DOSP Project 4 (Part 1)
Twitter Clone - Report

Team members:

Tushar Ranjan – 45562694
Sankalp Pandey – 92878142

Files:

Twitter_Server:
    Server.fs
    Twitter_Server.fsproj
Twitter_Client:
    Client.fs
    Twitter_Client.fsproj

Brief description:

The aim of the project is to clone twitter and its functionalities and test them using a simulator. We have covered almost all the basic features of twitter like secured login, tweet, retweet, follow etc.

Working:

All the actors in our Twitter Engine and Simulator communicate via JSON messages. It's a efficient and easy technique to communicate between different processes in a straightforward way. Because JSON messages are sting communications, all we have to do now is tell the actors to anticipate string messages.

Server: we have a boss that receives all of the client requests. We also employ many actors to help the boss server with time-consuming query requests, such as queries of all previous tweets, in order to improve speed. When the master server receives a query request, it assigns it to one of the workers at random, and the client just waits for the worker's response.

```
 ******** Status of Server ********
Requests processed: 5
Tweets in DataBase: 0
Registered Users: 1
Online Users: 0
**********************************
```

Client: if a client is waiting for a response to a previous request, the client will disregard any subsequent instructions from the simulator and wait until the response arrives. List of features have been provided in the read me file.

```
WELCOME TO TWITTER!
~~~~~~~~~~~~~~~~~~~~~~
Please choose from the list of following options:
Press 1 to login
Press 2 to sign up
Press 3 to know more
```

```
You are logged in

Please choose from the list of following options:
Press 1 to Tweet
Press 2 to Retweet
Press 3 to follow an user
Press 4 to log out
Press 5 to see your tweets
Press 6 to view tweets by hashtags
Press 7 to view tweets by metions
```

Simulate: The number of clients will be entered by the user, apart from this we have set few parameters for simulation mode, e.g.:
Average online – 50%
Average Tweets – 50%
Average Retweets – 20%
Average Tags – 10%
Average Mentions – 10%

```
~Simulation~

Enter the no. of users you would like to have in your testing
(Number): 3


...please wait...

[[[User1] Successfully registeredUser3] Successfully registeredUser2] Success
fully registered


[[[User3] User3 successfully connected to server
User2] User2 successfully connected to server
User1] User1 successfully connected to server
[User2[User1] [User3] No new tweets at the moment]
No new tweets at the moment
No new tweets at the moment
[User1] Subscirbe done![
User2] Subscirbe done!
[User1] Subscirbe done!
```

```
 ******** Status of Server ********
Requests processed: 12
Tweets in DataBase: 0
Registered Users: 3
Online Users: 3
**********************************


 ******** Status of Server ********
Requests processed: 12
Tweets in DataBase: 0
Registered Users: 3
Online Users: 3
**********************************


 ******** Status of Server ********
Requests processed: 12
Tweets in DataBase: 0
Registered Users: 3
Online Users: 3
**********************************
```

Observations:

1. In simulation mode, we attempt a different number of clients to see how they operate. The server receives a large number of requests so we use the information from the server status to determine the average number of requests per second:

No of Clients	Avg request per second
500	847
1000	913
2500	1262
5000	1879

2. In client mode we can observe that when the server's traffic grows, the response time for each request rises. It's worth noting that when there's less data to send in response to query, the response time may be shorter than anticipated. In addition, we discovered that the response time (*in seconds*) for each request is shorter at the beginning of the stage of simulations than the records below.

No. of clients	Tweets	Retweets	Follow	Query history
1000	0.11	0.05	0.09	12.47
2500	9.52	5.56	13.22	8.36
5000	7.41	15.22	16.01	16.58

3. For Zipf distribution, each customer is ranked from 1 to N, with N denoting the total number of clients that the simulator has simulated. Each client will send 1/rank requests to the server every millisecond. Each client's rank is inversely proportionate to the number of followers they have.