
Crowd Control

Senior Design Final Documentation

Bowtaps

Johnathan Ackerman

Daniel Andrus
Joseph Mowry

Charles Bonn

Evan Hammer

April 26, 2016

Contents

Title	i
Contents	vi
List of Figures	vii
List of Tables	ix
List of Algorithms	xi
Overview Statements	xiii
0.1 Mission Statement	xiii
0.2 Elevator Pitch	xiii
Document Preparation and Updates	xv
1 Overview and Concept of Operations	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
2 User Stories, Requirements, and Product Backlog	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	7
2.2.6 User Story #6	7
2.2.7 User Story #7	7

2.2.8	User Story #8	8
2.2.9	User Story #9	8
2.2.10	User Story #10	8
2.3	Requirements and Design Constraints	8
2.3.1	System Requirements	8
2.3.2	Network Requirements	8
2.3.3	Development Environment Requirements	8
2.3.4	Project Management Methodology	9
2.4	Specifications	9
2.5	Product Backlog	9
2.6	Research or Proof of Concept Results	9
2.6.1	iOS Proof of Concept Screen Shots	9
2.6.2	Android Proof of Concept Screen Shots	9
2.7	Supporting Material	9
3	Project Overview	17
3.1	Team Member's Roles	17
3.2	Project Management Approach	18
3.3	Stakeholder Information	18
3.3.1	Customer or End User (Product Owner)	18
3.3.2	Management or Instructor (Scrum Master)	18
3.3.3	Investors	18
3.3.4	Developers –Testers	18
3.4	Budget	18
3.5	Intellectual Property and Licensing	18
3.6	Sprint Overview	18
3.7	Terminology and Acronyms	19
3.8	Sprint Schedule	19
3.9	Timeline	19
3.10	Backlogs	19
3.11	Burndown Charts	19
3.12	Development Environment	19
3.13	Development IDE and Tools	19
3.14	Source Control	19
3.15	Dependencies	19
3.16	Build Environment	19
3.17	Development Machine Setup	19
4	Design and Implementation	21
4.1	Architecture and System Design	21
4.1.1	Design Selection	22
4.1.2	Data Structures and Algorithms	22
4.1.3	Data Flow	22
4.1.4	Communications	22
4.1.5	Classes	22
4.1.6	UML	22
4.1.7	GUI	22
4.1.8	MVVM, etc	22
4.2	Major Component #1	22
4.2.1	Technologies Used	22
4.2.2	Component Overview	22
4.2.3	Phase Overview	22
4.2.4	Architecture Diagram	22
4.2.5	Data Flow Diagram	22
4.2.6	Design Details	22

4.3	Major Component #2	23
4.3.1	Technologies Used	23
4.3.2	Component Overview	23
4.3.3	Phase Overview	23
4.3.4	Architecture Diagram	23
4.3.5	Data Flow Diagram	23
4.3.6	Design Details	23
4.4	Major Component #3	23
4.4.1	Technologies Used	23
4.4.2	Component Overview	23
4.4.3	Phase Overview	24
4.4.4	Architecture Diagram	24
4.4.5	Data Flow Diagram	24
4.4.6	Design Details	24
5	System and Unit Testing	25
5.1	Overview	25
5.2	Dependencies	26
5.2.1	Android Client Application	26
5.2.2	iOS Client Application	26
5.2.3	Parse Service and Cloud Code	26
5.2.4	Sinch Service	27
5.3	Test Setup and Execution	28
5.4	System Testing	28
5.5	System Integration Analysis	30
5.6	Risk Analysis	30
5.6.1	Team Risks and Mitigation	30
5.6.2	Technology Risks and Mitigation	30
5.6.3	Market Risks and Mitigation	31
5.7	Successes, Issues and Problems	31
5.7.1	Successes	31
5.7.2	Issues	31
5.7.3	Problems	32
5.7.4	Changes to the Backlog	32
6	Prototypes	33
6.1	Sprint 1 Prototype	33
6.1.1	Deliverable	33
6.1.2	Backlog	33
6.1.3	Success/Fail	33
6.2	Sprint 2 Prototype	33
6.2.1	Deliverable	33
6.2.2	Backlog	33
6.2.3	Success/Fail	33
6.3	Sprint 3 Prototype	33
6.3.1	Deliverable	33
6.3.2	Backlog	33
6.3.3	Success/Fail	33
6.4	Sprint 4 Prototype	33
6.4.1	Deliverable	33
6.4.2	Backlog	33
6.4.3	Success/Fail	33
6.5	Sprint 5 Prototype	33
6.5.1	Deliverable	33
6.5.2	Backlog	33

6.5.3	Success/Fail	34
7	Release – Setup – Deployment	35
7.1	Deployment Information and Dependencies	35
7.2	Setup Information	35
7.3	System Versioning Information	35
8	User Documentation	37
8.1	User Guide	37
8.2	Installation Guide	37
8.3	Programmer Manual	37
9	Class Index	39
9.1	Class List	39
10	Class Documentation	41
10.1	Poly Class Reference	41
10.1.1	Constructor & Destructor Documentation	41
10.1.2	Member Function Documentation	41
11	Business Plan	43
	Bibliography	45
	Software Agreement	SA-1
A	Product Description	A-1
1	GPS Features	A-1
1.1	Group Members	A-1
1.2	Suggestions	A-1
2	Group Messaging	A-1
3	Group Manangement Features	A-1
4	Parse Features	A-1
B	Sprint Reports	B-1
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
C	Industrial Experience and Resumes	C-1
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
D	Acknowledgment	D-1
E	Supporting Materials	E-1

List of Figures

1.1	Basic System Flow Diagram	3
2.1	iOS login select screen	10
2.2	iOS email login screen	10
2.3	iOS create account screen	11
2.4	iOS group infomation screen	11
2.5	iOS map view screen	12
2.6	iOS messaging main screen	12
2.7	Android login screen	13
2.8	Android create group screen	13
2.9	Android group information screen	14
2.10	Android group join screen	14
2.11	Android messaging main screen	15

List of Tables

List of Algorithms

1	Calculate $y = x^n$	21
---	-------------------------------	----

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

Overview and Concept of Operations

1.1 Bowtaps and Its Team

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

1.2 Crowd Control

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

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

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

1.2.1 Purpose of the System

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

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

1.3 Business Need

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

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

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

1.4 Deliverables

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

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

1.5 System Description

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

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

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

1.5.1 Integrated Group Messaging

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

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

1.5.2 GPS Location Services

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

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

1.5.3 Group Management Features

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

1.5.4 Suggestions

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

1.6 System Overview and Diagram

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

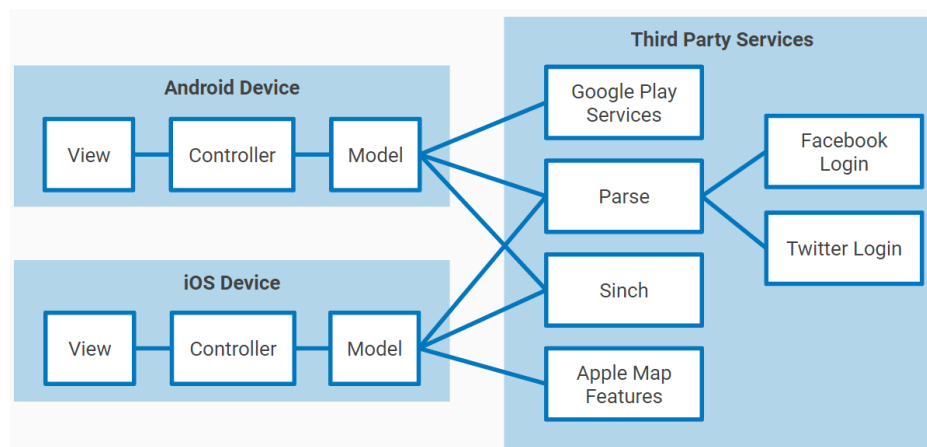


Figure 1.1: Basic System Flow Diagram

1.7 Technologies Overview

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

1.7.1 Google Play Services

1.7.1.a Description

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

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

1.7.1.b Usage

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

1.7.2 Apple Map Features

1.7.2.a Description

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

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

1.7.2.b Usage

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

1.7.3 Parse

1.7.3.a Description

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

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

1.7.3.b Usage

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

1.7.4 Sinch

1.7.4.a Description

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

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

1.7.4.b Usage

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

2

User Stories, Requirements, and Product Backlog

2.1 Overview

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

2.2 User Stories

2.2.1 User Story #1

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

2.2.1.a User Story #1 Breakdown

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

2.2.2 User Story #2

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

2.2.2.a User Story #2 Breakdown

2.2.3 User Story #3

As a user i want post agenda for the group.

2.2.4 User Story #4

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

2.2.5 User Story #5

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

2.2.6 User Story #6

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

2.2.7 User Story #7

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

2.2.8 User Story #8

As a user i want the abilitiy to login.

2.2.9 User Story #9

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

2.2.10 User Story #10

As a user i would like my information protected.

2.3 Requirements and Design Constraints

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

2.3.1 System Requirements

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

2.3.1.a iOS Requirements

- Use Apple Mapping Features
- Access Parse as the Database

2.3.1.b Android Requirements

- Use Google Maps
- Access Parse as the Database

2.3.1.c Parse Requirements

- Delete groups when group is not in use

2.3.2 Network Requirements

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

2.3.3 Development Environment Requirements

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

2.3.4 Project Management Methodology

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

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

2.4 Specifications

2.5 Product Backlog

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

2.6 Research or Proof of Concept Results

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

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

2.6.1 iOS Proof of Concept Screen Shots

Below are screen shots from the iOS version of CrowdControl.

2.6.2 Android Proof of Concept Screen Shots

Below are screen shots from the Android version of CrowdControl.

2.7 Supporting Material

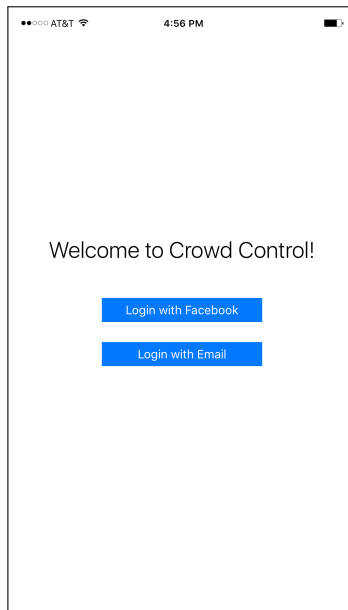


Figure 2.1: iOS login select screen

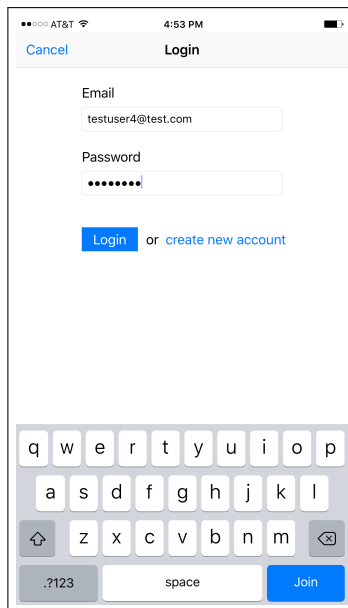
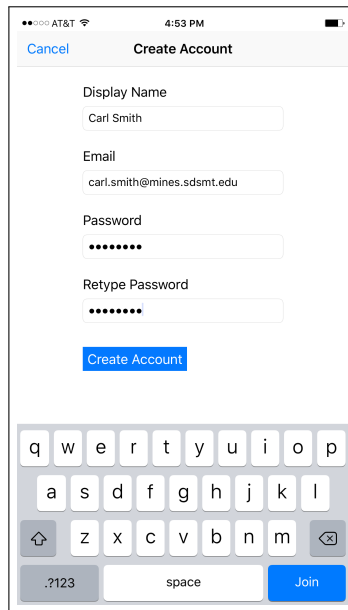
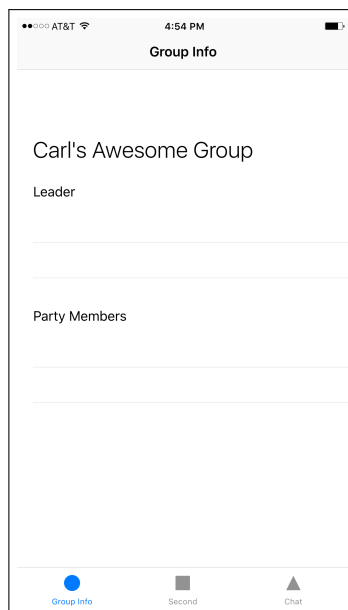


Figure 2.2: iOS email login screen



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

Figure 2.3: iOS create account screen



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

Figure 2.4: iOS group information screen

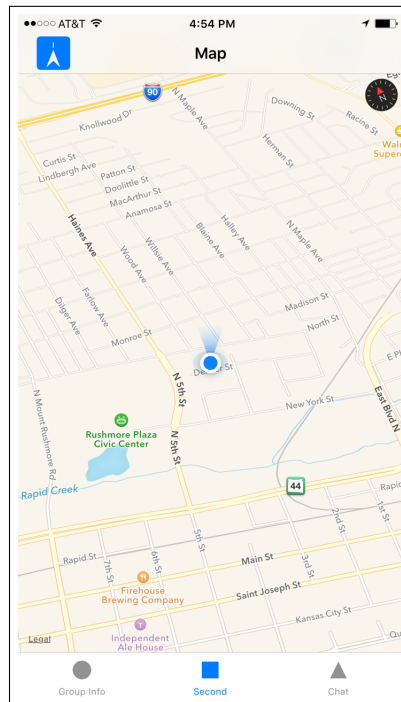


Figure 2.5: iOS map view screen

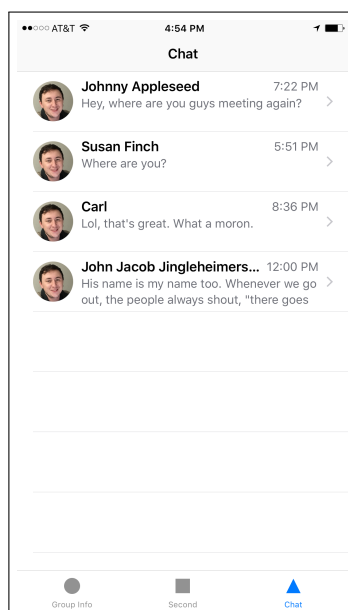


Figure 2.6: iOS messaging main screen

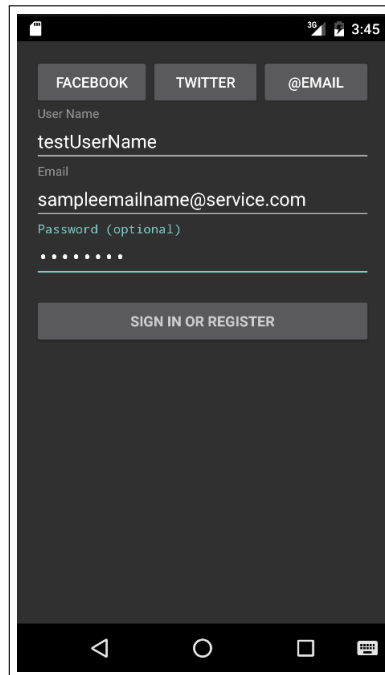


Figure 2.7: Android login screen

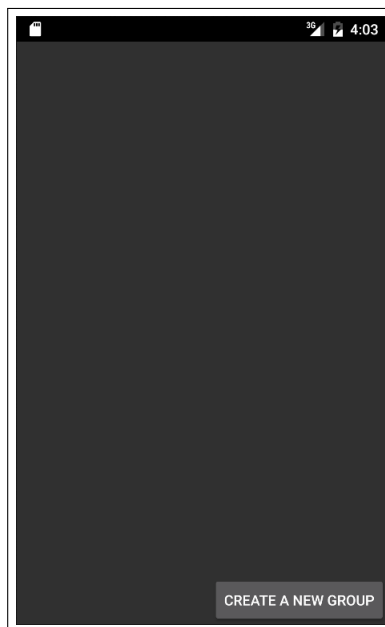


Figure 2.8: Android create group screen

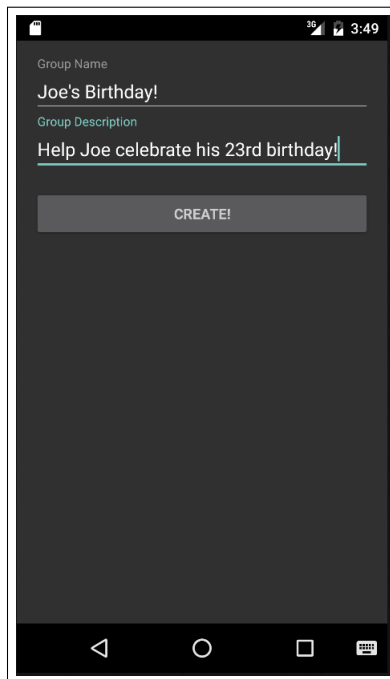


Figure 2.9: Android group information screen

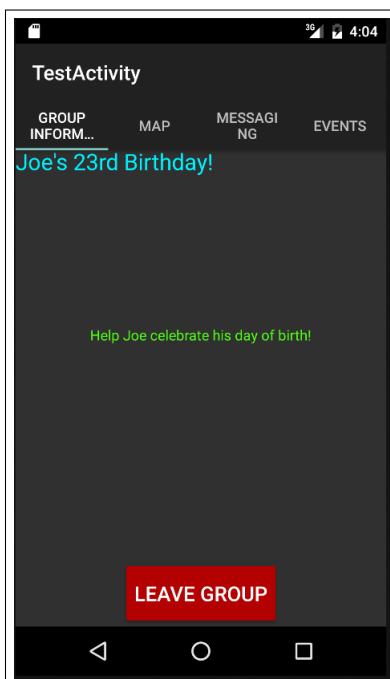


Figure 2.10: Android group join screen

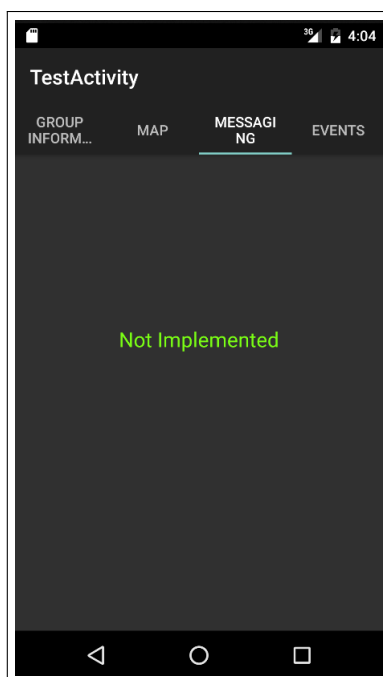


Figure 2.11: Android messaging main screen

3

Project Overview

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

3.1 Team Member's Roles

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

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

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

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

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

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

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

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

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

2. Creating links to Apple maps to the mobile application

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

1. Creating links from the database to the mobile application

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

2. Creating links to Apple maps to the mobile application

3.2 Project Management Approach

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

3.3 Stakeholder Information

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

3.3.1 Customer or End User (Product Owner)

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

3.3.2 Management or Instructor (Scrum Master)

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

3.3.3 Investors

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

3.3.4 Developers –Testers

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

3.4 Budget

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

3.5 Intellectual Property and Licensing

Describe the IP ownership and issues surrounding IP.

3.6 Sprint Overview

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

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

3.7 Terminology and Acronyms

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

3.8 Sprint Schedule

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

3.9 Timeline

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

3.10 Backlogs

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

3.11 Burndown Charts

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

3.12 Development Environment

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

3.13 Development IDE and Tools

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

3.14 Source Control

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

3.15 Dependencies

Describe all dependencies associated with developing the system.

3.16 Build Environment

How are the packages built? Are there build scripts?

3.17 Development Machine Setup

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

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  $\{N \text{ is odd}\}$ 
     $y \leftarrow y \times X$ 
     $N \leftarrow N - 1$ 
  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

Crowd Control utilizes multiple services across different platforms, requiring all parts to be operating correctly in order to provide consistent service to all users. We use system and unit testing to assist developers in identifying potential issues and to verify that the product being produced meets requirements. The goal of system and unit testing is to automate the process to increase development efficiency and reduce testing errors.

Given the time constraints and scope of Crowd Control, we have been forced to make compromises in regards to system and unit testing. Implementing and running automated unit testing, can be as time-consuming as writing production code. In fact, it is extremely common for a single function of production code to require many times as many lines of unit testing code to completely test the function. While automated testing provides many invaluable benefits, it would ultimately be a hindrance to the project if it means missing critical deadlines and not achieving crucial project milestones.

In this section, we describe our approach to testing, our goals, and the test cases that we have developed and implemented thus far. We also discuss the future plans for testing and how we will be able to devote more time and energy towards this project in the upcoming months.

5.1 Overview

For this project, we use a combination of “black-box” and “white-box” testing. This means that there are some aspects of the project that we have created ourselves and have full control over its execution, and there are others that we have little control over because it is provided by a third party. In practical terms, it means that we can test our own code with the full knowledge of how it is intended to work, allowing us to design tests that target potential problem areas. When designing tests that make use of a third-party service or a library that we have not written ourselves, we can only write tests that verify that the supplied functionality works as documented.

Because one of the goals of this project is to develop the applications using native languages and technology, we make use of the most established testing frameworks for the platform we are testing on. For the iOS application, we use the testing tools and frameworks built directly into Xcode. For the Android application, we use Espresso for testing our interface and JUnit for testing our classes and methods. To test Parse, the third party service we use for data storage and retrieval, we use unit testing on both Android and iOS, as interacting with Parse requires a running application on a device. To test Sinch, the third party service we use for instant messaging between users, we use the unit testing frameworks we use for testing our Android and iOS applications since interacting with Sinch also requires a running instance of the application on a device.

Running unit tests in all of the frameworks above yields a list of binary results; for each test executed, we receive either a “pass” or “fail” result. If a test fails, we may also receive a message describing the issue that caused the test to fail. There exist several conditions that could cause a test to fail.

- An uncaught exception is thrown
- A runtime error is encountered
- The application crashes
- A test assertion fails

When designing test cases, we begin with the set of user stories identified at the beginning of the project. A list of user stories can be found in the section 2.2. From these user stories, we build a testing matrix that will be referenced when writing our automated tests. Tests are then written as the code they are intended to test is written.

Tests are also to be run relatively frequently. Ideally, developers would enable the set of tests relevant for the aspect of the project they are currently working on and run those tests as needed during development and again before the code is merged in with the rest of the project. Then, the full set of tests would be run automatically and at a time that would not interfere with development or execution of the product.

5.2 Dependencies

This project can be broken into several pieces that all work together to form a single cohesive product. Taken individually, these pieces would be of little utility. It is important that we test each how well each component interacts with each other as well as each isolated part as best we can. The four main components of this project are as follows:

- Android client application
- iOS client application
- Parse service and cloud code
- Sinch service

Each of these aspects of the project need to be somehow tested, although some parts can be more accurately tested than others. More thorough requirement breakdowns of each aspect are described in the subsections below.

5.2.1 Android Client Application

To test our client code running locally on Android devices, we use two testing frameworks: JUnit for testing Android-independent code and Espresso for testing navigation and user interface requirements. These frameworks are run from within our integrated development environment, Android Studio. Additionally, for interface testing, we require a running emulator or a connected Android device on which Android Studio can run the unit tests.

While developing unit tests for the JUnit and Espresso frameworks, we use Google's Android testing Support Library Documentation website. This resource provides accurate information on installing and using these frameworks in conjunction with Android Studio. Writing unit tests and best practices, as well as class documentation are available from this website.

5.2.2 iOS Client Application

To test our client code running locally on iOS devices, we use the testing tools included in Xcode, our iOS integrated development environment. Backend unit testing and interface testing are all handled by the same framework. These tests are run from within Xcode. Interface tests require a running iOS emulator or an iOS device connected to the computer. Because the unit testing tools used to test iOS devices are tied to Xcode, this means that the tests can only be run on a Macintosh computer.

While developing unit tests for iOS devices, we reference iOS Developer Library website. This resource contains instructions on designing and setting up unit tests, as well as instructions on running unit tests. Class documentation can also be found on this website.

5.2.3 Parse Service and Cloud Code

Some aspects of our project exist as "cloud code", which is custom server-side logic executed by our database backend service, Parse. This cloud code is written in JavaScript and handles database operations that require security, data integrity, consistency, and efficiency. However, because this cloud code runs on a remote server to which we have limited access and concealed knowledge of its internal workings, we are forced to test our code by the only means we have; using the client applications.

Parse does not supply testing features, and thus we must devise a way to execute our own unit tests using the tools we have. To test the functionality of our cloud code and our Parse interaction code, we require these things:

1. The Parse libraries installed
2. Parse interactivity built into the application
3. An active Internet connection to the Parse service
4. An emulator/simulator/developer device
5. The testing framework associated with the client platform being used for testing
6. A Parse application key
7. A separate Parse database dedicated to testing

In order to run tests on our cloud code, we use both versions of our client (iOS and Android) to connect to the Parse server, perform database operations by calling cloud code, then verifying the results. We use the aforementioned testing frameworks to develop and execute these tests on both platforms. These tests can be performed on device simulators, device emulators, and physical developer devices alike.

When using client software to test server software, we must properly configure the clients to interact with the database as if it were no different than a production database. Thus, we require that the Parse libraries be included in the project and that actual Parse functionality built into the application. As with all Parse-enabled applications, this requires a unique application key be assigned by Parse and used to access the database.

Because running tests on Parse result in changes to the data in the database we are testing with, it is crucial that the automated tests be run on an independent Parse database dedicated to testing. Running such tests on either the production or development databases could result in irreversible destruction of data that we cannot afford to jeopardize. In order to avoid this, all unit tests that involve testing Parse functionality should be executed using a database where the data is used exclusively for testing.

5.2.4 Sinch Service

Another crucial third party service we utilize is Sinch, an instant messaging platform that we use in conjunction with Parse. This service does not have cloud code support, but that is okay since we have no need for cloud code here. However, it is important that we test how well our client applications connect to and use the service to ensure that communications run smoothly.

In order to test the functionality of Sinch and how well our applications interact with it, we require these things:

1. The Sinch libraries installed
2. Sinch functionality built into the application
3. An active internet connection to Sinch
4. An emulator/simulator/developer device running the application
5. The testing framework associated with the client platform being used for testing
6. A Sinch application key
7. A separate Sinch application dedicated to testing

As with Parse, Sinch is a third party service to which we have limited access and no knowledge of the implementation details. With the addition of the fact that we have no cloud code for Sinch to test, the only aspect of this service we need to test is how our client applications interact with the service.

To test our integration with Sinch, we run tests using our Android and iOS client applications on either device emulators, device simulators, or physical development devices. To develop and run these tests, we use the aforementioned testing frameworks used for testing the client code itself on each platform. We then run our

tests on these devices, which cause the application to connect to and communicate with Sinch using the provided developer API. Sinch requires an active Internet connection, so the success of the tests depends on the testing devices having a stable Internet connection.

Lastly, like Parse, Sinch supplies a unique application key which our clients use to authenticate with the Sinch servers. All versions of our client applications use this key and is what allows them to communicate with each other. Also like Parse, we require that all tests run through Sinch also be run using a special Sinch application key that is completely separate from the production and development versions of the application. We must do this in order to avoid user ID conflicts and to limit any operations that may affect service uptime and reliability.

5.3 Test Setup and Execution

As with most aspects of software development, test setup and execution requires two distinct phases: design and execution. As functionality and features are designed and built, correlating tests are designed concurrently and added to our testing matrix. After a test has been adequately designed with inputs and expected outputs defined, the unit tests can be implemented in whatever testing framework associated with the platform which the tests are targetting.

Setting up unit tests involves referencing the previously designed test parameters while writing code that will run on the target platform within the selected testing framework. For example, implementing a unit test that will run on the Android platform, we create a new unit test class for the portion of the code we aim to test, following framework usage. The specifics on implementing a unit test within the constraints of a framework often differ greatly between frameworks.

Essentially, we create a new testing class that extends some framework-supplied class. We then create functions that will perform the tests under different conditions, called test cases. Usually, if our goal is to test the functionality of a single class, we will create a single test class with the sole purpose of testing said class.

A full breakdown of our most current testing matrix can be found in the section 5.4.

5.4 System Testing

Most of the unit tests our team has designed are related to the functionality of the system as a whole, focusing on user experience and how we expect the application to respond to inputs at each given state. Rather than focus on testing a specific aspect of the project, this testing matrix includes test cases and expected results that often require the cooperation of every part of the project.

Requirements	Test Case	Expected
Log in with email	Valid credentials Invalid email Invalid password	Success Failure Failure
Log in with Facebook	Valid credentials Invalid credentials	Success Failure
Log in with Twitter	Valid credentials Invalid credentials	Success Failure
Sign up with email	Email already in use Valid credentials Invalid password Unconfirmed password Invalid email Whitespace-only username Empty username Empty email Empty password	Message Success Failure Failure Failure + Message Failure + Message Failure + Message Failure + Message Failure + Message
Create a group	No name Whitespace-only name No description Whitespace-only description Else	Message Failure + Message Success + Message Success + Message Success + trim whitespace + collapse whitespace in name
Search for users	Enter display name Enter email Enter phone number Enter incomplete phone number or incomplete email	Display list of matching users or "no matches" Detect email, no change Detect phone number, no change Display "no matches"
Send invite to users to join group	By email unassociated with user By email associated with user By phone number unassociated with user By phone number associated with user	Generate and send email Send app notification to user, display invite in-app Detect phone number, no change Send app notification to user, display invite in-app
Join a public group	Tap "join group" Accept confirmation message Decline confirmation message	Displays confirmation message (y/n) Message "request sent" Dismiss message
Join group via invite	Tap "join" Tap "delete"	Join group Delete invitation
Leave a group	Tap "Leave Group" Accept confirmation message Leader accepts confirmation Leader declines confirmation Decline confirmation message	Display confirmation message (y/n) If leader, display confirmation to choose new leader (y/n), else leave the current group and return to nearby groups Display list of group members Leave current group and return to nearby group list Dismiss message, do not leave group
Group leader disbands group	Leader taps "disband group" Accept confirmation Decline confirmation	Display confirmation message (y/n) Display message to all users and return to join group page Dismiss message, do not leave group
Enter display name	Empty string	Highlight, no accept

5.5 System Integration Analysis

System integration testing is a process by which we verify that a newly introduced piece of code or feature properly integrates with the existing code and that few errors or bugs arise. This usually involves performing black box testing on a very specific and well-defined subsystem. In order to accurately perform these tests such that the subject subsystem is in no way affected by any other aspect of the project, the unit test will often supply mock data and mock functionality that behaves according to specifications.

Due to the time constraints of this project, a majority of our testing design efforts were put towards developing the overall system testing matrix, which tests the entire system rather than just a specific subset. This allowed us to devote more time towards development, documentation, and business endeavors. It is unfortunately a side effect that the subsystems we have built will need to be revisited and have unit tests applied to them in order for us to verify that each component is functioning as expected.

5.6 Risk Analysis

When first starting this project, we quickly became aware of a number of risks that could hinder our progress. Many of these risks were related to our team and whether or not we would be capable of producing a market-ready product, while a fair number of these risks were about the technology we were utilizing and whether or not we could depend on it. The third area of risk was the in the business as a whole, which is an area in which none of the team was particularly knowledgeable in.

5.6.1 Team Risks and Mitigation

When design and development of this application was first underway, the biggest risk was that the team would not have the skills or training necessary to successfully produce the desired product. Each team member was an undergraduate student studying computer science with little experience in mobile application design and development. Few of us had any graphic design experience, and none had worked in a professional capacity on apps.

Additionally, being college students with at most part-time jobs, the availability of each team member was less than ideal. Each member was required to divide their attention between multiple concurrent projects and assignments as well as any jobs or internships they have.

To mitigate the lack of experience and education in mobile app development, graphic design, and user experience design, each team member spent a large amount of time researching and studying in detail the behavior and usage of each technology that they would be working with. Frequently, the team would meet to share research with each other and educate one another. On multiple occasions, team members would prepare impromptu presentations to communicate the results of their efforts with the other team members.

Mitigating the time management risk was a more challenging task. To do so, each team member was forced to cut back on work hours in order to devote more time to this project in order to keep up with requirements and deadlines. Weekends were often dedicated to working on this project, with many members spending 4-8 hours a day developing the core product. Frequent meetings with the rest of the team and strict time schedules helped keep each member on task and the team as a whole moving forward at a swift and consistent rate.

5.6.2 Technology Risks and Mitigation

The core product of this project depends heavily on an active internet connection and a consistent feed of data between third party services in order to function properly. This means that a major risk that lies beyond our control is the availability of internet service for our end users as well as service uptime. Because these services are provided by third parties, our team has little control over their availability, pricing, and features.

Mitigating this risk is a difficult one, as it involves depending on technology that we have no guarantee on how it will or will not change. The most we can do is ensure that our dependencies on these services is not so much that the project will utterly collapse if one of them becomes unavailable or otherwise unusable. The best way to do this is to minimize the amount of coupling between our software and the libraries provided by third parties and increase separability between the two.

This risk is of particular relevance because it was one of the first risks identified early on and as a result was addressed before any amount of code was written. In the end, this proved to be an extremely wise decision, as

one of our third party services, Parse, which is responsible for data storage, retrieval, account management, and cloud code, will be terminating its services on January, 2017. When this announcement was made, we realized that our initial efforts towards separability and separability were not in vain and will become useful shortly. This now introduces a new risk for us to address: finding a backup database server service.

Fortunately, with the announcement of Parse's closure, the company that operates the service also announced that the software used to power the service will be made open source so that we or any other company interested in continuing Parse service on their applications without needing to rewrite thousands of lines of code can do so by hosting their own servers. To ensure that we can continue development of this product without being severely affected by the closure of Parse, we plan on utilizing another third-party generic server host that will be capable of handling the capacity of our application using the server software provided by Parse.

5.6.3 Business Risks and Mitigation

The ultimate goal of this project is to develop a mobile application that can be released to market. Because of this, one huge aspect of this project was not just to produce a piece of software that functions well, but also to position the team and company to prepare for launch and to generate revenue from the product.

Rather than point out one particular area of the business side of things, we realized that as a whole we were inexperienced and largely ignorant on what went into building a successful business. This resulted in virtually every business decision we made into a potential risk; without fully understanding the consequences of each decision, it was entirely possible for us to work ourselves into a corner.

To mitigate this risk, we began reaching out to and taking advice from the entrepreneurs in residence at our school. These entrepreneurs who have had past and present business experience were kind enough to share their knowledge with our team. Decisions such as target market, how to promote our app, and alternative sources of revenue are all business aspects that we discussed at length with these entrepreneurs.

5.7 Successes, Issues and Problems

During the course of our project's development, we have encountered various successes, issues, and problems in regards to system and unit testing. Details on each of these are listed below, along with a summary on the changes made to our testing backlog throughout the course of this project.

5.7.1 Successes

The greatest success of our work in unit and system testing has been in the development of our testing matrix. Our testing matrix, while not comprehensive (as such a matrix requires years of refining to produce), will act as a set of blueprints when setting up automated tests. At over one hundred test cases strong, this matrix defines the behavior of our application in many different possible states. Using this information, we can implement automated unit tests more efficiently than if we did not have such a matrix.

A secondary success in our testing efforts was in discovering that all of the necessary testing frameworks are free and are mature enough to have exhaustive documentation and support. We have access to a plethora of information that has aided us in designing and implementing unit and system tests using these native tools. Additionally, the tools we are using integrate seamlessly into the IDEs we are using for development, reducing the number of tools required to use them.

Finally, we have succeeded in designing and implementing a number of unit tests for both iOS and Android versions of our client applications. As implementing unit tests takes a significant time investment, the ones we have built provide a good foundation upon which to build. On iOS for example, many interface tests related to the login and signup screens have been implemented and are used for testing account creation and login. On Android, we have implemented a similar set of unit tests as we did for iOS.

5.7.2 Issues

One major issue we encountered when designing unit tests was the fact that none of the team members have had significant experience doing so. Although a number of us have had a moderate amount of formal education on the subject through course and guest lectures, none have designed or implemented automated testing in a project

of this magnitude. Thus, an enormous part of testing for this project was devoted to studying testing methods used in other projects and researching how to use our tools to build these tests.

The second issue encountered was time constraints. Our goals to turn this project into a viable product fit for the market required that we focus heavily on progress towards a working prototype that can be demonstrated by the end of the semester. Automated testing, although an important development tool, requires a large time investment overhead that we could not afford if we desired to meet requirement deadlines and achieve our critical milestones.

When making the conscious decision to prioritize our efforts, we agreed that code separability and maintainability should not be neglected. These two tenants of software development are heavily promoted by automated testing. By choosing to prioritize separability and maintainability when designing our software, we are confident that delaying automated testing will be less of a hurdle when the time comes to implement them. Regardless, we are aware that by not investing the time now, we are accruing “technical debt”, which means that this task will be more difficult to accomplish the longer we delay.

5.7.3 Problems

The unit and system testing section of this project was not without its own set of problems. Many of these we did not foresee, as they are problems that have arisen from our tools or our testing devices, both of which we have limited control over.

Xcode, the IDE chosen for iOS development underwent an update that appeared to cause UI testing on iOS to become undependable. Often unit tests run using this tool would immediately report failures without apparently running the test properly. It is possible that this is a configuration error, but it is one that we have yet to solve if it is. Other Xcode users have reported similar problems using these features.

Another problem encountered was in using our hardware devices to run automated testing. For the most part, the device simulators and emulators are adequate for running most tests. In some cases, the devices would not cooperate with the testing frameworks. These devices would either run too slowly for the tests and sometimes reject the incoming connection to the IDE. In the end, we decided to forego solving these technical difficulties to focus on product development and business-related issues.

5.7.4 Changes to the Backlog

6

Prototypes

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

6.1 Sprint 1 Prototype

6.1.1 Deliverable

6.1.2 Backlog

6.1.3 Success/Fail

6.2 Sprint 2 Prototype

6.2.1 Deliverable

6.2.2 Backlog

6.2.3 Success/Fail

6.3 Sprint 3 Prototype

6.3.1 Deliverable

6.3.2 Backlog

6.3.3 Success/Fail

6.4 Sprint 4 Prototype

6.4.1 Deliverable

6.4.2 Backlog

6.4.3 Success/Fail

6.5 Sprint 5 Prototype

6.5.1 Deliverable

6.5.2 Backlog

6.5.3 Success/Fail

7

Release – Setup – Deployment

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

7.1 Deployment Information and Dependencies

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

7.2 Setup Information

How is a setup/install built?

7.3 System Versioning Information

How is the system versioned?

8

User Documentation

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

8.1 User Guide

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

8.2 Installation Guide

8.3 Programmer Manual

9

Class Index

9.1 Class List

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

Poly	41
----------------	----

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
Johnathan Ackerman:	johnny.ackerman@bowtaps.com
Daniel Andrus:	dan.andrus@bowtaps.com
▣Evan Hammer:	evan.hammer@bowtaps.com
Joseph Mowry:	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, implimenting, and distributing CrowdControl under the SDSMT Senior Design program.

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

2 EFFECTIVE DATE

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

3 DEFINITIONS

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

4 DEVELOPMENT OF SOFTWARE

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

5 COMPENSATION

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

6 CONSULTATION AND REPORTS

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

7 CONFIDENTIAL INFORMATION

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

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

8 INTELLECTUAL PROPERTY RIGHTS

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

9 WARRANTIES

The BowTaps Team represents and warrants to Sponsor that:

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

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

10 INDEMNITY

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

11 INDEPENDENT CONTRACTOR

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

12 TERM AND TERMINATION

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

13 GENERAL

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

14 SIGNATURES



10 / 6 / 2015

Charles Bonn

Date



10 / 6 / 2015

Evan Hammer

Date



10 / 6 / 2015

Joseph Mowry

Date



10 / 6 / 2015

Daniel Andrus

Date



10 / 6 / 2015

Johnathan Ackerman

Date



10 / 6 / 2015

Brian Butterfeild

Date

A

Product Description

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

1 GPS Features

1.1 Group Members

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

1.2 Suggestions

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

2 Group Messaging

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

3 Group Manangement Features

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

4 Parse Features

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

B

Sprint Reports

1 Sprint Report #1

Sprint Report #1

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Customer Overview

Customer Description

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

Customer Problem

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

Customer

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

Project Overview

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

Phase 1

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

Project Environment

Project Boundaries

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

Project Context

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

Deliverables

Phase 1

Deliverables will be UX design, database design and implementation.

Backlog

Phase 1

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

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

Sprint Report

Work for this sprint included:

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

2 Sprint Report #2

Sprint Report #2

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Work Summary

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

Backlog

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

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

Successes

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

Issues and Changes

Some issues that have been ran into have been

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

Team Details

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

3 Sprint Report #3

Sprint Report #3

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Work Summary

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

Backlog

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

Success

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

Issues and Changes

Some issues that have been ran into have been

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

Team Details

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

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

4 Sprint Report Winter Sprint

Winter Sprint Report

Team Overview

Name

Crowd Control

Members

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

Project Title

CrowdControl - Group Management Mobile Application

Company

Bowtaps

Deliverables

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

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

Remaining Backlog

Here are the incomplete items/features for this sprint:

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

Successes

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

Issues and Changes

Some issues that we encountered include:

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

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

Team Details

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

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

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

5 Sprint Report #4

Sprint Report #4

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control - Group Management Mobile Application

Company

Bowtaps

Backlog

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

Week 1

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

Week 2

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

Week 3

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

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

Deliverables

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

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

5. Check for Existing Email

- Misc/Transitional

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

Issues and Changes

Some issues that we encountered include:

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

Remaining Backlog

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

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

Team Details

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

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

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

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

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

6 Sprint Report #5

Sprint Report #5

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control - Group Management Mobile Application

Company

Bowtaps

Backlog

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

Week 1

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

Week 2

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

Week 3

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

Deliverables

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

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

Issues and Changes

Some issues that we encountered include:

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

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

Remaining Backlog

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

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

Team Details

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

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

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

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

C

Industrial Experience and Resumes

1 Resumes

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

Johnathan Ackerman

605-877-1757

Johnathan.ackerman@mines.sdsmt.edu

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

Education

South Dakota School of Mines and Technology

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

Central High School

- Graduated 2012

Programs

Team Projects

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

- Pong (https://github.com/Kiwii12/CSC433_Program1_Pong)
- Solar System Model (https://github.com/Kiwii12/CSC433_Program3_SolarSystem)

In C++

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

In Lisp

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

Solo Projects

In C++

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

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: danieleandrus@gmail.com
Twitter Handle: @deaboy100
Github Name: Deaboy

PROFILE

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

EXPERIENCE

INTERN DEVELOPER, 7400 CIRCUITS — SUMMER 2015 - PRESENT

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

SDSMT PROGRAMMING TEAM — 2014 - PRESENT

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

SERVER ADMINISTRATOR, PROGRAMMER — 2010 - PRESENT

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

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

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

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

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

SKILLS

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

EDUCATION

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

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

PERSONAL INFORMATION

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

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

C. Nicholas Bonn

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

Education:

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

Bachelor of Science in Computer Science

Anticipated Graduation: May 2016

Cumulative GPA: 2.5

Relevant Coursework:

Database

Software Engineering

Cyber Security

Graphic User Interface

Projects:

Crowd Control App – on-going senior design project

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

Technical Skills:

Languages:

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

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

Other Technical Services:

Databases: SQL Server, MySQL

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

Work Experience:

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

September 2009 – Current

Program Assistant

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

TMI Coatings Inc., Eagan, MN

May 2012 – August 2012

Summer Intern

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

Awards:

Butterfield Cup

May 2015

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

References:

Available upon request

Evan Paul Hammer

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

Objective

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

Education

South Dakota School Of Mines and Technology, Rapid City, SD **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 ^A T _E 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	May 2015 — August 2015 (Full-Time)	
EMPLOYER	Innovative Systems	Rapid City, SD
JOB TITLE	Software Developer (Intern)	
LANGUAGES	C#, SQL, Xamarin.Forms, .NET Framework	
	Cross-platform mobile development (MVVM) in Xamarin Forms, C# back-end development/stored procedures in MSSQL	
PERIOD	May 2014 — August 2014 (Full-Time)	
EMPLOYER	Emit Technologies	Sheridan, WY
JOB TITLE	Software Developer (Intern)	
LANGUAGES	C#, JavaScript, HTML, .NET Framework, SQL	
	Front-end (web) development in C#, stored procedures in MSSQL, followed MVC development pattern	

EDUCATION

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

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

2 ABET: Industrial Experience Reports

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

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

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

2.1 Johnathon Ackerman

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

2.2 Daniel Andrus

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

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

2.3 Charles Bonn

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

2.4 Evan Hammer

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

2.5 Joseph Mowry

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

D

Acknowledgment

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

E

Supporting Materials

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

