Andre Durant

Mohammad Abbas

Abhishek Pyakurel

Software Design Specification

1.0 Introduction

This program is a chat program that allows two persons to chat through a network to send and receive messages to each other. A user needs to log in with a username and password and needs to find the other username to chat with. A user can block other users if they want too so they can't contact them.

1.1 Goals and objectives

The goal of this project is to get all users to connect to the same port using a host, so that users can type in a username, have the program check if the username is in the server available, and start a chat with the username. The users can send and receive messages to each other until a user hits a button to end chat or type end in the chat to end the chat. A user can choose to block a user so that if a user tries to start a chat with someone that is on their block list, it won't be possible.

1.2 Statement of scope

Make sure that the gui is easy to use for the users chatting, making sure all features are easily accessible. This document will get 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 in chat with. We will use Hashmap for login and password. To avoid collisions we will not have two same usernames.

1.3 Software context

Make sure that the gui is easy to use for the users chatting, making sure all features are easily accessible. This document will get 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 in chat with. We will use Hashmap for login and password. To avoid collisions we will not have two same usernames.

1.4 Major constraints

Issue 1: Where should we store information regarding the establishment of blocking?
● All the username and password will be stored in a hashmap.
● With the hashmap, the difficult part is avoiding collision.
● Blocking is the easy part.
● We chose the option 1.1 for our implementation just like facebook.
Option 1.1: Make sure that blocking is easily accessible for all users that want to block another user. A hashmap is a way to guarantee that this method is easily accessible and the least amount of issues will take place.
Option 1.2: All blocked users will be stored in a specific array-like structure. This will be on the server and will be accessed by the client at all times when the program is running.
Decision: Information of the blocked users will be stored in an array-like structure that will be accessible for each user that will need another user to be blocked. This will be accessible between server and client at all times.

2.0 Data design

The client of the program will connect to the GUI along with being connected to the server. The client will be connected to the GUI and be given instructions by the GUI which is then passed onto the client in which the client will take that data and pass it onto the server to make sure that the data is transferred accordingly. The server will hold client information and will be passing information from client to client for the client to operate on.

2.1 Internal software data structure

Arrays will be passed. Hashmap key will play a major role in searching as well as blocking

2.2 Global data structure

Array and Hash map..

2.3 Temporary data structure

We will store all the information in file. Read and write username and password to file. Along with block and add friends.

2.4 Database description

We will use Array as a database. And Hash map for blocking and searching and sorting.

3.0 Architectural and component-level design

A description of the program architecture is presented.

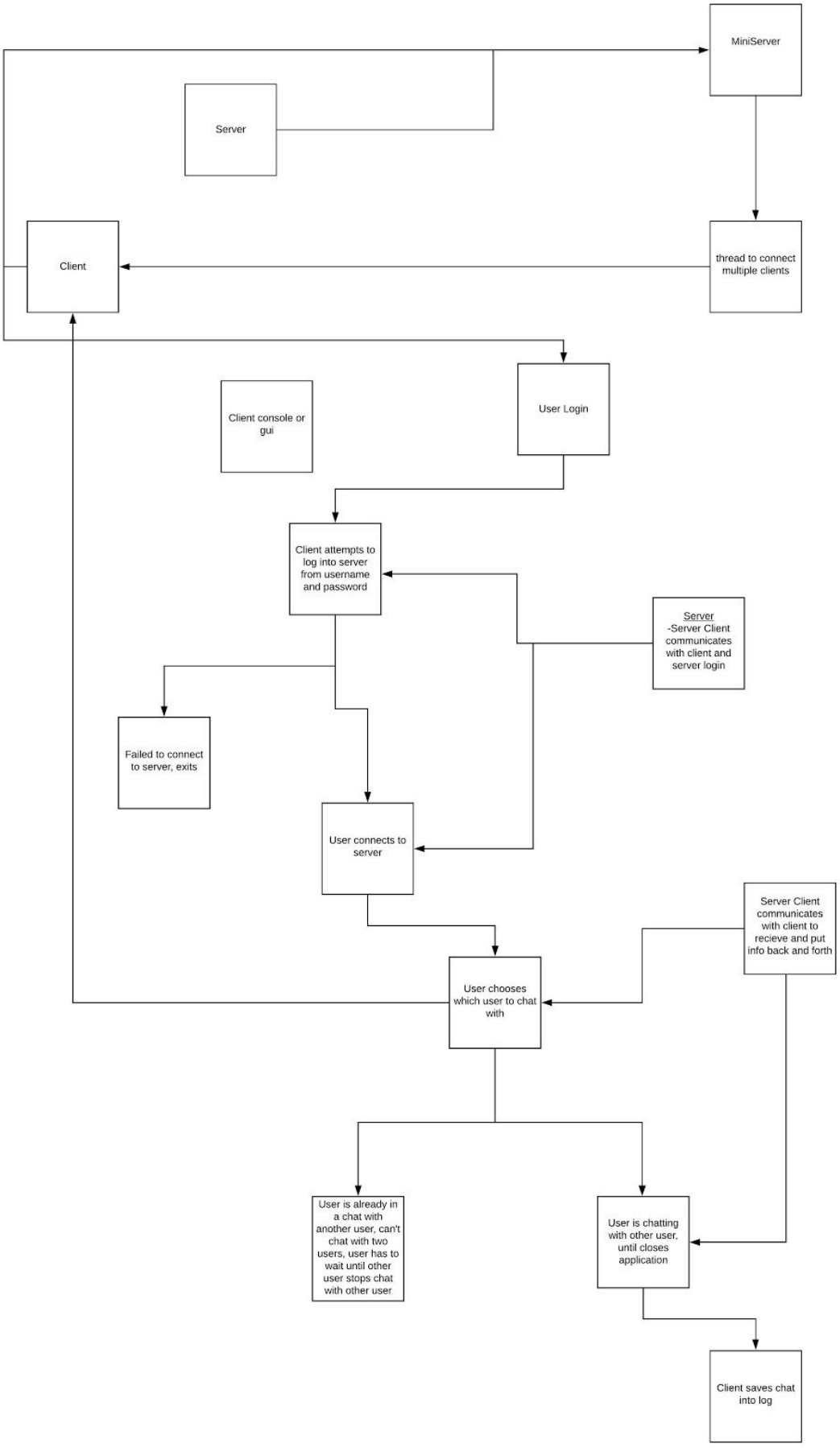
The GUI will send data to the client, the client will then pass the information to the server, and the server will respond to the client, then the client will then give information to the GUI to display. This will be an ongoing process while the chat GUI is running.

3.1 System Structure

A detailed description the system structure chosen for the application is presented.

There will be a main server and mini server classes so that the server can run efficiently. The main server will hold the port and a mini server will operate on the main server. There will be a client that connects to the mini server class that can pass back and forth information between server and client. The GUI will then connect to the client in order to give information back and forth.

3.1.1 Architecture diagram


MiniServer
Server
Client
thread to connect multiple clients
Client console or gui
User Login
Client attempts to log into server from username and password
Server
-Server Client communicates with client and server login
Failed to connect to server, exits
User connects to server
Server Client communicates with client to recieve and put info back and forth
User chooses which user to chat with
User is already in a chat with another user, can't chat with two users, user has to wait until other user stops chat with other user
User is chatting with other user, until closes application
Client saves chat into log

3.2 Description for Component n

Main Server connects to the server with a port. The MiniServer then connects to the server to operate on the server and also receives information from the client. Client then connects to MiniServer to send and receive information. The Client then connects to the GUI to send and receive information. This process continues until the program is done.

3.2.1 Processing narrative (PSPEC) for component and interface n

The program will be able to send and receive chats using a GUI after a user selects another user to chat with as long as that user does not have that user on a block list. A user will be able to make a user on their block list at any time. Once one of the users hits end chat, the connection will terminate between the two users and the chat session will end. Users will be able to send and receive messages whenever they like as long as the chat session is still in order.

3.2.3 Component n processing detail

MainServer connects to a port from an IP. MiniServer connects to mainserver using the port and IP from MainServer and then sends and receives information back and forth to the MainServer. MiniServer also sends and receives objects to the client. Client connects to MiniServer, and sends and receives objects. GUI connects to the client, sending and receiving commands and information. Message objects are used by clients to send and receive information.

3.2.3.1 Design Class hierarchy for component n

MiniServer uses MainServer objects, but uses thread. Client uses MiniServer objects, and objects revolving around data use in the application. Client uses GUI information to make commands to send and receive information to the MiniServer. Client uses Message objects to send and receive information to and from the server to each client.

3.2.3.2 Restrictions/limitations for component n

Users cannot send offline messages to other users.

3.2.3.3 Performance issues for component n

The more clients, the more data on the server, the slower server will be. Server does not have high capacity.

3.2.3.4 Design constraints for component n

Letting users send and receive messages whenever they want via GUI during the chat and not having to wait for a response in order to send a message. Cannot send messages to offline users.

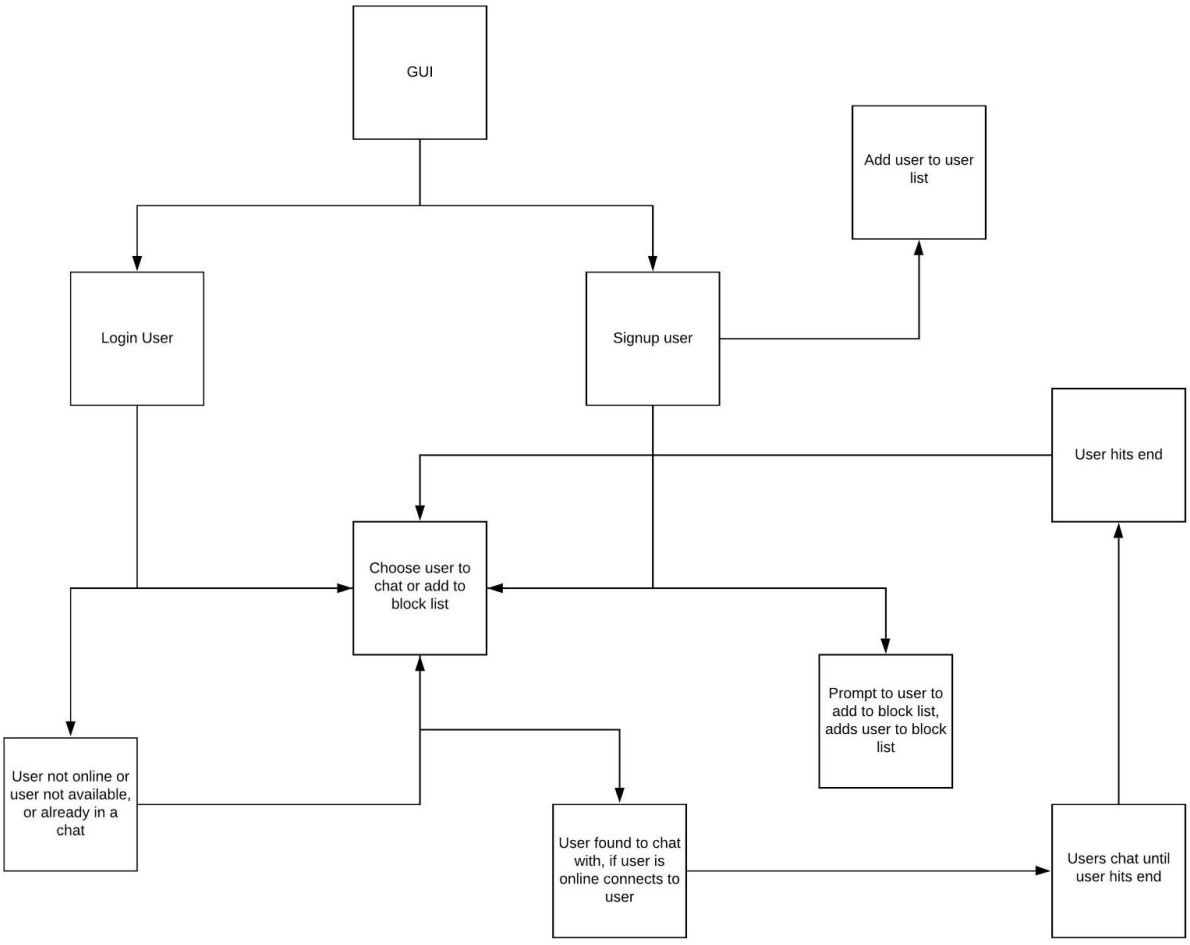
3.2.3.5.1 Processing narrative (PSPEC) for each operation

This program will mimic an aol chat instant messaging system, a user can sign up, login, then enter a user to chat with, if the user is a user and the user is online chat will start between the two and they will be able to send and receive messages to each other during the chat session. A user can enter a user to block at any time. A user can also end the chat, in which messages cannot be sent anymore between the two users.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

Message objects hold chat information like user, chat ID, messages, to be exchanged between chats. MiniServer will hold a list of users, list of blocked users, and will run off MainServer. MainServer just connects to the port using IP. Client exchanges message objects between clients and to and from MiniServer. GUI sends commands to Client to send message, receive message, block user, sign in, logout, end chat.

3.3.1 Interaction Diagrams



4.0 User interface design

GUI user has choice to login or sign up, after signing up user has choice to search for a user to chat with or block user, if user searched for another user and user is not blocked or is available for a chat, the chat will start between two users. Chat will continue until one user chooses to end the chat. Once chat ends, the user will go to the main menu where they can choose to search for a user to chat with or add a user to block list or logout.

4.1.2 Objects and actions

First user can sign up or login using buttons. After login is successful, the user can choose by button to search for a user to chat with or choose to enter the user to block. Once a user chooses someone to chat, there will be a box to enter chats to send, while received chats will appear in the GUI above the box to enter chats. Chat will keep going like this until the logout

button is pressed. Once the logout button is pressed in chat, chat will end for both users, and both users will be sent back to the menu to either logout, search for a user to chat with, or enter a user to block.

4.2 Interface design rules

When the client is launched, it will prompt for port number and IP address. Besides that, use of the GUI program will connect to the client sending and receiving information, then following the way the chat program was created and created via GUI.

4.3 Components available

There will be a popup box that will have a button to login into the application and a button to exit. Apon login the user will be prompted with buttons to search for a user and once that user is found the next phase will be to open a chat window similar to the structure of AOL chat. There will also be a button to exit the chat or to block the user.

4.4 UIDS description

GUI is created using swing in java. There is a large text box with all of the chats appearing, and there is a prompt for username and password to login, there is also a textbox for entering chats, and a button to send chats. All of these are connected to the client and are operations sent to the client.

5.0 Restrictions, limitations, and constraints

Users with usernames must be original because the way we store the usernames is with a hashmap and to avoid collisions the usernames must be original. There can not be a chat prompt with a user that has blocked one of the users. The application can only support a few users since the bandwidth of the server is not high.

6.0 Testing Issues

Testing the connection of the client to the socket and main server. We also need to test the message class to ensure that a message has been sent as an object. The main server needs to be tested to ensure a connection has been made and is not null.

6.1 Classes of tests

The program needs to be able to sign a user in and to be stored in an array of users. Usernames need to be unique and not duplicates. The messages need to be sent as an object from client to client. What we have tested so far is the creation of a server and mini server and assert that they are not null. The server needs to be connected.

6.2 Expected software response

The server creation should not be null. The clients who create messages should be an object of type message and not null. Users should be unique and no duplicates allowed.

6.3 Performance bounds

Currently the capacity of the users is limited and can not exceed 10 users. We have an arraylist of users who are currently logged in to the program. With this, the capacity of messaging drops to a minimum of 10 users.

6.4 Identification of critical components

The capacity of the server to handle so much traffic is something that is a critical issue. There needs to be a sign in before messages can be sent. Blocked users can not send messages after one user has blocked them.