**Project 4 Part II**

**Jagan Mohan 61619287**

**Sarah Samji 39372723**

The library **phoenix\_gen\_socket\_client** implements an Elixir client for Phoenix Channels protocol 2.x.

Twitterweb is the phoenix engine using websockets.

Project4 is the simulator simulating client behavior.

**HOW TO RUN**

**-project4**

mix escript.build

escript project <numclients>

**-twitterweb**

mix compile

mix phx.server

-Change the value of numClients in endpoint.ex according to the requirement for testing. This value has been set to 1000 for now. But as long as this value is greater than the <numclients> parameter value to project4 the implementation will work correctly.

-If upon running you receive error of not having installed required dependencies then run:

mix deps.get

-If you want to run on separate machines then change the URL in start\_link() of socket\_client.ex of project4.

**DESCRIPTION**

* Register account
* Send tweet. Tweets can have hashtags and mentions
* 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)
* Simulate a Zipf distribution on the number of subscribers. For accounts with a lot of subscribers, increase the number of tweets. Make some of these messages re-tweets

**MODULES IN THE PROJECT**

**project4**

socket\_client: Creates the socket connections for the required number of clients. Each client upon registration is associated with a topic. The client will then subscribe to the the topics of its followers, query. The API calls to the server to handle all the functionalities are initiated here.

simulator: Starts the required number of clients.

**Twitterweb**

This is the engine of our twitter clone.

endpoint.ex: Generates the Zipf distribution based on the number of followers each client has.

Channels:

These channels implement the callbacks for the various functionalities of this project.

-Tweet:\* to handle all tweet related topics (TweetChannel)

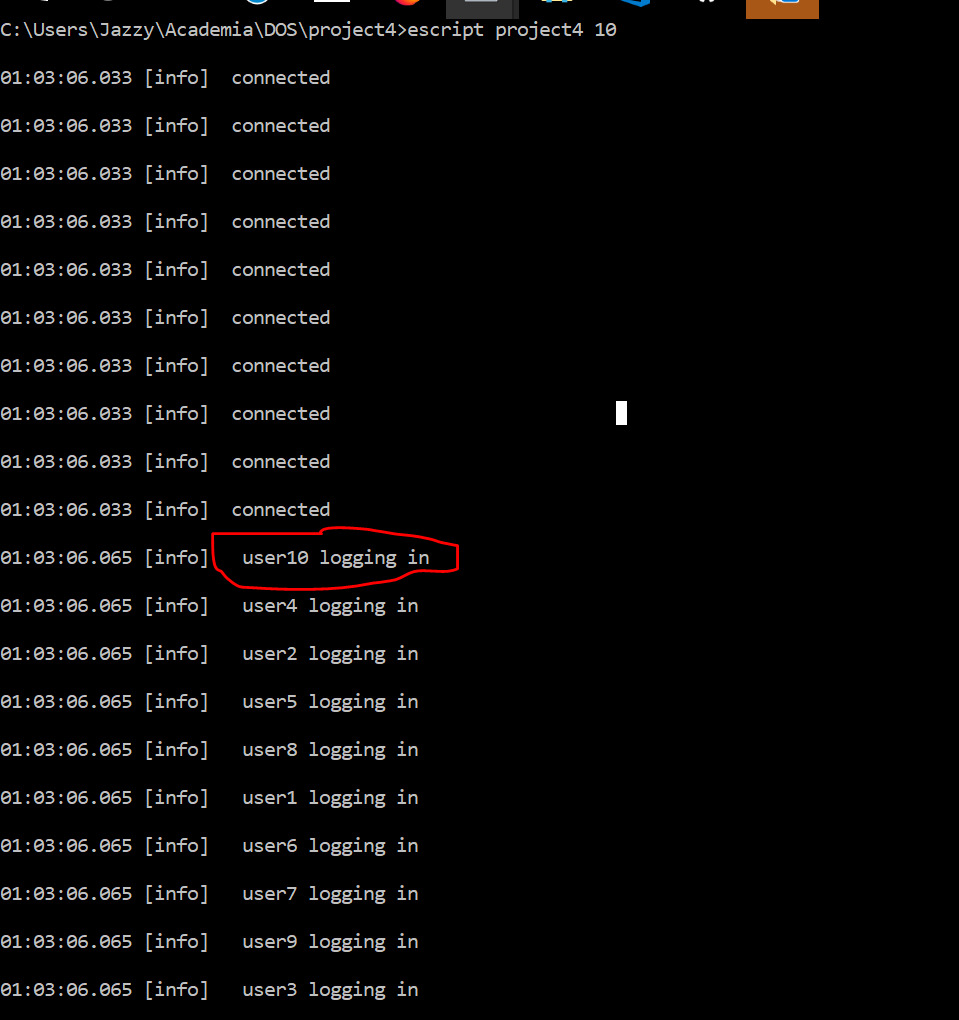
-Query:\* to handle all query related topics (QueryChannel)

Datastore: Holds the ETS tables. Basically, acts as the database on the server.

**RESULTS**

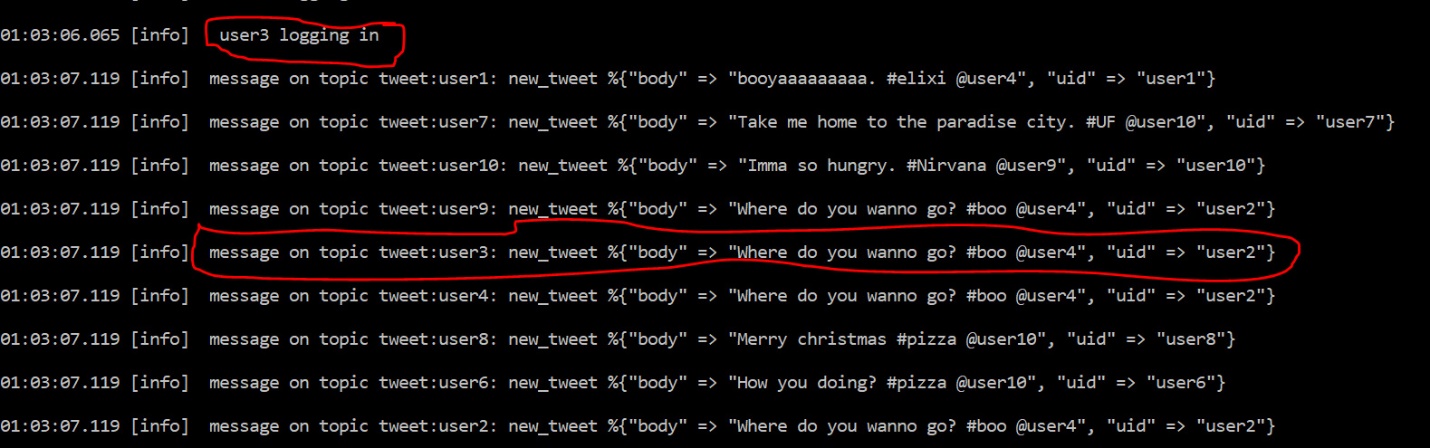
These are the results of our implemented functionalities.

**1.Logging Functionality**

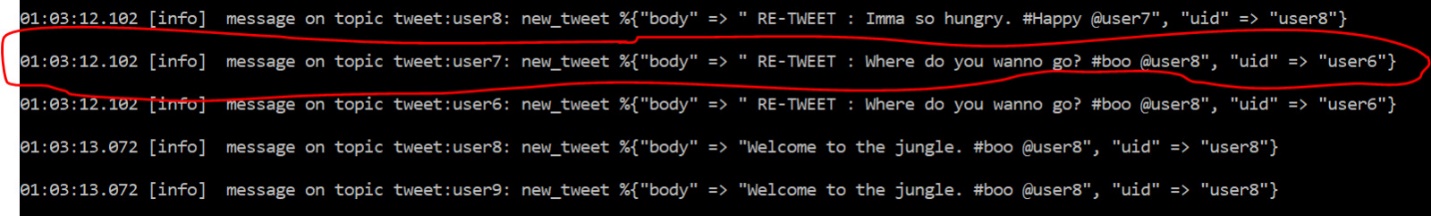


2. **Tweeting**

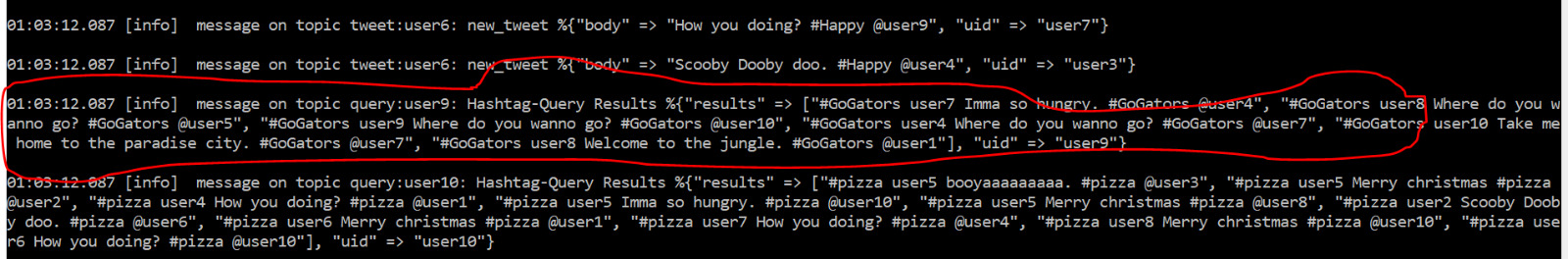
Messages being broadcasted to subscribers



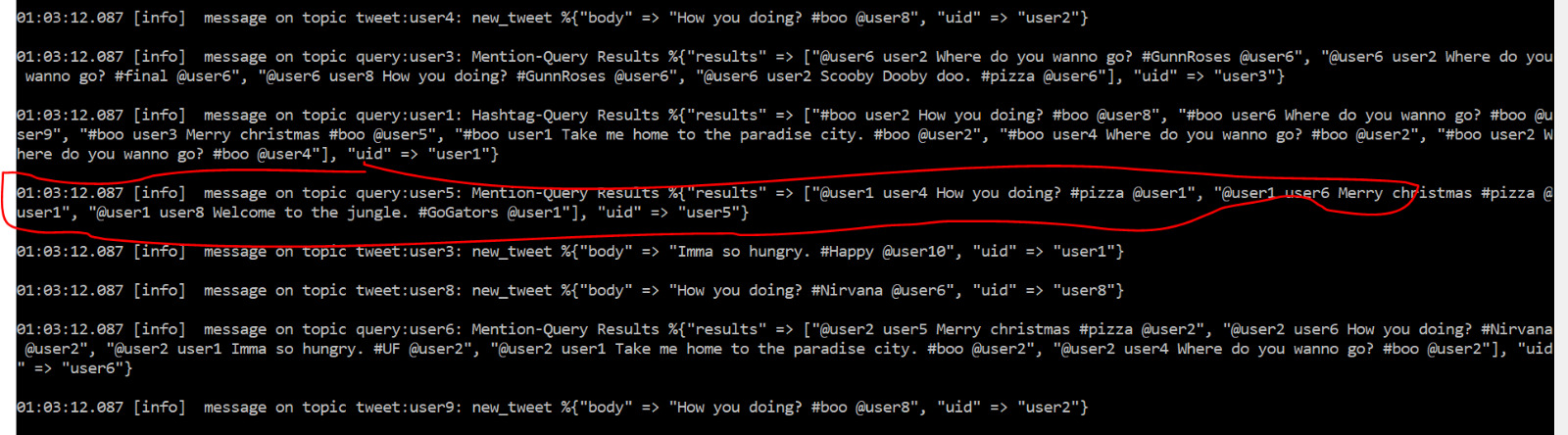
**3. Re-Tweeting**



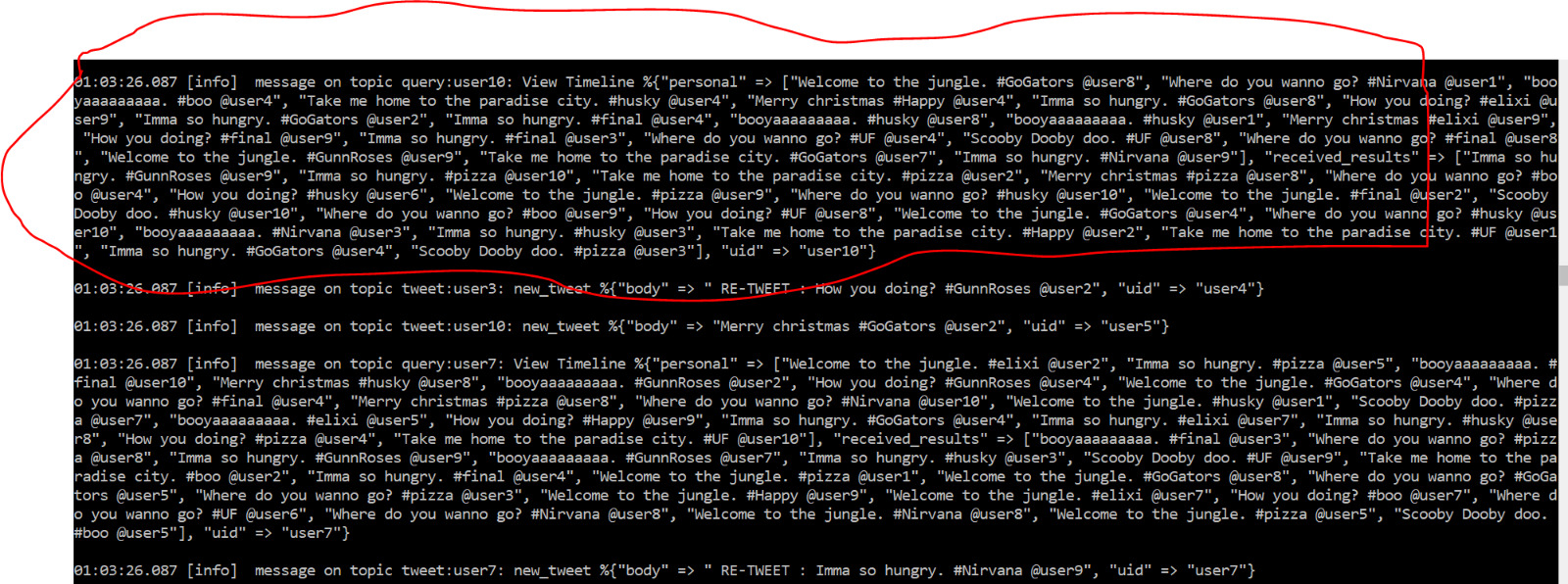
**4. Hashtag Query**



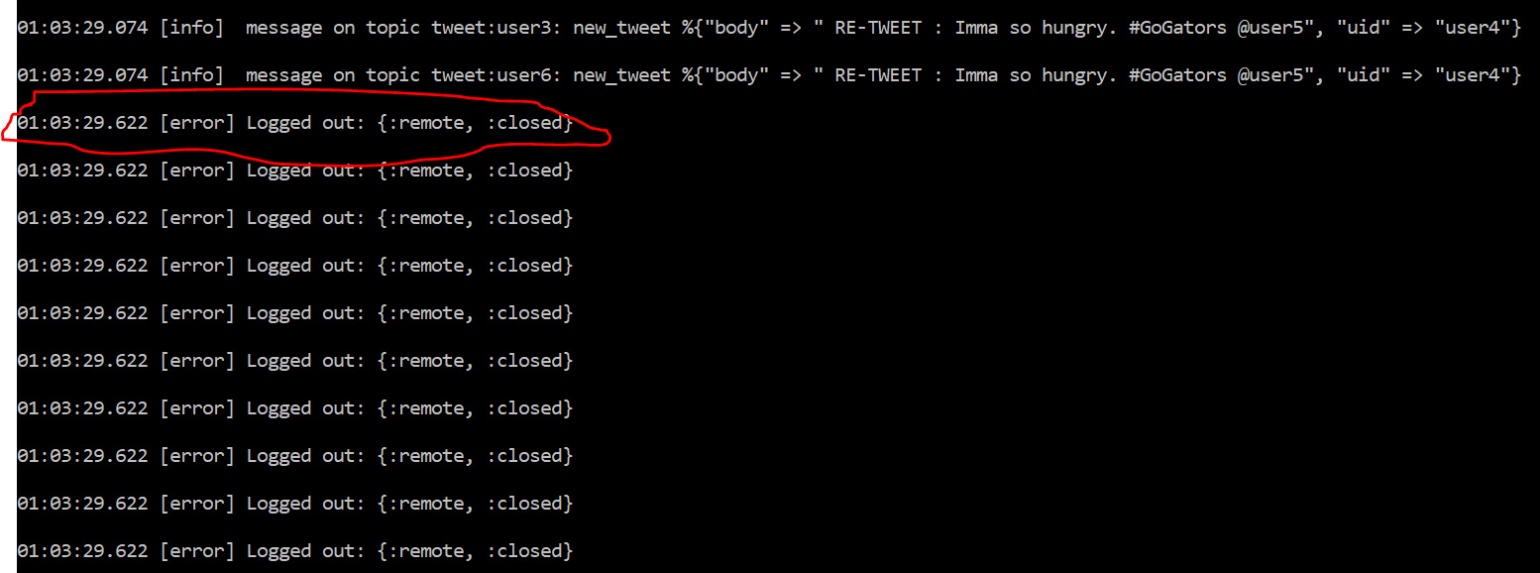
**5. Mention Query**



**6.View Timeline**



**7.Logging Out**

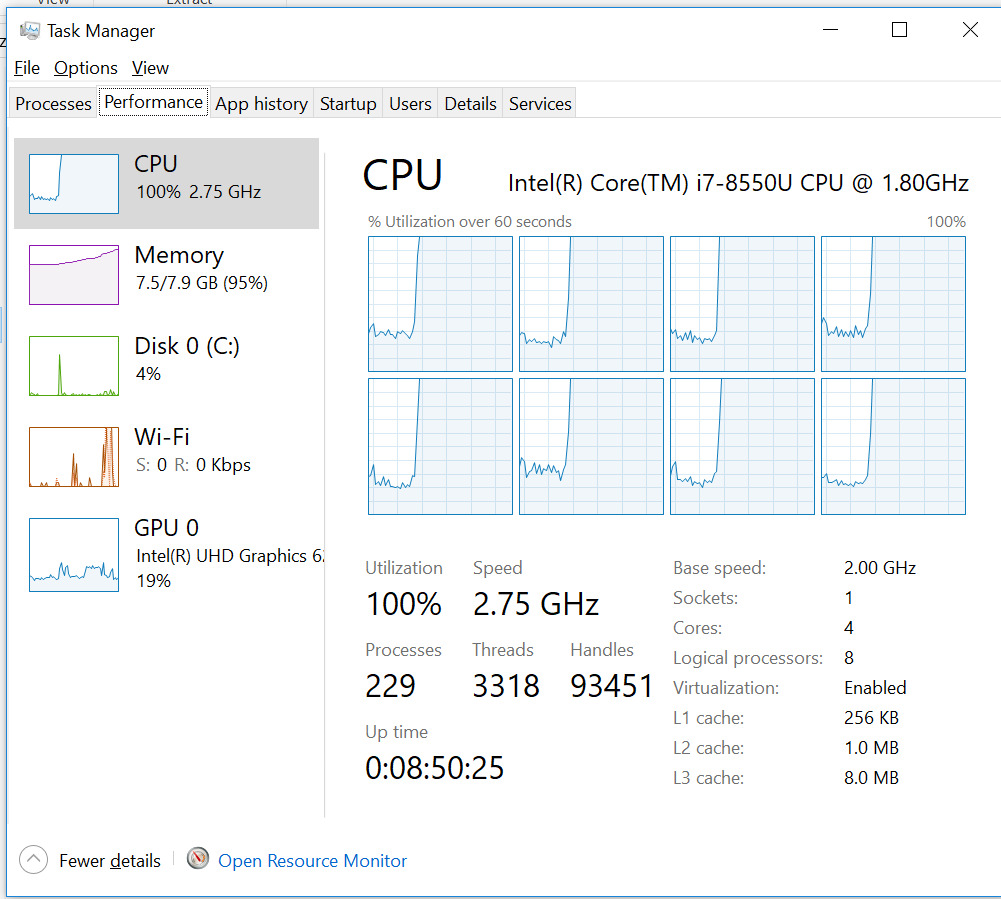


**PROJECT PERFORMANCE**

100% utilization on a quad core processor.

We tested for up to 1000 clients.

The client’s tweet and retweet is controlled by the Zipf distribution built based on the number of followers of each user.



**RESULTS AND CONCLUSION**

We did not implement UI as simulation did not seem feasible for so many clients as that many number of browser sessions must be opened. Instead we have used the library phoenix\_gen\_socket\_client to implement elixir like behavior to simulate clients in parallel without depending on browser sessions.