

## **DOSP (COP5615) Project 4**

Name: Srishti Jaiswal

UFID: 80385159

### **Description:**

This project aims to develop a client tester/simulator and a Twitter clone. A Twitter engine that allows multiple client connections has been built. I used the Cowboy Web Framework to create the use of web sockets, which lets users execute a variety of tasks including transmission and reception of tweets while the server handles receiving and sending.

### **The following features have been added to the Twitter engine (Server):**

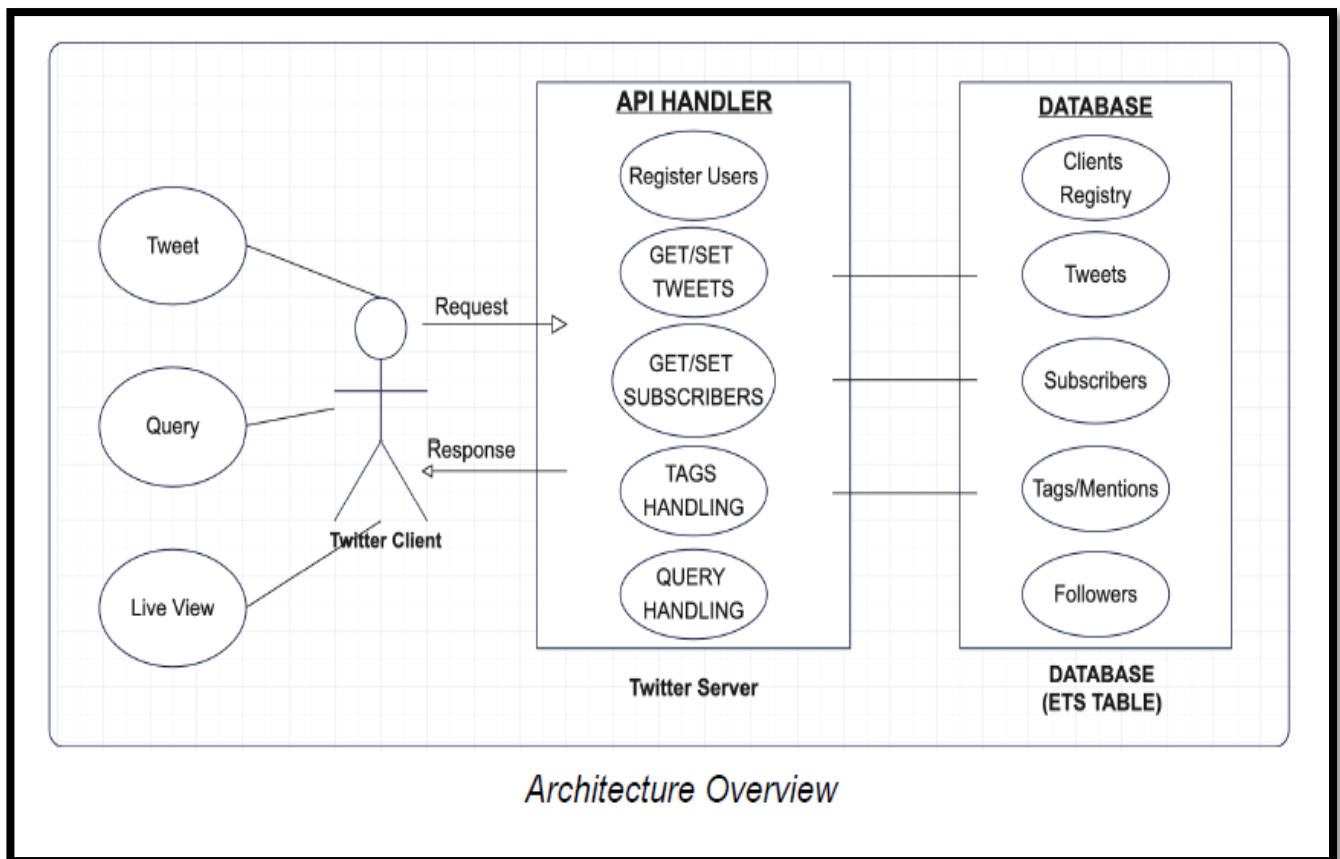
- Register User Account
- Send Tweet
- Tweets can include mentions (@username) and hashtags (for example, #whatsup).
- Subscribe to other user's tweets
- Retweet
- Enable browsing through tweets you've subscribed to, tweets using certain hashtags, and tweets that mention you

### **Usage Details:**

- Navigate to the project folder
- The Open command prompt, then type "make run" after selecting the Twitter website folder from the source folder.
- If issues are found, then type "make distclean" and again type "make run"
- Test the server using localhost:8080 after it has been launched.
- We can test all the features right here on the Twitter web client.

### **Architecture:**

In order to get data as needed and requested by the user, the Twitter Engine acts as a conduit between the client and the database. To execute the user's requests and activities, the engine manages a range of API calls. It also effectively utilizes databases (ETS tables) to deliver the desired outcomes. User registration, tweet processing, and query administration are all handled by the engine. A client's list of subscribers, followers, tweets, and mentions are all included in the database along with information about clients and users. Depending on the features the client requests, the client module is responsible for directing queries to the server to fulfill tasks. The engine provides a response to the client following each request.



## REST API

- Requests: We use an API request to send a JSON object from the server to the client.

A sample JSON Object is of the format:

```
{
  "username": "user",
  "password": "password"
}
```

A query by mentions would use the following format

```
{
  "tweet_no": "sample_tweet",
  "tweet": "@srishti srishti"
}
```

**GET, PUT, and POST are the three different request methods.**

GET - Used for reading or retrieving a resource

PUT - Used for modifying a resource

POST - Used for creating a new resource

### **All the files which have been used:**

- `twitter_client.erl` - The client makes queries to the server and manages various features, including tweeting, retweeting, and subscribing.
- `twitter.erl` - This is the Twitter engine or server that responds to client queries.
- Based on the file names, the following files are used to manage particular Twitter system capabilities:
  - `feed_handler.erl`
  - `register_handler.erl`
  - `retweet_handler.erl`
  - `search_hashtag_handler.erl`
  - `search_mention_handler.erl`
  - `search_subscribe_handler.erl`
  - `subscribe_handler.erl`
  - `Tweet_handler.erl`
- `Helper.erl`- This is used to manage many utility tasks, such as the production of random strings, etc.
- `twitter_app.erl` – This file takes care of hosting localhost and server applications
- `twitter_sup.erl` – Supervisor

### **Implementation Details:**

**twitter.erl:** This file contains the code for the Twitter engine implementation, which is in charge of handling and distributing tweets. The engine has a direct connection to the database to manage things like subscriptions, tweets, and searches. It also communicates directly with users in order to deliver the results of queries and distribute tweets to subscribers. The registry ETS table of the client keeps track of the statuses of several server actors.

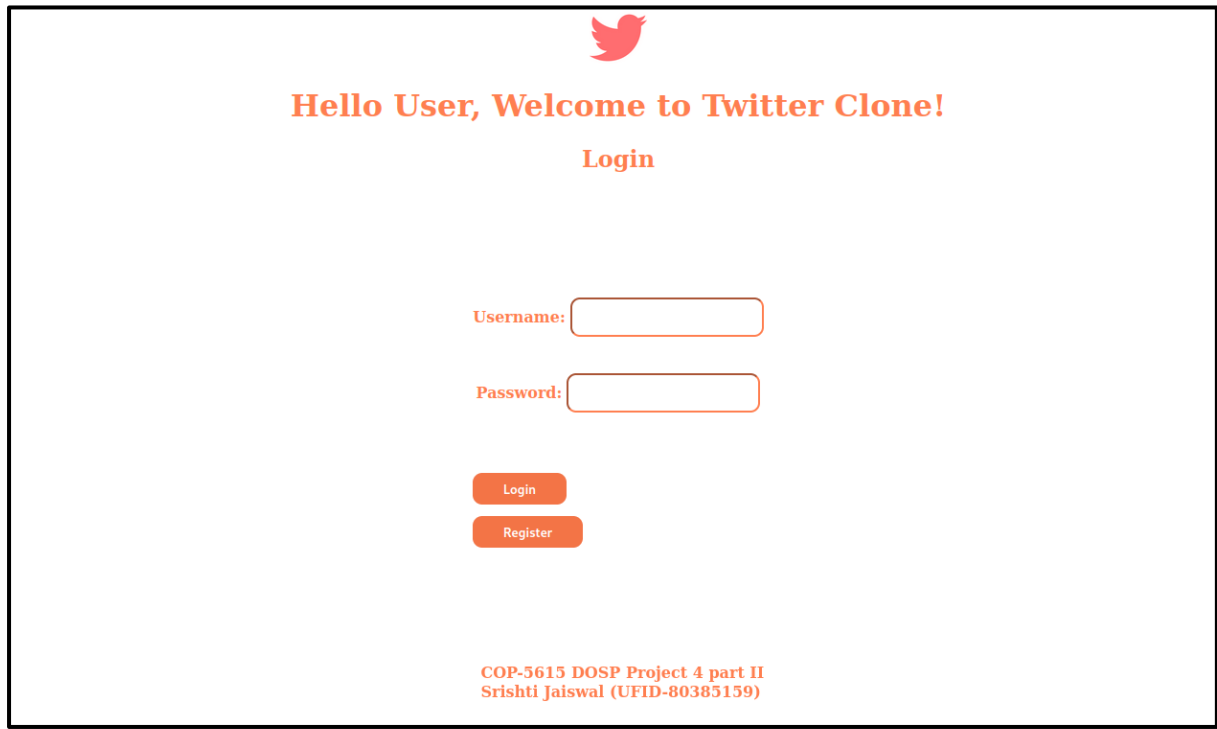
**client.erl:** This file includes information for a specific client participant, which is the same as a single Twitter user. This contains the information pertaining to its user ID, which is stored as a string given to it during the program's starting. The fundamental logic of maintaining the process state is carried out by the primary registry's ETS table, which keeps track of many actors.

**twitter\_app.erl:** This file provides a variety of application-related information. It includes a list of all the modules it uses, its name, description, and default configuration. This file begins a TCP-based listener for HTTP sessions. To listen for connections and map all URLs to our handlers, we pass a number of acceptors.

**twitter\_sup.erl:** All incoming connections are automatically started by this file, which also routes their requests to the handlers. On port 8080, the file opens a listening state. Finally, we must create a make file for the project named `erlang.mk` that specifies the dependence of the cowboy framework.

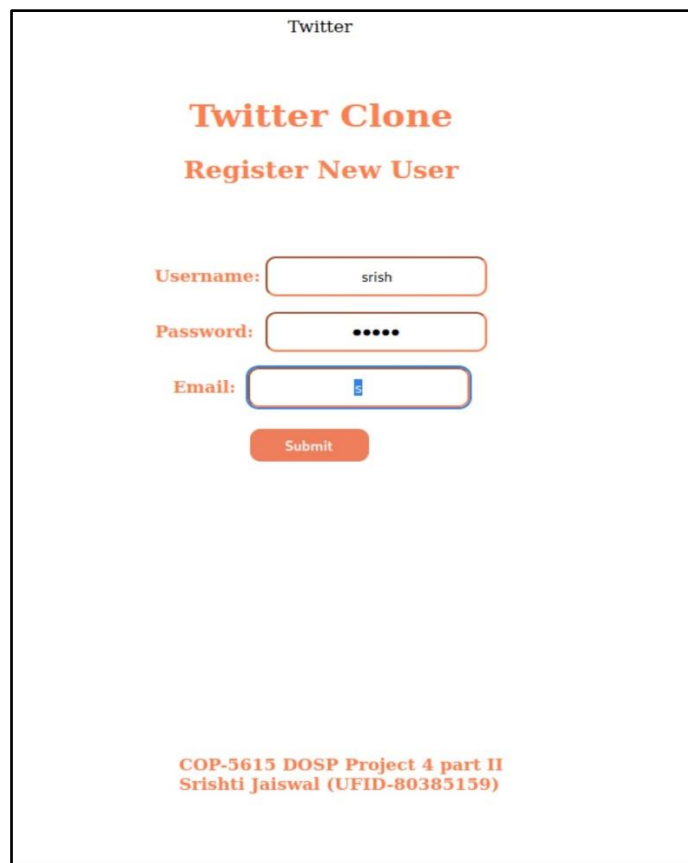
## Screenshots of the results

### *User Interface*




A screenshot of a web application interface for a 'Twitter Clone'. At the top center is a red Twitter bird logo. Below it, the text 'Hello User, Welcome to Twitter Clone!' is displayed in a bold, orange-red font. Underneath this is a 'Login' label in the same color. The login form consists of two input fields: 'Username:' and 'Password:', both with orange-red borders. Below these fields are two orange-red buttons labeled 'Login' and 'Register'. At the bottom of the page, there is a footer in a small, orange-red font that reads: 'COP-5615 DOSP Project 4 part II' and 'Srishti Jaiswal (UFID-80385159)'.

### *Registering a user (Username: Srish)*



A screenshot of a web application interface for a 'Twitter Clone' registration page. At the top center, the word 'Twitter' is written in a small, grey font. Below it, the text 'Twitter Clone' is displayed in a bold, orange-red font, followed by 'Register New User' in the same color. The registration form has three input fields: 'Username:' with the value 'srish', 'Password:' with masked characters '•••••', and 'Email:' with a blue icon. Below these fields is an orange-red button labeled 'Submit'. At the bottom of the page, there is a footer in a small, orange-red font that reads: 'COP-5615 DOSP Project 4 part II' and 'Srishti Jaiswal (UFID-80385159)'.

***Logging in a user (Registered user can log in instead of registering again)***



**Hello User, Welcome to Twitter Clone!**

**Login**

Username:

Password:

COP-5615 DOSP Project 4 part II  
Srishti Jaiswal (UFID-80385159)

***Subscribe to a user: This can be done by entering the username and clicking the subscribe button (Test is subscribing to Srish)***

twitter

localhost:8080/test/main

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Twitter

**Twitter Clone**

**Available Features**

Tweet:

Hashtag:

Mention:

Subscribed Username:

Username:

COP-5615 DOSP Project 4 part II  
Srishti Jaiswal (UFID-80385159)

*A tweet can be sent by the user when he types the message and hits the tweet button*

The screenshot shows a web browser at localhost:8080/srish/main. The page is titled "Twitter" and "Twitter Clone". Under "Available Features", there are input fields for "Tweet:", "Hashtag:", "Mention:", and "Subscribed Username:". Each has a corresponding "Search" button. At the bottom, there is a "Username:" field with a "Subscribe" button. A dropdown menu is open for the "Tweet:" field, showing options: "hello", "hello", and "hello #123". A red "tweet" button is visible next to the dropdown. At the bottom right, there is a "My Feed" button. The footer text reads: "COP-5615 DOSP Project | Test 1 | Srishti Jaiswal (UPII-801401130)"

*The same tweet will be visible from another user's page (i.e. Test) as he has subscribed to Srish*

*(Creating another client named "Test3" and subscribe to "Test")*

*"Test" can retweet the message sent by "Srish" and "Test 3" will be able to see this tweet as he has subscribed to "Test"*

The screenshot shows a web browser at localhost:8080/test/feed?. The page is titled "Tweets". There is a text input field containing "hello" and a blue "retweet" button next to it.

## QUERYING

*Typing the tweet “hello @srish”*

Twitter

### Twitter Clone

#### Available Features

Tweet:  Tweet

Hashtag:  Search

Mention:  Search

Subscribed Username:  Search

Username:  Subscribe

COP-5615 DOSP Project Part II  
Srishti Jaiswal (UP) My Feed

*We can find the tweets mentioning @srish by typing it in the search box*

## Twitter Clone

### Available Features

Tweet:

Tweet

Hashtag:

Search

Mention:

Search

Subscribed Username:

Search

Username:

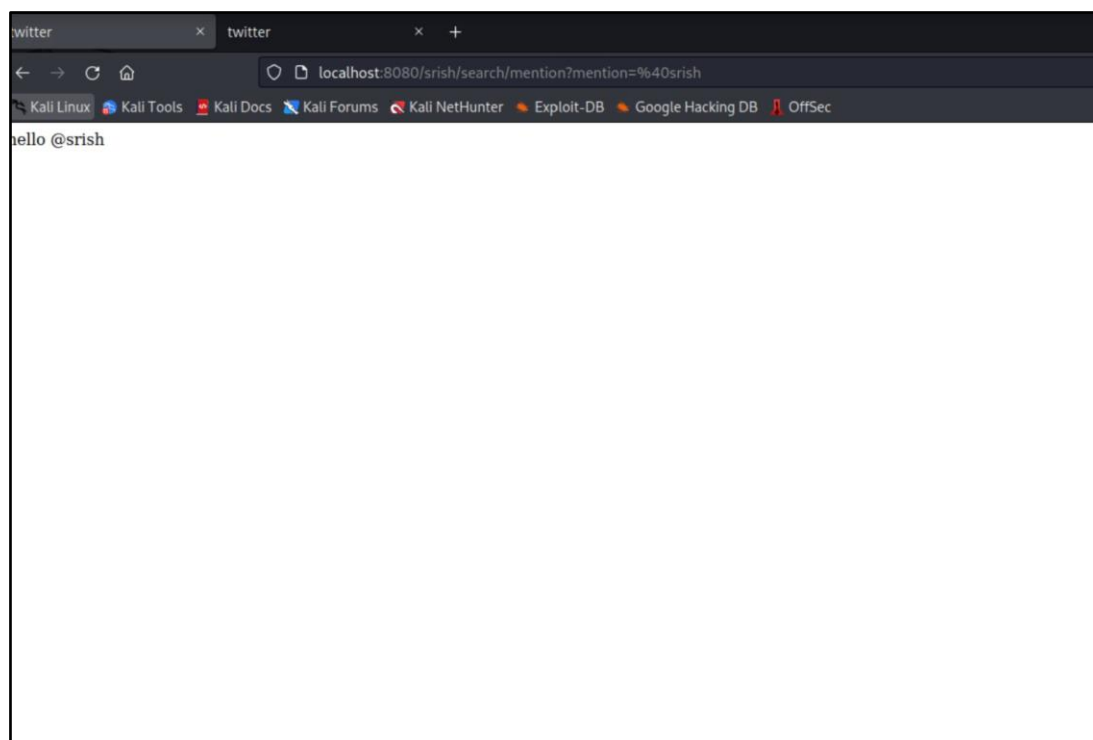
Subscribe

COP-5615 DOSP Project Part II

My Feed

Srishti Jaiswal (UID-89383559)

*As you can see below the tweet mentioning the user @srish is visible*





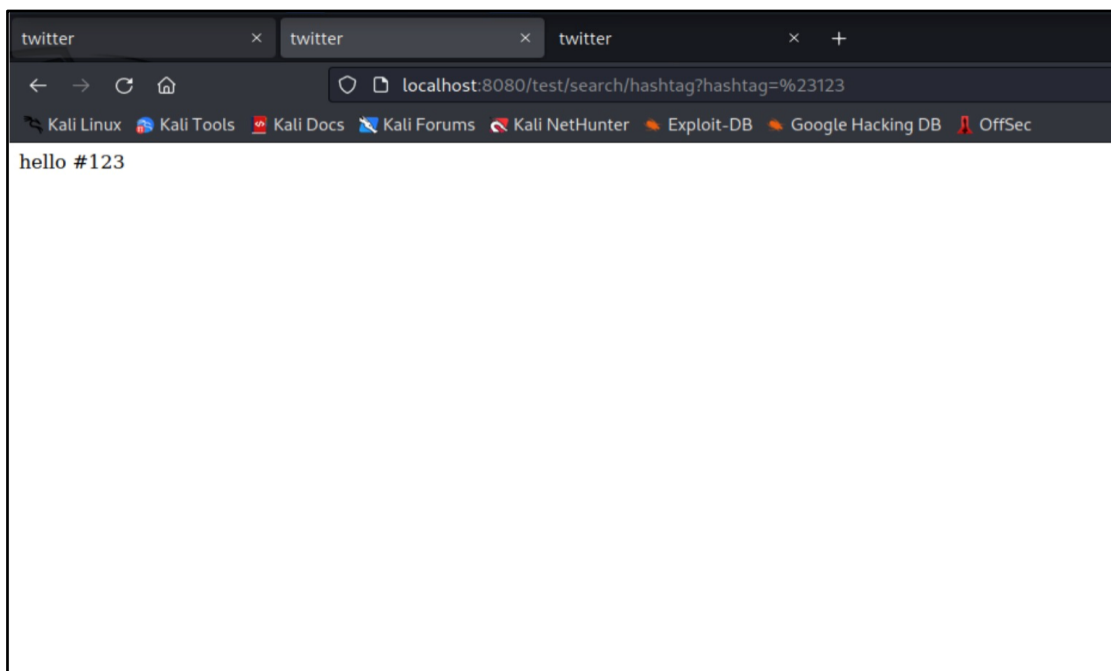
## HASHTAG

*We can also query for tweets having a specific hashtag like #123  
This will return all the tweets that contain the hashtag #123*

The screenshot shows a web browser window with the URL `localhost:8080/test/main`. The page title is "Twitter". The main heading is "Twitter Clone". Below it, the section "Available Features" lists several search and interaction options, each with a text input field and a corresponding button:

- Tweet:**  Tweet
- Hashtag:**  Search
  - #123
  - #12345
- Mention:**  Search
- Subscribed Username:**  Search
- Username:**  Subscribe

At the bottom, there is a "My Feed" button and a list of users: "COP-5615 DOSP Project Part II" and "Srishti Jaiswal (UP)".



## SERVER TRACE

*For logging purposes and to view the whole list of performed actions, utilize the server trace.*

```
heart_beat_kill_pid = 127111
Erlang/OTP 25 [erts-13.1.2] [source] [64-bit] [smp:3:3] [ds:3:3:10] [async-threads:1] [jit:ns]

jfdssddsfEshell V13.1.2 (abort with ^G)
(twitter@127.0.0.1)1> "srish" "Profile_Added"
"test" "Profile_Added"
"test":Tweet was Added
"Tweet_Added"
"srish":Adding to Feed
["hello @srish"]
["hello @srish"]
"test3" "Profile_Added"
"test3":Tweet was Added
"Tweet_Added"
"test3":Tweet was Added
"Tweet_Added"
"test3":Tweet was Added
"Tweet_Added"
"test3":Tweet was Added
"Tweet_Added"
```

## CONCLUSION

The erlang-based addition to part I of the Twitter clone project was successfully finished. The whole operation of the Twitter engine may be handled by the WebSocket-based client, and all queries and answers make use of a JSON-based API.

## VIDEO LINK