**SE3313ChatGroup: Design Document**

**Work Distribution**

Front end developer: msweene4: 250 848 729

Back end developer: dhong45: 250856960

Project: Multi user chat room

**Overview**

* Server and Client code was written in java in order to support running on various devices and environments (as long as they were capable of running java, which was a safe assumption due to the widespread use of java), as well as to taking advantage of Java’s runnable interface.
* The server is hosted on an Amazon Web Service (currently offline but server files are hosted)
* The address of the hosted server is 54.167.215.132 and the port is 2012
* Server is hosting an instance of a chat room that allows an unlimited number of users to be in the same chat room instance
* Any number of users will connect to the server, inputs can happen simultaneously, and chat messages will update accordingly
* Individual users can exit the chat room at any time without interrupting the other users and the server will continue to work as expected
* Upon termination of the server, clients will be incapable of communicating with each other.

**Client**

Each client in the chat room has their own individual threaded interface that is controlled inside the server. The attributes and functionality of the interface are described in UIDesign. The project implements two java files for the client side the main java file(startChat.java) and the runnable UI (chatGroupUI.java). The startChat creates a chatGroupUI, sets it visible, and creates a thread to run an instance of the chatGrouUI. The chatGroupUI creates the GUI for the UI display, and implements the functionality of the UI attributes and their communication with the server.

**Server**

The server is a single file that hosts a specific instance of a chat room. The server is reliant on threads based on an implementation of a runnable called server (misnamed due to initial misconceptions of design). The main function of the server file stays within a continuous loop, listening for connections at the specified port, and when connected, the server creates a new thread running the server runnable. Each thread will be responsible for taking in the input from one connection, based on a BufferedWriter Vector, each thread running on the server would be able to send messages to each other connected client by using the collection of BufferedWriters, clients who leave the application has their thread rendered non-functional and interrupted (and terminated in the future), as well as having their reader and writer closed. This let to the use of a try catch statement when iterating through and sending messages to all known users. If a client is disconnected and their resources non-active, the error is caught and the server continues to function as intended. Based on messages from the client (these messages cannot be sent through the regular text sending methods) the server will know to interrupt/close resources handled by a specific connection, and to kill the entire server process. Many of these terminations are determined using boolean variables, to indicate whether or not specific threads(runnables) should interrupt and wait for the process to end. At the end, sockets are closed as well.

***Additional notes:***

-Threading: our assignment took advantage of Java’s runnable class, implementing our client associated threads in the server through a runnable. These threads have access to a shared Vector, as well as some boolean values in order to know how to send messages to all other connected clients, as well as to handle termination.

-Termination: Termination is specifically handled through the threads connected to each individual client. The server receives a message from the client that is normally unsendable by the client (other than by the specific button that is designed to terminate the server). At reception, boolean values are set, and the thread which received the termination code closes the socket listening for new connections as well as unblocking progress, allowing for the server to complete its functions and terminate.

**UIDesign**

**Labels**

There are three labels that are used to display titles.

**TextArea**

There is one textArea used for inputting the client message.

**jLists**

Messages- Displays the messages after receiving them from the server.

Member Name- Displays the name of the client.

**Buttons**

Send - will send the message from the jTextArea to the server and from the server to all of the client’s messages jList.

SetMember - Will set the Member name to the value in the textArea and change the name displayed in the Member name jList.

Delete Comment - User selects a comment shown in the messages jList, then clicking the Delete Comment button will remove that comment from the messages jList.

Reset Name- User can select the Member name in the jList and click the Reset name to remove the name from the jList and set the Member name back to Anonymous (default).

End Chat - When this button is clicked the server ends and communication between different client interfaces stops.

WindowClosing - The X button in the top right of interface window for closing windows. This program overrides the default function of this button. The new functionality is shown in the void myWindowClosing() in the chatGroupUI source code. The new functionality sends a message to the server indicating that the current thread of the client is ending then proceeds to close the window.

**Communication**

**Listeners**

actionperformed listeners on each button. When each button above is clicked, a certain action will be proceed.

**Sockets**

Sockets are set in both the server and client side of the app. They are used for accepting and receiving signals by utilizing BufferedReader and BufferedWriter objects which send and receive messages at those those specified sockets.

**Final Thoughts**

Our original idea for the final project was originally set to be a cooperative minesweeper app made in android studio where two people can enter a game and simultaneously complete the challenge. After further consideration of the time investment and the approach of final exams, full completion of this was deemed unrealistic. It would have been nice to have had time for a more complicated project but with the final projects for other courses happening within a week of final exams starting this was not the case.

With the chat room project, we feel it was more suited for the project as we could implement all the things that were required to even under our time constraints. The project was enjoyable to work on and was a good learning experience. This project made us more familiar with the concepts covered in the labs and their application together in a larger project.