CS 513

Computer Networks

Class Project

Client Server Chat

Submitted By - Paritosh Goel

Student ID - 353157299

Date of submission - 25th November 2019

Statement of ownership - I hereby confirm that all the work included is mine unless marked to the contrary

Abstract

This report outlines the design and development of a program which is a Client Server Chat application. The program is written in Java which is implemented and tested on a MAC platform.

There are two different sections of this software - the client and the server. It is a multi-client solution, where multiple clients are connected to a single server and these clients can talk to each other.

Each client is notified of the active connections and disconnections of other clients.

The program uses extensively Java socket library and multithreading concepts to handle the communication between client-server-client. Different threads are implemented on both client and server side which parallelly perform the required task such as reading to the socket and writing to the sockets. The client also includes a GUI which is implemented using java.swing package. The report includes the detailed design details and also the usability information. Also includes the test cases that were being performed on this software.

The server includes a centralized queue. All the messages from clients are pushed in this queue and there is a separate parallel thread which reads the queue and transmits the messages to the appropriate socket. In this way we get a two way communication. The client on the other hand keeps listening to the socket and take actions on the ui such as adding a chat message or adding/deleting a connected user to its list.

Contents Page

Project Description

This project involved design and implementation of a client server chat program. Different clients can talk to each other by connecting to a single server. Each client is aware of the other clients who are connected to the server at each time.

Each client maintains a connection to the server with the help of Java Socket library. Server assigns a dedicated socket to each connected client. The client is either approved or rejected based on the name it chooses. If that name is already chosen the client will be rejected and asked to pick a different name. Once the client is connected, it will be shared with a list of connected users. The client uses that information from the server to display the list of connected users to the end user who is using that client application.

Now when the client can see all the users connected, it can whisper to anyone in that list.

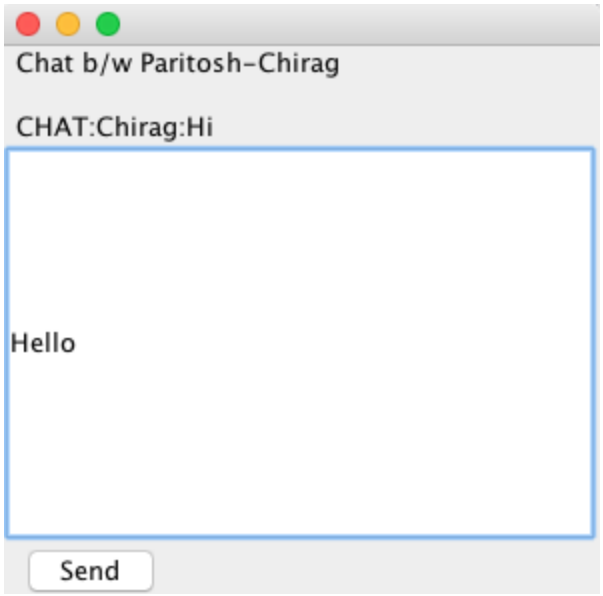
Below is the step wise detailed information with screenshots of the application.

Step 1 : The client mentions the server address and port number along with its Name.

Here client name- paritosh is trying to connect
At this point the server is already running and waiting for clients to connect.

Step 2 : After clicking the connect button, we get the below screen with list of users already connected. Here user - dummy and user Chirag are already connected with the server so it is displayed to the user.

Step 3 - At this point the client can wait for another user to whisper to it or can itself start a conversation with other client. Both the ways we get below frame seen on the client side



Here Paritosh received a Hi from chirag and paritosh is typing hello to be sent to Chirag, We have these frames separate for each connection to a different client.

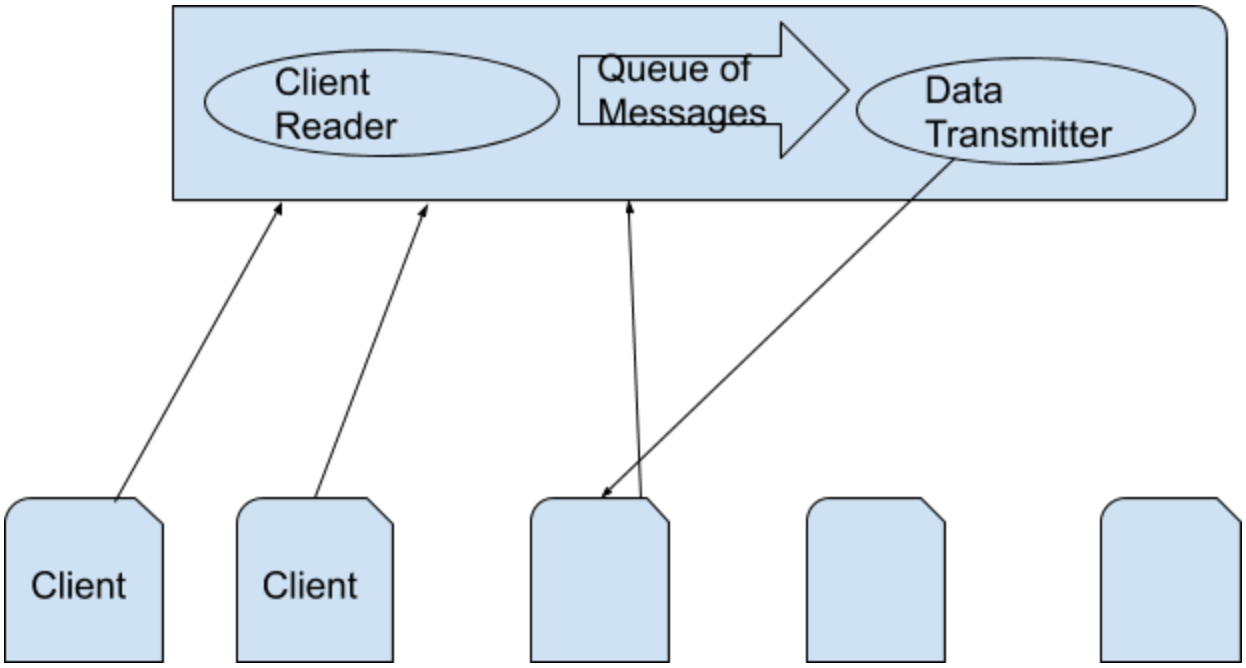
Step - 4

We will also notice the previous frame to keep getting updated with the list of users are being added or disconnected accordingly. Clients actively keeps on looking for server to send the updated list of users and updated its UI accordingly.

Below is the scenario where three clients are connectected and two clients are talking to each other-

Detailed Design

1. From Server Perspective -



Above is a detailed design of Server.

Once the server accepts a client, it will add it to the list of active users.

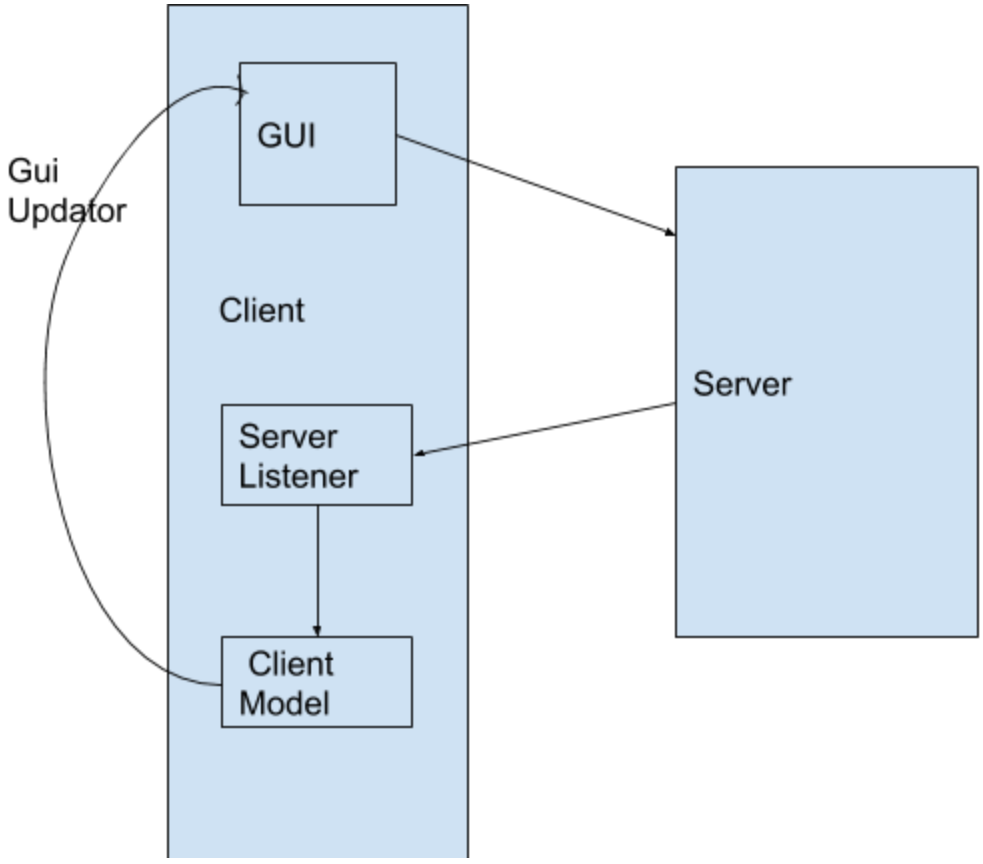
A thread named Client Reader will listen to the clients and push the messages to a central queue called Queue of messages.

A thread named Data Transmitter keeps listening the the queue of messaged and transmits these messages to the appropriate location.

All these threads work in parallel to achieve a two way communication between the clients.

The main thread on the other hand keeps listening to the incoming clients and assigns a new thread to each client which further looks into acceptance or rejection of a client. This leads new clients to not wait much for getting accepted.

From Client perspective



A client on the other hand has a background thread running called Server Listener, this thread as soon as receives a message from server, parses the message and hands it over to the GUIUpdator. The GuiUpdator takes the further responsibility to update the UI by using the client model.

Client Model is a Java Data class which holds up all the information to be used by Gui Updator.

When a client sends a message to the server, the GUI components issues a message to the output buffer of the socket.

For Client, there can be mainly three kinds of messages received from the server -

1. **Chat Message** - When a client sends a whisper to this client.
2. **Connect Message** - When a new client is added.
3. **Disconnect Message** - When a client is disconnected.

The server will tag each message from one of the above list, client will that ways get to know the action it needs to take on the UI

Testing and Evaluation

Overall Test Strategy - I deployed the server and multiple clients locally to test the functionality of the software. I initiated the client connections to the server, and monitored the logs. Below functionalities were tested -

1. Client should be rejected any connection if there is already the same name client
2. If the client is accepted, it should be sent the updated list of connected users and the client should display that list to the User Interface
3. If a client sends a message to another client, it should be logged to server and the receiving client should show that information to the user.
4. The server should log a new connect/disconnect information of the client and issue message to all other clients about it.

Overall Results - All the above functions are working. Sometimes in odd scenarios, there might be a crash.

Test Runs -

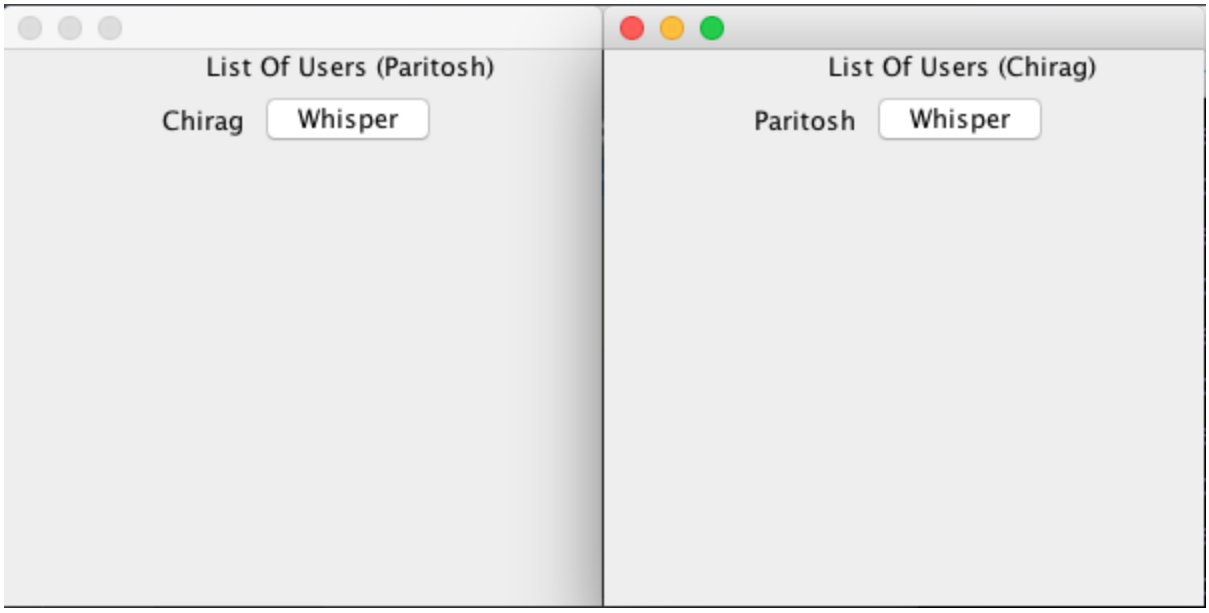
Below is one complete scenario tested along with screenshots attached.

1. Initially there are no clients connected to the server. And there is a client which enters the client name and serverip with port number.

After hitting the connect button, below screen comes -

At this time there is no other client connected so the list is empty

2. Now a second client comes up - (name is Chirag)
3. After it is connected we get to see the clients as below shown in the screenshot.



As we can see both the clients get to see the connected users.

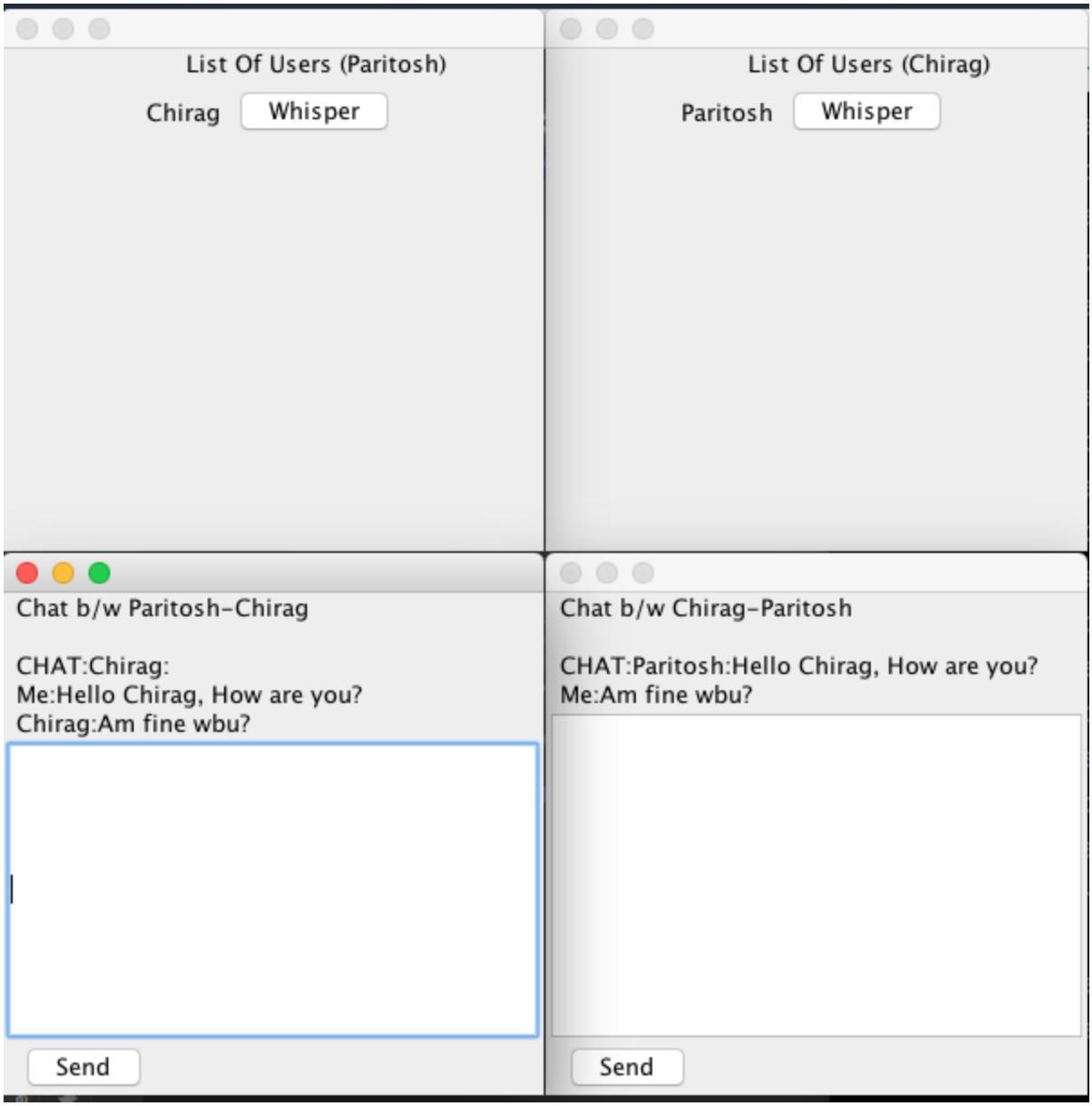
Below is the server logs till now -

NEW CLIENT CONNECTED:Paritosh

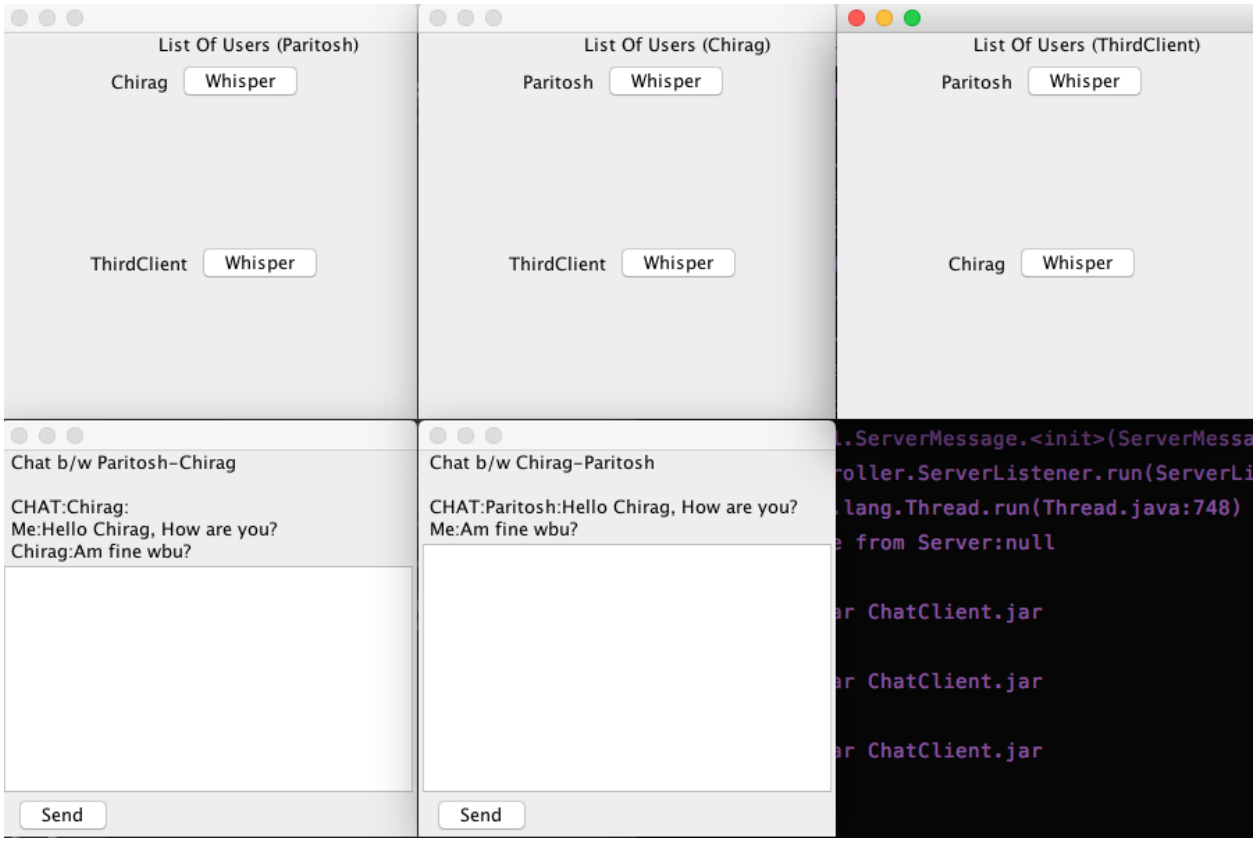
NEW CLIENT CONNECTED:Chirag

Written to client-Paritosh:CONNECT:Chirag:dummyvalue

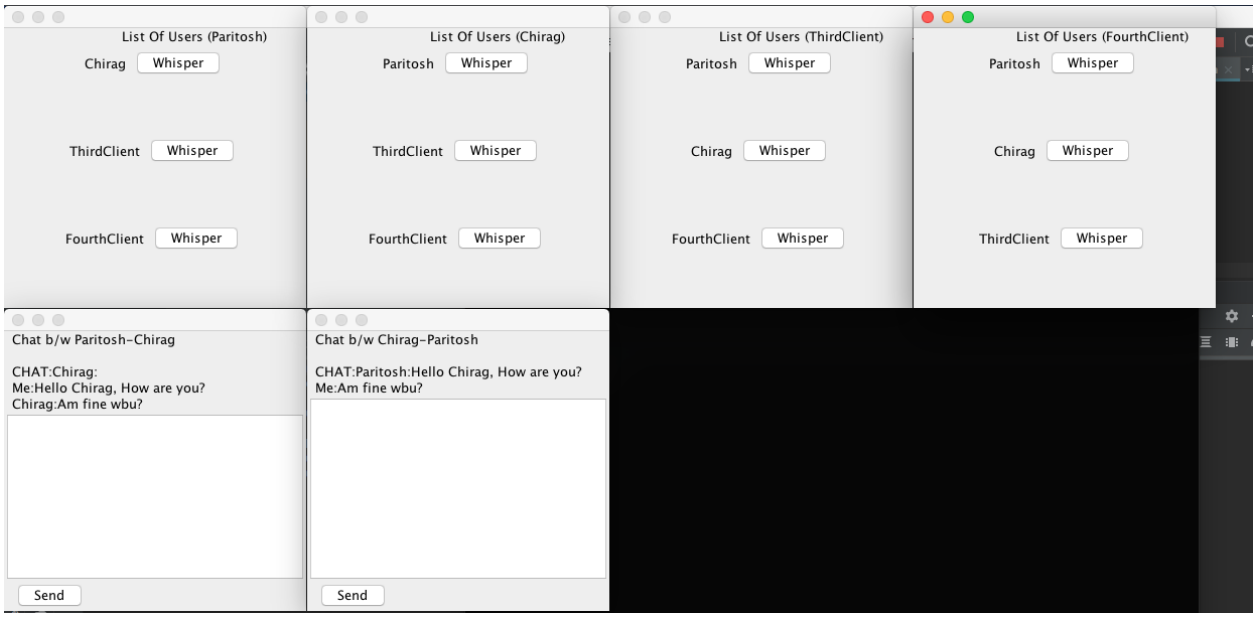
4. Now, Client - Paritosh initiated a chat with Chirag as seen below, each client gets a separate window for the chat



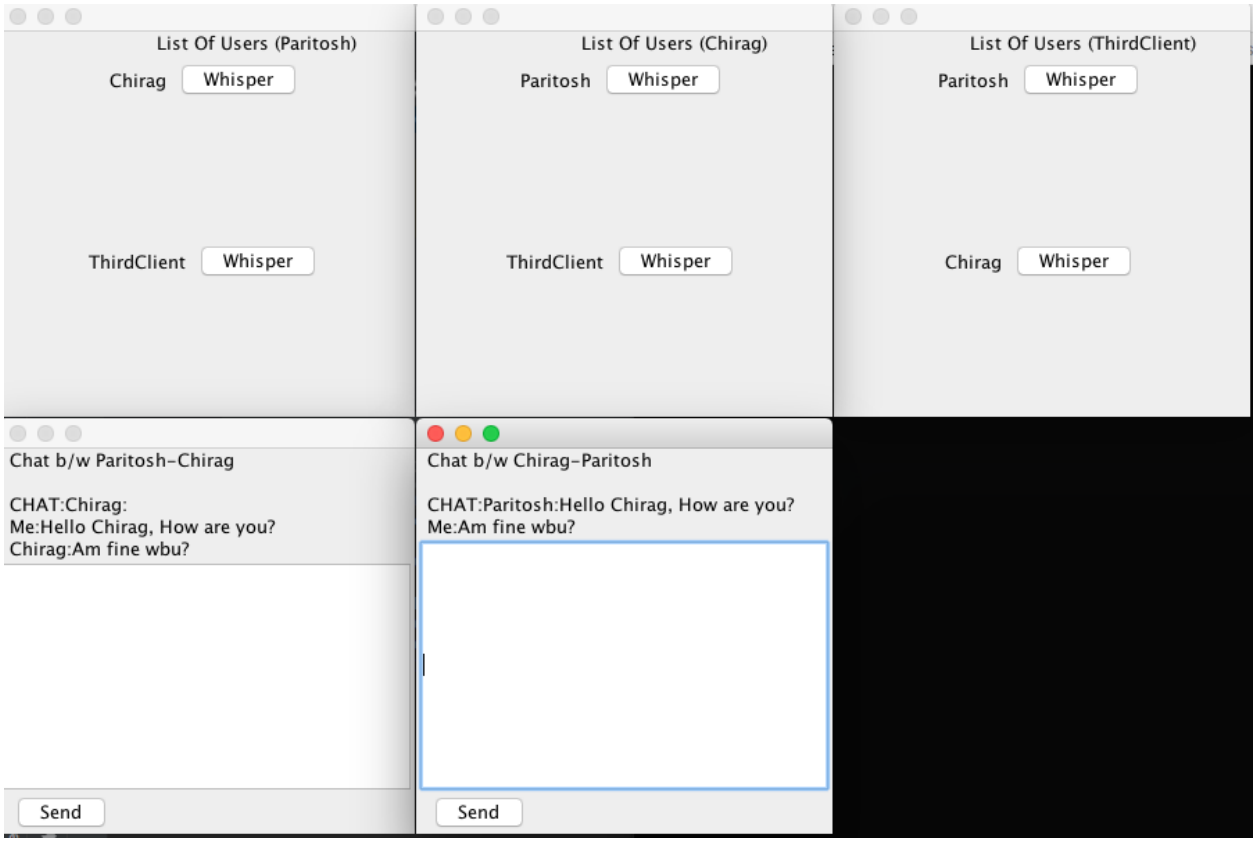
5. Now I initiate another client (name - Third client), which can be seen updated in both of the previous clients list -



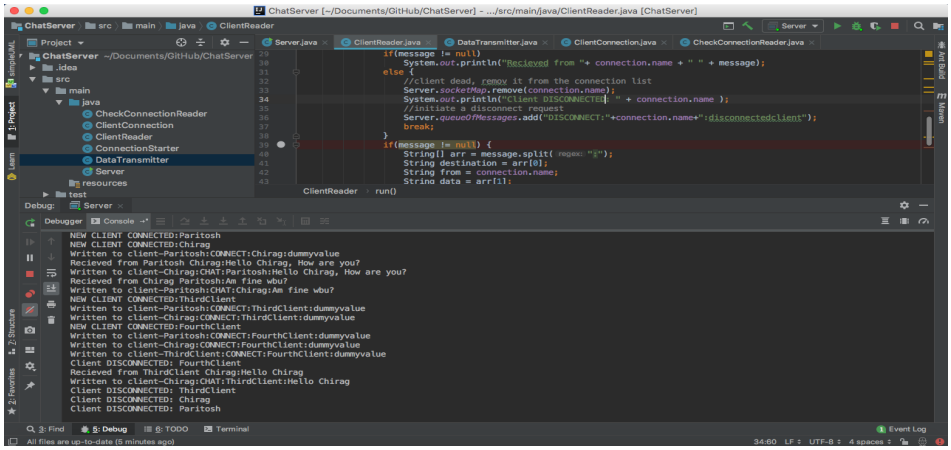
Now I add one more client to the server as shown below,



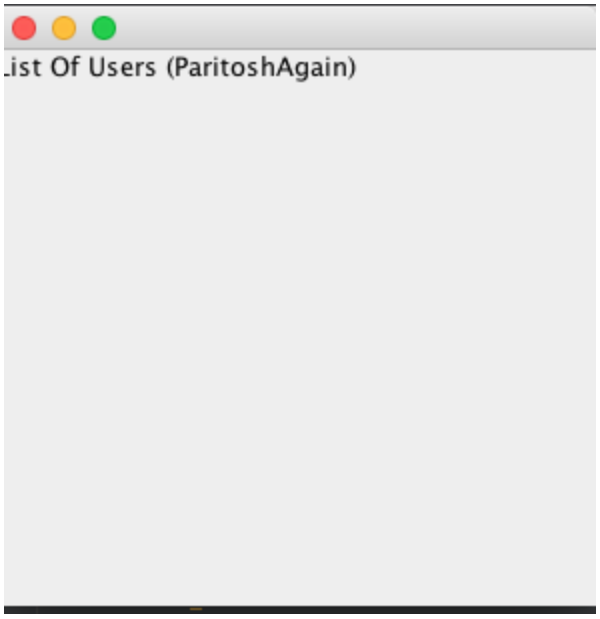
Now I test the DISCONNECT feature where i disconnect one of the clients and it is reflected to all other clients -



Now, I test the server stability by disconnecting all the clients, as we can see below, the server is healthy and still running -



Now, to confirm about the server stability I connect a clientname - “ ParitoshAgain”, which connects perfectly fine -



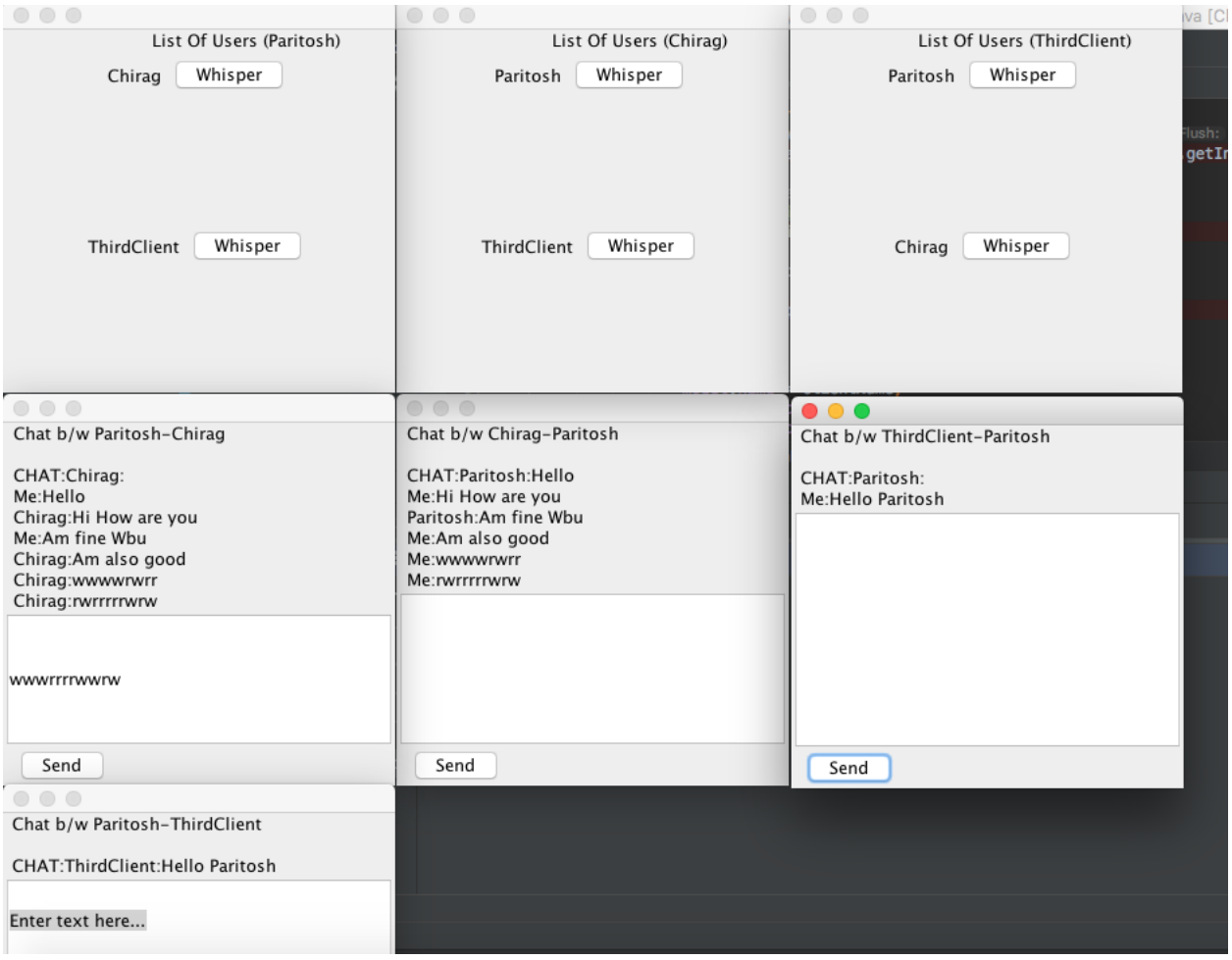
Server logs -
NEW CLIENT CONNECTED:Paritosh
NEW CLIENT CONNECTED:Chirag
Written to client-Paritosh:CONNECT:Chirag:dummyvalue
Recieved from Paritosh Chirag:Hello Chirag, How are you?
Written to client-Chirag:CHAT:Paritosh:Hello Chirag, How are you?
Recieved from Chirag Paritosh:Am fine wbu?
Written to client-Paritosh:CHAT:Chirag:Am fine wbu?
NEW CLIENT CONNECTED:ThirdClient
Written to client-Paritosh:CONNECT:ThirdClient:dummyvalue
Written to client-Chirag:CONNECT:ThirdClient:dummyvalue
NEW CLIENT CONNECTED:FourthClient
Written to client-Paritosh:CONNECT:FourthClient:dummyvalue
Written to client-Chirag:CONNECT:FourthClient:dummyvalue
Written to client-ThirdClient:CONNECT:FourthClient:dummyvalue
Client DISCONNECTED: FourthClient
Recieved from ThirdClient Chirag:Hello Chirag
Written to client-Chirag:CHAT:ThirdClient:Hello Chirag
Client DISCONNECTED: ThirdClient
Client DISCONNECTED: Chirag
Client DISCONNECTED: Paritosh

NEW CLIENT CONNECTED:ParitoshAgain
Client DISCONNECTED: ParitoshAgain

List features tested -

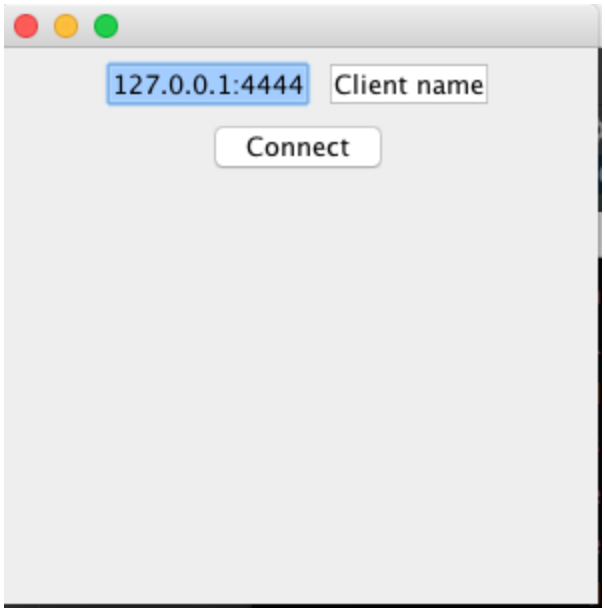
1. **Client GUI - Shown above**
2. **Sending / Receiving Messages - Working fine (shown above)**
3. **Sending / Receiving Messages (concurrency)**

Here is an example were paritosh is typing but concurrently receiving messages from chirag, also there is a separate connection between Paritosh and ThirdClient.



4. **List of currently connected users** - Working fine (shown above)
5. ***Updating user list after user connection / disconnection - Working fine (shown above)***

6. ***Client Stable after Server Termination -*** *Client is stable after server termination. On server termination, client reaches to it initial state as shown -*



7. ***Server Stable after Client Termination - Working fine (shown above)***
8. ***Whispering to a Non-Existing Clien****t - UI does not allow that*
9. ***Trying to Connect to Non-Existing Server -*** *The exception is shown in the logs and the client is stable*

java.net.ConnectException: Connection refused (Connection refused)
at java.net.PlainSocketImpl.socketConnect(Native Method)

10. **All Opened Sockets and Streams Closed on Client after Termination - ClientReader.Java**

```
else {
    //client dead, remov it from the connection list
    Server.socketMap.remove(connection.name);
    System.out.println("Client DISCONNECTED: " + connection.name );
    connection.in.close();
    connection.out.close();
```

11. **Server Notification System (notifying clients of activities)** - Working fine as shown above

Future Development

- One key area to improve is to give priority to those clients which are more active. Currently I am giving equal access to each client's thread in the server, but some clients would be more busy that other in real world scenarios. Threads should be given priority in terms of which clients to give more attention to.

- Currently there is a queue in the server which is a First IN First out basis, but we can also maintain a priority queue giving clients an additional functionality to choose the priority of a particular message.

- The GUI can be made with more effective looks with better use of Java Swing Library. Currently it is set to a very basic version.

- A GUI for server needs to be implemented for better visualization and controls.

Conclusion – Solution Summary

This report has described the design and implementation of a successful implementation of Client Server Chat Program using Java Socket Library for connection and Java Swing package for the UI.

It also shows the successful execution of all the required scenarios via screenshots.

Overall learning experience was good. It was captivating to see the role of multi-threading in socket programming. Java Socket library seemed to be pretty mature and performance achieved was good.

The UI library ie. Java.Swing seemed to be very complicated and outdated. Since most of the applications nowadays are web-based, I felt it would be better if we made a web - based project instead of desktop based.

- ## Appendices

    - References (if any)

Below is the link to the website which I used to get a grasp of Java Socket programming. I found this personally very helpful for my implementation.

https://www.baeldung.com/a-guide-to-java-sockets

For Java swing tutorial - https://www.javatpoint.com/java-swing

Some helpful youtube videos -

https://www.youtube.com/watch?v=3XB3in9Xqy8&list=PLfyq5A05w62_verkKr3DWKWbttvdgexH3