SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE
Department of Computer
Affiliated to University of Mumbai

Design..Develop..Deploy..Deliver

Experiment No.—1:
Aim: Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals: Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple "Hello World" program.

Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand Android application components.
2. Develop basic Android applications.

THEORY : Introduction to Android

Introduction to Android: This theory encompasses the basic understanding of the Android operating system, its history, architecture, and key components such as Activities, Services, Broadcast Receivers, and Content Providers. It also includes knowledge about the Android Software Development Kit (SDK) and the Android Development Tools (ADT) used for building Android applications.

Introduction to Android Studio IDE: This theory focuses on Android Studio, the official IDE for Android development. It covers the features, tools, and interface of Android Studio, including the layout editor, code editor, debugging tools, and project management capabilities. Additionally, it may include setup and configuration steps for creating Android projects in Android Studio.

Application Fundamentals: Application Fundamentals theory covers the basic concepts and principles underlying Android app development. This includes understanding the Android application lifecycle, user interface design with XML layouts and resources, handling user interactions, managing app resources, and navigating between different app screens. It also involves knowledge of Android manifest file, permissions, and application components' interactions.

User Interface Design: This theory focuses on designing user interfaces for Android applications, including principles of Material Design, creating responsive layouts, implementing various UI components such as buttons, text views, lists, and navigation patterns like tabs, drawers, and bottom navigation.

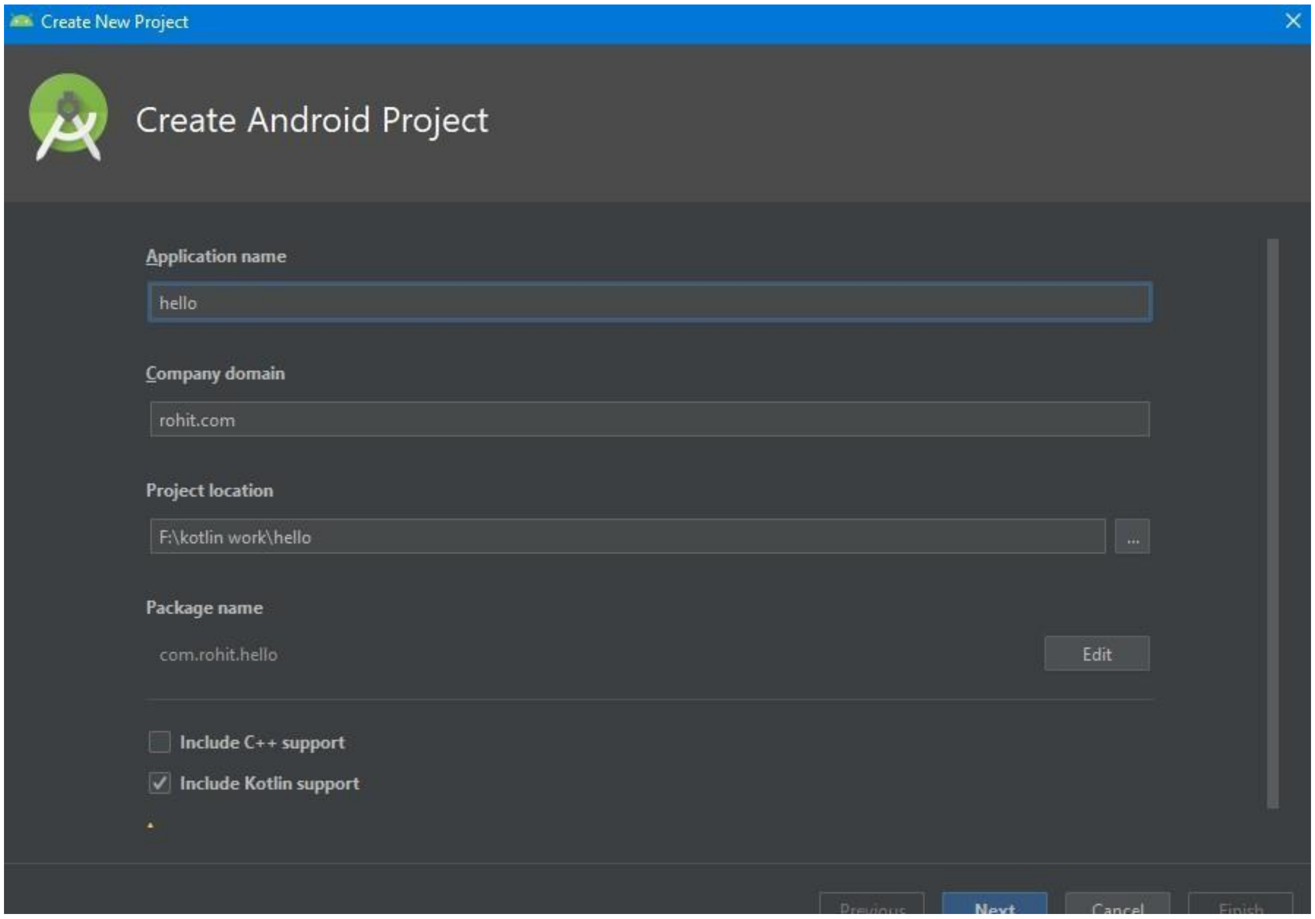
User Interface Design: This theory focuses on designing user interfaces for Android applications, including principles of Material Design, creating responsive layouts, implementing various UI components such as buttons, text views, lists, and navigation patterns like tabs, drawers, and bottom navigation.

Data Persistence: Data persistence theory covers techniques for storing and retrieving data in Android applications, including shared preferences, SQLite databases, file storage, and using libraries like Room Persistence Library for efficient database management.

Step:-

Solution:

Creating a project:


Create New Project

Create Android Project

Application name: hello

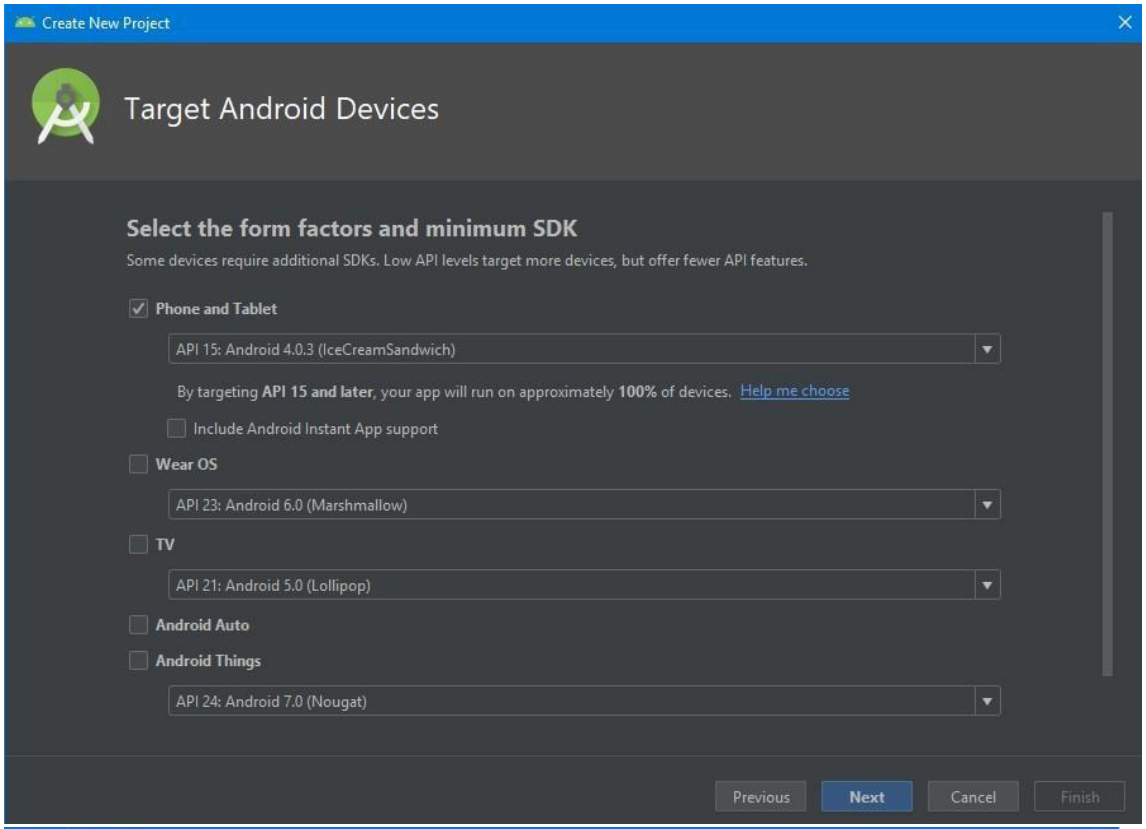
Company domain: rohit.com

Project location: F:\kotlin work\hello

Package name: com.rohit.hello

Include C++ support [ ]

Include Kotlin support [✓]



Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

[✓] Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

[ ] Include Android Instant App support

[ ] Wear OS

API 23: Android 6.0 (Marshmallow)

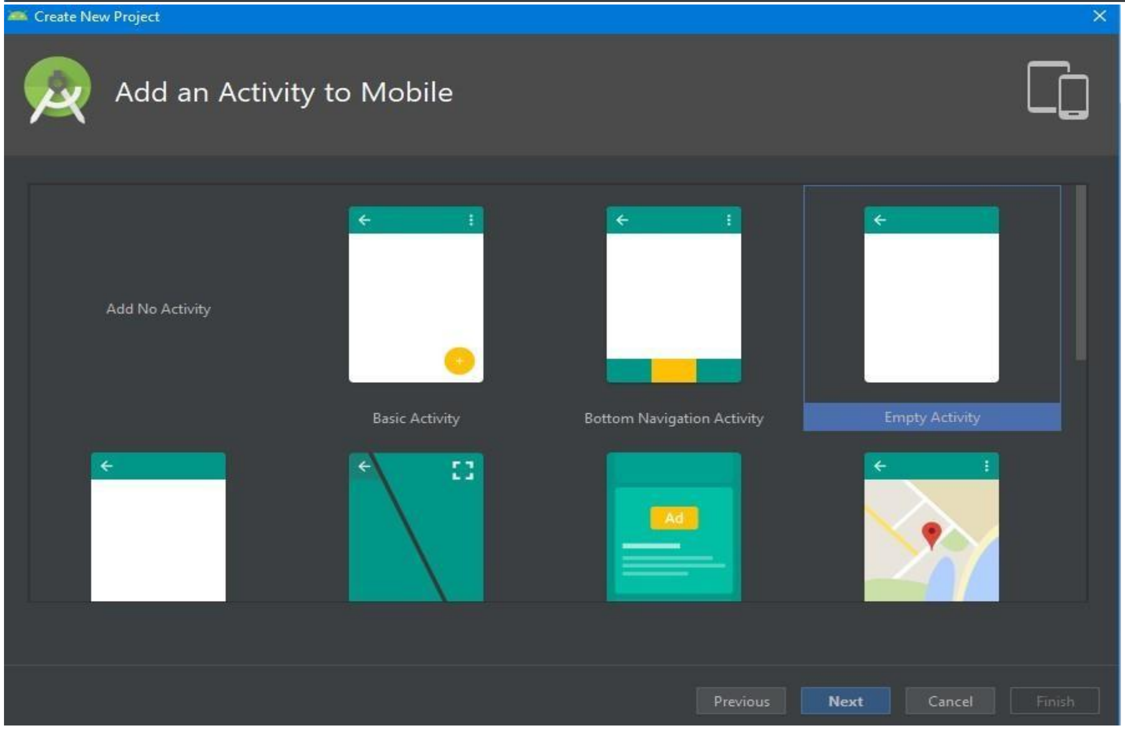
[ ] TV

API 21: Android 5.0 (Lollipop)

[ ] Android Auto

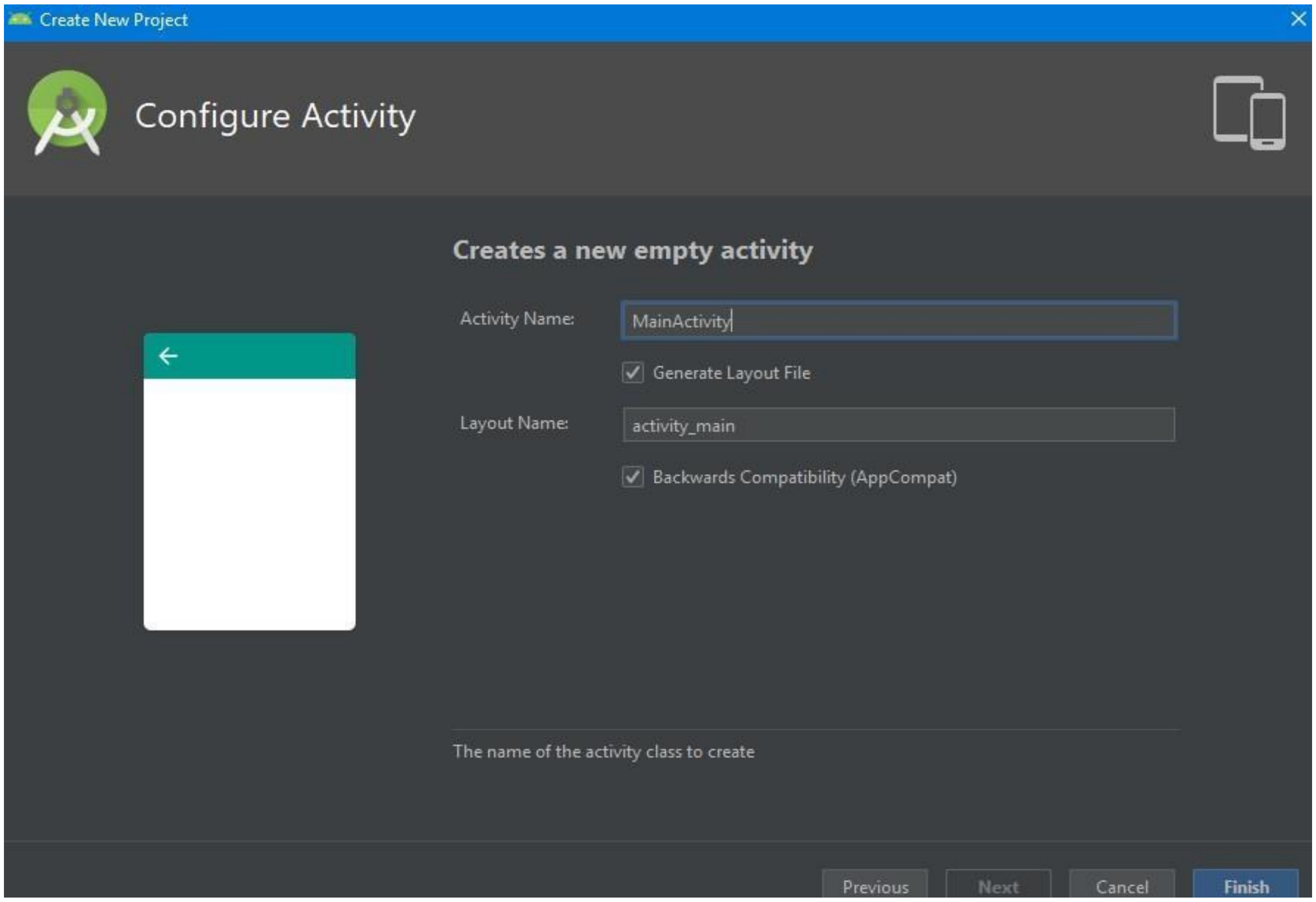
[ ] Android Things

API 24: Android 7.0 (Nougat)



Create New Project

Add an Activity to Mobile



Creates a new empty activity

Activity Name: MainActivity

✓ Generate Layout File

Layout Name: activity_main

✓ Backwards Compatibility (AppCompat)

The name of the activity class to create

Activity_Main.Kt

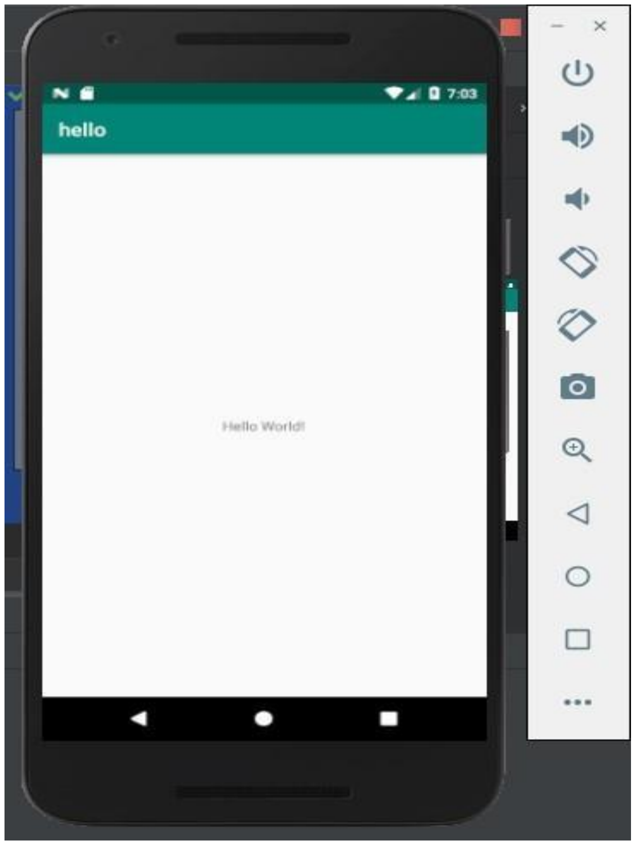
```
package com.rohit.hello

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

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)
    }
}
```

Activity_Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout

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

```
ndroid"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">


    <TextView
            android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>

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

Apk in avd:



BroadcastActivity:

How to receiving Broadcast

Apps can receive and android BroadcastReceiver in two ways: through manifest-declared receivers and context-registered receivers. In this example, we are approaching manifestdeclared Receiver. Learn step by step to the kotlin broadcast receiver example works.

Step 1. Create an android app, For creating an Android app with kotlin read this tutorial.

Step 2. Creating Broadcast Receiver

Create and extend Subclass and BroadcastReceiver implement. onReceive(Context, Intent) where onReceive method each message is received as an Intent object parameter.

MyReceiver.kt:

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

class MyReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
// an Intent broadcast.
        Toast.makeText(context, "Broadcast : Flight mode changed.",
                Toast.LENGTH_LONG).show()
    }
}
```

3.Declare a broadcast receiver in the manifest file add the

element<receiver> in your app’s manifest. Here is code snap <?xml

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="in.eyehunt.androidbroadcasts">
    <application         android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
```

```
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/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=".MyReceiver"
android:enabled="true"
android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.AIRPLANE_MODE"/>
            </intent-filter>
        </receiver>
    </application>

</manifest>
```

Note: If the app is not running and broadcast receiver declared in AndroidManifest.xml, then the system will launch your app.

Step 4. MainActivity code, no needs to do anything

<u>MainActivity.kt:</u> import

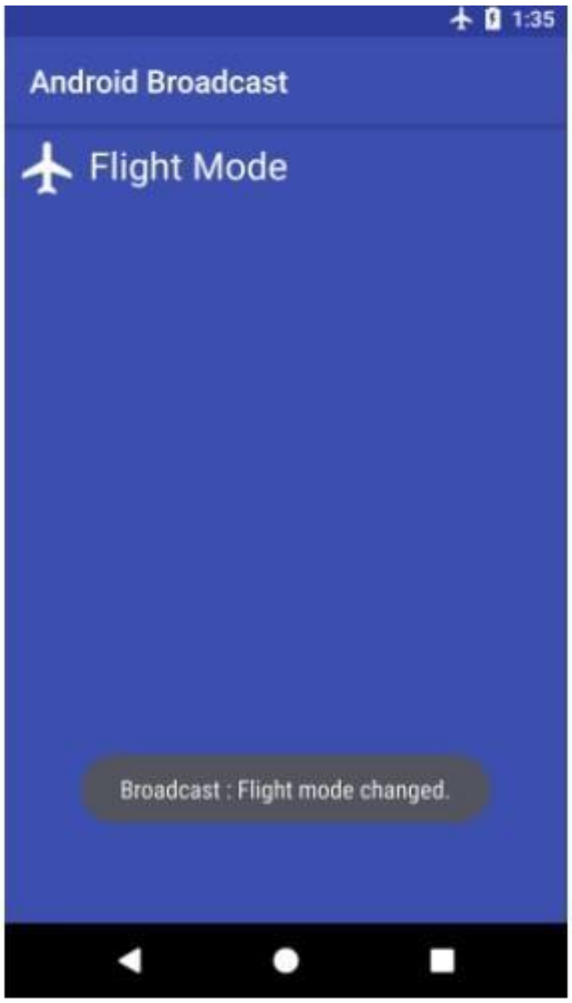
```
android.support.v7.app.AppCompatActivity import
android.os.Bundle class MainActivity :
AppCompatActivity() {     override fun
onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
    }
}
```

Step 5. Add following code in main_activity.xml add
<ImageView> and <TextView>widget layout file.

<u>main_activity.xml:</u>

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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="@color/colorPrimary"
tools:context="in.eyehunt.androidbroadcasts.MainActivity">
```

```
    <ImageView         android:id="@+id/imageView"
android:layout_width="40dp"          android:layout_height="40dp"
android:layout_margin="8dp"          android:layout_marginTop="16dp"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@mipmap/baseline_airplanemode_active_white_24" />

    <TextView         android:id="@+id/textView"
android:layout_width="300dp"
android:layout_height="36dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:gravity="center_vertical"          android:text="Flight
Mode"         android:textColor="@color/colorWhite"
android:textSize="24dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toEndOf="@+id/imageView"
app:layout_constraintTop_toTopOf="@+id/imageView" />
</android.support.constraint.ConstraintLayout>
```

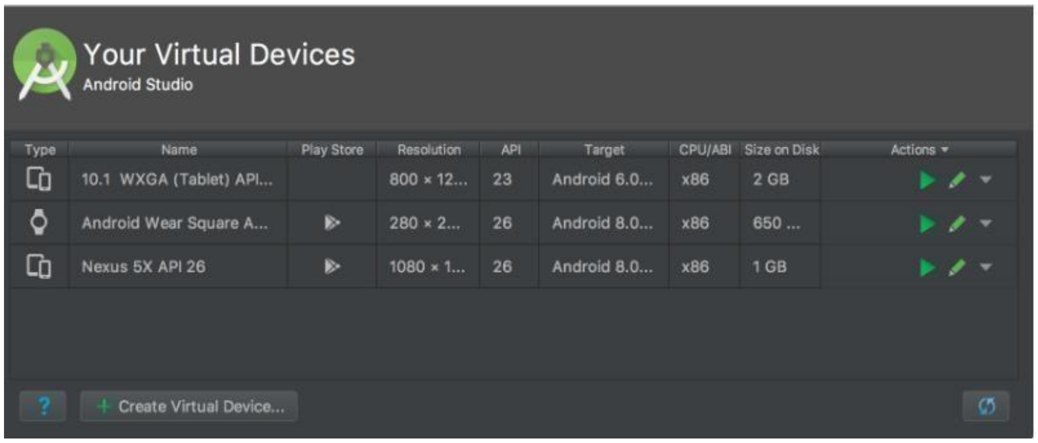


Broadcast : Flight mode changed.

Create and manage virtual devices:

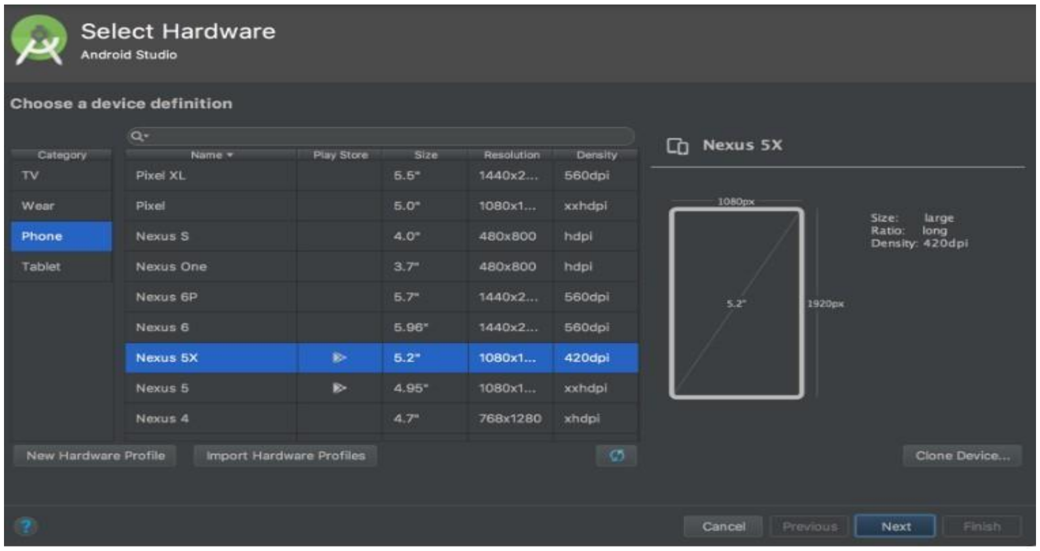
To open the AVD Manager, do one of the following:

- Select Tools > AVD Manager.
- Click AVD Manager AVD Manager icon in the toolbar.



Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	10.1 WXGA (Tablet) API...		800 × 12...	23	Android 6.0...	x86	2 GB	
	Android Wear Square A...		280 × 2...	26	Android 8.0...	x86	650 ...	
	Nexus 5X API 26		1080 × 1...	26	Android 8.0...	x86	1 GB	



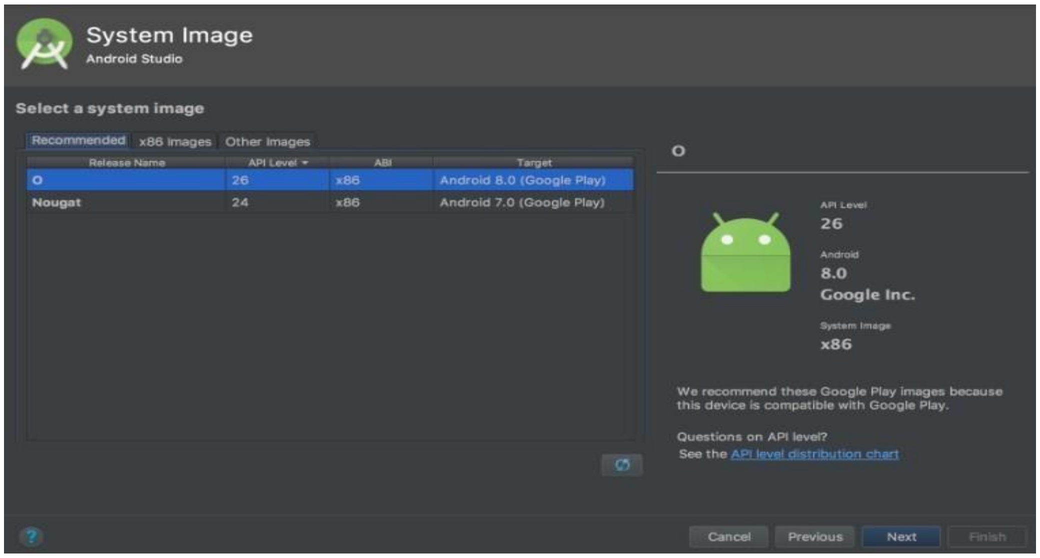
Select Hardware
Android Studio

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Pixel XL		5.5"	1440x2...	560dpi
Wear	Pixel		5.0"	1080x1...	xxhdpi
Phone	Nexus S		4.0"	480x800	hdpi
Tablet	Nexus One		3.7"	480x800	hdpi
	Nexus 6P		5.7"	1440x2...	560dpi
	Nexus 6		5.96"	1440x2...	560dpi
	Nexus 5X		5.2"	1080x1...	420dpi
	Nexus 5		4.95"	1080x1...	xxhdpi
	Nexus 4		4.7"	768x1280	xhdpi

Nexus 5X

Size: large
Ratio: long
Density: 420dpi



System Image
Android Studio

Select a system image

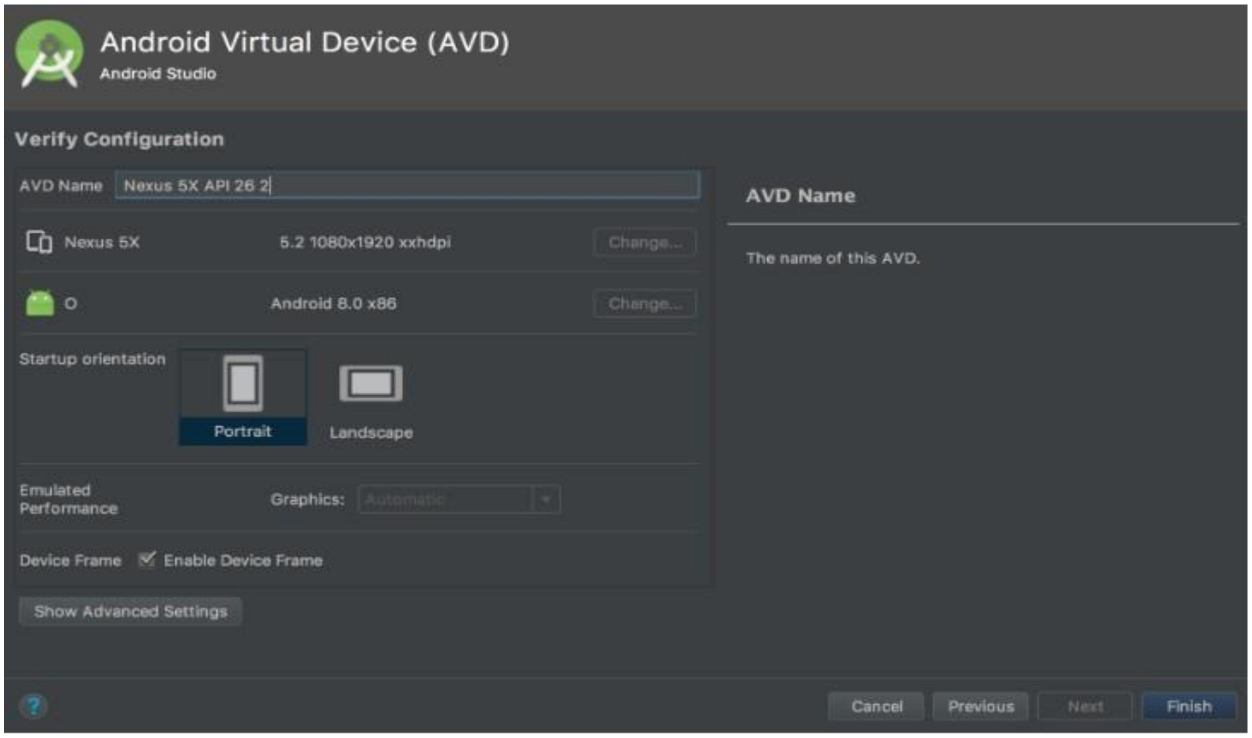
Release Name	API Level	ABI	Target
O	26	x86	Android 8.0 (Google Play)
Nougat	24	x86	Android 7.0 (Google Play)

O

API Level
26
Android
8.0
Google Inc.
System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart


Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Nexus 5X API 26 2

Nexus 5X — 5.2 1080x1920 xxhdpi

O — Android 8.0 x86

Startup orientation: Portrait

Emulated Performance — Graphics: Automatic

Device Frame: ✓ Enable Device Frame

AVD Name

The name of this AVD.

Learning Outcome:

1. **Create projects in Android Studio.**
2. **Implement Android components effectively.**

Course Outcome:

1. **Build functional Android applications.**
2. **Debug and test Android apps.**

Conclusion:

Viva Question:

1. **What is the role of an Activity in Android?**
2. **How do you enable USB debugging on an Android device?**
3. **Explain the purpose of a Content Provider in Android.**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—2:
Aim: Programming Resources: Android Resources: (Color, Theme, String, Drawable, Dimension, Image)
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand Android resource types.
2. Use resources for app design.

THEORY : Programming Resources

Official Android Developer Documentation: The Android Developer website offers comprehensive documentation, guides, and tutorials covering all aspects of Android app development. It's a great starting point for beginners and a valuable reference for experienced developers. Android Developer Documentation

Udacity Android Development Course: Udacity offers a free Android Development course created in collaboration with Google. It covers fundamental concepts, best practices, and hands-on projects to help you build real-world Android apps. Udacity Android Development Course

Coursera Android App Development Specialization: Coursera provides a series of courses on Android app development taught by industry professionals. The specialization covers topics such as Android fundamentals, UI design, and database management. Coursera Android App Development Specialization

Books: There are several books available on Android development, catering to different skill levels and interests. Some recommended titles include "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips and Chris Stewart, "Head First Android Development" by Dawn Griffiths and David Griffiths, and "Android Apprentice" by raywenderlich.com Team.

Stack Overflow: Stack Overflow is a popular platform for asking questions and finding solutions to programming problems, including Android development. You can search for specific issues you encounter or ask your own questions to get help from the community. Stack Overflow - Android

GitHub: GitHub hosts numerous open-source Android projects that you can explore to learn best practices, study code examples, and contribute to collaborative projects. You can also find libraries, frameworks, and tools for Android development on GitHub. GitHub − Android

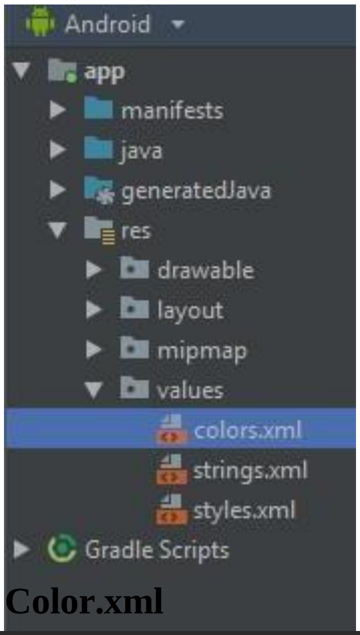
Android Weekly Newsletter: Android Weekly is a curated newsletter containing the latest news, articles, tutorials, and libraries related to Android development. Subscribing to Android Weekly can help you stay updated on industry trends and useful resources. Android Weekly

Step:-

Programming Resources

Android Resources: (Color, Theme, String, Drawable, Dimension, Image).

Color:



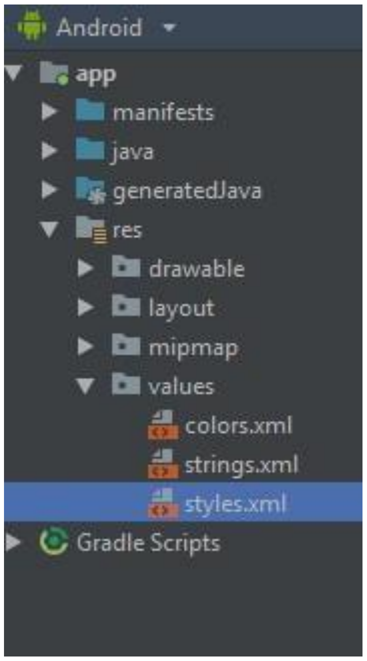
Color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color
name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

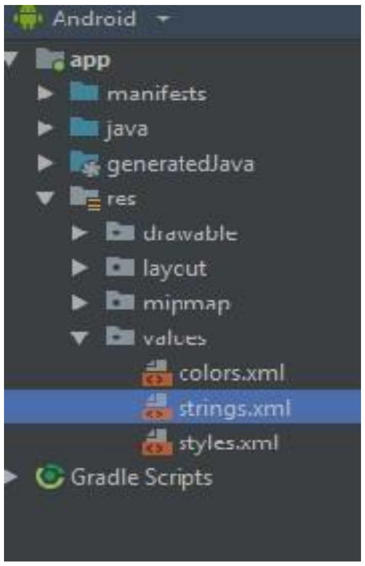
Theme:

Style.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```



String:

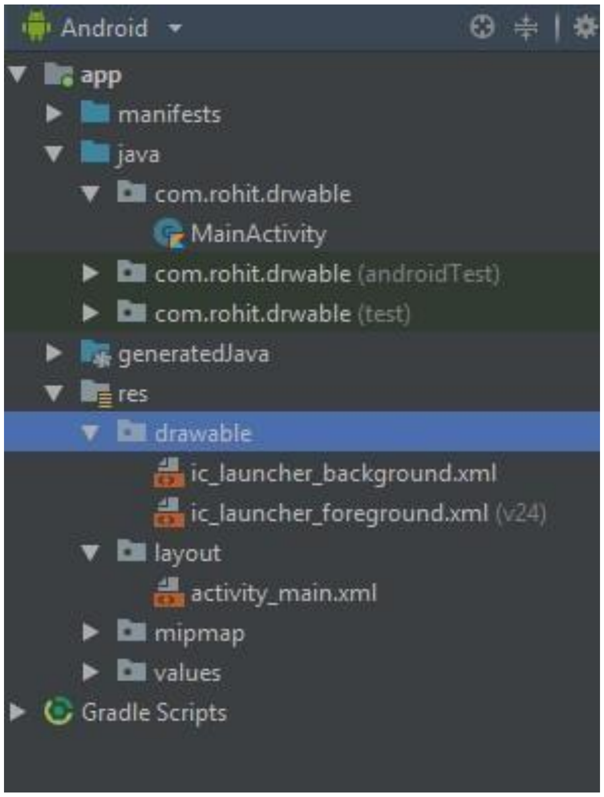


String.xml:

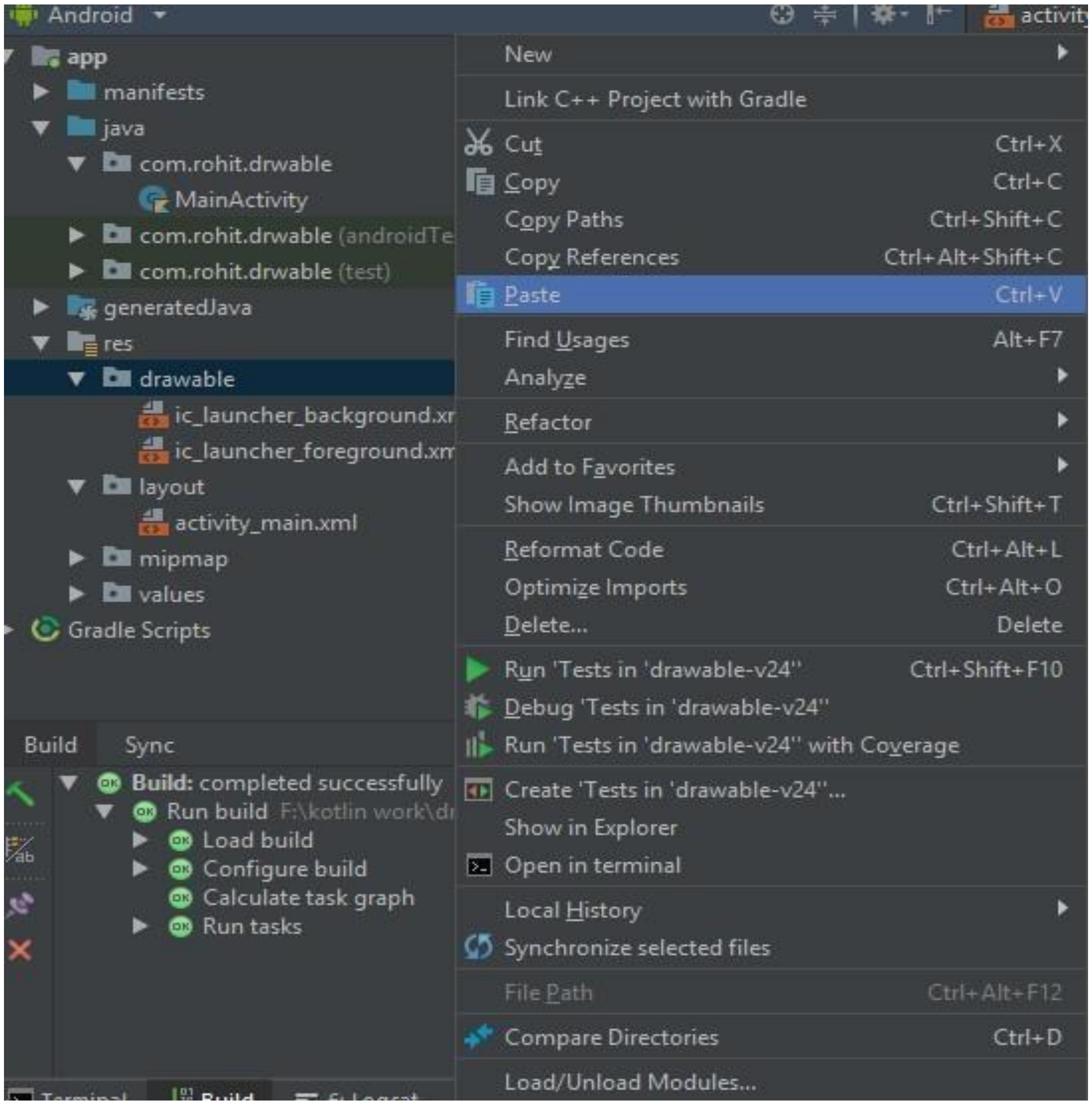
```
<resources>
    <string name="app_name">hello</string>
    <string name="numbers">
        <item>1</item>
        <item>2</item>
        <item>3</item>
        </item>
    </string>
</resources>
```

Drawable:

1. Right click on drawable folder



2. Copy the image if you want to create image drawable
3. Paste that image file inside the drawable folder


Android
app
manifests
java
com.rohit.drwable
MainActivity
com.rohit.drwable (androidTe
com.rohit.drwable (test)
generatedJava
res
drawable
ic_launcher_background.xr
ic_launcher_foreground.xm
layout
activity_main.xml
mipmap
values
Gradle Scripts
Build
Sync
Build: completed successfully
Run build F:\kotlin work\dr
Load build
Configure build
Calculate task graph
Run tasks
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Paths Ctrl+Shift+C
Copy References Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'Tests in 'drawable-v24'' Ctrl+Shift+F10
Debug 'Tests in 'drawable-v24''
Run 'Tests in 'drawable-v24'' with Coverage
Create 'Tests in 'drawable-v24''...
Show in Explorer
Open in terminal
Local History
Synchronize selected files
File Path Ctrl+Alt+F12
Compare Directories Ctrl+D
Load/Unload Modules...

Note: to create drawable resource, right click on drawable folder and select drawable resource file.

Dimension, Image:

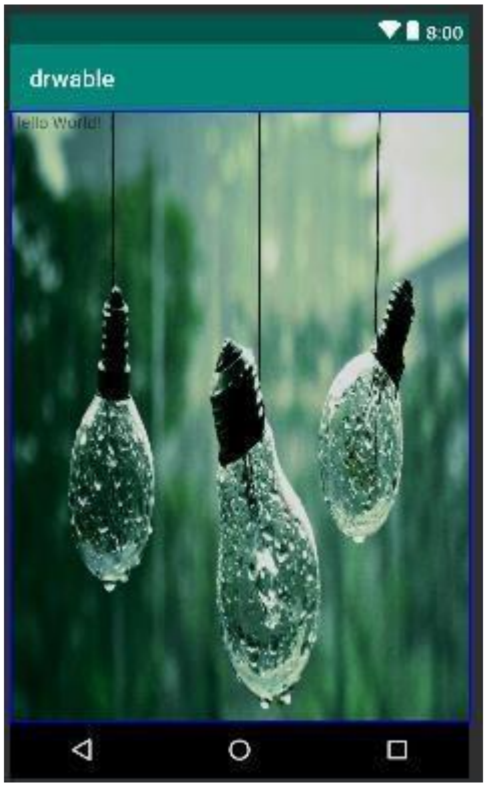
Main_Activity.kt:

```
package com.rohit.drwable

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

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
    }
}
```

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"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"          android:background="@drawable/one">

    <TextView
            android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"

app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
</LinearLayout>
```

Output:


8:00
drwable
Hello World!

Learning Outcomes:

1 **Implement color, theme, and string resources.**
2 **Use drawable and image resources effectively.**

Course Outcomes:

1 **Design visually appealing Android apps.**
2 **Manage app resources efficiently.**

Conclusion:

Viva Questions:

1 **What is the purpose of a drawable resource in Android?**
2 **How do you define a custom theme in Android?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—3:
Aim: Programming Activities and fragments Activity Life Cycle, Activity methods, Multiple Activities, Life Cycle of fragments and multiple fragments
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand Android activity lifecycle.
2. Manage multiple activities and fragments

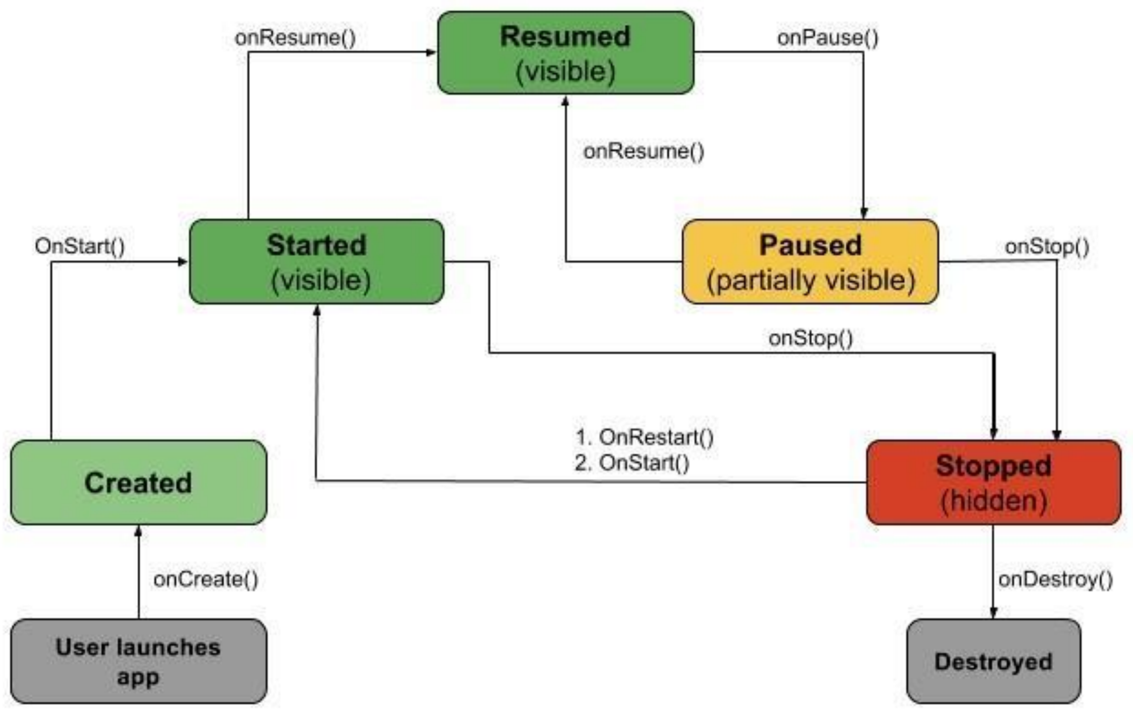
THEORY : Programming Activities and fragments

1. **Activities**:
    - An Activity is a fundamental building block of an Android application, representing a single screen with a user interface.
    - Activities are responsible for handling user interactions, managing UI components, and coordinating with other app components.
    - Each Activity typically corresponds to a specific user task or UI flow within the app.
2. **Fragments**:
    - Fragments represent reusable portions of UI within an Activity, allowing for modular and flexible design.
    - Fragments can be combined or swapped in and out of Activities dynamically, enabling more dynamic and responsive UIs.
    - Fragments have their lifecycle, similar to Activities, and can have their own UI layout and behavior.
3. **Activity Lifecycle**:
    - The Activity lifecycle comprises a series of states through which an Activity transitions during its lifetime.
    - Common lifecycle methods include onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy().
    - Understanding the Activity lifecycle is crucial for managing UI state, handling configuration changes, and ensuring proper resource management.
4. **Fragment Lifecycle**:
    - Fragments also have their lifecycle, which is closely related to the lifecycle of their host Activity.

- Fragment lifecycle methods include onAttach(), onCreateView(), onViewCreated(), onStart(), onResume(), onPause(), onStop(), onDestroyView(), and onDetach().
- Managing the Fragment lifecycle is essential for handling UI updates, managing resources, and ensuring proper communication between Fragments and their host Activity.

Step:-

Activity Lifecycle:



- **onCreate():** Called by the OS when the activity is first created. This is where you initialize any UI elements or data objects. You also have the savedInstanceState of the activity that contains its previously saved state, and you can use it to recreate that state.\

```
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
   setContentView(R.layout.activity_task_description)
```

- **onStart():** Just before presenting the user with an activity, this method is called. It’s always followed by onResume(). In here, you generally should start UI animations, audio based content or anything else that requires the activity’s contents to be on screen.

- **onResume():** As an activity enters the foreground, this method is called. Here you have a good place to restart animations, update UI elements, restart camera previews, resume audio/video playback or initialize any components that you release during onPause().
- **onPause():** This method is called before sliding into the background. Here you should stop any visuals or audio associated with the activity such as UI animations, music playback or the camera. This method is followed by onResume() if the activity returns to the foreground or by onStop() if it becomes hidden.

- **onStop():** This method is called right after onPause(), when the activity is no longer visible to the user, and it’s a good place to save data that you want to commit to the disk. It’s followed by either onRestart(), if this activity is coming back to the foreground, or onDestroy() if it’s being released from memory.

- **onRestart():** Called after stopping an activity, but just before starting it again. It’s always followed by onStart().

- **onDestroy():** This is the final callback you’ll receive from the OS before the activity is destroyed. You can trigger an activity’s desctruction by calling finish(), or it can be triggered by the system when the system needs to recoup memory. If your activity includes any background threads or other long-running resources, destruction could lead to a memory leak if they’re not released, so you need to remember to stop these processes here as well.

EXAMPLE:

```
import android.os.Bundle
import android.support.design.widget.Snackbar
import android.support.v7.app.AppCompatActivity
import android.view.Menu import
android.view.MenuItem import android.util.Log

import kotlinx.android.synthetic.main.activity_state_change.*

class StateChangeActivity : AppCompatActivity() {

    val TAG = "StateChange"

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_state_change)
setSupportActionBar(toolbar)
         fab.setOnClickListener { view -
>
            Snackbar.make(view, "Replace with your own action",
                Snackbar.LENGTH_LONG)
                .setAction("Action", null).show()
}
        Log.i(TAG, "onCreate")
    }
}
override fun onStart() {
super.onStart()      Log.i(TAG,
"onStart")
}

override fun onResume() {
super.onResume()      Log.i(TAG,
"onResume")
}

override fun onPause() {
super.onPause()      Log.i(TAG,
"onPause")
}

override fun onStop() {
super.onStop()      Log.i(TAG,
"onStop")
}

override fun onRestart() {
super.onRestart()      Log.i(TAG,
"onRestart")
}

override fun onDestroy() {
super.onDestroy()      Log.i(TAG,
"onDestroy")
}

override fun onSaveInstanceState(outState: Bundle?) {
    super.onSaveInstanceState(outState)
    Log.i(TAG, "onSaveInstanceState")
}

override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
super.onRestoreInstanceState(savedInstanceState)
    Log.i(TAG, "onRestoreInstanceState") }
```

Multiple Activities:

activity_first.xml code:

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

<Button
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="Ganesh"
android:text="click third activity" android:textColor="@color/colorPrimary"
app:layout_constraintTop_toTopOf="parent"


tools:layout_editor_absoluteX="168dp"
android:layout_alignParentBottom="true"
android:layout_toEndOf="@+id/text" android:layout_marginBottom="196dp"
/>
 <TextView android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="This s my first
app!" android:id="@+id/text" tools:layout_editor_absoluteY="8dp"
tools:layout_editor_absoluteX="8dp" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/button"
android:text="click second activity"
android:textColor="@color/colorPrimary" android:onClick="Ganesh"
tools:layout_editor_absoluteX="168dp"
app:layout_constraintTop_toTopOf="parent"
android:layout_above="@+id/button2"
android:layout_alignStart="@+id/button2"
android:layout_marginBottom="40dp" />

</RelativeLayout>
```

activity_second.xml code:

```
<?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:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20pt"
android:text="second acticity is working...." android:textAllCaps="true"
android:textColor="@color/colorPrimaryDark"/>

</LinearLayout>
```

activity_third.xml code:

```
<?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:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20pt"
android:text="Third activity is working ........." android:textAllCaps="true"
android:textColor="@color/colorPrimary"
/>

</LinearLayout>
```

Activity_first.kt

```
package rohit.technobeat

import android.content.Intent
import android.support.v7.app.AppCompatActivity import
android.os.Bundle
import kotlinx.android.synthetic.main.activity_login.*
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.activity_register.*
import rohit.technobeat.R.id.login import
rohit.technobeat.R.id.newaccount

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
second.setOnClickListener {
            val intent = Intent(this, Activity_second::class.java)
            // start your next activity
startActivity(intent)
        }

        third.setOnClickListener {              val intent =
Intent(this, Activity_third::class.java)
            // start your next activity
startActivity(intent)
        }



    }
}
```

Learning Outcomes:

1 **Implement activity lifecycle methods.**

2 **Handle multiple fragments in an app**

Course Outcomes:

1 **Build apps with multiple activities.**

2 **Control fragment life cycle effectively.**

Conclusion:

Viva Questions:

1 **What are the key methods in the Android activity lifecycle?**

2 **How do you manage communication between fragments?**

3 **What is the difference between an Activity and a Fragment?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—4:
Aim: Programs related to different Layouts: Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand various Android layouts.
2. Use different layouts for UI design.

THEORY : Programs related to different Layouts

1. **LinearLayout**:
   - Program to demonstrate a basic LinearLayout with vertical or horizontal orientation.
   - Program to show how to use layout weights to distribute space proportionally among child views.
2. **RelativeLayout**:
   - Program to create a RelativeLayout with multiple child views positioned relative to each other or to the parent layout.
3. **ConstraintLayout**:
   - Program to implement a ConstraintLayout with constraints defining the positioning of child views relative to each other and to the parent layout.
4. **FrameLayout**:
   - Program to illustrate the use of FrameLayout for stacking multiple child views on top of each other, such as in a layered UI.
5. **GridLayout**:
   - Program to demonstrate a GridLayout with a fixed number of rows and columns, arranging child views in a grid-like fashion.
6. **TableLayout**:
   - Program to create a TableLayout with rows and columns, similar to an HTML table, to organize child views in a tabular format.
7. **ScrollView**:
   - Program to show how to use a ScrollView to create a scrollable container for child views that exceed the available screen space.
8. **Nested Layouts**:

- Program to demonstrate nesting multiple layouts (e.g., LinearLayout inside a RelativeLayout) to create complex UI designs.

9. **Custom Layouts**:
   - Program to create a custom ViewGroup subclass implementing a custom layout manager to arrange child views in a specific manner.

Step:-

1. linear layout:

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

<Button android:id="@+id/btnStartService"
android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="start_service"/>

<Button android:id="@+id/btnPauseService"
android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="pause_service"/>

<Button android:id="@+id/btnStopService"
android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="stop_service"/>

</LinearLayout>
```

2. Relative:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent" android:layout_height="fill_parent"
android:paddingLeft="16dp" android:paddingRight="16dp" >

<EditText
android:id="@+id/name"
android:layout_width="fill_parent"
android:layout_height="wrap_content" android:hint="@string/reminder"
/>

<LinearLayout
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_alignParentStart="true"
android:layout_below="@+id/name">

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="New Button"
android:id="@+id/button" />


<Button android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="New Button" android:id="@+id/button2" />


</LinearLayout>


</RelativeLayout>
```

3. Table:
   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"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"          tools:context=".MainActivity">

        <TableLayout android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="50dp"          android:layout_marginTop="150dp">
            <TableRow>
<Button
                android:id="@+id/btn1"
android:text="1"
                android:layout_gravity="center"
                />
<Button
                        android:id="@+id/btn2"
android:text="2"
                        android:layout_gravity="center"
                />
<Button
                    android:id="@+id/btn3"
android:text="3"
                    android:layout_gravity="center"
                />

            </TableRow>
<TableRow>
<Button
                        android:id="@+id/btn4"
android:text="4"
                        android:layout_gravity="center"
                />
<Button
                        android:id="@+id/btn5"
android:text="5"
                        android:layout_gravity="center"

                /><Button
                    android:id="@+id/btn6"
android:text="6"
                    android:layout_gravity="center"
            />

            </TableRow>
<TableRow>
<Button
                        android:id="@+id/btn7"
android:text="7"
                        android:layout_gravity="center"
/>


                <Button   android:text="8"
```

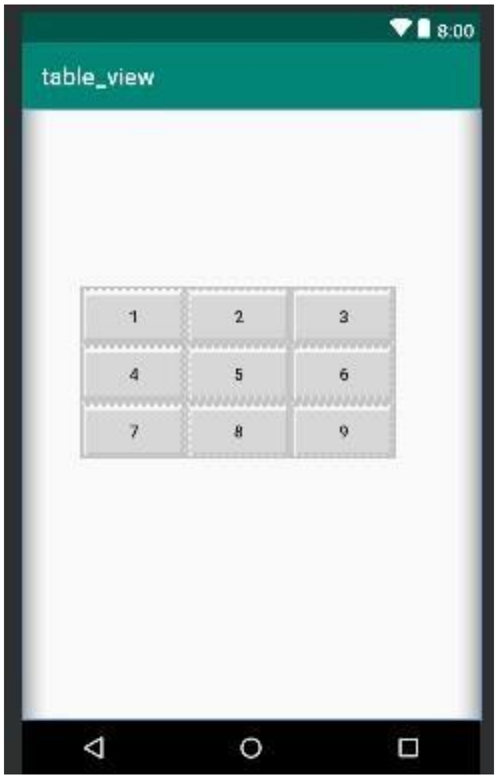
```
                              android:id="@+id/btn8"
android:layout_gravity="center"
                    /><Button
android:id="@+id/btn9"
```

```
android:text="9"
android:layout_gravity="center"
            />
            </TableRow>
        </TableLayout>

</LinearLayout>
```

Activity_main.kt

```
package com.r.table_view

import android.support.v7.app.AppCompatActivity import
android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
import org.jetbrains.anko.toast

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)

        btn1.setOnClickListener {
toast("1")
        }
        btn2.setOnClickListener {
toast("2")
        }
        btn3.setOnClickListener {
toast("3")
        }
        btn4.setOnClickListener {
toast("4")
        }
        btn5.setOnClickListener {
toast("5")
        }
        btn6.setOnClickListener {
toast("6")
        }
        btn7.setOnClickListener {
toast("7")
        }
        btn8.setOnClickListener {
toast("8")
        }
        btn9.setOnClickListener {
toast("9")
        }



    }
}
```

output:



4. Frame:

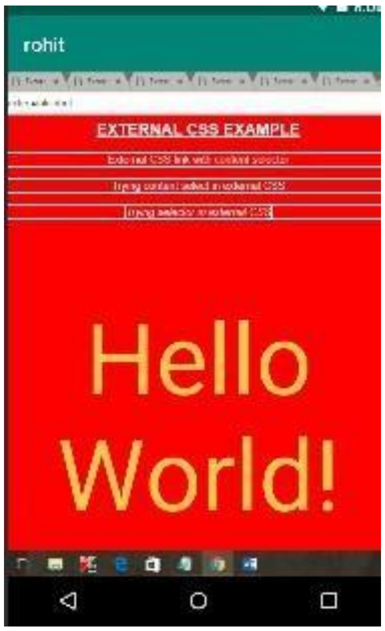
   Activity_main.xml

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

    <ImageView android:layout_width="match_parent"
android:layout_height="match_parent"
android:src="@drawable/red"                  android:scaleType="centerCrop"/>
    <TextView
            android:textSize="100dp"
            android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
android:gravity="center"
android:textColor="@color/rohit"
android:layout_marginTop="220dp"
            />
</FrameLayout>
```

Activity_main.kt

```
package com.rohit.frame_layout

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

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)
    }
}
```

output:



5. List View:

Activity_main.xml

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

xmlns:android="http://schemas.android.com/apk/res/a
ndroid"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

   <Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/btn"
android:text="Click me to view list"
android:layout_marginTop="200dp"
android:layout_marginLeft="90dp"/>
</LinearLayout>
```

String.xml

```
<resources>
    <string name="app_name">list</string>
```

```xml
    <array name="insert_list">
        <item>one</item>
        <item>two</item>
        <item>three</item>
        <item>four</item>
        <item>five</item>
        <item>six</item>
        <item>seven</item>
        <item>eight</item>
        <item>nine</item>
        <item>ten</item>
    </array>
</resources>
```

Activity_list_view.xml:

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

xmlns:android="http://schemas.android.com/apk/res/a
ndroid"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
        tools:context=".list_view"
android:entries="@array/insert_list">
 </ListView>
```

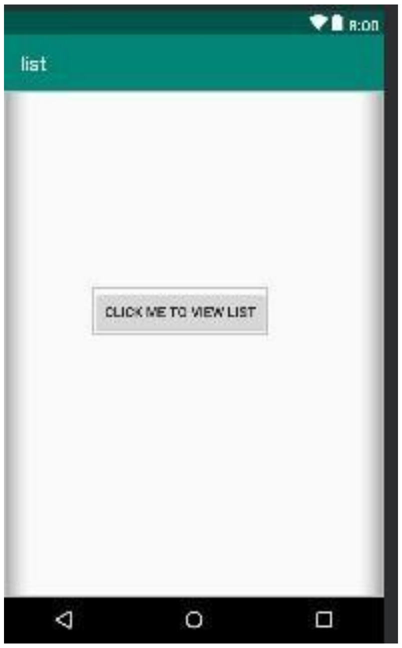
List_view.kt:

```kotlin
package com.rohit.list

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

class list_view : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_list_view)
    }
}
```

main_Activity.kt

```
package com.rohit.list

import android.content.Intent
import android.support.v7.app.AppCompatActivity import
android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)

        btn.setOnClickListener {
            val intent =Intent(this, list_view::class.java)
startActivity(intent)
        }

    }
}
```

output:



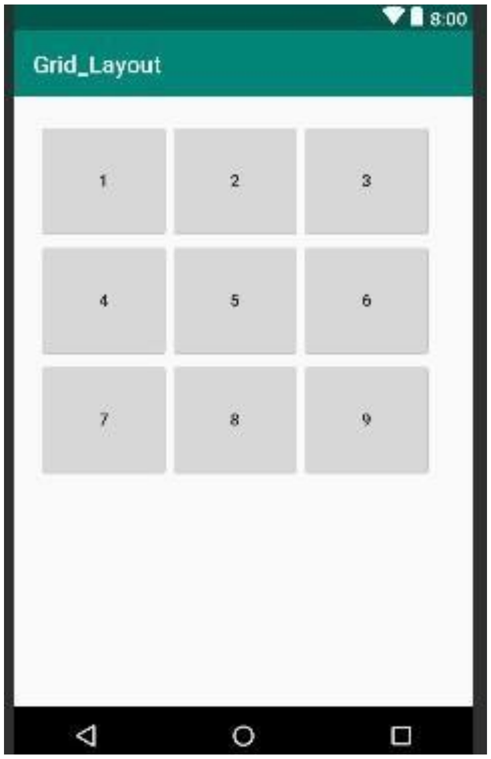
6. Grid layout:

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

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"          android:rowCount="3"
    android:columnCount="3"          android:padding="20dp">

        <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="1"/>

        <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="2"/>
        <Button
                android:layout_width="110dp"
    android:layout_height="100dp"

                android:text="3"/>
    <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="4"/>
         <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="5"/>      <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="6"/>      <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="7"/>
         <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="8"/>      <Button
                android:layout_width="110dp"
    android:layout_height="100dp"
    android:text="9"/>

    </GridLayout>
```

mainActvity.kt:

```
package com.rohit.grid_layout

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

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)
    }
```

output:



Learning Outcomes:

1 Implement coordinate and linear layouts.

2 Design UI using list and grid views.

Course Outcomes:

1 Build responsive UIs with layouts.

2 Design efficient views in Android apps.

Conclusion:

Viva Questions:

1 What is the difference between LinearLayout and RelativeLayout?

2 When should you use TableLayout in Android?

3 How does GridView differ from ListView in Android?

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—5:
Aim: Programming UI elements: AppBar, Fragments, UI Components
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand app bar and fragments.
2. Work with essential UI components.

THEORY : Programming UI elements

1. **Button**:
    - Program to create a Button and handle its onClick event to perform a specific action when clicked.
2. **TextView**:
    - Program to display a TextView with text formatting (e.g., bold, italic) and custom styling (e.g., text color, size).
3. **EditText**:
    - Program to create an EditText for user input, such as entering text or numbers, and retrieving the entered value.
4. **ImageView**:
    - Program to load and display an image using an ImageView, including handling different scale types and image resources.
5. **CheckBox**:
    - Program to implement CheckBoxes for selecting multiple options and handling their checked state changes.
6. **RadioButton** and **RadioGroup**:
    - Program to create RadioButtons and organize them within a RadioGroup, allowing users to select a single option from multiple choices.
7. **ToggleButton**:
    - Program to use ToggleButton to implement a switch-like UI element for toggling between two states.
8. **SeekBar**:

- Program to implement a SeekBar for selecting a value within a range, such as adjusting volume or brightness.

9. **DatePicker** and **TimePicker**:

    - Program to use DatePicker and TimePicker dialogs to allow users to select dates and times, respectively.

Step:-

Design App With UI:

mainActivity.kt:

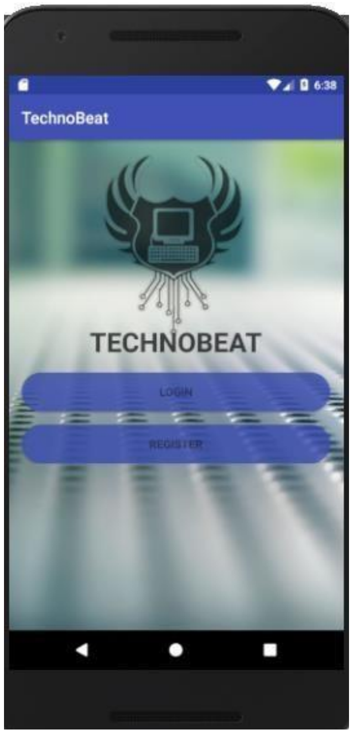
```
package rohit.technobeat

import android.content.Intent
import android.support.v7.app.AppCompatActivity import
android.os.Bundle
import kotlinx.android.synthetic.main.activity_login.*
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.activity_register.*
import rohit.technobeat.R.id.login import
rohit.technobeat.R.id.newaccount

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
login.setOnClickListener {
            val intent = Intent(this, LoginActivity::class.java)
            // start your next activity
startActivity(intent)
        }

        newaccount.setOnClickListener {              val intent =
Intent(this, RegisterActivity::class.java)
            // start your next activity
startActivity(intent)
        }



    }
}
```

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:gravity="center_horizontal"      android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:background="@drawable/home"      tools:context=".MainActivity">

    <ScrollView
        android:id="@+id/login_form"
android:layout_width="match_parent"
android:layout_height="match_parent">
        <LinearLayout
            android:layout_width="match_parent"

            android:layout_height="wrap_content"
android:orientation="vertical"
android:gravity="center">



            <android.support.v7.widget.AppCompatTextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="210dp"
android:alpha="0.7"
android:text="TECHNOBEAT"
android:textColor="#000000"
android:textSize="33dp"
android:textStyle="bold"
tools:layout_marginLeft="85dp" />



            <Button                  android:id="@+id/login"
style="?android:textAppearanceSmall"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
android:text="Login"
                android:background="@drawable/round_button"
android:alpha="0.8"                  android:textStyle="bold"
/>
            <Button
                android:id="@+id/newaccount"
style="?android:textAppearanceSmall"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
android:text="REGISTER"
android:background="@drawable/round_button"
```

```
android:alpha="0.8"                    android:textStyle="bold"
/>

        </LinearLayout>
    </ScrollView> </LinearLayout>
```

Output:



Learning Outcomes:

1 **Implement app bar in Android apps.**
2 **Use UI components effectively in fragments.**

Course Outcomes:

1 **Build interactive user interfaces.**
2 **Create apps with dynamic fragments.**

Conclusion:

Viva Questions:

1 **What is the purpose of an AppBar in Android?**
2 **How can you pass data between fragments?**
3 **What are the key UI components in Android?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—6:
Aim: Programming menus, dialog, dialog fragments
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand menu and dialog implementation.
2. Work with dialog fragments in Android.

THEORY : Programming menus, dialog, dialog fragments

1. **Menus**:
   - **Options Menu**: The options menu is a primary menu that appears when the user presses the menu button on the device. It typically contains actions related to the current context or activity.
   - **Context Menu**: The context menu appears when the user performs a long press on an item, such as a list item or view. It provides options relevant to the selected item.
   - **PopupMenu**: The popup menu is a menu that appears anchored to a view, typically in response to a user action like a button click. It displays a list of actions or options.
2. **Dialogs**:
   - **AlertDialog**: AlertDialog is a dialog box that displays a message to the user with optional buttons for user actions (e.g., OK, Cancel). It's commonly used for alerts, confirmations, or simple input gathering.
   - **ProgressDialog**: ProgressDialog is a dialog that shows a progress indicator, typically used to indicate the progress of a long-running operation.
   - **DatePickerDialog** and **TimePickerDialog**: These dialogs provide a convenient way for users to select a date or time using a calendar or clock interface.
3. **Dialog Fragments**:
   - **DialogFragment**: DialogFragment is a fragment that displays a dialog window, allowing for more complex dialogs with custom layouts and interactions.
   - **Creating Dialog Fragments**: DialogFragment can be created by extending the DialogFragment class and overriding methods such as onCreateView() to inflate a custom layout and onCreateDialog() to customize the dialog's appearance and behavior.

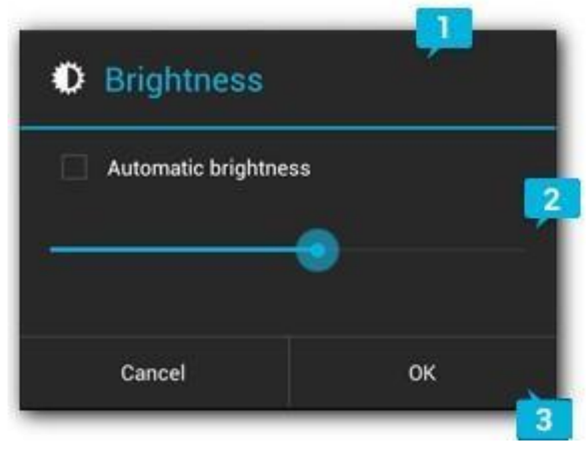
- **Managing Dialog Fragments**: DialogFragment instances are typically managed by the FragmentManager, which handles their lifecycle and displays them as dialogs when necessary.

Step:-

Alert:

```
val alertDialog: AlertDialog? = activity?.let {
val builder = AlertDialog.Builder(it)
builder.apply {
setPositiveButton(R.string.ok,
            DialogInterface.OnClickListener { dialog, id ->
                // User clicked OK button
            })
        setNegativeButton(R.string.cancel,
            DialogInterface.OnClickListener { dialog, id ->
                // User cancelled the dialog
            })
    }
    // Set other dialog properties
    ...

    // Create the AlertDialog
builder.create()
```

output:



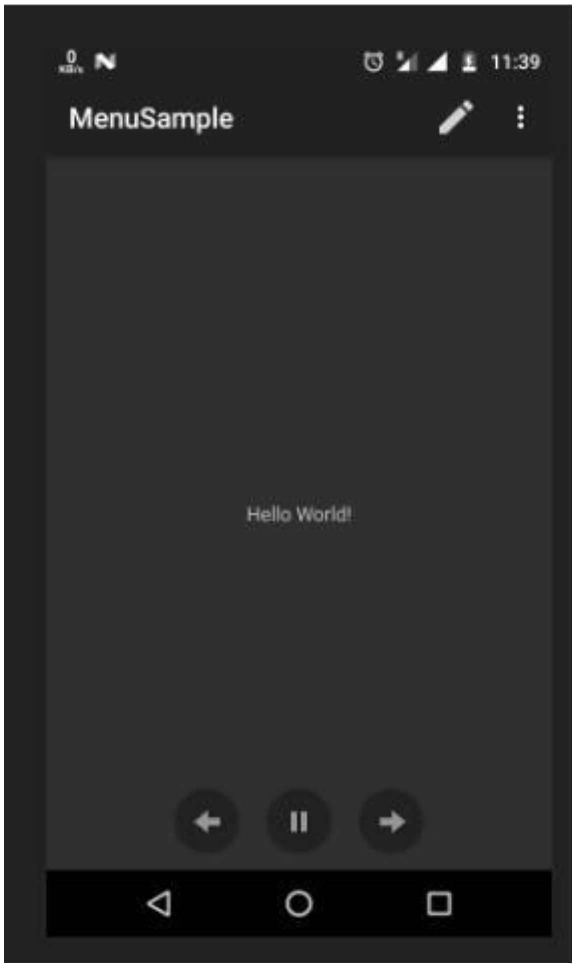
Menu: menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android&#8221;
xmlns:app="http://schemas.android.com/apk/res-auto"&gt;
<item
android:id="@+id/menu_1"
android:icon="@drawable/ic_menu_1"
android:title="Menu 1"
app:showAsAction="always" />


<item android:id="@+id/menu_2" android:icon="@drawable/ic_menu_2"
android:title="Menu 2" />

<item android:id="@+id/menu_3" android:icon="@drawable/ic_menu_3"
android:title="Menu 3" />

<item android:id="@+id/menu_4" android:icon="@drawable/ic_menu_4"
android:title="Menu 4" />

</menu>
```

MainActivity.kt:

```
package rohit.com

import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.view.Menu import
android.view.MenuItem import
android.widget.Toast
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean
{          menuInflater.inflate(R.menu.main, menu)
return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
{          when (item.itemId) {               R.id.menu_1 -> {
                Toast.makeText(this, “Menu 1 is selected”, Toast.LENGTH_SHORT).show()
return true
            }
            R.id.menu_2 -> {
                Toast.makeText(this, “Menu 2 is selected”, Toast.LENGTH_SHORT).show()
return true
            }
            R.id.menu_3 -> {
                Toast.makeText(this, “Menu 3 is selected”, Toast.LENGTH_SHORT).show()
return true
            }
            R.id.menu_4 -> {
                Toast.makeText(this, “Menu 4 is selected”, Toast.LENGTH_SHORT).show()
return true
            }
            else -> return super.onOptionsItemSelected(item)
        }
    }
}
```

Output:


11:39
MenuSample
Hello World!

Learning Outcomes:

1 **Create and manage Android menus.**
2 **Implement dialog and dialog fragments.**
3

Course Outcomes:

1 **Develop apps with interactive menus.**
2 **Handle user interactions with dialogs.**

Conclusion:

Viva Questions:

1 **How do you create an options menu in Android?**
2 **What is the difference between a dialog and a dialog fragment?**
3 **How can you customize a dialog in Android?**

For Faculty Use

Correction Parameter	**Formative Assessment [ ]**	**Timely Completion Of practical [ ]**	**Attendance learning Attitude [ ]**

Experiment No.—7:
Aim: Programs on Intents, Events, Listeners and Adapters: The Android Intent Class, Using Events and Event Listeners
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand the Android Intent class.
2. Implement event listeners and adapters.

THEORY : Programs on Intents, Events Listeners and Adapters

1. **Intents**:
    - Intents are messaging objects used to communicate between components (e.g., activities, services) within an Android application or between different applications.
    - Intents can be explicit, targeting a specific component within the same app, or implicit, allowing the system to determine the appropriate component based on the action and data provided.
    - Intents can carry data (extras) such as strings, numbers, or custom objects to be passed between components.
    - Intents are commonly used for starting activities, broadcasting messages, invoking services, and launching external applications.
2. **Event Listeners**:
    - Event listeners are mechanisms used to detect and handle user interactions with UI components, such as button clicks, text changes, or item selections.
    - Event listeners are typically implemented as interfaces with callback methods that are invoked when specific events occur.
    - Common event listeners in Android include OnClickListener for handling button clicks, OnItemSelectedListener for handling item selections in spinners or list views, and TextWatcher for monitoring text changes in EditText fields.
3. **Adapters**:
    - Adapters are components used to bind data to views and populate UI elements such as ListView, GridView, or RecyclerView with dynamic content.
    - Adapters act as a bridge between data sources (e.g., arrays, lists, databases) and UI components, providing methods for creating views for each item in the data set and managing view recycling and data binding.

- Android provides built-in adapter classes such as ArrayAdapter, CursorAdapter, and BaseAdapter for common use cases, and custom adapter implementations can be created to handle specific data structures and UI requirements.

Step:-

1. Intents:
    - Program to send an explicit intent to start another activity within the same app.

```
<!-- res/layout/activity_main.xml -->
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start Second Activity" />

// MainActivity.java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, SecondActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

2. Event Listeners:
    - Program to register and handle click events using OnClickListener for a Button.

```
<!-- res/layout/activity_main.xml -->
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:text="Click Me" />

    // MainActivity.java
    public class MainActivity extends AppCompatActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            Button button = findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // Handle button click
                    Toast.makeText(MainActivity.this, "Button clicked", Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
```

3. Adapters:
    - Program to create a custom ArrayAdapter to populate a ListView with data.

```
<!-- res/layout/activity_main.xml -->
<ListView
    android:id="@+id/list_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<!-- res/layout/list_item.xml -->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="10dp"/>

// MainActivity.java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ArrayList<String> dataList = new ArrayList<>();
        dataList.add("Item 1");
        dataList.add("Item 2");
        dataList.add("Item 3");

        ListView listView = findViewById(R.id.list_view);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this, R.layout.list_item, dataList);
        listView.setAdapter(adapter);
    }
}
```

Learning Outcomes:

1 **Use intents for inter-component communication.**
2 **Handle events using event listeners.**

Course Outcomes:

1 **Build apps with intent-based interactions.**
2 **Create interactive UIs using listeners and adapters.**

Conclusion:

Viva Questions:

1 **What is the purpose of an Intent in Android?**
2 **How do you register and handle an event listener in Android?**
3 **Explain the role of an Adapter in ListViews.**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—8:
Aim: Programs on Services, notification and broadcast receivers
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand services and notifications in Android.
2. Learn to use broadcast receivers in Android.

THEORY : Programs on Services, notification and broadcast receivers

1. **Services**:

   - Services are components that run in the background to perform long-running operations or handle tasks without requiring a user interface.
   - They can be started with startService() and stopped with stopService(), or they can be bound to an activity using bindService() and unbindService().
   - Services are used for tasks such as playing music, downloading files, performing network operations, or handling asynchronous processing.

2. **Notifications**:

   - Notifications are used to provide timely reminders, updates, or alerts to users, even when the app is not in the foreground.
   - They can include text, icons, actions, and other interactive elements, and can be displayed as status bar icons, expanded notifications, or heads-up notifications.
   - Notifications are created using the NotificationManager system service and NotificationCompat.Builder class, allowing customization of content, appearance, and behavior.

3. **Broadcast Receivers**:

   - Broadcast receivers are components that respond to system-wide broadcast messages or intents, allowing apps to receive and react to system events or custom broadcasts.
   - They are registered either statically in the manifest file or dynamically at runtime using Context.registerReceiver().

- Broadcast receivers are commonly used to listen for events such as device boot completion, network connectivity changes, battery status updates, or custom intents sent by other apps.

Step:-

1. **Programs on Services:**
   Services are commands which are used by kotlin in functions to execute the task. They are : IntentService, onStartCommand(),onHandleIntent() etc.

2. **notification and broadcast receivers:**
   Step 1. Create an android app, For creating an Android app with kotlin read this tutorial.
   Step 2. Creating Broadcast Receiver Create and extend Subclass and BroadcastReceiver implement.onReceive(Context, Intent) where onReceive method each message is received as an Intent object parameter.

   MyReceiver.kt:

```
package `in`.eyehunt.androidbroadcasts

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

class MyReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        Toast.makeText(context, "Broadcast : Flight mode changed.",
                Toast.LENGTH_LONG).show()
    }
}
```

   Step 3. Declare a broadcast receiver in the manifest file add the element<receiver> in your app's manifest. Here is code snap

   AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="in.eyehunt.androidbroadcasts">

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/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=".MyReceiver"
android:enabled="true" android:exported="true">
<intent-filter>
<action android:name="android.intent.action.AIRPLANE_MODE"/>
</intent-filter>
</receiver>
</application>

</manifest>
```

Note: If the app is not running and broadcast receiver declared in AndroidManifest.xml, then the system will launch your app.

Step 4. MainActivity code, no needs to do anything

MainActivity.kt:

```
package `in`.eyehunt.androidbroadcasts

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

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)
    }
}
```

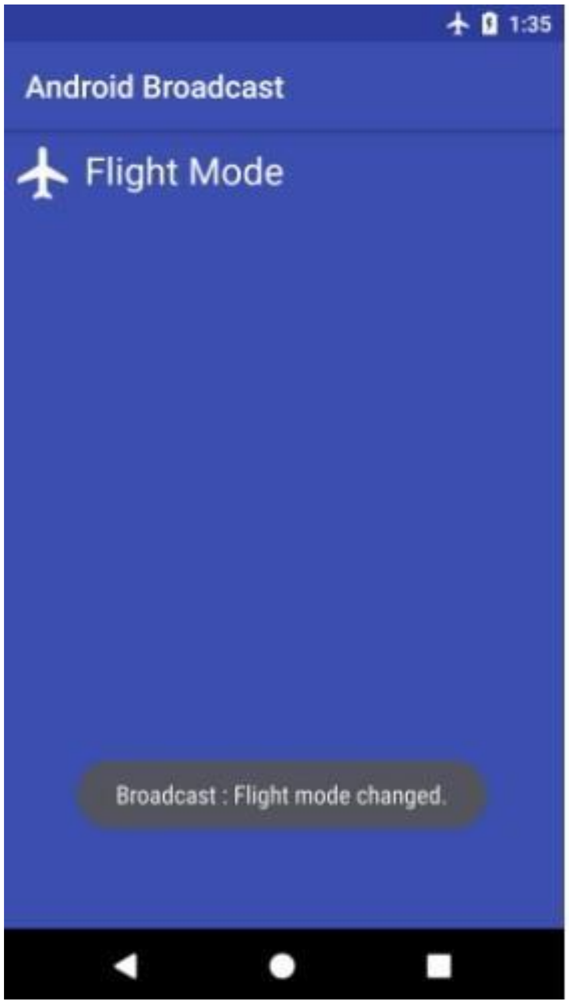
Step 5. Add following code in main_activity.xml add <ImageView> and <TextView>widget layout file.

main_activity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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="@color/colorPrimary"
tools:context="in.eyehunt.androidbroadcasts.MainActivity">
<ImageView
android:id="@+id/imageView" android:layout_width="40dp"
android:layout_height="40dp"
android:layout_margin="8dp"
android:layout_marginTop="16dp"
app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@mipmap/baseline_airplanemode_active_white_24" />

<TextView
android:id="@+id/textView"
```

```
android:layout_width="300dp" android:layout_height="36dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:gravity="center_vertical" android:text="Flight
Mode" android:textColor="@color/colorWhite"
android:textSize="24dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toEndOf="@+id/imageView"
app:layout_constraintTop_toTopOf="@+id/imageView" />
</android.support.constraint.ConstraintLayout>
```

Output:



Broadcast : Flight mode changed.

Learning Outcomes:

1 **Implement background services in Android apps.**

2 **Send and receive notifications effectively.**

Course Outcomes:

1 **Develop apps with background processing.**

2 **Manage broadcast receivers for app communication.**

3

Conclusion:

Viva Questions:

1 **What are the different types of services in Android?**

2 **How do you send a notification in Android?**

3 **What is the role of a broadcast receiver in Android?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—9
Aim: a. Database Programming with SQLite b. Programming Network Communications and Services (JSON)
T ools & Technologies used: Android Studio

Learning Objectives:

1. **Learn database handling with SQLite.**
2. **Understand network communication using JSON.**

THEORY : Database Programming with SQLite

1. **Introduction to SQLite**:
   - SQLite is a lightweight, embedded relational database engine that is built into Android devices.
   - It provides a simple and efficient mechanism for storing, querying, and managing structured data within Android applications.
   - SQLite databases are stored as files in the app's private storage or on external storage, accessible only to the app itself.
2. **Creating a Database**:
   - To create a SQLite database in an Android application, developers typically subclass the SQLiteOpenHelper class.
   - The SQLiteOpenHelper class provides methods for creating and upgrading the database schema, as well as for opening and closing dtabase connections.
3. **Defining Database Schema**:
   - The database schema defines the structure of tables and columns in the database.
   - Developers use SQL (Structured Query Language) statements to define tables, columns, constraints, and indexes.
   - The schema is typically defined in the onCreate() method of the SQLiteOpenHelper subclass.
4. **Performing Database Operations**:
   - Once the database is created and the schema is defined, developers can perform CRUD (Create, Read, Update, Delete) operations on the database.
   - SQLiteDatabase class provides methods for executing SQL queries,

- inserting, updating, deleting, and querying data from the database.

5. **Working with Cursors**:
    - Queries to the database return Cursor objects, which provide access to the result set returned by the query.

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/andr
oid" xmlns:app="http://schemas.android.com/apk/res-
auto" xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:gravity="center"
tools:context="com.tutorialkart.sqlitetutori
al.MainActivity">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="SQLite Tutorial
- User Management"
android:textSize="20dp"
android:padding="10dp" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
<EditText
android:id="@+id/edittext_use
rid" android:hint="User ID"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<EditText
android:id="@+id/edittext_name"
android:hint="User Name"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
<EditText
android:id="@+id/edittext_age"
android:hint="User Age"
android:gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">
<Button
```

```
android:id="@+id/button_ad
d_user"
android:layout_width="wrap
_content"
android:layout_height="wra
p_content"
android:layout_weight="1"
android:onClick="addUser"
android:text="Add" />

<
B
u
t
t
o
n
android:id="@+id/button_delete_user" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_weight="1"
```

```
<?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="wrap_content" android:orientation="vertical"
android:gravity="center"
tools:context="com.tutorialkart.sqlitetutorial.MainActivity">
<TextView
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:text="SQLite Tutorial - User Management"
android:textSize="20dp" android:padding="10dp"
/>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"  android:orientation="vertical">
<EditText
android:id="@+id/edittext_userid"
android:hint="User ID" android:gravity="center"
android:layout_width="match_parent" android:layout_height="wrap_content"
/>
<EditText
android:id="@+id/edittext_name"
android:hint="User Name" android:gravity="center"
android:layout_width="match_parent" android:layout_height="wrap_content"
/>
<EditText
android:id="@+id/edittext_age"
android:hint="User Age" android:gravity="center"
android:layout_width="match_parent" android:layout_height="wrap_content"
/>
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"  android:orientation="horizontal">
<Button
android:id="@+id/button_add_user"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:onClick="addUser" android:text="Add"
/>

<Button
android:id="@+id/button_delete_user"
android:layout_width="wrap_content"
android:layout_height="wrap_content"  android:layout_weight="1"
```

```
android:onClick="deleteUser"
```

```
package com.tutorialkart.sqlitetutorial

import android.content.ContentValues
import android.content.Context import
android.database.Cursor
import android.database.sqlite.SQLiteConstraintException
import android.database.sqlite.SQLiteDatabase import
android.database.sqlite.SQLiteException import
android.database.sqlite.SQLiteOpenHelper
android:text="Delete" />

<Button
android:id="@+id/button_show_all"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:onClick="showAllUsers" android:text="Show
All" />
</LinearLayout> <TextView
android:id="@+id/textview_result" android:layout_width="match_parent"
android:layout_height="wrap_content" />
<LinearLayout
android:id="@+id/ll_entries" android:padding="15dp"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content"></LinearLayout>
</LinearLayout>
```

UserModel.kt:

```
package com.tutorialkart.sqlitetutorial

class UserModel(val userid: String, val name: String, val age: String)
```

DBContract.kt

```
package com.tutorialkart.sqlitetutorial

import android.provider.BaseColumns

object DBContract {
```

```
import java.util.ArrayList

class UsersDBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
    override fun onCreate(db: SQLiteDatabase) {
db.execSQL(SQL_CREATE_ENTRIES)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
// This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
db.execSQL(SQL_DELETE_ENTRIES)          onCreate(db)
    }

    override fun onDowngrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
onUpgrade(db, oldVersion, newVersion)
    }

    @Throws(SQLiteConstraintException::class)
fun insertUser(user: UserModel): Boolean {
// Gets the data repository in write mode
val db = writableDatabase

        // Create a new map of values, where column names are the keys
val values = ContentValues()
        values.put(DBContract.UserEntry.COLUMN_USER_ID, user.userid)
values.put(DBContract.UserEntry.COLUMN_NAME, user.name)
values.put(DBContract.UserEntry.COLUMN_AGE, user.age)
        // Insert the new row, returning the primary key value of the new row
val newRowId = db.insert(DBContract.UserEntry.TABLE_NAME, null, values)

        return true
    }

    @Throws(SQLiteConstraintException::class)      fun
deleteUser(userid: String): Boolean {          // Gets the data
repository in write mode         val db = writableDatabase         //
Define 'where' part of query.         val selection =
DBContract.UserEntry.COLUMN_USER_ID + " LIKE ?"
        // Specify arguments in placeholder order.
        val selectionArgs = arrayOf(userid)
        // Issue SQL statement.
        db.delete(DBContract.UserEntry.TABLE_NAME, selection, selectionArgs)

        return true
    }

    fun readUser(userid: String): ArrayList<UserModel> {
val users = ArrayList<UserModel>()          val db =
writableDatabase          var cursor: Cursor? = null
try {
            cursor = db.rawQuery("select * from " + DBContract.UserEntry.TABLE_NAME + " WHERE " +
DBContract.UserEntry.COLUMN_USER_ID + "='" + userid + "'", null)
        } catch (e: SQLiteException) {
            // if table not yet present, create it
db.execSQL(SQL_CREATE_ENTRIES)              return
ArrayList()
        }

        var name: String
var age: String
        if (cursor!!.moveToFirst()) {
            while (cursor.isAfterLast == false) {
```

```
    /* Inner class that defines the table contents */
class UserEntry : BaseColumns {          companion object {
val TABLE_NAME = "users"              val COLUMN_USER_ID =
"userid"              val COLUMN_NAME = "name"              val
COLUMN_AGE = "age"
                age = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_AGE))

                users.add(UserModel(userid, name, age))
cursor.moveToNext()
            }
}
        return users
    }

    fun readAllUsers(): ArrayList<UserModel> {
val users = ArrayList<UserModel>()       val db
= writableDatabase         var cursor:
Cursor? = null         try {
            cursor = db.rawQuery("select * from " + DBContract.UserEntry.TABLE_NAME, null)
        } catch (e: SQLiteException) {
db.execSQL(SQL_CREATE_ENTRIES)              return
ArrayList()
        }

        var userid: String
var name: String         var
age: String
        if               (cursor!!.moveToFirst()) {
while (            cursor  .isAfterLast == false) {
userid =


cursor
```

```
            .getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_USER_ID))
             cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_NAME))
            cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_AGE))

                            users.add(UserModel(userid, name, age))
                      cursor.moveToNext()

    }          }
               }
             return users


            mpanion object {
              // If you change the database schema, you must increment the database
              version. val DATABASE_VERSION = 1
              val DATABASE_NAME = "FeedReader.db"

              private val SQL_CREATE_ENTRIES =
                  "CREATE TABLE " + DBContract.UserEntry.TABLE_NAME + " (" +
                          DBContract.UserEntry.COLUMN_USER_ID + " TEXT PRIMARY KEY," +
                          DBContract.UserEntry.COLUMN_NAME + " TEXT," +
                          DBContract.UserEntry.COLUMN_AGE + " TEXT)"

                            private val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS " +
    }                                     DBContract.UserEntry.TABLE_NAME

}
package com.tutorialkart.sqlitetutorial

import android.support.v7.app.AppCompatActivity
import android.os.Bundle import
android.view.View import
android.widget.TextView
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
        }
    }
}
```

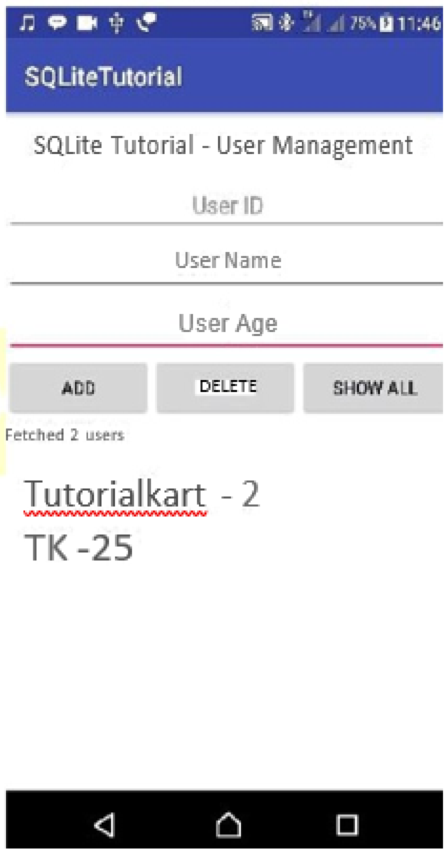
UserDBHelper.kt:

```
                name = cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_NAME))
```

MainActivity.kt:

```
    lateinit var usersDBHelper : UsersDBHelper

    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)

        usersDBHelper = UsersDBHelper(this)
    }

    fun addUser(v:View){
        var userid = this.edittext_userid.text.toString()
var name = this.edittext_name.text.toString()         var
age = this.edittext_age.text.toString()
        var result = usersDBHelper.insertUser(UserModel(userid = userid,name = name,age = age))
        //clear all edittext s
this.edittext_age.setText("")
this.edittext_name.setText("")
this.edittext_userid.setText("")
        this.textview_result.text = "Added user : "+result
this.ll_entries.removeAllViews()
    }

    fun deleteUser(v:View){
        var userid = this.edittext_userid.text.toString()
val result = usersDBHelper.deleteUser(userid)
this.textview_result.text = "Deleted user : "+result
this.ll_entries.removeAllViews()
    }

    fun showAllUsers(v:View){
        var users = usersDBHelper.readAllUsers()
this.ll_entries.removeAllViews()          users.forEach
{
            var tv_user = TextView(this)
tv_user.textSize = 30F
            tv_user.text = it.name.toString() + " - " + it.age.toString()
this.ll_entries.addView(tv_user)
        }
        this.textview_result.text = "Fetched " + users.size + " users"
    }
}
```

output:



SQLite Tutorial - User Management

Fetched 2 users

Tutorialkart - 2
TK -25

Learning Outcomes:

1 Create and manage SQLite databases.
2 Implement network communication with JSON.

Course Outcomes:

1 Develop apps with persistent data storage.
2 Integrate web services using JSON in apps.

Conclusion:

Viva Questions:

1 How do you perform CRUD operations in SQLite?
2 What is JSON and how is it used in Android?
3 How do you handle network communication in Android using JSON?

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—10:
Aim: Programming threads, handles and asynchronized programs
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand threading and handlers in Android.
2. Implement asynchronous programming techniques.

THEORY : Programming Security and permissions

1. **Android Security Model**:
    - Android has a layered security model designed to protect user data, system resources, and user privacy.
    - The security model includes sandboxing, permission-based access control, secure inter-process communication, and cryptographic APIs for data encryption.
2. **Permissions**:
    - Permissions are safeguards that control access to sensitive resources or functionality on the device.
    - Android defines a set of permissions, such as READ_EXTERNAL_STORAGE, CAMERA, INTERNET, etc., which apps must request in their manifest file to access corresponding features.
    - Permissions can be categorized into normal permissions, which are automatically granted at installation time, and dangerous permissions, which require user approval at runtime on Android 6.0 (API level 23) and above.
3. **Requesting Permissions**:
    - Apps must request dangerous permissions at runtime on Android 6.0 (API level 23) and above, using the requestPermissions() method.
    - Before requesting permissions, apps should check whether the permission has already been granted using the checkSelfPermission() method.
    - After requesting permissions, apps must handle the onRequestPermissionsResult() callback to process user responses.
4. **Handling Permissions**:
    - Apps should explain why they need a particular permission to users before requesting it, using informative dialogs or explanations.

- If the user denies a permission request, the app should gracefully handle the situation and adjust its behavior accordingly, possibly by disabling certain features or providing alternative functionality.

Step:-

Extra Packages requied in ManagePermission.kt (Class File)

```
import android.app.Activity
import android.content.pm.PackageManager import
android.support.v4.app.ActivityCompat    import
android.support.v4.content.ContextCompat import
android.support.v7.app.AlertDialog
```

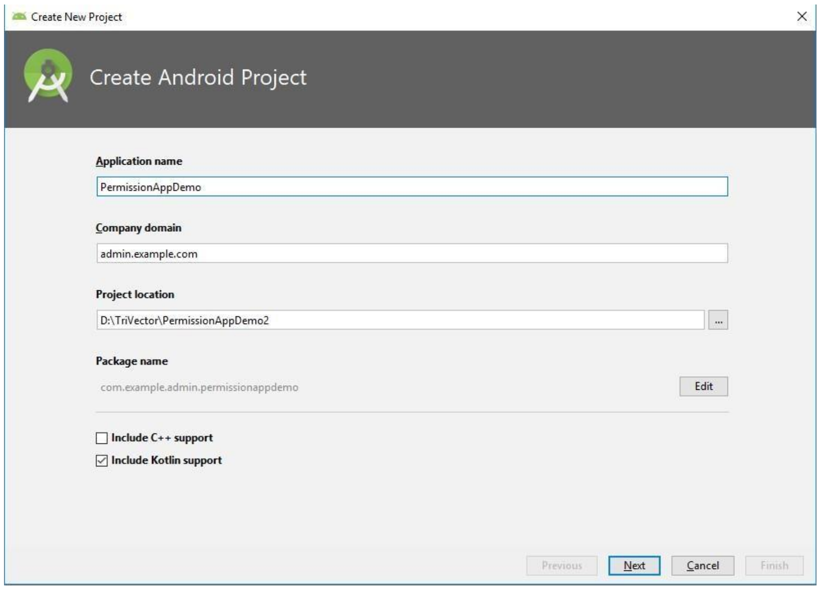
Extra Packages requied in MainActivity.kt

```
import android.Manifest
import
android.content.Context
import android.os.Build
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*
```

For Multple Permission Access,need to add following line in class MainActivity

```
private val PermissionsRequestCode = 123
```

1. **Create a new project in android studio**



Application name
PermissionAppDemo

Company domain
admin.example.com

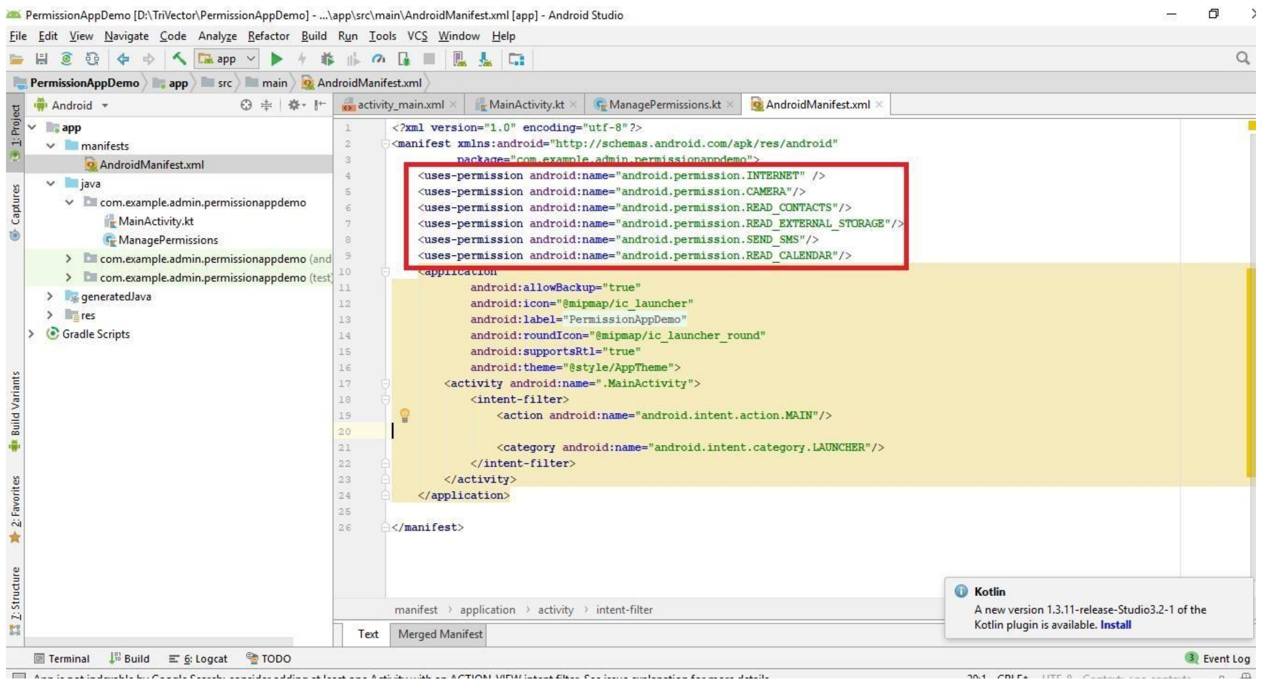
Project location
D:\TriVector\PermissionAppDemo2

Package name
com.example.admin.permissionappdemo

☐ Include C++ support
☑ Include Kotlin support

2. An app must publicize the permissions it requires by including <uses-permission> tags in the app manifest.

```
<uses-permission android:name="android.permission.INTERNET"  />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
```


PermissionAppDemo [D:\TriVector\PermissionAppDemo] - ...\app\src\main\AndroidManifest.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PermissionAppDemo › app › src › main › AndroidManifest.xml
Android
app
manifests
AndroidManifest.xml
java
com.example.admin.permissionappdemo
MainActivity.kt
ManagePermissions
com.example.admin.permissionappdemo (and
com.example.admin.permissionappdemo (test)
generatedJava
res
Gradle Scripts
activity_main.xml
MainActivity.kt
ManagePermissions.kt
AndroidManifest.xml
manifest › application › activity › intent-filter
Text
Merged Manifest
Terminal
Build
6: Logcat
TODO
Event Log
1: Project
Captures
Build Variants
2: Favorites
Z: Structure

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.admin.permissionappdemo">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CALENDAR"/>
    <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="PermissionAppDemo"
            android:roundIcon="@mipmap/ic_launcher_round"
            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>
    </application>

</manifest>
```

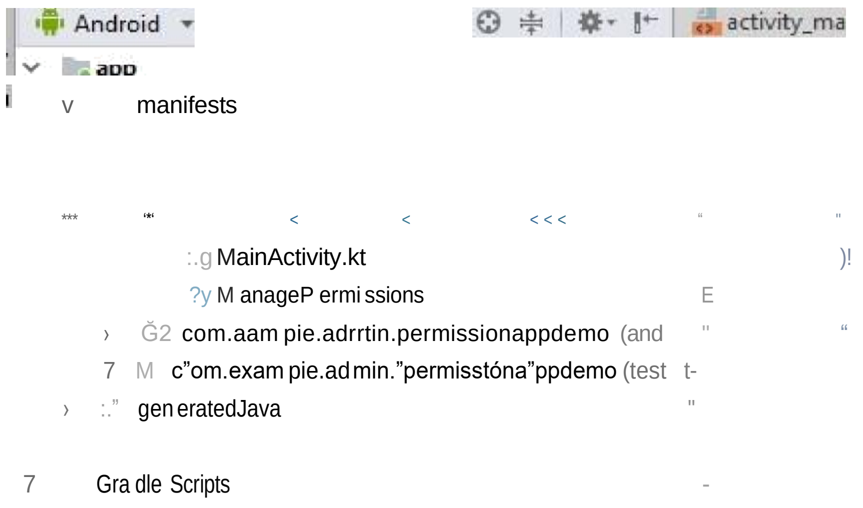
Kotlin
A new version 1.3.11-release-Studio3.2-1 of the Kotlin plugin is available. Install

3. MainActivity.kt

```
package com.example.admin.permissionappdemo

import android.Manifest import
android.content.Context import
android.os.Build
import android.support.v7.app.AppCompatActivity
import android.os.Bundle import
android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
private val PermissionsRequestCode = 123
    private lateinit var managePermissions: ManagePermissions
    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)          setContentView(R.layout.activity_main)


        // Initialize a list of required permissions to request runtime
val list = listOf<String>(
            Manifest.permission.CAMERA,
            Manifest.permission.READ_CONTACTS,
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.SEND_SMS,
            Manifest.permission.READ_CALENDAR
        )

        // Initialize a new instance of ManagePermissions class
        managePermissions = ManagePermissions(this,list,PermissionsRequestCode)
// Button to check permissions states          button.setOnClickListener{
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
managePermissions.checkPermissions()          }
    }



    // Receive the permissions request result
    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String>,
grantResults: IntArray) {          when (requestCode) {
            PermissionsRequestCode ->{                  val
isPermissionsGranted = managePermissions
                    .processPermissionsResult(requestCode,permissions,grantResults)

                if(isPermissionsGranted){
                    // Do the task now
toast("Permissions granted.")
                }else{
                    toast("Permissions denied.")
                }
return
            }
        }
    }
}



// Extension function to show toast message fun
Context.toast(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}
```

4. Create a New Kotlin Class

app->src->main->java->com.example.admin.permissionappdemo



:.g MainActivity.kt

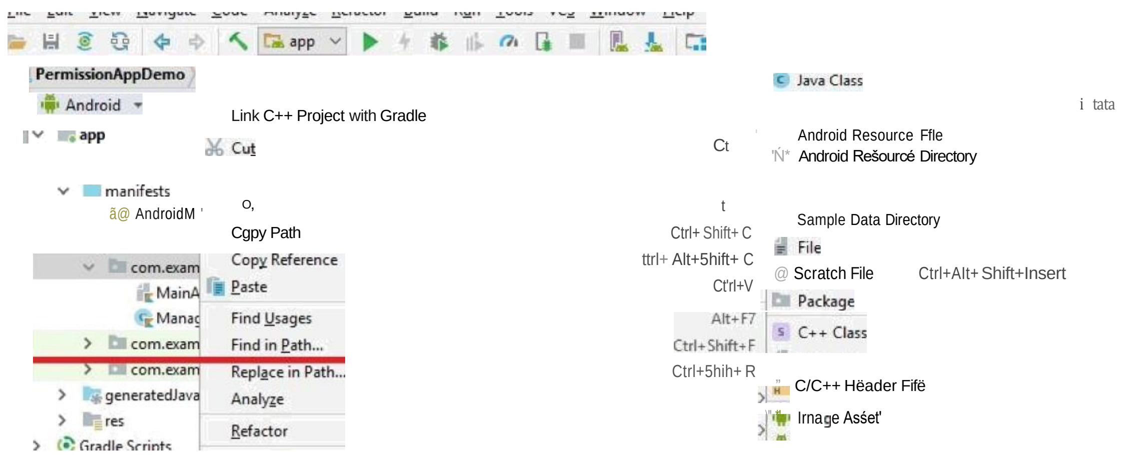
?y M anageP ermi ssions

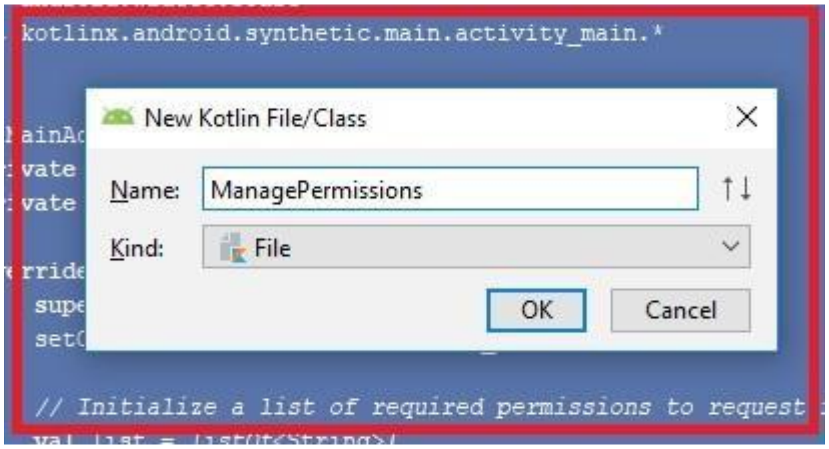
› Ğ2 com.aam pie.adrrtin.permissionappdemo (and

7 M c"om.exam pie.ad min."permisstóna"ppdemo (test t-

› :." gen eratedJava

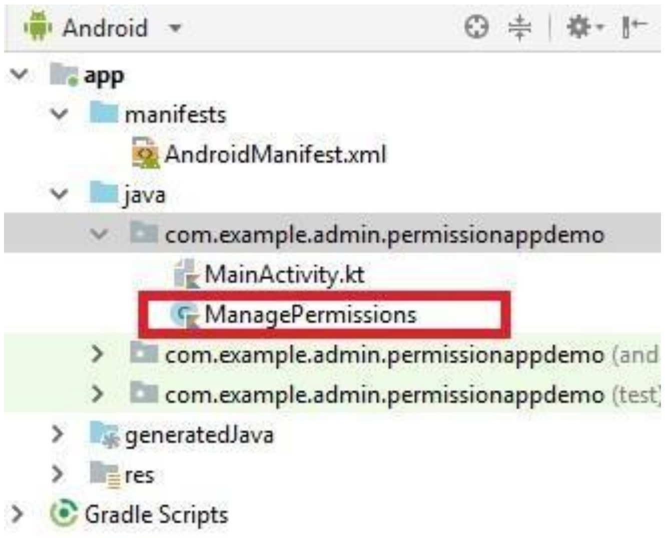
7 Gra dle Scripts




kotlinx.android.synthetic.main.activity_main.*

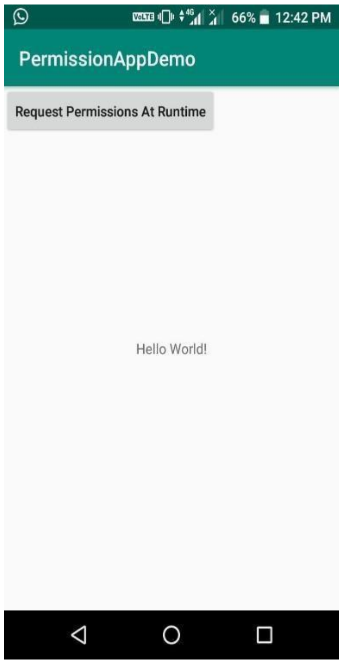
// Initialize a list of required permissions to request

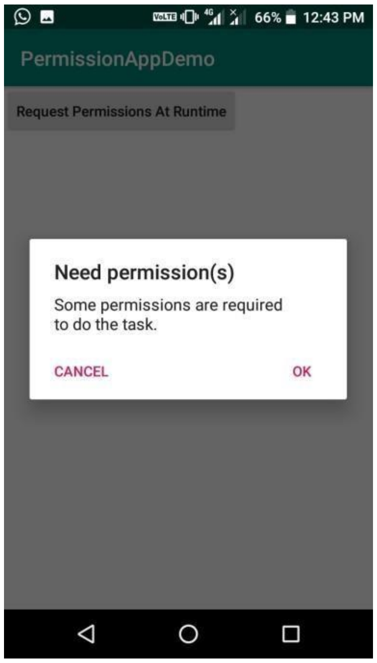
Class file is generated


Android
app
manifests
AndroidManifest.xml
java
com.example.admin.permissionappdemo
MainActivity.kt
ManagePermissions
com.example.admin.permissionappdemo (and
com.example.admin.permissionappdemo (test)
generatedJava
res
Gradle Scripts

5. Write the following code in the Class File

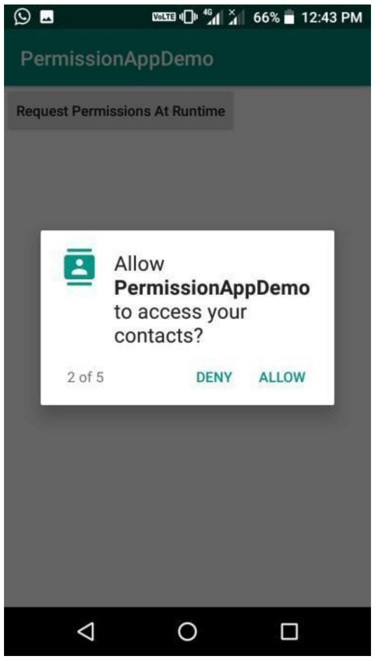
```
import android.app.Activity import
android.content.pm.PackageManager import
android.support.v4.app.ActivityCompat import
android.support.v4.content.ContextCompat import
android.support.v7.app.AlertDialog

class ManagePermissions(val activity: Activity,val list: List<String>,val code:Int) {
    // Check permissions at runtime
fun checkPermissions() {
        if (isPermissionsGranted() != PackageManager.PERMISSION_GRANTED) {
showAlert()           } else {
            activity.toast("Permissions already granted.")
        }
    }



    // Check permissions status
    private fun isPermissionsGranted(): Int {
// PERMISSION_GRANTED : Constant Value: 0
// PERMISSION_DENIED : Constant Value: -1
var counter = 0;          for (permission in list)
{
            counter += ContextCompat.checkSelfPermission(activity, permission)
        }
return counter
    }



    // Find the first denied permission
private fun deniedPermission(): String {
for (permission in list) {
            if (ContextCompat.checkSelfPermission(activity, permission)
== PackageManager.PERMISSION_DENIED) return permission
        }
return ""
    }



    // Show alert dialog to request permissions
private fun showAlert() {
        val builder = AlertDialog.Builder(activity)
builder.setTitle("Need permission(s)")
        builder.setMessage("Some permissions are required to do the task.")
builder.setPositiveButton("OK", { dialog, which -> requestPermissions() })
builder.setNeutralButton("Cancel", null)          val dialog = builder.create()
dialog.show()
    }



    // Request the permissions at run time
private fun requestPermissions() {
val permission = deniedPermission()
        if (ActivityCompat.shouldShowRequestPermissionRationale(activity, permission)) {
            // Show an explanation asynchronously
activity.toast("Should show an explanation.")
        } else {
            ActivityCompat.requestPermissions(activity, list.toTypedArray(), code)
        }
    }



    // Process permissions result
    fun processPermissionsResult(requestCode: Int, permissions: Array<String>,
grantResults: IntArray): Boolean {          var result = 0
        if (grantResults.isNotEmpty()) {
for (item in grantResults) {
result += item
            }
}
        if (result == PackageManager.PERMISSION_GRANTED) return true
return fa
```





Need permission(s)

Some permissions are required to do the task.



Allow **PermissionAppDemo** to access your contacts?

Learning Outcomes:

1 **Use threads and handlers for background tasks.**
2 **Develop apps with asynchronous operations.**
3

Course Outcomes:

1 **Improve app performance with threading.**
2 **Handle asynchronous tasks efficiently in Android.**

Conclusion:

Viva Questions:

1 **What is the role of a Handler in Android?**
2 **How do you manage threads in Android?**
3 **What is the difference between synchronous and asynchronous programming?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Experiment No.—11:
Aim: a. Programming Media API and Telephone API b. Programming Security and permissions
Tools & Technologies used: Android Studio

Learning Objectives:

1. Understand media and telephone APIs.
2. Learn about security and permissions in Android.

THEORY : Programming Network Communications and Services (JSON)

1. **Networking in Android**:
   - Android provides built-in classes and APIs for performing network operations, such as making HTTP requests and handling responses.
   - Commonly used classes for networking include HttpURLConnection, HttpClient (deprecated in API level 22), and third-party libraries like OkHttp, Retrofit, and Volley.
   - Networking operations should be performed asynchronously on background threads to avoid blocking the main UI thread and ensure a responsive user experience.
2. **HTTP Requests**:
   - HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web. In Android, developers use HTTP to send requests to web servers and receive responses.
3. **JSON (JavaScript Object Notation)**:
   - JSON is a lightweight data interchange format commonly used for transmitting data between a web server and a client application.
   - In Android, JSON data is typically received as a string in the HTTP response body, and developers use the built-in JSONObject and JSONArray classes to parse and manipulate the JSON data.
   - JSONObject represents an unordered collection of name/value pairs (similar to a dictionary or map), while JSONArray represents an ordered collection of values (similar to a list or array).
4. **JSON Parsing**:
   - JSON parsing in Android involves converting a JSON string into corresponding Java objects for further processing.
   - Developers can use the JSONObject class to parse JSON objects and extract individual values using methods like getString(), getInt(), getJSONObject(), and getJSONArray().

- Similarly, JSONArray class is used to parse JSON arrays and access individual elements by index.

5. **Network Services**:

- Network services in Android are background components responsible for performing network-related tasks, such as downloading files, synchronizing data, or fetching updates from a server.

Step:-

1. Handling connectivity errors in Android apps with Kotlin:

Open your build.gradle file and add the following dependencies:

implementation 'com.android.support:design:27.1.1'
implementation 'com.squareup.retrofit2:retrofit:2.3.0'
implementation 'com.squareup.retrofit2:converter-scalars:2.3.0'

Open your AndroidManifest.xml file and add the permissions like so:

```
  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.android.internetconnectivity">

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

[...]

</manifest>
```

When there is a network connection, we will fetch data from an API. Let's set up an interface to hold the endpoints we will access. Create a new Kotlin file named ApiService and paste this:

```
import retrofit2.Call import
retrofit2.http.GET

interface ApiService {
    @GET(".")
    fun getFeeds(): Call<String> }
```

For this demo, we are only going to access one endpoint, which is equivalent to our base URL. It's for this reason we used a dot instead of the usual /some-url in the @GET annotation.

When these items are fetched, we will display the items in a list. We, therefore, need a RecyclerView in the layout and a matching adapter. Create a new Kotlin file named RecyclerAdapter and paste this:

```
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater import
android.view.View import
android.view.ViewGroup import
android.widget.TextView

class RecyclerAdapter : RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {

    private var list = ArrayList<String>()

    fun setItems(newList: ArrayList<String>){
this.list = newList
        this.notifyDataSetChanged()
    }

    override fun getItemCount() = list.size

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {          val view = LayoutInflater.from(parent.context)
                .inflate(android.R.layout.simple_list_item_1, parent, false)

        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
holder.textView.text = list[position]
    }

    inner class ViewHolder(itemView: View?): RecyclerView.ViewHolder(itemView) {
var textView: TextView = itemView!!.findViewById(android.R.id.text1)
    }

}
```

he adapter handles the display of items on a list. It has some overridden methods like:

getItemCount – to tell the size of the list to be populated. onCreateViewHolder – used to choose a layout for a list row.
onBindViewHolder – to bind data to each row depending on the position, etc.
Next, we will update the layout of our MainActivity's activity_main.xml file like so:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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"
tools:context=".MainActivity">

<android.support.v7.widget.RecyclerView android:layout_width="match_parent"
android:layout_height="match_parent" android:id="@+id/recyclerView"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<ImageView
android:id="@+id/imageView" android:layout_width="match_parent"
android:layout_height="wrap_content"
android:src="@drawable/no_internet_connection" />

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

The layout contains a RecyclerView for our list items and an ImageView to show an error message.

We also need an error message image. Once you have an image, rename the file to no_internet_connection and save it to your drawable folder: NameOfProject/app/src/main/res/drawable. For us to monitor when the connectivity changes, we need broadcast receivers. Broadcast receivers are components that allow you to register and listen to Android system and application events. Usually, the Android system sends broadcast events when various system events occur and your app needs to register to get these events.

Let’s register a listener to be triggered when the internet connection is online or offline. Open your MainActivity file and paste the following code:

```
import android.content.BroadcastReceiver import
android.content.Context import
android.content.Intent import
android.content.IntentFilter import
android.net.ConnectivityManager import
android.support.v7.app.AppCompatActivity import
android.os.Bundle
import android.support.v7.widget.LinearLayoutManager
import android.util.Log import android.view.View
import kotlinx.android.synthetic.main.activity_main.*
import okhttp3.OkHttpClient import
org.json.JSONObject import retrofit2.Call import
retrofit2.Callback import retrofit2.Response import
retrofit2.Retrofit
import retrofit2.converter.scalars.ScalarsConverterFactory

class MainActivity : AppCompatActivity() {

    private val arrayList = ArrayList<String>()
private val adapter = RecyclerAdapter()     private
val retrofit = Retrofit.Builder()
.baseUrl("https://api.reddit.com/")
             .addConverterFactory(ScalarsConverterFactory.create())
             .client(OkHttpClient.Builder().build())
             .build()

    private var broadcastReceiver: BroadcastReceiver = object : BroadcastReceiver() {
override fun onReceive(context: Context, intent: Intent) {             val
notConnected = intent.getBooleanExtra(ConnectivityManager
                    .EXTRA_NO_CONNECTIVITY, false)
if (notConnected) {                  disconnected()
             } else {
connected()
             }
         }


    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
setupRecyclerView()
    }

}
```

Above, we initialized some variables:

arrayList – we will add fetched items to this list.

adapter – this is the instance of the adapter class. retrofit

– a Retrofit instance.

broadcastReciever – this instance implements the onRecieve callback. This callback method is called when the system has notified us of a change in the network connection. In the callback, we then check to know the connectivity status thereby calling either a private connected or disconnected function.

After creating the broadcast receiver, we have to register it to get updates and unregister if there are no more activities. To do this, add the following functions to the code above in the MainActivity:

```
override fun onStart() {
super.onStart()
    registerReceiver(broadcastReceiver, IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION))
}

override fun onStop() {
super.onStop()
    unregisterReceiver(broadcastReceiver)  }
```

In the onCreate function, we set up our RecyclerView by calling the setupRecyclerView. Create a private function in the MainActivity class and set it up like this:

```
private fun setupRecyclerView(){
with(recyclerView){
        layoutManager = LinearLayoutManager(this@MainActivity)
adapter = this@MainActivity.adapter      }
}
```

Remember we mentioned the connected and disconnected functions earlier in this post. We will now add them to the class. Add them to the MainActivity file like so:

```
private fun disconnected() {
    recyclerView.visibility = View.INVISIBLE      imageView.visibility =
View.VISIBLE
}

private fun connected() {
    recyclerView.visibility = View.VISIBLE      imageView.visibility =
View.INVISIBLE      fetchFeeds()
}
```

The disconnected function is called when there is no network connection. It hides the RecyclerView and shows the ImageView. The connected function is called when there is an active internet connection. It shows the RecyclerView, hides the ImageView, and finally calls the fetchFeeds function.

Next, in the same file, paste the following code:

```
private fun fetchFeeds() {
    retrofit.create(ApiService::class.java)
            .getFeeds()
            .enqueue(object : Callback<String> {
                override fun onFailure(call: Call<String>, t: Throwable) {
                    Log.e("MainActivityTag", t.message)
                }

                override fun onResponse(call: Call<String>?, response:
Response<String>) {                         addTitleToList(response.body()!!)
                }


            })
}
```

This function calls the API to get data. When the call is successful, we have another function that helps us add the title of the posts gotten from the endpoint to our list and then to our adapter. Create a function named addTitleToList and set it up like so:

```
private fun addTitleToList(response: String) {
    val jsonObject = JSONObject(response).getJSONObject("data")
val children = jsonObject.getJSONArray("children")

    for (i in 0..(children.length()-1)) {
        val item = children.getJSONObject(i).getJSONObject("data").getString("title")
arrayList.add(item)
        adapter.setItems(arrayList)
    }
}
```

Learning Outcomes:

1 **Implement media features like audio and video.**

2 **Handle telephony features like calls and SMS.**

3

Course Outcomes:

1 **Develop apps with media and telephony capabilities.**

2 **Manage app security and permissions effectively.**

Conclusion:

Viva Questions:

1 **How do you implement audio playback using the Media API in Android?**

2 **How can you send an SMS programmatically in Android?**

3 **What are the key steps to request and handle permissions in Android?**

For Faculty Use

Correction Parameter	Formative Assessment [ ]	Timely Completion Of practical [ ]	Attendance learning Attitude [ ]

Prof. Name: Ms. Akanksha Mishra	**Class / Sem: TYBsc.IT/ VI (2024-2025)**
Course Code: USIT6P6	**Subject Name: Android Programming**

Sr. No.	Date	Index	Page	Sign
1.		**Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals:** Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple "Hello World" program.		
2.		**Programming Resources** Android Resources: (Color, Theme, String, Drawable, Dimension, Image)		
3.		**Programming Activities and fragments** Activity Life Cycle, Activity methods, Multiple Activities, Life Cycle of fragments and multiple fragments		
4.		**Programs related to different Layouts** Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View.		
5.		**Programming UI elements** AppBar, Fragments, UI Components		
6.		**Programming menus, dialog, dialog fragments**		
7.		**Programs on Intents, Events, Listeners and Adapters** The Android Intent Class, Using Events and Event Listeners		
8.		**Programs on Services, notification and broadcast receivers**		
9.		**a.** Database Programming with SQLite **b**. Programming Network Communications and Services (JSON)		
10.		**Programming threads, handles and asynchronized programs**		
11.		**a**. Programming Media API and Telephone API **b.** Programming Security and permissions		

SHRI. G.P.M. DEGREE COLLEGE OF SCIENCE AND COMMERCE

2024 – 2025

Name: ____

Department: ___________________________________

Class: ____

Roll No: ______________________________________

SHRI G.P.M. DEGREE COLLEGE OF SCIENCE & COMMERCE

Rajarshi Sahu Maharaj Marg, Telli Galli, Andheri (E), Mumbai -400069,
Tel.: 8928387199

CERTIFICATE

This is to certify that Mr/Ms. ______-___________________________________a student of **T.Y.B.Sc IT** Roll No. ____________________has completed the required number of practical in the subject of ___________________________________ as prescribed by the **UNIVERSITY OF MUMBAI** under my supervision during the academic year 2024-2025.

Signatories:

Subject In-charge (Name & Sign) : ______________________

Principal Sign (Name & Sign) : _________________________

External Examiner (Name & Sign) : ______________________

Date: __________________________

College Stamp