CMPUT 291

Project 1

Team Members:

Brandon Smolley

Kathleen Baker

Steven Cherfan

Main Menu: startProgram.py

When the program is started the user is greeted by a prompt to login, after entering their information they will connect to their Oracle account, after a successful login the user is then given multiple options. The user can prescribe a test to a patient, look at medical history for a patient, update a patients key information, or perform a variety of searches, such as searching for doctors, patients and patients who have not taken tests that they qualify for. The user can also exit the program from the main menu.

Prescribe Test: prescribeTest.py

A user is allowed to prescribe a test to a patient if the user can provide the name of the doctor or the doctors employee id, the name of the test that the patient is going to take, as well as the name of the patient or the patients health care number. At each point the system checks to see if a valid doctor has been entered, if the doctor is valid the user is prompted for the test name, otherwise the user must enter another doctor. The second step is to enter the test name, after entry the system will check to see if the test is in the database, if it is the user proceeds, if it is not the user must enter another test. The last step is to enter the patients name or health care number. The system will check to see if the patient is in the database and if the patient can take the test, if the patient is either not found or unable to take the test the user is asked to enter another patient. If all of the entries are valid a test record is added with the information provided by the user.

Medical Test: enterTestResult.py

The function enterTestResult takes in the connection to the oracle database and the cursor to execute queries and inserts. The first requirement of this function is to find existing est records in which to edit.

A variable testID is assigned an int by calling the function findTestRecord which is also given the connection and cursor to the oracle database. findTestRecord asks the user to enter in information

into fields or to leave them blank. Based on if the field is blank or not the function will query the database and print information stored in the database relevant to that query. The point of the search is to

get a test ID, whether the user enters it first as that information is already known or finds the desired entry from a displayed list.

enterTestResult now has a test ID stored in testID. Now the function will ask for information pertaining to the test record that will be entered into the database. If a field is left blank then that value in the database will not be changed. Each entry if the database returns an error will require the user to enter a value that will not return an error from the database. At the end if all is done successfully the

function will print a success message and return to the main function.

Update Patient: updatePatient.py

Search Patient: searchPatient.py

Patient information can be obtained by the user by searching for a patient by name or by health care number. The information displayed includes the patient name, health care number, name of the test taken by the patient, date the test took place and the result of the test. If no test records are available for a patient the user is told that no records can be displayed. The user is then taken back to the main menu for further system requests.

Search Doctor: searchDoctor.py

The function searchDoctor takes in the connection to the oracle database and the cursor in order to execute queries. The function first asks for a doctor's employee number or the doctor's name. The

function checks to see if a integer was entered or a string. The function then asks for a start date and following that an end date in the form of YYYY-MM-DD. Both inputs are checked to be formatted

correctly and a month number cannot be out of the ranger 1:12. The search will query the oracle database for the doctor's information and all the prescriptions that the doctor has given within the

specified date interval.

Search Alarming Age: alarmingPatient.py