**Practical 1: Android Development Environment: To study design aspects of development environment like Android, Ios.**

**Android Development Environment:-**

1. Programming Languages

* Java: The primary language for Android development for many years.
* Kotlin: Officially supported by Google and preferred for new Android applications due to its modern features and interoperability with Java.

1. Integrated Development Environment (IDE)

* Android Studio: The official IDE for Android development, based on IntelliJ IDEA. It includes features like code completion, debugging, and a powerful layout editor.

1. Software Development Kit (SDK)

* Android SDK: Provides the tools necessary to build, test, and debug Android applications. Includes the Android emulator, platform tools, and various libraries.

1. Build System

* Gradle: The build automation tool used in Android Studio to compile, package, and manage dependencies for Android projects.

1. Emulator

* Android Emulator: Part of the Android SDK, it allows developers to run and test applications on virtual devices with different configurations and Android versions.

1. User Interface Design

* XML Layouts: UI components are typically defined in XML files.
* Jetpack Compose: A modern toolkit for building native UI using declarative programming.

1. Frameworks and Libraries

* Android Jetpack: A set of libraries, tools, and guidance to help developers write highquality apps more easily. Includes components like LiveData, ViewModel, Room, Navigation, and WorkManager.

1. Testing

* Junit: For unit testing.
* Espresso: For UI testing.
* Robolectric: For running tests on the JVM without an emulator.

**Ios Development Environment:-**

1. Programming Languages

* Objective-C: An older language used for Ios development.
* Swift: The modern language for Ios development, designed to be safe, fast, and expressive.

1. Integrated Development Environment (IDE)

* Xcode: The official IDE for Ios development, which includes a code editor, debugger, and Interface Builder for designing user interfaces.

1. Software Development Kit (SDK)

* Ios SDK: Provides the necessary tools and frameworks for developing Ios applications. Includes libraries like UIKit, Foundation, and Core Data.

1. Build System

* Xcode Build System: Integrated into Xcode for compiling, packaging, and managing dependencies.

1. Simulator

* Ios Simulator: Allows developers to test and debug applications on virtual devices with different Ios versions and device configurations.

1. User Interface Design

* Storyboard and XIB files: Interface Builder within Xcode is used to design Uis visually.
* SwiftUI: A modern declarative framework for building UI across all Apple platforms.

1. Frameworks and Libraries

* Cocoa Touch: The application development environment for Ios, including frameworks like UIKit, Foundation, and Core Graphics.
* Combine: A framework for handling asynchronous events by combining eventprocessing operators.

1. Testing

* XCTest: For unit and UI testing.
* Quick/Nimble: Popular third-party frameworks for behavior-driven development (BDD).

**Comparative Analysis**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Android Development** | **iOS Development** |
| Development Tools | Android Studio | Xcode |
| Programming Languages | Java, Kotlin | Swift, Objective-C |
| Frameworks | Jetpack, Retrofit, Dagger | SwiftUI, Combine, Alamofire |
| UI Design Tools | Layout Editor | Interface Builder |
| Virtual Device | Android Virtual Device (AVD) | iOS Simulator |
| Dependency Management | Gradle | CocoaPods, Swift Package Manager |
| Performance | Good with flexibility | Excellent with optimized hardware integration |

**Practical 2: Android Development Environment: To setup Android studio2 and study its basic components**

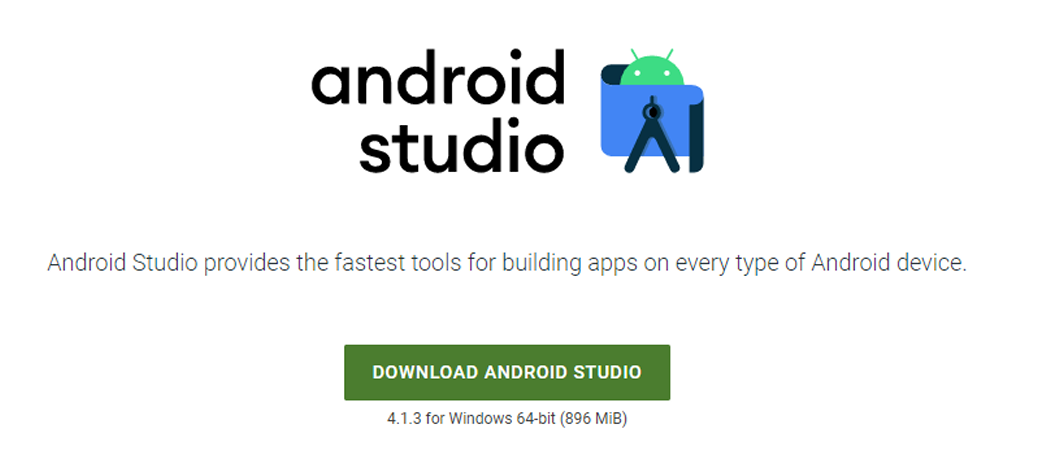
**Android Studio System Requirements for Windows**

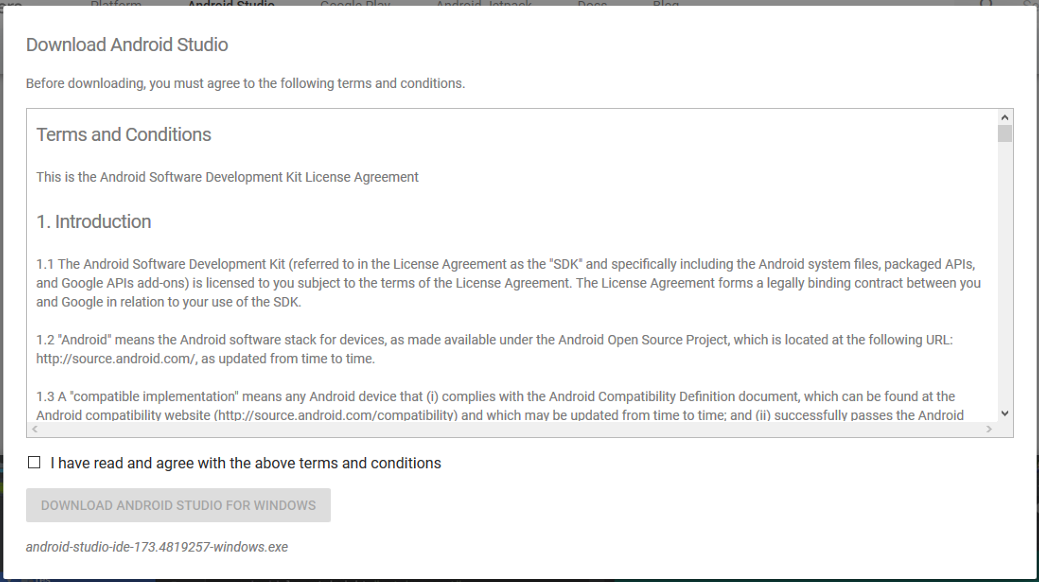
Microsoft Windows 7/8/10 (32-bit or 64-bit)

* 4 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
* 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus
* 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution

**Steps to Install Android Studio on Windows**

* Step 1: Head over to this link to get the Android Studio executable or zip file.
* Step 2: Click on the Download Android Studio Button

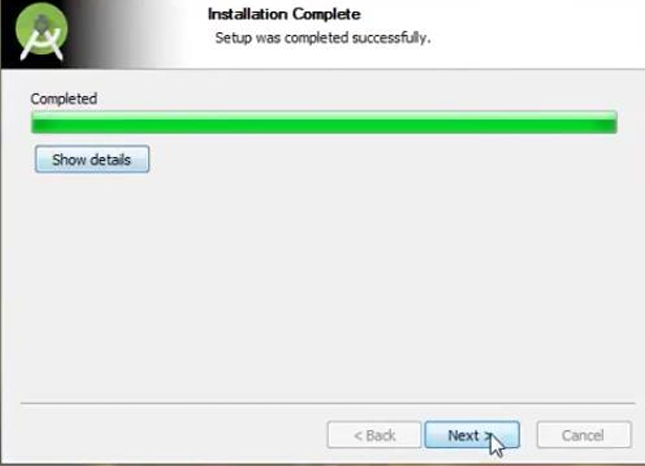




* Step 3: After the downloading has finished, open the file from downloads and run it. It will prompt the following dialog box.



* Step 4: It will start the installation, and once it is completed, it will be like the image shown below.





* Step 5: Once “Finish” is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the ‘Don’t import Settings option’.



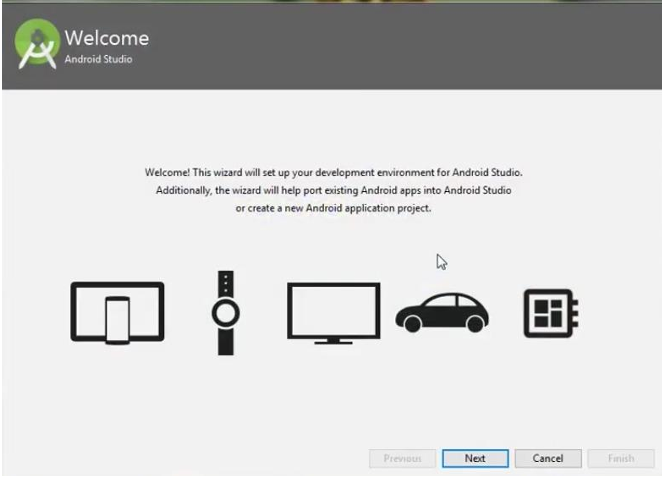
* Step 6: This will start the Android Studio.



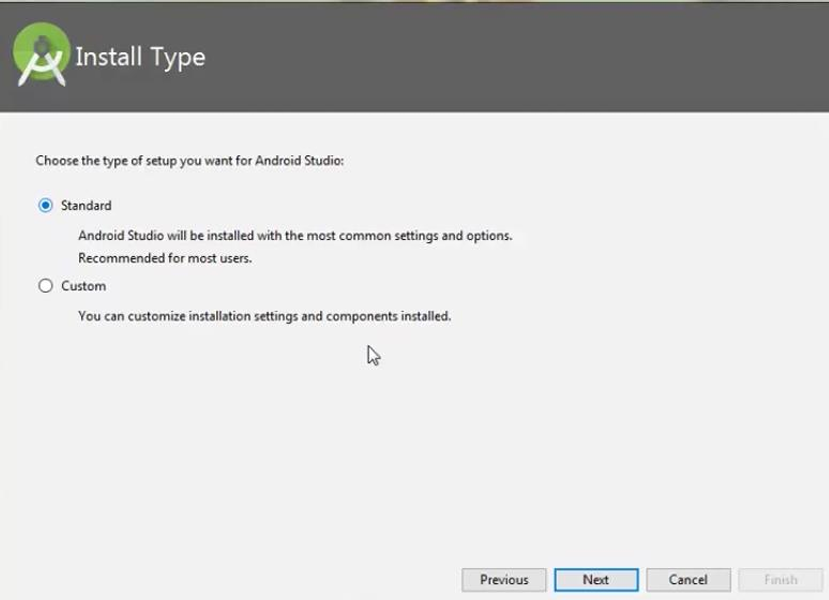
Meanwhile, it will be finding the available SDK components.



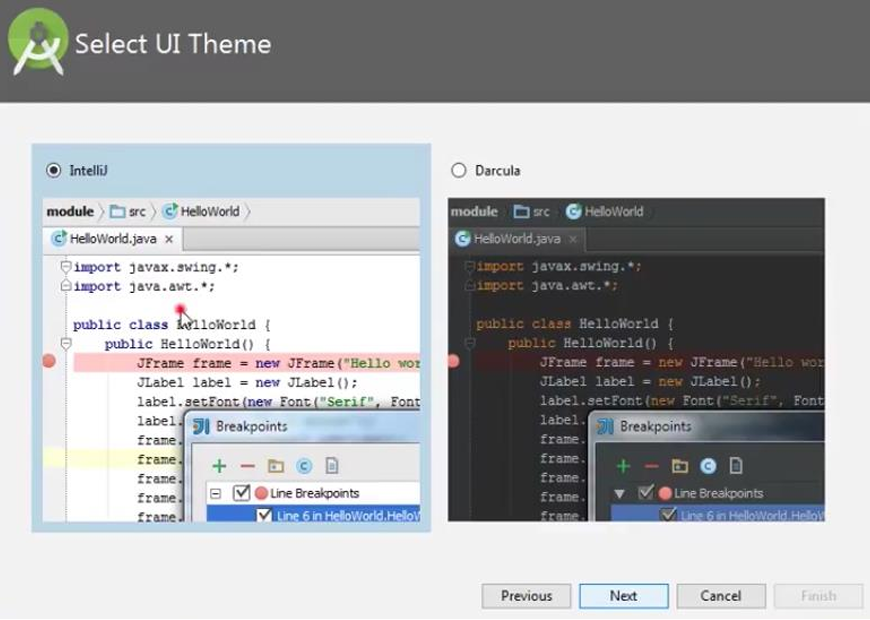
* Step 7: After it has found the SDK components, it will redirect to the Welcome dialog box.



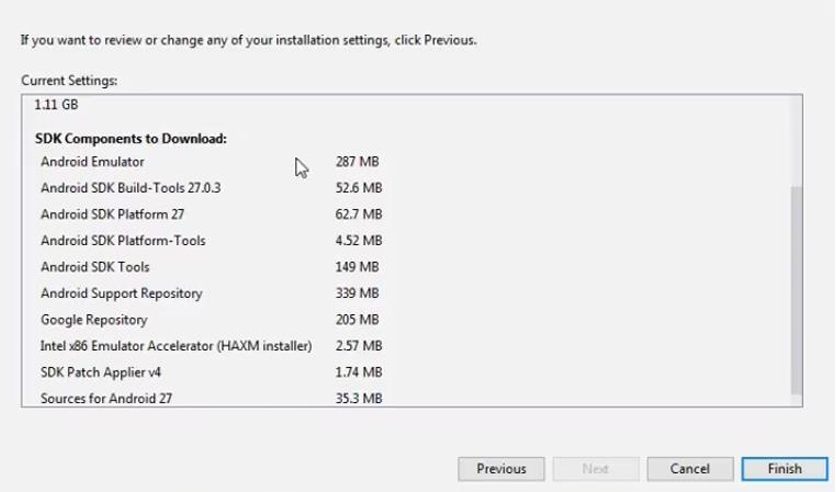
Click on Next.



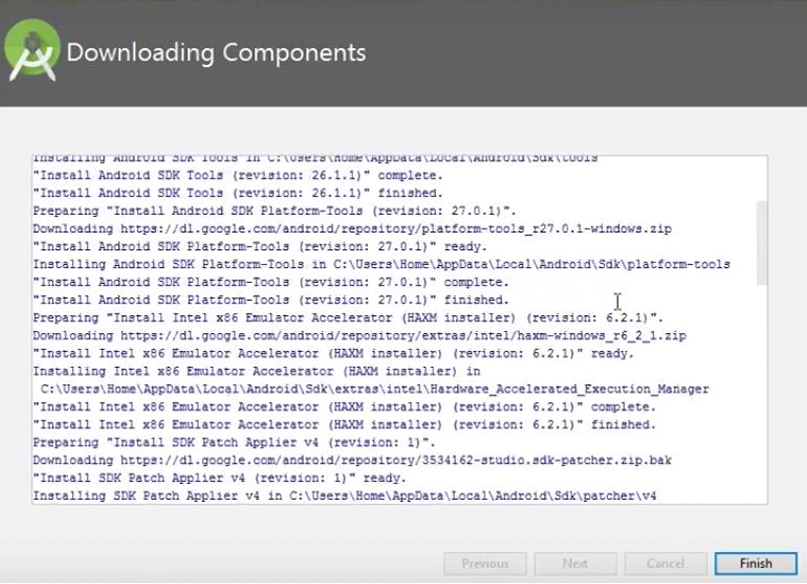
Choose Standard and click on Next. Now choose the theme, whether the Light theme or the Dark one. The light one is called the IntelliJ theme whereas the dark theme is called Dracula. Choose as required.



* Step 8: Now it is time to download the SDK components.

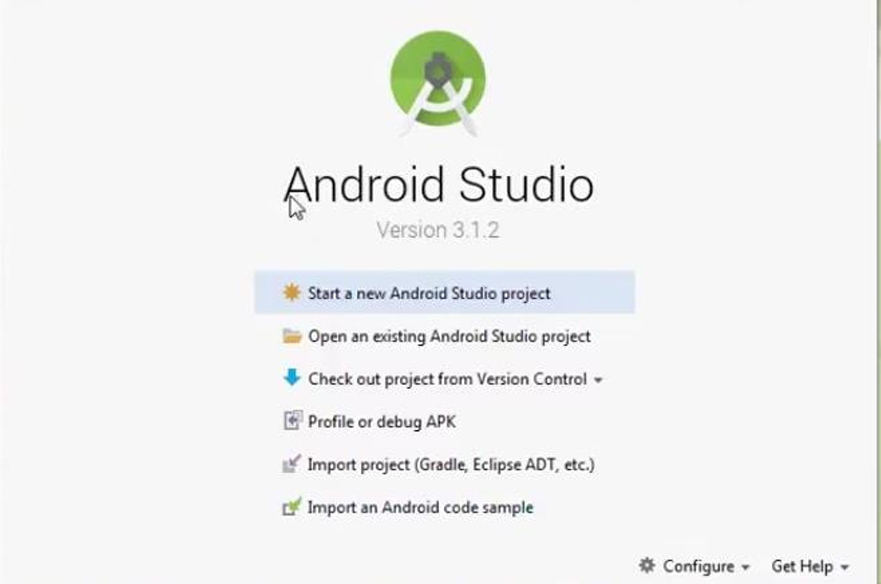


Click on Finish. Components begin to download let it complete.



The Android Studio has been successfully configured. Now it’s time to launch and build apps. Click on the Finish button to launch it.

* Step 9: Click on Start a new Android Studio project to build a new app.



**Practical 3: Android User Interface Design: To study various XML files needed for interface design.**

When designing the user interface (UI) for an Android application, several XML files are typically used. These files define the layout, styles, strings, dimensions, and other aspects of the UI. Here's an overview of the most common XML files:

1. Layout Files

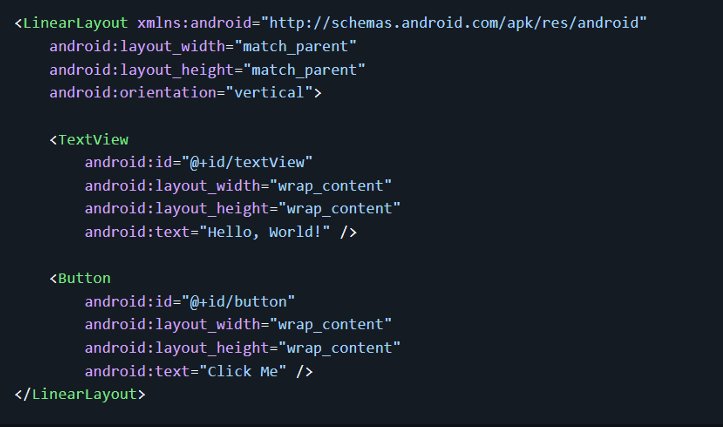
Layout files define the structure and appearance of the UI components in an activity or fragment.

* activity\_main.xml: Defines the main layout for an activity.
* fragment\_example.xml: Defines the layout for a fragment.

Common layout elements include:

* LinearLayout: Arranges its children in a single column or row.
* RelativeLayout: Positions its children relative to each other or to the parent.
* ConstraintLayout: Offers more flexibility and is recommended for complex layouts.
* FrameLayout: Designed to block out an area on the screen to display a single item.

Example of a simple layout file (activity\_main.xml):

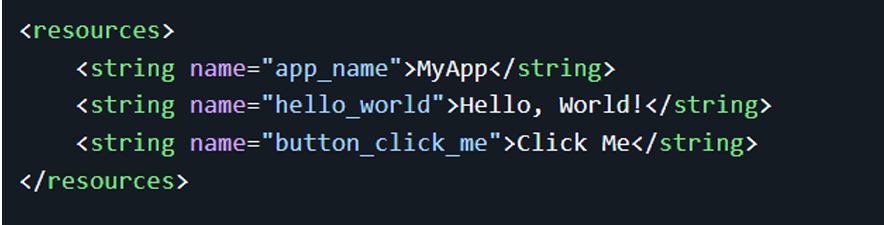


1. Resource Files

These files store various types of resources used in the app, such as strings, colors, dimensions, and styles.

* + strings.xml: Stores all the string resources used in the app.
  + colors.xml: Defines the color resources.
  + dimens.xml: Specifies dimension resources like padding and margins.
  + styles.xml: Contains style definitions to maintain a consistent look and feel.

Example of strings.xml:

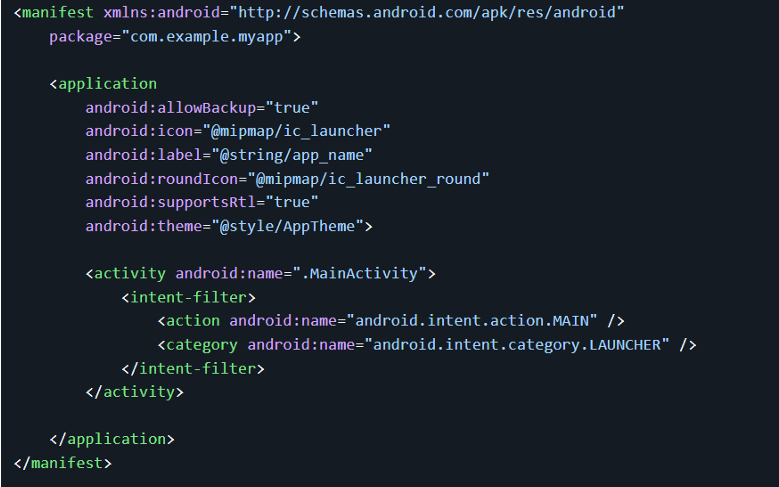


1. Manifest File

The AndroidManifest.xml file provides essential information about the app to the Android

system, including activities, permissions, services, and broadcast receivers.

Example of AndroidManifest.xml:

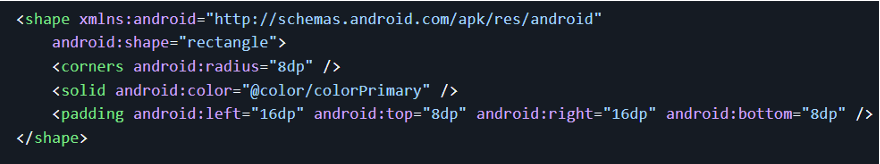


1. Drawable Resources

Drawable resources are graphics that can be drawn to the screen. These can be defined as

XML files (e.g., shapes, colors, state lists) or as image files (e.g., PNG, JPEG).

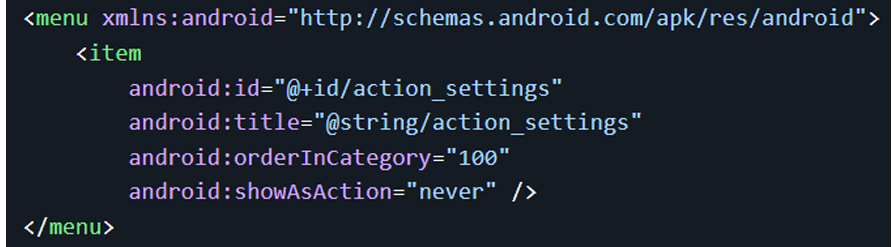
Example of a shape drawable (res/drawable/rounded\_button.xml):



1. Menu Resources

Menu resource files define the contents of app menus.

Example of a menu resource (res/menu/menu\_main.xml):



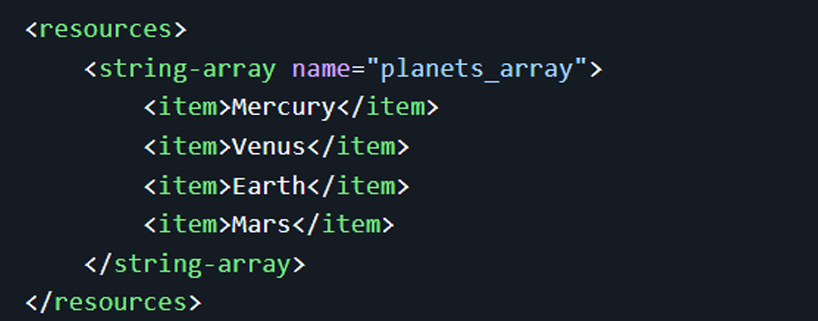
1. Values Resources

In addition to strings, colors, and dimensions, the values folder can contain other types of

resources:

* + bools.xml: Stores boolean resources.
  + integers.xml: Stores integer resources.
  + arrays.xml: Defines array resources.

Example of arrays.xml:



These XML files collectively define the visual and interactive aspects of an Android

application's user interface. Understanding and effectively utilizing these files is crucial for

creating well-structured and maintainable Android apps.

**Practical 4: Android User Interface Design: To implement different type of layouts like relative, grid, linear and table.**

1) Develop a program to implement constraint layout to display Hello World on screen:-

**Code:**

//Activity\_main.xml

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

<androidx.constraintlayout.widget.ConstraintLayout

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

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<!-- TextView for "Hello World" -->

<TextView

android:id="@+id/helloWorldText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World"

android:textSize="24sp"

android:textColor="#000000"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

// MainActivity.java

package com.example.practical1helloworld;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

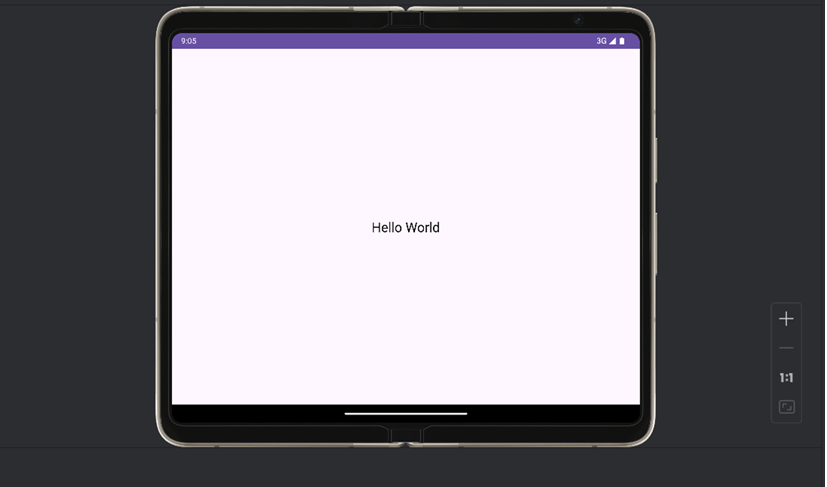
super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

**OUTPUT:**



2) Develop a program to implement linear layout to display send message and registration form (vertical):-

**Code:**

<?xml version=”1.0” encoding=”utf-8”?>

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp"

android:id="@+id/main">

<!-To Field ->

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="To"

android:height="50dp" />

<!-- Subject Field -->

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Subject"

android:layout\_marginTop="16dp"

android:height="50dp" />

<!-- Message Field -->

<EditText

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1"

android:hint="Message"

android:gravity="top"

android:inputType="textMultiLine"

android:height="50dp"

android:layout\_marginTop="16dp" />

<!-- Send Button -->

<Button

android:id="@+id/button\_send"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

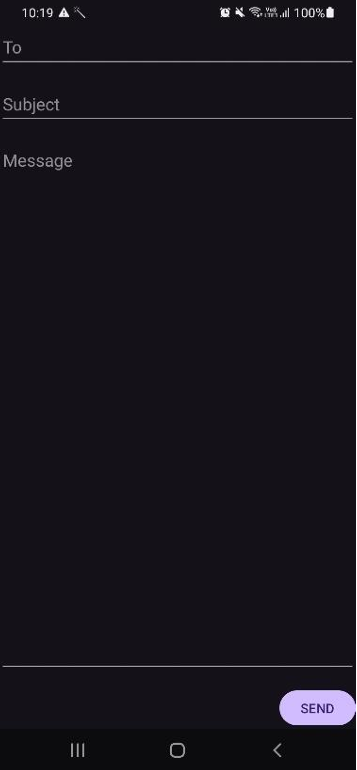
android:text="SEND"

android:layout\_gravity="end"

android:layout\_marginTop="16dp" />

</LinearLayout>

**OUTPUT:**



3) Develop a program to implement relative layout to display Login and sign up form:-

**Code:**

//Activity\_main.xml

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<!-- Login Form -->

<EditText

android:id="@+id/edtUsername"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Username"

android:layout\_marginTop="100dp"

android:padding="10dp"

android:inputType="textPersonName"/>

<EditText

android:id="@+id/edtPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:layout\_below="@id/edtUsername"

android:layout\_marginTop="20dp"

android:padding="10dp"

android:inputType="textPassword"/>

<Button

android:id="@+id/btnLogin"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_below="@id/edtPassword"

android:layout\_marginTop="20dp"

android:layout\_centerHorizontal="true"/>

<!-- Sign Up Form -->

<TextView

android:id="@+id/txtSignUp"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Don't have an account? Sign Up"

android:layout\_below="@id/btnLogin"

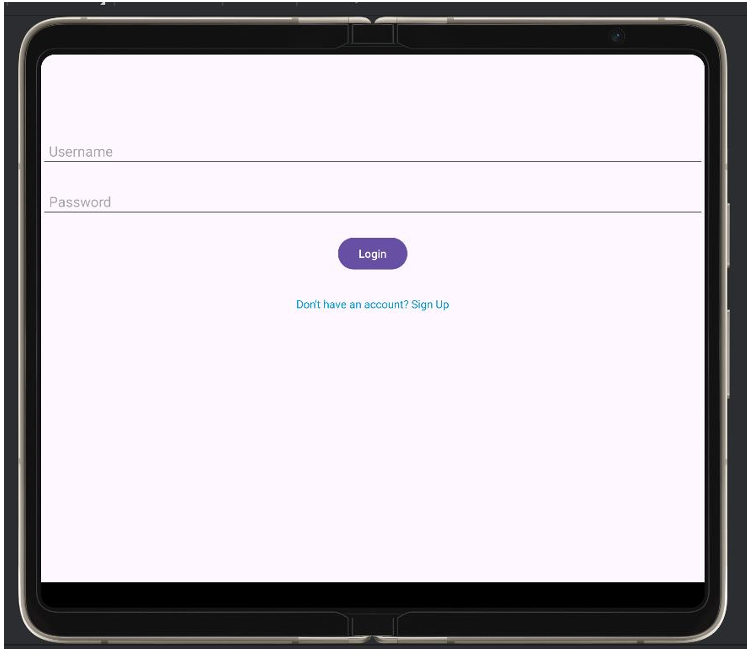
android:layout\_marginTop="30dp"

android:layout\_centerHorizontal="true"

android:textColor="@android:color/holo\_blue\_dark"/>

</RelativeLayout>

**Output:**

****

//Activity\_main.xml

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="16dp">

<!-- Username EditText -->

<EditText

android:id="@+id/username"

android:layout\_width="match\_parent"

android:layout\_height="48dp"

android:hint="Username"

android:inputType="textPersonName"

android:layout\_alignParentTop="true"

android:layout\_marginTop="40dp"/>

<!-- Email EditText -->

<EditText

android:id="@+id/email"

android:layout\_width="match\_parent"

android:layout\_height="48dp"

android:hint="@string/email"

android:inputType="textEmailAddress"

android:layout\_below="@id/username"

android:layout\_marginTop="20dp"/>

<!-- Password EditText -->

<EditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="48dp"

android:hint="Password"

android:inputType="textPassword"

android:layout\_below="@id/email"

android:layout\_marginTop="20dp"/>

<!-- Sign Up Button -->

<Button

android:id="@+id/signUpButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

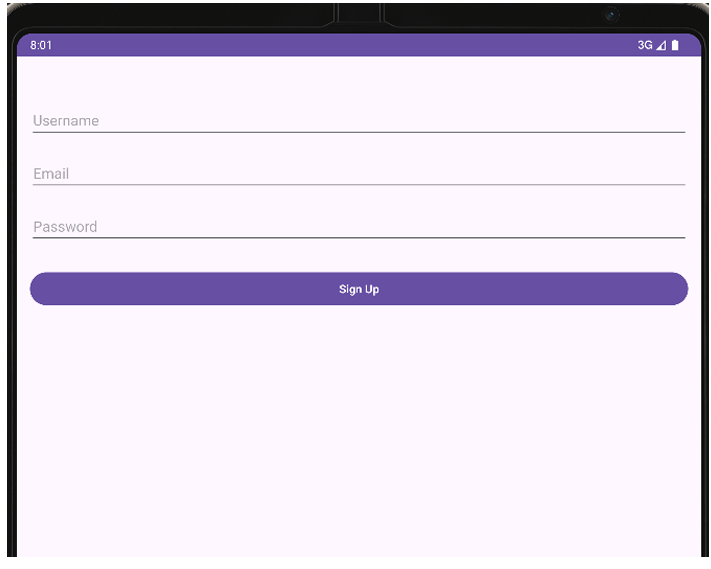
android:text="Sign Up"

android:layout\_below="@id/password"

android:layout\_marginTop="30dp"/>

</RelativeLayout>

**Output:**



4) Develop a program to implement table layout to display calculator:-

**Code:**

//Activity\_main.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp"

android:gravity="center">

<!-- Display Screen for the Calculator -->

<EditText

android:id="@+id/display"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:autofillHints=""

android:textSize="30sp"

android:inputType="none"

android:focusable="false"

android:gravity="end"

android:layout\_marginBottom="20dp"

tools:ignore="LabelFor" />

<!-- TableLayout for Calculator Buttons -->

<TableLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<!-- First Row -->

<TableRow style="?android:attr/buttonBarStyle">

<Button

android:id="@+id/button7"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_7" />

<Button

android:id="@+id/button8"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_8" />

<Button

android:id="@+id/button9"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_9" />

<Button

android:id="@+id/buttonDiv"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_h" />

</TableRow>

<!-- Second Row -->

<TableRow style="?android:attr/buttonBarStyle">

<Button

android:id="@+id/button4"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_4" />

<Button

android:id="@+id/button5"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_5" />

<Button

android:id="@+id/button6"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_6" />

<Button

android:id="@+id/buttonMul"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_k" />

</TableRow>

<!-- Third Row -->

<TableRow style="?android:attr/buttonBarStyle">

<Button

android:id="@+id/button1"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_1" />

<Button

android:id="@+id/button2"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_2" />

<Button

android:id="@+id/button3"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_3" />

<Button

android:id="@+id/buttonSub"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_l" />

</TableRow>

<!-- Fourth Row -->

<TableRow style="?android:attr/buttonBarStyle">

<Button

android:id="@+id/button0"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/\_0" />

<Button

android:id="@+id/buttonClear"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/c" />

<Button

android:id="@+id/buttonEqual"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/p" />

<Button

android:id="@+id/buttonAdd"

style="?android:attr/buttonBarButtonStyle"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="@string/a" />

</TableRow>

</TableLayout>

</LinearLayout>

**Output:**

****

**Practical 5: Apps Interactivity in Android: To incorporate element of interactivity using Android Fragment and Intent Class.**

// MainActivity.java

package com.cscorner.experiment5;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Load the fragment into the container

getSupportFragmentManager()

.beginTransaction()

.replace(R.id.fragment\_container, new MyFragment())

.commit();

}

}

//Activity\_main.xml

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

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/fragment\_container"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

//fragment\_my.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/fragment\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:orientation="vertical">

<Button

android:id="@+id/fragment\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Open SecondActivity" />

</LinearLayout>

//SecondActivity.java

package com.cscorner.experiment5;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

}

}

//activity\_second.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is SecondActivity"

android:textSize="18sp" />

</LinearLayout>

//MyFragment.java

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

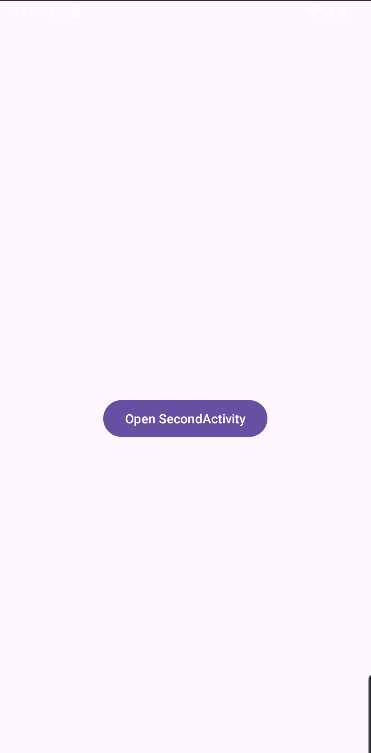
android:layout\_height="wrap\_content"

android:text="This is SecondActivity"

android:textSize="18sp" />

</LinearLayout>

**Output:**

**Practical 6: Persistent Data Storage: To perform database connectivity of android app using SQLite.**

**Practical 7: Android Services and Threads: To implement the concept of multithreading using Android Service class.**

Introduction: Multi-threading is defined as a feature through which we can run two or more

concurrent threads of a process. In this a process, the common data is shared among all these

threads also known as sub-processes exclusively. In android there are many ways through

which multi-threading can be established in the application.

Objective:

* Understanding the basic concept of multithreading.
* Understanding of Handler class in android
* Understanding of Runnable Interface.

Multi-Threading In Android: Multi-Threading in Android is a unique feature through which

more than one threads execute together without hindering the execution of other threads.

Multi-Threading in Android is not different from conventional multi-Threading. A class can

be thought of as a process having its method as it’s sub-processes or threads. All these

methods can run concurrently by using feature of Multi-Threading. In android, multi

Threading can be achieved through the use of many in-built classes. Out of them, Handler

class is most commonly used.

Handler Class In Android: Handler class come from the Package android.os.Handler package

and is most commonly used for multi-threading in android. Handler class provide sending

and receiving feature for messages between different threads and handle the thread execution

which is associated with that instance of Handler class. In android class, every thread is

associated with an instance of Handler class and it allows the thread to run along with other

threads and communicate with them through messages.

Runnable Interface:

Runnable interface is used in multi-threading to be called in a loop when the thread starts. It

is a type of thread that executes the statement in its body or calls other methods for a

specified or infinite number of times.This runable interface is used by the Handler class to

execute the multi-threading, i.e., to execute one or more thread in specified time.Runnable is

an interface which is implemented by the class desired to support multithreading and that

class must implements it’s abstract method public void run().Run() method is the core of

multithreading as it includes the statement or calls to other methods that the thread needs to

be made for multithreading.

**CODE:**

//mainactivity.java

package com.cscorner.experiment7;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

private TextView outputText;

private Button startThreadButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Initialize UI components

outputText = findViewById(R.id.outputText);

startThreadButton = findViewById(R.id.startThreadButton);

startThreadButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Start the thread when the button is clicked

startThread();

}

});

}

private void startThread() {

// Create and start a new thread

new Thread(new Runnable() {

@Override

public void run() {

// Simulating a task that runs in a loop

for (int i = 1; i <= 10; i++) {

final int count = i;

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

e.printStackTrace();

}

runOnUiThread(new Runnable() {

@Override

public void run() {

outputText.setText("Current count: " + count);

}

});

}

runOnUiThread(new Runnable() {

@Override

public void run() {

outputText.setText("Thread Finished!");

}

});

}

}).start(); // Start the thread

}

}

//activity\_main.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:gravity="center"

android:padding="16dp">

<TextView

android:id="@+id/outputText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Thread not started"

android:textSize="18sp"

android:gravity="center"

android:layout\_marginBottom="16dp" />

<Button

android:id="@+id/startThreadButton"

android:layout\_width="wrap\_content"

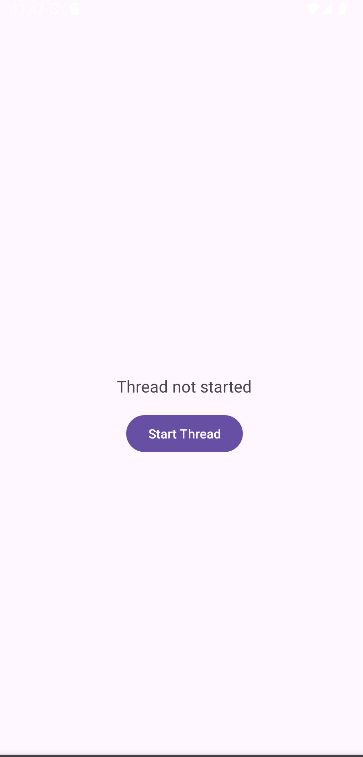
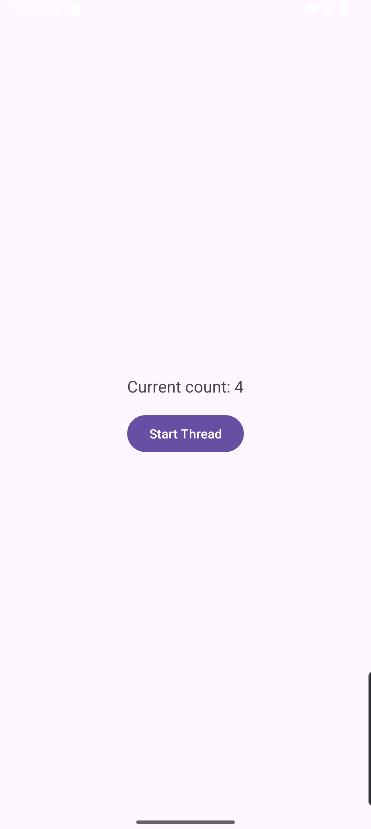
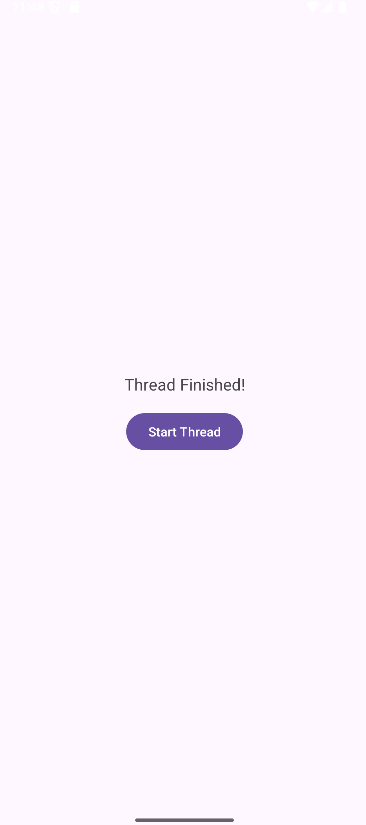
android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:text="Start Thread" />

</LinearLayout>

**OUTPUT:**

**Practical 8 To implement concept of permission and perform request for permission to access different hardware components of mobile.**

Permissions : The purpose of a permission is to protect the privacy of an Android user.

Android apps must request permission to access sensitive user data (such as contacts and

SMS), as well as certain system features (such as camera and internet). Depending on the

feature, the system might grant the permission automatically or might prompt the user to

approve the request.

Permission approval : An app must publicize the permissions it requires by including tags in

the app manifest. For example, an app that needs to access location would have this line in

the manifest:

//mainactivity.java

package com.cscorner.experiment8;

import android.Manifest;

import android.content.pm.PackageManager;

import android.os.Bundle;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

private static final int LOCATION\_PERMISSION\_REQUEST\_CODE = 1;

private TextView permissionStatusText;

private Button requestPermissionButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

permissionStatusText = findViewById(R.id.permissionStatusText);

requestPermissionButton = findViewById(R.id.requestPermissionButton);

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED &&

ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

permissionStatusText.setText("Location permission not granted.");

requestPermissionButton.setEnabled(true);

} else {

permissionStatusText.setText("Location permission granted!");

requestPermissionButton.setEnabled(false); // Disable button if permission is already granted

}

requestPermissionButton.setOnClickListener(v -> ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION, Manifest.permission.ACCESS\_COARSE\_LOCATION},

LOCATION\_PERMISSION\_REQUEST\_CODE));

}

@Override

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == LOCATION\_PERMISSION\_REQUEST\_CODE) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

// Permission granted, update the UI

permissionStatusText.setText("Location permission granted!");

Toast.makeText(this, "Location permission granted!", Toast.LENGTH\_SHORT).show();

} else {

// Permission denied, update the UI

permissionStatusText.setText("Location permission denied.");

Toast.makeText(this, "Location permission denied.", Toast.LENGTH\_SHORT).show();

}

}

}

}

//activity\_main.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp"

android:gravity="center">

<TextView

android:id="@+id/permissionStatusText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Location permission status will appear here."

android:textSize="18sp"

android:gravity="center"

android:layout\_marginBottom="16dp"/>

<Button

android:id="@+id/requestPermissionButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Request Location Permission"

android:layout\_gravity="center"

android:enabled="true"/>

</LinearLayout>

//AndroidManifest.xml

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

**OUTPUT:**

