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 type program. One of the objectives it should be able to do is allow a login for each user and look up other users when logged in to the chat system. Some of the things we required client end is user interference allowing effective communication and server end will be 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. Also 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. But through the design implementation we can figure out.
Maintainability: Maintainability is pretty 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 create a socket that will connect users with the server and whatever messages or info that.

1.4. Major Constraints

Issue 1: Where should we store information regarding the establishment of blocking? The Information is stored in Account class for example username and password will be string but each username will have a unique private ID all the user data will be stored in Linked List in this Class.

Option 1.1: Once two users connect, start chatting a new linked list will be created to store their chat history and once they log out the information will be saved in that linked list allowing them to continue their chat when they log back in.

Option 1.2: Files get stored on the server end and each packet of information will be unique.
Decision: Once a user puts his login info his information is checked in the database there is also status attached to each user like active or banned this is pulled from enum class and is stored along with username and password system checks the status of user if its not banned then user is allowed access to the chat room. Otherwise the user will not login.

2.0 Data Design

The categories used should be specific to your project

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 server will receive a unique ID from the user
- The information will be stored in linked list
- The usernames, passwords, chat history and other data will be stored
- Will have access to the accounts privacy and can revoke
- Have an organized structure to prevent data leakage

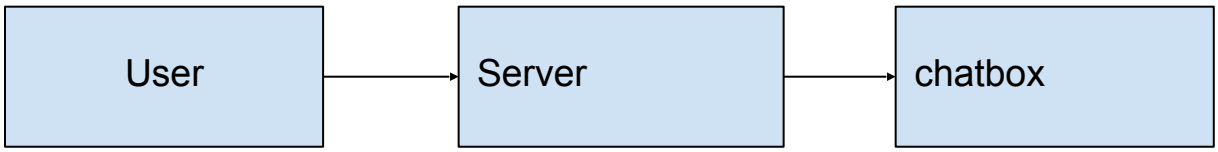
3.0 Architectural and Component-level Design

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



3.2 Description of Client

-Client is the user of the chat system who simplify make the account by the Email and password, so when next time user want to chat then login by the Username and password.

3.2.1 Client processing narrative

-Client will login with the user class and after getting the confirmation from Sever the user can start the chat or the change the account details by using the Logged In class and after done using the user can Logout by User class.

(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.

-The server can save the chat.

(input and output interfaces described)

3.2.3 Client processing details

-The details of the client will be stored in an array or a linked list.

-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 interference will flow the basic flow of the flow chart in SRS diagram but users will be able to check on buttons that make whatever task given execute for example log out will save the data and logout the user and save the chat.

5.0 Restrictions, Limitations, and constraints

-User can only reset his password if he answers his security question correctly otherwise there is no other way to rest.

- 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 invalide 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.**

7.0 Appendices

- SRS
- UML diagram
- Flow Chart