Massachusetts Association of Sober Housing Mobile Application

Project Team:

Christopher Healy: cjhealy@wpi.edu

Kaitlin McDermott: kmmcdermott@wpi.edu

Project Advisor

Professor Wilson Wong

Department of Computer Science

Project Co-Advisor

Professor Michael Engling

Department of Computer Science

Abstract

Sober homes are a significant step in the recovery process from drug and alcohol addiction. They are private residences free of drugs and alcohol where recovering addicts can go to live during their rehabilitation process. The Massachusetts Alliance of Sober Housing certifies sober homes that adhere to a set of rules and guidelines they created. The goal of this project is to create a mobile application for the Massachusetts Alliance of Sober Housing that will collect data from sober home residents. The data collected from the application will provide insight into the effectiveness of sober homes.

Table of Contents

Table of Figures

1. Introduction

Every year the Substance Abuse and Mental Health Services Administration (SAMHSA) conducts a National Survey on Drug Use and Health (NSDUH). According to SAMHSA substance use disorder (SUD) is "characterized by impairment caused by the recurrent use of alcohol or other drugs (or both), including health problems, disability, and failure to meet major responsibilities at work, school, or home" (SAMHSA, 2019a, 32). In 2018 the NSDUH found that 19,343,000 Americans aged 18 or older (7.8% of the population) suffered from SUD (SAMHSA, 2019b, Table 5.1A-5.1B). There is an increase from 2017 when the NSDUH reported that 18,708,000 Americans aged 18 or older (7.6% of the population) suffered from SUD (SAMHSA, 2019b, Table 5.1A-5.1B). The NSDUH also reports the number of Americans aged 18 or older who had received treatment in the past year. In 2018 that number was 1,014,000 (0.4% of the population) and in 2017 the number was 1,116,000 (0.5% of the population) (SAMHSA, 2019, Table 5.9A-5.9B). Showing that from 2017 to 2018, the number of Americans suffering from SUD increased while the number of those people receiving treatment decreased.

One potential reason for the decrease in those with SUD seeking treatment is a lack of confidence in the treatment programs. The Society of Community Research and Action (SCRA) reported that "more than half of individuals treated [for SUD] ... will resume [alcohol and other drugs] use following their discharge from treatment—most often in the first 90 days following discharge" (SCRA, 2013, para. 5). SCRA continues that the reason for these relapses is because there is no "post-treatment monitoring and support" and suggest the use of recovery residences, like sober homes, to fill this gap in treatment. This recommendation is made based on some early evidence showing recovery residences to have a positive impact on the recovery process. However, SCRA does acknowledge that there is not yet enough data on the full effects that recovery residence participation has because of the lack of funding for further research.

The Massachusetts Alliance of Sober Housing (MASH) wishes to collect data on the residents of certified sober homes for three reasons. First, the data will help MASH track the progress of residents and give them information on how to improve the sober home experience. Furthermore, MASH will be able to use the data to advocate for sober homes, which are currently under attack in many communities. The data will show that sober homes have no negative impact on the surrounding communities. Finally, MASH hopes that obtaining data will enable them to prove the usefulness of sober homes during the recovery process and in turn, receive some additional funding and support.

However, MASH has been unable to collect a sufficient amount of data to help them achieve these goals. Due to the high monetary and time cost of physical data collection, MASH cannot collect data. MASH does not have the staff or funding to send people to every sober home regularly to track the progress of the residents. An alternative to physical data collection is to use a mobile application to collect the data. A mobile app provides the benefit of having the residents report data on themselves rather than sending someone to go and get it from them. The only downside is that residents may feel less motivated to input this data without the presence of a person asking them for the data. The application must be easy to use and include some way of rewarding the user for inputting data.

The goal of this project is to create a mobile application that provides MASH with useful data on the residents of sober homes. To effectively and accurately collect this data, the goal has three primary objectives. The first objective is to work with MASH directors and sober homeowners to determine which metrics will be the most useful to collect. The second objective is to create a data collection tool that runs on a smartphone or tablet. The third objective is to implement a social networking feature that rewards users by allowing them to share their successes

with their peers. Once these objectives are complete additional features will be added, as time permits, based on feedback from received from the users.

2. Research

The project is a continuation of a Major Qualifying project completed during the 2018-2019 academic year. Our team continually used last year's report and project to successfully deliver our project (Hard et al., 2019). Last year's team conducted most of the preliminary research about different approaches for the project. We decided to conduct separate research; however, we took into account last year's team decisions to help formulate how we could move forward. The previous team could not contact our sponsors during the early stages of the project, so they failed to narrow the sponsors' requirements at the beginning of the project. Fortunately, our team was able to decide the requirements with our sponsors early so we can move forward with a stronger focus.

The team focused research on the libraries to use in our application. We looked into timeline libraries and how to implement it. We also researched creating a survey creation tool for the directors of MASH to allow more functionality in our application.

3. Methodology

The goal of this project is to create a mobile application to collect data on sober home residents and their recovery process. This chapter discusses the software development methodology that we chose to implement for the creation of the application. In deciding which software development methodology is best we compared two of the most popular methodologies in the industry; the waterfall model and agile scrum methodology. This chapter also covers our methodology for gathering information pertaining to our application, both before and during development.

3.1 Software Development Approach

3.1.1 The Waterfall Model

The waterfall model is a linear approach to software development consisting of six distinct stages: requirements analysis, system design, implementation, testing, deployment, and maintenance (Tutorials Point). These stages are completed independent of the others and each stage must be fully completed before the software team moves on to the next stage (Tutorials Point).

The requirements analysis phase consists of gathering all of the features required for the completed application. This is done by interviewing the client, surveying the target user base, and researching similar applications. Once a feature list is generated the team must ensure that each of the features is well defined and testable. For a feature to be well defined it must be stated in a way that everyone on the team has the same understanding about what the feature will do, but not necessarily how it will function. For a feature to be testable there must be a way to show that under any potential case the feature does exactly what it is expected to do.

In the system design phase, a software design is created for the application (Tutorials Point). This design is broken into high-level design and low-level design. The high-level design gives a description of how each of the features, that were previously generated, will interact with each other inside the application, as well as any hardware or software requirements. The low-level design describes how each of the features will function, what inputs they need to function, and what they will do as a result.

During the implementation phase, each of the individual features are written and tested (Tutorials Point). This process is usually broken up over many teams, each working somewhat independently. Then in the testing phase, the teams come together and integrate the features together to create the final working application. This application then gets extensively tested to

ensure that it meets all of the requirements. The team also tries to find and eliminate as many potential bugs in the code as they can.

Once the application has been created and tested it is ready for deployment where the application is made available to the consumer. The team must create a user manual to release alongside the application that teaches the end-user how to properly set up and use the application. Additionally, deployment can be segmented so that the application is initially released to only a small part of the population who may find additional bugs the development team did not consider (Tutorials Point).

The final step in the waterfall model is maintenance where a subset of the development team or a separate team continues to address issues that arise with the application (Tutorials Point). This phase is important because it helps to ensure that the application continues to function as intended after it is distributed. The length of the maintenance phase is determined by the company or group that created the application.

The waterfall model's advantages come from the fact that each stage is well defined and discrete. This makes the model easy to implement and follow and creating specific deliverables for each phase ensures that all aspects of the project will get done. The disadvantages of the waterfall model are due to the rigidity of the structure. Because the model does not allow for returning to a phase once it is complete adding or modifying requirements later in the project is nearly impossible. If something goes wrong in a phase, the team must back track and redo all previous work. The cost of going back to the previous phase is high and requires all of the models and documentation to be redone before moving forward. Additionally, because the application as a whole is not tested until the end of the project small bugs are more easily able to propagate through and become a much bigger issue.

3.1.2 The Agile Methodology

Agile Methodology is an approach to software development centered around the idea of iterative development. The Agile development cycle involves the following stages: requirements gathering, planning, product design, development, release, and tracking and monitoring ("What's the Difference", 2018). However, since Agile methodology is flexible, these stages do not necessarily need to happen in sequential order. Multiple stages can occur simultaneously, or in parallel with one another. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals (Cprime).

Agile methodology allows for more responsive development requests and high-level features are developed and delivered more quickly with short cycles compared to the long cycles of the waterfall method (Cprime). The short cycles are favorable to the customer due to the faster delivery of wanted features. The focus is on high-level features which helps reduce time-to-market because of efficiency as well as decreased overhead. The development team will see that their efforts are valued and used since agile reduces unproductive work through scrums. Requirements are prioritized to maximize the value of the team's work. Each short cycle, referred to as a sprint, focuses on the highest prioritized features to ensure maximum value to the customer.

Scrum Framework

Scrum is a framework for organizing and managing work that is based on a set of values, principles, and practices that provide the foundation for implementation (Rubin, 2013). Scrum is a people-centric framework based on the values of honesty, openness, courage, respect, focus, trust, empowerment, and collaboration (Rubin, 2013). Each of these values adds to the quality of

the project. Scrum practices are embodied in specific roles, activities, artifacts, and their associated rules (Rubin, 2013).

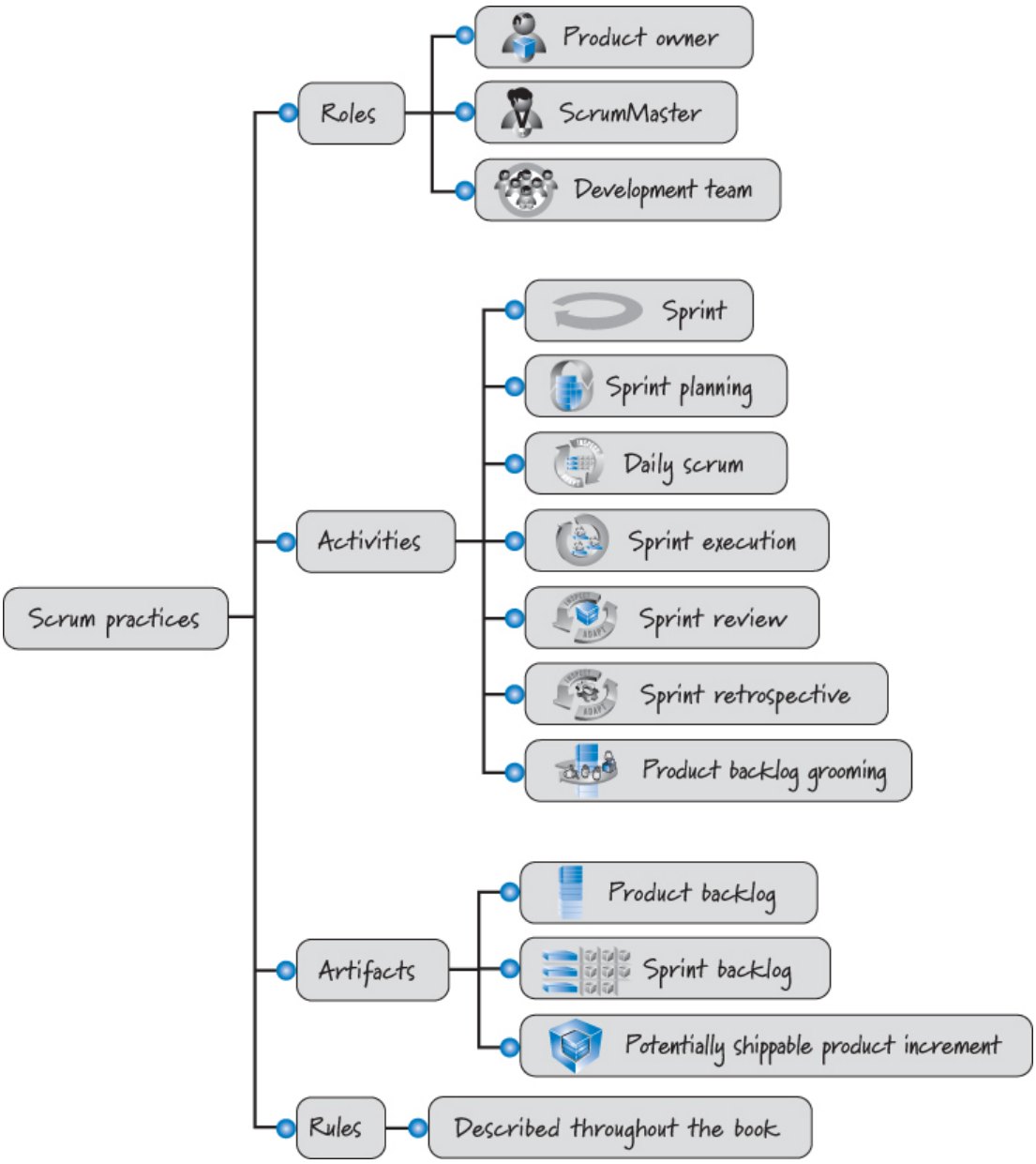

Figure 1: Scrum Framework (Rubin, 2013)

Scrum Roles

Scrum development consists of three Scrum roles: product owner, ScrumMaster, and the development team. There can be other roles when using Scrum, however, only those three roles are required. The product owner is responsible for what will be developed and in what order. The

product owner is the central point of product leadership and he or she is responsible for deciding what features and functionality to build and the order to build them. This role is in charge of maintaining communication with the team about the vision the team is aiming to achieve. The product owner takes responsibility for the overall success of the product development or maintenance. No matter the focus of the product, the product owner has the obligation to make sure that the most valuable work is always performed. The product owner actively collaborates with the ScrumMaster and development team to ensure the team builds what the product owner wants quickly.

The ScrumMaster is responsible for guiding the team in creating and following its own process based on the broader Scrum framework. He or she acts as a coach, providing leadership and helping the Scrum team to develop their own high-performance, organization-specific Scrum approach. The ScrumMaster is in charge of leading the daily scrums, scrum planning, and scrum retrospective meetings and helps organize the team through issues and makes improvements to its use of Scrum. The ScrumMaster is responsible for protecting the team from outside interference and takes a leadership role in removing obstacles that inhibit team productivity. The ScrumMaster functions as a leader, not a manager, and does not have the authority to exert control over the team.

The development team is responsible for determining how to deliver what the product owner has asked for. Scrum defines the role of a development team which is a diverse, cross-functional collection of these types of people who are responsible for designing, building, and testing the desired product. The development team self-organizes to determine the best way to accomplish the goal of the product owner.

Product Backlog

When using Scrum, the team does the most valuable work first. The product owner is responsible for determining and managing the sequence of this work and communicating it in the

form of a prioritized list known as the product backlog. The product backlog items are initially featured to meet the product owner's vision, however, the product backlog can also contain new features, changes to existing features, repairs, technical improvements, and so on.

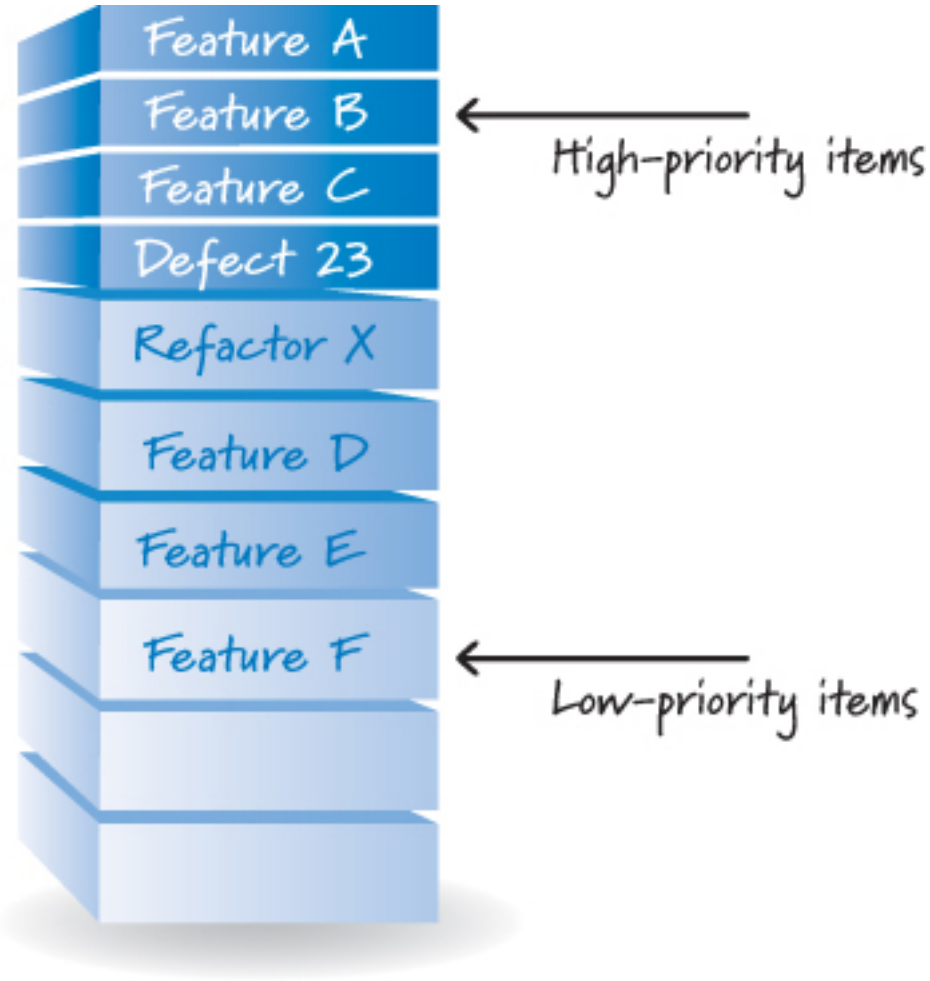

Figure 2: Product Backlog (Rubin, 2013)

The product owner ensures that the backlog is prioritized so that the high-level items appear at the top of the product backlog and the lower-value items appear toward the bottom. The product backlog is constantly evolving through each sprint the Scrum team completes. Before the team prioritizes and orders the items in the product backlog, each item needs to be sized. The product owner needs to know the item's size because size equates to cost. Once the product owner has the cost of each item, he or she can determine the priority of the item. Many teams will use a relative

size measure such as story points. A relative size measure expresses the overall size of an item compared to the other items on the backlog.

Sprints

Scrum teams perform work in iterations or cycles called sprints. The work completed by the team should be something of tangible value to the customer or user and should implement items from the product backlog. Each new sprint will implement the highest prioritized items in the backlog. Sprints always have a fixed start and end date, and they should all be the same duration. A new sprint starts immediately after the previous sprint finishes.

3.1.3 Think-Alouds

Think-alouds are a technique used in human-computer interaction that gives developers insight into how a user uses their application so that it can be modified to best fit their usage (Nielsen). A think-aloud can be conducted with a prototype, either on paper or digital, or with a working version of the application. The key component of a think-aloud is that the user verbalizes their thought process while they try to complete a task (Nielsen). During this process the observer should not give the user any hints about how to complete the task unless the user is completely stuck. While the user is completing the task, the observer should take note of places where the user has trouble and what caused the trouble. After the task is complete the observer should also ask the user what they liked and disliked about the application.

4. Software Development Environment

4.1 Project Management

4.1.1 Software

Version control is an essential part of any application development project because it allows the team to seamlessly collaborate without having to physically transfer code. Version control also allows programmers to develop new features while protecting their already

functioning code. GitHub is the most commonly used version control tool in the industry, however, BitBucket and GitLab are starting to see more use by developers (GitHub, Inc.). These three applications are all very similar and have almost the same features. For this reason, we chose to use GitHub for our version control because it was the application that we were most familiar with.

There are many project management platforms used by software development teams. GitHub Project is a repository hosting service that also includes many features built for collaboration (GitHub, Inc.). Within GitHub Project, a team can assign tasks and issues to a team member to work on. Issue tracking with GitHub allows for the team to stay on track and know what tasks each member is responsible for. There are other issue tracking platforms such as Azure DevOps, however, it is usually used with Visual Studio and for this reason, the team will use GitHub for project management and issue tracking.

4.1.2 Communication

Due to our small team size, we did not see the need for a specialized communication application like Slack or Microsoft Teams. We chose to communicate simply through text messages and emails.

For communication outside our team with our sponsors, interviewees, and survey participants we utilized an Outlook Group. This meant that emails sent to the team via the group were visible to the entire team. Outlook was chosen because it was the most commonly used email client of both members.

4.2 Programming Environment

4.2.1 Platform

The two most common platforms for mobile applications are Google's Android family of operating systems and Apple's iOS (StatCounter). While there are currently languages and tools being developed to allow for cross-platform development, they are not yet well tested or

documented. For this reason, we had to choose one of the two platforms to develop for. Ultimately, we decided that it would be best to program the application for the Android operating systems. Our reasoning for this was greater accessibility, experience, and cost.

Our sponsors informed us that the majority of sober home residents with a phone have Android phones, not Apple. Additionally, our sponsors indicated that it may be possible to obtain funding to supply each sober home with an Android tablet. This would mean that even residents who did not have Android phones would have access to the application. Finally, even if each sober home does not have a tablet all Android applications can be run on a PC using a free to download emulator. For these reasons developing on Android would make the application accessible to the greatest number of residents.

Our second reason for choosing Android is the cost to develop and publish. Android allows anyone to write and publish applications for its platform. On the other hand, developing applications for iOS requires software designed to run only on Apple operating systems. This would mean that the team would need to find access to a more expensive Apple product in order to program the application. Additionally, once the application is developed there is an Apple Developer license that must be purchased before the application can be made available on iOS. These hurdles further guided our reasoning to choose Android as a development platform over iOS.

Our final reason for choosing Android was the prior programming experience of the team members. Android applications can be written in Java, a language that both team members have extensive knowledge of. Apple iOS applications are primarily written in Objective C or Apple's proprietary language Swift. None of the team members had experience with either of these languages. Because of the prior Java experience the team would be able to start writing code for

an Android application while developing for iOS would require significant time to learn a new language before development could begin.

All of these reasons led to the team choosing to develop the application for the Android platform. Consideration was made toward cross-platform development but, it was decided that the tools required were not well developed enough to use for this application.

4.2.2 Programming Language

Once the decision was made to develop the application for Android, the team next needed to decide what language the application would be written in. There are two official Android programming languages Java and Kotlin (Android). Each language has unique advantages over the other, so an argument was made for each one. Ultimately Java was chosen primarily due to the teams previously mentioned experience with the language.

Java

Java was the first official programming language for the Android platform (Android). This means that there is a wealth of Java tools and libraries for Android development. Also, Android provides extensive documentation and tutorials for writing applications in Java. Because of these reasons alongside the team's background with developing applications using Java, we decided that Java would be the best choice to write our application in.

Kotlin

Kotlin is another official programming language for the Android platform (Android). This means that it has the same amount of support from Android that Java does. Kotlin is also interoperable with Java meaning that it has access to all of the same libraries and tools. In May of 2019 Android announced that Kotlin is now the preferred language, taking the spot from Java (Android). This made a strong argument for the team to choose Kotlin over Java even though they had no experience programming in Java. However, once it was found that Java code could easily

be converted to Kotlin code using Android's Android Studio the team decided to stick with Java for development and make a decision at the end of whether to publish in Kotlin or Java.

4.2.3 Integrated Development Environment

An Integrated Development Environment (IDE) is a powerful tool used by many programmers for application development (Android). An IDE allows you to manage and edit all of the files in the application all in the same place. Many IDEs also allow you to run and debug your application inside a virtual environment. This is helpful because it allows you to easily and quickly run the program rather than having to rebuild it in the terminal after each small change.

Android Studio is the IDE provided by Android for developing Android applications. Android Studio was developed by Android in partnership with JetBrains (Android). JetBrains specializes in making professional IDEs and their IntelliJ IDE is a Java IDE that Android Studio is based on and that both team members already have experience with (Android). Additionally, JetBrains are the creators of the Kotlin language making Android Studio by far the most powerful and feature-rich IDE for Android application development (Android). Because Android applications can be written entirely in Java any Java IDE could be used, but that would mean losing the added functionality of Kotlin and the support of using the official Android supported product.

4.3 Software Tools

4.3.1 Adobe XD

Adobe XD is a free desktop application that allows for the creation of user interface (UI) mockups (Adobe). XD is capable of creating mockups for mobile, desktop, and website applications. UI mockups are used by developers to design how users will interact with their application without implementing any code. XD does this by allowing for the creation of several screens that contain buttons. The designer creates connections from buttons to the desired screen to transition to. The designer is also able to drag and drop elements within each screen to create

the desired look. With these tools a designer is able to quickly and easily create a UI mockup that allows them to visualize what their application UI will look like. Because of its ease of use and the fact that XD is free to use we decided to use it for our UI mockups.

4.3.2 Lucid Chart

LucidChart is a web-based application that can also be used to create UI mockups (Lucid Software Inc.). LucidChart has a paid version and a free version, but only the paid version has the same functionality as XD. Also, because LucidChart is a web application and not a desktop application its performance is not as desirable as that of XD. Finally, the team settled on XD because they had more experience with it than LucidChart.

4.4 Database Management System

A database management system (DBMS) is needed when an application is collecting and storing large amounts of data. The DBMS you use is largely dependent on where the database will be stored, locally or on a remote server, and how the application will communicate with the server. In the case of our application because it is running on a mobile device the database will have to be stored on a remote server and the application will communicate with the database over a network connection. The server that is hosting our database is a Linux server so we could only use a DBMS compatible with Linux. In the end the team had to decide between two of the most commonly used DBMSs, MySQL and PostgreSQL (AltextSoft, 2019).

The two options are very similar in terms of features and functionality. Both are open source, both utilize relational databases, both are ACID compliant, and both have large amounts of community support. The main difference between the two systems is the situations in which each performs well. MySQL performs best with web applications that require a very large number of simple queries to be processed as fast as possible. PostgreSQL, on the other hand, does not operate quite as quickly but is better suited to handle more complex queries allowing for more

complex data analysis (AltextSoft, 2019). Because the purpose of our application is to collect data for later analysis, we decided that PostgreSQL would be the best fit for our application.

5. Software Requirements

Before beginning development on the application, the team first needed to generate a list of requirements that the application must meet. This list of requirements drives the development process and allows the developers to determine what features need to be implemented in the program. Because the team is working in an agile environment this list is not final and can be modified at any time, but it is still important to have an initial list before beginning development.

This requirements list is generated based on the needs of the client and usability for the end-users. In this case, the client is MASH and their needs were determined through interviews with MASH executives and follow up communication. To ensure that the application would be easy for the end-users to use online surveys were sent to residents of sober homes and sober homeowners. Interviews were also conducted with some sober homeowners to get a more in-depth look at how the application would be used.

5.1 Requirement Gathering Strategies

5.1.1 Surveys

When developing an application, it is important that it be usable by the target user base. In order to properly do this the team must have an understanding of the average skill set of the user base. The largest portion of the target user base for the application is the sober home residents. Because of a large number of sober home residents and the vast area that they occupy it is impractical to conduct interviews or focus groups with all of them. This made surveys the ideal method of data collection for this population. Surveys allow for the collection and comparison of data from a large population by using short and specific questions with a limited number of answers. The simplicity of surveys also enables them to be sent electronically allowing for data

collection over a large area. The negative of the simplicity is that surveys do not allow for detailed information about individuals. However, the team knew that it would be impossible to meet all the needs of every user, so it was not crucial to get this type of detailed information.

In order to distribute the survey and analyze the responses, the team used a tool called Qualtrics. Qualtrics allows for the creation and distribution of electronic surveys through a hyperlink (Qualtrics). Qualtrics also provides tools for data analysis that assist in locating trends in the responses. Another benefit of Qualtrics is that it is highly customizable allowing for skippable questions and conditional questions that are shown based on previous responses. Finally, Qualtrics made creating a survey that was compliant with the Institutional Review Board easy. There is an option to provide an informed consent question ahead of the survey and all responses are automatically kept anonymous.

5.1.2 Interviews

Determining the features that needed to go into the application required more complex feedback than could be gathered from a survey. In order to get this information, interviews were conducted with both executives from MASH and various sober homeowners. The interviews were conducted using two different interview types, semi-structured and unstructured, for different purposes.

Unstructured Interviews

Unstructured interviews are a type of interview where the interviewer comes into the interview with no prepared questions, instead, coming up with relevant questions during the interview (Berg, 2001). The lack of prepared questions allows for a more natural discussion that feels more like a conversation than an interview. This strategy also allows the interviewee to provide information which the interviewer can then ask clarifying questions about. For this reason, an unstructured interview often begins with the interviewer instructing the interviewee to tell them

everything about a subject. The interviewer only interjects to clarify a point that they are confused about or to ask the interviewee to elaborate on a point (Berg, 2001). The main weakness of an unstructured interview comes when the interviewee does not have a lot to say about the topic or is unwilling to speak about it. In this instance, because the interviewer does not have any prepared questions, it becomes difficult to continue the conversation and tends to lead to the interview ending before much is gathered.

The team chose to use this unstructured interview approach when talking with executives from MASH. This decision came from the fact that, at the time, the requirements of the application were not well understood. This allowed the executives to provide the team with the requirements they wanted from the application. Once a clear understanding of the requirements had been established, the team provided these executives with some possible features to provide full clarity across the two parties. This strategy generated the initial list of requirements for the application and features to meet these requirements.

Semi-Structured Interviews

In semi-structured interviews the interviewer compiles a basic list of questions to ask but also allows the conversation to stray from the questions when appropriate (Berg, 2001). This differs from a structured interview where the interviewer has a complete list of questions and is only interested in the answers to those questions. Semi-structured have the benefit of allowing the interviewee to answer questions that the interviewer may not have thought to ask. This differs from unstructured interviews because the interviewer may interrupt the conversation if they feel that it no longer pertains to what they are hoping to gather. In this case the interviewer can either restate the question if they feel they still do not have a clear answer or ask the next of their prepared questions (Berg, 2001). It is also important to note that through the course of the conversation the interviewee may answer a question they are not directly asked. The interviewer should make note

of this and be sure not to ask that question again later. Also, the order in which the questions are asked is not crucial so the interviewer may choose to ask the question most relevant to the conversation at that time. This type of interview when attempting to gather information on a few specific topics. This strategy can also have a natural conversation feel while avoiding the interview ending abruptly due to an interviewee with not much to say.

When interviewing sober homeowners, the team used semi-structured interview strategies to learn how sober homes operate as it pertains to the application. The team chose semi-structured interviews because the only information that pertained to the application was relevant so an unstructured interview could have resulted in too wide a range of information. Since these interviews happened after the interview with the MASH executives the team already had their initial list of features compiled. Some of these features were designed to assist sober homeowners in running and managing their homes. The purpose of these interviews was to determine how relevant each of these features was to how a sober home actually operates. The majority of the prepared questions were designed to gauge the team's understanding of sober home operation versus the reality of sober home operation. Additional questions were asked to try and get an idea of the common habits of the residents. This provided valuable insight into the trends of sober home residents without having to interview a large number of residents.

5.2 Functional Requirements

Functional requirements are those that are given by the client concerning what the application should do. These could be featured that the client specifically asks to be in the application. They could also be features that are not explicitly requested by the client but are necessary in order to implement other features (Futrell & Shafer, 2002). We gathered these requirements through interviews with the MASH directors. The requirements for this application are:

- Able to create an account so that information persists after the application is closed

- Accounts broken into three separate tiers with different permissions
- A timeline with which residents can document their recovery
- Incentives for residents to add to their timeline
- The ability of a resident to join a sober home with the owner's approval
- The ability for homeowners to moderate the posts made by residents in their homes
- The ability for homeowners to send messages to all residents in their home
- Survey creation tool so that MASH can create and distribute surveys to residents
- Survey data, stored anonymously, is accessible by MASH
- Account persistence after a resident leaves a home
- HTTP server for database connectivity

5.3 Non-Functional Requirements

Nonfunctional requirements are not requested by the client or related to those requested by the client (Futrell & Shafer, 2002). These requirements relate to performance metrics, scalability, maintainability, and security. We obtained these requirements from our previous software development experiences and knowledge of good development practices. The nonfunctional requirements for this application are:

- Database ability to support up to 1,000 users
- Data should be stored securely and not be susceptible to injection attacks
- Database queries should happen in a reasonable time less than 3 seconds
- The application should be able to run on any Android device
- The application should be able to run on any currently supported Android operating system
- The code should be well documented and easy to read

5.4 Epic and User Stories

In Agile methodology, the list of requirements is converted into a list of user stories. A user story is used to highlight a single specific action that a user may want to perform (Rubin, 2013). User stories are written as a sentence with the following format: As a TYPE OF USER I want to [action to perform] so that I may [reason for performing the action]. These user stories are sorted into epics based on commonality. Epics are collections of user stories that all apply to the same common theme of the application. Through initial requirements gathering the user stories were broken into 6 different epics: accounts, timeline, social, administration, data collection, and resident support. These epics and their corresponding user stories are listed below:

Epic - Accounts

As a USER I want to [create an account] so that I may [save my data in the app].

As a RESIDENT I want to [login to an account] so that I may [access my profile, timeline, and chat functionality].

As a USER I want to [keep my account private] so that I may [prevent others from viewing sensitive information].

As a USER I want to [add a profile picture] so that [others may know who I am].

As a RESIDENT I want to [publicize my account] so that I may [share my progress with others].

As a USER I want to [revise my account information] so that I may [fix any errors].

As a USER I want to [have a separate tier of account] so that I may [perform supervisory functions for my houses].

As a HOUSE OWNER I want to [authenticate that I am a home-owner] so that I may [perform supervisory functions for my houses]. * Score: 3

As a MASH DIRECTOR I want to [authenticate that I am mash director] so that I may [see data from all sober homes in MASH]. * Score: 3

Epic - Timeline

As a RESIDENT I want to [view my timeline] so that I may [see my recovery progress].

As a RESIDENT I want to [create a custom timeline event] so that I may [share personal progress].

As a RESIDENT I want to [set a goal] so that I may [remind myself what I am aiming for].

As a RESIDENT I want to [make goals and events public] so that I may [show others how I am progressing].

As a RESIDENT I want to [have a count showing how long I have been sober] so that I may [be motivated by the progress that I have made].

As a RESIDENT I want to [comment on other residents' timelines] so that I may [interact with fellow residents and give them encouragement].

As a HOUSE OWNER I want to [approve of resident timeline events] so that I may [verify or deny whether a timeline event actually took place].

Epic - Social

As a RESIDENT I want to [chat with other people who are recovering] so that I may [encourage others and be encouraged by others].

As a SOBER HOME GRADUATE, I want to [chat with other graduates] so that I may [network with other graduates]

As a HOUSE OWNER I want to [chat with other house owners] so that I may [share and get helpful tips for running a house].

As a HOUSE OWNER I want to [send announcements to the residents of my house] so that I may [update residents with events].

Epic - Administration

As a HOUSE OWNER I want to [remove residents from a sober home] so that I may [remove people that are disruptive to the sober home].

As a HOUSE OWNER I want to [graduate residents] so that I may [clear space if a resident has graduated].

As a MASH DIRECTOR I want to [create a sober home in the application] so that I may [add residents to it].

As a HOUSE OWNER I want to [approve new residents] so that I may [regulate who joins the application].

As a RESIDENT I want to [enter a house code] so that I may [have access to my house].

Epic - Data Collection

As a HOUSE OWNER I want to [explain why someone was kicked out of a sober home] so that I may [justify why a resident was removed].

As a MASH DIRECTOR I want to [collect basic data from residents] so that I may [know how best to support them].

As a MASH DIRECTOR I want to [collect information through periodic surveys] so that I may [know the state of the residents across houses].

As a MASH DIRECTOR I want to [create a survey] so that I may [distribute surveys to all sober home residents].

As a MASH DIRECTOR I want to [control the number of times a survey is taken by a resident] so that I may [ensure there is no data overload].

As a RESIDENT I want to [opt out of sensitive questions] so that I may [be comfortable with the information that I am sharing with the application].

As a MASH DIRECTOR I want to [compare data between sober homes] so that I may [know which sober homes are doing better and why].

As a MASH DIRECTOR I want to [access data in usable formats] so that I may [perform further analysis].

As a MASH DIRECTOR I want to [see visualizations of the data] so that I may [get a quick overview of the data].

As a MASH DIRECTOR I want to [display percentages of goals met] so that I may [get an overview of the data].

As a MASH DIRECTOR I want to [see why residents were kicked out of a sober home] so that I may [know if the sober homeowner is acting fairly].

As a RESIDENT I want to [rate the quality of my sober home] so that I may [let other people looking to be residents know the quality of the sober home].

As a RESIDENT I want to [provide the reason I left a sober home] so that I may [provide my side of the story].

<u>Epic - Resident Support</u>

As a RESIDENT I want to [access emergency info] so that I may [call for help if needed].

As a RESIDENT I want to [find local support group meetings] so that I may [know where to find local support groups].

As a RESIDENT I want to [report incidents to a higher authority] so that I may [prevent negligence or injustice].

6. Design

6.1 User Interface

6.1.1 Original Log In Screen Mock-Up

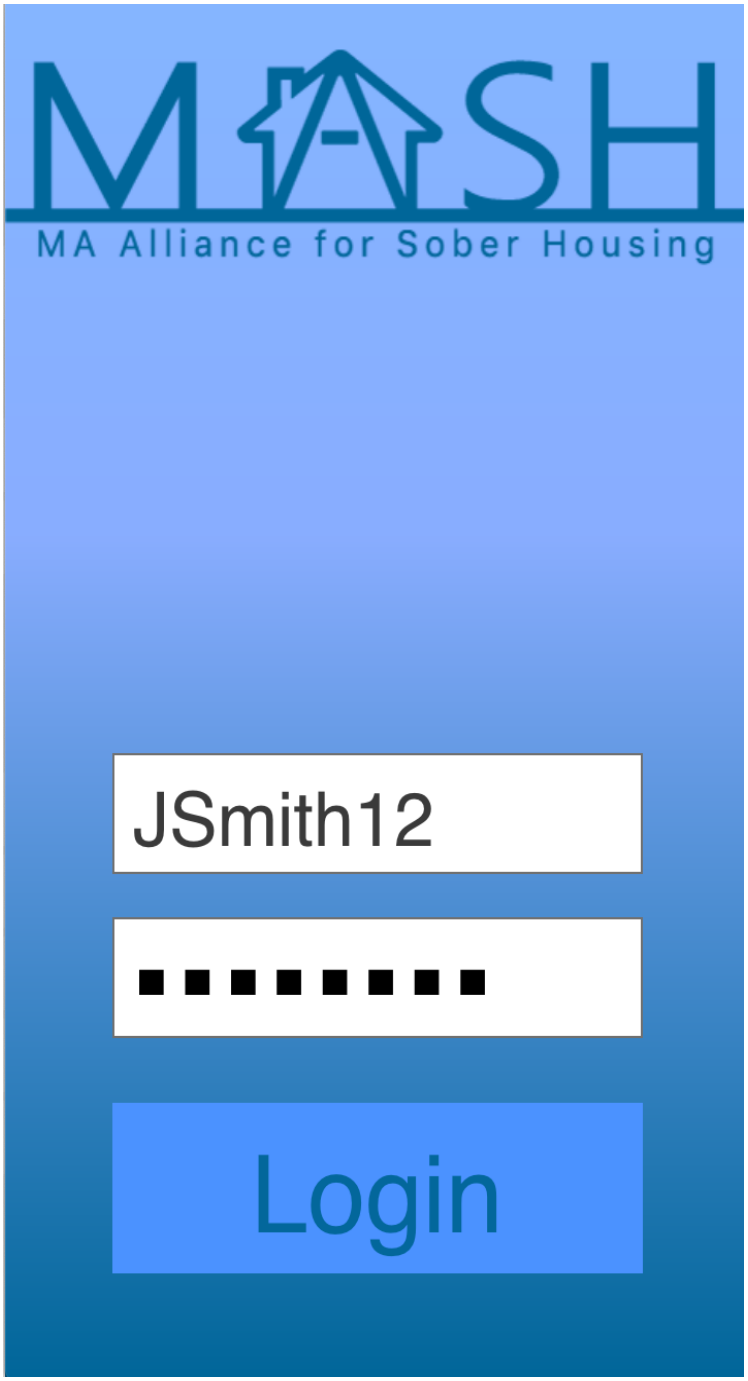

Figure 3: Original Log In Screen

6.1.2 Final Log In Screen

Figure 4: Final Log In Screen

6.1.3 Final Home Screen

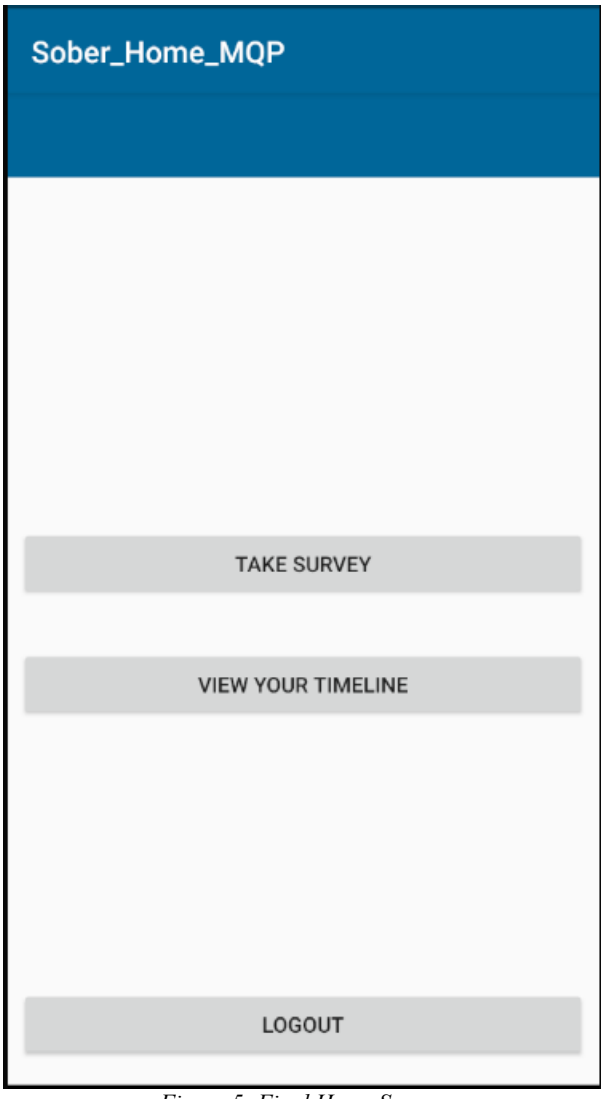

Figure 5: Final Home Screen

6.1.4 Final Timeline Screen

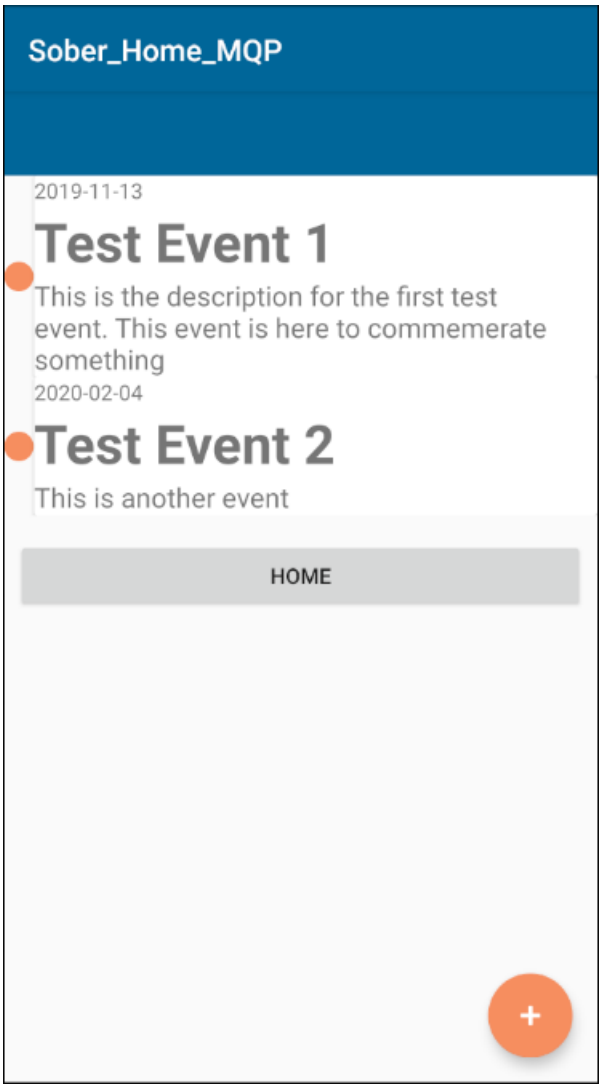

Figure 6: Final Timeline Screen

6.1.5 Final Survey Creation Screen

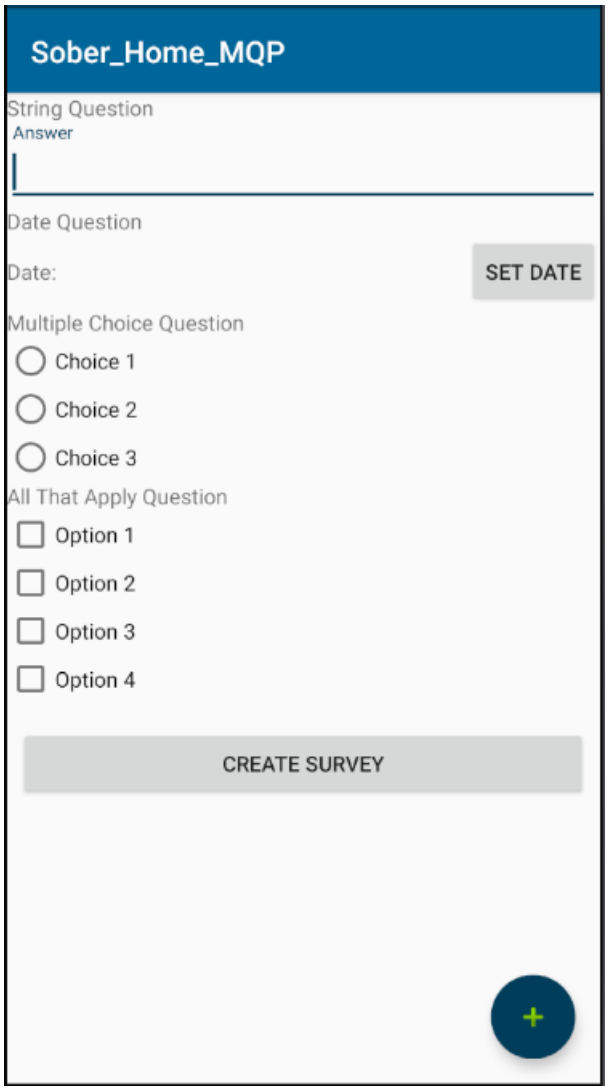

Figure 7: Final Survey Creation Screen

6.2 ERD

6.2.1 Original ERD

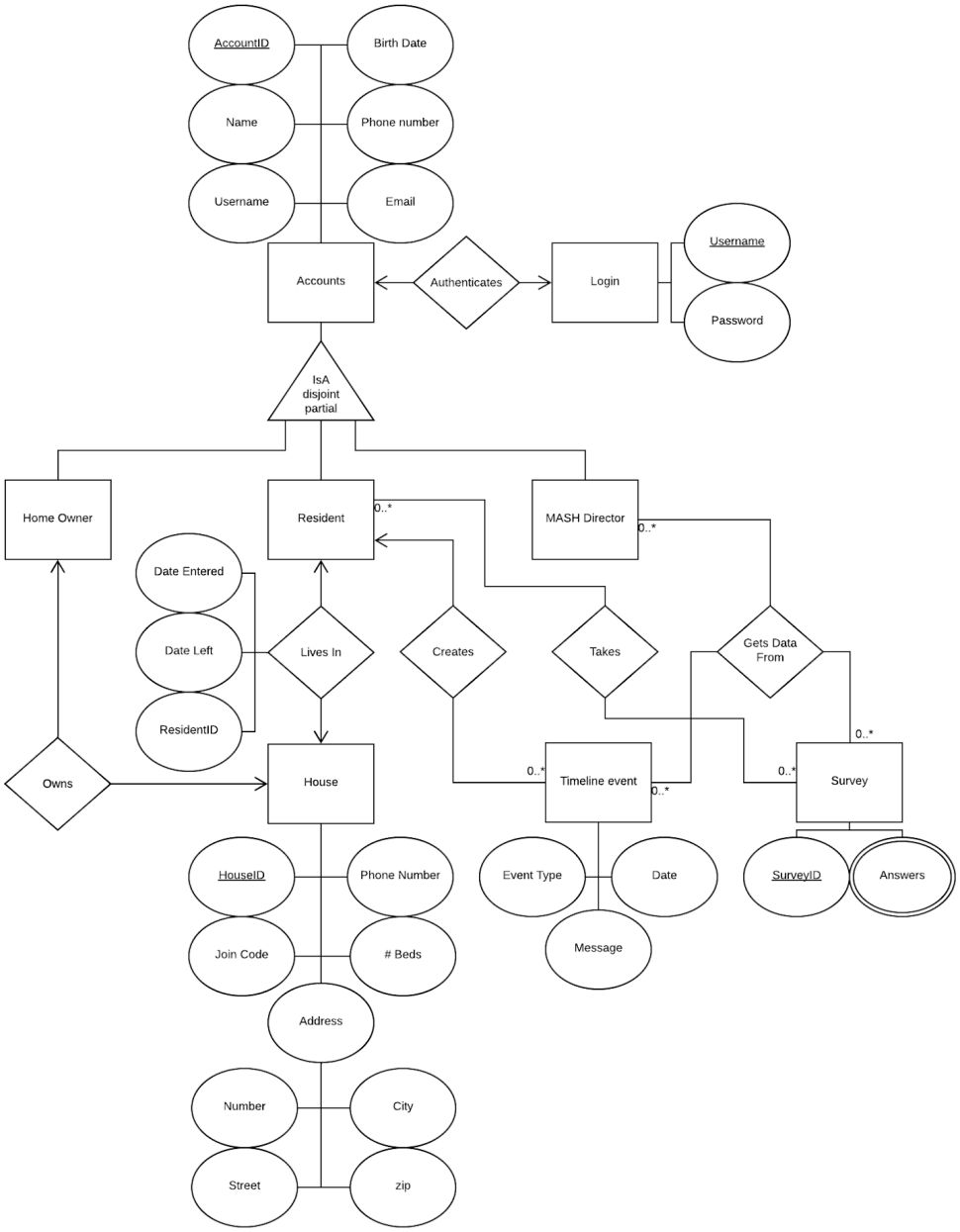

Figure 8: Original ERD

6.2.2 Final ERD

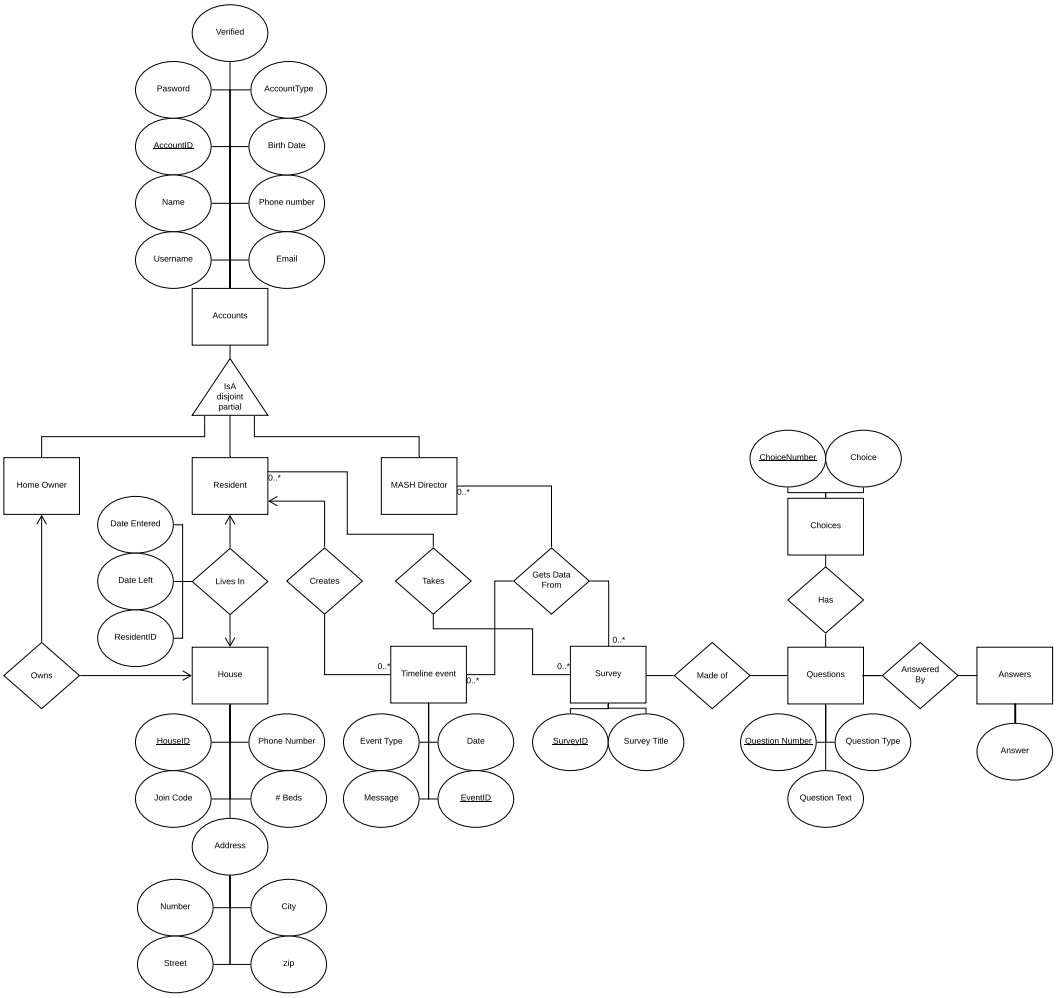

Figure 9: Final ERD

6.3 Class Diagram

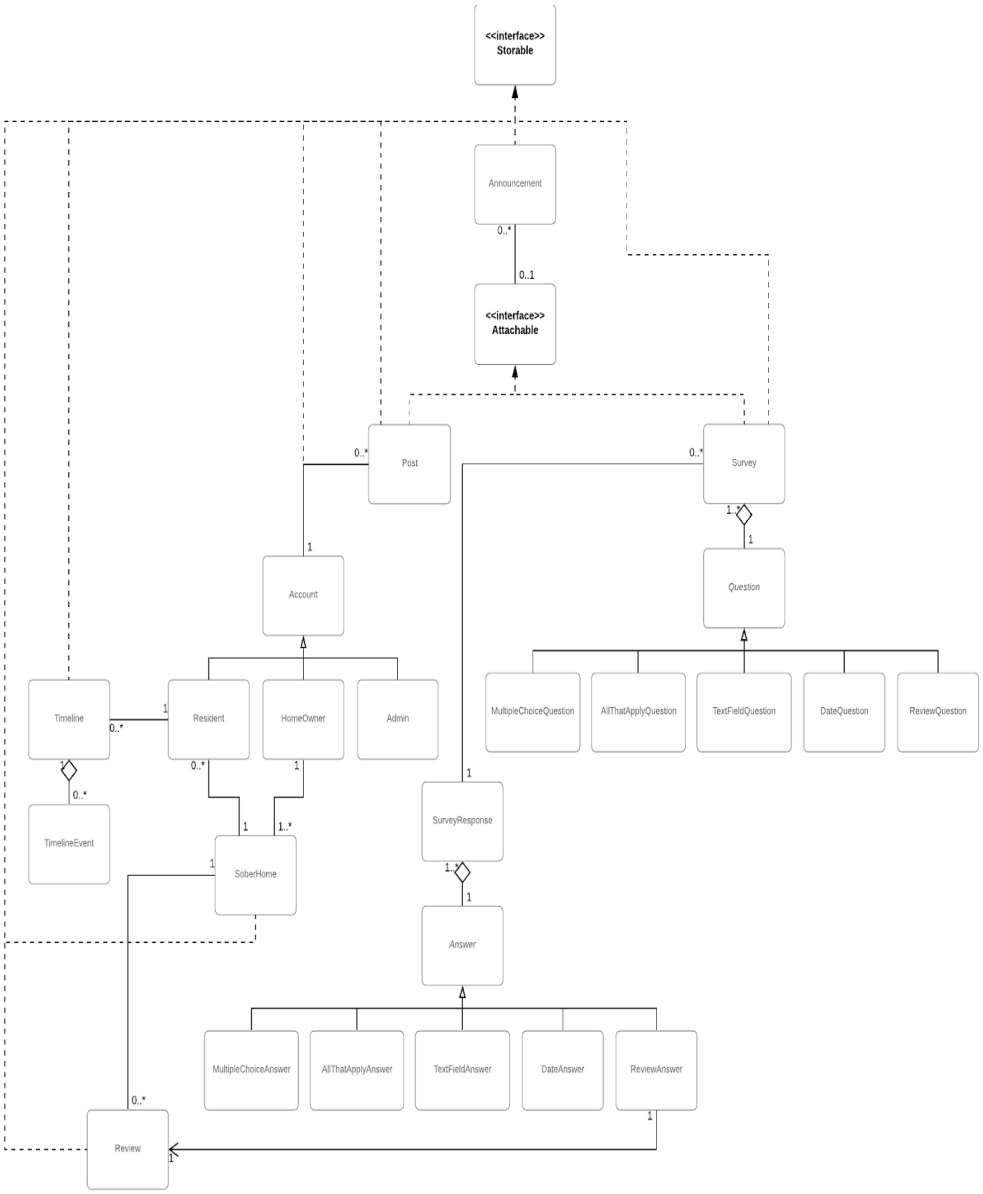

Figure 10: Class Diagram

The organization of the class diagram is used to represent the types of data that need to be saved from the application. These classes are related to the types of user accounts, components of the survey feature, and the resident timeline feature.

6.4 Context Diagram

6.4.1 Original Context Diagram

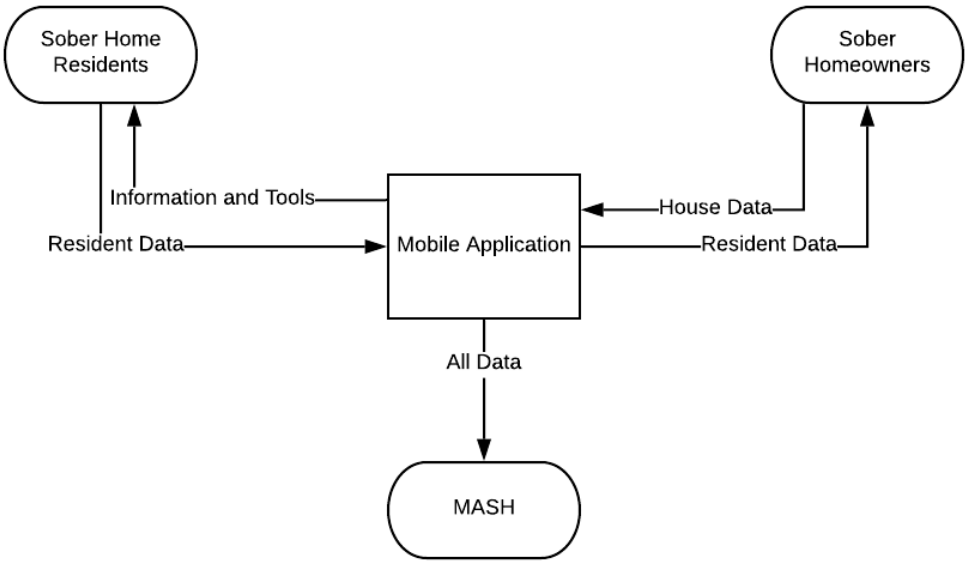

Figure 11: Original Context Diagram

6.4.2 Final Context Diagram

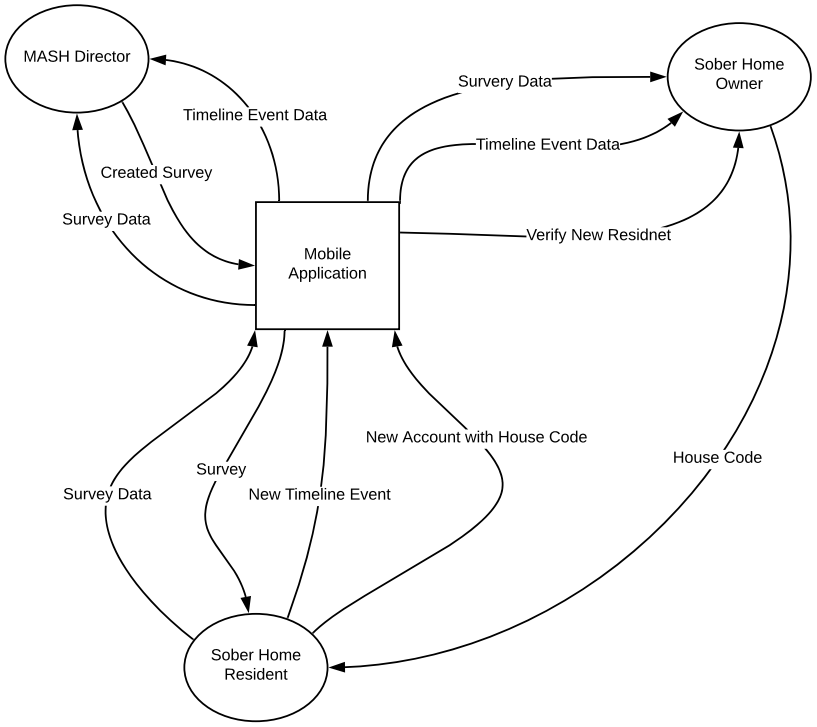

Figure 12: Final Context Diagram

6.5 Design Choices

Even though we had last year's team's source code, we wanted to also create our own application with our design. There were design decisions that we did not initially understand so we did our own research and used what we thought were simpler designs. However, we ran into issues at the beginning of development because of this. We realized that it is not good practice to directly connect Android apps to remote Postgres databases because the source code can be decompiled, and anyone would have access to our application. Due to the privacy we want to provide to our users, we decided that this would create a lack of security in our application and access to the confidential data could be compromised. So, we took a step back and looked deeper into what the previous team did and began to understand why they had to use the designs they

used. We used Volley to take care of connecting to the database. However, using Volley came with its own hurdles including how JSON or GSON objects are passed through.

7. Software Development

7.1 Agile Development

Iteration 1

Planned to complete:

- As a USER I want to [create an account] so that I may [save my data in the app]. Score: 1
- As a RESIDENT I want to [login to an account] so that I may [access my profile, timeline, and chat functionality]. Score: 1
- As a RESIDENT I want to [view my timeline] so that I may [see my recovery progress]. Score: 2
- As a USER I want to [revise my account information] so that I may [fix any errors]. Score: 2

Completed:

- As a RESIDENT I want to [login to an account] so that I may [access my profile, timeline, and chat functionality]. Score: 1
- As a RESIDENT I want to [view my timeline] so that I may [see my recovery progress]. Score: 2
- As a RESIDENT I want to [create a custom timeline event] so that I may [share personal Progress]. Score: 2

3 total user stories completed out of 40 total user stories

5 total user story points completed out of 70 total user story points

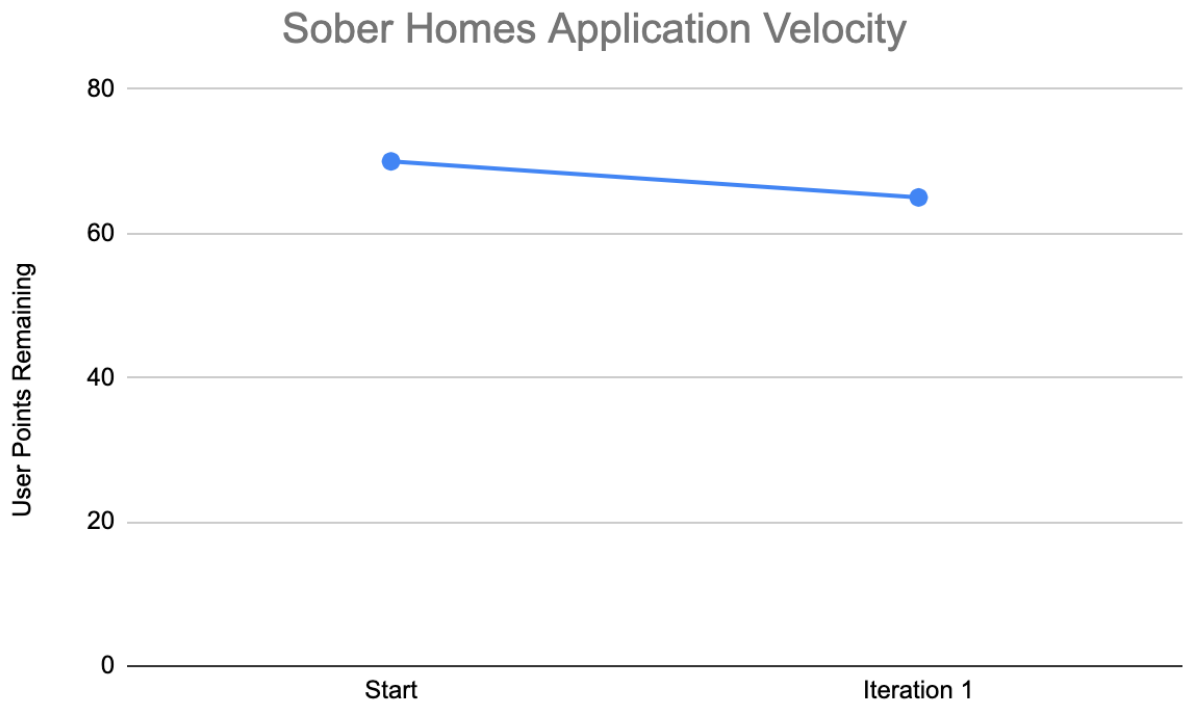

Figure 13: Iteration 1 Velocity Chart

During this iteration, we focused first on creating the activity screens for the application. We created the login screen, a home screen, a timeline screen, and an add event screen. We researched different timeline libraries and decided to go with the one that last year's team used. We decided what a timeline event would look like and created a screen that will later be used on the main timeline page. Once we created the activity screens, we started the backend functionality for the activity java classes. We logged in to the application without connecting to the database using two test strings. We also made the buttons that bring the users to different screens.

Unfortunately, we were unable to complete all of the original user stories we wanted to implement this iteration because we could not get the server and database running. Our contact who will be setting these up for us, was on vacation this week, therefore we decided to work on some screens and functionalities we weren't originally planning on. Overall this iteration went

smoothly. The team was able to create a basic functioning application even though this was the team's first real experience using Android Studio.

Iteration 2

Planned to complete:

- As a USER I want to [create an account] so that I may [save my data in the app]. Score: 1
- As a USER I want to [revise my account information] so that I may [fix any errors]. Score: 2
- As a USER I want to [have a separate tier of account] so that I may [perform functions based on my role]. Score: 1
- As a HOUSE OWNER I want to [authenticate that I am a home-owner] so that I may [perform supervisory functions for my houses]. Score: 3
- As a MASH DIRECTOR I want to [authenticate that I am mash director] so that I may [see data from all sober homes in MASH]. Score: 3

Started but not completed:

- As a USER I want to [create an account] so that I may [save my data in the app]. Score: 1
- As a USER I want to [revise my account information] so that I may [fix any errors]. Score: 2
- As a USER I want to [have a separate tier of account] so that I may [perform functions based on my role]. Score: 1

Completed:

- We were unable to fully complete any of our user stories because the server was not yet created. All of the stories we started working on require a database for full functionality.

3 total user stories completed out of 40 total user stories

5 total user story points completed out of 70 total user story points

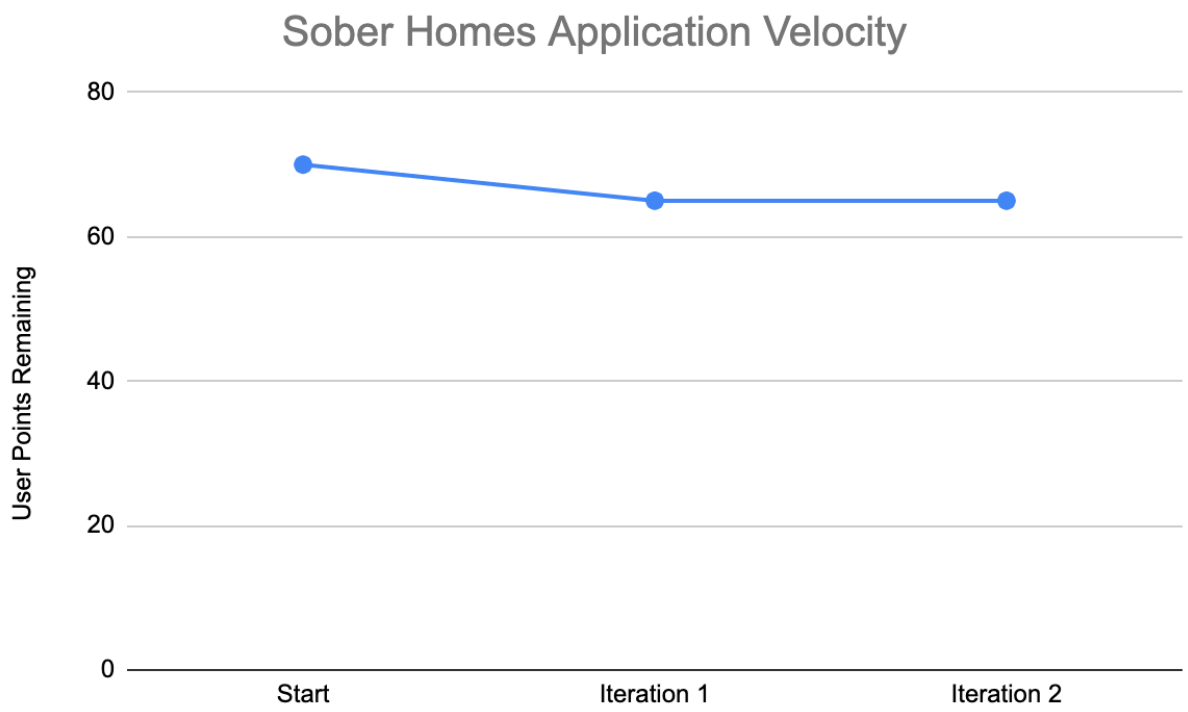

Figure 14 : Iteration 2 Velocity Chart

*Note the number of user stories went up because we broke one user stories into three smaller more specific stories.

We faced a large roadblock this iteration in still not having access to the database on the server. This meant that once again we were limited to only being able to construct the user interfaces for features without implementing the backend. However, we were hesitant to create too many more user interfaces without being sure that the current application was in working order. For this reason, we used this iteration to take a closer look at our user stories and adjust our plan after getting some experience with application development for Android.

One big decision we discussed was how surveys would be implemented in the app, specifically how they would be created. Previously, we had been working with MASH to come up with a list of questions they wanted to put on the survey that we would then hard code into the app.

After some discussion, we considered the possibility of implementing a survey creation feature for the app and decided it would be possible. We reached out to our sponsors to see if that would be a feature they would want before starting to implement.

Another topic of discussion this week was the UI theme. Currently, we have put no work into the UI theme and are just using defaults. After much discussion, we were unable to come to a final decision about what the theme of the app should be. However, we did install some tools that will help us to design a theme and easily apply it to our application when we are ready to.

Iteration 3

Planned to complete:

Get the database working

Get the back end working for these user stories and previous user stories

- As a USER I want to [create an account] so that I may [save my data in the app]. Score: 1
- As a USER I want to [revise my account information] so that I may [fix any errors]. Score: 2
- As a USER I want to [have a separate tier of account] so that I may [perform functions based on my role]. Score: 1
- As a MASH DIRECTOR I want to [create a survey in the app] so that I may [collect specific date]. Score: 8

Completed:

- Database is working
- Finalized stories previously marked as finished because they now utilize the database
- As a USER I want to [have a separate tier of account] so that I may [perform functions based on my role]. Score: 1

4 total user stories completed out of 40 total user stories

6 total user story points completed out of 70 total user story points

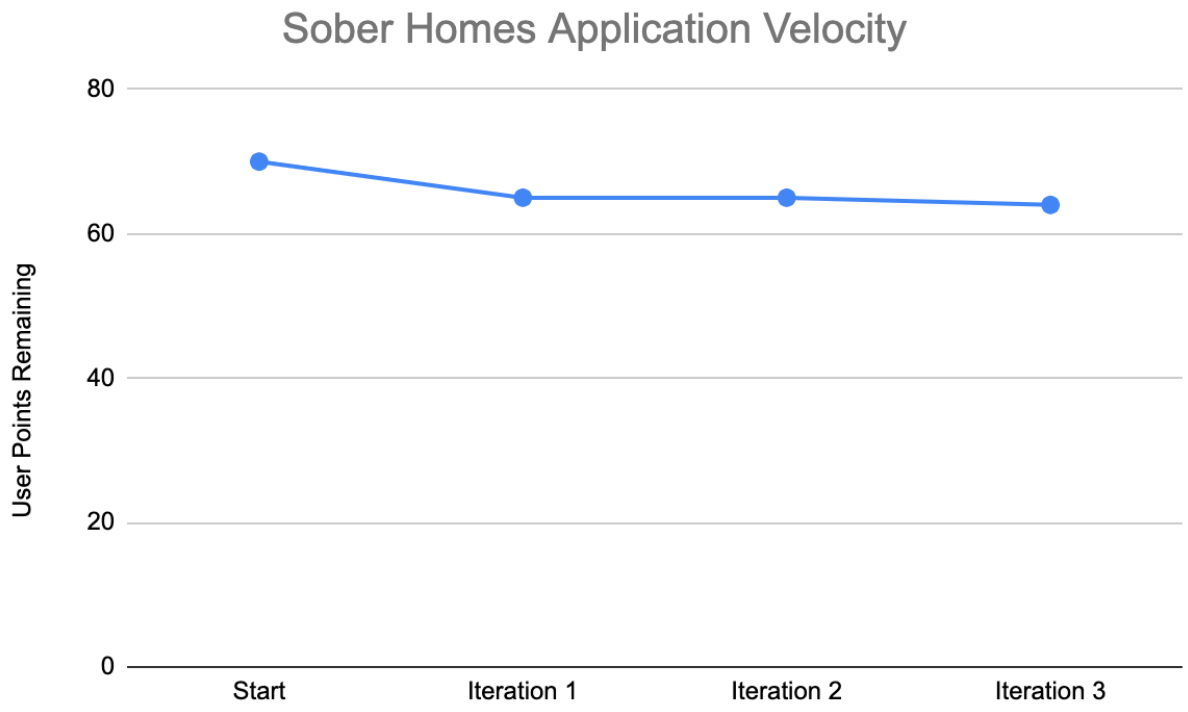

Figure 15: Iteration 3 Velocity Chart

During this iteration, the team completed 2 new user stories as well as implemented database functionality for previously finished user stories. The focus of this iteration was getting our application to be able to interface with the database. We finally got the database up and running on the server and were able to load the schema that we had previously created onto it. However, it took most of the week to get the database configured to allow remote devices to connect to the database. While we were working with the WPI IT professional who was configuring the database, we also implemented the backend classes that would allow us to interface with the database once it was complete. This included writing functionality for creating an account and querying the account database as part of the login process. Also, storing timeline events to the database and

loading all the events for a user so that they are able to be displayed. All of these features were able to connect to the database and were tested individually using JUnit testing and shown to work. However, when we tried to run these features through the application emulator, we were unable to connect to the database. Once that issue is solved our application will be able to fully interact with the database and we will be able to begin developing new features. We also spent some time this week brainstorming how a survey creation tool might look and work in our application.

Iteration 4

Planned to complete:

- Getting the database connection fully working with Volley and an HTTP Server

Completed:

- Seeing messages on the server and sending responses back (the server is completing all of the tasks)

4 total user stories completed out of 40 total user stories

6 total user story points completed out of 70 total user story points

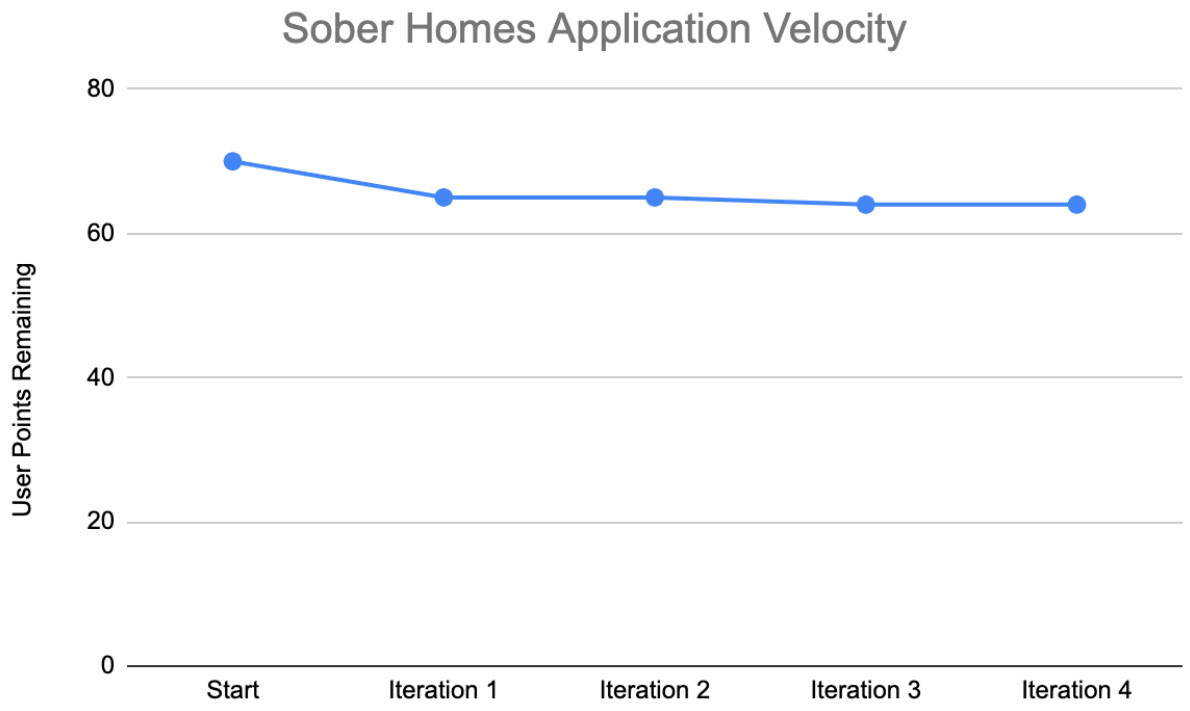

Figure 16: Iteration 4 Velocity Chart

During this iteration, we again were not able to complete any user stories as we still were unable to connect to the database. The trouble that we encountered this week is that Android apps are not able to connect directly to remote Postgres databases. The reason for this stems from a major security issue. Connecting to a Postgres database required putting an authorized username and password into the source code. Because Android source code can be decompiled, this means that anyone who downloaded the application could have access to edit and view the database. To address this issue, we set up an HTTP server that our application can communicate with that will then interface with the database.

The HTTP server was written using the Oracle HTTP server API that is built into Java. We used the database manager class we had already written for our application and put it on the server in order to save time. Then we set up an HTTP handler object for each method in the database class that simply got parameters from the HTTP request, ran the method, and passed the response

back to the application in an HTTP response. Any primitive parameters were passed as additional HTTP headers while object parameters were converted to JSON and passed in the body. To convert Java objects to JSON we used the Google GSON library that allows conversion between any Java class and JSON text and back.

To send and receive HTTP messages from the application we utilized the Android Volley library. The Volley library provides an API for sending HTTP requests from an Android application. This API handles formatting the HTTP messages, sending the requests to the specified URL and receiving the HTTP response. This API also allows for the creation of custom request types that convert the response into a specified type.

Currently, with this method we are able to send messages that are being received by the server and the server is responding. However, because Volley runs asynchronously our code does not wait for the response before continuing to run. This is causing crashes in our code because it is attempting to process the response before they are received. In order to resolve this, we looked into how to run Volley synchronously, but have not yet found a solution.

Iteration 5

Planned to complete:

- Getting the database connection fully working with Volley and an HTTP Server

Started:

- As a MASH DIRECTOR I want to [create a survey] so that I may [distribute surveys to all sober home residents].

Completed:

- As a USER I want to [create an account] so that I may [save my data in the app]. Score: 1
- Fully implemented the login and timeline event creation processes to utilize the database.

5 total user stories completed out of 40 total user stories

7 total user story points completed out of 70 total user story points

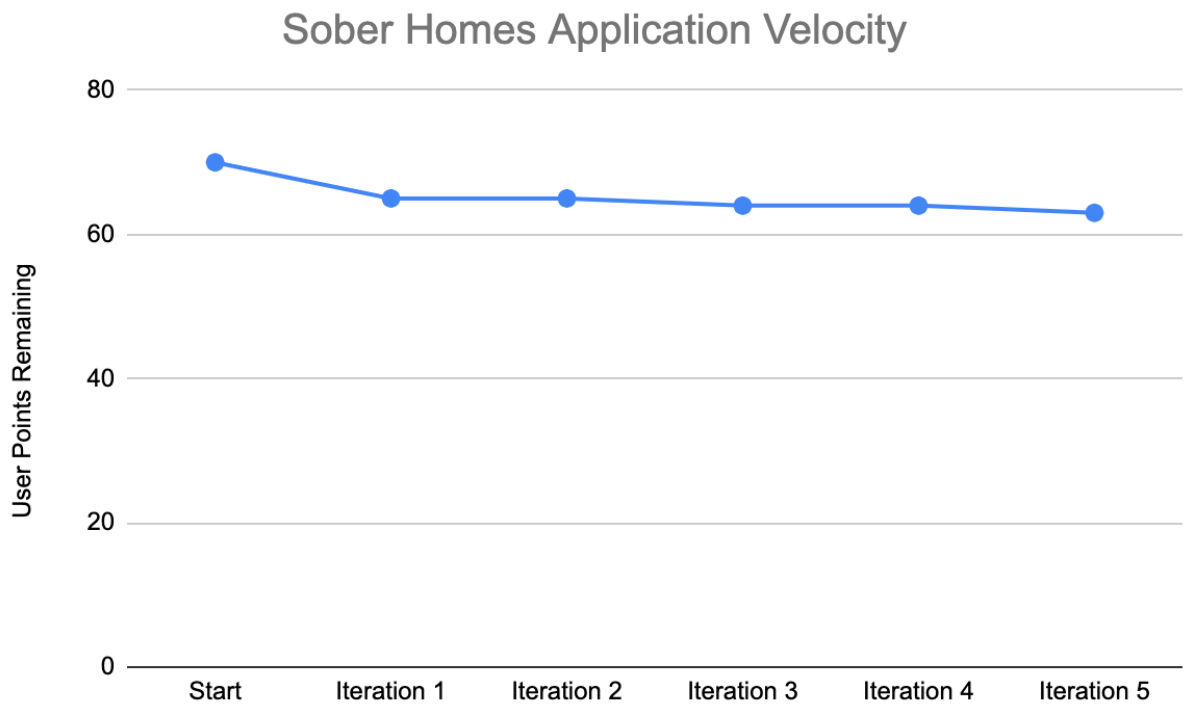

Figure 17: Iteration 5 Velocity Chart

During this iteration, we were finally able to log in, create an account, and add events to the timeline using the database.

The login functionality currently only works on a "good" login, and we are waiting to look into how to handle errors if there is a "bad" login attempt. We initially used HTTP error codes but ran into some issues with Volley when using this strategy. Another strategy to consider would be to use HTTP headers

Users can now create an account if they do not already have one, and the only constraint currently is that their username must be unique when compared to the database. The user will also select what type of user they are (Resident, MASH Director, or Home Owner) before they are logged into the home screen. The account information is saved to the database and that user can

now log in with their username and password from the login screen. We needed to talk with our sponsors about how to authenticate what level of account a user should have. The timeline events a user creates are now added to the database and will be loaded when the user logs in. The events stay after logout and subsequent login.

Now that we have a functioning application with database connection, we decided that the survey creation tool will be our next step. We discussed how a MASH Director or Home Owner would create a survey and what that screen would look like. We imagined it would be similar to creating a timeline event functionality, where a MASH Director or Home Owner would be able to select a question type and add the question and it would be loaded to the survey screen. The team was working on figuring out the best way to achieve this functionality. We also created the screens to display newly created surveys.

In looking through the previous team's application we discovered their use of Android Fragments and Dialogs. We thought that those types of tools may have been able to create some of the functionality we hoped to achieve. However, as neither of us had experience with those tools we planned to do further research into these tools to better understand how we could utilize them.

Iteration 6

Planned to complete:

- As a MASH DIRECTOR I want to [create a survey] so that I may [distribute surveys to all sober home residents]. Score: 20
- As a MASH DIRECTOR I want to [collect basic data from residents] so that I may [know how best to support them]. Score: 3
- As a MASH DIRECTOR I want to [collect information through periodic surveys] so that I may [know the state of the residents across houses]. Score: 3

Started:

- As a MASH DIRECTOR I want to [collect basic data from residents] so that I may [know how best to support them]. Score: 3
- As a MASH DIRECTOR I want to [collect information through periodic surveys] so that I may [know the state of the residents across houses]. Score: 3

Completed:

- As a MASH DIRECTOR I want to [create a survey] so that I may [distribute surveys to all sober home residents]. Score: 20

6 total user stories completed out of 40 total user stories

27 user story points completed out of 70 total user story points

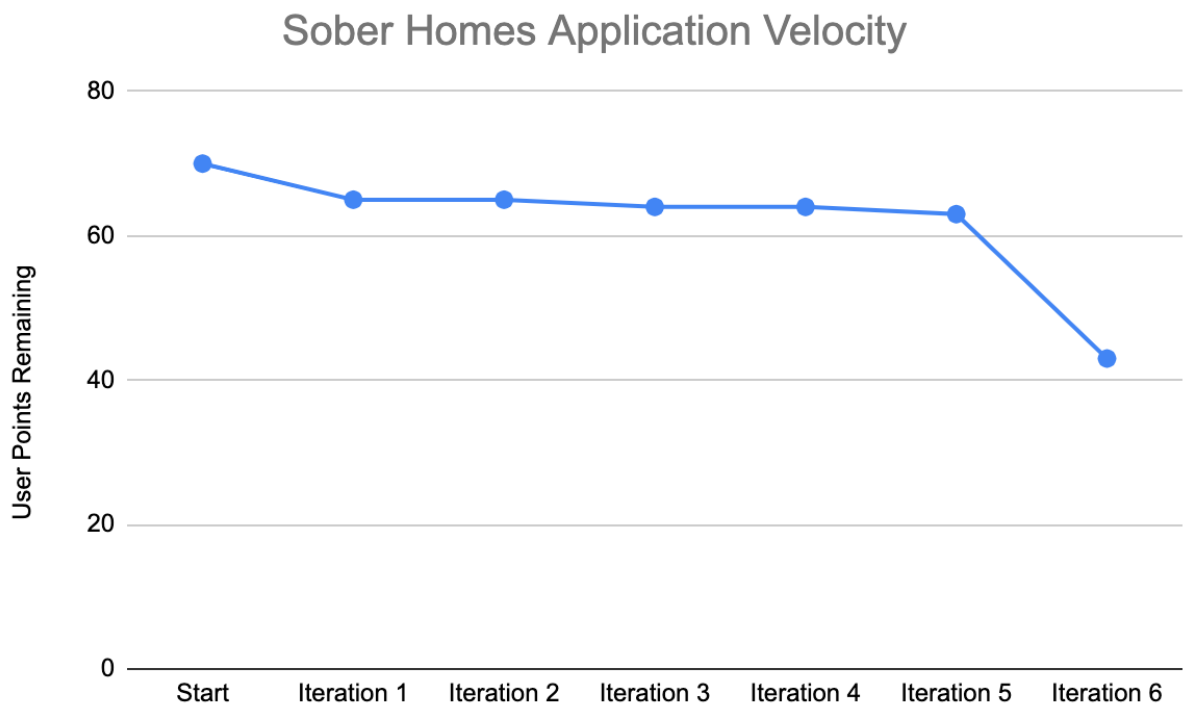

Figure 18: Iteration 6 Velocity Chart

During this iteration, we focused on the data collection aspect of the application through surveys. Through further discussion with our sponsors, we decided that implementing a tool that

allows MASH directors and home-owners to create and modify surveys would be useful. The tool that we implemented allows for the creation of four different types of questions: text field questions, date questions, multiple-choice questions and all that apply questions. Because all of the questions are stored as a list of an abstract Question class the default GSON class was not able to deserialize the list. To get around this we had to write a custom JSON handler to deserialize the abstract questions into their proper classes. We also had to implement database functions and triggers to create sequences that properly number the questions for each survey and the options for each survey. These will allow for proper alignment of the questions and options when the survey is loaded from the database.

In order to create the user interface for the survey creation tool, we made use of an Android feature known as dialog fragments. Dialog fragments allow for an additional screen to be displayed without closing the activity that spawns the dialog. This means that the list of questions created can be stored in the main activity without having to save to the database each time screens are switched. This allows MASH directors to see what the survey will look like as they are creating it. The next feature to be implemented is to allow for users to take the surveys that have been created and ensure their answers are properly saved. During this iteration, we created the screens to display the survey to the user but have not yet implemented the back end to store the answers. The next step is saving their answers to the database in a way that they will be associated with the correct questions. This will allow MASH to analyze the data that is collected from the surveys. In order to ensure this happens properly, we will have to be careful about how we load the surveys and ensuring proper ordering is maintained.

Iteration 7

Planned to complete:

- As a MASH DIRECTOR I want to [collect basic data from residents] so that I may [know how best to support them]. Score: 3
- As a MASH DIRECTOR I want to [collect information through periodic surveys] so that I may [know the state of the residents across houses]. Score: 3
- As a RESIDENT I want to [opt out of sensitive questions] so that I may [be comfortable with the information that I am sharing with the application]. Score: 1
- As a MASH DIRECTOR I want to [control the number of times a survey is taken by a resident] so that I may [ensure there is no data overload]. Score: 3

Completed:

- As a MASH DIRECTOR I want to [collect basic data from residents] so that I may [know how best to support them]. Score: 3
- As a MASH DIRECTOR I want to [collect information through periodic surveys] so that I may [know the state of the residents across houses]. Score: 3
- As a RESIDENT I want to [opt out of sensitive questions] so that I may [be comfortable with the information that I am sharing with the application]. Score: 1

9 total user stories completed out of 40 total user stories

34 user story points completed out of 70 total user story points

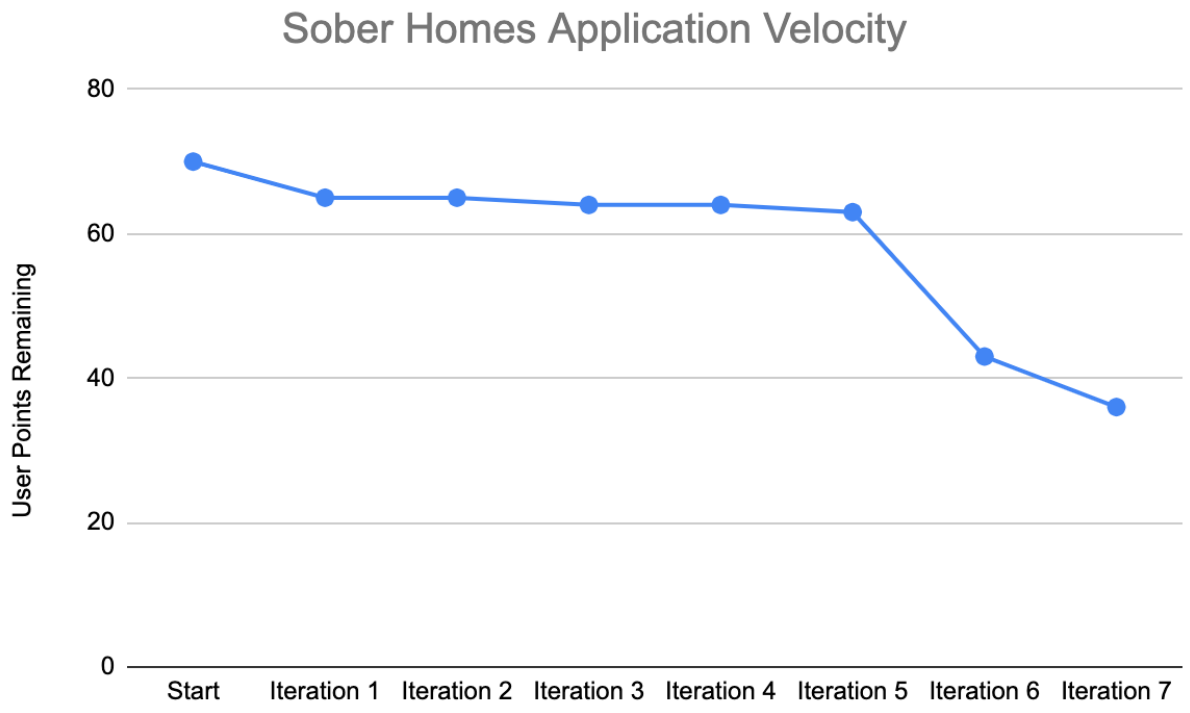

Figure 19: Iteration 7 Velocity Chart

During this iteration we implemented the ability for the residents to take surveys and have their answers recorded in the database. The first task required to complete this was allowing residents to select which survey they would take. In order to do this, we created an AlertDialog and loaded all the survey titles from the database into it. This passes the correct survey ID back to the database so that it loads the correct survey when entering the survey activity. Loading the survey required implementing the same customized JSON handler to decode abstract question objects on the application side. In order to indicate when to use this decoder the GsonListRequest constructor was overloaded allowing for a boolean that indicated if the decoder should be used. If no boolean is passed the decoder is not used.

The survey activity utilizes a recycler view, similar to the timeline activity, in order to display the questions. However, because there are four different types of questions, we had to create four different view holders and use the type field in the question object to determine which

view holder to create. We also created an abstract view holder that each of the four other view holders extended that contained a method allowing to retrieve the answer from each view holder. Getting this activity working correctly required the team to complete extensive research into how Android RecyclerView objects work. This included understanding using multiple view holders and accessing the view holders after creation.

The final task that needed to be completed for the survey activity was to store the answers in the database. We decided that the individual answers would all be submitted at once instead of after answering each question to minimize server calls. We also decided to store the answers anonymously so that MASH would not be able to see who answered what. Because of this decision, combined with the fact there is no local application database, there was no easy way for us to locally save a user's answers so they could return later. Also missing was a way to prevent residents from taking a survey multiple times. This was a feature we planned to implement in the next iteration.

Iteration 8

Planned to complete:

- As a MASH DIRECTOR I want to [control the number of times a survey is taken by a resident] so that I may [ensure there is no data overload]. Score: 3
- As a HOUSE OWNER I want to [approve new residents] so that I may [regulate who joins the application]. Score: 3
- As a RESIDENT I want to [enter a house code] so that I may [have access to my house]. Score: 1

Started:

- As a HOUSE OWNER I want to [approve new residents] so that I may [regulate who joins the application]. Score: 3

- As a RESIDENT I want to [enter a house code] so that I may [have access to my house]. Score: 1

Completed:

- As a MASH DIRECTOR I want to [control the number of times a survey is taken by a resident] so that I may [ensure there is no data overload]. Score: 3

10 total user stories completed out of 40 total user stories

37 user story points completed out of 70 total user story points

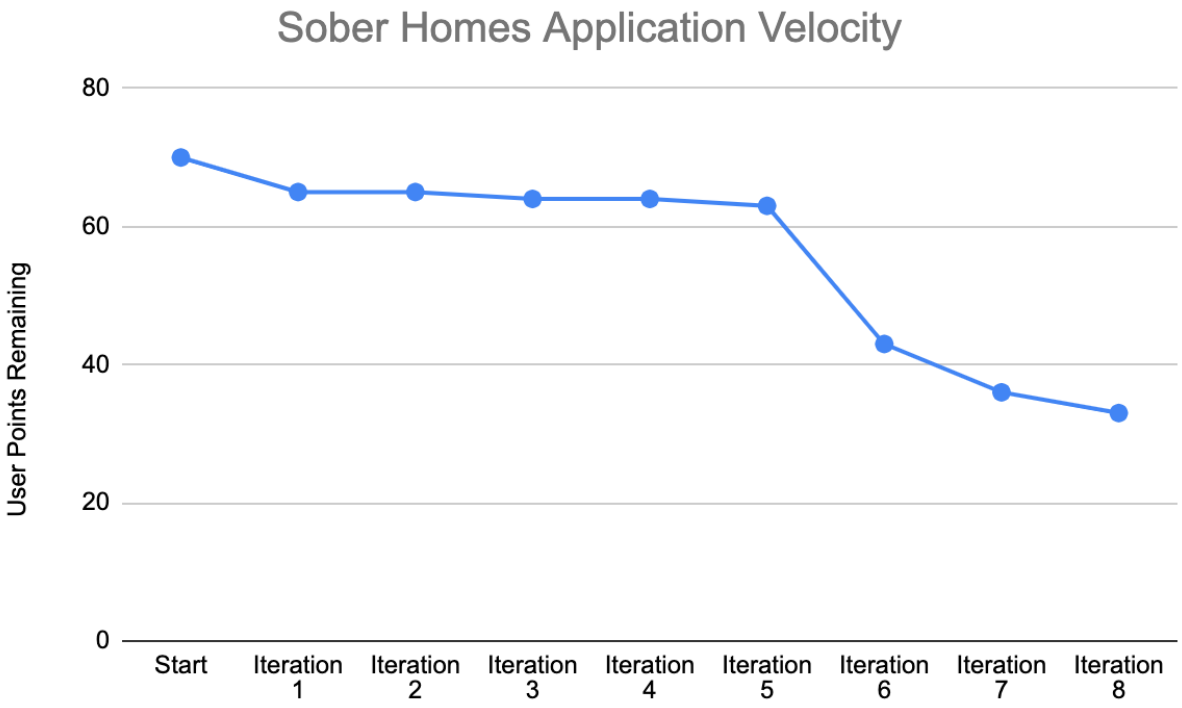

Figure 20: Iteration 8 Velocity Chart

During this iteration we completed the survey feature and began working on some administrative features for the application. In order to complete the survey, feature we needed to prevent users from taking surveys multiple times. Doing this required adding an additional table to the database that stored the AccountID of the user who took the survey and the SurveyID of the

survey that was taken. Then when displaying the available surveys to be taken by a user a subquery is used to only load surveys that have not yet been taken by the user.

Next we began working on a system for admitting residents into a sober home. This process works by a resident entering a four-digit code associated with the home they are trying to join then the owner of that home must approve that that user should be allowed to join the home. The four-digit code is a unique code tied to each house and would ideally be created by MASH and known by the owner of that home. In order to make this system work we needed to create a table in the database storing the IDs of accounts trying to join a home along with the ID of the home they are trying to join. We also modified the account table to include a verified boolean that was needed for this process.

During this iteration we were able to implement the code that allowed residents to apply to join a house, code that allowed sober homeowners to see a list of residents applying to their house, and code that allowed homeowners to accept or reject residents on the server side. However, we were not able to create the application side code for these processes, this was to be completed in the next iteration

Iteration 9

Planned to complete:

- As a HOUSE OWNER I want to [approve new residents] so that I may [regulate who joins the application]. Score: 3
- As a RESIDENT I want to [enter a house code] so that I may [have access to my house]. Score: 1

Completed:

- As a HOUSE OWNER I want to [approve new residents] so that I may [regulate who joins the application]. Score: 3

- As a RESIDENT I want to [enter a house code] so that I may [have access to my house].

  Score: 1

12 total user stories completed out of 40 total user stories

41 user story points completed out of 70 total user story points

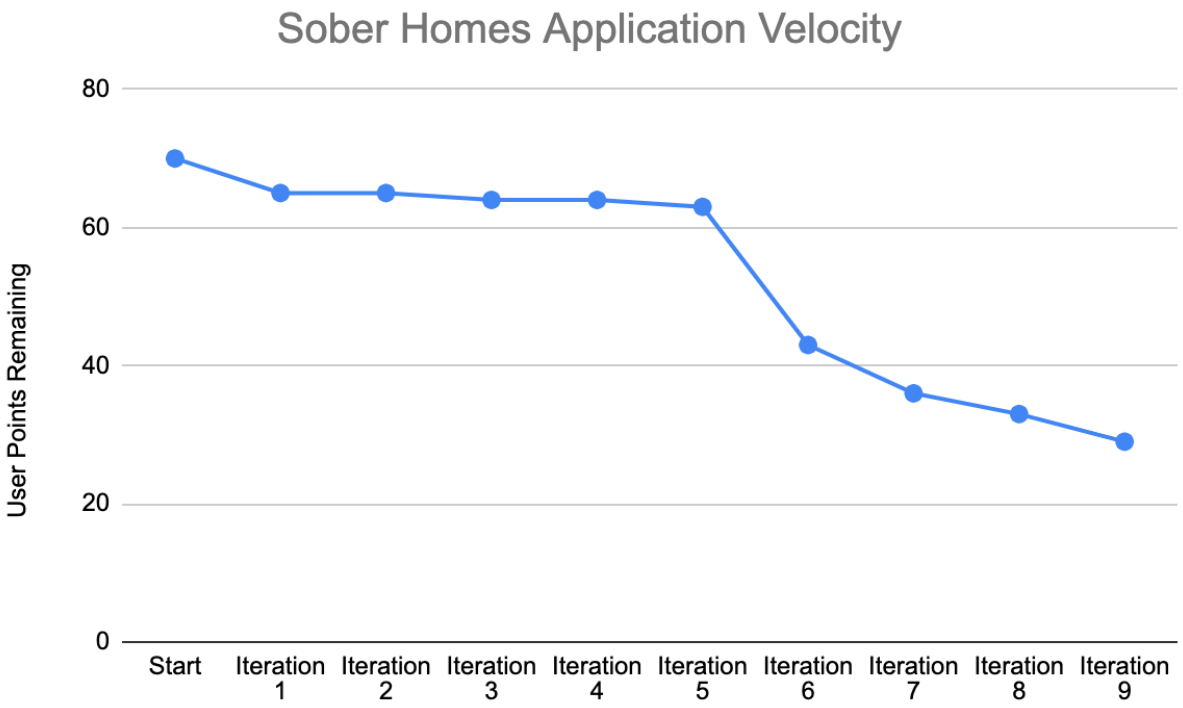

Figure 21: Iteration 9 Velocity Chart

During this iteration we implemented the application side code that allowed residents to join a sober home. For this application we decided that a resident account must be part of a sober home in order to use the features. Upon creating a new resident account, the user is immediately prompted to enter a four-digit house code that is given to them by their sober homeowner. Once this code is submitted their account is put in the database alongside the house, they are trying to join so that they are able to be approved. They are then shown a screen that displays a message telling them their account must be verified before they can enter the application. If they try to log in again before their account is verified, they are directed to the same screen.

We also created an activity that allows sober homeowners to accept or reject applications to their sober home. When the activity is started all applications to the house of that owner are loaded and displayed in a recycler view. Each application can be accepted or rejected, both actions will remove that application from the list, but only accepting will give the applicant access to their account. Due to the way the database is accessed, the owner will not be shown any applications that are made while they are on the accepting screen. In order to view these new applications, the owner must first go to the home screen then return to the accepting screen. Once an account is accepted, the resident associated with that account is then able to log in to their account and access the features of the application.

Iteration 10

Planned to complete:

- As a USER I want to [keep my account private] so that I may [prevent others from viewing sensitive information]. Score: 1

Completed:

- As a USER I want to [keep my account private] so that I may [prevent others from viewing sensitive information]. Score: 1

13 total user stories completed out of 40 total user stories

42 user story points completed out of 70 total user story points

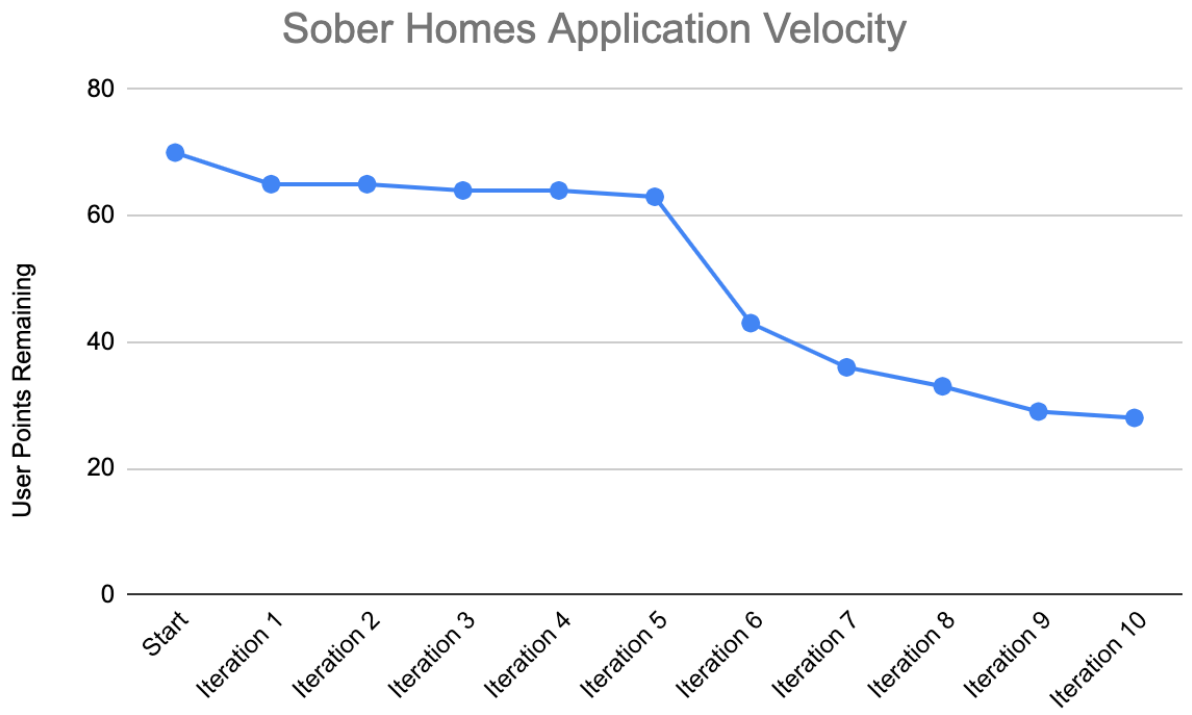

Figure 22: Iteration 10 Velocity Chart

During this iteration we put the finishing touches on the application. This included sending passwords in the encrypted body of the HTTP messages instead of the unencrypted headers to provide privacy. We also added a logout button to each of the home screens and a home button to each of the appropriate activities. This allowed for easier navigation throughout the application without using the Android back button. Finally, because we did not intend to add any additional features, we finalized the user interface to better match the application we ended up creating.

7.2 Software Security Implementation

After speaking with our sponsors, we decided we would create tiered accounts to create different privileges for different types of users. Since we have different tiers, we needed to authenticate each of those to ensure that only verified users would be using the application. The MASH directors would have the highest privileges and would be able to add new sober homes and add/ verify homeowners as well. The homeowners can also verify that users are actually

residents of their sober home. Sober home residents will be able to input the house code given by the owner and then the owner will be able to verify the user once their account is created.

The team wanted to receive the actual house codes that MASH would be using, but we decided to start by creating our own codes, knowing we could later edit these codes.

7.3 Testing Strategy

In order to test our application, we mainly used JUnit testing and the Android Studio debugger to solve any issues we faced. While at times using the debugger became tedious, it also made us learn about how exactly how code was running and how to write a better application.

When we ran into issues with connecting to the server and database, we created unit tests to ensure that the logic behind our code was doing what we expected. These tests were useful when we finally began connecting to the server and database because we could see exactly what test and at what point it was failing. From there we set breakpoints in our code and used the debugger to step through our code and find the line we were failing at. We set breakpoints to see if we were reaching certain lines of code that we expected to hit.

8. Assessment

Through the work on our Major Qualifying Project, our team gained knowledge about true software development processes and how to overcome issues. At the beginning of development, the team had high hopes for what we would accomplish in the following iterations. Our first iteration included success with user stories and we immediately saw the progress we made. However, the following iterations came with many unexpected “roadblocks” for the team. Our first issue was beyond our control since we relied on a third party to set up the server we needed and this did not take place in a timely manner. Looking back, the team would have made a local database to work on while we were waiting for the server to be up and running. Our next major

hurdle was connecting our Android application to the Postgres database. We were able to get the application and server “talking” but unable to have the information stored into the database. After some time, we realized that it was not good practice to directed connect an Android application with any database not provided by Android. From there we looked into how last year’s team was able to get over this hurdle because we initially did not understand all of the “weird” things they were doing in their code. We finally came across something called Volley which is an HTTP client request library for Android that last year’s team used. We realized that we needed to use Volley, GSON, and HTTP in order to have our application fully connected to the database.

As a team with no Android experience before the start of our project, we were forced to learn how to make all of these moving parts work. We learned what the process of true software development looks like and all of the hurdles that come with it. Due to our small team size and inexperience with the Android system we were not able to get our application to a point where it is ready to be released. However, we were able to develop tools that will be useful to other teams hoping to continue work on this application

9. Future Work

While the team created a functional application for MASH and completed many goals during this project, there are still many opportunities to continue and improve this project.

The team recommends that future teams work to implement communication between users in the application. After data collection, the sponsor’s next highest priority is for sober homeowners, as well as residents within a home, to have the ability to communicate through the application.

The team believes it would be beneficial to the user to provide more features of social networking to the application to go along with the data collection aspect.

The team also suggests bringing the application to iOS. The team predicted that most of the users' demographics for our application would be Android users. However, the team received information from one of our surveys that there are iOS users as well.

Additionally, the team would recommend a redesign of the database structure that makes use of a smaller locally hosted database included in the app. This database would use a background process to synchronize itself with a global database similar to our database structure. This would allow the application to make database calls without needing to make a network request each time.

It would be beneficial for future teams to create a website for the mobile application. A website would be best for MASH Directors so they can easily complete admin functions.

The team made the application usable for future teams. We wanted our application to be something that could be built upon so future teams would not have to start from scratch.

10. Conclusion

Overall, the project was a beneficial learning experience for the team. The team gained experience in developing a mobile application in Android which came with some challenges the team faced. These challenges provided us with experience on how to expand our programming skills and learn to work within new frameworks. These problem-solving skills will continue to be beneficial when programming for any system, not just Android. Working on a two-person team led to challenges, but an advantage was that both of us were involved in all aspects of the application.

References

Adobe. (n.d.). What is Adobe XD? Retrieved from https://helpx.adobe.com/xd/how-to/what-is-xd.html

AltexSoft. (2019, October 15). Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch and others. Retrieved from https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/

Android. (n.d.). Kotlin and Android : Android Developers. Retrieved from https://developer.android.com/kotlin

Android. (n.d.). Create a Java class or type: Android Developers. Retrieved from https://developer.android.com/studio/write/create-java-class

Android. (n.d.). Android Developers. Retrieved from https://developer.android.com/

Berg, B. L. (2001). *Qualitative research methods for the social sciences*. Boston: Allyn and Bacon.

Futrell, R. T., & Shafer, D. F. (2002). *Quality software project management*. Upper Saddle River, NJ: Prentice Hall.

GitHub, Inc. (n.d.). Build software better, together. Retrieved from http://www.github.com/

Hard, A. D., Charbonneau, J. L., Wong, J., Savell, K. Q., & White, T. (n.d.). *Massachusetts Sober Housing Mobile Application*.

Lucid Software Inc. (n.d.). Online Diagram Software & Visual Solution. Retrieved from http://www.lucidchart.com/

Nielsen, J. (n.d.). Thinking Aloud: The #1 Usability Tool. Retrieved from https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/

Qualtrics. (2020, March 30). Qualtrics XM - Experience Management Software. Retrieved from https://www.qualtrics.com/

Rubin, K. S. (2013). Essential Scrum: a practical guide to the most popular agile process. Upper Saddle River, NJ: Addison-Wesley.

StatCounter. (n.d.). Mobile Operating System Market Share Worldwide. Retrieved from https://gs.statcounter.com/os-market-share/mobile/worldwide

Substance Abuse and Mental Health Services Administration. (2019a). Key substance use and mental health indicators in the United States: Results from the 2018 National Survey on Drug Use and Health (HHS Publication No. PEP19-5068, NSDUH Series H-54). Rockville, MD: Center for Behavioral Health Statistics and Quality, Substance Abuse and Mental Health Services Administration. Retrieved from https://www.samhsa.gov/data/.

Substance Abuse and Mental Health Services Administration. (2019b). Results from the 2018 National Survey on Drug Use and Health: Detailed tables. Rockville, MD: Center for Behavioral Health Statistics and Quality, Substance Abuse and Mental Health Services Administration. Retrieved from https://www.samhsa.gov/data/.

The Society for Community Research and Action—Community Psychology, Division 27 of the American Psychological Association, (2013), The Role of Recovery Residences in Promoting Long-Term Addiction Recovery. American Journal of Community Psychology, 52: 406-411. DOI:10.1007/s10464-013-9602-6.

Tutorials Point. (n.d.). SDLC - Waterfall Model. Retrieved from https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

What is AGILE? - What is SCRUM? - Agile FAQ's. (n.d.). Retrieved September 23, 2019, from https://www.cprime.com/resources/what-is-agile-what-is-scrum/.