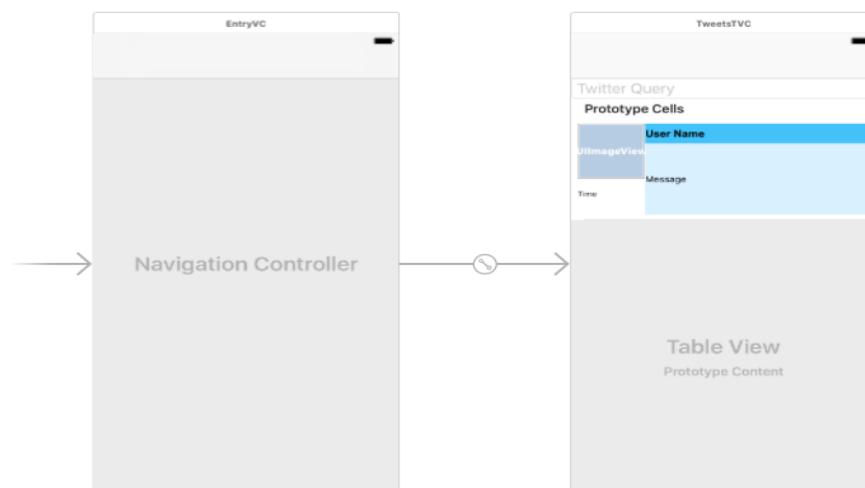# REPORT – TwitterTags
Part 1

OBJECTIVE:

Build an App to retrieve tweets from Twitter for a given hashtag and display the results in a tableview with custom cell view.
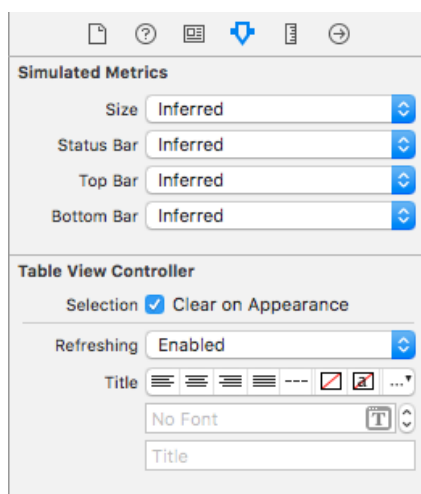
Step 1:
A table view is created with dynamic cells which have image view and 3 labels to display time, title and message respectively. The table view is embed in a navigation controller.
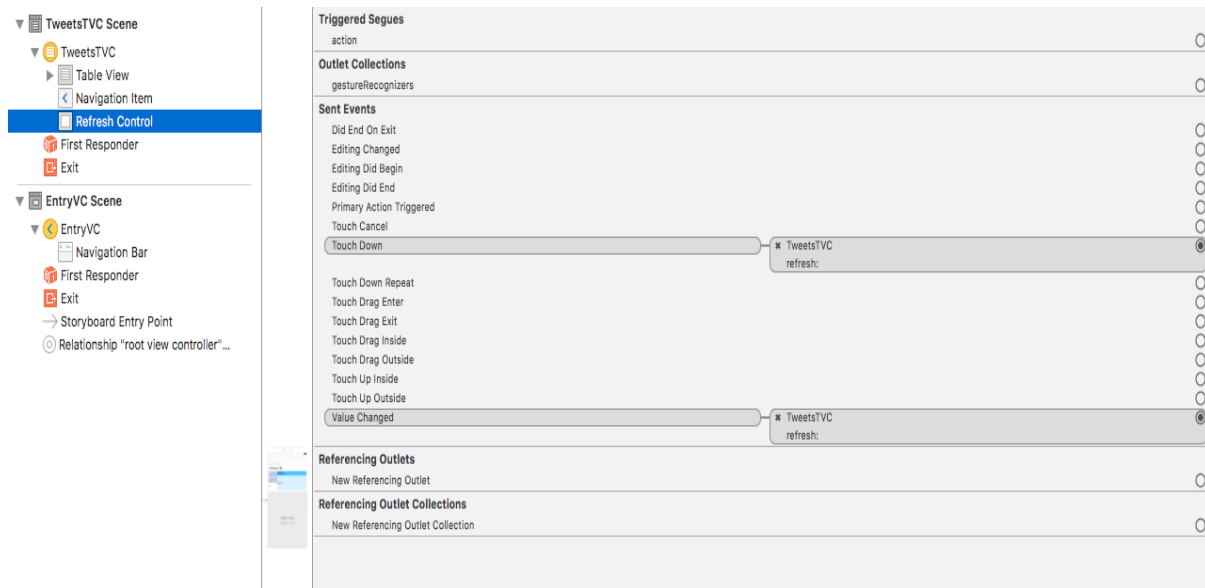
Step 2:
Add refresh action method where we fetch tweets. To add refresh action method, select the TweetsTVC view controller -> Enable Refreshing attributed in attributes inspector.
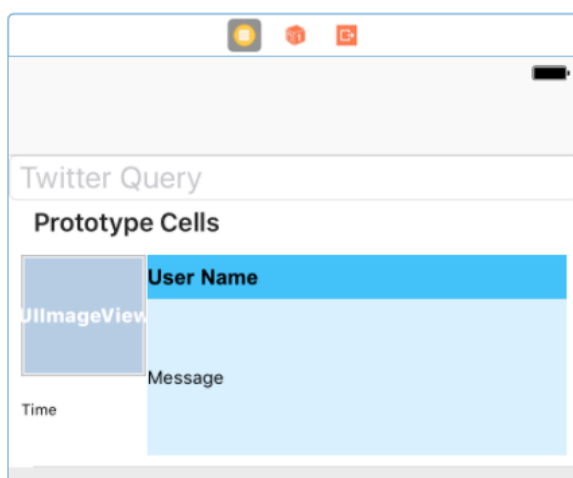
To connect the action method of refresh() to TweetsTVC class, select the Refresh control on document outline window, hold the control key and drag to TweetsTVC class.

Step 3:

Create IBOutlet for imageview and three labels on Table View to CustomCell.

Step 4:

Core Functions and its functionalities.

- **"RequestingTweets( ) function"** checks whether the request for the tweets is for new tag or existing one and return the tweets in regard to condition.

  Case 1:
  If the request is to search for new entered new tag, convenience initializer method of class *"TwitterRequest: NSObject"* in Twitter API is called.

```
//calls the convenience initializer method of class "TwitterRequest: NSObject" in Twitter API
return TwitterRequest(search: twitterQueryText!, count: 5)
```

Case 2:

If the request is to fetch new tweets of existing tag (means that the user refreshes the table). The "***RequestForNewer***" is property is called in twitter API via

TweetsTVC.SWIFT file:

```
//generating request to get new tweets of previous tag
return OldTweets!.requestForNewer!
```

TwitterAPI.SWIFT file:

```
// Create request for newer tweets
open var requestForNewer: TwitterRequest? {
    if max_id == nil {
        if parameters[TwitterAPI.Request.sinceID] != nil {
            return self
        }
    } else {
        return modifiedRequest(parametersToChange: [TwitterAPI.Request.sinceID : max_id!], clearCount: true)
    }
    return nil
}
```

- ***Request( ) function*** request the tweets and add the tweets to the in table.

TweetsTVC.SWIFT file:

```
@IBAction func refresh(_ sender: UIRefreshControl?) {

    let tweetsRequestHandler = requestingTweets()

    //fetchTweets returns tweets in "incomming Tweets"
    tweetsRequestHandler.fetchTweets{(incommingTweets) -> Void in
        if incommingTweets.count > 0
        {
            self.OldTweets = tweetsRequestHandler
            self.tweets.insert(incommingTweets, at: 0)
            self.tableView.reloadData()
        }

        //endRefreshing methond should call after begin refreshing to return control to its default
            state
        self.refreshControl?.endRefreshing()
    }
}
```

Step 5:

Whenever the user enter the tag to search the *"**textFieldShouldReturn function**"* is called to set the search tag to value of text box

TweetsTVC.SWIFT file:

```swift
func textFieldShouldReturn(_ textField: UITextField) -> Bool {

    //relinquish delegate status as first responder in its window
    textField.resignFirstResponder()

    //textField contains name of the new tag
    twitterQueryText=textField.text
    return true
}
```

and the control goes to "*didSet*" of twitterquery which call the "*refreshing()*" function(refresh the table) followed by "*refresh()*" and *"requestingTweets( )*". The control finally goes back to "*refresh()*" function to load new tweets in table.

TweetsTVC.SWIFT file:

```swift
var twitterQueryText: String? = "#ucd"
    {

    didSet {

        OldTweets = nil
        //displaying the new tag entered by user
        twitterQueryTextField?.text = twitterQueryText
        //reloading the tableview
        tableView.reloadData()

        refreshing()
    }
}


func requestingTweets() -> TwitterRequest
{

    //checking whether there is tweet request made for new tags or existing tags
    guard (OldTweets != nil) else {

        //calls the convenience initializer method of class "TwitterRequest: NSObject" in Twitter API
        return TwitterRequest(search: twitterQueryText!, count: 5)
    }

    //generating request to get new tweets of previous tag
    return OldTweets!.requestForNewer!
}

func refreshing()
{
    //var requestingTweets: TwitterRequest?
    refreshControl?.backgroundColor = UIColor.blue
    refreshControl?.tintColor = UIColor.white
    //refresh the table
    refreshControl?.beginRefreshing()
    refresh(refreshControl)
}

  @IBAction func refresh(_ sender: UIRefreshControl?) {

    let tweetsRequestHandler = requestingTweets()

    //fetchTweets returns tweets in "incomming Tweets"
    tweetsRequestHandler.fetchTweets{(incommingTweets) -> Void in
        if incommingTweets.count > 0
        {
            self.OldTweets = tweetsRequestHandler
            self.tweets.insert(incommingTweets, at: 0)
            self.tableView.reloadData()
        }

        //endRefreshing methond should call after begin refreshing to return control to its default
            state
        self.refreshControl?.endRefreshing()
    }
}
```

Step 6:

Whenever the user refresh the table to fetch new tweets of existing tags. The *refresh() function* is called followed by *requestingTweets().*

Step 7:

Date of tweets, User ID, Tweets and Image of people is retrieved from **TwitterAPI file.SWIFT** and these values are assigned to corresponding Objects in Custom Cell.

CustomCELL.SWIFT file:

```swift
func refreshCell()
{
    //date formatting
    let formatter = DateFormatter()
    formatter.dateFormat = "MMM d, h:mm a"
    var d:Date
    d=twitter!.created
    let dd=formatter.string(from: d)
    tweet_day_time?.text="\(dd)"

    user_screenname?.text="\(twitter!.user)"
    tweet_content?.text = twitter!.text

    //get the user image from url "profileImageURL" and set it inside UIImageView "user_image"
    let profile_image=twitter?.user.profileImageURL
    if let image = NSData (contentsOf: profile_image!)
    {
        user_image?.image=UIImage(data: image as Data)
    }
}
```
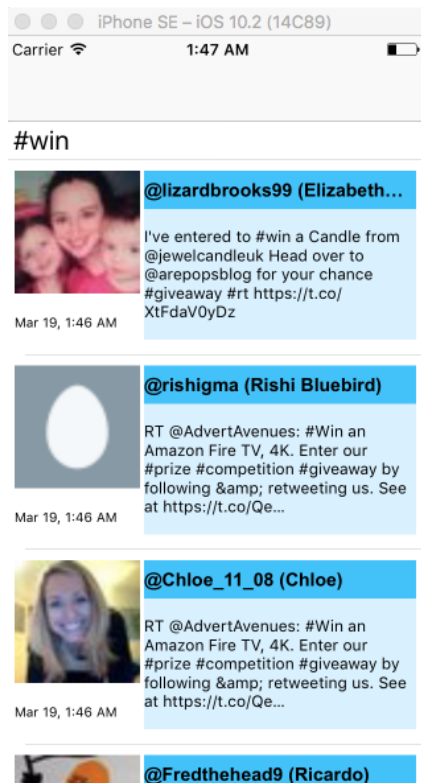
SCREEN SHOTS:

Initial screen:
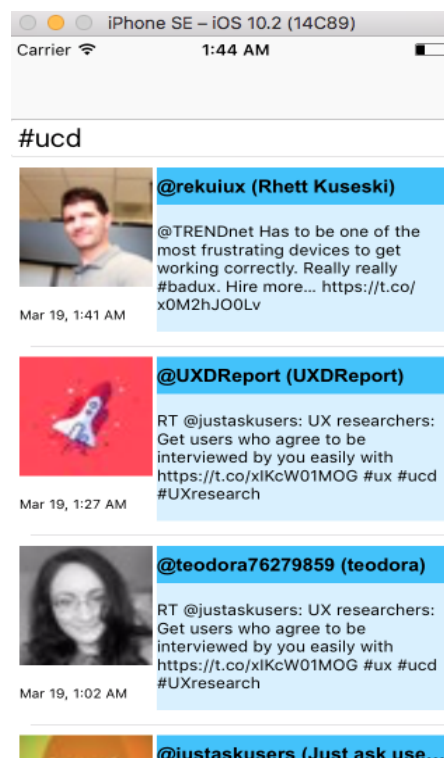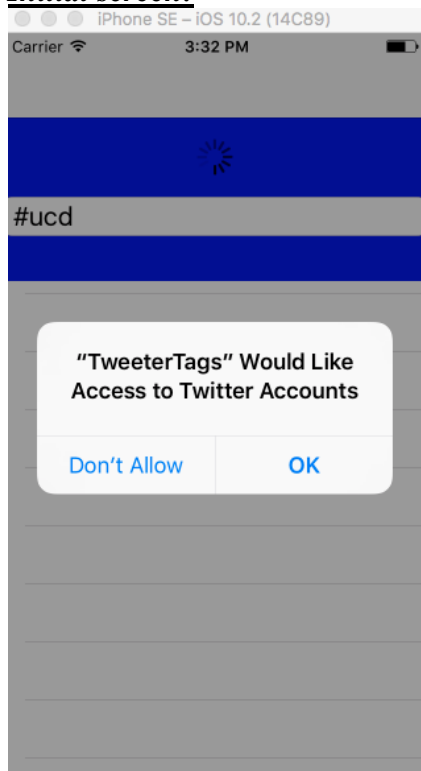
New HashTag Screen:

(note: response time to fetch new tweets might take time)

Screen after refreshing the table: