Crowd Control

Senior Design Final Documentation

Bowtaps

Johnathan Ackerman Daniel Andrus Charles Bonn Evan Hammer
Joseph Mowry

May 4, 2016

Contents

List of Figures

List of Tables

List of Algorithms

Overview Statements

0.1 Mission Statement

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

0.2 Elevator Pitch

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

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

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

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

Document Preparation and Updates

Current Version [1.5.3]

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

Revision History

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*
1/15/16	*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*
4/29/16	*Johnathan Ackerman*	*1.6.0*	*Completed Chapter 6*
4/22/16	*Joeseph Mowry*	*1.6.1*	*Completed Chapters 0, 1*
4/30/16	*Evan Hammer*	*1.6.2*	*Completed Chapters 2, 3*
5/2/2016	*Joeseph Mowry*	*1.6.3*	*Completed Chapter 4*
5/2/2016	*Charles Bonn*	*1.6.4*	*Completed Chapters 8, 9, 10*
5/2/2016	*Johnathan Ackerman*	*1.6.5*	*Completed Chapter 7*
5/3/2016	*Dan Andrus*	*1.6.6*	*Completed Chapter 5*

1

Overview and Concept of Operations

1.1 Bowtaps and Its Team

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

1.2 Crowd Control

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

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

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

1.2.1 Purpose of the System

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

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

1.3 Business Need

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

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

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

1.4 Deliverables

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

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

1.5 System Description

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

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

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

1.5.1 Integrated Group Messaging

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

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

1.5.2 GPS Location Services

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

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

1.5.3 Group Management Features

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

1.5.4 Suggestions

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

1.6 System Overview and Diagram

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

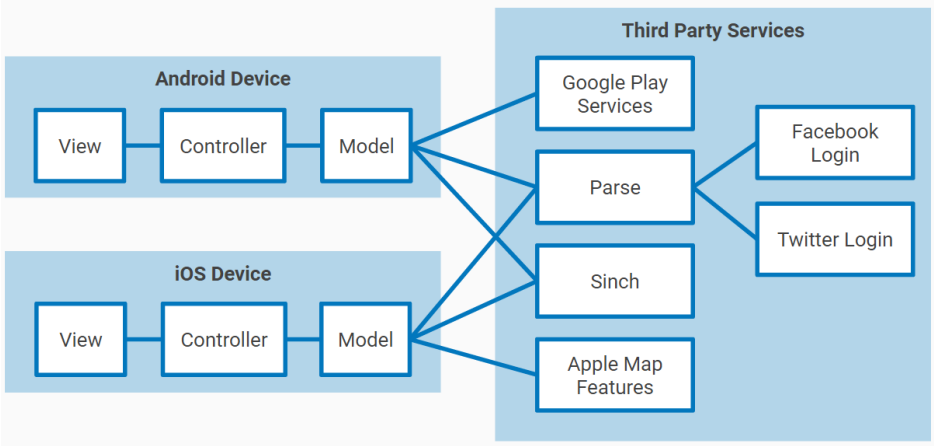


Figure 1.1: Basic System Flow Diagram

1.7 Technologies Overview

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

1.7.1 Google Play Services

1.7.1.a Description

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

1.7.1.b Usage

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

1.7.2 Apple Map Features

1.7.2.a Description

Apple Map Features is the native iOS API for mapping features. With this it allows for commiseration between a map and your gps location along with other mapping features.
REFERENCE LINK: `https://developer.apple.com/maps/`

1.7.2.b Usage

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

1.7.3 Parse

1.7.3.a Description

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

1.7.3.b Usage

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

1.7.4 Sinch

1.7.4.a Description

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

1.7.4.b Usage

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

2

User Stories, Requirements, and Product Backlog

2.1 Overview

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

2.2 User Stories

2.2.1 User Story #1

As a user I want to join a public group.

2.2.1.a User Story #1 Breakdown

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

2.2.2 User Story #2

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

2.2.2.a User Story #2 Breakdown

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

2.2.3 User Story #3

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

2.2.3.a User Story #3 Breakdown

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

2.2.4 User Story #4

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

2.2.4.a User Story #4 Breakdown

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

2.2.5 User Story #5

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

2.2.5.a User Story #5 Breakdown

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

2.2.6 User Story #6

As a user i want to leave a group.

2.2.6.a User Story #6

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

2.2.7 User Story #7

As a user I want the ability to log-in with a custom account.

2.2.7.a User Story #7 Breakdown

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

2.2.8 User Story #8

As a user I want to log-in with Facebook

2.2.8.a User Story #8 Breakdown

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

2.2.9 User Story #9

As a user I want to log-in with Twitter

2.2.9.a User Story #9 Breakdown

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

2.2.10 User Story #10

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

2.2.10.a User Story #10

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

2.2.11 User Story #10

As a user i would like my information protected.

2.2.11.a User Story #10 Breakdown

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

2.2.12 User Story #11

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

2.2.12.a User Story #11 Breakdown

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

2.2.13 User Story #12

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

2.2.13.a User Story #12 Breakdown

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

2.2.14 User Story #13

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

2.2.14.a User Story #13 Breakdown

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

2.3 Requirements and Design Constraints

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

2.3.1 System Requirements

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

2.3.1.a iOS Requirements

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

2.3.1.b Android Requirements

- Use Google Maps
- Connect with Parse API for back-end data storage

2.3.1.c Parse Requirements

- Add users to groups
- Remove users from groups

2.3.2 Network Requirements

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

2.3.3 Development Environment Requirements

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

2.3.4 Project Management Methodology

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

- GitHub issues will be used to keep track of current status as well as backlogs for the product.
- There will be 3 total sprints over 2 semesters for this products.
- The sprint cycles are 3 weeks long.
- Progress reports will be submitted to Dr. McGough and Brian Butterfield at the end of each sprint.
- 4 Github repositories will be used for source control, one for each platform, one for the server code, and one for documentation.

2.4 Specifications

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

2.5 Product Backlog

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

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

2.6 Research or Proof of Concept Results

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

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

2.6.1 iOS Proof of Concept Screen Shots

Below are screen shots from the iOS version of Crowd Control.

2.6.2 Android Proof of Concept Screen Shots

Below are screen shots from the Android version of CrowdControl.

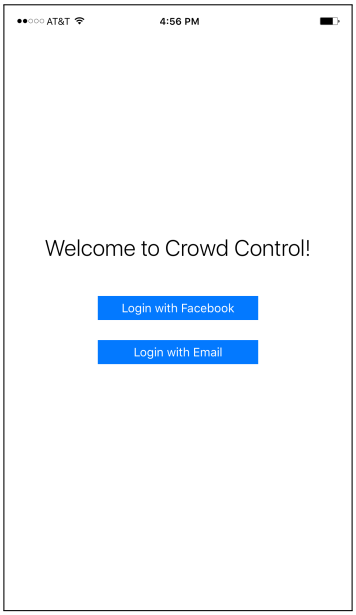


Figure 2.1: iOS login select screen

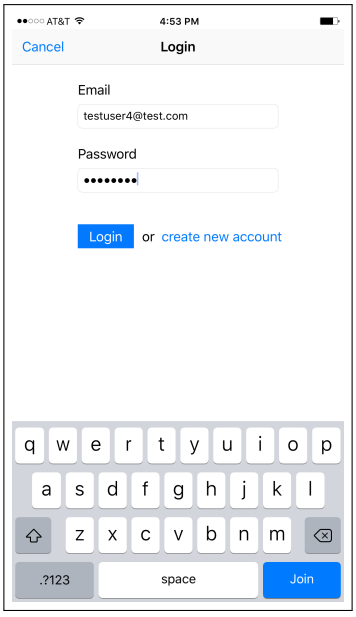


Figure 2.2: iOS email login screen

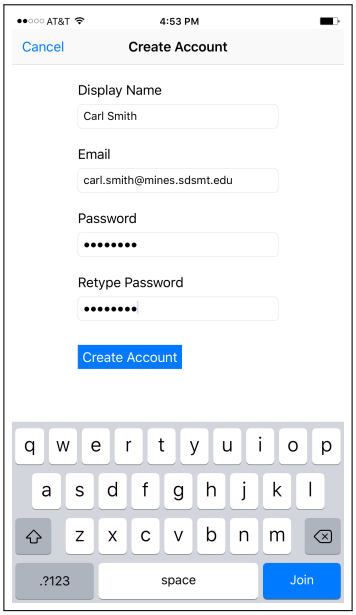


Figure 2.3: iOS create account screen

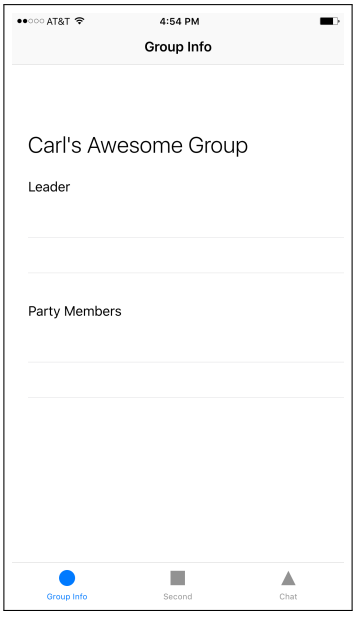


Figure 2.4: iOS group infomation screen

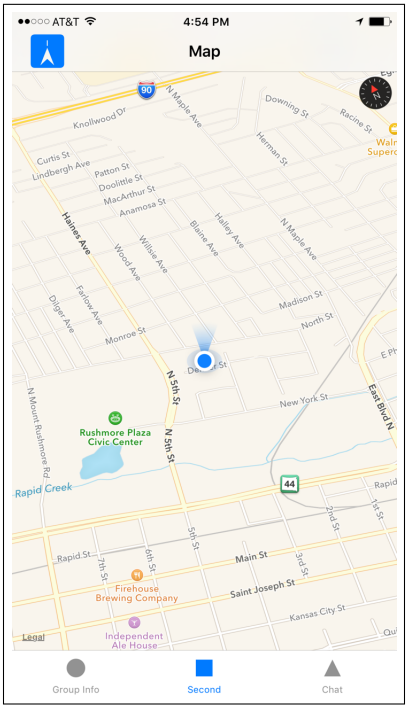


Figure 2.5: iOS map view screen

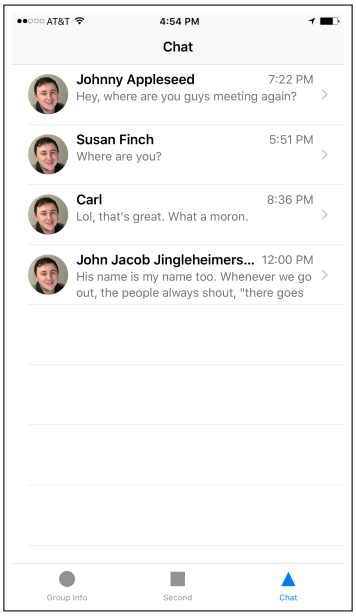


Figure 2.6: iOS messaging main screen

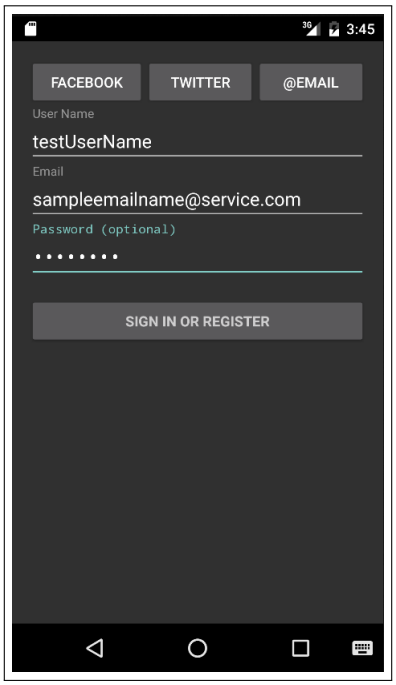


Figure 2.7: Android login screen

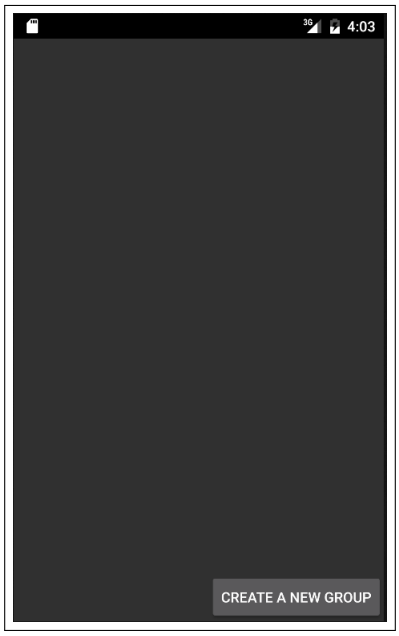


Figure 2.8: Android create group screen

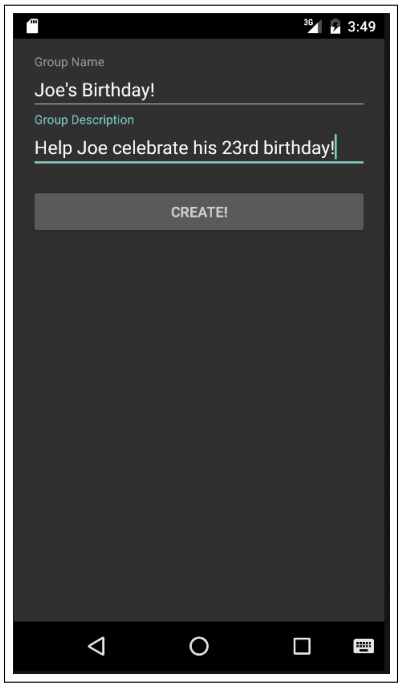


Figure 2.9: Android group information screen

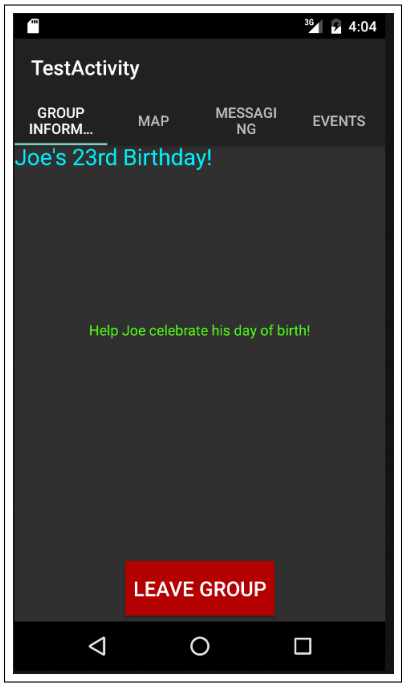


Figure 2.10: Android group join screen

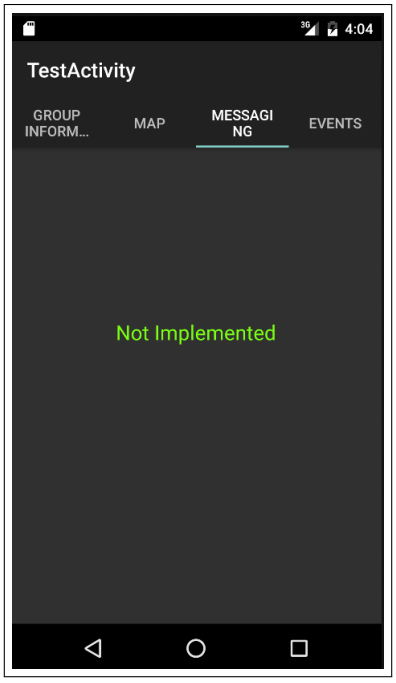


Figure 2.11: Android messaging main screen

3

Project Overview

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

3.1 Team Member's Roles

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

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

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

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

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

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

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

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

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

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

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

3.2 Project Management Approach

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

3.3 Stakeholder Information

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

3.3.1 Customer or End User (Product Owner)

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

3.3.2 Management or Instructor (Scrum Master)

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

3.3.3 Investors

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

3.3.4 Developers –Testers

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

3.4 Budget

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

3.5 Intellectual Property and Licensing

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

3.6 Sprint Overview

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

3.7 Terminology and Acronyms

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

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

3.8 Sprint Schedule

Below is a table of dates for each sprint.

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

3.9 Timeline

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

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

3.10 Backlogs

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

3.10.1 Sprint 1 Backlog

- Design UX
    1. Create groups
    2. Leave groups
    3. Group messaging
    4. Start page
- Database
    1. Design database schema
    2. Implement database on Parse

- Design application layers (MVC)
- Set up GitHub repository

3.10.2 Sprint 2 Backlog

- Code UX
    1. Mapping features
    2. Messaging UI
- Model
    1. User Model
    2. Communication Layer
    3. Link back-end and front end
- Implement Cloud code
- Business Plan

3.10.3 Sprint 3 Backlog

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

3.10.4 Sprint 3.5 Backlog

- iOS
    1. Login/Logout
        (a) Improved login/sign up screens
        (b) Logout feature added
    2. Settings
        (a) Settings screen implemented
        (b) Logout functionality nested in the Settings screen
    3. Groups
        (a) Leaving/Joining a group implemented
        (b) Basic group operations
        (c) Detect if users are in a group
- Android
    1. Login
        (a) Automatic login on startup (from data store)

   (b) Login to existing account via email address
2. Settings
   (a) Page layout created and linked from Group Join page
   (b) Logout functionality implemented
3. Groups
   (a) Leave button implemented
   (b) Tested adding/removing users from groups

- Misc/Transitional

1. Further documented Android code to prepare for team merge
2. Android code review with iOS team, to prepare for team merge

3.10.5 Sprint 4 Backlog

Week 1

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

Week 2

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

Week 3

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

3.10.6 Sprint 5 Backlog

Week 1

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

Week 2

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

Week 3

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

3.10.7 Sprint 6 Backlog

Week 1

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

Week 2

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

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

Week 3

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

3.11 Development Environment

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

3.12 Development IDE and Tools

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

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

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

Parse Parse Accounts can be obtained at parse.com

3.13 Source Control

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

3.14 Build Environment

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

3.15 Development Machine Setup

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

4

Design and Implementation

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

4.1 Architecture and System Design

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

4.1.1 Design Selection

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

4.1.1.a Early Design Ideas

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

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

4.1.1.b Improvements to Early Designs

In reflection of the shortcomings of the first design iteration, the database schema was restructured to better fit Crowd Control's needs. The current design consists of eight data tables, shown in Figure 4.2 below. More entities were added to the overall schema. One such addition was a "CCUser" table to hold the public data associated with a user. It is important to note that the "User" table was left to hold a user's private data. Those changes and some additions resulted in the below schema.

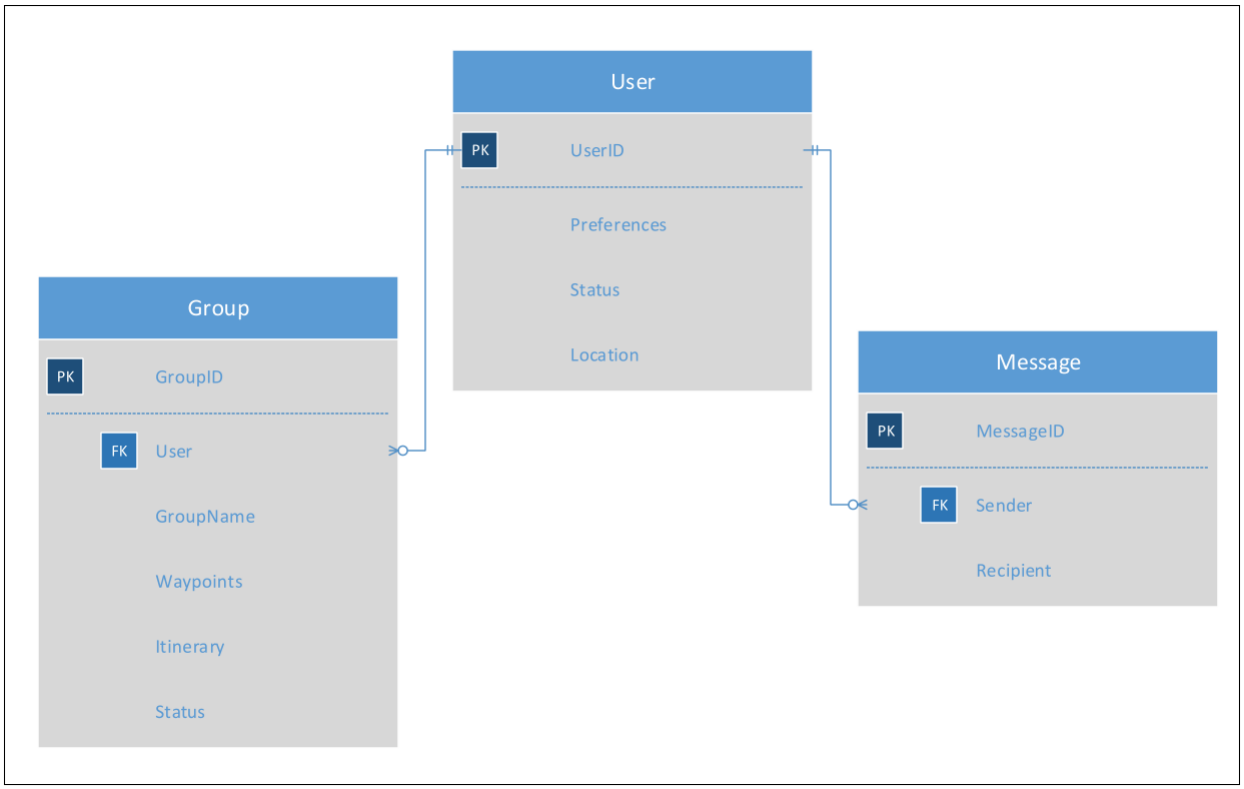


Figure 4.1: Early database schema.

4.1.2 Data Structures and Algorithms

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

4.1.3 Communications

One of the core features to the Crowd Control app is communication amongst its users. To achieve this, some third-party services are used, which in turn communicate data between users. If a user wishes to send another user a message, that message is sent to the Parse back-end, and delivered to the recipient via the Sinch service. The basic communication overview is outlined below in Figure 6.6.

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

4.1.4 Classes

4.1.4.a Sign up and Log in Classes

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

The SignupActivity.java is next on the list. It allows a user to sign into parse, it was created using the Android log-in/sign-up template. From here a user can access a Facebook log-in page and a Twitter log-in page. Additionally the class, LoginActivity.java allows the user to log into a registered email account.

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

Figure 4.2: Improved database schema.

4.1.4.b Menu Classes

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

The classes connected to the menu include: SettingsActivity.java, NotificationActivity.java and InviteNaviga-

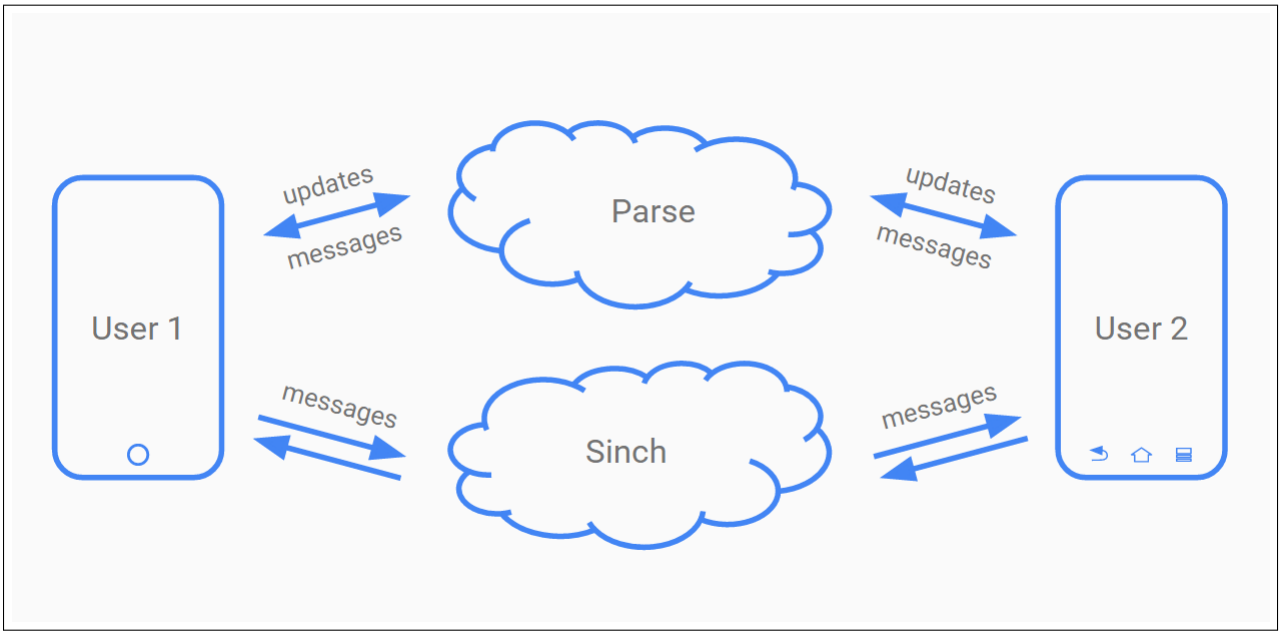


Figure 4.3: Communication flow diagram.

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

4.1.4.c Initial Tab Class

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

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

4.1.4.d Group Management Classes

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

Other group management classes can be accessed though the menu for group leaders. These classes include: InviteNavigationActivity.java, InviteConfirmFragment.java, and InviteSearchFragment.java. The InviteNavicationgActivity.java class is much like the GroupNavigationActivity.java class. It is a tab system holding the other two classes. The InviteSearchFragment.java allows the leader to search for other users they wish to add to the group. It creates a group list that is given to the InviteConfirmFragment.java class. Here the leader can make sure that these are the people that he/she want to invite and completes the process.

4.1.4.e Location Classes

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

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

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

4.1.4.f Messaging Classes

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

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

4.1.4.g Miscellaneous Classes

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

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

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

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

4.1.5 UML

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

4.1.6 GUI

Designing the user interface of a mobile application has many unique challenges, all of which influence design decisions. In order to present users with a satisfying and intuitive experience, it is suggested to follow material design pattern aspects such as using Floating Action Buttons (FABs), and simply adhering to a more unified experience that can be found across multiple new applications on many platforms. With that in mind, we also focused on keeping the overall design very clean and free from clutter.

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

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

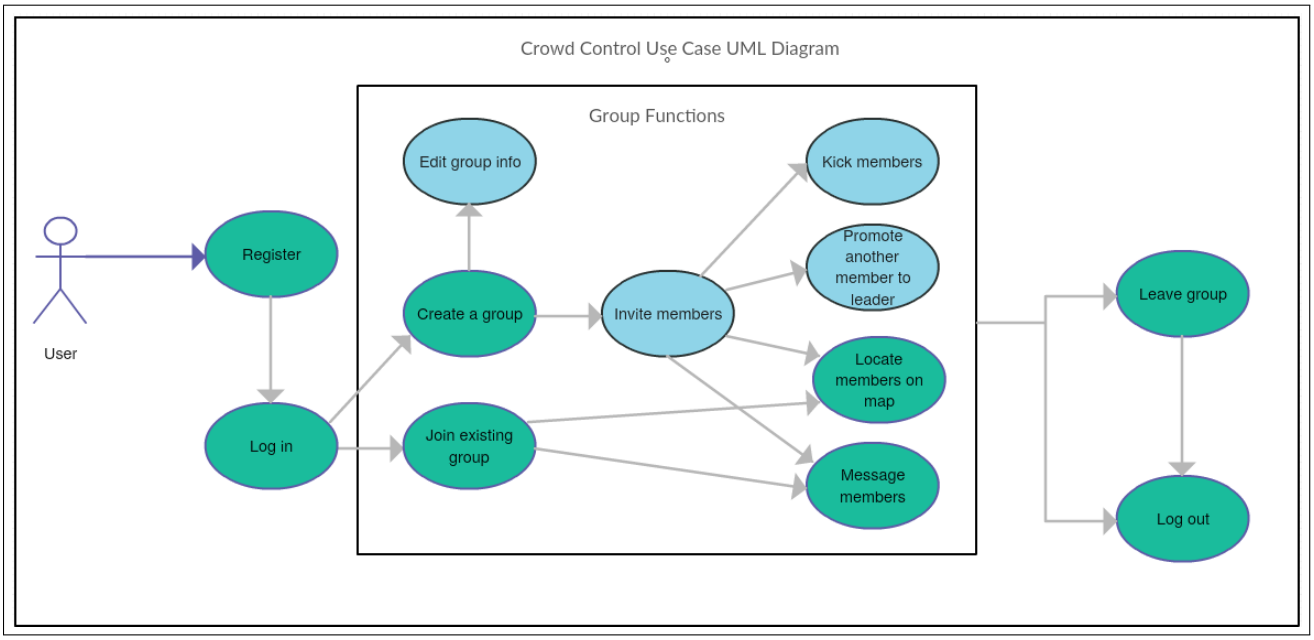


Figure 4.4: Use Case Diagram in UML.

4.1.7 MVC

4.1.7.a Introduction

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

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

4.1.7.b Models

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

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

4.1.7.c Controllers

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

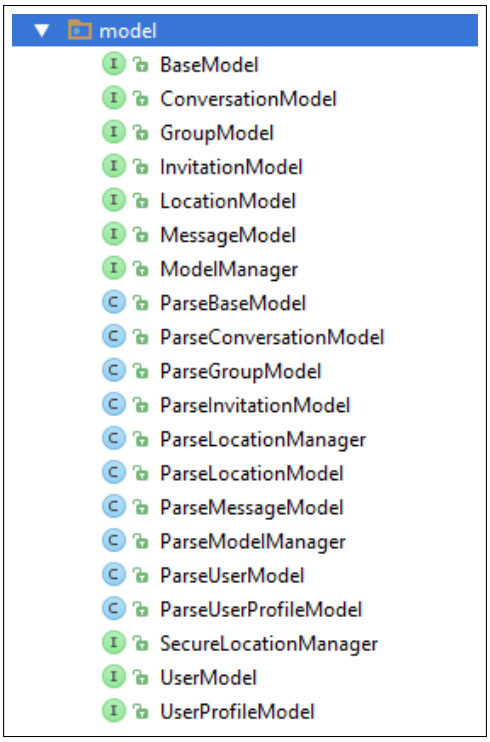


Figure 4.5: Model classes in the "model" folder of the project.

4.1.7.d Views

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

4.2 Group Messaging

4.2.1 Technologies Used

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

4.2.2 Component Overview

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

Phase 1 Construct message containers and begin Sinch integration

Phase 2 Construct messaging activity and view to encapsulate functionality

Phase 3 Construct conversation containers to associate groups of messages

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

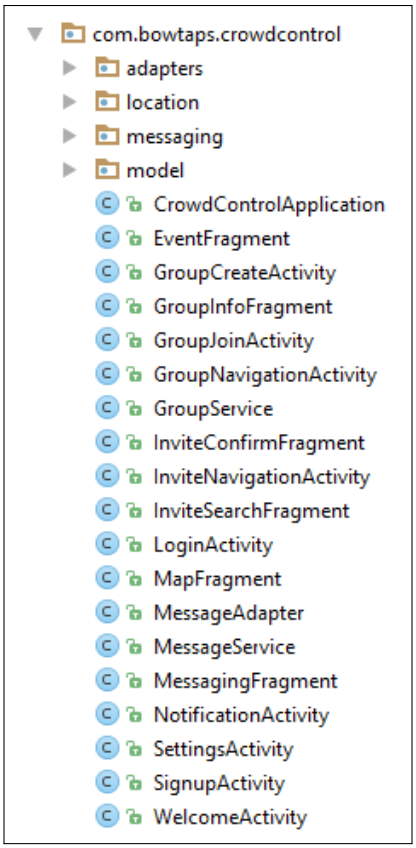


Figure 4.6: Model classes in the "model" folder of the project.

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

4.2.3 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.

4.2.4 Data Flow Diagram

4.2.5 Design Details

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

4.3 Location Tracking

4.3.1 Technologies Used

- Google Play Services / Apple Map Features

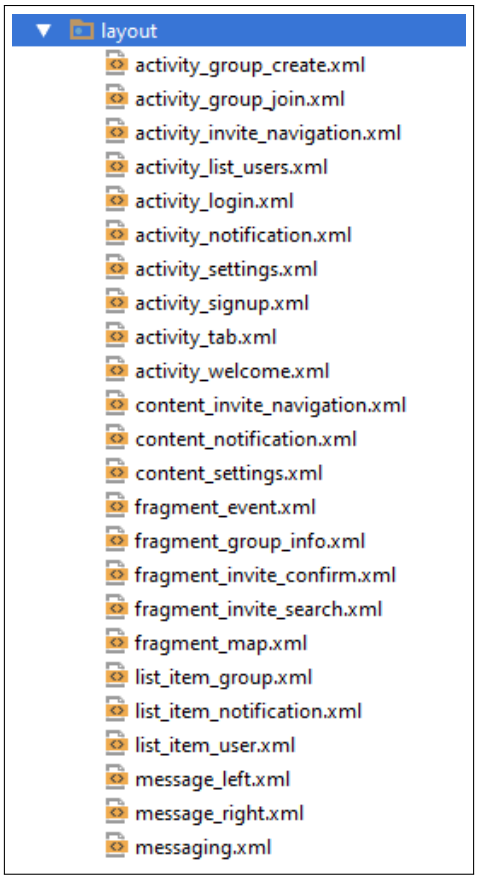


Figure 4.7: Model classes in the "model" folder of the project.

- Parse

4.3.2 Component Overview

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

4.3.3 Phase Overview

Phase 1 Construct location containers and begin location services integration

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

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

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

4.3.4 Architecture Diagram

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

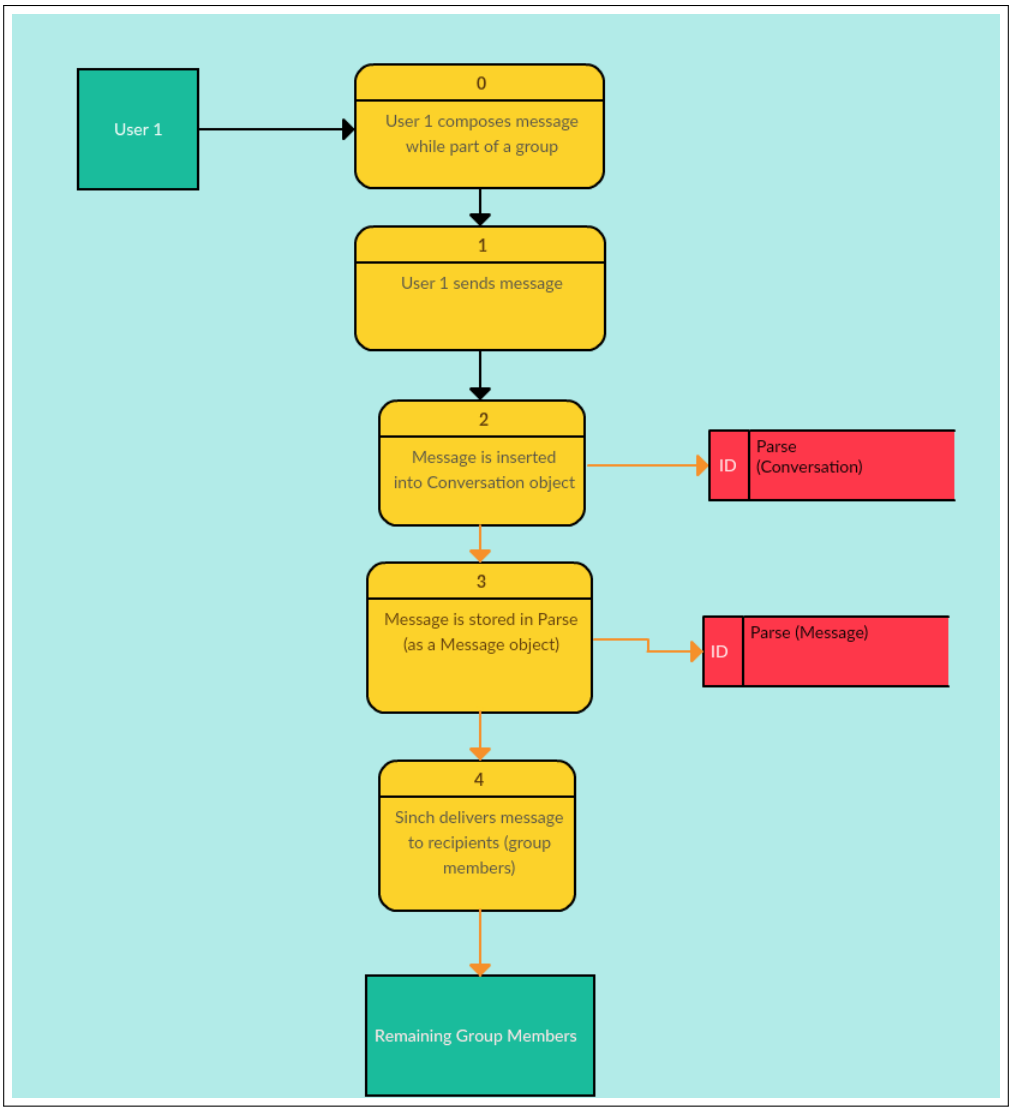


Figure 4.8: Messaging data flow diagram.

4.3.5 Data Flow Diagram

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

4.3.6 Design Details

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

4.4 Group Management

4.4.1 Technologies Used

- Parse

4.4.2 Component Overview

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

4.4.3 Phase Overview

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

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

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

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

4.4.4 Architecture Diagram

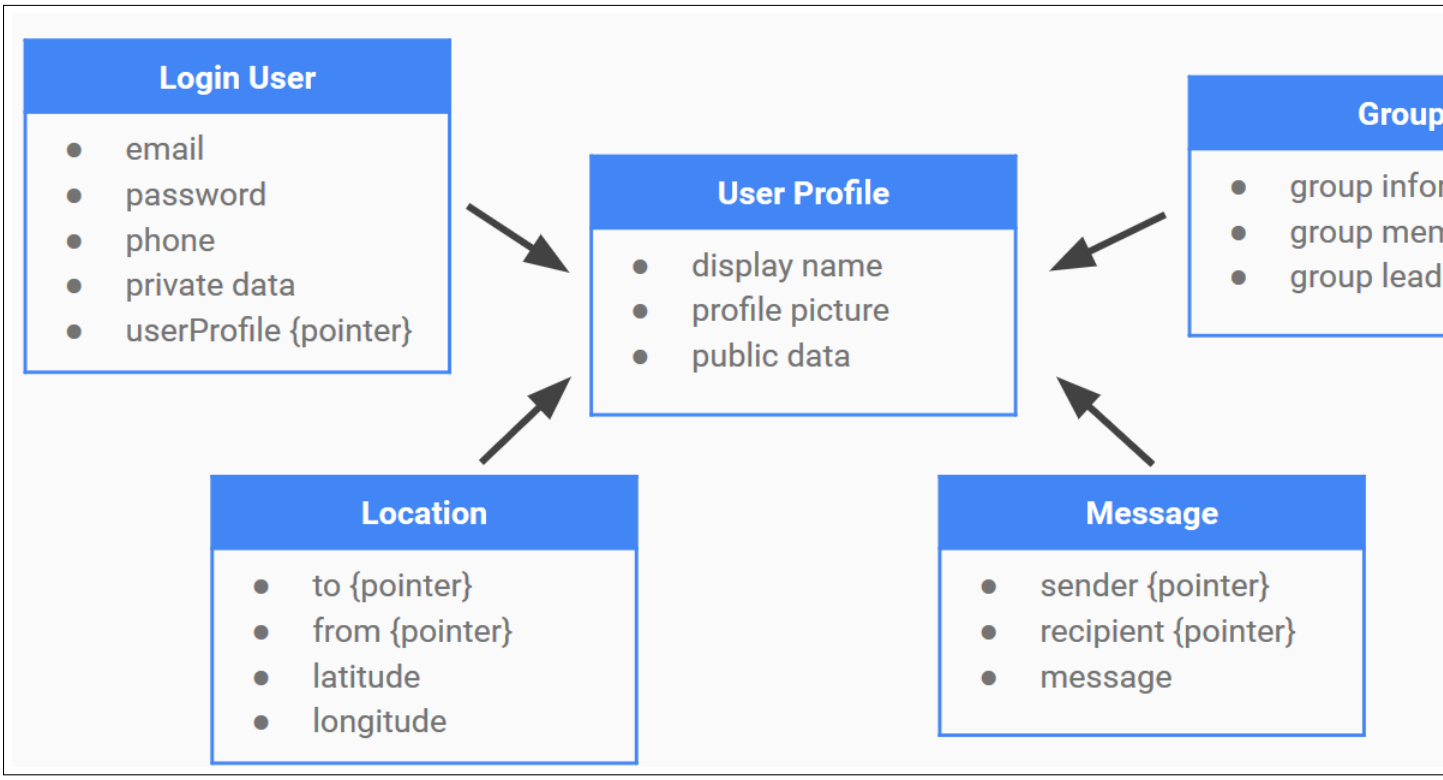


Figure 4.9: Architecutre of Group

4.4.5 Data Flow Diagram

See Figure 4.9

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 sevices 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 identifing 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 hinderance 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 was 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 **??**. 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 efficency. 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 the 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 irreversable destruction of data that we cannot afford to jeapordize. 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 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.

Test case	Result
Log in with email	
Valid credentials	Success
Invalid email	Failure
Invalid password	Failure
Log in with Facebook	
Valid credentials	Success
Invalid credentials	Failure
Log in with Facebook	
Valid credentials	Success
Invalid credentials	Failure

Sign up with email	
Email already in use	Message
Valid credentials	Success
Invalid password	Failure
Unconfirmed password	Failure
Invalid email	Failure + Message
Whitespace-only username	Failure + Message
Empty username	Failure + Message
Empty email	Failure + Message
Empty password	Failure + Message

Create a group	
No name	Message
Whitespace-only name	Failure + Message
No description	Success + Message
Whitespace-only description	Success + Message
Else	Success + trim whitespace + collapse whitepace in name

Search for users	
Enter display name	Display list of matching users or "no matches"
Enter email	Detect email, no change
Enter phone number	Detect phone number, no change
Enter incomplete phone number or incomplete email	Display "no matches"

Send invite to users to join group	
By email unassociated with user	Generate and send email
By email associated with user	Send app notification to user, display invite in-app
By phone number unassociated with user	Detect phone number, no change
By phone number associated with user	Send app notification to user, display invite in-app

Join a public group	
Tap "join group"	Displays confirmation message (y/n)
Accept confirmation message	Message "request sent"
Decline confirmation message	Dismiss message
Join group via invite	
Tap "join"	Join group
Tap "delete"	Delete invitation
Leave a group	
Tap "Leave Group"	Display confirmation message (y/n)
Accept confirmation message	If leader, display confirmation to choose new leader (y/n), else leave the current group and return to nearby groups
Leader accepts confirmation	Display list of group members
Leader declines confirmation	Leave current group and return to nearby group list
Decline confirmation message	Dismiss message, do not leave group
Group leader disbands group	
Leader taps "disband group"	Display confirmation message (y/n)
Accept confirmation	Display message to all users and return to join group page
Decline confirmation	Dismiss message, do not leave group
Enter display name	
Empty string	Highlight, no accept
Too short	Highlight, no accept
All whitespace	Highlight, no accept
Valid display name	Accept, trim whitespace, collapse consecutive whitespace
Sign out of app	
Tap "log out"	Display confirmation message (y/n)
Accept confirmation	Log out of the application and display login page
Decline confirmation	Dismiss message, do not change application state
Send a group message	
Emtpy string	Send button disabled
Non-empty string	Message sent to all members and immediately displayed in conversation + saved to storage

See locations of group members	
Tap on "map" tab	Map is displayed, group member locations displayed on map as pins (zoombed out to display all group members)

Opt-out location updates	
Toggle "enable location updates" off	Display toast/notify message
Toggle "enable location updates" on	Display confirmation message (y/n)
Accept confirmation	Share user location, begin auto location share
Decline confirmation	Toggle "enable location updates" off

Receive message	
Received message	Unread indicator in tab navigation bar

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 overal 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 endevaors. It is unfortunately a side affect 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 knowledgable 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 effots 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 ehavily 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 otherewise 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 marget, 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. Details on each of these are listed below, along with a summary on the changes made to our backlog throughout the course of this project. This section covers successes, issues, and problems for the entire project, including aspects that are not directly related to testing.

5.7.1 Successes in Client Application

The core deliverable of this project has been the iOS and Android client applications. These apps were the main focus of our design and development and are the product that end users will interact with directly. Thus, it was important to ensure that care and attention was devoted to this aspect of the project.

During this project, we made great progress in creating functional versions of the client application for both the iOS and Android mobile phone platforms. In both versions we successfully incorporated the following features:

- Create new account
- Log in to existing account using email address
- View list of joinable groups
- "Pull-to-refresh" list of joinable groups
- View information on a particualr group
- Join an existing group
- Create a new group
- View own location on a dynamic map
- View information on the current group
- Leave the current group
- Sign out of the application

During the later half of this project, the team shifted focus towards developing exclusively for the Android version of the client, as all team members already posessed the required tools to work on it. As a result, the Android version of our client application has an additional set of features compared to that of the iOS version. The following features exist in the Android version at the time of this writing:

- View dedicated settings screen
- Change user display name
- View other group members' locations on dynamic map
- View list of other members on group info screen
- Send instant messages to other group members
- Receive instant members from other group members
- Group leaders can choose to disband the group
- Group leaders can promote other members to replace them as the group leader
- Group leaders can remove other members from the group
- Users can view pending invitations to join existing groups
- Clients automatically fetch group changes periodically
- Clients automatically broadcast own location to other group members

Because one of the original deliverables of this project was both iOS and Android client applications, this fragmentation of features is less-than-ideal, yet ultimately necessary in order to produce at least one functional prototype before the end of this course. Over the summer, we plan to continue development of these features. We will also be dividing focus between both versions of the client so that the iOS version will be caught up to the progress we have made in the Android version.

5.7.2 Successes in Cloud Code

Our project encompasses both client-side and server-side development. To produce server functionality, we developed cloud code to run on our Parse data server. Much of the logic that requires chained queries into the database, cross references, and multi-row accesses we implemented in cloud code. By moving this logic to the server, we are able to serve complicated requests while minimizing network access and increasing effiency and data integrity across threads.

5.7.3 Successes in Testing

The greatest success of our work in unit and system tesing has been in the development of our testing matrix. Our tesing 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 fifty 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 tesing 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. Additionaly, 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 UI unit tests for the iOS version 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. For example, many interface tests relate to the login and signup screens and are used for tesing account creation and login.

5.7.4 Issues in Dependencies

Midway through this project, one of our core dependencies, Parse, which is a third party service that acts as a database backend and cloud code host, announced that it would be discontinuing service effective January 31st, 2017. This means that before the end of the year, our project's backend will need to be transferred to another service of similar functionality. However, because Parse will not be discontinuing service until after the end of 2016, we have prioritized feature development over mitigating this risk, as there is still time to address this issue.

5.7.5 Issues in Testing

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 couse and guest lectures, none have designed or implemented automated tesing 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 tesing, 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 concious decision to prioritize our efforts, we agreed that code separability and maintainability should not be neglected. These two tennants of software development are heavilty promoted by automated tesing. By choosing to prioritize separability and maintainability when designing our software, we are confident that delaying automated tesing 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.

5.7.6 Problems in Testing

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

limited control over.

Xcode, the IDE chosen for iOS development underwent an update that apeared to cause UI tesing 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 tesing. 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.

5.7.7 Changes to the Backlog

6

Prototypes

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

6.1 Sprint 1 Prototype

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

6.1.1 Deliverable

- UX for Android
- UX for iOS

6.1.2 Backlog

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

6.1.3 Success/Fail

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

6.2 Sprint 2 Prototype

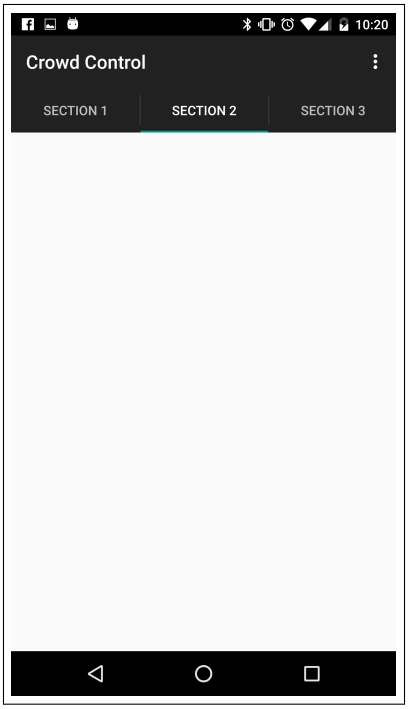


Figure 6.1: Sprint 2 Prototypes.

6.2.1 Deliverable

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

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

6.2.2 Backlog

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

6.2.3 Success/Fail

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

6.3 Sprint 3 Prototype

6.3.1 Deliverable

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

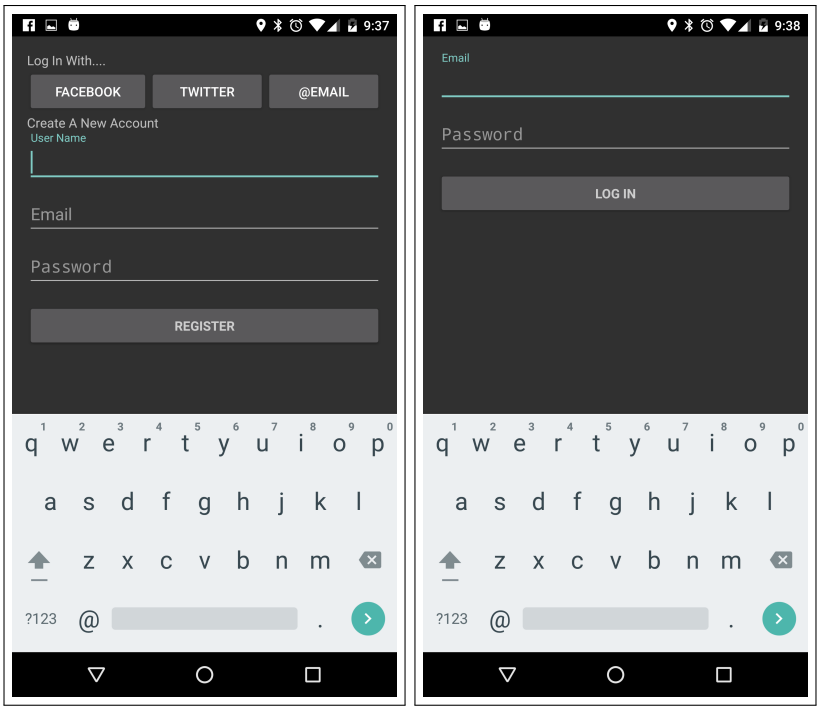


Figure 6.2: Sprint 3 Prototypes.

6.3.2 Backlog

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

6.3.3 Success/Fail

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

6.4 Winter Sprint Prototype

6.4.1 Deliverable

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

6.4.2 Backlog

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

6.4.3 Success/Fail

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

6.5 Sprint 4 Prototype

6.5.1 Deliverable

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

6.5.2 Backlog

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

6.5.3 Success/Fail

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

6.6 Sprint 5 Prototype

6.6.1 Deliverable

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

6.6.2 Backlog

- Senior Design Doc
  - Do a general revision of the doc
  - Business Plan
    - Finish business plan for 2016 Governor's Giant Vision Competition
- Android
  - Model Caching/ Uniformity
  - Clean up appearance
  - Clean up the appearance of the app
  - Display Group Members on group info page
  - Safe group operations(leaving/joining group)
  - Loading animations on homing and syncing
  - Integration Testing
  - Start Alpha Testing
- Cloud code
  - Safe group operations(leaving/joining group)
  - test join and leave functionality

6.6.3 Success/Fail

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

6.7 Sprint 6 Prototype

6.7.1 Deliverable

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

6.7.2 Backlog

Week 1

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

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

Week 2

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

Week 3

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

6.7.3 Success/Fail

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

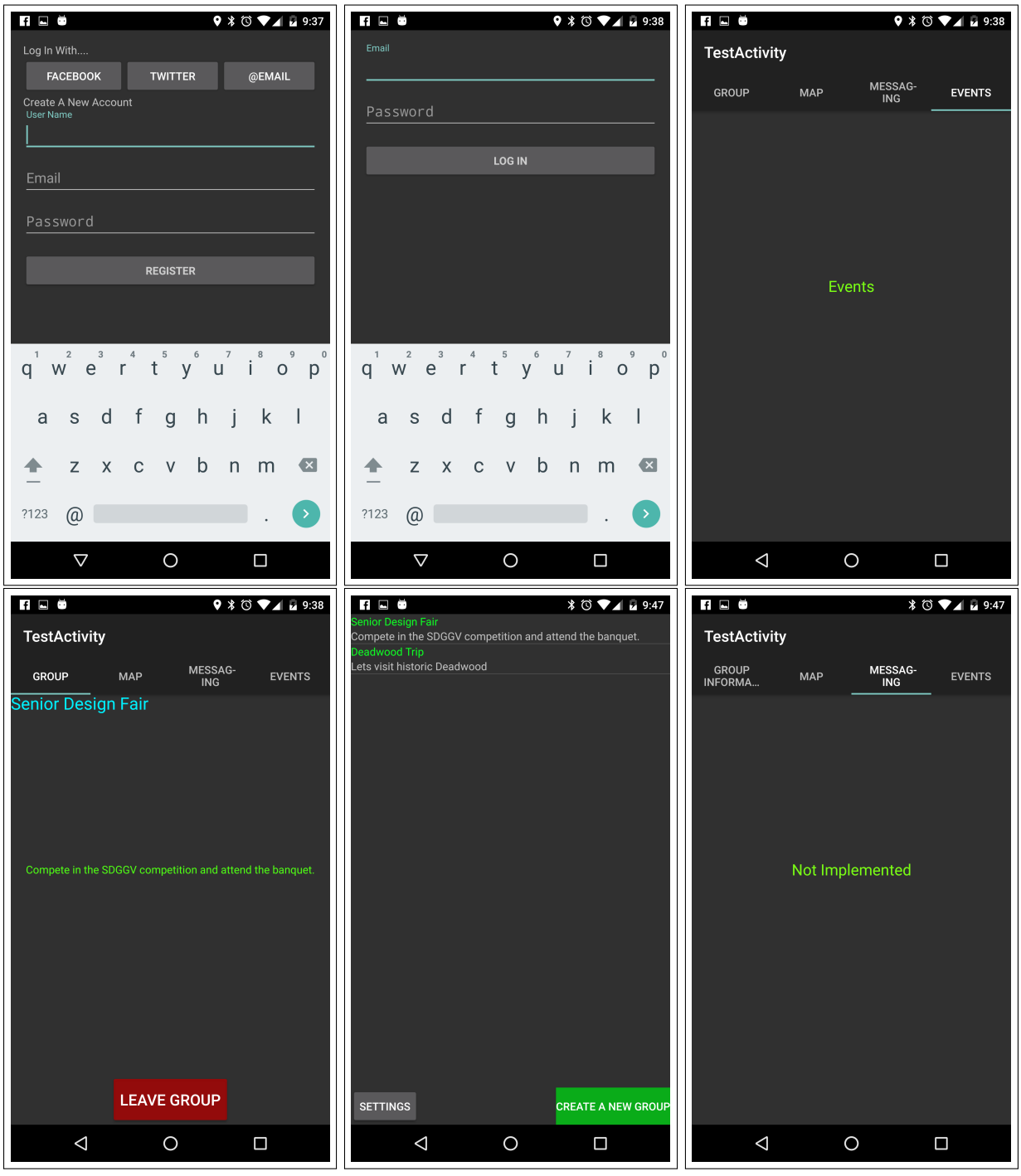


Figure 6.3: Winter Sprint Prototypes.

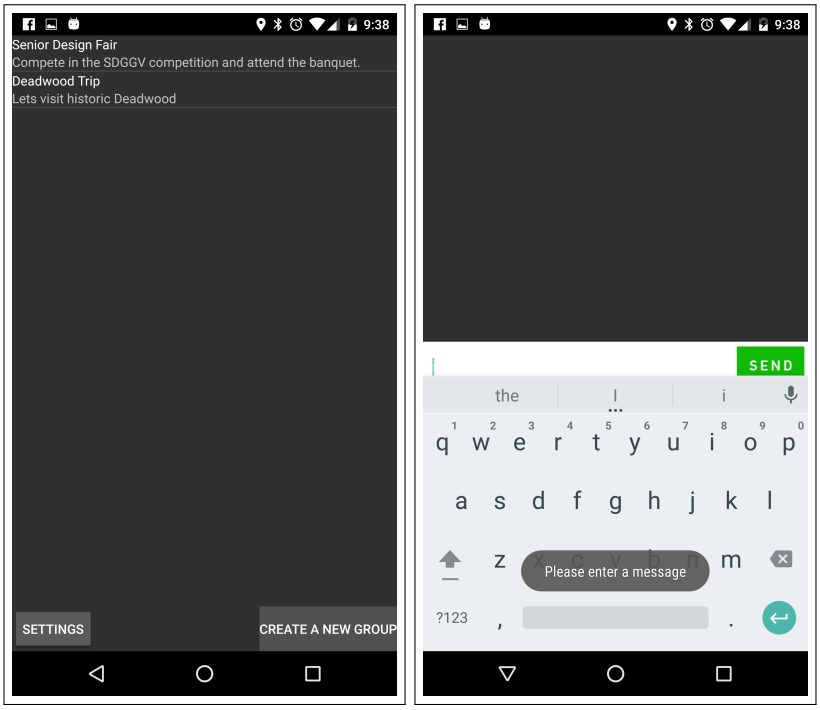


Figure 6.4: Sprint 4 Prototypes.

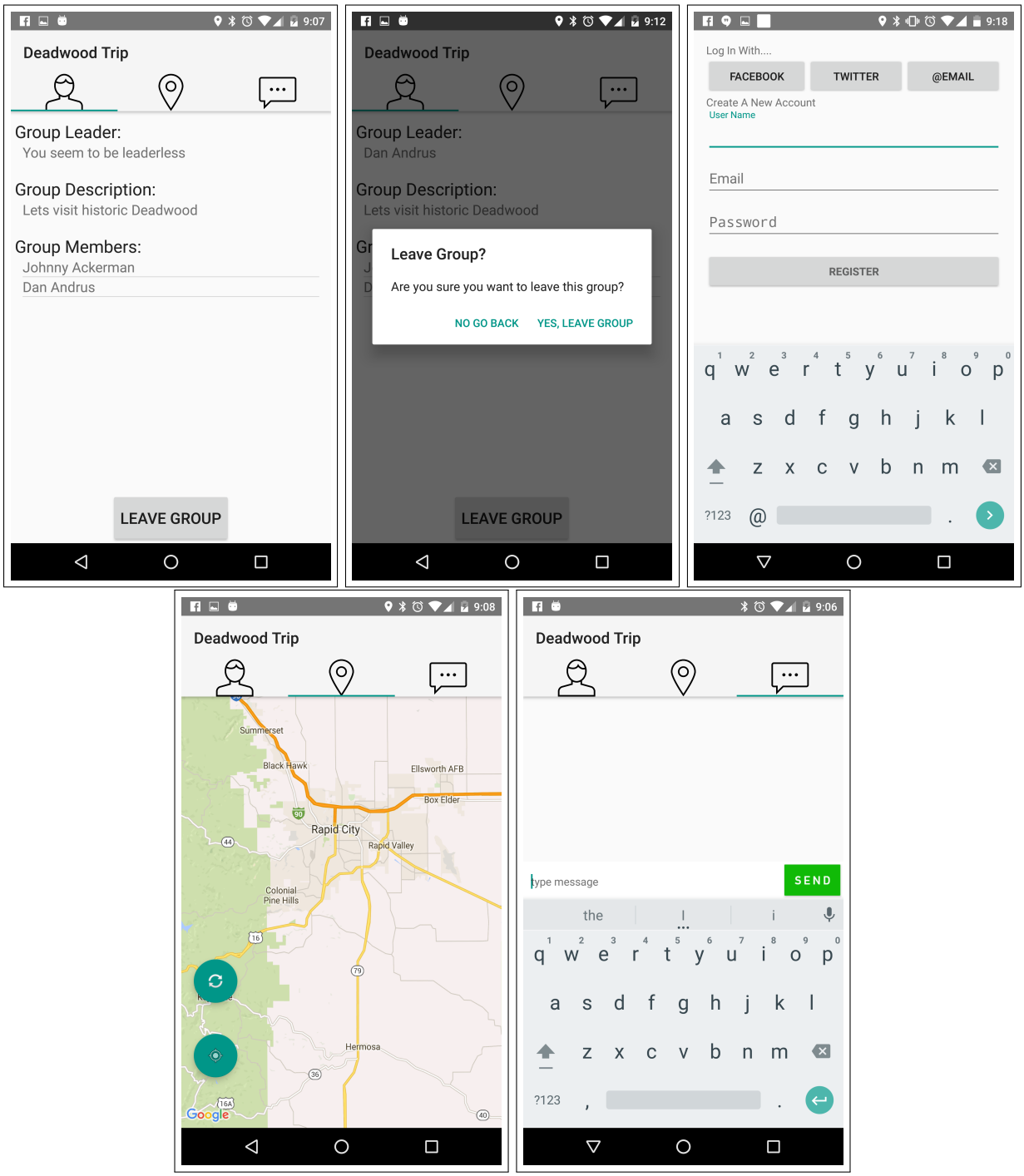
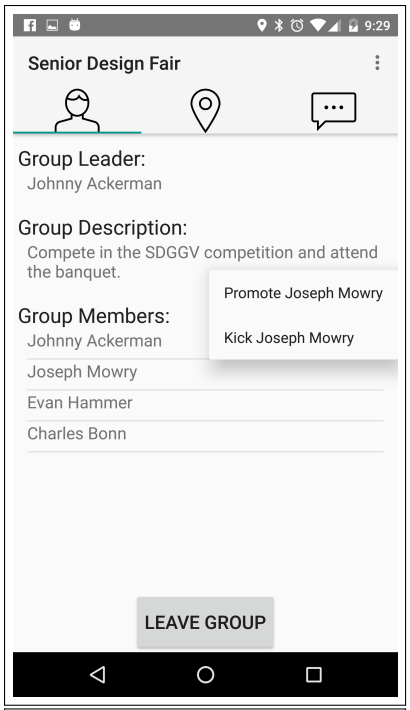
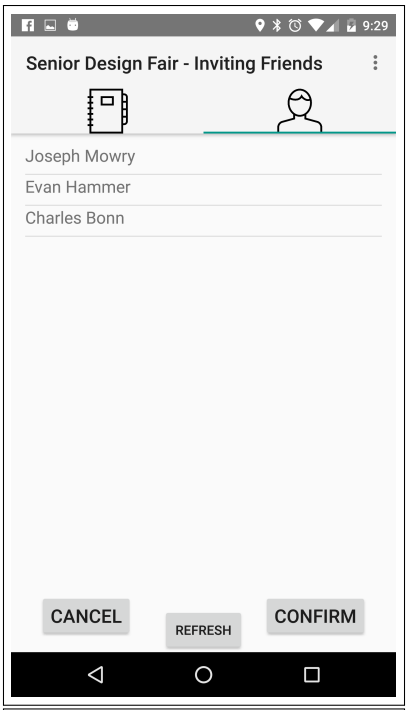
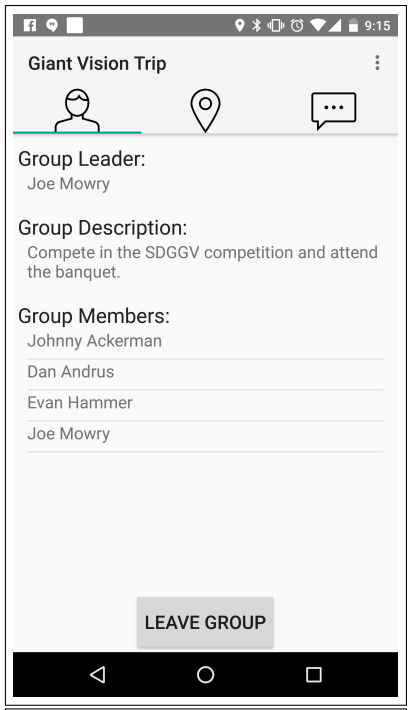
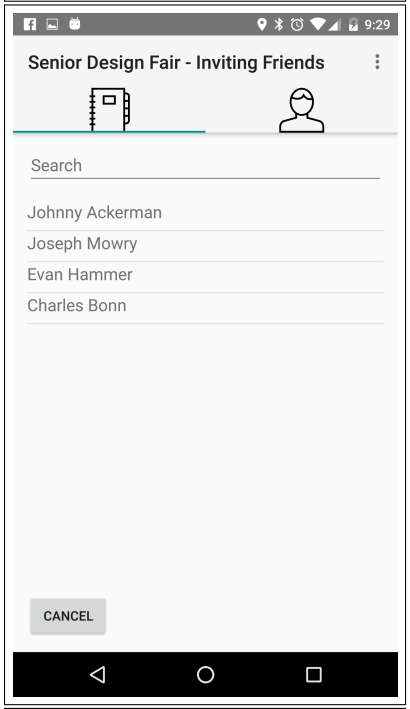


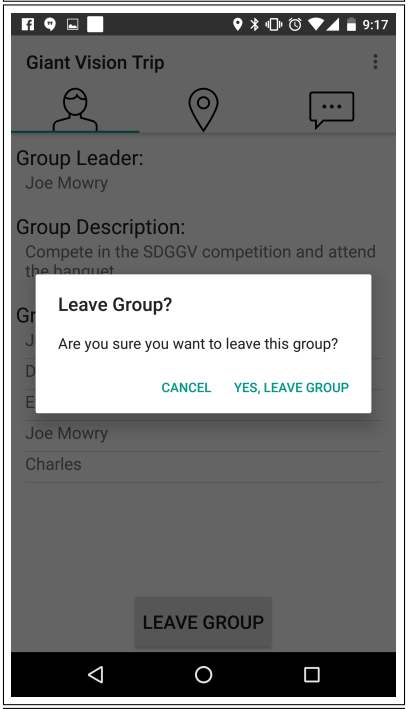
Figure 6.5: Sprint 5 Prototypes.

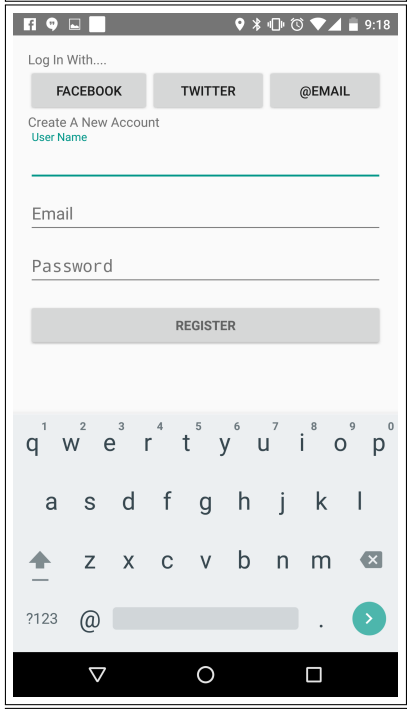

Senior Design Fair
Group Leader:
Johnny Ackerman
Group Description:
Compete in the SDGGV competition and attend the banquet.
Promote Joseph Mowry
Kick Joseph Mowry
Group Members:
Johnny Ackerman
Joseph Mowry
Evan Hammer
Charles Bonn
LEAVE GROUP

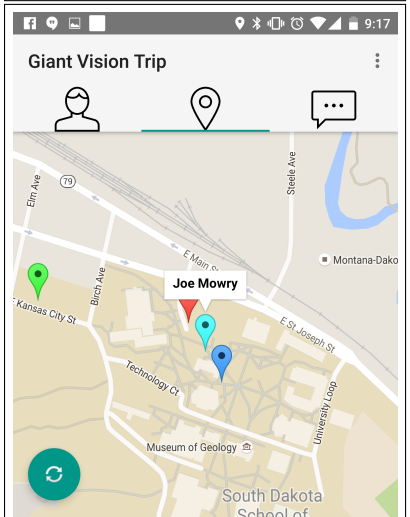

Senior Design Fair - Inviting Friends
Joseph Mowry
Evan Hammer
Charles Bonn
CANCEL
REFRESH
CONFIRM

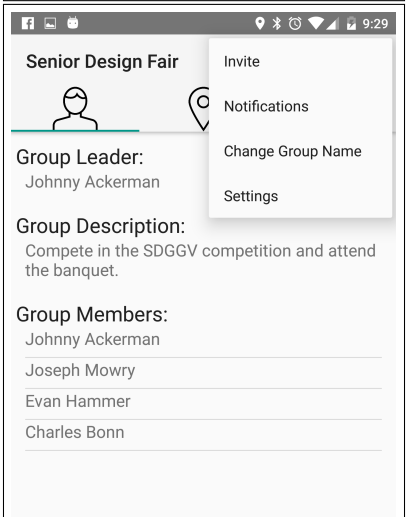

Giant Vision Trip
Group Leader:
Joe Mowry
Group Description:
Compete in the SDGGV competition and attend the banquet.
Group Members:
Johnny Ackerman
Dan Andrus
Evan Hammer
Joe Mowry
LEAVE GROUP

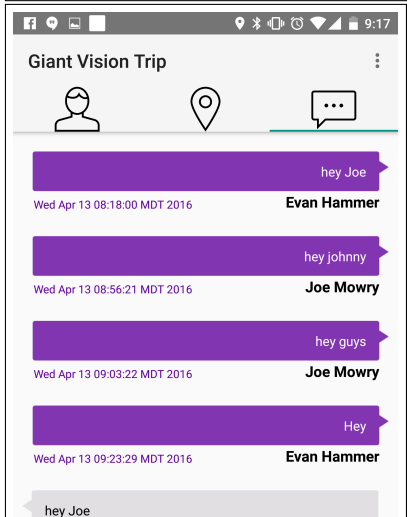

Senior Design Fair - Inviting Friends
Search
Johnny Ackerman
Joseph Mowry
Evan Hammer
Charles Bonn
CANCEL


Giant Vision Trip
Group Leader:
Joe Mowry
Group Description:
Compete in the SDGGV competition and attend the banquet.
Leave Group?
Are you sure you want to leave this group?
CANCEL
YES, LEAVE GROUP
Joe Mowry
Charles
LEAVE GROUP


Log In With....
FACEBOOK
TWITTER
@EMAIL
Create A New Account
User Name
Email
Password
REGISTER


Giant Vision Trip
Joe Mowry
Museum of Geology
South Dakota


Senior Design Fair
Invite
Notifications
Change Group Name
Settings
Group Leader:
Johnny Ackerman
Group Description:
Compete in the SDGGV competition and attend the banquet.
Group Members:
Johnny Ackerman
Joseph Mowry
Evan Hammer
Charles Bonn


Giant Vision Trip
hey Joe
Wed Apr 13 08:18:00 MDT 2016
Evan Hammer
hey johnny
Wed Apr 13 08:56:21 MDT 2016
Joe Mowry
hey guys
Wed Apr 13 09:03:22 MDT 2016
Joe Mowry
Hey
Wed Apr 13 09:23:29 MDT 2016
Evan Hammer
hey Joe

7

Release – Setup – Deployment

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

7.1 Deployment Information and Dependencies

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

7.2 Setup Information

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

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

7.3 System Versioning Information

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

Current Version [0.3.0]

Prepared By:
Daniel Andrus
Evan Hammer
iOS App Version History

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

Current Version [0.6.0]

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

Android App Version History

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

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

8

User Documentation

8.1 User Guide

8.1.1 Registering a new User

When you First open the app you will be required to register a user or you can login via Facebook or twitter. To create a Crowd Control account fill out the required page and click register.

8.1.2 Login

8.1.2.a Crowd Control User

From the main page click the email button. This will allow you to login in via a Crowd Control account.

8.1.2.b Facebook Login

From the main page click the Facebook button. This will allow you to login via a Facebook account that will link to a Crowd Control user account.

8.1.2.c Twitter Login

From the main page click the twitter button. This will allow you to login via a twitter account that will link to a Crowd Control user account..

8.1.3 Creating a Group

From the main page:

1. Click "CREATE A NEW GROUP" Button at the bottom of the page.
2. Fill out Group Name and Group Description
3. Click "create group"

8.1.4 Joining a Group

8.1.4.a From Main List

1. Find the group from the list that you wish to join.
2. Click on the group to join.

8.1.4.b From Invite

1. Go to notifications.
2. Click on the invitation to join the group that invited you.

8.1.5 Leaving a Group

Click the leave group button at the bottom of the main group page.

8.1.6 Group Main Page

This page shows the main information of the group including:

- Group Name.
- Description of the Group.
- List of Group Members.
- Leave Button

8.1.7 Map Page

This page displays a map with pins of the other members in your group

8.1.7.a Sync Map

The sync map button will sync the map with the current locations of other members if your group. This feature is for if your map becomes desynced.

8.1.7.b Center Map

Clicking on the center map button will center the map on your location.

8.1.8 Messaging Page

This is the messaging feature. It allows you to send messages to the rest of the group. To bring up the key board click the text feild at the bottom of the page.

8.1.9 Options Menu

8.1.9.a Notifications

1. Opens the notifications page.

8.1.9.b Settings

1. Opens the settings page for CrowdControl.

8.1.10 Group Leader Options

8.1.10.a Invite

1. Opens the Invite page

8.1.10.b Notifications

1. Opens the notifications page.

8.1.10.c Change Group Name

1. Opens a window to change the name of the group

8.1.10.d Settings

1. Opens the settings page for CrowdControl.

8.1.11 Inviting Group Members

From the invite page:

1. Select users from the list of users, or you can search for there user name using the search bar.
2. When you have selected the users you would like to invite click on the next tab for the comfermation screen.
3. If these are the people you would like to invite click confirm and invites will be sent to the user.

8.1.12 Settings

8.1.12.a Changing Profile Name

1. To change your profile name click the text feild and change the name to the name you wish to be displayed to the rest of the group.
2. click change name.

8.1.12.b Search Distance

Adjust the search radius for locating a group to join.

8.1.12.c Current Group

8.1.12.d Logout

8.1.12.e Finished

This will bring you back to the home page or the group main page if you are in a group.

8.2 Installation Guide

8.2.1 Android Install

8.2.1.a AppStore

Android Appstore version is not currently avalable.

8.2.1.b Android Studio

Set up your Device

1. Plug in your device to your development machine with a USB cable. If you're developing on Windows, you might need to install the appropriate USB driver for your device.
2. Enable USB debugging on your device. On Android 4.0 and newer, go to Settings > Developer options.

Run the app from Android Studio

1. Select one of your project's files and click Run from the toolbar.

2. In the Choose Device window that appears, select the Choose a running device radio button, select your device, and click OK .

Android Studio installs the app on your connected device and starts it.

Taken from http://developer.android.com.

8.2.2 iOS

8.2.2.a AppStore

Apple Appstore version is not currently avalable.

8.2.2.b XCode

Launching Your App on a Device

It takes just a few steps to launch your app on a device if you previously created your code signing identity and team provisioning profile, as described in Creating the Team Provisioning Profile. Otherwise, a series of dialogs and warnings may appear as Xcode resolves the code signing issues in the process of launching your app.

To launch an app on a device

1. Connect the device to your Mac.

2. In the project navigator, choose your device from the Scheme toolbar menu.
   Xcode assumes you intend to use the selected device for development and automatically registers it for you.

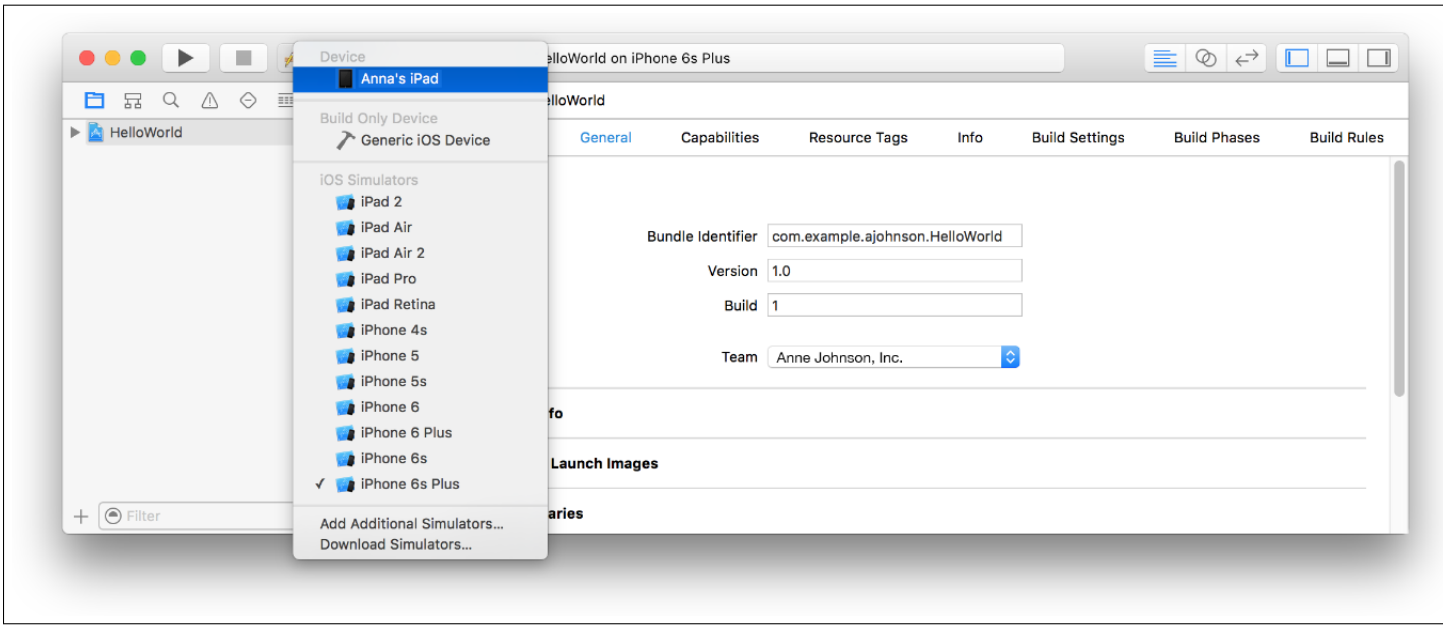


Figure 8.1: iOS Device selection

If your device is disabled in the Scheme toolbar menu because it is ineligible, fix the issue before continuing. Under Ineligible Devices, hover the mouse over the device to read the reason why it's ineligible. For example, if the OS version is lower than the deployment target, upgrade the OS version on the device or choose the version you want to target from the Deployment Target pop-up menu (located in the Deployment Info section). Then select the device from the Scheme toolbar menu.

3. If a prompt appears asking whether codesign can sign the app using a key in your keychain, click Always Allow.

4. Click the Run button.

   Xcode installs the app on the device before launching the app.

Taken from https://developer.apple.com.

8.3 Programmer Manual

8.3.1 Api key changes

The API keys are located in

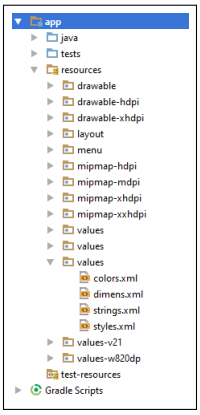


Figure 8.2: Android Api Location

This Is The file where the API keys are held.

8.3.2 Global Strings

Global strings are located in the same spot that the API keys are located. Theses strings are able to change to implement globalization for expansion to other languages.

8.3.3 Crash handeling

Crashalitics is integrated and handles and gives email reports of crashes. Crashalitics will give information such as number of crashes, time, file location and file line.

8.3.4 Database

Parse handles the database. Values here should not be changed or removed. Tables are able to be added as Crowd Control expands.

9

Class Index

9.1 Class List

9.1.1 Andriod

9.1.2 iOS

9.1.3 Cloud Code

10

Class Documentation

10.1 ApplicationTest Class Reference

Public Member Functions

- ApplicationTest ()
- ∼ApplicationTest ()

10.1.1 Constructor & Destructor Documentation

10.1.1.a ApplicationTest::ApplicationTest ( )

Applicationtest Constructor

10.1.1.b ApplicationTest::∼ApplicationTest ( )

ApplicationTest Deconstructor

10.1.2 Method Summary

10.1.2.a Methods inherited from class android.test.Application TestCase ( )

getApplication, getSystemContext, testApplicationTestCaseSetUpProperly

10.1.2.b Methods inherited from class android.test.AndroidTestCase ( )

assertActivityRequiresPermission, assertReadingContentUriRequiresPermission, assertWritingContentUriRequires-Permission, getContext, getTestContext, setContext, setTestContext, testAndroidTestCaseSetupProperly

10.1.2.c Methods inherited from class java.lang.Object ( )

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

10.1.3 Generation

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

- ApplicationTest

10.2 BaseModel Class Reference

Interface BaseModel

public interface ***BaseModel***

The base model interface, providing access to core model functionality, including: - unique object identifier - the initial object creation - the last updated date - model saving and loading.

All Known Subinterfaces:
GroupModel, UserModel, UserProfileModel

All Known Implementing Classes:
ParseBaseModel, ParseGroupModel, ParseUserModel, ParseUserProfileModel

Member Functions

- getCreated ()
- getId ()
- getUpdated ()
- load ()
- loadInBackground ()
- save ()
- saveInBackground ()
- wasModified ()

10.2.1 Method Detail

10.2.1.a getCreated ( )

java.lang.Date getCreated()

Gets the creation date and time for the model. This value is automatically generated and assigned when the object is first added to storage.

Modifier and Type:
java.util.Date

Returns:
Date object representing the date and time when the model was first introduced into storage.

10.2.1.b getId ( )

java.lang.String getId()

Gets the unique object identifier for the model. This value is usually determined by the storage medium, such as the database where the model is stored.

Modifier and Type:
java.util.String

Returns:
String representation of the unique identifier that can be used to reference this object.

10.2.1.c getUpdated ( )

java.lang.Date getId()

Gets the date and time that the model was last updated in storage. This timestamp is automatically assigned by storage and represents when the storage-side model was last changed.

Modifier and Type:
java.util.Date

Returns:
Date object representing the date and time when the model was last updated in storage.

10.2.1.d load ( )

void getId()

Loads this object from storage. This is a blocking function, so care should be taken to not call this function on the main thread.

Modifier and Type:
void

Throws:
java.lang.Exception

10.2.1.e loadInBackground (BaseModel.LoadCallback *callback*)

void loadInBackground(BaseModel.LoadCallback callback)

Loads this object from storage asynchronously. Spawns a separate thread so that this function can be called from the main thread without blocking the UI. Upon completion, whether successful or unsuccessful, returns control to the main thread by calling the.

Modifier and Type:
void

Parameters:

- callback - The callback object to pass control to once the operation is completed. If no object is provided (or null is given), then nothing will happen after the object has been loaded.

10.2.1.f load ( )

void save()

Saves this object to storage. This is a blocking function, so care should be taken to not call this function on the main thread

Modifier and Type:
void

Throws:
java.lang.Exception

10.2.1.g loadInBackground (BaseModel.SaveCallback *callback*)

void loadInBackground(BaseModel.saveCallback callback)

Saves this object to storage asynchronously. Spawns a separate thread so that this function can be called from the main thread without blocking the UI. Upon completion, whether successful or unsuccessful, returns control to the main thread by calling the.

Modifier and Type:
void

Parameters:

- callback - The callback object to pass control to once the operation is complete. If no object is provided (or null is given), then nothing will happen after the object has been saved.

10.2.1.h wasModified ( )

void wasModified()

Gets a flag indicating that the model has new changes that haven't been saved in storage. This value will change whenever a property is set in the model but hasn't been saved.

Modifier and Type:
Boolean

Returns:
Boolean flag indicating whether or not the model contains unsaved changes.

10.2.2 Nested Class Summary

10.2.3 Generation

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

- BaseModel

10.3 BaseModel.LoadCallback Class Reference

Interface BaseMode.LoadCallback

public static interface ***BaseModel.LoadCallback***
The callback interface that should be used for asynchronous loading operations.

Enclosing interfaces:
BaseModel

Member Functions

- doneLoadingModel ()

10.3.1 Method Detail

10.3.1.a doneLoadingModel (BaseModel *object* java.lang.Exception *ex*)

void doneLoadingModel(BaseModel object, java.lang.Exception ex)

This method is called after completion of an asynchronous background load on a model. It accepts the loaded model and an exception object should something go wrong.

Modifier and Type:
void

Parameters:

- object - The object that attempted to be loaded. This parameter cannot be null and will be valid whether or not the operation was successful.
- ex - The exception that occurred, if any. This value will be null if the operation was successful and will be a valid exception object if an error occurred.

10.3.2 Generation

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

- BaseModel.LoadCallBack

10.4 BaseModel.SaveCallBack Class Reference

Interface BaseMode.SaveCallback

public static interface ***BaseModel.SaveCallback***
The callback interface that should be used for asynchronous loading operations.

Enclosing interfaces:
BaseModel

Member Functions

- doneLoadingModel ()

10.4.1 Method Summary

10.4.1.a doneSavingModel (BaseModel *object* java.lang.Exception *ex*)

void doneLoadingModel(BaseModel object, java.lang.Exception ex)

This method is called after completion of an asynchronous background save on a model. It accepts the saved model and an exception object should something go wrong.

Modifier and Type:
void

Parameters:

- object - The object that attempted to be saved. This parameter cannot be null and will be valid whether or not the operation was successful.
- ex - The exception that occurred, if any. This value will be null if the operation was successful and will be a valid exception object if an error occurred.

10.4.2 Generation

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

- BaseModel.SaveCallBack

10.5 BuildConfig Class Reference

Class BuildConfig

public final class ***BuildConfig***

extends:
java.lang.Object

Feild Summary

- APPLICATION_ID ()
- BUILD_TYPE ()
- DEBUG ()
- FLAVOR ()
- VERSION_CODE ()
- VERSION_NAME ()

10.5.1 Feild Detail

10.5.1.a APPLICATION_ID

public static final java.lang.String APPLICATION_ID

Modifier and Type:
public static final

10.5.1.b BUILD_TYPE

public static final java.lang.String BUILD_TYPE

Modifier and Type:
public static final

10.5.1.c DEBUG

public static final boolean DEBUG

Modifier and Type:
public static final

10.5.1.d FLAVOR

public static final java.lang.String FLAVOR

Modifier and Type:
public static final

10.5.1.e VERSION_CODE

public static final int VERSION_CODE

Modifier and Type:
public static final int

10.5.1.f VERSION_NAME

public static final java.lang.String VERSION_NAME

Modifier and Type:
public static final java.lang.String

10.5.2 Method Summary

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.5.3 Generation

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

- BuildConfig

10.6 CreateAccountActivity Class Reference

Class CreateAccountActivity

public class ***CreateAccountActivity*** extends android.support.v7.app.AppCompatActivity
implements android.view.View.OnClickListener

All Implemented Interfaces:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2,
android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,
android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator,
android.support.v4.app.TaskStackBuilder.SupportParentable,
android.support.v7.app.ActionBarDrawerToggle.DelegateProvider, android.support.v7.app.AppCompatCallback,
android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2,
android.view.View.OnClickListener, android.view.View.OnCreateContextMenuListener, android.view.Window.Callback,
android.view.Window.OnWindowDismissedCallback

Member Functions

- onClick ()
- onCreateOptionsMenu ()
- onCreateView ()
- onCreateView ()
- onOptionItemSelected ()

10.6.1 Method Summary

10.6.1.a onClick (android.view.View *view*)

public void onClick(android.view.View view)

Called when a view has been clicked.

Modifier and Type:
public void

Speciified by:
onClick in interface android.view.View.OnClickListener

Parameters':

- view - The view that was clicked.

10.6.1.b onCreateOptionsMenu (android.view.Menu *menu*)

public boolean onCreateOptionsMenu(android.view.Menu menu)
Initialize the contents of the Activity's standard options menu. You should place your menu items in to menu.

This is only called once, the first time the options menu is displayed. To update the menu every time it is displayed, see Activity.onPrepareOptionsMenu(android.view.Menu).

The default implementation populates the menu with standard system menu items. These are placed in the Menu.CATEGORY_SYSTEM group so that they will be correctly ordered with application-defined menu items. Deriving classes should always call through to the base implementation.

You can safely hold on to menu (and any items created from it), making modifications to it as desired, until the next time onCreateOptionsMenu() is called.

When you add items to the menu, you can implement the Activity's Activity.onOptionsItemSelected(android.view.MenuItem) method to handle them there.

Modifier and Type:
public boolean

Overrides:
onCreateOptionsMenu in class android.app.Activity
Parameters:

- menu - The options menu in which you place your items.

Returns:

- You must return true for the menu to be displayed; if you return false it will not be shown.

10.6.1.c onCreateView (android.view.View *parent* java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
public

Specified by:
onCreateView in interface android.view.LayoutInflater.Factory2

Overrides:
onCreateView in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns:

- View Newly created view. Return null for the default behavior.

10.6.1.d onCreateView (java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
public

Specified by:
onCreateView in interface android.view.LayoutInflater.Factory

Overrides:
onCreateView in class android.app.Activity

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns:

- View Newly created view. Return null for the default behavior.

10.6.1.e onOptionsItemSelect (android.view.MenuItem *item*)

public boolean onOptionsItemSelected(android.view.MenuItem item)

This hook is called whenever an item in your options menu is selected. The default implementation simply returns false to have the normal processing happen (calling the item's Runnable or sending a message to its Handler as appropriate). You can use this method for any items for which you would like to do processing without those other facilities.

Derived classes should call through to the base class for it to perform the default menu handling.

Modifier and Type:
public boolean

Overrides:
onOptionsItemSelected in class android.app.Activity

Parameters:

- item - The menu item that was selected.

Returns:

- boolean Return false to allow normal menu processing to proceed, true to consume it here.

10.6.2 Method Summary

10.6.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.6.2.b Methods inherited from class android.support.v4.app.FragmentActivity

ump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.6.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.6.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.6.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClass-

Loader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.6.2.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.6.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.6.3 Nested Class Summary

10.6.3.a Nested classes/interfaces inherited from class android.app.Activity

android.app.Activity.TranslucentConversionListener

10.6.3.b Nested classes/interfaces inherited from class android.content.Context

android.content.Context.BindServiceFlags, android.content.Context.CreatePackageOptions
, android.content.Context.ServiceName

10.6.4 Feild Summary

10.6.4.a Fields inherited from class android.app.Activity

```
DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL,
    DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER,
    RESULT_OK
```

10.6.4.b Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
    CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE,
    CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY, CONTEXT_INCLUDE_CODE
    , CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR, DEVICE_IDLE_CONTROLLER,
     DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE, DROPBOX_SERVICE,
    ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE, INPUT_METHOD_SERVICE,
    INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE, LAUNCHER_APPS_SERVICE,
    LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE, MEDIA_PROJECTION_SERVICE,
    MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE, MODE_APPEND,
    MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE, MODE_WORLD_READABLE,
    MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE, NETWORK_SCORE_SERVICE,
    NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE, NOTIFICATION_SERVICE,
    NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE, PRINT_SERVICE, RADIO_SERVICE,
    RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE, SERIAL_SERVICE, SIP_SERVICE,
    STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE, TELEPHONY_SERVICE,
    TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE, TRUST_SERVICE,
    TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE, USAGE_STATS_SERVICE, USB_SERVICE,
     USER_SERVICE, VIBRATOR_SERVICE, VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE,
    WIFI_P2P_SERVICE, WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE,
    WIFI_SERVICE, WINDOW_SERVICE
```

10.6.5 Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.6.6 Generation

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

- CrowdControlApplication

10.7 CrowdControlApplication Class Reference

Class CrowdControlApplication

public class ***CrowdControlApplication*** extends android.app.Application

The official singleton object for the application.

All Implemented Interfaces:

ndroid.content.ComponentCallbacks, android.content.ComponentCallbacks2

Member Functions

- onCreate ()

10.7.1 Method Detail

10.7.1.a onCreate ( )

public void onCreate()

Called when the application is starting, before any activity, service, or receiver objects (excluding content providers) have been created. Implementations should be as quick as possible (for example using lazy initialization of state) since the time spent in this function directly impacts the performance of starting the first activity, service, or receiver in a process. If you override this method, be sure to call super.onCreate().

Modifier and Type:
public void

Overrides:
onCreate in class android.app.Application

10.7.2 Method Summary

10.7.2.a Methods inherited from class android.app.Application

onConfigurationChanged, onLowMemory, onTerminate, onTrimMemory, registerActivityLifecycleCallbacks, registerComponentCallbacks, registerOnProvideAssistDataListener, unregisterActivityLifecycleCallbacks, unregisterComponentCallbacks, unregisterOnProvideAssistDataListener

10.7.3 Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, canStartActivityForResult, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getResources, getSharedPreferences, getSharedPrefsFile, getSystemService, getSystemServiceName, getTheme, getThemeResId, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setTheme, setWallpaper, setWallpaper, startActivities, startActivities, startActivitiesAsUser, startActivity, startActivity, startActivityAsUser, startActivityAsUser, startActivityForResult, startInstrumentation, startIntentSender, startIntentSender, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.7.4 Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes

10.7.5 Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.7.6 Nested Class Summary

10.7.6.a Nested classes/interfaces inherited from class android.app.Activity

android.app.Activity.TranslucentConversionListener

10.7.6.b Nested classes/interfaces inherited from class android.content.Context

android.content.Context.BindServiceFlags, android.content.Context.CreatePackageOptions
, android.content.Context.ServiceName

Field Functions

- aGroup ()
- aUser ()

10.7.7 Field Detail

10.7.7.a aGroup ( )

aGroup
public static com.parse.ParseObject aGroup

10.7.7.b aUser ( )

aUser
public static com.parse.ParseUser aUser

10.7.8 Feild Summary

10.7.8.a Fields inherited from class android.app.Application

mLoadedApk

10.7.8.b Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
```

CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE, CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY, CONTEXT_INCLUDE_CODE , CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR, DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE, DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE, INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE, LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE, MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE, MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE, NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE, NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE, PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE, SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE, TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE, TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE, USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE, VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE, WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE, WINDOW_SERVICE

10.7.8.c Fields inherited from interface android.content.ComponentCallbacks2

TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE, TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE, TRIM_MEMORY_UI_HIDDEN

10.7.9 Generation

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

- CrowdControlApplication

10.8 EventFragment Class Reference

Class EventFragment

public interface ***EventFragment*** **extends:**
ndroid.support.v4.app.Fragment

A simple Fragment subclass. Handles showing the user potential Events they could attend

All Known Implementing Classes:
android.content.ComponentCallbacks, android.view.View.OnCreateContextMenuListener

Member Functions

- **newInstance ()**
- **onButtonPressed ()**
- **onCreate ()**
- **onCreateView ()**
- **onDetach ()**

10.8.1 Method Detail

10.8.1.a newInstance (java.lang.String *text*)

public void onCreate(android.os.Bundle savedInstanceState)

Use this factory method to create a new instance of this fragment using the provided parameters.

Modifier and Type:
public void

Parameters:

- text - Test text 1

Returns:
A new instance of fragment EventFragment.

10.8.1.b onButtonPressed (android.net.Uri *uri*)

10.8.1.c onCreate (android.os.Bundle *savedInstanceState*)

public void onCreate(android.os.Bundle savedInstanceState)

Called to do initial creation of a fragment. This is called after Fragment.onAttach(Activity) and before Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle).

Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see Fragment.onActivityCreated(Bundle).

Modifier and Type:
public void

Overrides:
onCreate in class android.support.v4.app.Fragment

Parameters:

- savedInstanceState - If the fragment is being re-created from a previous saved state, this is the state.

10.8.1.d onCreateView (android.view.LayoutInflater *inflater* android.view.ViewGroup *container* android.os.Bundle *avedInstanceState*)

public android.view.View onCreateView(android.view.LayoutInflater inflater,
android.view.ViewGroup container,
android.os.Bundle savedInstanceState)

alled to have the fragment instantiate its user interface view. This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between Fragment.onCreate(Bundle) and Fragment.onActivityCreated(Bundle).

If you return a View from here, you will later be called in Fragment.onDestroyView() when the view is being released.

Modifier and Type:
public android.view.View

Overrides:
onCreateView in class android.support.v4.app.Fragment

Parameters:

- inflater - The LayoutInflater object that can be used to inflate any views in the fragment,
- container - If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
- savedInstanceState - If non-null, this fragment is being re-constructed from a previous saved state as given here.

Overrides:
Return the View for the fragment's UI, or null.

10.8.1.e onDetach (android.view.LayoutInflater *inflater* android.view.ViewGroup *container* android.os.Bundle *avedInstanceState*)

public void onDetach())

Called when the fragment is no longer attached to its activity. This is called after Fragment.onDestroy().

Overrides:
onDetach in class android.support.v4.app.Fragment

10.8.2 Method Summary

10.8.2.a Methods inherited from class android.support.v4.app.Fragment

dump, equals, getActivity, getAllowEnterTransitionOverlap, getAllowReturnTransitionOverlap, getArguments, getChildFragmentManager, getContext, getEnterTransition, getExitTransition, getFragmentManager, getHost, getId, getLayoutInflater, getLoaderManager, getParentFragment, getReenterTransition, getResources, getRetainInstance, getReturnTransition, getSharedElementEnterTransition, getSharedElementReturnTransition, getString, getString, getTag, getTargetFragment, getTargetRequestCode, getText, getUserVisibleHint, getView, hashCode, hasOptionsMenu, instantiate, instantiate, isAdded, isDetached, isHidden, isInLayout, isMenuVisible, isRemoving, isResumed, isVisible, onActivityCreated, onActivityResult, onAttach, onAttach, onConfigurationChanged, onContextItemSelected, onCreateAnimation, onCreateContextMenu, onCreateOptionsMenu, onDestroy, onDestroyOptionsMenu, onDestroyView, onHiddenChanged, onInflate, onInflate, onLowMemory, onOptionsItemSelected, onOptionsMenuClosed, onPause, onPrepareOptionsMenu, onRequestPermissionsResult, onResume, onSaveInstanceState, onStart, onStop, onViewCreated, onViewStateRestored, registerForContextMenu, requestPermissions, setAllowEnterTransitionOverlap, setAllowReturnTransitionOverlap, setArguments, setEnterSharedElementCallback, setEnterTransition, setExitSharedElementCallback, setExitTransition, setHasOptionsMenu, setInitialSavedState, setMenuVisibility, setReenterTransition, setRetainInstance, setReturnTransition, setSharedElementEnterTransition, setSharedElementReturnTransition, setTargetFragment,

setUserVisibleHint, shouldShowRequestPermissionRationale, startActivity, startActivityForResult, toString, unregisterForContextMenu

10.8.2.b Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

10.8.3 Nested Class Summary

10.8.4 Generation

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

- **BaseModel**

10.9 GroupCreateActivityl Class Reference

Class GroupCreateActivity

public class GroupCreateActivity **extends android.support.v7.app.AppCompatActivity implements android.view.View.OnClickListener**

The base model interface, providing access to core model functionality, including: - unique object identifier - the initial object creation - the last updated date - model saving and loading.

All Known Implementing Classes:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2, android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback, android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator, android.support.v4.app.TaskStackBuilder.SupportParentable, android.support.v7.app.ActionBarDrawerToggle.Delegatel android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2, android.view.View.OnClickListener, android.view.View.OnCreateContextMenuList android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- **onClick ()**
- **onCreateView ()**
- **onCreateView ()**

10.9.1 Method Detail

10.9.1.a onClick (android.view.View *view*)

public void onClick(android.view.View view)

Called when a view has been clicked.

Modifier and Type:
public android.view.View

Parameters:

- view - The view that was clicked.

10.9.1.b onCreateView (android.view.View *parent* java.lang.String *name* android.content.Context *context* ndroid.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
public android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory2

Overrides: onCreateView in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns:

- View Newly created view. Return null for the default behavior.

10.9.1.c onCreateView (java.lang.String *name* android.content.Context *context* ndroid.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

tandard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St
This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
public android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateView in class android.app.Activity

Parameters:

- name - Tag name to be inflated.

- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns:

- View Newly created view. Return null for the default behavior.

10.9.2 Method Summary

10.9.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.9.2.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.9.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreateOptionsMenu, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsItemSelected, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState,

onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.9.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.9.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.9.2.f Methods inherited from class android.content.Context

etColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.9.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.9.3 Feild Summary

10.9.3.a Fields inherited from class android.app.Activity

```
DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL,
    DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER,
    RESULT_OK
```

10.9.3.b Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
    CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE,
    CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY,
    CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR,
    DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE,
    DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE,
    INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE,
    LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE,
    MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE,
    MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE,
    MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE,
    NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE,
    NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE,
    PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE,
    SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE,
    TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE,
    TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE,
    USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE,
    VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE,
    WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE,
    WINDOW_SERVICE
```

10.9.3.c Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.9.4 Generation

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

- GroupCreateActivity

10.10 GroupInfoFragment Class Reference

Class GroupInfoFragment

public class GroupInfoFragment extends android.support.v4.app.Fragment

A simple Fragment subclass. Activities that contain this fragment must implement the to handle interaction events. Use the newInstance(java.lang.String) factory method to create an instance of this fragment. Will display information of the current group and allow the user to leave the group

All Known Implementing Classes:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2,
android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,
android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator,
android.support.v4.app.TaskStackBuilder.SupportParentable,
android.support.v7.app.ActionBarDrawerToggle.DelegateProvider,
android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback,
android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2,
android.view.View.OnClickListener, android.view.View.OnCreateContextMenuListener,
android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- newInstance ()
- onCreate ()
- onCreateView ()

10.10.1 Method Detail

10.10.1.a newInstance (java.lang.String *text*)

newInstance(java.lang.String text))

Use this factory method to create a new instance of this fragment using the provided parameters.

Modifier and Type:
static GroupInfoFragment

Parameters:

- text - Test text 1

Returns: A new instance of fragment GroupInfoFragment.

10.10.1.b onCreate (java.lang.String *text*)

public void onCreate(android.os.Bundle savedInstanceState)

Called to do initial creation of a fragment. This is called after Fragment.onAttach(Activity) and before Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle).

Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see Fragment.onActivityCreated(Bundle).

Modifier and Type:
void

Overrides: onCreate in class android.support.v4.app.Fragment.

Parameters:

- savedInstanceState - If the fragment is being re-created from a previous saved state, this is the state.

10.10.1.c onCreateView (android.view.LayoutInflater *inflater* android.view.ViewGroup *container* android.os.Bundle *savedInstanceState*)

public android.view.View onCreateView(android.view.LayoutInflater inflater,
android.view.ViewGroup container,
android.os.Bundle savedInstanceState)

Called to have the fragment instantiate its user interface view. This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between Fragment.onCreate(Bundle) and Fragment.onActivityCreated(Bundle).

If you return a View from here, you will later be called in Fragment.onDestroyView() when the view is being released.

Modifier and Type:
void

Overrides: onCreate in class android.support.v4.app.Fragment.

Parameters:

- inflater - The LayoutInflater object that can be used to inflate any views in the fragment,
- container - If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
- savedInstanceState - If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns: Return the View for the fragment's UI, or null.

10.10.2 Method Summary

10.10.2.a Methods inherited from class android.support.v4.app.Fragment

dump, equals, getActivity, getAllowEnterTransitionOverlap, getAllowReturnTransitionOverlap, getArguments, getChildFragmentManager, getContext, getEnterTransition, getExitTransition, getFragmentManager, getHost, getId, getLayoutInflater, getLoaderManager, getParentFragment, getReenterTransition, getResources, getRetainInstance, getReturnTransition, getSharedElementEnterTransition, getSharedElementReturnTransition, getString, getString, getTag, getTargetFragment, getTargetRequestCode, getText, getUserVisibleHint, getView, hashCode, hasOptionsMenu, instantiate, instantiate, isAdded, isDetached, isHidden, isInLayout, isMenuVisible, isRemoving, isResumed, isVisible, onActivityCreated, onActivityResult, onAttach, onAttach, onConfigurationChanged, onContextItemSelected, onCreateAnimation, onCreateContextMenu, onCreateOptionsMenu, onDestroy, onDestroyOptionsMenu, onDestroyView, onDetach, onHiddenChanged, onInflate, onInflate, onLowMemory, onOptionsItemSelected, onOptionsMenuClosed, onPause, onPrepareOptionsMenu, onRequestPermissionsResult, onResume, onSaveInstanceState, onStart, onStop, onViewCreated, onViewStateRestored, registerForContextMenu, requestPermissions, setAllowEnterTransitionOverlap, setAllowReturnTransitionOverlap, setArguments, setEnterSharedElementCallback, setEnterTransition, setExitSharedElementCallback, setExitTransition, setHasOptionsMenu, setInitialSavedState, setMenuVisibility, setReenterTransition, setRetainInstance, setReturnTransition, setSharedElementEnterTransition, setSharedElementReturnTransition, setTargetFragment, setUserVisibleHint, shouldShowRequestPermissionRationale, startActivity, startActivityForResult, toString, unregisterForContextMenu

10.10.2.b Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

10.10.3 Nested Class Summary

10.10.3.a Nested classes/interfaces inherited from class android.support.v4.app.Fragment

- android.support.v4.app.Fragment.InstantiationException
- android.support.v4.app.Fragment.SavedState

10.10.4 Generation

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

- GroupCreateActivity

10.11 GroupJoinActivity Class Reference

Class GroupJoinActivity

public class GroupJoinActivity
extendsandroid.support.v7.app.AppCompatActivity
implements android.view.View.OnClickListener

A simple Fragment subclass. Activities that contain this fragment must implement the to handle interaction events. Use the newInstance(java.lang.String) factory method to create an instance of this fragment. Will display information of the current group and allow the user to leave the group

All Known Implementing Classes:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2,

android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,
android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator,
android.support.v4.app.TaskStackBuilder.SupportParentable,
android.support.v7.app.ActionBarDrawerToggle.DelegateProvider, android.support.v7.app.AppCompatCallback,
android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2,
android.view.View.OnClickListener, android.view.View.OnCreateContextMenuListener,
android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- onClick ()
- onCreateOptionsMenu ()
- onCreateView ()
- onCreateView ()
- onOptionsItemSelected ()

10.11.1 Method Detail

10.11.1.a onClick (java.lang.String *text*)

public void onClick(android.view.View view)

Called when a view has been clicked.

Modifier and Type:
void

Specified by: onClick in interface android.view.View.OnClickListener

Parameters:

- view - The view that was clicked.

10.11.2 Method Detail

10.11.2.a onCreateOptionsMenu (android.view.Menu *menu*)

public boolean onCreateOptionsMenu(android.view.Menu menu)

nitialize the contents of the Activity's standard options menu. You should place your menu items in to menu.

This is only called once, the first time the options menu is displayed. To update the menu every time it is displayed, see Activity.onPrepareOptionsMenu(android.view.Menu).

The default implementation populates the menu with standard system menu items. These are placed in the Menu.CATEGORY_SYSTEM group so that they will be correctly ordered with application-defined menu items. Deriving classes should always call through to the base implementation.

You can safely hold on to menu (and any items created from it), making modifications to it as desired, until the next time onCreateOptionsMenu() is called.

When you add items to the menu, you can implement the Activity's Activity.onOptionsItemSelected(android.view.Mer method to handle them there.

Modifier and Type:
boolean

Overrides: onCreateOptionsMenu in class android.app.Activity

Parameters:

- menu - The options menu in which you place your items.

Returns: You must return true for the menu to be displayed; if you return false it will not be shown.

10.11.2.b onCreateView (java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateView in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.11.2.c onCreateView (java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St
This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateView in class android.app.Activity

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.11.2.d onOptionsItemSelected (android.view.MenuItem *item*)

public boolean onOptionsItemSelected(android.view.MenuItem item)

This hook is called whenever an item in your options menu is selected. The default implementation simply returns false to have the normal processing happen (calling the item's Runnable or sending a message to its Handler as appropriate). You can use this method for any items for which you would like to do processing without those other facilities.

Derived classes should call through to the base class for it to perform the default menu handling.

Modifier and Type:
boolean

Overrides: onOptionsItemSelected in class android.app.Activity

Parameters:

- item - The menu item that was selected.

Returns: boolean Return false to allow normal menu processing to proceed, true to consume it here.

10.11.3 Method Summary

10.11.3.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.11.3.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.11.3.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.11.3.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.11.3.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.11.3.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.11.3.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.11.4 Nested Class Summary

10.11.4.a Nested classes/interfaces inherited from class android.app.Activity

- android.app.Activity.TranslucentConversionListener

10.11.4.b Nested classes/interfaces inherited from class android.content.Context

- android.content.Context.BindServiceFlags
- android.content.Context.CreatePackageOptions
- android.content.Context.ServiceName

10.11.5 Feild Summary

10.11.6 Fields inherited from class android.app.Activity

ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE, APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT, BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE, BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE, BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI, BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE, CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE, CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY, CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR, DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE, DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE, INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE, LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE, MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE, MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE, NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE, NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE, PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE, SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE, TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE, TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE, USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE, VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE, WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE, WINDOW_SERVICE

10.11.7 Fields inherited from class android.content.Context

ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE, APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT, BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE, BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE, BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI, BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE, CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE, CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY, CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR, DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE, DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE, INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE, LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE, MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE, MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE, NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE, NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE, PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE, SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE, TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE,

```
    TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE,
    USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE,
    VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE,
    WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE,
    WINDOW_SERVICE
```

10.11.8 Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.11.9 Generation

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

- **GroupCreateActivity**

10.12 GroupModel Interface Reference

Interface GroupModel

Interface GroupModel
extends BaseModel

The interface for group models.

All Known Implementing Classes:

Member Functions

- **getGeneralLocation ()**
- **getGroupDescription ()**
- **getGroupMembers ()**

10.12.1 Method Detail

10.12.1.a getGeneralLocation ( )

com.parse.ParseGeoPoint getGeneralLocation()

Gets the general location of the group.

Modifier and Type:
com.parse.ParseGeoPoint

Returns: **The location of the group in the form of a ParseGeoPoint object.**

10.12.1.b getGroupDescription ( )

java.lang.String getGroupDescription()

Gets the description of the current group.

Modifier and Type:
java.lang.String

Returns: The description string attached to the group.

10.12.1.c getGroupMembers ( )

com.parse.ParseUser getGroupMembers()

Gets the list of users associated with the current group.

Modifier and Type:
com.parse.ParseUser

Returns: The list of users as ParseUserModel objects that belong to the group.

10.12.2 Method Summary

10.12.2.a Methods inherited from interface com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.12.3 Nested Class Summary

10.12.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback ()
- BaseModel.SaveCallback ()

10.12.4 Generation

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

- GroupModel

10.13 GroupNavigationActivity Class Reference

Class GroupNavigationActivity

class GroupNavigationActivity
extendsandroid.support.v7.app.AppCompatActivity

This Activity will manage all the tabs related to the current group It uses a tab based system to switch between fragments.

All Known Implementing Classes:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2, android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,

android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator, android.support.v4.app.TaskStackBuilder.SupportParentable, android.support.v7.app.ActionBarDrawerToggle.DelegateProvider, android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2, android.view.View.OnCreateContextMenuListener, android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- onCreateView ()
- onCreateView ()

10.13.1 Method Detail

10.13.1.a onCreateView ( )

public android.view.View onCreateView(android.view.View parent, java.lang.String name, android.content.Context context, android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory2

Overrides: onCreateView in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.13.1.b onCreateView ( )

public android.view.View onCreateView(java.lang.String name, android.content.Context context, android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St
This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateView in class android.app.Activity

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.13.2 Method Summary

10.13.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.13.2.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.13.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot,

isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreateOptionsMenu, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsItemSelected, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.13.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.13.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, send-

Broadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.13.2.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.13.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.13.3 Nested Class Summary

10.13.3.a Nested classes/interfaces inherited from class android.app.Activity

- **android.app.Activity.TranslucentConversionListener**

10.13.3.b Nested classes/interfaces inherited from class android.content.Context

- **android.content.Context.BindServiceFlags**
- **android.content.Context.CreatePackageOptions**
- **android.content.Context.ServiceName**

10.13.4 Feild Summary

10.13.5 Fields inherited from class android.app.Activity

```
DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL,
    DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER,
    RESULT_OK
```

10.13.6 Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
    CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE,
    CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY,
    CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR,
    DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE,
    DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE,
```

```
    INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE,
    LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE,
    MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE,
    MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE,
    MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE,
    NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE,
    NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE,
    PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE,
    SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE,
    TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE,
    TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE,
    USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE,
    VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE,
    WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE,
    WINDOW_SERVICE
```

10.13.7 Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.13.8 Generation

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

- GroupNavigationActivity

10.14 LoginActivity Class Reference

Class LoginActivity

class LoginActivity
extends android.support.v7.app.AppCompatActivity
implements android.app.LoaderManager.LoaderCallbacks<android.database.Cursor>

All Known Implementing Classes:
android.app.LoaderManager.LoaderCallbacks<android.database.Cursor>,
android.content.ComponentCallbacks, android.content.ComponentCallbacks2,
android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,
android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator,
android.support.v4.app.TaskStackBuilder.SupportParentable,
android.support.v7.app.ActionBarDrawerToggle.DelegateProvider,
android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback,
android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2,
android.view.View.OnCreateContextMenuListener, android.view.Window.Callback,
android.view.Window.OnWindowDismissedCallback

Member Functions

- onCreateLoader ()
- onCreateView ()

- onCreateView ()
- onLoaderReset ()
- onLoadFinished ()
- onRequestPerissionsResult ()

10.14.1 Method Detail

10.14.1.a onCreateLoader (int *i* android.os.Bundle *bundle*)

public android.content.Loader<android.database.Cursor> onCreateLoader(int i,
android.os.Bundle bundle)

onCreateLoader in interface android.app.LoaderManager.LoaderCallbacks<android.database.Cursor>

Modifier and Type:
android.content.Loader<android.database.Cursor>

Specified by: onCreateLoader in interface android.app.LoaderManager.LoaderCallbacks<android.database.Cur

Parameters:

- i - The ID whose loader is to be created.
- bundle - Any arguments supplied by the caller.

Returns: Return a new Loader instance that is ready to start loading.

10.14.1.b onCreateView (android.view.View *parent* java.lang.String *name* android.content.Context *context* ndroid.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
android.content.Loader<android.database.Cursor>

Specified by: onCreateView in interface android.view.LayoutInflater.Factory2

Overrides: onCreateView in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.14.1.c onCreateView (java.lang.String *name* android.content.Context *context* ndroid.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
android.content.Loader<android.database.Cursor>

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateView in class android.app.Activity

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.14.1.d onLoaderReset (android.content.Loader<android.database.Cursor> *cursorLoader*)

public void onLoaderReset(android.content.Loader<android.database.Cursor> cursorLoader)

Called when a previously created loader is being reset, and thus making its data unavailable. The application should at this point remove any references it has to the Loader's data.

Modifier and Type:
void>

Specified by: onLoaderReset in interface android.app.LoaderManager.LoaderCallbacks<android.database.Curs

Parameters:

- cursorLoader - The Loader that is being reset.

10.14.1.e onLoadFinished (android.content.Loader<android.database.Cursor> *cursorLoader* android.database.Cursor *cursor*)

public void onLoadFinished(android.content.Loader<android.database.Cursor> cursorLoader,
android.database.Cursor cursor)

Called when a previously created loader has finished its load. Note that normally an application is not allowed to commit fragment transactions while in this call, since it can happen after an activity's state is saved. See FragmentManager.openTransaction() for further discussion on this.

This function is guaranteed to be called prior to the release of the last data that was supplied for this Loader. At this point you should remove all use of the old data (since it will be released soon), but should not do your own release of the data since its Loader owns it and will take care of that. The Loader will take care of management of its data so you don't have to. In particular:

- The Loader will monitor for changes to the data, and report them to you through new calls here. You should not monitor the data yourself. For example, if the data is a Cursor and you place it in a CursorAdapter, use the CursorAdapter.CursorAdapter(android.content.Context, android.database.Cursor, int) constructor without passing in either CursorAdapter.FLAG_AUTO_REQUERY or CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER (that is, use 0 for the flags argument). This prevents the CursorAdapter from doing its own observing of the Cursor, which is not needed since when a change happens you will get a new Cursor throw another call here.
- The Loader will release the data once it knows the application is no longer using it. For example, if the data is a Cursor from a CursorLoader, you should not call close() on it yourself. If the Cursor is being placed in a CursorAdapter, you should use the CursorAdapter.swapCursor(android.database.Cursor) method so that the old Cursor is not closed.

Modifier and Type:
void

Specified by: onLoadFinished in interface android.app.LoaderManager.LoaderCallbacks<android.database.Curs

Parameters:

- cursorLoader - The Loader that has finished.
- cursor - The data generated by the Loader.

10.14.1.f onRequestPermissionsResult (android.content.Loader<android.database.Cursor> *cursorLoader* android.database.Cursor *cursor*)

public void onRequestPermissionsResult(int requestCode,
java.lang.String[] permissions,
int[] grantResults)

Callback received when a permissions request has been completed.

Modifier and Type:
void

Specified by: onRequestPermissionsResult in interface android.support.v4.app.ActivityCompat.OnRequestPerm

Overrides: onRequestPermissionsResult in class android.support.v4.app.FragmentActivity

Parameters:

- requestCode - The request code passed in Activity.requestPermissions(String[], int).
- permissions - The requested permissions. Never null.
- grantResults - The grant results for the corresponding permissions which is either PackageManager.PERMISSION_GRANTED or PackageManager.PERMISSION_DENIED. Never null.

10.14.2 Method Summary

10.14.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.14.2.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.14.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreateOptionsMenu, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsItemSelected, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOut-

side, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.14.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.14.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.14.2.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.14.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.14.3 Nested Class Summary

10.14.3.a Nested Classes

- **UserLoginTask ()**

10.14.3.b Nested classes/interfaces inherited from class android.content.Context

- **android.app.Activity.TranslucentConversionListener**

10.14.3.c Nested classes/interfaces inherited from class android.content.Context

- **android.content.Context.BindServiceFlags**
- **android.content.Context.CreatePackageOptions**
- **android.content.Context.ServiceName**

10.14.4 Feild Summary

10.14.5 Fields inherited from class android.app.Activity

```
DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL,
    DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER,
    RESULT_OK
```

10.14.6 Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
    CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE,
    CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY,
    CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR,
    DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE,
    DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE,
    INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE,
    LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE,
    MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE,
    MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE,
    MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE,
    NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE,
    NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE,
    PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE,
    SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE,
    TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE,
    TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE,
    USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE,
```

```
    VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE,
    WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE,
    WINDOW_SERVICE
```

10.14.7 Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.14.8 Generation

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

- **LoginActivity**

10.15 LoginController Class Reference

Class LoginController

class LoginActivity
extends java.lang.Object

10.15.1 Constructor Summary

10.15.1.a Constructor and Description

LoginController(android.content.Context app_context)

10.15.2 Method Summary

10.15.2.a Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.15.3 Constructor Detail

10.15.3.a LoginController

public LoginController(android.content.Context app_context)

10.15.4 Generation

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

- **LoginController**

10.16 MapFragment Class Reference

Class MapFragment

class MapFragment
extendsandroid.support.v4.app.Fragment

A simple Fragment subclass. Will Display a Google Map and place group members on it

Member Functions

- newInstance ()
- onCreate ()
- onCreateView ()

10.16.1 Method Detail

10.16.1.a newInstance (java.lang.String *text*)

public static MapFragment newInstance(java.lang.String text)

Use this factory method to create a new instance of this fragment using the provided parameters.

Modifier and Type:
static MapFragment

Parameters:

- text - Test text 1

Returns: A new instance of fragment MapFragment.

10.16.1.b onCreate (android.os.Bundle *savedInstanceState*)

public void onCreate(android.os.Bundle savedInstanceState)

Called to do initial creation of a fragment. This is called after Fragment.onAttach(Activity) and before Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle).

Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see Fragment.onActivityCreated(Bundle).

Modifier and Type:
void

Overrides: onCreate in class android.support.v4.app.Fragment

Parameters:

- savedInstanceState - If the fragment is being re-created from a previous saved state, this is the state.

10.16.1.c onCreateView (android.view.LayoutInflater *inflater* android.view.ViewGroup *container* android.os.Bundle *savedInstanceState*)

public void onCreate(android.os.Bundle savedInstanceState)

Called to have the fragment instantiate its user interface view. This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between Fragment.onCreate(Bundle) and Fragment.onActivityCreated(Bundle).

If you return a View from here, you will later be called in Fragment.onDestroyView() when the view is being released.

Modifier and Type:
android.view.View

Overrides: onCreate in class android.support.v4.app.Fragment

Parameters:

- inflater - The LayoutInflater object that can be used to inflate any views in the fragment
- container - If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
- savedInstanceState - If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns: Return the View for the fragment's UI, or null.

10.16.2 Method Summary

10.16.2.a Methods inherited from class android.support.v4.app.Fragment

dump, equals, getActivity, getAllowEnterTransitionOverlap, getAllowReturnTransitionOverlap, getArguments, getChildFragmentManager, getContext, getEnterTransition, getExitTransition, getFragmentManager, getHost, getId, getLayoutInflater, getLoaderManager, getParentFragment, getReenterTransition, getResources, getRetainInstance, getReturnTransition, getSharedElementEnterTransition, getSharedElementReturnTransition, getString, getString, getTag, getTargetFragment, getTargetRequestCode, getText, getUserVisibleHint, getView, hashCode, hasOptionsMenu, instantiate, instantiate, isAdded, isDetached, isHidden, isInLayout, isMenuVisible, isRemoving, isResumed, isVisible, onActivityCreated, onActivityResult, onAttach, onAttach, onConfigurationChanged, onContextItemSelected, onCreateAnimation, onCreateContextMenu, onCreateOptionsMenu, onDestroy, onDestroyOptionsMenu, onDestroyView, onDetach, onHiddenChanged, onInflate, onInflate, onLowMemory, onOptionsItemSelected, onOptionsMenuClosed, onPause, onPrepareOptionsMenu, onRequestPermissionsResult, onResume, onSaveInstanceState, onStart, onStop, onViewCreated, onViewStateRestored, registerForContextMenu, requestPermissions, setAllowEnterTransitionOverlap, setAllowReturnTransitionOverlap, setArguments, setEnterSharedElementCallback, setEnterTransition, setExitSharedElementCallback, setExitTransition, setHasOptionsMenu, setInitialSavedState, setMenuVisibility, setReenterTransition, setRetainInstance, setReturnTransition, setSharedElementEnterTransition, setSharedElementReturnTransition, setTargetFragment, setUserVisibleHint, shouldShowRequestPermissionRationale, startActivity, startActivityForResult, toString, unregisterForContextMenu

10.16.2.b Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

10.16.3 Nested Class Summary

10.16.3.a ested classes/interfaces inherited from class android.support.v4.app.Fragment

- **android.support.v4.app.Fragment.InstantiationException**
- **android.support.v4.app.Fragment.SavedState**

10.16.4 Generation

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

- **MapFragment**

10.17 MessagingFragment Class Reference

Class MessagingFragment

class MessagingFragment
extendsandroid.support.v4.app.Fragment

A simple Fragment subclass. Use the newInstance(java.lang.String) factory method to create an instance of this fragment. This Fragment will handle all messaging between user and group

All Known Implementing Classes:
android.content.ComponentCallbacks, android.view.View.OnCreateContextMenuListener

Member Functions

- **newInstance ()**
- **onButtonPress ()**
- **onCreate ()**
- **onCreateView ()**
- **onDetach ()**

10.17.1 Method Detail

10.17.1.a newInstance (java.lang.String *text*)

public static MessagingFragment newInstance(java.lang.String text)

Use this factory method to create a new instance of this fragment using the provided parameters.

Modifier and Type:
static MessagingFragment

Parameters:

- **text - Test text 1**

Returns: **A new instance of fragment MessagingFragment.**

10.17.1.b onButtonPress (android.net.Uri *uri*)

public void onButtonPressed(android.net.Uri uri)

Modifier and Type:
void

Parameters:

- **uri**

10.17.1.c onCreate (android.os.Bundle *savedInstanceState*)

public void onCreate(android.os.Bundle savedInstanceState)

Called to do initial creation of a fragment. This is called after Fragment.onAttach(Activity) and before Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle).

Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see Fragment.onActivityCreated(Bundle).

Modifier and Type:
void

Overrides: **onCreate in class android.support.v4.app.Fragment**

Parameters:

- **savedInstanceState - If the fragment is being re-created from a previous saved state, this is the state.**

10.17.1.d onCreateView (android.view.LayoutInflater *inflater* android.view.ViewGroup *container* android.os.Bundle *savedInstanceState*)

public android.view.View onCreateView(android.view.LayoutInflater inflater,
android.view.ViewGroup container,
android.os.Bundle savedInstanceState)

Called to have the fragment instantiate its user interface view. This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between Fragment.onCreate(Bundle) and Fragment.onActivityCreated(Bundle).

If you return a View from here, you will later be called in Fragment.onDestroyView() when the view is being released.

Modifier and Type:
void

Overrides: **onCreateView in class android.support.v4.app.Fragment**

Parameters:

- **inflater - The LayoutInflater object that can be used to inflate any views in the fragment**

- container - If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
- savedInstanceState - If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns: Return the View for the fragment's UI, or null.

10.17.1.e onDetach ( )

public void onDetach()

Called when the fragment is no longer attached to its activity. This is called after Fragment.onDestroy().

Modifier and Type:
void

Overrides: onCreateView in class android.support.v4.app.Fragment

10.17.2 Method Summary

10.17.2.a Methods inherited from class android.support.v4.app.Fragment

dump, equals, getActivity, getAllowEnterTransitionOverlap, getAllowReturnTransitionOverlap, getArguments, getChildFragmentManager, getContext, getEnterTransition, getExitTransition, getFragmentManager, getHost, getId, getLayoutInflater, getLoaderManager, getParentFragment, getReenterTransition, getResources, getRetainInstance, getReturnTransition, getSharedElementEnterTransition, getSharedElementReturnTransition, getString, getString, getTag, getTargetFragment, getTargetRequestCode, getText, getUserVisibleHint, getView, hashCode, hasOptionsMenu, instantiate, instantiate, isAdded, isDetached, isHidden, isInLayout, isMenuVisible, isRemoving, isResumed, isVisible, onActivityCreated, onActivityResult, onAttach, onAttach, onConfigurationChanged, onContextItemSelected, onCreateAnimation, onCreateContextMenu, onCreateOptionsMenu, onDestroy, onDestroyOptionsMenu, onDestroyView, onHiddenChanged, onInflate, onInflate, onLowMemory, onOptionsItemSelected, onOptionsMenuClosed, onPause, onPrepareOptionsMenu, onRequestPermissionsResult, onResume, onSaveInstanceState, onStart, onStop, onViewCreated, onViewStateRestored, registerForContextMenu, requestPermissions, setAllowEnterTransitionOverlap, setAllowReturnTransitionOverlap, setArguments, setEnterSharedElementCallback, setEnterTransition, setExitSharedElementCallback, setExitTransition, setHasOptionsMenu, setInitialSavedState, setMenuVisibility, setReenterTransition, setRetainInstance, setReturnTransition, setSharedElementEnterTransition, setSharedElementReturnTransition, setTargetFragment, setUserVisibleHint, shouldShowRequestPermissionRationale, startActivity, startActivityForResult, toString, unregisterForContextMenu

10.17.2.b Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

10.17.3 Nested Class Summary

10.17.3.a Nested classes/interfaces inherited from class android.support.v4.app.Fragment

- android.support.v4.app.Fragment.InstantiationException
- android.support.v4.app.Fragment.SavedState

10.17.4 Generation

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

- MessagingFragment

10.18 ParseBaseModel Class Reference

Class ParseBaseModel

class ParseBaseModel
extendsjava.lang.Object
implementsBaseModel ()

The base Parse implementation of models. Fully implements the

Member Functions

- getCreated ()
- getId ()
- getUpdated ()
- load ()
- loadInBackground ()
- save ()
- saveInBackground ()
- wasModified ()

10.18.1 Method Detail

10.18.1.a getCreated ( )

public java.util.Date getCreated()

Gets the object's creation timestamp.

Modifier and Type:
java.util.Date

Specified by:

Returns: The object's last update timestamp.

10.18.1.b getId ( )

public java.lang.String getId()

Gets this object's unique identifier.

Modifier and Type:
java.lang.String

Specified by: getId in BaseModel

Returns: The object's unique identifier.

10.18.1.c getUpdated( )

public java.util.Date getUpdated()

Gets the object's last updated timestamp.

Modifier and Type:
java.util.Date

Specified by: **wasModified in BaseModel**

Returns: **Whether or not the object has unsaved changes.**

10.18.1.d load( )

public void load()
throws com.parse.ParseException
Attempts to load the model from parse synchronously. This is a blocking function and thus should never be used on the main thread.

Modifier and Type:
void

Specified by: **load in BaseModel**

Throws: **java.lang.Exception - If an exception is thrown by Parse, it will be passed on to this function's caller.**
com.parse.ParseException.

10.18.1.e loadInBackground(BaseModel.LoadCallback *callback*)

public void loadInBackground(BaseModel.LoadCallback callback)

Attempts to save the model to Parse asynchronously and passes control back to the main thread by using the object passed as a parameter to this function.

Modifier and Type:
void

Specified by: **loadInBackground in BaseModel**

Parameters:

- **callback - The callback object to pass control to once the operation is completed. If no object is provided (or null is given), then no callback will be executed.**

10.18.1.f save( )

public void save()

throws com.parse.ParseException
Attempts to save the model to Parse synchronously. This is a blocking function and thus should never be used on the main thread.

Modifier and Type:
void

Specified by: save in BaseModel

Throws: java.lang.Exception - If an exception is thrown by Parse, it will be passed on to this function's caller. com.parse.ParseException

10.18.1.g saveInBackground(BaseModel.SaveCallback *callback*)

public void saveInBackground(BaseModel.SaveCallback callback)

Attempts to save the model to Parse asynchronously and passes control back to the main thread by using the object passed as a parameter to this function.

Modifier and Type:
void

Specified by: saveInBackground in BaseModel

Parameters:

- callback - The callback object to pass control to once the operation is completed. If no object is provided (or null is given), then no callback will be executed.

10.18.1.h wasModified( )

public java.lang.Boolean wasModified()

Gets whether or not the object has unsaved changes.
Modifier and Type:
java.lang.Boolean

Specified by: wasModified in BaseModel

Returns: Whether or not the object has unsaved changes.

10.18.2 Method Summary

10.18.2.a Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.18.3 Nested Class Summary

10.18.3.a Nested classes/interfaces inherited from class android.app.Activity

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.18.4 Constructor Summary

10.18.4.a Constructor and Description

ParseBaseModel(com.parse.ParseObject object)
The class constructor.

10.18.5 Generation

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

- **ParseBaseModel**

10.19 ParseGroupModel Class Reference

Class ParseGroupModel

class ParseGroupModel
extendsBaseModel ()
implementsGroupModel ()

Member Functions

- **getGeneralLocation ()**
- **getGeneralDescription ()**
- **getGroupMembers ()**

10.19.1 Method Detail

10.19.1.a getGeneralLocation ( )

public com.parse.ParseGeoPoint getGeneralLocation()

Gets the general location of the group.

Modifier and Type:
com.parse.ParseGeoPoint

Specified by: **getGeneralLocation in GroupModel**

Returns: **The location of the group in the form of a ParseGeoPoint object.**

10.19.1.b getGroupDescription ( )

public com.parse.ParseGeoPoint getGeneralLocation()

Gets the description of the current group.

Modifier and Type:
java.lang.String

Specified by: **getGroupDescription in GroupModel**

Returns: The description string attached to the group.

10.19.1.c getGroupMembers ( )

public com.parse.ParseUser getGroupMembers()

Gets the list of users associated with the current group.

Modifier and Type:
com.parse.ParseUser

Specified by: getGroupMembers in GroupModel

Returns: The list of users as ParseUserModel objects that belong to the group.

10.19.2 Method Summary

10.19.2.a Methods inherited from class com.bowtaps.crowdcontrol.model.ParseBaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.19.2.b Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.19.2.c Methods inherited from interface com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.19.3 Nested Class Summary

10.19.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.19.4 Constructor Summary

10.19.4.a Constructor and Description

ParseGroupModel(com.parse.ParseObject object)
The class constructor.

10.19.5 Constructor Detail

10.19.5.a ParseGroupModel

The class constructor. Initializes the model from an existing ParseUser.
Parameters: object - The object to use as a handle.

10.19.6 Generation

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

- ParseGroupModel

10.20 ParseUserModel Class Reference

Class ParseUserModel

class ParseUserModel
extendsParseBaseModel ()
implementsUserModel ()

The Parse implementation for the @link UserModel interface.

Member Functions

- getAuthData ()
- getEmail ()
- getEmailVerified ()
- getObjectID ()
- getPhone ()
- getUsername ()
- getUsername ()
- setEmail ()
- setPhone ()

10.20.1 Method Detail

10.20.1.a getAuthData ( )

public java.lang.Object getAuthData()

Modifier and Type:
java.lang.Object

Specified by: getAuthData in UserModel

10.20.1.b getEmail ( )

public java.lang.String getEmail()

Gets the user's email.

Modifier and Type:
java.lang.String

Specified by: getEmail in UserModel

10.20.1.c getEmailVerified ( )

public java.lang.Boolean getEmailVerified()

Gets the user's email.

Modifier and Type:
java.lang.Boolean

Specified by: **getEmailVerified in UserModel**

10.20.1.d getObjectID ( )

public java.lang.String getObjectID()

Modifier and Type:
java.lang.String

Specified by: **getObjectID in UserModel**

10.20.1.e getPhone ( )

public java.lang.String getPhone()

Gets the user's phone number.

Modifier and Type:
java.lang.String

Specified by: **getPhone in UserModel**

10.20.1.f getUsername ( )

public java.lang.String getUsername()

Gets the user's username.

Modifier and Type:
java.lang.String

Specified by: **getUsername in UserModel**

10.20.1.g getUsername ( )

public java.lang.String getUsername()

Modifier and Type:
java.lang.String

Specified by: **getUsername in UserModel**

10.20.1.h setEmail (java.lang.String *email*)

public void setEmail(java.lang.String email))

Sets the user's email.

Modifier and Type:
void

Specified by: setEmail in UserModel

Parameters:

- email - The new email address to assign to the user.

10.20.1.i setPhone (java.lang.String *phone*)

public void setPhone(java.lang.String phone)

Sets the user's email.

Modifier and Type:
void

Specified by: setPhone in UserModel

Parameters:

- phone - The new phone number to assign to the user.

10.20.2 Method Summary

10.20.2.a Methods inherited from class com.bowtaps.crowdcontrol.model.ParseBaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.20.2.b Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.20.2.c Methods inherited from interface com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.20.3 Nested Class Summary

10.20.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.20.4 Constructor Summary

10.20.4.a Constructor and Description

ParseUserModel(com.parse.ParseUser object)
The class constructor.

10.20.5 Constructor Detail

10.20.5.a ParseUserModel

public ParseUserModel(com.parse.ParseUser object)

The class constructor. Initializes the model from an existing ParseUser.

Parameters: **object - The object to use as a handle.**

10.20.6 Generation

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

- **ParseUserModel**

10.21 ParseUserProfileModel Class Reference

Class ParseUserProfileModel

class ParseUserProfileModel
extendsParseBaseModel ()
implementsUserModel ()

The Parse implementation for the @link UserModel interface.

Member Functions

- **getDisplayName ()**
- **setDisplayName ()**

10.21.1 Method Detail

10.21.1.a getDisplayName ( )

public java.lang.String getDisplayName()

Gets the user's display name.

Modifier and Type:
java.lang.String

Specified by: **getDisplayName in UserProfileModel**

Returns: **The user's display name.**

10.21.1.b setDisplayName (java.lang.String *displayName*)

public void setDisplayName(java.lang.String displayName)

Sets the user's display name.

Modifier and Type:
java.lang.Object

Specified by: setDisplayName in UserProfileModel

Parameters:

- displayName - The new display name for the user.

10.21.2 Method Summary

10.21.2.a Methods inherited from class com.bowtaps.crowdcontrol.model.ParseBaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.21.2.b Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.21.2.c Methods inherited from interface com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.21.3 Nested Class Summary

10.21.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.21.4 Generation

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

- ParseUserProfileModel

10.22 SignupActivity Class Reference

Class SignupActivity

class SignupActivity
extendsandroid.support.v7.app.AppCompatActivity
implements android.app.LoaderManager.LoaderCallbacks<android.database.Cursor>, android.view.View.OnClick

A login screen that offers login via email/password.

All Known Implementing Classes:
android.app.LoaderManager.LoaderCallbacks<android.database.Cursor>,

android.content.ComponentCallbacks, android.content.ComponentCallbacks2, android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback, android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator, android.support.v4.app.TaskStackBuilder.SupportParentable, android.support.v7.app.ActionBarDrawerToggle.DelegateProvider, android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2, android.view.View.OnClickListener, android.view.View.OnCreateContextMenuListener, android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- onClick ()
- onCreateLoader ()
- onCreateView ()
- onCreateView ()
- onLoaderReset ()
- onLoaderFinished ()
- onRequestPermissionResult ()

10.22.1 Method Detail

10.22.1.a onClick (android.view.View *view*)

public void onClick(android.view.View view)

Called when a view has been clicked.

Modifier and Type:
void

Specified by: onClick in interface android.view.View.OnClickListener

Parameters:

- view - The view that was clicked.

10.22.1.b onCreateLoader (int *i* android.os.Bundle *bundle*)

public android.content.Loader<android.database.Cursor> onCreateLoader(int i, android.os.Bundle bundle)

Instantiate and return a new Loader for the given ID.

Modifier and Type:
android.content.Loader<android.database.Cursor>

Specified by: onClick in interface android.view.View.OnClickListener

Parameters:

- i - The ID whose loader is to be created.
- bundle - Any arguments supplied by the caller.

Returns: Return a new Loader instance that is ready to start loading.

10.22.1.c onCreateView (android.view.View *parent* java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory2

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.22.1.d onCreateView (java.lang.String *name* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St
This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.22.1.e onLoaderReset (android.content.Loader<android.database.Cursor> *cursorLoader*)

public void onLoaderReset(android.content.Loader<android.database.Cursor> cursorLoader)

Called when a previously created loader is being reset, and thus making its data unavailable. The application should at this point remove any references it has to the Loader's data.

Modifier and Type:
void

Specified by: onLoaderReset in interface android.app.LoaderManager.LoaderCallbacks<android.database.Curs

Parameters:

- cursorLoader - The Loader that is being reset.

10.22.1.f onLoadFinished (android.content.Loader<android.database.Cursor> *cursorLoader* android.database.Cursor *cursor*)

public void onLoadFinished(android.content.Loader<android.database.Cursor> cursorLoader, android.database.Cursor cursor)

Called when a previously created loader has finished its load. Note that normally an application is not allowed to commit fragment transactions while in this call, since it can happen after an activity's state is saved. See FragmentManager.openTransaction() for further discussion on this.

This function is guaranteed to be called prior to the release of the last data that was supplied for this Loader. At this point you should remove all use of the old data (since it will be released soon), but should not do your own release of the data since its Loader owns it and will take care of that. The Loader will take care of management of its data so you don't have to. In particular:

- The Loader will monitor for changes to the data, and report them to you through new calls here. You should not monitor the data yourself. For example, if the data is a Cursor and you place it in a CursorAdapter, use the CursorAdapter.CursorAdapter(android.content.Context, android.database.Cursor, int) constructor without passing in either CursorAdapter.FLAG_AUTO_REQUERY or CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER (that is, use 0 for the flags argument). This prevents the CursorAdapter from doing its own observing of the Cursor, which is not needed since when a change happens you will get a new Cursor throw another call here.
- the Loader will release the data once it knows the application is no longer using it. For example, if the data is a Cursor from a CursorLoader, you should not call close() on it yourself. If the Cursor is being placed in a CursorAdapter, you should use the CursorAdapter.swapCursor(android.database.Cursor) method so that the old Cursor is not closed.

Modifier and Type:
void

Specified by: onLoaderReset in interface android.app.LoaderManager.LoaderCallbacks<android.database.Curs

Parameters:

- cursorLoader - The Loader that has finished.
- cursor - The data generated by the Loader.

10.22.1.g onRequestPermissionsResult (int *requestCode* java.lang.String[] *permissions* int[] *grantResults*)

ublic void onRequestPermissionsResult(int requestCode,
java.lang.String[] permissions,
int[] grantResults)

Callback received when a permissions request has been completed.

Modifier and Type:
void

Specified by: nRequestPermissionsResult in interface android.support.v4.app.ActivityCompat.OnRequestPermi

Overrides: onRequestPermissionsResult in class android.support.v4.app.FragmentActivity

Parameters:

- requestCode - The request code passed in Activity.requestPermissions(String[], int).
- permissions - The requested permissions. Never null.
- grantResults - The grant results for the corresponding permissions which is either PackageManager.PERMISSION_GRANTED or PackageManager.PERMISSION_DENIED. Never null.

10.22.2 Method Summary

10.22.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.22.2.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.22.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild,

finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemSelected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreateOptionsMenu, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsItemSelected, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.22.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.22.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver,

getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.22.2.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.22.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.22.3 Constructor Detail

10.22.3.a SignupActivity

public SignupActivity()

10.22.4 Nested Class Summary

10.22.4.a Nested classes

- SignupActivity.UserLoginTask - Represents an asynchronous login/registration task used to authenticate the user.

10.22.4.b Nested classes/interfaces inherited from class android.app.Activity

- android.app.Activity.TranslucentConversionListener

10.22.4.c Nested classes/interfaces inherited from class android.content.Context

- android.content.Context.BindServiceFlags
- android.content.Context.CreatePackageOptions
- android.content.Context.ServiceName

10.22.5 Feild Summary

10.22.6 Fields inherited from class android.app.Activity

DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL, DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER, RESULT_OK

10.22.7 Fields inherited from class android.content.Context

ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE, APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT, BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE, BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE, BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI, BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE, CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE, CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY, CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR, DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE, DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE, INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE, LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE, MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE, MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE, NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE, NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE, PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE, SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE, TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE, TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE, USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE, VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE, WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE, WINDOW_SERVICE

10.22.8 Fields inherited from interface android.content.ComponentCallbacks2

TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE, TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE, TRIM_MEMORY_UI_HIDDEN

10.22.9 Generation

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

- **SignupActivity**

10.23 UserModel Class Reference

Class UserModel

class UserModel

extendsBaseModel ()

All Superinterfaces: BaseModel ()

All Known Implementing Classes: ParseUserModel ()

The Parse implementation for the @link UserModel interface.

Member Functions

- **getAuthData ()**
- **getEmail ()**
- **getEmailVerified ()**
- **getObjectID ()**
- **getPhone ()**
- **getUsername ()**
- **getUsername ()**
- **setEmail ()**
- **setPhone ()**

10.23.1 Method Detail

10.23.1.a getAuthData ( )

public java.lang.Object getAuthData()

Modifier and Type:
java.lang.Object

10.23.1.b getEmail ( )

public java.lang.String getEmail()

Gets the user's email.

Modifier and Type:
java.lang.String

Returns:
The user's email address.

10.23.1.c getEmailVerified ( )

public java.lang.Boolean getEmailVerified()

Gets the user's email.

Modifier and Type:
java.lang.Boolean

Returns: **True if the user's email address has been verified, false if not.**

10.23.1.d getObjectID ( )

public java.lang.String getObjectID()

Modifier and Type:
java.lang.String

10.23.1.e getPhone ( )

public java.lang.String getPhone()

Gets the user's phone number.

Modifier and Type:
java.lang.String

Returns: **The user's phone number.**

10.23.1.f getUsername ( )

public java.lang.String getUsername()

Gets the user's username.

Modifier and Type:
java.lang.String

10.23.1.g getUsername ( )

public java.lang.String getUsername()

ets the username for the user. This value should not be changed, as it is set at creation time and is a unique identifier for this object.

Modifier and Type:
java.lang.String

Returns : **The user's unique username.**

10.23.1.h setEmail (java.lang.String *email*)

public void setEmail(java.lang.String email))

Replaces the user's existing email address with a new one.

Modifier and Type:
void

Parameters:

- **email - The new email address to assign to the user.**

10.23.1.i setPhone (java.lang.String *phone*)

public void setPhone(java.lang.String phone)

Replaces the user's phone number with a new one

Modifier and Type:
void

Parameters:

- phone - The new phone number to assign to the user.

10.23.2 Method Summary

10.23.2.a Methods inherited from class com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.23.3 Nested Class Summary

10.23.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.23.4 Generation

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

- UserModel

10.24 UserProfileModel Interface Reference

Interface UserProfileModel

class ParseUserProfileModel
extendsParseBaseModel ()
implementsUserModel ()

The interface for user profile models, providing access to public-facing user profile data, such as display name and profile image.

Member Functions

- getDisplayName()
- setDisplayName ()

10.24.1 Method Detail

10.24.1.a getDisplayName ( )

public java.lang.String getDisplayName()

Gets the display name of the current user. This value is not unique, is chosen by the user, should always be used to represent the user when presented to other users, and should never be used as an index key. This value can and should, however, be indexed for use in text searches. Additionally, the user's display name can be changed and thus should calls to this function should be performed relatively frequently.

Modifier and Type:
java.lang.String

Returns: String of the user's self-chosen display name.

10.24.1.b setDisplayName (java.lang.String *displayName*)

public void setDisplayName(java.lang.String displayName)

Sets the user's display name that will be seen by other users.

Modifier and Type:
java.lang.Object

Parameters:

- displayName - The new display name for the user.

10.24.2 Method Summary

10.24.2.a Methods inherited from interface com.bowtaps.crowdcontrol.model.BaseModel

getCreated, getId, getUpdated, load, loadInBackground, save, saveInBackground, wasModified

10.24.3 Nested Class Summary

10.24.3.a Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- BaseModel.LoadCallback
- BaseModel.SaveCallback

10.24.4 Generation

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

- UserProfileModel

10.25 WelcomeActivity Class Reference

Class WelcomeActivityl

class WelcomeActivity
extendsandroid.support.v7.app.AppCompatActivity
implementsndroid.view.View.OnClickListener

All Known Implementing Classes:
android.content.ComponentCallbacks, android.content.ComponentCallbacks2,
android.support.v4.app.ActivityCompat.OnRequestPermissionsResultCallback,
android.support.v4.app.ActivityCompatApi23.RequestPermissionsRequestCodeValidator,

android.support.v4.app.TaskStackBuilder.SupportParentable,
android.support.v7.app.ActionBarDrawerToggle.DelegateProvider,
android.support.v7.app.AppCompatCallback, android.view.KeyEvent.Callback,
android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2,
android.view.View.OnClickListener, android.view.View.OnCreateContextMenuListener,
android.view.Window.Callback, android.view.Window.OnWindowDismissedCallback

Member Functions

- onClick ()
- onCreateOptionsMenu ()
- onCreateView ()
- onCreateView ()
- onOptionsItemSelected ()

10.25.1 Method Detail

10.25.1.a onClick (android.view.View *view*)

public void onClick(android.view.View view)

Called when a view has been clicked.

Modifier and Type:
void

Specified by: onClick in interface android.view.View.OnClickListener

Parameters:

- view - The view that was clicked.

10.25.1.b onCreateOptionsMenu (android.view.Menu *menu*)

public boolean onCreateOptionsMenu(android.view.Menu menu)

Initialize the contents of the Activity's standard options menu. You should place your menu items in to menu.

This is only called once, the first time the options menu is displayed. To update the menu every time it is displayed, see Activity.onPrepareOptionsMenu(android.view.Menu).

The default implementation populates the menu with standard system menu items. These are placed in the Menu.CATEGORY_SYSTEM group so that they will be correctly ordered with applicationdefined menu items. Deriving classes should always call through to the base implementation.

You can safely hold on to menu (and any items created from it), making modifications to it as desired, until the next time onCreateOptionsMenu() is called.

When you add items to the menu, you can implement the Activity's Activity.onOptionsItemSelected(android.view.Mer
method to handle them there.

Modifier and Type:
boolean

Overrides: onCreateOptionsMenu in class android.app.Activity

Parameters:

- menu - The options menu in which you place your items.

Returns: You must return true for the menu to be displayed; if you return false it will not be shown.

10.25.1.c onCreateView (android.view.View *parent* java.lang.String *nam* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(android.view.View parent,
java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String). This implementation handles tags to embed fragments inside of the activity.

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory2

Overrides: onCreateOptionsMenu in class android.app.Activity

Parameters:

- parent - The parent that the created view will be placed in; note that this may be null.
- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.25.1.d onCreateView (java.lang.String *nam* android.content.Context *context* android.util.AttributeSet *attrs*)

public android.view.View onCreateView(java.lang.String name,
android.content.Context context,
android.util.AttributeSet attrs)

Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.St
This implementation does nothing and is for pre-Build.VERSION_CODES.HONEYCOMB apps. Newer apps should use Activity.onCreateView(View, String, Context, AttributeSet).

Modifier and Type:
android.view.View

Specified by: onCreateView in interface android.view.LayoutInflater.Factory

Overrides: onCreateOptionsMenu in class android.app.Activity

Parameters:

- name - Tag name to be inflated.
- context - The context the view is being created in.
- attrs - Inflation attributes as specified in XML file.

Returns: View Newly created view. Return null for the default behavior.

10.25.2 Method Summary

10.25.2.a Methods inherited from class android.support.v7.app.AppCompatActivity

addContentView, getDelegate, getDrawerToggleDelegate, getMenuInflater, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onMenuItemSelected, onMenuOpened, onPanelClosed, onPrepareSupportNavigateUpTaskStack, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onWindowStartingSupportActionMode, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask

10.25.2.b Methods inherited from class android.support.v4.app.FragmentActivity

dump, getLastCustomNonConfigurationInstance, getSupportFragmentManager, getSupportLoaderManager, onAttachFragment, onBackPressed, onCreatePanelMenu, onKeyDown, onLowMemory, onPreparePanel, onRequestPermissionsResult, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode

10.25.2.c Methods inherited from class android.app.Activity

canStartActivityForResult, closeContextMenu, closeOptionsMenu, convertFromTranslucent, convertToTranslucent, createPendingResult, dismissDialog, dispatchEnterAnimationComplete, dispatchGenericMotionEvent, dispatchKeyEvent, dispatchKeyShortcutEvent, dispatchPopulateAccessibilityEvent, dispatchTouchEvent, dispatchTrackballEvent, findViewById, finish, finishActivity, finishActivityFromChild, finishAffinity, finishAfterTransition, finishAndRemoveTask, finishFromChild, getActionBar, getActivityToken, getApplication, getCallingActivity, getCallingPackage, getChangingConfigurations, getComponentName, getContentScene, getContentTransitionManager, getCurrentFocus, getFragmentManager, getIntent, getLastNonConfigurationInstance, getLayoutInflater, getLoaderManager, getLocalClassName, getMediaController, getParent, getParentActivityIntent, getPreferences, getReferrer, getRequestedOrientation, getSearchEvent, getSystemService, getTaskId, getTitle, getTitleColor, getVoiceInteractor, getVolumeControlStream, getWindow, getWindowManager, hasWindowFocus, isBackgroundVisibleBehind, isChangingConfigurations, isChild, isDestroyed, isFinishing, isImmersive, isResumed, isTaskRoot, isVoiceInteraction, isVoiceInteractionRoot, managedQuery, managedQuery, moveTaskToBack, navigateUpTo, navigateUpToFromChild, onActionModeFinished, onActionModeStarted, onActivityReenter, onAttachedToWindow, onAttachFragment, onBackgroundVisibleBehindChanged, onContextItemS-

elected, onContextMenuClosed, onCreate, onCreateContextMenu, onCreateDescription, onCreateNavigateUpTaskStack, onCreatePanelView, onCreateThumbnail, onDetachedFromWindow, onEnterAnimationComplete, onGenericMotionEvent, onKeyLongPress, onKeyMultiple, onKeyShortcut, onKeyUp, onNavigateUp, onNavigateUpFromChild, onNewActivityOptions, onOptionsMenuClosed, onPostCreate, onPrepareNavigateUpTaskStack, onPrepareOptionsMenu, onProvideAssistContent, onProvideAssistData, onProvideReferrer, onRestoreInstanceState, onSaveInstanceState, onSearchRequested, onSearchRequested, onTouchEvent, onTrackballEvent, onTrimMemory, onUserInteraction, onVisibleBehindCanceled, onWindowAttributesChanged, onWindowDismissed, onWindowFocusChanged, onWindowStartingActionMode, onWindowStartingActionMode, openContextMenu, openOptionsMenu, overridePendingTransition, postponeEnterTransition, recreate, registerForContextMenu, releaseInstance, removeDialog, reportFullyDrawn, requestPermissions, requestVisibleBehind, requestWindowFeature, runOnUiThread, setActionBar, setContentTransitionManager, setDefaultKeyMode, setEnterSharedElementCallback, setExitSharedElementCallback, setFeatureDrawable, setFeatureDrawableAlpha, setFeatureDrawableResource, setFeatureDrawableUri, setFinishOnTouchOutside, setImmersive, setIntent, setMediaController, setPersistent, setProgress, setProgressBarIndeterminate, setProgressBarIndeterminateVisibility, setProgressBarVisibility, setRequestedOrientation, setResult, setResult, setSecondaryProgress, setTaskDescription, setTitle, setTitle, setTitleColor, setVisible, setVolumeControlStream, shouldShowRequestPermissionRationale, shouldUpRecreateTask, showAssist, showDialog, showDialog, showLockTaskEscapeMessage, startActionMode, startActionMode, startActivities, startActivities, startActivity, startActivity, startActivityAsCaller, startActivityAsUser, startActivityAsUser, startActivityForResult, startActivityForResult, startActivityForResultAsUser, startActivityForResultAsUser, startActivityFromChild, startActivityFromChild, startActivityFromFragment, startActivityFromFragment, startActivityIfNeeded, startActivityIfNeeded, startIntentSender, startIntentSender, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromChild, startIntentSenderFromChild, startLockTask, startManagingCursor, startNextMatchingActivity, startNextMatchingActivity, startPostponedEnterTransition, startSearch, stopLockTask, stopManagingCursor, takeKeyEvents, triggerSearch, unregisterForContextMenu

10.25.2.d Methods inherited from class android.view.ContextThemeWrapper

applyOverrideConfiguration, getResources, getTheme, getThemeResId, setTheme

10.25.2.e Methods inherited from class android.content.ContextWrapper

bindService, bindServiceAsUser, checkCallingOrSelfPermission, checkCallingOrSelfUriPermission, checkCallingPermission, checkCallingUriPermission, checkPermission, checkPermission, checkSelfPermission, checkUriPermission, checkUriPermission, checkUriPermission, clearWallpaper, createApplicationContext, createConfigurationContext, createDisplayContext, createPackageContext, createPackageContextAsUser, databaseList, deleteDatabase, deleteFile, enforceCallingOrSelfPermission, enforceCallingOrSelfUriPermission, enforceCallingPermission, enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAssets, getBaseContext, getBasePackageName, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir, getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences, getSharedPrefsFile, getSystemServiceName, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight, getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper, registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast, removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastAsUser, sendBroadcastMultiplePermissions, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser, sendStick-

yOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper, startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser, stopService, stopServiceAsUser, unbindService, unregisterReceiver

10.25.2.f Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, unregisterComponentCallbacks

10.25.2.g Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10.25.3 Nested Class Summary

10.25.3.a Nested classes/interfaces inherited from class android.app.Activity

- **android.app.Activity.TranslucentConversionListener**

10.25.3.b Nested classes/interfaces inherited from interface com.bowtaps.crowdcontrol.model.Base

- **android.content.Context.BindServiceFlags**
- **android.content.Context.CreatePackageOptions**
- **android.content.Context.ServiceName**

10.25.4 Field Summary

10.25.4.a Fields inherited from class android.app.Activity

```
DEFAULT_KEYS_DIALER, DEFAULT_KEYS_DISABLE, DEFAULT_KEYS_SEARCH_GLOBAL,
    DEFAULT_KEYS_SEARCH_LOCAL, DEFAULT_KEYS_SHORTCUT, RESULT_CANCELED, RESULT_FIRST_USER,
    RESULT_OK
```

10.25.4.b Fields inherited from class android.content.Context

```
ACCESSIBILITY_SERVICE, ACCOUNT_SERVICE, ACTIVITY_SERVICE, ALARM_SERVICE, APP_OPS_SERVICE,
    APPWIDGET_SERVICE, AUDIO_SERVICE, BACKUP_SERVICE, BATTERY_SERVICE, BIND_ABOVE_CLIENT,
    BIND_ADJUST_WITH_ACTIVITY, BIND_ALLOW_OOM_MANAGEMENT, BIND_AUTO_CREATE,
    BIND_DEBUG_UNBIND, BIND_FOREGROUND_SERVICE, BIND_FOREGROUND_SERVICE_WHILE_AWAKE,
    BIND_IMPORTANT, BIND_NOT_FOREGROUND, BIND_NOT_VISIBLE, BIND_SHOWING_UI,
    BIND_TREAT_LIKE_ACTIVITY, BIND_VISIBLE, BIND_WAIVE_PRIORITY, BLUETOOTH_SERVICE,
    CAMERA_SERVICE, CAPTIONING_SERVICE, CARRIER_CONFIG_SERVICE, CLIPBOARD_SERVICE,
    CONNECTIVITY_SERVICE, CONSUMER_IR_SERVICE, CONTEXT_IGNORE_SECURITY,
    CONTEXT_INCLUDE_CODE, CONTEXT_REGISTER_PACKAGE, CONTEXT_RESTRICTED, COUNTRY_DETECTOR,
    DEVICE_IDLE_CONTROLLER, DEVICE_POLICY_SERVICE, DISPLAY_SERVICE, DOWNLOAD_SERVICE,
    DROPBOX_SERVICE, ETHERNET_SERVICE, FINGERPRINT_SERVICE, HDMI_CONTROL_SERVICE,
    INPUT_METHOD_SERVICE, INPUT_SERVICE, JOB_SCHEDULER_SERVICE, KEYGUARD_SERVICE,
    LAUNCHER_APPS_SERVICE, LAYOUT_INFLATER_SERVICE, LOCATION_SERVICE,
    MEDIA_PROJECTION_SERVICE, MEDIA_ROUTER_SERVICE, MEDIA_SESSION_SERVICE, MIDI_SERVICE,
    MODE_APPEND, MODE_ENABLE_WRITE_AHEAD_LOGGING, MODE_MULTI_PROCESS, MODE_PRIVATE,
```

```
    MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, NETWORK_POLICY_SERVICE,
    NETWORK_SCORE_SERVICE, NETWORK_STATS_SERVICE, NETWORKMANAGEMENT_SERVICE, NFC_SERVICE,
    NOTIFICATION_SERVICE, NSD_SERVICE, PERSISTENT_DATA_BLOCK_SERVICE, POWER_SERVICE,
    PRINT_SERVICE, RADIO_SERVICE, RESTRICTIONS_SERVICE, SEARCH_SERVICE, SENSOR_SERVICE,
    SERIAL_SERVICE, SIP_SERVICE, STATUS_BAR_SERVICE, STORAGE_SERVICE, TELECOM_SERVICE,
    TELEPHONY_SERVICE, TELEPHONY_SUBSCRIPTION_SERVICE, TEXT_SERVICES_MANAGER_SERVICE,
    TRUST_SERVICE, TV_INPUT_SERVICE, UI_MODE_SERVICE, UPDATE_LOCK_SERVICE,
    USAGE_STATS_SERVICE, USB_SERVICE, USER_SERVICE, VIBRATOR_SERVICE,
    VOICE_INTERACTION_MANAGER_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE,
    WIFI_PASSPOINT_SERVICE, WIFI_RTT_SERVICE, WIFI_SCANNING_SERVICE, WIFI_SERVICE,
    WINDOW_SERVICE
```

10.25.4.c Fields inherited from interface android.content.ComponentCallbacks2

```
TRIM_MEMORY_BACKGROUND, TRIM_MEMORY_COMPLETE, TRIM_MEMORY_MODERATE,
    TRIM_MEMORY_RUNNING_CRITICAL, TRIM_MEMORY_RUNNING_LOW, TRIM_MEMORY_RUNNING_MODERATE,
    TRIM_MEMORY_UI_HIDDEN
```

10.25.5 Generation

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

- **WelcomeActivity**

10.26 AppDelegate Class Reference

Class AppDelegate

public class AppDelegate

class AppDelegate: UIResponder, UIApplicationDelegate

The core delegate for the app. Contains pseudo-globa variables and functions that can be accessed from anywhere in the app

Member Functions

- **instance ()**
- **window ()**
- **modelManager ()**
- **application) ()**
- **application ()**
- **applicationWillResignActive ()**
- **applicationDidEnterBackground ()**
- **applicationWillEnterForeground ()**
- **applicationDidBecomeActive ()**
- **applicationWillTerminate ()**

10.26.1 Method Summary

10.26.1.a instance ( )

The main instance of this class for the entire application.

Modifier and Type:
static var instance

Declaration: Swift

```
static var instance: AppDelegate
```

10.26.1.b window ( )

The main instance of this class for the entire application.

Modifier and Type:
var window

Declaration: Swift

```
var window: UIWindow?
```

10.26.1.c modelManager ( )

Optional reference to the current model manager.

Modifier and Type:
var modelManager

Declaration: Swift

```
var modelManager: ModelManager?
```

10.26.1.d application ( )

application(_:didFinishLaunchingWithOptions:)
Modifier and Type:
class AppDelegate

Declaration: Swift

```
class AppDelegate: UIResponder, UIApplicationDelegate
```

10.26.1.e application ( )

application(_:openURL:sourceApplication:annotation:)
Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.1.f applicationWillResignActive ( )

applicationWillResignActive(_:)

Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.1.g applicationDidEnterBackground ( )

applicationDidEnterBackground(_:)

Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.1.h applicationWillEnterForeground ( )

applicationWillEnterForeground(_:)

Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.1.i applicationDidBecomeActive ( )

applicationDidBecomeActive(_:)

Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.1.j applicationWillTerminate ( )

applicationWillTerminate(_:)

Modifier and Type:
class AppDelegate

Declaration: Swift

class AppDelegate: UIResponder, UIApplicationDelegate

10.26.2 Generation

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

- AppDelegate

10.27 BaseModel Class Reference

Class BaseModel

public class BaseModel
protocol BaseModel

Base model protocol, which is to be implemented by all model classes in the project. Provides access to basic information about the model, such as:

- Unique identifier
- Creation timestamp
- Last updated timestamp
- Flag indicating modification since last save
- Methods for loading from and saving to storage

Member Functions

- id ()
- created ()
- updated ()
- modified ()
- load ()
- loadInBackground ()
- save ()
- saveInBackground ()

10.27.1 Method Summary

10.27.1.a id ( )

The ID of the object as determined by remote storage. This value is automatically generated when the object is stored and is usually determined by a remote server. It is valid for this value to be nil, and thus should be used with care and only if absolutely necessary.

Modifier and Type:
var id:

Declaration: Swift

var id: String get

10.27.1.b created ( )

Timestamp of when this object was first created and stored. This value may be automatically determined by the server if the model is using a remote machine for storage.

Modifier and Type:
var created:

Declaration: Swift

var created: NSDate get

10.27.1.c updated ( )

Timestamp of the last time this object was updated in storage. This value may be automatically determined by the server if the model is using a remote machine for storage.

Modifier and Type:
var created:

Declaration: Swift

var created: NSDate get

10.27.1.d modified ( )

Flag indicating whether or not this object has been modified since it was last pulled from or pushed to storage.

Modifier and Type:
var modified:

Declaration: Swift

var modified: Bool get

10.27.1.e load ( )

Loads this object from storage, either local or remote as determined by the implementation.

This is a blocking function and should be executed on a thread separate from the main thread. See loadInBackground(_:) for loading this object on a separate thread. This function will throw an exception if an error occurs.

Modifier and Type:
func

Declaration: Swift

func load() throws

10.27.1.f loadInBackground ( )

loadInBackground(_:) Loads this object from storage, either local or remote as determined by the implementation.

This function spawns a new thread and reloads this object from storage. After successful execution or if an error occurs, this function passes control back to the main thread by calling the closure that was optionally passed as an argument.

Modifier and Type:
var modified

Declaration: Swift

func loadInBackground(callback: ((object: BaseModel?, error: NSError?) -> Void)?)

Parameters:

- **callback - Optional callback function to call after successful execution or if an error occurs. If nil is provided, then the callback will not be called.**

10.27.1.g save ( )

Saves this object storage, either local or remote as determined by the implementation.

This is a blocking function and should be executed on a thread separate from the main thread. See saveInBackground(_:) for saving this object on a separate thread. This function will throw an exception if

Modifier and Type:
func

Declaration: Swift

save() throws

10.27.1.h save ( )

saveInBackground(_:)

Saves this object to storage, either local or remote as determined by the implementation.

This function spawns a new thread and sends this object to storage. After successful execution or if an error occurs, this function passes control back to the main thread by calling the closure that was optionally passed as an argument.

Modifier and Type:
func

Declaration: Swift

func saveInBackground(callback: ((object: BaseModel?, error: NSError?) -> Void)?)

Parameters:

- callback - Optional callback function to call after successful execution or if an error occurs. If nil is provided, then the callback will not be called.

10.27.2 Generation

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

- BaseModel

10.28 ChatViewController Class Reference

Class ChatViewController

public class ChatViewController

class ChatViewController: UITableViewController

Controller for the UITableView used to display a list of active conversations on the chat screen. Contains methods that fill the rows in the table with relevant content as well as making calls to the backend to load the current conversations

Member Functions

- conversations ()
- viewDidLoad ()
- numberOfSectionsInTableView ()
- tableView ()
- tableView ()

10.28.1 Method Summary

10.28.1.a conversations ( )

Array of ConversationModel objects to display in the table.

Modifier and Type:
var conversations:

Declaration: Swift

var conversations: [ConversationModel] = [

10.28.1.b viewDidLoad ( )

Modifier and Type:
class ChatViewController:

Declaration: Swift

class ChatViewController: UITableViewController

10.28.1.c numberOfSectionsInTableView ( )

Modifier and Type:
class ChatViewController:

Declaration: Swift

```
class ChatViewController: UITableViewController
```

10.28.1.d tableView ( )

tableView(_:numberOfRowsInSection:)
Modifier and Type:
class ChatViewController:

Declaration: Swift

```
class ChatViewController: UITableViewController
```

10.28.1.e tableView ( )

tableView(_:cellForRowAtIndexPath:)
Modifier and Type:
class ChatViewController:

Declaration: Swift

```
class ChatViewController: UITableViewController
```

10.28.2 Generation

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

- **ChatViewController**

10.29 ConversationModel Class Reference

Class ConversationModel

public class ConversationModel
class ConversationModel
Provides a simple model to be used as placeholder content for the UITableView.

Member Functions

- **name ()**
- **message ()**
- **time ()**
- **init ()**

10.29.1 Method Summary

10.29.1.a name ( )

Name of conversation

Modifier and Type:
let name:

Declaration: **Swift**

```
let name: String
```

10.29.1.b message ( )

Contents of the message

Modifier and Type:
let message:

Declaration: **Swift**

```
let message: String
```

10.29.1.c time ( )

Conversation timestamp

Modifier and Type:
let time:

Declaration: **Swift**

```
let time: String
```

10.29.1.d init ( )

Main constructor for the class. Initializes values using provided parameters. - Parameter name: The name of the conversation or person with whom the conversation is with. - Parameter message: The last message sent in the conversation. - Parameter time: The last time stamp a message was sent in this conversation.

Modifier and Type:
init

Declaration: **Swift**

```
init(name: String, message: String, time: String)
```

Parameters:

- **name - The name of the conversation or person with whom the conversation is with.**
- **message - The last message sent in the conversation.**

- time - The last time stamp a message was sent in this conversation.

10.29.2 Generation

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

- ConversationModel

10.30 GroupInfoviewController Class Reference

Class GroupInfoviewController

public class GroupInfoviewController
class GroupInfoViewController: UIViewController
Controller for manipulating the group info view, which displays basic group information and lists of members and group leaders.

Member Functions

- viewDidLoad ()
- didReceiveMemoryWarning ()
- onLeaveGroupButtonTapped ()

10.30.1 Method Summary

10.30.1.a viewDidLoad ( )

Override of super class

Modifier and Type:
override func

Declaration: Swift

override func viewDidLoad()

10.30.1.b didReceiveMemoryWarning ( )

Override of super class

Modifier and Type:
override func

Declaration: Swift

override func didReceiveMemoryWarning()

10.30.1.c onLeaveGroupButtonTapped ( )

Group leave button callback - Parameter sender: Button that called

Modifier and Type:
override func

Declaration: Swift

@IBAction func onLeaveGroupButtonTapped(sender: AnyObject)

Parameters:

- sender - Button that called

10.30.2 Generation

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

- GroupInfoviewController

10.31 GroupModel Class Reference

Class GroupModel

public class GroupModel
protocol GroupModel: BaseModel

Model protocol representing a group of users using the app. This model contains data accessible to all users of the app, including group name, group description, group leader, and members of the group. This model does not provide direct access to messages related to the group, nor does does it allow access to member locations, although it does provide a generic group location.

Member Functions

- generalLocation ()
- groupDescription ()
- groupName ()
- groupMembers ()
- groupLeader ()
- addGroupMember ()
- removeGroupMember ()

10.31.1 Method Summary

10.31.1.a generalLocation ( )

Group's general location, used for location filtering

Modifier and Type:
PFGeoPoint

Declaration: Swift

var generalLocation: PFGeoPoint get set

10.31.1.b groupDescription ( )

Group description set by the group leader during creation

Modifier and Type:
String

Declaration: **Swift**

var groupDescription: String get set

10.31.1.c groupName ( )

Group name set by the group leader during creation

Modifier and Type:
String

Declaration: **Swift**

var groupName: String get set

10.31.1.d groupMembers ( )

List of users who are members of the group. This property can only be modified by the addGroupMember(_:) and removeGroupMember(_:) methods.

Modifier and Type:
Array

Declaration: **Swift**

var groupMembers: [UserProfileModel] get

10.31.1.e groupLeader ( )

The user who is the designated leader of the group, usually the user who created the group.

Modifier and Type:
Array

Declaration: **Swift**

var groupLeader: UserProfileModel? get set

10.31.1.f addGroupMemeber ( )

addGroupMemeber(_:)
Method for adding a user as a member of the group. Model must be saved afterwards, as this method does not automatically save the model.

Modifier and Type:
Array

Declaration: func

func addGroupMember(member: UserProfileModel) -> Bool

10.31.1.g removeGroupMemeber ( )

removeGroupMemeber(_:)
Method for removing a user from the group. Model must be saved afterwards, as this method does not automatically save the model.

Modifier and Type:
Array

Declaration: func

func removeGroupMember(member: UserProfileModel) -> Bool

10.31.2 Generation

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

- GroupModel

10.32 GroupOverviewController Class Reference

Class GroupOverviewController

public class GroupOverviewController
class GroupOverviewController: UIViewController
Group Overview Controller contains the logic for the Group Overiew page.

Member Functions

- groupToDisplay ()
- groupLeader ()
- groupNameLabel ()
- groupLeaderLabel ()
- groupDescriptionLabel ()
- onRequestButtonTapped ()
- viewDidLoad ()

10.32.1 Method Summary

10.32.1.a groupToDisplay ( )

Group to display in current view

Modifier and Type:
var groupToDisplay:

Declaration: Swift

var groupToDisplay: GroupModel?

10.32.1.b groupLeader ( )

User display name of the Group Leader

Modifier and Type:
var groupLeader:

Declaration: Swift

var groupLeader: UserProfileModel?

10.32.1.c groupNameLabel ( )

UILabel for group name

Modifier and Type:
@IBOutlet weak var groupNameLabel:

Declaration: Swift

@IBOutlet weak var groupNameLabel: UILabel!

10.32.1.d groupLeaderLabel ( )

UILabel for group leader name

Modifier and Type:
@IBOutlet weak var groupLeaderLabel:

Declaration: Swift

@IBOutlet weak var groupLeaderLabel: UILabel!

10.32.1.e groupDescriptionLabel ( )

UILable for group description text

Modifier and Type:
@IBOutlet weak var groupDescriptionLabel:

Declaration: Swift

@IBOutlet weak var groupDescriptionLabel: UILabel!

10.32.1.f onRequestButtonTapped ( )

When user clicks on the group to join - Parameter sender: Caller of button

Modifier and Type:
@IBAction func onRequestButtonTapped

Declaration: Swift

@IBAction func onRequestButtonTapped(sender: AnyObject)

Parameters:

- sender - Caller of button

10.32.1.g viewDidLoad ( )

Overrides superclass then loads data from group object into view.

Modifier and Type:
override func

Declaration: Swift

override func viewDidLoad()

10.32.2 Generation

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

- GroupOverviewController

10.33 GroupTableController Class Reference

Class GroupTableController

public class GroupTableController
class GroupTableController: UITableViewController
Contains the logic for the Group table view

Member Functions

- groupTable ()
- groups ()
- SelectedGroup ()
- ViewDidLoad ()
- viewDidAppear ()
- rewindToGroupList ()
- numberOfSectionsInTableView ()
- tableView ()
- tableView ()
- tableView ()
- doRefresh ()
- prepareForSegue ()

10.33.1 Method Summary

10.33.1.a groupTable ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

```
class GroupTableController: UITableViewController
```

10.33.1.b groups ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

```
class GroupTableController: UITableViewController
```

10.33.1.c selectedGroup ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

```
class GroupTableController: UITableViewController
```

10.33.1.d viewDidLoad ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

```
class GroupTableController: UITableViewController
```

10.33.1.e viewDidAppear ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

```
class GroupTableController: UITableViewController
```

10.33.1.f rewindToGroupList ( )

Function for rewinding to this view, providing a nice target for screens to rewind to.

Parameter segue: The segue object communicated during transfer.

Modifier and Type:
@IBAction func rewindToGroupList

Declaration: Swift

@IBAction func rewindToGroupList(segue: UIStoryboardSegue)

Parameters:

- seque - The segue object communicated during transfer.

10.33.1.g numberOfSectionsInTableView ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.1.h tableView ( )

tableView(_:numberOfRowsInSection:)

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.1.i tableView ( )

tableView(_:cellForRowAtIndexPath:)

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.1.j tableView ( )

tableView(_:didSelectRowAtIndexPath:)

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.1.k doRefresh ( )

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.1.l prepareForSegue ( )

prepareForSegue(_:sender:)

Modifier and Type:
class GroupTableController:

Declaration: Swift

class GroupTableController: UITableViewController

10.33.2 Generation

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

- GroupTableController

10.34 LocationModel Class Reference

Class LocationModel

public class LocationModel
protocol LocationModel: BaseModel

Base model protocol, which is to be implemented by all model classes in the project. Provides access to basic information about the model, such as:

Member Functions

- longitude ()
- latitude ()
- recipient ()
- sender ()

10.34.1 Method Summary

10.34.1.a longitude ( )

String holding the longitudinal coordinate

Modifier and Type:
String

Declaration: **Swift**

```
var longitude: String {get set}
```

10.34.1.b latitude ( )

String holding the latitudel coordinate

Modifier and Type:
String

Declaration: **Swift**

```
var latitude: String {get set}
```

10.34.1.c recipient ( )

UserProfileModel holding the recipient information when the locations are being sent to parse so that they can be sent encrypted to the intended recipient

Modifier and Type:
String

Declaration: **Swift**

```
var recipient: UserProfileModel {get}
```

10.34.1.d sender ( )

This is the Current users' UserProfileModel holding the reference for the storage on parse.

Modifier and Type:
String

Declaration: **Swift**

```
var sender: UserProfileModel {get}
```

10.34.2 Generation

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

- **LocationModel**

10.35 LoginViewController Class Reference

Class LoginViewController

public class LoginViewController

class LoginViewController: UIViewController, UITextFieldDelegate
Custom view controller class for handling user logins.

Custom UIViewController class for handling user logins. Processes requests, validates input, and passes data to backend for checking with the server, and properly handles both successful and unsuccessful login requests.

Member Functions

- scrollView ()
- emailField ()
- passwordField ()
- submitButton ()
- loginFormFields ()
- init ()
- deinit ()
- viewDidLoad ()
- registerForKeyboardNotifications ()
- deregisterFromKeyboardNotifications ()
- adjustForKeyboard ()
- submitButtonTapped ()
- textFieldShouldReturn ()
- submitForm ()

10.35.1 Method Summary

10.35.1.a scrollView ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.b emailField ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.c passwordField ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.d submitButton ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.e loginFormFields ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.f init ( )

init(coder:)

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.g deinit ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.h viewDidLoad ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.i registerForKeyboardNotifications ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.j deregisterFromKeyboardNotifications ( )

Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.k adjustForKeyboard ( )

adjustForKeyboard(_:)
Modifier and Type:
class LoginViewController:

Declaration: Swift

class LoginViewController: UIViewController, UITextFieldDelegate

10.35.1.l submitButtonTapped ( )

submitButtonTapped(_:) Callback executed when the submit button has been tapped.
Modifier and Type:
@IBAction func submitButtonTapped

Declaration: Swift

@IBAction func submitButtonTapped(sender: AnyObject)

10.35.1.m textFieldShouldReturn ( )

textFieldShouldReturn(_:) Callback executed when the return button is tapped on the keyboard.

Modifier and Type:
@IBAction func submitButtonTapped

Declaration: Swift

func textFieldShouldReturn(textField: UITextField) -> Bool

10.35.1.n submitForm ( )

Hide the keyboard, validate form input, and attempt to log in the user.

Modifier and Type:
func

Declaration: Swift

func submitForm()

10.35.2 Generation

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

- LoginViewController

10.36 MapViewController Class Reference

Class MapViewController

public class MapViewController
lass MapViewController: UIViewController, CLLocationManagerDelegate
Controller for the map view, which displays an interactive map that displays the user's current location and the location of others in the group

Member Functions

- manager ()
- map ()
- viewDidLoad ()
- didReceiveMemoryWarning ()

10.36.1 Method Summary

10.36.1.a manager ( )

The location manager that handles automatic location tracking and display on the map. - SeeAlso: CLLocationManager

Modifier and Type:
let manager

Declaration: Swift

let manager = CLLocationManager()

10.36.1.b map ( )

Outlet to Interface Builder for the map view - SeeAlso: MKMapView

Modifier and Type:
@IBOutlet weak var map:

Declaration: Swift

@IBOutlet weak var map: MKMapView!

10.36.1.c viewDidLoad ( )

Modifier and Type:
class MapViewController:

Declaration: Swift

```
class MapViewController: UIViewController, CLLocationManagerDelegate
```

10.36.1.d didReceiveMemoryWarning ( )

Modifier and Type:
class MapViewController:

Declaration: Swift

```
class MapViewController: UIViewController, CLLocationManagerDelegate
```

10.36.2 Generation

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

- MapViewController

10.37 ModelManager Class Reference

Class GroupModel

public class GroupModel
class PModelManager
A class dedicated for providing model functionality that does not belong in any individual model, such as logging users in, signing up new users, or getting the currrent user

Member Functions

- logInUser ()
- logInUserInBackground ()
- createUser ()
- createUserInBackground ()
- currentUser ()
- logOutCurrentUser ()
- fetchGroups ()
- fetchGroupsInBackground ()
- currentGroup ()
- setCurrentGroup ()
- fetchCurrentGroup ()
- fetchCurrentGroupInBackground ()

10.37.1 Method Summary

10.37.1.a logInUser ( )

logInUser(_:password:)
Compares the submitted username and password to storage and returns a UserModel object if the user was successfully logged in. Otherewise, it throws an exception.

Modifier and Type:
init

Declaration: Swift

func logInUser(username: String, password: String) throws -> UserModel

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.

Return Value: A user object if the login was successful.

10.37.1.b logInUserInBackground ( )

logInUserInBackground(_:password:callback:)
Compares the submitted username and password to storage on a separate thread, executing the given callback if successful. If login is unsuccessful, it throws an exception.

Modifier and Type:
init

Declaration: Swift

func logInUserInBackground(username: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.
- callback - An optional callback to call once the operation is complete.

10.37.1.c createUser ( )

createUser(_:email:password:)
Attempts to create a new user in the system, ensuring that the given username is unique. Also creates the corresponding user profile object. Both objects are then stored in the server.
This is a blocking function that can take several seconds to complete. If an operation fails, then an exception will be thrown.

Modifier and Type:
init

Declaration:
Swift

func createUser(username: String, email: String, password: String) throws -> UserModel

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.

Return Value:
The newly created UserModel if the operation is successful.

10.37.1.d createUserInBackground ( )

createUserInBackground(_:email:password:callback:)
Attempts to create a new user in the system, ensuring that the given username is unique. Also creates the corresponding UserProfileObject. Both objects are then stored in the server.

This is an asynchronous function that will pass control back to the main thread by executing the given callback prameter if it is not nil.

Modifier and Type:
init

Declaration:
Swift

func createUserInBackground(username: String, email: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.
- callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.

10.37.1.e currentUser ( )

Retrieves the currently logged-in user. If no user is logged in, returns nil.

Modifier and Type:
init

Declaration:
Swift

func createUserInBackground(username: String, email: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

10.37.1.f logOutCurrentUser ( )

Logs out the currently logged in user, removing them from any caches and returning whether or not the operation was successful.

Modifier and Type:
init

Declaration:
Swift

func logOutCurrentUser() -> Bool

Return Vales: true if the operation was successful and the user was successfully logged out, false if not.

10.37.1.g fetchGroupsInBackground ( )

etches all gropus in storage asynchronously.

This is an asynchronous function that will pass control back to the main thread by executing the given callback parameter if it is not nil.

Modifier and Type:
init

Declaration:
Swift

func fetchGroupsInBackground(callback: ((results: [GroupModel]?, error: NSError?) -> Void)?) -> Void

Return Vales: Array of all GroupModel objects in storage.

10.37.1.h fetchGroups ( )

fetchGroupsInBackground(_:) Fetches all groups in storage synchronously.

This is a blocking function that can take several seconds to complete. If an operation fails, then an exception will be thrown.

Modifier and Type:
init

Declaration:
Swift

func fetchGroups() throws -> [GroupModel]

Parameters:

- **callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.**

10.37.1.i currentGroup ( )

Fetches all groups in storage synchronously.

Gets the cached currently active group of which the current user is member, if any. If no user is logged in or the logged in user is not a member of any groups, this method will return nil. This function does not access storage in any way.

This method deals exclusively with cached values. In order to update the cached value, either set the cached value directly using setCurrentGroup(_:) or allowing it to be set automatically using fetchCurrentGroup() and fetchCurrentGroupInBackground(_:).

Modifier and Type:
init

Declaration:
Swift

func fetchGroups() throws -> [GroupModel]

Parameters:

- callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.

Returns Value:
The GroupModel of which the logged in user (if any) is a member of, or nil if no such group exists.

10.37.1.j setCurrentGroup ( )

setCurrentGroup(_:)
Sets the current cached value of the active group. Set the value to nil to indicate that there are no currently active groups. This function does not modify storage in anyway.

Modifier and Type:
init

Declaration:
Swift

func setCurrentGroup(group: GroupModel?)

Parameters:

- group - The current active GroupModel, or nil if no groups are currently active.

10.37.2 Generation

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

- ParseModelManager

10.38 ParseBaseModel Class Reference

Class ParseBaseMode

public class ParseBaseMode
class ParseBaseModel: BaseModel
The core Parse implementation for models. Implements the BaseModel protocol and provies functionality to access the basic information about the model, such as:

- Unique identifier
- Creation timestamp
- Last updated timestamp
- Flag indicating modification since last save
- Methods for loading from and saving to storage

Member Functions

- parseObject ()
- init ()
- id ()
- created ()
- updated ()
- modified ()
- load ()
- loadInBackground ()
- save ()
- saveInBackground ()

10.38.1 Method Summary

10.38.1.a parseObject ( )

Internal reference to the Parse API object representing this object in the remote database.

Modifier and Type:
let parseObject:

Declaration: Swift

let parseObject: PFObject

10.38.1.b init ( )

init(withParseObject:) Class constructor. Initializes the instance from a PFObject.

Modifier and Type:
withParseObject object:

Declaration: Swift

init(withParseObject object: PFObject)

Parameters:

- withParseObject - The Parse object to tie this model to the Parse Parse database.

10.38.1.c id ( )

init(withParseObject:) Read-only computed value representing the ID of the object as defined in the BaseModel protocol.

Modifier and Type:
var id:

Declaration: Swift

var id: String

10.38.1.d created ( )

init(withParseObject:) Read-only computed value for the timestamp when this object was initially created as defined in the BaseModel protocol.

Modifier and Type:
var created:

Declaration: Swift

var created: NSDate

10.38.1.e updated ( )

init(withParseObject:) Read-only computed value for the timestamp when this object was initially created as defined in the BaseModel protocol.

Modifier and Type:
var created:

Declaration: Swift

var created: NSDate

10.38.1.f modified ( )

init(withParseObject:) Read-only computed value indicating whether or not the data contained in this model is "dirty", which is to say that this model contains changes that have not been saved to the server.

Modifier and Type:
var modified:

Declaration: Swift

var modified: Bool

10.38.1.g load ( )

init(withParseObject:) Reloads this object from Parse as defined by the BaseModel protocol.

Modifier and Type:
func

Declaration: Swift

func load() throws

10.38.1.h loadInBackground ( )

init(withParseObject:) Reloads this object from Parse asynchronously as defined by the BaseModel protocol.

Modifier and Type:
func

Declaration: Swift

func loadInBackground(callback: ((object: BaseModel?, error: NSError?) -> Void)?)

10.38.1.i save ( )

init(withParseObject:) Reloads this object from Parse asynchronously as defined by the BaseModel protocol.

Modifier and Type:
func

Declaration: Swift

func save() throws

10.38.1.j saveInBackground ( )

init(withParseObject:) Saves this object to Parse as defined by the BaseModel protocol.

Modifier and Type:
func

Declaration: Swift

func saveInBackground(callback: ((object: BaseModel?, error: NSError?) -> Void)?)

10.38.2 Generation

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

- MapViewController

10.39 ParseGroupModel Class Reference

Class ParseGroupModel

public class ParseGroupModel

class ParseGroupModel: ParseBaseModel, GroupModel

The Parse implementation of the GroupModel protocol. Extends ParseBaseModel class and implements the GroupModel protocol and is designed to allow access to a group's information, including the group members, group name, and group description.

Member Functions

- init ()
- init ()
- generalLocation ()
- groupDescription ()
- groupName ()
- groupMembers ()
- groupLeader ()
- addGroupMember ()
- removeGroupMember ()
- load ()
- loadInBackground ()
- createGroup ()
- getAll ()
- getAllInBackground ()
- getGroupContainingUser ()
- getGroupContainingUserInBackground ()

10.39.1 Method Summary

10.39.1.a init ( )

Default class contructor. Creates a new entry in the database if saved.

Modifier and Type:
init

Declaration: Swift

init

10.39.1.b init ( )

init(withParseObject:)
Class constructor. Initializes the isntance from a PFObject.

Modifier and Type:
init

Declaration: Swift

init

Parameteres:

- withParseObject - The Parse object to tie this model to the Parse database.

10.39.1.c generalLocation ( )

PFGeoPoint Object to store the groups general location. This field is used for looking up groups as well as future support with finding ads by location

Modifier and Type:
var generalLocation:

Declaration: Swift

```
var generalLocation: PFGeoPoint
```

10.39.1.d groupDescription ( )

String to hold the description of the group.

Modifier and Type:
var groupDescription:

Declaration: Swift

```
var groupDescription: String
```

10.39.1.e groupName ( )

String to hold the Name of the group.

Modifier and Type:
var groupName:

Declaration: Swift

```
var groupName: String
```

10.39.1.f groupMembers ( )

An array of UserProfileModel objects to keep track of the members of the group.

Modifier and Type:
var groupMembers:

Declaration: Swift

```
var groupMembers: [UserProfileModel]
```

10.39.1.g groupLeader ( )

The UserProfileModel object who is the designated leader of the group.

Modifier and Type:
var groupMembers:

Declaration: Swift

var groupLeader: UserProfileModel?

10.39.1.h addGroupMember ( )

addGroupMember(_:)

Method for adding a user as a member of the group. Model must be saved afterwards, as this method does not automatically save the model.

Modifier and Type:
func addGroupMember

Declaration: Swift

func addGroupMember(member: UserProfileModel) -> Bool

10.39.1.i removeGroupMember ( )

Method for removing a user from the group. Model must be saved afterwards, as this method does not automatically save the model.

Modifier and Type:
func removeGroupMember

Declaration: Swift

func removeGroupMember(member: UserProfileModel) -> Bool

10.39.1.j load ( )

Loads this object from Parse storage synchronously. In addition to the normal functionality inherited from ParseBaseModel, this function also fetches and caches the users who are members of this group.

Modifier and Type:
override func

Declaration: Swift

override func load() throws

10.39.1.k loadInBackground ( )

loadInBackground(_:)
Loads this object from Parse storage asynchronously. In addition to the normal functionality inherited from ParseBaseModel, this function also fetches and caches the users who are members of this group.

Modifier and Type:
override override func

Declaration: Swift

override func loadInBackground(callback: ((object: BaseModel?, error: NSError?) -> Void)?)

10.39.1.l createGroup ( )

createGroup(_:description:)
Function to create a group if there is not one that exists

Modifier and Type:
override override func

Declaration: Swift

static func createGroup(name: String, description: String) -> ParseGroupModel

Parameters:

- name - String containing the group name
- description - String containing the description of the group

Return Value: Object of type ParseGroupModel that contains the information

10.39.1.m getAll ( )

Fetches all groups in storage synchronously.

This is a blocking function that can take several seconds to complete. If an operation fails, then an exception will be thrown.

Modifier and Type:
override static func

Declaration: Swift

static func getAll() throws -> [ParseGroupModel]

Return Value: Array of group models in storage.

10.39.1.n getAllInBackground ( )

Fetches all groups in storage synchronously.

Fetches all groups in Parse storage asynchronously, returning control to the main thread through the provided callback (if any).

Modifier and Type:
override static func

Declaration: Swift

static func getAllInBackground(callback: ((results: [ParseGroupModel]?, error: NSError?) -> Void)?)

Parameters:

- callback - Optional closure that will be called on completion or if an error is encountered.

10.39.1.o getGroupContainingUser ( )

Fetches the first group to which the provided user belongs, if any. If no group is found that contains the provided user, then nil is returned.

This is a blocking function and should be executed on a thread separate from the main thread. See getGroupContainingUserInBackground(_:callback:) for fetching on a separate thread. This function will throw an exception if an error occurs.

Modifier and Type:
override static func

Declaration: Swift

static func getGroupContainingUser(user: ParseUserProfileModel) throws -> ParseGroupModel?

Parameters:

- user - The user to search for.

Returns: Optional ParseGroupModel object that has user as a member, or nil if no such group could be found. This object will be fully loaded from storage such that a call to ParseGroupModel.load() is not necessary.

10.39.1.p getGroupContainingUserInBackground ( )

getGroupContainingUserInBackground(_:callback:)

Fetches the first group to which the provided user belongs, if any. If no group is found that contains the provided user, then nil is returned.

This function spawns a new thread for querying storage. After successful execution or if an error occurs, this function passes control back to the main thread by calling the closure that was optionally passed as an argument.

Modifier and Type:
override static func

Declaration: Swift

static func getGroupContainingUserInBackground(user: ParseUserProfileModel, callback: ((result: ParseGroupModel?, error: NSError?) -> Void)?)

Parameters:

- user - The user to search for.
- callback - Optional callback function to call after successful execution or if an error occurs. If nil is provided, then the callback will not be called.

Returns: Optional ParseGroupModel object that has user as a member, or nil if no such group could be found. This object will be fully loaded from storage such that a call to ParseGroupModel.load() is not necessary.

10.39.2 Generation

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

- ParseGroupModel

10.40 ParseModelManager Class Reference

Class ParseGroupModel

public class ParseGroupModel
class ParseModelManager: ModelManager
A Parse implementation of the ModelManager protocol. Class designed to query Parse for various models, including users and groups.

Member Functions

- logInUser ()
- logInUserInBackground ()
- createUser ()
- createUserInBackground ()
- currentUser ()
- logOutCurrentUser ()
- fetchGroups ()
- fetchGroupsInBackground ()
- currentGroup ()
- setCurrentGroup ()
- fetchCurrentGroup ()
- fetchCurrentGroupInBackground ()

10.40.1 Method Summary

10.40.1.a logInUser ( )

logInUser(_:password:)
Compares the submitted username and password to storage and returns a UserModel object if the user was successfully logged in. Otherwise, it throws an exception.

Modifier and Type:
init

Declaration: Swift

func logInUser(username: String, password: String) throws -> UserModel

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.

Return Value: A user object if the login was successful.

10.40.1.b logInUserInBackground ( )

logInUserInBackground(_:password:callback:)

Compares the submitted username and password to storage on a separate thread, executing the given callback if successful. If login is unsuccessful, it throws an exception.

Modifier and Type:
init

Declaration: Swift

func logInUserInBackground(username: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.
- callback - An optional callback to call once the operation is complete.

10.40.1.c createUser ( )

createUser(_:email:password:)

Attempts to create a new user in the system, ensuring that the given username is unique. Also creates the corresponding user profile object. Both objects are then stored in the server.

This is a blocking function that can take several seconds to complete. If an operation fails, then an exception will be thrown.

Modifier and Type:
init

Declaration:
Swift

func createUser(username: String, email: String, password: String) throws -> UserModel

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.

Return Value:
The newly created UserModel if the operation is successful.

10.40.1.d createUserInBackground ( )

createUserInBackground(_:email:password:callback:)

Attempts to create a new user in the system, ensuring that the given username is unique. Also creates the corresponding UserProfileObject. Both objects are then stored in the server.

This is an asynchronous function that will pass control back to the main thread by executing the given callback prameter if it is not nil.

Modifier and Type:
init

Declaration:
Swift

func createUserInBackground(username: String, email: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

Parameters:

- username - The username of the user to log in.
- password - The password to use to log in.
- callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.

10.40.1.e currentUser ( )

Retrieves the currently logged-in user. If no user is logged in, returns nil.

Modifier and Type:
init

Declaration:
Swift

func createUserInBackground(username: String, email: String, password: String, callback: ((user: UserModel?, error: NSError?) -> Void)?) -> Void

10.40.1.f logOutCurrentUser ( )

Logs out the currently logged in user, removing them from any caches and returning whether or not the operation was successful.

Modifier and Type:
init

Declaration:
Swift

func logOutCurrentUser() -> Bool

Return Vales: true if the operation was successful and the user was successfully logged out, false if not.

10.40.1.g fetchGroupsInBackground ( )

etches all gropus in storage asynchronously.

This is an asynchronous function that will pass control back to the main thread by executing the given callback parameter if it is not nil.

Modifier and Type:
init

Declaration:
Swift

func fetchGroupsInBackground(callback: ((results: [GroupModel]?, error: NSError?) -> Void)?) -> Void

Return Vales: Array of all GroupModel objects in storage.

10.40.1.h fetchGroups ( )

fetchGroupsInBackground(_:) Fetches all groups in storage synchronously.

This is a blocking function that can take several seconds to complete. If an operation fails, then an exception will be thrown.

Modifier and Type:
init

Declaration:
Swift

func fetchGroups() throws -> [GroupModel]

Parameters:

- callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.

10.40.1.i currentGroup ( )

Fetches all groups in storage synchronously.

Gets the cached currently active group of which the current user is member, if any. If no user is logged in or the logged in user is not a member of any groups, this method will return nil. This function does not access storage in any way.

This method deals exclusively with cached values. In order to update the cached value, either set the cached value directly using setCurrentGroup(_:) or allowing it to be set automatically using fetchCurrentGroup() and fetchCurrentGroupInBackground(_:).

Modifier and Type:
init

Declaration:
Swift

func fetchGroups() throws -> [GroupModel]

Parameters:

- callback - The callback function that will be executed after the operation is complete, either successfully or unsuccessfully.

Returns Value:
The GroupModel of which the logged in user (if any) is a member of, or nil if no such group exists.

10.40.1.j setCurrentGroup ( )

setCurrentGroup(_:)
Sets the current cached value of the active group. Set the value to nil to indicate that there are no currently active groups. This function does not modify storage in anyway.

Modifier and Type:
init

Declaration:
Swift

func setCurrentGroup(group: GroupModel?)

Parameters:

- group - The current active GroupModel, or nil if no groups are currently active.

10.40.2 Generation

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

- ParseModelManager

10.41 ParseUserModel Class Reference

Class ParseUserModel

public class ParseUserModel
class ParseUserModel: ParseBaseModel, UserModel
This class extends the ParseBaseModel class and implements the UserModel protocol and is the class to access the current user's information from Parse

Member Functions

- profile ()
- init ()
- username ()
- emailVerified ()
- email ()
- phone ()
- createFromSignUp ()

10.41.1 Method Summary

10.41.1.a profile ( )

The model corresponding to this user's public profile.

Modifier and Type:
init

Declaration: Swift

```
let profile: UserProfileModel
```

10.41.1.b init ( )

init(withParseUser:profile:)
Class constructor. Initializes the instance from a PFObject.

Modifier and Type:
init

Declaration: Swift

```
init(withParseUser user: PFUser, profile: UserProfileModel? = nil)
```

Parameters:

- **withParseUser - The Parse user to tie this model to the Parse database.**
- **profile - An optional UserProfileModel to connect to this user. If no value is or nil is provided, this constructor will attempt to get the profile from the withParseUser model.**

10.41.1.c username ( )

String containing the current user's username as defined in the UserModel protocol.

Modifier and Type:
var username:

Declaration: Swift

```
var username: String
```

10.41.1.d emailVerified ( )

Boolean to store if the user has verified their email with parse as defined by the UserModel protocol.

Modifier and Type:
var emailVerified:

Declaration: Swift

```
var emailVerified: Bool
```

Return Value:
true if their email has been verified, false if their email has not been verified.

10.41.1.e email ( )

String containing the current users email as defined in the UserModel protocol.

Modifier and Type:
var emailVerified:

Declaration: Swift

var email: String

10.41.1.f createFromSignUp ( )

createFromSignUp(_:password:)

Main function for creating a new user. Automatically creates a corresponding UserProfileModel and returns both in a tuple.

Modifier and Type:
var phone:

Declaration: Swift

var phone: String

10.41.1.g createFromSignUp ( )

createFromSignUp(_:password:)

Main function for creating a new user. Automatically creates a corresponding UserProfileModel and returns both in a tuple.

Modifier and Type:
var phone:

Declaration:
Swift

var phone: String

Parameters:

- username - The new user's username.
- password - The new user's password.

Declaration:
A tuple containing the newly created UserModel object and its corresponding UserProfileModel.

10.41.2 Generation

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

- ParseUserModel

10.42 ParseUserProfileModel Class Reference

Class ParseUserProfileModel

public class ParseUserProfileModel

class ParseUserProfileModel: ParseBaseModel, UserModel

This class extends the ParseBaseModel class and implements the UserProfileModel protocol and is the class to access a user's public profile information from Parse.

Member Functions

- init ()
- init ()
- displayName ()

10.42.1 Method Summary

10.42.1.a init ( )

Default class contructor. Creates a new entry in the database if saved.

Modifier and Type:
init

Declaration: Swift

init()

10.42.1.b init ( )

init(withParseObject:)

Default class contructor. Creates a new entry in the database if saved.

Modifier and Type:
init

Declaration: Swift

override init(withParseObject object: PFObject)

Parameters:

- withParseObject - The Parse object to tie this model to the Parse database.

10.42.1.c displayName ( )

String containing a user's display name as defined in the UserProfileModel protocol.

Modifier and Type:
var displayName

Declaration: Swift

var displayName: String

Parameters:

- withParseObject - The Parse object to tie this model to the Parse database.

10.42.2 Generation

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

- ParseUserProfileModel

10.43 SettingsViewController Class Reference

Class SettingsViewController

public class SettingsViewController

Member Functions

- logoutButton ()
- onLogoutTapped ()

10.43.1 Method Summary

10.43.1.a logoutButton ( )

10.43.1.b onLogoutTapped ( )

10.43.2 Generation

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

- SettingsViewController

10.44 SignupViewController Class Reference

Class SignupViewController

public class SignupViewController
class SignupViewController: UIViewController, UITextFieldDelegate
Custom view controller class for handling user signups.

Custom UIViewController class for handling user signups. Processes requests, validates input, and passes data to backend for checking with the server, and properly handles both successful and unsuccessful signup requests.

Member Functions

- scrollView ()
- facebookButton ()
- twitterButton ()
- emailButton ()
- nameField ()
- emailField ()

- passwordField ()
- passwordConfirmField ()
- submitButton ()
- signupFormFields ()
- init ()
- deinit ()
- viewDidLoad ()
- viewDidAppear ()
- facebookButtonTapped ()
- twitterButtonTapped ()
- submitButtonTapped ()
- registerForKeyboardNotifications ()
- deregisterFromKeyboardNotifications ()
- adjustForKeyboard ()
- textFieldShouldReturn ()
- submitForm ()
- unwindIfLoggedIn ()
- rewindToWelcomeView ()

10.44.1 Method Summary

10.44.1.a scrollView ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.b facebookButton ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.c twitterButton ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.d emailButton ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.e nameField ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.f emailField ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.g passwordField ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.h passwordConfirmField ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

```
class SignupViewController: UIViewController, UITextFieldDelegate
```

10.44.1.i submitButton ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.j signupFormFields ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.k init ( )

init(coder:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.l deinit ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.m viewDidLoad ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.n viewDidAppear ( )

viewDidAppear(_:)
Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.o facebookButtonTapped ( )

facebookButtonTapped(_:)
Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.p twitterButtonTapped ( )

twitterButtonTapped(_:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.q submitButtonTapped ( )

submitButtonTapped(_:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.r registerForKeyboardNotifications ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.s deregisterFromKeyboardNotifications ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.t adjustForKeyboard ( )

adjustForKeyboard(_:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.u textFieldShouldReturn ( )

textFieldShouldReturn(_:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.v submitForm ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.w unwindIfLoggedIn ( )

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.1.x rewindToWelcomeView ( )

rewindToWelcomeView(_:)

Modifier and Type:
class SignupViewController:

Declaration: Swift

class SignupViewController: UIViewController, UITextFieldDelegate

10.44.2 Generation

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

- ParseUserProfileModel

10.45 UserModel Class Reference

Class UserModel

public class UserModel
class UserModel: BaseModel, UserModel
This class extends the BaseModel class and implements the UserModel protocol and is the class to access the current user's information from

Member Functions

- profile ()
- init ()
- username ()
- emailVerified ()
- email ()
- phone ()
- createFromSignUp ()

10.45.1 Method Summary

10.45.1.a profile ( )

The model corresponding to this user's public profile.

Modifier and Type:
init

Declaration: Swift

let profile: UserProfileModel

10.45.1.b init ( )

init(withUser:profile:)
Class constructor. Initializes the instance from a PFObject.

Modifier and Type:
init

Declaration: Swift

init(withUser user: PFUser, profile: UserProfileModel? = nil)

Parameters:

- withUser - The user to tie this model to the database.
- profile - An optional UserProfileModel to connect to this user. If no value is or nil is provided, this constructor will attempt to get the profile from the withUser model.

10.45.1.c username ( )

String containing the current user's username as defined in the UserModel protocol.

Modifier and Type:
var username:

Declaration: Swift

var username: String

10.45.1.d emailVerified ( )

Boolean to store if the user has verified their email with as defined by the UserModel protocol.

Modifier and Type:
var emailVerified:

Declaration: Swift

var emailVerified: Bool

Return Value:
true if their email has been verified, false if their email has not been verified.

10.45.1.e email ( )

String containing the current users email as defined in the UserModel protocol.

Modifier and Type:
var emailVerified:

Declaration: Swift

var email: String

10.45.1.f createFromSignUp ( )

createFromSignUp(_:password:)

Main function for creating a new user. Automatically creates a corresponding UserProfileModel and returns both in a tuple.

Modifier and Type:
var phone:

Declaration: Swift

var phone: String

10.45.1.g createFromSignUp ( )

createFromSignUp(_:password:)

Main function for creating a new user. Automatically creates a corresponding UserProfileModel and returns both in a tuple.

Modifier and Type:
var phone:

Declaration:
Swift

var phone: String

Parameters:

- username - The new user's username.
- password - The new user's password.

Declaration:
A tuple containing the newly created UserModel object and its corresponding UserProfileModel.

10.45.2 Generation

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

- UserModel

s

10.46 UserProfileModel Class Reference

Class UserProfileModel

public class UserProfileModel
class UserProfileModel: BaseModel, UserModel
This class extends the BaseModel class and implements the UserProfileModel protocol and is the class to access a user's public profile information from .

Member Functions

- init ()
- init ()
- displayName ()

10.46.1 Method Summary

10.46.1.a init ( )

Default class contructor. Creates a new entry in the database if saved.

Modifier and Type:
init

Declaration: Swift

```
init()
```

10.46.1.b init ( )

init(withObject:)

Default class contructor. Creates a new entry in the database if saved.

Modifier and Type:
init

Declaration: Swift

```
override init(withObject object: PFObject)
```

Parameters:

- **withObject - The object to tie this model to the database.**

10.46.1.c displayName ( )

String containing a user's display name as defined in the UserProfileModel protocol.

Modifier and Type:
var displayName

Declaration: Swift

```
var displayName: String
```

Parameters:

- **withObject - The object to tie this model to the database.**

10.46.2 Generation

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

- **UserProfileModel**

10.47 Waypoint Struct Reference

Struct Waypoint

Struct Waypoint

class Waypoint: BaseModelI
Object to store group waypoints location and message

Member Functions

- waypointId ()
- longitude ()
- latitude ()
- message ()

10.47.1 Method Summary

10.47.1.a waypointId ( )

Waypoint's unique id to be generated on creation

Modifier and Type:
int

Declaration: Swift

var waypointId: Int

10.47.1.b longitude ( )

Waypoint longitude

Modifier and Type:
double

Declaration: Swift

var longitude: Double

10.47.1.c latitude ( )

Waypoint longitude

Modifier and Type:
double

Declaration: Swift

var latitude: Double

10.47.1.d message ( )

Message to display at location

Modifier and Type:
String

Declaration: Swift

var message: String

10.47.2 Generation

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

- **WayPoint**

10.48 Maint Class Reference

Class main

public class main

Cloud functions that allow for background functions with Parse.

Member Functions

- **fetchGroupUpdates ()**
- **joinGroup ()**
- **leaveGroup ()**
- **fetchNotifications ()**

10.48.1 Method Detail

10.48.1.a fetchGroupUpdates ( )

Parse.Cloud.define('fetchGroupUpdates', function(request, response)

This function allows for group updates.

Parameters:

- **group - current group of the user.**
- **userProfile - current user calling the function**
- **timestamp - current timestamp**

Returns: **changes in the group status.**

10.48.1.b joinGroup ( )

parse.Cloud.define('joinGroup', function(request, response)

This function allows for an user to join a group.

Parameters:

- **group - current group of the user.**
- **userProfile - current user calling the function**

Returns: **user join group and group information.**

10.48.1.c leaveGroup ( )

parse.Cloud.define('leaveGroup', function(request, response)

This function allows for an user to leave a group.

Parameters:

- group - current group of the user.
- userProfile - current user calling the function

Returns: removes the user from group information

10.48.1.d fetchNotifications ( )

parse.Cloud.define('lfetchNotifications', function(request, response)

This function fetches notification ipdates for the group.

Returns: new notifications for current group

10.48.2 Generation

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

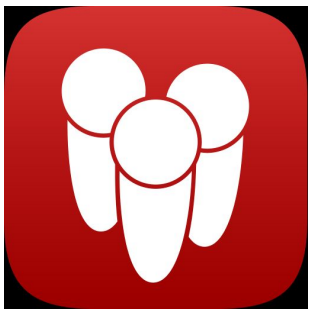
- main

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 supervison 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

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

9 WARRANTIES

The BowTaps Team represents and warrants to Sponsor that:

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

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

10 INDEMNITY

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

11 INDEPENDENT CONTRACTOR

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

12 TERM AND TERMINATION

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

13 GENERAL

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

14 SIGNATURES

10 / 6 / 2015

Charles Bonn — Date

10 / 6 / 2015

Evan Hammer — Date

10 / 6 / 2015

Joseph Mowry — Date

10 / 6 / 2015

Daniel Andrus — Date

10 / 6 / 2015

Johnathan Ackerman — Date

10 / 6 / 2015

Brian Butterfeild — Date

A

Product Description

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

1 GPS Features

1.1 Group Members

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

1.2 Suggestions

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

2 Group Messaging

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

3 Group Manangement Features

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

4 Parse Features

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

B

Sprint Reports

1 Sprint Report #1

Sprint Report #1

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Customer Overview

Customer Description

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

Customer Problem

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

Customer

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

Project Overview

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

Phase 1

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

Project Environment

Project Boundaries

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

Project Context

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

Deliverables

Phase 1

Deliverables will be UX design, database design and implementation.

Backlog

Phase 1

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

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

Sprint Report

Work for this sprint included:

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

2 Sprint Report #2

Sprint Report #2

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Work Summary

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

Backlog

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

1. User Model
2. Communication Layer
3. Link back-end and front end

- Implement Cloud code
- Business Plan

Successes

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

Issues and Changes

Some issues that have been ran into have been

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

Team Details

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

3 Sprint Report #3

Sprint Report #3

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control Group Management Mobile Application

Company

Bowtaps

Work Summary

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

Backlog

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

Success

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

Issues and Changes

Some issues that have been ran into have been

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

Team Details

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

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

4 Sprint Report Winter Sprint

Winter Sprint Report

Team Overview

Name

Crowd Control

Members

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

Project Title

CrowdControl - Group Management Mobile Application

Company

Bowtaps

Deliverables

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

(a) Leave button implemented
(b) Tested adding/removing users from groups

- Misc/Transitional

1. Further documented Android code to prepare for team merge
2. Android code review with iOS team, to prepare for team merge

Remaining Backlog

Here are the incomplete items/features for this sprint:

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

Successes

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

Issues and Changes

Some issues that we encountered include:

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

        * Encountered NullPointerException in the UserModel model. Had to change the structure to use an application global variable.
    - Changes
        * Further development on Settings is now added to the backlog
        * Sign out functionality is now added to the backlog
        * Leave Group functionality is now added to the backlog

- iOS
    - Issues
        * Unexpected complications with database design
        * Layout complications
        * Issues with the underlying data models
        * Parallel programming complications

- Misc/Transitional
    - iOS development will be postponed, in favor of an Android prototype. This is to ensure that Android will meet expectations for the design fair.
    - Team communication and long-distance coordination was difficult.
    - Holidays and vacations imepeded our ability to be productive.

Team Details

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

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

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

5 Sprint Report #4

Sprint Report #4

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control - Group Management Mobile Application

Company

Bowtaps

Backlog

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

Week 1

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

Week 2

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

Week 3

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

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

Deliverables

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

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

5. Check for Existing Email

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

Issues and Changes

Some issues that we encountered include:

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

Remaining Backlog

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

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

Team Details

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

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

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

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

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

6 Sprint Report #5

Sprint Report #5

Team Overview

Name

Crowd Control

Members

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

Project Title

Crowd Control - Group Management Mobile Application

Company

Bowtaps

Backlog

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

Week 1

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

Week 2

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

Week 3

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

Deliverables

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

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

Issues and Changes

Some issues that we encountered include:

- Android
  - Issues
    * Another bug came up in messaging which took precious time from other parts of the sprint
  - Changes
    * Many code related things were pushed to later in the sprint
- Server (cloud code)
  - Issues
    * Unable to do proper stress tests
    * Unrepeatable errors popped up (could have been a network erro r)
  - 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 Ubunto)
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 an iOS and Android game called *Trouble with Robots*. I also worked on a cross-platform desktop application that interacted via USB with a handheld game cartridge reader and writer that allows users to create and play Neo Geo Pocket and WonderSwan games on their handheld game devices.

SDSMT PROGRAMMING TEAM — 2014 - PRESENT

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

SERVER ADMINISTRATOR, PROGRAMMER — 2010 - PRESENT

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

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

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

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

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

SKILLS

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

EDUCATION

Black Hills State University, Spearfish, SD — 2010-2012
South Dakota School of Mines and Technology, Rapid City, SD — 2012-2016

PERSONAL INFORMATION

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

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

C. Nicholas Bonn

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

Education:

South Dakota School of Mines and Technology, Rapid City, SD — Anticipated Graduation: May 2016
Bachelor of Science in Computer Science — 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, LaTeX, SQL Server Management Studio

Organizations/Misc

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

Work Experience

Period	**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

2 ABET: Industrial Experience Reports

As a group we have attended the SD Engineering Accelerator. We have compeated 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 buisness 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 experence. What industry experence 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 pirior 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.