ENCOURAGING DEVELOPMENT OF MOBILE APPLICATIONS AS A SERVICE TO THE COMMUNITY

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Vanessa Forney

November 2016

COMMITTEE MEMBERSHIP

TITLE:	Encouraging Development of Mobile Applications as a Service to the Community
AUTHOR:	Vanessa Forney
DATE SUBMITTED:	November 2016
COMMITTEE CHAIR:	Chris Lupo, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	John Bellardo, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	David Janzen, Ph.D. Professor of Computer Science

ABSTRACT

Encouraging Development of Mobile Applications as a Service to the Community

Vanessa Forney

The convenience of mobile applications combined with the efficiency and effectiveness provided by technology has contributed to an increased interest in mobile applications. Local groups and non-profit organizations often utilize outdated, manual processes and don't have the resources or time to look into improving these systems. For Cal Poly students and other members of the community, this means there is an opportunity to apply technical skills and school projects to address these inefficiencies. This work explores whether a better system can be developed to provide the functionality of the existing system and enhance the experience of users through technology, data tracking, and automation. Two apps demonstrate the application of technology to meet needs within the San Luis Obispo community:

Poly Rides is an iOS and Android application that improves ridesharing for Cal Poly students. The idea stemmed from the Cal Poly Rideshare Facebook page, where the posting format for a ride is inconsistent, making it difficult to find a ride match. The Poly Rides app provides an improved user interface for posting, searching, and coordinating rides with other students. Its success has been validated through the popularity of the app. There were 3734 installations and 7925 messages sent as of May 27, 2016.

Woods is an iOS application for iPhone, iPad, and Apple Watch that improves the care tracking process for dogs at Woods Humane Society. The previous technique involved updating a whiteboard to manage the information and care for the available dogs. The whiteboard had inconsistent and limited information, was often out of date, and did not have room to list all of the dogs. An app was created which automatically pulls the dogs from the shelter database, provides more detailed in-

formation and instructions, and is available for volunteers on their personal devices. The results revealed a much larger and more positive impact than originally anticipated; volunteers reported feeling more confident providing appropriate care to the dog (65.9%) and that they have more trust in the dogs' information (52.3%). Of staff and volunteers, 83.9% prefer the app over the whiteboard and 10.7% have no preference. Dog breed, size, age, and photo, details not available on the whiteboard, were reported to be "Very" or "Extremely" important by 56.8% of volunteers.

This thesis describes some important requirements for developing community service mobile applications, offers suggestions for facilitating the development of a high quality product, and lists some useful resources for iOS development. Both apps not only reach their goal of improving a manual process in the local community, but also have the potential to improve and impact other communities around the world.

ACKNOWLEDGMENTS

Thank you for everyone who has supported my journey in Computer Science, with special thanks to:

- Myra Lukens and Michael Wong for helping with the Poly Rides app.
- Carson Carroll and Vivian Fong for helping with the Woods app.
- Zoe Wood for sparking my love of Computer Science.
- John Bellardo for teaching me iOS and for guidance, kindness, and support.
- Chris Lupo for never giving up on me and allowing me to follow my passions.
- My dog, Scruff, for making me smile every day.
- My mom for endless encouragement and for being my rock.
- My sister, Christina Forney, because without her I would not have tried Computer Science at all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Computer scientists have a tremendous opportunity to influence and contribute to society through the effective use of technology. This is evident in the widespread use of mobile phones and web applications today, and from the impact of companies such as Google, Facebook, and Apple on every day lives. Most businesses have the resources to keep up with advances in technology, but local communities and non-profit organizations generally do not and may represent an under-served sector of the technological market. These organizations often lack the resources and expertise to change their current systems, which are generally outdated, inefficient, and require unnecessary manual processes. Additionally, the entire community has to be willing to migrate to the new system.

At Cal Poly, the "learn by doing" philosophy is a highly valued component of education. Students are encouraged to be innovative, to contribute to the community, and to apply their skills and expertise to improve people's lives. The main motivation behind this thesis is to investigate how mobile apps can be developed to improve an aspect of the community. The process of recognizing a need, and working with others to create solutions that not only solve a problem but also replace and improve manual and outdated systems with functional, highly effective alternatives using modern technology is described. The contribution includes two mobile applications which demonstrate the impact of technology on a local community: Poly Rides, an iOS and Android application, makes ridesharing easier for Cal Poly students, and Woods, an iOS application, manages care provided to dogs at Woods Humane Society.

1.2 Mobile Applications

The number of mobile applications is growing every year, and as of July 2015 it is estimated that there are more than 1.6 million apps in the Google Play Store and 1.5 million apps in the Apple App Store [43]. While having mobile apps was previously considered an added benefit, this technology is quickly becoming an expected service.

Gurtner and Reinhardt found that "human behavior focuses on practical issues like functionality and effectiveness" and so the number of apps not meeting this requirement "will decline over the next few years." The working environment may be one of the areas where the use of apps grows because they "might be able to add functionality and effectiveness" [38]. Gurtner and Reinhardt also found that the most important features for adoption of a mobile application include "convenience, perceived quality, enjoyment, perceived ease of use and perceived usefulness" [38]. In communities or business organizations, mobile applications have the potential to improve the current systems and increase effectiveness of its members.

1.3 Contribution

This thesis aims to discover whether mobile applications can be developed to improve the community and to provide a framework and recommendation for development. This is explored through the development of two mobile iOS apps. The projects were selected out of personal interest in communities where there was a manual process that needed improvement:

1. Poly Rides, a ridesharing application for iOS and Android that helps Cal Poly students find rides for weekend trips.

2. Woods, an organizational iPad, iPhone, Apple Watch app for Woods Humane

Society in San Luis Obispo, which helps the shelter staff manage the dogs in the shelter and provides care tracking for volunteers.

Background for development of mobile applications is discussed in Chapter 2, and requirements are discussed in Chapter 3. The first mobile app, Poly Rides, is discussed in Chapter 4, while the second, Woods, is discussed in Chapter 5. Finally, conclusions are discussed in Chapter 6.

Chapter 2

BACKGROUND

Mobile applications should be developed utilizing available techniques and resources that improve the development process. This chapter discusses Swift technologies, backends, platforms, and resources for developing mobile apps that are independent of a single implementation.

2.1 Swift

Swift is the language created by Apple which is used for developing iOS apps. It replaced Objective-C in June 2014 [2]. This section discusses implementation techniques and recommended practices for app development in Swift.

2.1.1 Constants

Using a file to contain various constants within structs can end up saving a lot of time because there is only one location for each value, instead of having to look up that value any time it is needed. Constants can be useful for consistently using the same colors, fonts, keys for information transfer from the watch to the phone, various strings, and more. Specifically with colors, it helps to have one location for the color instead of having to look up the RGB values for each use. By defining the color as a constant, the color could be accessed with `Color.Blue` if the `UIColor Blue` is defined within the struct `Color`.

2.1.2 Enumerations

Enumerations (enums) are extremely useful for keeping things organized and they are used to "declare types with finite sets of possible states and accompanying values." Terhechte describes the power of enums in Swift by saying "with nesting, methods, associated values, and pattern matching, however, enums can define any hierarchically organized data" [44]. Enums can be simple and also extremely powerful. They can represent values, be nested, or be embedded in other classes. Enums can be further customized through methods and properties. This allows for one location to relate the enum value to an image, color, string representation, or other field, therefore eliminating the need for if-statements throughout the code. More information about the use of enums can be found in Apple's Swift documentation on enums [3].

2.1.3 Arrays

With Swift, arrays are passed by value. This means that changes to an array passed between view controllers or to a function will not be reflected in the original array. Therefore, it is important to store arrays in objects to ensure consistency of the array between views. A common design technique is to define a model object and have an array of that model object in a table view controller. This design is problematic if the array is passed from the previous view controller or the `AppDelegate`. This is because the array will be a copy and so changes will not be reflected in the original array. The original array may be changed because of updates from the database, and changes to the original array will not be seen in the copy of the array. A better strategy is to encapsulate the array into a separate object, to ensure the changes to the array are seen everywhere in the program.

2.1.4 Protocols

In Swift, there are two main ways to communicate between a model and a view controller. The first is using `NSNotifications`, which allow view controllers to register to receive callbacks when the model posts a notification of a change. However, using `NSNotifications` is similar to using a `goto` in C, where the program can jump to any location, and this strategy is discouraged. The preferred strategy is to use protocols. A protocol defines the functions that must be implemented when a view controller conforms to that protocol. The model can have references to one or more protocols which call the defined functions as necessary. This way, the calls are directed at the specific view controllers, rather than being broadcast as a global notification that may be caught anywhere in the program. More information about the use of protocols can be found in Apple's Swift documentation on protocols [5].

2.1.5 Extensions

Extensions can be a convenient way to subdivide the code within a class. This is a design choice that keeps the code clean and makes it easier to identify functions related to a specific protocol. The use of extensions is demonstrated below. In the first example, the code for the view controllers is all contained within the same class.

```
class MyViewController: UITableViewController,
 UITableViewDelegate, UITableViewDatasource, MyProtocol,
 LibraryDelegate {
    // All the function implementations for each protocol.
}
```

Compare this to using extensions, which divide the code as follows:

```
class MyViewController: UITableViewController {
    // Functions for the class.
}
```

```
extension MyViewController: UITableViewDelegate {
    // Functions for UITableViewDelegate.
}

extension MyViewController: UITableViewDatasource {
    // Functions for UITableViewDatasource.
}

extension MyViewController: MyProtocol {
    // Functions for MyProtocol.
}

extension MyViewController: LibraryDelegate {
    // Functions for LibraryDelegate.
}
```

While this is a simple example, it is easy to see how the code for the view controllers can get extremely long and complicated with multiple functions for each protocol. This can be avoided with the use of extensions.

2.1.6 Xcode

Xcode is the environment for iOS app development. Often, when a developer is learning to use Xcode, issues are assumed to be programmer errors. However, there are several known bugs relating to the use of storyboards in Xcode that occasionally prevent the code from working as expected. A storyboard is a file that graphically lays out the user's path through an iOS app [1]. In a storyboard, when different custom fonts are specified for each device, the program resorts to using the default system font with no styling. Special consideration is needed to maintain custom layouts. For a small number of font size changes, adjust the font programmatically. For more complex changes, create a separate storyboard for each device that connects to the same view controller, and set the fonts within each storyboard.

Another bug that exists in Xcode is that stale values occasionally remain in sto-

ryboards after resizing images. When this happens, the XML code has to be opened directly and the component has to be located in the code in order to update the value. From there, the value has to be manually changed or the entire image needs to be removed and added again.

Xcode provides many useful features, including stack views. Stack views are used within storyboards to support layouts on different devices. They allow for several components to be laid out automatically within a variable width or height. In addition, the connection between the storyboard and the view controller is defined through an outlet specified in the view controller. Separate iPad and iPhone storyboards allow the user interface and layout to change without impacting the code in the view controller. This is especially useful when the iPad version has extra information not needed for the iPhone version. This suggests the use of optional outlets rather than explicitly unwrapped outlets. An example is that on the iPad version, there could be an extra label that does not relate to the iPhone version. When this label is set in the view controller it will continue through the optional label on the iPhone version, but will set the value of the label on the iPad version.

2.2 Tools

There are many libraries and other resources available for iOS development that may not be evident to a beginning developer. CocoaPods (Pods) simplifies library management. Pods allow for one file to manage all included libraries, which are installed in one command: `pod install` [13]. Several additional techniques for keeping code organized and decreasing development time are described in the following sections.

2.2.1 Libraries

Before adding new features, it is useful to search for existing libraries that may already provide the needed functionality. Additionally, looking at open source libraries can give insights and ideas for features that can be incorporated into the app. A useful compilation of library resources are found in Table 2.1.

Table 2.1: Library Resources

Name	Description
Open Source Libraries [7]	Lists 27 of some of the most useful and popular open source libraries to speed up development.
Awesome Swift [15]	Extensive list of Swift related guides and libraries broken down by category such as images, layout, forms, menus, and buttons.
Awesome watchOS [30]	Similar to Awesome Swift but contains libraries related to the watch.
DZNEmptyDataSet [55]	Provides a user friendly view when a table has no items.
GeoFire [24]	Uses Firebase to allow for storing and querying of a set of items based on their geographic location. Selectively loads data near certain locations to keep applications fast even with large datasets.

2.2.2 Resources

Table 2.2 describes resources for development, including Swiftlint, Synx, and VVDocumenter that can help maintain organized code. Watch specific resources for understanding communication and data transfer are highlighted in Table 2.3.

Table 2.2: Swift Resources

Name	Description
Swiflint [36]	Script that runs every time Xcode builds, and verifies that the code meets the standards as defined in the custom or default style guidelines.
Synx [49]	Script that moves Xcode represented directories into actual directories for code organization, because in Xcode the directory structure is only contained within the project itself.
VVDocumenter [53]	Autofills the function description, parameters, and return value as needed in order to facilitate writing documentation for each function.
iOS Good Practices [23]	Guidelines to make iOS development easier and more consistent.
Fastlane [18]	A series of scripts which include the ability to deliver screenshots to the App Store and renew push notification profiles.
Icons8 [26]	Contains thousands of free icons in any size and any color, specific to iOS9 and other styles.
NSDateFormatter [39]	Helps determine the appropriate date format string.
Instant Logo Search [27]	Database of thousands of SVG and PNG logos.
iOS Dev Weekly [50]	Weekly email that summarizes the latest iOS news, including best practices and Apple announcements.
Prepo [29]	Resizes images for a specific purpose, such as generating various app icon sizes or screenshots for export to the App Store.

Table 2.3: Watch Resources

Name	Description
WCSession [34]	Suggested practices for wrapping a WCSession into a singleton.
Communication [45]	Comparison of different types of watch communication.
Data Transfer [25]	Overview of watch communication and data synchronization.
Watch Connectivity [21]	Combination of the above three resources into best practices for transferring data between devices.

2.3 Platforms

Implementation on several platforms allows for the solution to reach more users. However, with a small team or a single developer there may not be the resources to maintain multiple systems. For this reason, the platforms should be determined based on the audience. For a single corporation, an iPad app might suffice, if there is no need to use the app outside of the facility. A mobile version provides the ability to add personalization and improves the portability of the solution, but might require multiple platform support and does not reach those without smart phones. A web version makes the system available to most people, but removes the convenience and mobility of an app. The platforms developed depend on the audience and how the app will be used in the community.

2.4 Backends

At the time of implementation, there were two options considered for a backend: Parse and Firebase. Both are easy to setup and use with minimal overhead, however, Parse is a deprecated solution and will no longer be supported after January 2017.

2.4.1 Parse

Parse is a relational database similar to SQL that was created by Facebook. Parse handles the backend data store such that push notifications and user management can be easily handled through various login sources [37]. Facebook announced that Parse hosting will be discontinued starting January 2017, and all existing applications will be required to migrate to a different data store, or to move the data to a separate database, such as MongoDB [28].

2.4.2 Firebase

Firebase is a free, easy to use backend that stores data in a JSON format. It is available for iOS, Android, and web applications. Push notifications are supported as of May 2016. Data is accessed through a query to a Firebase URL, which returns all the data at that key and any data below. For this reason, Firebase documentation suggests avoiding nested data and using keys to limit the data returned. As a result, data has to be updated or deleted in two or more locations: where it is defined and anywhere that links to the data [20]. For example, a list of messages and the related data could be stored directly under a user. A better alternative is to store the list of message keys under a user, and the list of messages accessed by key at a separate location.

Firebase provides the ability to continuously monitor changes to the data or to get an immediate one time snapshot. To get a one time snapshot of the data, use the `observeSingleEventOfType` function. To continuously monitor the data, use the `observeEventType` function. One of the following four parameters is passed to these functions which invokes a callback:

- `ChildAdded`: Callback occurs when a child node is added at the URL. Returns

only the added child.

- `ChildRemoved`: Callback occurs when a child node is removed at the URL. Returns only the removed child.
- `ChildChanged`: Callback occurs when any of the fields are changed. Returns only the affected child.
- `Value`: Callback occurs when anything changes at the URL. Returns all of the data including the affected child.

In comparison to Firebase, the Parse user interface provides a much cleaner overview of the data and allows for easier searching and changing of database fields. With Firebase, loading the data on the web interface can be slow since all objects are nested below the initial key and there is no option to search for a specific piece of data.

Chapter 3

REQUIREMENTS

This chapter describes the design methodology used for the mobile applications presented in this thesis. These include needs assessment, defining the solution, platform selection, user experience, moving to production, and maintenance considerations. While these methodologies are presented in the context of this thesis, they are common software engineering practice, and are appropriate for any software development.

3.1 Identifying a Need

In order to solve a problem within a program, organization, or community, a need should be identified and thoroughly understood. This can be done by experiencing the problems firsthand or by speaking with the group about what problems they are facing. Firsthand experience is the most powerful; observations and experience with the current, often manual process, allows for a complete understanding of the problem and can provide insights on how to develop a better system.

Speaking to others is necessary to validate that the problem experienced personally or by a few members of the community is a problem for the majority and warrants development of a new system. This can be done informally, as in the case of the selected apps, or more formally and systematically through a needs assessment, which is a "set of procedures that are used to determine needs, examine their nature and causes, and set priorities for future action." Needs assessments are conducted to help "identify and select the right job before doing the job right" [35].

The needs for the apps presented in this thesis were first recognized through personal experience. For Poly Rides, it was difficult to search for rides using the

Facebook page, which was the prior technique for finding ridesharing partners. In John Bellardo's iOS class, project selection was open-ended, and developing Poly Rides with Myra Lukens was a way to help the community and learn app development.

Lubomir Stanchev's Knowledge Management class required development of a recommendation system that could be applied to any topic. With Vivian Fong and Carson Carroll, it was originally decided an app to match homeless dogs to potential adopters would be developed. The project changed into a care recommendation project after personal experience of having difficulty selecting a dog to care for on the whiteboard while volunteering at Woods Humane Society. The Woods app turned into much more than a simple recommendation system, but also demonstrated working with a larger organization to meet their needs, encouraging use of the system, and providing maintenance and improvements as the community adapted to the new system. The implementation for both apps began as class projects, which then were expanded and fully developed into production ready functional apps that serve their respective communities.

3.2 Defining a Solution

Reaching out to the community that will be affected will ensure their opinions and desires are addressed by the solution. The current system represents the organization's best effort at a solution, and it works, even if the process may be inefficient. The new solution may save time and energy, yet the community may not be willing to adopt the new solution because of familiarity with the existing system. It is up to the developer to prioritize the needs in a systematic way, and to differentiate between user suggestions and the important requirements for a new system.

For Poly Rides, this meant talking to other students and sending out a survey for requested features. The needs were identified as:

1. Passengers need to be able to easily search for and find posted rides.
2. Users need to be able to trust the app.
3. Drivers need to be able to post, track, and manage their rides.

Searching by date and location was determined to be the highest priority due to the inconsistencies of the posts on the Facebook page.

For Woods, this meant talking to the other volunteers and meeting with the volunteer coordinator. The needs were identified as:

1. Volunteers need a way to easily select a dog to care for.
2. Volunteers need a consistent way of knowing which dogs could have which care.
3. Staff need a way to easily maintain the list of dogs.

After discussion with the volunteer coordinator it was realized staff were not going to update the app, and it was determined the dogs had to be automatically updated to ensure adoption of the app. An elegant solution not only provides the functionality of the existing system, but also enhances the experience for the users. There is a reason the existing system needs to be updated, and while the users are experts with the current system, the developer may have insights into simplifying the system and adding functionality only available with technology. The system is not meant to be a recreation of the current system in an app, but meant to improve the process through technology, data tracking, and automation.

While designing and implementing a solution, it is important to get preliminary feedback through beta testing or trial days. However, this feedback should be taken as suggestions and not requirements. While feedback from the community is important, it is even more important to remember there was a reason why the system needed updating in the first place.

3.3 Selecting a Platform

As discussed in Section 2.3, the platform should be selected based on the community using the app. For Poly Rides, mobile versions were the highest priority. The app was developed for iOS as part of the contribution of this thesis, and the Android version was developed by Michael Wong. For Poly Rides, an additional web version could be beneficial but due to resource constraints the focus was on mobile platforms.

For Woods, a self-contained community operating within a single location, an iPad app would have sufficed since there is no need to use the app outside the facility. However, the mobile version provided the ability to personalize the user experience and improved the portability of the solution. The iPhone eliminated the need for the iPad to be available and allowed the volunteers to enter the data while providing care to the dogs. The app also allowed volunteers to enter additional comments from home, and to follow the status of their favorite dogs which increases volunteer engagement and commitment. For Woods there was no need to consider a web version because the shelter has an iPad for everyone to use.

3.4 User Experience

The design should be simple, clean, and have a "consistent visual theme" in order to make the app easy to use [4]. This means using a consistent color scheme and paying attention to minor details such as using the same font size throughout the app in order to "be internally consistent" [4]. The app should utilize consistent button coloring and styling so users can easily tell the difference between a button and text. Apple describes this as using a "key color" which "highlights important state information and subtly indicates interactivity" [4]. The other important aspect is to keep the most important information on the main page so that users can find what they need

without digging too deep into the app. Overall, the look and feel should be friendly, easy to use, and aesthetically pleasing.

Apple recommends taking "advantage of the whole screen" which is demonstrated in the Woods iPhone app because information goes to the eight pixel border on each screen. Letting "translucent UI elements hint at the content behind them" is demonstrated in Woods with the care entry and comment forms showing the dog info page behind them [4]. For buttons in the navigation bar, the size was chosen to be 44 by 44 pixels so they can be "accurately be tapped with a finger" [6].

Apple says to "use plenty of negative space" which is shown in Poly Rides in the messaging tab. This can make the app look "more focused and efficient." Apple also says to "embrace borderless buttons" which is consistent with the use of green buttons in Poly Rides [4].

3.5 Moving to Production

In the final phase, attending to the minor details that may have been overlooked during the development process should be addressed to meet production standards. These details distinguish a good app from a great app, and could mean the difference between the community accepting use of the app or continuing with the current process. Addressing these minor details carefully and early on in the development process contributes to a successful release.

Before releasing Poly Rides, it was necessary to add in-app messaging, improve the user interface to meet Apple's standards, and add mutual friends for acceptance to the App Store. After releasing Poly Rides, it was important to work with the admins to advertise on the Cal Poly Facebook Rideshare page as well as to work with commuter services to advertise the app to students. Before releasing Woods, it was necessary to create an iPhone version, maintain a consistent user interface,

and add additional features identified during initial trial days. Many details had to be addressed before release, including using the same font and size throughout the app, adding placeholder images, using consistent spacing and colors, and keeping the user informed about current states and their actions. After releasing Woods, it was important to convince Woods Humane Society to commit to using the app.

3.6 Maintenance

Once an iOS app is in use, it must be continuously maintained to support the users who rely on the app. Newer versions must support previous versions and major changes need to be introduced carefully. The developer is responsible for keeping the app in a functional state, while providing updates and bug fixes, adjusting to the changing needs of the organization, and responding to requests for additional features. Occasionally, parts of the app may need to be completely rewritten in order to take advantage of new development techniques which improve speed and efficiency.

Poly Rides has been used since October 2015 and has required a few bug fixes and updates, while Woods has been used since March 2016 and has had several enhancements. Both apps have required minimal upkeep since their initial release, and because of their success have led to plans for implementing improved versions. The future plans for Poly Rides include releasing a new version and continuing to provide maintenance as necessary until another solution is available. The future plans for Woods include beginning development on an app that works for all shelters. Additional details relating to future plans are described for Poly Rides in Section 4.10 and for Woods in Section 5.10.

Chapter 4

POLY RIDES

4.1 Introduction

Ridesharing is a simple way to help the environment and decrease the number of cars on the road. Prior to the work of this thesis, Cal Poly did not have a convenient way for students to share rides. This section discusses the motivation, context, and contributions of Poly Rides, a ridesharing app developed for Cal Poly students, and demonstrates the process of identifying a need within the community.

4.1.1 Motivation

University students have relatively consistent schedules, and they often travel home during weekends and holidays. A mobile rideshare application enables students who have extra seats available in their car to fill those seats and offset the cost of travel, while making it easier for students without cars to travel home. Ridesharing is often much more convenient for the passenger in terms of money and time, and so both the driver and passenger benefit from sharing rides.

Several rideshare applications already exist, but no program was available specifically for Cal Poly students and most students are unaware or uninterested in exploring those alternatives. Creating trust between the users is a key component in a successful rideshare application, because choosing the right driver or passenger for a long distance car ride is important for a safe, comfortable and enjoyable ride. Students use various means of communication in order to share rides, from verbal communication between friends to Facebook groups that aim to connect them with other users. While sometimes successful, these paths do not provide an easy and effective way to

find others with whom to share rides. Additionally, a widely used app could organize the available rides and provide an improved searching mechanism.

The goal of Poly Rides is to provide an easier and more efficient way to share rides. By lowering the barriers to adoption of ridesharing, Poly Rides aims to encourage students to make the effort to rideshare. Ridesharing adds an extra step for a student trying to go home, but allows those without rides to find transportation and benefits the driver with shared fuel costs. Overall, this benefits the environment and decreases the number of cars on the road.

4.1.2 Context

The idea for Poly Rides stems from the Cal Poly Rideshare Facebook group, with over 13,000 members as of May 2016 [17]. This platform enables Cal Poly students to collaborate and find rides, but it takes a significant amount of time to find a ride match. Hundreds of rides are posted every day, making it difficult to find a relevant ride for the desired date and destination. There are inconsistencies due to the formatting for both the date and location, which makes it harder to search for rides. For the date, the driver may post "Jan. 1st," "January 1," "next Friday," "1/1," or many other date variations that inhibit passengers from being able to search for a ride on a specific date. For the location, a user may post the nearest big city, the general direction they are heading, or the general region, such as the "Bay Area" or "SF Bay." The challenge of finding a ride match influences students' decision to share rides.

The Facebook search process was inefficient and needed improvement to encourage ridesharing among students. To meet this need, the initial iOS application was developed in the iOS class during Winter quarter 2015 with the help of Myra Lukens. The initial version of Poly Rides allowed Cal Poly students to offer rides, search for

rides, and to connect by exchanging phone numbers. A new version was released in Fall 2015, which had in-app messaging and the ability to see Facebook mutual friends. The popularity of Poly Rides grew rapidly, and with limited advertising there were over 1,000 downloads in the first month. The interest in Poly Rides during the early stages demonstrates that students are looking for, and would benefit from, an improved way to share rides.

4.1.3 Contribution

While a few ridesharing applications already exist, there are barriers to widespread adoption of these applications, including motivation for installation of the app and the potential for an unpleasant driver or passenger. With a narrower scope of targeting college students at Cal Poly, Poly Rides demonstrates how a ridesharing app can be successful in the hands of like-minded individuals. Exploring the components that students deem necessary in a ridesharing application is essential in getting them to adopt the new technology. A user survey was conducted in order to determine Cal Poly student interest in ridesharing.

The contribution of this thesis is to provide an accessible, easy to use application that introduces a ride sorting algorithm that best matches passengers with drivers. This application is provided for iOS and Android, and allows students to utilize the empty seats in their cars while also being able to share the cost of gas. The Android application was developed by Michael Wong during Fall quarter 2015.

Background information about Poly Rides is described in Section 4.2, followed by Related Work in Section 4.3. Requirements are discussed in Section 4.4, with Design and Workflows in Section 4.5 and Section 4.6. Implementation is discussed in Section 4.7 and Testing and Validation are discussed in Section 4.8. Finally, Conclusions and Future Work are discussed in Section 4.9 and Section 4.10.

4.2 Background

Ridesharing is the act of accepting a ride as a passenger, or providing a ride to another person as a driver. This includes various types of ridesharing: commuting, taxi services, and long distance rides.

4.2.1 Commuting

Commuting is for short distance trips, generally within cities. Commuting is used for real-time ridesharing to go to school, work, or other nearby locations. The driver will make the trip whether or not they have a passenger.

4.2.2 Taxi Services

Taxi services refer to apps like Uber and Lyft, which provide monetary benefits for the driver. The ride exists for the convenience of the passenger, and the driver will not make the trip without a passenger. Taxi services are generally within the same city or nearby cities.

4.2.3 Long Distance

Long distance rides are meant for trips back home or to other cities around the state or nearby states. The drivers share rides to save on fuel costs, and passengers pay per seat. Often times, the rides are shared with friends of the driver. Long distance rides are the focus of the Poly Rides app.

4.3 Related Work

Many ridesharing applications exist today. After analyzing many rideshare apps, none were found to be available at Cal Poly and provide the level of trust and ease of use students require. Level of trust is increased through student email verification or Facebook mutual friends. This section presents an overview of the existing rideshare applications, followed by details of the following apps: Volt, Zimride, BlaBlaCar, Tripda, and Uber and Lyft.

4.3.1 Overview

A total of 77 existing rideshare applications were analyzed. The methods included searching for any apps related to ridesharing online, on the Apple App Store, or on the Google Play Store in the fall of 2015. The results show an increase in the number of rideshare applications released in recent years, seen in Figure 4.1. When analyzing rideshare apps, the focus was on finding apps suitable for Cal Poly students, which include Facebook mutual friends and student email verification. The Cal Poly Rideshare Facebook group has an approval process to ensure members have a relation to Cal Poly, and mutual friends increase the level of trust between users.

Out of the 77 surveyed apps, the majority were meant for commuting, and 21 were meant for long distance rides. Of these 21 apps, 23.8% provided student email verification and 33.3% provided Facebook login to enable viewing of mutual friends. Only two ridesharing apps, Volt and Zimride, provided student email verificaiton and Facebook mutual friends. However, Volt is only available outside of the U.S. and Zimride does not have a mobile application. More information about the Volt app is discussed in Section 4.3.2 and more infromation about Zimride is discussed in Section 4.3.4. While Zimride is available in the U.S. at select universities and

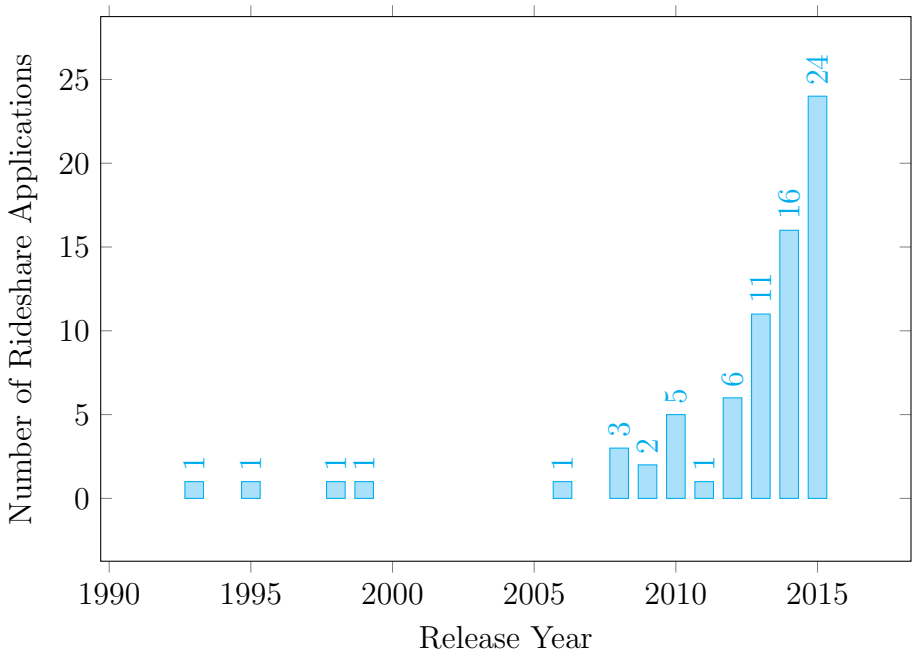

Figure 4.1: Rideshare Applications By Release Year

provides student email verification and Facebook mutual friends, it is only available as a web application. Table 4.1 shows an overview of the surveyed apps and indicates the percentage of each type of ridesharing app compared with the percent that have student email verification, Facebook mutual friends, or both.

Table 4.1: Overview of Rideshare Applications

Type	**Count**	**Student Email Verification**	**Facebook Mutual Friends**	**Both Features**
Taxi	13	7.7%	0%	0%
Commuting	43	9.3%	4.7%	2.3%
Long Distance	**21**	**23.8%**	**33.3%**	**4.8%**

An overview of the six apps discussed in the following sections are summarized in Table 4.2. No mobile application that is also available in the U.S. was found to

provide both student email verification and Facebook mutual friends.

Table 4.2: Details of Rideshare Applications

Name	Platforms	Release Year	Student Email Verification	Facebook Mutual Friends	Available in U.S.
Volt	iOS, Android	2014	Yes	Yes	No
BlaBlaCar	iOS, Android, Web	2006	No	Yes	No
Zimride	Web	2007	Yes	Yes	Yes
Tripda	iOS, Android, Web	2014	No	Yes	Yes

4.3.2 Volt

Volt is an inner-city ridesharing app in Turkey that was founded in 2015. Volt utilizes real-time ridesharing to help decrease traffic congestion by filling seats in cars of drivers who are already traveling to a destination. Volt is available on both iOS and Android.

Volt focuses on building trust between users by allowing members to join various communities. In order to join a community, users must verify an email address within that community or scan a community ID card. Communities can include university students, company employees, or residential compound residents. These communities

are predefined on Volt, and users can suggest a community if they don't see their community in the list. Volt requires Facebook login for security purposes so users can see mutual friends and be provided with an additional layer of trust. Drivers can review profiles of potential passengers before offering to give them a ride, and every mile driven earns the driver Volt Miles which can be exchanged for rewards. Drivers can also earn extra Volt miles for high ratings, paying within the app, and having passengers. Rewards can be small, including a coffee for 300 Volt Miles, or larger including 35,000 Volt Miles for an Apple Watch. Rewards are meant to encourage drivers to continue to post rides on the app even if they do not have any passengers. Screenshots of the Volt app are shown in Figure 4.2 [52].

Figure 4.2: Screenshots of the Volt iOS App

Volt connects people by using their current location and desired destination. The goal is to utilize rides that are already happening, and to prevent creating a taxi service where the driver profits. Volt requires passengers to pay a predetermined cost per kilometer based on the drivers expenses, including "gas, parking, tolls, car insurance, depreciation, and maintenance" [52]. Drivers are limited to earning 500

Turkish lira per month, which is equivalent to just over $150 [14]. Volt earns its revenue from a 20% commission on payments made in the app. According to the Volt website, people in Instanbul spend an average of two hours in traffic every day, with 80% of cars carrying only the driver. Volt reports that 85% of passenger seats in traffic are empty and people are spending 18% of their income on transportation [52].

Volt allows passengers to pay with cash or to pay within the app. Paying in the app is recommended, and is rewarded with Volt Miles. Volt's ten-year strategy for solving traffic problems in the most congested cities is described in Table 4.3 [52].

Table 4.3: Volt's Ten-Year Plan

Strategy	**Time Frame**	**Goal**
Short-term	3 years	"Build a reliable platform that connects passengers-drivers going in the same direction at the same time" [52].
Mid-term	3-7 years	"Connect drivers-drivers going in the same direction, thus creating a reliable mechanism to take cars off the road" [52].
Long-term	8+ years	"As the number of cars on the road declines, fill the gap with autonomous driverless cars, hence the name Volt" [52].

The Volt app shows ratings from past rides, Volt interviews, verification of a Turkish ID, and the number of rides completed. User profiles include communities the person is a member of, current job, past or current University, and current address to create additional layers of trust. When Volt interviews a driver, they inspect the car and verify the person's driver's license, Turkish ID, insurance policy, car permit, and license plate number.

Volt tries to solve the problem of congestion by connecting individuals who need

a ride with those who are already traveling. Volt provides many ways of establishing trust between users, and makes ridesharing more enticing by integrating with Facebook, allowing members to join communities, and providing drivers with rewards for participating.

4.3.3 BlaBlaCar

BlaBlaCar is a successful ridesharing application in Europe that was founded in 2006 [8]. They have over 20 million users in 19 countries, and have over 15 million mobile downloads. BlaBlaCar is available as a web application, and on both iOS and Android.

BlaBlaCar is the most popular ridesharing app in the world. The founders identified what they describe as the foundation for creating trust in an online community. They believe this foundation can be expanded not only through rideshare apps, but also to other aspects of the sharing economy. They call the six pillars of online trust the D.R.E.A.M.S. framework, which they developed based on research and feedback from their users. Each pillar of the D.R.E.A.M.S. framework is described in Table 4.4.

The D.R.E.A.M.S. framework started with a survey of 631 BlaBlaCar participants to determine the level of trust in familiar people compared to strangers. The results are summarized in Table 4.5, where zero indicates "do not trust" and five indicates "trust a lot" [8].

For members of an online community, providing a complete profile of an individual (including ratings, verified number, and a photo), increases the level of trust to 4.23. This level is almost equivalent to the level of trust given to a family member, 4.68, or a friend, 4.71. The results are summarized in Table 4.6 [12].

BlaBlaCar utilizes Facebook integration and allows users to see information about

Table 4.4: BlaBlaCar D.R.E.A.M.S Framework

Letter	Acronym	Definition
D	Declared	Declared information is the basis of a users profile. This is information a user volunteers about themselves to the community, and includes information such as name, age, description, or other preferences.
R	Rated	Ratings create inter-personal trust in a community by building reputations, since users often trust information from a third party.
E	Engaged	Financial commitment to a transaction through a pre-payment service provides users a sense of security that both parties will remain committed to participating in the service.
A	Activity-based	Active users within the community increases trust in a smooth transaction. For example, seeing the last time a user was active or a user's response rate can encourage other users to complete the transaction.
M	Moderated	Information verified by a third-party, such as contact information or bank details, ensures that the information meets a minimum level of goodwill and authenticity.
S	Social	Social networks hold users accountable for maintaining positive interactions on sharing services.

Table 4.5: BlaBlaCar's Degree of Trust

Individual	Degree of Trust
Friend	4.71
Family member	4.68
Neighbor	3.33
Stranger in the street	2.15
Stranger online	1.92

Table 4.6: BlaBlaCar's Degree of Trust Online

Individual	Degree of Trust
With a complete profile	4.23
With only positive ratings	3.39
With only a verified number	3.20
With only a photo	2.52
With no info - Stranger online	1.92

other users, including ratings, age, photo, and car make and model. They provide a full refund to the passenger if the driver does not show up. Drivers can specify whether they permit smoking or pets, and passengers have to pay up front for the ride. The name BlaBlaCar comes from users being able to specify their chattiness level, which can be "Bla," "BlaBla," or "BlaBlaBla." BlaBlaCar shows users' level of experience, which is determined based on verification of email and phone number, profile completion, number of positive ratings, and duration of use. For monetization, BlaBlaCar takes between 10 to 15% of each trip it facilitates. BlaBlaCar "ensures that co-traveller costs help offset real costs incurred by car owners and that car owners do not make a profit," and so there are no effects to the driver's insurance [8]. Screenshots

of the BlaBlaCar iOS application appear in Figure 4.3 [8]. While successful in Europe, BlaBlaCar is not available in the United States and doesn't allow users to join school communities.

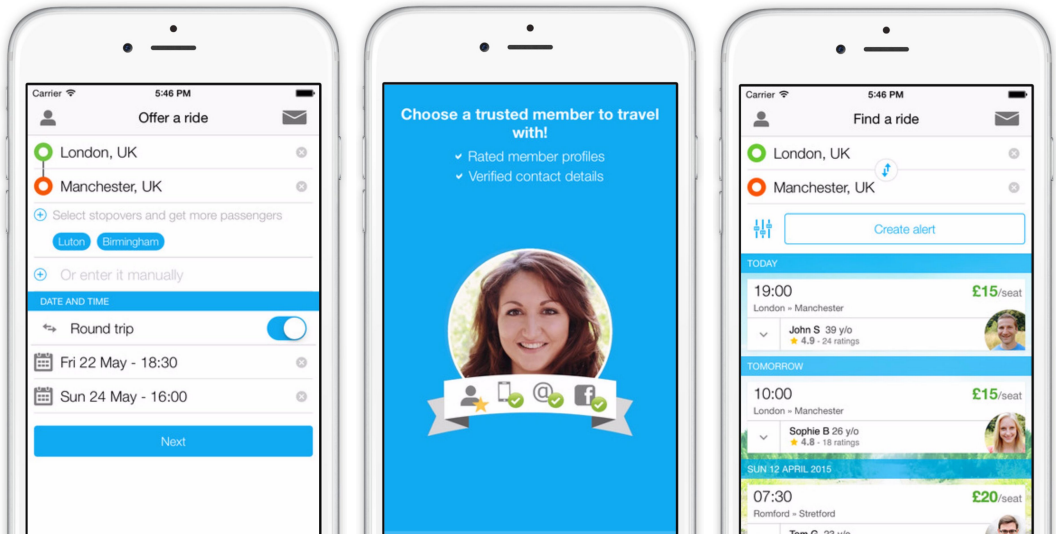

Figure 4.3: Screenshots of the BlaBlaCar iOS App

4.3.4 Zimride

Zimride is a rideshare platform founded in 2007. It is available at over 130 universities within the United States [54]. Zimride focuses on marketing to schools and requires school email verification for use. While Zimride can be used on a mobile browser, the application is optimized for use on a computer.

Zimride handles the payments for the rides through PayPal. Payments are distributed to drivers one day after the ride. This benefits the passengers since they receive a full refund if the ride isn't complete. When a passenger books a ride, the driver can review the passenger's profile before accepting or rejecting them. The benefits of using Zimride include posting recurring rides, showing an "About Me" page for the driver and the passenger, and linking information to Facebook. Ratings are available for both passengers and drivers. Zimride costs universities about $10,000

per year for their services, and is not offered at Cal Poly.

4.3.5 Tripda

Tripda was a ridesharing platform for the web, iOS, and Android that was founded in 2014 [46]. Tripda had over 220,000 registered users and serviced areas in North America, Latin America, and Asia. Since beginning this research, Tripda has shut down because "operating costs became too high" [46].

Tripda allowed passengers to view ratings, a photo, and Facebook friends of their potential driver. Drivers were able to accept or reject requests from passengers, and passengers had the opportunity to select a ladies-only option for passengers who only wished to travel with other women. For monetization, Tripda received funding from investors and had planned to take a percentage of the cost of each ride. Tripda focused on sharing rides with the general public and not within specific communities. Tripda did not provide authentication by university, and so users were unable to verify whether other users were students.

4.3.6 Uber and Lyft

Uber, founded in 2009, and Lyft, founded in 2012, are both applications available on Android and iOS. Lyft was founded by the people who started Zimride because they discovered there was more money in the taxi services aspect of ridesharing. Both apps offer the public the opportunity to make money by using their car to provide taxi services [32, 47]. Uber and Lyft generate income by taking a share of each ride, and drivers get paid for rides they complete. Since drivers are earning income through the app, Uber and Lyft have gotten into several legal battles. Both apps provide passenger and driver ratings, and drivers can select who they take as passengers.

4.4 Requirements

The goal for Poly Rides is to provide an easy to use application that students can use on the go to coordinate rides. As discussed, the first step was to get community input to understand the problem. This meant talking to other students about their experiences with the Cal Poly Rideshare Facebook page, as well as personal experience of using the Facebook page to find rides.

There were many possible features that could have been implemented, but the goal was to keep ridesharing simple and make the app easy to use. When defining a solution, the following features were selected as the most important in a university rideshare application.

- REQ-1: Passengers should be able to search by date and location.
- REQ-2: Users should not have to share personal information.
- REQ-3: Users should have to verify their accounts on Facebook.
- REQ-4: Drivers should be able to submit and manage their rides.

By providing the ability to search by date and geographic location (REQ-1), passengers are able to search for rides that meet their criteria instead of needlessly searching through the Facebook page. Through a messaging system, users will not have to share personal contact information (REQ-2). Since users have to login through Facebook, their accounts are verified by Facebook (REQ-3). Finally, drivers can submit and manage rides on the app and be able to remove them when they fill up (REQ-4).

The audience for Poly Rides is generally mobile savvy students, and therefore the iOS and Android platforms were chosen for the application.

4.5 Design

The Poly Rides app was designed to meet the requirements specified in Section 4.4, requiring users to log in through Facebook and having in-app messaging so users don't have to share personal information. Drivers can submit and manage rides, and passengers can search through the available rides by specifying a date and location. This section defines some important terms and discusses the user experience decisions.

4.5.1 Definitions

A few terms used in Poly Rides are listed in Table 4.7.

Table 4.7: Definitions for Poly Rides

Term	Definition
Message	A single message sent from one user to another.
Conversation	A series of messages between two users. There is one conversation for each pair of users, even if they share multiple rides.
Installation	An installation is a current install of Poly Rides app on either an iPhone or an Android device.

4.5.2 User Experience

Consistent with the guidelines presented in Section 3.4, the design of Poly Rides is simple and straightforward. The color scheme is based on Cal Poly green for familiarity. The buttons are consistently green to differentiate them from text. Four tabs are present on every screen to allow the user to easily access the main functionality for: driver, passenger, my rides, and messages. The iOS version of the tabs are presented in Figure 4.4.

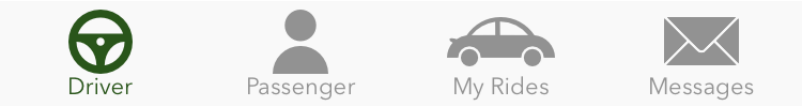

Figure 4.4: Screenshot of Tabs on the Poly Rides App

Screenshots of the Android version, developed by Michael Wong, are in Figure 4.5 and screenshots of the iOS app are in Figure 4.6. When new users install the app, it is important for them to have an immediately positive experience so they continue to use the app. This is encouraged through careful design choices and a simple user interface.

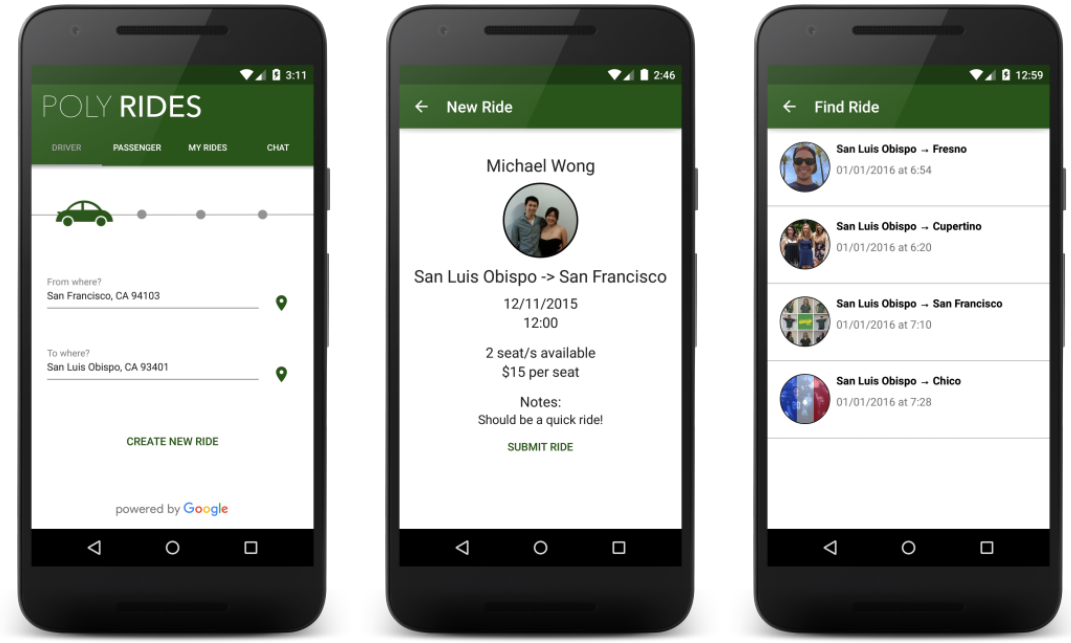

Figure 4.5: Screenshots of the Poly Rides Android App

4.6 Workflows

This section discusses adding, removing, searching for rides, and requesting a ride.

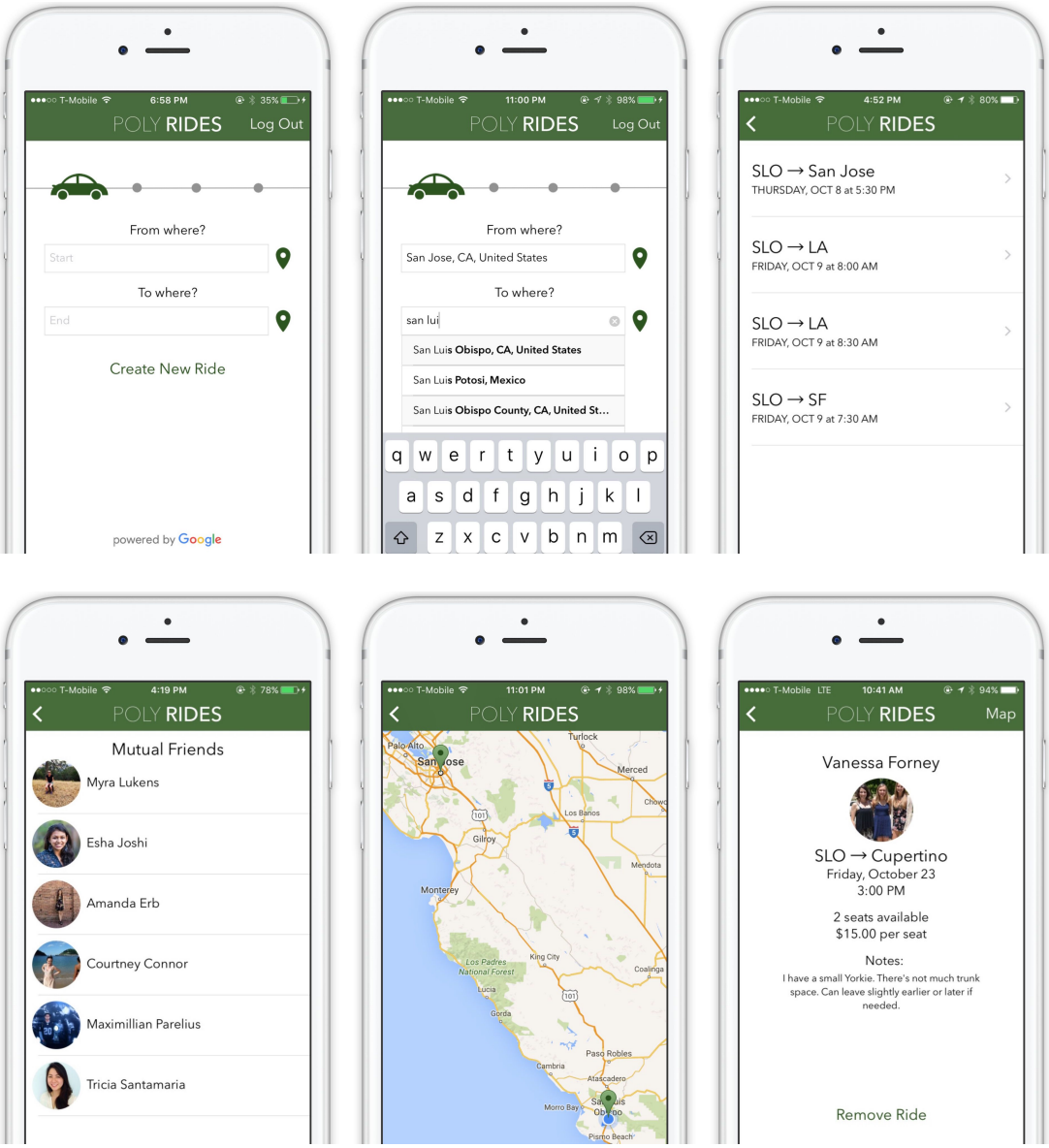

Figure 4.6: Screenshots of the Poly Rides iOS App

4.6.1 Adding a Ride

Adding a ride occurs from the "Driver" tab. Users answer a series of questions, including their departure city, destination city, date, cost per seat, and additional notes about the ride. After the information has been entered, a screen summarizing the ride is displayed before the driver submits the ride.

Adding a ride involves creating a `PFObject` from the Parse API which is set with the fields of a ride. The fields of a ride are detailed in Section 4.7.4. After setting the fields, the ride is saved with a call to `saveInBackgroundWithBlock`.

4.6.2 Removing a Ride

Drivers can delete their ride from the "My Rides" tab when they have enough passengers. To do this, they select the ride, view the ride details, and then remove the ride from the available rides. Only rides that depart in the future will show up on this tab.

A ride is removed in Parse by calling `getObjectInBackgroundWithId` on the `PFObject` with the specified ride ID to delete. Then, the object is deleted with `deleteInBackgroundWithBlock`.

4.6.3 Searching for a Ride

From the "Passenger" tab, users enter the start and end location along with a date. After tapping search, they are presented with a list of rides available within 24 hours of their desired departure date. From there, users can request a ride.

When searching, all rides within 24 hours of the requested date are returned. This means adding a filter to the query where `dateTime` of the ride falls within range of the requested date. The `NSDate` object has a `dateByAddingTimeInterval` function which allows for adding a positive or negative value of 24 hours in seconds. A check makes sure no rides in the past are returned, and the lower bound of the date is always greater than the current date. This same principle applies when users are viewing their own rides, which adds another parameter to make sure the `userId` of the ride is equal to the user's id. For ride searching, the opposite is true. During the search, a user does not see their own rides and so `userId` must not be equal to the

user's id. The ride search algorithm is discussed in Section 4.7.3.

4.6.4 Requesting a Ride

After selecting a suitable ride, the ride details page is displayed and the passenger can see the driver's full name, profile picture, and any Facebook mutual friends with the driver. When the passenger messages the driver, they are taken to the messaging tab where they can send a custom message to the driver. From there, the driver is alerted that they have a message and the users can continue to coordinate.

When requesting a ride, a conversation is created which represents the series of messages between two users. The group ID used is the concatenation of the two user's ID's ordered ascending. This ensures that no matter which user sends the message first, the two user's will always map to the same conversation. First, the conversation table is searched for the group ID. If the group ID does not exist, then it is initialized and created. Then, the user is taken to the messages screen.

4.7 Implementation

The model design, autocomplete, ride search algorithm, backend decisions, messaging, and Facebook integration implementation of Poly Rides are discussed in this section.

4.7.1 Model

An overview of the model is presented in the UML diagram in Figure 4.7. For simplicity, only the most relevant functions and properties are displayed.

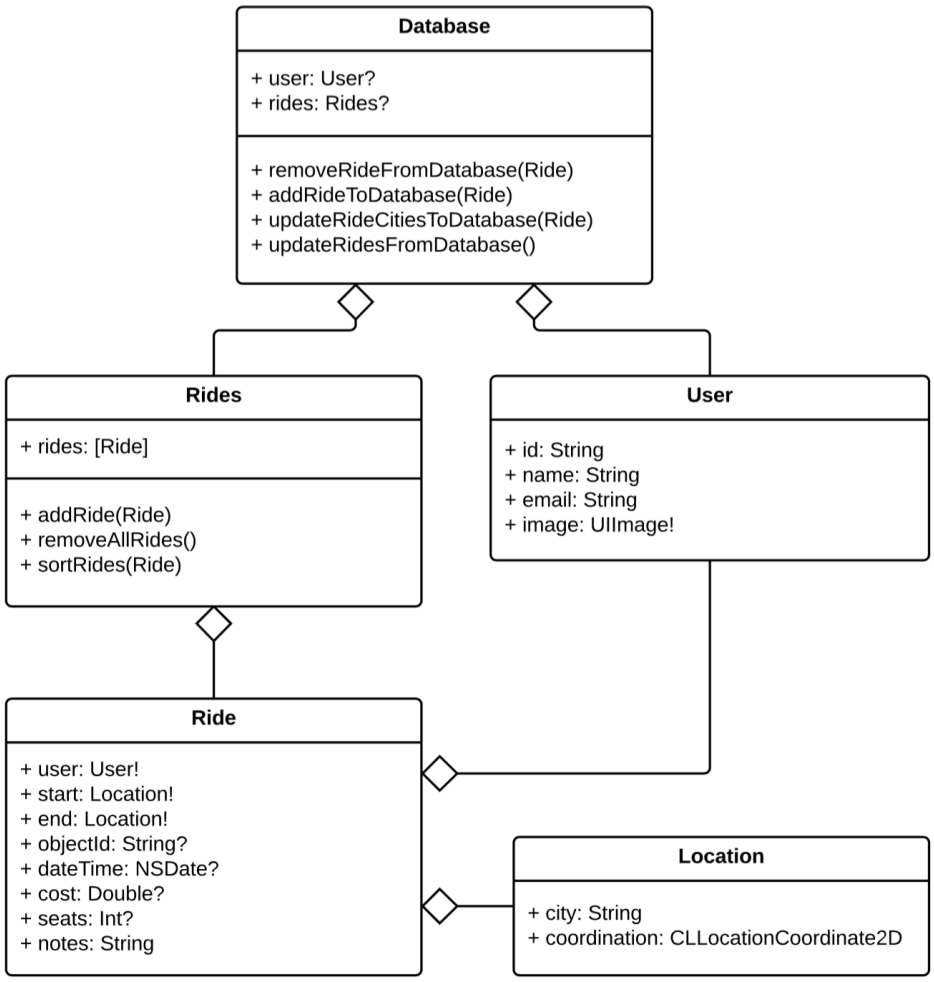

Figure 4.7: Poly Rides UML Diagram

4.7.2 Autocomplete

An autocomplete feature has been implemented in the search fields on the passenger and driver tabs. The implementation uses the `MVAutocompletePlaceSearchTextField` library [48]. The return place ID from the call is looked up to get the `GMSPlace`. The new version of Poly Rides uses Google Maps Autocomplete directly and displays the results in a custom table view. This was found to be easier because a list of `GMSAutocompletePrediction` is returned, which eliminates the need to look up every

place ID. Instead, when an autocomplete row is selected, one call to get the `GMSPlace` occurs. The implementation includes initializing a `GMSAutocompleteFetcher` and its delegate. The delegate functions are called as necessary, which includes a call to `didAutocompleteWithPredictions`.

4.7.3 Ride Search Algorithm

When a user searches for a ride, the ride search algorithm displays all the rides within 24 hours of the requested date. These rides are sorted by proximity to the desired route. Proximity is determined by the smallest geographic distance between the latitude and longitude coordinates of the searched ride and the rides in the database. The distance between the coordinates of the ride start and the passenger's ride start is added to the distance between the coordinates of the ride end and passenger's ride end. The rides with the shortest total distance appear first. The formula below compares two rides (`ride1` and `ride2`) to the ride searched by the passenger (`search`), for all the rides available within 24 hours of the searched ride (`rides`):

```
rides.sortInPlace {
   let distance1 = GMSGeometryDistance(ride1.start, search.start) +
                   GMSGeometryDistance(ride1.end, search.end)
   let distance2 = GMSGeometryDistance(ride2.start, search.start) +
                   GMSGeometryDistance(ride2.end, search.end)

   return distance1 < distance2
}
```

As seen in the UML diagram in Section 4.7.1, `start` and `end` are of type `Location`, and so to get the `CLLocationCoordinate2D` the `coordinate` field is accessed on the `Location`. For demonstration purposes, accessing the `coordinate` field for each `Location` was omitted above.

4.7.4 Backend

Parse was chosen as the backend for Poly Rides before the shutdown was announced. The reason for this decision was convenience and ease of use. Later during development, having the capability to add push notifications was beneficial. Parse worked well for Poly Rides, but because the free hosting will be discontinued, the new version of Poly Rides uses Firebase.

In Parse, the ride has the fields in Table 4.8. The focus was getting something to work with the initial version of Poly Rides, and so the user's information is stored within a ride. In the new version of Poly Rides, the user's information is stored in a separate location and the start and end locations are stored in a place ID. The start and end cities are still stored to make it easier to display each ride, rather than make a query to get the city from the `GMSPlace` each time.

Table 4.8: Poly Rides Parse Ride Fields

Field	Type	Description
`name`	String	Name of the user.
`userId`	String	ID of the user.
`startLat`	Number	Start latitude.
`startLong`	Number	Start longitude.
`endLat`	Number	End latitude.
`endLong`	Number	End longitude.
`startCity`	String	City where ride begins, corresponds to start coordinates.
`endCity`	String	City where ride ends, corresponds to end coordinates.
`seats`	String	Number of seats available.
`cost`	String	Cost per seat.
`notes`	String	Special notes for the ride.

4.7.5 Messaging

Messaging was implemented using JSQMessages, which is a library wrapper for messaging that mimics Apple's iMesssage. The Poly Rides messages view controller is a subclass of the `JSQMessagesViewController` class, which provides methods that can be overridden to determine the color of the bubble, avatar, timestamp, message, and other data [42]. The badge numbers on the app icon, over the messages tab, and on the unread symbol next to the message had to be configured so they were always in sync.

4.7.6 Facebook Integration

Facebook integration was essential to establish trust between users. Since the app requires users to login through Facebook, their name and profile picture are kept up to date.

4.8 Testing and Validation

This section discusses the survey provided to the Cal Poly community and the results and analysis of the survey.

4.8.1 Survey

To gain additional understanding of students' ridesharing needs, a survey was distributed to various members of the Cal Poly community. This survey focused on learning why students might be afraid of ridesharing and what features they preferred in a ridesharing application. The survey was provided through the Cal Poly class pages on Facebook as well as the Cal Poly Rideshare page. Overall, there were 190 responses. After the study was conducted, the results were evaluated in order

to understand what features were needed for Poly Rides 2.0, which is discussed in Section 4.10.3.

Since the study involved people from Cal Poly, the survey, Informed Consent Form, and Human Subjects Protocol Approval Form have been reviewed and approved by the Cal Poly Human Subjects Committee. Full documentation of the survey can be found in Appendix A.

4.8.2 Results

The majority of respondents were students (94.8%), which included first year students (38%), second year students (18.7%), third year students (16.6%), fourth year students (7.5%), fifth year students (7%), and master's students (7%). The rest included faculty or staff (0.5%) or other non-students (4.8%). Respondents indicated that they generally go home only for breaks (28.5%), breaks and 1-2 weekends (36.6%), and breaks and 3-5 weekends (17.2%). Of these, 42.5% indicated they use a rideshare to get home while 38.7% indicated they drive themselves.

The most reported regions that respondents were from included the SF Bay Area (38.7%), Los Angeles County (12.9%), the Central Valley (8.6%), Orange County (7.5%), San Diego (6.5%), or outside of California (11.3%). Of all respondents, 99% indicated they had a smart phone, 65.8% having iPhones and 33.2% having Android devices. Most students in the community are interested in ridesharing for saving the cost of gas (44.9%) or convenience (34.8%).

In terms of transportation, 41.2% of respondents would prefer to be a passenger, but 34.8% said they are interested in being a driver and 20.9% said they could be either a passenger or a driver. Willingness of users to share rides with various individuals is summarized in Table 4.9.

Of the respondents, 96.8% had used the Cal Poly Rideshare Facebook page while

Table 4.9: Willingness to Rideshare

Type	Stranger	Cal Poly Student (No Mutual Friends)	Cal Poly Student (Mutual Friends)
Driver	24.1%	80.6%	93.5%
Passenger	21.3%	79.2%	92.3%

only 29.1% had used Poly Rides. The reason for this is partially due to not having heard of Poly Rides (17.2%) or not installing the app (34.4%). Most people prefer using a mobile interface to find rides (48.9%), but 32.6% said they prefer a web interface. Regarding payment for the ride, 34.9% prefer to pay within the app while 59.7% prefer to handle payments outside of the app.

The highest ranked features that the respondents wish to see in a ridesharing app are verification of student email address, description of driver and car, response rate of drivers, and ratings of potential drivers. Ratings of potential passengers were ranked much lower than ratings of potential drivers. Payments inside the app and ladies-only rides were deemed less important.

Of those not interested in ridesharing, 77.8% of respondents reported that flexibility affected their lack of interest in ridesharing. Safety was the second highest concern, selected by 51.9% of respondents. Coordinating pick-up and drop-off times was the third highest concern, selected by 48.1%. However, when looking at the most important reason that prevents people from ridesharing, 28.6% of respondants said it was due to safety, 21.4% was due to coordinating departure and arrival times, 14.3% due to the lack of flexibility, and 14.3% due to no room in the car. Additionally, 10.7% said that it was due to the difficulty of finding a carpool partner.

4.8.3 Analysis

Many people within the community already use ridesharing to get home, which indicates that providing a more convenient way to rideshare could increase the number of users that participate. Even though a significant number of students reported preference for a web interface (32.6%), this may be due to the Facebook page and what they are used to. The process people are used to is not necessarily the best way, and showing the community the convenience and improved functionality of an app could help them transition and prefer the app.

The gain of 12.9% more students being willing to give a ride to a Cal Poly student with mutual friends compared to a student without mutual friends shows the importance of providing mutual friends in the app. The criteria for a passenger was slightly higher than that of a driver, which makes sense because passengers have to trust a driver in order to be willing to ride with them.

The regions reported indicated that most students interested in ridesharing are from the SF Bay Area or Los Angeles, which means that separating rides into these categories could be useful for those users. In terms of features, people reported the importance of a description of potential drivers and their car and the response rate of drivers and so these are important features in a ridesharing app for the community. By improving the rideshare experience, this can leverage those who are not interested in ridesharing due to the difficulty of finding a rideshare partner and increase the amount of trust users can have between each other.

4.9 Conclusions

This section explains how the app meets the requirements and further discusses the results of the app in the community.

4.9.1 Requirements

This section explains how the system meets the requirements defined in Section 4.4.

- REQ-1: *Passengers should be able to search by date and location.*

  When passengers search for rides with a start location, end location, and date they are presented with a list of available rides sorted by proximity to their desired route. These locations are converted into latitude and longitude coordinates and compared to the available rides. The resulting list of rides are the ones within 24 hours of the passenger's desired departure date.

- REQ-2: *Users should not have to share personal information.*

  In-app messaging allows for users to avoid sharing personal contact information. Users can communicate in the app complete with push notifications and badge icons.

- REQ-3: *Users should have to verify their accounts on Facebook.*

  Users must login using Facebook before using the app, to verify their account information including profile picture and name.

- REQ-4: *Drivers should be able to submit and manage their rides.*

  On the "My Rides" tab, drivers can submit and manage their rides. This includes viewing a list of the currently available rides as well as deleting rides when they fill up.

4.9.2 Discussion

Submission to the App Store caused an initial setback for Poly Rides, and the app was not accepted until the mutual friends feature was added as required by Apple. Using

Facebook for the purposes of user verification and name validation was inadequate. The overall process took six months because of an internship which prohibited work on side projects.

Once the app was accepted into the App Store, gaining users was the next step. In order for the app to be successful, it required having a significant number of users in order to have rides available at any given time. While Poly Rides makes ridesharing easier, it was made with no iOS experience and so has a lot of room for improvement. One aspect that helped spread the word about Poly Rides was creating fliers that were posted around Cal Poly's campus. The quarter page fliers can be seen in Figure 4.8. Users could easily resort back to the Facebook page, and so it was important to immediately advertise and continue to encourage users to post on the app because the app wouldn't be successful without the majority of the school using it.

Figure 4.8: Poly Rides Flier

Cooperation with the Cal Poly Rideshare Facebook page allows for the advertisement on that page and informed the rideshare community about the app. This meant that users who saw and used that page daily would see the logo for Poly Rides on the page. With push notifications, users can easily see when passengers or drivers message them. The growth of Poly Rides over time can be seen in Figure 4.9, which summarizes the number of installations every year, the number of messages, the num-

ber of conversations, and the number of rides. The growth in the number of rides are provided, but this is not necessarily a good measurement of the success since full rides are deleted from the database. This design choice was made early on and in the redesign of Poly Rides, the rides will be marked as full rather than be removed from the database. However, even with the rides being deleted the number of rides can be seen increasing over time. In terms of messages, the growth is increasing over time, with a more exponential growth since February. The number of users shows a steady incline since the start of the app, and as of May 20, 2016 there are 3655 installations, 7840 messages, 2350 conversations, and 687 rides.

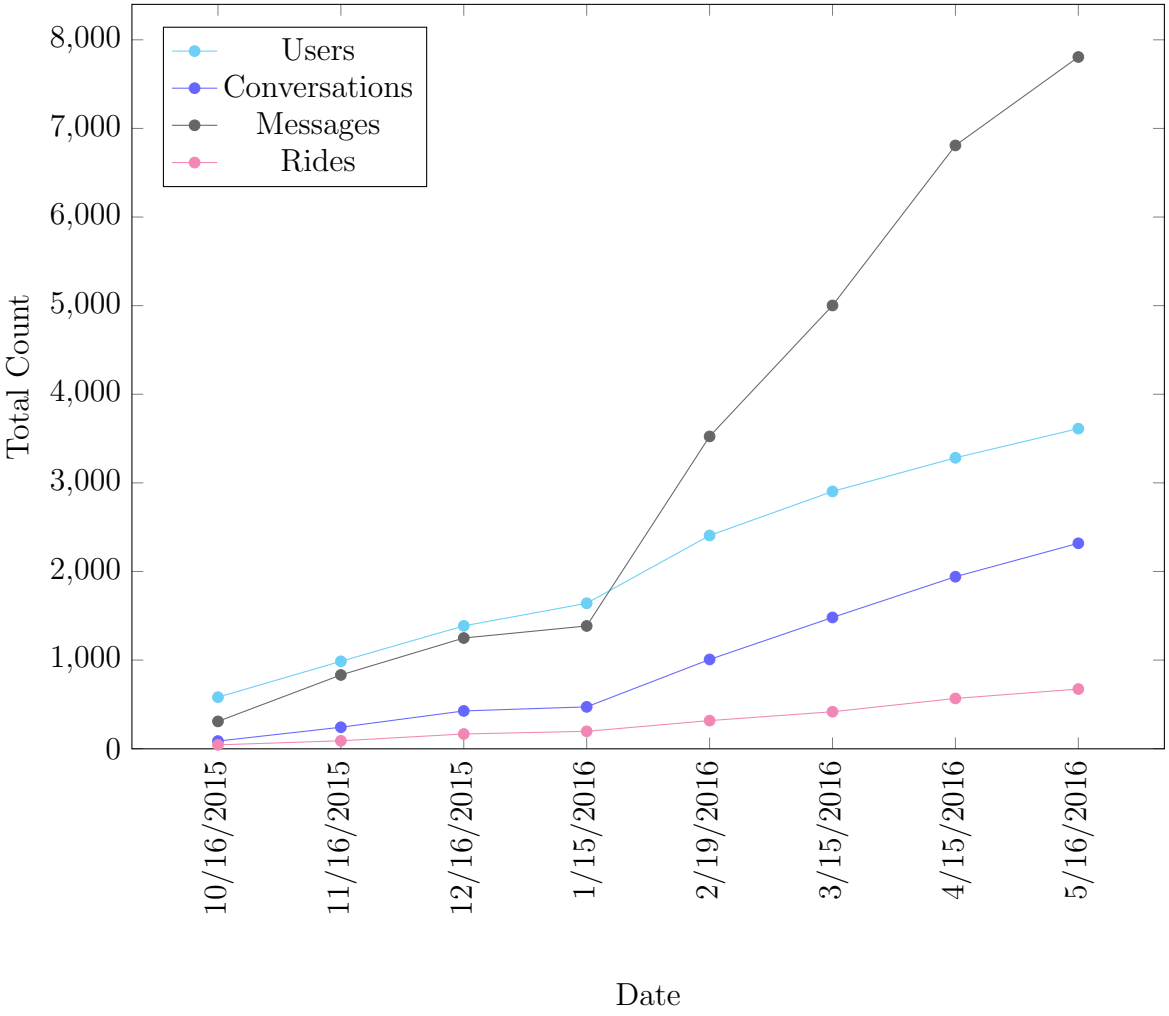

Figure 4.9: Growth of Poly Rides

Poly Rides has made an impact on the Cal Poly community, and demonstrates one example of how mobile apps can improve the community through automation of a manual process. Poly Rides' success led to plans for developing a new version with added features, as discussed in the next section.

4.10 Future Work

Poly Rides has the potential to continue expanding around Cal Poly and even spread to other universities with several improvements. As it is now, the current version of Poly Rides cannot handle a huge number of users and needs to be updated before January 2017 when the Parse shutdown occurs. Because a large number of users rely on Poly Rides, it is important to continue maintenance on the app.

4.10.1 Ratings

Ratings would be an improvement to the app, but could get complicated with one bad review ruining someones profile even if there was a valid emergency that caused them to miss their ride. For this reason, ratings have been prioritized lower than other features.

4.10.2 Verifications

Verifications could involve work places or school email addresses. While the current version of Poly Rides does not have any verifications, Poly Rides 2.0 allows for Cal Poly email verification using SendGrid [40]. More details of Poly Rides 2.0 are discussed in Section 4.10.3. The idea of a verification is to allow students to enter their Cal Poly email address and receive an email with a verification code that they can then enter on the app to confirm their attendance at that university.

4.10.3 Poly Rides 2.0

The new version of Poly Rides includes Cal Poly email verification and an improved ride search experience. The ride search experience is improved through a list of regions. These regions include several of the top areas in California that Cal Poly students are from: Central Coast, SF Bay, Los Angeles, Orange County, Sacramento, and San Diego. All other rides are grouped into the "Other" category. Within each region, users can see rides going to or leaving from that region. This screen will provide new users with a more friendly experience that can help them feel like Poly Rides is popular and used among other students. A few screenshots of the preliminary implementation can be seen in Figure 4.10.

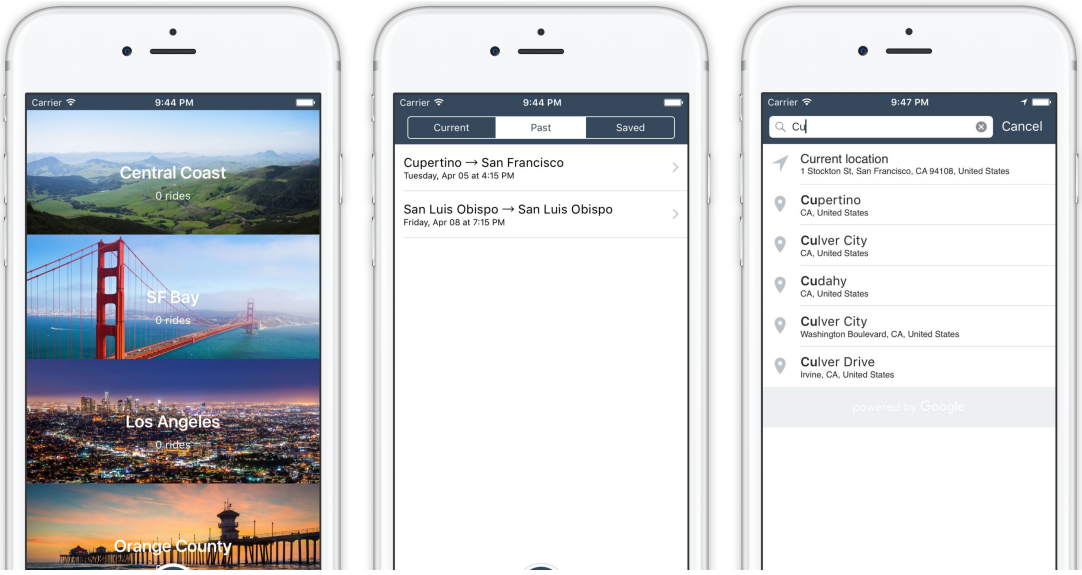

Figure 4.10: Screenshots of the Poly Rides 2.0 App

Although the designs and color schemes are not finalized, the general layout of the app has been determined. Besides Cal Poly email verification with SendGrid, the new version will allow grouping of passengers under a specific ride. This will allow drivers to organize their rides and see which passengers have inquired about which rides. Additionally, drivers will be able to accept or reject passengers into their rides. This

will allow drivers to more easily see who they have accepted into the ride, and also allow them to remove a request. This could be useful for drivers in case a passenger backs out, they could refer back to the list of pending passengers in order to find a replacement.

Finally, passengers will be able to save rides in case they don't want to immediately message the driver but they want to be able to refer back to the ride. Drivers will be able to manage their current rides and view their past rides, which could eventually work for leaving reviews.

Chapter 5

WOODS

5.1 Introduction

Woods Humane Society was founded in 1955 in San Luis Obispo County. Woods Humane Society is a nonprofit organization that is "supported by donations, grants, bequests, fundraising events and fees for services." Woods Humane Society works with neighboring San Luis Obispo County Animal Services in order to improve the welfare of all animals in the county. Woods Humane Society provides many opportunities for volunteers, including surgery support, customer service, special events, cat ambassador, and canine enrichment positions [41].

5.1.1 Motivation

One of the missions of Woods Humane Society is to "serve, protect and shelter homeless companion animals" and the motivation behind this thesis is to facilitate that mission [41]. Prior to the work of this thesis, volunteers and staff at Woods Humane Society used a whiteboard to monitor the care of the available dogs. The whiteboard required manual updating multiple times per day, which led to the whiteboard often being out of date.

Even for someone with extensive dog handling experience, the process of getting started as a volunteer is overwhelming. Inconsistent handling and instructions can be confusing and dangerous to the dogs, volunteers, and the public. For this reason, there are many guidelines that must be followed in order to provide proper care for the dogs. Making sure each dog is provided with the care and training it needs is extremely important in helping the dogs become more adoptable. Volunteers are expected to

adhere to the guidelines as well as provide individual care to each dog based on its requirements as defined on the whiteboard, shown in Figure 5.1. The whiteboard listed a subset of the available dogs and was used to record initials whenever a care was provided, and was used at Woods Humane Society before the app was introduced.

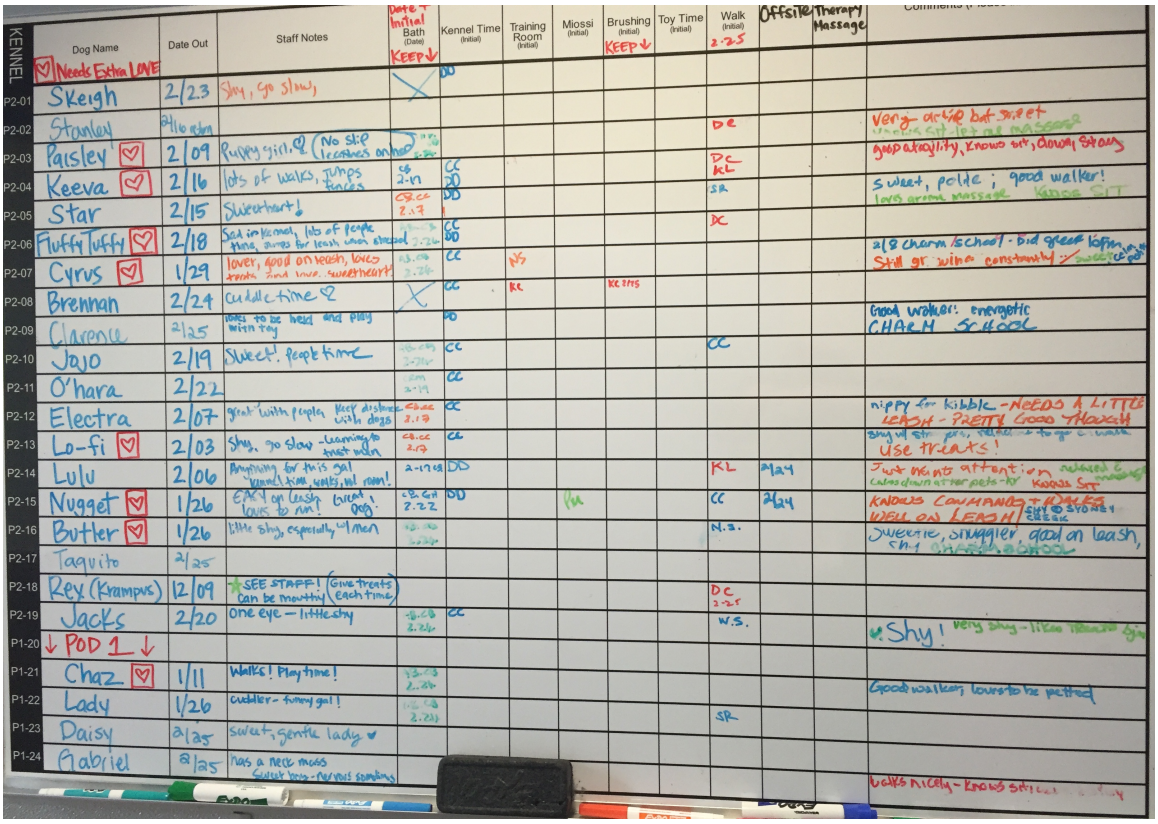

Figure 5.1: Whiteboard at Woods Humane Society

The information presented on the whiteboard about each dog was minimal, confusing, and inconsistent. Maintaining the whiteboard was an ongoing task performed mainly by the volunteer coordinator. Often, this led to the whiteboard being inconsistent with the dogs available for adoption because the coordinator has many other responsibilities and could not maintain the whiteboard at all times. The available dogs at Woods Humane Society are constantly changing because of high adoption rates. In April 2016 alone, there were over 100 adoptions. This means that the dogs on the whiteboard had to be continuously updated in order to be accurate, but often

did not reflect the available dogs. The staff used the technique of writing down all the dogs names in a notepad while walking around the kennels, and then updating the whiteboard based on that list. This technique involved a lot of manual work and meant that the volunteers couldn't rely on the information on the whiteboard because it was not always accurate. Dogs were updated and erased so often on the whiteboard that it led to mistakes. Sometimes only the dog's name was updated and the rest of the information would be left from the previous dog. This is potentially dangerous, because some dogs may be listed as friendly and good with other dogs, when in reality they may be dog aggressive.

The process was complicated, and unfair to the dogs that wouldn't fit on the whiteboard. There were only 24 slots even though the shelter often has 30 to 40 dogs at once available for adoption. Since most volunteers relied on the whiteboard to select a dog, the dogs that weren't on the whiteboard were often ignored and given no attention for several days at a time. Because the space to write comments was so small, volunteers would only write a few words such as "energetic" and would not leave their initials. If the trainer or the staff had questions about a comment, they weren't able to find the volunteer who wrote it.

The initial motivation behind creating the app was to sort the dogs based on their care history. Before, when a volunteer walked into the shelter for volunteering, they saw a whiteboard that was hard to read, cluttered, and generally incomplete with many initials. Even though the volunteers are allowed to walk any dogs with a blue or pink label, they would stick to the dogs that were on the whiteboard. Although inconsistent, the information was available. The dogs in the main room or on the whiteboard would get attention, but this left the dogs in the other room to have much less social interaction with volunteers. Not only were these dogs not getting volunteer attention, but also they weren't getting attention from the public because this room is generally only open for staff and volunteers.

A lot of time was wasted while volunteers selected a dog. This process was stressful because the volunteers could never feel like they had all the information they needed before taking a dog. This led to a lot of double checking of the information on the whiteboard to make sure instructions were followed. There was a definite need to improve the manual process. The whiteboard was the technique that had always been used, and the staff never thought of automating or improving the process. They didn't have the resources or the time to invest. Cal Poly students are always looking for ways to solve real-world problems, and so as part of a Knowledge Management class the idea was presented to recommend which dogs volunteers should provide care to. The result had a much larger and more positive impact than was ever intended.

5.1.2 Context

The canine enrichment volunteers at Woods Humane Society, referred to as volunteers in the rest of this document, select a dog for care based on its name and occasional staff notes or volunteer comments written on the whiteboard. Each dog has different requirements, and providing personalized care for each dog can help it feel at home even in a stressful kennel environment. Sometimes, dogs have special requirements which are important to share with the volunteers who work with them. This was the initial intention of the whiteboard. Some dogs have fears of certain types of people or other animals, some are recovering from injury, and others can only be given certain forms of care.

Using the whiteboard technique, the volunteer had to select a dog and then locate the dog in the kennels by looking at its kennel card and matching the name to the dog. Sometimes, the dog would be out in the play yard or with another volunteer, so the volunteer had to return to the whiteboard in the other room to select a new dog. Volunteers were unable to select a dog they thought they can handle based on size,

age, or breed because this information was not listed on the whiteboard.

For the dogs not on the whiteboard, volunteers would select a dog in the kennels and determine which dog to take based on if it had an adoptable kennel card. This card was blue or pink rather than green, which distinguishes the new, not yet available dogs that volunteers are unauthorized to work with. In this case, the volunteer has to decide for themselves if they feel comfortable taking the dog not knowing its background, or talk to staff to see if there is anything they need to know. Even if volunteers provided care for dogs not on the whiteboard, there was no way to record this information.

5.1.3 Contribution

Part of the contribution of this thesis is Woods, an iPad, iPhone, and Apple Watch app which improves the experience of volunteers and increases the quality of care provided for the dogs at Woods Humane Society. The project was started in the Knowledge Management class with Carson Carroll and Vivian Fong during Winter quarter 2016. In the app, staff can mark dogs with flags depending on their care requirements and leave special instructions, and volunteers can write comments about each dog. The iPad app is used at the shelter, and volunteers can optionally download the iPhone app to their personal devices. The contribution is an improved system for managing the care of the dogs in the shelter, which eliminates the need for whiteboard maintenance. The dogs in the app are updated automatically from the database of available dogs. The time spent updating the whiteboard can now be spent adding detailed instructions for each dog to help volunteers provide appropriate care. The system is also improved through a sorting mechanism which provides a way for volunteers to quickly determine which dog needs care.

The goal of the app is to minimize the amount of time volunteers spend selecting

a dog for care, and to remove the need to manually update the whiteboard. Another goal is to increase volunteer confidence when handling the dogs and enable them to provide more consistent care. Overall, the main goal is to help improve adoptability of the shelter dogs while in the care of Woods Humane Society.

Adoptability is improved by increasing the amount of time dogs are socialized, trained, walked, and given other forms of care. With more adoptable behaviors, these dogs may get adopted faster which can lead to an increase in the number of dogs that can be saved from high kill shelters. This is because decreased length of stay in a shelter leads to fewer dogs at any given time. With fewer dogs in the shelter, more dogs can receive care. By providing dogs with more social interaction, they are more likely to present adoptable behaviors.

Background information about Woods is described in Section 5.2, followed by Related Work in Section 5.3. Requirements are discussed in Section 5.4, with Design and Workflows in Section 5.5 and Section 5.6. Implementation is discussed in Section 5.7 and Testing and Validation are discussed in Section 5.8. Finally, Conclusions and Future Work are discussed in Section 5.9 and Section 5.10.

5.2 Background

Woods Humane Society allows volunteers to provide various types of care to the dogs. Several of the terms used at the shelter are discussed below.

5.2.1 Pods

The pods refer to the series of kennels housing the adoptable dogs. There are two pods, one that is always open to the public and one that is occasionally open depending on the number of dogs available.

5.2.2 Pet Finder

A website on which almost every shelter in the United States posts their dogs. Pet Finder is the database of the dogs that are available to the public [19].

5.2.3 Length of Stay

The length of stay is the duration of time an animal remains at the shelter. This is often used as a measurement of the success of a shelter and is measured in days. This is in contrast to the number of adoptions, which is affected by the number of dogs available. Length of stay indicates how quickly the shelter places dogs into forever homes, and provides insights into how crowded the shelter is at any given time.

Comparing length of stay between two shelters, if the first shelter has a shorter length of stay and the second has a longer, even if the shelters have the same adoption rates and incoming rates, the second shelter will have more overcrowding of dogs. This leads to fewer dogs receiving care and even more negative behaviors being developed. In summary, if the adoption turnarounds are faster then the shelter will have fewer dogs to handle at any given time, and each individual dog gets more attention.

5.3 Related Work

The statistics on dogs that enter shelters are disheartening. In 1999, there was a study done on 182 shelters in 42 different states. Of the dogs that entered the shelter, 52% of them were euthanized. Of these 52%, 48% were deemed adoptable by the staff and were euthanized due to shelter space rather than behavioral issues [16].

Braun reports that "socialization and training can significantly improve the chance for adoption of shelter dogs" and that taking a shelter dog for walks significantly increases their quality of life and chances for adoption. Braun also reports a decrease

in the length of stay in shelters that have volunteer programs, which means that the extra socialization, training, and exercise by volunteers help improve adoption rates [9].

Menor-Campos and others found that 25 minutes of gentle exercise, play and human contact were successful in reducing stress in dogs placed in canine shelters. Stress in the animal shelter can lead to behavioral problems that can be an impediment to adoption or increase the return rates of the dogs. They found "exercise, play and human contact leads to lower cortisol levels and better test scores, and hence improves the behavior of the dogs" [33].

Similarly, a study by Luescher and Medlock found "20 min of daily positive training of shelter dogs, in addition to human contact provided by volunteers, increases the dogs' chances of being adopted" [31]. There is clearly a benefit of human interaction and socialization for the dogs, and increasing the number of minutes spent with each dog can lower stress and improve their chances of being adopted. While there are several apps to manage the staff and volunteers at the shelter, there are no public apps that specifically target care and enrichment for the animals. A few of the top shelter management systems are discussed below.

5.3.1 Overview

Shelter Buddy, Volgistics, and Chameleon are a subset of the animal shelter management tools. The animal shelter tools focus on managing the dogs and the people at the shelter, rather than providing a way to track specific care performed by volunteers. Even if care can be recorded, there is no animal shelter software that allows volunteers to easily record care and comments on the dogs like the Woods app.

5.3.2 Shelter Buddy

Shelter Buddy was developed for RSPCA Queensland for use as an effective database solution to manage the shelter animals. They allow animal shelters to pay a monthly service fee to be able to use the software. The software allows shelters to push their animals to the Shelter Buddy database, which then pushes the data to both popular adoption sites: Pet Finder and Adopt-a-Pet. They have features such as Finding Rover, which helps return dogs to their owners with facial recognition technology. The software "removes the need for separate databases for each department, with password access protecting different screens." Therefore, they only need one location for each animal, which can include "everything from vet treatment received or due and their physical whereabouts in the shelter." The cost of Shelter Buddy ranges from $495 to $3,500 per year depending on the features needed [10].

Shelter Buddy is a popular tool among animal shelters, and is necessary for keeping track of the animals in the shelter. However, Shelter Buddy does not provide a way to track volunteer care for the animals.

5.3.3 Volgistics

Volgistics is a management tool for tracking volunteer hours. It is separate from the database of animals, and so while it can track the hours of a type of care done by volunteers, it does not keep track of the different types of care for each animal. Volgistics provides features such as volunteer scheduling, tracking, database, application forms, and text and email functionality. The price is determined based on the number of volunteers registered, but ranges from under $10 per month for the basic level to several hundred dollars for over 1000 volunteers [51]. Volgistics keeps track of the volunteers and what type of care they are providing, but does not connect to the shelter database.

5.3.4 Chameleon

Chameleon helps with "shelter management, licensing, field operations, cashiering, and veterinary record-keeping." Chameleon allows shelters to manage and track people, animals, kennels, clinic, licenses, fields, donors, finances, reports, emails, and more. The cost for Chameleon is a one time fee of approximately $9,800 and so is on the higher end of the available software. They also provide a way to improve kennel inventory by using bar codes [11]. Chameleon does allow storing photos and behavior profiles of animals, but still does not track the care provided by the volunteers.

5.4 Requirements

The requirements were determined by personal experience and input from other volunteers and staff. The input included hearing objections and uncertainties about older volunteers being able to adjust to the use of the app. For this reason, the app had to be developed while keeping in mind an older generation who might not familiar with apps or comfortable using technology. It was determined that Woods Humane Society needed a way to manage the care provided to the dogs in a more efficient way. The high-level requirements of the software are seen as follows:

- REQ-1: All volunteers should be able to use the system.
- REQ-2: Dogs should be automatically updated from the shelter's database.
- REQ-3: Staff should be able to leave custom notes for each dog.
- REQ-4: Volunteers should be able to record care and leave comments.
- REQ-5: The information should be updated in real-time between devices.
- REQ-6: Only staff and volunteers at Woods should be able to use the system.

Based on those requirements, it was necessary to build a system that remained at the shelter for all volunteers to use (REQ-1), which contained the updated list of available dogs at all times (REQ-2) and where staff could record notes and custom flags for the dogs (REQ-3). This information, along with the information recorded by volunteers (REQ-4), must sync together (REQ-5). An approval system was required in order to distinguish volunteers and staff from the public who download the app (REQ-6).

Initially, only an iPad app developed that was meant to replace the whiteboard, while making it easier to maintain and update the information. The original solution required updating the iPad just as the whiteboard had been. After realizing that this was infeasible for staff because it involved manual updates, REQ-2 was added which required the dogs to be updated automatically. An API which syncs to Pet Finder was found that enabled automatic updating [22]. The dogs are pulled and updated in the database every five minutes. This addition eliminated the need for staff to update the whiteboard, and created an incentive for Woods Humane Society to move away from the whiteboard technique. The iPhone app was created after realizing it might be difficult for more than one volunteer to be using the iPad at once. Because most volunteers were meant to use the iPad, only an iOS application was necessary for use of the app. Since it was thought the app would only be used at the shelter, there was no need to consider a web application.

As discussed in Section 3.2, trial days can be beneficial to shaping the app. In the few trial days at the shelter, the feedback from staff required adding a few additional care categories before they would be willing to consider using the app full time. These additional categories were massage and brushing, which are not provided often or important for volunteers to record. Because of personal experience with volunteering, these categories were not determined to be essential to the app. The other feedback received was that they need to know exactly which pod and kennel each dog was in

on the main list of dogs. Both of these features accounted for several weeks in total of development time, and yet both were eliminated in the end. This was a reminder that the new system was not meant to replace, but to improve the current system.

5.5 Design

The Woods app was designed to meet the requirements specified in Section 5.4 which provides Woods Humane Society with an improved system to record and monitor the care for their dogs.

5.5.1 Definitions

The different forms of care that volunteers can provide to the dogs are listed in Table 5.1 below. The top care types performed by volunteers are walks, enrichment, and play yard.

5.5.2 User Experience

The app was designed to be easy to use and straightforward enough, that no instructions would be needed for volunteers and staff. The designs progressed over time, and the current designs are showcased below in Figure 5.2 for the iPhone app and Figure 5.3 for the iPad app.

In order to record a care type, volunteers have to tap three times: one on the dog, one on the care type, and lastly one to confirm and record their initials. On the iPad app, volunteers have to enter their own initials. As seen in Figure 5.2, some of the forms have an optional entry for duration so volunteers can estimate the amount of time they spend with the dog. This is useful because some volunteers spend a longer time with each dog while others try to maximize the number of dogs they

Table 5.1: Woods Definitions of Care Types

Type of Care	Description
Walk	Taking the dogs on leash around the shelter and on the walking trail.
Enrichment	Includes petting the dog in their kennel, bringing them to the volunteer room, or bringing them to an office space or other area of the shelter. After the initial training, all volunteers can provide enrichment for the dogs.
Play Yard	Taking the dogs off leash in a fenced play area. Some dogs cannot go to the play yard because they jump the fence.
Training	Working with the dog, including basic commands, walking on leash, etc. Can be done on leash or in a play yard or other room in the shelter.
Offsite	Taking the dogs on hikes, to the beach, or to events that Woods Humane Society attends.
Bath	Giving the dogs a bath. Some dogs cannot have a bath because they are already too stressed in the kennel environment or they do not do well with water.

visit. Volunteers can also select which care type to use to order the dogs, and the dogs listed at the top will need that care type most. This makes it more fair for the dogs and helps volunteers because they can see the timestamp and duration of when a dog was walked, rather than only the initials of the volunteer.

In Figure 5.3, Henry has a cross through the Play Yard care category. This indicates to volunteers that he can't go to the play yard because he jumps the fence. Additionally, the same process occurs for dogs who are unable to have a bath. The

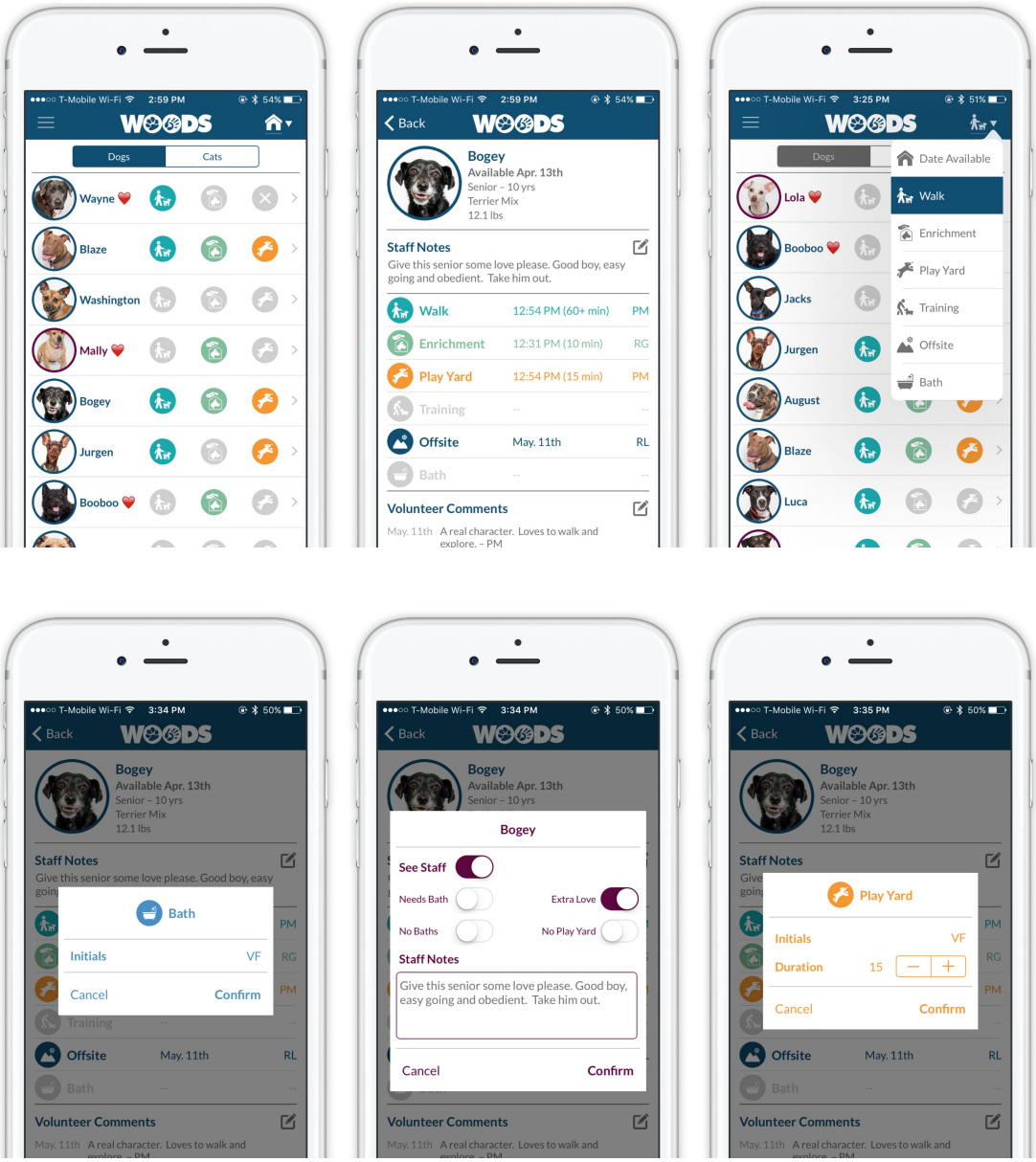

Figure 5.2: Screenshots of the Woods iPhone App

information is presented in a straightforward way so that volunteers don't have to spend time figuring out the information and instead can easily select a dog for which to provide care.

Each care entry is displayed from the current day, except offsite and bath whose entries persist for the last 30 days. The information about each dog's bath and offsite history is much more reliable and available.

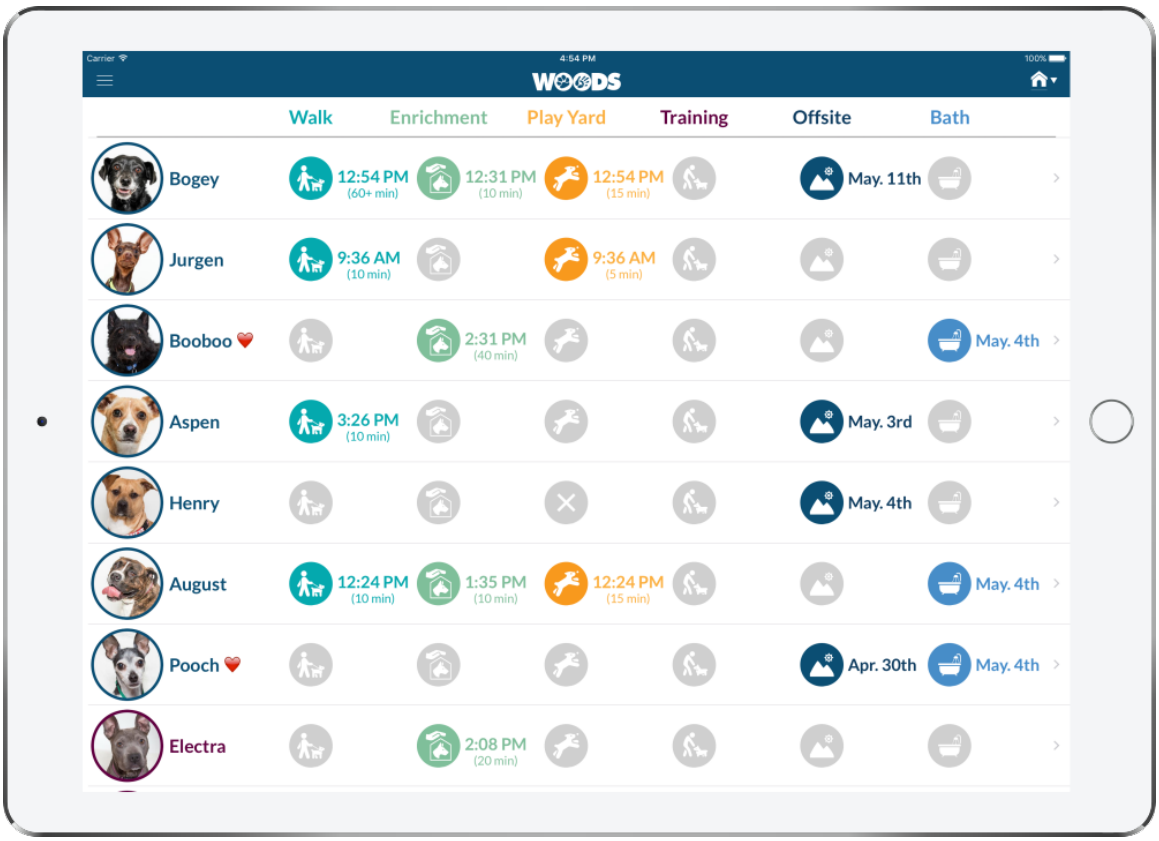

Figure 5.3: Screenshot of the Woods iPad Pro App

5.6 Workflows

This section describes the process of editing staff notes, adding and removing care, and adding and removing comments.

5.6.1 Staff Notes

Staff notes include special information about a dog that is necessary for volunteers to provide proper care, specifically a description and the following optional flags:

- **See staff**: Indicates to volunteers that this dog may have special handling requirements. Generally, newer and less experienced volunteers will stay away from these dogs.

- **Needs bath**: Indicates to the bathing volunteers that a dog needs a bath. When the dogs are sorted by bath, these dogs appear at the top. On the iPad app, they will have a flag. On the iPhone app, "NEEDS BATH!" will appear in the dog's description.

- **No baths**: Indicates to volunteers the dog cannot have a bath. Dogs with this flag will have a cross through their bath entry, and volunteers will not be able to record a bath.

- **Extra love**: Indicates the dog needs extra attention from volunteers, and displays a heart next to the dog's name. Generally this is for special needs dogs or dogs that have been at the shelter the longest.

- **No play yard**: Indicates to volunteers the dog cannot go in the play yard. Dogs with this flag will have a cross through their play yard entry, and volunteers will not be able to record a play yard.

For users with staff permissions, they see an edit button in the staff notes section on the dog info page, which allows them to turn on and off certain flags for the dogs and edit the description for the dog. Further information about user permissions is discussed in User Management, Section 5.6.4.

5.6.2 Care

Care types are defined in Table 5.1. Volunteers can record entries with their initials. For walks, enrichment, play yard, and training, volunteers additionally submit the duration for which they provided care. Each of these four care types are cleared every day. For the remaining two care types, offsite and bath, volunteers only need to record their initials. This is because the duration of a bath isn't important, and an offsite is generally for several hours. For this reason, when a volunteer records an

offsite, a 60-minute walk is added to the dog. Since minutes walked are capped at 60, the result shows "60+ mins."

For dogs flagged with "no play yard" or "no baths," these care entries will not be displayed in the list of care types available for that dog. Walks, play yard, and offsites are automatically not in the list of care types for puppies.

Volunteers can delete the care entries that they record with their personal account, but are unable to delete care entries recorded by others. For monitoring purposes, staff are allowed to delete all care entries. When users are logged in on their personal account, their initials are automatically entered. Users logged in on the Woods account have to enter their initials. This is because the Woods account is used on the iPad by all volunteers at the shelter.

5.6.3 Comments

Comments are notes left by volunteers about their experience with the dogs. They can range from a few words to very detailed recommendations for handling the dogs. Like care entries, volunteers' initials are automatically entered, while on the Woods volunteer account, initials must be entered. Similarly, users can delete their own comments and staff can delete all comments.

5.6.4 User Management

Users with staff permissions have access to a "Manage Users" tab which allows them to authorize or deny new user accounts, and maintain existing volunteer and staff accounts. Once approved as a staff or volunteer, users can be moved between the categories. Any user can be deleted permanently, whether they are a new user requesting access, a volunteer, or approved as staff. The user management process is shown in Figure 5.4. The first screenshot shows the message new users receive until

their accounts are approved. The second shows how to get to the user management screen, and the third shows the options available for new requests. More details about how users are stored in the database are found in Section 5.7.2.

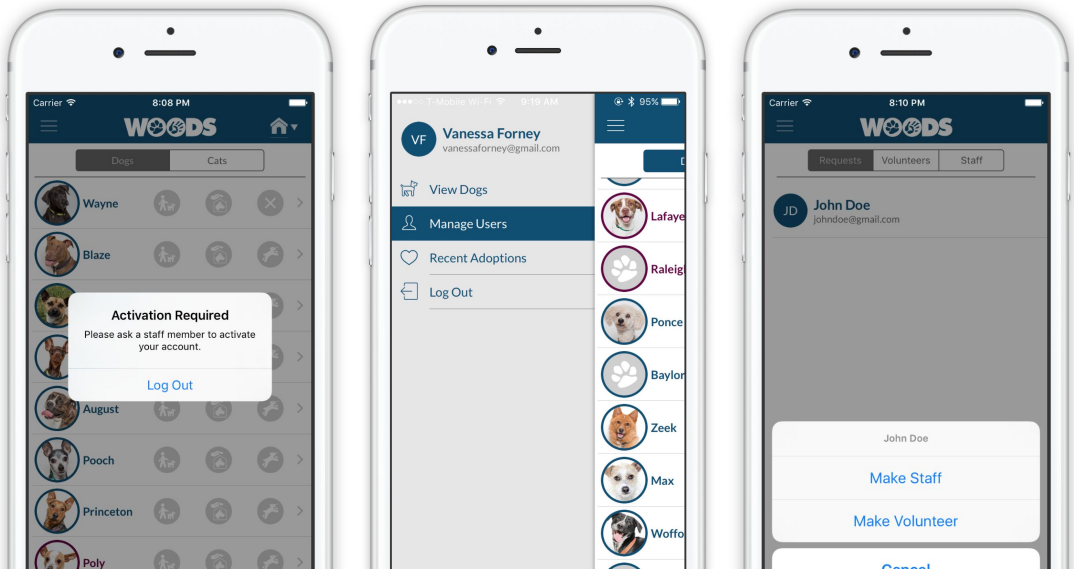

Figure 5.4: Screenshots of User Management on the Woods App

5.7 Implementation

The model design, backend decisions, recommendations, and watch implementation are discussed in this section. There were two iterations of implementation. The first was solely an iPad app, followed by an iPad Pro app and an iPhone app. Each time, a substantial piece of the app was rewritten and refactored. One example of a substantial refactor is the original use of strings to represent the different forms of care. After the realizations discussed in Section 2.1.2, the care types were organized into an enum of `CareType` holding the different care categories. For example, from `CareType.Walk` it was now possible to access the walk image, color, Firebase path, and much more.

Another iteration included abstracting the list of dogs into a `Shelter` object because of how arrays are passed, as discussed in Section 2.1.3. When the dog app was first written, the dogs were loaded in the main table view controller which contained the list of dogs. This caused problems when the watch app was implemented. When the watch app is opened, only the `AppDelegate` on the phone side is called. For this reason, the list of dogs was moved from the dog table view controller into the `AppDelegate`. The list of dogs on the watch were kept up to date because the watch received the summary of the data from the list of dogs in the `AppDelegate`. However, since the array of dogs was passed from the `AppDelegate` to the dog table view controller, the dogs on the phone app were not being updated due to the changes only affecting the array in the `AppDelegate`. Because of this, a `Shelter` object was created to store the list of dogs which created consistency between the watch and the phone. The list of dogs in the `Shelter` object is passed safely between the `AppDelegate` and the view controllers.

5.7.1 Model

An overview of the model can be seen in the UML diagram in Figure 5.5. For simplicity, only the most relevant functions and properties are displayed.

5.7.2 Backend

Firebase was chosen as the backend for Woods because the dogs and care need to update between devices in real-time and the platform needed to be free. The general strategy is to monitor dogs being added to Firebase, and have observers with callbacks that fire on the registered classes, including the currently displayed view. That way, on the dog table view there will be callbacks when any of the care items change, the dog's info changes, or a dog is added or removed. On the dog info page, callbacks

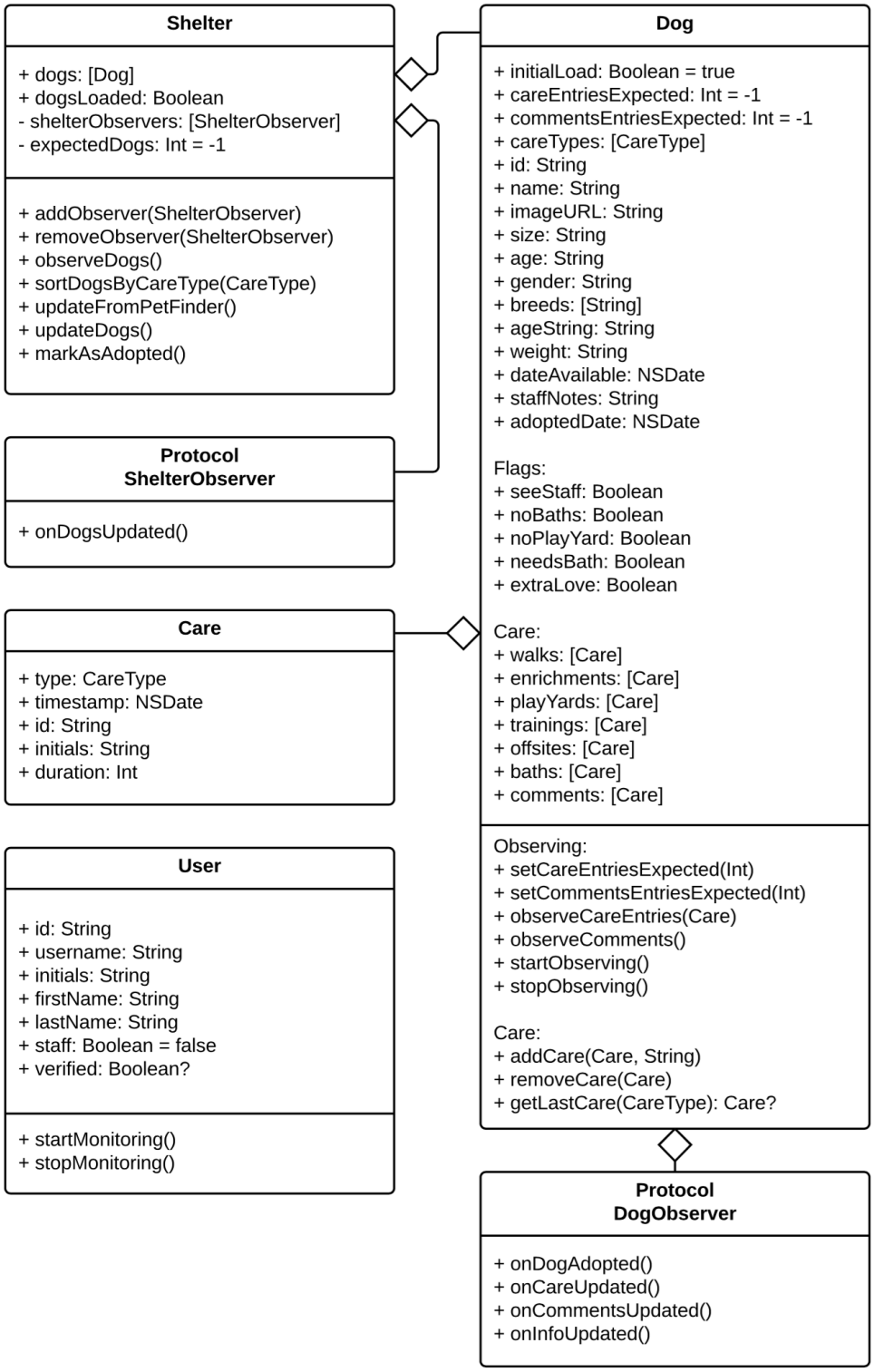

Figure 5.5: Woods UML Diagram

will only happen for changes to the currently viewed dog.

In the initial version of the app, all the care entries were stored within a dog in the database. This became problematic after adding users since there was no way to track which user added the care entry. After abstracting the care categories into their own Firebase reference, then both the user and the dog could hold a reference to the specified care entry. With the new structure, the initial loading of the dogs became more complicated. Each dog had to be loaded, followed by a nested query to get the care entries for that dog. This meant a more complex callback system had to be implemented.

The overall app has one instance of a `Shelter` object, described in Section 5.7.1. For the callbacks, the `Shelter` has a list of `ShelterObservers`. This means that any time there is a change to the dogs in the shelter anything listening for changes will receive a callback. The dogs in Firebase are observed with an `observeEventType` on `ChildAdded` and `ChildRemoved` at the URL containing the list of dogs. By narrowing the query to limit to the `adoptedDate` being equal to "nil," then the callback happens when either a new dog is added or a dog is adopted. For each dog in the shelter, the `Shelter` object also observes the dog. When changes occur to the `Shelter` object that affect the main dog page, any listener gets a callback. Finally, a single event is observed of type `Value`. When this callback returns, it is guaranteed that each `ChildAdded` event has been fired. This means the total number of dogs in Firebase is now known. Recording this value keeps track of how many more dogs need to be loaded before the data is complete.

In a similar way, the dogs observe each of their care fields and comments. First, by monitoring all of the `ChildAdded` and `ChildRemoved` calls and second by a single event to get the number of expected children for each care type and the number of comments. When a dog finishes loading, the `Shelter` object receives a callback that

the dog finished loading. Finally, when all dogs finish loading the `Shelter` object calls its listeners to notify them that the dogs have been updated. The observer pattern for the `Shelter` is outlined below:

```
func addObserver<T where T: ShelterObserver,
 T: Equatable>(observer: T) {
  if dogsLoaded {
    observer.onDogsUpdated()
  }
  shelterObservers.append(observer)
}

func removeObserver<T where T: ShelterObserver,
 T: Equatable>(observer: T) {
  for (index, shelterObserver) in shelterObservers.enumerate() {
    if let shelterObserver = shelterObserver as? T
     where shelterObserver == observer {
      shelterObservers.removeAtIndex(index)
      break
    }
  }
}
```

The `ShelterObserver` protocol is called when the data in the shelter changes. Each observer to the shelter implements this protocol:

```
protocol ShelterObserver {
  func onDogsUpdated()
}
```

The way this data is structured makes it easy to add additional observers. For the Apple Watch app, an observer is added when the watch is connected. This observer receives the callbacks on the phone and sends the updated data to the watch when needed, as described in Section 5.7.5.

Dog are stored in Firebase with the fields in Table 5.2.

Users are stored in Firebase with the fields in Table 5.3.

Care entries are stored in Firebase with the fields in Table 5.4.

Table 5.2: Woods Firebase Dog Fields

Field	Type	Description
adoptedDate	String	Available dogs will have the value "nil".
age	String	Either "Baby," "Young," "Adult," or "Senior.".
ageString	String	Formatted age in years, months, and weeks if less than two months (ex. "1 year 9 mths").
breeds	String	List of strings of the breed mixes.
dateAvailable	String	The date the dog appeared on Pet Finder.
expertHandler	Boolean	Flag for a confident handler.
extraLove	Boolean	Flag for extra attention.
needsBath	Boolean	Flag for needing a bath.
noBaths	Boolean	Flag for no baths.
noPlayYard	Boolean	Flag for no play yard.
gender	String	"M" or "F."
imageURL	String	URL of the Pet Finder photo.
size	String	Ranges from "XS" to "XL."
weight	Boolean	Parsed from the Pet Finder description.
staffNotes	String	Special requirements to handle the dog.
walks	String	List of strings of the IDs for the walk entries.
enrichments	String	List of strings of the IDs for the enrichment entries.
playYards	String	List of strings of the IDs for the play yard entries.
trainings	String	List of strings of the IDs for the training entries.
offsites	String	List of strings of the IDs for the offsite entries.
baths	String	List of strings of the IDs for the bath entries.
comments	String	List of strings of the IDs for the comments.

Table 5.3: Woods Firebase User Fields

Field	Type	Description
`firstName`	String	First name of the user.
`lastName`	String	Last name of the user.
`initials`	String	Optional value that is populated for users when recording care and comments. Removed for general volunteer and staff accounts so users can edit their initials.
`username`	String	The email of the user for sign in purposes.
`verified`	Boolean	Provides access to the app. True for both volunteers and staff. If omitted, the user will not be able to log in at all which means their request was denied or they were removed as a user from the app.
`staff`	Boolean	Optional value that allows the user to edit dog information.

Table 5.4: Woods Firebase Care Entry Fields

Field	Type	Description
`duration`	String	Length of time spent providing that care.
`initials`	String	Initials of the user who provided that care.
`timestamp`	Double	Date when the care was recorded in time since 1970.

5.7.3 Additional Details

As discussed in Section 3.5, minor details can mean the difference between the community accepting use of the app or not, and addressing these details is necessary before moving the app to production. For Woods, this meant adding notifications when a user is viewing a dog that gets adopted, showing empty table views where appropriate, and replacing the image displayed when no photo of the dog is available.

When viewing the dog info page, a notification is presented when a dog is adopted to alert the user to select another dog. Upon dismissing the notification, the user is taken back to the list of dogs. This saves the user from searching for the dog in the kennels, and is demonstrated in the screenshot in Figure 5.6.

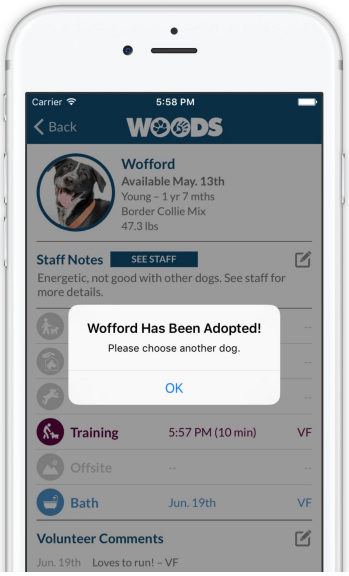

Figure 5.6: Screenshots of Adoption Notification

Additionally, it was important to improve the apperance of the app by replacing the image for a dog with no photo. By default, the shelter software pushes a photo to Pet Finder that says "No Photo Available." Therefore, the API returns this image with no indication that it's an empty image rather than a dog's image. This was replaced with a clean image of a dog paw, using the `UIImagePNGRepresentation` function to generate two `UIImage`'s which were then compared using the method `isEqualToData` on the `UIImage`. The difference can be seen in Figure 5.7, where the images before and after the change are compared.

Finally, the library DZNEmptyDataSet, discussed in Section 2.2.1, was utilized in order to provide the user with an informational message rather than a blank area of the screen. This is shown in Figure 5.8, which shows the empty table views for user

Figure 5.7: Screenshots of Dog Without Photo

requests, volunteer comments, and when no dogs are available for adoption. These are a few of many enhancements made to improve the quality of the app, and were a contributing factor for the shelter being willing to try out the app.

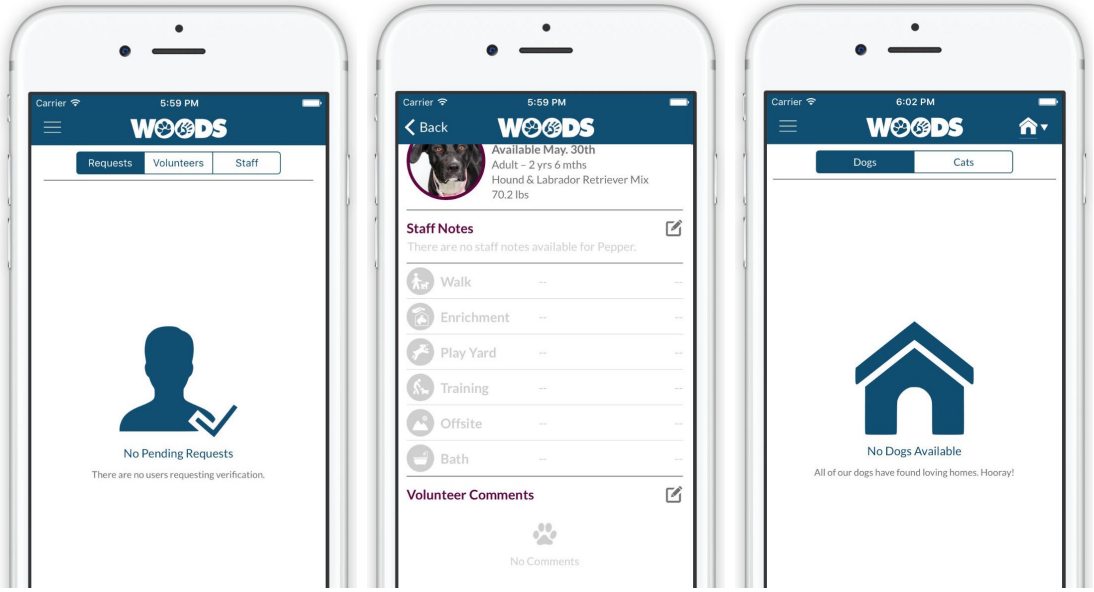

Figure 5.8: Screenshots of Empty Table Views

5.7.4 Recommendations

Volunteers use recommendations on the app to determine which dog needs care. Recommendations are generated for each care based upon the dog's characteristics and the minutes that have passed since it received that care. Puppies aren't allowed to have walks, play yard, or offsites and so they have the lowest rank for those care types. In the app, a puppy is defined as having age "Baby" on Pet Finder. Dogs with flags "no play yard" or "no baths" are given the lowest priority for those care categories. The formula was developed with Vivian Fong, and was created after generating a table of the priorities for the dog. As the duration becomes longer, the needs care level increases. As the minutes or hours ago increases, the needs care level decreases. Therefore, dogs who had a shorter amount of care farther in the past are prioritized above those with longer care that was received more recently.

This formula sorts the dogs based on the timestamp of the last care provided, but also considers the total duration of that care for the day. Since a dog may have more than one entry and the duration varies, it was important to take more information than simply the latest timestamp of the care provided. Depending on the care, the weight of the dog is included and provides a way to prioritize the larger dogs over the smaller dogs, since larger dogs tend to have more energy and require longer walks.

5.7.5 Watch

There are two types of communication that can be used between the watch and the phone: interactive messaging and background transfers. Interactive messaging is used for immediate communication, and requires both apps to be in a reachable state. Background transfers are used to send data between devices at a time that is convenient, as determined by the operating system. Within background transfers, there are three types of communication: application context, file transfer, and user

info. Application context will overwrite the existing data, and is used when only the latest information is required on either device. File transfer is used to send files between devices. User info transfers the information in FIFO order, so the information is not overwritten but received in a specific order. Resources for learning more about watch communication can be found in Section 2.2.2.

Programming on the watch app began at Cal Poly's Design & Dev Hackathon in April 2016, with the help of Carson Carroll. During the hackathon, live messaging was used to update the values on the watch. Since the watch app only uses the latest information about the number of dogs available and the number of care entries recorded, it was later determined that application context is the best way to send information between the devices. Now, the phone uses application context to send the information about the available dogs when the connection to the watch is established, and then again whenever it receives a request from the watch to update the application context. This request is sent whenever the watch app is opened.

The watch app summarizes the information on the main app so the volunteer coordinator can determine the number of adoptions and have a quick overview of the number of volunteers that have been in that day.

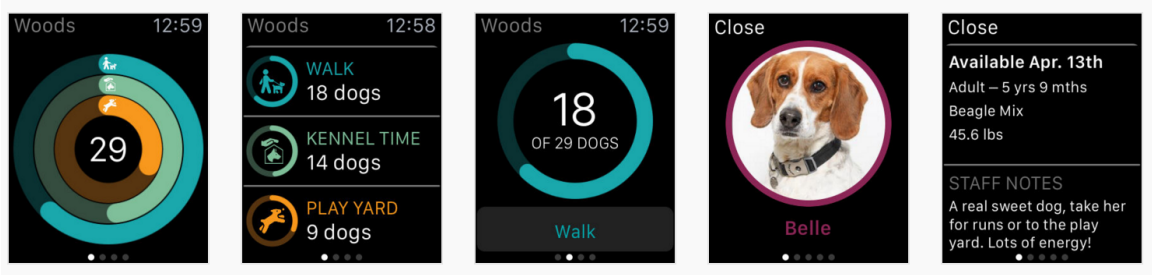

Figure 5.9: Screenshots of the Apple Watch app

5.8 Testing and Validation

This section describes the survey, results, and analysis of the Woods app. The app relieves staff members from maintaining the list of available dogs on the whiteboard, because it is now done automatically in the app. The list of dogs in the app is updated every five minutes from Pet Finder. This makes the app extremely versatile for other shelters, and easily expandable to other types of animals. From home, volunteers can check in to see if their favorite dog has been walked or played with that day. This encourages volunteers to volunteer if there haven't been enough volunteers on a certain day. The app also allows volunteers to select a dog based on breed, size, and age, which is especially useful for less confident volunteers.

Volunteers can sort the dogs based on the care they wish to give, and they can easily select a dog based on their skill and training. Using the whiteboard technique, a new day means the whiteboard is erased. Using the app, the information is maintained from previous days and so the sorting algorithm considers care history. This data can eventually be used to analyze adoption rates of dogs that were given more walks, training, enrichment, and other forms of care and potentially lead to insights which increase adoption rates of future dogs.

5.8.1 Survey

A survey was distributed to Woods volunteers and staff members in order to gain feedback about their experience with use of the whiteboard compared to use of the app. The survey was provided through email and also through the Facebook group for Woods volunteers. Out of a total of 71 responses, 56 were from people who were volunteers or staff members while the whiteboard technique was used. Of these, 45 were volunteers and 11 were staff members. The survey can be seen in Appendix B.

5.8.2 Results

In total, 83.9% of survey participants preferred the app, 10.7% had no preference, and 5.4% preferred the whiteboard. Three people reported a preference for the whiteboard. However, the answers from one of these people indicate that this was a mistake. The results for this individual are highly in favor of the app, saying that the app makes them feel more confident and including a comment that the user hopes Woods continues using the app. The remaining two who selected the whiteboard were in the age category of 51 to 70 years, and so are likely not as familiar with technology. While this coincides with the assumption that the older volunteers would have a hard time using the app, the majority of volunteers over age 51 prefer the app. Of these 16 individuals, 11 prefer the app, three have no preference, and two prefer the whiteboard.

Of the volunteers, 27.3% indicate that they volunteer more often now that they can see the dogs from home. Of the remaining, 40.9% say they volunteer the same amount and 31.8% say they cannot see the dogs from home (no iPhone or app not installed). The majority (52.3%) of volunteers indicated that it takes less time to select a dog, while only 6.8% indicated that it takes more time. In terms of where volunteers select the next dog to care for, 37.8% indicate they select the next dog to care for while working with the current dog.

The frequency of comments reported by volunteers on the whiteboard compared to the app can be seen in Table 5.5. Volunteers reported a jump of 12.6% of the amount of comments they leave, from 40.9% of volunteers often or always leaving comments on the whiteboard to 53.5% on the app. Regarding comments, 59.1% of volunteers report an increase in the level of detail in their comments, while only one volunteer feels that they write less detailed comments.

Volunteers report that they have more trust in the accuracy of the dogs' infor-

Table 5.5: Frequency of Comments

Type	Never	Rarely	Sometimes	Often	Always
Whiteboard	11.4%	18.2%	29.5%	22.7%	**18.2**%
App	9.3%	14.0%	23.3%	25.6%	**27.9**%

mation on the app (52.3%), they feel more confident taking a dog to the play yard (56.8%), and they feel more confident providing appropriate care for each dog (65.9%). Additionally, 47.7% of volunteers report they feel more a part of the volunteer community because of the app. The rest report that they do not feel any differently. Not one of the responses indicates a negative reaction to the app (i.e. less confidence or trust).

When selecting a dog they are comfortable handling, 56.8% of volunteers report that dog breed, age, size, and photo are "Very" or "Extremely" important. Similarly, staff believe that dog breed, age, size, and photo are "Very" or "Extremely" important to volunteers (90.1%). The whiteboard method only provided the dog's name, and so this information was not available before the app.

Only 26.3% of volunteers reported they would "Often" or "Always" provide care to a dog not on the whiteboard. Since the whiteboard was not always up to date, this means that the majority of dogs not on the whiteboard did not receive care. In contrast, 61.5% of volunteers indicate they would "Often" or "Always" provide care to a dog with only dog details (breed, size, age, photo). Volunteers are more comfortable taking new dogs out that are on the app, but not the new dogs that were not yet added to the whiteboard or not on the whiteboard due to a lack of space.

For staff, 63.7% believe that the quality of care for the dogs is "Very" or "Extremely" improved due to the app. Staff believe that they update the app less often than the whiteboard. On the whiteboard, five individuals reported updating the dogs'

information "Always" or "Often." On the app, three reported updating the dogs' information "Often." Finally, 36.4% of staff feel more connected to the volunteers because of the app.

5.8.3 Analysis

In summary, the dogs' lives are improved because they are getting more attention. Volunteers have more information about each dog, are spending less time deciding which dog to care for, and can more easily select a dog that meets their skill and comfort levels. The staff benefits since the app maintains the list of available dogs automatically and doesn't require daily monitoring and upkeep. This could be the reason staff reported updating the dogs' information on the app less often than the whiteboard. However, some staff may be discouraged from updating because don't have a personal iPhone and require use of the Woods iPhone to update the notes.

Previously, only a few staff updated the whiteboard. Now, all staff can update the app from the iPhone at the back desk. Even the bathing volunteer who previously had to maintain his own list of dogs he bathed can now rely on the information in the app being up to date and accurate. On the whiteboard, the dogs would inconsistently get a cross or a star if they either needed a bath or couldn't have a bath. Now, the staff can mark this clearly on a dog's profile so that the bather can sort by baths and see the dogs that need a bath at the top, and the dogs that cannot have baths at the bottom.

For the 37.8% of volunteers who now select the next dog while working with the current dog, this can add an additional five to ten minutes of attention per dog. According to research discussed in the related work, just 20 to 25 minutes of attention per day can improve a dog's adoptability [31,33]. If each dog was previously receiving ten minutes of attention on average, they could now be receiving over twenty minutes.

The extra few minutes of walks, running in the play yard, or attention in their kennel may not seem like much, but when a dog spends 24 hours a day in a cage it makes a big difference in their lives and their behavior.

Despite volunteers saying they only write comments slightly more (12.6%), the results show otherwise. As seen in Figure 5.10, there is a complete page of comments for one dog. Before, on the whiteboard, the dog with the longest and most number of comments said: "Just wants attention. Sweet and massaged. Calms down after pets. Knows sit." Now, a single comment can contain more details than all of the previous whiteboard comments combined. More experienced volunteers are even leaving comments before the staff members do. As seen in Figure 5.10, Woffard had yet to have comments added by staff, however, one of the more experienced volunteers left a note about him. The majority (54%) of volunteers indicated they would walk a dog with only volunteer comments. While previously Woffard wouldn't have been on the whiteboard as a newly available dog, he is immediately listed on the app and the majority of volunteers are willing to walk him. Finally, Princeton shows how volunteers can leave additional information about the dog that staff leaves out.

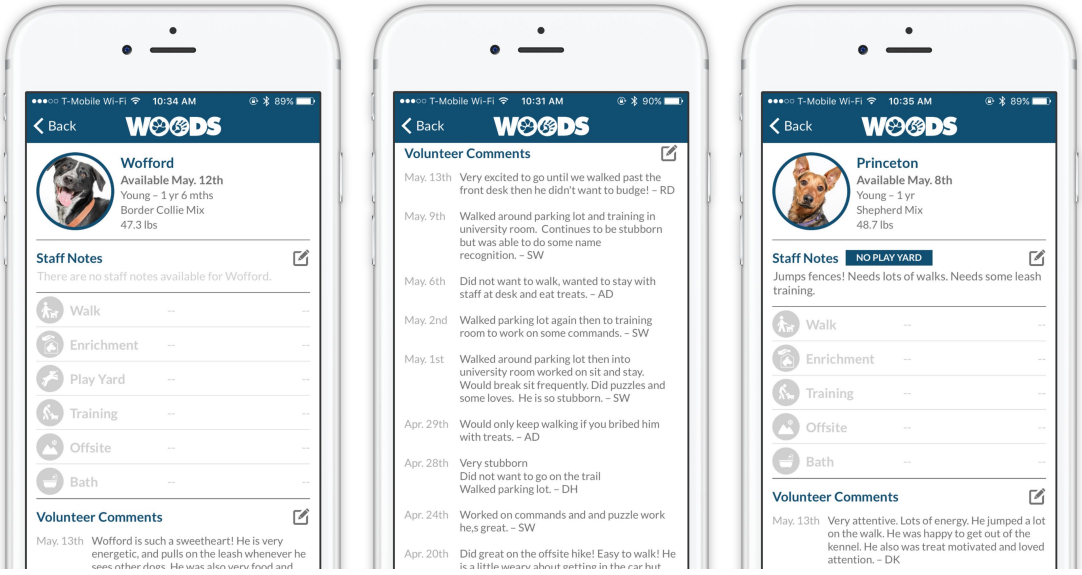

Figure 5.10: Screenshots of Comments on the Woods App

5.9 Conclusions

This section explains how the app meets the requirements and further discusses the results of the app in the community.

5.9.1 Requirements

This section explains how the system meets the requirements defined in Section 5.4.

- REQ-1: *All volunteers should be able to use the system.*

  By using an iPad Pro at the shelter, all volunteers have access to the system. Through careful design, the system is easy to use for volunteers of all familiarity levels with technology.

- REQ-2: *Dogs should be automatically updated from the shelter's database.*

  The dogs are automatically synced from Pet Finder, where the shelter uploads their dogs available for adoption. This is done every five minutes.

- REQ-3: *Staff should be able to leave custom notes for each dog.*

  Staff can provide custom "Staff Notes" as well as leave optional flags for each dog. These flags include "no play yard," "no baths," "needs bath," "see staff," and "extra love."

- REQ-4: *Volunteers should be able to record care and leave comments.*

  Volunteers can record care and comments through the iPad or their personal iPhone. Each dog has the categories of care they are allowed to receive.

- REQ-5: *The information should be updated in real-time between devices.*

Using Firebase, the data is updated in real-time between devices. When a care or comment is recorded on one device, it is immediately reflected on all other devices.

- REQ-6: *Only staff and volunteers at Woods should be able to use the system.*

  By requiring new members to have approval before they can use the app, the app restricts access to people who are authorized. Staff can edit the dog's information and set flags, and volunteers can record care and comments.

5.9.2 Discussion

After the Woods app was developed and fully functional, Woods Humane Society was still reluctant to adopt the new technology. Two trial days were held where volunteers and staff used the app instead of the whiteboard, and feedback was recorded. The results from both staff and volunteers were extremely positive, but they were still adamant about continuing to use the whiteboard. Finally, the the staff at Woods Humane Society agreed to erase the whiteboard temporarily and test out the system. It was agreed that if there were any issues, then the data from the app would be copied back to the whiteboard. After a few weeks, the staff began to appreciate the benefits of the app and noticed that the older volunteers had no trouble using it. Since that day, the whiteboard remains untouched.

If there had been even one issue with a dog not showing up, a volunteer not being able to record care, or older volunteers being confused about the app, then the shelter would have stopped use of the app. There wasn't any room for error, and the app had to be fully functional from the start.

Larger organizations have invested a lot of time and energy into their old process, and so can be hesitant to try new technology or experience change. When given the opportunity to see how technology can improve their lives, they are more willing

to accept that change. Without seeing firsthand how the new system can improve efficiency, there is too much risk for an organization to migrate to the new technology. There was a notable transformation at Woods Humane Society; staff went from not even considering switching to the app from the whiteboard to embracing the app and never wanting to return to the old process. The results were extremely rewarding, and exemplified the power of technology. When there is a problem in the world, computer scientists have opportunity to write software to improve an inefficient system.

The Woods app provides many benefits for staff and volunteers. The most notable changes are the improvement in volunteer confidence, the automatic updating of the dogs from Pet Finder, and more detailed information on the recently available dogs. Finally, volunteers are leaving more comments and including more details. The Woods app is another demonstration of how mobile apps can provide automation and data tracking to improve the community.

5.10 Future Work

The future for the Woods app includes implementation of new features and improvements to the code organization and design. As new features are added, the app must be kept simple and easy to use for older volunteers and those who are not tech savvy.

5.10.1 Expansion

Expanding to other shelters would allow the benefits of the app to expand beyond the Woods Humane Society community. In order to expand to other shelters, there needs to be a way for new shelters to request access to the app, and to associate users with corresponding shelters. Shelter managers need to be able to personalize the care types offered at their shelter.

5.10.2 Other Animals

Another useful feature would be to add cats and other animals to the app. Woods Humane Society only handles dogs and cats, but other shelters have different animals that need other forms of care.

5.10.3 Data Analysis

Because all the data is tracked, it could be analyzed to determine how care affects the adoption rates of the dogs. The app could provide recognition for top volunteers, including the volunteer with the most minutes spent for each care or the highest number of comments. By helping volunteers feel more responsible for a dog's adoption success, they can feel like they are making more of a difference and volunteer more often. Including a list of the volunteers who provided care for a dog that was adopted could be useful to help volunteers feel more responsible. Volunteers may also be interested in the number of adoptions each week or month, or the total number of care entries recorded for each category.

5.10.4 Recommendations

The recommendations for care could be improved by including additional information, such as taking into account other forms of care. For example, when sorting by walks, a dog with no attention that day should be sorted above a dog who had a period of training or a trip to the play yard. Right now, each type of care is independent.

5.10.5 Returns

Sometimes, dogs are returned to the shelter. When they are added back to Pet Finder, they are listed as a new dog with a new ID. If there was an algorithm to check if the

dog was previously at the shelter, then the dog's entire history could be imported. The algorithm could include a combination of the dog's name, breed, age, weight, and photo to determine if the dog was a match. This would prevent staff from needing to rewrite the dog's description and the comments would be maintained.

5.10.6 Android

Porting the app to Android would be another significant contribution. While only the iPad version is necessary at the shelter, it is convenient to have the app available on phones so volunteers can update their comments from home. Cal Poly students from the Android class have started development, and they hope to release a beta version for Android devices in the next few weeks.

Chapter 6

CONCLUSIONS

As the interest in app development continues to grow, there are many opportunities to develop apps that can make a difference. As demonstrated in this thesis, apps can benefit the community by improving manual processes, increasing communication, and providing useful data for analysis. Those experiencing a problem firsthand can provide the strongest insights into how the process could be improved.

In San Luis Obispo, students are a big part of the community. In accordance with Cal Poly's "learn by doing" philosophy, students can and should apply their knowledge and use school projects to meet the needs of non-profit organizations and other groups within the community. Many students have already followed this path, including the students who developed the SLO transit app, Punch'd, Poly Ratings, and Poly View, which aim to improve an experience the community.

The Poly Rides and Woods iOS applications are two examples where needs in a community were met, and the introduction of technology in the form of mobile apps improved a manual process in the community. The Poly Rides app provides Cal Poly students with an easy way to find ridesharing partners. The Woods app automates the process of updating information about the dogs and provides care tracking by volunteers at Woods Humane Society. The app eliminates the need for staff to update the whiteboard, and increases time spent with the dogs, increases volunteers' confidence in handling the dogs appropriately, and increases the number and details of comments left by volunteers. The impact of the Woods app was more far-reaching than was ever expected. In the future, the goals are to find a committed development team and to improve and prepare Poly Rides and Woods for widespread use. Poly Rides has the potential to transform ridesharing at Cal Poly and other

universities. Woods has the potential to reduce the length of stay and improve the quality of care each animal receives, and to change the way care is managed at animal shelters.

Because of the success of both apps, there are plans for future work to improve user experience and functionality, and expand beyond the local community. This thesis provides a framework for app development to benefit the community, and describes considerations and resources to facilitate the process. Technology has the power to provide functionality to improve manual systems and to enable organizations to more effectively and efficiently serve their communities. We hope that this work encourages readers to give back to their communities, either by bringing technology to organizations in need or providing additional skills or expertise.

BIBLIOGRAPHY

[1] Apple. About storyboards, scenes, and connections. https://developer.apple.com/library/ios/recipes/xcode_help-IB_storyboard/Chapters/AboutStoryboards.html, 2015.

[2] Apple. Apple press info. http://www.apple.com/pr/library/2014/06/02Apple-Releases-iOS-8-SDK-With-Over-4-000-New-APIs.html, May 2016.

[3] Apple. Enumerations. https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Enumerations.html, May 2016.

[4] Apple. ios human interface guidelines. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html, May 2016.

[5] Apple. Protocols. https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Protocols.html, March 2016.

[6] Apple. Ui design dos and don'ts. https://developer.apple.com/design/tips/, May 2016.

[7] P. Biaecki. 27 ios open source libraries to skyrocket your development. https://medium.com/app-coder-io/27-ios-open-source-libraries-to-skyrocket-your-development-301b67d3124c#.vql4qnj78, May 2016.

[8] BlaBlaCar. Connecting people who need to travel with drivers who have empty seats. https://www.blablacar.co.uk/, November 2015.

[9] G. Braun. Taking a shelter dog for walks as an important step in the resocialization process. *Journal of Veterinary Behavior: Clinical Applications and Research*, 6(1):100, 2011.

[10] S. Buddy. Shelter buddy. `http://www.shelterbuddy.com/`, May 2016.

[11] Chameleon. Chameleon. `http://www.chameleonbeach.com/`, May 2016.

[12] Chronos. Trusted online communities: Signs of a brighter future. `http://www.betrustman.com/`, November 2012.

[13] CocoaPods. What is cocoapods. `https://cocoapods.org/`, May 2016.

[14] T. M. Converter. Convert turkish lira to united states dollar. `http://themoneyconverter.com/TRY/USD.aspx`, February 2016.

[15] M. Crippa. Awesome swift. `https://github.com/matteocrippa/awesome-swift`, May 2016.

[16] J. Deleeuw. Animal shelter dogs: Factors predicting adoption versus euthanasia. Master's thesis, Wichita State University, December 2010.

[17] J. Ellis. Cal poly ride share. `https://www.facebook.com/groups/250502971675365/`, May 2016.

[18] Fastlane. Fastlane tools. `https://fastlane.tools/`, May 2016.

[19] P. Finder. Pet finder. `http://petfinder.com/`, May 2016.

[20] Firebase. Structuring data. `https://www.firebase.com/docs/ios/guide/structuring-data.html`, May 2016.

[21] V. Forney. ios development: Watch connectivity. https://medium.com/@vanessaforney/ios-development-watch-connectivity-32415d415854, May 2016.

[22] J. Fox. Pet finder api ios framework. https://github.com/atljeremy/PetFinderAPI, May 2016.

[23] Futurice. iOS good practices. https://github.com/futurice/ios-good-practices, June 2016.

[24] A. Haskins. Geofire. https://github.com/firebase/geofire/, May 2016.

[25] C. Herbert. Getting data to your watchos 2 app. https://blog.curtisherbert.com/data-synchronization-with-watchos/, May 2016.

[26] Icons8. Icons. https://icons8.com/, May 2016.

[27] Kogg. Instant logo search. http://instantlogosearch.com/, May 2016.

[28] K. Lacker. Mongodb cost breakdown. https://www.compose.io/articles/mongodb-cost-breakdown/, May 2016.

[29] M. S. Limited. Prepo. https://itunes.apple.com/us/app/prepo/id476533227?mt=12, May 2016.

[30] Y. Lin. Awesome watchos. https://github.com/yenchenlin1994/awesome-watchos, May 2016.

[31] M. R. Luescher, A. The effects of training and environmental alterations on adoption success of shelter dogs. *Applied Animal Behaviour Science*, 117(12):63 – 68, 2009.

[32] Lyft. Lyft. https://www.lyft.com/, November 2015.

[33] M. J. L. R. Menor-Campos, D. Effects of exercise and human contact on animal welfare in a dog shelter. *Veterinary Record*, 169(15):388, 2011.

[34] N. Nazari. Watchconnectivity: Say hello to wcsession. `https://www.natashatherobot.com/watchconnectivity-say-hello-to-wcsession/`, May 2016.

[35] O. of Migrant Education. Comprehensive needs assessment. `https://www2.ed.gov/admins/lead/account/compneedsassessment.pdf`, 2001.

[36] Realm. Swiftlint. `https://github.com/realm/SwiftLint`, May 2016.

[37] E. Redmond. Moving on. `http://blog.parse.com/announcements/moving-on/`, May 2016.

[38] K. S. S. Gurtner, R. Reinhardt. Designing mobile business applications for different age groups. *Technological Forecasting and Social Change*, 88:177 – 188, 2014.

[39] B. Scheirman. Nsdateformatter. `http://nsdateformatter.com/`, May 2016.

[40] SendGrid. Email delivery & transactional email service. `https://sendgrid.com/`, May 2016.

[41] W. H. Society. About woods. `http://www.woodshumanesociety.org/about-woods/`, May 2016.

[42] J. Squires. Jsqmessagesviewcontroller. `https://github.com/jessesquires/JSQMessagesViewController`, May 2016.

[43] Statista. Number of apps available in leading app stores as of july 2015. http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, May 2016.

[44] B. Terhechte. Advanced & practical enum usage in swift. https://appventure.me/2015/10/17/advanced-practical-enum-examples/, May 2016.

[45] C. Thai. watchos 2: How to communicate between devices using watch connectivity. http://kristina.io/watchos-2-how-to-communicate-between-devices-using-watch-connectivity/, May 2016.

[46] Tripda. Tripda. https://www.tripda.com/, November 2015.

[47] Uber Technologies, Inc. Uber. https://www.uber.com, November 2015.

[48] M. Vansadia. Mvautocompleteplacesearchtextfield. https://github.com/mrugrajsinh/MVAutocompletePlaceSearchTextField, May 2016.

[49] Venmo. Synx. https://github.com/venmo/synx, May 2016.

[50] D. Verner. ios dev weekly. https://iosdevweekly.com/, May 2016.

[51] Volgistics. Volgistics. https://www.volgistics.com/, May 2016.

[52] Volt. Instanbul's inner city ridesharing app. https://www.thevoltapp.com/, February 2016.

[53] W. Wang. Vvdocumenter. https://github.com/onevcat/VVDocumenter-Xcode, May 2016.

[54] Zimride. Zimride by enterprise. https://zimride.com/, November 2015.

[55] I. Zurbuchen. Dznemptydataset.

`https://github.com/dzenbot/DZNEmptyDataSet`, June 2016.

APPENDICES

Appendix A

POLY RIDES SURVEY

The following 12 pages contain a copy of the documents sent to the Cal Poly Human Subjects Committee (HSC), and the survey given to the Cal Poly community. The questions in the survey were optional with the exception of the initial consent form, and the final questions relating to reasons for not sharing were given to those who said they were not interested in ridesharing. The survey was reviewed and accepted by the Cal Poly HSC.

User Survey: Developing a Scalable Rideshare Application for University Students

By Vanessa Forney

vforney@calpoly.edu

Advisor: Chris Lupo

clupo@calpoly.edu

Computer Science

California Polytechnic State University

January 5, 2016

Abstract

Contents

1 Statement of Purpose

The purpose of this survey is to accompany my thesis document in determining what university students find necessary in a ridesharing application. The goal is to determine which features are necessary to provide the level of trust students expect. One of the features that is found to be necessary will then be implemented in the app, Poly Rides, and a follow up study will be conducted. Poly Rides is an iOS and Andorid application that conveniently lets students find and share long distance rides. Poly Rides enables students who have extra seats in their car to open them up to others who would previously drive separately or don't have a car themselves.

1.1 Benefits

This survey will provide data for my thesis as well as benefit Cal Poly students by determining which features they want in Poly Rides.

1.2 Hypothesis

The hypothesis of the survey are as follows: Students expect a high level of trust in a ridesharing app for their university, and requiring verification of the school email address is important.

2 Methods

The methods section describes the subjects, experimenters, and materials and procedures that will be used when conducting this user survey.

2.1 Subjects

The main source of subjects will be from Cal Poly Facebook groups, including the "Class of 20.." and rideshare groups. The goal is to have the majority of the responders be from Cal Poly.

2.2 Experimenters

Vanessa Forney, the author of the thesis, will administer the survey by posting on the previously mentioned Facebook groups.

2.3 Materials and Procedures

The questionnaire will be given as a Google survey with the following questions that can be seen in this Google form.

3 Informed Consent Form

The informed consent form will be provided on the Google form as the first page, and the user will be able to accept to decline consent to participate in the survey.

INFORMED CONSENT TO PARTICIPATE IN A RESEARCH PROJECT: "Developing a Scalable Rideshare Application for University Students."

A research project on Developing a Scalable Rideshare Application for University Students is being conducted by Vanessa Forney, a graduate student in the Department of Computer Science at Cal Poly, San Luis Obispo under the supervision of Dr. Chris Lupo. The purpose of the study is to determine which features are important to Cal Poly students in a rideshare application.

You are being asked to take part in this study by completing the following questionnaire. Your participation will take approximately 5-10 minutes. Please be aware that you are not required to participate in this research, you may omit any items that you prefer not to answer, and you may discontinue your participation at any time without penalty.

There are no risks anticipated with participation in this study. Your responses will be provided anonymously to protect your privacy. Potential benefits associated with the study include improving Poly Rides by determining which features students find important.

If you have questions regarding this study or would like to be informed of the results when the study is completed, please feel free to contact Vanessa Forney at vforney@calpoly.edu, or her advisor Chris Lupo at clupo@calpoly.edu. If you have concerns regarding the manner in which the study is conducted, you may contact Dr. Michael Black, Chair of the Cal Poly Human Subjects Committee, at (805) 756-2894, mblack@calpoly.edu, or Dr. Dean Wendt, Dean of Research, at (805) 756-1508, dwendt@calpoly.edu.

If you agree to voluntarily participate in this research project as described, please indicate your agreement by completing and submitting the following questionnaire. Please print a copy of this consent form now for your reference, and thank you for your participation in this research.

Appendices

Appendix A Below is the survey presented to students from Cal Poly, shared via Facebook on the various class pages and rideshare page.

User Survey

* Required

INFORMED CONSENT TO PARTICIPATE IN A RESEARCH PROJECT: "Developing a Scalable Rideshare Application for University Students."

A research project on Developing a Scalable Rideshare Application for University Students is being conducted by Vanessa Forney, a graduate student in the Department of Computer Science at Cal Poly, San Luis Obispo under the supervision of Dr. Chris Lupo. The purpose of the study is to determine which features are important to Cal Poly students in a rideshare application.

You are being asked to take part in this study by completing the following questionnaire. Your participation will take approximately 5-10 minutes. Please be aware that you are not required to participate in this research, you may omit any items that you prefer not to answer, and you may discontinue your participation at any time without penalty.

There are no risks anticipated with participation in this study. Your responses will be provided anonymously to protect your privacy. Potential benefits associated with the study include improving Poly Rides by determining which features students find important.

If you have questions regarding this study or would like to be informed of the results when the study is completed, please feel free to contact Vanessa Forney at vforney@calpoly.edu, or her advisor Chris Lupo at clupo@calpoly.edu. If you have concerns regarding the manner in which the study is conducted, you may contact Dr. Michael Black, Chair of the Cal Poly Human Subjects Committee, at (805) 756-2894, mblack@calpoly.edu, or Dr. Dean Wendt, Dean of Research, at (805) 756-1508, dwendt@calpoly.edu.

If you agree to voluntarily participate in this research project as described, please indicate your agreement by completing and submitting the following questionnaire. Please print a copy of this consent form now for your reference, and thank you for your participation in this research.

1. **Statement of Consent** *

   Mark only one oval.

   ◯ I have read the above information and I agree to voluntarily participate in this research project as described. *Skip to question 2.*

   ◯ I do not agree to participate in this research. *Stop filling out this form.*

Background

Poly Rides is an iOS and Android application released last fall that provides an easy to use interface allowing students to share rides. Since rides are saved as GPS locations, we conveniently sort the rides based on proximity to your desired route and limit rides to within 24 hours of your specified departure time. I am collecting data for my thesis research and to improve the app to suit the needs of Cal Poly students. Your participation is much appreciated!

2. **What year are you in school?**
   Mark only one oval.

   - ◯ 1st year
   - ◯ 2nd year
   - ◯ 3rd year
   - ◯ 4th year
   - ◯ 5th year
   - ◯ Master's student
   - ◯ Faculty or staff
   - ◯ Not a student
   - ◯ Other: _____

3. **What is your gender?**
   Mark only one oval.

   - ◯ Male
   - ◯ Female
   - ◯ Prefer not to answer

4. **How often do you go home per year?**
   Mark only one oval.

   - ◯ Never
   - ◯ Breaks only
   - ◯ Breaks and 1-2 weekends
   - ◯ Breaks and 3-5 weekends
   - ◯ Breaks and 6-10 weekends
   - ◯ Breaks and more than 10 weekends
   - ◯ Other: _____

Areas in California

5. **Which area are you from in California?**
   Please use the diagram above.
   Mark only one oval.

   - ◯ San Francisco Bay Area
   - ◯ Central Coast
   - ◯ Central Valley
   - ◯ Los Angeles County
   - ◯ Orange County
   - ◯ Island Empire
   - ◯ San Diego
   - ◯ Deserts
   - ◯ North Coast
   - ◯ Gold Country
   - ◯ Shasta Cascade
   - ◯ High Sierra
   - ◯ Outside of California

6. **Do you have a a smart phone?**
   Mark only one oval.

   - ◯ Yes, Android
   - ◯ Yes, iPhone
   - ◯ Yes, Other
   - ◯ No

7. **How do you normally get home for breaks or weekend trips?**
   Mark only one oval.

   - ◯ Drive myself
   - ◯ Rideshare
   - ◯ Bus
   - ◯ Train
   - ◯ Fly
   - ◯ Other: _____

8. **Have you used any of the following for ridesharing?**
   Select all that apply.
   Check all that apply.

   - ☐ Poly Rides
   - ☐ Cal Poly Facebook Rideshare page
   - ☐ Craigslist
   - ☐ Other: _____

9. **Have you heard of the Cal Poly Facebook Rideshare page?**
   Mark only one oval.

   - ◯ Yes
   - ◯ No

10. **Have you posted a ride using the Cal Poly Rideshare Facebook page?**
    Mark only one oval.

    - ◯ Yes
    - ◯ No

11. **Have you found a ride using the Cal Poly Rideshare Facebook page?**
    Mark only one oval.

    - ◯ Yes
    - ◯ No, but I tried searching and found no matches
    - ◯ No

12. **Have you heard of Poly Rides?**
    Poly Rides is an iOS and Android application for Cal Poly students to coordinate long distance rides.
    Mark only one oval.

    - ◯ Yes, I have it installed
    - ◯ Yes, but I have not installed it
    - ◯ No

13. **Have you posted a ride using Poly Rides?**
Mark only one oval.

- ◯ Yes
- ◯ No

14. **Have you found a ride using Poly Rides?**
Mark only one oval.

- ◯ Yes
- ◯ No, but I tried searching and found no matches
- ◯ No

15. **What option best describes how you feel about ridesharing interfaces?**
Mark only one oval.

- ◯ I prefer using a mobile interface to find rideshares.
- ◯ I prefer using a web interface to find rideshares.
- ◯ I prefer using word of mouth to find rideshares.
- ◯ I prefer not to participate in rideshares.
- ◯ Other: _____

16. **What is the main reason you are interested in ridesharing?**
Mark only one oval.

- ◯ Saving gas cost
- ◯ Meeting new people
- ◯ Benefiting environment
- ◯ Traveling in carpool lanes
- ◯ Convenience
- ◯ I am not interested in ridesharing
- ◯ Other: _____

17. **Are you interested in being a passenger or driver?**
Mark only one oval.

- ◯ Passenger or driver
- ◯ Passenger
- ◯ Driver
- ◯ Neither

18. **How much time would you be willing to add to your drive for pick-up/drop-off?**
Mark only one oval.

- ◯ 0 minutes
- ◯ 1-15 minutes
- ◯ 16-45 minutes
- ◯ 46 minutes or more

19. **As a driver, who would you be willing to give a ride to?**
Select all that apply.
Check all that apply.

- ☐ Stranger
- ☐ Cal Poly student with NO mutual friends
- ☐ Cal Poly student with mutual friends
- ☐ Friend
- ☐ Other: _____

20. **As a passenger, who would you be willing to accept a ride from?**
Select all that apply.
Check all that apply.

- ☐ Stranger
- ☐ Cal Poly student with NO mutual friends
- ☐ Cal Poly student with mutual friends
- ☐ Friend
- ☐ Other: _____

21. **As a driver, would you be willing to provide a short description about yourself in the app?**
This includes a few sentences about yourself, car make/model, and your Facebook photo.
Mark only one oval.

- ◯ Yes, but only to other Cal Poly students
- ◯ Yes, to anyone with the ridesharing app
- ◯ No

22. **How would you prefer to share gas costs?**
Mark only one oval.

- ◯ Pay within the app
- ◯ Handle payments outside of app
- ◯ Other: _____

23. **Please rank the following features in order of importance.**

1 = Least Important, 9 = Most Important

Mark only one oval per row.

	1	2	3	4	5	6	7	8	9
Verification of student (calpoly.edu email)	◯	◯	◯	◯	◯	◯	◯	◯	◯
Ladies-only rides (for women)	◯	◯	◯	◯	◯	◯	◯	◯	◯
Description of driver and car	◯	◯	◯	◯	◯	◯	◯	◯	◯
Payments for rides inside app	◯	◯	◯	◯	◯	◯	◯	◯	◯
Ratings of potential drivers	◯	◯	◯	◯	◯	◯	◯	◯	◯
Past passengers of potential drivers	◯	◯	◯	◯	◯	◯	◯	◯	◯
Ratings of potential passengers	◯	◯	◯	◯	◯	◯	◯	◯	◯
Mutual friends with drivers/passengers	◯	◯	◯	◯	◯	◯	◯	◯	◯
Response rate of drivers	◯	◯	◯	◯	◯	◯	◯	◯	◯

24. **Do you have any other suggestions for features or feedback about Poly Rides?**

.....

.....

.....

.....

.....

25. **Are you willing and interested in participating in ridesharing?**

Mark only one oval.

- ◯ Yes *Stop filling out this form.*
- ◯ No *Skip to question 26.*
- ◯ Not sure *Skip to question 26.*

26. **For what reasons are you not interested in ridesharing?**

Select all that apply.

Check all that apply.

- ☐ Safety
- ☐ Flexibility of departure time
- ☐ Coordinating pick-up/drop-off locations
- ☐ No need (use bus/train/plane)
- ☐ Unaware of Poly Rides
- ☐ Hard to find carpool partner
- ☐ Already have carpool partner
- ☐ No room in car
- ☐ Other: .....

27. **What is the MOST IMPORTANT reason that prevents you from ridesharing?**
Mark only one oval.

- Safety
- Flexibility of departure time
- Coordinating pick-up/drop-off locations
- No need (use bus/train/plane)
- Unaware of Poly Rides
- Already have carpool partner
- Hard to find carpool partner
- No room in car
- Other: _____

Appendix B

WOODS SURVEY

The following six pages contain a copy of the surveys given to volunteers at Woods Humane Society. The initial question determined if the users were the right group to take the survey and a later question divides the users into staff and volunteers, which directs the rest of the survey questions.

Woods Humane Society App Survey

This survey should take no more than five minutes.

* Required

1. **Were you a volunteer or a staff member while we were using the whiteboard?**
   Mark only one oval.

   - ◯ Yes *Skip to question 2.*
   - ◯ No *Skip to question 23.*

Woods Humane Society App Survey

2. **What age group do you fall in?**
   Mark only one oval.

   - ◯ Under 18
   - ◯ 19 to 30
   - ◯ 31 to 50
   - ◯ 51 to 70
   - ◯ 71 or older

3. **Which method do you prefer?**
   Mark only one oval.

   - ◯ Whiteboard
   - ◯ iPad/iPhone app
   - ◯ No preference

4. **Are you a staff member or volunteer?** *
   Mark only one oval.

   - ◯ Staff *Skip to question 17.*
   - ◯ Volunteer *Skip to question 5.*

Volunteers

Care is defined as walks, kennel time (enrichment), play yard, training, offsites, or baths.

5. **Indicate the importance of the following when selecting a dog you are comfortable handling.**
Mark only one oval per row.

	Not at all	Slightly	Moderately	Very	Extremely
Staff notes in the app	◯	◯	◯	◯	◯
Verbal information from staff	◯	◯	◯	◯	◯
Volunteer comments	◯	◯	◯	◯	◯
Dog breed, age, size, photo	◯	◯	◯	◯	◯

6. **How has seeing the dogs from home in the app influenced your frequency of volunteering?**
Mark only one oval.

- ◯ I volunteer more often
- ◯ I volunteer the same amount
- ◯ I volunteer less often
- ◯ I cannot see the dogs from home (no iPhone or app not installed)

7. **How has the app affected the amount of time you need to select a dog?**
Mark only one oval.

- ◯ It takes more time
- ◯ The time it takes has not changed
- ◯ It takes less time

8. **Where do you most often select the next dog to care for?**
Mark only one oval.

- ◯ In the hallway with the whiteboard/iPad
- ◯ While working with a dog (kennel time, walk, play yard, etc)
- ◯ Other: .....

9. **How often did you write volunteer comments on the whiteboard?**
Mark only one oval.

- ◯ Always
- ◯ Often
- ◯ Sometimes
- ◯ Rarely
- ◯ Never

10. **How often do you write volunteer comments in the app?**
Mark only one oval.

- Always
- Often
- Sometimes
- Rarely
- Never

11. **Indicate the frequency you would provide care to a dog with the following.**
Mark only one oval per row.

	Never	Rarely	Sometimes	Often	Always
Not on whiteboard (pink/blue card)					
Whiteboard with only volunteer comments					
Whiteboard with only staff notes					
App with only dog details (breed, size, age, photo)					
App with only volunteer comments (and dog details)					
App with only staff notes (and dog details)					

These questions compare the app to the whiteboard.

12. **Describe the level of detail in your comments since the app was introduced.**
Mark only one oval.

- I write more detailed comments
- The level of detail in my comments has not changed
- I write less detailed comments

13. **Describe your trust in the accuracy of the dogs' information since the app was introduced.**
Mark only one oval.

- I have more trust in the information
- I trust the information the same
- I have less trust in the information

14. **Compared to whiteboard, how has the app changed your confidence in taking a dog to the play yard?**
Mark only one oval.

- I feel more confident knowing which dogs can go in the play yard
- My confidence has not changed
- I feel less confident knowing which dogs can go in the play yard

15. **Compared to whiteboard, how has the app changed your confidence in providing appropriate care for each dog?**
Mark only one oval.

- I feel more confident caring for the dogs appropriately
- My confidence has not changed
- I feel less confident caring for the dogs appropriately

16. **Compared to whiteboard, how has the app changed your sense of community with other volunteers?**
Mark only one oval.

- I feel more a part of the community
- I don't feel any differently
- I feel less a part of the community

Skip to question 23.

Staff

17. **How often did you update staff notes on the whiteboard?**
Mark only one oval.

- Always (every day)
- Often (3+ times per week)
- Sometimes (1-2 times per week)
- Rarely (once a month)
- Never

18. **How often do you update staff notes on the app?**
Mark only one oval.

- Always (every day)
- Often (3+ times per week)
- Sometimes (1-2 times per week)
- Rarely (once a month)
- Never

19. **How often do you use the Woods iPhone to update staff notes?**
Mark only one oval.

- Always (every day)
- Often (3+ times per week)
- Sometimes (1-2 times per week)
- Rarely (once a month)
- Never

20. **Indicate how important you think the following information is to volunteers.**
Mark only one oval per row.

	Not at all	Slightly	Moderately	Very	Extremely
Staff notes in the app					
Verbal information from staff					
Volunteer comments					
Dog breed, age, size, photo					

21. **How much do you feel the app helps improve the quality of care the dogs receive?**
Mark only one oval.

- Not at all
- Slightly
- Moderately
- Very
- Extremely

22. **How has the app changed your sense of community with the volunteers?**
Mark only one oval.

- I feel more connected to the volunteers
- I don't feel any differently
- I feel less connected to the volunteers

Skip to question 23.

Additional Feedback

23. **Do you have any suggestions or comments?**

.....

.....

.....

.....

.....