PROJECT 4 PART 2: TWITTER CLONE

HOW TO RUN THE PROGRAM

Video Description -
https://youtu.be/noxN0aTtiGs

- Save the file on your system as twitter.erl
- The latest version of Erlang/OTP 25.1 is recommended.
- Run the erl command on your system to ensure it is installed correctly.
- Ubuntu or any other Linux based operating system is recommended.
- Move to the directory where the program is saved.
- Sudo make run
- *You might need to install `make -f erlang.mk bootstrap bootstrap-rel`

  If the dependencies are not installed
- You can simulate multiple users on http://localhost:8080
- All the working functions have been displayed in the video description.

README REQUIREMENTS AND ANALYSIS

- What is working
  We created a Twitter-cloning engine with a client simulator and all the functionality of the back end of a Twitter application for part 1 of this assignment. For this part, we implemented a WebSocket interface which includes an API calls for all the functions with necessary error handling. Parts of the engine and client code were rewritten to support this interface. The performance is detailed below.

- We included a demo video that demonstrates
  (1) Postman is used to show JSON based API for all the functions
  (2) Cowboy and Erlang frameworks were used for WebSockets
  (3) Users utilizing WebSockets. This is also explained in detail below along with distributed implementation and the Pid's of the Users.

BONUS
The project since part 1 uses the sha256 authentication for the passwords.
The JSON replies for authentication are included in the video for the system sha256 challenge authentication.
The encryption and its use is described in detail below.

Description

Actor Model

The actors are users generated in the terminal and stored in a list in the engine. When they log in, this list is pulled and the password they registered, which is secured using SHA-256 encryption, is compared with the hashed password they entered to log in. If it matches, the user is able to send tweets, retweet, subscribe to users and hashtags, query the engine for tweets with particular hashtags or mentions, and receive the tweets they have subscribed to live.

Web Interface

This version of the twitter clone implements a real time web interface. We use Cowboy for this project, which is a web/server framework for the Erlang programming language for HTTP stack using TCP.

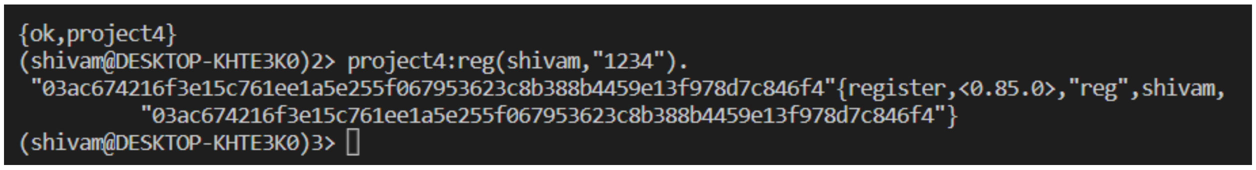
Run

- This program includes a Makefile so the process for running is shown below.

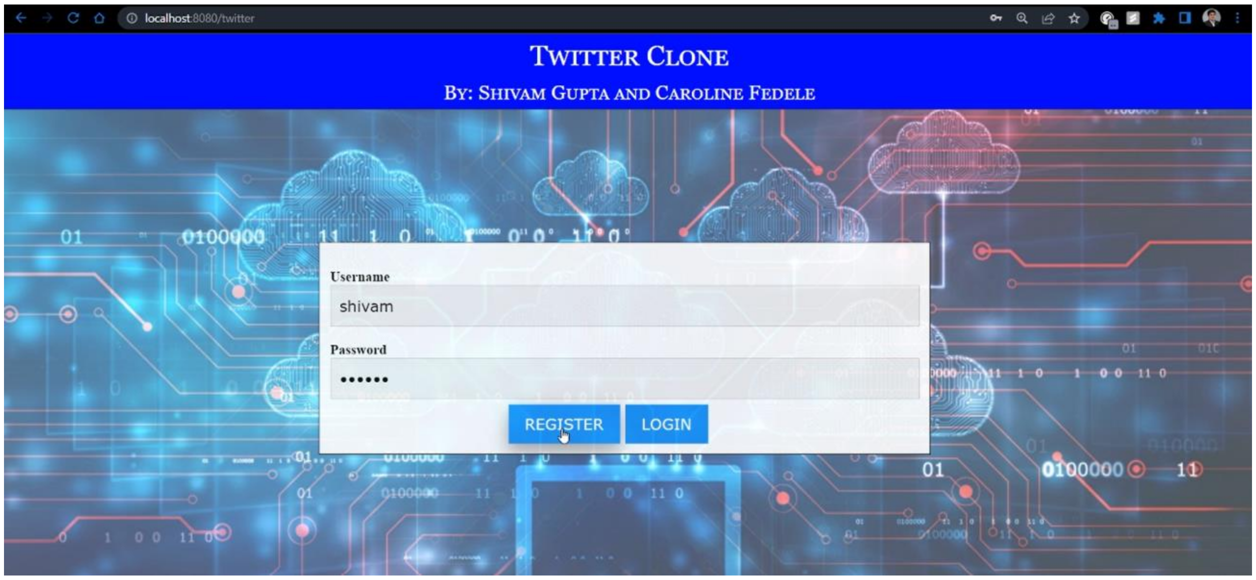
Functions

- Register Account
  - This is done with the **reg(Username, “Password”)** where a new user is appended to the saved in the Users List generated in **server(User_List).**

User1: Shivam registers with the encrypted password (sha256) stored for authentication.



```
{ok,project4}
(shivam@DESKTOP-KHTE3K0)2> project4:reg(shivam,"1234").
 "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4"{register,<0.85.0>,"reg",shivam,
          "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4"}
(shivam@DESKTOP-KHTE3K0)3> 
```



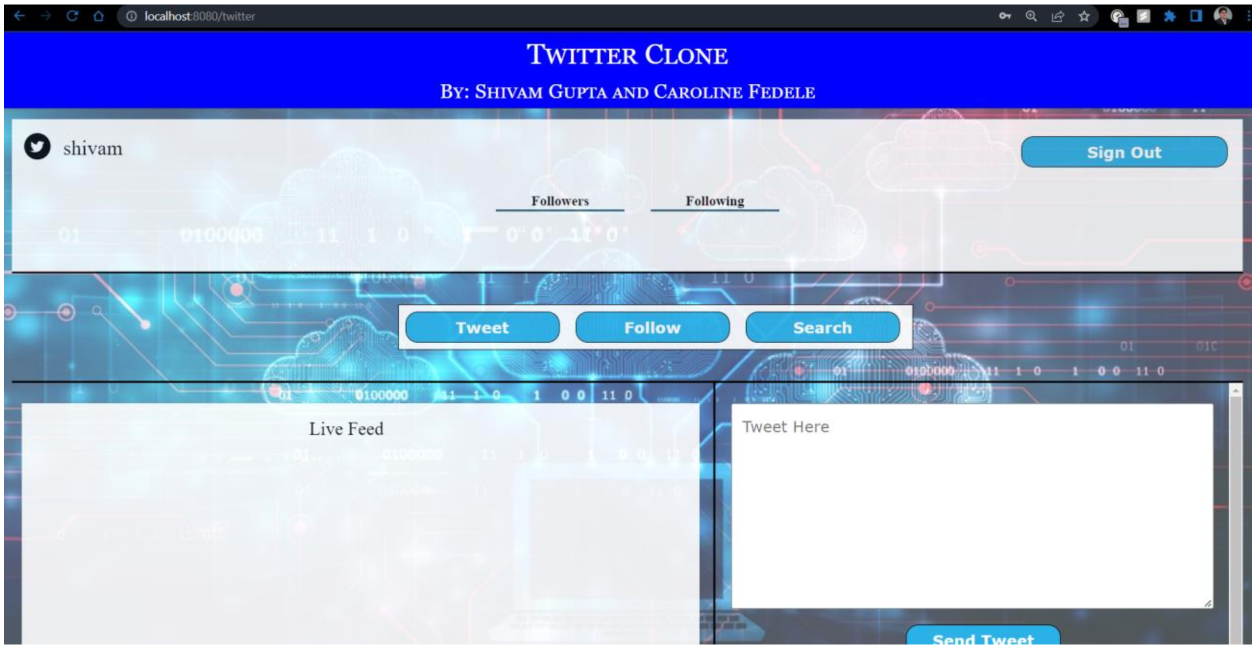
localhost:8080/twitter

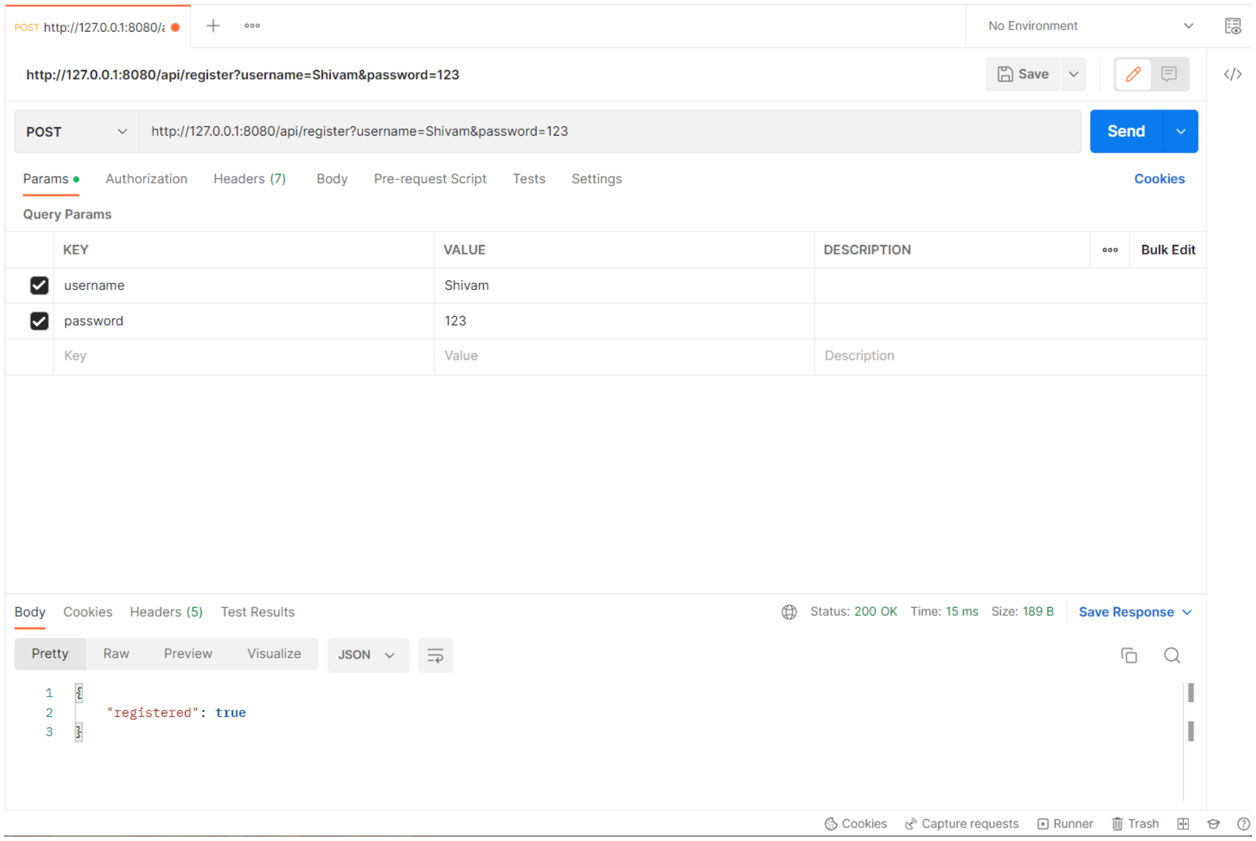
Twitter Clone

By: Shivam Gupta and Caroline Fedele

Username: shivam

Password: ••••••




POST http://127.0.0.1:8080/api/register?username=Shivam&password=123

Query Params

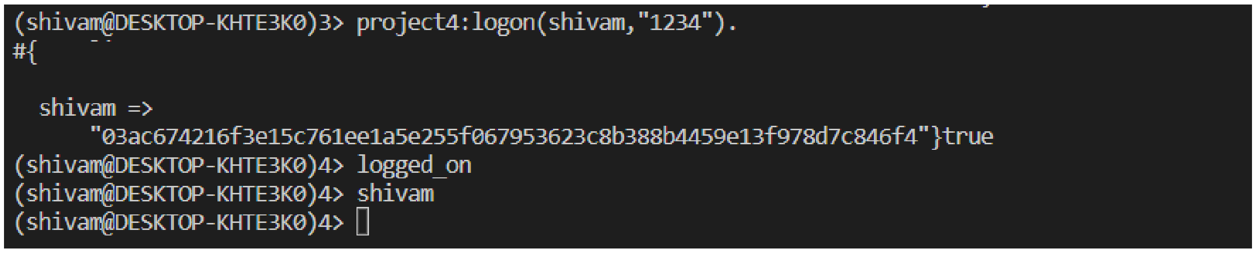
KEY	VALUE	DESCRIPTION
✓ username	Shivam	
✓ password	123	
Key	Value	Description

Status: 200 OK Time: 15 ms Size: 189 B

```
{
    "registered": true
}
```

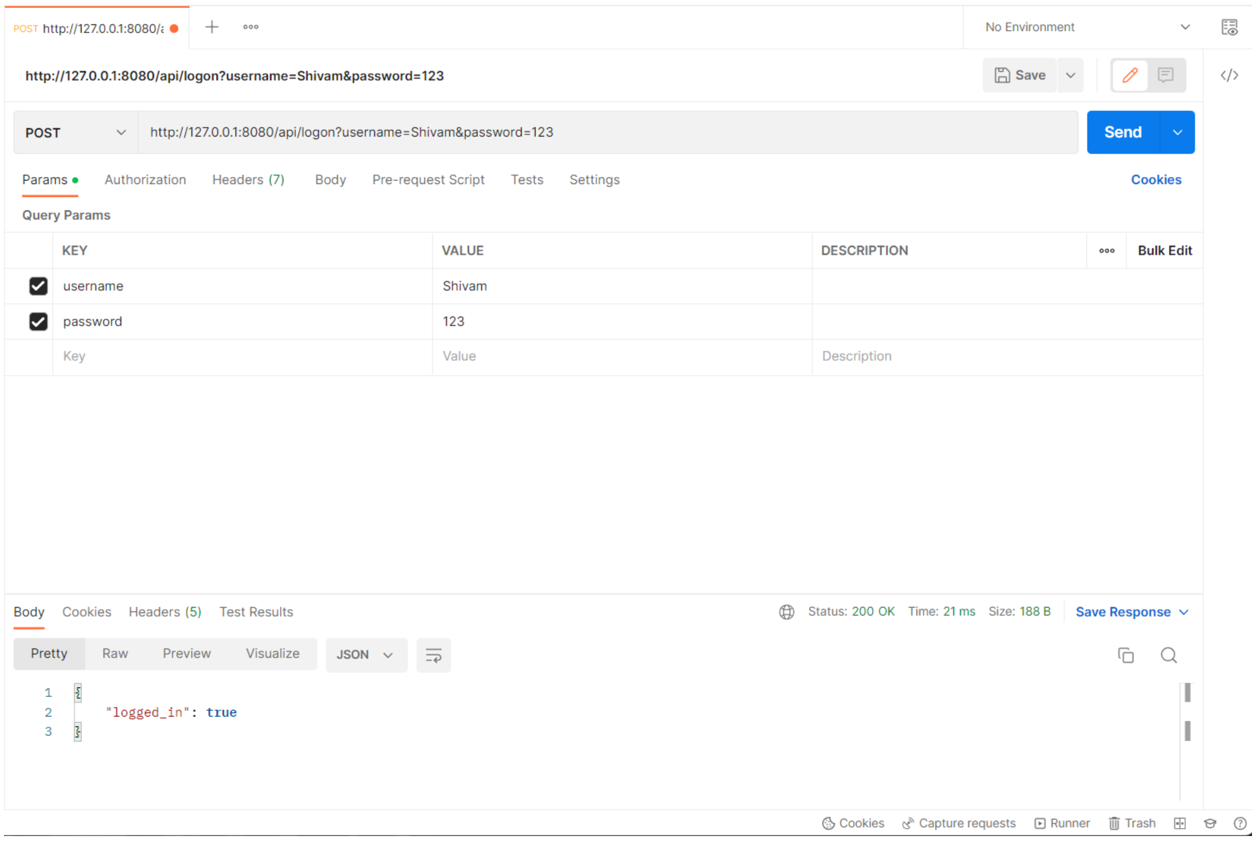
○ The user can then login using **logon(Username, “Password”)**, which first checks that the user exists and then checks that the hash of the password they gave matches the registered password (both encrypted using SHA-256).

User1: Shivam logon after authentication logged_on message is displayed.



```
(shivam@DESKTOP-KHTE3K0)3> project4:logon(shivam,"1234").
#{

  shivam =>
      "03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4"}true
(shivam@DESKTOP-KHTE3K0)4> logged_on
(shivam@DESKTOP-KHTE3K0)4> shivam
(shivam@DESKTOP-KHTE3K0)4> 
```



POST http://127.0.0.1:8080/api/logon?username=Shivam&password=123

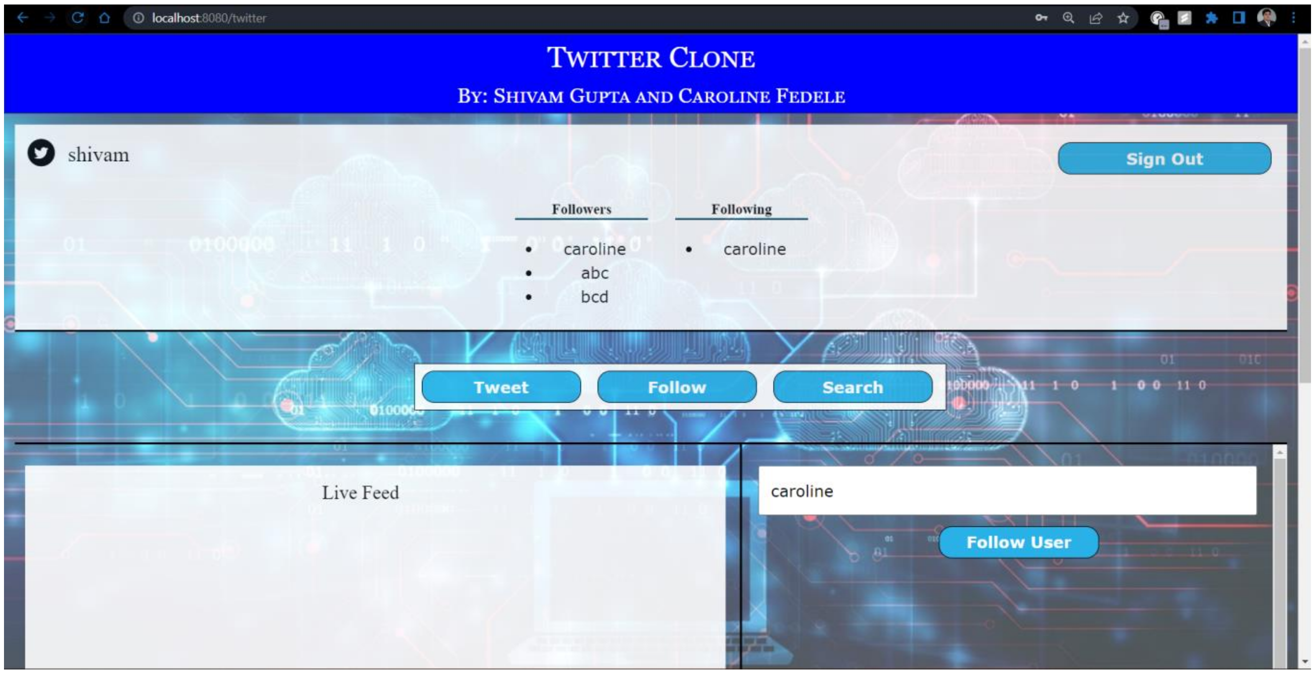
Query Params

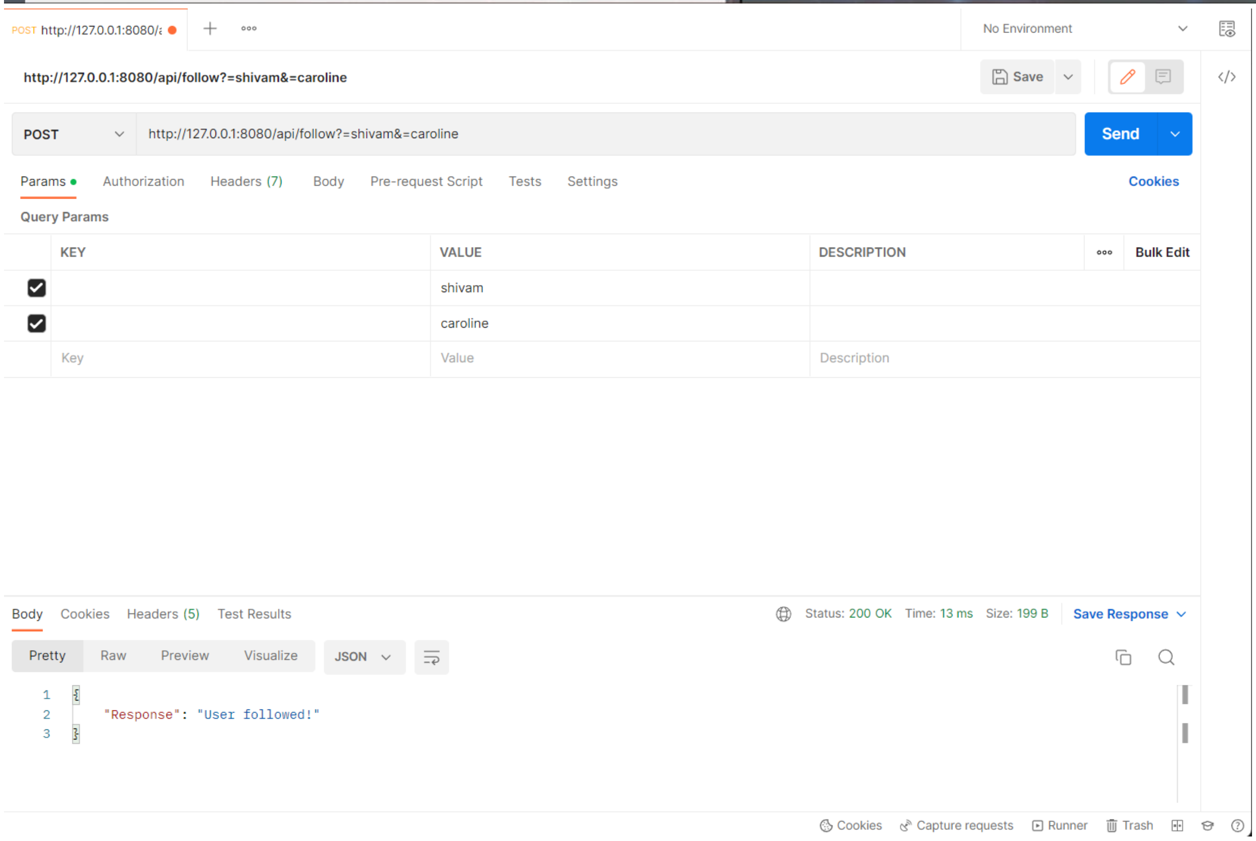
KEY	VALUE	DESCRIPTION
username	Shivam	
password	123	

Status: 200 OK Time: 21 ms Size: 188 B

```
{
    "logged_in": true
}
```

- Subscribe
    - This is done using the **follow(ToName)** function which first checks that the given username exists and then adds the current user to the username's list of followers.
    Server Output: Updated Map that User2 now Follows User1





http://127.0.0.1:8080/api/follow?=shivam&=caroline

POST http://127.0.0.1:8080/api/follow?=shivam&=caroline

Query Params

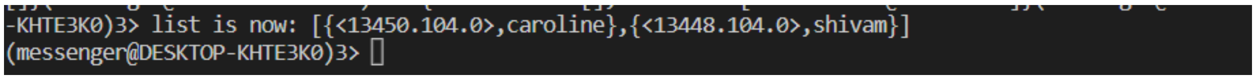
	KEY	VALUE	DESCRIPTION
☑		shivam	
☑		caroline	
	Key	Value	Description

Status: 200 OK Time: 13 ms Size: 199 B

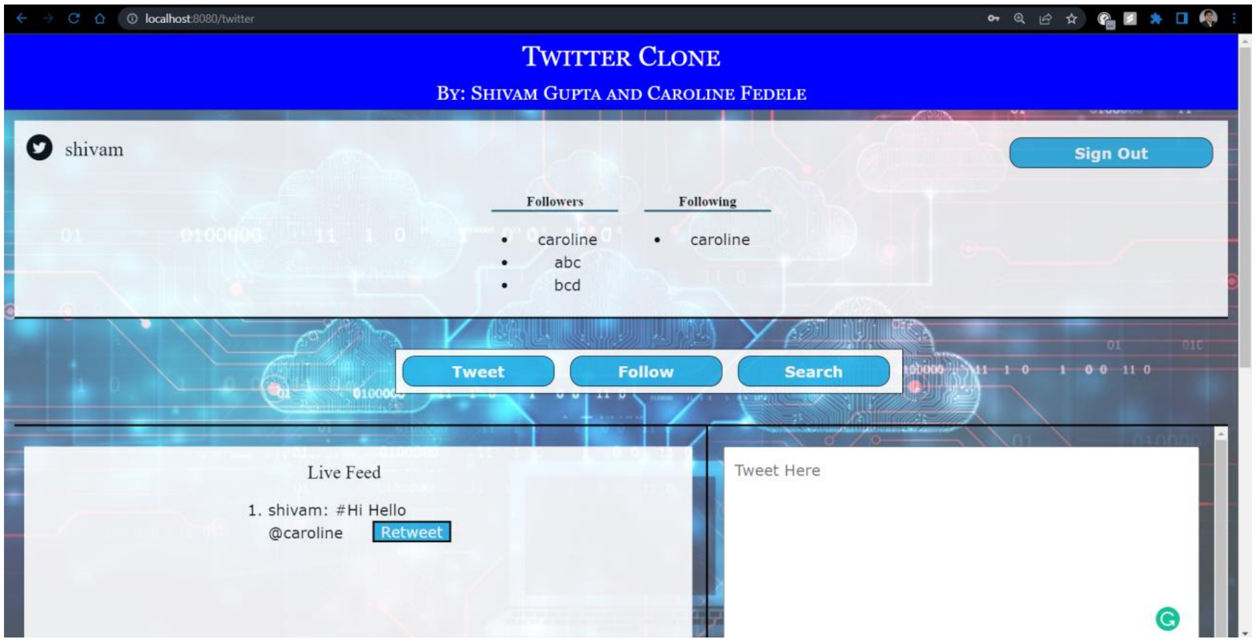
```
{
    "Response": "User followed!"
}
```

- Send Tweet
    - This is done using the **tweet(Message)** function and updates the live feed for all those who are following this user. This additionally updates the list of tweets from this user which can be queried in the future.

    Server Output: Different process id's for both users for **distributed programming**



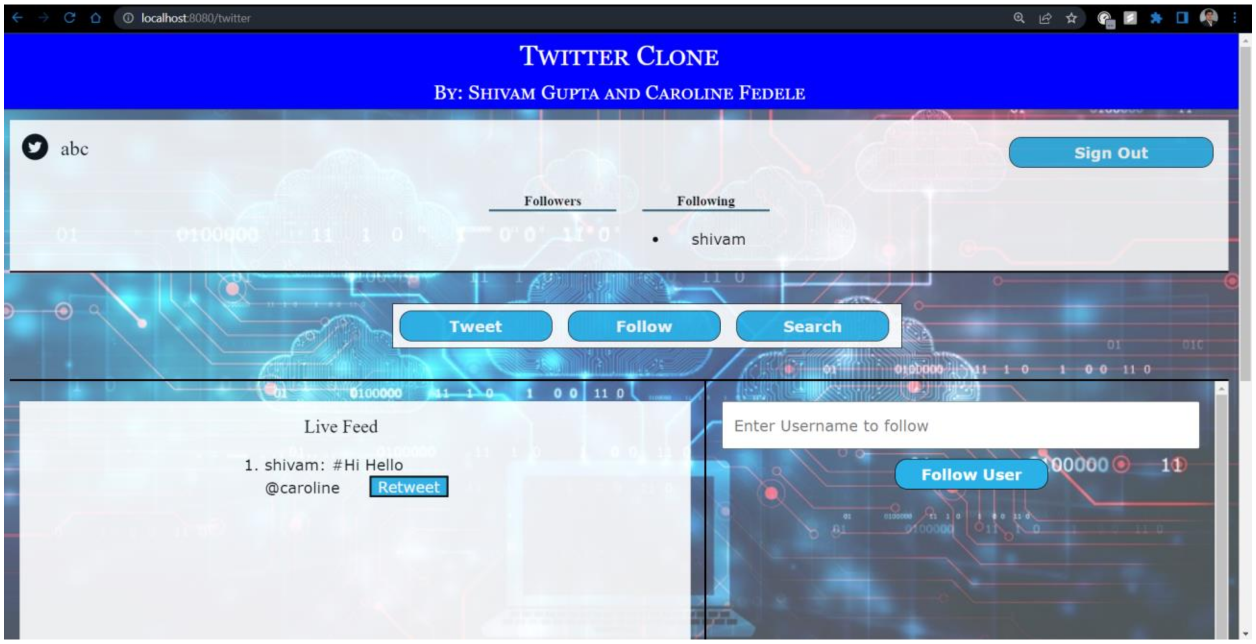
```
-KHTE3K0)3> list is now: [{<13450.104.0>,caroline},{<13448.104.0>,shivam}]
(messenger@DESKTOP-KHTE3K0)3> 
```



Followers: caroline, abc, bcd

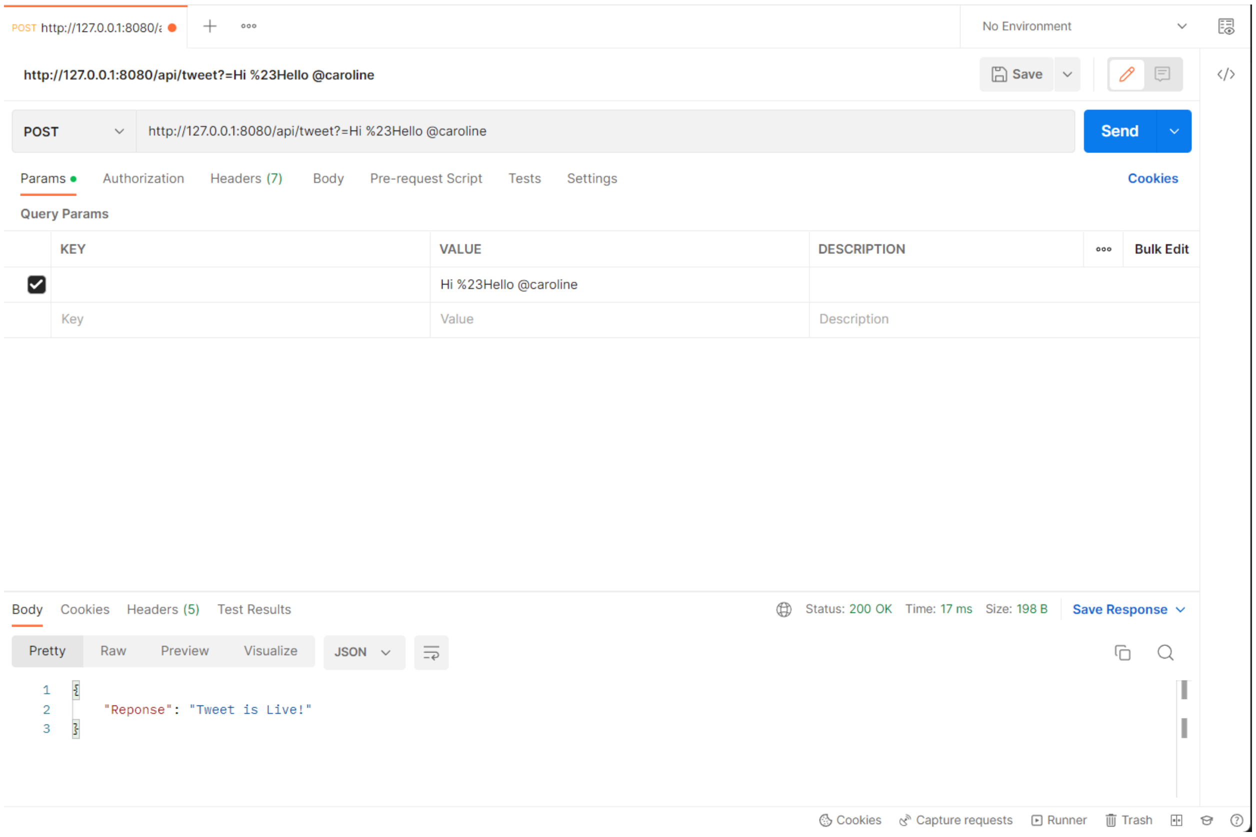
Following: caroline

1. shivam: #Hi Hello @caroline



Following: shivam

1. shivam: #Hi Hello @caroline



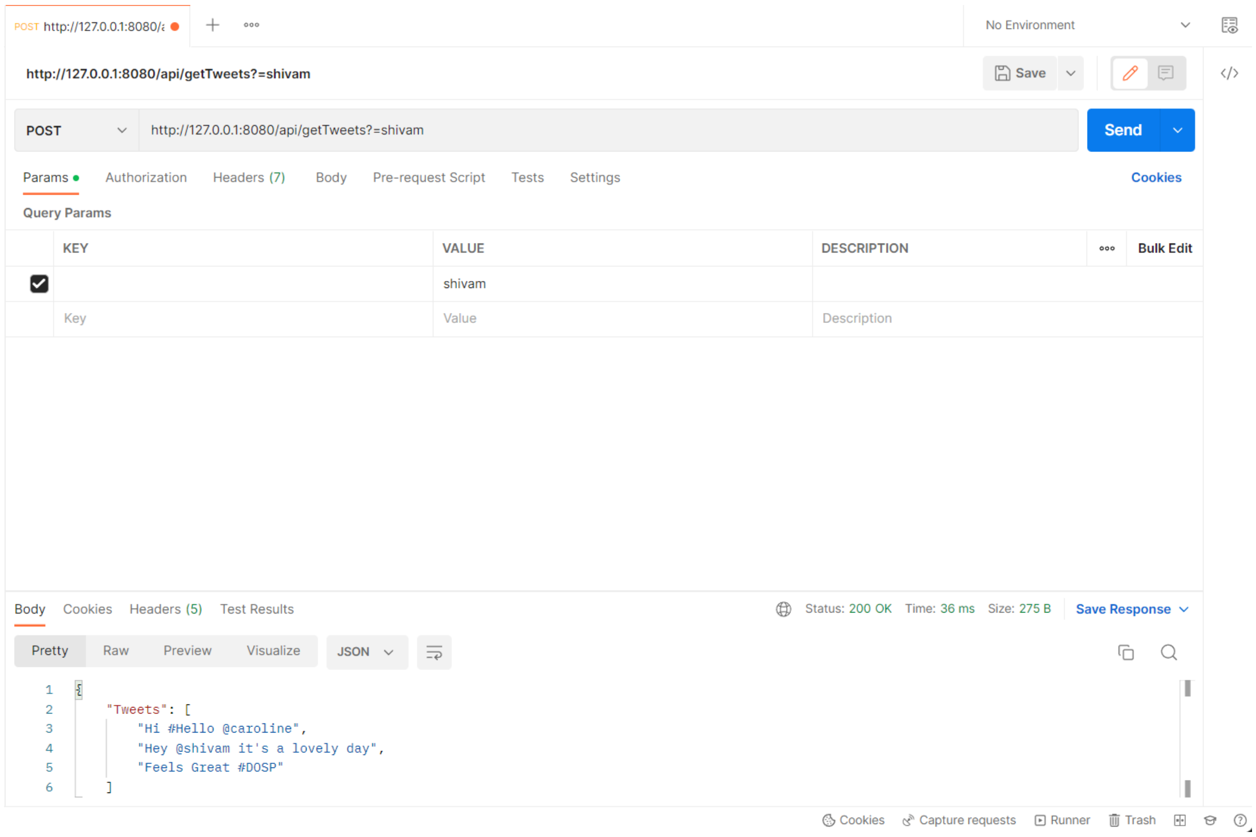
http://127.0.0.1:8080/api/tweet?=Hi %23Hello @caroline

POST http://127.0.0.1:8080/api/tweet?=Hi %23Hello @caroline Send

KEY	VALUE	DESCRIPTION
✓	Hi %23Hello @caroline	

Status: 200 OK Time: 17 ms Size: 198 B

```
{
    "Reponse": "Tweet is Live!"
}
```



http://127.0.0.1:8080/api/getTweets?=shivam

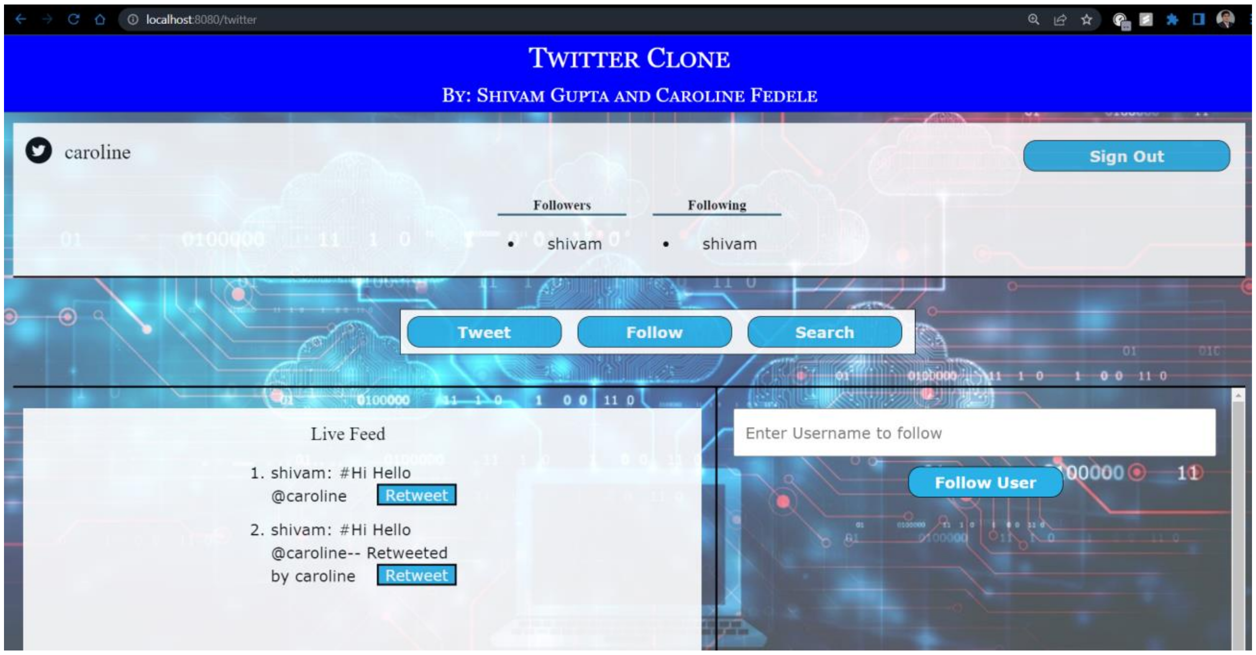
POST http://127.0.0.1:8080/api/getTweets?=shivam Send

KEY	VALUE	DESCRIPTION
✓	shivam	

Status: 200 OK Time: 36 ms Size: 275 B

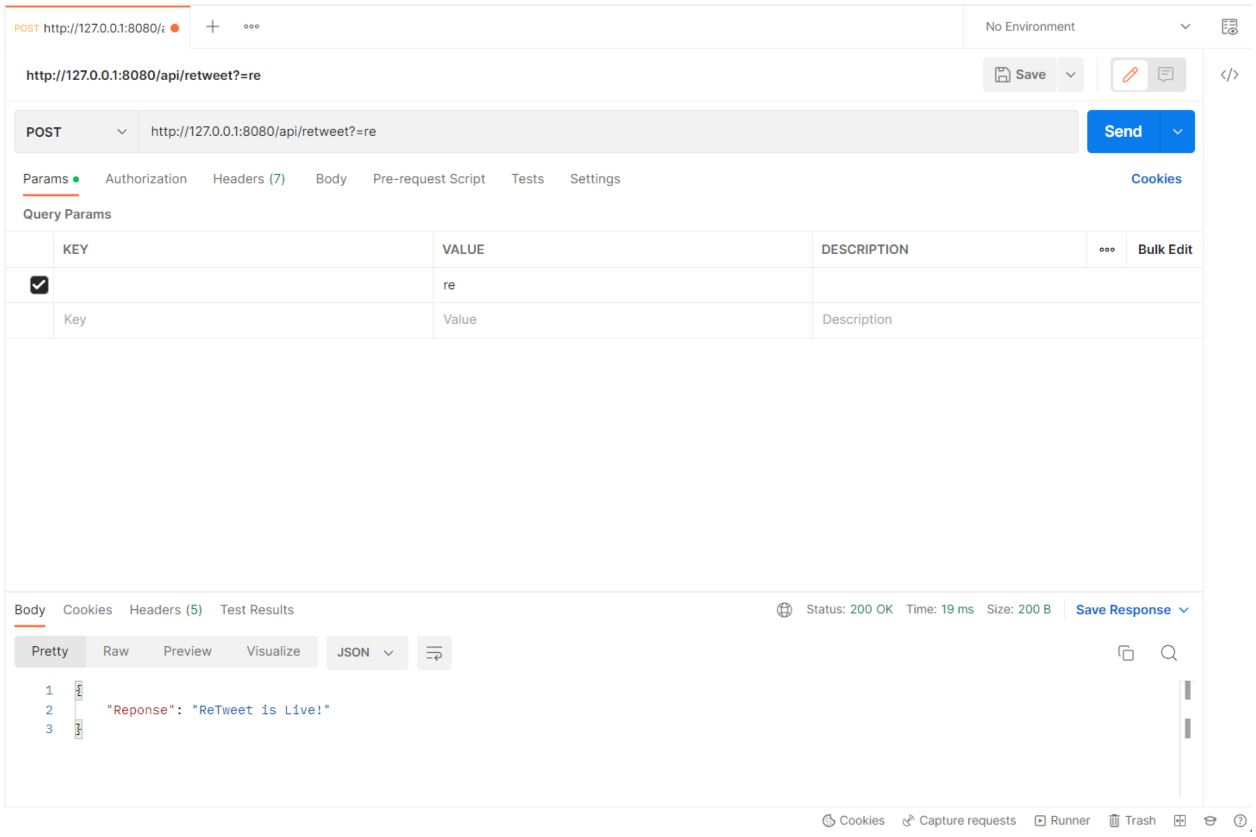
```
{
    "Tweets": [
        "Hi #Hello @caroline",
        "Hey @shivam it's a lovely day",
        "Feels Great #DOSP"
    ]
```

- Re-tweet
    - This is done using the **retweet()** function and updates the user's list of tweets as well as the followers of this user with the new tweet.



Live Feed

1. shivam: #Hi Hello @caroline Retweet
2. shivam: #Hi Hello @caroline-- Retweeted by caroline Retweet



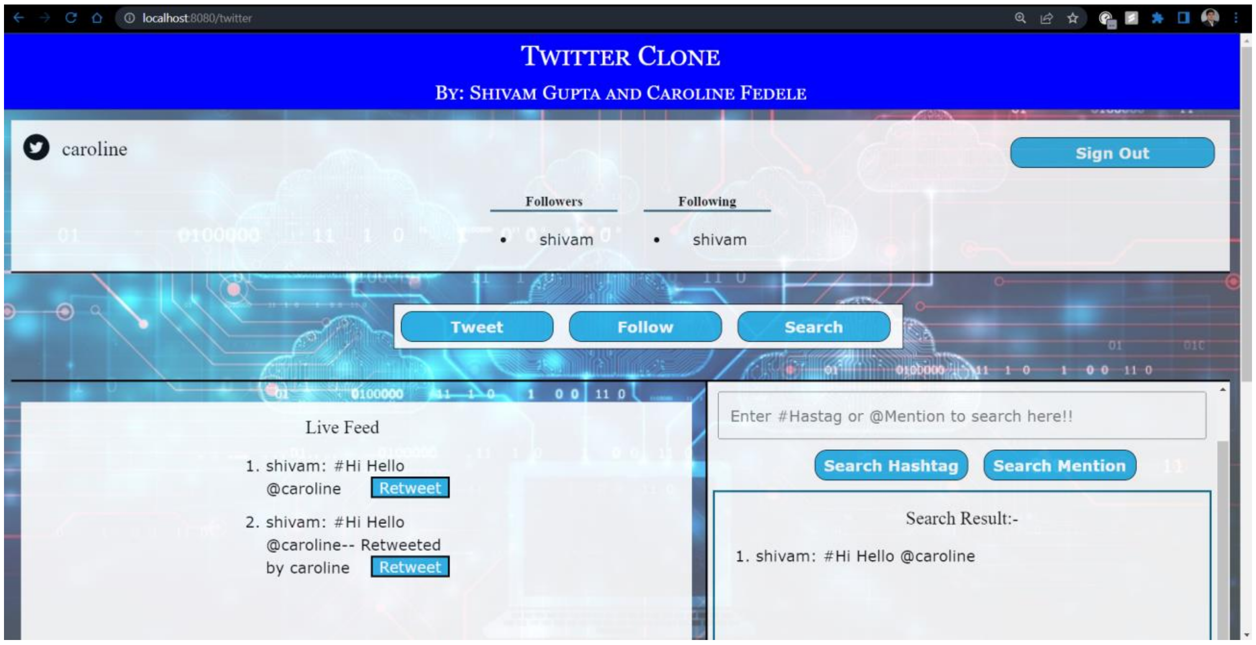
POST http://127.0.0.1:8080/api/retweet?=re

KEY	VALUE	DESCRIPTION
✓	re	
Key	Value	Description

Status: 200 OK Time: 19 ms Size: 200 B

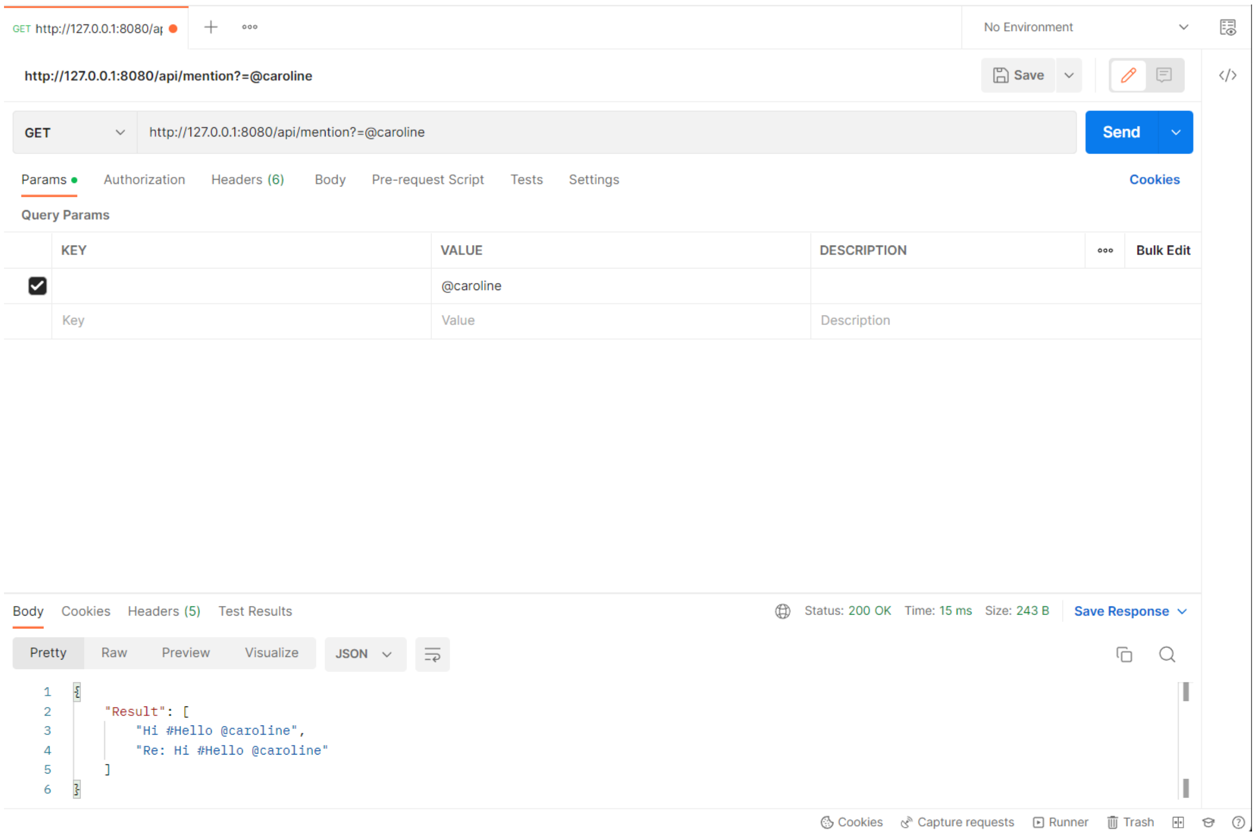
```
{
    "Reponse": "ReTweet is Live!"
}
```

- Mention
  - This is done using the **mention()** function which first checks that the given username exists among the tweets and then displays all the tweets in which the user is mentioned who queried.



1. shivam: #Hi Hello @caroline
2. shivam: #Hi Hello @caroline-- Retweeted by caroline

Search Result:-
1. shivam: #Hi Hello @caroline



http://127.0.0.1:8080/api/mention?=@caroline

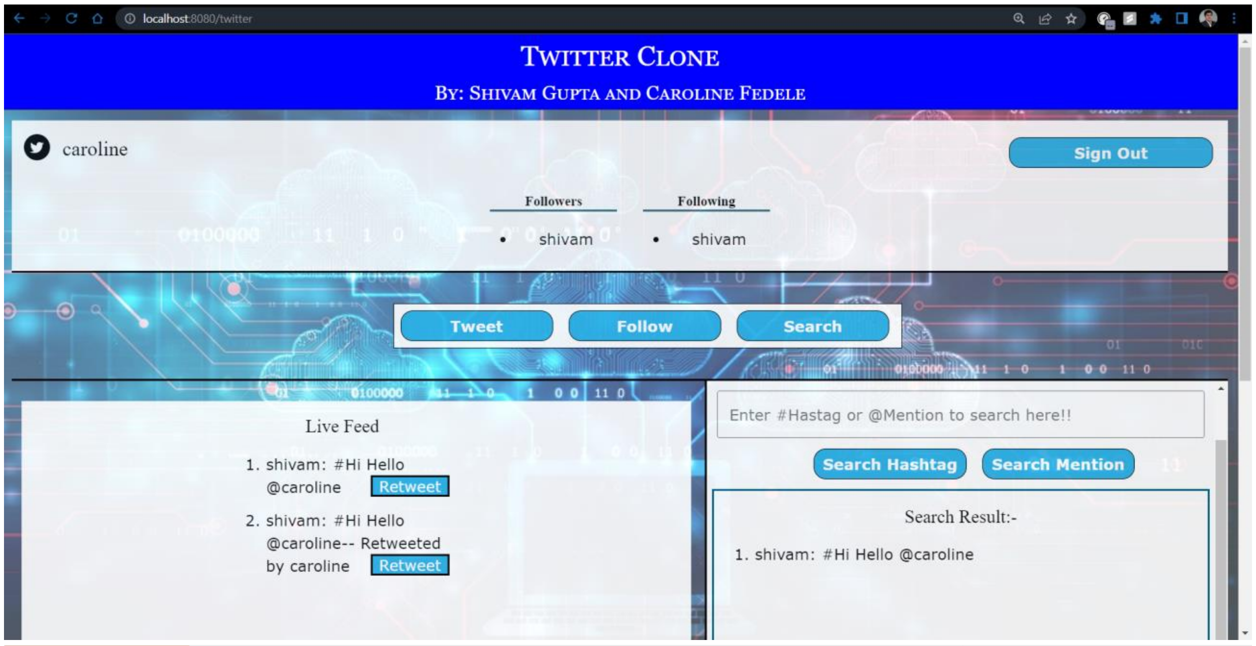
GET http://127.0.0.1:8080/api/mention?=@caroline

KEY	VALUE	DESCRIPTION
	@caroline	
Key	Value	Description

Status: 200 OK Time: 15 ms Size: 243 B

```
{
    "Result": [
        "Hi #Hello @caroline",
        "Re: Hi #Hello @caroline"
    ]
}
```

- Search Tweets for Hashtags
  - This is done in **search(Query)** and allows users to search by hashtag or substring of a hashtag and returns tweets that match the criteria.



- shivam

- shivam

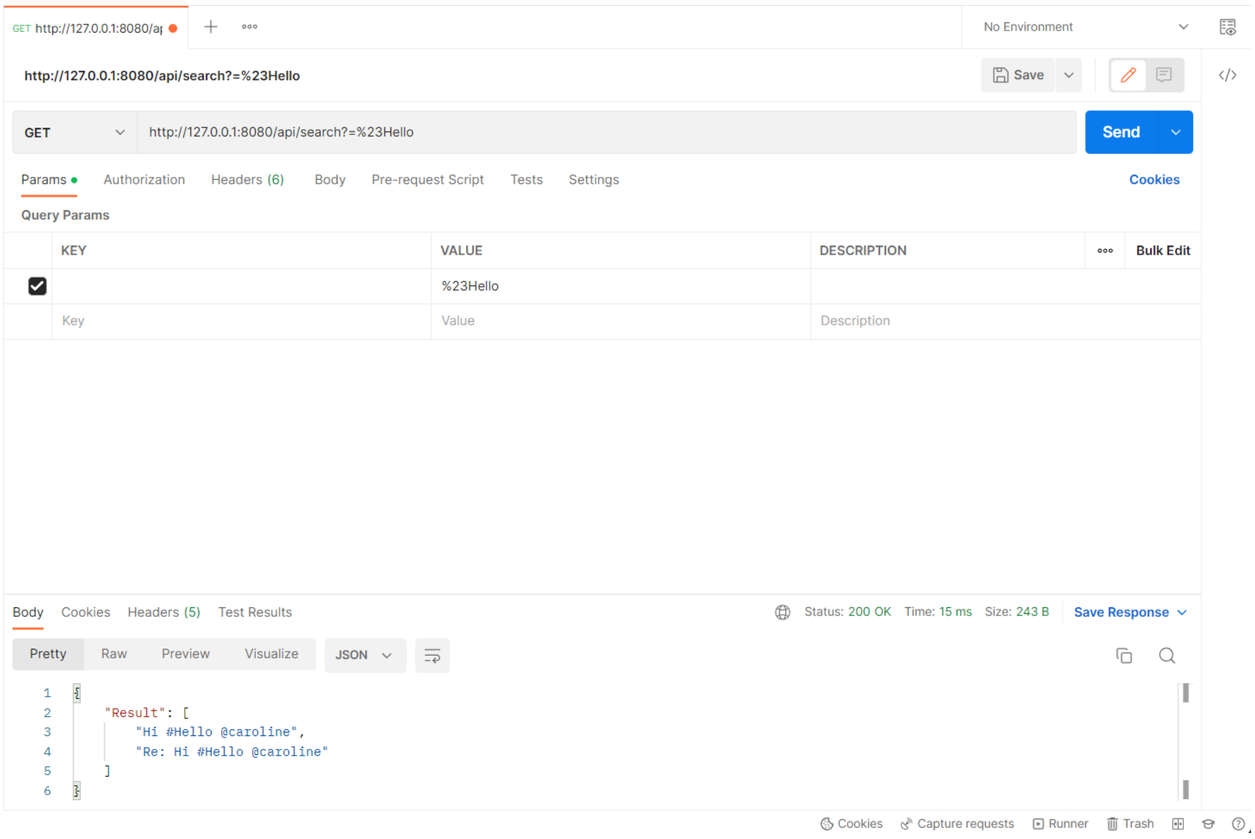
Live Feed

1. shivam: #Hi Hello @caroline
2. shivam: #Hi Hello @caroline-- Retweeted by caroline

Enter #Hastag or @Mention to search here!!

Search Result:-

1. shivam: #Hi Hello @caroline



http://127.0.0.1:8080/api/search?=%23Hello

GET http://127.0.0.1:8080/api/search?=%23Hello

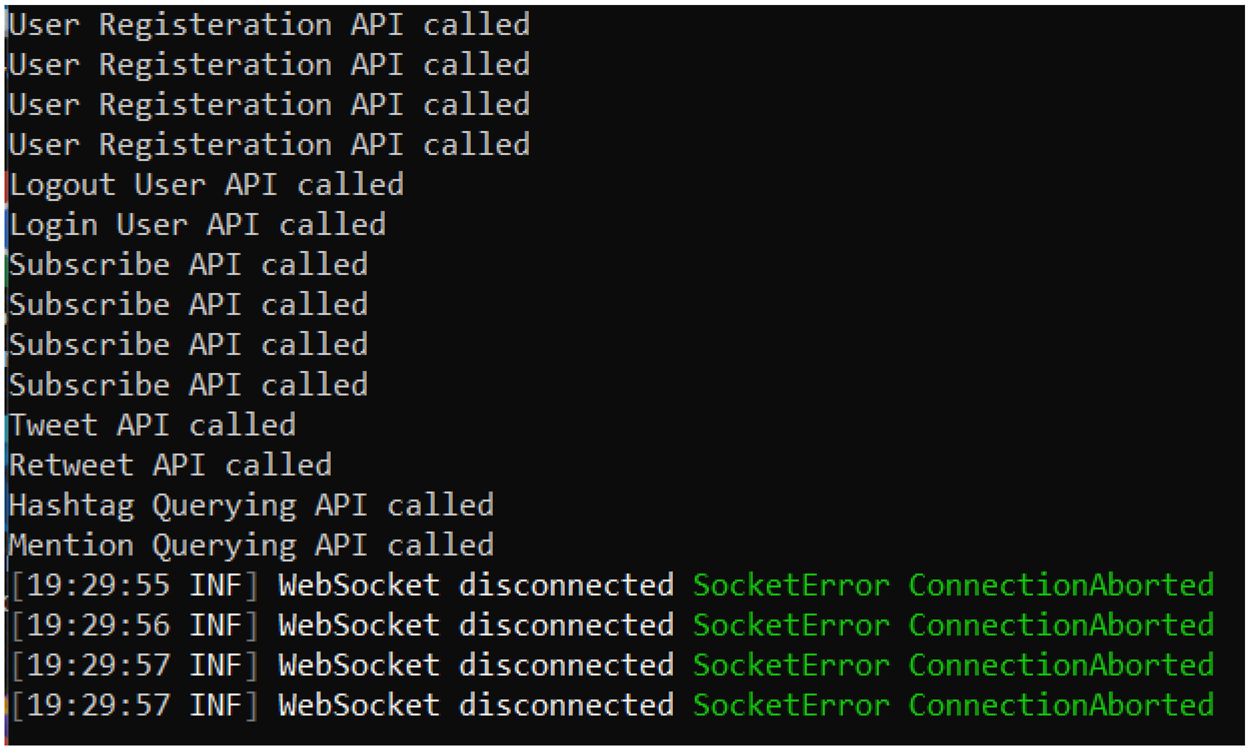
Query Params

	KEY	VALUE	DESCRIPTION
✓		%23Hello	
	Key	Value	Description

Status: 200 OK Time: 15 ms Size: 243 B

```
{
    "Result": [
        "Hi #Hello @caroline",
        "Re: Hi #Hello @caroline"
    ]
}
```

- Server keeps on updating the API's used



```
User Registeration API called
User Registeration API called
User Registeration API called
User Registeration API called
Logout User API called
Login User API called
Subscribe API called
Subscribe API called
Subscribe API called
Subscribe API called
Tweet API called
Retweet API called
Hashtag Querying API called
Mention Querying API called
[19:29:55 INF] WebSocket disconnected SocketError ConnectionAborted
[19:29:56 INF] WebSocket disconnected SocketError ConnectionAborted
[19:29:57 INF] WebSocket disconnected SocketError ConnectionAborted
[19:29:57 INF] WebSocket disconnected SocketError ConnectionAborted
```