**Title**
Map navigation app with social posting



**Author**
Xu, Jian

**Publication Date**
2016-11-17

Peer reviewed

# Map navigation app with social posting

Jian (Kevin) Xu

## 1. Introduction

MapNav is a location based image social app for the iPhone. For activities like Picnic Day at UC Davis, people might be interested to discover appealing events that are not widely promoted. Using MapNav, they could take and post photos in a map (as posters), and others could like or dislike the photo post (as browsers). Users can choose interesting events to go to by browsing the posts near their location, or they can view a rank board showing all the posts.

This map-based social network application can be useful in the scenario of navigation in general. Such navigation can be city navigation, museum navigation and campus navigation, etc. By posting indoor photos, MapNav fills a need left open by tools like Google Earth. The accuracy of indoor locations measured by GPS will be slightly lower compared to outdoor locations because of GPS signal attenuation caused by construction materials. However, indoor location service can be improved using hybrid technology, which incorporates Wi-Fi, GPS, cell towers, IP address and device sensors [1].

There are some commercial products that are built with map and social features. Spottesetters combines friend's recommendations, information from social media and online reviews with a map feature [2]. CityMaps is an urban planner mapping the New York City and adding tags to different locations based on popularity and recommendations [3]. Met is a museum navigation app allowing pin the location of events, exhibitions and artworks [4]. To make a difference, MapNav can target to the campus navigation or some temporary events like Picnic Day or career fair.

The current work is a frontend implementation. A server will be implemented as a backend in the future to store and update posts. The backend work needs to solve the problem that which and how many photo posts should be pushed to the user's map view, according to the user's location and activity relevance.

## 2. Methods

MapNav is written in Objective-C with Xcode with 4 basic screens: (1) photo shoot and edit, (2) photo gallery and marker function, (3) like and dislike feature for the post in the map view, (4) ranked list of photo posts.

### 2.1 Photo shoot and edit

A photo of an interesting event can be taken using the iPhone camera. To access the iPhone camera, the user should add a new photo into the photo gallery. This could be done by tapping "+" button at the upper right corner of the navigation bar (As is shown in *Figure 1.a*). When the "+" is clicked, a grey photo background is added into the photo gallery. Then tapping the grey photo background will lead to the detailed editing interface in 3.2. There is a camera button at the left bottom in the shoot and edit view. In the photo and edit interface, a post name and description can be added. The location where the photo is taken will be recorded as latitude and longitude, and then be saved as metadata for the post.

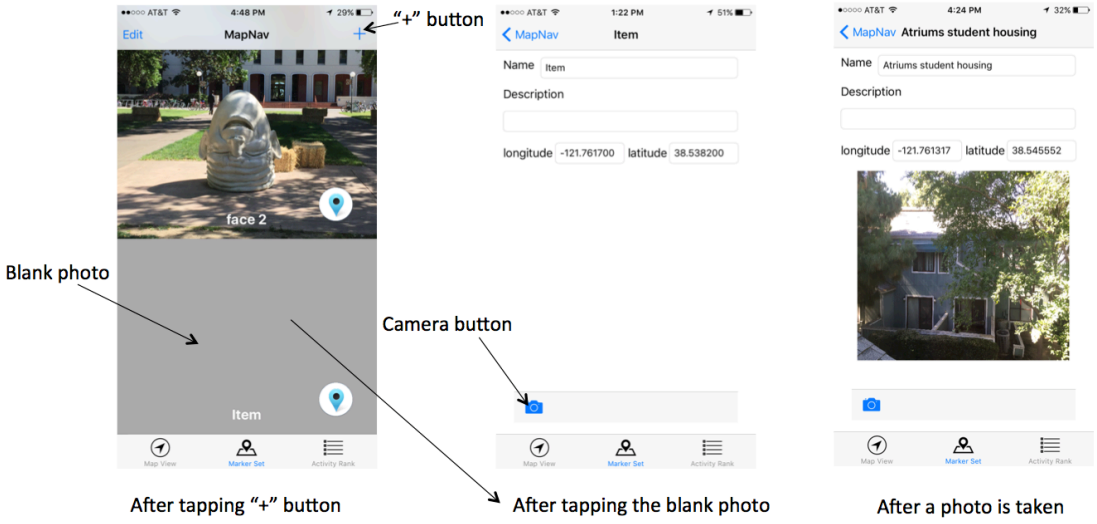


*Figure 1.a View of shoot and edit*

### 2.2 Photo gallery of browsing photos and marker method

A photo gallery is implemented for users to browse the photos. For a clear view doing browsing, each photo is presented at a large size. The post name is positioned at the

middle bottom of each photo, and a mark button is positioned at the right bottom of each photo.

In addition to adding a new photo, users can also edit, and delete posts in the photo gallery. To edit name and description of a photo, detailed editing interface (the third picture in *Figure 1.a*) can be accessed at anytime by tapping the photo. To delete a photo in the gallery, user taps the "Edit" button in the upper left of navigation bar. A red delete button appears. Tapping the red delete button removes the photo from the photo gallery. Tapping the marker button adds the photo post to the map view.

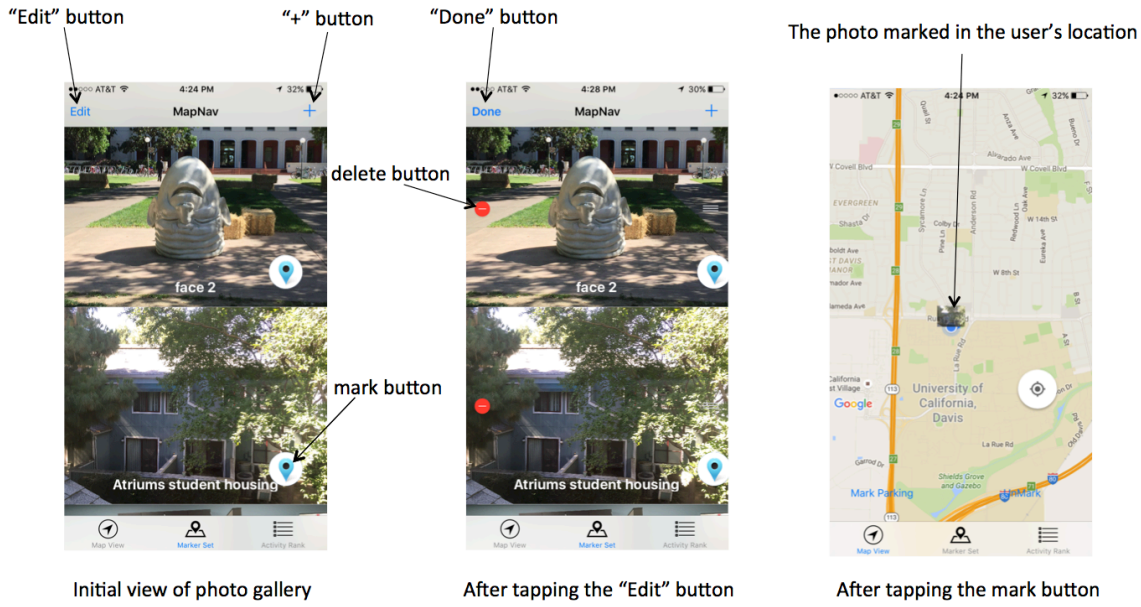


*Figure 1.b Buttons and states in the photo gallery*

### 2.3 Like and Dislike feature for a post in the map view

After the mark button is tapped in the photo gallery, a photo post is marked in the map interface. The map view is implemented using the Google Maps API. The markers remain at a constant size during zoom of the map view.

When a marker post is tapped, a post view information window pops out in the map view. A photo with a larger size and the post name will appear in the window. Also, like and dislike buttons and the numbers of likes and dislikes are displayed in the window.

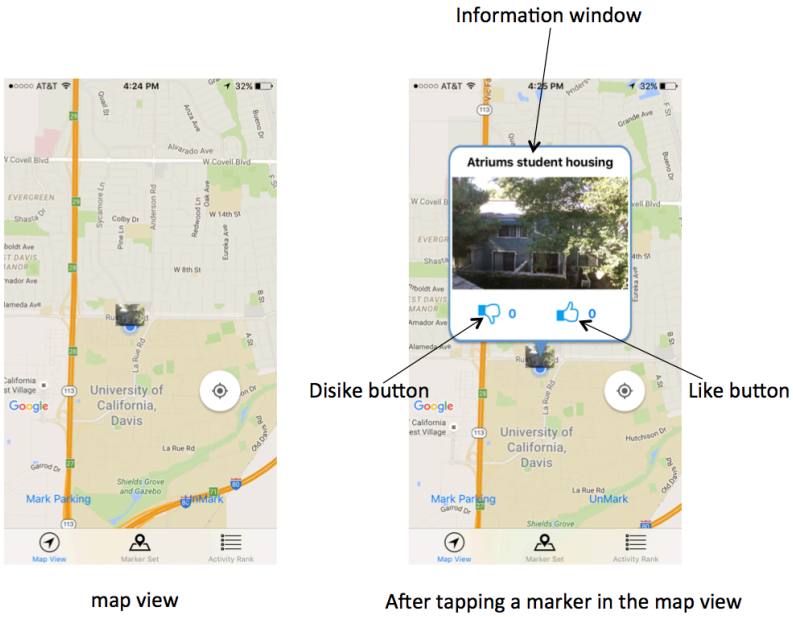


*Figure 1.c Information window, like and dislike buttons*

### 2.4 Rank algorithms for photo posts

Three rank lists are provided, which are (1) ranking by likes, (2) ranking by LBW (lower bound Wilson algorithm) and (3) ranking by weighted rating. A screenshot of ranked items view is in *Figure 8*.

## 3. App framework: Model-View-Controller

MapNav follows the MVC (Model-View-Controller) paradigm. In MVC, every object is either a model object, a view object, or a controller object. View objects are visible to users. Examples of view objects are buttons and text fields. Model objects hold data. Controller objects configures views the user sees and makes sure that the view and model objects are in sync.

As is shown in *Figure 2*, there are 4 basic views, which are (1) map view for displaying markers and user current location, (2) photo gallery view for adding and setting markers, (3) marker editing view for getting image and editing name of marker, (4) ranking view for ranking the marker post according to their likes and dislikes. Corresponding to the 4 views, there are 4 basic controllers, which are (1) MapNavViewController, (2) ItemsViewContoller, (3) DetailViewController, (4) RankItemsViewController. A tab bar at the bottom is set between the 3 views (1), (2) and

(4) by at each screen. The shoot and edit view is accessed via photo gallery view by tapping a photo. To go back, the user can either use the tab bar, or the top navigation bar.

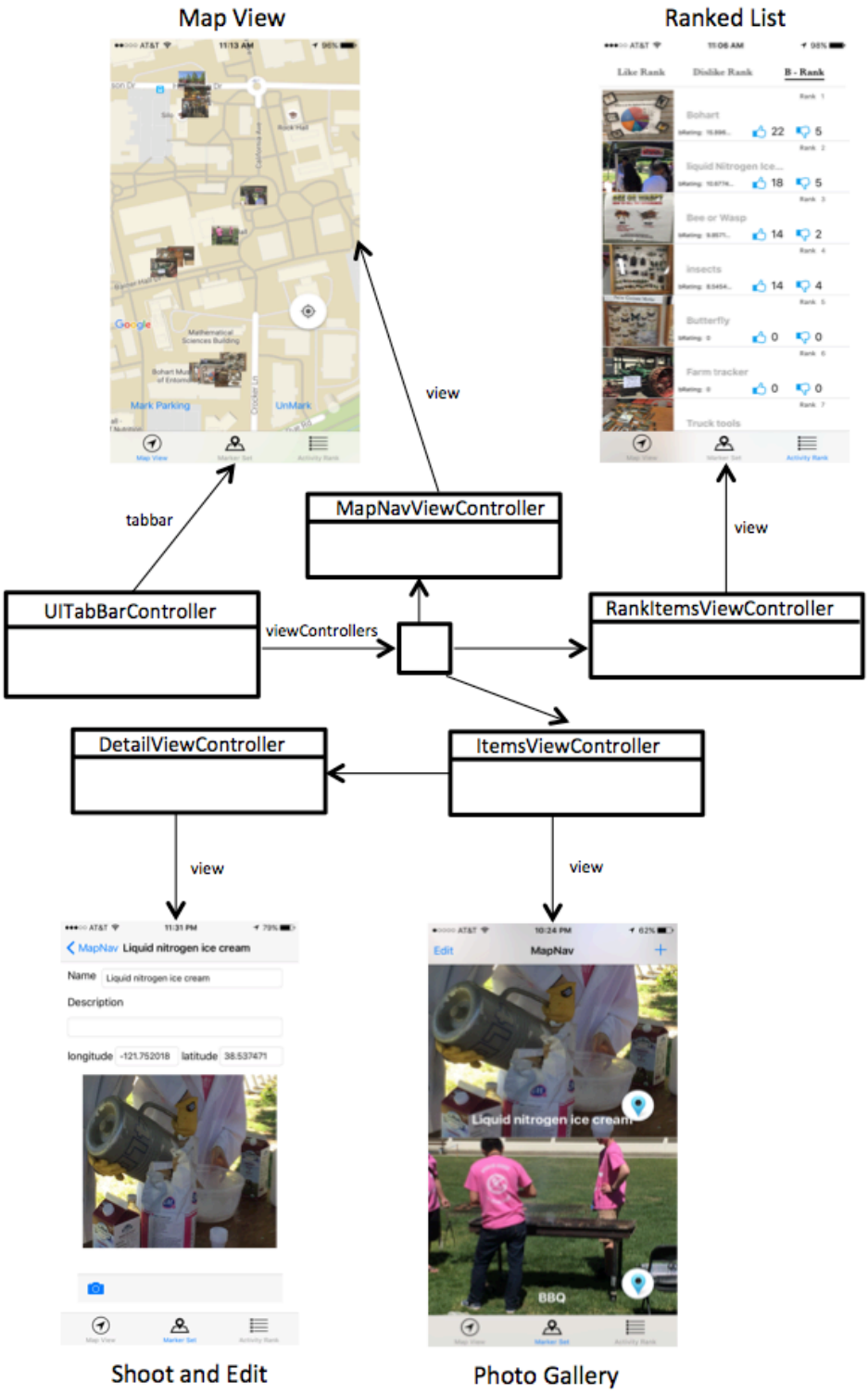


*Figure 2. Model-View-Controller framework for MapNav*

### 3.1 Data storage in model and ItemsViewController

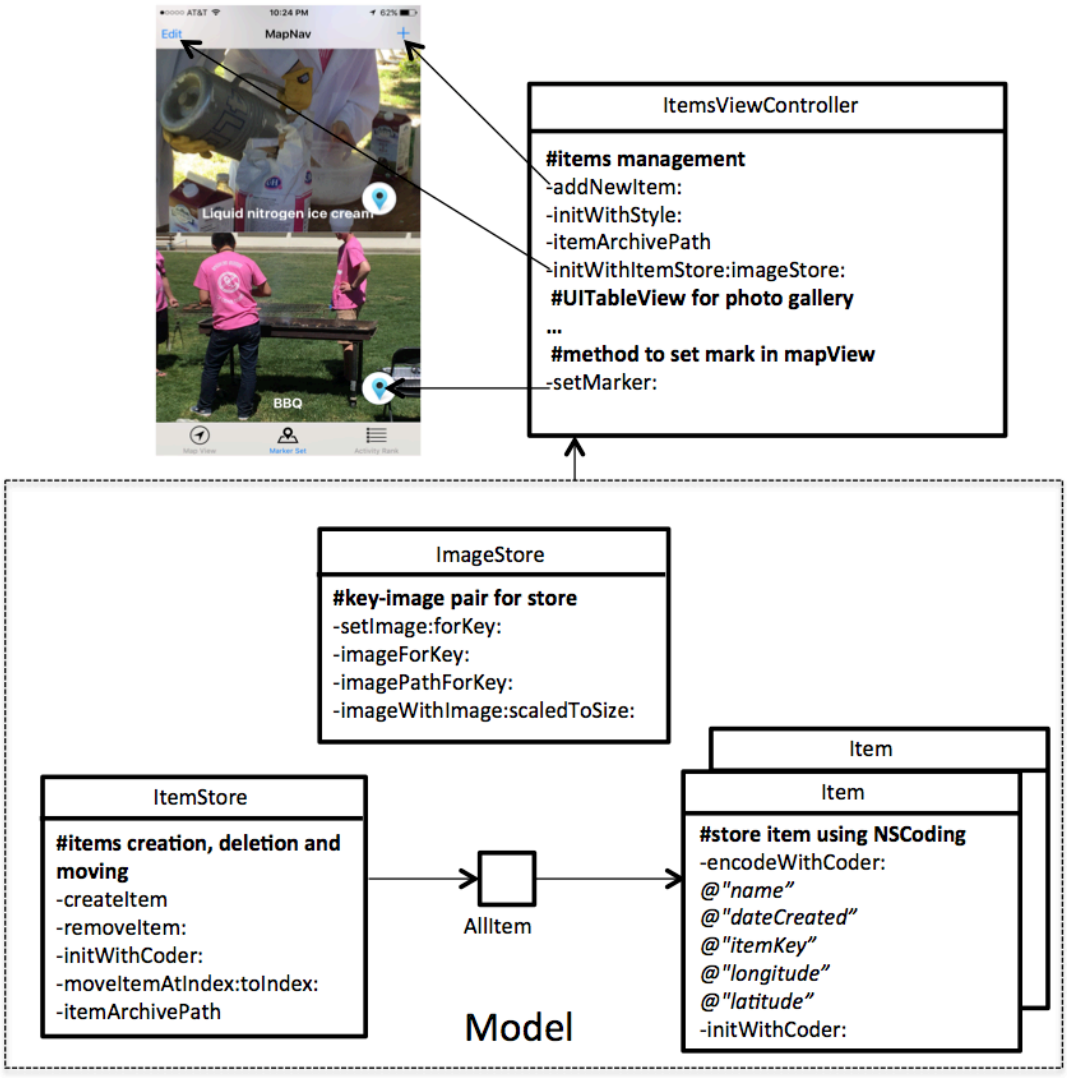


*Figure 3. Model and ItemsViewController*

The ItemsViewController is implemented as the basic data controller in MapNav, because it controls the data in the model. The basic data structure is a decoded image and a set of strings, including *name, dateCreated, itemKey, longitude and latitude*. Archiving to the phone is used to record some of the object's properties and save them to the phone's file system. Two iOS methods **encodeWithCoder:** and **initWithCoder:** are implemented for archiving. As a result, the photos and marker names can be stored locally in the iPhone after the app is closed.

### 3.2 Marker information editing and DetailViewController

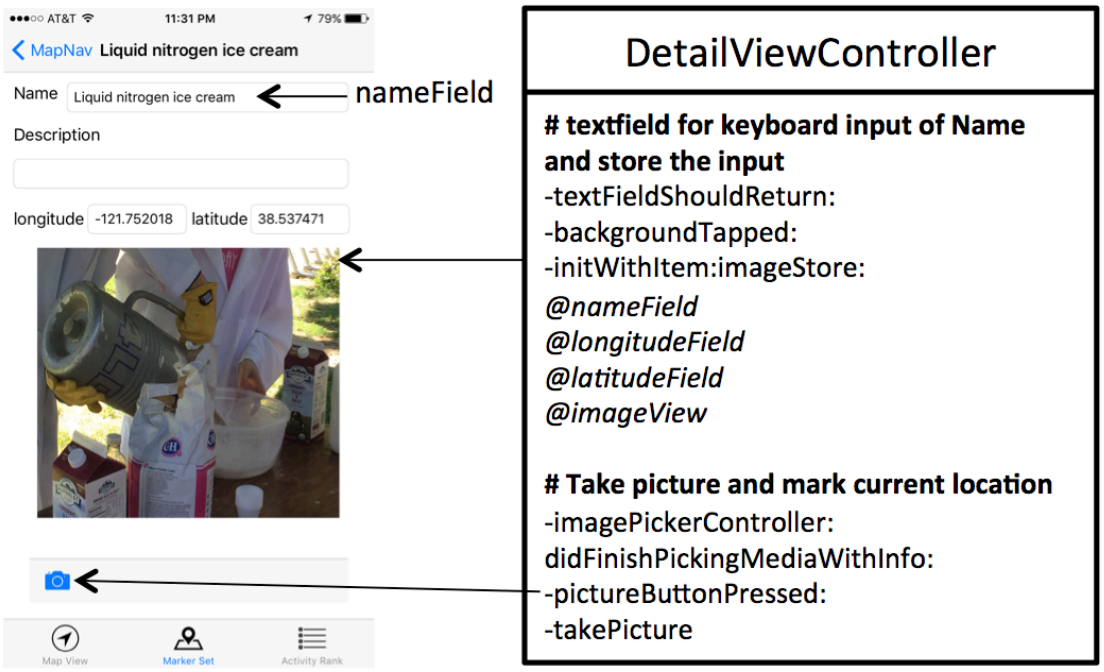


*Figure 4. Methods in DetailViewController*

After tapping a photo in the photo gallery, the app brings up the shoot and edit view. The two basic functions are setting the marker name and taking a photo while recording the location. The associated methods are coded in the DetailViewController. The marker name is input to the *nameField* (as is shown in *Figure 4*) and will be stored in the marker item after going back to the photo gallery view. When the camera button is tapped, the user's current location is recorded as longitude and latitude. Then an iOS method **imagePickerController:** will run a camera function to take and store a photo.

### 3.3 Marker setting in mapView and MapNavViewController

The MapNavViewController provides the interface from the backend Google Maps server. It displays the default map view, which is set to the main campus of UC Davis. Also, it uses several location methods to get the user's current location. The longitude and latitude are passed to ItemsViewController after the photo is taken. When the marker button is tapped, variables *markerItemName*, *markerItemImage*, *markerItemLongitude*, *markerItemLatitude* are passed back from ItemsViewController to the MapNavViewController to place the marker into the map view (as is shown in *Figure 5*).

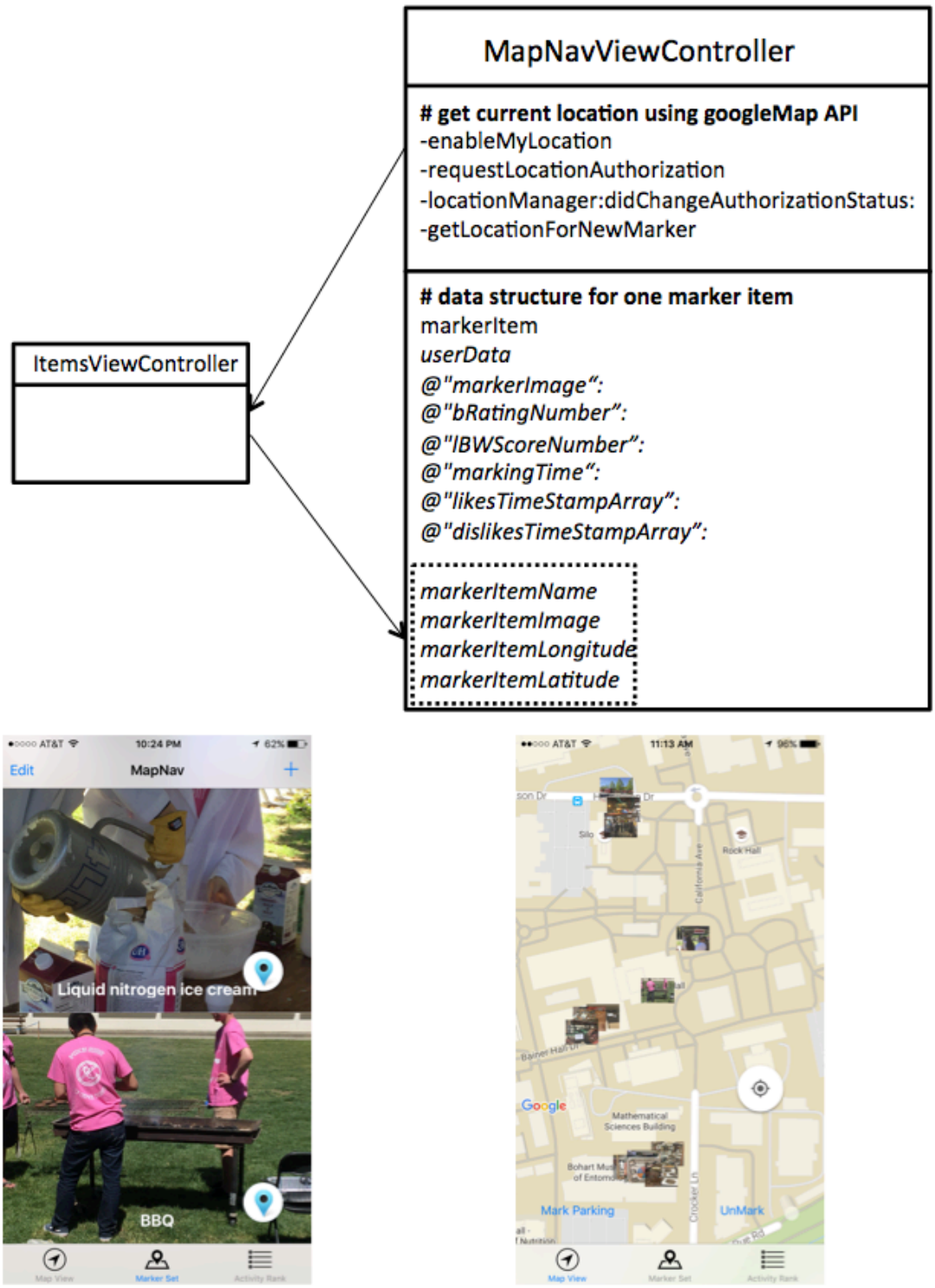


*Figure 5. Flow of marker placement in map view and the data structure of one marker item*

In the map view, after a marker is tapped, an information window pops out. In the window, the marker name, image, number of likes and dislikes are displayed. The default Google Maps API has a window method, but it cannot provide interactive content (like button events) in the window. Therefore, the window is implemented separately using the location information of the marker. When the map is scrolled horizontally or vertically, the information window will move along with the marker position. This is achieved by self-written methods including **didChangeCameraPosition** and **idleAtCameraPosition**.

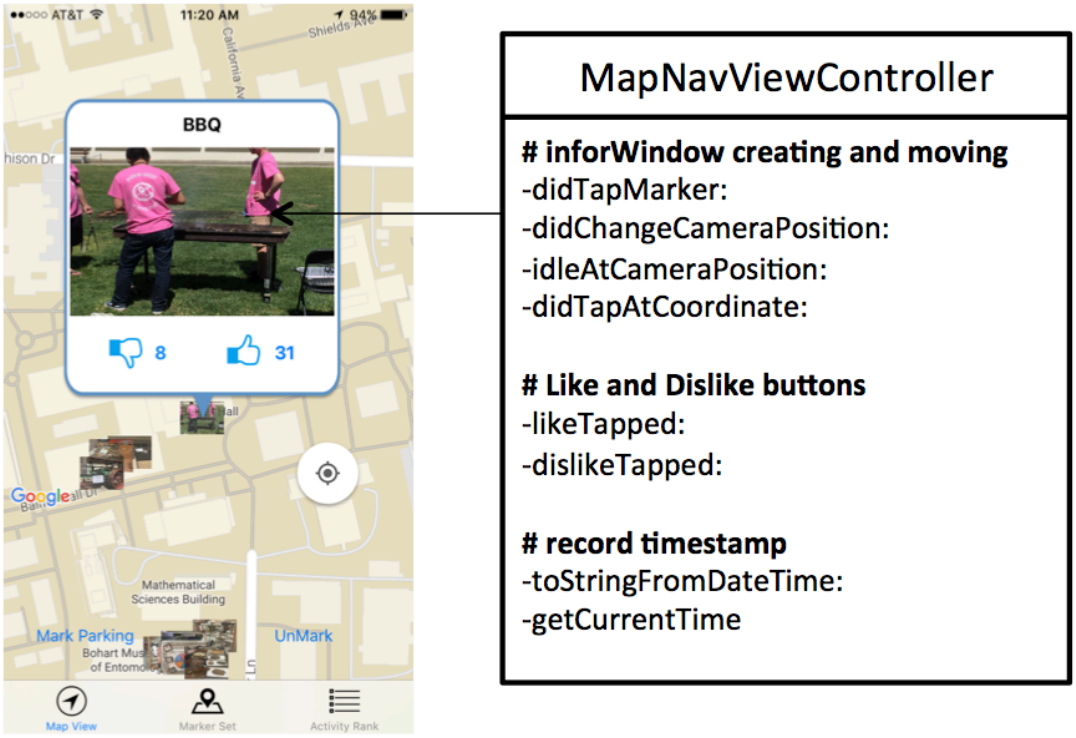


*Figure 6. Information window for one marker in the map view*

The Google Maps API provides a data structure for each marker to store additional information besides the marker name. It is called *.userData* and implemented as a set of key-value pairs. The *userData* we create has *bRatingNumber and IBWScoreNumber* that are computed and used for ranking marker posts. Also, the time stamps are recorded in UTC (Coordinated Universal Time) in ISO 8601 format, such as "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'". The time stamps include the marker setting time, likes tapped time stored in *likesTimeStampArray* and dislikes tapped time stored in dis*likesTimeStampArray*. One example of *.userData* for one marker is shown in *Figure 7*.

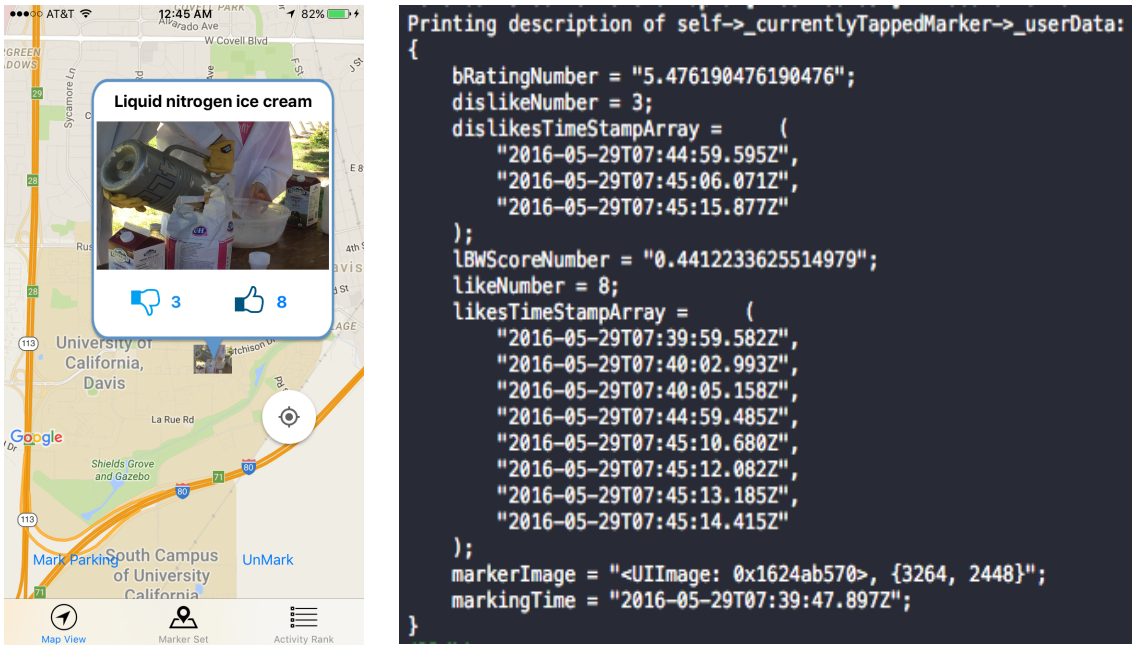


*Figure 7. Data Structure example of one marker with 8 likes and 3 dislikes*

### 3.4 item rank algorithms and RankItemsViewController

Three sort algorithms are implemented in the MapNavViewController. Then the result arrays are passed to the RankItemsViewController, which displays the ranked items in the form of UITableView, which is a widely used view format in iOS to display a single column of data with a variable number of row. The three sort algorithms are (1) like sort, (2) weighted rating and (3) LBW sort.

Like sort orders the photos by the number of likes for each post. Weighted rating is based on rating = likes – dislikes.

$$Weighted\ Rating\ = \frac{AveRating * AveVotes + ThisRating * ThisVotes}{AveVotes + ThisVotes}$$

in which:

*AveRating = (sum of all likes – sum of all dislikes) / number of photo posts,*

*AveVotes = (sum of all likes + sum of all dislikes) / number of photo posts,*

*ThisRating =ThisLikes – ThisDislikes,*

*ThisVotes = ThisLikes + ThisDislikes,*

*ThisLikes: number of likes of this post,*

*ThisDislikes: number of dislikes of this post.*

Another ranking algorithm based on the fraction of likes is also implemented. The algorithm is called lower bound of Wilson score (LBW Score). It represents a confidence interval for a Bernoulli parameter. The LBW Score algorithm can work well even when the vote number of a post is low. The rank method is developed and revised in in Evan Miller's blog [5], which also mentions its application in Yelp, Reddit, and Digg.

$$LBW\ Score\ = \frac{\frac{ThisLikes + 1.9208}{ThisVotes} - \frac{1.96 * \sqrt{\frac{ThisLikes * ThisDisLikes}{ThisVotes} + 0.9604}}{ThisVotes}}{1 + \frac{3.8416}{ThisVotes}}$$

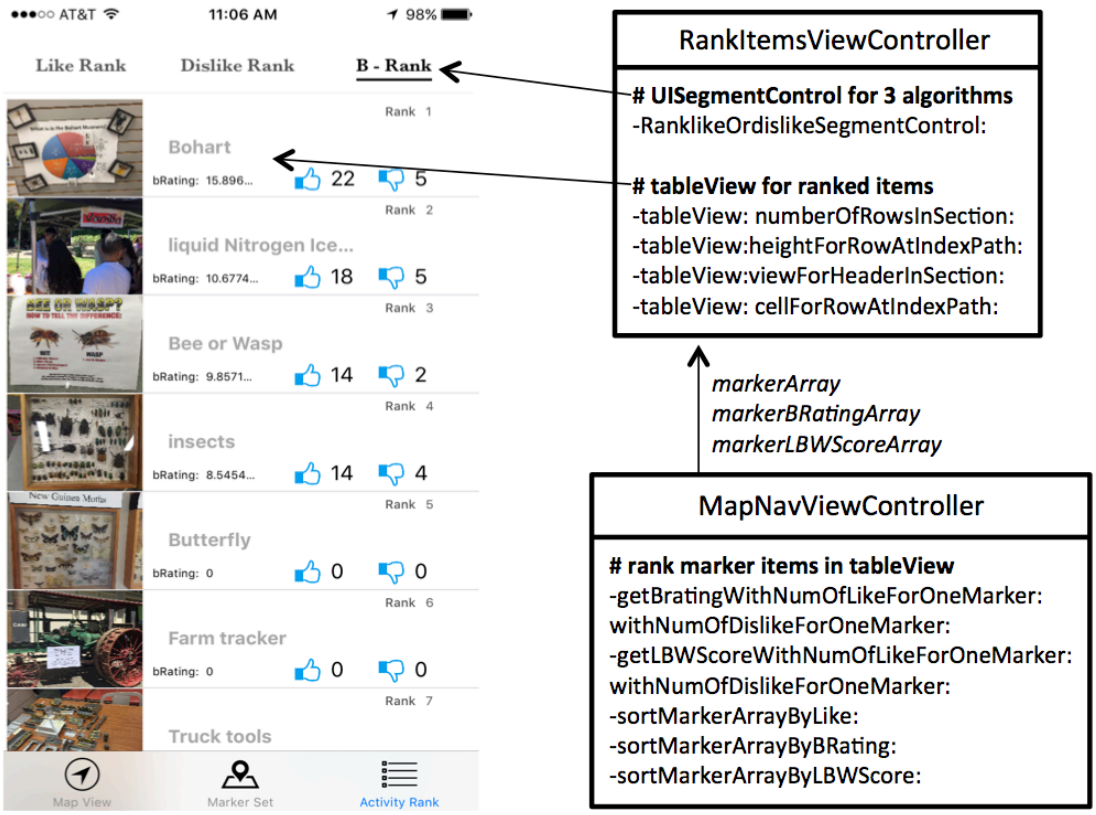


*Figure 8. Ranked items view*

## 4. Discussion

### 4.1 Use of dislike button

A dislike button is implemented in MapNav. Large social network products like Facebook and Linkedin do not include dislike. The main reason is that if the dislike button is not used sensibly, it may lead to negativity [6]. For an individual user, the posting and sharing might be inhibited by dislikes. For commercial users, many dislikes feedbacks might hurt the brand reputation, even if the dislikes may come for instance from competitors.

Facebook added emotion buttons recently [7]. Seven emojis (Like, Love, Haha, Yay, Wow, Sad, and Angry) were added as a new feature. The emojis enable Facebook user to express their feelings more specifically. Negative feedback such as sad or angry targets the post content itself, while "dislike" is a more general negative feedback, which can be interpreted as the dislike of the post or even of sharing behavior of the poster.

Other platforms like YouTube use both positive and negative feedbacks to rate the quality of content. "Dislike" is useful for rating. Also, it is not personal since the video upload users and the viewers are mostly anonymous. In MapNav, the dislike button is used for rating activities. For example, activities with long waiting time may get many dislike votes thus having a lower ranking.

### 4.2 Comparison of ranking algorithms

*Table 1. Comparison of different rank algorithms*

| Algorithm | Comments |
|---|---|
| Score = Likes – Dislikes | Post A: 80 likes, 20 dislikes, Score = 60.<br>Post B: 550 likes, 450 dislikes, Score = 100.<br>Post B will be ranked above Post A, even if the percentage of likes is higher for Post A. |
| Score = Likes / (Likes + Dislikes) | Post A: 2 likes, 0 dislikes, Score = 1.<br>Post B: 98 likes, 2 dislikes, Score = 0.98.<br>Post A will be ranked above Post B, even if there is just 1 like for it. |

| | |
|---|---|
| $Weighted\ Rating = \frac{AveRating * AveVotes + ThisRating * ThisVotes}{AveVotes + ThisVotes}$ | Post A: 20 likes, 0 dislikes,<br>Post B: 100 likes, 70 dislikes,<br>Post B will be ranked above Post A, since it has higher ratings votes. However, large number of dislikes is not reflected. |
| Score = Lower bound of Wilson score confidence interval for a Bernoulli parameter. [5]<br>$\left(\hat{p} + \frac{z^2_{\alpha/2}}{2n} \pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p}) + z^2_{\alpha/2}/4n]/n}\right)/(1 + z^2_{\alpha/2}/n).$<br>in which $\hat{p}$ is the *observed* fraction of positive ratings, $z_{\alpha/2}$ is the (1 - α/2) quantile of the standard normal distribution, and $n$ is the total number of ratings (total likes + total dislikes). | Based on fraction algorithm. Solved the problem when the total voting number is small. If the setting confidence to 95%, the equation will compute the lower bound of "real" fraction of positive ratings, when there is 95% chance it is correct ($z_{\alpha/2}$ can be set to 1.96 when confidence level is 0.95). |

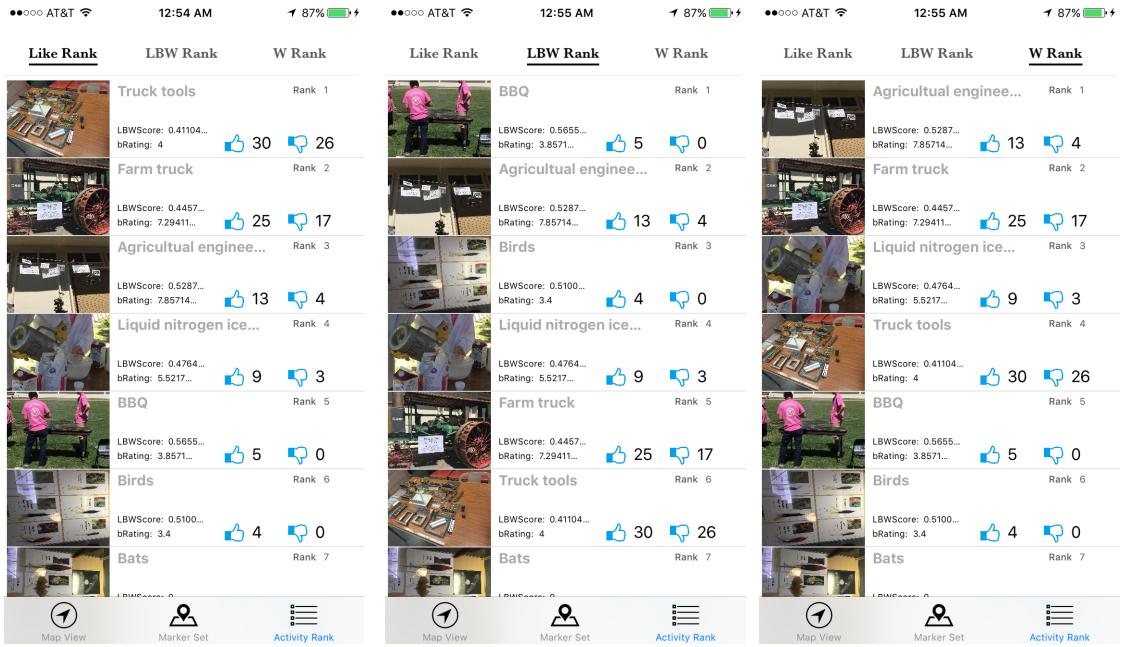


*Figure 9. Example of difference of 3 Controller ranks*

As is shown if Figure 9, the LBW rank algorithm ranks the 5-0 item to the first one, which has the potential of ending up with the most like ratio. The disadvantages of the

other two algorithms are that the like-algorithm only considers the number of likes and W-Rank may prefer the item with more votes, even if it may have lower like ratio.

### 4.4 Future work

1. MapNav can be integrated with other social app like Facebook, Twitter or Instagram in order to access more photo sources. This is necessary as a start point to build content.
2. When zooming out in the map view, how to present the photo that is more relevant. When the zoom of map view to a certain point, how many photo frames to show? For the ones that are not showing, how to overlay them? An algorithm based on location (longitude and latitude) and post popularity (likes and dislikes) should be designed.
3. Considering more parameters like time of marking, and time of views, how should the photo post be updated?
4. Create objects to contain photos about a certain activity. People would like to see posts that are relevant to a special event.
5. Implement a navigation function. It could be actuated by tapping a cell in the rank view or a marker in the map view. A route lead from the user's current location to the marker location could be displayed.

### 4.5 Reason for crash

During the test of MapNav, the app crashes sometimes when taking a photo. After tapping the camera button in the marker editing view. A black view in the photo interface shows and the app exits. This happens sometimes for the items that are created early, and will definitely happen when there are a number of items. The message *"Snapshotting a view that has not been rendered results in an empty snapshot. Ensure your view has been rendered at least once before snapshotting or snapshot after screen updates."* appears. This is a iOS bug present since iOS 8.0 when the iOS method **UIImagePickerController** was introduced [8].

For general reasons of app crash besides coding and design errors by programmer, there are other reasons that may lead an iOS app to crash, such as is shown in *Table 2*. The crash rates of iOS are 3.15% for iOS 7, 2.83% for iOS 8 and 2.74% for iOS 9. (May 29, 2016) [9]. Table 2 introduces some general reasons for an iOS app to crash.

*Table 2. Reasons for an iOS app to crash*

| Reasons of crash | Explanation |
|---|---|
| Low memory | If memory pressure exists, the system will terminate background process to free up some memory. If there is still not enough memory, the current process will be terminated [10]. |
| Network changes | Switch between cellular network and WiFi may cause instability. |
| Device incompatibility | Some size variables set as a particular number for iPhone 6 may exceed the frame of window size in iPhone 5. |
| Browser incompatibility | Third-party mobile version browser like chrome is not tested in iOS, which may cause potential risk for crash [11]. |
| Database contention | Bad queries and excessive sessions will slow performance thus causing stalk of app [11]. |

**Reference**


[1] "Skyhook Wireless," *Wikipedia, the free encyclopedia*. 17-May-2016.
[2] "Apple Acquires Social Map App Spotsetter," *Global Dating Insights*, 09-Jun-2014. [Online]. Available: http://globaldatinginsights.com/2014/06/09/09062014-apple-acquires-social-map-app-spotsetter/. [Accessed: 23-May-2016].
[3] J. Biggs, "CityMaps Launches Official iOS App That Aims To Make Mapping Social," *TechCrunch*. .
[4] "The Met App," *The Metropolitan Museum of Art, i.e. The Met Museum*. [Online]. Available: http://www.metmuseum.org/visit/met-app. [Accessed: 24-May-2016].
[5] "How Not To Sort By Average Rating." [Online]. Available: http://www.evanmiller.org/how-not-to-sort-by-average-rating.html. [Accessed: 20-May-2016].
[6] P. Sawers, "Facebook Dislike Button: Why it Will Never Happen," *The Next Web*, 10-Oct-2010. [Online]. Available:

http://thenextweb.com/socialmedia/2010/10/10/facebook-dislike-button-why-it-will-never-happen/. [Accessed: 20-May-2016].
[7] "Facebook explains how its new Like and Dislike emojis will impact post ranking," *VentureBeat*. [Online]. Available: http://venturebeat.com/2015/10/08/facebook-explains-how-its-new-like-and-dislike-emojis-will-impact-post-ranking/. [Accessed: 13-May-2016].
[8] "objective c - iOS 8 Snapshotting a view that has not been rendered results in an empty snapshot - Stack Overflow." [Online]. Available: http://stackoverflow.com/questions/25884801/ios-8-snapshotting-a-view-that-has-not-been-rendered-results-in-an-empty-snapshot. [Accessed: 30-May-2016].
[9] "iOS Crash Rate by Version." [Online]. Available: https://data.apteligent.com/ios-crash-rate-by-version. [Accessed: 31-May-2016].
[10] "Technical Note TN2151: Understanding and Analyzing iOS Application Crash Reports." [Online]. Available: https://developer.apple.com/library/ios/technotes/tn2151/_index.html. [Accessed: 31-May-2016].
[11] "Top 10 reasons your iOS and Android apps crash - TabTimes." [Online]. Available: http://tabtimes.com/top-10-reasons-your-ios-and-android-apps-crash-4783/. [Accessed: 31-May-2016].