Vetilance

Project Analysis

Nicolas Metzler, Ruchi Patel

Table of Contents

Project Status Reports

Project Proposal

1.1 (Status Report 1)

Mobile App to Improve the Management of Chronic Disease in Cats and Dogs

1- Idea:

The development of a mobile app that helps improve the monitoring of feline and canine patients suffering from chronic diseases or health conditions.

2- Motivation/reason:

There are several chronic diseases or health conditions, such as kidney failure, diabetes, hyperthyroidism, hypothyroidism, cardiac insufficiency, etc. that affect feline (cat) and canine (dog) patients.

These chronic diseases require regular medication (daily or half-daily), regular monitoring and assessment by the owner and regular (every month or three-month) follow up and tests by the veterinarian.

Medications are often provided irregularly and signs of disease progression (frequency of urination) are often poorly monitored. Animals cannot communicate verbally and their wellbeing needs to be monitored through subtle changes in their behavior. Some tests can be carried out at home (blood and urine tests) but need to be interpreted carefully.

Poor management of these health conditions negatively affect the quality of life and the life expectancy of the patients. It also results in a more frequent need for emergency care treatment and hospitalization and the high associated costs.

Mobile apps could allow us to more closely monitor patients. It would improve communication between the owners and the veterinarian by providing a direct line of communication. The app could, for example, remind owners to provide medication and refill these medications. The app could remind owners about the need to schedule testing (e.g. blood glucose, urine test, blood pressure, etc.) on a regular basis. The app could help monitor early signs of disease progression (e.g. monitor drinking and toilet habits, appetite, activity level, frequency of vomiting, coughing, wheezing, etc.)

3- Scope:

- **a. Boundary of the topic:** The boundary would be a mobile app and a piece of software for the vet (Android Tablet/PC Software). The mobile app's user base will be the clients, while the veterinarian office will utilize its own version of the software.
- **b. Big picture:** The main components of the mobile app would be two interfaces for our software: One interface that would be installed on the client's smartphone and an interface that would be for the veterinarian.

  Veterinarian Interface Description: The veterinarian would access the interface and would see the list of patients monitored using this software and their status (green for "all clear", yellow for "need attention" and red for "follow up immediately").

  The veterinarian could click on any of the patient and this would open a "patient information card" with the name of the patients, basic information about the patient (sex, age, weight, the name and contact information of the owner, a brief description of the medical condition affecting the patient (e.g. hyperthyroidism) and a series of expandable menus, including
  1) "Prescription medication and Food" menu with information about the medications and prescription pet food that the patient is receiving/consuming and refill reminders or request for refill by owner,
  2) A "behavior monitoring": menu with a summary of abnormal behavior reported of the patient by the owner.
  3) An overview screen listing the patients and current medical status (Green, Yellow, or Red) as mentioned previously.

  Client Interface: The client interface would have basic medical information about the patient, contact information of the veterinarian, emergency contact information, and general health recording/checking systems.

  The interface would also have the same three menus as the veterinarian interface but with different content:
  1) Medical health and information screen. Which would list the patient, any conditions, and supplementary information that the Vet may need (such as behaviors exhibited).
  2) The information screen would also include medications and a reminder to refill the prescription or food and allow the owner to send in a refill

request directly from the app.
3) The behavior monitoring menu would allow the owner to note / record information about the behavior of the patient (e.g. increase urination frequency, more frequent drinking, restlessness of the patient, etc.).

The app would need to be able to access the phone function of the smartphone and directly call the veterinary clinic, it would have to be connected to a scheduling calendar that is managed by the veterinary clinic and an order management for the clinic for the refill of prescription medication or food.

The app would have to include some algorithms to predict the seriousness of behavior signs for different animals to send a signal (green, yellow, or red) to the veterinarian interface.

- **c. Inputs:** patient/owner information, drop down list of behaviors, submit button, type of medication and direction.

  Outputs: Push notifications, early warnings, reminders, list of possible conditions based on behaviors.
- **d. Functions:** The benefits of the app for the client would be better health outcomes for the animal, increase their longevity and well-being.

  The benefits of the app for veterinarians would be better follow up of patients, more regular visits of the animals, more regular refilling of prescription and booking of follow up testing and more streamlined and efficient communication with owners.
- **e. Limitations:** Limited to 2 people who are proficient at coding, and 2 other people with some experience coding and in computer science.

  Time constraint is a large one, with 4 other classes and a job, time is hard to come by to work for hours on any project.

  Another constraint is lack of experience in android development or coding. Up until this point classes were primarily focused on program development, and not super intricate or exhaustive.

  Background systems will be coded in a Java environment which should lend a hand in the background functionality being easier to code due to experience in the language.

.

Project Plan Structure

1.2 (Status Report 2)

Work Breakdown Structure:

Main subsystems: Client, User

- User-Interface/Menu navigation
- Reminder (Push notifications for medication/food)
- PDF/Text Document of the Pet's Medical Record/Info

Main subsystems: Veterinarian, User

- List of patients (A class for pet, species listed, and health indicator)
- Patient information card listing any relevant Pet Info
- Abnormal behavior notifications
- If something abnormal, the ability to send notification to the owner of said pet.

Important components: Database Components

- Finding a platform to code on for this function
- Coding the classes for the pets
- Designing UI elements
- Coding notifications
- Coding sub menus

Project Work Breakdown:

1. **Client Interface**
2. **Vet Interface**

Client Interface

1.1 Main Screen

1.1.1 Buttons to the other screens
1.1.2 Next appointment/checkup listed at the top of the screen

1.2 Behavior Tracker Screen
1.2.1 Drop Down list of behaviors
1.2.1.1 List of behaviors (Most common generally being shown first)
1.2.2 Add behavior button
1.2.2.1 Automatically track day/time it's recorded
1.2.2.2 Add to pet variables to calculate how often they show that behavior
1.2.2.2.1 How often is calculated in Pet's class and is a variable checked for health

1.3 Search Screen
1.3.1 Choose if you want to search by behavior, or search by condition (Drop down selector)
1.3.2 Have a search that returns matching keywords in the name or summary of the conditions

1.4 Pet Information Screen
1.4.1 Choose Pet list (Displays Pet Information that is chosen from list)
1.4.1.1 Can be reused for Vet choosing and displaying patients information
1.4.1.2 Pet name and species listed
1.4.1.3 Vector of Dog or Cat shown
1.4.1.4 Each one is a button
1.4.1.4.1 List of medications
1.4.1.4.1.2 Name of medication (Added by Vet)
1.4.1.4.1.3 Dosage Instructions (Take with/without food, Important for liquid medication)
1.4.1.4.1.4 Schedule of administration (Frequency/Time)
1.4.1.4.1.4.1 Basis for Push Notification Reminders
1.4.1.5 Summary of Health in General
1.4.1.5.1 Red/
1.4.2 Add/Remove Button
1.4.2.1 To remove, input pets name and click remove.
1.4.2.2 To add, input name, species (from drop down list (cat/dog))

1.5 Settings Screen
1.5.1 Push Notification Settings

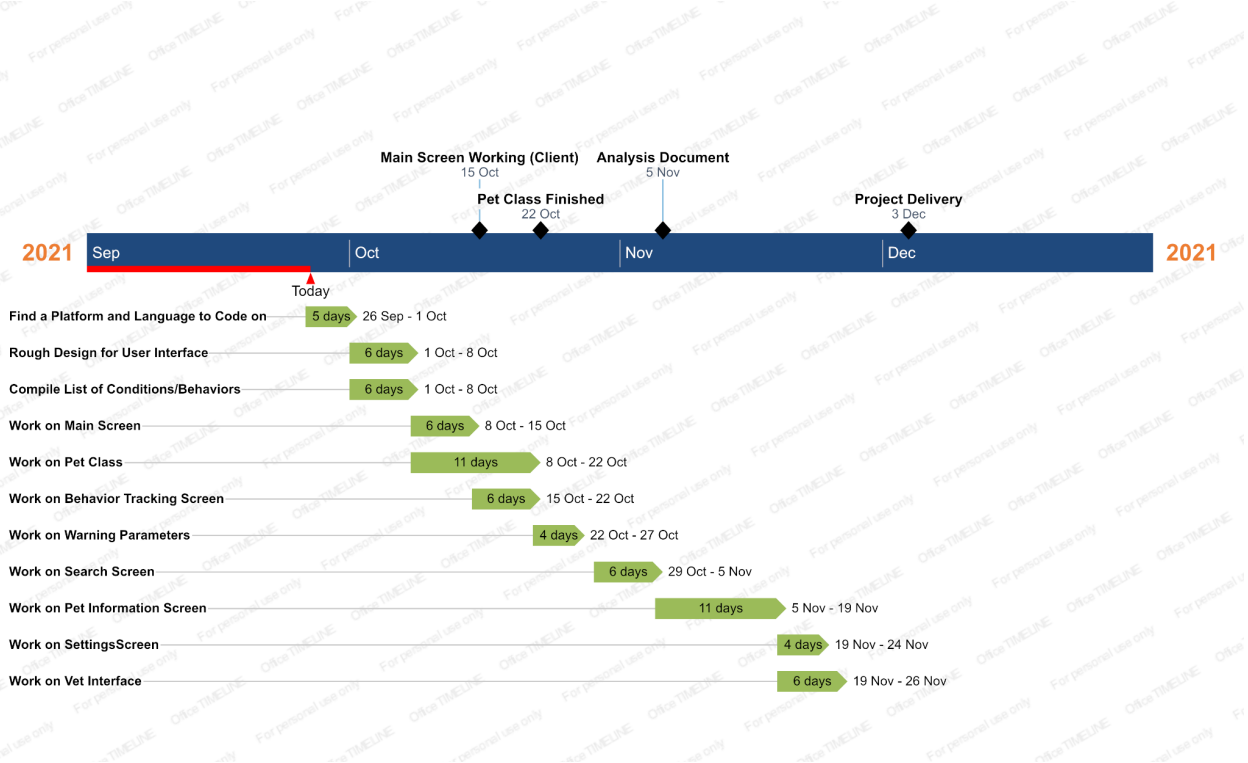
1.5.1.1 Food/Water Reminders
1.5.1.1.1 Set time for Food Reminders (2-3/day)
1.5.1.1.1 Water refill reminder automatically at beginning of Day

Vet Interface

2.1 Main Screen

2.1.1 Choose Pet list (Displays Pet Information that is chosen from list)
2.1.1.2 Pet name and species listed
2.1.1.3 Vector of Dog or Cat shown
2.1.1.4 Each one is a button
2.1.1.4.1 List of medications
2.1.1.4.1.2 Name of medication (Added by Vet)
2.1.1.4.1.3 Dosage Instructions (Take with/without food, Important for liquid medication)
2.1.1.4.1.4 Schedule of administration (Frequency/Time)
2.1.1.4.1.4.1 Basis for Push Notification Reminders
2.1.1.5 Summary of Health in General (each
2.1.4.1.5 Add/Remove Button
2.1.1.4.1.5.1 To remove, input Medication name and click remove
2.1.1.4.1.5.2 To add, input name, dosage amount, and any instructions

Gantt Chart:



Project Business Case
1.3 (Status Report 3)

Reason and Motivation

Value Added/Final Product Attributes

It is estimated that roughly 38% of households in the USA own a dog and roughly 25% own a cat. That means that over 48 million households own at least one dog and 34 million households own at least one cat. The canine population in the USA is over 76 million and the feline population is over 58 million. Average annual veterinary expenditure in the USA is $490 for dog owning households and close to $200 for cat owning households (American Veterinary Medical Association, accessed on 2 October 2021).

There are several chronic diseases or health conditions, such as, chronic kidney failure, diabetes, hyperthyroidism, hypothyroidism, cardiac insufficiency, hepatic lipidosis, etc. that affect feline (cat) and canine (dog) patients.
These chronic diseases or health conditions require regular medication (daily or half-daily), regular monitoring, assessment by the owner, and regular (every month or three-month) follow up and tests by the veterinarian.

Medication is often provided irregularly and signs of disease progression (frequency of urination) are often poorly monitored. Animals cannot communicate verbally and their wellbeing needs to be monitored through subtle changes in their behavior. Some tests can be carried out at home (blood and urine tests) but need to be interpreted carefully. Poor management of these health conditions negatively affect the quality of life and the life expectancy of the patients. It also results in a more frequent need for emergency care, treatment, and hospitalization and the high associated costs.

One example case would be Chronic Kidney Disease otherwise known as CKD. The prevalence of CKD is estimated to be around 0.2% in dogs and 1.0-3.0% in cats. The occurrence of CKD increases greatly with age, especially in cats. The prevalence of CKD can reach 35-80% in geriatric cats (> 12 years of age) (Brown et al., 2016). Millions of cats and dogs consequently suffer from CKD in North America and CKD is one of the major causes of hospitalization and premature death for feline patients. While less prevalent than CKD, other health conditions also affect millions of canine and feline patients in North America.

Consumer Market

The project focuses on the development of a mobile app to monitor and better manage chronic health conditions in feline and canine patients. It will be developed for pet owners (owners of pets with chronic health conditions) and for the veterinarians who are treating these patients.

A mobile app would allow owners to more closely monitor patients. It would improve communication between the owners and the veterinarian by providing a direct line of communication and information. The app will, for example, remind owners to provide medication and refill these medications. It could also remind owners about the need to schedule testing (e.g. blood glucose, urine test, blood pressure, etc.) on a regular basis. It would monitor early signs of disease progression (e.g. monitor drinking and toilet habits, appetite, activity level, frequency of vomiting, coughing, wheezing, etc.) and relay them to the veterinarian.

The benefits for the pet owners would be able to provide better health outcomes for their animals. The app could help increase their longevity and well-being. The app would also limit the needs for costly emergency care since changes (degradation) in the condition of the patient would be identified in a timely fashion. The benefits of the app for veterinarians would be better follow up of patients, more regular visits of the animals, more regular refilling of prescription and booking of follow up testing and more streamlined and efficient communication with owners.

Target Market/Sector

The app could be distributed by veterinary clinics and a monthly subscription would be charged or the veterinary clinic could offer the app for free as part of a care and medication package for the patient.

How the Concept Came About

If you have ever owned a pet you know how difficult it can be to keep up with caring for them, especially in their later years. You have to remember medication, directions to administer it, sometimes dosage, and then even multivitamins. The app we're creating looks to address these concerns to take that load off of the pet owner.

The communication of behavior data with the vet also helps facilitate care of the animal. This way the owner may not need to take their pet to the vet all the time. The vet can remotely monitor and see if the behaviors warrant further investigation. It also allows the pet owner to ask the vet if the behaviors reported are of concern.

Market Study

Synopsis

A large number of veterinary management software exists. Most of the software is stand-alone or cloud-based. These are essential tools for operating a veterinary practice. The software systems are only accessed by veterinarians and the veterinary hospital staff.

While our proposal is also software related to veterinary care. It is more centered around the pet owner, while providing the vet with useful information to improve the quality of care for the pet.

Other Veterinary Software

VitusVet is a cloud-based software that integrates within common veterinary management software and that offers a client mobile app that helps pet owners manage their pet's health care, while also enabling communication with the veterinary practices. VitusVet is mainly focused on simple monitoring and communication using two way messaging. It doesn't appear to offer the features that are planned for our mobile app.

PetDesk is an app designed to help clients and vets communicate with each other about pet management in the most efficient way possible. The app allows clients to have regular reminders for upcoming appointments, checkup/vaccination dates, access medical records, test results and prescriptions. The vet and their client can engage in a 1:1 conversation as well. PetDesk has the most amount of similarities with our app.

Petable is an app that contains all the pets' health details like breed type, existing health condition, etc. It also contains reminders about medications and upcoming appointments/checkups. It doesn't allow for 1:1 conversion with the vet through the app, but has their contact information.

Royal Veterinary College also offers free tracker for health conditions, including a pet epilepsy tracker and pet diabetes tracker. The pet epilepsy tracker has features such as a seizure log and medication log which allow the sharing of information with the Royal Veterinary College, which is specific to the UK.. These trackers are useful tools but they are not as comprehensive and integrated as what we are planning.

There are dozens of mobile apps that have been developed to help gather and store information like the pets breed, weight, microchip number, vaccination history, medications, planned visits to the veterinarian, weight, etc. Most apps are similar. They allow storage of information and reminders to be set up. However, they are not true health management tools. None of these apps offer an interface for veterinarians, or offer a seamless communication between the veterinarians and clients.

DogHealth offers a paid version of its mobile app that allows communication with veterinary practice but the scope of this feature appears to be very limited.

Sylvester AI is an innovative Canadian company that creates predictive healthcare products powered by artificial intelligence to improve the health of animals across various species. Their mobile app **Tably** is a remote patient monitoring system that uses AI to detect subtle facial cues to predict changes in the health and welfare of cats. The system can notify the owner and veterinarians when issues arise. The app allows the client to take pictures and send updates to the veterinarian through the remote monitoring tool. Tably would likely be a major competitor for our mobile app. However, our approach focuses on the active monitoring of existing conditions using well established approaches.

Vetrax offers a mobile app that connects to the Vetrax behavior monitoring system that uses a lightweight sensor placed on the collar of dogs. The sensor captures detailed data around the clock and transmits these data to the veterinarian through the mobile app.

Comparison Chart:

	VitusVet	**PetDesk**	**Vertax**	**Our App Idea**
Communication with Vet	Yes	Yes	Yes	Yes
Access to medical records	Yes	Yes	No	Yes
Medication details and reminders	Yes, access to medical details	Yes, access to medical reports and appointment reminders	No	Yes

Success in the Market

Features such as notifications/reminders for checkups and medication, the communication without the need for calling, syncing of information with associated vet, and the remote care features that come with this.

The reminders for appointments and medication are very useful, especially if you lead a busy life, or are prone to forgetting things. People use their phones all day everyday, so a notification will get their attention for important things such as this.

The communication aspect will free up the phone lines, and allow a much more accessible way to communicate for the clients. Clients can ask even general questions about their pets' health without the need to call the clinic and wait for the receptionist. This includes attempting to schedule an appointment with the clinic. Instead of trying to get through to a receptionist, the chat feature can be used, or alternatively an appointment booking feature.

The synced information will help the vet determine if action is needed. They can reach out to the pet owner either by chat feature, or by phone in urgent cases. The information can also be easily brought up during a health checkup to help the veterinarian better treat the patient.

All of this kind of bundles into the benefit of remote care. It will greatly benefit the pets to have a system in place that can help their owner take care of them better. It also facilitates a way that pet owners can ask general health questions about their pet.

As for the business indications, we would know that our product has fulfilled the business case if:

- We can provide an easy to use app which tracks the health of animals
- The app has a feature for monitoring the medicine deliveries
- Push notifications are sent to the users device reminding them of medication

Project Requirements

1.4 (Status Report 4)

Stakeholders in the Project

In terms of internal stakeholders we have us as the students. Our stake in the project is our marks. In terms of external stakeholders, there aren't any. You could say the professor is one since he has expectations of our project. But otherwise for this project, there is not an external stakeholder.

For the sake of the project however let's suppose the external stakeholders will be Veterinarian Clients/Pet Owners. And suppose we are a company investing money into the project.

Formal Requirements

System

- **Main System**
- Summary
    - The system formally is a Veterinarian/Pet Owner Application. It contains many subsystems, however we will be outlining the few key ones here. It acts as a sort of culmination of all the elements in tandem/harmony.
- Limitations
    - Since we're talking about the system as a whole here. If one subsystem malfunctions it could cause a catastrophic failure
    - Must make sure that Pet medical records be kept confidential
        - Would be unacceptable if one pet's information is displayed in place of another.

Sub-Systems/Main Functions

- **Pet Information System**
- Summary
    - The Pet Information System is the core of our application. It is the basis for all the information/organization of information within our system. All important information will be contained within this sub-system, in a Class of sorts. You could almost say this is the system itself since it's such a foundational feature of this application.

- Limitations
    - Since this system/class is so integral to our application, if anything fails, it could be catastrophic for our app.
    - Since it also relies on our major back-end functions, if one of those fails, it may cause the Pet Information System to fail (either minorly, or catastrophically)
- Functions
    - The Pet Information System will contain all the information for each pet
    - It will act like a Class. Each instance of it being a different pet.
    - It will contain functions such as, but not limited to
        - getMedications() which returns an array of medications specific to that pet
        - getName()
            - returns the pet's name
        - getAge()
            - returns the pet age
        - getSpecies()
            - returns the pet species

        - getMedication(name)
            - searches the Array of medications for a specific medication, and returns the information
        - getMedicationsAll()
            - returns the whole medication Array
            - useful for medication screen, and monitoring all the medications by the Vet
        - addMedication(medname, dosage, adminInstructions, adminTimes)
            - calls the Medication Information System, and appends the Medication to the Array of Medications if successful.
                - If failed, return the reason (invalid dosage, etc.)
        - rmMedication(medname)
            - removes medication from medications Array
                - garbage collector gets rid of it

        - getPetStatus(behavior Array, timeFrame)
            - Calculates how at risk your pet is based on behaviors in the past week
            - Green: Behaviors are healthy, nothing of concern
            - Yellow: Please continue to monitor behaviors
            - Red: Contact your Vet immediately

                - Call is expedited to Behavior Tracking Subsystem
        - getBehaviors(timeframe)
            - Returns behaviors that happened in specified timeframe (last x days)
            - Useful for Vets to check on behaviors they may see as concerning, and how long they've been occuring
            - Also useful for calculating the status of the pet
        - addBehavior(behavior)
            - Calls the Behavior Class to create one
            - Add behavior with specified time to behavior array
            - Call is expedited to Behavior Tracking Subsystem

- **Medication Information System**
- Summary
    - This system is used by the Veterinarian to add/remove medications for a specific pet. For example, the Vet will have a set of requirements to fill in for the medication.
- Limitations
    - Requires a very specific input for each parameter of the medication.
    - An issue that may have to be overcome is how to deal with multiple doses per day.
- Functions
    - Will also act like a class
    - Will be only be called in the Pet Information System
    - Each medication will need a
        - Name of the Medication (string)
        - Dosage (integer/float) + (string)
            - “1.2 mg”
        - Administration instructions
        - Administration times (array of times)
            - Must check for a valid time input and format
            - Or alternatively use a drop down menu to choose to ensure valid input

- **Behavior Tracking System**
- Summary
    - Will contain a list of behaviors a pet can exhibit and track at what time, how frequently they happen, and store them for future reference. Will also have a function that returns how at risk (generally) a pet is.
- Limitations

    - There is no possible way to cover every single behavior an animal can do, based on how specific such a behavior is
        - Example: Pet barking or meowing whenever the lights turn on
        - This can be overcome however by including additional notes
- Functions
    - addBehavior(behavior, time, notes)
        - Takes specified behavior, the system time when submit is clicked, and a string value which will have additional notes if the Pet Owner chose to include them
    - riskAssessment(behaviorArray, timeFrame)
        - Takes the arrays in the behavior, and returns a value (Red, Yellow, or Green) based on if the behaviors are any cause for concern

- **Condition/Search System**
- Summary
    - Will store all the datasets for possible conditions
        - Allows for a clean and concise information pages on each condition
    - The storage of all the information plays into the search system by allowing it to access a specific portion of the conditions information
        - Example: If we're searching by behavior, we can access the behavior portion of the condition to compare
- Limitations
    - There has to be an element to simplicity to fit it all in
    - Need to rank conditions on an order of severity
- Functions
    - The Condition/Search System will contain all the information for each pet
    - It will act like a Class. Each instance of it being a different Condition
    - The conditions will not be able to be edited, since it is rare for a new condition to be discovered suddenly
        - This can be updated by the developers
    - It will contain functions such as, but not limited to
        - getByBehavior(behavior, species)
            - Returns all associated conditions that contain that behavior and are specific to that species, or can happen in both species
        - getByName(searchString)
            - Returns all associated conditions that contain that the whole or part of the string
            - Essentially a normal search function
        - getByAge(age, species)

- Returns all associated conditions that are most common within that age range of the species

Non-Functional Requirements

Clean Modern UI

- Medical themed (light blue and white design palette)
- Use light blue as a basis for the rest of the alette
- Menus are easy to navigate as an extension of a clean UI

Profile for each Pet

- Translucent outline of each pet species
- Name, Age, and General Health displayed
    - Red, Yellow, and Green as previously mentioned

In ease of use, have all the elements on a profile clickable.

- Ie anywhere you click within the Pet Info box brings you into the pet info screen.
- Makes it easier than needing to click a specific element such as a “learn more” button

General Technical Requirements

- Language

- We have taken time and consideration into choosing our programming language. This is due to the sheer importance of our application. When offering a service that is in the health industry, a simple error or delay can alter many lives regardless if it is human lives or not. This means before choosing our programming language we had to find the Qualities in the language that we chose. Because we are generally working with a database we needed a language that could communicate with a database as it stands. This code should allow us to do basic functions such as update data on a database, or even retrieve the data we had placed within the database. This led us to our first choice of programming languages which is SQL.
- SQL is commonly used to manage relational databases and carry out numerous amounts of operations. For example with SQL you can, Create a database, Create Tables, Select and Extract data, update data, delete data, drop tables and databases and insert new data into a database. This comes as an asset due to our apps needing to perform tasks as mentioned before. Some of the tasks we are required to do is, “getName()”… Medications, PetStatus, Age and Species. This is because we will need to retrieve information that has already been placed

into our database. Another Important task we require SQL to perform is allowing us to “addBehavior()” This is important because information will be required to be submitted into our system whether by an employee or customer (End User). We will also need to perform tasks such as “rmMedication(medName), this is important because we will need to remove information from the database we created.

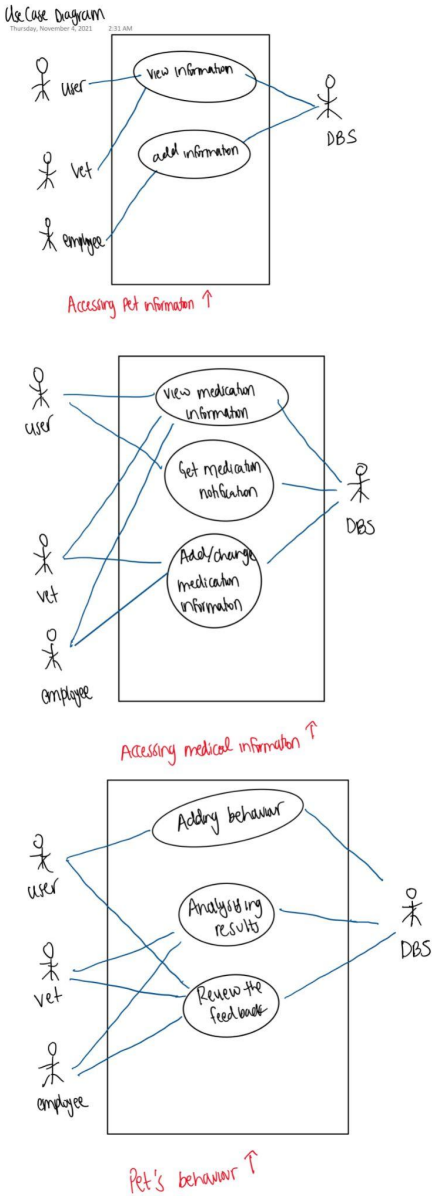
- IDE
- The IDE we choose to program in is Microsoft SQL management studio. This is because SSMS is an integrated environment for managing any SQL infrastructure. This means that we will have the ability to access, configure, manage, administer and develop all components.
- Language
- We also take note that we will require this application to be on apple devices hence why we will be tying the entire application in Objective C. Objective C is the primary language for developing for OS X and iOS. This allows us to use Objective C as an extension for our database system that we coded in SQL.
- IDE
- The IDE that we will use for objective C is Swift (Objective C and Swift are two different languages, Swift is not the IDE for Objective C). It would be preferable to use Swift . Swift is a refined and efficient programming language that was developed by Apple in order to create apps for iOS devices. It boasts on its ability to give developers freedom while also being easy to use.

- Reuse
- If we were to reuse any code we would be copying code from Dev Groups such as, DB-Pros, Inc. This is because they have made plenty of applications such as “Hospital Database Software”. This is a medical software that makes managing patient medical data easier.
- HardWare
- The Hardware we will require is a Lenovo Think System SR550 Rack Server. This is because the application will need a particular place to store and process information whether over the cloud or physically punched in. We chose this hardware because it is affordable and it has Flexible storage configurations, and centralized automated management. The price of this hardware is $2499.41.
  It would be preferable to use a cloud-based server and bypass the need for a physical server. Cloud native applications deliver faster time to market, higher scalability, simpler managementHosting the app would be more economical and flexible on a cloud server like AWS.
  AWS offers a collection of tools designed to help build, test, configure, and release cloud-based applications like our app. The building of the app will be

done on a Jenkins server, and the prototype from Jenkins will go through a test phase, which is configured with AWS Device Farm to test the application on real devices. AWS CodePipeline will provide the orchestration and helps automate the build and test phases

Use-Case Diagrams

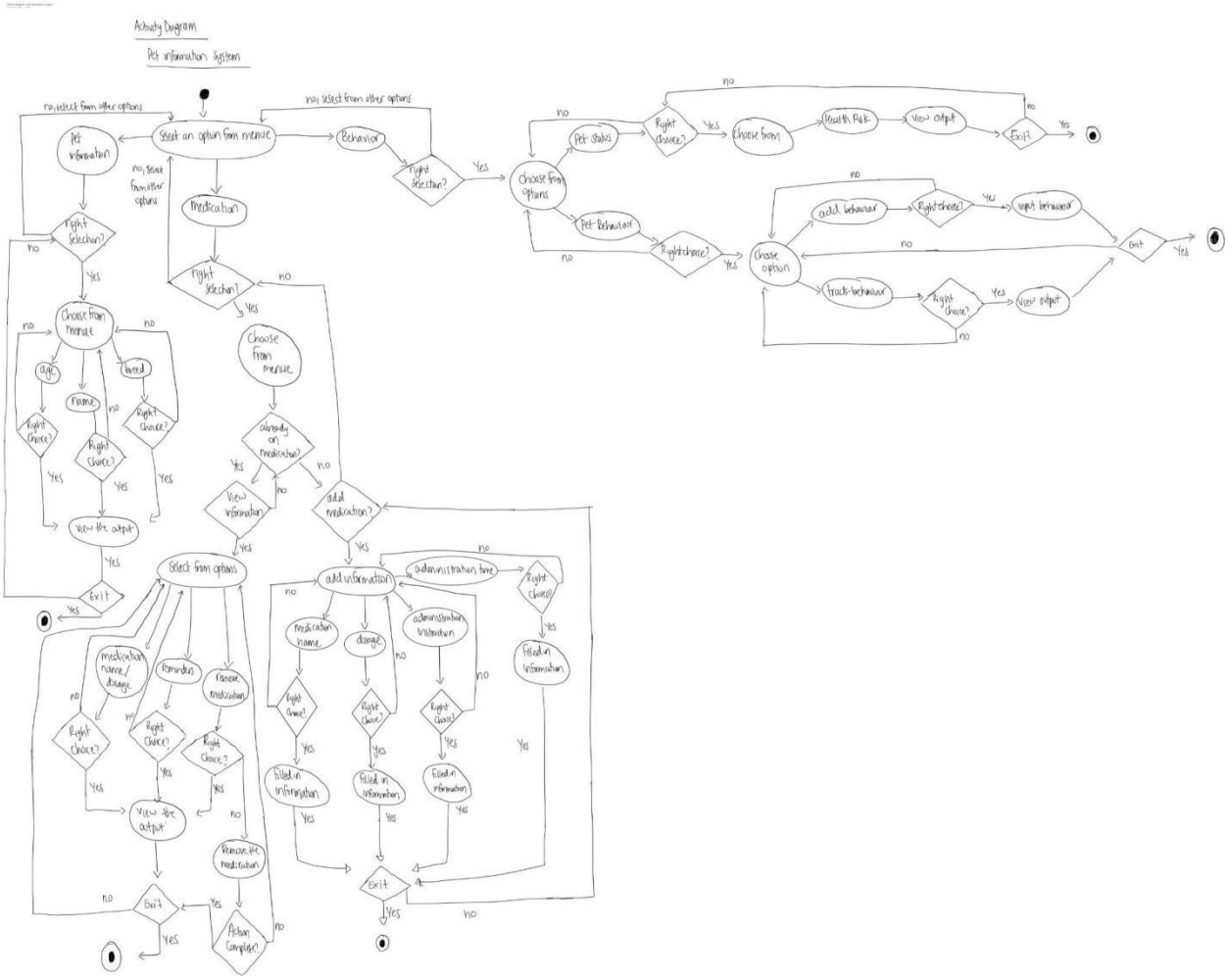
In the use case diagrams below, three scenarios are illustrated, pets information, medication and behavior tracker. The first system pets information. All the parties involved can view the information, but only the vet or its employee can change it.In the second scenario, the client can view the information and get medical notifications, but only the vet or their employee can make any changes to medical information. Lastly, the user can enter the behavior, and see the comments that the vet or its employee might have posted after analyzing the behavior. The vet and their team would analyze and review the behavior of their pet from a pattern created by the inputs.



Use case diagram for pets information, medical records and behavior

Activity Diagrams

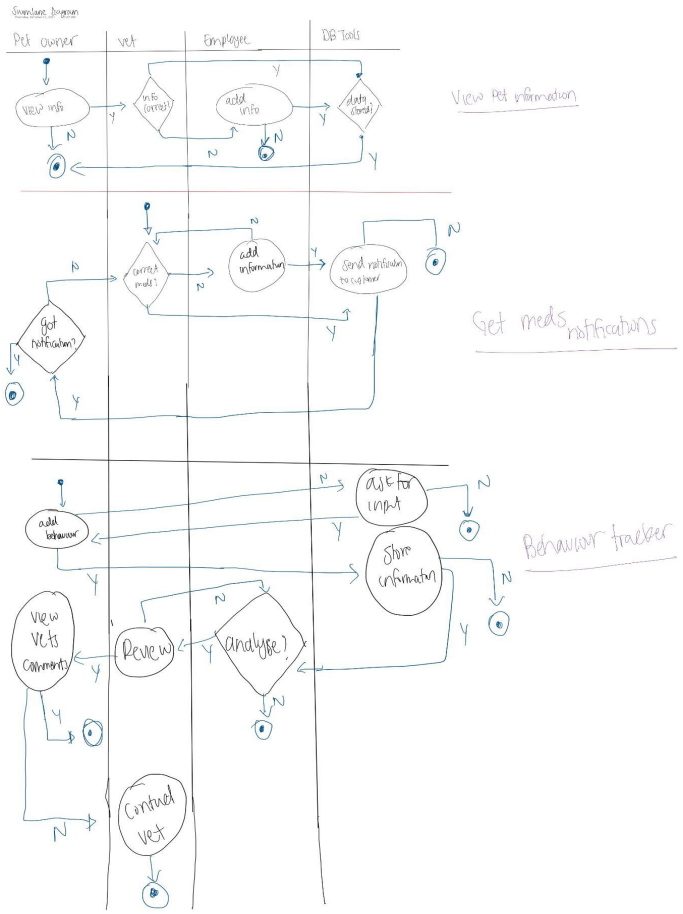
The activity diagram below describes three main subsystems. The first system is when clients want to enter pet information, the second subsystem describes accessing medical information of the pet, and the last subsystem describes the behavior sub system. In the pet information, the user can access the pets information. In the case more information needs to be added, they would need to privately contact the vet offline. Secondly, in the medication subsystem, the user can view all the necessary mediation related information which is prescribed to their pet. The vet can also add any medication information necessary and set notifications reminders as well. Lastly, the user can enter and analyze the behavior of their pet from a pattern created by their own input.



Activity diagram for accessing pets information, medical records and behavior

Swimlane Diagram

In the swimlane diagram below, three scenarios are illustrated, viewing pets information, medication and behavior tracker. Similarly to the activity diagram above, the first system is when clients want to enter pet information. In the pet information, the user can access the pets information. In the case more information needs to be added, they would need to privately contact the vet offline. Secondly, in the medication subsystem, the user can view all the necessary mediation related information which is prescribed to their pet. The vet(or their staff) can also add any medication information necessary and set notifications reminders as well. Lastly, the user can enter the behavior. The vet and their team would analyze and review the behavior of their pet from a pattern created by the inputs.



Swimlane diagram of accessing pets information, medical records and behavior.

Analysis Document

Risks

Our project at this stage faces many risks. Mainly due to the size of work and documentation needed to be completed. We've found due to other commitments, such as other classes, work outside of school, scheduling the group, and documentation to come into the way of actually getting the project done. This means that our risks have significantly increased over the course of this project and semester.

Product Size

- The size is doesn't seem to be an issue in theory, however that may change when we begin coding and see the true scope of the project

Development Environment

- There are plenty of tools available to code on the android platform
- Java is a pretty simple language to learn/get familiar with
- The main issue here is getting used to both coding on an android centred IDE, and the figuring out the functionality of an android application/how to implement functions
- Android Studio is a notoriously difficult IDE to use, not lending itself well to new users. Although it is the most popular Android development software.

Staff Size and Experience

- None of us have experience with Android development
- The majority of us haven't coded in Java in at least a year

Number of Developers

- The number of people on this project, and time constraints do seem adequate
- However with the inclusion of major pieces of documentation and conflicting schedules it is a very real challenge
- Perhaps if this was a full time effort, or even 20 hours/week it would be much more realistic for this amount of people

Schedule Risk

- This is perhaps one of the largest issues that we are currently facing
- The budgeted time for work is very uncertain with all different schedules and workloads across classes
- This creates uncertainty that the project will be completed on time, and in full functionality

Performance Risk

- This plays into the Schedule Risk aspect

- Due to the above risk it is uncertain that the software will be in it's full functionality
- Even in full functionality, it's uncertain if there will be time to properly debug the software for fatal errors

Support Risk

- This is one of the items of least concern
- Due to the Re-Use of code and (hopefully) simplicity of the software. We imagine that it should be relatively easy to update/change

Risk Table

Risk	Probability (1-10)	Impact (1-10)	Risk Estimate (Prob x Impact)%	RMMM Risk Mitigation Monitoring & Management
Product Size	**3**	**4**	**12%**	**Mitigation** - Set a realistic size of the project **Monitoring** - Be mindful about the scope of the project in terms of our current skills in development **Management** - Scale down the project to provide basic functionality
Learning the Development Environment (Android fundamentals and Android Studio)	**9**	**10**	**90%**	**Mitigation** - Use resources such as Youtube to aid us in learning the development environment/process **Monitoring** - Progress on learning the development of android apps **Management** - Use the processes mentioned in Mitigation and Monitoring to keep up. Assign 2 members to focus solely on coding

Coding and Implementing the Software (Prototyping)	**6**	**8**	**48%**	**Mitigation** - Have a set schedule of time allocated to work on developing the software itself **Monitoring** - The progression of the software and how far away the submission date is. **Management** - Allocate time to two people to code, and two people to work on the project status reports. Effectively splitting the team into two roles
Polishing and Debugging the Software (Completion)	**7**	**5**	**35%**	**Mitigation** - Debug during the development process to prevent any fatal errors in the first place **Monitoring** - Testing the software as it's being developed, and testing for any anomalies **Management** - Set aside a couple days at the end of the semester to go through the code extensively to comment, debug, and ensure the software is acceptable
Further Documentation (Status Reports and Final Report)	**7**	**7**	**49%**	**Mitigation** - Have roles assigned early each week for each portion of the Status Report that needs to be completed. **Monitoring** - Keep

				tabs on how the Status Report is progressing through the week **Management** -

Re-Use

Re-use is a very important concept in Object Oriented Programming (OO). It allows programs to function much more efficiently, and eliminates the need to use a new redundant process in your code, which has already been written elsewhere.

Our product takes advantage of this and uses it to improve efficiency. Some examples of this are the re-use of code structure, the implementation of classes, and using similar components.

The re-use of code structure comes in the form of the classes. The classes are built off functions of each other. In the case for example where we need to calculate the overall health of a pet, the process is quite similar to taking the symptoms and finding associated conditions. We can take the similar portions of code and tweak it towards it's needed functionality.

In terms of the implementation of classes, the base class that this whole app will be based on is the Pet/Information Class. This class stores the specific pet's information, and can then call all of the necessary sub-functions/classes to calculate whatever data is needed.

Then we also have the re-use of components. The best examples of these would be both visual, and interactive elements. These include the interactive buttons, drop down menus, and other UI elements. Essentially for all these the values are about the only component of each that needs to be tweaked.

All the reusable content of our software is also very useful to avoid possible debugging issues. If we know a function works as expected in one aspect, we can pinpoint the reason it won't in another portion of the software.

Quality Assurance

Software engineering process is outlined by some core principles which are followed throughout the industry. These principles provide structure and help keep the practice consistent throughout. The two main stages in software engineering are process and practice level.

We have done an extensive amount of planning and modeling of our application in the project so far. We have utilized some planning principles to help us guide through developing the best map for the use case of the completed application. Additionally, we have also utilized some modeling principles to help us analyze and design software that is as detailed as possible. Modeling principles provided a visual guide for all the work, technical and non-technical, which needs to be completed for a successful application.

Software Methodology:

Planning principles:

During the planning process, we utilized some agile methodologies where we made sure to be realistic with our expectations for this project. We were not only limited on time, but during few moments, it felt like we were a group of 3, not 4. Due to this, we were realistic with how much we can accomplish within a short period of time and limited amount of skills. Additionally, we were also quick to adapt and adjust as we further defined the plan for this application. There were a lot of things to consider when it came to coming up with a plan and adjusting accordingly. For example, the big one is the limited amount of resources and time, the closer we get to the due date, the more adjustments we make to the requirements for this application.

Modeling principles:

For creating the model, we applied a combination of agile and waterfall models. To elaborate, we made an effort to keep requirement models as simple as possible while describing the problem/scenarios. Additionally, we also made sure to build them in a way that we would be able to make any changes if the situation takes a turn at the last minute, especially since we value feedback from our professor as his feedback might make us strive to make drastic changes in a short time.. Being adaptable is very important and a good indicator of the effectiveness of our application.

Insort, we focused on being agile when it comes to planning for our project's requirements.

Appendices

Appendix I

3.1 (Weekly Meeting Minutes)

Week	Meeting Minutes	Additional Details
1	30	- No group leader selected as of yet - Different ideas proposed
2	60	- Still no definite group leader - Worked on portions of the status report
3	120	- Met and worked on Status Report 3 as a group
4	60	- Meeting during the reading week - Portions revised and assigned to each member of the group - Started work on the status report
5	-	- Analysis Report questions refined and divided between the group members - Done over Discord, and confirmed through email or messages

Appendix II

3.2 (Work Hours)

Embedded Excel Sheet

Date	Duration	Members
17/09/2021	0:30	Ruchi, Clara, Nick
22/09/2021	1:00	Ruchi, Clara, Nick
23/09/2021	0:30	Ruchi
25/09/2021	1:00	Nick
26/09/2021	1:30	Nick, Ruchi
25/09/2021	1:00	Clara
26/09/2021	0:30	William
5/10/2021	2:00	Clara, Nick
5/10/2021	3:00	Ruchi
12/10/2021	1:00	Nick, Ruchi, Clara, William
13/10/2021	1:30	Nick
21/10/21	2:30	Nick
15/10/21	1:30	Ruchi
21/10/21	30	Ruchi
25/10/21	3:00	Nick
28/10/21	3:00	Nick
28/10/21	10:00	Clara
1/11/2021	10:30	Nick
4/11/2021	3:00	Ruchi
3/11/2021	1:00	Nick
4/11/2021	2:00	Ruchi
5/11/2021	4:00	Nick
5/12/2021	6:00	Ruchi

Total Hours/Member

Member	Total Hours
Nick	32.5
Ruchi	20.5
Clara	15.5
William	12.5

Appendix III

3.3 (Software/Hardware Details)

Algorithms/Utilities Used and OO

The main algorithm applied to the software/application is our search/calculate functions based off of the behaviors of the pet. This function is not only the basis, but it is constantly in re-use in one form or another.

You have one application of the algorithm where we use it to search through an array of possible conditions. The other application of this same algorithm is to take into account just how serious these symptoms are, and if they warrant further investigation.

Then we also have the other algorithm for medication. We need to systematically remind the owner of when they're scheduled to get a new prescription for their pet. In order to do this, we take the prescribed date, the amount prescribed, and the dosage. Then calculate the estimated date where the medication is at a critical level (10-20%), and remind the owner accordingly.

Three SW Eng Principles Relevant to this Effort

1. Be agile

We are working according to an established plan but we are trying to be very agile and adaptable and focus on the economy of action.

2. Focus on quality

We have limited time and every element that we develop and program needs to be of high quality so that the work is well done at once and we do not have to experience too many bugs that would result in software failure.

3. Create work products that provide value for others

Because we are all working remotely and are not working with a very experienced team leader that can coordinate and address issues, we have to develop the different parts of thinking about other members of the team that are tasked with related or dependent tasks.

Software/Hardware Details:

After doing extensive research the hardware of choice is a Lenovo IdeaPad 3i. Our criteria was based on the demands of the Language used and their corresponding IDEs. We are mainly using the language of C and SQL. We plan on creating an android and OSX app, which is based on Java and C.

This means that IDEs such as Microsoft SQL(For Query tools), Android Studio and Apple Swift are our main choices. Microsoft SQL states that the IDE requires minimum 6 Gb of available hard disk space, 2.0GHz processor speed and 4GB memory. They are very lenient with the processors that can be used with the IDE so we decided to use Intel because it is the largest processor company so far.

We also chose the Idea Pad because it is capable of dual booting the mac OS giving us the ability to use swift. This is great because we are able to save money instead of buying another device. Altogether these resources allow us to progress smoothly as a company.

Similarly, in terms of software, we would need to deploy the software on a cloud service so we all can work together. We have decided to take advantage of the numerous free services available like Github at this stage. Our thought process is that GitHub would allow us to host applications for free.

HR Effort Size:

Eva Calculations

Thursday, November 4, 2021 12:23 PM

Total hours spent By the group = 50.5 hrs

Individual contributions:

Clarca = 15.5 hrs = 15.5/50.5 = 31%
William = 1.5 hrs = 1.5/50.5 = 3%
Ruchi = 14.5 hrs = 14.5/50.5 = 29%
Nick = 28.5 hrs = 28.5/50.5 = 56%

EVA Analysis:

Person-month calculations:

4 team members, 6 hrs/week/person.

hence: 6 x 4 = 24 hrs/week (team)

∴ 24 hrs x 6 weeks = 144 hrs

For person-month, assume an FTE works 40 hrs/week

∴ 40 x 4 = 160 hrs.

$\frac{144}{160\text{ hrs}} = 0.9$ person-month effort size

Computations:

At week 6/12, we should have been $\left(\frac{32}{49}\right) = 65\%$ of the way into project implementation. Currently, we are at 25% of the way into implementation since status report takes a lot of time.

Schedule performance index (SPI) = $\frac{BCWP}{BCWS} = 25\%$

Hence, Schedule variance = BCWP - BCWS

$\frac{65}{100} \times 144\text{hrs} = 93.6\text{ hrs (BCWS)}$

BCWP = 0.25 (93.6 hrs)

= 23.4 hrs

SV = BCWP - BCWS

= 23.4 - 93.6

= -70.2

∴ our team is 70.2 hrs behind our ideal schedule timeline.

For resource planning, we are planning on allocating more group members towards the implementation of the project as we need to put a lot of effort in that area. As you can see from the appendix, we put a lot of effort behind planning and brainstorming our project conceptually. But now, we need to allocate our resources towards implementation. The last couple of weeks, we also had issues managing the group dynamic as not all members decided to pull their share of weight. And to address that, we basically need to have one person doing twice the work in case one of our team members decides to not contribute.

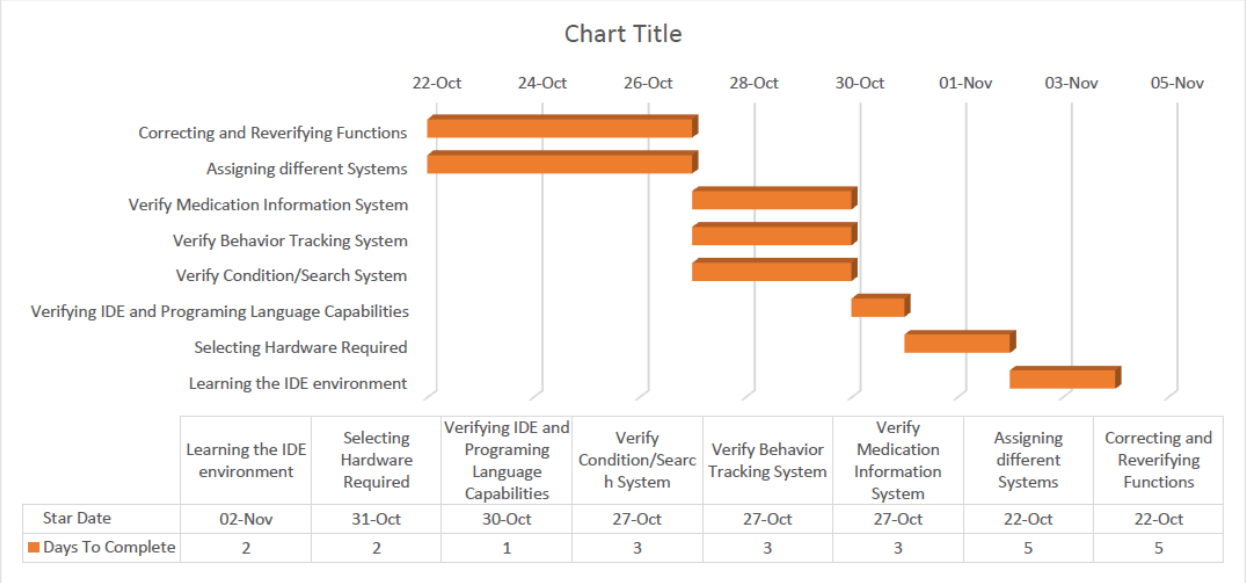
Meeting Minutes	Additional Details
30	- No group leader selected as of yet - Different ideas proposed
60	- Still no definite group leader - Worked on portions of the status report
120	- Met and worked on Status Report 3 as a group
60	- Meeting during the reading week - Portions revised and assigned to each member of the group - Started work on the status report
-	- Analysis Report questions refined and divided between the group members - Done over Discord, and confirmed through email or messages

This is our chart before, but now, we plan on making it look like, and only have one group member take care of the reports.

Meeting Minutes	Additional Details
-	- Implementation of the app

-	- Implementation of the app
-	- Implementation of the app

Gantt Chart



	Learning the IDE environment	Selecting Hardware Required	Verifying IDE and Programing Language Capabilities	Verify Condition/Search System	Verify Behavior Tracking System	Verify Medication Information System	Assigning different Systems	Correcting and Reverifying Functions
Star Date	02-Nov	31-Oct	30-Oct	27-Oct	27-Oct	27-Oct	22-Oct	22-Oct
Days To Complete	2	2	1	3	3	3	5	5