BLG 433-E Project Report

Project No	Project 1
Lecturer's Name	Fatma Sema Oktuğ
Student's Name	Alperen KANTARCI – 150140140 – Computer Engineering
Delivery Date	04.11.2018

1. RUNNING THE PROJECT FILES

In this project you need Python 3.7.0 to run the both *Client.py* and *Server.py* files. You will also need threading and socket modules to run. Application only works at console so there is no additional UI modules. All writing and reading operations will be handled via terminal. There is two python files; *Client.py* and *Server.py*. You should first run the *Server.py* with python *Server.py* command and then you should run Client.py for successful running scenario. If you try to run Client before the Server than you won't be able to connect with the server. Also you should specify the server IP in the *Client.py* file if you don't do this, at the start of the program it will be asked in the terminal and you will enter server IP from the terminal. You do not have to give any arguments for the script. After running the script required inputs will be asked from the terminal.

2. USAGE OF THE PROJECT

When you run the *Server.py* script it will give you message about if it's successful to create server socket or not. Then, at each Client connection server will print the incoming requests and incoming messages to the terminal. When you run *Client.py* if no default IP is given by the source code, server IP will be requested from the terminal. If connection is successful with the server than server will print you to the current users' usernames to inform you the claimed usernames. Then you will be asked to choose username which is not present at the current chatroom. If you have selected valid username then you are able to talk with other users or get messages from the others. If you selected wrong username (already present name) then you will be disconnected from the server but you are free to connect again. You are free to write messages any time you want. At the same time you will get the all messages that coming to the server excluding yours. You can see the username of the message's sender and the send time. You are also able to see the disconnected users, so that you won't be talking to someone who is not using the application anymore.

3. EXAMPLE USAGE SCENARIO

Here you see 3 clients chatting with each other and you also see the server logs too. Clients connecting to the server with this order; Alperen, Furkan, Gamze.

```
[arch@arch Homework_1]$ python Client.py
Welcome to the server. Here is users in the chat room:
*No user found*
Please choose username:
Alperen
You have selected your username. You can now chat with the others.
Aa kimseler yok :(
Furkan @ 23:03:10 PM >> Aa Alperen selam
Furkan @ 23:03:16 PM >> ben de kimse yoktur diye bekliyordum
Selam Furkan
valla kimsecikler yoktu
Furkan @ 23:03:38 PM >> Naber nasılsın ?
İyidir ne olsun
Sen nasılsın ?
Gamze @ 23:04:08 PM >> Selam gençler doluşmuşsunuz
Hoşgeldin Gamze
Furkan @ 23:04:35 PM >> Hoşgeldin Gamze ben de tam çıkıyordum gitmeden görüşmüş olduk
Gamze @ 23:04:43 PM >> Hoşbulduk
Gamze @ 23:04:52 PM >> Ben de derse gitmeden bir selam vereyim dedim
Furkan @ 23:05:03 PM >> İyi yaptın :D.
Furkan @ 23:05:11 PM >> Derste görüşürüz o zaman ben çıkıyorum
Furkan is disconnected from the chat  @ 23:05:14 PM
Ben de çıkıyorum Gamze
Görüşürüz
Gamze @ 23:05:58 PM >> Görüşürüz
exit
Thank you for using chat application. You will be disconnected...
[arch@arch Homework_1]$
```

Figure 1: Client 1 Alperen

```
[arch@arch Homework_1]$ python Client.py
Welcome to the server. Here is users in the chat room:
Alperen
Please choose username:
Furkan
You have selected your username. You can now chat with the others.
Aa Alperen selam
ben de kimse yoktur diye bekliyordum
Alperen @ 23:03:23 PM >> Selam Furkan
Alperen @ 23:03:29 PM >> valla kimsecikler yoktu
Naber nasılsın ?
Alperen @ 23:03:43 PM >> İyidir ne olsun
Alperen @ 23:03:49 PM >> Sen nasılsın ?
Gamze @ 23:04:08 PM >> Selam gençler doluşmuşsunuz
Alperen @ 23:04:17 PM >> Hoşgeldin Gamze
Hoşgeldin Gamze ben de tam çıkıyordum gitmeden görüşmüş olduk
Gamze @ 23:04:43 PM >> Hoşbulduk
Gamze @ 23:04:52 PM >> Ben de derse gitmeden bir selam vereyim dedim
İyi yaptın :D.
Derste görüşürüz o zaman ben çıkıyorum
exit
Thank you for using chat application. You will be disconnected...
```

Figure 3: Client 2 Furkan

```
[arch@arch Homework_1]$ python Client.py
Welcome to the server. Here is users in the chat room:

Alperen
Furkan
Please choose username:
Gamze
You have selected your username. You can now chat with the others.
Selam gençler doluşmuşsunuz
Alperen @ 23:04:17 PM >> Hoşgeldin Gamze
Furkan @ 23:04:35 PM >> Hoşgeldin Gamze ben de tam çıkıyordum gitmeden görüşmüş olduk
Hoşbulduk
Ben de derse gitmeden bir selam vereyim dedim
Furkan @ 23:05:03 PM >> İyi yaptın :D.
Furkan @ 23:05:11 PM >> Derste görüşürüz o zaman ben çıkıyorum
Furkan is disconnected from the chat  @ 23:05:14 PM
Alperen @ 23:05:37 PM >> Ben de çıkıyorum Gamze
Alperen @ 23:05:41 PM >> Görüşürüz
Görüşürüz
Alperen is disconnected from the chat  @ 23:06:02 PM
exit
Thank you for using chat application. You will be disconnected...
```

Figure 2: Client 3 Gamze

```
[arch@arch Homework_1]$ python Server.py
Socket is successfully created.
Socket is being used.
Binding is done.
The server is ready to receive users.
('160.75.196.214', 33810) is connected to the server
('160.75.196.214', 33810) is claimed Alperen as its username
('160.75.196.214', 33810)  says:  Aa kimseler yok :(
('160.75.196.214', 33812) is connected to the server
('160.75.196.214', 33812) is claimed Furkan as its username
('160.75.196.214', 33812)  says:  Aa Alperen selam
('160.75.196.214', 33812)  says:  ben de kimse yoktur diye bekliyordum
('160.75.196.214', 33810)  says:  Selam Furkan
('160.75.196.214', 33810)  says:  valla kimsecikler yoktu
('160.75.196.214', 33812)  says:  Naber nasılsın ?
('160.75.196.214', 33810)  says:  İyidir ne olsun
('160.75.196.214', 33810)  says:  Sen nasılsın ?
('160.75.196.214', 33818) is connected to the server
('160.75.196.214', 33818) is claimed Gamze as its username
('160.75.196.214', 33818)  says:  Selam gençler doluşmuşsunuz
('160.75.196.214', 33810)  says:  Hoşgeldin Gamze
('160.75.196.214', 33812)  says:  Hoşgeldin Gamze ben de tam çıkıyordum gitmeden görüşmüş olduk
('160.75.196.214', 33818)  says:  Hoşbulduk
('160.75.196.214', 33818)  says:  Ben de derse gitmeden bir selam vereyim dedim
('160.75.196.214', 33812)  says:  İyi yaptın :D.
('160.75.196.214', 33812)  says:  Derste görüşürüz o zaman ben çıkıyorum
('160.75.196.214', 33812)  is disconnected
('160.75.196.214', 33810)  says:  Ben de çıkıyorum Gamze
('160.75.196.214', 33810)  says:  Görüşürüz
('160.75.196.214', 33818)  says:  Görüşürüz
('160.75.196.214', 33810)  is disconnected
('160.75.196.214', 33818)  is disconnected
```

Figure 4: Server Log

4. EXPLANATION OF THE SOURCE CODE

In *Server.py* there is a dictionary that initialized as empty which named *client_names*. This dictionary has username keys (because every username will be unique) and the corresponding values are the sockets of that user. So in this dictionary there will be only users with the active socket connection. If user wants to exit from the application by sending “exit” as a message then the username with its connection will be removed from this dictionary. There is a while loop at the main function which works at the main thread to accept new incoming socket requests from the different clients. If user selects username appropriately then another thread will be created in order to listen that client at the background. In thread listenClient function is running continuously. It listens the client and whenever user send message, server sends the same message to all the users that currently connected except the sender. There is two if condition for the message, if message is “exit” than user will be disconnected and socket will be closed. If message is empty string like “” then this means that user disconnected unexpectedly because TCP socket streams cannot send empty strings through the stream. So if message is empty then user will be disconnected from the Server as well. If message is not one of them then message is valid and will be sent to the all users.

In *Client.py* there is a main function that ask server IP if there is no default IP after that Client object will be created just after that server connection will be tried with supplied IP and port. If it’s successful two different threads will be created for listening and sending messages. Receive message function runs on the one thread it continuously listens socket and if there is any message then it will check the message. If message is empty (“”) then this means that connection is closed by the server or unexpectedly closed so program will be closed. If not

message is valid so it will be printed to the terminal. sendMessage function is running on the another thread and its function is taking input from the console and send the message to the server. If message is “exit” then this message will be sent to the server and connection will be closed by the client then program will be terminated. If user try to send empty messages then informing message will be given to the user because empty messages cannot be sent over the socket stream. If message is not one of them then it will be sent to the server.