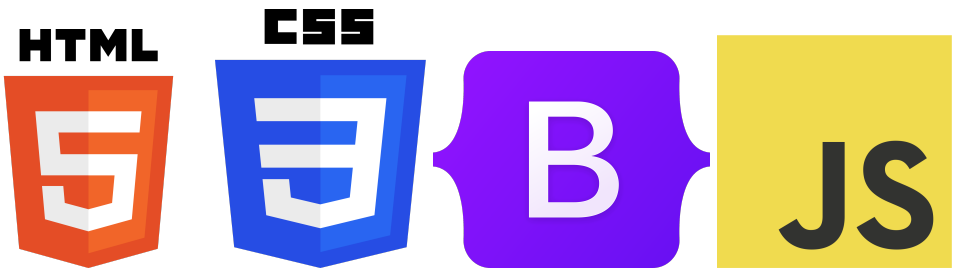
1. Schema of the underlying database



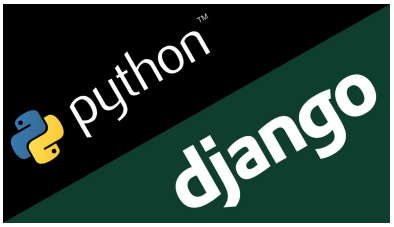
2. Languages and tools used

Protocol used : HTTP

Front end : HTML, CSS, Bootstrap,Java Script



Back end : Django, PostgreSQL, Psycopg2



Operating systems: Windows, Linux, MacOS

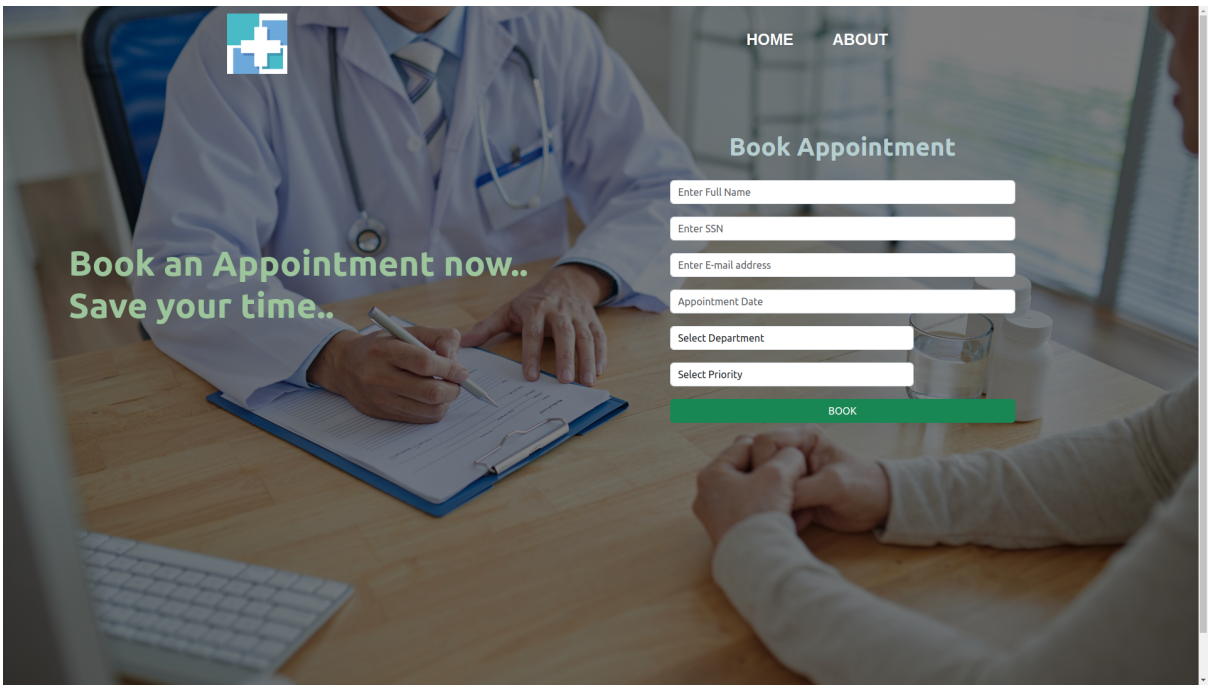
For viewing and manipulating the database used **pgadmin4.**

3 . Design of the Web application

1. Book Appointment :

This page allows the users to book their appointment on the required date, also able to choose the physician department and can mark their case emergency or general. They need to enter their full name, ssn, email.

Here the user receives the appointment details with physician details, while the doctor only receives mail of the appointment if it is only an emergency case.



Book appointment page

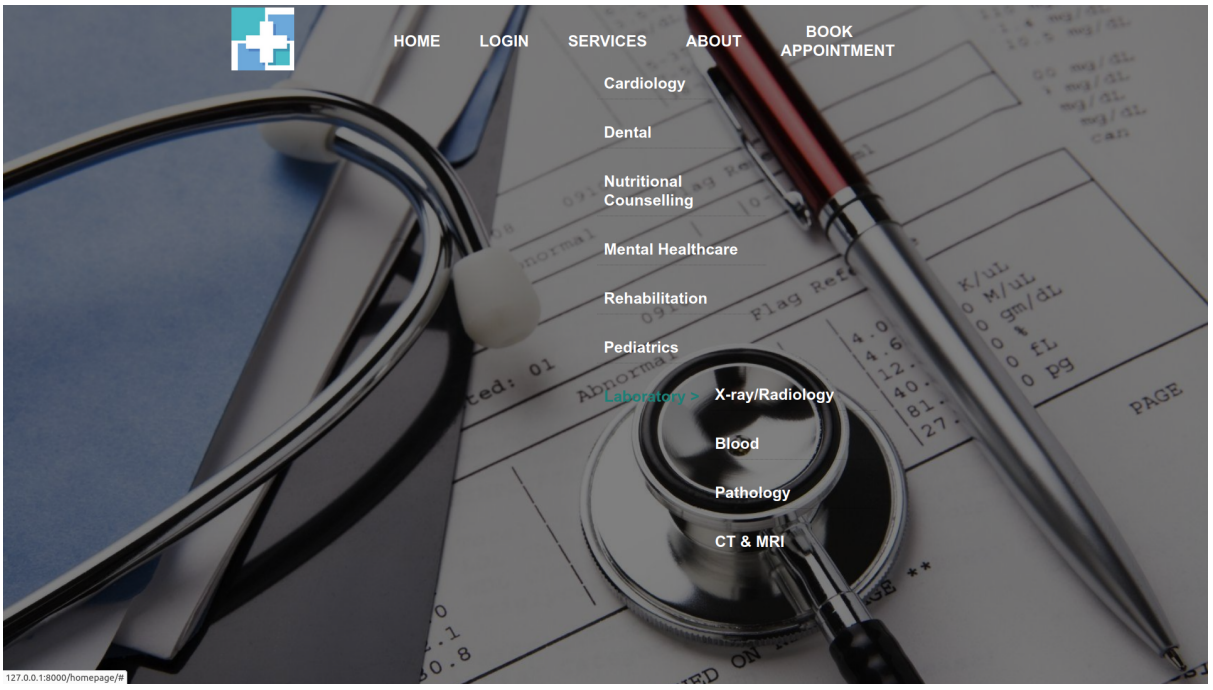
However the appointment approval is on the front desk operator.

2. Home page:

Home page is a default option for logout from any user home pages. Its navigation bar has direct access to login, services,

and book appointment pages.

Home : redirects the user to the home page.
Services: It lists the various services available at the hospital.



3. Login page :

This page is the same for different types of users. Their login credentials like username and password are separated for different types of users with a drop down where they can choose their type.

Login page

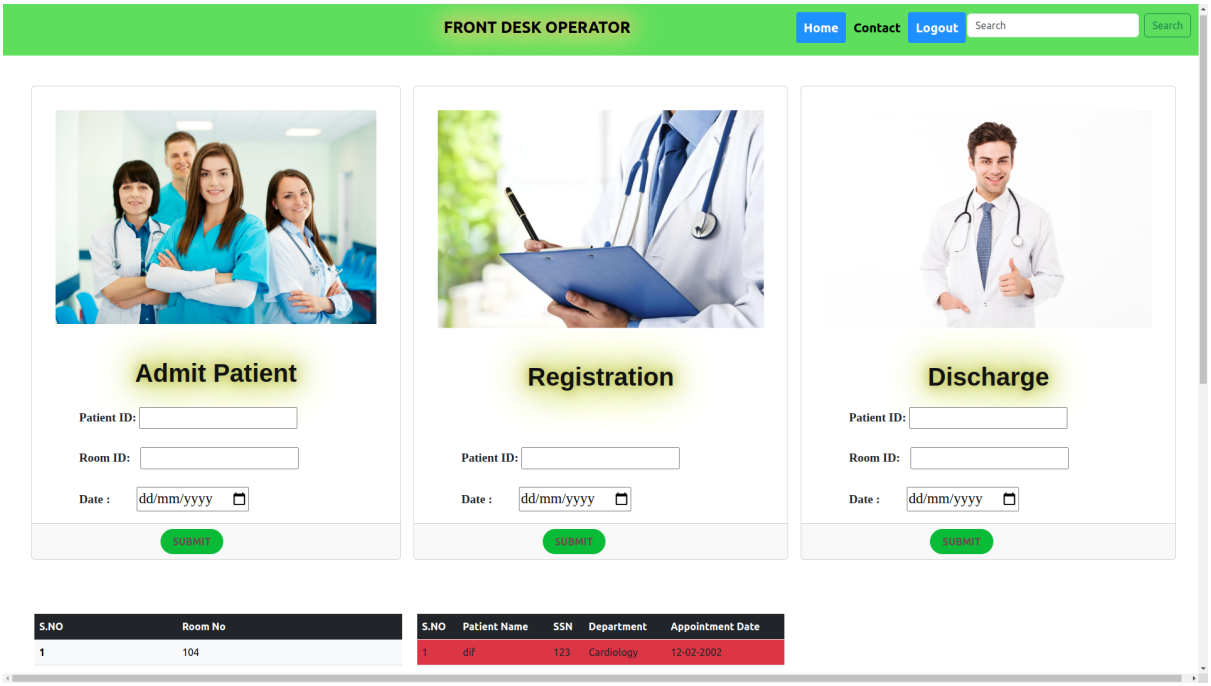
- Database Administrator: can add/delete users .
- Doctors: can query patient information.
- Data Entry Operator: can enter patient data about tests and treatments.
- Front desk Operator: can register/admit/discharge patients.

Login : directs the user to the login page.

- Login also lists the available types of users who can login into our system.
- A user can login to his account by entering his particular credentials in this page.
- The login page consists of a home button option to return back to the homepage.

4. Front desk Operator:

Front desk operator able to admit , register and discharge patients with the available room ids if needed. If the case is an emergency one it shows in red on his page and on registering/admitting the patient a mail goes to the physician.



Front desk operator

Here the physician is allotted randomly by checking their on_leave status and their appointment timings on that particular date.

In this design every doctor has five slots daily, in which the patient is assigned in free slots of the doctor. In case of no free slots the appointment request will be declined.

This design ensures particularly different appointments are allotted in different slots even if they are assigned the same doctor.

Slot timings:

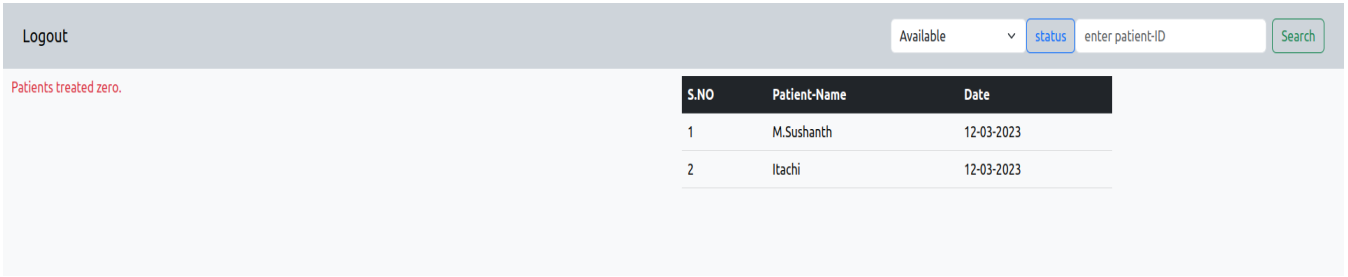
Slot-1 : 9:00 am -10:00 am
Slot-2 : 10:00 am -11:00 am
Slot-3 : 11:00 am -12:00 pm
Slot-4 : 2:00 pm -3:00 pm
Slot-5 : 3:00 pm - 4:00 pm

The algorithm implemented in the backend randomly picks a doctor with a free slot available on that date and assigns him/her to the patient.

In his page an available room list is also displayed, from which the front desk operator assigns a room when a patient is admitted.

5. Doctor:

In the Doctor homepage, navbar contains a status button with a dropdown menu which lets the doctor set his availability(available, onLeave).
It has two fields where the left side displays the patients treated by the doctor and the right side displays the appointments on that day.



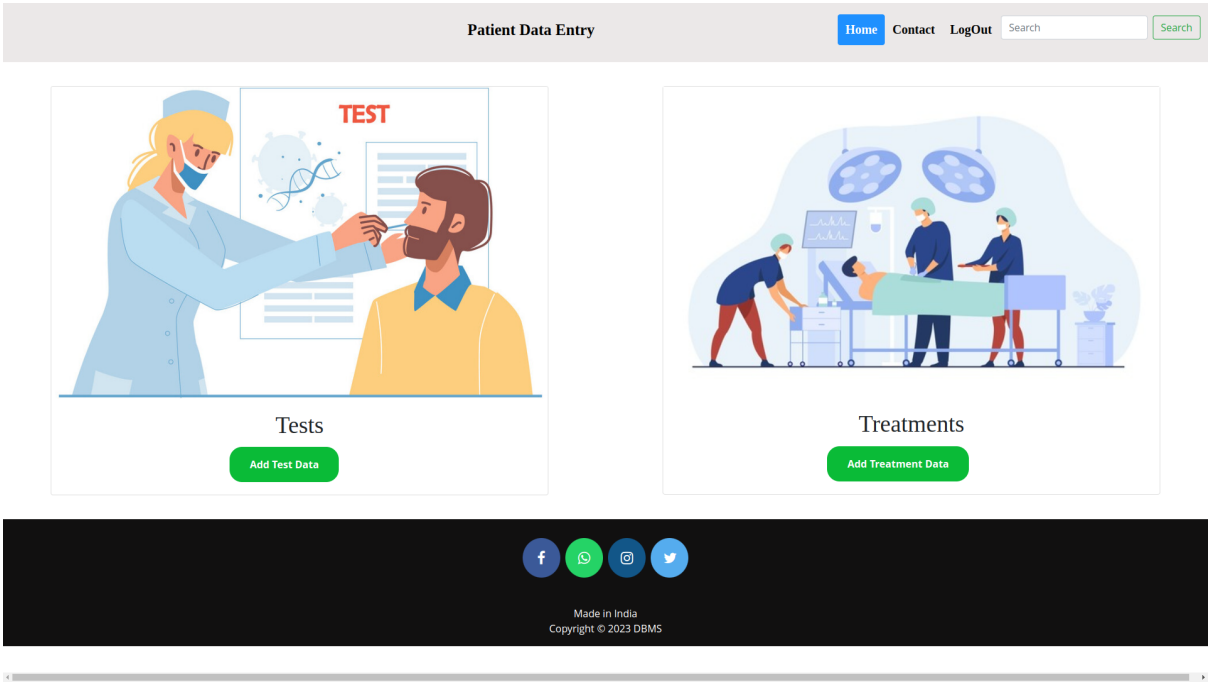
Doctor

Also can query patient details like patient information, drugs prescribed and treatments undergone from the search field in the navigation bar.

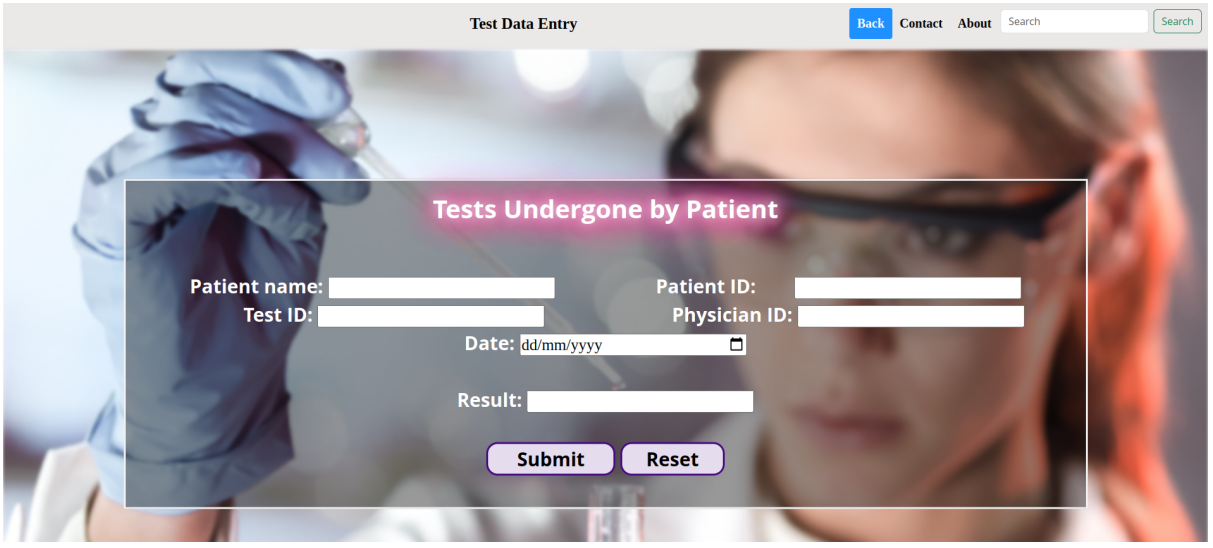
Patient information query page

6. Data Entry Operator:

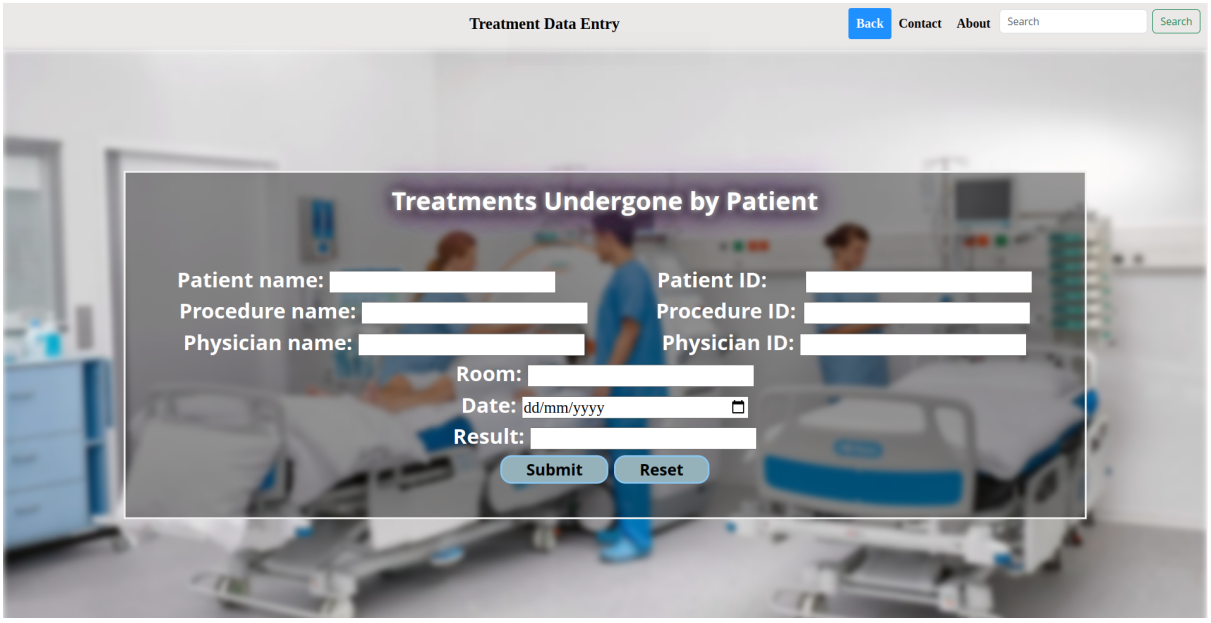
He can enter the data about the tests and treatments undergone by the patient, this info can later be accessible to the doctor for proper diagnosis of the patient.



Patient data entry



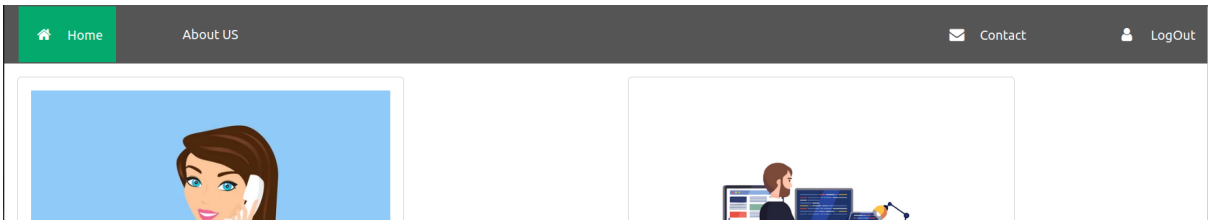
Tests undergone by the patient



Treatments undergone by patient

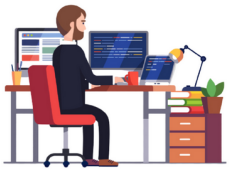
7. Database Administrator:

The administrator can log out using the logout button on the top right, this sends him/her to the main homepage.



Front Desk Operator

Add Delete



Data Entry Operator

Add Delete

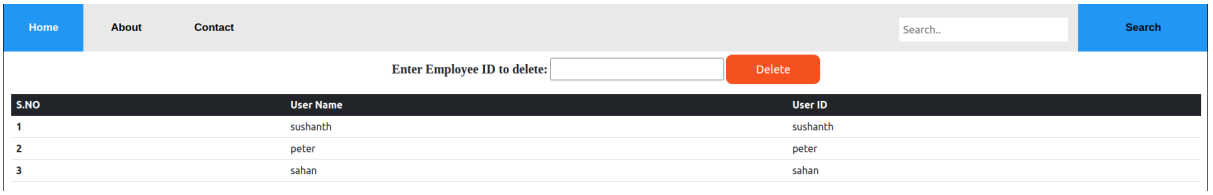
Doctor

Add Delete

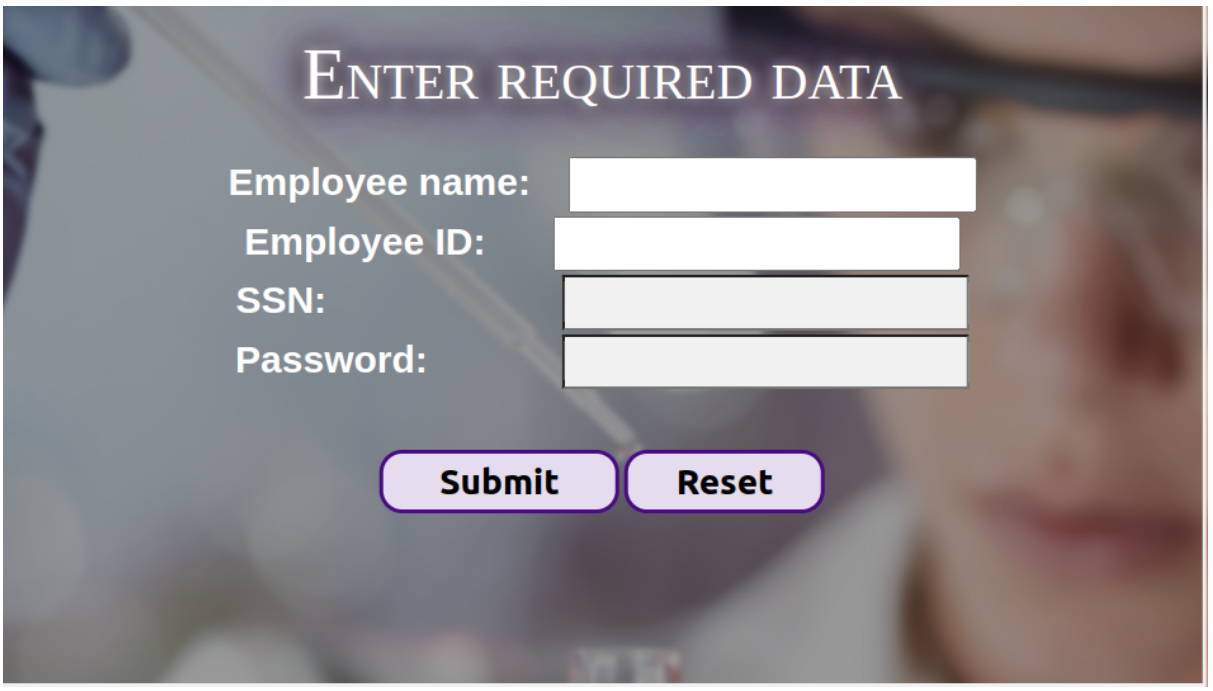
Database Administrator

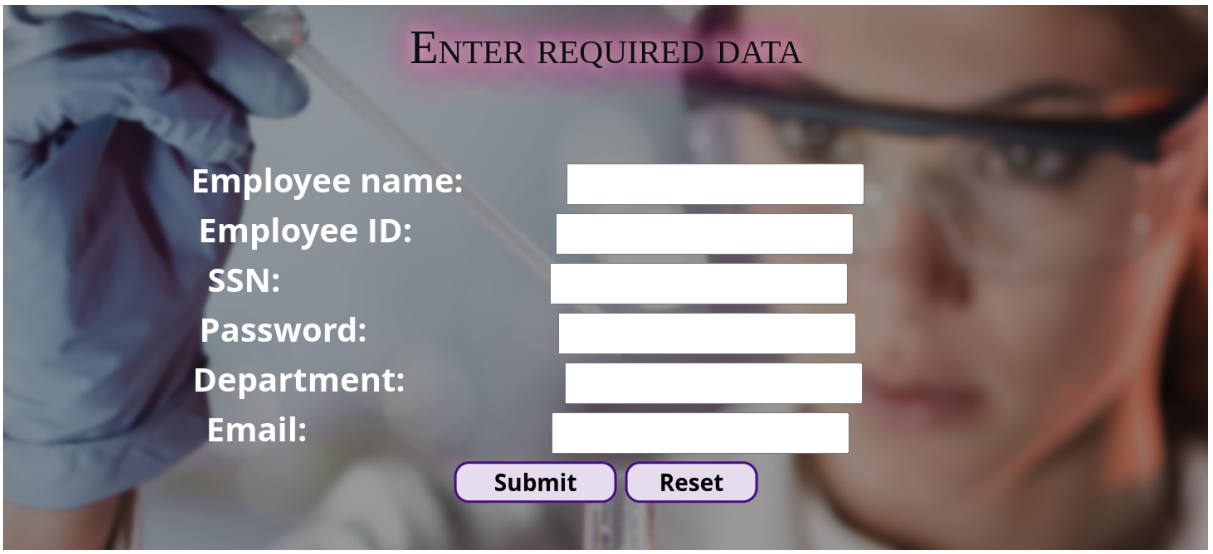
Add Delete

Administrators can add/delete different types of users. The delete page is same for all the users while the add pages for front desk operator, data entry operator and database administrator are similar but for the doctor page where it has some extra fields of entry.



User delete page

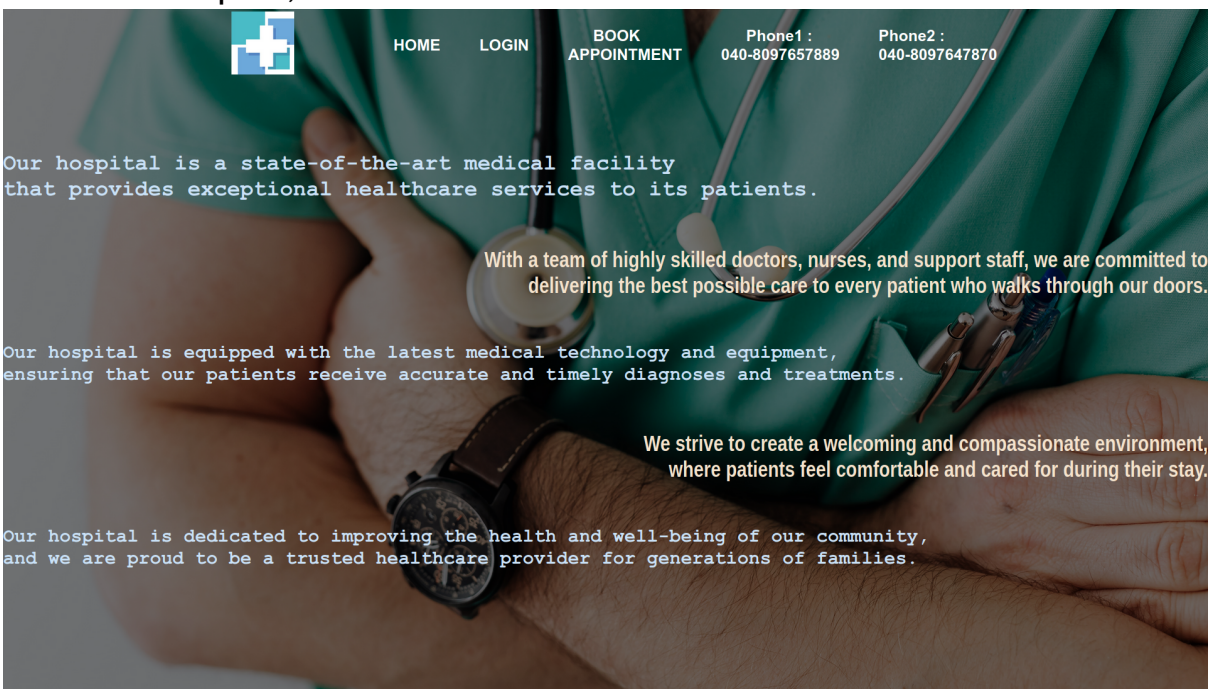




User add page

8. About page:

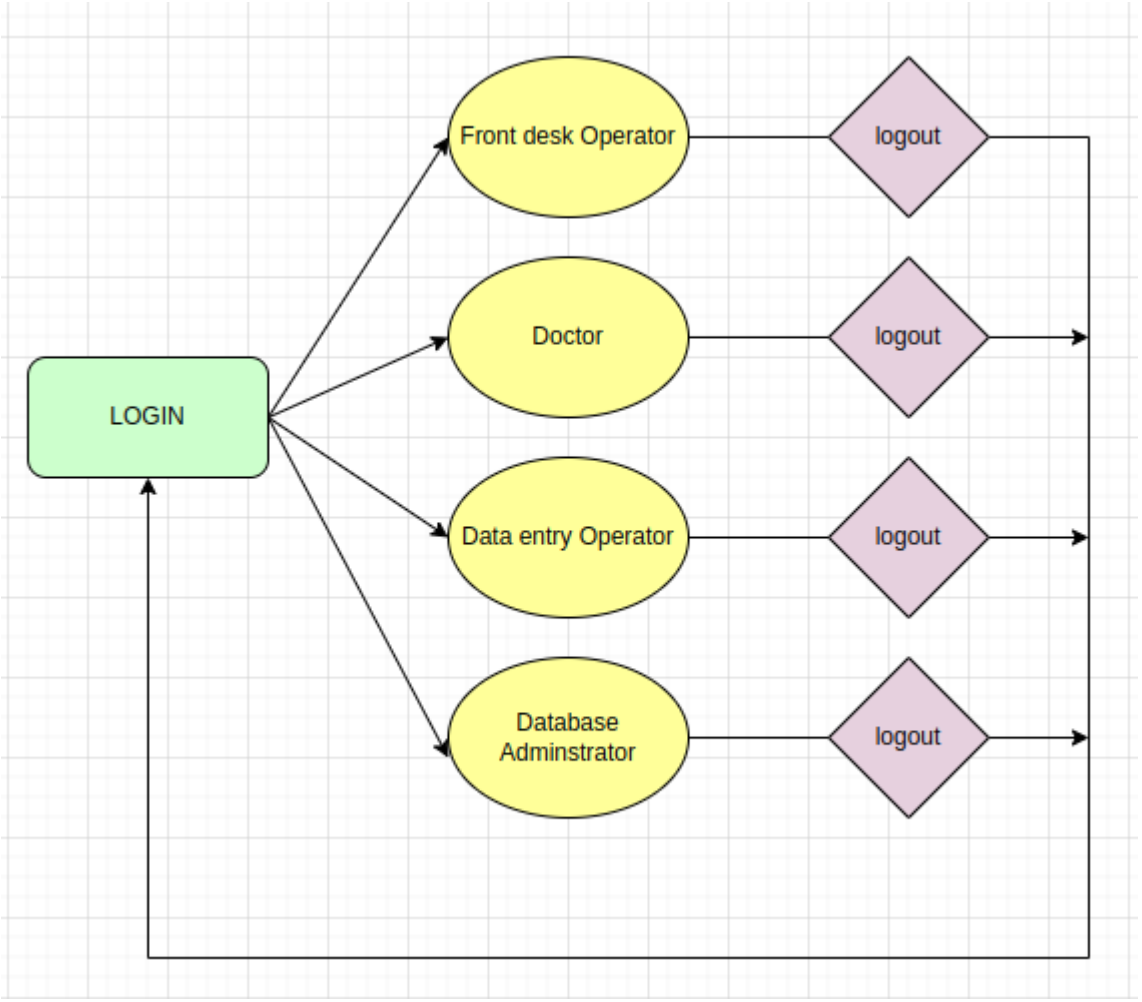
This page contains some contact information of the hospital, our mission and motto.



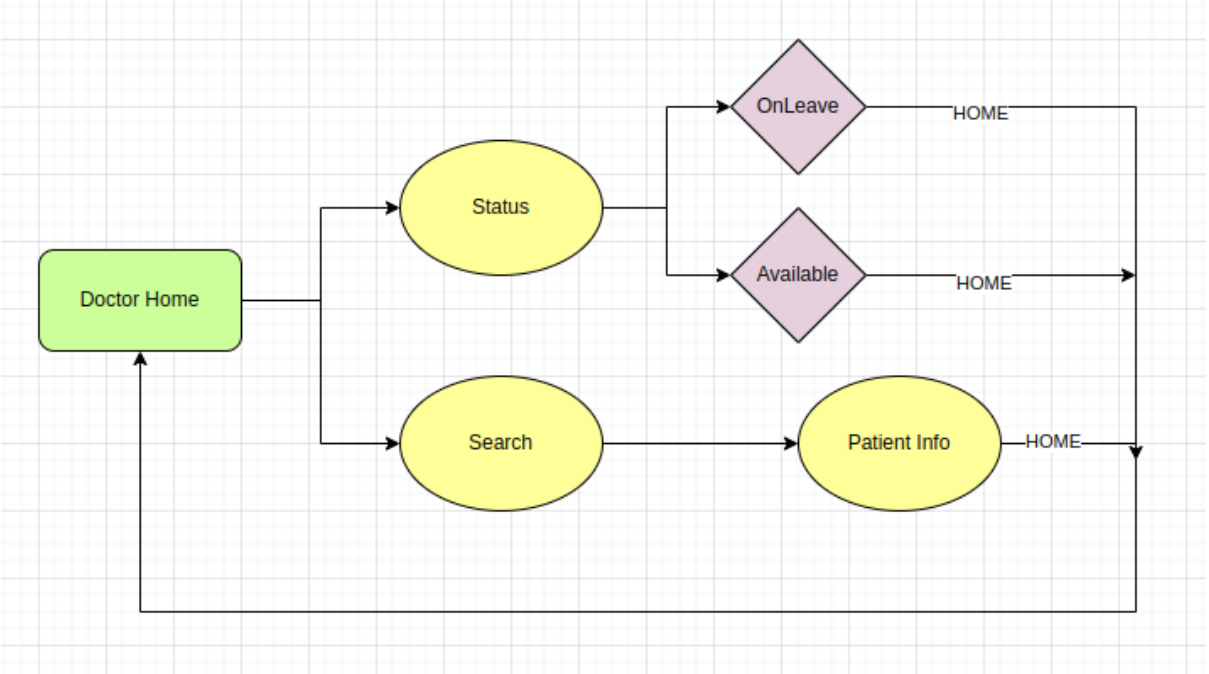
About page

4. WorkFlow:

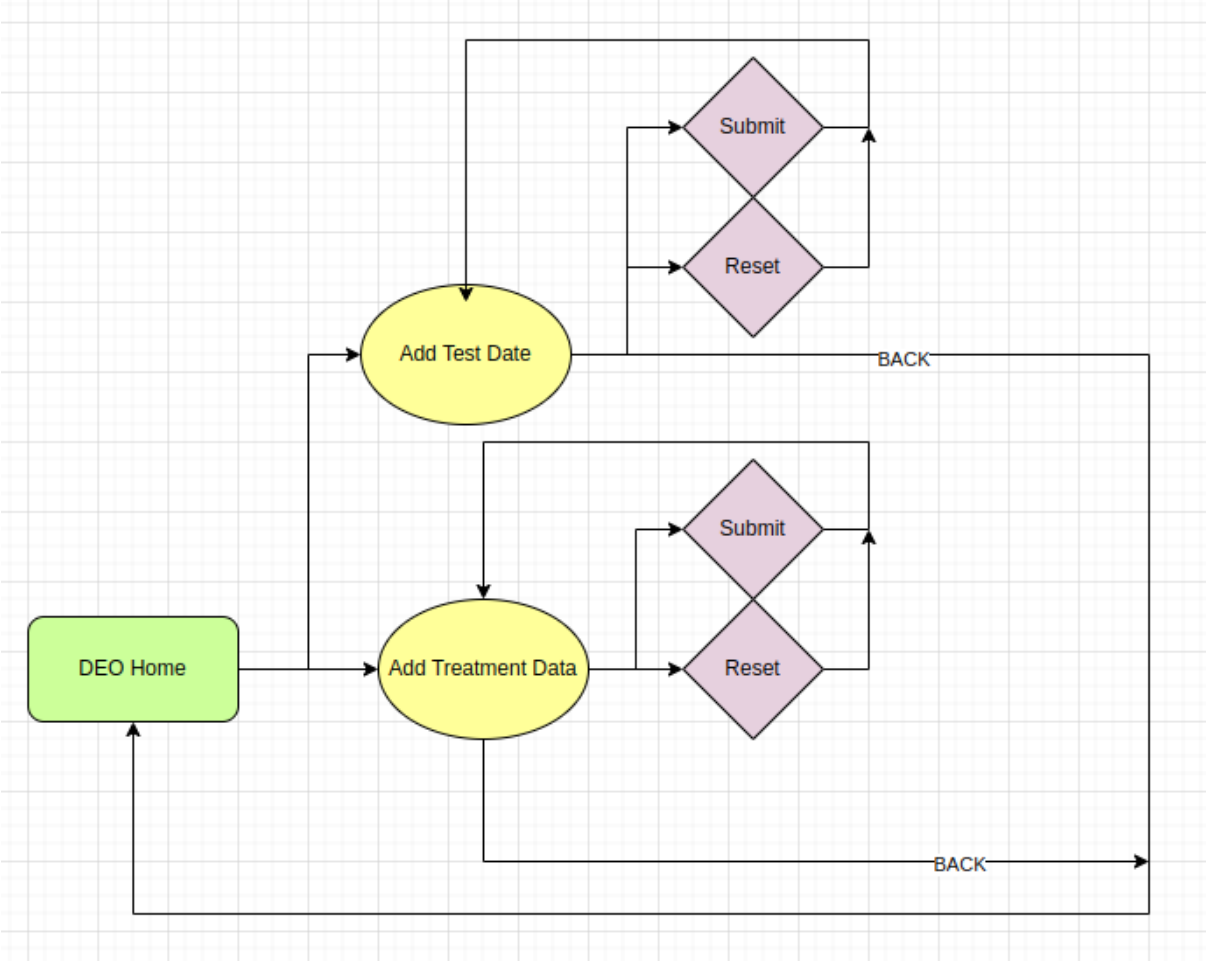
Login:



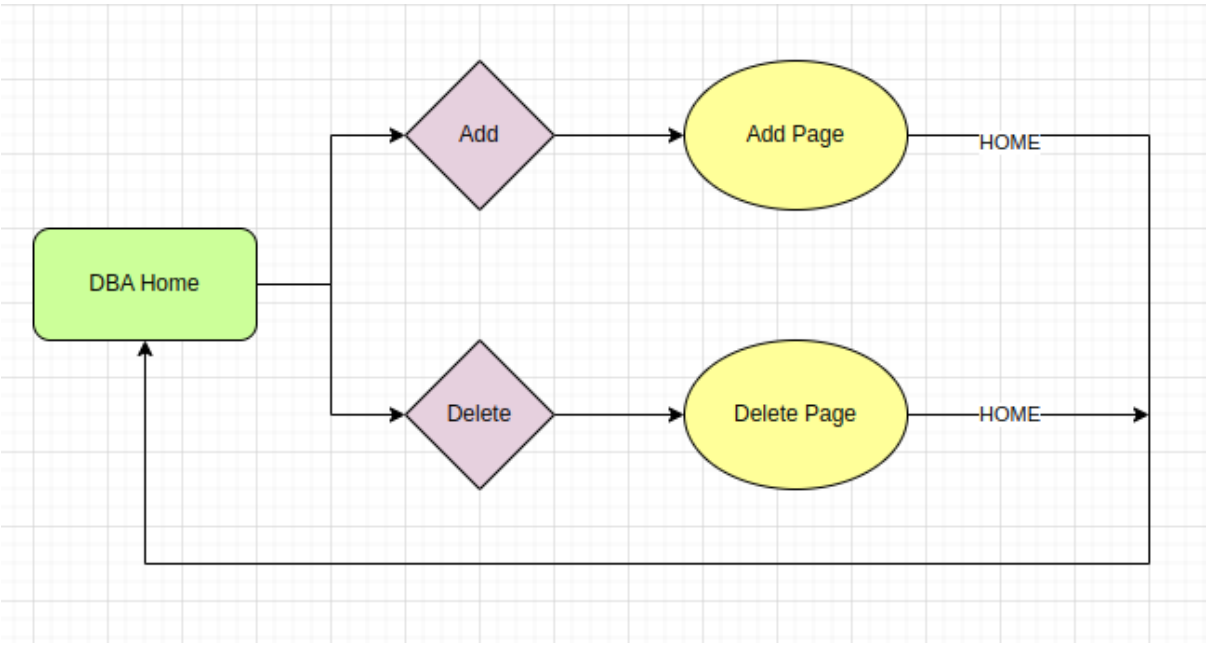
Front desk operator:



Data Entry Operator:



Database Operator:



5. Additional Functionalities:

1. Using a calendar to schedule:

We let the front desk operator schedule appointments using a calendar.

2. Sending emergency emails:

```
    def SendMail(to, subject, body, attachment=None):
        yag =
yagmail.SMTP('hospitalfrontdesk400@gmail.com',
'prjcwyudioqhhxbi')
        yag.send(to=to, subject=subject, contents=body,
                 attachments=[attachment,
'static/logo.png'])

        return
```

We displayed all appointment requests at the dashboard of the front desk operator. In the front desk operator dashboard emergency appointments are displayed in red background while the general appointments are in white.

The front desk operator needs to approve the patient appointment. In case of an emergency appointment doctor and patient both receive the appointment details through email.

To accomplish this task we have used the yagmail module in python.

If in case of failure on appointment approval(due to no free slots for the doctors in the respective department requested by the patient), the patient receives an email stating the decline of appointment request.

3. Doctor can set his/her availability status.

6. Triggers:

Since many data tables such as undergoes, test_undergoes, prescribes, appointment, admit, patient, onLeave, slot have a foreign key referencing physician table.

If there is a need to delete a row in the physician table we have to delete the rows associating to that physician in the above described tables.

```
CREATE TRIGGER del_doctor
 BEFORE DELETE
 ON physician
 FOR EACH ROW
 EXECUTE PROCEDURE del_before_doctor();
```

To accomplish this task, trigger(del_doctor) is used here. This trigger is bound with del_before_doctor().

```
CREATE OR REPLACE FUNCTION del_before_doctor()
 RETURNS trigger AS $$
BEGIN
 DELETE FROM undergoes U WHERE U.physician=OLD.employee_id;
 DELETE FROM test_undergoes T WHERE
T.physician=OLD.employee_id;
 DELETE FROM prescribes P WHERE P.physician=OLD.employee_id;
 DELETE FROM  appointment A where
A.physician=OLD.employee_id;
 DELETE FROM Admit A  WHERE A.patient in (SELECT P.ssn FROM
patient P WHERE P.pcp=OLD.employee_id);
 DELETE FROM patient P WHERE P.PCP=OLD.employee_id;
 DELETE FROM OnLeave O WHERE O.physician=OLD.employee_id;
 DELETE FROM Slot S WHERE S.physician=OLD.employee_id;
 RETURN OLD;
END;
$$
LANGUAGE 'plpgsql';
```

Here del_before_doctor(), a procedure defined which is triggered when a delete event happens on the physician table. It deletes all those rows associated with the physician.

7. List Of SQL Queries used:

1.
```
   curr.execute("DELETE FROM onleave WHERE physician=%s", (employee_id,))
```

2.
```
       curr.execute("DELETE FROM onleave WHERE physician=%s",(employee_id,))
```

3.
```
       curr.execute("INSERT INTO onleave VALUES(%s,%s) ON CONFLICT DO
NOTHING",
                    (employee_id, formatted_date,))
```

4.
```
   curr.execute("SELECT P.name,P.SSN,P.email,A.start FROM patient
P,Appointment A WHERE P.SSN=A.patient AND A.physician=%s AND A.start!=%s",
                 (employee_id, formatted_date,))
```

5.
```
   curr.execute(
       "SELECT p.name,a.start from patient p,appointment a  where
p.SSN=a.patient AND a.start=%s", (formatted_date,))
```

6.
```
       curr.execute(
           "SELECT * FROM AppointmentRequests A WHERE A.SSN=%s;",
(patient_id,))
```

7.
```
       curr.execute("select P.employee_id,P.name from Physician P where
P.department=%s and P.employee_id not in (select P1.physician from OnLeave
P1);", (department,))
```

8.
```
                curr.execute(
                    "SELECT * FROM SLOT S WHERE S.physician=%s AND
S.date=%s", (doc[0], appointmentdate,))
```

9.

```
                    curr.execute(
                        "INSERT INTO SLOT(physician,date) VALUES(%s,%s) ON
CONFLICT DO NOTHING", (doc[0], appointmentdate,))
```

10.

```
            curr.execute(
                "SELECT
P.employee_id,P.name,S.Slot1,S.slot2,S.slot3,S.slot4,S.slot5 FROM Physician
P,Slot S WHERE P.employee_id=S.physician AND S.free='1' AND
P.department=%s", (department,))
```

11.

```
                        curr.execute("UPDATE Slot SET Slot{id}='0' WHERE
physician=%s".format(
                            id=(i-1)), (physician,))
```

12.

```
                    curr.execute(
                        "UPDATE Slot  SET free='0' WHERE physician=%s",
(physician,))
```

13.

```
                curr.execute("INSERT INTO Patient(SSN,name,PCP,email)
VALUES(%s,%s,%s,%s) ON CONFLICT DO NOTHING",
                             (SSN, name, physician, email,))
```

14.

```
                curr.execute("INSERT INTO
Appointment(appointment_id,patient,physician,start) VALUES(%s,%s,%s,%s) ON
CONFLICT DO NOTHING",
                             (appointmentID, SSN, physician,
appointmentdate))
```

15.

```
                curr.execute(
                    "SELECT P.email FROM physician P WHERE P.employee_id=%s",
(physician,))
```

16.

```
                curr.execute(
                    "SELECT A.Appointment_ID,A.patient,A.physician,A.start
FROM Appointment A WHERE A.patient=%s", (patient_id,))
```

17.

```
        curr.execute(
            "DELETE FROM AppointmentRequests A WHERE A.SSN=%s",
(patient_id,))
```

18.

```
        curr.execute(
            "UPDATE ROOM  SET availability='0' WHERE number=%s", (room_id,))
```

19.

```
        curr.execute("INSERT INTO Admit(patient,room,date)  VALUES(%s,%s,%s)
ON CONFLICT DO NOTHING",
                     (patient_id, room_id, date))
```

20.

```
        curr.execute(
            "UPDATE ROOM  SET availability='1' WHERE number=%s", (room_id,))
```

21.

```
        curr.execute(
            "UPDATE Undergoes SET end_date=%s WHERE patient=%s", (date,
patient_id,))
```

22.

```
 curr.execute("SELECT R.number FROM Room R WHERE R.availability='1';")
```

23.

```
  curr.execute(
        "SELECT A.name,A.SSN,A.department,A.appointmentdate FROM
AppointmentRequests A WHERE A.priority='emergency'")
```

24.

```
   curr.execute(
        "SELECT A.name,A.SSN,A.department ,A.appointmentdate FROM
AppointmentRequests A WHERE A.priority='general'")
```

25.

```
        curr.execute(
            "DELETE FROM databaseadminstrator D WHERE D.employee_id=%s",
(employee_id,))
```

26.

```
        curr.execute(
            "SELECT D.name,D.employee_id FROM databaseadminstrator D ")
```

27.

```
        curr.execute(
            "DELETE FROM physician D WHERE D.employee_id=%s", (employee_id,))
```

28.

```
        curr.execute("SELECT D.name,D.employee_id FROM physician D ")
```

29.

```
        curr.execute(
            "DELETE FROM dataentryoperator D WHERE D.employee_id=%s",
(employee_id,))
```

30.

```
        curr.execute("SELECT D.name ,D.employee_id FROM dataentryoperator D
")
```

31.

```
        curr.execute(
            "DELETE FROM frontdeskoperator D WHERE D.employee_id=%s",
(employee_id,))
```

32.

```
 curr.execute("SELECT D.name,D.employee_id FROM frontdeskoperator D ")
```

33.

```
    curr.execute("SELECT D.name,D.employee_id FROM databaseadminstrator D ")
```

34.

```
    curr.execute("SELECT D.name,D.employee_id FROM physician D ")
```

35.

```
    curr.execute("SELECT D.name,D.employee_id FROM frontdeskoperator D ")
```

36.

```
   curr.execute("SELECT D.name,D.employee_id FROM dataentryoperator D ")
```

37.

```
   curr.execute("INSERT INTO
AppointmentRequests(name,SSN,email,appointmentdate,department,priority)
VALUES (%s,%s,%s,%s,%s,%s) ON CONFLICT DO NOTHING",
                (name, SSN, email, appointmentdate, department, priority))
```

38.

```
   curr.execute("INSERT INTO
test_undergoes(patient,test_id,date,physician,result) VALUES
(%s,%s,%s,%s,%s) ON CONFLICT DO NOTHING",
                (SSN, test_id, date, physician_id, result))
```

39.

```
   curr.execute("INSERT INTO
Undergoes(patient,procedure,room,date,physician,result) VALUES
(%s,%s,%s,%s,%s,%s) ON CONFLICT DO NOTHING",
                (patient_id, procedure_id, room, date, physician_id,
result,))
```

40.

```
   curr.execute("SELECT P.name,P.SSN,P.email,A.start FROM patient
P,Appointment A WHERE P.SSN=A.patient AND A.physician=%s AND A.start!=%s",
(doctor_id, today,))
```

41.

```
 curr.execute(
       "SELECT p.name,a.start from patient p,appointment a  where
p.SSN=a.patient AND a.start=%s", (today,))
```

42.

```
   curr.execute("SELECT * FROM onleave O Where O.physician=%s",
(doctor_id,))
```

43.

```
   curr.execute(
       "select P.SSN,P.name,P.email from patient P where P.SSN= %s",
(patient_id,))
```

44.

```
   curr.execute("select P2.name,M.name,M.brand,P1.dose,P1.date from
Prescribes P1,medication M,Physician P2 where P1.patient=%s and
M.code=P1.medication and P2.employee_id=P1.physician; ", (patient_id,))
```

45.

```
   curr.execute("select P.name,U.date,P.cost,P2.name from undergoes
U,procedure P,Physician P2 where U.patient=%s and P.code=U.procedure and
P2.employee_id=U.physician;", (patient_id,))
```

46.

```
       curr.execute(
           "SELECT * FROM databaseadminstrator D where D.employee_Id=%s AND
D.password=%s", (employee_id, password,))
```

47.

```
    curr.execute(
           "SELECT * FROM physician D WHERE D.employee_id=%s AND
D.password=%s", (employee_id, password,))
```

48.

```
  curr.execute(
               "SELECT P.name,P.SSN,P.email,A.start FROM patient
P,Appointment A WHERE P.SSN=A.patient AND A.physician=%s AND A.start!=%s",
(doctor_id, today,))
```

49.

```
curr.execute(
               "SELECT p.name,a.start from patient p,appointment a  where
p.SSN=a.patient AND a.start=%s", (today,))
```

50.

```
 curr.execute(
               "SELECT * FROM onleave O Where O.physician=%s", (doctor_id,))
```

51.

```
curr.execute(
           "SELECT * FROM dataentryoperator D WHERE D.employee_id=%s AND
D.password=%s", (employee_id, password,))
```

52.

```
       curr.execute(
           "SELECT * FROM frontdeskoperator D WHERE D.employee_id=%s AND
D.password=%s", (employee_id, password,))
```

53.

```
 curr.execute("INSERT INTO {}(name,employee_id,password,SSN) VALUES
(%s,%s,%s,%s) ON CONFLICT DO NOTHING".format(
       user_type), (name, employee_id, password, SSN))
```

54.

```
   curr.execute("INSERT INTO physician
(employee_id,name,SSN,password,department,email) VALUES (%s,%s,%s,%s,%s,%s)
ON CONFLICT DO NOTHING",
                (employee_id, name, SSN, password, department, email,))
```