Project Anchronize

Complete Documentation

Yijia Chang, Jason Lin, Rohan Sharma, Xinghan Wang,
Tianlin Zhang, Glarence Zhao,

Advisor: Prof. Jeffrey Miller

Principles of Software Development

April 24, 2016

Preface

This is a proposal for CSCI 201 Group Project. This proposal provides the scope and context of the project to be undertaken. The intended audience of this document is the course faculty so that they can determine whether the project should be approved as proposed, approved with modifications, or not approved.

Table of Contents

1 - Proposal

Group Members

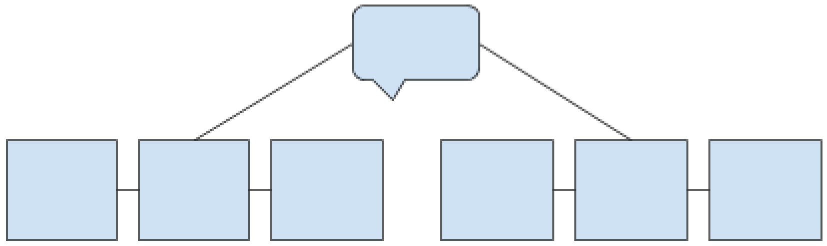
Name	Email	Phone Number	Section number
Jason Lin	jasonjli@usc.edu	213-587-0447	29928
Glarence Zhao	glarencz@usc.edu	415-769-8936	29928
Xinghan Wang	xinghanw@usc.edu	213-263-1339	30381
Yijia Chang	yijiacha@usc.edu	323-740-2803	29928
Tianlin Zhang	mr.tianlinzhang@gmail.com	213-270-3407	30393
Rohan Sharma	sharmaro@usc.edu	408-329-0464	30393

Meeting times:
- General Meeting 8PM-10PM Tuesday @ Leavey Library
- Status check 10AM Friday @ Slack "updates" channel

Project Overview:
The project is an Android application that marks events and incidents based on their physical location (geo proximity) on a map (Google Maps API). The information/events, which could be subscribed to by users & authorities, will be crowd-sourced by users who submit incidents they've seen and events they know of. Within a given event, there is a chat room that allows people who are interested to chat with other potential subscribers or the host/original poster of the event. The app will also allow users to filter the events based on the type/category by #Genre, similar to Slack. For example, #Bad could denote fire, robbery, car crash, other crimes; #Fun could be parties, free food, etc.

Class Requirements:
- Technology & APIs: Android 5.1.1+ (Build API 22), NoSQL, Everyblock, Houndify
- Multi-threading: Upon application launch, app will fetch & sync nearby events (represented by classes/subclasses) asynchronously, while rendering main activity UI to the user. It'll also initiate a connection to the chat log database.
- GUI: The GUI will be developed using XML on Android Studio.
- Networking: Chat rooms for each event. Discover! allows for P2P communication between client devices nearby. A graph will be used to store events and users.

Team Structure



2 - High Level Requirements

Rationale: According to IEEE 830, good software requirements have the following attributes:
Correct, Unambiguous, Complete, Consistent, Ranked, Verifiable, Modifiable, Traceable

Requirement Priorities Legend

- **Class Req (6)** - Requirements from class
- **Mandatory (7)** - Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.
- **Desirable (5)** - Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.
- **Optional/Stretch (3)** - Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.

2.1 Product Functionality

#	Feature	Priority ()
1	User login functionality and a central server to authenticate	Class Req
2	Users can interact with the software as a guest but experience limited functionality compared to authenticated users ● Authenticated/Paying/Premium features ○ exclusive event notifications ○ Ability to chat with event host & other participants	Class Req
3	Users must be able to register so they can become an authenticated user.	Class Req
4	When a user registers, the data should be stored in a database on the server	Class Req
5	Program incorporate graphics - Android Studio XML	Class Req
6	Multithreaded and network functionality outside of chatting, i.e. fetching events data on launch	Class Req
7	Display all crowdsourced events on a map (Google Maps API)	Mandatory
8	Chat room that allows people who are interested to chat with other potential subscribers or the host/original poster of the event	Mandatory
9	User should be able to submit events - crowdsourcing information	Mandatory
10	User information, preference, and other data are stored in the form of an account (authenticated/paying or guest)	Mandatory

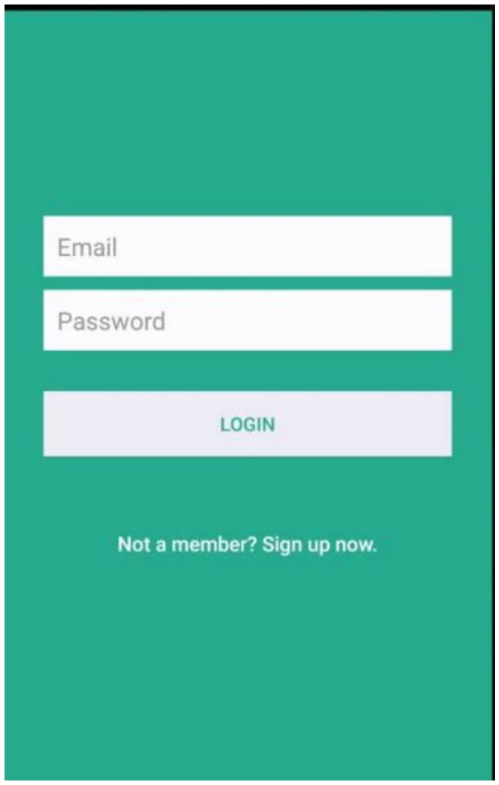
11	User could choose to login with Facebook, and if they do, he/she has the option of syncing his/her FB events and adding them to the app's database of events (Facebook Graph API: https://developers.facebook.com/docs/graph-api/reference/event)	Mandatory
12	Allow users to filter the events based on the type/category by #Genre For example: ● #Bad = fire, robbery, car crash, other crimes ● #Fun = parties, free food	Mandatory
13	User preference that could be configured - Default values for criteria in filters	Mandatory
14	Each event should be supported by pictures, GIFs, and other multimedia	Desirable
15	Ability to highlight/bookmark/favorite events in the app	Desirable
16	User credibility rating - honor/upvote system or reports. This allows credible users/event hosts to more easily contribute while censoring spamming/abusive users	Desirable
17	Random event generator - activated by shaking phone A large number of similar events of the same category within a set radius will be grouped into one pin. Each individual event submission could be viewed when clicked.	Desirable
18	Check-in to places and earn points like Snapchat (outside concepts)	Desirable
19	A large number of similar events of the same category within a set radius will be grouped into one pin. Each individual event submission could be viewed when clicked.	Optional
20	Voice commands with Houndify API http://www.soundhound.com/houndify	Optional
21	List of places with the most frequent event occurrences	Optional

2.2 - High Level Walkthrough

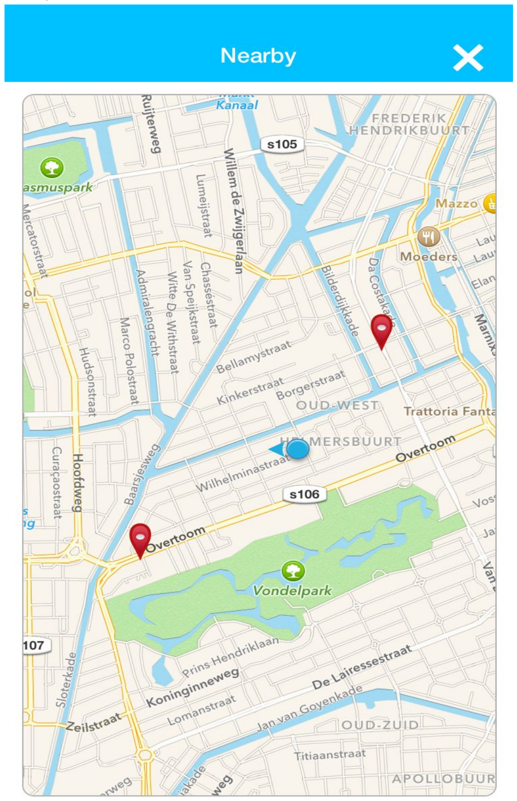
- When a user launches the app, he/she will be welcomed with a splash screen followed by a login screen with 3 options: Sign In/Sign Up or Sign In as Guest
- If the user chooses to Sign In as a guest, he/she would be able to see all of the events around, but is limited to opening chatrooms under each event or RSVPing to them.

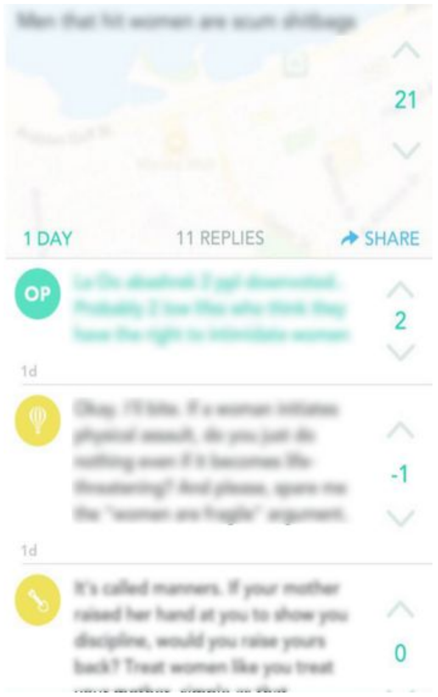
- OR if the user logs in successfully, the app will reveal a full screen sized map centered around the user's current location, mapping all relevant events around.
- To accommodate for the wait time of fetching data from the database (in background thread), the app will display a "What you've missed" activity that details the user of the most popular upcoming event nearby
- Once the background thread has finished syncing with the database, the button "Okay, got it" at the bottom of "What you've missed" activity will be enabled for the user to return to the mainscreen.
- Houndify's speech listener and the device's motion listener should be enabled to listen for the user's voice commands and shaking motion.
- On the mainscreen, there will be a search bar (center, top) and a hamburger menu on the top right that could be expanded into a card menu that pushes the current homescreen to the left. The search bar could be used to search AND submit events, even though the user could also use voice commands to perform similar tasks.
- The card menu will display a list of submenus including user preference configuration, event bookmarks, nearby users (if logged in), news feed, trend lines, etc.
- Under each event, logged in users are able to view other participants and RSVP to the event
    - There is a news feed stream where you can see images and videos posted by users
    - A User may check in to an event and receive points for attendance
    - By attending an event, if the event is set as private by the host (or not open to public), the user is open to communicate in the chatroom
    - Event host contact information is listed
    - If a user wants to remember this event, he or she can bookmark it in a list
- Along a vertical menu, there would also be difference criteria users can use to filter the displayed events. One possibility is the # system specified in the requirements above. The criteria displayed should be by popularity/usage by default or a customized selection if the user set his/her preferences in the app settings.
- Guest Features:
    - Can look at most popular recipes (based on likes) for the day/week
    - Can't do anything else unless they sign up

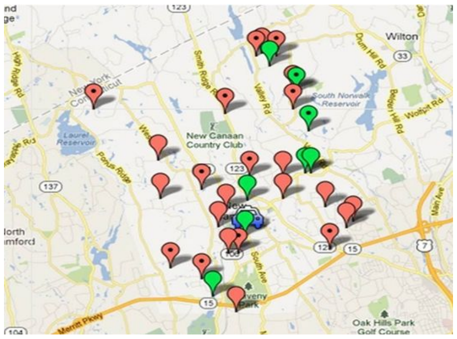
3 - Technical Specifications

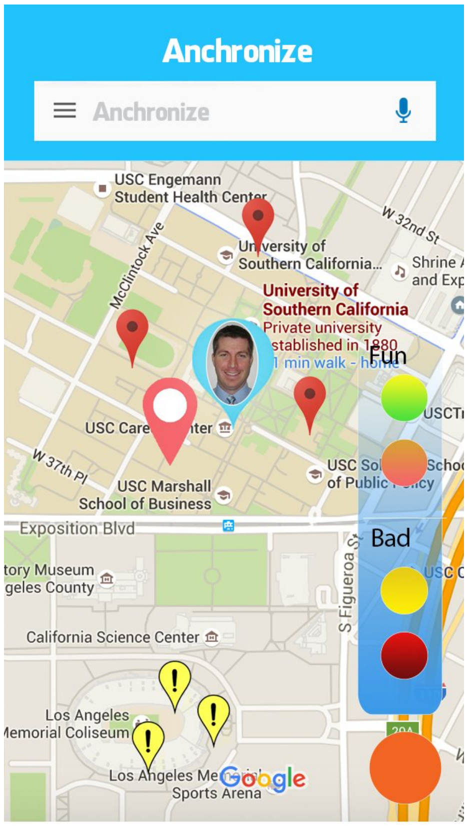
#	Hours	Feature	Priority ()	Assigned to
1	2-3 hrs	 User login functionality (UI) and a central server to authenticate ● When the app runs for the first time, a modular Activity for users' login should pop up. ● This login activity should at least have following elements ○ A unique account email address ○ Password field ○ A link(Button) for user without accounts to sign up ■ This will bring up the register activity described below(feature 3-4) ○ A link(Button) for user to skip this part and use app with limited functionality ● When users click login button to authenticate, a requested should be sent to the central server(Firebase) to authenticated. This will be done using Firebase API and JSON.	Class Req	Xinghan

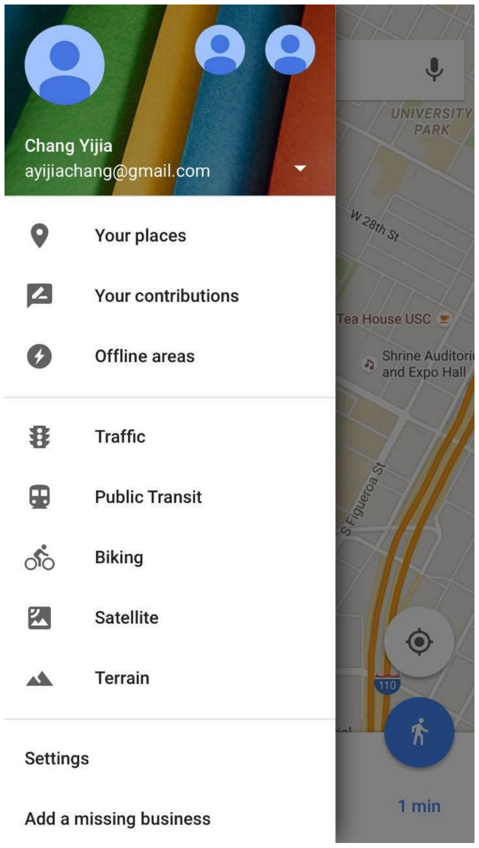
		● (Optional) User may login with facebook or google account.		
2	2-3 hrs	Users can interact with the software as a guest but experience limited functionality compared to authenticated users ● Authenticated/Paying/Premium features ● Exclusive event notifications ● Ability to chat with event host & other participants ● Guests can essentially only view public events	Class Req	Glarence
3	1-2 hrs	Users must be able to register so they can become an authenticated user. ● In the login activity, when register button is pressed, a new user registration activity should be added to activity stack. ● This activity should be similar to login activity, except that login button is replaced with register button. ● There should also be a button next to email field to querying Firebase if such email address is already being used.	Class Req	Xinghan
4	2-3hrs	When a user registers, the data should be stored in a database on the server When user press “if taken” button, a request should be fetched to Firebase to see if account name already exists in database. - Android app should buffer the new account info once the user hits “submit”, and attempts to write to Firebase the new account info - username, password, email - If Firebase returns a successful registration, user should be logged in automatically with his/her new account and directed to main activity (map view)	Class Req	Xinghan Jason
5	5 hrs	Program will incorporate graphics using Android Studio XML ● When the user opens up the list menu for all his saved eventsthe events near him, he will see a linearLayout with a single column filled with all the events ○ The user can then scroll horizontally	Class Req	Rohan

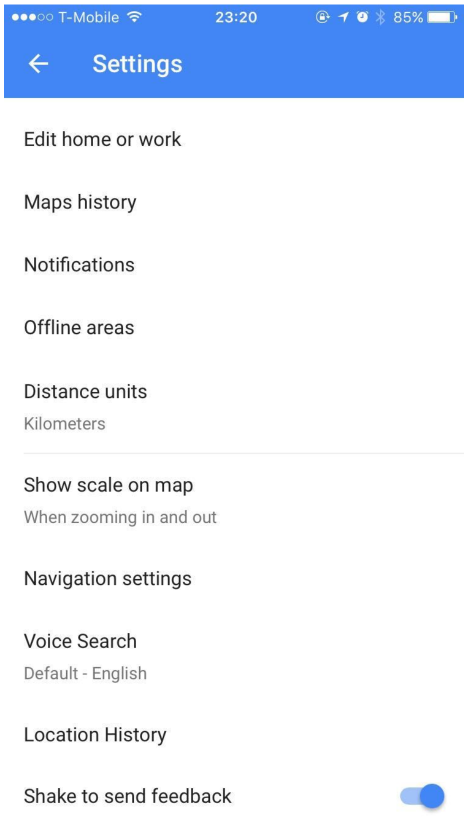
		through the events ● The main screen of the app will feature a map of the specified radius neighborhood of the user with events shown on the map		
6	5-7hrs	Multithreaded and network functionality outside of chatting, i.e. fetching events data on launch, ● When a user logs in (including as guest), a background thread will be initiated to fetch for events data ● Another thread will be initiated to populate the hidden side panel in the background ● A thread will be used to fetch nearby users	Class Req	Jason
7	6 hrs	Display all crowdsourced events on a map (Google Maps API)  Nearby FREDERIK HENDRIKBUURT OUD-WEST Overtoom Vondelpark OUD-ZUID APOLLOBUURT ● The Map will be supported by the Google Maps Android API ○ https://developers.google.com/maps/documentation/android-api/ ● Events will be marked with custom markers, info windows and polylines that are provided by the API based on geo-location. ● The map will be populated with nearby events	Mandatory	Danny

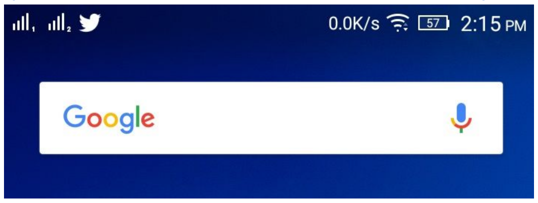
		that fit the default criteria set for the specific user. If the user did not set any default criteria in the app settings, all nearby events will be displayed. ● The map filter/criteria will determine the type of events that are shown. More details on this specific function in later parts of this document in feature 16.		
8	4 hrs	Chat room that allows people who are interested to chat with other potential subscribers or the host/original poster of the event ● Guest accounts will not be allowed to communicate with authorized users ○ But, they will be allowed to view events and chats between the authorized users  21 1 DAY 11 REPLIES SHARE OP 2 1d -1 1d 0	Mandatory	Rohan Yijia
9	2-3 hrs	User should be able to submit events - crowdsourcing information ● Before events are submitted, the users will have to choose a category the events fall under ○ Emergency, Free/For Sale/Social Event/Gathering etc. ● Additionally, users will be given the option to make their submitted events unique by allowing them to put hashtags in the description portion of their events	Mandatory	Rohan

		○ #Safe/#freeFood/#Fire etc. ● Events will show up as icons on the map near the user's location ● It will look similar to the following: 		
1 0	4 hrs	User information, preference, and other data are stored in the form of an account (authenticated/paying or guest) ● Using a NoSQL database, we can always modify the existing account's username and profile picture information and insert new value fields to each key, indexed by a unique user ID ● Backup plan: if we are to use SQL, we could have a table for user authentication, another table containing user preferences, another table with user/event subscription data… Relationships between the tables will be based on the common userID foreign key.	Mandator y	Jason
1 1	2-3 hrs	User could choose to login with Facebook, which will create a user on the database and set its profile picture.	Mandator y	Sean Jason
1 2	3 hrs	Allow users to filter the events based on the type/category by #Genre ● (The GUI and specific behavior of this function are detailed in later parts of the document at feature 16) ● The purpose of these filters is to create categorization so users can find relevant information quickly and effectively. ● Examples: ○ #Bad could denote fire, robbery, car crash, other crimes ○ #Fun could denote parties, free food	Mandator y	Danny

		○ #Polite could denote political demonstrations or police activities.. ● For the events class, a filter type would be a required variable value. An event cannot be submitted without having a set event filter type.		
13	5 hrs	 Anchronize Anchronize Fun Bad USC Engemann Student Health Center University of Southern California USC Marshall School of Business Exposition Blvd California Science Center Los Angeles Memorial Coliseum S Figueroa St W 32nd St W 37th Pl McClintock Ave Google A floating menu bar which incorporates a slidebar menu, a search text area and voice recognizer should always display at the top of the map. The default text in the search text box should be “Anchronize”. A filter button (orange) is positioned at the bottom right corner. When clicked, four extra buttons, representing filters for “fun” and “bad” events should appear right above the orange button. In actual implementation, the four filter buttons should be icons reflecting the nature of the events, instead of solid color in the mockup above. The filter buttons should work like radio buttons, only selected types of pins (events) should appear on the map.	Mandatory	Yijia

		 Chang Yijia ayijiachang@gmail.com Your places Your contributions Offline areas Traffic Public Transit Biking Satellite Terrain Settings Add a missing business UNIVERSITY PARK W 28th St Tea House USC Shrine Auditori and Expo Hall S Figueroa St 110 1 min When the sidebar menu (hamburger) icon is clicked, a sidebar navigation panel resembling that of the Google Map app should appear to the left, pushing the main activity to the right in a FrameLayout. It should contain at least login/logout functionality and display current user's profile picture.		

		 ●●●○○ T-Mobile 23:20 85% Settings Edit home or work Maps history Notifications Offline areas Distance units Kilometers Show scale on map When zooming in and out Navigation settings Voice Search Default - English Location History Shake to send feedback A “setting” menu item should be included in this sidebar navigation menu. When clicked, a panel like the one shown above should appear.		
1 5	2 hrs	Ability to highlight/bookmark/favorite events in the app ● Each event will have a star option, and once clicked, said event will be added to a list of bookmarked events that is in chronological order from the time it was added ●	Desirable	Glarence
1 7	1-2 hrs	Random event generator ● When in the map view (the default view after logging in, with no other panels, menu bar open), a user should be able to shake the device to enter the event detail of a random event.	Desirable	Yijia
1 9	5 hrs	A large number of similar events of the same category within a set radius will be grouped into one pin. Each individual event submission could be viewed when clicked. ● Events that are similar(essentially the same)	Optional	Danny Xinghan

		should be merged into a standard one. ○ For example, an Event named “fire” should be the same as another named “Fire” in the same street block. ○ The threshold, or criteria to determine the similarity of two events should be discussed in upcoming group meeting. ● When one place marker (red pin) is pressed, the map will zoom in appropriately and the “main” pin will split into the individual pins, designating distinct events. Google API supports this based on the mobile device’s screen size and the spread of the map markers.		
20	4 hrs	Voice commands with Houndify API ● http://www.soundhound.com/houndify  ● ● As seen above, a search bar always exists on the top of the screen. The microphone button to the right will be used to trigger Houndify, which listens to the user’s voice and process commands like “adding an event”, “view nearby”, etc.	Optional	Jason

4 - Detailed Design Document

Hardware/software Requirements
Android Mobile Device/Android Virtual Device
Android Mobile Phone:

Android 5.0.1 and above (API Version. 23+)
Screen Size: 5.0”
Screen Resolution: 1920x1080 (~441 ppi)
Minimum CPU Clock Speed: 1.0 GHz
Available Memory: 50MB
Device Sensors: Accelerometer, Gyro, Proximity
Connectivity: WiFi, GPS, 4G
Reference Device: OnePlus X
Tested On: Nexus 5, Galaxy S6 Edge, Moto X Pure Edition

Software Requirements

Android Developer Mode
Google Play Services
USB debugging Enabled
Allow installation of apps from unknown source
Permissions:

- android.permission.INTERNET
- android.permission.ACCESS_FINE_LOCATION
- android .permission.GET_ACCOUNTS
- android.permission.READ_PROFILE
- android.permission.READ_CONTACTS

Development:

- Java 7
- Android Studio 2.0
- Mac/Windows

API Requirements

Google Maps Android API 8.4+ (API Key required)
Firebase Android API 2.5.x+ (Key required)
Facebook Graph API
Houndify Android API (Key required)
Everyblock API*

Difficult algorithms

- Search events with hashtags
    - When each event is created, scan through the description field and parse all the hashtags.

        - We will make a HashMap<string: hashtagName, ArrayList<String>: eventIDs>
            - This way, we can loop through the map and see which events are associated with the hashtags
- Display all the events near the user
    - When user successfully login, the main activity will be loaded. There will be an asynchronize task addEvent(). This methods will loop through every single event and place a marker on the GoogleMap. Then the GoogleMap will handle scoping in camera perspective.
- What's trending (interactive map):
    - An interactive map (see "what's trending activity" mockup) whose X-axis is the time (from 1 in the morning to 24 midnight) and Y-axis the events that have the most posts on that time represented by dots. The user should be able to click on a dot and the event Detail activity for that event should show up.
    - Each post has a timestamp when it is created on Firebase server.
    - The program assigns each event a Hashmap with keys being the time (i.e., 8am, 9am) and values being number of posts of that time.
    - When the application starts up, the program sorts the events based on their number of posts at a specific time. Therefore we need run such sorting 24 times.(one for each o'clock) We use merge sort so the time complexity will be 24 nlog(n).
    - Lastly, we put the most popular event (event with most posts) onto the map.
- ChatWall(Multithreading)
    - Each user client will maintain a database Listener listening to any updates in Firebase(When users submit a comment). When a change is made to the database, new comments will show up in clients' comments wall.

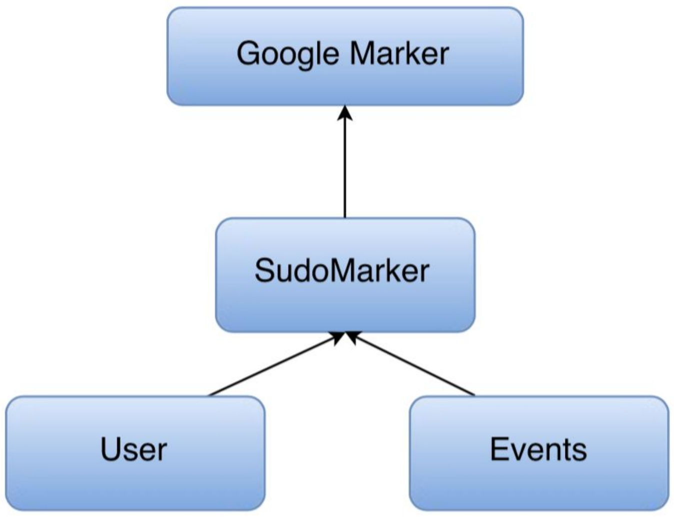
The Database Schema
NoSQL uses Dynamic Schema
Firebase stores users and events

- Flatten (Demoralize) data
- Storing One-to-Many Relations or Static Lists
- Maintaining Many-to-Many Relations
- Use Nested Data Sparingly

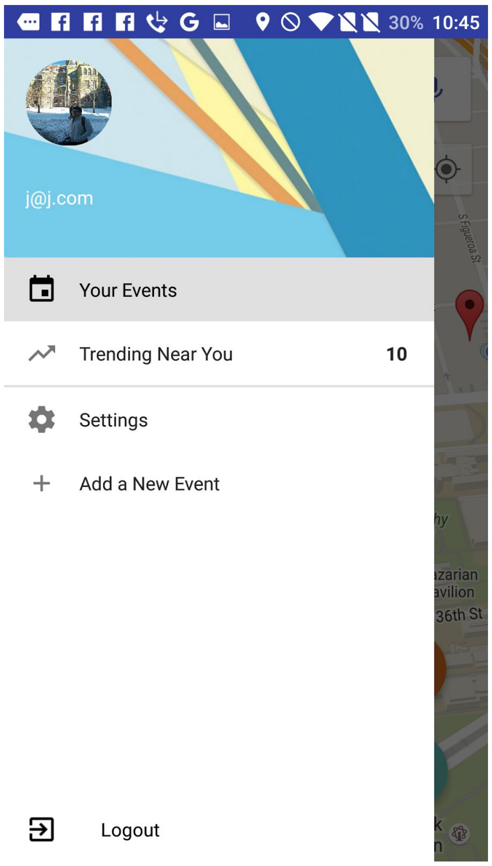
Reference for structuring data in NoSQL (JSON):
https://www.firebase.com/docs/web/guide/structuring-data.html

Example Code:
```
{
  user: {
    event: {
      title: "Rohan's House Party",
      latlng: "39.1212, 243,3232",
      isPublic: true,
      Description: “Example event held at Rohan’s house.”
    }
  }
}
```

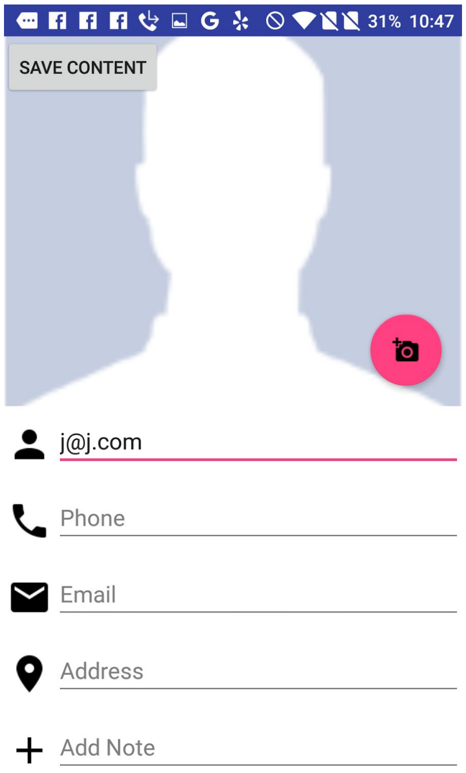
Inheritance Hierarchies(number denotes the access level hierarchy)



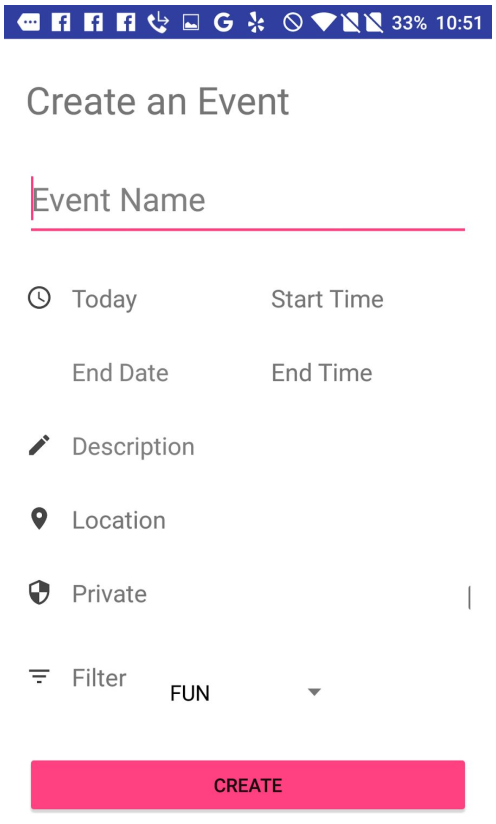
Exact GUI (incorporated alongside ER Diagrams)



Navigation Drawer (1)



User Account Page



Create an Event

Event Name

Today Start Time

End Date End Time

Description

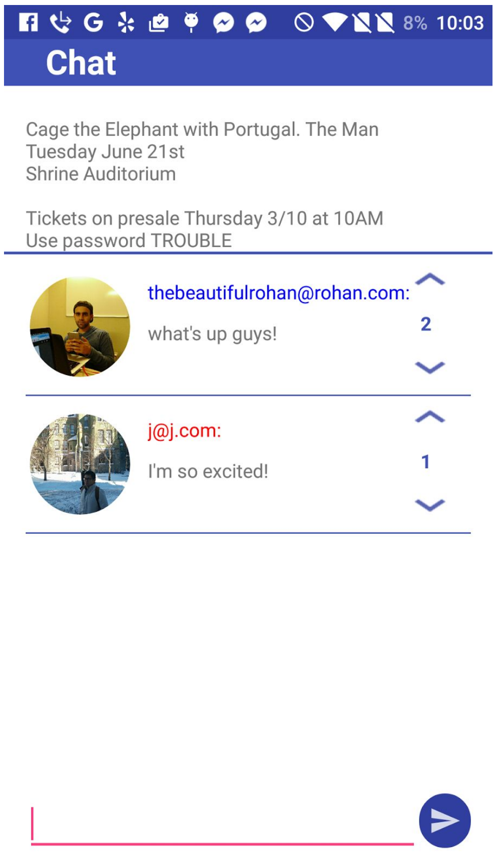
Location

Private

Filter FUN

CREATE

Adding Event Activity (3)



Chat

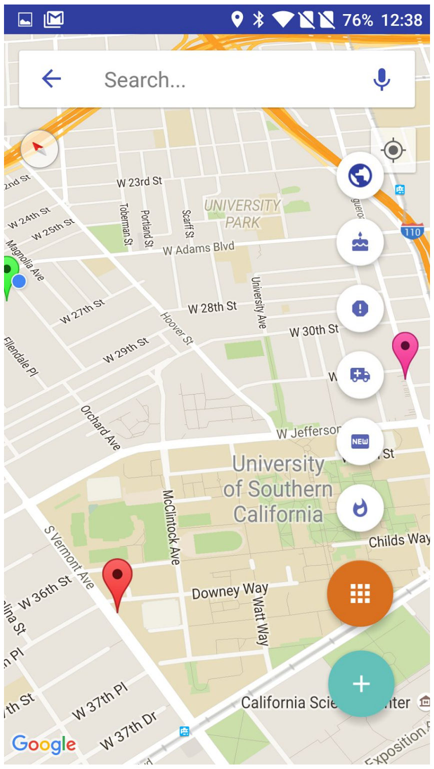
Cage the Elephant with Portugal. The Man
Tuesday June 21st
Shrine Auditorium

Tickets on presale Thursday 3/10 at 10AM
Use password TROUBLE

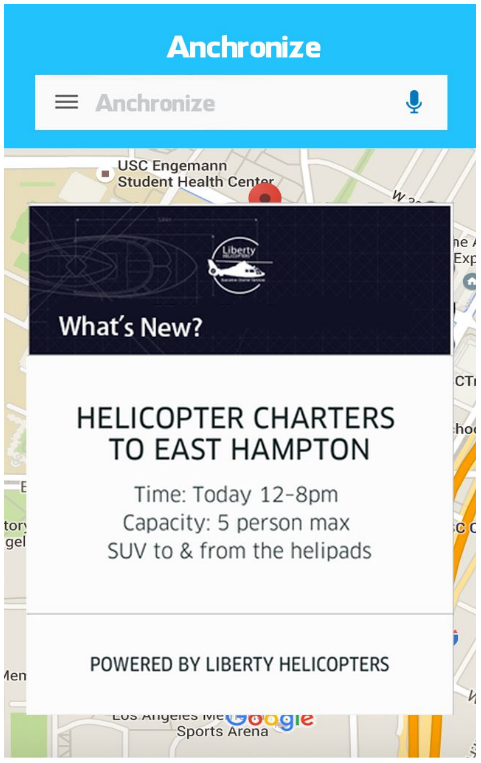
thebeautifulrohan@rohan.com:
what's up guys!
2

j@j.com:
I'm so excited!
1

Chat Room Activity (3)



Homepage Activity (1)



Anchronize

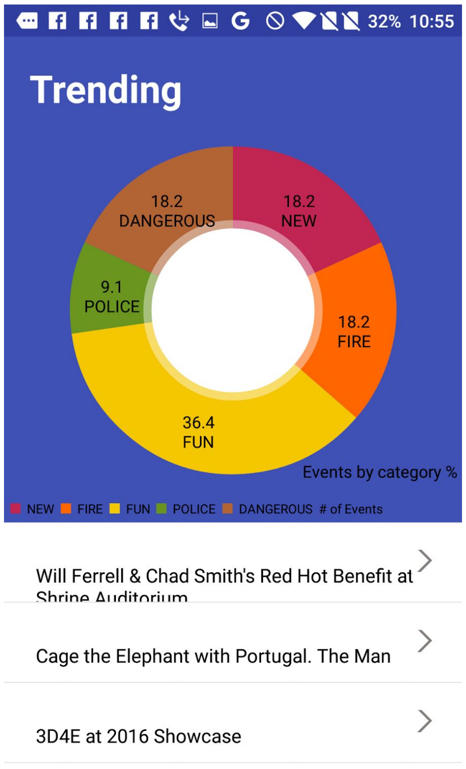
What's New?

HELICOPTER CHARTERS TO EAST HAMPTON

Time: Today 12-8pm
Capacity: 5 person max
SUV to & from the helipads

POWERED BY LIBERTY HELICOPTERS

New Trending Info Fragment (2)


32% 10:55

Trending

Events by category %

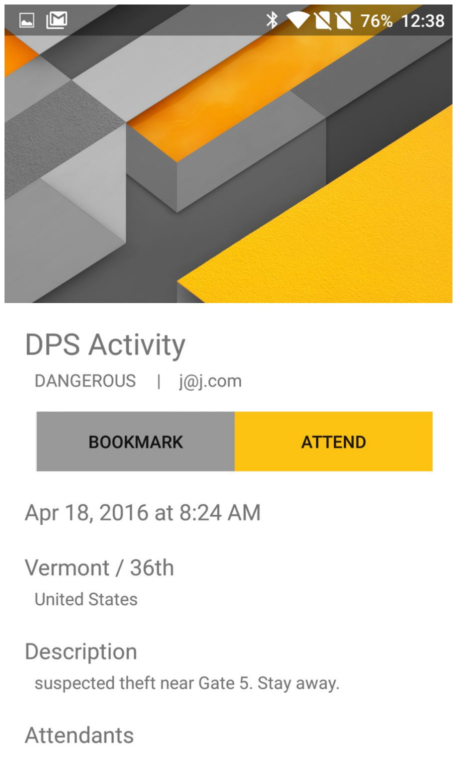
Category	%
DANGEROUS	18.2
NEW	18.2
POLICE	9.1
FIRE	18.2
FUN	36.4

🟥 NEW 🟧 FIRE 🟨 FUN 🟩 POLICE 🟫 DANGEROUS # of Events

Will Ferrell & Chad Smith's Red Hot Benefit at Shrine Auditorium

Cage the Elephant with Portugal. The Man

3D4E at 2016 Showcase


76% 12:38

DPS Activity

DANGEROUS | j@j.com

BOOKMARK ATTEND

Apr 18, 2016 at 8:24 AM

Vermont / 36th

United States

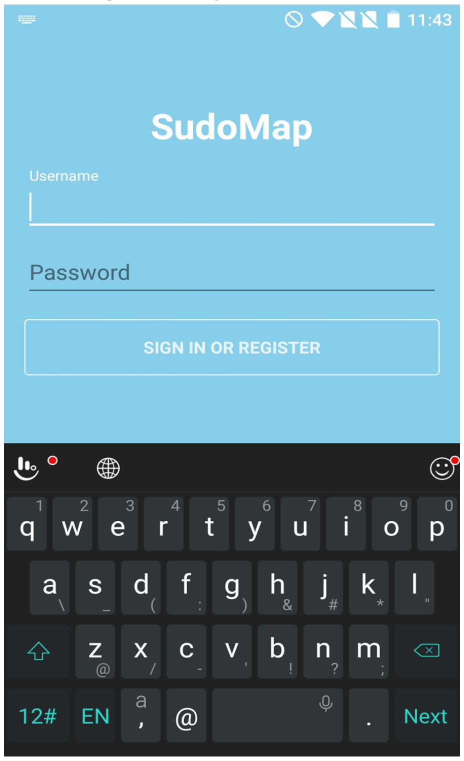
Description

suspected theft near Gate 5. Stay away.

Attendants

Trending Activity (2)

Main Event Activity (2)


11:43

SudoMap

Username

Password

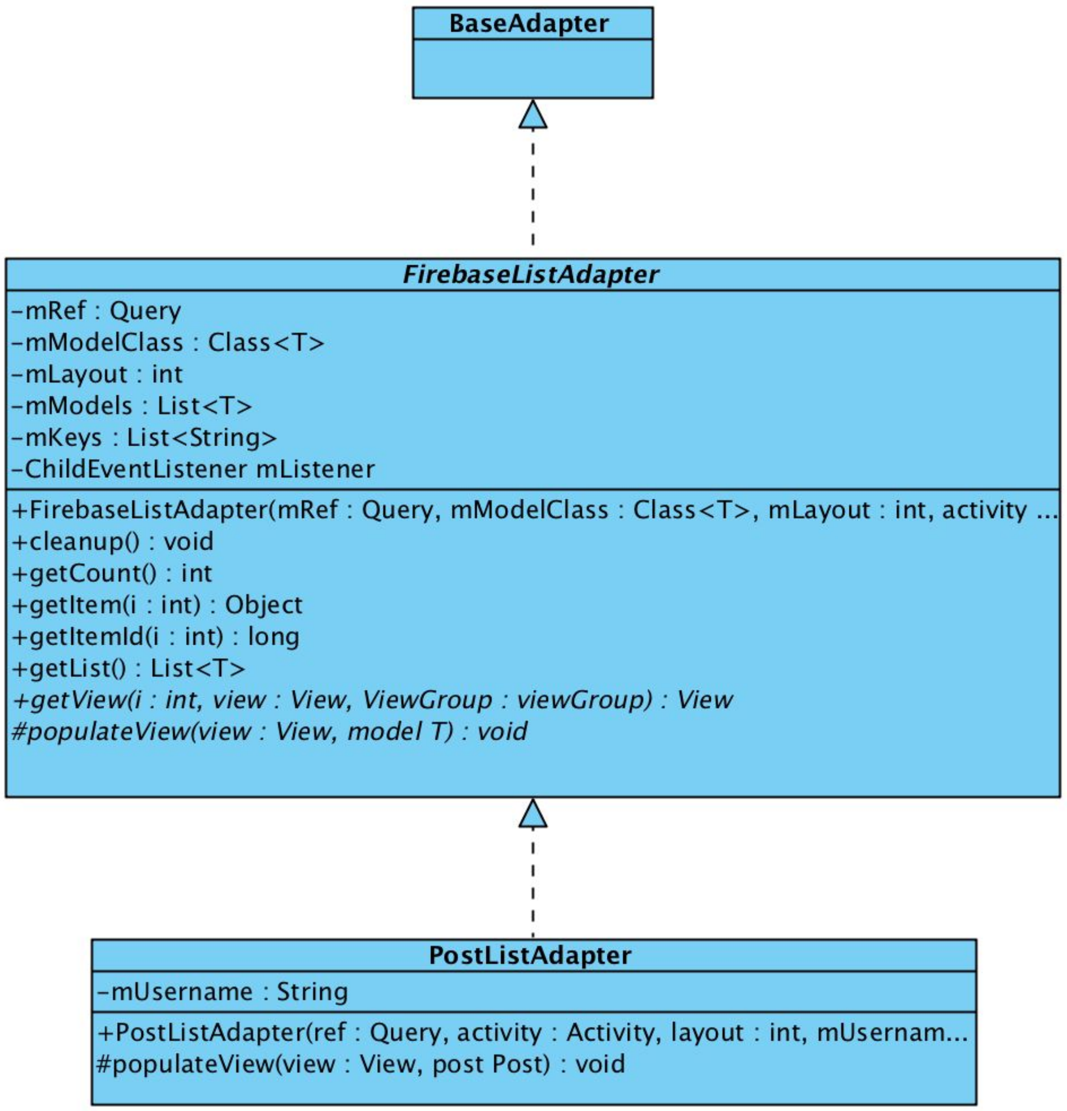
SIGN IN OR REGISTER

User Log-In Page (0)

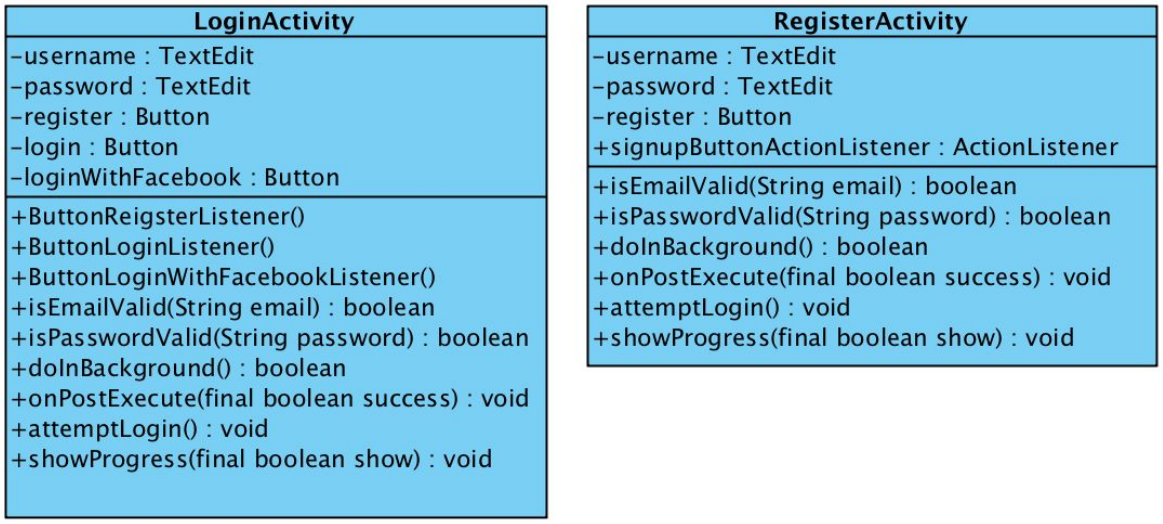
Class Diagrams

<u>Post:</u>

Adapters required for Firebase interface:


BaseAdapter
FirebaseListAdapter
-mRef : Query
-mModelClass : Class<T>
-mLayout : int
-mModels : List<T>
-mKeys : List<String>
-ChildEventListener mListener
+FirebaseListAdapter(mRef : Query, mModelClass : Class<T>, mLayout : int, activity ...
+cleanup() : void
+getCount() : int
+getItem(i : int) : Object
+getItemId(i : int) : long
+getList() : List<T>
+getView(i : int, view : View, ViewGroup : viewGroup) : View
#populateView(view : View, model T) : void
PostListAdapter
-mUsername : String
+PostListAdapter(ref : Query, activity : Activity, layout : int, mUsernam...
#populateView(view : View, post Post) : void

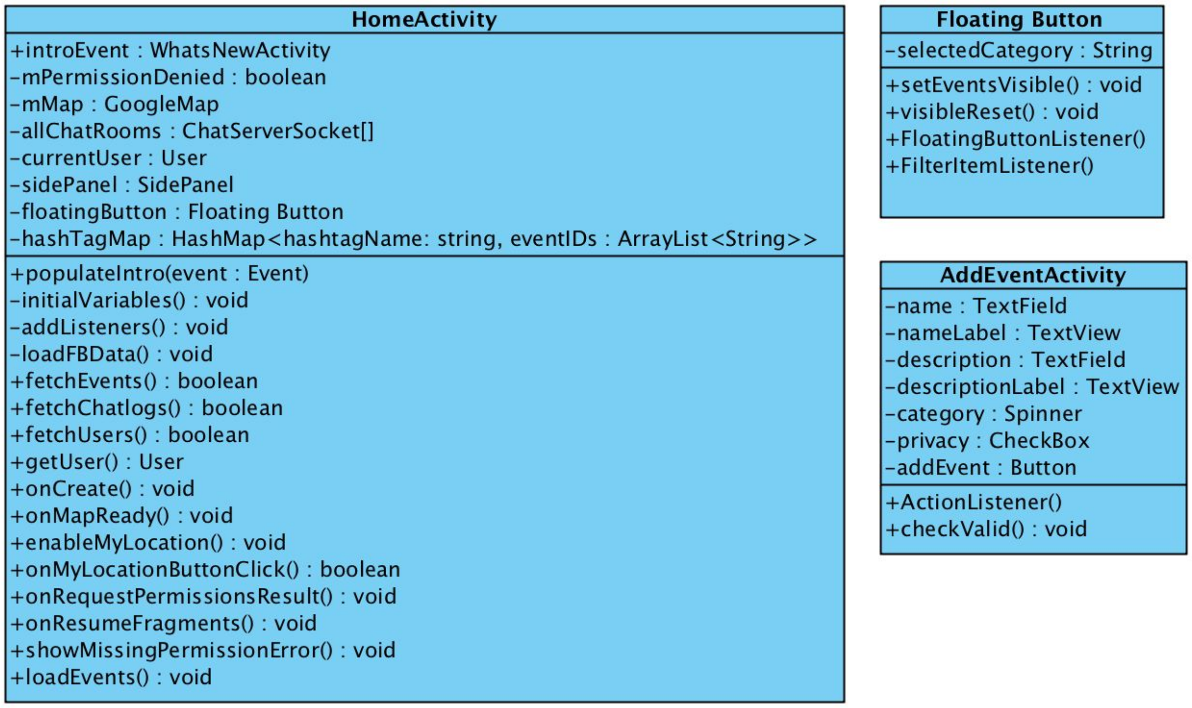
Authentication/Login:



```
LoginActivity
-username : TextEdit
-password : TextEdit
-register : Button
-login : Button
-loginWithFacebook : Button
+ButtonReigsterListener()
+ButtonLoginListener()
+ButtonLoginWithFacebookListener()
+isEmailValid(String email) : boolean
+isPasswordValid(String password) : boolean
+doInBackground() : boolean
+onPostExecute(final boolean success) : void
+attemptLogin() : void
+showProgress(final boolean show) : void
```

```
RegisterActivity
-username : TextEdit
-password : TextEdit
-register : Button
+signupButtonActionListener : ActionListener
+isEmailValid(String email) : boolean
+isPasswordValid(String password) : boolean
+doInBackground() : boolean
+onPostExecute(final boolean success) : void
+attemptLogin() : void
+showProgress(final boolean show) : void
```

Home Activity

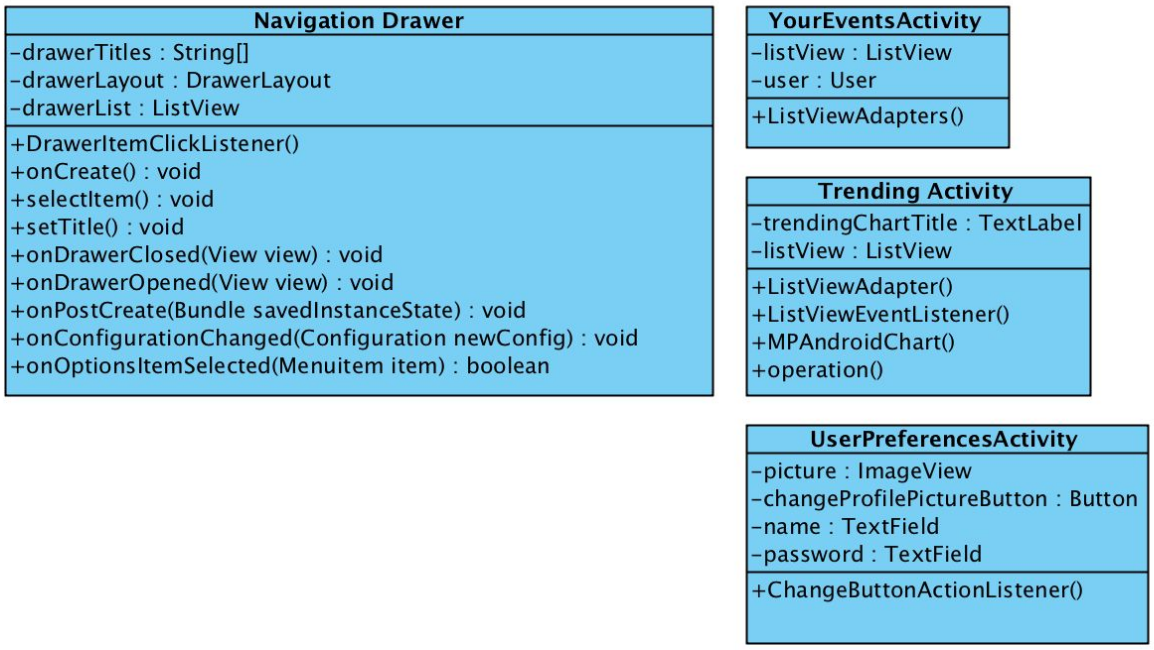


```
HomeActivity
+introEvent : WhatsNewActivity
-mPermissionDenied : boolean
-mMap : GoogleMap
-allChatRooms : ChatServerSocket[]
-currentUser : User
-sidePanel : SidePanel
-floatingButton : Floating Button
-hashTagMap : HashMap<hashtagName: string, eventIDs : ArrayList<String>>
+populateIntro(event : Event)
-initialVariables() : void
-addListeners() : void
-loadFBData() : void
+fetchEvents() : boolean
+fetchChatlogs() : boolean
+fetchUsers() : boolean
+getUser() : User
+onCreate() : void
+onMapReady() : void
+enableMyLocation() : void
+onMyLocationButtonClick() : boolean
+onRequestPermissionsResult() : void
+onResumeFragments() : void
+showMissingPermissionError() : void
+loadEvents() : void
```

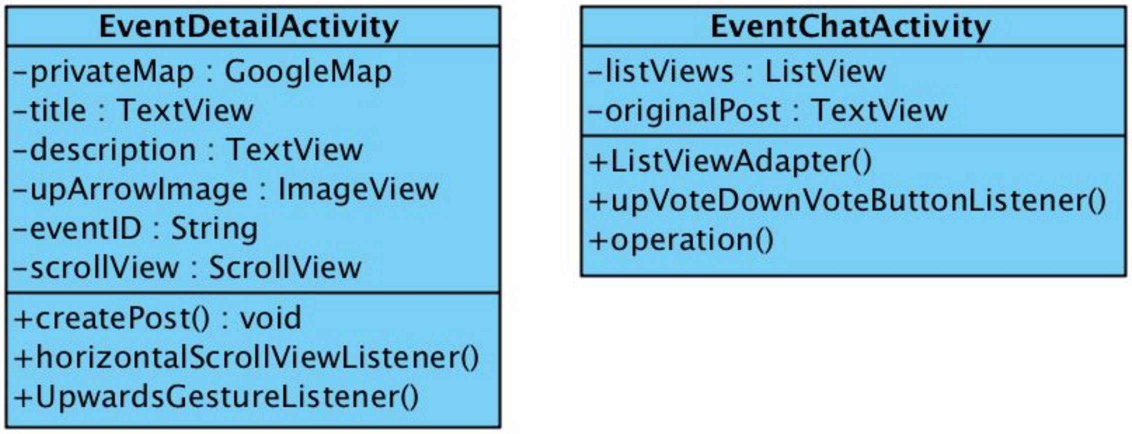
```
Floating Button
-selectedCategory : String
+setEventsVisible() : void
+visibleReset() : void
+FloatingButtonListener()
+FilterItemListener()
```

```
AddEventActivity
-name : TextField
-nameLabel : TextView
-description : TextField
-descriptionLabel : TextView
-category : Spinner
-privacy : CheckBox
-addEvent : Button
+ActionListener()
+checkValid() : void
```

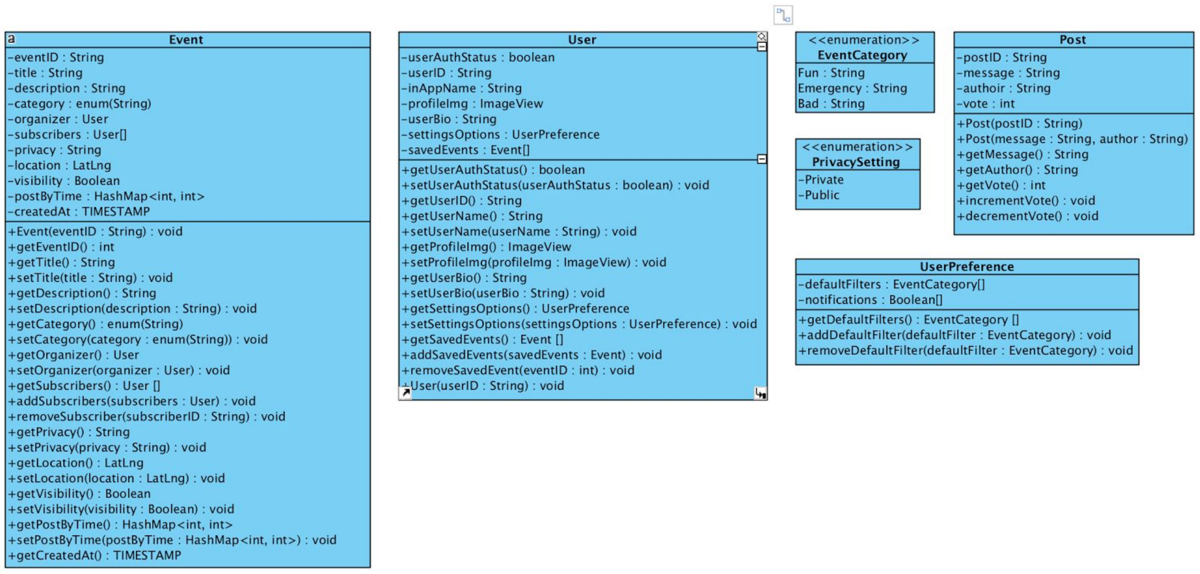
Navigation Drawer/Floating Button:



Event Activity:



User/Event/Enums(Pure Java objects):


Event
-eventID : String
-title : String
-description : String
-category : enum(String)
-organizer : User
-subscribers : User[]
-privacy : String
-location : LatLng
-visibility : Boolean
-postByTime : HashMap<int, int>
-createdAt : TIMESTAMP
+Event(eventID : String) : void
+getEventID() : int
+getTitle() : String
+setTitle(title : String) : void
+getDescription() : String
+setDescription(description : String) : void
+getCategory() : enum(String)
+setCategory(category : enum(String)) : void
+getOrganizer() : User
+setOrganizer(organizer : User) : void
+getSubscribers() : User []
+addSubscribers(subscribers : User) : void
+removeSubscriber(subscriberID : String) : void
+getPrivacy() : String
+setPrivacy(privacy : String) : void
+getLocation() : LatLng
+setLocation(location : LatLng) : void
+getVisibility() : Boolean
+setVisibility(visibility : Boolean) : void
+getPostByTime() : HashMap<int, int>
+setPostByTime(postByTime : HashMap<int, int>) : void
+getCreatedAt() : TIMESTAMP
User
-userAuthStatus : boolean
-userID : String
-inAppName : String
-profileImg : ImageView
-userBio : String
-settingsOptions : UserPreference
-savedEvents : Event[]
+getUserAuthStatus() : boolean
+setUserAuthStatus(userAuthStatus : boolean) : void
+getUserID() : String
+getUserName() : String
+setUserName(userName : String) : void
+getProfileImg() : ImageView
+setProfileImg(profileImg : ImageView) : void
+getUserBio() : String
+setUserBio(userBio : String) : void
+getSettingsOptions() : UserPreference
+setSettingsOptions(settingsOptions : UserPreference) : void
+getSavedEvents() : Event []
+addSavedEvents(savedEvents : Event) : void
+removeSavedEvent(eventID : int) : void
+User(userID : String) : void
<<enumeration>>
EventCategory
Fun : String
Emergency : String
Bad : String
<<enumeration>>
PrivacySetting
-Private
-Public
Post
-postID : String
-message : String
-authoir : String
-vote : int
+Post(postID : String)
+Post(message : String, author : String)
+getMessage() : String
+getAuthor() : String
+getVote() : int
+incrementVote() : void
+decrementVote() : void
UserPreference
-defaultFilters : EventCategory[]
-notifications : Boolean[]
+getDefaultFilters() : EventCategory []
+addDefaultFilter(defaultFilter : EventCategory) : void
+removeDefaultFilter(defaultFilter : EventCategory) : void

5 - Testing Document

Rationale

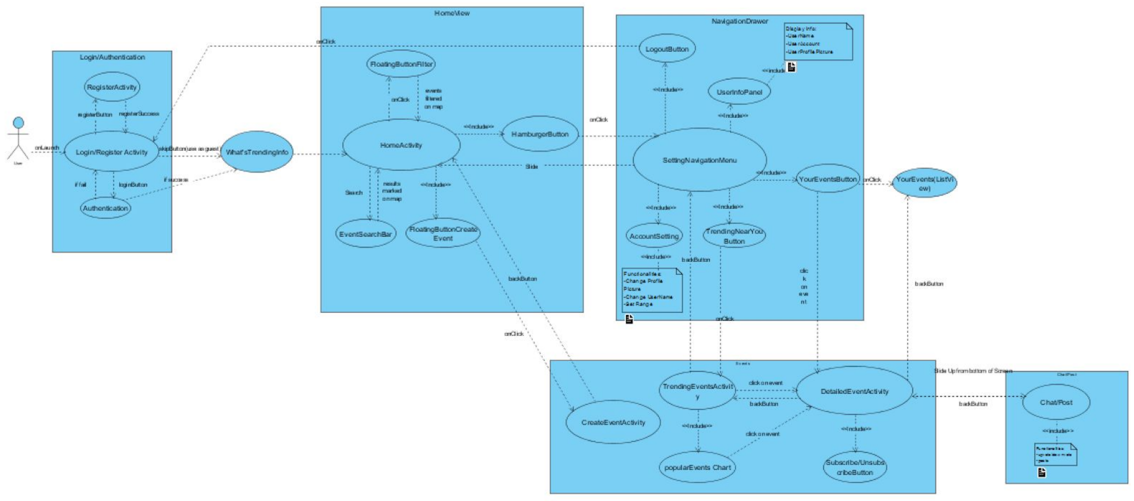
The testing process has two goals:

- To demonstrate to the developer and the customer that the software meets its requirements
- To discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification

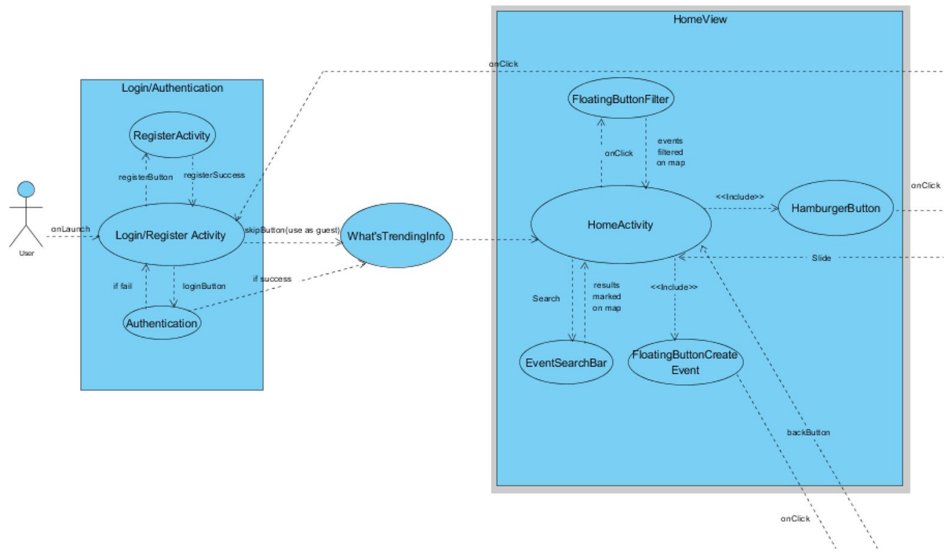
For the purpose of this document, since we have not developed the complete code, we will focus on Development Testing and User Testing

Use Case Diagram

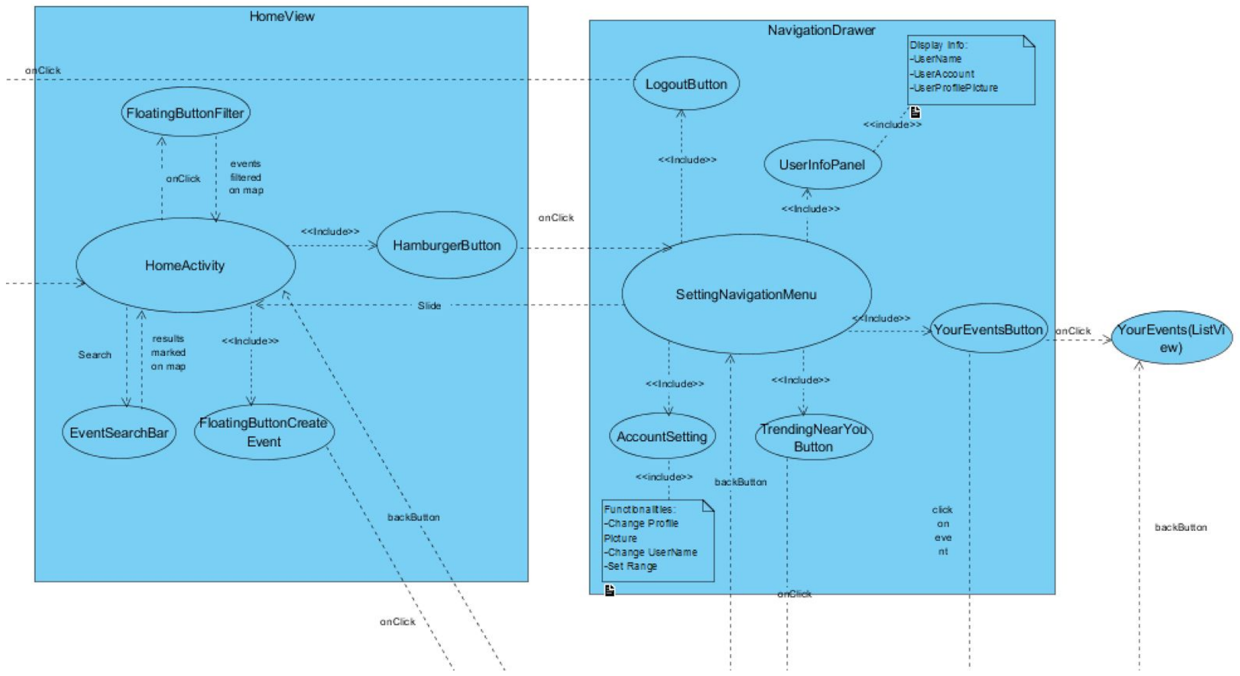
Overview



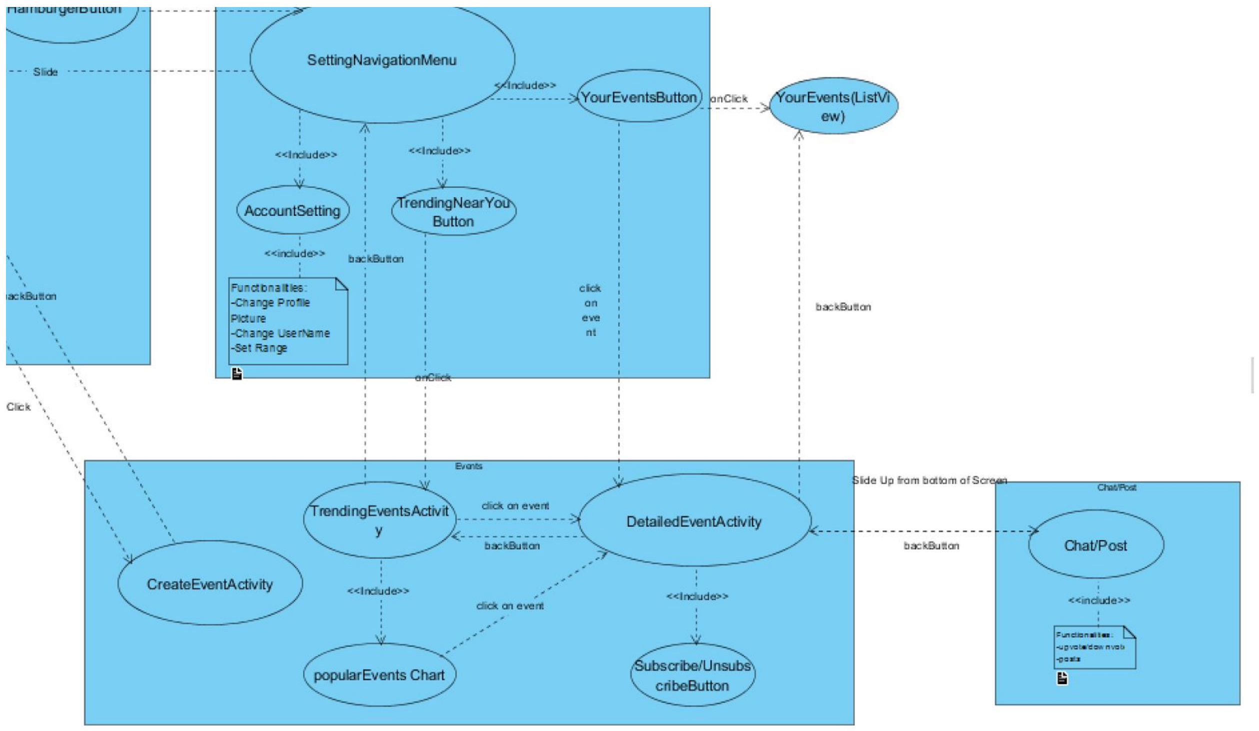
Login/Register -> HomeView



HomeView -> NavigationDrawer



NavigationDrawer & Chat/Post options



Testing Frameworks

- JUnit Tests
- Google Test API

Common Types of Testing

- Instrumented Unit Testing - for tests on physical device & emulators that uses the device's context, i.e. Current Location determined by phone GPS.

Based on the type of test you want to create, configure the test code source location and the project dependencies in Android Studio as described in the following sections.

Test Code References
http://developer.android.com/training/testing/start/index.html
http://developer.android.com/training/testing/unit-testing/local-unit-tests.html
http://www-scf.usc.edu/~csci201/lectures/Lecture11/Testing.pdf
http://www.tutorialspoint.com/android/android_testing.htm

Example. JUnit local testing - validate sign-in/register email address format

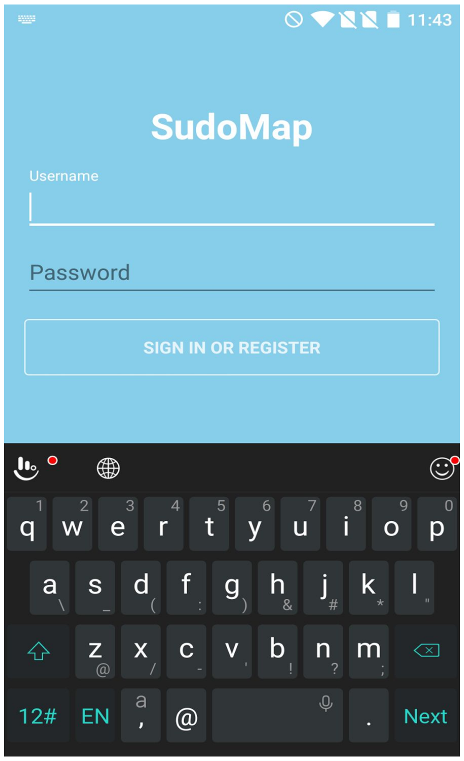
```
import org.junit.Test;
import java.util.regex.Pattern;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

public class EmailValidatorTest {

  @Test
  public void emailValidator_CorrectEmailSimple_ReturnsTrue() {
    assertThat(EmailValidator.isValidEmail("name@email.com"), is(true));
  }
  ...
}
```

Test Cases **TBA: To Be Added*

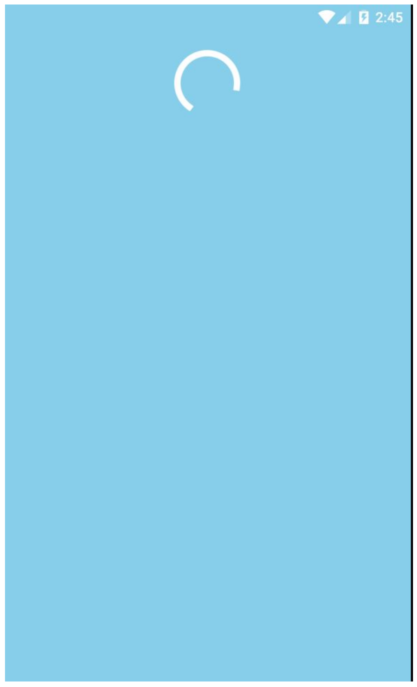
5.2 - Database Tests

1. Users must be able to register so they can become an authenticated user.

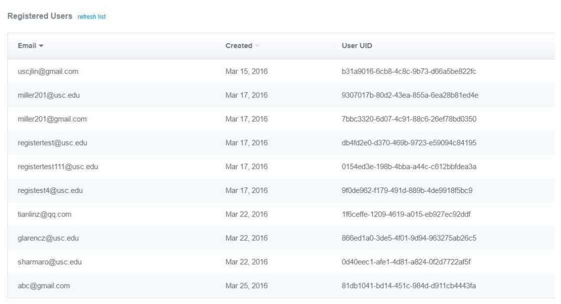
Test #	D-01
Test Description	• User can use password and username combination to register

	● Username is unique, so a user cannot register if there's already an account registered with the same name ● Username should contain "@" sign, since users should register with their email accounts. Username without @ sign should be checked and refused before notifying Firebase with registration.
Steps to Run Test	1. Registration with Username: miller201@usc.edu and Password: csci201. 2. Register again with Username: miller201@usc.edu and Password: csci201. 3. Register with Username: miller201 and Password: csci201 .
Expected Result	1.Success. 2.Refused, and a message should pop up saying that the account already exists. 3.Refused, and no query should be sent to Firebase. An error message should notify that username is invalid
Actual Result	N/A
Screenshot	 11:43 SudoMap Username Password SIGN IN OR REGISTER 12# EN Next

2. User must be able to log in

Test #	D-02
Test Description	The server should be able to authenticate log-in requests, reject wrong and accept correct ones.
Steps to Run Test	At this point, assume test 1 is executed. 1. Login with Username: miller201@usc.edu and Password: csci104. 2. Login with Username: miller201@usc.edu and Password: csci201.
Expected Result	1.Fail, a message should pop up notifying user of login failure. 2.Success, the app should launch an intent and proceed to MapActivity.
Actual Result	N/A
Screenshot	

Test #	D-03
Test Description	Users are able to register a new account, and have it saved in the database.
Steps to Run Test	When user press “if taken” button, a request should be fetched to Firebase to see if account name already ex
Expected Result	1.Fail, a message should pop up notifying user of login failure.

	2.Success, the app should launch an intent and proceed to MapActivity.
Actual Result	N/A
Screenshot	

Registered Users refresh list

Email	Created	User UID
uscjlin@gmail.com	Mar 15, 2016	b31a9016-6cb8-4c8c-9b73-d66a5be822fc
miller201@usc.edu	Mar 17, 2016	9307017b-80d2-43ea-855a-6ea28b81ed4e
miller201@gmail.com	Mar 17, 2016	7bbc3320-6d07-4c91-88c6-26ef78bd0350
registertest@usc.edu	Mar 17, 2016	db4fd2e0-d370-469b-9723-e59094c84195
registertest111@usc.edu	Mar 17, 2016	0154ed3e-198b-4bba-a44c-c612bbfdea3a
registest4@usc.edu	Mar 17, 2016	9f0de962-f179-491d-889b-4de9918f5bc9
tianlinz@qq.com	Mar 22, 2016	1f6ceffe-1209-4619-a015-eb927ec92ddf
glarencz@usc.edu	Mar 22, 2016	866ed1a0-3de5-4f01-9d94-963275ab26c5
sharmaro@usc.edu	Mar 22, 2016	0d40eec1-afe1-4d81-a824-0f2d7722af5f
abc@gmail.com	Mar 25, 2016	81db1041-bd14-451c-984d-d911cb4443fa

Test #	D-04
Test Description	Unauthorized users will have limited functionality because they will be experiencing the app in a “Guest Mode”
Steps to Run Test	1. Have a button with the label “Guest Mode” or something similar 2. If the user pushes this button, he/she will have access to the app with limited access
Expected Result	● User can view all events that have been added ● User cannot communicate with other users on events ● User cannot post any events ● User cannot bookmark any events for later reference ● User cannot update user preferences because they are not signed in
Actual Result	N/A
Screenshot	TBA

Test #	D-05
Test Description	Authorized users will have complete control of the app with no limitations on access
Steps to Run Test	1. Give the user the option to log in with their account information 2. If the user has an authorized account, he/she will have full

	access to the app
Expected Result	● User will be allowed to post events ● User will be allowed to comment in other event chatrooms ● User will be allowed to save/bookmark events for later reference ● User has the ability to chat with event host & other participants ● User has full access to their preference settings
Actual Result	N/A
Screenshot	TBA

5.3 - Home Tests

1. User should be able to see some of the current, trending events at the start of the app

Test #	H-01
Test Description	“Whats New” popup should be displayed once HomeActivity has loaded
Steps to Run Test	1. Open the app 2. Log In, register, or sign in as a guest
Expected Result	1. Success: “Whats New” popup is displayed on the HomeActivity after all events are loaded 2. Failure: Popup not displayed or done so before all events have been loaded
Actual Result	N/A
Screenshot	TBA

2. Events should be loaded and displayed correctly on the Map View

Test #	H-02
Test Description	The 10 hard coded test events should be displayed as blimps once the app is loaded
Steps to Run Test	1. Open the app 2. Log In, register, or sign in as a guest 3. Close the What’s New popup

	4. Click each blimp to check which event each is
Expected Result	1. Success: 10 blimps should be loaded and the following events should exist a. Springfest b. In n out Frenzy c. Crime Scene on Jefferson d. Chemical Spill on McClintock e. Elon Musk talk f. Biggest Street Party Ever!!! g. #PraiseYeezus h. Trump Rally i. Roots & Shoots j. Viterbi Career Fair 2. Failure: Some or none of the following events are displayed
Actual Result	N/A
Screenshot	TBA

3. User must be able to navigate to account page from side panel

Test #	H-03
Test Description	The account link should lead to the account page
Steps to Run Test	1. Tap the account link or account picture
Expected Result	1. Success: All the information should be displayed on the account page, such as: a. Name b. Picture c. Email address d. Password e. Birthdate
Actual Result	N/A

4. User must be able to change account information

Test #	H-04
Test Description	User should be able to change information with account settings displayed below info
Steps to Run Test	1. Change the name from Jeffrey Miller to Kenneth Chang

	2. Change the email address to kenneth_chang@usc.edu 3. Change the password from "csci201" to "kenChangswag" 4. Change the username from "miller201" to "ken201" 5. Change the picture to any one from the user's phone or leave it blank 6. Log out and log in to check changes
Expected Result	1. Success: All changes made should be displayed after user confirms it and should be accordingly be updated in the database 2. Failure: None or not all changes are made or updated
Actual Result	N/A

5. User must be able to navigate to "Trending Places" page from side panel

Test #	H-05
Test Description	User should be able to see the trending, current events and navigate back
Steps to Run Test	1. Click the "Trending" link 2. Click the back arrow button at the top left or the Android back button
Expected Result	1. Success: From after app has just started up, 5 of the 10 hardcoded events should show up a. Springfest b. In n Out Frenzy c. Crime Scene on Jefferson 2. Success: User should be able to go to previous activity from tapping either back button 3. Failure: Cannot see any of the trending test events and/or navigate back to previous activity
Actual Result	N/A

6. User should be able to see their own events from side panel

Test #	H-06
Test Description	Any events the user puts Interested or Going, said events should show up in "Your Events" or empty if the use has not committed to any events yet
Steps to Run Test	1. Click the "Your Events" link to check if empty

	2. Go back to the homepage and hit “Going” for Springfest and “Interested” for In n Out Frenzy 3. Click the “Your Events” link again
Expected Result	1. Success: List should be initially empty and then have Springfest (old event) in the Past pane and In n Out Frenzy in the Upcoming pane 2. Failure: List does not have the updated events
Actual Result	N/A
Screenshot	TBA

7. User changing an event to “Not Going” should update accordingly

Test #	H-07
Test Description	A committed/saved event changed to “Not Going” should remove event from user’s “your events” list
Steps to Run Result	1. Go to Springfest from “Your Events” 2. Update from “Going” to “Not Going” 3. Go back to “Your Events” to check (back button)
Expected Result	1. Success: Springfest is removed from “Your Events” 2. Failure: Springfest is still there
Actual Result	N/A
Screenshot	TBA

8. User should be able to log out

Test #	H-08
Test Description	Tapping the “Log Out” link from the side panel should log the user out
Steps to Run Result	1. Tap the “Log Out” link
Expected Result	1. Success: User should be led back to the homepage, where they can either log in, register, or sign in as a guest
Actual Result	N/A
Screenshot	TBA

9. User must be able to create a new event

Test #	H-09
Test Description	Tapping the add button from the floating button should allow the user to create a new event
Steps to Run Test	1. Tap the floating button 2. Tap the create a new event 3. Type “Conquest” for the name 4. Type “2016 Conquest - Best Concert Ever!” for the description 5. Choose the category “fun” 6. Make the event public 7. Tap the add event button
Expected Result	1. Success: User should be able to reach AddEventActivity and create the “Conquest” event with the above details showing 2. Failure: Cannot reach AddEventActivity and/or “Conquest” event cannot be successfully created
Actual Result	N/A
Screenshot	TBA

10. User should be able to filter events

Test #	H-10
Test Description	Tapping the filter “fun” should display all events that are fun
Steps to Run Test	1. Tap the floating button 2. Tap the filter item “fun” in the fanning list
Expected Result	1. Success: All the following “fun” events should be shown a. Springfest b. Conquest c. Biggest Street Party Ever!!! d. #PraiseYeezus 2. Failure: Some or None of the events are shown or filters are not displayed from floating button
Actual Result	N/A
Screenshot	TBA

5.4 - Event Tests

1. User should be able to filter events and browse through them

Test #	E-01
Test Description	Tapping on the floating button will expand the filter fragment menu
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press the floating button on the bottom right
Expected Result	● A vertical bar displaying filter options as circle buttons will expand upwards ● The floating button will now turn into a “+” sign, allowing the user to tap on it to add an event
Actual Result	N/A
Screenshot	TBA

Test #	E-02
Test Description	Pressing a filter will properly hide and show corresponding events
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press the floating button on the bottom right 3. Press a filter icon
Expected Result	The floating fragment should disappear, a spinner will show up in the center for a few seconds, and the event markers will be refreshed
Actual Result	N/A
Screenshot	TBA

Test #	E-03
Test Description	Tapping on an event marker will bring up Event Details
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press on one of the event marker balloons

Expected Result	A new “EventDetailActivity” should pop up and populate with details of the event.
Actual Result	N/A
Screenshot	TBA

2. User should be able to interact with events and bookmark them.

Test #	E-04
Test Description	Sliding up in Event Details will bring up chatroom associated with the event
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press on one of the event marker balloons 3. Hold the up arrow button and slide up
Expected Result	A “EventChat” activity should enter with a customized “sliding up” animation. Hitting the Android back button will close the activity and return to the previous event details activity.
Actual Result	N/A
Screenshot	TBA

Test #	E-05
Test Description	EventDetailActivity should load information of an event correctly.
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press on one of the event marker balloons to get to EventDetailActivity
Expected Result	The following event information should be populated to the corresponding GUI components correctly ● Title ● Description ● Category ● Organizer’s name ● Location’s name ● Location in map view
Actual Result	N/A

Screenshot	TBA

Test #	E-06
Test Description	User should be able to mark himself/herself as “attending/going to” and event.
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press on one of the event marker balloons to get to EventDetailActivity 3. Click “going”
Expected Result	● In the user’s “your events” page, the said event should appear. ● In the event’s list of attendants, the said user should appear.
Actual Result	N/A
Screenshot	TBA

Test #	E-07
Test Description	User should be able to bookmark an event
Steps to Run Test	1. Launch app into HomeActivity (map view) 2. Press on one of the event marker balloons to get to EventDetailActivity 3. Click “bookmark”
Expected Result	● In the user’s “your events” page, the said event should appear with a label that says “bookmark” ● In the event’s list of attendants, the said user should not appear since the user only bookmarked the event.
Actual Result	N/A
Screenshot	TBA

6 - Deployment Document

Android Studio Installation/Setup
(Disclaimer: modified from ITP-341 Installation Files)

- Download and install Java 7
    - Visit the following site: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
    - Accept the license
    - Download for Windows or Mac
- Download and install Android Studio
    - Visit the following site: http://developer.android.com/sdk/index.html
    - **Windows**
        - Launch the .exe file
        - Install studio and any necessary SDK tools
        - Some Windows systems' launcher scripts can't find where Java is installed
            - Set the environment variable indicating the correct location: SelectStartmenu>Computer>SystemProperties>AdvancedSystem Properties. Then open Advanced tab > Environment Variables and add a new system variable JAVA_HOME that points to your JDK folder, for example C:\Program Files\Java\jdk1.7.0_21.
    - **Mac**
        - Launch the .dmg file
        - Drag and drop Android Studio into Applications folder
        - Open Android Studio and follow the setup wizard to install any necessary SDK tools
- Install SDKs and tools (API Level 21 and above)
    - Tools
        - Android SDK Tools
        - Android SDK Platform Tools
        - Android SDK Build-Tools (highest version)
    - Android 5.1.1 (API 21)
        - Documentation for Android SDK
        - SDK Platform
        - Samples for SDK
        - Intel x86 Atom System Image

        - Google APIs (x86 System Image)
    - Extras
        - Android Support Library
        - Android Support Repository
        - Google Play Services
        - Google Repository
        - Google USB Driver
    - Intel x86 Emulator Accelerator
- Setup AVD (if doing emulator)
    - Click the “Run app” Button (green play button) in the middle of the tool bar
    - Click the “Create new Emulator” in the “Select Deployment Target” window
    - Search for “Nexus 5”, select it, and click “Next”
    - Select “Marshmallow” (64 bit) and click “Next”
    - Click “Finish”

Once Android Studio is properly set up, download the source code. If it is in a ZIP file, extract its contents to any location of choice. When Android Studio is launched, select “Open an existing Android Studio Project”, and navigate to the location where the source code was saved. There is no need to setup a server instance on the local machine because our application connects to a Firebase server over the cloud.

Using an Android Device

To deploy this application onto an Android device, install the drivers for your corresponding phone. Note that drivers for different host operating systems may vary (Windows/Mac). On Mac OS X, you may simply install Android Task Manager and Android Studio will recognize most devices. If a Windows PC is used, individual drivers specific to the phone model need to be installed. Before deploying the application to the phone, make sure Location Services is turned on to High Accuracy, and allow any additional permissions asked. Make sure the phone is also connected to the Internet through WiFi or 4G cellular.

Using an Emulator (limited functionality w/o GPS)

To deploy this application in an emulator, click the “Run app” Button (green play button) in the middle of the tool bar. Select the AVD previously setup that’s a Nexus 5 device on API 23 and press “OK”. Wait for the AVD to setup and the app to load up. Accept any necessary permissions and follow any upcoming setup instructions and begin testing.