COP5615 - DOSP Project 4 Part 2
Erlang Twitter Clone

Team Members:
1. Aadithya Kandeth - 69802791
2. Anol Kurian Vadakkeparampil - 56268544

OBJECTIVE

As a part of part 1 of this project, we had to implement a twitter server using Erlang along with multiple clients that could run on separate systems. The clients are used to send and receive tweets to/from other subscribed clients and the server is used to distribute the tweets. Both of these have to be run in separate processes. For part 2, the objective is to use a WebSharper web framework to add a websocket interface to part 1 project. The three additional factors were:

1. All requests and responses had to be through a JSON based API
2. Parts of the twitter engine has to use the WebSharper framework
3. The client has to use WebSockets

PROJECT REQUIREMENTS

- The project has to implement the following functionality:
- Twitter Engine: -
- Register: The twitter engine has to be able to register a new user
- Sending tweets: The tweets can have hashtags and can mention users.
- Subscribe to a user's tweets: A user should receive any tweets made by someone who they are subscribed to.
- Retweeting: Re-tweets are used for sharing tweets made by another user.
- Tweet Querying: query tweets that a user has subscribed to, tweets that hold specific hashtags, and tweets where the user is mentioned.
- Live tweets: A connected user should receive these tweets without having to query them.
- All messages and their responses must be represented via a JSON-based API.
- The client has to be written using websockets.
- A websocket interface has to be added using the WebSharper framework

INTRODUCTION

Pre-requisites -

The program uses Erlang and needs it installed on the system to run.
- Two separate terminals have to be created for the client and server.

Note: The program is best executed on Linux

Execution Instructions and simulation

- Navigate to the project folder
- open cmd

>make run

- if issues are found

In cmd:

>make distclean

>make run

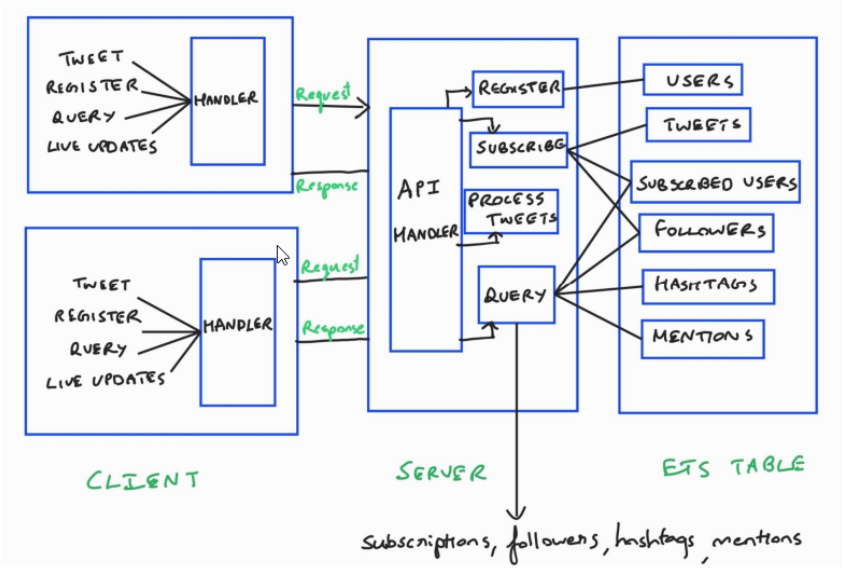
- Once the server is started, test it using localhost:8080.
- This should bring up the twitter websocket based client.
- Here on the Twitter web client, we can test all the features
- The first step should be to register two new users beyond which all twitter engine functionality like tweeting, retweeting, subscribing and querying can be tested.

IMPLEMENTATION

Architecture

The architecture remains the same as that of project 1. The websocket interface is implemented on top of this. All messages and responses are represented via a JSON-based API. The server is an interface between the client and the ETS table and retrieves information as specified by the user. The twitter server is responsible for registering a user, handling tweets and processing queries that are sent. The database stores the user list, information about the tweets, and the details like subscribers, followers for a user along with hashtags and mentions. The server can handle API calls to perform tasks based on the inputs entered by the user and then it retrieves data from the ETS table to return what the user expects. The client is responsible for sending requests to the server. We use multiple clients to test the functionality of the server in a twitter clone system.

Architecture Diagram



REST API

• Requests:We use an API request to send a JSON object from the server to 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”:”@aadi aadi”
}
There are three different request methods: GET, PUT, and POST
GET - Used for reading or retrieving a resource
PUT - Used for modifying a resource
POST - Used for creating a new resource

WEBSOCKET IMPLEMENTATION

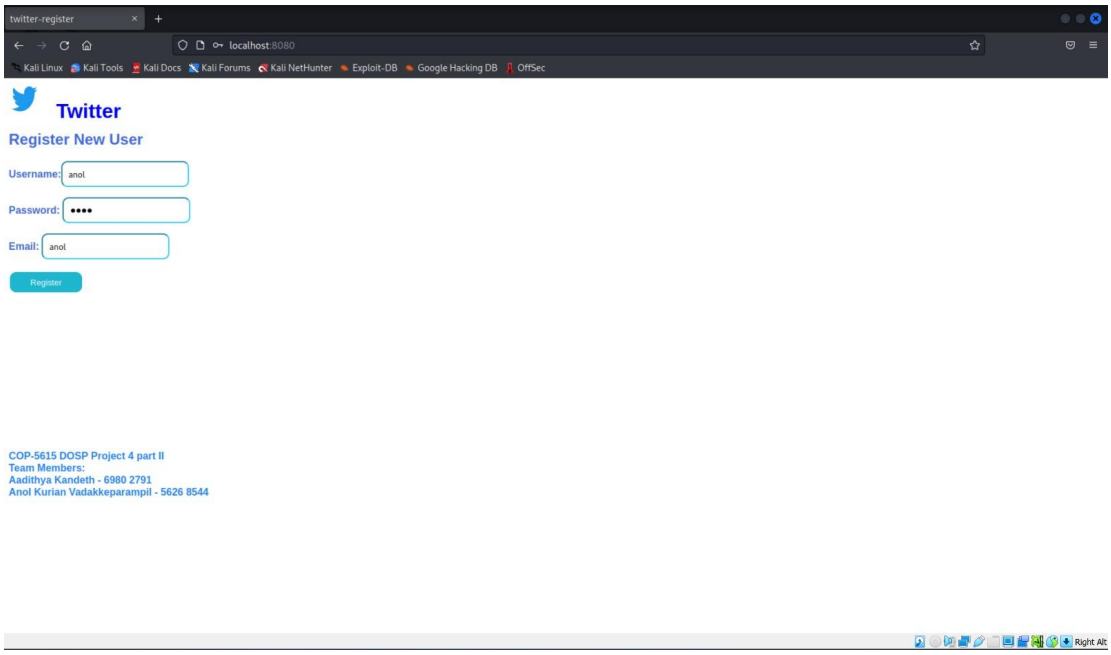
We use Cowboy for the WebSocket implementation. Cowboy is a modern HTTP server for Erlang.

FEATURES:

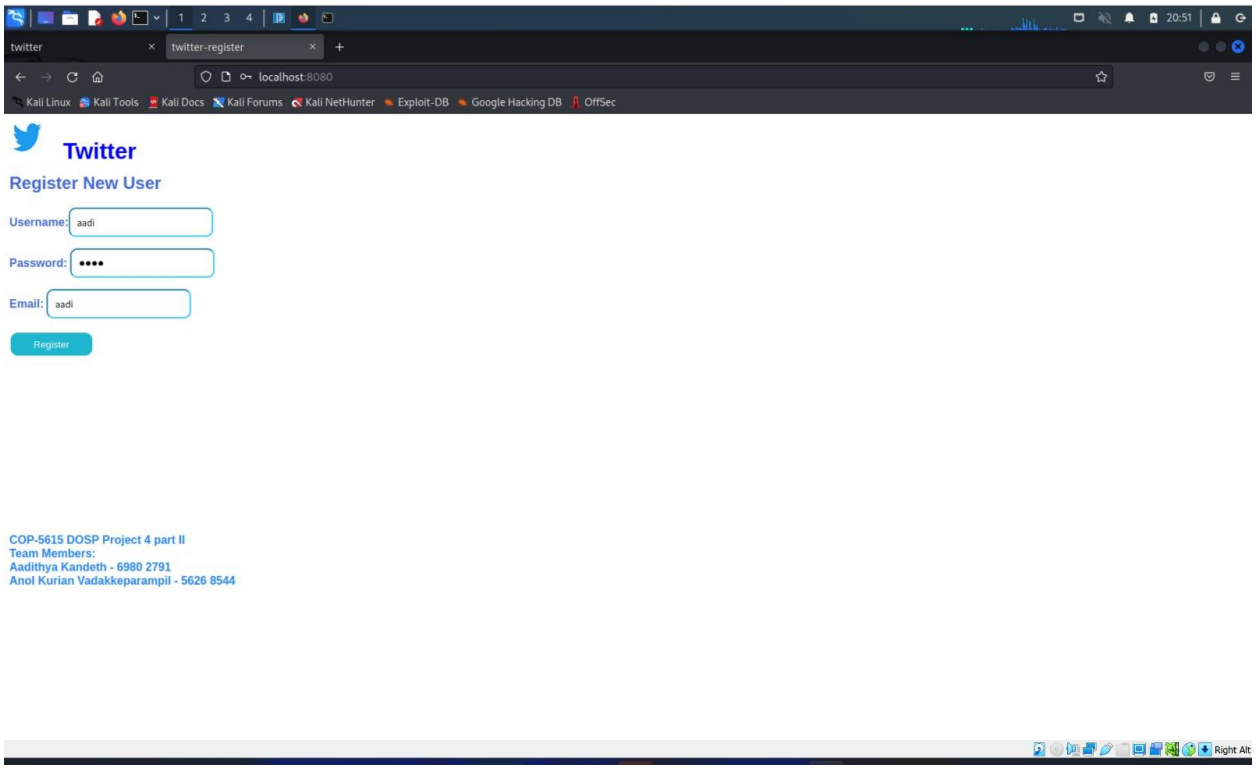
The following screenshots show the complete working of the twitter web client.

- Register User: First we add two users, aadi and anol for testing the tweet system.

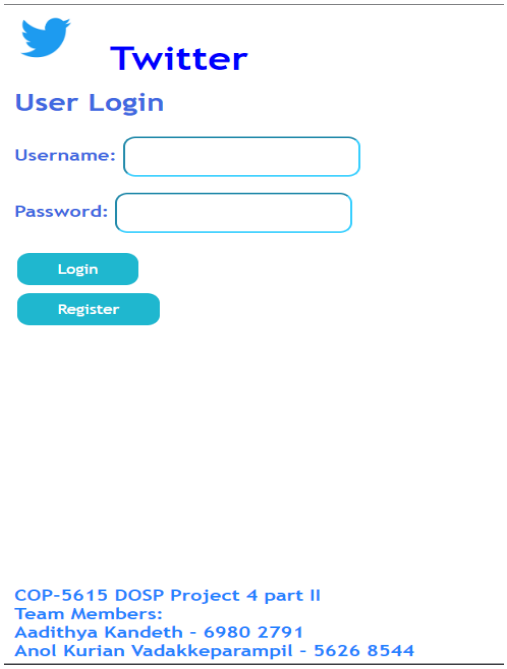
Registering anol:



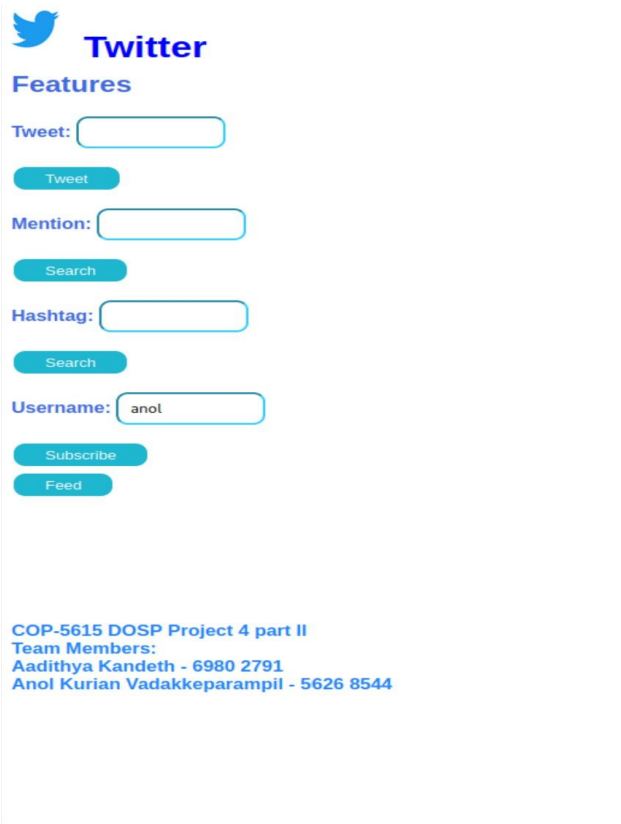
Registering Aadi



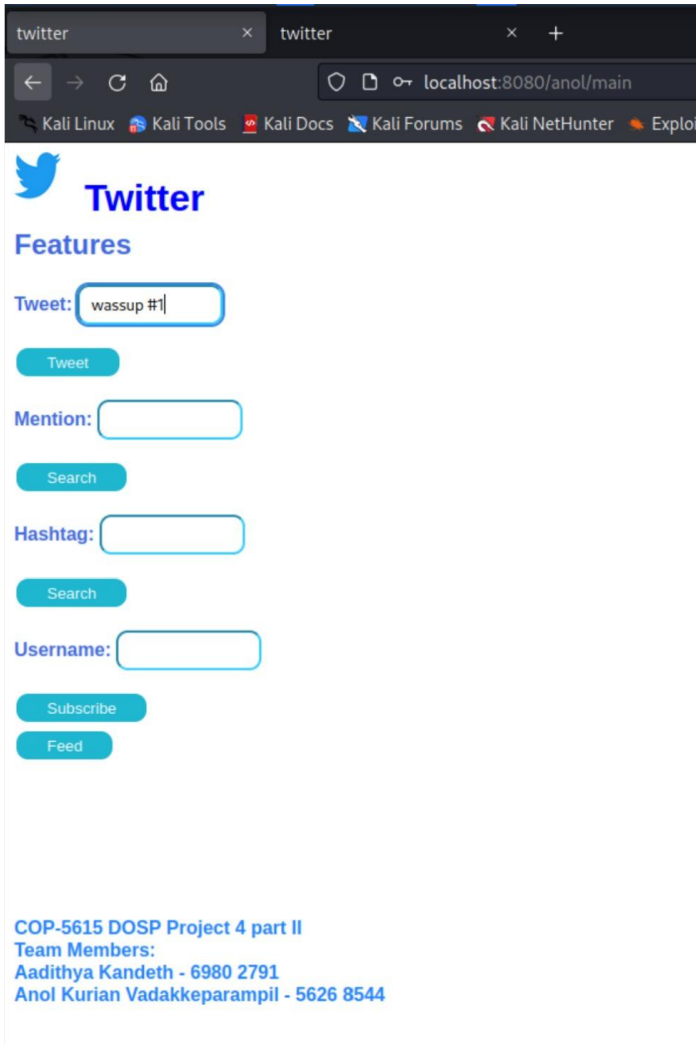
Login in required:
Once users are registered, we have the option to log in rather than registering again.



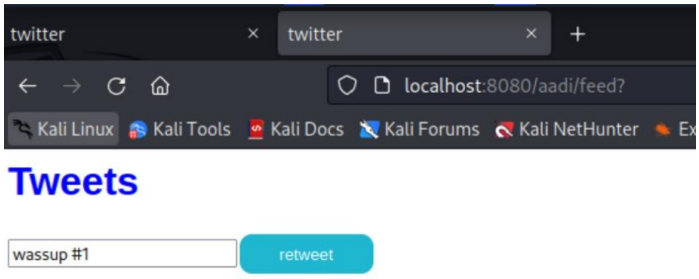
Subscribe to a user: We can subscribe to any user by typing the user name from a user clients page and hitting subscribe.



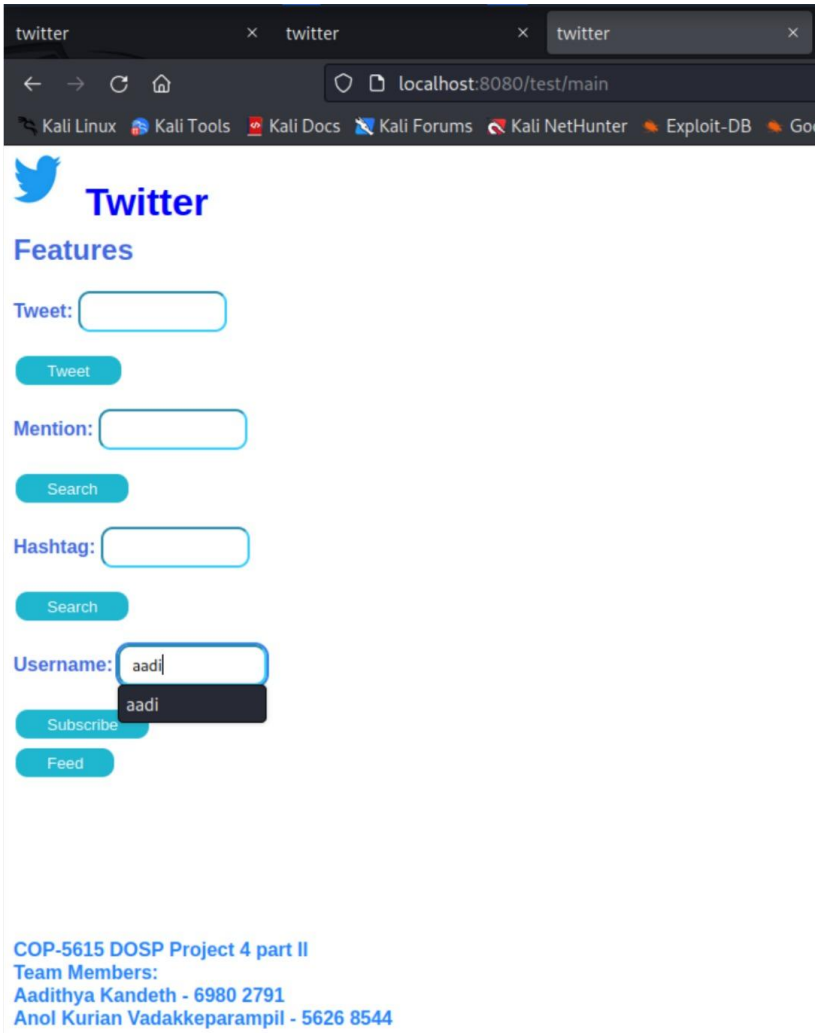
Tweet: A user tweets by typing and hitting tweet. Here, anol tweets “wassup #1”



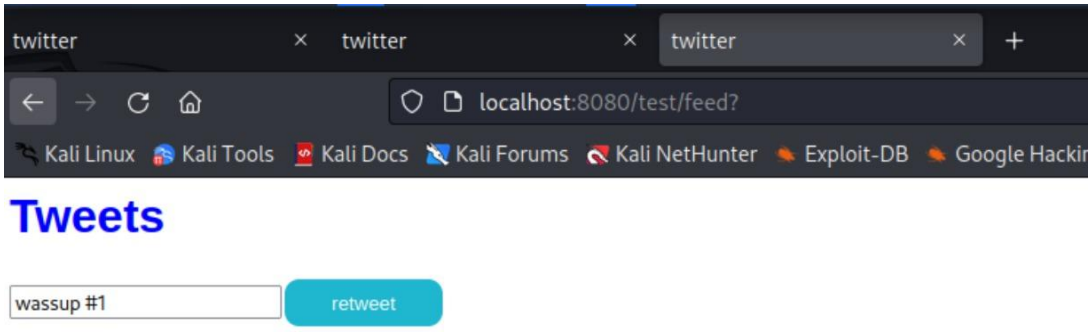
This tweet can be visible when feed is selected from aadi’s page since he is subscribed to anol.



The next step is to create a third client called test and subscribe to aadi.

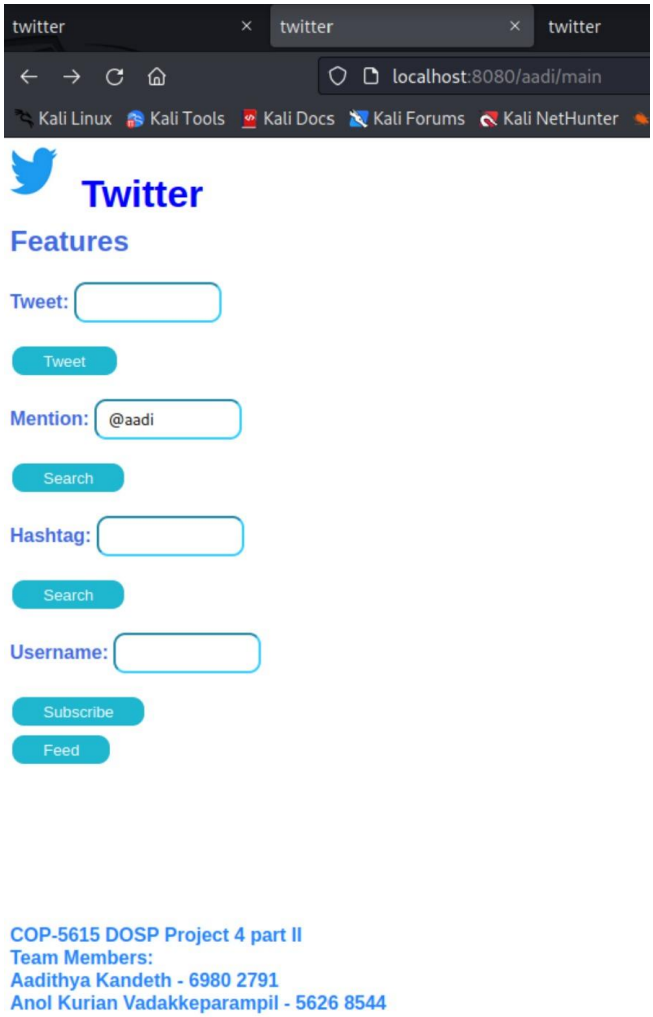


Retweet: Aadi retweets anol's original tweet. This tweet is then visible on test clients feed since it is subscribed to aadi.

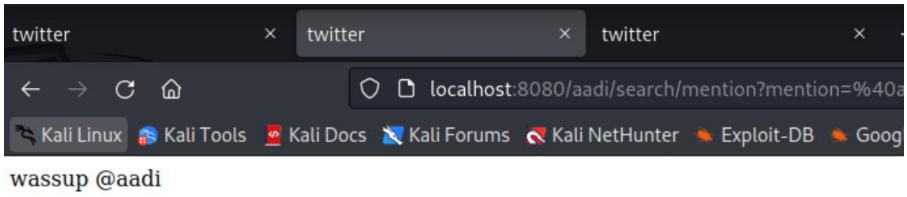


Querying:

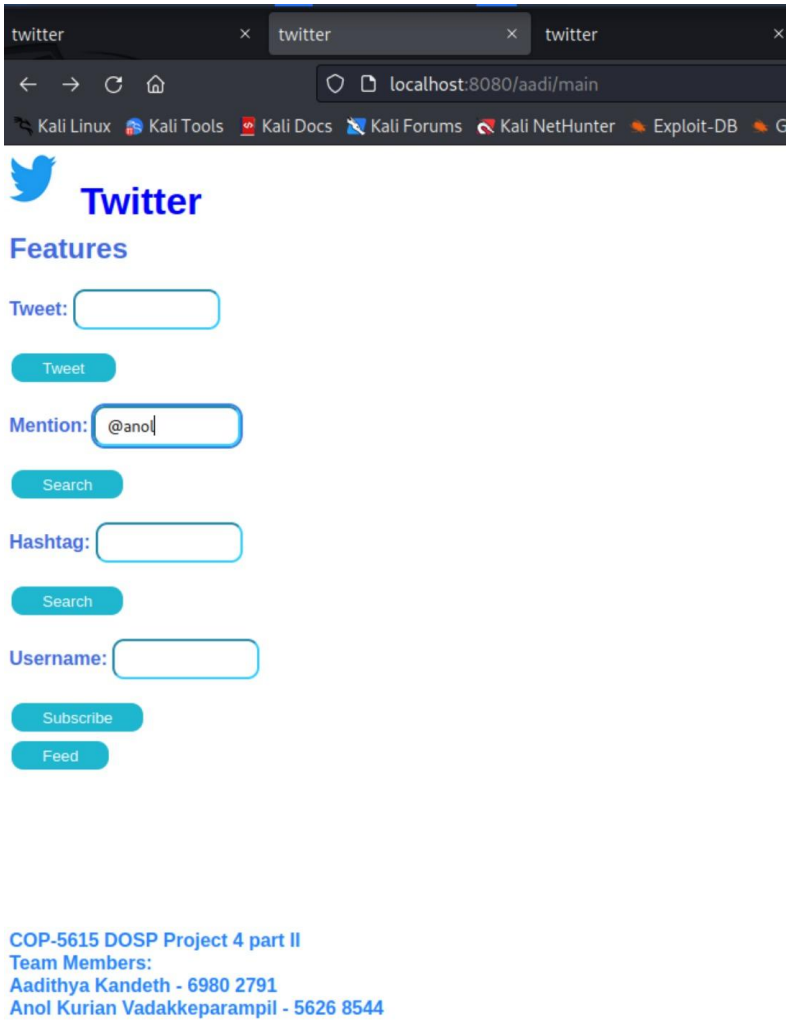
Tweets mentioning aadi: We search for @aadi in the search box.



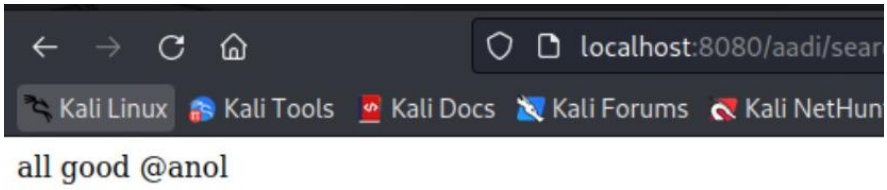
This returns a set of tweets that mention user aadi (@aadi)



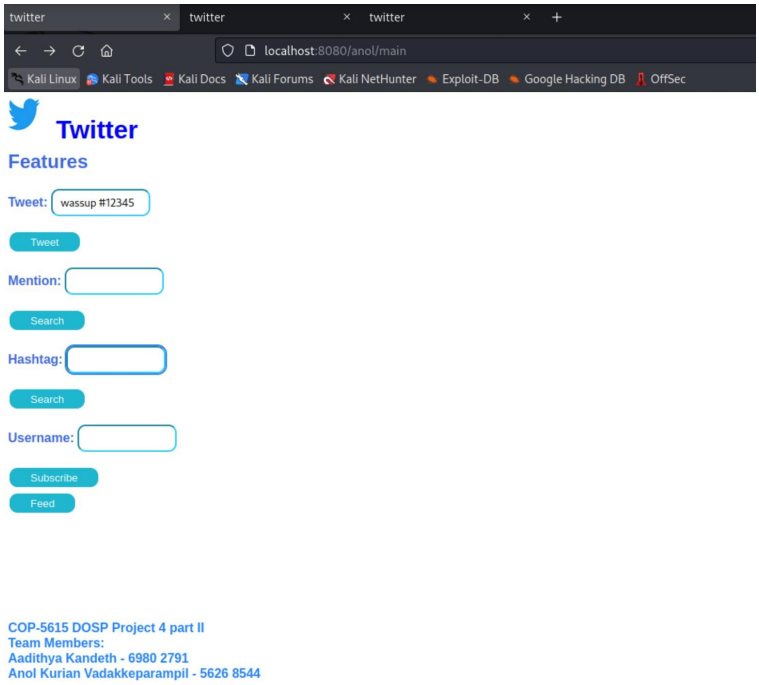
Similarly, we can try with another user @anol



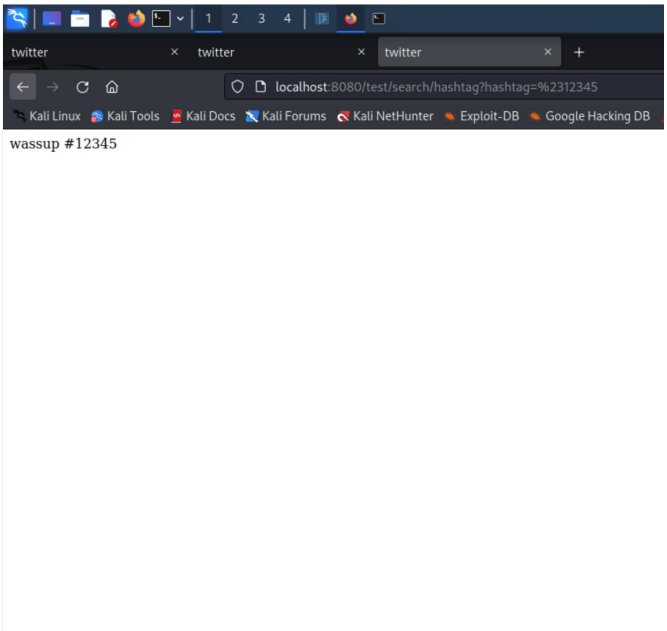
This will return a set of tweets that mention the user @anol



Hashtags: We can also query for tweets containing specific hashtags like #12345
This will return a set of tweets that have the hashtag #12345



Querying a hashtag will return the set of tweets that contain that specific hashtag.



Server Trace: The server trace is used for logging and to see the entire list of operations that have been executed.

```
/home/kali/Desktop/Twitter-Clone-Main/twitter/_rel/twitter_release
heart_beat_kill_pid = 80600
Erlang/OTP 25 [erts-13.1.2] [source] [64-bit] [smp:3:3] [ds:3:3:10] [async-th
reads:1] [jit:ns]

jfdssddsfEshell V13.1.2  (abort with ^G)
(twitter@127.0.0.1)1> "anol" "Profile_Added"
"anol":Tweet was Added
"Tweet_Added"
"aadi" "Profile_Added"
"test" "Profile_Added"
"Subscription_Done"
"anol":Tweet was Added
"Tweet_Added"
"aadi":Adding to Feed
"Subscription_Done"
"aadi":Tweet was Added
"Tweet_Added"
"test":Adding to Feed
"anol":Tweet was Added
"Tweet_Added"
"aadi":Adding to Feed
"aadi":Adding to Feed
"aadi":Tweet was Added
"Tweet_Added"
"anol":Adding to Feed
"test":Adding to Feed
["wassup @aadi"]
["wassup @aadi"]
["all good @anol"]
["all good @anol"]
["wassup @aadi"]
["wassup @aadi"]
["all good @anol"]
["all good @anol"]
["wassup #1","wassup #1","wassup #1"]
["wassup #1","wassup #1","wassup #1"]
"anol":Tweet was Added
"Tweet_Added"
"aadi":Adding to Feed
["wassup #12345"]
["wassup #12345"]
```

FILES USED :

- client.erl - The client handles different functionality like tweet, retweet, subscribe, etc and sends requests to the server.
- Twitter.erl - This is the twitter engine/ server that handles requests from the client

- The following files are used for handling specific functionality of the twitter system based on the name of the file:
  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 for handling various utility functions like random string generation, etc
- Twitter_app.erl - Handles the localhost and server application hosting
- Twitter_sup.erl - Supervisor

CONCLUSION:

The project was successfully completed as an extension to the part I of the twitter clone using erlang. The websocket based client was able to handle the complete functionality of the twitter engine and all requests and responses use a JSON based API.

VIDEO DEMONSTRATION LINK:

https://www.youtube.com/watch?v=5qoCKLSJXpw

https://drive.google.com/file/d/12ArGHA5BseqWcXg6zTvymp3dXbPuSLca/view?usp=sharing