COMPUTER NETWORKS

PROJECT – REPORT

PEER-TO-PEER CHATTING

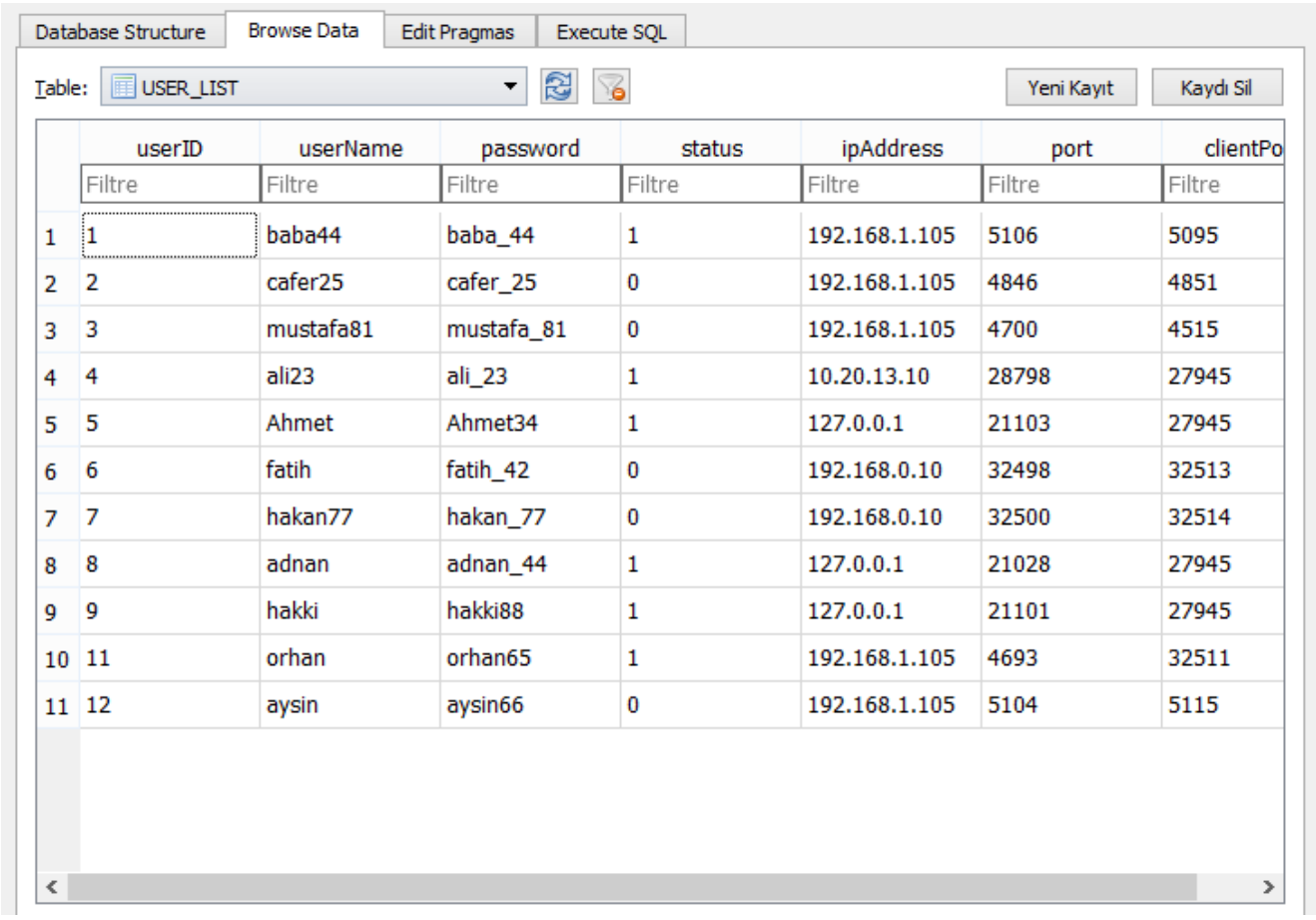
UFUK GÜRBÜZ – 150113058

MUSTAFA YEMURAL – 150113053

PROCEDURES

First of all, we designed this project to provide peer-to-peer group messaging. That is, one person can group with another person or with a desired number of people. We used the "loby" logic to design such a messaging system. You absolutely have to play online games online (like Facebook 101 okey). In this game system, a room ("loby") is established and users communicate with each other by joining this room. Thus, you can systematically manage an unlimited number of users according to their own wishes. Our system works in a similar way as the general structure. A "peer" is selected as the "loby leader" before the beginning of the conversation, and the other "peers" connect to this "peer" and communicate with each other through it. With this structure we have designed, you can group messaging with an unlimited number of people. Of course the unlimited expression here may change in the direction of the resources of your computer system. However, we tested this system with 20 different "peer" participation and we did not encounter any problems.

In this project, we first need a central server at the design stage of our system. So, we have created 1 central server. This server will keep the users' names, passwords, ip addresses, port information and online status. We created a database for these operations and made the corresponding column assignments. We will then be able to retrieve user information from the database when necessary. Below you can see the data base table where user information is stored on the central server side.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: USER_LIST — Yeni Kayıt — Kaydı Sil

	userID	userName	password	status	ipAddress	port	clientPo
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	baba44	baba_44	1	192.168.1.105	5106	5095
2	2	cafer25	cafer_25	0	192.168.1.105	4846	4851
3	3	mustafa81	mustafa_81	0	192.168.1.105	4700	4515
4	4	ali23	ali_23	1	10.20.13.10	28798	27945
5	5	Ahmet	Ahmet34	1	127.0.0.1	21103	27945
6	6	fatih	fatih_42	0	192.168.0.10	32498	32513
7	7	hakan77	hakan_77	0	192.168.0.10	32500	32514
8	8	adnan	adnan_44	1	127.0.0.1	21028	27945
9	9	hakki	hakki88	1	127.0.0.1	21101	27945
10	11	orhan	orhan65	1	192.168.1.105	4693	32511
11	12	aysin	aysin66	0	192.168.1.105	5104	5115

Central Server - Database Table

When a "peer" enters the system, it communicates with the "central server" and gives the program two options: "register" and "join". If this "peer" is not already registered in the database on the "central server", it will register to the "central server" by specifying a unique username and password. The "peer" a "Succesful" message is returned if there is no error in the "central server" register phase. If you have already registered to the server, you are expected to log in. When "peer" enters the system, the "central server" updates that person's 'online' status. Since 'Peer' is online, they constantly send a 'HELLO' message every 10 seconds. These messages coming from the 'central server' know that 'peer' is online. If the 'peer' has been removed from the system and the 'HELLO' messages have stopped appearing, the 'peer' in 'online' status is updated to 'offline'. In this way, 'central server' real-time 'peer' s situation is followed. Likewise, if the "central server" does not encounter any errors during the 'join' phase, the message "peer" returns a 'Succesful' message.

Then the program offers 3 different options to this "peer":

1) 'Show online user list',

2) 'Search a peer (getOtherPeerInfos)',

3) 'Start to chat'.

If a 'Peer' selects a fist option, the program will return a list ('userName', 'ipAddress', 'port') of all online 'peers' of that moment. Thus, the ‘peer’ can see the 'userName' information of the all ‘peers’ which want to communicate with and can send a 'CHAT_REQUEST' to it.

If 'Peer' selects the second option, the program will request a 'userName' from it. Thus 'Peer' learns 'peer' ın information he wants to know about contact information. The program then asks 'peer' if this person wants to be added to the group chat list. If 'Peer' enters the ‘YES’ reply, it is added to the group chat list.

If 'Peer' selects the third option, the second step is to send 'CHAT_REQUEST' to all the 'peers' that the group added to the messaging list. Other 'peers' can give 'OK' or 'REJECT' responses to this message request in their own desire. If a 'peer' is already in a conversation, the message 'BUSY' is automatically returned to the requesting 'peer'.

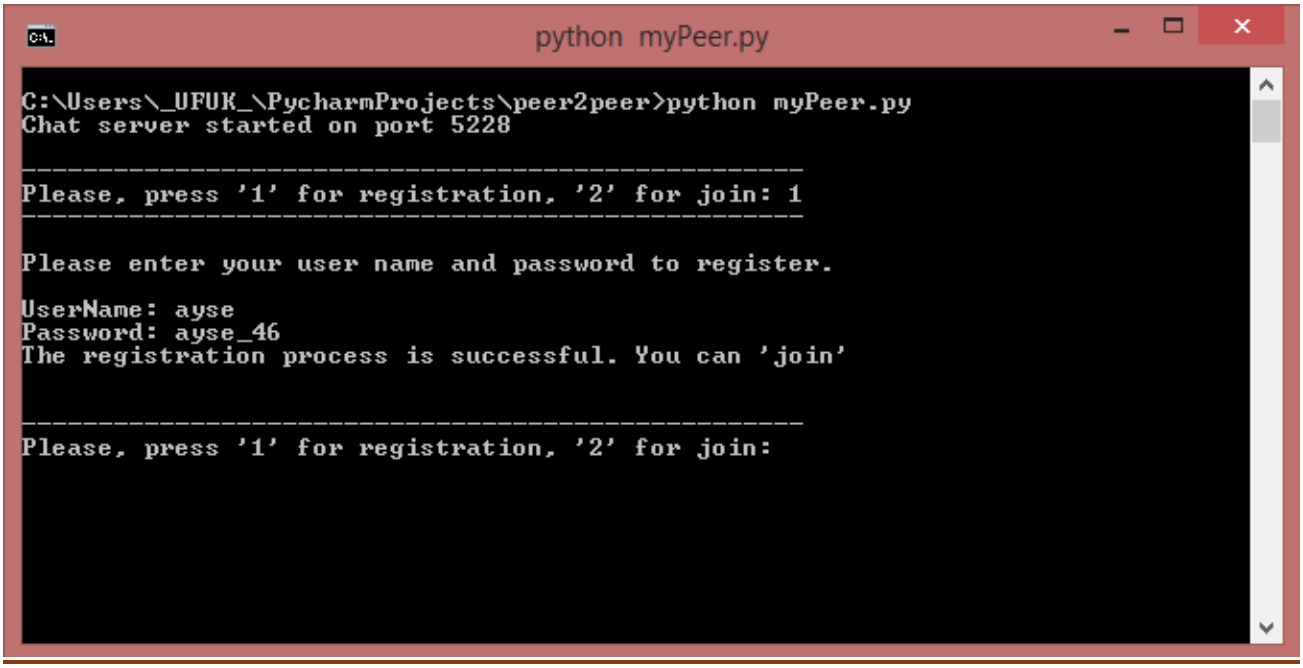
Then, A conversation room ('loby') is established with the 'OK' responses from the 'peers' we are asking to connect and the messaging starts.

Finally, when it comes out of any 'peer' room ('lobby'), it sends a message to other 'peers' that the person is 'offline'. Then the other 'peers' in the group can continue messaging smoothly. If the person leaving the system is the owner (lobbyLeader) 'peer', the group messaging ends. In this case, the other 'peers' in the system need to build 'lobby' again.

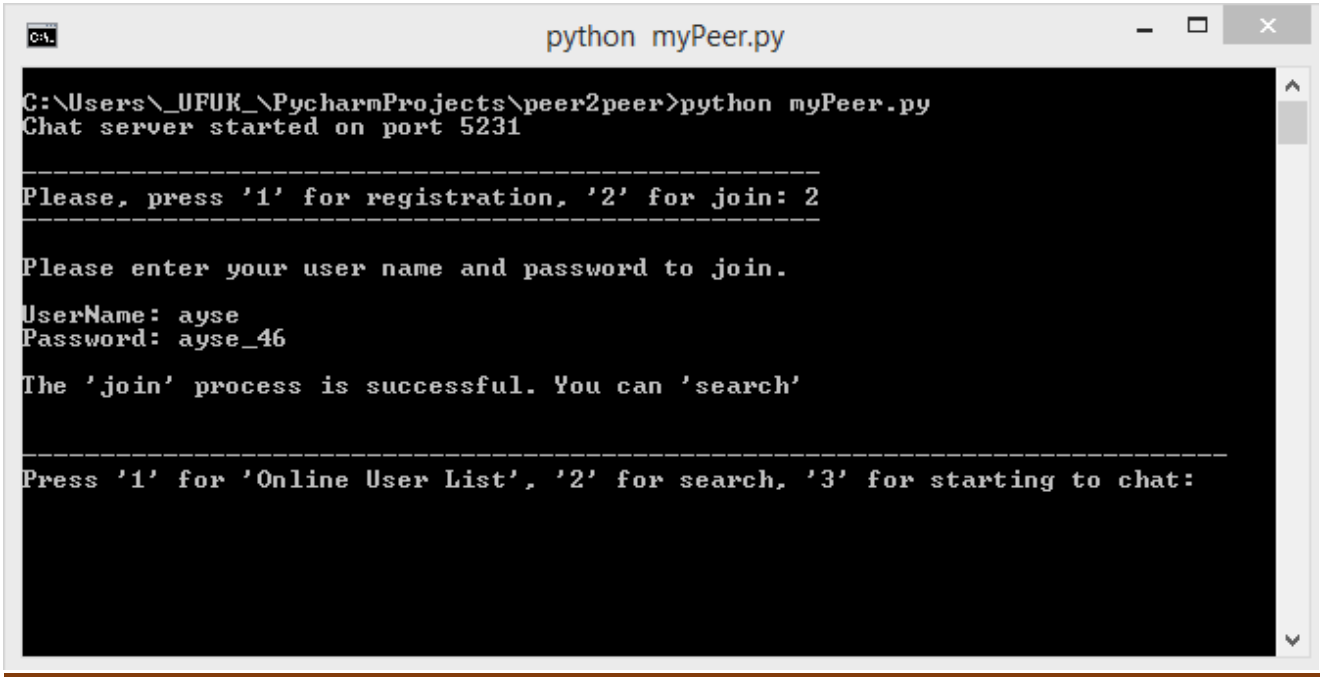
Also, we have created a ‘LOGGING’ purpose file system. Thus, all previous 'peer' talkings will be logged on a date-time basis.

SCREENSHOTS OF OUR PROGRAM

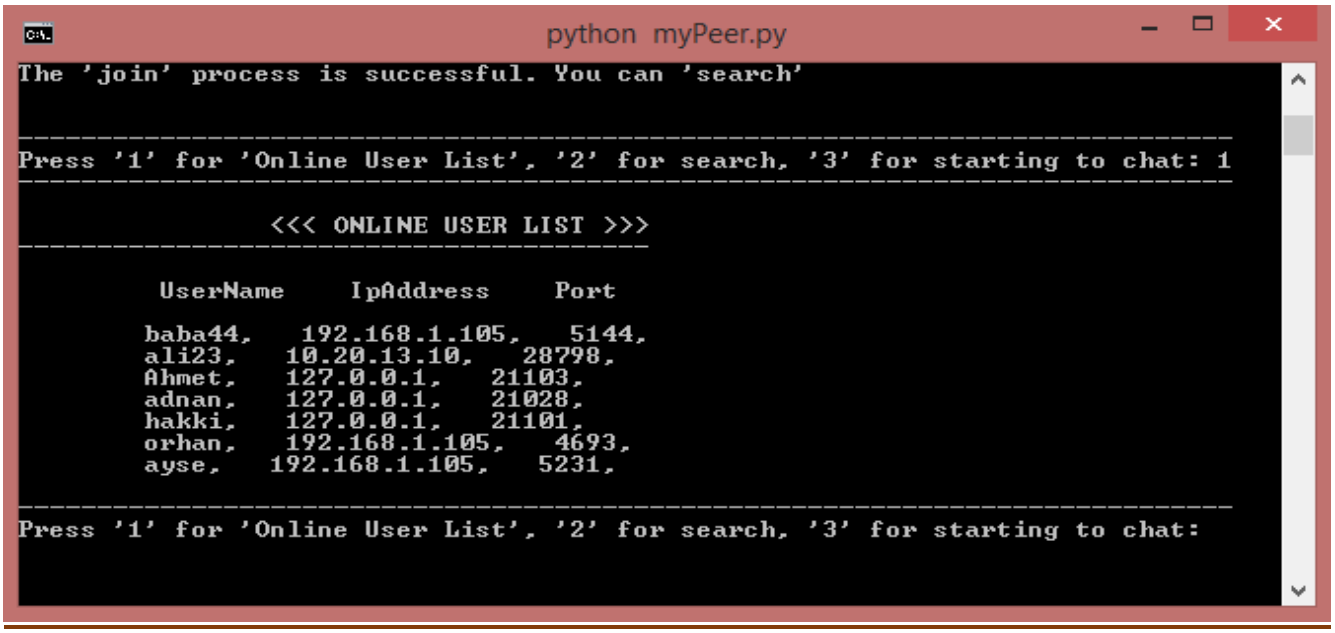
1) Registration Phase



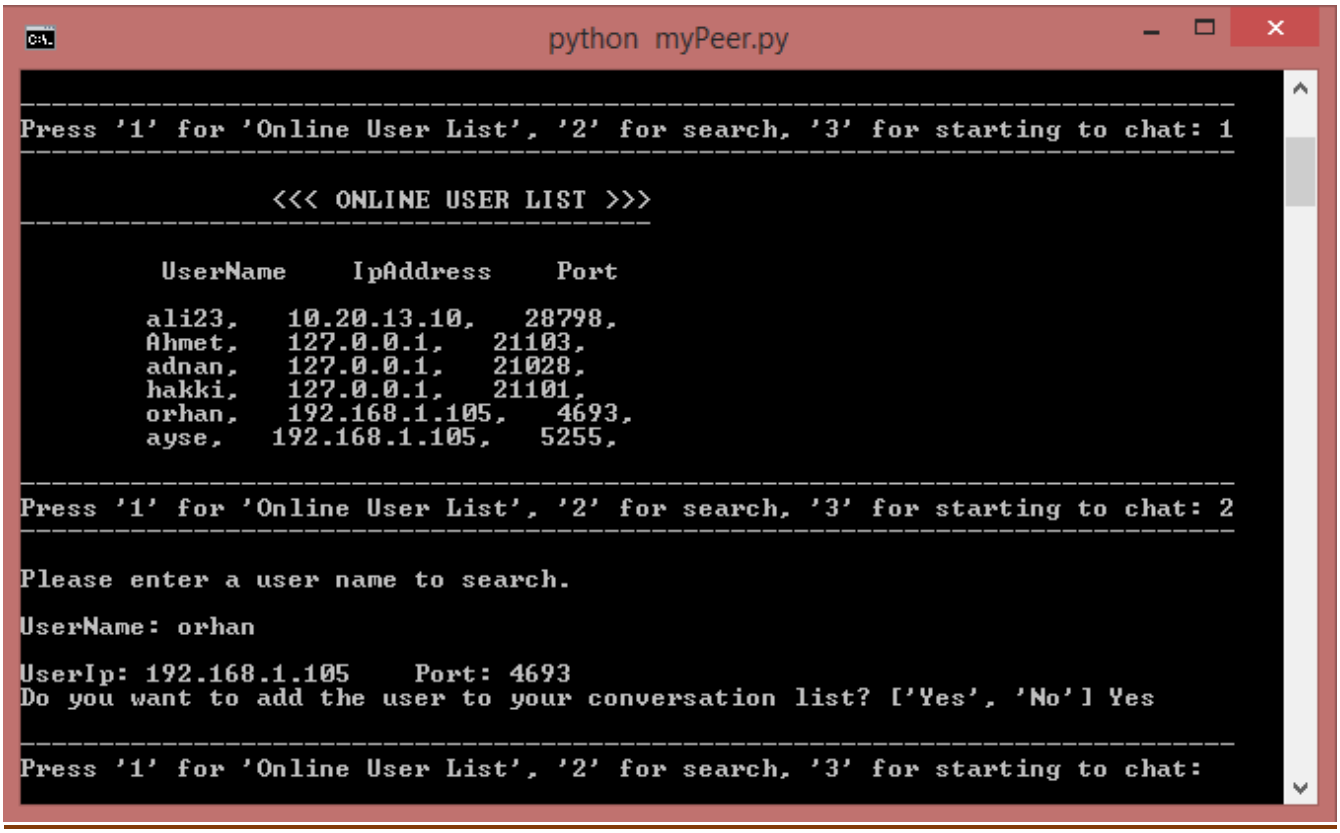
2) Join Phase



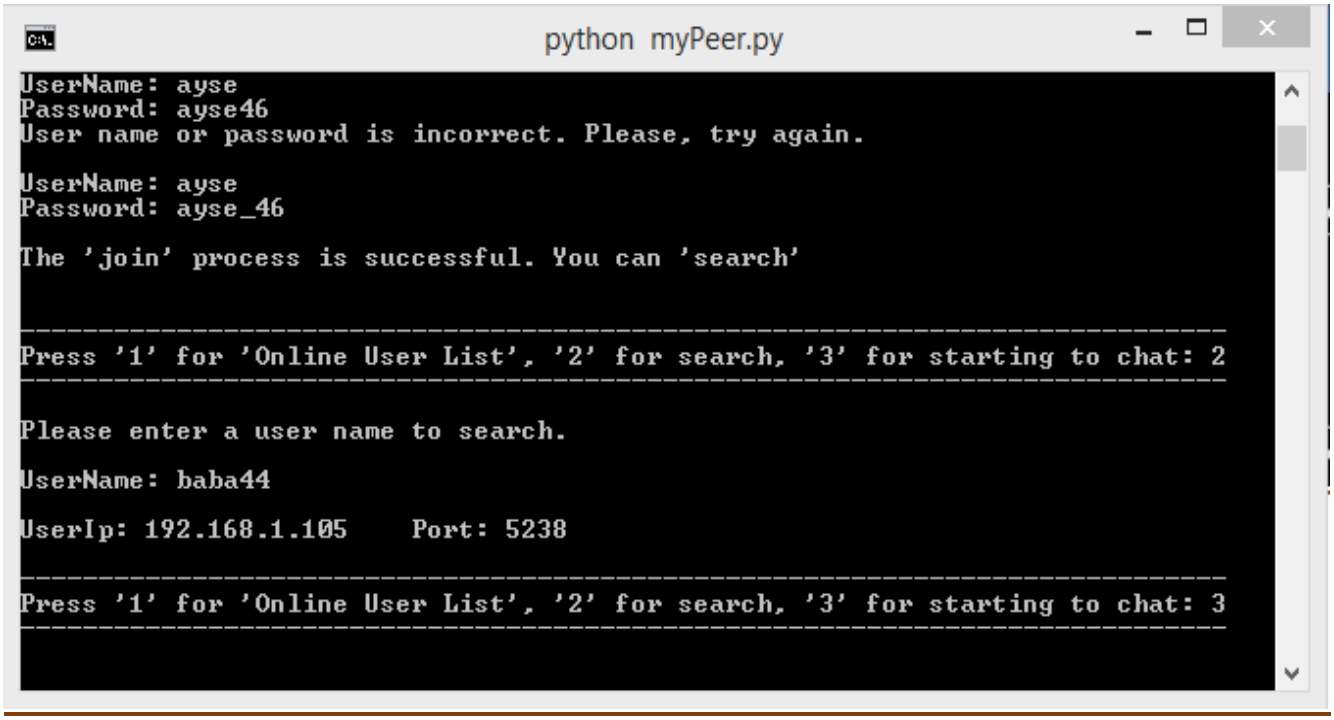
3) Action Choise Phase



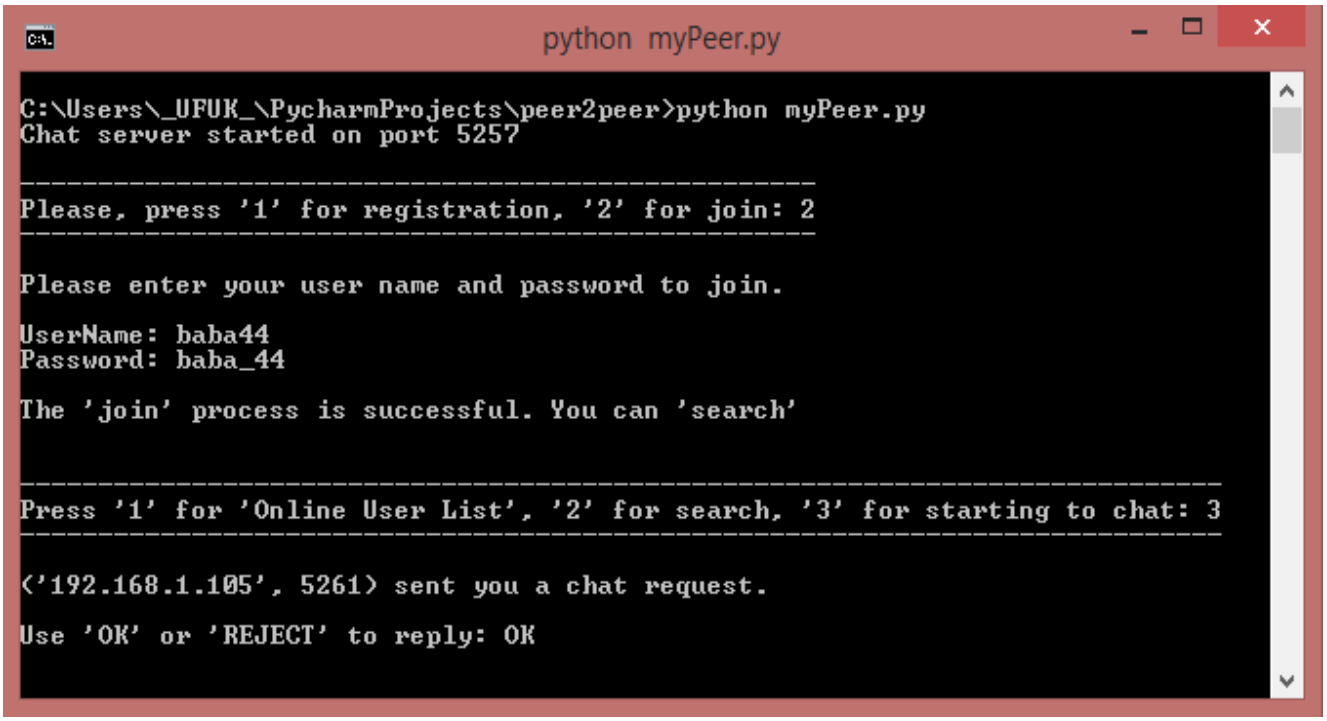
Show Online User List



Search A User

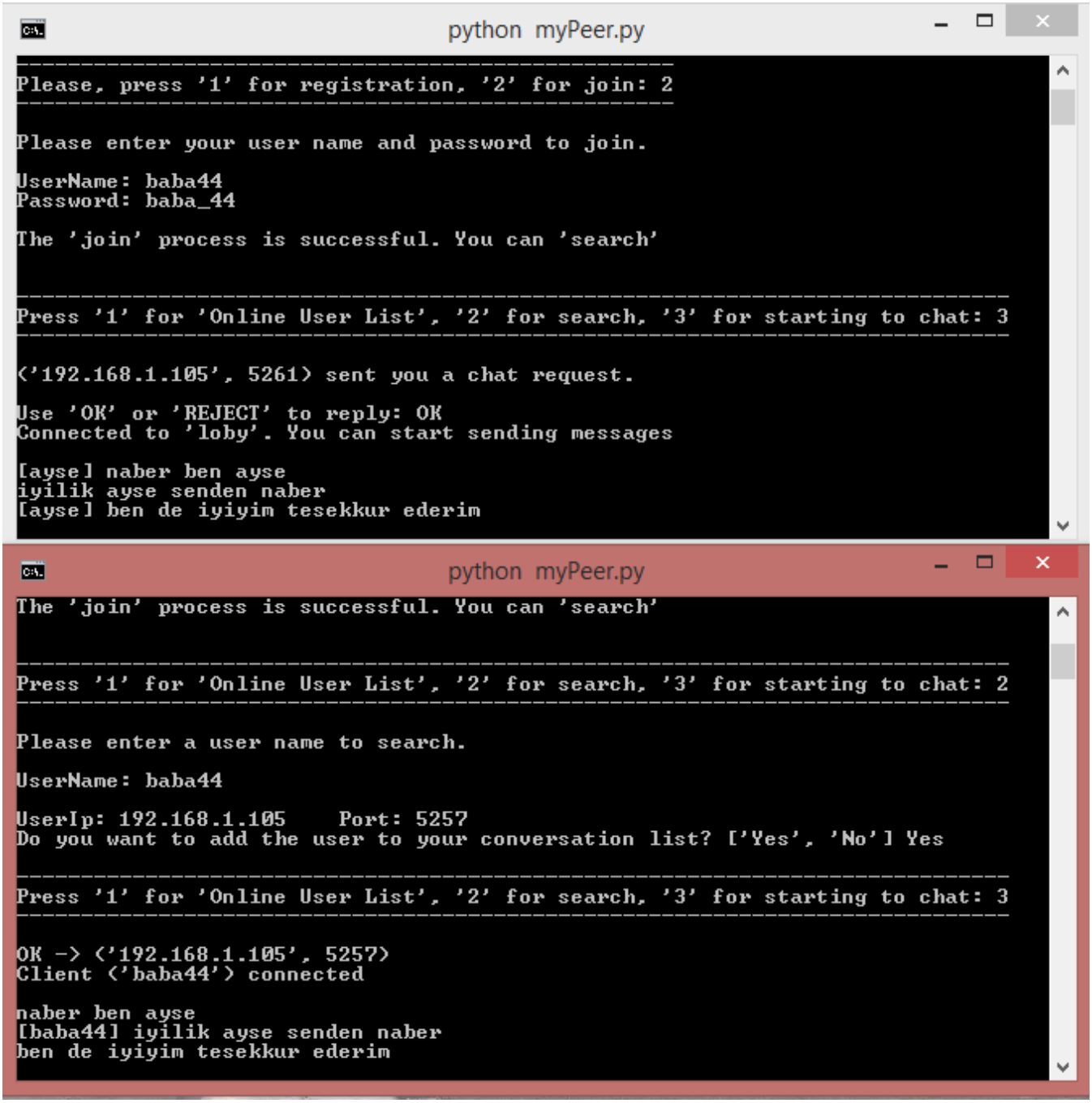


Sending Chat Request



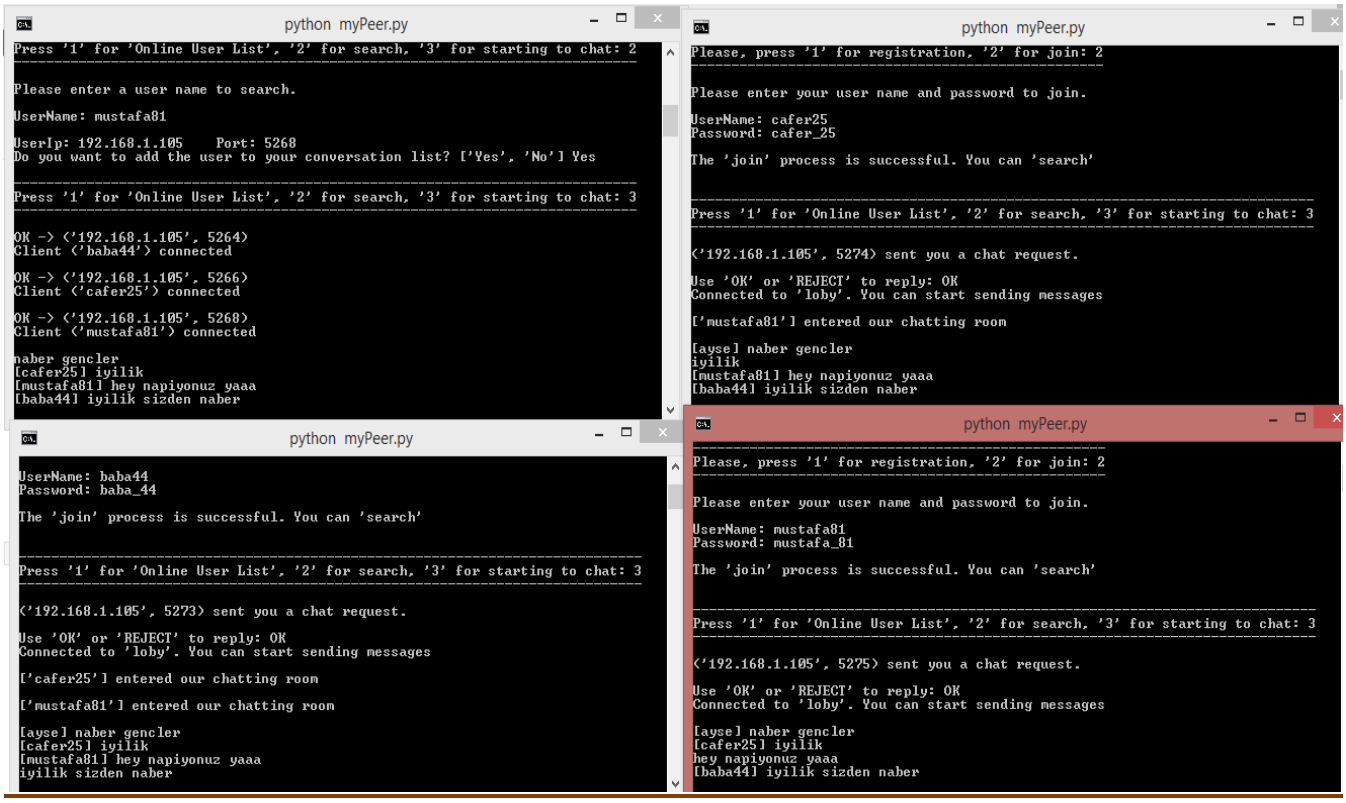
Accepting Chat Request

4) Chatting Phase



Chatting Between Peers (2-Peer)

5) Group Chatting Phase



Chatting Between Peers (Group Chatting)

6) Chatting-User Offline Phase