**MULTI PARTY CONFERENCE CHAT**

**MPCC\_DesignDocument-v0.1**

High Level Design & Low Level Design

.

**Document Control :**

| **Project Revision History** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 28/8/2022 | version 1 | Shreya, Rajesh, Nikhitha, Sneha | Class diagram, use case diagram, and some theory parts. | | | |  | |
|  |  |  |  | | | |  | |
|  |  |  |  | | | |  | |

[**1. Introduction 6**](#_heading=h.3vac5uf)

[1.1. Intended Audience 6](#_heading=h.2et92p0)

[1.2. Acronyms/Abbreviations 6](#_heading=h.tyjcwt)

[1.3. Project Purpose 6](#_heading=h.3dy6vkm)

[1.4. Key Project Objectives 6](#_heading=h.2afmg28)

[1.5. Project Scope and Limitation](#_heading=h.1t3h5sf) 6

[1.5.1. In Scope](#_heading=h.4d34og8) 6

[1.5.2. Out of scope 7](#_heading=h.2s8eyo1)

[1.6. Functional Overview 7](#_heading=h.17dp8vu)

[1.7. Assumptions, Dependencies & Constraints 7](#_heading=h.3rdcrjn)

[1.8. Risks 7](#_heading=h.26in1rg)

[**2. Design Overview**](#_heading=h.lnxbz9) **8**

[2.1. Design Objectives 8](#_heading=h.35nkun2)

[2.1.1. Recommended Architecture 8](#_heading=h.1ksv4uv)

[2.2. Architectural Strategies 8](#_heading=h.pkwqa1)

[2.2.1. Design Alternative](#_heading=h.2jxsxqh) 8

[2.2.2. Reuse of Existing Common Services/Utilities 9](#_heading=h.z337ya)

[2.2.3. Creation of New Common Services/Utilities 9](#_heading=h.3j2qqm3)

[2.2.4. User Interface Paradigms](#_heading=h.1y810tw) 9

[2.2.5. System Interface Paradigms 9](#_heading=h.4i7ojhp)

[2.2.6. Error Detection / Exceptional Handling 9](#_heading=h.2xcytpi)

[2.2.7. Memory Management 9](#_heading=h.1ci93xb)

[2.2.8. Performance](#_heading=h.3whwml4) 9

[2.2.9. Security 1](#_heading=h.2bn6wsx)0

[2.2.10. Concurrency and Synchronization 1](#_heading=h.qsh70q)0

[2.2.11. Housekeeping and Maintenance 1](#_heading=h.3as4poj)0

[**3. System Architecture**](#_heading=h.1pxezwc) **10**

[3.1. System Architecture Diagram. (Not Necessary) 1](#_heading=h.49x2ik5)1

[3.2. System Use-Cases 1](#_heading=h.2p2csry)1

[3.3. Subsystem Architecture](#_heading=h.147n2zr) 12

[3.4. System Interfaces](#_heading=h.3o7alnk) 12

[3.4.1. Internal Interfaces](#_heading=h.23ckvvd) 13

[3.4.2. External Interfaces 1](#_heading=h.ihv636)3

[**4. Detailed System Design**](#_heading=h.32hioqz) **13**

[4.1. Key Entities](#_heading=h.1hmsyys) 13

[4.2. Detailed-Level Database Design](#_heading=h.2grqrue) 14

[4.2.1. Level - 0 Diagram 1](#_heading=h.39kk8xu)4

[4.2.2. Level - 1 Diagram](#_heading=h.1opuj5n) 14

4.2.3. Level - 2 Diagram 15

4.2.4. Data Mapping Information 16

4.2.5. Data Conversion 16

4.2.6. Requirement Specification 17

4.2.6.1. Client Server Connection 17

4.2.6.2. Client Registration 17

4.2.6.3. Client Authentication 17

4.2.6.4. Active Client Session 17

4.2.6.5. Concurrent Server to Serve Multiple Clients 17

4.2.6.6. Message Broadcast 18

4.2.6.7. Session Termination 18

4.2.6.8. Encryption and Decryption 18

[4.3. Archival and retention requirements 1](#_heading=h.1v1yuxt)8

[4.4. Disaster and Failure Recovery](#_heading=h.4f1mdlm) 18

[4.5. Business Process workflow](#_heading=h.2u6wntf) 19

[4.6. Business Process Modeling and Management (as applicable)](#_heading=h.19c6y18) 20

[4.7. Business Logic](#_heading=h.3tbugp1) 20

[4.8. Variables](#_heading=h.28h4qwu) 21

[4.9. Activity / Class Diagrams (as applicable)](#_heading=h.nmf14n) 22

4.9.1 Class Diagram 22

4.9.2 Activity Diagram 23

4.9.3 Sequence Diagram 24

4.9.4 USE CASE DIAGRAM 24

[**5. Environment Description**](#_heading=h.48pi1tg) **25**

[5.1. Time Zone Support 2](#_heading=h.3ygebqi)5

[5.2. Language Support 2](#_heading=h.2dlolyb)5

[5.3. User Desktop Requirements 2](#_heading=h.sqyw64)5

[5.4. Server-Side Requirements 2](#_heading=h.1rvwp1q)5

[5.4.1. Deployment Considerations 2](#_heading=h.4bvk7pj)5

[5.4.2. Application Server Disk Space 2](#_heading=h.2r0uhxc)6

[5.4.3. Database Server Disk Space 2](#_heading=h.1664s55)6

[5.4.4. Integration Requirements 2](#_heading=h.25b2l0r)6

[5.4.5. Jobs 2](#_heading=h.kgcv8k)6

[5.4.6. Network 2](#_heading=h.34g0dwd)6

[5.4.7. Others 2](#_heading=h.1jlao46)6

[5.5. Configuration 2](#_heading=h.43ky6rz)6

[5.5.1. Operating System 2](#_heading=h.xvir7l)6

[5.5.2. Database 2](#_heading=h.3hv69ve)7

[5.5.3. Network 2](#_heading=h.1x0gk37)7

[5.5.4. Desktop 2](#_heading=h.4h042r0)7

[**6. References 2**](#_heading=h.2w5ecyt)**7**

[**7. Appendix 2**](#_heading=h.1baon6m)**7**

# INTRODUCTION

The Multi Party Conference Chat should be able to broadcast messages from one client to all the other active clients connected to the server.

## 1.1 Intended Audience

This project is a prototype for everyone who wants to communicate with multiple users at a time. This document is intended to be read by developers, testers, project managers and customers. This is a technical document, and the terms should be understood by all of them.

## 1.2 Acronyms/Abbreviations

| MPCC | Multi Party Conference Chat |
| --- | --- |
| TCP | Transmission Control Protocol |

## 1.3 Project Purpose

Purpose of this project is to provide communication between multiple users. Whenever the client sends a message, it will be broadcasted to all other active clients who are connected to the server.

**1.4 Key Project Objectives**

* Establishing connection between client user and the remote server.
* User Registration and authentication.
* Broadcasting of user messages.
* Encryption and Decryption of login credentials and messages.

## 1.5 Project Scope and Limitation

The scope and limitation of the project are listed below,

### 1.5.1 In Scope

In this project the multi party conference chat allows multiple clients to connect with a server and each client will have their own unique user ID, when each client sends a message it will be broadcasted to all other active clients and these messages will be displayed along with the respective user ID.

### 1.5.2 Out of scope

Operating Systems such as Windows are out of scope. Access to the client is only after successful registration with the same  port and IP Address.

## 1.6 Functional Overview

MPCC works in a client-server model which uses TCP/IP protocol.  The MPCC allows the user to get registered and upon successful registration communication with other users is enabled .It also enables broadcasting of messages sent by one user to all the other users active in the conference chat. The MPCC supports encryption and decryption of login credentials such as user ID and password. The server performs client authentication and broadcasts the message from one user to all the other users in the chat.

## 1.7 Assumptions, Dependencies & Constraints

Issues facing if any to avoid we need a unix like environment to perform the task and reliable connectivity while file searching.

## 1.8 Risks

To describe the risk, we are considering the user must have the Linux platform and any Linux operating systems having good internet connection to avoid the uninterrupted failures.

# DESIGN OVERVIEW

This system includes a server which is able to handle multiple clients at the same time.

## 2.1 Design Objectives

* + First, the server must be configured to start its service at the boot time.
  + The  users can start sending messages after successful registration.
  + The server will record the entries of the client user and broadcast the messages.

### 2.1.1 Recommended Architecture

Recommended System architecture are as follows,

1. Server-side Architecture:

1. RAM - 1GB or more.
2. Processor - 500MHz or higher.
3. Disk Space - 120GB or more.
4. Internet Connection - Yes.

2. Client-side Requirements:

1. Desktop or Linux Machine.
2. Need Terminal - Yes.
3. Internet Connection - Yes.

## 2.2 Architectural Strategies

The architectural strategy used in this system is that the concurrent server can serve multiple clients. The server accepts the request of multiple client users to connect in session to start communication. The server should be available to establish connection with client users and reconnect if the connection is lost.

### 2.2.1 Design Alternative

This system is based on TCP/IP protocol to establish connection between the server and client users. An alternative to TCP/IP is the UDP Protocol. The TCP/IP protocol is used as it is a reliable protocol which provides acknowledgement regarding the packet

### 2.2.2 Reuse of Existing Common Services/Utilities

This system uses the existing common services or utilities such as, TCP/IP protocol, file handling and other system calls.

### 2.2.3 Creation of New Common Services/Utilities

This system doesn’t create or use any new common service or utilities.

### 2.2.4 User Interface Paradigms

* Desktop or a linux machine.
* Stable internet connection.
* Command Line Interface.

### 2.2.5 System Interface Paradigms

* Operating system based on UNIX.
* Linux Kernel Version: WSL2 4.4.0-19041-Microsoft
* Bash shell: x84\_64 GNU/Linux.

### 2.2.6 Error Detection / Exceptional Handling

* Error detection will be provided for all possibilities that happen during the server and client connection.
* There are four levels of debug log messages that will be included like FATA, INFO, WARNING, DEBUG.
* Appropriate error messages for file handling will be included in this system.

### 2.2.7 Memory Management

NA.

### 2.2.8 Performance

TCP connections tune themselves over time, initially limiting the maximum speed of the connection and increasing the speed over time as data is transmitted successfully. TCP organizes data so that it can be transmitted between a server and a client.

### 2.2.9 Security

In this system, encryption and decryption techniques are used to enhance the security. Any external user who's trying to access the session, the server won’t allow them. Normally, client users will get authenticated by the server to start the conversation. Registered user details can be stored in a file and accessible by server only.

### 2.2.10 Concurrency and Synchronization

Server and client will be in synchronization in receiving and sending messages. Server will handle multiple clients for sending and receiving messages. Each and every message can be viewed by both server and clients.

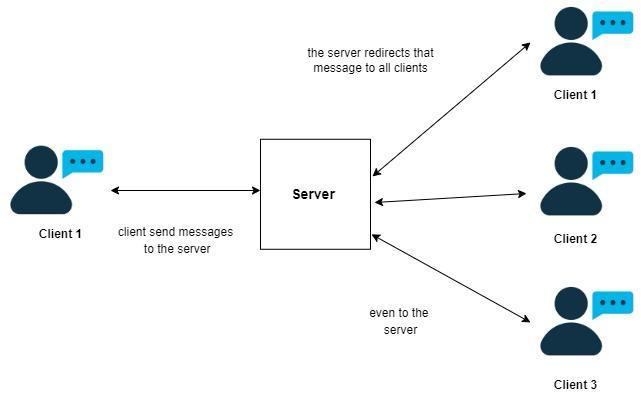
### 2.2.11 Housekeeping and Maintenance

* Clearing the memory buffers from the system.
* Registered client users data will be maintained by the server.

# 3. SYSTEM ARCHITECTURE

The Multi party conference chat is mainly based on enabling communication with multiple users easily. The encryption and decryption techniques are used to ensure the secure communication among the users.

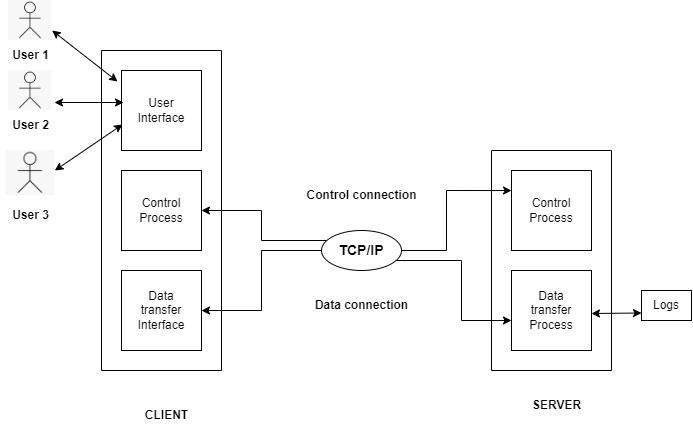
## 3.1 System Architecture Diagram. (Not Necessary)



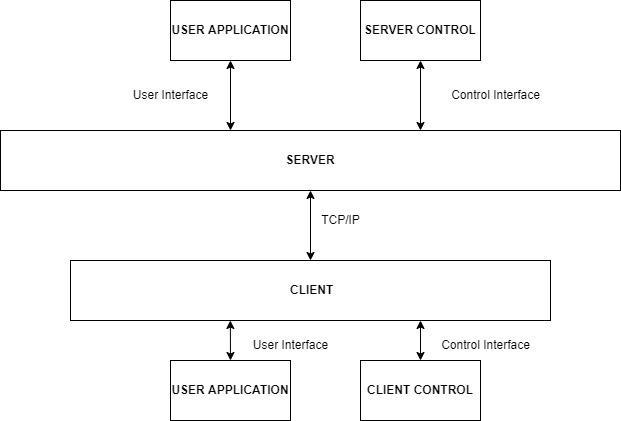
## 3.2 System Use-Cases

* To check whether a connection is established between the client and server.
* Authentication is required to allow only registered users to access the session.
* The server should have the ability to handle multiple clients at the same time.
* Whenever the client sends a message, it will be broadcasted to all other active clients who are connected to the server.
* In case of connection loss, the client has to reconnect.

## 3.3 Subsystem Architecture



## 3.4 System Interfaces



### 3.4.1 Internal Interfaces

NA

### 3.4.2 External Interfaces

The external interface comprises interfaces through which the users interact with the system.

* Desktop or Linux Machine
* Internet

# 4. DETAILED SYSTEM DESIGN

The Multi Party Conference Chat has a single server and multiple client connection. The connection is established using TCP/IP protocol.

Every client has to register through certain valid credentials and when one of the Client sends a message it will be broadcasted to all the other clients through the server.

**4.1 Key Entities**

The key entities associated with the system are:

Server:

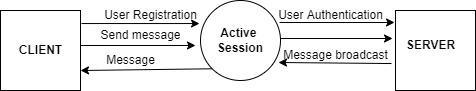
* The server stores the list of registered users in a file.
* It verifies the IP address and port of the client making the connection.
* Broadcasts the message request from one client to all the other users
* Logs are maintained regarding server functions.

Client:

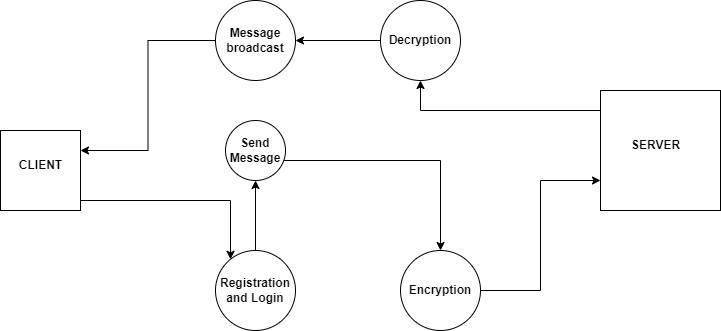
* The client is an entity that is run by the user on their system.
* It requests to connect with the server.
* Client requests the server to register itself using valid credentials and sends message request.

## 4.2 Detailed-Level Database Design

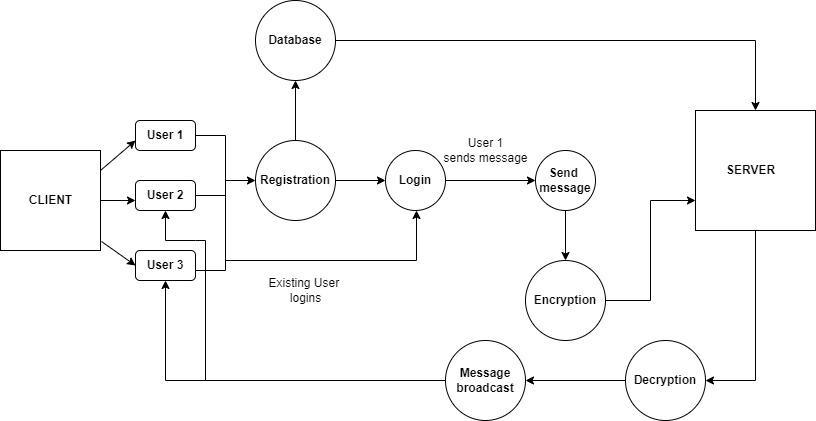
**4.2.1 Level - 0 Diagram**

****

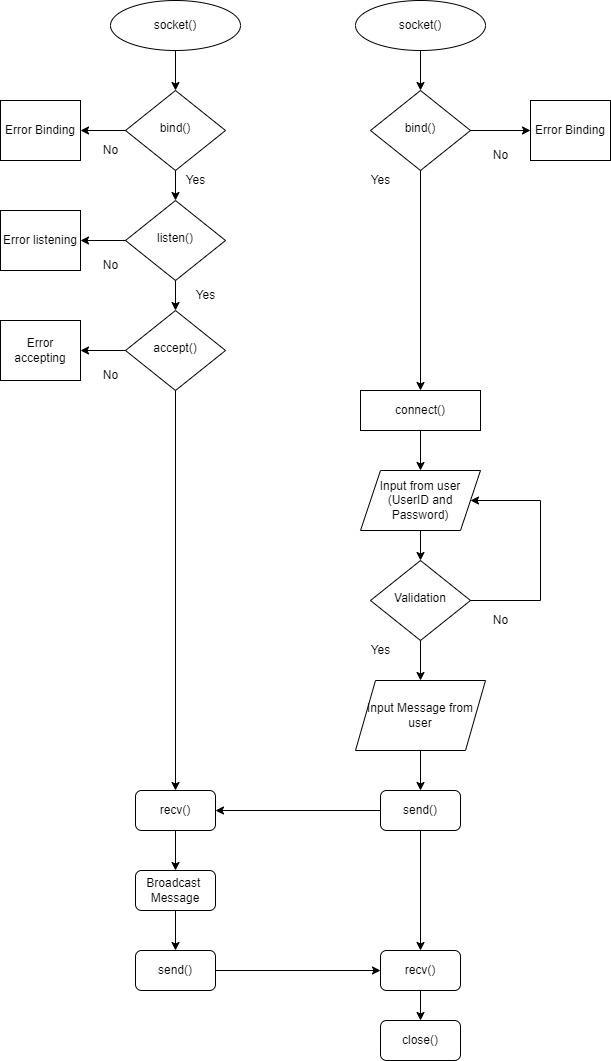
**4.2.2 Level - 1 Diagram**

****

**4.2.3 Level - 2 Diagram**

****

### 4.2.4 Data Mapping Information



### 4.2.5 Data Conversion

NA.

**4.2.6 Requirement Specifications**

**4.2.6.1 Client-Server Connection**

The Client-Server Connection is achieved by TCP/IP protocol. Transmission Control Protocol (TCP) is a connection-oriented protocol .

**4.2.6.2 Client Registration**

After we establish a successful connection between client and server, the users have to register using valid UserID and password at the server side. The registered credentials are stored into a file.

**4.2.6.3 Client Authentication**

Authentication is required to verify the credentials when the user logins to access the session. It is crucial because it's a key step in the process that keeps unauthorized users from gaining access to session.

**4.2.6.4 Active Client Session**

The server should be always available to establish connection with the clients. If the client connection is lost then the client needs to reconnect with the server.

**4.2.6.5 Concurrent Server to Serve Multiple Clients**

Concurrent Server can handle multiple clients at the same time. A concurrent server can be achieved by multithreading or multiprocessing. Every client will send the chat request and this has to be maintained by the concurrent server.

**4.2.6.6 Message Broadcast**

Multiple clients are connected with a server. Each client will have their own unique user ID. Whenever the client is sending a message, it will be broadcasted to all other active clients who are connected to the server.These messages will be displayed and it can be identified by the client’s user ID.

**4.2.6.7 Session Termination**

Every client will have their own session while connected to the server. Once the session is completed the client gets disconnected..

**4.2.6.8 Encryption and Decryption**

Encryption is the process of encoding information to prevent anyone other than its intended recipient from viewing it. Decryption is the process that transforms the encrypted information into its original format.

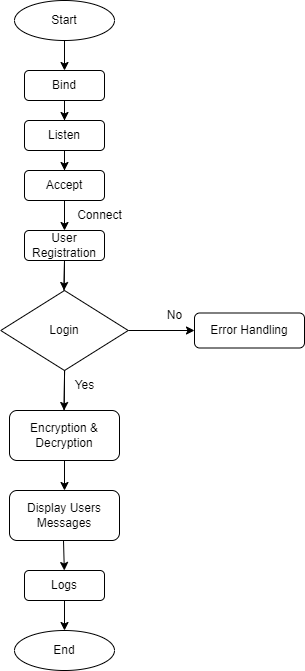
## 4.3 Archival and retention requirements

* Logs have been maintained by the server side, according to what function server is performing.
* RSE includes debug log messages with at least 4 levels as FATA, INFO, WARNING, DEBUG.

## 4.4 Disaster and Failure Recovery

NA.

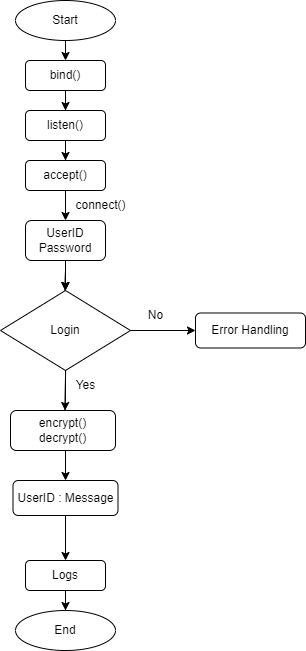
## 4.5 Business Process workflow



## 4.6 Business Process Modeling and Management (as applicable)

NA.

## 4.7 Business Logic



## 4.8 Variables

int port;

string IP Address;

int sockfd;

int newsockfd;

int RetBind;

int RetConnect;

struct sockaddr\_in server\_addr, client\_addr.

char server\_msg[1024].

char client\_msg[1024].

socklen\_t client\_len;

socklen\_t server\_len;

vector<string>vs\_sock;

char uid[20];

char password[];

## 

## 

## 

## 

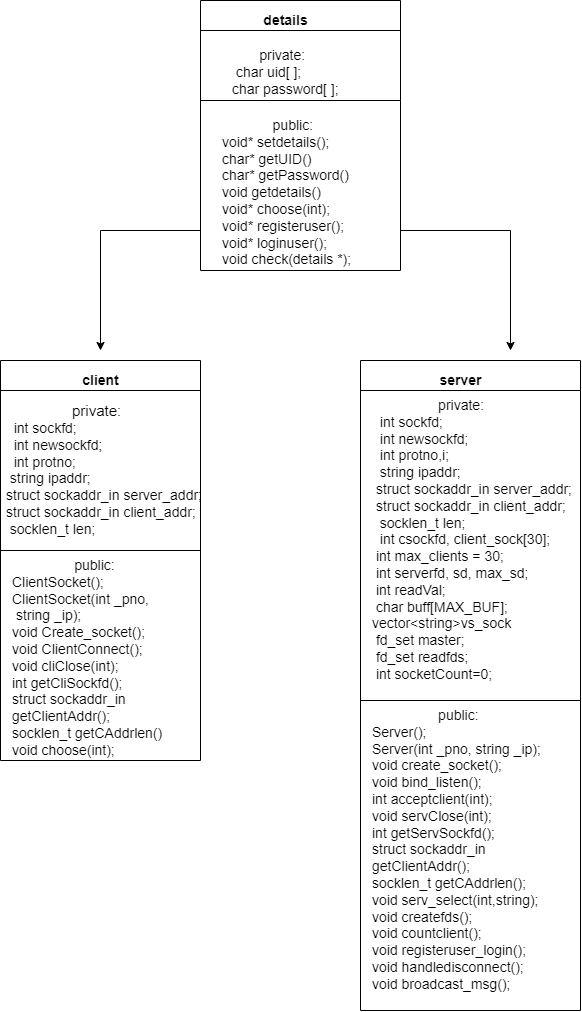
## 

## 

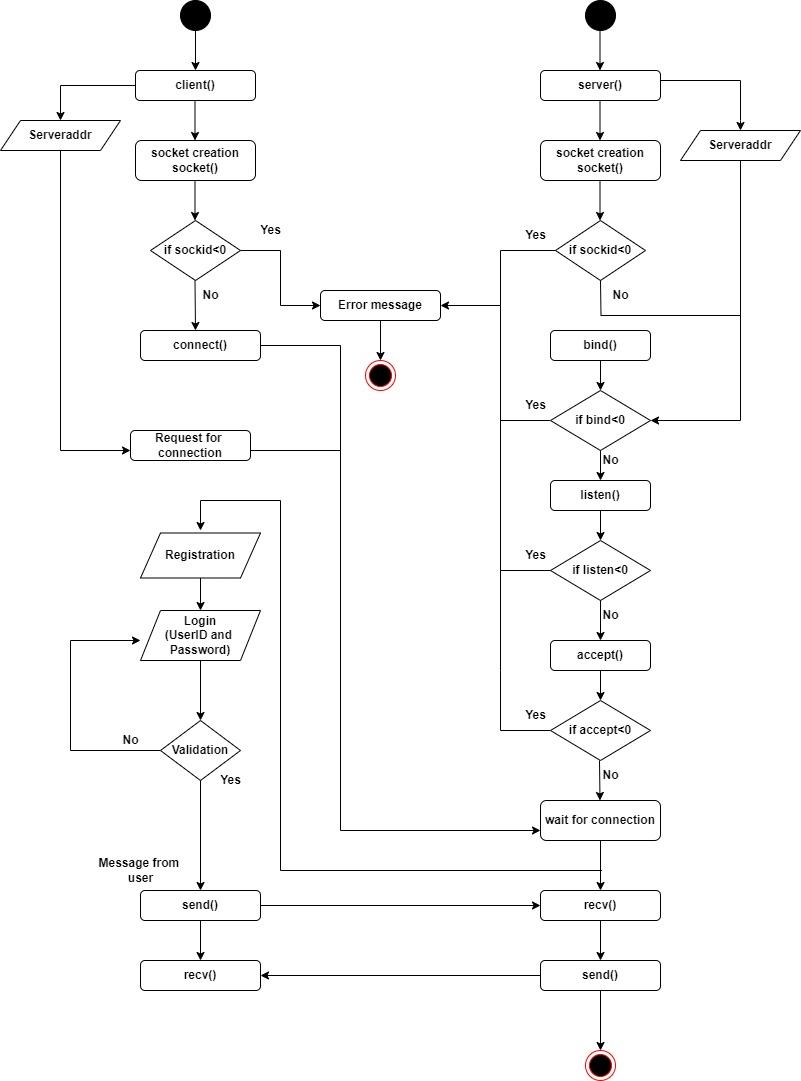
## 

## 4.9 Activity

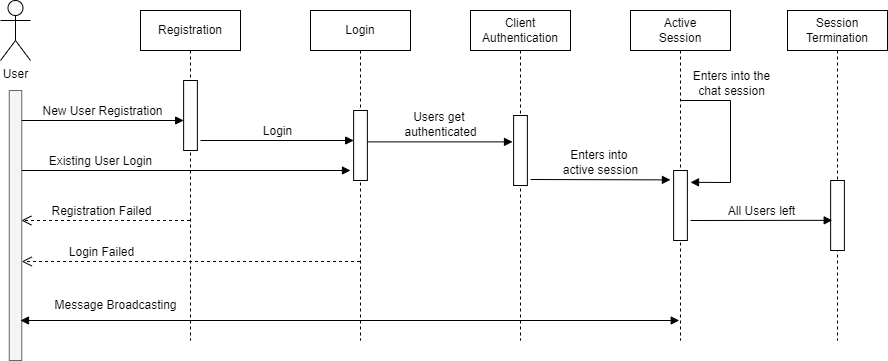
## 4.9.1 Class Diagrams (as applicable)



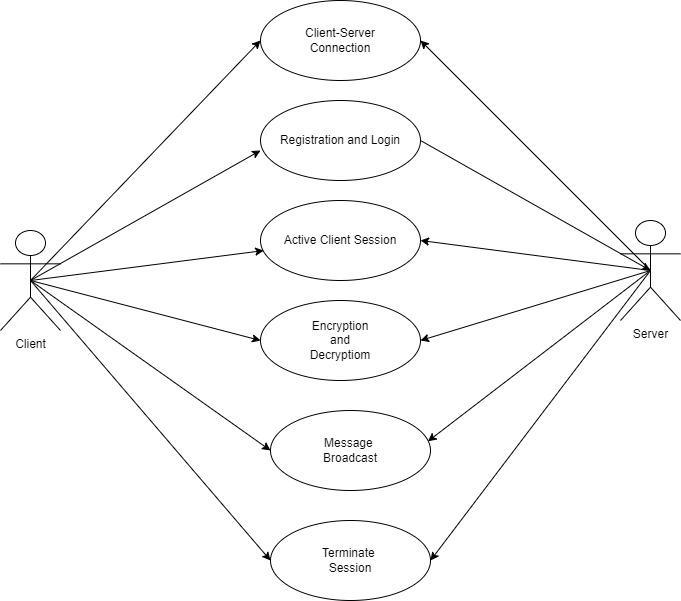
#### 4.9.2 Activity Diagram



#### 4.9.3 Sequence diagram:



**4.9.4 Use Case Diagram**

****

# 

# 5. ENVIRONMENT DESCRIPTION

The System allows multiple clients to connect to a server and every client will send a chat request, it will be maintained by the concurrent server. Each client  will have their own unique user ID and whenever one client is sending a message, it is broadcasted to all other active clients who are connected to the server. These messages will be displayed and it can be identified by the client’s user ID.

## 5.1 Time Zone Support

It will support time zones as per Indian standard time(IST) in (GMT +5:30) and UST standard.

## 5.2 Language Support

C++ language and compilation using g++.

## 5.3 User Desktop Requirements

## User desktop requires a Linux environment, Operating system of Linux Debian or Terminal x86\_64 GNU/Linux and Kernel version 4.4.0-19041-Microsoft #1237 Microsoft and reliable internet connectivity.

## 5.4 Server-Side Requirements

In server side,

* Disk space – Minimum 120GB.
* Uninterrupted connectivity 24x7.

### 5.4.1 Deployment Considerations

Deployment considerations are,

* 500Mhz Processor.
* 120GB HDD CPU.
* 4GB RAM
* Network connectivity.

### 5.5.2 Application Server Disk Space

Disk space – Minimum 120GB.

### 5.4.3 Database Server Disk Space

NA.

### 5.4.4 Integration Requirements

The PWD Displays the current working directory on the server for the logged in user.

### 5.4.5 Jobs

NA.

### 5.4.6 Network

The network connects the system for the purpose of file searching therefore stable Internet connectivity is required.

### 5.4.7 Others

NA.

## 5.5 Configuration

Multi Party Conference Chat works in a client - server model which uses TCP/IP protocol. The MPCC allows the user to get registered and communicate with other users. The server performs client authentication and broadcasts the message from one user to all the other users in the chat. It shall work in all standard UNIX based Operating systems.

### 5.5.1 Operating System

* Operating system – Unix.
* RAM - 8GB.
* Processor - i3/i5/i7.

### 5.5.2 Database

NA.

### 5.5.3 Network

The following are the network details regarding the project:

* The client and server communicate over a UDP/IP protocol.
* The IP address used can be either IPv4 or IPv6.

### 5.5.4 Desktop

Unix based environment is required.

# REFERENCES

1. <https://stackoverflow.com/questions/18681742/what-is-better-for-instant-messenger-tcp-or-udp>
2. System Requirements Specification Document
3. <https://www.ijert.org/research/end-to-end-multi-client-encrypted-windows-chat-system-IJERTV7IS030178.pdf>

# APPENDIX

NA.

**Change Log**

| **QMS Template Version Control (Maintained by QA)** | | | | | |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |