Rohit Vinnakota - 1399495 LAB D07
Kaiyi Song -1431897 LAB D05
Musaed Alsobaie -1463975 LAB D07

README FIRST

The program will first ask the user to specify a database. We are including ‘try2.db’ with the source code to perform queries on. The user may query the database directly to see the data and the following are logins for the database

1) Doctor→ u:slimshady123 p:loseyourself123
2) Nurse→ u:sorry123 p:notsorry123
3) Staff→ u:iamdonald p:heartbeat123

Small User Guide:
To run the program against the try2 database, run the python file and enter “try2.db” when prompted. Then, enter one of the above usernames and passwords to access the corresponding options and test the data.

Design of software:

The main function first gives us an option to login or to create a new user.

If the option is chosen to login, the login screen is displayed and is used in conjunction with the user_checker function to validate the user name and password. The username and the encrypted password is stored in the database and the password is encrypted using hashlib and 14ssh224 in python.

Once the user is logged in, their role is determined(doctor, nurse, or staff) by checking the staff table in the database. Based on this role, a class instance is constructed and a function from that corresponding class is called to show a further list of options. There is a while loop that runs indefinitely displaying the corresponding options till the staff logs out. This loop comes in handy later on and provides a mechanism for the staff to logout but still have access to the welcome screen.

If the role is a doctor, the doctor has 4 options as per the assignment specifications. If option one is chosen, the doctor can see all charts for the patient and whether they are open or closed. This can be accomplished with a simple database query. Then, the symptoms, medications, and diagnoses tables are joined to the charts table to list all relevant information. All of this is accomplished by the charts() function. If option 2 or 3 is chosen, the symptoms and diagnoses dates are updated with a simple SELECT date('now') call. If option 4 is chosen, the function creates 2 queries, one to check if the suggested amount is greater than the prescribed amount and one that checks if the patient may be allergic to the drug. Both calls are executed

independent of each other and both display their own warning message. This way the doctor can decide what warnings to ignore. This is accomplished by breaking the while loop.

If the role is a nurse, the nurse has 4 options as per the assignment specifications. Options 3 and 4 are same as options 1 and 2 for doctors so the same corresponding functions are called. If option 1 is chosen, the code first takes in the patient's hcno and looks for all corresponding open charts for the patient. If there is no open chart, a new chart is created by calling the create_chart function(). If there is an open chart, the nurse can close that open chart by calling the close_chart function which closes an open chart given the chart_id. The create_chart function includes a patient_checker query to see if the patient exists in the hospital's database. If they do not, then the nurse is forced to enter the details for the patient in addition to opening and entering the details for the chart. For option 2, the close_chart function is called.

For when the user is administrative staff, they also have 4 options.

The First Option: Within the time interval that the user requests(input into start_date, end_date), the user may view the doctor, drug and total drug amount that was prescribed. Additionally, if a doctor part of the hospital did not prescribe a drug in the time specified, their name will not appear in the output. Functions created and used for option one are:

-drug_sums(lists, staff_id_list, i,start_date, end_date)
    -purpose: Contains the query that finds the total amount of each drug(within the specified time). This function is called in the for loop that prints the output for the user to see

-is_no_prescription_during_time_period(dic, name_of_doctor ,lists)
    -purpose: Checks if the doctor has prescribed a drug within the time frame specified. If the doctor does prescribe, 1 is returned. If the doctor has not prescribed, then 0 is returned. This prevents the doctor’s name from being printed if they have not prescribed.

The Second Option: Within the time interval that the user requests, the user may view the categories as well as the drugs, both with their totals. For example, there may be two drugs within the category “analgesic”: obelcet and obizur. Individually, obelcet could have a total of 2 and obizur a total of 4. These amounts are displayed. For analgesic, the total displayed would be 6. Note, that these are arbitrary numbers used for explaining what option 2 does.
Two queries were used in option two only differing by the GROUP BY statement. The formatting for the output was created with two for loops.

The Third Option: In the third option, an if-statement is used to check if the user entered 3. Then user input is taken, where the user can specify a diagnosis in order to see a list of all drugs that were prescribed after the diagnosis was made (in all charts). A query to do this job is then executed. After that the result is stored in an array, and all elements are printed using a for-loop.

Rohit Vinnakota - 1399495 LAB D07
Kaiyi Song -1431897 LAB D05
Musaed Alsobaie -1463975 LAB D07

The Fourth Option:In the fourth option, an if-statement is used to check if the user entered 4. The user enters input to specify a drug. A query is then executed to find all diagnosis that were made before the drug was prescribed (in allo charts). The result is then stored in an array, and all elements in the array are then printed using a for-loop

TESTING

The code will work under the following assumptions
->The username and password are strings. We tested empty usernames, usernames that do not exist, and incorrect passwords.
->For option 1 for a doctor, any chart can be selected as long as the user provides a chart id listed in the database. If they provide one that does not exist the function returns an empty list
->It is assumed for option 2 and option 3 that the doctor lists an open chart
->It is assumed that the nurse enters a non existent chart id. When creating a new chart, it is assumed that the nurse also wants to create a patient entry if one does not exist.
->For all options, it is assumed that open charts are specified when asked for and they exist in the database. The program does not cover unexpected data that either does not exist or is inconsistent with the outlined parameters(open/closed charts, medications, hcno, etc)
->We used try2.db to test the code. We added pertinent values into the tables if they were missing or if our queries were returning empty sets.

PROJECT BREAKDOWN

Rohit: Nurse option 1 and 2

Kaiyi: staff option1 and 2

Musaed: Staff option 3 and 4

All of us discussed how to approach the queries including how to deal with joins and an abstract approach to them. We also helped each other out with python itself, particularly syntax errors and relevant functions. We largely worked on the project in person and spent a total of over a 60 hours combined. The vast majority of the project was done together. The options for the doctors were done working together and the code was worked on individually with versioning using slack and github.