Comp20081: Systems Software:

The people presenting this particular implementation:

-James Earle, N0719214

-Samuel Brandenburg, N0707417

**Below contains all the functionality we have included for the different aspects of this project.**

Functionality included for the client:

-The client is able to make a connection to the server.

-The client is able to register as a valid user, sending all necessary data that will eventually be stored. This data will include the name of the user and their profile (age, music preferences, etc.).

-The user will also be able to log in, if the valid username and password is sent.

-User is then able to be removed from the server through a log off button which they are able to click.

-Users can send a message to the server for selecting the desired person who they want to request a friendship with.

-Client can also receive notifications letting them know that their friend has accepted their request (added to friend box).

-Client is also able to view an up-to date list of all the users they are currently friends with.

-Client can see all relevant information about different friends when they are clicked. This is displayed to the user, letting them see anything that may be of interest (friends age for example).

-Client will receive and show posts from friends. This will be displayed to their screen and they will be able to view only posts from users that they are friends with (also will be in chronological order).

-User will be able to view all songs that friends have shared. When the user clicks on their friend, that users shared songs will be displayed.

-When play button is pressed on a selected song, the song will play.

\*\*-send or receive text from other users\*\*

Functionality of the social network server:

-The server will keep a record of all members that are currently online. This is read from a text file when loaded, so that it can remember even when not running.

-Server will also keep the profiles of these online members. These profiles will store who this user is friends with and any requests that they have received.

-Server is able to accept new member registrations and add them to active participant list. This will store the user profile with all the necessary information explained above. This new user is then written to the text file that stores users.

-Can receive new requests to be friends, which is then stored in the received requests for the relevant user. This will also be written to a text file, so that the requests will be saved even if the user logs off without handling request.

-Will be able to receive new friendships and add to the relevant user profiles, will also be written to a text file for storage so that user friends is remembered when logging back on.

-In addition to this, all the songs that the users have shared will be stored on the server. This will include all information necessary regarding each upload (song name, Artist, etc.).

-New music received will be added to music of the user that uploaded. Will also be written to a file, so that the music can be stored when program is not running.

-List of a friend’s music is sent to user when their name is clicked, along with any information needed about that user.

-Will be able to accept requests from users to log off and then that user will be removed from records.

-lists all current users and sends them to a new client.

-receives post messages and sends the relevant post information to the corresponding users.

-The social network server is capable of handling multiple member connections concurrently.

Chat server Functionality:

-Receives a notification that a chat request has been sent from client.

-Searches for each user and sets a notification for the user requesting a chat and the user receiving one. This information is then passed from the server to the client where they will be notified resulting in a chat box to appear.

**Below will provide an explanation of how our system has been designed.**

The nature of the network connection used

The server is multithreaded where it creates a new connection each time a request has been sent to the client. We chose to pass an object over the port to give the server and client more flexibility on how to send and respond to a request. This was performed by creating a serializable class named Messages. The class ObjectoutputStream and ObjectInputStream was used to pass this class from one end to the other. Inside the server class, it has a list of different if statements it can respond to. For user generated response on the client end a connection is opened if this action needs to pass information to the server and closed after the request has been completed. There is a separate thread in the GUI class that creates a connection to the server that doesn’t close. This thread is responsible for asking the server to send the client any updates it has receive. This allows the clients to be constantly up to date on who is online and when a user signs off. This continuous connection also is responsible for sending notifications to the client; such as, sending and receiving friend request; it also updates the GUI to show a friend connection if the user decides to accept the request, and when a client has made a post on his or her timeline.

The data structure used by the server to record active participant information

As a user turns on the application the user is given the option to log in where their information is stored in a file. One a user has logged on the server stores the information in a userProfile class. There is then an ArrayList created which stores all of the different user profile objects (one for each user).

The UserProfile class also has an ArrayList which stores all Muiscinfo classes (this stores all data needed when a song is uploaded). As a result of this, a song a user has uploaded can be found from a user class.

A similar structure has been used with the user’s friends and the requests that they receive.