Assemble: an iOS App for Simple Group Attendance Tracking

Ryan Huebner
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/cistechlib

Recommended Citation
http://scholarworks.gvsu.edu/cistechlib/235

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.
Assemble: An iOS App for Simple Group Attendance

By
Ryan K. Huebner
April, 2016
Assemble: an iOS App for Simple Group Attendance Tracking

By
Ryan K. Huebner

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
Table of Contents

Abstract ................................................................................................................................. 4
Introduction .......................................................................................................................... 4
Background and Related Work ............................................................................................ 5
Program Requirements ......................................................................................................... 7
Implementation .................................................................................................................... 7
Results, Evaluation, and Reflection .................................................................................... 13
Conclusions and Future Work ............................................................................................ 14
Bibliography ....................................................................................................................... 15
Appendices .......................................................................................................................... 15
Abstract

We all belong to groups of some sort. Some of these groups exist solely online, while others bring us out into the physical realm, co-locating with other members. For those that are primarily online, we have tools like social networking, email groups, and online forums. However, for groups that require physical meetings, these tools often do not fit the bill. This is especially true for groups that care about attendance at each meeting.

Assemble is an iOS app that was developed to fit into this niche. It allows for the quick creation of a group. All that is needed is a group name. Within a group, the owner of the group can create sessions, which can easily be joined by others. Upon joining a session, the user becomes a member of the group and can see when subsequent sessions take place, able to join on each occasion. The owner of the group can see all of the members and additionally the attendance for each session.

The goal of this project was to make the taking of attendance as quick and painless as possible for everyone involved. This allows for the group to focus on the purpose of their meeting, instead of worrying about signing in or calling roll. This paper will detail and reflect on the implementation of the project, examine other tools that are used in this space, and discuss future plans for the Assemble app.

Introduction

We are by nature social creatures, interacting with others is a fundamental part of our existence. Historically, this has meant that for various reasons we gathered with others, co-located to achieve some end. As time has passed, technology has changed this dynamic. At first this change was gradual, public forums gave way to newspapers, plays to television. Recently however, the pace of this change has quickened. The Internet has caused a fundamental change in the social dynamic. We can now accomplish a multitude of things that previously required co-location without ever encountering another human being.

As a result of this, we should not be surprised to find that many Internet-based tools focus on social interactions as being performed remotely. We can edit documents together, have discussions, play games, attend conferences, all from the comfort of our home or office. Often these tools are more focused on the distribution of content or the sharing of an experience, and not the simple act of taking attendance. This basic task is often excluded, or a buried functionality of such applications.

The concept for this project was to create an application that focused on taking attendance as the primary function, and to make it incredibly simple to accomplish. The goal was to create an iOS application in which a user can quickly create a group and allow other users to join in sessions for which attendance could be recorded. The idea was that the onus of recording attendance should be distributed across the entire group. The group owner should not add each individual into the group, and they should not mark
attendance for each session. Instead, users should join the group and note their own attendance in any given session. As a result of multiple people interacting with a group’s data, the application required the use of a cloud based storage solution.

**Background and Related Work**

There are several existing applications, across various platforms, that contain the concept of group attendance within them. Some of them have a narrow focus, on classroom attendance for instance. While others take a broader approach, like Facebook does with its groups. When looking at the current offerings, two categories were formed. First, social networks. Specifically, Facebook was considered as it contains support for creating both groups and events. Then, applications in both the iOS App Store and the Google Play Store were considered.

**Facebook**

Looking into Facebook was the first step taken. It contains the concepts of groups and events. These can both be created independent of each other, but it is also possible to create events within a group. An obvious positive to using Facebook to keep track of your group is the massive user base. In their 2015 *Fourth Quarter and Full Year Results*, Facebook reported 1.59 billion monthly active users [1]. If a group leader chose to use Facebook to track their groups, it is highly likely that some of those people joining would already be users of Facebook. This means that some, if not all, of the group members could avoid an additional username/password combo to remember. Facebook also shows attendance for its events as depicted below.

![Facebook Event Page](image1)

![Event Page Attendance (enlarged)](image2)

However, these images also point to a great weakness of using Facebook to track group attendance, it is far from the primary focus of the application. There is also no control over when a member declares their attendance. On top of this, there is no apparent way to view attendance statistics across multiple events. Which means it would have to be done manually. All of these reasons combined really eliminate Facebook as an efficient way of tracking group attendance.
Kinvo

Kinvo (formerly Kinvolved) created by Kinvolved, Inc. is an iOS and Android application that is focused on primary and secondary education settings. The iOS App Store page describes it as an “app [that] offers educators, administrators, and youth program leaders an efficient, instant communication tool so that they can communicate with all stakeholders in a student’s life with just a click”.

Other than the tighter focus of Kinvo, it served as a model for what this project aimed to achieve. It appears to have a simple and usable design. Its primary function is taking attendance, and it allows for the tracking of attendance across sessions. Unfortunately, it was not possible to directly interact with the application as credentials from a participating school are required to get past the login screen.

Another way in which this project varied from Kinvo’s model was that taking attendance in Kinvo is a function solely fulfilled by the teacher or program leader, with students not having access to the application at all. This is not a weakness of Kinvo, it is a smart decision for the intended use case. The application is designed to deter truancy and absenteeism in young students. It seems unlikely that it would achieve the desired results if the students were the one’s deciding if a text message or phone call would be made to their parents about an incident. That being said, the idea behind Assemble is to simplify the attendance taking process by having the whole group participate. Because of this, Kinvo does not quite fit the desired use case for this project, but is a great blueprint overall.

Top Hat Lecture

Top Hat Lecture by Top Hat Corporation is a multi-platform application that goes beyond taking group attendance and provides a rich array of features for teaching in the college classroom. Achieving something on this level was outside of the scope of this project. However, it does provide an attendance feature and some of the other features are considered future possibilities for Assemble. This made it a valuable application to review.

The attendance module in Top Hat Lecture is a very simple way to take attendance. The professor presses the attendance button for the given course and receives a four-digit code that (s)he then conveys to students
in the class. Students type in the code to register their attendance. This is very similar to how taking attendance in Assemble is accomplished, spreading the responsibility across the group.

Beyond attendance, Top Hat Lecture includes a variety of other tools for professors to utilize. The feature that was most interesting and a possible future feature addition to Assemble, is the ability to quiz students during a class session. Even though Assemble aims at a broader audience than just classrooms, this feature still seems valuable. Polling any group for their opinions can provide valuable feedback to the groups leader.

**Program Requirements**

This project aimed to create a simple and efficient user experience for taking attendance in any group setting. To accomplish this overall goal, there were a number of requirements that existed. Those requirements will be divided into three categories. The application has two defined user roles, and so the requirements for each role will be listed separately. The third category will be for general requirements related to user interface and backend design.

**Group Owner**

- Creating a group must be a simple process.
- Groups must be able to have multiple sessions.
- Each session must keep track of members in attendance.
- Owners must be able to restrict access to their groups.
- Owners must be able to remove members from groups and sessions.
- Owners must be able to post a message to group members.

**Group Member**

- Members must be able to easily join a group.
- Members must be able to add themselves to a session attendance list.
- Members must be able to view the group’s message.
- Members must be able to leave a group.

**General Requirements**

- The application must use cloud-based storage to allow multiple users to interact with a group.
- The user interface must notify users of state changes caused by their actions.
- Where possible, design decisions should support future plans for the app.

**Implementation**

**Planning: Architecture**
As soon as the type of application for the project was decided upon, it was apparent that it would require the use of cloud-based storage. With that decision made, the first steps taken were to do some basic entity relationship models for the data. These were reworked a few times before they seemed to fit the needs of the project. While this was a good exercise to start thinking about the design of the project, the models ended up being reworked as the result of a change to the initial plan for cloud storage.

One of the biggest hurdles to overcome during the development of this project happened as the project was just getting off the ground. On January 28, the cloud service provider Parse sent out an email to their user base notifying them that the service would be fully retired one year from that day. This presented an interesting dilemma. The initial plan coming into the project had been to leverage existing knowledge of the Parse framework for iOS to accomplish the requirement of having a cloud-based data store. This was still possible, as the service would still be online as of the project completion date. However, taking this path ran counter to one of the project’s goals; to prepare for the future wherever possible. The downside to not using Parse was that time would have to be spent identifying possible replacements and learning a new framework.

The decision was made to abandon the initial plan and seek another solution for the applications backend. In the end, Firebase was selected as a replacement. The next few weeks were spent learning the Firebase iOS SDK. As mentioned above, this meant that some of the original data models had to be tweaked as Firebase does not have native support for arrays. Some of the data was also denormalized to avoid making additional network calls and improve performance. For instance, members for each group are stored by the unique key assigned to their profile when they create an account. One way to handle getting their name and email address to display in the group would have been to retrieve the user object for each member. This is not an issue for a small group, but what if someone ended up using Assemble to track attendees of a large conference? This could result in a great deal of additional network traffic, and on top of that the user object contains a lot more data than is required. As a result, the user’s first name, last name, and email address are redundantly stored within the group object.

**Planning: User Interface**

The next phase of planning was to do some brainstorming on the user interface design. This phase started with some very basic pencil and paper sketches. But, before proper time and consideration were given to designing the user interface, the urge to get started had set in and coding commenced. It was not long before the error of this decision was made apparent as it felt with each passing day that the lack of an overall plan for the design was slowing progress and reducing the cohesive feel of the application.

Once this realization was made, coding stopped and the work of creating mock-ups of screen designs started. When thinking about the visual design of Assemble, an important consideration was that if it were
ever launched, it would have to be on both the Android and iOS platforms. Desiring visual consistency across both platforms, it was decided that Google’s Material Design would be the inspiration for the look and feel of Assemble. The mock-ups below were created with that in mind.

Eventually some of the Android elements were removed in favor of elements that felt more appropriate for the iOS platform. One example of this is the removal of the floating action button from the main screen, which was eventually replaced with a popover menu anchored to the navigation bar. This decision was made with the thought that using too many Android elements in the user interface may turn off some long-time iOS users.

Logging In and Signing Up
The log in and sign up screens for Assemble were custom designs. Firebase does offer sign up and log in screens through an additional SDK, but they did not fit with the desired appearance of the application and designing the appearance and logic behind the screens was a good way to build momentum for the project. Other than the two text labels on the sign in screen, which are stock iOS components, both pages use components from a third party library called Material by CosmicMind. The library provides various components and colors inspired by Google’s Material Design. The text fields show labels over the field when the user begins inputting data (see Figure 6). Error notifications show up below the text field in a red color as displayed in Figure 5.
Although the appearance and logic were custom, the application does utilize the authentication service provided by Firebase. Both pages send the data provided by the user to Firebase and handle the response accordingly, by either populating an error message below a text field or authenticating the user and performing a segue to the groups screen.

Displaying the User’s Groups

Upon a successful login, all users are taken to the groups screen. This screen was built using a combination of stock iOS components and components from the Material library. A combination of storyboards and programmatic implementation were used.

The underlying view controller is a standard iOS UICollectionViewController embedded in a UINavigationController. The buttons in the navigation controller are directly from Google’s Material Design icons repository on GitHub. Each button displays a popover menu when pressed. These two buttons replaced a floating action button menu that was originally placed in the bottom right corner of the screen. As mentioned previously, this is one Android element that was removed in favor of an iOS element. The popovers were both implemented using storyboards with freeform sizing and have delegate protocols which notify the main screen of the option selected.
The collection view uses a custom collection view cell which contains an ImageCardView provided by the Material library. Only the very basic setup for the cell was done in the storyboard for this screen. The rest of the layout was accomplished programmatically in the custom cell class. Assemble does not currently support images for groups or users, but the decision was made to use the ImageCardView instead of the CardView based on future intentions to incorporate this feature. To view a group, a user can select the group by pressing on the teal name box on the group’s card. This causes one of two segues depending on whether the user is the owner of the group or a member.

Creating and Joining Groups
One of the goals of the project was to make creating and joining groups as simple as possible. To do so, two screens were created that require a minimum of one text input and the press of a button. Both screens use a combination of iOS and Material components to achieve their visual appearance.

The underlying view controller for the create view screen is a UITableViewController and uses static cells in its layout. The table view simplifies layout and makes it easy to add additional options in the future if it is necessary. In addition to the text field and button, there are three options on the create group screen, which are set to off by default. The “require code for each session” option is intended to require users to input a code even if they are already a member of the group. The default behavior only requires a code when joining for the first time. The other two options, were intended to provide the ability to hide personal details of group members from other group members for the sake of privacy, but not from the group’s owner. However, as of this writing, these three features are not implemented however. Having a code for each session is an important feature that needs to be implemented, and will be a focus before the project is submitted. The issue with the other two options is that there is question as to whether the group members need to be able to see each other at all. These options may be removed until there is a need for group members to see each other, such as the ability to send messages or generate an email from within the application.

The join group screen was built using a basic UIViewController because of the simplicity of the action. Joining a group consists of entering a five-digit code that is provided to the group leader every time a session is started. These codes are removed from the group at the end of each session, making it impossible to join a group without the group’s owner activating a session.
Viewing a Group as the Owner

As mentioned in the discussion of the groups screen, selecting a group will bring the user to one of two places. If the user is the owner of the selected group, the application will segue to the group owner’s screens. There are three separate screens related to the group which are all embedded in a UITabBarController, as well as the UINavigationController within which the rest of the application is embedded. The tab bar buttons were also taken from Google’s Material Design icons repository on GitHub to provide a consistent appearance throughout the application. One interesting thing learned about performing a segue into a UITabBarController from a screen that is not a member of that collection is that the data for each screen can be passed in the prepare for segue method by referring to each screen controlled by the tab bar through array index notation.

The initial screen is the group’s home screen (Figure 9) which provides the owner with some statistics about the group, the ability to set the group message, and also the ability to start and end sessions. The underlying view controller is a UITableViewController that utilizes static cells. The group message field is a TextView from the Material library, which allows for the addition of a styled cursor and label when the field is selected to notify the user the field is active. This also matches with the text fields throughout the application to provide a cohesive style. The buttons are RaisedButtons also from the Material library. This provides a nice pulse and animation on the buttons when they are pressed. The start/end session button changes its text and color based on the group’s state, giving a clear indication to the group’s owner if a session is active or not. When a session is active, the Attendees button is made visible and can be pressed to view current attendees of the active session.

The members and sessions screens were both built on UITableViewControllers using dynamic cell types. The members screen (Figure 10), displays the name and email address of each member in the group. The sessions screen (not pictured) displays the date of each session as a label on the left of each cell and the percentage of members in attendance on the right of each cell. Originally, the right label was displayed as a ratio with attendees
over group members. This was changed so that as group memberships fluctuate, the owner can still get a quick sense of attendance over time. In retrospect, the being able to see increases in group membership seems just as important and this should probably be reverted to the original design, or a combination of the two.

**Viewing a Group as a Member**

As opposed to the collection of screens provided to a group’s owner, a group’s members only see a single screen. The view displays the group’s message, the group owner’s name and email address, the number of members in the group, and a button to join a session. The “session in progress” label and join session button are only displayed if a session is active. The Material TextView that displays the groups message is not selectable from the group members view. The join session button is a Material RaisedButton providing the same look and feel as the other screens. The underlying view controller is a UITableViewController that utilizes static cells.

**Results, Evaluation, and Reflection**

This project met all of the goals it set out to achieve, barring one; which was to allow for members to leave a group. This will not be a difficult thing to implement, time simply ran short and focus needed to shift from implementing the application to preparing this paper and the presentation materials for the project. There are other outstanding issues that need to be resolved; like implementing the options on the group creation screen, but they do not prevent the primary functions of the application from being performed.

The primary use cases for group owner’s and members have been implemented effectively. The desire to keep these tasks as simple as possible was achieved. Users can create a group and start allowing members to join through an active session quickly. Joining a group is also a simple task, only requiring the five-digit code provided to the group’s owner upon creating a session.

The overall look and feel of the application is a strong point. The design is simple and usable while still achieving an appealing appearance. The use of the Material library greatly aided in achieving the requirement of keeping the user aware of the program’s state with the ability to apply error message to text fields. That being said, this library also provided another important lesson.

The Material library is currently being very actively developed. During the course of this project there were numerous versions released. On one hand, this was great because some of those versions introduced
features that were helpful in the implementation of this project. On the other hand, updating to the new version of the library often broke certain parts of the project. Eventually, the decision was made to stop updating to the latest version to avoid breaking existing features within the application.

Something that was not done as efficiently as it could have been was the use of the Firebase library to update data in a more real-time fashion. Much of the data within Assemble is passed through segues as opposed to being retrieved through open connections to the Firebase data store. In some instances, this is desirable. For instance, there is no need to keep an open listener to a user’s groups list because groups can only be added through actions performed by the user. In other cases, such as when viewing the attendees list for an active session, having a listener open and adding members as each one joins would be desirable. Unfortunately, that is not how Assemble is currently handling the attendees list. It loads the session’s attendees list based on the current value when the screen appears. This is the result of a misunderstanding of the way the “child added” function works in the Firebase framework.

Conclusions and Future Work
A personal goal for Assemble was to launch it in the iOS App Store at the conclusion of the project. Unfortunately, that will not be happening any time soon. There remains a great deal of work left to be done before it reaches that point. Some of that work is on existing bugs, while some is related to enhancing features. There are also expansions that are planned to the current feature offering. And, of course, the greatest hurdle for launching Assemble is that it must be available on more platforms before it is usable in any group.

Besides the issues that are listed throughout the description of the project above, there are also some issues with the log in error states as well. It is believed that these are related to a release of the Material library and that they went unnoticed after updating, but further investigation is required to truly determine the cause.

As for the enhancements of current features, one thing that would be beneficial to have would be location-based attendance confirmation. This could be accomplished either using GPS coordinates or by using one of the beacon technologies that are available. The idea would be that a user could not join a group without being in some predefined range of the session. Once this was implemented, members of a group could also automatically join a session they were in range of, further simplifying the attendance process.

Looking ahead to expanding the feature offering, a few things come to mind. First, allowing for the exchanging of messages amongst members of the group. Because Assemble already utilizes Firebase as a backend, adding on a real-time chat system would be a possibility. Additionally, the ability to poll members of a group for opinions (or quiz members in the case of classroom groups) would be a great way
to allow group owners to easily get feedback from their group. Finally, allowing users to sign up and log in through other services such as Facebook or Google would provide relief for people who do not want another username and password combination to remember.

As stated above, the greatest hurdle for Assemble is being available to a wider variety of people. While an iOS first approach works great for many applications, it would greatly hinder the function Assemble. In any group of people, only a subset of them are iOS users and only they would be able to interact with the group. Another possibility would be to add the ability for the group owner to manually add members to groups and sessions, but this goes against the intention of the application to distribute the attendance process. The only reasonable solution is to make the application available on more platforms.

When initially considering the idea of reaching more platforms, the first thing that came to mind was to implement an Android version as well. This is a reasonable solution to the problem, as it would reach the majority of people if it was on the two major mobile platforms. There would still be occasional outliers of course, but overall it would probably work. The better solution however, may be to implement a responsive web application. Doing so would reach anyone who had access to the Internet. It would also be helpful when implementing the member polling feature in the future, as asking group owners to create a multiple part questionnaire with a mobile device may be off-putting.

Bibliography

Appendices

Source Control
Bitbucket [ https://bitbucket.org/ ]

Libraries
Material by CosmicMind [ http://www.cosmicmind.io/material ]
Material-design-icons by Google [ https://github.com/google/material-design-icons ]
SwiftSpinner [ https://cocoapods.org/pods/SwiftSpinner ]
Firebase [ https://www.firebase.com/ ]