

Healthcare United Patient Portal Electronic Health Record System

Rebecca Hassett, 110809470
Thomas Vetere, 110236188
Daeun Park, 111255579

Github Link: https://github.com/daeunnpark/hospital_portal

1.1 Project Scope

The Healthcare United Patient Portal is a database application that keeps doctors, nurses, patients, and department administrators connected in the healthcare community to ensure that patient care is personalized and efficient. Patients interact with the portal for scheduling appointments, validating medical history, updating their profile, and paying their bills. Nurses use the portal to collect important patient information based on their visit. Doctors use the portal to analyze patient medical history to determine appropriate treatment. Department administrators work with the portal to set up appointments and handle patient financial information concerning their medical bills and insurance. The Healthcare United Patient Portal prides itself on keeping patient information secure, and enabling doctors and nurses to provide patients with premier care through maintaining their personal health information in real time.

1.2 System Users

1. Patients
2. Doctors
3. Nurses
4. Department Administrators

Assume users have minimal experience using computers to ensure that the system is interactive and easy to operate.

2.0 Data

Specific Entity Types included in the system: Patient, Doctor, Nurse, Department Administrator and Appointment. Nurse, Doctor, and Department Administrator are subtypes of the Employee entity type. Patient and Employee are subtypes of the Person entity type.

2.1 Person Data

Attributes included:

1. Person ID (Key)
2. First Name
3. Last Name
4. Username
5. Password
6. Email Address
7. Phone Number

2.2 Employee Data

Attributes included:

1. Department ID
2. Availability

2.3 Doctor Data

Attributes included:

1. Doctor ID (Key)
2. Speciality
3. Medical License

4. Insurance List

2.4 Nurse Data

Attributes included:

1. Nurse ID (Key)
2. Speciality
3. Medical License

2.5 Department Administrator Data

Attributes included:

1. Department Administrator ID (Key)
2. Security Code
3. Payment List
4. Insurance List

2.6 Patient Data

Attributes included:

1. Patient ID (Key)
2. Address
3. Social Security Number
4. Credit Card Number
5. Age
6. Height
7. Weight
8. Doctor ID

3.0 Relationships

The relationship types between the entities include:

1. Employee and Patient IsA Person
2. Doctor, Nurse and Department Administrator IsA Employee
3. Doctor and Nurse and Patient Attends Appointment
4. Department Administrator Schedules Appointment
5. Doctor Reports to Insurance
6. Department Administrator Makes a Claim to Insurance
7. Patient is Covered by Insurance

3.1 Appointment

Attributes included:

1. Appointment ID (Key)
2. Doctor ID
3. Nurse ID
4. Patient ID
5. Date
6. Start Time
7. End Time
8. Location

3.2 Insurance

1. Insurance ID (Key)
2. Insurance Company Name
3. Appointment Cost
4. Cardholder Name
5. Cardholder ID

4.0 User Level Transactions

4.1 Patient Level Interactions:

A patient should be able to:

- View their personal medical history
- Update only their personal information (credit card number, insurance ID, email address, and phone number)
- Schedule an appointment
- View their billing

4.2 Doctor Level Interactions:

A doctor should be able to:

- View patient medical history and information
- Update patient medical history and information
- View the appointments that they must attend
- Able to request billing of a patient's insurance
- View their own personal information
- Update their own information
- Cannot view extremely sensitive information (SSN and Security Key)
- Can only view and update patients under their care

4.3 Nurse Level Interactions:

A nurse should be able to:

- Update patient medical history and information
- View the appointments that they must attend
- Update their own personal information
- View patient medical history and information
- Cannot view extremely sensitive information (SSN and Security Key)

- Can only view and update patient's under their care

4.4 Department Administrator Level Interactions:

A department Administrator should be able to:

- Schedule appointment availabilities
- View patient credit card information and patient insurance information to handle billing transactions with insurance company
- Can put a billing claim to a patient's insurance
- Has highest security clearance - can see all information for anyone within their department

5.0 Assumptions for Database System:

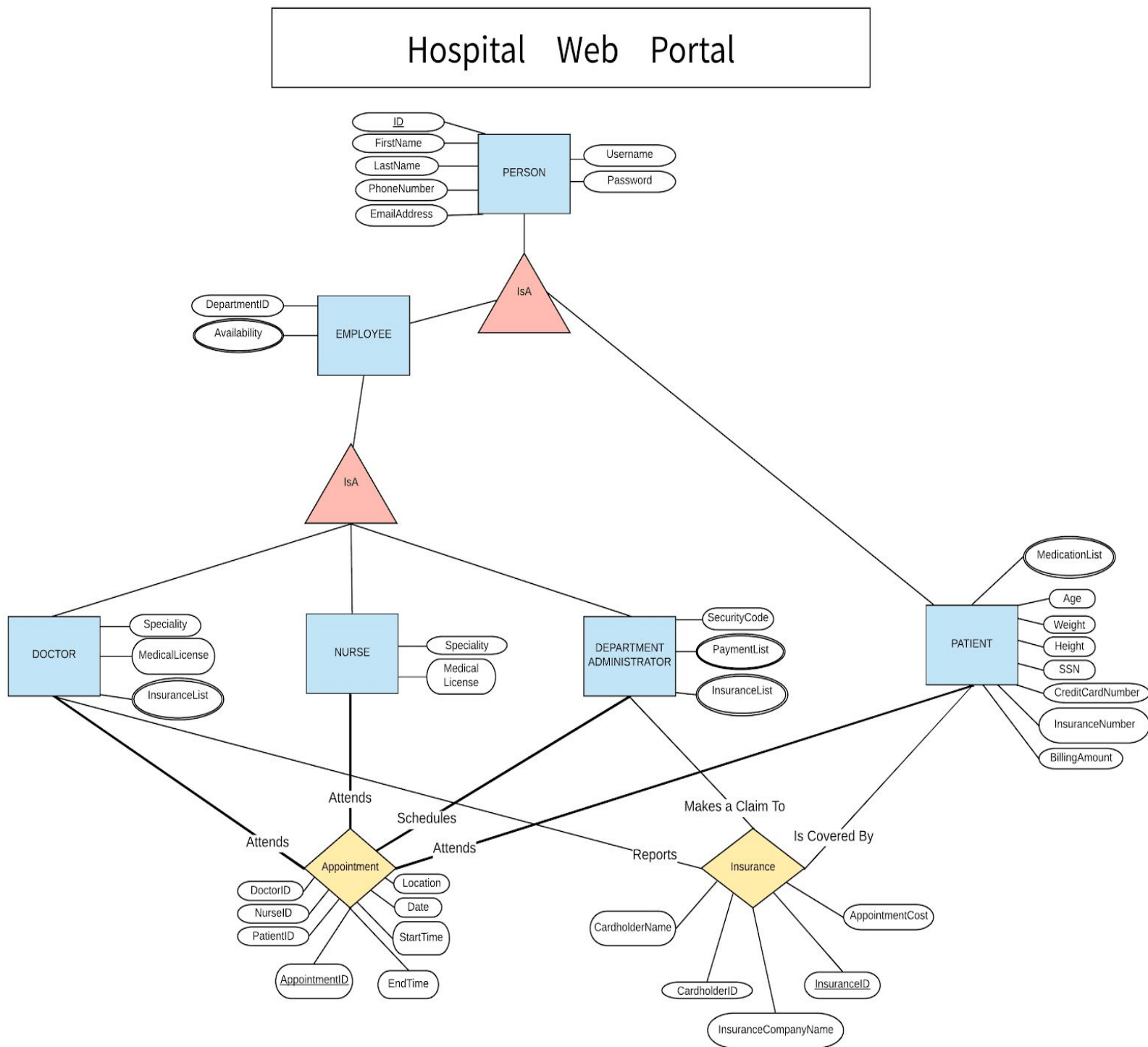
- Every patient and doctor has only one insurance plan
- Every patient is covered by some insurance
- Every patient has at least one scheduled appointment in the database at any particular time
- Employee(doctor, nurse and department administrator) and patient can not be the same person
- Patient cannot be an employee
- Employee cannot be a patient

6.0 ER Diagram Tools

The ER Diagrams are designed in the Chen style using Lucidchart tools

6.1 Entity-Relationship Diagram

7.0 Mapping E.R Diagram to SQL Tables:



1. SQL statements

```
CREATE TABLE hospital.Person (  
    ID INT UNSIGNED NOT NULL,  
    FirstName VARCHAR(20) NOT NULL,  
    LastName VARCHAR(20) NOT NULL,  
    PhoneNumber CHAR(10) NOT NULL,  
    EmailAddress VARCHAR(30),  
    Username VARCHAR(20) NOT NULL,  
    UserPassword VARCHAR(20) NOT NULL,  
    PRIMARY KEY(ID),  
    UNIQUE(ID, Username)  
)
```

```
CREATE TABLE hospital.Employee (  
    EmployeeID INT UNSIGNED NOT NULL,  
    DepartmentID TINYINT UNSIGNED NOT NULL,  
    PRIMARY KEY(EmployeeID),  
    FOREIGN KEY(EmployeeID) REFERENCES Person(ID)  
)
```

```
CREATE TABLE Doctor (  
    DoctorID INT UNSIGNED NOT NULL,  
    Specialty VARCHAR(20),  
    MedicalLicense VARCHAR(20) NOT NULL,  
    FOREIGN KEY(DoctorID) REFERENCES Employee(EmployeeID)  
)
```

```
CREATE TABLE Nurse (  
    NurseID INT UNSIGNED NOT NULL,  
    Specialty VARCHAR(20),  
    MedicalLicense VARCHAR(20) NOT NULL,  
    FOREIGN KEY (NurseID) REFERENCES Employee(EmployeeID)  
)
```



```
CREATE TABLE DepartmentAdmin (  
    SecurityCode INT UNSIGNED NOT NULL,  
    DepartmentAdminID INT UNSIGNED NOT NULL,  
    FOREIGN KEY (DepartmentAdminID) REFERENCES  
Employee(EmployeeID)  
)
```

```
CREATE TABLE Patient (  
    PatientID INT UNSIGNED NOT NULL,  
    Age INT UNSIGNED NOT NULL,  
    Weight FLOAT(4, 1) UNSIGNED NOT NULL,  
    Height FLOAT(3, 1) UNSIGNED NOT NULL,  
    SSN CHAR(9),  
    CreditCardNumber VARCHAR(20) NOT NULL,  
    BillingAmount FLOAT(10,2) UNSIGNED,  
    InsuranceNumber INT UNSIGNED NOT NULL,  
    MedicationList VARCHAR(20),  
    FOREIGN KEY(PatientID) REFERENCES Person(ID),  
    CHECK(Weight > 0),  
    CHECK(Height > 0)  
)
```

```
CREATE TABLE Insurance(  
    InsuranceID INT UNSIGNED NOT NULL,  
    InsuranceCompanyName VARCHAR(20) NOT NULL,  
    CardholderID INT UNSIGNED NOT NULL,  
    CardholderName VARCHAR(20) NOT NULL,  
    AppointmentCost FLOAT(10,2) UNSIGNED NOT NULL,  
    DepartmentAdminID INT UNSIGNED NOT NULL,  
    Primary Key(InsuranceID),  
    FOREIGN KEY(CardHolderID) REFERENCES Patient(PatientID),  
    FOREIGN KEY(DepartmentAdminID) REFERENCES  
DepartmentAdmin(DepartmentAdminID),
```

```

        UNIQUE (InsuranceID)
    )

CREATE TABLE Appointment (
    AppointmentID INT UNSIGNED NOT NULL,
    DoctorID INT UNSIGNED NOT NULL,
    NurseID INT UNSIGNED NOT NULL,
    PatientID INT UNSIGNED NOT NULL,
    DepartmentAdminID INT UNSIGNED NOT NULL,
    Location VARCHAR(20),
    Date DATE,
    StartTime TIME,
    EndTime TIME,
    PRIMARY KEY(AppointmentID),
    FOREIGN KEY(DoctorID) REFERENCES Doctor(DoctorID),
    FOREIGN KEY(NurseID) REFERENCES Nurse(NurseID),
    FOREIGN KEY(PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY(DepartmentAdminID) REFERENCES
DepartmentAdmin(DepartmentAdminID),
    UNIQUE(AppointmentID)
)

```

```

CREATE TABLE AccessCodes (
    AccessCodes INT UNSIGNED NOT NULL
)

```

SQL Language: MariaDB 10.3.9

2. Report

The above SQL statements were generated after mapping the ER diagram created in the previous document(Assignment #1, attached at the bottom) to a relational schema.

The choice was made to set ID attributes to INT UNSIGNED 0 to 4294967295 because we are not allowing negative ID numbers and we do not expect more than 4 billion users. The PersonID, EmployeeID, DoctorID, NurseID, PatientID, DepartmentAdminID, AppointmentID, InsuranceID, CardholderID are the ID values set as INT UNSIGNED. Attributes with decimal values are specified as FLOAT(m, n) where m is the total number of digits in the number and n is the number of digits after the decimal. We use TINYINT UNSIGNED for DepartmentID because we do not expect more than 256 departments. The ID and Username values are also set to UNIQUE since they must be unique.

PhoneNumber and SSN is CHAR(n) because every phone number and SSN must be exactly n characters long, not including dashes between numbers. We set strings to VARCHAR(n) when we do not know the exact string length, but we want to set the maximum string length to n. For example, FirstName can be a different length for different users, but we assume that there is no user with a FirstName greater than 20 characters. FirstName and LastName are set to NOT NULL because each user is expected to have a first and last name.

The Date attribute for Appointment uses the DATE data type. The StartTime and EndTime attributes for Appointment use the TIME data type. The PRIMARY KEY for each table is the ID value for that table because PRIMARY KEY must be a unique value and each ID must be unique. We set values to NOT NULL for required values that cannot be empty.

We include CHECK statements for Weight and Height since Weight and Height cannot be zero or negative. We use FOREIGN KEYS to signify relationships between entities. The attribute in the tuple that is REFERENCED is the primary key of that tuple.

The assumptions we made for the database system is that every person must have a person. Employees cannot be patients, and patients cannot be employees. Every employee belongs to exactly one department.

8.0 Transaction Implementations:

Environmental Information:

- Database: MariaDB
- Front-End Graphical User Interface Toolkit: PyQt
- Front-End Language: Python

Transactions Description:

- **Transaction #1: Authenticate User:**

- This transaction authenticates whether or not a user exists in the system based on the username and password that the user entered. It checks whether or not username and password exists in the PERSON database table using a SELECT statement. If user exists and user is patient, the system checks whether or not ID exists in patient using SELECT statement. If user exists and user is doctor, the system checks whether or not ID exists in doctor using SELECT statement. If user is nurse, the system checks whether or not ID exists in nurse using SELECT statement. If user is admin, system checks if ID exists in admin using SELECT statement. If user exists, then user information is loaded into application with SELECT statements.

Before:

A login form on a light blue background. It contains two input fields: 'Username' with the value 'applejuice' and 'Password' with two dots '••'. Below the fields is a grey 'Login' button.

Username	<input type="text" value="applejuice"/>
Password	<input type="password" value="••"/>

Login

After:

A user profile form with three tabs: 'Profile' (selected), 'Appointment', and 'Billing Info'. The form contains several input fields for personal and medical information, an 'Edit' button at the top right, and a 'Save' button at the bottom right.

First Name		<input type="text" value="ee"/>	Last Name		<input type="text" value="BB"/>
Phone Number		<input type="text" value="222-122-2222"/>	Email Address		<input type="text" value="eunnm@gmail.com"/>
ID		<input type="text" value="111"/>	SSN		<input type="text" value="121-12-1212"/>
Weight		<input type="text" value="150.0"/>	Height		<input type="text" value="1000.0"/>
Age		<input type="text" value="100"/>	Medication List		<input type="text" value="N"/>

Edit

Save

SQL Statements:

```
cur.execute("SELECT * FROM Person P WHERE Username = (%s) AND  
UserPassword = (%s)",(self.lineEdit1.text(), self.lineEdit2.text()))
```

```
cur.execute("SELECT * FROM Patient P WHERE PatientID =  
(%s)",(userID),)
```

```
cur.execute("SELECT * FROM Doctor D WHERE DoctorID = (%s)",  
(userID),)
```

```
cur.execute("SELECT * FROM Nurse N WHERE NurseID = (%s)",(userID),)
```

```
cur.execute("SELECT * FROM DepartmentAdmin A WHERE  
DepartmentAdminID = (%s)",(userID),)
```

```
cur.execute("SELECT FirstName FROM Person P WHERE Username = (%s)  
AND UserPassword = (%s)",(self.lineEdit1.text(), self.lineEdit2.text()),)
```

```
cur.execute("SELECT LastName FROM Person P WHERE Username = (%s)  
AND UserPassword = (%s)",(self.lineEdit1.text(), self.lineEdit2.text()),)
```

```
cur.execute("SELECT PhoneNumber FROM Person P WHERE Username =  
(%s) AND UserPassword = (%s)",(self.lineEdit1.text(), self.lineEdit2.text()),)
```

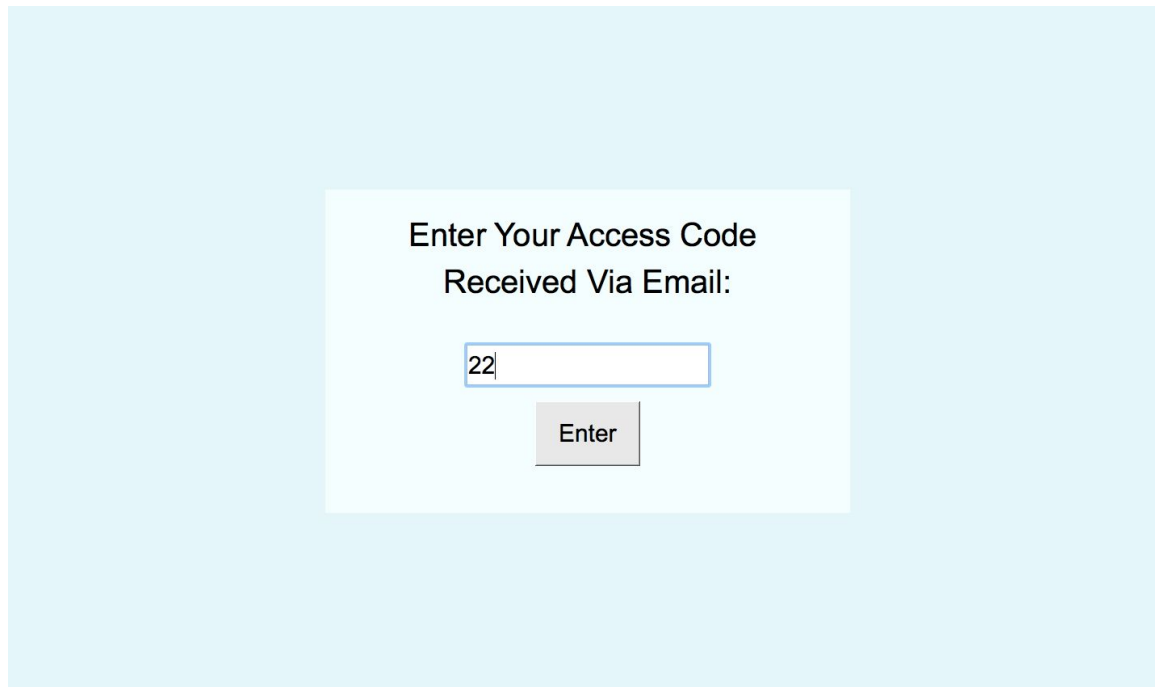
```
cur.execute("SELECT EmailAddress FROM Person P WHERE Username =  
(%s) AND UserPassword = (%s)",(self.lineEdit1.text(), self.lineEdit2.text()),)
```

[Remaining SELECT statements for every attribute needed for Patient, Doctor, and Nurse]

- **Transaction #2: Authentication of Access Code For Signing Up:**

- The user provides an access code that must exist in the AccessCodes table in order for the user to sign up. This access code is provided to the user via email by the department admin who adds the access code to the Access Codes table. The access code is deleted directly after authentication.

Before:



Enter Your Access Code
Received Via Email:

After:

Sign Up Form			
FirstName	<input type="text"/>	LastName	<input type="text"/>
Phone Number	<input type="text"/>	Email Address	<input type="text"/>
Username	<input type="text"/>		
Password	<input type="text"/>	Re-Enter Password	<input type="text"/>
Age	<input type="text"/>	Weight	<input type="text"/>
Height	<input type="text"/>	SSN	<input type="text"/>
Credit Card Number	<input type="text"/>	Insurance Number	<input type="text"/>

SQL Statements:

```
cur.execute("SELECT AccessCodes FROM AccessCodes A WHERE  
AccessCodes = (%s)",self.lineEdit.text(),)
```

```
cur.execute('DELETE FROM AccessCodes WHERE AccessCodes = (%s)',  
access[0][0])
```

- **Transaction #3: Creating a New User:**

After signing up, the information entered is used to insert a new user into the system. An SQL SELECT statement checks whether or not

the username entered already exists in the system to prevent reuse. An SQL SELECT statement checks whether or not the password entered already exists in the system to prevent reuse. An INSERT statement is used to insert a new user into the system.

Before:

Sign Up Form			
FirstName	<input type="text" value="apple"/>	LastName	<input type="text" value="juice"/>
Phone Number	<input type="text" value="312-123-1234"/>	Email Address	<input type="text" value="apple@juice.com"/>
Username	<input type="text" value="applejuice"/>		
Password	<input type="text" value="aj"/>	Re-Enter Password	<input type="text" value="aj"/>
Age	<input type="text" value="23"/>	Weight	<input type="text" value="20"/>
Height	<input type="text" value="5"/>	SSN	<input type="text" value="123-23-2345"/>
Credit Card Number	<input type="text" value="1111222233334444"/>	Insurance Number	<input type="text" value="2222"/>
			<input type="button" value="Sign Up!"/>

After:

Sign Up Form

FirstName	<input type="text" value="apple"/>	LastName	<input type="text" value="juice"/>
Phone Number	<input type="text" value="312-123-1234"/>	Email Address	<input type="text" value="apple@juice.com"/>
Username	<input type="text" value="applejuice"/>		
Password	<input type="text" value="aj"/>	Password	<input type="text" value="aj"/>
Age	<input type="text" value="23"/>	Weight	<input type="text" value="20"/>
Height	<input type="text" value="5"/>	SSN	<input type="text" value="123-23-2345"/>
Credit Card Number	<input type="text" value="1111222233334444"/>	Insurance Number	<input type="text" value="2222"/>

Success!
You've joined United HealthCare.

After receiving success, user is redirected to login or sign up page.

Before Inserting Doctor:

```
mysql> SELECT * FROM Doctor;
+-----+-----+-----+
| DoctorID | Specialty | MedicalLicense |
+-----+-----+-----+
|      222 | SPE2     | THISISLICENSE2 |
|    54321 | rr       | rr              |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

After Inserting Doctor:

DoctorID	Specialty	MedicalLicense
222	SPE2	THISISLICENSE2
54321	rr	rr
104	Oncology	MD

SQL Statements:

```
cur.execute('SELECT * FROM Person P WHERE Username = (%s)',
```

```
(self.lineEdit_5.text()))
```

```
cur.execute('SELECT * FROM Person P WHERE UserPassword = (%s)',
```

```
(self.lineEdit_6.text()))
```

```
cur.execute('INSERT INTO Person(ID, FirstName, LastName, PhoneNumber,
```

```
EmailAddress, Username, UserPassword) VALUES (%s, %s, %s, %s, %s,
```

```
%s, %s)', (accessCodeReceived, self.lineEdit.text(), self.lineEdit_2.text(),
```

```
self.reformat(self.lineEdit_3.text()), self.lineEdit_4.text(), self.lineEdit_5.text(),
```

```
self.lineEdit_6.text()))
```

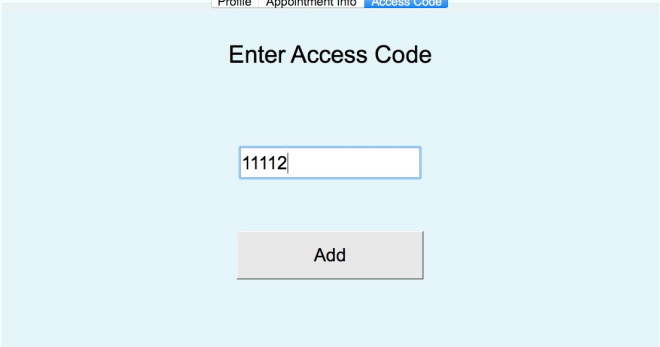
```
cur.execute('INSERT INTO Patient(PatientID, Age, Weight, Height, SSN,
CreditCardNumber, BillingAmount, InsuranceNumber, MedicationList,
DoctorID, NurseID, AdminID) VALUES (%s, %s, %s, %s, %s, %s, 0.0, %s,
"none", %s, %s, %s)', (accessCodeReceived, self.lineEdit_8.text(),
self.lineEdit_9.text(), self.lineEdit_10.text(), self.reformat(self.lineEdit_11.text()),
self.lineEdit_12.text(), self.lineEdit_13.text(), doctorid, nurseid, adminid))
```

[Different INSERT statements are used depending on whether or not inserting into Patient table or Doctor table or Nurse table or Department Admin table]

- **Transaction #4: Department Admin Inserts New Access Codes:**

Department Admin inserts new access codes for patients to use to sign up into the database system with

Before:



The screenshot shows a web application interface with a light blue background. At the top, there is a navigation bar with three tabs: 'Profile', 'Appointment Info', and 'Access Code'. The 'Access Code' tab is currently selected and highlighted in blue. Below the navigation bar, the main content area is titled 'Enter Access Code'. It features a text input field containing the number '11112'. Below the input field is a button labeled 'Add'.

```
mysql> SELECT * FROM AccessCodes
```

AccessCodes
54321
103
104
105
106
5554
5432
2221
9865

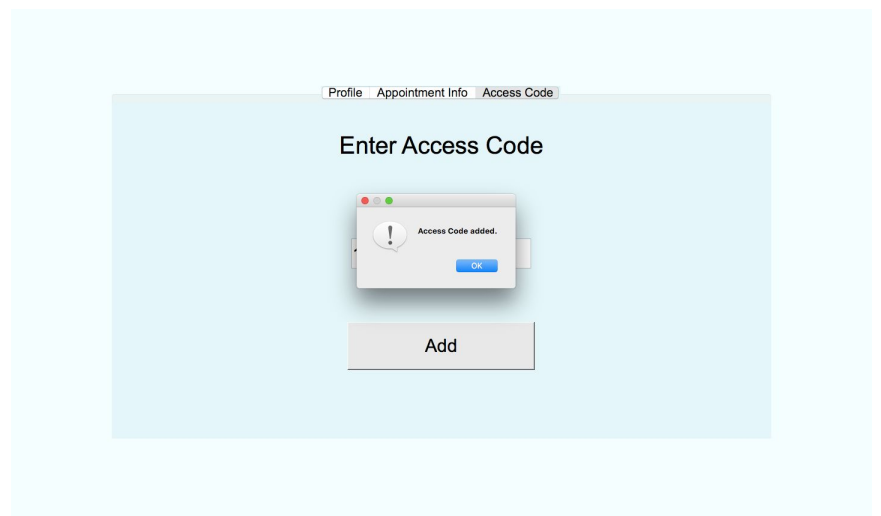
9 rows in set (0.01 sec)

After:

```
mysql> SELECT * FROM AccessCodes
```

AccessCodes
54321
103
104
105
106
5554
5432
2221
9865
11112

10 rows in set (0.01 sec)



SQL Statement:

```
cur.execute('INSERT INTO AccessCodes(AccessCodes) VALUES (%s)',
accessEdit.text())
```

- **Transaction #5: Employees View Appointment Times:**

Employees have the ability to view appointment times when they select the appointment date on the calendar.

Result:

The screenshot shows a web application interface with two tabs: 'Profile' and 'Appointment Info'. The 'Appointment Info' tab is active. On the left is a calendar for December 2018. The date '3' (Monday) is selected and highlighted in brown. To the right of the calendar is a table displaying appointment details for the selected date.

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
		25	26	27	28	29	30
	2	3	4	5	6	7	8
	9	10	11	12	13	14	15
	16	17	18	19	20	21	22
	23	24	25	26	27	28	29
	30	31	1	2	3	4	5

Appointment ID	Doctor ID	Nurse ID	Patient ID	DeptAdmin ID	Location	Date	Start Time	End Time
0	222	333	111	444	SBU2	2018-12-03	0:00:00	19:19:04
1	222	333	111	444	SBU2	2018-12-03	0:00:00	19:19:04

SQL Statements:

```
cur.execute("SELECT * FROM Appointment A WHERE Date = (%s) AND  
DoctorID = (%s)", (dateString, ID))
```

```
cur.execute("SELECT * FROM Appointment A WHERE Date = (%s) AND  
NurseID = (%s)", (dateString, ID))
```

```
cur.execute("SELECT * FROM Appointment A WHERE Date = (%s)",  
dateString
```

- **Transaction #6: Payment**

User has the ability to pay for the bill for their appointment stored in the Patient table. This is done using UPDATE query to update the billing amount in Patient table

Before:

PatientID	Age	Weight	Height	SSN	CreditCardNumber	BillingAmount	InsuranceNumber	MedicationList	DoctorID	NurseID	AdminID
111	100	150	1000	121121212	1234567891234567	400	44	N	222	333	444

Profile Appointment **Billing Info**

Insurance Number 44

Credit Card Number 1234567891234567

Billing Amount 400

Payment Amount 200 Pay

After:

PatientID	Age	Weight	Height	SSN	CreditCardNumber	BillingAmount	InsuranceNumber	MedicationList	DoctorID	NurseID	AdminID
111	100	150	1000	121121212	1234567891234567	200	44	N	222	333	444

Profile Appointment **Billing Info**

Insurance Number 44

Credit Card Number 1234567891234567

Billing Amount 400

Payment Amount 200 Pay

SQL Statements:


```
cur.execute('UPDATE Patient SET BillingAmount = (%s) WHERE PatientID  
= (%s)', (diff2, ID))
```

- **Transaction #7: Create New Appointment:**

Patient has ability to create schedule new appointment. This involves an INSERT statement into Appointment table. It also UPDATES the billing amount

Before:

AppointmentID	DoctorID	NurseID	PatientID	DepartmentAdminID	Location	Date	StartTime	EndTime
0	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
1	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04

Profile Appointment Billing Info

December 2018

You can set up to 4 appointments.

Date: 12/3/18

Start Time: 12:00 AM

End Time: 7:23 PM

Schedule

Current Appointments

12/03/2018 0:0 19:19 12/03/2018 0:0 19:19

Cancel Appointment Cancel Appointment

After:

AppointmentID	DoctorID	NurseID	PatientID	DepartmentAdminID	Location	Date	StartTime	EndTime
0	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
1	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
2	222	333	111	444	SBU2	2018-12-03	00:00:00	19:23:14
3	222	333	111	444	SBU2	2018-12-03	00:00:00	19:23:14

Profile Appointment Billing Info

December 2018

You can set up to 4 appointments.

Date: 12/3/18

Start Time: 12:00 AM

End Time: 7:23 PM

Schedule

Current Appointments

12/03/2018 0:0 19:19 12/03/2018 0:0 19:19 12/03/2018 00:00 19:23 12/03/2018 00:00 19:23

Cancel Appointment Cancel Appointment Cancel Appointment Cancel Appointment

SQL Statements:

```
cur.execute('INSERT INTO Appointment(AppointmentID, DoctorID,  
NurseID, PatientID, DepartmentAdminID, Location, Date, StartTime,  
EndTime) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)',  
(largestApptID, doctorID, nurseID, self.ID, departmentAdminID, "SBU2",  
Date.toString("yyyy-MM-dd"), StartTime.toString("hh:mm:ss"),  
EndTime.toString("hh:mm:ss"))  
  
cur.execute('SELECT BillingAmount FROM Patient WHERE PatientID =  
(%s)', (self.ID))  
  
cost = bill[0][0] + 200  
  
cur.execute('UPDATE Patient SET BillingAmount = (%s) WHERE PatientID  
= (%s)', (cost, self.ID))
```

- **Transaction #8: Cancelling Appointments:**

Patients can cancel their appointments. This involves using the DELETE statement on the Appointment Table. Billing amount is also updated with UPDATE statement

Before:

AppointmentID	DoctorID	NurseID	PatientID	DepartmentAdminID	Location	Date	StartTime	EndTime
0	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
1	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
2	222	333	111	444	SBU2	2018-12-03	00:00:00	19:23:14
3	222	333	111	444	SBU2	2018-12-03	00:00:00	19:23:14

Profile Appointment Billing Info

December 2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

You can set up to 4 appointments.

Date: 12/3/18

Start Time: 12:00 AM

End Time: 7:23 PM

Schedule

Current Appointments

12/03/2018 0:0 19:19	12/03/2018 0:0 19:19	12/03/2018 00:00 19:23	12/03/2018 00:00 19:23
----------------------	----------------------	------------------------	------------------------

Cancel Appointment Cancel Appointment Cancel Appointment Cancel Appointment

After:

AppointmentID	DoctorID	NurseID	PatientID	DepartmentAdminID	Location	Date	StartTime	EndTime
0	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04
1	222	333	111	444	SBU2	2018-12-03	00:00:00	19:19:04

Profile Appointment Billing Info

December 2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

You can set up to 4 appointments.

Date: 12/3/18

Start Time: 12:00 AM

End Time: 7:23 PM

Schedule

Current Appointments

12/03/2018 0:0 19:19	12/03/2018 0:0 19:19
----------------------	----------------------

Cancel Appointment Cancel Appointment

```
cur.execute('DELETE FROM Appointment WHERE AppointmentID = (%s)',
appointNum)

cur.execute('SELECT BillingAmount FROM Patient WHERE PatientID =
(%s)', (self.ID))

cost = bill[0][0] - 200

cur.execute('UPDATE Patient SET BillingAmount = (%s) WHERE PatientID
= (%s)', (cost, self.ID))
```

Users have ability to update their profile information. Involves UPDATE statements.

Comprehensive Patient Data Report - Q3 2023									
Patient Information					Financial & Medical Details				
PatientID	Age	Weight	Height	SSN	CreditCardNumber	BillingAmount	InsuranceNumber	MedicationList	
DoctorID	NurseID	AdminID							
111	255	999.9	5.0	123456789	1111222233334444	0.00	0	R	
222	333	444							

```
mysql> SELECT * FROM Patient WHERE PatientID = 111;
```

PatientID	Age	Weight	Height	SSN	CreditCardNumber	BillingAmount	InsuranceNumber	MedicationList
111	255	999.9	34.0	123456789	1111222233334444	0.00	0	R

```
row in set (0.00 sec)
```

SQL Statements:

'UPDATE Patient SET SSN = (%s), Weight = (%s), Height = (%s), Age = (%s) WHERE PatientID = (%s)'

[Update statements available for most Patient, Doctor, and Nurse attributes]

Other Implementations:

Our Healthcare Portal is unique because it provides services to patients, nurses, doctors, and department administrators including different functionalities for each role. This means the portal loads dynamically for each user and for each user's individual role.

9.0 Contributions:

- Each group member contributed equally to designing and implementing the front end and back end of the project
- Focus:
 - Daeun: Designing Front End
 - Tom: Implementing Front End
 - Rebecca: Implementing Back End