COP5615- Distributed Operating System Principles

Project 4.2

Developed by –
Shaik, Fayaz - UFID: 6326-9935
Mullapudi, Aseesh - UFID: 9175-1971

Brief Description:

The goal of the project is to design and use a proper WebSocket interface and REST API to develop a Twitter Clone using Suave web framework on top of part I implementation using AKKA messaging to allow client-server implementation.

Our project supports all the basic operations like Registration, Authentication, Tweeting, Retweeting, Querying by Hashtags and Mentions that a twitter clone is expected to have.

Functionalities implemented are:

- Front-end for the Twitter engine
- **REST API and Web Sockets** interface implementation
- Register User
- Login and Logout User
- Subscribe/Follow a User
- Send Tweets. Tweets can have mentions and hashtags
- Retweeting
- The News Feed of the active users is updated real-time and when the user comes online, they can also see the tweets which they have missed
- Querying all the mentions (when the user is mentioned in a tweet) and hashtags

Demo – YouTube Link – https://youtu.be/SOCOlX8IvK8

How to Run the Project:

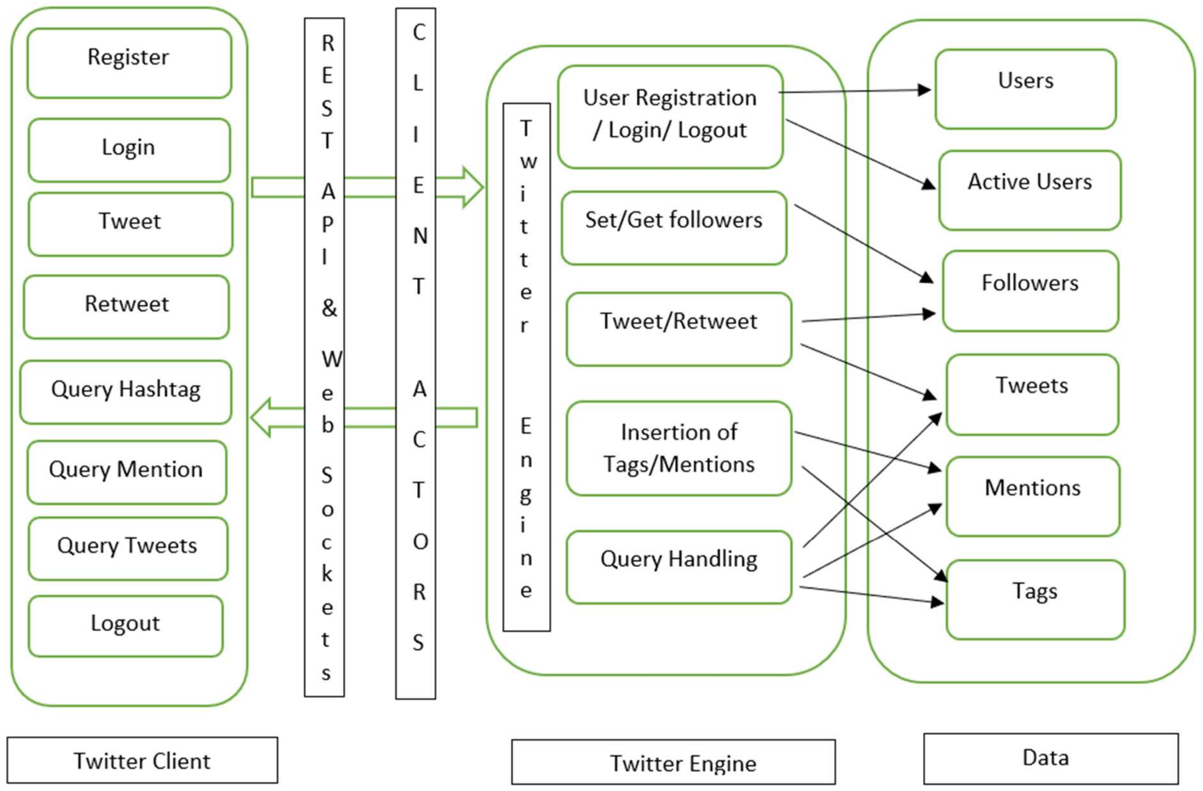
Program.fs – This file contains the main function (entry point) of the web server.

Inorder to start the Twitter Engine (or the web server of the twitter), extract the zip file and then from the project directory, run the F# project in the below fashion.

To Run – dotnet run

The above command helps in starting the webserver at **localhost:8080**. From now, all the clients can access the twitter clone at localhost:8080.

We can open 3-4 tabs in the browser (to access the server) to test as 3-4 different twitter users.



Architecture

Implementation Details:

Front end – UI

The UI of the twitter that has been implemented (code present in ***index.html***), helps the user to have a great user experience and the display is self-explanatory and sufficient for the user to perform their operations. A detailed explanation of the user flow is explained later in the report below.

Web Server (RESI API & WebSocket)

The ***Program.fs*** file contains the Suave Web server that has the REST API and the web socket initialization code. Whenever a user registers, a Client Actor is spawned and a WebSocket is created, and all the user’s operations are handled by this client actor. There is also a mapping between the user and their corresponding WebSocket inorder to send the response messages to the UI of the client.

We are using REST API for handling all the GET requests for Registration and Login purposes.
We are using Web Sockets for all the Server to Client communication purposes.

Twitter Client & Twitter Engine

The ***TwitterClient.fs*** file contains the code that can help each client invoke their own operations to the Twitter Engine. The ***TwitterEngine.fs*** file contains the actual Twitter engine code changes that takes care of the client's requests.

The ***Program.fs*** file is where all the client actors are spawned. The Suave web server calls the client (in ***TwitterClient.fs***), and the client passes the type of operation to be done to the server. Also, each client has an active news Feed and an inactive news Feed. Whenever a user which the client follows tweets, that tweet is pushed into the active news Feed and if our client is inactive or has logged out, that tweet is pushed in the inactive news feed list so that the client can view these tweets when it comes back online. Each client also has a mentions list which on querying collects the tweets in which the user/client is mentioned.

The ***TwitterEngine.fs*** is where all the operations take place. Each operation has its own function. All the storage of the data is being done in-memory by using HashMaps. There are. There are maps for –

- User and user's tweetIds – userId as key and set of tweetIds as value
- User and user's followers – userId as key and set of followers as value
- tweetId and tweet – tweetId as key and tweet as value
- mention and tweetIds – mention as key and set of tweetIds
- hashtag and tweetIds – hashtag as key and set of tweetIds
- Set of active Users

Whenever a user tweets, the tweet is sent to the user's followers if they are active (logged in).

In the Retweet functionality – a user can select a tweet from their news Feed and tweet it.

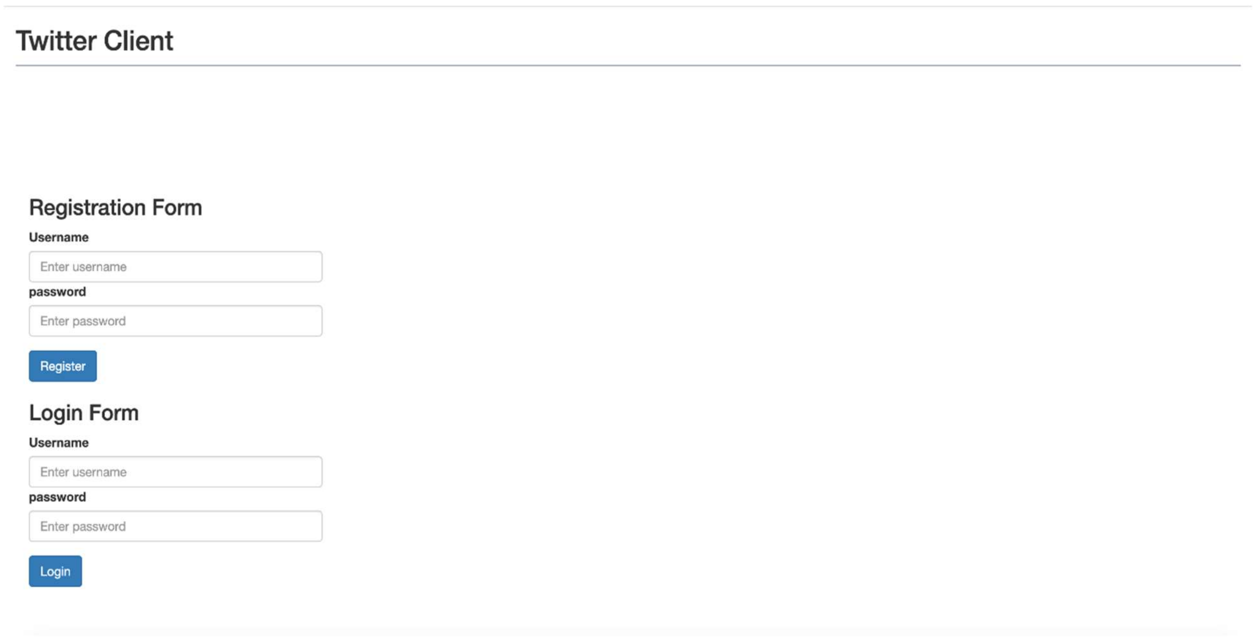
<u>Screenshots of a user flow:</u>

Let us assume that for simplicity, once the server is started, there are 2 clients, user1 and user2 that are using the twitter platform.
Let user2 follow user1. The below screenshots show the UI screens of both the users (user1 and user2) at different stages during the operations.

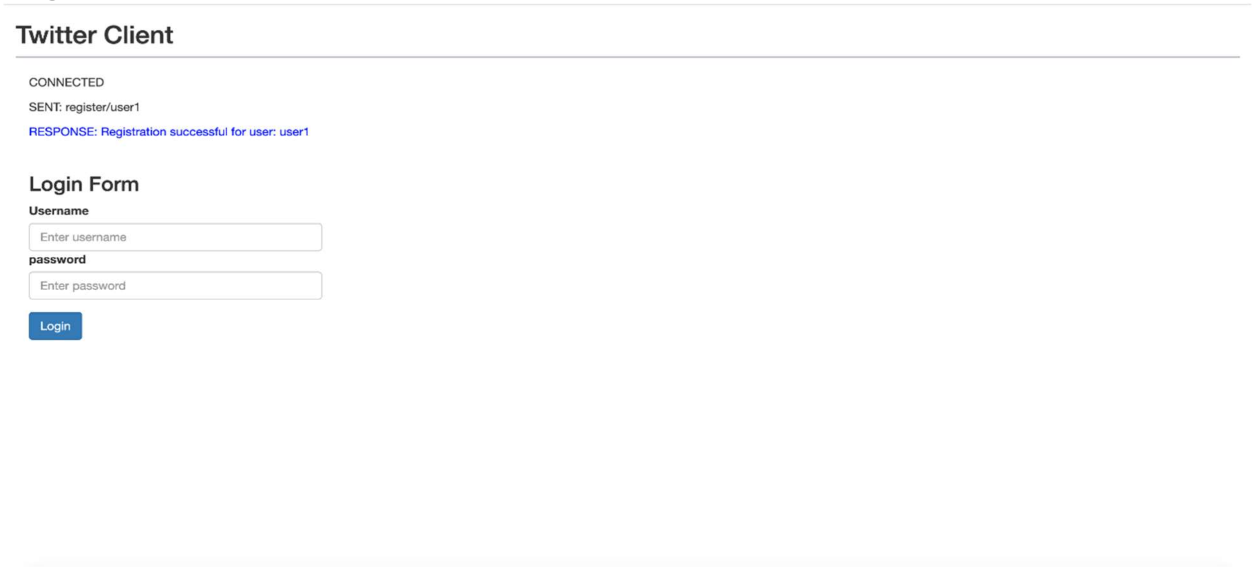
Operations –

1) user1, user2 registrations and logins
2) user2 follow user1
3) user1 tweets
4) user2 retweets user1's tweet found on their timeline
5) user2 queries mention "@user2"
6) user2 queries hashtag "#greeting"

Home page –

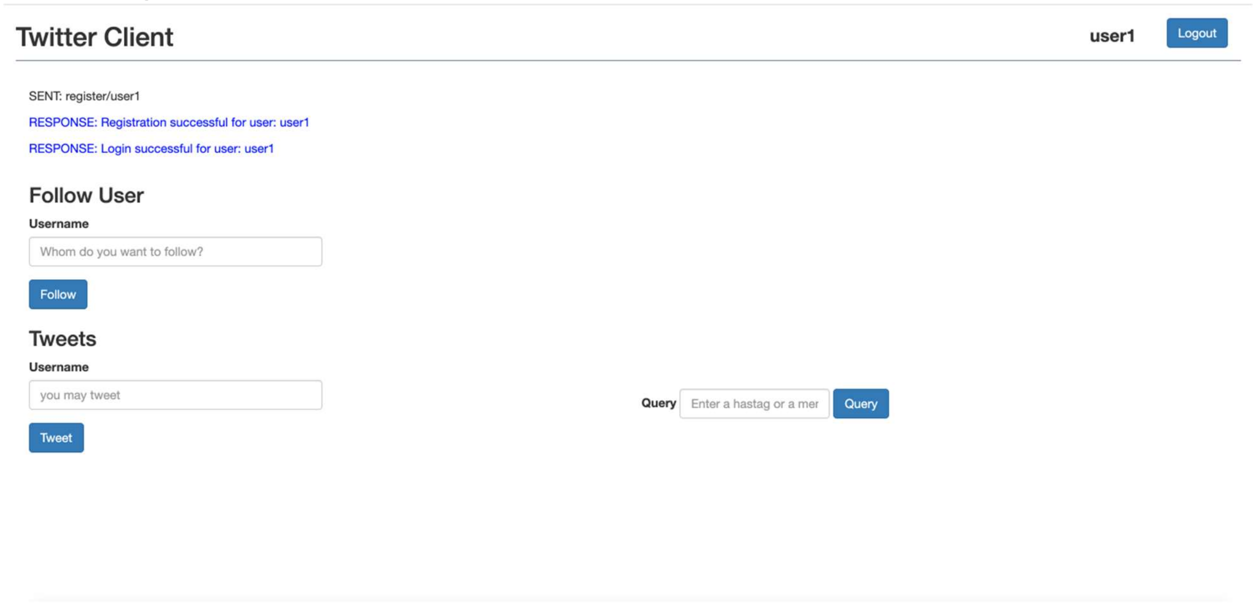


Registration Successful –



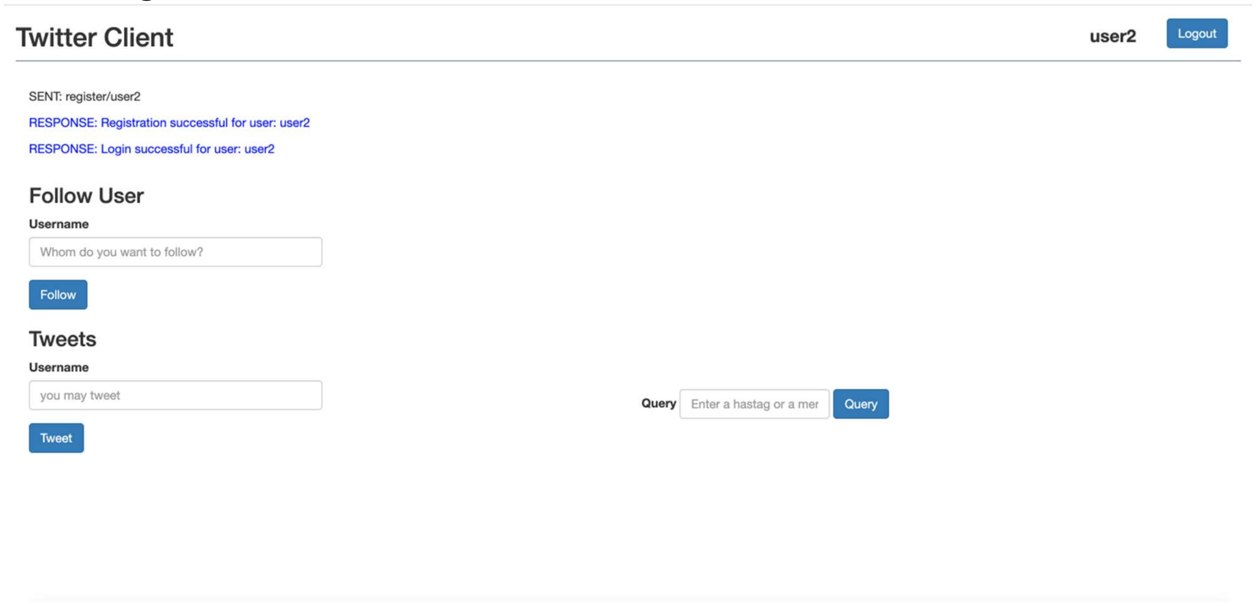
```
CONNECTED
SENT: register/user1
RESPONSE: Registration successful for user: user1
```

User1 Login –



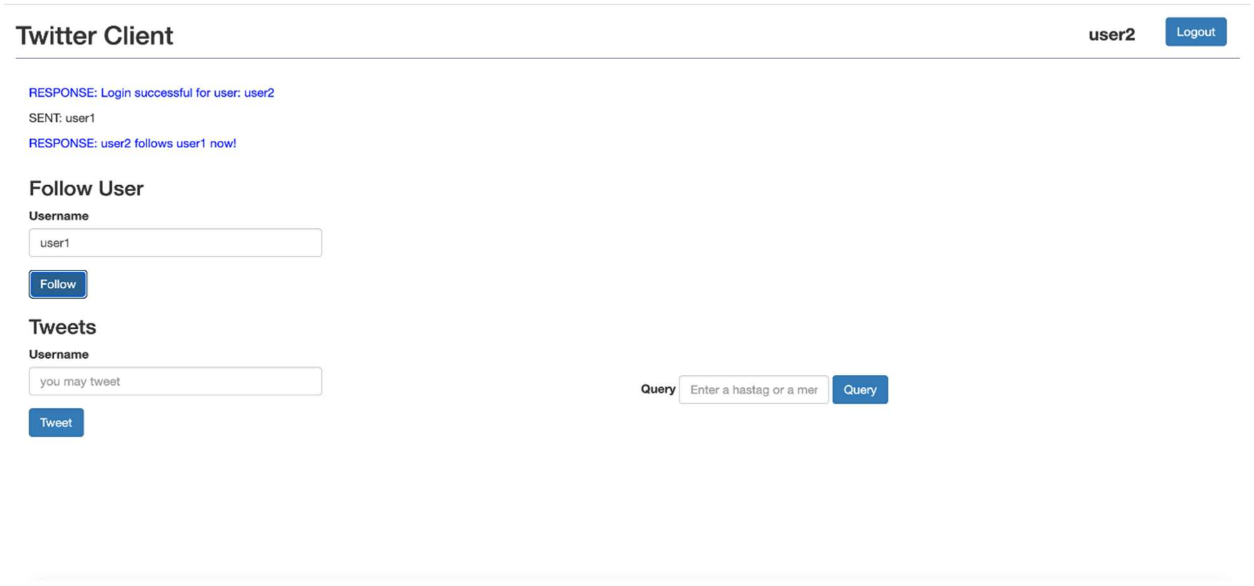
```
SENT: register/user1
RESPONSE: Registration successful for user: user1
RESPONSE: Login successful for user: user1
```

User2 Login –



```
SENT: register/user2
RESPONSE: Registration successful for user: user2
RESPONSE: Login successful for user: user2
```

Now User2 follows/subscribes to User1 –



Twitter Client

user2

RESPONSE: Login successful for user: user2

SENT: user1

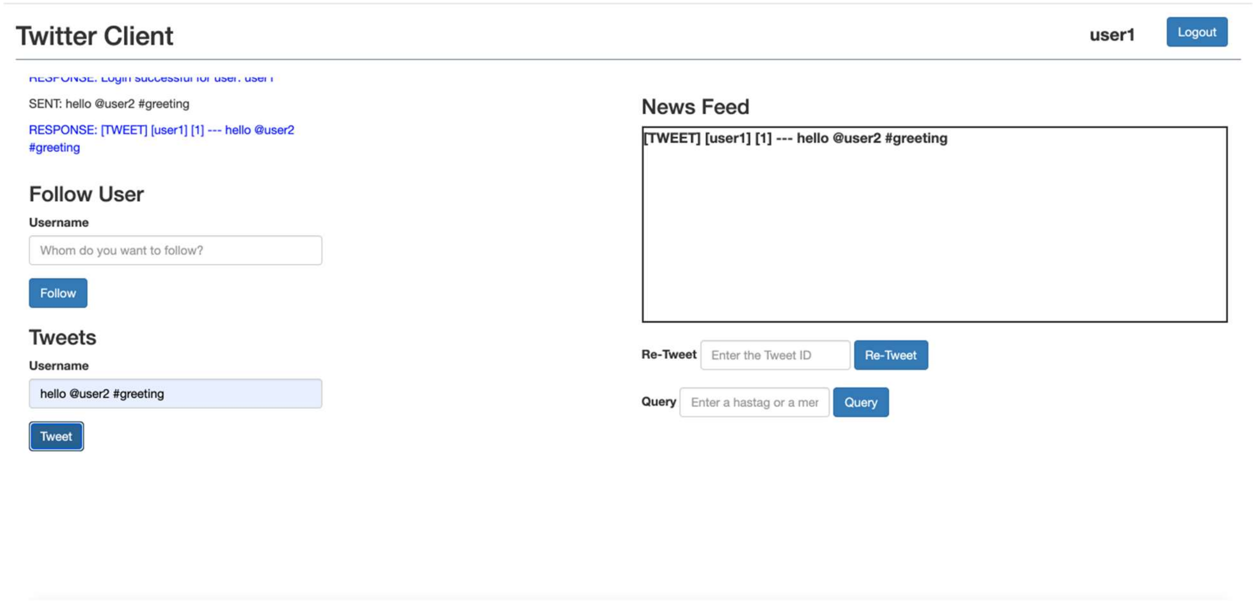
RESPONSE: user2 follows user1 now!

Follow User

Username

user1

User1 tweets –



Twitter Client

user1

RESPONSE: Login successful for user: user1

SENT: hello @user2 #greeting

RESPONSE: [TWEET] [user1] [1] --- hello @user2 #greeting

News Feed

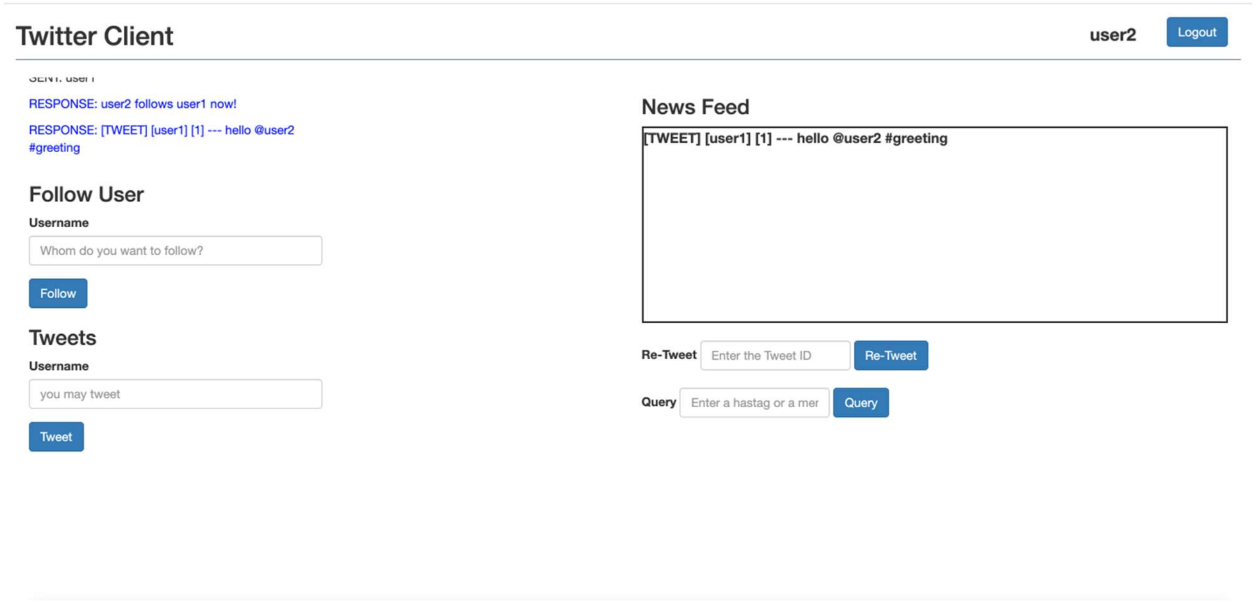
[TWEET] [user1] [1] --- hello @user2 #greeting

Tweets

Username

hello @user2 #greeting

User2 gets User1’s tweet on his News feed –



Twitter Client

user2

Logout

SENT: user1

RESPONSE: user2 follows user1 now!

RESPONSE: [TWEET] [user1] [1] --- hello @user2 #greeting

Follow User

Username

Whom do you want to follow?

Follow

Tweets

Username

you may tweet

Tweet

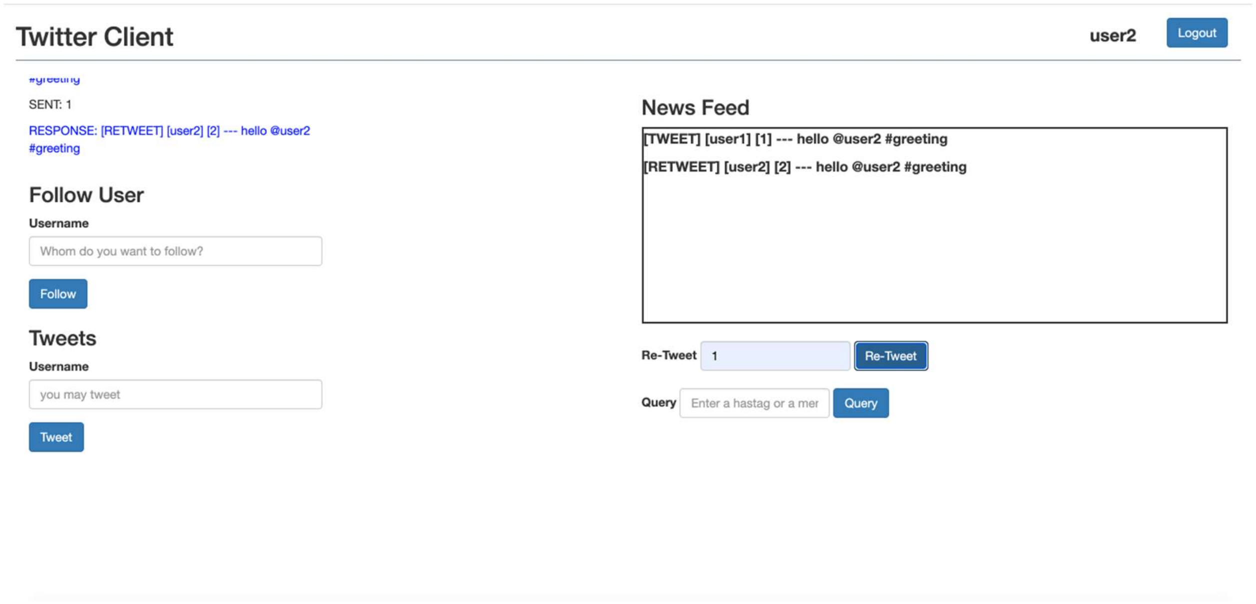
News Feed

[TWEET] [user1] [1] --- hello @user2 #greeting

Re-Tweet | Enter the Tweet ID | Re-Tweet

Query | Enter a hastag or a mer | Query

User2 retweets User1’s tweet –



Twitter Client

user2

Logout

#greeting

SENT: 1

RESPONSE: [RETWEET] [user2] [2] --- hello @user2 #greeting

Follow User

Username

Whom do you want to follow?

Follow

Tweets

Username

you may tweet

Tweet

News Feed

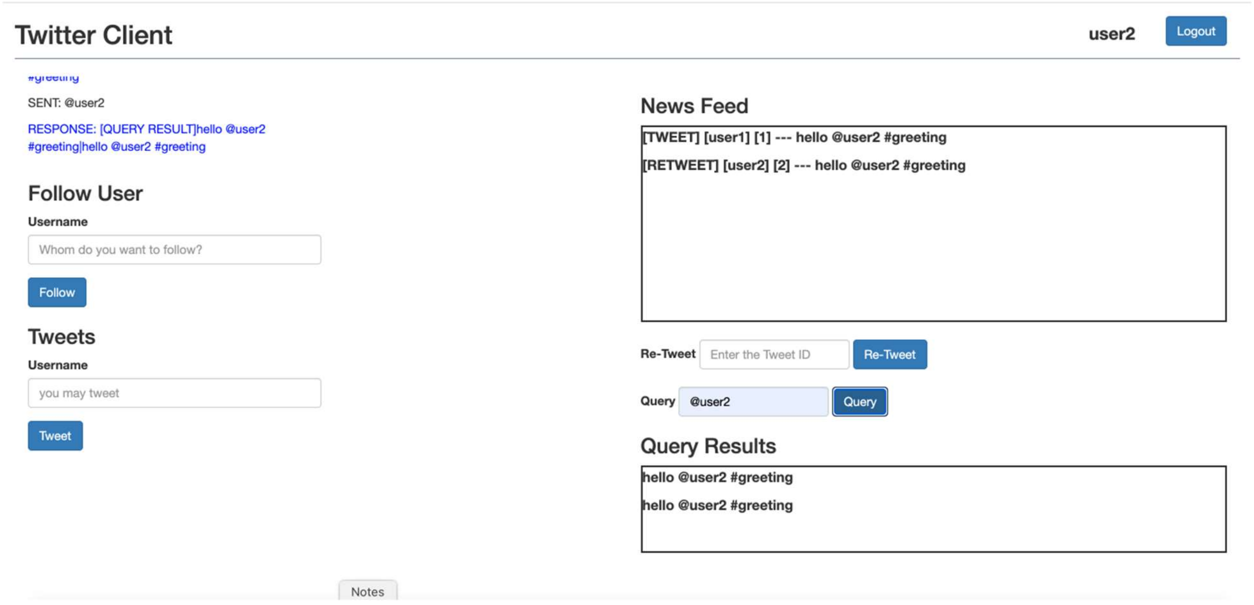
[TWEET] [user1] [1] --- hello @user2 #greeting

[RETWEET] [user2] [2] --- hello @user2 #greeting

Re-Tweet | 1 | Re-Tweet

Query | Enter a hastag or a mer | Query

User2 mention query with tag "@user2" –



Twitter Client user2 Logout

```
SENT: @user2
RESPONSE: [QUERY RESULT]hello @user2
#greeting|hello @user2 #greeting
```

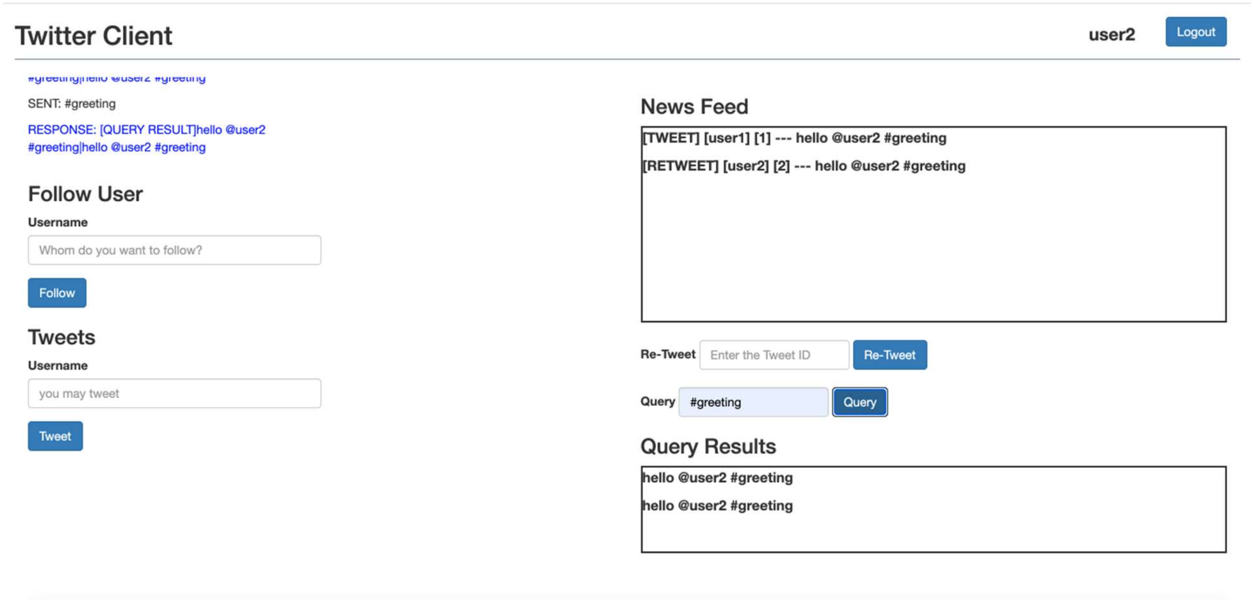
News Feed:

```
[TWEET] [user1] [1] --- hello @user2 #greeting
[RETWEET] [user2] [2] --- hello @user2 #greeting
```

Query Results:

```
hello @user2 #greeting
hello @user2 #greeting
```

User2 hashtag query with tag "#greeting" –



Twitter Client user2 Logout

```
SENT: #greeting
RESPONSE: [QUERY RESULT]hello @user2
#greeting|hello @user2 #greeting
```

News Feed:

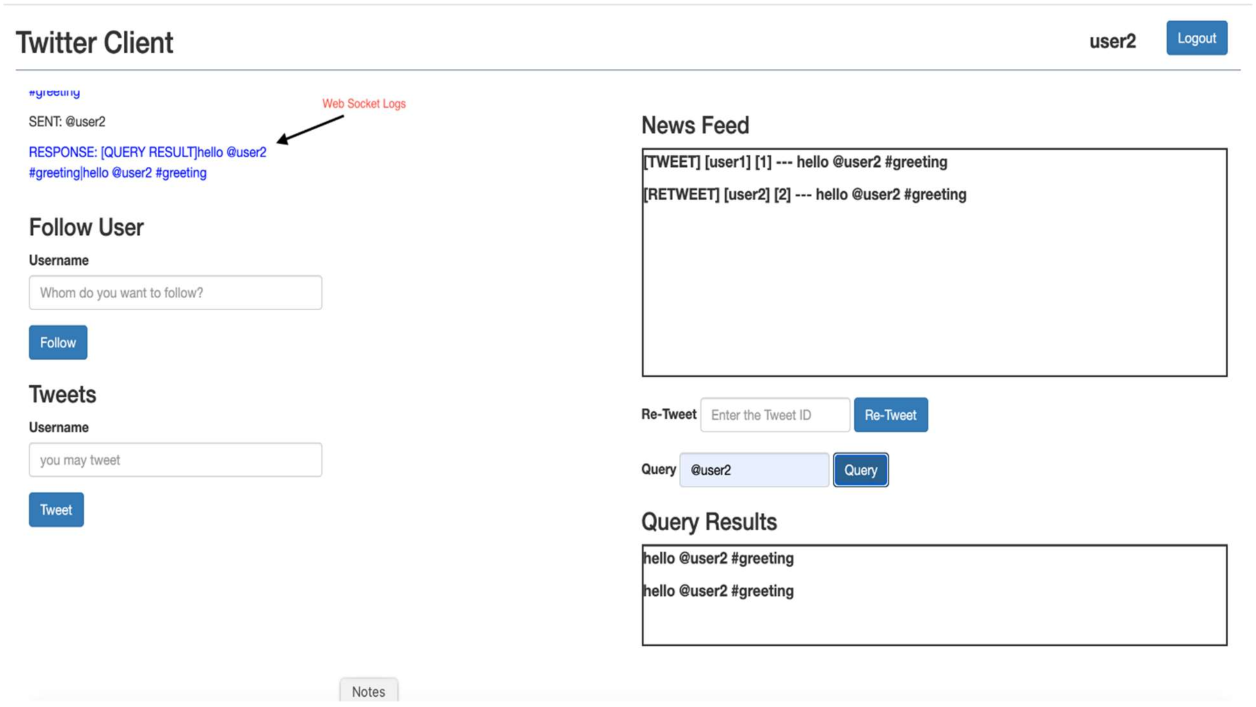
```
[TWEET] [user1] [1] --- hello @user2 #greeting
[RETWEET] [user2] [2] --- hello @user2 #greeting
```

Query Results:

```
hello @user2 #greeting
hello @user2 #greeting
```

We are printing the twitter user's Web Socket logs on the UI just to help the user to see all his twitter activity in a particular session (From the time a user logs in till the time they logout)

All the **Error Logs and Error Messages** are printed in this Web Socket logs view.



```
#greeting
SENT: @user2
RESPONSE: [QUERY RESULT]hello @user2
#greeting|hello @user2 #greeting
```

News Feed

[TWEET] [user1] [1] --- hello @user2 #greeting

[RETWEET] [user2] [2] --- hello @user2 #greeting

Query: @user2

Query Results

hello @user2 #greeting

hello @user2 #greeting

All the users will get their updated news feeds when they logout and log back in again (Think of a scenario where, user2 logs out and user1 tweets, then again when user2 logs back in, user2 will have all their news feed tweets along with the new tweet that user1 tweeted when the user2 is inactive).

All the scenarios mentioned above are shown in the demo video.