Justin Widney

Michael Hackman

Colton Begert

General OverView:

Main Screen:

User starts up program and is presented with 3 options:

Logging in

Adding Data

Shutdown

Logging in will take the user to the Login Screen where they can login or add a new user. Choosing to add data allows the user to enter a path of the sql file they wish to parse (as long as it’s in in the same folder as hospital.py). Shutdown allows the user to close the program.

Login Screen:

The login screen prompts the user for their login and password and allows them restricted access to the database depending on their role. It presents them with 4 options:

Enter ‘newuser’ to add a new user into the system

Enter ‘shutdown’ to close the program entirely

Enter ‘exit’ to go back to the Main Screen

Enter your login to begin the login process.

The login process consists of entering the username and password pair.

If the pair match a pair in the database, then the user moves to the Action Screen

The new user entry takes the user to the New User Screen.

New User Screen:

From the New User Screen the user can choose to exit back to the Main Screen by typing ‘exit’ or begin by entering their desired staff\_id.

The New User Screen will prompt users for desired entries for the tuple:

(staff\_id, role, name, login, password)

And then prompts them to ask if they want to commit their decision (y), reject it and try again (n) or exit to the Login Screen.

After they choose their option, they will be returned to the Login Screen if they choose yes or exit, or back to the start of the New User Screen if they choose no.

Action Screen:

From here, the user can choose to execute actions related to their Role by entering one of 5 numeric commands. For example, the nurse Action Screen has the following appearance:

What option would you like to choose?

(1) Create a new chart/add a patient

...

Where the user can enter 1 through 4 to choose an action, or 5 to logout and return to the Login Screen.

Flow of data:

We start by calling Hospital.py which builds the databse and calls Login.py. The login file is for all basic user features( logging in/new user) and calling functions for each role. These functons are imported from Doctor.py, Admin.py and Nurse.py. Doctor and Nurse are passed the role and staff id from our login file. Doctor also can call Nurse’s functions 3 & 4 which passes SID. The role files and hospital.py import function calls from Utilities which have helper functions eg.(get\_hcno). These may return the data for the specifed helper function call. Also each role file has the ability to return/exit to the login file to restart the process.

Detailed Design

Note all Roles(D,N, A) have a 5 option to return to the login screen

Doctor.py

Doctors:

(1) Display Chart information for a patient (2) Record a symptom

(3) Record a diagnosis (4) Prescribe a medication

Our first option for Doctor is listing all charts for a selected patient and listing all entries associated with that chart. This is done by calling a function get\_hcno\*. A sql statement then queries that hcno against the database to return a name and all charts associated with that person. The user can then can choose to select the chart and display all entries associated with the specified chart.

Our second option is to record a symptom. This is done by calling get\_hcno\* then querying the database to find the open open chart associated with that person. Calling a function get\_symptom\* we get the specified symptom. Then we use that symptom, the current time, and already known hcno, chart\_id and staff \_id we automatically insert a entry into their chart and return to the doctor options page.

Our third option is to add a diagnosis. This is done identical to our second option. Except we call get\_diagnosis()\* to get the specified diagnosis. The chart is automatically inserted with the current time, staff\_id, hcno and specified diagnosis.

Our fourth option is to add a medication. This is done by calling get\_hcno\* and finding all open charts associated with that hcno. The open chart Id and name is printed. A sql statement is used to get all symptoms associated with that chart\_id. If there are symptoms they are displayed. Allergy\_check is called in which a series of functions is called. The first being get\_medication is called. Then a SQL statement is called to report all allergies for the specified medication. Then another SQL statement is called to get all inferred allergies from the medication or reported allergies. If the person has a reported allergies they are able to recall the function or prescribe that allergic medication. If the person has an inferred allergy to the medication they are able to change the prescription or give it anyways. The user is prompted to enter a medication. A series of queries gets the suggest amount for that medication for the patients age group. If the specified amount prescribed is larger than the suggest amount. A warning is displayed. The user can choose to continue or specify an new amount. Two function are called get\_s\_med\_date()\* and get\_e\_med\_date()\* to get the start and end date. The current time is automatically gathered. All this data is entered into the chart.

(\* functions called from utilities)

Utilities.py:

Contains functions that are either common to multiple parts of the whole program or were placed here to clean up the code and make it more readable. Every function in Utilities that starts with “get” is a fetcher for whatever value follows and it does constraint checking to ensure the value retrieved will not cause an error/cause problems with other segments when entered into the database (like hcno must be exactly 5 digits and must be only numeric).

The only different ones are:

Access\_hcno: a function used when adding a new patient into the database, skips the check to see if the hcno already exists and instead implements it in Nurse system as a check for whether to enter into the new user system or just open a new chart. If the hcno is already in the system, then open a new chart, otherwise add the new patient.

Greeting: used in hospital.py as the main screen. Get’s a correct option from the three provided

New\_chart\_id: starts at ID 00001 and increments until it finds one that isn’t in use, then returns it. Used in Nurses to open a new chart.

Parse\_file: takes the path of a .sql file and then runs the contents.

Login.py

Login:

This function prompts the user to enter one of [‘newuser’, ‘shutdown’, ‘exit’] or their already registered login. Newuser will bring them to the newuser system for adding in new staff members, shutdown exits the program entirely and exit brings users back to the main screen. When a user enters their password, it is hashed immediately and the username and password pair are checked against the usernames and hashed passwords in the db.

Login\_system:

This gets a correct staff\_id and role from Login, and then uses them to determine which action screen to display (nurse, admin or doctor).

New User System

This is a series of segments that get the required entries for the staff table with checking to ensure the entries won’t cause any errors (for example, the staff\_id isn’t someone elses’ staff\_id and is exactly 5 digits). We chose not to implement these functions in utilities as they are only used once. Once it has collected all the data, it returns it in the format required to enter it into the staff table. And returns this information to Login where it is added.

***Administrative staff users***

The functions of this user are implemented in the file Admin.py.

By calling Admin() you are greeted with the menu of options of that were required by the project spec. Selecting an option calls the respective function which handles the requirement. The flow of the sub functions is to connect to the database, request some information from the User and then make and display the query.

The first task was to list each doctor and the drugs that they are prescribing. The program gets a period from the user and passes it into the query. The query is the cross product of staff and medications and is filtered to the specified date region. They are grouped by the doctor and the drug that they prescribe and the amount that is returned is the amount per day X number of days that the prescription lasts. Printing is simple but it only prints the name of the doctor once with an if statement.

The Second task was to find amounts of drugs prescribed broken into the category the drug belongs to. This query operates on the cross product of the drugs and medication tables and is filtered by the date range specified. The result is grouped into the drugs categories. This query gets complicated only in the summing for the totals which is done in SQL by having a nested aggregate function which sums the total for each category and returns the value as a column.

The Third Task is to give the medications with mdate after the ddate if the diagnosis. This qorks by taking the cross product of charts, diagnoses and medications and looking at the common rows. It returns all values that have a common chart\_id and the medication was recorded after the diagnosis. Group by and order by counts are to list the data in the required way.

The Fourth task was to find all diagnoses that are made before a specified drug was prescribed. Similar to the third task makes a cross product of charts, diagnoses and medications and returns the values that have the specified drug name and return all diagnoses that happened before it. It is ordered by the amount per day X the number of days and returns this in the select statement

Nurse: Nurse.py

(1) Create a new chart/add a patient (2) Close a patient's chart

(3) Display Chart information for a patient (4) Record a symptom

Our first option allows the Nurse to create a new open chart for a given hcno, close a currently open chart if there is one (then open a new chart if desired) or add a patient into the system and then open a chart for them. If a hcno is supplied as an argument, then a chart is immediately opened for that patient (provided there isn’t one already open). Access\_hcno gets a valid hcno that may or may not be in the db, if it isn’t in the db we use a series of ‘get\_attribute’ calls to get all the values needed for a patient (in the correct format for entering into the db), and then insert them.

Then we check if there are any open charts in the db for that hcno, if there are, then it asks if you’d like to close it. If you answer ‘yes’ then it closes that chart and opens a new one, if ‘no’ then it returns you to the Nurse’s action screen. To open a new chart, it calls new\_chart\_ID in utilites, which generates an unused chart ID. It then inserts this new chart into the db.

The second option allows the nurse to close a chart. It uses get\_hcno to get a correct hcno that’s already in the db. Then it checks if there are any open charts, if no, then it asks the nurse if they want to open one, if there is an open chart then it asks if the nurse wants to “close” it by updating edate on that chart to the current time.

For brevity, the third option is the same as the doctor’s first option.

For brevity, the fourth option is almost the same as the doctor’s second option. The important difference is that if you are a nurse and the patient has no open chart, you are given the option to open a new one, which takes you to the special case of the first nurse option that creates an open chart for a hcno passed as an argument. Then it drops the nurse right back into adding a symptom.

Testing Strategy:

1st: Testing SQL Statements-

For each option we created a query against the database and ran it though SQL until we got the desired results were achieved. As we created non-trivial test data.

2nd: Testing python-

For the login in screen, function calling, new user, encryption we tested them in python. With blank function calls to test the functionally of our python calls.

3rd: Testing python SQL-

We then tested the SQL statements inside functions calls created earlier. Seeing if the ouput matched what the question asked for.

4th: Testing together-

We linked/combined all files of our program to be able to access any option for each of the specified roles. Then tested all the functionally of the program and being able to switch through the roles and options.

Extra Decisions:

- added feature to update or change schema within python by parsing a file

- We added a option for nurse to create a chart for a patient if there wasn’t one.

Bugs – Null vs None type problems

- importing Methods calls of specifed modules (eg. Importing Admin from Admin vs Importing Nurse then calling Nurse.Nurse)

- (u’text’,) printing errors

Group work Breakdown-

Michael- Test Data, Nurse, Doctor, Login System(Encryption, New user) (25hr)

Justin – Doctor , Design Document (7hr)

Colton – System Admin, Doctor (10hr)

coordination:

GitHub and Group session to work and discuss the Design Document

Group session to combine individual parts.