

---

# **Crowd Control**

---

## **Senior Design Final Documentation**

### **Bowtaps**

Johnathan Ackerman

Daniel Andrus

Charles Bonn

Evan Hammer

Joseph Mowry

May 2, 2016



---

# Contents

---

<b>Title</b>	i
<b>Contents</b>	vii
<b>List of Figures</b>	ix
<b>List of Tables</b>	xi
<b>List of Algorithms</b>	xiii
<b>Overview Statements</b>	xv
0.1 Mission Statement . . . . .	xv
0.2 Elevator Pitch . . . . .	xv
<b>Document Preparation and Updates</b>	xvii
<b>1 Overview and Concept of Operations</b>	1
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>	7
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 . . . . .	8
2.2.6 User Story #6 . . . . .	8
2.2.7 User Story #7 . . . . .	8

2.2.8 User Story #8 . . . . .	8
2.2.9 User Story #9 . . . . .	8
2.2.10 User Story #10 . . . . .	8
2.2.11 User Story #10 . . . . .	9
2.2.12 User Story #11 . . . . .	9
2.2.13 User Story #12 . . . . .	9
2.2.14 User Story #13 . . . . .	9
2.3 Requirements and Design Constraints . . . . .	9
2.3.1 System Requirements . . . . .	9
2.3.2 Network Requirements . . . . .	10
2.3.3 Development Environment Requirements . . . . .	10
2.3.4 Project Management Methodology . . . . .	10
2.4 Specifications . . . . .	10
2.5 Product Backlog . . . . .	10
2.6 Research or Proof of Concept Results . . . . .	11
2.6.1 iOS Proof of Concept Screen Shots . . . . .	11
2.6.2 Android Proof of Concept Screen Shots . . . . .	11
<b>3 Project Overview</b> . . . . .	<b>19</b>
3.1 Team Member's Roles . . . . .	19
3.2 Project Management Approach . . . . .	20
3.3 Stakeholder Information . . . . .	20
3.3.1 Customer or End User (Product Owner) . . . . .	20
3.3.2 Management or Instructor (Scrum Master) . . . . .	20
3.3.3 Investors . . . . .	20
3.3.4 Developers –Testers . . . . .	21
3.4 Budget . . . . .	21
3.5 Intellectual Property and Licensing . . . . .	21
3.6 Sprint Overview . . . . .	21
3.7 Terminology and Acronyms . . . . .	21
3.8 Sprint Schedule . . . . .	21
3.9 Timeline . . . . .	21
3.10 Backlogs . . . . .	22
3.10.1 Sprint 1 Backlog . . . . .	22
3.10.2 Sprint 2 Backlog . . . . .	23
3.10.3 Sprint 3 Backlog . . . . .	23
3.10.4 Sprint 3.5 Backlog . . . . .	23
3.10.5 Sprint 4 Backlog . . . . .	24
3.10.6 Sprint 5 Backlog . . . . .	25
3.10.7 Sprint 6 Backlog . . . . .	25
3.11 Development Environment . . . . .	26
3.12 Development IDE and Tools . . . . .	26
3.13 Source Control . . . . .	26
3.14 Build Environment . . . . .	27
3.15 Development Machine Setup . . . . .	27
<b>4 Design and Implementation</b> . . . . .	<b>29</b>
4.1 Architecture and System Design . . . . .	29
4.1.1 Design Selection . . . . .	30
4.1.2 Data Structures and Algorithms . . . . .	30
4.1.3 Data Flow . . . . .	30
4.1.4 Communications . . . . .	30
4.1.5 Classes . . . . .	30
4.1.6 UML . . . . .	30
4.1.7 GUI . . . . .	30

4.1.8 MVVM, etc . . . . .	30
4.2 Major Component #1 . . . . .	30
4.2.1 Technologies Used . . . . .	30
4.2.2 Component Overview . . . . .	30
4.2.3 Phase Overview . . . . .	30
4.2.4 Architecture Diagram . . . . .	30
4.2.5 Data Flow Diagram . . . . .	30
4.2.6 Design Details . . . . .	30
4.3 Major Component #2 . . . . .	31
4.3.1 Technologies Used . . . . .	31
4.3.2 Component Overview . . . . .	31
4.3.3 Phase Overview . . . . .	31
4.3.4 Architecture Diagram . . . . .	31
4.3.5 Data Flow Diagram . . . . .	31
4.3.6 Design Details . . . . .	31
4.4 Major Component #3 . . . . .	31
4.4.1 Technologies Used . . . . .	31
4.4.2 Component Overview . . . . .	31
4.4.3 Phase Overview . . . . .	32
4.4.4 Architecture Diagram . . . . .	32
4.4.5 Data Flow Diagram . . . . .	32
4.4.6 Design Details . . . . .	32
<b>5 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 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 . . . . .	36
6.2.1 Deliverable . . . . .	36
6.2.2 Backlog . . . . .	37
6.2.3 Success/Fail . . . . .	37
6.3 Sprint 3 Prototype . . . . .	37
6.3.1 Deliverable . . . . .	37
6.3.2 Backlog . . . . .	38
6.3.3 Success/Fail . . . . .	38
6.4 Winter Sprint Prototype . . . . .	39
6.4.1 Deliverable . . . . .	39
6.4.2 Backlog . . . . .	39
6.4.3 Success/Fail . . . . .	40
6.5 Sprint 4 Prototype . . . . .	40
6.5.1 Deliverable . . . . .	40
6.5.2 Backlog . . . . .	41
6.5.3 Success/Fail . . . . .	41

6.6	Sprint 5 Prototype . . . . .	42
6.6.1	Deliverable . . . . .	42
6.6.2	Backlog . . . . .	42
6.6.3	Success/Fail . . . . .	43
6.7	Sprint 6 Prototype . . . . .	43
6.7.1	Deliverable . . . . .	43
6.7.2	Backlog . . . . .	43
6.7.3	Success/Fail . . . . .	44
<b>7</b>	<b>Release – Setup – Deployment</b>	<b>49</b>
7.1	Deployment Information and Dependencies . . . . .	49
7.2	Setup Information . . . . .	49
7.3	System Versioning Information . . . . .	49
<b>8</b>	<b>User Documentation</b>	<b>51</b>
8.1	User Guide . . . . .	51
8.2	Installation Guide . . . . .	51
8.3	Programmer Manual . . . . .	51
<b>9</b>	<b>Class Index</b>	<b>53</b>
9.1	Class List . . . . .	53
<b>10</b>	<b>Class Documentation</b>	<b>55</b>
10.1	Poly Class Reference . . . . .	55
10.1.1	Constructor & Destructor Documentation . . . . .	55
10.1.2	Member Function Documentation . . . . .	55
<b>11</b>	<b>Business Plan</b>	<b>57</b>
<b>Bibliography</b>		<b>59</b>
<b>Software Agreement</b>		<b>SA-1</b>
<b>A 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 Management Features . . . . .	A-1
4	Parse Features . . . . .	A-1
<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 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 Acknowledgment</b>	<b>D-1</b>
<b>E Supporting Materials</b>	<b>E-1</b>



---

## **List of Figures**

---

1.1	Basic System Flow Diagram	3
2.1	iOS login select screen	12
2.2	iOS email login screen	12
2.3	iOS create account screen	13
2.4	iOS group infomation screen	13
2.5	iOS map view screen	14
2.6	iOS messaging main screen	14
2.7	Android login screen	15
2.8	Android create group screen	15
2.9	Android group information screen	16
2.10	Android group join screen	16
2.11	Android messaging main screen	17
6.1	Sprint 2 Prototypes.	36
6.2	Sprint 3 Prototypes.	38
6.3	Winter Sprint Prototypes.	45
6.4	Sprint 4 Prototypes.	46
6.5	Sprint 5 Prototypes.	47
6.6	Sprint 6 Prototypes.	48



---

## **List of Tables**

---



---

## **List of Algorithms**

---

1	Calculate $y = x^n$	29
---	---------------------	----



---

## **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**

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



# 1

---

## 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.

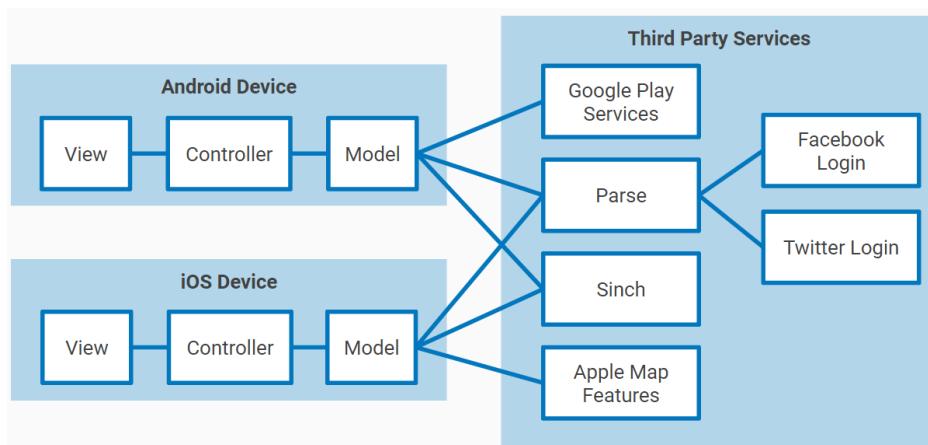


Figure 1.1: Basic System Flow Diagram

## 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 ables 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 section contains the features, creation, and development of crowd control. It covers prerequisite user stories, to the design and implementation of the application its self.

## 2.2 User Stories

### 2.2.1 User Story #1

As a user I want to join a public group.

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

The user can request to join any public group. Their request will be sent to the group leader for approval. Depending on the group leaders action then the user will be accepted or denied access to the group.

### 2.2.2 User Story #2

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

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

For the user to join a private group they must first recieve an invitation from the group leader of the private group of interest. Then the user can either accept or reject the invitation to join the group. If the invitation is accepted then the user is placed into the group and then starts sharing information.

### 2.2.3 User Story #3

As a user i want to see where the other members of my group are on a map

#### 2.2.3.a User Story #3 Breakdown

Once the user is in a group then the user will have access to a map that displays markers representing the locations of all users that are members of the group.

### 2.2.4 User Story #4

As a user i want to see a list of public groups nearby

### **2.2.4.a User Story #4 Breakdown**

For a user to have quick access to groups nearby that have been listed as public, once the user has logged in a list of public groups near the users current location will be displayed.

### **2.2.5 User Story #5**

As a user i want to have suggestions of local activities and locations.

### **2.2.5.a User Story #5 Breakdown**

Once the user joins a group there will be a tab that lists possible locations and activities nearby the user. These suggestions will be generated by local events and businesses that advertise with Crowd Control.

### **2.2.6 User Story #6**

As a user i want to leave a group.

### **2.2.6.a User Story #6**

Much like joining a group the user would like to leave a group at any time and this process to be as painless as possible. Using a button on the main group page will allow a user to leave the current group for whatever their reason.

### **2.2.7 User Story #7**

As a user I want the ability to login with a custom account.

### **2.2.7.a User Story #7 Breakdown**

For a user to be registered with our system they will have to login. This gives the option for the user to login with a custom Bowtaps login.

### **2.2.8 User Story #8**

As a user I want to login with Facebook

### **2.2.8.a User Story #8 Breakdown**

If the user does not want to create a custom Bowtaps login to have access to our app, then they can login with their facebook account. This gives the user easy access to the app and deals with authentication with facebook.

### **2.2.9 User Story #9**

As a user I want to login with Twitter

### **2.2.9.a User Story #9 Breakdown**

If the user does not want to create a custom Bowtaps login to have access to our app, then they can login with their Twitter account. This gives the user easy access to the app and authentication takes place with Twitter.

### **2.2.10 User Story #10**

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

### 2.2.10.a User Story #10

To keep users within our app rather than having to use another app to send a message to the group, the user can access the message tab once inside a group. Then the user can send a single message to each member of the group.

### 2.2.11 User Story #10

As a user i would like my information protected.

### 2.2.11.a User Story #10 Breakdown

To users data protection is a major concern. In the app all of the communication is done with user data protection in mind. All of the data being passed back and forth between users will be sent using encrypted strings.

### 2.2.12 User Story #11

As a group leader I want to be able to create an Itinerary for the group

### 2.2.12.a User Story #11 Breakdown

The group leader can create an Itinerary for all group members to be able to view. This can contain waypoints or just times when special events in the group will occur.

### 2.2.13 User Story #12

As a user I want to be able to look at my groups itinerary.

### 2.2.13.a User Story #12 Breakdown

If the group leader has created an itinerary for the group the users would like access to see upcoming group events and locations.

### 2.2.14 User Story #13

As a group leader I would like to invite members to a private group

### 2.2.14.a User Story #13 Breakdown

To join a private group the group leader must send an invitation to the members that would like to join.

## 2.3 Requirements and Design Constraints

This section contains the requirements and constraints of all aspects of Crowd Control. This includes requirements and constraints for both iOS and Android operating systems, server side, and network.

### 2.3.1 System Requirements

Crowd Control is being developed to run on two different mobile operating systems, iOS and Android. Although both applications will adhear to the same user stories, some requirements and implementation details differ. Below each operating system is outlined with its specific requirements.

#### 2.3.1.a iOS Requirements

- Use Apple Mapping Features
- Connect with Parse API for backend data storage

### **2.3.1.b Android Requirements**

- Use Google Maps
- Connect with Parse API for backend data storage

### **2.3.1.c Parse Requirements**

- Add users to groups
- Remove users from groups

## **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**

To develop Crowd Control natively for Android and iOS, two different IDEs will be used. For iOS, development must be done on an Apple device. The IDE used is XCode which has all of the necessary SDKs and emulation software to develop and test the iOS version of Crowd Control. For Android, development happens on the Android Studio IDE which like Apple's XCode, also contains the necessary development tools required to develop for Android. Android Studio can be run on either Windows and Mac.

## **2.3.4 Project Management Methodology**

We have set restrictions on the development 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 3 total sprints over 2 semesters for this product.
- The sprint cycles are 3 weeks long.
- Progress reports will be submitted to Dr. McGough and Brian Butterfeild at the end of each sprint.
- 4 GitHub repositories will be used for source control, one for each platform, one for the server code, and one for documentation.

## **2.4 Specifications**

Crowd Control is to be built for the two largest mobile operating systems, Android and iOS. This choice was made to support the two most used smartphone operating systems because between these two, they support 94.7% of the smartphone market. Also considering the time restriction of full-time students and the length of Senior Design, the main operating system for development will be Android.

## **2.5 Product Backlog**

- UX
  - Create groups
  - Leave groups
  - Group messaging
  - Start page

- Mapping features
- Messaging UI
- Database
  - Design database schema
  - Implement database on Parse
- Cloud Code
  - Safe group operations(leaving/joining group)
- Android
  - User Model
  - Communication Layer
  - Link back-end and front end
  - Messaging API
  - Join Group Implementation
  - Settings screen implemented
  - Logout functionality nested in the Settings screen
  - Implement Sinch
  - Create location and messaging views and managers
  - Design models and manager classes for messaging and location
  - Retrieve locations of group members, place their locations on the map via pins
  - Model Caching/ Uniformity
- iOS
  - User Model
  - Communication Layer
  - Link back-end and front end
  - Messaging API
  - Join Group Implementation
  - Model Manager
- Business Plan

## 2.6 Research or Proof of Concept Results

The Proof of concept is a rough design that implements 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 Crowd Control.

### 2.6.2 Android Proof of Concept Screen Shots

Below are screen shots from the Android version of CrowdControl.

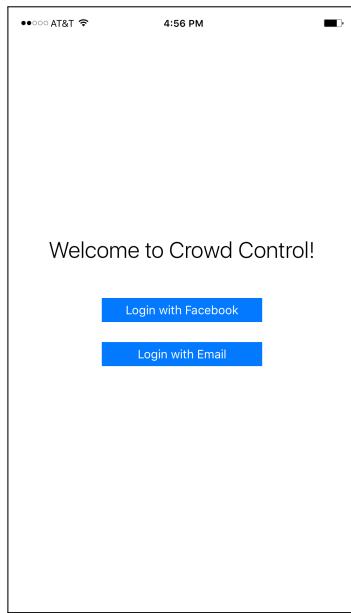


Figure 2.1: iOS login select screen

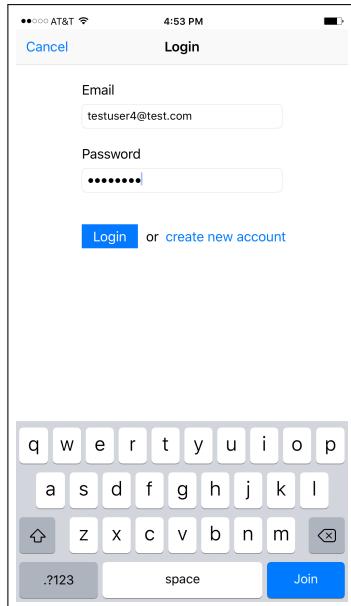


Figure 2.2: iOS email login screen

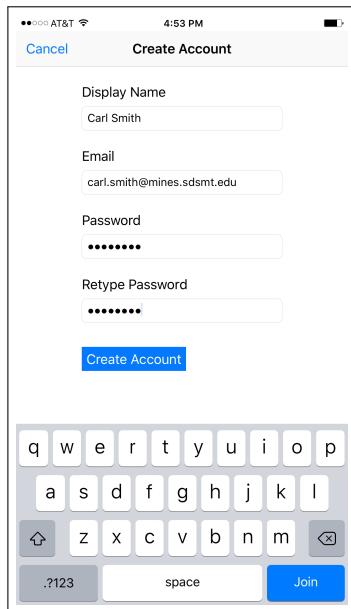


Figure 2.3: iOS create account screen

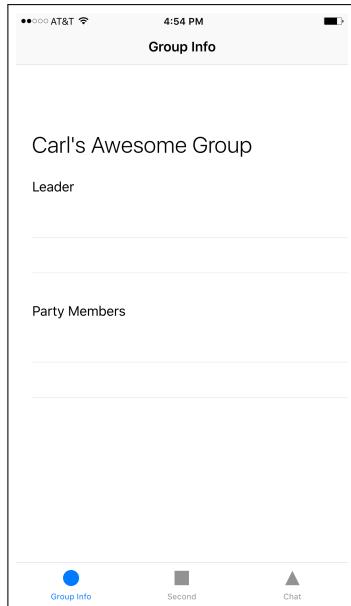


Figure 2.4: iOS group infomation screen

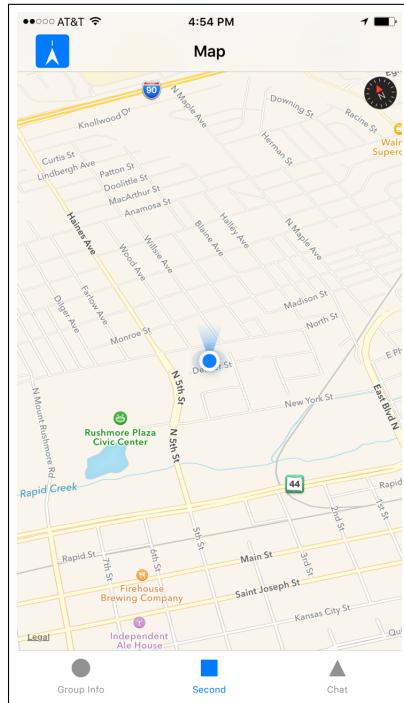


Figure 2.5: iOS map view screen

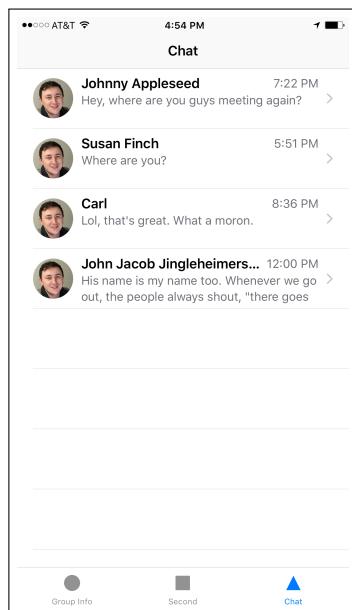


Figure 2.6: iOS messaging main screen

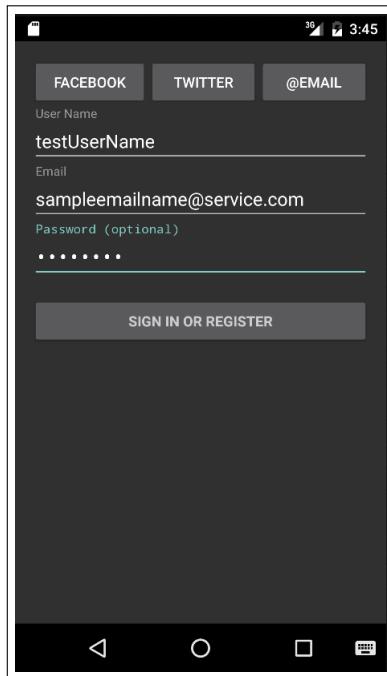


Figure 2.7: Android login screen

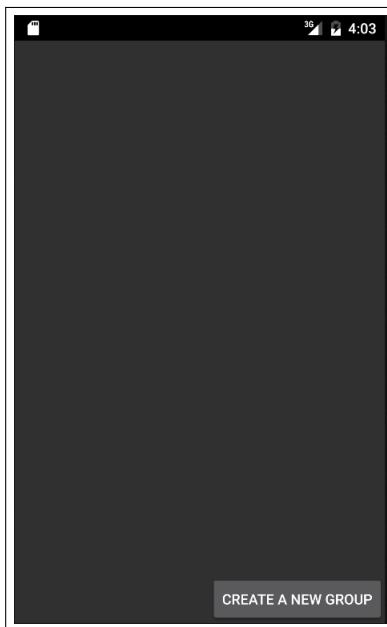


Figure 2.8: Android create group screen

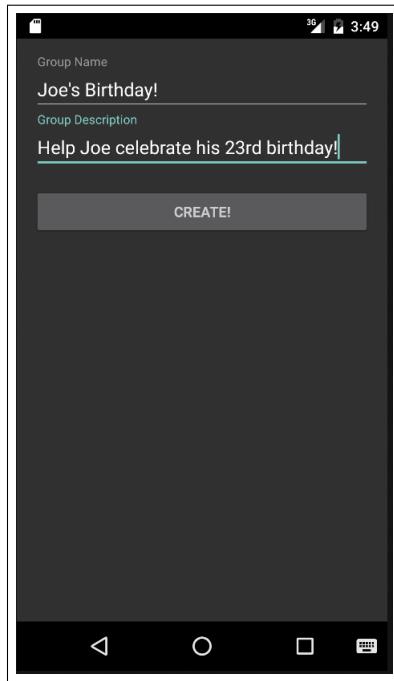


Figure 2.9: Android group information screen

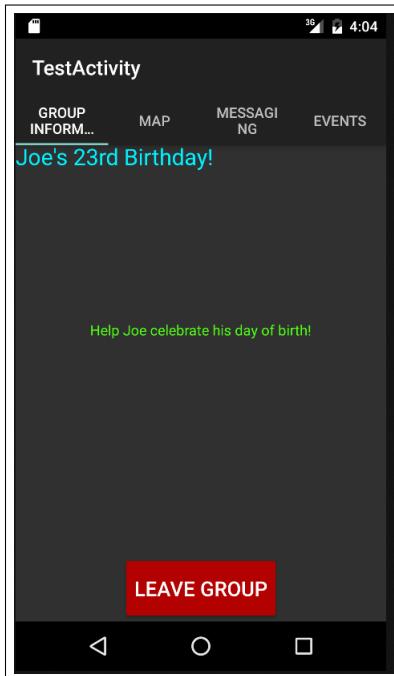


Figure 2.10: Android group join screen

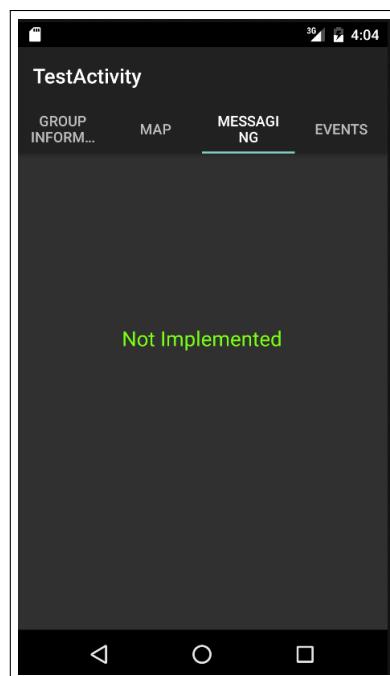


Figure 2.11: Android messaging main screen



# 3

---

## Project Overview

---

Outlined within this section are details involving the development of Crowd Control. This chapter includes details about the team organization and structure. Further included is information outlining the financial aspects and the development requirements in these regards as well.

### 3.1 Team Member's Roles

Each team member plays a pivotal role in the development of Crowd Control. Listed below are each members and their corresponding responsibilities within the production of Crowd Control. Some of the duties of each member changed half way through production as development of iOS was halted for more progress on Android. The entire projects duties are outlined below for each individual developer.

**Johnathan Ackerman**- Johnathan is leading the front end design and messaging implementation for the Android version of Crowd Control. This entails:

1. Creating and designing GUI elements for android
2. Designing messaging Layout
3. Implementing logic behind group join, group info, and messaging views for Android

**Daniel Andrus** - Daniel for the first half was leading the Gui design ad implementation for the iOS version of Crowd Control. During the second half he worked with Johnathan on Messaging as well as creating our background service. Daniel has also been the structural designer for the model classes. These duties entail:

1. Creating and designing GUI elements for iOS.
2. Implementing logic behind most views for iOS
3. Implementing logic behind messaging for Android
4. Model structure design
5. Creating and implementing a background group service for Android that checks the server for group data updates

**Charles Bonn** - Charles is leading the server side of Crowd Control which involves server functions for both iOS and Android versions. This entails:

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

**Evan Hammer** - Evan was leading the backend side for the iOS version of Crowd Control for the first half of development. During the second half of development Evan headed up the location handling and management for Android. This entails:

1. Implementing models for iOS
2. Creating Location Managers for both iOS and Android
3. Organizing location updates automatically between users
4. Working with Apple Maps and Google Maps

**Joseph Mowry** - Joseph is leading the backend side for the Android version of Crowd Control. Also Joe has been involved with the creation of location management for the Android version. This entails:

1. Implementing models for Android
2. Organizing location updates automatically between users
3. Working with Google Maps

## 3.2 Project Management Approach

Crowd Control was developed using the Agile software development method. The project was split up into 7 sprints each lasting 3 weeks. For the first 4 sprints, each sprint was planned out with goals for each sprint. The last 3 sprints had plans for each week, creating goals for all of the weeks inside of the sprints. To track sprint goals, meeting minutes were used to define each members goals at the beginning of each sprint. Another way to track goals and bugs along the way was to use the GitHub Issue tracker, each issue contained data about the feature or bug and any questions or extra data necessary. All of this was wrapped up in scrum meetings five days a week. Scrum meetings are stand-up meetings usually lasting five to ten minutes were each member would report on what they had accomplished, what they were working on next, and if there were any issues holding back progress.

## 3.3 Stakeholder Information

The stakeholders of this project were the members of Bowtaps LLC. These five members were the same five members developing Crowd Control, their names are listed above in section 3.1.

### 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)

For Crowd Control, Bowtaps chose to make Daniel Andrus the scrum master. His duties included helping guide scrum meetings and helping to break up features into week long goals.

### 3.3.3 Investors

Currently Bowtaps has not sought out investment from external sources, instead the members developing Crowd Control have done so using funding from other sources. To fund the development of Crowd Control the team has entered into a few competitions with their intellectual property to help generate funds. The team has competed in the South Dakota Governor's Giant Vision, The Innovation Expo, and the Mines CEO Business Plan competitions to gain capitol to cover their current costs.

### 3.3.4 Developers –Testers

For the term of our development each member of the team was a developer and a tester. This meant that before a member of the team merged their code from their branch to a main branch that they would go through and thoroughly test.

## 3.4 Budget

The budget for Crowd Control is fairly simple. For starting out and before a user base has been developed, e.g. less than 1000 users, Crowd Control can be developed for free sans developer salaries. For this project Bowtaps has run on a complete development track where all members of the group have not received a salary for any of their work. The only funds that have been spent by Bowtaps have not involved the development of Crowd Control and only have been for Bowtaps themselves.

## 3.5 Intellectual Property and Licensing

The intellectual property Crowd Control is owned by Bowtaps LLC which is currently made up of the team currently developing Crowd Control. All source code, documentation and presentation materials are protected by copyright.

## 3.6 Sprint Overview

Crowd Control has been developed in seven phases or sprints. Each sprint has spanned three weeks and each will have a unique set of goals for the development of Crowd Control. Further in this document each sprint will be broken down into its own goals and also outlined are each of their successes and issues.

## 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.

- iOS - Apple's mobile operating system for smartphones
- Android - Google's mobile operating system for smartphones

## 3.8 Sprint Schedule

Below is a table of dates for each sprint.

Sprint	Date
Sprint 1	9/14/2015 - 10/2/2015
Sprint 2	10/12/2015 - 10/30/2015
Sprint 3	11/9/2015 - 11/27/2015
Sprint 3.5	12/21/2015 - 1/8/2016
Sprint 4	1/18/2016 - 2/5/2016
Sprint 5	2/15/2016 - 3/4/2016
Sprint 6	3/21/2016 - 4/15/2016

## 3.9 Timeline

Below is an overview of the timeline of the project by sprint.

Sprint	Tasks	Date
Sprint 1	Design UX	10/2/2015
	Design Database	10/2/2015
	Design Application Layers	10/2/2015
	Set up GitHub repository	10/2/2015
Sprint 2	Code UX	10/30/2015
	Create Models	10/30/2015
	Research public/private key passing	10/30/2015
Sprint 3	iOS Login	11/27/2015
	iOS Facebook integration	11/27/2015
	Mapping	11/27/2015
	Work on Group Join	11/27/2015
Sprint 3.5	iOS Logout	1/8/2016
	iOS Settings Page	1/8/2016
	iOS Group Join/Leave	1/8/2016
	Android Automatic Login	1/8/2016
	Android Settings Page	1/8/2016
	Android Group Leave	1/8/2016
Sprint 4	Begin implementing Sinch	1/22/2016
	Create location and messaging views and managers	1/22/2016
	Design models and manager classes for messaging and location	1/22/2016
	Broadcast/receive messages to/from all members in a group	1/29/2016
	Create a layout for messaging	1/29/2016
	Create a MapFragment to display a map	1/29/2016
	Leaving and joining groups handled on Cloud Code	1/29/2016
	Retrieve locations of group members, place their locations on the map via pins	2/5/2016
	Update group settings and data when changed	2/5/2016
	Update Group members if someone leaves or joins a group	2/5/2016
Sprint 5	Moved Location functionality to Model Manager	2/20/2016
	Caching of all objects	2/27/2016
	Android application theme upgrade	2/27/2016
	Group information displayed in more detail	3/4/2016
	Cloud functions created for group join/leave	3/4/2016
Sprint 6	Updated group leader functionality	3/21/2016
	Messaging Polished	3/21/2016
	Created notification view layer and started implementation	3/27/2016
	Worked out bugs for Giant Vision Demo	4/8/2016

## 3.10 Backlogs

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

### 3.10.1 Sprint 1 Backlog

- 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

### 3.10.2 Sprint 2 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

### 3.10.3 Sprint 3 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

### 3.10.4 Sprint 3.5 Backlog

- iOS
  - 1. Login/Logout
    - (a) Improved login/sign up 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 data store)

- (b) Login to existing account via email address
- 2. Settings
  - (a) Page layout created and linked from Group Join 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

### 3.10.5 Sprint 4 Backlog

#### 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)

### 3.10.6 Sprint 5 Backlog

#### 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

### 3.10.7 Sprint 6 Backlog

#### Week 1

- Android
  - Messaging
    - \* Messages now include the names of the sender
    - \* Leader can now kick or promote members
    - \* Messages now load per group from parse
  - Group Management Tools
    - \* Leader can now kick a member
    - \* Leader can now promote a member
  - Location
    - \* Update location automatically using service
    - \* Set Group Location

#### Week 2

- Android
  - fixed leader bug, now leader loads properly
  - broke ground for notification system
    - \* created tab system for invites and accepts

- \* created fragments for invite and confirm
- \* created model for notification system
- \* items can be transferred from the invite fragment to the confirm fragment
- Option Menu's added
  - \* group join now has an option menu
  - \* settings activity moved to option menu
  - \* Group name can be changed in option menu
  - \* option menu is different if you are a group leader
  - \* option menu leads to invite system
  - \* option menu leads to blank notification page
- Reformatted settings activity
  - \* now displays and can change display name
  - \* displays a current group if in one
  - \* added a finish button for clarity
- Cloud code
  - Safe group operations(leaving/joining group)

### Week 3

- Giant Vision Competition
  - Create the pitch
  - Create expo supporting materials

## 3.11 Development Environment

To develop Crowd Control the team used a few different products to develop, test, and run. For development of Crowd Control for Android, the team used the Android Studio IDE which supports all aspects of Android development. Using Android Studio's Layout Editor to help create the GUI, and using its built in support for Java to code the controller and model layers. The Android Studio download online comes with all of the necessary products to build, run, and test Crowd Control. The iOS development all took place on XCode, an Apple Development IDE, much like Android Studio XCode contains a layout editor and all of the necessary build scripts to build, run, and test the iOS version of Crowd Control. For the back end of development, the team chose to use parse and created two databases on the platform. One database for development and one for deployment. Both databases are wrapped by the app so there is little trouble to switch the code when time for deployment.

## 3.12 Development IDE and Tools

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

Android Android Studio - <http://developer.android.com/sdk/index.html>

iOS XCode - Xcode can be found on the Apple App Store for free

Parse Parse Accounts can be obtained at [parse.com](http://parse.com)

## 3.13 Source Control

To manage source control of Crowd Control, Bowtaps is currently using Git, and the server is hosted by GitHub.

## 3.14 Build Environment

To build Crowd Control there is little difference on the developers end between Android and iOS. On the Android side, Android Studio uses a build environment called Gradle that builds and compiles the source code and packages it into an apk. However for the developer, little has to be done besides running the Gradle scripts to receive an apk. The only thing to note on the Android build phases would be that there must be a match between the Java Version used for building and compilation and the SDK version. This correlation can be found online as new SDK release and as new Java versions are available. On the iOS side there is even less management necessary. For iOS, XCode manages all of the building and compiling of the Swift code. Only at the beginning of a project or upon the choice to update the lowest supported version of iOS are changes necessary to the build settings. However, if needed these settings can be accessed in XCode under the build settings tab.

## 3.15 Development Machine Setup

To develop Crowd Control for iOS or Android on an Apple device, such as a MacBook or other Apple Product, the only setup required is to download the IDEs necessary. On the windows side however it should be noted that after installing the Android Studio IDE depending on the type of physical device used for testing, there may be necessary driver downloads to interface with Android Studio's built in ADB(Android Debug Bridge). These drivers can be found online either from the manufacturer of the mobile device or some can be found on third party sites depending on the phone.



# 4

---

## Design and Implementation

---

This section is used to describe the design details for each of the major components in the system. Note that this chapter is critical for all tracks. Research tracks would do experimental design here where other tracks would include the engineering design aspects. This section is not brief and requires the necessary detail that can be used by the reader to truly understand the architecture and implementation details without having to dig into the code. Sample algorithm: Algorithm 1. This algorithm environment is automatically placed - meaning it floats. You don't have to worry about placement or numbering.

---

**Algorithm 1** Calculate  $y = x^n$ 

---

**Require:**  $n \geq 0 \vee x \neq 0$

**Ensure:**  $y = x^n$

```
 $y \Leftarrow 1$ 
if  $n < 0$  then
     $X \Leftarrow 1/x$ 
     $N \Leftarrow -n$ 
else
     $X \Leftarrow x$ 
     $N \Leftarrow n$ 
end if
while  $N \neq 0$  do
    if  $N$  is even then
         $X \Leftarrow X \times X$ 
         $N \Leftarrow N/2$ 
    else { $N$  is odd}
         $y \Leftarrow y \times X$ 
         $N \Leftarrow N - 1$ 
    end if
end while
```

---

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 is where you will place the overall system design or the architecture. This section should be image rich. There is the old phrase *a picture is worth a thousand words*, in this class it could be worth a hundred points (well if you sum up over the entire team). One needs to enter the design and why a particular design has been done.

### 4.1.1 Design Selection

Failed designs, design ideas, rejected designs here.

### 4.1.2 Data Structures and Algorithms

Describe the special data structures and any special algorithms.

### 4.1.3 Data Flow

### 4.1.4 Communications

### 4.1.5 Classes

### 4.1.6 UML

### 4.1.7 GUI

### 4.1.8 MVVM, etc

## 4.2 Major Component #1

### 4.2.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.2.2 Component Overview

This section can take the form of a list of features.

### 4.2.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

### 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.

### 4.2.5 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.

### 4.2.6 Design Details

This is where the details are presented and may contain subsections. Here is an example code listing:

```
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;
```

```
// Line comment.  
puts("Hello world!");  
  
for (i = 0; i < N; i++)  
{  
    puts("LaTeX is also great for programmers!");  
}  
  
return 0;  
}
```

This code listing is not floating or automatically numbered. If you want auto-numbering, put it in the algorithm environment (not algorithmic however) shown above.

## 4.3 Major Component #2

### 4.3.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.3.2 Component Overview

This section can take the form of a list of features.

### 4.3.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

### 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 Major Component #3

### 4.4.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.4.2 Component Overview

This section can take the form of a list of features.

#### **4.4.3 Phase Overview**

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

#### **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

This sprint was focused on learning Android and iOS development tools, as well as designing the UX and the database schema.

#### 6.1.1 Deliverable

- UX for Android
- UX for iOS

#### 6.1.2 Backlog

- UX design (MVC)
  - Create group
  - Leave group
  - Group messaging
  - Start page
- Database
  - Design database schema
  - Implement database on Parse

#### 6.1.3 Success/Fail

- 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

## 6.2 Sprint 2 Prototype



Figure 6.1: Sprint 2 Prototypes.

### 6.2.1 Deliverable

- Android
  - UX
    - \* Designed map screen
    - \* Designed group info screen
    - \* Designed start page
    - \* Designed group messaging
    - \* Created initial tabs holder
  - Model
    - \* Created skeleton user model
    - \* Created skeleton group model
    - \* Created skeleton abstractions for models
- iOS
  - UX
    - \* Created map display
    - \* Added homing button to map display
    - \* Created simple group display
  - Model
    - \* Created skeleton user model
    - \* Created skeleton profile model

- \* Created skeleton group model
- \* Created skeleton abstractions for models
- Parse
  - Created User table
  - Created CCUser table
  - Created Group table
- Work was also done on the Business Plan

### 6.2.2 Backlog

- Code UX
  - Implement mapping features
  - Create messaging UI
- Model
  - Create user Model
  - Create communication Layer
  - Link back-end and front end
- Create parse tables
- Business Plan

### 6.2.3 Success/Fail

- We had a hard time finding a version of Public/Private key encryption that fit our project.
- Differences between iOS and android coding standards made it hard to create a similar look and feel.
- iOS is simply easier to make these initial pieces in, and has jumped far ahead android.
- iOS map features are proving very difficult to test.

## 6.3 Sprint 3 Prototype

### 6.3.1 Deliverable

- iOS
  - Created a log-in system
    - \* Added ability to create a user
    - \* Added Facebook integration for user creation
  - Added ability for users to join an existing group
- Android
  - Added ability to create a user with an email address
  - Added group functionality
    - \* Added a way for groups to be created
    - \* Added a group join feature
- Server
  - Fixed connection issues between parse and app
  - User table and profile(CCUser) are connected by apps in parse
- Prepped business plan for SDSM&T Business Plan Competition

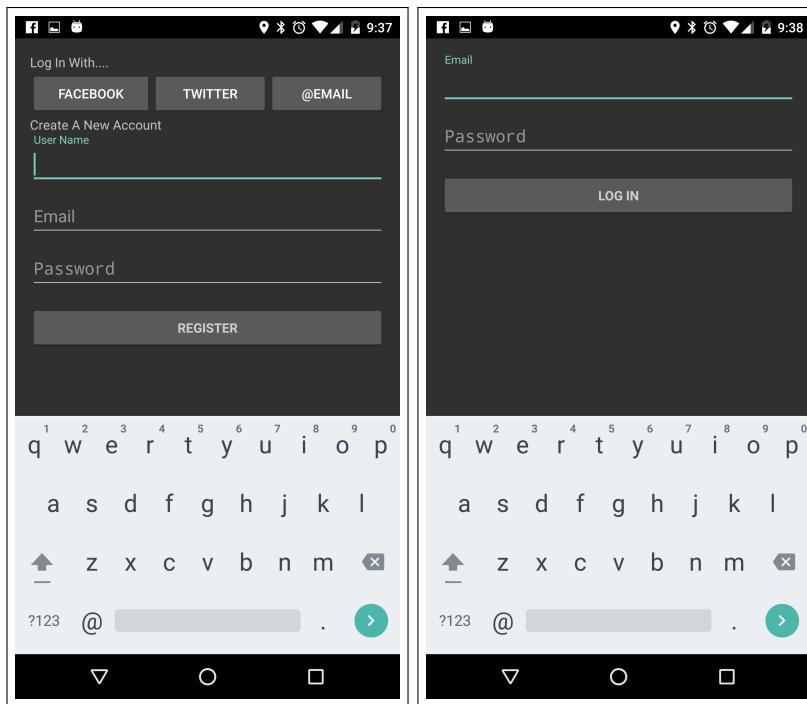


Figure 6.2: Sprint 3 Prototypes.

### 6.3.2 Backlog

- Install a messaging API
- Allow users to join a group
- Cloud Code
  - Create safe group operations
  - Link user information
- Business Plan
  - Submit to SDSM&T Business Plan Competition
  - Prep for South Dakota Giant Vision

### 6.3.3 Success/Fail

- Failures
  - Not all of the group functionality is working properly
  - We were unable to get to messaging this sprint
- Changes
  - Rewrote the database schema
  - Modified GUI appearance
- Successes
  - iOS was successful in connecting to Facebook
  - Android managed to start creating groups
  - Our business plan was accepted by SDSM&T Business Plan Competition

## 6.4 Winter Sprint Prototype

### 6.4.1 Deliverable

- iOS
  - Log-in/Log-out
    - \* Improved Log-in/sign-up screens
    - \* Added Log-out feature
  - Settings
    - \* Implemented settings screen
    - \* Nested log-out functionality into settings screen
  - Groups
    - \* Implemented leaving/Joining a group
    - \* Improved basic group operations
    - \* Detects if users are in a group
- Android
  - Log-in
    - \* Automatic log-in on start-up (from data-store)
    - \* Log-in to existing account via email address
  - Settings
    - \* Page layout created and linked from GroupJoin page
    - \* Log-out functionality implemented
  - Groups
    - \* Leave button implemented
    - \* Tested adding/removing users from groups
- Misc/Transitional
  - Further documented Android code to prepare for team merge
  - Android code review with iOS team, to prepare for team merge

### 6.4.2 Backlog

- Android
  - Messaging (Sinch API)
  - GPS Location (back-end models)
  - Persistent groups through local data-store
- iOS
  - Messaging (Sinch API)

### 6.4.3 Success/Fail

- - Messaging wasn't touched
  - Android is still missing GPS location work
  - User does not automatically connect to a group if they are in one
- Successes
  - Android
    - \* Log-in through email
    - \* Settings page (layout and implementation)
    - \* Local Data-store (individual automatic log-in)
  - iOS
    - \* Log-in/Log-out
    - \* Settings page (layout and implementation)
    - \* Group functionality written

## 6.5 Sprint 4 Prototype

### 6.5.1 Deliverable

- Android
  - Group Messaging
    - \* Created a Layout
    - \* Used Sinch code to create a service
    - \* Implemented group messaging
    - \* Group messaging is working with no known bugs
  - Location
    - \* Page layout created and linked from GroupJoin page
    - \* MapFragment has buttons for homing and syncing group locations
    - \* Retrieving the user's location on instantiation of the MapFragment
    - \* User and group locations implemented
  - Group update service
    - \* Checks for updates in near real-time
    - \* Updates group settings when changed
    - \* Updates group members if someone leaves or joins
- Server (cloud code)
  - Functional Group update indicator
  - Returning group update information
  - Join group function (created but not functioning)
  - Leave group function (created but not functioning)
  - Check for Existing Email
- Misc/Transitional
  - Business Plan filled out, also a version tailored towards the Governor's Giant Vision contest (converted to latex)
  - Documentation done inside and outside of the source code files

### 6.5.2 Backlog

- Android
  - Begin implementing Sinch
  - Create location and messaging views and managers
  - Design models and manager classes for messaging and location
  - Broadcast/receive messages to/from all members in a group
  - Create a layout for messaging
  - Create a MapFragment to display a map
  - Create buttons over top the MapFragment to correspond to syncing and homing locations
  - 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
  - Create Group messaging unit tests
  - Create GPS Location unit tests
- Cloud Code
  - Start Group data parsing
  - Handled Leaving and joining groups
  - Validated Checking existing email upon log-in
  - Return group information upon changes
  - Create Functional Group update indicator complete
  - Implement Basic group functionality fully (log-in/log-out, join/leave groups, update on change)
- Turn in business plan to Governor's Giant Vision

### 6.5.3 Success/Fail

- Failures
  - Android
    - \* Did not create group messaging unit test
    - \* Did not create GPS location unit test
  - Cloud Code
    - \* Did not implement group function tests
    - \* Did not complete Join and Leave functions
- Success
  - Completed basic messaging
  - Completed basic GPS functionality
  - Created an update service that is the backbone of the app
  - Progress made on safe group operations
  - Passed into the Governor's Giant Vision contest finals

## 6.6 Sprint 5 Prototype

### 6.6.1 Deliverable

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

### 6.6.2 Backlog

- 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
  - 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
  - Integration Testing
  - Start Alpha Testing
- Cloud code
  - Safe group operations(leaving/joining group)
  - test join and leave functionality

### 6.6.3 Success/Fail

- Failures
  - Android
    - \* Messaging still needs an appearance update for usability
    - \* Cloud Code
      - Needs more testing on group functions
      - Completing join and leave functions
- Successes
  - Added safe group operations though cloud code
  - Reformatted the theme of the app
  - Added tool bar to tab section of groups
  - Finalist for The Governor's Giant Vision Competition
  - Fixed a few bugs in the location code

## 6.7 Sprint 6 Prototype

### 6.7.1 Deliverable

- Android
  - Group Messaging
    - \* Messages now include the names of the sender
    - \* Leader can now kick or promote members
    - \* Messages now load per group from parse
  - Broke ground for notification system
    - \* created tab system for invites and accepts
    - \* created fragments for invite and confirm
    - \* created model for notification system
    - \* items can be transferred from the invite fragment to the confirm fragment
  - Group Management Tools
    - \* Leader can now kick a member
    - \* Leader can now promote a member
    - \* fixed leader bug, now leader loads properly
  - Giant Vision
    - \* Pitch complete
    - \* Logo finalized and printed
    - \* Video backup created

### 6.7.2 Backlog

#### Week 1

- Android
  - Messaging
    - \* Messages now include the names of the sender
    - \* Leader can now kick or promote members
    - \* Messages now load per group from parse

- Group Management Tools
  - \* Leader can now kick a member
  - \* Leader can now promote a member
- Facebook and Twitter log-in
- Location
  - \* Update location automatically using service
  - \* Set Group Location

## Week 2

- Android
  - fixed leader bug, now leader loads properly
  - broke ground for notification system
    - \* created tab system for invites and accepts
    - \* created fragments for invite and confirm
    - \* created model for notification system
    - \* items can be transferred from the invite fragment to the confirm fragment
  - Option Menu's added
    - \* group join now has an option menu
    - \* settings activity moved to option menu
    - \* Group name can be changed in option menu
    - \* option menu is different if you are a group leader
    - \* option menu leads to invite system
    - \* option menu leads to blank notification page
  - Reformatted settings activity
    - \* now displays and can change display name
    - \* displays a current group if in one
    - \* added a finish button for clarity
- Cloud code
  - Safe group operations(leaving/joining group)

## Week 3

- Giant Vision Competition
  - Create the pitch
  - Create expo supporting materials
- Android
  - Notification and invite system incomplete

### 6.7.3 Success/Fail

- Android
  - Facebook and Twitter log-in incomplete
  - Notification and invite system incomplete
- Misc
  - Did not place at the Giant Vision Competition

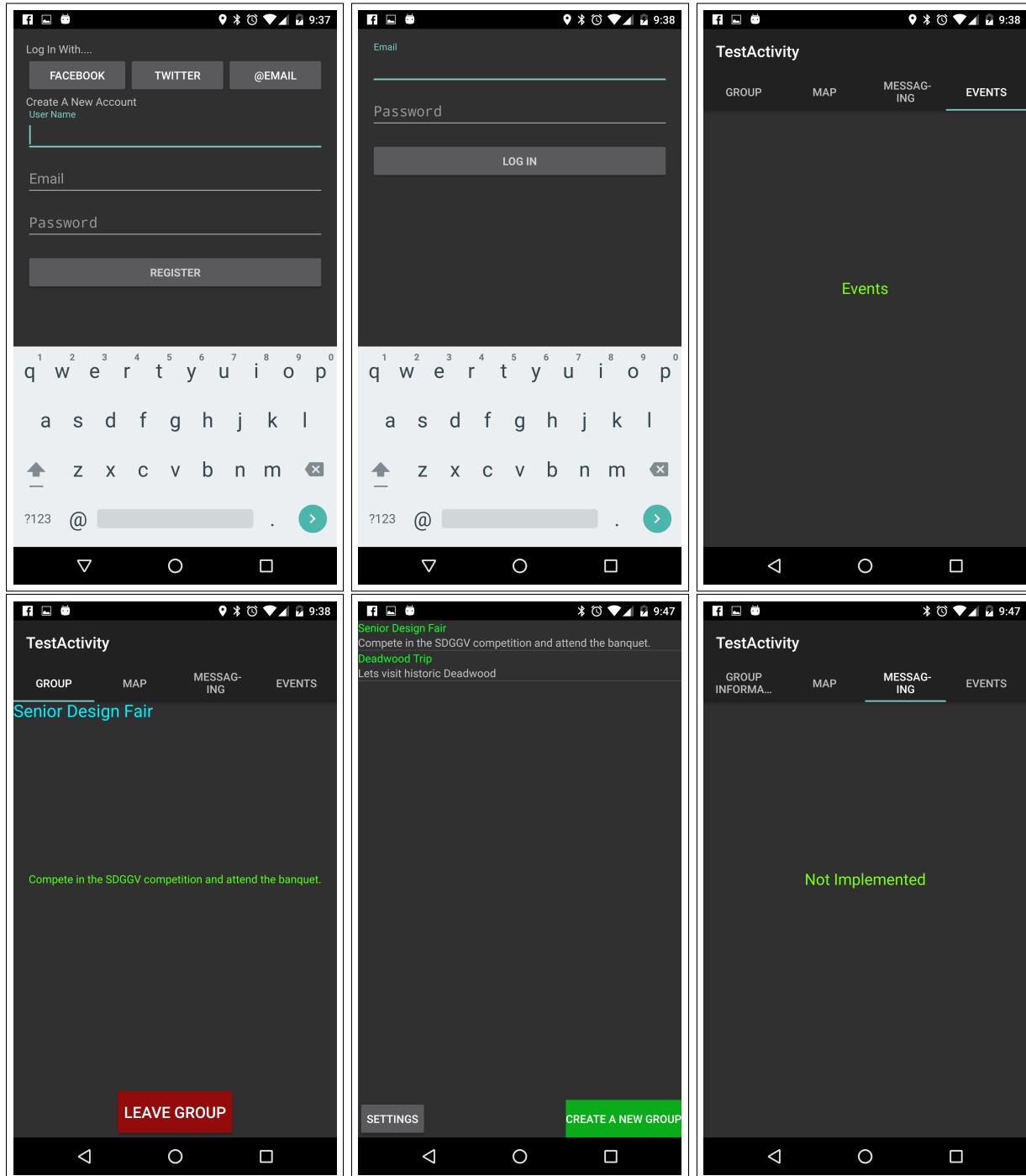


Figure 6.3: Winter Sprint Prototypes.

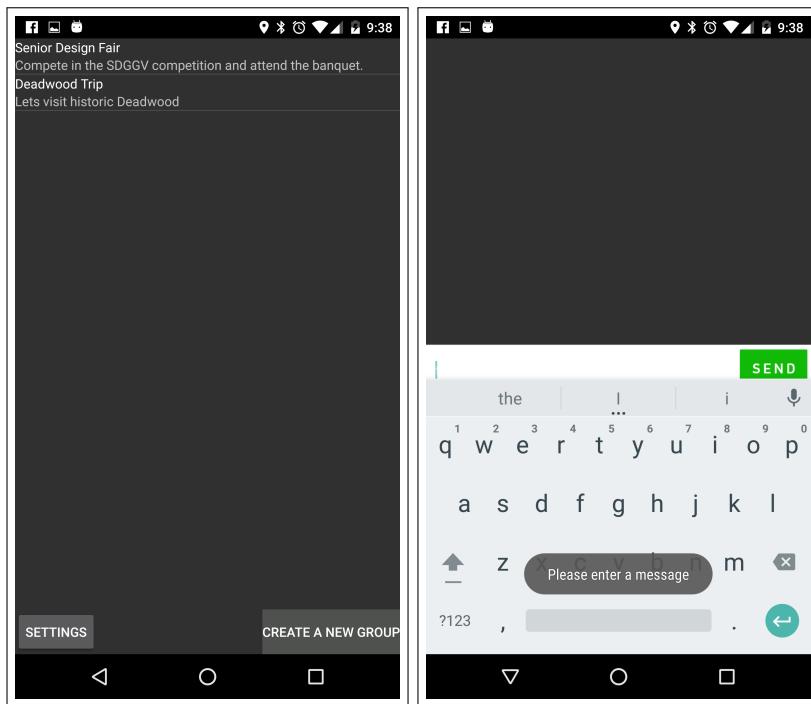


Figure 6.4: Sprint 4 Prototypes.

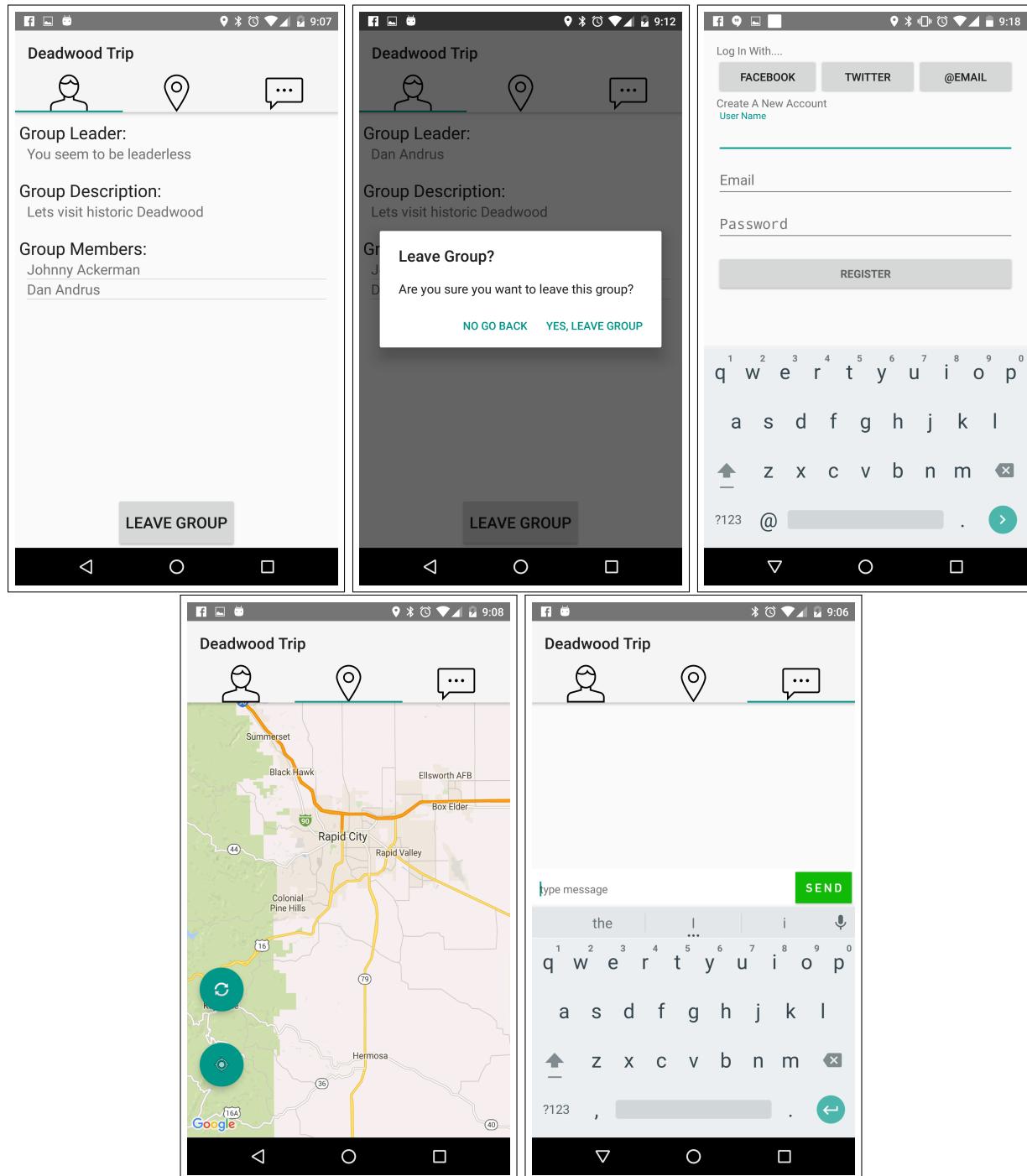


Figure 6.5: Sprint 5 Prototypes.

The image displays a 4x3 grid of screenshots from a mobile application prototype. The top row shows three screens related to the "Senior Design Fair" group:

- Screen 1:** Shows the group leader as "Johnny Ackerman" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Johnny Ackerman, Joseph Mowry, Evan Hammer, and Charles Bonn. Buttons for "Promote Joseph Mowry" and "Kick Joseph Mowry" are visible.
- Screen 2:** Shows the group leader as "Joseph Mowry" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Evan Hammer and Charles Bonn.
- Screen 3:** Shows the group leader as "Joe Mowry" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Johnny Ackerman, Dan Andrus, Evan Hammer, and Joe Mowry.

The middle row shows two screens related to the "Giant Vision Trip" group:

- Screen 4:** Shows the group leader as "Joe Mowry" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Joe Mowry and Charles.
- Screen 5:** A login screen with fields for "User Name" and "Email", and buttons for "REGISTER" and "Facebook", "Twitter", and "@Email".

The bottom row shows two screens related to the "Giant Vision Trip" group:

- Screen 6:** Shows the group leader as "Johnny Ackerman" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Johnny Ackerman, Joseph Mowry, Evan Hammer, and Charles Bonn. A map of a campus area is shown with markers for "Elli Ave", "Stelle Ave", "Montana-Dakota", "Kansas City St", "Birch Ave", "Technology Ct", "E Main St", "E St Joseph St", and "University Loop".
- Screen 7:** Shows the group leader as "Evan Hammer" and the group description as "Compete in the SDGGV competition and attend the banquet". It lists group members: Johnny Ackerman, Joseph Mowry, Evan Hammer, and Charles Bonn. A messaging interface shows messages from "Evan Hammer" to "Joe Mowry" and "Johnny Ackerman" at various times on April 13, 2016.

# 7

---

## Release – Setup – Deployment

---

After the proper licenses are attained. The both Android and iOS will be prepared for release in their respective app stores. This is much more difficult to do for the Apple Play Store due to stricter requirements on app quality. Once the apps are released, updates will also be available to download through each respective store.

### 7.1 Deployment Information and Dependencies

There are no dependencies that need to be installed after app installation. The app will be available in the app store for download.

### 7.2 Setup Information

The app will be downloadable from Google Play Store, or from the Apple Play Store, as a ready to run package.

For testing, a device may be connected to a computer running Android studio or xCode, depending on an Android or Apple device respectively. After the device is connected and the proper drivers are installed (if applicable) the app may be installed using the IDE to the device for whichever version the IDE was currently running.

### 7.3 System Versioning Information

The following is our versioning for our app during development. Due to being before release, the both iOS and Android are still in their 0 build. Each sprint during the 0 build is denoted by the second number. If there were any substantial changes that required an additional note, it would be denoted by the third number. Thus making our versions look like pre-release(0).sprint(x).substantial change(0), as an example.

Current Version [0.3.0]

*Prepared By:*

*Daniel Andrus*

*Evan Hammer*

#### iOS App Version History

Date	Version	Comments
10/1/15	0.0.0	Napkin Designs and Ideas
11/3/15	0.1.0	App Initialization, basic framework
12/11/15	0.2.0	mapping and group functionality
1/18/16	0.3.0	Model Restructure

Current Version [0.6.0]

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

#### **Android App Version History**

<b>Date</b>	<b>Version</b>	<b>Comments</b>
<b>10/1/15</b>	0.0.0	<i>Napkin Designs and Ideas</i>
<b>11/3/15</b>	0.1.0	<i>App Initialization and Basic Framework</i>
<b>12/11/15</b>	0.2.0	<i>Group Functionality</i>
<b>1/18/16</b>	0.3.0	<i>Bug Fixes and Model Reorganization</i>
<b>2/12/16</b>	0.4.0	<i>Group Messaging and Mapping</i>
<b>2/18/16</b>	0.5.0	<i>Theme Overall and Bug Fixes</i>
<b>5/1/16</b>	0.6.0	<i>Leader Controls</i>

After release, our application will finally hit build 1. From this stage, any further sprints will of course increment the version, and if any bug fixes are required to be public during a sprint, we will increment the last number previously reserved for substantial changes in a sprint.

# **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 . . . . .	55
----------------	----



# 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: [nick.bonn@bowtaps.com](mailto:nick.bonn@bowtaps.com)  
Johnathan Ackerman: [johnny.ackerman@bowtaps.com](mailto:johnny.ackerman@bowtaps.com)  
Daniel Andrus: [dan.andrus@bowtaps.com](mailto:dan.andrus@bowtaps.com)  
Evan Hammer: [evan.hammer@bowtaps.com](mailto:evan.hammer@bowtaps.com)  
Joseph Mowry: [joe.mowry@bowtaps.com](mailto:joe.mowry@bowtaps.com)

%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, implementing, 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 development 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 advisers 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 imepeded 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/Kiwi12>

## 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/Kiwi12/CSC433\\_Program1\\_Pong](https://github.com/Kiwi12/CSC433_Program1_Pong) )
- Solar System Model ( [https://github.com/Kiwi12/CSC433\\_Program3\\_SolarSystem](https://github.com/Kiwi12/CSC433_Program3_SolarSystem) )

In C++

- Simulated a B17 computer ( <https://github.com/Kiwi12/B17> )

In Lisp

- Missionary Vs Cannibals ( <https://github.com/Kiwi12/missionaryVsCannibal> )

### Solo Projects

In C++

- WVX playlist creator ( <https://github.com/Kiwi12/WVX-Playlist-Creator> )
- Basic Picture Editor ( [https://github.com/Kiwi12/Basic\\_Picture\\_Editor](https://github.com/Kiwi12/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 an 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      **Expected Graduation:** May 2016  
B.S. Computer Science; **GPA:** 2.9      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

---

**Computer Languages** C/C++, C#, ARM, SQL, HTML5, JavaScript, Java, Visual Basic, Python (3.X+)

**Protocols & APIs** JSON, XML, .NET, REST

**Databases** Microsoft SQL

**Tools/Misc.** GitHub, Mercurial(Hg), Team Foundation Server, Android Studio, Visual Studio, Xamarin, L<sup>A</sup>T<sub>E</sub>X, 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 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 held 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 Butterfield. 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.

