UNIT 5 : Activity and Multimedia with databases

5.1 : Intent , Intent filter

- **Intent:**

1. An Intent is a messaging object you can use to request an action from another app component.
2. Intents are used for facilitating communication between components like Activities, Services and Broadcast Receivers.
3. Android uses intent for communicating between components of an application and also from one application to another application
4. There are two intents available in android as Implicit Intents and Explicit Intents.
   Explicit Intent − It going to connect the internal world of an application such as start activity or send data between two activities.
   Implicit Intents − It going to connect with out side application such as call, mail, phone, see any website ..etc.
5. Android intents are mainly used to:
   - ✓ Start the service
   - ✓ Launch an activity
   - ✓ Display a web page
   - ✓ Display a list of contacts
   - ✓ Dial a phone call etc.

i)Explicit intent:

1. Explicit Intents are used to connect the application internally.
2. In Explicit we use the name of component which will be affected by Intent.
3. For Example: If we know class name then we can navigate the app from One Activity to another activity using Intent. In the similar way we can start a service to download a file in background process.
4. Explicit Intent work internally within an application to perform navigation and data transfer. The below given code snippet will help you understand the concept of Explicit Intents
5. Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
   startActivity(intent);

Here SecondActivity is the JAVA class name where the activity will now be navigated.
Example with code in the end of this post will make it more clear.

ii)Implicit Intent:

1. In Implicit Intents we do need to specify the name of the component. We just specify the Action which has to be performed and further this action is handled by the component of another application.
2. The basic example of implicit Intent is to open any web page Let's take an example to understand Implicit Intents more clearly. We have to open a website using intent in your application. See the code snippet given below
3. Intent intentObj = new Intent(Intent.ACTION_VIEW);

```
intentObj.setData(Uri.parse("https://www.abhiandroid.com"));
startActivity(intentObj);
```

Unlike Explicit Intent you do not use any class name to pass through Intent(). In this example we has just specified an action. Now when we will run this code then Android will automatically start your web browser and it will open AbhiAndroid home page.

Different Methods Used in Intent

- ✓ ACTION_PICK
  It is using for picking the image from CAMERA or GALLERY.
  Eg : Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
- ✓ ACTION_CHOOSER
  Use: It is used for choosing the image from the gallery.
- ✓ ACTION_DIAL
  Use – Display the phone dialer with the given number filled in.
- ✓ ACTION_SEND
  Use: Sending Text content from one activity to other.
- ✓ ACTION_VIEW
  Use : to open window

Getting a Result from an Activity startActivityForResult() method

Starting another activity doesn't have to be one-way. You can also start another activity and receive a result back. To receive a result, call startActivityForResult() (instead of startActivity()). For example, your app can start a camera app and receive the captured photo as a result.
By the help of android startActivityForResult() method, we can send information from one activity to another and vice-versa.
Syntax :
startActivityForResult(intent , request _code)
Intent : specify the action type
Request _code : specify your request

onActivityResult

onActivityResult method that is invoked automatically when second activity returns result. When the user is done with the subsequent activity and returns, the system calls your activity's onActivityResult() method. This method includes three arguments:

void onActivityResult() (int requestCode, int resultCode, Intent data)

The request code you passed to startActivityForResult().

A result code specified by the second activity. This is either RESULT_OK if the operation was successful or RESULT_CANCELED if the user backed out or the operation failed for some reason.

An Intent that carries the result data.

requestCode

The integer argument is a "request code" that identifies your request.

The requestCode helps you to identify from which Intent you came back. For example, imagine your Activity A (Main Activity) could call Activity B (Camera Request), Activity C (Audio Recording), Activity D (Select a Contact).

Whenever the subsequently called activities B, C or D finish and need to pass data back to A, now you need to identify in onActivityResult() from which Activity you are returning from and put your handling logic accordingly.

Eg :

```
public static  int  CAMERA_REQUEST = 1;
 public static int  CONTACT_VIEW = 2;
public void onActivityResult(int requestCode, int resultCode, Intent data)
 {  if (requestCode == CAMERA_REQUEST) {
// code to handle data from CAMERA_REQUEST }
 else if (requestCode == CONTACT_VIEW) {
// code to handle data from CONTACT_VIEW
```

Intent methods to set data n get data

how strings are being added to Extras:

```
Intent i = new Intent();
i.putExtra("Name", edt_name.getText());
i.putExtra("Description", edt_desc.getText());
```

how to extract them :

```
String name = data.getStringExtra("Name");
String desc = data.getStringExtra("Description");
```

- **Logcat**

Dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app.

e() method is used to log errors.

w() method is used to log warnings.

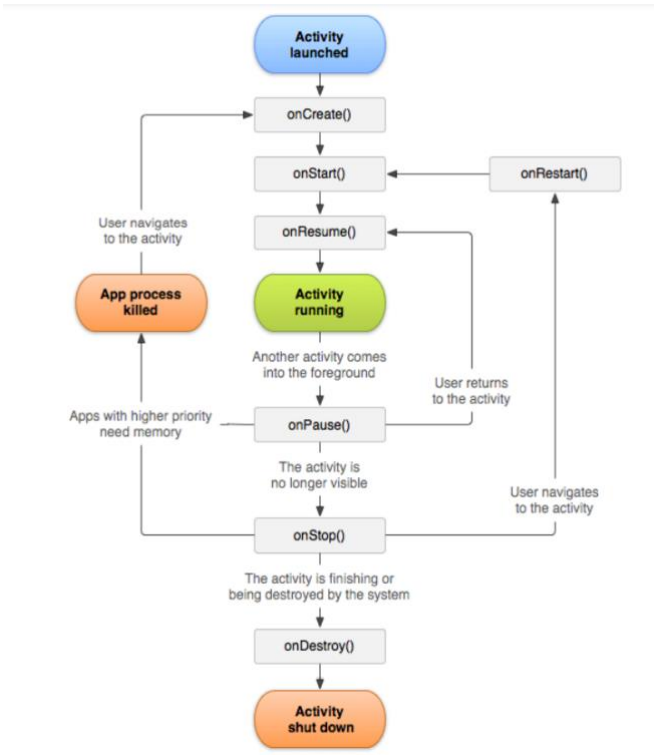
i() method is used to log informational messages.

d() method is used to log debug messages.

5.2 Activity lifecycle , Broadcast lifecycle.

- **Activity:**
  1. Activity is one of the building blocks of Android OS
  2. Activity is a screen that user interact with.
  3. Every Activity in android has lifecycle like created, started, resumed, paused, stopped or destroyed.
  4. Every Activity in android has lifecycle like created, started, resumed, paused, stopped or destroyed. - These different states are known as Activity Lifecycle.
  5. In other words, we can say Activity is a class pre-written in Java Programming. - Below Activity Lifecycle Diagram Shows Different States:

Activity lifecycle :



onCreate (): Called when the activity is created. Used to initialize the activity, for example create the user interface.Only called once during an activity's lifecycle.

onStart ():called when activity is becoming visible to the user.Can be called more than once during lifecycle. Followed by onResume()

onResume (): Called if the activity get visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc.
In this state activity start interacting with user that means user can see the functionality and designing part of an application on the single screen.
The app will stays in this Resumed state until an another activity happens to take focus away from the app like getting a phone call or screen turned off, etc.
It always gets called after activity start and when activity paused.

onPause (): Called once another activity gets into the foreground. Always called before the activity is not visible anymore. Used to release resources or save application data. For example you unregister listeners, intent receivers, unbind from services or remove system service listeners.

onStop():The system will invoke onStop() callback method when an activity no longer visible to the user.
Application main thread is not always the UI thread. For example, when Activity is stopped, the onStop() is invoked, hence the UI thread is taken away from that Activity and moved to another Activity within the same or a different application. However it doesn't mean the application is no longer active, it can continue working in the background until it is closed either by OS or by user.

onRestart()
The system will invoke onRestart() method when an activity restarting itself after stopping it. The onRestart() method will restore the state of activity from the time that is being stopped.

onDestroy()
The system will invoke this onDestory() callback method either the activity is finishing or system destroying the activity to save space.

Broadcast Intent

Basically intent are of two types:
Implicit - Explicit
Broadcast Intent :
which don't start activities but instead are delivered to broadcast receiver
1. system broadcast intent: delivered by the system.
Eg : For example, applications can register for various system events like boot complete or battery low, and Android system sends broadcast when specific event occur.

2.custom broadcast intent : those that your app delivers.
Eg : when you want let other app know that data has been downloaded to the device.

Difference between Activity intent and Broadcast Intent

- The intent used to start activity makes changes to an operation and user is aware of the process.
- In case of broadcast intent the operation runs completely in background and it is invisible to user.

System broadcast intent

For example, applications can register for various system events like boot complete or battery low, and Android system sends broadcast when specific event occur.
android.intent.action.BATTERY_LOW
Indicates low battery condition on the device.
android.intent.action.BATTERY_OKAY
Indicates the battery is now okay after being low.
android.intent.action.BOOT_COMPLETED
This is broadcast once, after the system has finished booting.

- **Broadcast receiver:**

1. Broadcast Receivers A broadcast receiver (receiver) allows you to register for system or application events.
2. All registered receivers (App) for an event are notified by the Android runtime once this event happens.
3. If the event for which the broadcast receiver(App) has registered happens, the onReceive() method of BroadcastReceiver class is called by the Android system.
4. Two steps

Creating the Broadcast Receiver.
Registering Broadcast Receiver

Creating Broadcast Receiver :

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as a Intent object parameter.

```
public class MyReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent)
{ Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show(); } }
```

Registering Broadcast Receiver :

An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file. Consider we are going to register MyReceiver for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.

```
<application android:icon="@drawable/ic_launcher"
android:label="@string/app_name" android:theme="@style/AppTheme" >
<receiver android:name="MyReceiver">
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED">
</action>
</intent-filter>
</receiver>
</application>
```

Now whenever your Android device gets booted, it will be intercepted by Broadcast Receiver MyReceiver and implemented logic inside onReceive() will be executed. - There are several system generated events defined as final static fields in the Intent class. -

-android.intent.action.BATTERY_LOW : It is used to call an event when battery is low on device.
- android.intent.action.BATTERY_OKAY : It is used to call an event when battery is OKAY again.
- android.intent.action.REBOOT : It call an event when the device rebooted again.
- android.intent.action.BOOT_COMPLETED : It raise an event, once boot completed.
- android.intent.action.POWER_CONNECTED : It is used to trigger an event when power connected to the device
- android.intent.action.POWER_DISCONNECTED : It is used to trigger an event when power got disconnected from the device

Custom Broadcast intent

Broadcast intent that your application send out by using the sendBroadcast()

Eg : when you want let other app know that data has been downloaded to the device.

TO create custom boradcast intent , create custom intent action. To deliver a custom broadcast to other app, pass the intent to SendBroadcast()

Custom Broadcast steps….

1.Create broadcast reciever MyReceiver :

Eg ://MyReciever class which extends BroadcastReceiver

```
public class MyReceiver extends BroadcastReceiver {
 public void onReceive(Context context, Intent intent) {
 Toast.makeText(this , “Intent deteced “ ).show }
```

2.register or Define BroadcastReceiver in AndroidManifest.xml file .

Eg :

```
<receiver android:name="MyReceiver" >
    <intent-filter>
      <action android:name="com.example1.CUSTOM_INTENT” />
    </intent-filter>
</receiver>
```

3. Send Broadcast event : Create new intent and broadcast it

eg :

```
Intent intent1 = new Intent();
 intent1.setAction(" com.example1 .CUSTOM_INTENT");
 sendBroadcast(intent);
```

5.3 Content Provider , fragments

- **Content Provider:**
  1. A content provider component supplies data from one application to others on request.
  2. Content providers let you centralize content in one place and have many different applications access it as needed.
  3. In android, Content Provider will act as a central repository to store the applications data in one place and make that data available for different applications to access whenever it’s required.
  4. In android, we can configure Content Providers to allow other applications securely access and modify our app data based on our requirements.
  5. Generally, the Content Provider is a part of an android application and it will act as more like relational database to store the app data. We can perform a multiple operations like insert, update, delete and edit on the data stored in content provider using insert(), update(), delete() and query() methods.
  6. In android, content provider is having different ways to store app data. The app data can be stored in a SQLite database or in files or even over a network based on our requirements. By using content providers we can manage data such as audio, video, images and personal contact information.

Access Data from Content Provider

To access a data from content provider, we need to use ContentResolver object in our application to communicate with the provider as a client. The ContentResolver object will communicate with the provider object (ContentProvider) which is implemented by instance of class.

Generally, in android to send a request from UI to ContentResolver we have another object called CursorLoader which is used to run the query asynchronously in background.

In android application the UI components such as Activity or Fragment will call a CursorLoader to query and get a required data from ContentProvider using ContentResolver.

The ContentProvider object will receive a data requests from client, performs the requested actions (create, update, delete, retrieve) and return the result.

Content URI

Content URI(Uniform Resource Identifier) is the key concept of Content providers. To access the data from a content provider, URI is used as a query string.

Structure of a Content URI: content://authority/optionalPath/optionalID

1 content://

Mandatory part of the URI as it represents that the given URI is a Content URI

2 authority

This specifies the name of the content provider, for example contacts, browser etc. For third-party content providers, this could be the fully qualified name. This part must be unique for every content provider.

3 data_type

This indicates the type of data that this particular provider provides. For example, if you are getting all the contacts from the Contacts content provider, then the data path would be people and URI would look like this content://contacts/people

4 id

This specifies the specific record requested. It is a numeric value that is used when there is a need to access a particular record.

For example, if you are looking for contact number 5 in the Contacts content provider then URI would look like this content://contacts/people/5.

Create Content Provider

First of all you need to create a Content Provider class that extends the ContentProvider baseclass.

Second, you need to define your content provider URI address which will be used to access the content.

Next you will need to create your own database to keep the content. Usually, Android uses SQLite database

Next you will have to implement Content Provider queries to perform different database specific operations.

Finally Register content provider in AndroidManifest.xml file using <provider.../> tag

Methods of Content Provider

onCreate() This method is called when the provider is started.
query() This method receives a request from a client. The result is returned as a Cursor object.
insert()This method inserts a new record into the content provider.
delete() This method deletes an existing record from the content provider.
update() This method updates an existing record from the content provider.

- **Fragments:**
  1. Android Fragment is the part of activity, it is also known as sub-activity.
  2. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.
  3. single activity but each activity can comprise of multiple fragments which will have their own layout, events and complete life cycle.
  4. Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.
  5. You can add or remove fragments in an activity while the activity is running.
  6. A fragment can be used in multiple activities.
  7. Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.

Uses of Fragment

Flexible user interface across different screen size
Fixed / scrolling / swipe tab display eg : watsApp tabs 3 different fragment on single activity.
To display dialog boxes eg : Do u want to exit
Tab modes possible with fragment.

Types of Fragments

Single frame fragments − Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.
List fragments− This Fragment is used to display a list-view from which the user can select the desired sub-activity. ...
Fragments transaction − Fragment Transaction: This kind of fragments supports the transition from one fragment to another at run time. Users can switch between multiple fragments like switching tabs.

How to add fragment

Add a fragment using XML
Define fragment within mainActivity xml by calling fragment class name in android:name
If you add fragment using xml , then it is not possible to remove the fragment at run time.
Add fragment programmatically to remove or replace.Add a fragment using runtime
Create object of FragmentManager class and FragmentTransaction class

How to use Fragments

1.decide how many fragments you want to use in an activity.
2.based on number of fragments, create classes which will extend the Fragment class.
3.Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.
4.Finally modify activity file to define the actual logic of replacing fragments

onCreateView() The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View component from this method
Syntax :
public View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

Inflater : This LayoutInflater object that can be used to inflate any views in the fragment.
Container : If non-null, this is the parent view that the fragment's UI should be attached to.
savedInstanceStateBundle: If non-null, fragment is being re-constructed from a previous saved state as given here.
ReturnsView : Return the View for the fragment's UI, or null.

Fragment Manager

Every activity has its own fragment manager
It maintain references to all fragments inside the activity.
findFragmentByid() method to get a references to a particular fragment inside the activity.

Fragment Transaction

The act of adding and removing fragments to an activity are considered as transaction inside android and managed by an object called FragmentTransaction.
Two methods : add(); and replace();

Case 1: When there is no fragment attached in a container
In this case, both methods will add the fragment to the container.

Case 2: When the fragmentContainer already has fragment/fragments than ,
add(): adds the new fragment on the top another fragment
replace(): removes everything then adds the new fragment

5.4 Service -Features of service , android platform service Defining new service , service lifecycle , permission

- **Service :**

1. A service is an application component which runs without direst interaction with the user in the background.
2. Services are used for repetitive and potentially long running operations, i.e., Internet downloads, checking for new data, data processing, updating content providers and the like.
3. Service can either be started or bound we just need to call either startService() or bindService() from any of our android components. Based on how our service was started it will either be "started" or "bound"
4. There can be two forms of a service. The lifecycle of service can follow two different paths: started or bound. 1. Started 2. Bound

Started Service or unbound -

1) If a service can be started by the application component then it is called started service.
2) A service is started when an application component, such as an activity, starts it by calling startService().
3) Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
4) Started service performs a single operation and doesn't return a result to the caller.
5) Eg : when download a file over the network , when upload completes the service stop itself by calling stopService().

Bound Service -

1) A service is bound when an application component binds to it by calling bindService().
2) A bound service offers a client-server interface that allows components to interact with the service, send requests, get results.
3) Multiple components can bind to service at once , but when all of them unbind , the service is destroyed.

DIFFERENCE BETWEEN BOUND AND UNBOUND SERVICE

UnBound Service:

1.Basically used for long repetitive task.

2.startService() is the method use to start unbound service.

3.stopService() is the method use to stop explicitly.

4.It is independent of the component from which it starts.

Bound Service:

1.Basically used for long repetitive task but bound with the component.

2.starts by **bindService().**

3.unbindService() is the method use to stop explicitly.

4.It is dependent of the component from which it starts

➢ **Features of Service:**

- Service is an Android Component without an UI.
- It is used to perform long running operations in background. Services run indefinitely unless they are explicitly stopped or destroyed.
- It can be started by any other application component. Components can even infact bind to a service to perform Interprocess
- Communication.
- It can still be running even if the application is killed unless it stops itself by calling stopself() or is stopped by an Android component by calling stopService().
- If not stopped it goes on running unless is terminated by Android due to resource shortage.
- The android.app.Service is subclass of ContextWrapper class.

Android Service Callback Methods

onStartCommand()

The system will invoke this callback method when main activity requests the service to be started by calling startService(). When this method executed, the service will start and run indefinitely in background.

And stop the service once code execution is done by calling stopService().

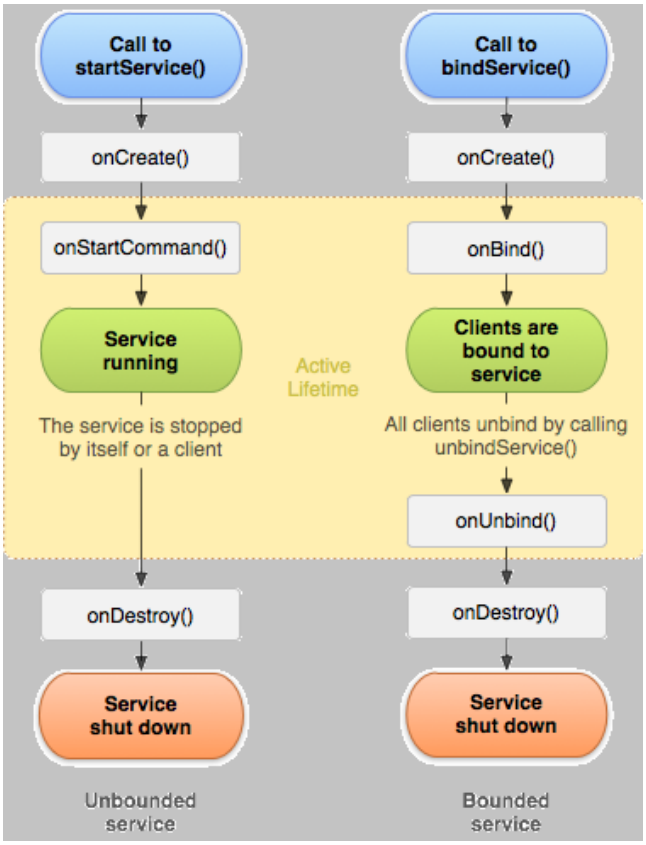
The onStartCommand() method will return a value from one of the following constants.

START_STICKY: It will restart the service in case if it terminated.

START_NOT_STICKY:It will not restart the service and it is useful for the services which will run periodically.

It's a best option to avoid running a service in case if it is not necessary.

➢ **Service lifecycle :**



1. Started
a. A service is started when an application component, such as an activity, starts it by calling startService().
b. Now the service can run in the background indefinitely, even if the
component that started it is destroyed.

2. Bound
a. A service is bound when an application component binds to it by calling bindService().
b. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with InterProcess Communication (IPC).
c. Like any other components service also has callback methods. These will be invoked while the service is running to inform the application of its state. Implementing these in our custom service would help you in performing the right operation in the right state. •
d. There is always only a single instance of service running in the app. If you are calling startService() for a single service multiple times in our application it just invokes the onStartCommand() on that service. Neither is the service restarted multiple times nor are its multiple instances created

1. onCreate():
This is the first callback which will be invoked when any component starts the service. If the same service is called again while it is still running this method wont be invoked. Ideally one time setup and intializing should be done in this callback.

2. onStartCommand() /startSetvice()
This callback is invoked when service is started by any component by calling startService(). It basically indicates that the service has started and can now run indefinetly.

3. onBind()
To provide binding for a service, you must implement the onBind() callback method. This method returns an IBinder object that defines the programming interface that clients can use to interact with the service.

4. onUnbind()
This is invoked when all the clients are disconnected from the service.

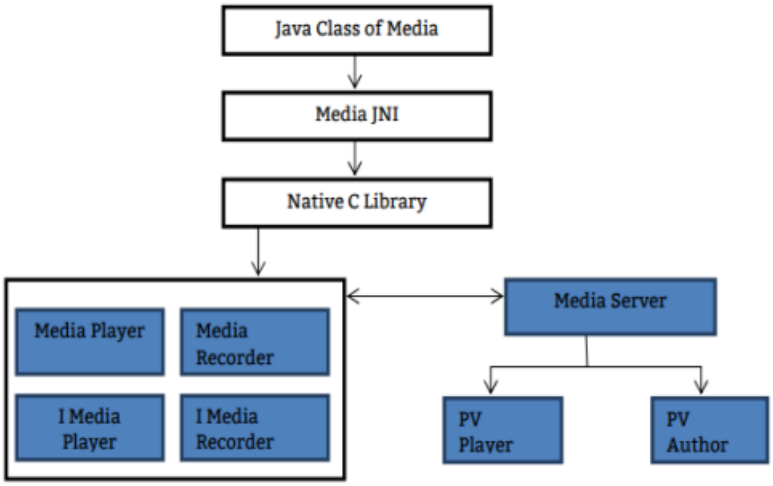
5. onRebind()
This is invoked when new clients are connected to the service. It is called after onRebind

6. onDestroy()
This is a final clean up call from the system. This is invoked just before the service is being destroyed.

5.5 Multimedia framework , play audio and video , text to speech , sensors , async tasks

- **Multimedia framework :**

1. The android multimedia system includes multimedia applications, multimedia frameworks, OpenCore engine and hardware abstract for audio/video input/output devices. And the goal of the android multimedia framework is to provide a reliable interface for java services. The multimedia framework consists of several core dynamic libraries such as libmediajni, libmedia, libmediaplayservice and so on.
2. Java classes call the Native C library Libmedia through Java JNI(Java Native Interface). Libmedia library communications with Media Server guard process through Android's Binder IPc (inter process communication) mechanism.
3. Media Server process creates the corresponding multimedia service according to the Java multimedia applications.
4. The whole communication between Libmedia and Media Server forms a Client/Server model



Android Multimedia Framework Architecture

- ✓ Typical video/audio data stream works in Android as follows. Particularly, Java applications first set the URI of the media (from file or network) to PVPlayer through Java framework, JNI and Native C. In this process, there are no data stream flows.
- ✓ Then PVPlayer processes the media data stream with the steps: demux the media data to separate video/audio data stream, decode video/audio data, sync video.audio time, send the decoded data out.
- ✓ The below is the description of media codec/format, container and network protocol supported by the Android platform.

1. Container: The audio file format is a file for storing digital audio data on a system. This data can be manipulated to reduce the size or change the quality of the audio. It is a kind of container to store audio information.
2. Audio Format: Any format or codec can be used including the ones provided by Android or those which are specific devices. However it is recommended to use the specified file formats as per devices.
3. Network Protocol: Protocols such as RTSP, HTTP,HTTPS are supported in audio and video playback.

➢ **Play audio and video :**

We can play and control the audio files in android by the help of MediaPlayer class.
Here, we are going to see a simple example to play the audio file. In the next page, we will see the example to control the audio playback like start, stop, pause etc.
MediaPlayer class
The android.media.MediaPlayer class is used to control the audio or video files.

Methods of MediaPlayer class
There are many methods of MediaPlayer class. Some of them are as follows:

Methods

- **getCurrentPosition() :** It is used to get the current position of the song in milliseconds.
- **getDuration() :** It is used to get the total time duration of the song in milliseconds.
- **isPlaying() :** It returns true / false to indicate whether song playing or not.
- **pause():** It is used to pause the song playing.
- **setAudioStreamType() :** it is used to specify the audio streaming type.
- **setDataSource() :** It is used to specify the path of audio / video file to play.
- **setVolume() :** It is used to adjust media player volume either up / down.
- **seekTo(position) :** It is used to move song to particular position in milliseconds.
- **start() :** It is used to start playing the audio/video.
- **stop() :** It is used to stop playing the audio/video.
- **reset() :** It is used to reset the MediaPlayer object.

Name two classes used to play audio and video in Android.

1) MediaPlayer
2) MediaController
3) AudioManager

➢ **Text to speech :**

Android allows you convert your text into voice. Not only you can convert it but it also allows you to speak text in variety of different languages.
Text to Speech converts the text written on the screen to speech like you have written “Hello World” on the screen and when you press the button it will speak “Hello World”. Text-to-speech is commonly used as an accessibility feature to help people who have trouble in reading
Instantiate an object of TextToSpeech class and also specify the initListener. Its syntax is given below:

```
private EditText ;
ttobj=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
  @Override
  public void onInit(int status) {
```

```
  }
});
```

OnInitListener to notify the completion of initialization. During the initialization, we can set the audio pitch rate, audio speed, type of language to speak, etc. based on our requirements.

Eg : ttobj.setLanguage(Locale.UK);
void onInit (int status) : Called to signal the completion of the TextToSpeech engine initialization. The status can be SUCCESS or ERROR

call speak method of the class to speak the text.
Eg : ttobj.speak(toSpeak , TextToSpeech.QUEUE_FLUSH , null);

Method	Description
int speak (String text, int queueMode, HashMap params)	converts the text into speech. Queue Mode may be QUEUE_ADD or QUEUE_FLUSH. Request parameters can be null, KEY_PARAM_STREAM, KEY_PARAM_VALUME etc.
int setSpeechRate(float speed)	it sets the speed for the speech.
int setPitch(float speed)	it sets the pitch for the speech.
int setLanguage (Locale loc)	it sets the locale specific language for the speech.
void shutdown()	it releases the resource set by TextToSpeech Engine.
int stop()	it interrupts the current utterance (whether played or rendered to file) and discards other utterances in the queue.

TextToSpeech.QUEUE_FLUSH

Each TextToSpeech instance can manage its own queue in order to control which utterance will **interrupt** the current one and which one is simply queued. Here the first speak () request would interrupt whatever was currently being synthesized: the queue is flushed and the new utterance is queued.

Android **TextToSpeech QUEUE_FLUSH** Queue mode where all entries in the playback queue (media to be played and text to be synthesized) are dropped and replaced by the new entry.

QUEUE_ADD : Queue mode where the new entry is added at the end of the playback queue.

➢ Sensors :

Sensors can be used to monitor the three-dimensional device movement or change in the environment of the device.

Android provides sensor api to work with different types of sensors.

Category	Description
Motion Sensors	These sensors are useful to measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
Environmental Sensors	These sensors are useful to measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.
Position Sensors	These sensors are useful to measure the physical position of a device. This category includes orientation sensors and magnetometers.

Sensor framework

Access all the sensors available on device and to get all the raw sensor data.

by using sensor framework we can perform following things

-It lists all the available sensors on the device

-It determine the capabilities of each sensor, such as its maximum range, manufacturer, power requirements, and resolution.

-It can acquire raw sensor data.

-Register and unregister sensor event listeners that monitor sensor changes.

Android sensor framework provide following classes

Class	Description
Sensor Manager	By using this class we can create an instance of sensor service and this class provides a various methods for **accessing and listing sensors, registering and unregistering sensor event listeners (listener will get informed, if the sensor data changes) and acquiring orientation information.**
Sensor	By using this class we can **create an instance of a specific sensor** and this class provides a various methods that let you **determine the sensor's capabilities.**
SensorEvent	The system uses this class to create a sensor event object and it provides the **raw sensor data, type of sensor that generated the event**
SensorEventListener	It provide two callback methods that give information when sensor values (x,y and z) change or sensor accuracy changes.

Public and abstract methods	Description
void onAccuracyChanged(Sensor sensor, int accuracy)	it is called when sensor accuracy is changed.
void onSensorChanged(SensorEvent event)	it is called when sensor values are changed.

➢ Async tasks :

Main thread / UI thread

All components are instantiate
Dispatching events
Handle call back methods
All UI components are created

Main thread is the thread used to run your application , handle processing data, handling requests, and other main tasks.

The UI thread would be responsible for handling user requests via the UI. It would put a request to the main thread for information or send an instruction.

Do not perform long running operation on main thread
If we perform long running task on main thread it will hang app

Async Task

In Android, AsyncTask allows us **to run the instruction in the background and then synchronize again with our main thread.**

gives us the liberty to perform heavy tasks in the background and keep the UI thread light thus making the application more responsive.

Exmaple : AsyncTask is just like a side road, while large-slowly moving truck makes traffic problem on the main road, Asynctask keeps those away to make a better path for other traffic.

When to use : if you want to do any resource intensive task which takes more than 5 seconds you should not do it on main thread as it will lead to Application Not Responding (ANR) dialogue pop up.

AsyncTask Class:

Use AsyncTask class to implement an asynchronous long running task.

Need of AsyncTask?

Assume you have created a simple Android application which downloads MP3 file from Internet on launching the application.
As the response (MP3 file) from server is awaited, the application has become unresponsive

AsyncTasks are designed for **once-off time-consuming tasks** that cannot be run on UI thread.
Used for handling those tasks that you cannot make to work on the main thread. presenting the result of that work to the UI thread.
example if you want to load data to a listview from a server after hitting some button example is fetching/processing data when a button is pressed.

An Async Task is launch when you need it and executed in background. When it's done, the thread is destroyed.

	Service	AsyncTask
When to use ?	Task with no UI, but shouldn't be too long. Use threads within service for long tasks.	- Long task having to communicate with main thread. - For tasks in parallel use multiple instances OR Executor [1]
Trigger	Call to method onStartService()	Call to method execute()
Triggered From (thread)	Any thread	Main Thread
Runs On (thread)	Main Thread	Worker thread. However, Main thread methods may be invoked in between to publish progress.
Limitations / Drawbacks	May block main thread	- one instance can only be executed once (hence cannot run in a loop) [2] - Must be created and executed from the Main thread

OR

- A Service is a component in Android that runs in the background, independent of any UI components.
- It is used for long-running operations or tasks that need to continue even when the app's UI is not visible.
- Services can perform tasks such as playing music, downloading files, or handling network requests.
- Services can run indefinitely or be started and stopped as needed.
- They can run in the same process as the app or in a separate process.

- An AsyncTask is a class that allows you to perform background operations and update the UI thread with the results.
- It is typically used for short-lived operations that are tied to a specific Activity or Fragment.
- AsyncTask runs on a separate thread from the main UI thread, allowing you to perform time-consuming tasks without blocking the UI.
- It provides methods like `onPreExecute()`, `doInBackground()`, and `onPostExecute()` to handle the different stages of the background operation.

Need of AsyncTask

- By default, our application code runs in our main thread and every statement is therefore execute in a sequence.
- If we need to perform long tasks/operations then our main thread is blocked until the corresponding operation has finished.
- For providing a good user experience in our application we need to use **AsyncTasks class** that runs in a separate thread.
- This class will executes everything in doInBackground() method inside of other thread which doesn't have access to the GUI where all the views are present. **Hence use of AsyncTask in android application keeps the UI thread responsive at all times.**

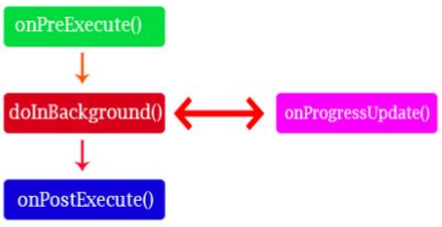
Method of AsyncTask In Android:

onPreExecute() – It invoked on the main UI thread before the task is executed. Before doing background operation we should show something on screen like progressbar or any animation to user.

doInBackground(Params..) – In this method we have to do background operation on background thread. result of the operations must be returned to method i.e onPostExecutes().

It also call publishProgress(progress..) to publish one or more units of progress.

AsyncTask Flow



onProgressUpdate(Progress...) – While doing background operation, if you want to update some information on main UI, we can use this method.

This method is used to. display any form of progress in the user interface while the background operations are executing eg : showing progress

onPostExecute(Result) This method is invoked on the main UI thread after the background operation finishes in the doInBackground method.

The result of the background operation is passed to this step as a parameter and then we can easily update our UI to show the results.

AysncTask parameters

Syntax :

AsyncTask<"Params" , "Progress" , "Result">

Three parameters:

"params" : specifies the type of parameters passed to doInBackground() as an array.

"progress" : specifies the type of parameters passed to publishProgress() on the background thread . These parameters are passed to onProgressUpdate() method.

"Result" : specifies the type of parameters that doInBackground() returns. This parameters are automatically passed to onPostExecute() on the main thread

5.6 Audio capture , camera

- **Camera :**

Camera is mainly used to capture picture and video. We can control the camera by using methods of camera API.
Android provides the facility to work on camera by 2 ways:
By Camera Intent
By Camera API

Using existing android camera application in our application

You will use MediaStore.ACTION_IMAGE_CAPTURE to launch an existing camera application installed on your phone. Its syntax is given below:-

```
Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

Sr.No	Intent type and description
1	**ACTION_IMAGE_CAPTURE_SECURE** It returns the image captured from the camera , when the device is secured
2	**ACTION_VIDEO_CAPTURE** It calls the existing video application in android to capture video
3	**EXTRA_SCREEN_ORIENTATION** It is used to set the orientation of the screen to vertical or landscape
4	**EXTRA_FULL_SCREEN** It is used to control the user interface of the ViewImage
5	**INTENT_ACTION_VIDEO_CAMERA** This intent is used to launch the camera in the video mode
6	**EXTRA_SIZE_LIMIT** It is used to specify the size limit of video or image capture size

Using Camera By using Camera Api

This class is used for controlling device cameras. It can be used to take pictures when you are building a camera application.

Camera API works in following ways:

1.Camera Manager: This is used to get all the cameras available in the device like front camera back camera each having the camera id.

2.CameraDevice: You can get it from Camera Manager class by its id.

3.CaptureRequest: You can create a capture request from camera device to capture images.

4.CameraCaptureSession: To get capture request's from Camera Device create a CameraCaptureSession.

5.CameraCaptureSession.CaptureCallback: This is going to provide the Capture session results.

- **Bluetooth :**

Bluetooth is a way to send or receive data between two different devices.

Android provides Bluetooth API (Provides classes that manage Bluetooth functionality, such as scanning for devices, connecting with devices, and managing data transfer between devices) to perform these different operations.

Scan for other Bluetooth devices

Get a list of paired devices

Connect to other devices

1. android bluetooth Adapter class:

Android provides BluetoothAdapter class to communicate with Bluetooth. Create an object of this class by calling method getDefaultAdapter().

By using BluetoothAdapter object, we can interact with device's Bluetooth adapter to perform Bluetooth related operations.

syntax:

private BluetoothAdapter BA; BA = BluetoothAdapter.getDefaultAdapter();

2.Android Enable or Disable Bluetooth: enable the Bluetooth of your device, call the intent with the following Bluetooth constant ACTION_REQUEST_ENABLE.

Syntax : Intent turnOn = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
startActivityForResult(turnOn, 0);

other constants are :

Sr.No	Constant & description
1	**ACTION_REQUEST_DISCOVERABLE** This constant is used for turn on discovering of bluetooth
2	**ACTION_STATE_CHANGED** This constant will notify that Bluetooth state has been changed
3	**ACTION_FOUND** This constant is used for receiving information about each device that is discovered

3. Android List paired Devices :

get a list of paired devices by calling getBondedDevices() method. It returns a set of bluetooth devices.

Syntax :

private Set<BluetoothDevice>pairedDevices;
pairedDevices = BA.getBondedDevices();

Sr.No	Method & description
1	**enable()** This method enables the adapter if not enabled
2	**isEnabled()** This method returns true if adapter is enabled
3	**disable()** This method disables the adapter
4	**getName()** This method returns the name of the Bluetooth adapter
5	**setName(String name)** This method changes the Bluetooth name
6	**getState()** This method returns the current state of the Bluetooth Adapter.
7	**startDiscovery()** This method starts the discovery process of the Bluetooth for 120 seconds.

Permissions

```
<manifest ... >
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
</manifest>
```

BLUETOOTH_ADMIN to discover the available Bluetooth devices or manipulate Bluetooth settings

BULETOOTH : perform any Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data.

LOCATION : to gather the information about the location of user.

Note: If you use `BLUETOOTH_ADMIN` permission, then must also have the `BLUETOOTH` permission.

5.8 SQL

SQLite

SQLite is an open-source relational **database** i.e. used to perform **database** operations on **android** devices such as storing, manipulating or retrieving persistent data from the **database**. It is embedded in **android** by default. So, there is no need to perform any **database** setup or administration task.

Support standard relational db features.

To create and upgrade a db in your android app , create subclass of SQLiteOpenHelper class .

SQLiteOpenHelper

contains a useful set of APIs for managing your database

In android, by using **SQLiteOpenHelper** class we can easily create required database and tables for our application. To use **SQLiteOpenHelper**, we need to create a subclass that overrides the **onCreate()** and **onUpgrade()** call-back methods. The SQLiteOpenHelper only require the DATABASE_NAME to create database.

After extending SQLiteOpenHelper you will need to implement its methods onCreate, onUpgrade and constructor.

SQLiteOpenHelper will ease the management of your SQLite database tremendously, by opening databases when needed, creating databases if they do not exist as well as upgrading as necessary.

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {
 public final static String DATABASE_NAME = "Student.db";
   public final static String TABLE_NAME = "student_info";
   public final static String COL_1 = "ID";
   public final static String COL_2 = "Roll_No";
   public final static String COL_3 = "Name";

  public DBHelper(Context context){
    super(context,DATABASE_NAME,null,1);
  }
  public void onCreate(SQLiteDatabase db) {
 db.execSQL("CREATE TABLE "+TABLE_NAME+" (ID INTEGER PRIMARY KEY
AUTOINCREMENT, ROLL_NO TEXT, NAME TEXT)");
}
  public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion)
{db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
      onCreate(db);
}
```

Database - Creation

openOrCreateDatabase() method to create a database.

eg:

SQLiteDatabase db = openOrCreateDatabase("your database name", MODE_PRIVATE,null);

Parameter:

String -> It is the database name, If the given name already exist in your package it will open the database, if not it will create a database with the given name.

Int -> The mode for opening the database usually it is MODE_PRIVATE.

CursorFactory -> It is nullable or optional.

- **Describe with example, how to create a simple database in SQLite (Assume suitable data).**

1. The package imported into the application is android.database.sqlite.SQLiteDatabase.
2. Here the class used is SQLiteDatabase.
3. The method used to create the database or connect to the database is openOrCreateDatabse() method.

XML

```
<RelativeLayout>
<Button android:text="Create SQLite Database" android:layout_width="fill_parent“
 android:layout_height="wrap_content“
 android:layout_centerHorizontal="true“
 android:layout_marginTop="46dp“
 android:id="@+id/button" />
</RelativeLayout>
```

Java code

```
public class MainActivity extends AppCompatActivity {
SQLiteDatabase db;
Button createData;
@Override
 protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
 createData = (Button)findViewById(R.id.button);
createData.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
db = openOrCreateDatabase(“student", Context.MODE_PRIVATE, null);
 } }); }
```

- **Table creation**

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class.

eg:

```
db.execSQL("CREATE TABLE "+stud_info +" (ID INTEGER PRIMARY KEY AUTOINCREMENT, ROLL_NO TEXT, NAME TEXT)");
```

Insertion

```
public boolean insertData(String id,String roll_no, String name){
SQLiteDatabase db = this.getWritableDatabase();
      ContentValues contentValues = new ContentValues();

      contentValues.put(COL_2, roll_no);
      contentValues.put(COL_3, name);

      long result= db.insert(stud_info,null, contentValues);
if(result == -1){
         return false; }
      else{
         return true; }
```

Database - Fetching

retrieve anything from database using an object of the Cursor class.

rawQuery :it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Eg: //1st parameter is query , 2nd is use to pass value in where clause

```
Cursor resultSet = db.rawQuery("Select * from employee", null);   resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

Steps to access sqlite database

1) Run the Code
2) Go to Views
3) In Views Select Tool Windows
4) In Tool Windows Select Device File Explorer
5) In Device File Explorer go to data
6) In data again go to data
7) Now In data Select the name package of the Project
8) If the database has been created than in that folder a databases folder will appear
9) In the databases folder you will see the database that has been created
10) Right Click on .db file and Select Save As
11) Save the .db file
12) Now go to https://sqliteonline.com/ and Select on File
13) In File Select Open DB and Choose the .db file that you have saved
14) Now you can View the Database and its data