Server description:

Upon server startup, firstly the servers and the rooms are set up. 3 rooms are created by creating a <int <vector<SOCKET>> pair with an id and empty SOCKET vector for the client sockets. A thread is created and detached for each room as well.

```
std::map<int, std::vector<SOCKET>> rooms;
rooms.insert(std::make_pair(0, std::vector<SOCKET>()));
rooms.insert(std::make_pair(1, std::vector<SOCKET>()));
rooms.insert(std::make_pair(2, std::vector<SOCKET>()));
```

After room setup is complete, the server is set up and it begins listening for connections on port 8080. Once a new client connection has been accepted, a new thread is created for the connected client. When a new thread for the client is being created, the handleClient function is called, with the clientSocket variable passed to it as an argument. The clientSocket is added to the appropriate room vector, according to the user's selected room id. The client can leave by typing "/leave" and can join another room by entering an id 0 to 2 right after leaving a room.
Each time a client sends a new message into any of the rooms, it is added to the messageQueue. When the messageQueue is not empty, the threads created to handle rooms are notified, after which they send out the message to all the appropriate clients, ergo, to all the clients in the room except the sender.

Client description:

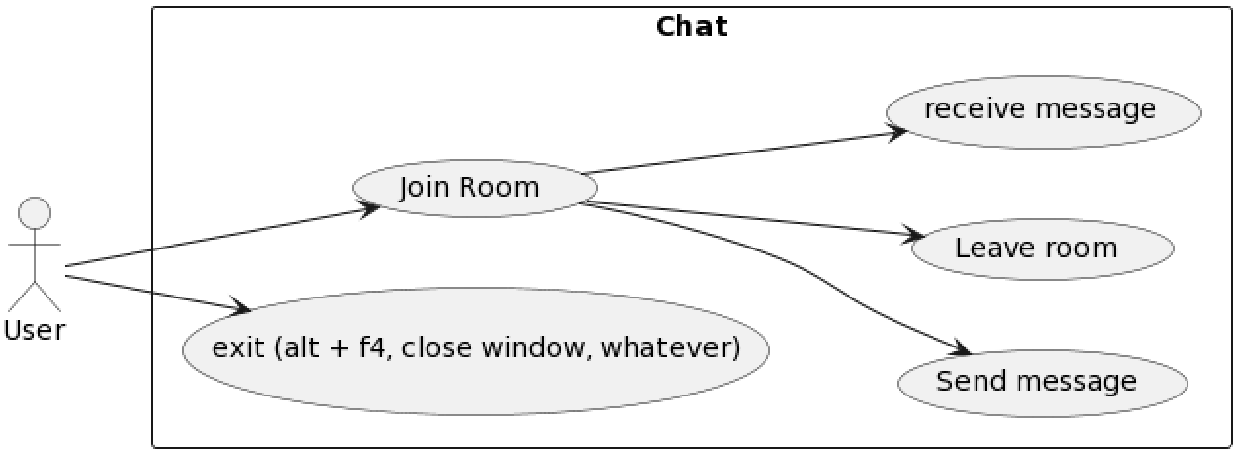
Upon client startup, the client is set up and tries to connect to the server port 8080. After connecting successfully, the client creates a new thread for receiving messages, the main thread is responsible for sending messages. After that, the client must send an id for the room they want to join. They can't proceed without entering the room id. After the room has been picked, the client can begin to send messages. The user can leave and rejoin rooms at will.

Application communication protocol:

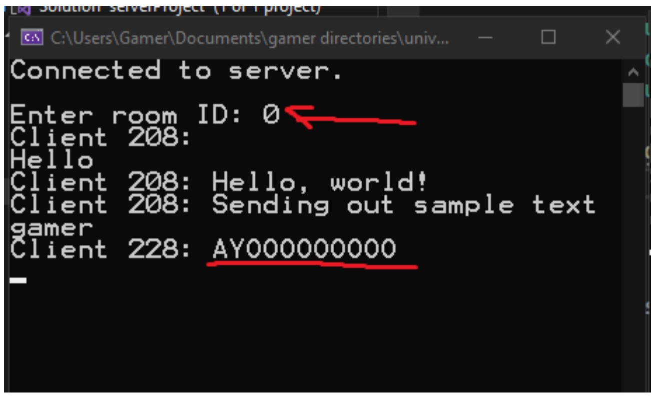
The client sends 2 bytes, containing the user's desired room id as a char.
The server receives the 2 bytes, processes the id by casting it into an int.
Afterwards the server adds the client to the appropriate room by adding their socket to the vector<SOCKET> of the desired room.
The client can receive a message of up to 4096 bytes, not including the null terminator. Message of the size 4096 is also the largest possible message the server can process. The client application can send a message of up to 2,049 bytes, since the amount of bytes sent by the client is bound to the maximum size of a string in C++.

After the client has joined a room, they can communicate with other clients in that room by sending messages through the client. This message will then be broadcasted to all the other clients in that room, except for the sender.

Use case diagram:

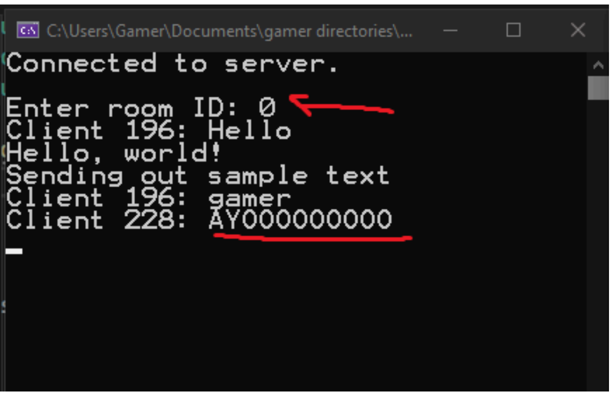


Different program use cases:



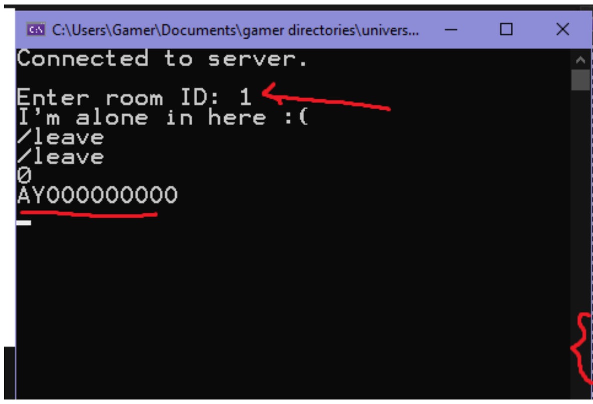
```
Connected to server.

Enter room ID: 0
Client 208:
Hello
Client 208: Hello, world!
Client 208: Sending out sample text
gamer
Client 228: AY000000000
```



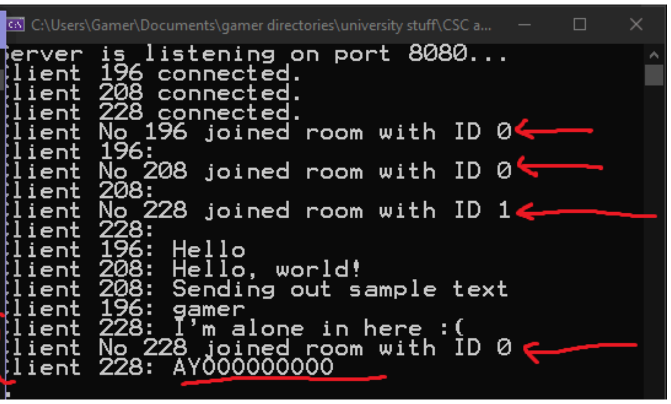
```
Connected to server.

Enter room ID: 0
Client 196: Hello
Hello, world!
Sending out sample text
Client 196: gamer
Client 228: AY000000000
```



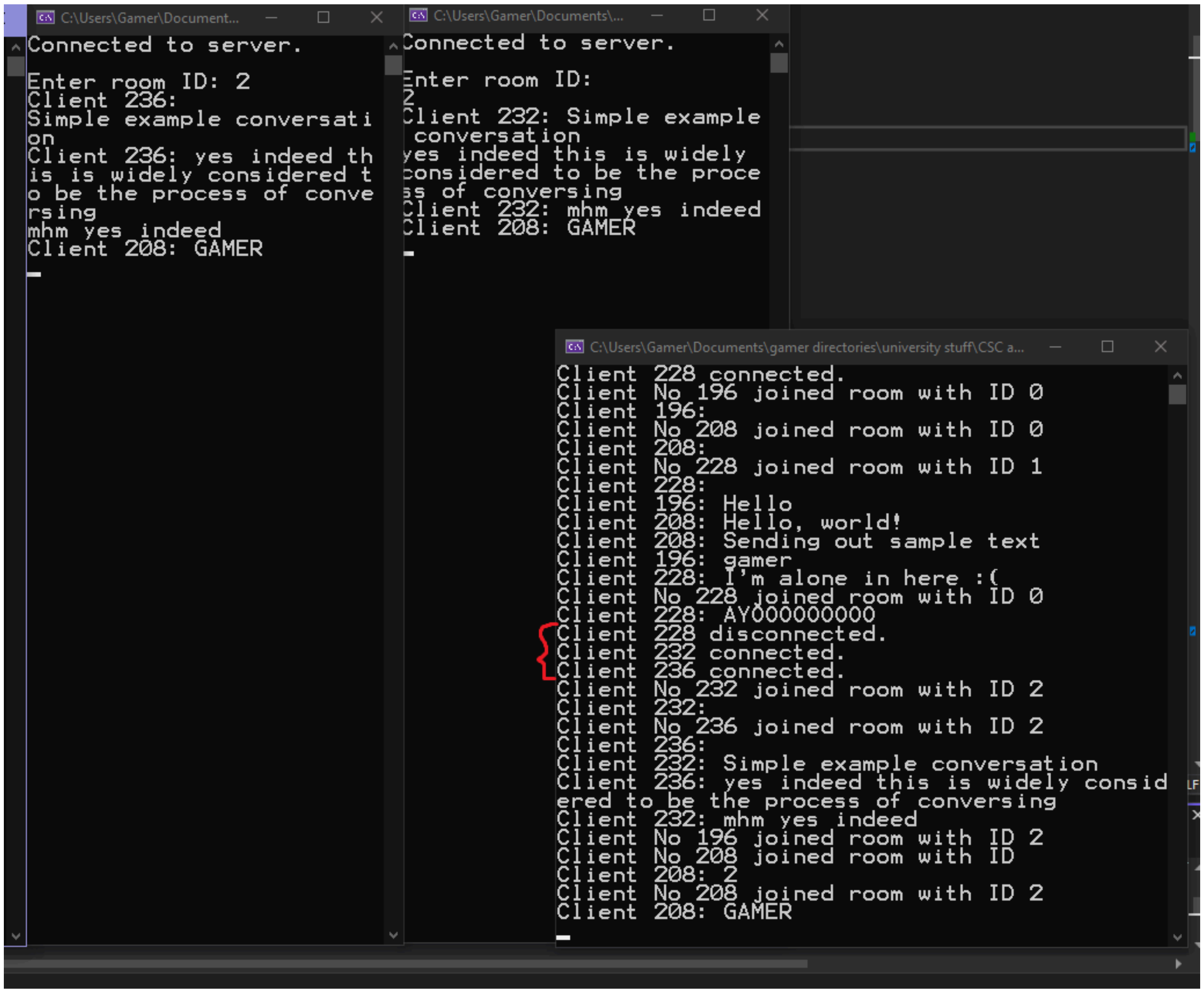
```
Connected to server.

Enter room ID: 1
I'm alone in here :(
/leave
/leave
0
AY000000000
```


C:\Users\Gamer\Documents\gamer directories\university stuff\CSC a...

```
erver is listening on port 8080...
lient 196 connected.
lient 208 connected.
lient 228 connected.
lient No 196 joined room with ID 0
lient 196:
lient No 208 joined room with ID 0
lient 208:
lient No 228 joined room with ID 1
lient 228:
lient 196: Hello
lient 208: Hello, world!
lient 208: Sending out sample text
lient 196: gamer
lient 228: I'm alone in here :(
lient No 228 joined room with ID 0
lient 228: AY000000000
```

Here 3 clients connect to the server. The first 2 clients connect to room 0, while the 3rd one connects to room 1. The clients in room 0 talk for a bit, then client 228 leaves room 1 and joins room 0 to talk with the other clients.


C:\Users\Gamer\Document...
C:\Users\Gamer\Documents\...
C:\Users\Gamer\Documents\gamer directories\university stuff\CSC a...

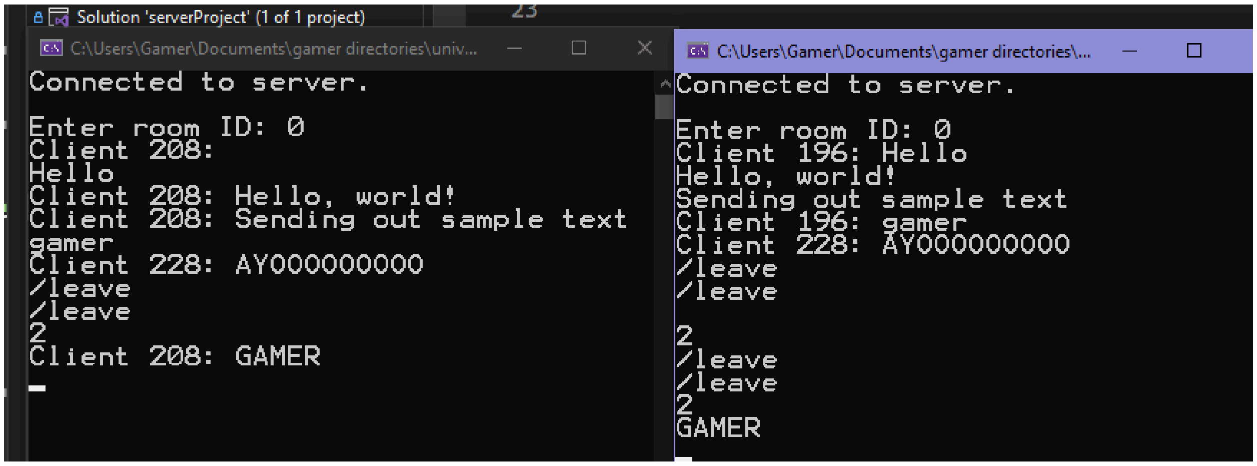
```
Connected to server.

Enter room ID: 2
Client 236:
Simple example conversati
on
Client 236: yes indeed th
is is widely considered t
o be the process of conve
rsing
mhm yes indeed
Client 208: GAMER
```

```
Connected to server.

Enter room ID:
2
Client 232: Simple example
 conversation
yes indeed this is widely
considered to be the proce
ss of conversing
Client 232: mhm yes indeed
Client 208: GAMER
```

```
Client 228 connected.
Client No 196 joined room with ID 0
Client 196:
Client No 208 joined room with ID 0
Client 208:
Client No 228 joined room with ID 1
Client 228:
Client 196: Hello
Client 208: Hello, world!
Client 208: Sending out sample text
Client 196: gamer
Client 228: I'm alone in here :(
Client No 228 joined room with ID 0
Client 228: AY000000000
Client 228 disconnected.
Client 232 connected.
Client 236 connected.
Client No 232 joined room with ID 2
Client 232:
Client No 236 joined room with ID 2
Client 236:
Client 232: Simple example conversation
Client 236: yes indeed this is widely consid
ered to be the process of conversing
Client 232: mhm yes indeed
Client No 196 joined room with ID 2
Client No 208 joined room with ID
Client 208: 2
Client No 208 joined room with ID 2
Client 208: GAMER
```


Solution 'serverProject' (1 of 1 project)
C:\Users\Gamer\Documents\gamer directories\univ...
C:\Users\Gamer\Documents\gamer directories\...

```
Connected to server.

Enter room ID: 0
Client 208:
Hello
Client 208: Hello, world!
Client 208: Sending out sample text
gamer
Client 228: AY000000000
/leave
/leave
2
Client 208: GAMER
```

```
Connected to server.

Enter room ID: 0
Client 196: Hello
Hello, world!
Sending out sample text
Client 196: gamer
Client 228: AY000000000
/leave
/leave

2
/leave
/leave
2
GAMER
```

These screenshots continue the previous example. One of the users from the previous example disconnects by closing the program window. In his place 2 new clients join. They join room 2, talk for a bit there and then the other 2 clients join room 2 (4 clients in total). Then one of the clients sends a message to the room and in the screenshot it can be seen that everyone received the message.