

# **Multi-user Chat Application**

## *Software Requirements Specification*

**Andre Durant,  
Mohammed Abbas  
Abhishek Pyakurel**

**7.15.2020**

## Revision History

[illegible]

## Table of Contents

1.		
Purpose.....		
... 4		
1.1.		
Scope.....		
4		
1.2.	Definitions, Acronyms, Abbreviations.....	
4		
1.3.		
References.....		
4		
1.4.		
Overview.....		4
2.	Overall Description.....	
5		
2.1.	Product	
Perspective.....		5
2.2.	Product	
Architecture.....		5
2.3.	Product Functionality/Features.....	
5		
2.4.		
Constraints.....		5
2.5.	Assumptions and	
Dependencies.....		5
3.	Specific	
Requirements.....		6
3.1.	Functional	
Requirements.....		6

3.2.	External Interface Requirements.....	6
3.3.	Internal Interface Requirements.....	7
4.	Non-Functional Requirements.....	8
4.1.	Security and Privacy Requirements.....	8
4.2.	Environmental Requirements.....	8
4.3.	Performance Requirements.....	8

# 1. Purpose

This document outlines the requirements for the Multi User Communications System.

## 1.1. Scope

This document will get a username and password from a user, then sign the user into the server in which the user can select a user that is also logged into the server to chat with. There will be a log of each chat conversation saved, and once someone leaves the chat room, the chat will be over between the two parties.

## 1.2. Definitions, Acronyms, Abbreviations

Server - program or device that provides functionality for other programs and devices called clients

Client - a piece of computer hardware or software that accesses a service made by a server.

Chat room - an area on the internet or computer network where users can communicate

Data log - process of collecting and storing data over a period of time

## 1.3. References

Use Case ID: 1

Use Case Name: User login

Relevant Requirements: The user must log into the server using a username. The user will then use this username to chat with other people on the server with their usernames.

Primary Actor: The login uses the server to login through the client to then connect to the server to connect to other users to chat to.

Pre-Conditions: Successful server activation and logging into the server so that the user can log into the server

Post-Conditions: User successfully logged into server using client and can connect to other users to chat.

Basic Flow: User enters login info. Client connects to server to put user inside server to chat.

Extensions: Each username can only be used once; clients will check the server and make sure the username is not used twice.

Exceptions: If server login fails, user will not be able to login.

Related Use Case: Use Case 2

## Use Case ID: 2

Use Case Name: Server activation

Relevant Requirements: Connection to server so that the client can operate on the server

Primary Actor: Connecting to a server using an IP address so that operations can be performed on it.

Pre-conditions: Successful server connection to IP.

Post-conditions: Successful connection between server and client

Basic Flow: Client inserts information into the server to operate on. Data keeps getting exchanged in this manner.

Extensions: Server can keep taking in clients.

Exceptions: If server login fails, will need to figure out connection issues.

Related Use Case: 1 and 3

### Use Case ID: 3

Use Case Name: Using server and client to chat with other users

Relevant Requirements: Getting a username to put into the client to use to login to chat with other usernames on the server. Then to send and receive messages using the client and the server to successfully chat between two usernames.

Primary Actor: Successful communication between server and client to let two users chat in the system.

Pre-conditions: Successful use of server and client to save users into the client to be able to chat with other users on the server.

Post-Conditions: Users can chat with other users on the server using the client.

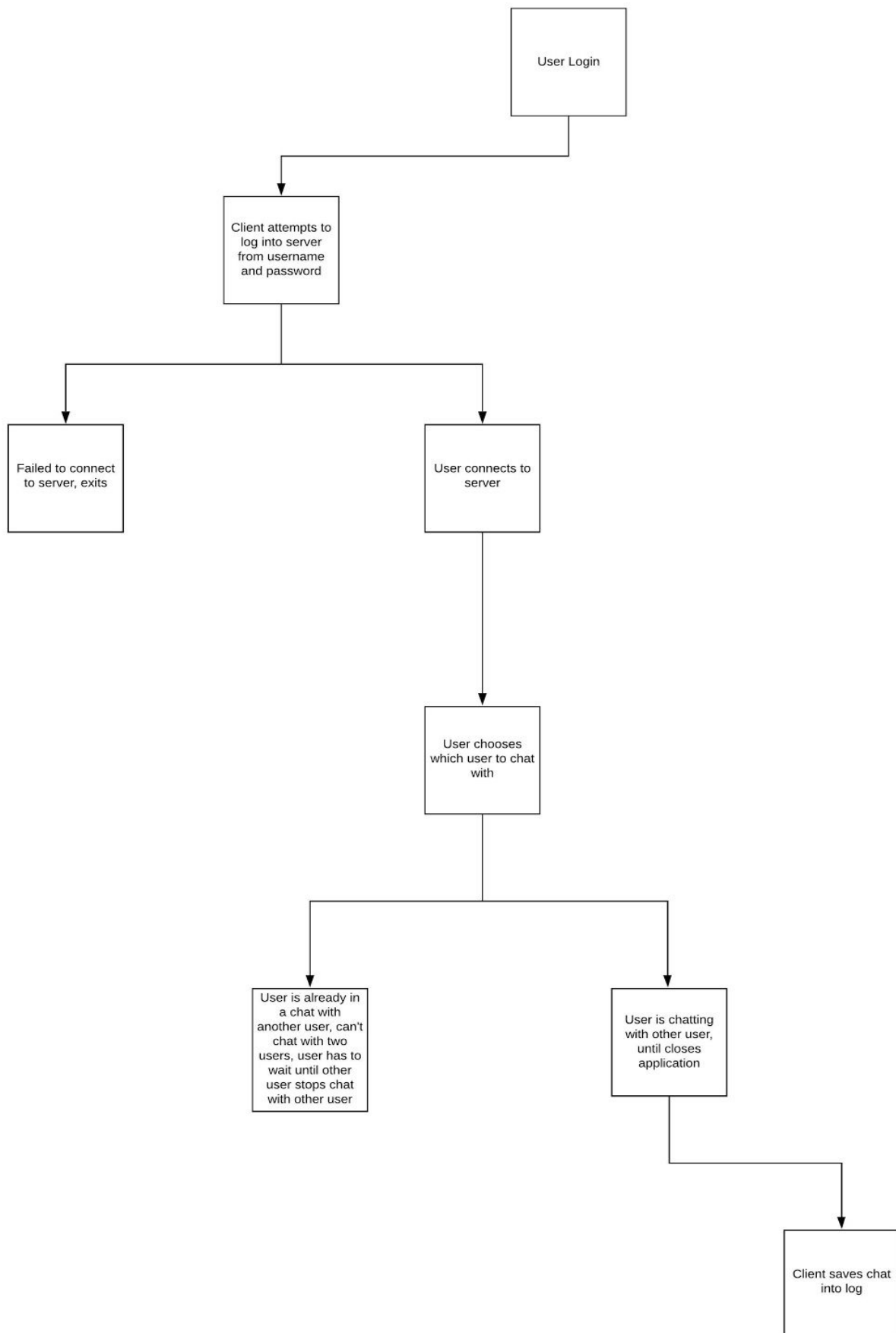
Basic Flow: Client logs in with username, then can chat with other usernames on the system.

Extensions: Chat is one on one with users

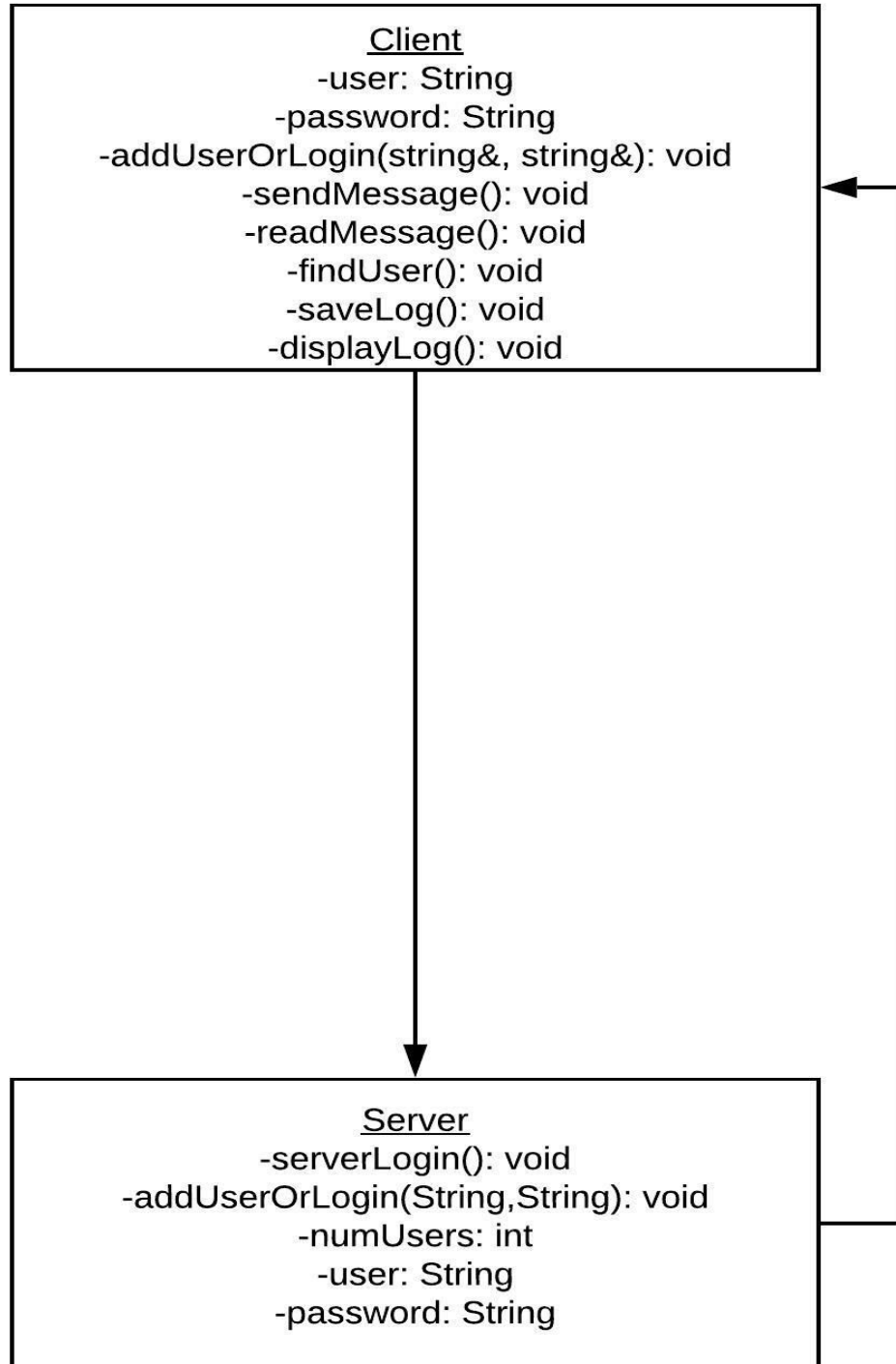
Exceptions: Usernames can only be used once, and the server must be running.

Related Use Case: 1 and 2

UML Diagram

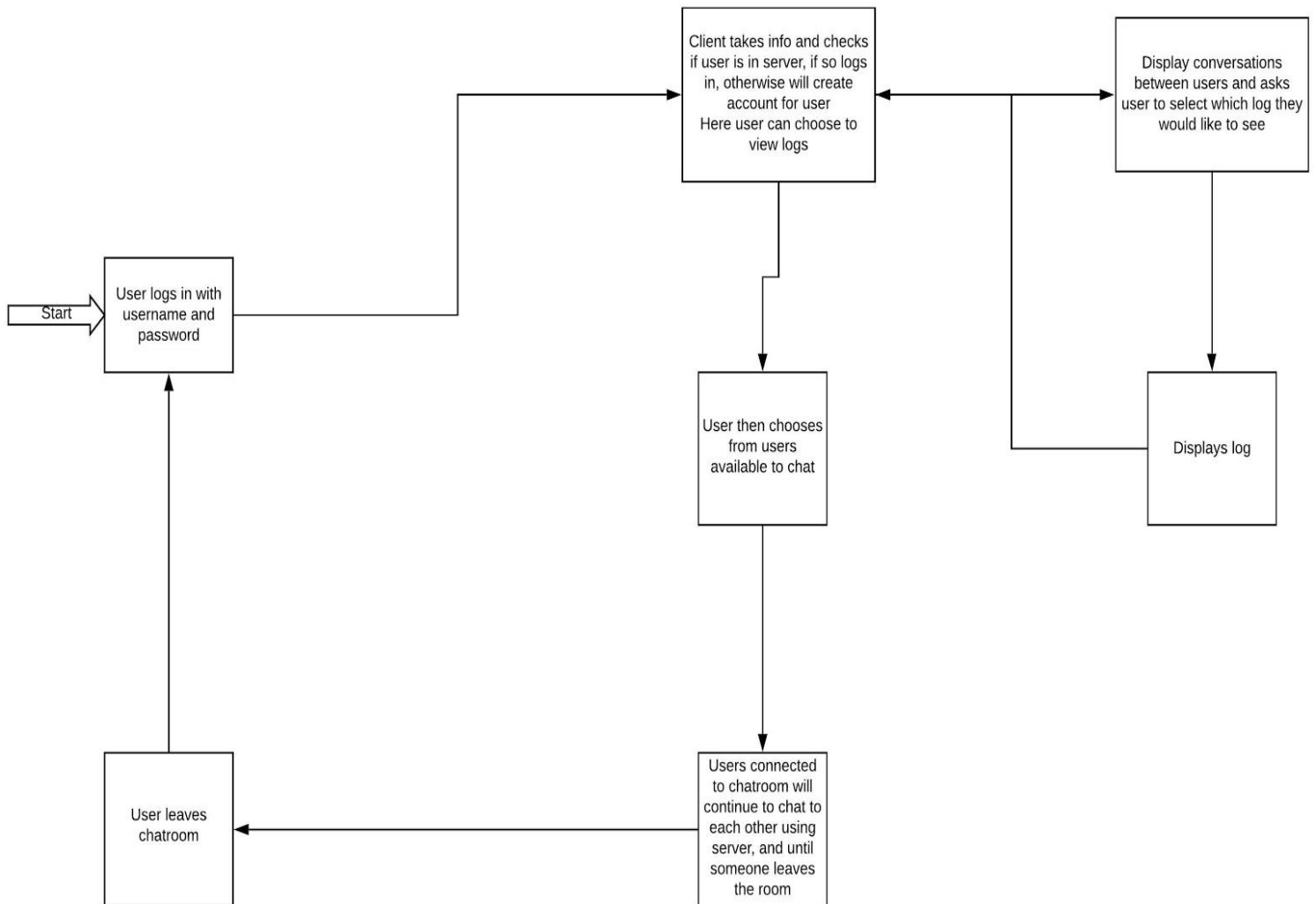


# Class Diagram





## Sequence Diagram



### 1.4. Overview

The Multi User Communication System will provide an instant messaging service between two users that are on the server. Each user will sign into the server using the client by providing a username and password. Users will then be able to chat with other users on the server that are currently online. There can be as many users as the server can handle, along with the ability to have as many chats between users that the server can handle. Each chat session between users will be recorded in a log saved into a file that will readily be accessible.

## **2. Overall Description**

### **2.1. Product Perspective**

### **2.2. Product Architecture**

The system will be organized into 2 main modules: the Server module that will handle communication and authentication between the clients and the Client module that will be the physical node that tries to communicate with other clients.

### **2.3. Product Functionality/Features**

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

### **2.4. Constraints**

The users of the system should have the option to create usernames that are unique to each user. There can not be two users that have the same login information.

Users of the system have to decide how many users are in the chatroom. If there is one chat already present then there can no be a chat available till the user leaves the room and joins again

The users must login successfully to be visible in the server.

User information will have to be accessed by the server.

### **2.5. Assumptions and Dependencies**

The users have logins that are unique to them and login successfully

The server can handle many users to be able to perform a multi user chat room

The system presents an option to make a chat once the user is available and seen in the server

Logs of all chats will be recorded and made available after conversation is finished

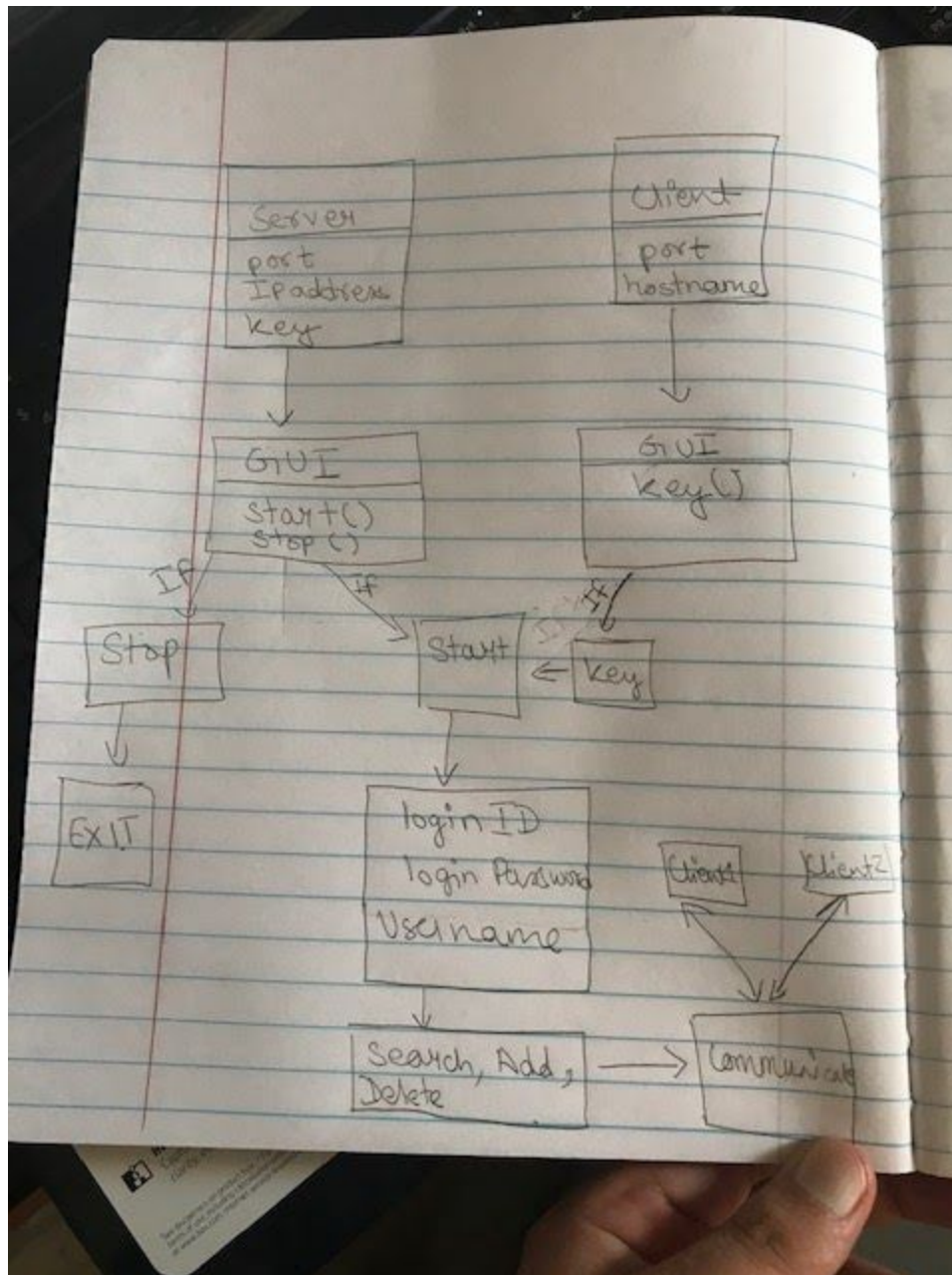
Users should be able to add and remove users from the chat

Server will hold user information, blocked user information, and will communicate with the client, while the client communicates with the GUI.

## **3. Specific Requirements**

### **3.1. Functional Requirements**

#### **3.1.1. Common Requirements:**



### 3.1.2. \_\_\_\_\_ Module Requirements:

In our communication application server is the main, that listens to incoming communication along with giving privilege to its right full members. All major functions are on servers. The primary function of Servers are as follows -:

1. Start connection, listen to client trying to connect
2. Server has a specific key
3. GUI for setting username and password.
4. After successful login, clients can add other users available on that server.
5. Some internal Functions of Server are as follows -:

- a. Search for user
- b. Chat with a user
- c. Delete user

### **3.1.3. \_\_\_\_\_ Module Requirements:**

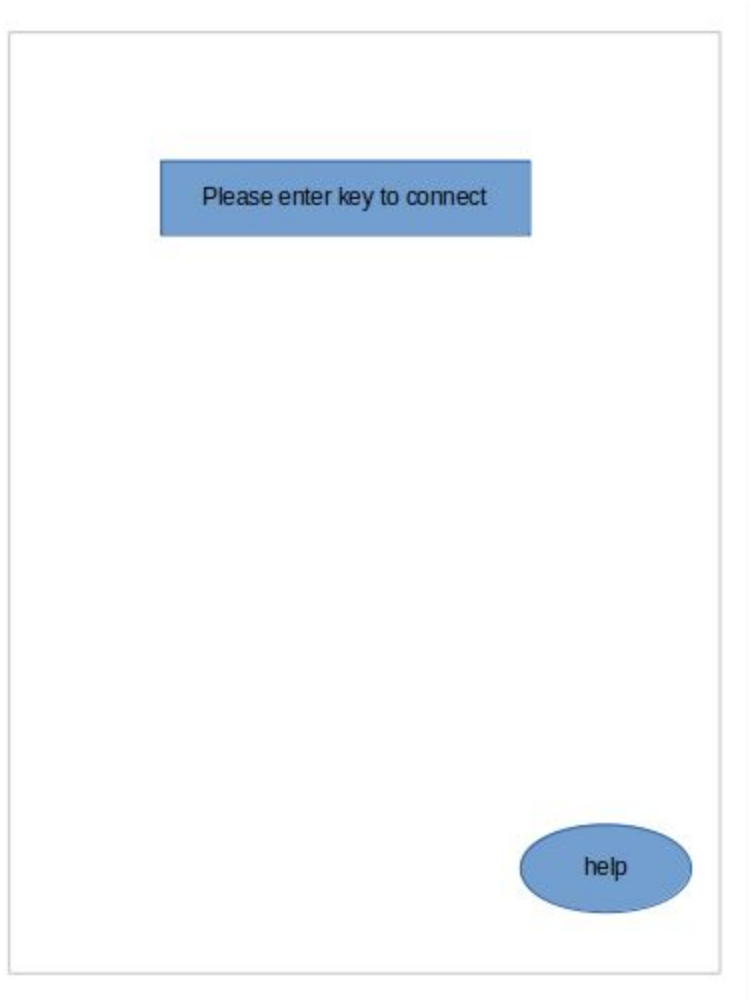
On our client-side, there will be a GUI asking for a key to connect to the server. Once the user provides the exact key the user can enter the Server to use our application. Key is similar to http address. The port and host name are self coded in our application. Users only need to provide a key. In addition there will be help to guide users too. GUI will communicate with clients.

### **3.1.4. \_\_\_\_\_ Module Requirements:**

The server-side authenticates all the users trying to log in to our communication chat. The database to store the added friend list will be handled by server-side. This will be more of an internal function of the application. List in the server will be checked by GUI in order to login users to chat with other users. Other users will be searched for to chat in this same manner. GUI communicates with clients.

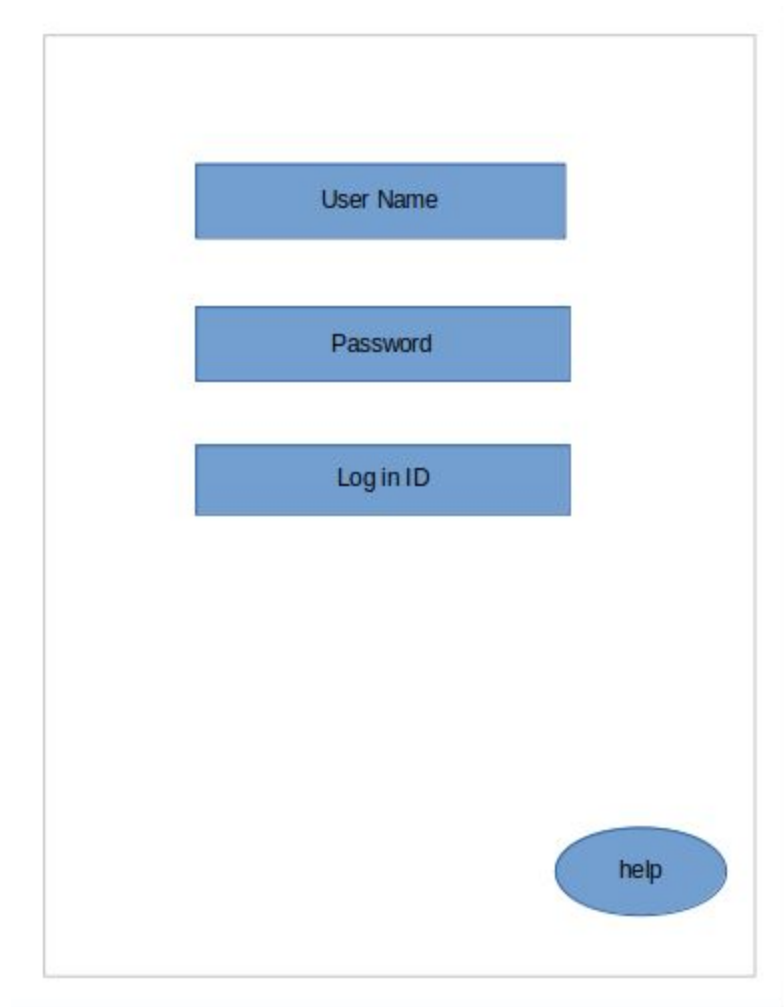
## **3.2. External Interface Requirements**

User interface of our client-side of our application:



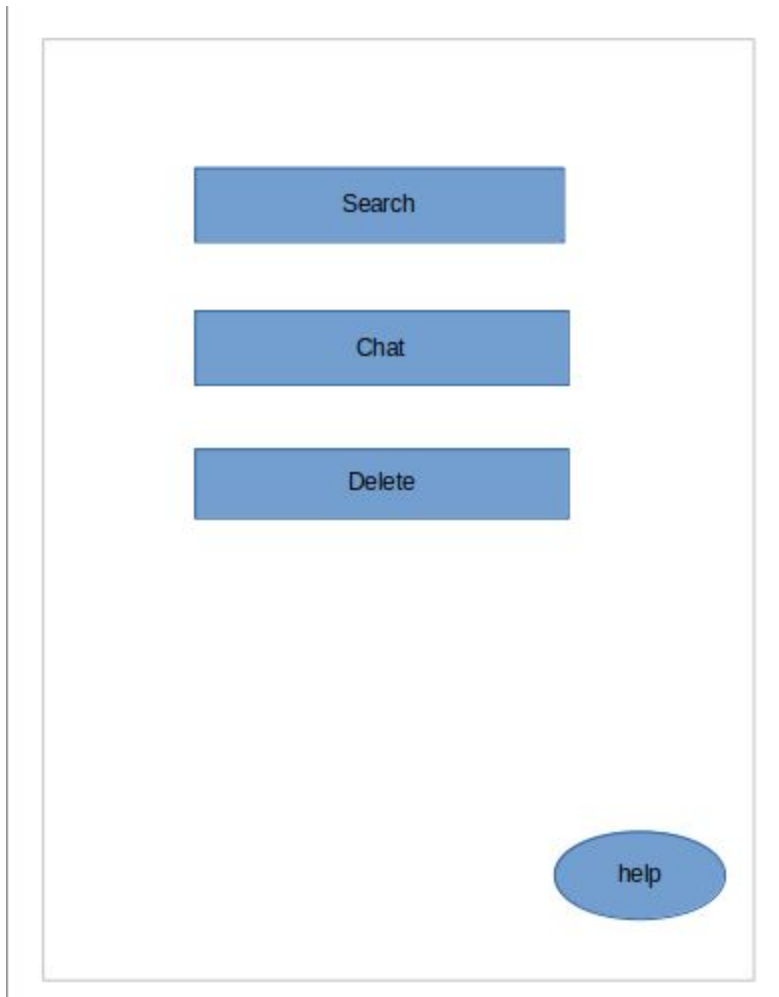
### 3.3. Internal Interface Requirements

User interface of the server-side of our application :



A diagram of a login form. It consists of a large light gray rectangle containing three blue rectangular input fields stacked vertically. The top field is labeled "User Name", the middle field is labeled "Password", and the bottom field is labeled "Log in ID". In the bottom right corner of the large rectangle is a blue oval button labeled "help".

After successful login, our user interface will include the function such as search, delete user. Along with that chat to communicate with the existing users/ friends.



## 4. Non-Functional Requirements

### 4.1. Security and Privacy Requirements

4.1.1 Our users are required to have numbers, alphabets and special characters at least 10 characters for password.

4.1.2 All of our users messages sent and received are required to be encrypted through crypto algorithms.

4.1.3 Data collected from users through our application are required to be stored in our database.

4.1.4 Data from our application's database will not be traded to any organizations, government agencies, or other applications for any financial purposes.

4.1.5 If a users data be needed to organizations, government agencies, or other applications, it cannot be done without users consent.



## **4.2. Environmental Requirements**

4.2.1 Our application is required to run on all mobile computer devices regardless of operating system used.

4.2.2 Our application should not require applications to be installed on laptops or desktops. It should be accessible through a web browser on any operating system.

4.2.3 Installation of our application on laptops and desktops should be optional for users and not a requirement. If a user wants to install our application, they should be able to install it in any operating system.

4.2.4 Users are required to be connected to the internet or use data provided by mobile communication companies to use our application.

## **4.3. Performance Requirements**

4.3.1 In mobile devices once a user opens our application, GUI and all of its subclasses should be collected and processed in no more than 10 seconds.

4.3.2 Once a user types username and password, our server should match it within 10 seconds.

4.3.3 Once a user types his messages and sends it, the receiver should receive it within 5 seconds.

4.3.3 Once a user logs out and closes our application, our GUI and its subclasses should exit within 10 seconds.