DOS Project 4 Part I Report

Team Members:
Suprith Reddy Gurudu (UF ID: 9961-2134)
Hima Tejaswi Gummadi (UF ID: 2455-9492)

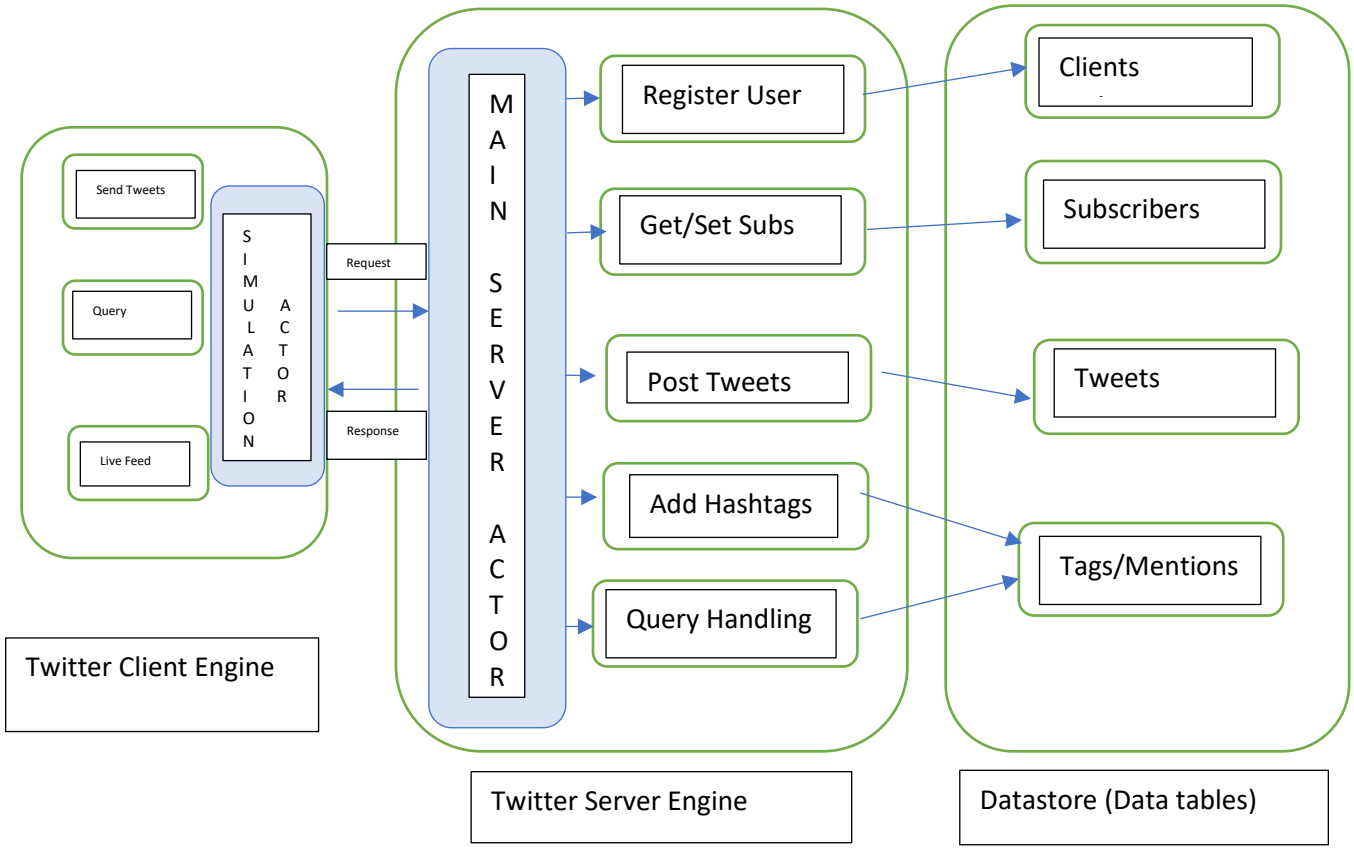
Brief Description:

The goal of this project is to implement a Twitter Clone and a client tester/simulator. The problem statement is to implement an engine that (in part II) will be paired up with Web Sockets to provide full functionality.

The Twitter engine/Server has been implementation details:

- Register account
- Send tweet. Tweets can have hashtags (e.g. #COP5615isgreat) and mentions (@bestuser)
- Subscribe to user's tweets
- Re-tweets (so that your subscribers get an interesting tweet you got by other means)
- Allow querying tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned (my mentions)
- If the user is connected, deliver the above types of tweets live (without querying)

Architecture:



Implementation Details:

Server.fsx: This file contains Twitter engine implementation code for processing and distributing tweets. All the information related to subscribers, tweets, queries, etc. are handled by the engine communicating with the server. The main server actor is used to communicate directly with the clients to distribute the tweets to subscribers and returns the query results. The data model uses all data tables in a dataset and server.fsx uses utility.fs for all the utility functions.

Client.fsx: This file has information related to a single client participant for a single twitter user. This includes the information of its simulation and server requests. The twitter client engine interacts with the main server actor of the twitter server engine for sending requests and getting responses.

Utility.fs: This file includes functions to get the information about username, user id, followers tweets, mentions along with the functionality of adding users, tweets, hashtags mentions and user tweet mapping.

Datamodel.fs: This file includes structure of data tables along column headers specifying the relation of the entities.

Project4.fsproj: This file is a project dependency file which is written in order of dependencies. The Utility.fs is dependent on the Datamodel.fs file and thus it needs to be written first in order before the utility file.

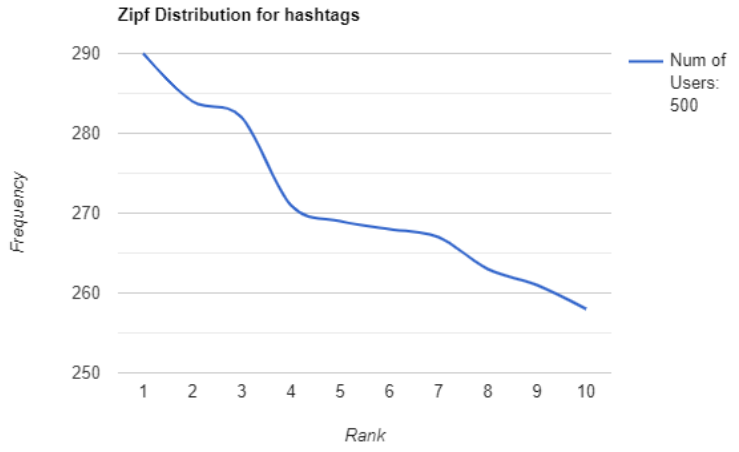
Note: Please add this two lines to your .fsproj file in your folder in the same order same as Project4.fsproj

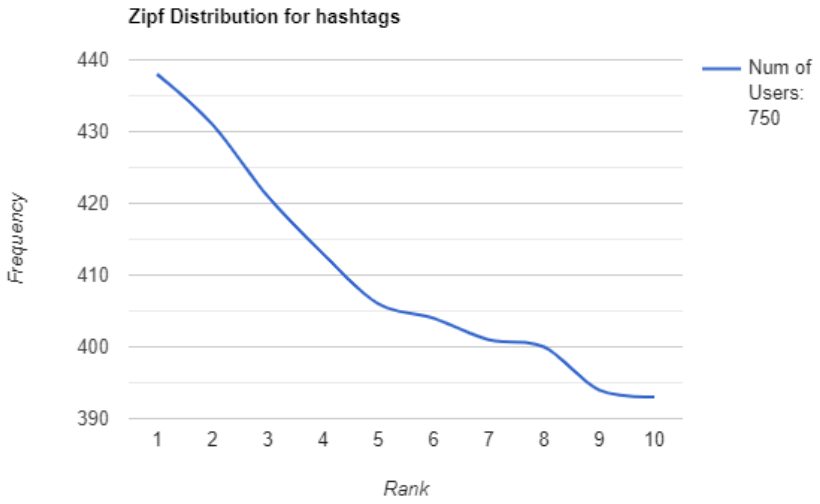
```
<Compile Include="Datamodel.fs" />
<Compile Include="Utility.fs" />
```

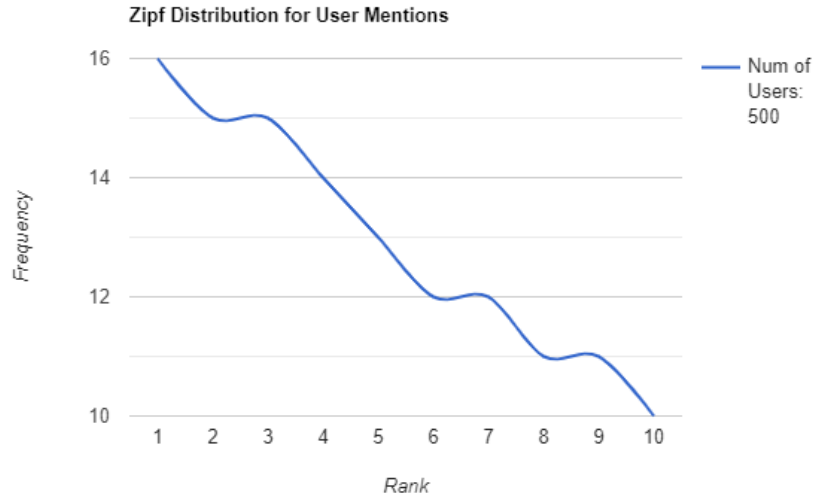
datastore.xml (gets created when we run the code): This file contains the data mapping and stores data the in form of tags.

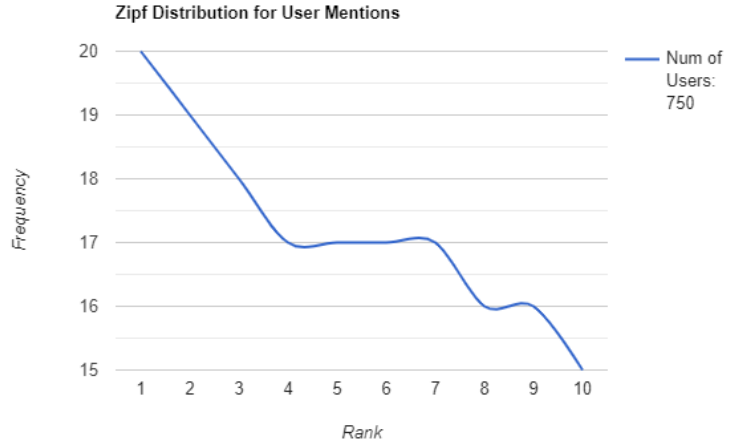
Zipf distribution - Zipf distribution for this project is created based on the hashtags and mentions. The distribution is studied between the rank order and frequency of occurrence of the hashtags and mentions where the frequency of a particular observation is inversely proportional to its rank.

Below is the distribution curve plotted for number of clients = 500 and 750,




Zipf Distribution for hashtags
440
430
420
410
400
390
Frequency
1
2
3
4
5
6
7
8
9
10
Rank
Num of Users: 750


Zipf Distribution for User Mentions
16
14
12
10
Frequency
1
2
3
4
5
6
7
8
9
10
Rank
Num of Users: 500


Zipf Distribution for User Mentions
20
19
18
17
16
15
Frequency
1
2
3
4
5
6
7
8
9
10
Rank
Num of Users: 750

Running the code:

- Open two terminals – one for server code and other for client simulation and follow to the code directory
- Go to the project folder in both terminals using command line.
- Run the following server code in one terminal to start the twitter server engine
  > dotnet fsi --langversion:preview server.fsx
- Then run the client code with the following command in another terminal to start the client simulator
  > dotnet fsi --langversion:preview client.fsx <serverIP> <serverPort> <SimulatorNo> <numOfUsers>
  Note: here serverIP is localhost and serverPort is 1800
- To simulate the system with different parameter values, start again from the client simulation step.

Performance Results:

Performance analysis with Number of Clients:
Max-subscribers = clients * 10

Number of Clients	Time to Tweet, Retweet (milli seconds)	Query Tweets by Hashtag (milli seconds)	Query Tweets by Mention (milli seconds)	Query All Tweets (milliseconds)	Complete Execution (milli seconds)	Total No. of req served by server	Avg no. of req per sec by server
100	1128	167	108	326	1402	1403	1403
200	11725	584	525	2056	12046	4206	350
300	18945	1005	946	2689	19263	5203	337
500	28353	2649	2590	8577	28724	7002	250
750	64612	5762	5703	17932	65141	10501	161
1000	145291	14273	14214	41529	145996	14003	96