COP 5615 Distributed Operating Systems Principles and Paradigms
Project 4 Part 2

Names of Members-

Ekleen Kaur 94919717

Youtube link-
https://youtu.be/_WW9QL5G87o

Server-
We have used the .NET server in our project. It helps to build and run web services. The advantage of using .NET is that it supports all kinds of code execution whether the code is stored and executed locally or remotely.

Implementing a web socket interface for real time messaging between client and server-

- We have used the google extension web server.
- We have designed a JSON build API as asked for in the question.
- **JSON API-** Javascript Object Notation.
- It helps in the communication of an application with the servers without using ad-hoc.

- The advantage of using JSON API is that it reduces the amount of requests and amount of data transmitted between servers and clients.
- We have used the web development library of F#, **Suave**, which provides a lightweight web server.
- In this project we have used it to implement a web socket interface for establishing communication between the client and server. There can be many operations happening between the server and client.
- It is quite easy to set up the web sockets using Suave.
- **Web sockets** make it possible to develop a bidirectional communication between the client and the server.
- The advantage of using a web socket is that it does not require to poll the server for a response.
- Suave has many advantages like it is quite efficient. It even builds the applications fast.
- Suave consists of a set of combinators for task composition and changing the path flow of messages.

- Suave is an operating system friendly web server as well as it runs very well on Linux, OS X and Windows.
- We have used **websharper** in our project.
- It is a toolset based on F# for developing web applications and web services.
- The framework used in our project is the **Akka** framework.
- Akka is a set of open source libraries.
- It is used to build scalable systems.
- In this project we have designed the server in order to use web sockets and the client on the other hand dispatches all the messages to the web socket interface.
- It uses the actor model that provides abstraction to an extent and the advantage of the abstraction is that it helps us to design systems which are-
    1. Concurrent
    2. Parallel
    3. Distributed
- The actors used by Akka relieves the developer from problems like locking and thread management which makes the system parallel and concurrent as mentioned above.
- The actors use a message passing mechanism which avoids both locking and blocking as the entire thread of execution is not passed from the sender to the receiver. This helps us to achieve more in less time.
- **Error handling-** Whenever there is an error it gets reflected at the client side and the server side as well. We implicitly fetch the web socket errors and handle them

in our codebase.

- The client gets to see the error at the UserInterface(UI) or the web socket interface for eg- if the password doesn't match the client gets informed by the message **Wrong Credentials**.
- The server on the other hand gets the error at the twitter engine and this gets reflected as a status object of whether there is a success or failure of the requests made.
- The other frameworks that the web socket supports are-
    1. Python
    2. Javascript

The steps to run our project-

- The first step is to build the code at the server side using the command-

  dotnet build
- The next step is to run the code at the server side using the command **dotnet run** which instantiates the server. The server starts then go to the following url http://127.0.0.1:8080/signup to show the project demonstration

- The same steps will be repeated for client side as well firstly we will build the code at client side using the command-

dotnet build

- The last step is to run the code at the client side by using the command-

dotnet run

Note- the client side and the server side need to be executed in different windows or else different terminals first the server runs and then the client one after the other.

Implementation-

About UI-

- The user at first signs himself up and gets added to the database.
- Then whenever the user tries to log in he is asked for the username and password.
- The username and password entered by the user gets checked using a hashing mechanism, SHA-256 and once the password is correct he is redirected to the home page of our twitter system.
- The user is then redirected to the homepage and now if he clicks on the Home tab then a socket gets created which allows him to use all the UI functionalities.
- The user can execute operations like search, follow any user and tweet.
- When the user logs out the user is redirected to the login page.

Bonus-

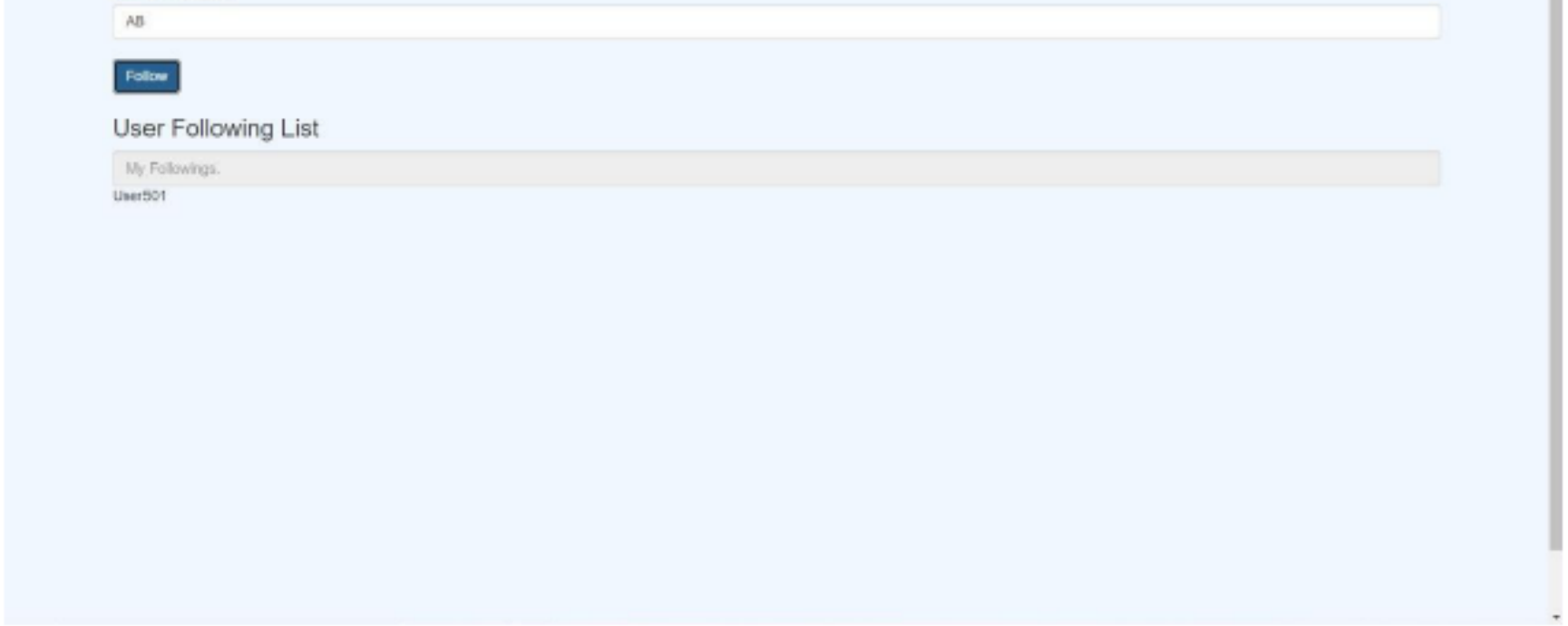
- When the user is at the login page he enters the username and password, and the details are checked using **SHA-256**.
- The above process uses a public key-based authentication method.
- In this method the user needs to enter his public key at the login page.
- The public key then gets authenticated using a challenge based algorithm.
- In this algorithm the system will send a 256-bit challenge.
- The user authentication gets validated after the client successfully solves the challenge.
- Then the client digitally signs it.
- If the public key gets authenticated successfully the client sends a success otherwise an error message.
- We need to keep in mind that the challenge can be used by the system only once.
- When the above authentication is done the user receives a secret key.
- The messages sent by the user are signed by **HMAC(keyed-hash message authentication code or hash-based message authentication code)** which is

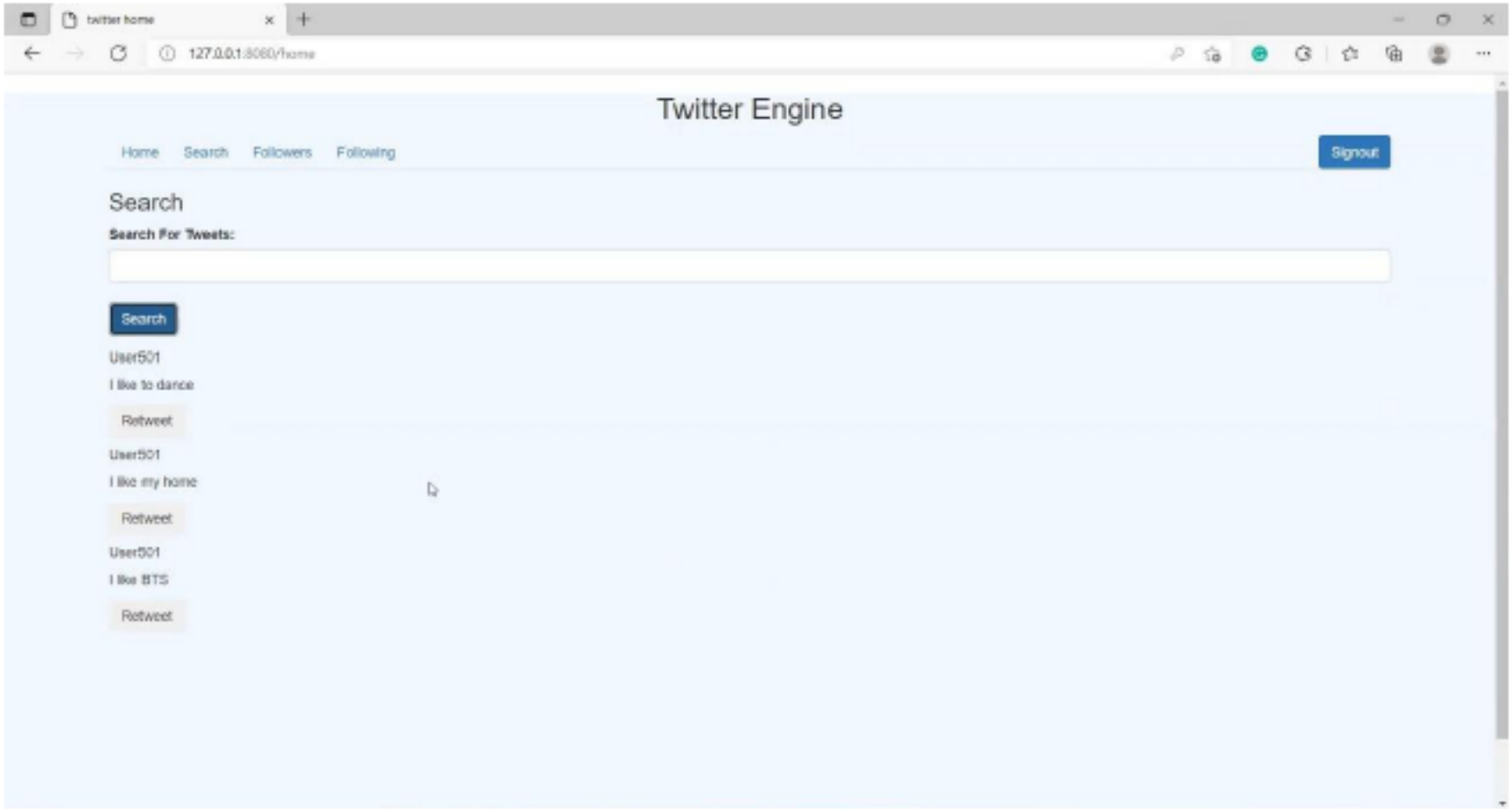
an authentication code and the advantage of using HMAC is that it combines all public keys, private keys and a hash into a mix in such a way that hackers can't hack it.

- Due to time constraints we were not able to achieve the HMAC part of the bonus currently we have executed a hash based user authentication.

Output-

```
t run
Twitter clone of 500 Users Intitated:
95229
95230
Time taken to for circulating 95230 tweets : 751.621000
-------------------------------------------------------------------------------
Time taken to for circulating 9523 retweets : 57.945400
-------------------------------------------------------------------------------
[03:09:35 INF] Smooth! Suave listener started in 48.008ms with binding 127.0.0.1:8080
```


AB
Follow
User Following List
My Followings.
User501


twitter home
127.0.0.1:8080/home
Twitter Engine
Home
Search
Followers
Following
Signout
Search
Search For Tweets:
Search
User501
I like to dance
Retweet
User501
I like my home
Retweet
User501
I like BTS
Retweet