



Group 26

AMCGUS001
MRJRAH001
VLMTAB001

Networking Assignment
CSC3002F

Table of Contents

GROUP 26 REPORT	3
1. BACKGROUND	3
2. FEATURES	3
3. CONSTRAINTS:	5
4. PROTOCOL	5
5. CHATROOM SCREENSHOTS	8

Group 26 Report

1. Background

Network Applications can be defined as a client-server architecture where each process or computer connected to the specified network is a client or server. Essentially the main requirement for the proposed assignment is that the server is programmed in such way that it can provide a service to the client in the form of a multi-person chatroom.

For this assignment, we have developed a 1 to many chat application, thus implying that many user's may connect to the server to begin their communication stream with other users. Our application will make specific use of TCP sockets, thus implying that the sockets which are defined as an endpoint for two programs connected to a particular network are bound to both the IP address and the port number for the respective clients.

2. Features

Server Class

The server class will mainly create and distribute threads specifically related to a particular client, to create the functionality which will allow the client to both read and reply to messages. The server will be fully concurrent thus implying that it can listen to and respond to multiple requests from multiple clients.

Methods:

Main():

The main class will initially create a new instance of the socket and restrict the size of the number of ports to 12038. Our instantiated socket will be set to null to ensure that the socket's "memory" is cleared out before a new client connects to the server.

socket.accept()

Within the while loop the server will make a call to the accept method which simply waits for the client to connect to the server. The server will make a call to a new server which has the sole purpose of handling the client's requests; It will also accept clients based on the queue of pending connections given to the server on a first come first serve basis. This server is supplied with the clients IP address and port numbers. Should the queue have no pending connections, the server will block the client making a call to the server.

The while loop will also serve to get input and output information on the behalf of the sever, furthermore a new instance of a client will be created. The new clients will be formatted into various threads to allow the clients to connect to the server.

ReceiveInput() & PrintMessage():

The method ReceiveInput() of type DataInputStream will receive the input from the client through the use of the call to the DataInputStream method and the method PrintMessage() of type DataOutputStream will create a Data Output Stream from the clients input.

CloseSocket ():

The method simply closes the sockets.

Sub-Class ClientControl

Client control has a constructor which takes in the data input, data output, the socket and the clients name. The class itself implements runnable, which will invoke the threads created by the main method in the Server class.

Methods:

Run():

The method contains a while loop which will simply check whether the client has requested to leave the chat by typing out the message “exit”. If the client has not requested to exit the chat, the embedded for each loop will check what message each client has sent and will distribute the messages accordingly ensuring that the sender does not receive the message they have sent and every other member of the chat receives the broadcasted message.

Client Class:

Constructor:

The client constructor will take in one parameter which is the client’s name. This will ensure that the correct client establishes a TCP/IP connection between the client and the server through the use of the IP address and the port number provided.

Methods:

ClientRun():

The ClientRun() method will ask for the IP address and the Host Name of the server. It will then create a new instance of the socket with the newly found IP address and the port number. Furthermore, two threads can be found within the ClientRun() method. The first thread will read the input provided by the client and should the client’s input be an “exit” then the system will notify all the other users by displaying a message that the particular client has “left the chat”. The second thread will, write the user’s output to the GUI for all the other chatroom members to view the message sent by the respective client.

ReceiveInput() & PrintMessage():

The method ReceiveInput() of type DataInputStream will receive the input from the client through the use of the call to the DataInputStream method and the method PrintMessage() of type DataOutputStream will create a Data Output Stream from the clients input.

CloseSocket ():

The method simply closes the sockets.

Login Class:

The login GUI, registers clients for the chatroom.

Methods:

SignUp()

The SignUp() method will save the client's login details, into a text file and will store their data in the format of a username followed by the client's password on a different line. The SingUp() method will use the FindUser() method to check if the client who is requesting to sign-up has already registered. This restriction will ensure that there aren't any duplicate registrations.

FindUser()

FindUser() is a method that will check if the user is already registered. Should the user be already registered, they will be permitted to enter the chatroom once they select the login button.

setCaptcha()

The setCaptcha() method generates a random number which will be placed in a TextField. The user will be required to enter the displayed number to validate that they are not a robot.

checkCaptcha()

The checkCaptcha() method will check if the user has entered the correct captcha details as depicted in the Text Field once the user selected the register or login button. Should the user not enter the correct captcha details they will be prompted to re-try by the registration system.

3. Constraints:

- a. Our inclusion of the captcha system, ensures that only humans have access to the chatroom and not robots.
- b. The login system will ensure that only registered users are able to converse in the chatroom, thus ensuring that there is transparency and accountability between the chatroom users.
- c. The registration system ensures that chatroom users have their private data such as chats protected through the use of setting a username and password to ban any forbidden/ un-authorized entry.

4. Protocol

The following protocol was adopted for the Chatroom Application:

Transmission Mode

- The system will follow a multi-cast format, due to unique conversations occurring between pairs of parties. However, each message sent by an individual user is sent as multiple packets of data to multiple receivers.
- The system we have chosen will unicast messages to a server and the server will multi-cast the specific message to the receivers. We are however, aware of the problem which user's may face when using this methodology is that messages may be received in different orders depending on the sender and receivers network connection.

Messages in Application

- The chatroom application will adopt a text-based messaging method due to formats such as Binary, providing communication barriers for those who do not easily understand binary format. The usage of text-based messaging also ensures that data is not lost in communication as it is designed to be read and understood by humans.
- All error messages will be presented by pop-up message boxes which will inform user's if they have submitted the incorrect information and can subsequently not proceed to the next action.

Types and Structure of Messages

- Command Messages: The system will provide command messages when the user initially logs onto the chatroom or registers, a message such as: “Successfully logged on”/ “Successfully Registered” will be presented. Another command messages will be presented when a user successfully terminates or exits the chatroom, a message such as: “Successfully logged off”/ “<name> exited the chat” will be depicted in the chatroom window for all users to see.
- Data Transfer Messages: These messages are simply presented as the queued and fragmented messages a client will send to other users of the chat. For example, each message that a user logged into the chatroom sends is considered to be a data transfer message, as data is being transferred from one individual (sender) to another (receiver).
- Control Messages: A sender will be informed when and if their message was successfully sent, thus ensuring that the chatroom provides a good protocol design framework similar to other successful chatroom applications.
- The chatroom application will have the following message structure: a header which will inform the receiver of the message who has sent the message through the format of <name>, a body which contains a queue of the messages sent by a member of the chat. The messages will be queued in the order in which the sender sent the messages.
- Communication Rules: The communication rules and the process in which the chatroom system operates is represented in the sequence diagram below. The diagram clearly depicts the operations of the system from its initiation when a user/client enters their username and password to the termination of the processes when the user requests to exit the chatroom.

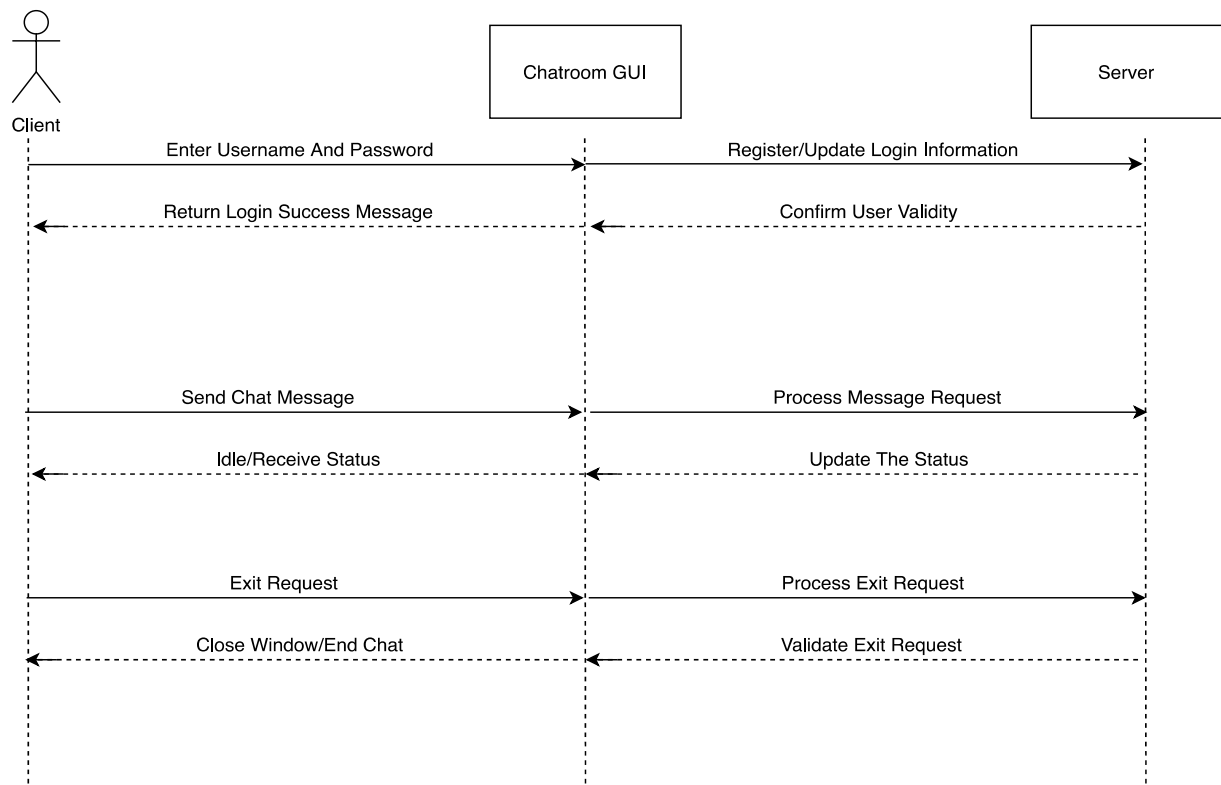
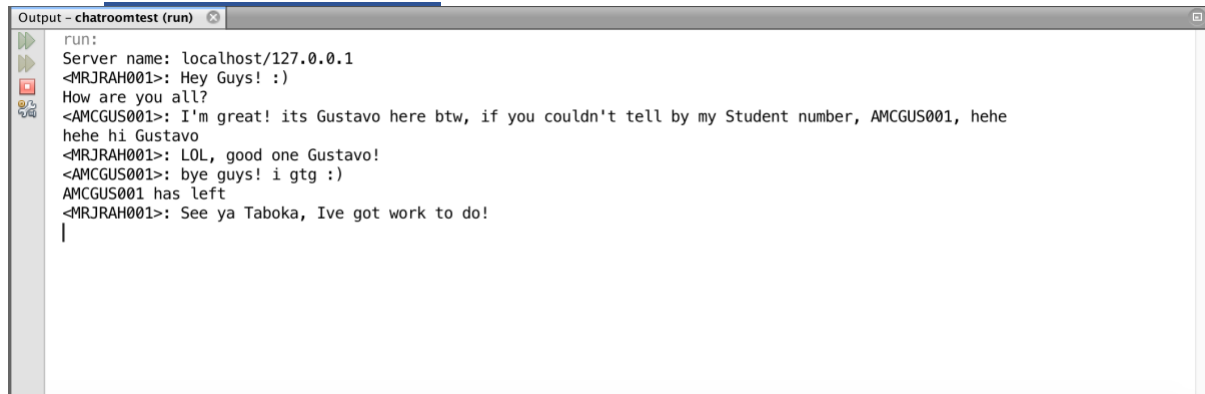


Figure 1- Sequence Diagram

5. Chatroom Screenshots

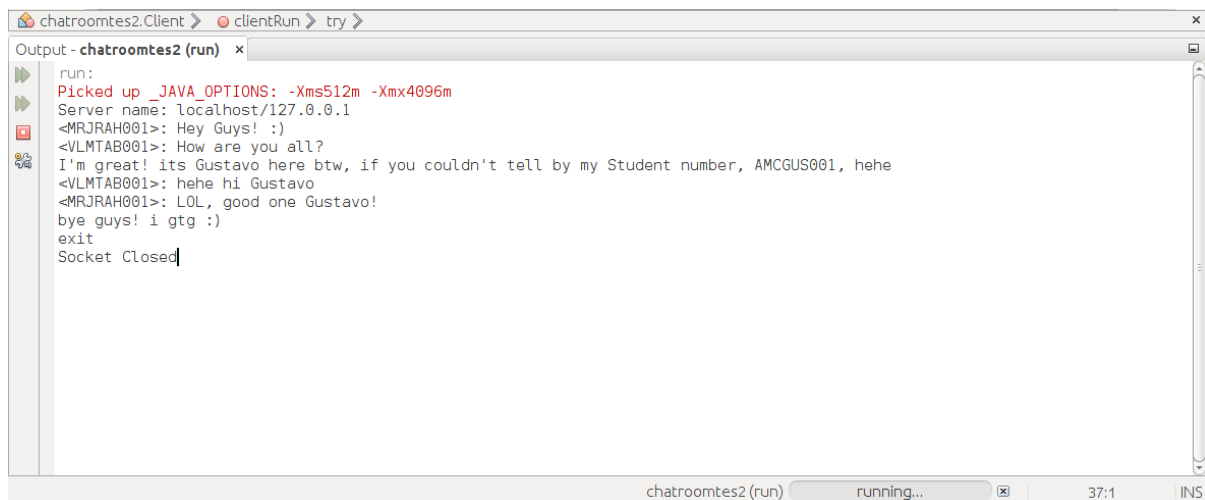


```
run:
Server name: localhost/127.0.0.1
<MRJRAH001>: Hey Guys! :)
How are you all?
<AMCGUS001>: I'm great! its Gustavo here btw, if you couldn't tell by my Student number, AMCGUS001, hehe
hehe hi Gustavo
<MRJRAH001>: LOL, good one Gustavo!
<AMCGUS001>: bye guys! i gtg :)
AMCGUS001 has left
<MRJRAH001>: See ya Taboka, Ive got work to do!
```

Figure 2- VLMTAB001 Screenshot

```
run:
Server name: localhost/127.0.0.1
Hey Guys! :)
<VLMTAB001>: How are you all?
<AMCGUS001>: I'm great! its Gustavo here btw, if you couldn't tell by my Student number, AMCGUS001, hehe
<VLMTAB001>: hehe hi Gustavo
LOL, good one Gustavo!
<AMCGUS001>: bye guys! i gtg :)
AMCGUS001 has left
```

Figure 3- MRJRAH001



```
run:
Picked up _JAVA_OPTIONS: -Xms512m -Xmx4096m
Server name: localhost/127.0.0.1
<MRJRAH001>: Hey Guys! :)
<VLMTAB001>: How are you all?
I'm great! its Gustavo here btw, if you couldn't tell by my Student number, AMCGUS001, hehe
<VLMTAB001>: hehe hi Gustavo
<MRJRAH001>: LOL, good one Gustavo!
bye guys! i gtg :)
exit
Socket Closed
```

Figure 4-AMCGUS001