Final Project Report – Hospital Portal Database

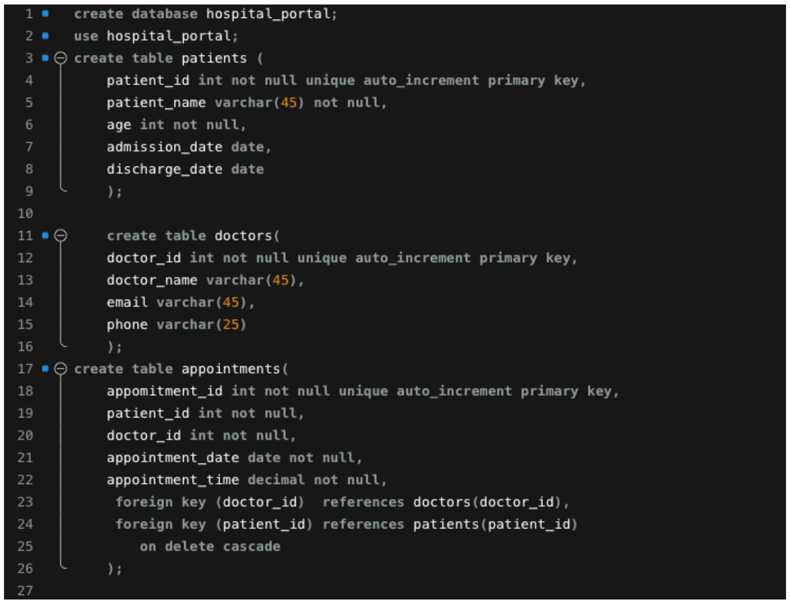
Ralston Nembhard

CIS 344 – Fall 2023

Professor Yanilda Peralta Ramos

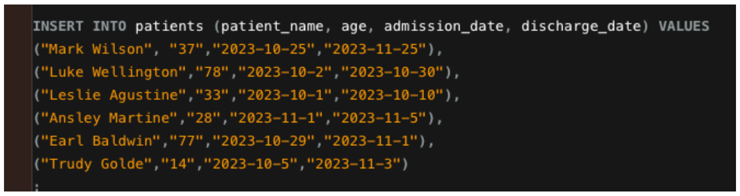
The final project required that students create a database scheme for a Hospital Portal, consisting of three tables, Patient, Doctor, and Appointment tables, and modify the starter Python code to interface with the database to allow for the management of hospital operations.

The creation of the database was pretty straightforward and was done with the “create database” and “create table” commands as shown below:



```
create database hospital_portal;
use hospital_portal;
create table patients (
    patient_id int not null unique auto_increment primary key,
    patient_name varchar(45) not null,
    age int not null,
    admission_date date,
    discharge_date date
    );

    create table doctors(
    doctor_id int not null unique auto_increment primary key,
    doctor_name varchar(45),
    email varchar(45),
    phone varchar(25)
    );
create table appointments(
    appomitment_id int not null unique auto_increment primary key,
    patient_id int not null,
    doctor_id int not null,
    appointment_date date not null,
    appointment_time decimal not null,
     foreign key (doctor_id)  references doctors(doctor_id),
     foreign key (patient_id) references patients(patient_id)
        on delete cascade
    );
```

The next task given was to insert initial tuples into each table. Again this was done using the standard MySQL “insert into…” query command in MySQL Workbench. Example below.



```
INSERT INTO patients (patient_name, age, admission_date, discharge_date) VALUES
("Mark Wilson", "37","2023-10-25","2023-11-25"),
("Luke Wellington","78","2023-10-2","2023-10-30"),
("Leslie Agustine","33","2023-10-1","2023-10-10"),
("Ansley Martine","28","2023-11-1","2023-11-5"),
("Earl Baldwin","77","2023-10-29","2023-11-1"),
("Trudy Golde","14","2023-10-5","2023-11-3")
;
```

After the creation of tables and insertion of records, it was time to move on to the major problem that needed to be solved: Modifying the PortalDatbase.py and PortalServer.py Python starter code to accomplish the given tasks: to interact with the database server to facilitate hospital operation (patient management.)

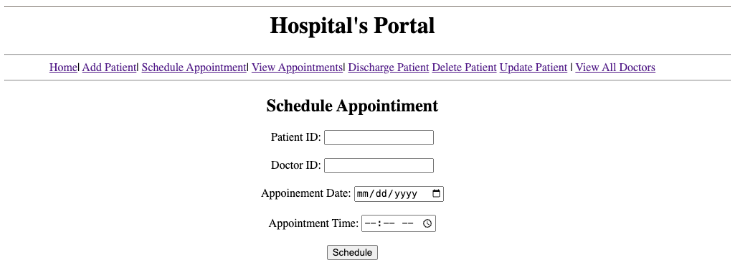
Two initial tasks for patient management, View Patient (Home) and Add Patient, as well as the initial menu options were already completed and students were expected to use those as a template to complete the remainder of the project.

When a menu option in the website is selected, the do_GET method of the portalServer is called. This method allows the user to enter data into the webform, for example, entering patient details. The do_POST method is then called from the do_GET method, this method reads the form data into variables before passing these variables to and calling specific methods in the portalDatabase class.

My approach to the project was to use stored procedures as much as possible to facilitate database updates. Doing this would eliminate the need to write complex queries directly in the Database class, and instead call the stored procedures by passing the necessary parameters.

Schedule Appointment

My ScheduleAppointment stored procedure accepts patient_id, doctor_id, appoinemtnet_date and appointment_time. These parameters are entered by the user through the do_GET method of the portalServer. The do_GET method calls the do_POST method which reads the data from the form and then calls the ScheduleAppointment stored procedure in the portalDatabase class, passing to it the values entered by the user. The stored procedure then updates the appointment table with the entered values. All menu options requiring database updates follow this principle.


Hospital's Portal

Home| Add Patient| Schedule Appointment| View Appointments| Discharge Patient Delete Patient Update Patient | View All Doctors

Schedule Appointment

Patient ID:

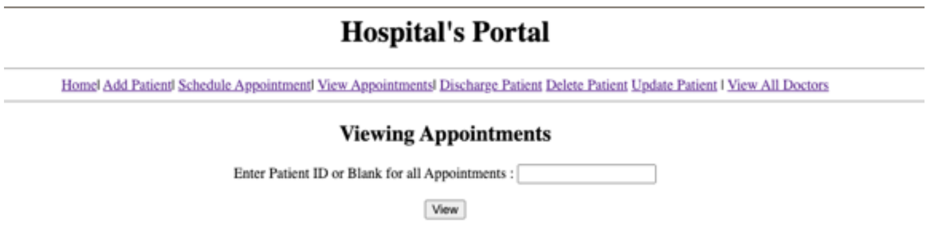
Doctor ID:

Appoinement Date: mm/dd/yyyy

Appointment Time: --:-- --

View Appointments.

The view appointments module allows the user to view appointments for a single patient or all patients, based on the value entered in the patient ID field. If a patient ID is entered, any appointment on file for that patient will be displayed. However, if the patient ID field is left blank, all appointments for all patients are displayed. This was accomplished by creating a stored procedure that created a join of the Patients, the doctors, and the appointments table, then using the WHERE clause to filter for the entered patient, or to view the entire joined data if no patient ID was specified.

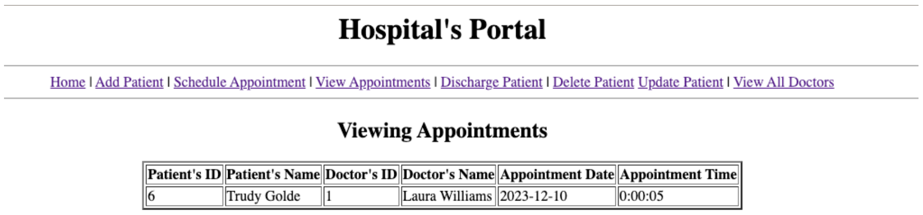

Home| Add Patient| Schedule Appointment| View Appointments| Discharge Patient Delete Patient Update Patient | View All Doctors
View

Hospital's Portal

Viewing Appointments

Enter Patient ID or Blank for all Appointments :

If the user enters a patient’s ID in the field, the following is displayed.

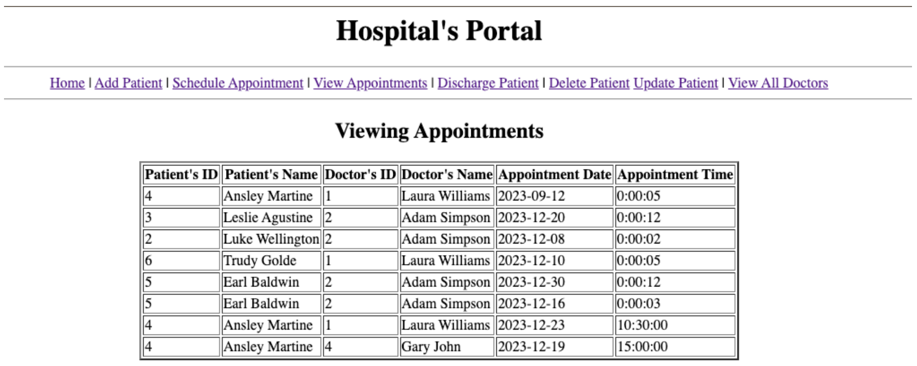

Home | Add Patient | Schedule Appointment | View Appointments | Discharge Patient | Delete Patient Update Patient | View All Doctors

Hospital's Portal

Viewing Appointments

Patient's ID	Patient's Name	Doctor's ID	Doctor's Name	Appointment Date	Appointment Time
6	Trudy Golde	1	Laura Williams	2023-12-10	0:00:05

If the user leaves the patient ID field blank the following is displayed


Home | Add Patient | Schedule Appointment | View Appointments | Discharge Patient | Delete Patient Update Patient | View All Doctors

Hospital's Portal

Viewing Appointments

Patient's ID	Patient's Name	Doctor's ID	Doctor's Name	Appointment Date	Appointment Time
4	Ansley Martine	1	Laura Williams	2023-09-12	0:00:05
3	Leslie Agustine	2	Adam Simpson	2023-12-20	0:00:12
2	Luke Wellington	2	Adam Simpson	2023-12-08	0:00:02
6	Trudy Golde	1	Laura Williams	2023-12-10	0:00:05
5	Earl Baldwin	2	Adam Simpson	2023-12-30	0:00:12
5	Earl Baldwin	2	Adam Simpson	2023-12-16	0:00:03
4	Ansley Martine	1	Laura Williams	2023-12-23	10:30:00
4	Ansley Martine	4	Gary John	2023-12-19	15:00:00

<u>Update Patient Details</u>

In working on the Update Patient Detail option, I wanted to allow the user to enter only those pieces of data that needed to be entered or corrected, leaving the other fields blank. The stored procedure would then check the data and update the table with non-blank fields that were passed to it using the IFNULL function. This proved to be problematic for me as I kept getting a type error whenever I left the Patient age field blank. Researching the error messages, I found I was converting a noneType to an integer, and that was illegal.

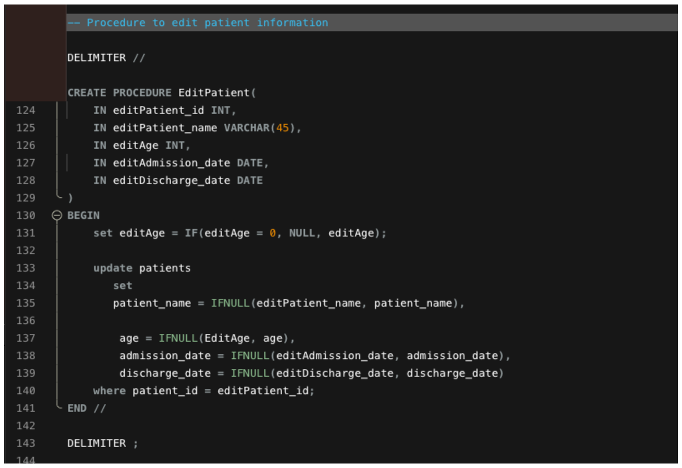
```
(TypeError: int() argument must be a string, a bytes-like object or a
real number, not 'NoneType')
```

The below logic was used to resolve the issue in the code and the EditPatient stored procedure. In PortalServer, I checked if the patient_age variable was returning a None value and if so, assigned the age variable a value of zero (0), if not I converted the number value to an integer and assigned this value to the age variable, so that an integer would be passed to the stored procedure in either case.

PortalServer:
```
if form.getvalue("patient_age") is not None:
        age = int(form.getvalue("patient_age"))
else:
        age = 0
```

MySQL Workbench: in the workbench, I edited the Edited the stored procedure to check if the EditAge variable was equal to zero and if so, converted it to null before updating the table, since the procedure checks for null variables and retains old values if found.

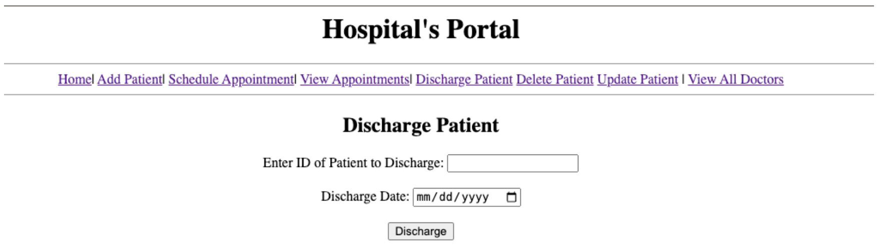
```
SET editAge = IF(editAge = 0, NULL, editAge);
```



```
-- Procedure to edit patient information

DELIMITER //

CREATE PROCEDURE EditPatient(
    IN editPatient_id INT,
    IN editPatient_name VARCHAR(45),
    IN editAge INT,
    IN editAdmission_date DATE,
    IN editDischarge_date DATE
)
BEGIN
    set editAge = IF(editAge = 0, NULL, editAge);

    update patients
       set
       patient_name = IFNULL(editPatient_name, patient_name),

        age = IFNULL(EditAge, age),
        admission_date = IFNULL(editAdmission_date, admission_date),
        discharge_date = IFNULL(editDischarge_date, discharge_date)
    where patient_id = editPatient_id;
END //

DELIMITER ;
```

Discharge Patient

The discharge patient option allows the user to enter a discharge date into the patient's record. When the patient ID and discharge date are entered, the discharge date for the patient tuple identified by the entered patient ID is updated. The DischargePatient stored procedure used the alter table command to accomplish this task.

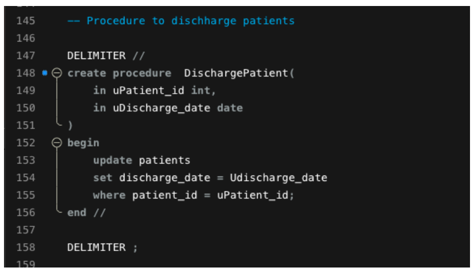

Hospital's Portal

Home| Add Patient| Schedule Appointment| View Appointments| Discharge Patient Delete Patient Update Patient | View All Doctors

Discharge Patient

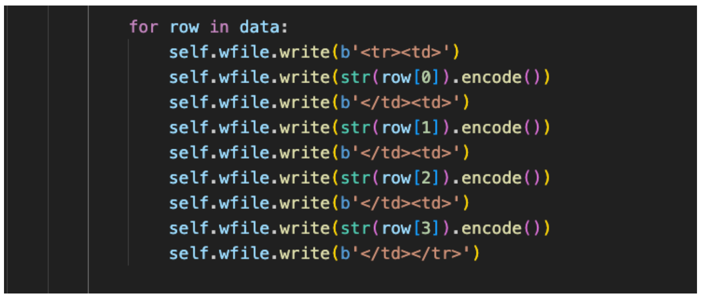
Enter ID of Patient to Discharge:

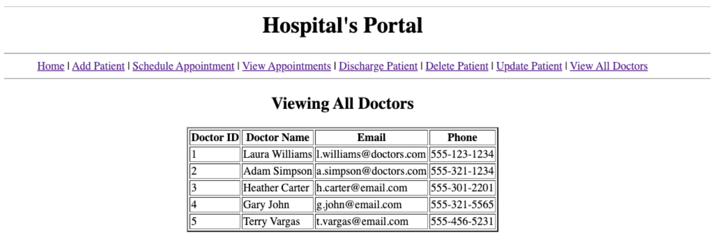
Discharge Date: mm/dd/yyyy


```
-- Procedure to dischharge patients

DELIMITER //
create procedure  DischargePatient(
    in uPatient_id int,
    in uDischarge_date date
)
begin
    update patients
    set discharge_date = Udischarge_date
    where patient_id = uPatient_id;
end //

DELIMITER ;
```

View All Doctors

The View All Doctors option is identical to the View All Patients option, in that it simply reads and displays all the tuples in the doctor table. This option calls the viewAllDoctors method in portalDatabase. The method simply issues the “Select * from doctors” query. The result is passed back to the portalServer where it is stored in an array, data. The array is then read and displayed with each element, from 0 to n, representing a displayed column.


```
for row in data:
    self.wfile.write(b'<tr><td>')
    self.wfile.write(str(row[0]).encode())
    self.wfile.write(b'</td><td>')
    self.wfile.write(str(row[1]).encode())
    self.wfile.write(b'</td><td>')
    self.wfile.write(str(row[2]).encode())
    self.wfile.write(b'</td><td>')
    self.wfile.write(str(row[3]).encode())
    self.wfile.write(b'</td></tr>')
```


Hospital's Portal

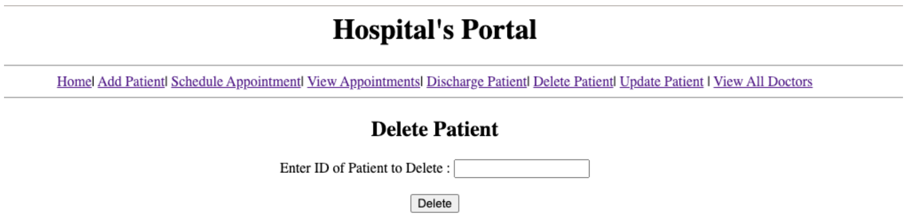
Home | Add Patient | Schedule Appointment | View Appointments | Discharge Patient | Delete Patient | Update Patient | View All Doctors

Viewing All Doctors

Doctor ID	Doctor Name	Email	Phone
1	Laura Williams	l.williams@doctors.com	555-123-1234
2	Adam Simpson	a.simpson@doctors.com	555-321-1234
3	Heather Carter	h.carter@email.com	555-301-2201
4	Gary John	g.john@email.com	555-321-5565
5	Terry Vargas	t.vargas@email.com	555-456-5231

Delete Patient

I added a Delete Patient option to demonstrate foreign key constraints.
This option checks for and deletes patients' tuples using the entered patient ID. When creating the tables, a foreign key constraint was placed on the patient ID in the appointments table. The constraint, "…on delete cascade" allows for the deletion of all related tuples in the appointments table when the parent patient ID is deleted for the patient's table. This way we prevent orphan appointment tuples from being left in the appointments table.


Hospital's Portal

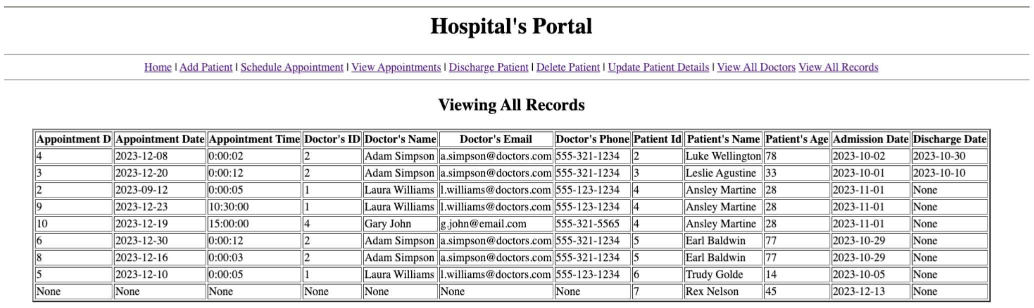
Home| Add Patient| Schedule Appointment| View Appointments| Discharge Patient| Delete Patient| Update Patient | View All Doctors

Delete Patient

Enter ID of Patient to Delete :

View All Records

The View All Records option uses a view, appointment_doctors_patients, containing of the left inner join of all tables in the database. Like the View All Doctors option, it stores data obtained from the query, "select * from appointment_doctors_patients" in an array, then displays the data from the array in column form.


Hospital's Portal

Home | Add Patient | Schedule Appointment | View Appointments | Discharge Patient | Delete Patient | Update Patient Details | View All Doctors View All Records

Viewing All Records

Appointment D	Appointment Date	Appointment Time	Doctor's ID	Doctor's Name	Doctor's Email	Doctor's Phone	Patient Id	Patient's Name	Patient's Age	Admission Date	Discharge Date
4	2023-12-08	0:00:02	2	Adam Simpson	a.simpson@doctors.com	555-321-1234	2	Luke Wellington	78	2023-10-02	2023-10-30
3	2023-12-20	0:00:12	2	Adam Simpson	a.simpson@doctors.com	555-321-1234	3	Leslie Agustine	33	2023-10-01	2023-10-10
2	2023-09-12	0:00:05	1	Laura Williams	l.williams@doctors.com	555-123-1234	4	Ansley Martine	28	2023-11-01	None
9	2023-12-23	10:30:00	1	Laura Williams	l.williams@doctors.com	555-123-1234	4	Ansley Martine	28	2023-11-01	None
10	2023-12-19	15:00:00	4	Gary John	g.john@email.com	555-321-5565	4	Ansley Martine	28	2023-11-01	None
6	2023-12-30	0:00:12	2	Adam Simpson	a.simpson@doctors.com	555-321-1234	5	Earl Baldwin	77	2023-10-29	None
8	2023-12-16	0:00:03	2	Adam Simpson	a.simpson@doctors.com	555-321-1234	5	Earl Baldwin	77	2023-10-29	None
5	2023-12-10	0:00:05	1	Laura Williams	l.williams@doctors.com	555-123-1234	6	Trudy Golde	14	2023-10-05	None
None	None	None	None	None	None	None	7	Rex Nelson	45	2023-12-13	None

====End===