COP5615 – Fall 2019
PROJECT - 2

1. Group Members

Name	UFID
Gurpreet Nagpal	0698-9051
Floura Angel Nadar	2303-6958

Steps to run:

1. Unzip the zip file
2. From terminal go to folder twitter
3. Run command “mix run proj4.exs numofusers noofRequests” or “mix test”

Ex. mix run proj4.exs 100 5

Following are implemented:

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

Implementation Details:

1. Register account and delete account: When a client id is created, then the server is informed and it stores its user is in the client table. The user logged in also updates the active Users table, while disconnecting log the user out from the active user but still is present in the client table. Deleting an account removes the user from both the tables.
2. Send tweet: Each user generates the mentioned number of tweets, and then picks a random user id from the client table and mentions it in one of it’s tweets. Similarly for hashtag we used #COP5615isgreat, every user tweets it at least once.
3. Subscribe to user's tweets: The user can subscribe to another user, this updates it own table with following attributes and increases the table of followers for the one it subscribes to.
4. Re-tweets: The retweets is done for a user picking any random tweet from a random user and then this tweet is reflected in its followers table.
5. Allow querying tweets subscribed to: Incase the user is mentioned in someone’s tweet or if it requests for a specific hashtag then that tweet containing that is retrieved and shown in live feeds.
6. If the user is connected, deliver the above types of tweets live: dynamic updation of tweet is done.

Test cases:

1. Registration: We check if the user is registered, that is if it is in the table of client server.

Unit test: The user id is correctly updated in the system, as the new id is a member of the client table.
Functional test: Since registration test succeed, we were able to test it login and hence updation in the active user table.

2. Deleting an account:
   Unit test: While deleting an account, the user id is not a member of the client table anymore
   Functional test: Since deleting test succeed, we were able to test that the user is not able to tweet anymore and is removed from active users as well

3. Send tweet
   Unit test: While sending a tweet, all the followers of the user id are able to view it in their live feed.
   Functional test: The tweetData is updated on the server is updated.

4. Subscription:
   Unit test: While subscribing to a hashtag or a tweet, the user is able to view the details in a live feed. Or while subscribing or following another user, the data from that user id is accessible to this one.
   Functional test: It updates both the table of the users with respective to following and followers table.

5. Retweet:
   Unit test: While retweeting the tweet is picked from the following list
   Functional test: All the followers of the process receive the tweet and it is also updated in the tweet data list of the sender.

6. Query Tweet:
   Unit test: While querying the requests, the tweets for hashtag irrespective of any particular user is retrieved, whereas for mentions it just tests that the user is specifically mentioned.

7. Disconnection:
   Unit test: While disconnecting an account, the process is removed from the active users but still is present in the client table
   Functional test: Since disconnection test succeed, we were able to test that the user is not able to tweet anymore.

Performance Analysis:

Performance metrics is measured on basis of the client formation, tweeting , retweeting, add subscriptions and processing requests.

No of nodes	No of requests	Time taken(sec)
10	5	2.661
100	5	11.407
500	5	52.38
1000	5	57.714
10	10	2.788

100	10	11.564
10	30	2.998
100	30	11.555

The simulator time is proportional to the number of Nodes and Number of requests. The distribution of nodes was random, hence had to handle cases where the sequence of events were not in an order