QA Ask

Network18

1. **What is app/apk Profiling**

Fixing performance problems involves identifying areas in which your app makes inefficient use of resources such as the **CPU Analysis**(Which thred & process taking huge CPU), **Memory Analysis(**Java, Native, Graphic, Stack, Code, Others => how & when once is taking more memory), Energy(CPU & Network taking more energy).

1. **What is app/apk analyzer**
2. Using the APK Analyzer can reduce the time you spend debugging issues with DEX files and resources within your app and help reduce your APK size
3. View the absolute and relative size of files in the app, such as the DEX and Android resource files.
4. Perform a side-by-side comparison of two APKs
5. **App Quality Insights**:

You can see and act on app crash data **from Firebase Crashlytics directly in Android Studio**. This integration pulls stack trace data and crash statistics from Crashlytics into the App Quality Insights tool window in the Studio IDE, so you don't have to jump back and forth between your browser and the Studio IDE

1. **App Inspection**:

Through App inspection we can inspect the **LocalDb Inspection**(SqLite, Room), **Backgroud Task Inspection**(Worker, Job, Alarm) and **Network Inspection**.

1. What is Diff between LiveData & Flow

**Live data** is part of Android Architecture Components which respect the other application component like activity & fragment and hold the updated/latest data only.

**Flow** : you can use a flow to receive live updates from a database. Flow can handle streams of values, and transform data in complex multi-threaded ways. StateFlow and SharedFlow are Flow APIs that enable flows to optimally emit state updates and emit values to multiple consumers

**(1..5).asFlow()**

**.filter {**

**println("Filter $it")**

**it % 2 == 0**

**}**

**.map {**

**println("Map $it")**

**"string $it"**

**}.collect {**

**println("Collect $it")**

**}**

**OR**

**fun simple(): Flow<Int> = flow {**

**println("Flow started")**

**for (i in 1..3) {**

**delay(100)**

**emit(i)**

**}**

**}**

**simple().collect { value -> println(value) }**

**Note:**

1. **asFlow**: flow the data
2. **filter**: we can filter the coming data
3. **map**: For re-form/ modify the comming data
4. **emit**: push/comit the daat
5. **collect** => collect is basically use for recivve or collect daat from flow
6. Bothe have similer prperty but beliw litlediffrent b/w them
   1. Flow is not lifecycle aware but LiveData is lifecyle aware
   2. Flow has got a bunch of different operators(map, filtter, tranform) which livedata doesn't have
   3. Flow continuously emits results while LiveData will update when all the data is fetched and return all the values at once
7. What is obserbable

Observability refers to the capability of an object to notify others about changes in its data. Like LiveData, Flow

It’s a Reactive programiing concet. Obserser have ability to notifty his object whne it’s data / data set has been chnage. Observer can emit strem of data and his obserbable/consumer can observe /receive the emited data.

1. What is design patter and how many type of design patter
2. Suppose we have login page and how we can handle it through MVVM
3. How you are handing the bug/crash/arn issue in android
4. How you can handle device specifi issue
5. List down the type of jetpack librery
6. What is work manager
7. What is diffrent biteween service and work manager
8. How to render & work UI in flutter with native UI
9. How to call method fron Flutter ⬄ Native
10. What is diff between REST and SOPA

|  |  |
| --- | --- |
| **SOAP API** | **REST API** |
| Relies on SOAP (Simple Object Access Protocol) | Relies on REST (Representational State Transfer) architecture using HTTP. |
| Transports data in standard XML format. | Generally transports data in JSON. It is based on URI. Because REST follows stateless model, REST does not enforces message format as XML or JSON etc. |
| Because it is XML based and relies on SOAP, it works with WSDL | It works with GET, POST, PUT, DELETE |
| Works over HTTP, HTTPS, SMTP, XMPP | Works over HTTP and HTTPS |
| Highly structured/typed | Less structured -> less bulky data |
| Designed with large enterprise applications in mind | Designed with mobile devices in mind |

1. How to make multiple request and wait until data is come from all the requests in retrofit 2.0 – android
   * 1. <https://www.digitalocean.com/community/tutorials/android-rxjava-retrofit>

**@** **Persistent Systems**

1. What is solid Principal and pls deiscribe it’s all terms and use benifits

it is the best practices of software development to deliver good quality of software/app because it’s help us to write clean, extendable, and testable of codebase/software.

1. **S => SRP = Single Responsibility principa**l : A class should have only one reason to change.
2. **O => OCP = Open-Close Principal:** open for extension but closed for modification
3. **L => LSP = Liskov substitution of principal**: “Derived types must be completely substitutable for their base types”
4. **I => ISP = Interface Segregation Principal:**  the interface segregation principle (ISP) states that no code should be forced to depend on methods it does not use
5. **D => DIP = Dependancy Inversion principal:** High-level modules should not depend on low-level modules. Both should depend on abstractions
6. What Design Pattern and How many type of design pattern and why we are using it.

Design patterns are solutions to general problems that software developers faced during software development. Design patterns are a set of solutions to common software development problems that have been proven to be effective through years of experience.

Three Type of Design Pattern

1. **Creation DP**: Singleton, Dependancy Injection, Fectory, Builder
2. **Structure DP**: Adapter, Fecade, Composite
3. **Bahviour DP**: Observer
4. What is singleton

The Singleton is a creational design pattern that ensures that a class has only one instance and provides a global access point to this instance.

1. What is AndroidViewModel and ViewModel

If you need to use context inside your Viewmodel you should use AndroidViewModel (AVM), because it contains the application context. To retrieve the context call getApplication(), otherwise use the regular ViewModel (VM).

AndroidViewModel and ViewModel is the same, the only difference is that AndroidViewModel contains the application context.

1. List of jetpack librery name

LiveData, ViewModel, DataBinding, ViewBinding, WorkManager, RoomDB, Navigation, Pagination

1. How to find the dublicate element in list using kotlin

val concrete2 = listOf(1, 3, 5, 7, 2, 8, 2, 1);

var uniqList2 = mutableListOf<Int>();

for(elementIndex in 0..concrete.size-1){

if(uniqList2.contains(concrete[elementIndex])){

println("Dublicate element found: ${concrete[elementIndex]}")

}else{

uniqList2.add(concrete[elementIndex])

}

}

**@Loma Technolog**

1. Tell me about your previous project feature and which all library you have used.
2. How you can integrate LIveStrimg
3. What is Push notification and steps we need to flow to implement it
4. What is DI and how we can achive it
5. What is lateinit and what is use
6. What is lazy
7. What is ViewModel class
8. What is MVVM & MVP and which one is best
9. What does means of if some variable declear using ? & !
10. How you can achive Data Binding in UI
11. Have ever write any test cases
12. Have knowledge of CI/CD pipeline or DevOps
13. What is pending intent and stiky intent
14. How to handle the notification
15. How to open Actvity from Notification
16. Is it possible to received notification if app is not open

**TechM**:

1. How to open another app activity
2. How to open and pass the data to another app
3. What all hardware functionality till you use
4. What is customeTab and how handle data
5. How is info.plist file
6. How we can open IOS activity/UIViewController
7. How to set Oriantion in IOS
8. Which state you need to check 1st before uploading/pushing the app

**@KOCH Ind**

1. Tell me about your current project
2. What is MVVM

The key idea behind MVVM is to separate the concerns and responsibilities of each component.

The ViewModel contains the application logic, state management, and data manipulation. The Model handles the data storage and retrieval, as well as any business rules.

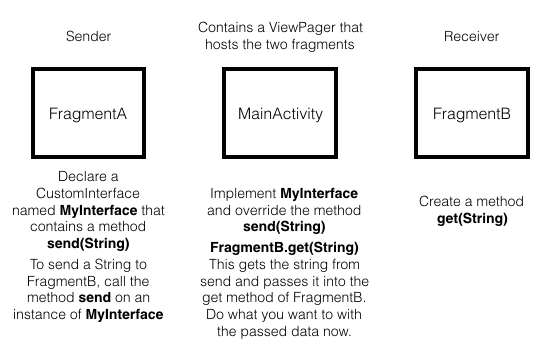
* 1. **Model:** The Model represents the data and business logic of the application. It encapsulates the data structures, database operations, network requests, and any other operations related to manipulating and retrieving data. The Model handles the data storage and retrieval, as well as any business rules.
  2. **View:** The View represents the UI components that are visible to the user. It could be an Activity, Fragment, or any other UI element. The View is kept as lightweight as possible and only handles UI-related tasks, such as rendering the data and capturing user input.
  3. **ViewModel**:

1. How many layer / class is required to to integrate MVVM
2. what is LiveData
   1. Live data or Mutable Live Data is an observable data holder class. We need this class because it makes it easier to handle the data inside ViewModel because it's aware of Android Lifecycles such as app components, activities, fragments, and services. The data will always update and aware of its Lifecycles.
3. Supposes you are getting a data from LiveData and screen is get roted/oriantaion changes and what happed with LiveData
   1. When the (...) fragment is re-created after a screen rotation, it moves from an inactive to an active state. The observer in the fragment is re-connected to the existing ViewModel and receives the current data.
4. **which method will be call at time of ViewMode object is distroying**

[onCleared](https://developer.android.com/reference/androidx/lifecycle/ViewModel#onCleared())(): This method will be called when this ViewModel is no longer used and will be destroyed.

1. **Data is DataBinding**
   1. Data Binding Library is a support library that allows you to bind UI components in your layouts to data sources in your app using a declarative format rather than programmatically.
2. **what is ViewBinding** 
   1. **View Binding** is one of the best features which provides the views to bind with the activity which is ongoing. Replacing the **findViewById()**method, hence reducing the boilerplate code, generated the instances of the views of the current layout.
3. **what is constrant layout**
4. **how to support multiple screen or design the layout to support all screen**
5. **Fragment Lifecycle** 
   1. onAttach(), onCreate(), onCreateView(), onActivityCreated(), onStart(), onResume(), onPouse(), onStop(), onDistroyedView(), onDistroyed(), onDiAttached()
6. **How to pass the data from once frgment to onother fragment**

We can pass the data b/w two fragment using Interface OR LiveData



1. **ViewHolder class in Recycler view**

**ViewHolder** The ViewHolder is a wrapper around a View that contains the layout for an individual item in the list. The Adapter creates ViewHolder objects as needed and also sets the data for those views. The process of associating views to their data is called binding. RecyclerView.ViewHolder: It is mandatory to use with recyclerView and helps us to draw the UI for individual items that we want to draw on the screen.

**onCreateViewHolder():** RecyclerView calls this method whenever it needs to create a new ViewHolder. The method creates and initializes the ViewHolder and its associated View, but does not fill in the view's contents—the ViewHolder has not yet been bound to specific data.

**onBindViewHolder():** RecyclerView calls this method to associate a ViewHolder with data. The method fetches the appropriate data and uses the data to fill in the view holder's layout. For example, if the RecyclerView displays a list of names, the method might find the appropriate name in the list and fill in the view holder's TextView widget.

**getItemCount():** RecyclerView calls this method to get the size of the dataset. For example, in an address book app, this might be the total number of addresses. RecyclerView uses this to determine when there are no more items that can be displayed.

1. **What is Build variants**
2. **what is build product Flover**
3. **How to check memory leakage in Android**
   1. By APK profiling
4. APK profiling
   1. Help us to understand the CPU utilization, Memory Utilization, Battry & Network Utilization
5. Can we extent the String class
   1. We can't extend String class because it's final
6. **What is Abstract class**

Abstract class is a class which declare by using Abstract keyword.. It can abstract(0 t0 100%) and non-abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

1. **What is Interface class**

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 5) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6) An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| 9)**Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

1. **What is default method and when/which varsion it’s introduce**

**Default methods**(Introdiced in Java-8) enable you to add new functionality to existing interfaces and ensure binary compatibility with code written for older versions of those interfaces. In particular, default methods enableyou to add methods that accept lambda expressions as parameters to existing interfaces. If you add/create new default method in your exiting Interface class which you have already implemented with multiple class, then you no need/not mandatory/option only to implement newaly added method.

1. **What is sirialization and Parceable class and which one is the best for Android**

Parcelable is an Android-specific interface that enables an object to be passed as a parameter from one activity to another. This is a more efficient method compared to serialization, as it doesn’t require the object to be converted into a byte stream. When an object is passed using parcelable, it is passed directly from one activity to another. Parcelable also has the advantage of being able to pass a large amount of data in a single call, making it more efficient than serializatio

1. **What is singleton class and write the logic**

Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the Java Virtual Machine. The singleton class must provide a global access point to get the instance of the class. Singleton Pattern says that just"define a class that has only one instance and provides a global point of access to it".

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

class A{

private static A obj;

private A(){}

public static A getA(){

if (obj == null){

synchronized(Singleton.class){

if (obj == null){

obj = new Singleton();//instance will be created at request time

}

}

}

return obj;

}

public void doSomething(){

//write your code

}

}

1. **How many way you can storethe data in Android**
   1. SharePrefrance, RoomDb, SqLite, File Manager
2. **Steps for Notification Integration**
3. **Lunch Mode**
   1. standard, singleTop, singleTask, singleInstance
4. **what is SingleTask and singleInstance**
   1. Single task will store single instance in activity stack and clear all activty instance from top of that. Like we have activity ABCD and we declear C as single Task. Right now we are on activity D and we are again calling Activty C from activty D and D instance we be distroyed and activity instance in stack will be ABC.
   2. Single Instance will be create saprate task . like we have an activity ABC and we decleared C as single Instance and we are lunching C from B. then C instance will be in separet Task Stack
   3. Task Stack -1 : AB
   4. Task Stack -2 : C
5. .

**@Adani**

1. Create App with TableView & bind data from local JSON

2. Can we override static method

3. We have Abstract class with 3 Abstract method and i want to implement only 1 method in child class

4. Parent P = new Child C();, Child C = new Child C(); , Parent P = new Parent P()

5. Activity life cycle

6. Life cycle state of=> ActivityA To ActivityB

7. Back Life cycle state of=> ActivityB To ActivityA

8. How to count number of Int in String

9. How to remove duplicate value from List

10. What singleton

11. what is MVVM

12. How to check, given number is odd or even

13. How to check given number is prime or not

14. what is coroutine

**@R Bharat on 17/09/21 at 12.00 PM**

1. What is PWA(Progressive Web App)

**@ArteriaTech on 20/09/21 at 6PM**

1. Tell Me About your self

2. Latest Architecture pattern

3. Latest architecture components

4. What is diff b/w MVVM & MVP

5. What is firebase

6. which ALC method will call on clicking on device Home Button

7. which ALC method will call on change of device orientation

8. what is Abstract & Interface

9. what is polymorphism

10. What is function overloading

11. what is final & static

12. what is build-Type, build-Flavour, build-Variant

**@PetroIT on 01/10/21 at 1PM**

1. Activity lice cycle

2. How you can save the data when app get terminated by system

3. CI/CD or CICD (continuous delivery or continuous deployment)

4. How you are sharing the testing build to tester

5. How you are performing Unit Testing

6. How push notification work

7. How may type of push notification

8. What is releam DB

9. How to check & status of notification is deliver or not

10. How to save Dynamic form data based on category

**@Aventior for IOS on 01/10/21 at 12PM**

1. IOS App is working on Single or Multi thread

2. Which catch policy have Session URL

3. What is the basic step for IOS Core Data

4. How to run code on background thread

5. How to update UI after run all n(5)

**@MyLoanCare for Android on at 1PM**

1. What is activity life cycle

2. what life cycle method will call on press on Home Button

3. what life cycle method will call on back press

4. what life cycle method will call on open app from bak stack

5. Fragment life cycle

6. what life cycle method will call when we add B fragment on top of A fragment

7. How we can handle camera/take picture and handle in Android 11

8. What is the broadcast receiver

9. What is changes in Android 8 in the broadcast receiver

10. what is WorkManager and how we can achieve it

11 what is the JobScheduler and how we can achieve it

12. which service is best for background job

13. what is corountine

14. what is difference b/w lifecycleScope & viewModelScope

15. What is MVVM and how we can achieve it

14 What is MVC & how we c an acheive it

14. what is difference b/w MVVM & MVC

15. what is RxJava

15. what is difference Rxjava and corountine

16. what is abstract class

17. what is open method

18.

**@CondecoSoftware for Android11PM on 13 Sept 22**

1. **What is scope function and how may type?**

**Scope:** scope functions are used to execute a block of code within the scope of an object. Generally, you can use scope functions to wrap a variable or a set of logic and return an object literal as your result.

**Type:** let, apply, with, run, also

1. **What is diffrence between use of let & when**
2. **What is scope function and what is diffrance between let, apply, also, when**
3. **what is diffrance between varlable?.let & varible.let**
4. **What dirrance between lateinit & lazy**

**lateinit can only be used with a var property whereas lazy will always be used with val property**. A lateinit property can be reinitialised again and again as per the use whereas the lazy property can only be initialised once

1. **whay use by lazy keyword with lazy**

lazy() is **a function that takes a lambda and returns an instance of lazy which can serve as a delegate of lazy properties upon which it has been applied**. It has been designed to prevent unnecessary initialization of objects. Lazy can be used only with non-NULLable variables. Variable can only be val.

1. **what is RxJava and how many type of Obserbable**

**RxJava** =>RxJava is a Java base implementation of ReactiveX. The ReactiveX (or Reactive Extensions) project aims to provide a reactive programming concept. It's a combination of the Observer pattern, the Iterator pattern, and functional programming.RxJava is **a reactive programming library** for composing asynchronous and event-based programs by using observable sequences

**Type of Observable : Observable** , **Flowable, Single, Maybe, Completable**

**RxAndroid :** RxAndroid is **an extension of RxJava with few added classes related to Android**. To be specific there are schedulers introduced in RxAndroid which plays a major role in supporting multi-thread operations in android. Schedulers decide if the block of code should run on a worker thread or the main thread

1. **How to handle onTextChnage Listerser & call the API on every text changes**

**Ans:** There is 3 metho for listen the text change listner in edittext/serach text box

**.1 BeforeTextChnaged(), OnTextChnaged(), AfterTextChnage()**

**Cancel The Retofir APi Call:**

**val retrfitCall = Retrofi.getInstance.getSearchData(search\_texy)**

**retrfitCall.enque()…………{**

**………**

**……….  
 }**

**if(!retrfitCall.isCanceled){**

**isCanceled. cancel()**

**}**

1. **What is services & background services**

**Ans:** Background or worker thread can be created within the app to run long running tasks in background without blocking the UI thread.

1. **what is diffrance between UI & Main Thered**

And: The UI thread, that is responsible for handling the UI events like Draw, Listen and receive the UI events.UI-Thread in Android is **a Thread element responsible for updating the layout elements of the application implicitly or explicitly**. This means, to update an element or change its attributes in the application layout ie the front-end of the application, one can make use of the UI-Thread

**Main thread** is what which start the process/app. In Android UI thread is main thread.

1. **Activity life cycle**

**Ans=>** oncreate, onstrat, onresume, onpouse, onstop, onrestart, ondestroyed

1. **when OnRestart Method will Call**

**Ans=>**When user back from actvity, Switching back from others app, relaes device back luck button, reopen the app from backstack

(onrestart, start, resume)

1. **is Alert dialog is part/subclass of activity**

**Ans=>** NO, AlertDialog is a subclass of Dialog

There is no any activity life cycle method will call when you open AlertDialog on Actvity or fragmneet

**@GlobalLogic @12.30PM on 13 Spet**

1. **Activity LC**
   1. oncreate(), onStart(), onResume(), onPouses(), onStop(), onRestart(), onDestroyed()
2. **Android Configuraction Chage method**
3. **What is configuration**
4. **Which method will call after orination change** 
   1. onPouse(), onStop(), onSaveInstanceState(), onDistroyed(), onCreate(), onStart(), onRestoreInstanceState(), onResume()
5. **What is services, Give me a real example**
   1. Service is android application component which run in background to for indefinite period of time
6. **What is Abstraction, Give me real example**
   1. Abstraction is process of hiding hiding their implimantion and internal logic and show only essential feature to user.
   2. *Hiding internal details and showing functionality* is known as abstraction. For example phone call, we don't know the internal processing.
7. **what class & Object, Gice me real example**
   1. **OOPs=>Object-Oriented Programming(system)** is a methodology or paradigm to design a program using classes and objects.
   2. class is a blueprint for object. It is template or blueprint or prototype where every object can be created. It is logical Entity. A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
   3. Object is an instance of class and every object having some identity & behaviour. Every entity is an object . Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.
8. **What is polymorphisum** 
   1. Polymorphism is way to perform a single task in different way.
9. **What is Inheritance**
   1. Inheritance is a mechanism by which once class(child) accrue the property & behaviour of others class(Parent)
   2. *When one object acquires all the properties and behaviors of a parent object*, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
10. **Reverse The String & remove all value from that string**

String str = “Abhishek”

char arr[] = new char[str.lenght]

for(int i= arr.length-1; i>=0; i—){

arr[arr.length - i - 1] = str.charAt(i)

}

String rev = String.valueOf(arr)

**@PWC @03.00PM on 14 Spet**

1. **What is singleton & Why we r using it**

Singleton is a design pattern that ensures that a class can only have one object. To create a singleton class. Using with like: Retrofitt, DB Class, Network call

**Java:** create constructor as private and Write a static method that has the return type object of this singleton class.

Class Singleton {

//private static instance variables

Privatestatic Singleton singleInstance = null;

//Private constructor

privateSingleton(){

}

//Static method to create instance of Singleton class

Public static Singleton Singleton(){

//To ensure only one instance is created

if(singleInstance == null){

singleInstance = new Singleton();

}

return singleInstance

}

}

**Kotlin:** In kotlin we can create singleton class by using Object keyword. The object class can have functions, properties, and the init method. The constructor method is not allowed in an object so we can use the init method if some initialization is required and the object can be defined inside a class. when we declare the class by using object keyword then kotlin compiler create private constructor & static reference for that class.

object  NewsService{

     val newsInstance:NewsInterface

     init {

         newsInstance=NewsService();

     }

}

1. **What is DI & why we r using it**
2. **What is Dragger**
3. **List Android Archetecture component**
4. **What is live Data**
5. **What is MVVM**
6. **What is Diff b/w MVC & MVVM**
7. **What is retrofit & why we are using it**

Retrofit is a networking librery which used for get/post the between client & server.

1. **Type of mysql joining**

retrieving data from two or more than two tables based on a common field. In other words, JOINs combine data from multiple tables in a result table based on a related column between those tables.

1. **What is Diff b/w inner & outer join**
2. **What is Primary key**

A primary key generally focuses on the uniqueness of the table. It assures the value in the specific column is unique

1. **what is foren key**

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

1. **what is relation b/w primary & foren key**

A primary key generally focuses on the uniqueness of the table. It assures the value in the specific column is unique. A foreign key is generally used to build a relationship between the two tables. The table allows only one primary key.

**@Incido @02.30PM on 15 Spet**

1. **How to get highest / top second (2nd highest) salery by qal command**

fun getSecondHighestNo(){

val arrr :MutableList<Int> = mutableListOf(3,76,8,5,7,9,2)

val sizex = arrr.size -1

for (i in 0..sizex){

var lowstIndex = i;

for (j in i..sizex){

if(arrr[j] < arrr[lowstIndex]){

lowstIndex = j

}

}

val temp = arrr[i];

arrr[i] = arrr[lowstIndex]

arrr[lowstIndex] = temp

}

println("sortest list ${arrr}")

println("second highest ${arrr[1]}")

}

1. **How to get multple user by his id/name**

SELECT \* FROM audit\_log\_webservice\_16\_09\_22 WHERE id IN (1, 2, 4)

1. **What is ROOM DB**

Room is a persistent library that is part of the Android jetpack. It is built on top of SQLite. The room persistent library has many advantages over raw SQLite

1. **Main class in ROOM DB**

**@Entity:** A Room entity includes fields for each column in the corresponding table in the database, including one or more columns that comprise the primary key

**@DAO:** The DAO is an interface that defines all the database operations which we want to do/performce on our entity/table to get/put the data. For this we declare methods without a method body and annotate them with @Insert, @Update, @Delete or the generic @Query, where we can pass an SQLite query.

**@DataBase:** Room Database class as Abstract to enable your class become flexible and skip implementing unnecessary methods of the RoomDatabase Base Class.

1. **Which are abstract / Interface class in ROOM DB**

@DataBase class is abstract class, @DAO is interface class

1. **what is transient**

**transient** : It will ignore this field while saving into database, BUT it will also ignore this field while parsing data which comes from server. transient is a Java construct, indicating that this field should not be serialized in standard Java serialization.By default, all of object's variables get converted into a persistent state. In some cases, you may want to avoid persisting some variables because you don't have the need to persist those variables. So you can declare those variables as transient. If the variable is declared as transient, then it will not be persisted.

**private transient String fullName;**

1. **what is the use of @Ignore**

**@Ignore** is a Room-specific annotation, saying that Room should ignore that field or method. It will only ignore this field while inserting data into database, but this field will participate into json parsing. **android.arch.persistence.room.Ignore.Ignores** the marked element from Room's processing logic. This annotation can be used in multiple places where Room processor runs. For instance, you can add it to a field of an Entity and Room will not persist that field.

**@Ignore val picture: Bitmap?**

1. **what is the diff b/w int, Int, Integer**

int: Java based Primitive Data Type

**Int:** Int is primitive data type and Int is a Kotlin Class derived from Number

**Integer:** Integer is wapper class. The Integer class wraps a value of the primitive type int in an object. An object of type Integer contains a single field whose type is int

1. **what is Int and IntArray**

An array of ints. When targeting the JVM, instances of this class are represented as int[]. Constructor: Creates a new array of the specified size, with all elements initialized to zero.

1. **What is Default ArrayList &LinkedList Size: 10**
2. **what is MVVM**

**Model: Model**

**View:**

**ViewModel:**

1. **How Model view internally work (instance are maange) (create & distored)**
2. **what is diff b/w ViewModel & AndroidView Model**
3. **what is mutableData & how it’s working**
4. **is mutableData is class or abstract class or interface?**
5. **How you can call multiple API and bind there data as per her response(bind order by responce not oredr by calling)**
6. **what is Diff b/w lateinit var and lazy**
7. **what is use the use of by with lazy and give example where you are using it**
8. **FCM feature**
9. **FFM notifiction class name & his manadatory method**
10. **whta is Application class**
11. **what is data class**
12. **what is object class**
13. **what is diff b/w const & val**
14. **how to check the nullability**
15. **how to check letinit var variable is null or not**
16. **what is lunch & aync**
17. **how to get out/result in aync coroutine**
18. **what is thread**
19. **what is Diff b/w wait() and await()**
20. **what is DI**
21. **what is Dragger & Hilt**
22. **can we layout on onstart() or onResume()**
23. **what services**
24. **Type of services**
25. **what is work manager**
26. **How to chedule task in work manager**
27. **can we schedule the task with alarm maneger**
28. **can work manager task resume/calss after reboot the device**
29. **can alarm manager work resume after reboot the device**
30. **what is init command**
31. **what is base and …**
32. **how to get update from other branch**
33. **What is JVM, JRE, JDK**