**COP - 5615**

**Distributed Operating System Principles**

Project 4 Part 2 Readme

Name- **Sarthak Kapoor**

Email - [**sarthak.kapoor@ufl.edu**](mailto:sarthak.kapoor@ufl.edu)

Name- **Sachet Nayyar**

Email - [**sachet.nayyar@ufl.edu**](mailto:sachet.nayyar@ufl.edu)

**Instructions for running the program:  
  
Server:**

The project requires dotnet sdk 5 or above. Create a new F# project in VSCode and copy the contents of the 'Program.fs' file into the .fs file of the project. We need to include certain libraries into the project for it to work this can be achieved by executing the following commends via the terminal in VSCode

**dotnet add package Suave --version 2.6.0**

**dotnet add package Newtonsoft.json --version 12.0.3**

**dotnet add package Akka.FSharp**

**dotnet add package FSharp.Data --version 3.3.3**

To start the server use

**dotnet run**

**Client:**

The client is written in python. We need to install the 'websockets' package in order to run it successfully. This can be done by opening a command prompt in the location while the 'Register.py' file is located and running the command

**pip install websockets**

To start a client use

**python register.py**

**Implementation**

**Server:**

The server is built in F# and builds upon the work done in project 4 part 1  
  
The data relating to the users, tweets, relations between users etc. is stored in Data Tables. We have several different tables to store different data and maintain relations between the data.  
  
The communication between client and server takes place throught the use of JSON and websockets.   
  
We formulate all the messages sent over from server to client and vice-versa as JSON objects. We ustilize the FSharp.Data.JsonProvider class to define the structure of our JSON messages. The JSON objects contain properties that define the type of message being sent and any parameters associated with the message, for e.g the message can be of type 'Tweet' and containg the userId of the sender along with the content of the tweet as parameters.  
  
The websocket implementation handles the sending and recieving of the messages. We first determine the type of websocket message sent as in wether it is a regular communication with data in it, or is it a request to close the connection etc. After we have idetified a data message,we re-direct it to a piece of code that determines the specific type of message by parsing the JSON payload and extracting it's various properties.. From here, based on the kind of message we have, the control flows to the appropriate piece of code to perform the logical operations like registering a new user, or fulfilling a search request.  
  
After the operation has been completed, the JSON response is prepared in accordance with teh operation adn the reply is sent back to the appropriate client over the websocket.

**Client:**

The client is written in python. We went with Python as it provided us with good options to build a menu driven interface for each client that can be used easily through a terminal. It also has strong support for websockets, allowing for proper communication with the server over the websocket. The client has the URI for the server configured within it and utilizes it while establishing connection to the server

On startup the client is presented with the option to register as a user if they are new or to login to the system in case they are an exisiting user. After succesful authentication, they are provided with a menu driven interface, providing them the option to choose from amongst a number of different options.  
  
The options on offer fulfil all the stated requirements of this project - the user can tweet or re-tweet, they can follow other users, search of specific hastags, search for mentions, look at their feed etc.  
  
Based on the option selected by the user and other inputs specific to the option, we prepared a JSON object by taking a JSON dump of the prepared dictionary object. This JSON message is then sent over the websocket to the server which subsequently performs the actions pertaining to the specific option selected and sends an appropriate response.  
  
After the client has completed their use of the application, they can logout with the option to log back in at any point while the server remains active.   
**Demo Video Link**

<https://youtu.be/Ecbk6sY-6EE>