Android

HOW TO SET TOOLBAR AS ACTION BAR (Action-Bar is no more used in the recent update of android, use toolbar instead)

Toolbar is a view group that you can add other views like, EditText and TextView into it. Using it is as action bar is more easy because it’s just like any view and you can customize it anyhow you want. You can also customize the views in the Toolbar anyhow you want.

(But if you want a static actioBar that will contain only icon and a back-arrow, safely use Action-Bar )

Remember = It doesn’t mean Action-Bar does not work, no. It still works but you just replace it with Toolbar

Steps to Set Toolbar as Action Bar

1. Make Sure the Toolbar Dependency is added to your build.gradle file:

implementation 'androidx.appcompat:appcompat:1.0.0'

1. Disable the theme action-bar to parent="Theme.AppCompat.Light.NoActionBar" in your style.xml where your theme is (This will remove the ActionBar So that you can replace it with ToolBar)
2. Create a separate layout xml file to hold your Toolbar with a TextView, or anyView so that you can include it in any activity with the <include> tag. Make sure to add android:fitsSystemWindows="true" to the Toolbar container so that it will fit the width of every phone perfectly.

Inside res/layout/mytoolbar.xml

<androidx.constraintlayout.widget.ConstraintLayout

    android:fitsSystemWindows="true">

    <androidx.appcompat.widget.Toolbar

        app:layout\_constraintTop\_toTopOf="parent">

        <TextView

            android:text="TextView"

            android:layout\_gravity="center"/>

    </androidx.appcompat.widget.Toolbar>

</androidx.constraintlayout.widget.ConstraintLayout>

1. Then in your Activity.xml, include the above toolbar

 <androidx.constraintlayout.widget.ConstraintLayout>

 <!--INCLUDE THE TOOLBAR IN THE ACTIVITY XML-->

   <include

       layout="@layout/mytoolbar"

       android:layout\_height="wrap\_content"

       android:layout\_width="match\_parent"

       android:id="@+id/logintoobar"/> <!—give the toolbar id Here-->

</androidx.constraintlayout.widget.ConstraintLayout>

1. Add option Menu to the Toolbar. (This is optional if you don’t want any menu on the toolbar just ignore it and move to option 8 below)

Create a menu item in res/menu/mymenu.xml and inflate it to the Tool Bar. This represents the menu items that will show on the tool-bar

Inside res/menu/mymenu.xml

<menu>

<item

android:title="refresh"

<!-- The text that will be displayed on this menu -->

android:id="@+id/refresh

<!-- The id of this particular item -->

android:showAsAction="ifRoom"

<!-- Display this item in the Toolbar if there is space in the Toolbar

(if there is no space, it will be appended into the overflow on the Toolbar. The overflow is the 3 dot on the right that comes by default)

never = the item will always be shown in the overflow even if there is room -->

android:icon="@drawable/refresh"

<!-- Set icon for this item. -->

android:orderInCategory="100"

    <!-- the order/arrangement  of items in this menu container. 100 means arrange the items based on

    how you have coded them. if you want other items to come first before this item,

    add android:orderInCategory="99" -->

    </item>

</menu>

1. Then inside mainActivity.class, override the “onCreateOptionsMenu” method to inflate the above menu to the toolbar:

Inside MainActivity.class

@Override

    public boolean onCreateOptionsMenu(Menu menu) {

//Inflate the above menu

        getMenuInflater().inflate(R.menu.mymenu,menu);

        return super.onCreateOptionsMenu(menu);

    }

1. Handle clicks of the above inflated menu items by overriding the “onOptionsItemSelected” method in MainActivity.class. So that when any menu item is clicked in the toolbar, you can take some action.

Inside MainActivity.class

@Override

    public boolean onOptionsItemSelected(@NonNull MenuItem item) {

        // get the id of the item that was clicked

        switch (item.getItemId()){

            // if is the "refresh" item in res/menu/mymenu.xml

            case R.id.refresh:

            //display a toast

                Toast.makeText(MainActivity.this,"refresh click", Toast.LENGTH\_SHORT).show();

                break;

            default:

                Toast.makeText(MainActivity.this,"No function Click", Toast.LENGTH\_SHORT).show();

        }

        return super.onOptionsItemSelected(item);

    }

1. Then in your mainActivity.java, The Action-Bar is working by default. So, you need to tell android that you want to replace it with the above Toolbar:

  @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        //CREATE AN OBJECT OF THE TOOLBAR

       Toolbar toolbar =(Toolbar) findViewById(R.id.logintoolbar);

       // SET THE TOOLBAR TO WORK AS THE ACTION BAR BY PASSING THE OBJECT OF ABOVE TOOLBAR TO IT

        setSupportActionBar(toolbar);

//If you want to grab the id of the TextView in the Toolbar use: TextView txt = (TextView) toolbar.findViewById(R.id.textView\_id\_name)

    }

HOW TO ADD Title, subtitle And icon To Action Bar

You can manually set your own title and icons to show on the Action bar.

  protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        getSupportActionBar().setTitle("Justy App");  // set title

        getSupportActionBar().setSubtitle("welcome to my app"); //set a small SubTititle

        //SET ICON FOR THE ACTION BAR

        // these two below must be set to true before the icon will show

        getSupportActionBar().setDisplayShowHomeEnabled(true);

        getSupportActionBar().setDisplayUseLogoEnabled(true);

        // get the icon name from the drawable folder

        getSupportActionBar().setIcon(R.drawable.menu);

    }

HOW TO ADD BACK ICON TO ACTION-BAR TO NAVIGATE TO PARENT ACTIVITY

1. The Action-Bar back icon-arrow is set to false by default. Set it to true to display it in your MainActivity.class with:

*//show the default back-arrow so that it can be click to go back*getSupportActionBar().setDisplayHomeAsUpEnabled(true);

1. Then Inside Manifest.xml set the parent activity the back arrow should navigate to when clicked by adding android:parentActivityName attribute to the activity tag.

// this will navigate to the main Activity when the back-arrow icon is clicked

<activity android:name=".Login" android:parentActivityName=".MainActivity"></activity>

**===============================================================================================**

Meaning Of Application Context and Activity Context

In programming as a general, context represent the life of something.

In real life, the context of a man is the life God has given him. If God takes this context/life the man will die.

So context in android represent the life of an Activity or an Application.

Application context = This represent the life of your whole application. That’s when your app is running and the user terminate it, your application context/life dies.

Activity context = This represent the life of a specific activity in your application. That’s when that activity is running and the user switch to a different activity, it context/life dies.

So context is nothing but just the life-cycle of an activity or the application.

**===============================================================================================**

HOW TO CHANGE THE STATUS-BAR TO TRANSLUCENT

The status bar is the area of the user phone that shows the time when any app is opened.

Inside MainActivity.class do the following.

Build.VERSION.SDK.INT below will give you the API version of the user phone

   protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

// if the user phone api is 19 or greater than that

         if (Build.VERSION.SDK\_INT >= 19) {

             // set his status bar to translucent

            getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);

        }

// if his phone api is less than 19

        else {

            // don't set the status-bar to translucent

            getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);

        }

    }

**===============================================================================================**

To Hide The Status Bar

To hide the status bar, add below code in that activity oncreate() method

getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);

**===============================================================================================**

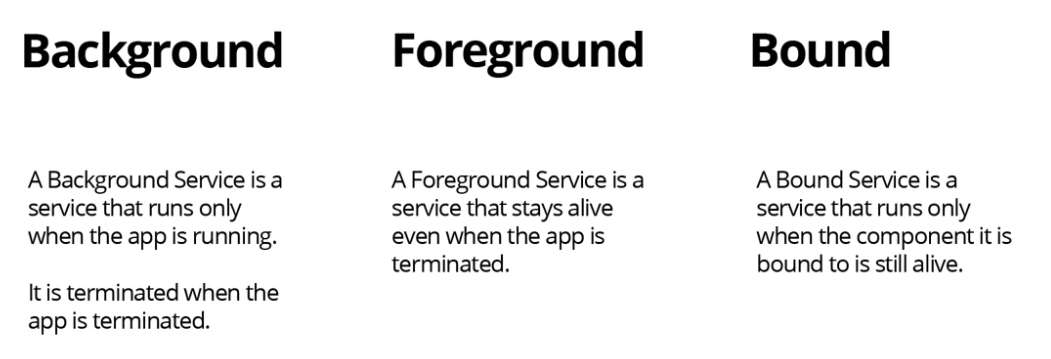
service

A service is an android application components for performing a long running operation in the background (that’s on a separate threads.)

Notes:

1. A Services has no UI or an interface where you can click on view items like Button and Textview.
2. A service will run even when the application that created the service is not running/ destroyed
3. Another application like Gmail can bound/call a service in another application (bound is just about binding and unbounding of other component to service)
4. By default, service runs on the main-thread. So, remember to explicitly put your service in a differently thread.

Types Of Service



**===============================================================================================**

VIEWMODEL And LiveDATA

View model is a container for storing data when configuration changes, like when the phone rotates.

When the phone rotate, the activity get destroyed and recreated again. This means if you have performed addition of two numbers and displayed the result on a Textview. When the phone rotate, the result will be lost because the activity will be destroyed and recreated again.

So we use Viewmodel to keep data for activity on-configuration changes, because View-Models don’t know anything about activity life-Circle. So when the activity is destroyed or recreated the data in the view model stays the same.

ViewModel has a method called “onCleared()” this method will be called in the viewmodel to clear all the data inside it only when the application is destroyed.(not when an activity is destroyed). You need to override this method if you want to save some data in db before the user exit the app.

NOTE ===, when the device rotate, it calls the activity “ondestroyed()” life-cycle method, but it will not call the “onCleared*()” method in viewmodel.*

But when the application is exited, it calls the current openned activity “ondestroyed()” life-cycle method and the “onCleared*()”*” method In ViewModel as well. That’s how google has structured it (Note= the difference very well)

ViewModel is just like Service in Angular, get data from database to ViewModel and let activities or fragment fetch the data from the ViewModel.

Note🡺 never let your ViewModel reference any Activity class or make reference to any View. That’s the general rule from goggle. Because if you do, the ViewModule will be tied to the activity life-cycle which will cause it data to change when the activity life circle changes.

LiveData= is a generic abstract class that ViewModel exposed it live data to. What it means is that, whenever the data in your viewModel changes, the viewmodel will expose such data together with the old data to liveData class.

Let say you have selected a list of data from a database and stored it in a viewModel, Then you have fetched these data from the viewModel and displayed it in your application UI.

Now, if one more data is added to the database. How will you show it automatically on the UI without running another sql query?

That’s where LiveData comes in. Whenever viewModel sees a change in it Data, it will quickly expose the new changes to a LiveData class with the help of MutableLiveData.

MutableLiveData is just like a road that, ViewModel walks on it to put data into LiveData class.

So know the difference.

ViewModel, is a data Storage center for Activities and fragment which is not affected by configuration changes, like when the phone rotate and the activity is destroyed.

LiveData, Is a storage center where ViewModel exposes it data to.

So all, you have to do, is to let your activity or fragment observe the ViewModel if you want to get data from it.

**How To Create And use ViewModel**

To create a view Model, Add the ViewModel and liveData life-cycle dependency to your build.gradle file (Below is both viewModel and LiveData dependency combined):

implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

Then simply creates a new class and let it extend ViewModel. This is the class where all your UI data will be fetched from. Example like array of Names or data from database.

Then in your activity, create a new ViewModelProvider that takes in the activity context and pass the above extended class to it get method like the format below.

ViewModel\_extended\_class\_name object = new ViewModelProvider(this).get(ViewModel\_extended\_class\_name.class);

Then let your activity, Observe the viewModel for data changes so that the UI can be updated automatically whenever there is new data.

Eg:

(1) Create a new Class that extends ViewModel

public class ViewModelData extends **ViewModel** {

// create an object of type MutableLiveData<T>

    private MutableLiveData<int> value1 = new MutableLiveData<>();

    //create getters, setters and toString method for the above mutableLivedata Object “value1”

// Setter for storing values, this is the method where your activity will send data

    public void setValue1(int value1) {

        // add +1 to the value ( setValue() is a method used for setting values for MutableLiveData Objects)

        this.value1.setValue(this.value1 + value1);

    }

//LiveData type getter for getting values

// This is the method your activity needs to observe for data to update the ui. Remember it has a type of LivedData<T>

    public LiveData<int> getValue1() {

        return value1;

    }

// + toString method goes below but I have cut it off for simplicity

// you can override the ViewModel oncleared() method to see when the above values will be cleared and check it on the log panel of the IDE

//  This method will only be called when the application is destroyed or exited. It will never be called when configuration changes.

        @Override

    protected void onCleared() {

        Log.d("ViewModelONdestroyed", "app terminated, so view model cleared ");

        super.onCleared();

    }

}

Then Inside your MainActivity, Set and Get the value of the above extended ViewlModel class property “value1” when I clicked a button and Assign it to a TextView

Inside MainActivity.java

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

// Object of TextView, Button and the Extended ViewModel class

TextView  textView ;

Button button;

ViewModelData viewModelData;

//inside the oncreate method

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

//Create a TextView and a Button Object

        textView = (TextView) findViewById(R.id.textView);

        button = (Button) findViewById(R.id.button);

        //set onclicklistener for the button

        button.setOnClickListener(this);

// Create object of the above extend viewModel class by using ViewModelProvider

// new ViewModelProvider(which\_activity\_is\_to\_receive\_the\_data?).get(Which\_class\_is\_the\_data\_stored?);

      viewModelData = new ViewModelProvider(this).get(ViewModelData.class);

    }

// Custom method to set and get Data From the Extended ViewModel Class When the above button is clicked

 public void getViewModelData(int value){

     // set a value for the extended ViewModel class property value1

            viewModelData.setValue1(value);

     //get the data in extended ViewModel

     // (That's observe it so that whenever there is changes in the viewModal data the blow textView value will be updated automatically)

    viewModelData.getValue1().observe(this, somedata ->{

        // "somedata" above is the data that is coming from the getValue1() method in the viewModel

        // then assign it to the TextView

            textView.setText(somedata);

        });

 }

// Button onClicked listerner inferface overrriden method

        @Override

    public void onClick(View v) {

       if (v.getId()==R.id.button){

           // call the abvoe "getViewModelData" method when the button is clicked so that the Textview value will be increased

           getViewModelData(1);

       }

    }

/\* if the application get Destroyed and there is some data you want to show to the user when he is back on the application,

Override the onDestroyed method and save the Data in the viewModel to a database or shared preference.

\*/

  @Override

    protected void onDestroy() {

        super.onDestroy();

    }

}

The MVVM Architecture (Model view, viewModel)

The MVVM architecture gives you a clean way to organize your project. Example. The model inserts or fetch data from db into viewModel, then the activities or fragmant fetch those data in view model.



**===============================================================================================**

BOTTOM NAVIGATION

Bottom navigation is simply the horizontal list of menus that will show in the bottom of the app.

NOTE= It can contain only up to 5 menu items

1. Add the material design dependency to your build.gradel file: implementation 'com.google.android.material:material:1.3.0'
2. Then Create a menu item that will be shown on your bottom navigation in res/menu/mymenu.xml with text and icon
3. Inside res/menu/mymenu.xml
4. <menu xmlns:android="http://schemas.android.com/apk/res/android"
5. xmlns:app="http://schemas.android.com/apk/res-auto">
7. <!--item One-->
8. <item
9. android:id="@+id/home"
10. android:title="home"
11. android:enabled="true"
12. android:icon="@drawable/home"
13. android:orderInCategory="100"
14. app:showAsAction="always"/>
15. <!--item two-->
16. <item
17. android:id="@+id/about"
18. android:title="about"
19. android:enabled="true"
20. android:icon="@drawable/about"
21. android:orderInCategory="100"
22. app:showAsAction="always"/>
23. </menu>
24. Then inside your Activity.xml file, Add a BottomNavigationView and add the above menu to the it with app:menu="@menu/menu\_Name"

Inside mainActivity.xml

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android">

<!--Add BottomNavigationView-->

    <com.google.android.material.bottomnavigation.BottomNavigationView

        android:id="@+id/mybottomnav"

         app:menu="@menu/mymenu" <!--add the Above menu here-->

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:background="@color/cardview\_light\_background"

        app:elevation="8dp" <!— Lift the navigation view up, it like z-index in css-->

        app:layout\_constraintBottom\_toBottomOf="parent"

        app:layout\_constraintEnd\_toEndOf="parent"

        app:layout\_constraintHorizontal\_bias="1.0"

        app:layout\_constraintStart\_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

1. Then create an object of your BottomNavigationView in your mainActivity.java.

Then handle clicks of menu items with OnNavigationItemSelectedListener

You can also handle clicks of menu items Reselection, that’s when an item is re- clicked with OnNavigationItemReselectedListener

Inside MainActivity.java

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    BottomNavigationView bottomnav;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

       // Create Object of Bottomnavigationview

        bottomnav = (BottomNavigationView) findViewById(R.id.mybottomnav);

        //handle clicks of Bottom navigation

        bottomnav.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

            @Override

            public boolean onNavigationItemSelected(@NonNull MenuItem item) {

                if (item.getItemId()==R.id.about) {

                    Toast.makeText(getApplicationContext(), "about clicked", Toast.LENGTH\_SHORT).show();

                }

                //make sure it always return true so that, it will highlight the clicked menu item

                return true;

            }

        });

        // Use to handle reclick or ReSelection of bottom navigation menu items (eg: when home menu is clicked twice or more)

        bottomnav.setOnNavigationItemReselectedListener(new BottomNavigationView.OnNavigationItemReselectedListener() {

            @Override

            public void onNavigationItemReselected(@NonNull MenuItem item) {

                if (item.getItemId()==R.id.about) {

                    Toast.makeText(getApplicationContext(), "about menu ReClicked", Toast.LENGTH\_SHORT).show();

                }

            }

        });

    }

}

**===============================================================================================**

FRAGEMENT

Because a single fragment can be used multiple times in the same activity and other activities. Your fragment should only contain a logic to

manage that particular fragment ui. Never let a fragment reference another fragment, or manipulate one fragment in the other. Or to communicate directly with

other fragment or the activity the fragment will be embeded in.

You can remove, add, replace a fragment while the activity that the fragment is embeded in is running.

A fragment has it own life cycle.

fragment require a dependency of : implementation "androidx.fragment:fragment:1.3.2"

To create a fragment, add the fragment dependency and simply create a new custom xml layout file to represent the view of your fragment.

Then create a new class that extends Fragment and inflate the above xml layout for this fragment. Eg:

// Inside fragment\_my.xml

<FrameLayout>

    <TextView/>

</FrameLayout>

//Inside MyFragment.java

public class MyFragment extends Fragment {

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the above layout for this fragment (so the above xml will be the ui of this fragment)

        return inflater.inflate(R.layout.fragment\_my, container, false);

    }

}

Then to add this above fragment programatically to an Activity:

We use FragmentTransaction to perfom transaction of fragment, that's to add, remove and replace fragment in activities, it called fragment transaction.

You need to create an object of fragmentManager which is use for managing transactions.

But first make sure your Activity.xml layout contains a fragmentcontainerview for embeding fragment in.

(You can use any other viewGroup though, but google recomend fragmentcontainerview )

Eg:

Inside Activity.xml

<!--Use FragmentContainerView as the container for embeding fragment in-->

    <androidx.fragment.app.FragmentContainerView

        android:id="@+id/myfragment"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        app:layout\_constraintBottom\_toTopOf="@+id/textView"

        app:layout\_constraintEnd\_toEndOf="parent"

        app:layout\_constraintStart\_toStartOf="parent"

        app:layout\_constraintTop\_toTopOf="parent" />

Then inside Activity.Class

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // savedInstanceState will ensure that the fragment will be added only once instead of getting recreated whenever

        // there is a configuration changes. Becuase when the device rotate savedInstanceState will no longer be null and

        // that means the activity will be created only once. Becuase when the device rotate, the instance of the fragment

        // will be fetch from the savedInstanceState instead of recreating it whenever there is a configuration changes.

        // Note when configuration changes, the activity get destroyed and recreated.

        if(savedInstanceState==null){

            // get an object of fragmentManager

            FragmentManager fM=getSupportFragmentManager();

            //use fragment manager to begin a transaction

            FragmentTransaction ft= fM.beginTransaction();

            // add the fragment, Note=

            // ft.add(the\_id\_of\_view\_group\_you-want-to-embed-the-fragment, Class\_of\_the\_fragment, bundle, tag\_to\_find\_the\_fragment)

            ft.add(R.id.myfragment,MyFragment.class,null,"myFragementtag");

            // always se this to true, it helps in avoiding crash of fragment during transaction

            ft.setReorderingAllowed(true);

            //put the fragment to backstack, so that when the user navigate forward and moves backward he can access the fragment

            ft.addToBackStack("any\_string\_here");

            //make sure to commit

            ft.commit();

            // to replace a fragment use: ft.replace(viwgroup\_to\_replace, Newfragment\_class\_to\_be\_replaced, bundle)

            // ft.replace(R.id.myfragment, NewFragmentClass.class, null)

            // ft.commit();

            // To Remove A fragment use: (1) find the fragment, (2) remove it (3) commit it

            // MyFragment myFragment = (MyFragment) fM.findFragmentByTag("myFragementtag");

            // ft.remove(myFragment);

            // ft.commit();

            // To find a fragment by tag use

            // MyFragment fragment = (MyFragment) fm.findFragmentByTag("tag");

        }

    }

**TO ADD ANIMATIONS TO FRAGMENT**

You can easily add animations to framgment during transactions. Does when a fragment is being replaced or being added, the kind of animation it should have.

you can easily do it withe the setCustomAnimations() method of the fragmentTransaction.

This setCustomAnimations() can takes in any number of animation resource id. but mainly 2 is enough. The syntess is:

setCustomAnimations(the\_outgoing\_flagment\_animation, the\_incoming\_fragment\_animation);

The parameters passed into the setCustomAnimations(); are the names of different animation file define in res/anim/anim\_fileName.xml

Eg.

Create a new animation file res/anim/fadein.xml

Inside res/anim/fadein.xml

<?xml version="1.0" encoding="utf-8"?>

<alpha xmlns:android="http://schemas.android.com/apk/res/android"

    android:duration="@android:integer/config\_shortAnimTime"

    android:interpolator="@android:anim/decelerate\_interpolator"

    android:fromAlpha="0"

    android:toAlpha="1">

</alpha>

Create another animation file slideout.xml

<?xml version="1.0" encoding="utf-8"?>

<translate xmlns:android="http://schemas.android.com/apk/res/android"

    android:duration="@android:integer/config\_shortAnimTime"

    android:interpolator="@android:anim/decelerate\_interpolator"

    android:fromXDelta="0%"

    android:toXDelta="100%">

</translate>

Then inside your MainActivity.java replace a new flagment with animation when a button is clicked.

 mybutton.setOnClickListener(new View.OnClickListener() {

                @Override

                public void onClick(View v) {

                     FragmentManager fM=getSupportFragmentManager();

                     FragmentTransaction ft= fM.beginTransaction();

                    // setCustomAnimations(outgoing\_fragment\_animation, incoming\_fragment-animation);

                   //  these are the above two animations created in res/anim

                    ft.setCustomAnimations(R.anim.slideout,R.anim.fadein);

                    ft.replace(R.id.myfragment,FragmentB.class,null,"fragb");

                    ft.setReorderingAllowed(true);

                    ft.addToBackStack("fragb");

                    ft.commit();

                }

            });

**Inter Fragment Communication + Activity**

The use of interface for sending data from one fragment to the other is no more recommended by google. They use viewModel instead.

All you need to do is to let one fragment send data to a viewModel and let the other fragment grab it.

Example: Let send data from one fragement to the other using viewModel…

// Inside the custome view Model class

public class SharedViewModel extends ViewModel {

    // create an object of MutableLiveData to set and store values

    private final MutableLiveData<Item> selected = new MutableLiveData<Item>();

// setter to set value

    public void setSelected(Item item) {

        selected.setValue(item);

    }

// getter to get valus

    public LiveData<Item> getSelected() {

        return selected;

    }

}

//Inside MasterFragment.Java send some value to the above viewModel when a button is clicked

public class MasterFragment extends Fragment {

    private SharedViewModel model;

    Button button;

    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {

        super.onViewCreated(view, savedInstanceState);

        // use the above "view" to define object for your views

        button= (Button) view.findViewById(R.id.mybtn);

        /\* make sure to use requireActivity() below in the ViewModelProvider() and not "this", becuase this fragment will be embeded

into activity.

         Fragments have their on activity context. So if you use "this" and when the fragment is attatched to an activity,

         it will try to use the fragment context, which will cause an error in the activity

        But when you use ViewModelProvider(requireActivity())

        the activity will replace below requireActivity() with this. so it will be ViewModelProvider(this) in the activity

         \*/

        model = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);

        // button onclick lister

        button.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

            // sed a value to the above ViewModel class

               model.setSelected(item);

            }

        });

    }

}

// Inside Detail fragment.java (Grab the data above MasterFragment has send into the ViewModel)

public class DetailFragment extends Fragment {

    TextView texview;

    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {

        super.onViewCreated(view, savedInstanceState);

         textview= (TextView) view.findViewById(R.id.mytext);

        SharedViewModel model = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);

        //below "item" represent the data coming from the getSelected() method in the viewModel

        // Make sure to pass getViewLifecycleOwner() to the below observe method and not "this"

        model.getSelected().observe(getViewLifecycleOwner(), item -> {

           texview.setText(item);

        });

    }

}

// So simple, let one fragment send data to a view model and retrieve that data in another fragment.

// (You can use the same approach to send data from fragment to activity and vicy versa)

**===============================================================================================**

Dependencies in Build.Gradle File

The dependencies that we add to the build.gradle file are just like the dependencies we install in node.js with npm.

Adding some dependencies on the build.graddle file will give you access to all the methods and properties of that particular library class.

So, dependency actually means, adding some third-party library functionalities into your application.

**===============================================================================================**

**Lottie Animation**

Littie is a loading animations you can download and include it in your project. It just like any view that you need to include in your layout.

All you need is to simply go to <https://lottiefiles.com/featured> and download any animation you like and use it in your project. Remember the animations are in json, so always download the json format and follow below to see how to use it. (you can also edit any animation from the website before you include it in your project)

It requires a dependency of implementation 'com.airbnb.android:lottie:3.7.0'

**How to Use Lottie**

1. include the lottie dependency in your build.gradle file implementation 'com.airbnb.android:lottie:3.7.0'
2. Download the animation you want to use from <https://lottiefiles.com/featured> to your local drive.
3. Then go to android studio and create new res/raw file and put the above downloaded json animation in the raw folder. If it shows an error, simply rename the animation file you just downloaded because most their animation file names start with a number, which is not allowed in android studio.
4. Then include the Lottie animation view to your layout file and position it where you want it to be. Then assign the downloaded animation to it like:

<androidx.constraintlayout.widget.ConstraintLayout >

    <TextView

        android:id="@+id/appnameimage"

        android:layout\_width="300dp"

        android:layout\_height="wrap\_content"/>

// add LottieAnimationView

    <com.airbnb.lottie.LottieAnimationView

        android:id="@+id/lottieopenactivity"  // give it an id

        app:lottie\_autoPlay="true"           // set this to true if you want the animation to start automatically

        app:lottie\_rawRes="@raw/diamond"    //  choose downloaded lottle json animation you want

        android:layout\_width="wrap\_content"

        android:layout\_height="200dp"

        app:layout\_constraintStart\_toStartOf="parent"

        app:layout\_constraintTop\_toBottomOf="@+id/appnameimage"/>

</androidx.constraintlayout.widget.ConstraintLayout>

1. Then inside your mainActivity.java, create an object of your above lottie animation and animate it Like:

// create an object of your lottie Animation View

LottieAnimationView mylottie= (LottieAnimationView) findViewById(R.id.lottieopenactivity);

// Then animate it

        /\*

        animate()==> animate this ui

        translationY(-10)=> move from buttom to top

        setDuration(1000)=> how long the animation should stop

        setStartDelay(4000)=> seconds to waite before starting the animation (1000 == 1 seconds)

         \*/

        mylottie.animate().translationY(14).setDuration(1000).setStartDelay(5000);

// you will not see any result until you run the application

        /\*

        Remember - the above animation methods are from android not lottie.

        So you can use it to animate any other view programtically like:

        myTextView.animate().translationY(-10).setDuration(1000).setStartDelay(4000);

        logoImage.animate().translationY(14).setDuration(1000).setStartDelay(4000);

        appNameImage.animate().translationY(14).setDuration(1000).setStartDelay(4000);

         \*/

**===============================================================================================**

**Splash Screen**

Splash screen is the first page that shows in your application whenever it starts. It like a welcome screen, then after it waits for some seconds and navigates to the homepage of the application.

All you need is to create a splash screen activity with it own layout. Then set this activity in your manifest file to be the activity to run first always.

Then use below code to wait for some seconds and start the homepage activity. (We usually animate our splash screen activity layout)

Note = There is no class or adapter in android for implementing splash screen. All we do is to set one activity to be the one to open first always, then we delay this activity for some seconds before we start the homepage activity.

(Always hide both statusbar and the ActionBar in your splash screen Activity, google how to hide status bar)

Inside mainActivity.jave Below code goes into the oncreate method

// new Handlere()  is a class that has a method PostDelayed() which is used to schedule task that should be run after some time

// the postDelay method takes in two parameters: new Runnable(){}, the\_time\_to\_waite\_before\_executing\_the-code\_inthe-new-runnable-block

 new Handler().postDelayed(new Runnable() {

            @Override

            public void run() {

                // after 3 seconds below block-code will run (1000== 1 seconds )

                // start the new Activity

                Intent intent = new Intent(MainActivity.this, Login.class);

                startActivity(intent);

                // destroy the activity so that if the user press the back-button he will not find it

                finish();

            }

        }, 3000);

**===============================================================================================**

Glide

Glide is a library for loading images from the server, it really good because it implement image caching and it implementation is easy.

It request below two dependency of:

implementation 'com.github.bumptech.glide:glide:4.12.0'

annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'

HOW TO USE GLIDE

1. Include the above two dependency in your build.gradle file
2. then add an ImageView in your xml file, but don’t assign any image to it

<androidx.constraintlayout.widget.ConstraintLayout

    android:layout\_width="match\_parent">

    <TextView

        android:id="@+id/textView"

        android:layout\_width="wrap\_content"/>

// Don't assign any src image to this image

    <ImageView

        android:id="@+id/imageView"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"/>

</androidx.constraintlayout.widget.ConstraintLayout>

1. Then inside MainActivy.java, create an object of the above image, and use glidle to load the image and assign it to it

Inside MainActivity.java

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

ImageView imageView = (ImageView) findViewById(R.id.imageView);

// the url the image will be loaded from

String url= "https://i.picsum.photos/id/667/200/200.jpg?hmac=Dqc51PnEPXpiiStRcDoPxytal0MOGvzg-eDZ4BsVIz8";

// load the image and display it in the above imageview.

// placeholder(a\_default\_image\_which\_will\_be\_replaced\_when\_the\_image\_load)

// into(the\_object\_of\_the\_imageView\_to\_display\_the\_loaded\_image)

Glide.with(this).load(url).placeholder(R.drawable.bg).into(imageView); // there are more methods, check their site to see

}

**==============================================================================================**

**RecycleView**

Recycleview is an advance version of Listview. It actually help you to show list of element in a customize way you prefer than listview. Google recommend using Recycleview over listview.

It requires the following dependencies:

    implementation "androidx.recyclerview:recyclerview:1.2.0"  
    // For control over item selection of both touch and mouse driven selection  
    implementation "androidx.recyclerview:recyclerview-selection:1.1.0"

How to Create RecycleView

1. Create a custom xml layout and define how you want each element in your Recycleview to look like

Inside customlayout.xml

<!--below means, Each recycleView is going to have a linearlayout with element of Button & ImaageVies-->

<androidx.appcompat.widget.LinearLayoutCompat

    android:orientation="horizontal"

    android:layout\_margin="2dp"

    android:layout\_height="50dp">

    <Button

        android:id="@+id/mytextview"

        android:layout\_width="wrap\_content"

        android:textColor="@color/white"

        android:layout\_height="wrap\_content"

        android:text="button"/>

    <ImageView

        android:id="@+id/myimage"

        android:layout\_width="50dp"

        android:layout\_height="50dp" />

</androidx.appcompat.widget.LinearLayoutCompat>

1. Then inside your mainActivity.xml, define a <RecycleView> tag where you will embed the above custom layout into it programmatically

Inside MainActivity.xml

<androidx.constraintlayout.widget.ConstraintLayout

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    tools:context=".MainActivity">

<!--Define recycleView tag for embeding custom layout in programatically-->

    <androidx.recyclerview.widget.RecyclerView

        android:layout\_width="match\_parent"

        android:id="@+id/myrecycleview"

        android:layout\_height="match\_parent"

        app:layout\_constraintBottom\_toBottomOf="parent"

        app:layout\_constraintEnd\_toEndOf="parent"

        app:layout\_constraintStart\_toStartOf="parent"

        app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

1. Inside mainActivity.java, create object of above recycleView and set a layout manager and an adapter for it.

Inside MainActivity.java

public class MainActivity extends AppCompatActivity {

RecyclerView recyclerView;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        //create object of the recyclview in the xml

        recyclerView= (RecyclerView) findViewById(R.id.myrecycleview);

        // set a layout manager for your recyleview, thats either you want vertical, horizontal or grid order of your Recyview

        LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this, LinearLayoutManager.VERTICAL,false);

        recyclerView.setLayoutManager(linearLayoutManager);

        // these are the list of Strings and images that are going to be assign to your Recyview itmes

        String[] myNames= {"Justice", "Efo", "Justilato", "iniesta", "Ankomah", "Os3se", "45", "onesan", "asay"};

        int[] images = {R.drawable.home};

        // pass the above data to the recyviewAdapter and set it as and adapter for your recycleView

        MyRecycleViewAdapter myRecycleViewAdapter=new MyRecycleViewAdapter(myNames,images);

        recyclerView.setAdapter(myRecycleViewAdapter);

    }

}

1. Create a new adapter class that extends RecyclerView.Adapter<InnerClass\_that\_extends\_”RecyclerView.ViewHolder”\_here> and implements it methods. Remember this class must have an inner class that extends “RecyclerView.ViewHolder”

Inside this adapter class is where you inflate your custom xml layout into the mainActivity.xml <recycleview> tag

Then you create object of your views and assign values to it, then you handle click of items.

Inside MyRecycleViewAdapter.java

// create a class that extends RecyclerView.Adapter<Inner\_Class\_that\_extends\_"RecyclerView.ViewHolder"\_here>

public class MyRecycleViewAdapter extends RecyclerView.Adapter<MyRecycleViewAdapter.CustomViewHolder> {

    String[] names;

    int[] images;

    public MyRecycleViewAdapter(String[] names, int[] images) {

        this.names = names;

        this.images=images;

    }

    @NonNull

    @Override

            // purpose is == to inflate the custom xml layout into the <recyclview> tag of mainActivity.xml

    public CustomViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

        /\*(1) Inflate the custom xml layout you created into the <RecycleView> tag in mainActivity.xml

             so now, mainActivity.xml contains <Recycleview> tag contains a child element of:

            <RecycleView>

               <Linearlayout>

                   <EditText>

                   <Imageview>

                </linerlayout>

            <RecycleView>

             so this "view" object represent the <recycleView> tag in mainActivity.xml with it above child elements

        \*/

        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.recyclelayout,parent,false);

        // (2) now pass the the above <recycleview> tag object with it inner Element to CutomerViewholder Class

        return new CustomViewHolder(view);

    }

    // purpose ==> is to set values for the views object in your

    // so if you have an ImageView inflated into your recycleview, here is wher to set the src of the image

    @Override

    public void onBindViewHolder(@NonNull CustomViewHolder holder, int position) {

        // above "holder" is an Object of "CustomViewHolder" use it to set values for the views

   holder.button.setText(names[position]);

   holder.myImageLogo.setImageResource(images[0]);

    }

    // this is where the recycleview adapter get to know the number of Recyviews it should create

    // all you need is to return the length of datas or values you want to assign to your views.

    // Thats the array of values you want to assign to your recycleviews object

    @Override

    public int getItemCount() {

        return names.length;

    }

    // Beginning of CustomViewHolder (purpose== create object of your inflated Views and handle OnClickListenrs)

    class  CustomViewHolder extends RecyclerView.ViewHolder{

        Button button;

        ImageView myImageLogo;

        Context context;

        // (3) This method is called by onCreateViewHolder() method above

        // "recyleViewTag" represent the <recycleViewtag> object with it child element

        public CustomViewHolder(@NonNull View recyleViewTag) {

            super(recyleViewTag);

            // Create Object of ImageView and a Button that are the child elemtent of the Recycleview

            // (Their data or values will be set to it by onBindViewHolder method Above)

            myImageLogo = (ImageView) recyleViewTag.findViewById(R.id.myimage);

            button= (Button) recyleViewTag.findViewById(R.id.mytextview);

            // DEFINE OR HANDLE ONCLICK LISTENER HERE

            button.setOnClickListener(new View.OnClickListener() {

                @Override

                public void onClick(View v) {

                    Log.d("clicked", (String) button.getText());

                }

            });

        }

    }

}

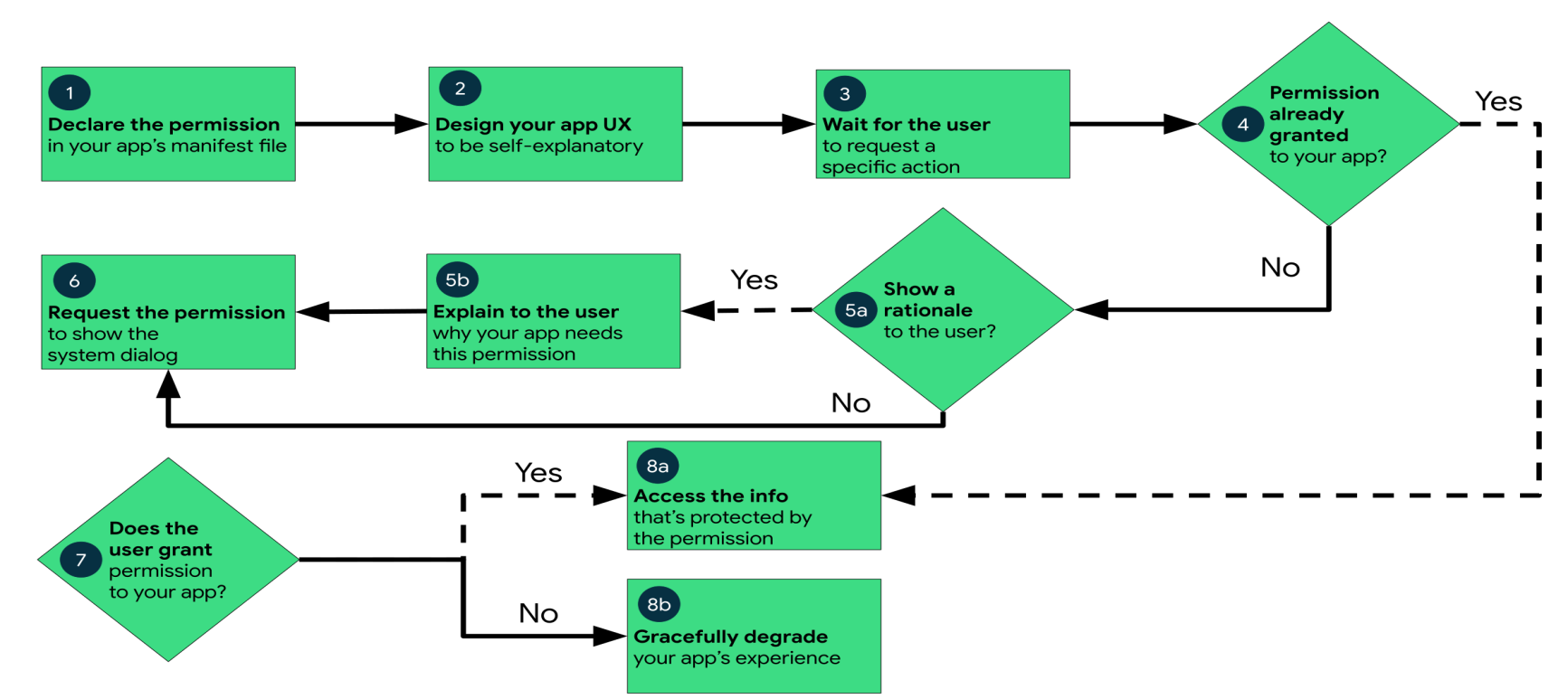
**==============================================================================================**

RUN TIME PERMISSION

Run time permission is a way of requesting for a specific feature from the user phone before you take some actions.

Let say you want to capture a camera on a click of a button.

You have to declare that camera permission in your manifest file first. Then whenever the button is clicked, you can check to see first if the user has granted your app the permission to capture a camera. If he has not granted the permission, you have to tell him why your app need such permission, and then request it again. But if the permission is granted, then execute the camera api to shot a camera/photo.



You don’t need to request over and over again for a permission. Once a user grant your app a specific permission, your app will have will have access to the permission by default even when the app is terminated and re-opened.

So your job is to check if your app is granted a permission first.

To check if the user has already granted your app a particular permission, pass that permission into the ContextCompat.checkSelfPermission(mainActivity.this, manifest\_permission) method.

This method returns either PERMISSION\_GRANTED or PERMISSION\_DENIED depending on whether your app has the permission.

So if you want to check if your app is already granted the permission to access the camera, it will be…

<uses-permission android:name="android.permission.CAMERA" /> // this goes in your manifest file

// this goes in your mainActivity

if(ContextCompat.*checkSelfPermission*(MainActivity.this,Manifest.permission.*CAMERA*)==PackageManager.*PERMISSION\_GRANTED*){

// take camera because you already have permission

}

else{

// tell the user why your app need such permission and request it with shouldShowRequestPermissionRationale(permission)

// then request the permission with requestPermissions(new String[]{Manifest.permission.*CAMERA*}, requestCode\_here);

// now the user will decide either to grant permission or not, and you can see the result in Activity overridden method onRequestPermissionsResult()

}

Example: asking runtime camera permission and take photo:

1. Inside your manifest file, add the camera permission feature to your app (this tells the user OS that your app may request for the camera permission anytime)

Inside Manifest

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.example.runtimepermission">

    <!-- add the camera permission feature -->

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

    <application

        android:allowBackup="true"

        android:icon="@mipmap/ic\_launcher"

        android:label="@string/app\_name"

        android:roundIcon="@mipmap/ic\_launcher\_round"

        android:supportsRtl="true"

        android:theme="@style/Theme.RunTimePermission">

        <activity android:name=".MainActivity">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

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

            </intent-filter>

        </activity>

    </application>

</manifest>

1. Then if you want to take a camera, check if permission is granted or not, then do what you want to do

Inside MainActivity.java

public class MainActivity extends AppCompatActivity {

 Button button;

ImageView image;

int permissionRequestCode = 200;

static final int REQUEST\_IMAGE\_CAPTURE = 1;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // Initialize properties

        button=(Button) findViewById(R.id.button);

        image=(ImageView) findViewById(R.id.image);

        // Onclick listener to take camera

        button.setOnClickListener(new View.OnClickListener() {

            @RequiresApi(api = Build.VERSION\_CODES.M)

            @Override

            public void onClick(View v) {

                // check if Camera permission is already granted

// if permission is granted it will return, PackageManager.PERMISSION\_GRANTED

// else it will return PackageManager.PERMISSION\_DENIED

                if(ContextCompat.checkSelfPermission(MainActivity.this,Manifest.permission.CAMERA)==PackageManager.PERMISSION\_GRANTED){

                  // show the camera and take photo

                     Toast.makeText(getApplicationContext(),"camera permission already granted", Toast.LENGTH\_SHORT).show();

                     // take photo

                    Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

                    try {

                        startActivityForResult(takePictureIntent, REQUEST\_IMAGE\_CAPTURE);

                        // if a camera is taken, the result will be sent to the Acvtivity overrriden method onActivityResult() below

                    } catch (ActivityNotFoundException e) {

                        // display error state to the user when error happen when opening camera app

                        Toast.makeText(getApplicationContext(),"error opening camera", Toast.LENGTH\_SHORT).show();

                    }

                }

                // if camera permission is not granted, tell him why you need it  and request it

                else {

                    // if the user has already disabled the camera feature or click on "don't show this request again for the camera permision.."

                    if (shouldShowRequestPermissionRationale(Manifest.permission.CAMERA)){

                        Toast.makeText(getApplicationContext(),"the camera perssion is required for this feature to work", Toast.LENGTH\_SHORT).show();

                    }

                    // request the permission (The result will be sent to the Activity onRequestPermissionsResult() method)

                    requestPermissions(new String[]{Manifest.permission.CAMERA}, permissionRequestCode);

                }

            }

        });

    }

    // When you request for a permission the result will be sent here onRequestPermissionsResult() (does either the permission is granted or not, is seen here)

    // if the permission is granted the requestCode will be equal to the and the

    // grantResult[] will contains int array positions of permissions granted

    // String[] permissions will contain string array list of permisions granted

    @Override

    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

        // check if the permission is granted based on the request code

        if(requestCode==permissionRequestCode){

            // check if the camera permission is granted

            if(grantResults[0]== PackageManager.PERMISSION\_GRANTED){

                Toast.makeText(getApplicationContext(),"user just granted the camera permission", Toast.LENGTH\_SHORT).show();

            }

            // if camera permisssion is denied

            else{

                Toast.makeText(this,"camera permission dednied for this feature",Toast.LENGTH\_SHORT).show();

            }

        }

        // if the request code did not match the permmision request code then, the use did not granted the permission

else {

            Toast.makeText(getApplicationContext(),"all permission rejected", Toast.LENGTH\_SHORT).show();

            super.onRequestPermissionsResult(requestCode, permissions, grantResults);

        }

    }

// This image is taken the result will be sent here to decide what you want to do with the image

    @Override

    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

        super.onActivityResult(requestCode, resultCode, data);

        // check if the user has taken some photto

        if (requestCode==REQUEST\_IMAGE\_CAPTURE && resultCode==RESULT\_OK && data!=null){

            Bundle extras = data.getExtras();

            Bitmap imageBitmap = (Bitmap) extras.get("data");

            image.setImageBitmap(imageBitmap);

        }

        else{

            Toast.makeText(getApplicationContext(),"no Photo taken", Toast.LENGTH\_SHORT).show();

        }

    }

}

**==============================================================================================**

Dialog Fragment

Dialog fragment is a more customize way to create a dialog alert, prompt or confirmation. You can customize it anyhow you want because just a fragment, but it extends DialogFragment class instead of the Fragment class.

The DialogFragment class provides all the methods for handling the dialog.

All you need is to create a custom fragment with it own layout. This fragment should extends DialogFragment class.

But remember, you don’t need to use fragment transaction to show this fragment dialog in an activity. The DialogFragment calss has a method called “show()” this method is use to display a dialog fragment in an activity. It takes 2 parameters which is, FragmentManager and a string tag.

Also note these two DialogFragment Class methods:

dismiss() ==> exit or dismiss the dialog.

setCancelable(false) ===> exit or do not exit the dialog when the out of the dialog is clicked.

Example:

1. Create a custom fragment that extends DialogFragment class with it own layout.

Inside mydialogfragment.xml

<androidx.appcompat.widget.LinearLayoutCompat

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical"

    android:layout\_gravity="center"

    android:minHeight="350dp"

    android:minWidth="320dp"

    android:padding="10dp"

    android:gravity="center">

    <TextView

        android:id="@+id/textView2"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_marginTop="100dp"

        android:text="Do You Want To Cancel?"

        android:textColor="@color/white" />

        <Button

            android:id="@+id/no"

            android:layout\_width="match\_parent"

            android:layout\_height="wrap\_content"

            android:layout\_margin="5px"

            android:layout\_gravity="left"

            android:layout\_marginEnd="8dp"

            android:layout\_marginRight="8dp"

            android:text="no" />

        <Button

            android:id="@+id/yes"

            android:layout\_gravity="right"

            android:layout\_width="wrap\_content"

            android:layout\_height="wrap\_content"

            android:layout\_margin="5px"

            android:layout\_marginStart="8dp"

            android:layout\_marginLeft="8dp"

            android:layout\_marginTop="24dp"

            android:text="yes" />

</androidx.appcompat.widget.LinearLayoutCompat>

1. Inflate the above layout into the fragment and handle click of buttons.

Inside MyDialogFragment .java

public class MyDialogFragment extends DialogFragment implements View.OnClickListener {

Button yes, no;

    @Nullable

    @Override

    // Inflate the fragment layout in this onCreateView() method and return it (the view will be sent to onViewCreated() method below)

    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.mydialogfragment,container);

        return view;

    }

    // the object "view" below represent the mydialogfragment.xml layout that was inflated in the above onCreateView() method

    // use it to create object of all the views that's in it and handle button clicks

    @Override

    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {

        super.onViewCreated(view, savedInstanceState);

//Create object of the buttons in the layout

        yes = (Button) view.findViewById(R.id.yes);

        no=(Button) view.findViewById(R.id.no);

// set onclick listener for the above buttons

        yes.setOnClickListener(this);

        no.setOnClickListener(this);

//  when the outside of the dialog is click, don't close the dialog

        setCancelable(false);

    }

    // Onclick listenere for the above "yes" and "no" buttons

    @Override

    public void onClick(View v) {

        if (v.getId()==R.id.no){

            // close the dialog fragment when the "no" button is click

            dismiss();

        }

        if (v.getId()==R.id.yes){

            // do business operations here if the yes button is click, am just show a toast.

            Toast.makeText(getActivity(),"yes clicked", Toast.LENGTH\_SHORT).show();

        }

    }

}

1. Then, inside MainActivity.java, Create an object of a button that will clicked to display the dialog fragment. And use the show() method to show the dialog fragment

Inside MainActivity.java

public class MainActivity extends AppCompatActivity {

Button cancelBtn;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_login);

        // Create an object of myDialogFragment and FragmentMager

        MyDialogFragment  myDialogFragment  = new MyDialogFragment ();

        FragmentManager fm = getSupportFragmentManager();

        // create an object of a button that will be clicked to show the dialog fragment

        cancelBtn = (Button) findViewById(R.id.cancel);

        //Set an onclickListener for the above button an show the dialog fragment when it clicked

        cancelBtn.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                // use the show() method to show the dialog fragement

                myDialogFragment.show(fm,"my dialog");

            }

        });

    }

}

**==============================================================================================**

Cardview

A cardview is just a view container for grouping other element in, it like html div.

You can include other TextView, EditText and Images or any view in it to customize how you want your view to be.

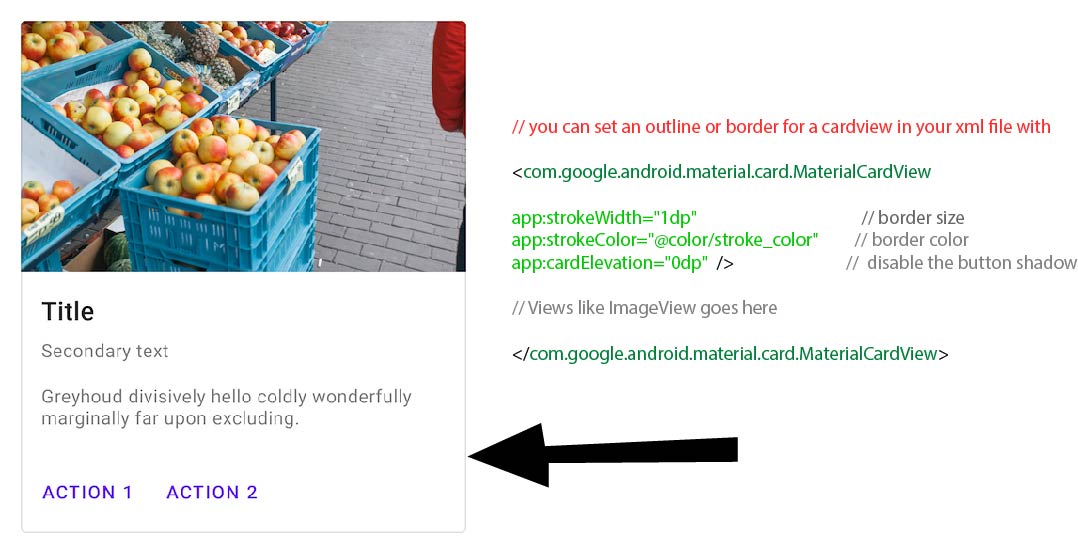
You can give it a border shadow using app:cardElevation="Number\_here\_dp”

First make sure the material dependency below is added to your build.gradle file

implementation 'com.google.android.material:material:1.3.0'

You can set an outline / border for your CardView in the xml layout as:

app:cardCornerRadius="10dp"



**==============================================================================================**

HOW TO START ACTIVITY WITH ANIMATION

The AppCompatActivity class has a method in it called A overridePendingTransition() which is called after startActivity() method is called.

Use it to show animations when starting an activity. It takes in two animations:

1. The incoming activity animation
2. The exiting activity animation

You need to create custom animation in your res/anim/animation\_name.xml and pass the name of the animations into the overridePendingTransition() method immediately after calling startActivity.

Example:

Imagine I have these two animations inside res/anim/…

<!--Inside res/anim/slide\_in\_right.xml -->

<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:fromXDelta="-100%p" android:toXDelta="0"

        android:duration="@android:integer/config\_mediumAnimTime"/>

</set>

<!--Inside res/anim/stay.xml -->

<translate xmlns:android="http://schemas.android.com/apk/res/android"

android:duration="@android:integer/config\_longAnimTime"

android:fromYDelta="0%p"

android:toYDelta="0%p" />

Then inside MainActivity.java, pass in the above two animation to the overridePendingTransition() method immediately after calling startActivity()

Inside MainActivity.java

public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_login);

        // create a buttom object here and set onClicklistener for it to call bellow method to start the activity

    }

// when this method is called it will start loginActivity with animation

    public void myMethodToActivity(View View){

        Intent myintent = new Intent(this, LoginActivity.class);

        startActivity(myintent);

        // this method will be called to triger animmation between the incomming and the exiting actitivty

        overridePendingTransition(R.anim.slide\_in\_right,R.anim.stay);

    }

}

**==============================================================================================**

VIEWPAGER2

viewPager2 is a way of providing slide between fragment. It has an adapter that takes in a list of fragment to slide between. When everything is set, the user can slide left and right of the phone screen to slide between the lists of fragment.

There is already a viewpager API in android, but Google has deprecated it, so they recommend using viewpager2.

It requires below dependency:

implementation "androidx.viewpager2:viewpager2:1.0.0"

viewPager2 is like any other view like EditText and TextView, you need to add it to your activity.xml like:

<!-- mainActivity.xml -->

<androidx.viewpager2.widget.ViewPager2

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:id="@+id/pager"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent" />

ViewPager2 uses "FragmentStateAdapter" class to supply the new pages or fragment to display.

So you need to Creates a class that extends the FragmentStateAdapter abstract class and implements the createFragment() method to supply instances of the list of fragment to show as new pages. You also need to implement the getItemCount() method,

which returns the number of fragments or pages the adapter will create.

This is how viewPager works:

You create a different list of fragment you want to slide between. Then create a class that extends "FragmentStateAdapter" and override it two methods:

createFragment() = use to create the list of fragment to slide between

getItemCount() = return the lenght of fragment, so that adater will know the total list of fragment to slide between.

Then inside your mainActivity.xml add the viewpager2 view with an id and create an object of it in mainActivity.java. Then set an adapter for the viewpager2. The viewpager2 has a method called registerOnPageChangeCallback() which takes in a a class that implement **OnPageChangeCallback** abstratct class. This OnPageChangeCallback abstract class has a method called onPageSelected() which is called whenever a new page is slide.

If you want to perfrom some action whenever a new page is slide, you can create an inner class that extends OnPageChangeCallback class and override the onPageSelected(). Then after register it as to the viewpager using the viewpager method registerOnPageChangeCallback()

Example:

1. Create two different fragments with their on xml layout. Let say i have created FirstFragment.java and SecondFragment.java with both having their own layout.

2. Create a custom adapter class that extends "FragmentStateAdapter" and implement it two methods.

Then pass the instance of the list of fragment to slide through in createFragment(). And return the length of the fragment in getItemCount()

// inside custome ViewPagerAdapter.java

// create a class that extends "FragmentStateAdapter" abstract class and implement it methods

public class ViewPagerAdapter extends FragmentStateAdapter {

    // this is a defualt constructor that will be created when you implements the methods don't worry about it

    public ViewPagerAdapter(@NonNull FragmentActivity fragmentActivity) {

        super(fragmentActivity);

    }

// Create an array of fragment to slide between

    Fragment[] fragments= {new FirstFragment(), new SecondFragment()};

    @NonNull

    @Override

    public Fragment createFragment(int position) {

        // then  return each fragment,

        return fragments[position];

    }

    @Override

    public int getItemCount() {

        // return the length of the fragment you have

        return fragments.length;

    }

}

1. Then inside mainActivity. Xml, add the viewpager2 view to your layout.

//inside mainActivity.xml

<androidx.constraintlayout.widget.ConstraintLayout

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent">

<!--Add the viewpager2 view in mainActivity.xml-->

    <androidx.viewpager2.widget.ViewPager2

        android:id="@+id/viewpager"

        android:layout\_width="match\_parent"

        android:layout\_height="match\_parent"

        tools:layout\_editor\_absoluteX="32dp"

        android:layout\_marginBottom="50dp"

        tools:layout\_editor\_absoluteY="355dp"/>

<EditText>

</EditText>

</androidx.constraintlayout.widget.ConstraintLayout>

1. Then inside mainActivity.java, create object of your viewpager2 and set adapter for it. Then do something when a new fragment is slide

//Inside mainActivity.java

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // create object of your viewpager2 object

           ViewPager2 viewpager = (ViewPager2) findViewById(R.id.viewpager);

        // create object of your above ViewPagerAdapter class and set is an adapter for your viewpager2

       ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(this);

        // set adapter for the viewpager

        viewpager.setAdapter(viewPagerAdapter);

// create object of below inner class and register "registerOnPageChangeCallback" method so that you can do something whenever the fragment changes

BB bb = new BB();

viewpager.registerOnPageChangeCallback(bb);

    }

 // Override the activity onBackPressed() method so that you can do something when the user press the back button

    @Override

    public void onBackPressed() {

        // if the first element in the fragment is what the user is on and he presses the back button,

        // kill the activity, this will call finnish()

        if (viewpager.getCurrentItem() == 0) {

            super.onBackPressed();

        }

        // if the user is on fragment number 4 and he presses the back button, send him back to fragment number 3 (it applies to all numbers)

        else {

            viewpager.setCurrentItem(viewpager.getCurrentItem() - 1);

        }

    }

// create an inner class that extends "ViewPager2.OnPageChangeCallback" and implements it onPageSelected() method

// to do something whenever the fragment changes. (Remember you need to register this class to the viewpager2 object above before)

    class BB extends ViewPager2.OnPageChangeCallback {

        // This overriden method "onPageSelected" will be called Whenever the fragment or page changes

      @Override

      public void onPageSelected(int position) {

          super.onPageSelected(position);

          // do something whenever a new fragement is slide (am just printing, the position of the current fragment)

           String post =  Integer.toString(position);

          Toast.makeText(MainActivity.this,post,Toast.LENGTH\_SHORT).show();

          }

      }

  }

}

/\*

Remember:

viewpager.getCurrentItem() == will give you the position of the current slide fragment

 viewpager.setCurrentItem(Number\_here) = will allow you to set which fragment to slide

 \*/

**==============================================================================================**

**RxAndroid**

RxAndroid is all about two key components: **Observable** and **Observer**. In addition to these, there are other things like **Schedulers**, **Operators** and **Subscription**.

*Observable:* Observable is a data stream that do some work and emits data. An Observable emits items or sends notifications to its observers by calling the observers’ methods.

*Observer:* Observer is the counter part of Observable. It receives the data emitted by Observable.

*Subscription:* The bonding between Observable and Observer is called as Subscription. There can be multiple Observers subscribed to a single Observable.

*Operator / Transformation:* Operators modifies the data emitted by Observable before an observer receives them.

*Schedulers:* Schedulers decides the thread on which Observable should emit the data and on which Observer should receives the data i.e background thread, main thread etc.,

Observer

An observer subscribes to an Observable.

Observer provides the below interface methods to know the state of Observable that this particular observer is subscribed to.

onSubscribe()

Method will be called when an Observer subscribes to Observable.

onNext(T)

An Observable calls this method whenever the Observable emits an item. This method takes as a parameter the item emitted by the Observable.

onError(T)

An Observable calls this method to indicate that it has failed to generate the expected data or has encountered some other error. It will not make further calls to onNext or onCompleted. The onError method takes as its parameter an indication of what caused the error.

onCompleted()

An Observable calls this method after it has called onNext for the final time, if it has not encountered any errors.

Schedulers

Schedulers basically decides the thread on which a particular code runs whether on background thread or main thread.

There are lot of Schedulers available, Schedulers.io() and AndroidSchedulers.mainThread() are extensively used in android programming. Below are the list of schedulers available and their brief introduction.

Schedulers.io() – This is used to perform non CPU-intensive operations like making network calls, reading disc / files, database operations etc., This maintains pool of threads and it run on a background thread.

AndroidSchedulers.mainThread() – This provides access to android Main Thread / UI Thread. Usually operations like updating UI, user interactions happens on this thread. We shouldn’t perform any intensive operations on this thread as it makes the app glitchy or ANR dialog can be thrown.

How to make an observer subscribe to observable

Make Observer subscribe to Observable so that it can start receiving the data. Here, you can notice two more methods, observeOn() and subscribeOn().

subscribeOn(Schedulers.io()): This tell the Observable to run the task on a background thread. Eg, making http request

observeOn(AndroidSchedulers.mainThread()): This tells the Observer to receive the data on android UI thread so that you can take any UI related action

Disposable

Disposable is used to dispose the subscription when an Observer no longer wants to listen to Observable to avoid memory leaks.

(it simply used to unsubscribe from Observables) it has a method called "dispose()" for doing this work

Let’s say you are making a long running network call and updating the UI. By the time network call completes its work, if the activity / fragment is already destroyed, and the Observer subscription is still alive, it tries to update the already destroyed activity. In this case it can throw a memory leak error. observers uses Disposables, to unsubscribe from observables to avoid this problem.

To get Started With RxAdnroid….

You need to add these two rxandroid and rxjava dependencies to your build.gradle file before you can work with rxandroid.

implementation 'io.reactivex.rxjava3:rxandroid:3.0.0'

implementation 'io.reactivex.rxjava3:rxjava:3.0.0'

Simple example.

public class MainActivity extends AppCompatActivity {

Disposable disposable;  // create a disposable object

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        mytext = (TextView) findViewById(R.id.mytext);

        mybutton = (Button) findViewById(R.id.mybutton);

 // create observable that emit/return some data to it subscribed observers

        // the colors represent the data this observable is emiting to it subscribed observers

        Observable<String> observable = Observable.just("red", "green", "yellow", "pink");

 // Create a new observer that will subscribe and fetch data from the above observable

        Observer<String> observer = new Observer() {

            // this will be called immediately the observer subscribe to an observable

            @Override

            public void onSubscribe(@NonNull Disposable d) {

                Log.d("onsubscribe", "onSubscribe");

   disposable = d; // assigned the desposable to this above "d" disposable object

            }

            /\* this will be called whenever the observable emit the data.

            "object o" represent the data the observable is sending to this observer

            here is where to update view items

             \*/

            @Override

            public void onNext(@NonNull Object o) {

                Log.d("TAG", "Name: " + o);

            }

           // this will be called if only the observable encounters an error

            @Override

            public void onError(@NonNull Throwable e) {

                Log.e("TAG","onError: " + e.getMessage());

            }

            // this will be called when the oberservable is done emiting it all data

            @Override

            public void onComplete() {

                Log.d("tag", "onComplete: All items emited");

            }

        };

 // Subscribe to the observer

        observable.subscribeOn(Schedulers.io()) // tell the observable to run the task on a background thread (maybe fetch data from api)

                .observeOn(AndroidSchedulers.mainThread())  // tell the observa to receive the data on a the mainthread

                .subscribe(observer);  // let the observer subscribe to the observable

    }

// Override Activity ondestoryed method

  @Override

  protected void onDestroy() {

      super.onDestroy();

      // unsubscribe from the Observable once the activity is destroyed

      disposable.dispose();

  }

}

/\* RESULT WILL BE:

onSubscribe: onSubscribe

Name: red

Name: green

Name: yellow

Name: pink

All items emited!

\*/

**==============================================================================================**

RecycleView

Recycleview is a widget for displaying scrollable list of vertical or horizontal items. It very good because it only loads the data the user scrolls instead of loading everything at once. This increase app performance a lot.

**Example:**

 // Inside article.xml

// 1. First create a custom layout file which represent each single list items to be show in the recycleview

 ?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:orientation="vertical"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent">

    <TextView

        android:id="@+id/articletitle"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_gravity="center\_horizontal"

        android:text="TextView"

        android:textAlignment="center"

        android:textColor="@color/design\_default\_color\_error"

        android:textAppearance="@style/TextAppearance.AppCompat.Medium"

        android:textSize="16sp" />

    <ImageView

        android:id="@+id/articleimage"

        android:layout\_width="match\_parent"

        android:layout\_gravity="center\_horizontal"

        android:layout\_height="wrap\_content"

    />

    <TextView

        android:id="@+id/articledesc"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_gravity="center\_horizontal"

        android:textAlignment="center"

        android:text="TextView" />

</LinearLayout>

 // Inside MainActivity.xml

// 2. Create a new Activity and add the recyview Widget to the MainActivity.xml file

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:orientation="vertical"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent">

    <androidx.recyclerview.widget.RecyclerView

        android:id="@+id/recycleviewId"

        android:layout\_width="409dp"

        android:layout\_height="294dp"

        android:layout\_marginStart="1dp"

        android:layout\_marginEnd="1dp"

        app:layout\_constraintEnd\_toEndOf="parent"

        app:layout\_constraintHorizontal\_bias="1.0"

        app:layout\_constraintStart\_toStartOf="parent"

        app:layout\_constraintTop\_toBottomOf="@+id/title" />

</LinearLayout>

// iniside ArticlesAdapter.java

// 3. Then Create a Custom class that extends RecyclerView.Adapter to be use as an adapter for the recycleview

// Then Create an Inner Class below <ArticlesView> and make it a Datatype to the extended RecyclerView.Adapter below

public class ArticlesAdapter extends RecyclerView.Adapter<ArticlesAdapter.ArticlesView> {

 // Create a property that will represent the data to be shown on your Recycle view widget

 public List<ArticleModel> articles;

 Context context;

 // Constructor for your RecyleView Adapter

 // Initialize the above <articles>

 // the idea is to allow Activities to Create object of this Adepter and pass the data to it here

 public  ArticlesAdapter(Context context, List<ArticleModel> articles){

     this.articles=articles;

     this.context = context;

 }

    // Here is where you inflate/put the custome xml in the mainActivity.xml file

    // So at the end, the become one view and returns it.

    @NonNull

    @Override

    public ArticlesView onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

     View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.article, parent,false);

        return new ArticlesView(view);

    }

    // This is where you assign values to the views in the RecycleView widget

    @Override

    public void onBindViewHolder(@NonNull ArticlesView articlesView, int position) {

    String artTi = articles.get(position).getTitle();

    String artDs = articles.get(position).getDescription();

    String imageUrl = articles.get(position).getUrlToImage();

    articlesView.setData(artTi, artDs, imageUrl);

    }

    // This method is what the recyviewView adapter will use

    // to know the number of items to show in the recyleview

    // This helps it to identify the last items in the recycleview too

    // All you need is to return the length or size of your items

    @Override

    public int getItemCount() {

        return articles.size();

    }

     // Create a custeom inner Class That extends RecyclerView.ViewHolder

    // Inside this class Is where We Define an object of view to be shown in the RecycleView

    class ArticlesView extends RecyclerView.ViewHolder{

        // Create object of your View Items

        TextView articleTitle, articleDesc;

        ImageView articleImage;

        // Create a Constructor and Initialze your views

        // <itemView> below is the RecycleView widget itself in the MainActivity.xml file

        public ArticlesView(@NonNull View itemView) {

            super(itemView);

            articleTitle = (TextView) itemView.findViewById(R.id.articletitle);

            articleDesc = (TextView) itemView.findViewById(R.id.articledesc);

            articleImage =(ImageView) itemView.findViewById(R.id.articleimage);

        }

        // Create a Public method to be used to assign values to your views

        // The values will be bind to it in the above implemented <onBindViewHolder> method

        public void setData(String artTitle, String artDesc, String imageResource){

            articleTitle.setText(artTitle);

            articleDesc.setText(artDesc);

            // Glide is just a library for loading http images, don't mind, it has nothing to do with RecycleView adapters

            Glide.with(context).load(imageResource).placeholder(R.drawable.ic\_launcher\_background).into(articleImage);

        }

    }

}

// 4. inside MainActivity.java

// Create an object of your recycleview widget in mainActivity.xml

// Then set a layout manager and the above adapter for the recyleview

public class MainActivity extends AppCompatActivity {

    // Create object of recycleview and layoutManager

    RecyclerView recyclerView;

    LinearLayoutManager layoutManager;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        recyclerView =(RecyclerView) findViewById(R.id.recycleviewId);

        layoutManager = new LinearLayoutManager(getApplicationContext());

        // set the orientation of your layout manager and set it to your recycleview

        layoutManager.setOrientation(LinearLayoutManager.VERTICAL);

        recyclerView.setLayoutManager(layoutManager);

       // Create a list of ArticlModel

       // "ArticlModel" is just a model class I have left it out

       List<ArticleModel> articles = new ArrayList<>();

       articles.add(new ArticleModel("title 1", "desc 1", "https://image.jpg"));

       articles.add(new ArticleModel("title 2", "desc 2", "https://image.jpg"));

    // Create object of the RecycleView Adapter above and the above <articles> to the parameter

     ArticlesAdapter articlesAdapter = new ArticlesAdapter(getApplicationContext(), articles);

      // Set the above adapter for your recycleview

       recyclerView.setAdapter(articlesAdapter);

    }

}

**==============================================================================================**

MVVM ACHITECTURE WITH RETROFIT AND RECYCLEVIEW

MVVM is the best architecture for developing mobile apps. In all your android apps, Please use what am about to show you.

Folders/Packages needed:

1. Adapters = This is where you should group different kind of adapter class. Eg. Recycleview adapter, listview adapter
2. Models = This is where to group list of model classes for your app
3. Retrofit = This is where to, create a class for your retrofit instance and a DAO interface for your API Endpoint
4. Services = This is where you have to group difference service classes
5. Utiles = This is where you have to group different constants data to use in your app. Eg. API URLS
6. ViewModels = This is the folder to group different viewModel classes
7. Views = This is where to group different list of activities

**Example:**

First add these dependencies to your gradle file

// Dependencies for rxjava, lifecycle, retrofit, glide and gson

dependencies {

    implementation 'io.reactivex.rxjava3:rxjava:3.1.3'

    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

    implementation 'com.google.code.gson:gson:2.8.5'

    implementation group: 'com.squareup.retrofit2', name: 'retrofit', version: '2.9.0'

    implementation group: 'com.squareup.retrofit2', name: 'converter-gson', version: '2.9.0'

    implementation 'com.github.bumptech.glide:glide:4.12.0'

}

//inside the package: Utils->ApiUrl.java

// this is where to define all your url address

public class ApiUrl {

    public static final String myApiKey ="9a284a31566545e3bd747b33f1cf4b3f";

    public static final   String baseUrl ="https://newsapi.org";

    public static final String articleUrl = "/v2/everything?q=tesla&from=2022-01-17&sortBy=publishedAt&apiKey="+myApiKey;

}

//inside the package: models->ArticleModel.java

// This is a model class for the Articles

public class ArticleModel {

    // @SerializedName("author") means use the key in the annotation and not

    // the below property <author> when doing encoding/decoding

    @SerializedName("author")

    // Allow Gson to encode/serialize & decode/deserialize this <firstName> property

    // If you avoid encode or decode, you can use:  @Expose(serialize = false, deserialize = false)

    @Expose

    private String author;

    @SerializedName("title")

    @Expose

    private String title;

    @SerializedName("description")

    @Expose

    private String description;

    @SerializedName("urlToImage")

    @Expose

    private String urlToImage;

    // Create a Contructor

    public ArticleModel(String author, String title, String description, String urlToImage) {

        this.author = author;

        this.title = title;

        this.description = description;

        this.urlToImage = urlToImage;

    }

    // getters & setters

    public String getAuthor() {

        return author;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getUrlToImage() {

        return urlToImage;

    }

    public void setUrlToImage(String urlToImage) {

        this.urlToImage = urlToImage;

    }

  // toString method goes here but i have cut it off

}

//inside the package: retrofit->RetrofitInstance.java

// Create A Custom Class with a method that returns instance of retrofit

public class RetrofitInstance {

    // Create a property to represent the instance of Retrofit

   private static Retrofit retrofit=null;

   // Create a Method that returns an Instance of Retrofit

   public static Retrofit getRetrofitInstance(){

       if (retrofit ==null){

           // You need to include the package  implementation group: 'com.squareup.retrofit2', name: 'converter-gson', version: '2.9.0'

           // package for GsonConverterFactory

           // it means use Gson with Retrofit to perform serialization & Deserialization

           retrofit = new Retrofit.Builder()

                   .baseUrl(ApiUrl.baseUrl)

                   .addConverterFactory(GsonConverterFactory.create()).build();

       }

       // return the Retrofit instance to the calling method

       return  retrofit;

   }

}

//inside the package: retrofit->ApiRequest.java

// this is the interface to define all your api Endpoint

// It is DAO (google DAO for meaning)

public interface ApiRequest {

    // Create an interface method to make a GET request

    // @GET("/api/articles") means = make a GET request to the url /api//api/articles together with the base class in retrofit instance

    @GET(ApiUrl.articleUrl)

    Call<HashMap<String,Object>> getArticles();

    // post request

    // the "@Body ArticleModel articleModel" means = you will post object of ArticleModel to the server

    @POST(ApiUrl.articleUrl)

    Call<HashMap<String,Object>> AddArticle(@Body ArticleModel articleModel);

}

//inside the package: service->ArticleService.java

public class ArticleService {

    //====PROPERTIES

    //  get the name of the class. so it will be: TAG = "ArticleService"

    private static final String TAG = ArticleService.class.getSimpleName();

    // Create a property to represent the Interface that contains all the retrofit api request/endpoint

    // Or the DAO

    private final ApiRequest apiRequest;

    //Create a new object of Gson for doing Encoding/Decoding of json data

    // Check my Java learning note, I have documentation of Gson in it, its the same

    Gson gson;

    //====CONSTRUCTORS

    public ArticleService(){

        gson = new Gson();

        // initialize the above <apiRequest> to the retrofit instance

        // All that you're doing here is, linking the retrofit instance to your DAO or Api Endpoint

        apiRequest = RetrofitInstance.getRetrofitInstance().create(ApiRequest.class);

    }

    //====METHODS

    // Service Method to fetch Articles From Server (remember the return type is LiveData)

    public LiveData<List<ArticleModel>> getArticles(){

        // Create a MutableLiveData where you will assign the data from the server to it

         MutableLiveData<List<ArticleModel>> all\_Articles = new MutableLiveData<>();

         // make the get request to fetch the articles

        apiRequest.getArticles().enqueue(new Callback<HashMap<String, Object>>() {

            // if request successful

            @Override

            public void onResponse(Call<HashMap<String, Object>> call, Response<HashMap<String, Object>> response) {

                List<ArticleModel> allArticles = new ArrayList<>();

                // Get the list of Articles Object

                Object serverData = response.body().get("articles");

                //Then convert it to a list

                List articles = (ArrayList) serverData;

                // Loop Through it

                // NOTE = the code in this for-loop block will be run on each single article

                for (Object bb: articles){

                    // convert this article to jsonString/string

                    String tojson = gson.toJson(bb);

                    // deserialize/decode it to <ArticleModel> class

                    ArticleModel articleModel = gson.fromJson(tojson, ArticleModel.class);

                    // add it to the list of <ArticleModel> to be returned

                    allArticles.add(articleModel);

                }

               // use the setValue() method because below <all\_Articles> is of type MutableLiveData

                all\_Articles.setValue(allArticles);

                System.out.println("All articles is:" + allArticles.toString());

            }

            // if request fail

            @Override

            public void onFailure(Call<HashMap<String, Object>> call, Throwable t) {

                System.out.println(t.getMessage());

            }

        });

       // return the list of articles tot he calling code

        // it of type MutableLiveData

        // (Note= MutableLiveData class extends LiveData, so you can return MutableLiveData object from a method of type LiveData)

      return all\_Articles;

  }

}

//inside the package: viewmodels->ArticleViewModel.java

// This is a ViewModel class

public class ArticleViewModel extends ViewModel {

    //====== PROPERTIES

    //Create object of the service Class

    private ArticleService articleService;

    // Var that represent all the Articles

    // Observe it getter below <getArticles> in other Activities to get all the List of Articles

    private LiveData<List<ArticleModel>> articles;

    //====== CONSTRUCTOR

    public ArticleViewModel() {

        // make sure to call the extended ViewModel super class Constructor.

        super();

        // Initialise the above <articleService> object of the Service class

        this.articleService = new ArticleService();

        // Get all the Articles from the server through the Service class Ands

        // assign the server response to the above <articles>

        articles = articleService.getArticles();

    }

    //====== GETTERS & SETTERS

    // Getter for the above <articles>

    public LiveData<List<ArticleModel>> getArticles() {

        return articles;

    }

    @Override

    protected void onCleared() {

        // clear the viewModel data when the app is exited

        super.onCleared();

    }

}

//inside the package: adapters->ArticlesAdapter.java

// this is an adapter class for the recycleview

public class ArticlesAdapter extends RecyclerView.Adapter<ArticlesAdapter.ArticlesView> {

 public List<ArticleModel> articles;

 Context context;

 public  ArticlesAdapter(Context context, List<ArticleModel> articles){

     this.articles=articles;

     this.context = context;

 }

    @NonNull

    @Override

    public ArticlesView onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

     View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.article, parent,false);

        return new ArticlesView(view);

    }

    @Override

    public void onBindViewHolder(@NonNull ArticlesView articlesView, int position) {

    String artTi = articles.get(position).getTitle();

    String artDs = articles.get(position).getDescription();

    String imageUrl = articles.get(position).getUrlToImage();

    articlesView.setData(artTi, artDs, imageUrl);

    }

    @Override

    public int getItemCount() {

        return articles.size();

    }

    class ArticlesView extends RecyclerView.ViewHolder{

        TextView articleTitle, articleDesc;

        ImageView articleImage;

        public ArticlesView(@NonNull View itemView) {

            super(itemView);

            articleTitle = (TextView) itemView.findViewById(R.id.articletitle);

            articleDesc = (TextView) itemView.findViewById(R.id.articledesc);

            articleImage =(ImageView) itemView.findViewById(R.id.articleimage);

        }

        public void setData(String artTitle, String artDesc, String imageResource){

            articleTitle.setText(artTitle);

            articleDesc.setText(artDesc);

            Glide.with(context).load(imageResource).placeholder(R.drawable.ic\_launcher\_background).into(articleImage);

        }

    }

}

//inside the package: views->MainActivity.java

public class MainActivity extends AppCompatActivity {

    // create object fo the articleViewModel class

    ArticleViewModel articleViewModel;

    RecyclerView recyclerView;

    LinearLayoutManager layoutManager;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        recyclerView =(RecyclerView) findViewById(R.id.recycleviewId);

        layoutManager = new LinearLayoutManager(getApplicationContext());

        layoutManager.setOrientation(LinearLayoutManager.VERTICAL);

        recyclerView.setLayoutManager(layoutManager);

        // initialize the Object of the ViewModel class with the android ViewModelProvider

        articleViewModel =new ViewModelProvider(this).get(ArticleViewModel.class);

        // Observe the getArticles() method to get all the list of Articles

        articleViewModel.getArticles().observe(this, data->{

            // check if the list of articles coming from the ArticleViewModel class is empty

            if(data == null){

                Log.e("MainActivity", "There's no Articles");

            }

            // else assign the data to your to a recycleview adapter

            else{

                // pass the list of articles to the recycleview Adapter

                ArticlesAdapter articlesAdapter = new ArticlesAdapter(getApplicationContext(), data);

                // then set the above <articlesAdapter> for your recycleview

                recyclerView.setAdapter(articlesAdapter);

            }

        });

    }

}

**==============================================================================================**

Room database

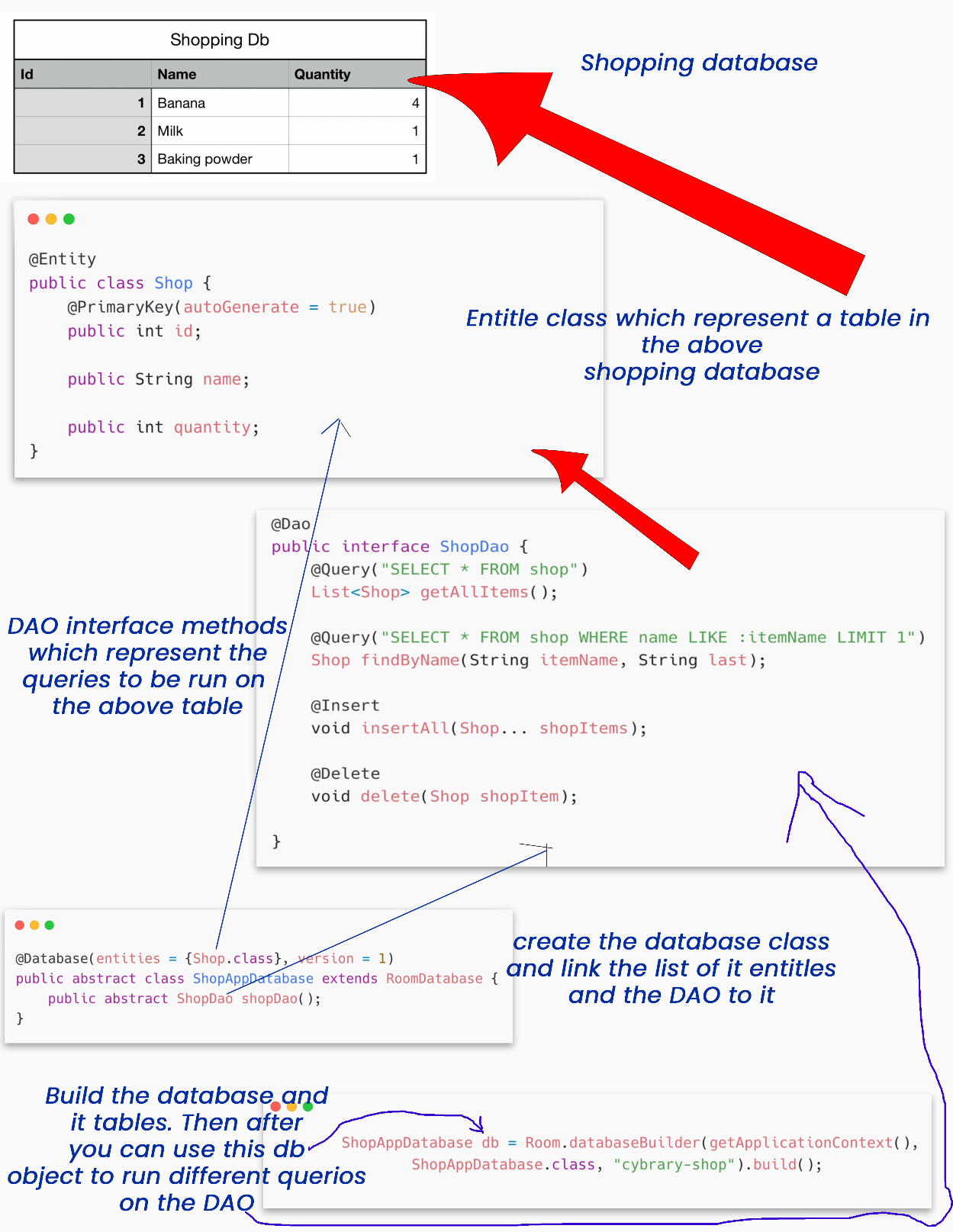
Room database is a way of saving structural large data locally on the user device. At first, we used to use SQLite but google has now come up with room, which makes it so easy to work with SQLite.

**Basic Concepts:**

**Database**: Main access point for the connection to your apps Data

**Entity:** Represent a table inside the database

**DAO:** Interface method used for accessing the database tables e.g for inserting and updating a table



**Example:**

First include this room dependency in your gladle file

*// def room\_version = "2.4.1" means define a room version as a variable  
// and use it, so you can see it being use as $room\_version below*

def room\_version = "2.4.1"  
implementation "androidx.room:room-runtime:$room\_version"  
annotationProcessor "androidx.room:room-compiler:$room\_version"

**==============================================================================================**

Exoplayer

Exoplayer is an advance api for playing media items like audio and video. It more advance than the default android mediaplayer and google itself recommend developers to use exoplayer.

**Note:** To use it, your project minimum sdk needs to be 20 and above.

Note: You don’t have to worry at all, they have an official website where you can follow to learn everything: https://exoplayer.dev/

First Add the dependency to your build.gradle file

//    Add the dependencies

implementation 'com.google.android.exoplayer:exoplayer:2.17.0'

Then inside your MainActivity.xml file, add the StylePrayerView for displaying the media item. This will be the player UI which comes with options to skip, pause, play, rewind, play previous, and to play next media item.

<com.google.android.exoplayer2.ui.StyledPlayerView

android:id="@+id/myexoplayerui"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

</com.google.android.exoplayer2.ui.StyledPlayerView>

**Note:** there is more opptions you can set for the StylePrayerView in the above .xml file. Kindly check the official exoplayer website to see all

**Media Item**

In ExoPlayer every piece of media (audio/video) is represented by a MediaItem. To play a piece of media you need to:

1. build a corresponding MediaItem,
2. add it to the player,
3. prepare the player,
4. and call play to start the playback:

So inside MainActivity.java, create object of the styledplayerview and use mediaItem to play a video/audio

public class MainActivity extends AppCompatActivity {

    // Create an object of Exoplayer

    private ExoPlayer player;

    //create object of the StyledPlayerView in the MainActivity.xml

   // It's a prebuilt ui controllers for displaying videos by Exoplayer

    // use StyledPlayerControlView when working with audio

    private StyledPlayerView playerView;

    // create object of track slector, it for deciding which media item to play

    private DefaultTrackSelector trackSelector;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        // Get an object of the playerView in the MainActivity.xml

        playerView = (StyledPlayerView) findViewById(R.id.myexoplayerui);

    }

    // do everthing in the onstart lifecycle

    @Override

    protected void onStart() {

        super.onStart();

         // initialize the tracSelector

        trackSelector = new DefaultTrackSelector(this);

        // Create an instance of ExoPlayer with the builder

        player = new ExoPlayer.Builder(this).setTrackSelector(trackSelector).build();

        // Keep the device screen on (if true, the device screen will never go off when playing any media)

        playerView.setKeepScreenOn(true);

        // Set the Exoplayer to use for this PlayerView

        playerView.setPlayer(player);

         // Build the media item.

        // This will use default Okhttp to fetch the video from the url

        // so make sure to add internet permission in your manifest file

        MediaItem mediaItem =  MediaItem.fromUri(Uri.parse("https://i.imgur.com/7bMqysJ.mp4"));

        // tell the player what media item you want to play

        player.setMediaItem(mediaItem);

         // Prepare the player, this will let the prayer adjust itself

         player.prepare();

        // start playing the media

         player.play();

        // NOTE: FOR AUTOMATIC PLAYLIST OF MEDIA ITEMS

       // ExoPlayer supports playlists directly, so it’s possible to prepare the player

       // with multiple media items to be played one after the other:

       // EXPLAMPLE

            // Build the media items.

            MediaItem firstItem = MediaItem.fromUri("https://i.imgur.com/7bMqysJ.mp");

            MediaItem secondItem = MediaItem.fromUri("https://youtube.com/7bMqysJ.mp");

            // Add the media items to be played. Remember am using addMediItem() not setMediaItem()

            player.addMediaItem(firstItem);

            player.addMediaItem(secondItem);

            // Prepare the player.

            player.prepare();

            // start playing the media

            player.play();

    }

      // When the Activity Is Destroyed

      @Override

      protected void onDestroy() {

          super.onDestroy();

          // Free the device memory by releasing the player

          playerView.setPlayer(null);

          player = null;

          player.release();

      }

}

**MEDIASOURCE**

IT RECOMMENDED TO USE A MEDIASOURCE INSTEAD OF MEDIAITMES ABOVE, FROM EXOPLAYER VERSION 2 UPWARDS. SO USE MEDIASOURCE INSTEAD OF MEDIAITEMS

**LIST OF MEDIASOURCE YOU CAN USE**

Am only showing few here, check https://exoplayer.dev/media-sources.html for all the media sources

1. **HlsMediaSource (For playing Network Media files)**

TO PLAY A REMOTE/HTTP MEDIA FILE THAT DOESN'T PLAY AUTOMATICALLY WHEN YOU PREVIEW THE URL IN A BROWSER (THAT'S WHEN IT PROMPTS YOU TO DOWNLOAD THE FILE In A BROWSER) USE: HlsMediaSource

**HLS means:** HTTP Live Streaming (it for displaying audio and video from remote server.)

   // do everthing in the onstart lifecycle

    @Override

    protected void onStart() {

        super.onStart();

         // initialize the tracSelector

        trackSelector = new DefaultTrackSelector(this);

        // Create an instance of ExoPlayer with the builder

        player = new ExoPlayer.Builder(this).setTrackSelector(trackSelector).build();

        // Keep the device screen on (if true, the device screen will never go off when playing any media)

        playerView.setKeepScreenOn(true);

        // Set the Exoplayer to use for this PlayerView

        playerView.setPlayer(player);

         // first create am MediaItem that will be passed to a media Source with the Url of the media file

         MediaItem mediaItem = new MediaItem.Builder()

         //if your url does not end with m3u8, you will get an error. to fix it check online for help

                 .setUri(Uri.parse("https://5b44cf20b0388.streamlock.net:8443/vod/smil:bbb.smil/playlist.m3u8"))

                 // Use a MIME type if the uri extension is different from .m3u8

                 .setMimeType(MimeTypes.APPLICATION\_M3U8).build();

         // Create a data source factory.

         DataSource.Factory dataSourceFactory = new DefaultHttpDataSource.Factory();

         // Create a HLS media source pointing to a media/playlist uri

         HlsMediaSource hlsMediaSource = new HlsMediaSource.Factory(dataSourceFactory).createMediaSource(mediaItem);

         // set the media source for the player

         player.setMediaSource(hlsMediaSource);

          // Prepare the player.

         player.prepare();

         // Start the playback.

         player.play();

    }

1. **ProgressiveMediaSource (For playing Network Media files)**

TO PLAY NETWORK MEDIA FILES THAT PLAY AUTOMATICALLY WHEN YOU PREVIEW THE URL IN A BROWSER

    // do everthing in the onstart lifecycle

    @Override

    protected void onStart() {

        super.onStart();

         // initialize the tracSelector

        trackSelector = new DefaultTrackSelector(this);

        // Create an instance of ExoPlayer with the builder

        player = new ExoPlayer.Builder(this).setTrackSelector(trackSelector).build();

        // Keep the device screen on (if true, the device screen will never go off when playing any media)

        playerView.setKeepScreenOn(true);

        // Set the Exoplayer to use for this PlayerView

        playerView.setPlayer(player);

        //FIRST Create a DefaultHttpDataSource data source factory

        DataSource.Factory dataSourceFactory = new DefaultHttpDataSource.Factory();

        // Create a ProgressiveMediaSourceFactory to stream the media from the remote file

        MediaSource mediaSource = new ProgressiveMediaSource.Factory(dataSourceFactory)

                .createMediaSource(MediaItem.fromUri(Uri.parse("https://i.imgur.com/7bMqysJ.mp4")));

        // Set the media-source item to be played for the player

        player.setMediaSource(mediaSource);

        // Prepare the player.

        player.prepare();

        // Start the playback.

        player.play();

    }

**Listeners**

You can add listeners to the player and perform some operation when certain things happens to the player like: when it playing media, when it paused playing, when volume increase.

    // do everthing in the onstart lifecycle

    @Override

    protected void onStart() {

        super.onStart();

         // initialize the tracSelector

        trackSelector = new DefaultTrackSelector(this);

        // Create an instance of ExoPlayer with the builder

        player = new ExoPlayer.Builder(this).setTrackSelector(trackSelector).build();

        // Keep the device screen on (if true, the device screen will never go off when playing any media)

        playerView.setKeepScreenOn(true);

        // Set the Exoplayer to use for this PlayerView

        playerView.setPlayer(player);

        //FIRST Create a DefaultHttpDataSource data source factory

        DataSource.Factory dataSourceFactory = new DefaultHttpDataSource.Factory();

        // Create a ProgressiveMediaSourceFactory to stream the media from the remote file

        MediaSource mediaSource = new ProgressiveMediaSource.Factory(dataSourceFactory)

                .createMediaSource(MediaItem.fromUri(Uri.parse("https://i.imgur.com/7bMqysJ.mp4")));

        // Set the media-source item to be played for the player

        player.setMediaSource(mediaSource);

        // Prepare the player.

        player.prepare();

        // Start the playback.

        player.play();

                // Add Listener to listen to various event like: if video is paused or not

        // You can add this same below listener to any of the above explained mediaItem or list of mediaSources

        // and it will work perfectly

        player.addListener(

                new Player.Listener() {

                    // Method to check if video is playing or not

                    @Override

                    public void onIsPlayingChanged(boolean isPlaying) {

                        if (isPlaying) {

                            Log.e("onIsPlayingChanged", "Is Playing ");

                        } else {

                            Log.e("onIsPlayingChanged", "Is NOt Playing ");

                        }

                    }

                    // When error happens

                    // The following example shows how to detect when a playback has failed due to an HTTP networking issue:

                    @Override

                    public void onPlayerError(PlaybackException error) {

                        Log.e("onPlayerError", error.getMessage());

                    }

                    @Override

                    public void onVolumeChanged(float volume) {

                        Log.e("onVolumeChanged", String.valueOf(volume));

                    }

                    // wHEN THE PLAYER CHANGES TO A NEW MEDIA ITEM.

                    //  The reason indicates whether this was an automatic transition, a seek (for example after calling player.next()),

                    //  a repetition of the same item, or caused by a playlist change (e.g., if the currently playing item is removed).

                    @Override

                    public void onMediaItemTransition(@Nullable MediaItem mediaItem, int reason) {

                    }

                    // When device volume changed & to check if device is muted

                    @Override

                    public void onDeviceVolumeChanged(int volume, boolean muted) {

                        if (muted) {

                            Log.e("onDeviceVolumeChanged", "Device is muted, you will not hear any sound ");

                        } else {

                            Log.e("onDeviceVolumeChanged", "device not muted");

                        }

                        Log.e("onDeviceVolumeChanged", String.valueOf(volume));

                    }

                    /\*

                     method to check the status of the fetching of the video from a remote server

                     DIFFERENT TYPES OF STATES THE PLAYER CAN BE IN:

                    Player.STATE\_IDLE: This is the initial state, the state when the player is stopped, and when playback failed.

                        The player will hold only limited resources in this state.

                    Player.STATE\_BUFFERING: The player is not able to immediately play from its current position.

                        This mostly happens because more data needs to be loaded.

                    Player.STATE\_READY: The player is able to immediately play from its current position.

                    Player.STATE\_ENDED: The player finished playing all media.

                     \*/

                    @Override

                    public void onPlaybackStateChanged(int playbackState) {

                        // IF the player is fetching the video from the remote server

                        if (playbackState == player.STATE\_BUFFERING) {

                            Log.e("onPlayerStateChanged", "buffering");

                        }

                        // If player is done fetching video from the remote server

                        else if (playbackState == player.STATE\_READY) {

                            Log.e("onPlayerStateChanged", "Done buffering");

                        }

                    }

                }

        );

        // Play the video immediately when it done buffering/fetching from the http Network

        player.setPlayWhenReady(true);

    }

**==============================================================================================**

Android Tips:

If You want To know How to use Android or Any UI Element, Like NavigationBottom And Anyone, simply go to <https://material.io/components> And Look for the Android Component, There is Web Component Too there.

Also any class or package you don’t understand simply go to <https://developer.android.com> to learn it yourself, it really easy because most youtubers just teaches what they understand.

Calling finish() When starting an activity in an intent. Will not put that activity in the backstack, that’s the activity will be destroyed and when the user press on the back button, he won’t find it.

<https://www.vectorizer.io/> === Very nice site for converting images to vectors but it is paid