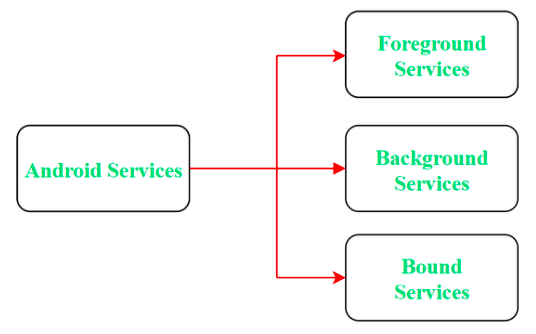
**1. What is a Service?**

* Service is running on MAIN Thread like Activity and Broadcast receiver.
* It's A Application Component which performs long-running operations.
* A service can run continuously in the background, even if the application is closed or the user switches to another application.

*A service does****not****create its own thread and does****not****run in a separate process unless you specify. You should run any blocking operations on a separate thread within the service to avoid Application Not Responding (ANR) errors.*



**2. What is a Broadcast Receiver?**

* Broadcast Receivers are used to respond to **system-wide events**
* Broadcast Receivers allow us to register for the system and application events, when the event happens, then the registered receivers get notified.
* Do not start activities from broadcast receivers because the user experience is jarring; especially if there is more than one receiver. Display a Notification if required.
* For security purposes, **try to use a local broadcast Receiver** instead of a Global.
* We can control BR by using some tags in the manifest. For example-exported = false (the receiver does not receive broadcasts from sources outside the app)

*Types of receivers.*

* **Static Broadcast Receivers:**These types of Receivers are declared in the manifest file and work even if the app is closed.
* **Dynamic Broadcast Receivers:**These types of receivers work only if the app is active or minimized.

**3. What is the memory leak in Android?**

Please refer to this [link](https://proandroiddev.com/everything-you-need-to-know-about-memory-leaks-in-android-d7a59faaf46a), very well explained.

**4. What are the possible states in which a process is based?**

* **State 1:** Foreground activity
* **State 2:** Visible activity
* **State 3:** Background activity
* **State 4:** Empty activity

**5. What are the states in an activity?**

There are four states in an activity. These include the following.

1. **Active state:** The activity is in the foreground
2. **Paused state:** The activity is in the background and visible
3. **Stopped state:** The activity is in the background but not visible, even hidden or obscuring other activities
4. **Destroyed state:** The activity is completely terminated or killed/removed

**6. Explain the different launch modes in Android (most asked question)**

1. **Standard:**

* *Always create an activity’s new instance in the task. Possible to create multiple instances of the same Activity*

*for Example — If the Activity stack is****X⇾ Y⇾ Z****and if I want to launch Y Again then the new stack is****X⇾ Y⇾ Z⇾ Y***

2. **Single Top:**

* *Single Top is similar to Standard, but there is only one difference, if the same Activity exists on the Top of the stack then a new instance of the Same Activity will not be created. But the intent will be sent to the activity’s existing instance.*
* *I tried to explain these two examples-*

*if the Activity stack is****X⇾ Y⇾ Z and I want to launch Y again because the recent stack is not Y it's Z, so the new Flow will be X⇾ Y⇾ Z⇾ Y***

*if the Activity stack is****X⇾ Y⇾ Z****and I want to****launch Z again because the recent stack is Z, so the new Flow will be X⇾ Y⇾ Z (instead of creating a new flow, it will send the Intent to onNewIntent()).***

3. **Single task:**

* *This launch mode will create a new task and push a new instance to the task as the root.*
* *For example, suppose our current stack is W****⇾ X⇾ Y⇾ Z***
* *Now, if we launch activity****X****again with the “single task” launch mode, then the new stack will be****W⇾ X****. Here, a callback has been received on the old instance and****Y****and****Z****activities are destroyed.*

4. **Single Instance:**

* *This launch mode is similar to the****SingleTask****launch mode. But the system doesn’t support launching any new activities in the****same task****.*
* Suppose you have A, B, and C activities(**A →B →C)** and your activity D has a **singleInstance** launch mode. In this case, if we launch D then D will be launched in a different task. A new task for D will be created. *In a situation where the new activity is launched, it is launched in a separate task.*
* *For example, Suppose our current stack is W****⇾ X⇾ Y****. Now, if we launch the activity****Z****with the “singleInstance” launch mode, then there will be two tasks:*
* *Task 1 — W****⇾ X⇾ Y***
* *Task 2 —**Z(*(here Z will be in the different task)*) Now* *if you continue this and start P and Z then Stack will look like*
* *Task 1 — W****⇾ X⇾ Y⇾******P***
* *Task 2 — Z*
* *Again, if you Call activity****Z,****then it will call the same activity from the 2nd stack and pass the intent to****onNewIntent()****.*

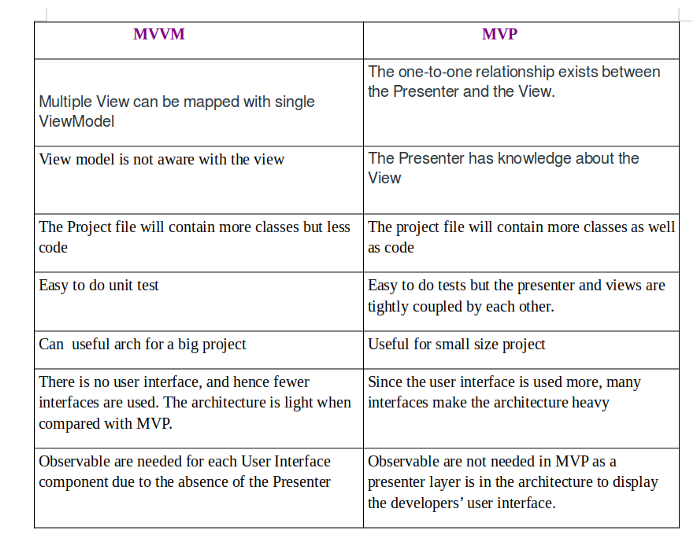
**6. If Apps are crashing frequently, what approach you will take to troubleshoot the problem?**

* We can check the app if it is not compatible with the given **OS version**.
* Try to check the logs if the problem is related to the**proguard** (signed build).
* will check if any **library or component is deprecated**, and it causes a crash of applications.
* Will use[***strict mode***](https://p-mishra2905.medium.com/strictmode-in-android-cd1b2799c689)to check ANR or any other crash.
* App Data Usage: You can delete the application’s data, which will clear its cache memory and allow some free space on your device and might boost the app’s performance.
* Memory Management: Some applications run perfectly on one mobile device, but may crash on other devices. This is where processing power, memory management, and CPU speed come into play.  
  ***Check the application memory requirements if the application crashes constantly****.*
* Always implement **crashlytics like**[***Countly***](https://support.count.ly/hc/en-us/articles/360037754031-Android)**or Firebase crashlytics**to see the crash in classes.

**8. How many ways we can define Android Permission?**

1. **Normal Permissions** — The system will automatically provide authorization to your app during the installation process.
2. **Signature Authorization**— The app requesting permission must be signed with the same signature as the app defining the needed permission
3. **Dangerous Permission** — The app requesting permission is directly related to users' privacy. If the user refuses the permission, your application will be unable to utilize that permission.

**9. Difference between MVP and MVVM architecture?**



**10. Details about RX Java, Live Data and Coroutines-**

Why Rx Java, live data or coroutine?

*Consider the scenario: we are doing some background work which needs to run as a long-running task if we do this on the main thread then maybe after sometimes our app will through ANR or crash, so we will do it with the async task (already deprecated) the other approach we have right now is RX Java, Coroutines, flows etc.*

**RX Java :**

RxJava is a JVM library that uses observable sequences to perform asynchronous and event-based programming

*RxJava, on the other hand, is designed primarily for Java projects. RxAndroid is required to use RxJava in Android.*

It simplifies multithreading in our project. It assists us in determining which thread we want to run the task on

**Live Data**

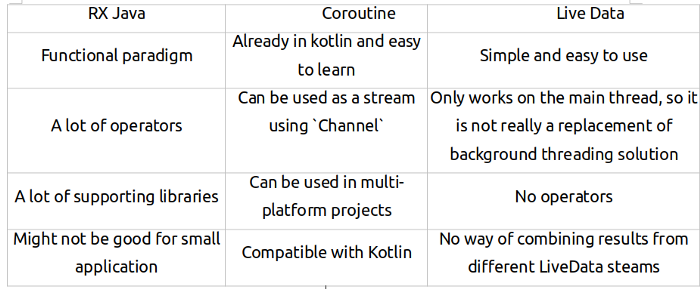
Live Data is a reactive solution for delivering events to UI in Android. It is aware of the life cycle of the owner and because of that, it is simple to use. You don’t need to save any subscription-like object to unsubscribe when needed. However, its simplicity carries some drawbacks — it only works on the Main Thread. Another drawback is that there are no operators or ways of combining results from different Live Data streams.

**Coroutines**

A coroutine is compatible with Kotlin language, which works for asynchronous code. It was developed by JetBrains.

A coroutine is a lightweight thread. Kotlin compiler automatically generates state machines from the code, so the code will behave as normal synchronous code.

**Overall view of all three-**



**10. How to reduce APK size?**

Please follow this [link](https://p-mishra2905.medium.com/reduce-apk-size-ce6488b87acb)

**11. Dependency Injection in Android**

Why Dependency Injection- if we have a larger code base, and we are going to create an object which depends on many other objects, then what will be the complex scenario? If we don't use it in any class, we need to write a bunch of extra code. This time we might need something external to handle this

in Android, we have Dagger2, Hilt. Koin dependency injection to make our code easy to understand.

**The benefit of Dependency Injection-**

* Code Reusability
* Ease to refactoring
* Ease to Test
* Make our architecture-independent

**12**. **Design Pattern in Android?**

Please follow the [link](https://p-mishra2905.medium.com/android-design-pattern-4c9ca9655844)

**13. Why Manifest.xml?**

Every Android App must have Manifest.xml. The manifest File is defined as the Essential operations in the app.

Here is the main list of data which we require to register in Manifest for the successful App run

1. **App component** — Activity, Services, Broadcast receiver, content provider. These all the Component may have some additional attribute to register, for example — Intent filters that describe how the component can be started

2. **Permissions** — It is used to create permissions to restrict access to protected parts of the system or other apps.

3. **Application —** Manifest can contain only one application node. In the application Tag, we can set some more attributes like title, icon and theme.

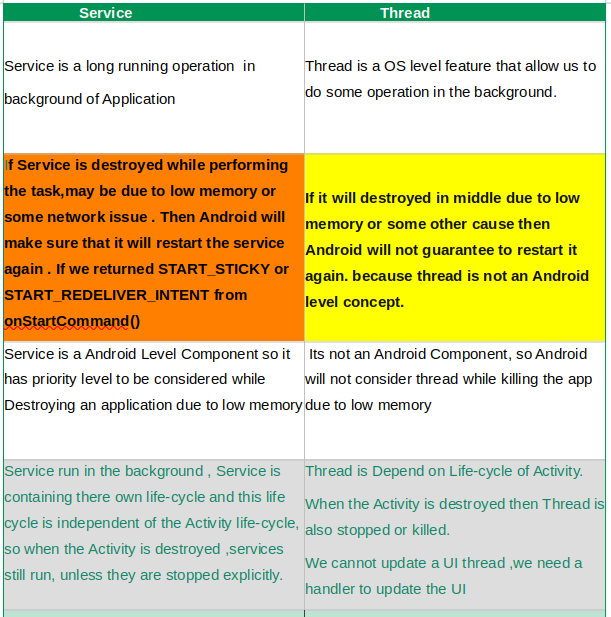
The Application node is to act as a container for the Activity, service, content provider and broadcast receiver.

4. **Uses-Feature —**It is used to specify which hardware features your application requires. This will prevent our application from being installed on a device that does not include a required piece of hardware.

5. **supports-screens — I**t is used to describe the screen support for our application

*<supports-screens  
android:smallScreens=”false”  
android:normalScreens=”true”  
android:largeScreens=”true”  
android:xlargeScreens=”true”/>*

**14. What is the difference between service and thread?**



**15. how to secure the Android Applications?**

When we communicate between two apps or websites, that time we need to take care of the stability of the apps and protect the data that we are sending or receiving from other apps. According to the [Android Developer](https://developer.android.com/topic/security/best-practices#kotlin) site, we should take care this way-

*Use implicit intents and non-exported content providers-  
Apply for signature-based permissions-  
Disallow access to your app’s content providers-  
Ask for credentials before showing sensitive information-  
Apply network security measures-  
Add a network security configuration-  
Create your own trust manager-  
Use WebView objects carefully-  
Use HTML message channels-  
Provide the right permissions-  
Store data safely-  
Store private data within internal storage-  
Store data in external storage based on the use case-  
Store only non-sensitive data in cache files-  
Use SharedPreferences in private mode-  
Keep services and dependencies up-to-date-*

The best explanation is in this [link](https://developer.android.com/codelabs/android-network-security-config#0) for the network security config.

16. What is the difference between **Handler** vs **Async Task** vs **Thread**?

* The **Handler** class can be used to register to a thread and provides a simple channel to send data to this thread. A Handler allows you to communicate back with the UI thread from other background threads.
* The **AsyncTask** class encapsulates the creation of a background process and the synchronization with the main thread. It also supports reporting the progress of the running tasks.
* And a **Thread** is basically the core element of multithreading which a developer can use with the following disadvantage: Handle synchronization with the main thread if you post back results to the user interface

**17. What is the difference between a Bundle and an Intent?**

* A Bundle is a collection of key-value pairs
* An Intent contains the information about the new Action, We can send the Data from one screen to another by using Intent. The Intent is used to start the new Component etc.

**18. Provide some tips to reduce battery usage in an android application**

The battery optimisation of Apps depends on many factors as-

1. **Reduce network calls if we can for some of the scenarios**— Cache your data and retrieve it from the cache when required next time.
2. **Batch operations**- we can batch the network calls if possible so that we can prevent the device from waking every time.
3. **Check All background process**
4. **Use GPS carefully-** will use GPS when required.
5. **Use Work Manager**- WorkManager incorporates the features of its predecessors in a modern, consistent API that works back to API level 14 while also being conscious of **battery life**.
6. **Avoid Wake locks**- Using a wake lock to keep a phone’s screen or CPU on is one of the most common ways an app will drain a battery.

**19. What’s the difference between a file, a class and an activity in android?**

* ***File —***It's a resource for storing information.
* ***Class*** — It’s a compiled form of a Java file. Android uses the .class files to produce an executable APK
* ***Activity*** — An activity is comparable to a Frame/Window in GUI toolkits. It is not a file or a file type, it is just a class that can be extended in Android for loading UI elements on view.

20. **What is Sticky Intent vs Pending Intent?**

**Sticky Intent —** When the Android System send the sticky Broadcast for any changes in battery status, the intent is available after the broadcast receiver has completed its work. after some time we can retrieve the Data from the intent regarding the battery status by using registerReceiver(method)

**Pending Intent —** When you want to open some application components like**Activity, Service or BroadcastReceiver** after some specified time interval you have to send **PendingIntent**in this case.

It's something we are giving permission to another app to run the app after some time. for example notification(if we want to open some activity on click on notification here we can use pending intent)