COMP3040 Coursework 1-2 MP3Player

Summary

In this coursework you are required to build an Android MP3 player application. This is an assessed coursework and will account for **15% of your final module mark**. This is an individual coursework, and your submission must be entirely your own work – please pay particular attention to the section of this document regarding plagiarism. This document sets out general requirements and broad instructions for developing the application.

Your application should be submitted no later than:

- **5pm on Wednesday, 2 Dec 2020**

Submissions should be made electronically via Moodle. Standard penalties of 5% per working day will be applied to late submissions.

Your application should be submitted as a .zip or .tar.gz file containing all relevant source code, configuration and related files, and a compiled .apk file – i.e. the contents of the directory containing your Android Studio project. Do not submit RAR files.

Specification

You should create an application with the functionality of a simple music player for Android, which allows users to select from a number of music files stored on the SD card storage of the device to be played, and allows the music to continue to play in the background while the user performs other tasks.

Your application must consist of:

- An *Activity* presenting an interface for the user that:
  - Displays and allows the user to select from and play music files from the /sdcard/Music folder
  - Allows the user to stop or pause playback
  - Displays the current progress of the playback (i.e. the elapsed time)
- A *Service* that provides continued playback in the background
- A *Notification* that is displayed while the music is playing

You must implement a **Service** to handle the music-playing element of the application, as this is a long-running task and the user can be expected to leave the initial activity to perform other tasks. You should think carefully about the relationship between the Activity and Service in your application, and how these should be used appropriately to perform the task. There is **no requirement** that your service will be used remotely, i.e. you do not need to use *AIDL*.

A simple MP3Player class is provided that wraps a basic MediaPlayer object for loading and playing an MP3 file. It is left up to you to decide how best to design and implement Activities for selecting and controlling the music playback.

Your application must be written in Java and make use of the Android SDK. There are no requirements to target a specific Android API version, however you should assume that your application would be tested on an emulater running Android API version 27 (Android 8.1 Oreo).

You should consider the following when implementing your application:

- Appropriate use of Activities, Intents and appreciation of the Activity life-cycle
- Appropriate use of Widgets and ViewGroups for layouts that support devices of differing screen sizes and resolutions
- Appropriate use of Services, Notifications and management of the Service life-cycle
- Appropriate communication between components
- Your application should have appropriate comments and variable / class names, so that a reader can easily understand how it works at the code level

Plagiarism

N.B. Use of third party assets (tutorials, images, example code, libraries etc.) MUST be credited and referenced, and you MUST be able to demonstrate that they are available under a license that allows their reuse.

Making significant use of tutorial code while referencing it is poor academic practice, and will result in a lower mark that reflects the significance of your own original contribution.

Copying code from other students, from previous students, from any other source, or soliciting code from online sources and submitting it as your own is plagiarism and will be penalized as such. FAILING TO ATTRIBUTE a source will result in a mark of zero – and can potentially result in failure of coursework, module or degree.

All submissions are checked using both plagiarism detection software and manually for signs of cheating. If you have any doubts, then please ask.

Assessment Criteria

	Marks
Basic functionality	
The application has an Activity that displays audio files on the device	1
The application has a Service that handles playing the track when another application is in foreground use	4
The Activity allows the user to select, play, pause and stop a track	1
The Activity displays the current progress of the playback	2
The application displays a notification when a track is playing	1
Best Practice	
The application supports appropriate navigation, and appropriate communication between and management of components	6
Total	15

There are no additional marks available for additional functionality in this coursework.

Instructions

Note that this coursework aims to serve as an assessment of the Services material covered in lectures and used in lab exercises – as such try to think about how to make use of the concepts covered in those. It is a mistake to start by googling "how to build an mp3 player app".

MP3Player

Begin by creating a new application in Android Studio as usual.

Add the class MP3Player.java to your app project (you may change the project name to suit your requirements). This class is available on Moodle.

You can either copy the file directly into your project's source directory (*…/projectname/app/src/main/java/…*) or create a new MP3Player Java class in your project and copy / paste the code into it, updating the package qualifier accordingly.

MP3Player is a simplistic wrapper for an Android MediaPlayer object. MediaPlayer has its own internal thread for actually doing the work of playing an MP3, so there is no need to spawn a new thread to contain it. There is also no need to asynchronously load an MP3 in a separate thread. It is worth adding the MP3Player class directly to an Activity and controlling it directly with buttons to make sure you understand how it works before moving it into a Service.

The MP3Player has a *state* variable that reflects the stateful nature of the underlying MediaPlayer, and getState() will return one of the following:

```
public enum MP3PlayerState {
ERROR,
PLAYING,
PAUSED,
```

```
    STOPPED
    }
```

The MP3Player begins in the *STOPPED* state on instantiation.

The class has a few simple methods for loading and playing an MP3:

```
    public void load(String filePath)
```

attempts to load and play the file specified by *filePath*. If all goes well the music will start playing and the MP3Player will now be in the *PLAYING* state. If something went wrong – usually if the file is not found, or is not a playable type, the MP3Player will be in the *ERROR* state.

The music can be paused or unpaused when playing:

```
    public void play()
    public void pause()
```

Or finally stopped:

```
    public void stop()
```

Note that stopping releases and cleans up the MediaPlayer, so the MP3 must be loaded again from the beginning if it needs to start playing again.

The duration and current position of the MP3 can be queried, in milliseconds, and so can the current filename:

```
    public int getDuration()
    public int getProgress()
    public String getFilePath()
```

Files

To transfer mp3 files onto the **sdcard** of the emulator you can drag them from your computer onto the emulator display, and they will be copied to the Downloads folder. From there, move them into the Music folder either using the Files app, or using the command line via **adb shell, cd /sdcard/** to change to the external storage directory. Alternatively you may use the Device File Explorer within Android Studio to browse the device's storage, or directly push files from the command line via adb:

adb push myfile.mp3 /sdcard/Music/myfile.mp3

If you don't have access to any music in mp3 format, a variety of royalty free / creative commons licensed files are available here https://freemusicarchive.org/home

The Android permissions system by default prevents the application from reading from the sdcard, so you will need to add the READ_EXTERNAL_STORAGE permission to the manifest as shown below:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

You will also need to enable this permission from within the phone settings via

Settings->Apps->**My MP3Player**->Permissions->Storage

The code below handles much of the work of enumerating audio files on the device by querying the MediaStore, and which are primarily stored in the /Music/ directory of the sdcard, and then populating a ListView with the resultant list. An alternative is to open the /sdcard/Music location as a directory and to list the files in the usual way although this is generally discouraged due to Android's increasing abstraction over the underlying file system. In general the MediaStore will also find audio files elsewhere on the device.

onItemClick is called when one of the entries in the list is selected by the user. You are free to extend this code as you wish.

```
import android.widget.ListView;
import android.database.Cursor;
import android.provider.MediaStore;
import android.widget.AdapterView;
import android.widget.SimpleCursorAdapter;

final ListView lv = (ListView) findViewById(R.id.listView);

Cursor cursor = getContentResolver().query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
        null,
        MediaStore.Audio.Media.IS_MUSIC + "!= 0",
        null,
        null);

lv.setAdapter(new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_1,
        cursor,
        new String[] { MediaStore.Audio.Media.DATA},
        new int[] { android.R.id.text1 }));

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(AdapterView<?> myAdapter,
                            View myView,
                            int myItemInt,
                            long mylng) {

         Cursor c = (Cursor) lv.getItemAtPosition(myItemInt);
         String uri = c.getString(c.getColumnIndex(MediaStore.Audio.Media.DATA));
         Log.d("g53mdp", uri);
         // do something with the selected uri string...
   }
});
```

Note that ListView can be found in Legacy->ListView in the view design palette. It has largely been superseded by RecyclerView in newer Android API versions, however is used here for simplicity.

https://developer.android.com/guide/topics/ui/layout/recyclerview

Also, MediaStore.Audio.Media.DATA is technically deprecated, however is used here again for convenience given that Android media storage is such a mess.

Notes

You might notice that if the phone is allowed to go to sleep, the service is *eventually* destroyed. The solution here is to hold a partial wake lock via the power manager, however there is no requirement to do this for this coursework.

You may edit MP3Player.java as you see fit.

You do not need to include any music with your submission as resource files or otherwise. The application should allow files in the /sdcard/Music/ directory to be played and this is how it will be tested.

If you cannot see any music files check:
a) that you have put them in the right place on the device and
b) that you have enabled the permission.

Your application will be tested by pushing the apk contained within your submission onto a device and running it, and as such you should ensure that
a) there is an apk included and
b) that it works as intended when deployed and run as such.

References

http://developer.android.com/guide/components/services.html

http://developer.android.com/guide/topics/ui/notifiers/notifications.html