

# **Multi-user Chat Application**

## *Software Requirements Specification*

**Andre Durant,  
Mohammed  
Abhishek Pyakurel  
Suraj**

**6.15.2020**

## 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.

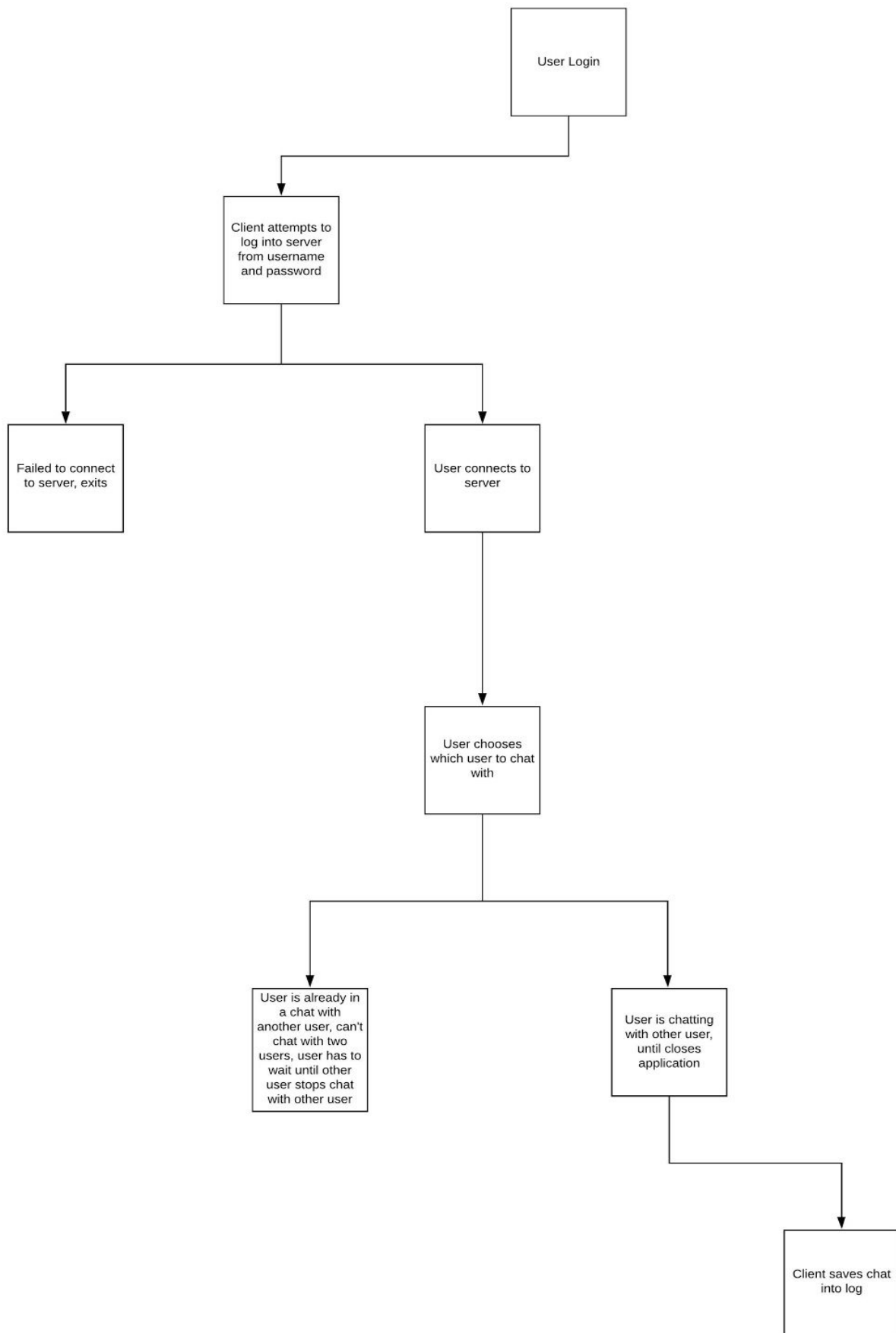
Basic Flow: Client logs in with username, then can chat with other usernames on the system.

Extensions: Chat is one on one with users

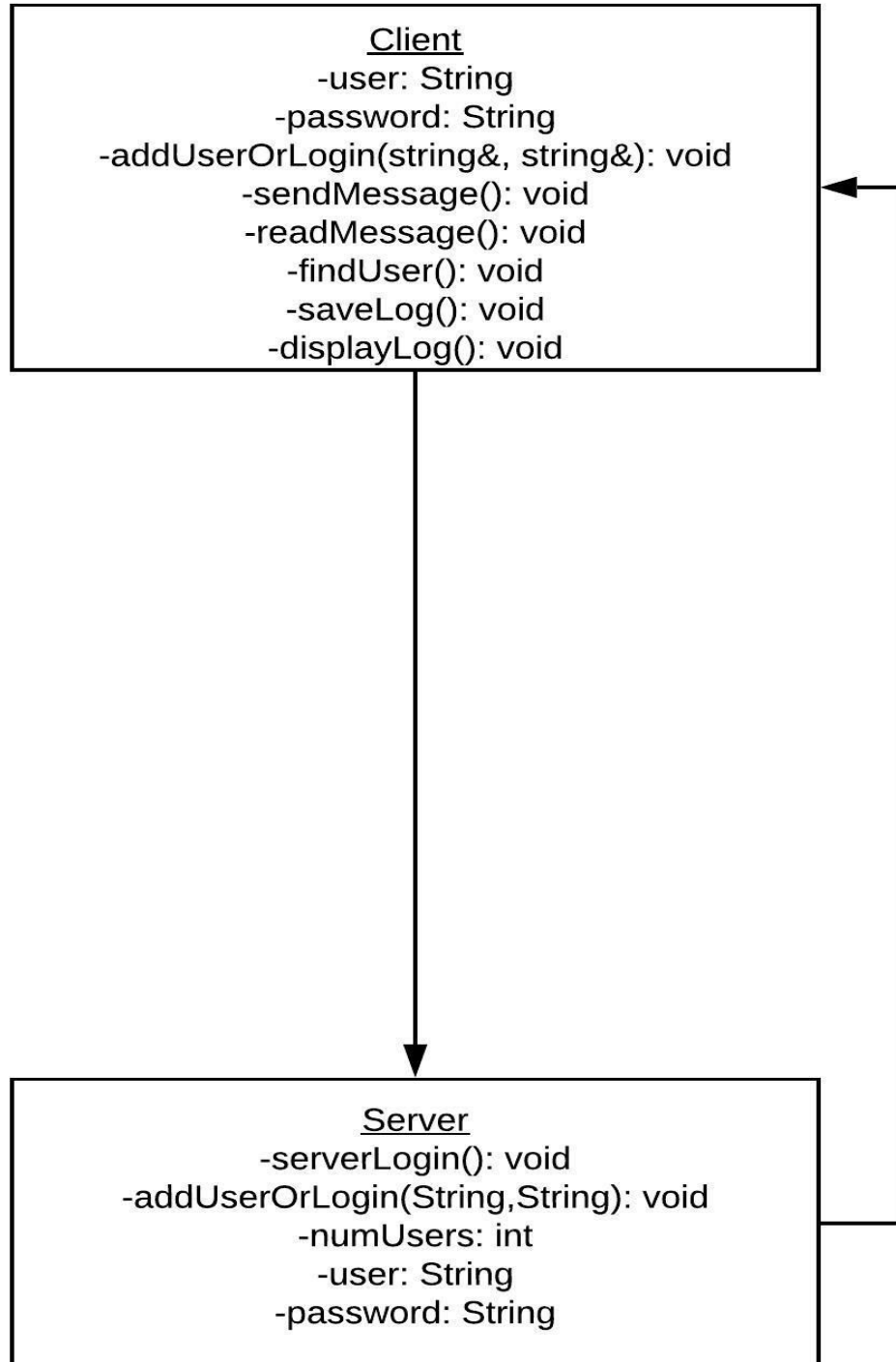
Exceptions: Usernames can only be used once, and the server must be running.

Related Use Case: 1 and 2

UML Diagram

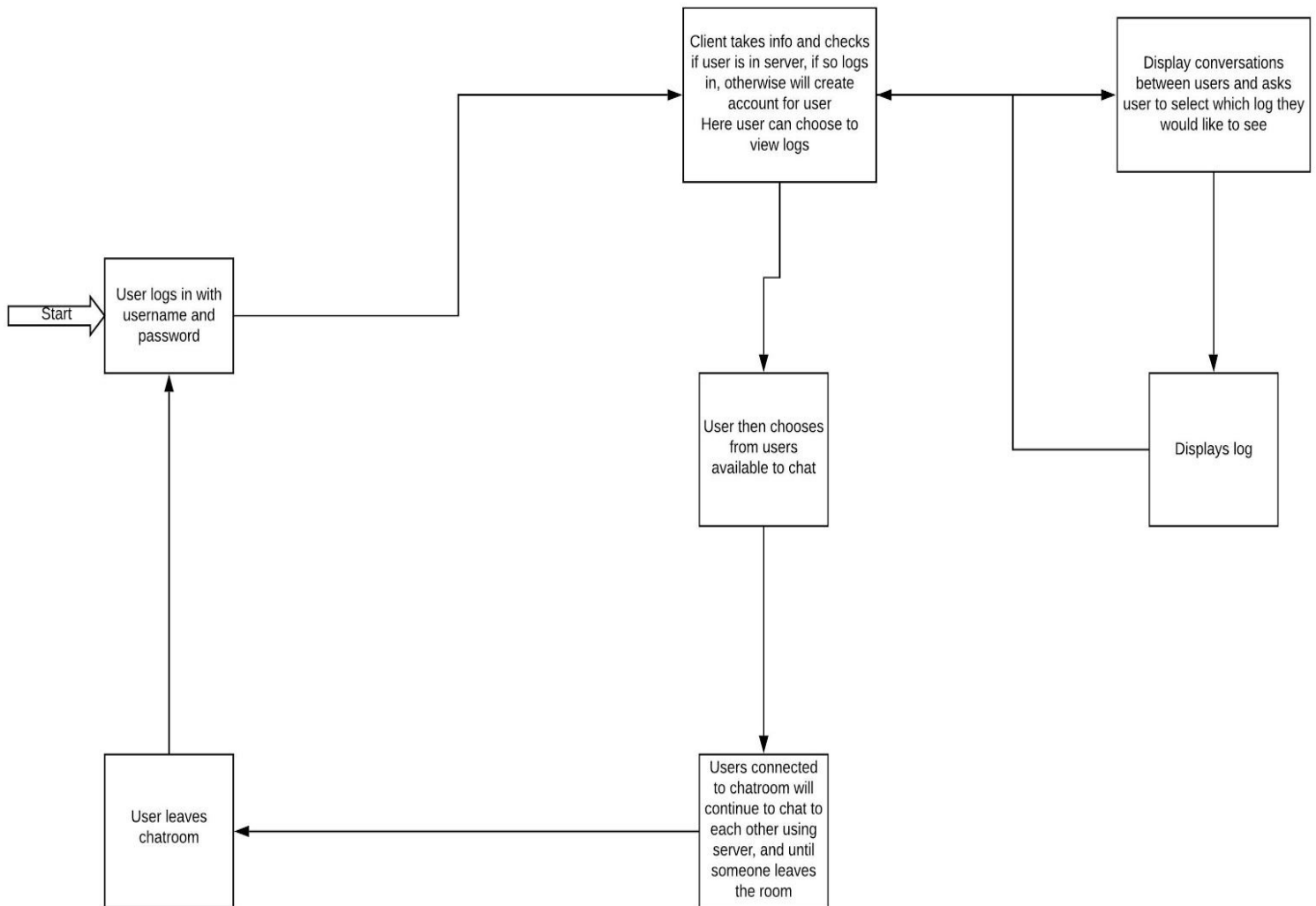


# Class Diagram





## 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 \_\_\_\_ major modules: the \_\_\_\_ module, the \_\_\_\_ module, and the \_\_\_\_ module.

Note: System architecture should follow standard OO design practices.

### **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**

List appropriate constraints.

Constraint example: SR7 Since users may use any web browser to access the system, no browser-specific code is to be used in the system.

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

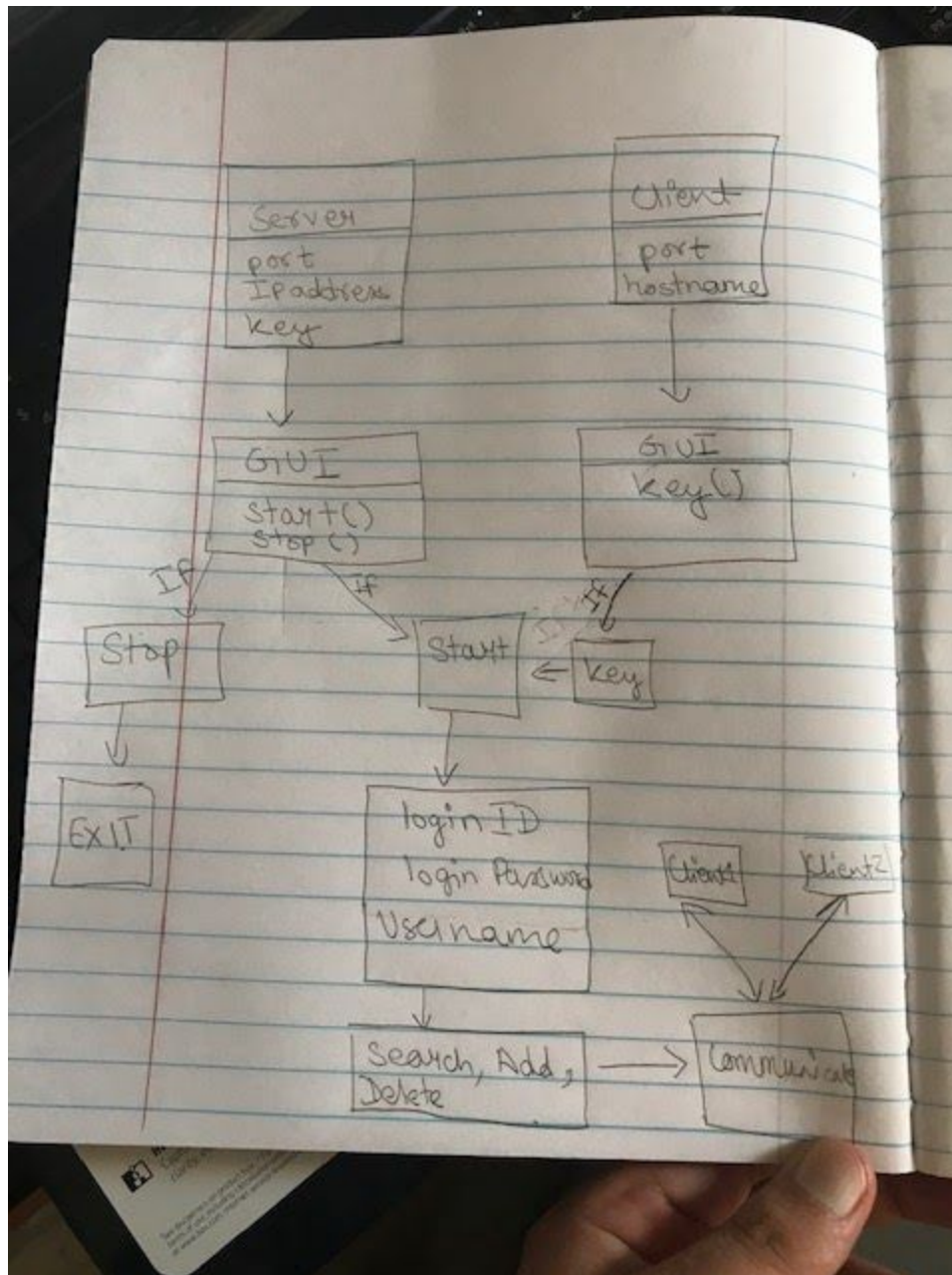
List appropriate assumptions

Assumption Example: It is assumed that the maximum number of users at a given time is 15,000.

## **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:**

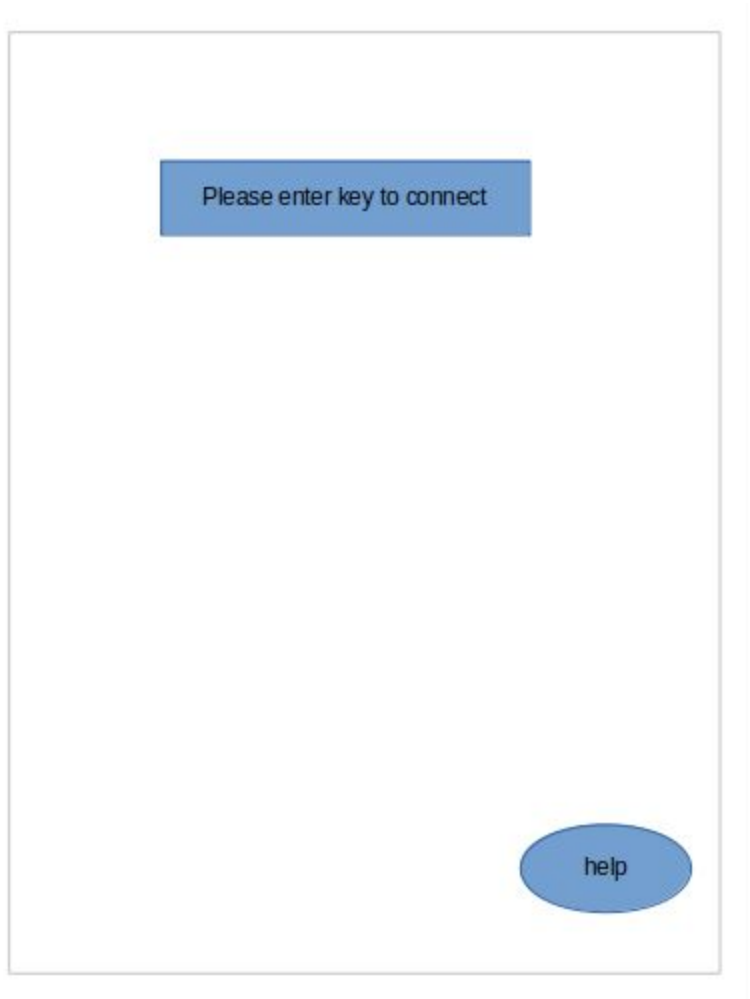
On our client-side, there will be a GUI asking for a key to connect to the server. Once the user provides the exact key the user can enter the Server to use our application. Key is similar to http address. The port and host name are self coded in our application. Users only need to provide a key. In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

### **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.

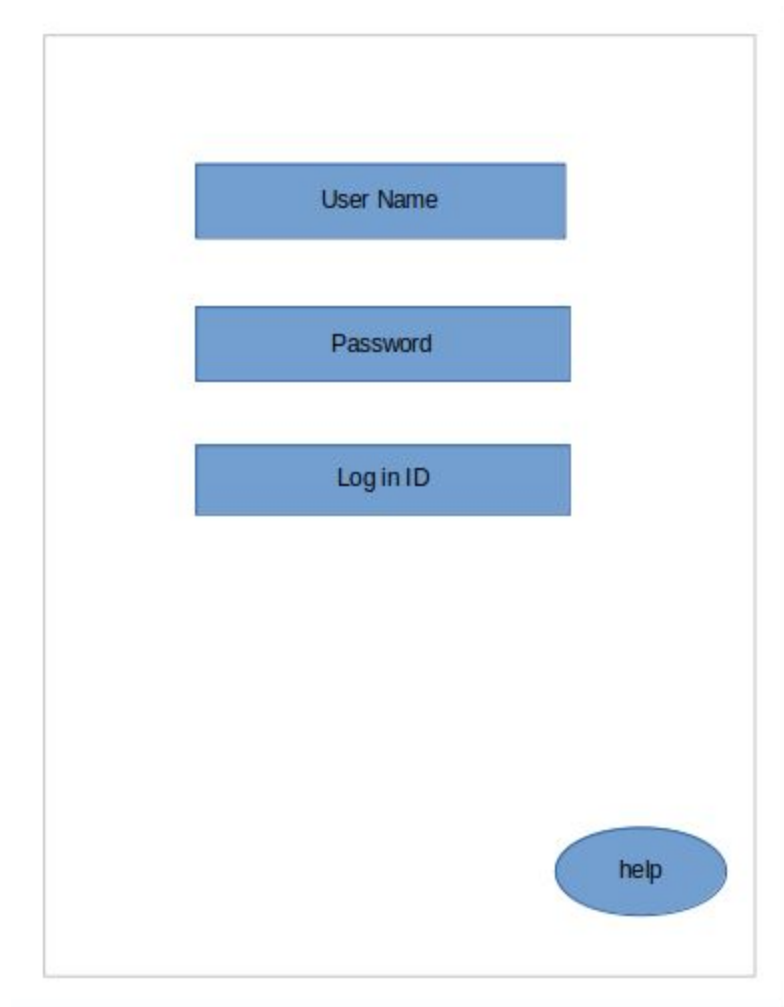
## **3.2. External Interface Requirements**

User interface of our client-side of our application:



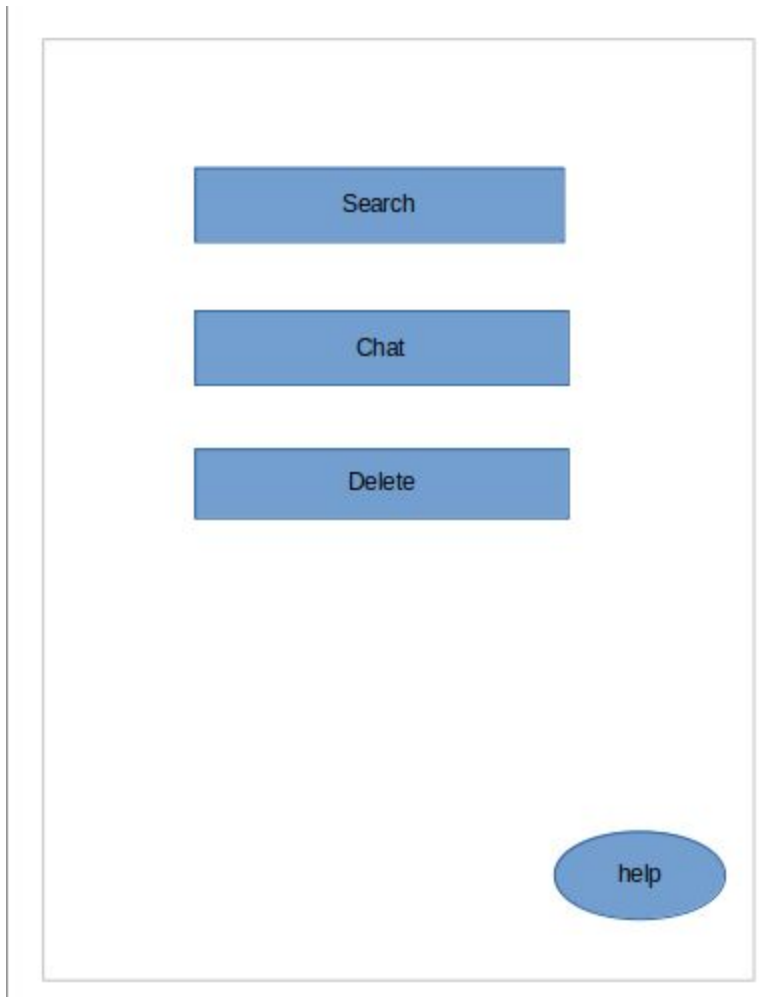
### **3.3. Internal Interface Requirements**

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

Example:

4.1.1 The SR8 System must encrypt data being transmitted over the Internet.

### 4.2. Environmental Requirements

Example:

4.2.1 SR20 System cannot require that any software other than a web browser be installed on user computers.

4.2.2 SR25 System must make use of the University's existing Oracle 9i implementation for its database.

4.2.3 SR26 System must be deployed on existing Linux-based server infrastructure.

## **4.3. Performance Requirements**

Example:

4.3.1 SR27 System must render all UI pages in no more than 9 seconds for dynamic pages. Static pages (HTML-only) must be rendered in less than 3 seconds.