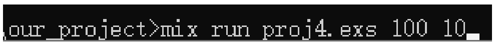
COP 5615 Project 4 Report

Zhengyu Li, 29453969
Yue Yu, 96141901

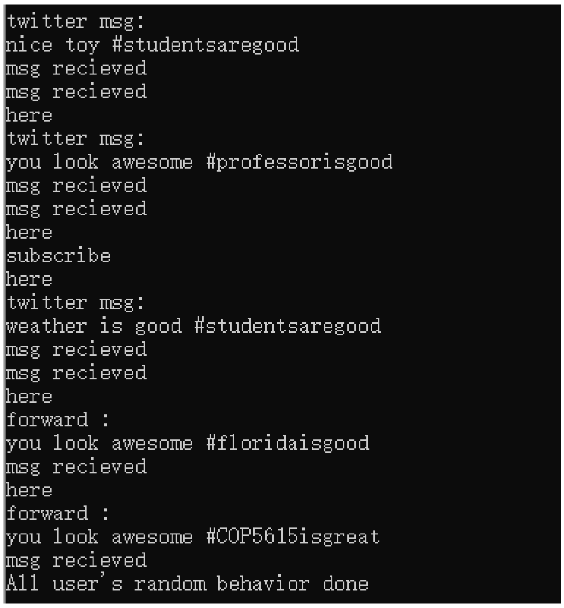
Performance analysis:

Our project is a Twitter-like engine. The functionality of our project includes register account and delete the account, send tweets with hashtags and mentions, subscribe other users, re-tweets, and querying tweets with specific hashtags and messages in the mentioned inbox.

The maximum number of users we tested is 100 users, and each user does 10 random operations such as send tweets with hashtags and mentions, re-tweets. The project can handle larger users and operations of each users, but takes more time.



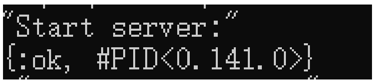
```
our_project>mix run proj4.exs 100 10
```



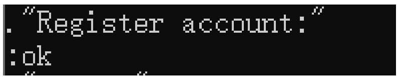
```
twitter msg:
nice toy #studentsaregood
msg recieved
msg recieved
here
twitter msg:
you look awesome #professorisgood
msg recieved
msg recieved
here
subscribe
here
twitter msg:
weather is good #studentsaregood
msg recieved
msg recieved
here
forward :
you look awesome #floridaisgood
msg recieved
here
forward :
you look awesome #COP5615isgreat
msg recieved
All user’s random behavior done
```

We used a single engine process as the engine to distribute tweets to inboxes, and each process stands for a user. We achieved these by using GenServer actor model in Elixir. In order to improve the performance, we distributed most of jobs into user’s processes so that the pressure of the engine is not that high, since we only have one engine process. Thus, we save different types of messages and querying the tweets in several inboxes like mention inbox, hashtag message inbox, personal inbox etc. in each user’s process. The live message will be sent to follower’s personal inbox. Messages with hashtags will be sent to the follower and hashtag boxes, and the same to mentions. Users can also query subscribed users’ tweets by search the subscribed user’s send box. We predefined several hashtags and twitter messages so that these users can send random messages with random hashtags and mention the existed users when they send Twitters. This twitter liked engine can handle whatever numbers of hashtags and mentions in a tweet, and the hashtags and mentions can be written at any place in the message. We printed each operation of each user in their own processes so that we can know what are their random behaviors. We tested each functionality using ExUnit test in Elixir in the test folder.

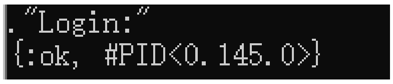
Performance Screenshots:



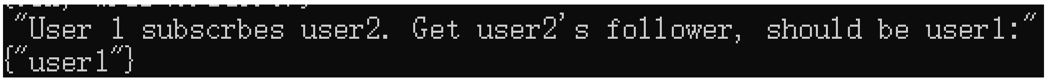
```
"Start server:"
{:ok, #PID<0.141.0>}
```



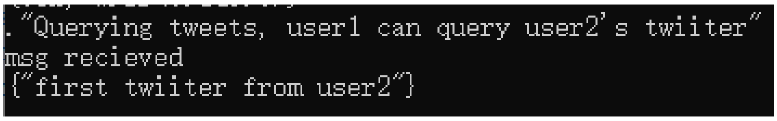
```
."Register account:"
:ok
```



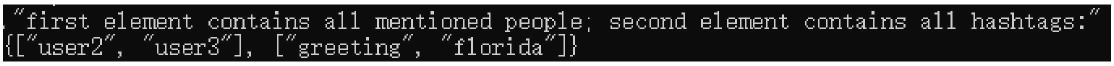
```
."Login:"
{:ok, #PID<0.145.0>}
```



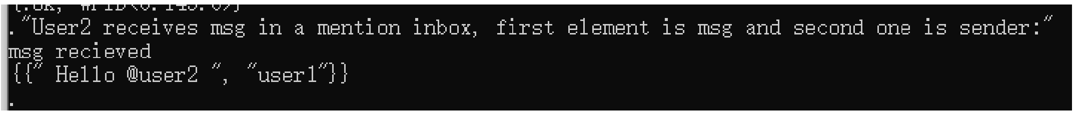
```
 "User 1 subscrbes user2. Get user2's follower, should be user1:"
{"user1"}
```



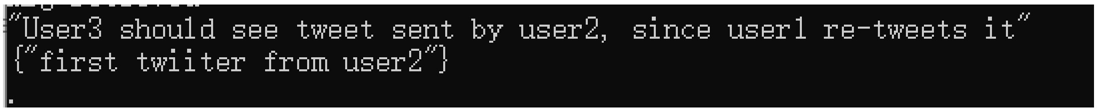
```
."Querying tweets, user1 can query user2's twiiter"
msg recieved
{"first twiiter from user2"}
```



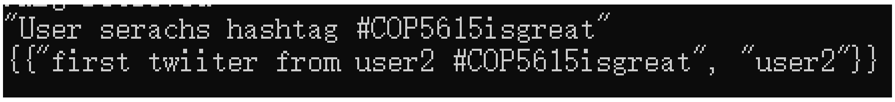
```
."first element contains all mentioned people; second element contains all hashtags:"
{["user2", "user3"], ["greeting", "florida"]}
```



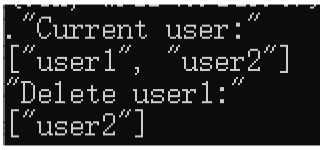
```
."User2 receives msg in a mention inbox, first element is msg and second one is sender:"
msg recieved
{{" Hello @user2 ", "user1"}}
.
```



```
"User3 should see tweet sent by user2, since user1 re-tweets it"
{"first twiiter from user2"}
.
```



```
"User serachs hashtag #COP5615isgreat"
{{"first twiiter from user2 #COP5615isgreat", "user2"}}
```



```
."Current user:"
["user1", "user2"]
"Delete user1:"
["user2"]
```