Chat: Client & Server (asmt3)

In this report I will go through my assignment and explain my reasoning and logic of the server and client code. I will start with the server code as that is what gets run first.

```
struct client { // the information for each
    int index; // index of client on list
    int sockID; //socket descriptor
    std::string username;
};

struct client ClientArray[1024]; // array to
pthread_t thread[1024]; // allow for 1024 cl
int clientCount = 0; // how many clients we
std::string history[12];    // Holds 12 line
int histCount = 0; // holds how many message
int x; // index
```

The code in here contains all the global information that the server will hold, such as the index of clients which is held in “ClientArray”, the chat history and finally the thread ids of each client which correlate to the client. Ex, Client[0] would correlate to thread[0]. I initially had the “history” array as a 2D char array ([12][512]), this made it difficult to store strings due to conversions and I wasn’t able to figure it out so I changed the type. As you can see I’ve also used a struct to store some information of ever client such as username and “sockID”, which would later be used to cycle through clients to send them a message that a user has sent.

When running the server code you will notice a compiler warning “Warning: NULL used in arithmetic”, this error comes from me checking if a socket ID is NULL. A fix for this is to cycle through each client in “ClientArray” and set each index to 0, though I would have to do this every time a client is made which is very time consuming. An alternative may also be use a list, though to-do this it would require me to refactor the entire and with the interest of time I chose to ignore the warning(it also doesn’t cause any bugs).

Both C++ files contain one method; on the client side the function is receive() which is what “receives” messages from the server to output to the client. This function is used when the server wants to send a message to the client and it will keep listening using the while loop. This function is key in sending the history to all clients and send the current message to all other clients. As for the server side, there is a handle Networking() method, which stores the history, using the array data structure as described above. The function is contains two writes that write to the receive function which allow client to see messages from others.