Name : Sareena Abdul Razak
UNI : sta2378
CSEE W4119 – PA1 – Peer to peer writeup

- When a client wants to talk to another client without server interruption, the client types getaddress < username> in the command line interface
- tcp_chat_client.py has a While True loop monitoring for any input from sockets or stdin.
- When it gets an input from stdin , the client program calls a function ***"client.processCommand(command,msg)"*** (line 86 in tcp_chat_client.py)
- processCommand function checks for the command and matches to getaddress string. Then client creates a socket, connects to server and let the server know the client is requesting IP address of the other client
- On the server side, server also has a processRequest function. The function again matches strings, extracts sender name, and the username of the client of which the sender is requesting the IP address.
- Server then checks if the sender is in the particular user's block list, if yes, server replies saying the user has blocked the sender. If not server sends a message to the client program of the user, letting it know someone is requesting the IP address of the user.
- In the client side, when it gets the message, client program displays a message asking for consent (this is a bonus feature). If the user says no, the client program sends a message to server, the server in turn sends a notification to the original sender
- If the user says yes, the server sends IP address of the user to the original sender.
- Once the client gets the IP address of another peer, the user can type private <username> < message> and start the chat. My client program starts a TCP connection directly to the peer without server intervention. This is also a non-persistent connection.
- To verify this, you can take a look at Client.py (class for client) line no: 262
- I am copying the code below

```
elif(command == "private"):
        if(len(msg.split(' ')) > 2):
          user = msg.split(' ')[1]
          message = msg.split(' ',2)[2]
          final_message = str(self.name+":"+message)
          # Check if the client already knows the IP and port of the user
          if(self.privateMessageDB.has_key(user)):
```

```
                ip,port = self.privateMessageDB[user]
                status = self.connectToHost(ip,port) Direct connection to peer
                if status:
                    self.sendMsg(self.pvtHostSocket,final_message)
                    self.deInitConn(self.pvtHostSocket) Non persistent connection
            else:
                # If the user doesn't have the address, user can use getaddress
command to get it
                sys.stdout.write(">Request could not be processed.\n")
                sys.stdout.write(">Use 'getaddress <username' to get the
address of the user\n")
                sys.stdout.flush()
        else:
            # Error handle wrong format of command
            sys.stdout.write(">note: Command to private message a user is
private <username> <msg>\n")
            sys.stdout.flush()
```