

Class Chat System

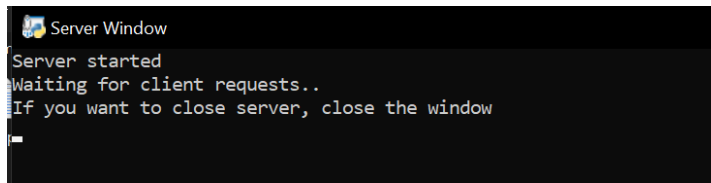
Class Chat System is a basic chat application that allows client-to-client communication via server. The concept of threading is applied to obtain multi-threaded client that can support multiple clients at a same time. In this chat system, clients connected to servers can also communicate with each other or in a group.

Steps to follow to run the files:

1. Navigate to the folder containing server.py and client.py files

2. Double click server.py file to start the server.

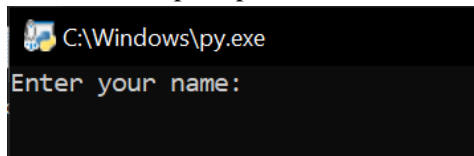
A terminal will appear that notifies that server is ready to accept connections. Note the change in title name of terminal.



```
Server Window
Server started
Waiting for client requests..
If you want to close server, close the window
```

3. Double click client.py to start a client. client.py can be double clicked multiple times to run multiple clients.

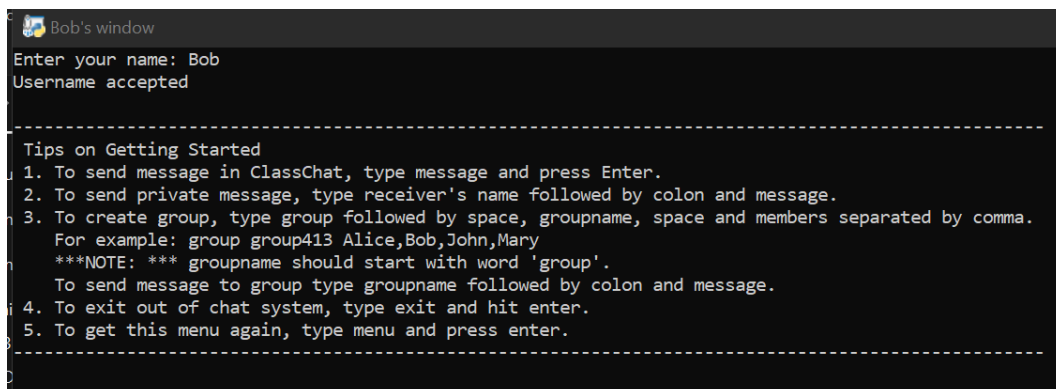
Client will be prompted to enter username.



```
C:\Windows\py.exe
Enter your name:
```

Enter username of your choice and press Enter. Client will see message to indicate whether their username is accepted or not.

Case 1: If username is accepted, client will be connected to server and the terminal's title name will be changed according to the username. Client will also be able to see the display menu.



```
Bob's window
Enter your name: Bob
Username accepted

-----
Tips on Getting Started
1. To send message in ClassChat, type message and press Enter.
2. To send private message, type receiver's name followed by colon and message.
3. To create group, type group followed by space, groupname, space and members separated by comma.
   For example: group group413 Alice,Bob,John,Mary
   ***NOTE: *** groupname should start with word 'group'.
   To send message to group type groupname followed by colon and message.
4. To exit out of chat system, type exit and hit enter.
5. To get this menu again, type menu and press enter.
-----
```

Case 2: If username is not accepted or is duplicate, user will be prompted to enter new username.

```
C:\Windows\py.exe
Enter your name: Bob
Username is taken. Use another name.
Enter your name:
```

When the clients are connected, server window will be updated accordingly as well.

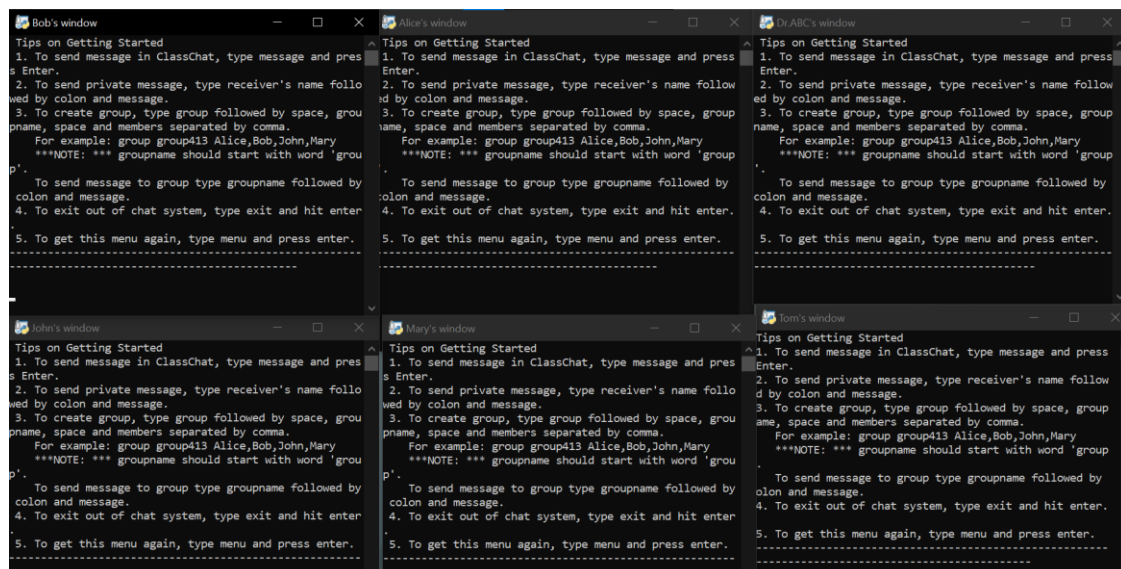
Case 1: When username is accepted, it displays the total number of threads, name of client connected, and number of active clients connected to server.

```
Server Window
Server started
Waiting for client requests..
If you want to close server, close the window
Total thread count: 1
Bob has connected
Number of active clients: 1
```

Case 2: When username is not accepted, it shows message saying “Duplicate username”

```
Server Window
Server started
Waiting for client requests..
If you want to close server, close the window
Total thread count: 1
Bob has connected
Number of active clients: 1
Total thread count: 2
Duplicate username
```

I ran client.py six times to connect six different clients to the server. Notice the difference in terminal’s title name. This helps us keep track of client when handling multiple clients at the same time.

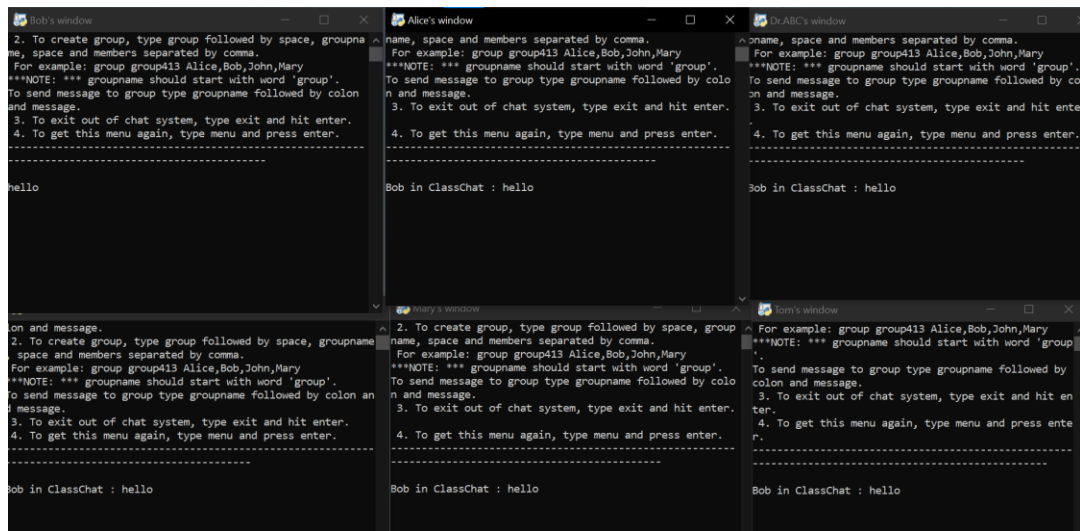


The server terminal is thus updated as:

```
Server Window
Server started
Waiting for client requests..
If you want to close server, close the window
Total thread count: 1
Bob has connected
Number of active clients: 1
Total thread count: 2
Duplicate username
Total thread count: 3
Alice has connected
Number of active clients: 2
Total thread count: 4
John has connected
Number of active clients: 3
Total thread count: 5
Mary has connected
Number of active clients: 4
Total thread count: 6
Dr.ABC has connected
Number of active clients: 5
Total thread count: 7
Tom has connected
Number of active clients: 6
```

Notice: The number of total thread count is 7 but number of active clients is 6. It is because I used a duplicate name for one of the connections. Whenever there is duplicate name, a new socket connection is established with server.

- Each connected client will be able to send the message to all the clients connected to server by typing message and pressing Enter. For example, if Bob types hello and presses Enter, all connected clients but Bob will receive it.

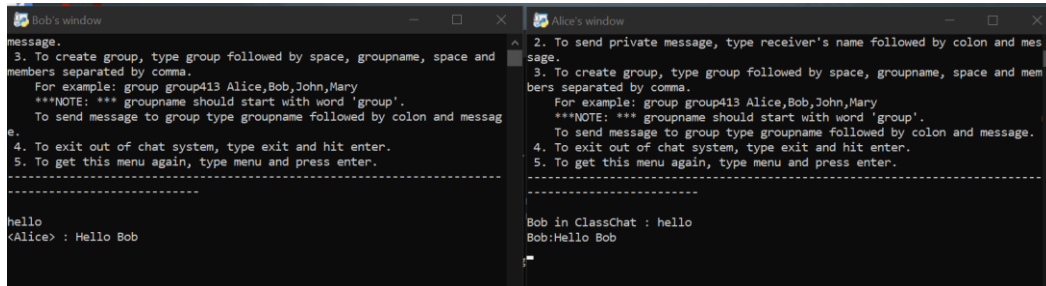


The server is updated as:

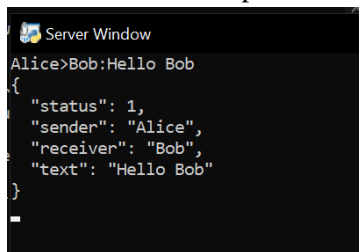
```
Bob>hello
{
  "status": 1,
  "sender": "Bob",
  "receiver": "everyone",
  "text": "hello"
}
```

Notice that the receiver is everyone, that is every client connected to server receives it.

5. Clients can communicate with each other privately. The sender should follow format - receiver:message. For example, if Alice wants to send message to Bob, it would be - Bob:Hello Bob - in Alice's window.
- The receiver will receive message in format <Sender> : message. In this case, Bob will receive <Alice> : Hello Bob



The server will be updated as follows:

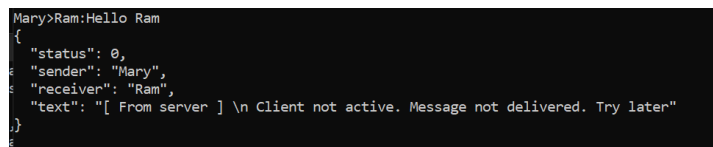


Alice>Bob:Hello indicates that Alice is sending message to Bob and the message is Hello Bob. The summary of communication is also displayed in json file format. If the status is 1, it means that message has been successfully delivered to at least one another client.

If the receiver is not connected to server, an error message is displayed. For example, in the screenshot below, Mary is trying to message Ram, but he is not connected to server. So, Mary receives an error message from server.



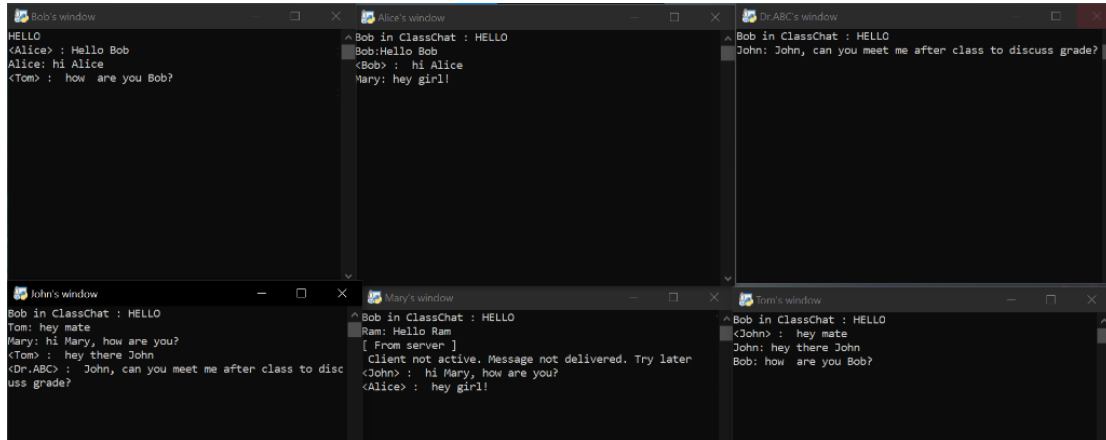
The server window is updated as:



Notice that the status is 0 which means that communication was unsuccessful, or message was not delivered.

Clients can communicate with any other clients connected to server in this format. This chat system is case sensitive, that is, it is important to notice the cases of usernames while sending messages.

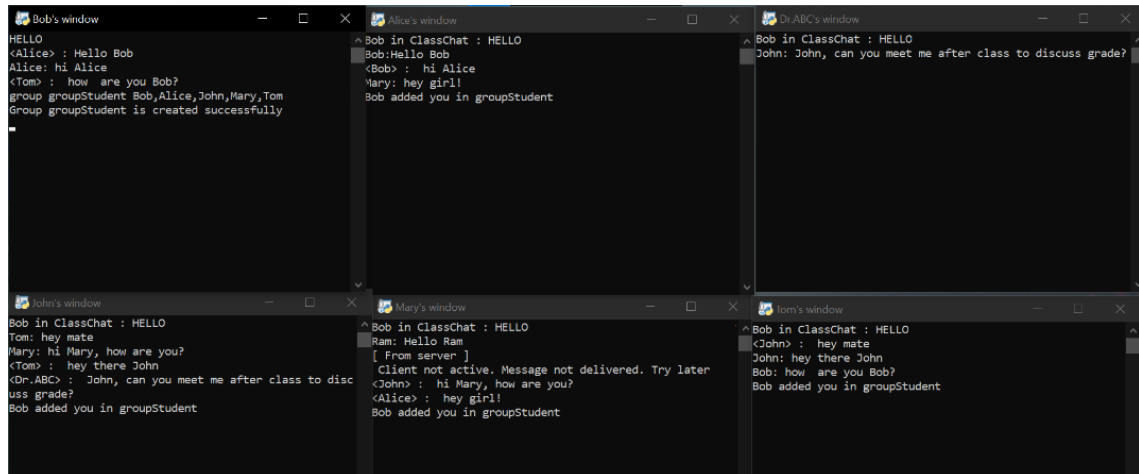
Screenshots of few individual communications:



As the communication progresses, server will be updated in the earlier format as well.

6. Clients can create their own group with keyword *group*. The keyword group is to be followed by space, group name, space and members separated by comma. Note that the group name should start word group and as mentioned earlier usernames are case-sensitive.

For example, if Bob wants to create group of students, he can type -
group groupStudent Bob,Mary,John,Alice,Tom



As shown in the screenshot above, the members of the groups are notified accordingly. Notice that Dr.ABC is not in the group and hence he does not receive any notification about it.

The server terminal is updated with group name, number, and name of people in group.

```
Bob>group groupStudent Bob,Alice,John,Mary,Tom
GroupName: groupStudent
Group of 5 people created.
['Bob', 'Alice', 'John', 'Mary', 'Tom']
```

Another important thing to remember is that group should have at least three members in it. As a group of two users is same as private messaging, user will not be able to create group of two users even when they try to do so. For example, if John tries to create a group with Bob and himself, he will get an error message.

```
John's window
group groupJohnBob Bob,John
Group should have atleast three members
```

The server is updated as:

```
Server Window
John>group groupJohnBob Bob,John
Group creation unsuccessful
```

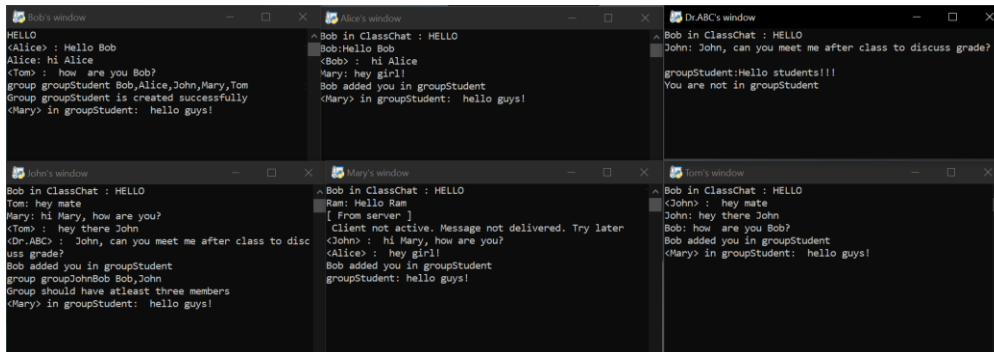
7. The members of group can send message in the group using the format described earlier (receiver:message), however the receiver name should be the group name. For example, Mary can send in group as - groupStudent: hello guys! . All the group members will get the message.

The screenshot displays six chat windows arranged in a 2x3 grid. Each window shows a history of messages. Bob's window shows a 'HELLO' message and a group creation announcement. Alice's window shows a 'HELLO' message and a group creation announcement. Dr. ABC's window shows a 'HELLO' message and a group creation announcement. John's window shows a 'HELLO' message and a group creation announcement. Mary's window shows a 'HELLO' message and a group creation announcement. Tom's window shows a 'HELLO' message and a group creation announcement. The messages in the windows are as follows:

- Bob's window: HELLO, <Alice>: Hello Bob, Alice: hi Alice, <Tom>: how are you Bob?, group groupStudent Bob,Alice,John,Mary,Tom, Group groupStudent is created successfully, <Mary> in groupStudent: hello guys!
- Alice's window: Bob in ClassChat: HELLO, <Bob>: hi Alice, <Mary>: hey girl!, Bob added you in groupStudent, <Mary> in groupStudent: hello guys!
- Dr. ABC's window: Bob in ClassChat: HELLO, <John>: John, can you meet me after class to discuss grade?
- John's window: Bob in ClassChat: HELLO, Tom: hey mate, Mary: hi Mary, how are you?, <Tom>: hey there John, <Dr.ABC>: John, can you meet me after class to discuss grade?, Bob added you in groupStudent, group groupJohnBob Bob,John, Group should have atleast three members, <Mary> in groupStudent: hello guys!
- Mary's window: Bob in ClassChat: HELLO, Ram: Hello Ram, [From server], Client not active. Message not delivered. Try later, <John>: hi Mary, how are you?, <Alice>: hey girl!, Bob added you in groupStudent, groupStudent: hello guys!
- Tom's window: Bob in ClassChat: HELLO, <John>: hey mate, John: hey there John, Bob: how are you Bob?, Bob added you in groupStudent, <Mary> in groupStudent: hello guys!

Notice that Dr.ABC is not receiving any messages from the group.

A client who is not the member of group will not be able to message in group. For example, if Dr.ABC wants to send message in group and tries to do so, he will get an error message and the group members will not receive it either.



The server terminal as group communication proceeds is updated as:

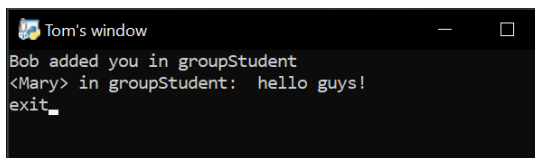
```

[Mary]>groupStudent: hello guys!
sent to 4
{
  "status": 1,
  "sender": "Mary",
  "receiver": "groupStudent",
  "text": " hello guys!"
}
Dr.ABC>groupStudent:Hello students!!!
{
  "status": 0,
  "sender": "Dr.ABC",
  "receiver": "groupStudent",
  "text": "Hello students!!!"
}

```

When Mary sends the message in group, it displays the number of clients the message is sent to, and the status of communication is 1. But when Dr.ABC tries to send message to group, it shows that status is 0 which means that communication was unsuccessful.

8. Any client can choose to disconnect from the server using *exit* keyword and pressing Enter. For example, when Tom types exit and hits enter, his terminal will automatically be closed.



The server will be updated as

```

Tom>exit
Tom disconnected...

```

Now when any client messages Tom, they will get a message saying client is not active.

```
Mary's window
Bob added you in groupStudent
groupStudent: hello guys!
Tom:Are you there Tom?
[ From server ]
Client not active. Message not delivered. Try later
```

When any group member tries to message in group, all but Tom will receive the message.

```
Bob's window
HELLO
<Alice> : Hello Bob
Alice: hi Alice
<Tom> : how are you Bob?
group groupStudent Bob,Alice,John,Mary,Tom
Group groupStudent is created successfully
<Mary> in groupStudent: hello guys!
<Alice> in groupStudent: How is everybody doing?

Alice's window
Bob in ClassChat : HELLO
Bob:Hello Bob
<Bob> : hi Alice
Mary: hey girl!
Bob added you in groupStudent
<Mary> in groupStudent: hello guys!
groupStudent:How is everybody doing?

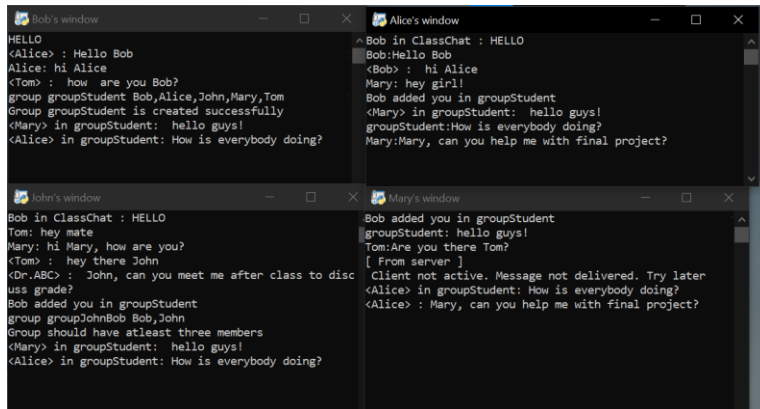
John's window
Bob in ClassChat : HELLO
Tom: hey mate
Mary: hi Mary, how are you?
<Tom> : hey there John
<Dr.ABC> : John, can you meet me after class to discuss grade?
Bob added you in groupStudent
group groupJohnBob Bob,John
Group should have atleast three members
<Mary> in groupStudent: hello guys!
<Alice> in groupStudent: How is everybody doing?

Mary's window
Bob added you in groupStudent
groupStudent: hello guys!
Tom:Are you there Tom?
[ From server ]
Client not active. Message not delivered. Try later
<Alice> in groupStudent: How is everybody doing?
```

When Tom disconnects, the number of group members will also decrease by 1, which is indicated in server.

```
Alice>groupStudent:How is everybody doing?
sent to 3
{
  "status": 1,
  "sender": "Alice",
  "receiver": "groupStudent",
  "text": "How is everybody doing?"
}
```


9. As the group chat goes on, each client will still be able to directly message to each other. For example, Alice can message Mary in a personal chat.



```
Bob's window
HELLO
<Alice> : Hello Bob
Alice: hi Alice
<Tom> : how are you Bob?
Group groupStudent Bob,Alice,John,Mary,Tom
Group groupStudent is created successfully
<Mary> in groupStudent: hello guys!
<Alice> in groupStudent: How is everybody doing?

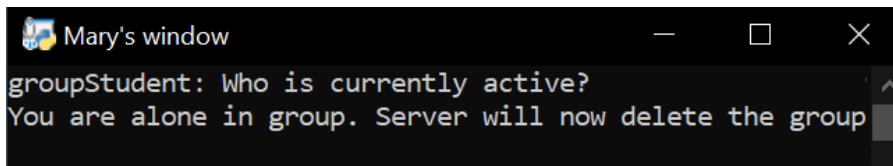
Alice's window
^ Bob in ClassChat : HELLO
Bob:Hello Bob
<Bob> : hi Alice
Mary: hey girl!
Bob added you in groupStudent
<Mary> in groupStudent: hello guys!
groupStudent:How is everybody doing?
Mary:Mary, can you help me with final project?

John's window
Bob in ClassChat : HELLO
Tom: hey mate
Mary: hi Mary, how are you?
<Tom> : hey there John
<Dr.ABC> : John, can you meet me after class to discuss grade?
Bob added you in groupStudent
group groupJohnBob Bob,John
Group should have atleast three members
<Mary> in groupStudent: hello guys!
<Alice> in groupStudent: How is everybody doing?

Mary's window
Bob added you in groupStudent
groupStudent: hello guys!
Tom:Are you there Tom?
[ From server ]
Client not active. Message not delivered. Try later
<Alice> in groupStudent: How is everybody doing?
<Alice> : Mary, can you help me with final project?
```

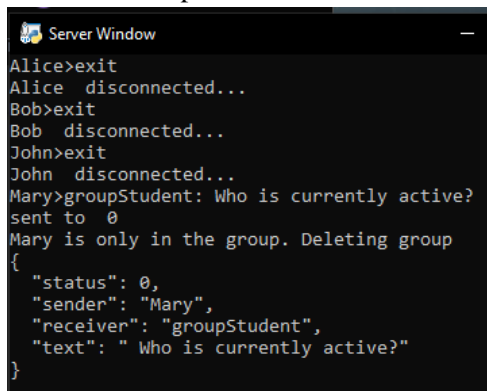
Notice in screenshot above, only Mary receives Alice's message. Bob and John do not receive it.

10. Suppose Alice, Bob, and John also decide to disconnect from the server with *exit* keyword. When Mary messages in the group, she will get receive message which notifies that group will be deleted.



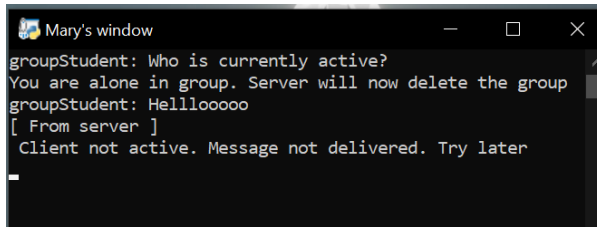
```
Mary's window
groupStudent: Who is currently active?
You are alone in group. Server will now delete the group
```

The server is updated as follows:



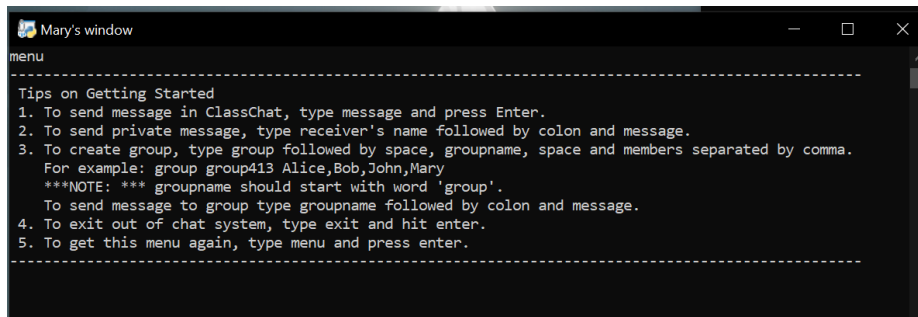
```
Server Window
Alice>exit
Alice disconnected...
Bob>exit
Bob disconnected...
John>exit
John disconnected...
Mary>groupStudent: Who is currently active?
sent to 0
Mary is only in the group. Deleting group
{
  "status": 0,
  "sender": "Mary",
  "receiver": "groupStudent",
  "text": " Who is currently active?"
}
```

Now if Mary tries to send message in group, she will get “client not connected” message from the server.



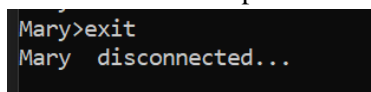
```
Mary's window
groupStudent: Who is currently active?
You are alone in group. Server will now delete the group
groupStudent: Helllooooo
[ From server ]
Client not active. Message not delivered. Try later
```

- Any client connected to server can choose to view the initial menu using *menu* keyword and pressing Enter. For example, Mary is still connected. When she types menu and presses Enter, she will get the menu displayed to her when she first connected to the server.



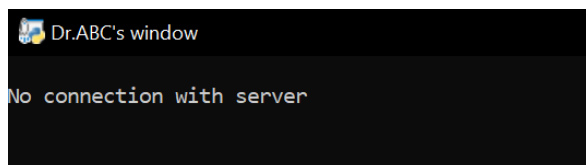
```
Mary's window
menu
-----
Tips on Getting Started
1. To send message in ClassChat, type message and press Enter.
2. To send private message, type receiver's name followed by colon and message.
3. To create group, type group followed by space, groupname, space and members separated by comma.
   For example: group group413 Alice,Bob,John,Mary
   ***NOTE: *** groupname should start with word 'group'.
   To send message to group type groupname followed by colon and message.
4. To exit out of chat system, type exit and hit enter.
5. To get this menu again, type menu and press enter.
-----
```

- If Mary no longer wishes to connect to server, she can use *exit* keyword to disconnect herself. The server will be updated as:



```
Mary>exit
Mary disconnected...
```

- When the new clients join the server, they can use usernames Bob, Alice, John, Tom, or Mary as they are no longer connected to the server.
- If the server is closed with clients still having connection with sever or any error is detected, it displays an error message in client’s terminal. For example, Dr.ABC is still connected to the sever. So, when the server window is closed, he will receive the following message.



```
Dr.ABC's window
No connection with server
```

Techniques used to implement the Class Chat System:

1. TCP protocol is used to establish connection between client and server. Host is the local host '127.0.0.1' and port 12000 is used in the system.
2. The concept of threading is used in server.py to handle multiple clients. Threading is also used in client.py to be able to send and receive message at the same time.
3. server.py has four functions.
 - a. Function menu displays menu to clients.
 - b. Function create_group creates the group with given group name and specified active clients and one to listen for clients. If clients are not active, client and server is notified.
 - c. Function group_message handles messaging in group. If someone who is not a group member tries to send messages, it displays an error. Also, if only one person in the group, it deletes the group from server.
 - d. Function listen_for_clients listens for client messages and performs as per the request. Based on the request from client, it either closes the client socket, displays menu, creates group or sends the messages to desired receiver (either individual or group).
When any of the client decide to disconnect with server, the arrays holding client sockets are also updated.
4. client.py also has two functions and they are comparatively very simple. One function is to handle sending message and the other is to handle receiving messages.

This chat system follows the general architecture of client-server communication through socket programming using TCP, that is,

- a. server is started
- b. Client connects to server
- c. Client sends request to server
- d. Server responds to client's request
- e. Close connection when no longer needed.

It can be represented as:

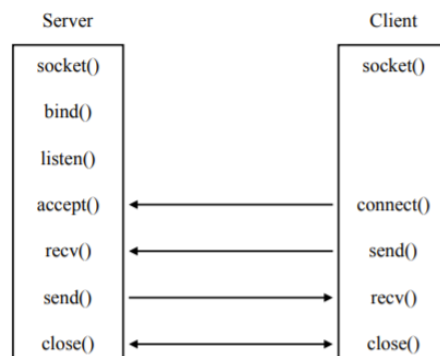


Figure 1: Client-server communication using TCP/IP

Please run the server.py and client.py to test the code.