Project 4.2 – Twitter Clone using Web

COP5615 - Distributed Operating System Principles

Goal:

To implement a Twitter Clone in Erlang and implement the basic functionalities in Twitter using multiple clients using WebSocket interface.

Project Members:

- Ratna Prabha Bhairagond (UFID – 8827 4983)
- Varad Rajeev Sanpurkar (UFID – 1782 9883)

Requirements:

- Latest version of Erlang
- Multicore terminals on the same machine

Video Link:

DemoLink.mp4

Steps for compiling and running the code:

To run the server:

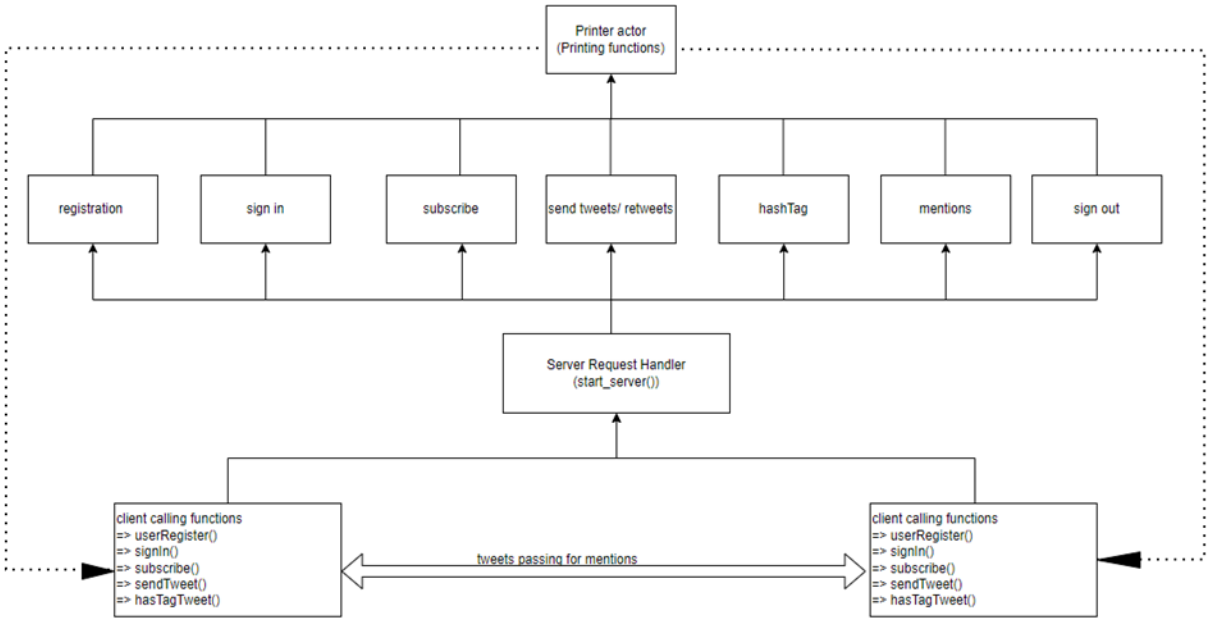
Open a terminal in the correct directory and execute the below commands:

- erl -sname server
- c(startmodule).
- c(userregistration).
- c(tweetspassing).
- startmodule:start_server().

To run the clients:

- erl -sname client1
- c(startmodule).
- c(userregistration).
- c(tweetspassing).
- startmodule:userRegister(). (To register the user for the first time)
- startmodule:signIn(). (To sign In the user for the first time)
- startmodule:subscribe(). (To subscribe to the user)
- startmodule:sendTweet(). (To send the tweet to the user)
- startmodule:signInOut(). (To sign out the user)

Architecture:



Flow chart of Twitter server engine and client simuation

Program flow:

1. The method start:start_server() will start the server where the device name is fetched using inet:gethostname(). The server starts listening on port 4000 for client messages. Then the hostname is concatenated to the hardcoded string “server@”. This name is registered while we register the server in the first command on the server and client machines. In this method, we initiated the actors required for all the processes to which a client will have access.
2. To show the simulation of multiple clients, respective methods are created for functionalities like user registration, sign-in, subscribe, sending tweets, etc. in the startmodule. We create client actors which send message to server on the 400 The functionality of these modules is defined in the userregistration.erl file and tweetspassing.erl files. Userregistration.erl module handles the logic related to the signing In, signing out, registration of users, and subscribing to users, while message passing logic is written in the tweetspassing.erl module.
3. While registering the user, input is accepted from the command line in the variables Username and Password using io:fread(). The entered username and password are stored in the **Maps** in erlang. If user is not present in the map then

the new entry is added in the map with its process ID and password. Whereas, while signing in the user, the password is matched to the respective entered username in this Map.

4. For subscribe functionality, another **Map** (subscribersMap) is maintained which contains the list of the subscribers for the given user. All the subcribed users are able to recieve the tweets posted by the user to which users have subscribed. All the users need to be signed to receive the posted tweets. The user which is not subscribed and mentioned using "@" in any tweet also receives the tweets successfully. Similar functionality is used while handling the hashTag (#) functionality of twitter.
5. Finally, signout functionality is written in the "signInOut" function. In this function, the username of the user is deleted from the persistent term as well as map.

What is working:

This program is executes a Twitter server and simulation of multiple clients over multiple terminals. **Twitter engine** registers the processes/ actors required for all the processes to be executed by all the users. Server mainly supports functions like **Registering new user, sign in of existing users, subscribing to users or hashtags, sending tweets, mentioning another user using @, searching using particular name or hashtag and signing out of users.** The same actor is used to serve individual process so that congestion is avoided between the users.

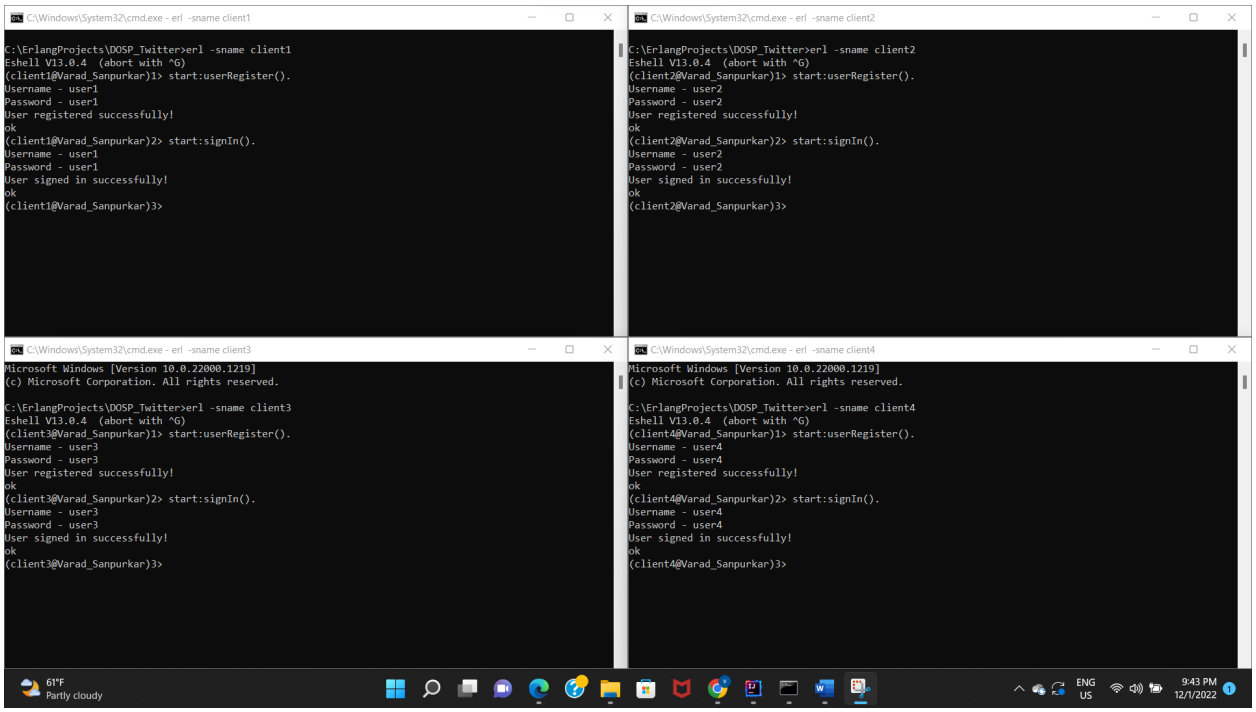
WebSocket Handler:

Installed and made use of Cowboy to implement a WebSocket interface. At server we have created a web socket with port 4000, which listens at port 4000. Similarly at client, we created a webSocket, which then sends data to Server at port 4000. Firstly, we use handshaking method to create a connection between the server and the clients. We have used AKKA messaging to exchange information between clients and servers. The Server and Clients acts as a JSON based API. Where clients sends to messages to server in JSON format and also gets JSON messages from the server.

Program Output:

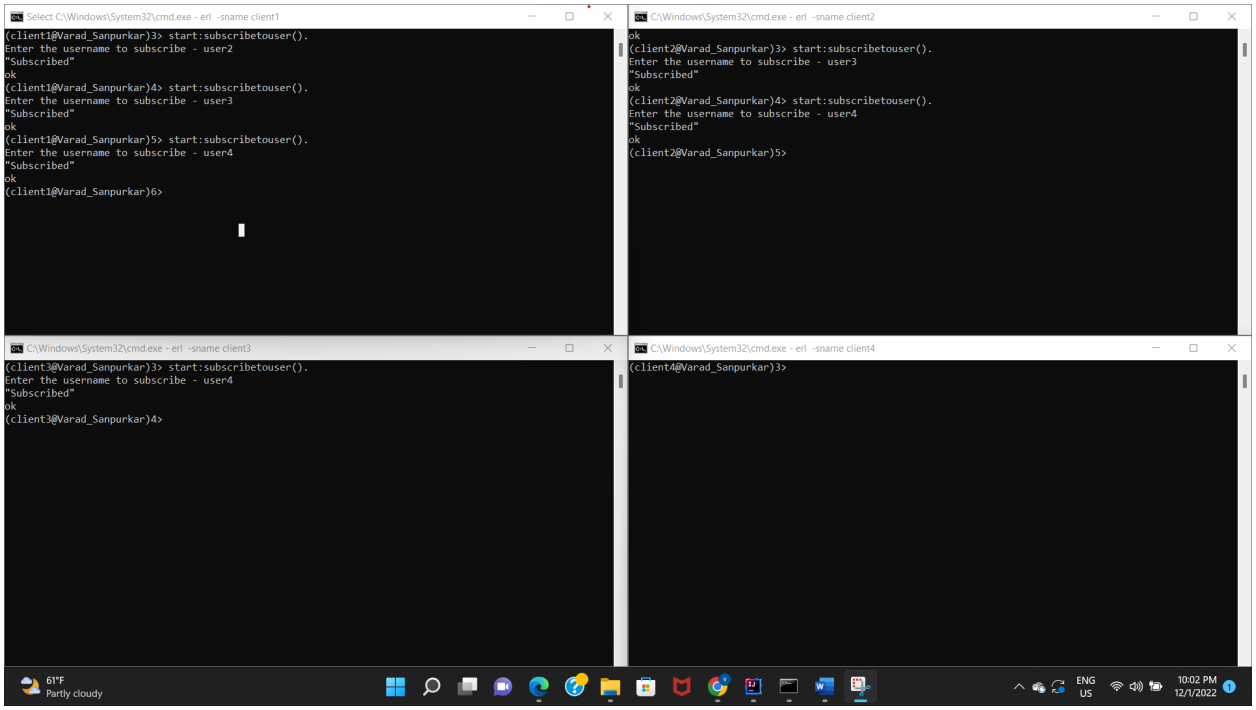
• Registration and sign in

Registers and singin using username and password of a user.



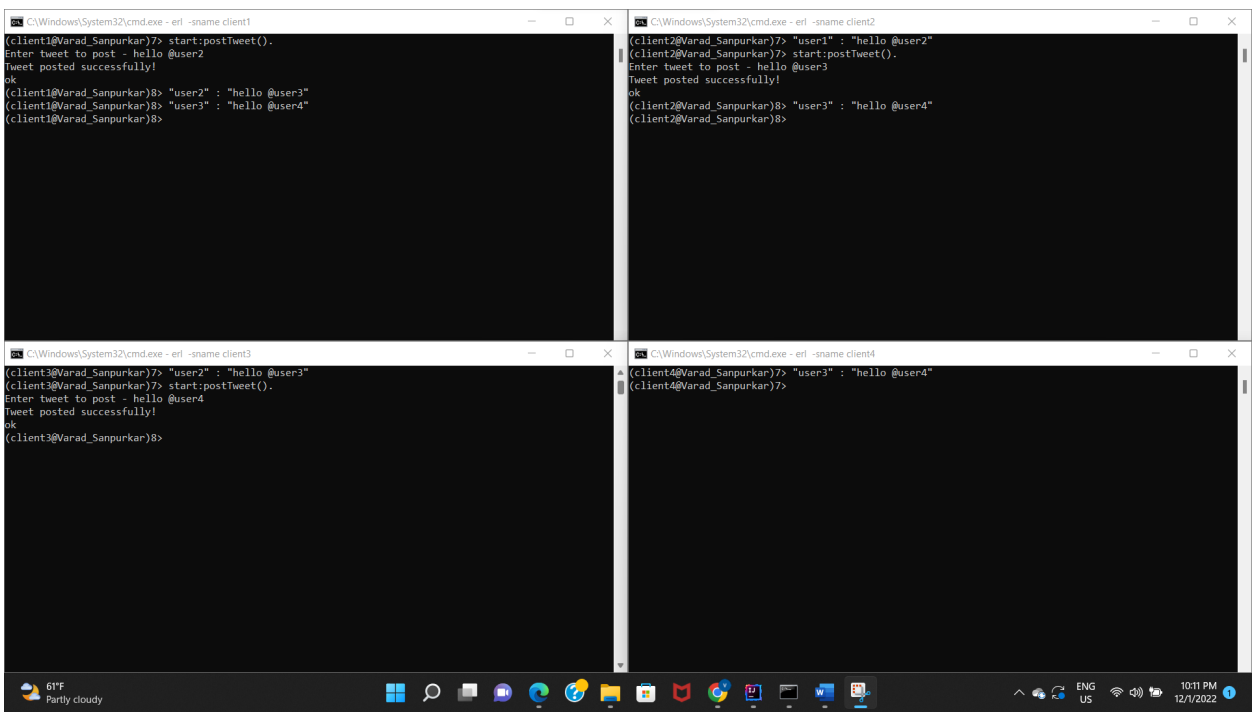
• Subscribe

Subscriber a particular user, to get tweets from them.



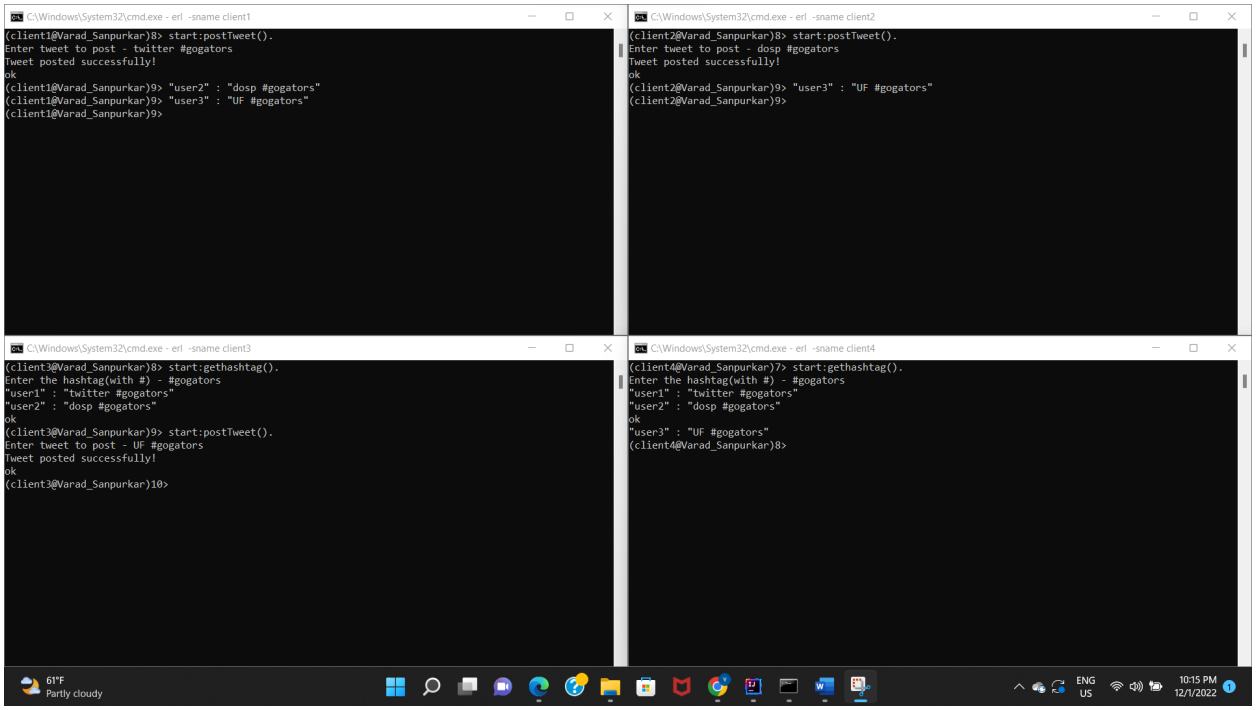
• Mentions using @

Can mention any user, subscribed or unsubscribe to in there tweet. And it will be visible to the user mentioned.



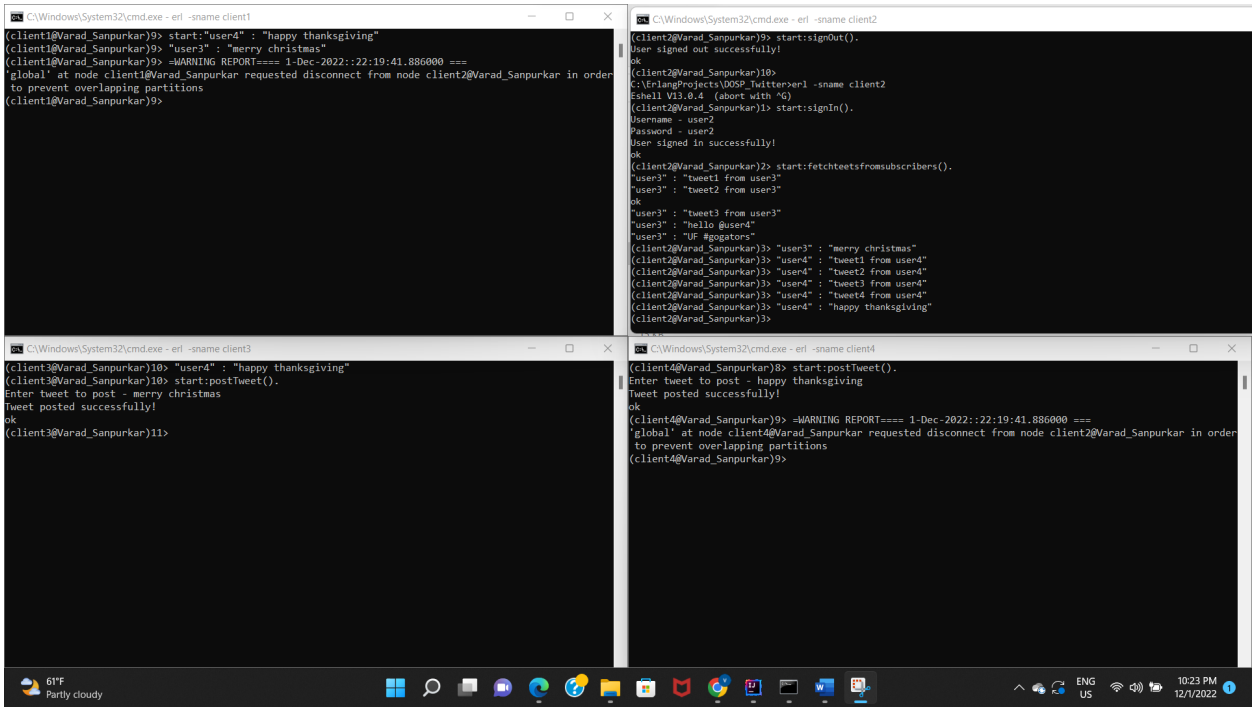
• Querying using hashtags

Displays the lists of tweets conating a particular hashtag



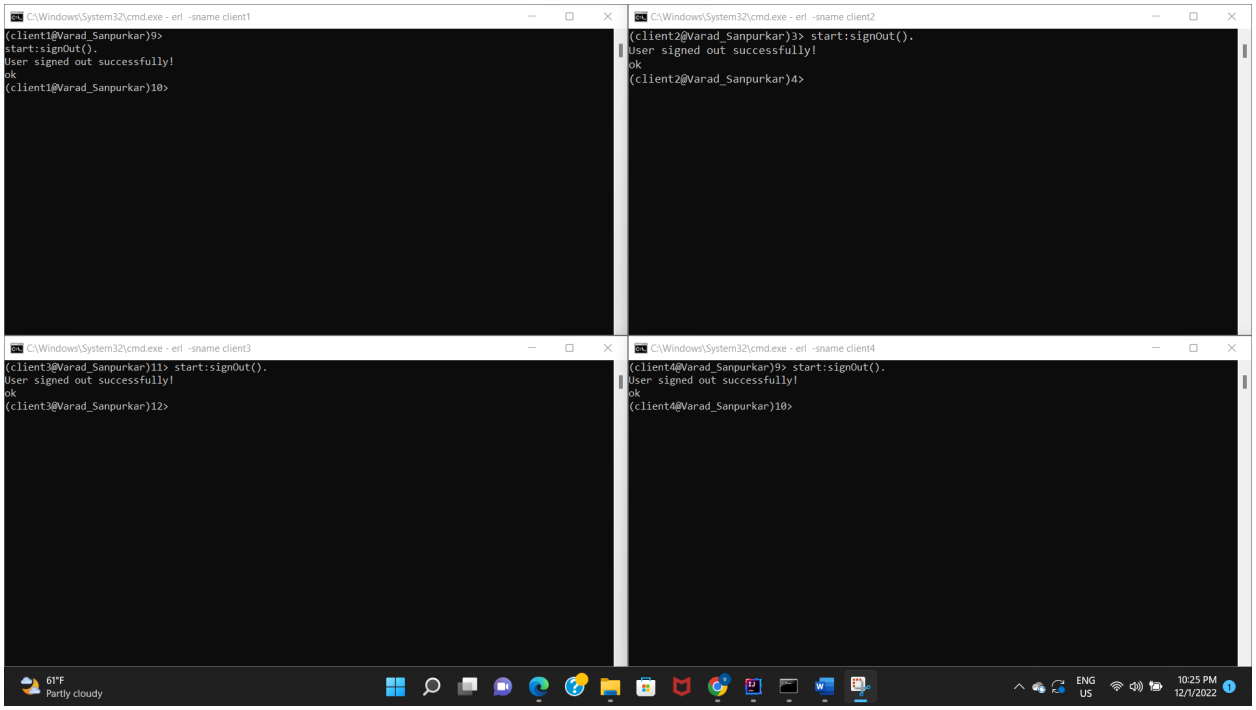
• Display subscribed tweets to the disconnected clients

Displace tweets to the clients that have been offline using fetchtweetsfromsubscribers()



• Sign out

Signes out the user.


C:\Windows\System32\cmd.exe - erl -sname client1
(client1@Varad_Sanpurkar)9>
start:signOut().
User signed out successfully!
ok
(client1@Varad_Sanpurkar)10>
C:\Windows\System32\cmd.exe - erl -sname client2
(client2@Varad_Sanpurkar)3> start:signOut().
User signed out successfully!
ok
(client2@Varad_Sanpurkar)4>
C:\Windows\System32\cmd.exe - erl -sname client3
(client3@Varad_Sanpurkar)11> start:signOut().
User signed out successfully!
ok
(client3@Varad_Sanpurkar)12>
C:\Windows\System32\cmd.exe - erl -sname client4
(client4@Varad_Sanpurkar)9> start:signOut().
User signed out successfully!
ok
(client4@Varad_Sanpurkar)10>
61°F
Partly cloudy
ENG
US
10:25 PM
12/1/2022