

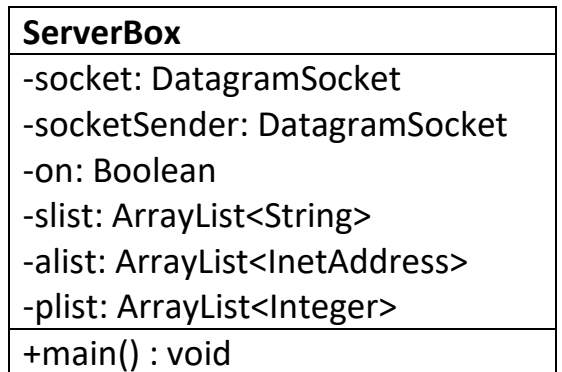
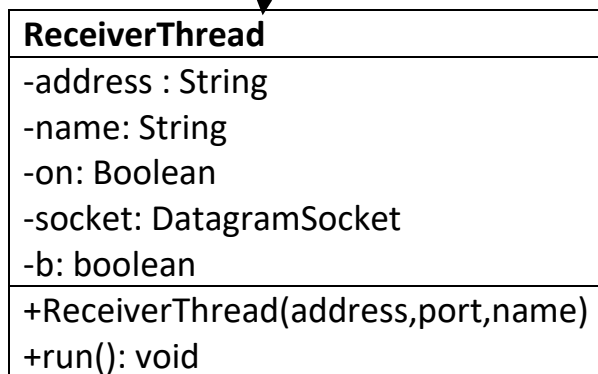
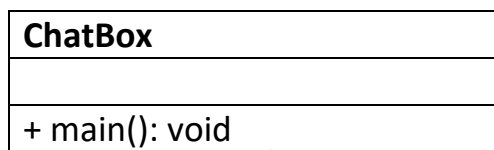
12 April 2021

CSC3002F Networks Assignment 1: DatagramSocket programming in Java

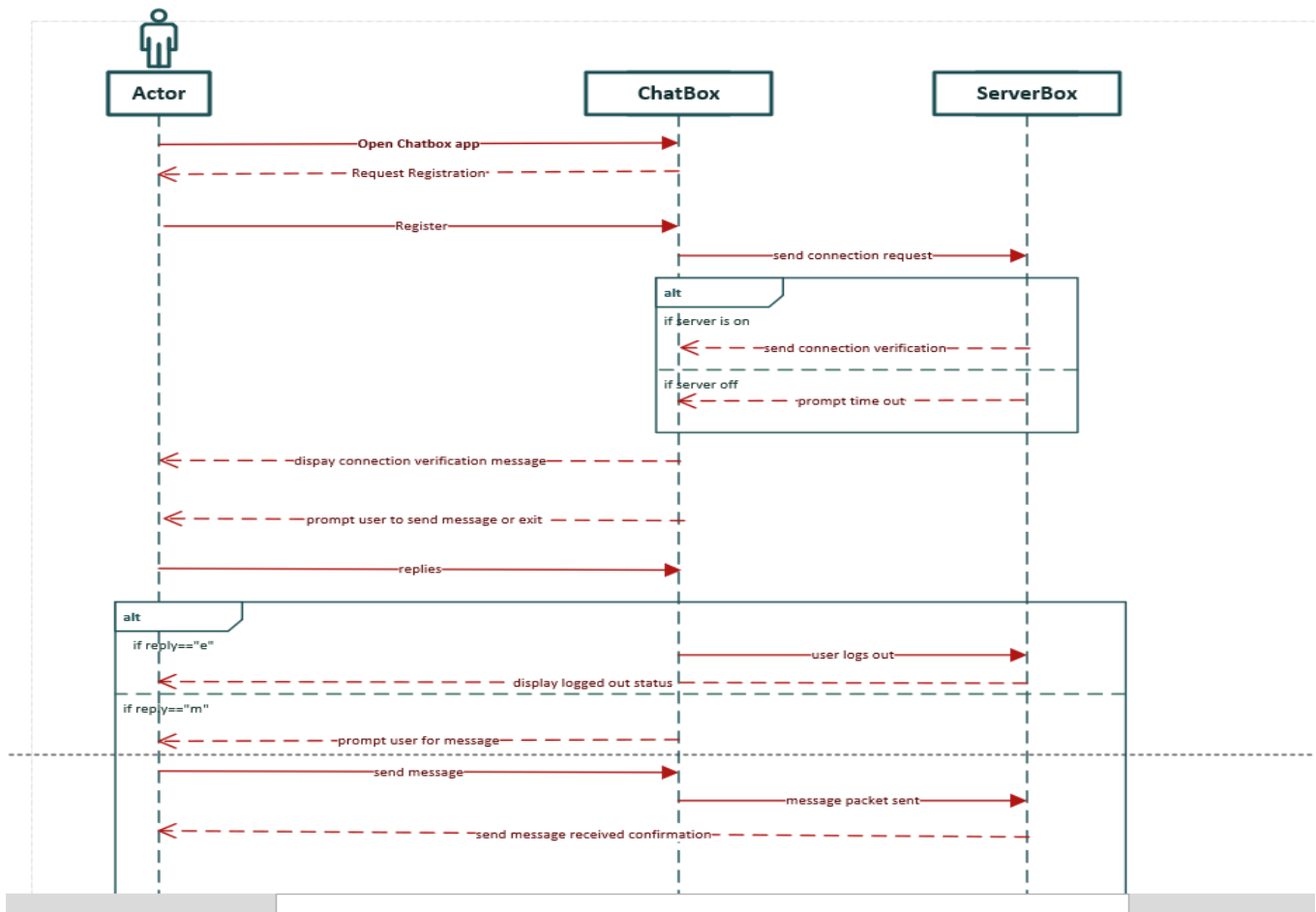
Name and Surname	Student Number
Segomotso Petlele	PTLLES002
Kundai Chatambudza	CHTKUN003
Brighton Tandabandu	TNDBRIO01

Networks Assignment 1: UDP Socket programming in Java

UML Class Diagrams



Sequence Diagram



Introduction

This assignment is about programming a chat application called ChatBox in which two or more users can send messages to each other in real time. The architecture used for the ChatBox is a client-server architecture which means all communication between users happen through the server. The server created for this assignment is called the ServerBox. A User Datagram Protocol (UDP) is used in the transport layer. UDP is an unreliable connectionless protocol. This means that no connection is established with server prior to making communication with server and that no confirmation of server receiving Datagram Packets and no mechanism of knowing if Datagram Packets were sent successfully.

Datagram Packets are the capsules that contain the data to be sent over the network together with containing receiver, sender's IP address and Port number, as well as containing the sender's IP address and Port number. Datagram Packets are sent through Datagram Sockets which are used to create a network connection between different computers.

Design and Functionality of Classes

The protocol design is such that before users can start communication with each other, a connection request to the server has to be established first. After successful connection, users connected to server may exchange messages freely while online.

❖ ServerBox (Server)

Design:

The ServerBox class is the class that runs and hosts the server. The ServerBox must first be run on a remote computer dedicated only to running the server alone. No more than one user should connect to the server on the same computer running the server. This will cause the clients on the same computer as the server be it using a virtual environment to simulate other operating systems or using multiple windows to simulate different clients will cause them to receive a “message received” message to each other because the server uses the machine’s IP address to notify the user that their message(packet) was received by the server. See screenshots on figure1 below of a Linux (on virtual box) and Windows client sending messages back and forth on the same computer in which the ServerBox was run on the Windows operating system.

The server has two sockets, one to constantly receive messages and request while the other is responsible for sending messages to every client connected to the ServerBox.

An unlimited number of people may connect to the server remotely using their personal computers or laptops for as long as they are all connected on the same network.

The server can be monitored by the person hosting it. When online, the server will communicate this by printing to screen a message, see figure2. Server will also display a message confirming every time if it receives a packet through it.

Functionality:

Command messages and Control messages: the ServerBox first needs a user to connect to it before user can start sending and receiving messages. The ServerBox will than notify the connected user that they are connected to the ServerBox by sending back a confirmation message, see figure 3 of the expected confirmation message. After this message is received than all users connected on the server can exchange messages as when one user sends a message, all users connected on the ServerBox will receive the message. The server will always notify each client that their message is sent by sending back a confirmation message, see figure three. If

user does not get this confirmation than user should try resending the message as the server would have most likely not received the message.

The server is also responsible for notifying all clients when a user connects and disconnects to and from it. See figure three below.

❖ **ReceiverThread (client helper)**

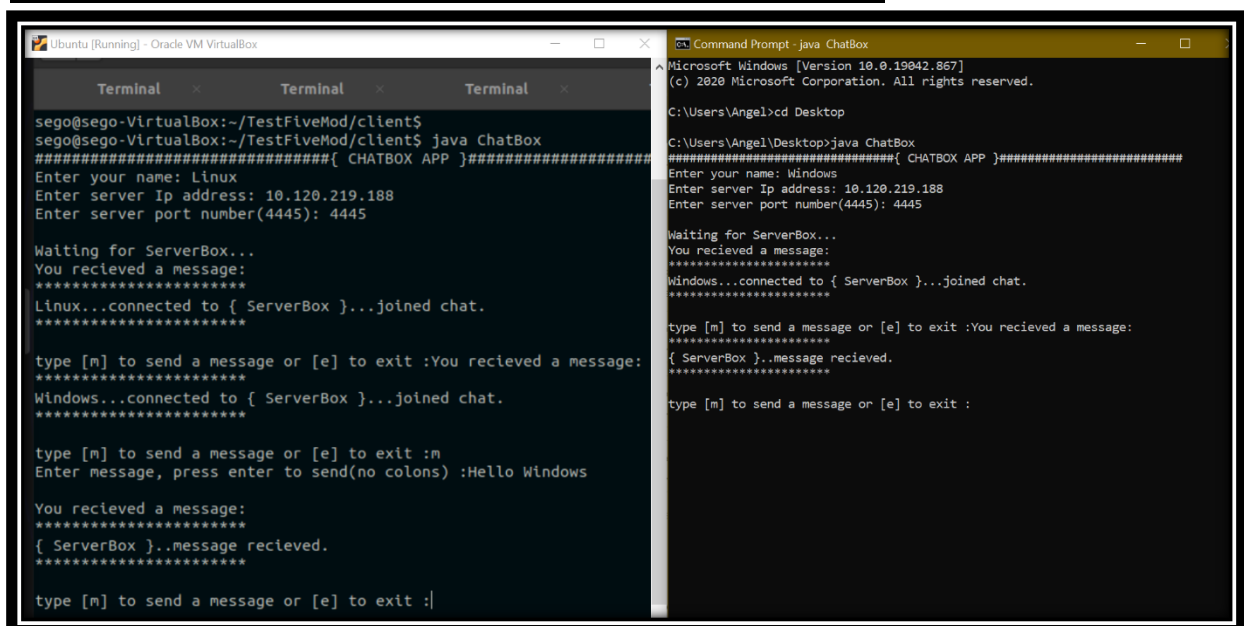
This class is responsible for first trying to establish a connection to the ServerBox and extends thread. If three seconds pass than user will be notified to quit the program and to try to re-connect to the Server. Otherwise, user will receive a confirmation from server that user is connected. After this registration, the ReceiverThread functions only to receive messages meaning that the user can send many messages and receive many messages continuously and in parallel without delay and conflict.

❖ **ChatBox (Client)**

The ChatBox is the application which all users use to connect to the server. It calls the ReceiverThread and passes down parameters needed for registration purposes such as name, server port number and server IP address. This class is solely responsible for users to send messages constantly to the server once a connection is established with the server.

Diagrams(illustrations):

Figure 1: Two clients connected on host machine on one computer



```
Ubuntu [Running] - Oracle VM VirtualBox
Terminal
sego@sego-VirtualBox:~/TestFiveMod/client$ java ChatBox
#####{ CHATBOX APP }#####
Enter your name: Linux
Enter server Ip address: 10.120.219.188
Enter server port number(4445): 4445

Waiting for ServerBox...
You recieved a message:
*****
Linux...connected to { ServerBox }...joined chat.
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
Windows...connected to { ServerBox }...joined chat.
*****

type [m] to send a message or [e] to exit :m
Enter Message, press enter to send(no colons) :Hello Windows

You recieved a message:
*****
{ ServerBox }..message recieved.
*****

type [m] to send a message or [e] to exit :

Command Prompt - java ChatBox
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

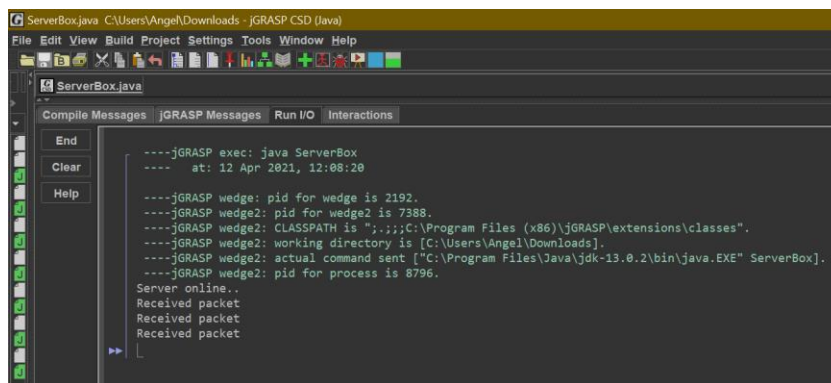
C:\Users\Angel\Desktop>cd Desktop
C:\Users\Angel\Desktop>java ChatBox
#####{ CHATBOX APP }#####
Enter your name: Windows
Enter server Ip address: 10.120.219.188
Enter server port number(4445): 4445

Waiting for ServerBox...
You recieved a message:
*****
Windows...connected to { ServerBox }...joined chat.
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
{ ServerBox }..message recieved.
*****

type [m] to send a message or [e] to exit :
```

Figure2: ServerBox monitoring



```
ServerBox.java C:\Users\Angel\Downloads - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help

ServerBox.java
Compile Messages jGRASP Messages Run I/O Interactions

End
Clear
Help

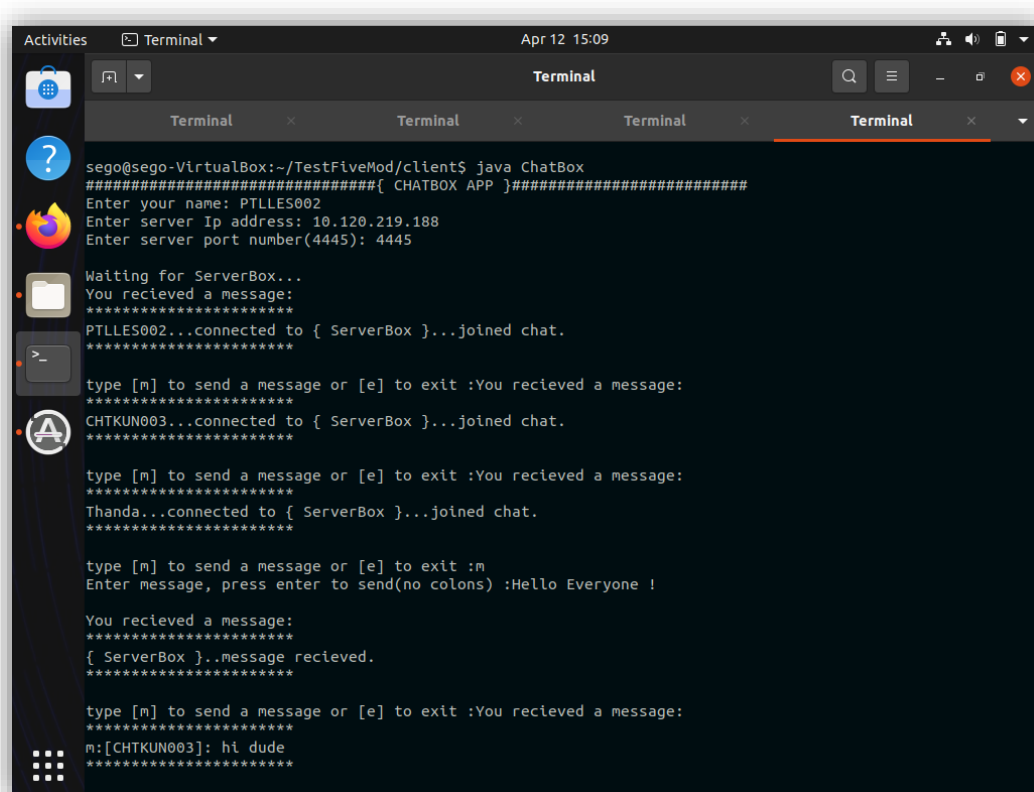
----jGRASP exec: java ServerBox
---- at: 12 Apr 2021, 12:08:20

----jGRASP wedge: pid for wedge is 2192.
----jGRASP wedge2: pid for wedge2 is 7388.
----jGRASP wedge2: CLASSPATH is ".;.;C:\Program Files (x86)\jGRASP\extensions\classes".
----jGRASP wedge2: working directory is [C:\Users\Angel\Downloads].
----jGRASP wedge2: actual command sent ["C:\Program Files\Java\jdk-13.0.2\bin\java.EXE" ServerBox].
----jGRASP wedge2: pid for process is 8796.

Server online..
Received packet
Received packet
Received packet
```

ServerBox will display “server online when it becomes online and wait to receive packets from users. When a packet is received by the ServerBox, the Server will display “Received packet”.

Figure3: Multiple clients on different computers chatting using ChatBox



```
Activities Terminal Apr 12 15:09
Terminal
Terminal x Terminal x Terminal x Terminal x

sego@sego-VirtualBox:~/TestFiveMod/client$ java ChatBox
#####{ CHATBOX APP }#####
Enter your name: PTLLE5002
Enter server Ip address: 10.120.219.188
Enter server port number(4445): 4445

Waiting for ServerBox...
You recieved a message:
*****
PTLLE5002...connected to { ServerBox }...joined chat.
*****

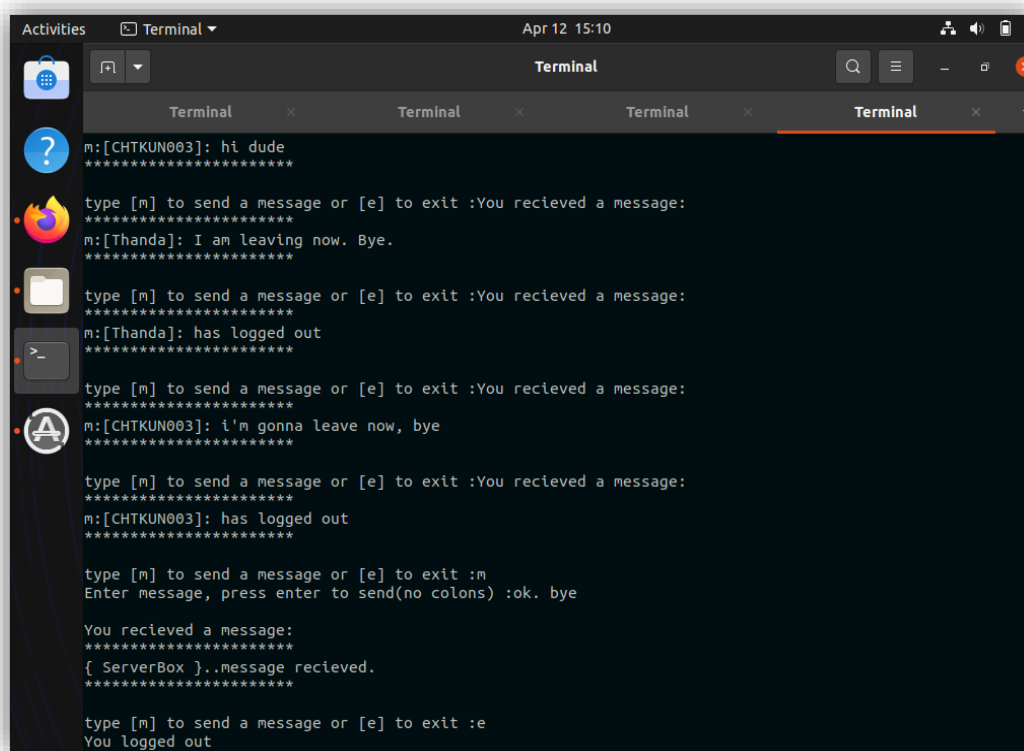
type [m] to send a message or [e] to exit :You recieved a message:
*****
CHTKUN003...connected to { ServerBox }...joined chat.
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
Thanda...connected to { ServerBox }...joined chat.
*****

type [m] to send a message or [e] to exit :m
Enter message, press enter to send(no colons):Hello Everyone !

You recieved a message:
*****
{ ServerBox }..message recieved.
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
m:[CHTKUN003]: hi dude
*****
```



A terminal window titled "Terminal" with a search bar and window controls. It displays a chat application interface with a dark background and light text. The chat log shows messages from "m:[CHTKUN003]" and "m:[Thanda]". The interface includes prompts like "type [m] to send a message or [e] to exit :You recieved a message:" and "Enter message, press enter to send(no colons) :ok. bye". The chat log shows messages from "m:[CHTKUN003]" and "m:[Thanda]". The interface includes prompts like "type [m] to send a message or [e] to exit :You recieved a message:" and "Enter message, press enter to send(no colons) :ok. bye". The chat log shows messages from "m:[CHTKUN003]" and "m:[Thanda]". The interface includes prompts like "type [m] to send a message or [e] to exit :You recieved a message:" and "Enter message, press enter to send(no colons) :ok. bye".

```
m:[CHTKUN003]: hi dude
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
m:[Thanda]: I am leaving now. Bye.
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
m:[Thanda]: has logged out
*****

type [m] to send a message or [e] to exit :You recieved a message:
*****
m:[CHTKUN003]: i'm gonna leave now, bye
*****

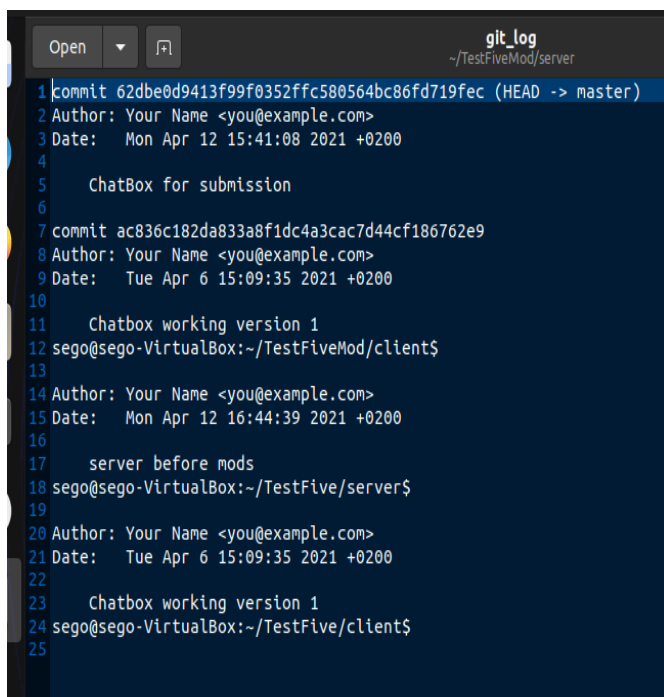
type [m] to send a message or [e] to exit :You recieved a message:
*****
m:[CHTKUN003]: has logged out
*****

type [m] to send a message or [e] to exit :m
Enter message, press enter to send(no colons) :ok. bye

You recieved a message:
*****
{ ServerBox }..message recieved.
*****

type [m] to send a message or [e] to exit :e
You logged out
```

Figure 4: Git log



A terminal window titled "git_log" showing the output of the "git log" command. The output lists commit hashes, authors, dates, and commit messages. The commits are numbered 1 through 25. The messages include "ChatBox for submission", "Chatbox working version 1", and "server before mods".

```
git_log
~/TestFiveMod/server

1 commit 62dbe0d9413f99f0352ffc580564bc86fd719fec (HEAD -> master)
2 Author: Your Name <you@example.com>
3 Date: Mon Apr 12 15:41:08 2021 +0200
4
5 ChatBox for submission
6
7 commit ac836c182da833a8f1dc4a3cac7d44cf186762e9
8 Author: Your Name <you@example.com>
9 Date: Tue Apr 6 15:09:35 2021 +0200
10
11 Chatbox working version 1
12 sego@sego-VirtualBox:~/TestFiveMod/client$
13
14 Author: Your Name <you@example.com>
15 Date: Mon Apr 12 16:44:39 2021 +0200
16
17 server before mods
18 sego@sego-VirtualBox:~/TestFive/server$
19
20 Author: Your Name <you@example.com>
21 Date: Tue Apr 6 15:09:35 2021 +0200
22
23 Chatbox working version 1
24 sego@sego-VirtualBox:~/TestFive/client$
25
```

List of features:

- Registration: when first using ChatBox, you register using your name
- Online and offline feature: all users are notified when a user becomes online and when a user goes offline.
- Message confirmation: all communication through the app is met with confirmation messages for users to know exactly what's happening at all times during their user experience with the app.
- Real time chatting: all communication between participants happens in real time.

Closing Remarks:

The code attached has been tested and works well when being run. It is highly advised that when testing, only one user can be a user on the same computer running the server. The rest of the users should connect remotely on independent computers.

Lastly, the code runs well when server is run on a windows machine than a Linux machine. Overall, it is preferred that the code is entirely run on windows machines however, we have experienced inconsistencies with connections and messages being displayed when some clients use Linux machines.