Distributed Operating Systems Project – 4 (Part 2):

Group Members:

Vaishnavi Dasanapu – 9005 6271

Pavan Kumar Vattikuti – 1830 2668

Outline:

The main aim of this project is to implement a webSocket interface to the part 1 implementation of the project which was "the implementation of Twitter clone engine and an Actor - model based client tester/simulator".

What is working?

- Basic Requirements - completed
- This project mainly consists of the following parts:

  - **Backend** – The functions for every activity that is supposed to take place when the user calls that function have been methodized in the backend. So, when the user tries to login or logout or get mentions or hashtags etc, he/she would press the button or enter the text depending on the activity and the button then acts according to the functions written in the server side to give the user the requested result. This was mainly covered in part 1 of the twitter project.

  - **WebSockets**:
    - These are connections that make it possible for us to establish a two-way interactive channel between the client and the server. So, when the server is running and a client logs into the server, we can see that a webSocket connection is established with the address followed by /websocket URL. We can observe this in the mainConnect module of the code.
    - Upon the establishment of the connection, handshake messaged are exchanged between the client and the server and the server receives the username from the client and this message will be stored by the server with reference to the id of the client.
    - Further, to push any updated to this current client, the corresponding actor exchanges the messages and updates through the webSocket address associated with this client.
    - On the other hand, if the client receives any messaged from the server through the webSocket, it is displayed on the live feed in the UI where the client can see it.

Pre-requisites:

- Visual Studio code, JSON, TCP/IP

Language used:

Erlang

A separate server and client simulator must be executed on two different terminals in order to see the results.

Steps to compile and run

1. Unzip the Project 4 – Part 2 folder to retrieve the client and server.erl files.
2. Open these files using the VSCode (Visual Studio Code) editor
3. Open terminal and execute the command " c(server). And server:start(). on the server machine which starts the server.
4. Open the file client.erl file
5. Perform any twitter operations on the client terminal.
6. For more detailed explanation on how to run the project and how everything works, please find the following link for a video explaining the execution, implementation and working of the code and a tour on how the twitter functions work.
The demo video link -

https://www.youtube.com/watch?v=DW2TbQafdoo&ab_channel=vyshnavidasanapu

The first step is to create an account with any service
1) **Register:** Upon completing the registration process, you will be able to login.

2) **Subscribe:** It is possible to subscribe to any account/follow any hashtag by clicking on the following button: In this case, @Vaish, @Pavan, @Harshi is an example.

3) **Tweet:** There are two ways to Tweet. When tweets include hashtags, this means that the tweets will be sent to accounts who have subscribed to that hashtag as well as those who have not subscribed to that hashtag.

4) **Retweet** - When said tweet is sent by person1 and retweeted by person2 when the same Tweet is sent by person1. It will now be possible for person 2 to retweet the same message as person 1, by selecting retweet and entering the tweet ID of person 1.

Results –

Output examples –

Command used –

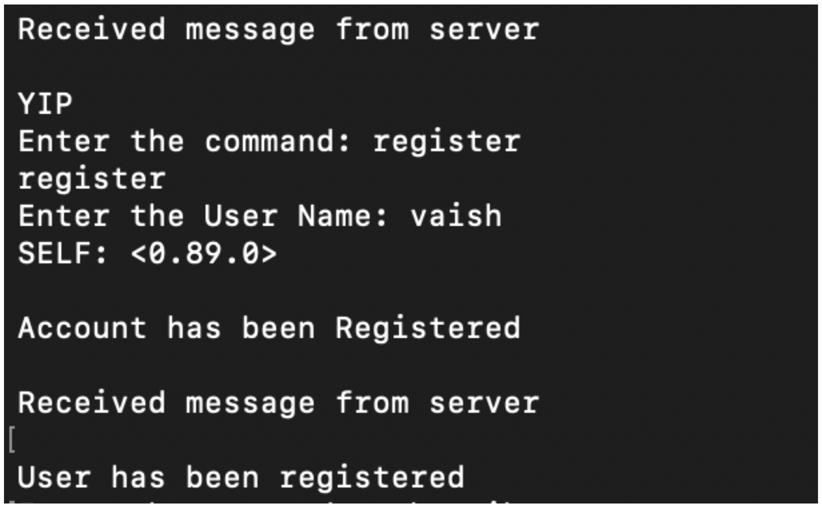
c(server).

server:start().

c(client).

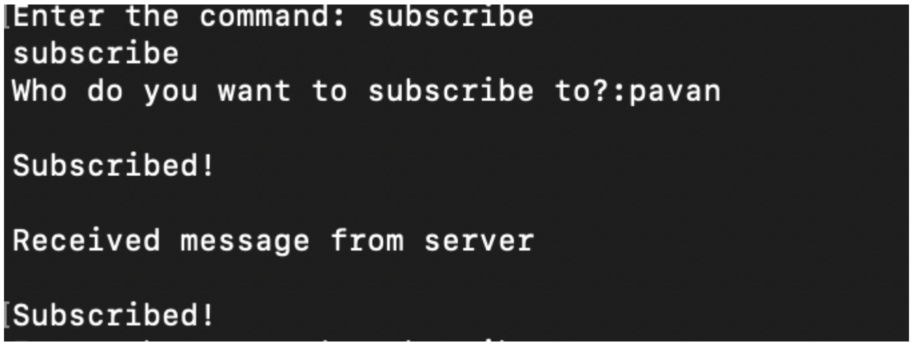
client:start().

1. **Register user**



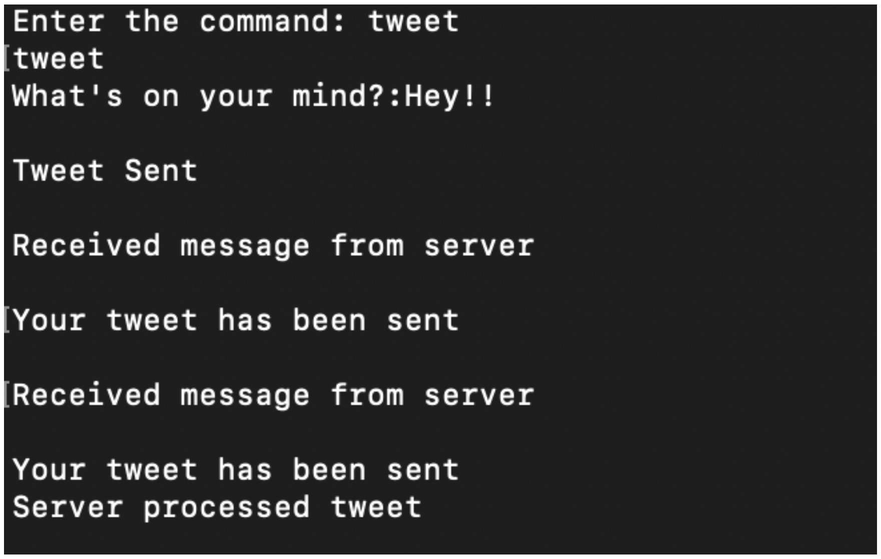
```
Received message from server

YIP
Enter the command: register
register
Enter the User Name: vaish
SELF: <0.89.0>

Account has been Registered

Received message from server
[
User has been registered
```

2. **Subscribe to another user**



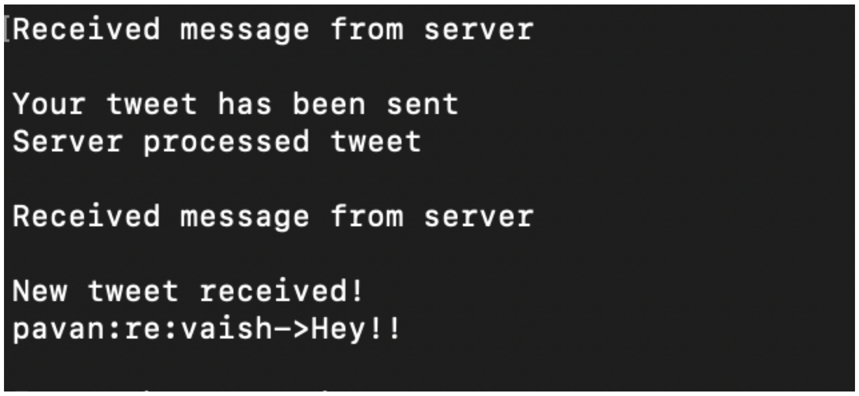
```
[Enter the command: subscribe
subscribe
Who do you want to subscribe to?:pavan

Subscribed!

Received message from server

[Subscribed!
```

3. **Tweet after login**



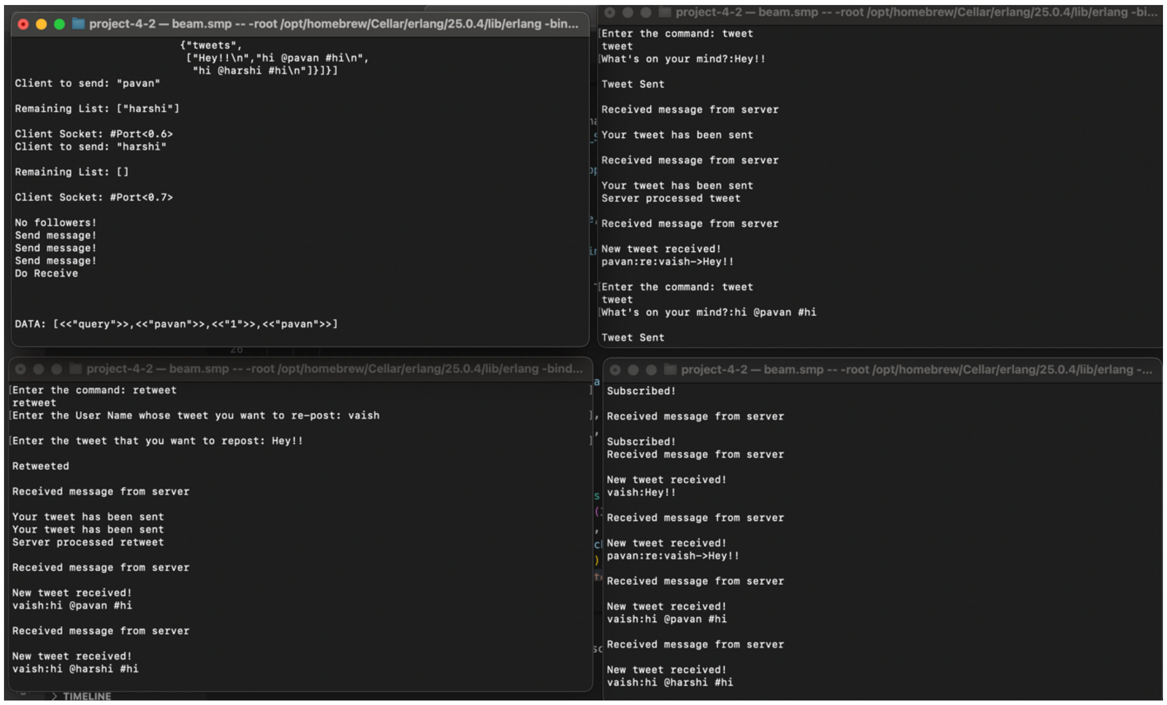
```
Enter the command: tweet
[tweet
What's on your mind?:Hey!!

Tweet Sent

Received message from server

[Your tweet has been sent

[Received message from server

Your tweet has been sent
Server processed tweet
```

4. **Retweet another user's tweet**



```
[Received message from server

Your tweet has been sent
Server processed tweet

Received message from server

New tweet received!
pavan:re:vaish->Hey!!
```

5. Retweets getting displayed in other users accounts



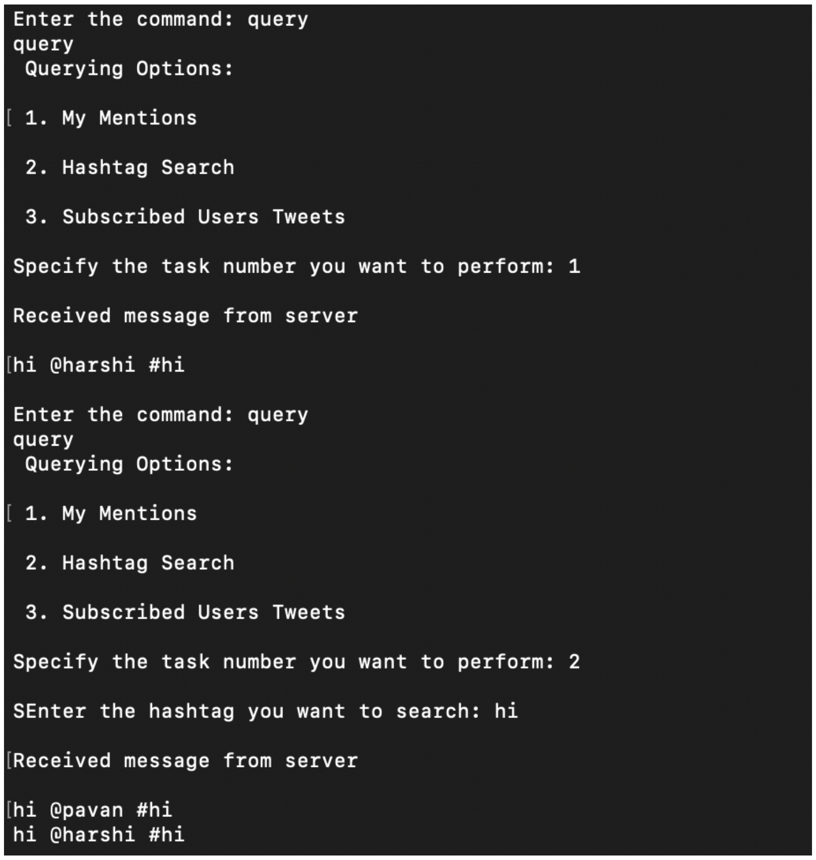
```
                          {"tweets",
                           ["Hey!!\n","hi @pavan #hi\n",
                            "hi @harshi #hi\n"]}]}]
Client to send: "pavan"

Remaining List: ["harshi"]

Client Socket: #Port<0.6>
Client to send: "harshi"

Remaining List: []

Client Socket: #Port<0.7>

No followers!
Send message!
Send message!
Send message!
Do Receive



DATA: [<<"query">>,<<"pavan">>,<<"1">>,<<"pavan">>]
```

```
Enter the command: tweet
tweet
What's on your mind?:Hey!!

Tweet Sent

Received message from server

Your tweet has been sent

Received message from server

Your tweet has been sent
Server processed tweet

Received message from server

New tweet received!
pavan:re:vaish->Hey!!

Enter the command: tweet
tweet
What's on your mind?:hi @pavan #hi

Tweet Sent
```

```
Enter the command: retweet
retweet
Enter the User Name whose tweet you want to re-post: vaish

Enter the tweet that you want to repost: Hey!!

Retweeted

Received message from server

Your tweet has been sent
Your tweet has been sent
Server processed retweet

Received message from server

New tweet received!
vaish:hi @pavan #hi

Received message from server

New tweet received!
vaish:hi @harshi #hi
```

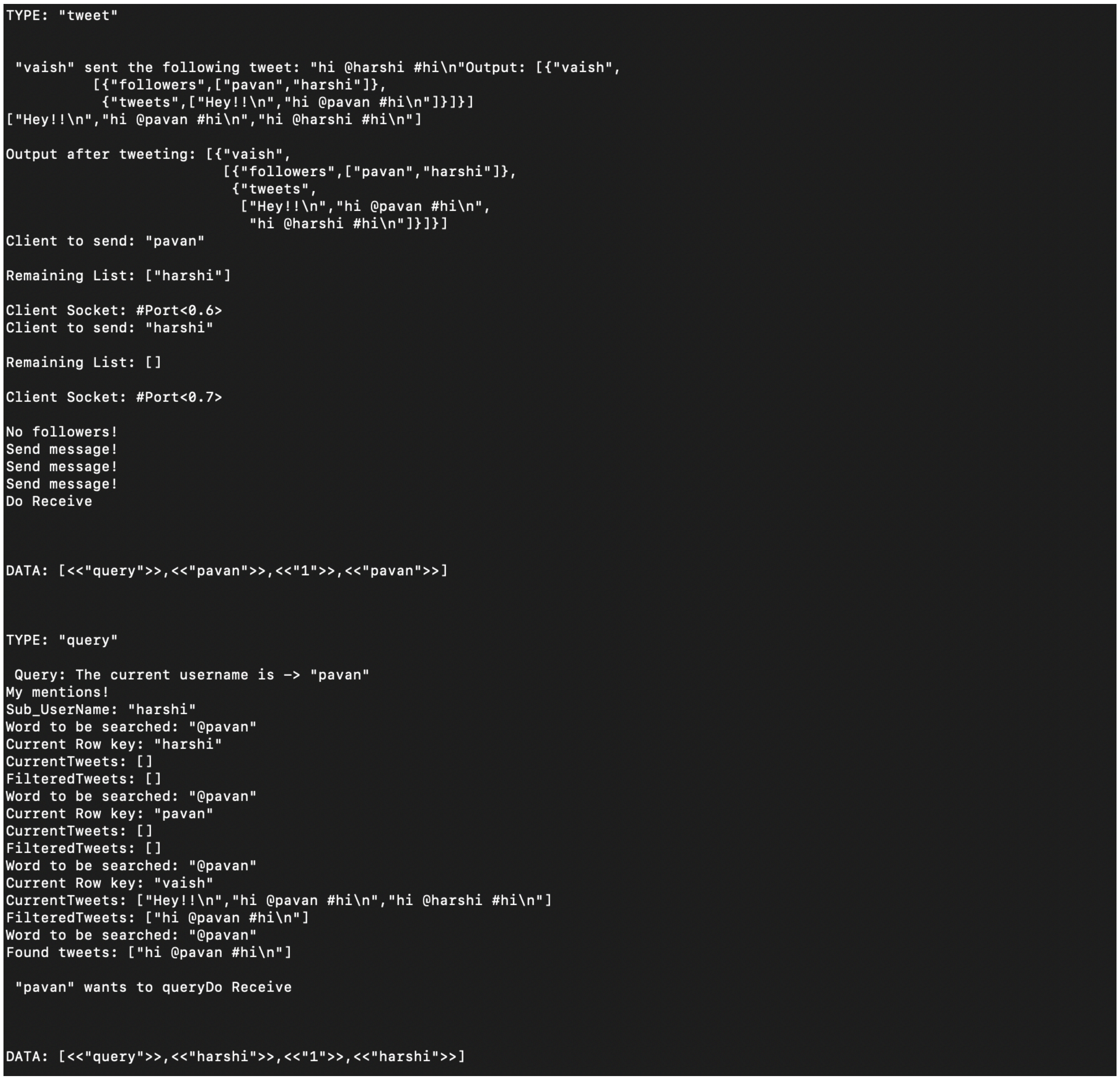
```
Subscribed!

Received message from server

Subscribed!
Received message from server

New tweet received!
vaish:Hey!!

Received message from server

New tweet received!
pavan:re:vaish->Hey!!

Received message from server

New tweet received!
vaish:hi @pavan #hi

Received message from server

New tweet received!
vaish:hi @harshi #hi
```

6. Querying



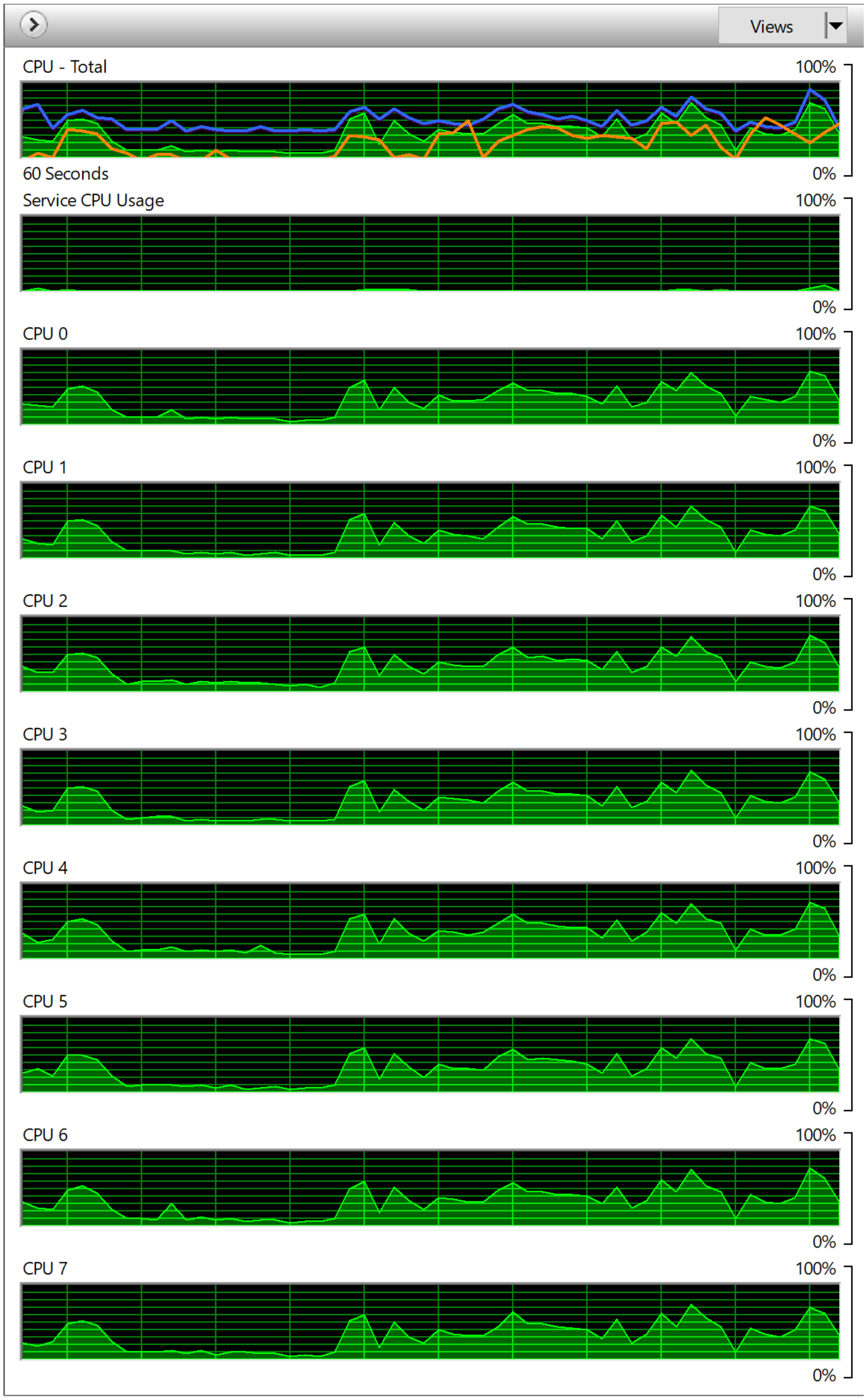
```
Enter the command: query
query
 Querying Options:

 1. My Mentions

 2. Hashtag Search

 3. Subscribed Users Tweets

Specify the task number you want to perform: 1

Received message from server

hi @harshi #hi

Enter the command: query
query
 Querying Options:

 1. My Mentions

 2. Hashtag Search

 3. Subscribed Users Tweets

Specify the task number you want to perform: 2

SEnter the hashtag you want to search: hi

Received message from server

hi @pavan #hi
hi @harshi #hi
```

7. Server Side outputs



```
TYPE: "tweet"


 "vaish" sent the following tweet: "hi @harshi #hi\n"Output: [{"vaish",
          [{"followers",["pavan","harshi"]},
           {"tweets",["Hey!!\n","hi @pavan #hi\n"]}]}]
["Hey!!\n","hi @pavan #hi\n","hi @harshi #hi\n"]

Output after tweeting: [{"vaish",
                         [{"followers",["pavan","harshi"]},
                          {"tweets",
                           ["Hey!!\n","hi @pavan #hi\n",
                            "hi @harshi #hi\n"]}]}]
Client to send: "pavan"

Remaining List: ["harshi"]

Client Socket: #Port<0.6>
Client to send: "harshi"

Remaining List: []

Client Socket: #Port<0.7>

No followers!
Send message!
Send message!
Send message!
Do Receive



DATA: [<<"query">>,<<"pavan">>,<<"1">>,<<"pavan">>]



TYPE: "query"

 Query: The current username is -> "pavan"
My mentions!
Sub_UserName: "harshi"
Word to be searched: "@pavan"
Current Row key: "harshi"
CurrentTweets: []
FilteredTweets: []
Word to be searched: "@pavan"
Current Row key: "pavan"
CurrentTweets: []
FilteredTweets: []
Word to be searched: "@pavan"
Current Row key: "vaish"
CurrentTweets: ["Hey!!\n","hi @pavan #hi\n","hi @harshi #hi\n"]
FilteredTweets: ["hi @pavan #hi\n"]
Word to be searched: "@pavan"
Found tweets: ["hi @pavan #hi\n"]

 "pavan" wants to queryDo Receive



DATA: [<<"query">>,<<"harshi">>,<<"1">>,<<"harshi">>]
```

System CPU Utilization – This machine has 4 cores (8 logical processors)


Views
CPU - Total
100%
60 Seconds
0%
Service CPU Usage
100%
0%
CPU 0
100%
0%
CPU 1
100%
0%
CPU 2
100%
0%
CPU 3
100%
0%
CPU 4
100%
0%
CPU 5
100%
0%
CPU 6
100%
0%
CPU 7
100%
0%