

---

# Crowd Control

---

## Senior Design Final Documentation

### Bowtaps

Johnathan Ackerman

Daniel Andrus  
Joseph Mowry

Charles Bonn

Evan Hammer

May 2, 2016



---

# Contents

---

<b>Title</b>	<b>i</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Algorithms</b>	<b>ix</b>
<b>Overview Statements</b>	<b>xi</b>
0.1 Mission Statement . . . . .	xi
0.2 Elevator Pitch . . . . .	xi
<b>Document Preparation and Updates</b>	<b>xiii</b>
<b>1 Overview and Concept of Operations</b>	<b>1</b>
1.1 Bowtaps and Its Team . . . . .	1
1.2 Crowd Control . . . . .	1
1.2.1 Purpose of the System . . . . .	1
1.3 Business Need . . . . .	1
1.4 Deliverables . . . . .	2
1.5 System Description . . . . .	2
1.5.1 Integrated Group Messaging . . . . .	2
1.5.2 GPS Location Services . . . . .	3
1.5.3 Group Management Features . . . . .	3
1.5.4 Suggestions . . . . .	3
1.6 System Overview and Diagram . . . . .	3
1.7 Technologies Overview . . . . .	4
1.7.1 Google Play Services . . . . .	4
1.7.2 Apple Map Features . . . . .	4
1.7.3 Parse . . . . .	4
1.7.4 Sinch . . . . .	5
<b>2 User Stories, Requirements, and Product Backlog</b>	<b>7</b>
2.1 Overview . . . . .	7
2.2 User Stories . . . . .	7
2.2.1 User Story #1 . . . . .	7
2.2.2 User Story #2 . . . . .	7
2.2.3 User Story #3 . . . . .	7
2.2.4 User Story #4 . . . . .	7
2.2.5 User Story #5 . . . . .	7
2.2.6 User Story #6 . . . . .	7
2.2.7 User Story #7 . . . . .	7

2.2.8	User Story #8 . . . . .	8
2.2.9	User Story #9 . . . . .	8
2.2.10	User Story #10 . . . . .	8
2.3	Requirements and Design Constraints . . . . .	8
2.3.1	System Requirements . . . . .	8
2.3.2	Network Requirements . . . . .	8
2.3.3	Development Environment Requirements . . . . .	8
2.3.4	Project Management Methodology . . . . .	9
2.4	Specifications . . . . .	9
2.5	Product Backlog . . . . .	9
2.6	Research or Proof of Concept Results . . . . .	9
2.6.1	iOS Proof of Concept Screen Shots . . . . .	9
2.6.2	Android Proof of Concept Screen Shots . . . . .	9
2.7	Supporting Material . . . . .	9
<b>3</b>	<b>Project Overview</b>	<b>17</b>
3.1	Team Member's Roles . . . . .	17
3.2	Project Management Approach . . . . .	18
3.3	Stakeholder Information . . . . .	18
3.3.1	Customer or End User (Product Owner) . . . . .	18
3.3.2	Management or Instructor (Scrum Master) . . . . .	18
3.3.3	Investors . . . . .	18
3.3.4	Developers –Testers . . . . .	18
3.4	Budget . . . . .	18
3.5	Intellectual Property and Licensing . . . . .	18
3.6	Sprint Overview . . . . .	18
3.7	Terminology and Acronyms . . . . .	19
3.8	Sprint Schedule . . . . .	19
3.9	Timeline . . . . .	19
3.10	Backlogs . . . . .	19
3.11	Burndown Charts . . . . .	19
3.12	Development Environment . . . . .	19
3.13	Development IDE and Tools . . . . .	19
3.14	Source Control . . . . .	19
3.15	Dependencies . . . . .	19
3.16	Build Environment . . . . .	19
3.17	Development Machine Setup . . . . .	19
<b>4</b>	<b>Design and Implementation</b>	<b>21</b>
4.1	Architecture and System Design . . . . .	21
4.1.1	Design Selection . . . . .	21
4.1.2	Data Structures and Algorithms . . . . .	22
4.1.3	Data Flow . . . . .	22
4.1.4	Communications . . . . .	22
4.1.5	Classes . . . . .	22
4.1.6	UML . . . . .	25
4.1.7	GUI . . . . .	25
4.1.8	MVC . . . . .	26
4.2	Group Messaging . . . . .	27
4.2.1	Technologies Used . . . . .	27
4.2.2	Component Overview . . . . .	28
4.2.3	Phase Overview . . . . .	28
4.2.4	Architecture Diagram . . . . .	28
4.2.5	Data Flow Diagram . . . . .	29
4.2.6	Design Details . . . . .	29

4.3	Location Tracking . . . . .	29
4.3.1	Technologies Used . . . . .	29
4.3.2	Component Overview . . . . .	29
4.3.3	Phase Overview . . . . .	30
4.3.4	Architecture Diagram . . . . .	30
4.3.5	Data Flow Diagram . . . . .	31
4.3.6	Design Details . . . . .	31
4.4	Group Management . . . . .	31
4.4.1	Technologies Used . . . . .	31
4.4.2	Component Overview . . . . .	31
4.4.3	Phase Overview . . . . .	31
4.4.4	Architecture Diagram . . . . .	31
4.4.5	Data Flow Diagram . . . . .	31
4.4.6	Design Details . . . . .	31
<b>5</b>	<b>System and Unit Testing</b>	<b>33</b>
5.1	Overview . . . . .	33
5.2	Dependencies . . . . .	33
5.3	Test Setup and Execution . . . . .	33
5.4	System Testing . . . . .	33
5.5	System Integration Analysis . . . . .	33
5.6	Risk Analysis . . . . .	33
5.6.1	Risk Mitigation . . . . .	33
5.7	Successes, Issues and Problems . . . . .	33
5.7.1	Changes to the Backlog . . . . .	33
<b>6</b>	<b>Prototypes</b>	<b>35</b>
6.1	Sprint 1 Prototype . . . . .	35
6.1.1	Deliverable . . . . .	35
6.1.2	Backlog . . . . .	35
6.1.3	Success/Fail . . . . .	35
6.2	Sprint 2 Prototype . . . . .	35
6.2.1	Deliverable . . . . .	35
6.2.2	Backlog . . . . .	35
6.2.3	Success/Fail . . . . .	35
6.3	Sprint 3 Prototype . . . . .	35
6.3.1	Deliverable . . . . .	35
6.3.2	Backlog . . . . .	35
6.3.3	Success/Fail . . . . .	35
6.4	Sprint 4 Prototype . . . . .	35
6.4.1	Deliverable . . . . .	35
6.4.2	Backlog . . . . .	35
6.4.3	Success/Fail . . . . .	35
6.5	Sprint 5 Prototype . . . . .	35
6.5.1	Deliverable . . . . .	35
6.5.2	Backlog . . . . .	35
6.5.3	Success/Fail . . . . .	36
<b>7</b>	<b>Release – Setup – Deployment</b>	<b>37</b>
7.1	Deployment Information and Dependencies . . . . .	37
7.2	Setup Information . . . . .	37
7.3	System Versioning Information . . . . .	37

<b>8</b>	<b>User Documentation</b>	<b>39</b>
8.1	User Guide . . . . .	39
8.2	Installation Guide . . . . .	39
8.3	Programmer Manual . . . . .	39
<b>9</b>	<b>Class Index</b>	<b>41</b>
9.1	Class List . . . . .	41
<b>10</b>	<b>Class Documentation</b>	<b>43</b>
10.1	Poly Class Reference . . . . .	43
10.1.1	Constructor & Destructor Documentation . . . . .	43
10.1.2	Member Function Documentation . . . . .	43
<b>11</b>	<b>Business Plan</b>	<b>45</b>
	<b>Bibliography</b>	<b>47</b>
	<b>Software Agreement</b>	<b>SA-1</b>
<b>A</b>	<b>Product Description</b>	<b>A-1</b>
1	GPS Features . . . . .	A-1
1.1	Group Members . . . . .	A-1
1.2	Suggestions . . . . .	A-1
2	Group Messaging . . . . .	A-1
3	Group Manangement Features . . . . .	A-1
4	Parse Features . . . . .	A-1
<b>B</b>	<b>Sprint Reports</b>	<b>B-1</b>
1	Sprint Report #1 . . . . .	B-1
2	Sprint Report #2 . . . . .	B-5
3	Sprint Report #3 . . . . .	B-8
4	Sprint Report Winter Sprint . . . . .	B-11
5	Sprint Report #4 . . . . .	B-15
6	Sprint Report #5 . . . . .	B-20
<b>C</b>	<b>Industrial Experience and Resumes</b>	<b>C-1</b>
1	Resumes . . . . .	C-1
2	ABET: Industrial Experience Reports . . . . .	C-7
2.1	Johnathon Ackerman . . . . .	C-7
2.2	Daniel Andrus . . . . .	C-7
2.3	Charles Bonn . . . . .	C-7
2.4	Evan Hammer . . . . .	C-7
2.5	Joseph Mowry . . . . .	C-7
<b>D</b>	<b>Acknowledgment</b>	<b>D-1</b>
<b>E</b>	<b>Supporting Materials</b>	<b>E-1</b>

---

## List of Figures

---

1.1	Basic System Flow Diagram . . . . .	3
2.1	iOS login select screen . . . . .	10
2.2	iOS email login screen . . . . .	10
2.3	iOS create account screen . . . . .	11
2.4	iOS group information screen . . . . .	11
2.5	iOS map view screen . . . . .	12
2.6	iOS messaging main screen . . . . .	12
2.7	Android login screen . . . . .	13
2.8	Android create group screen . . . . .	13
2.9	Android group information screen . . . . .	14
2.10	Android group join screen . . . . .	14
2.11	Android messaging main screen . . . . .	15
4.1	Early database schema. . . . .	22
4.2	Improved database schema. . . . .	23
4.3	Communication flow diagram. . . . .	24
4.4	Use Case Diagram in UML. . . . .	26
4.5	Model classes in the “model” folder of the project. . . . .	27
4.6	Model classes in the “model” folder of the project. . . . .	28
4.7	Model classes in the “model” folder of the project. . . . .	29
4.8	Messaging data flow diagram. . . . .	30





---

## List of Tables

---



---

## List of Algorithms

---



---

## Overview Statements

---

### 0.1 Mission Statement

Our mission at Bowtaps is to develop innovative mobile software applications to provide solutions to inconveniences that trouble the everyday user. With our software, we plan on changing the mobile environment by creating applications that are easy to use with intuitive interfaces and reliable services for everyday use.

### 0.2 Elevator Pitch

Our company, Bowtaps, is developing an iPhone/Android app to help young adults and event-goers stay in contact with friends while in loud and crowded places using group messaging and GPS features.

Our product, Crowd Control, is designed to become an essential element for groups looking to go out together by providing both powerful group-management tools and interesting nearby outing suggestions, such as local events, concerts, and pub crawls.

We will work with local businesses and event planners to sponsor these suggestions. This will generate content for our users, visibility for our sponsors, and revenue for ourselves.

We plan to release the app for free in early-to-mid summer of 2016.



---

## Document Preparation and Updates

---

Current Version [1.5.3]

*Prepared By:*  
*Johnathan Ackerman*  
*Daniel Andrus*  
*Charles Bonn*  
*Evan Hammer*  
*Joseph Mowry*

### **Revision History**

<b>Date</b>	<b>Author</b>	<b>Version</b>	<b>Comments</b>
<b>10/1/15</b>	Charles Bonn	1.0.0	Sprint 1 & Senior Design Contract
<b>11/3/15</b>	Charles Bonn	1.1.0	Sprint 2 Documentation
<b>12/11/15</b>	Charles Bonn	1.2.0	Sprint 3 finished, résumés added
<b>1/15/16</b>	Daniel Andrus	1.2.1	iOS documentation added
<b>1/18/16</b>	Joseph Mowry	1.3.0	Winter Sprint Report added
<b>2/12/16</b>	Charles Bonn	1.4.0	Sprint 4 Report, general content added
<b>2/18/16</b>	Joseph Mowry	1.5.0	Sprint 5 Report, various chapter revisions
<b>2/19/16</b>	Charles Bonn	1.5.1	Sprint 5 Report, organizational changes, chapter revisions
<b>2/24/16</b>	Johnathan Ackerman	1.5.2	Overview rewrite
<b>3/17/16</b>	Evan Hammer	1.5.3	Document cleanup, organizational changes





## Overview and Concept of Operations

---

### 1.1 Bowtaps and Its Team

Bowtaps is a start up company from Rapid City, SD that began working together at the SDSM&T campus. Our goal is to create easy to use software applications that help ease the everyday life of the user. Their software aims to be both well-constructed and maintainable, and their products are made with sustainability in mind. Bowtaps currently consists of the members Charles Bonn, Johnathan Ackerman, Daniel Andrus, Evan Hammer, and Joesph Mowry. Their roles and experience are further detailed in section C of this document.

### 1.2 Crowd Control

Crowd Control is our flagship product designed and created by Bowtaps. Its goal is to combine GPS tracking, group messaging and group management features into one easy to use application. A primary focus of this product is to be maintainable and modular, so that if better solutions arise, those can be implemented with minimal refactoring.

User information is perhaps our greatest focus in Crowd Control's design; careful measures were taken to assure that sensitive user information is kept safely, vulnerabilities are removed, and risks are mitigated. Before Crowd Control is released to the public, we will implement an encryption scheme called AES-256. Bowtaps also uses third-party services such as Parse and Sinch that guarantee safety in storage and transmission of data through their access points.

Crowd Control, in addition to being written in a maintainable, well-designed manner, is aimed to be a sustainable commercial product. Bowtaps is using what they have gathered from various business accelerators and entrepreneurship-focused learning material to apply those concepts to the real-life startup company, Bowtaps, LLC. Bowtaps uses their recently acquired business skills to market Crowd Control, project associated finances, and seek investors for expansion.

#### 1.2.1 Purpose of the System

Crowd Control is a mobile application designed to ease the experience of going out though the implementation of integrated group messaging, GPS tracking and group management features. Along with the features to manage your group at the event Crowd Control also gives suggestions of local events, restaurants and attraction. This allows the group to continue even when the next item on the agenda is a mystery.

Even though Crowd Control is initially designed for the event-goer scene, it's uses can be expanded to fit more purposes. Crowd Control can be used to help manage any kind of group at an event such as church groups, tour groups, or school field trips.

### 1.3 Business Need

Currently, there is no product on the market that encompasses all the features of group management, in-app messaging, and GPS tracking in one easy-to-use package. For example, Facebook is a popular option for group

messaging and event creation, but it doesn't allow users to track each other for a duration of time. Crowd Control allows for both group messaging and event creation, as well as GPS tracking and better group management, specifically tailored to those at an event. Facebook's limited group management features are specifically designed for planning an event ahead-of-time, not necessarily managing during an event.

Crowd Control not only addresses the needs of those seeking group management/communication during events, but also helps serve as a platform for businesses to advertise themselves to users in the form of events and promotions. Those events are available in the form of nearby events that users can choose to set as their destination upon creating a group. This is a mutually beneficial feature for the users and for the businesses themselves.

## 1.4 Deliverables

The deliverables for this two-semester senior design project are the following items:

- Crowd Control mobile application
- Associated JavaDoc documentation
- Senior Design document - some included items include:
  - General documentation
  - User manual and various code documentation
  - Business plan for Bowtaps, LLC
  - Sprint reports
- Design fair presentation

## 1.5 System Description

Behind the UI, Crowd Control is written using an MVC architecture. IOS is written natively in Swift under the IDE XCode. On the Android side, the code is written in Native Mobile Java under Android Studio.

The code for Android uses XML files for storing layout-related code, which act as the view in the MVC pattern. Each activity acts as a controller. Each one of the models has an interface, which is how the controllers get access to the functions and data provided by the model. Each interface is set up in such a way that it uses OOP inheritance to generalize the models, thus abstracting our third party software. This contract allows us to write our code in such a way that, if ever needed, we could change the underlying implementation of certain modules with minimal refactoring.

The iOS platform follows a very similar MVC design pattern, too.  
(TODO: iOS details needed.)

### 1.5.1 Integrated Group Messaging

Integrated group messaging is an important feature of Crowd Control. It allows for communication across different platforms, different phone brands, and different carriers. This allows for seamless communication between users without the issues associated with SMS such as messages not using the same format, messages not going to all recipients, and messages with users in the group that you do not want to have your personal information.

Bowtaps integrates a third-party called Sinch as our message-handling service. Sinch handles the encryption of messages to ensure security of user data. Sinch uses app to app messaging, so that any device running Crowd Control can send a message to other group members, independent of platform or carrier. However, since our app is currently only fully implemented on Android, messages can only be passed between Android users.

### 1.5.2 GPS Location Services

Crowd Control utilizes GPS functionality to track users that belong to groups through the Google Play services location application program interfaces (APIs). This allows users to find fellow group members by retrieving their current or last known location. This is useful to help locate members of the group that maybe lost or unable to be located, perhaps towards the end of an event, or when group members want to meet up again.

Because GPS functionality can draw heavily from a device's power supply, users are able to opt out of a GPS retrieval on their device. If a user does not want to have their location known to the group, or simply has a low battery level, their GPS retrieval can be shut off. Alternatively, when the user's battery is low, it will allow for the GPS check-in interval to be extended or turned off completely to save battery life.

### 1.5.3 Group Management Features

Perhaps the most important feature set, are those pertaining to group management. The party leader sets the initial information for a group for the other members to view and interact with. The party leader can edit certain information about the group, and even kick certain members if needed. A group management menu allows for a group agenda to be posted, updating members when the agenda changes. Pairing with the GPS features, Crowd Control also allows for the group leader to set way-points for the group.

### 1.5.4 Suggestions

Sponsorship by local businesses take the form of an unobtrusive advertisement called a "suggestion". Suggestions are both a plus for the user and serve as our way of making revenue. Although these are not traditional ads, they alert the user to local points of interest such as restaurants, bars, amusement parks, concerts, and many more. With our suggestion interface, users can browse events for their group to attend while local businesses gain exposure, and we gain revenue.

## 1.6 System Overview and Diagram

The basic overview of Crowd Control can be seen in the diagram below. See Figure 1.1. Crowd Control will be using a model-view-controller design structure. With the model view controller design method we are able to abstract the user interface from the control structures that will communicate with the third party services such as Parse, Google play services, or Sinch. The model of each respective operating system ( Android or iOS ) will be able to communicate with the respective mapping feature ( Google Play Services or Apple Map Features ). While both models will be able to communicate with Parse, our back end server. Though Parse, using their features, will be able to connect user profiles to their Facebook and twitter accounts for faster log in.

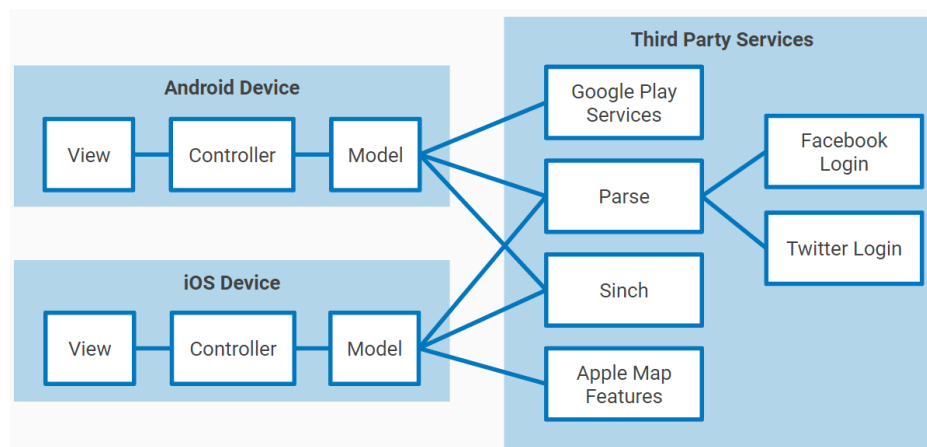


Figure 1.1: Basic System Flow Diagram

## 1.7 Technologies Overview

Crowd Control accesses various different external technology entities. Some technologies used in the creation of Crowd Control are Google Play Services, Apple Map Features, Parse, Sinch, and Android Studio.

### 1.7.1 Google Play Services

#### 1.7.1.a Description

Google Play Services contains a number of APIs that allows Crowd Control to access Google-powered features. One such feature is Google Maps, which allows the app to access mapping capabilities managed by Google. Using their API, users can place pins and find their friends on a map that is smoothly integrated into Crowd Control, without Bowtaps having to maintain the map functionality.

REFERENCE LINK: <https://developers.google.com/android/guides/setup>

#### 1.7.1.b Usage

Google Play Services will be used on the Android device as the default map. Google Play Services provides a more native feel to Android users when interacting with mapping features. This allows for a more consistent user experience when it comes to using Crowd Control. At a set interval which can be changed, the map is updated with the user's current location, as well as all other users in the group. From the map, data is securely stored and transmitted safely to other group members.

### 1.7.2 Apple Map Features

#### 1.7.2.a Description

Apple Map Features is the native iOS API for mapping features. With this it allows for commiseration between a map and your gps location along with other mapping features.

REFERENCE LINK: <https://developer.apple.com/maps/>

#### 1.7.2.b Usage

Apple Map Features will be used on the iOS device as default. We chose to go with Apple Map Features to give iOS users a more native, less intrusive feel. This will be used for displaying your location, displaying other users from your group, and displaying local event suggestions on the map.

### 1.7.3 Parse

#### 1.7.3.a Description

Parse is a third-party service that serves as a secure no-SQL datastore. The service is free under a certain number amount of data transacted to and from Parse. The Parse API allows Crowd Control to access various methods to store and fetch data pertaining to the datastore. It should be noted that Parse will cease server hosting in January of 2017. Parse has provided tools to migrate existing applications to use another database solution such as MongoDB.

REFERENCE LINK: <http://parse.com/>

#### 1.7.3.b Usage

Crowd Control currently saves all group and user information on Parse servers. This data is to be considered sensitive and is treated with the utmost consideration for security and protecting that data. Some data is also cached to the local device, to both reduce data transmission to and from the server, as well as increasing the speed of the user's overall performance. Some of the cached data includes any previous existing user data and group data, so that if the user closes the app, they should not need to re-enter their login information.

## **1.7.4 Sinch**

### **1.7.4.a Description**

Sinch is a third party device-to-device communication API. Bowtaps selected it for its built-in encryption, and ready-to-use messaging platform. The Sinch API provides message-passing functionality various message broadcasting methods. This platform requires either a Wi-Fi connection, or cellular data service.

REFERENCE LINK: <https://www.sinch.com/>

### **1.7.4.b Usage**

Sinch enables Crowd Control to handle the sending and receiving of messages between group members. Through use of the Sinch help manuals, Bowtaps has constructed a fragment to control a user interface that fetches messages sent by users. We have also modified the basic one-to-one message sending implementation to broadcast the message to the entire group.



## 2

---

# User Stories, Requirements, and Product Backlog

---

## 2.1 Overview

This document contains the features, creation and development of crowd control. It covers prerequisite user stories, to the design and implementation of the application itself.

## 2.2 User Stories

### 2.2.1 User Story #1

As a user i want to be able to join a group.

#### 2.2.1.a User Story #1 Breakdown

As a user i want the ability to join a group. Group joining options would be from a list or from an invite from a user.

### 2.2.2 User Story #2

As a user i want the ability to track locations of other members in the group.

#### 2.2.2.a User Story #2 Breakdown

### 2.2.3 User Story #3

As a user i want post agenda for the group.

### 2.2.4 User Story #4

As a user i want to i want the ability to look for local groups

### 2.2.5 User Story #5

As a user i want the ability to have suggestions of local activities.

### 2.2.6 User Story #6

As a user i want the ability to leave a group.

### 2.2.7 User Story #7

As a user i want the ability to have a list of local groups.

### 2.2.8 User Story #8

As a user i want the abilitiy to login.

### 2.2.9 User Story #9

As a user i would like to message other members of the group.

### 2.2.10 User Story #10

As a user i would like my information protected.

## 2.3 Requirements and Design Constraints

This section will cover the main design requirement in all aspects of crowd control.

### 2.3.1 System Requirements

Sense there we are creating Crowd Control to run on two different platforms, both iOS and Android, there are two sets of requirements that will be similar between both platforms. Even though they are both similar, implimentation between both will be differnet. With them both being different they are split into two sections as listed below.

#### 2.3.1.a iOS Requirements

- Use Apple Mapping Features
- Access Parse as the Database

#### 2.3.1.b Android Requirements

- Use Google Maps
- Access Parse as the Database

#### 2.3.1.c Parse Requirements

- Delete groups when group is not in use

### 2.3.2 Network Requirements

Network requirements are mobile networks as this is a mobile applications. The requirement on our part is making sure that the application is able to reach the server and use at little data as possible when connected to the network. Making sure we use as little data as possible will help our users not use all of their data.

### 2.3.3 Development Environment Requirements

The development enviroment requirement is that Crowd Control be avalabe on both iOS and Android platforms. Being cross platform allows for us to reach as many users as possible. Android development will be handled with Android Studio and iOS will be developed with xCode.



### 2.3.4 Project Management Methodology

We have set restrictions on the developemnt of Crowd Control and are listed as follows:

- GitHub issues will be used to keep track of current status as well as backlogs for the product.
- There will be 6 total sprints over 2 scimesters for this products.
- The sprint cycles are 3 weeks long.
- Progress reports will be summited to Dr. McGough and Brian Butterfeild at the end of each sprint.
- Github will be used for source control.

## 2.4 Specifications

### 2.5 Product Backlog

- What system will be used to keep track of the backlogs and sprint status?
- Will all parties have access to the Sprint and Product Backlogs?
- How many Sprints will encompass this particular project?
- How long are the Sprint Cycles?
- Are there restrictions on source control?

### 2.6 Research or Proof of Concept Results

The Proof of conecpt is a rough design that impliments basic features of Crowd Control. Basic features are currently under construction. This is currently a functional prototype with improvements in the future.

Below are screen shots of both android and iOS proof of concepts. (current formatting issues need to fix)

#### 2.6.1 iOS Proof of Concept Screen Shots

Below are screen shots from the iOS version of CrowdControl.

#### 2.6.2 Android Proof of Concept Screen Shots

Below are screen shots from the Android version of CrowdControl.

## 2.7 Supporting Material

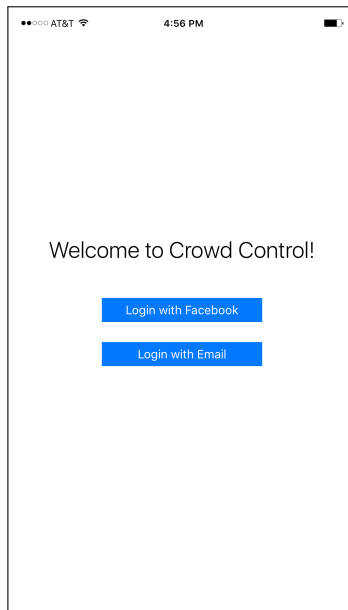


Figure 2.1: iOS login select screen

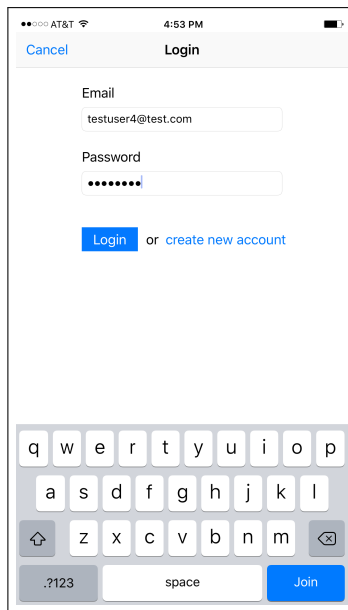
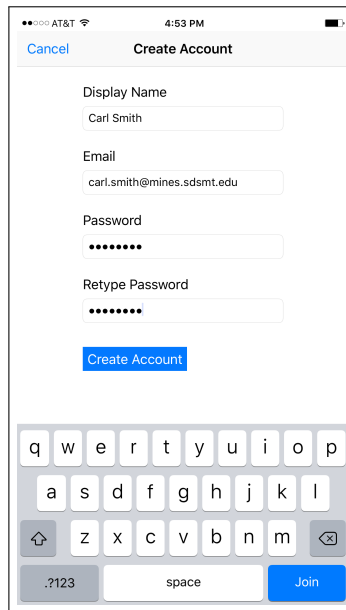
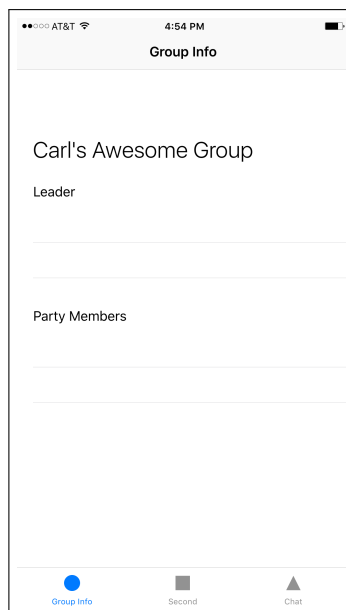


Figure 2.2: iOS email login screen



The image shows an iOS 'Create Account' screen. At the top, the status bar displays 'AT&T' and '4:53 PM'. The screen has a 'Cancel' link and a 'Create Account' title. Below the title are four text input fields: 'Display Name' (containing 'Carl Smith'), 'Email' (containing 'carl.smith@mines.sdsmt.edu'), 'Password' (containing seven dots), and 'Retype Password' (containing seven dots). A blue 'Create Account' button is positioned below the password fields. At the bottom, a standard iOS keyboard is visible, featuring a blue 'Join' button in the bottom right corner.

Figure 2.3: iOS create account screen



The image shows an iOS 'Group Info' screen. The status bar at the top shows 'AT&T' and '4:54 PM'. The screen title is 'Group Info'. Below the title, the group name 'Carl's Awesome Group' is displayed. Underneath, there are two sections: 'Leader' and 'Party Members', each followed by a horizontal line representing a list of members. At the bottom, there is a tab bar with three icons: a blue circle labeled 'Group Info', a grey square labeled 'Second', and a grey triangle labeled 'Chat'.

Figure 2.4: iOS group information screen

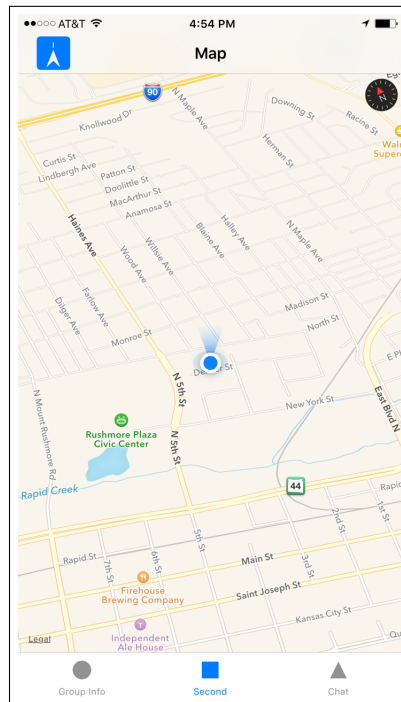


Figure 2.5: iOS map view screen

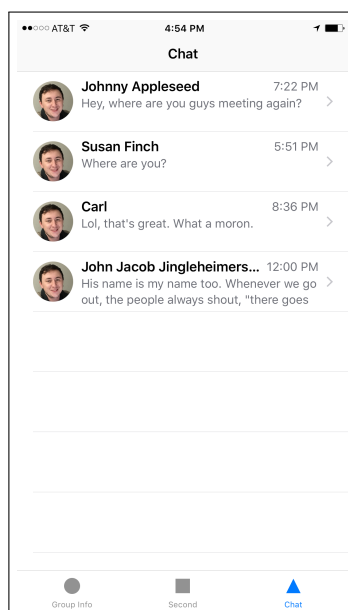


Figure 2.6: iOS messaging main screen

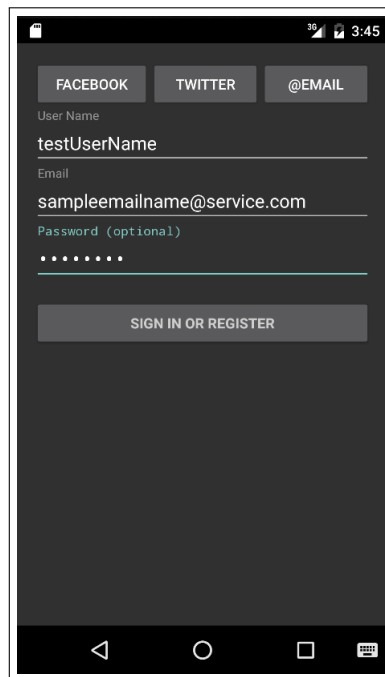


Figure 2.7: Android login screen

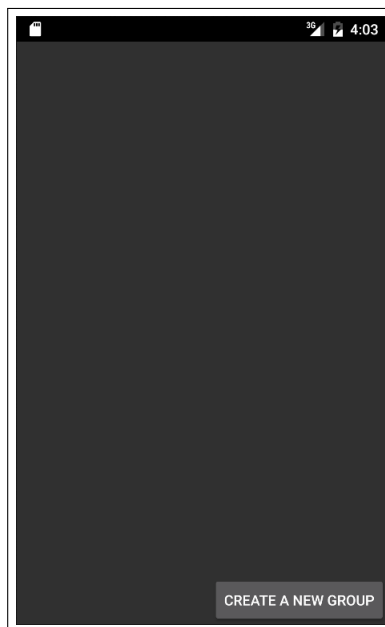


Figure 2.8: Android create group screen

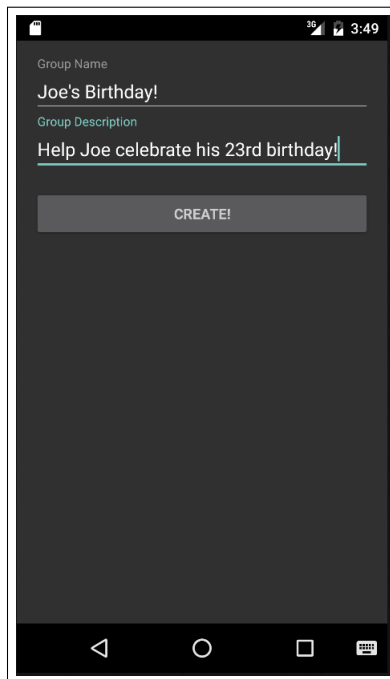


Figure 2.9: Android group information screen

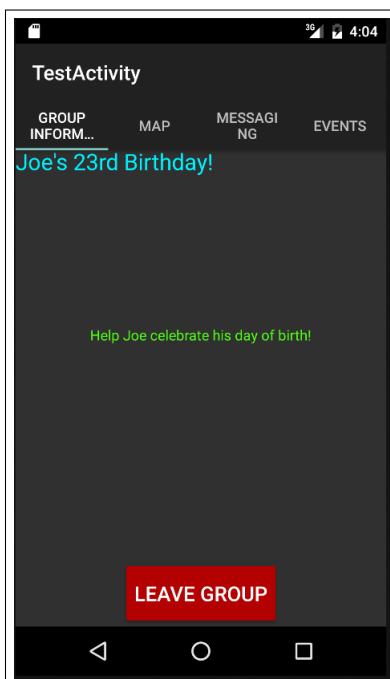


Figure 2.10: Android group join screen

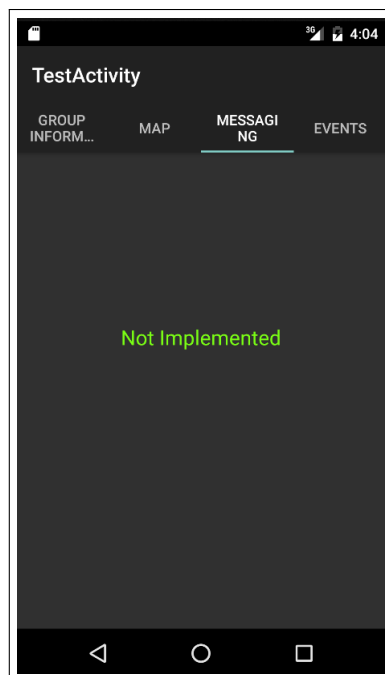


Figure 2.11: Android messaging main screen





## 3

---

### Project Overview

---

This section provides some housekeeping type of information with regard to the team, project, environment, etc.

#### 3.1 Team Member's Roles

Johnnathon Ackerman - Johnnathon is leading the GUI design and implimentation side for the android version of Crowd Control. This entails:

1. Graphical Design
2. Smooth Moving Interfaces
3. Easy to Use and learn layout

Daniel Andrus - Daniel is leading the Gui design ad implimentation for the iOS version of Crowd Control. This entails:

1. Graphical Design
2. Smooth Moving Interfaces
3. Easy to Use and learn layout

Charles Bonn - Charles is leading the database side of Crowd Control. This database is for both iOS and andriod versions. This entails:

1. Creating and managing database qurries
2. Creating Cloud Code to manage database information
3. Database load testing

Charles is also working on future encryption of data going to and from the database.

Evan Hammer - Evan is leading the backend side for the iOS version of Crowd Control. This entails:

1. Creating links from the database to the mobile application
  - (a) Login link
  - (b) Group Join Link
  - (c) Group Member
2. Creating links to Apple maps to the mobile application

Joseph Mowry - Joseph is leading the backend side for the android version of Crowd Control. This entails:

1. Creating links from the database to the mobile application

- (a) Login link
- (b) Group Join Link
- (c) Group Member

2. Creating links to Apple maps to the mobile application

## 3.2 Project Management Approach

This section will provide an explanation of the basic approach to managing the project. Typically, this would detail how the project will be managed through a given Agile methodology. The sprint length (i.e. 2 weeks) and product backlog ownership and location (ex. Trello) are examples of what will be discussed. An overview of the system used to track sprint tasks, bug or trouble tickets, and user stories would be warranted.

## 3.3 Stakeholder Information

This section would provide the basic description of all of the stakeholders for the project. Who has an interest in the successful and/or unsuccessful completion of this project?

### 3.3.1 Customer or End User (Product Owner)

Who? What role will they play in the project? Will this person or group manage and prioritize the product backlog? Who will they interact with on the team to drive product backlog priorities if not done directly?

### 3.3.2 Management or Instructor (Scrum Master)

Who? What role will they play in the project? Will the Scrum Master drive the Sprint Meetings?

### 3.3.3 Investors

Are there any? Who? What role will they play?

### 3.3.4 Developers –Testers

Who? Is there a defined project manager, developer, tester, designer, architect, etc.?

## 3.4 Budget

Describe the budget for the project including gifted equipment and salaries for people on the project.

## 3.5 Intellectual Property and Licensing

Describe the IP ownership and issues surrounding IP.

## 3.6 Sprint Overview

If the system will be implemented in phases, describe those phases/sub-phases (design, implementation, testing, delivery) and the various milestones in this section. This section should also contain a correlation between the phases of development and the associated versioning of the system, i.e. major version, minor version, revision.

All of the Agile decisions are listed here. For example, how do you order your backlog? Did you use planning poker?

## 3.7 Terminology and Acronyms

Provide a list of terms used in the document that warrant definition. Consider industry or domain specific terms and acronyms as well as system specific.

## 3.8 Sprint Schedule

The sprint schedule. Can be tables or graphs. This can be a list of dates with the visual representation given below.

## 3.9 Timeline

Gantt chart or other type of visual representation of the project timeline.

## 3.10 Backlogs

Place the sprint backlogs here. The product backlog will be in the chapter with the user stories.

## 3.11 Burndown Charts

Place your burndown charts, team velocity information, etc here.

## 3.12 Development Environment

The basic purpose for this section is to give a developer all of the necessary information to setup their development environment to run, test, and/or develop.

## 3.13 Development IDE and Tools

Describe which IDE and provide links to installs and/or reference material.

## 3.14 Source Control

Which source control system is/was used? How was it setup? How does a developer connect to it?

## 3.15 Dependencies

Describe all dependencies associated with developing the system.

## 3.16 Build Environment

How are the packages built? Are there build scripts?

## 3.17 Development Machine Setup

If warranted, provide a list of steps and details associated with setting up a machine for use by a developer.



---

## Design and Implementation

---

This section is used to describe the design details for each of the major components in the system. The major components include: Parse for our database, Sinch for messaging, and native APKs (Google and Apple) for maps.

(TODO) Citations look like [?, ?, ?] and [?, ?, ?]. These are done automatically. Just fill in the database `designrefs.bib` using the same field structure as the other entries. Then `pdflatex` the document, `bibtex` the document and `pdflatex` twice again. The first `pdflatex` creates requests for bibliography entries. The `bibtex` extracts and formats the requested entries. The next `pdflatex` puts them in order and assigns labels. The final `pdflatex` replaces references in the text with the assigned labels. The bibliography is automatically constructed.

### 4.1 Architecture and System Design

This section will detail the overall system design and general architecture of Crowd Control. The software was designed in such a way that minimizes dependency from third-party services such as Sinch and Parse. It does this by abstracting everything using an extra layer of models. The extra models stop the third party calls from being visible in the main activities in Android as well as the view logic in iOS.

#### 4.1.1 Design Selection

Sprint 1 was centered around designing of the database schema and the general system architecture. Bowtaps produced various high-level designs on both the front-end and back-end of the system that were deeply inspected before deciding on our current implementation. Later, the architecture evolved as the year progressed, and there were changes to the overall UX as well as how the apps themselves retrieved the data from parse. In addition, the database schema was changed over winter break to include the knowledge we learned over the first sprint. These changes still persist though our current product.

##### 4.1.1.a Early Design Ideas

The original database design for Crowd Control consisted of three tables with associated data. Though this design provided a good sense of direction and foundation to build upon, it eventually would need to expand as Crowd Control's feature set expanded. See Figure 4.1 below.

Another caveat to this design is the failure to differentiate between public and private user data. In Crowd Control, each user has data that is private to that user, such as their email and password. However, there is also a set of public data that can be viewed by other users, such as their display name or their location. This iteration of the schema only has one user entity, that stores their information. A small lack of understanding about Parse's no-SQL implementation lead us to improperly design our entities in this way. It was later deemed that this separation between front-facing and hidden user data entities was necessary in terms of ease-of-access and information privacy.

##### 4.1.1.b Improvements to Early Designs

In reflection of the shortcomings of the first design iteration, the database schema was restructured to better fit Crowd Control's needs. The current design consists of eight data tables, shown in Figure 4.2 below. More entities

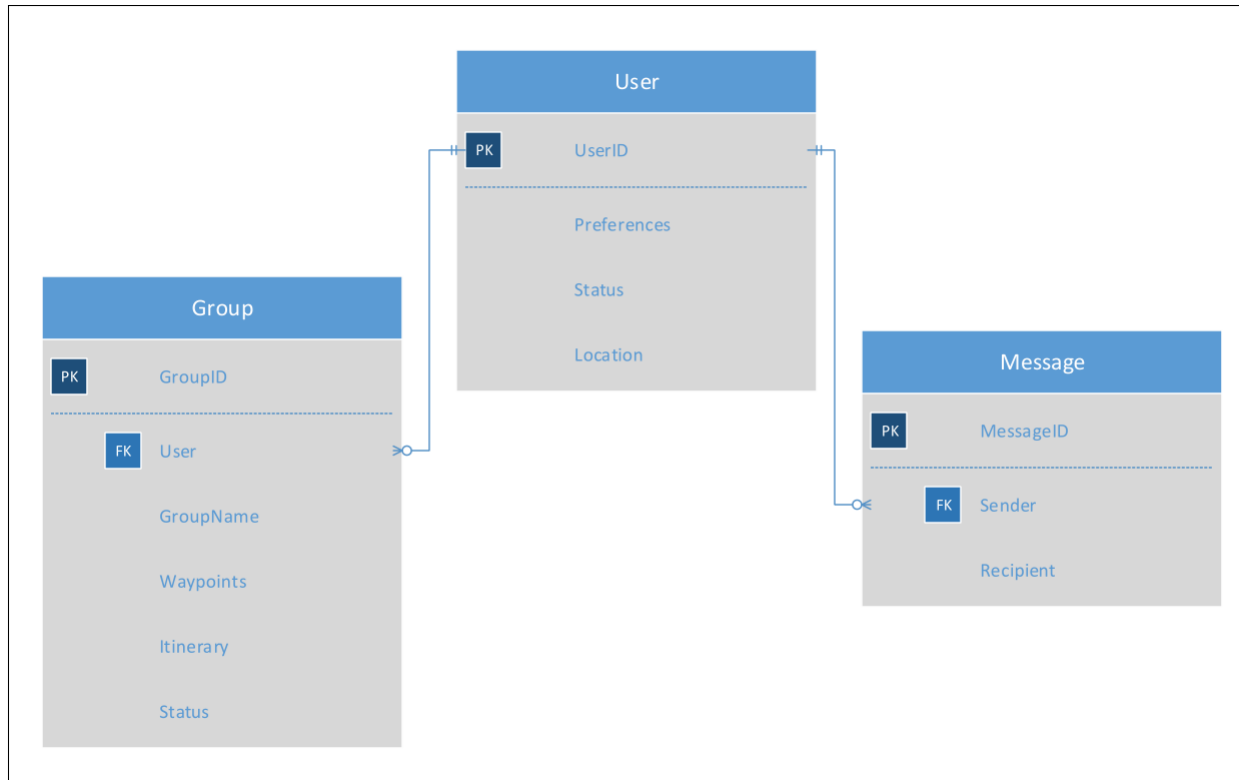


Figure 4.1: Early database schema.

were added to the overall schema. One such addition was a “CCUser” table to hold the public data associated with a user. It is important to note that the “User” table was left to hold a user’s private data. Those changes and some additions resulted in the below schema.

### 4.1.2 Data Structures and Algorithms

By developing Crowd Control in the Android Studio environment, all the data structures available to Java are available to the project itself. That being said, there are no particularly noteworthy structures or algorithms that need to be expound upon, but there are class hierarchies that will be explained in later detail in this chapter.

### 4.1.3 Data Flow

TODO: Create Data Flow diagram of overall process

### 4.1.4 Communications

One of the core features to the Crowd Control app is communication amongst its users. To achieve this, some third-party services are used, which in turn communicate data between users. If a user wishes to send another user a message, that message is sent to the Parse back-end, and delivered to the recipient via the Sinch service. The basic communication overview is outlined below in Figure 4.3.

In addition to message-passing, various other data is being transferred from the users’ devices. One such example is GPS data. Upon retrieval of a user’s location, that value is stored in Parse, and able to be fetched by any group member that wishes to see that location.

### 4.1.5 Classes



Figure 4.2: Improved database schema.

#### 4.1.5.a Sign up and Log in Classes

Everything starts with the `WelcomeActivity.java` class. This initial activity checks if the user has logged in or not by looking against its local Parse database. If a user has logged in or signed up it will be stored in the local Parse database.

The `SignupActivity.java` is next on the list. It allows a user to sign into parse, it was created using the

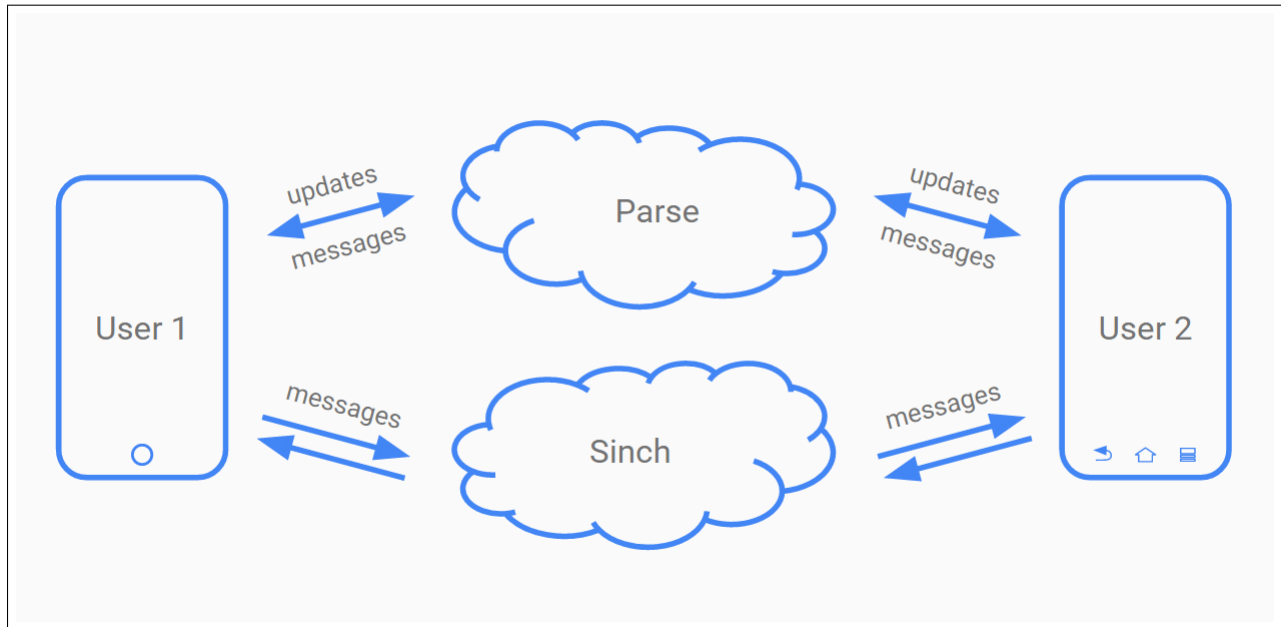


Figure 4.3: Communication flow diagram.

Android log-in/sign-up template. From here a user can access a Facebook log-in page and a Twitter log-in page. Additionally the class, LoginActivity.java allows the user to log into a registered email account.

For either of the previous paths, the next class the user will enter is GroupJoinActivity.java.

#### 4.1.5.b Menu Classes

The menu can be accessed from the top right of the app after a user has either logged in or created an account. The code for the creation of the menu can be found in GroupJoinActivity.java (for a basic menu) and in GroupInfoFragment.java (which can be basic or a leader menu).

The classes connected to the menu include: SettingsActivity.java, NotificationActivity.java and InviteNavigationActivity.java. The SettingsActivity.java class allows a user to log out and return to the SignUpActivity.java class, as well as see what group they are in, and gives the user the ability to change their user name. The NotificationActivity.java allows the user to access all their in app notifications. The InviteNavigationActivity.java will be discussed later because it is only accessible if the user is a leader of a group.

#### 4.1.5.c Initial Tab Class

After a user has joined a group in the class GroupJoinActivity.java, they will reach GroupNavigationActivity.java. This activity is in charge of displaying the four Fragment class: EventFragment.java, GroupInfoFragment.java, MessagingFragment.java, and MapFragment.java. It is the main UX for the user for the duration of a group.

The EventFragment.java is not implemented and is no longer accessible in the app.

#### 4.1.5.d Group Management Classes

The GroupInfoFragment.java class is the main class for group management tools. This class allows a leader of a group to kick a member or promote a new leader. This class also holds the code for the group menu. The group menu also allows the user to change the group name.

Other group management classes can be accessed though the menu for group leaders. These classes include: InviteNavigationActivity.java, InviteConfirmFragment.java, and InviteSearchFragment.java. The InviteNavigationActivity.java class is much like the GroupNavigationActivity.java class. It is a tab system holding the other two classes. The InviteSearchFragment.java allows the leader to search for other users they wish to add to the group.



It creates a group list that is given to the InviteConfirmFragment.java class. Here the leader can make sure that these are the people that he/she want to invite and completes the process.

#### 4.1.5.e Location Classes

Location functionality took place in three major locations: the GoogleLocationListener.java file in the “location” folder, the MapFragment.java file in the “com.bowtaps.crowdcontrol” folder, and the GroupService.java file in that same folder.

The map fragment serves as the controller for the fragment\_map.xml layout file. Since this is a fragment, in contrast to an activity, it must be nested in an activity class. The MapFragment.java class is nested inside the actual tab adapter.

The two other files work together to call methods to update and store location data when desired. This functionality is done via cloud code in order to carry these tasks out asynchronously. In short, the listeners on the service communicate with the cloud code, which carries out the desired functions, and reports it back to the listener. The data is then updated on the model, for the user and for the group members.

#### 4.1.5.f Messaging Classes

Messaging consisted of a three major classes. These class include MessageService.java, MessageAdapter.java, and MessagingFragment.java. The MessagingFragment.java class is part of the tab system found in the InviteNavigationActivity.java. This way it can be a key component to the flow of groups. The MessageAdapter.java is a helping class. Its sole purpose is to modify message objects into views so that they can be displayed in the MessagingFragment. The MessageService.java handles our apps connection to Sinch. It's a stand alone service that sends and receives messages with other Sinch servers. Without this service the app wouldn't be able to listen for new messages. Finally, our overarching server, that handles all the updates for data in the database, also handles storing conversations. Each group has its own conversation in the database, that way the conversation can be loaded into the MessageService.java class so that they can be loaded for new members or if something happens to an existing members local copy.

I explicitly want to note that the messaging service and many of the GUI elements were taken from a GitHub repository owned by Sinch [?]. This repository gives full permission for commercial use [?, ?].

#### 4.1.5.g Miscellaneous Classes

There were a few classes not explained from the ones listed above including: CrowdControlApplication.java, GroupJoinActivity.java, and GroupService.java.

The CrowdControlApplication.java is the very first class that initializes the app on launch, it loads up any locally stored data as well as holds some global classes so that other activities can access them.

The GroupJoinActivity.java class loads in the groups for the user to select a group. It also holds a basic menu for a user to get to their own user related info. This class also uses the GroupModelAdapter.java class. This is used to change group models loaded in from Parse into objects that can be displayed in a list.

The SimpleTabsAdapter.java is used to create views to hold multiple fragments in one activity. This used used in GroupNavigationActivity.java and in InviteNavigationActivity.java.

### 4.1.6 UML

Below in 4.4 is a high-level use case UML diagram that shows basic user functionality of the Crowd Control application. The light-blue nodes are reserved for group leaders only, but the green colored nodes are available to all users. Once a group leader promotes another user to become the new group leader, the previous leader simply becomes a group member and loses their leader privileges. Of course, after creating a group, one must have group members in order to kick or promote members, as well as message or locate other members in the first place.

### 4.1.7 GUI

Designing the user interface of a mobile application has many unique challenges, all of which influence design decisions. In order to present users with a satisfying and intuitive experience, it is suggested to follow material

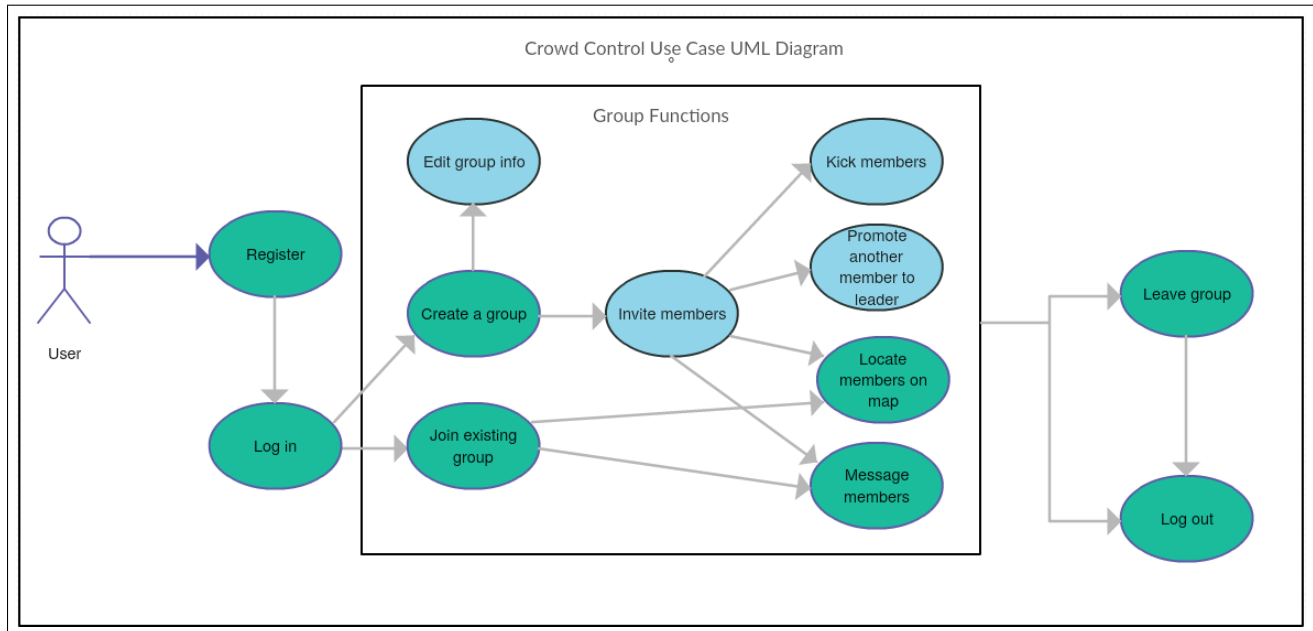


Figure 4.4: Use Case Diagram in UML.

design pattern aspects such as using Floating Action Buttons (FABs), and simply adhering to a more unified experience that can be found across multiple new applications on many platforms. With that in mind, we also focused on keeping the overall design very clean and free from clutter.

Another influence on our design choices, was the integration of Google Services in Crowd Control. Since Google's components currently follow a material design approach, we chose to do so as well to ensure a consistent experience when a user interacts with one of their interfaces.

Lastly, Crowd Control was designed to look similar to users, regardless of their platform; an Android user should have a very similar interface, compared to that of an iOS user, and vice versa. By choosing elements from each platform's respective layout styles, we are also able to create a design that conforms to the guidelines set by the device or platform itself. Ultimately, our design choices ensure that users have a consistent user experience, independent of their platform or their device.

## 4.1.8 MVC

### 4.1.8.a Introduction

MVC, or "model-view-controller" is a design pattern used widely in mobile applications to separate an application's core functionalities.

- Model - Model represents an object to carry or store data. It can also have logic to update a controller if its data changes.
- View - Views represent the visualization of the data that model contains, such as a screen or an activity in Android. These are represented as a layout generally.
- Controller - Controllers act on both the model and the views. They control the data flow into model objects and updates the view whenever data changes. It maintains a separation between the view and the model.

### 4.1.8.b Models

In Android, the models are a collection of model classes, such as the GroupModel and the UserModel, as well as other various storage classes. They can be found in a "models" folder at "`~/CrowdControl/app/src/main/java/-com/bowtaps/crowdcontrol/model/`", shown below in 4.5:

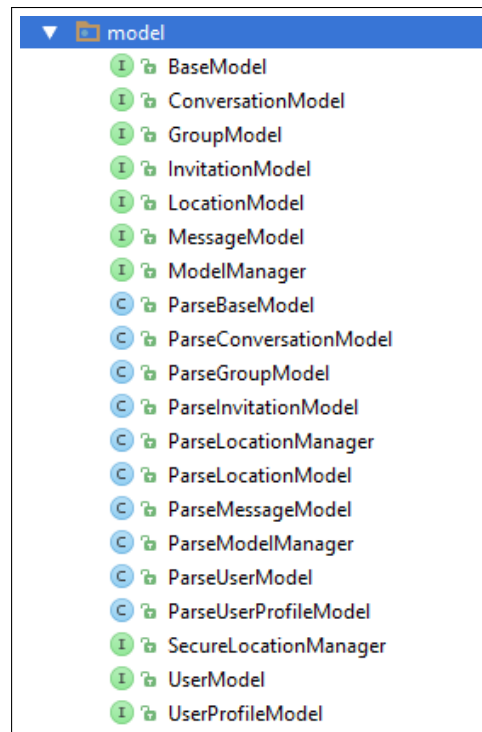


Figure 4.5: Model classes in the “model” folder of the project.

Our models were designed to be abstracted away from our third party software in order to keep API-specific code of the activities. In this way, one could implement a different back-end service instead of Parse with minimal refactoring. We have several models that keep track of local data and keep up to date with our Parse database using a service. These models can be accessed by controllers. This is initialized `CrowdControlApplication.java`, which is the global main class of the app, serving as the application's entry point.

### 4.1.8.c Controllers

Each activity file is a controller, and is responsible for calling functions from the models to either change data in the database, or to update the data in the layouts; it facilitates information between the models and the views. Each activity has one layout in general. These layouts take place in the form of XML files, and thus their tags are specific to the Android Studio/Java environment. Controller files can be found at “~/CrowdControl/app/src/main/java/com/bowtaps/crowdcontrol/”, shown below in 4.6:

### 4.1.8.d Views

As stated, views typically take the form of XML files in the Crowd Control project. These files contain no business logic, and simply contain layout information that pertains to the appearance and attributes of the page itself. These layouts are constructed via the Android Studio layout designer, then edited to tailor to our specific needs. Views can be found at “~/CrowdControl/app/src/main/res/layout/”, shown below in 4.7:

## 4.2 Group Messaging

### 4.2.1 Technologies Used

- Parse - back-end database tool.
- Sinch - back-end messaging tool.

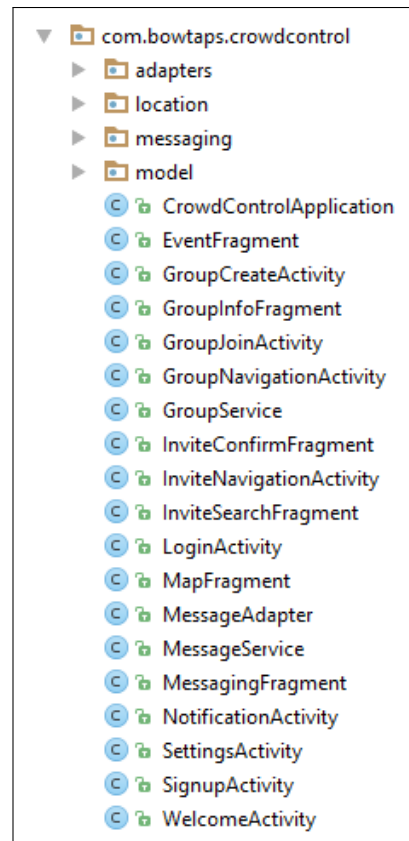


Figure 4.6: Model classes in the “model” folder of the project.

### 4.2.2 Component Overview

- Send/receive messages to/from group members (many-to-many)
- Store messages in Parse (data persistence)
- Message transfer service is independent of carrier/device/platform

### 4.2.3 Phase Overview

**Phase 1** Construct message containers and begin Sinch integration

**Phase 2** Construct messaging activity and view to encapsulate functionality

**Phase 3** Construct conversation containers to associate groups of messages

**Phase 4** Implement many-to-many message passing/broadcasting through Sinch delivery

**Phase 5** Store associated messages inside conversations, both on Parse, as well as caching locally on the user device

### 4.2.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

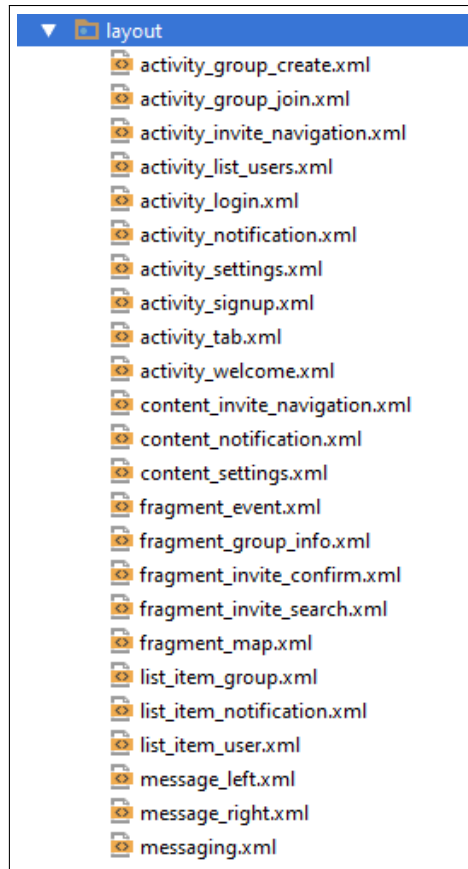


Figure 4.7: Model classes in the “model” folder of the project.

## 4.2.5 Data Flow Diagram

## 4.2.6 Design Details

Messaging interfaces with the user through the `MessagingFragment.java` class. This class uses the `MessageAdapter.java` class to take the data from the `MessageService.java` class, and display it to the user. The `MessagingFragment.java` class also allows the user to send messages through the `MessageService.java` class to all other members in the group. The `MessagingFragment.java` class relies on the app's overarching service to keep its messages stored in Parse. This allows the class to load a conversation (group object that holds messages) into view for the user, if that user doesn't have them or loses their local copy.

## 4.3 Location Tracking

### 4.3.1 Technologies Used

- Google Play Services / Apple Map Features
- Parse

### 4.3.2 Component Overview

- Track all group members in a map fragment
- Homing functionality on the user's location pin
- Sync group locations automatically (interval-based) and manually (on button-press)

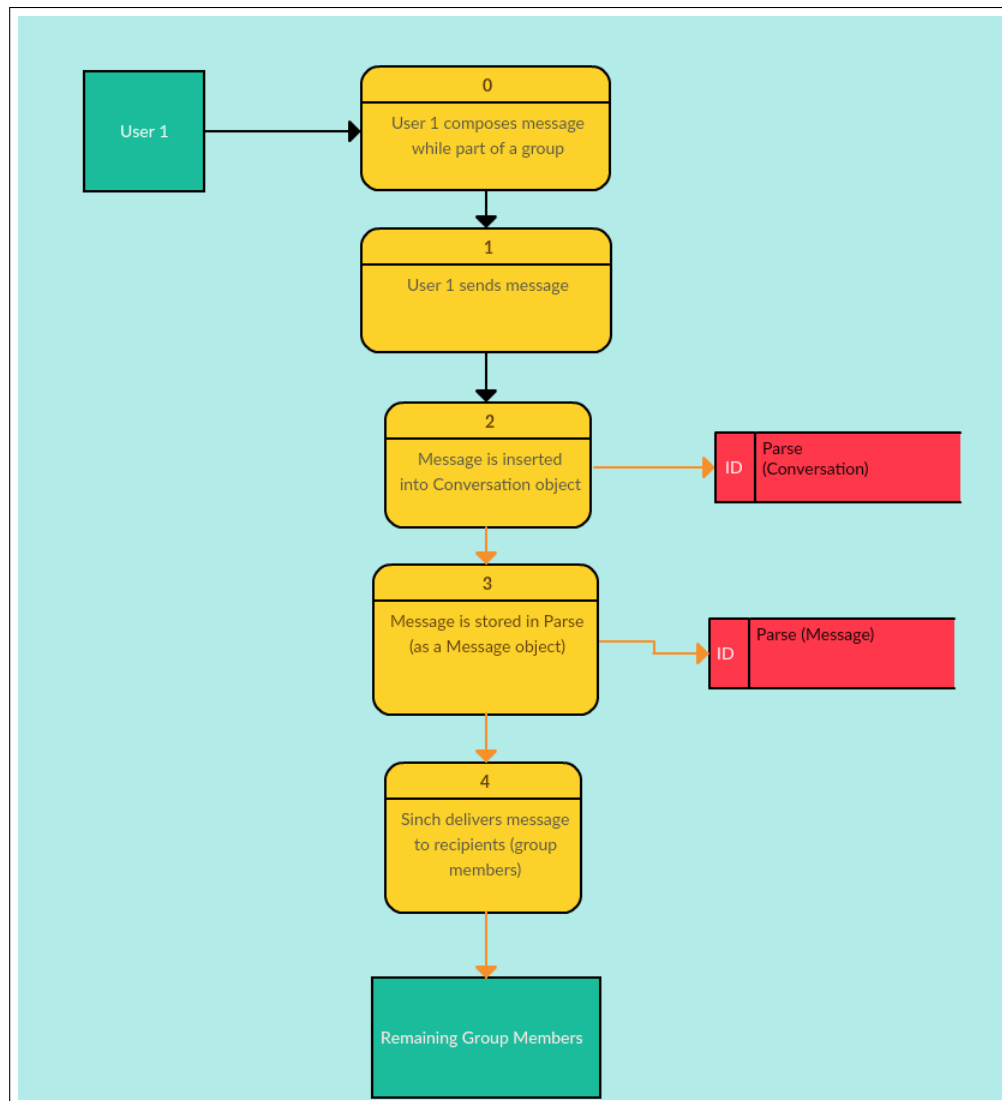


Figure 4.8: Messaging data flow diagram.

### 4.3.3 Phase Overview

**Phase 1** Construct location containers and begin location services integration

**Phase 2** Construct map fragment, buttons for future homing/syncing functionality

**Phase 3** Implement homing and location syncing features, as well as interval-based location syncing

**Phase 4** Extract location update functionality from client, place into cloud-based service for better (faster) asynchronous user experience

### 4.3.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

### 4.3.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

### 4.3.6 Design Details

This is where the details are presented and may contain subsections.

## 4.4 Group Management

### 4.4.1 Technologies Used

- Parse

### 4.4.2 Component Overview

- Store group members in a group
- Incorporate a party leader to manage the group (has special privileges)
- Create a group with specific attributes that can be changed by the leader

### 4.4.3 Phase Overview

**Phase 1** Construct model layer and base model classes to communicate with Parse and store data application-side

**Phase 2** Allow users to create or join groups, manage group members that join and create groups

**Phase 3** Add Group Leader functionality to manage group (kick/promote members, edit description, etc)

**Phase 4** Begin design/integration of messaging components for group management

### 4.4.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

### 4.4.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

### 4.4.6 Design Details

This is where the details are presented and may contain subsections.





# 5

---

## System and Unit Testing

---

This section describes the approach taken with regard to system and unit testing.

### 5.1 Overview

Provides a brief overview of the testing approach, testing frameworks, and general how testing is/will be done to provide a measure of success for the system.

Each requirement (user story component) should be tested. A review of objectives and constraints might be needed here.

### 5.2 Dependencies

Describe the basic dependencies which should include unit testing frameworks and reference material.

### 5.3 Test Setup and Execution

Describe how test cases were developed, setup, and executed. This section can be extremely involved if a complete list of test cases was warranted for the system. One approach is to list each requirement, module, or component and describe the test.

The unit tests are described here.

### 5.4 System Testing

### 5.5 System Integration Analysis

### 5.6 Risk Analysis

#### 5.6.1 Risk Mitigation

### 5.7 Successes, Issues and Problems

#### 5.7.1 Changes to the Backlog



# 6

---

## Prototypes

---

This chapter is for recording each prototype developed. It is a historical record of what you accomplished in 464/465. This should be organized according to Sprints. It should have the basic description of the sprint deliverable and what was accomplished. Screen shots, photos, captures from video, etc should be used.

### 6.1 Sprint 1 Prototype

#### 6.1.1 Deliverable

#### 6.1.2 Backlog

#### 6.1.3 Success/Fail

### 6.2 Sprint 2 Prototype

#### 6.2.1 Deliverable

#### 6.2.2 Backlog

#### 6.2.3 Success/Fail

### 6.3 Sprint 3 Prototype

#### 6.3.1 Deliverable

#### 6.3.2 Backlog

#### 6.3.3 Success/Fail

### 6.4 Sprint 4 Prototype

#### 6.4.1 Deliverable

#### 6.4.2 Backlog

#### 6.4.3 Success/Fail

### 6.5 Sprint 5 Prototype

#### 6.5.1 Deliverable

#### 6.5.2 Backlog

### 6.5.3 Success/Fail

# 7

---

## Release – Setup – Deployment

---

This section should contain any specific subsection regarding specifics in releasing, setup, and/or deployment of the system.

### 7.1 Deployment Information and Dependencies

Are there dependencies that are not embedded into the system install?

### 7.2 Setup Information

How is a setup/install built?

### 7.3 System Versioning Information

How is the system versioned?



## 8

---

### User Documentation

---

This section should contain the basis for any end user documentation for the system. End user documentation would cover the basic steps for setup and use of the system. It is likely that the majority of this section would be present in its own document to be delivered to the end user. However, it is recommended the original is contained and maintained in this document.

#### 8.1 User Guide

The source for the user guide can go here. You have some options for how to handle the user docs. If you have some `newpage` commands around the guide then you can just print out those pages. If a different formatting is required, then have the source in a separate file `userguide.tex` and include that file here. That file can also be included into a driver (like the senior design template) which has the client specified formatting. Again, this is a single source approach.

#### 8.2 Installation Guide

#### 8.3 Programmer Manual





## 9

---

# Class Index

---

### 9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Poly . . . . .	43
----------------	----



# 10

---

## Class Documentation

---

### 10.1 Poly Class Reference

#### Public Member Functions

- Poly ()
- ~Poly ()
- int myfunction (int)

#### 10.1.1 Constructor & Destructor Documentation

##### 10.1.1.a Poly::Poly ( )

My constructor

##### 10.1.1.b Poly::~~Poly ( )

My destructor

#### 10.1.2 Member Function Documentation

##### 10.1.2.a int Poly::myfunction ( int *a* )

my own example function fancy new function

new variable

The documentation for this class was generated from the following file:

- hello.cpp



**11**

---

**Business Plan**

---

# Crowd Control Business Plan



BowTaps, LLC

Charles Bonn:	<a href="mailto:nick.bonn@bowtaps.com">nick.bonn@bowtaps.com</a>
Johnathan Ackerman:	<a href="mailto:johnny.ackerman@bowtaps.com">johnny.ackerman@bowtaps.com</a>
Daniel Andrus:	<a href="mailto:dan.andrus@bowtaps.com">dan.andrus@bowtaps.com</a>
▣Evan Hammer:	<a href="mailto:evan.hammer@bowtaps.com">evan.hammer@bowtaps.com</a>
Joseph Mowry:	<a href="mailto:joe.mowry@bowtaps.com">joe.mowry@bowtaps.com</a>

---

%sectionMetrics and Milestones





# SDSMT SENIOR DESIGN SOFTWARE DEVELOPMENT AGREEMENT

This Software Development Agreement (the "Agreement") is made between the SDSMT Computer Science

Senior Design Team: \_\_\_\_\_ CrowdControl \_\_\_\_\_  
( "Student Group" )

consisting of team members: Charles Bonn, Evan Hammer, Joseph Mowry, Daniel Andrus, Johnathan Ackerman,  
( "Student Names" )

and Sponsor: \_\_\_\_\_ Bowtaps ( self ) \_\_\_\_\_,  
( "Company Name" )

with address: \_\_\_\_\_ 2326 Lance Street, Rapid City , SD 57702 \_\_\_\_\_.

## 1 RECITALS

1. The Bowtaps team will be designing, implimenting, and distributing CrowdControl under the SDSMT Senior Design program.

NOW, THEREFORE, in consideration of the mutual covenants and promises herein contained, Bowtaps and Brian Butterfeild agree as follows:

## 2 EFFECTIVE DATE

This Agreement shall be effective as of \_\_\_\_\_ 9/30/2015. \_\_\_\_\_

## 3 DEFINITIONS

1. "Software" shall mean the computer programs in machine readable object code and any subsequent error corrections or updates created by Bowtaps for CrowdControl pursuant to this Agreement.
2. "Acceptance Criteria" means the written technical and operational performance and functional criteria and documentation standards set out in the backlog.
3. "Acceptance Date" means the date for each Milestone when all Deliverables included in that Milestone have been accepted by BowTaps under the supervision of Brian Butterfeild in accordance with the Acceptance Criteria and this Agreement.
4. "Deliverable" means the product requirements specified in the backlog under the acceptance date.
5. "Delivery Date" shall mean, with respect to a particular sprint, the date on which BowTaps will evaluate all of the Deliverables for that sprint in accordance with the backlog and this Agreement.
6. "Documentation" means the documents, manuals and written materials (including end-user manuals) referenced, indicated or described in the project plan or otherwise developed pursuant to this Agreement.
7. "Milestone" means the completion and delivery of all of the Deliverables or other events which are included or described in backlog scheduled for developement and/or completion on a given target date; a Milestone will not be considered completed until the Acceptance Date has occurred with respect to all of the Deliverables for that Milestone.

## 4 DEVELOPMENT OF SOFTWARE

1. The BowTaps Team will use its best efforts to develop the Software described in backlog The Software development will be under the direction of Its members with the supervision of Brian Butterfeild. BowTaps will deliver the Software to the satisfaction of the course instructor that reasonable effort has been made to design and release CrowdControl as a mobile application. The Team understands that failure to deliver the Software is grounds for failing the course.
2. Brian Butterfeild understands that the Senior Design course's mission is education and advancement of knowledge, and, consequently, the development of Software must further that mission. The Senior Design Course does not guarantee specific results or any results, and the Software will be developed only on a best efforts basis. The Software created will be intened as a beta release for future refinement before the release of CrowdControl.
3. The Senior Design instructor will act as mediator for BowTaps to help guide twords a start up software engineering company

## 5 COMPENSATION

NONE. This is a company start up with the goals of releasing a mobile application and starting a software developement company.

## 6 CONSULTATION AND REPORTS

1. Sponsor's designated representative for consultation and communications with the BowTaps team shall be \_\_\_\_\_ Brian Butterfeild \_\_\_\_\_ or such other person as consultant(s) may from time to time designate to the BowTaps team.
2. During the Term of the Agreement, consultant's representatives may consult informally with course instructor regarding the project, both personally and by telephone. Access to work carried on in University facilities, if any, in the course of this Agreement shall be entirely under the control of University personnel but shall be made available on a reasonable basis.
3. BowTaps will submit written progress reports. At the conclusion of this Agreement, the BowTaps team shall submit a comprehensive final report in the form of the formal course documentation at the conclusion of the Senior Design II course.

## 7 CONFIDENTIAL INFORMATION

1. The parties may wish, from time to time, in connection with work contemplated under this Agreement, to disclose confidential information to each other ("Confidential Information"). Each party will use reasonable efforts to prevent the disclosure of any of the other party's Confidential Information to third parties for a period of three (3) years after the termination of this Agreement, provided that the recipient party's obligation shall not apply to information that:
  - (a) is not disclosed in writing or reduced to writing and so marked with an appropriate confidentiality legend within thirty (30) days of disclosure;
  - (b) is already in the recipient party's possession at the time of disclosure thereof;
  - (c) is or later becomes part of the public domain through no fault of the recipient party;
  - (d) is received from a third party having no obligations of confidentiality to the disclosing party;

- (e) is independently developed by the recipient party; or
  - (f) is required by law or regulation to be disclosed.
2. In the event that information is required to be disclosed pursuant to subsection (6), the party required to make disclosure shall notify the other to allow that party to assert whatever exclusions or exemptions may be available to it under such law or regulation.

## **8 INTELLECTUAL PROPERTY RIGHTS**

Intellectual Property created during the development, testing, deployment, and updating of CrowdControl. Intellectual Property consists of any documents drafted, products designed, and code written and implemented by BowTaps. The Intellectual Property belongs to the development team, BowTaps, under the direction and guidance of SDSM&T and consultants.

## **9 WARRANTIES**

The BowTaps Team represents and warrants to Sponsor that:

- 1. the Software is the original work of the BowTaps Team in each and all aspects;
- 2. the Software and its use do not infringe any copyright or trade secret rights of any third party.

No agreements will be made beyond items (1) and (2).

## **10 INDEMNITY**

- 1. BowTaps is responsible for claims and damages, losses or expenses held against the BowTaps team.
- 2. NEITHER PARTY TO THIS AGREEMENT NOR THEIR AFFILIATED COMPANIES, NOR THE OFFICERS, AGENTS, STUDENTS AND EMPLOYEES OF ANY OF THE FOREGOING, SHALL BE LIABLE TO ANY OTHER PARTY HERETO IN ANY ACTION OR CLAIM FOR CONSEQUENTIAL OR SPECIAL DAMAGES, LOSS OF PROFITS, LOSS OF OPPORTUNITY, LOSS OF PRODUCT OR LOSS OF USE, WHETHER THE ACTION IN WHICH RECOVERY OF DAMAGES IS SOUGHT IS BASED ON CONTRACT TORT (INCLUDING SOLE, CONCURRENT OR OTHER NEGLIGENCE AND STRICT LIABILITY), STATUTE OR OTHERWISE. TO THE EXTENT PERMITTED BY LAW, ANY STATUTORY REMEDIES WHICH ARE INCONSISTENT WITH THE PROVISIONS OF THESE TERMS ARE WAIVED.

## **11 INDEPENDENT CONTRACTOR**

For the purposes of this Agreement and all services to be provided hereunder, the parties shall be, and shall be deemed to be, independent contractors and not agents or employees of the other party. Neither party shall have authority to make any statements, representations or commitments of any kind, or to take any action which shall be binding on the other party, except as may be expressly provided for herein or authorized in writing.

## **12 TERM AND TERMINATION**

1. This Agreement shall commence on the Effective Date and extend until the end of classes of the second semester of Senior Design (CSC 467), unless sooner terminated in accordance with the provisions of this Section (“Term”).
2. This Agreement may be terminated by the written agreement of both parties.
3. In the event that either party shall be in default of its materials obligations under this Agreement and shall fail to remedy such default within thirty (30) days after receipt of written notice thereof, this Agreement shall terminate upon expiration of the thirty (30) day period.
4. Any provisions of this Agreement which by their nature extend beyond termination shall survive such termination.

## **13 GENERAL**

1. This Agreement constitutes the entire and only agreement between the parties relating to the Senior Design Course, and all prior negotiations, representations, agreements and understandings are superseded hereby. No agreements altering or supplementing the terms hereof may be made except by means of a written document signed by the duly authorized representatives of the parties.
2. This Agreement shall be governed by, construed, and enforced in accordance with the internal laws of the State of South Dakota.

## 14 SIGNATURES



10 / 6 / 2015

---

Charles Bonn

---

Date



10 / 6 / 2015

---

Evan Hammer

---

Date



10 / 6 / 2015

---

Joseph Mowry

---

Date



10 / 6 / 2015

---

Daniel Andrus

---

Date



10 / 6 / 2015

---

Johnathan Ackerman

---

Date



10 / 6 / 2015

---

Brian Butterfeild

---

Date



# A

---

## Product Description

---

CrowdControl is a group management application that will be an application that has gps features, group messaging, group management features.

### 1 GPS Features

#### 1.1 Group Members

The Group member gps features will allow for users to track other users in the same group as they are. This will be under user permission to allow other user to see there location.

#### 1.2 Suggestions

The suggestion side of the GPS will take a user or group location and give even suggestions of places to go or things to do in the area of the group.

### 2 Group Messaging

Integrated group messaging on a single platform uniform to iOS and android.

### 3 Group Manangement Features

This will allow for members to join a group, add a member to a group, and leave a group.

### 4 Parse Features

Parse will be used to store user data and group data.





**B**

---

## **Sprint Reports**

---

### **1 Sprint Report #1**

# Sprint Report #1

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

Crowd Control Group Management Mobile Application

### Company

Bowtaps

---

## Customer Overview

---

### Customer Description

Bowtaps is a start up company based out of Rapid City, SD. Bowtaps plans on having their initial market presence with the mobile application Crowd Control.

### Customer Problem

The design, creation, and marketing of the mobile application Crowd Control along with the creation of the company Bowtaps.

### Customer

- GPS mapping of Members in the group
- Integrated group messaging
- Group management features ( add/remove members )
- Intuitive UI
- Product testing
- Marketing plan and strategies
- Business plan
- End-user Documentation

---

## Project Overview

---

The creation of Crowd Control, a mobile application on Android and iOS platforms for group management.

### Phase 1

The design of the database and the basic design of the user interface.

---

## Project Environment

---

### Project Boundaries

- Crowd Control will be a free app available for download on the Android and iOS marketplaces.
- The product will be coded in Java ( Android ) , swift ( iOS ), and parse (back-end server).
- Source code will be kept in a private GitHub repository.
- Crowd Control will be planned on release by summer of 2016.

### Project Context

- There will be 2 versions of the application ( one for iOS and one for Android )
- Crowd Control will access a parse server
- Crowd Control will access GPS information

---

## Deliverables

---

### Phase 1

Deliverables will be UX design, database design and implementation.

---

## Backlog

---

### Phase 1

- Design UX
  1. Create groups
  2. Leave groups
  3. Group messaging
  4. Start page
- Database

1. Design database schema
  2. Implement database on Parse
- Design application layers ( MVC )
  - Set up GitHub repository

---

## **Sprint Report**

---

### **Work for this sprint included:**

- Designs for Create Group page
- Design for Leave Group page
- Design for Group Messaging page
- Design for Start Page
- Design for Database Schema
- Database implementation
- Git Repository Initialization

## 2 Sprint Report #2

# Sprint Report #2

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

Crowd Control Group Management Mobile Application

### Company

Bowtaps

---

## Work Summary

---

- Code UX
  1. Map Screen
  2. Group info Screen
  3. Group Messaging
  4. Start page
  5. Group Info UI
- Model
  1. User Model
  2. Communication layer
- Research on public/private key passing

---

## Backlog

---

- Code UX
  1. Mapping features
  2. Messaging UI
- Model

1. User Model
  2. Communication Layer
  3. Link back-end and front end
- Implement Cloud code
  - Business Plan

---

## **Successes**

---

Successes have been jumps in the code progress. Testing has been going well and progress has been made towards the end goal.

---

## **Issues and Changes**

---

Some issues that have been ran into have been

- Public/Private key passing for increased security
- Differences between iOS and android coding standards not allowing for similar looks between operating systems.
- Testing of mapping features

---

## **Team Details**

---

The team is going strong. With a busy semester, not all meeting times have worked out. But with a hard drive, we are working towards our goal of creating an app and starting our own business. We are still currently meeting with advisors to better our business plan and create marketing plans.

### **3 Sprint Report #3**



# Sprint Report #3

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

Crowd Control Group Management Mobile Application

### Company

Bowtaps

---

## Work Summary

---

- iOS
  1. Login
    - (a) Create User
    - (b) Facebook integration
  2. Mapping
  3. Working on Join Group
- Android
  1. Login
    - (a) Create User
    - (b) Facebook integration
  2. Mapping
  3. Working on Join Group
- Server
  1. Fixed Connection Issues
  2. User Connections Created

---

## Backlog

---

- Messaging API
- Join Group Implementation
- Cloud Code
  1. Group Clean Up
  2. User Information Links
- Business Plan
  1. South Dakota Giant Vision
  2. SDSM&T Business Plan Competition

---

## Success

---

Successes have been group team work towards the business plan competitions on the business side. On the development side was recreating some of the database to increase efficiency with parse. Logging in has been connected to Facebook accounts.

---

## Issues and Changes

---

Some issues that have been ran into have been

- Public/Private key passing for increased security
- Server connection issues from table to table with group creation
- Changes in the database schema
- GUI updates to more modern standards.

---

## Team Details

---

With business plan competitions, and the end of the semester, we have all been busy. We have come together to fix issues that where not planned for in the beginning, and furthered development of features in general.

The business plan and business plan competition are coming along well, and allowing us to focus more on the primary goals of the direction of the company, as well as development of Crowd Control.

## **4 Sprint Report Winter Sprint**

# Winter Sprint Report

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

CrowdControl - Group Management Mobile Application

### Company

Bowtaps

---

## Deliverables

---

- iOS
  1. Login/Logout
    - (a) Improved login/signup screens
    - (b) Logout feature added
  2. Settings
    - (a) Settings screen implemented
    - (b) Logout functionality nested in the Settings screen
  3. Groups
    - (a) Leaving/Joining a group implemented
    - (b) Basic group operations
    - (c) Detect if users are in a group
- Android
  1. Login
    - (a) Automatic login on startup (from datastore)
    - (b) Login to existing account via email address
  2. Settings
    - (a) Page layout created and linked from GroupJoin page
    - (b) Logout functionality implemented
  3. Groups

- (a) Leave button implemented
- (b) Tested adding/removing users from groups
- Misc/Transitional
  1. Further documented Android code to prepare for team merge
  2. Android code review with iOS team, to prepare for team merge

---

## Remaining Backlog

---

Here are the incomplete items/features for this sprint:

- Android
  - Messaging (Sinch API)
  - GPS Location (backend models)
  - Persistent groups through local datastore
- iOS
  - Messaging (Sinch API)

---

## Successes

---

- Android
  - Login through email
  - Settings page (layout and implementation)
  - Local Datastore (individual automatic login)
- iOS
  - Login/Logout
  - Settings page (layout and implementation)
  - Group functionality written

---

## Issues and Changes

---

Some issues that we encountered include:

- Android
  - Issues
    - \* Tried to manually create queries in the Parse API. We were unaware of built-in methods to accomplish the tasks. This set us back a bit.

- \* Encountered NullPointerException in the UserModel model. Had to change the structure to use an application global variable.
- Changes
  - \* Further development on Settings is now added to the backlog
  - \* Sign out functionality is now added to the backlog
  - \* Leave Group functionality is now added to the backlog
- iOS
  - Issues
    - \* Unexpected complications with database design
    - \* Layout complications
    - \* Issues with the underlying data models
    - \* Parallel programming complications
- Misc/Transitional
  - iOS development will be postponed, in favor of an Android prototype. This is to ensure that Android will meet expectations for the design fair.
  - Team communication and long-distance coordination was difficult.
  - Holidays and vacations impeded our ability to be productive.

---

## Team Details

---

Our team fell behind in the first semester, and in an effort to mitigate this, we allocated work towards the Winter Sprint. From here, unsatisfactory progress was still met, and we decided on another large refactor.

For the remainder of our project development, the iOS team will halt development and assist the Android team, so that Bowtaps can guarantee a satisfactory product for the design fair in Spring 2016.

Finally, to hopefully achieve better group management, we have elected Daniel Andrus to serve as acting Scrum Master.

## 5 Sprint Report #4

# Sprint Report #4

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

Crowd Control - Group Management Mobile Application

### Company

Bowtaps

---

## Backlog

---

The following items/features were assigned at the beginning of the sprint, and worked on throughout its duration. It is broken down by week as such:

### Week 1

- Android
  - Begin implementing Sinch
  - Create location and messaging views and managers
  - Design models and manager classes for messaging and location
  -
- Cloud Code
  - Group data parsing started

### Week 2

- Android
  - Broadcast/receive messages to/from all members in a group
  - Create a layout for messaging
  - Create a MapFragment to display a map
  - Created buttons overtop the MapFragment to correspond to syncing and homing locations
- Cloud code
  - Leaving and joining groups handled
  - Checking existing email upon login (validation)



## Week 3

- Android
  - Retrieve locations of group members, place their locations on the map via pins
  - Update group settings and data when changed
  - Update Group members if someone leaves or joins a group
  - Group messaging unit tests
  - GPS Location unit tests
- Cloud Code
  - Returning group information upon changes
  - Functional Group update indicator complete
  - Basic group functionality implemented fully (login/logout, join/leave groups, update on change)

Documentation and Business Plan work was carried out through all weeks of the sprint, and is ongoing.

---

## Deliverables

---

During this sprint, these are the items/features from the backlog that were successfully achieved:

- Android
  1. Group Messaging
    - (a) Created a Layout
    - (b) Used Sinch code to create a service
    - (c) Implemented group messaging
    - (d) Group messaging is working with no known bugs
  2. Location
    - (a) Page layout created and linked from GroupJoin page
    - (b) MapFragment has buttons for homing and syncing group locations
    - (c) Retrieving the user's location on instantiation of the MapFragment
    - (d) User and group locations implemented
  3. Group update service
    - (a) Checks for updates in near real-time
    - (b) Updates group settings when changed
    - (c) Updates group members if someone leaves or joins
- Server (cloud code)
  1. Functional Group update indicator
  2. Returning group update information
  3. Join group function (created but not functioning)
  4. Leave group function (created but not functioning)

## 5. Check for Existing Email

- Misc/Transitional

1. Business Plan filled out, also a version tailored towards the Governor's Giant Vision contest (converted to latex)
2. Documentation done inside and outside of the source code files

---

## Issues and Changes

---

Some issues that we encountered include:

- Android
  - Issues
    - \* Permissions to obtain contacts and locations from the device posed a challenge - still not handling the request gracefully
    - \* Had difficulty implementing a custom AlertDialogFragment that extends DialogFragment, inside of other fragments such as MapFragment, GroupInfoFragment, etc.
  - Changes
    - \* Added group update service - was not part of original backlog
    - \* Added user location homing on MapFragment - was not part of original backlog
- Server (cloud code)
  - Issues
    - \* Cloud functions improperly writing data.
  - Changes
    - \* Added join and leave cloud functions - was not part of original backlog

---

## Remaining Backlog

---

The following items/features remain either incomplete or need improvement for this sprint, and will carry onto the next sprint:

- Android
  - Group messaging unit tests
  - GPS Location unit tests
- Cloud Code
  - Testing on Group Functions.
  - Completing Join and Leave functions

---

## Team Details

---

Here are some auxiliary details about our workflow and division of responsibilities, during the sprint:

Dan and Johnny started off Sprint 4 by working on messaging-related features, while Joe and Evan worked on GPS and map features. Nick focused on the business plan, updating the existing documentation to use the updated layout, and was tasked with installing the Fabric SDK.

For week two of the sprint, Johnny focused on group messaging. Dan and Nick also worked together on cloud code, targeting the leaving/joining groups, and a group update service in Android. Evan wrote a `LocationManager` class and stubbed out methods, that Joe wrote an interface for and made a UI for in the `MapFragment`.

Week three continued with Joe and Evan working on various location features, while Dan and Johnny worked on the update service and messaging features respectively. Nick focused on cloud code and business plan/documentation writing.

Additionally, this was the first sprint in which we had Dan serve as acting Scrum Master, to aid in organization and appointment of responsibilities. Though he did officially take this role on during our Winter Sprint (Sprint 3.5), most of us were either working remotely or unavailable, thus we were unable to fully utilize this new organizational change until now.

## 6 Sprint Report #5

# Sprint Report #5

---

## Team Overview

---

### Name

Crowd Control

### Members

Johnathan Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, Joseph Mowry

### Project Title

Crowd Control - Group Management Mobile Application

### Company

Bowtaps

---

## Backlog

---

The following items/features were assigned at the beginning of the sprint, and worked on throughout its duration. It is broken down by week as such:

### Week 1

- Senior Design Doc
  - Do a general revision of the doc
  - Business Plan
    - \* Finish business plan for 2016 Governor's Giant Vision Competition
- Android
  - Model Caching/ Uniformity
  - Clean up appearance

### Week 2

- Android
  - Clean up the appearance of the app
  - Display Group Members on group info page
  - Safe group operations(leaving/joining group)
  - Loading animations on homing and syncing
- Cloud code
  - Safe group operations(leaving/joining group)

### Week 3

- Android
  - Integration Testing
  - Start Alpha Testing
- Cloud Code
  - test join and leave functionality

---

## Deliverables

---

During this sprint, these are the items/features from the backlog that were successfully achieved:

- Android
  1. Group Messaging
    - (a) Discovered and removed a bug
  2. Location
    - (a) Moved remote functionality to model manager
    - (b) Updated location model to reflect changes
    - (c) Caching objects
  3. App Appearance
    - (a) Reformatted the entire theme of the app (all pages are based of the same theme now - no more custom themes per page)
    - (b) Added a tool bar to the group join page/ removed settings button (now in tool bar)
    - (c) Group Information Page
      - i. Added a group leader display
      - ii. Added padding to appearance of display and modified text sizes
      - iii. Displays all group members
      - iv. Displays Dialog box if user attempts to leave the group
- Server (cloud code)
  1. Join function
  2. Leave function
- Misc/Transitional
  1. Business Plan revised and submitted to The Governor's Giant Vision Competition
  2. Finalist for The Governor's Giant Vision Competition
  3. Some of the overall Senior Design Doc has been touched up

---

## Issues and Changes

---

Some issues that we encountered include:

- Android
  - Issues
    - \* Another bug came up in messaging which took precious time from other parts of the sprint
  - Changes
    - \* Many code related things were pushed to later in the sprint
- Server (cloud code)
  - Issues
    - \* Unable to do proper stress tests
    - \* Unrepeatable errors popped up ( could have been a network error)
  - Changes
    - \* added more functions
    - \* more comments.

This is no excuse, but the team was seriously hindered by the amount of other responsibilities due during this sprint.

---

## Remaining Backlog

---

The following items/features remain either incomplete or need improvement for this sprint, and will carry onto the next sprint:

- Android
  - Messaging still needs an appearance update for usability
- Cloud Code
  - Testing on Group Functions.
  - Completing Join and Leave functions

---

## Team Details

---

Here are some auxiliary details about our work-flow and division of responsibilities, during the sprint:

The team focused heavily the first week on the Business Plan.

For week two of the sprint, very little was accomplished due to other responsibilities

Week three: Johnny was able to get many little things in the app to display better, such as more information on the group page. Joe put a lot of work into the theme and got custom pictures displaying in the tabs bar for groups. Evan got the map to function properly with syncing. Dan was able to finally fix the messaging bug. Nicks loud code was created to have safe group alteration functions such as joining and leaving groups.



# C

---

## Industrial Experience and Resumes

---

### 1 Resumes

Below are the resumes for the group members: Johnathon Ackerman, Daniel Andrus, Charles Bonn, Evan Hammer, and Joseph Mowry.

# Johnathan Ackerman

605-877-1757

Johnathan.ackerman@mines.sdsmt.edu

GitHub profile <https://github.com/Kiwii12>

## Education

South Dakota School of Mines and Technology

- **Computer Science Major**
- Start Date: Fall 2012
- Expected Graduation Date: **December 2016**
- Going for a Bachelor's Degree
- Enrolled Currently as a Senior

Central High School

- Graduated 2012

## Programs

### Team Projects

With Glut and C++, in teams of two, I have made the following:

- Pong ( [https://github.com/Kiwii12/CSC433\\_Program1\\_Pong](https://github.com/Kiwii12/CSC433_Program1_Pong) )
- Solar System Model ( [https://github.com/Kiwii12/CSC433\\_Program3\\_SolarSystem](https://github.com/Kiwii12/CSC433_Program3_SolarSystem) )

In C++

- Simulated a B17 computer ( <https://github.com/Kiwii12/B17> )

In Lisp

- Missionary Vs Cannibals ( <https://github.com/Kiwii12/missionaryVsCannibal> )

### Solo Projects

In C++

- WVX playlist creator ( <https://github.com/Kiwii12/WVX-Playlist-Creator> )
- Basic Picture Editor ( [https://github.com/Kiwii12/Basic\\_Picture\\_Editor](https://github.com/Kiwii12/Basic_Picture_Editor) )

## Skills

I have worked in the Operating Systems of Windows and Linux ( Fedora and Ubuntu )

I am very comfortable in **C++** and **Python**.

I am comfortable in **Android Studios**

I have also done work in SQL, HTML, Assembly, and PHP.

## Goals

I wish to work with computer graphics, in virtual reality or augmented reality.

## Work Experience

Pizza Ranch – 3 years, currently employed

- Rapid City, South Dakota, 57701
- 605-791-5255

# DANIEL ANDRUS

Phone: (605) 269-1728  
Email: danielandrus@gmail.com  
Twitter Handle: @deaboy100  
Github Name: Deaboy

## PROFILE

I am an undergraduate college student at the South Dakota School of Mines and Technology. I have a passion for video games and technology, and my career goal is to become a developer in the games industry, the mobile application industry, or the desktop application industry. I grew up in Los Angeles, California, then moved to South Dakota in Summer, 2010. I attended Black Hills State University for two years before transferring to South Dakota School of Mines and Technology, where I plan to graduate with a bachelors degree of computer science in May, 2016 and immediately begin working in software or game development.

## EXPERIENCE

### INTERN DEVELOPER, 7400 CIRCUITS — SUMMER 2015 - PRESENT

I held an internship at 7400 Circuits, a circuit board company located in Rapid City. Here I worked to improve an existing iOS and Android game called *Trouble with Robots*. I also worked on a cross-platform desktop application that interacted via USB with a handheld game cartridge reader and writer that allows users to create and play Neo Geo Pocket and WonderSwan games on their handheld game devices.

### SDSMT PROGRAMMING TEAM — 2014 - PRESENT

In fall 2014, I joined the SDSMT programming team and participated in the ACM regional Programming Competition where my team finished 14th in the region out of over 285 competing teams and 1st in the school.

### SERVER ADMINISTRATOR, PROGRAMMER — 2010 - PRESENT

Since 2010, I have owned and operated a public game server for which I and another developer have written hundreds of lines of server software to help manage the community. Through this, I have become greatly acquainted with Linux, SSH, and managing small communities.

### WEB DESIGNER AND DEVELOPER, BLACKHILLS.COM — 2013 - 2015

In May 2013, I started working for a local web development company as a full-time web developer. The job entailed designing and building websites of diverse sizes and varieties. Many sites were for small businesses located throughout the Black Hills, but a few were for large, high-traffic businesses such as BlackHillsNews.com and Sturgis.com.

### INTERN, FTW INTERACTIVE (NOW RED SHED TECHNOLOGY) — SUMMER 2012

I held an internship at FTW Interactive, now known as Red Shed Technology where I worked with experienced developers on mobile app projects. I gained experience working with server and client communications and data processing.

## SKILLS

- Programming in the **Java**, **C**, **C++**, **C#**, **PHP**, **Python**, **Objective-C**, and **Swift** programming languages.
- **OS X**, **iOS**, and **Android** development.
- Working with web technologies, including **HTML5**, **CSS**, **JavaScript**, and **PHP**.
- **Designing database systems** using **MySql**
- Working on **team projects**, **object-oriented program design**, and source control systems such as **Git** and **Subversion**

## EDUCATION

Black Hills State University, Spearfish, SD — 2010-2012

South Dakota School of Mines and Technology, Rapid City, SD — 2012-2016

## PERSONAL INFORMATION

I am good at math, am a fast learner, can pick up on new programming languages and standards quickly, and am a stickler for the proper usage of the word "literally". I can easily adapt to design patterns as well as programming paradigms and am perpetually learning the technologies and techniques employed in the software development, UX design, and games industries.

In my spare time, I enjoy playing and creating video games, creating YouTube videos, and learning more about the ever-changing technology industry. I love spending time with friends who enjoy similar things as I do. My career goals are to go into mobile application design and development, desktop application design and development, or game design and development. My ultimate personal goal with technology is to create applications that make people's lives better.

# C. Nicholas Bonn

---

2326 Lance Street, Rapid City SD 57702  
(651) 503-2877 charlesnicholasbonn@gmail.com

## Education:

**South Dakota School of Mines and Technology**, Rapid City, SD

*Bachelor of Science in Computer Science*

Anticipated Graduation: May 2016

Cumulative GPA: 2.5

### Relevant Coursework:

Database

Software Engineering

Cyber Security

Graphic User Interface

### Projects:

Crowd Control App – on-going senior design project

*Description:* a phone app designed to manage groups in a social setting, to track the members of the groups and ease social gatherings

## Technical Skills:

### **Languages:**

*Proficient in:* C/C++, Python, C#

*Familiar with:* Java, ARM Assembly, HTML/XML, Lisp, Qt Environment, Visual Basic

### **Other Technical Services:**

*Databases:* SQL Server, MySQL

*Platforms:* Microsoft Windows (Active Directory), Mac OSX, and Linux

## Work Experience:

**Discover Program - Rapid City School District**, Rapid City, SD

September 2009 – Current

*Program Assistant*

- Co-leader for after school and summer programs for elementary aged children
- Coordinate activities for 2nd and 3rd grade program
- Tutor children with their homework
- Mentor children and provide a positive environment for learning and activities

**TMI Coatings Inc.**, Eagan, MN

May 2012 – August 2012

*Summer Intern*

- Traveled to potential clients in Midwest region to collect specifications for job bids
- Drove equipment and job supplies to job sites in the Midwest
- Assisted in shop preparing equipment and supplies
- Oversaw scanning and organization of job components into electronic storage database

## Awards:

**Butterfield Cup**

May 2015

Award from local entrepreneurs to the best mobile app business plan, product and investor pitch

## References:

Available upon request

# Evan Paul Hammer

402 South St  
Rapid City, SD 57701  
Phone: 763-257-5060  
E-mail: evan.hammer@mines.sdsmt.edu

## Objective

Looking for a Full-Time opportunity in a competitive and leading edge company with a focus on intrapreneurship.

## Education

**South Dakota School Of Mines and Technology**, Rapid City, SD  
B.S. Computer Science; **GPA:** 2.9

**Expected Graduation:** May 2016  
August 2009 - Present

### Activities:

- Member in Triangle Fraternity, a fraternity of Engineers, Architects and Scientists
- Member of SDSM&T's Society of Mining, Metallurgy, and Exploration Engineers

## Experience

### Software Developer

May 2015 – Present

Golden West Telecommunications, Rapid City, SD

- Used mostly Python and JavaScript for development
- Mobile development with the use of Sencha Touch and Apache Cordova
- Proof of Concept work with SDK's and API's

### Operator

January 2014 – September 2014

Deadwood Biofuels, Rapid City, SD

- General shop cleaning
- Help with maintenance of equipment and Machines

### Night Chaperone/Office Assistant

September 2009 - July 2013

SDSM&T Youth Programs, Rapid City, SD

- Work with students attending the SDSM&T Engineering and Science camps.
- Teach the students about Engineering and Science
- Trained all the other chaperones and TA's
- Assisted in general office work

## Skills and Interests

### Leadership:

- Taught leadership skills to upcoming Boy Scout Leaders at a camp called Grey Wolf
- Eagle Scout

### Computer Science:

- C,C++, Python, ARM Assembly, JavaScript, Lisp
- Experience with Native Mobile Development
- Experience with Cross-Platform Development and MVC
- Experience with Open GL
- Operating Systems: Windows, Linux, Mac OS
- Experience in Database Management - MySql, PostgreSQL
- Experience with Git and Subversion

### Awards:

- Butterfield Cup - 2015

# JOSEPH MOWRY

## SKILLS

---

<b>Computer Languages</b>	C/C++, C#, ARM, SQL, HTML5, JavaScript, Java, Visual Basic, Python (3.X+)
<b>Protocols &amp; APIs</b>	JSON, XML, .NET, REST
<b>Databases</b>	Microsoft SQL
<b>Tools/Misc.</b>	GitHub, Mercurial(Hg), Team Foundation Server, Android Studio, Visual Studio, Xamarin, $\LaTeX$ , SQL Server Management Studio

## ORGANIZATIONS/MISC

---

- Educated in over four years of Spanish
- SDSM&T ACM Chapter Member
- SDSM&T Programming Team
- Attended the Black Hills Engineering Business Accelerator
- Awarded the Butterfield Cup for “Excellence in Software Engineering”

## WORK EXPERIENCE

---

PERIOD	<b>May 2015 — August 2015 (Full-Time)</b>	
EMPLOYER	<b>Innovative Systems</b>	Rapid City, SD
JOB TITLE	<b>Software Developer (Intern)</b>	
LANGUAGES	<b>C#, SQL, Xamarin.Forms, .NET Framework</b>	
	Cross-platform mobile development (MVVM) in Xamarin Forms, C# back-end development/stored procedures in MSSQL	
PERIOD	<b>May 2014 — August 2014 (Full-Time)</b>	
EMPLOYER	<b>Emit Technologies</b>	Sheridan, WY
JOB TITLE	<b>Software Developer (Intern)</b>	
LANGUAGES	<b>C#, JavaScript, HTML, .NET Framework, SQL</b>	
	Front-end (web) development in C#, stored procedures in MSSQL, followed MVC development pattern	

## EDUCATION

---

UNIVERSITY	<b>South Dakota School of Mines &amp; Technology</b>	
MAJOR	<b>B.S. in Computer Science</b>	
GPA	<b>2.7</b>	
GRAD DATE	<b>Spring 2016</b>	(Projected)

2326 LANCE STREET, RAPID CITY, SD, 57702 ·  
✉ JOE.MOWRY92@GMAIL.COM ☎ (605) 209-0208 ·  
[HTTPS://GITHUB.COM/JMOWRY](https://github.com/jmowry)

## 2 ABET: Industrial Experience Reports

As a group we have attended the SD Engineering Accelerator. We have competed in multiple business plan competitions including:

- Butterfield Cup
- SD Innovation Expo Business Plan Competition
- 2015 SD Mines CEO Student Business Plan Competition

We also have also have and regular meetings with SDSMT EIR's to help format our business plan and Crowd Control.

### 2.1 Johnathon Ackerman

I have had no Internship experience. However, before the project Crowd Control, I worked with C++, lisp, and python. I have worked with Visual Studios on Windows side, and Vim and G edit in Linux.

### 2.2 Daniel Andrus

I first learned the basics of web design and development in high school. After my second year of college, I obtained an internship with FTW Interactive (now known as Red Shed Technologies). Later, I hold a position as Web Developer for 2 years before becoming an intern software developer at 7400 Circuits.

My course experience has ranged from data structures, image processing, database design, web development, group projects, computer graphics (including 3D graphics), mobile app development, and even compression.

### 2.3 Charles Bonn

I currently have little internship experience. What industry experience i do have is HTML. In my personal/professional life i help manage a website and a minecraft server. Though this is work i have worked with HTML and C code. I have also worked with game code that is java based.

### 2.4 Evan Hammer

I am working for Golden West Telecommunications(GW), a rural telecommunications provider in the state of South Dakota. Since May of 2015 I have been a Software Developer for GW working on both mobile and back-end products. For the mobile side, I have been working with a product called Cordova that is wrapped with another product called Sencha Touch. Together these two products allow a developer to use JavaScript, HTML, CSS and more to produce a mobile application for Android, iOS and many other mobile platforms. I have also written the back-end for this app, using Python and a PostgreSQL Database creating a server-side API for the mobile application. While I am not working on the mobile application I have spent my time working on other in-house products using languages like Python and JavaScript. These projects have ranged from updating existing code to ground-up projects. Also as a Software Developer for GW, I have been tasked with creating some proof of concept work. This work has ranged from testing possible new services as well as testing new platforms for development. My work continues to grow and change as I continue to work for Golden West Telecommunications.

### 2.5 Joseph Mowry

In his prior industry experience, Joseph specialized in C# development and database management. His employers gave him a solid footing in AGILE and Scrum methodologies, as well as general product development. Though his experience lies primarily on the Visual Studio/C# side of things, there is a large amount of skill overlap in Android Studio and Java that he can bring to the table for this project.





## D

---

### Acknowledgment

---

As a special thanks we would like to thank Brian Butterfeild. His mentoring has made this project possible. Another thanks goes to Dr. Logar, With out your soft engineering class this would have never been possible.



# E

---

## Supporting Materials

---

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

