ETRT Database Project

CS 157A, Database Management Systems
Leonel De Anda, Alvin Jiang, Trevor O' Neil

i. Problem statement – analysis of the problem area to be solved

a. Entering data means that we will need to create appropriate tables to store information regarding patients and visits. According to the full requirements specifications, users will be storing their personal info and potentially their demographic info. As a result, we will need to be able to communicate with the database to store user data, update user data, and delete user data. Furthermore, optional data (such as middle name) may not be filled out by the user, so we need to ensure that if no data is filled, that the field Middle Name in the database contains the value null. Similarly, visit info contains the patient name and THC number, information about their problem, tools used, and future visit dates. We need to store visit data, update visit data, and delete visit data. Since we are only storing these pieces of information, many of our menu options will not be required to work (such as Interview) because it is outside out scope. We shall only work on the portion of the menu that requires patient data or visit data.

ii. Design of the solution – high-level architecture of your system, database design (table model in 3NF), UI design.

a. Our design of the system was to create different tables for each important part of the system. The main components (tables) are the patient and visit table. Every other table was used to reference data for the main tables. By implementing the tables this way, we achieved third normal form. The frontend was created with Java GUI libraries. We decided to use Java for our frontend because it allowed for less dependencies. We followed the layout given by the instructions and created something similar. Our backend was created through JDBC and MYSQL. Our program works by having buttons on the UI which, when clicked, call functions from our other classes and creates a SQL statement or query that returns/adds a response/entry.

iii. Description of the chosen technology stack for the solution and any other IT tools used (i.e. for the design, for teamwork, etc.)

a. The technology stack that we used was Java for the frontend because it allowed for our system to be simpler due to only using one language. For our database, we used MYSQL because it seemed to be more prevalent than Oracle. And to connect the frontend to the backend, we used the JDBC library

iv. Description of the implementation, (i.e. description of the tables, attributes, constraints, classes etc.). For Java, it is recommended to use Javadoc utility to generate documentation.

a. For our implementation, we created 8 different tables: Patient, Visit, Country, Zip, State, Work status, Occupation, and Demographics. For each table, we converted them into third normal form. The patients have name, Date of Birth, Gender, Phone, Email, Street Address, City, State, Zip, Country, Photo, Social Security Number, and Insurance. There were three constraints, of which all were foreign keys that provided the patients state, zip, and country. The demographics table had THC #, name, date, occupation, work status, educational degree, tinnitus onset, tinnitus etiology, hyperacusis onset, hyperacusis etiology, and additional comments. The Visit table had the Visit ID as the UID, visit date, THC, name, visit #, problems, category of problem, protocol, instrument used for treatment, rem, follow up info, additional comments, and the next visit date. The Demographics table has the information of the patient and any relevant medical history (hyperacusis/tinnitus). It also has foreign key relations to work status and occupation of the patient. The remaining tables are used for references. The country table had only its name and is a reference table. The zip had only its Zip code and is used as a reference table as well. The state table only has the state name and is a reference table. The work status table has an identifier of Yes or No, which identifies them as employed or not. The Occupation table has the occupation title.

   We created 9 different classes: TableViewer, Menu, SQLEntry, SQLEntryVisit, SQLHelper, SQLPatientLoader, SQLTableBuilder, SQLUpdater, and SQLVisitLoader. The TableViewer class lists all the tables in the database. The menu class contains all of our interfaces and functions for editing, adding, or deleting data using the functions from the other classes. The SQLEntry class adds entries to our database. The SQLEntryVisit class creates SQL statements that edits entries. The SQLHelper class provides exception handling for incorrect or invalid SQL statements/requests. The SQLPatientLoader class adds patients and

their information into the database. The SQLTableBuilder class creates all the tables in our database. The SQLUpdater class updates rows through calls to the database. The SQLVisitLoader gets the visits from the database.

v. Instructions to deploy and run your application.

a. Installation:

i. Download and install source code from Github located here: https://github.com/cs157AProjectTaskForce/patients_and_visits

ii. Download and install the MySQL database program.

b. Running:

i. Create a new project in eclipse.

ii. Import the files that were downloaded from github into the eclipse project.

iii. Ensure that your database is working and that an appropriate user has been made for accessing the databases.

iv. Run the SQLTableBuilder file to build the tables that the program will be using into the database.

v. Run the Main.java file to start the program.

vi. Listed contributions of each team member.

a. Leonel
   i. GUI and linking visit stuff with database.

b. Alvin
   i. Created database, reports, and relations.

c. Trevor
   i. Linking patient stuff with database and prepopulate scripts and some reports.