Android ?

Android is a platform for Mobile , Tablet , TV, Wearable devices, desktop and AutoMobiles, which consists

- Operating System
- middleware
- key applications

Why Android ?

- Android is Open Source (the complete source will be visible to the user, user can see the code and modify the code according to their requirement).

- Application framework provides infrastructures to the application developer, it is providing infrastructures as ready made libraries (e.g.: LocationManager , SensorManager , WifiManager ……).

- DVM is optimised for mobile devices , it is a customised JVM to work on low power , memory and RAM.

- SQLite Database is used to maintain structured data in Android.
- supports GPS and media formats.
- OpenGLES native library is used to display graphics in Android.
- Android Studio provides rich development environment.
- Android is a product from OHA [Open Handset Alliance] which is lead by Google.
- Android plays a keyrole in IOT [Internet of Things : Machine - Machine communication is called as IOT].

Android Components :

- Activity
- Service
- Broadcast Receiver
- Content Provider

Activity :

- a single screen in the application with UI components user will interact with the device through Activity.

Service :

- a long running background process with out any user interaction.

Broadcast Receiver :

- Broadcast receivers are registered for System announcements.

e.g. headset plugin, power connected/disconnected ,
screen on/off, making/receiving call, sending/receiving SMS

Content Provider :

- ContentProvider is used to share the data between multiple applications. [In Android one of the security feature is we can't access the other applications data into our application directly
but if the application is providing Content Provider then we can access the data into our application, in Android following builtin applications are providing Content Provider.

contacts , call log, media, settings , messages , calendar]

AndroidManifest.xml :

- AndroidManifest.xml provides complete description (activities , services , broadcast receivers , app icon , initial screen , permission, features ..) about the application, Android platform before installing the application and before start the application it reads Manifest.xml.

R.java :

- R is termed as resource.
- R.java is used to provide communication between resource folder and Activity(java file).
- for every resource(file) in res folder it will create a static integer field in R.java, with the help of integer field only we can access the resource into java.

e.g. : R.drawable.sample

R.raw.test

- it is a auto generated file by Android Studio.

.apk :

- apk is termed as Application Package Kit, it is a installation file in Android platform.
- apk consist
  - .de
  - res
  - assets
  - lib
  - Manifest.xml

XML :

- XML is termed as Extensible Markup Langauge.

(enclosing the data with in tags is called as Markup Language).

- XML is called as interoperable format (platform independent , language independent).
- because of interoperability XML is used to transfer the data between multiple applications.
- XML is used as textual database.
- XML is used as deployment descriptor.

(java : web.xml , Android : Manifest.xml , iOS : info-plist.xml, .Net : web-config.xml …..)

- XML is used to create user interface

(HTML is a child of XML).

sample xml :

```
<employees>
    <employee id="123">
        <name>Mahesh</name>
        <desig>TechLead</desig>
        <dept>Mobility</dept>
    </employee>
</employees>
```

- every XML file must contain only one root element(tag).
- every XML element must be properly nested.
- XML element shouldn't contain space, shouldn't start with number and special characters.
- use the following characters to represent XML entities.

<	<
>	>
&	&
‘	'
“	"

- if any XML follows above rules then it called as well-formed XML, XML well-formness we can check in browser.

-GUI LAYOUTS:-

- UIGroup is used to specify how to arrange the UI components, following are the major UI groups.

- LinearLayout
- Table Layout
- Grid Layout
- Relative Layout
- Contraint Layout
- Frame Layout
- Coordinator Layout

LinearLayout :

- LinearLayout is one of the UI group in Android which is used to present the list of UI components in a vertical or horizontal format one after another.

syntax :

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical | horizontal">
// UI components ….
</LinearLayout>
```

for every UI component and UI group we have to specify width and height, following are the possible parameters to specify width and height.

- match_parent
- fill_parent
- wrap_content
- pixel [px]

- density pixel [dp]
- scaled pixel [sp]

Steps to create an Activity [java] :

step1: create a class as a child of android.app.Activity class [?].

Activity Life Cycle :

- Activity is having 4 states
  - Doesn't Exist
  - Foreground
  - Pause
  - Background
- following are the major methods in Activity class.
  - onCreate( )
  - onStart( )
  - onPause( )
  - onStop( )
  - onResume( )
  - onRestart( )
  - onDestroy( )
- to maintain Activity life cycle the class should be a child android.app.Activity class.

step 2 :

same like main( ) method in C,C++ , Java in android Activity onCreate( ) method will be invoked first so provide the implementation for onCreate( ) method.

step3 :

we designed the UI in xml we have to set the xml file to java, use the following method to set the xml file to java.

setContentView(R.layout.file_name);

- Bundle class is one of the input parameter for onCreate( ) method, Bundle is used to get the statement of an Activity and it is used to maintain(storage) the with in the application.

syntax :

```
protected void onCreate( Bundle savedInstanceState)
    {
        super.onCreate( );
        ……..   // logic to execute when Activity is started
    }
```

- savedInstanceState is a reference of Bundle class.

```
public class MainActivity extends android.app.Activity
{
   @Override
   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);
   }
}
```

- Button is having a click event, we can configure the click event in 2 ways.

- XML
- Java

XML :

- to configure the click event using XML, configure the following attribute to the UI component.

android:onClick="method_name"

- if we click the button it will invoke the specified method in

java, if the method is not available it will throw a MethodNotFoundException.

e.g.: XML :

android:onClick="getText"

Java :

```
public void getText(View v) {

// logic to execute when button is clicked..
                                                                    }
```

- to get the UI component from XML to java we have to configure an id for the UI component , use the following attribute to configure id.

android:id="@+id/id_name"

- use the following Activity (java file) to get the component from XML to java.

findViewById(R.id.id_name);

Configure click event using java :

- to configure the click event using java, get the UI component from XML to Java.

```
Button  b = findViewById(R.id.id_name);

b.setOnClickListener(new OnClickListener( ) {
                        public void onClick(View v) {
                                                // logic to execute when button is clicked
```

```
        }
    });
```

Intent :

- Intent is used to provide a communication between Activity - Activity , Activity - Service and Activity - Broadcast Receiver.

- with respect to Activity - Activity communication there are 2 types of Intents.

  - Implicit Intent
  - Explicit Intent

Implicit Intent :

- Implicit Intent is used to call builtin Activities.

syntax :

```
Intent i=new Intent( );
i.setAction(Intent.ACTION_NAME);
startActivity(i);
```

Explicit Intent :

- Explicit Intent is used to call user defined Activity.

syntax :

```
Intent i=new Intent(context, ActivityName.class);
startActivity(i);
```

- by using explicit Intent we can invoke other application activities from our application.

syntax :

Intent i = getPackageManager( ).getLaunchIntent

ForPackage(“package_name”);
startActivity(i);

- following are the major methods in Intent class.

setAction( ) setData( ) setType( )
putExtra( ) getExtra( )

(write the following code inside onCreate( ) method,
after setContentView( ) statement)

```
Button b1 = findViewById(R.id.b1);  // assign a value
b1.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      EditText et1 = findViewById(R.id.et1);
      Intent i = new Intent( );
      i.setAction(Intent.ACTION_DIAL);
      i.setData(Uri.parse("tel:"+et1.getText().toString()));
      startActivity(i);
   }
});
```

```
i.setAction(Intent.ACTION_VIEW);
i.setData(Uri.parse("http://"+et1.getText().toString()));
```

[Write the following code inside onCreate( ) method of MainActivity class]

```
b2 = findViewById(R.id.b2);
b2.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      EditText et2 = findViewById(R.id.et2);
      Intent i=new Intent(MainActivity.this,
                                          WelcomeActivity.class);
```

```
        i.putExtra("myname",et2.getText().toString());
        startActivity(i);
    }
});
```

[write the following code inside onCreate( ) method of WelcomeActivity.java file]

```
TextView tv1 = findViewById(R.id.tv1);
String name =   getIntent().getStringExtra("myname");
tv1.setText("Welcome 2 NIT : "+name);
```

Note :

- every user defined Activity must be configured in Manifest.xml with the following tag inside <application> tag.

<activity android:name="package_name.ActivityName" />

- which Activity you want to display first when the application is stated , for that Activity configure the following tag in Manifest.xml.

e.g. :

```
<activity android:name=".MainActivity">
  <intent-filter>
  <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />

  </intent-filter>
</activity>
```

AutoCompleteTextView :

- AutoCompleteTextView is one of the UI component in Android it is a child of EditText component, it is used to provide an auto completion support to the user.

e.g. :

xml: <AutoCompleteTextView

android:id="@+id/actv"

…… />

java :

AutoCompleteTextView actv =
findViewById(R.id.actv);

to provide an auto completion support we have to configure the values, in Android there are 2 approaches to configure the values.

- XML
- JAVA

XML :

- configure the values in the following xml file.
(res >> values >> strings.xml)

```
<string-array name="array_name">
    <item> value1 </item>
    <item> value2 </item>
    ……..
</string-array>
```

- use the following method in java to get the XML configured values into Activity.

String[ ] values = getResources( ).getStringArray(R.array.

array_name);

JAVA :

```
String[ ] values = new String[ ] {value1,value2,…. };
            (or)
ArrayList<String> list = new ArrayList<String>( );
        list.add(value1);
        list.add(value2);
 ……..
```

- to present the data we have to configure an Adapter, in Android there are 3 types of adapters.

- ArrayAdapter
- CustomAdapter
- CursorAdapter

ArrayAdapter :

- ArrayAdapter is used to present String type of data on screen.

syntax :

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>
                        (context ,  android.R.layout.xml_file, values);
actv.setAdapter(adapter);
actv.setThreshold(int);
```

- to achieve Internationalization (multi language support) we shouldn't hardcode the value, configure the value in Strings.xml with the following tag.

ex :

```
<string name="ecn"> Enter Country </string>
```

- use the following attribute in the xml to access the value.

```
android:text="@string/ecn"
```

- use the following method in to access the string value.

String value = getResources( ). getString(R.string.ecn);

Spinner :

- Spinner is one of the UI component in Android which is used to present the list of configured values in a drop down menu.

syntax :

xml : <Spinner
android:id="@+id/sp1"
…… />

java :

Spinner sp1 = findViewById(R.id.sp1);

same ACTV to present the values we have to configure the values in XML or Java.

- if the values are configured in XML, use the following attribute to present the values.

android:entries="@array/array_name"

- if the values are configured in java, set the adapter to present the values.

- Spinner is having Item Selected event , to get this event configure the following listener.

```
sp.setOnItemSelectedListener(
new OnItemSelectedListener(  ) {
public void onItemSelected( ) {  }
public void onNothingSelected( ) { }
                                    });
```

Toast :

- Toast is one of the notification method in Android which is used to display text on the screen for few seconds.

syntax :

```
Toast.makeText(context, message , duration).show( );
```

ListView :

- ListView is one of the UI component in Android which is used to present the list of configured values.

xml :

```
<ListView

android:id="@+id/lview"

…….. />
```

java :

```
ListView lview =

findViewById(R.id.lview);
```

- same like ACTV & Spinner to present the values we have to configure the values in XML / Java, if the values are configured in XML use android:entries="@array/array_name"
attribute to set the values , if the values are configured in java create an adapter to present the values.

```
lview = findViewById(R.id.lview);
String path = "/storage/emulated/0/";
File f = new File(path);
String[ ] files =  f.list();
```

```
 ArrayAdapter<String> adapter =
       new ArrayAdapter<String>(
       this,

   android.R.layout.simple_list_item_single_choice,
            files);
lview.setAdapter(adapter);
```

- to access the device storage information , we have to get the permission from the user, use the following tag in Manifest.xml to request the permission.

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

How to provid
e runtime permission :

- get the permission status.

```
int status =    ContextCompat.checkSelfPermission(this,
      Manifest.permission.READ_EXTERNAL_STORAGE);
```

- check the permission is granted or not, if the permission is granted we can access the storage info, if the permission is not granted request the user to grant a permission.

```
   if(status == PackageManager.PERMISSION_GRANTED)
    {
     readFiles();
    }else{
        ActivityCompat.requestPermissions(this,
         new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
         123);
     }
```

- override the following method in Activity class to recognise user is accepted the permission / denied permission.

```
public void onRequestPermissionsResult(int requestCode,

String[] permissions,  int[] grantResults) {
   super.onRequestPermissionsResult(requestCode,

permissions, grantResults);
   if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
   {
      readFiles();
   }
}
```

- by using ArrayAdapter we can present only String type of data,
  to present our own UI on individual item we have to use CustomAdapter.

- create a class as a child of android.widget.BaseAdapter.
- it is an abstract class having following abstract methods
  - getCount( )
  - getItem( )
  - getItemId( )
  - getView( )
- from Activity class use the following method to set the custom adapter.

```
ui_comp . setAdapter(new CustomAdapterClass( ));
```

- LayoutInflater : LayoutInflater class is used to convert UI XML file into View object.

```
LayoutInflater  inflater = LayoutInflater.from(activity_object);
```

View v = inflater.inflate(R.layout.xml_file ,

view_group(mostly null));

WebView :

- WebView is one of the UI component in Android which is used to display webpages in Android application.

xml :

```
<WebView

    android:id="@+id/wview"
    ……….
/>
```

java :

```
WebView wview =
findViewById(R.id.wview);
wview.loadUrl("http://url_address");
```

method is used to display a specific webpage on WebView component.

- by using WebView we can perform the following operations.

1. call browser
2. integrate the browser in our application
3. display .html files
4. we can provide communication between HTML UI and Android Activity.

- to access internet from application , add the following permission in Manifest.xml.

<uses-permission android:name="android.permission.

INTERNET”/>

- set the following method to web view component to display web page on web view itself (instead of navigating to browser).

```
wview.setWebViewClient( new WebViewClient( ));
```

- following are the major methods in WebViewClient class.

- onPageStarted( )
- shouldOverrideUrlLoading( )
- onPageFinished( )

- by default WebView will not enable Java Script and zoom controls, set the following methods to WebView component to enable java script and zoom controls.

```
wview.getSettings( ).setJavaScriptEnable(true);
wview.getSettings( ).setBuiltinZoomControls(true);
```

- to display .html file put .html file in assets folder, use the following code to display .html file on WebView component.

```
wview.loadUrl(“file:///android_asset/file_name.html”);
```

- JavaScript interface is used to provide communication between HTML UI and Android Activity.
- use the following method to configure java script interface.

```
wview.addJavaScriptInterface(class_object,
“interface_name” );
```

- by using second parameter (interface_name) we can communicate with the specified class methods from JavaScript.

- which method u want to call from JavaScript for that method add the following annotation.

@JavaScriptInterface

fragment :

- fragment is one of the UI component in Android.
(or)
- fragment is called as subtype of an Activity.

xml :

<fragment

android:id="@+id/fragment1"

…… />

- fragment is used for code reusability and easy to maintain the project.

- for every fragment we have to create an XML file and java file.

- fragment life-cycle is depends on Activity life cycle.

Steps to create fragment java file :

- create a class as a child of android.app.Fragment class.

- same like Activity onCreate( ) method in Fragment onCreateView( ) method will be invoke first.

eg :

```
class MyFragment extends Fragment
{
```

```
public View onCreateView(LayoutInflater inflater,

ViewGroup vgroup, Bundle b) {
View v = inflater.inflate(R.layout.xml_file,vgroup,false);
return v;
}
}
```

- use the following code in Activity to display fragment.

```
FragmentManager fManager = getFragmentManager( );
FragmentTransaction tx = fManager.beginTransaction( );
tx.add/replace/remove(fragment_id,new FragmentName());
tx.commit( );
```

- if the UI component is available on fragment XML we can't configure the events through XML.
- use the View object to get the UI component from fragment xml file (inside onCreateView( ) method before return statement).

eg:

```
Button b = v.findViewById(R.id.xxxx);
```

- fragment component is introduced from Android 3.0.

Storage Methods :

- SharedPreferences
- SQLite Database
- Files [XML / JSON]

SharedPreferences :

- SharedPreferences is used to used to maintain the data in a key, value pair.
- SharedPreferences interface is used to maintain the data in a Shared Preferences, getSharedPreferences( ) method is used to get the object of SharedPreferences.

```
SharedPreferences spf = getSharedPreferences("spf_name",

Context.MODE_NAME);
```

- if the SPF is not available with the specified name then it will create a new SPF, if the SPF is already available with the specified name then it will get the data from existing SPF.

- following are the possible parameters to specify mode.

Context.MODE_PRIVATE
Context.MODE_WORLD_READABLE
Context.MODE_WORLD_WRITABLE

- by using above object (spf) we can create SPF and we can get the data from SPF, to write the data into SPF we required an object for SharedPreferences.Editor class.

```
SharedPreferences.Editor spe = spf.edit( );
spe.putX("key",value);      // X  - data type
spe.commit( );
```

- use the following code to read the data from SPF.

```
spf.getX("key",default_value);     // X - data type
```

if the value is available with the specified key, it will return the Actual value otherwise it will return the default value.

- internally SPF will maintain the data in a XML file , we can explore the XML file using ADM [Android Device Monitor].

[Tools >> Android >> Device Monitor >> select device >> data >> data >> package name >> preferences >> spf_name.xml]

Note :

by using SPF we can maintain huge amount of data, but for every value we have to specify a unique key its difficult to assign and

remember the unique key thats why SPF is recommended to maintain limited data like (pattern lock key , application pin number , keep me signin credentials , game lovel high scores …) to maintain huge amount of data Android is recommend to use SQLite database.

SQLite Database :

- SQLiteDatabase is used to maintain structured data in Android.
- SQLiteDatabase class is used to manage SQLite database, openOrCreateDatabase( ) method is used to get the Object of SQLite database.

```
SQLiteDatabase dBase = openOrCreateDatabase(
                        "db_name", mode , cursor_factory (optional));
```

- SQLiteDatabase is providing builtin methods to perform CRUD operations.

  - insert( )
  - query( ) // read
  - update( )
  - delete( )

- apart from the builtin methods we can execute the native SQL statements by using the following methods.

  - execSQL(sql_query) // C I U D
  - rawQuery( ) // R

- Steps to Achieve MVP design Pattern :

  - create following packages as a childs of root package.

view

presenter model beans

  - move Activity/fragment files into View package.

- create Bean classes under beans package (for every table we have to create a separate Bean class).

- create interface under View package, create following abstract methods in interface.

eg :

```
interface  DBResult
{
        void insertResult(String s);
        void readResult(ArrayList<Employee> list);
        void updateResult(String s);
        void deleteResult(String s);
}
```

- create interface under presenter package, this interface should contain abstract methods for which operations you want to perform on database.

```
interface DBOperationsAPI
{
    void  insert(Employee e);
    void  read(Employee e);
    void update(Employee e);
    void delete(Employee e);
}
```

- create a class under model package , the class has to provide the implementation for presenter interface, create a constructor which is taking View interface as a input.

eg :

```
class DBOperationsImpl  implements DBOperationsAPI
{
        DBResult dr;
        DBOperationsImpl(DBResult dr){
                this.dr = dr ;
```

```
                    }
                    insert(Employee e) {   ….. }
                    read(Employee e) {   ….. }
                    update(Employee e) {   ….. }
                    delete(Employee e) {   ….. }
            }
```

- The Activity class should provide the implementation for View interface.

eg :

```
                    class MainActivity implements DBResult
  {
                        onCreate( ){   };
                        insertResult(String s) {….};
                        readResult(ArrayList<Employee> list){ ….. };
                        updateResult(String s){…..};
                        deleteResult(String s){….};
  }
```

- in Model class , write database creation and table creation logic inside a constructor.

Android Telephony :

- SMS
- Call
- Email
  - builtin Activity
  - java mail api

SMS :

android.telephony.SmsManager class is used to send text messages in Android.

```
SmsManager sManager = SmsManager.getDefault( );
sManager.sendTextMessage(receiver_number ,
source_number, message , deliver_intent , send_intent);
```

PendingIntent :

- Pending Intent is a child of Intent which is called by later.

syntax :

```
Intent i =new Intent(context,ActivityName.class);
PendingIntent pIntent =  PendingIntent.getActivity(
context , request_code[0] , intent_object , req_flag[0]);
```

permission :

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

Call :

Android is providing a builtin Activity for making calls, use implicit intent to call builtin Activity.

```
Intent i = new Intent(  );
i.setAction(Intent.ACTION_CALL);
i.setData(Uri.parse("tel:"+number));
startActivity(i);
```

permission :

```
<uses-permission android:name =

"android.permission.CALL_PHONE"/>
```

Email (Builtin Activity) :

for sending an Email android is providing a builtin Activity use implicit intents to call builtin Activity.

```
Intent i = new Intent( );
i.setAction(Intent.ACTION_SEND);
i.putExtra(Intent.EXTRA_EMAIL ,
                    new String[]{mail_id1,mail_id2….});
```

```
i.putExtra(Intent.EXTRA_SUBJECT,
                                        "subject_here");
i.putExtra(Intent.EXTRA_TEXT,
                                            "text_here");
i.putExtra(Intent.EXTRA_STREAM,uri_object);
i.setType("message/rfc822");    // enable MIME
startActivity(i.createChooser(i,"message_here"));
```

permission :

```
<uses-permission android:name=
```

"android.permission.INTERNET"/>

Attachment functionality :

```
Intent i = new Intent( );

i.setAction(Intent.ACTION_GET_CONTENT);
                    i.setType("*/*");
startActivityForResult(i,request_code(positive_integer);
```

in the above method (startActivityForResult( )) if we specify the second parameter(request_code) >= 0 after the next Activity is completed it will invoke the following method in the current Activity.

```
public void onActivityResult(int req_code , int res_code ,

Intent i)
{
        Uri  u = i.getData( );
}
```

permission :

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

Share functionality :

```
Intent i = new Intent( );
i.setAction(Intent.ACTION_SEND);
i.putExtra(Intent.EXTRA_TEXT,“text_here”);
i.setType("text/plain");
startActivity(i.createChooser(i,
        "Select Any Application”));
```

by using builtin Activity we can’t send an email in the background user has to select any one of the email client application for sending an email , to send an email in the background with out any user interaction use Java Mail API.

Steps to work with Java Mail API :

- add the following .jar files as a modules , add these modules as a dependency modules.

  - activation.jar
  - additional.jar
  - mail.jar

- copy the following .java files into package folder.

GMailSender.java

JSSEProvider.java

LongOperation.java

- configure the from credentials in LongOperation.java

- LongOperation is an AsyncTask[?] , execute the AsyncTask from Activity by using the following code.

```
LongOperation at = new LongOperation
(to_mail_id,subject,message);
at.execute( );
```

AsyncTask :

- by default every Activity will run on a Thread called Activity Thread / UI thread, from Android 3.0 we can't execute long operations [network calls] on Activity Thread because there may be a chance the Activity will be freeze (it is also called ANR State (Application Not Responding)), to execute long operations there are 2 approaches.

- execute the long operations in a separate Thread.
- execute the long operations in background on Activity Thread by using AsyncTask.

- by creating separate Thread we can execute long operations but we can't present the response on Activity from separate Thread.
- to execute long operations and to present response on Activity use AsyncTask , AsyncTask will execute on current Activity Thread in the background.

- to create AsyncTask , create a class as a child of android.os.AsyncTask.

```
class MyTask extends AsyncTask
{
        onPreExecute( ){   }
        doInBackground( ) {  }      // abstract method.
        onPostExecute( ) { }
}
```

- use the following statements in Activity to execute AsyncTask.

```
MyTask task = new MyTask( );
task.execute( );
```

- if we are executing AsyncTask , we have to inform to Activity , add the following statement inside Activity onCreate( ) method.

```
StrictMode.ThreadPolicy policy =
new StrictMode.ThreadPolicy.Builder( ).permitAll().build( );
StrictMode.setThreadPolicy(policy);
```

Service :

- A long running background process with out any user interaction is called as Service.
- Service will be execute on MainThread , recommend to perform long operations (download video file).
- there are 2 types of Services
  - Service
  - Intent Service
- In android we can create a new service or we can get the builtin services (eg : location service , notification service …) .

Steps to create a new Service.

- create a class as a child of android.app.Service class.
- it is an abstract class having an abstract method called onBind( ) method.
- following are the major methods in Service class.
  - onCreate( )
  - onStartCommand( )
  - onDestroy( )
- when we start the service if the service is not available in the stack , it will invoke onCreate( ) and onStartCommand( ) methods.
- if the service is already available in the stack it will invoke onStartCommand( ) method.
- if we stop the service it will invoke onDestroy( ) method.
- Service doesn't contain any UI , we will manage service from Activity by using Intent.

```
Intent i = new Intent( context , ServiceName.class);
startService(i);
stopService(i);
```

- same like Activity every service class should be configured in Manifest.xml with the following tag inside <application> tag.

```
<service android:name=
                                        "package_name.class_name"
/>
```

Broadcast Receiver :

- Broadcast receivers are registered for system annoucements.

eg : headset plugin , power connected /disconnected ,
making / receiving call , sending / receiving SMS.

Steps to work with Broadcast Receiver :

- create a class as a child of android.app.Broadcast Receiver.
- it is an abstract class having an abstract method called onRecieve( ) method so provide the implementation for onReceive( ) method.
- from Activity class for which events you want to get the broadcast announcements configure the events using IntentFilter.

```
IntentFilter filter = new IntentFilter( );
filter.addAction(Intent.ACTION_NAME);
…….
registerReceiver(object_of_BR,intent_filter_obj);
```

- if any one of the configured event is happened , it will invoke onReceive( ) method broadcast receiver class.

Intent Service :

- Intent Service is a child of android.app.Service class.
- It is recommend to perform long operations in the background.

- IntentService will work on Separate Thread (not on MainThread).
- Intent service will stop automatically after the background task is completed.
- IntentService is an abstract class having an abstract method called onHandleIntent( ).
- We can send the data from IntentService to Activity by using Broadcast Receiver.

Intent Service & Own Broadcast Receiver:

Content Provider :

- Content Provider is used to share the data between multiple applications [In Android one of the security feature is we can't access the other applications data into our application, but if the application is providing content provider then we can access the other applications data into our application , in Android following builtin applications are providing content provider] .

Steps to work with Content Provider [Reading Other applications data into our application] :

- to get any content provider data first we have to get an object for ContentResolver.

```
ContentResolver resolver = getContentResolver( );
```

- resolver.query(CP_URI, projection, selection,

selection_args , order) method is used to read the data from Content Provider , return type of this method is Cursor.

- if the data is available in a cursor, for presenting the data Android is providing an adapter called SimpleCursorAdapter.

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter
                    ( context, xml_file , cursor_object , from , to);

ui_comp.setAdapter(adapter);
```

permission :

READ_CONTACTS

from - String[ ] { db_column1, db_column2 }
to - int[ ] { R.id.XX,R.id.XX}

getSystemService(Context.SERVICE_NAME) method is used to get the builtin services.

Location Service :

- getSystemService(Context.LOCATION_SERVICE) method is used to get the builtin location service into Activity.
- application framework is providing a class called LocationManager to manage location service.

```
LocationManager lManager = (LocationManager)
                    getSystemService(Context.LOCATION_SERVICE);
```

- lManager.getLastKnownLocation(LocationManager.
PROVIDER_NAME) method is used to get the last updated location.

- lManager.requestLocationUpdates(PROVIDER_NAME,
min_time (milli_seconds) , min_distance (meters) ,
new LocationListener( ) {
public void onLocationChanged(Location l) {
double latitude = l.getLatitude( );
double longitude = l.getLongitude( );
lManager.removeUpdates(this); // to stop location updates
});

Permissions :

ACCESS_FINE_LOCATION

ACCESS_COARSE_LOCATION

Notification Service :

- every android device will have a notification bar / status bar we can display notifications on notifications on Notification bar by using Notification Service.

- getSystemService(Context.NOTIFICATION_SERVICE) method is used to get the builtin notification service, application framework is providing a class called NotificationManager to manage notification service.

```
NotificationManager nManager = (NotificationManager)
                    getSystemService(Context.NOTIFICATION_SERVICE);
```

- use the following code to display notification on Notification bar.

Sensor Service :

- getSystemService(Context.SENSOR_SERVICE) method is used to get the built-in sensor service into Activity, application framework is providing a class called SensorManager to manage Sensor service.

```
SensorManager sManager = (SensorManager)
                    getSystemService(Context.SENSOR_SERVICE);
```

- sManager.getDefaultSensor(Sensor.SENSOR_TYPE) method is used to get the Sensor.

```
Sensor s = sManager.getDefaultSensor(Sensor.SENSOR_TYPE);
```

- following are the major sensors in Android .

- configure the following listener to get the sensor events.

- sManager.registerListener(

```
new SensorEventListener( ){
public void onSensorChanged(float[ ] values, int accuracy){ }
……
                                        },sensor_object,delay_milli_seconds);
```

permissions :

```
<uses-feature
                android:name="android.hardware.sensor.proximity"
                android:required="true" />

<uses-feature
            android:name="android.hardware.sensor.gyroscope"
            android:required="true" />
```

Wifi Service :

- getSystemService(Context.WIFI_SERVICE) method is used to get the builtin wifi service into Activity, application framework is providing a class called WifiManager to manage wifi service.

```
WifiManager wManager = (WifiManager)
                getSystemService(Context.WIFI_SERVICE);
```

- wManager.getWifiState( ) method is used to get the wifi state, return type of this method is int (0 - disabled , 1 - disabling , 2- enabled , 3 - enabling).

- wManager.setWifiEnabled(boolean) method is used to change the wifi state.

- wManager.getScanResults( ) method is used to get the list of available wifi devices, return type of this method is List<ScanResult>.

```
List<ScanResult> list = wManager.getScanResults( );

for(ScanResult sr:list) {
                        sr.SSID;   // to get wifi device name
                    sr.frequency ; // to signal strength
}
```

- wManager.getConfiguredNetworks( ) method is used to get the list of paired wifi devices, return type of this method is List<WifiConfiguration> .

```
List<WifiConfiguration> list =
```

```
wManager.getConfiguredNetworks( );

for(WifiConfiguration config:list) {
                                        config.SSID ;   // to get network
name
                                    config.status ; // status , connected /
disconnected
            }
```

permissions :

ACCESS_WIFI_STATE

CHANGE_WIFI_STATE

ACCESS_COARSE_LOCATION

Bluetooth Service :

- In android there are different ways to manage BT service, one of the best approach is manage BT service by using BluetoothAdapter.

```
BluetoothAdapter  bAdapter  =

BluetoothAdapter.getDefaultAdapter( );
```

- bAdapter.isEnabled( ) method is used to get the BT state.
- bAdapter.enable( ) / bAdapter.disable( ) methods are used to change the BT state.
- to get the list of available BT devices we have to configure a Broadcast Receiver.

```
bAdapter.startDiscovery( ) ;

IntentFilter filter = new IntentFilter( );
filter.addAction(BluetoothDevice.ACTION_FOUND);
registerReceiver(new BroadcastReceiver( ){
```

public void onReceive(Context c, Intent i) {
// method will be called when BT device found
} }, filter);

permissions :

BLUETOOTH
BLUETOOTH_ADMIN
ACCESS__COARSE_LOCATION

Vibrator :

Vibrator vib = (Vibrator)getSystemService (Context.VIBRATOR_SERVICE);
vib.vibrate(milli_seconds);

permission :

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

Telephony Service :

- getSystemService(Context.TELEPHONY_SERVICE) is used to get the builtin Telephony service, application framework is providing a class called TelephonyManager to manage Telephony service.

TelephonyManager tManager = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);

- by using telephony we can get the device information like [IMEI number , Sim Serial Number , Network Name, Device Type (CDMA/GSM…) , Provider Country , isRoaming ….)

permissions :

READ_PHONE_STATE

following are the major methods Telephony Service class.

Android Client Application for REST WS using Retrofit :

- create a Project, add the following libraries as a dependency libraries.

- GSON
- Retrofit
- Retrofit-GSON Converter

- create euqalant Bean classes based on JSON response.
- create an interface, create an abstract method with return type of Call<MainBeanClassName>.
- on top of interface abstract method , specify the request type and configure the sub_url.

- from Activity class initialize Retrofit Object.

- create an implementation class for interface with help of Retrofit.

- call the abstract method of interface.
- make WS call using Retrofit , by executing the following method.

Google Maps :

- create a project, add google-play-services:maps project as a dependency library project.
- in Activity Xml file, create a fragment UI component with the following name.

```
<fragment
    android:id="@+id/frag1"
    android:name=
            "com.google.android.gms.maps.SupportMapFragment"
     ……. />
```

- get the SupportMapFragment from Activity XML file to java file.

```
SupportMapFragment frag = (SupportMapFragment)
        getSupportFragmentManager( ).
        findFragmentById(R.id.frag1);
```

- get the GoogleMap object from SupportMapFragment.

- to work with any GoogleAPI we required an API key from Google, go through the following URL to get an API key.

http://code.google.com/apis/console

AIzaSyCZDz3dahkFYWARcT2sxcB0c-ESjdM4R8Y

- configure the API key in Manifest.xml with the following tag inside <application> tag.

```
<meta-data
android:value="YOUR_API_KEY"
android:name="com.google.android.geo.API_KEY" />
```

use the following method to change Google Map style.

```
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Dialogs & Menu :

- In android there are 5 type of dialg boxes.

- Custom Dialog
- Alert Dialog
- DatePickerDialog
- TimePickerDialog
- ProgressDialog
    - Determinent
    - Indeterminent

Custom Dialog :

- by using Custom dialog we can create our own UI, and display the UI on dialog.

```
Dialog d = new Dialog(context);
d.setContentView(R.layout.xml_file);
d.show( );
```

- use the dialog object, before findViewById( ) to get the UI component from dialog xml.

eg : Button b1 = d.findViewById(R.id.b1);

- if the UI component is available on Dialog XML, we can't configure the events using XML.

AlertDialog :

AlertDialog is one of the builtin dialog box with few builtin functionalities like title , message , icon and max we can create 3 buttons (positive , negative and neutral).

syntax :

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("title_here");
builder.setMessage("message_here");
builder.setIcon(R.drawable.image_name);
builder.setPositiveButton("text",listener);
builder.setNegativeButton("text",listener);
builder.setNeutralButton("text",listener);
builder.show( );
```

if we click alert dialog buttons it will invoke the following listener.

```
DialogInterface.OnClickListener listener =
                    new DialogInterface.OnClickListener( ) {
    public void onClick(DialogInterface dialog, int which) {
                    // second parameter will tell u the Button type.
```

} };

DatePickerDialog :

- DPD is used to get date as input from the user.

```
DatePickerDialog did = new DatePickerDialog
(context , listener , day , month , year);
dpd.show( );
```

- if we select the date it will invoke the following listener.

```
DatePickerDialog.OnDateSetListener listener =
                    new DatePickerDialog.OnDateSetListener( ){
                            public void onDateChanged(int year , int
month, day)
                                                    {
        // code to execute when date is changed.
                                                    }
                    });
```

TimePickerDialog :

Progress Dialog :

- In Android there are 2 types of progress dialogs.
  - Determinent
  - Indeterminent

syntax :

```
ProgressDialog pDialog = new

ProgressDialog(context);
pDialog.setTitle("title_here");
pDialog.setMessage("message_here");
pDialog.setIcon(R.drawable.image_name);
```

```
pDialog.setProgressStyle(
                    ProgressDialog.STYLE_SPINNER);   //Indeterminent
    ProgressDialog.STYLE_HORIZONTAL) //determinet
pDialog.show( );
```

Material Design :

- RecyclerView + CardView - Floating Action Button
- Snack Bar
- ToolBar
- NavigationDrawer
- Menu

RecyclerView :

- RecyclerView is one of the advanced UI component and it is a replacement of ListView, Gallery and GridView.

limitations of custom adapter :

- for individual item custom adapter will inflate the xml. (load XML , convert the XML file into View object).
- for every individual item it will get the UI components from XML file.
- these are limitations are overcome in RecyclerView by using RecyclerView.Adapter.
- we can present best UI with combination of RecyclerView and CardView.

Steps to Work with RecyclerView + CardView :

- create a project add RecyclerView and CardView library projects as a dependency projects.

- in activity XML file , create a RecyclerView UI component.

```
<RecylerView
                    android:id="@+id/rview"
```

…… />

- get the RecyclerView component from XML to Java.

```
RecyclerView review = findViewById(R.id.rview);
```

- RecyclerView will act as a ListView, Gallery and GridView. In which format u want to present the UI we specify.

```
/*    LinearLayoutManager lManager =
        new LinearLayoutManager(this,
LinearLayoutManager.VERTICAL,false); */
GridLayoutManager gManager =
      new GridLayoutManager(this,2);
 rview.setLayoutManager(gManager);
```

- create an individual XML file with CardView as a root UI group.

- Create a holder class to hold the UI components of individual XML file, to create a holder class , create a class as a child of RecyclerView.ViewHolder class.

- for presenting the data on RecylerView we have to create an Adapter , to create an Adapter create a class as a child of RecyclerView.Adapter<ViewHolder>.

- it is an abstract class having following abstract methods.

  - onCreateViewHolder( )
  - onBindViewHolder( )
  - getCount( )

- set the RecyclerView adapter to the RecyclerView component.

```
rview.setAdapter(new RecyclerViewAdapterClass( ));
```

- to display menu options in Activity , configure the menu items in the following xml.

res >> menu >> any_file_name.xml

```
<menu>
        <item
        android:title="title_here"
android:icon="@drawable/image_name"/>
        ……….
</menu>
```

- override the following methods in Activity to display menu options.

```
public void onCreateOptionsMenu(Menu menu)
 {
        …….
getMenuInflater( ).inflate(R.menu.menu_xml_file,menu);
}
```

- if we select a menu option it will invoke the following method in Activity.

```
public void onOptionItemSelected(MenuItem item) {
        // logic to execute when item is selected…
}
```

- to display menu icon on toolbar configure the following attribute to the menu item.

app:showAsAction="@drawable/image_name"

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670,151.1957&radius=500&types=food&name=cruise&key=AIzaSyAOs_EhelHci1mWenmt7bCZ_Jn3yQt4lzM

Firebase :

- Authentication
- Database

- Storage
- Notifications (push notifications)

- Admob