MEDIA & MP3 PLAYER

A Project-I Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

AAYUSH AGRAWAL EN18CS301003
ABHINAV JOSHI EN18CS301005
AKSHAT JAIN EN18CS301018

Under the Guidance of

Mr. Lakhan Singh

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331
AUG-DEC 2021

Report Approval

The project work **"MEDIA & MP3 PLAYER"** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the "Project Report" only for the purpose for which it has been submitted.

Internal Examiner
Name: Mr. Gaurav Sharma
Assistant Professor
Computer Science Engineering
Medi-caps University, Indore

External Examiner
Name:
Designation
Affiliation

Declaration

I/We hereby declare that the project entitled **"MEDIA & MP3 PLAYER"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications in Computer Science & Engineering completed under the supervision of **Mr. Gaurav Sharma, Computer Science,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Aayush Agrawal
Abhinav Joshi
Akshat Jain

Date:23-11-2021

Certificate

I, **Mr. Gaurav Sharma** certify that the project entitled **"MEDIA & MP3 PLAYER"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications by **Akshat Jain, Abhinav Joshi, Aayush Agrawal** is the record carried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

_____

Mr. Lakhan Singh Sir

Computer Science & Engineering

Medi-Caps University, Indore

_____

Dr. Pramod Nair

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) D. K. Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) Suresh Jain,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod Nair** for his continuous encouragement for betterment of the project.

It is their help and support, due to which we became able to complete the design and technical report

Without their support this report would not have been possible.

Aayush Agrawal
Abhinav Joshi
Akshat Jain

B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

Abstract

This project is about the mp3 music player application development using JavaFX. The biggest difference between the music player and existing applications is that it is completely free for users to use. It will integrate the advantages of existing music players on the market, as far as possible to mining out the existing music players' function, and then do the filtering in order to eliminate function that not practical or low cost-effective.

The agile development cycle consists of six phases, which is requirements analysis, planning, design, implementation or development, testing, and deployment. Due to the iterative and flexible nature of this approach, it is able to effectively adapt to users with changing requirements.

Table of Contents

List of Figures

List of Tables

Abbreviations

Apps – Applications

iOS – Iphone Operating System

MP3 – MPEG 1 Audio Layer 3

MP4 – MPEG Layer 4

XML – Extensible Markup Language

GPU – Graphics Processing Unit

Chapter 1

1.1 Introduction

Media player have grabbed huge attention over a past decade and attracted majority of computer users thus making the users addicted to videos and music. Over the last decade, human computer interaction has become an active research area, which releases people from inactive, inflexible communication with machine .

Maximum computer users switch on to the media player as soon as they start the computers and then move to their respective woks. Also, many have a habit of dragging the songs into the list pane of media player, tune into music and then work. So we know how much are the media player used and required . Interacting with computers intelligently makes a significant contribution to the future application of human computer interaction .

Today's era is to do work with high efficiency but at the same time it should consume very less time. And thus answer to the problems arising in the use of traditional media players and the problems persisting in media players. In this paper we aim to develop a Media and MP3 Player which will resolve the problems of user regarding audio and videos. This player gives numerous facilities which differentiate it from the conventional media players. The various features included in the Media and MP3 Player are 4k video will run smoothly , no issue of volume , no data taken, extra features added to player while playing music.

Thus we can say our Media and MP3 Player is a fully loaded player and a better option over the traditional media players. Media and MP3 Player is a good and dynamic option for music lovers, computers and mobile use

1.2 Literature Review

There are many free media players for Windows 10, offering a variety of tools, options, and support for different video formats. If you're struggling to figure out the best option, here are the best free media players available for Windows 10.

1. VLC Media Player

VLC Media Player is the most popular media player in the world. If you look on a site like Alternative To, you'll see that VLC ranks up top with over 6,000 Likes. VLC is clearly the king. But is it right for you? Maybe, maybe not.

VLC is complex and powerful. "All-in-one solution" describes it best, and you can do a lot with it, particularly with all of the advanced settings and options to tweak. The downside? VLC verges on "bloated" status and may not offer the best performance on older, slower hardware.

But if you hate tinkering and just want a media player that's free and works right out of the box, VLC is the answer. It can stream video URLs in real-time, and it can play all standard media types, including CDs, DVDs, and most popular video formats like MP4, AVI, and MKV. No need to download, install, and fiddle with codecs.

All in all, we understand why VLC ranks as one of the most popular GitHub projects to date. Considering it's been in active development since 2001, it's safe to say that VLC won't be going anywhere any time soon.

For all of this, VLC keeps its spot on our list of best Windows software.

Summary of benefits and notable features:

Supports most media codecs out of the box.

Supports playback from files, discs, external devices, webcams.

Supports online streaming with most mainstream protocols.

Hardware acceleration for fast GPU playback.

Customize appearance with the VLC Skin Editor.

Available on Windows, Mac, Linux, iOS, and Android.

2. Pot Player

Pot Player is a media player app from South Korea. If VLC didn't have such a strong brand identity, Pot Player would probably stand in its place as the king of the free media players.

VLC and Pot Player share a lot in common, notably that they both serve as easy all-in-one media players for users who just want an out-of-the-box solution. But unlike VLC, which can run into trouble when dealing with larger files or cutting-edge video formats, PotPlayer always works.

If you want to tweak options and customize them to your liking, Pot Player lets you. In fact, you'll find it packed with more settings than even VLC and lots of advanced features like scene previews, bookmarks, clip recording, and more. And best of all, it uses fewer resources than VLC.

Most users probably won't care, but you should know that Pot Player is free but proprietary software (i.e., not open source). Regardless, it's a great free media player for Windows 10.

Summary of benefits and notable features:

Detailed interface that shows a lot without being cluttered.

Supports most media codecs out of the box.

Supports playback from files, discs, external devices.

Supports online streaming with most mainstream protocols.

Better handling of large files (Blu-ray) and cutting-edge formats.

Available on Windows only.

3. Media Player Classic

Media Player Classic is one of those apps that can stir up nostalgia and send you down memory lane.

Released back in 2003, it was the favored alternative to Windows Media Player back during the Windows XP days. It stalled development in 2006 and has since forked into two separate projects: Home Cinema (MPC-HC) and Black Edition (MPC-BE).

Home Cinema is the better choice for everyday users, aiming to remain as lightweight as possible while supporting the latest standards and video formats. Black Edition is the superpowered version with more features, improvements, and enhancements, but it isn't as simple to use.

The Home Cinema version is what you see in the above screenshot, illustrating its basic yet effective (and familiar!) layout.

And that's really the biggest selling point of Media Player Classic: fast performance, low resource usage, small installation size—truly lightweight in every way. It handles most formats without issue, and it supports some advanced features like subtitle downloads, video capture, and integration with Skype.

Media Player Classic is the largest open-source alternative to VLC. If you don't like how much bloat VLC took on over the past few years, and if you don't like PotPlayer's closed-source development, then this is the free Windows 10 media player for you.

Unfortunately, MPC-HC was discontinued in July 2017, but it still works well and is worth using.

Summary of benefits and notable features:

Intuitive and easy-to-use interface.

Supports most media codecs out of the box.

Supports playback from files, discs, external devices.

Extremely lightweight, which means great performance on old machines.

Advanced features like subtitle downloads and Skype integration. Available on Windows only.

4. ACG Player

Now that we've got the "Big Three" out of the way, I want to highlight this gem of a video player that's in the Microsoft Store. It's called ACG Player, and it's tragically underrated. If you thought Media Player Classic was lightweight, this will blow you away.

The first thing you'll notice is the simplified interface and touch-based controls. Tap the top half for Play/Pause, or tap the bottom half to toggle the controls. Swipe left-right to rewind and fast-forward, swipe up-down for volume. Obviously, this app was designed for Windows 10 tablets, but keyboard shortcut alternatives exist too.

Advanced features include gesture customizations, multiple window mode, playlist management, online stream playback, and the ability to tweak subtitle appearances and animations. You can also opt for Ax-Lite Video Player, which is a faster version with some of the features cut out.

ACG Player does feature in-app ads and includes an in-app purchase you can use to remove them. However, they're not intrusive, displaying when you pause a video. It doesn't detract from your video watching experience at all.

Summary of benefits and notable features:

Simple, gesture-based interface. Great for tablets!

Supports most media codecs out of the box.

Supports playback from files, discs, external devices.

Lightweight and barebones design.

Available on Windows only.

5. MPV

In MPV, we have another free media player for Windows 10, which also happens to be an open-source, cross-platform contender to VLC. This particular project is a fork of both MPlayer and mplayer2, keeping the good bits of those, throwing out the junk, and introducing a whole lot more goodies.

The first thing that stands out is the lack of a traditional user interface. It's a pure video player with minimal controls overlaid at the bottom, and it's mostly controlled using mouse movements. Nifty and convenient for touchscreen devices.

Between MPV and VLC, MPV is definitely harder to use if you want to do anything more than just watch stuff. It's generally more efficient and less demanding resource-wise, but things like fiddling with subtitles or streaming to Chromecast can be a pain.

Overall, it has a bit of a niche appeal, but it's certainly worth giving a try. What do you have to lose?

Summary of benefits and notable features:

Minimal, mouse-based user interface.

Supports most media codecs out of the box.

Excellent video quality, even when scaling.

Lightweight and efficient video playback.

Available on Windows, Mac, Linux, Android.

1.3 Objectives

The various facilities and features have been planned to be incorporated in our Media and MP3 Player to differentiate it from traditional media players. We will attempt to attain this features in our Media and MP3 player.

[1] 4K video will runs smoothly without crash/lag/freeze.
[2] Volume issue of lower without command will be resolved.
[3] NO user data is stored, privacy will be maintained.
[4] Extra features will be added while playing music like +/- 10 and +/- 20.
[5] Specific to Basic User so less storage application.
[6] New Song can be added by user in the Playlist.
[7] Both Desktop and Mobile Application

1.4 Significance

In modern society, people live a fast-paced life, and pressure is constantly present in lives. Due to the wide use of technology music has become the daily essential spiritual food, everyone's mobile phone inside there must be a music player. An application like MP3 music players is used to balance stress and happiness. It accompanies people anytime, anywhere and anyplace such as when people taking the bus and exercising.

1.5 Research & Design

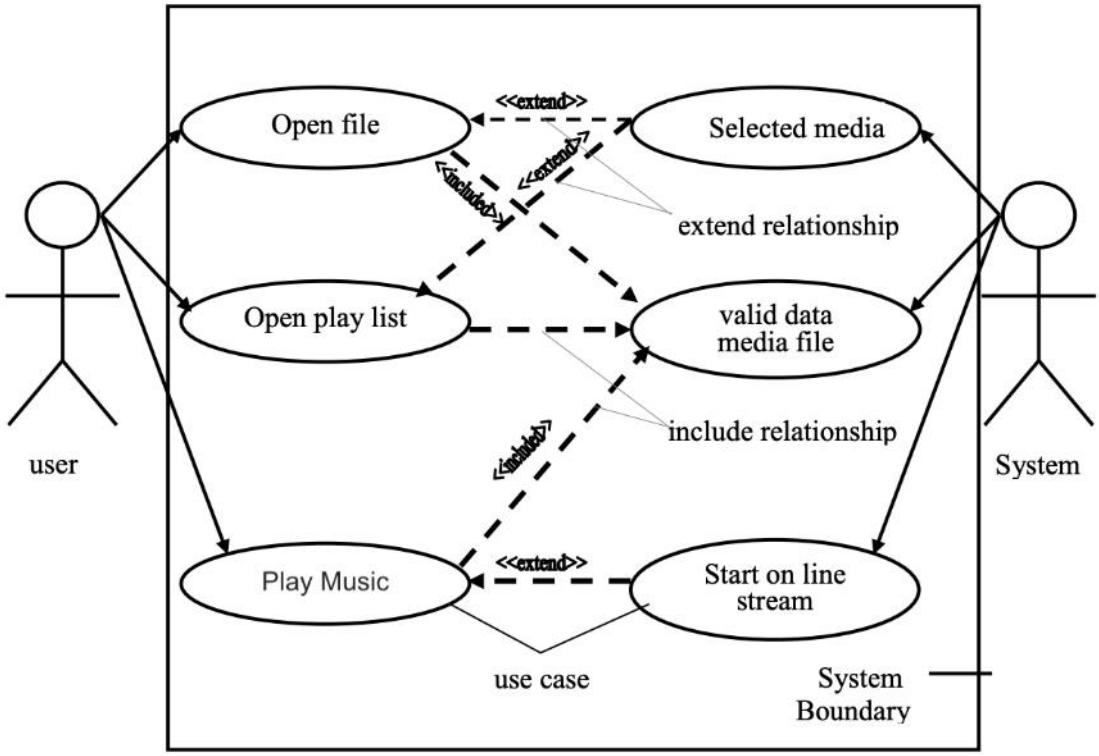

Figure-1-5-1 Use Case Diagram

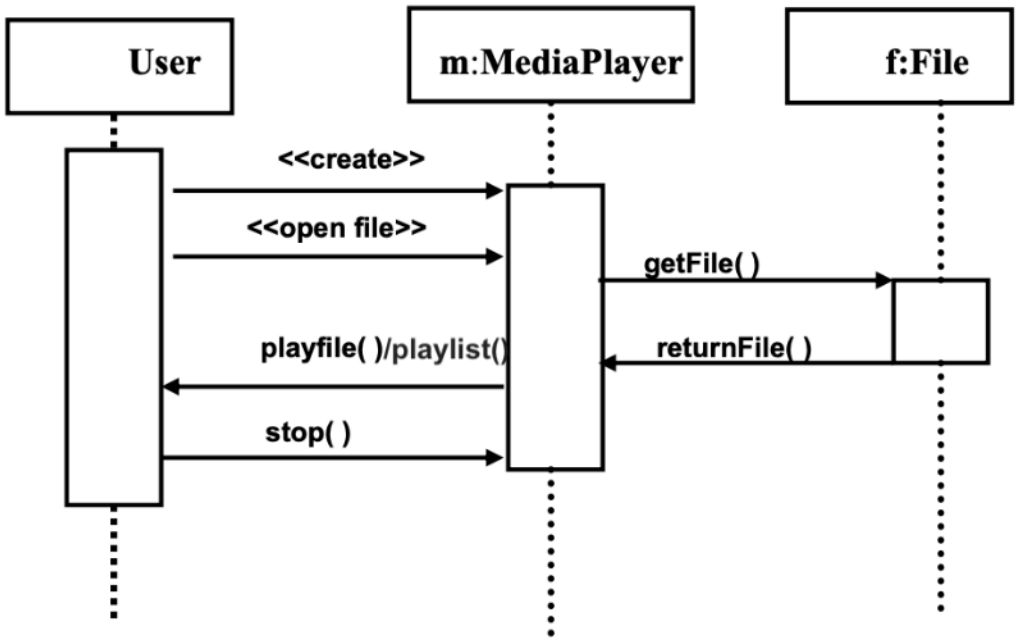

Figure-1-5-2 Sequence Diagram

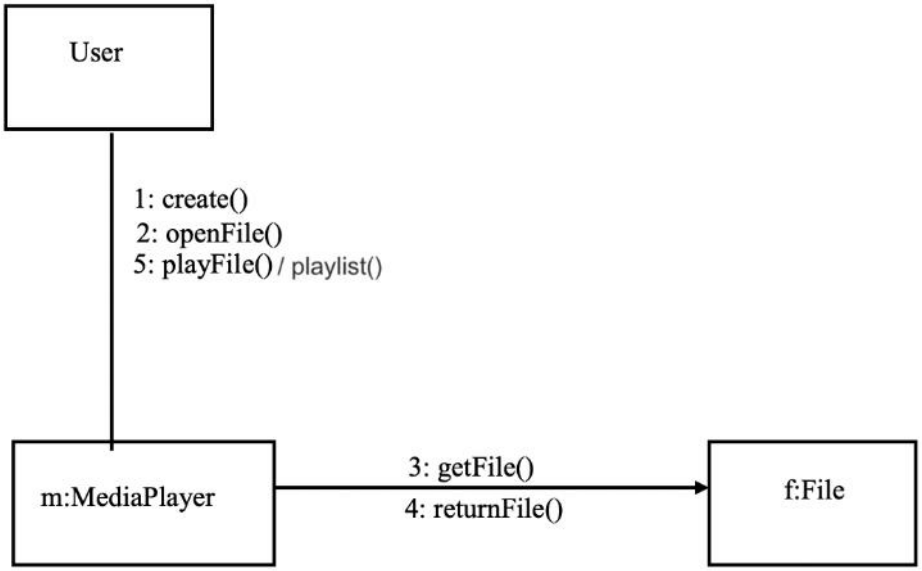

Figure-1-5-3 Collaboration Diagram

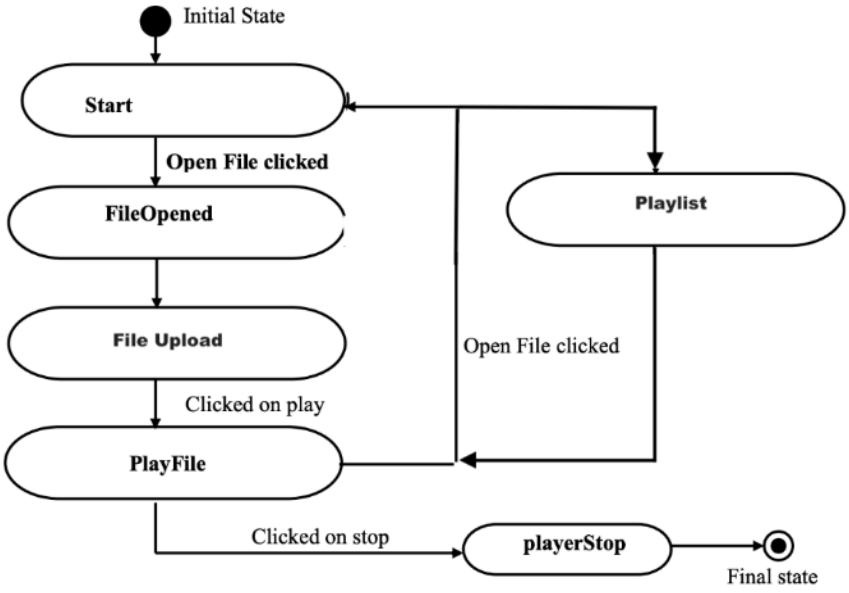

Figure-1-5-4 State Diagram

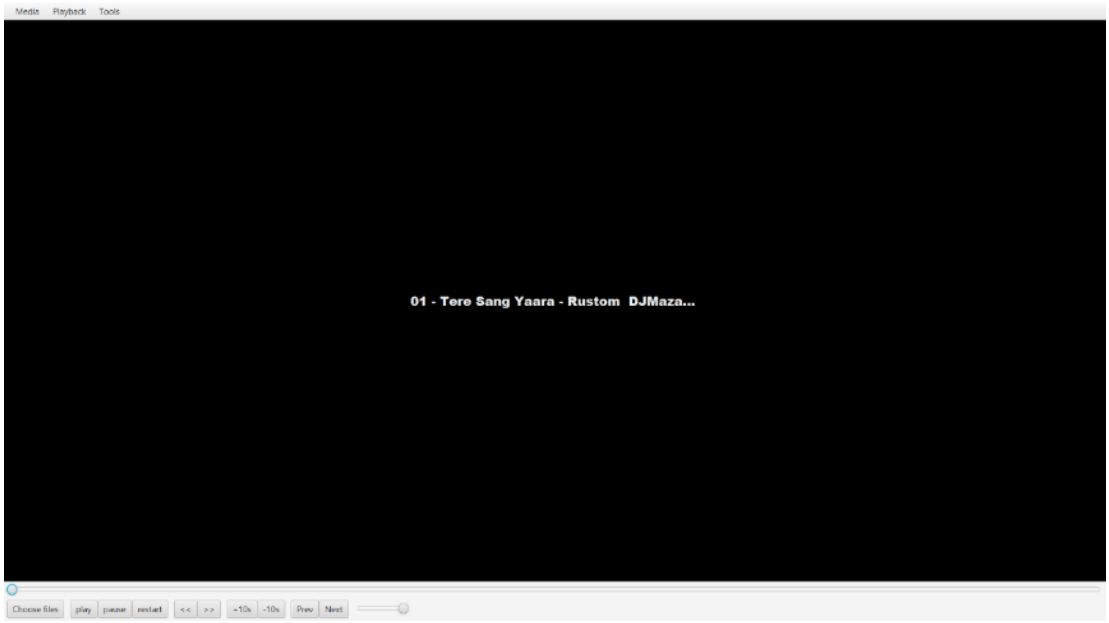

Figure-1-5-5 Main Screen of Media Player

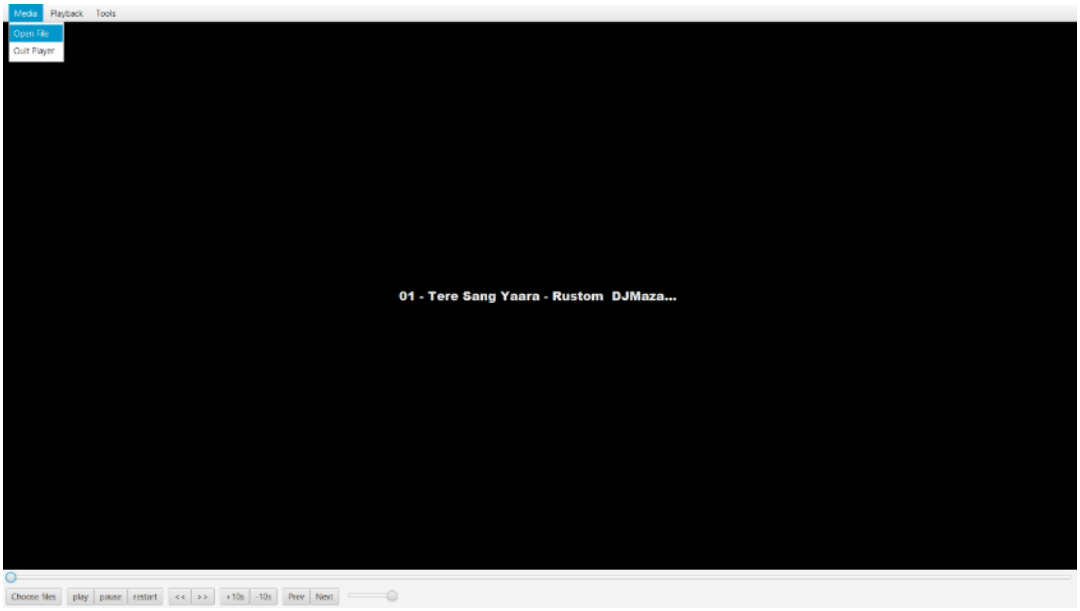

Figure-1-5-6 Media dropdown of Media Player

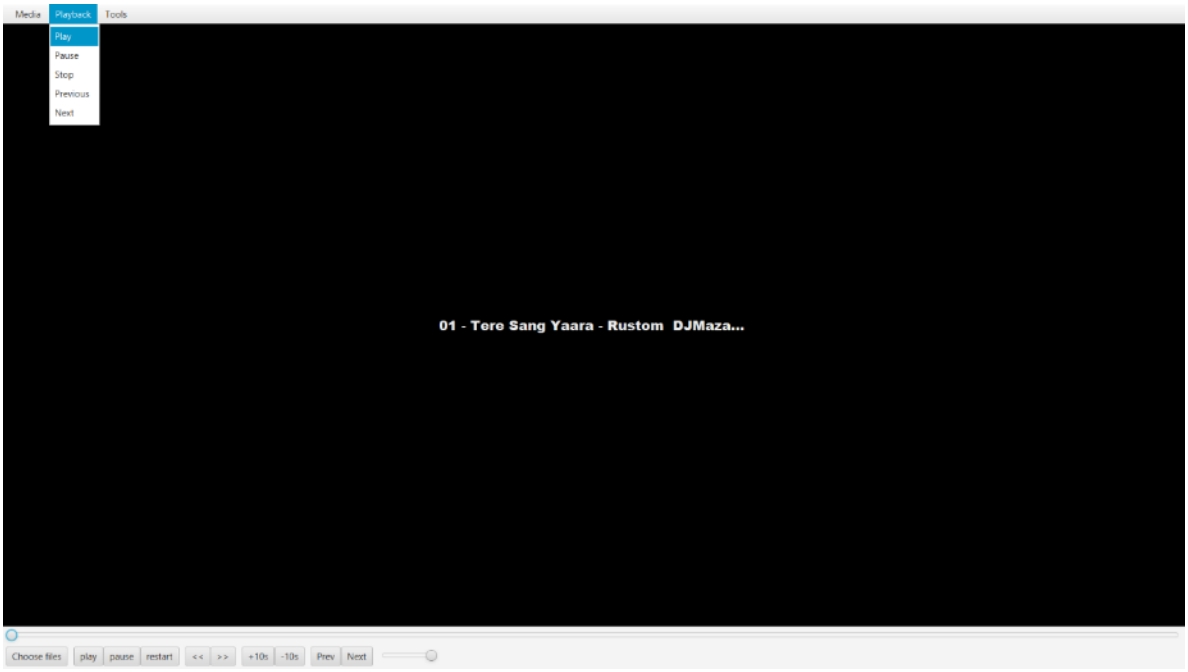

Figure-1-5-7 Playback dropdown of Media Player

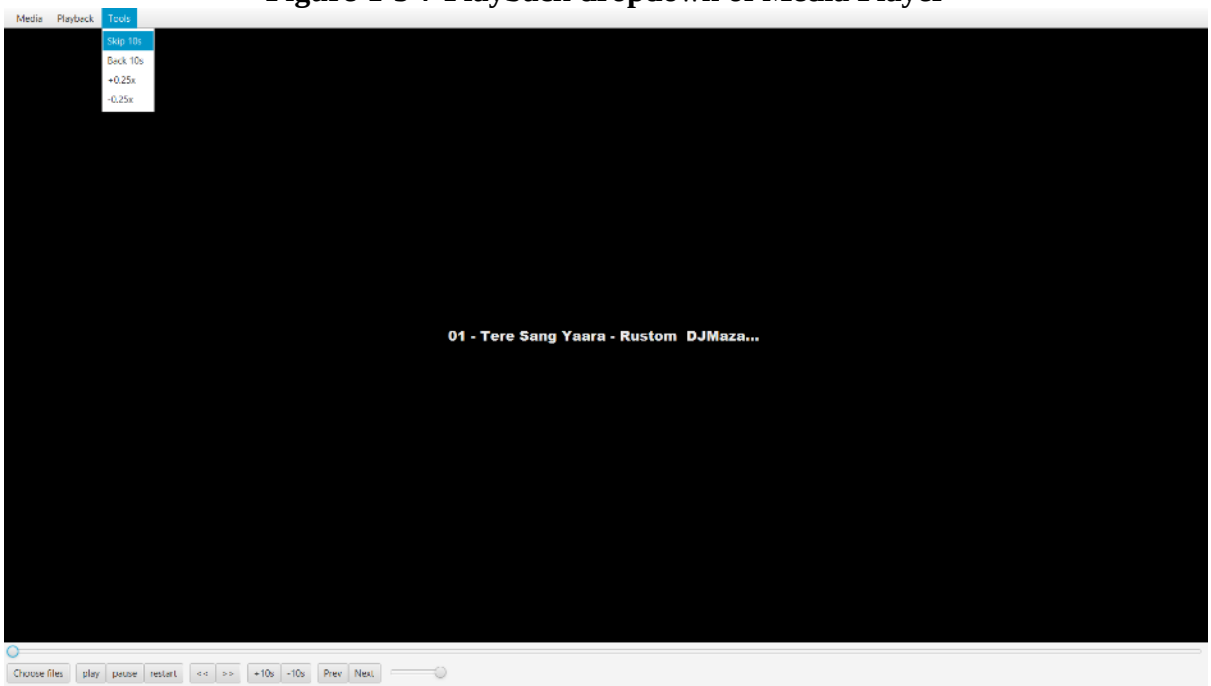

Figure-1-5-8 Tools dropdown of Media Player

Chapter 2

2.1 Experimental Setup

Feasibility:

Economic feasibility: To design media and mp3 player as long as a devices is high end it will runs. In addition, mobile phone media and mp3 player is basic needs for public. The information that which functions are necessary form all the consumers , which functions are needed for some people, and which features are seldom to use is easy to understand. And a lot of research is eliminated, thus saved the spending. Therefore, the whole process of development doesn't need to spend any money that is economic feasibility.

Technical feasibility: To design a music player which meets the basic requirements, a deep
understand of JAVA language, application of scene builder and other technical knowledge are needed. Based on the related technology information and resources for JAVA FX on the market, and equipped with technical personnel of technology and the spirit of willing to learn, the technology is feasible.

Social Feasibility : With the rapid development of the mobile phone and computer market, all kinds of audio and video resources are widely circulated on the Internet. These resources seem ordinary, but have gradually become an indispensable part of people life, which derived the development of all kinds of mobile phone/desktop player. But a lot of players devoted to fancy appearance, strong function causing a lot of wasted resources to the user's mobile phone/desktop and bringing a lot of inconvenience to the user as multitasking operation is needed. Some functions are useless to ordinary people. Powerful player is a good thing, but a lot of functions are actually useless for most users. Aimed at these problems, developing media and mp3 player which owns the features of simplified functions, common play function, meeting the needs of most users, less required memory and high quality of playing music, maximises the optimisation in performance.

Software/Hardware Requirement:

User Requirement: The application is intended for users who use and distribute music electronically. The application shall allow both novice users and more competent computer users to use the system effectively. For more frequent computer users, there will be options for advanced functions to allow them to generate their own playlists along with storing and loading multiple files. The target clients for our software are ordinary computer users who distribute music media. It is assumed that the target group does not have to be familiar with the use of any software of similar functionality, but to have a basic computer and Internet skills that will enable them to use this software.

Hardware Requirement: Hardware is the backbone of any up and running software. Every system demands a good hardware to support its functions. The hardware requirements are as follows:

a)Processor

Minimum: 733 MHz Pentium

b)Memory

Minimum: 256 MB (Application Area) and 128 MB (development purposes).

Recommended: 512 MB or higher.

c)Disk

Minimum: 20 MB

Software Requirement : Any android phones and for desktop windows 7 or above.

The required software of the developing environment

Operation system: Windows 10,Linux

Software：Netbeans ,Scene builder

JDK：Java Runtime Environment virtual machine、Java Development Kit(JDK)

2.2 Procedure Adopted

In this project, the agile development cycle will be used to guide the development process. The reason for using agile methods is that desktop applications have a short software life cycle and rapidly changing technologies, so users will constantly change their requirement and needs in response to technological changes. Therefore, the agile development cycle is more suitable for desktop application development because of iterative and flexible, so it can adapt effectively to changing customers.

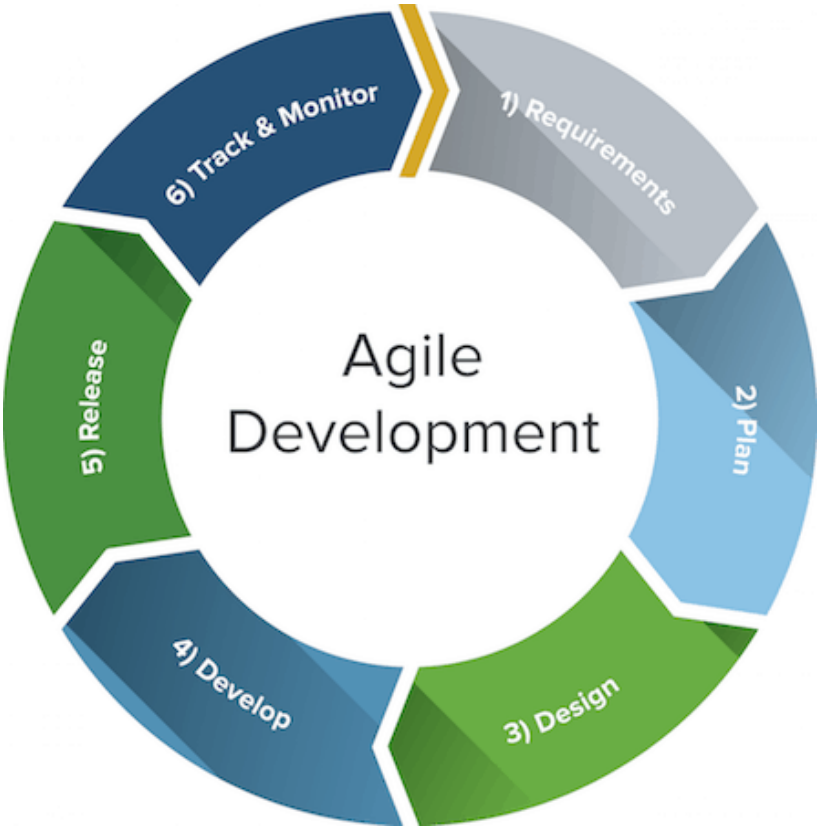

Figure-2-2-1 Agile Methodology

The agile development cycle contains 6 phrase which is requirement analysis, planning, design, implementation or development, testing, and deployment.

Requirement analysis: At this stage, we will review existing MP3 & Media players on the market. After the review, we will find out what current users need and idea to improve the existing music players and collect their comments and suggestions for further analysis.

Planning: In the planning stage, we should first try to explore out the features that the music player can have. Next, we will eliminate the features that users

feel no really useful or low cost-effective. Finally, each feature is prioritized and assigned to an iteration.

Design: The design stage is prepared according to the requirements of users. Since there are many details and problems encountered during development to be considered for each feature. Therefore, we will discuss and formulate solutions and test strategies to verify the product at this stage.

Implementation or Development: During the development phase, we will iteratively implement each of the features listed during the planning phase. At this stage, there will be many setbacks and obstacle, so the team needs to constantly overcome these obstacles. Moreover, we will prioritize the most important features and need to make intelligent trade-offs between the depth of completeness of a single feature and the breadth of implementation of multiple features.

Testing: In this stage, we will test the performance of each feature in order to check whether it meets the requirements of users. For example, we will test whether the application can be properly installed and run on a real device, and check whether any errors occur in the running process and each feature is up to standard.

Deployment: In this final phase, we will begin to deliver this application to the customer. For instance, we will deploy this MP3 & Media player application. In addition, we will anticipate that users will encounter unpredictable problems when using the player in this process, so we will solve these problems in a future version.

Chapter 3

3.1 Implementation

The proposed application completed the debugging task during the testing phase, then it should enter the deployment phase. In the deployment phase, the developer needs to publish the application's installation package which is the "Executable" file, to a platform such as Microsoft Store for users to download. However, due to the number of users are limit so far and the proposed application is not in the final public version, there are still many modules that should be improved and updated. Therefore, it will be uploaded to the relevant platform to promote to users after the final public version is released. In addition, users can execute the app in a non-network state.

Below are the steps to describe how a new user will execute the proposed application:

1. The user first execute the application load songs into the music playlist.

2. Users can play a song by clicking on next or previous button.

3. In the song playback interface, the user is allowed to drag the progress bar, as well as perform media control through the buttons.

4. Users can click on the +10 or -10 to skip a part of the song or video.

5. The user can use volume slider to adjust the volume.

6. The user can use the progress bar to navigate through the media.

3.2 Testing

Unit Testing 1: Menu Bar(Media)

Test Objective: To test Media dropdown functions are working correctly.

Input	Expected Output	Actual Output
Click "Open File" in the Media	File chooser should be opened and desired media file would be loaded in the player successfully.	Pass
Click "Quit Player" in the Media	Media & MP3 player application will be terminated.	Pass

Table-3-2-1 Unit Testing of Menu Bar(Media)

Unit Testing 2: Menu Bar(Playback)

Test Objective: To test Playback dropdown functions are working correctly.

Input	Expected Output	Actual Output
Click "Play" in the Playback	Loaded media file would be played without lag or crash.	Pass
Click "Pause" in the Playback	Current media file would be paused successfully.	Pass
Click "Stop" in the Playback	Stop the running media file and resets the progress bar to the start.	Pass
Click "Previous" in the Playback	Play the previous song from the playlist in the music directory.	Pass
Click "Next" in the Playback	Play the next song from the playlist in the music directory.	Pass

Table-3-2-2 Unit Testing of Menu Bar(Playback)

Unit Testing 3: Menu Bar(Tools)

Test Objective: To test Tools dropdown functions are working correctly.

Input	**Expected Output**	**Actual Output**
Click "Skip 10s" in the Tools	It would skip the song and progress bar 10 seconds forward.	Pass
Click "Back 10s" in the Tools	It would skip the song and progress bar 10 seconds backward.	Pass
Click "+0.25x" in the Tools	It would increase the playback speed of media file by 0.25 times.	Pass
Click "-0.25x" in the Tools	It would decrease the playback speed of media file by 0.25 times.	Pass

Table-3-2-3 Unit Testing of Menu Bar(Tools)

Unit Testing 4: The display area

Test Objective: To test the double click feature and other things of display area are working properly.

Input	**Expected Output**	**Actual Output**
Double mouse left click on the display area	It would be full screen or escape from the full screen mode.	Pass
Loading any media file	Showing video properly if it is a video file or the name of the audio if it is a audio file.	Pass

Table-3-2-4 Unit Testing of the display area

Unit Testing 5: Buttons playback control

Test Objective: To test the functionalities of buttons.

Input	**Expected Output**	**Actual Output**
Click “Choose files” button	File chooser should be opened and desired media file would be loaded in the player successfully.	Pass
Click “play” button	Loaded media file would be played without lag or crash.	Pass
Click “pause” button	Current media file would be paused successfully.	Pass
Click “restart” button	Stop the running media file and resets the progress bar to the start.	Pass
Click “<<” button	It would decrease the playback speed of media file by 0.25 times.	Pass
Click “>>” button	It would increase the playback speed of media file by 0.25 times.	Pass
Click “+10s” button	It would skip the song and progress bar 10 seconds forward.	Pass
Click “-10s” button	It would skip the song and progress bar 10 seconds backward.	Pass
Click “Prev” button	Play the previous song from the playlist in the music directory.	Pass
Click “Next” button	Play the next song from the playlist in the music directory.	Pass

Table-3-2-5 Unit Testing of buttons playback control

Unit Testing 6: Volume playback control

Test Objective: To test the functionalities of volume slider.

Input	Expected Output	Actual Output
Click mouse left button on any part of the volume slider	Volume and position of volume slider would be set according to the position of click of mouse.	Pass
Drag the volume bar using mouse hold	Volume and position of volume slider would be set according to the position of release of mouse click.	Pass

Table-3-2-6 Unit Testing of volume playback control

Unit Testing 7: Progress bar playback control

Test Objective: To test the functionalities of Progress bar.

Input	Expected Output	Actual Output
Click mouse left button on any part of the progress bar	Playing time and position of progress bar would be set according to the position of click of mouse.	Pass
Drag the progress bar using mouse hold	Playing time and position of progress bar would be set according to the release of click of mouse.	Pass

Table-3-2-7 Unit Testing of progress bar playback control

Chapter 4

Results and Discussions

Firstly, the proposed music player had achieved its first objective, which is to make the music player become a simple, easy-to-use, and well-run application. The proposed application had become faster startup, smaller size, and less memory usage by eliminating some unrealistic features. The application also adding some useful features like +10s and -10s.

Next, the proposed music player achieves a second objective which is to reduce the unnecessary controls and enhance the way the app interacts with the user, such as using double click . In addition, the app can also run on the lock screen or in the background.

Lastly, the proposed music player achieves a third objective which is a quick playlist. The application will use the predefined database and directory to add and operate on songs and traverse them faster.

Chapter 5

Summary and Discussions

In a nutshell, when users hold the mentality of venting and relaxation to expect the music player to bring them relief pressure, in result the application with a dazzling and complex interface, a variety of multifarious functions, from time to time prompt out of the advertising, as well as the function that requires be a members to use, which will only make users feel more depressed and feel the pressure. Moreover, most people who use a music player, usually don't leave the music player open in the foreground, but start playing music and then go on to do something else at hand such as take a break, read a book and news, or play a game.

In short, the proposed application will combine the strengths of most music players on the existing market and eliminate some unrealistic features, allowing users to focus on listening to music rather than store, communities or various VIP packages or features. The proposed Media & MP3 player will focus on improving the experience of users of the music player experience.

Chapter 6

Future Scope

The global portable media player market size is anticipated to reach USD 27.80 billion by 2025. It is projected to register a CAGR of 4.0% during the forecast period. The top notch features of the products such as high video and audio quality, compatibility, flexible storage capacity, ease of use, and portability among others, are anticipated to drive the demand.

Portable media players can be easily connected through Wi-Fi, Bluetooth, and USB, which makes them compatible with a number of devices. With advancements in technology and rise in Internet of Things devices, the demand for portable media players is increasing.

Appendices A

A1. MediaPlayer.java

```
package mediaplayer;

import javafx.application.Application;
import javafx.event.Event;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.input.MouseEvent;
import javafx.stage.Stage;

public class MediaPlayer extends Application {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource( name: "FXMLDocument.fxml"));

        Scene scene = new Scene(root);

        stage.setTitle("Media & MP3 Player");

        //stage.getIcons().add(new Image(""));

        stage.setFullScreenExitHint("Press ESC or Double click to exit the full Screen");

        scene.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent clicked) {
                if (clicked.getClickCount() == 2) {
                    if (!stage.isFullScreen()) {
                        stage.setFullScreen(true);
                    } else {
                        stage.setFullScreen(false);
                    }

                }
            }

        });

        stage.setScene(scene);
        stage.show();
    }

}
```

A2. FXMLDocument.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.StackPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.media.MediaView?>
<?import javafx.scene.text.Font?>

<BorderPane xmlns:fx="http://javafx.com/fxml/1" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
            prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/17"
            fx:controller="mediaplayer.FXMLDocumentController">
    <center>
        <StackPane prefHeight="150.0" prefWidth="200.0" style="-fx-background-color: black;"
                   BorderPane.alignment="CENTER">
            <children>
                <Label fx:id="Lb1" prefHeight="151.0" prefWidth="504.0" text="song name" textFill="#ededed">
                    <font>
                        <Font name="Arial Black" size="22.0"/>
                    </font>
                </Label>
                <MediaView fx:id="mediaView" fitHeight="200.0" fitWidth="200.0">
                    <StackPane.margin>
                        <Insets/>
                    </StackPane.margin>
                </MediaView>
            </children>
        </StackPane>
    </center>
    <top>
        <MenuBar BorderPane.alignment="CENTER">
            <menus>
                <Menu mnemonicParsing="false" text="Media">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#chooseFileMethod" text="Open File"/>
                        <MenuItem mnemonicParsing="false" onAction="#closeVideo" text="Quit Player"/>
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Playback">
                    <items>
                        <MenuItem mnemonicParsing="false" onAction="#play" text="Play"/>
                        <MenuItem mnemonicParsing="false" onAction="#pause" text="Pause"/>
                        <MenuItem mnemonicParsing="false" onAction="#stop" text="Stop"/>
                        <MenuItem mnemonicParsing="false" onAction="#previousMedia" text="Previous"/>
                        <MenuItem mnemonicParsing="false" onAction="#nextMedia" text="Next"/>
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Tools">
```

```
            <Menu mnemonicParsing="false" text="Tools">
                <items>
                    <MenuItem mnemonicParsing="false" onAction="#skip10" text="Skip 10s"/>
                    <MenuItem mnemonicParsing="false" onAction="#back10" text="Back 10s"/>
                    <MenuItem mnemonicParsing="false" onAction="#fastForward" text="+0.25x"/>
                    <MenuItem mnemonicParsing="false" onAction="#slowRate" text="-0.25x"/>
                </items>
            </Menu>
        </menus>
    </MenuBar>
</top>
<bottom>
    <VBox alignment="BOTTOM_CENTER" prefHeight="52.0" prefWidth="592.0" BorderPane.alignment="CENTER">
        <children>
            <Slider fx:id="progressBar">
                <VBox.margin>
                    <Insets bottom="5.0" left="5.0" right="5.0" top="5.0"/>
                </VBox.margin>
            </Slider>
            <HBox prefHeight="39.0" prefWidth="577.0">
                <children>
                    <Button mnemonicParsing="false" onAction="#chooseFileMethod" text="Choose files">
                        <HBox.margin>
                            <Insets bottom="5.0" left="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#play" text="play">
                        <HBox.margin>
                            <Insets bottom="5.0" left="10.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#pause" text="pause">
                        <HBox.margin>
                            <Insets bottom="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#stop" text="restart">
                        <HBox.margin>
                            <Insets bottom="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#slowRate" text="&amp;lt;&amp;lt;">
                        <HBox.margin>
                            <Insets bottom="5.0" left="10.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#fastForward" text="&amp;gt;&amp;gt;">
                        <HBox.margin>
                            <Insets bottom="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#skip10" text="+10s">
                        <HBox.margin>
                            <Insets bottom="5.0" left="10.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#back10" text="-10s">
                        <HBox.margin>
                            <Insets bottom="5.0" right="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#previousMedia" text="Prev">
                        <HBox.margin>
                            <Insets bottom="5.0" left="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Button mnemonicParsing="false" onAction="#nextMedia" text="Next">
                        <HBox.margin>
                            <Insets bottom="5.0" top="5.0"/>
                        </HBox.margin>
                    </Button>
                    <Slider fx:id="volume" prefHeight="14.0" prefWidth="94.0">
                        <HBox.margin>
                            <Insets bottom="5.0" left="10.0" top="10.0"/>
                        </HBox.margin>
                    </Slider>
                </children>
            </HBox>
        </children>
    </VBox>
</bottom>
</BorderPane>
```

A3. FXMLDocumentController.java

```
package mediaplayer;

import java.io.File;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import javafx.beans.InvalidationListener;
import javafx.beans.Observable;
import javafx.beans.binding.Bindings;
import javafx.beans.property.DoubleProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.Event;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.input.MouseEvent;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.scene.shape.Path;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.util.Duration;
import javax.swing.JOptionPane;


public class FXMLDocumentController implements Initializable {

    private String path;
    private MediaPlayer mediaPlayer;
    private Media media;
    @FXML
    private MediaView mediaView;
    @FXML
    private Slider progressBar;
    @FXML
    private Slider volume;
    @FXML
    private Label Lb1;
    private File directory;

    private File[] files;

    private ArrayList<File> songs;

    private int songNumber;
```

```
private double i=1;

@FXML
private void closeVideo(ActionEvent event){
    System.exit( status: 0);
}

public void play(ActionEvent event){
    mediaPlayer.play();
    mediaPlayer.setRate(1);
}
public void pause(ActionEvent event){
    mediaPlayer.pause();
}
public void stop(ActionEvent event){
    mediaPlayer.stop();
}



 public void skip10(ActionEvent event){
    mediaPlayer.seek(mediaPlayer.getCurrentTime().add(Duration.seconds(10)));
}
public void back10(ActionEvent event){
    mediaPlayer.seek(mediaPlayer.getCurrentTime().add(Duration.seconds(-10)));
}


 public void chooseFileMethod(ActionEvent event){
    FileChooser fileChooser= new FileChooser();
    File file = fileChooser.showOpenDialog( ownerWindow: null);
    path=file.toURI().toString();
    mediaPlayer.stop();
    mediaPlayer.dispose();
    if(path !=null){
        Media media=new Media(path);
        mediaPlayer =new MediaPlayer(media);

        //mediaPlayer.dispose();
        mediaView.setMediaPlayer(mediaPlayer);
        Lb1.setText(" ");
        DoubleProperty widthProp = mediaView.fitWidthProperty();
        DoubleProperty heightProp = mediaView.fitHeightProperty();

        widthProp.bind(Bindings.selectDouble( mediaView.sceneProperty(),  ...steps: "width"));
        heightProp.bind(Bindings.selectDouble( mediaView.sceneProperty(),  ...steps: "height"));
```

```
        mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
            @Override
            public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
             progressBar.setValue(newValue.toSeconds());
            }
        });



        progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         mediaPlayer.setOnReady(new Runnable() {
            @Override
            public void run() {
                Duration total=media.getDuration();
                progressBar.setMax(total.toSeconds());
            }
         });


         volume.setValue(mediaPlayer.getVolume()*100);
         volume.valueProperty().addListener(new InvalidationListener() {
            @Override
            public void invalidated(Observable observable) {
                mediaPlayer.setVolume(volume.getValue()/100);
            }
         } );
        mediaPlayer.play();
    }

}


public void slowRate(ActionEvent event){
    mediaPlayer.setRate(i=i-0.25);
}
```

```
public void fastForward(ActionEvent event){
    mediaPlayer.setRate(i=i+0.25);
}


public void previousMedia(ActionEvent event) {
           i=1;
    if(songNumber > 0) {


        songNumber--;


        mediaPlayer.stop();
                   mediaPlayer.dispose();



        media = new Media(songs.get(songNumber).toURI().toString());

        mediaPlayer = new MediaPlayer(media);
                   mediaPlayer.setOnEndOfMedia(new Runnable() {
   public void run() {
     media = new Media(songs.get(songNumber).toURI().toString());

        mediaPlayer = new MediaPlayer(media);
                   playMedia();
   }
});
                   Lb1.setText(songs.get(songNumber).getName());
                       mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
           @Override
           public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
            progressBar.setValue(newValue.toSeconds());

           }
        });



        progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
           @Override
           public void handle(MouseEvent event) {
               mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
           }
        });

         progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
           @Override
           public void handle(MouseEvent event) {
               mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
           }
        });
```

```
    mediaPlayer.setOnReady(new Runnable() {
       @Override
       public void run() {
           Duration total=media.getDuration();
           progressBar.setMax(total.toSeconds());
       }
    });
           volume.setValue(mediaPlayer.getVolume()*100);
    volume.valueProperty().addListener(new InvalidationListener() {
       @Override
       public void invalidated(Observable observable) {
           mediaPlayer.setVolume(volume.getValue()/100);
       }
    } );

    playMedia();

}

else {

    songNumber = songs.size() - 1;

    mediaPlayer.stop();
                mediaPlayer.dispose();



    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);


                Lb1.setText(songs.get(songNumber).getName());
                  mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
       @Override
       public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
        progressBar.setValue(newValue.toSeconds());
       }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
       @Override
       public void handle(MouseEvent event) {
           mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
       }
    });
```

```
     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });

            volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );

    playMedia();

}
        mediaPlayer.setOnEndOfMedia(new Runnable() {
public void run() {
 if(songNumber < songs.size() - 1) {

    songNumber++;

    mediaPlayer.stop();
                mediaPlayer.dispose();


    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());

                mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });
```

```
    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });

            volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );

    playMedia();

}

else {

    songNumber = 0;

    mediaPlayer.stop();
                mediaPlayer.dispose();

    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());
                    mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });
```

```
        progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         mediaPlayer.setOnReady(new Runnable() {
            @Override
            public void run() {
                Duration total=media.getDuration();
                progressBar.setMax(total.toSeconds());
            }
         });

                volume.setValue(mediaPlayer.getVolume()*100);
         volume.valueProperty().addListener(new InvalidationListener() {
            @Override
            public void invalidated(Observable observable) {
                mediaPlayer.setVolume(volume.getValue()/100);
            }
         } );

        playMedia();

    }
   }
});

 }
```

```
public void nextMedia(ActionEvent event) {

          i=1;
    if(songNumber < songs.size() - 1) {

        songNumber++;

        mediaPlayer.stop();
                    mediaPlayer.dispose();


        media = new Media(songs.get(songNumber).toURI().toString());

        mediaPlayer = new MediaPlayer(media);

                    Lb1.setText(songs.get(songNumber).getName());

                    mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
            @Override
            public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
             progressBar.setValue(newValue.toSeconds());
            }
        });


        progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
            }
        });

         mediaPlayer.setOnReady(new Runnable() {
            @Override
            public void run() {
                Duration total=media.getDuration();
                progressBar.setMax(total.toSeconds());
            }
         });

                volume.setValue(mediaPlayer.getVolume()*100);
         volume.valueProperty().addListener(new InvalidationListener() {
            @Override
            public void invalidated(Observable observable) {
                mediaPlayer.setVolume(volume.getValue()/100);
            }
         } );

        playMedia();

    }
```

```
else {

    songNumber = 0;

    mediaPlayer.stop();
                mediaPlayer.dispose();

    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());
                    mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });

            volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );
```

```
    playMedia();

}
        mediaPlayer.setOnEndOfMedia(new Runnable() {
public void run() {
 if(songNumber < songs.size() - 1) {

    songNumber++;

    mediaPlayer.stop();
                mediaPlayer.dispose();


    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());

                mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });
```

```
            volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );

    playMedia();

}

else {

    songNumber = 0;

    mediaPlayer.stop();
                mediaPlayer.dispose();

    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());
                    mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });
```

```
        mediaPlayer.setOnReady(new Runnable() {
            @Override
            public void run() {
                Duration total=media.getDuration();
                progressBar.setMax(total.toSeconds());
            }
        });

                volume.setValue(mediaPlayer.getVolume()*100);
        volume.valueProperty().addListener(new InvalidationListener() {
            @Override
            public void invalidated(Observable observable) {
                mediaPlayer.setVolume(volume.getValue()/100);
            }
        } );

        playMedia();

    }
   }
});


 }




@Override
public void initialize(URL url, ResourceBundle rb) {
        songs = new ArrayList<File>();

    directory = new File( pathname: "music");

    files = directory.listFiles();

    if(files != null) {

        for(File file : files) {

            songs.add(file);

        }



    }
```

```
    media = new Media(songs.get(songNumber).toURI().toString());
    mediaPlayer = new MediaPlayer(media);

    Lb1.setText(songs.get(songNumber).getName());

    mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });

     volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );

      mediaPlayer.setOnEndOfMedia(new Runnable() {
public void run() {
  if(songNumber < songs.size() - 1) {

    songNumber++;

    mediaPlayer.stop();
                mediaPlayer.dispose();
```

```
    media = new Media(songs.get(songNumber).toURI().toString());

    mediaPlayer = new MediaPlayer(media);

                Lb1.setText(songs.get(songNumber).getName());

                mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
        @Override
        public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
         progressBar.setValue(newValue.toSeconds());
        }
    });


    progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
        }
    });

     mediaPlayer.setOnReady(new Runnable() {
        @Override
        public void run() {
            Duration total=media.getDuration();
            progressBar.setMax(total.toSeconds());
        }
     });

            volume.setValue(mediaPlayer.getVolume()*100);
     volume.valueProperty().addListener(new InvalidationListener() {
        @Override
        public void invalidated(Observable observable) {
            mediaPlayer.setVolume(volume.getValue()/100);
        }
     } );

    playMedia();

}
```

```
        else {

            songNumber = 0;

            mediaPlayer.stop();
                        mediaPlayer.dispose();

            media = new Media(songs.get(songNumber).toURI().toString());

            mediaPlayer = new MediaPlayer(media);

                        Lb1.setText(songs.get(songNumber).getName());
                            mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>() {
                @Override
                public void changed(ObservableValue<? extends Duration> observable, Duration oldValue, Duration newValue) {
                 progressBar.setValue(newValue.toSeconds());
                }
            });


            progressBar.setOnMousePressed(new EventHandler<MouseEvent>() {
                @Override
                public void handle(MouseEvent event) {
                    mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
                }
            });

             progressBar.setOnMouseDragged(new EventHandler<MouseEvent>() {
                @Override
                public void handle(MouseEvent event) {
                    mediaPlayer.seek(Duration.seconds(progressBar.getValue()));
                }
            });

             mediaPlayer.setOnReady(new Runnable() {
                @Override
                public void run() {
                    Duration total=media.getDuration();
                    progressBar.setMax(total.toSeconds());
                }
             });

                    volume.setValue(mediaPlayer.getVolume()*100);
             volume.valueProperty().addListener(new InvalidationListener() {
                @Override
                public void invalidated(Observable observable) {
                    mediaPlayer.setVolume(volume.getValue()/100);
                }
             } );

            playMedia();

        }
       }
   });
    }

    private void playMedia() {
        mediaPlayer.play();
    }

}
```