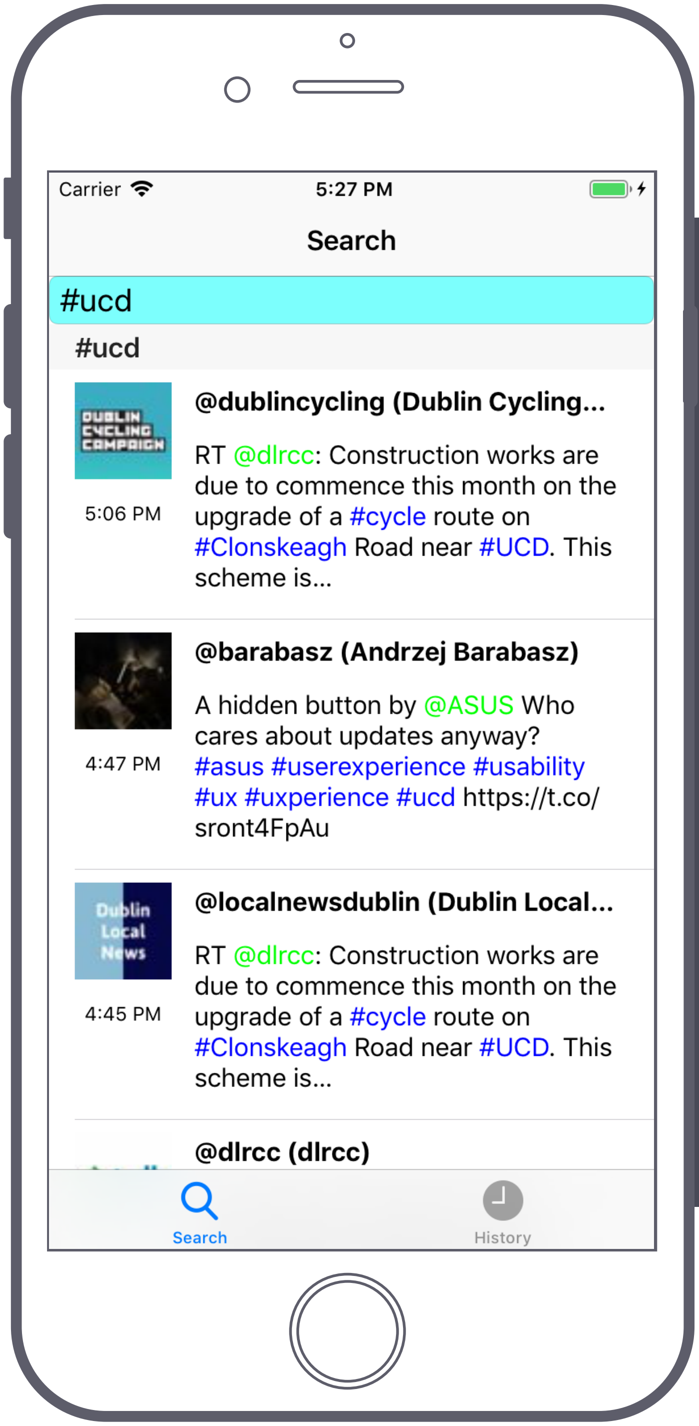
IOS: TweeterTags

2019

Student Number: 18204188,
Student Email: gregoire.cousin@ucdconnect.ie



#ucd

#ucd

@dublincycling (Dublin Cycling...

5:06 PM

RT @dlrcc: Construction works are due to commence this month on the upgrade of a #cycle route on #Clonskeagh Road near #UCD. This scheme is...

@barabasz (Andrzej Barabasz)

4:47 PM

A hidden button by @ASUS Who cares about updates anyway? #asus #userexperience #usability #ux #uxperience #ucd https://t.co/sront4FpAu

@localnewsdublin (Dublin Local...

4:45 PM

RT @dlrcc: Construction works are due to commence this month on the upgrade of a #cycle route on #Clonskeagh Road near #UCD. This scheme is...

@dlrcc (dlrcc)

Twitter Bot

Application to search for tweets using tags and usernames.

Twitter API

Twitter API provides access to the tweets, mentions, media, tags and urls for each tweet.

Table of Contents

TweeterTags Introduction

TweeterTags is an iOS application which is intended to allow users to search for tweets using "hashtags" or "usernames" as keywords. The tweets which are in turn presented to the users in a tableview then highlight the associated usernames, mentions, URL's, screen names, images and hashtags. This aids users in sifting through tweets while also giving them the option to click on them. Additionally, the application stores information about the previous 100 searches made by the user. Finally, the app allows users to find and show images associated with these tweets.

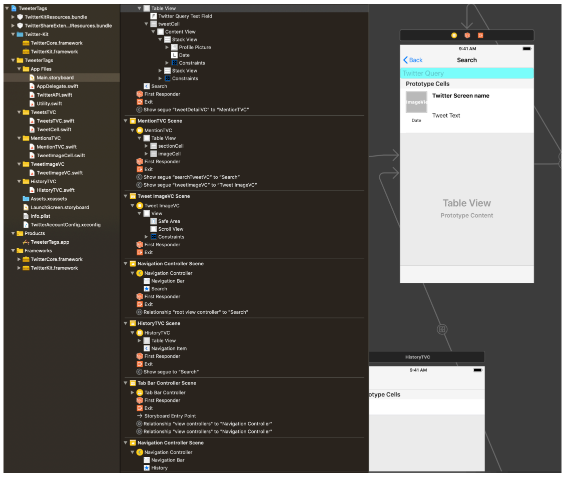
Technology and Design

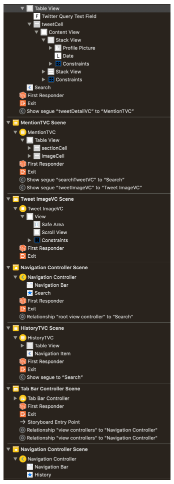
The TweeterTags application is an iOS application created in Swift 4 and the IDE used is Xcode. The application consists of many different components that are for ease of function for the user. The application design is made in the Storyboard of Xcode. The main storyboard uses a tab bar controller to present two main screens comprising of :

a) The tweets screen containing the tableview of tweets

b) The history screen containing the previous 100 searched by the user

App Design





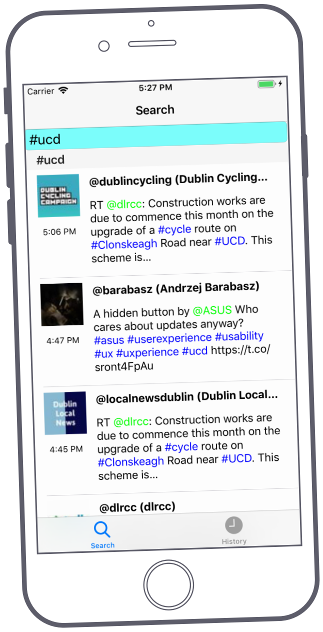
Tweets TableViewController

The Tweets screen is UITableviewController embedded in a UINavigationController. A search bar is included at the top to facilitate users specific tweet queries. The TableView has UITableViewCell subclasses called "TweetTVCell" which outputs information about the tweets. This information includes screen name, tweet text, the date as well as the user's avatar. The design of the cell is based on UILabels and UIImageView.

The screen name and tweet text are held together in a StackView. The UIImageView and the date Label on the left are held together in a separate StackViews in order for them to facilitate the presentation of the main view. Clicking on the TweetTVCell performs a "Show Segues" to the "Mentions" screens.

Mentions TableViewController

The Mentions screen is a UITableviewController which is connected to the main Tweets screen. This display contains an image and information cell to show the relevant information with a section header. There are 2 segues connected to this screen.



@dublincycling (Dublin Cycling...

RT @dlrcc: Construction works are due to commence this month on the upgrade of a #cycle route on #Clonskeagh Road near #UCD. This scheme is...

5:06 PM

@barabasz (Andrzej Barabasz)

A hidden button by @ASUS Who cares about updates anyway? #asus #userexperience #usability #ux #uxperience #ucd https://t.co/sront4FpAu

4:47 PM

@localnewsdublin (Dublin Local...

RT @dlrcc: Construction works are due to commence this month on the upgrade of a #cycle route on #Clonskeagh Road near #UCD. This scheme is...

4:45 PM

@dlrcc (dlrcc)

The first one is connected to the Tweet Image screen which is presented as "Show Segue" when the user clicks on it. The other is connected back to Tweets screen which is also presented as "Show Segue" upon clicking on it. It must be noted that the second segue is not directly connected to a cell but to the ViewController.

TweetImage TableViewController

The Tweet Image screen consists of a UIScrollView to allow the user to zoom into the image. The constraints are set to cover full screen apart from the TabBar and Navigation.

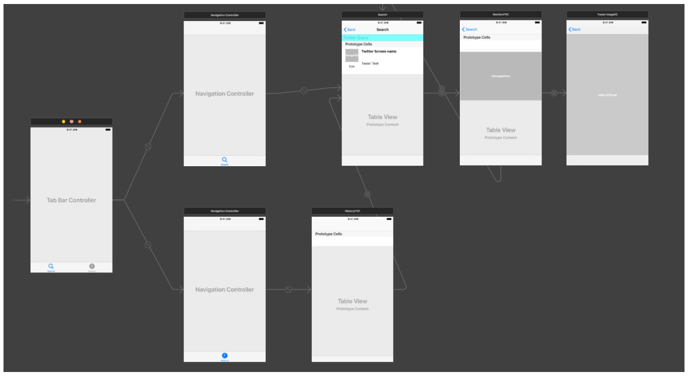
History TableViewController

The History display is made of a TableViewController and shows the users search history. This is embedded in a UINavigationController and has one cell. Clicking on the cell will perform a show segue to the Tweets screen.

Components

An iOS application consists of different components which work together to make it work. TweeterTags components are as below

- UITabbarController
- UINavigationController
- UITableViewController
- UITableViewCell
- UITextField
- UIScrollView
- UIImageView
- UILabel



UITabbarController

UITabbarController allows the application to have tabs at the bottom. It embeds up-to 5 ViewControllers and a user can switch between them. This class is generally used as-is, but may also be subclassed. TweeterTags has 2 tabs which are a Tweets screen and a History screen.

UINavigationController

A navigation controller is a container view controller that manages one or more child view controllers in a navigation interface. TweeterTags has 2 UINavigationController which shows the Tweets screen and History screen. This class is generally used as-is and allows basic functionality but can be also subclassed for expansion. In TweeterTags, whenever a user moves from one screen to another, the UINavigationController is used to perform the functionality.

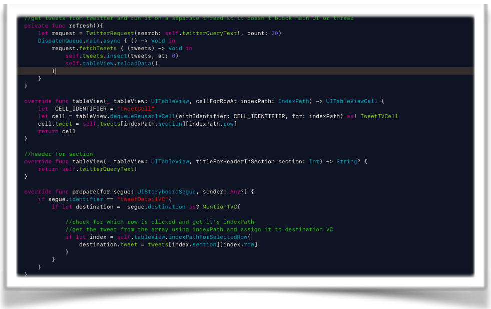
UITableViewController

A table view displays a list of items in a single column. UITableViewController is a subclass of UIViewController which allows users to scroll through the content. TweeterTags currently has 2 such view controllers.

Tweets screen uses UITableViewController to show all the tweets. It shows the tweets using independent cells which are a class of UITableViewCell. Clicking on a cell allows the user to activate the segue. The Mentions screen shows the information related to the tweet and uses a UITableViewController. The information embedded includes media, urls, hashtags and other mentions. The tweet object is sent to the Mentions screen during the “prepareForSegue” method which basically allows information to be sent across two view controllers. The History screen also uses the UITableViewController and shows the search history performed by the user.

UITableViewCell

This component includes properties and methods for setting and managing cell content and background including text, images, and custom views. In TweeterTags, multiple screens are using this concept. The Tweets screen uses UITableViewCells to show the contents of a tweet. In order to do that, it is subclassed to TweetTVCell to achieve the functionality.



```
//get tweets from tweitter and run it on a separate thread so it doesn't block main UI or thread
private func refresh(){
    let request = TwitterRequest(search: self.twitterQueryText!, count: 20)
    DispatchQueue.main.async { () -> Void in
        request.fetchTweets { (tweets) -> Void in
            self.tweets.insert(tweets, at: 0)
            self.tableView.reloadData()
        }
    }
}

override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let  CELL_IDENTIFIER = "tweetCell"
    let cell = tableView.dequeueReusableCell(withIdentifier: CELL_IDENTIFIER, for: indexPath) as! TweetTVCell
    cell.tweet = self.tweets[indexPath.section][indexPath.row]
    return cell
}

//header for section
override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    return self.twitterQueryText!
}

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "tweetDetailVC"{
        if let destination =  segue.destination as? MentionTVC{

            //check for which row is clicked and get it's indexPath
            //get the tweet from the array using indexPath and assign it to destination VC
            if let index = self.tableView.indexPathForSelectedRow{
                destination.tweet = tweets[index.section][index.row]
            }
        }
    }
}
```

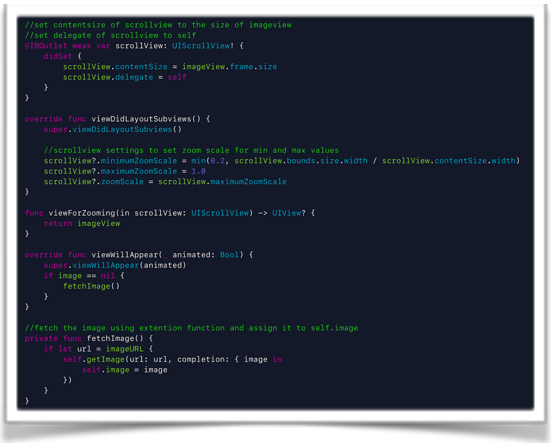
In cellForRow method for the UITableViewController, each tweet is assigned to a tweet variable in TweetTVCell which then updates the contents of the cell. Mentions and History uses this to show the details of a tweet in different sections and the users search history respectively. It's important to note that the UITableViewCell is not subclassed.

UITextField

Textfields are use to gather text-based input from the user using the onscreen keyboard. In this case, Tweets screen has a textfield at the top to get the user input. The user input is used to search for tweets. The textfield also has delegate methods in the Tweets screen which are used to detect when to close the keyboard presented on the screen.

UIScrollView

UIScrollView is the superclass of several UIKit classes including UITableView and UITextView. In TweeterTags it is used in the TweetImage screen which allows the user to look at the image and zoom in and out.



```
//set contentsize of scrollview to the size of imageview
//set delegate of scrollview to self
@IBOutlet weak var scrollView: UIScrollView! {
    didSet {
        scrollView.contentSize = imageView.frame.size
        scrollView.delegate = self
    }
}

override func viewDidLayoutSubviews() {
    super.viewDidLayoutSubviews()

    //scrollview settings to set zoom scale for min and max values
    scrollView?.minimumZoomScale = min(0.2, scrollView.bounds.size.width / scrollView.contentSize.width)
    scrollView?.maximumZoomScale = 1.0
    scrollView?.zoomScale = scrollView.maximumZoomScale
}

func viewForZooming(in scrollView: UIScrollView) -> UIView? {
    return imageView
}

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
    if image == nil {
        fetchImage()
    }
}

//fetch the image using extention function and assign it to self.image
private func fetchImage() {
    if let url = imageURL {
        self.getImage(url: url, completion: { image in
            self.image = image
        })
    }
}
```

Code

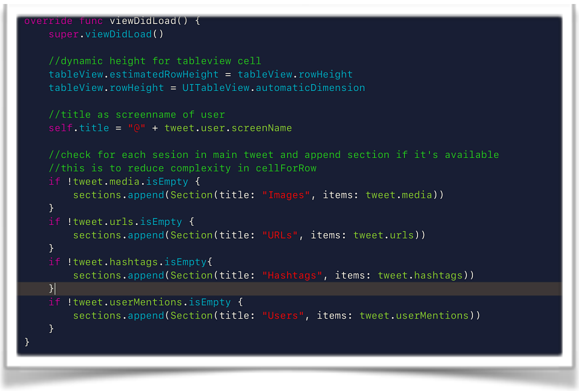
This section will explain the key code components used to develop the application.

TweetsTVC

The Tweets screen has a class name of TweetsTVC. It has a main object tweets which is an array. After opening the view controller, a refresh( ) method is called to load the tweets. This method is called each time a user inputs a new search query in the twitterQueryTextField. A saveHistory method is used to store the user search history every time new tweets are fetched. This information is stored in UserDefaults in the form of an array. This UITableView Controller used a custom cell class of TweetTVCell.

In cellForRow( ) tableview method, a tweet is assigned to the cell object which then runs showTweetInformation( ) to update the cell with the information. The information in the cell is highlighted by multiple colors. To do that, the range of each word that should be highlighted is assigned to an object of NSMutableAttributedString where the color and the range to highlight are mentioned in the tweet text.

MentionTVC



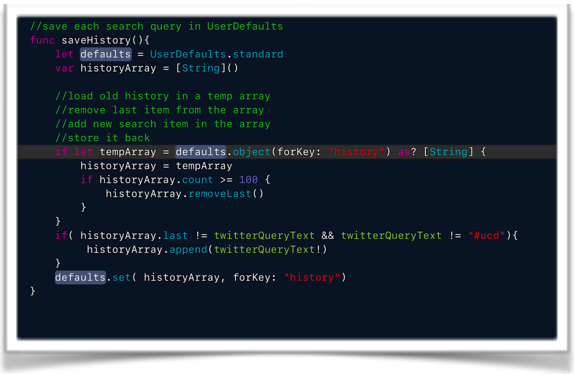
```
override func viewDidLoad() {
    super.viewDidLoad()

    //dynamic height for tableview cell
    tableView.estimatedRowHeight = tableView.rowHeight
    tableView.rowHeight = UITableView.automaticDimension

    //title as screenname of user
    self.title = "@" + tweet.user.screenName

    //check for each sesion in main tweet and append section if it's available
    //this is to reduce complexity in cellForRow
    if !tweet.media.isEmpty {
        sections.append(Section(title: "Images", items: tweet.media))
    }
    if !tweet.urls.isEmpty {
        sections.append(Section(title: "URLs", items: tweet.urls))
    }
    if !tweet.hashtags.isEmpty{
        sections.append(Section(title: "Hashtags", items: tweet.hashtags))
    }
    if !tweet.userMentions.isEmpty {
        sections.append(Section(title: "Users", items: tweet.userMentions))
    }
}
```

The Mentions screen has a class name of MentionTVC. It has an object name tweet which comes from the Tweets screen's prepareForSegue( ) method. This screen uses a sections array object to store information about media, urls, hashtags and mentions. It's type is a custom struct called "Section" and stores the section name and the text to show in an array. The tableview used a custom cell class of TweetImageCell to show the image. It allows the aspectRatio to be set as well as the scale of the image so the image shows fully in the cell. The sections for urls, hashtags and mentions doesn't use any custom class.

TweetImageVC

TweetImage screen has a class name of TweetImageVC. It has a private property named imageView which is added in scrollView as a subview. It also has a variable for imageURL which comes from the Mention screen's prepareForSegue( ) method. During the viewWillAppear( ) method, fetchImage( ) is called to download the image and store in image variable which is then shown in the scrollView.

HistoryTVC



```
//save each search query in UserDefaults
func saveHistory(){
    let defaults = UserDefaults.standard
    var historyArray = [String]()

    //load old history in a temp array
    //remove last item from the array
    //add new search item in the array
    //store it back
    if let tempArray = defaults.object(forKey: "history") as? [String] {
        historyArray = tempArray
        if historyArray.count >= 100 {
            historyArray.removeLast()
        }
    }
    if( historyArray.last != twitterQueryText && twitterQueryText != "#ucd"){
         historyArray.append(twitterQueryText!)
    }
    defaults.set( historyArray, forKey: "history")
}
```

History screen has a class name of MentionTVC. It's purpose if to show the last 100 search requests made by the user. A method historyArray( ) retrieves the history from the UserDefaults every time user visits this screen. The order of historyArray is reversed after retrieving so that the user sees the latest searches they made.

Conclusion

In conclusion, each of the proposed specifications of the assignment highlighted in the introduction were achieved. However, I had a difficult time with the app design, particularly with constraints and their coordinates, the mapping of views and making the link between each screen. I also found it challenging to gather and reorganize the api's data in a table view, working with cells and coming up with logistical algorithms to conduct the inner working of each function.

Due to the flexibility of each function, there is ample opportunity for this application to be expanded and improved upon in a more creative way, which would be in my interest to pursue.

This assignment helped me understand how to build and construct a functional modern application, understanding how model view controllers worked and benefiting from the skills learned in class as well as what Xcode has to offer.