**Software Requirements Specification**

1. **Introduction:**

It's a chatting system implemented in C - Programming Language by using TCP-Socket programming. In this project multiple Client can send and receive message between each other until they are connected to the Server. Clients can use multiple commands and command format provided by the server to communicate with the server and other actively connected clients via server.

**1.1. Purpose:**

* + It helps to communicate between multiple users in real time. TCP ensures reliable and accurate delivery of messages, and it also allows for synchronous communication.Single server allows the system to scale as more user join and it provides a centralized point for message handling and distribution.

**1.2. Scope:**

The scope of chatting system in C using TCP-socket programming with multiple clients includes creating a server-client architecture where the server manages connections and facilitates communication among multiple clients.

**1.3. Definitions, Acronyms, and Abbreviations:**

|  |  |
| --- | --- |
| TCP | Transmission Control Protocol |

**1.4. Overview:**

In chatting system we are using server side and client side. In server side we are using server connection in which socket is created using socket() function , socket binding is done by using bind() function. Use listen() to make server listen for connection. Use accept() to accept incoming client connection. Implement a loop to handle multiple client concurrently, For parallel client handling we use threads. We use send() and recv() to exchange messages between server and client.

For the client side, a similar setup would be needed, including socket creation, connection to the server and handling send/receive operations.

1. **Overall Description:**

While running the program open multiple windows first for server and then for multiple clients. Client programs when run should get response from server.

Message sent by Client1 intended for Client2 should reach client2 thru the server using TCP/IP socket programming.

**3.Specific Requirements:**

The specific requirements are –

**3.1. Functionality:**

**3.1.1 Socket Creation**

- To create end point for communication.

**3.1.2 Bind**

- To assign the address to the socket.

**3.1.3 Listen**

- To listen for connections on a socket.

**3.1.4 Accept**

**-** To accept the connection on a socket. It extracts the first connection request on the queue of pending connections for the listening socket.

**3.1.5 Send**

- To transmit the message to another socket. It can be used only when socket is in connected state.

**3.1.6 Receive**

-To receive message from a socket. If no messages are available at the socket, the receive calls wait for message to array.

**3.1.7 LIST**

- Server command to get clients list and status.

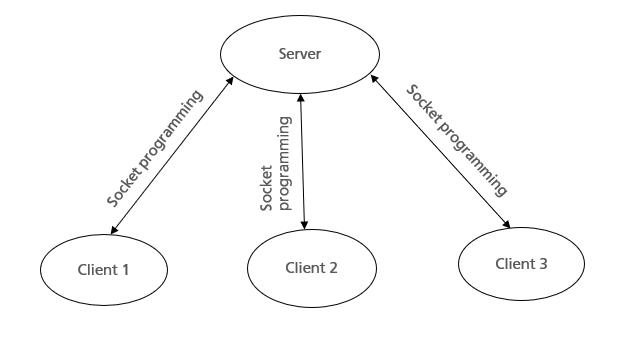
**3.1.8 HELP**

- Server command to get command list.

**3.1.9 EXIT**

- Server command to end the connection.

**3.2. Use Case Diagram:**



**3.3. Interface:**

The external interface compromises interfaces through which the users interact with the system

* Linux operating system.
* GCC Compiler.

Team Leader:- Deepika S

Team Members:-

Sairaj Bhagat

Sunil Patil

Punitha H R

Pavithra M