Project 4 part II
Team: Swarnabha Roy

In this project, I implemented a Twitter-like system in F# using the WebSocketSharper and Suave web framework to implement a WebSocket interface. I have designed a JSON based API that represents all messages and their replies, along with errors.

Suave was used in the Server side code, and WebSocketSharper was used in the Client side code to create web sockets and ensure communication between the twitter engine and client simulator..

The Twitter Engine has the following functionalities, each of which is handled by a different actor:

Register account - takes a string as an argument for username and adds a new user to a map. Duplicate entries are not allowed ie. no two users can have the same username

Send tweet - takes two strings as arguments for the username of the user sending the tweet and the tweet content. Only existing users can send tweets.

Subscribe to user's tweets - takes two strings as arguments for the username of the user wanting to subscribe, and the username of the user he/she wants to subscribe. By default, all users are subscribed to their own accounts, ie. if a user has not subscribed to any other user, he/she will be able to view only his/her own tweets.

Re-tweets - similar to Send tweet. If a user wants to share a tweet with all of his/her subscribers, he/she can just retweet it.

Query tweets subscribed to - takes a string as an argument for the username of the user. This displays all the tweets a user has subscribed to, ie. his/her own tweets plus tweets from people he/she has subscribed to.

Query tweets with specific hashtags - takes a string as an argument for the hashtag which a user wants to query. This displays all the tweets containing the specified hashtag.

Query tweets in which the user is mentioned - takes a string as an argument for the username of the user who wants to see his/her mentions. This displays all the tweets in which the specified user has been mentioned.

connect/ disconnect - takes a string for the username of the user wanting to connect or disconnect. This is similar to going online/offline. When a user is connected, if he/she does any activity ie. tweet/retweet, all of his/her tweets and retweets are sent as a live feed to all of his/her subscribers. That is, the tweets/ retweets are available without querying. This stops and returns to the previous scenario as soon as the user disconnect (goes offline).

How to run:

Server:

```
dotnet run
```

Client:

```
dotnet fsi TwitterClient.fsx
```

Command options:

Register: `register,<username>,<password>`
Login: `login,<username>,<password>`
Subscribe: `subscribe,<username>`
Send Tweet: `send,<tweet content>`
Retweet: `retweet,<tweet ID>`
Query the subscribed tweets: `queryST`
Query Hashtags: `queryHashtag,<#hashtag>`
Query Mentions: `queryMention,<@mention>`