HITIFY

SYSTEMS SOFTWARE COURSEWORK

EDWARD CELELLA (N0687265)
SAM MILWARD (N0690641)

NOTTINGHAM TRENT UNIVERSITY | 2018

Contents

Implemented Features **2**

Client *2*

Music Server *2*

Chat Server *2*

Overview of System **3**

Client *3*

GUI 3

Sending and Receiving Data 3

Music Player 3

Chat Server *4*

Sending and Receiving Data 4

Storing Chat History 4

Multi-Threading 4

Music Server *4*

Sending and Receiving Data 4

Multi-Threading 4

Database 5

System Overview Diagram **5**

Implemented Features

Client

- Uses swing library to provide a GUI.
- Can send log in and log off requests to music server.
- Can receive user information including active friends and friend's posts from music server.
- Can receive search information when adding new friends.
- Can upload songs to server and download previously uploaded songs by user and friends.
- Can send messages to friends
- Can send multimedia files to friends
- Music files can be played.
- Music files can be paused.
- Music files when played can be tracked by progress bar.

Music Server

- Receive new account information and add to user database (including music preferences, profile picture and user details).
- Receive login request and check details against database.
- Can access database and send active friends list to client.
- Can access database and send all friends of user to client.
- Can access database and send to client all user friends posts.
- Can send and receive new friend requests and rejections.
- Can remove friends from users table in database.
- Can remove user from active users table in database when they have logged off.
- Can retrieve friends uploaded songs and information from the database.
- Can retrieve users own uploaded songs from the database and send to the client.
- Can search database for users with certain music preferences.

Chat Server

- Receive request for new chat between users.
- Store previous chat history between users.
- Can receive text messages and relay to other user.
- Can receive and type of sent files and send them to user.
- Can store messages and files if one user isn't currently online and send them to user when they're next online.
- Multi-threaded so multiple users can connect and talk at the same time

Overview of System

Client

GUI

The client uses the swing library in order to provide a GUI for the user. The look and feel of the GUI is mainly altered by changing the properties of each item. However, for more advanced customization (e.g. the scroll bar and combo boxes) the basic UI provided had to be extended using classes which returned the basic UI with the required changes. Each screen is a separate class which is disposed to free up resources when switched. The only exception to this is the chat screen which can have multiple windows open, with the main screen.

Sending and Receiving Data

Music Server

Object output and input streams are used when connecting to the music server. This method involves the use of a serializable object in order to be sent. The class 'InfoPacket' is used for this purpose. Whenever data needs to be sent an object is created, the functions which best apply to the type of data being sent are called to set the data. This object is then sent to the server. The server also uses the class 'InfoPacket' to send and receive data. A new connection is made either on an event, or after a set amount of time. There are two threads on the client side, which refresh the data every two seconds, and the other every sixty seconds. Connections are also made when a user interacts with the GUI, such as uploading a song, or sending a friend request.

Chat Server

Data input and output streams are used to send and receive data with the chat server. If the message being sent just contains text then the '.writeUTF()' and '.readUTF()' functions are used to send and receive the data. These messages are then appended to the local chat log so in future the chat history can be loaded locally instead of all being sent. When a file is being sent or received the functions '.write()' and '.read()' are used in order to send a byte array (which the file has been converted to). When receiving a file, it is sent to the downloads folder. A thread is created which continuously listens for new messages. The connection to the server is not continuous.

Music Player

When a song is selected is downloaded from the music server. This mp3 file is then converted to a wav file using an external javazoom library. Then using the sun.audio library an audio stream is created which is used to play and pause the file. As well as this a new timer is created in order to increment the progress bar. To ensure the progress bar ends when the song ends, the duration of the song is calculated using the formula:
Runtime = File_Size / (Sample Rate * Channels * Bits per sample /8)

Chat Server

Sending and Receiving Data

As previously discussed in the client section data input and output streams are used. With the functions '.write()', '.writeUTF()', '.read()', and '.readUTF()' used based on if a string or byte array is being sent or received.

Storing Chat History

When a message is sent by a user, the message is appended to the end of that specifics chats text file. If a file is sent then a message saying a file has been sent is appended to the bottom of the text file and the file stored in the files folder to be sent. Messages are stored in the text file using the following format: "Marker§Username§Message§FileSize§FileName". The marker is either 'T' or 'F' to indicate if a file needs to be sent or not. This icon '§' is used to split the strings components.

Multi-Threading

When a client connects a new thread is created to handle the client. This thread creates a second thread. Two threads are required so data can both be sent and received. The receiving thread will append messages to the chat text file and save sent files. The second thread continuously checks if the text file has been modified and if it has it will read the last line in the file and send the message and if required the file to the user.

Music Server

Sending and Receiving Data

As previously discussed in the client section object output and input streams are used when communicating with the server. The class InfoPacket is used to store the data. It implements serializable so that the object is converted to a byte array so it can be sent on an object input/output stream. This object is sent to the client or received via the object output/input stream. The InfoPacket object contains a variable called 'Service'. This string contains a three-letter code which tells the server which operation needs to be performed (e.g. 'LGN' would run the login code).

Multi-Threading

The music server is multithreaded, allowing the server to handle multiple clients and request simultaneously on one port. A ServerSocket is initialised with a port number. The server has an infinite loop, so that it constantly waits for a client. Within this loop, a new instance of the class MusicServerExtended is initialised. This is the class that handles the client's requests. This class has a private attribute called client, of the type Socket. This Socket is initialized by the SetSocket() method of the class, called in the MusicServer loop. A new thread is then started and passed the class MusicServerExtended which extends Thread. This thread is then started, and the loop begins again, listening for another client.

Database

An instance of my Database class is made within the music extended thread. This class is used throughout the server to access a SQLite database stored in the local area of the music server. This class contains functions that run SQL Queries to insert, delete or manipulate data. Most functions are passed parameters to insert into the database, or values to search a table for.

The database consists of five tables:

Table Name	Columns within the table
ActiveMembers	Username, IpAddress (Of current online member)
Friends	FirstUsername, SecondUsername, Status (Accepted,Declined,Pending)
Posts	Username, DateTimePosted, TypeOfPost, MessageOrFilename
Songs	Filename, Artist, SongName, Genre
UserTable	Username, Password, FirstName, SecondName, Email, MusicPreferences

System Overview Diagram

Below is a basic diagram showing the relationship between classes/objects when the program runs.