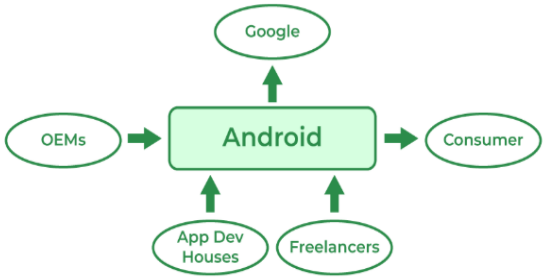
WINTER – 2024 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Mobile Application Development **Subject Code:** 22617

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**Explain the android ecosystem.**	**2 M**
	Ans	 • **Ecosystem** in Market terminology refers to the inter-dependence between demand and supply. • In the **Android ecosystem** this translates to inter-dependence between users, developers, and equipment makers. One cannot exist without the other: • **Google** develops android	Explanation 2 M

		• **Users** buy devices and applications • **Original Equipment makers** sell devices, sometimes bundled with applications • **Developers** buy devices, then make and sell applications • **Freelance Android Developer** developers have the skills to contribute to the ecosystem for android development, they are who creates their own applications and published them on googles play store.	
	b)	**Define Emulator.**	**2 M**
	Ans	An **Android emulator** is a tool that creates virtual Android devices on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device	Correct Definition 2 M
	c)	**Name two classes used to play audio and video in Android.**	**2 M**
	Ans	1) Media Player 2) Media Controller 3) Audio Manager	Any two class names 1 M each
	d)	**List any four folders from directory structure of Android project and elaborate in one line.**	**2 M**
	Ans	**Folders from directory structure:** 1)app: The App folder contains three subfolders (manifests, java and res) that make up our application. They are divided so that it should be fairly easy to determine which resources go in which folder. 2)Manifest: This is where we would put our manifest files. Most Android apps have single manifest file. But an app may have several manifest files due to application versioning, or for supporting specific hardware. 3)Java: This is the folder in our project where we will be storing all of the source code files written in Java programming language. 4)res: It contains folders that help us to separate and sort the resources of our application. Resources basically mean all the needed files except the source code. 5)drawable: The drawable folder contains graphics that can be drawn to the screen. 6)layout: The layout folder contains XML files used for your layouts. These file are used to set up the layout for your Activity and is used for basic alignment of your layouts, components, widgets, and similar resources that are used for the UI of your application. 7)mipmap : The mipmap folder contains the launcher icon files for the app. A launcher icon is a graphic that represents your app to users. 8)values: The values folder contains XML files that contain simple values, such as	List of names of any four folders: 1 M and elaboration in one line: 1 M

		strings, integers, and colors. The values folder is used to keep track of the values we will be using in our application.	
	e)	**List types of permissions in android.**	**2 M**
	Ans	1. Normal Permissions 2. Dangerous Permissions 3. Signature Permissions	Any two, 1 M for each
	f)	**List attributes of radio button. (Any Four)**	**2 M**
	Ans	• id • text • textcolor • textsize • checked • layout_width • layout_height • gravity	Any four: ½ M for each
	g)	**Describe the process of getting the map API key.**	**2 M**
	Ans	Step 1: Create a Google Cloud Project Step 2: Enable the Maps SDK for Android Step 3: Get the API Key Step 4: Get Your SHA-1 Fingerprint (for Android Apps) Step 5: Add the API Key to Your Android Project Step 6: Verify and Test the API Key	Correct steps 2 M
2.		**Attempt any THREE of the following:**	**12 M**
	a)	**Describe Android Architecture with diagram.**	**4 M**
	Ans	**1. Applications:** • The top layer of android architecture is Applications. The native and third-party applications like Contacts, Email, Music, Gallery, Clock, Games, etc. whatever we will build those will be installed on this layer only. • The application layer runs within the Android run time using the classes and services made available from the application framework.	Explanation for Android Architecture=2M and Diagram=2M

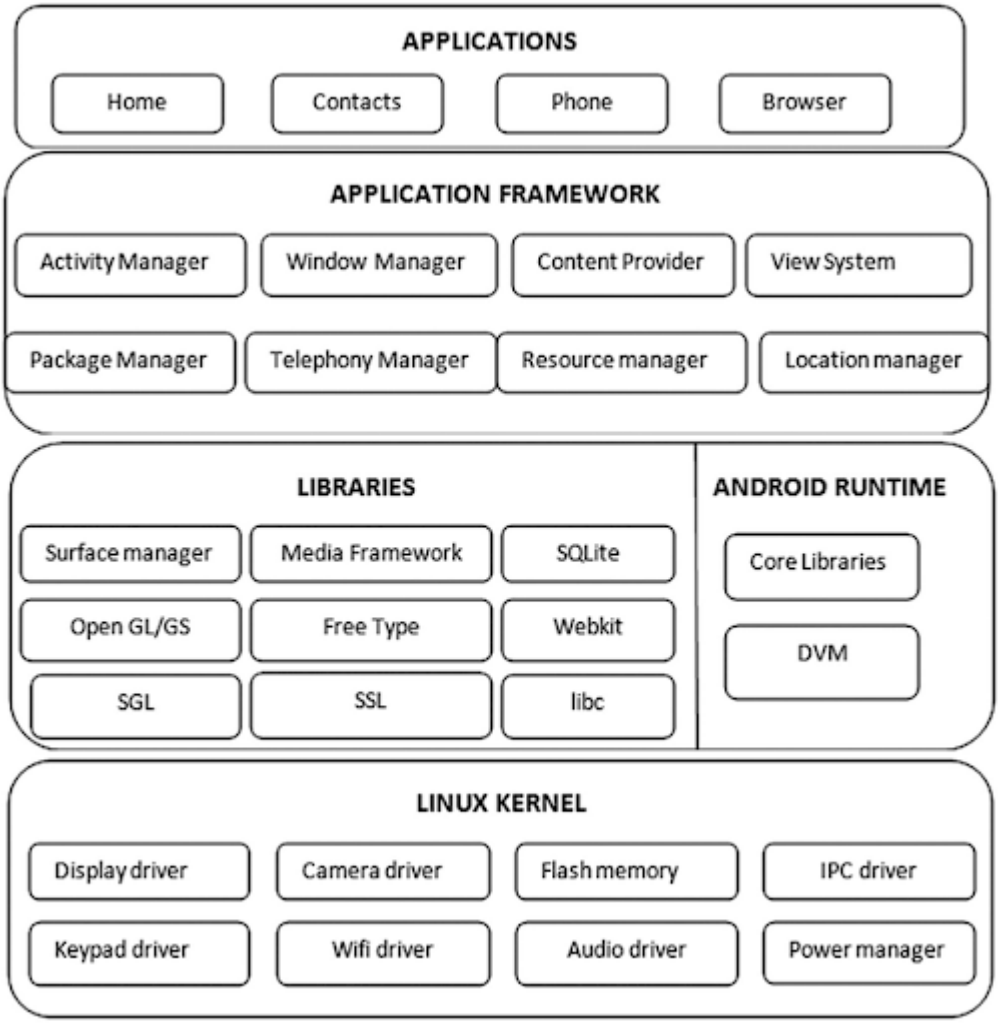


Fig: Android Architecture

2. Application Framework:

- The Application Framework provides the classes used to create an Android application. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- It basically provides the services through which we can create the particular class and make that class helpful for the Applications creation.
- The application framework includes services like telephony service, location services, and notification. manager, NFC service, view system, etc. which we can use for application development as per our requirements.

3. Android Runtime:

- Android Runtime environment is an important part of Android rather than an

		internal part and it contains a components like core libraries and the Dalvik virtual machine. • The Android run time is the engine that powers our applications along with the libraries and it forms the basis for the application framework. • Dalvik Virtual Machine (DVM) is a register-based virtual machine like Java Virtual Machine (JVM). • It is specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory • management. • The core libraries in android runtime will enable us to implement an android applications using standard JAVA programming language. **4. Platform Libraries:** • The Platform Libraries includes various C/C++ core libraries and Java based libraries such as SSL,libc, Graphics, SQLite, Webkit, Media, Surface Manger, OpenGL etc. to provide a support for android development. **5. Linux Kernel:** • Linux Kernel is a bottom layer and heart of the android architecture. It is heart of Android architecture that exists at the root of android architecture and contains all the low-level device drivers for the various hardware components of an Android device. • Linux Kernel is responsible fro device drivers, power management, memory management, device management and resource access. It manage all the drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are mainly required for the android device during the runtime. • The Linux Kernel will provide an abstraction layer between the device hardware and the remainder of the stack. It is responsible for memory management, power management, device management, resource access, etc.	
	b)	**Explain relative layout with suitable example.**	**4 M**
	Ans	A RelativeLayout is a type of **ViewGroup** that allows you to position its child views relative to each other or to the parent container. This layout is powerful because it lets you align views based on rules such as being above, below, to the left, to the right, or centered relative to other views or the parent layout.	Explanation 2 M, Example 2 M (Consider any suitable example)

Attributes Used in RelativeLayout:

- **android:layout_alignParentTop**: Aligns the view to the top of the parent.
- **android:layout_alignParentLeft** or **android:layout_alignParentStart**: Aligns the view to the left or start of the parent.
- **android:layout_alignParentRight** or **android:layout_alignParentEnd**: Aligns the view to the right or end of the parent.
- **android:layout_below**: Places the view below another view.
- **android:layout_above**: Places the view above another view.
- **android:layout_toLeftOf** or **android:layout_toStartOf**: Places the view to the left or start of another view.
- **android:layout_toRightOf** or **android:layout_toEndOf**: Places the view to the right or end of another view.
- **android:centerHorizontal**: Centers the view horizontally.
- **android:centerVertical**: Centers the view vertically.
- **android:layout_centerInParent**: Centers the view both horizontally and vertically within the parent layout.

Example of RelativeLayout:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, RelativeLayout!"
      android:textSize="20sp"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"/>
   <Button
```

		android:id="@+id/button1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Click Me" **android:layout_below="@id/textView1"** **android:layout_centerHorizontal="true"**/> <ImageView android:id="@+id/imageView1" android:layout_width="100dp" android:layout_height="100dp" android:src="@drawable/ic_launcher_foreground" **android:layout_below="@id/button1"** **android:layout_centerHorizontal="true"**/> </RelativeLayout>	
	c)	**Design a student registration form.**	**4 M**
	Ans	<?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:padding="16dp"> <TextView android:id="@+id/formTitle" android:layout_width="wrap_content" android:layout_height="wrap_content"	Any Correct Logic of XML code: 4 M

```
    android:text="Student Registration Form"
    android:textSize="24sp"
    android:textColor="#000000"
    android:layout_gravity="center"
    android:layout_marginBottom="20dp"/>
<TextView
    android:id="@+id/labelName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Full Name:"
    android:textSize="16sp"
    android:layout_marginBottom="8dp"/>

<EditText
    android:id="@+id/etName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Full Name"
    android:inputType="textPersonName"
    android:layout_marginBottom="16dp"/>

<TextView
    android:id="@+id/labelEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Email Address:"
```

```
    android:textSize="16sp"
    android:layout_marginBottom="8dp"/>

<EditText
    android:id="@+id/etEmail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter Email"
    android:inputType="textEmailAddress"
    android:layout_marginBottom="16dp"/>

<TextView
    android:id="@+id/labelGender"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Gender:"
    android:textSize="16sp"
    android:layout_marginBottom="8dp"/>

<RadioGroup
    android:id="@+id/radioGroupGender"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="16dp">
```

```
    <RadioButton
        android:id="@+id/rbMale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Male"
        android:layout_marginEnd="16dp"/>

    <RadioButton
        android:id="@+id/rbFemale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Female"
        android:layout_marginEnd="16dp"/>

    <RadioButton
        android:id="@+id/rbOther"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Other"/>
</RadioGroup>

  <TextView
   android:id="@+id/labelCourse"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Select Course:"
```

```
    android:textSize="16sp"
    android:layout_marginBottom="8dp"/>

  <Spinner
    android:id="@+id/spinnerCourse"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp" />

    <Button
    android:id="@+id/btnSubmit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Submit"
    android:textSize="18sp"
    android:layout_marginTop="16dp" />

</LinearLayout>
```

	d)	**Describe the steps for publishing android app.**	**4 M**
	Ans	**Step 1: Create a Developer Account** • Before you can publish any app on Google Play, you need to create a Developer Account. You can easily sign up for one using your existing Google Account. • You'll need to pay a one-time registration fee of $25 using your international credit or debit card. It can take up to 48 hours for your registration to be fully processed. **Step 2: Plan to Sell? Link Your Merchant Account** • If you want to publish a paid app or plan to sell in-app purchases, you need to	Correct Steps 4 M

create a payments center profile, i.e. a merchant account.

- A merchant account will let you manage your app sales and monthly payouts, as well as analyze your sales reports right in your Play Console.

Step 3: Create an App

- Now you have create an application by clicking on 'Create Application'. Here you have to select your app's default language from the drop-down menu and then type in a title for your app. The title of your app will show on Google Play after you've published.

Step 4: Prepare Store Listing

- Before you can publish your app, you need to prepare its store listing. These are all the details that will show up to customers on your app's listing on Google Play. You not necessarily complete it at once , you can always save a draft and revisit it later when you're ready to publish.
- The information required for your store listing is divided into several categories such as Product Details containing title, short and full description of the app, Your app's title and description should be written with a great user experience in mind. Use the right keywords, but don't overdo it. Make sure your app doesn't come across as spam-y or promotional, or it will risk getting suspended on the Play Store.
- Graphic Assets where you can add screenshots, images, videos, promotional graphics, and icons that showcase your app's features and functionality.
- Languages & Translations, Categorization where in category can be selected to which your app belong to. Contact Details , Privacy Policy for apps that request access to sensitive user data or permissions, you need to enter a comprehensive privacy policy that effectively discloses how your app collects, uses, and shares that data.

Step 5: Upload APK to an App Release

- Finally upload your app, by uploading APK file. Before you upload APK, you need to create an app release. You need to select the type of release you want to upload your first app version to.
- You can choose between an internal test, a closed test, an open test, and a production release. The first three releases allow you to test out your app among a select group of users before you make it go live for everyone to access.
- This is a safer option because you can analyze the test results and optimize or

		fix your app accordingly if you need to before rolling it out to all users. • Once you create a production release, your uploaded app version will become accessible to everyone in the countries you choose to distribute it in and click on 'Create release.' **Step 6: Provide an Appropriate Content Rating** • If you don't assign a rating to your app, it will be listed as 'Unrated'. Apps that are 'Unrated' may get removed from Google Play. • To rate your app, you need to fill out a content rating questionnaire An appropriate content rating will also help you get to the right audience, which will eventually improve your engagement rates. **Step 7: Set Up Pricing & Distribution** • Before you can fill out the details required in this step, you need to determine your app's monetization strategy. Once you know how your app is going to make money, you can go ahead and set up your app as free or paid. • You can always change your app from paid to free later, but you cannot change a free app to paid. For that, you'll need to create a new app and set its price. **Step 8: Rollout Release to Publish Your App** • The final step involves reviewing and rolling out your release after making sure you've taken care of everything else. • Before you review and rollout your release, make sure the store listing, content rating, and pricing and distribution sections of your app each have a green check mark next to them. • Once you're sure about the correctness of the details, select your app and navigate to 'Release management' – 'App releases.' • You can always opt for reviews by clicking on 'Review' to be taken to the 'Review and rollout release' screen. Here, you can see if there are any issues or warnings you might have missed out on. • Finally, select 'Confirm rollout.' This will also publish your app to all users in your target countries on Google Play.	
3.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Difference between JVM and DVM any four point.**	**4 M**

	Ans		4 correct points- 4M

DVM	JVM
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs the ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance	A single instance of JVM is shared with multiple applications.
DVM supports the Android operating system only.	JVM supports multiple operating systems.
There is a constant pool for every application.	It has a constant pool for every class.
Here the executable is APK.	Here the executable is JAR

	b)	**Describe sensors use in Android.**	**4 M**
	Ans	Most of the android devices have built-in sensors that measure motion, orientation, and various environmental condition. The android platform supports three broad categories of sensors. 1. Motion Sensors These are used to measure acceleration forces and rotational forces along with three axes. 2. Environmental sensors These are used to measure the environmental changes such as temperature, humidity etc. 3. Position sensors These are used to measure the physical position of device. Example: TYPE_ACCELE_ROMETER TYPE_GRAVITY TYPE_LIGHT TYPE_ORIENTATION TYPE_PRESSURE Some of the sensors are hardware based and some are software based sensors. Whatever the sensor is, android allows us to get the raw data from these sensors and use it in our application.	1 M for each sensor's use

		Android provides SensorManager and Sensor classes to use the sensors in our application. 1) **SensorManager class** The android.hardware.SensorManager class provides methods : o to get sensor instance, o to access and list sensors, o to register and unregister sensor listeners etc. You can get the instance of SensorManager by calling the method getSystemService() and passing the SENSOR_SERVICE constant in it. SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE); 2) **Sensor class** The android.hardware.Sensor class provides methods to get information of the sensor such as sensor name, sensor type, sensor resolution, sensor type etc. 3**) SensorEvent class** Its instance is created by the system. It provides information about the sensor. 4) **SensorEventListener interface** It provides two call back methods to get information when sensor values (x,y and z) change or sensor accuracy changes.	
	c)	**Develop a program to implement - List View of 6 item.**	**4 M**
	Ans	**1. XML Layout (res/layout/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:padding="16dp"> <ListView android:id="@+id/listView" android:layout_width="match_parent"	2 M for xml, 2 M for Java code

android:layout_height="wrap_content" />

</LinearLayout>

2. **Java Code (MainActivity.java)**

```
package com.example.listviewdemo;

import android.os.Bundle;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);

      ListView listView = findViewById(R.id.listView);


      String[] items = {"Item 1", "Item 2", "Item 3", "Item 4", "Item 5", "Item 6"};

      ArrayAdapter<String> adapter = new ArrayAdapter<>(

            this,

            android.R.layout.simple_list_item_1,

            items

      );

      listView.setAdapter(adapter);

   }

}
```

	d)	**Explain importance of developer console in android application development.**	**4 M**
	Ans	**1. Error Debugging:** The console logs runtime errors, crashes, and warnings. It provides detailed stack traces, making it easier to identify and fix issues during development.	1 M for each importance

		2. Real-Time Logs: It displays real-time application logs, allowing developers to track events, actions, and method calls while the app is running. **3. Performance Monitoring:** Helps measure memory usage, CPU activity, and network operations to optimize the app's performance. **4. Debugging UI Issues:** The console highlights UI-related errors like invalid layouts or missing resources, helping ensure proper user interface design. **5. Testing and Feedback:** Supports testing by showing outputs of log messages (Log.d, Log.e, etc.), making it easier to verify functionality and behavior during development.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Write a steps to install android studio.**	**4 M**
	Ans	**Pre-Installation Check List** 1. Before installing Android SDK, there is need to install Java Development Kit (JDK). Ensure that JDK is at or above 1.8. 2. Uninstall older version(s) of "Android Studio" and "Android SDK", if any. **We need to install two packages:** 1. Android Studio (IDE), which is an Integrated Development Environment (IDE) 2. Android SDK (Software Development Kit) for developing and running Android apps. **Steps to install Android studio:** Download Android Studio 1. Click Download Android Studio. The Terms and Conditions page with the Android Studio License Agreement opens. 2. Read the License Agreement. 3. At the bottom of the page, if you agree with the terms and conditions, select the I have read and agree with the above terms and conditions checkbox. 4. Click Download Android Studio to start the download. 5. When prompted, save the file to a location where you can easily locate it, such as the Downloads folder. 6. Wait for the download to complete. **Install Android Studio** a) Open the folder where you downloaded and saved the Android Studio installation file.	1 M for each correct step

b) Double-click the downloaded file.
c) If you see a User Account Control dialog about allowing the installation to make changes to your computer, click Yes to confirm the installation.
d) Click Next to start the installation.
e) Accept the default installation settings for all steps.
f) Click finish when installation is done.

Installing Android SDK

Within Android Studio, you can install the Android SDK as follows:
1. Click Tools > SDK Manager.
2. In the SDK Platforms tab, select Android Tiramisu Preview.
3. In the SDK Tools tab, select Android SDK Build.
4. Click OK to install the SDK.

	b)	**List UI component explain any one with help of example.**	**4 M**
	Ans	• TextView • EditText • Button • ImageView • ListView • CheckBox • RadioButton • ProgressBar **Example Code:** **XML Layout** (activity_main.xml): <Button android:id="@+id/myButton" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Click Me" /> **Java Code** (MainActivity.java): package com.example.uicomponents;	2 M for Listing , 2 M for Example code XML or Java Code

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button myButton = findViewById(R.id.myButton);
        myButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "Button Clicked!",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

** Note: Consider any suitable example.*

	c)	**Develop a program for providing bluetooth connectivity.**	**4 M**
	Ans	**1. Add Permissions** **XML file**	3 M for program, 1 M for permission

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:padding="40dp"
    android:orientation="horizontal"
```

```
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Bluetooth"
        android:id="@+id/text"
        android:textSize="20dp"
        android:gravity="center"/>

    <Button
        android:id="@+id/on"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/text"
        android:layout_marginTop="62dp"
        android:text="ON" />

    <Button
        android:id="@+id/discoverable"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/on"
        android:layout_marginTop="74dp"
        android:text="DISCOVERABLE" />

    <Button
        android:id="@+id/off"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/discoverable"
        android:layout_marginTop="104dp"
        android:text="OFF" />

</RelativeLayout>
```

Menifest File:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ifcdiv">
   <uses-permission android:name="android.permission.BLUETOOTH"/>
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Java File

```
package com.example.ifcdiv;
import androidx.appcompat.app.AppCompatActivity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.os.Bundle;
```

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

public class MainActivity extends AppCompatActivity {
Button on,off,dis;
int REQUEST_ENABLE=0;
int REQUEST_DIS=0;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      on=findViewById(R.id.on);
      off=findViewById(R.id.off);
      dis=findViewById(R.id.discoverable);

     BluetoothAdapter bluetoothAdapter=BluetoothAdapter.getDefaultAdapter();

      on.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
           if(!bluetoothAdapter.isEnabled())
           {
         Intent i=new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
           startActivityForResult(i,REQUEST_ENABLE);
           }
         }
      });

      dis.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if(!bluetoothAdapter.isDiscovering())
            {
    Intent i=new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
      startActivityForResult(i,REQUEST_DIS);
            }
         }
      });

      off.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
           bluetoothAdapter.disable();
         }
      });
   }
}
```

	d)	**Explain Geocoding and reverse geocoding in android with example.**	**4 M**
	Ans	**Geocoding:** Converts an address (like "123 Main St, Mumbai") into geographical coordinates (latitude and longitude). **Reverse Geocoding:** Converts geographical coordinates into a readable address (like "123 Main St, Mumbai"). **Example: Geocoding and Reverse Geocoding** **1. Add Permissions** Include location permissions in the AndroidManifest.xml: <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> **2. Layout File (res/layout/activity_main.xml)** A layout with a button and two TextView elements to show results. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent" android:layout_height="match_parent" android:orientation="vertical" android:padding="16dp"> <Button android:id="@+id/btnGeocode" android:layout_width="match_parent" android:layout_height="wrap_content" android:text="Get Coordinates from Address" /> <TextView android:id="@+id/tvGeocodeResult" android:layout_width="match_parent" android:layout_height="wrap_content" android:paddingTop="10dp" /> <Button	3 M for Code, 1 M for Permissions

```
        android:id="@+id/btnReverseGeocode"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Address from Coordinates"
        android:layout_marginTop="20dp" />
    <TextView
        android:id="@+id/tvReverseGeocodeResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="10dp" />
</LinearLayout>
```

3. Java Code (MainActivity.java)

Using Geocoder to perform geocoding and reverse geocoding.

```
package com.example.geocodingdemo;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.io.IOException;
import java.util.List;
import java.util.Locale;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

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

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

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

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

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

        Geocoder geocoder = new Geocoder(this, Locale.getDefault());

        // Geocoding: Address to Coordinates

        btnGeocode.setOnClickListener(v -> {

            String address = "123 Main St, Mumbai";

            try {

                List<Address> addresses = geocoder.getFromLocationName(address, 1);

                if (addresses != null && !addresses.isEmpty()) {

                    Address location = addresses.get(0);

                    tvGeocodeResult.setText("Coordinates: " + location.getLatitude() + ", " +
location.getLongitude());

                } else {

                    tvGeocodeResult.setText("Address not found");

                }

            } catch (IOException e) {

                e.printStackTrace();

                Toast.makeText(this, "Error: " + e.getMessage(),
Toast.LENGTH_SHORT).show();

            }

        });

        // Reverse Geocoding: Coordinates to Address

        btnReverseGeocode.setOnClickListener(v -> {

            double latitude = 19.0760;  // Example: Mumbai's latitude

            double longitude = 72.8777; // Example: Mumbai's longitude
```

```
        try {
            List<Address> addresses = geocoder.getFromLocation(latitude, longitude,
1);
            if (addresses != null && !addresses.isEmpty()) {
                Address address = addresses.get(0);
                tvReverseGeocodeResult.setText("Address: " +
address.getAddressLine(0));
            } else {
                tvReverseGeocodeResult.setText("Address not found");
            }
        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText(this, "Error: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    });
  }
}
```

	e)	**Explain Service Life Cycle.**	**4 M**
	Ans	A service is an application component which runs without direst interaction with the user in the background. ● 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. ● 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" **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	1 M for diagram, 3 M for Explanation

started it is destroyed.

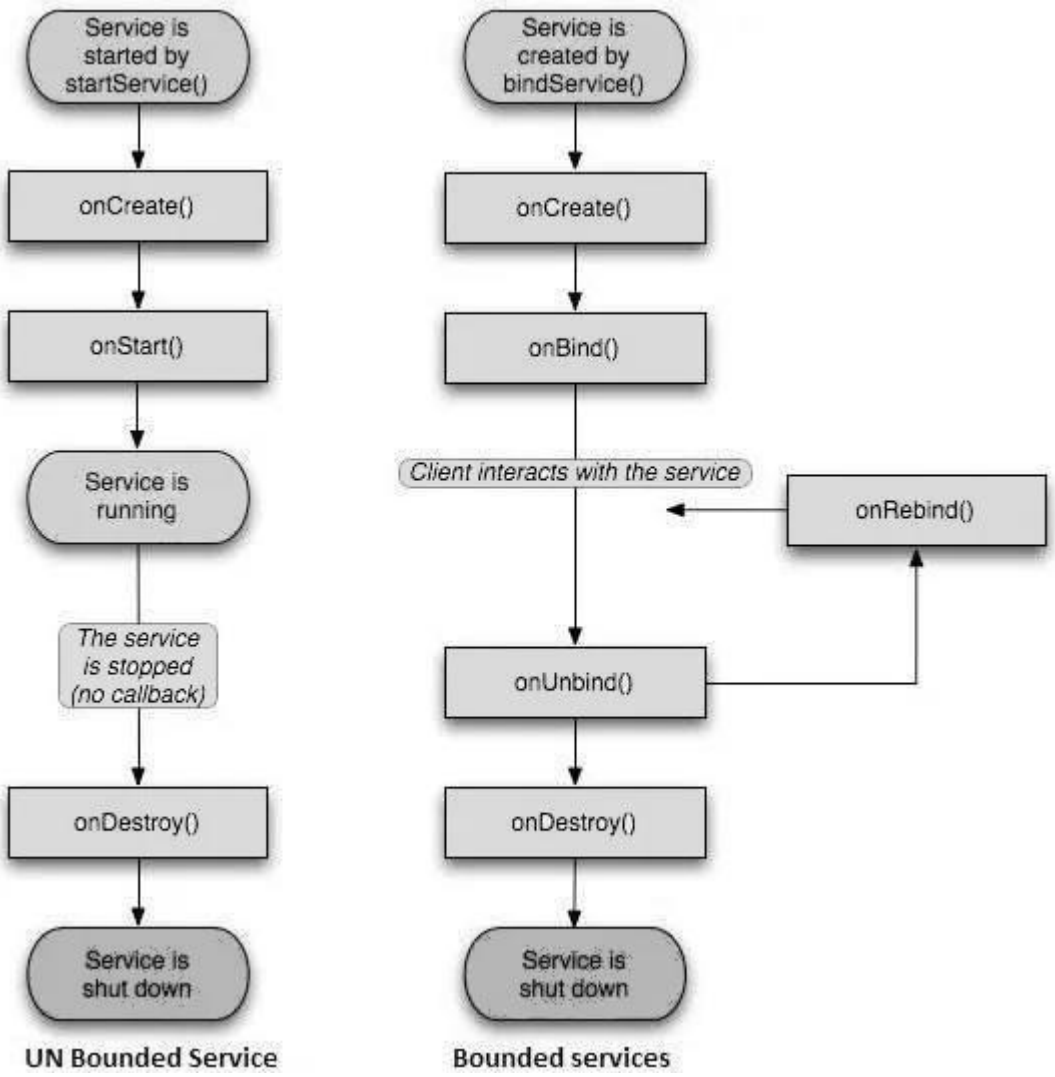


Fig: Service Lifecycle

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 Won't 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.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Develop the android application for student marksheets using table layout atleast five subject marks with total and percentage. (Write both Java and xml code)**	**6 M**
	Ans	<?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:padding="16dp">	Xml file 3 M, Java file 3M

```
<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Student Marksheet"
    android:textSize="24sp"
    android:layout_gravity="center"
    android:marginBottom="20dp" />
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1"
    android:padding="8dp">
    <!-- Subject 1 -->
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Subject 1"
            android:paddingEnd="8dp"/>
        <EditText
            android:id="@+id/subject1_marks"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
```

```
        android:hint="Marks"/>
    </TableRow>
    <!-- Subject 2 -->
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Subject 2"
            android:paddingEnd="8dp"/>
        <EditText
            android:id="@+id/subject2_marks"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
            android:hint="Marks"/>
    </TableRow>

    <!-- Subject 3 -->
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Subject 3"
            android:paddingEnd="8dp"/>
        <EditText
```

```
        android:id="@+id/subject3_marks"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="number"
        android:hint="Marks"/>
    </TableRow>

    <!-- Subject 4 -->
    <TableRow>
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Subject 4"
        android:paddingEnd="8dp"/>
      <EditText
        android:id="@+id/subject4_marks"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="number"
        android:hint="Marks"/>
    </TableRow>

    <!-- Subject 5 -->
    <TableRow>
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Subject 5"
        android:paddingEnd="8dp"/>
    <EditText
        android:id="@+id/subject5_marks"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="number"
        android:hint="Marks"/>
</TableRow>

<!-- Total and Percentage -->
<TableRow>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Total Marks"
        android:paddingEnd="8dp"/>
    <TextView
        android:id="@+id/total_marks"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
            android:text="0"
            android:gravity="end"/>
    </TableRow>

    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Percentage"
            android:paddingEnd="8dp"/>
        <TextView
            android:id="@+id/percentage"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="0%"
            android:gravity="end"/>
    </TableRow>
</TableLayout>

<!-- Submit Button -->
<Button
    android:id="@+id/submit_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Calculate"
```

```
    android:layout_gravity="center"/>


</LinearLayout>
```

Java Code

```
package com.example.studentmarksheet;


import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {


   // Declare the UI components
   EditText subject1, subject2, subject3, subject4, subject5;
   TextView totalMarks, percentage;
   Button calculateButton;


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

```
// Initialize the UI components
subject1 = findViewById(R.id.subject1_marks);
subject2 = findViewById(R.id.subject2_marks);
subject3 = findViewById(R.id.subject3_marks);
subject4 = findViewById(R.id.subject4_marks);
subject5 = findViewById(R.id.subject5_marks);
totalMarks = findViewById(R.id.total_marks);
percentage = findViewById(R.id.percentage);
calculateButton = findViewById(R.id.submit_button);

// Set up the calculate button click listener
calculateButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Capture the marks from the EditTexts
        try {
            int marks1 = Integer.parseInt(subject1.getText().toString());
            int marks2 = Integer.parseInt(subject2.getText().toString());
            int marks3 = Integer.parseInt(subject3.getText().toString());
            int marks4 = Integer.parseInt(subject4.getText().toString());
            int marks5 = Integer.parseInt(subject5.getText().toString());

            // Calculate the total marks and percentage
            int total = marks1 + marks2 + marks3 + marks4 + marks5;
            double percentageValue = (total / 500.0) * 100;
```

```
                // Display the results
                totalMarks.setText(String.valueOf(total));
                percentage.setText(String.format("%.2f%%", percentageValue));

            } catch (NumberFormatException e) {
                // Handle the case where input is not valid
                Toast.makeText(MainActivity.this, "Please enter valid marks",
Toast.LENGTH_SHORT).show();
            }
        }
    });
  }
}
```

b) **Develop an application to store student details like roll no, name, marks and retrieve student information using roll no. in SQLite database. (Write java and xml file).** **6 M**

Ans

```
<?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:padding="16dp">

  <EditText
    android:id="@+id/editRollNo"
    android:layout_width="match_parent"
```

Xml file 3 M

Java file 3 M

```
    android:layout_height="wrap_content"
    android:hint="Roll No"
    android:inputType="number" />

<EditText
    android:id="@+id/editName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Name"
    android:inputType="text" />

<EditText
    android:id="@+id/editMarks"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Marks"
    android:inputType="number" />

<Button
    android:id="@+id/saveButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Save Student Info" />

<EditText
    android:id="@+id/searchRollNo"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Roll No to Search"
        android:inputType="number" />

    <Button
        android:id="@+id/searchButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Search Student" />

    <TextView
        android:id="@+id/studentInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Student Info will appear here"
        android:textSize="18sp"
        android:gravity="center"
        android:layout_marginTop="20dp" />

</LinearLayout>
```

Java Code

```
package com.example.studentinfo;

import android.content.ContentValues;
import android.database.Cursor;
```

```
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    // UI components
    EditText editRollNo, editName, editMarks, searchRollNo;
    Button saveButton, searchButton;
    TextView studentInfo;

    // SQLite helper class
    DBHelper dbHelper;

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

        // Initialize UI components
        editRollNo = findViewById(R.id.editRollNo);
```

```
    editName = findViewById(R.id.editName);
    editMarks = findViewById(R.id.editMarks);
    searchRollNo = findViewById(R.id.searchRollNo);
    saveButton = findViewById(R.id.saveButton);
    searchButton = findViewById(R.id.searchButton);
    studentInfo = findViewById(R.id.studentInfo);

    // Initialize DB helper
    dbHelper = new DBHelper(this);

    // Save button click listener
    saveButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            saveStudentInfo();
        }
    });

    // Search button click listener
    searchButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            searchStudentInfo();
        }
    });
}
```

```
// Save student info to the database
private void saveStudentInfo() {
    String rollNo = editRollNo.getText().toString();
    String name = editName.getText().toString();
    String marks = editMarks.getText().toString();

    if (rollNo.isEmpty() || name.isEmpty() || marks.isEmpty()) {
        Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
    } else {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(DBHelper.COLUMN_ROLL_NO, rollNo);
        values.put(DBHelper.COLUMN_NAME, name);
        values.put(DBHelper.COLUMN_MARKS, marks);

        long result = db.insert(DBHelper.TABLE_NAME, null, values);

        if (result == -1) {
            Toast.makeText(this, "Failed to save student info",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Student info saved",
Toast.LENGTH_SHORT).show();
            clearFields();
        }
    }
```

```
    }


    // Search student info from the database
    private void searchStudentInfo() {
        String rollNo = searchRollNo.getText().toString();


        if (rollNo.isEmpty()) {
            Toast.makeText(this, "Please enter roll number",
Toast.LENGTH_SHORT).show();
        } else {
            SQLiteDatabase db = dbHelper.getReadableDatabase();


            String[] projection = {
                DBHelper.COLUMN_ROLL_NO,
                DBHelper.COLUMN_NAME,
                DBHelper.COLUMN_MARKS
            };


            String selection = DBHelper.COLUMN_ROLL_NO + " = ?";
            String[] selectionArgs = {rollNo};


            Cursor cursor = db.query(DBHelper.TABLE_NAME, projection, selection,
selectionArgs, null, null, null);


            if (cursor != null && cursor.moveToFirst()) {
                String name =
cursor.getString(cursor.getColumnIndexOrThrow(DBHelper.COLUMN_NAME));
```

```
            String marks =
cursor.getString(cursor.getColumnIndexOrThrow(DBHelper.COLUMN_MARKS));

            studentInfo.setText("Roll No: " + rollNo + "\nName: " + name + "\nMarks: "
+ marks);

            cursor.close();

        } else {

            studentInfo.setText("No student found with this roll number");

        }

    }

}


// Clear input fields

private void clearFields() {

    editRollNo.setText("");

    editName.setText("");

    editMarks.setText("");

    searchRollNo.setText("");

}

}
```

c) Explain Grid view and image view with suitable example. **6 M**

Ans

GridView just works like ListView, The only difference is that GridView is used to display grid of View objects.

The view objects can be a Text view, an Image view or a view group which has both an image and some text.

Vertical and horizontal spacing between every single items of gridView can be set by verticalSpacing and horizontalSpacing.

GridView Example

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

<android.support.constraint.ConstraintLayout
```

Grid view definition 1 M

Example of grid view 2 M

Image view definition 1 M

Example of image view 2 M

```
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:background="#FFEB3B"

   tools:context="com.example.android.studytonightandroid.MainActivity">


   <GridView

      android:id="@+id/gridView"

      android:layout_width="match_parent"

      android:layout_height="match_parent"

      android:verticalSpacing="10dp"

      android:horizontalSpacing="10dp"

      android:numColumns="2"/>

      </android.support.constraint.ConstraintLayout>
```

Now we will create a new XML file, with name grid_item.xml in the layout folder, and add a TextView

```
<?xml version="1.0" encoding="utf-8"?>
 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textStyle="bold"
   android:layout_marginLeft="10dp"
   android:layout_marginTop="5dp"
   android:padding="4dp"
   android:textColor="#000000"
   />
```

Java File

```
public class MainActivity extends AppCompatActivity
 {
   GridView gridView;
   TextView textView;
   String[] carBrands = { "Ferrari", "McLaren", "Jaguar", "Skoda", "Suzuki",
```

```
    "Hyundai", "Toyota", "Renault", "Mercedes", "BMW", "Ford",
    "Honda", "Chevrolet", "Volkswagon"  };

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

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

    final ArrayAdapter adapter = new ArrayAdapter(this,
        android.R.layout.grid_item, android.R.id.textView, carBrands);

    listView.setAdapter(adapter);


  } }
```

ImageView

ImageView class is used to display an image file in application.

```
<ImageView

  android:id="@+id/simpleImageView"

  android:layout_width="fill_parent"

  android:layout_height="wrap_content"

  android:background="#000"

  android:src="@drawable/lion"

  android:padding="30dp"/>
```

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Explain date picker with suitable example.**	**6 M**
	Ans	DatePicker is used to display date selection widget in android application. It can be used in either spinner mode or calendar mode (date picker).	Date picker explanation 2 M Xml file 2 M Java file 2 M

DatePicker Properties:

DatePickerMode :
Value can be spinner or calendar. If set to calendar, it will display a calendar which let you choose date. If set to spinner, it will display a spinner to let you choose date.

XML file:

```
<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:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <DatePicker
      android:id="@+id/simpleDatePicker"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="#150"
      android:datePickerMode="spinner" />

   <Button
      android:id="@+id/submitButton"
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:layout_below="@+id/simpleDatePicker"
```

```
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"
        android:background="#150"
        android:text="SUBMIT"
        android:textColor="#fff"
        android:textSize="20sp"
        android:textStyle="bold" />
</RelativeLayout>
```

Java File:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.DatePicker;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

   DatePicker simpleDatePicker;
   Button submit;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker);
        submit = (Button) findViewById(R.id.submitButton);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String day = "Day = " + simpleDatePicker.getDayOfMonth();
                String month = "Month = " + (simpleDatePicker.getMonth() + 1);
                String year = "Year = " + simpleDatePicker.getYear();
              Toast.makeText(getApplicationContext(), day + "\n" + month + "\n" + year,
Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

	b)	**Write a program to find the direction from ·user's current location to MSBTE, Bandra (Write only java and manifest file))**	**6 M**
	Ans	AndroidManifest.xml <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.msbte.google_map_currentlocationroute"> <uses-permission android:name="android.permission.INTERNET" /> <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />	Manifest file 2 M Java file 4 M

MainActivity.java

```
import android.Manifest;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.net.Uri;
import android.provider.Settings;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
```

```
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.MultiplePermissionsReport;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.DexterError;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.PermissionRequestErrorListener;
import com.karumi.dexter.listener.multi.MultiplePermissionsListener;
import java.util.List;
public class MainActivity extends AppCompatActivity implements
GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener , OnMapReadyCallback,
TaskLoadedCallback{
 //variables for map and route
 private GoogleMap mMap;
 private MarkerOptions place1, place2;
 Button getDirection;
 private Polyline currentPolyline;
 private MapFragment mapFragment;
 private boolean isFirstTime = true;
//variables for current location
 private static final String TAG = "MainActivity";
 private TextView tvLocation;
 private GoogleApiClient mGoogleApiClient;
 private Location mLocation;
 private LocationRequest mLocationRequest;
 private com.google.android.gms.location.LocationListener listener;
```

```
private long UPDATE_INTERVAL = 2 * 1000; /* 10 secs */
private long FASTEST_INTERVAL = 2000; /* 2 sec */
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
//code for getting current location
requestMultiplePermissions();
tvLocation = (TextView) findViewById((R.id.tv));
mGoogleApiClient = new GoogleApiClient.Builder(this)
.addConnectionCallbacks(this)
.addOnConnectionFailedListener(this)
.addApi(LocationServices.API)
.build();
} //code for drawing route
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
mMap.clear();
Log.d("mylog", "Added Markers");
mMap.addMarker(place1);
mMap.addMarker(place2);
CameraPosition googlePlex = CameraPosition.builder()
.target(new LatLng(22.7739,71.6673))
.zoom(7)
.bearing(0)
.tilt(45)
.build();
mMap.animateCamera(CameraUpdateFactory.newCameraPosition(googlePlex),
```

```
5000,
null);
 }
 private String getUrl(LatLng origin, LatLng dest, String directionMode) {
 // Origin of route
 String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
 // Destination of route
 String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
 // Mode
 String mode = "mode=" + directionMode;
 // Building the parameters to the web service
 String parameters = str_origin + "&" + str_dest + "&" + mode;
 // Output format
 String output = "json";
 // Building the url to the web service
 String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" +
parameters + "&key=" + getString(R.string.google_maps_key);
 return url;
 }
 @Override
 public void onTaskDone(Object... values) {
 if (currentPolyline != null)
 currentPolyline.remove();
 currentPolyline = mMap.addPolyline((PolylineOptions) values[0]);
 }
 //runtime permission method
 private void requestMultiplePermissions(){
 Dexter.withActivity(this)
```

```
withPermissions(
 Manifest.permission.ACCESS_FINE_LOCATION,
 Manifest.permission.ACCESS_COARSE_LOCATION )
 .withListener(new MultiplePermissionsListener() {
 @Override
 public void onPermissionsChecked(MultiplePermissionsReport report) {
 // check if all permissions are granted
 if (report.areAllPermissionsGranted()) {
 Toast.makeText(getApplicationContext(), "All permissions are granted by
user!", Toast.LENGTH_SHORT).show();
 }
 // check for permanent denial of any permission
 if (report.isAnyPermissionPermanentlyDenied()) {
 // show alert dialog navigating to Settings
 openSettingsDialog();
 }
 }
 @Override
 public void onPermissionRationaleShouldBeShown(List<PermissionRequest>
permissions, PermissionToken token) {
 token.continuePermissionRequest();
 }
 }).
 withErrorListener(new PermissionRequestErrorListener() {
 @Override
 public void onError(DexterError error) {
 Toast.makeText(getApplicationContext(), "Some Error! ",
Toast.LENGTH_SHORT).show();
```

```
}
})
.onSameThread()
.check();
}
private void openSettingsDialog() {
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setTitle("Required Permissions");
builder.setMessage("This app require permission to use awesome feature. Grant them in
app settings.");
builder.setPositiveButton("Take Me To SETTINGS", new
DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
dialog.cancel();
Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
Uri uri = Uri.fromParts("package", getPackageName(), null);
intent.setData(uri);
startActivityForResult(intent, 101);
}
});
builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
dialog.cancel();
}
});
```

```
builder.show();
}
//methods for getting current location
@Override
public void onConnected(Bundle bundle) {
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
return;
}
startLocationUpdates();
mLocation =
LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
if(mLocation == null){
startLocationUpdates();
}
if (mLocation != null) {
// mLatitudeTextView.setText(String.valueOf(mLocation.getLatitude()));
//mLongitudeTextView.setText(String.valueOf(mLocation.getLongitude()));
} else {
Toast.makeText(this, "Location not Detected", Toast.LENGTH_SHORT).show();
} }
@Override
public void onConnectionSuspended(int i) {
Log.i(TAG, "Connection Suspended");
mGoogleApiClient.connect();
```

```
}
@Override
public void onConnectionFailed(ConnectionResult connectionResult) {
Log.i(TAG, "Connection failed. Error: " + connectionResult.getErrorCode());
}
@Override
protected void onStart() {
super.onStart();
if (mGoogleApiClient != null) { mGoogleApiClient.connect();
} }
@Override
protected void onStop() {
super.onStop();
if (mGoogleApiClient.isConnected()) {
mGoogleApiClient.disconnect();
} }
protected void startLocationUpdates() {
// Create the location request
mLocationRequest = LocationRequest.create()
.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
.setInterval(UPDATE_INTERVAL)
.setFastestInterval(FASTEST_INTERVAL);
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
return;
```

```
}
LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
}
@Override
public void onLocationChanged(Location location) {
String msg = "Updated Location: " +
Double.toString(location.getLatitude()) + "," +
Double.toString(location.getLongitude());
tvLocation.setText(String.valueOf(location.getLatitude() +"
"+String.valueOf(location.getLongitude())));
Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
if(isFirstTime){
//code to draw path on map
getDirection = findViewById(R.id.btnGetDirection);
getDirection.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
new FetchURL(MainActivity.this).execute(getUrl(place1.getPosition(),
place2.getPosition(), "driving"), "driving");
}
});
place1 = new MarkerOptions().position(new LatLng(location.getLatitude(),
location.getLongitude())).title("Location 1");
place2 = new MarkerOptions().position(new
LatLng(19.021824,72.8662016)).title("MSBTE");
mapFragment = (MapFragment)
getFragmentManager().findFragmentById(R.id.mapNearBy);
```

		mapFragment.getMapAsync(this);
		isFirstTime = false;
		} }}
		Note: Imports are not required in coding.	

	c)	**Develop and application to send SMS (Design minimal UI as per your choice. Write XML ,Java and manifest file).**	**6 M**

	Ans	**activity_main.xml:**
		
		MainActivity.java:
		package com.example.al_libaansapp;
		import androidx.appcompat.app.AppCompatActivity;
		import androidx.core.app.ActivityCompat;
		import androidx.core.content.ContextCompat;
		import android.Manifest;
		import android.content.pm.PackageManager;
		import android.os.Bundle;
		import android.telephony.SmsManager;
		import android.view.View;
		import android.widget.Button;
		import android.widget.EditText;
		import android.widget.Toast;
		
		public class MainActivity extends AppCompatActivity {
		SmsManager sm;
		Button send;
		EditText msg, phone;
		
		@Override
		protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		
		send = findViewById(R.id.send);
		msg = findViewById(R.id.sms);
		phone = findViewById(R.id.phone);
		
		requestMessage();
		
		send.setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View view) {
			Manifest file 1 M
			Xml file 2 M
			Java file 3 M

```
                    if(checkPermission()) {
                        sendSMS();
                    } else {
                        requestMessage();
                    }
                }
            });
        }

        public void requestMessage(){
            ActivityCompat.requestPermissions(this, new
    String[]{Manifest.permission.SEND_SMS},0);
        }
        public void sendSMS(){
            try {
                String PhoneText = phone.getText().toString();
                String MsgTxt = msg.getText().toString();

                sm = SmsManager.getDefault();
                sm.sendTextMessage(PhoneText , null, MsgTxt,null,null);
                Toast.makeText(MainActivity.this, "Message sent",
    Toast.LENGTH_SHORT).show();
            } catch (Exception e) {
                Toast.makeText(MainActivity.this, "Message not sent",
    Toast.LENGTH_SHORT).show();
            }
        }
        public boolean checkPermission(){
            if (ActivityCompat.checkSelfPermission(this,
    Manifest.permission.SEND_SMS) ==
    PackageManager.PERMISSION_GRANTED) {
                return true;
            } else return false;
        }
    }
```

AndroidManifest.xml:

- <uses-feature android:name="android.hardware.telephony" />
- <uses-permission android:name="android.permission.SEND_SMS"/>

Note: Consider the simple code.