PROJECT 4 PART II

Goal of the Project:

The Twitter API Websocket is a server that uses a Web socket interface to communicate with a database of tweets, retweets, hashtags, and mentions. The server distributes tweets and enables users to register accounts, create tweets, and add hashtags and mentions. Akka actor nodes facilitate communication between the server and the Web socket, and the client testing is done through a browser interface.

Contribution:

Nikhil Yerra, UFID: 95453265

Ujwala Guttikonda, UFID: 57914323

Steps to execute the code:

1. we need to extract the files from the folder project4.zip
2. we have to use the command to run it: 'make' in the terminal which will compile the Nitrogen server.
3. Execute the below command

   Make app PROJECT = project4

   The above command will establish the twitter project
4. For starting the server, use the command

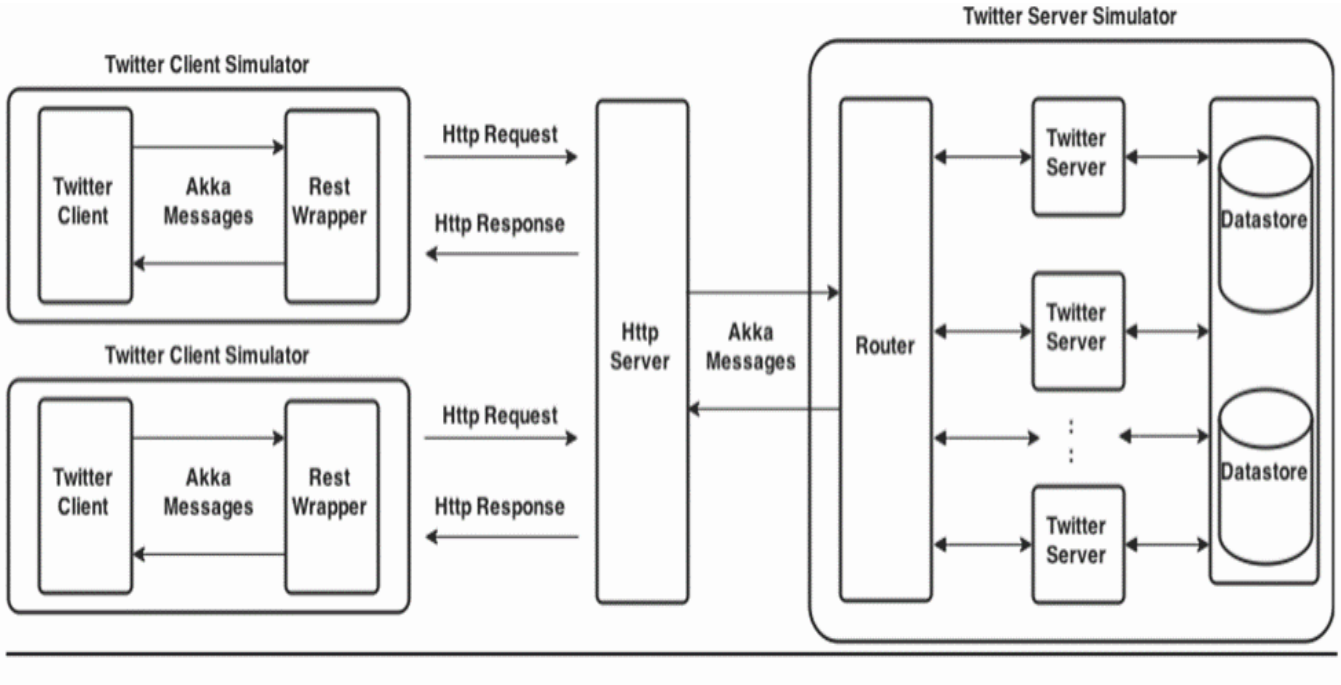
   ./init-server.sh
5. Next, run the terminal and run the below command

   twitterserver:start().
6. Parallely open one more terminal and start the client by running – twitterclient:start().

   Now you can see the twitter app running in the browser.

Architecture:

The architecture we designed for building the Twitter is as follows:



The twitter architecture above depicts client simulator, http server and server simulator as the main basic blocks. The requests and responses are being handled to and fro from the client and server. Router block from the server simulator helps in communicating the messages received from the client through http server and are communicated with the database (datastore).

Implementation:

The web user interface has been created using HTML, JavaScript, with frameworks like Bootstrap and JQuery. Project 4.1's server and backend are being linked with the user interface using Phoenix Channels. The project, which also includes project 4.1 and the new web interface's source code, is an umbrella undertaking.

When the Phoenix Server is started, the backend server—which contains the application logic and in-memory data—is also started. All of the tables are set up in this way. The web interface

application is then launched; to examine it, open your web browser and navigate to localhost:8080.

To access their dashboard on the Twitter API WebSocket, the user can register and log in. They can send new tweets, retweet tweets from other users, and follow other users all from the dashboard. Additionally, they may use hashtags to search for tweets and display the results in a table on the sidebar.

WebSocket: A server will establish a new, two-way communication with a client in a web - based application utilising WebSockets. As an outcome, the client doesn't need to frequently query the server for recent information in order for the server to update the client.

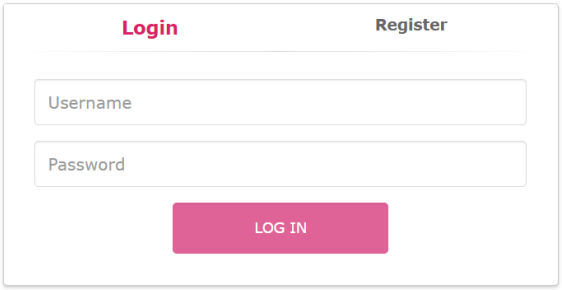
- The first step, when the client wants to connect is to login to a twitter server it will creat a WebSocket connection from the client side with a name /websocket URL.
- Next when the twitter server receives a request from the client, handshake messages will be interchanged between the client and server. i.e the client will be sending a username through the WebSocket we used to the twitter server. The username client going to be sent will be in the format of UserName:username

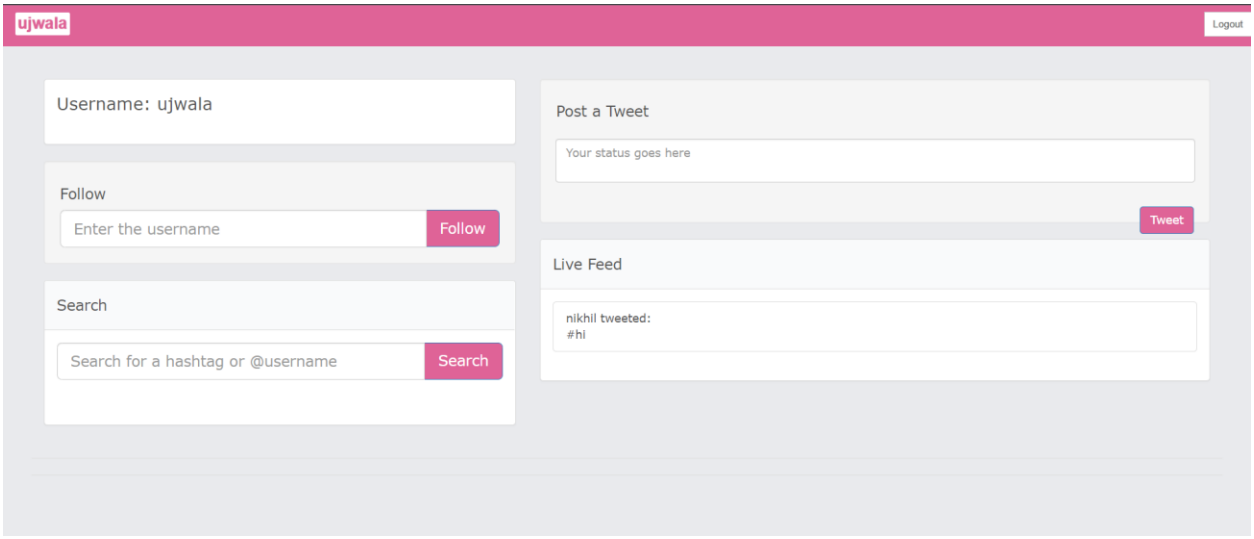
- We used unique client IDs to store the messages which were being sent by the client. i.e the server is going to keep the websocket with a unique client id.

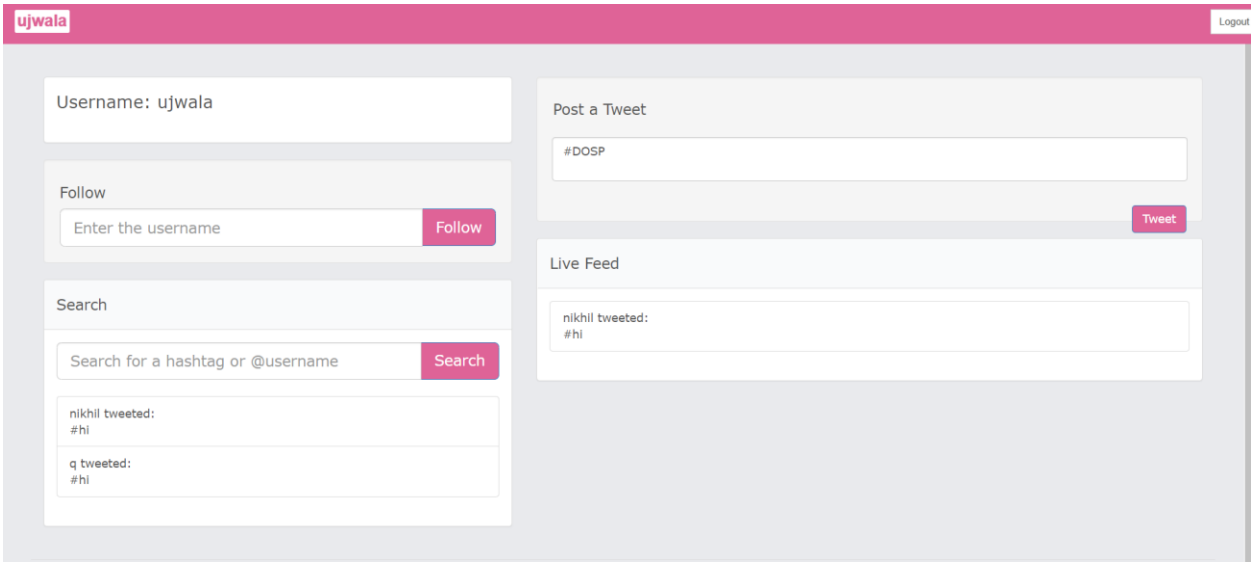
- The actor chooses the appropriate WebSocket address and posts the message to the client on WebSocket whenever it has to make a news update towards the client.
- When the server will send a message to the client through the WebSocket, then it will present or display the live Twitter page. i.e. on the feed.

Results:

1. We show the user's tweets, success messages, and error messages on the application's screen.
2. For access to other users' tweets, users can subscribe to them.
3. Users who have subscribed will send tweets to their followers.







YouTube Link:

https://youtu.be/VF5sP4XOC_s