Internet Relay Chat (IRC) Protocol Specification

Course Name: Internetworking Protocols
Project Name: Internet Relay Chat Project
Team Members:
Jing Li
Deepika Parshvanath Velapure
Saisatvika Gunipati

Table of Contents:

1. Introduction:

IRC is a text-based chat system used for instant messaging. This app supports multiple users communicating in designated communication chat rooms known as ‘Channels’ where they can share various forms of data. An IRC (Internet Relay Chat) protocol has been designed for the smooth functioning of the client server communication. This IRC protocol usually operates on TCP/IP network Protocol. The basic components of this chat system involve a single point server for client communications. This chat app also supports private communication between two users.

2. Basic Information:

2.1 Servers: The server forms the backbone of IRC, providing a point to which clients may connect to talk to each other, and a point for other servers to connect to, forming an IRC network.

2.2 Clients: A client is anything connecting to a server that is not another server. Each client is distinguished from other clients by a unique username having a maximum length of nine (9) characters.

2.3 Room: A user can create a room or join an existing room. Users can create a room by providing the room number using the command Create. (Create <room number). All the users in a room can communicate with each other. When a message is addressed to the room all the users in that room receive the message.

3. Specification:

3.1 Types of communication: This IRC can support one -one communication, one -many communications and one-to-all communications through channels / rooms and personal chat. When a user sends a message the server receives it and forwards it to respective receivers based on the type of communication. This represents dynamic multicasting.

3.2 Character codes: The application can accept ASCII characters of any length as a message. Space or tab can be used as delimiters.

3.3 Messages: Server and clients send messages to each other. Communication is asynchronous in nature between them i. e., the client and server may not receive or generate a reply immediately. Messages are encoding in plain ASCII. This message consists of different (3) parts that are separated by one or more ASCII special characters. The three parts of a message format are prefix (optional only nickname of client), command and the command parameters (up to 15). These messages are exchanged for various purposes like socializing and collaboration among communities.

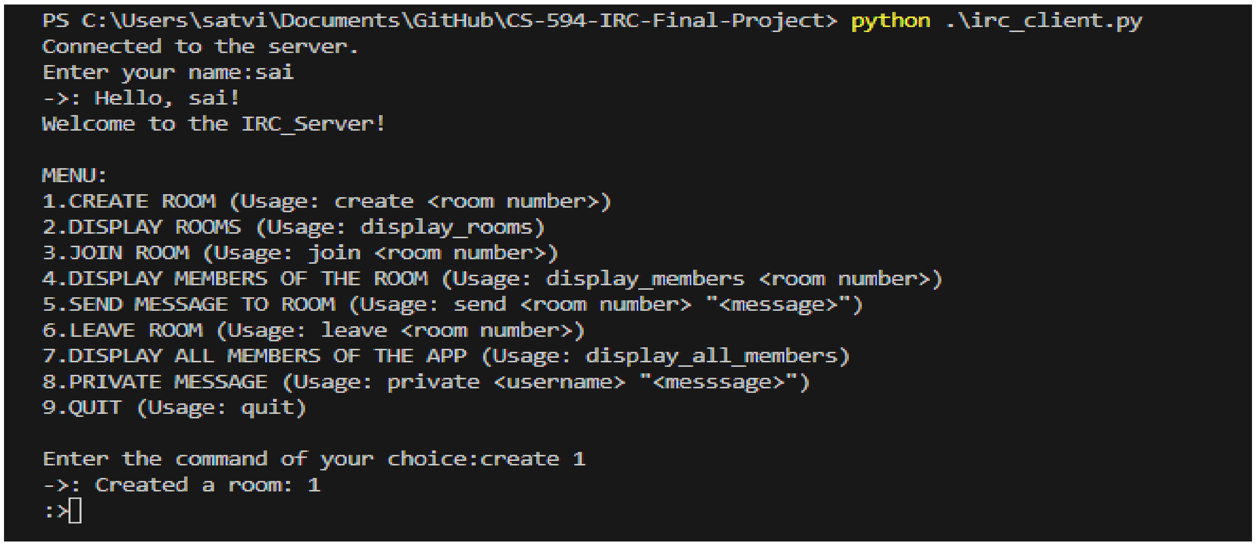
3.4 Replies: The messages sent to the server generate a reply. The IRC reply is a message sent by an IRC server in response to a command issued by any client. IRC replies are an important part of the protocol, as they allow clients to communicate with the server and receive information about the status of their requests.

4. Message Infrastructure:

4.1 Create Room:

Creating rooms refers to the process of setting up a new chat room or channel on an IRC server. To begin, the client server connection must be established. The first client entering the room creates the room. Now this room is available for other users who wish to join. The users in the same room can now communicate with each other.

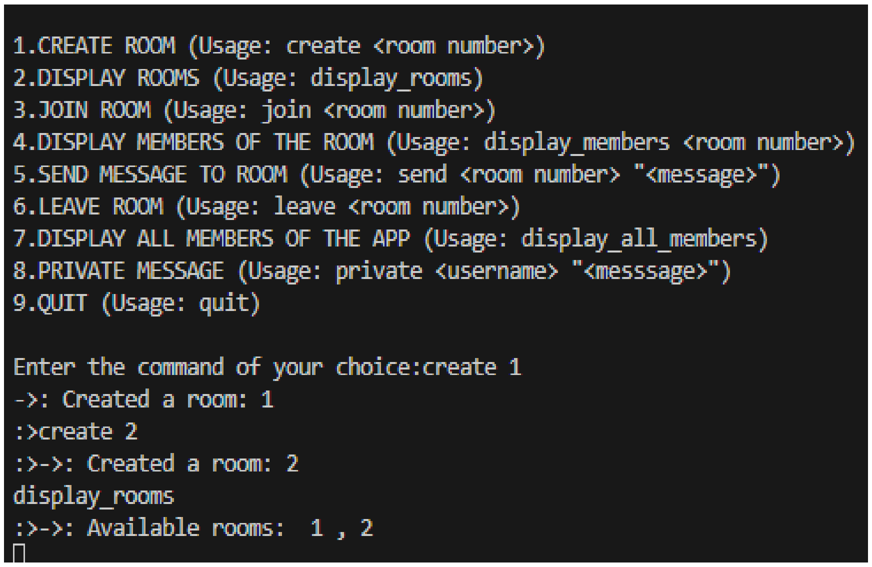
Users can create room using the command “create <room name >”



```
PS C:\Users\satvi\Documents\GitHub\CS-594-IRC-Final-Project> python .\irc_client.py
Connected to the server.
Enter your name:sai
->: Hello, sai!
Welcome to the IRC_Server!

MENU:
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room number>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.QUIT (Usage: quit)

Enter the command of your choice:create 1
->: Created a room: 1
:>
```

4.2 Display Rooms:

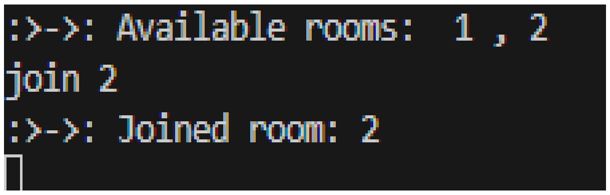
Users can see a display of list of rooms available created by other users. If required, users can create a new room or can join the existing room. Users can display rooms using the command “display_rooms”.



```
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room number>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.QUIT (Usage: quit)

Enter the command of your choice:create 1
->: Created a room: 1
:>create 2
:>->: Created a room: 2
display_rooms
:>->: Available rooms:  1 , 2
```

4.3 Join Room:

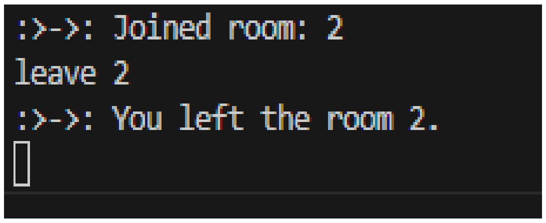
Users can see the list of rooms and join any of the listed rooms. Once the user joins the room, he /she can message them accordingly. Users can even join multiple rooms and send messages to multiple rooms. Users can join any room using the command join<room number>.



```
:>->: Available rooms:  1 , 2
join 2
:>->: Joined room: 2
```

4.4 Leave Room:

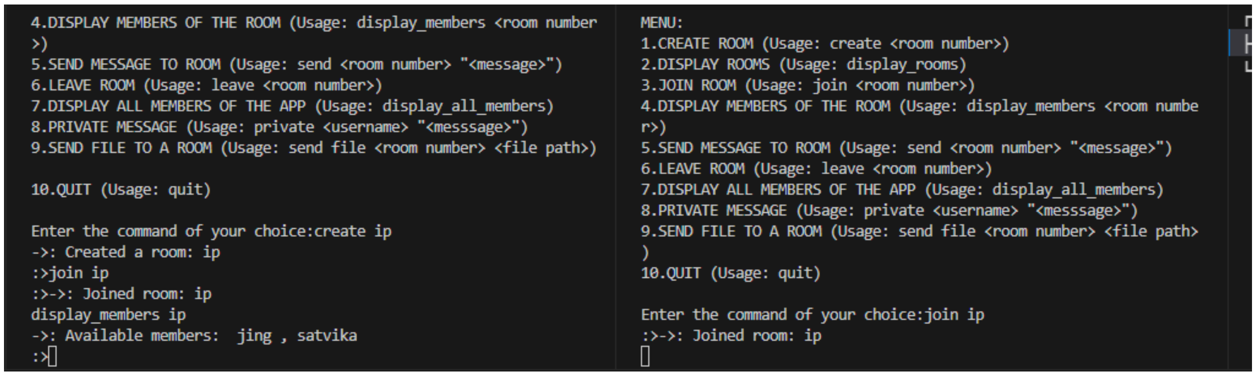
User can see the list of rooms he is part of and can leave the room at any moment of the time. The room will still be available as there are other users in the room. Users can leave the room using the command" leave <room number>"



```
:>->: Joined room: 2
leave 2
:>->: You left the room 2.
```

4.5 Display Members of a room:

Users can see the list of members available in the selected room. Users can use the command "display_members <room number>".



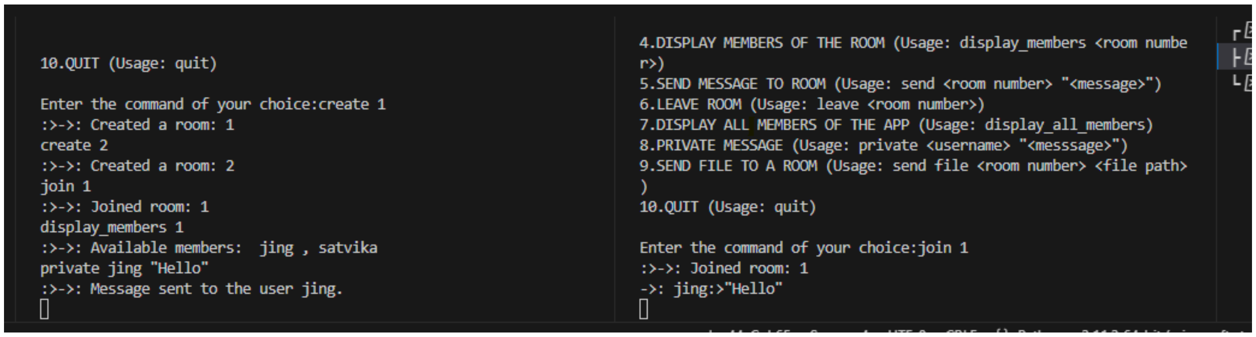
```
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room number
>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room number> <file path>)

10.QUIT (Usage: quit)

Enter the command of your choice:create ip
->: Created a room: ip
:>join ip
:>->: Joined room: ip
display_members ip
->: Available members:  jing , satvika
:>
```

```
MENU:
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room numbe
r>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room number> <file path>
)
10.QUIT (Usage: quit)

Enter the command of your choice:join ip
:>->: Joined room: ip
```

5. IRC Concepts Implementation:

5.1. One-to-one communication (private Messaging):

One to One message usually represents personal chat. When a user sends a message with the nickname of the other user, the server receives the message and forwards the message to the user whose nickname was mentioned. Command used for private messaging is “private <username> <message> “.

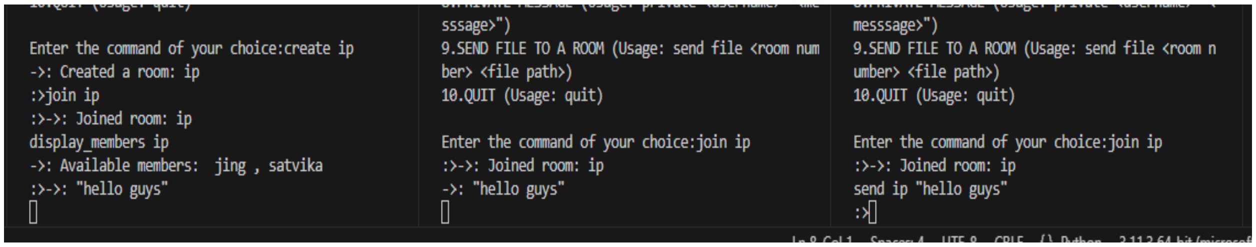


```
10.QUIT (Usage: quit)

Enter the command of your choice:create 1
:>->: Created a room: 1
create 2
:>->: Created a room: 2
join 1
:>->: Joined room: 1
display_members 1
:>->: Available members:  jing , satvika
private jing "Hello"
:>->: Message sent to the user jing.
```

```
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room numbe
r>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room number> <file path>
)
10.QUIT (Usage: quit)

Enter the command of your choice:join 1
:>->: Joined room: 1
->: jing:>"Hello"
```

5.2. One-to-many communication (Broadcast messaging):

One-to-Many communications represents messaging in a particular group or channel consisting of multiple users. When a user sends a message, it is received by the server and then it forwards to all the users of that group. Command used to broadcast the message is send <username>”“<message>”



```
Enter the command of your choice:create ip
->: Created a room: ip
:>join ip
:>->: Joined room: ip
display_members ip
->: Available members:  jing , satvika
:>->: "hello guys"
```

```
sssage>")
9.SEND FILE TO A ROOM (Usage: send file <room num
ber> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:join ip
:>->: Joined room: ip
->: "hello guys"
```

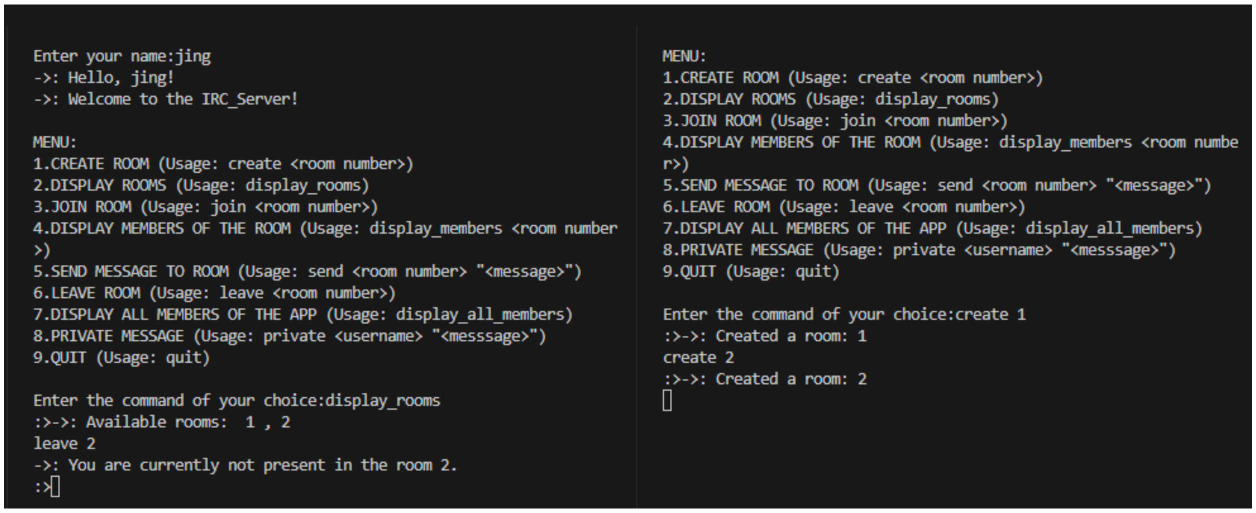
```
messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room n
umber> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:join ip
:>->: Joined room: ip
send ip "hello guys"
:>
```

5.3. One-to-all communication

One-to-All messages are described as broadcast messages sent to all the clients or servers or both. These messages are broadcasted to all the servers so that the state information held by each server is reasonably consistent between servers.

6. Error Scenarios:

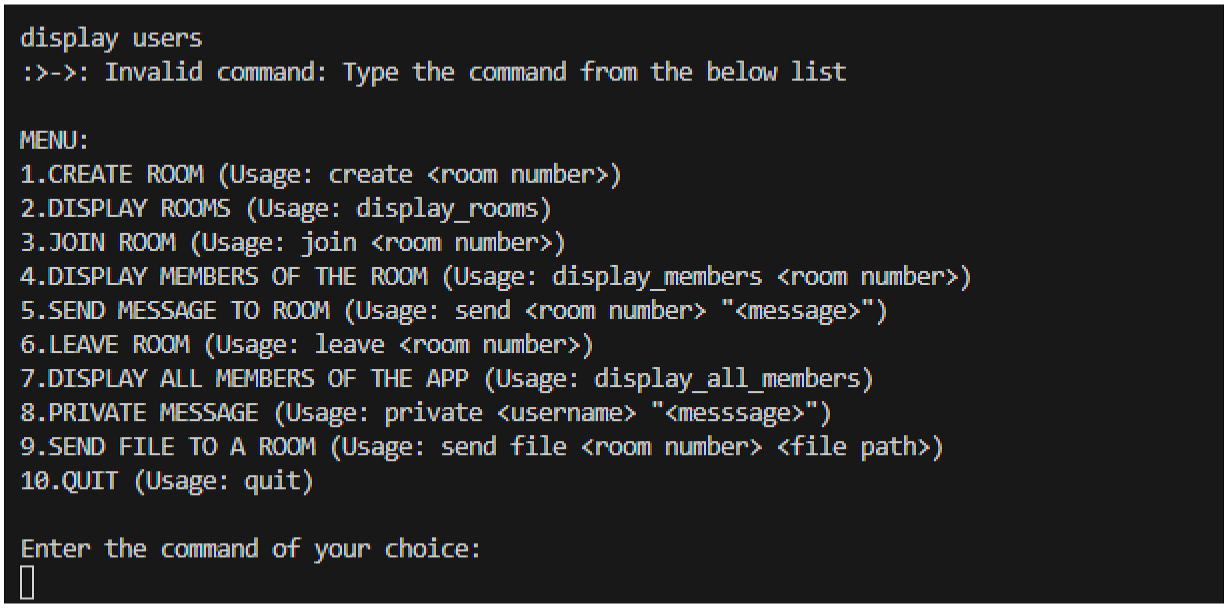
a. When a user tries to leave a room which he is not part of an error message is prompted saying you are not part of this room.



```
Enter your name:jing
->: Hello, jing!
->: Welcome to the IRC_Server!

MENU:
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room number
>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.QUIT (Usage: quit)

Enter the command of your choice:display_rooms
:>->: Available rooms:  1 , 2
leave 2
->: You are currently not present in the room 2.
:>
```

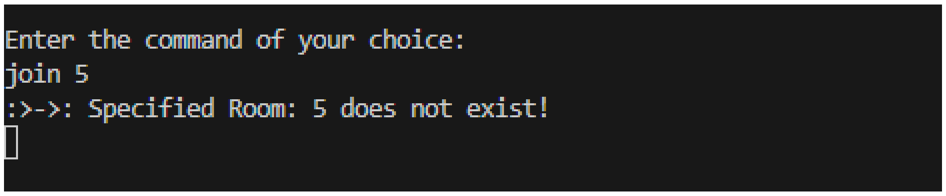
```
MENU:
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room numbe
r>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.QUIT (Usage: quit)

Enter the command of your choice:create 1
:>->: Created a room: 1
create 2
:>->: Created a room: 2
```

b. Server does not recognize the command issued by the client. When server enters the command display users which is not present in the list then error message prompts saying type the command from the below list



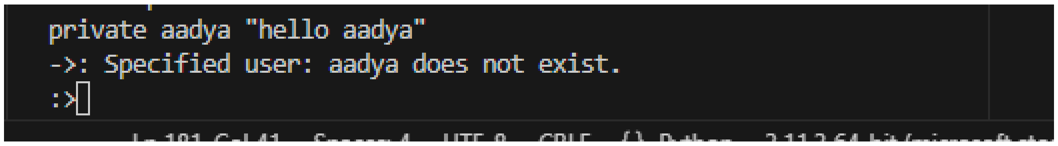
```
display users
:>->: Invalid command: Type the command from the below list

MENU:
1.CREATE ROOM (Usage: create <room number>)
2.DISPLAY ROOMS (Usage: display_rooms)
3.JOIN ROOM (Usage: join <room number>)
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_members <room number>)
5.SEND MESSAGE TO ROOM (Usage: send <room number> "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room number> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:
```

c. When a user tries to join a room which doesn't exist an error message pops saying room doesn't exist.



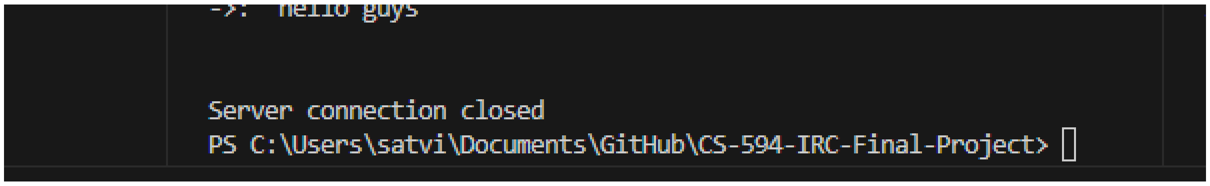
```
Enter the command of your choice:
join 5
:>->: Specified Room: 5 does not exist!
```

d. when user doesn't exist then error message pops saying user doesn't exist.



```
private aadya "hello aadya"
->: Specified user: aadya does not exist.
:>
```

e. If there is a crash on the server connection closes then the users are notified.

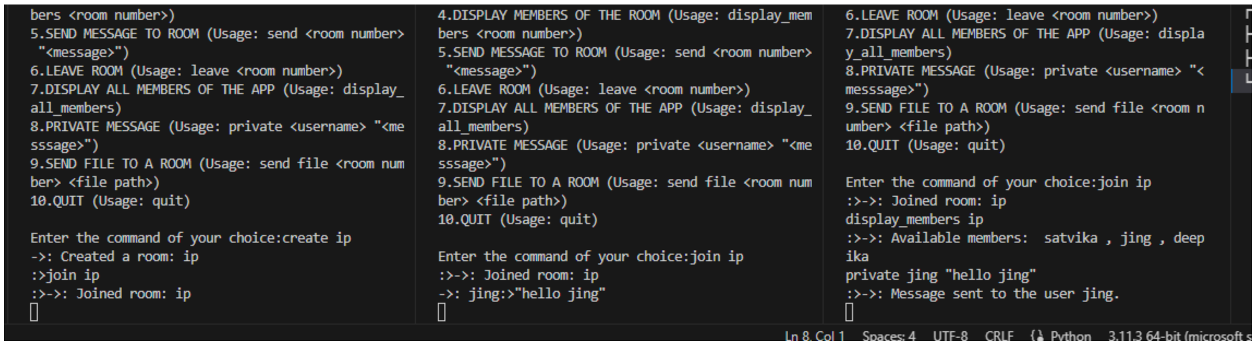


```
->:  hello guys


Server connection closed
PS C:\Users\satvi\Documents\GitHub\CS-594-IRC-Final-Project>
```

7. Extra Features:

Below mentioned extra features will be considered while implementing the IRC protocol:

- **Private messaging between the clients/users:**

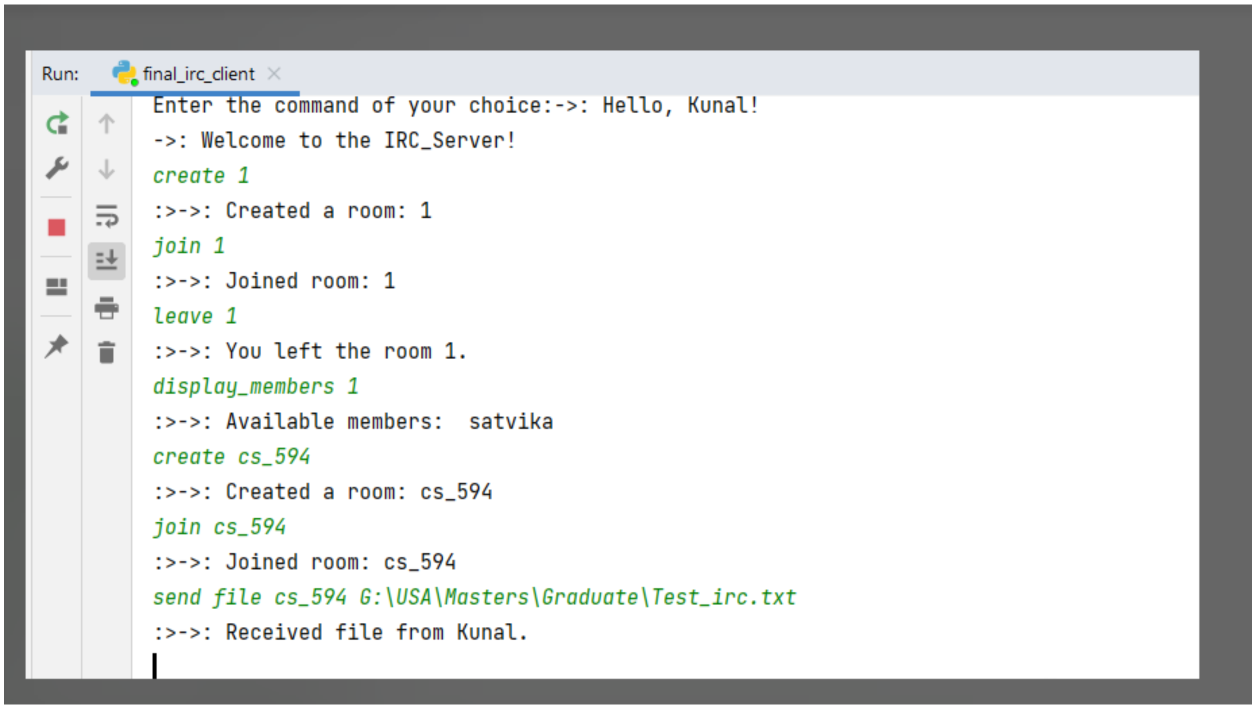


```
bers <room number>)
5.SEND MESSAGE TO ROOM (Usage: send <room number>
 "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_
all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<me
sssage>")
9.SEND FILE TO A ROOM (Usage: send file <room num
ber> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:create ip
->: Created a room: ip
:>join ip
:>->: Joined room: ip
```

```
4.DISPLAY MEMBERS OF THE ROOM (Usage: display_mem
bers <room number>)
5.SEND MESSAGE TO ROOM (Usage: send <room number>
 "<message>")
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: display_
all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<me
sssage>")
9.SEND FILE TO A ROOM (Usage: send file <room num
ber> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:join ip
:>->: Joined room: ip
->: jing:>"hello jing"
```

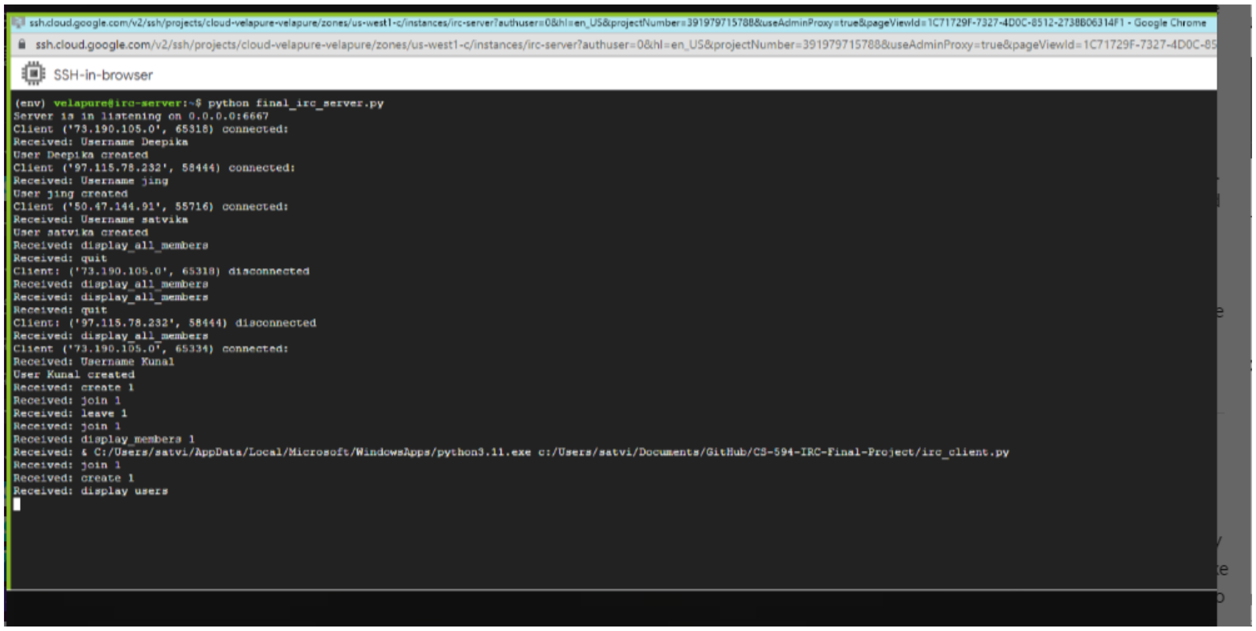
```
6.LEAVE ROOM (Usage: leave <room number>)
7.DISPLAY ALL MEMBERS OF THE APP (Usage: displa
y_all_members)
8.PRIVATE MESSAGE (Usage: private <username> "<
messsage>")
9.SEND FILE TO A ROOM (Usage: send file <room n
umber> <file path>)
10.QUIT (Usage: quit)

Enter the command of your choice:join ip
:>->: Joined room: ip
display_members ip
:>->: Available members:  satvika , jing , deep
ika
private jing "hello jing"
:>->: Message sent to the user jing.
```

- **File Transfer:**



```
Enter the command of your choice:->: Hello, Kunal!
->: Welcome to the IRC_Server!
create 1
:>->: Created a room: 1
join 1
:>->: Joined room: 1
leave 1
:>->: You left the room 1.
display_members 1
:>->: Available members:  satvika
create cs_594
:>->: Created a room: cs_594
join cs_594
:>->: Joined room: cs_594
send file cs_594 G:\USA\Masters\Graduate\Test_irc.txt
:>->: Received file from Kunal.
```

- **Cloud connected Server:**


ssh.cloud.google.com/v2/ssh/projects/cloud-velapure-velapure/zones/us-west1-c/instances/irc-server?authuser=0&hl=en_US&projectNumber=391979715788&useAdminProxy=true&pageViewId=1C71729F-7327-4D0C-8512-2738B06314F1 - Google Chrome
ssh.cloud.google.com/v2/ssh/projects/cloud-velapure-velapure/zones/us-west1-c/instances/irc-server?authuser=0&hl=en_US&projectNumber=391979715788&useAdminProxy=true&pageViewId=1C71729F-7327-4D0C-85
SSH-in-browser

```
(env) velapure@irc-server:~$ python final_irc_server.py
Server is in listening on 0.0.0.0:6667
Client ('73.190.105.0', 65318) connected:
Received: Username Deepika
User Deepika created
Client ('97.115.78.232', 58444) connected:
Received: Username jing
User jing created
Client ('50.47.144.91', 55716) connected:
Received: Username satvika
User satvika created
Received: display_all_members
Received: quit
Client: ('73.190.105.0', 65318) disconnected
Received: display_all_members
Received: display_all_members
Received: quit
Client: ('97.115.78.232', 58444) disconnected
Received: display_all_members
Client ('73.190.105.0', 65334) connected:
Received: Username Kunal
User Kunal created
Received: create 1
Received: join 1
Received: leave 1
Received: join 1
Received: display_members 1
Received: & C:/Users/satvi/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/satvi/Documents/GitHub/CS-594-IRC-Final-Project/irc_client.py
Received: join 1
Received: create 1
Received: display users
```

8. Conclusion:

IRC is a popular messaging app meeting different needs but there are some issues related to scalability, security and flexibility working with other systems. More improvements, including creating new rules and standards, are required to face these challenges. Besides this IRC remains an efficient messaging app for users online.