

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI-590018, KARNATAKA



A Mini Project Report
On

“Music Player Application”

Submitted in the partial fulfilment of the requirement for the completion of **Mobile Application Development Laboratory with Mini Project (18CSMP68)** and award of degree of

**BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING**

Submitted By

**Jyotsna B
1va18is010**

**Pooja Guttal
1va18is023**

**Srima Shetty
1va18is045**

Under the Guidance of

**Mr. Abhijit Das
Assistant Professor
Dept. of ISE, SVIT**



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
SAI VIDYA INSTITUTE OF TECHNOLOGY**

(Affiliated to Visvesvaraya Technological University, Belagavi | Recognized by Govt. of
Karnataka | Approved by AICTE, New Delhi)

RAJANUKUNTE, BENGALURU – 560 06

2021-22

SAI VIDYA INSTITUTE OF TECHNOLOGY

Rajankunte, Bengaluru- 560 064

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the Mini project work entitled “**MUSIC PLAYER APPLICATION**” carried out by **Ms. Jyotsna B (1VA18IS010), Ms. Pooja Guttal (1VA18IS023), Ms Srima Shetty (1VA18IS023)**, students of **SAI VIDYA INSTITUTE OF TECHNOLOGY**, Bengaluru, in partial fulfilment for the award of Bachelor of Engineering in Information Science and Engineering of **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belagavi during the year **2021-22**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of mini-Project work prescribed for the **Mobile Application Development Laboratory with Mini Project (18CSMP68)**.

Mr. Abhijit Das

Assistant Professor,
Dept. of IS&E, SVIT

Dr. Vrinda Shetty

HOD
Dept. of IS&E, SVIT

Dr. H S Ramesh Babu

Principal

External Viva:

Name

Signature

1. _____

2. _____

ABSTRACT

This project is about the mp3 music player application development using Android. 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. Also, it will be kept improved based on user feedback.

In addition, depending on the user's usage scenario, it allows users to use the application in any situation or environment. On the other hand, the existing music players pay less attention to the control of gestures. Therefore, the music player will solve the limitation by adding more gestures and use your voice phone feature for media control to make it more user-friendly.

In a nutshell, the methodology for developing the mp3 music application used in this project is the agile development cycle. 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.

ACKNOWLEDGEMENT

The completion of the mini project brings with a sense of satisfaction, but it is never complete without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **Sai Vidya Institute of Technology**, for providing us an opportunity to do our education.

I would like to thank the **Management, Prof. M R Holla**, Director, Sai Vidya Institute of Technology and **Prof. A M Padma Reddy**, Director (A), Sai Vidya Institute of Technology for providing the facilities.

I extend my deep sense of sincere gratitude to **Dr. H S Ramesh Babu**, Principal, Sai Vidya Institute of Technology, Bengaluru, for having permitted to carry out the project work on “**ROCK PAPER SCISSOR**” successfully.

I express my heartfelt sincere gratitude to **Dr. Vrinda Shetty**, HOD, Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for her valuable suggestions and support.

I express my special in-depth, heartfelt, sincere gratitude to **Mr. Abhijit Das**, Assistant Professor, Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their constant support.

Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Information Science and Engineering, Sai Vidya Institute of Technology, Bengaluru, for their support.

JYOTSNA B **1VA18IS010**

POOJA GUTTAL **1VA18IS023**

SRIMA SHETTY **1VA18IS045**

TABLE OF CONTENTS

ABSTRACT	1
ACKNOWLEDGEMENT	2
LIST OF FIGURES	4
LIST OF TABLES	5

Chapter No	Chapter Name	Page No
1	INTRODUCTION	1-3
2	LITERATURE SURVEY	4-5
3	SYSTEM REQUIREMENT SPECIFICATION	6-7
4	SYSTEM MODELING	8-11
5	IMPLEMENTATION	12-21
6	TESTING AND RESULTS	22-24
7	CONCLUSION	25
8	REFERENCE	26
APPENDIX A	SNAPSHOTS	27-33

LIST OF FIGURES

Figure No	Title	Page No
4.1.1	Architectural Design	8
4.2.1	Flow Chart	10
4.3.1	Data Flow Diagram	11

LIST OF TABLES

Table No	Caption	Page No
6.2.1	Test Case 1	22
6.2.2	Test Case 2	22
6.2.3	Test Case 3	22
6.2.4	Test Case 4	22
6.2.5	Test Case 5	22
6.2.6	Test Case 6	23

CHAPTER 1

INTRODUCTION

One of the more popular forms of coding in recent times is the creation of applications, or apps, that run on mobile devices like phones and tablets. You probably use a range of different apps in your everyday life. Wouldn't it be cool to create one of your own?

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS.

The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser.

The programming and markup languages used for this kind of software development include Java, Kotlin, C++ [along with NDK (Native development Kit)], C#, Python and many more.

1.1 What is the need of Mobile Application Development?

Mobile app development is rapidly growing. From retail, telecommunications and e-commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transactions and access information. Today, mobile devices and the mobile applications that unlock their value are the most popular way for people and businesses to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employee's demand.

1.2 Android Studio:

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

1.3 Features of Android Studio:

- A flexible Gradle-based build system.
- A fast and feature-rich emulator.
- A unified environment where you can develop for all Android devices.
- Apply Changes to push code and resource changes to your running app without restarting your app.
- Code templates and GitHub integration to help you build common app features and import sample code.
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems.
- C++ and NDK support.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

1.4 Benefits of an Android Application

- 1** Improves Efficiency.
- 2** Offers High Scalability.
- 3** Secures Your App Data.
- 4** Integrates With Existing Software.
- 5** Easy to Maintain.
- 6** Improves Customer Relationship.
- 7** Facilitates New Client Data Retrieval.
- 8** Provides Real-time Project Access.
- 9** Ease in Project Management.
- 10** Record Digital Files for Accountability

1.5 Objective of the project

The primary aim of this “MUSIC PLAYER APPLICATION” is to make users feel comfortable and relaxed because it will pay more attention to the features commonly used by users. The MP3 music player will add features triggered by gestures to make it easier for users to use as well as less dependent on touch buttons.

1.5.1 Problem Statement

In the world of software development there is lots of improvement in the area of management applications. The implementation details are changing depending upon the need of the customers.

Due to the fierce competition between music player applications, many developers tried to add many features, advertise and content to their respective music player in order to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. Most music player apps use touch buttons to play, pause and switch between previous and next songs while ignoring the convenience of using gesture swiping to control the music player. Consumers will become more comfortable and confident in this way of interaction, so we should consider using gesture control on more mobile applications.

Scope of the Project

The main highlight of the project is to make the proposed application become a high learnability application without too many complex features, enhance the interaction between the user and the media control so that the user can have a better experience to achieve real relief of listening to music. The ability to enhance the interaction between users and media control is that the application can play, pause, rewind, forward, play previous or next song by using their voice. The application utilizes the voice gesture controls to get rid of its reliance on touch buttons.

CHAPTER 2

LITERATURE SURVEY

2.1 Problems with the existing application

- The existing music applications are all reliant on the touch gesture to play, pause, forward, rewind or to select a song.
- The applications available have advertisements and other additional features which make it hard for the user to enjoy the actual content.
- The complexity of the features make it confusing to follow the path to the desired function like searching a song in a playlist, adding songs to libraries and so on.

2.2 Proposed Application

- The proposed system is easy to use and does not have any unwanted complex features.
- The ability to enhance the interaction between users and media control by giving options on how they want to interact with the music player application.
- The proposed system eliminates the reliance on touch buttons as a user can opt for using voice gesture to interact with the music player.
- A user can now play a song, pause a song, rewind or forward a song, skip a song or play the previous song verbally.
- User can have a better experience to achieve real relief of listening to music.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

3.1 Functional Requirements

A description of the facility or feature required. Functional requirements deal with what the system should do or provide for users. They include description of the required functions, outlines of associated reports, and details of data to be held in the system.

3.1.1 Interface Requirements:

- Songs are to be downloaded on your phone.
- Scroll and select the song to be played.
- Songs can be paused, played, fast forwarded and rewind.
- User can hold and speak to access the controls available in the application.

3.2 Non-Functional Requirements:

Non-functional requirements define the overall qualities or attributes of the resulting system.

3.2.1 Usability

User would be able to understand the flow of App easily i.e users will able to use App without any guideline or help from experts/manuals.

3.2.2 Responsiveness

User would be able to understand the flow of App easily i.e users will able to use App without any guideline or help from experts/manuals.

3.2.3 Reliability

The Application is responsive to the user's voice and touch to select a song, play, pause, rewind, forward, play next song and to play previous song.

3.3 Software Requirements

- Operating System : ANDROID
- Programming Language : JAVA and XML
- Tool Used : ANDROID STUDIO

3.4 Hardware Requirements

- CPU Type : Intel Premium 5 and above.
- Clock speed : 3.0 GHz
- RAM Size : 8 GB and above
- Mobile : Android Mobile

CHAPTER 4

SYSTEM MODELING

This chapter of the report describes the structure of the project, followed by Design, Flow Chart, Data Flow Diagram.

4.1. ARCHITECTURAL DESIGN:

Architectural design is a process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.”

Architectural design is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication. The output of this design process is a description of the software architecture.

It represents the link between specification and design processes and is often carried out in parallel with some specification activities. It involves identifying major system components and their communications.

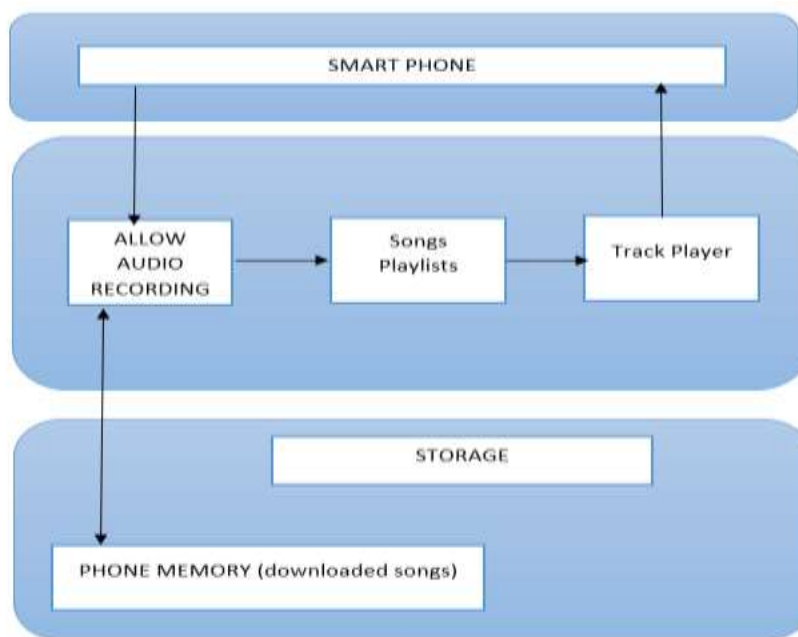







Fig 4.1.1: Architectural design of Music Player Application

4.2. Flow Chart:

A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. With proper design and construction, it communicates the steps in a process very effectively and efficiently.

FLOW CHART NOTATIONS:

Flowcharts consist of a few common geometric shapes representing steps.

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

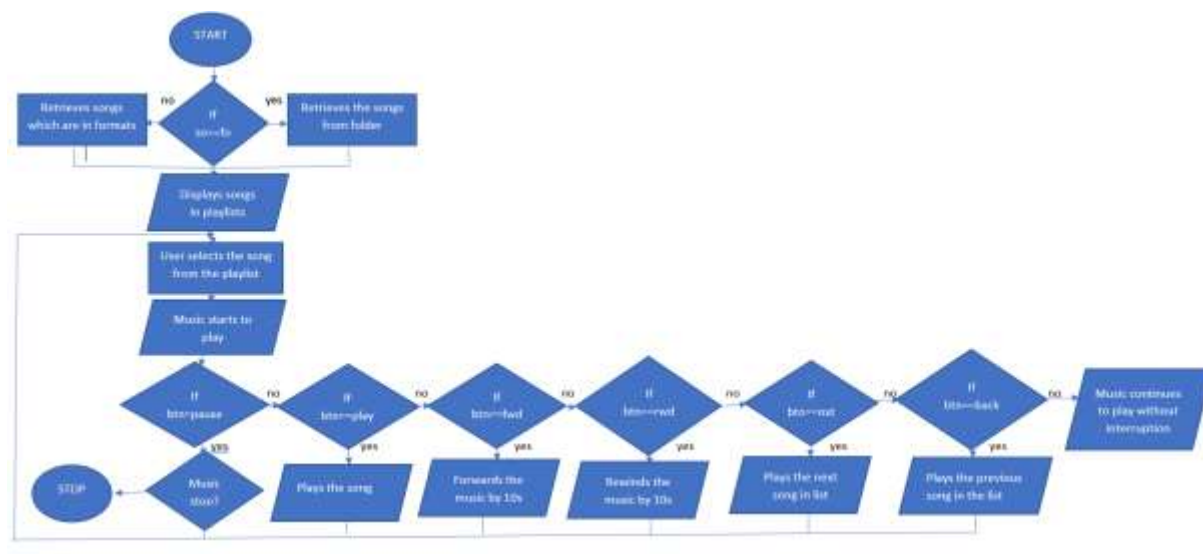


Fig 4.2.1: Flow Chart of Music Player Application

4.3 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

DFD (data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher-level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 3 levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

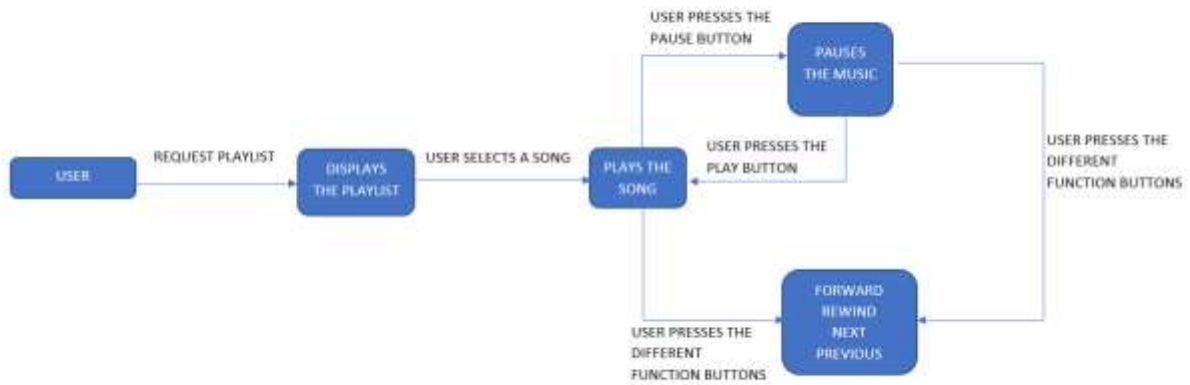


Fig 4.3.1: Data Flow Diagram of Music Player Application

CHAPTER 5

IMPLEMENTATION

This chapter of the report describes the Functions, packages and modules used in the project:

5.1 Source code

5.1.1 XML code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/parentLinearLayout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_img"
    android:orientation="vertical"
    android:weightSum="10"
    tools:context=".PlayerActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center"
        android:layout_weight="7"
        android:orientation="vertical">

        <TextView
            android:id="@+id/txttsn"
            android:layout_width="370dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="100dp"
            android:ellipsize="marquee"
            android:marqueeRepeatLimit="marquee_forever"
            android:padding="10dp"
            android:singleLine="true"
            android:text="Song name"
            android:textAlignment="center"
            android:textColor="#FFF"
            android:textSize="22sp"
            android:textStyle="italic">

        </TextView>

        <ImageView
            android:id="@+id/imageview"
            android:layout_width="252dp"
            android:layout_height="250dp"
            android:layout_gravity="center"
```

```
        android:layout_marginTop="20dp"
        android:src="@drawable/music">

</ImageView>
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="60dp">
    <SeekBar
        android:id="@+id/seekbar"
        android:layout_centerInParent="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="20dp"
        android:layout_marginBottom="250dp"
        android:layout_width="250dp"
        android:layout_height="wrap_content">

    </SeekBar>
    <TextView
        android:id="@+id/txtsstart"
        android:layout_toLeftOf="@+id/seekbar"
        android:layout_centerInParent="true"
        android:layout_alignParentLeft="false"
        android:layout_marginLeft="20dp"
        android:text="0:10"
        android:textSize="14sp"
        android:textColor="#FFF"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

    </TextView>

    <TextView
        android:id="@+id/txtsstop"
        android:layout_toRightOf="@+id/seekbar"
        android:layout_centerInParent="true"
        android:layout_alignParentRight="false"
        android:layout_marginRight="20dp"
        android:text="0:10"
        android:textSize="14sp"
        android:textColor="#FFF"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

    </TextView>
</RelativeLayout>

</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="3">

    <RelativeLayout
        android:id="@+id/lower"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="0dp">
```

```
<Button
    android:id="@+id/playbtn"
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:layout_centerHorizontal="true"
    android:background="@drawable/ic_pause"
    app:backgroundTint="#FF362E">

</Button>

<Button
    android:id="@+id/btnnext"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginTop="15dp"
    android:layout_toRightOf="@+id/playbtn"
    android:background="@drawable/ic_next"
    app:backgroundTint="#FFFFFF">

</Button>

<Button
    android:id="@+id/btnprev"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginTop="15dp"
    android:layout_toLeftOf="@+id/playbtn"
    android:background="@drawable/ic_previous"
    app:backgroundTint="#FFFFFF">

</Button>

<Button
    android:id="@+id/btnff"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_marginLeft="15dp"
    android:layout_marginTop="20dp"
    android:layout_toRightOf="@id/btnnext"
    android:background="@drawable/ic_fast_forward"
    app:backgroundTint="#FFFFFF">

</Button>

<Button
    android:id="@+id/btnfr"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="15dp"
    android:layout_toLeftOf="@id/btnprev"
    android:background="@drawable/ic_fast_rewind"
    app:backgroundTint="#FFFFFF">

</Button>
```

```
        </RelativeLayout>
    </LinearLayout>

    <Button
        android:id="@+id/voice_enabled_btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="Voice Enabled Mode - ON"
        app:backgroundTint="#E84242"
        android:textSize="16dp"
        android:textAllCaps="false">

    </Button>

</LinearLayout>
```

5.1.1 MainActivity.java code

```
package com.example.music2;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.Settings;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.karumi.dexter.Dexter;
import com.karumi.dexter.MultiplePermissionsReport;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.multi.MultiplePermissionsListener;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    ListView listView;
    String[] items;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        listView= findViewById(R.id.listViewSong);
        runtimePermission();
    }

    public void runtimePermission()
    {
        Dexter.withContext(this).withPermissions(Manifest.permission.READ_EXTERNAL_STORAGE
```

```
, Manifest.permission.RECORD_AUDIO)
    .withListener(new MultiplePermissionsListener() {
        @Override
        public void onPermissionsChecked(MultiplePermissionsReport
multiplePermissionsReport) {
            displaySongs();
        }

        @Override
        public void
onPermissionRationaleShouldBeShown(List<PermissionRequest> list, PermissionToken
permissionToken) {
            permissionToken.continuePermissionRequest();
        }
    }).check();
}
public ArrayList<File> findSong (File file)
{
    ArrayList<File> arrayList = new ArrayList<>();
    File[] files = file.listFiles();
    for(File singlefile: files)
    {
        if(singlefile.isDirectory() && !singlefile.isHidden())
        {
            arrayList.addAll(findSong(singlefile));
        }
        else {
            if(singlefile.getName().endsWith(".mp3") ||
singlefile.getName().endsWith(".wav") || singlefile.getName().endsWith(".xoh"))
            {
                arrayList.add(singlefile);
            }
        }
    }
    return arrayList;
}
void displaySongs() {
    final ArrayList<File> mySongs =
findSong(Environment.getExternalStorageDirectory());
    items = new String[mySongs.size()];
    for (int i = 0; i < mySongs.size(); i++) {
        items[i] = mySongs.get(i).getName().toString().replace("mp3",
"").replace(".wav", "").replace(".xoh", "");
    }
    /*ArrayAdapter<String> myAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, items);
    listView.setAdapter(myAdapter);*/

    customAdapter customAdapter = new customAdapter();
    listView.setAdapter(customAdapter);

    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i,
long l) {
```

```
        String songName = (String) listView.getItemAtPosition(i);
        startActivity(new Intent(getApplicationContext(),
PlayerActivity.class)
                    .putExtra("songs", mySongs)
                    .putExtra("songname", songName)
                    .putExtra("pos", i));
    }
});
}
class customAdapter extends BaseAdapter {

    @Override
    public int getCount() {
        return items.length;
    }

    @Override
    public Object getItem(int i) {
        return null;
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        View myView = getLayoutInflater().inflate(R.layout.list_item, null);
        TextView textsong = myView.findViewById(R.id.txtsongname);
        textsong.setSelected(true);
        textsong.setText(items[i]);

        return myView;
    }
}
```


5.2 Libraries and Frameworks

5.2.1 XML

Android layouts are written in **extensible Markup Language**, also known as **XML**. Much like HTML (or) Hypertext *Markup Language*), XML is also a markup language. It was created as a standard way to encode data in internet-based applications.

However, *unlike* HTML, XML is case-sensitive, requires each tag is closed properly, and preserves whitespace. Android XML layouts are also part of a larger umbrella of Android files and components called resources. **Resources** are the additional files and static content an application needs, such as animations, colour schemes, layouts, menu layouts. Anatomy of Android XML Layouts is each layout file must contain one root element. Linear Layouts, Relative Layouts, and Frame Layouts may all be root elements. Other layouts may not be. All other XML elements will reside within this root object.

5.2.2 Java

Android App are mostly developed in JAVA language using Android SDK (Software Development Kit). Other languages like C, C++, Scala etc. can also be used for developing Android App, but JAVA is most preferred and mostly used programming language for Android App Development. If you are a beginner in Android then JAVA language and complete knowledge of OOPS concepts is the first thing you need to learn before beginning Android Development.

Java is the technology of choice for building applications using managed code that can execute on mobile devices. It is class based and object-oriented programming whose syntax is influenced by C++.

5.2.3 Animation

Animation is the process of adding a motion effect to any view, image, or text. With the help of an animation, you can add motion or can change the shape of a specific view. Animation in Android is generally used to give your UI a rich look and feel.

The basic Ways of animations are:

- Fade In Animation.
- Fade Out Animation.
- Cross Fading Animation.
- Blink Animation.
- Zoom In Animation.
- Zoom Out Animation.
- Rotate Animation.
- Move Animation.
- Slide Up Animation.
- Slide Down Animation.
- Bounce Animation.
- Sequential Animation.
- Together Animation.

In our project we have used **Bounce animation** basically bounce effect bounces on element, the element can be of any view such as Text View, Image View, Edit Text and more.

5.3 Functional Modules

The functional modules included in the project are listed below:

5.3.1 Music List module:

This module allows the user to select the songs from the displayed music playlist.

5.3.2 Play Music module:

This module allows the user to play the song by pressing the Play button. This is done after the selection of the song.

5.3.3 Pause Music module:

This module allows the user to pause the song by pressing the Pause button. This is done after the selection of the song.

5.3.4 Play Next Music module:

This module allows the user to play the next song in the list by pressing the next button.

5.3.4 Play Previous Music module:

This module allows the user to play the previous song in the list by pressing the previous button.

5.3.5 Forward Music module:

This module allows the user to forward the song by 10se by pressing the forward button.

5.3.6 Rewind Music module:

This module allows the user to rewind the song by 10secs by pressing the rewind button.

CHAPTER 6

TESTING AND RESULTS

6.1 Testing

Software testing determines the correctness, completeness, and quality of software being developed. Validation refers to the process of checking that the developed software meets the requirements specified by the user. The activities involved in the testing phase basically evaluate the capability of that system meets its requirements. The main objective of software testing is to detect errors in the software. Errors occur if some part of the developed system is found to be incorrect, incomplete or inconsistent. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects). It involves the execution of a software component or system to evaluate one or more properties of interest.

- Meet the requirements that guided its design and development.
- Responds correctly to all kinds of inputs.
- Performs its functions within an acceptable time.
- Is sufficiently usable.
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

6.2 Test Results

6.2.1 Test Case 1

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
1	Play	Play the song	Plays the song	Plays the song	Plays the selected song

6.2.2 Test Case 2

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
2	Pause	Pause the song	Pauses the song	Pauses the song	Pauses the selected song

6.2.3 Test Case 3

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
3	Play next song	Play next song	Plays next song	Plays next song	Plays next song in the row

6.2.3 Test Case 4

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
3	Play previous song	Play previous song	Plays previous song	Plays previous song	Plays previous song.

6.2.5 Test Case 5

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
4	Forward the song	Forward the song	Forwards the song	Forwards the song	Forwards the song by 10sec

6.2.6 Test Case 6

Sl no	Button Enabled	Voice Enabled	Expected output	Actual output	Remarks
5	Rewind the song	Rewind the song	Rewinds the song	Rewinds the song	Rewinds the song by 10sec

CHAPTER 7

CONCLUSION

The project work titled “Music Player Application” has been designed using Java and Xml (**extensible Markup Language**).

The System developed has proved to be user friendly and efficient in achieving basic goals. The system takes care of all the constraints which have been specified.

The system comprises of features like pause and play the music file, forward and rewind the file, and toggle between previous and next audio file.

The system is found to be really beneficial for the concerned aspects. Application developed is realistic and secure.

FUTURE ENHANCEMENT

The system comprises of features like pause and play the music file, forward and rewind the file, and toggle between previous and next audio file. We can enhance the music player further with providing options

- To include an option for the user to create playlists of their favorite songs.
- To improve the audio quality.
- To play the songs online.

REFERENCES

CHAPTER 1:

- [1]. [//www.ibm.com/cloud/learn/mobile-application-development-explained](https://www.ibm.com/cloud/learn/mobile-application-development-explained)
- [2]. https://en.m.wikipedia.org/wiki/Android_Studio

CHAPTER 2:

- [3]. <https://www.tutlane.com/tutorial/android/android-audio-media-player-with-examples>

CHAPTER 3:

- [4]. <https://medium.com/@vishwasng/non-functional-requirement-of-the-mobile-development-system-e0ed98f2a872>

CHAPTER 4:

- [5]. <https://www.google.com/amp/s/www.geeksforgeeks.org/software-engineering-architectural-design/amp/>
- [6]. <https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial>
- [7]. <https://www.google.com/amp/s/www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/amp/>

CHAPTER 5:

- [8]. [Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017](#)
- [9]. <https://developer.android.com/training/basics/firstapp/starting-activity>
- [10]. <https://youtu.be/3krzPY9SOoE>
- [11]. <https://www.learnhowtoprogram.com/android/introduction-to-android/introduction-to-xml-and-android-layouts>
- [12]. <https://www.javatpoint.com/java-tutorial>
- [13]. <https://www.journaldev.com/9481/android-animation-example>

APPENDIX A

SNAPSHOTS

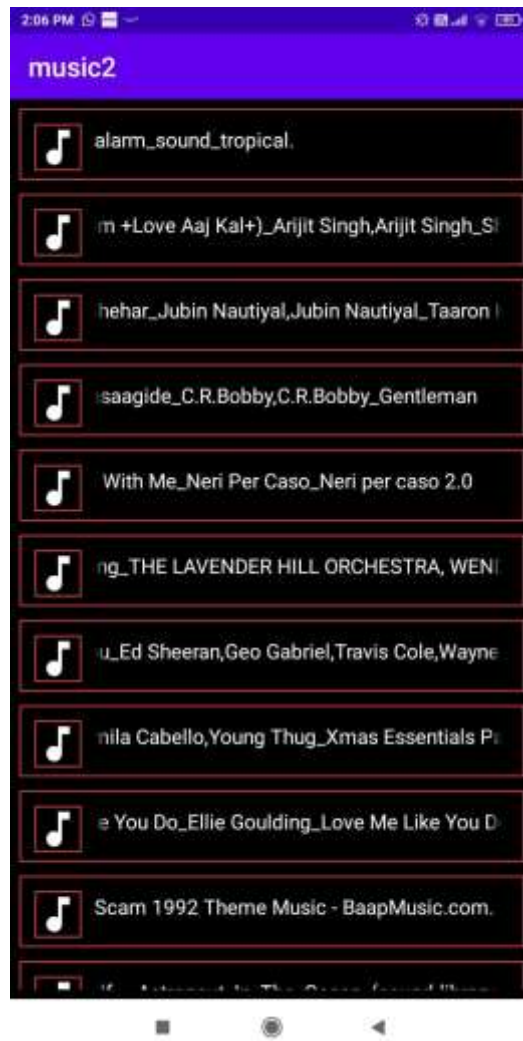


Fig: Playlist Screen/Launch Screen

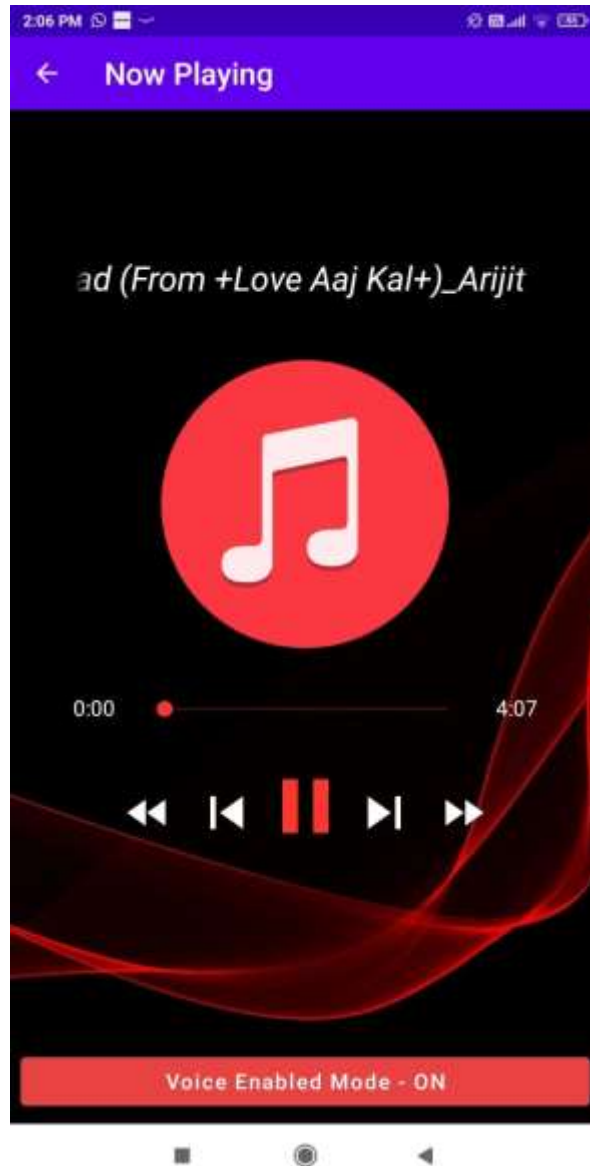


Fig: Song Playing Screen

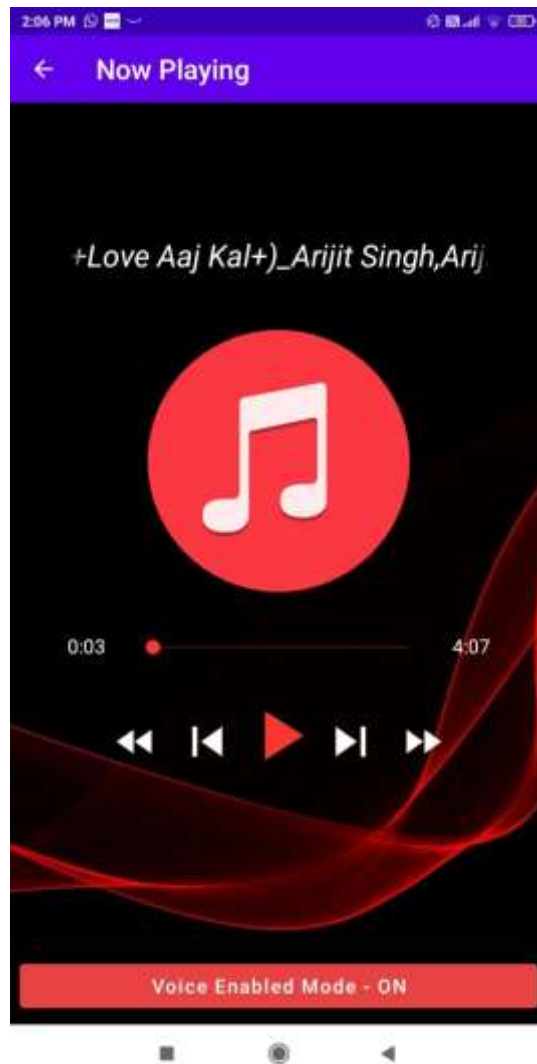


Fig: Song Paused Screen

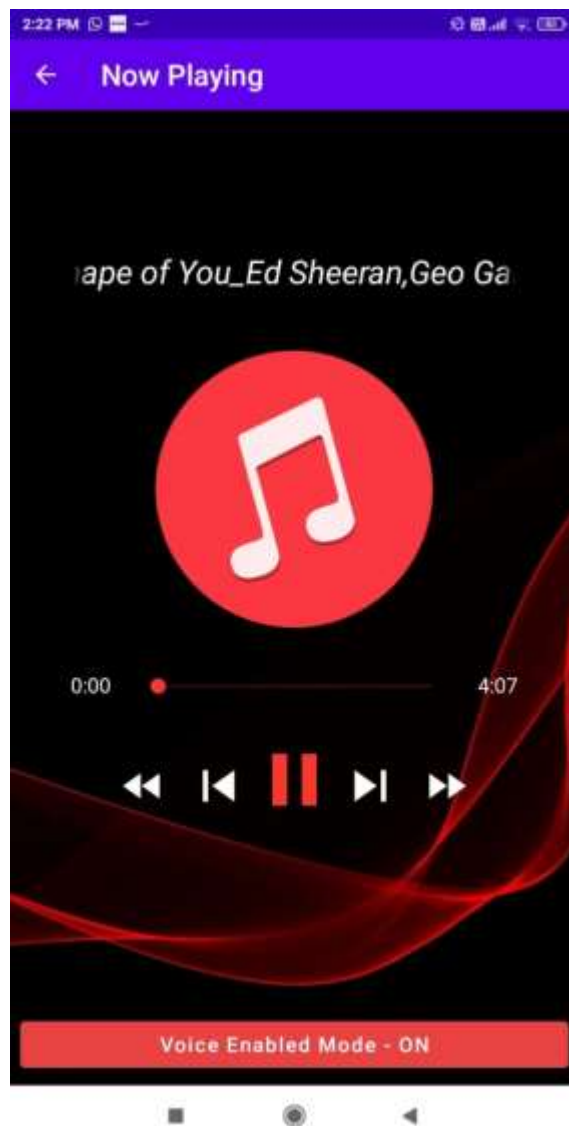


Fig: Next Song Played Screen

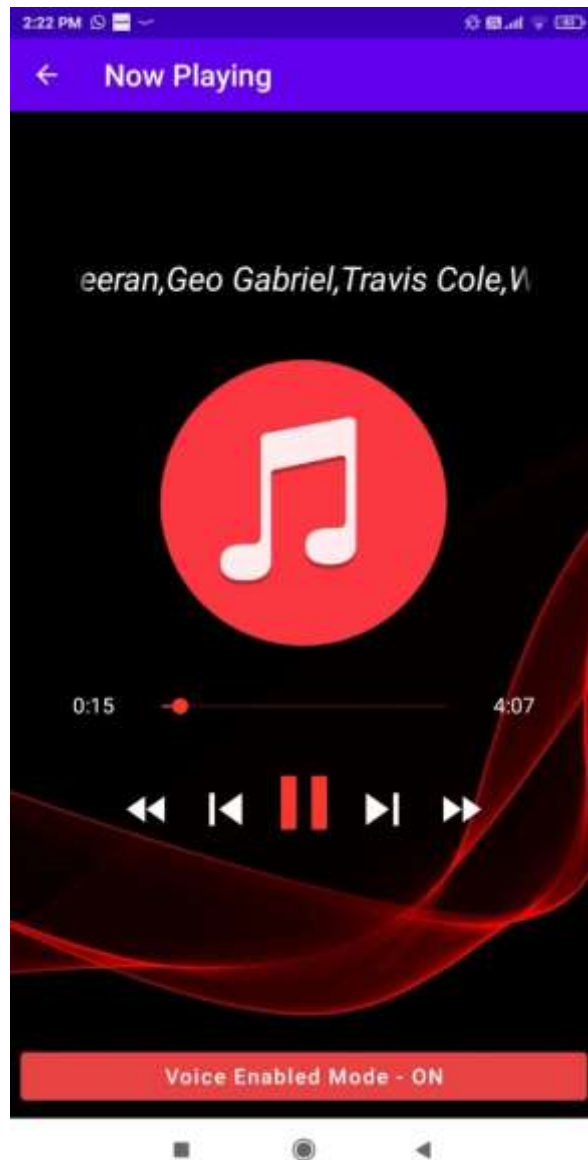


Fig: Song Forwarded by 15sec screen

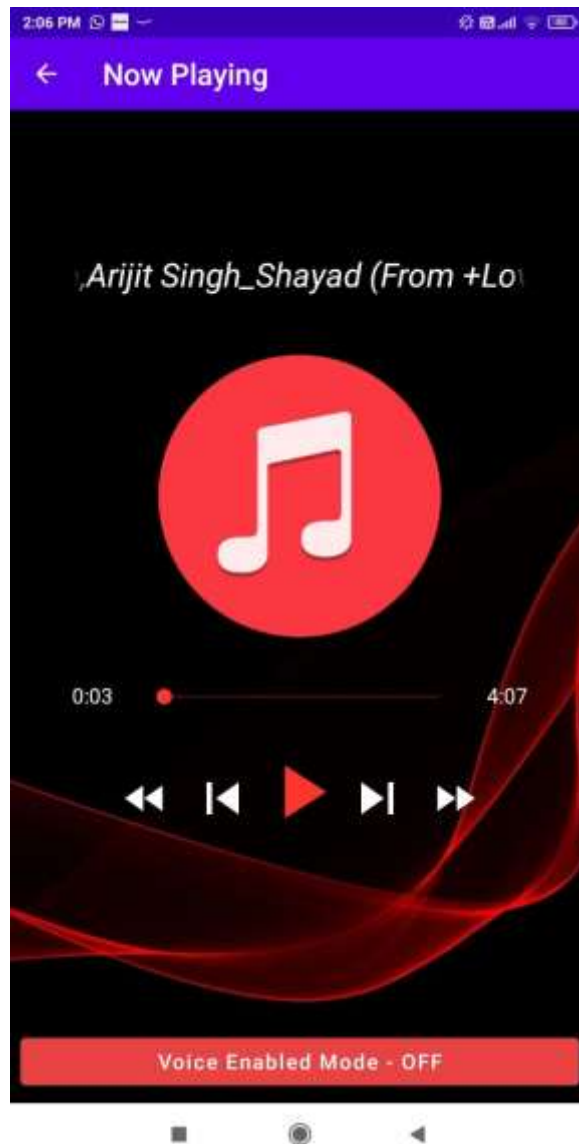


Fig: Voice Enabled Mode-OFF