Project Final Report

Group: ShahSDwyerFYarnellC
Group Members: Faith Dwyer, Sanaya Shah, Chloe Yarnell
CS 3200

README

<u>Prerequisites:</u>

- Python (https://www.python.org/downloads/) and MySQL Workbench (https://dev.mysql.com/downloads/workbench/) must be downloaded
- Install the following libraries:
    - import traceback
    - import pymysql
    - import pandas as pd
    - from matplotlib import pyplot as plt
    - import numpy as np
    - import matplotlib.patches as mpatches
    - import tkinter as tk
- The application code for this assignment is **project_application.py**
- The database dump file for this assignment can be found in **project_dbdump.sql**
- The extra credit visualizations can be found in **read.py** as well as the other csv files found in **proj_extracred.zip**. The extra credit GUI extension code can be found in **gui.py**.

<u>Step By Step Instructions:</u>

1. Open MySQL Workbench and ensure you are connected to your MySQL server
2. Download database dump **'project_dbdump.sql'** into your MySQL Workbench and run the file so that the UniversityHealthServices database is accessible
    a. Select Data Import and choose Import from Self-Contained file
    b. Select **'project_dbdump.sql'** and check that the target schema is set to 'UniversityHealthServices'
    c. Click start import
    d. Double check that the database is accessible by checking the Schema tab for the database name 'UniversityHealthServices'
3. Download **'project_application.py'** onto your computer and open it in your python application of choice
4. Run **'project_application.py'** and provide your MySQL Workbench username and password when prompted
5. After you enter your password, you will be taken to the menu with the following options
    a. Manage Patients:
        i. Update: This option allows you to update the information of a patient in the database. You will be prompted to enter the patient's ID, and then you can provide the updated information such as first name, last name, college, phone number, professor status, student status, TA status, and last treatment date.

    ii. Delete: Choosing this option enables you to delete a patient from the database. You will be prompted to enter the patient's ID, first name, and last name and the corresponding record will be removed from the Patients table along with associated records in other related tables (medical history and health insurance).
    iii. Select: This option allows you to retrieve and display information about a specific patient. You will be prompted to enter the patient's ID, and their details, including medical history, will be shown on the screen.
    iv. Insert: This option allows you to insert the patient's profile information and medical history information.

b. Manage Universities:
    i. Insert: This option lets you add a new university to the database. You will be prompted to enter the university's name, street number, street name, zip code, city, and state. Once entered, the new university will be added to the University table.
    ii. Delete: Choosing this option allows you to delete a university from the database. You will be prompted to enter the university's name, and the corresponding record will be deleted from the University table.
    iii. Select: This option allows you to display all university information.

c. Manage Health Center:
    i. Select: This option allows you to retrieve and display information about a specific health center. You will be prompted to enter the health center’s name, and its details will be shown on the screen.
    ii. Insert: This option lets you add a new health center to the database. You will be prompted to enter the health center’s name, street number, street name, zip code, city, state, and associated university. Once entered, the new health center will be added to the table.
    iii. Delete: Choosing this option allows you to delete a health center from the database. You will be prompted to enter the health center’s name, and the corresponding record will be deleted from the HealthCenter.

d. Manage Staff:
    i. Select: This option allows you to retrieve and display information about a specific staff member. You will be prompted to enter the staff member’s ID, and their details will be shown on the screen.
    ii. Insert: This option lets you add a new staff member to the database. You will be prompted to enter the staff member's ID, first name, last name, associated health center, and role. Once entered, the new staff member will be added to the Staff table.

    iii. Delete: Choosing this option allows you to delete a staff member from the database. You will be prompted to enter the staff member's ID, and the corresponding record will be deleted from the Staff table.
    iv. Update: This option allows you to update a staff member's first name, last name, or role.
e. Manage Health Insurance:
    i. Select: This option allows you to retrieve and display information about a specific health insurance policy. You will be prompted to enter the patient's ID, and their health insurance details will be shown on the screen.
    ii. Insert: This option lets you add a new health insurance policy to the database. You will be prompted to enter the provider's name, policy number, associated patient, and copay amount. Once entered, the new health insurance policy will be added to the healthInsurance table.
    iii. Delete: Choosing this option allows you to delete a health insurance policy from the database. You will be prompted to enter the patient's ID, and the corresponding record will be deleted from the healthInsurance table.
    iv. Update: This option allows you to alter the patient's insurance provider, copay amount, or policyNum.
f. Manage Exam:
    i. Select: This option allows you to retrieve and display information about a specific exam. You will be prompted to enter the exam number, and its details will be shown on the screen.
    ii. Insert: This option lets you add a new exam to the database. You will be prompted to enter the room number, associated patient, associated staff member, and date. Once entered, the new exam will be added to the Exam table.
    iii. Delete: Choosing this option allows you to delete an exam from the database. You will be prompted to enter the exam number, and the corresponding record will be deleted from the Exam table.
    iv. Update: This option allows you to update the exam’s room number, patient, staff, and date.
g. Manage Diagnosis:
    i. Select: This option allows you to retrieve and display information about a specific diagnosis. You will be prompted to enter the diagnosis number, and its details will be shown on the screen.
    ii. Insert: This option lets you add a new diagnosis to the database. You will be prompted to enter the diagnosis number, description, illness status, associated exam, and patient. Once entered, the new diagnosis will be added to the Diagnosis table.

        iii. Delete: Choosing this option allows you to delete a diagnosis from the database. You will be prompted to enter the diagnosis number, and the corresponding record will be deleted from the Diagnosis table.

    h. Manage Treatment:
        i. Select: This option allows you to retrieve and display information about a specific treatment. You will be prompted to enter the treatment number, and its details will be shown on the screen.
        ii. Insert: This option lets you add a new treatment to the database. You will be prompted to enter the treatment number, associated diagnosis, date, patient, and type. Once entered, the new treatment will be added to the Treatment table.
        iii. Delete: Choosing this option allows you to delete a treatment from the database. You will be prompted to enter the treatment number, and the corresponding record will be deleted from the Treatment table.
    i. Manage Bills:
        i. Select: This option allows you to retrieve and display information about a specific bill. You will be prompted to enter the bill number, and its details will be shown on the screen.
        ii. Insert: This option lets you add a new bill to the database. You will be prompted to enter the patient, bill number, amount due, and payment status. Once entered, the new bill will be added to the Bills table.
        iii. Delete: Choosing this option allows you to delete a bill from the database. You will be prompted to enter the bill number, and the corresponding record will be deleted from the Bills table.
    j. Manage Appointment:
        i. Select: This option allows you to retrieve and display information about a specific appointment. You will be prompted to enter the appointment number, and its details will be shown on the screen.
        ii. Insert: This option lets you add a new appointment to the database. You will be prompted to enter the patient, date, time, and reason. Once entered, the new appointment will be added to the Appointment table.
        iii. Delete: Choosing this option allows you to delete an appointment from the database. You will be prompted to enter the appointment number, and the corresponding record will be deleted from the Appointment table.
        iv. Update: This option will allow you to update the appointment date, time, reason, or exam.
    k. If you enter the exit option on the main menu, the application will close.
6. After you are done using the application for the main project, download **‘read.py’** as well as the other csv files associated with the datasets required libraries onto your computer

and then open it in a python application of your choice. All of these files can be found in **‘proj_extracred.zip’**.

7. Simply run this file in order to produce five visualizations about the data
8. If you would like to access the GUI extension, download **‘gui.py’** and make sure you have the necessary libraries imported like tkinter and pymysql to your computer and run this file to access this part of our project.

Technical Specifications

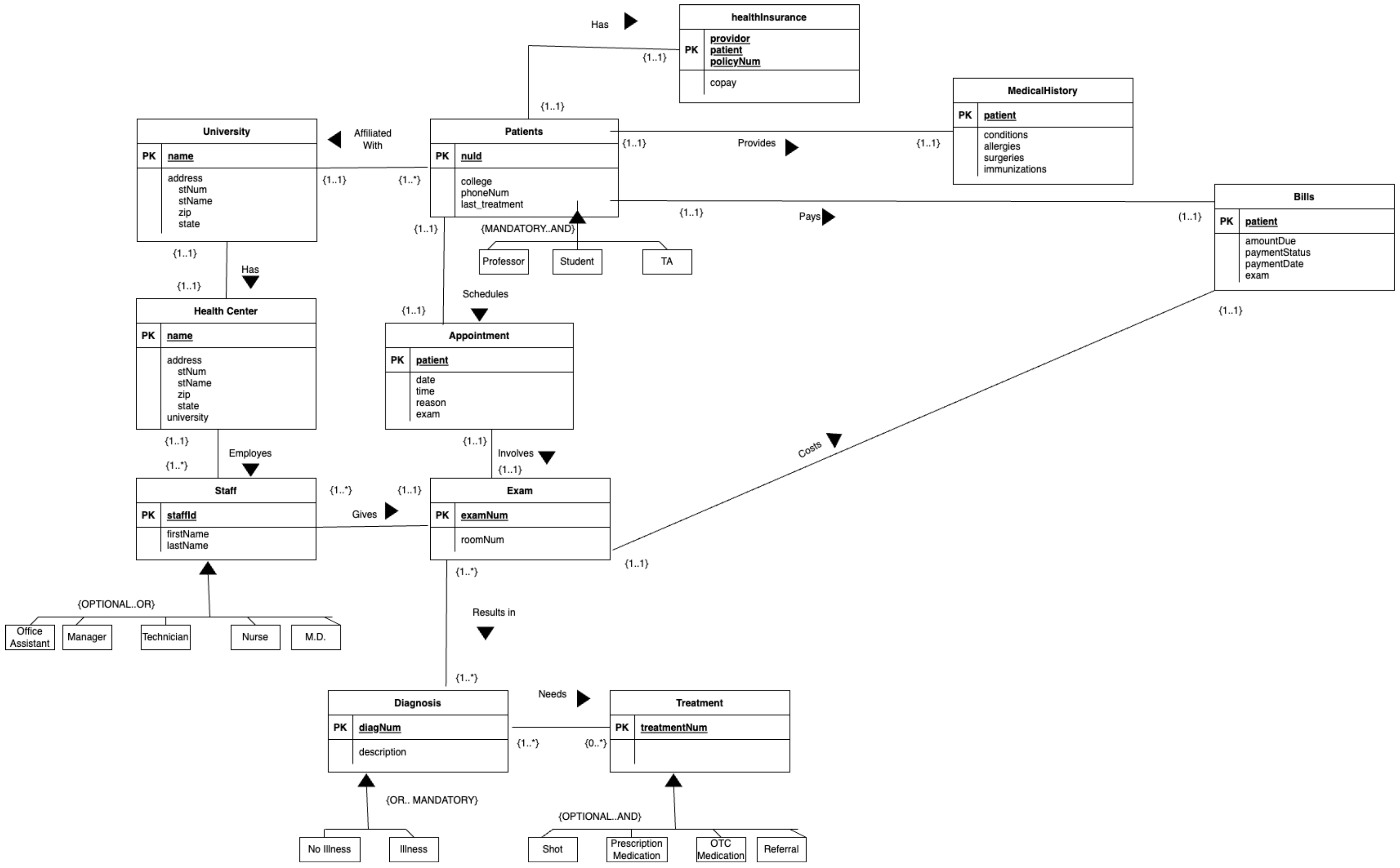
This project uses Python and MySQL Workbench.

Libraries imported for the project are pymysql to connect to MySQL Workbench as well as traceback which helps elaborate on errors that are thrown and help in debugging.

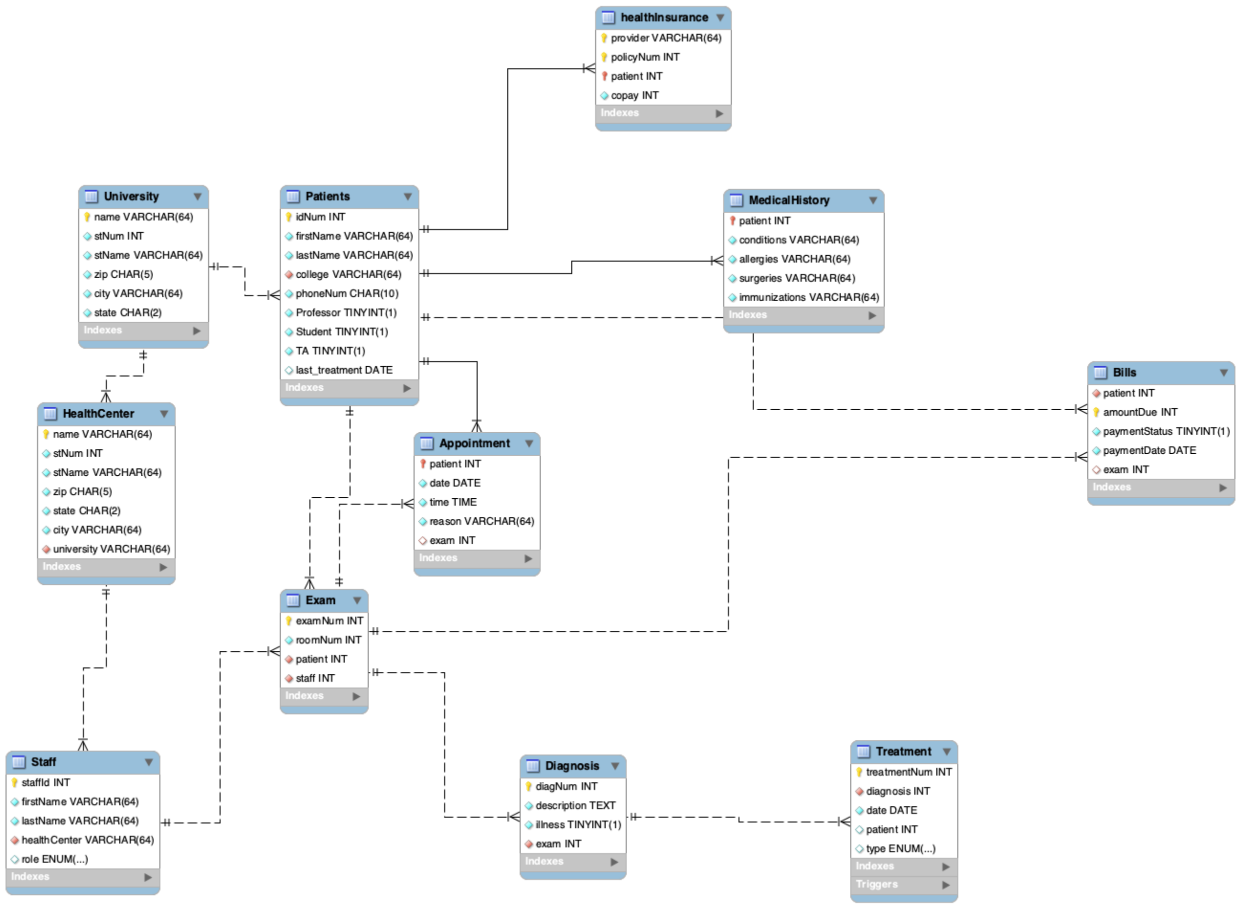
For the extra credit visualizations, pandas, numpy, matplotlib.pyplot, and matplotlib.patches were used.

The extra credit GUI extension uses tkinter and pymysql.

Current conceptual design as a UML:



Logical design for the submitted database schema:



Final user flow of the system:

Start: user is prompted to give the login
If an invalid login is entered, the user will not be taken to the main menu and will have to rerun the program.
If the user has a valid login, they will be taken to the Main Menu
The Main Menu will offer the following options: bills, university, patient, staff, exam, diagnosis, treatment, insurance, appointment, health center, or exit.
If university is entered, user inputs the name of the university

- If it is an invalid name, they can choose to add the university or they can choose not to add and retry and input a new name. If they choose to add the university, they will be prompted to input the stNum, stName, zip, state, and city, and then they will be taken back to the main menu.
- If it is a valid name, the user can choose to either select the university information or delete it
- The user will then return to the Main Menu

If bills is entered, the user inputs the BillNum

- If it is an invalid billNum, the user will be able to insert new values. These attributes will check that the inserted valid.
- If the patient is invalid, the user can either retry and be directed back to the add bill menu or add the patient and be directed to the main menu.
- If the patient is valid, there will then be a check to make sure that exam and patient are correctly associated. If the exam is invalid, the user can either retry and be directed back to the add bill menu or add the exam and be directed to the main menu. If both are valid, then the bill will be inserted.
- If the billNum is valid, the user can either select that bill and display all of the information, or update that bill and choose a valid attribute to change. If the attribute is invalid, the user will be prompted to try again.

If patient is entered, the user inputs an idNum

- If the idNum is invalid, the user can either choose to retry where they are redirected to the patient menu and can enter a valid id, or add a patient where they are redirected to the main menu and can insert a patient. While inserting the patient, the University attribute will be validated. If it is valid, the patient will be inserted, if not the user will be prompted to retry
- If the idNum is valid, the user can choose to select and display the patient's information, update and choose an attribute to update. This attribute will be validated and if an invalid attribute is given, the user will be prompted to try again or return to the main menu to edit that attribute's data.

- Lastly, there is an option to delete that patient's information. If this option is chosen, their patient profile, medical history, and insurance will be deleted from the database.

If exams is entered, the user inputs the examNum

- If the examNum is invalid, the user can choose to retry and enter a valid exam number, or can choose to add an exam and be taken back to the main menu. While inserting a new exam, the staffId and patient idNum will be validated. If an invalid value is entered for either, the user will be prompted to retry or add a new staff/patient and be directed back to the main menu. If all entries are valid, the exam will be created. If the attribute chosen is invalid, the use will be prompted to try again.
- If the examNum is valid, the user will be prompted to either select the exam and display the exam information, delete that exam information, or update the exam information.
- While updating, the patient attribute will be validated. If the patient and staff entered is invalid, the user will be given the options of adding the patient/staff and being directed back to the main menu, or retrying and being directed back to the exams menu and entering a valid patient idNum or staffId.
- Once all actions are completed, user is directed back to main menu

If insurance is chosen, the user will enter a patient idNum

- If that patient has insurance in our database, the user will be given the option to select that information and display the results or update that information and choose an attribute to update.
- If the patient does not have insurance in our database, the user will be prompted to create a profile for them. During the creation, the patient idNum will be validated. If it is invalid, the user will be asked to either retry and be taken back to the insurance menu, or add the patient and be taken back to the main menu. If all entries are valid, the insurance tuple will be created.
- Once all actions are completed, user is directed back to main menu

If treatment is chosen, the user will enter a treatmentNum

- If that treatmentNum exists, the user will be given the option of selecting the treatment and displaying the information or deleting the treatment and then being redirected to the main menu
- If the treatmentNum does not exist, the user will be given the option of creating a new treatment. If they enter create, they will fill in the attributes The patient and diagnosis attribute will be validated. If either are invalid, the user will be prompted to either try again and start the entry over, or add the patient/diagnosis and be directed back to the main menu. If all entries are valid, the tuple will be created.
- Once all actions are completed, user is directed back to main menu

If diagnosis is selected the user will enter the diagNum

- If the diagnum exists in the database, the user will be prompted to either select and display the information, or delete that diagnosis. Both will take the user back to the main menu when completed
- If the diagNum does not exist, the user will be prompted to either create a new diagnosis and enter the fields. Patient and exam will be validated. If either attribute entered doesn't exist, the user will be prompted to either try again or create that patient/exam and be directed back to the main menu. If all attributes are valid, the user will be directed back to the main menu.
- Once all actions are completed, user is directed back to

If staff is selected, the user will be prompted to enter a staffId

- If the staffId exists in the database, the user will be asked to either select and display that information, or delete that tuple. Both operations will bring the user back to the main menu when complete. If the user decides to update the staffId, they will select an attribute to update and enter the new value. If the attribute doesn't exist, they will be prompted to try again with a valid attribute.
- If the staffId doesn't exist, the user will be prompted to either create a new staff member or try again with a valid idNum. The patient will enter new values and the health center value will be validated. If the health center is invalid, the user will be prompted to either retry or add the center and be directed back to the main menu. If all entries are valid, the user will create the tuple.
- Once all actions are completed, user is directed back to main menu

If appointment is selected, the user will enter a patient idNum

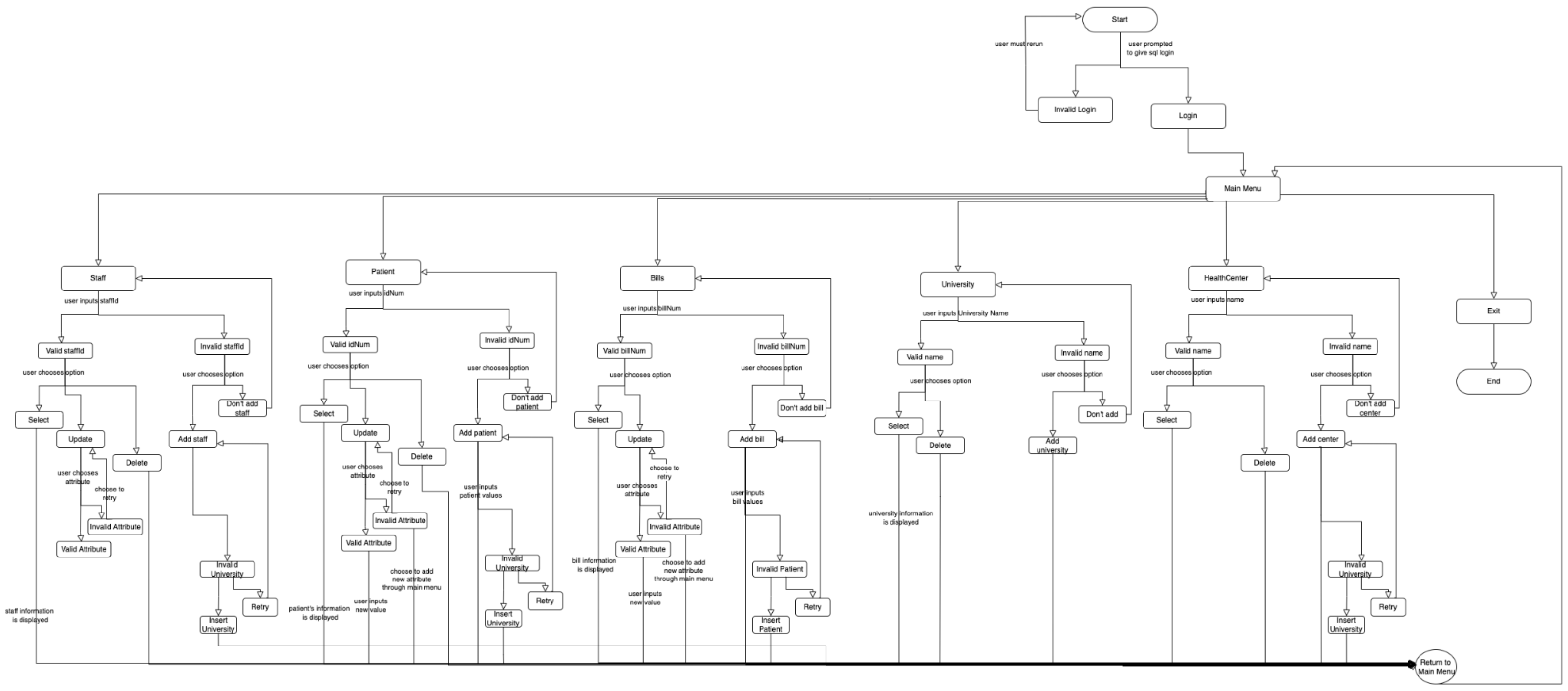
- If the patient has an appointment, the user is prompted to either select the appointment information and display it, delete that appointment, or update the appointment. If the attribute they would like to update doesn't exist, they will be prompted to try again. If they try to update the exam attribute to an exam that doesn't exist or an exam that doesn't reference the correct patient, then they will be given the option to retry or to go to the main menu and edit exams. If all attributes are valid, the tuple will be updated.
- If the patient does not have an appointment, the user is prompted to either create one or is redirected back to the appointment menu. If they choose to create an appointment, the user will enter attributes. The patient attribute will be validated and if the patient doesn't exist in the database, the user will be prompted to either retry with a valid number or add the patient and be taken back to the main menu. If the patient does exist, then there will be a check to make sure the inputted exam is associated with the given patient.
- If the patient and the exam are valid, then the tuple will be created.
- Once all actions are completed, user is directed back to main menu

If health center is selected, the user will enter the name of a health center

- If the name exists, they will have the option to select the health center and display the information, or delete that tuple.
- If the name doesn't exist, the user will be prompted to create the health center and enter all fields. The university field will be validated. If it is invalid, the user will be prompted to retry or add the university and be taken back to the main menu. If all fields are valid, the tuple will be added.

If exit is entered, the application will close

Here is a flowchart depicting the start of the system, as well as the main menu if the user selects staff or patients (to demonstrate select, update, delete, and insert), bills (to demonstrate select, update, and insert), health center or university (to demonstrate select, delete, and insert), or exit. This framework is followed for all other attributes as well.



Lessons Learned

Technical expertise gained:

Practice connecting SQL to Python and generating an API helped piece together all of the homeworks and lessons we worked on.

Insights, time management insights, data domain insights, etc:

This project was challenging in the sense that we had to think about the real-life implications of our database. Unlike many of the assignments we work on in class which are hypothetical, this project required a higher level of creativity and practicality. Especially since healthcare has become much more prevalent in recent years due to the pandemic. Additionally, developing this database requires much more problem solving than just simply querying information like we do in homework assignments. It took a lot of thought to dive deeply into the relationships between entities and ensure that they are represented correctly in order to develop a smooth process for the client. Additionally, thinking through creating a system that both catches and signals all of the potential errors required a lot of effort.

Realized or contemplated alternative design / approaches to the project:

During the process of designing this project, we contemplated using different host languages for the front end code. As 2 of our 3 group members are Computer Science majors and familiar with Java, at first this seemed like the easiest approach. However, we decided to use Python as it would also offer the ability to visualize our data. As we continued to develop our project, we realized that additional tables, such as the health insurance table, were needed in order to make our schema more realistic.

Document any code not working:

Currently, our GUI extension isn’t working completely. We have some connectivity, but it will need to be developed more in order to make it functional for real-life use.

Future work

Planned uses of the database:

This database would be a helpful reference for healthcare systems in universities. It would help them manage their patient information, such as appointments, exams, medical history, and insurance information.

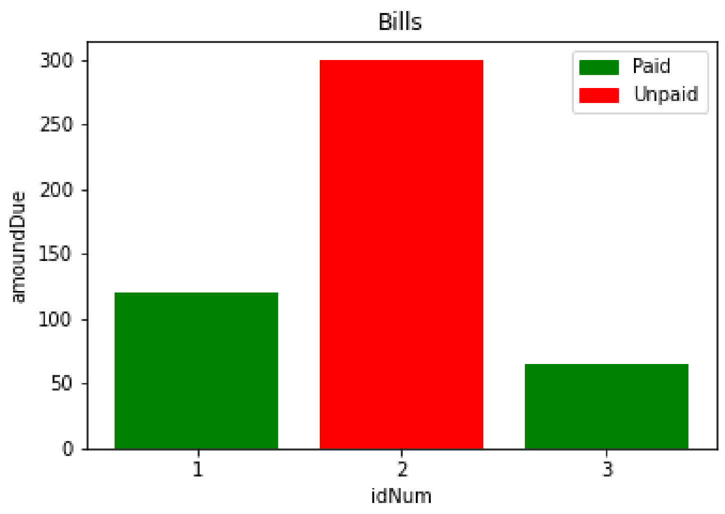
Potential areas for added functionality:

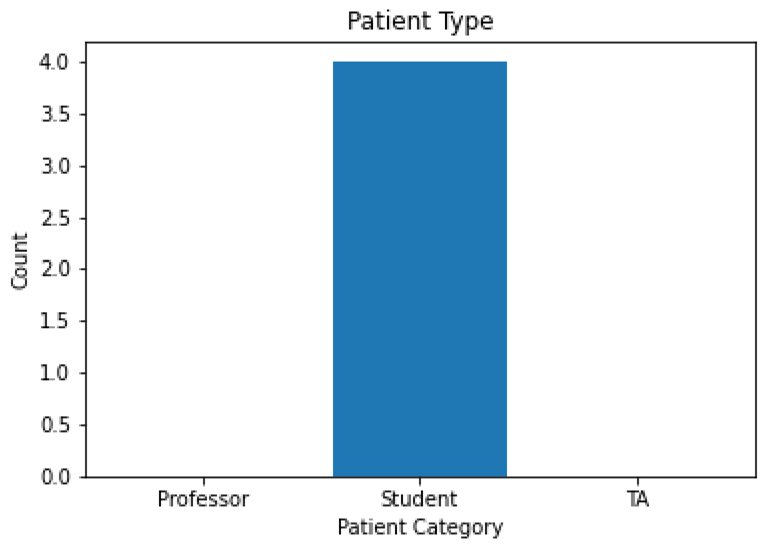
Enhancing security would provide additional functionality and provide an extra layer of support for sensitive information that patients may submit (such as their Social Security Number or address). The encryption of column names could be helpful in this area. Additionally, there could be an archive for patient information that is wiped from the database from the data clean event. Another small change could be to add password masking when we ask the user to input their mysql password.

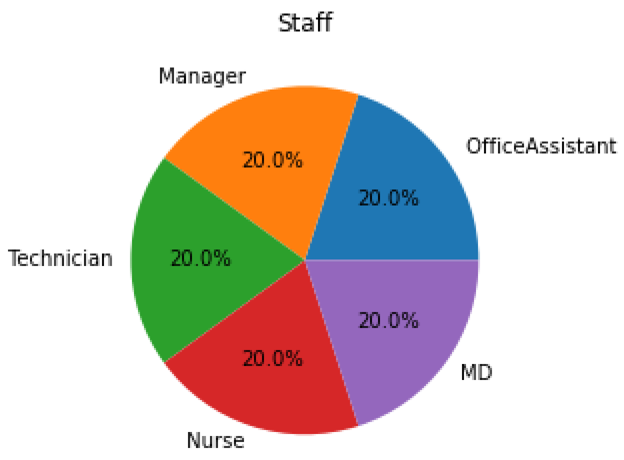
Right now, this system is set up for internal use, since anyone with the proper credentials can login and see all patient information. We had an idea that the system could be altered to allow patients to log on to see limited information, perhaps to see information about their own university's health center(s), to see their scheduled appointments, or to check if they have any outstanding bills. We started to develop a GUI extension for this idea, but there is a lot of room for improvement. Enhancing the GUI will provide a more user-friendly and seamless experience for users compared to using a python terminal interface. By improving the displays and functionality of the GUI, users can interact with our system more intuitively and efficiently. Again, increased security and encryption measures will need to be a priority, since this database contains a lot of sensitive patient information and we don't want patients able to access other patients' information.

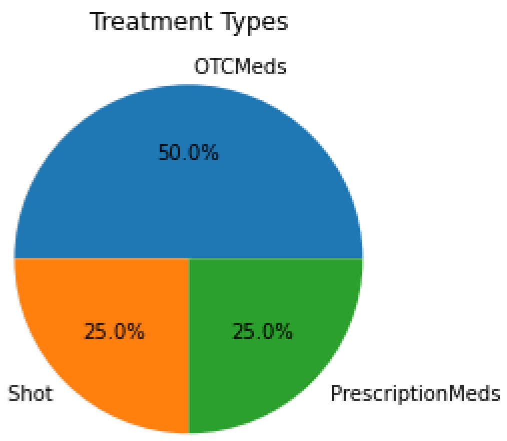
Extra Credit

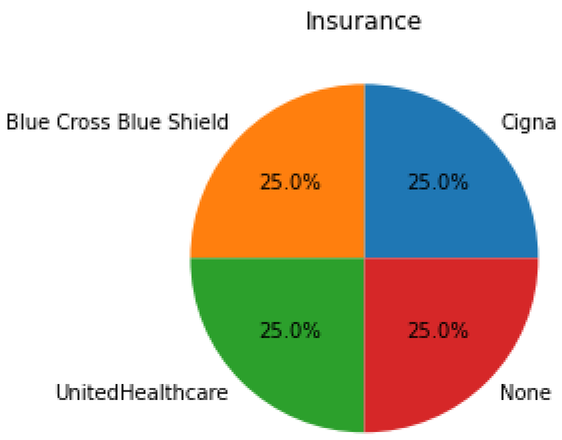
1. Visualizations











2. Our project includes more than 10 tables

   University, Health Center, Staff, Patients, Medical History, Insurance, Appointment, Exam, Diagnosis, Treatment, and Bills

3. We added a GUI extension for patients

   This can be found in **'gui.py'**.