COMP20081:

Systems Software.

Java program for a Music Social Network System.

By:

Ahmed Ahmed(N0570451)

Lewis Pressley (N0557446)

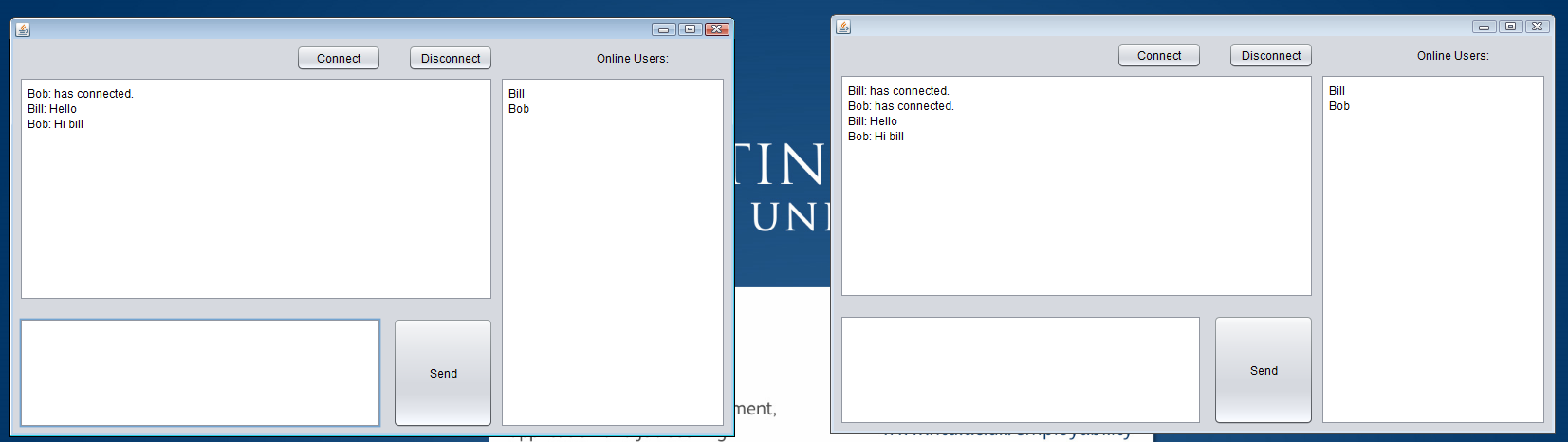
**Introduction:**

To start, in the following report about Graphical Social Network System, we as a pair were asked to work together to produce a centralised sever implementing a Spotify-like service. This contains client connection, registration, upload information, share music and finally to have a chat system server. During its period, we both have faced many battles with the lack of knowledge with java NetBeans. however, with the research skills as well as the determination to complete and know more we have managed to complete a fully working project which we can be proud of.

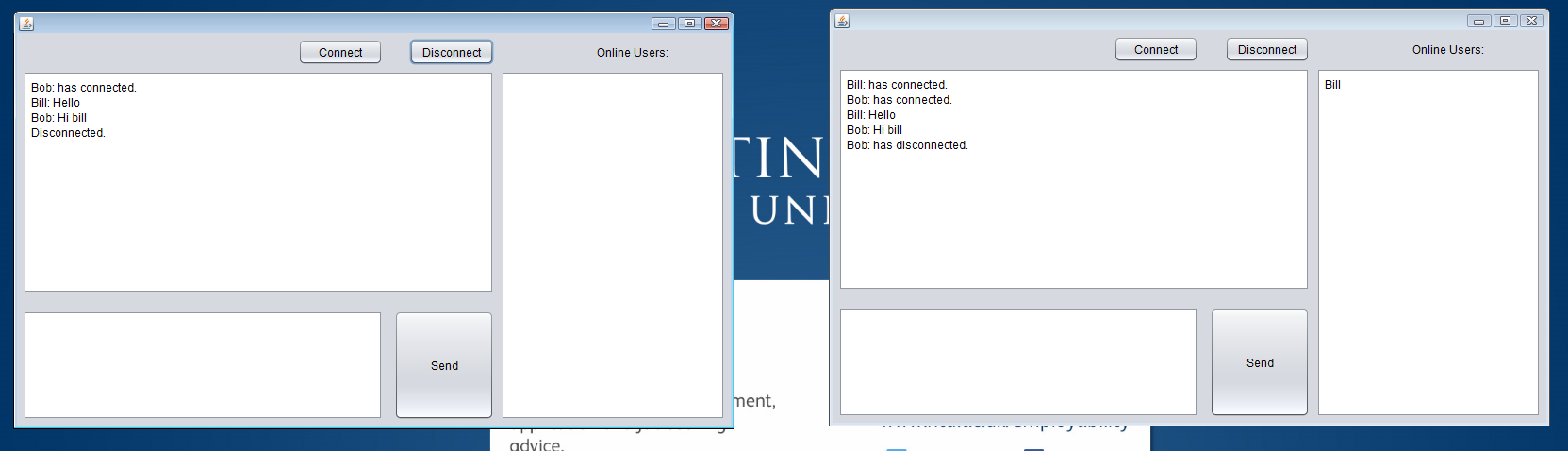
**Server Chat and features:**

First of all, we have created a Chat Server, this works by having a Host and Join button. If someone wants to Host then they click the Host button. This then activates a Server with their own private port number and opens a chat window. In the chat window they then click connect to connect to the chat and their username will be displayed in online users. The user can then send messages to the chat text area and any online users can see it and communicate. The host can then leave by disconnecting. If a user wants to join this chat they will need to select the friend from the friends list and then click join. This will open a chat window, which is exactly the same as the hosts chat window.

This shows the chat when they connect and communicate:



This shows when a user disconnects:



Secondly, now an in-depth look into the client side code. So first of all we have a void function run(). This function has a string array called data and strings with different keywords to differentiate what type of message it is. These are Connect, Disconnect, Done and Chat. Data is the split up by ‘:’ and this puts it into three different parts, if the final part (Data [2]) equals one of these commands e.g. chat, then it follows that IF statement. So chat appends the chat text area. Connect, empties the chat area from any previous chats and calls function userAdd, which adds the Data [0] (the username) to the userList text area. This allows all users online to be seen. The disconnect calls userRemove function and this takes the username and removes it from userList. Finally done sets user list to clear and calls the writeUsers function. We have two more functions called sendDisconnect and Disconnect and these ensure the user is disconnected and write it into chat so users know that user has disconnected. Finally, the client side has three different buttons.

**Server connect button:**

This checks to see if the person is connected, if they are it sends message to say already connected, otherwise it goes through these stages: Using a try catch it creates a socket using the serverIP and the hosts port. Then we have created a streamreader, which is an input stream reader, that gets the input stream from the socket. Then we have a reader which equals a new BufferedReader function taking the streamreader as its parameter and then we create a writer, which equals a new PrintWriter function that takes the output stream from the sock. Now the user is connected to the chat and we send a message to the chat saying that user has connected. We then flush the writer and set the connection to true, so that users don’t do this more than once.

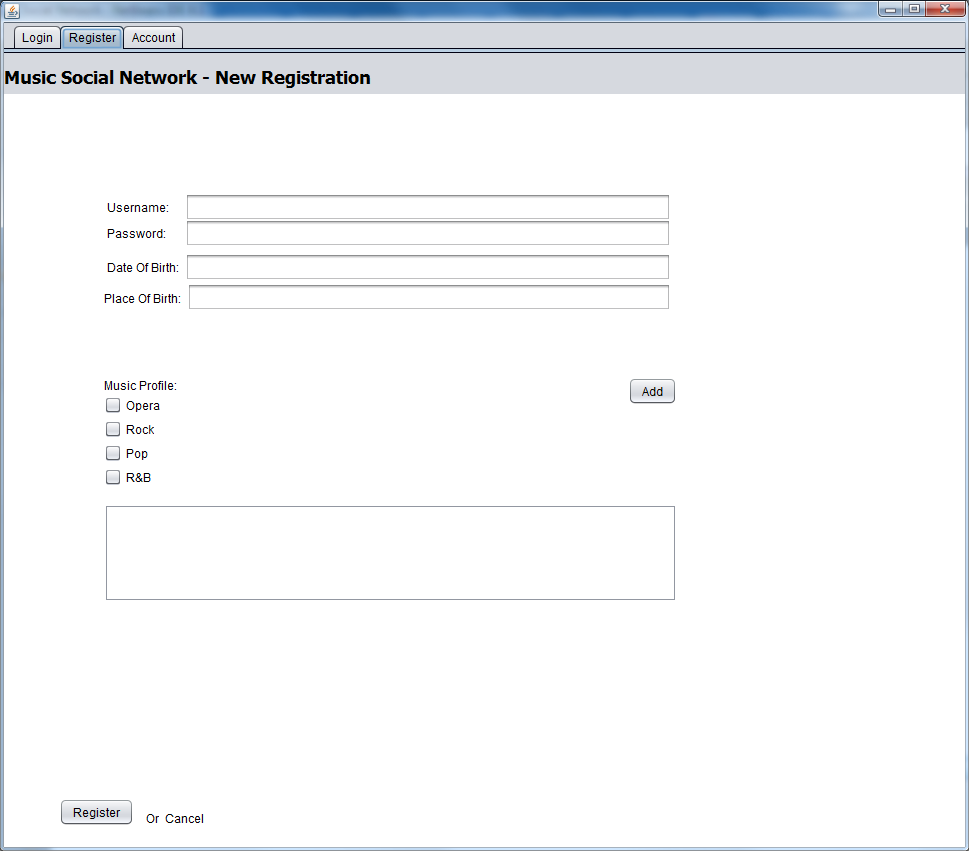
Another button is the send button. This first of creates a string called nothing, which equals “ ”. We can then do a check to see if the inputTextArea equals nothing. If it does, then no message is sent. If there are words then it will use the writer and print line the message username, inputTextArea and keyword Chat, so that it can identify what command is needed in run. The writer is then flushed. If this didn’t work, then the chatTextArea says message was not sent. The final button is Disconnect, which simply sendDisconnect and Disconnect functions.

Finally, with our server we start the server that will be always active for the social networking. This allows for adding friends, listing people online, showing who your friends are online and posting to friends. For the posting it is basically the same as chat but posts to all your friends. It also lists people online in pretty much the same manner as the chat, in that user’s names are displayed in the chat. Users can add each other as friends by typing their name from the people connected and pressing the friend request button. The user being accepted can then accept their request and add them as their friend and then send a request back to the other user so they can have that person their friends list.

**SQL:**

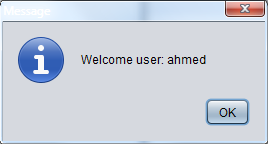
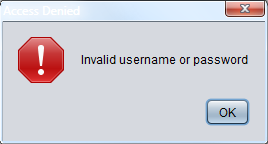
Within the client side, it allows users to register, login and use their account which holds all their correct re-inputted information. All this is done with the help of database (SQLite). One of the reasons why SQLite was picked, due to being our first choice as well as having the functionality to be save into our project.

When a new user starts the project they will need to register first. This is where SQLite comes in and stores it into an external database in a schema called systems which contains the table name of login. Within the register tab, there are many text fields as well as check boxes which enables items such as music profile to be picked. The music profile we are using a string inside a button which collects all the input from the user every time it’s been pressed. It subtracts if the user unticks an box too. All this is being happening within a ActionPerformed button.



After having inputted all the required information, there is a register button which is linked to all the text fields as well as the built functionality to INSERT into the SQLite database with all the correct data.

After registering the users can use their username and password to login into their account. For this we are using a text field with a password field which users can type their username and password. From this the project is using a String to call out or SELECT all of the data that is sorted in the database and look for the username and password associated with that user. After having found the user associated, what the SQLite will do, all the information will be transferred into the AccountGUI for the user page. It also uses SELECT statement to recall all the data. However, if the username and password is wrong due to being given a CATCH, will give errors and not allow information to be passed on. In the other hand, if it is correct it will welcome the user, with showMessageDialog.



This is putting in a validation to the client, which makes everything under control of the server as well as the client.

This is a diagram of how the SQLite works:

Logging in

Stores Information

Registering

Database

SQLite

User requests logging in

Checks details

Passing information back. If user exists or not.

User Logging in

**MP3 Player.**

Within out project, we have implemented an MP3 player. What this does, it allows users to Play songs, choice to play what they like and finally to stop playing. For this we have implemented functions as well as buttons which handles such tasks. To start of, we have included libraries which gives the functionality to do the following tasks above. This is by including JL1.0.jar. After a main java class has been created to hold all the functions of play, stop and pick a file. this contains also the FileInputStream and BufferedInputStream, as well as all the variables.

Firstly, Public void stop is designed to allow the stop button to stop playing the current playing song. Within this there are functions which enables to identify if a song is actually playing, so IF statements are being implemented. Within the IF statement there is a statement which statements is player != null, this means if the player is playing something then simply close and stop player, and from there I have called that function by MC.Stop();.

Secondly, we have used JFileChooser for selecting the music we ant to play once selected. Within the select function, we have includedFileFilter which allows you to filter all files. For this we have included mp3 within the perimeter. Below we have JFileChooser which enables users to start from any place to select, within the perimeter it conations the file path. Below we have applied the filter so we can get or apply the mp3 files. This is done by chooser.addChooseableFileFilter(filter);. For getting the return value we have showOpenDialog to get the return value. At the end we have an APPROVE\_OPTION. This is the open option, it will firstly get the selected file, by File myFile = chooser.getSelectedFile();. After will save it as a file and convert it into a string. This will after allow to implement the way to play the song/music via by play button.

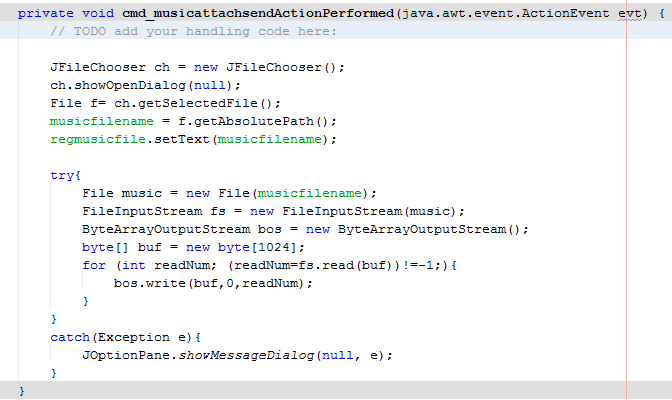
Secondly, another public void was created but this time for play(), but this contained string in the perimeter, as this was for the path of the music, which will be playing. This path contained the directory of the music when the select music has been selected. From there we have made the FileInputStream equal to new FileInputStream which contains the path inside the the brackets. At the same time setting the BufferedInputStream to the FileInputStream. To make the song play, we created a new Thread, this is because we want the music to play in the background. Furthermore, within an button for play which contains ActionPerformed this will allow music to play, by using MC.play().

**File Sharing:**

One of the other main features from our social network form is that it can allow sharing of music between friends. It works by a user selecting music using the attach button and then clicking the share button. This then starts a server with the users own personal port and posts a message in the post saying there sharing their song. Friends of this user can then select that users name and press the download button and they will receive the song.

Now if we dive into the code of this a bit more, we can look at the first button of attach music. This uses JFileChooser to open up a dialog so that user can select what file they want to upload and it sets the text field to that file path. Then using a try-catch we create a new file called music, which uses the music file name and then this gets all the file information.

Here is code for this button:



The next button share is basically what uploads the file to the server so it can be downloaded. It starts it as a Thread, then it sends a message to the Newsfeed that says they are sharing a song and the writer gets flushed. Then using a try-catch the share creates a socket and then confirms a connection. It then creates a new file using the file path from the text field. It then creates an array of bytes form the file length. Then it creates a new file input stream (FIS) and buffered input stream (BIS). Then it gets the output stream (OS) from the socket. It then writes to the byte array to the output stream and then that is flushed. Finally, it closes the BIS, OS and sock. It also has created another stream which runs the main program.

Here is main code for this button:



The final button is Download. This uses and SQL statement. The statement checks the username selected in friends list is in the database and then finds the portid number, which is the port of the person sharing. It then uses this port number to create a socket to access the file on the server. It creates a new file output stream (FOS) and new buffered output stream (BOS). It reads the byte array and bytesread equals that and then current equals that. Using BOS it writes my byte array and current. Then BOS is flushed. Finally, everything is closed and a pop up message says the file is downloaded.

Here is the code for this button:





**Conclusion:**

In conclusion, overall we believe we have demonstrated good system software skills. We have put in so much effort into research as well as self learning (how servers work). This is something which neither of us had ever done before and our first exposure to Java, but we worked very hard and managed to produce a very good quality piece of coursework. This has been demonstrated, we also have a very good knowledge of SQL and use it to good effect to create a good database to hold our clients. The layout of the Jframe is easy to use and has some complicated features.

Finally, if we were to do the coursework again, we would try to react fast as we needed bit more time. However, putting that to one side, we both have gain huge experience from working with systems and software by producing top demanding quality piece of work.

**References:**

cgermany77 (12/05/2012) *Java - Sockets - Coding a Client-Server CHAT Room - 2 of 3,* Available at: *https://www.youtube.com/watch?v=Uo5DY546rKY&index=78&list=PL28D4CB13F443964B* (Accessed: 05/04/2016).

(18/09/2014) *Transfer a file via Socket,* Available at: *http://www.rgagnon.com/javadetails/java-0542.html* (Accessed: 03/04/2016).

ProgrammingKnowledge (25/03/2012) *Whole java netbeans Project (jtable , jdatepicker ,itext , Login , Signout ,iReport and more...),* Available at: *https://www.youtube.com/watch?v=gp-rmc78YCo&list=PLB04B4E5D9B58C13D* (Accessed: 20/03/2016).

varietytubechan (21/04/2011) *How to create a Simple Chat Client w/ GUI in JAVA #1,* Available at: *https://www.youtube.com/watch?v=qWYn1omeqqs&list=PLCBF515A460A08185&index=1* (Accessed: 10/03/2016).

ProgrammingKnowledge (18/09/2012) *Java prog#1. Create netbeans project and download SQLite Manager from firefox (web browser),* Available at: *https://www.youtube.com/watch?v=v-JM749op-8&index=3&list=PLB04B4E5D9B58C13D* (Accessed: 03/03/2016).