Eric Su

Es34826

Prawal Sharma

Ps27933

Team Plan Project 7

Github url: https://github.com/EE422C-F17-HW3/422c-proj7-F17-422c-pr7-f17-pair10

On the first day, Eric and Prawal decided to review the concepts behind socket programming. After discussing what was needed, they came together to think of the basic architecture and the requirements that they needed to implement.

They decided that Eric would handle the client application while Prawal would handle the server application. The two would coordinate which features they would implement through messaging and also talk about communication protocol between the server and client to effectively communicate commands and actions. To make this communication uniform and easier, Eric created a new javafile called Commands.java that included some constant strings that can be shared commands between the client and server. This way there would be no mix up of the exact spelling and text of commands. Additionally Eric also created a java file called ports.java which contained information about the sockets used so that the sockets would be uniform and could be easily changed in one place. Eric and Prawal would continue to update the commands.java file as they added in new functionality. They would also cross reference each other’s code to ensure the commands were used to do the intended action across both ends.

Before starting, Eric made some UML diagrams to illustrate a general sense of what the finished product would do. Eric created a case diagram to include the possible things a client could do and a class diagram to show where functions needed to be to implement these actions.

When Eric began to make the client side application, he first started off by creating some of the socket code. He then built some of the logic and infrastructure for the operation of the client side application before making the GUI. As he implemented each feature, he communicated with Prawal about what he made and how the server should handle each command. He made the client side code by creating several different listeners that waited for events such as users entering text into the boxes and users pressing the buttons on the controller. Many of these listeners would then interface with the print writers that are connected to the socket to send commands and information to the server. Additionally he created a javafx timeline that would listen for commands sent by the server to perform certain actions. He communicated with Prawal about which commands the server would send and under which conditions these commands would be sent. He also created several data structures to store information about friends and group chats. Finally he also added in some functionality to play sounds within the application. In total, Eric spent around 19 hours working on the client side application.

Prawal worked on the server side application. He created 4 classes to implement the functionality of the Server program. These classes include: Char_R_Obj, Client_H, O_Print, and ServerMain. The Chat_R_Obj takes care of the Observable layout as it updates all of the observers on tasks performed. Client_H is the main logical piece of the server side, it contains the multi-threaded process that performs the specific commands that need to be executed from the client end. The O_Print class extends the printwriter class and overrides the update method in order to perform local updates. Finally, the ServerMain class, as it makes clear contains the main method in order to create the connections with the Server and client ports. It also contains the main Data Structures for Server data allocation.

To test the functionality of the project, Eric and Prawal connected to the UT wifi and began by testing one on one chat on one computer. They then proceeded to test on two computers and also group chat functionality.