**Problem Statement**

**Chat Application using socket and Multi-threading**

**Group No.7**

**Design Document**

**1.Introduction:**

The main objective of the chat room project is to create a chat application which helps different clients or users to communicate with each other through a server connected.The main objective of the Simple Chat Room project is to create a chat application which helps different users to communicate with each other through a server connected. This is a simple chat program with a server and can have many clients. The server needs to be started first and clients can be connected later. Simple Chat Room provides a bidirectional communication between server and clients. It enables users to seamlessly communicate with each other. The user has an option to login to the chat room. The user can chat using this chat application. The server will display the user names who are active in the chat room and make those users visible. If the user at the other end is active then they can start a chat session.

**1.1 Project purpose:**

creating a chat box to communicate between different clients and a server.

**1.2 Functionalities of the system :**

These functionalities have been implemented in the the program :

PUT : Transfer a file from client to server

GET : Transfer a file from server to client

MGET (example : MGET .txt): Transfer all files of a given extension from server to

client.

MPUT (example:MPUT .txt): Transfer all files of a given extension from client

to the server.

**1.3 Operating environment :**

Operating environment for Emulating SFTP using TCP socket are:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu/C++

**2.Software requirements :**

This application has two primary parts - Client and Server.

And the reliable transfer is ensured using sockets and multithreading.

**Server flow:**

The socket binds the listening file descriptor to the specified client.It creates a thread of its process for every incoming connection to handle the clients.Takes the necessary actions.

**Client flow:**

Checks for validity of arguments - needs server’s ip and port to fire up.Opens a TCP connection and starts listening to incoming connections.Binds the listening file descriptor to the specified port.

**3.UNIT TEST:**

**1.Client:**

SR1-UT1: Create a valid socket to initiate the communication.

SR2-UT2: The connect() system call connects the socket referred to by the file

descriptor sockfd to the address specified by addr. Server’s address

and the port is specified in the address.

SR3-UT3: Multiple clients connect to the server to communicate.

SR4-UT4: Server will display all the clients joined for the chat with their user

name.

SR5-UT5: Client messages will be displayed on server side and the

other clients side.

**2.Server:**

SR1-UT1: Creating a structure named Terminal and it contains name, id and

thread.

SR2-UT2:After the creation of the socket, the bind function binds the socket

to the address and port number specified in the address.

SR3-UT3:It puts the server socket in a passive mode, where it waits for the

client to approach the server to make a connection.

SR3-UT4:Creates a new connected socket,and returns a new file

descriptor referring to that socket.

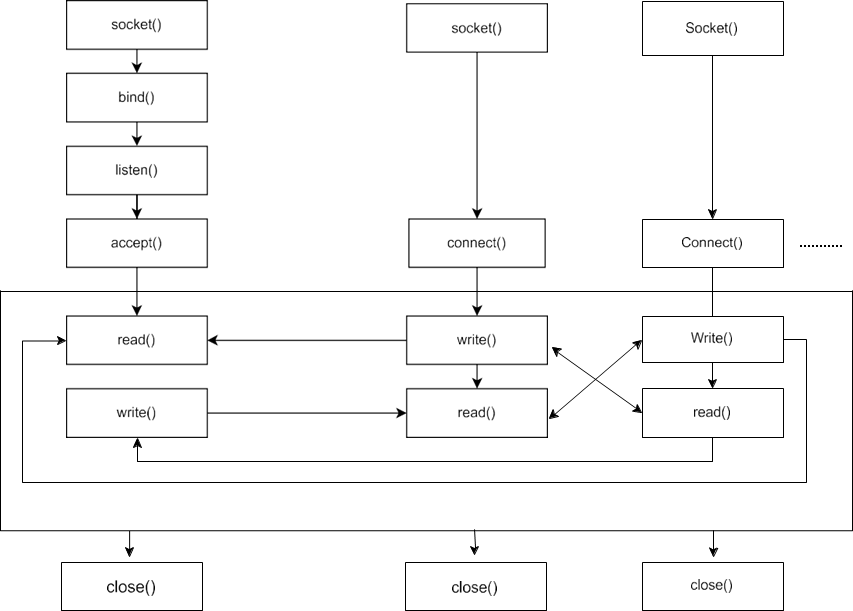
SR5-UT5: Send and receive the request

Use the read() and write() file descriptor to send and receive the data

from the server.

**3. FLOW DIAGRAM**

**Server process Client process Client process**

****

1. Start

This is the start block which indicates the start of the program,where socket is created and consists of server and multiple clients.where server accepts the clients credentials like username.Then the server will allow the client and give access for further communication.

1. Client Login

This is the module used for client login where the client can connect to the server by their username and then join to the server to communicate between multiple clients.

1. Login Credentials

In this module the credentials entered by the client are then validated by the system. If the client enters a valid user name then it will move to the further step.

1. Enter your domain name

Once the server has validated the credentials, it will now be connected to the server’s port number. Once connected to the server’s socket the client will access.

1. Catching the domain name

Once the server validate the credentials, the server socket will be created and will be binded to the server’s port no and now the server will remain listening on the port waiting for any client connection. Once a connection is established between the client, the server will catch the domain name entered by the client in this module.

1. Server provides respective IP address

When a domain name corresponding to the client entered domain name is found, the server will give the respective IP address to the client.

1. End

This ensures that the program has terminated.

**CONCLUSION:**

Hence by using chat applications multiple clients can able to communicate with each other with a server as a mediator.