Distributed Operating Systems Principles - Twitter Clone Part II

Team Members:

- Simran Sunil Kukreja (UFID: 72070369, s.kukreja@ufl.edu)
- Rachana Nitinkumar Gugale (UFID: 63532454, rgugale@ufl.edu)

Execution Steps:

- Navigate to the program folder i.e., project4
- Use the following commands to run the Twitter engine:
  - Navigate to the src folder i.e., cd websocket/src
  - Start the repository first in one terminal:
    - repository: start().
  - Run the make file in another terminal from the project4 folder:
    - Navigate to the websocket folder i.e., cd websocket
    - make run
  - Launch http://localhost:8080 in the browser (launch multiple such windows to execute the Twitter engine for multiple clients.

Project Video

Implementation details:

The aim of our project is to use an Erlang web framework that supports a WebSocket implementation for communication between the client and the server.

Since the cowboy server includes routing features for distributing client requests to handlers written in Erlang, we used it for our project. Moreover, It also functions as a complete HTTP server and supports REST.

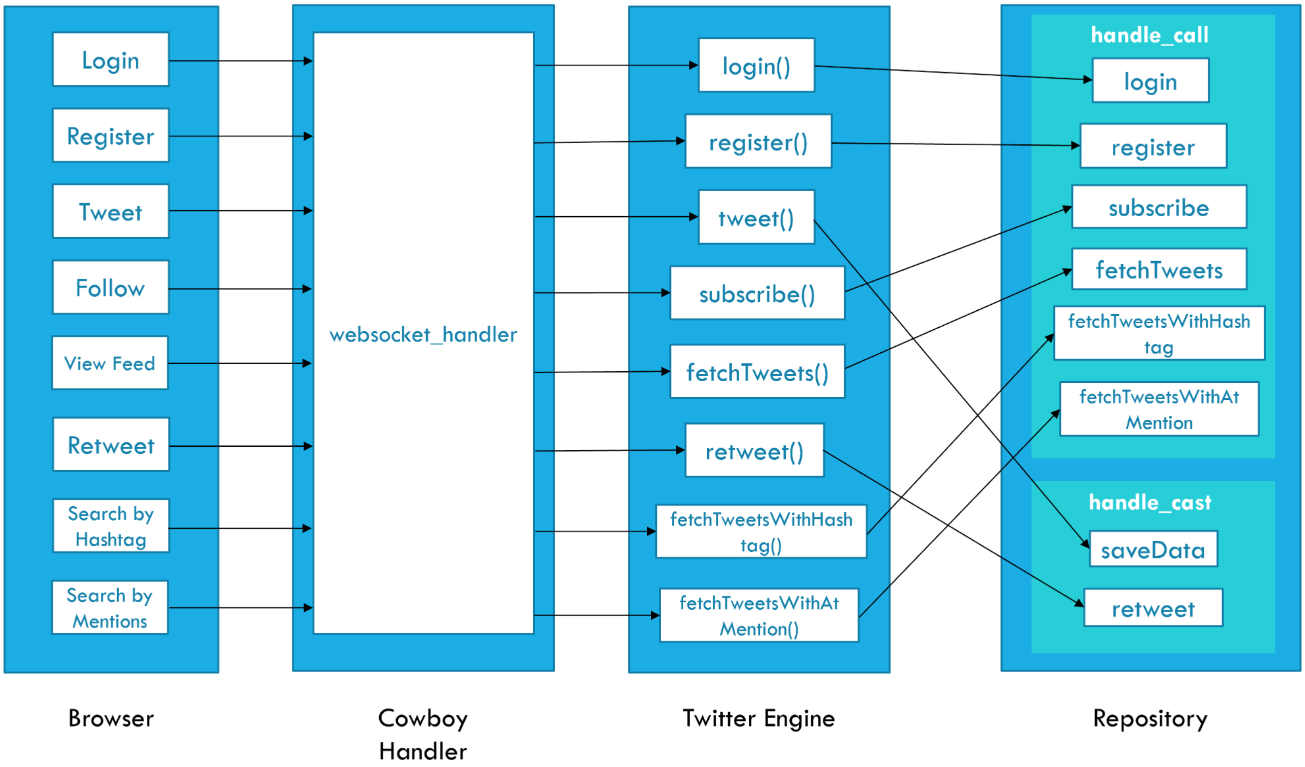
Listing the core functionalities we have implemented:

- Using cowboy, we implemented the WebSocket interface and adapted our twitter_engine.erl and repository.erl code accordingly
- We have also created a frontend for the client, which communicates with the engine by utilizing the WebSocket.

- The client can register/ login into the Twitter clone using the interface launched on localhost port 8080. Upon registering or logging in with the correct username and password, the client can perform all the Twitter engine functionalities.
- As a part of the Twitter engine functionalities, we are allowing the client to:
  - Follow another user by specifying their username
  - Tweet by entering the tweet message
  - Retweet a tweet made by themselves or another user
  - View feed is used by the user to view tweets made by the users that they are following
  - Search tweets using hashtag mentioned in the tweet
  - Search tweets using @mention mentioned in the tweet
  - Logout

Additionally, we have incorporated error handling, making sure the same user cannot register twice i.e., for a registered username, we ask the user to login since they are already registered. Similarly, if a user enters an incorrect username/password then we are showing them the prompt to login again with the correct username/password.

Architecture:



- As shown in the architectural diagram above, the client sends a websocket request through the browser UI to perform one of the following twitter engine operations: login, logout, register, tweet, view feed, retweet, search tweets by hashtag, search tweets by mentions.

- This request is then received on the backend by the cowboy's websocket_handler() method. We have implemented a switch case in this handler method so that the incoming request is routed to the right gen_server method for further processing.

- Depending on the action the user wants to perform, the websocket_handler() method forwards the request to the appropriate twitter_engine method.

- The twitter_engine then creates an actor for each of the clients and depending on the action the client wants to perform, calls the corresponding repository method.

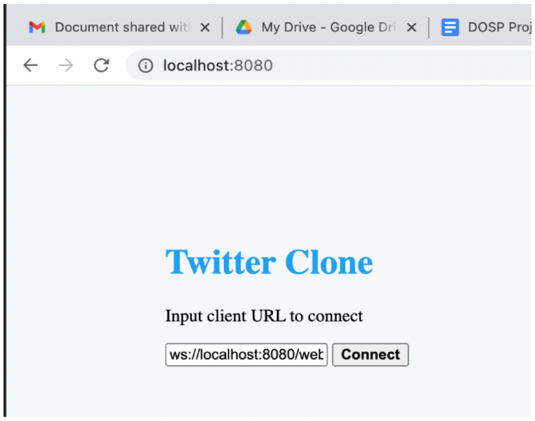
- The repository implements gen_server behavior which makes it easy to maintain state tables across requests.

- For the methods in the repository, we have used either handle_call or handle_cast depending on whether a the method returns a value or not.

- The result of the client request generated by the repository method then goes to the twitter engine, the cowboy handler and is finally displayed on the browser.

Execution Results:

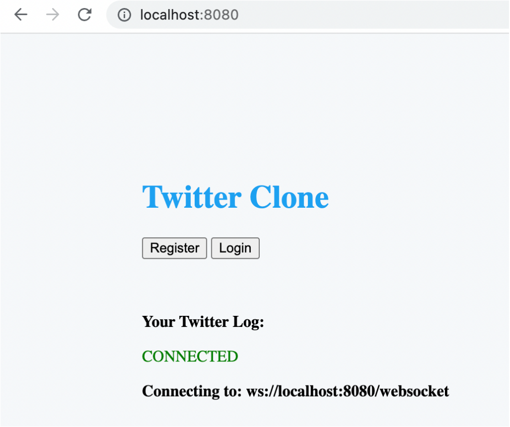
1. Starting the client



Twitter Clone

Input client URL to connect

ws://localhost:8080/wet



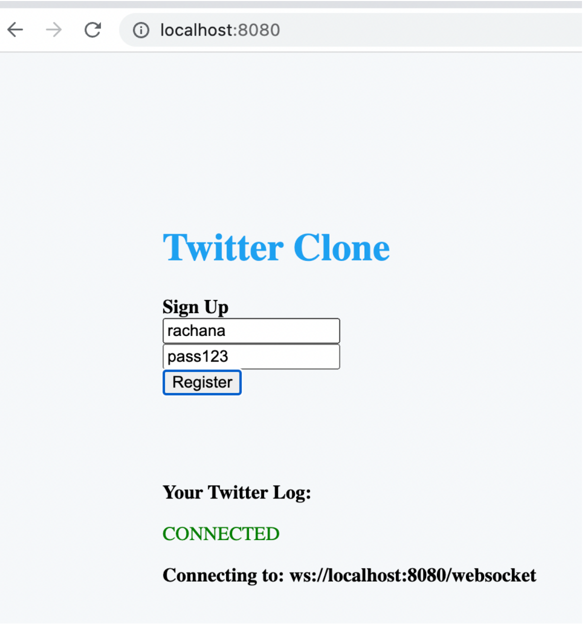
Twitter Clone

Your Twitter Log:

CONNECTED

Connecting to: ws://localhost:8080/websocket

2. Register



Twitter Clone

Sign Up

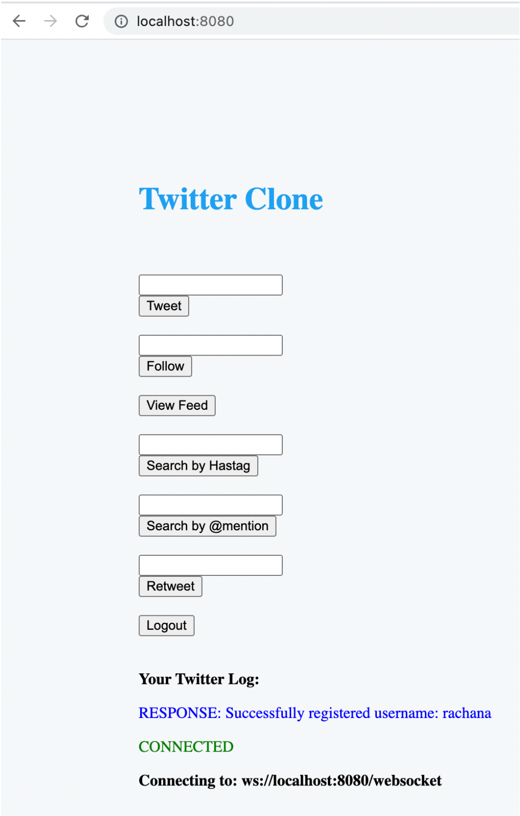
rachana

pass123

Your Twitter Log:

CONNECTED

Connecting to: ws://localhost:8080/websocket



Twitter Clone

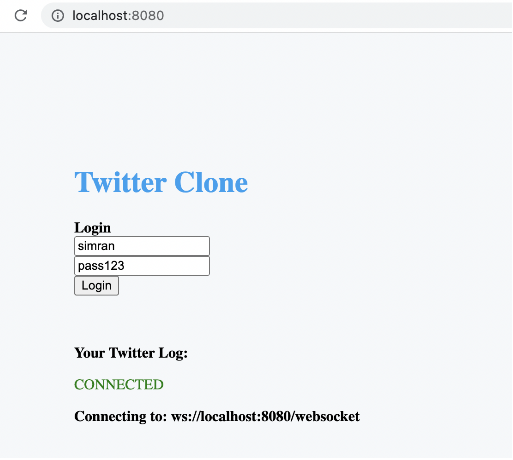
Your Twitter Log:

RESPONSE: Successfully registered username: rachana

CONNECTED

Connecting to: ws://localhost:8080/websocket

3. Login



Twitter Clone

Login

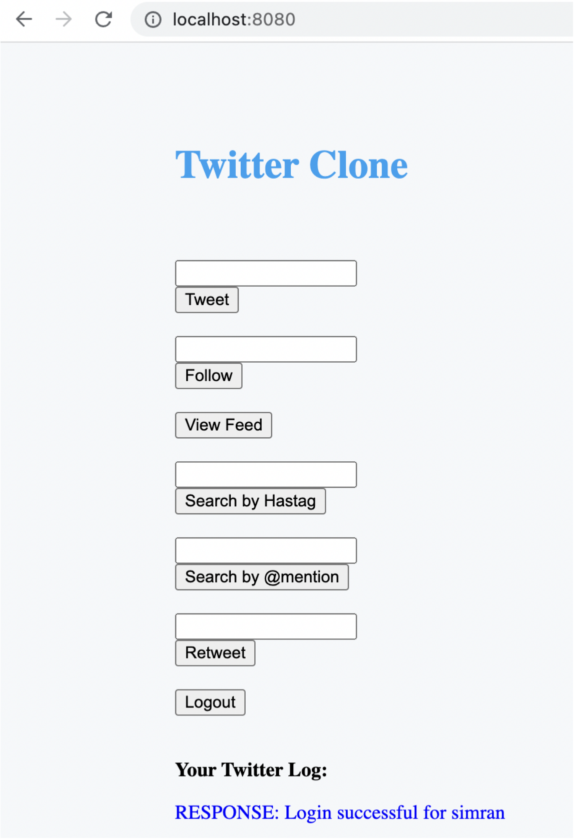
simran

pass123

Your Twitter Log:

CONNECTED

Connecting to: ws://localhost:8080/websocket

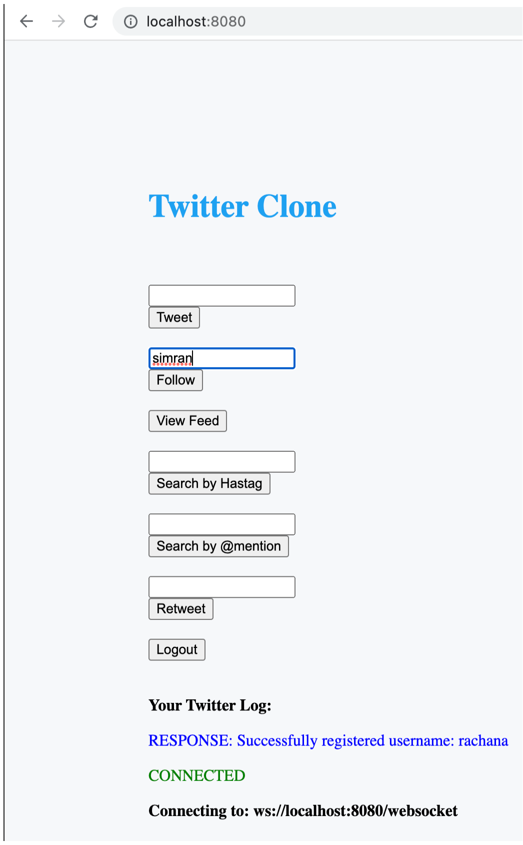


Twitter Clone

Your Twitter Log:

RESPONSE: Login successful for simran

4. Follow

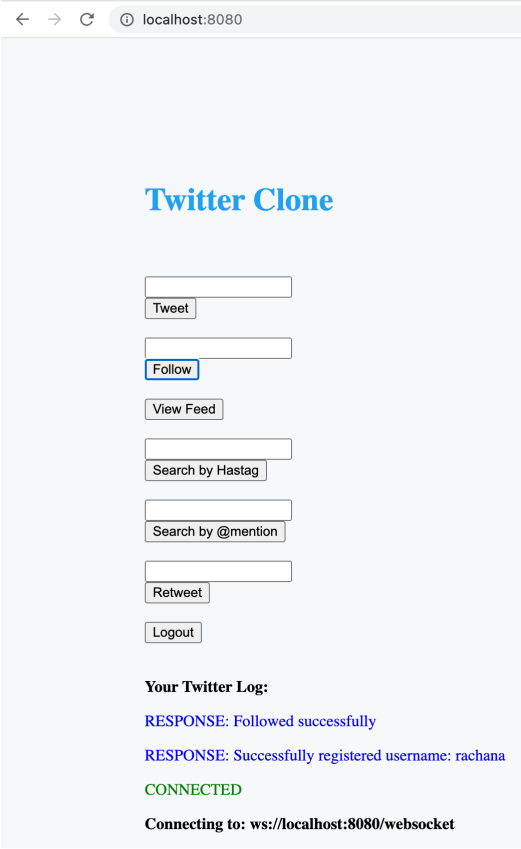


Your Twitter Log:

RESPONSE: Successfully registered username: rachana

CONNECTED

Connecting to: ws://localhost:8080/websocket



Your Twitter Log:

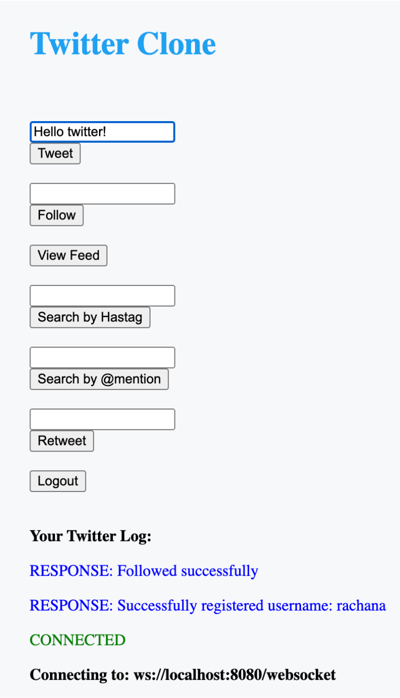
RESPONSE: Followed successfully

RESPONSE: Successfully registered username: rachana

CONNECTED

Connecting to: ws://localhost:8080/websocket

5. Tweet



Twitter Clone

Hello twitter!

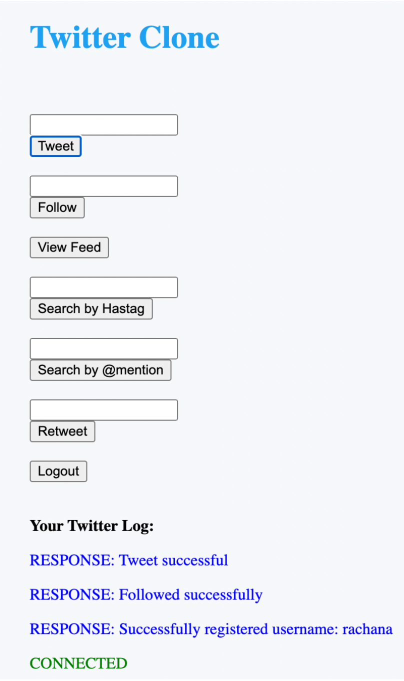
Your Twitter Log:

RESPONSE: Followed successfully

RESPONSE: Successfully registered username: rachana

CONNECTED

Connecting to: ws://localhost:8080/websocket



Twitter Clone

Your Twitter Log:

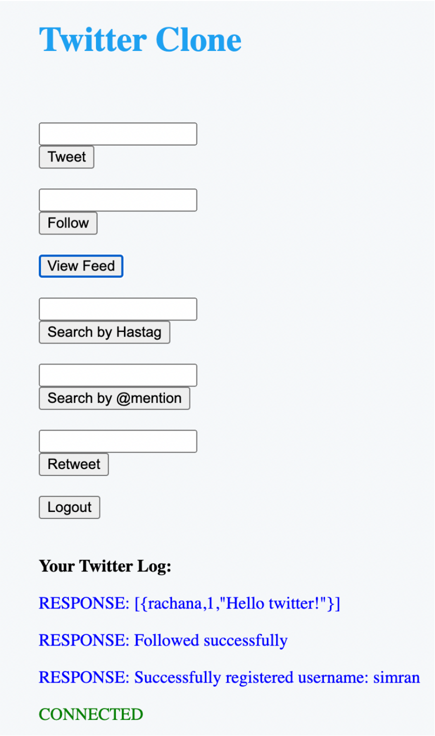
RESPONSE: Tweet successful

RESPONSE: Followed successfully

RESPONSE: Successfully registered username: rachana

CONNECTED

6. View Feed



Twitter Clone

Your Twitter Log:

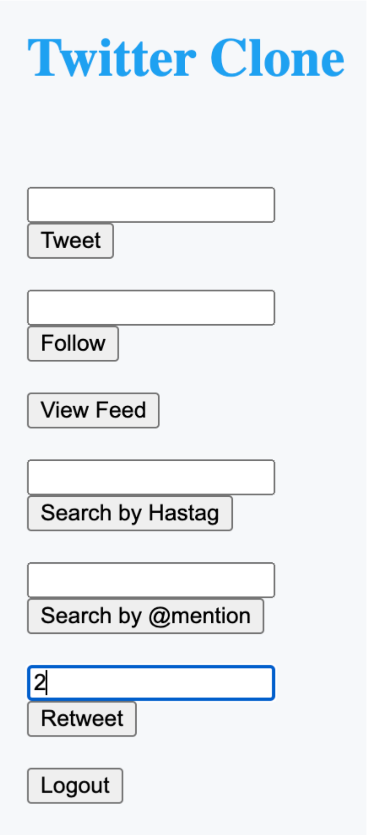
RESPONSE: [{rachana,1,"Hello twitter!"}]

RESPONSE: Followed successfully

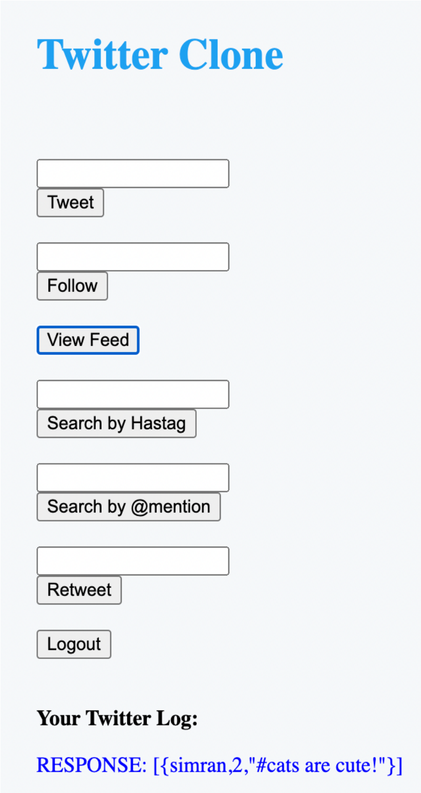
RESPONSE: Successfully registered username: simran

CONNECTED

7. Retweet



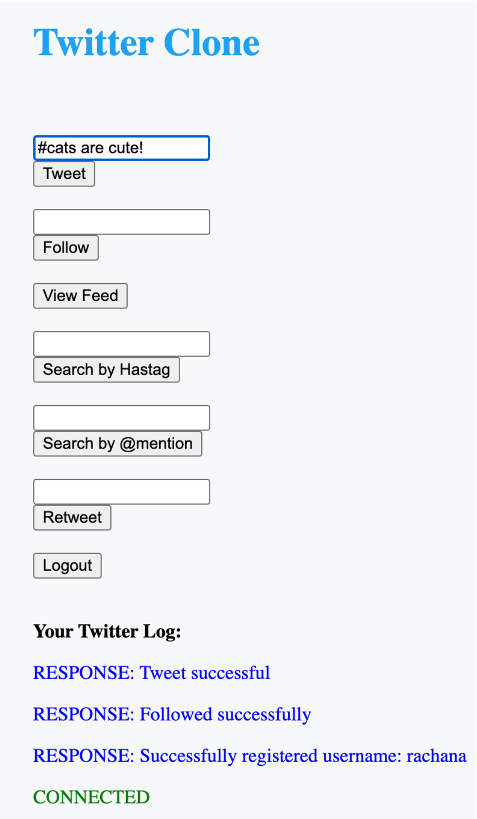
2



Your Twitter Log:

```
RESPONSE: [{simran,2,"#cats are cute!"}]
```

8. **Search by hashtag**



Twitter Clone

#cats are cute!

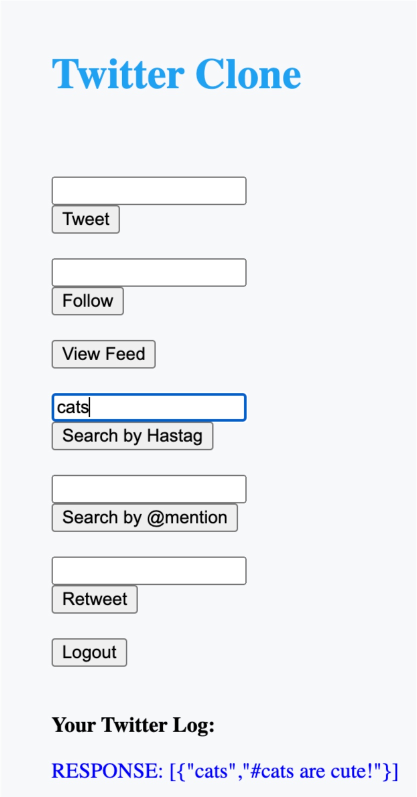
Your Twitter Log:

RESPONSE: Tweet successful

RESPONSE: Followed successfully

RESPONSE: Successfully registered username: rachana

CONNECTED



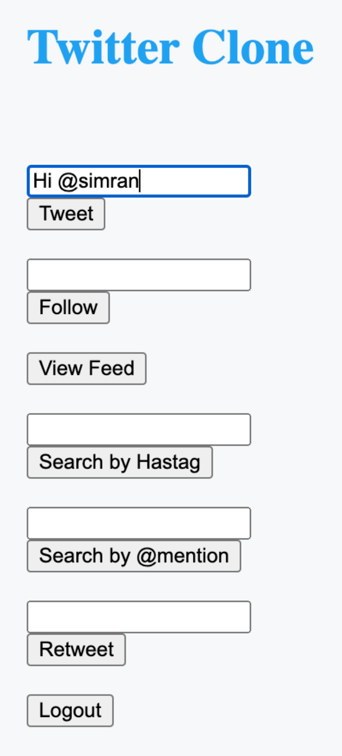
Twitter Clone

cats

Your Twitter Log:

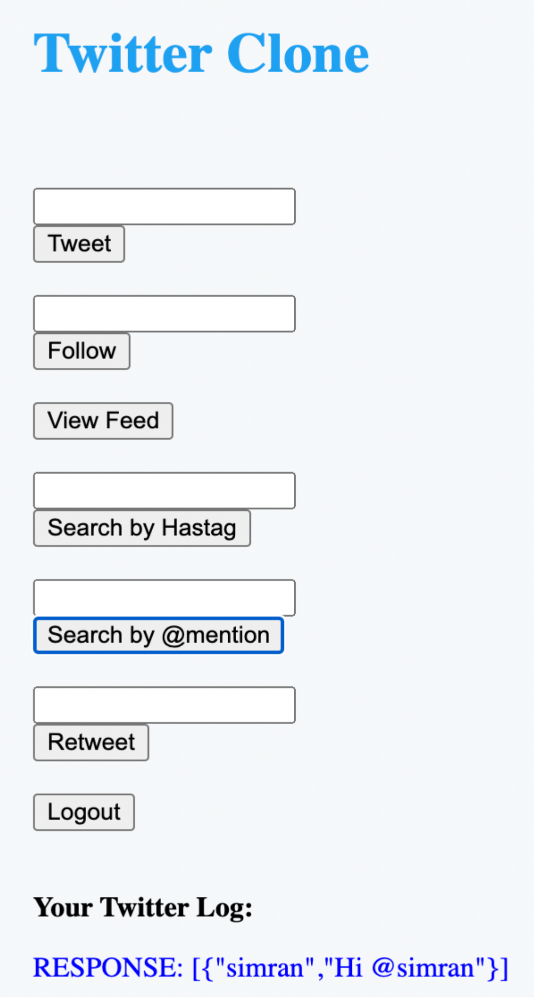
RESPONSE: [{"cats","#cats are cute!"}]

9. **Search by mention**



Twitter Clone

Hi @simran



Twitter Clone

Your Twitter Log:

RESPONSE: [{"simran","Hi @simran"}]