20CS605

Hall Ticket Number:

III/IV B. Tech (Regular) DEGREE EXAMINATION

July/August, 2023 — **Computer Science & Engineering**

Sixth Semester — **Mobile Application Development**

Time: Three Hours — **Maximum:** 70 Marks

Answer question 1 compulsory. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

			CO	BL	M
1	a)	List any two challenges of Android app development.	CO1	L1	1M
		Compatibility, performance, security, limited resources.			
	b)	List types of android applications.	CO1	L1	1M
		Social media apps, gaming apps, ecommerce apps, educational apps, health apps, finance apps.			
	c)	Define android.	CO1	L2	1M
		Android is a mobile platform which consists of middleware, operating system and some key applications.			
	d)	What is an activity?	CO2	L1	1M
		An activity is one screen of an app. In that way the activity is very similar to a window in the Windows operating system. The most specific block of the user interface is the activity.			
	e)	List the building blocks of android.	CO2	L2	1M
		Activity, Broadcast receiver, content provider, widgets and live wallpapers, intent, notifications, services.			
	f)	What is the importance of having an emulator within the Android environment?	CO2	L1	1M
		An emulator is crucial in Android app development as it simulates a virtual Android device on a developer's computer. It enables developers to test, debug, and optimize their apps without needing physical devices.			
	g)	What is the importance of settings permissions in app development?	CO3	L1	1M
		Settings permission plays a crucial role in android app development which ensures the security of users and sometimes some apps may require to set permissions for that app in order to function properly.			
	h)	What is the function of an intent filter?	CO3	L1	1M
		In Android app development, an Intent Filter is a component used to declare the types of intents that a component can handle. It defines the types of actions, data, and categories that a component is capable of responding to.			
	i)	When is the onStop() method invoked?	CO3	L2	1M
		When the app goes from foreground state to background state then the onStop() is called.			

	j)	What is an intent.	CO3	L1	1M

Intent is used to perform actions on the screen. It is mainly used to start an activity, start services and transfer the data between the activities.

	k)	What is Android Database?	CO4	L1	1M

In Android development, a database is a structured collection of data that an app uses to store, retrieve, and manage information efficiently.

	l)	What is the use of content provider.	CO4	L2	1M

Content provider acts as a medium between the database and our application. It transfers the data from database to our required application.

	m)	What is a service.	CO4	L2	1M

Service is a basic building block of android which is used to perform long running tasks without any UI(User Interface).

	n)	What is a bound service?	CO4	L1	1M

A bond service is used to bind our components with services. A bound service in Android is a communication mechanism that allows components to bind to a service and interact with it, enabling data sharing and method invocation between the service and its clients in a controlled manner.

Unit-I

2	a)	Explain about Android SDK features.	CO1	L2	6M

ANDROID SDK FEATURES

The true appeal of Android as a development environment lies in its APIs. Android gives you the opportunity to create applications that are as much a part of the phone. The following list highlights some of the most noteworthy Android features:

GSM (Global System for Mobile), EDGE(Edge is the best browser for shopping), 3G, 4G, and LTE(Long Term Evolution, LTE. It's a standard for wireless data transmission that allows you to download your favorite music, websites, and video really fast—much faster than you could with the previous technology, 3G.)

Networks for telephony or data transfer, enabling you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks .

Comprehensive APIs for location-based services such as GPS and network-based location detection.

Wi-Fi hardware access and peer-to-peer connections.

Full multimedia hardware control, including playback and recording with the camera and microphone.

Media libraries for playing and recording a variety of audio/video or still-image formats.

APIs for using sensor hardware, including accelerometers, compasses, and barometers (The digital barometer is now an important tool in many of today's smartphones. This type of digital barometer uses atmospheric pressure data to make accurate elevation readings).

Libraries for using Bluetooth and NFC hardware for peer-to-peer data transfer.

IPC (Inter process Communication) message passing.

Shared data stores and APIs for contacts, social networking, calendar, and multi-media

Background Services, applications, and processes.

Home-screen Widgets and Live Wallpaper.

The ability to integrate application search results into the system searches.

An integrated open-source HTML5 WebKit-based browser.

Mobile-optimized, hardware-accelerated graphics, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0.

Localization through a dynamic resource frame work .

An application framework that encourages the reuse of application components and the replacement of native applications.

b) Draw and explain Android System Architecture. CO1 L6 8M

Android Architecture:

Android is a mobile platform, which consists of operating system, middleware, and key applications.

To justify the definition of android, if we take any electronic device which consists of 2 things (i.e) software component & hardware component.

Software component directly interact with the hardware component.

Operating system acts as a middle layer in between software component & hardware component.

If we take an android mobile as an electronic device which contains 2 things (i.e) software component & hardware component.

Operating system is an abstraction between the software component & hardware component.

The operating system in android devices is “Linux kernel”.

The main functionalities of an operating system are:

Memory management.

Resource management. (At a time we can perform multiple operations. Operating system is going to allocate the required resources to a process.)

Driver management.

Power management.

We can develop android applications by using various languages like C, C++, Java, .NET.

Initial versions of android will support only C, C++ & .NET.

Latest versions of android will support only Java.

If we are writing the code using Java, we need to take the support of SDK.

If we are writing the code using C (or) C++, we need to take the support of NDK.

If we are writing the code using .NET, we need to take the support of Mon Android framework.

If we are writing the code using Java, we need to import some java libraries.

By using Java, we cannot interact with the lowlevel components, so we need to import some native libraries.

"Application Framework" is the main component of Android architecture. It plays a key role while developing android applications.

This framework provides infrastructure for android application development.

"Dalvic Virtual Machine (DVM)" is the other main component of Android architecture.

In android application development, we are writing the code using java and it is going to produce a ".class" file (Java byte code).

DVM takes the byte code as input and it is going to produce a light weight format called ".dex" file.

DVM follows some compression techniques and it reduces the redundant information in the classes and it is going to produce a single file. (.dex)

"dex" is termed as "Dalvic Executable Code".

Application Framework :

In application framework, we have readymade libraries.For ex, if you want to work with wifi, it is going to provide wifi manager class(you need not write huge amount of code).If you want to get list some of wifi networks there is a method called wifiManager.ConfiguredNetwork(), with this you will get total no:of wifi connected devices.

If you want to get your current position by using GPS you can get current position , if you want to get current position you need to write some code to interact with the GPS,GPS has to interact with the satellite.You need to write huge amount of code but by using Application Framework you need not take care of that one bcz if provides location Manager class. By using this class just call getPosition() method.Similarly for working with Bluetooth,we have bluetooth Manager class and for sensors,we have sensorManager class.

Therefore Application Framework provide infrastructure for Application Development.

Dalvic Virtual Machine(DVM) :

DVM is the other main component of Android Architecture . In Android Application Development,we are writing the code using java and it is going to produce a .class file(java byte code).DVM takes the bytecode as input and it is going to produce a light weight format called ".DEX" file.DVM follows some compression techniques and it reduces the redundant info in the classes and it is going to produce a single file (.DEX). DEX is termedas Dalvic Executable file.

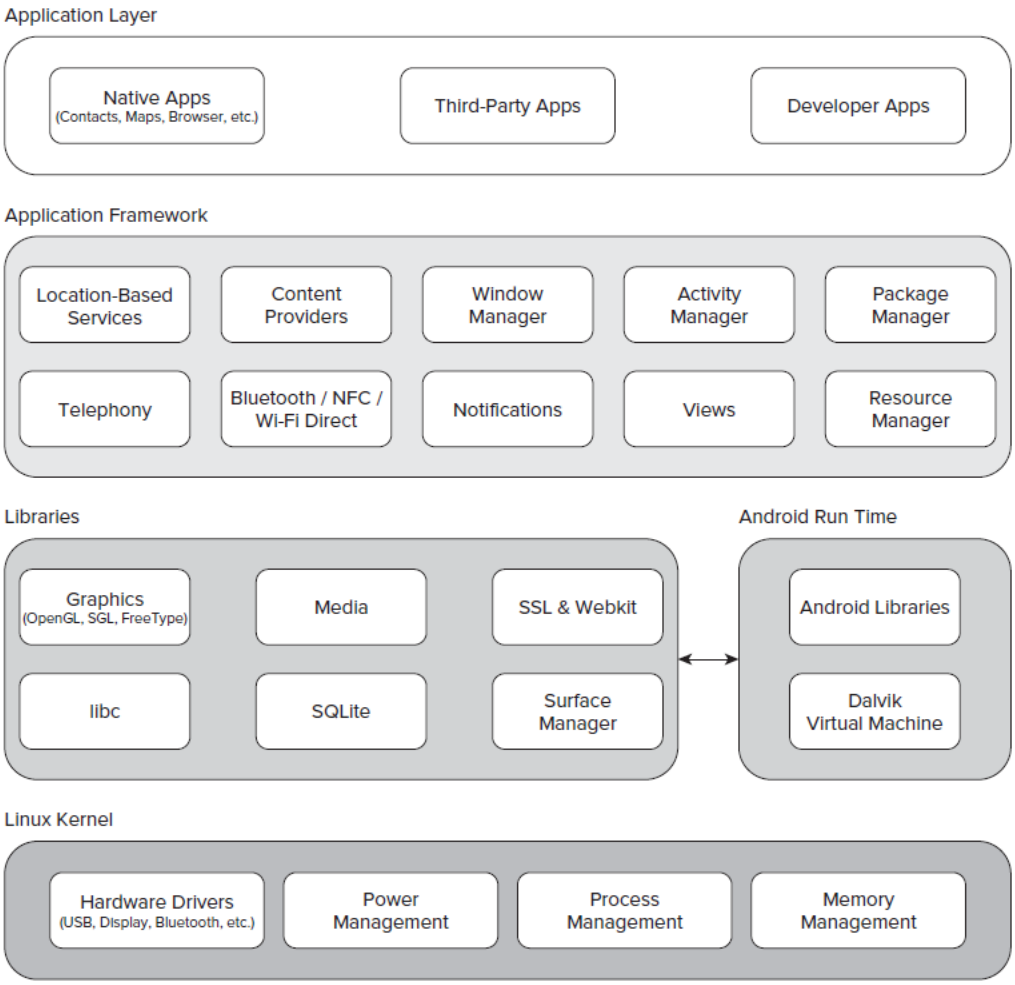

FIGURE 1-1

(OR)

3 a) Design an android app to develop interactive user interface. CO1 L3 8M

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="com.example.hellotoast.MainActivity">

   <Button
      android:onClick="showToast"
      android:layout_gravity="center"
      android:gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Show Toast"/>
   <TextView
      android:id="@+id/tv"
      android:layout_gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="0"
      android:textSize="50sp"
      />
   <Button
      android:onClick="countUp"
```

```
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="count"/>

</LinearLayout>
```

MainActivity.java

```
package com.example.hellotoast;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    int count=0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void showToast(View view) {
        Toast.makeText(this, "Hello Toast", Toast.LENGTH_SHORT).show();
    }

    public void countUp(View view) {

        count++;
        //Get the id of the TextView from activity_main.xml layout file
        TextView tV=(TextView)findViewById(R.id.tv);
        tV.setText(String.valueOf(count));//setText() method displays the text on the
TextView

        //if we want to display the same count on toast
        //Toast.makeText(this, String.valueOf(count), Toast.LENGTH_SHORT).show();

    }
}
```

b) Demonstrate a code snippet to Change the background color. CO1 L6 6M

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/ll">


    <Button
        android:id="@+id/b1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:layout_gravity="center"
```

```
        android:onClick="show"
    android:text="CHANGE BACKGROUND COLOR"/>

</LinearLayout>
```

MainActivity.java

```
package com.example.background;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Color;
import android.os.Bundle;
import android.view.View;


public class MainActivity extends AppCompatActivity {
   LinearLayout l;
   int i=1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
     l=(LinearLayout)findViewById(R.id.ll);
   }
   public void show(View view) {
          switch(i){
             case 1:
                l.setBackgroundColor(Color.parseColor("#
        0000FF"));
                          i++;
                break;
             case 2:
                l.setBackgroundColor(Color.parseColor("#
        FF00FF"));
                          i++;
                break;
             case 3:
                          l.setBackgroundColor(Color.parseColor("#
        FF0000"));
                          i++;
                break;
             case 4:
                          l.setBackgroundColor(Color.parseColor("#
        0000FF"));
                          i=1;
                break;
         }
   }

}
```

Unit-II

4 a) What is use of AndroidManifest.xml file in any android application? Explain with an example. CO2 L2 7M

The AndroidManifest.xml file is a crucial component in Android app development. It serves as a metadata descriptor for your app, providing essential information to the Android operating system about your app's components, permissions, configuration, and more. Here's a concise overview of the uses of the Android manifest file:

App Identification: The manifest file contains information that uniquely identifies your app to the Android system, including the app's package name, version code, and version name.

App Components Declaration: It lists all the app's components, such as activities, services, broadcast receivers, and content providers, along with their attributes and properties.

Activity Configuration: The manifest specifies the entry points of your app through activities. It defines which activity should be launched when the app is started and how different activities are related to each other.

Intent Filters: The manifest file contains intent filters for activities, broadcast receivers, and content providers. These filters define what kinds of intents each component can handle, allowing other apps and the system to interact with your app.

Permissions: The manifest declares the permissions your app requires to access sensitive data or perform specific actions. It helps users understand what data and resources the app will use and request their permission to access them.

App Permissions: For Android 6.0 (API level 23) and higher, apps must explicitly request dangerous permissions at runtime. The manifest file lists the permissions that your app requires and provides a basis for explaining to users why these permissions are needed.

Hardware and Software Requirements: You can specify hardware features or software requirements your app needs to function properly, ensuring compatibility with a range of devices.

App Theme and Styles: The manifest can specify the theme and styles used by your app's activities, influencing the overall look and feel of the user interface.

Application Icon and Label: You define the app's icon and user-friendly label in the manifest, which are displayed on the device's launcher screen and settings.

Configuration Changes Handling: You can indicate how your app handles configuration changes like screen orientation or keyboard availability, ensuring smooth user experiences during device rotations or other changes.

App's Process Name: You can define a custom process name for your app, which might be useful for debugging or for specifying a separate process for certain components.
Example:

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

<application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">

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

      <receiver android:name=".MyBroadcastReceiver">
         <intent-filter>
            <action
android:name="android.provider.Telephony.SMS_RECEIVED"></action>
         </intent-filter>
      </receiver>
   </application>
</manifest>
```

b) What are view and view group classes in android? Why layouts are created using xml file? Explain with an example. CO2 L4 7M

A View is a component of the user interface (UI) like a button or a text box, and it's also called a widget.
Ex:Button,TextView,EditText.

A ViewGroup is a layout, that is, a container of other Views and ViewGroups
Ex: Linear Layout, Relative Layout.

Creating layouts using XML files in Android offers a structured, readable, and maintainable way to define the user interface of an app. This approach promotes separation of concerns, supports collaboration between developers and designers, facilitates localization, and enables efficient runtime adjustments to the UI.

Example:
activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="100dp">

        <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="30dp"
                android:text="Details Form"
                android:textSize="25sp"
                android:gravity="center"/>
</LinearLayout>

<GridLayout
        android:id="@+id/gridLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="100dp"
        android:layout_marginBottom="200dp"
        android:columnCount="2"
        android:rowCount="3">

<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="0"
        android:layout_column="0"
        android:text="Name"
        android:textSize="20sp"
        android:gravity="center"/>

<EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_margin="10dp"
        android:layout_row="0"
        android:layout_column="1"
        android:ems="10"/>

<TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="1"
        android:layout_column="0"
        android:text="Reg.No"
        android:textSize="20sp"
        android:gravity="center"/>

<EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="1"
        android:layout_column="1"
        android:inputType="number"
        android:ems="10"/>

<TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="2"
        android:layout_column="0"
        android:text="Dept"
        android:textSize="20sp"
        android:gravity="center"/>

<Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="2"
        android:layout_column="1"
        android:spinnerMode="dropdown"/>
</GridLayout>

<Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerInParent="true"
        android:layout_marginBottom="150dp"
        android:text="Submit"/>
</RelativeLayout>
```

Activity_second.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.devang.exno2.SecondActivity"
android:orientation="vertical"
android:gravity="center">

<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="New Text"
        android:textSize="30sp"/>

<TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="New Text"
        android:textSize="30sp"/>

<TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="New Text"
        android:textSize="30sp"/>
</LinearLayout>
```

MAINACTIVITY.JAVA:

```
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity
{
        EditText e1,e2;
        Button bt;
        Spinner s;
        String [] dept_array={"CSE","ECE","IT","Mech","Civil"};
        String name,reg,dept;

        @Override
        protected void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                e1=(EditText)findViewById(R.id.editText);
                e2=(EditText)findViewById(R.id.editText2);
                bt=(Button)findViewById(R.id.button);
                s=(Spinner)findViewById(R.id.spinner);
                ArrayAdapter adapter= new ArrayAdapter
                (MainActivity.this,
                android.R.layout.simple_spinner_item,
                dept_arra s.setAdapter(adapter);
                bt.setOnClickListener(new View.OnClickListener()
                {
```

```
                @Override
                public void onClick(View v)
                {
                        //Getting the Values from Views(Edittext & Spinner)
                        name=e1.getText().toString();
                        reg=e2.getText().toString();
                        dept=s.getSelectedItem().toString();
                        Intent i = new
Intent(MainActivity.this,SecondActivity.class);
                        i.putExtra("name_key", name);
                        i.putExtra("reg_key",reg);
                        i.putExtra("dept_key", dept);
                        startActivity(i);
                }
        });
    }
}
```

SECONDACTIVITY.JAVA:

```
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class SecondActivity extends AppCompatActivity
{
    TextView t1,t2,t3;
    String name,reg,dept;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        t1= (TextView) findViewById(R.id.textView1);
        t2= (TextView) findViewById(R.id.textView2);
        t3= (TextView) findViewById(R.id.textView3);

        Intent i = getIntent();
        name=i.getStringExtra("name_key");
        reg=i.getStringExtra("reg_key");
        dept=i.getStringExtra("dept_key");
        //Setting the Values to Intent
        t1.setText(name);
        t2.setText(reg);
        t3.setText(dept);
    }
}
```

(OR)

5	a)	Explain Fragment Activity and Fragment Manager, and demonstrate their activities with a small piece of code.	CO2	L2	8M

Activity:
An Activity is a fundamental building block of an Android application. It represents a single, focused screen that the user can interact with. Activities manage the user interface and handle user interactions. An app can have multiple activities that work together to provide a complete user experience. Each activity has its own lifecycle, and they can be used to switch between different screens or functionalities within an app.

Fragment:
A Fragment is a modular and reusable UI component that represents a portion of an activity's user interface or behavior. Fragments are used to create flexible and responsive user interfaces, especially for larger screen sizes or devices with varying orientations. Fragments can be combined within a single activity to create multi-pane layouts, and they have their own lifecycle. They can be thought of as "sub-activities" that can be added, removed, or replaced within an activity's layout dynamically.

FragmentManager:
FragmentManager is a class in Android that manages the fragments associated with an activity. It's responsible for handling the transactions involving fragments, such as adding, removing, replacing, and showing fragments within an activity. It also manages the back stack, which allows users to navigate through the previously added fragments by pressing the back button. The FragmentManager ensures proper management of fragment lifecycles and UI changes.

Example:

```
public class MainActivity extends AppCompatActivity {

        @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
        bn= (Button)findViewById(R.id.bn);


  bn.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {

FragmentManager fragmentManager=getFragmentManager();
FragmentTransaction fragmentTransaction=FragmentTransaction.BeginTransaction();
if(!status)
{
        FragmentOne f1=new FragmentOne();
        fragmentTransaction.add(R.id.fragment_container,f1);
    fragmenttransaction.addToBackStack(null);
    fragmenttransaction.commit();
    bn.setText("Load second fragment");
    status=true;
}
else
{
        FragmentTwo f2=new FragmentTwo();
        fragmentTransaction.add(R.id.fragment_container,f2);
        fragmenttransaction.addToBackStack(null);
        fragmenttransaction.commit();
        bn.setText("Load first fragment");
        status=false;
}
//FragmentOne
//fragment_onelayout
<FrameLayout
android:layout_width="match_parent"
android:layout_height="400"
android:background="8F3A1E"

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Welcome To Fragment One"
    android:textSize="20sp"
```

```
        android:layout_gravity="center"
        android:gravity="center"/>
</FrameLayout>

//FragmentTwo
//fragment_two_layout
<FrameLayout
android:layout_width="match_parent"
android:layout_height="400dp"
android:background="077751"

<TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome To Fragment Two"
        android:textSize="20sp"
        android:layout_gravity="center"
        android:gravity="center"/>
</FrameLayout>

<LinearLayout
                android:paddingLeft="16dp"
                android:paddingRight="16dp"
                android:paddingTop="16dp"
                android:paddingBottom="16dp"
                android:oreintation="vertical"

<Button
        android:id="@+id/bn"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:text="Load First Fragment"
        android:layout_gravity="center-horizantal"/>


<RelativeLayout
            android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="400dp"
        android:marginTop="20dp"/>

</LinearLayout>
```

b) Enlist and explain any eight types of Android UI control components. CO2 L4 6M

Android provides a wide range of UI control components that developers can use to create interactive and visually appealing user interfaces. Here are some of the key UI control components available in Android:

Button: A simple clickable button that triggers an action when pressed.

EditText: A text input field that allows users to enter text or numbers.

TextView: A component used to display text content on the screen.

ImageView: Displays images or icons on the screen.

CheckBox: Represents a toggleable checkbox that users can select or deselect.

RadioButton: A single-choice option that is part of a radio group.

RadioGroup: Groups multiple radio buttons to create exclusive single-choice selections.

ToggleButton: A switch that toggles between two states.

SeekBar: A slider that allows users to select a value from a range.

Spinner: A dropdown menu that presents a list of options for selection.

ProgressBar: Displays an indicator of progress, such as loading or processing.

Switch: A two-state toggle switch.

Unit-III

6 a) Explain the following with small piece of code: CO3 L3 8M
i. Creating the Broadcast Receiver ii. Registering the Broadcast Receiver.

```
Activity_main.xml
-----------------
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="BROADCAST RECEIVER APP STARTED"
        android:textSize="18sp"/>

</LinearLayout>
```

```
MainActivity.java
------------------


public class MainActivity extends AppCompatActivity {

    private int MY_PERMISSION;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.layout_main_activty);


public class MyBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context,"Broadcast Receiver
Triggered",Toast.LENGTH_SHORT).show();
    }
}
```

```
MANIFEST.XML:

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

<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

```
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

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

        <receiver android:name=".MyBroadcastReceiver">
            <intent-filter>
                <action
android:name="android.provider.Telephony.SMS_RECEIVED"></action>
            </intent-filter>
        </receiver>


    </application>


package broadcast;

import android.content.BroadcastReceiver;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private BroadcastReceiver broadcastReceiver;


    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_main_activty);
        broadcastReceiver=new MyBroadcastReceiver();

    }

    @Override
    protected void onStart() {
        super.onStart();
        IntentFilter intentFilter=new
IntentFilter("android.provider.Telephony.SMS_RECEIVED");
        registerReceiver(broadcastReceiver,intentFilter);

    }

    @Override
    protected void onStop() {
        super.onStop();
        unregisterReceiver(broadcastReceiver);
    }
}

package broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;
```

```
public class MyBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
    Toast.makeText(context,"Broadcast Receiver
Triggered",Toast.LENGTH_SHORT).show();
    }
}
```

b) Design an android app to develop Implicit Intent. CO3 L6 6M

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/t1"
      android:text="Implicit Intent"
      android:layout_marginStart="160dp"
      android:layout_marginLeft="160dp"
       />

   <Button
      android:id="@+id/b1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="50dp"
      android:onClick="dosome"
      android:text="Open Web" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/b2"
      android:onClick="dosome"
      android:text="Open Call"/>
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/b3"
      android:onClick="dosome"
      android:text="Open Map"/>

</LinearLayout>
```

MainActivity.java

```
package com.example.myapplication1;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
```

```
import android.view.View;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void dosome(View v){
      switch(v.getId()){
         case R.id.b1:
            Intent i1=new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.google.com"));
            startActivity(i1);
            break;
         case R.id.b2:
            Intent callIntent = new Intent(Intent.ACTION_DIAL);
            callIntent.setData(Uri.parse("tel:9392309140"));
            startActivity(callIntent);
            break;
         case R.id.b3:
            Intent i3=new
Intent(Intent.ACTION_VIEW,Uri.parse("http://maps.google.com/maps?saddr=20.344,34.3
4&daddr=20.5666,45.345"));
            startActivity(i3);
            break;
      }
   }
}
```

(OR)

7 a) Demonstrate Creating and Saving of shared preferences with proper syntactic representations. CO3 L6 7M

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="com.example.ExplicitIntent.MainActivity">

   <TextView
      android:layout_gravity="center"
          android:gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Register"
      android:textSize="25dp"
      />

    <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="20dp"
      android:id="@+id/edittext_email"
          android:hint="Enter Email ID"
          />
    <EditText
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:id="@+id/edittext_name"
        android:hint="Enter Name"
        />

  <Button
        android:id="@+id/button_save"
    android:layout_margin="20dp"
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Save"/>

</LinearLayout>
```

MainActivity.java

```
package com.example.mysharedpreference;

import android.support.v7.app.AppCompatActivity ;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class MainActivity extends AppCompatActivity {

        EDITTEXT edittext_name,edittext_email;
        Button button_save;
        SharedPreferences sharedperferences;

    private static final String SHARED_PREF_NAME="MYPREF"
    private static final String KEY_NAME="name";
    private static final String KEY_EMAIL="email";
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

        editText_name=findViewById(R.id.edittext_name);
    editText_email=findViewById(R.id.edittext_email);
    button_save=findViewById(R.id.button_save);

        SharedPreferences=getSharedPreferences(SHARED_PREF_NAME,MODE_PRIV
ATE);
        String name=SharedPreferences.getString(KEY_NAME,defvalue:null);
        if(name!=null)
        {
                Intent intent=new INTENT(MainActivity.this,HOMEACTIVITY.class)
                startActivity(intent);
    }
        button_save.OnClickListener(new View.OnClickListener(){
  @Override
  public void onClick(View v)
  {
        SharedPreferences.Editor editor=SharedPreferences.Edit();
        editor.putString(KEY_NAME.editText_name.getText().toString());
    editor.putString(KEY_NAME.editText_email.getText().toString());
    editor.apply();

        Intent intent=new INTENT(MainActivity.this,HOMEACTIVITY.class)
```

```
        startActivity(intent);
            Toast.makeText(this, "LOGIN SUCCESS", Toast.LENGTH_SHORT).show();
    }
}
```

activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.ExplicitIntent.SecondActivity">

    <TextView
        android:gravity="center"
            android:textStyle="Bold"
            android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome"
        android:textSize="25dp"
        />
    <TextView
        android:id="@+id/text_fullname"
            android:gravity="center"
            android:textStyle="Bold"
            android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Full Name"
        android:textSize="18dp"
        />
    <TextView
        android:id="@+id/text_email"
            android:gravity="center"
            android:textStyle="Bold"
            android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email"
        android:textSize="18dp"
        />
   <Button
        android:id="@+id/button_logout"
            android:gravity="center"
            android:textStyle="Bold"
            android:layout_margin="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="LOG OUT"
        android:textSize="18dp"
        />
</LinearLayout>
```

SecondActivity.java

```
package com.example.mysharedpreference;

import android.support.v7.app.AppCompatActivity ;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.TextView;


public class HomeActivity extends AppCompatActivity {

   TextView textView_name,textView_email
   Button button_logout;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_second);
         textView_name=findViewById(R.id.text_fullname);
         textView_email=findViewById(R.id.text_email);
         button_logout=findViewById(R.id.button_logout);

         SharedPreferences=getSharedPreferences(SHARED_PREF_NAME,MODE_PRIV
ATE);
         String name=SharedPreferences.getString(KEY_NAME,defvalue:null);
         String email=SharedPreferences.getString(KEY_EMAIL,defvalue:null);

         if(name!=null || email!=null )
         {
                  textView_name.setText("Full Name=" +name);
                  textView_email.setText("Email ID=" +name);
         }
         button_save.OnClickListener(new View.OnClickListener(){
      @Override
      public void onClick(View v)
      {

                  SharedPreferences.Editor editor=SharedPreferences.edit();
                  editor.clear();
                  editor.commit();
                  finish();
                  Toast.makeText(this,text:"LOG out successful",
Toast.LENGTH_SHORT).show();
      }
  }
}
```

b) Explain about any four Input Controls while developing an Android application. CO3 L4 7M

Explanations of four commonly used input controls in Android application development:

EditText:
The EditText is a versatile input control that allows users to enter text or numbers. It's often used for capturing user input, such as names, passwords, emails, and more. Developers can customize the appearance of an EditText and apply input validation and formatting.

Example attributes:

android:hint: Displays a hint or placeholder text inside the EditText to guide users.
android:inputType: Defines the type of input (text, numbers, passwords, etc.).
android:maxLines: Specifies the maximum number of lines for multi-line text input.
RadioButton and RadioGroup:
RadioButton is used to present a single-choice option to the user. When multiple RadioButton controls are grouped within a RadioGroup, the user can select only one option at a time. This setup is ideal for scenarios where the user needs to choose from a set of mutually exclusive options.

Example attributes:

android:text: Sets the label for the radio button.
android:checked: Specifies whether the radio button is initially selected.
android:id: Provides a unique identifier for each radio button within the radio group.
CheckBox:
The CheckBox allows users to toggle between two states (checked and unchecked). It's used when the user can select one or more options independently. Unlike RadioButton, CheckBox controls are not mutually exclusive.

Example attributes:

android:text: Sets the label for the checkbox.
android:checked: Determines the initial checked state of the checkbox.
android:onCheckedChanged: Specifies a method to be called when the checkbox's state changes.
SeekBar:
A SeekBar is a slider control that enables users to select a value from a specified range by dragging a thumb along the slider track. It's often used for settings that involve selecting a value within a range, such as volume control.

Example attributes:

android:max: Sets the maximum value of the seek bar.
android:progress: Sets the initial progress value.
android:onSeekBarChangeListener: Defines a listener to track changes in seek bar progress.
These input controls play a significant role in creating user-friendly and interactive Android applications. By incorporating them effectively, developers can capture user input, provide options for selection, and allow users to interact with the app in meaningful ways.

Unit-IV

8 a) What are different types of Android Databases? Write different ways to host a database. CO4 L3 7M

There are several types of databases that can be used in Android development to store and manage data. Here are some common types:

SQLite Database:
SQLite is a lightweight, embedded relational database engine that comes bundled with Android. It's the most widely used database type for Android apps due to its efficiency, simplicity, and integration with the Android platform.

Room Database:
Room is a higher-level abstraction over SQLite that simplifies database operations and offers compile-time verification of SQL queries. It provides entities, DAOs (Data Access Objects), and a database layer, making it easier to work with databases while leveraging the advantages of SQLite.

Firebase Realtime Database:
Firebase Realtime Database is a cloud-hosted NoSQL database offered by Google's Firebase platform. It's suitable for real-time applications, as changes made to the database are immediately synced across connected devices.

Firebase Firestore:
Firestore is another cloud-hosted NoSQL database from Firebase. It offers more advanced querying capabilities and supports offline data synchronization.

Content Providers:
While primarily designed for data sharing between apps, content providers can also be used to manage local data storage. They provide a standardized interface to query and manipulate data.

External Databases:
Android apps can also interact with external databases hosted on remote servers. This can involve using APIs to retrieve data in JSON or XML format and parsing it to display within

the app.

Ways to Host a Database:

Local Storage: The database can be stored directly on the user's device. SQLite databases are commonly used for this purpose, providing efficient local data storage.

Cloud Hosting: Cloud-based databases are hosted on remote servers, often using cloud services like Firebase, AWS (Amazon Web Services), or Azure. This allows data to be accessed from multiple devices and enables real-time synchronization.

Remote Server: Apps can connect to remote servers running databases like MySQL, PostgreSQL, or MongoDB to retrieve and update data. RESTful APIs are commonly used to communicate between the app and the server.

Content Providers: While mainly used for data sharing between apps, content providers can also be used to expose local data to other apps or to interact with external data sources.

Local Caching: Some apps use a combination of local and cloud databases, using local storage for caching frequently accessed data and syncing it with a remote server when necessary.

Hybrid Approaches: Apps can use a combination of different databases and storage mechanisms, depending on the specific needs of the application. For instance, using SQLite for offline data storage and a cloud-based database for real-time updates.

The choice of database type and hosting approach depends on factors such as data complexity, real-time requirements, scalability, and user experience goals.

b) What is meant by Binding Services to Activities? Create a bound service with a binder class. CO4 L3 7M

Binding services to activities in Android involves establishing a connection between an activity and a service, allowing them to communicate and share data or functionality. This is achieved using the bindService() method, which creates a two-way communication channel between the activity and the service.

Example:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
     android:id="@+id/textView2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_centerHorizontal="true"
     android:layout_marginLeft="100dp"
     android:text="MY SERVICE "
     android:textColor="@color/colorPrimaryDark"
     android:textSize="50dp" />

  <Button
     android:id="@+id/btnStart"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginLeft="50dp"
```

```
        android:layout_marginTop="50dp"
        android:onClick="startService"
        android:text="Start Service" />

    <Button
        android:id="@+id/btnstop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="50dp"
        android:onClick="stopService"
        android:text="Stop Service" />
</LinearLayout>
```

MainActivity.java

```
package com.Services;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void startService(View view) {
        startService(new Intent(this, MYService.class));
    }

    public void stopService(View view) {
        stopService(new Intent(this, MYService.class));
    }
}
```

MYService.java

```
package com.Services;
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.provider.Settings;
import android.widget.Toast;

public class MYService extends Service {
    private MediaPlayer player;
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        Toast.makeText(this, "Service was Created", Toast.LENGTH_LONG).show();
```

```
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        player = MediaPlayer.create(this, Settings.System.DEFAULT_RINGTONE_URI);
        player.setLooping(true);
        player.start();
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        player.stop();
        Toast.makeText(this, "Service Stopped", Toast.LENGTH_LONG).show();
    }
}
```

(OR)

9 a) Design an android app to develop menus. CO4 L2 7M

ACTIVITY_MAIN.XML:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
 tools:context=".MainActivity">

    <TextView android:text="@string/count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:textSize="18sp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />



</RelativeLayout>
```

MAINACTIVITY.JAVA:

```
package com.example.actionbar;

import androidx.appcompat.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
```

```
    TextView count;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) { switch(item.getItemId()) {
        case R.id.add:
            count=(TextView)findViewById(R.id.textView);
            count.setText("Add is clicked");
            return(true);
        case R.id.reset:
            count=(TextView)findViewById(R.id.textView);
            count.setText("Nothing is selected");
            return(true);
        case R.id.about:
            Toast.makeText(this, R.string.about_toast, Toast.LENGTH_LONG).show();
            return(true);
        case R.id.exit:
            finish();
            return(true);

    }
        return(super.onOptionsItemSelected(item));
    }
}
```

strings.xml:

```
<resources>
    <string name="app_name">ACTIONBAR</string>
    <string name="count">Nothing is selected</string>
    <string name="action_settings">Settings</string>
    <string name="reset">Speak</string>
    <string name="about">About</string>
    <string name="exit">Exit</string>
    <string name="about_toast">About is clicked</string>
    <string name="add">Add</string>
</resources>
```

main-menu.xml:

```
<menu xmlns:android="https://schemas.android.com/apk/res/android"
    xmlns:app="https://schemas.android.com/apk/res-auto"
    xmlns:tools="https://schemas.android.com/tools" tools:context=".MainActivity">

    <item
        android:id="@+id/add" android:icon="@android:drawable/ic_menu_add"
app:showAsAction="always"    android:title="@string/add"/>
```

```
    <item
        android:id="@+id/reset" android:icon="@android:drawable/ic_menu_revert"
app:showAsAction="always|withText" android:title="@string/reset"/>
    <item
        android:id="@+id/about" android:icon="@android:drawable/ic_dialog_info"
app:showAsAction="never" android:title="@string/about">
    </item>
    <item
        android:id="@+id/exit"  app:showAsAction="never" android:title="@string/exit">
    </item>
</menu>
```

b) Explain in detail about Opening and Closing Android SQLite Database Connection. CO4 L3 7M

Opening and closing an SQLite database connection in Android is essential for efficient data access and resource management. The process involves properly initializing the database connection when needed and ensuring that it's closed when no longer required. Here's a detailed breakdown of how to open and close an SQLite database connection in Android:

Opening a Database Connection:

To open an SQLite database connection, you need to create an instance of the SQLiteDatabase class. This class provides methods for performing various database operations.

```
SQLiteDatabase database = context.openOrCreateDatabase("mydatabase.db",
Context.MODE_PRIVATE, null);
```

Executing Database Operations:

Once the database connection is open, you can perform various database operations using the methods provided by the SQLiteDatabase class. These operations include creating tables, inserting, updating, querying, and deleting data.

java
Copy code

```
// Example of creating a table
String createTableQuery = "CREATE TABLE IF NOT EXISTS users (id INTEGER
PRIMARY KEY, name TEXT)";
database.execSQL(createTableQuery);
```

Closing a Database Connection:

It's important to close the database connection when you're done using it to release system resources and prevent potential memory leaks. This is typically done in the onDestroy() method of an activity or when the application is no longer using the database.

```
database.close();
```

Closing the database connection should be done properly to avoid issues. It's also good practice to close any cursors or other resources associated with the database.

Overall, properly managing the opening and closing of SQLite database connections is crucial for maintaining the performance and stability of your Android application.