DOS Project 4 Part 1:

TEAM MEMBERS:
Name: Poornima Kumar
UFID: 5684-4925

Name: Harika Bukkapattanam
UFID: 3683-6895

PROGRAM EXECUTION:

1. **Compiling the project**:
   mix escript.build

2. **Running the project:**

   A. **Running the Twitter simulation**
   escript project4part1final no_of_clients
   Eg.
   To start simulation of the project with 1000 clients, run the following command:
   escript project4part1final 1000

   B. **Manually testing the project:**
   To test each client manually, run the following commands:

   Execution steps:

   iex -S mix

   1. Starting a server:
      Please make sure that the name passed is “Tweeter”
      Tweeter.start_link(“Tweeter”)

   2. Registering users:
      Open different terminals or on the same terminal, create multiple users by executing the following commands multiple times with different client_name.
      Client.register(client_name)
      Eg. Client.register(“Client1”)

   3. Subscribing to other users:
      Client.subscribe(your_name, name_of_client_to_subscribe)
      Eg. Client.subscribe(“Client1”, “Client2”)

   4. Sending tweets:
      Client.send_tweet(name_of_the_client_sending_tweet, tweet)
      Eg. Client.send_tweet(“Client1”, “#DOS project #part1 @Client2”)

   5. Retweet:

For retweeting, the user must pass the tweet_id which he wants to retweet. This tweet_id can be retrieved from the console, where the tweet has been displayed along with the tweet_id.
Client.retweet(name_of_the_client_retweeting, tweet_id)
Eg. Client.retweet(“Client1”, 3)

6. Disconnecting a user:
   Client.stop(name_of_client_to_be_disconnected)
   Eg. Client.stop(“Client1”)

7. Starting the disconnected client again:
   Client.start_link(name_of_disconnected_client)
   Eg. Client.start_link(“Client1”)
   Make sure to pass the name correctly to start the disconnected client

8. Searching “my mentions”:
   A client can query the tweets in which he has been mentioned
   Client.search_my_mentions(name_of_the_client)
   Eg. Client.search_my_mentions(“Client1”)

9. Searching for tweets with specific hashtags:
   Client.search_hashtag(name_of_the_client, hashtag)
   Eg. Client.search_hashtag(“Client1”, “#DOS”)

10. Querying the tweets subscribed to:
    Client.get_user_tweets(name_of_the_client)
    Eg. Client.get_user_tweets(“Client1)

IMPLEMENTATION:

Modules:

GenerateTweet.ex

This module implements the functionality to return a random tweet containing alphanumeric characters, mentions and hashtags.

Project4part1simulator.ex

The simulator can be started using the commands as specified above. The module achieves the following tasks:

Registers a specific number of clients which is passed as the input parameter to the program.

It subscribes each of the client processes to specific number of other clients to simulate a Zipf distribution.

Every client sends tweets according to the Zipf distribution (ie. Clients having more number of subscribers send more tweets)

The simulator picks random clients and retweet their tweets which are received by the subscribed users.

The module also simulates disconnection of 25% of the connected users, simultaneously continuing the process of sending tweets by the existing connected users. It then restarts the disconnected users. The disconnected users on restarting displays all the tweets received when they were disconnected and then starts tweeting again.

Tweeter.ex

The Tweeter.ex module acts as the engine which distributes the tweets. All the client requests like registration, sending tweet, retweet, disconnection, reconnection and querying of tweets, hashtags and my_mentions, are handled by this module which is also a Genserver.

In order to start the project, the Tweeter Genserver needs to be started first. The name of the server is registered which is used by the other users to connect to the server.

On starting the Genserver, it creates :ets tables as mentioned below which are accessible and maintained by the Twitter engine (ie. Tweeter.ex module):
List of :ets tables:

:ets Table Name	**Key**	**Value**	**Description**
tweet	tweet_id	[tweet, hashtags, mentions, sender]	Maintains a list of tweets and its original sender.
subscribers	subscribed_to_process_name	subscribed_from_process_name	Maintains a list of all the process that has subscribed to the current process
following	subscribed_to_process_name	subscribed_from_process_name	Maintains a list of all the processes which the current process is following.
Hashtags	hashtag	tweet_id	Associates the hashtags with the tweets. The table is used for querying tweets based on hashtags.
My_mentions	mentioned_process_name	tweet_id	Maintains a list of all the processes who have been mentioned in different tweets.
User_tweets	process_name	sent_tweet_id	Maintains a list of all the tweets by the current user.

User_offline_tweets	process_name	received_offline_tweet_id	Stores all the tweets received by a client which is disconnected. The table is used to display all the tweets that were tweeted by its subscribers when the client was disconnected.

Client.ex

Each client is a process which make the API calls to perform a specific action. The different tasks are performed by making the following API calls:

Method Name	**Description**
register(name)	Used to register the client when it tries to connect for the first time.
start_link(name)	Restarts the client which was disconnected.
stop(name)	Disconnects the client
subscribe(name, process_name)	Current client can subscribe to any other client by specifying the name of the other client.
send_tweet(name, tweet)	Used to send tweets to all the subscribers.
search_hashtag(name, hash_tag)	Used to query the tweets having a specific hashtag.
search_my_mentions(name)	Used to query the tweets in which the current user is mentioned.
get_user_tweets(name)	Used to query all the tweets posted by the current user
get_user_retweets(name)	Queries all the tweets that has been retweeted by the current user.
retweet(name, tweet_id)	Used to retweet a tweet which the current user has received from its subscriber.

PERFORMANCE:

1. The maximum number of clients with which the program has been tested is 100000. The registration for the same took 12024ms

2. The average number of tweets/second sent and received by subscribed clients is 359907

Number of clients	Percentage of clients disconnected	Time for registration	Time for subscribing according to Zipf Distribution	Time for tweeting according to Zipf distribution	Total tweets	Time for tweeting	Time for disconnection(in ms)	Time for reconnection	Time for retweeting 1000 tweets with fixed tweet
100	25%	16	15	495	495	8	0	3	5482
500	25%	47	281	2495	2495	28	2	12	4052
1000	25%	78	869	4995	4995	58	8	31	4483
5000	25%	341	21768	3192	24995	326	43	180	7925
10000	25%	709	101237	9104	49995	621	133	4107	10669

NOTES:

1. Used the github repository for generating random strings: https://gist.github.com/ahmadshah/8d978bbc550128cca12dd917a09ddfb7

2. Process.sleep(100000) has been used at the end of the Simulation.ex module so that all the running processes are completed. If the statement "Restarted users sending tweets completion time" is displayed on the console then the program can be terminated by pressing "Ctrl + C"

3. The above values are without the IO.puts statement for the live tweets and retweets.

4. Also, the time may vary according to the tweet method used for generation for random tweets. For higher number of clients, tweets from a defined list of tweets is selected. For less clients, random tweets is generated using the GenerateTweet module.\

5. To allow remote node connections, we have made use of the command :inet.getif and :inet.ntoa to get the ip addresses of the client and tweeter engine nodes.