Documentation

Project Description: Design and implement a small veterinary management system using a mysql backend combined with a front end GUI. The database had the following specifications:

- Pets have a name, an owner, age, a medical record, a list of vaccinations (if applicable), and a list of appointments. There are four different kinds of pets: dogs, cats, birds, and fish. If applicable, pets also have a breed/variety.
- A medical record contains a list of diseases, and status for each disease (current or past).
- Appointments have a date, time, and client information, and a status (outstanding, and resolved).
- Owners have a name, address, and phone number.

We also had to implement the following functionality:

- See a list of appointments, with filters for: appointments for today, by owner, by date, by pets, outstanding, and resolved. Appointments status can be changed from outstanding to resolved.
- Search medical records by pet, owner, and animal kind.
- Check appointments and add appointments for a particular animal.
- Of course, add pet records to the database

Preliminary Development Plan: The database will be developed using MySQL Server. We will use netbeans to create a graphical user interface. Data will be placed in by the user with categories manually, and then the program will place the data into different subcategories. Whenever the data is searched for, it will check the appropriate subcategories. If the subcategory is searched, it will display everything in that subcategory. An item stored in will be saved as a node that holds the name, owner, age, medical record, vaccinations, list of appointments, and kind of pet. The appointment will be a node stored inside our node that can hold the date, time, and client information, and status.

Design and Architecture Approach: When we were designing this project we decided to use Java Swing in Netbeans to create our front end. Many of us had familiarity with this, so it would be simple for us to push and pull code through get without having to ask questions every step of the way. For the most part, the design tab was used to simply add components to the frame, and also allowed for quick references to the front end making database connection much easier. As for the database, we went with MySQL

Server since it was readily available to us. At the same time, MySQL Server has a command line which allows for quick table definitions and searches when trying to connect to the GUI. We decided to have one master table which houses most of the main information about the pets and called it pets_info. Then the other features such as medical records, appointments, and vaccinations could be in their own table with references to the master table. The master would have a reference to the owner table which is just owner information while the pet_type table is just an intermediate table with all the possible pet types.

Database Implementation: For the database we used a MySQL Server to create several tables in a database called pets. First is the pets_info table which has columns for an auto incrementing petID, a pet name, an owner name, an age in months, a pet type, and a breed. Then there's the owners table which has columns or auto incrementing ownerID, owner name, address, and phone number. Then the appointments table which has an auto incrementing appointmentID, an appointment date and time, a pet name, an owner name, and a status. Then the medical_records table has an auto incrementing recordID, a pet name, a disease name, and a status. Finally there's the vaccinations table which has columns for an auto incrementing vaccineID, a pet name, and a vaccine name.

GUI Implementation: Much the design of the tables there would be separate pages to handle inserting data such appointments, pet information, medical records, vaccinations, and owner information. There was also a separate page which would display the appointments for the day and provide a line for the user to enter an appointment ID which they would like to update. Extra logic was added to ensure that the pet and owner do not exist when creating a new record as well as checking if they do exist so we do not create duplicate records. User input is taken in through text fields while all the data from the databse is formatted and output to a text area which cannot be edited. If the user has to enter a specific format for the database to accept it, the label includes those format specifications.

Issues: The only major issue was ensuring that the GUI had all the necessary fields, text areas, and buttons to appropriately handle getting data for the database. There were several iterations of the GUI which saw several minor flow changes through the implementation process. Simultaneously that meant making changes to the database design to ensure we weren’t storing more information than was necessary in any one table. After making a couple design changes to the database, it was much easier for us to combine the two.

Testing: For testing purposes, the database was populated with several rows of data for each table to ensure that search filters were working as expected. Simultaneously, we used the GUI to create new records in the necessary tables, and then used the search filters to make sure the records were added correctly. We also ensured that the user was not allowed to do things like create a new record for a pet or owner that already exists.

Improvements: In terms of improvement, I think we could have done a little better in terms of collaboration. While we were sharing ideas back and forth, a sit collaboration session probably would have been nice at the very beginning. A time for us to look at what we would like to do for both the GUI and the database, and how the two will need to look in order to work well together. For the most part, we discussed what needed to be done individually, but it wasn’t until we started implementing that we looked at how we can make the two work together easily. This lead to some last minute changes that probably could have been avoided.