Drug Drawer - Medicine and Vaccine Management Application

Final Report

MIE350 - Design and Analysis of Information Systems

Matteo Bruzzese - 1004108967
Salma Dessouki - 1003790418
Nolan Mandel - 1004161836
Momo Mitrovich - 1004353255
Dante Mondelli - 1003870643
Chris Overvelde - 1003912741
Maria Papadimitriou - 1003913624

Table of Contents

1.0 Executive Summary

In the ever-changing world of personal healthcare applications, there was a need identified in the realm of personal management that has yet to be tended to in the readily available market. No accessible application has been made available to monitor and track both an individual's vaccine and medication history. Vaccines serve as the key moments in one’s medical history. It is commonly agreed that this information is not as accessible as it should be. The issues that can arise from common non-adherence to prescribed medication scheduling arises from human error and memory slips. Drug Drawer was developed as an ambitious effort to satisfy the gap presented of a medical tracking system that both maintains past records and current medication schedule.

All available vaccines in Canada were maintained and updated within a relational database containing dosage rounds, legally binding vaccines and exact serial codes which are preserved with government standards. Medications were collected and stored in a relation in the database which are stored with associated information relevant to each type. Users are assigned with all pertinent information needed to hold an account, including if they are a parent account responsible for other child accounts.

Four dynamic pages presented are the Medicine Drawer, Vaccine Drawer, Add Medicine and Add Vaccines. The requirements that should be fulfilled by the transactions that occur with the database should be: users can create accounts for themselves and their children (as well as the editing of such information at any time), users are able to navigate through medication list and add selection, system must inform users when to take medication along with the recommended dosage, and users must be able to add selected vaccines to their overall list. Requirements regarding other aspects such as the privacy of user information, language accessibility being English and consistency of the overall design were all taken into deep consideration. They helped to target certain design heuristics that warrant a better user response by making the application more efficient, effective, easy-to-use and have a great utility.

A strong and iterative process was carried out throughout the full development of the application in order to try and fix certain obstacles faced with the implementation of the user-centered system. All challenges faced were met with an analysis of the overall system architecture and front-end user interaction to target specifically the issue at hand while maintaining a direction of total refinement in the development process.

The team learned from all of the challenges faced, whether technical or group-focussed, to allow for a fully effective system to become developed and honed into filling the need that was presented . It is now believe wholeheartedly by all of the members of the team that the gap has been filled and the need has been tended to.

2.0 Motives and Context

With the technological advancement in recent years, the use and popularity of healthcare applications has increased drastically. As of 2018, the number of health related applications has jumped to a number of approximately 318,000 on the top app stores worldwide -- nearly doubling the number of applications in 2015 [1]. For example, *Weight Watchers* is an application focusing on nutrition, whereas *ZocDoc* is used to link patients to doctors based on user insurance [2]. With activity trackers such as the *Apple Watch* or *Fitbit*, users are able to track their personal metrics involved in fitness at any point throughout a given day [2].

The health application of focus in the context of this project is medication and vaccination management systems. Many individuals that are prescribed pills often find it difficult and are inconsistent in managing their medication. In 2018 alone, 89,000 premature deaths were caused in the US due to hypertension, which all could have been avoided if for proper medication adherence [3]. This lapse in memory included 1 in 2 people missing a dose, 1 in 3 forgetting when their last dose was, and 1 in 4 not receiving their re-filled medication on time [3]. Additionally, 55% of all Americans take at least one regularly prescribed medication [4]. Implementing a medication and vaccination management system into a web application not only assists in the convenience to the patient but also assists in the scheduling, consumption, and replenishing of required doses.

There is no current implemented software that accurately displays in a timely manner the needs of the current vaccine-conscious and pill-regimented person. This information is in dire need to be made accessible for the general public as per the issues expressed with overall medical knowledge. This application serves the purpose of assisting users in the awareness and scheduling of health requirements to improve their overall well being.

3.0 Requirements

3.1 Requirements Discussion

After coming to a final decision on the objectives of the website and the gap the team wanted it to fill, the next step was to determine the requirements.

The requirements for the drug medication system were determined through a team brainstorming session. The question that was trying to be answered was, what does the system need to do in order to properly meet its objectives. The requirements that the team came up with enable the system to operate as designed and allow it to meet all of its objectives.

3.2 Functional Requirements

Functional requirements are requirements that pertain to the processes the system must be capable of performing, and the information that the system must be able to contain. Below is a list of the functional requirements for DrugDrawer, divided into the categories of *must haves, should haves,* and *could haves*.

Must Haves: These are requirements that are crucial for the website to properly operate and to perform as the team designed. Missing any one of these requirements would lead to the website being a failure.

Users must be able to log into their account with a username and password. All child accounts must be associated with an account from a user over the age of 18. Users must be able to navigate through a medication list and select which medications they want to add to their personal list. The website must inform users when to take their medications. The website must display all of the vaccines in the users drawer (i.e. the vaccines that the user has had in the past). Users must be able to navigate through a vaccine list and select which vaccines they want to add to their personal list. The website must display a table that shows a history of the vaccines they've taken, whether it's a recurring vaccine or not, the last date they took it, whether the vaccine the live, the diseases the vaccine prevents, the date of next vaccination, and the status.

Should Haves: These are requirements that would be very beneficial in aiding the system in achieving its goal. A system should have all of these functionalities, but it is not absolutely necessary for it to.

The website should inform users the recommended dosage when notifying them to take their medication. The website should say if a medication requires a prescription or not. The website should update the remaining amount of a medication a user has, after the user takes the medication.

Could Haves: These are requirements that would be nice for the system to have, but would not drastically affect how well the system meets it purpose.

A could have requirement for the website would be to have chat windows where users can interact with medical professionals.

3.3 Non Functional Requirements

Non functional requirements are the quality attributes, design, and implementation constraints that must apply to the entire system. The team agreed upon a set of non functional requirements to ensure a quality user experience.

Operational: The system must be accessible on all Google Chrome, FireFox, and Internet Explorer. A consistent interface must be present across all pages.

Performance: The system must be available for 24 hours a day.

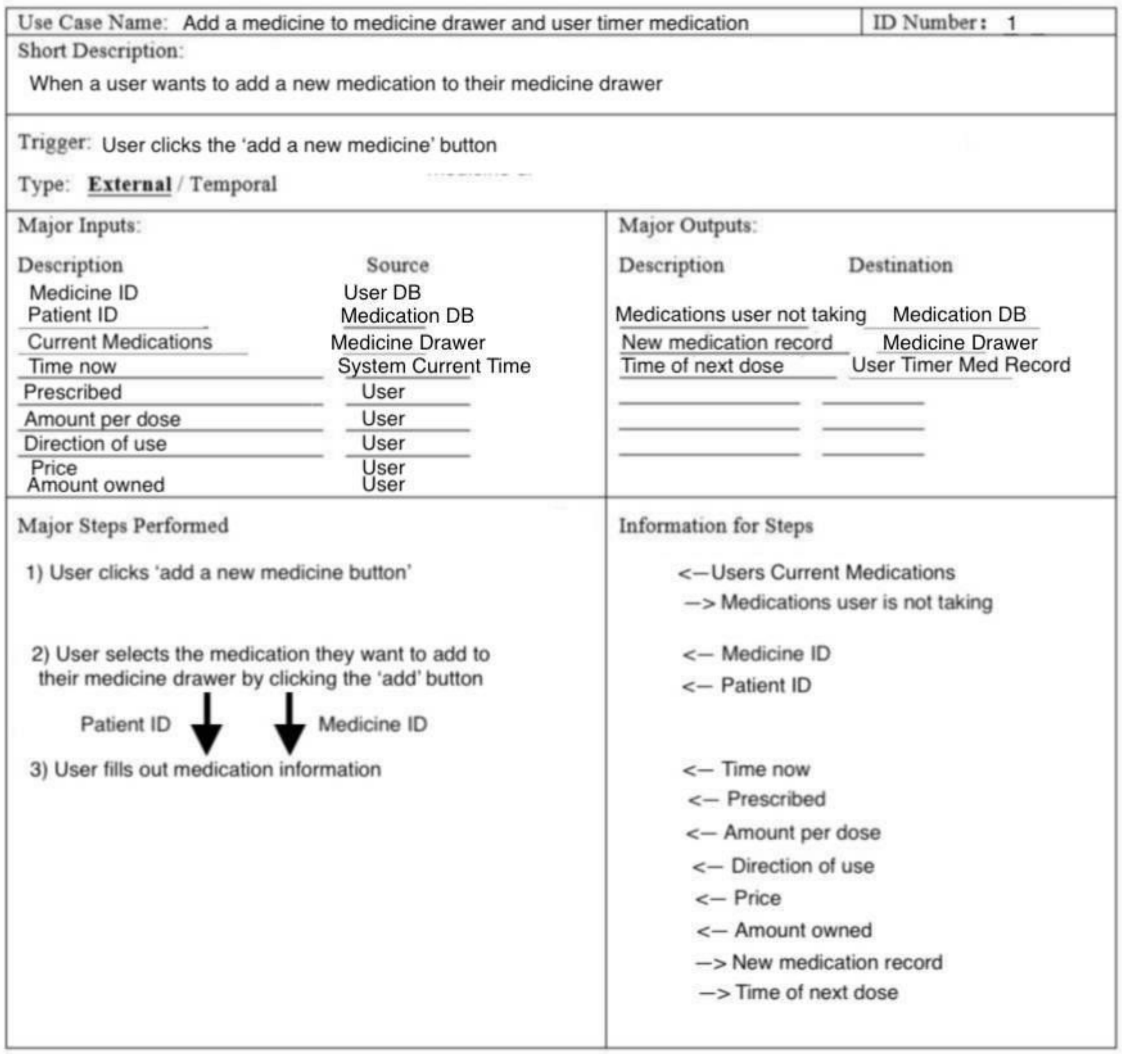
Security: Only users can see their personal medication and vaccine history.

4.0 Use Cases

The three use cases shown and described below were selected because the team believes they demonstrate the key functionalities of the website. Each use case below describes a different task that the user can perform on the website.

4.1 Use Case 1

Use Case 1 shows the task of a user adding a medicine to their medicine drawer. This will be one of the more commonly used and important functions of the website, as this is how users can personalize their medication list. The user must input information regarding the medication, and the website will add it to their list and will start notifying the user when they need to be taking it.



Use Case Name: Add a medicine to medicine drawer and user timer medication | ID Number: 1

Short Description:
When a user wants to add a new medication to their medicine drawer

Trigger: User clicks the 'add a new medicine' button

Type: [✓ External] / Temporal

Major Inputs:

Description	Source
Medicine ID	User DB
Patient ID	Medication DB
Current Medications	Medicine Drawer
Time now	System Current Time
Prescribed	User
Amount per dose	User
Direction of use	User
Price	User
Amount owned	User

Major Outputs:

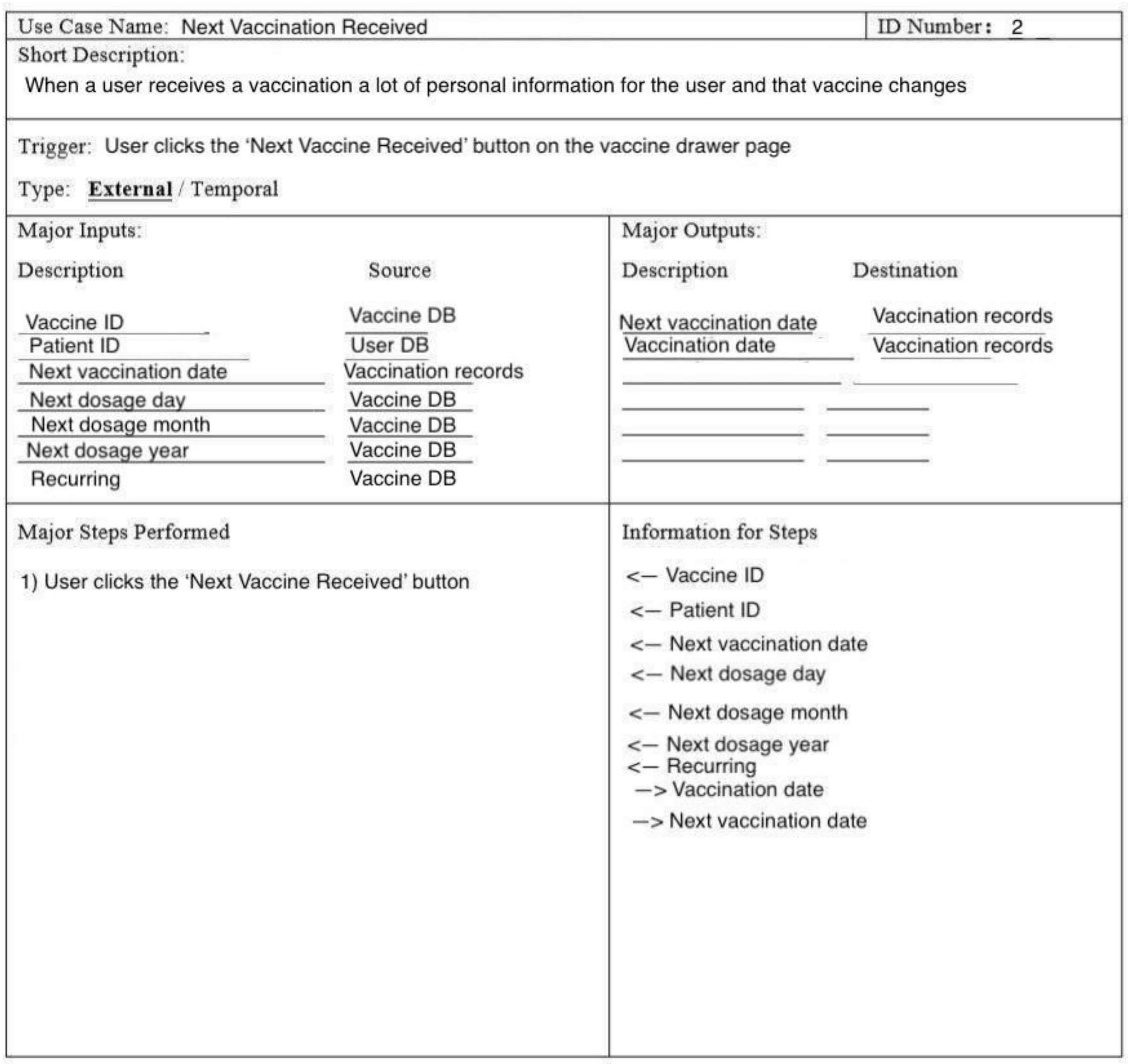
Description	Destination
Medications user not taking	Medication DB
New medication record	Medicine Drawer
Time of next dose	User Timer Med Record

Major Steps Performed	Information for Steps
1) User clicks 'add a new medicine button'	<—Users Current Medications —> Medications user is not taking
2) User selects the medication they want to add to their medicine drawer by clicking the 'add' button Patient ID ↓ ↓ Medicine ID	<— Medicine ID <— Patient ID
3) User fills out medication information	<— Time now <— Prescribed <— Amount per dose <— Direction of use <— Price <— Amount owned —> New medication record —> Time of next dose

Figure 1: Adding a medicine to medicine drawer and user timer medication Use Case

4.2 Use Case 2

Use Case 2 deals with when a user has received a vaccine. The user doesn't need to input any information to the website, as the system will handle everything internally. If the vaccine is not recurring, then there will be no date put for the next vaccination date, but if it is a recurring vaccine then the website will display when the user needs to get that vaccine again.



Use Case Name: Next Vaccination Received

ID Number: 2

Short Description:
When a user receives a vaccination a lot of personal information for the user and that vaccine changes

Trigger: User clicks the 'Next Vaccine Received' button on the vaccine drawer page

Type: **External** / Temporal

Major Inputs:

Description	Source
Vaccine ID	Vaccine DB
Patient ID	User DB
Next vaccination date	Vaccination records
Next dosage day	Vaccine DB
Next dosage month	Vaccine DB
Next dosage year	Vaccine DB
Recurring	Vaccine DB

Major Outputs:

Description	Destination
Next vaccination date	Vaccination records
Vaccination date	Vaccination records

Major Steps Performed

1) User clicks the 'Next Vaccine Received' button

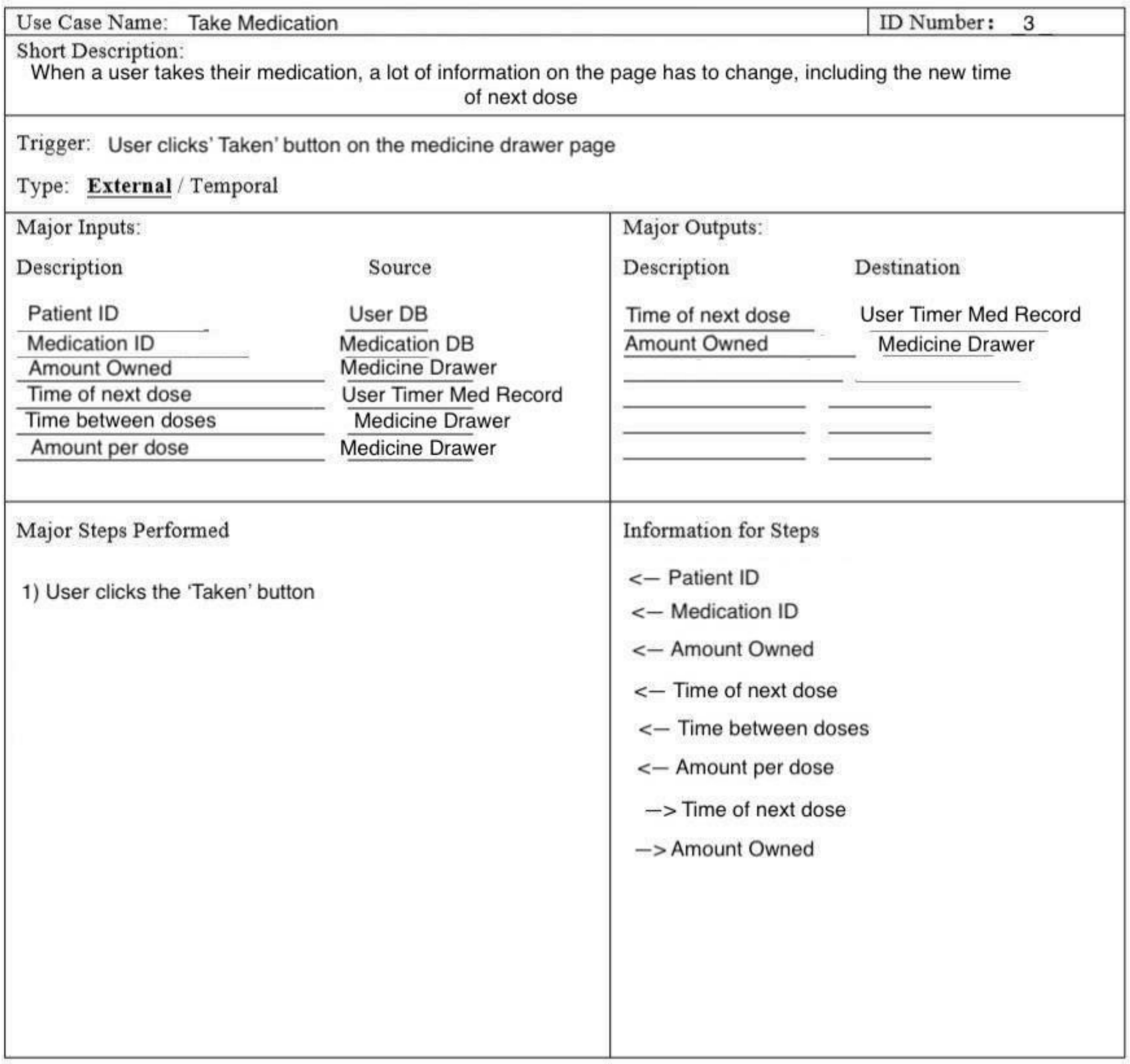
Information for Steps

<— Vaccine ID
<— Patient ID
<— Next vaccination date
<— Next dosage day
<— Next dosage month
<— Next dosage year
<— Recurring
—> Vaccination date
—> Next vaccination date

Figure 2: Create, add, modify profile Use Case

4.3 Use Case 3

Use Case 3 deals with what happens internally when a user takes their medication. When the user clicks the ‘Taken’ button for a given medication, a lot of information is updated, such as the time of the next dose, and the amount the user has left. This is important because if this information is not updated, then the user will not know when to take their medication next, which is one of the main goals of the system.



Use Case Name: Take Medication | ID Number: 3

Short Description:
When a user takes their medication, a lot of information on the page has to change, including the new time of next dose

Trigger: User clicks' Taken' button on the medicine drawer page

Type: **External** / Temporal

Major Inputs:

Description	Source
Patient ID	User DB
Medication ID	Medication DB
Amount Owned	Medicine Drawer
Time of next dose	User Timer Med Record
Time between doses	Medicine Drawer
Amount per dose	Medicine Drawer

Major Outputs:

Description	Destination
Time of next dose	User Timer Med Record
Amount Owned	Medicine Drawer

Major Steps Performed

1) User clicks the ‘Taken’ button

Information for Steps

<— Patient ID
<— Medication ID
<— Amount Owned
<— Time of next dose
<— Time between doses
<— Amount per dose
—> Time of next dose
—> Amount Owned

Figure 3: User takes medicine or refills medicine Use Case

5.0 Design Principles and Software Development Methodologies

Key design principles were considered before implementation of the prototype. These principles included efficiency, ease of use, good utility, and effectiveness. The efficiency of the system is significant as it provides clients with a convenient and practical system that can be integrated into their schedule. The ease of use of the system is important because a system with an easy navigation is more likely to attract users. Good utility of the system allows users to access a wide range of medical information based on their profile. Lastly, the effectiveness of the system will be bore through consistently update medical information whether it be medication or vaccinations taken by the user.

In order to achieve the aforementioned design principles, the web application was created such that the client must be able to use the web application while all these principles are followed diligently. The web application was developed using HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JS (Javascript). The layout of the front-end of the system was completed with the intention of maximizing the number of users due to easy navigation and ease of use. Therefore a minimalistic approach was taken which incorporated the combination of both turquoise, navy and white colourings consistent throughout the entire application. Additionally, the web application was given a brand name and logo *Dose Drawer* such that users would associate these colours with the application when seen. It is imperative that the interface acquire these principles as it assists in the learning curve faced by new users exposed to the application.

The back-end functionality of the website was implemented using a combination of code written in Java, a microsoft database build in microsoft access, and data manipulation using SQL (Structured Query Language). These tools interact with Apache Tomcat and consist of the server component of the web application.

The entire system integrates with one another through the data manipulation. More specifically, requests sent by the user from the front-end interface are sent to the back-end server where transactions occur. These transactions include data pulls from the database, database updates, and additions to the database. Once these transactions occur, feedback is given back to the user to inform them that their desired request was completed, acting as a dynamic page.

The system contains only one microsoft access database and a total of six relations; PatientProfile, MedicineDrawer, MedicineList, VaccineRecords, VaccineList, UserTimerMedicines. The database contains a total of 23 attributes across all relations, and at least 50 tuples were implemented in each relation.

The technical structure of the application prompts users to log into the website as soon as the website is initialized. For the sake of this project, users are unable to make profiles for themselves and instead must be created through the database by our team. Creating (adding) profiles to the database was taken out of the scope of the project as it was found less significant than the actual functionality of the rest of the dynamic pages. When the login is prompted, users will enter their username and password. If this information is matched with a current tuple in the *PatientProfile* database, the user will successfully enter the system. Conversely, if these criteria do not match, they are redirected to an invalid login page. Users are then able to either update their vaccines (more specifically declare they have received their next vaccine), add a new vaccine, update their medication (more specifically declare that they have received their dose and request for a refill), and add a new medication to their vaccine drawers and medicine drawers respectively. The *PatientProfile* relation has one primary key, *patientID*. The *MedicineList* and *VaccineList* relations each have primary keys of *medicineID* and *vaccineID*. The relations *VaccineRecords* and *MedicineDrawer* contain both the *patientID* and *medicineIDs (or vaccineIDs)* as foreign keys in the *MedicineDrawer* and *VaccineRecords* relations. These relations signify a record of a patient and the health item they are seeking to track. The combination of the *patientID* and the *medicineID* is then used to add a transaction in *UserTimerMedicine* to return the time of next dose for the user. Overall, the 6 relations account for the dynamic pages of Vaccine Drawer, Medicine Drawer, Add Medicine, and Add Vaccine.

6.0 UML Dynamic Diagrams

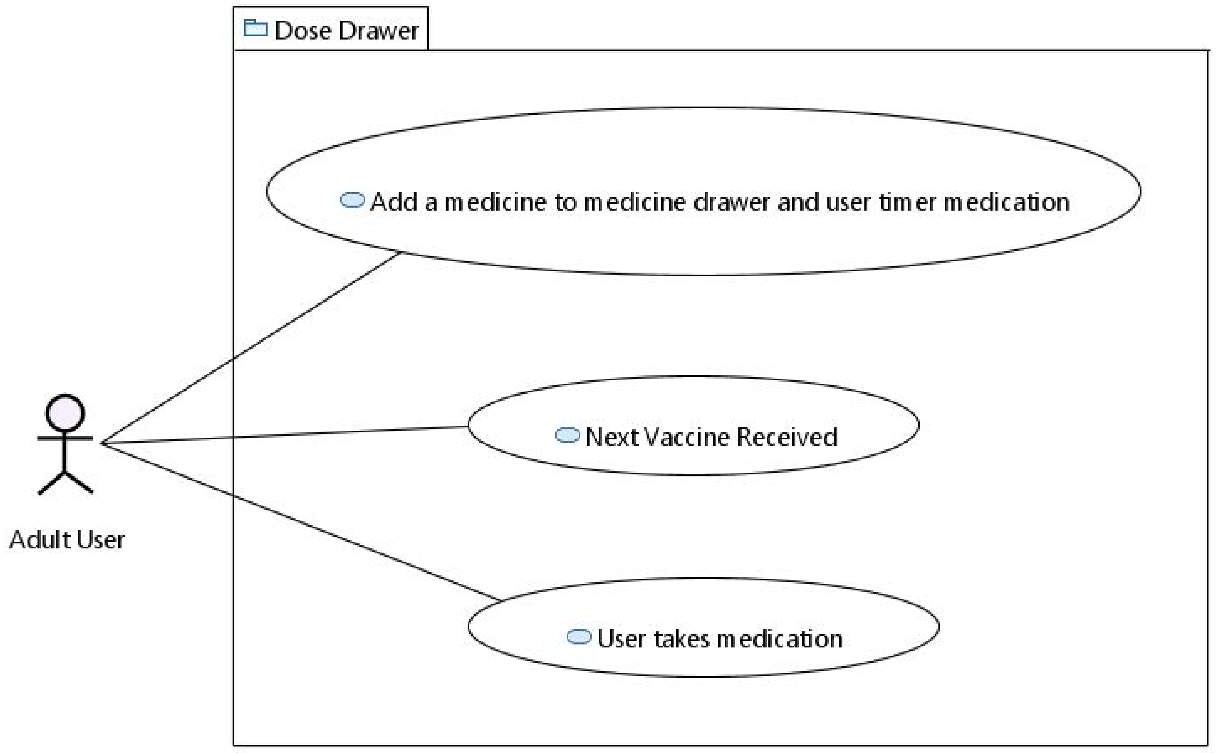


Figure 4: UML Use case diagram

This diagram is meant to show all actors involved in the progression of the system. Specifically, this use case diagram shows the types of interactions a user can have with the system to achieve certain tasks. This includes adding medicine to their drawer, reporting taken vaccines, and taking medication.

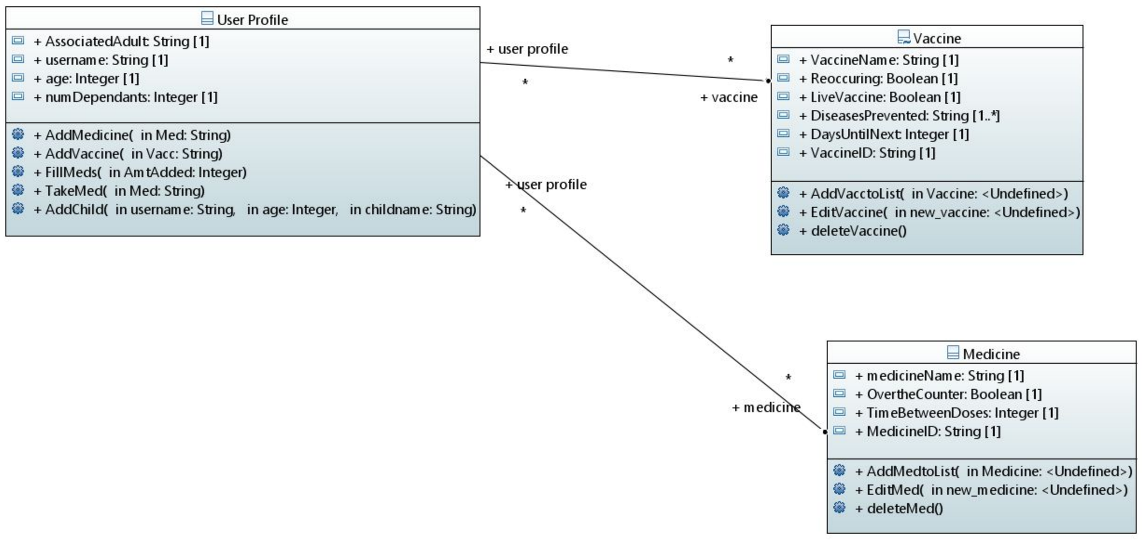

User Profile
+ AssociatedAdult: String [1]
+ username: String [1]
+ age: Integer [1]
+ numDependants: Integer [1]
+ AddMedicine(in Med: String)
+ AddVaccine(in Vacc: String)
+ FillMeds(in AmtAdded: Integer)
+ TakeMed(in Med: String)
+ AddChild(in username: String, in age: Integer, in childname: String)
+ user profile
*
*
+ vaccine
Vaccine
+ VaccineName: String [1]
+ Reoccuring: Boolean [1]
+ LiveVaccine: Boolean [1]
+ DiseasesPrevented: String [1..*]
+ DaysUntilNext: Integer [1]
+ VaccineID: String [1]
+ AddVacctoList(in Vaccine: <Undefined>)
+ EditVaccine(in new_vaccine: <Undefined>)
+ deleteVaccine()
+ user profile
*
*
+ medicine
Medicine
+ medicineName: String [1]
+ OvertheCounter: Boolean [1]
+ TimeBetweenDoses: Integer [1]
+ MedicineID: String [1]
+ AddMedtoList(in Medicine: <Undefined>)
+ EditMed(in new_medicine: <Undefined>)
+ deleteMed()

Figure 5: Class Diagram and Relationships

This type of diagram, class diagram, shows a static model of the relationship among the classes in the system. This diagram shows attributes and methods belonging to a specific class, as well as graphical signaling of how these classes interact with each other. This class diagram shows the relationship between the user profiles and the server side pre-stored medicines and vaccines. Each profile can have a relationship with every medicine or vaccine once and the relationships between the two are shown in the vaccine and medicine record pages.

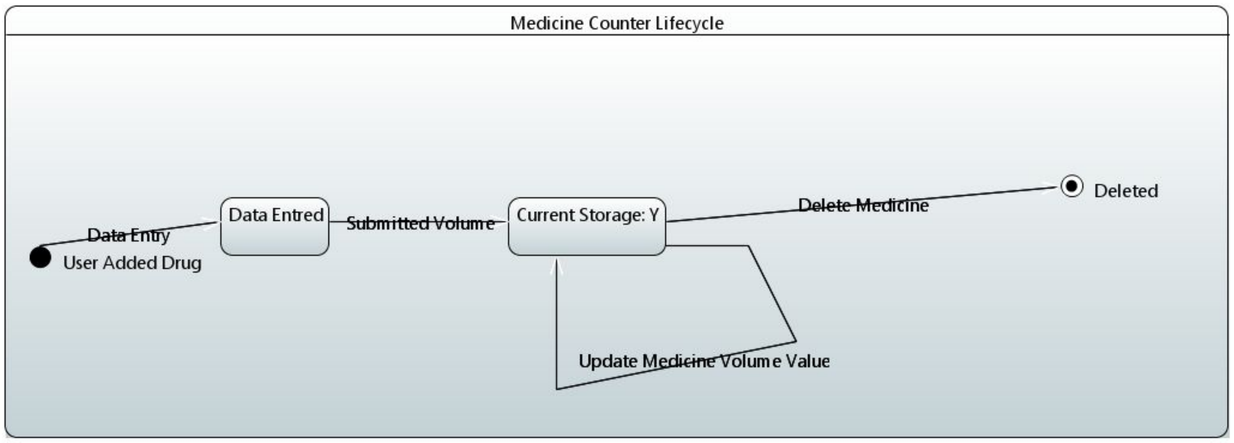


Figure 6: State Machine Diagram of the Medicine of a User

This diagram is meant to show the state change of an object over the lifecycle of that object. This object is a medicine in a medicine drawer. After instantiation, the current storage value varies however the user deems to manage their own inventory. This constantly updates the storage state value for every interaction. To end this endless cycle of taking medicine and refilling it, the user is able to delete deprecated medicines from their list.

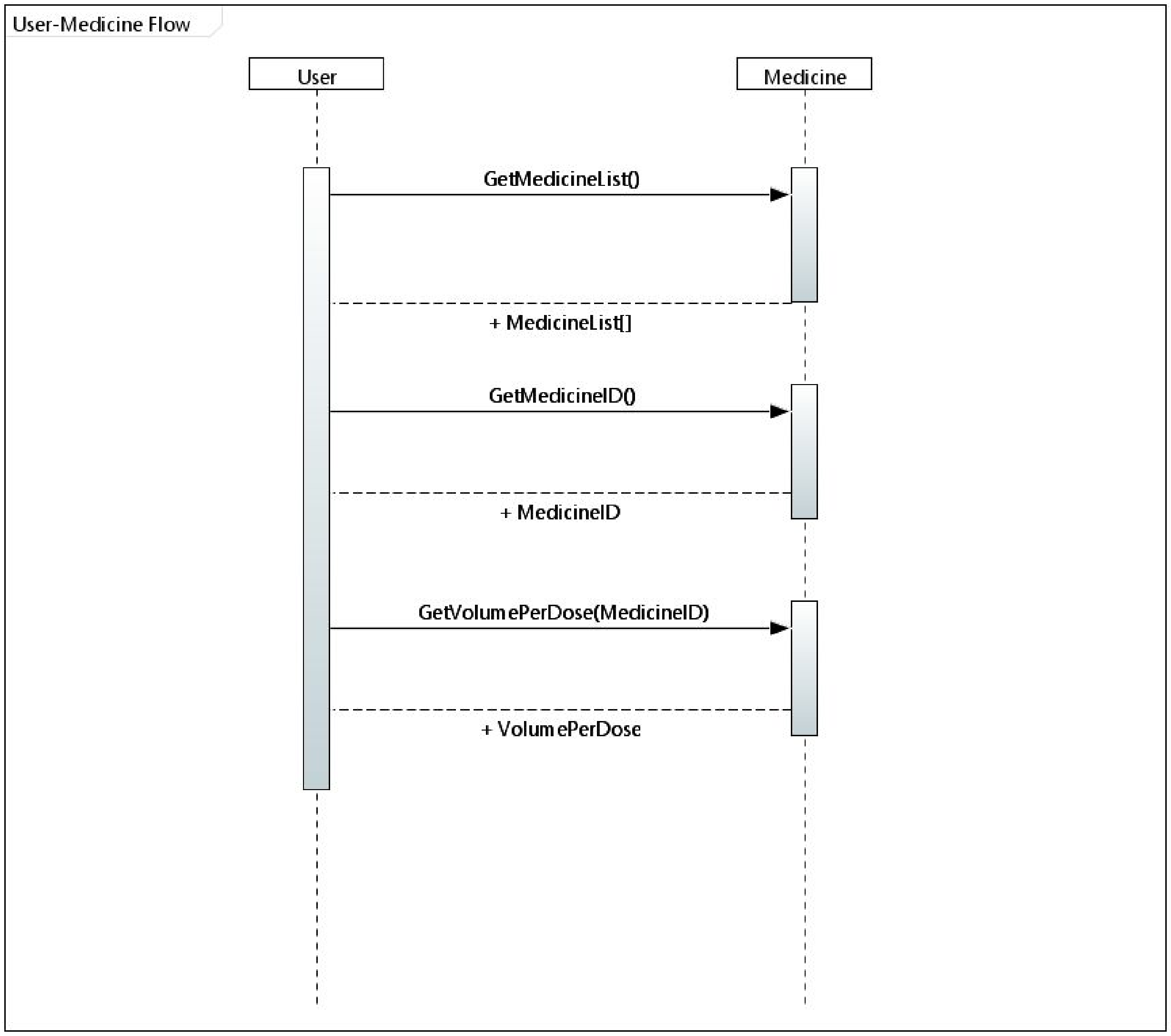


Figure 7: Sequence Diagram of User and Medicine Interaction

A sequence diagram provides a picture of the dynamic relationships that must occur between different objects. In this case, the user is calling upon the medicine class to both get a list of all available medicines as well as get specific information about one medicine in particular.

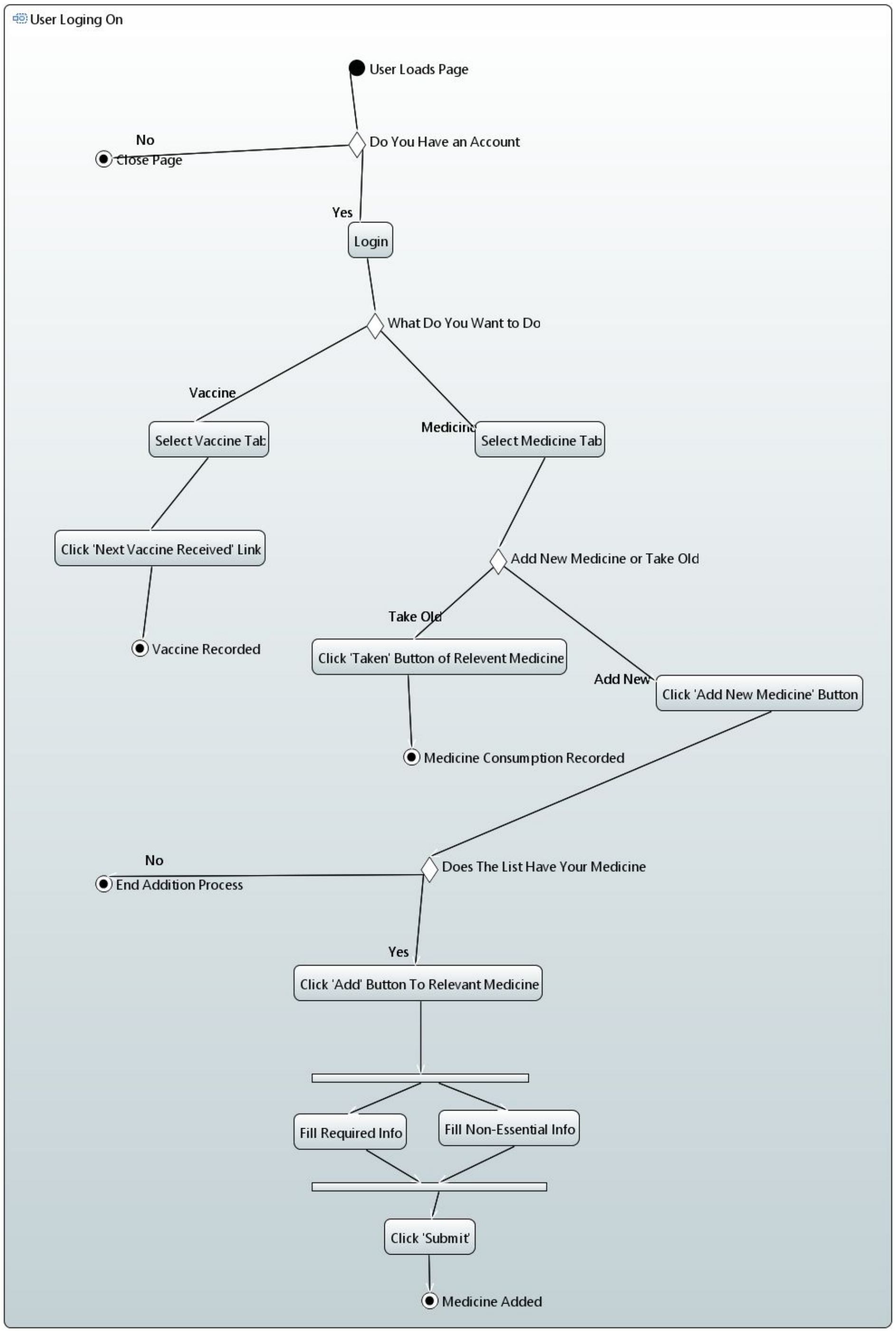


Figure 8: Activity Diagram of a User Logging onto the Site

This diagram depicts the flow of action within the system from one step to another. Activity diagrams are very useful to show the dynamic behaviour of the system from a control flow perspective. In this one specifically, a user looking to interact with the site in any of the possible use cases listed in figure 4. All activities have a definite end.

7.0 Difficulties Encountered

Throughout numerous stages of the medicine and vaccine management application design and implementation processes, the team has, at times, faced challenges. Most importantly, it was challenging to send variables from jsp files to the back end code. In addition, it was difficult to differentiate the 'doPost' method in the controllers for both the 'update' and 'add' methods. This is because these two require two different methods from the DAO's. Moreover, in terms of relating to the databases and the data, one difficulty was in determining the one specific source to extract the information from when required for the medications and vaccinations. With this complication of determining the respective information required, this brought up difficulties once again in the coding aspect as it was challenging to establish dates such as the next vaccine date. In terms of front-end coding, it was very difficult to learn multiple coding languages without many resources supplied to the group from the course as the team had to learn these languages based on their own research. As a result, this limited the extent that group members could do as the only resources supplied are the slides. The team also found the project challenging in the aspect that this group contains seven members. This difficulties regard coordinating with the group as a whole as it is rare all seven members were always available for team meetings at the same time. Lastly, with the various projects assigned to group members throughout their courses, there were hurdles faced in being able to complete the assigned projects for the team's medicine and vaccine management application design within a timely manner.

8.0 Lessons Learned

The group succeeded in numerous ways whether it be frequent team meetings, constructive criticism and effective communication amongst team members, and successful coding strategies. Though, an aspect of the group dynamic that would have been done differently if the team was able to start over would be setting more strict internal deadlines and starting to work earlier. Internal deadlines were set, but with the heavy course load group members experience, it was not always manageable to complete assigned tasks unless members started their sections immediately. If the team enforced strict internal deadlines, which were more closely followed by group members, this would allot more time for the editing stages.
Moreover, in terms of a technical aspect, the team experienced difficulty in locating large amounts of data regarding medications, vaccinations, and their properties. If the group were to restart, they would first determine one data source for the medication information rather than having to do research collating numerous resources. This need of using a lot of links drastically increased the amount of time taken to collect the data necessary for the end product leaving less

time to work on other aspects of the project.

9.0 Wireless Device Accommodation

Over the past 10 years, wireless devices have become increasingly popular. The increased amount of people who have wireless devices, and the increased amount of time people spend on them means that they have a very strong and growing user population. Implementing DrugDrawer on wireless devices would greatly increase the reach of the app to more users.

A change that would need to be made for the wireless devices would be changing the layout of the pages. The current layout on the website would not translate over well to wireless devices, as most wireless devices nowadays have functionalities that computers do not, such as touch screens. Different functionalities such as this would require the system to be more ergonomic for the wireless devices for it to be successful.

A functionality that would make the users experience with the application much easier would be to have the ability to log users in with their fingerprint or facial recognition (if their wireless device has these functionalities). This would reduce the time to log into the app, and could increase the frequency that users check the app.

Another useful functionality to include would be to send all of the updates the system gives out as push notifications on the wireless device. This makes it easy for the user to view when they need to take their medication, and when one of their medications is running low, without having to unlock their phone and log into their account. The app could also send out notifications to the user a week in advance of a vaccine, giving the user ample time to plan for getting it.

Taking DrugDrawer to wireless devices would be challenging and would take a considerable amount of time to properly implement, but if done well, it would be useful in helping expand the reach of the application, as well as improve the overall user experience.

10.0 References

[1]"4 Digital Health Trends (Apps) to Consider for 2018," Liquid State, 19-Jul-2019. [Online]. Available: https://liquid-state.com/digital-health-app-trends-2018/ [Accessed: 05-Oct-2019].

[2]"Editors," Online Medical Care, 03-Oct-2019. [Online]. Available: https://www.onlinemedicalcare.org/15-best-online-medical-apps-that-make-personal-health-easier/ [Accessed: 06-Oct-2019].

[3] J. L., "15 Frightening Stats on Medication Adherence (Plus Infographic)," Pillsy. [Online]. Available: https://www.pillsy.com/articles/medication-adherence-stats [Accessed: 30-Oct-2019].

[4] R. Preidt, "Americans Taking More Prescription Drugs Than Ever," WebMD, 03-Aug-2017. [Online]. Available: https://www.webmd.com/drug-medication/news/20170803/americans-taking-more-prescription-drugs-than-ever-survey [Accessed: 30-Oct-2019].

11.0 Appendix

Appendix A: Project Log sheets

Team Members

Name	Role(s)	Email
Maria Papadimitriou	Project Manager	maria.papadimitriou@mail.utoronto.ca
Chris Overvelde	Coder, System Designer	chris.overvelde@mail.utorotno.ca
Matteo Bruzzese	Coder, System Designer, Researcher	matteo.bruzzese@mail.utoronto.ca
Momo Mitrovich	Writer/Editor, System Visualization, Researcher	momo.mitrovich@mail.utoronto.ca
Nolan Mandel	System Designer, Researcher	nolan.mandel@mail.utoronto.ca
Salma Dessouki	Coder, System Designer	salma.dessouki@mail.utoronto.ca
Dante Mondelli	Writer/Editor, System Visualization, Presenter	dante.mondelli@mail.utoronto.ca

Section A: Project Administration Tasks

Task	Task	Assigned to	Milestone	Due Date	Date Achieved
1.0	Select topic and assign roles	All	Topic and roles	Sept 15	Sept 15
2.0	Project Plan	All	Tasks identified and assigned	Oct 3	Oct 3
2.1	Project background	Salma	Project background paragraph written	Oct 7	Oct 5
2.2	Application description - 3 dynamic pages	Matteo	Application description paragraph written	Oct 7	Oct 5
2.3	Establish website functionalities	Maria, Nolan	Four Functionalities identified and clearly stated in project plan	Oct 7	Oct 6

2.4	Application description - Relational Database	Maria, Nolan	Relational database identified and clearly stated in project plan	Oct 7	Oct 6
2.5	Web content - about us, references, links	All	About us, references, and external links to be provided to indicate what research the web app is based on	Oct 7	Oct 6
2.6	Problem statement	Salma, Maria	Problem Statement is clearly stated	Oct 7	Oct 5
2.7	Project Proposal Submission	Maria	Project Proposal Submitted as required	Oct 7	Oct 7
2.8	Internal Deadlines	Maria	Internal Deadlines for when we would like categories above completed	Oct 10	Oct 10
3.0	Prepare written progress report	All	Project log updated	Nov 2	Nov 4
3.1	Context and motivation for the project	Salma, Dante	Paragraph written	Nov 2	Nov 3
3.2	System Requirements	Chris	Requirements established and documented	Nov 2	Oct 30
3.3	Software development methodologies	Maria	Software development methodologies established and implemented	Nov 2	Oct 15
3.4	Three use cases	Nolan	Use cases completed and made into diagram	Nov 2	Nov 1
3.5	Context diagram	Nolan	DFD completed	Nov 2	Nov 1
3.6	Level 0 data flow diagram	Nolan	DFD completed	Nov 2	Nov 1

	(3)				
3.7	Level 1 data flow diagram (1)	Nolan	DFD completed	Nov 2	Nov 1
3.8	System Architecture	Matteo, Maria	System architecture established	Nov 2	Oct 20
3.9	Project Log and discussion of the status of implementatio n	Maria	Documented	Nov 2	Nov 1
3.10	Challenges	Momo	Documented	Nov 2	Nov 3
3.11	Submission	Maria	Report submitted	Nov 4	Nov 4
4.0	Prepare Final Report	All	Project log updated	Dec 3	Dec 4
4.1	Context and Motivation	Dante	Paragraph written	Dec 4	Dec 3
4.2	Requirements Summary and Use Cases	Chris	Paragraph written	Dec 4	Dec 3
4.3	Class Diagram and UML	Nolan	Paragraph written	Dec 4	Dec 3
4.4	Difficulties encountered	Momo	Paragraph written	Dec 4	Dec 3
4.5	Lessons learned	Momo	Paragraph written	Dec 4	Dec 3
4.6	Wireless Device Modifications	Chris	Paragraph written	Dec 4	Dec 3
4.7	References	Dante	References completed	Dec 4	Dec 3
4.8	Appendices	Momo	Appendices completed	Dec 4	Dec 3

4.9	Final Report Submission	Maria	Report submitted	Dec 4	Dec 3
4.10	Web App Submission	Maria	Website submitted	Dec 4	Dec 3

Section B: Web Application Development Task List

Task #	Task Assigned to	Milestone Due Date	Date Achieved
7.0 Build database	Maria, Chris	Nov 1	Oct 28
8.0 Build Server Prototype	Salma	Nov 1	Nov 2
9.0 Back End Coding	Matteo	Dec 4	Dec 3
10.0 Front End Coding	Maria, Salma	Dec 4	Dec 3

Section C: Progress Summary

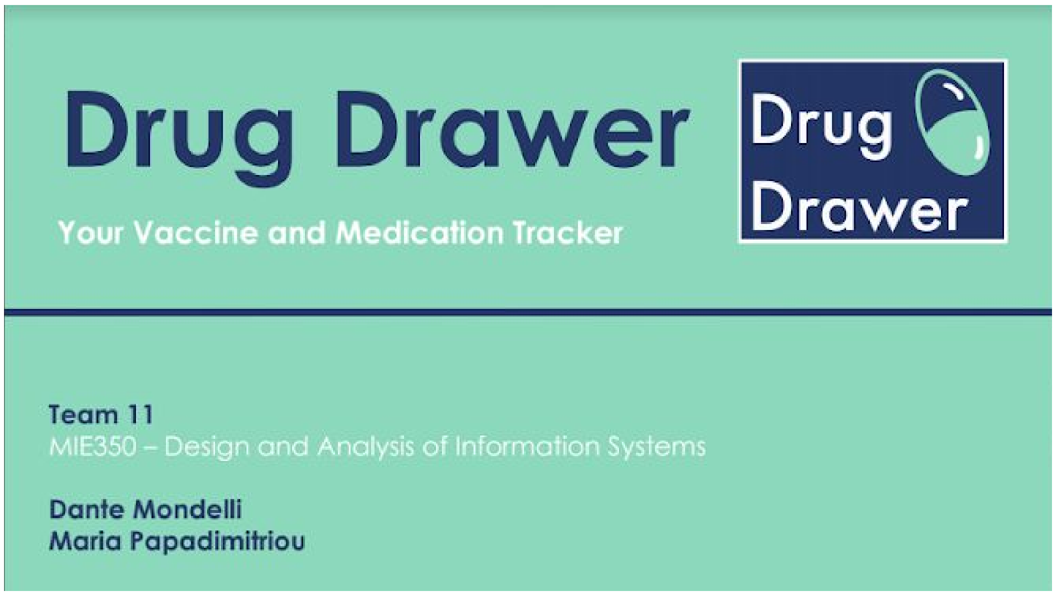
Task	Date	Status	Comments
1.0	Sept 15	Date Achieved	Topic is a personal health application for doctors and patients.
2.0	Oct 3	Completed	Project Plan clearly identifies all dynamic pages, functionalities, relational database, and web content.
3.0	Nov 2	Completed	Progress report completed
4.0	Dec 4	Completed	Final Report completed
7.0	Nov 1	Completed	Database filled with relations and attributes required
8.0	Nov 2	Completed	Server prototype completed and ready for implementation

9.0	Dec 3	Completed	Back end coding complete and website is fully functional
10.0	Dec 3	Completed	Front end coding complete and satisfies desired heuristics

Appendix B: Electronic Copy of Website

The web application is submitted through Quercus.

Appendix C: Final class presentation powerpoint



Drug Drawer

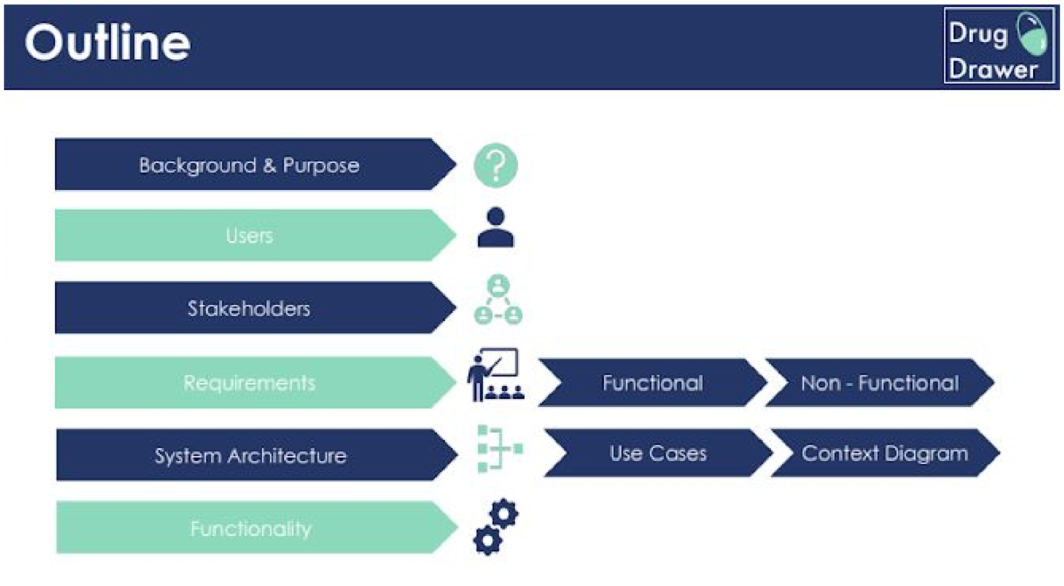
Your Vaccine and Medication Tracker

Drug Drawer

Team 11

MIE350 – Design and Analysis of Information Systems

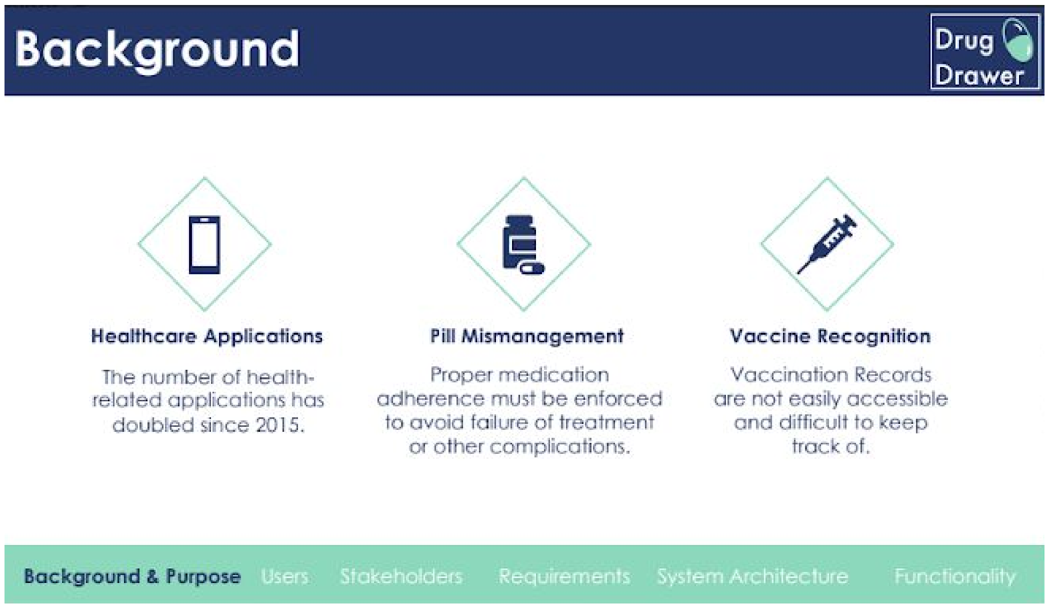
Dante Mondelli
Maria Papadimitriou



Outline

Drug Drawer

- Background & Purpose
- Users
- Stakeholders
- Requirements — Functional — Non - Functional
- System Architecture — Use Cases — Context Diagram
- Functionality



Background

Drug Drawer

Healthcare Applications

The number of health-related applications has doubled since 2015.

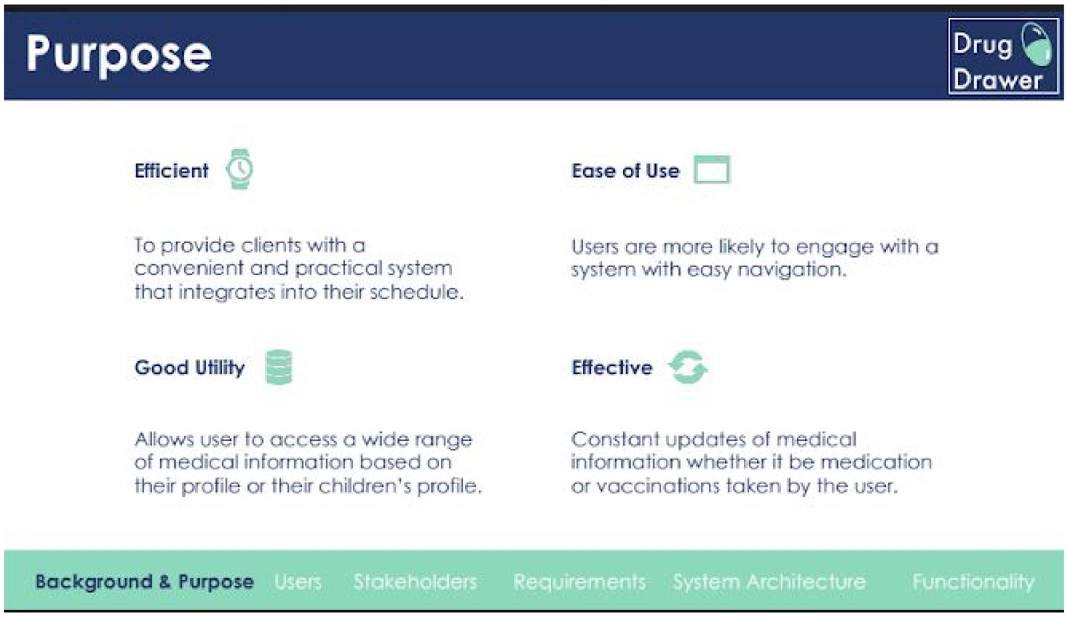
Pill Mismanagement

Proper medication adherence must be enforced to avoid failure of treatment or other complications.

Vaccine Recognition

Vaccination Records are not easily accessible and difficult to keep track of.

Background & Purpose Users Stakeholders Requirements System Architecture Functionality



Purpose

Drug Drawer

Efficient

To provide clients with a convenient and practical system that integrates into their schedule.

Ease of Use

Users are more likely to engage with a system with easy navigation.

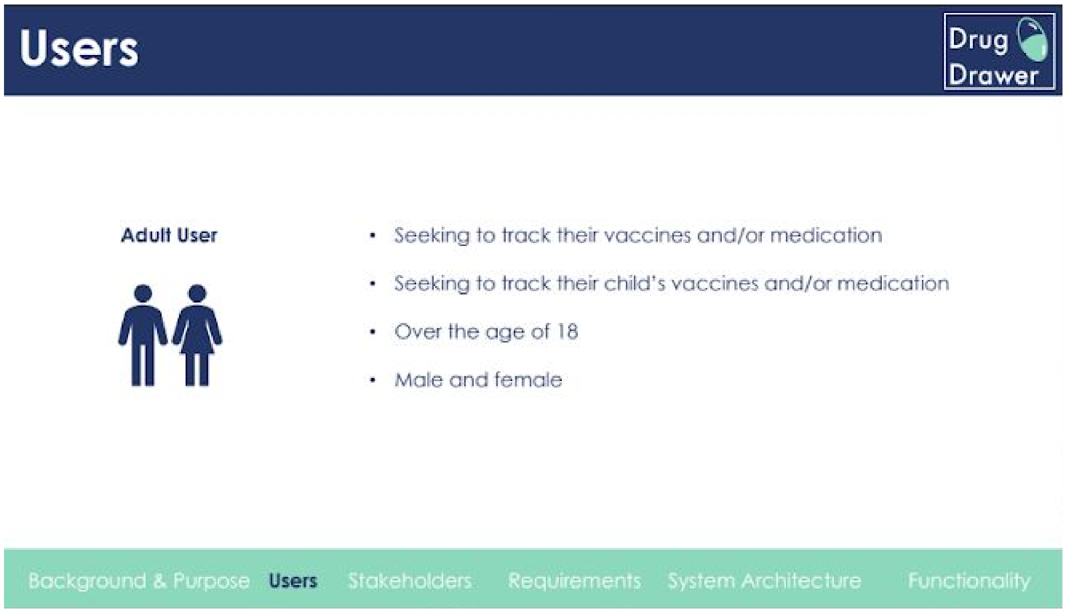
Good Utility

Allows user to access a wide range of medical information based on their profile or their children's profile.

Effective

Constant updates of medical information whether it be medication or vaccinations taken by the user.

Background & Purpose Users Stakeholders Requirements System Architecture Functionality



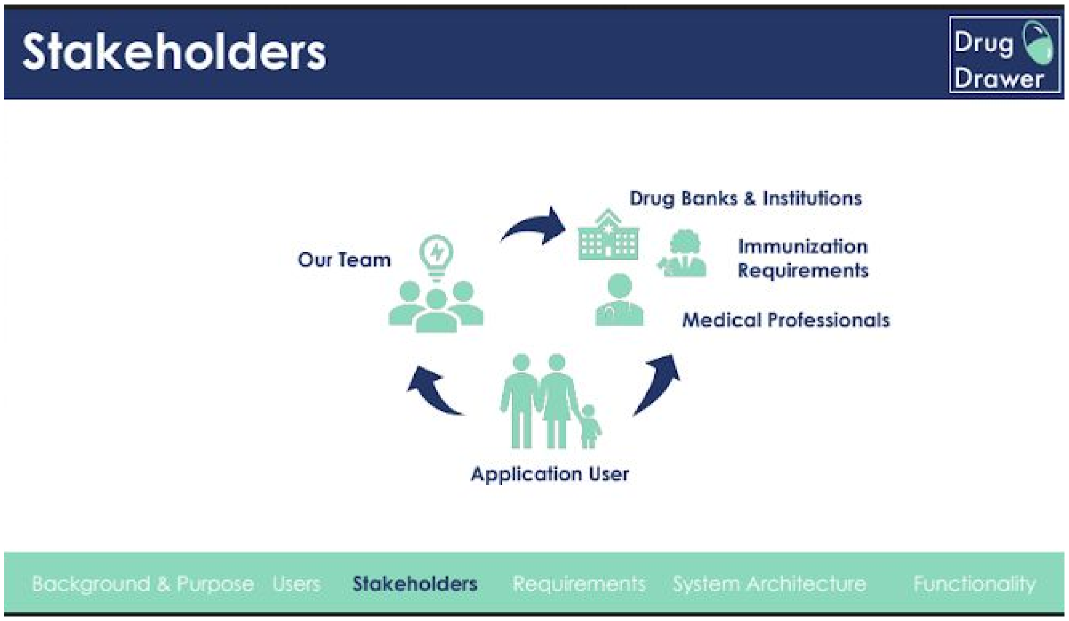
Users

Drug Drawer

Adult User

- Seeking to track their vaccines and/or medication
- Seeking to track their child's vaccines and/or medication
- Over the age of 18
- Male and female

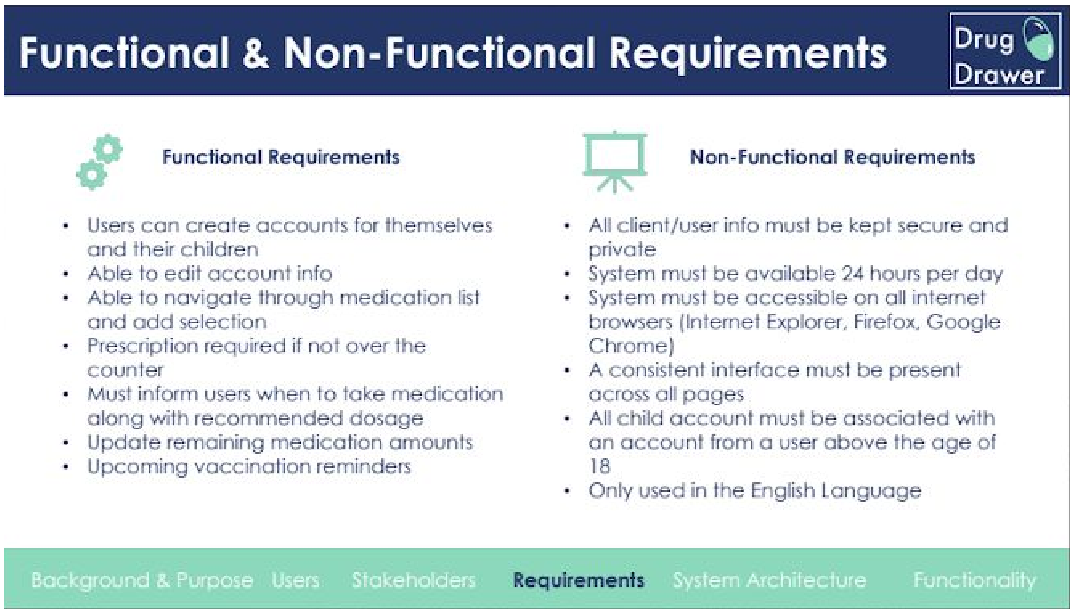
Background & Purpose **Users** Stakeholders Requirements System Architecture Functionality



Stakeholders

Drug Drawer

Background & Purpose Users **Stakeholders** Requirements System Architecture Functionality



Functional & Non-Functional Requirements

Drug Drawer

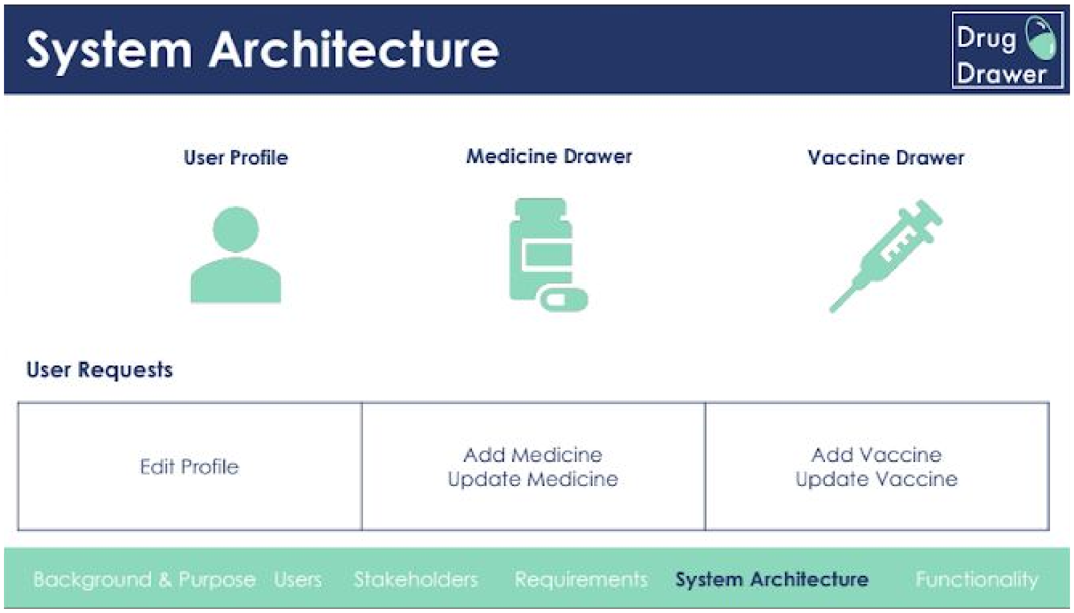
Functional Requirements

- Users can create accounts for themselves and their children
- Able to edit account info
- Able to navigate through medication list and add selection
- Prescription required if not over the counter
- Must inform users when to take medication along with recommended dosage
- Update remaining medication amounts
- Upcoming vaccination reminders

Non-Functional Requirements

- All client/user info must be kept secure and private
- System must be available 24 hours per day
- System must be accessible on all internet browsers (Internet Explorer, Firefox, Google Chrome)
- A consistent interface must be present across all pages
- All child account must be associated with an account from a user above the age of 18
- Only used in the English Language

Background & Purpose Users Stakeholders **Requirements** System Architecture Functionality



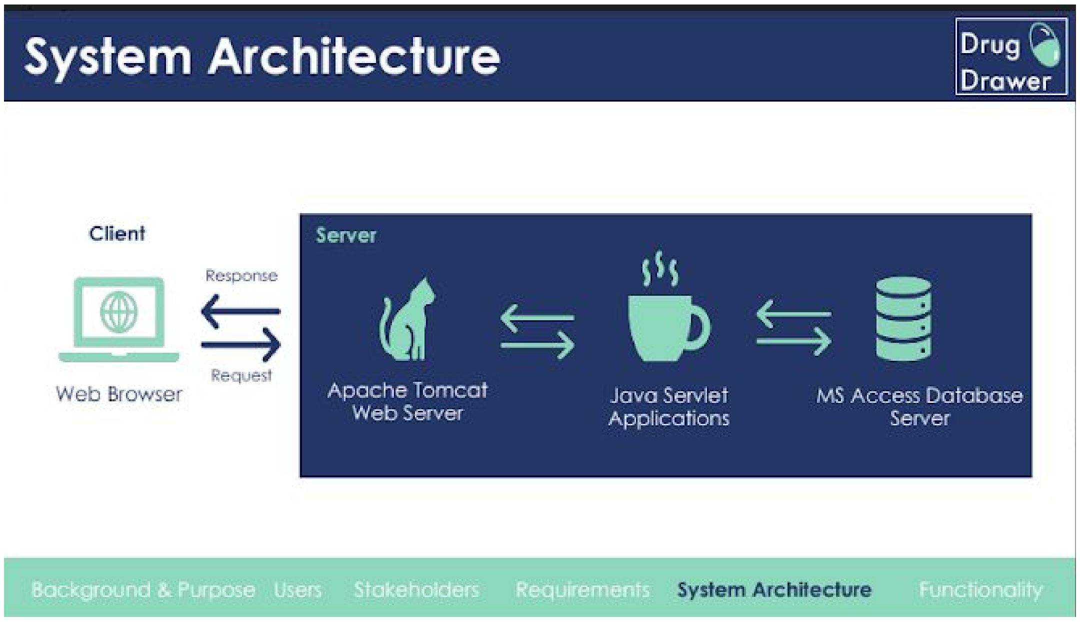
System Architecture

Drug Drawer
User Profile
Medicine Drawer
Vaccine Drawer

User Requests

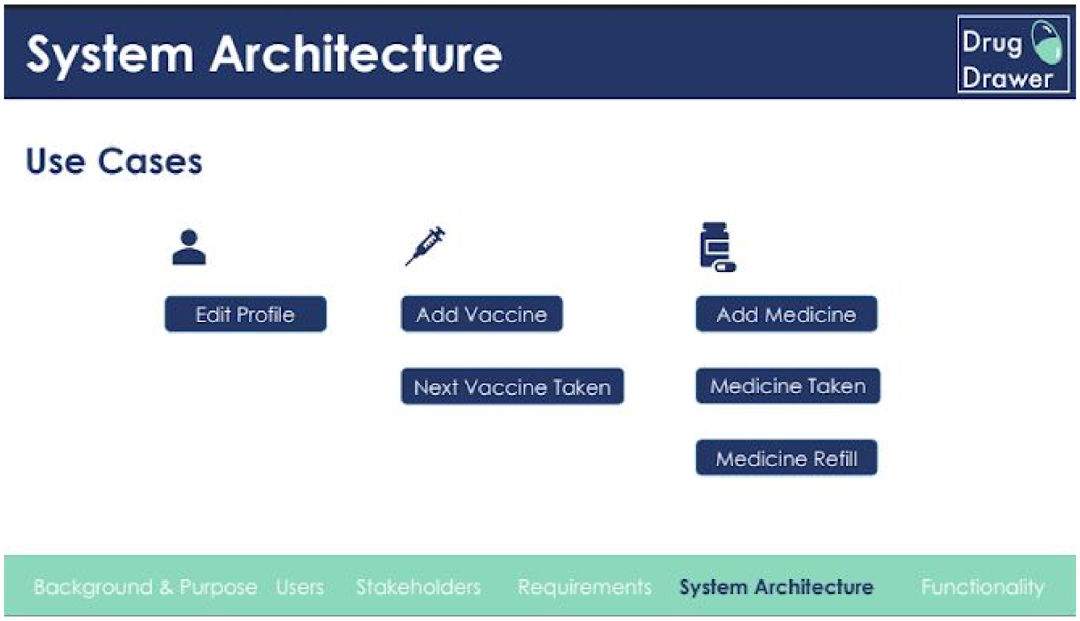
Edit Profile	Add Medicine Update Medicine	Add Vaccine Update Vaccine

Background & Purpose Users Stakeholders Requirements **System Architecture** Functionality



System Architecture

Background & Purpose Users Stakeholders Requirements **System Architecture** Functionality



System Architecture

Drug Drawer

Use Cases

Edit Profile

Add Vaccine

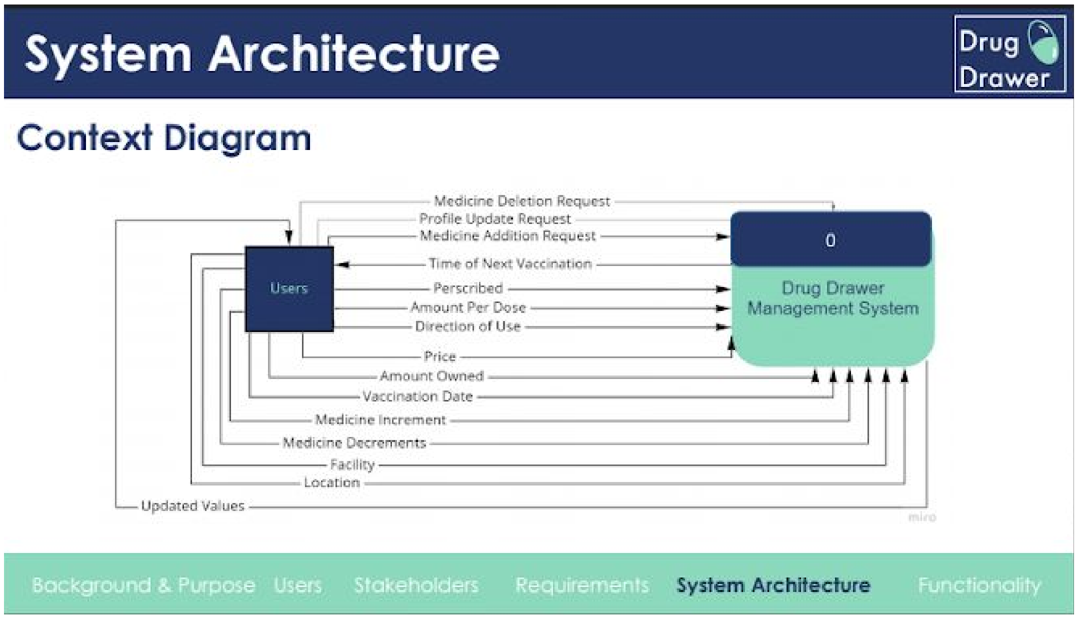
Next Vaccine Taken

Add Medicine

Medicine Taken

Medicine Refill

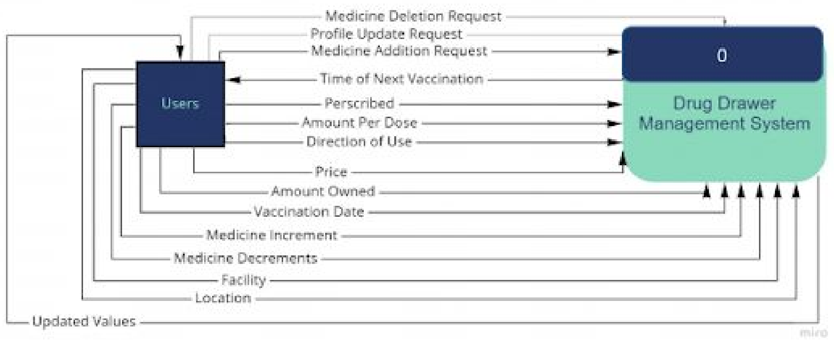
Background & Purpose Users Stakeholders Requirements **System Architecture** Functionality



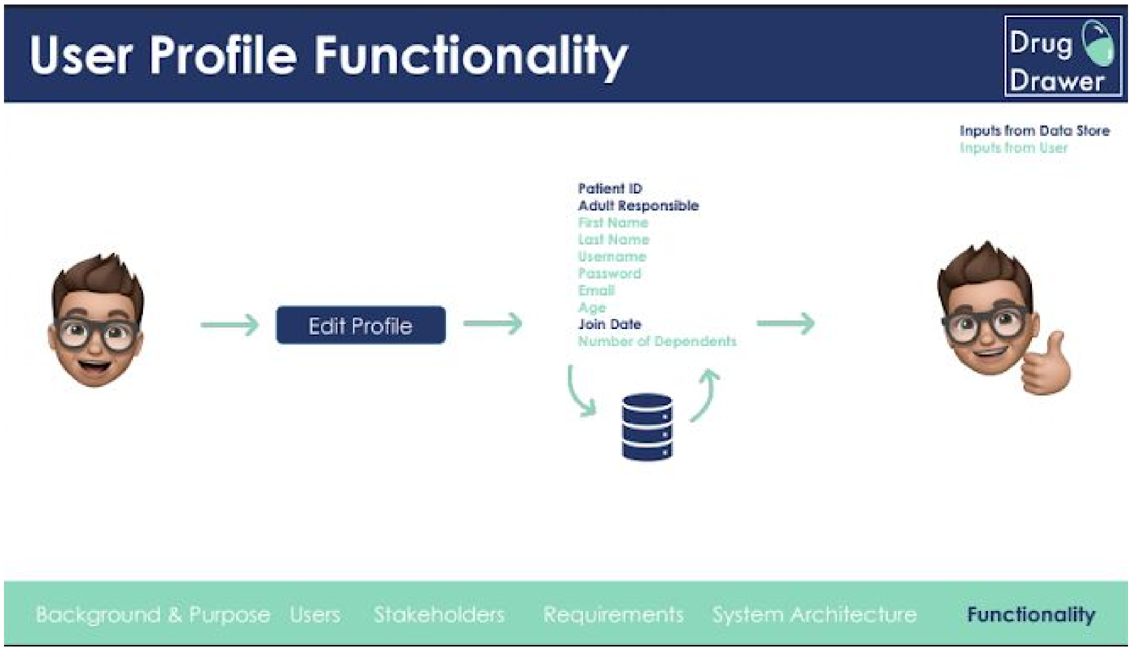
System Architecture

Drug Drawer

Context Diagram



Background & Purpose Users Stakeholders Requirements **System Architecture** Functionality



User Profile Functionality

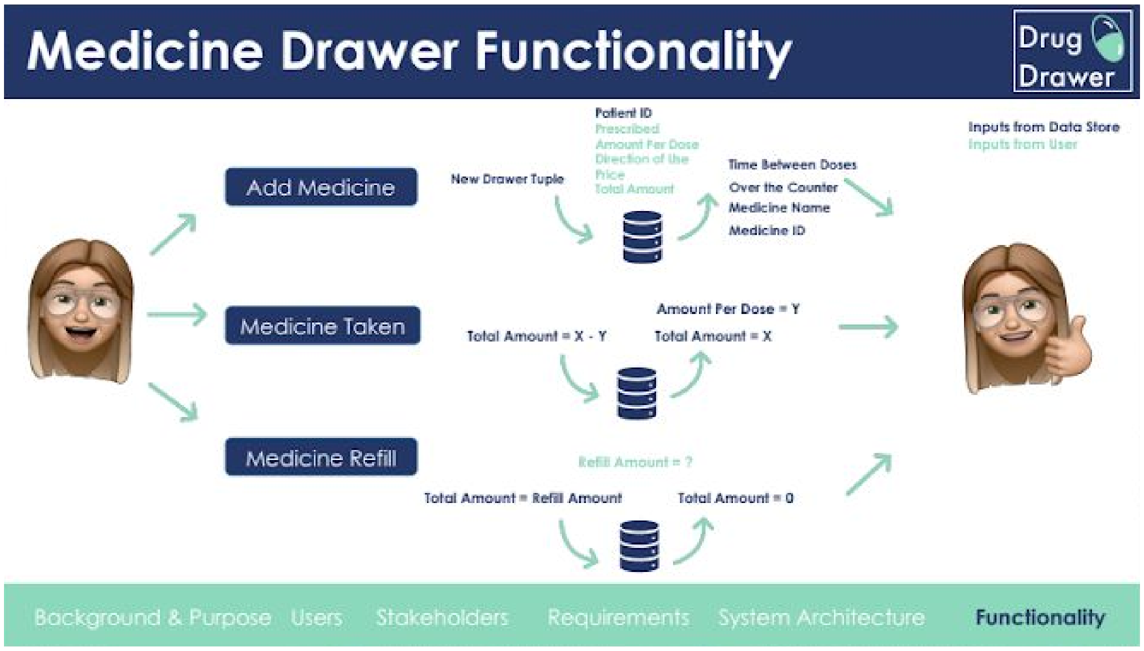
Drug Drawer

Inputs from Data Store
Inputs from User

Edit Profile

Patient ID
Adult Responsible
First Name
Last Name
Username
Password
Email
Age
Join Date
Number of Dependents

Background & Purpose Users Stakeholders Requirements System Architecture **Functionality**



Medicine Drawer Functionality

Drug Drawer

Inputs from Data Store
Inputs from User

Add Medicine

New Drawer Tuple

Patient ID
Prescribed
Amount Per Dose
Direction of Use
Price
Total Amount

Time Between Doses
Over the Counter
Medicine Name
Medicine ID

Medicine Taken

Total Amount = X - Y

Amount Per Dose = Y
Total Amount = X

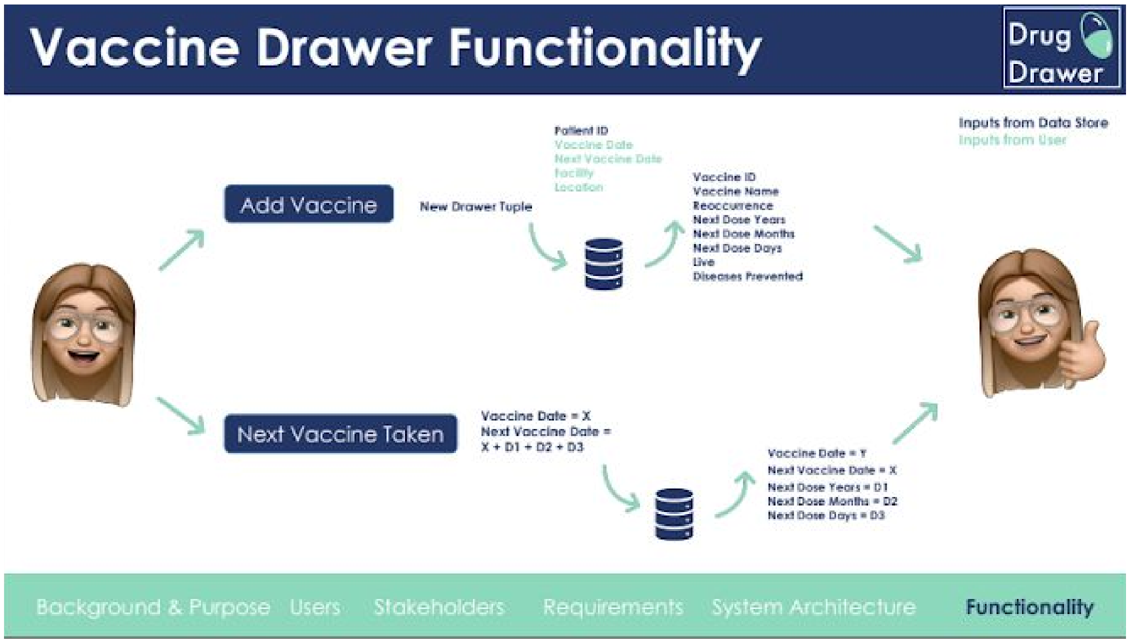
Medicine Refill

Refill Amount = ?

Total Amount = Refill Amount

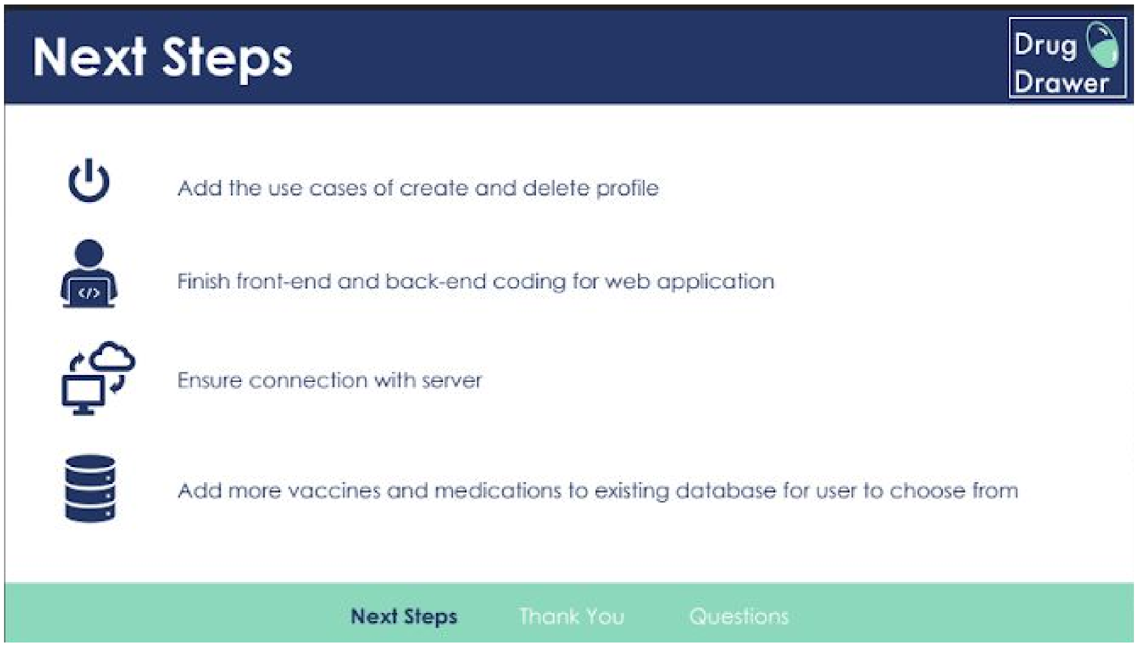
Total Amount = 0

Background & Purpose Users Stakeholders Requirements System Architecture **Functionality**



Vaccine Drawer Functionality

Background & Purpose Users Stakeholders Requirements System Architecture **Functionality**



Next Steps

Drug Drawer

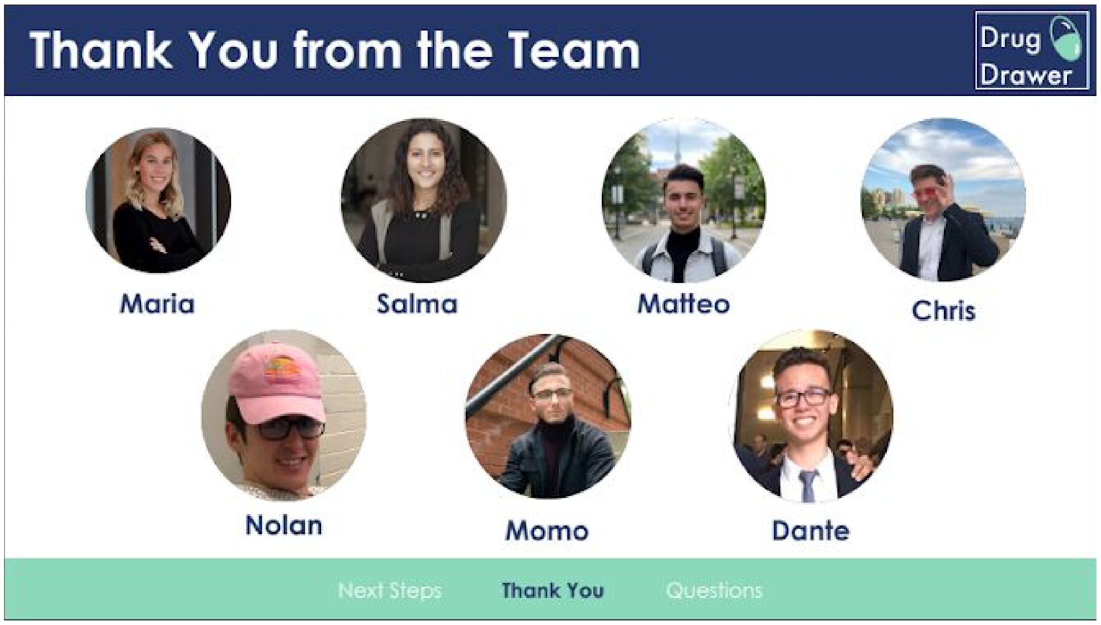
Add the use cases of create and delete profile

Finish front-end and back-end coding for web application

Ensure connection with server

Add more vaccines and medications to existing database for user to choose from

Next Steps Thank You Questions



Thank You from the Team

Drug Drawer

Maria

Salma

Matteo

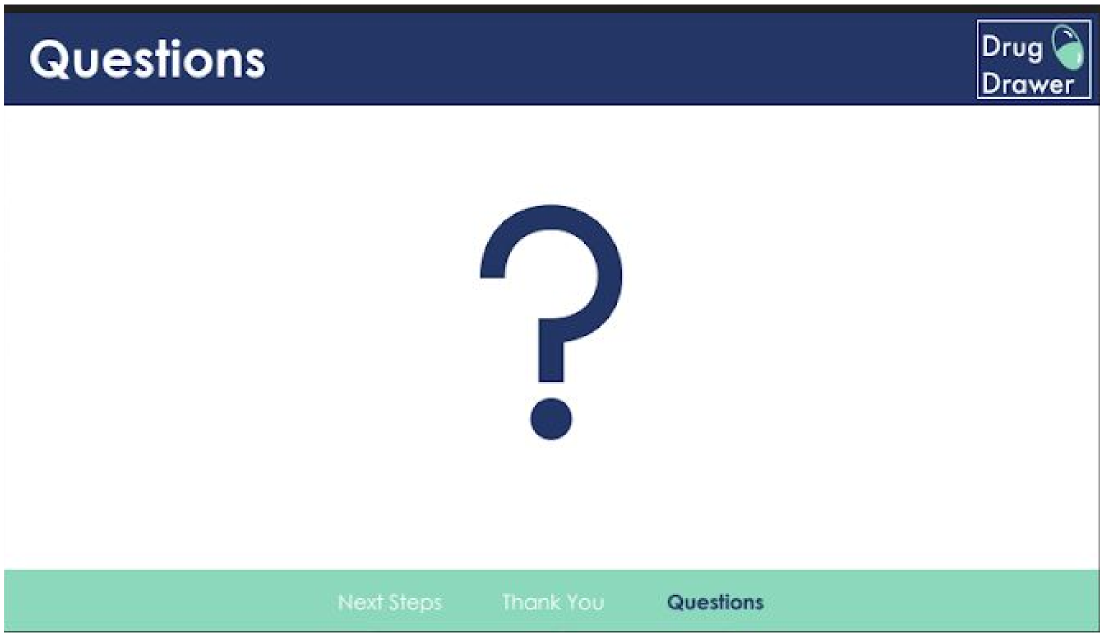
Chris

Nolan

Momo

Dante

Next Steps **Thank You** Questions



Questions

Drug Drawer

?

Next Steps Thank You **Questions**

Appendix D: Roles and responsibilities of team members

Team Roles and Responsibilities

Writer/Editor: Momo, Dante, Chris

- Make the english pretty
- Work back and forth with tutorial TAs to establish proper report structure
- Ensure all components of documents are included
- Brainstorming

System Visualization: Maria, Momo, Salma, Dante

- HF design of website
- Visual Appearance and Cohesiveness
- The ‘texture’ and broader ‘unity’

System Designer: Chris, Nolan, Salma

- Front end stuff
- The actual architecture of data
- Visualization of all the components and how they are related
- Define functionality / steps for a user to execute tasks

Researcher: Nolan, Momo, Matteo

- Info required for project
- Validate statements with facts (show why we made decisions we did)

Coder (back end): Everyone

- Functionality coding
- Pseudo-Code Prep
- Setting up classes and Abstract Classes
- Set up efficient practices for coding (searching online / starter code from prof, etc.)

Presenter: Dante, Maria

- Powerpoint
- Preparing script
- Practice

General Manager: Maria

- Organize to do lists
- Keep track of action items and current status
- Setup meetings (agenda + schedule or doodle)
- Submit assignments
- Talk to the professor about any questions / attend office hours