James Aruz
12/18/23
CIS 344 Final Project Report

MySQL:

```
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 Appointments (
appointment_id int not null unique auto_increment primary key,
patient_id int not null references patients(patient_id),
doctor_id int not null references doctors(id),
appointment_date date not null,
appointment_time time not null
);
```

Starting off with the MySQL side of the project, I created the database “hospital_portal” and created 2 tables; patients and appointments. Each table is filled with the necessary attributes and foreign keys. I then added sample values into the patients table to test and made sure the table is initialized and visible shown here:

patient_id	patient_name	age	admission_date	discharge_date
1	Maria Jozef	67	2023-10-01	2023-12-18
2	Steve Wonder	42	2023-10-08	2023-12-18
3	Tony Hawk	56	2023-10-16	2023-12-18
4	Clark Kent	47	2023-10-16	2023-12-18
5	Wally West	29	2023-12-15	2023-12-18
6	Harley Quinn	38	2024-01-03	2024-01-10
7	Peter Parker	26	2023-12-20	2023-12-27
8	Eddie Brock	34	2023-12-21	2023-12-28
9	Tony Stark	51	2023-12-29	2024-01-24

Next I made a stored procedure to create appointments for patients with doctors and a date and time of the appointment:

```
Delimiter //
Create Procedure App_Schedule (
in appointment_patid int,
in appointment_docid int,
in appointment_scheduledate date,
in appointment_scheduletime time
)

Begin
 Insert into Appointments (patient_id, doctor_id, appointment_date, appointment_time)
 values (appointment_patid, appointment_docid, appointment_scheduledate, appointment_scheduletime);

 end //
Delimiter ;
```

I inserted variables into the procedure so when the procedure is called it will take the information and send it into the appointments table. After Begin, I insert the values from the procedure into the Appointments table. This allows the user to call the procedure, insert the necessary information and the information gets passed and stored into the appointments table.

Next, I made a stored procedure to discharge patients:

```
Delimiter //
Create Procedure discharge_patients()
Begin
 UPDATE patients
 SET discharge_date = current_date
 WHERE patient_id = 5;
end //
Delimiter ;
```

By using UPDATE for the patients table I can change the discharge_date attribute and set it to the current date that the patient discharges from the hospital. Using the WHERE clause I can specifically input the patient_id that is being discharged and the table will update the discharge_date to the date that they are being discharged and it will show in the table.
In this case, patient_id 5 Wally West is getting discharged so I input his id into the procedure and call it and this is the result:

patient_id	patient_name	age	admission_date	discharge_date
5	Wally West	29	2023-12-15	2023-12-18

After calling the procedure the date has change to today's current date and Mr. West is officially discharged.
Next, I created a table for doctors and like the patients and appointments table filled the table with the necessary attributes and filled the table with sample data and made sure the table was initialized and visible.

Lastly on the MySQL side of things, I created a view containing the join of patients, appointments, and doctors shown here:

```
'   CREATE VIEW d_a_p_veiw AS
    SELECT
    Appointments.appointment_id,
    Appointments.appointment_date,
    Appointments.appointment_time,
    patients.patient_id,
    patients.age,
    patients.admission_date,
    patients.discharge_date,
    doctors.d_name,
    doctors.specialization
    FROM appointments Appointments
    join patients patients ON Appointments.patient_id = patients.patient_id
    JOIN doctors doctors on Appointments.doctor_id = doctors.doc_id;
```

Python:
I started off by modifying the portaldatabase.py file with the appropriate MySQL Credentials:

```
class Database():
    def __init__(self,
                 host="localhost",
                 port="3306",
                 database="hospital_portal",
                 user='root',
                 password='Steven1200@'):
```

To first communicate with MySQL, I used the same port number database that's in MySQL so python can retrieve the information.

```python
def addPatient(self, patient_name, age, admission_date, discharge_date):
    ''' Method to insert a new patient into the patients table '''
    if self.connection.is_connected():
        self.cursor = self.connection.cursor()
        query = "INSERT INTO patients (patient_name, age, admission_date, discharge_date) VALUES (%s, %s, %s, %s)"
        self.cursor.execute(query, (patient_name, age, admission_date, discharge_date))
        self.connection.commit()
        return
```

Next, I added methods in python to do various tasks such as adding and viewing patients, Adding and viewing appointments and discharging patients. Each method in this portion of the python code is similar except for changing the “query” portion to each method's respective statements that correspond to the statements in MySQL.

Now in the portalserver.py file
Mainly I called the methods from the portaldatabase.py and made sure it was visible on the local host web page.

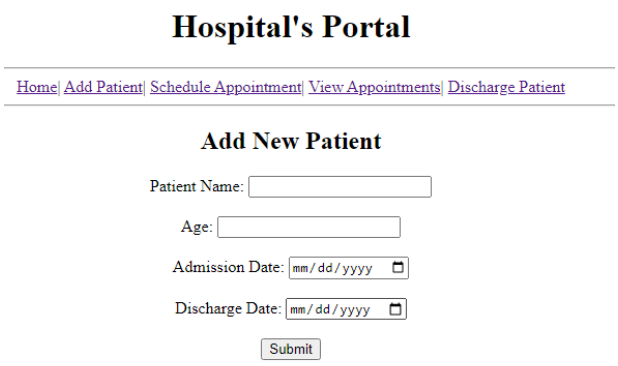
For adding patients, in do_POST I initialize the variables from the patients table then called the method to add a new patient from the database file.

```python
def do_POST(self):
    try:
        if self.path == '/addPatient':
            self.send_response(200)
            self.send_header('Content-type','text/html')
            self.end_headers()
            form = cgi.FieldStorage(
                fp=self.rfile,
                headers=self.headers,
                environ={'REQUEST_METHOD': 'POST'}
            )

            patient_name = form.getvalue("patient_name")
            age = int(form.getvalue("patient_age"))
            admission_date = form.getvalue("admission_date")
            discharge_date = form.getvalue("discharge_date")
            # Call the Database Method to add a new patient. Attention please read this comment!!! See Example call!

            self.database.addPatient(patient_name, age, admission_date,discharge_date)


            print("Patient added:", patient_name, age, admission_date)
```

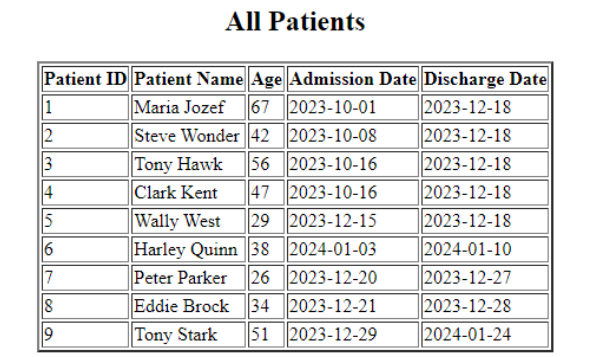
To create the web page itself, I added the necessary fields to input the information shown here.

```
##addPatient Implemented : Complete code in do_POST /addPatient, Read comment in do_POST
if self.path == '/addPatient':
    self.send_response(200)
    self.send_header('Content-type','text/html')
    self.end_headers()
    self.wfile.write(b"<html><head><title> Hospital's Portal </title></head>")
    self.wfile.write(b"<body>")
    self.wfile.write(b"<center><h1>Hospital's Portal</h1>")
    self.wfile.write(b"<hr>")
    self.wfile.write(b"<div> <a href='/'>Home</a>| \
                      <a href='/addPatient'>Add Patient</a>|\
                       <a href='/scheduleAppointment'>Schedule Appointment</a>|\
                       <a href='/viewAppointment'>View Appointments</a>|\
                       <a href='/dischargePatient'>Discharge Patient</a></div>")
    self.wfile.write(b"<hr><h2>Add New Patient</h2>")

    self.wfile.write(b"<form action='/addPatient' method='post'>")
    self.wfile.write(b'<label for="patient_name">Patient Name:</label>\
          <input type="text" id="patient_name" name="patient_name"/><br><br>\
          <label for="patient_age">Age:</label>\
          <input type="number" id="patient_age" name="patient_age"><br><br>\
          <label for="admission_date">Admission Date:</label>\
          <input type="date"id="admission_date" name="admission_date"><br><br>\
          <label for="discharge_date">Discharge Date:</label>\
          <input type="date"id="discharge_date" name="discharge_date"><br><br>\
          <input type="submit" value="Submit">\
          </form>')

    self.wfile.write(b"</center></body></html>")
    return
```

And this is the result on the webpage:



I'm able to add new patients and after pressing on the home tab at the top I can view the entire patients table and information.

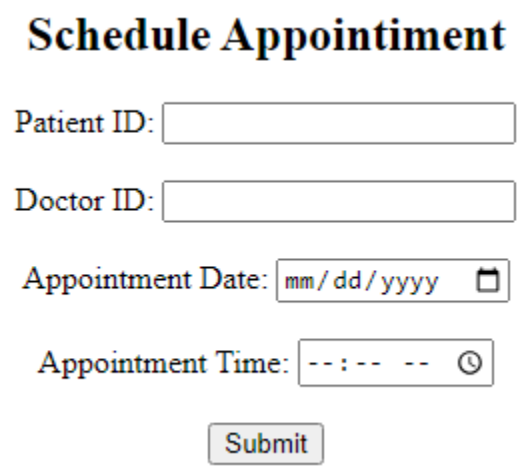


Patient ID	Patient Name	Age	Admission Date	Discharge Date
1	Maria Jozef	67	2023-10-01	2023-12-18
2	Steve Wonder	42	2023-10-08	2023-12-18
3	Tony Hawk	56	2023-10-16	2023-12-18
4	Clark Kent	47	2023-10-16	2023-12-18
5	Wally West	29	2023-12-15	2023-12-18
6	Harley Quinn	38	2024-01-03	2024-01-10
7	Peter Parker	26	2023-12-20	2023-12-27
8	Eddie Brock	34	2023-12-21	2023-12-28
9	Tony Stark	51	2023-12-29	2024-01-24

Like adding patients, in do_POST I initialized the necessary variables from the appointments table and then called the method from the database file.

```
if self.path == '/scheduleAppointment':
    self.send_response(200)
    self.send_header('Content-type','text/html')
    self.end_headers()
    form = cgi.FieldStorage(
        fp=self.rfile,
        headers=self.headers,
        environ={'REQUEST_METHOD': 'POST'}
    )
    patient_id = int(form.getvalue("patient_id"))
    doctor_id = int(form.getvalue("doctor_id"))
    appointment_date = form.getvalue("appointment_date")
    appointment_time = form.getvalue("appointment_time")

    self.database.scheduleAppointment(patient_id, doctor_id, appointment_date, appointment_time

    print("Appointment added:", patient_id, doctor_id, appointment_date, appointment_time)
```

Like adding patients, I added the necessary fields to take in the information on the webpage. And this is what the web page looks like



To view the appointments table on the web page, I called the database method and printed it on the webpage. I then created a table to showcase the data inside the table using the template in viewing patients method in a similar manner.

```
data=[]
records = self.database.viewAppointments()
print(records)
data=records
self.send_response(200)
self.send_header('Content-type','text/html')
self.end_headers()

self.wfile.write(b"<hr><h2>All Appointments</h2>")
self.wfile.write(b"<table border=2> \
                    <tr><th> Appointment ID </th>\
                        <th> Patient ID </th>\
                        <th> Doctor ID </th>\
                        <th> Appointment Date </th>\
                        <th> Appointment Time </th></tr>")

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><td>')
    self.wfile.write(str(row[4]).encode())
    self.wfile.write(b'</td></tr>')


self.wfile.write(b"</table></center>")
self.wfile.write(b"</body></html>")
return
```

And this is the result:

All Appointments

Appointment ID	Patient ID	Doctor ID	Appointment Date	Appointment Time
1	1	2	2023-12-20	9:30:00
2	1	2	2023-12-20	9:30:00
3	1	2	2023-12-20	9:34:00
4	1	2	2023-12-20	6:39:00
5	1	2	2023-12-28	20:37:00
6	2	3	2023-12-29	10:47:00
7	5	2	2024-01-17	10:00:00
8	5	2	2024-01-17	10:00:00
9	6	1	2023-12-30	10:00:00
10	2	2	2023-12-25	13:35:00

Lastly for discharge patients on the table I called the method from the database file and intilized the patient_id variable.

```
if self.path == '/dischargePatient':
    self.send_response(200)
    self.send_header('Content-type','text/html')
    self.end_headers()
    form = cgi.FieldStorage(
        fp=self.rfile,
        headers=self.headers,
        environ={'REQUEST_METHOD': 'POST'}
    )

    patient_id = int(form.getvalue("patient_id"))
    self.database.dischargePatient(patient_id)


    print("Patient Discharged:", patient_id)
```

Then to view it on the web page I added a field to take an input for the patient's id.
And this is the result:

After inputting a Patient ID and submitting, the changes can be viewed on the home page on the patients table shown here:

Patient ID	Patient Name	Age	Admission Date	Discharge Date
1	Maria Jozef	67	2023-10-01	2023-12-18
2	Steve Wonder	42	2023-10-08	2023-12-18
3	Tony Hawk	56	2023-10-16	2023-12-18
4	Clark Kent	47	2023-10-16	2023-12-18
5	Wally West	29	2023-12-15	2023-12-18