# Report for part 1

Name: Xiaosheng Liang

Student Number: 15211913

- Please click TweeterSpace in the folder to run the program and run TweeterTags in iPhone Device. (if Twitter.framework in TweeterTags is red, please run Twitter as Generic iOS Device before running TwitterTags.)

- Structure of program:

  There are two parts in the workspace of this program:

  - Twitter: TwitterAPI (including TwitterTweet, TwitterRequest, TwitterUser, TwitterMedia). It runs as a framework which is embedded into TwitterTages.
  - TwitterTags: MVC structure of Twitter tag

Solution of Part 1:

- There are two var which are needed. One is tweets which is an array, the other is searchText.

```
14    var tweets = [Array<Tweet>]() {
15        didSet{
16            tableView.reloadData()
17        }
18    }
19
20    var searchText: String?{
21        didSet{
22            tweets.removeAll()
23            searchForTweets()
24            title = searchText
25        }
26    }
27
```

- Once searchText is decided, it would call searchText var, then methods of removeAll(), searchForTweet() are invoked and find tweets. So the reloadData() of tweets is called.
- Then all the methoda following would be called. Send the request, search the text which was determined here call "#ucd" and fetch tweets.

```
    private var twitterRequest: TwitterRequest?{
        if let query = searchText, !query.isEmpty{
            return TwitterRequest(search: query + " -filter:retweets", count:
                100)
        }
        return nil
    }

    private var lastTwitterRequest: TwitterRequest?

    private func searchForTweets(){
        if let request = twitterRequest{
            lastTwitterRequest = request
                request.fetchTweets { [weak weakSelf = self] newTweets in
                    DispatchQueue.main.async{
                        if request == weakSelf?.lastTwitterRequest{
                            if !newTweets.isEmpty{
                                weakSelf?.tweets.insert(newTweets, at: 0)
                            }
                        }
                    }
                }
            }
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.estimatedRowHeight = tableView.rowHeight
        tableView.rowHeight = UITableViewAutomaticDimension
        searchText = "#ucd"
    }
```

- The code below is the heart of table view data source. tableView.reloadData() would call all these methods. Load the data and create cell for tweets.

```
    // MARK: - Table view data source

    override func numberOfSections(in tableView: UITableView) -> Int {
        // #warning Incomplete implementation, return the number of sections
        return tweets.count
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection
        section: Int) -> Int {
        // #warning Incomplete implementation, return the number of rows
        return tweets[section].count
    }

    private struct Storyboard{
        static let TweetCellIdentifier = "Tweet"
    }


    override func tableView(_ tableView: UITableView, cellForRowAt indexPath:
        IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: Storyboard.
            TweetCellIdentifier, for: indexPath)
        let tweet = tweets[indexPath.section][indexPath.row]
        if let tweetCell = cell as? TweetTVCell{
            tweetCell.tweet = tweet
        }
        return cell
    }
```

- Create ImageView and UILabels for each cell. Once cells are loaded, all

tweets would be updated. So updateUI() is invoked.

```
    @IBOutlet weak var tweetCreatedLabel: UILabel!
    @IBOutlet weak var tweetProfileImageView: UIImageView!
    @IBOutlet weak var tweetScreenNameLabel: UILabel!
    @IBOutlet weak var tweetTextLabel: UILabel!

    var tweet:  Twitter.Tweet?{
        didSet{
            updateUI()
        }
    }
```

- In updateUI(), it would clear all cells and reset all exist tweet information. Then set one by one and load new information based on tweets.
- The let profileImageView is for image which tweet has.

```
    private func updateUI(){
        // reset any existing tweet information
        tweetTextLabel?.attributedText = nil
        tweetScreenNameLabel?.text = nil
        tweetProfileImageView?.image = nil
        tweetCreatedLabel?.text = nil

        // load new information from our tweet (if any)
        if let tweet = self.tweet
        {
            tweetTextLabel?.text = tweet.text
            if tweetTextLabel?.text != nil  {
                for _ in tweet.media {
                    tweetTextLabel.text! += " 📷"
                }
            }

            tweetScreenNameLabel?.text = "\(tweet.user)" // tweet.user.description

            if let profileImageURL = tweet.user.profileImageURL {
                if let imageData = NSData(contentsOf: profileImageURL as URL) {
                    tweetProfileImageView?.image = UIImage(data: imageData as Data)
                }
            }

            let formatter = DateFormatter()
            if NSDate().timeIntervalSince(tweet.created as Date) > 24*60*60 {
                formatter.dateStyle = DateFormatter.Style.short
            } else {
                formatter.timeStyle = DateFormatter.Style.short
            }
            tweetCreatedLabel?.text = formatter.string(from: tweet.created as Date)
        }

    }
}
```

- Set searchTextField as a delegate. Then get the keyword in searchTextField and grab all tweets which is included the keyword.

```swift
@IBOutlet weak var searchTextField: UITextField!{
    didSet{
        searchTextField.delegate = self
        searchTextField.text = searchText
    }
}

func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    textField.resignFirstResponder()
    searchText = textField.text
    return true
}
```