**CSC3002F Assignment 1 Report**

**Group NET 32**

**Members: Matthew Schramm, Connor Thompson, Josh Birkholtz**

Design

The design of the chat application is of a repeating menu displaying the available options to the user of the app, where they can enter the according number to what feature they would like to use. The application’s features include:

* Choosing of user status – this was included to give users the freedom to choose their privacy and to not be contacted during private messaging sessions. Also allows for users to stay on the application and leave their computer without being bombarded with requests.
* Viewing of online peers – this was included so the user can know which other peers are online and able to be chatted with.
* Private messaging of peers – this was included to allow users to privately connect to each other through the rendezvous server and privately chat.
* Sending of files/media to other peers – this was included so that images and text files can be easily shared between two connected peers.
* Encryption of passwords on the server – included for safety reasons

Functionality

1. The user enters the chosen IP and Port of the chat server they wish to connect to.
2. A TCP connection is made between the user and the server officially making a connection between the two. The clients IP and Port is sent to the server which stores it.
3. The client then enters their Username, the server then checks if the client has logged in before by seeing if the username is stored on the server. If it is a new username then the client is asked for a password to associate with their username successfully registering them in the server. If it is an existing user, they are asked for their password which is continuously asked until the correct password is given.
4. Once the user has successfully logged in or registered, they are asked for their status. They can either be Public (they are seen by other clients) or Private (hidden from other clients)
5. After this they are finally given access to the application in the form of a repeating menu with 4 options being to Exit, Set Status, View Online Peers and to Message A Peer.
6. If the user enters 0 as their input, they will be disconnected from the server
7. If the user enters 1 as their input, they will be able to change their status to either Available, Connected or Away. Available meaning they can be messaged, Connected meaning they are currently in a call and Away to let everyone know they are away from the application.
8. If the user enters 2 as their input, the peer list along with their status will be printed out for the user to read.
9. If the user enters 3 as their input, the peer will be able to type in the username of the peer they would like to connect to. They are only connected once the other peer connects to them using the same method. Once a request is sent back, a peer-to-peer connection is made using UDP. While connected to another peer they are able to send files to each other by typing “send” as their message to the other peer. After typing send they are asked for the filename which if a valid filename is given, is directly sent to the other peer. If one peer chooses to exit the peer-to-peer chat they can type “exit” at any point making the UDP connection close.
10. The menu will be outputted, and the users input asked for after every feature chosen by the user is completed.

**Client Implementation’s**

SCHMAT041

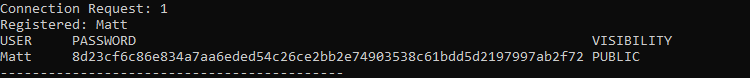
The way I have implemented my client is that the user inputs their chosen server’s IP and port from which an ipv4 socket object is created connecting it to the server. After this the user logs into the server and gives their visibility. After the client has either logged into the server or registered successfully. They are presented with the menu which is the base of the application. They can either exit, change their status, view the online peers or message another peer. All the data is received from the server as it stores all the information on the peer and is used as a rendezvous server to relay information to the clients and is not used to connect the clients. Connecting to another peer uses a UDP socket object and is connected using UDP allowing files and media to be shared between them along with normal text.

**Protocol Specification**

**Screenshots**

Server







Client:

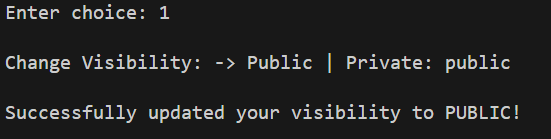
A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated



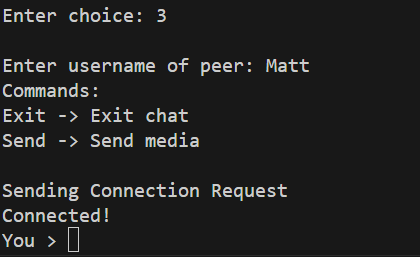


A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated



A black background with white text

Description automatically generated