Design of a Python Command-Line Interface for Hospital Operations Coordination

Samuel Platzman

Jake Ashkenase

CS3200 Database Design

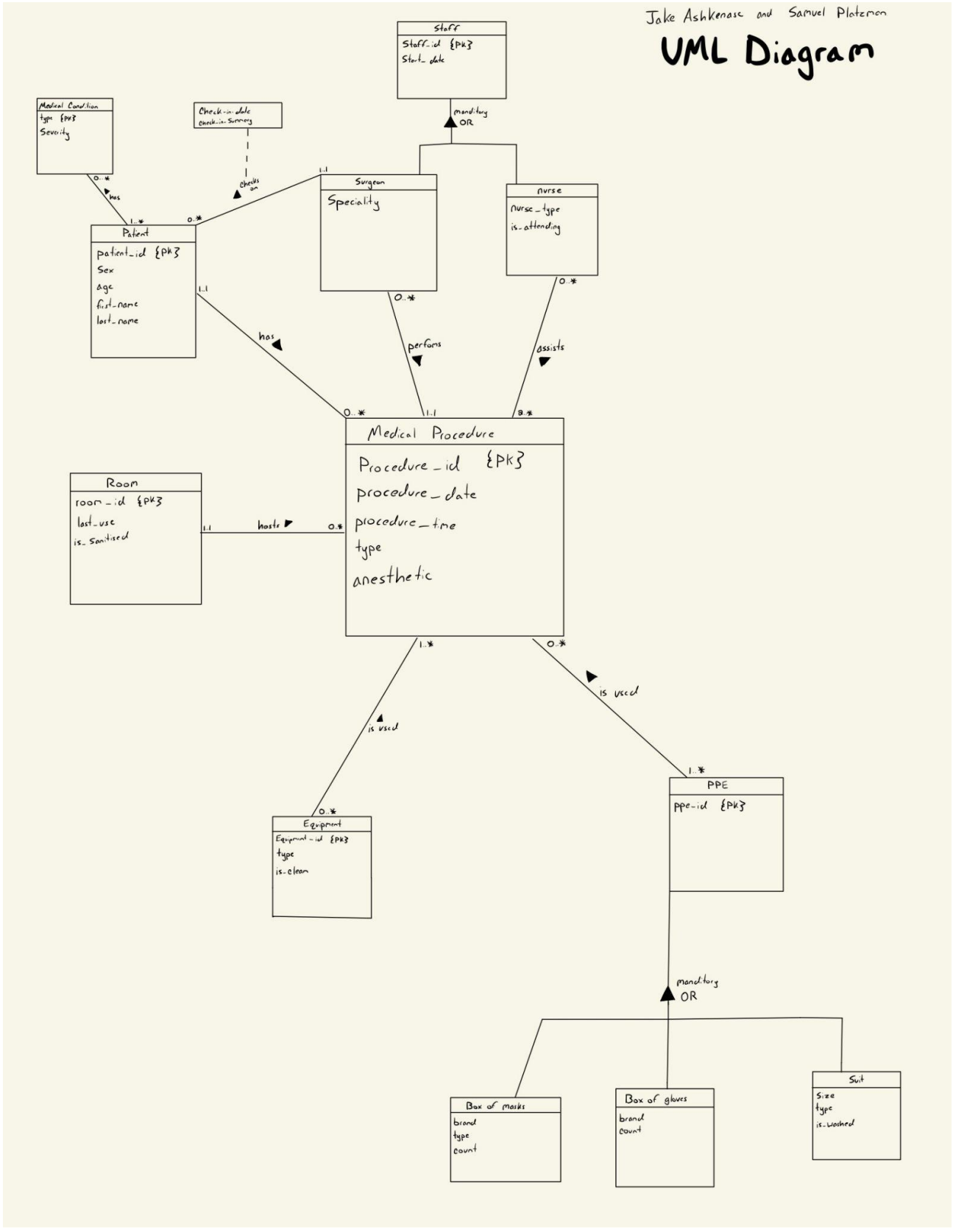
README:

Our project must be run on Python and Pycharm is the preferred python interface to use. The libraries you will need to have installed for our project are pandas and pymysql. Once those are installed all that needs to be done is run our dump file in SQL. When this file is run our database will be loaded onto your local SQL server. Once that is complete all there is left to do is run our Python file, enter your username and password for SQL and then use our interface.

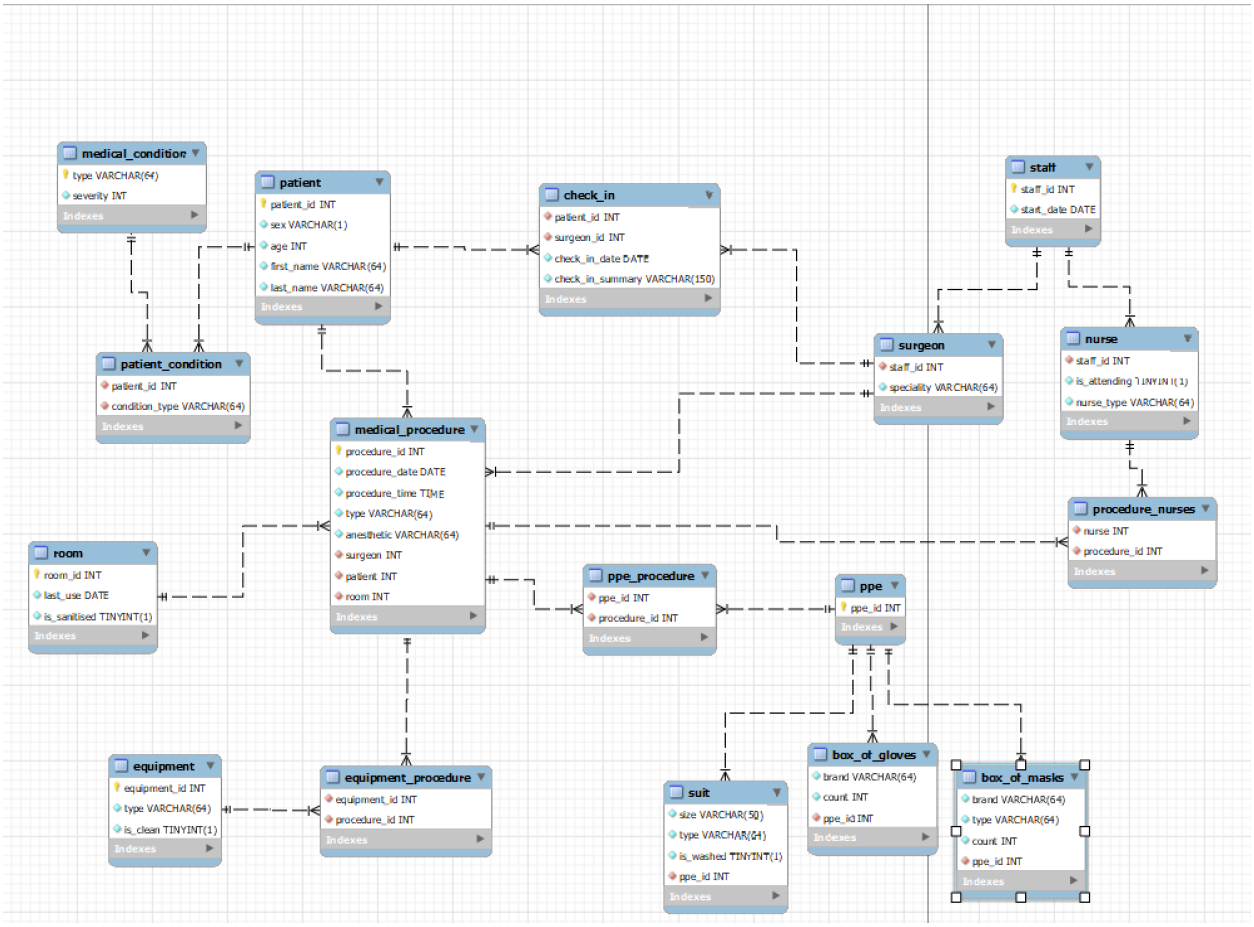
Technical Specifications:

For our project we used SQL to hold our database and then used Python to create a frontend command line interface for the user to interact with the database. We used Pycharm specifically in order to create our interface and the library we used to connect our python code to the database was pymysql. The only other library used in the project was pandas within python in order to show our tables in neat dataframes. This is all the technology we needed in order to complete the project.

Conceptual Design:



Logical Design:



User Flow:

Our user for this project is an operations coordinator, or a staff member that works with the PPE and equipment inventories, sanitation, and personnel pertaining to surgical procedures in a hospital. The user would provide a command within the Python script and then Python would send that command to SQL.The options that the user has includes reading tables from the database, inserting PPE/ deleting PPE, viewing the count of PPE, viewing a medical procedure and the PPE used in that procedure, and updating rooms/suits to be marked as clean. Once SQL has the command it would perform it, assuming the command was plausible, and then sends its results back to Python. From there Python makes minor adjustments to make the results easily

readable and then provides it back to the user. This cycle is repeated until the user no longer wants to send it requests and exits the program.

Lessons Learned:

Technical Expertise Gained: This project has done a lot to help with our technical expertise on multiple different platforms. To start, our SQL skills have become developed and fine tuned through this project. The concepts we have been working on all year have been reinforced and put to use in order to make a sound database. Being able to bring together all of the different concepts we learned into one project was very rewarding. However, this project didn't only allow us to practice learned skills in SQL, but it also pushed us to use our knowledge in circumstances that we have not previously seen. We were able to apply concepts to new and creative ideas in order to make our database structure and also to make procedures for functionality. This led us to work in Python, and while we both had prior skill in this language, we had to push our knowledge further in order to create a user interface. We had to learn about using libraries related to databases but also learn about how to deal with errors given bad user input. Dealing with all sorts of different errors and making sure the user was inputting values that we deemed acceptable and hard at first because it was something neither of us had previously done. It was however rewarding to get the hang of it and implement it throughout our project, as was the expertise we gained in the many other aspects of this project.

Insights: One major insight we gained from this project is that it is very difficult to build our interface around a specific user and being able to put constraints on them based on their role. We had to do a lot of thinking about what exactly an operations coordinator would be able to do and how much access to the database we should be giving them. There were a lot of gray areas

and we spent a lot of time discussing what was an acceptable thing for our user to do and what was not. This also led us to think about how this can be applied for different users using the same database. Could you perhaps create a similar interface for a custodian but take away some of the power that an operations coordinator has in theirs. We thought a lot about how this concept could be applied to a very large scale operation like a hospital. How many different variations of an interface would have to be created in order to suit the hundreds of workers that work in a hospital. Overall while we find the work we did very useful, we also realized how much work is needed to allow our database to be used by dozens of different positions who all need different permissions to our database.

Alternate Approaches: Initially, we planned to utilize a Java-based command line argument to integrate with the SQL database for this project. We switched to Python for the sake of our mutual understanding of the language and procedures needed. If we continued as planned and integrated Java into our model, we believe that further implementing additional functionality per hospital would become a more succinct process, and would improve functionality as the model grows.

Non-functioning Code: Our project did not have any code that was not functioning but we do have one part that does not do exactly what we want it to. This happens when a user inputs a request to perform an action on the database and the input is not one specified in our instructions. When this happens instead of immediately prompting the user for another input, they must first specify that they want to perform another request before they can try and type the right request. For example, if a user wants to request to read tables and inputs READ TABLES instead of the READ like the instructions says, they will be told that is not valid but instead of immediately being able to try again they must specify that they want to perform another action. This is in no

way detrimental to our interface functionality but it does create an extra step for the user that is not needed.

Future Work:

Planned Use As it stands, this project represents a proof of concept for software intended for a hospital operations coordinator to utilize in order to coordinate the rooms, equipment, personnel and PPE needed for surgical procedures to occur smoothly. Minor edits in the SQL dump for this project would allow for the database to be tailored to specific hospital needs (e.g. specific personnel additions and room numbers). These differences can be accounted for upon installation of the software, and then all CRUD operations needed for full functionality of the operations coordinator will be accessible through the python script.

Added Functionality For ease of use and legibility, a graphical user interface could be integrated alongside a locally accessible webpage for any computers on the hospital's private network. This would allow for operations coordinators to access the data they need without having to navigate PyCharm. Either embedding JavaScript into the webpage, or having it interface with the already existing Python script would allow for the SQL database to be accessed in such a way. This model can also be expanded to encompass greater hospital data storage needs, like patients in rooms outside of just the ones for operations, pharmaceutical inventory, or the plethora of patient-facing or practitioner-facing data.