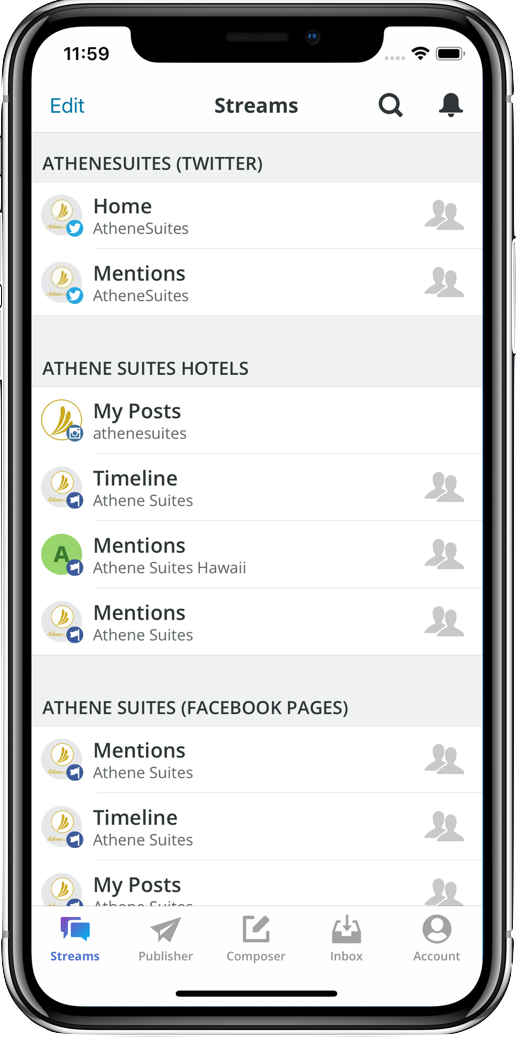
Mobile Applications Design Document for

“TweetCards”

TweetCards is built to enhance the user experience whilst using twitter. Previously if the user wanted to only see a certain set of users’ tweets, they would have to create a new account to follow those specific users. Unless they used ab alternative such as the in browser TweetDeck or one of the more social marketing focused apps such as HootSuite.

The focus of this App will be to make twitter feel more like the classic RSS feed. Where you can just specify a set number of users or hashtags you would like to be updated on without having to create an account. The user could then set up multiple of these Cards for different specialised timelines. For example, one card could be for following local news accounts and then a separate one for status updates for the busses in the user’s city. This allows the user to quickly be able to check the whether or not the busses are running in the morning without having to filter through the news at the same time and vice versa.

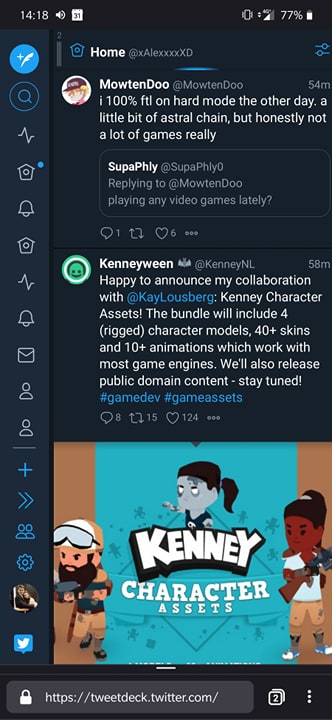
# Research

Whilst looking for related products to the proposed idea I have found some that do operate in similar manners but have some downfall or differ in some way that doesn’t quite line up to the specific use case TweetCards fulfils.

Firstly, there are the social marketing apps such as HootSuite. These apps revolve around pushing messages out to all social media platforms at once (Facebook, Twitter, etc) and keeping all customer facing social media platforms for a company updated with the latest information. These also include analytics for the interactions on each social media platform. These only really focus on keeping the specific users’ feeds across social media platforms updated, as opposed to collating tweets from multiple accounts on a single feed. As seen in Figure 1 HootSuite supports multiple social media accounts which will not be required in TweetCards. HootSuites interface also looks very outdated which will be improved upon in TweetCards.

Secondly, there are apps that focus on managing multiple accounts for twitter such as Seesmic. Allowing the user to technically curate custom feeds through the use of multiple accounts, however, if the only thing you wanted was the custom feeds and none of the account specific features then this is a very cumbersome way of getting these custom feeds.

Figure : HootSuite account overview



Finally, the main inspiration for this project, TweekDeck. TweetDeck is a website which allows the user to create multiple curated feeds of various twitter accounts and hashtags and view them all simultaneously. Whilst this website is more feature rich and fulfils the use cases TweetCards proposes it has a fatal flaw. Due to being designed with primarily desktop in mind it feels very clunky to use on a mobile device. The elements used to distinguish timelines are repeated multiple times as seen in Figure 2. Making it more difficult to see which timeline is which at a glance and requiring the user to open an extra menu. TweetDeck’s feature set is also a double-edged sword, whilst it does contain more features and use cases it also ends up being cluttered and confusing to users using it for the first time. Making the learning curve a longer and steeper process. By keeping TweetCards simple and fulfilling one purpose well, it will have a much shallower learning curve.

TweetCards’ will target professionals aged 20 onwards. The primary reason is due to the app allowing the professional using it to keep updated on various topics that are related to their profession or interests. Through this the design of the app is going to be focused around three main paradigms; ease of use, clarity and fast navigation. The reasoning behind these paradigms is that professionals generally do not have much free time throughout their day. By creating TweetCards with that in mind it allows for an app which will be a practical addition to their smart phone.

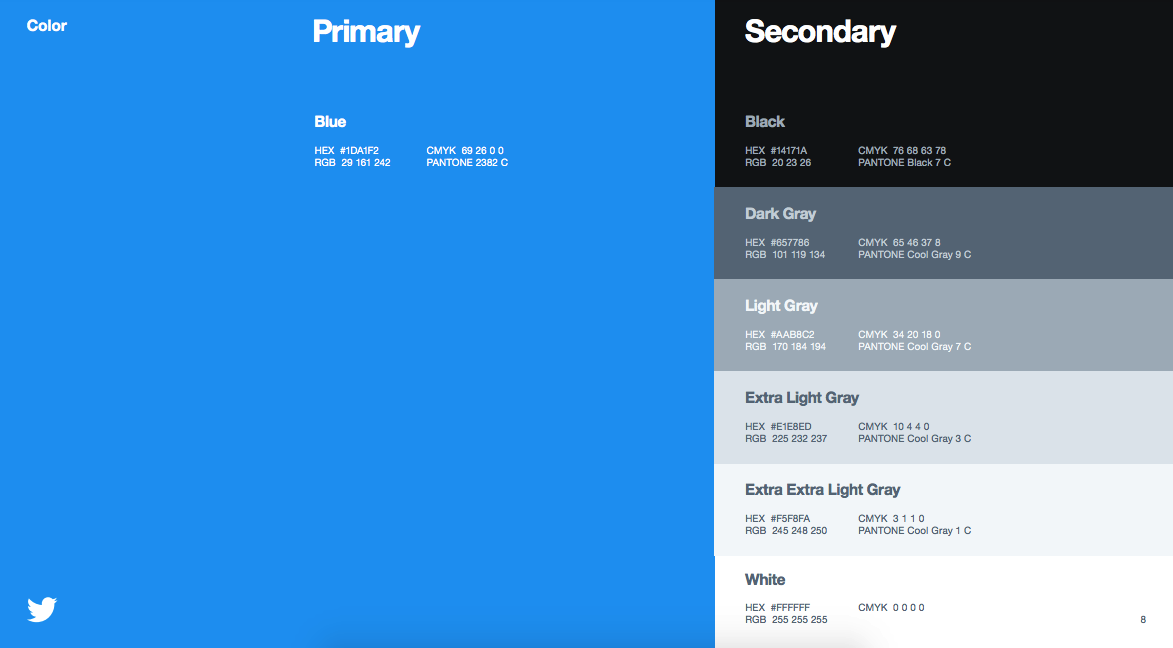
Figure : TweetDeck's Mobile Interface

# Design

TweetCards design will utilise a minimalistic style, making sure to only populate the screen with elements that will have a practical use. This ensures that clarity of app is maximised and that the app does not get in the way of using the app itself.

## Colour and Typeface

Since TweetCards will purely focus on the Twitter experience; to keep it familiar to users who already use Twitter, TweetCards will adopt Twitters design specifications of Colour and typeface.

TweetCards will mostly utilise “Extra Extra Light Gray” for background and whitespace. This is mainly to reduce the harshness of the contrast between the background of the app and the use of “Light Gray” in other parts of the app which are mentioned below in the Layout Section. Anything buttons or text that are used for navigating through the app or any accenting in the app will utilise Twitters primary “Blue” and this will be the case regardless of dark mode. The dark mode version of the app will also utilise Twitters dark mode colour palette. It will use # 141d26 for background and whitespace in the app but where elements were previously contrasted by going darker than the background colour this time, they will be contrasted by going lighter to #243447. The colour palettes can be seen below in Figure 3 and 4.

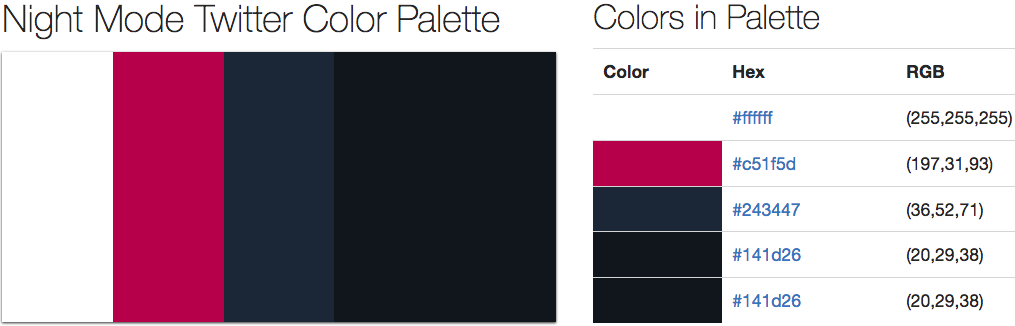
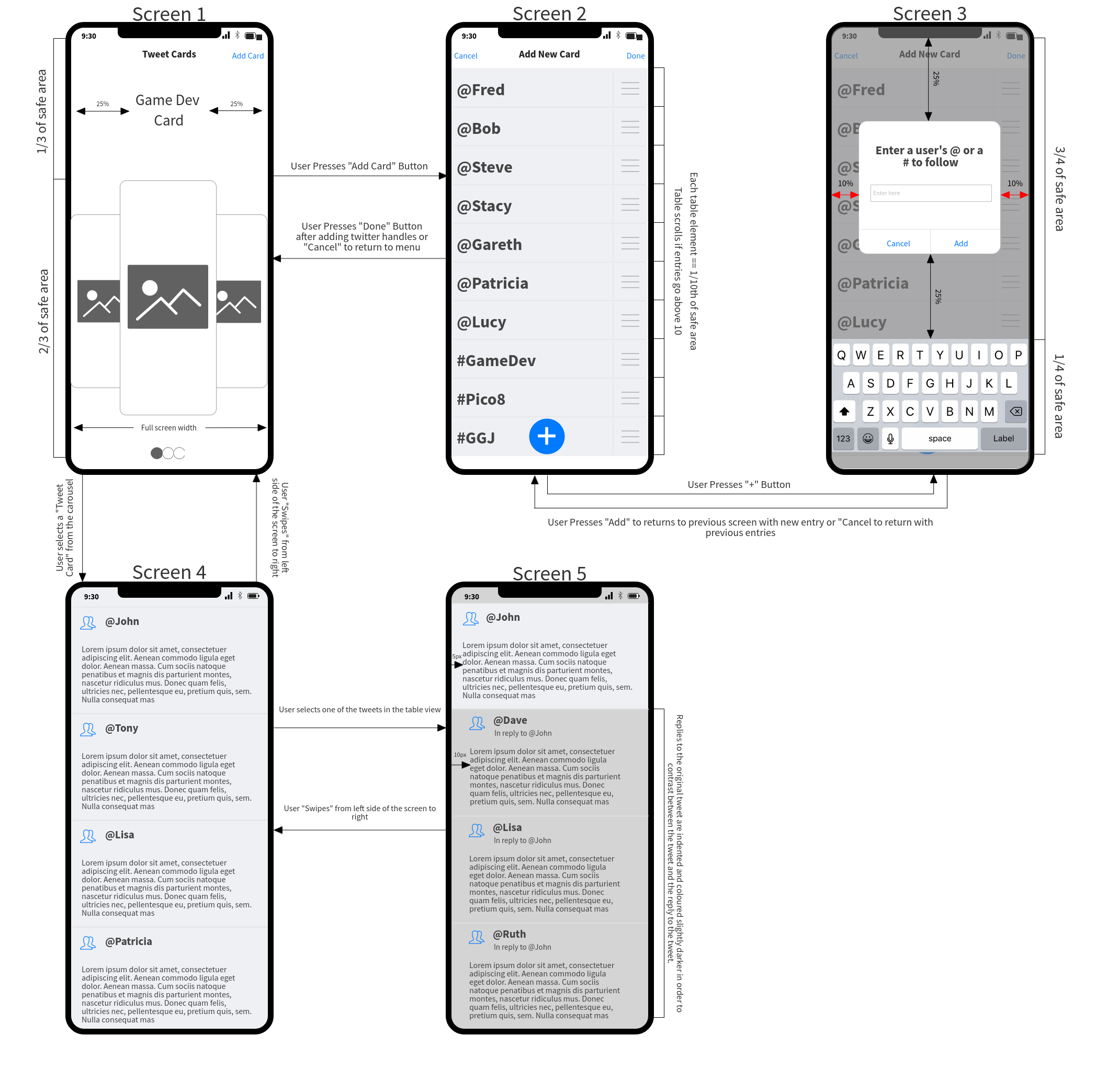


Figure and 4: The Light and Dark mode colour palettes for Twitter

The typeface of the app will also follow Twitters brand guidelines. This means that for Tweet text, timestamps and users name will use “Helvetica Neue Bold”. For the users @, “Helvetica Neue Regular” will to used.

## Layout

In order for the user experience of the app to feel natural there has been a focus on using gestures and animations. Gestures will be used for navigating around the app and the animations will act as affordances on which gestures to use to navigate. For example, a new view being displayed by sliding onto the screen from the right side and being dismissible by using a “Swipe” gesture from the left side of the screen to the right side of the screen. The technical wireframe for the app can be seen below:



### Screen 1

Screen 1 is the entry point to the app and the main screen for navigating between cards. The bottom two thirds of the screen are taken up by a carousel of the users’ cards. The carousel items can either be one of two things. A cached screenshot of last time the user was viewing the timeline on the card or it can be a picture of their choice. The latter choice would help the user identify the card they are currently looking at with a glance as opposed to reading the title for said card in the upper third of the screen. To change the title of a card a simple long press on the title will bring up a keyboard to edit it. In the top third of the screen there is also the option to add a new card with the “Add Card” button. This will then take you onto screen 2.

### Screen 2

Screen 2 is where the user defines what Twitter handles and hashtags they want to be updated on in this specific card. Every new Handle / hashtag added will be appended onto the end of the Table view in a new row. To remove an entry that the user added they will simply have to Swipe from Right to left on the row they want to remove in a similar fashion to the alarm Table view found in the iOS clock app. This makes the action fairly intuitive as its already been used elsewhere in the base OS itself for the same purpose. By pressing the “+” button the user will then be taken to Screen 3 to specify a new entry for the Table view.

### Screen 3

Screen 3 is a very simple screen. In essence it’s just an overlay to Screen 2. It has a text field where the user can enter the Twitter handle or hashtag they would like to add to the card and once the user presses “Add”, the pop-up disappears. Screen 2 is then returned to focus with the new added element in the Table view. If “Cancel” is pressed, Screen 2 is returned to without adding any elements to the Table view. This screen intentionally dims the background to bring focus to the Pop-up, so the user knows this is the element they need to diverge their attention to.

### Screen 4

Screen 4 is the first screen in which content is requested from twitter. To navigate to this screen the user selects the desired card on Screen 1. The card then expands outwards to fill the entire screen with a boilerplate Table view which is Screen 4. This screen is initially displayed with a boilerplate Table view because even if the data is yet to be received from Twitter, it allows the user to get their bearings of the screen and how it will be laid out once it does load. This screen will request and load the latest 100 Tweets from Twitter from the Twitter handles and hashtags defined when creating the card in the rows of the Table view. Once the user scrolls down around 80 Tweets, another request will be sent to get the next 100 Tweets and so on. The Table view’s rows will dynamically scale depending on the size of the content. For example, the row will be significantly larger if it contains an image in comparison to a Tweet that contains only 4 words. By using the animation of the card expanding into view the impression has been given that this screens view is simply stacked on top of the previous one. Therefore, to navigate back to Screen 1 the user dismisses this screen simply by doing a Swipe gesture from the right side of the screen to the left. The user can also view replies to a Tweet by pressing a Tweet in the timeline, this will then take them to Screen 5.

### Screen 5

Screen 5 in essence is very similar to Screen 4 with a few key differences. The top fifth of the screen is taken up by the Tweet previously selected in Screen 4. This element is anchored and does not move as it is the root of the responses. In the lower 4/5’s of the screen is a Table view which is built up of the replies to the root Tweet. This Table view will work similarly to the one in Screen 4 where the replies will dynamically resize depending on the content and gets populated with more content the further the user scrolls through it. This Table view has some differences to the one found in Screen 4 and all are in appearance. The most obvious change is that the background colour of each row is contrasted to the root Tweet. This makes it much more obvious to the user that they are looking at a reply to a Tweet as opposed to a root Tweet. To further reinforce this, the content in the row is also indented further in towards the middle of the screen in comparison to the root Tweet. Finally, there is an extra element on each row which acts as a subheading to the repliers Twitter handle specifying which user they are replying to. This element will use the “Helvetica Neue Regular” typeface as it is not meant to specifically attract the attention of the user. By tapping a reply in this screen, it makes a new view of this screen but with the reply as the root Tweet and the replies to the reply in the Table view below it. This can be nested as many times as necessary. To return to the previous screen the user will Swipe from Right to left to dismiss the current view and return to the previous in the stack.

# Requirements

To assist in the creation of this app, various existing resources will be utilised. This will help speed up production of the app and allow for quicker iteration of certain elements if changes that diverge from the design document are required. For example, if something is found to be counterintuitive in user testing then that aspect of the app can be adjusted.

## CocoaPods

CocoaPods are small libraries that extend upon the existing Swift codebase. Some of these make certain aspects of the app much easier to develop for as the interface is either built for you or acts as a wrapper around more difficult interfaces to simplify them. The ones planned for this app can be seen below:

### Alamofire

Alamofire is a networking library that wraps around Swifts networking protocol. It simplifies the GET and POST requests into easy to understand function calls that are chainable, which will save on a lot of code. Alamofire will be used primarily for the GET requests received from the Twitter API. This returns JSON strings which then can be processed to populate various parts of the app such as the Tweet Table views.

### SwiftyJSON

SwiftyJSON is a JSON library that allows for much more ease when working with JSON objects. The difficulty mainly arises as Swift is strict about types but by nature JSON files are quite the opposite. SwiftyJSON allows you to construct a JSON object which you can simply access elements in the JSON object using [] and either a key as a string or an index if its an array. This will make handling the data received back from Twitters API much each as SwiftyJSON can serialize the response into an object which we can then just query for the data we require.

### AlamofireSwiftyJSON

AlamofireSwiftyJSON is an extension to the Alamofire library, allowing the developer to automatically serialize a response from a GET request into a SwiftyJSON object, saving a couple extra steps of creating a SwiftyJSON object and passing it the string received from the request.

### TweenKit

TweenKit is a simple but powerful animation library that lets you animate various screen elements with different speeds, easings and the potential for scrubbing between animations. This will make many of the screen transitions planned possible and will be much easier to manage as opposed to lerping values with lots of different timers for each animated element.

## Assets

# Strategy