Project 4 Part 2 Report

1.Team Members

Name: Zixun Wang UFID: 3725-9823

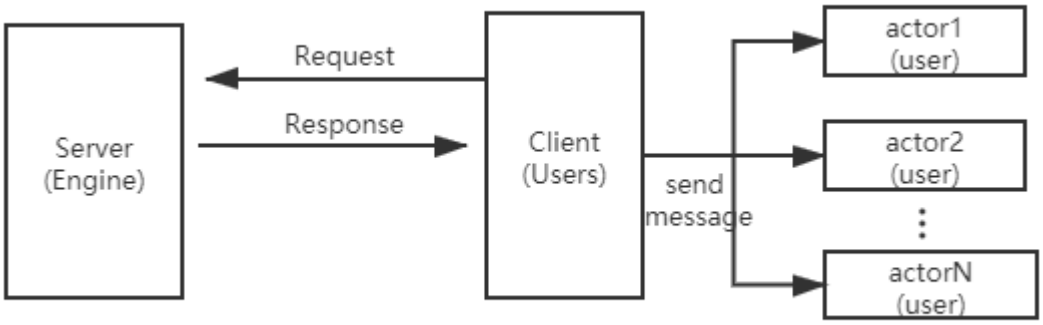
Name: Yixin Wei UFID: 5114-6181

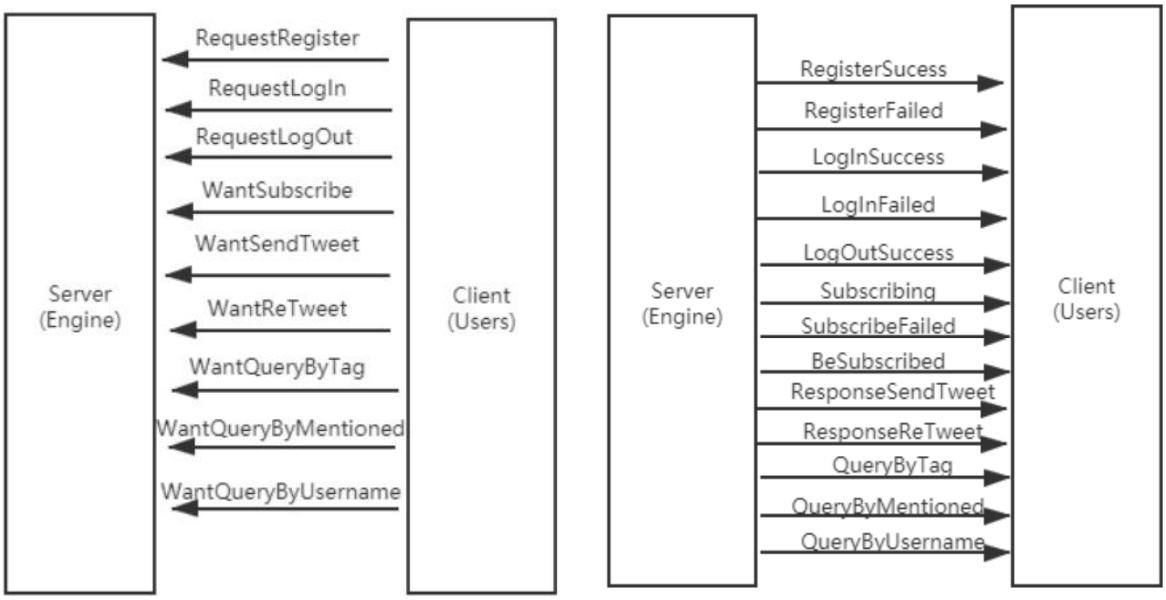
2.Implementation

In part I, we used one engine process and one user process to simulate the twitter. In part II, we used WebSharper web framework to implement a WebSocket interface to our part I implementation.

We implemented part II with Websharper framework, which supports remote procedure calls from the client (JavaScript environment) to the server (ASP.NET). This project used a JSON based API that represents all messages and their replies.

The architecture of Client, Server and Message passing that are used in our Twitter Simulator project are shown below:





Firstly, a user sends a request to the server, for example, a user wants to register an account, which is “RequestRegister” in the left figure above. And then, the server response the request, for example, the server can check whether the username exist or not and accept or refuse the request, which is “RegisterSuccess” or “RegisterFailed” in the right figure above. Receiving response from server, the client can send message to corresponding actor, which represents one user in our twitter simulator.

In detail, we use two types (as the figures shown bellow), type C2SMessage and type S2CMessage in code of server part to exchange messages in server and client. We use type C2SMessage to send message from client to server, and S2CMessage to send message from server to client.

And in code of client, we build multiple actors to execute different tasks, including sign up, send tweets, search, subscribe, retweet and query.

```
type [<JavaScript; NamedUnionCases>]
    C2SMessage =
    | RequestRegister of username0:string * psw0:string  //username,password
    | RequestLogIn of username1:string * psw1:string  //become active
    | RequestLogOut of username2:string  //become inactive
    | WantSubscribe of username3:string * subscribeto3:string
    | WantSendTweet of username4:string * tweetcontent4:string
    | WantReTweet of username5:string * tweetcontent5:string
    | WantQueryByTag of username6:string * tag:string
    | WantQueryByMentioned of username7:string * mentioned:string
    | WantQueryByUsername of username8:string * queriedusername:string
```

```
and [<JavaScript; NamedUnionCases "type">]
    S2CMessage =
    | RegisterSuccess of username9:string * psw9:string  //username,password
    | RegisterFailed of username19:string
    | LogInSuccess of username10:string * psw10:string  //become active
    | LogInFailed
    | LogOutSuccess of username11:string  //become inactive
    | Subscribing of username12:string* subscribeto12:string
    | SubscribeFailed of username20:string * subscribeto20:string
    | BeSubscribed of subscribeto13:string * username13:string
    | ResponseSendTweet of username14:string * tweetcontent14:string * followers14: string list
    | ResponseReTweet of username15:string * tweetcontent15:string
    | QueryByTag of username16:string * tag16:string
    | QueryByMentioned of username17:string * mentioned17:string
    | QueryByUsername of username18:string * queriedusername18:string
```

We also use the functions in the WebSharper.UI.Html module, construct a simple HTML with a text input box and a "send" button. We can input commands to simulate the circumstance when an user uses the twitter simulator. Use the following code in code of client part to display the contents of the web page.

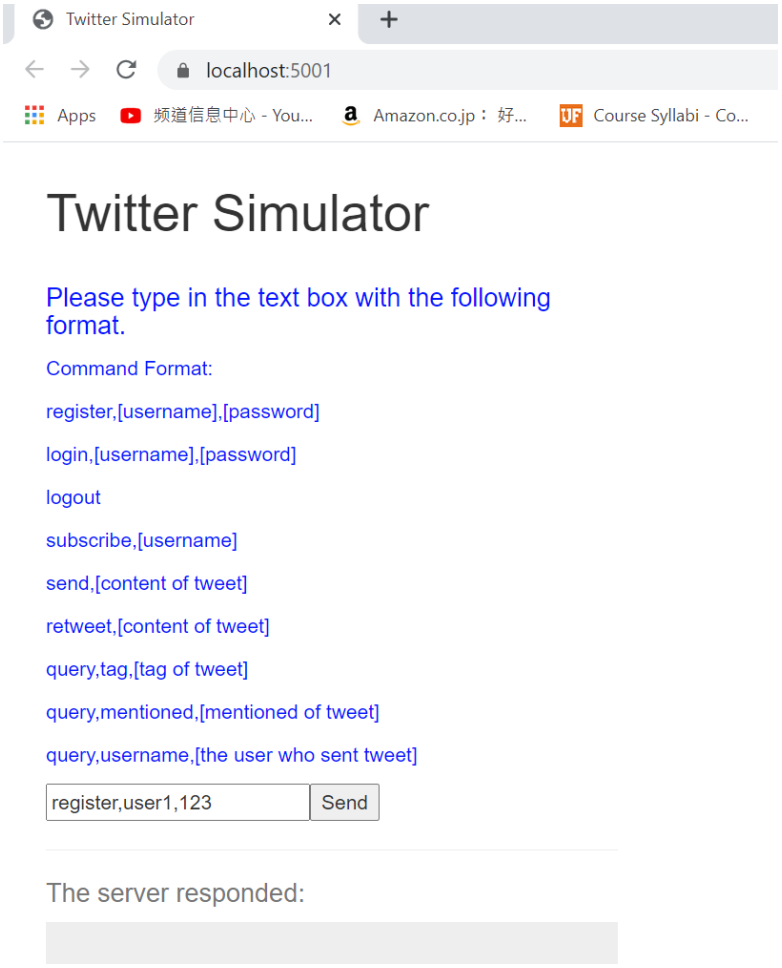
```
div [] [

    Doc.Input [] vInput
    Doc.Button "Send" [] submit.Trigger
    hr [] []
    h4 [attr.``class`` "text-muted"] [text "The server responded:"]
    div [attr.``class`` "jumbotron"] [h4 [] [textView vTextView]]

    container

]
```

3.Results and Observations

We showed how to run the code and how to use functions of our program in the video: https://recordit.co/5VoWFv1hyh .

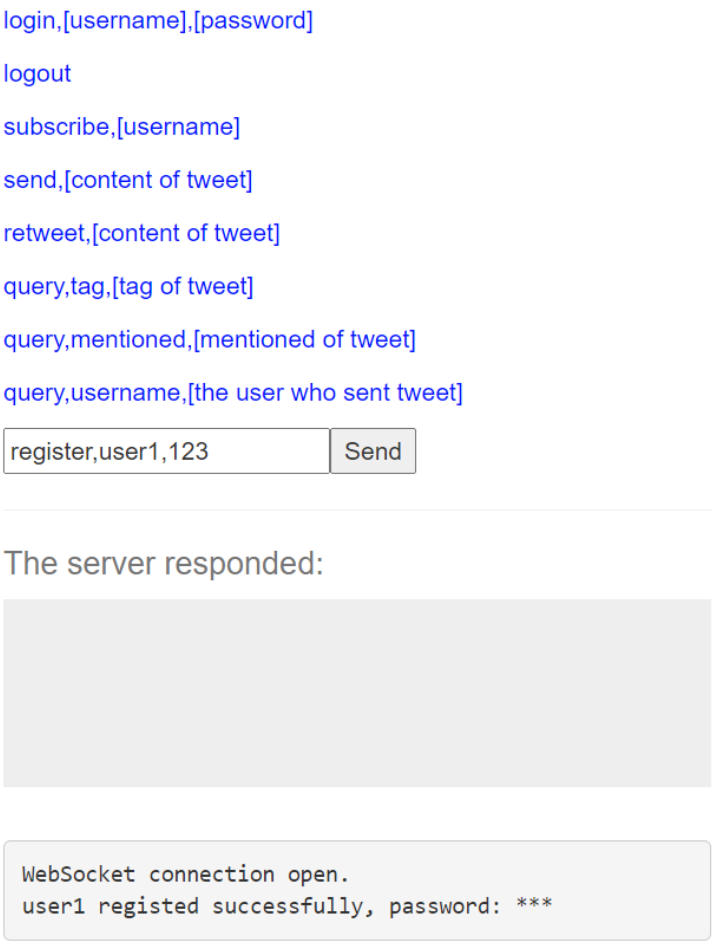
Here are some screenshots of our program, which show the functions of our twitter simulator.

a. Here is the home page of our program, it uses “localhost:5001” to open the page.



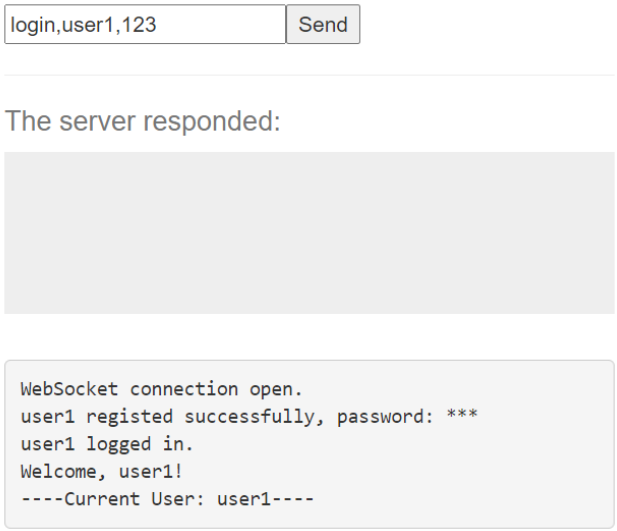
b. Register account

Firstly, use command "register,[username],[password]" to register an account. (Ex: register,user1,12345).



c. Log in

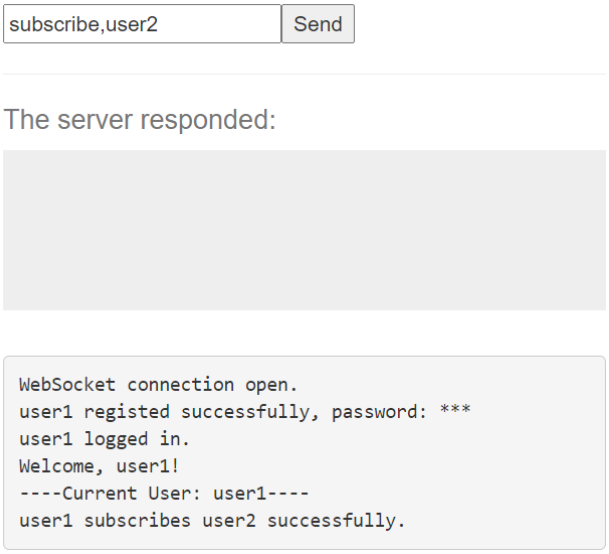
Use command "login,[username],[password]" to log in an existing account. (Ex: login,user1,12345)



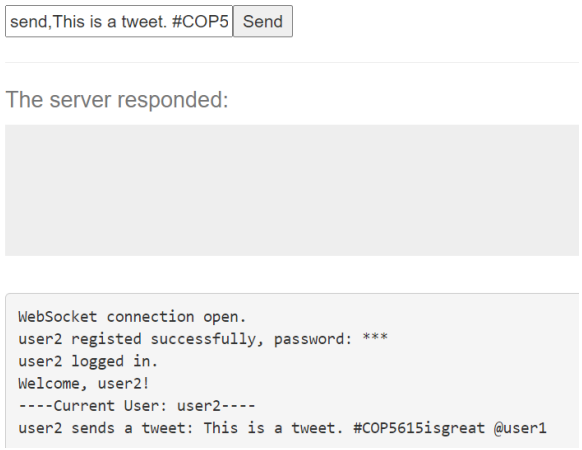
d. Subscribe to user's tweets

Use command "subscribe,[username]" to subscribe another user. (Ex: subscribe,user2)

First register another account in a new page. And login the second account.



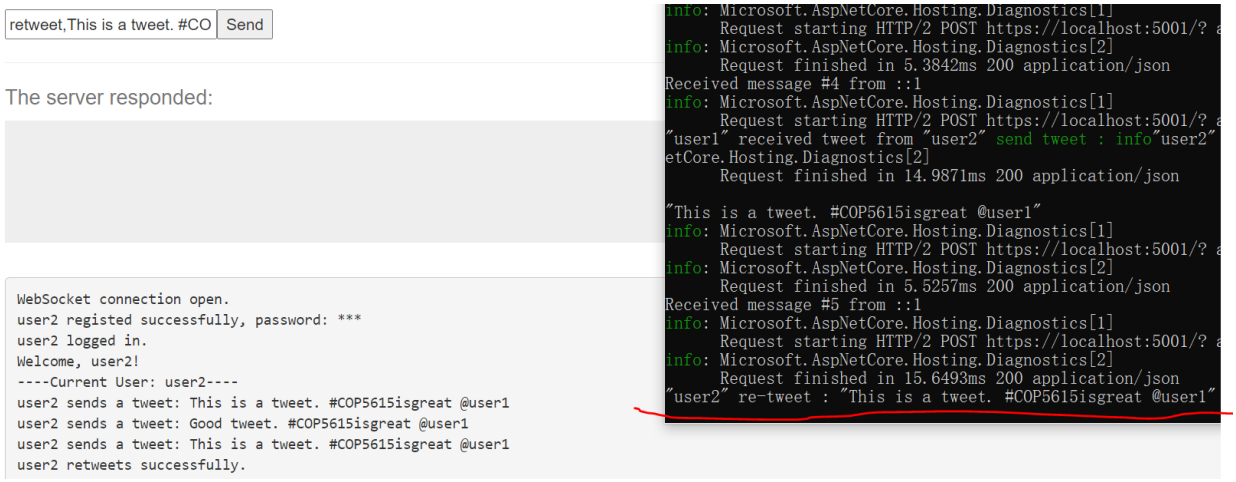
e. Send tweet



f. Re-tweet

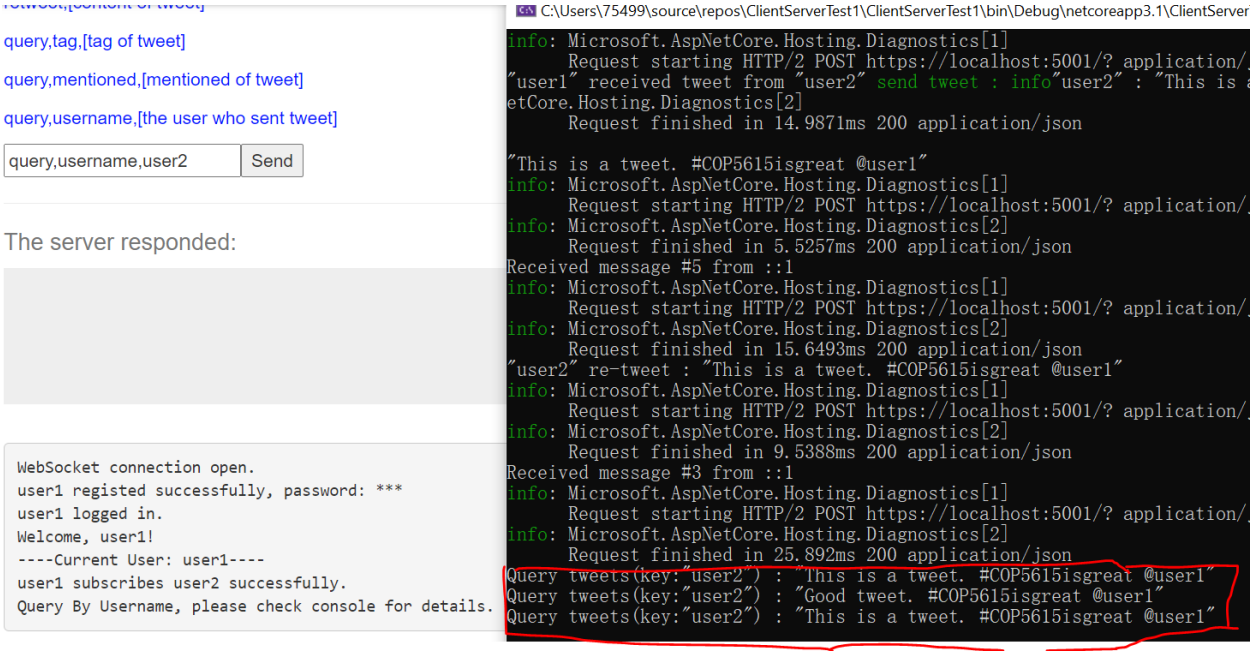
User command "retweet,[content of tweet]" to retweet to the current user's followers.(Ex: retweet,This is a tweet #tag1 @user3)

Now we can see that the user2 retweets successfully from the console and the html page.



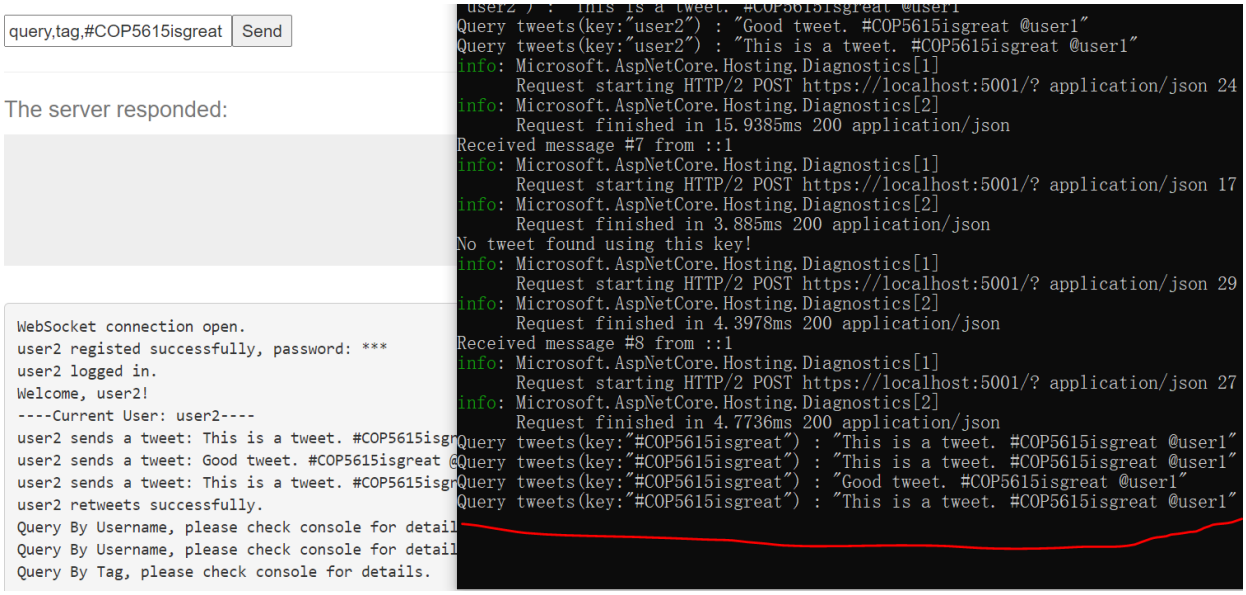
g. Querying tweets subscribed to

User command "query,username,[the user who sent tweet]" to query tweets by the sender of the tweet. (Ex: query,username,user1)



h. Querying tweets with specific hashtags (e.g. #COP5615isgreat)

User command "query,tag,[tag of tweet]" to query tweets by the content of tag. (Ex: query,tag,#tag1)



i. Querying tweets in which the user is mentioned (e.g. @user1)

User command "query,mentioned,[mentioned of tweet]" to query tweets by the content of mentioned. (Ex: query,mentioned,@user3)

query,mentioned,@user1 Send

The server responded:

```
WebSocket connection open.
user2 registed successfully, password: ***
user2 logged in.
Welcome, user2!
----Current User: user2----
user2 sends a tweet: This is a tweet. #COP5615isgreat @user1
user2 sends a tweet: Good tweet. #COP5615isgreat @user1
user2 sends a tweet: This is a tweet. #COP5615isgreat @user1
user2 retweets successfully.
Query By Username, please check console for details.
Query By Username, please check console for details.
Query By Tag, please check console for details.
Query By Mentioned, please check console for details.
```

C:\Users\75499\source\repos\ClientServerTest1\ClientServerTest1\bin\Debug\netcoreapp3.1\ClientServ

```
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 3.885ms 200 application/json
No tweet found using this key!
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 POST https://localhost:5001/? application
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 4.3978ms 200 application/json
Received message #8 from ::1
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 POST https://localhost:5001/? application
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 4.7736ms 200 application/json
Query tweets(key:"#COP5615isgreat") : "This is a tweet. #COP5615isgreat
Query tweets(key:"#COP5615isgreat") : "This is a tweet. #COP5615isgreat
Query tweets(key:"#COP5615isgreat") : "Good tweet. #COP5615isgreat @use
Query tweets(key:"#COP5615isgreat") : "This is a tweet. #COP5615isgreat
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 POST https://localhost:5001/? application
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 4.3722ms 200 application/json
Received message #9 from ::1
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/2 POST https://localhost:5001/? application
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished in 12.0835ms 200 application/json
Query tweets(key:"@user1") : "This is a tweet. #COP5615isgreat @user1"
Query tweets(key:"@user1") : "This is a tweet. #COP5615isgreat @user1"
Query tweets(key:"@user1") : "Good tweet. #COP5615isgreat @user1"
Query tweets(key:"@user1") : "This is a tweet. #COP5615isgreat @user1"
```

j. Log out

You can use command "logout" at any time to disconnect the current user. (Ex: logout)

logout Send

The server responded:

user2 has logged out!

```
WebSocket connection open.
user2 registed successfully, password: ***
user2 logged in.
Welcome, user2!
----Current User: user2----
user2 sends a tweet: This is a tweet. #COP5615isgreat @user1
user2 sends a tweet: Good tweet. #COP5615isgreat @user1
user2 sends a tweet: This is a tweet. #COP5615isgreat @user1
user2 retweets successfully.
Query By Username, please check console for details.
Query By Username, please check console for details.
Query By Tag, please check console for details.
Query By Mentioned, please check console for details.
user2 logged out.
```

And also, the program can show errors when command is wrong or information doesn’t exist. We can see these circumstances in our video link.

In conclusion, we implemented the websocket and Json-based API which communicate through websocket with WebSharper framework and realized the above functions.