COP5615 Project-4 - Part 2 - Report

Venkata Sai Samanth Kommi - UFID 2793-5597
Venkata Maneesh Reddy Konkala - UFID 7637-9205

December 15 2021

1 Twitter Clone with GUI

1.1 Problem Statement

Extend the twitter-clone implemented before by constructing a GUI that allows users to follow other users, search tweets and make tweet. Implement this using web sockets and JSON based API for the twitter clone implemented using Akka actor model in F#.

1.2 Process

Web sockets were added to the twitter clone engine which provides the ability to maintain the session of a user. By using multiple JSON based API endpoints, functionalities like Follow a user, Post a Tweet, Query about tweets with specified Hash tags and or with specific user mentioned. First the user must be registered in then the user can use the above specified functionalities. This whole functionality was liked to the twitter clone engine by using Web Sharper implementation provided by Suave.

2 Working Unit

Implemented a simple user interface through which the user can perform the actions like Tweet, Retweet, Hash Tagging, Mentioning other users, subscribing to other users, searching for tweets with required hashtag and searching for tweets in which an user is mentioned. We have implemented a client server architecture. The two parts are implemented as below,

2.1 Server

- The server exposes API endpoints on which the client can send messages (actions) to the server.
- The server receives requests in JSON format and gives responses in return as JSON format.

- The server returns response in the following format,
  {
  userName: string
  service: string
  message: string
  code: string
  }
- UserName is the name of the user that receives the response.
- Service is the type of service (it can be tweet, retweet, follow, hashtag, mention, query, register, login) that has been handled.
- Message is the response to each of the above services. The response for each of the messages is as follows,
  - Tweet: The text of the tweet is in the message. It is returned to the user's subscribers. If a user mentions any other user then it receives the mentioned user.
  - ReTweet: It is the same as tweet except retweet text is added in front of the message.
  - Query: Searches for the tweets with given user name or hashtag ad returns the tweets if found. If not found it returns "No users found" or "No hashtags found" with an error code.
  - Follow: Searches for the given user to subscribe for. If the user is not found it returns "No users found". If found it returns the message subscribed to the given user.
  - Login: If the password is not correct or if the entered user has not been registered it returns "Invalid user name / password". If the credentials are correct. It returns the message "User logged in successfully".
  - Register: If the user has already registered it returns user has been already registered. If the user is not registered it returns the message "User successfully registered".

2.2 Client

- The client consists of user interface through which actions are performed.
- In order to perform any action the users has to first register and then login.
- The client requests are in the following format,
  {
  userName: string,
  value: string,

reqType: string,
}

- The userName is the name of the user that is performing the action.
- The value for various actions would be as follows,
  - Register: The value would be the password of the user trying to register.
  - Login: The value would be the password of the user trying to login.
  - Tweet: The value would be the content of the tweet.
  - Retweet: The value would be the content of the retweet.
  - Subscribe/Follow: The value would be the name of the user subscribing to.
  - Query: The value would be the name of the user or the hashtag to be searched for.
- The reqType is the type of the request made by the user. It can be any of the above actions.
- The logged in user would be getting tweets from the user subscribed to.
- The user would also be notified if the user has been mentioned by any other user.
- After performing the desired actions the user can logout by clicking the logout button.

2.3 Websocket implementation

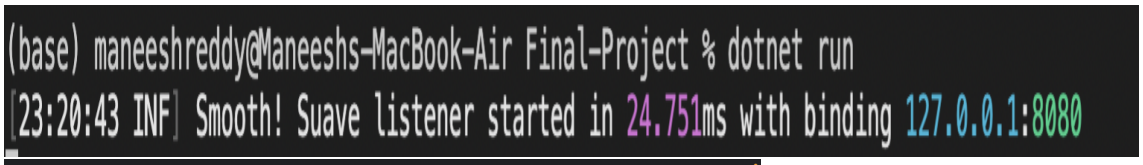
We have implemented the websockets for the functionality of tweets. In real world scenario any number of users can have any number of subscribers. Therefore if we use API requests each time an user does a tweet there would be very high number of requests. Also the tweet has to be sent to every user in its subscribers list. That would be very costly as well as time taking. Websockets help us in this situation by continuously maintaining a connection to the server. The messages can be exchanged freely without performing requests each time a tweet is performed. Websockets are popularly known for their usage in live streaming, online games and many other crucial areas. That is the reason we have implemented the Websockets interface for the functionality of tweets. As soon as the user successfully logins, a webscoket connection is opened with the server. By using the Web Sharper tool available for F# applications, the web sockets were implemented. Inorder to make the implementation easy, Suave framework is used which uses Web Sharper internally to establish the web sockets.

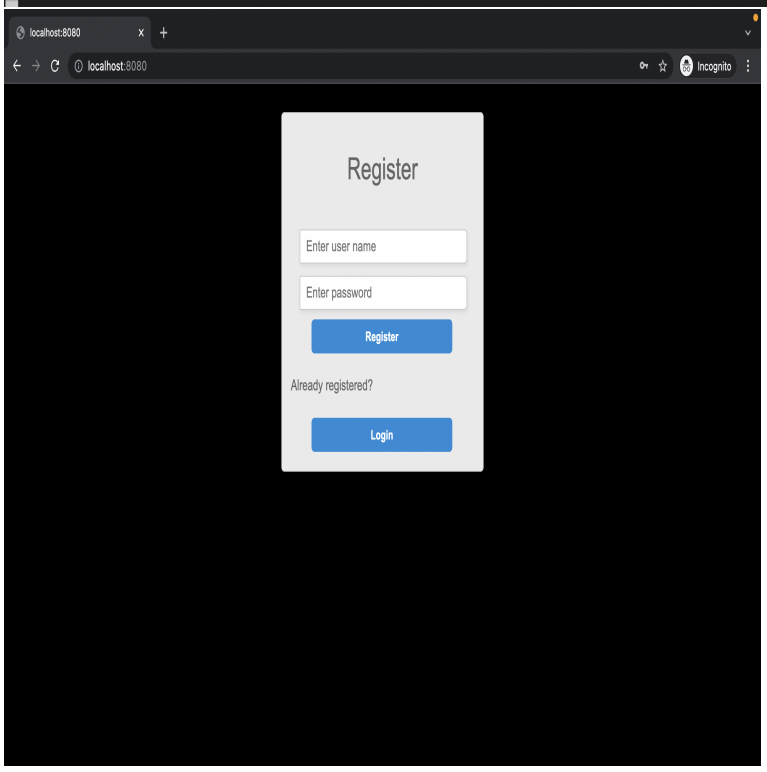
3 Running the project

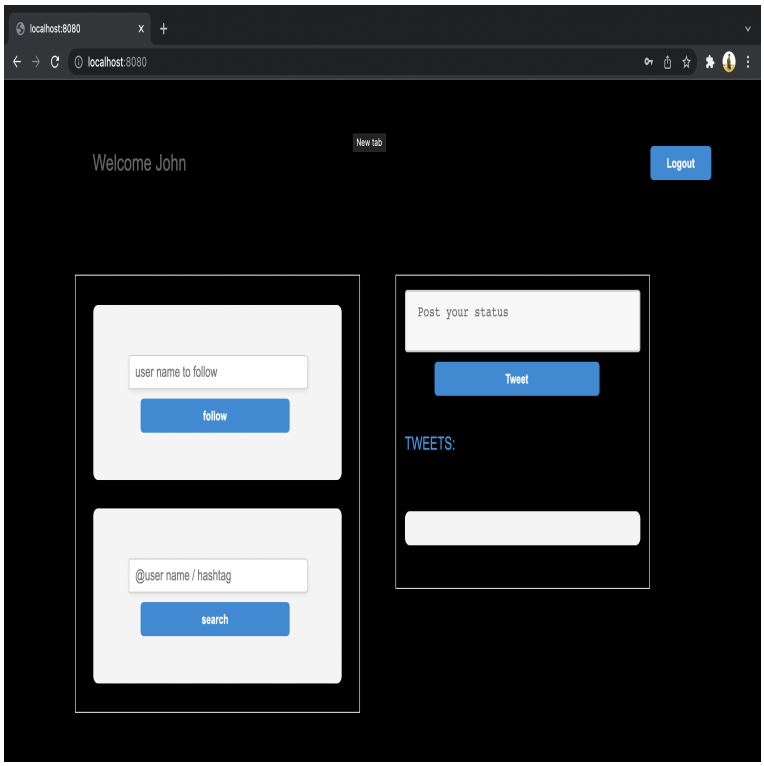
3.1 How to run the project

- Extract the zip file - twitter-clone.zip
- To run the project some dependencies must be installed first. They are:
  - dotnet core sdk with version 6 and above.
  - Suave library of version 2.6.1 and above.
  - Package Fsharp.Json.
  - Akka.Fsharp package.
- Now run the project by executing the following command:
  - **dotnet run**
- The application will start running on the port 8080. It can be accessed by url “localhost:8080” in any browser.

4 Output Screens






localhost:8080
localhost:8080
New tab
Welcome John
Logout
user name to follow
follow
@user name / hashtag
search
Post your status
Tweet
TWEETS: