

SPL-1 Project Report

A Simple Software for Audio Editing & Searching

Course: Software Project Lab I

Course No: SE 305

Submitted by

Md. Aquib Azmain

BSSE Roll No. : 0718

Submitted to

Rezvi Shahariar

Assistant Professor, IIT, DU

Amit Seal Ami

Lecturer, IIT, DU

Project supervisor:

Sheikh Muhammad Sarwar

Assistant Professor, IIT, DU



Institute of Information Technology

University of Dhaka

30-04-2016

Table of Contents

| | |
|-------------------------------------|----|
| 1. Introduction..... | 1 |
| 1.1. Broad Domain | 1 |
| 1.2. Challenges | 2 |
| 1.3. Requirements..... | 2 |
| 1.3.1. Deliverables | 2 |
| 1.3.2. Cost | 3 |
| 1.3.3. Software | 3 |
| 1.3.4. Hardware | 4 |
| 1.4. Methodology | 4 |
| 1.5. Achievements..... | 5 |
| 1.6. Conclusion | 6 |
| 2. Background Study | 6 |
| 3. Analysis and Design..... | 9 |
| 4. Implementation and Testing | 13 |
| 4.1. Implementation | 13 |
| 4.2. Testing | 14 |
| 5. Program Output | 15 |
| 6. User Manual..... | 17 |
| 7. Conclusion..... | 22 |
| 8. Appendix | 23 |

Index of Tables

| | |
|-----------------------------------|----|
| Table 1: Disk Space for JRE | 3 |
| Table 2: ID3 format..... | 7 |
| Table 3: Code for MergeMP3..... | 13 |
| Table 4: LCS..... | 23 |

Index of Figures

| | |
|--|----|
| Figure 1: Methodology | 4 |
| Figure 2: Intro | 9 |
| Figure 3: Merge | 10 |
| Figure 4: Trim | 10 |
| Figure 5: Search via Metadata | 11 |
| Figure 6: Search via Bytes | 11 |
| Figure 7: Class Diagram..... | 12 |
| Figure 8: Output for metadata searching..... | 15 |
| Figure 9: Output for searching via bytes | 16 |
| Figure 10: Start up Interface | 17 |
| Figure 11: Sub Modules for editing | 18 |
| Figure 12: Sub Module for searching..... | 18 |
| Figure 13: Merge Panel | 19 |
| Figure 14: File Browser | 19 |
| Figure 15: Merge Panel | 20 |
| Figure 16: Trim Panel | 20 |
| Figure 17: Search via Metadata..... | 21 |

Acknowledgement

At first I would like to thank almighty God for helping me accomplish my goals.

I would like to express our deepest gratitude to all those who provided us the possibility to complete this project. A special thanks to my supervisor of Software Project Lab-1, Sheikh Muhammad Sarwar, who inspired me to do this project and gave necessary lessons to do this project.

I would also like to thank our respected teachers Rezvi Shahariar and Amit Seal Ami for giving us invaluable guidelines.

I am grateful to the Institute of Information Technology for giving me the opportunity to do a project of such caliber.

Lastly I would like to thank all my classmates for assisting me and providing valuable insights throughout the project.

Sincerely,

Md. Aquib Azmain

1. Introduction

This section explains the motivation for this project, by considering the role of the audio and the impact of editing on a music file - flow and practice. This application is able to recognize a short audio sample of music that had been broadcast and can also do some editing such as merge and trim. This section presents a statement of requirements for the software product.

1.1. Broad Domain

I have started my project with thinking over a scenario like "You heard a familiar song in the club or the restaurant and the sentimentality of the song really touched your heart. You desperately want to hear it tomorrow, but you can't remember its title!" So my plan was record the part of a song, fingerprint it and run a search to match the part with the songs of database and find the original song. Then I develop a project in java that can search music and also it can edit music.

It is an application which does music editing and searching through:

- Raw file reading
- Implementation of algorithm (LCS).
- Providing the user with a very handy user interface.

This application can do some editing on a music file like:

- Trim a music file: Cut a music file from a specific point to given duration and export the part as a different music file.
- Merge two music files

This application can also search music from a folder. It can search by two different ways like:

- Search by metadata
- Search by finding the Longest Common Subsequence (LCS)

1.2. Challenges

Implementing a new software solution carries with it a number of challenges. The process can be overwhelming, confusing and lengthy—all reasons that can cause companies to avoid making the switch at all. Knowing the common challenges implementation projects present can help organizations avoid them. With a solid project plan and realistic goals, even the most complex implementation can realize success and return on investment in a reasonable amount of time. In the end, the benefits gained from implementing a better solution far outweigh the potential hazards along the way. The main challenges of my project are:

- Length of time: The average length of a typical software implementation process varies widely. For a robust, integrated solution the average implementation can last from 11 to 18 months. Here I have only 4 months.
- Implementing Algorithms: To conduct my project I have learnt some new algorithms like finding longest common subsequence.
- To conduct my project I had to work with different formats of music files. Then I had to write bytes in a text file and match them. The files are sometimes very large in size. So sometimes I had to face heap space out of memory error.

1.3. Requirements

The requirements of project can be divided into some points like Deliverables, Software and hardware requirements, costs etc.

1.3.1. Deliverables

- Software executable.
- Source code.
- User manual.
- Documentation.

1.3.2. Cost

The project has to be completed without any monetary expenses. All software to be used are free-to-use and are downloadable on the internet. Open source and free software are mostly used to ascertain that pirated software is not used for the project.

1.3.3. Software

This package will be for desktop PCs, which runs Windows operating system and Linux operating system. All variants of Windows(Vista, 7, 8,10) and Linux (Mint, Ubuntu) versions are compatible. Operating system may be 32 bit or 64 bit. The operating environment must have Java Development Kit installed.

There are some constraints found implementing the software. They are pointed out below:

- Run-time is a bit high
- Only handle .mp3 and .WAV files.

The executable JAR (Java Archive File) of the software must be executed before the software is used. The software searches for the required environment (Java Runtime Environment). If it is found, the file is executed automatically. Otherwise, the executable file will not compile properly. The software will not recover the previous information if it encounters a crash while running. But the rate of encountering a crash while running is ignorable. Thus no measures have been taken against this.

Table 1: Disk Space for JRE [2]

| Disk Space Requirements for JRE | |
|--|------------------------|
| JRE | Installed Image |
| Java Runtime Environment, including JavaFX Runtime | 124 MB |
| Java Update | 2 MB |

Security Requirement:

This software will not rely on any personal data of the user. So, security related to the user's privacy will not be a concern here.

1.3.4. Hardware

The only hardware interface requirement is a personal computer. As it is just a desktop software, no internet is needed. The configuration of the software is given below:

- Processor: Minimum Pentium 2 266 MHz processor
- Disk space: 124 MB for JRE; 2 MB for Java Update
- 128MB RAM or better

1.4. Methodology

I have followed waterfall approach to do my project. The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) My methodology to do this project is given below:

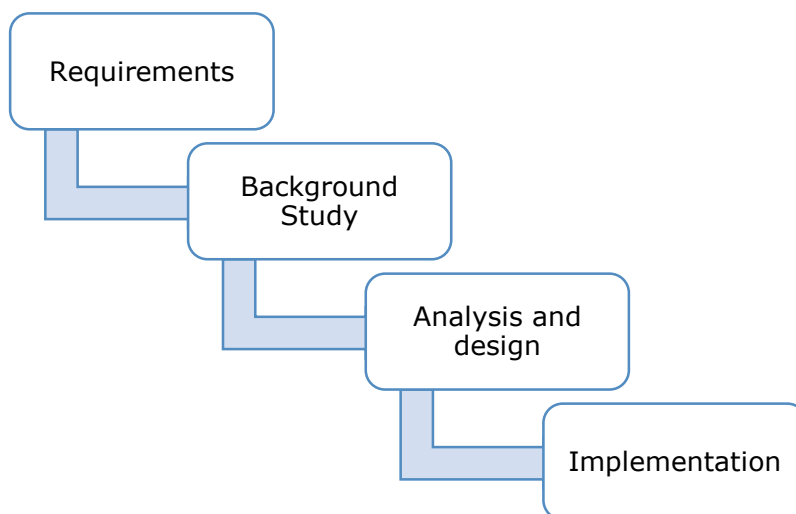


Figure 1: Methodology

Implementation Technologies:

I have used Java to implement the algorithms. There are many speed issues associated with Java which make it a less popular choice for high performance programs. However, recent releases of the API have shown great improvements, so much so that some newsgroups claim it is as fast as, if not faster than C++ for many applications. Regardless of the accuracy of this claim, Java is a well-established programming language which I have tested using Sun's own Java Sound Demo and found to perform without any problems. Java has a dedicated sound API which provides methods for the input of audio data through a computer's ports.

Project Resource:

The main resource of the project is the hardware and software requirements. I am working on this project directly with my supervisor to guide me. Reference materials, guides, instructions are found in plenty on the internet. Free version of the Java IDE Eclipse is used and the Operating System is the Open Source Linux Ubuntu and Microsoft Windows as well.

1.5. Achievements

The software has many quality attributes. It is very much user friendly and flexible. Some attributes are defined here:

- Performance: It is ensured that it provides maximum performance through least system resource usage.
- Security: As it does not require personal information of the user, the security and privacy is secured.
- Usability: I have ensured to make the software as more usable as possible.
- Platform independency: This software is platform independent which means that the performance of the software will not vary in different operating systems. This is a huge advantage for common people and this kind of quality increases the usage of any software.

Having a fully functioning and tested section of code that represents a conceptual milestone ensures that this risk of errors is reduced. The baseline points are ordered such that the highest risks are tackled first. The application may be divided into two main risk areas – audio processing and audio searching. The ordered baseline points are based on the success of,

- Audio input
- Extract metadata
- Merge two .mp3 files
- Trim a part from a music file
- Search files to match finding longest common subsequence (LCS)
- Implementation of a GUI
- Export of application into an executable file

1.6. Conclusion

This project is all about music files. Editing music files is a real life problem. Sometimes we need a part of a music file to use it as a ringtone of mobile phone. Sometimes we record our own music in different slots. We need to merge them. My program is able to solve this problem. Moreover, my program is able to find the original song getting a very small part of that song. It also helps us to get some important information (duration, author name, artist name) about any music file.


2. Background Study

I would have study on particular topic to complete my project. I would have to read through the books to learn the algorithm of longest common subsequence. Moreover, I would have to know about different formats of audio files.

- *MP3 file:*

MP3 (MPEG-1 Audio Layer-3) is a standard technology and format for compressing a sound sequence into a very small file (about one-twelfth the size of the original file) while preserving the original level of sound quality when it is played. MP3 provides near CD quality audio. It is one of the most common music file types. It is not an MPEG 3 but uses the audio compression found in layer III in MPEG 1 or 2 video files, the audio stream layer. Sound quality varies by such settings as bit rate (fixed or variable),

sample rate, joint or normal stereo. Mp3 has a proposed replacement in MP3pro with

better sound for a given file size. 

- *WAV file:*

Waveform Audio File Format (WAVE, or more commonly known as WAV due to its filename extension) (rarely, Audio for Windows) is a Microsoft and IBM audio file format standard for storing an audio bitstream on PCs. It is an application of the Resource Interchange File Format (RIFF) bitstream format method for storing data in "chunks", and thus is also close to the 8SVX and the AIFF format used on Amiga and Macintosh computers, respectively. It is the main format used on Windows systems for raw and typically uncompressed audio. The usual bitstream encoding is the linear pulse-code

modulation (LPCM) format. 

- *Metadata:*

Metadata is data that describes other data. Meta is a prefix that in most information technology usages means "an underlying definition or description. Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified and file size are examples of very basic document metadata. Having the ability to filter through that metadata makes it much easier for someone to locate a specific document. The metadata of an audio file is held in the ID3 format. The Information is stored in the last 128 bytes of an MP3. [1]

Table 2: ID3 Format

| Field | Length | Offsets |
|------------|--------|---------|
| Tag | 3 | 0-2 |
| Song title | 30 | 3-32 |
| Artist | 30 | 33-62 |
| Album | 30 | 63-92 |
| Year | 4 | 93-96 |
| Comment | 30 | 97-126 |
| Genre | 1 | 127 |

- *Longest common subsequence:*

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. Longest common subsequence (LCS) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences. Given two sequence of integers, $A=[a_1,a_2,\dots,a_n]$ and $B=[b_1,b_2,\dots,b_m]$, find any one longest common subsequence. In case multiple solutions exist, print any of them. It is guaranteed that at least one non-empty common subsequence will exist. [3]

- *Audio Data Structure:*

The main input to the system is analogue (or possibly digital if an optical or S/PDIF line in were used) audio data from the computer's soundcard. This information must be read and the results passed. There are many concepts and considerations involved in this process.

- *Digital Audio Format:*

Once the audio signal has travelled through the soundcard it is represented in digital form. Although made up of a series of bits, it is organised into a particular audio format definition. Briefly, there are two major factors in a digital audio format:

- Sampling Rate, measured in Hz, expresses the number of samples recorded per second. The maximum frequency which can be represented is double the sampling rate

- Sample Size, indicates how many bits are used to store each sample's amplitude. This determines both the maximum dynamic range and the signal to noise ratio. The audio format chosen for the application effects both the speed and results; the higher the quality of format used, the more accurate the results, at the cost of execution speed. As the application is predominantly concerned with the frequency content of the input, sampling rate is kept at the CD quality of 44100Hz. Initially a sample size of 16 bits was used, but extra implementation is required for handling the data. As sample size bears its greatest impact on the quality of sound, a factor that does not affect this application, 8 bits was seen as sufficient to express the amplitude of the signal.

3. Analysis and Design

I have divided my project into two different modules. Each module has two different submodules. So I have four submodules in total. They are:

- Merge
- Trim
- Search via metadata
- Search via bytes

Total 8 classes are used to achieve the results required. Below they are described as simply as I can:

❖ Class 'Intro': This class is for start-up interface. It connects the modules. It has two module like:

- Edit
- Search

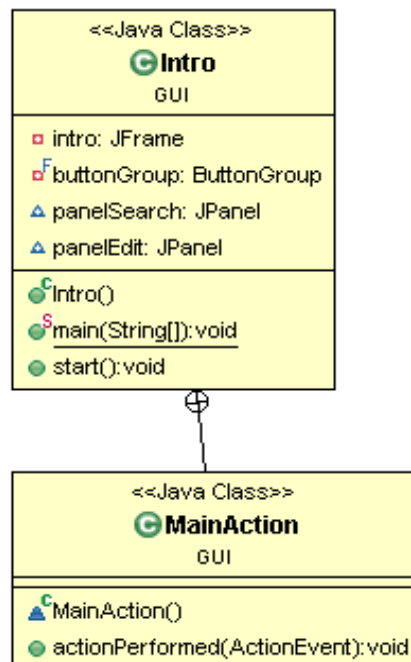


Figure 2: Intro

- ❖ Class 'Merge': This class has the methods to merge two different files.

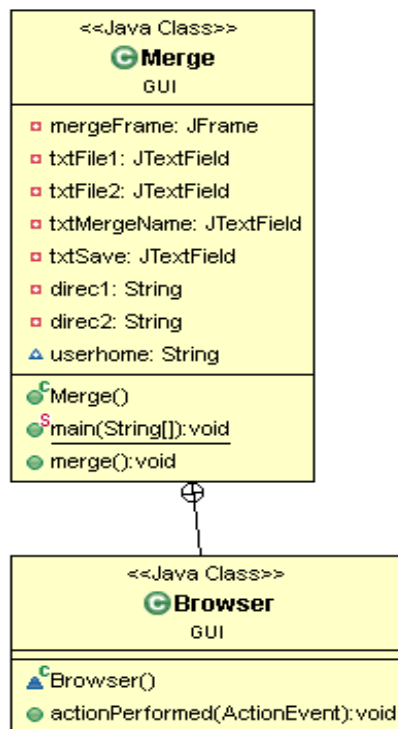


Figure 3: Merge

- ❖ Class 'Trim': This class have the method to trim a part of an mp3. It converts the mp3 file into wav. Then run the algorithm of trimming.

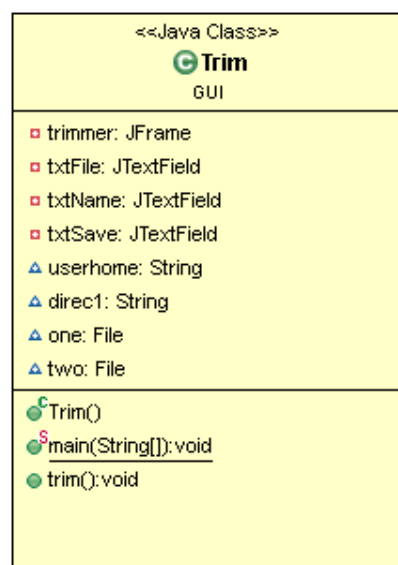


Figure 4: Trim

- ❖ Class 'GetMetadata' and Class 'SearchMeta': These classes are interconnected. The SearchMeta class is fully dependent on 'GetMetadata'. Class 'GetMetadata' extracts the information of an mp3 then passes them through a function. 'SearchMeta' class uses the information to search the original file.

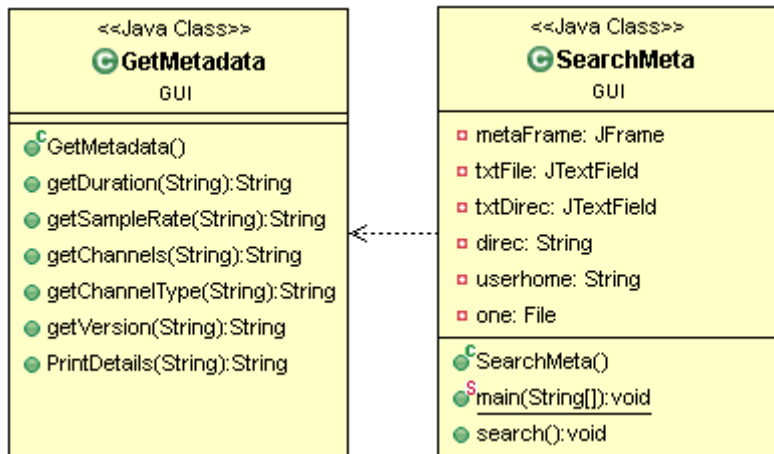


Figure 5: Search via Metadata

- ❖ Class 'LCS', Class 'ReadWAV' and Class 'SearchByte': These three classes are also interconnected for the purpose of searching via reading bytes. 'ReadWAV' reads a wav file and writes the bytes as characters into a text file. Class 'LCS' finds the value of longest common subsequence of the text files.

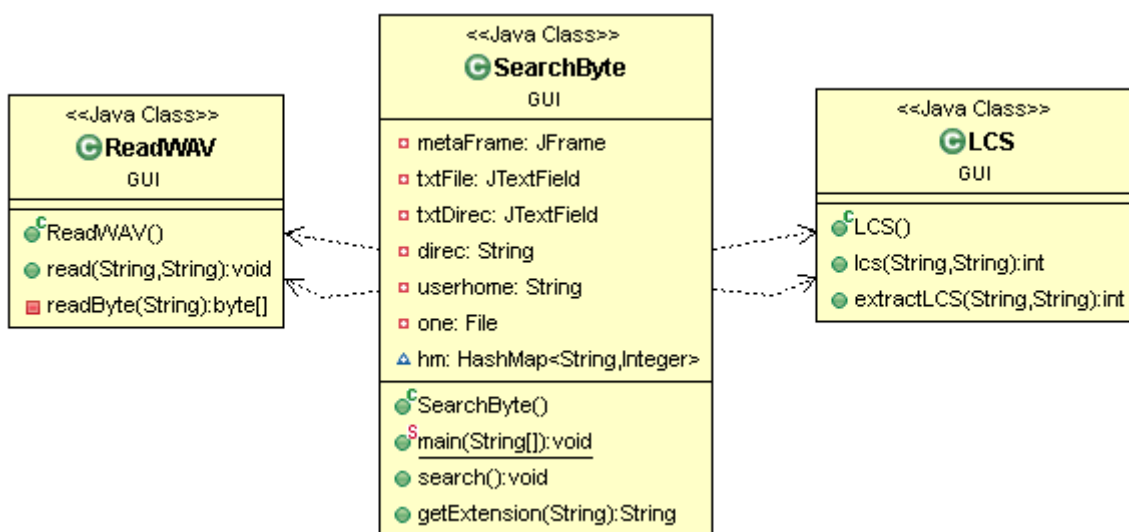


Figure 6: Search via Bytes

- ❖ All the eight classes are interrelated to each other. The connections are shown in the class diagrams given below:

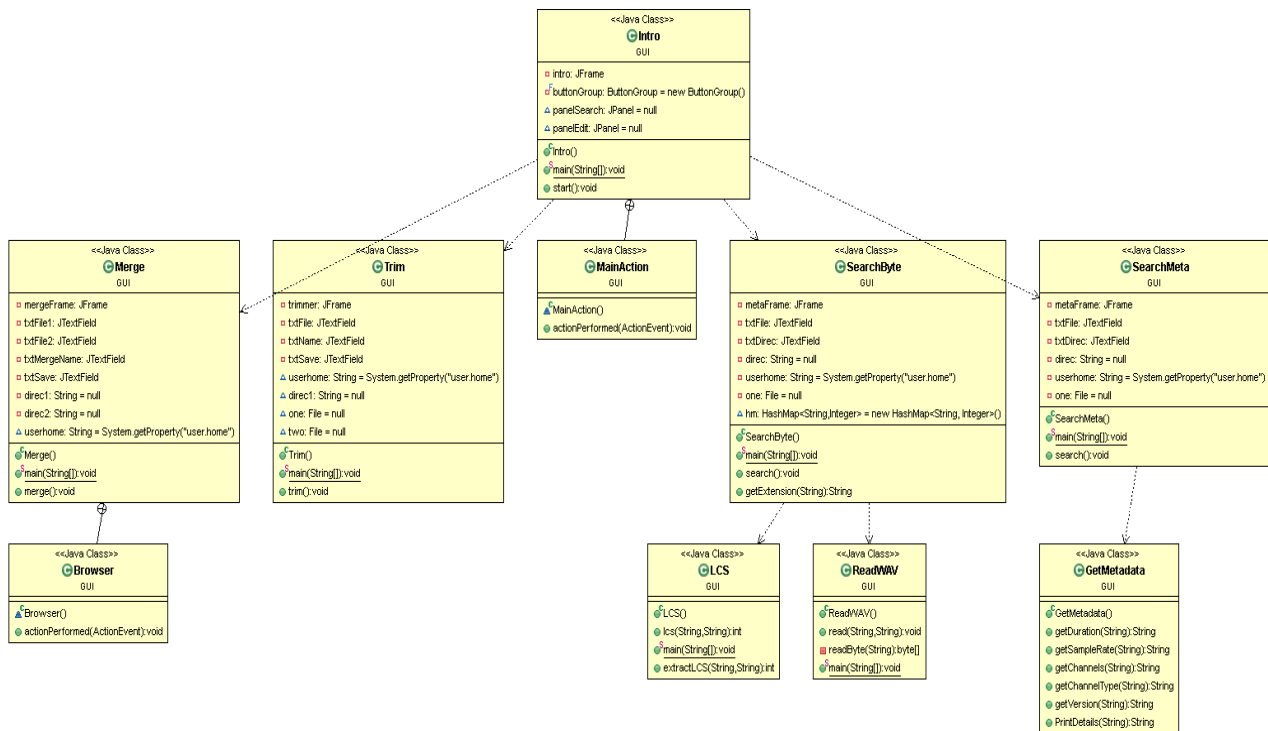


Figure 7: Class Diagram

4. Implementation and Testing

I have implemented the whole project divided into four parts such as merge, trim, search by metadata and search by reading bytes.

4.1. Implementation

- Search via metadata: I have used The Apache Tika™ toolkit in this class.[5] The Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). I have used the methods of this library and made my own method to extract some data of the music file like:
 - Duration
 - Audio Sample Rate
 - Channels
 - Audio Channel Type
 - Version

I have different methods to get those information from a music file. Then I have created a folder of 10 music files to use as my database. I take an input in class "MetadataMatching" and extract the metadata and then match it with all of the files in my database.

- Merge: In this class I have opened two BufferedInputStream and then read them one after another and write them in another file. The code is like this:

Table 2: Code for MergeMP3

```
BufferedInputStream br1 = new BufferedInputStream(in1);
    BufferedInputStream br2 = new
BufferedInputStream(in2);
    BufferedOutputStream bw = new
BufferedOutputStream(out);

    int i;
    while ((i = br1.read()) != -1) {
        bw.write(i);
        // System.out.println(s);
    }
    while ((i = br2.read()) != -1) {
        bw.write(i);
    }
}
```

- Search via bytes: I have implemented the longest common subsequence algorithm using dynamic programming. The code of LCS is included in appendix. In Class "ReadWAV" I can read a WAV file and convert it to byte Array in another text file. The text file contains some characters. So I have to read all files and convert them to byte file (.txt) .Then I have used the methods of LCS and got the most LCS which is my result.
- Trim: This module only takes mp3 files as input. It converts the mp3 files into wav file using an external library 'xuggle-xuggler' .Then the file can be trimmed. Java sound API gives us the value of audio frame rate and audio frame size. Using these information I have used a formula to get bytes per second of the file. The formula is: bytes Per Second = Frame Size * Frame Rate. Then I can get the time in seconds using bytes per seconds. Then I have opened a new audioInputStream which holds the part of the audio.

4.2. Testing

Various methods of testing could be performed through the entire project. But I have used only one type of testing to ensure the software functioned without error and that it met the required specifications.

User Testing: With all of the requirements satisfied and a usable piece of software complete, a small number of times I have completed thoroughly test application and gathered experiences with it. My program was slow when I had used brute force method to match the bytes of two files. Then I have used LCS.

I have faced some errors also. The program can't handle a large file while searching via bytes. "Heap space out of memory error" has occurred several times. So this is a limitation of my program.

5. Program Output

This program has several modules. One of the modules is search by metadata. After matching the files on the basis of their metadata we can get such output:



Figure 8: Output for metadata searching

If we use search by Longest Common Subsequence (LCS) algorithm the output shows some number for each songs. This number shows the LCS of that file after matching the given file. The audio file with most LCS is our result.

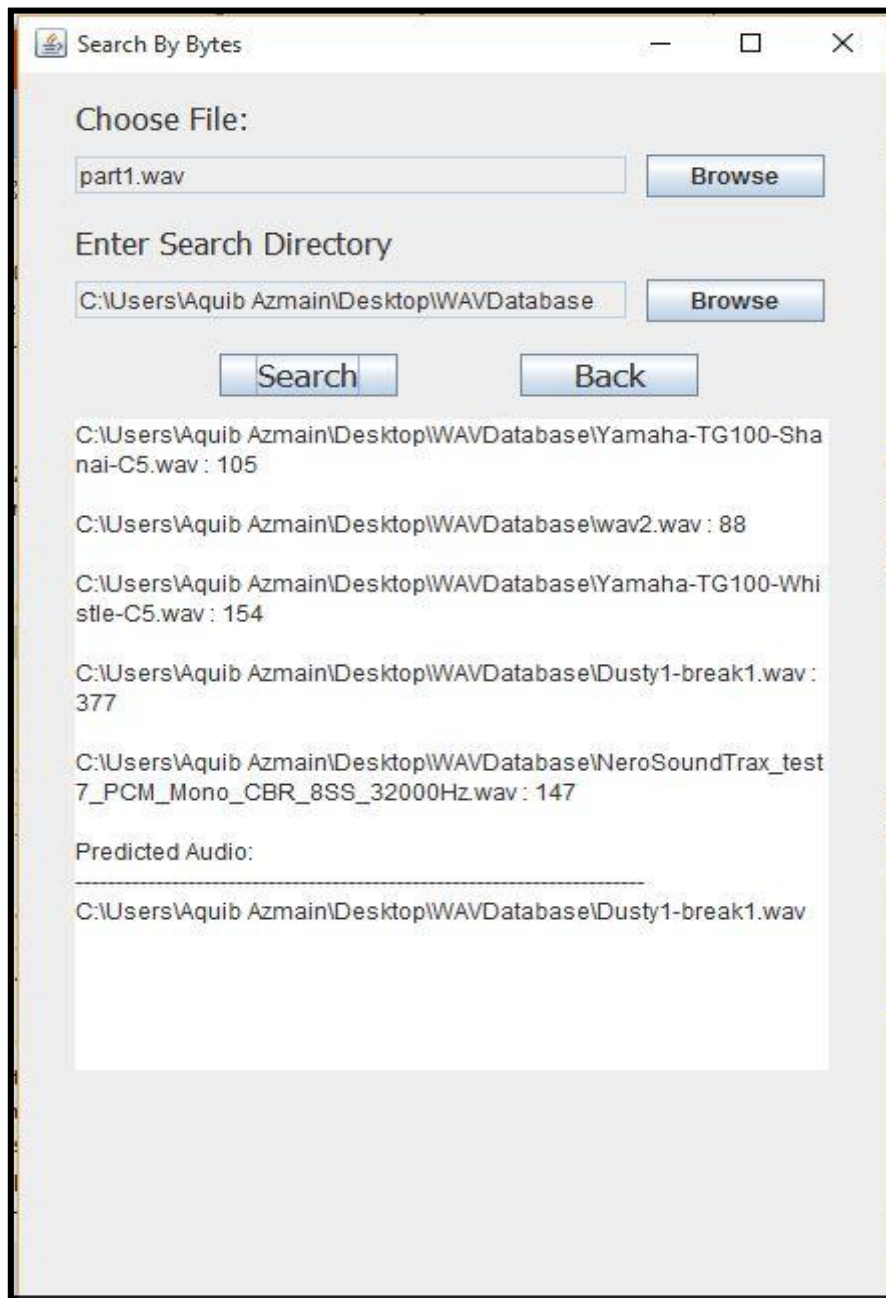


Figure 9: Output for searching via bytes

The module of trim and merge file can export a playable .mp3 file. This output cannot be shown on a report because the output is a music file.

6. User Manual

❖ Install JRE in windows:

- Windows x86 Online: jre-8version-windows-i586-iftw.exe (The letters iftw mean "install from the web.")
- Windows x86 Offline: jre-8version-windows-i586.exe
- Windows x64: jre-8version-windows-x64.exe
- The public JRE installed with the JDK is not registered. (This also applies to the 64-bit version of the JDK.) You must set the PATH environment variable to point to JAVA_HOME\bin (where JAVA_HOME is the location where you installed the public JRE) to register the JRE. See "Private Versus Public JRE" for more information about the public JRE.
- After installation, use the Java item in the Windows Start menu to get access to essential Java information and functions, including help, the Java Control Panel, and checking for updates.

❖ Open the Executable Jar File (.jar)

❖ Then we can see the Start Up Interface

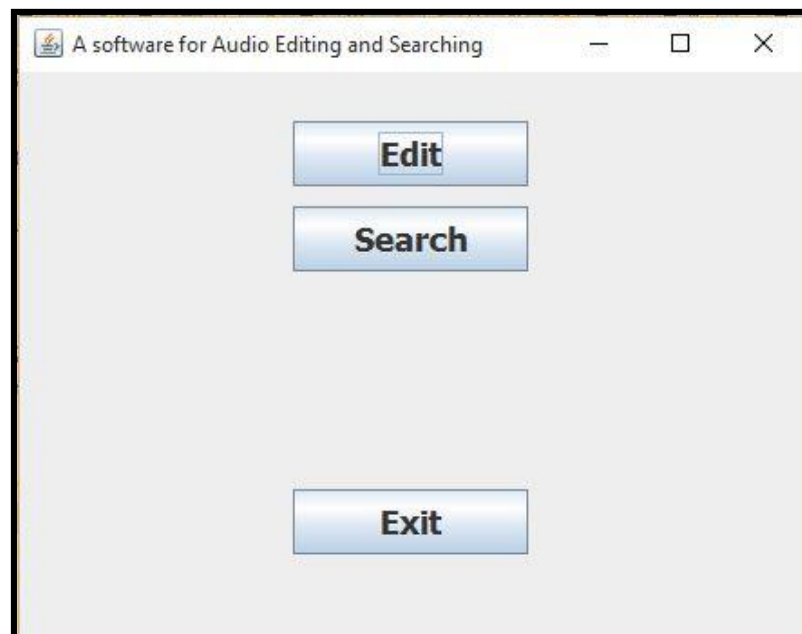


Figure 10: Start up Interface

- ❖ Then select a module (Edit or Search) and the sub modules will be appeared.

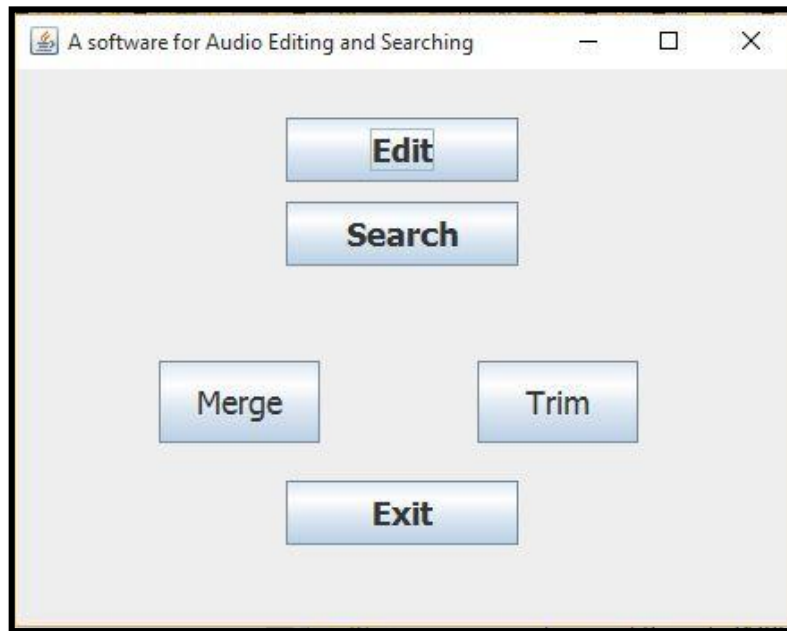


Figure 11: Sub Modules for editing

- ❖ If we select 'Search' the sub modules will be like this :



Figure 12: Sub Modules for searching

- ❖ If we want to merge then click merge and this panel will be opened:

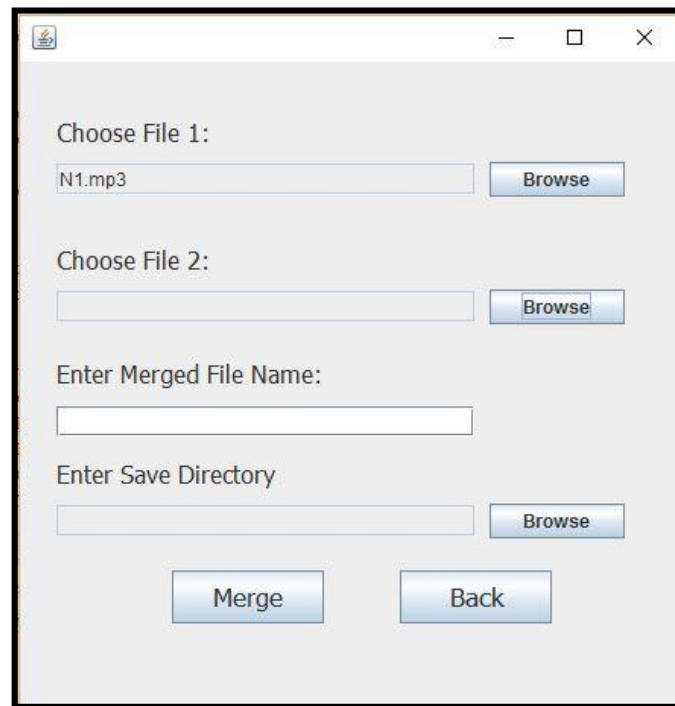


Figure 13: Merge Panel

- ❖ We have to select two files from the pc and also enter the name of the merged file and its directory.

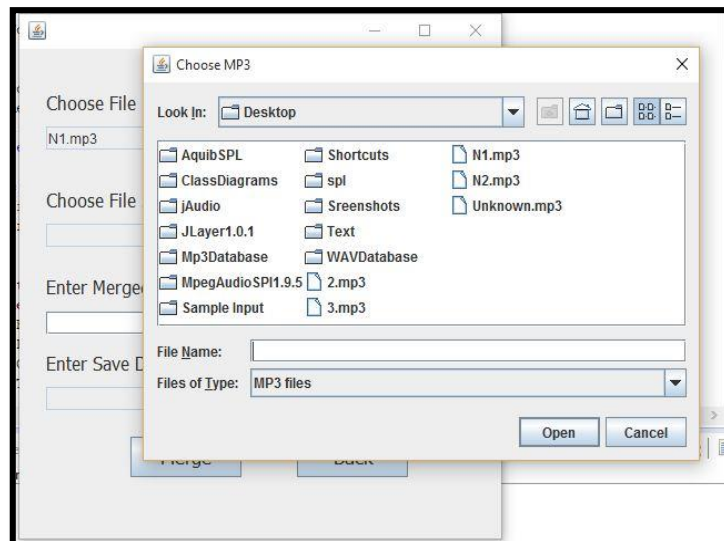


Figure 14: File Browser

- ❖ After selecting the files and directory hit "Merge". A new file will be created in the particular directory.

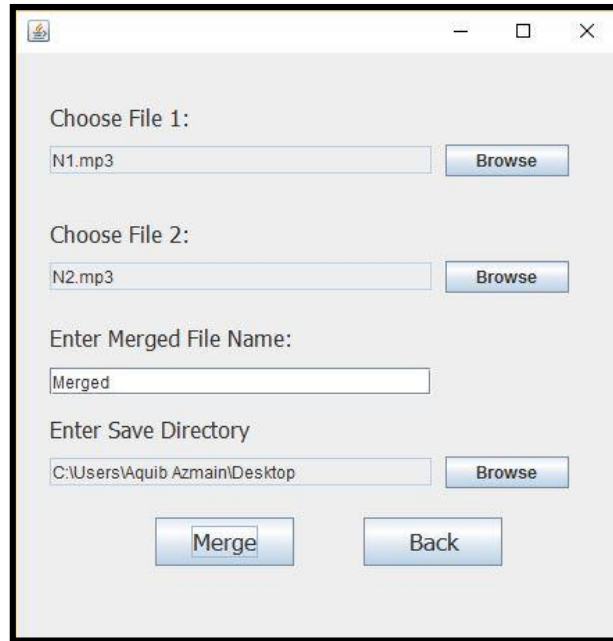


Figure 15: Merge Panel

- ❖ If we hit "Trim" we get trim panel. We have to select start second and the duration of the desired part of the file. This part will be exported as a new file in a defined directory.

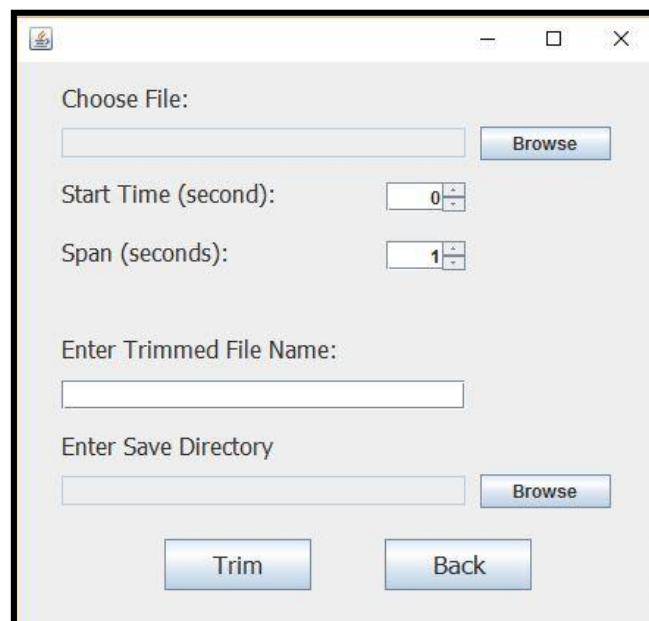


Figure 16: Trim Panel

- ❖ In search module we should choose “Using Metadata” if we have a whole .mp3 file. It will find the original song and also show the metadata of that file. We have select the target file and also select the source folder.

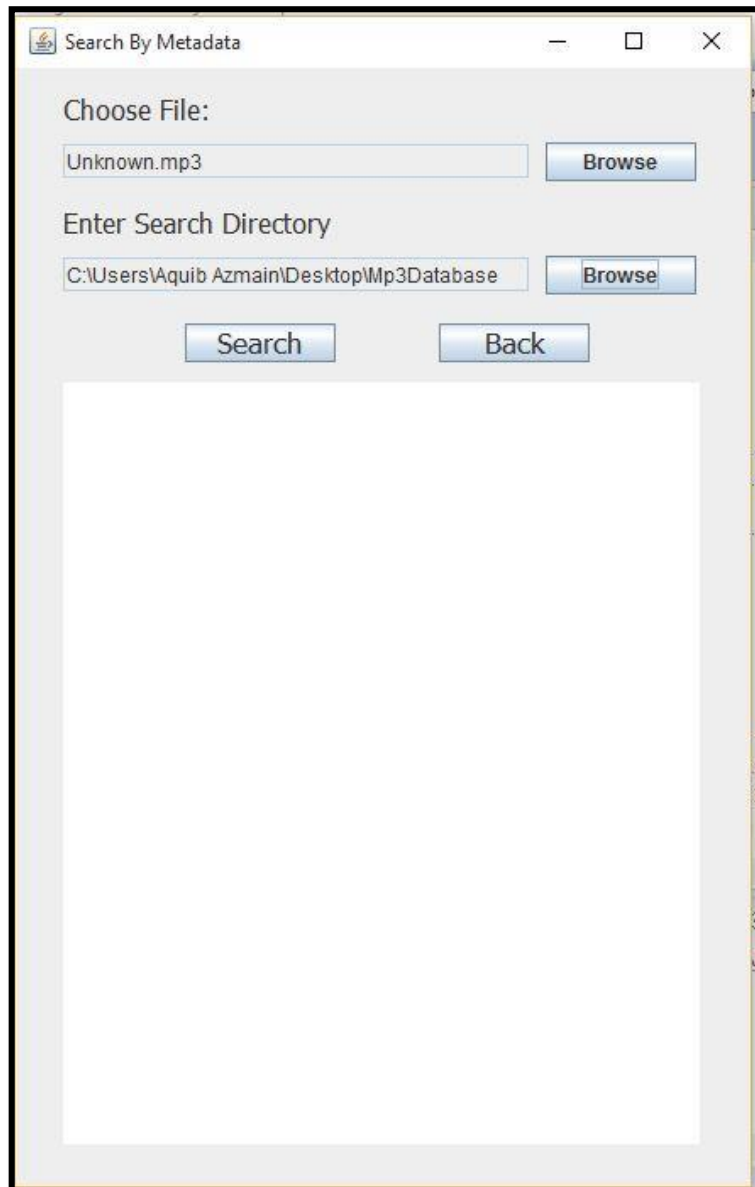


Figure 17: Search via Metadata

- ❖ If we have a part of a wav file we should choose “Using Byte Data”. It will find the matched file. This module is not applicable for mp3 files. The outputs are shown in figure-7 and figure-8.

7. Conclusion

With all requirements, aims and objectives satisfied I consider this project to be a success. As previously mentioned, there are a few minor aspects of the assignment I would approach differently in hindsight. However, I don't believe any issues raised have been a major detriment to the overall success of the project.

This project is all about music files. Editing music files is a real life problem. Sometimes we need a part of a music file to use it as a ringtone of mobile phone. Sometimes we record our own music in different slots. We need to merge them. My program is able to solve this problem. Moreover, my program is able to find the original song getting a very small part of that song. It also helps us to get some important information (duration, author name, artist name) about any music file.

The work carried out on this project opens itself to a number of future uses. Of course, many of the classes and structures in the software were designed with code reuse in mind. The pattern matcher and audio generator are of particular interest, as they could lend themselves to a host of other uses. I myself fully intend to continue work on the software when the assignment comes to end. I would like to remove the hard coded synthesis engine and instead implement MIDI functionality, an industry standard protocol allowing electronic instruments and computers to communicate. This would allow the pattern matcher to drive external hardware such as synthesizers, drum machine and samplers, allowing the focus of development to shift to the artificial life aspect of the system.

8. Appendix

Table 3: LCS

```
char * A;
char * B;
array L;

int lcs_length(char * AA, char * BB)
{
    A = AA; B = BB;
    allocate storage for L;
    for (i = 0; i <= m; i++)
        for (j = 0; j <= m; j++)
            L[i,j] = -1;

    return subproblem(0, 0);
}

int subproblem(int i, int j)
{
    if (L[i,j] < 0) {
        if (A[i] == '\0' || B[j] == '\0') L[i,j] = 0;
        else if (A[i] == B[j]) L[i,j] = 1 + subproblem(i+1, j+1);
        else L[i,j] = max(subproblem(i+1, j), subproblem(i, j+1));
    }
    return L[i,j];
}
```

- **Sample Rate:** The originally proposed sample rate of 48000Hz was reduced to 22200Hz, this resulted in a noticeable boost in performance. Due to the relatively simple nature of the tones being generated there were little to negative changes in their perceived quality.
- **Bit Rate:** Likewise the bit rate, the originally 16bit, was dropped to 8bit without noticeable detriment to the audio quality.

Reference

- [1] <http://whatis.techtarget.com/definition/metadata>, Metadata, last accessed on 05.02.2016
- [2] <https://docs.oracle.com/javase/7/docs/webnotes/install/windows/windows-system-requirements.htm>, Oracle, last accessed on 10.02.2016
- [3] Introduction to Algorithms (Third Edition) by Thomas H. Cormen Charles, E. Leiserson Ronald and L. Rivest Clifford Stein.
- [4] Audiovisual Systems Project Report, Corbett Technology Solutions Inc. (CTSI), Published on 05.03.2013
- [5] <https://tika.apache.org/>, Java Library, last accessed on 07.03.2016