A Project Report

on

Android Multi-Media Player

Submitted in partial fulfillment of the requirement for the award ofthe degree of

B. TECH COMPUTER SCIENCE ENGINEERING

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr.M.Thirunavukkarasn

Submitted By:
Nikhil Ranjan
18SCSE1010361

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA MONTH

October 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **" Android Multi-Media Player"** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Science and Engineering** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater **Dr.M.Thirunavukkarasn** Assistant Professor , Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Nikhil Ranjan(18SCSE1010361)This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr.M.Thirunavukkarasn

Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of Nikhil Ranjan (18SCSE1010361) has been held on DEC 2021 and his work is recommended for the award of **Bachelor of Technology in Computer Science and Engineering.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: DECEMBER, 2021

Place: Greater Noida

ABSTRACT

In latest years, the emergence of clever phones has modified the definition of cell telephones. The phone is not just a communique tool, but additionally a crucial part of the humans' communique and everyday lifestyles. And this same thing is with music, it is also a part of life nowadays. Everyone in this world loves a different type of music and to listen is they had to either download it or to visit online site for it.Just to provide a better and useful platform for music lovers I came up with the idea of an Android App. This app is going to be small in size but larger in functionalities as compared to other applications. The UI is going to be great to get a better UX.Media players Are One Of The Most Used And Important Software Application In Today's World. Maximum Computer Users Switch On To The Media Player As Soon As They Start The Computers And Then Move To Their Respective Woks. Today's Era Is To Do Work With High Efficiency But At The Same Time, It Should Consume Very Less Time. And Thus Answer To The Problems Arising In The Use Of Traditional Media Players And The Lack Of Features In Proposed Media Players Is "Versatile Media Player" Versatile Media Player Is A Unique Player Developed To Fulfill Maximum Requirements Of User Regarding Audio And Video Songs. This Player Gives Numerous Facilities Which Differentiate It From The Conventional Media Players. The Various Features Included In The Versatile Media Player Are Shut-Down Facility, Alarm Facility, Lyrics Display, Splitting Windows, Access To More Than One Media File Simultaneously

LIST OF FIGURES

LIST OF CONTENTS

INTRODUCTION

Media players have commanded tremendous notice over the previous decade and pulled in most PC clients in this manner making the clients dependent on recordings and music. In the course of the last decade, human-PC cooperation has become a functioning exploration region, which deliveries individuals from latent, firm correspondence with the machine. Most extreme PC clients switch on to the media player when they start the PCs and afterward move to their works. Additionally, many have a propensity for hauling the melodies into the rundown sheet of the media player, check out music, and afterward work. So we realize what amount are the media player utilized and required. Communicating with PCs cleverly make a critical commitment to the future use of human-PC collaboration. Today's time is to tackle the job with high productivity and yet, it ought to devour next to no time. Also, accordingly, reply to the issue emerging in the utilization of customary media players and the absence of elements inaccessible media players in the planning and advancement of the Versatile Media Player. In this paper, we intend to create a Flexible Media Player which will satisfy the most extreme prerequisites of the client concerning sound and recordings. This player gives various offices which separate it from the ordinary media players. The different components remembered for the Versatile Media Player are Shut-down office, Alarm office, Lyrics show, Splitting windows,
Admittance to more than one media records at the same time. Consequently, we can say Versatile Media Player is a completely stacked media player and a superior choice over the conventional media players. Flexible Media Player is a decent and dynamic choice for music darlings, PC clients, and that load of dynamic individuals and love changes.

1.2 Formulation of Problem

1.2.1 Bloated software and user interfaces

Due to the fierce competition between music player applications, many developers tried to add many features, advertise, and content to their respective music players to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. With the continuous iteration of the application and a growing number of features, the music player will become even more bloated and the user's experience will become less smooth. Based on Mehul (2018), users tend to feel frustrated and angry if they take a long time to get a reply from the mobile application, so they will never return to the same application, and 48% of users will simply uninstall or stop using it.

1.2.2 Lack of gestures to control

Most music player apps use touch buttons to play, pause and switch between previous and next songs while ignoring the convenience of using gesture swiping to control the music player. For instance, when a user is working and intends to skip to the next song in the music player, he/she have to switch their attention to the console from work and click the button. This problem does not affect music players' proper work, but it does have some inconvenience. However, Scacca (2020) said that as our physical devices and appliances develop the button-free design, consumers will become more comfortable and confident in this way of interaction, so we should consider using gesture control on more mobile applications.

1.2.3 Lack of sorting and searching features

When users continuously add new songs into the playlist, the difficulty of the songs the user wants to filter will increase. After the songs in the playlist are added to reach hundreds of songs, the user can only search songs by continuously swipe up or down. If not carefully check the content, it is possible to miss the songs that the user wants to filter, and then repeat the behavior until the result is found. Therefore, it is an extremely poor experience for users

About Music Player App:

The music player app that we will develop in this article would allow the users to play the songs present on the device. You can download songs on your device and then use the music player to play those songs. Let’s see the quite exciting features that you get along with this app.

Features of Music Player App:

1. Alluring user interface with loaded animations and music bars.
2. You can access all the songs present on your device.
3. Play your songs on the go without even having an internet connection.
4. Play or pause your music.
5. Listen to music even on your device background—no need to keep the app open all the time.
6. Music Visualizer that helps you to visualize the beats of a song.

LITERATURE SURVEY

There are various media player available in this world developed and manufactured by various other companies. The variety of players provide user with the plenty of features and characteristics. As it goes the saying that you cannot have happiness without sorrows, applies here. Every player has a drawback which pulls it back from the race. The pros and cons of the various media players are given in details here. And to overcome these problems we are devising the versatile media player

2.1 METHODOLOGY:

The technologies used to develop this system were as follows:-

Working on android studio with the help of Java, XML, and another language according to the need of frontend and backend. Such as:-

Android Studio with :-

- SDK tools
- SDK platform tools
- SDK build tools

And other libraries which are required are as:-

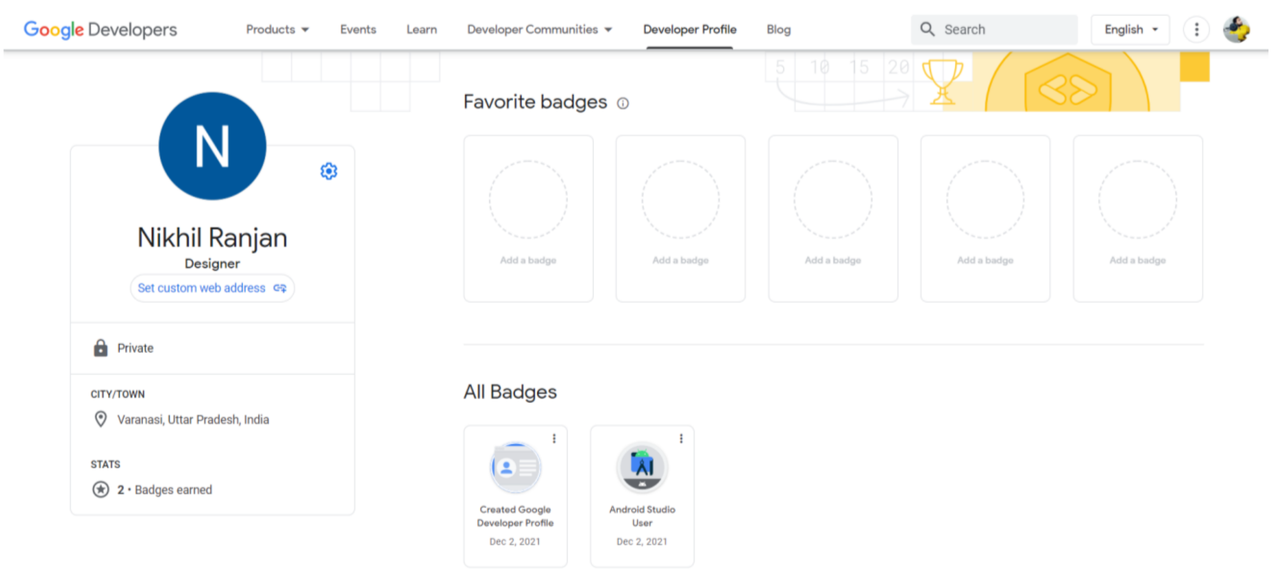
- Google APIs
- ARM EABI V7A System Image
- Intel x86 Atom_ 64 System image.. ETC..

Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator

- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- Implementing MediaPlayer class and using its methods like pause, play and stop.
- Using external files(images, audios, etc) in our project.
- Building the interface of our Music Player Android App.



Nikhil Ranjan
Designer
Set custom web address
Private
CITY/TOWN
Varanasi, Uttar Pradesh, India
STATS
2 • Badges earned

Created Google Developer Profile
Dec 2, 2021

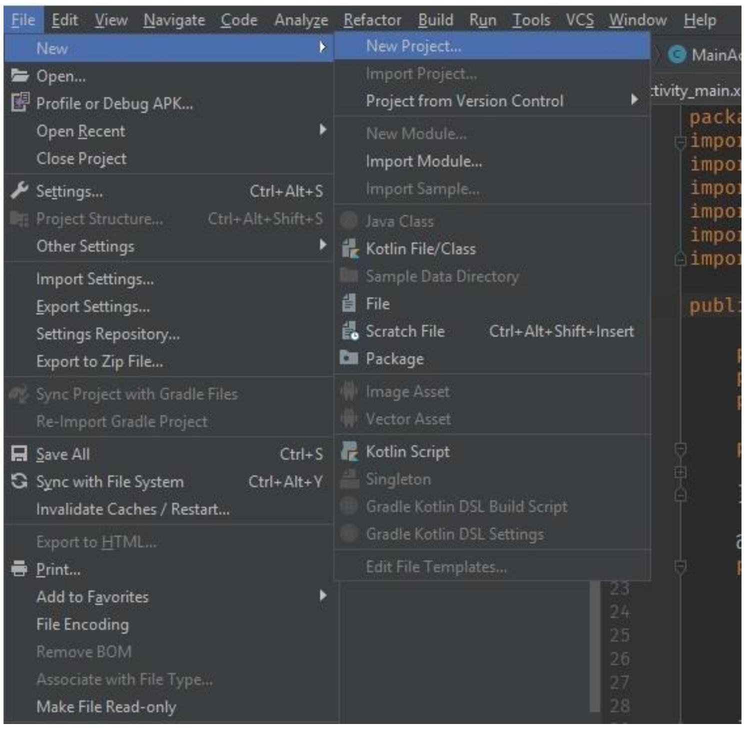
Android Studio User
Dec 2, 2021

After finishing all the following subsequent steps our app will look like this:

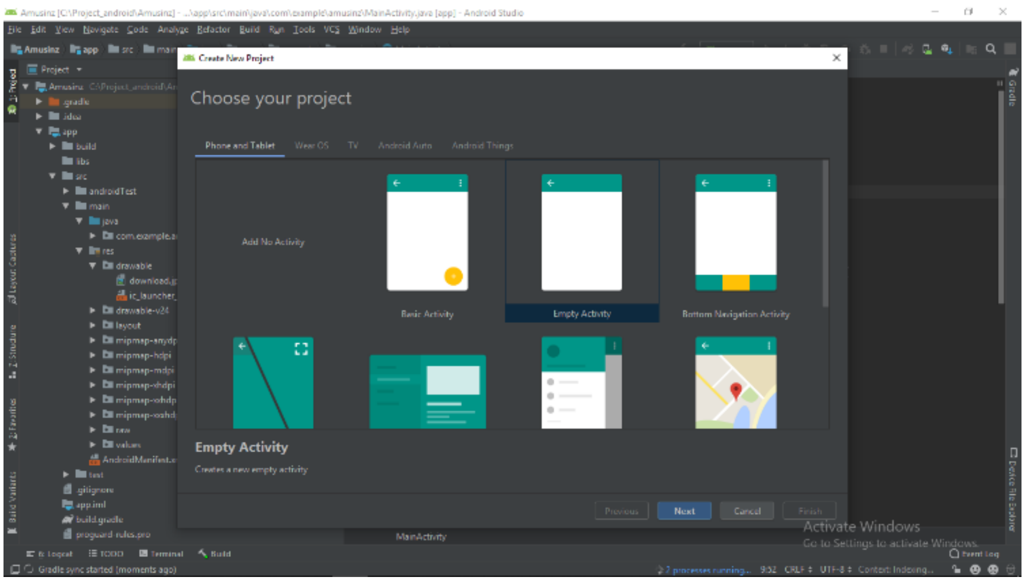
Step 1: Open a new android project
After opening the Android Studio you have to create a **new project** using

the **Empty Activity** with language as **Java** and give your project a unique name as you wish but don't forget to keep the first alphabet capital.

1. Go to the top left corner and then hit **File->New->New Project** as shown in the following screenshot.



2.Select **Empty Activity** as shown in the following screenshot.
As shown in the fig. below.

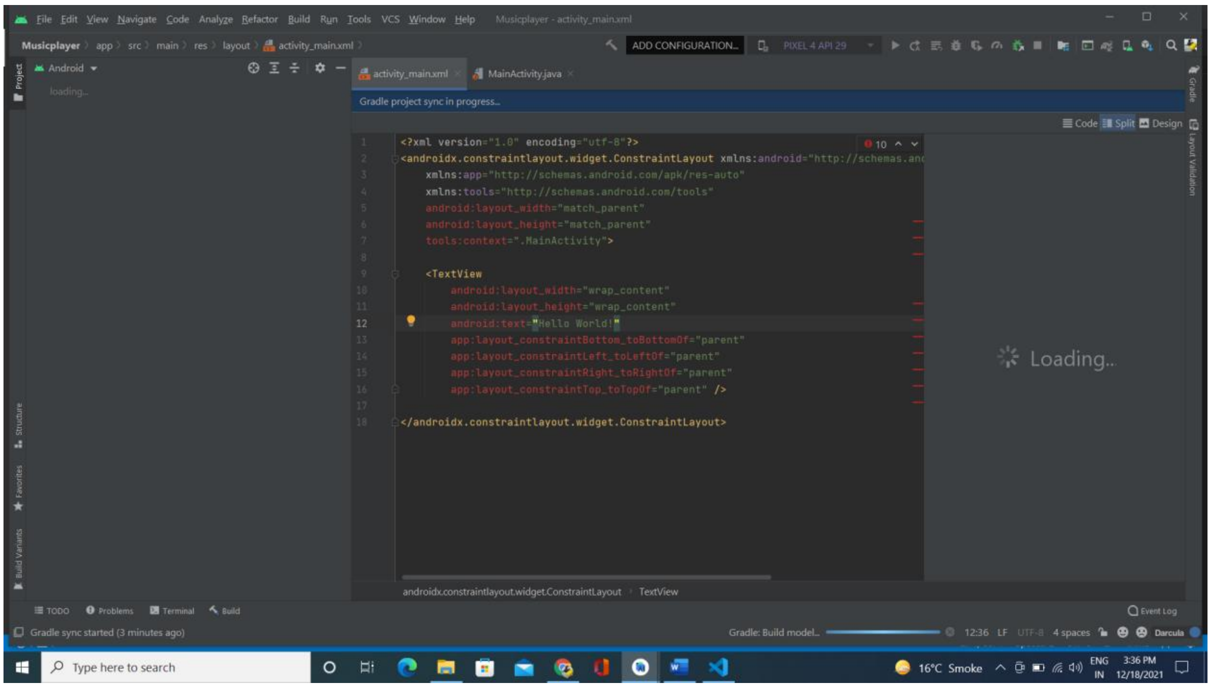


Step 2: Designing the User Interface of the app

In this app, we have used 4 components:

- **a imageView– to show our given image for the song**
- **3 Buttons:**
    - **a play button to play our song**
    - **a pause button to pause our song**
    - **a stop button to stop our song**

after stop then our song will play from the beginning) (Note: if we press play after pressing the pause then our song will continue playing immediately after where it was paused but if we press play button



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Musicplayer - activity_main.xml
Musicplayer > app > src > main > res > layout > activity_main.xml
ADD CONFIGURATION...
PIXEL 4 API 29
Android
activity_main.xml
MainActivity.java
Gradle project sync in progress...
Code
Split
Design
Loading..
TODO
Problems
Terminal
Build
Event Log
Gradle sync started (3 minutes ago)
Gradle: Build model...
Type here to search

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

These components are implemented on the below two layouts:

- **Vertical LinearLayout**
- **Horizontal LinearLayout**

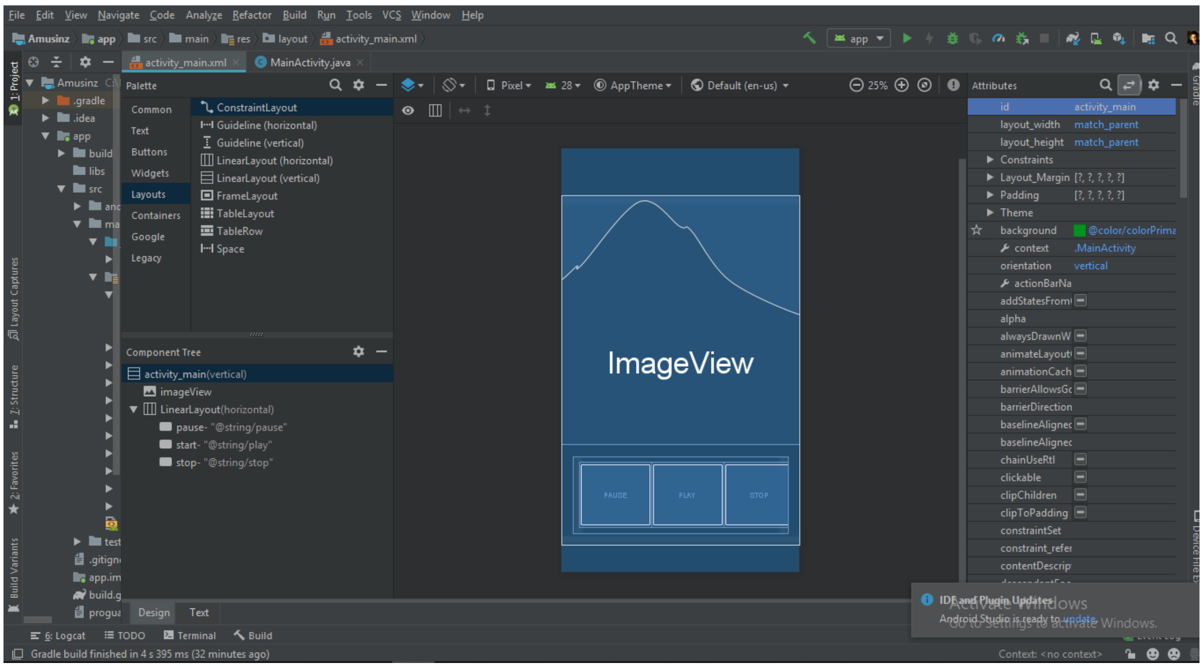
Inside the LinearLayout (vertical) there are two components:

- imageView component
- LinearLayout(horizontal)

This layout will vertically divide our app screen in two halves. The **imageView** component will be on upper half and the **Horizontal Linear**

Layout will be on the lower half. The horizontal layout will contain three buttons (play, pause and stop button). This horizontal layout will align these three buttons one after another horizontally on the lower half of our app screen.

To understand this clearly, please go through the following blue print and **Component Tree** of our app:



Component Tree
activity_main(vertical)
imageView
LinearLayout(horizontal)
pause- "@string/pause"
start- "@string/play"
stop- "@string/stop"

id	activity_main
layout_width	match_parent
layout_height	match_parent
orientation	vertical
background	@color/colorPrima
context	.MainActivity

Gradle build finished in 4 s 395 ms (32 minutes ago)

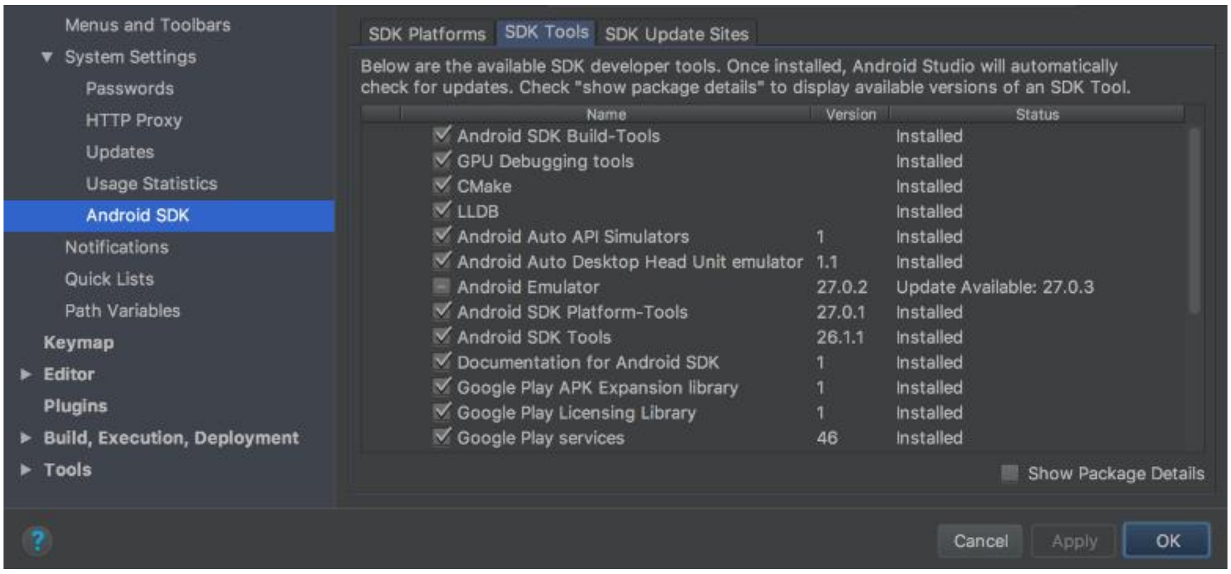
2.2 REQUIRED TOOLS:-

- A software development toolkit is set of software tools and programs provided by hardware and software vendors that developers can use to build applications for specific platforms. These providers make their SDKs available to help developers easily integrate their apps with their services.
- Android **SDK Platform-Tools** is a component for the Android SDK. It includes tools that interface with the Android platform, such as ADB, fastboot, and systrace. These tools are required for Android app development. They're also needed if you want to unlock your device

bootloader and flash it with a new system image.

- Android **SDK Build-Tools** is a component of the Android SDK required for building Android apps. It's installed in the<sdk>/build-tools/ directory.

- **Google APIs** are a set of application programming interfaces (API) developed by Google which allows communication with Google Services. APIs adhere to specific rules and methods to communicate requests and responses.



Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		Installed
✓ GPU Debugging tools		Installed
✓ CMake		Installed
✓ LLDB		Installed
✓ Android Auto API Simulators	1	Installed
✓ Android Auto Desktop Head Unit emulator	1.1	Installed
– Android Emulator	27.0.2	Update Available: 27.0.3
✓ Android SDK Platform-Tools	27.0.1	Installed
✓ Android SDK Tools	26.1.1	Installed
✓ Documentation for Android SDK	1	Installed
✓ Google Play APK Expansion library	1	Installed
✓ Google Play Licensing Library	1	Installed
✓ Google Play services	46	Installed

- Using the Google Play Developer API, you can automate a variety of app-management tasks, including:
  - ➢ Uploading and releasing new versions of your app
  - ➢ Editing your app Google Play Store listings, including localized text and graphics.

    - Managing your in-app product catalog, your products purchase status, and your app subscriptions

- **ARM EABI V7A System Image** The system image is used by the emulator to virtualize the Android OS so Apps can be tested/debugged on a PC without the need to upload the App to a phone whenever a change in the code is made. This speeds up the development process and allows testing for phones that aren't physically available.
- **Dexter** We all know that Android Marshmallow introduced runtime permissions letting user to allow or deny any permission at runtime. Implementing runtime permissions is a tedious process and developer needs to write lot of code just to get a single permission.
- we are going to simplify the process of adding the runtime permissions using Dexter library. Using this library, the permissions can be implemented in few minutes.

- **onPermissionGranted()** will be called once the permission is granted.
- **onPermissionDenied()** will be called when the permission is denied. Here you can check whether the permission is permanently denied by using response.isPermanentlyDenied() condition.

- Google recently released a developer preview of the upcoming Android 11 and this shows just how far android development has come over the last few years. One of the major changes in regards to development came in Android 6 i.e Marshmallow and this was how apps dealt with user permissions.

- Permissions authorize app access to the user’s device resources and data and are mainly grouped into two types of permissions**:**

- **Normal Permissions**

- This refers to permissions that cover the app’s need to access device resources but that does not pose any danger to the user’s privacy and functionality of other apps. An example is the permission to access the Internet.

- As long as the permission is declared in the app’s **AndroidManifest.xml**, the system automatically grants the app access to the defined permissions and the user does not have the option of revoking them.

- **Dangerous Permissions**

- This refers to permissions that grant the app access to potentially manipulate the user’s stored data and could infringe on the user’s personal information or affect the functionality of other apps. An example is the permission to access the user’s contact list or device storage.

- In devices prior to Marshmallow, once declared in the **AndroidManifest.xml,** the permissions were requested at install time from the Google Play Store and the app would only install once the user granted them.

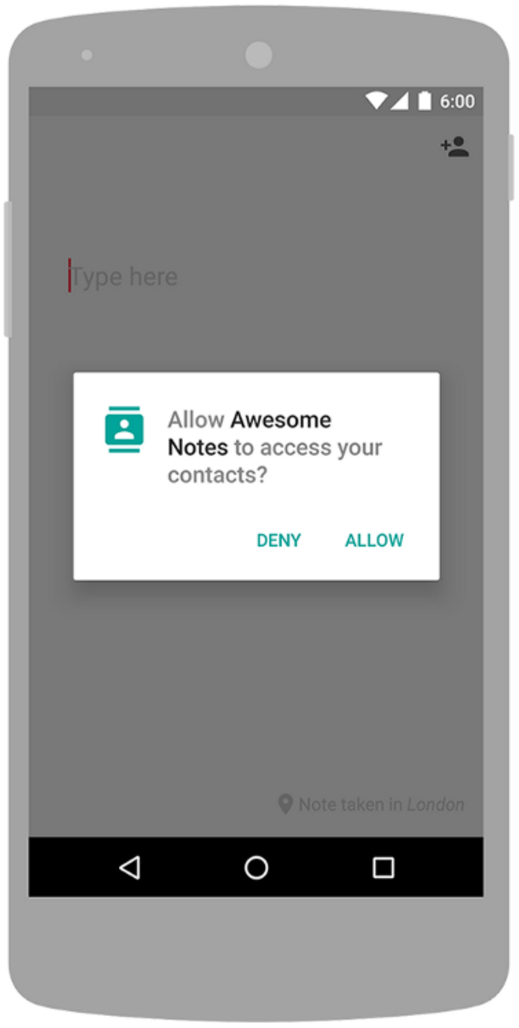
- Starting from Marshmallow however, the app explicitly needs to request the permission during run time by prompting the user to grant access.

- In this article, we are going to see a simple and convenient way to request permissions on the fly.

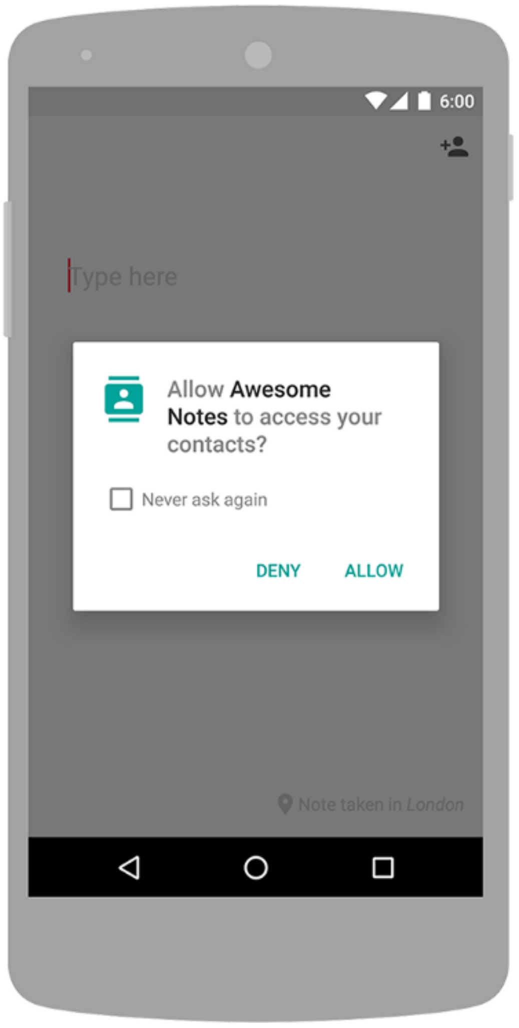
DEXTER code for premission

```
class MainActivity
:
AppCompatActivity()
{
                          ...


                          // Request for permission
                          private fun requestContactsPermission() {
                              Dexter.withActivity(this)
                                  .withPermission(Manifest.permission.READ_CONTACTS)
                                  .withListener(object : PermissionListener {
                                      override fun onPermissionGranted(response:
                      PermissionGrantedResponse?) {
                                          // User has granted permission
                                          // Proceed and read list of contacts
                                      }


                                      override fun
                      onPermissionRationaleShouldBeShown(permission: PermissionRequest?,
                      token: PermissionToken?) {
                                          // User previously denied the request, proceed
                      and ask them again

                                          token?.continuePermissionRequest()
                                      }


                                      override fun onPermissionDenied(response:
                      PermissionDeniedResponse?) {
                                          // User has rejected request
                                      }
```

```
                    } )
                    .check()
            }


            ...
        }
```

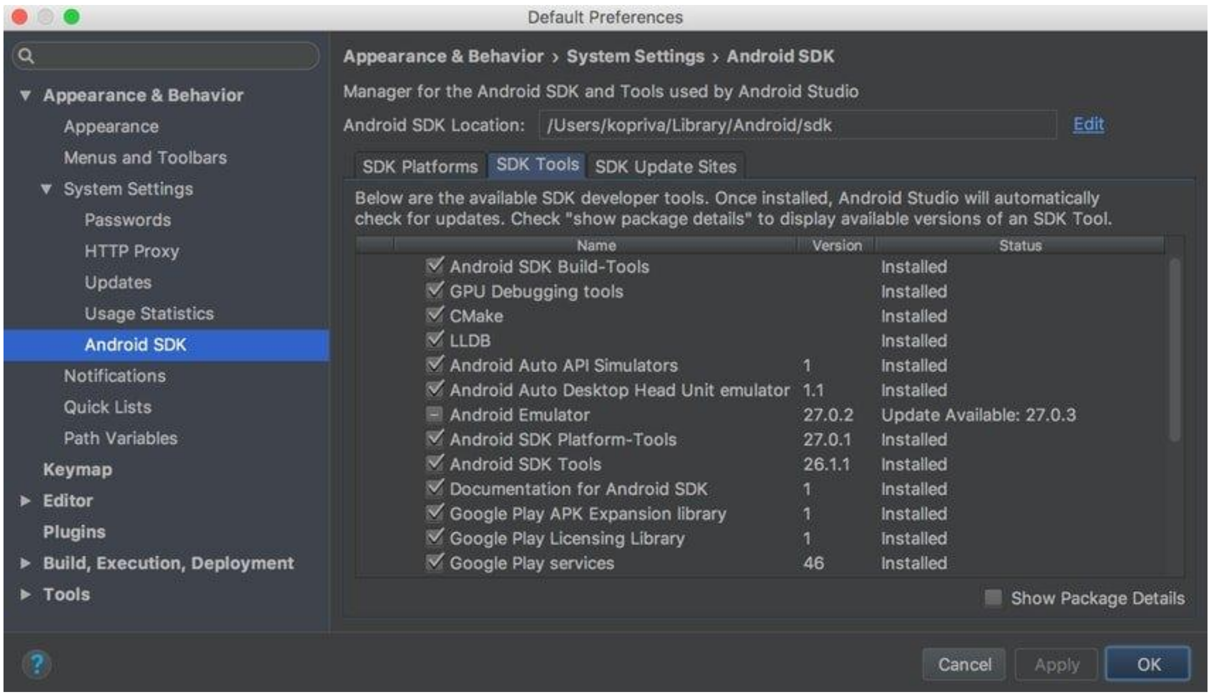


6:00
Type here
Allow Awesome Notes to access your contacts?
DENY
ALLOW
Note taken in London



6:00
Type here
Allow Awesome Notes to access your contacts?
Never ask again
DENY
ALLOW
Note taken in London

What Is the Android SDK?



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/kopriva/Library/Android/sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		Installed
✓ GPU Debugging tools		Installed
✓ CMake		Installed
✓ LLDB		Installed
✓ Android Auto API Simulators	1	Installed
✓ Android Auto Desktop Head Unit emulator	1.1	Installed
▬ Android Emulator	27.0.2	Update Available: 27.0.3
✓ Android SDK Platform-Tools	27.0.1	Installed
✓ Android SDK Tools	26.1.1	Installed
✓ Documentation for Android SDK	1	Installed
✓ Google Play APK Expansion library	1	Installed
✓ Google Play Licensing Library	1	Installed
✓ Google Play services	46	Installed

The Android SDK is a collection of software development tools and libraries required to develop Android applications. Every time Google releases a new version of Android or an update, a corresponding SDK is also released which developers must download and install. It is worth noting that you can also download and use the Android SDK independently of Android Studio, but typically you'll be working through Android Studio for any Android development.

The Android SDK comprises all the tools necessary to code programs from scratch and even test them. These tools provide a smooth flow of the development process from developing and debugging, through to packaging.

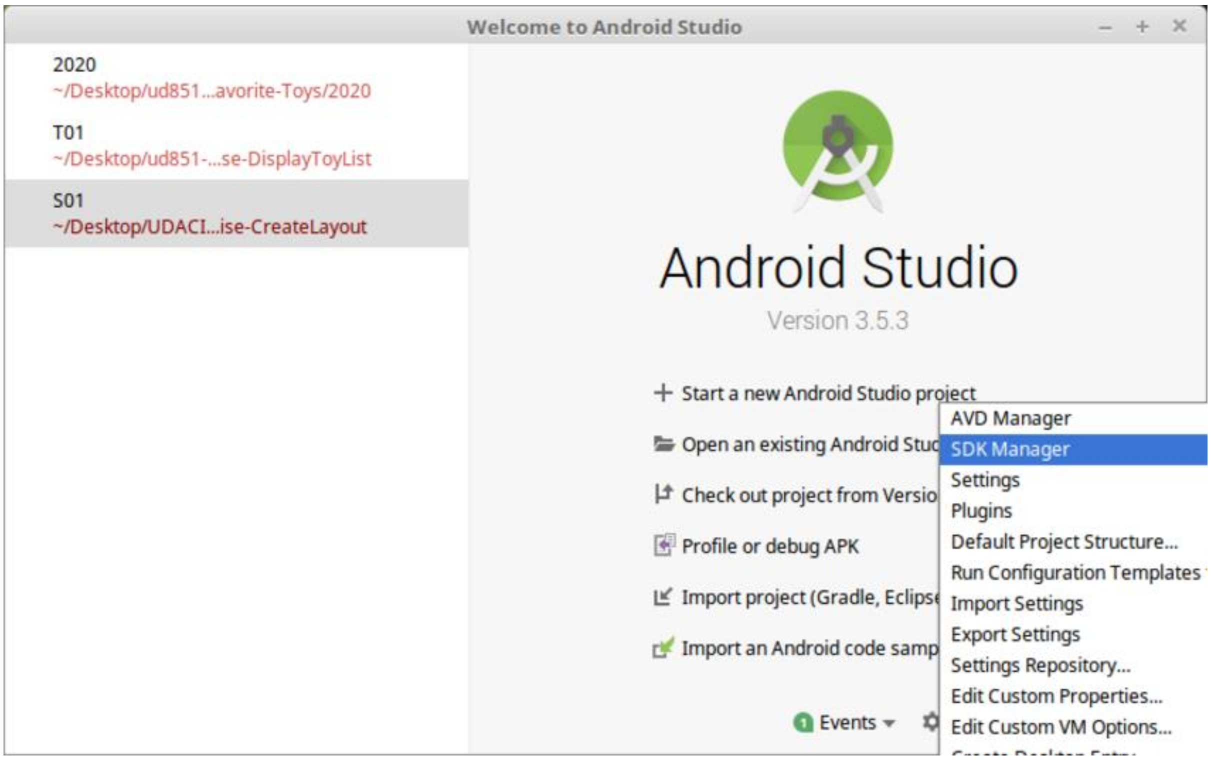
The Android SDK is compatible with Windows, macOS, and Linux, so you can develop on any of those platforms.

How to Install the Android SDK

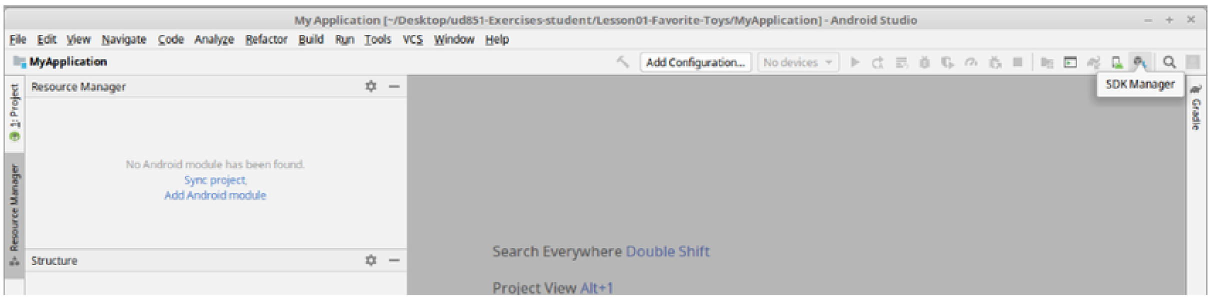
The Android SDK is optimized for Android Studio, and hence to effectively reap its benefits, you will need to install Android Studio. Having the Android SDK managed from within Android Studio is easier since support for languages like Java, Kotlin, and C++ is handled automatically. Not only that, but updates to the Android SDK are handled automatically by Android Studio.

To install the Android SDK from within Android Studio, first start Android Studio.

- From the Android Studio start page, select **Configure > SDK Manager.**

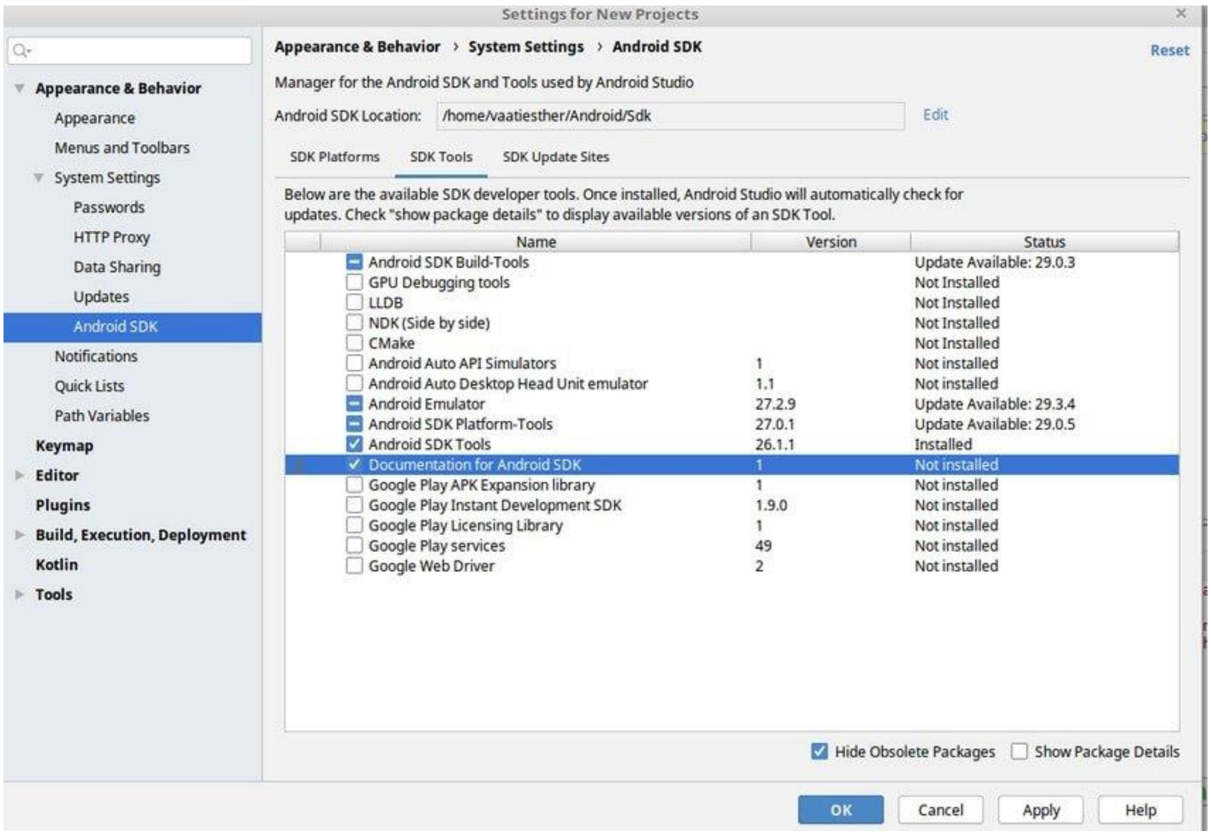

Welcome to Android Studio
2020
~/Desktop/ud851...avorite-Toys/2020
T01
~/Desktop/ud851-...se-DisplayToyList
S01
~/Desktop/UDACI...ise-CreateLayout
Android Studio
Version 3.5.3
Start a new Android Studio project
Open an existing Android Stu
Check out project from Versio
Profile or debug APK
Import project (Gradle, Eclipse
Import an Android code samp
Events
AVD Manager
SDK Manager
Settings
Plugins
Default Project Structure...
Run Configuration Templates
Import Settings
Export Settings
Settings Repository...
Edit Custom Properties...
Edit Custom VM Options...

- If you already have Android Studio open, the SDK Manager icon is found on the top right corner, as shown below.


My Application [~/Desktop/ud851-Exercises-student/Lesson01-Favorite-Toys/MyApplication] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication
Add Configuration...
No devices
SDK Manager
Resource Manager
No Android module has been found.
Sync project,
Add Android module
Structure
Search Everywhere Double Shift
Project View Alt+1
1: Project
Resource Manager
Gradle

Install the required Android SDK platform packages and developer tools. A good start is to install:

- Android SDK Build-Tools
- Android Emulator
- Android SDK Platform-Tools
- Android SDK Tools
- Documentation for Android SDK



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/vaatiesther/Android/Sdk Edit

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

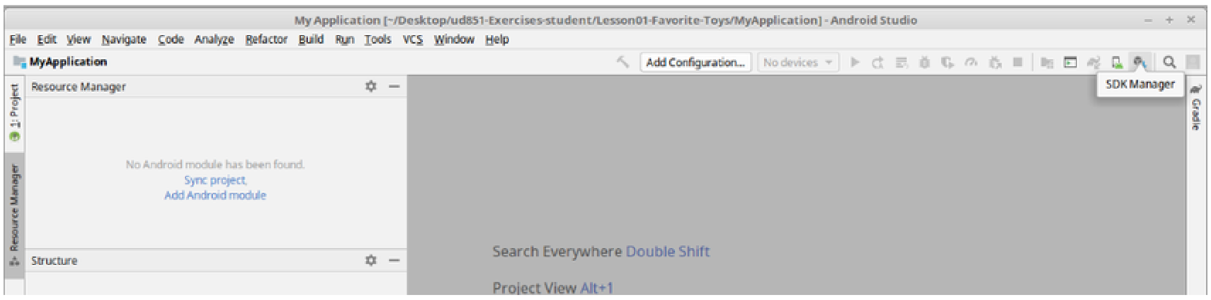
Name	Version	Status
▬ Android SDK Build-Tools		Update Available: 29.0.3
☐ GPU Debugging tools		Not Installed
☐ LLDB		Not Installed
☐ NDK (Side by side)		Not Installed
☐ CMake		Not Installed
☐ Android Auto API Simulators	1	Not installed
☐ Android Auto Desktop Head Unit emulator	1.1	Not installed
▬ Android Emulator	27.2.9	Update Available: 29.3.4
▬ Android SDK Platform-Tools	27.0.1	Update Available: 29.0.5
✓ Android SDK Tools	26.1.1	Installed
✓ Documentation for Android SDK	1	Not installed
☐ Google Play APK Expansion library	1	Not installed
☐ Google Play Instant Development SDK	1.9.0	Not installed
☐ Google Play Licensing Library	1	Not installed
☐ Google Play services	49	Not installed
☐ Google Web Driver	2	Not installed

✓ Hide Obsolete Packages ☐ Show Package Details

Click **Apply**, and Android Studio will install the selected tools and packages.

What Is the **SDK Manager**?

The Android SDK is composed of modular packages that you can download, install, and update separately using the Android SDK Manager. The SDK Manager helps to update new SDK releases and updates whenever a new Android platform is released. The SDK manager can be found in the top-right corner of the Android Studio screen, as shown below.



My Application [~/Desktop/ud851-Exercises-student/Lesson01-Favorite-Toys/MyApplication] - Android Studio

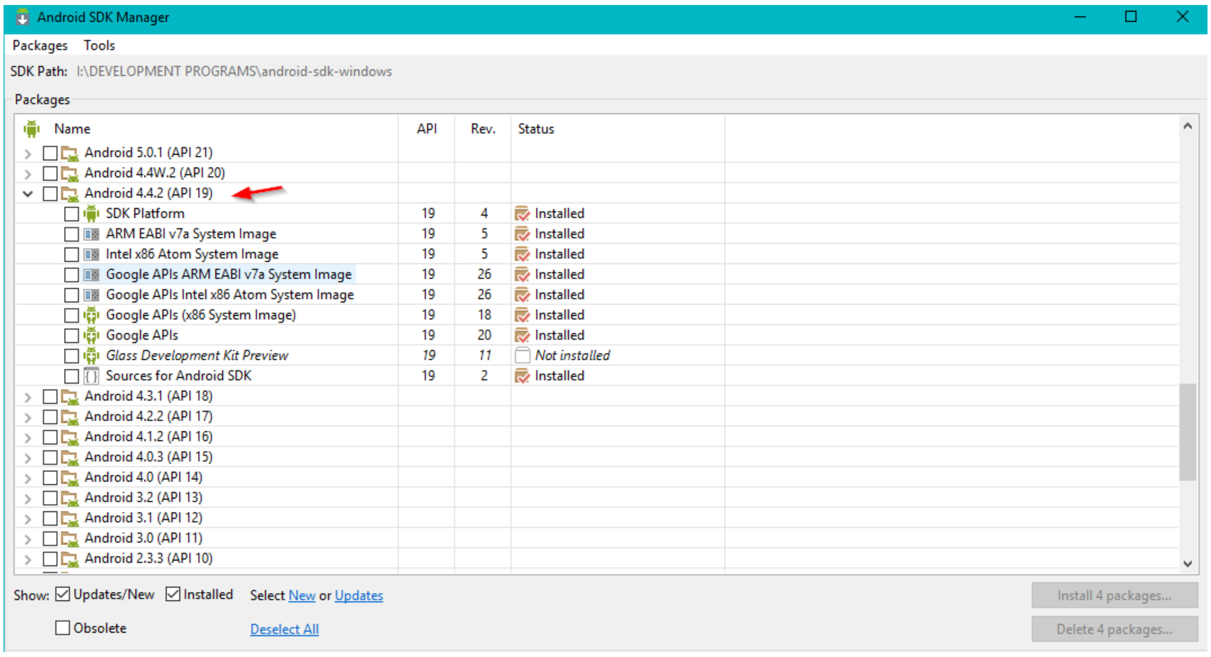
Resource Manager

No Android module has been found.
Sync project
Add Android module

Search Everywhere Double Shift

Project View Alt+1

All that is required to follow the instructions provided, and the updates will be immediately downloaded to your environment.



SDK Path: I:\DEVELOPMENT PROGRAMS\android-sdk-windows

Packages

Name	API	Rev.	Status
Android 5.0.1 (API 21)			
Android 4.4W.2 (API 20)			
Android 4.4.2 (API 19)			
SDK Platform	19	4	Installed
ARM EABI v7a System Image	19	5	Installed
Intel x86 Atom System Image	19	5	Installed
Google APIs ARM EABI v7a System Image	19	26	Installed
Google APIs Intel x86 Atom System Image	19	26	Installed
Google APIs (x86 System Image)	19	18	Installed
Google APIs	19	20	Installed
Glass Development Kit Preview	*19*	*11*	*Not installed*
Sources for Android SDK	19	2	Installed
Android 4.3.1 (API 18)			
Android 4.2.2 (API 17)			
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 4.0 (API 14)			
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			
Android 2.3.3 (API 10)			

Show: [✓] Updates/New [✓] Installed Select New or Updates

[ ] Obsolete Deselect All

Android KTX

Android KTX is a set of Kotlin extensions that are included with Android Jetpack and other Android libraries. KTX extensions provide concise, idiomatic Kotlin to Jetpack, Android platform, and other APIs. To do so, these extensions leverage several Kotlin language features, including the following:

- Extension functions
- Extension properties
- Lambdas
- Named parameters
- Parameter default values
- Coroutines
- As an example, when working with SharedPreferences, you must create an editor before you can make modifications to the preferences data. You must also apply or commit those changes when you are finished editing, as shown in the following example:

-
  ```
  sharedPreferences
          .edit()  // create an Editor
          .putBoolean("key", value)
          .apply() // write to disk asynchronously
  ```
- Kotlin lambdas are a perfect fit for this use case. They allow you to take a more concise approach by passing a block of code to execute after the editor is created, letting the code execute, and then letting the SharedPreferences API apply the changes atomically.

- Here's an example of one of the Android KTX Core functions, SharedPreferences.edit, which adds an edit function to SharedPreferences. This function takes an optional boolean flag as its first argument that indicates whether to commit or apply the changes. It also receives an action to perform on the SharedPreferences editor in the form of a lambda.

-
  ```
  // SharedPreferences.edit extension function signature from Android KTX - Core
  // inline fun SharedPreferences.edit(
  //         commit: Boolean = false,
  //         action: SharedPreferences.Editor.() -> Unit)

  // Commit a new value asynchronously
  sharedPreferences.edit { putBoolean("key", value) }

  // Commit a new value synchronously
  sharedPreferences.edit(commit = true) { putBoolean("key", value) }
  ```

- The caller can choose whether to commit or apply the changes. The action lambda is itself an anonymous extension function on SharedPreferences.Editor which returns Unit, as indicated by its signature. This is why inside the block, you are able to perform the work directly on the SharedPreferences.Editor.

- Finally, the SharedPreferences.edit() signature contains the inline keyword. This keyword tells the Kotlin compiler that it should copy and paste (or *inline*) the compiled bytecode for the function each time the function is used. This avoids the overhead of instantiating a new class for every action each time this function is called.

- This pattern of passing code using lambdas, applying sensible defaults that can be overridden, and adding these behaviors to existing APIs using inline extension functions is typical of the enhancements provided by the Android KTX library.

AndroidX Modules

Android KTX is organized into modules, where each module contains one or more packages.

You must include a dependency for each module artifact in your app's build.gradle file. Remember to append the version number to the artifact. You can find the latest version numbers in each artifact's corresponding section in this topic.

Android KTX contains a single core module that provides Kotlin extensions for common framework APIs and several domain-specific extensions.

With the exception of the core module, all KTX module artifacts replace the underlying Java dependency in your build.gradle file. For example, you can replace androidx.fragment:fragment dependency,with androidx.fragment:fragm ent-ktx. This syntax helps to better manage versioning and does not add additional dependency declaration requirements.

Core KTX

The Core KTX module provides extensions for common libraries that are part of the Android framework. These libraries do not have Java-based dependencies that you need to add to build.gradle.

Collection KTX

The Collection extensions contain utility functions for working with Android's memory-efficient collection libraries,
including ArrayMap, LongSparseArray, LruCache, and others.

To use this module, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
   implementation "androidx.collection:collection-ktx:1.2.0"
}
```

Collection extensions take advantage of Kotlin’s operator overloading to simplify things like collection concatenation, as shown in the following example:

```
// Combine 2 ArraySets into 1.
val combinedArraySet = arraySetOf(1, 2, 3) + arraySetOf(4, 5, 6)

// Combine with numbers to create a new sets.
val newArraySet = combinedArraySet + 7 + 8
```

Fragment KTX

The Fragment KTX module provides a number of extensions to simplify the fragment API.

To include this module, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
   implementation "androidx.fragment:fragment-ktx:1.4.0"
}
```

With the Fragment KTX module, you can simplify fragment transactions with lambdas, for example:

```
fragmentManager().commit {
  addToBackStack("...")
  setCustomAnimations(
        R.anim.enter_anim,
        R.anim.exit_anim)
  add(fragment, "...")
}
```

You can also bind to a ViewModel in one line by using the viewModels and activityViewModels property delegates:

```
// Get a reference to the ViewModel scoped to this Fragment
val viewModel by viewModels<MyViewModel>()

// Get a reference to the ViewModel scoped to its Activity
val viewModel by activityViewModels<MyViewModel>()
```

Lifecycle KTX

Lifecycle KTX defines a LifecycleScope for each Lifecycle object. Any coroutine launched in this scope is canceled when the Lifecycle is destroyed. You can access the CoroutineScope of the Lifecycle by using the lifecycle.coroutineScope or lifecycleOwner.lifecycleScope properties.

To include this module, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.4.0"
}
```

The following example demonstrates how to use lifecycleOwner.lifecycleScope to create precomputed text asynchronously:

```
class MyFragment: Fragment() {
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        viewLifecycleOwner.lifecycleScope.launch {
            val params = TextViewCompat.getTextMetricsParams(textView)
            val precomputedText = withContext(Dispatchers.Default) {
                PrecomputedTextCompat.create(longTextContent, params)
            }
            TextViewCompat.setPrecomputedText(textView, precomputedText)
        }
    }
}
```

LiveData KTX

When using LiveData, you might need to calculate values asynchronously. For example, you might want to retrieve a user's preferences and serve them to your UI. For these cases, LiveData KTX provides a liveData builder function that calls a suspend function and serves the result as a LiveData object.

To include this module, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.4.0"
}
```

In the following example, loadUser() is a suspend function declared elsewhere. You can use the liveData builder function to call loadUser() asynchronously, and then use emit() to emit the result:

```
val user: LiveData<User> = liveData {
    val data = database.loadUser() // loadUser is a suspend function.
    emit(data)
}
```

For more information on using coroutines with LiveData, see Use Kotlin coroutines with Architecture components.

Navigation KTX

Each component of the Navigation library has its own KTX version that adapts the API to be more succinct and Kotlin-idiomatic.

To include these modules, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
    implementation "androidx.navigation:navigation-runtime-ktx:2.3.5"
    implementation "androidx.navigation:navigation-fragment-ktx:2.3.5"
    implementation "androidx.navigation:navigation-ui-ktx:2.3.5"
}
```

Use the extension functions and property delegation to access destination arguments and navigate to destinations, as shown in the following example:

```
class MyDestination : Fragment() {

   // Type-safe arguments are accessed from the bundle.
   val args by navArgs<MyDestinationArgs>()

   ...
   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
      view.findViewById<Button>(R.id.next)
         .setOnClickListener {
            // Fragment extension added to retrieve a NavController from
            // any destination.
            findNavController().navigate(R.id.action_to_next_destination)
         }
    }
    ...

}
```

Palette KTX

The Palette KTX module offers idiomatic Kotlin support for working with color palettes.

To use this module, add the following to your app's build.gradle file:

GroovyKotlin

```
dependencies {
   implementation "androidx.palette:palette-ktx:1.0.0"
}
```

As an example, when working with a Palette instance, you can retrieve the selected swatch for a given target by using the get operator ([ ]):

```
val palette = Palette.from(bitmap).generate()
val swatch = palette[target]
```

Other KTX modules

You can also include additional KTX modules that exist outside of AndroidX.

Firebase KTX

Some of the Firebase SDKs for Android have Kotlin extension libraries that enable you to write idiomatic Kotlin code when using Firebase in your app. For more information, see the following topics:

- Firebase Android SDK
- Firebase Common Kotlin Extensions

Google Maps Platform KTX

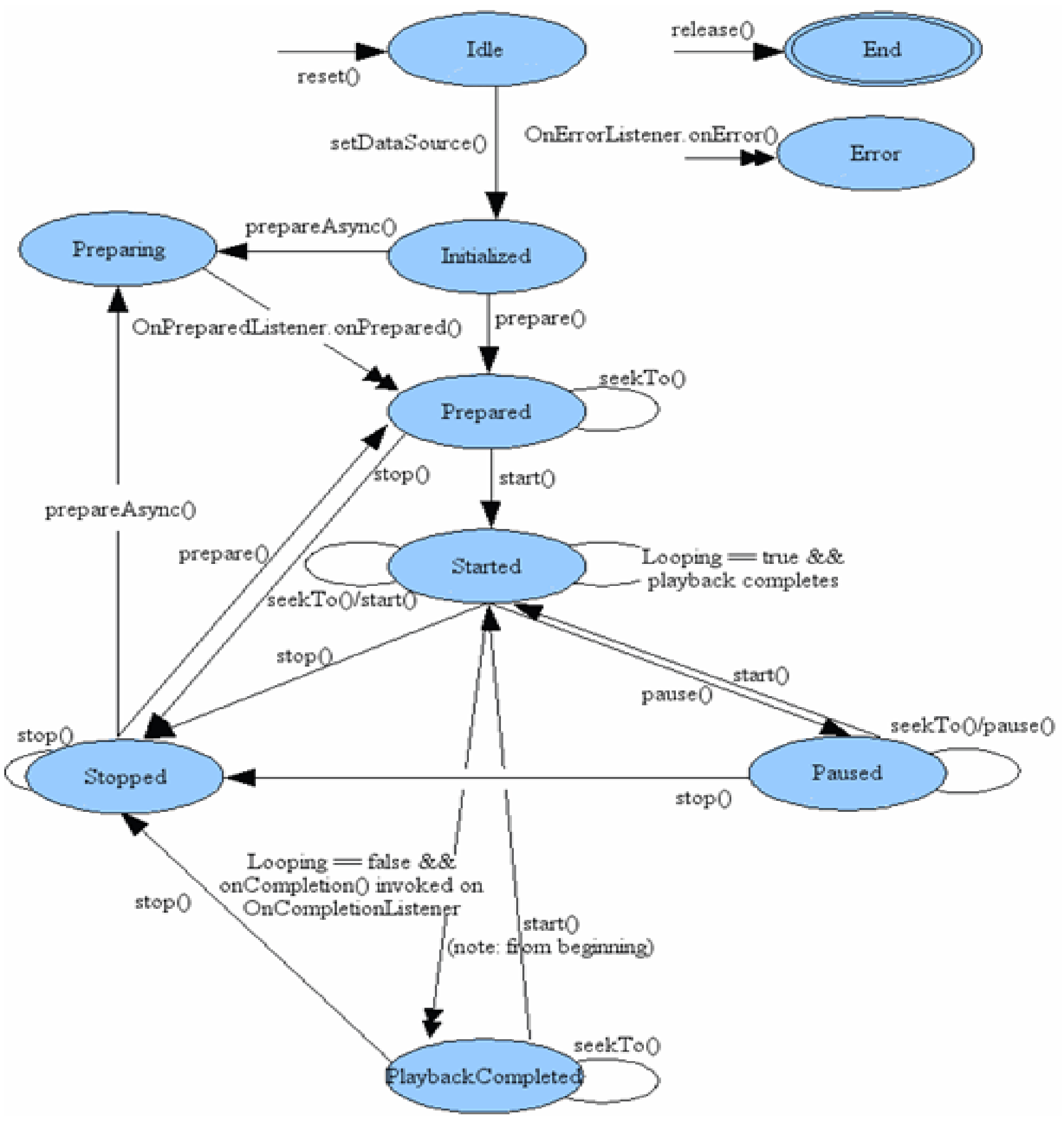
There are KTX extensions available for Google Maps Platform Android SDKs which allow you to take advantage of several Kotlin language features such as extension functions, named parameters and default arguments, destructuring declarations, and coroutines. For more information, see the following topics:

- Maps Android KTX
- Places Android KTX

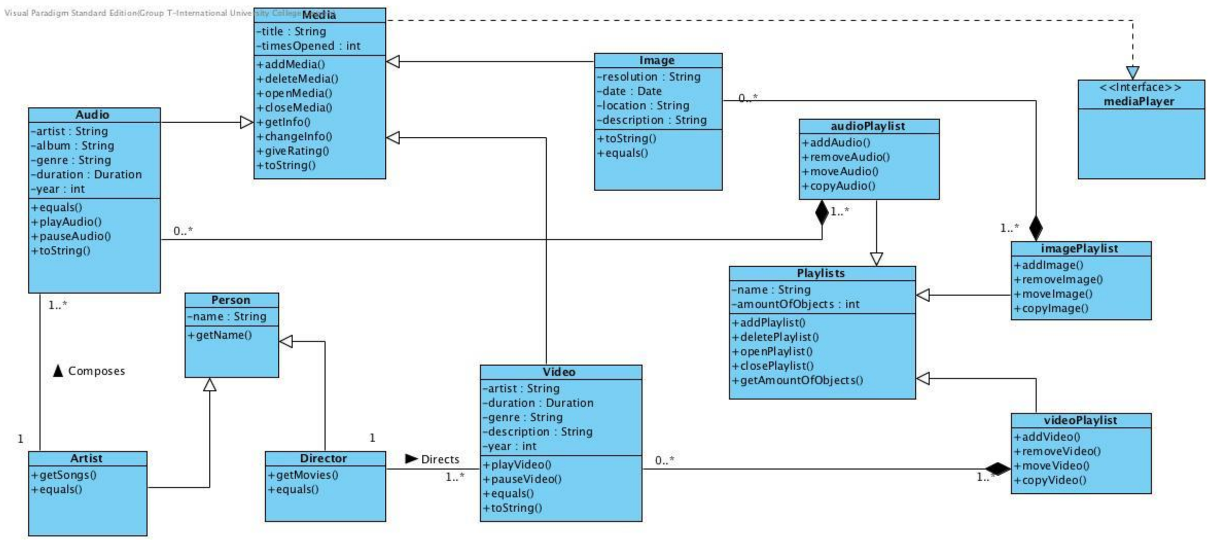
Play Core KTX

Play Core KTX adds support for Kotlin coroutines for one-shot requests and Flow for monitoring status updates by adding extension functions to SplitInstallManager and AppUpdateManager in the Play Core library.

2.3 WORKING & UML DIAGRAM OF **Media Player :**



2.3.1 Object diagram



3.0 System Requirement

3.1 Windows

- 64-bit Microsoft® Windows® 8/10.
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor.
- 8 GB RAM or more.
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution.

3.2 Mac

- MacOS® 10.14 (Mojave) or higher

- ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

Advantage of proposed system

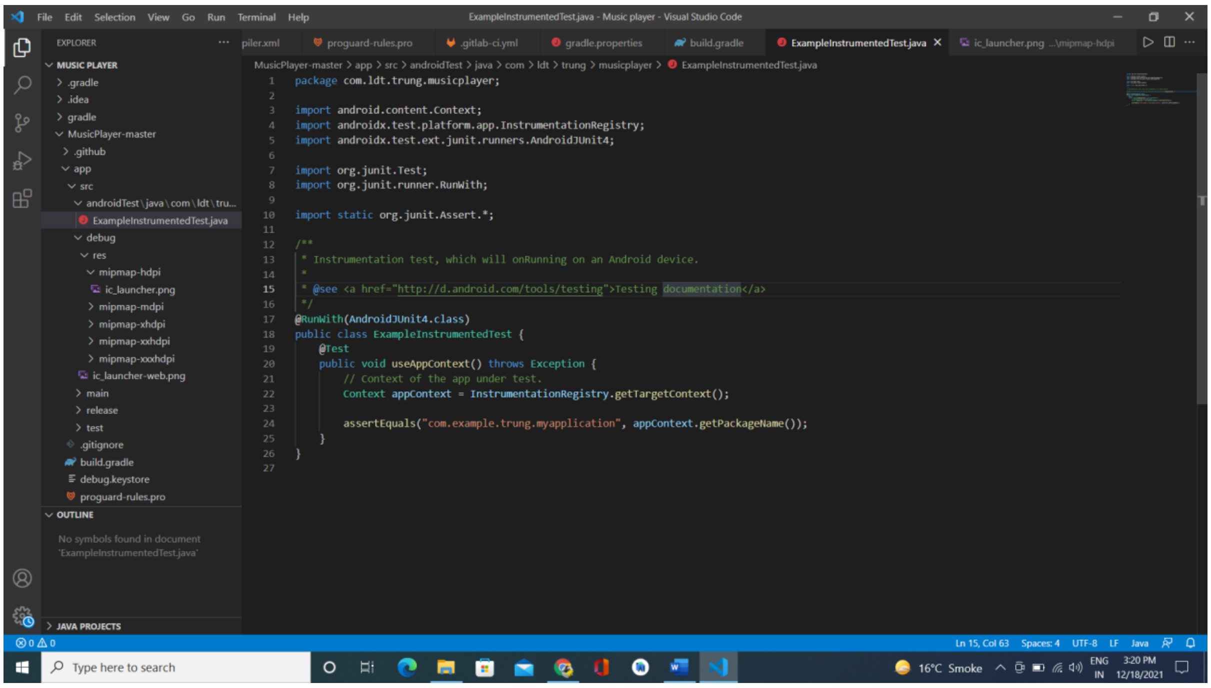
The return on investment for good requirements gathering is virtually always higher than the cost. In almost all cases if you spend the time to develop the requirements you can develop a far superior product with much less hassle and frustration. At the same time not doing the right amount of requirements gathering can create a chaotic environment in which stakeholders are upset with the managers, managers are upset with the developers, developers are upset with managers and in the end, what suffers is the product. Some of the benefits from good requirements gathering can be:

- Fewer defects in the delivered product
- Less development rework
- Faster delivery of the finished product
- Less unused features
- Lower cost of development
- Less miss-communicated requirements
- Reduced project chaos
- Higher levels of satisfaction from stakeholders
- Higher levels of satisfaction from developers
- Higher levels of satisfaction from consumers and users
- Products that work well and have a useful feature set

4.0 AndroidManifest.xml code

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.ldt.musicr">
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission
android:name="android.Manifest.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.BROADCAST_STICKY" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"
/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <application
        android:name=".App"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher"
        android:supportsRtl="true"
        android:theme="@style/AppThemeNoWallpaper"
        tools:ignore="UnusedAttribute">
        <activity
            android:name=".ui.AppActivity"
            android:theme="@style/SplashThemeV2"
            android:screenOrientation="portrait"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:launchMode="singleInstance"
            android:windowSoftInputMode="adjustPan|adjustResize"
            tools:ignore="LockedOrientationActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <action android:name="android.intent.action.MUSIC_PLAYER" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.APP_MUSIC" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />

            <data android:scheme="content" />
            <data android:mimeType="audio/*" />
            <data android:mimeType="application/ogg" />
            <data android:mimeType="application/x-ogg" />
            <data android:mimeType="application/itunes" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />

            <category android:name="android.intent.category.DEFAULT" />

            <data android:scheme="file" />
            <data android:mimeType="audio/*" />
            <data android:mimeType="application/ogg" />
            <data android:mimeType="application/x-ogg" />
            <data android:mimeType="application/itunes" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />

            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />

            <data android:scheme="http" />
            <data android:mimeType="audio/*" />
            <data android:mimeType="application/ogg" />
            <data android:mimeType="application/x-ogg" />
            <data android:mimeType="application/itunes" />
        </intent-filter>
            <intent-filter>
                <action android:name="com.cyanogenmod.eleven.AUDIO_PLAYER" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.PICK" />

                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.OPENABLE" />

                <data android:mimeType="vnd.android.cursor.dir/audio" />
            </intent-filter>

            <intent-filter>
                <action android:name="android.intent.action.VIEW" />

                <category android:name="android.intent.category.DEFAULT" />

                <data android:mimeType="vnd.android.cursor.dir/playlist" />
                <data android:mimeType="vnd.android.cursor.dir/albums" />
                <data android:mimeType="vnd.android.cursor.dir/artists" />
            </intent-filter>
        </activity>
        <service
            android:name=".service.MusicService"
            android:enabled="true"/>
        <meta-data
            android:name="android.max_aspect"
            android:value="2.1" />
        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="${applicationId}"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths" />
        </provider>
        <receiver android:name=".service.MediaButtonIntentReceiver">
            <intent-filter>
                <action android:name="android.intent.action.MEDIA_BUTTON" />
            </intent-filter>
        </receiver>
        <receiver android:name=".appwidgets.BootReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.QUICKBOOT_POWERON"
/>
            </intent-filter>
        </receiver>
        <receiver
            android:name=".appwidgets.AppWidgetBig"
            android:exported="false"
            android:label="@string/app_widget_big_name">
            <intent-filter>
                <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/app_widget_big_info" />
        </receiver>
        <receiver
            android:name=".appwidgets.AppWidgetClassic"
            android:exported="false"
            android:label="@string/app_widget_classic_name">
            <intent-filter>
                <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/app_widget_classic_info" />
        </receiver>
        <receiver
            android:name=".appwidgets.AppWidgetSmall"
            android:exported="false"
            android:label="@string/app_widget_small_name">
            <intent-filter>
                <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/app_widget_small_info" />
        </receiver>
        <receiver
            android:name=".appwidgets.AppWidgetCard"
            android:exported="false"
            android:label="@string/app_widget_card_name">
            <intent-filter>
                <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>

            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/app_widget_card_info" />
        </receiver>
```

```
    </application>
</manifest>
```



MusicPlayer-master > app > src > androidTest > java > com > ldt > trung > musicplayer > ExampleInstrumentedTest.java

```java
package com.ldt.trung.musicplayer;

import android.content.Context;
import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumentation test, which will onRunning on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() throws Exception {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getTargetContext();

        assertEquals("com.example.trung.myapplication", appContext.getPackageName());
    }
}
```

```java
package com.ldt.trung.musicplayer;


import android.content.Context;


import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.ext.junit.runners.AndroidJUnit4;


import org.junit.Test;
import org.junit.runner.RunWith;


import static org.junit.Assert.*;



/**
```

```
 * Instrumentation test, which will onRunning on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing
documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test

    public void useAppContext() throws Exception {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getTargetContext();

        assertEquals("com.example.trung.myapplication",
appContext.getPackageName());
    }
}
```

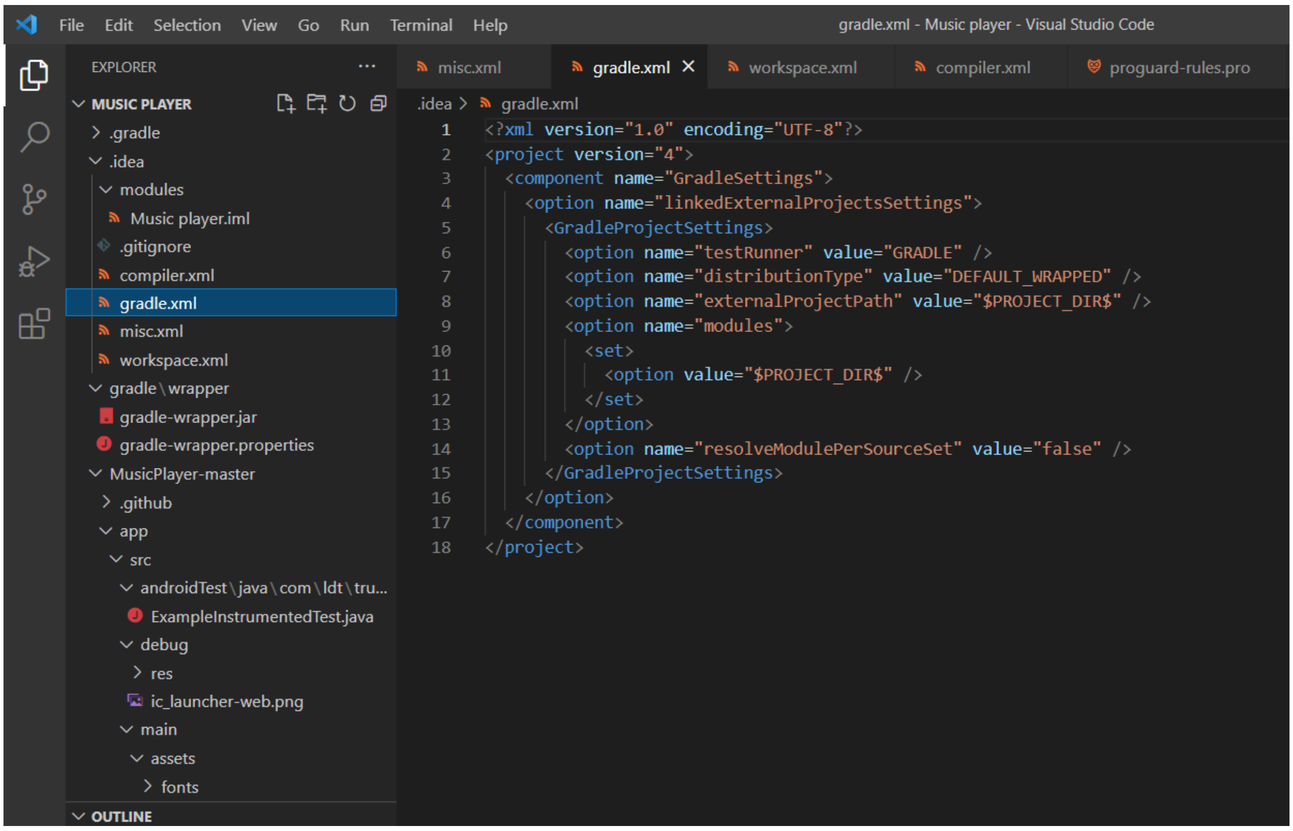
Gradle.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">

  <component name="GradleMigrationSettings" migrationVersion="1" />
  <component name="GradleSettings">
    <option name="linkedExternalProjectsSettings">
      <GradleProjectSettings>

        <option name="testRunner" value="GRADLE" />
        <option name="distributionType" value="DEFAULT_WRAPPED" />
        <option name="externalProjectPath" value="$PROJECT_DIR$" />
        <option name="modules">
          <set>
            <option value="$PROJECT_DIR$" />
          </set>
        </option>

        <option name="resolveModulePerSourceSet" value="false" />
      </GradleProjectSettings>
    </option>
  </component>

</project>
```

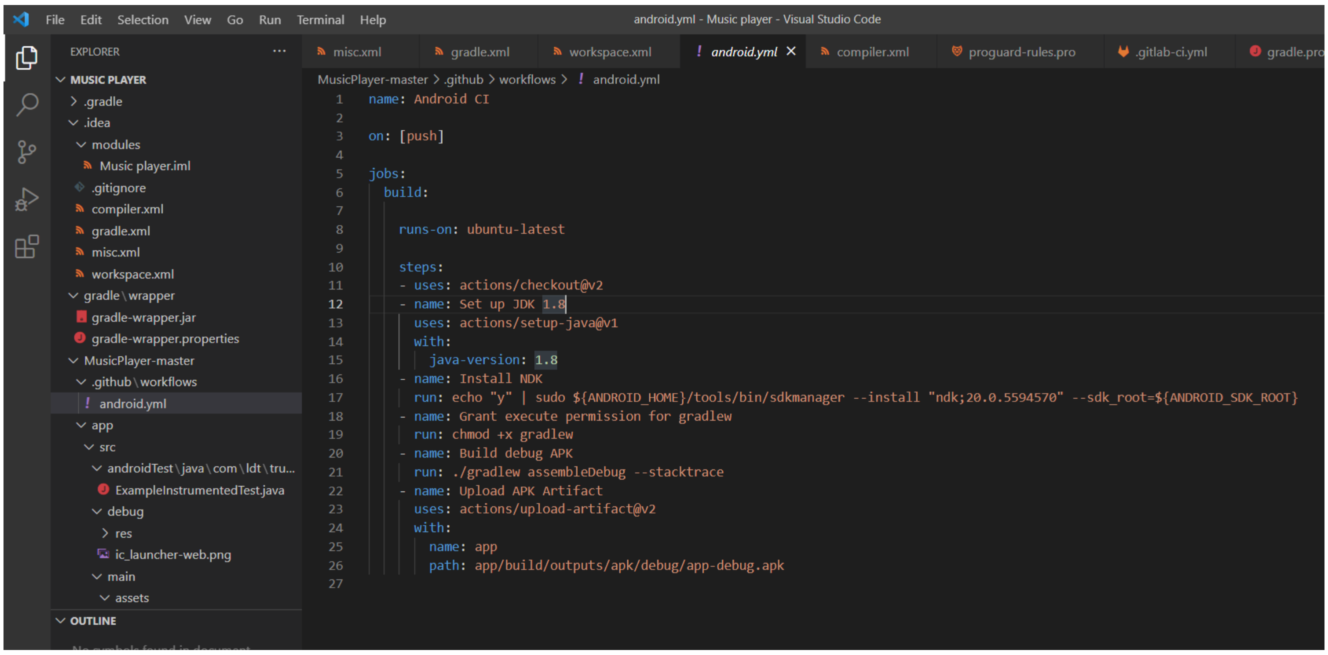
WorkSpace.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="AutoImportSettings">
    <option name="autoReloadType" value="NONE" />

  </component>
  <component name="ChangeListManager">
    <list default="true" id="cbf6d689-a3a4-43c4-ac6d-cba155fff8c6"
name="Default Changelist" comment="" />
    <option name="SHOW_DIALOG" value="false" />

    <option name="HIGHLIGHT_CONFLICTS" value="true" />
    <option name="HIGHLIGHT_NON_ACTIVE_CHANGELIST" value="false" />
    <option name="LAST_RESOLUTION" value="IGNORE" />
  </component>

  <component name="ExternalProjectsData">
    <projectState path="$PROJECT_DIR$">
      <ProjectState />
    </projectState>
  </component>

  <component name="ProjectId" id="21ryHF2giGnV7YRabQkcndV6g5T" />
  <component name="ProjectViewState">

    <option name="hideEmptyMiddlePackages" value="true" />
    <option name="showLibraryContents" value="true" />
  </component>
  <component name="PropertiesComponent">
```

```
    <property name="RunOnceActivity.OpenProjectViewOnStart" value="true" />
    <property name="RunOnceActivity.ShowReadmeOnStart" value="true" />
  </component>

  <component name="SpellCheckerSettings" RuntimeDictionaries="0"
Folders="0" CustomDictionaries="0" DefaultDictionary="application-level"
UseSingleDictionary="true" transferred="true" />
  <component name="TaskManager">

    <task active="true" id="Default" summary="Default task">
      <changelist id="cbf6d689-a3a4-43c4-ac6d-cba155fff8c6" name="Default
Changelist" comment="" />
      <created>1638713420323</created>
      <option name="number" value="Default" />
      <option name="presentableId" value="Default" />

      <updated>1638713420323</updated>
    </task>
    <servers />

  </component>
</project>
```

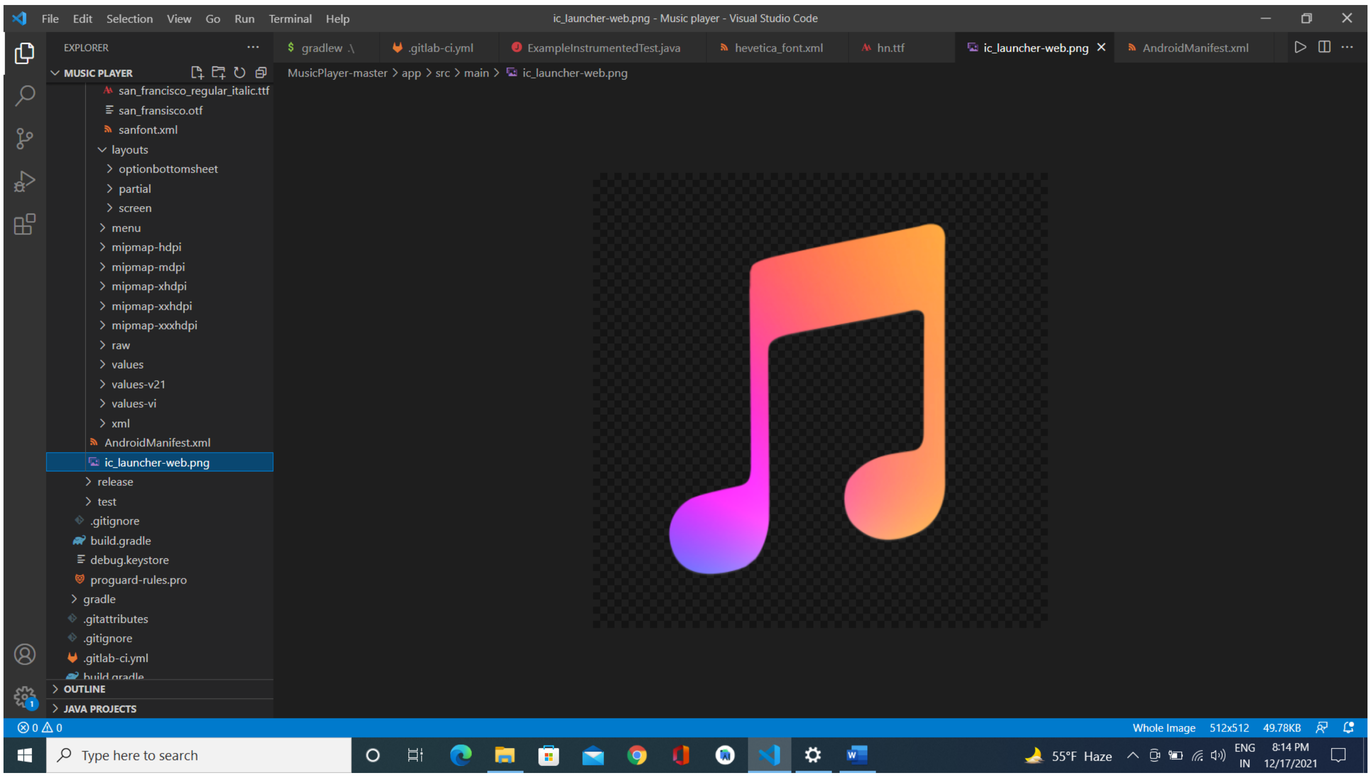


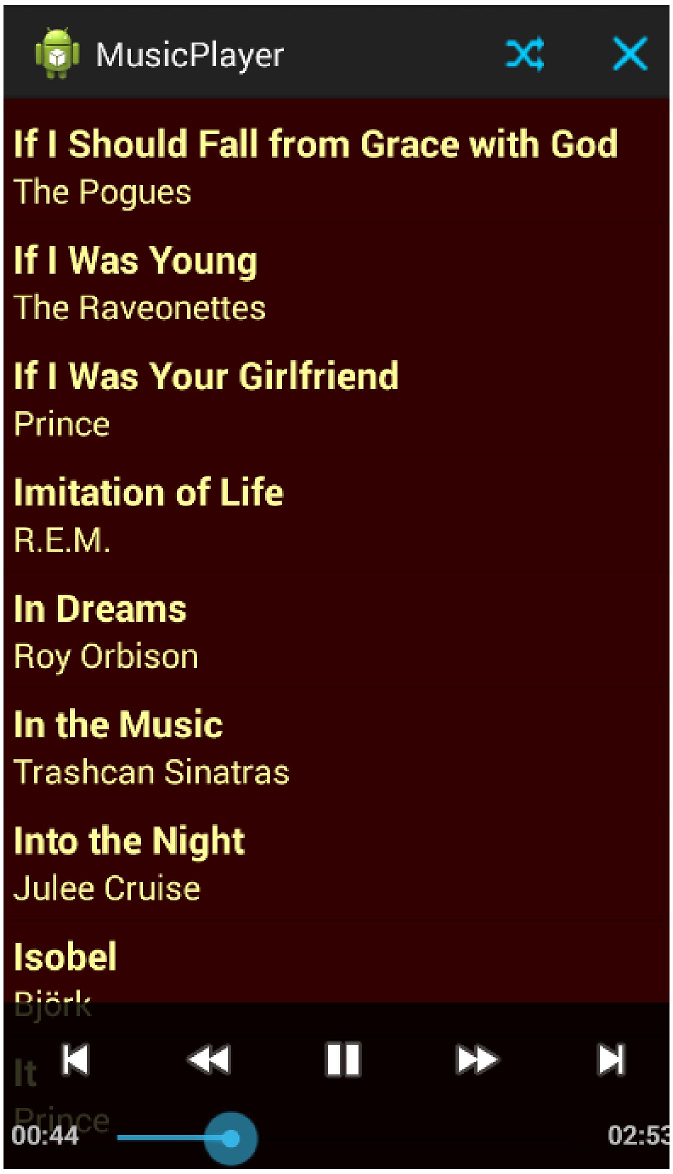
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="GradleSettings">
    <option name="linkedExternalProjectsSettings">
      <GradleProjectSettings>
        <option name="testRunner" value="GRADLE" />
        <option name="distributionType" value="DEFAULT_WRAPPED" />
        <option name="externalProjectPath" value="$PROJECT_DIR$" />
        <option name="modules">
          <set>
            <option value="$PROJECT_DIR$" />
          </set>
        </option>
        <option name="resolveModulePerSourceSet" value="false" />
      </GradleProjectSettings>
    </option>
  </component>
</project>
```



```yaml
name: Android CI

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Install NDK
      run: echo "y" | sudo ${ANDROID_HOME}/tools/bin/sdkmanager --install "ndk;20.0.5594570" --sdk_root=${ANDROID_SDK_ROOT}
    - name: Grant execute permission for gradlew
      run: chmod +x gradlew
    - name: Build debug APK
      run: ./gradlew assembleDebug --stacktrace
    - name: Upload APK Artifact
      uses: actions/upload-artifact@v2
      with:
        name: app
        path: app/build/outputs/apk/debug/app-debug.apk
```

Icon


ic_launcher-web.png - Music player - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MUSIC PLAYER
san_francisco_regular_italic.ttf
san_fransisco.otf
sanfont.xml
layouts
optionbottomsheet
partial
screen
menu
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
raw
values
values-v21
values-vi
xml
AndroidManifest.xml
ic_launcher-web.png
release
test
.gitignore
build.gradle
debug.keystore
proguard-rules.pro
gradle
.gitattributes
.gitignore
.gitlab-ci.yml
build.gradle
OUTLINE
JAVA PROJECTS
gradlew .\
.gitlab-ci.yml
ExampleInstrumentedTest.java
hevetica_font.xml
hn.ttf
ic_launcher-web.png
AndroidManifest.xml
MusicPlayer-master > app > src > main > ic_launcher-web.png
Whole Image 512x512 49.78KB
Type here to search
55°F Haze
ENG IN
8:14 PM 12/17/2021



MusicPlayer

If I Should Fall from Grace with God
The Pogues

If I Was Young
The Raveonettes

If I Was Your Girlfriend
Prince

Imitation of Life
R.E.M.

In Dreams
Roy Orbison

In the Music
Trashcan Sinatras

Into the Night
Julee Cruise

Isobel
Björk

It
Prince

00:44 02:53

5.0 Conclusion

Through the development of the music player on the Android platform, we get a clear understanding of the overall process of the system. The core part of the music player is mainly composed of the main interface, playlists, menus, play Settings, file browsing, and song search. Grasping the development of the six parts, the music player has had the preliminary scale. Based on the function of the six categories, add some other small features. Music player system realized the basic function of player: play, pause, and stop, up/down a volume adjustment, lyrics display, play mode, song search, file browser, playlists query, and other functions. This development implicated the popular mobile terminal development technology.

6.0. Reference

- https://developer.android.com/guide
- developer.android.com/guide/topics/media/mediaplayer
- developer.android.com/guide/topics/media/mediaplayer#releaseplayer
- https://developer.android.com/google/play/developer-api
- https://www.tutorialspoint.com/android/android_mediaplayer.htm