FORMATION

CS 4911: Senior Capstone Final Report

LAUNCHED BY

 STEPHEN CONWAY

 SHREYYAS VANARASE

 WESLEY CARLAN

 TIMOTHY CHU

 CHARLES HANCOCK

COLLEGE OF COMPUTING

GEORGIA INSTITUTE OF TECHNOLOGY

Table of Contents

Introducing Formation

Vision: Motivation & Opportunity

Today, there is an absence of applications that allow individuals to quickly organize and actively direct group members to convenient meeting sites on a real time map. Such a functionality would be very useful to organizations or groups that meet frequently but do not have a standard meeting location. Currently individuals must manually determine teammates' locations and arrange for a convenient meeting site through numerous texts or phone calls. Often, miscommunication happens when individuals are not familiar with the nearby area. An application that resolves this group meeting issue would be able to quickly direct group members to convenient meeting sites on a real-time map.

This application, *Formation*, seeks to simplify the group meeting process by enabling individuals a form of communication with location. Individuals would be able to see the precise location of their teammates on a real time map. Based on the proximities of group members across the map, an individual could propose an optimal meeting location. The map would then place a marker at this site and allow group members to be navigated to the arranged meeting spot.

A simple example of a group that would benefit from such an app would be a fantasy sports league. A local league commissioner could create a group and add all the members of the league to the group. On draft day, the commissioner would see the locations of each member and propose a meeting location convenient for all members. On top of deciding a last minute meeting location, the application might serve as an interface to communicate with other team managers to either discuss trades or make announcements.

Tracking members' locations in real time has professional use cases as well. In fact, GTPD has previously requested "a custom application to integrate the camera and *officer tracking applications*". This application would allow GTPD members to track officer locations and even notify members of the GTPD group to request for backup by placing markers at specified locations. Police members would be able to congregate at the marker and the scene of a crime much more quickly. As another example, in the past, the Georgia Aquarium has asked a VIP project team to develop an application to track their tour guides and janitorial staff. Given our expectations for this app, *Formation* would potentially even allow Georgia Aquarium to easily perform this task.

Overall, *Formation* is meant to supplement normal communication by displaying users' locations and featuring group management functionality in a convenient, aggregate format. To facilitate further communication, the *Formation* would have group and personal chat features.

Problem Statement

The Problem of	Determining the location of team members either for deployment or for deciding meeting locations.
Affects	Project teams, friends, police departments, businesses.
The impact of which is	A multi-step process to figure out the locations of all other members with current solutions (text, phone, etc.) and the process becomes more difficult as the number of members grows.
A successful solution would be	An application that resolves this group meeting issue would be able to quickly direct group members to convenient meeting sites on a real-time map.

Figure 1: Problem Statement

Product Position Statement

For	Teams and businesses that need information about the location of group members and employees when making decisions
Who	Professional meetings and Social Gatherings
Our System	Android Application
That	Manages Locations and Group meetings
Unlike	GroupMe, GPS Tracker Pro
Our Product	Allows members of groups to see each other location. It will also be able to locate good meetup locations for group members.

Figure 2: Product Position Statement

Users

Any individual who is involved in any sort of group or organization is our potential user. Groups can be classified as either social or professional. Individuals will use our application to communicate and stay in contact with their peers. Also, the application's GPS will allow users to not only see where other members are, but also inform them where they are needed for assignments or meet ups. Though we will try ensure an intuitive UX, we assume our users will have the basic knowledge to navigate an average application.

Current Feature List

We currently incorporated basic GPS features that include tracking and displaying a user's current location to all members in the group. User will be able to route members to other members, which will display a red line over the route on the map. Users can also drop pins on the map with a message that can be seen by all members in the group. In general, a user will be able to click and drag a member's pin on a map or place a new marker at a preferred location to direct him to the new site. The app also supports basic communication. Users can send out messages that everyone in the group can see. Also, users can pin messages that they deem important so they do not get lost in numerous messages. Users can also message other users individually if they do not want the entire group to see a message.

Future Feature List & Constraints

In the future, we will incorporate basic group formation features into our application including creating a group, adding members, requesting addition into a group, deleting a group, removing a member. Adding to the communication features, users will also be able to mute the group if they do not want notifications at the time. In case users do not wish to have their location display on the map, users will have the ability to hide their locations from group members. Additionally, because location of all the members can be seen, we will implement a search function to help users easily find meeting locations based on the overall proximity of members to nearby landmarks.

Finally, there are two main constraints we aim to achieve within the application:

1) Availability - This application must run 24/7 and be able to be backed up at any time.
2) Portability - The application must be supported by all recent versions of Android.

Summary

Having understood the original vision and motivation behind *Formation*, in the following sections, we explore our fully developed Android application in greater detail. We begin by providing the final high level architecture of the application. We discuss the interaction of the individual classes in our final UML diagram and provide ER model and database schema diagrams to understand the structure of our database. The Activity flow diagram displays the various states of the client side application that an individual would perceive while using the application. After discussing the details of the software underlying *Formation*, we provide analysis on the qualitative parts of our capstone project. We consider the use cases of our application and tie in the effectiveness of Pivotal Tracker in managing the project scope. To ensure the functionality of our application we report on the tests we made and the significance of testing to our application. Finally, we reflect on our team dynamics and the overall capstone experience through the post mortem. For our future users, we also provide a succinct user manual and installation instructions for a seamless introduction to *Formation*.

Final Application Design

High Level Architecture Diagram

Note that in sprint 1, there were some concerns about readability of some of our diagrams. To rectify this, we have increased the size of the images as much as possible. Additionally, we have included links to the original images so that they may be more closely inspected.

Figure 3 displays our high level architecture diagram. The backend is a simple Node.js/MySQL interface. It communicates in JSON over HTTP, allowing for portability between platforms. Having the Node.js wrapper also allows us to switch out the database implementation with whatever suits the application's needs.

The front end consists of two layers: the main Android application with the usual Activities and supporting classes and the API interface. The API interface is not application-specific and can be reused with ease on another version of the Android application designed for tablets, a version of the application without the interface (for uses like tracking the locations of machinery rather than people), or even a Java desktop application for administrators. Please click here to access the full size image of the high level architecture diagram below.

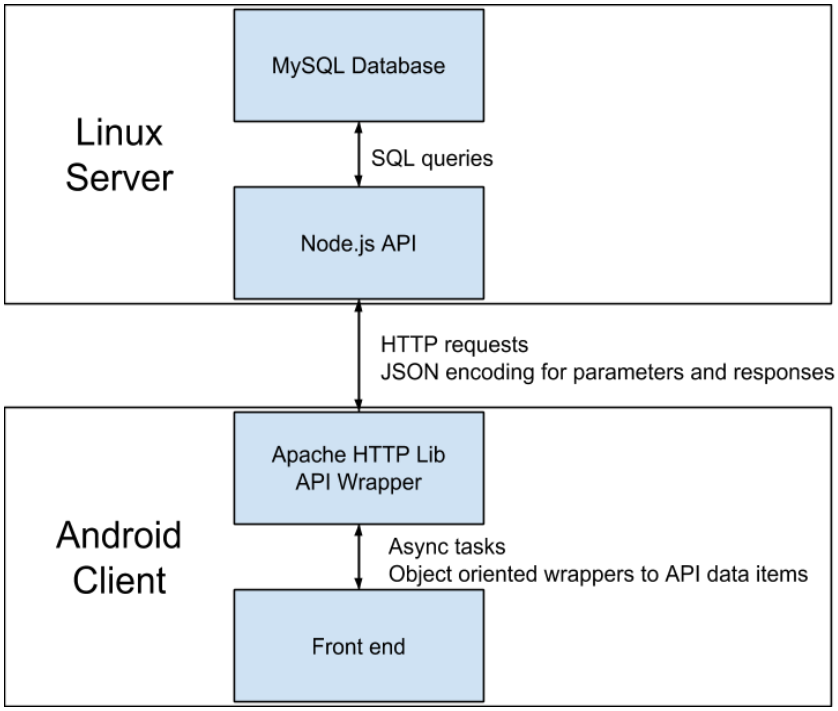

Figure 3: High Level Architecture Diagram

Database ER Diagram

Figure 4 below shows our database ER diagram. As we can see, Users have a member-of relationship to Teams such that each user can be a member of many teams. For each user, we store the minimum data required to run the group management and location features of our application to minimize privacy issues. For our chatting feature, users can author messages and pin them to specific locations on the map. Each team holds its own messages so that messages are not displayable to members outside the team users are members of. Messages contain location data so that users can geotag & pin messages to the map and access them at any point in time. Please click here to access the full image for the database ER diagram below.

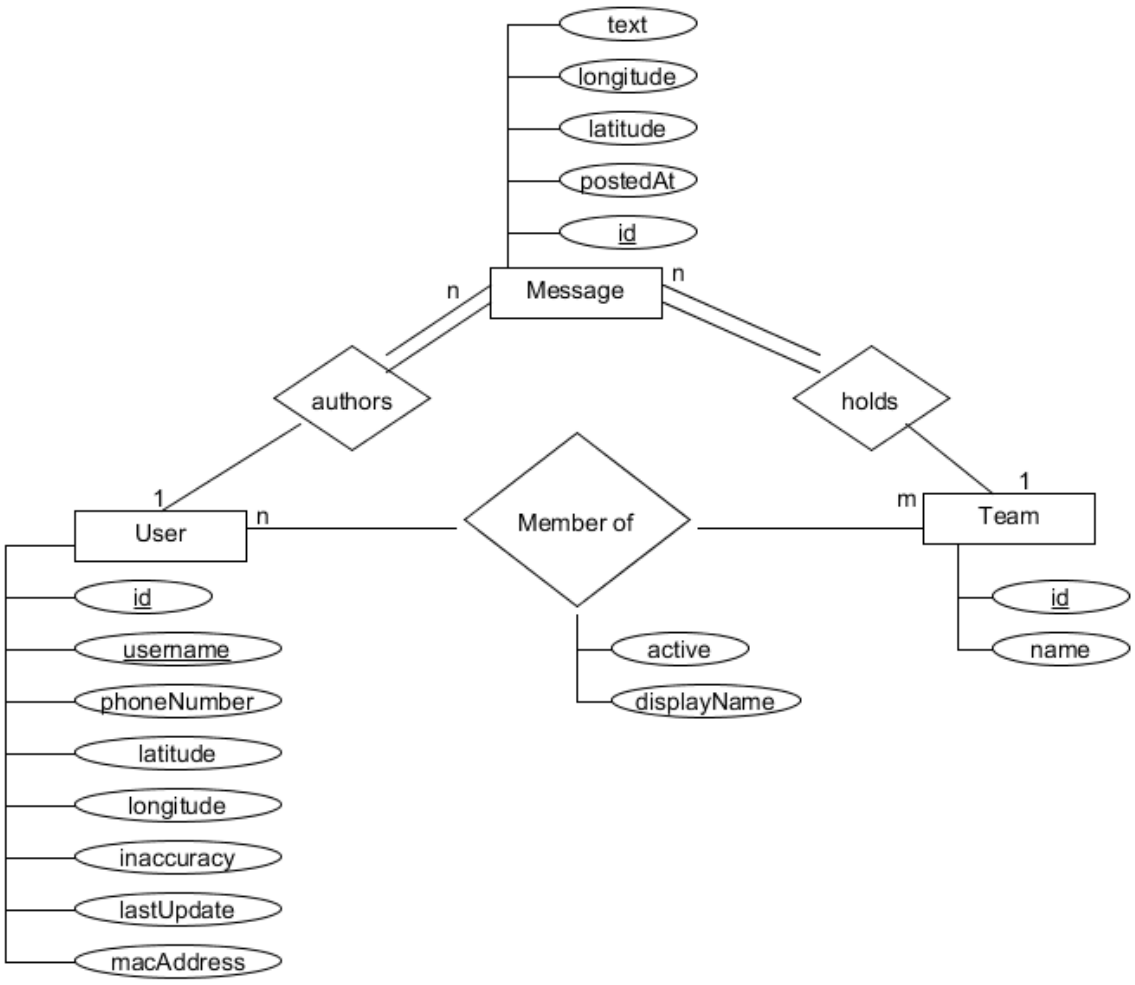

Figure 4: Database ER Diagram

Database Schema Diagram

Figure 5 below displays our database schema. As mentioned previously, Users have Teams they are a member of, and can set preferences for each Team. The messaging system in the application is also done through the database, where messages are primarily associated with Teams (in the API wrapper they are accessed through the representative Team objects), but also associated with the user that authored them. Please click here to access the full image for the database schema below.

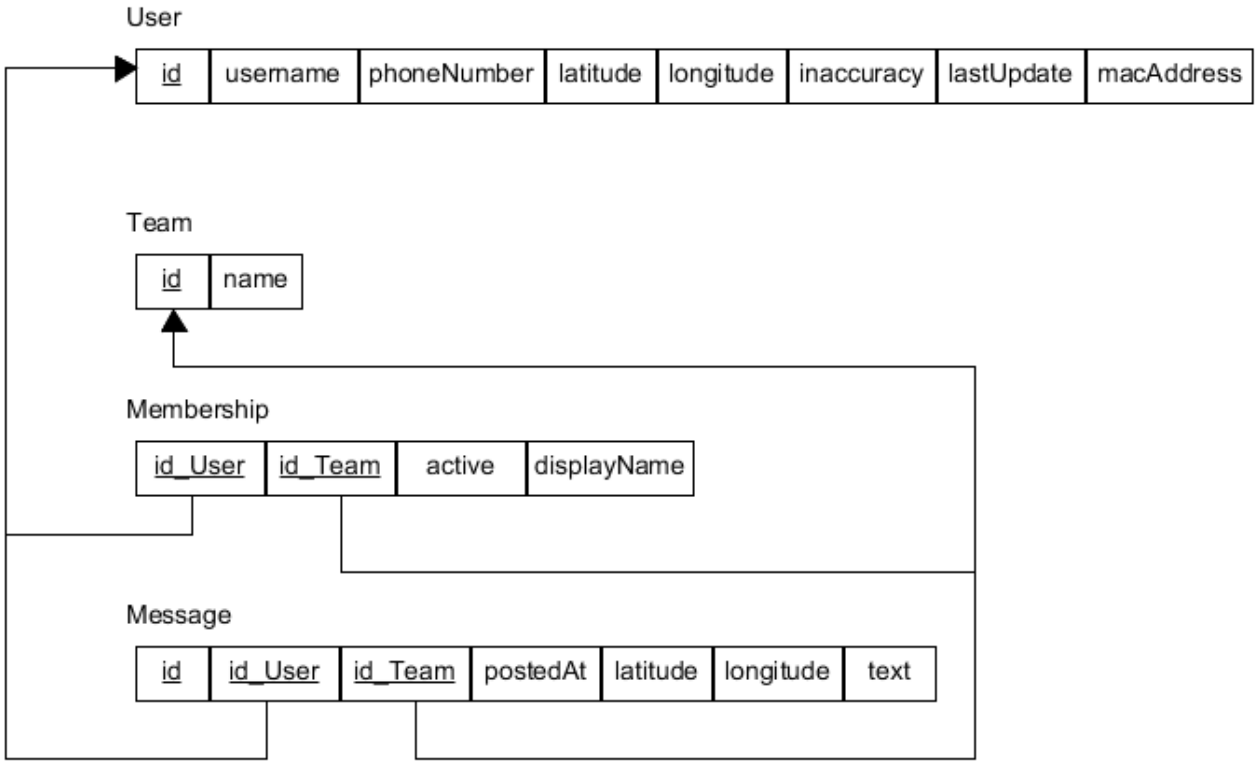

Figure 5: Database Schema Diagram

UML Class Diagram

The UML class diagram is too large to display in this document format, so we have attached it here. It is an SVG file that can be displayed nicely by a browser such as Chrome or Internet Explorer once downloaded. Please make sure that the zoom is set to 100% on IE to view clearly.

The detailed diagram also contains package information, including references to notable external libraries. We also provide the more abstracted package dependency diagram in **Figure 6** for convenience. Please click here to access the full size image of the package dependency diagram below.

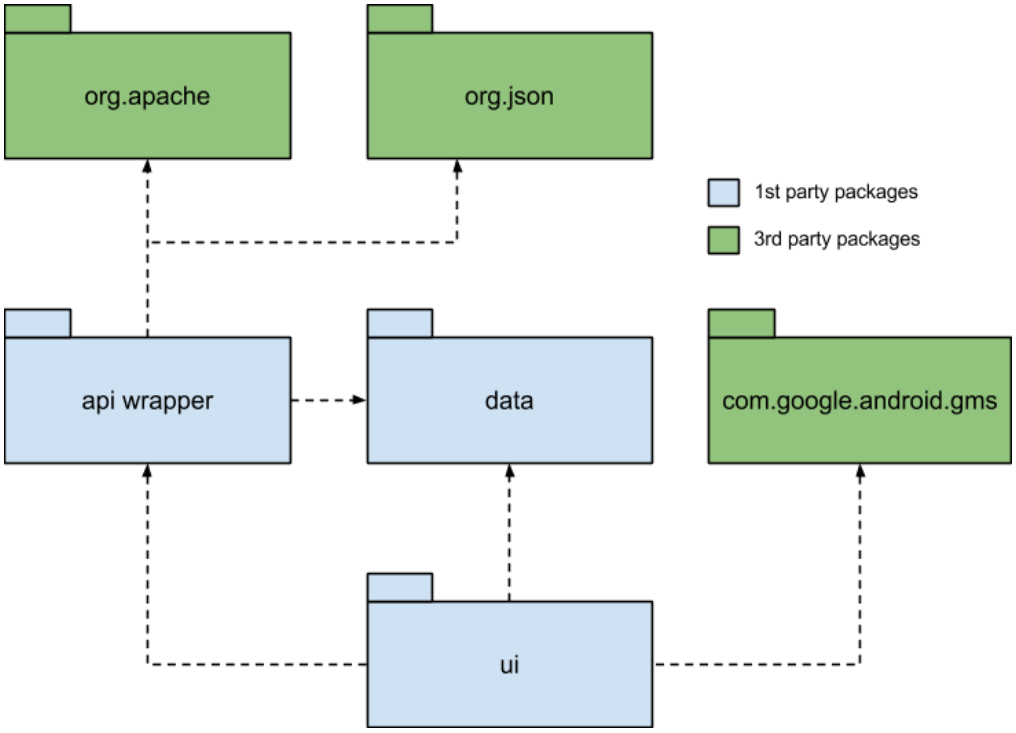

Figure 6: Package Dependency Diagram

API

We implemented the API on the server using Node.js. We found that documenting the functionality using a Google Sheet was extremely useful, since no one had to decipher Node.js code to figure out how to interact with the server and there was one authoritative source on the current operations. In this same spreadsheet we documented the database schema and JSON structures. These documents are all organized into different tabs in the Google Sheet here.

Activity Flow

Figure 7 below displays the activity flow diagram. It consists of the various states of the client side application that an individual would perceive while using the application. The core client side functionality of *Formation* resides in the Google Maps Activity. Most UI components and transitions occur around it and add/remove information on top of it. The main menu portion was left for debugging and settings purposes, but will be removed in later versions of the application. Please click here to access the full size image of the activity flow diagram.

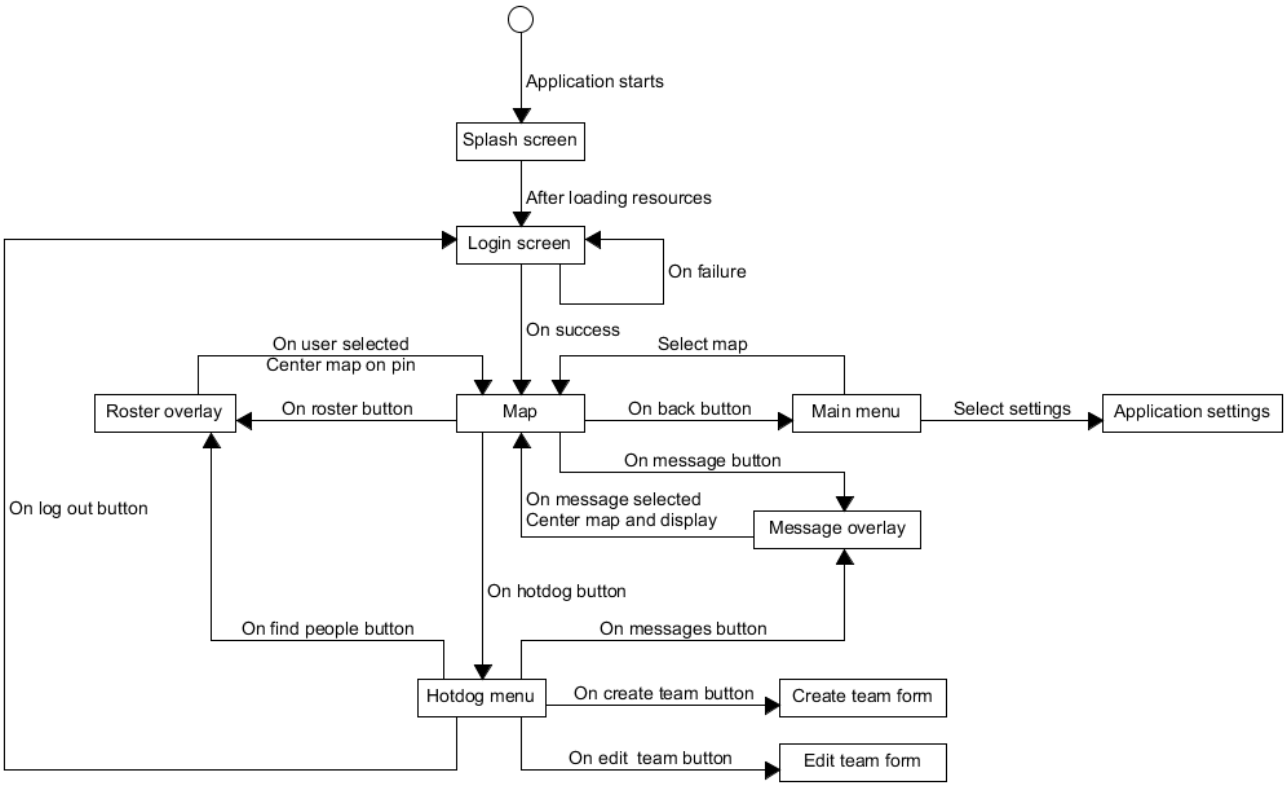

Figure 7: Activity Flow Diagram

User Stories and Test Report

We used Pivotal Tracker to mange our user stories and feature tests. **Figure 8** below shows an example of one of our completed user stories and the feature tests. Each user story contains procedures to test the functionality of the feature and contains EXPECTED RESULTS. For any test that fails we have also included the current functionality under ACTUAL RESULTS. Please see the following link to access our Pivotal Tracker site.

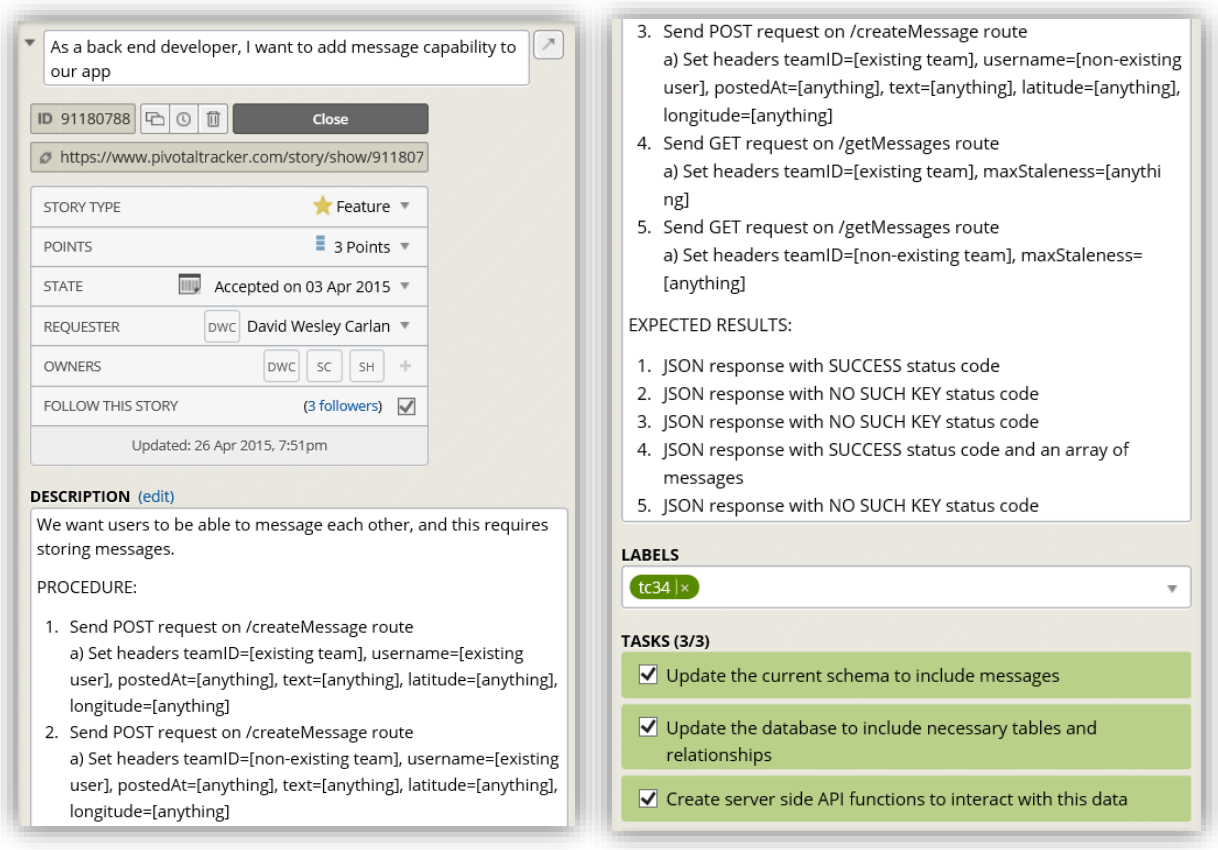

Figure 8: User Story and Test Cases

Post Mortem

Terminology Overview

Figure 9 is a list of definitions for the terminology used throughout this literature.

Acronym/Terminology	Definition
Formation	The name of our Android application
Georgia Tech Police Department (GTPD)	Police Department located at the Georgia Institute of Technology
Integrated Development Environment (IDE)	A software application that provides a simple tools and facilities for programmers for software development
Application Program Interface (API)	A set of tools and protocols that help with software development
User Interface (UI)	Design of the way users interact with the application
Android Studio	An IDE created specifically for Android software development
Eclipse	An IDE that is commonly used by many software developers
Pivotal Tracker	An online application that manages tasks and work to be done by a group.

Figure 9: Terminology Overview

Historical Overview

The project was split up into three sprints over the course of the semester. Our team met every Monday at 6:30pm to discuss progress made every week and the direction the team should take over the next week

Sprint 1

At the beginning of sprint 1, we were still brainstorming ideas. After pitching three different ideas, the team decided that an application that managed group activity and group location was the most interesting. A lot of research went into deciding the best way to go about producing the application. Because our product is an Android application, we elected to use Android Studio over Eclipse as our IDE. Android Studio has many helpful tools and is supported by Google to make Android development much simpler. Also, future updates and maintenance will be easier with Android Studio. After

experimenting with Google Maps API, we decided to use that instead of building the map from scratch. With this API, we were able to obtain real time location information and add markers to the map. We used MySQL as our database since we were most familiar with it. We also began to set up servers, pull together old code segments that would be useful, and coded a very generic initial app framework for testing and debugging.

In addition to back end decisions, we discussed how the application would function and how it would work. We also created mockups of initial UI design as shown in **Figure 10** below.

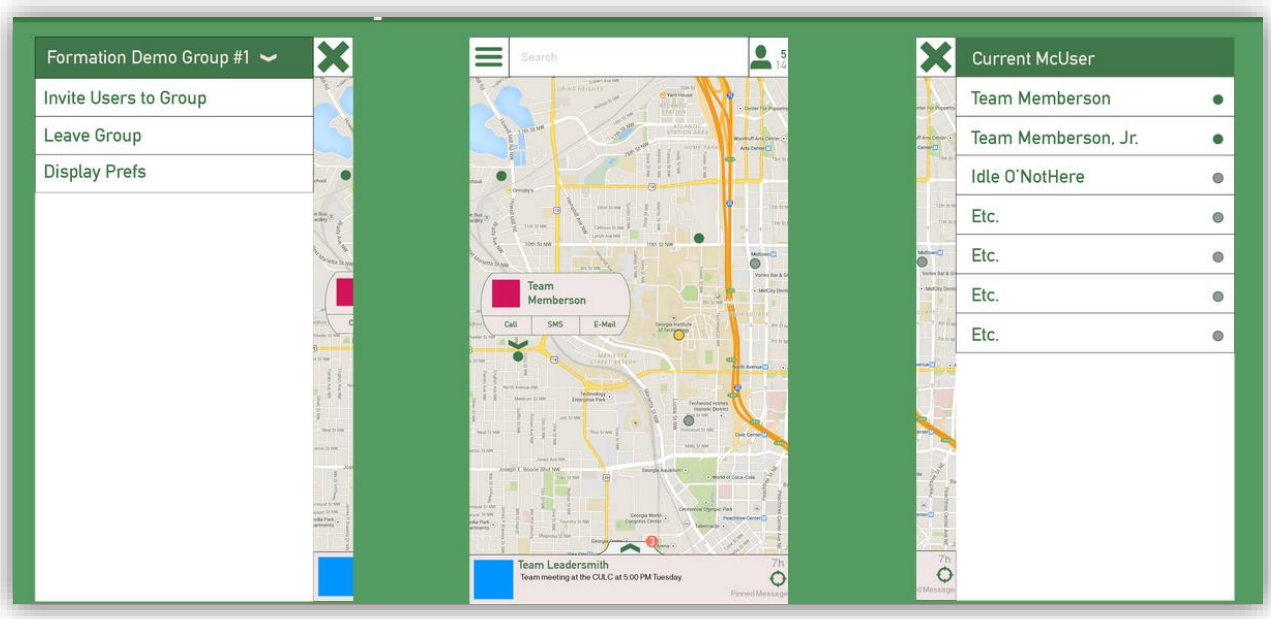

Figure 10: Sprint 1 UI Mockups

By the end of Sprint 1, we had a Node.js database and Google Maps set up and working. In this phase of our development, users would be able to see themselves on a map and drop pins on a map. Users could also search for locations.

Sprint 2

Having planned most of the application architecture and user stories, Sprint 2 was focused on completing tasks and improving app functionality. We started by figuring out the Google API authentication to get the map to display. Once we had that, we separately started researching map features, started coding the API wrapper package, and started implementing the API functions on the server. A lot of work was done on the API to get information about the user such as contact information and location. We performed some basic testing of our API calls to make sure they worked. Also, a basic login UI was added.

Once the wrapper package was completed, we connected everything together. The front end invoked calls and received information to display from the wrapper which communicated with the server via HTTP. The user and team creation and management were faked as this point, since the focus was on functionality and these would provide logistical barriers. By the end of Sprint 2, we had full front to back capabilities and users could see not only themselves, but other users on a map.

Sprint 3

At this stage we implemented the more polishing aspects of the application: Google plus authentication, the splash screen, the sliding menus, and a general UI overhaul. We completed the original proposed API implementation and integrated it with the map features we researched, including route finding and location search. We also revamped the entire stack (database, node.js, Android application) to support real time messaging. We added third party login authentication with Google+, which allowed users to login using their already existing Google+ accounts. **Figure 11** below shows *Formation* in the map screen.

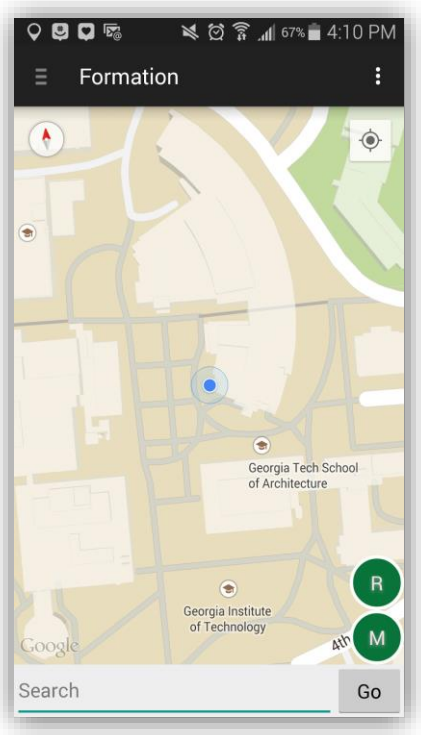

Figure 11: Sprint 3 Formation Map Screen

Organizational Structure

Figure 12 shows our organizational structure over the past semester. Our group was organized quite evenly; each member focused on completing his work for each week and then assisted others to meet weekly goals.

Name	Role 1	Role 2
Prof. Omojokun	Sponsor	Course Professor
Stephen	Lead Backend Database & API Developer	Client Side – Backend Functionality Integration
Shreyyas	Lead Client Side Features Developer	Code Testing
Wesley	Lead Server Side Developer	Scrum Master
Tim	Head Google+ Authentication	UI – Client side Integration
Charles	Head UI Developer	UI – Client side Integration

Figure 12: Group Member Task Allocation

Project Metrics

As of the v1.0 release on April 27, 2015, these are the metrics of Formation.

Burndown

Figures 13-15 are the burndown charts for each of the sprints. These graphs are generated through Pivotal Tracker, a tool we used to record our progress throughout the semester.

Sprint 1

As **Figure 13** shows, in the first phase of the project until Jan 26 we decided on our project and spent time planning our application architecture. Afterwards we made steady progress development and completed our expected tasks by the sprint 1 deadline.

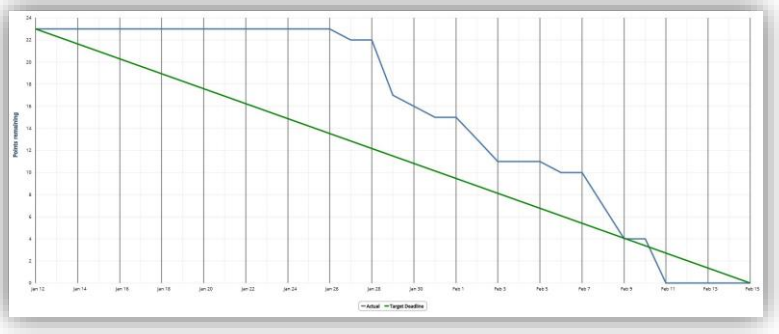

Figure 13: Sprint 1 Burndown Chart

Sprint 2

Figure 14 shows our progress during Spring 2. During this sprint, we focused on adding varied features to the client side and server side of the application. We kept on track for most of this sprint until we reached Spring break at the beginning of March.

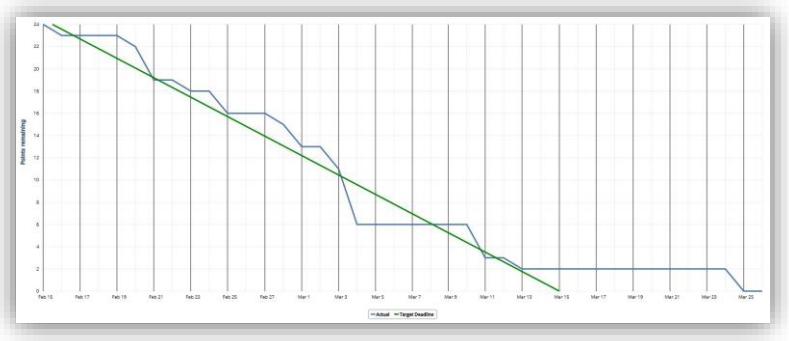

Figure 14: Sprint 2 Burndown Chart

Sprint 3

Figure 15 shows our progress for Sprint 3. In this sprint, we polished the application and produced a demo ready application by our final presentation date mid-April. For the first couple weeks of this sprint our teammates had midterms and as soon as these were over we focused back to our application.

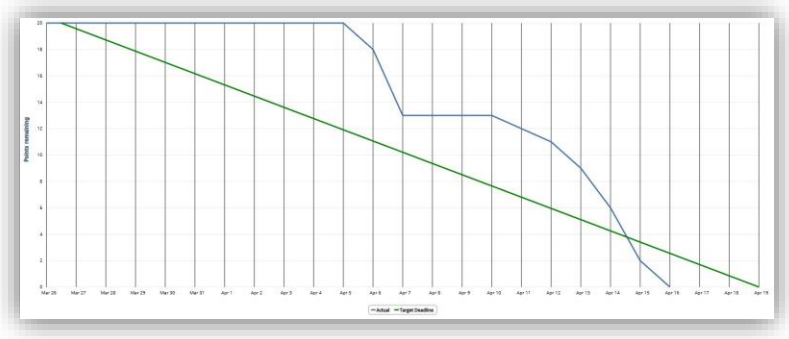

Figure 15: Sprint 3 Burndown Chart

Lines of Code

In our code, we produced 14,700 lines of code. 50% of those lines were source code, 40% of those lines were comments, and 10% were empty lines. Overall we did a good job commenting the code we produced so it would be easier for future use. These stats were found using the statistics plugin found at https://plugins.jetbrains.com/plugin/?idea&id=4509.

Lessons Learned

This project allowed us to design an application from scratch to completion with full stack integration. During Sprint 1 we brainstormed various ideas for the potential semester long capstone project. We quickly realized the scope of the application would be difficult to measure at the beginning and so intensive planning was required to ensure that we would have a successful product by the end of the semester. After finalizing on the application idea, we allocated tasks to each of us based on our strengths. We quickly learned that weekly progress report meetings were the most efficient manner of keeping our project on track. Since each of us worked on tasks that may have been interrelated with each

other, we found that working in small sub-teams was more effective than having large weekly coding sessions with the team of five.

During Sprint 2 we focused on adding features and backend support to our client side GUI. As a team we communicated well among one another about tasks we needed to finish and asking for help when we individually needed assistance in completing that week's task. We learned to improve our Pivotal Tracker management and assigned a Scrum Master to allocate work for each week. This allowed us to view our objectives more clearly and visually track our progress more efficiently.

During Sprint 3 we focused on refining existing and adding new features to the application. We also work on overall application robustness. By this phase our time management and task allocation was very well handled. We were able to complete the demo ready application within time.

Overall, our team has had a spectacular senior capstone experience. By the end of the semester, each of us had contributed to developing a working full stack Android application. The expertise everyone brought to building *Formation* was truly fantastic and it shows through the success of our application. As we look into the future, we hope to expand *Formation* even further and produce a marketable app after improving its robustness, UI capabilities and group management features.

Formation Installation Instructions

To install Formation onto your Android device:

1. Download the Formation APK at https://github.gatech.edu/sconway30/Formation/blob/master/app/app-release.apk
2. Make sure your device allows third-party APKs to be installed in Settings → Applications.
3. Install the APK to your device.
4. Run the *Formation* app.

NOTE: If Google Maps does not properly display on your android device

5. Consider generating your own debug SHA1 key by following the instructions in the link below.
    a. https://developers.google.com/maps/documentation/android/start#get_an_android_certificate_and_the_google_maps_api_key.
6. Add this SHA1 to the Google developers console under **APIs and Credentials** by logging into console.developers.google.com with the following login criterion.
    a. username: formation4908@gmail.com
    b. password: formation4911
7. Repeat Steps 3 and 4.

Formation User Manual

Introduction

Formation is an Android application that allows groups of users to create and manage teams of users with a cohesive location-based experience. In a Formation Team, you can see where each member is and send messages to a group chat system. You can also view a directory of the users in the team and call their phones right out of the app. This manual covers functionality through April 27 2015.

Main Map Screen

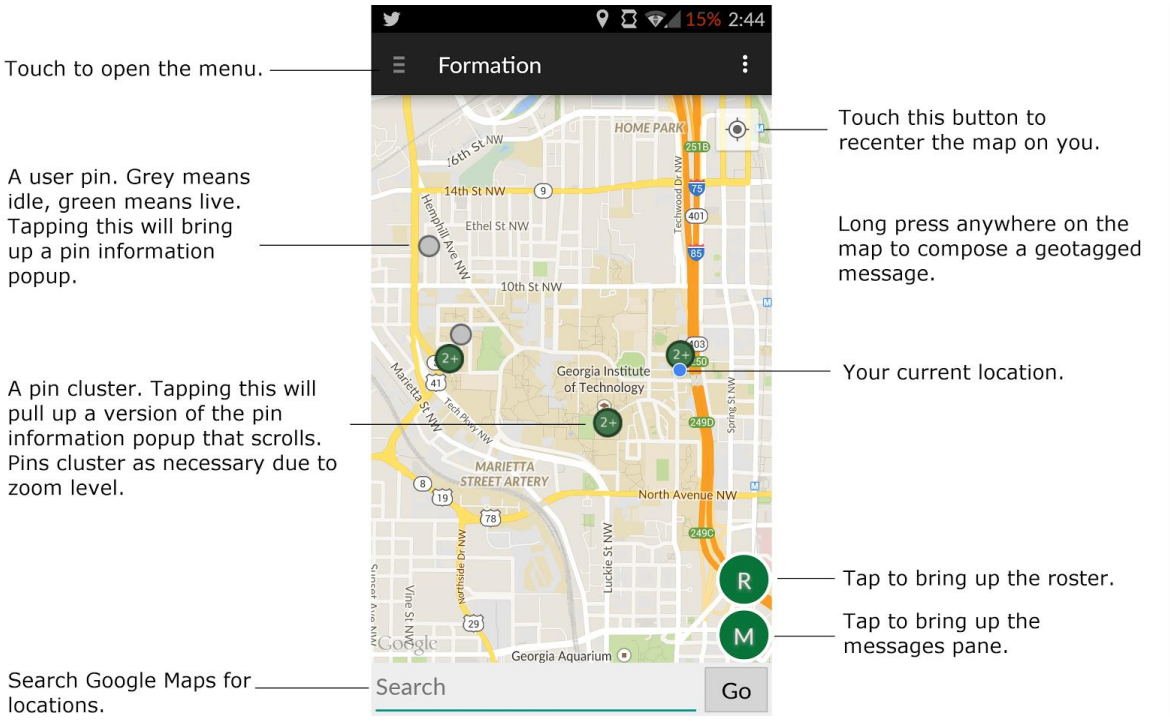

Figure 16: Main Map Screen

Creating a Team

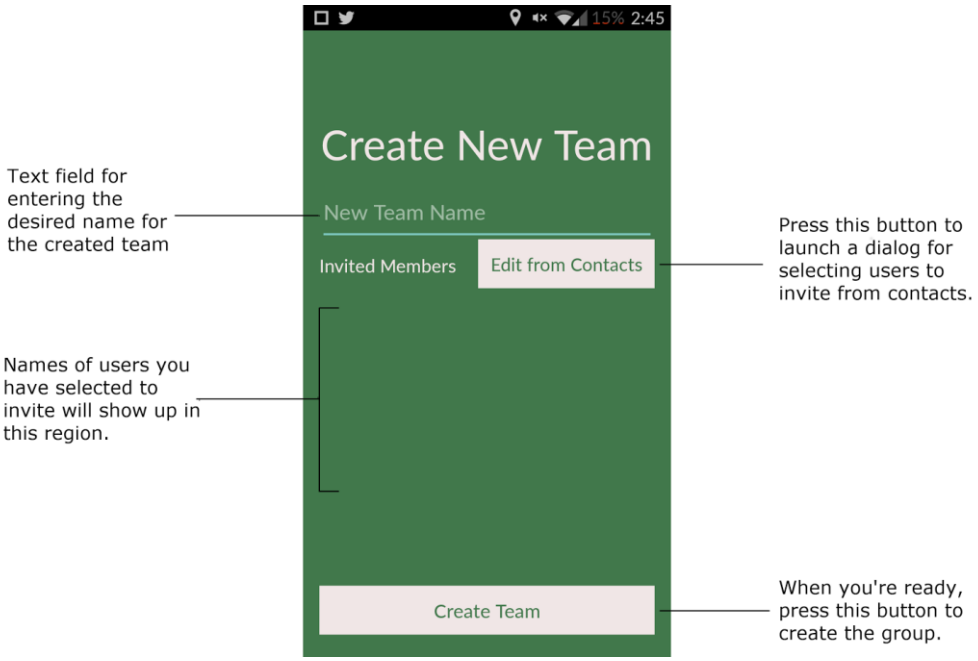

Figure 17: Creating a Team

Inviting Users to Team

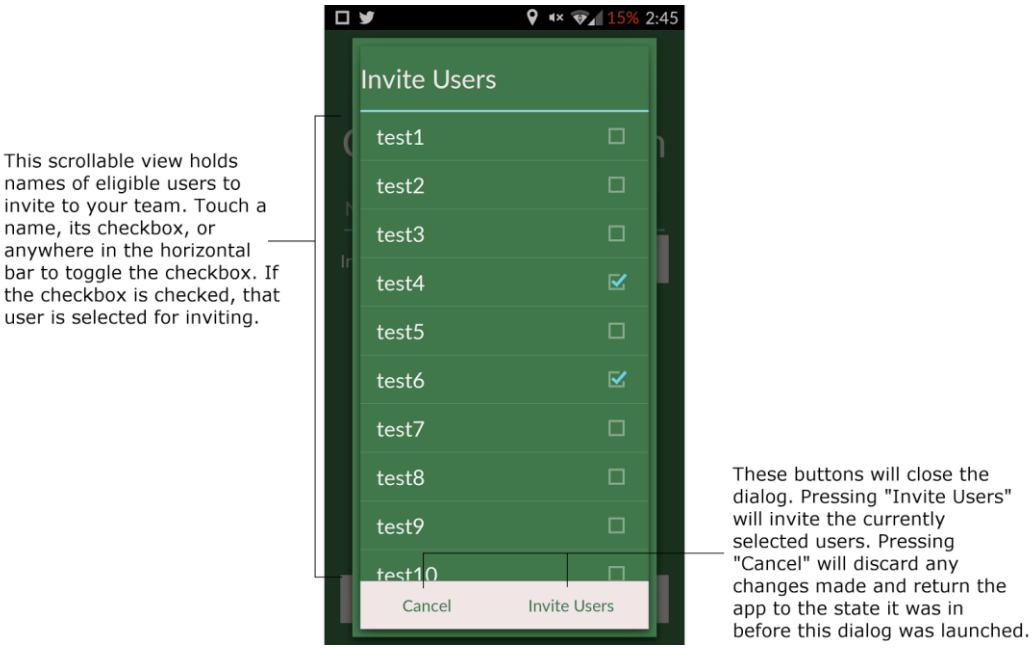

Figure 18: Inviting Users to Team

Messaging

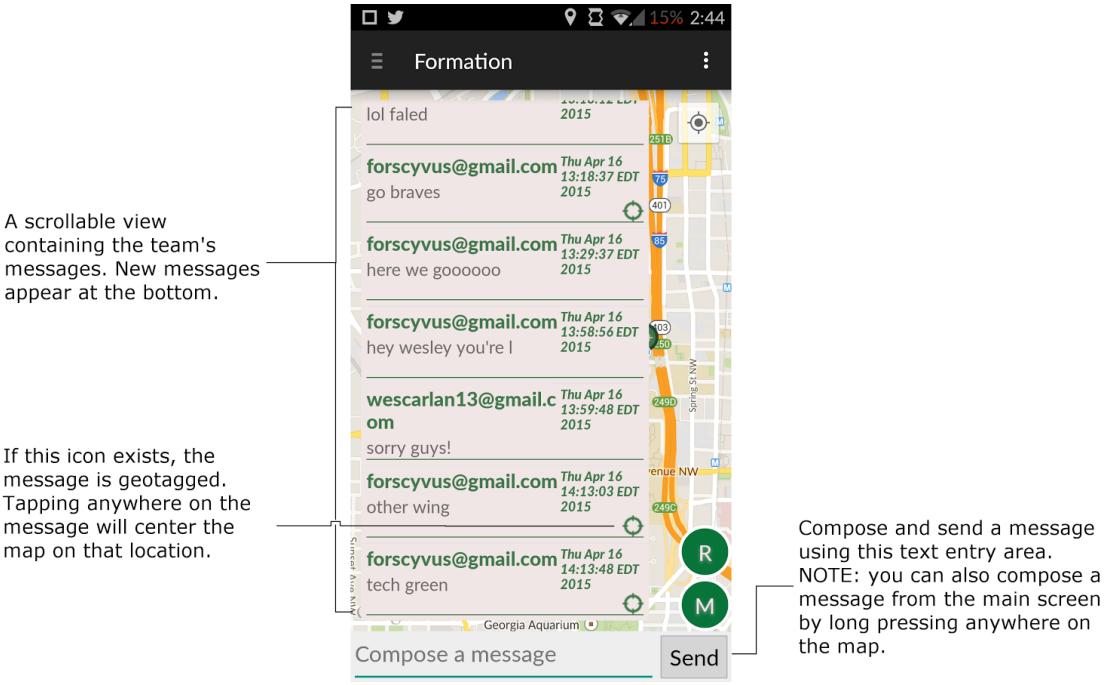

Figure 19: Messaging

Roster

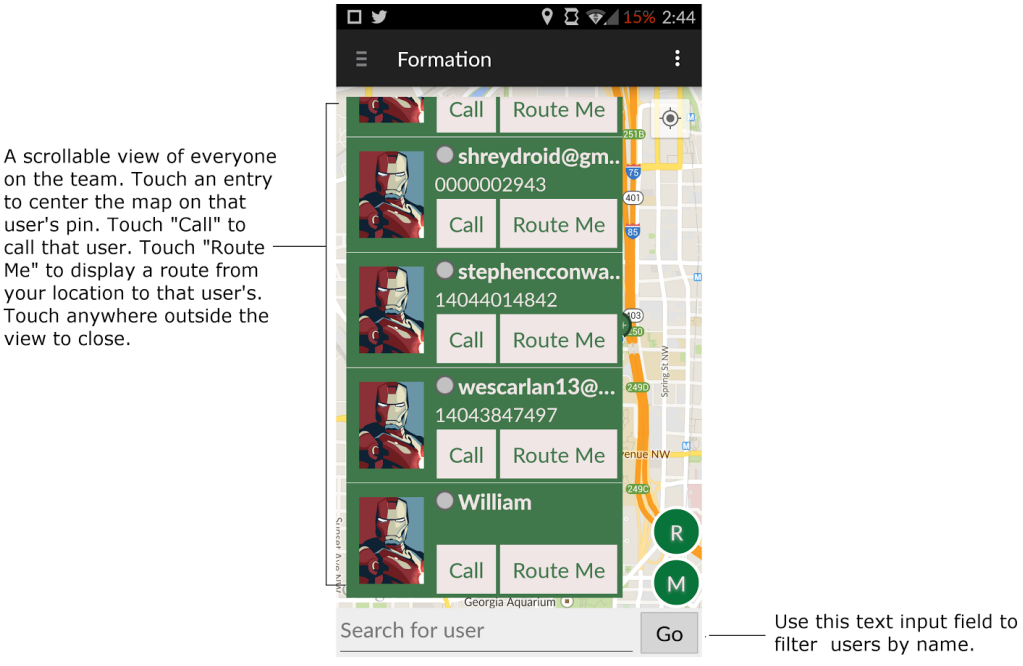

Figure 20: Roster

Pin Info Boxes

Pin info boxes look and function the same as entries in the Roster and Messages views.

The pins touched to trigger the popups. The green M pin is a geotagged message. It will also appear in the messages pane.

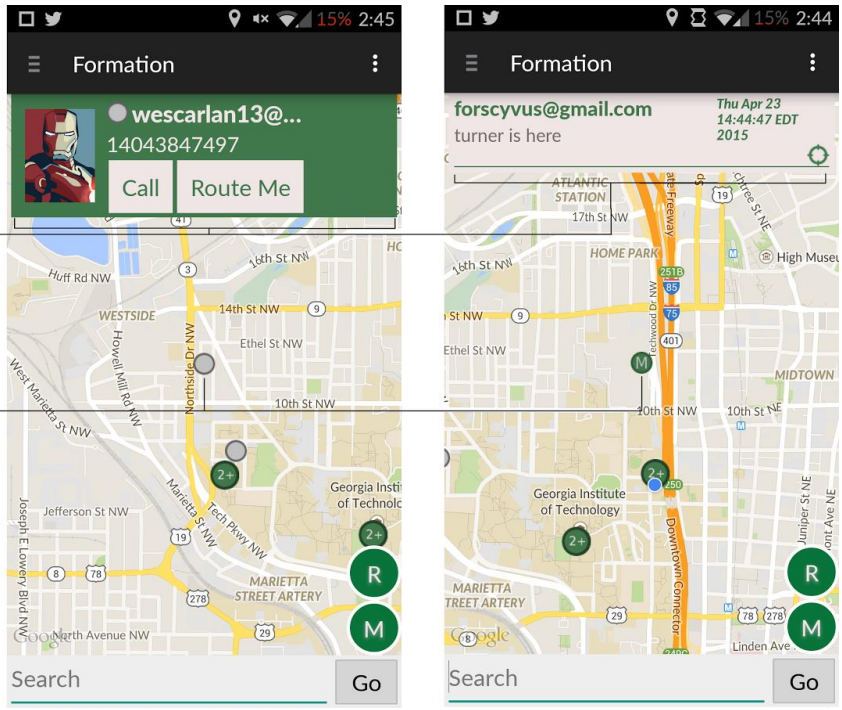

Figure 21: Pin Info Boxes