2016

Laker Mobile 2.0: Rewriting GVSU's official mobile app for iOS Project

Camila Peñaloza
Grand Valley State University

Roland Heusser
Grand Valley State University

Laker Mobile 2.0: Rewriting GVSU's official mobile app for iOS Project

By
Camila Peñaloza, Roland Heusser
April, 2016

Laker Mobile 2.0: Rewriting GVSU's official mobile app for iOS

By
Camila Peñaloza, Roland Heusser

A project submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Computer Information Systems

at

Grand Valley State University

April, 2016

Jonathan Engelsma **04/29/16**

Table of Contents

Abstract

This project seeks to renovate the Laker Mobile iOS app by following the most recent standards of the mobile application development industry. The original Laker Mobile app was released back in 2010 with some updates until 2014. This first version of the app relied mostly on default user interface components, which are provided by the operating system, and it was developed fully in Objective-C.

Since the introduction of Swift 1.0 by Apple in 2014 the language gained wide acceptance. With this project Laker Mobile follows this trend by making Swift its main language. The app has been entirely rewritten from ground up and besides the exception of a few frameworks that are still in Objective-C it only uses Swift 2.x.

Additional to this, Laker Mobile features a complete redesign of the user interface and user experience in order to be more appealing to its target audience. As a result, the application should have an increased usability and retention of users than the previous version.

Introduction

The Mobile Applications and Services Lab (MASL) created an iOS application for GVSU for students in 2010. The application serves as a data aggregator of information around GVSU and while it also officially represents GVSU in the app store. The purpose of this application is to allow GVSU students to:

- stay informed about the latest campus news
- allows them to look up contact information of other fellow students, faculty, staff or departments
- take a look at recent media like pictures, videos, radio and TV
- keep up with GVSU's Twitter feed
- get an interactive overview of the campus maps and bus schedule

Starting in July 2013 Laker Mobile incorporated tracking using Google Analytics. Over time the analytics can be used to retrieve interesting data, such as that in the past 3 years the app has been used by over 17 thousand students (Figure 1). The latest version of Laker Mobile, before this project, was released in July 2014 – version 1.6.

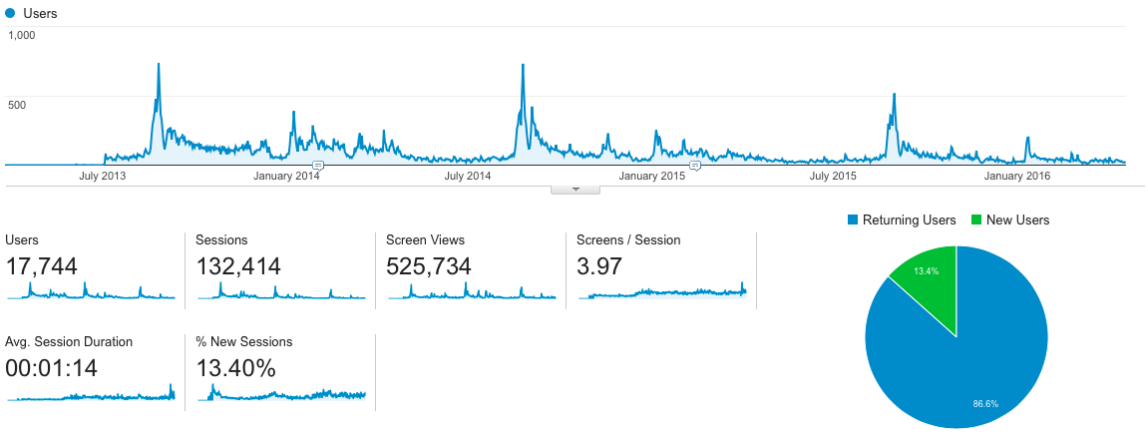

Figure 1 - Google Analytics of Laker Mobile 1.6 (July 2013 - April 2016)

The application has quite aged a bit and in order to continue serving many GVSU students its needed to be improved and maintained. Some of its features even stopped working. Since this application is a project that belongs to the Mobile Application and Services Lab, MASL (1), we

were very interested in improving it. Our main goal for this project are outlined in the following objectives:

Objectives of this project

- Refresh and improve the look and feel of Laker Mobile with the latest user experience and user interface practices.
- Review functionality from version 1.6 and create an updated set of features.
- Write a new code-base for Laker Mobile in the new programming language Swift 2.

Project

This project started with Agile in mind. A backlog of tasks that were required for a new Laker Mobile was created. In a next step the feature-set for Laker Mobile 2.0 was defined and the tasks were scheduled accordingly. After every iteration the prototype was reviewed, the plan and tasks were checked and rescheduled if needed. TestFlight played an important part for build distribution to test users and gathering feedback.

Laker Mobile 1.6

The project started with analyzing the previous version of Laker Mobile, the version 1.6. Following existing features have been identified:

Campus News	*news, including RSS feeds*
Campus Directory	*find student, faculty and staff*
Media	*social media pictures, videos, radio, tv, live cams*
Twitter	*@GVSU Twitter-feed*
Campus Bus Map	*live-information about bus GPS info and schedule*
Helpful numbers	*2020-desk, main switchboard and more*
Library	*lab availability, library usage (iBeacon)*
Events	*event calendar of a large amount of departments and groups*
Bookstore	*embedded web-view of university bookstore*
Analytics	*Google Analytics integration*

The user interface for 1.6 is old and needs a complete overhaul. Furthermore, some of the above listed features stopped working or do not work reliably anymore. An example of such an API that

no longer reliably worked was the one used for the bus service - *the RideTheRapid-API* – it has been changed and the application needed to be adapted to make use of the new API.

Desired features of Laker Mobile 2.0

After reviewing Laker Mobile 1.6, a listed with desired features has been created:

New Design	*a new fresh, more usable and more pleasing interface*
Campus News	*better integration of news-feeds*
Events	*better events integration - maybe even add them to calendar*
Campus Cams	*make the "most popular feature" of 1.6 more accessible*
Social Stream	*integrate with the latest social media streams of GVSU like Facebook, Twitter, YouTube, Instagram, Flickr, Pinterest Show images, videos and more. Add #gvsu etc.*
Maps	*integrate points of interest (Café, Food, Parking, Buildings)*
Bus	*create complete new, working user experience for bus service*
People Finder	*update Campus Directory to People Finder API and UI refresh*
Useful links	*integrate useful links around GVSU*

More experimental ideas

Search API	*make Laker Mobile searchable, increase usage / usability*
Apple Watch	*integrate Apple Watch to show news / bus departures*
Apple TV	*create TV app to make usage of the big screen*
Campus Dining	*integrate campus daily menus*
Room booking	*book a room in the library*
Athletics	*integrate with athletics website: show scores, streams etc.*
Dynamic Marketing	*space reserved for institutional marketing content*
Q&A	*integrate a crowd sourced Questions & Answers system into LM*

Plan

In order to organize the project, we used Trello (2) to plan all the features and organize them into Laker-Mobile versions, creating a roadmap. We created cards, that would define a feature each of the team-members could work on – mostly independently. Since we collaborated on the creation and assignment of these cards it was pretty clear what these cards entailed, but by

working co-located on this project it was very easy to briefly discuss questions that may arise at any given point in time during the implementation of such a card. Each card entailed following steps:

Review feature requirements

Review UI/UX design and carry out adjustments

Carry out implementation

Review implementation

In some exceptional cases, where the structure or a feature or an implementation was unclear we worked on entire features together, did variations of pair-programming or reviewed problems and possible solutions in collaboration.

Influence of analytical information

The analytical information of the previous Laker Mobile also helped planning this project, since it helped finding the most important aspects of the current application and the least used features. By using the integrated Google Analytics, we observed that most of the users are using very few features of the app for example and that the most popular feature are the campus live-cameras. Hence it was clear, that the new Laker Mobile should continue to integrate this feature and might even improve that particular feature.

User Interface Design and User Experience

One of the main motivations for taking on this project was the fact that, in the last few years, there had not been a user interface update of the previous version of Laker Mobile. The mobile applications environment moves very fast and it constantly changes. Mobile applications have evolved over the past few years, and existing app versions often fall below the users' growing expectations.

The initial approach to come up with a redesign of the user interface for Laker mobile was to identify mobile applications from different schools in the area to see what they had done and what their applications had to offer. Then, we went through a series of brainstorming sessions in the Mobile Applications and Services Lab, in order to select all the features (from *Desired features of Laker Mobile 2.0, page 7*) we wanted, and the different UI elements or ideas that we found appealing and attractive. The next step was start designing from scratch and begin making the

decisions of how much of the user interface would be customized and how much would be done using standard iOS components outlined in the iOS Human Interface Guidelines (3).

iOS Standard elements vs. Customized UI

One big challenge faced during the user interface ideation, was finding a middle-ground between default iOS user interface components and customized ones. Since one of our main goals was to make the app more attractive and engaging to college students, we tried to take full advantage of the iOS elements while we also introduced a custom and more "playful", yet functional, side-menu navigation.

Initial design iterations

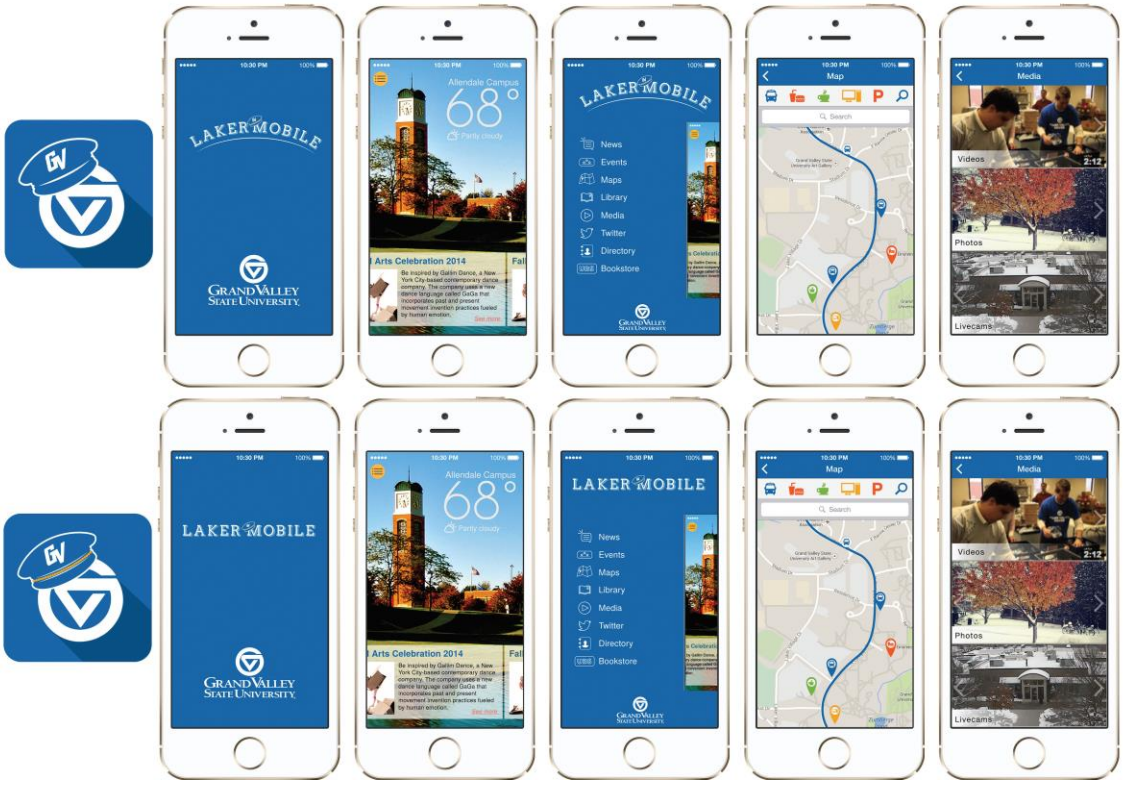

Figure 2 - Initial iteration of user interface design

For this redesign, the main goal was to make Laker Mobile an app that students could relate to and that it is not only useful but also attractive and easy to use. Laker Mobile version 1.6, did a great job at this when it first became available but given the advancements in mobile user

interfaces and the advent of flat design, there was a clear need for a new and more fresh-looking Laker Mobile, this is shown in Figure 3.

Final design:

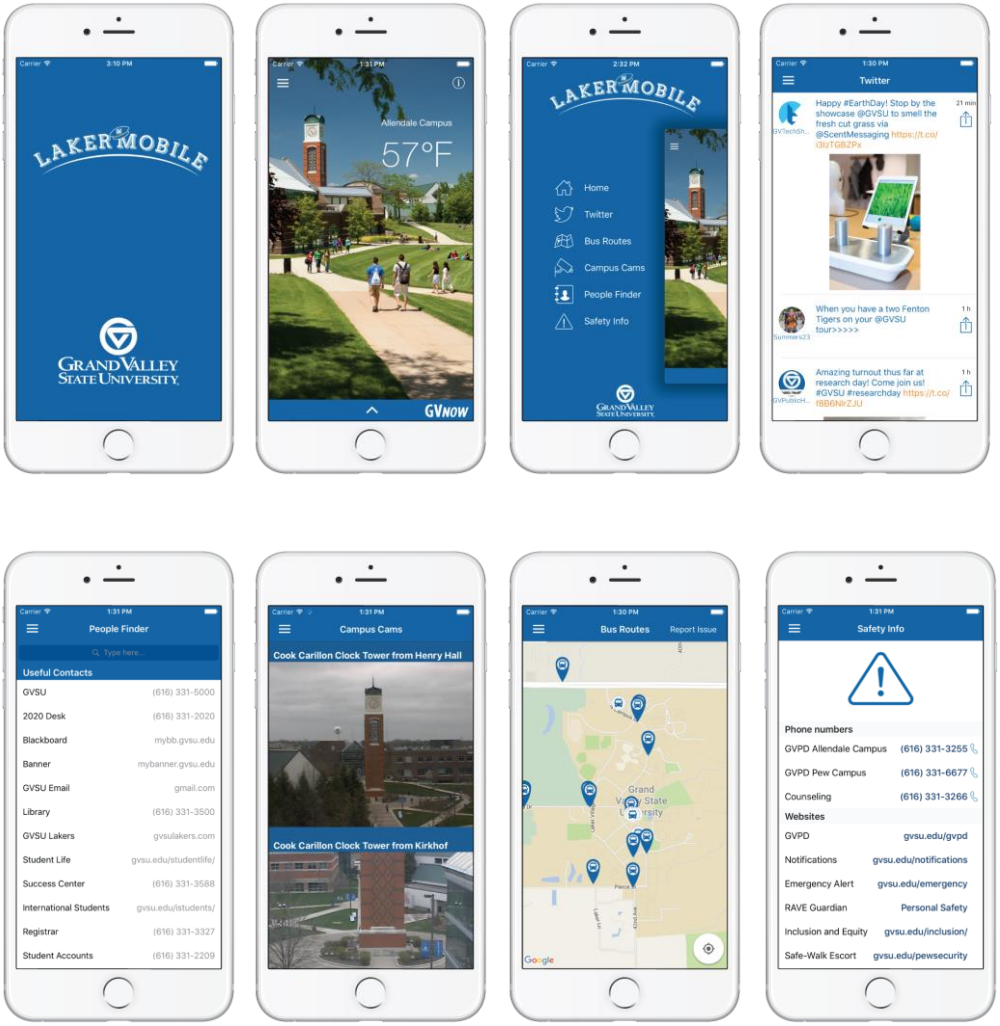

Figure 3 - Final user interface design

Elements of the design

These are some of the key elements that were considered and researched for UI/UX ideation when deciding on the user interface for Laker Mobile 2.0.

Laker Blue

Color was a very important element of the design of the app, and in order to represent the institutional branding and make the app feel familiar to its users, the decision was to use Laker

blue as the main color for the user interface. According to the style guidelines available online (4), Laker blue improves the recall and recognition of the GVSU brand.

Laker Mobile logo

One of the drivers of the general user interface design was coming up with an identity for Laker Mobile. The previous version of the app was very strong in functionality but for its user interface, it relied only in the branding guidelines from GVSU and on the operating system standard UI components. For this version of Laker Mobile, the intention was to develop an identity for Laker Mobile on its own, but always taking GVSU's style guidelines into consideration. The main idea for this logo was to make it playful while also making a reference to one of the main icons of GVSU, the arc at the entrance of the main campus. See Figure 4.

Figure 4 - Laker Mobile logo

App icon

For this element, the idea was to have GVSU institutional branding stand out and improve the proportions between the GVSU mark and the white space around it, while also keeping Laker Blue as the main color, the final icon is shown in Figure 5.

Figure 5 - Laker Mobile app icon

Iconography

For the icons in the app, the approach was to follow the flat design trend that many mobile applications adopted, inspired by Microsoft's metro user interface. The concepts behind flat design are simplicity and readability, this is the reason why the icons are very simple, composed only by outlines, with no shadows or textures.

Another reason to adopt flat design for mobile is the fact that the screen space is usually very limited. Removing complexity of the shapes maximizes the use of the pixels on the screen, an view of the final design of the icons is presented in Figure 6.

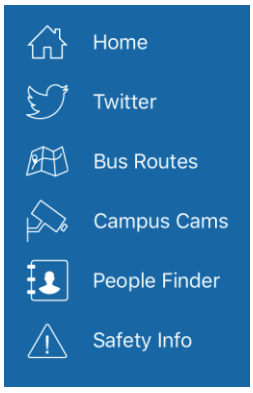

Figure 6 - Laker Mobile icons and navigation

Architecture

Since Laker Mobile was redesigned from ground up, it was not bound to its previous architecture. As an aggregator of multiple data sources it made sense to create an architecture that is heavily influenced resulting in a modular design that can be split into four core parts. The navigation, services & controllers, data sources and data types.

Core areas

The most important classes can be grouped into the four core areas of the Laker Mobile application.

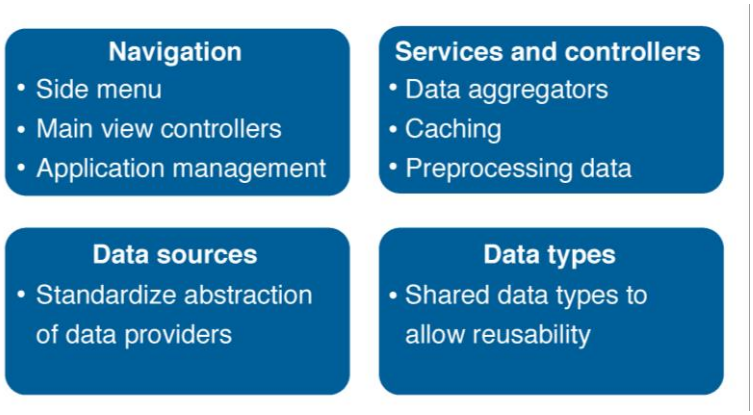

Figure 7 - Core areas of the architecture

Navigation

The purpose of all navigation-related components is to manage what is currently displayed and what should be displayed next. All interaction with the user is managed in this section and very device specific. A centralized controller manages which view will be presented.

Services & Controllers

Most screens require more complex data or capabilities to carry out their tasks. The services & controllers allow reducing a lot of the complexity by creating a user-interface independent collection of services for accessing data, preparing it for usage and more.

Data sources

Every service has a different interface, different API and different way of communicating and exchanging data. In order to make development easier, the data source classes take care of all

provider specific implementations and provide a standardized interface, that – at least application-internally – makes retrieving data very easy.

Note: Since we are pushing the current limitations of Swift Protocols, that don't allow abstract classes (only interfaces), some compromises had to be made here and some data sources have slightly more complex interactions.

Data types

As a further step towards standardization all data sources and services use application internally well-known data types that are inheriting a common "GVData"-Model. This requires data sources to map the data onto these types and makes it very easy for all other classes to deal with data passed to them.

Architecture Overview

The main architecture starts out with the navigation-controllers at the top and how they interact with the services. The services facilitate the interaction with the data sources and further refine the retrieved data for usage. The data-sources abstract the different data providers. Please note that navigation, services, data sources and data types have a common parent in each of their areas from which they inherit from. This allows for the implementation default behavior and reduces the complexity of each other class of that group. Figure 8 and Figure 9 show the architecture including the classes relevant for the data aggregation. Classes such as other *View Controllers* are excluded.

Important in this architecture is that data sources, services and navigation controllers rely on shared *Data Types*. All of the data types, used in Laker Mobile, extend a simple template-data type one called *Data*. This allows the use of shared parent per group as mentioned earlier but still allows passing specific and different data. Figure 10 - Data Types in Laker Mobile 2.0Figure 10 shows the types used in the final version of the application. The types aim to be balanced between being as general as possible but also not making them to abstract for easier usage.

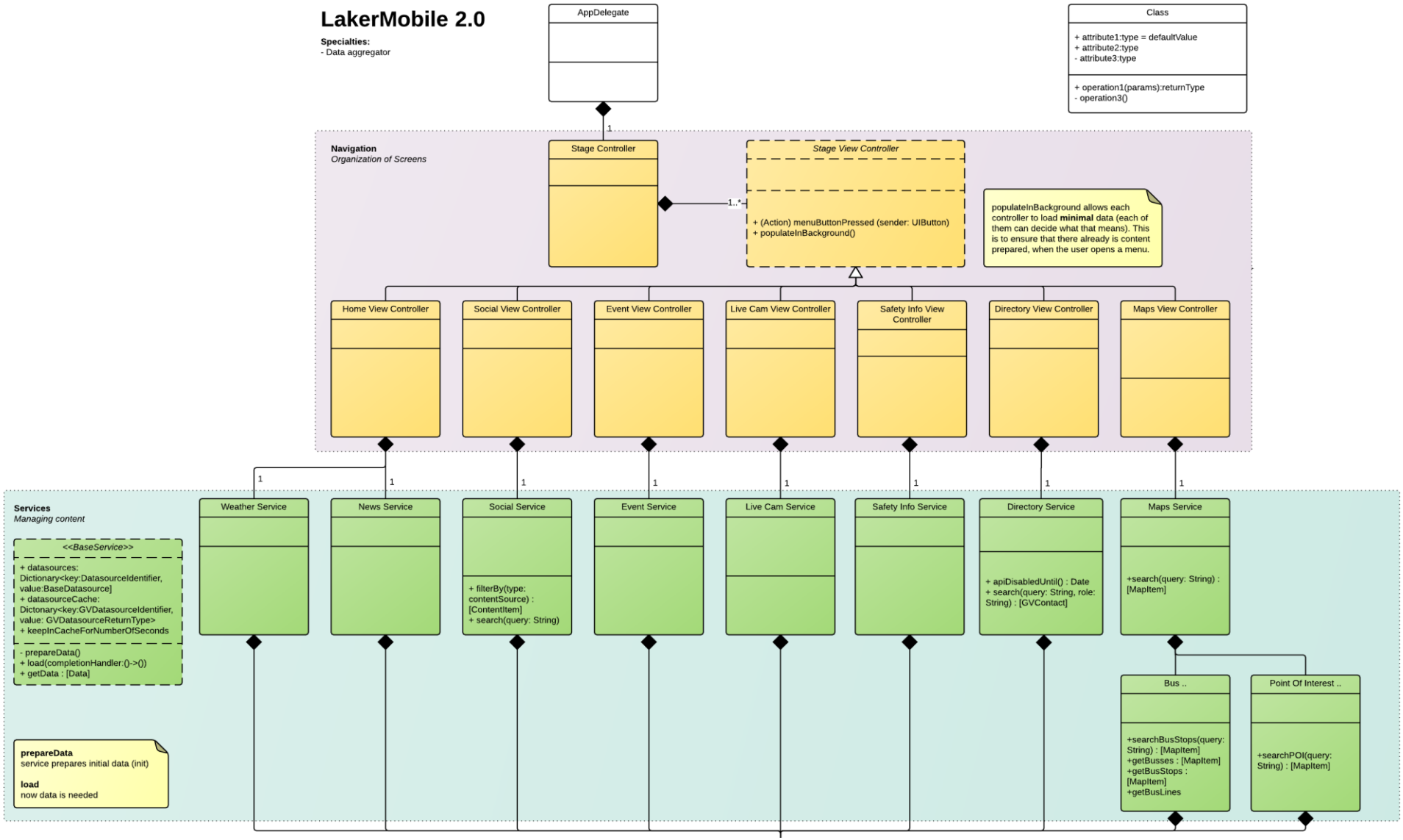

Figure 8 - Laker Mobile Architecture

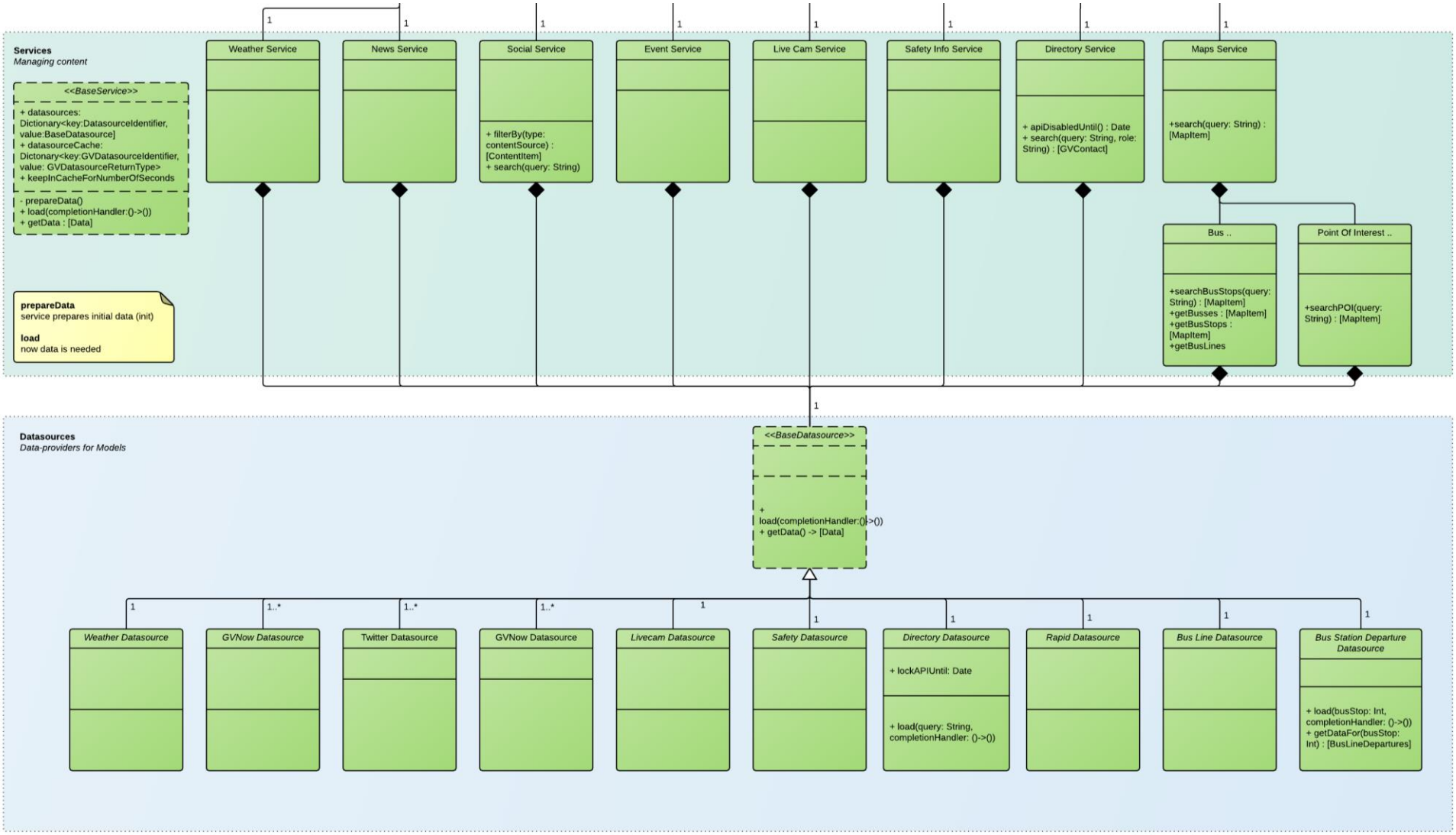

Figure 9 - Laker Mobile Architecture (cont.)

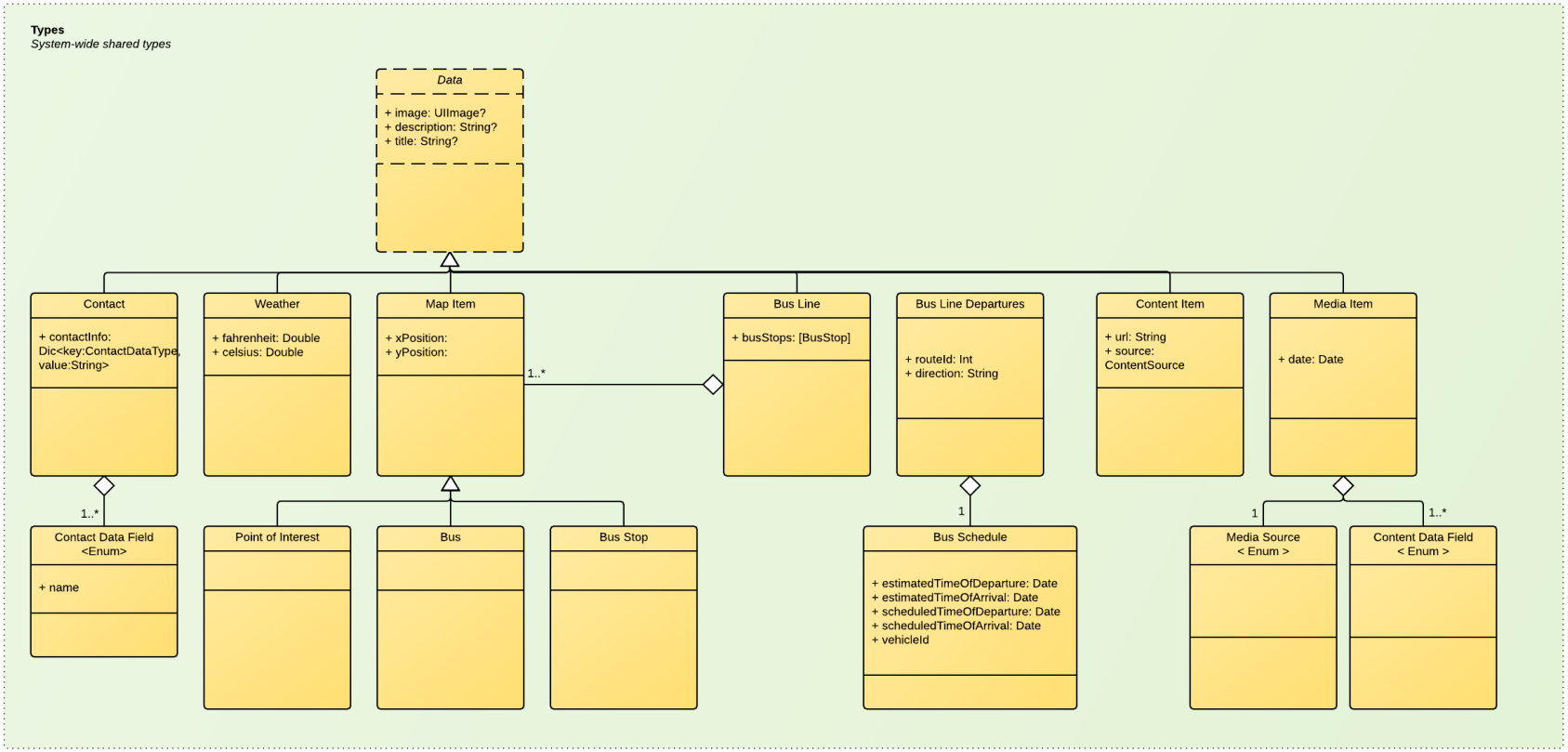

Figure 10 - Data Types in Laker Mobile 2.0

Implementation

The project targeted Apple's latest major iOS platform - version 9.0 – which is used by 84% of all active iOS devices (5). The application is implemented using Apple's standard development environment using the application Xcode. For facilitation and easier customization, we highly relied on the package manager Cocoapods (6). Each package is called a pod and the used ones are listed in

Table 1.

Table 1 - Cocoapods pods used in project

Name	Purpose	URL
KGFloatingDrawer	Side menu	https://git.io/vwMeg (fork of this project)
LKAlertController	Customizable alert popup	https://git.io/vwMeO
GoogleMaps	Maps	https://git.io/vwMeG
Google/Analytics	Collect usage data	https://git.io/vwMeZ
UIImageViewNetwork	Asynchronous image load	https://git.io/vwMen
SwiftyJSON	Simplified JSON parsing	https://git.io/vB3Sq
Alamofire	Simplified networking	https://git.io/OiNxAw
Hex	UIColors from HEX-codes	https://git.io/vwMe4
Parse	MBaaS (to be replaced)	https://git.io/vwMeR
Timepiece	Simplifies date/time usage	https://git.io/fdrScQ
SWXMLHash	Simplifies parsing XML	https://git.io/vwMez

Role of MBaaS

As part of the architecture we considered moving some of the data providing parts onto a Mobile Backend as a Service (MBaaS). This would allow to provide the application with dynamic content, or with temporarily static content, without requiring a new release.

Protocols and Abstract Classes

A problem we faced when implementing the common classes, especially for services and data sources was, that Swift-Protocols do not allow any partial implementations (7), yet. Partial

```
protocol ExampleProtocol {
    func add(input1: Int, input2: Int)
    func multiply(factor1: Int, factor2: Int)
}

extension GVBaseService {
    func add(input1: Int, input2: Int) {
        return input1 + input2
    }
}
```

Figure 11 - Protocol example with extension

implementations, or *abstract classes (Java)* have the advantage to define common methods, instance variables and more, while not allowing an instance of the partial class to be created. It is a mixture between defining how a class should look like and actually defining it. Apple might change Swift in the near future to allow it, but currently it is not possible. In order to get around this problem, an alternative solution is to create a protocol and then "extend" it as shown in Figure 11. This has several limitations. For example, no instance variable can be instantiated or consequently be used - since protocols cannot define them anyways. Protocols to-date only allow the definition of properties that define *computed values* (8), which means, that every class that implements the protocol would have to calculate it with redundant code, at least in our case.

In the final implementation of Laker Mobile 2.0, it was no longer possible to avoid instance variables, so the protocol was changed to describe a class, while retaining the now *not necessary extension* (Figure 12).

```
class GVBaseService {
    var datasources = [GVDatasourceIdentifier:GVBaseDatasource]()
    var datasourceCache = [GVDatasourceIdentifier:GVDatasourceReturnType]()
    var keepInCacheForNumberOfSeconds = 300
}

extension GVBaseService {
    func load(completionHandler:()->()) {
        for (_, datasource) in datasources {
            _ = try? datasource.load(completionHandler)
        }
    }
}
```

Figure 12 - Protocol replacement: Using a class

Results, Evaluation, and Reflection

We are happy that we successfully finished this project and learned a lot.

Release

We successfully released Laker Mobile 2.0.0 on April 19, 2016. The release was followed by a quick follow-up release (2.0.1) on April 21, due to Google Analytics being turned off by accident, even though commits got reviewed carefully. In the future a checklist should be created to avoid this from happening again.

App Store

The iOS app has been released in the app store and can be downloaded using:

https://appsto.re/i6Yb6wy

Feedback & Reviews

The application received positive feedback so far. Users we directly interacted with liked the design and the fresh look. Three reviews on the iTunes App Store highlight the redesign of the application. They like the design and the updated functionality.

> *This is an amazingly helpful app! Thank you to the development team for providing essentials such as bus routes at our fingertips! I very much appreciate having an app available to download onto my phone for quick access. I already love the 2.0 version and find myself using it. Fabulous job! Every school should have this available for their students, faculties and guests!*
> *- Amy Manderscheid, 5 stars, Apr 27, 2016*

> *Not only is Laker Mobile 2.0 a very beautiful update, the real-time bus data is also working. Bonus! :-)*
> *- SpkyDog, 5 stars, Apr 23, 2016*

> *This app is so easy to use and I love that I can see when the bus will be at the bus stop. I also love seeing what the weather is on campus. Such a wonderful update!*
> *- lifetimelaker, 5 stars, Apr 23, 2016*

Critics / Improvements

Even though most of the feedback was positive we received feedback that the hamburger-menu or side-menu was not the best choice for current designs. To be honest, deciding which navigation structure should be used is a difficult decision, and we had serious discussions about this and multiple sessions where we looked into alternatives. At this point *we think* that the hamburger menu using a side-menu is a *widely accepted* and *very popular* navigation structure. Therefore, it should be familiar to most users and not require a lot getting used to. We are open to change that though, if an appropriate alternative is found. What we currently expect to be more important are abundant gesture-support. Therefore, we worked hard on adding them to open or close various screens or menus. There are a few more gestures missing, the most important one of them, is the one to open the side menu. After integrating these swipes, the actual usage of the hamburger-menu might even be less important.

Working in a group of two participants was good. It always helps to be able to discuss questions and issues quickly together and find out how to address a more complex issue. It also allows to do temporary pair-programming. One of the hardest challenges we faced was not to get to enthusiastic about new ideas and wanting to incorporate too many features. We think our tight collaboration helped a lot to prevent this from becoming an issue.

One of the biggest fallbacks was, that initially we planned on doing TDD (Test-Driven Development), but when we started writing the tests we felt like not being able to accomplish all the implementation in time if we would have continued with TDD. Furthermore, we ran into issues on how to write tests or unclear cases of how a certain test should be done in Swift. Looking back that was probably the biggest mistake and we should have continued writing the tests nevertheless, since even the few we wrote helped uncover problems with protocols.

Conclusions and Future Work

Working on Laker Mobile was a very interesting and broad project. We touched a lot of areas from analysis of the current application, ideating new features, creating a design-theme for the new application, planning, implementing, deploying, getting actual usage data and evaluation the results.

There is still a lot left, that needs to be worked on, such as continuing integrating some of the features we were not able to re-implement in Swift. Among them, just to name a few, would be bringing back the full media-section, giving students access to videos, images, radio and the GVTV-

channel. But there are also a lot of small improvements that need to be done – as small as allowing users to save contacts into their address book, have the app cache more data or make use from technologies like Apple's CloudKit.

It was a great pleasure to work on an app that will serve current and future GVSU-students and we are honored by working on this app. We hope that the new version will help them in their everyday life at Grand Valley State University.

References

Credits

We would like to thank Jonathan Engelsma for mentoring us, all the previous work he did on Laker Mobile, helping us communicating with the university and everything else he did. Furthermore, we would like thank for the support and testing of our fellow students in the MASL-lab (also known as the *MASLers*). A thank you also to faculty, staff, students and Alumni of GVSU that helped us testing the app and that gave us feedback. Lastly a big thanks also to the online community on Stackoverflow and others, that facilitated a lot of our work by posting questions and answers.

Tools

Here is a list of the most important tools we used for creating Laker Mobile 2.0.

Table 2 - Tools used to create Laker Mobile

Name	Company	URL
Illustrator	Adobe	http://adobe.com/illustrator
Xcode	Apple	https://developer.apple.com/xcode
Trello	Trello	https://trello.com
Lucidchart	Lucid Software	https://lucidchart.com
Cocoapods	Cocoapods	http://cocoapods.org
Sketch	Bohemian Coding	http://sketchapp.com
SourceTree	Atlassian	https://sourcetreeapp.com
iTunesConnect / TestFlight	Apple	https://itunesconnect.apple.com
LaunchKit	LaunchKit	https://launchkit.io

Figures

Tables

Bibliography

1. **MASL.** Mobile Applications and Services Lab. [Online] GVSU, April 2016. [Cited: April 29, 2016.] http://masl.cis.gvsu.edu/.
2. **Trello.** Trello. [Online] Trello, Inc., 2016. [Cited: April 29, 2016.] https://www.trello.com.
3. **Apple.** iOS Human Interface Guidelines. [Online] Apple. [Cited: 4 29, 2016.] https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/.
4. **Institutional Marketing.** GVSU Identity. [Online] February 2, 2016. [Cited: April 29, 2016.] https://www.gvsu.edu/identity/.
5. **Apple.** App Store - Support. [Online] Apple Inc, April 18, 2016. [Cited: April 29, 2016.] https://developer.apple.com/support/app-store/.
6. **CocoaPods.** CococaPods Dependency Manager. [Online] CocoaPods, 2016. [Cited: April 29, 2016.] https://cocoapods.org/.
7. **Scrève, David.** (SE-0026) Abstract classes and methods. [Online] Apple Inc, April 18, 2016. [Cited: April 29, 2016.] https://github.com/apple/swift-evolution/blob/master/proposals/0026-abstract-classes-and-methods.md.
8. **Apple Inc.** The Swift Programming Language (Swift 2.2) - Protocols. [Online] Apple Inc, March 21, 2016. [Cited: April 29, 2016.] https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Protocols.html.
9. **Lucidchart.** Flow Chart Maker & Online Diagram Software. [Online] Lucid Software Inc., 2016. [Cited: April 29, 2016.] https://www.lucidchart.com/.