Distributed Operating System Principles

(Fall 2020)

Project 4 Part 2

Venkata Vikramaditya Varma Kunaparaju 1537-8659

Prajan Tikayyolla 6690-9943

Demo Video Link

Project Description:

In this project we have used Suave web framework to implement a WebSocket interface to our part I implementation.

The twitter clone we have implemented supports register an account, login using username and password, send tweets, subscribe to user's tweets, do retweets of user's tweets, and deliver all these functions live without querying for logged in users.

We have designed a JSON based API that represents all messages and their replies (including errors)

How to Run the Project:

First after opening the Twitter Folder we need to run **dotnet build** command in the terminal.

Once this command is executed, before starting the running of the project, make sure that following .dll files are included in the twitter folder.

#r ".\\bin\\Debug\\netcoreapp3.1\\Suave.dll"

#r ".\\bin\\Debug\\netcoreapp3.1\\FSharp.Json.dll"

#r ".\\bin\\Debug\\netcoreapp3.1\\Newtonsoft.Json.FSharp.dll"

#r ".\\bin\\Debug\\netcoreapp3.1\\Twitter.dll" (To share Common Data Types)

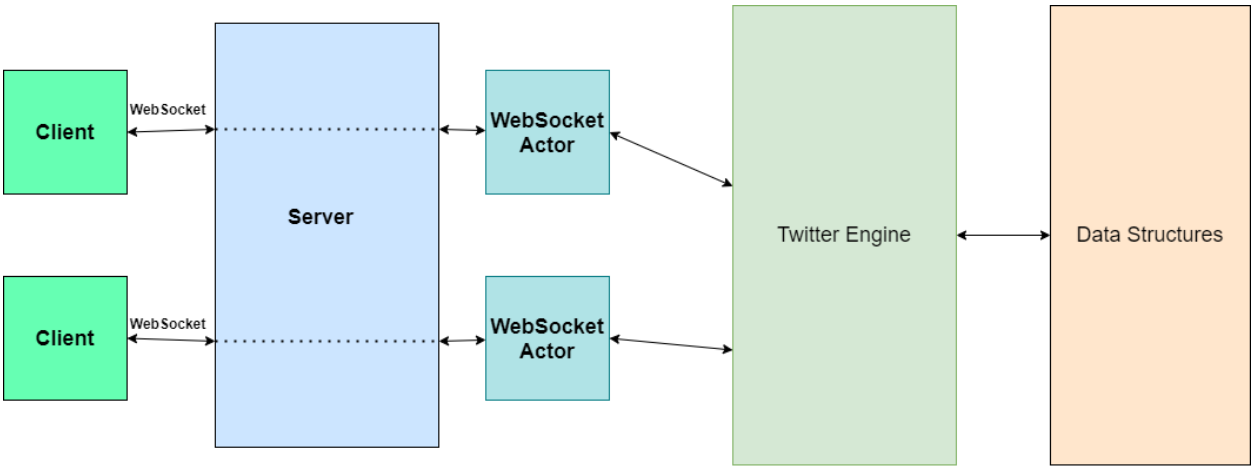
Then we need to start the twitter engine server using the following command.

dotnet fsi --langversion:preview Server.fsx

After running the server, we need to run the clients in another terminals. The following command should be used to run clients in another terminal.

dotnet fsi --langversion:preview SuaveClient.fsx

Twitter Clone Architecture:



Twitter Client

Implementation Details:

The server is based on Suave web framework and once it is triggered it will start server on localhost:8080 and will be waiting to get connection requests from web socket clients. Once the connection is made it has the web socket and context details. For each web socket we will be creating an actor and as socket receives the message the message is forwarded to its actor and then actor reads the received JSON object and based on request URL it deserialize the body into corresponding type record and based on request URL it also forwards the deserialized object to corresponding actors (subscribe, tweet, retweet, follow, …) which in turn performs the requested task and sends the response back to WebSocket actor. The response message of 3 types Response, QueryResponse and LiveTweet it then serializes them and send to corresponding WebSocket client. The WebSocket actor in turn puts into socket channel for client to receive.

In client we are using ClientWebSocket class for dotnet. Client initially creates a web socket connection to server URL. Once the socket is created, we take input from console and serialize it and send it to server along with this task there will be another task running parallelly on client side for receiving response. As it receives the responses it deserializes them and formats and prints the response on the console.

Request JSON with headers example:

RequestType: POST

RequestAPI: Tweet

Body: {

"UserId": "1",

"Content": "hi#dos",

"TimeStamp": 1608159276826

}

Response JSON example:

{ Status = 200

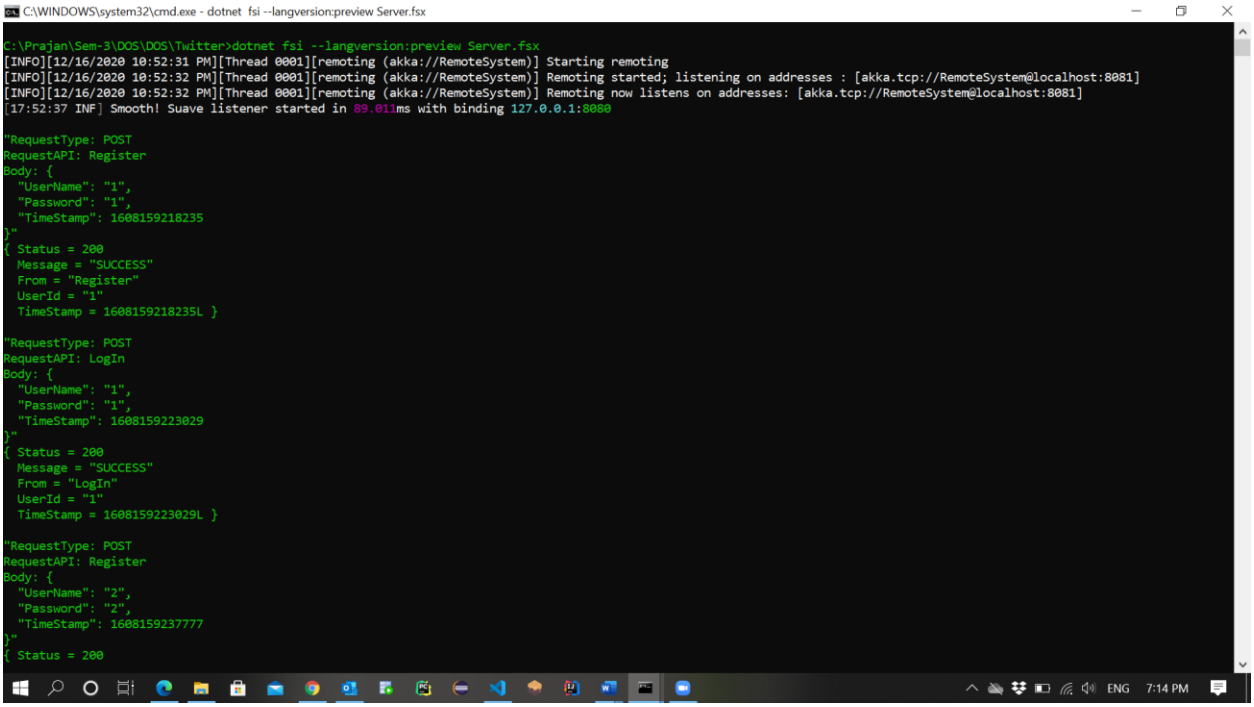
Message = "SUCCESS"

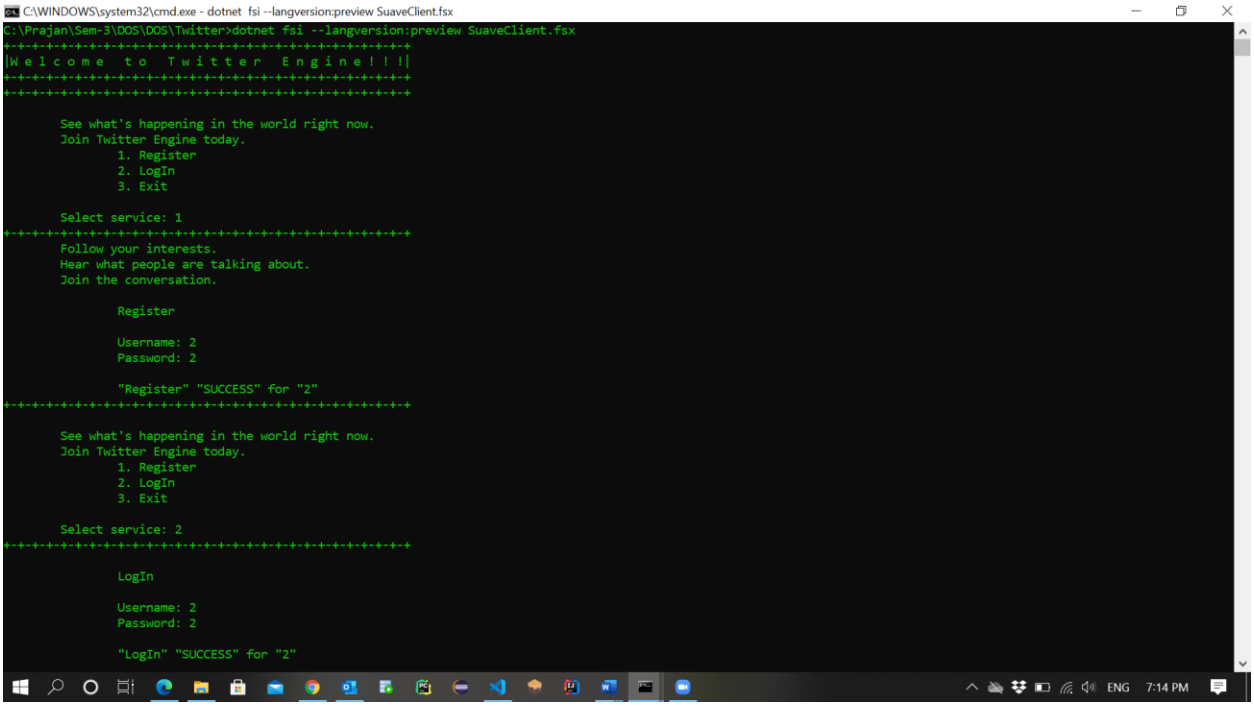
From = "Tweet"

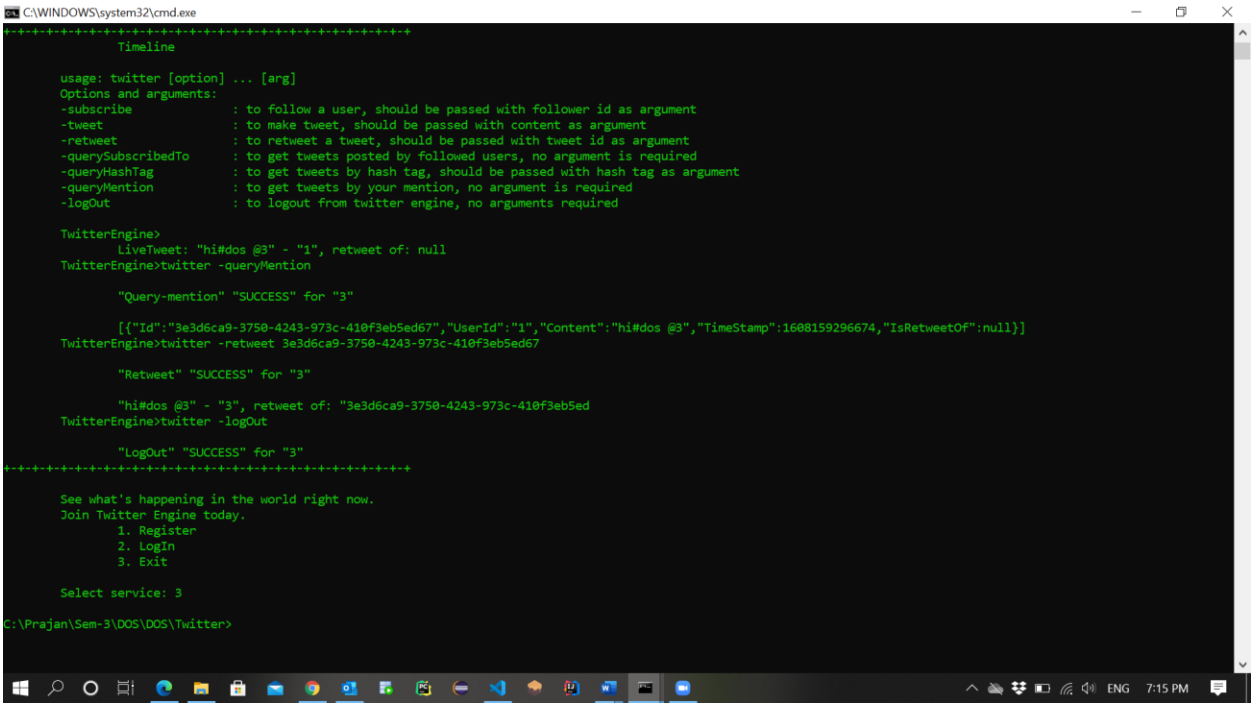
UserId = "1"

TimeStamp = 1608159276826L }

Some Snapshots:




C:\WINDOWS\system32\cmd.exe - dotnet fsi --langversion:preview SuaveClient.fsx
C:\Prajan\Sem-3\DOS\DOS\Twitter>dotnet fsi --langversion:preview SuaveClient.fsx
+-+
|W e l c o m e t o T w i t t e r E n g i n e ! ! !|
+-+
+-+
See what's happening in the world right now.
Join Twitter Engine today.
1. Register
2. LogIn
3. Exit
Select service: 1
+-+
Follow your interests.
Hear what people are talking about.
Join the conversation.
Register
Username: 2
Password: 2
"Register" "SUCCESS" for "2"
+-+
See what's happening in the world right now.
Join Twitter Engine today.
1. Register
2. LogIn
3. Exit
Select service: 2
+-+
LogIn
Username: 2
Password: 2
"LogIn" "SUCCESS" for "2"
ENG 7:14 PM


C:\WINDOWS\system32\cmd.exe
+-+
Timeline
usage: twitter [option] ... [arg]
Options and arguments:
-subscribe : to follow a user, should be passed with follower id as argument
-tweet : to make tweet, should be passed with content as argument
-retweet : to retweet a tweet, should be passed with tweet id as argument
-querySubscribedTo : to get tweets posted by followed users, no argument is required
-queryHashTag : to get tweets by hash tag, should be passed with hash tag as argument
-queryMention : to get tweets by your mention, no argument is required
-logOut : to logout from twitter engine, no arguments required
TwitterEngine>
LiveTweet: "hi#dos @3" - "1", retweet of: null
TwitterEngine>twitter -queryMention
"Query-mention" "SUCCESS" for "3"
[{"Id":"3e3d6ca9-3750-4243-973c-410f3eb5ed67","UserId":"1","Content":"hi#dos @3","TimeStamp":1608159296674,"IsRetweetOf":null}]
TwitterEngine>twitter -retweet 3e3d6ca9-3750-4243-973c-410f3eb5ed67
"Retweet" "SUCCESS" for "3"
"hi#dos @3" - "3", retweet of: "3e3d6ca9-3750-4243-973c-410f3eb5ed
TwitterEngine>twitter -logOut
"LogOut" "SUCCESS" for "3"
+-+
See what's happening in the world right now.
Join Twitter Engine today.
1. Register
2. LogIn
3. Exit
Select service: 3
C:\Prajan\Sem-3\DOS\DOS\Twitter>
ENG 7:15 PM