The Twitter Clone Project

Distributed Systems Project 4.2

Team Member:

BHANU PRAKASH REDDY PALAGATI: 2623-3738; Akshith Sagar: 8788-8206

Project Statement:

A Twitter clone should be implemented which supports *registering, login, subscribing, tweet, retweet, direct messages, query, logout* features. A Client and Server mechanism should be implemented with the help of web sockets and provide a video explaining the project.

How to Run Me:

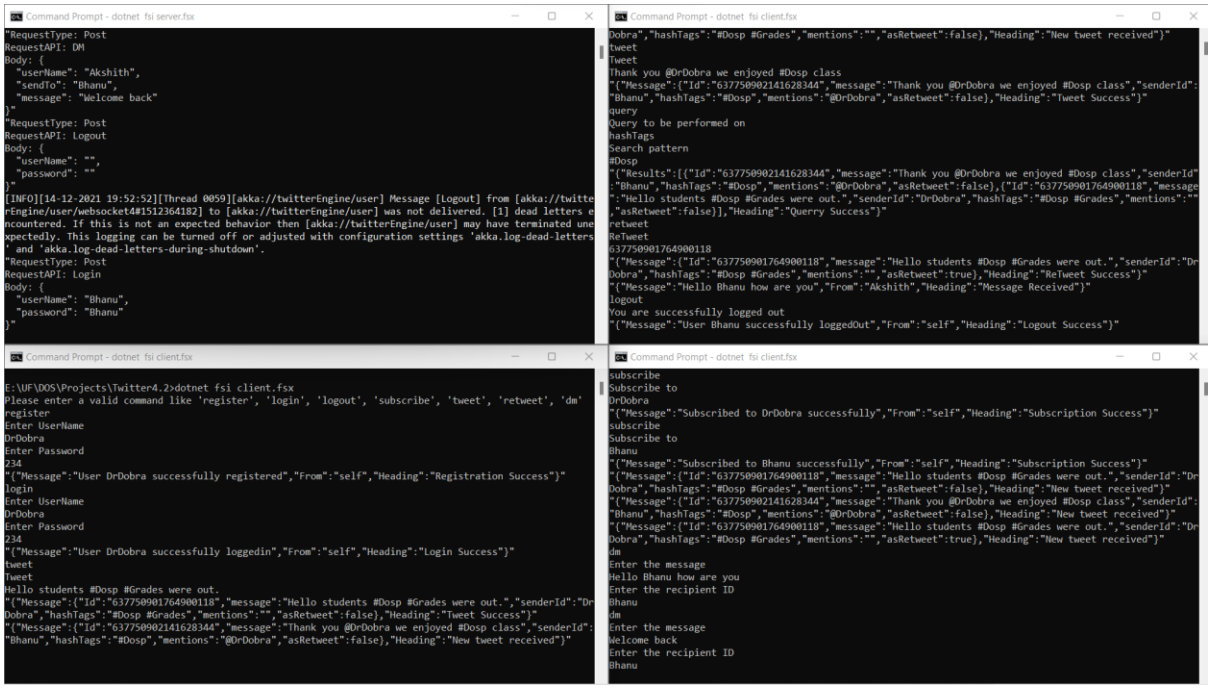
This F# project has three files - server.fsx, client.fsx, and dataDefnitions.fs.

First, you have to run server.fsx because this acts as a server.

"*dotnet fsi server.fsx*" will run the server

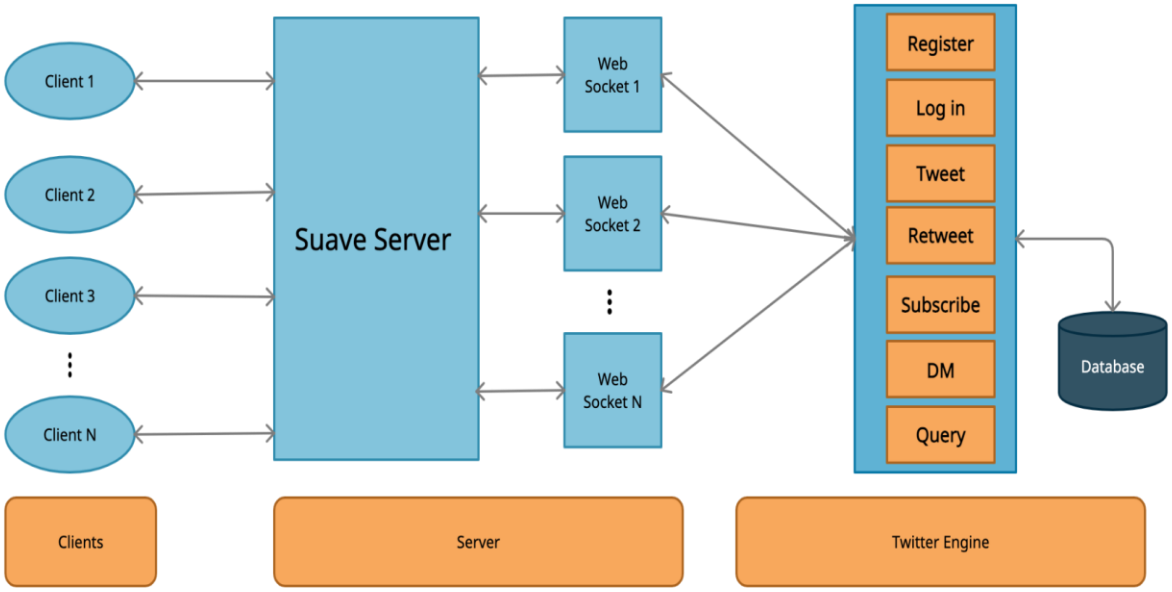
Then you have to run as many clients as you want to simulate the behavior

"*dotnet fsi client.fsx*" this will run a client.



A running example screenshot with one server and three clients. The top left one is the server.

Engine and Simulator Interaction Architecture:



How Twitter Clone Implemented:

Upon running the server, a Suave instance will be started listening in the localhost:8080 port. Then we can start as many clients as we want with each such client a web socket handshake is made. With the help of the Web Socket connection, we can open the connection between two instances for a long time on the other hand the REST API is just a request-response one. On each web socket connection, a web actor will be generated this helps us to process the requests by user actors and complete the connection. Finally, when there is a request from the client to the server the web socket identifies the request name based on the request the data will be deserialized. Then the deserialized data will be provided to the respective actor with the respective verb. Once the user actor processes the data it will provide the appropriate response to the web socket actor with provides this response back to the client.

From the client side, we have communicated with the server using the request name and request verb. For example, “Login Post” will send the login post request to the server. Since there is a web socket it allows the server to directly communicate with the client.

API Table:

API End Points	Request/Response
Register	**{userName: string; password: string}**
Login	**{userName: string; password: string}**
Subscribe	**{userName: string; subscriberId: string}**

Tweet	**{Id: string; message: string; senderId: string; hashTags: string; mentions: string; asRetweet: bool}**
ReTweet	**{tweetId: string; userName: string}**
DM	**{userName: string; sendTo: string; message: string}**
Query	**{searchOn: string; searchWith: string; userName: string}**
LogOut	**{userName: string; password: string}**
RegisterSuccess	**{Message: string; From: string; Heading: string}**
LoginSuccess	**{Message: string; From: string; Heading: string}**
FollowedSuccess	**{Message: string; From: string; Heading: string}**
SubscriptionSuccess	**{Message: string; From: string; Heading: string}**
TweetSuccess	**{Message: tweet; Heading: string}**
QuerySuccess	**{Results: seq<tweet>; Heading: string}**
ReTweetSuccess	**{Message: tweet; Heading: string}**
DMSuccess	**{Message: string; From: string; Heading: string}**
ErrorReponse	**{Message: string; Heading: string}**

What are all Achieved In this Project:

- Register
- Login
- Tweet with mentions and hashtags
- ReTweet using tweetId
- Tweet Notifications to Subscribers and mentioned users
- DM
- Query on message, mentions, and hashtags
- Subscription
- Logout
- Implemented JSON-based request-response.
- Implemented Web Socket interface using Suave framework.
- A demo video is created and uploaded to YouTube.

https://www.youtube.com/watch?v=XkLegQuhqhY