**Manpreet Singh**

**Amna Tiwana**

**Umaimah Syed**

**Bhavan Patel**

**1 Introduction**

**This is the design document for the Multi User Communications System.**

**1.1. Goals and Objectives**

The goal of our application is to allow two users to communicate over the internet through a chat program. One of the objectives is to allow each user to login and search for their friends. In addition, through the client end UI system there will be effective communication and server end will be a median connecting two different users.

**1.2. Statement of Scope**

Decisions in this document are made based on the following priorities (most important first): Maintainability, Usability, Portability, Efficiency

Portability: The user needs to have internet connection to be able to access our server from anywhere. In addition, they need to be able to run the Java Program.

Usability: The chat system can be reusable if need be, where the transmission over the network is needed.

Efficiency: Efficiency is not the goal at the first implication of the chat system. In the process of the design implementation, the efficiency can be established.

Maintainability: Maintainability is low for the chat system assuming the number of users can control and operate by the server.

**1.3 Software Context**

The server class will be creating a socket which will connect users with the server and be able to send and receive messages.

**1.4. Major Constraints**

**Issue 1:** There should not be a space in the username, password, and security answer.

**Option 1.1**: Before sending the username, password, security answer to the server, there should be a check to see if there is a space or not

**Option 1.2**: The server will be designed in a way so that it accepts the username, password, and security answer without the spaces.

**Decision**: The decision that we made in this version, the server is assuming that there is no space in the username, password, and security answer. However, option 1.1 will be the best option for sign up and login.

**2.0 Data Design**

**2.1 Client side**

* The client can communicate with the server if there is an issue. They can send multiple messages to the server.
* Once the client is logged in, they can send and receive messages from other users.
* The messages from the server will be sent directly to the client’s chat box.
* The client can notify the server when their problem has been fixed.

**2.2 Server side**

* The usernames, passwords, security answer, and output stream will be stored
* Have an organized structure to prevent data leakage

**3.0 Architectural and Component-level Design**

**3.1 Program Structure**

The Chat system runs as a client-server application…

-This system makes sure that the client can send and receive messages from other users who have their email/username. They can also send and receive messages from the server. The server can notify the user if someone tries to log in with a different password.

**•3.1.1 Architecture diagram**

User

Server

chatbox

**3.2 Description of Client**

-Client is the user of the chat system who can make the account by entering a username, password, security answer. After signing up, they can login with their username and password.

**3.2.1 Client processing narrative**

- Client will make the connection, then verify the user, and then receive confirmation from the server, then proceed to launch the chat box

(functions and processes described)

**3.2.2 Client interface description**

-Clients will utilize the Listen\_Receive\_From User to send and receive messages.

-They will utilize the Listen\_Receive\_From User to communicate with the server.

(input and output interfaces described)

**3.2.3 Client processing details**

-The details of the client will be stored in an array.

-Users can add other users if they have the username.

(algorithmic description)

**3.3 Description of Server**

**3.3 Software Interface Description**

**3.3.1 External Interfaces**

-The chat system will require a database in the server where the chat and user’s data are being stored.

**3.3.2 Internal Interfaces**

-Hard Drive or the SSD on the server will interact with the server code and will provide the data as per the user request.

**3.3.3 Human Interfaces**

-Human interface is a simple GUI interface that will be presented to users to access their own personal data and to access the chat system.

*(an overview of the human interface to be designed is presented. The next section outlines the design in more details.)*

**4.0 User Interface Design**

-The user interface is divided into three parts: signup GUI, login GUI, and the message box. Through the signup and login GUI the user will get the message GUI after getting the authentication from the server.

**5.0 Restrictions, Limitations, and constraints**

-User can only reset their password if they answer their security question correctly otherwise there is no other way to rest.

-In order to chat with other users, the user needs to input the starting string as “msg” + username. (username should be the username of the person they want to chat with).

- only two users can communicate with one another at one time meaning there is no group chat

**6.0 Testing Issues**

* While chatting with the user, if the user sends the invalid message format then it can cause the testing issue.
* Lack of the testing tools and environment because the chat system will use the computer as the source of the server.
* If there is not enough testing that can be done by the JUnit then it can cause the testing Issues.
* ServerSide needs to run through the Junit testing in order to cover the server side class methods

**7.0 Appendices**

* SRS
* UML diagram
* Flow Chart