Hendy Duran
Prof. Yanilda Peralta Ramos

12/11/2023
CIS 344

CIS344 Final Project Report Hospital Portal

The Database:

For the creation of the database I used the CREATE DATABASE hospital_portal followed by the USE hospital_database to ensure that is the database I am currently working with. Moving on, we can create 3 tables using the CREATE TABLE command. The patients table is made with patient_id, patient_name, age, admission_date, discharge_date as its attributes, making sure to make patient_id the PRIMARY KEY and setting it to AUTO_INCREASE which we will use all primary key ids in the rest of the database. Following is the doctors table with doctor_id, doctor_name, and doctor_field as the attributes, doctor_id being the PRIMARY_KEY. Finally, we make the appointments table with appointment_id (PRIMARY KEY), patient_id, doctor_id, appointment_date, and appointment_time, but this time making sure to set patient_id and doctor_id as FOREIGN KEYs and referencing them to their respective tables. Below are some images of how these tables were created and also originally populated before using the server to handle that moving forward.

```
34
35 • ALTER TABLE appointments
36   ADD FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id);
37
```

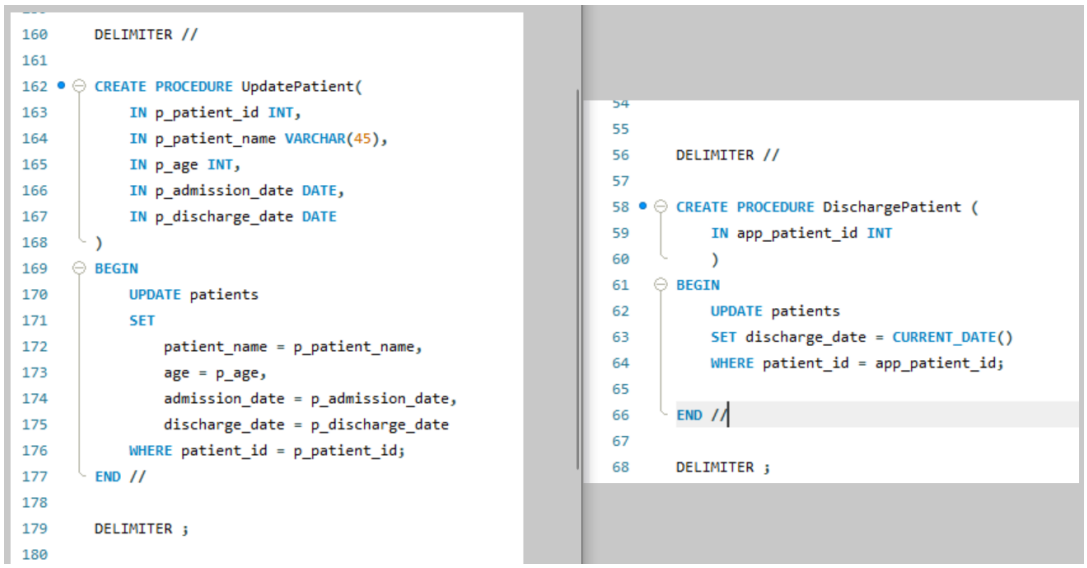
```
1 • CREATE DATABASE hospital_portal;
2 • USE hospital_portal;
3
4 • CREATE TABLE patients (
5       patient_id int not null unique auto_increment primary key,
6       patient_name varchar(45) NOT NULL,
7       age int not null,
8       admission_date date,
9       discharge_date date
10      );
11
12
13 • CREATE TABLE Appointments (
14      appointment_id int not null unique auto_increment primary key,
15      patient_id int not null,
16      doctor_id int not null,
17      appointment_date date not null,
18      appointment_time decimal not null,
19      foreign key (patient_id) references patients(patient_id)
20      );
21
22 • INSERT INTO patients (patient_name, age, admission_date, discharge_date)
23  VALUES ("Maria Jozef", 67, "2023/10/01", "2023/10/07"),
24         ("Breanna Olsen", 35, "2023/06/25", "2023/06/29"),
25         ("Henry Smith", 55, "2023/09/17", "2023/09/25");
26
27 • SELECT * FROM patients;
28
29 • CREATE TABLE doctors (
30      doctor_id int not null unique auto_increment primary key,
31      doctor_name varchar(45) NOT NULL,
32      doctor_field varchar(45) NOT NULL
33      );
34
```

```
69
70 • INSERT INTO doctors (doctor_name, doctor_field)
71  VALUES ("Steven Strange", "Neural Surgeon"),
72         ("Wanda Maximoff", "Pediatrics"),
73         ("Vladimir Tepes", "Cardiology");
```

Hendy Duran
Prof. Yanilda Peralta Ramos

12/11/2023
CIS 344

The next thing to do within the database is create a couple of stored procedures to facilitate certain actions, namely scheduleAppointments, discharePatient, and UpdatePatient. Using a DELIMITER and the CREATE PROCEDURE command, we begin by making ScheduleAppointment which will take four parameters (app_patient_id, app_doctor_id, app_appointment_date, app_appointment_time), using “app” in front of each parameter simply helps to differentiate it from the attribute they will be inserted into, which leads us to using INSERT INTO to insert said parameter value to each respective attribute within the appointments table. The DischargePatient procedure is much shorter and simpler than the last one as it only takes in one parameter (app_patient_id) to tell the UPDATE call WHERE it should be making the changes and SET the date to today. Leaving us with the call “UPDATE patients SET discharge_date = CURRENT_DATE() WHERE patient_id = app_patient_id;”. Finally I made an UpdatePatient procedure which took five parameters matching a patient entry set of attributes but this time using the patient_id to identify the patient to be updated and the rest of the parameters as the values to be changed to.



```
160    DELIMITER //
161
162 •  CREATE PROCEDURE UpdatePatient(
163        IN p_patient_id INT,
164        IN p_patient_name VARCHAR(45),
165        IN p_age INT,
166        IN p_admission_date DATE,
167        IN p_discharge_date DATE
168    )
169    BEGIN
170        UPDATE patients
171        SET
172            patient_name = p_patient_name,
173            age = p_age,
174            admission_date = p_admission_date,
175            discharge_date = p_discharge_date
176        WHERE patient_id = p_patient_id;
177    END //
178
179    DELIMITER ;
180
```

```
54
55
56    DELIMITER //
57
58 •  CREATE PROCEDURE DischargePatient (
59        IN app_patient_id INT
60        )
61    BEGIN
62        UPDATE patients
63        SET discharge_date = CURRENT_DATE()
64        WHERE patient_id = app_patient_id;
65
66    END //
67
68    DELIMITER ;
```

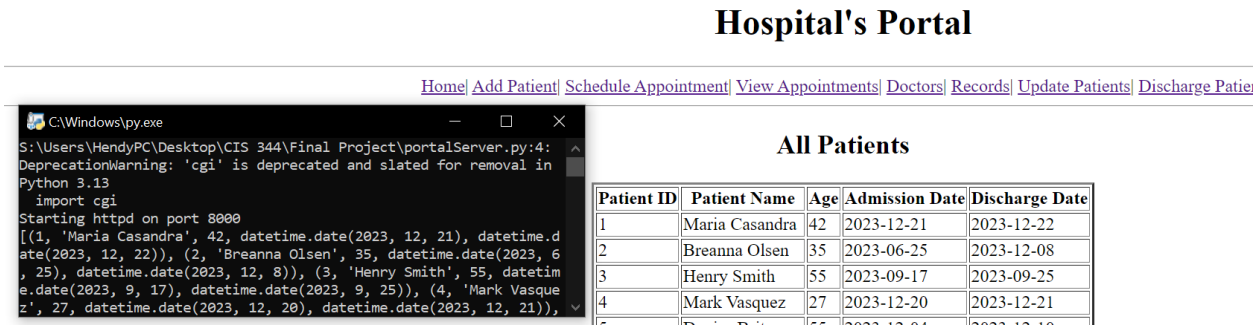
```
160    DELIMITER //
161
162 •  CREATE PROCEDURE UpdatePatient(
163        IN p_patient_id INT,
164        IN p_patient_name VARCHAR(45),
165        IN p_age INT,
166        IN p_admission_date DATE,
167        IN p_discharge_date DATE
168    )
169    BEGIN
170        UPDATE patients
171        SET
172            patient_name = p_patient_name,
173            age = p_age,
174            admission_date = p_admission_date,
175            discharge_date = p_discharge_date
176        WHERE patient_id = p_patient_id;
177    END //
178
179    DELIMITER ;
```

Hendy Duran
Prof. Yanilda Peralta Ramos

12/11/2023
CIS 344

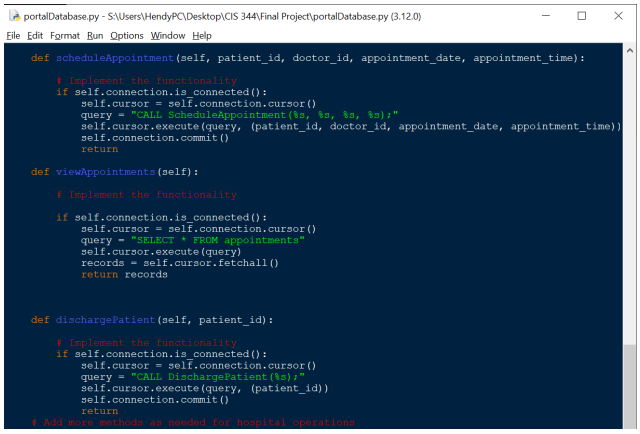
Connecting Database to Server:

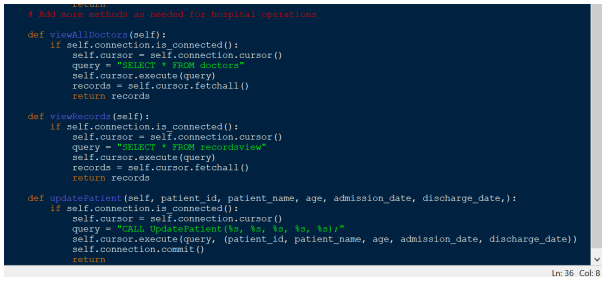
Here I need to first download and install mysql.connector in order to connect the database to our python code. Then changing the password from portalDatabase.py to my database password I manage to make a successful connection allowing the portal to run on my browser pulling data from the database.



The Server and Methods:

For the portalDatabase.py since the addPatient and getAllPateints methods were mostly completed for us I decided to use those as my template for the rest of my methods. I went on to implement the methods scheduleAppointment(), viewAppointments(), dischargePatient(), viewAllDoctors(), viewRecords(), and updatePatient() using the code from addPatient and getAllPatient as guideline only making sure to change the query to the appropriate queries. All code shown below:





Hendy Duran
Prof. Yanilda Peralta Ramos

12/11/2023
CIS 344

For the do_GET section of the server I needed to complete implementing the webpages for where the when the user would be taken to when they clicked any of the hyperlinks in the header as initially they were leading to blank pages. Since Home already showed us a list of all patients. I used that code and changed it to use the viewAppointment() method instead so it would pull data from the database from the appointments table and show a very similar list but displaying appointments instead of patients. The same was done for viewAllDoctors and viewRecords respectively.

```
#View Doctors code
if self.path == '/viewAllDoctors':
    data=[]
    records = self.database.viewAllDoctors()
    print(records)
    data=records
    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='/viewAppointments'>View Appointments</a>|\
                     <a href='/viewAllDoctors'>Doctors</a>|\
                     <a href='/viewRecords'>Records</a>|\
                     <a href='/updatePatient'>Update Patients</a>|\
                     <a href='/dischargePatient'>Discharge Patient</a></div>")
    self.wfile.write(b"<hr><h2>All Doctors</h2>")


    self.wfile.write(b"<table border=2> \
                        <tr><th> Doctor ID </th>\
                            <th> Doctor Name </th>\
                            <th> Medical Field </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></tr>')

    self.wfile.write(b"</center></body></html>")
    return
#End of view doctors
```

Hendy Duran
Prof. Yanilda Peralta Ramos

For the dischargePatient() and updatePatient() part of the I decided to have the forms to submit the data required along with a list of all the patients so that it would be easy to look at the list of patients when having to decide which patient needs discharging or updating as both of those methods only ask for an ID when choosing a patient and having the list of patients ID and the rest of their information on that same page would be of great help. Here is an example of that code:

```
        if self.path == '/updatePatient':
            data=[]
            records = self.database.getAllPatients()
            print(records)
            data=records
            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='/viewAppointments'>View Appointments</a>|\
                              <a href='/viewAllDoctors'>Doctors</a>|\
                              <a href='/viewRecords'>Records</a>|\
                              <a href='/updatePatient'>Update Patients</a>|\
                              <a href='/dischargePatient'>Discharge Patient</a></div>")

            self.wfile.write(b"<table border=2> \
                               <tr><th> Patient ID </th>\
                                   <th> Patient Name</th>\
                                   <th> Age </th>\
                                   <th> Admission Date </th>\
                                   <th> Discharge Date </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"<hr><h2>Update Patient</h2>")


            self.wfile.write(b"<form action='/updatePatient' method='post'>")
            self.wfile.write(b'<label for="patient_id">ID of patient to be updated:</label>\
                  <input type="number" id="patient_id" name="patient_id" required><br><br>\
                  <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"<br><br>")
            self.wfile.write(b"</center></body></html>")
            return

        #End Update Patient
```

Lastly for the portalSever.py file I used a similar strategy as I did for the portalDatabase code. Using the addPatient code already provided to us under do_POST as well as the one for the home screen which showed a list of all the patients I managed to create template code for the rest of the actions under do_POST. For example below you can see I used the template from addPatient to make a post screen for when an appointment is made, a patient is discharged, or a patient's details are updated confirming these actions were successful and allowing the option of repeating said action or visiting another part of the portal.

```
        #Update Patient code start

        try:
            if self.path == '/updatePatient':
                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"))
                patient_name = form.getvalue("patient_name")
                age = int(form.getvalue("patient_age"))
                admission_date = form.getvalue("admission_date")
                discharge_date = form.getvalue("discharge_date")

                self.database.updatePatient(patient_id, patient_name, age, admission_date, discharge_date)

                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='/viewAppointments'>View Appointments</a>|\
                                 <a href='/viewAllDoctors'>Doctors</a>|\
                                 <a href='/viewRecords'>Records</a>|\
                                 <a href='/updatePatient'>Update Patients</a>|\
                                 <a href='/dischargePatient'>Discharge Patient</a></div>")
                self.wfile.write(b"<hr>")
                self.wfile.write(b"<h3>Patient information updated</h3>")
                self.wfile.write(b"<div><a href='/updatePatient'>Update Another Patient</a></div>")
                self.wfile.write(b"</center></body></html>")

        except IOError:
            self.send_error(404, 'File Not Found: %s' % self.path)


        #Update Patient code ends

        return
```

Github Repository Link:
https://github.com/HendyDuranHD/CIS344-Final-Project-Hospital-Portal