

```

package com.internshala.echo.fragments
import android.app.Activity
import android.content.Context
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.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
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.DbHandler
import com.cleveroad.audiovisualization.GLAudioVisualizationView
import com.internshala.echo.CurrentSongHelper
import com.internshala.echo.R
import com.internshala.echo.Songs
import com.internshala.echo.databases.EchoDatabase
import com.internshala.echo.fragments.SongPlayingFragment.Staticated.onSongComplete
import com.internshala.echo.fragments.SongPlayingFragment.Staticated.playNext
import com.internshala.echo.fragments.SongPlayingFragment.Staticated.processInformation
import com.internshala.echo.fragments.SongPlayingFragment.Staticated.updateTextViews
import com.internshala.echo.fragments.SongPlayingFragment.Statified.audioVisualization
import com.internshala.echo.fragments.SongPlayingFragment.Statified.currentPosition
import com.internshala.echo.fragments.SongPlayingFragment.Statified.currentSongHelper
import com.internshala.echo.fragments.SongPlayingFragment.Statified.endTimeText
import com.internshala.echo.fragments.SongPlayingFragment.Statified.fab
import com.internshala.echo.fragments.SongPlayingFragment.Statified.favoriteContent
import com.internshala.echo.fragments.SongPlayingFragment.Statified.fetchSongs
import com.internshala.echo.fragments.SongPlayingFragment.Statified.glView
import com.internshala.echo.fragments.SongPlayingFragment.Statified.loopImageButton
import com.internshala.echo.fragments.SongPlayingFragment.Statified.mediaPlayer
import com.internshala.echo.fragments.SongPlayingFragment.Statified.myActivity
import com.internshala.echo.fragments.SongPlayingFragment.Statified.nextImageButton
import com.internshala.echo.fragments.SongPlayingFragment.Statified.playPauseImageButton
import com.internshala.echo.fragments.SongPlayingFragment.Statified.previousImageButton
import com.internshala.echo.fragments.SongPlayingFragment.Statified.seekBar
import com.internshala.echo.fragments.SongPlayingFragment.Statified.shuffleImageButton
import com.internshala.echo.fragments.SongPlayingFragment.Statified.songArtistView
import com.internshala.echo.fragments.SongPlayingFragment.Statified.songTitleView
import com.internshala.echo.fragments.SongPlayingFragment.Statified.startTimeText
import com.internshala.echo.fragments.SongPlayingFragment.Statified.updateSongTime
import java.util.*
import java.util.concurrent.TimeUnit

/**
 * A simple [Fragment] subclass.
 */
class SongPlayingFragment : Fragment() {
    /*Here you may wonder that why did we create two objects namely Statified and
    Staticated respectively
    * These objects are created as the variables and functions will be used from another
    class
    * Now, the question is why did we make two different objects and not one single object
    * This is because we created the Statified object which contains all the variables and
    * the Staticated object which contain all the functions*/
    object Statified {
        var myActivity: Activity? = null
        var mediaPlayer: MediaPlayer? = null
        var startTimeText: TextView? = null
        var endTimeText: 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 songArtistView: TextView? = null
var songTitleView: TextView? = null
var currentPosition: Int = 0
var fetchSongs: ArrayList<Songs>? = null
var currentSongHelper: CurrentSongHelper? = null
/*Declaring variable for handling the favorite button*/
var fab: ImageButton? = null
/*Variable for using DB functions*/
var favoriteContent: EchoDatabase? = null
var audioVisualization: AudioVisualization? = null
var glView: GLAudioVisualizationView? = null
var updateSongTime = object : Runnable {
    override fun run() {
        val getCurrent = mediaPlayer?.currentPosition
        startTimeText?.setText(String.format("%d:%d",
            TimeUnit.MILLISECONDS.toMinutes(getCurrent?.toLong() as Long),
            TimeUnit.MILLISECONDS.toSeconds(TimeUnit.MILLISECONDS.toMinutes(getCurrent?.toLong() as
Long))))
        seekBar?.setProgress(getCurrent?.toInt() as Int)
        Handler().postDelayed(this, 1000)
    }
}
}

object Staticated {
    var MY_PREFS_SHUFFLE = "Shuffle feature"
    var MY_PREFS_LOOP = "Loop feature"
    fun onSongComplete() {
        if (currentSongHelper?.isShuffle as Boolean) {
            playNext("PlayNextLikeNormalShuffle")
            currentSongHelper?.isPlaying = true
        } else {
            if (currentSongHelper?.isLoop as Boolean) {
                currentSongHelper?.isPlaying = true
                var nextSong = fetchSongs?.get(currentPosition)
                currentSongHelper?.currentPosition = currentPosition
                currentSongHelper?.songPath = nextSong?.songData
                currentSongHelper?.songTitle = nextSong?.songTitle
                currentSongHelper?.songArtist = nextSong?.artist
                currentSongHelper?.songId = nextSong?.songID as Long
                updateTextViews(currentSongHelper?.songTitle as String,
currentSongHelper?.songArtist as String)
                mediaPlayer?.reset()
                try {
                    mediaPlayer?.setDataSource(myActivity,
Uri.parse(currentSongHelper?.songPath))
                    mediaPlayer?.prepare()
                    mediaPlayer?.start()
                } catch (e: Exception) {
                    e.printStackTrace()
                }
            } else {
                playNext("PlayNextNormal")
                currentSongHelper?.isPlaying = true
            }
        }
    }
}

```

```

        }
    }
    if (favoriteContent?.checkIfIdExists(currentSongHelper?.songId?.toInt() as Int)
as Boolean) {
        fab?.setBackgroundResource(R.drawable.favorite_on)
    } else {
        fab?.setBackgroundResource(R.drawable.favorite_off)
    }
}

fun updateTextViews(songTitle: String, songArtist: String) {
    songTitleView?.setText(songTitle)
    songArtistView?.setText(songArtist)
}

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

fun playNext(check: String) {
    if (check.equals("PlayNextNormal", true)) {
        currentPosition = currentPosition + 1
    } else if (check.equals("PlayNextLikeNormalShuffle", true)) {
        var randomObject = Random()
        var randomPosition = randomObject.nextInt(fetchSongs?.size?.plus(1) as
Int)

        currentPosition = randomPosition
    }
    if (currentPosition == fetchSongs?.size) {
        currentPosition = 0
    }
    currentSongHelper?.isLoop = false
    var nextSong = fetchSongs?.get(currentPosition)
    currentSongHelper?.songPath = nextSong?.songData
    currentSongHelper?.songTitle = nextSong?.songTitle
    currentSongHelper?.songArtist = nextSong?.artist
    currentSongHelper?.songId = nextSong?.songID as Long
    updateTextViews(currentSongHelper?.songTitle as String,
currentSongHelper?.songArtist as String)
    mediaPlayer?.reset()
    try {
        mediaPlayer?.prepare()
        mediaPlayer?.start()
        processInformation(mediaPlayer as MediaPlayer)
    } catch (e: Exception) {
        e.printStackTrace()
    }
    if (favoriteContent?.checkIfIdExists(currentSongHelper?.songId?.toInt() as Int)
as Boolean) {

```

```

        fab?.setBackgroundResource(R.drawable.favorite_on)
    } else {
        fab?.setBackgroundResource(R.drawable.favorite_off)
    }
}

override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                           savedInstanceState: Bundle?): View? {
    val view = inflater!!.inflate(R.layout.fragment_song_playing, container, false)
    seekBar = view?.findViewById(R.id.seekBar)
    startTimeText = view?.findViewById(R.id.startTime)
    endTimeText = view?.findViewById(R.id.endTime)
    playPauseImageButton = view?.findViewById(R.id.playPauseButton)
    nextImageButton = view?.findViewById(R.id.nextButton)
    previousImageButton = view?.findViewById(R.id.previousButton)
    loopImageButton = view?.findViewById(R.id.loopButton)
    shuffleImageButton = view?.findViewById(R.id.shuffleButton)
    songArtistView = view?.findViewById(R.id.songArtist)
    /*Linking it with the view*/
    fab = view?.findViewById(R.id.favoriteIcon)
    /*Fading the favorite icon*/
    fab?.alpha = 0.8f
    glView = view?.findViewById(R.id.visualizer_view)
    return view
}

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

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

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

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

override fun onPause() {
    audioVisualization?.onPause()
    super.onPause()
}

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

override fun onActivityCreated(savedInstanceState: Bundle?) {
    super.onActivityCreated(savedInstanceState)
    /*Initialising the database*/
    favoriteContent = EchoDatabase(myActivity)
    currentSongHelper = CurrentSongHelper()
    currentSongHelper?.isPlaying = true
    currentSongHelper?.isLoop = false
    currentSongHelper?.isShuffle = false
    var path: String? = null
    var _songTitle: String? = null

```

```

var _songArtist: String? = null
var songId: Long = 0
try {
    path = arguments.getString("path")
    _songTitle = arguments.getString("songTitle")
    _songArtist = arguments.getString("songArtist")
    songId = arguments.getInt("songId").toLong()
    currentPosition = arguments.getInt("position")
    fetchSongs = arguments.getParcelableArrayList("songData")
    currentSongHelper?.songPath = path
    currentSongHelper?.songTitle = _songTitle
    currentSongHelper?.songArtist = _songArtist
    currentSongHelper?.songId = songId
    currentSongHelper?.currentPosition = currentPosition
    updateTextViews(currentSongHelper?.songTitle as String,
currentSongHelper?.songArtist as String)
} catch (e: Exception) {
    e.printStackTrace()
}

    /*Here we check whether we came to the song playing fragment via tapping on a song
or by bottom bar*/
var fromFavBottomBar = arguments.get("FavBottomBar") as? String
if (fromFavBottomBar != null) {

    /*If we came via bottom bar then the already playing media player object is
used*/

    Statified.mediaPlayer = FavoriteFragment.Statified.mediaPlayer
} else {

    /*Else we use the default way*/
    mediaPlayer = MediaPlayer()
    mediaPlayer?.setAudioStreamType(AudioManager.STREAM_MUSIC)
    try {
        mediaPlayer?.setDataSource(myActivity, Uri.parse(path))
        mediaPlayer?.prepare()
    } catch (e: Exception) {
        e.printStackTrace()
    }
    mediaPlayer?.start()
}
processInformation(mediaPlayer as MediaPlayer)
if (currentSongHelper?.isPlaying as Boolean) {
    playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
} else {
    playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
}
mediaPlayer?.setOnCompletionListener {
    onSongComplete()
}
clickHandler()
var visualizationHandler = DbmHandler.Factory.newVisualizerHandler(myActivity as
Context, 0)
audioVisualization?.linkTo(visualizationHandler)
var prefsForShuffle = myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE,
Context.MODE_PRIVATE)
var isShuffleAllowed = prefsForShuffle?.getBoolean("feature", false)

```

```

        if (isShuffleAllowed as Boolean) {
            currentSongHelper?.isShuffle = true
            currentSongHelper?.isLoop = false
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_icon)
            loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
        } else {
            currentSongHelper?.isShuffle = false
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
        }
        var prefsForLoop = myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP,
Context.MODE_PRIVATE)
        var isLoopAllowed = prefsForLoop?.getBoolean("feature", false)
        if (isLoopAllowed as Boolean) {
            currentSongHelper?.isShuffle = false
            currentSongHelper?.isLoop = true
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
            loopImageButton?.setBackgroundResource(R.drawable.loop_icon)
        } else {
            loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
            currentSongHelper?.isLoop = false
        }
        /*Here we check that if the song playing is a favorite, then we show a red colored
heart indicating favorite else only the heart boundary
        * This action is performed whenever a new song is played, hence this will done in
the playNext(), playPrevious() and onSongComplete() methods*/
        if (favoriteContent?.checkifIdExists(currentSongHelper?.songId?.toInt() as Int) as
Boolean) {
            fab?.setBackgroundResource(R.drawable.favorite_on)
        } else {
            fab?.setBackgroundResource(R.drawable.favorite_off)
        }
    }
    fun clickHandler() {
        /*Here we handle the click of the favorite icon
        * When the icon was clicked, if it was red in color i.e. a favorite song then we
remove the song from favorites*/
        fab?.setOnClickListener({
            if (favoriteContent?.checkifIdExists(currentSongHelper?.songId?.toInt() as Int)
as Boolean) {
                fab?.setBackgroundResource(R.drawable.favorite_off)
                favoriteContent?.deleteFavourite(currentSongHelper?.songId?.toInt() as
Int)

                /*Toast is prompt message at the bottom of screen indicating that an
action has been performed*/
                Toast.makeText(myActivity, "Removed from Favorites",
Toast.LENGTH_SHORT).show()
            } else {
                /*If the song was not a favorite, we then add it to the favorites using
the method we made in our database*/
                fab?.setBackgroundResource(R.drawable.favorite_on)
                favoriteContent?.storeAsFavorite(currentSongHelper?.songId?.toInt(),
currentSongHelper?.songArtist, currentSongHelper?.songTitle, currentSongHelper?.songPath)
                Toast.makeText(myActivity, "Added to Favorites",
Toast.LENGTH_SHORT).show()
            }
        })
        shuffleImageButton?.setOnClickListener({
            var editorShuffle =
myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE, Context.MODE_PRIVATE)?.edit()
            var editorLoop = myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP,
Context.MODE_PRIVATE)?.edit()

```

```

        if (currentSongHelper?.isShuffle as Boolean) {
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
            currentSongHelper?.isShuffle = false
            editorShuffle?.putBoolean("feature", false)
            editorShuffle?.apply()
        } else {
            currentSongHelper?.isShuffle = true
            currentSongHelper?.isLoop = false
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_icon)
            loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
            editorShuffle?.putBoolean("feature", true)
            editorShuffle?.apply()
            editorLoop?.putBoolean("feature", false)
            editorLoop?.apply()
        }
    })
    nextImageButton?.setOnClickListener({
        currentSongHelper?.isPlaying = true
        if (currentSongHelper?.isShuffle as Boolean) {
            playNext("PlayNextLikeNormalShuffle")
        } else {
            playNext("PlayNextNormal")
        }
    })
    previousImageButton?.setOnClickListener({
        currentSongHelper?.isPlaying = true
        if (currentSongHelper?.isLoop as Boolean) {
            loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
        }
        playPrevious()
    })
    loopImageButton?.setOnClickListener({
        var editorShuffle =
myActivity?.getSharedPreferences(Staticated.MY_PREFS_SHUFFLE, Context.MODE_PRIVATE)?.edit()
        var editorLoop = myActivity?.getSharedPreferences(Staticated.MY_PREFS_LOOP,
Context.MODE_PRIVATE)?.edit()
        if (currentSongHelper?.isLoop as Boolean) {
            currentSongHelper?.isLoop = false
            loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
            editorLoop?.putBoolean("feature", false)
            editorLoop?.apply()
        } else {
            currentSongHelper?.isLoop = true
            currentSongHelper?.isShuffle = false
            loopImageButton?.setBackgroundResource(R.drawable.loop_icon)
            shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
            editorShuffle?.putBoolean("feature", false)
            editorShuffle?.apply()
            editorLoop?.putBoolean("feature", true)
            editorLoop?.apply()
        }
    })
    playPauseImageButton?.setOnClickListener({
        if (mediaPlayer?.isPlaying as Boolean) {
            mediaPlayer?.pause()
            currentSongHelper?.isPlaying = false
            playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
        } else {
            mediaPlayer?.start()

```

```

        currentSongHelper?.isPlaying = true
        playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
    }
})
}
fun playPrevious() {
    currentPosition = currentPosition - 1
    if (currentPosition == -1) {
        currentPosition = 0
    }
    if (currentSongHelper?.isPlaying as Boolean) {
        playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
    } else {
        playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
    }
    currentSongHelper?.isLoop = false
    var nextSong = fetchSongs?.get(currentPosition)
    currentSongHelper?.songPath = nextSong?.songData
    currentSongHelper?.songTitle = nextSong?.songTitle
    currentSongHelper?.songArtist = nextSong?.artist
    currentSongHelper?.songId = nextSong?.songID as Long
    updateTextViews(currentSongHelper?.songTitle as String,
currentSongHelper?.songArtist as String)
    mediaPlayer?.reset()
    try {
        mediaPlayer?.setDataSource(myActivity, Uri.parse(currentSongHelper?.songPath))
        mediaPlayer?.prepare()
        mediaPlayer?.start()
        processInformation(mediaPlayer as MediaPlayer)
    } catch (e: Exception) {
        e.printStackTrace()
    }
    if (favoriteContent?.checkIfIdExists(currentSongHelper?.songId?.toInt() as Int) as
Boolean) {
        fab?.setBackgroundResource(R.drawable.favorite_on)
    } else {
        fab?.setBackgroundResource(R.drawable.favorite_off)
    }
}
}
}

```