CS 594: Internetworking Protocols – FALL 2024

Internet Relay Chat Protocol

Team Members:

Vinodh Raju, PSU ID: 914214354

Siddhanth Darshan Jain Gouder Nagpal, PSU ID: 965786213

Kanishka Kopperla, PSU ID: 945991088

Memo Overview:

This project is part of the course in Internetworking Protocol and reflects a deep understanding of the core concepts of the subject. This document has been prepared for examination and review purposes, and any feedback is welcome for the better enhancement of the model. We are committed to the delivery of a quality project and appreciate all the support in making the project better.

Abstract:

Internet Chat Relay is a flexible and efficient client-server architecture designed to facilitate seamless communication between clients and servers through socket programming. This text-based protocol allows users to exchange private messages with ease, while supporting concurrent connections from multiple clients to a single server. It allows users to create, join, leave, and broadcast messages within chat rooms for an interactive and engaging communication experience.

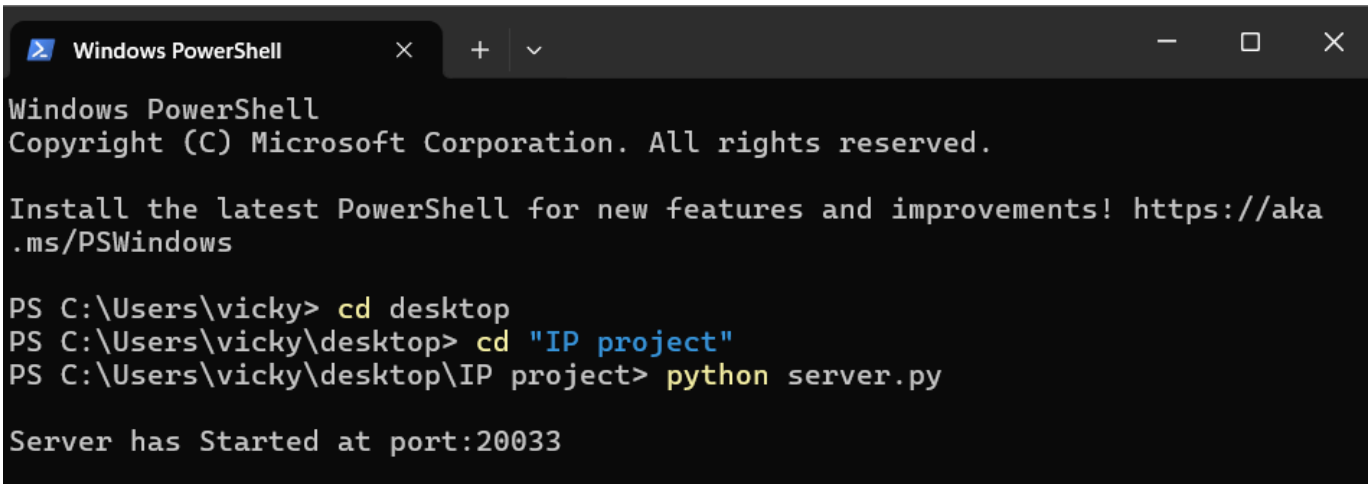
Table of Contents:

1. Introduction

Socket programming is used for the establishment of a relationship between a server and its client. After the server is started, it listens to a client connection. When the request of a client is received, it accepts the request, and then communication can be started. Various clients connected to the server may send messages to each other for effective and interactive communication.

1.1 Servers

Socket programming enables connecting to multiple clients on a predefined port. The server address includes an IP address and port number, either user-defined or default-which is port 1024. Once a client logs onto the specified port, it reflects on the server interface. Furthermore, in the creation of a chat room, all commands run by the connected clients are monitored and logged by the server. Socket programming uses the `bind` method for address assignment, the `listen` method to process the incoming client request, and the `accept` method for establishing the connection to manage the connections of servers.



1.2 Clients

Any user who connects to the server is considered a client, and each client is given a unique name for differentiation. Multiple or single threads can connect to the server by specifying the same port or defaulting to port 1024 if no port is provided. After establishing a connection, clients can select a username, which the server uses to track messages and manage threads effectively.

1.3 Rooms

It allows clients to create chat rooms that support multiple members, enabling group communication. When a client creates a room, they automatically become added as a member. Other clients can view the list of available rooms on the server and join any room that they prefer. Additionally, clients are able to leave rooms they have joined, list the members of their current room, and broadcast messages to all rooms they are part of for increased flexibility and interactivity in the chat system.

2. Conventions used in this document

This text-based communication system is built upon a client-server architecture that depends on command prompt interactions. Upon establishing a connection with the server, a number of functionalities are made available to clients and can be activated via specific commands. A user can execute any of these commands starting with `$command_name` along with parameters. Listed here is the set of commands along with the description of the functionalities a client will be presented with once a connection is established with the server.

```
                          IRC COMMANDS
Below are the list of commands

$create room_name                 : To create a new room
$join room_name                   : To join a room
$switch                           : To switch rooms
$list room_name                   : To list members of a room
$listAll                          : To list all the available rooms and clie
nts
$send_room room_name message      : To send a message to specified rooms
$menu                             : To print menu
$private_message username message : To send a direct personal message
$broadcast_everyone message       : To broadcast message to everyone
$leave                            : To leave current room
$bye                              : To exit
```

3. Basic Information

It utilizes the TCP/IP three-way handshake protocol to establish reliable connections. The clients connect to the server using the IP address and port number of the server. Once the connection is established, the client can send messages and requests to the server using the open channel and receive responses from the server through the same channel. Communication occurs entirely over TCP/IP, with the server continuously listening for incoming connections on a specified port. Clients connect to this port to maintain a persistent connection. The client-server model operates asynchronously, allowing clients to send requests as needed while the server processes them

immediately. The messaging protocol also follows an asynchronous approach, ensuring efficient communication.

4. Messaging Infrastructure

A client can use the following command formats for sending messages:

```
                          IRC COMMANDS
Below are the list of commands

$create room_name                 : To create a new room
$join room_name                   : To join a room
$switch                           : To switch rooms
$list room_name                   : To list members of a room
$listAll                          : To list all the available rooms and clie
nts
$send_room room_name message      : To send a message to specified rooms
$menu                             : To print menu
$private_message username message : To send a direct personal message
$broadcast_everyone message       : To broadcast message to everyone
$leave                            : To leave current room
$bye                              : To exit
```

5. Client and Server Messages

5.1 Establishing the connection

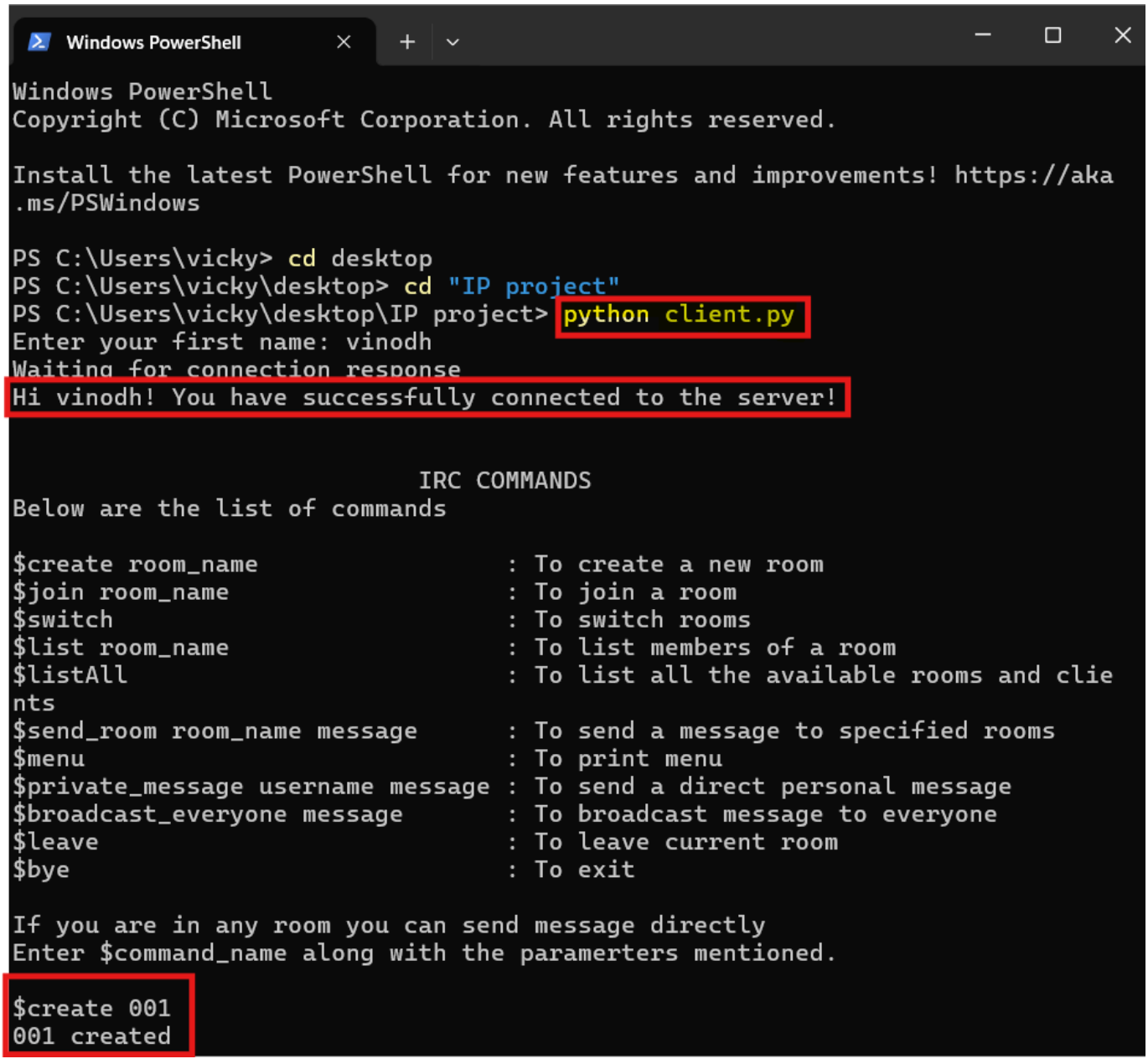
To establish a connection, the host needs to initiate the server application and run it. Once a valid connection is established, numerous clients can be connected to the server. Clients connected are able to broadcast messages to all the other clients.

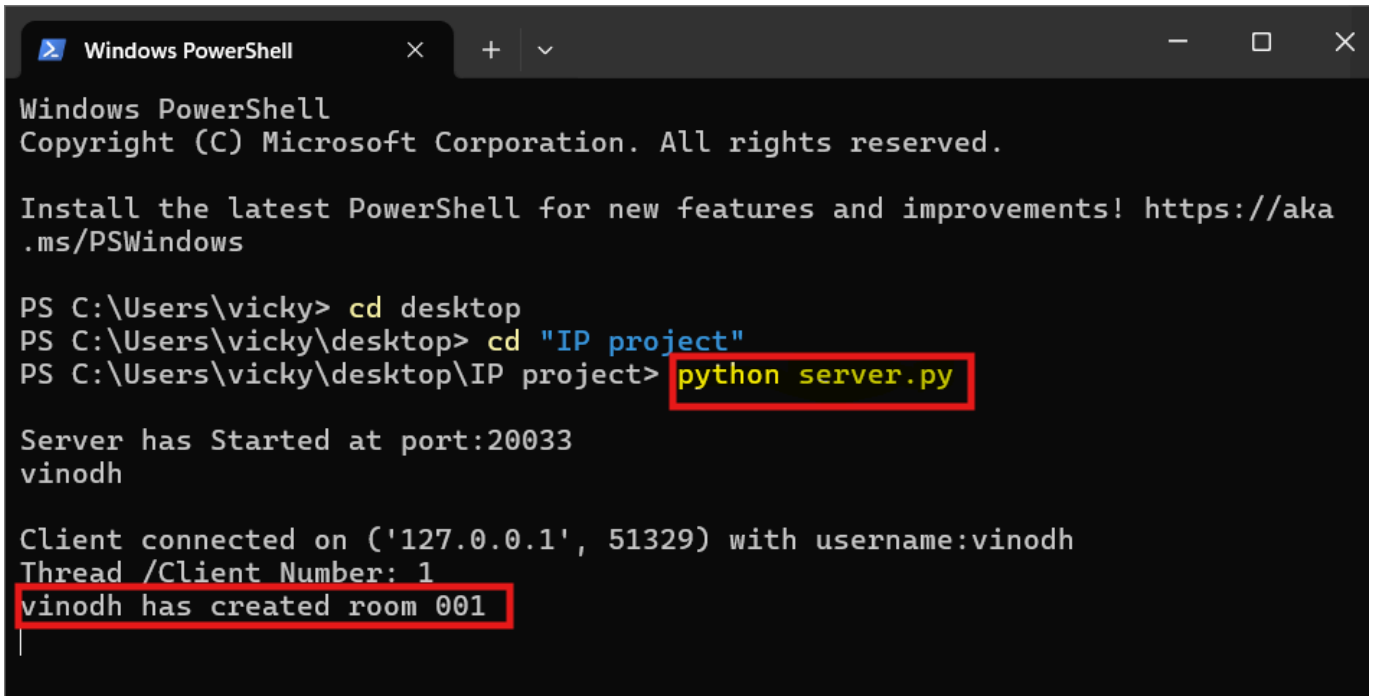
```
PS C:\Users\vicky> cd desktop
PS C:\Users\vicky\desktop> cd "IP project"
PS C:\Users\vicky\desktop\IP project> python client.py
Enter your first name: vinodh
Waiting for connection response
Hi vinodh! You have successfully connected to the server!
```

5.2 Creating Room

After the connection is established, the user can create a room by using the command below. The client who created the room automatically becomes a member of that room and does not need to join it separately.

$CREATE ROOMNAME





5.3 Joining the Room

Once a room is created, other clients can join the room using the following command. All previously joined clients of the same room will receive a confirmation message when someone joins.

$JOIN ROOMNAME

```
Client connected on ('127.0.0.1', 51413) with username:vinodh
Thread /Client Number: 1
vinodh has created room 001
vinodh has joined room 001
```

The below figure shows confirmation to other users joins the room.

```
Client connected on ('127.0.0.1', 51417) with username:siddhant
Thread /Client Number: 2
siddhant has joined room 001
```

```
Client connected on ('127.0.0.1', 51418) with username:kanishka
Thread /Client Number: 3
kanishka has joined room 001
```

5.4 Switch room

With the `$switch` command, clients can switch between rooms without problems. The main idea behind this functionality is to broadcast messages in one particular group or leave one group to join another. If a client attempts to switch to a room that they are already a member of, a message will be shown stating they are already part of the room.

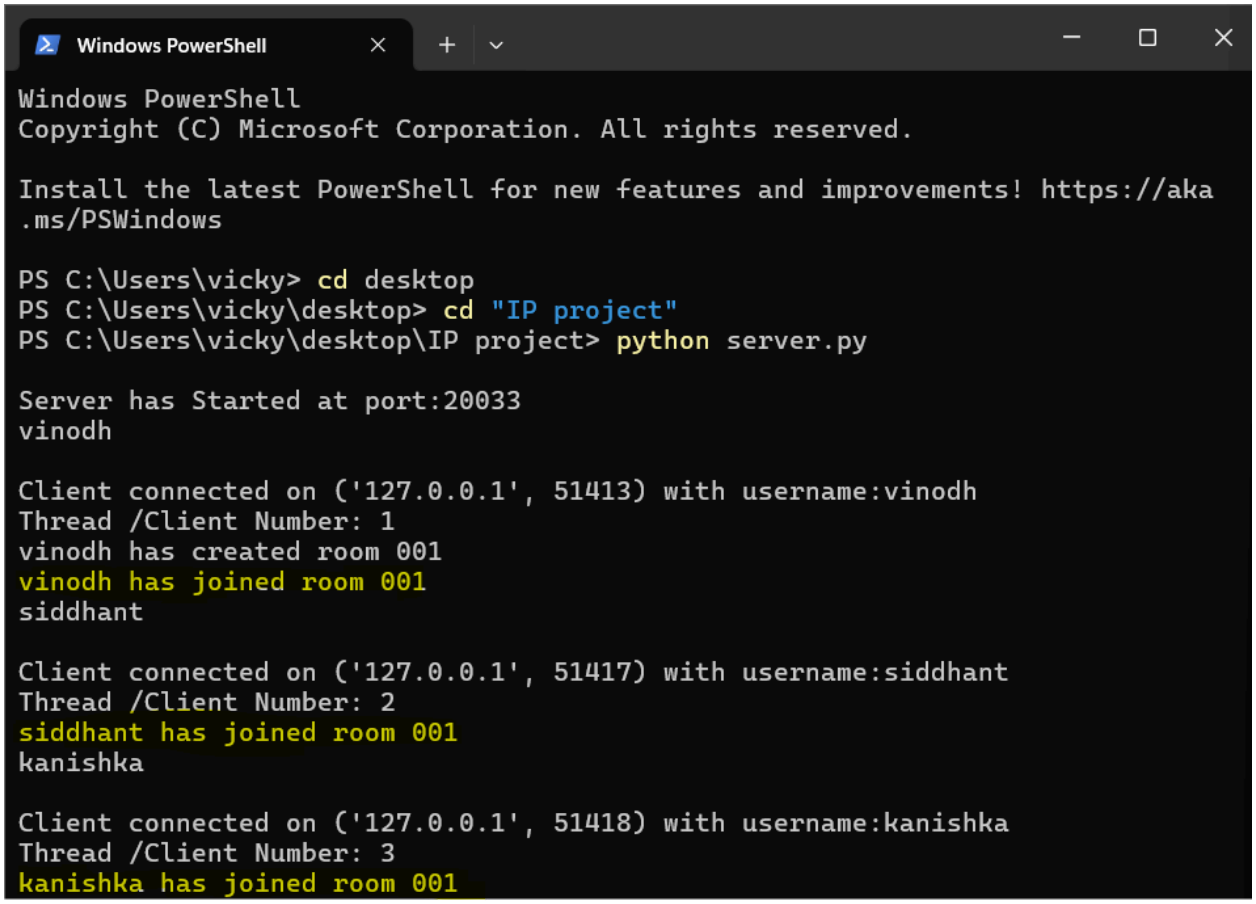
```
Members of Room '002':

vinodh

$switch 001
youve been switched successfully to the room
```

5.5 List Members of a room

The list of members in a particular room can be viewed by the clients using the `$list` command followed by the name of the room. This command will list all the room names along with the members present in each room, giving a clear view of who is active in the system.

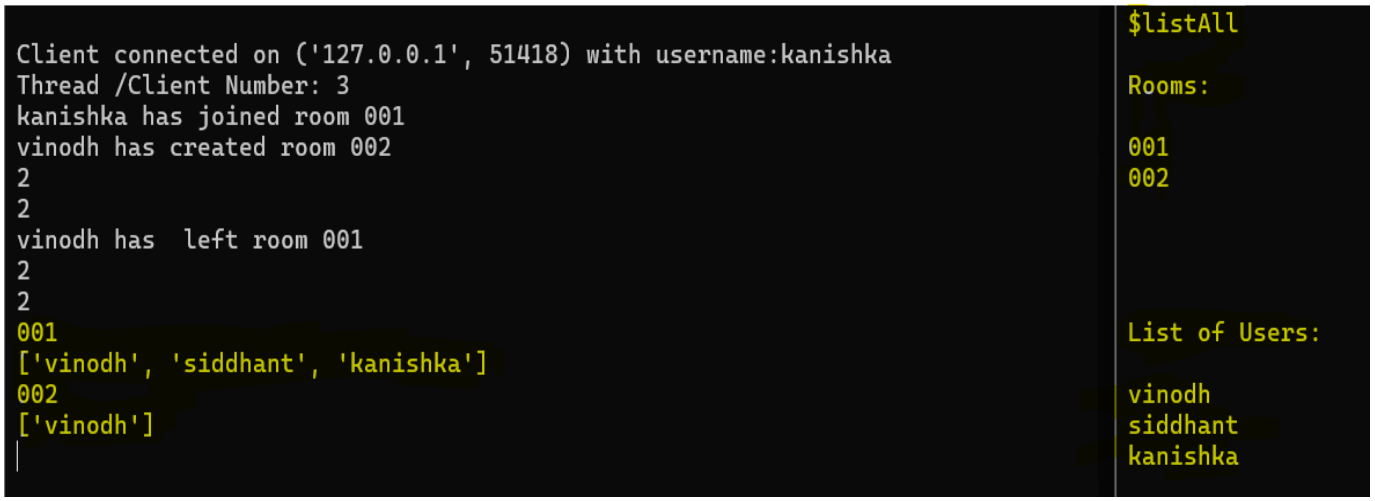
```
$list 001



Members of Room '001':

vinodh
siddhant
kanishka
```

5.6 Multiple clients can connect to a server:



5.7 List all members and room names:

The $listAll command helps clients to list all the members connected with the server and all the names of rooms available. Consequently, this helps the client to enter a room or message with anybody on the server easily.



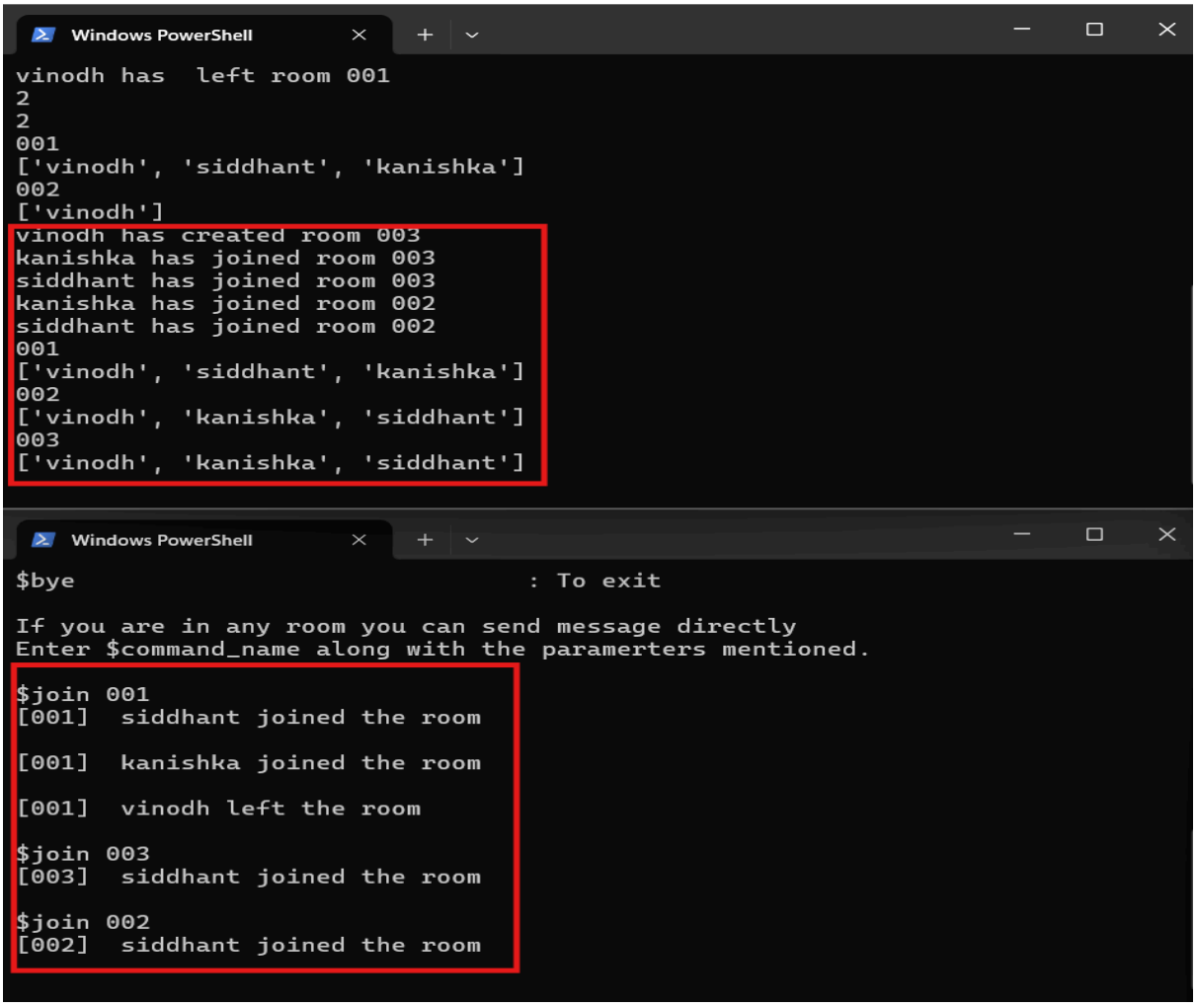
5.8 Leave a Room

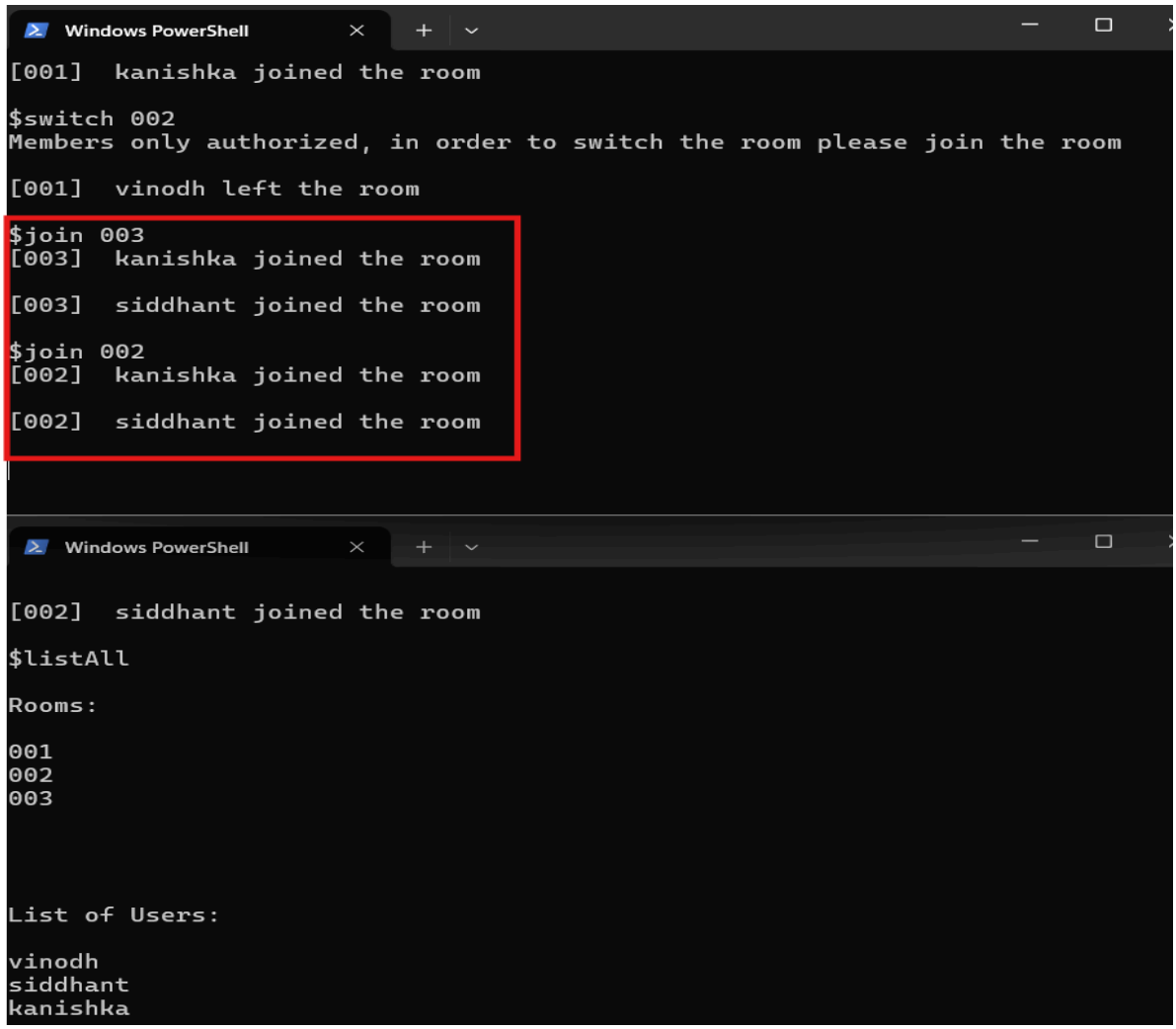
The client can leave a room by first switching to it using the `$switch` command followed by the room's name. Once the client has successfully switched to the room, they can exit the room by using the `**$leave**` command.

```
$leave
You left the room

$leave
You are not part of any room
```

5.9 Client can join multiple (selected) rooms -

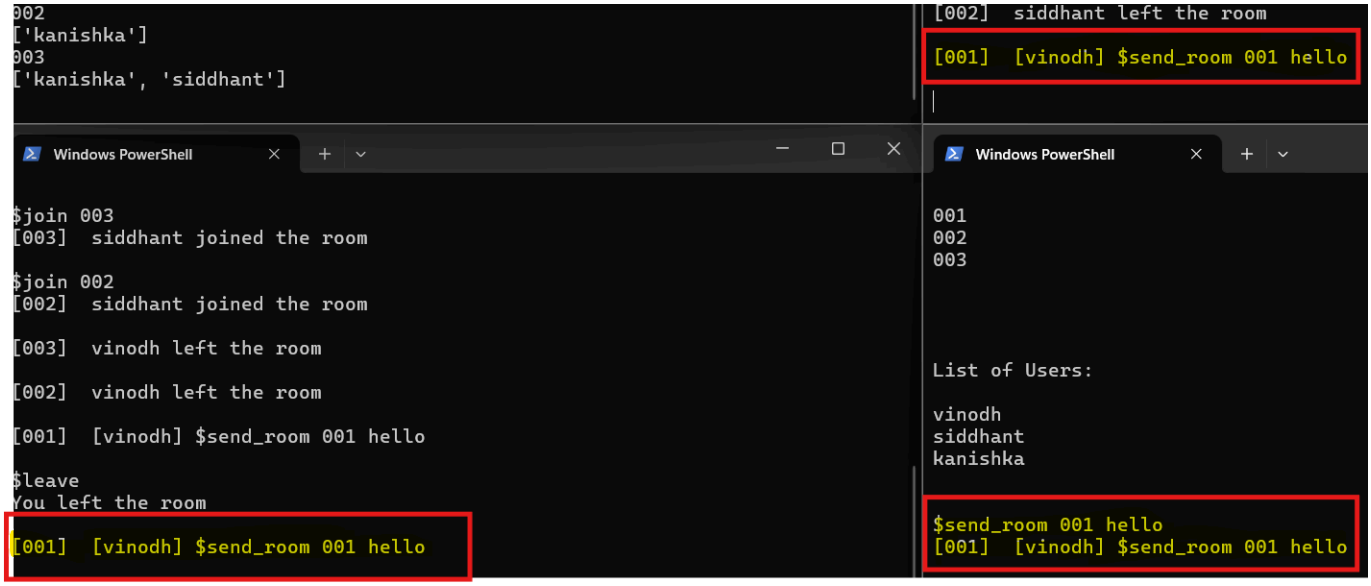
Clients are able to join multiple rooms using **$join room_name** command




Windows PowerShell
[001] kanishka joined the room
$switch 002
Members only authorized, in order to switch the room please join the room
[001] vinodh left the room
$join 003
[003] kanishka joined the room
[003] siddhant joined the room
$join 002
[002] kanishka joined the room
[002] siddhant joined the room
Windows PowerShell
[002] siddhant joined the room
$listAll
Rooms:
001
002
003
List of Users:
vinodh
siddhant
kanishka

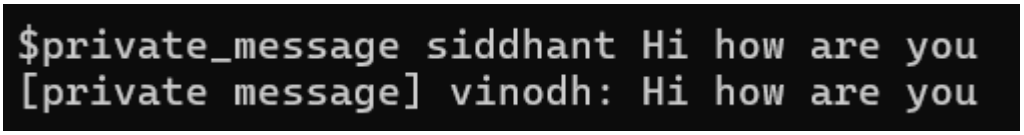
5.10 Send distinct message to multiple rooms

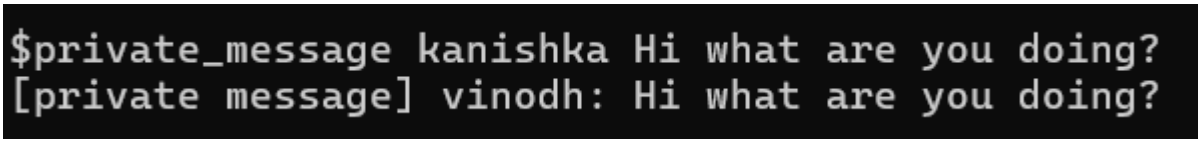
It will allow clients to send messages with the `$broadcast_message` command, followed by the room name and the message. The command above sends a specified message to all members in a selected room. A client should target a room and a message, such as `**$broadcast_message roomname message**`.



5.11 Send Private Message

This application allows for one-to-one communication where a client can send a message directly to another client without it being broadcasted to other members or rooms. To send a private message, the client can use the **$private_message** command followed by the intended recipient's name and the message. If the specified user does not exist, an error message will be displayed indicating that the user was not found.





5.12 Send a broadcast message to all the clients

A broadcast message is sent across to all the clients that are currently connected with the server. It will be received by all the active clients irrespective of which room they are in. Each client that receives the broadcast will also know which client sent the message.

$broadcast_everyone message

```
$broadcast_everyone Hello all
[Broadcast message from vinodh] Hello all
```

```
[001]  [vinodh] $send_room 001 hello
```

```
[Broadcast message from vinodh] Hello all
```

5.13 Menu

Clients have the ability to view a list of available commands at any time by using the **$menu** command on the command prompt.

```
$menu
                          IRC COMMANDS
Below are the list of commands

$create room_name                 : To create a new room
$join room_name                   : To join a room
$switch                           : To switch rooms
$list room_name                   : To list members of a room
$listAll                          : To list all the available rooms and clie
nts
$send_room room_name message      : To send a message to specified rooms
$menu                             : To print menu
$private_message username message : To send a direct personal message
$broadcast_everyone message       : To broadcast message to everyone
$leave                            : To leave current room
$bye                              : To exit

If you are in any room you can send message directly
Enter $command_name along with the paramerters mentioned.
```

5.14 Disconnect from server

Clients have the option to disconnect from the server by using the **$bye** command. Once the client has disconnected, the server will display a message indicating that the client has left the room. Clients who have disconnected will no longer receive any further communication from the server or other clients connected to that server. Additionally, the room members who were in the room with the disconnected client will receive a message notifying them that the client has left the room.

6. Error Handling:

All negative cases are handled using error handling. If a user encounters an error, an exception will be thrown. Below are a few examples of when error handling is utilized: -

When a user attempts to switch to a room they are not a member of.

```
$switch 002
Members only authorized, in order to switch the room please join the room
```

If the client is trying to send message to user that doesn't exists, user gets user not found error

```
$private_message kanis Hi what are you doing?
User not found
```

7. Additional Functionalities

Under extra credits we have implemented sending a private message to clients.

8. Security Considerations

This document is a basic standard for communication between multiple clients via a centralized forwarding server using TCP. It is primarily a text-based protocol with the facility for protocol extensions via a scheme of keywords based on single words or a combination of words with an optional parameter. This memo describes the basic specification of the Internet Relay Chat (IRC) Project, a framework through which several clients can interact with each other via a server, with an added functionality to suit the particular application.

9. Conclusion

The following are some of the limitations of the current IRC protocol implementation. This protocol does not support multiple servers running at the same time, unlike the RFC 1459 standard. The private messaging feature can be improved to at least ensure that the message is encrypted and can only be viewed by the intended client. In the future, features that can be added include login credentials for channel access and the ability to share files or media.

10. Future work:

There are various shortcomings in the IRC protocol. Unlike the RFC 1459, this system doesn't have support for use with many servers operating together. It could also work on better development with its private message to ensure encrypted messages will be open to the client end. More enhancements could be made further on login credentials for the channels and addition of file/media sharing options.

11.Reference

Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

https://datatracker.ietf.org/doc/html/rfc2119
https://en.wikipedia.org/wiki/Internet_Relay_Chat

12. Acknowledge

For these project deliverables, we used Google resources and the Google product suite. For the design, we also researched Wikipedia and other sources.