Affiliated to GGSIP University, New Delhi
Approved by AICTE & Council of Architecture

Major Project Report

On

“Echo Music player Application”

Submitted in the partial fulfillment

for

The degree of Computer Science Engineering

Submitted by:

ANAND KUMAR

&

NIKITA SHRMA

Under the guidance of:

MR. NITESH BHATI (Assistant Professor)

COMPUTER ENGINEERING DEPARTMENT

DELHI TECHNICAL CAMPUS, GREATER NOIDA UP-201306

CERTIFICATE

This is to certify that project entitled "**Echo Music player Application**" being submitted by **ANAND KUMAR & NIKITA SHARMA** partial fulfillment for the Degree in Computer Engineering at Delhi Technical Campus, Greater Noida UP-201306.

Mr. NITESH BHATI
(Assistant Professor)
Computer Engineering

Delhi Technical Campus Knowledge Park-III Greater Noida 201306

ACKNOWLEDGEMENT

We take up this occasion of expressing of thankfulness towards all the persons who have been instrumental in the compilation.

We are feeling short of words of expressing ones feeling of gratitude towards my highly respective and esteemed guide,

ANAND KUMAR
&
NIKITA SHARMA

DECLARATION

I certify that

a) The work contained in this report is original and has been done by me under the guidance of my supervisor.
b) The work has not been submitted to any other institute for any degree or diploma.
c) I have followed the guidance provided by the institute in the preparing the report.
d) I have confirmed to the norms and guidelines given in ethical code of conduct of institutions.
e) Whenever I have used materials (data, theoretical, analysis, figures and text) from other sources. I have given credit to them by citing in the report and giving their details in the bibliography. Further I will take permission from the copyright owners of the sources, whenever necessary.

Name	**Roll No.**
ANAND KUMAR	**35118002716**
NIKITA SHARMA	**75118007217**

INDEX

ABSTRACT

This project work describe in detail, the project work undertaken by us during the final year of degree by us during the final year of degree at **DELHI TECHNICAL CAMPUS.** The contents of this report include a complete description of the **'Echo Music player Application'**

The purpose to develop a music player application is to develop a smart music player application that provided smart feature Shake to change song'.basically it an offline music player. The app should be able to fetch and play .mp3 and .wav files.music player with the following functionality:

1.A Splash screen (gradient background and app logo in center)

2.A Navigation drawer with app logo section at the top along with links to 'All Songs', 'Favorites', 'Settings' and 'About Us'.

3.An 'All songs' screen (where of list all the tracks fetched from offline storage are displayed and user can sort the tracks by name or recently added).

4.A 'Favorites' screen (where list of all the favorite
songs are displayed)

5.A 'Settings' screen (where the 'Shake to change song' feature can be enabled or disabled)

An 'About us' screen (where we will display information about the app developer and the app version)

INTRODUCTION

PURPOSE

The purpose to develop a music player application is to develop a smart music player application that provided smart feature Shake to change song'.basically it an offline music player. The app should be able to fetch and play .mp3 and .wav files.music player with the following functionality:

FEATURES:

1.A Splash screen (gradient background and app logo in center)

2.A Navigation drawer with app logo section at the top along with links to 'All Songs', 'Favorites', 'Settings' and 'About Us'.

3.An 'All songs' screen (where of list all the tracks fetched from offline storage are displayed and user can sort the tracks by name or recently added). This will the home screen of the app.

4.The app should be able to fetch and play .mp3 and .wav files.

5.A 'Favorites' screen (where list of all the favorite songs are displayed)

6.A 'Settings' screen (where the 'Shake to change song' feature can be enabled or disabled)

7.An 'About us' screen (where we will display information about the app developer and the app version)

8.A 'Now playing' screen with following features:

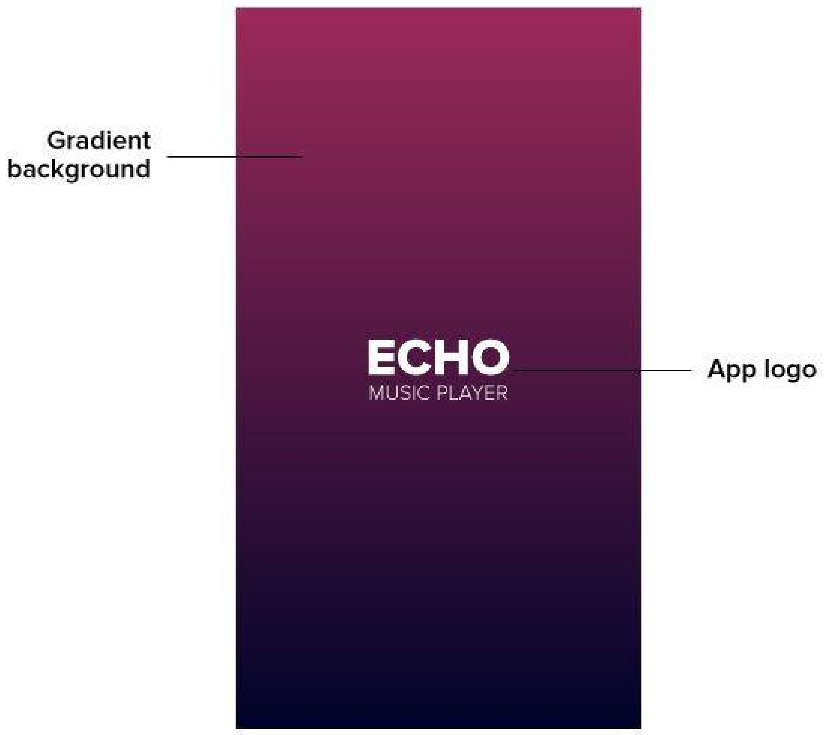
- Track title and track artist
- Play / Pause button
- Next button
- Previous button
- Shuffle button
- Loop button
- Seek bar
- Mark track as favorite or unfavorite it
- Third party visualiser in upper half background
- A 'Back to list' button in the header which should take the user to the screen he came from (kind of like back button behaviour).
- Shake to change song

9.A 'Now playing' bar at the bottom with name of the track playing and play or pause feature. This would appear if the user has moved from 'Now playing' screen to 'All songs' screen or 'Favorites' screen without pausing the track.

10.Background play. The app will continue playing the track if the app gets closed (not killed) without the music being paused.

MODULES WITH DIAGRAM

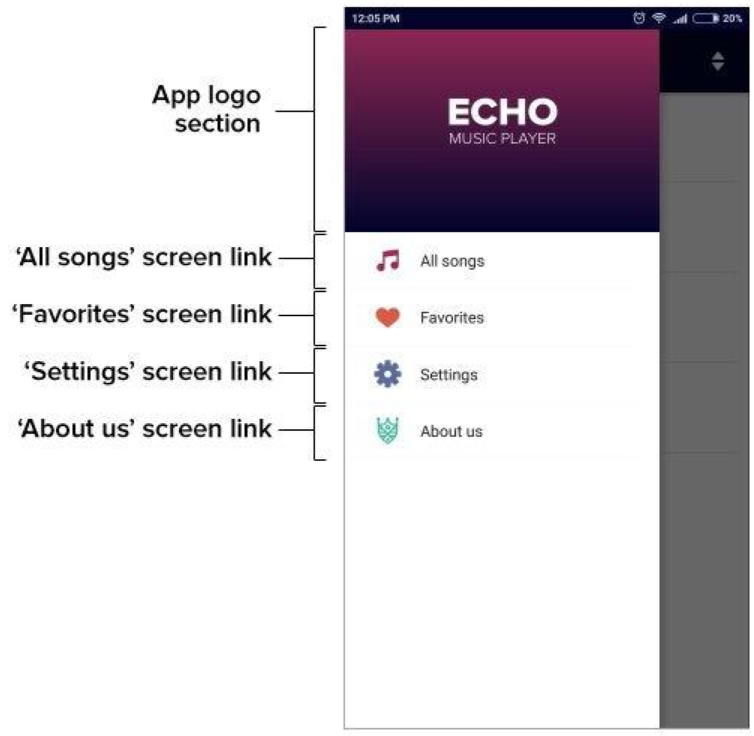
1. Splash screen



This would be the first screen that gets displayed when a user opens the app. It would have a linear gradient background and the app logo in the center of the screen. The user will see the screen for 1 second and then the home screen will pop up.

2:-Navigation drawer

The navigation drawer is needed so that the user can navigate to different screens within the app. The user should be able to access the navigation drawer on all the app screens by clicking hamburger button on the left side of the header or by swiping right from the left edge. It would have an app logo section at the top with gradient in the background and app logo in center (similar to Splash screen but smaller). Below the app logo section, there would links to 'All songs', 'Favorites', 'Settings' and 'About us' screen in a list style. Clicking on any link will open the corresponding screen.

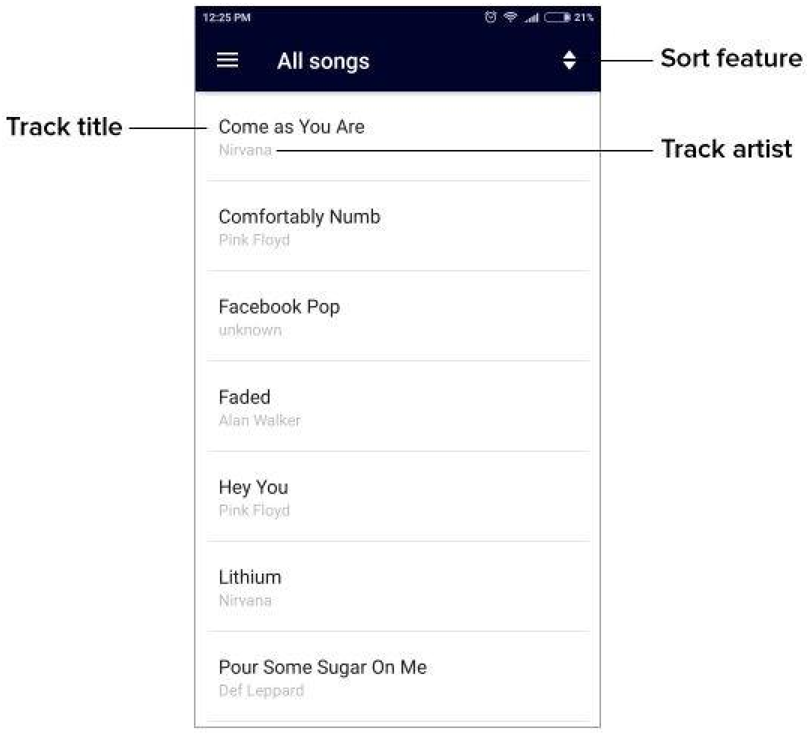


3.All songs' screen

'All songs' screen will be the home screen of the app, meaning when the app is launched, user will see the 'All songs' screen after the splash screen.

Once the app is launched, all the tracks (.mp3 and .wav) will be fetched from the offline storage and will be displayed on the 'All songs' screen in a list view. By default, all the tracks would be sorted by name. For each track in the list, the user should see the title of the track and the artist of the track. If there is no title, the track name will be displayed in place of the title. If there is no artist, 'unknown' will be displayed in place of the artist. There would also be a line separating each track.

When the user clicks on any track, the 'Now playing' screen should open and



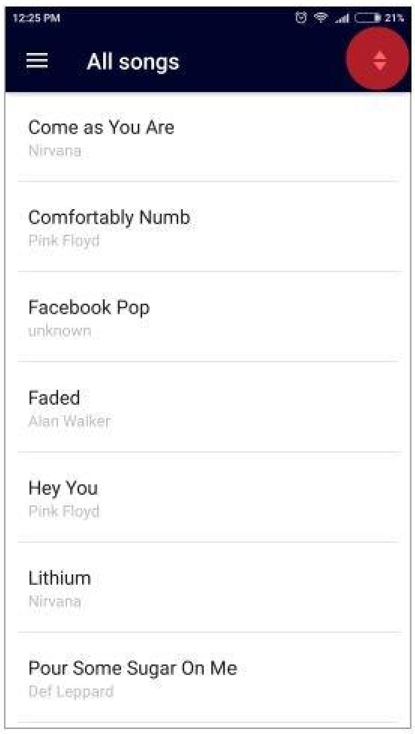
the track should start to play.

4.Sort feature:

The ‘All songs’ screen will have a Sort feature. Using the Sort feature, the user should be able to sort the tracks by name or by recently added. A sort icon will be there on the right side of the header, when clicked on, then a dropdown will appear with two options:

- By recently added
- By name

If the user clicks on ‘By recently added’, all the tracks would get sorted by recency with most recently added track on the top. ‘By name’, all the tracks would get sorted in alphabetical order (or in this order: track title starting with symbols, track title starting with numbers and then track title starting with letters).



Come as You Are
Nirvana

Comfortably Numb
Pink Floyd

Facebook Pop
unknown

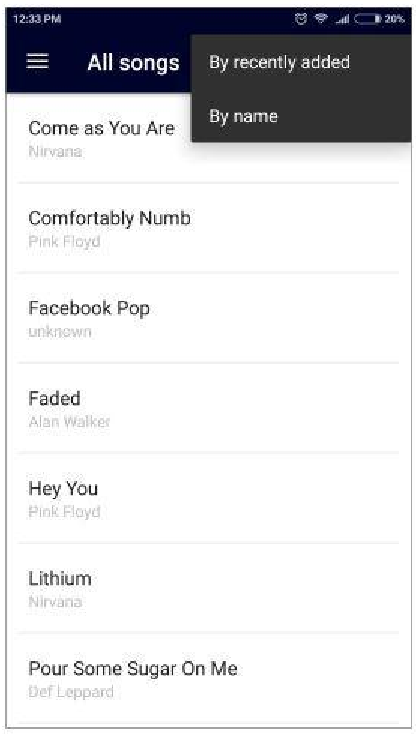
Faded
Alan Walker

Hey You
Pink Floyd

Lithium
Nirvana

Pour Some Sugar On Me
Def Leppard

$\longrightarrow$



Come as You Are
Nirvana

Comfortably Numb
Pink Floyd

Facebook Pop
unknown

Faded
Alan Walker

Hey You
Pink Floyd

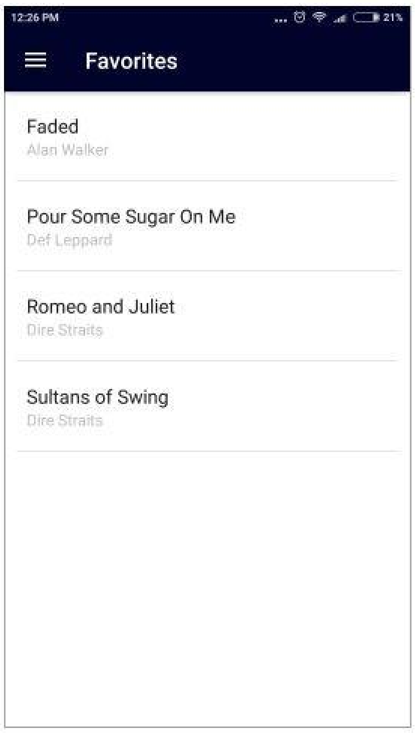
Lithium
Nirvana

Pour Some Sugar On Me
Def Leppard

5.‘Favorites’ screen

The ‘Favorites’ screen resembles the ‘All songs’ screen. It will display all the tracks that have been marked favorite by the user. The user would be able to mark a track favorite or unfavorite it only on the ‘Now playing’ screen. Ensure that if the user marks a track favorite and then later deletes it from the offline storage, the track shouldn’t appear on the ‘Favorites’ screen.

If the user has no favorite tracks, there would be a message in the center on the screen saying, “You haven’t got any favorites yet!”.

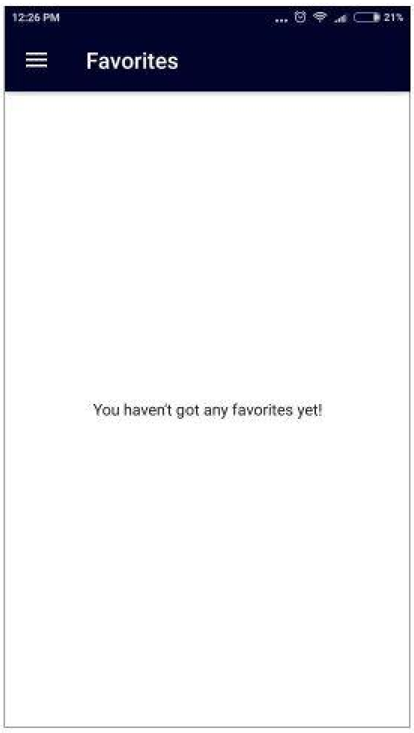


Faded
Alan Walker

Pour Some Sugar On Me
Def Leppard

Romeo and Juliet
Dire Straits

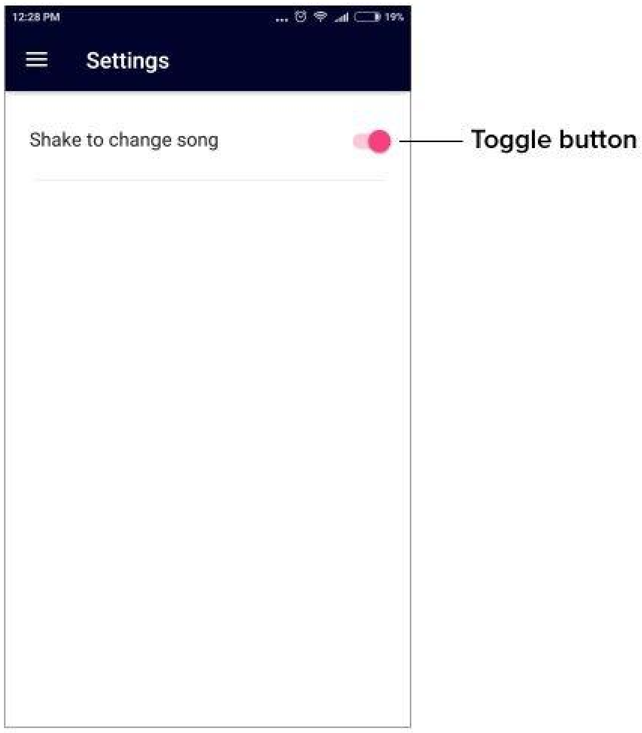
Sultans of Swing
Dire Straits



You haven't got any favorites yet!

6.‘Settings’ screen

The ‘Settings’ screen will have the option to enable or disable ‘Shake to change song’ feature using the toggle button. The default state of ‘Shake to change song’ feature would be ‘disabled’. If the user enables the ‘Shake to change song’ feature, the app should remember it, so when the user kills the app and launches it again, the ‘Shake to change song’ should stay enabled.



Shake to change song

7.‘About us’ screen

The ‘About us’ screen will display the information about the app developer and the app version. As part of the app developer information, there would developer’s photograph and a few lines about him/her.

8.'Now playing' screen

The 'Now playing' screen is the actual music player in the app. It houses all the elements which let the user interact with the current playing track or the next/previous track. It will open when the user clicks on a track on the 'All songs' screen or the 'Favorites' screen.
It would have following features and functionalities:

- **Track title and track artist**
  The track title and track artist would be displayed in the center of bottom half of the screen. If the track has no title, track name will be displayed. If the track has no artist, 'unknown' will be displayed in its place. If the track title or name is too long, an ellipsis will be added to the title or name.
- **Play/Pause button**
  As the names suggest, this button will be used to play or pause a track. When a track is playing in the app, this button will become Pause button. When a track is paused in the app, this button will become Play button. When a user clicks on the Pause button, the track will pause and when the Play button is clicked, the track will start playing from the same place where it was paused.
- **Next button**
  This button will allow the user to play the next track. When the Next button is clicked, the consecutive track in the list (from which the 'Now playing' screen was triggered) will start playing.
- **Previous button**
  This button will allow the user to play the previous track. When the Previous button is clicked, the consecutive track in the list (from which the 'Now playing' screen was triggered) will start

playing.

- **<u>Shuffle button</u>**

  The default state of Shuffle button would be ‘switched off’ (white). When the Shuffle button is toggled (switched on) by the user, the button would turn yellow indicating that Shuffle feature has been turned on. When the Shuffle feature is on, the player would randomly choose a track (different from the one currently playing) from the list (from which the ‘Now playing’ screen was triggered) and play it when the next button is clicked or when the current track ends. When the Shuffle button is toggled again (switched off), the button would again turn white indicating that Shuffle feature has been turned off. When Shuffle feature is white or switched off, the player would play the consecutive track in the list when the next button is clicked or when the current track ends.

  If the user switches the Shuffle feature on, the app should remember it, so when the user kills the app and launches it again, the Shuffle feature stays on.

  If the Shuffle feature is on and the Loop feature is turned on, Shuffle feature would move back to its default state.

  The Shuffle feature and the Loop feature can’t be in ‘switched on’ (yellow) state simultaneously.

- **<u>Loop button</u>**

  The default state of the Loop button would be ‘switched off’ (white). When the Loop button is toggled (switched on) by the user, the button would turn yellow indicating that Loop feature has been turned on. When the loop feature is on, the player would play the same track again when the track ends. The loop button won’t affect the behaviour of Next button. When the Loop button is toggled again (switched off), the button would again turn white indicating that Loop feature has been turned off.

If the user switches the Loop feature on, the app should remember it, so when the user kills the app and launches it again, the Loop feature stays on.

If the Loop feature is on and the Shuffle feature is turned on, Loop feature would move back to its default state.

The Loop feature and the Shuffle feature can't be in 'switched on' (yellow) state simultaneously.

- **Seek bar**

  This screen features a *seek-bar* which displays the track progress throughout the track's lifetime.A user can click on the seekbar to skip in between the track or simply to drag the controller to reach a certain part of the track.

- **'Mark as favourite' button**

  Clicking this button adds the current track to the favorites list, the button then turns red indicating that the track has been added to the favorites list. A toast message is displayed on the screen saying "Added to favorites".

  Clicking this button again will remove the track from the favourites list, the button then turns back to white indicating that the track has been removed from the favorites list. A toast message is displayed on the screen saying "Removed from favorites".

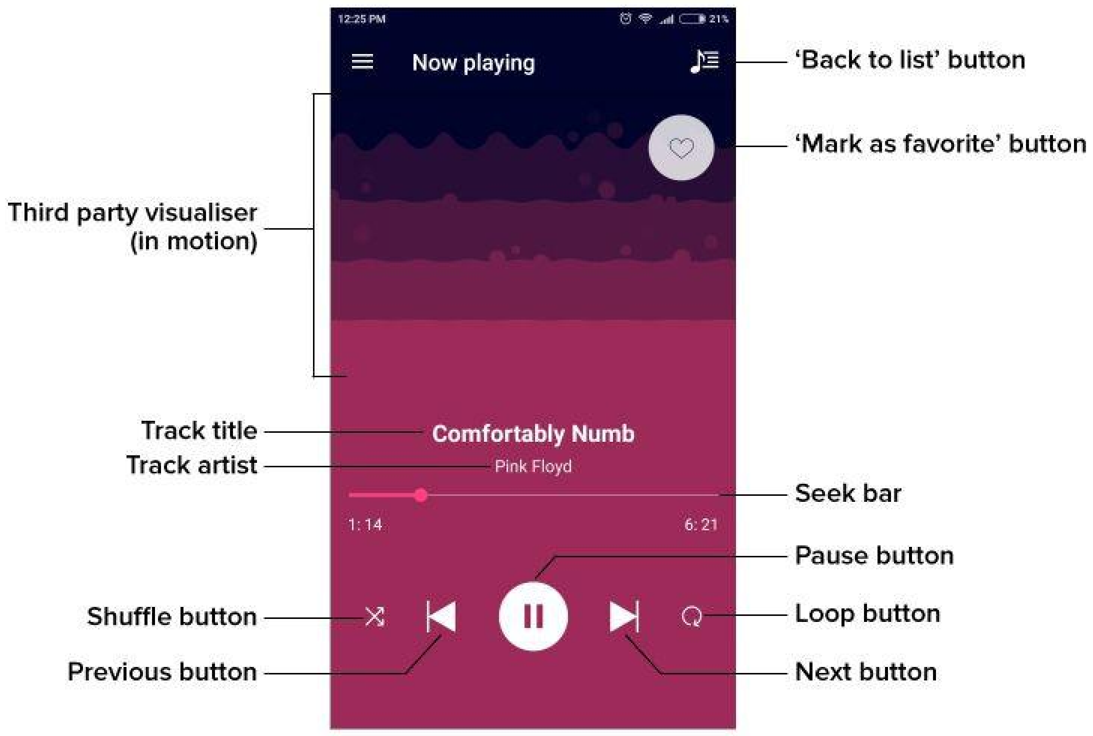
  The default state of the 'Mark as favorite' button is white, that means, initially there would be no tracks in the favorites list.

- **Third party visualiser**

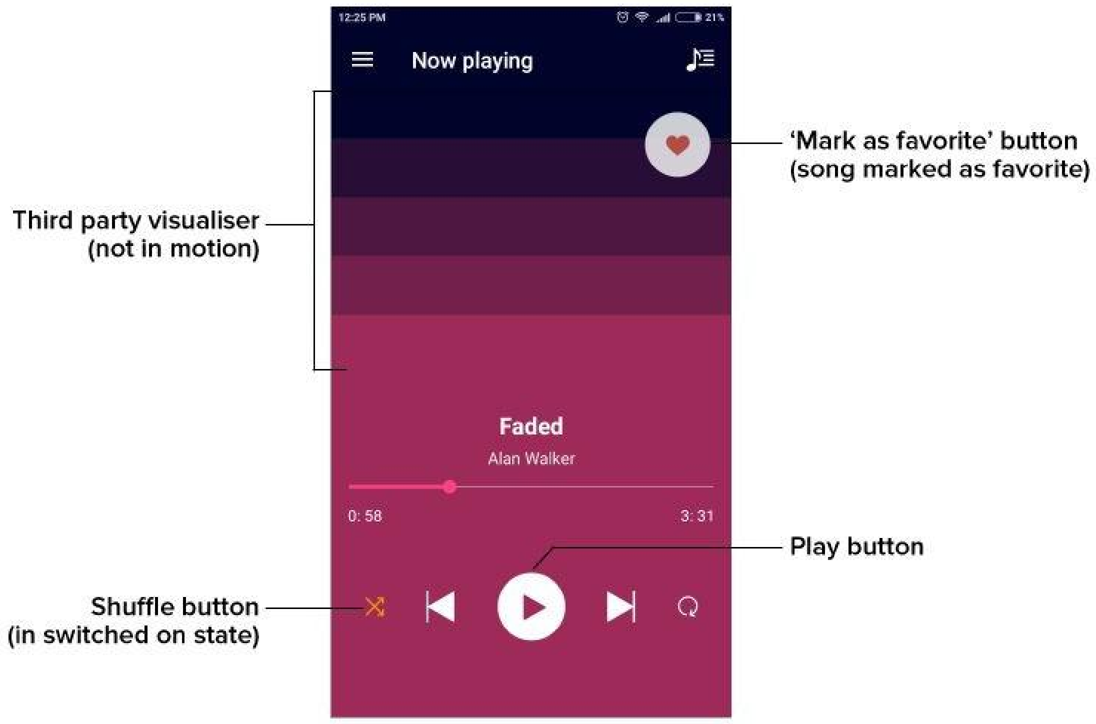
  The 'Now playing' screen would have a 4 bar visualiser in the upper half of the screen. As expected, the visualiser would move in the rhythm of the music. The visualiser's motion would be volume sensitive meaning that if you turn down the volume, the visualiser will also tone down its motion and vice-versa. The visualiser should start moving once a track is played and should stop moving when a track is paused.

- **<u>‘Back to list’ button</u>**
  The ‘Back to list’ button would take the user to the screen he came from. For ex: if a user clicked on a track on the ‘All songs’ screen and lands on the ‘Now playing’ screen, the ‘Back to list’ button should take the user back to the ‘All songs’ screen. This button would be placed in the right side of the header.
- **<u>Shake to change song</u>**
  As the name suggests, this feature would allow the user to change the track just by shaking his/her phone. We would use the accelerometer on the mobile phones to make this feature work.

Now playing screen (track playing)

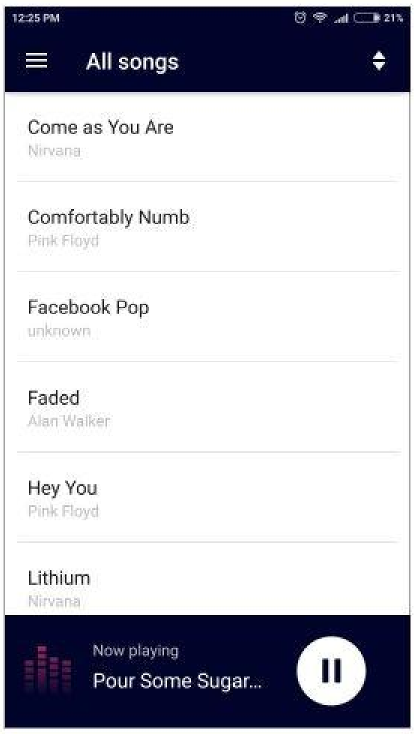


Now playing screen (track paused)


12:25 PM
21%
Now playing
'Mark as favorite' button (song marked as favorite)
Third party visualiser (not in motion)
Faded
Alan Walker
0: 58
3: 31
Play button
Shuffle button (in switched on state)

‘Now playing’ bar

The ‘All songs’ screen and the ‘Favorites’ screen will have a ‘Now playing’ bar at the bottom if the app is playing a track. This bar would display the title of the track playing and play or pause feature. It would also have an image of some equaliser bars in the left side and a static text above the track title name saying “Now playing”. If the track title is too long to be displayed in one line, an ellipsis would appear in the track title. The Play/Pause button would work same as it did on the ‘Now playing’ screen.


12:25 PM

All songs

Come as You Are
Nirvana

Comfortably Numb
Pink Floyd

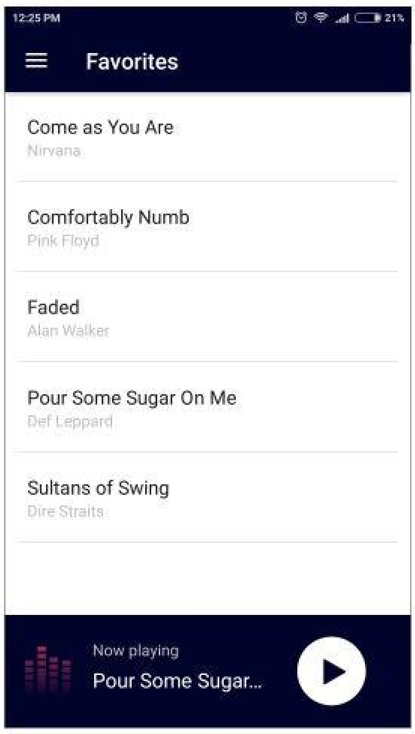
Facebook Pop
unknown

Faded
Alan Walker

Hey You
Pink Floyd

Lithium
Nirvana

Now playing
Pour Some Sugar...


12:25 PM

Favorites

Come as You Are
Nirvana

Comfortably Numb
Pink Floyd

Faded
Alan Walker

Pour Some Sugar On Me
Def Leppard

Sultans of Swing
Dire Straits

Now playing
Pour Some Sugar...

Background play feature

Just like any other music player app, this app will allow the user to listen to music even when the app is running in the background (when a user plays a track in the app and switches to a different app from notification bar or recent apps section or simply goes to the home screen, the app will keep running in the background and keep playing the track.)

If there is an incoming/outgoing call while the app is playing a track, the app should pause the track. As it is a basic music player, the app wouldn’t pause a track for any other case. It means that if user starts another music player or video player or any such app while our app is playing a track in the background, our app wouldn’t pause the track and the other app can play music over our app.

IMPLEMENTATION

HARDWARE AND SOFTWARE TO BE USED:-

➢ **Hardware Specifications**

HARDWARE	SPECIFICATIONS
Processor speed	200 MHz processor
RAM	32 MB
Hard Disk	32 MB

➢ **Software Specifications**

FRONT END	UI of android application by Using XML
BACK END	Kotlin programming Language ,SQLITE
OPERATING SYSTEM	Android Os Ice Cream Sandwich 4.00 to higher version .
API	MediaPlayer API

EXPLANATION:-

1.UI

1.Introduction to android:

Android is an operating system designed primarily for touchscreen mobile devices Such as smart phones and tables.

2.UI(user interface in android)

Arranging the info + adding the style= is called user interface.
A user interface can be defined as the means by which the user and the system interact.

3.How do we create UI in android?

We create UI in android using XML(Extensible markup language)

4.What are layout and widgets?

Container is layout and all its element is widgets.

5.What are tags?

Tags are the way to define layouts and weight in XML

Example:

```
<LinearLayout>-------------   opening tag
        Content 1
        Content 2
        Content3
        ……….
        ………….
        ………….
        …………
</LinearLayout>-------------   closing tag
```

6.What is empty tag?

Without any content is called empty tag.

Note :

1.Xml tags are case-senstive.

2.Xml tags must be closed In an approprite order.

7.Linear Layouts: LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.

All children of a LinearLayout are stacked one after the other, so a vertical list will only have one child per row, no matter how wide they are, and a horizontal list will only be one row high (the height of the tallest child, plus padding). A LinearLayout respects margins between children and the gravity (right, center, or left alignment) of each child.

Setting Up the first layout and adding images

..<ImageView>

8.What are attribute?

Attribute are used to describe the properties of an XML tag Orientation of a LinearLayout.

You can add wieght and element on after another vertically or Horizonally.Add

image,text and button in your application:
You need to go into an specific folder called
Drawable folder for adding image.
Fixing the values of the image and adding text

9.DP vs SP

.<TextView>

What is Pixel Density?

Pixel density is the number of pixels in a given area of a digital screen. We usually measure it in Pixels per inch or PPI.

Type of pixel density
1.High Pixel Density
Screen
2.Medium pixel density
Screen
3.Low pixel density

Density Independent Pixels(Dp):

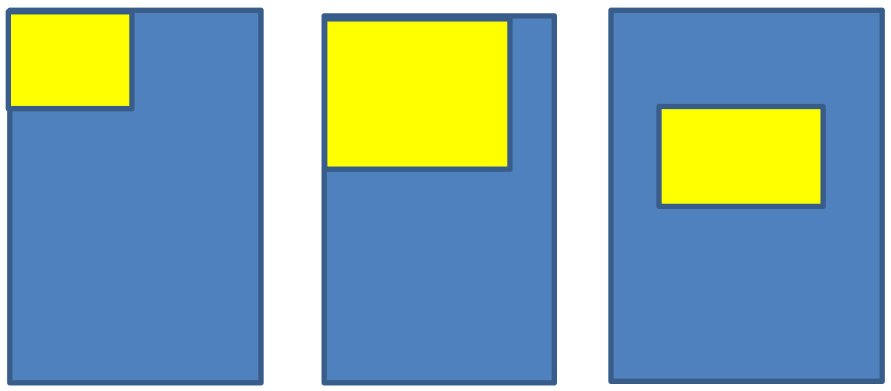
It is a unit we use to define height, weight and similar Properties in android.It occupies similar space in android.

10.Adding button and styling them

<Button>

11.Adding spaces around UI elements performed by adding two attribute

Margin:adding spce around the element.
Padding:add space around the weight.



Layout padding :10dp margin: 10dp

Note:Padding adding space around the content but inside the element.Margin adding space around the element.

12.Relative Layout:

Just like linear layout,relative layout is a type of layout and it is used to make complex UI.RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center). A RelativeLayout is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance. If you find yourself using several nested LinearLayout groups, you may be able to replace them with a single RelativeLayout.

<u>2.Kotlin</u>

1.Introduction to Kotlin?

- Kotlin is programming language that runs on the Java Virtual machine or JVM.

- JetBrains is a software development company which makes tools mainly for software Developers and product manager.

● Android studio is developed by jetBrain.JetBrains released the first officially Stable version of Kotlin ,kotlin v1.0.Google announced first-class support For kotlin on android.

Why ????

If kotlin is new to android , which language was being used to build apps on Android before Kotlin.

->JAVA

Java has few limitation But kotlin have lot of feature

2.Fundamental of kotlin

Where and how do we write Kotlin?

MainActivity.kt file

Note:

.kt is the file extention of the kotlin

3.General variable or var

Var variablename = variablevalue // whoes value may change may Latter.

4.Constant variable or 'val'

Val variablename = variablevalue// whoes value would not change.

.Example:

```
fun main() {
val characterOneName = "keto"
var characterOneAge = 18
val characterTwoName = "Lino"
var characterTwoAge = 18
println("we are learning kotlin!!!")
}
```

6.Data type

What is data type?

A data type is a classification that specifies which type of value a variable has.

String Data type

String is a combination of characters. These characters can be letters,numbers ,spaces or different symbols.And string is written between two double code.

For example

“ we ate 3 apple ”

Integer Data Type

Integer is any zero,postive or negative number without any decimal.

Example

```
fun main() {
   //String data type
   val characterOneName:String = "keto"
   //integer data type
   var characterOneAge:Int = 18

   val characterTwoName:String = "Lino"
   var characterTwoAge:Int = 18
   //double data type
   var aDecimalValue:Double = 22.3
   //boolean data type
   var booleanValueOne: Boolean = true
   var booleanValueTwo: Boolean = false

   println("we are learning kotlin!!!")
}
```

Boolean data type:
Boolean is data type which can have only 2 value
True or False
You can compare it simple electric switch It can either in on or off state.
Boolean is same way depending what condition you have you can set in your program.
Variables and data types.
Mutable Variables: The variables which can be edited at a later stages in our program.
e.g. :
```
var x = 5 // `Int` type is inferred
```
Immutable variables: The variables which can be initialised only once can whose values cannot change. e.g. :
```
val a: Int = 1 // immediate assignment
val b = 2 // `Int` type is inferred
```
Comments: The comments are the lines which are for the developers only. The compiler does not compiles these lines of code. The main usage of these lines are to make the code readable and easily understandable. e.g :
```
// This is an end-of-line comment
/* This is a block comment
on multiple lines. */
```
Conditional Statements and Operators
Conditional statements explanation:
Assignment:Build a magical lock spell which will allow only you to enter your room.
What is conditional statement:
Conditional statements are features of programming languages which perform certain things when a condition is met.
```
if (condition){
do this}else {do this}

If(given password = actual password){
Open the door}else{
don't open the door}
```

Example:

```
var characterOneAge:Int = 18

    val characterTwoName:String = "Lino"
    var characterTwoAge:Int = 18
    // This is the code lock on the door
    var password: String = "Alohomora"
    var whatIsThePassword: String  = "let me opend"
    if(whatIsThePassword.equals(password)){
        println("Door opened . welcome!")
    }else{
        println("Incorrect password can not open the door")
    }


    println("we are learning kotlin!!!")
}
```

# The other ways to write the if statement can be :

```
val max = if (a > b) a else b
```

-> Also if you need to put a statement also then we can do the following:

```
val max = if (a > b) {
print("Choose a")
a
} else {
print("Choose b")
b
}
```

Operators:

An operator in a programming language is symbol that tells the computer to perform Specific mathematical,Relational or logical operation and produce final result.

Assignment operator

= so this is assignment operator and its job is to assign value into the variable.

Or operator

||

Here are some operators you should keep in mind:
The Arithmetic Operators
1. '+' - Addition
2. '-' - Subtraction
3. '*' - Multiplication
4. '/' - Division
5. '%' - Modulus (gives the remainder when two operands are divided)
6. '++' - Increment(Increases the value of integer by 1)
7. '--' - Decrement(Decreases the value of integer by 1)
The Relational Operators
1. '==' - Equal to
2. '!=' - Not equal to
3. '>' - Greater than
4. '<' - Less than
5. '>=' - Greater than or equal to
6. '<=' - Less than or equal to
The Logical Operators
1. '&&' - And operator
2. '||' - Or operator

Examples:
```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */

fun main() {
   //String data type
   val characterOneName:String = "keto"
   var characterOneAge:Int = 18

   val characterTwoName:String = "Lino"
   var characterTwoAge:Int = 18
   // This is the code lock on the door

   var kotoPassword:String = "kotoRock123"
   var linoPassword:String = "LinoIsTheBest007"
```

```
    var whatIsThePassword: String  = "kotoRock123"
    if(whatIsThePassword.equals(kotoPassword)||
whatIsThePassword.equals(linoPassword)){
        println("Door opened . welcome!")
    }else{
        println("Incorrect password can not open the door")
    }


    println("we are learning kotlin!!!")
}
```

When statement

Day	class1	Class2
Monday	History of Magic	Flying
Tuesday	Portions	spell Making
Wednesday	Flying	Dark Arts
Thursday	Astronomy	Music
Friday	Portions	spell Making
Saturday	History of Magic	Flying
Sunday	NoClass	No class

```
When (variable ){
    Variablevalue1 -> { do this}
Variablevalue2 -> { do this}
Variablevalue3-> {do this }
Variablevalue4-> {do this}
Else->{do this}
}
```

Example

```
println("we are learning kotlin!!!")
    // this the code for the bag-pack program.
    var day: String = "Monday"
    when(day){
        "Monday" -> { println("Books of History of Magic")}
        "Tuesday" -> { println("Books of History of Magic")}
        "Wednesday" -> { println("Books of History of Magic")}
        "Thursday" -> { println("Books of History of Magic")}
        "Friday" -> { println("Books of History of Magic")}
        "Saturday" -> { println("Books of History of Magic")}
        "Sunday" -> { println("Books of History of Magic")}

    else -> {println("Incorrect value of the day")}
    }

}
```

# The when statement can also be used for checking a value in or not in a range. E.g. :

```
when (x) {
in 1..10 -> print("x is in the range")
in validNumbers -> print("x is valid")
!in 10..20 -> print("x is outside the range")
else -> print("none of the above")
}
```

# The when statement can also be used as a replacement for if-else chain E.g.:

```
when {
x.isOdd() -> print("x is odd")
x.isEven() -> print("x is even")
else -> print("x is funny")
}
```

Loop in Kotlin…

What is while Loops

Loops are features of programming languages which let you repeat a set of instructions Over and over until a certain condition met.

For example

Dice

Roll a dice until a certain condition is met.

To build this repetition step we will use

While loop.

//This is the code for line repetition program

```
    var i =1
    while(i<=100){
        println("I'll never be late again")
        i =i+1;
    }
}
```

Explanation:
i = 1
I is less than or equal to 100 (condition is true)
{
Println("I'll never be late again")
i =i+1
}
i = 2
I is less than or equal to 100 (condition is true)
{
Println("I'll never be late again")
i =i+1
}
i = 100
I is less than or equal to 100 (condition is true)
{
Println("I'll never be late again")
i =i+1
}
i = 101

I is less NOT than or equal to 100(condition is false)
Loop ends
Loops
For loops

```
 for(item in 1..100){
        println("I'll never be late again")
```

```
    }
    //item= 1,item=2,item=3,item=4,item=4,item=5....item=100
```
-> Another loop which is available is the do-while loop. It executes the expression at least once regardless of the condition provided to be true. For e.g.
```
do {
val y = retrieveData()
} while (y != null)
```
-> If you want the counter to increment with a different value other than 1 then you need to give the "step" size in the for loop
```
for (i in 1..4 step 2) print(i) // prints "1 3"
```
-> And if you want a decrementing counter we use “downTo ” keyword. E.g.:
```
for (i in 4 downTo 1 step 2) print(i) // prints "4 2"
```

Array and Indexing
Array is a type of data structure used to store different elements in it. Usually we store similar type of elements in an array.

Explanation
Dragon wars
Registration open

```
 var teamMightwarriors:Array<String> = arrayOf("joey| 17yo|Good with position",
                             "Sam| 18yo| Good with flying",
                            "Amy | 17yo |Good with positions",
                             "Tim|18yo | Good with making spells",
                             "Sara| 18yo| Good with flying")
  // var primeNumbers:Array<Int> = arrayOf(2,3,5,7,11)
   //var randomArray = arrayOf("Magic",23,67,.3,true)
   var teamMightwarriors2:Array<String> = arrayOf("joey| 17yo|Good with position",
                             "Sam| 18yo| Good with flying",
                            "Amy | 17yo |Good with positions",
                             "Tim|18yo | Good with making spells",
                             "Sara| 18yo| Good with flying")
   var teamMightwarriors3:Array<String> = arrayOf("joey| 17yo|Good with position",
                             "Sam| 18yo| Good with flying",
                            "Amy | 17yo |Good with positions",
                             "Tim|18yo | Good with making spells",
                          "Sara| 18yo| Good with flying")
   var teamInvincibles:Array<String> = arrayOf("joey| 17yo|Good with position",
                          "Sam| 18yo| Good with flying",
                         "Amy | 17yo |Good with positions",
                          "Tim|18yo | Good with making spells",
                          "Sara| 18yo| Good with flying")
   var teamWonderWomen:Array<String> = arrayOf("joey| 17yo|Good with position",
                          "Sam| 18yo| Good with flying",
                         "Amy | 17yo |Good with positions",
                          "Tim|18yo | Good with making spells",
                          "Sara| 18yo| Good with flying")

   for (member in teamMightwarriors)
    {
     println("Member")
    }
}
```

Indexing

What is indexing?

Indexing is simply assigning a sequence number in array.

Example

Roll Number	Array Index
Akansha	0.Element1
Akash	1.Element2
Aaron	2.Elemnet3
Barkha	…….
Chetan	
Daniel	
Deepak	

Indexing is very help full when you want a specific data in a Array

# We can also print the each member in an array using array.indices. For e.g.:

```
var team = arrayOf("Member1 | Task1 | Quality1",
"Member2 | Task2 | Quality2",
"Member3 | Task3 | Quality3",
"Member4 | Task4 | Quality4",
"Member5 | Task5 | Quality5",
"Member6 | Task6 | Quality6")
for(i in team.indices){
println(team[i])
} // prints each member in a new line
```

Data structure

.**ArrayList**

arrayList Is an extention of array and lets you add,delete and do more with element of an array!
It Is comanly reffer to as list.

Conclusion

>Both Array and arrayLIst are data structures which let you store different element
.Array are non -resizable ,which means you cannot add or delete element in an array.
.ArrayLists are resizable which means you can add or delete and so a lot more with the elements
Using different functions.
There are various different properties of ArrayList. Some of them are:
ArrayList.size: Returns the size of ArrayList.
ArrayList.add(element: E): Adds the specified element to the collection.
ArrayList.clear(): Removes all elements from this collection.
ArrayList.get(index: Int): Returns the element at the specified index in the list.
ArrayList.indexOf(element: E): Returns the index of the first occurrence of the specified element in the list, or -1 if the specified element is not contained in the list.
ArrayList.remove(element: E): Removes a single instance of the specified element from this collection, if it is present.
The above are the basic needed functions for ArrayLists.

Function in kotlin

.**Addition Funtion**

.**Substraction Function**

->Function are pieces of code that contain instructions to create an output from some input.They are offen reffer as method too.

->Some of the programming language function are predefine.

->functions which do not have return type are considered to return a Unit value.

Example

Consider a function :

```
fun printHello(name: String?) {
if (name != null)
println("Hello ${name}")
else
println("Hi there!")
}
```

The above function can be written as :

```
fun printHello(name: String?): Unit {
if (name != null)
println("Hello ${name}")
else
println("Hi there!")
// `return Unit` or `return` is optional
}
```

Example:

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun addition(a: Int ,b:Int){
   println(a+b) }
fun subtraction(c: Int ,d: Int):Int{
      var s = c -d
      return s}
fun printHello(name: String?):Unit {
   if (name != null)
      println("Hello ${name}")
   else
       println("Hi there!") }
fun main() {
  addition(2,3)
  println(subtraction(2,1) + 1)
  printHello("anand")
}
```

Object oriented programming Language in Kotlin.

1.What is Object oriented programming ?
2.Classes and Objects
3.Constructor
4.Types of programming language
There are two type of programming language.
Procedural programming language.
Object oriented programming language.

Explanation

Daily work:
Making your bed
Cleaning your house
Cooking
Dong Laundry
These above task performed by a robot .
This would be example of procedural programming language.

Object oriented programming language
1 robot for Daily work
2 Roboat for Making Your bed
3 roboat for cleaning House
4 Roboat for Cooking
5 Roboat for dong laundary

We break down the programming into Different part or object to make the hole process more efficient and manageable.

Classes and Objects

In Object Oriented Programming ,we make objects from the class so we don't have to write the same code again and again.

Example.

```
class Robot(name: String,color: String){
   var name:String
   var color:String

   init{
      this.name = name
      this.color = color
   }

fun makeBed(){
   println("I make your bed.")
}
fun cleanHouse(){
   println("I clean your house.")
}
fun coocking(){
   println("I cook for you")
}
fun doLaandry(){
   println("I do your laundary")
}
}

fun main() {
   var robot1 = Robot("Robot1","Black")
   println(robot1.name)
   println(robot1.color)
   robot1.makeBed()
   var robot2 = Robot("Robot2","White")
   println(robot2.name)
   println(robot2.color)
   robot1.cleanHouse()
   var robot3 = Robot("Robot3","Yello")
   println(robot3.name)
   println(robot3.color)
   robot1.coocking()
   var robot4 = Robot("Robot4","Pink")
```

```
    println(robot4.name)
    println(robot4.color)
    robot1.doLaandry()


}
```

Inheritance

Inheritance is the ability to create a new class from an existing class

Example.

```
open class Robot(){




fun makeBed(){
    println("I make your bed.")
}
fun cleanHouse(){
    println("I clean your house.")
}
open fun coocking(){
    println("I cook for you")
}
fun doLaandry(){
    println("I do your laundary")
}
}

class MikeRobots: Robot(){
    override fun coocking(){
        println("I coock for mike")

    }
}

fun main() {
   // var robot1 = Robot("Robot1","Black")
    //println(robot1.name)
   // println(robot1.color)
    //robot1.makeBed()
    //var robot2 = Robot("Robot2","White")
    //println(robot2.name)
    //println(robot2.color)
    //robot1.cleanHouse()
    //var robot3 = Robot("Robot3","Yello")
    //println(robot3.name)
    //println(robot3.color)
    //robot1.coocking()
    //var robot4 = Robot("Robot4","Pink")
    //println(robot4.name)
    //println(robot4.color)
    // robot1.doLaandry()

   var mikesCookingRobot = MikeRobots()
   mikesCookingRobot.coocking()

}
```

Encapsulation

Encapsulation is a concept of object oriented Programming and it refers to the idea of bundling data Together in packets so the code becomes modular and safe. A class is an example of encapsulation Interfaces.

Crash Handling

Try and catch statement

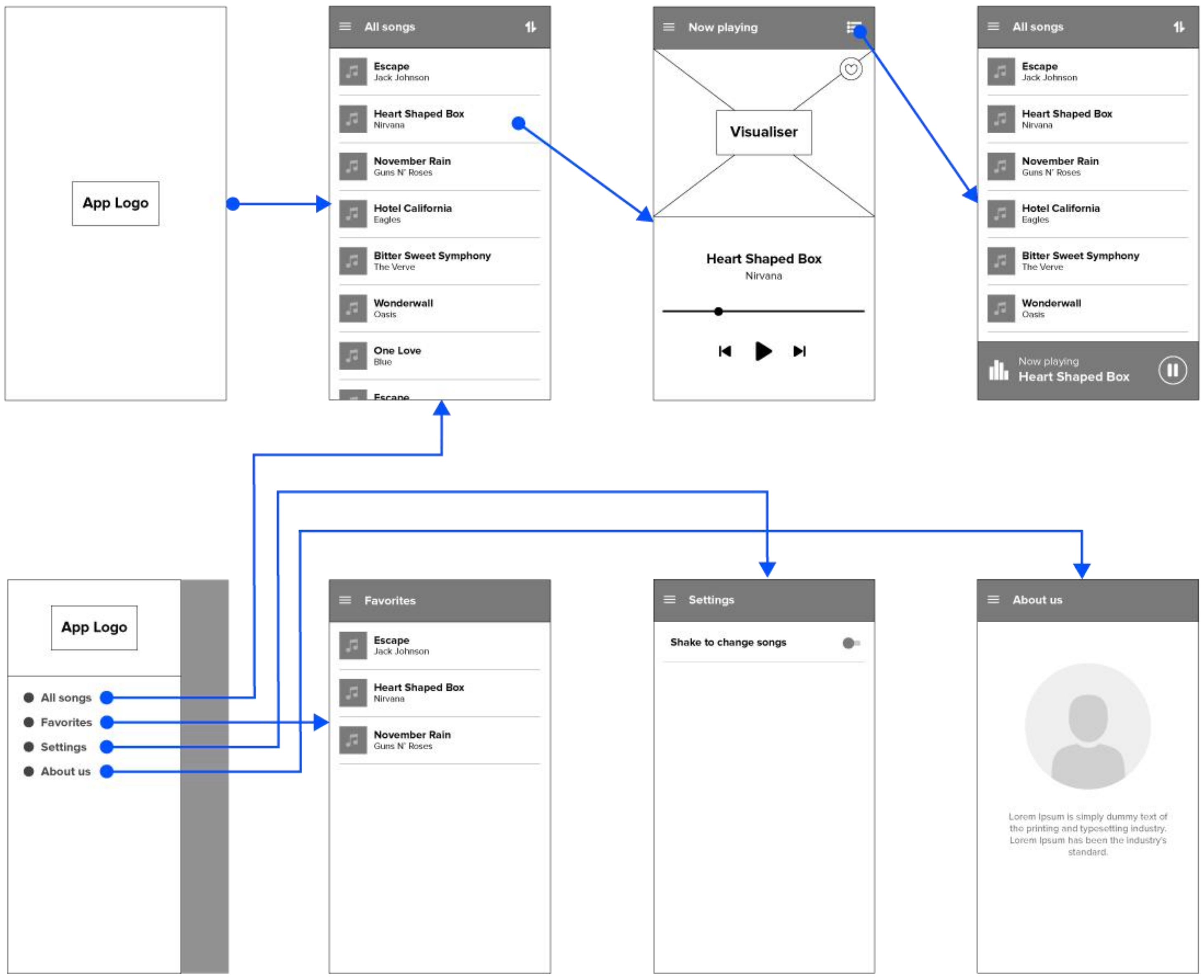
Exception

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

Example:

```
fun main(){
    try{
    division(20,0)
    }catch(e: ArithmeticException){
        println("Exception has occured")
        e.printStackTrace()
    }
}
fun division(a: Int, b:Int){
    println(a/b)
}
```
Null safety
```
fun main(){
    var temprature:Int? = null
    println(temprature)
}
```

DIAGRAM


App Logo
All songs
Escape
Jack Johnson
Heart Shaped Box
Nirvana
November Rain
Guns N' Roses
Hotel California
Eagles
Bitter Sweet Symphony
The Verve
Wonderwall
Oasis
One Love
Blue
Escape
Now playing
Visualiser
Heart Shaped Box
Nirvana
All songs
Escape
Jack Johnson
Heart Shaped Box
Nirvana
November Rain
Guns N' Roses
Hotel California
Eagles
Bitter Sweet Symphony
The Verve
Wonderwall
Oasis
Now playing
Heart Shaped Box
App Logo
All songs
Favorites
Settings
About us
Favorites
Escape
Jack Johnson
Heart Shaped Box
Nirvana
November Rain
Guns N' Roses
Settings
Shake to change songs
About us
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard.

Some Modules Coding

Source Code of the Module
A.

1. activity_splash.xml

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg_gradient">

    <ImageView
        android:layout_width="160dp"
        android:layout_height="82dp"
        android:layout_centerInParent="true"
        android:background="@drawable/echo_logo"
        android:minHeight="0dp"
        android:minWidth="0dp" />

</RelativeLayout>
```

2. SplashActivity.kt

```
package com.example.anandkumar.echo.activities

import android.Manifest
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.support.v4.app.ActivityCompat
import android.widget.Toast
import com.example.pratyushkarmakar.echo.R

class SplashActivity : AppCompatActivity() {

    var permissionsString = arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.MODIFY_AUDIO_SETTINGS,
            Manifest.permission.READ_PHONE_STATE,
            Manifest.permission.PROCESS_OUTGOING_CALLS,
            Manifest.permission.RECORD_AUDIO)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        if (!hasPermissions(this@SplashActivity, *permissionsString)) {
            //we have to ask for permissions
            ActivityCompat.requestPermissions(this@SplashActivity, permissionsString, 131)
        } else {
            Handler().postDelayed({
                val startAct = Intent(this@SplashActivity, MainActivity::class.java)
                startActivity(startAct)
                this.finish()
            }, 2000)
        }
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        when (requestCode) {
```

```
        131 -> {
            if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED
                    && grantResults[1] == PackageManager.PERMISSION_GRANTED
                    && grantResults[2] == PackageManager.PERMISSION_GRANTED
                    && grantResults[3] == PackageManager.PERMISSION_GRANTED
                    && grantResults[4] == PackageManager.PERMISSION_GRANTED) {
                Handler().postDelayed({
                    val startAct = Intent(this@SplashActivity, MainActivity::class.java)
                    startActivity(startAct)
                    this.finish()
                }, 2000)
            } else {
                Toast.makeText(this@SplashActivity, "Please grant all permissions!", Toast.LENGTH_SHORT).show()
                this.finish()
            }
            return
        }
        else -> {
            Toast.makeText(this@SplashActivity, "Something went wrong!", Toast.LENGTH_SHORT).show()
            this.finish()
            return
        }
    }
}

fun hasPermissions(context: Context, vararg permissions: String): Boolean {
    var hasAllPermissions = true
    for (permission in permissions) {
        var res = context.checkCallingOrSelfPermission(permission)
        if (res != PackageManager.PERMISSION_GRANTED) {
            hasAllPermissions = false
        }
    }
    return hasAllPermissions
}

}
```

B.

1.activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout 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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:id="@+id/header">

            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@drawable/bg_gradient"/>

            <ImageView
                android:layout_width="100dp"
                android:layout_height="50dp"
                android:background="@drawable/echo_logo"
                android:layout_centerInParent="true"/>

        </RelativeLayout>

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_below="@id/header"
            android:id="@+id/navigation_recycler_view"></android.support.v7.widget.RecyclerView>

    </RelativeLayout>

  </android.support.design.widget.NavigationView>

</android.support.v4.widget.DrawerLayout>
```

3. MainActivity.kt

```
package com.example.anandkumar.echo.activities

import android.app.Notification
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.support.annotation.RequiresApi
import android.support.v4.widget.DrawerLayout
import android.support.v7.app.ActionBarDrawerToggle
import android.support.v7.app.AppCompatActivity
import android.support.v7.widget.DefaultItemAnimator
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.support.v7.widget.Toolbar
import com.example.anandkumar.echo.R
import com.example.anandkumar.echo.adapters.NavigationDrawerAdapter
import com.example.anandkumar.echo.fragments.MainScreenFragment
import com.example.anandkumar.echo.fragments.SongPlayingFragment


class MainActivity : AppCompatActivity() {

  var navigationDrawerIconsList: ArrayList<String> = arrayListOf()
  var images_for_navdrawer = intArrayOf(R.drawable.navigation_allsongs, R.drawable.navigation_favorites,
        R.drawable.navigation_settings, R.drawable.navigation_aboutus)
  var trackNotificationBuilder: Notification?=null

  object Statified{
    var drawerLayout : DrawerLayout?=null
    var notificationManager: NotificationManager?=null
  }

  @RequiresApi(Build.VERSION_CODES.JELLY_BEAN)
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main)
        val toolbar = findViewById<Toolbar>(R.id.toolbar)
        setSupportActionBar(toolbar)
        MainActivity.Statified.drawerLayout = findViewById(R.id.drawer_layout)

        navigationDrawerIconsList.add("All Songs")
        navigationDrawerIconsList.add("Favorites")
        navigationDrawerIconsList.add("Settings")
        navigationDrawerIconsList.add("About Us")

        var toggle = ActionBarDrawerToggle(this@MainActivity, MainActivity.Statified.drawerLayout, toolbar,
                R.string.navigation_drawer_open, R.string.navigation_drawer_close)
        MainActivity.Statified.drawerLayout?.setDrawerListener(toggle)
        toggle.syncState()

        val mainScreenFragment = MainScreenFragment()
        this.supportFragmentManager
                .beginTransaction()
                .add(R.id.details_fragment, mainScreenFragment, "MainScreenFragment")
                .commit()

        var _navigationAdapter = NavigationDrawerAdapter(navigationDrawerIconsList, images_for_navdrawer, this)
        _navigationAdapter.notifyDataSetChanged()

        var navigation_recycler_view = findViewById<RecyclerView>(R.id.navigation_recycler_view)
        navigation_recycler_view.layoutManager = LinearLayoutManager(this)
        navigation_recycler_view.itemAnimator = DefaultItemAnimator()
        navigation_recycler_view.adapter = _navigationAdapter
        navigation_recycler_view.setHasFixedSize(true)

        val intent = Intent(this@MainActivity, MainActivity::class.java)
        val pIntent = PendingIntent.getActivity(this@MainActivity, System.currentTimeMillis().toInt(),
                intent, 0)
        trackNotificationBuilder = Notification.Builder(this)
                .setContentTitle("A track is playing in the background")
                .setSmallIcon(R.drawable.echo_logo)
                .setContentIntent(pIntent)
                .setOngoing(true)
                .setAutoCancel(true)
                .build()
        Statified.notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

    }

    override fun onStart() {
        super.onStart()
        try {
            Statified.notificationManager?.cancel(345)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    override fun onStop() {
        super.onStop()
        try {
            if (SongPlayingFragment.Statified.mediaplayer?.isPlaying as Boolean) {
                Statified.notificationManager?.notify(345, trackNotificationBuilder)
            }
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    override fun onResume() {
        super.onResume()
        try {
            Statified.notificationManager?.cancel(345)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }


}
```

C.

1. **fragment_song_playing.xml**

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <com.cleveroad.audiovisualization.GLAudioVisualizationView
                xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:app="http://schemas.android.com/apk/res-auto"
                android:id="@+id/visualizer_view"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                app:av_bubblesSize="25dp"
                app:av_bubblesRandomizeSizes="true"
                app:av_wavesHeight="60dp"
                app:av_wavesFooterHeight="170dp"
                app:av_wavesCount="50"
                app:av_layersCount="4"
                app:av_wavesColors="@array/rainbow"
                app:av_backgroundColor="#00032a"
                app:av_bubblesPerLayer="16" />

            <ImageButton
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/favoriteIcon"
                android:layout_alignParentRight="true"
                android:layout_margin="11dp"
                android:background="@drawable/white_circle_icon"
                android:src="@drawable/favorite_off"/>

        </RelativeLayout>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#9d2a58">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/information_song">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/songTitle"
        android:text="Castle of Glass"
        android:textColor="#ffffff"
        android:textSize="21sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:ellipsize="marquee"
        android:singleLine="true"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/songArtist"
        android:text="Linkin Park"
        android:textColor="#eeeeee"
        android:textSize="15sp"
        android:layout_below="@id/songTitle"
        android:layout_centerHorizontal="true"
        android:ellipsize="marquee"
        android:singleLine="true"/>

</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/seekBarLayout"
    android:layout_below="@id/information_song">

    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/seekBar"
        android:layout_centerHorizontal="true"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/startTime"
        android:layout_below="@id/seekBar"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="15dp"
        android:textColor="#ffffff"
        android:textAppearance="?android:attr/textAppearanceSmall"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/endTime"
        android:layout_below="@id/seekBar"
        android:layout_alignParentRight="true"
        android:layout_marginRight="15dp"
        android:textColor="#ffffff"
        android:textAppearance="?android:attr/textAppearanceSmall"/>

</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/controlPanel"
    android:layout_alignParentBottom="true"
    android:layout_centerVertical="true"
    android:layout_marginBottom="60dp"
    android:layout_marginTop="25dp"
    android:layout_below="@id/seekBarLayout">
```

```
        <ImageButton
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:minWidth="0dp"
            android:minHeight="0dp"
            android:id="@+id/playPauseButton"
            android:layout_centerInParent="true"
            android:background="@drawable/play_icon"/>

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="0dp"
            android:minWidth="0dp"
            android:id="@+id/previousButton"
            android:layout_centerVertical="true"
            android:layout_marginRight="19dp"
            android:layout_toLeftOf="@id/playPauseButton"
            android:background="@drawable/play_previous_icon"/>

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="0dp"
            android:minWidth="0dp"
            android:id="@+id/nextButton"
            android:layout_centerVertical="true"
            android:layout_marginLeft="19dp"
            android:layout_toRightOf="@id/playPauseButton"
            android:background="@drawable/play_next_icon"/>
        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="0dp"
            android:minWidth="0dp"
            android:id="@+id/loopButton"
            android:layout_centerVertical="true"
            android:layout_marginLeft="20dp"
            android:layout_toRightOf="@id/nextButton"
            android:background="@drawable/loop_white_icon"/>

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minHeight="0dp"
            android:minWidth="0dp"
            android:id="@+id/shuffleButton"
            android:layout_centerVertical="true"
            android:layout_marginRight="20dp"
            android:layout_toLeftOf="@id/previousButton"
            android:background="@drawable/shuffle_white_icon"/>

      </RelativeLayout>

    </RelativeLayout>

  </LinearLayout>

</LinearLayout>
```

2. SongPlayingFragment.kt

```
package com.example.pratyushkarmakar.echo.fragments


import android.app.Activity
import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.media.AudioManager
import android.media.MediaPlayer
import android.net.Uri
import android.os.Bundle
import android.os.Handler
import android.support.v4.app.Fragment
import android.support.v4.content.ContextCompat
import android.view.*
import android.widget.ImageButton
import android.widget.SeekBar
import android.widget.TextView
import android.widget.Toast
import com.cleveroad.audiovisualization.AudioVisualization
import com.cleveroad.audiovisualization.DbmHandler
import com.cleveroad.audiovisualization.GLAudioVisualizationView
import com.example.anandkumar.echo.CurrentSongHelper
import com.example.anandkumar.echo.R
import com.example.anandkumar.echo.Songs
import com.example.anandkumar.echo.databases.EchoDatabase
import java.util.*
import java.util.concurrent.TimeUnit


/**
 * A simple [Fragment] subclass.
 */
class SongPlayingFragment : Fragment() {

    object Statified {
        var myActivity: Activity? = null
        var mediaplayer: MediaPlayer? = null

        var startTimeText: TextView? = null
        var endTimeText: TextView? = null
        var songTitleView: TextView? = null
        var songArtistView: TextView? = null
        var playPauseImageButton: ImageButton? = null
        var previousImageButton: ImageButton? = null
        var nextImageButton: ImageButton? = null
        var loopImageButton: ImageButton? = null
        var shuffleImageButton: ImageButton? = null
        var seekbar: SeekBar? = null

        var currentSongHelper: CurrentSongHelper? = null
        var currentPosition: Int = 0
        var fetchSongs: ArrayList<Songs>? = null

        var audioVisualization: AudioVisualization? = null
        var glView: GLAudioVisualizationView? = null

        var fab: ImageButton? = null
        var favoriteContent: EchoDatabase? = null

        var mSensorManager: SensorManager?=null
        var mSensorListener: SensorEventListener?= null
        var MY_PREFS_NAME = "ShakeFeature"

        var updateSongTime = object : Runnable {
            override fun run() {
                val getcurrent = mediaplayer?.currentPosition
                var s = TimeUnit.MILLISECONDS.toSeconds(getcurrent?.toLong() as Long) -
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(getcurrent?.toLong()))
                startTimeText?.setText(String.format("%d:%d", TimeUnit.MILLISECONDS.toMinutes(getcurrent?.toLong()), s))

                Handler().postDelayed(this, 1000)
            }
        }
```

```
}

object Staticated {

    var MY_PREFS_SHUFFLE = "Shuffle feature"
    var MY_PREFS_LOOP = "Loop feature"

    fun playNext(check: String) {
        if (check.equals("PlayNextNormal", true)) {
            Statified.currentPosition = Statified.currentPosition + 1
        } else if (check.equals("PlayNextLikeNormalShuffle", true)) {
            var randomObject = Random()
            var randomPosition = randomObject.nextInt(Statified.fetchSongs?.size?.plus(1) as Int)
            Statified.currentPosition = randomPosition
        }
        if (Statified.currentPosition == Statified.fetchSongs?.size) {
            Statified.currentPosition = 0
        }
        Statified.currentSongHelper?.isLoop = false
        var nextSong = Statified.fetchSongs?.get(Statified.currentPosition)
        Statified.currentSongHelper?.songTitle = nextSong?.songTitle
        Statified.currentSongHelper?.songPath = nextSong?.songData
        Statified.currentSongHelper?.songId = nextSong?.songID as Long
        Statified.currentSongHelper?.currentPosition =Statified. currentPosition

        updateTextViews(Statified.currentSongHelper?.songTitle as String, Statified.currentSongHelper?.songArtist as String)

        Statified.mediaplayer?.reset()
        try {
            Statified.mediaplayer?.setDataSource(Statified.myActivity, Uri.parse(Statified.currentSongHelper?.songPath))
            Statified.mediaplayer?.prepare()
            Statified. mediaplayer?.start()
            processInformation(Statified.mediaplayer as MediaPlayer)
        } catch (e: Exception) {
            e.printStackTrace()
        }
        if (Statified.favoriteContent?.checkifIdExists(Statified.currentSongHelper?.songId?.toInt() as Int) as Boolean) {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_on))
        } else {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_off))
        }
    }

    fun onSongComplete() {
        if (Statified.currentSongHelper?.isShuffle as Boolean) {
            playNext("PlayNextLikeNormalShuffle")
            Statified.currentSongHelper?.isPlaying = true
        } else {
            if (Statified.currentSongHelper?.isLoop as Boolean) {
                Statified.currentSongHelper?.isPlaying = true

                var nextSong = Statified.fetchSongs?.get(Statified.currentPosition)
                Statified.currentSongHelper?.songTitle = nextSong?.songTitle
                Statified.currentSongHelper?.songPath = nextSong?.songData
                Statified.currentSongHelper?.songId = nextSong?.songID as Long
                Statified.currentSongHelper?.currentPosition = Statified.currentPosition

                updateTextViews(Statified.currentSongHelper?.songTitle as String, Statified.currentSongHelper?.songArtist as String)

                Statified.mediaplayer?.reset()
                try {
                    Statified.mediaplayer?.setDataSource(Statified.myActivity, Uri.parse(Statified.currentSongHelper?.songPath))
                    Statified.mediaplayer?.prepare()
                    Statified.mediaplayer?.start()
                    processInformation(Statified.mediaplayer as MediaPlayer)
                } catch (e: Exception) {
                    e.printStackTrace()
                }
            } else {
                playNext("PlayNextNormal")
                Statified.currentSongHelper?.isPlaying = true
            }
        }
        if (Statified.favoriteContent?.checkifIdExists(Statified.currentSongHelper?.songId?.toInt() as Int) as Boolean) {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_on))
        } else {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_off))
        }
    }

    fun updateTextViews(songtitle: String, songArtist: String) {
        var songTitleUpdated = songtitle
        var songArtistUpdated = songArtist
        if (songtitle.equals("<unknown>", true)) {
            songTitleUpdated = "unknown"
        }
        if (songArtist.equals("<unknown>", true)) {
            songArtistUpdated = "unknown"
        }
        Statified.songTitleView?.setText(songTitleUpdated)
        Statified.songArtistView?.setText(songArtistUpdated)
    }

    fun processInformation(mediaPlayer: MediaPlayer) {
        val finalTime = mediaPlayer.duration
        val startTime = mediaPlayer.currentPosition
        var ft = TimeUnit.MILLISECONDS.toSeconds(finalTime.toLong()) -
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(finalTime.toLong()))
        var st = TimeUnit.MILLISECONDS.toSeconds(startTime.toLong()) -
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(startTime.toLong()))
        Statified.seekbar?.max = finalTime
        Statified.startTimeText?.setText(String.format("%d:%d", TimeUnit.MILLISECONDS.toMinutes(startTime.toLong()), st))
        Statified.endTimeText?.setText(String.format("%d:%d", TimeUnit.MILLISECONDS.toMinutes(finalTime.toLong()), ft))
        Statified.seekbar?.setProgress(startTime)
        Handler().postDelayed(Statified.updateSongTime, 1000)
    }

}

var mAcceleration: Float = 0f
var mAccelerationCurrent: Float = 0f
var mAccelerationLast: Float = 0f

override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                          savedInstanceState: Bundle?): View? {
    // Inflate the layout for this fragment
    var view = inflater!!.inflate(R.layout.fragment_song_playing, container, false)
    setHasOptionsMenu(true)
    activity.title = "Now Playing"
    Statified.startTimeText = view?.findViewById(R.id.startTime)
    Statified.endTimeText = view?.findViewById(R.id.endTime)
    Statified.songTitleView = view?.findViewById(R.id.songTitle)
    Statified.songArtistView = view?.findViewById(R.id.songArtist)
    Statified.playPauseImageButton = view?.findViewById(R.id.playPauseButton)
    Statified.nextImageButton = view?.findViewById(R.id.nextButton)
    Statified.previousImageButton = view?.findViewById(R.id.previousButton)
    Statified.loopImageButton = view?.findViewById(R.id.loopButton)
    Statified.shuffleImageButton = view?.findViewById(R.id.shuffleButton)
    Statified.seekbar = view?.findViewById(R.id.seekBar)
    Statified.glView = view?.findViewById(R.id.visualizer_view)
    Statified.fab = view?.findViewById(R.id.favoriteIcon)
    Statified.fab?.alpha = 0.8f

    return view
}

override fun onViewCreated(view: View?, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    Statified.audioVisualization = Statified.glView as AudioVisualization
}
```

```
override fun onAttach(context: Context?) {
    super.onAttach(context)
    Statified.myActivity = context as Activity
}

override fun onAttach(activity: Activity?) {
    super.onAttach(activity)
    Statified.myActivity = activity
}

override fun onResume() {
    super.onResume()
    Statified.audioVisualization?.onResume()
    Statified.mSensorManager?.registerListener(Statified.mSensorListener,
            Statified.mSensorManager?.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
            SensorManager.SENSOR_DELAY_NORMAL)
}

override fun onPause() {
    Statified.audioVisualization?.onPause()
    super.onPause()
    Statified.mSensorManager?.unregisterListener(Statified.mSensorListener)
}

override fun onDestroyView() {
    Statified.audioVisualization?.release()
    super.onDestroyView()
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    Statified.mSensorManager = Statified.myActivity?.getSystemService(Context.SENSOR_SERVICE) as SensorManager
    mAcceleration = 0.0f
    mAccelerationCurrent = SensorManager.GRAVITY_EARTH
    mAccelerationLast = SensorManager.GRAVITY_EARTH
    bindShakeListener()
}

override fun onCreateOptionsMenu(menu: Menu?, inflater: MenuInflater?) {
    menu?.clear()
    inflater?.inflate(R.menu.song_playing_menu, menu)
    super.onCreateOptionsMenu(menu, inflater)

}

override fun onPrepareOptionsMenu(menu: Menu?) {
    super.onPrepareOptionsMenu(menu)
    val item: MenuItem?= menu?.findItem(R.id.action_redirect)
    item?.isVisible = true
    val item2: MenuItem?= menu?.findItem(R.id.action_sort)
    item2?.isVisible = false
}

override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    when (item?.itemId) {
        R.id.action_redirect -> {
            Statified.myActivity?.onBackPressed()
            return false
        }
    }
    return false
}

override fun onActivityCreated(savedInstanceState: Bundle?) {
    super.onActivityCreated(savedInstanceState)

    Statified.favoriteContent = EchoDatabase(Statified.myActivity)

    Statified.currentSongHelper = CurrentSongHelper()
    Statified.currentSongHelper?.isPlaying = true
    Statified.currentSongHelper?.isLoop = false
    Statified.currentSongHelper?.isShuffle = false

    var path: String? = null
    var _songTitle: String? = null
    var _songArtist: String? = null
    var songId: Long = 0

    try {
        path = arguments.getString("path")
        _songArtist = arguments.getString("songArtist")
        _songTitle = arguments.getString("songTitle")
        songId = arguments.getInt("songId").toLong()
        Statified.currentPosition = arguments.getInt("songPosition")
        Statified.fetchSongs = arguments.getParcelableArrayList("songData")

        Statified.currentSongHelper?.songPath = path
        Statified.currentSongHelper?.songTitle = _songTitle
        Statified.currentSongHelper?.songArtist = _songArtist
        Statified.currentSongHelper?.songId = songId
        Statified.currentSongHelper?.currentPosition = Statified.currentPosition

        Staticated.updateTextViews(Statified.currentSongHelper?.songTitle as String, Statified.currentSongHelper?.songArtist as String)

    } catch (e: Exception) {
        e.printStackTrace()
    }

    var fromFavBottomBar = arguments.get("FavBottomBar") as? String
    if (fromFavBottomBar != null) {
        Statified.mediaplayer = FavoritesFragment.Statified.mediaPlayer
    } else {
        Statified.mediaplayer = MediaPlayer()
        Statified.mediaplayer?.setAudioStreamType(AudioManager.STREAM_MUSIC)
        try {
            Statified.mediaplayer?.setDataSource(Statified.myActivity, Uri.parse(path))
            Statified.mediaplayer?.prepare()
        } catch (e: Exception) {
            e.printStackTrace()
        }
        Statified.mediaplayer?.start()
    }
    Staticated.processInformation(Statified.mediaplayer as MediaPlayer)

    if (Statified.currentSongHelper?.isPlaying as Boolean) {
        Statified.playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
    } else {
        Statified.playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
    }
    Statified.mediaplayer?.setOnCompletionListener {
        Staticated.onSongComplete()
    }
    clickHandler()
    var visualizationHandler = DbmHandler.Factory.newVisualizerHandler(Statified.myActivity as Context, 0)
    Statified.audioVisualization?.linkTo(visualizationHandler)

    var prefsForShuffle = Statified.myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE, Context.MODE_PRIVATE)
    var isShuffleAllowed = prefsForShuffle?.getBoolean("feature", false)
    if (isShuffleAllowed as Boolean) {
        Statified.currentSongHelper?.isShuffle = true
        Statified.currentSongHelper?.isLoop = false
        Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_icon)
        Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
    } else {
        Statified.currentSongHelper?.isShuffle = false
        Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
    }

    var prefsForLoop = Statified.myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP, Context.MODE_PRIVATE)
```

```
        var isLoopAllowed = prefsForLoop?.getBoolean("feature", false)
        if (isLoopAllowed as Boolean) {
            Statified.currentSongHelper?.isLoop = true
            Statified.currentSongHelper?.isShuffle = false
            Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_icon)
            Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
        } else {
            Statified.currentSongHelper?.isLoop = false
            Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
        }

        if (Statified.favoriteContent?.checkifIdExists(Statified.currentSongHelper?.songId?.toInt() as Int) as Boolean) {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_on))
        } else {
            Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_off))
        }

    }

    fun clickHandler() {

        Statified.fab?.setOnClickListener({
            if (Statified.favoriteContent?.checkifIdExists(Statified.currentSongHelper?.songId?.toInt() as Int) as Boolean) {
                Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_off))
                Statified.favoriteContent?.deleteFavourite(Statified.currentSongHelper?.songId?.toInt() as Int)
                Toast.makeText(Statified.myActivity, "Removed from favorites", Toast.LENGTH_SHORT).show()
            } else {
                Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_on))
                Statified.favoriteContent?.storeAsFavorite(Statified.currentSongHelper?.songId?.toInt(),
Statified.currentSongHelper?.songArtist,
                        Statified.currentSongHelper?.songTitle, Statified.currentSongHelper?.songPath)
                Toast.makeText(Statified.myActivity, "Added to favorites", Toast.LENGTH_SHORT).show()
            }
        })

        Statified.shuffleImageButton?.setOnClickListener({

            var editorShuffle = Statified.myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE,
Context.MODE_PRIVATE)?.edit()
            var editorLoop = Statified.myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP, Context.MODE_PRIVATE)?.edit()

            if (Statified.currentSongHelper?.isShuffle as Boolean) {
                Statified.currentSongHelper?.isShuffle = false
                Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
                editorShuffle?.putBoolean("feature", false)
                editorShuffle?.apply()
            } else {
                Statified.currentSongHelper?.isShuffle = true
                Statified.currentSongHelper?.isLoop = false
                Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_icon)
                Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
                editorShuffle?.putBoolean("feature", true)
                editorShuffle?.apply()
                editorLoop?.putBoolean("feature", false)
                editorLoop?.apply()
            }
        })
        Statified.loopImageButton?.setOnClickListener({

            var editorShuffle = Statified.myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE,
Context.MODE_PRIVATE)?.edit()
            var editorLoop =Statified. myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP, Context.MODE_PRIVATE)?.edit()

            if (Statified.currentSongHelper?.isLoop as Boolean) {
                Statified.currentSongHelper?.isLoop = false
                Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
                editorLoop?.putBoolean("feature", false)
                editorLoop?.apply()
            } else {
                Statified.currentSongHelper?.isLoop = true
            Statified.currentSongHelper?.isShuffle = false
            Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_icon)
            Statified.shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
            editorLoop?.putBoolean("feature", true)
            editorLoop?.apply()
            editorShuffle?.putBoolean("feature", false)
            editorShuffle?.apply()
        }
    })
    Statified.nextImageButton?.setOnClickListener({
        Statified.currentSongHelper?.isPlaying = true
        if (Statified.currentSongHelper?.isShuffle as Boolean) {
            Staticated.playNext("PlayNextLikeNormalShuffle")
        } else {
            Staticated.playNext("PlayNextNormal")
        }
    })
    Statified.previousImageButton?.setOnClickListener({
        Statified.currentSongHelper?.isPlaying = true
        Statified.playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
        if (Statified.currentSongHelper?.isLoop as Boolean) {
            Statified.loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
        }
        playPrevious()
    })
    Statified.playPauseImageButton?.setOnClickListener({
        if (Statified.mediaplayer?.isPlaying as Boolean) {
            Statified.mediaplayer?.pause()
            Statified.currentSongHelper?.isPlaying = false
            Statified.playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
        } else {
            Statified.mediaplayer?.start()
            Statified.currentSongHelper?.isPlaying = true
            Statified.playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
        }
    })
}

fun playPrevious() {
    Statified.currentPosition = Statified.currentPosition - 1
    if (Statified.currentPosition == -1) {
        Statified.currentPosition = 0
    }
    if (Statified.currentSongHelper?.isPlaying as Boolean) {
        Statified.playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
    } else {
        Statified.playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
    }
    Statified.currentSongHelper?.isLoop = false
    val nextSong = Statified.fetchSongs?.get(Statified.currentPosition)
    Statified.currentSongHelper?.songTitle = nextSong?.songTitle
    Statified.currentSongHelper?.songPath = nextSong?.songData
    Statified.currentSongHelper?.songId = nextSong?.songID as Long
    Statified.currentSongHelper?.currentPosition = Statified.currentPosition

    Staticated.updateTextViews(Statified.currentSongHelper?.songTitle as String, Statified.currentSongHelper?.songArtist as String)

    Statified.mediaplayer?.reset()
    try {
        Statified.mediaplayer?.setDataSource(Statified.myActivity, Uri.parse(Statified.currentSongHelper?.songPath))
        Statified.mediaplayer?.prepare()
        Statified.mediaplayer?.start()
        Staticated.processInformation(Statified.mediaplayer as MediaPlayer)
    } catch (e: Exception) {
        e.printStackTrace()
    }
    if (Statified.favoriteContent?.checkifIdExists(Statified.currentSongHelper?.songId?.toInt() as Int) as Boolean) {
        Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_on))
    } else {
        Statified.fab?.setImageDrawable(ContextCompat.getDrawable(Statified.myActivity, R.drawable.favorite_off))
        }
    }

    fun bindShakeListener() {
        Statified.mSensorListener = object : SensorEventListener {
            override fun onAccuracyChanged(p0: Sensor?, p1: Int) {
                }

            override fun onSensorChanged(p0: SensorEvent) {
                val x = p0.values[0]
                val y = p0.values[1]
                val z = p0.values[2]

                mAccelerationLast = mAccelerationCurrent
                mAccelerationCurrent = Math.sqrt(((x*x + y*y + z*z).toDouble())).toFloat()
                val delta = mAccelerationCurrent - mAccelerationLast
                mAcceleration = mAcceleration * 0.9f + delta

                if (mAcceleration > 12) {
                    val prefs = Statified.myActivity?.getSharedPreferences(Statified.MY_PREFS_NAME, Context.MODE_PRIVATE)
                    val isAllowed = prefs?.getBoolean("feature", false)
                    if (isAllowed as Boolean) {
                        Staticated.playNext("PlayNextNormal")
                    }
                }

                }

        }
    }

}// Required empty public constructor
```

D.

1. fragment_setting

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:clickable="true"
    android:paddingTop="11dp"
    android:background="#ffffff">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#212121"
        android:text="Shake to change song"
        android:layout_marginLeft="25dp"
        android:textSize="17sp"
        android:id="@+id/shaketochange"/>

    <Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/switchShake"
        android:layout_alignParentRight="true"
        android:backgroundTint="#212121"
        android:layout_marginRight="25dp"/>

    <View
        android:layout_width="match_parent"
        android:layout_height="0.2dp"
        android:layout_below="@id/shaketochange"
        android:layout_margin="25dp"
        android:background="#bdbdbd"/>

</RelativeLayout>
```

2. SettingFragment.kt

```
package com.example.anandkumar.echo.fragments


import android.app.Activity
import android.content.Context
import android.os.Bundle
import android.support.v4.app.Fragment
import android.view.LayoutInflater
import android.view.Menu
import android.view.View
import android.view.ViewGroup
import android.widget.Switch
import com.example.anandkumar.echo.R


/**
 * A simple [Fragment] subclass.
 */
class SettingsFragment : Fragment() {

    var myActivity: Activity?=null
    var shakeSwitch: Switch?=null

    object Statified {
        var MY_PREFS_NAME = "ShakeFeature"
    }

    override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        val view = inflater!!.inflate(R.layout.fragment_settings, container, false)
        shakeSwitch = view.findViewById(R.id.switchShake)
        activity.title = "Settings"
        return view
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setHasOptionsMenu(true)
    }

    override fun onAttach(context: Context?) {
        super.onAttach(context)
        myActivity = context as Activity
    }

    override fun onAttach(activity: Activity?) {
        super.onAttach(activity)
        myActivity = activity
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        val prefs = myActivity?.getSharedPreferences(Statified.MY_PREFS_NAME, Context.MODE_PRIVATE)
        val isAllowed = prefs?.getBoolean("feature", false)
        if (isAllowed as Boolean) {
            shakeSwitch?.isChecked = true
        } else {
            shakeSwitch?.isChecked = false
        }
        shakeSwitch?.setOnCheckedChangeListener({compoundButton, b ->
            if (b) {
                val editor = myActivity?.getSharedPreferences(Statified.MY_PREFS_NAME, Context.MODE_PRIVATE)?.edit()
                editor?.putBoolean("feature", true)
                editor?.apply()
            } else {
                val editor = myActivity?.getSharedPreferences(Statified.MY_PREFS_NAME, Context.MODE_PRIVATE)?.edit()
                editor?.putBoolean("feature", false)
                editor?.apply()
            }
        })
    }

    override fun onPrepareOptionsMenu(menu: Menu?) {
        super.onPrepareOptionsMenu(menu)
        val item = menu?.findItem(R.id.action_sort)
        item?.isVisible = false
    }

}// Required empty public constructor
```

E

1. fragment_main_screen

```
<RelativeLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:clickable="true"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content_main">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/visibleLayout">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/contentMain"></android.support.v7.widget.RecyclerView>

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:visibility="invisible"
            android:id="@+id/hiddenBarMainScreen"
            android:background="@color/colorPrimary"
            android:layout_alignParentBottom="true">

            <ImageButton
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/playPauseButton"
                android:layout_centerVertical="true"
                android:layout_alignParentRight="true"
                android:layout_marginRight="31dp"
                android:background="@drawable/pause_icon"/>

            <ImageView
                android:layout_width="50dp"
                android:layout_height="50dp"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:id="@+id/defaultMusic"
                android:background="@drawable/now_playing_bar_eq_image"
                android:layout_marginLeft="13dp"
                android:minHeight="0dp"
                android:minWidth="0dp"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Now Playing"
                android:id="@+id/nowPlaying"
                android:layout_toRightOf="@+id/defaultMusic"
                android:layout_marginLeft="15dp"
                android:layout_marginTop="11dp"
                android:textColor="#bdbdbd"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/songTitleMainScreen"
                android:layout_below="@id/nowPlaying"
                android:ellipsize="end"
                android:maxLines="1"
                android:maxWidth="160dp"
                android:singleLine="true"
                android:layout_marginTop="5dp"
                android:layout_alignLeft="@id/nowPlaying"
                android:layout_alignStart="@id/nowPlaying"
                android:text="..."
                android:textColor="#ffffff"
                android:textAppearance="?android:attr/textAppearanceMedium"/>

        </RelativeLayout>

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="invisible"
        android:background="#ffffff"
        android:id="@+id/noSongs">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="No songs available!"
            android:background="#000000"
            android:layout_centerInParent="true"/>

    </RelativeLayout>

  </RelativeLayout>
```

2. MainScreenFragment.kt

```
package com.example.anandkumar.echo.fragments


import android.app.Activity
import android.content.Context
import android.os.Bundle
import android.provider.MediaStore
import android.support.v4.app.Fragment
import android.support.v7.widget.DefaultItemAnimator
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.view.*
import android.widget.ImageButton
import android.widget.RelativeLayout
import android.widget.TextView
import com.example.anandkumar.echo.R
import com.example.anandkumar.echo.Songs
import com.example.anandkumar.echo.adapters.MainScreenAdapter
import java.util.*


/**
 * A simple [Fragment] subclass.
 */
class MainScreenFragment : Fragment() {

  var getSongsList: ArrayList<Songs>? = null
  var nowPlayingBottomBar: RelativeLayout? = null
  var playPauseButton: ImageButton? = null
  var songTitle: TextView? = null
  var visibleLayout: RelativeLayout? = null
  var noSongs: RelativeLayout? = null
  var recyclerView: RecyclerView? = null
  var myActivity: Activity? = null
  var _mainScreenAdapter: MainScreenAdapter? = null

  override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                            savedInstanceState: Bundle?): View? {
      // Inflate the layout for this fragment
      val view = inflater!!.inflate(R.layout.fragment_main_screen, container, false)
      setHasOptionsMenu(true)
      activity.title = "All Songs"
      visibleLayout = view?.findViewById<RelativeLayout>(R.id.visibleLayout)
      noSongs = view?.findViewById<RelativeLayout>(R.id.noSongs)
      nowPlayingBottomBar = view?.findViewById<RelativeLayout>(R.id.hiddenBarMainScreen)
      songTitle = view?.findViewById<TextView>(R.id.songTitleMainScreen)
      playPauseButton = view?.findViewById<ImageButton>(R.id.playPauseButton)
      recyclerView = view?.findViewById<RecyclerView>(R.id.contentMain)

      return view
  }

  override fun onActivityCreated(savedInstanceState: Bundle?) {
      super.onActivityCreated(savedInstanceState)
      getSongsList = getSongsFromPhone()

      val prefs = activity.getSharedPreferences("action_sort", Context.MODE_PRIVATE)
      val action_sort_ascending = prefs.getString("action_sort_ascending", "true")
      val action_sort_recent = prefs.getString("action_sort_recent", "false")

      if (getSongsList == null) {
          visibleLayout?.visibility = View.INVISIBLE
          noSongs?.visibility = View.VISIBLE
      } else {
          _mainScreenAdapter = MainScreenAdapter(getSongsList as ArrayList<Songs>, myActivity as Context)
          val mLayoutManager = LinearLayoutManager(myActivity)
          recyclerView?.layoutManager = mLayoutManager
          recyclerView?.itemAnimator = DefaultItemAnimator()
          recyclerView?.adapter = _mainScreenAdapter
      }

      if (getSongsList != null) {
          if (action_sort_ascending!!.equals("true", ignoreCase = true)) {
              Collections.sort(getSongsList, Songs.Statified.nameComparator)
              _mainScreenAdapter?.notifyDataSetChanged()
          } else if (action_sort_recent!!.equals("true", ignoreCase = true)) {
              Collections.sort(getSongsList, Songs.Statified.dateComparator)
              _mainScreenAdapter?.notifyDataSetChanged()
          }
      }
  }

  override fun onCreateOptionsMenu(menu: Menu?, inflater: MenuInflater?) {
      menu?.clear()
      inflater?.inflate(R.menu.main, menu)
      return
  }

  override fun onOptionsItemSelected(item: MenuItem?): Boolean {
      return super.onOptionsItemSelected(item)
      val switcher = item?.itemId
      if (switcher == R.id.action_sort_ascending) {
          val editor = myActivity?.getSharedPreferences("action_sort", Context.MODE_PRIVATE)?.edit()
          editor?.putString("action_sort_ascending", "true")
          editor?.putString("action_sort_recent", "false")
          editor?.apply()
          if (getSongsList != null) {
              Collections.sort(getSongsList, Songs.Statified.nameComparator)
          }
```

```
            _mainScreenAdapter?.notifyDataSetChanged()
            return false

        } else if (switcher == R.id.action_sort_recent) {
            val editortwo = myActivity?.getSharedPreferences("action_sort", Context.MODE_PRIVATE)?.edit()
            editortwo?.putString("action_sort_ascending", "false")
            editortwo?.putString("action_sort_recent", "true")
            editortwo?.apply()
            if (getSongsList != null) {
                Collections.sort(getSongsList, Songs.Statified.dateComparator)
            }
            _mainScreenAdapter?.notifyDataSetChanged()
            return false
        }
    }

    override fun onAttach(context: Context?) {
        super.onAttach(context)

        myActivity = context as Activity

    }

    override fun onAttach(activity: Activity?) {
        super.onAttach(activity)

        myActivity = activity

    }

    fun getSongsFromPhone(): ArrayList<Songs> {

        var arrayList = ArrayList<Songs>()
        var contentResolver = myActivity?.contentResolver
        var songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI
        var songCursor = contentResolver?.query(songUri, null, null, null, null)
        if (songCursor != null && songCursor.moveToFirst()) {
            val songId = songCursor.getColumnIndex(MediaStore.Audio.Media._ID)
            val songTitle = songCursor.getColumnIndex(MediaStore.Audio.Media.TITLE)
            val songArtist = songCursor.getColumnIndex(MediaStore.Audio.Media.ARTIST)
            val songData = songCursor.getColumnIndex(MediaStore.Audio.Media.DATA)
            val dateIndex = songCursor.getColumnIndex(MediaStore.Audio.Media.DATE_ADDED)
            while (songCursor.moveToNext()) {
                var currentId = songCursor.getLong(songId)
                var currentTitle = songCursor.getString(songTitle)
                var currentArtist = songCursor.getString(songArtist)
                var currentData = songCursor.getString(songData)
                var currentdate = songCursor.getLong(dateIndex)
                arrayList.add(Songs(currentId, currentTitle, currentArtist, currentData, currentdate))
            }
        }
        return arrayList
    }

}// Required empty public constructor
```

F.

1. fragment_favorites.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:clickable="true">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/favoriteRecycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"></android.support.v7.widget.RecyclerView>

    <RelativeLayout
        android:id="@+id/hiddenBarFavScreen"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_alignParentBottom="true"
        android:background="@color/colorPrimary"
        android:visibility="invisible">

        <ImageButton
            android:id="@+id/playPauseButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:layout_marginRight="31dp"
            android:background="@drawable/pause_icon" />

        <ImageView
            android:id="@+id/defaultMusic"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:layout_marginLeft="13dp"
            android:background="@drawable/now_playing_bar_eq_image"
            android:minHeight="0dp"
            android:minWidth="0dp" />

        <TextView
            android:id="@+id/nowPlaying"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15dp"
            android:layout_marginTop="11dp"
            android:layout_toRightOf="@+id/defaultMusic"
            android:text="Now Playing"
            android:textColor="#bdbdbd" />

        <TextView
            android:id="@+id/songTitleFavScreen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@id/nowPlaying"
            android:layout_alignStart="@id/nowPlaying"
            android:layout_below="@id/nowPlaying"
            android:layout_marginTop="5dp"
            android:ellipsize="end"
            android:maxLines="1"
            android:maxWidth="160dp"
            android:singleLine="true"
            android:text="..."
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:textColor="#ffffff" />

    </RelativeLayout>

    <TextView
        android:id="@+id/noFavorites"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="No Favorites!"
        android:textColor="#212121"
        android:visibility="invisible" />

</RelativeLayout>
```

2. FavouritesFargment.kt

```kotlin
package com.example.anandkumar.echo.fragments


import android.app.Activity
import android.content.Context
import android.media.MediaPlayer
import android.os.Bundle
import android.provider.MediaStore
import android.support.v4.app.Fragment
import android.support.v7.widget.DefaultItemAnimator
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.Menu
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.RelativeLayout
import android.widget.TextView
import com.example.anandkumar.echo.R
import com.example.anandkumar.echo.Songs
import com.example.anandkumar.echo.adapters.FavoriteAdapter
import com.example.anandkumar.echo.databases.EchoDatabase


/**
 * A simple [Fragment] subclass.
 */
class FavoritesFragment : Fragment() {

    var myActivity: Activity? = null

    var noFavorites: TextView? = null
    var nowPlayingBottomBar: RelativeLayout? = null
    var playPauseButton: ImageButton? = null
    var songTitle: TextView? = null
    var recyclerView: RecyclerView? = null
    var trackPosition: Int = 0

    var favoriteContent: EchoDatabase? = null

    var refreshList: ArrayList<Songs>? = null
    var getListfromDatabase: ArrayList<Songs>? = null

    object Statified {
        var mediaPlayer: MediaPlayer? = null
    }

    override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        val view = inflater!!.inflate(R.layout.fragment_favorites, container, false)

        activity.title = "Favorites"
        noFavorites = view?.findViewById(R.id.noFavorites)
        nowPlayingBottomBar = view.findViewById(R.id.hiddenBarFavScreen)
        playPauseButton = view.findViewById(R.id.playPauseButton)
        songTitle = view.findViewById(R.id.songTitleFavScreen)
        recyclerView = view.findViewById(R.id.favoriteRecycler)

        return view
    }

    override fun onAttach(context: Context?) {
        super.onAttach(context)
        myActivity = context as Activity
    }
```

```
override fun onAttach(activity: Activity?) {
    super.onAttach(activity)
    myActivity = activity
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
}

override fun onActivityCreated(savedInstanceState: Bundle?) {
    super.onActivityCreated(savedInstanceState)

    favoriteContent = EchoDatabase(myActivity)
    display_favorites_by_searching()
    bottomBarSetup()

}

override fun onResume() {
    super.onResume()
}

override fun onPrepareOptionsMenu(menu: Menu?) {
    super.onPrepareOptionsMenu(menu)
    val item = menu?.findItem(R.id.action_sort)
    item?.isVisible = false
}

fun getSongsFromPhone(): ArrayList<Songs> {

    var arrayList = ArrayList<Songs>()
    var contentResolver = myActivity?.contentResolver
    var songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI
    var songCursor = contentResolver?.query(songUri, null, null, null, null)
    if (songCursor != null && songCursor.moveToFirst()) {
        val songId = songCursor.getColumnIndex(MediaStore.Audio.Media._ID)
        val songTitle = songCursor.getColumnIndex(MediaStore.Audio.Media.TITLE)
        val songArtist = songCursor.getColumnIndex(MediaStore.Audio.Media.ARTIST)
        val songData = songCursor.getColumnIndex(MediaStore.Audio.Media.DATA)
        val dateIndex = songCursor.getColumnIndex(MediaStore.Audio.Media.DATE_ADDED)
        while (songCursor.moveToNext()) {
            var currentId = songCursor.getLong(songId)
            var currentTitle = songCursor.getString(songTitle)
            var currentArtist = songCursor.getString(songArtist)
            var currentData = songCursor.getString(songData)
            var currentdate = songCursor.getLong(dateIndex)
            arrayList.add(Songs(currentId, currentTitle, currentArtist, currentData, currentdate))
        }
    }
    return arrayList
}

fun bottomBarSetup() {
    try {
        bottomBarClickHandler()
        songTitle?.setText(SongPlayingFragment.Statified.currentSongHelper?.songTitle)
        SongPlayingFragment.Statified.mediaplayer?.setOnCompletionListener({
            songTitle?.setText(SongPlayingFragment.Statified.currentSongHelper?.songTitle)
            SongPlayingFragment.Staticated.onSongComplete()
        })
        if (SongPlayingFragment.Statified.mediaplayer?.isPlaying as Boolean) {
            nowPlayingBottomBar?.visibility = View.VISIBLE
        } else {
            nowPlayingBottomBar?.visibility = View.INVISIBLE
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

```
    fun bottomBarClickHandler() {
        nowPlayingBottomBar?.setOnClickListener({
            Statified.mediaPlayer = SongPlayingFragment.Statified.mediaplayer
            val songPlayingFragment = SongPlayingFragment()
            var args = Bundle()
            args.putString("songArtist", SongPlayingFragment.Statified.currentSongHelper?.songArtist)
            args.putString("songTitle", SongPlayingFragment.Statified.currentSongHelper?.songTitle)
            args.putString("path", SongPlayingFragment.Statified.currentSongHelper?.songPath)
            args.putInt("songId", SongPlayingFragment.Statified.currentSongHelper?.songId?.toInt() as Int)
            args.putInt("songPosition", SongPlayingFragment.Statified.currentSongHelper?.currentPosition?.toInt() as Int)
            args.putParcelableArrayList("songData", SongPlayingFragment.Statified.fetchSongs)
            args.putString("FavBottomBar", "success")
            songPlayingFragment.arguments = args
            fragmentManager.beginTransaction()
                    .replace(R.id.details_fragment, songPlayingFragment)
                    .addToBackStack("SongPlayingFragment")
                    .commit()
        })

        playPauseButton?.setOnClickListener({
            if (SongPlayingFragment.Statified.mediaplayer?.isPlaying as Boolean) {
                SongPlayingFragment.Statified.mediaplayer?.pause()
                trackPosition = SongPlayingFragment.Statified.mediaplayer?.getCurrentPosition() as Int
                playPauseButton?.setBackgroundResource(R.drawable.play_icon)
            } else {
                SongPlayingFragment.Statified.mediaplayer?.seekTo(trackPosition)
                SongPlayingFragment.Statified.mediaplayer?.start()
                playPauseButton?.setBackgroundResource(R.drawable.pause_icon)
            }
        })
    }

    fun display_favorites_by_searching() {
        if (favoriteContent?.checkSize() as Int > 0) {
            refreshList = ArrayList<Songs>()
            getListfromDatabase = favoriteContent?.queryDBList()
            var fetchListfromDevice = getSongsFromPhone()
            if (fetchListfromDevice != null) {
                for (i in 0..fetchListfromDevice?.size - 1) {
                    for (j in 0..getListfromDatabase?.size as Int - 1) {
                        if ((getListfromDatabase?.get(j)?.songID) === (fetchListfromDevice?.get(i)?.songID)) {
                            refreshList?.add((getListfromDatabase as ArrayList<Songs>)[j])
                        }
                    }
                }
            }

            if (refreshList == null) {
                recyclerView?.visibility = View.INVISIBLE
                noFavorites?.visibility = View.VISIBLE
            } else {
                var favoriteAdapter = FavoriteAdapter(refreshList as ArrayList<Songs>, myActivity as Context)
                val mLayoutManager = LinearLayoutManager(activity)
                recyclerView?.layoutManager = mLayoutManager
                recyclerView?.itemAnimator = DefaultItemAnimator()
                recyclerView?.adapter = favoriteAdapter
                recyclerView?.setHasFixedSize(true)
            }

        } else {
            recyclerView?.visibility = View.INVISIBLE
            noFavorites?.visibility = View.VISIBLE
        }
    }

}
```

G.

1. fragment_about_us.xml

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/aboutUs"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <ImageView
        android:id="@+id/joeyImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:src="@drawable/joey"
        android:padding="15dp"/>

    <TextView
        android:id="@+id/anand"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/joeyImage"
        android:text="->This app is made by Anand Kumar"

        android:layout_centerHorizontal="true"
        android:layout_margin="15dp"
        android:textColor="#212121"/>
    <TextView
        android:id="@+id/anand2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/anand"
        android:text="->ak.cse101@gmail.com"
        android:layout_centerHorizontal="true"
        android:layout_margin="5dp"
        android:textColor="#212121"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/anand2"
        android:text="->software Developer"
        android:layout_centerHorizontal="true"
        android:layout_margin="5dp"
        android:textColor="#212121"/>


</RelativeLayout>
```

2. AboutUsFragment.kt

```
package com.example.anandkumar.echo.fragments


import android.os.Bundle
import android.support.v4.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.anandkumar.echo.R


/**
 * A simple [Fragment] subclass.
 */
class AboutUsFragment : Fragment() {


    override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
```

```
        // Inflate the layout for this fragment
        return inflater!!.inflate(R.layout.fragment_about_us, container, false)
    }

}// Required empty public constructor
```

##Dtabase

```
package com.example.anandkumar.echo.databases

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import com.example.pratyushkarmakar.echo.Songs

/**
 * Created by anand kumar
 */

class EchoDatabase: SQLiteOpenHelper {

    var _songList = ArrayList<Songs>()
    object Staticated {
        var DB_VERSION = 1
        val DB_NAME = "FavoriteDatabase"
        val TABLE_NAME = "FavoriteTable"
        val COLUMN_ID = "SongID"
        val COLUMN_SONG_TITLE = "SongTitle"
        val COLUMN_SONG_ARTIST = "SongArtist"
        val COLUMN_SONG_PATH = "SongPath"
    }

    override fun onCreate(sqliteDatabase: SQLiteDatabase?) {
        sqliteDatabase?.execSQL("CREATE TABLE " + Staticated.TABLE_NAME + "( " + Staticated.COLUMN_ID + "
INTEGER," + Staticated.COLUMN_SONG_ARTIST + " STRING," +
                Staticated.COLUMN_SONG_TITLE + " STRING," + Staticated.COLUMN_SONG_PATH + " STRING);")
        }

    override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {
        TODO("not implemented") //To change body of created functions use File | Settings | File
Templates.
    }

    constructor(context: Context?, name: String?, factory: SQLiteDatabase.CursorFactory?, version: Int) :
super(context, name, factory, version)
    constructor(context: Context?) : super(context, Staticated.DB_NAME, null, Staticated.DB_VERSION)


    fun storeAsFavorite(id: Int?, artist: String?, songTitle: String?, path: String?) {
        val db = this.writableDatabase
        var contentValues = ContentValues()
        contentValues.put(Staticated.COLUMN_ID, id)
        contentValues.put(Staticated.COLUMN_SONG_ARTIST, artist)
        contentValues.put(Staticated.COLUMN_SONG_TITLE, songTitle)
        contentValues.put(Staticated.COLUMN_SONG_PATH, path)
        db.insert(Staticated.TABLE_NAME, null, contentValues)
        db.close()
    }

    fun queryDBList(): ArrayList<Songs>? {
        try {
            val db = this.readableDatabase
            val query_params = "SELECT * FROM " + Staticated.TABLE_NAME
            var cSor = db.rawQuery(query_params, null)
            if (cSor.moveToFirst()) {
                do {
                    var _id = cSor.getInt(cSor.getColumnIndexOrThrow(Staticated.COLUMN_ID))
                    var _artist =
cSor.getString(cSor.getColumnIndexOrThrow(Staticated.COLUMN_SONG_ARTIST))
                    var _title = cSor.getString(cSor.getColumnIndexOrThrow(Staticated.COLUMN_SONG_TITLE))
                    var _songPath =
cSor.getString(cSor.getColumnIndexOrThrow(Staticated.COLUMN_SONG_PATH))
                    _songList.add(Songs(_id.toLong(), _title, _artist, _songPath, 0))
                } while (cSor.moveToNext())
            } else {
                return null
            }
        } catch (e: Exception) {
            e.printStackTrace()
        }
        return _songList
    }

    fun checkifIdExists(_id: Int): Boolean {
        var storeId = 345
        val db = this.readableDatabase
        var query_params = "SELECT * FROM " + Staticated.TABLE_NAME + " WHERE SongId = '$_id'"
        var cSor = db.rawQuery(query_params, null)
        if (cSor.moveToFirst()) {
            do {
                storeId = cSor.getInt(cSor.getColumnIndexOrThrow(Staticated.COLUMN_ID))
            } while (cSor.moveToNext())
        } else {
            return false
        }
        return storeId != 345
    }

    fun deleteFavourite(_id: Int) {
        val db = this.writableDatabase
        db.delete(Staticated.TABLE_NAME, Staticated.COLUMN_ID + "=" + _id, null)
        db.close()
    }

    fun checkSize(): Int {
        var counter = 0
        val db = this.readableDatabase
        var query_params = "SELECT * FROM " + Staticated.TABLE_NAME
        var cSor = db.rawQuery(query_params, null)
        if (cSor.moveToFirst()) {
            do {
                counter = counter + 1
            } while (cSor.moveToNext())
        } else {
            return 0
        }
        return counter

    }

}
```

REFRENCES

REFERENCES

[1]Android,"MediaPlayer".Android Developers,13 feb. 2020,https://developer.android.com/reference/android/media/MediaPlayer.
[2]Internshala,"Android Tarining",2 jan 2020,https://trainings.internshala.com/android-training.