Tweet District Design Document

Victor Frolov

November 24, 2015

1 Introduction

Tweet District is an app that lets you find and interact with hundreds of tweets in your area (or any location of your choosing), and provides a way to find out about parties, local events, sales, promotions, and job hirings, and its all shown on a map. Although it is currently a web app, it's use would be far greater on a mobile device. The purpose of this paper is to come up with a dream interface for this app, and use it as a guide during future development. Interaction design principles and theories are imperative to its development, and will range from usability measures, mental models, and many others.

2 Landing Page

Once a user downloads and opens the app, they will be greeted with a landing page with the logo, and app name that takes up a third of the screen. There will be one large button that allows the user to login to their existing twitter account (which will allow them to interact with the tweets, such as retweeting, sending direct messages, etc). Below the button will simply be clickable text that says "Use as Guest", allowing users to use the app without a twitter account. The logo, app name, button and text will be over a background color that will stay true to the app's theme. The background color will have opacity, and behind it there will be animations of users tweeting, partying, and shopping. Please see figure 1.

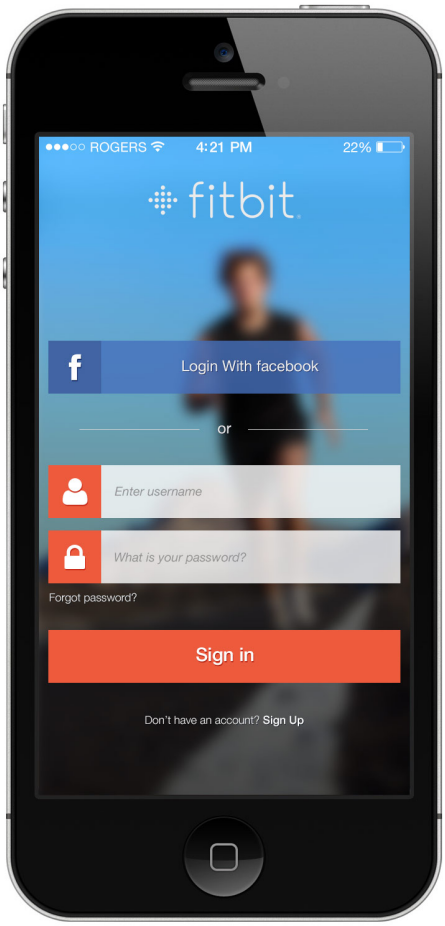


Figure 1: Landing/Login Page Example

The "Login with Facebook" would be moved down to the "Sign in" location, and all other boxes and buttons would be removed. Login with Facebook would be changed to Twitter, and below it there would be the aforementioned text of "Use as Guest". The Fitbit logo and Fitbit name would be of course replaced with the Tweet District Logo and Tweet District name.

3 Main Functionality

The app's main functionality can be broken up into three sections:

- Searching
- Displaying the tweets on a map
- Displaying the tweets in an ordered fashion that is useful

Once a user either logs in to their Twitter account, or continues as guest, they would be greeted

with the search page. All pages, including the search page, would have a navigational dock with buttons on the bottom of the screen, like in iOS's native Podcast app. Please see Figure 2 below.

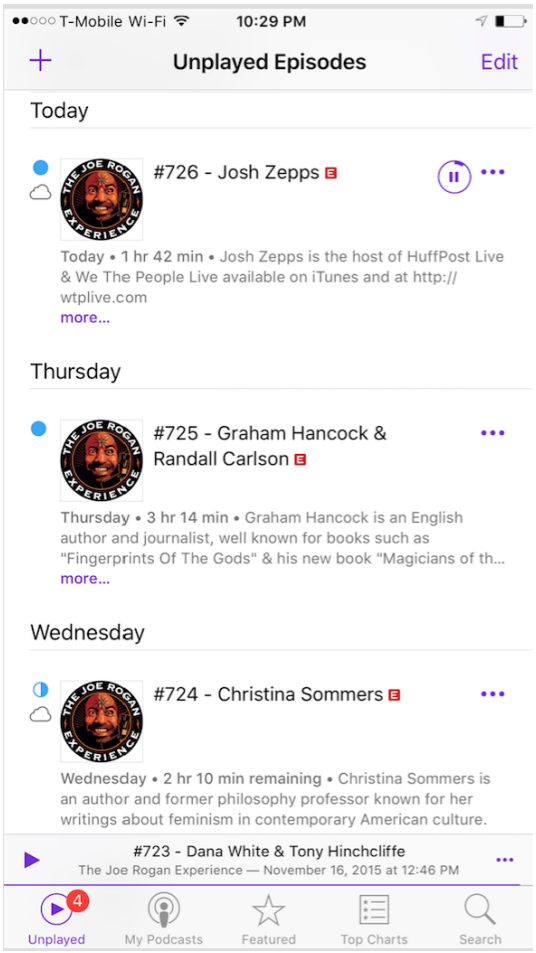


Figure 2: Navigational Buttons at the bottom of the Page

The buttons would be Search, Tweets, Map, Profile and a Sandwich. Search, Tweet, Map and Profile would bring the user to the respective page, allowing them to search, see found tweets, see tweets on a map, and access their saved tweets where they can interact with them (such as retweeting, direct messaging, sharing, etc). The sandwich button would be similar to Figure 4 (b) but the sliding animation would occur from the right side, towards the left. Please see the Advanced Search section for more information on the sliding animation. Once the slide occurs, the user would have the ability to be directed to an FAQ page, the ability to log out, and the ability to send a message to the response team.

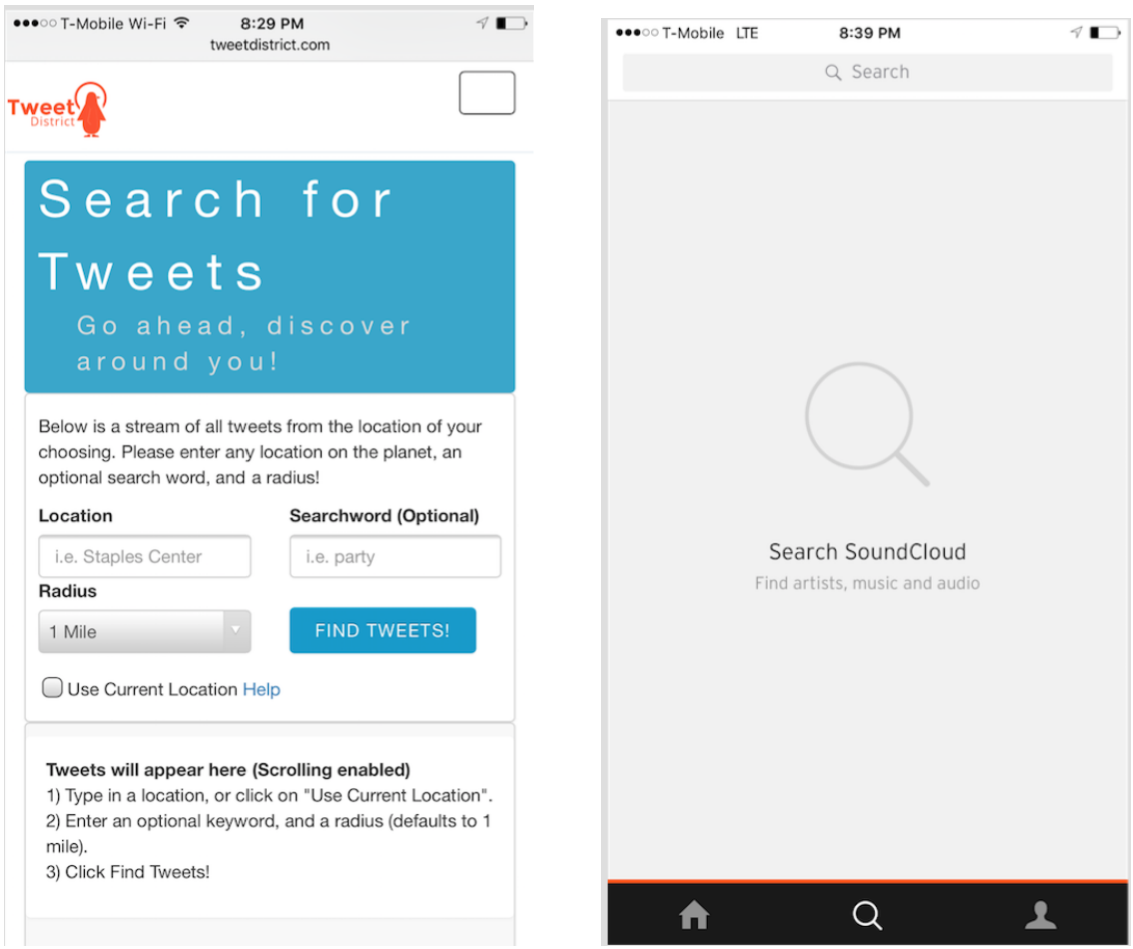
The button on the active page would glow, just as in Figure 2 where all the inactive buttons would not, visually showing the user where they are in the app.

3.1 Searching

Before diving into the search functionality, it is important to note that all of the suggestions and current limitations are directly correlated to the usability metric of efficiency. The process of searching should be quick, simple and almost instantaneous, with optional extra steps for extra precision and only if the user demands it. It should not be forced upon them.

3.1.1 Simplified Search

When I tried using the app on my mobile device when I was out and about with some friends, we found that there were a few issues impeding us from having a great experience. The first issue was having to type in a search word, check the current location box, and clicking search each and every single time we tried looking up something around us. See Figure 3.



(a) Current Search Appearance

(b) Simplified Search Appearance

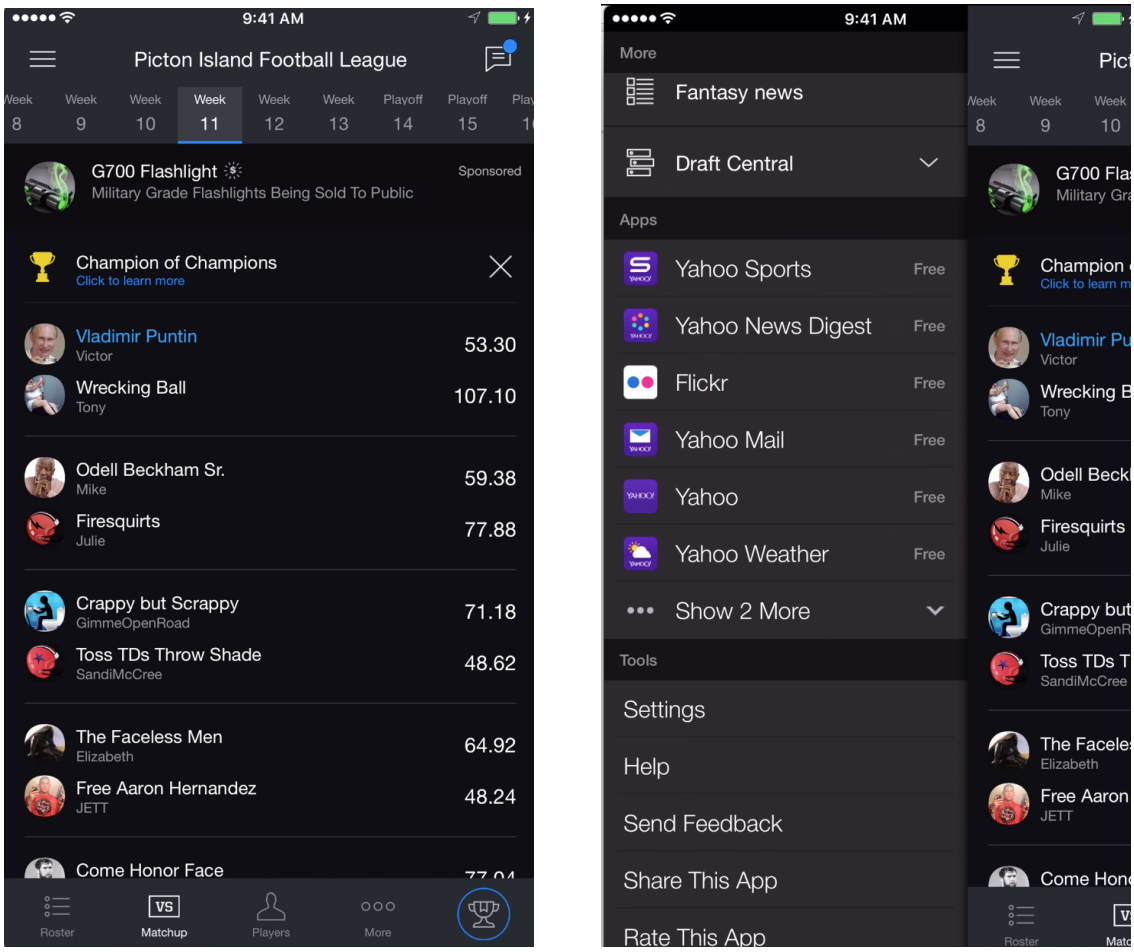
Figure 3: Old Search and New Search Template

In order to overcome the first hurdle of having to repetitively do a series of steps (we tried looking up about 10 keywords to find events around us, and although it ultimately worked, the

process was arduous), a simplified search, as in figure 3 (b), would be ideal. It would be preset to search a certain radius from the user's current location. That way they simply just type in a word or sequence of words, and hit search.

3.1.2 Advanced Search

Since the app is all about searching, it is important to have advanced search still available. Text near the top that says "More Search Options", once pressed, would animate the app to swipe to the right, similar to Yahoo Fantasy Football's top left button. See Figure 4.



(a) Before button was pressed

(b) After button was pressed

Figure 4: Advanced Searching

So before the button was pressed, the search tab would simply look like Figure 3 (b), and after the button is pressed, it would slide everything on the screen to the right, and would look close to what Figure 4 (b) looks like. But of course, instead of displaying Yahoo information, it would show optional fields such as:

- Custom radius

- Custom location (using geocoding and geodecoding)
- Excluding certain users (this was a small issue where sometimes a user would tweet 20 times about the same event)
- Enabling predefined searches
- Searching within a date range (so that a tweet from a few weeks ago that is no longer relevant doesn't show up)
- Only displaying one tweet per user
- Excluding words and hashtags.

The ability to save what the user input and make it become the "default" search should also be implemented. That way if the user always looks for "party" within a 3 mile radius of a specific location, they can do that with a single tap. Tapping on show less would slide the search back onto the screen, and would again look like figure 3 (b) above. The purpose of the animation is to draw attention to what is changing.

3.1.3 Predefined Searches

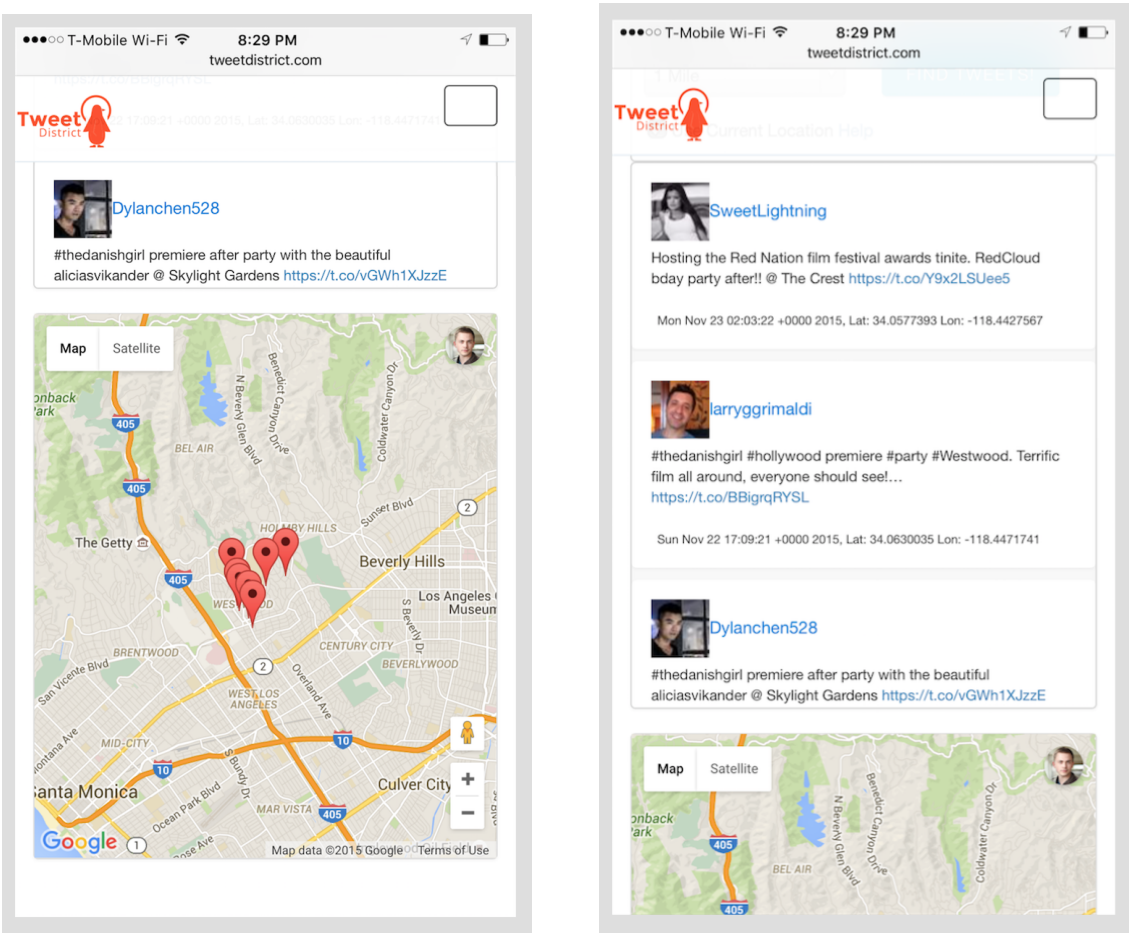
One final feature, which again would be to streamline searching, would be predefined searches. The user enables "predefined searches", in the advanced search options. This would then turn the search bar into a dropdown list of predefined searches. So, for example, if they were to select "party" in the dropdown list and hit search, it would automatically search the API for multiple keywords predefined by the app. Twitter's API allows the use of "OR" to string together multiple searches, so a potential combination could be "party OR frat OR houseparty OR happyhour OR bar OR club". Of course, custom predefined searches should also be implemented so that a user can look up series of words without having to type them fully out each time they do a search. Approximately 5 categories would be built in (party, events, sales, hiring, nightlife).

3.1.4 Twitter's API Get/Searches Limitations

As excellent as Twitter's API is, it has some limitations from Tweet District's perspective. They have the ability to ignore an ID's tweets, but, the ID must be known beforehand. Unfortunately, Twitter does not have the ability to cap or ignore users that have posted more than "x" tweets on a specific topic. This is a problem because, Twitter allows up to 100 tweets to be returned, and if there is a user spamming a certain topic, there is no clean way to ignore their tweets. One way would be to simply keep track of unique user ID's, and once there is repetition, to not append that tweet to the body, but of course, this would cause less than 100 tweets to be displayed. So if a user posted 60 times, and there was a total of 100 tweets, only 40 tweets would be displayed. A workaround would be to make an initial call, track the amount of unique ID's, and if a user, for example, has tweeted more than 5 or 10 times on a topic, to then do a second search, and ignore specifically his username in that search, for a more unique return.

3.2 Map

The map section would show a map with markers dropped that represent the location of every tweet found by the API. The way the app currently appears on a mobile device is very difficult to navigate, as it takes almost the full screen width. Scrolling to and away from the map was difficult, since when the user tries to scroll down the page but does so by putting their finger within the map box boundary, he would zoom in or out of the map. There was about a quarter inch of space both to the left and right of the map that allowed the user to scroll the page. See Figure 5 below. Navigating between the map that had markers for all the tweets, and the box that displayed all the tweets in list form did not feel intuitive, and was tedious. This would of course be solved with separating Search, Tweets, and Map into their own pages with the nav bar on the bottom for easy navigation.



(a) Current Map layout

(b) Another image of Map/Tweet layout

Figure 5: Advanced Searching

The grey boxes engulfing the two above images are to show what is not on the screen. So, the user would have to put their finger far left or far right on the screen, in that narrow white gap, to navigate the page without zooming on the map, or scrolling through the displayed tweets.

3.2.1 Markers Storing Data

Currently, the markers cannot be interacted with. Each marker should (and will) store the data of a tweet, so that when it is clicked on, it displays the user, and what they tweeted. It is hard to give an estimate of what the box width and length would be, but it should be large enough to display sufficient information, and small enough not to take up too much room on the mobile device. Some tweets would be cut off due to them being too long, or if they contain an image, and the cut off would be displayed with an ellipsis. Pressing anywhere on the box would make the box get larger, displaying all of the information. When the box is larger and displaying all of it's information, the users would also be able to click on a user ID to go to their twitter page, press on links to go wherever they link to, retweet the tweet, star it, create a new tweet with the same hashtag, and direct message the user who tweeted. Pressing again would make the box shrink to it's original size (as long as the user does not press on a link or a user ID).

Below is an example of how a marker would contain data. It is a screenshot of Padmapper, and a clicked marker that display information about a room being leased. See figure 6

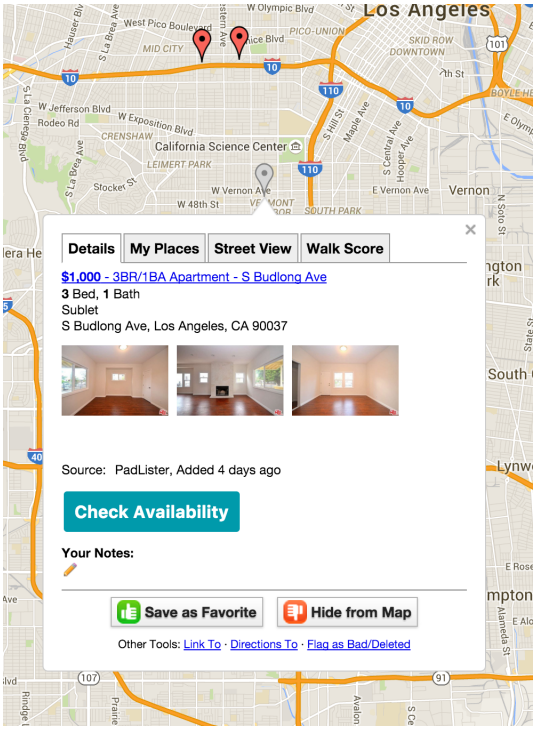


Figure 6: A marker displaying the information it contains in a box

Note: the following paragraph also applies to what the Tweets would visually appear like in the "Tweets" section, and as it is written here, will not be reiterated again. The tweet organization should be minimal and similar to what is currently implemented. All the data will be stored in a white box. The user profile will be far left with subtle rounded corners. Padded from the left, and from the right. Padding top would be aligned with the padding of the username, which would be to the right of the image (containing some padding between image). The user's name will be bolded. The user's ID will be on the same line as their name, separated by a space, but in smaller

font that is grey. After the user ID, the time the tweet was posted would be displayed, also in grey font and smaller size. Below this line, the actual tweet will be seen. Please see figure 7 for what the tweet should look like.

Figure 7: An example of a tweet's appearance

Notice the ability to interact with the tweet. A user should be able to reply, retweet, star, delete and direct message the user. Basic principles of presentation are being incorporated, with bold text attracting the user's eye, grey and harder to see text not being as important. A minimal approach for a sleek look is the goal, with just a thin grey line to separate all the tweets. During testing, attempts to make the boxes more colorful were made, but they were far too distracting.

3.2.2 Marker States

Similar to general button guidelines, markers should be in obvious states based on what the user has done. At first, all markers will be of the same color, let's for now assume that color will be red. Once a user clicks on a tweet, the marker turns grey, to indicate it was already viewed. If a user decides to delete a tweet, it will remove it form the map, and from the Tweet list. If a user decided to save the tweet, it will now glow gold. The ability to delete the tweet will be locked, and it will only be deletable after the user removes the star.

3.2.3 Force Touch Implementation

Force touch should also be implemented, so that if a user hard-presses on a marker, it instantaneously shows them all the information, and the larger box that can be interacted with.

3.2.4 Searching Map Location

If a user doesn't know the address of a location, he should still be able to search it. There would be an unobstructive button in the top right corner of the map that says "search map location". If clicked, this would automatically jump to the search page. A thin banner at the top would display "map location saved for search", and it would disappear after 5 seconds. Now, when the user searches, the center location of the map that he had shown on the device would the location searched, and the radius would be the furthest distance from the center to an edge of the screen. This could be easily cleared in the advanced settings if the user wanted to return to searching his current location.

3.3 Tweets

Tweets are at the core of this app. When on the tweets page, it will simply display each tweet and all their information in white boxes, and approximately 4 to 5 tweets should be seen on the screen. the user can then scroll through the list to see what they found.

Tweet layouts should be simple, minimal, and easy to distinguish. All data pulled from the API should be properly structured. Please see Markers Storing Data section for a description and an image of what the tweets should appear like.

3.3.1 Deleting/Saving Search Results

The ability to flick individual tweets that are of no interest should also be implemented. That way, if a user gets 100 tweets back from their search, if one or some of them are of interest to them, they can swipe it left, which would not only remove the tweet from the list, but also remove a marker on the map associated to it. On the other end of the spectrum, they should be able to save tweets so that they can interact with them later, by swiping right. This would be saved and found under their "profile" tab. That way, they can at a later time share the tweet with friends, retweet it, direct message the user, or simply save it for whatever reason they want. Visually, this would look identical to the process of deleting or archiving mail on most mail apps, and would be accompanied by two different noises for each action. See Figure 8.

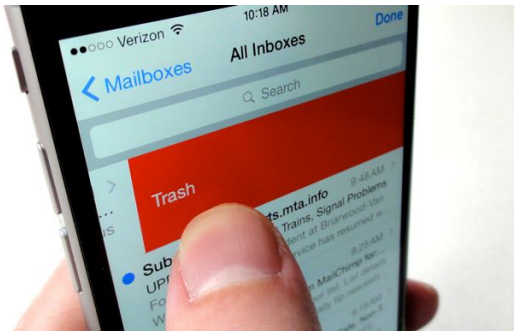


Figure 8: Swiping to delete

In case it is not obvious, as the user swipes, the entire box that contains the tweet would slide in a color (red or blue, depending on which action is occuring), and the more they swipe, the more the color slides in, until they finally reach the edge of the screen, on release, the tweet would be deleted/saved.

3.3.2 Map Integration

Navigating between tweets and the map page should be effortless and streamlined. A user should be able to press on a tweet (at the moment, I am thinking that pressing anywhere in the tweet box would bring you to the map, but I also want to test having a button that says "Show on map" within the tweet box. The problem with the "Show on map" is that every tweet will have it, and it will dirty up the screen, but if it is not there, the learnability of clicking on a tweet to get to the map part will be bad, as it is not an apparent action). Just like the markers on the map will turn grey once a user pressed on one, the same would happen for the tweets in the list. The box would

change colors to clearly show the user has already looked up that tweet. And similarly to a gold hue to the marker if the user saved the tweet, the box would turn gold to indicate that it is a saved tweet, and swiping to delete it will not be possible.