

MOBILE APPLICATION DEVELOPMENT LABORATORY

PRACTICAL REPORT FILE

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
Information Technology



Submitted By:

Rohit Kumar — 1905389

Information Technology Department
Guru Nanak Dev Engineering College
Ludhiana - 141006

Contents

1	To study design aspects of development environment like Android, IOS.	3
2	Android Development Environment: To setup Android studio2 and study its basic components.	6
3	Android User Interface Design: To study various XML files needed for interface design.	12
4	Android User Interface Design: To implement different type of layouts like relative, grid, linear and table.	18
5	Apps Interactivity in Android: To incorporate element of interactivity using Android Fragment and Intent Class	22
6	Persistent Data Storage: To perform database connectivity of android app using SQLite.	27
7	Android Services and Threads:To implement the concept of multithreading using Android Service class	36
8	Android Security and Debugging: To implement concept of permission and perform request for permission to access different hardware components of mobile	40
9	Android Security and Debugging: To perform debugging and testing of android app using tools like Logcat, Android debug bridge, DDMS	44

1 To study design aspects of development environment like Android, IOS.

1.0.1 Android

Android's Java environment can be broken down into a handful of key sections. When you understand the contents in each of these sections, the Javadoc reference material that ships with the SDK becomes a real tool and not just a pile of seemingly unrelated material. You might recall that Android isn't a strictly Java ME software environment, but there's some commonality between the Android platforms and other Java development platforms. The next few sections review some of the Java packages (core and optional) in the Android SDK and where you can use them. Android Studio is the official IDE for Android development, and with a single download it includes everything you need to begin developing Android apps as you can see below

- IntelliJ IDE + Android Studio plugin
- Android SDK Tools
- Android Platform-tools
- Android Emulator with an Android system image including Google Play Services

Android Studio provides tools for building apps on every type of Android device. Code editing, debugging, performance tooling, a flexible build system, and an instant build or deploy system are included in Android studio.

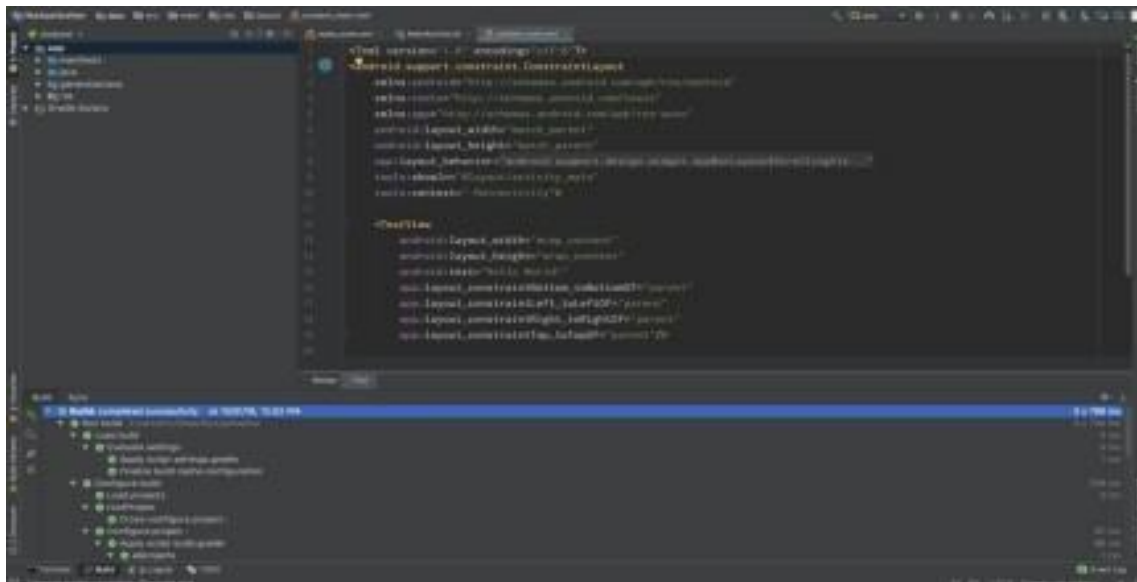


Figure 1.1: Android Studio

1. Command Line Tools:- The Android SDK tools available from the SDK Manager provide additional command-line tools to help you during your Android development. The tools are classified into two groups: SDK tools and platform tools. SDK tools are platform independent and are required no matter which Android platform you are developing on. Platform tools are customized to support the features of the latest Android platform.

2. **SDK Tools:-** The SDK tools are installed with the SDK starter package and are periodically updated. The SDK tools are required if you are developing Android applications. The most important SDK tools include the Android SDK Manager (Android sdk), the AVD Manager (Android AVD) the emulator (emulator), and the Dalvik Debug Monitor Server (DDMS).
3. **Virtual Device Tools:-**
 - Android Virtual Device Manager
 - Android Emulator (emulator)
 - mksdcard
4. **Development Tools:-** Hierarchy Viewer (hierarchyviewer) - Provides a visual representation of the layout's View hierarchy with performance information for each node in the layout, and a magnified view of the display to closely examine the pixels in your layout.
5. **SDK Manager:-** SDK Manager lets you manage SDK packages, such as installed platforms and system images. sqlite3 - Lets you access the SQLite data files created and used by Android applications.
6. **Debugging tools:-**
 - Android Monitor
 - adb
 - Dalvik Debug Monitor Server (DDMS)
 - Device Monitor
 - systrace
7. **Build Tools:-**
 - apksigner
 - JOBB
 - ProGuard
 - zipalign

1.0.2 iOS

iOS application development is the process of making mobile applications for Apple hardware, including iPhone, iPad and iPod Touch. The software is written in the Swift programming language or Objective-C and then deployed to the App Store for users to download.

Requirements for iOS development environment:-

- An Apple Mac computer running the latest version of macOS.
- Xcode, which is the integrated development environment (IDE) for macOS, available as a free download from the Mac App Store.
- An active Apple Developer account, which requires a USD 99 annual fee.

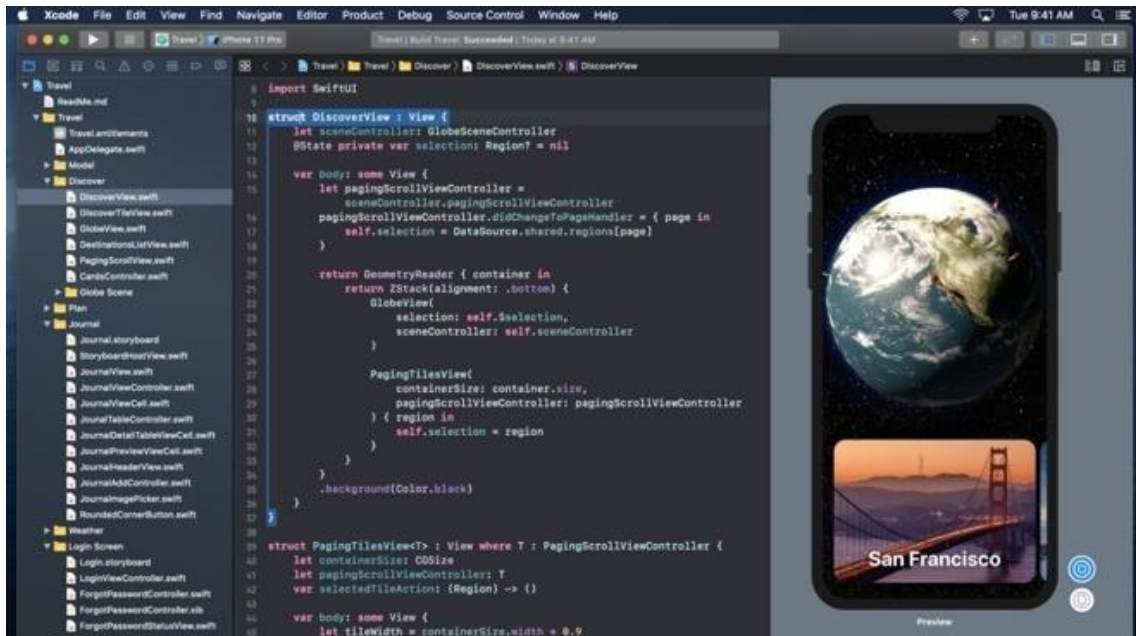


Figure 1.2: xCode iOS IDE

iOS development programming languages:-

- **Objective-C:** Developed in the early 1980s, Objective-C was the primary programming language for all Apple products for decades. Derived from the C language, Objective-C is an object-oriented programming language centered on passing messages to different processes (as opposed to invoking a process in traditional C programming). Many developers choose to maintain their legacy applications written in Objective-C instead of integrating them into the Swift framework, which was introduced in 2014.
- **Swift:** The Swift programming language is the new “official” language of iOS. While it has many similarities to Objective-C, Swift is designed to use a simpler syntax and is more focused on security than its predecessor. Because it shares a run time with Objective-C, you can easily incorporate legacy code into updated apps. Swift is easy to learn, even for people just beginning to program. Because Swift is faster, more secure and easier to use than Objective-C, you should plan to use it to develop your iOS app unless you have a compelling reason to stick with Objective-C.

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

1. Head over to <https://developer.android.com/studidownloads> to get the Android Studio executable or zip file.
2. Click on the Download Android Studio Button.

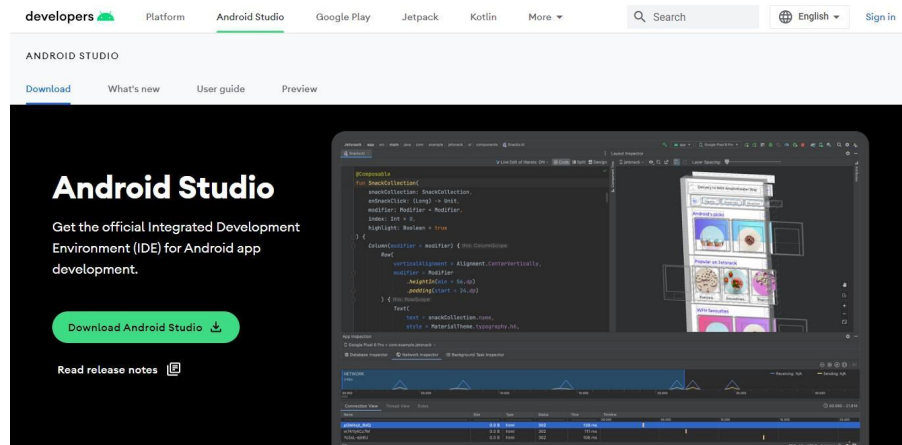


Figure 2.1: Official Android Studio download page

Click on the “I have read and agree with the above terms and conditions” checkbox followed by the download button.

Figure 2.2: Terms and conditions

Click on the Save file button in the appeared prompt box and the file will start down- loading.

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



Figure 2.3: Download completed

Click on next. In the next prompt, it'll ask for a path for installation. Choose a path and hit next.

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

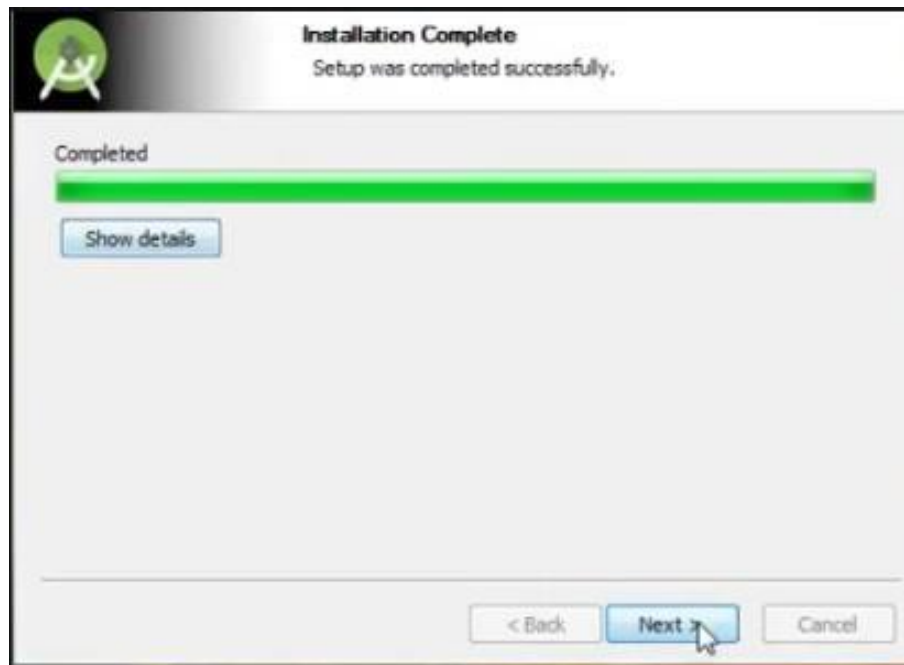


Figure 2.4: Installation process

Click on next.



Figure 2.5: Installation process

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’.

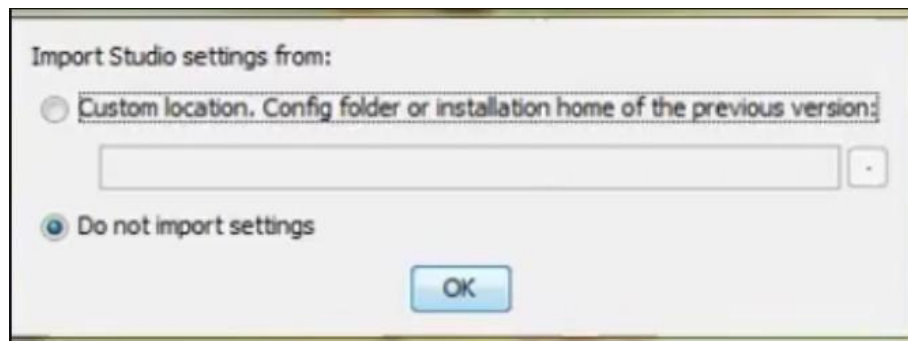


Figure 2.6: Installation process

Click on OK.

6. This will start the Android Studio.

Figure 2.7: Installation completed

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

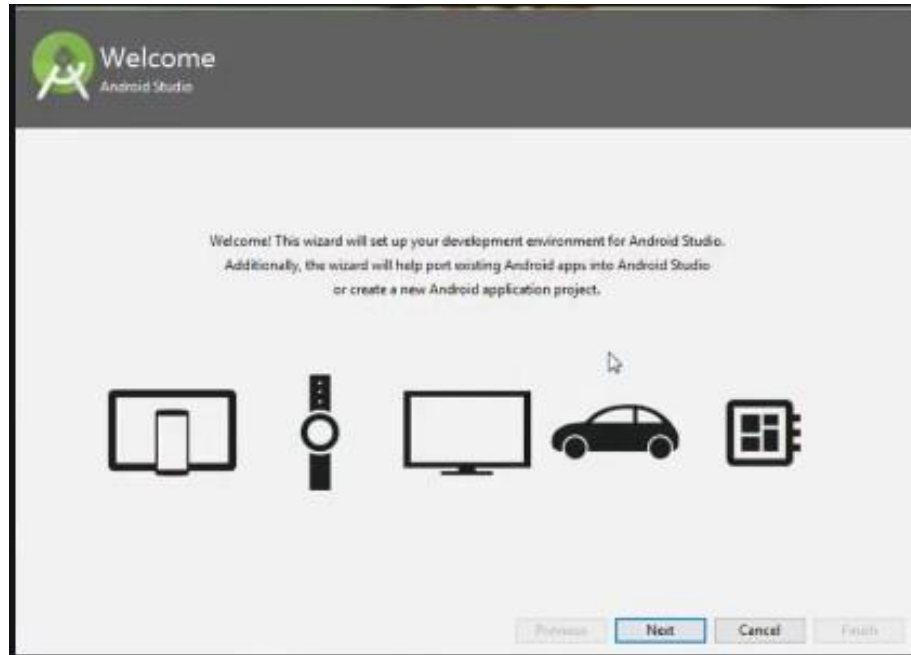


Figure 2.8: Welcome screen

8. 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. Click on Next.

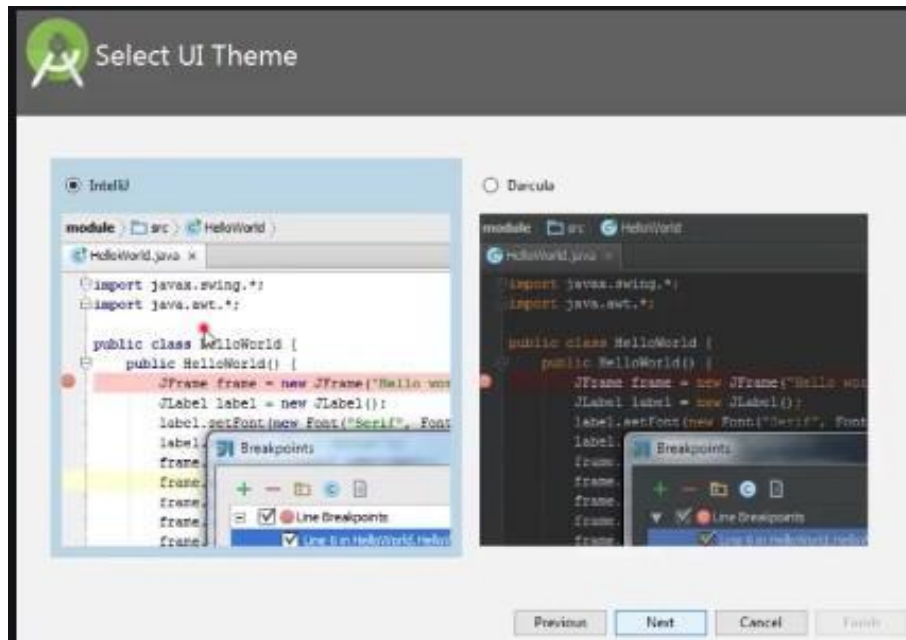


Figure 2.9: Theme selection

9. Now it is time to download the SDK components.

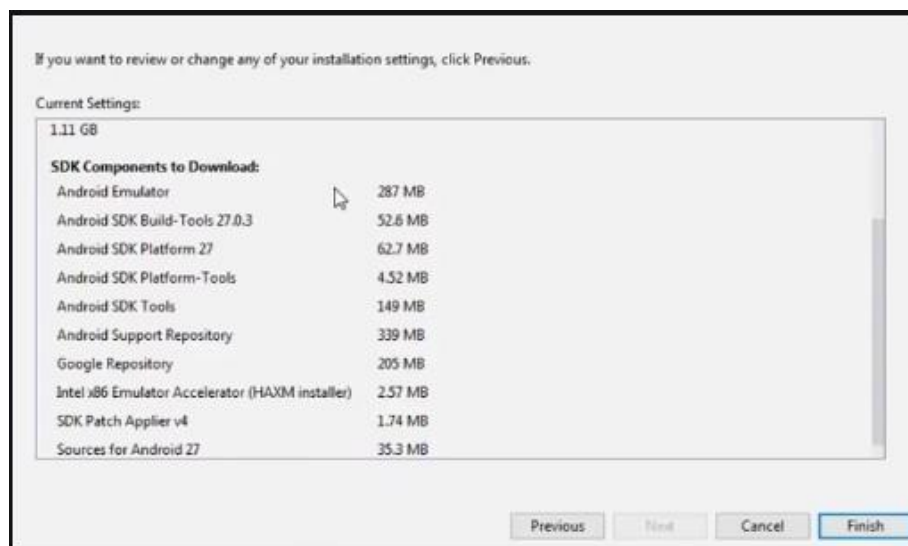


Figure 2.10: Installation process

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

10. Click on Start a new Android Studio project to build a new app.

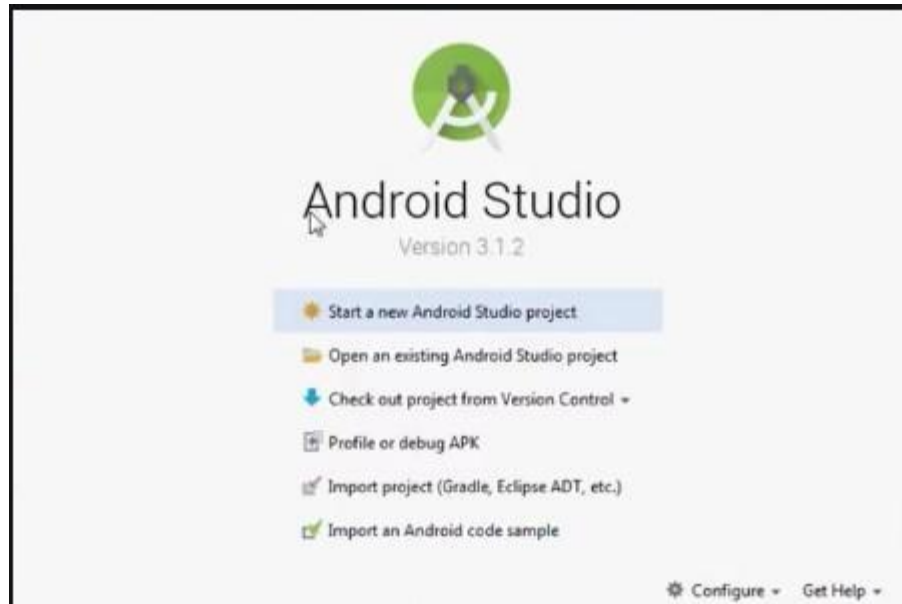


Figure 2.11: First Project

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

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). In Android, the XML is used to implement UI-related data, and it's a lightweight markup language that doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked. Syntax for XML tag:-

```
<tag_name>Hello World!</tag_name>
```

Different XML files serve different purposes in Android Studio. The list of various XML files in Android Studio with their purposes is discussed below.

1. **Layout XML files in android:** The Layout XML files are responsible for the actual User Interface of the application. It holds all the widgets or views like Buttons, TextViews, EditTexts, etc. which are defined under the ViewGroups. The Location of the layout files in Android is:

```
app -> src -> main -> res -> layout
```

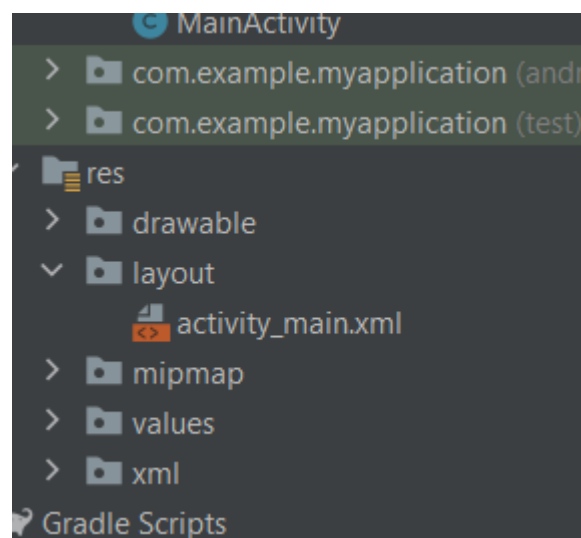


Figure 3.1: Layout XML location

2. **AndroidManifest.xml file:** This file describes the essential information about the application's, like the application's package names which matches code's namespaces, a component of the application like activities, services, broadcast receivers, and content providers. Permission required by the user for the application features also mentioned in this XML file. Location of the AndroidManifest.xml file:

app -> src -> main

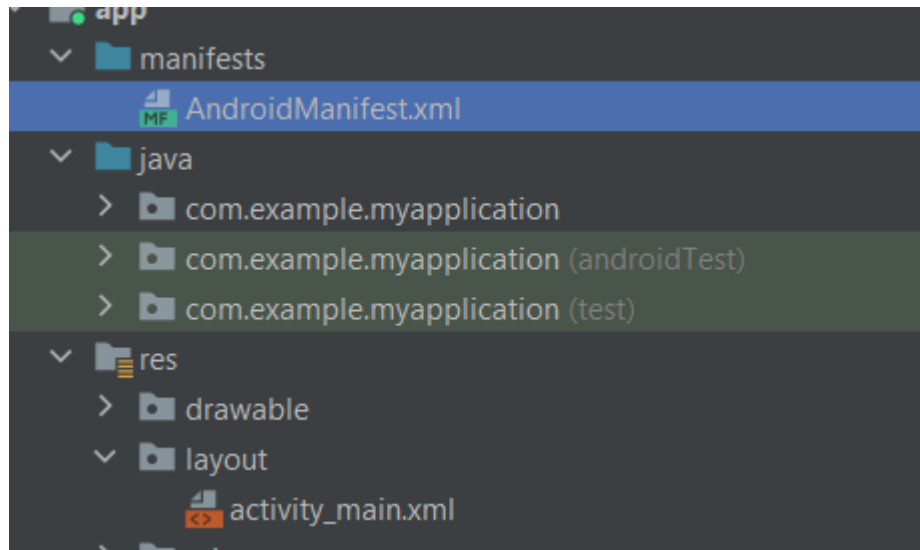


Figure 3.2: Manifest XML location

3. **strings.xml file:** This file contains texts for all the TextViews widgets. This enables reusability of code, and also helps in the localization of the application with different languages. The strings defined in these files can be used to replace all hardcoded text in the entire application. Location of the strings.xml file

app -> src -> main -> res -> values

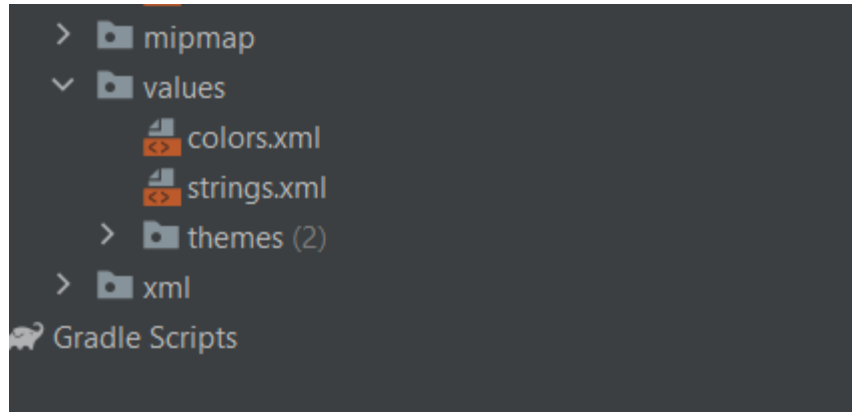


Figure 3.3: String XML location

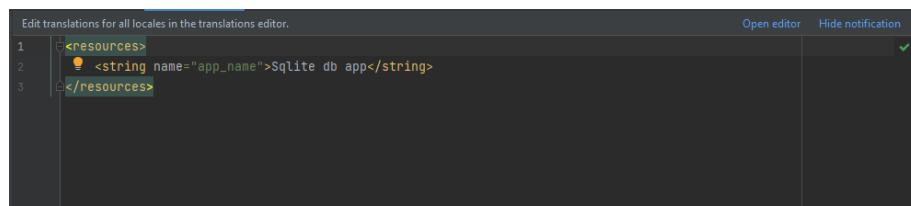


Figure 3.4: String XML Code

4. **themes.xml file:** This file defines the base theme and customized themes of the application. It also used to define styles and looks for the UI(User Interface) of the application. By defining styles we can customize how the views or widgets look on the User Interface. Location of styles.xml file.

app -> src -> main -> res -> values

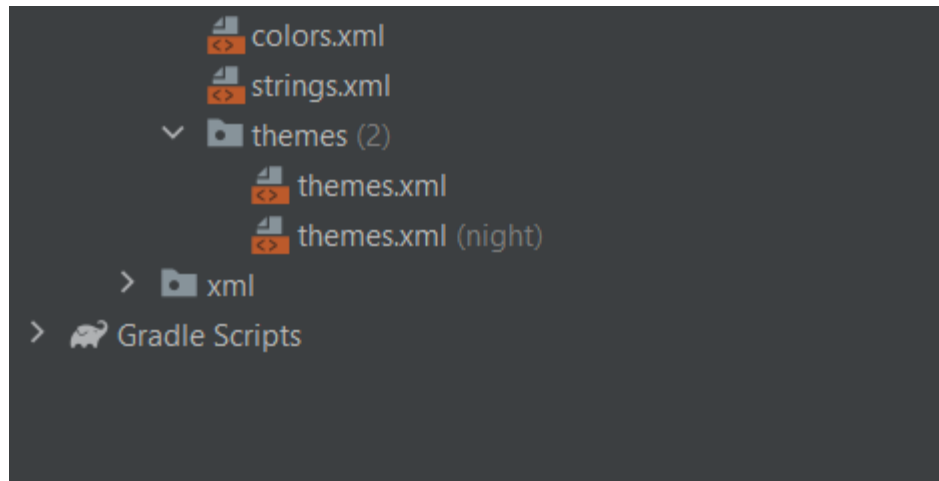


Figure 3.5: Themes XML location

```

1 <resources xmlns:tools="http://schemas.android.com/tools">
2   <!-- Base application theme. -->
3   <style name="Theme.SQLiteDbApp" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
4     <!-- Primary brand color. -->
5     <item name="colorPrimary">@color/purple_500</item>
6     <item name="colorPrimaryVariant">@color/purple_700</item>
7     <item name="colorOnPrimary">@color/white</item>
8     <!-- Secondary brand color. -->
9     <item name="colorSecondary">@color/teal_200</item>
10    <item name="colorSecondaryVariant">@color/teal_700</item>
11    <item name="colorOnSecondary">@color/black</item>
12    <!-- Status bar color. -->
13    <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
14    <!-- Customize your theme here. -->
15  </style>
16 </resources>

```

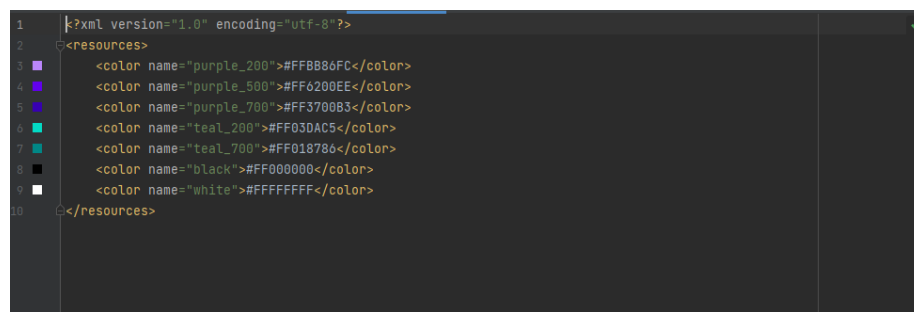
Figure 3.6: Themes XML code

5. **Drawable XML files:** These are the XML files that provide graphics to elements like custom background for the buttons and its ripple effects, also various gradients can be created. This also holds the vector graphics like icons. Using these files custom layouts can be constructed for EditTexts. Location for the Drawable files are:

app -> src -> main -> res -> drawable

Figure 3.7: Drawable XML location

6. **colors.xml file:** The colors.xml file is responsible to hold all the types of colors required for the application. It may be primary brand color and its variants and secondary brand color and its variants. The colors help uphold the brand of the applications. So the colors need to be decided cautiously as they are responsible for the User Experience. The colors need to be defined in the hex code format. Location of colors.xml file:

A screenshot of an IDE showing the content of a colors.xml file. The file is located at app/src/main/res/values/colors.xml. The code is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="purple_200">#FFBB86FC</color>
4     <color name="purple_500">#FF6200EE</color>
5     <color name="purple_700">#FF3700B3</color>
6     <color name="teal_200">#FF03DAC5</color>
7     <color name="teal_700">#FF018786</color>
8     <color name="black">#FF000000</color>
9     <color name="white">#FFFFFFFF</color>
10 </resources>
```

Figure 3.8: colors XML code

7. **dimens.xml file:** As the file name itself suggests that the file is responsible to hold the entire dimensions for the views. it may be the height of the Button, padding of the views, the margin for the views, etc. The dimensions need to in the format of pixel density(dp) values. Which replaces all the hard-coded dp values for the views. This file needs to be created separately in the values folder. Location to create dimens.xml file:

app -> src -> main -> res -> values

4 Apps Interactivity in Android: To incorporate element of inter-activity using Android Fragment and Intent Class

4.0.1 UI development (XML files)

1. activity_main.xml:

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

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

        <Button
            android:id="@+id/btn1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:text="First" />

        <Button
            android:id="@+id/btn2"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:text="Second" />

        <Button
            android:id="@+id/btn3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Third" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="682dp"
        android:orientation="vertical">

        <androidx.fragment.app.FragmentContainerView
            android:id="@+id/fragmentContainer"
            android:name="com.example.fragments.FirstFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>

</LinearLayout>
```

2. fragment_first.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"
android:background="#ff4564"
tools:context=".FirstFragment">

<!-- TODO: Update blank fragment layout -->
<TextView
    android:id="@+id/t1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:gravity="center"
    android:text="fragment1"
    android:textColor="#ffffff"
    android:textSize="30sp"
    android:textStyle="bold" />

</FrameLayout>

```

3. fragment_second.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"
    android:background="#bbb321"
    tools:context=".SecondFragment">

<!-- TODO: Update blank fragment layout -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="fragment2"
    android:layout_gravity="center"
    android:gravity="center"
    android:textSize="30sp"
    android:textStyle="bold"
    android:id="@+id/t2" />

</FrameLayout>

```

4. fragment_third.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"
    android:background="#15EC43"
    tools:context=".ThirdFragment">

<!-- TODO: Update blank fragment layout -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="fragment 3"
    android:layout_gravity="center"
    android:gravity="center"
    android:textSize="30sp"
    android:textStyle="bold"
    android:id="@+id/t3" />

```

```
</FrameLayout>
```

4.0.2 Java Code

1. MainActivity.java

```
package com.example.fragments;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b1=findViewById(R.id.btn1);
        Button b2=findViewById(R.id.btn2);
        Button b3=findViewById(R.id.btn3);

        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FragmentManager fm= getSupportFragmentManager();
                FragmentTransaction ft= fm.beginTransaction();
                ft.replace( R.id.fragmentContainer, FirstFragment.class, null);
                ft.commit();
            }
        });

        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FragmentManager fm= getSupportFragmentManager();
                FragmentTransaction ft= fm.beginTransaction();

                ft.replace( R.id.fragmentContainer, new SecondFragment());
                ft.commit();
            }
        });

        btn3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FragmentManager fm= getSupportFragmentManager();
                FragmentTransaction ft= fm.beginTransaction();

                ft.replace( R.id.fragmentContainer, new ThirdFragment());
                ft.commit();
            }
        });
    }
}
```

2. FirstFragment.java:

```
package com.example.fragments;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class FirstFragment extends Fragment {

    // Important method
    public FirstFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view= inflater.inflate(R.layout.fragment_first, container, false);
        TextView t1= view.findViewById(R.id.t1);
        return view;
    }
}
```

4.0.3 Output Screens:-



Figure 4.1: Fragment 1



Figure 4.2: Fragment 2

Figure 4.3: Fragment 3

5 Persistent Data Storage: To perform database connectivity of android app using SQLite.

5.0.1 UI Development (XML code)

1. DatabaseHelper class:

```
public class DataBaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "Student.db";
    public static final String TABLE_NAME = "Student_table";

    public static final String COL_1 = "ID";
    public static final String COL_2 = "NAME";
    public static final String COL_3 = "SURNAME";
    public static final String COL_4 = "MARKS";

    public DataBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
        SQLiteDatabase db = this.getWritableDatabase();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT, SURNAME TEXT,MARKS INTEGER)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
    }
}
```

2. Login Facebook: activity_login.xml

```
<!--<?xml version="1.0" encoding="utf-8"?>-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingLeft="20dp"
    android:paddingTop="5dp"
    android:paddingEnd="5dp"
    android:paddingRight="20dp"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/fbimg"
        android:layout_width="match_parent"
        android:layout_height="50pt"
        android:layout_marginTop="13dp"
        app:srcCompat="@drawable/fb_img" />

    <TextView
        android:id="@+id/ask"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Where do you want to go?"
        android:textSize="20sp">
```

```

        android:textStyle="bold" />

<Button
    android:id="@+id/fbapp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="50dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="50dp"
    android:backgroundTint="#4B5CBA"
    android:text="Facebook App"
    app:cornerRadius="23pt" />

<TextView
    android:id="@+id/forgotBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Or"
    android:textSize="20sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/fbweb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="50dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="50dp"
    android:backgroundTint="#AEF44336"
    android:text="Facebook on web"
    app:cornerRadius="23pt" />

</LinearLayout>

```

5.0.2 Java Code

1. MainActivity.java

```

package com.example.ui_practice;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_intro);
        ImageView next;
        DataBaseHelper myDb = new DataBaseHelper(this);
        next=findViewById(R.id.next);

        next.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent iNext= new Intent(MainActivity.this, Login.class);
                startActivity(iNext);
                finish();
            }
        })
    }
}

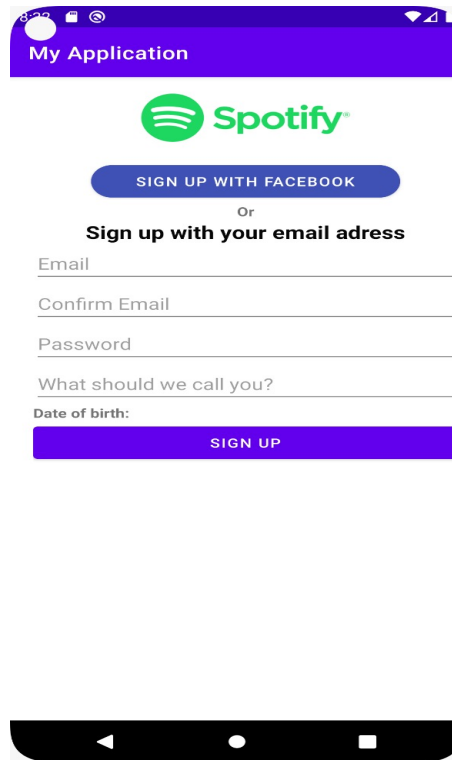
```



```
}  
    }  
};
```

5.0.3 Output Screens:

1. Register Form:-



A screenshot of a mobile application's registration form. The app's title bar is blue and says "My Application". Below it is the Spotify logo. A blue button labeled "SIGN UP WITH FACEBOOK" is followed by the word "Or". Below that is the heading "Sign up with your email address". The form contains five input fields: "Email", "Confirm Email", "Password", "What should we call you?", and "Date of birth:". At the bottom is a large blue button labeled "SIGN UP". The bottom of the screen shows a black Android navigation bar.

Figure 5.1: Registration form

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

7.0.1 XML Code

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

    <TextView
        android:id="@+id/tv1"
        android:layout_width="75sp"
        android:layout_height="75sp"
        android:background="@color/colorPrimary"
        android:textColor="@color/colorAccent"
        android:gravity="center"
        android:textSize="10sp"
        android:layout_centerVertical="true"
        android:layout_toLeftOf="@id/tv2"
        />

    <TextView
        android:id="@+id/tv2"
        android:layout_width="75sp"
        android:layout_height="75sp"
        android:background="@color/colorPrimary"
        android:textColor="@color/colorAccent"
        android:gravity="center"
        android:textSize="10sp"
        android:layout_centerInParent="true"
        />

    <TextView
        android:id="@+id/tv3"
        android:layout_width="75sp"
        android:layout_height="75sp"
        android:background="@color/colorPrimary"
        android:textColor="@color/colorAccent"
        android:gravity="center"
        android:textSize="10sp"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@id/tv2"
        />

    <Button
        android:id="@+id/btnStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/tv2"
        />

</RelativeLayout>
```

7.0.2 Java Code

```

class MainActivity : AppCompatActivity() {
    @SuppressWarnings("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Assigning Layout elements
        val tv1
        = findViewById<TextView>(R.id.tv1)
        val tv2
        = findViewById<TextView>(R.id.tv2)
        val tv3
        = findViewById<TextView>(R.id.tv3)
        val btn
        = findViewById<Button>(R.id.btnStart)

        // Boolean for Button (initially False)
        var boolbtn
        = false
        // Button onClick action
        btn.setOnClickListener
        {
            // Button (True)
            boolbtn = !boolbtn
            // Case where Button is False
            if (!boolbtn)
            {
                tv1.text = "Stopped1"
                tv2.text = "Stopped2"
                tv3.text = "Stopped3"
                btn.text = "Start"
            }
            // Case where Threads are running
            else
            {
                // Setting the button text as "Stop"
                btn.text = "Stop"

                // Thread 1
                Thread(Runnable {

                    // Runs only when Button is True
                    while (boolbtn) {

                        runOnUiThread
                        {
                            tv1.text = "Started1"
                        }
                        Thread.sleep(1000)
                        runOnUiThread
                        {
                            tv1.text = "Activity1"
                        }
                        Thread.sleep(1000)
                    }
                }).start()

                // Thread 2
                Thread(Runnable {

                    // Runs only when Button is
                    // True
                    while (boolbtn) {
                        runOnUiThread
                        {
                            tv2.text = "Started2"

```


7.0.3 Output Screens:

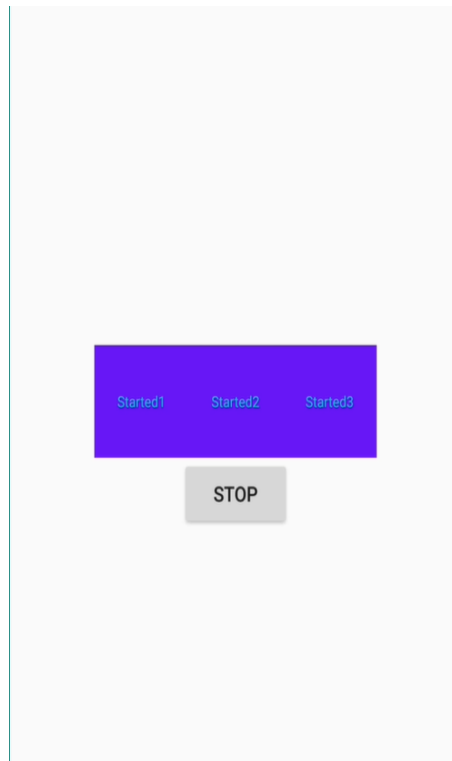


Figure 7.1: Activity state1

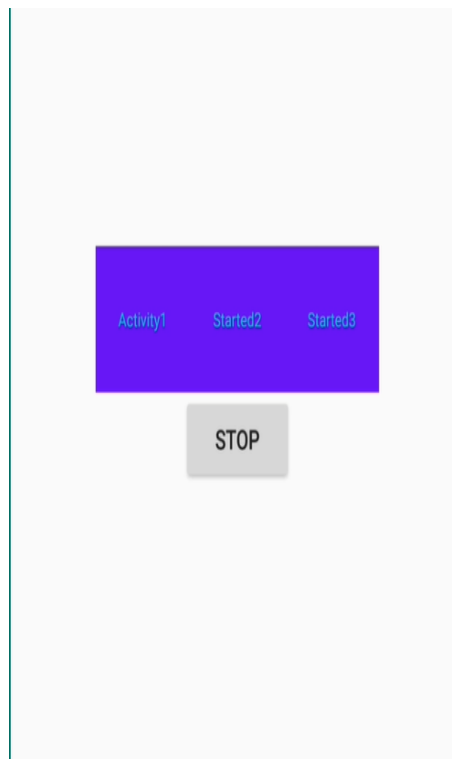


Figure 7.2: Activity state 2

8 Android Security and Debugging: To implement concept of permission and perform request for permission to access different hardware components of mobile

1. Declare the permission in the Android Manifest file: In Android, permissions are declared in the AndroidManifest.xml file using the uses-permission tag.

```
<!--Declaring the required permissions-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
```

2. Modify activity_main.xml file to Add two buttons to request permission on button click: Permission will be checked and requested on button click. Open the activity_main.xml file and add two buttons to it.

```
<!--Button to request storage permission-->
<Button
    android:id="@+id/storage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Storage"
    android:layout_marginTop="16dp"
    android:padding="8dp"
    android:layout_below="@id/toolbar"
    android:layout_centerHorizontal="true"/>

<!--Button to request camera permission-->
<Button
    android:id="@+id/camera"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Camera"
    android:layout_marginTop="16dp"
    android:padding="8dp"
    android:layout_below="@id/storage"
    android:layout_centerHorizontal="true"/>
```

3. Check whether permission is already granted or not. If permission isn't already granted, request the user for the permission: In order to use any service or feature, the permissions are required. Hence we have to ensure that the permissions are given for that. If not, then the permissions are requested.

Check for permissions: Beginning with Android 6.0 (API level 23), the user has the right to revoke permissions from any app at any time, even if the app targets a lower API level. So to use the service, the app needs to check for permissions every time.

Request Permissions: When `PERMISSION_DENIED` is returned from the `checkSelfPermission()` method.

```
// Function to check and request permission
public void checkPermission(String permission, int requestCode)
{
    // Checking if permission is not granted
    if (ContextCompat.checkSelfPermission(MainActivity.this, permission) == PackageManager.
        PERMISSION_DENIED) {
        ActivityCompat.requestPermissions(MainActivity.this, new String[] { permission }, requestCode);
    }
    else {
```

```

        Toast.makeText(MainActivity.this, "Permission already granted", Toast.LENGTH_SHORT).show();
    }
}

```

4. Override onRequestPermissionsResult() method: onRequestPermissionsResult() is called when user grant or decline the permission. RequestCode is one of the parameters of this function which is used to check user action for the corresponding requests. Here a toast message is shown indicating the permission and user action.

```

@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == CAMERA_PERMISSION_CODE) {

        // Checking whether user granted the permission or not.
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

            // Showing the toast message
            Toast.makeText(MainActivity.this, "Camera Permission Granted", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(MainActivity.this, "Camera Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
    else if (requestCode == STORAGE_PERMISSION_CODE) {
        if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(MainActivity.this, "Storage Permission Granted", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(MainActivity.this, "Storage Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}
}

```

8.0.1 Output Screens

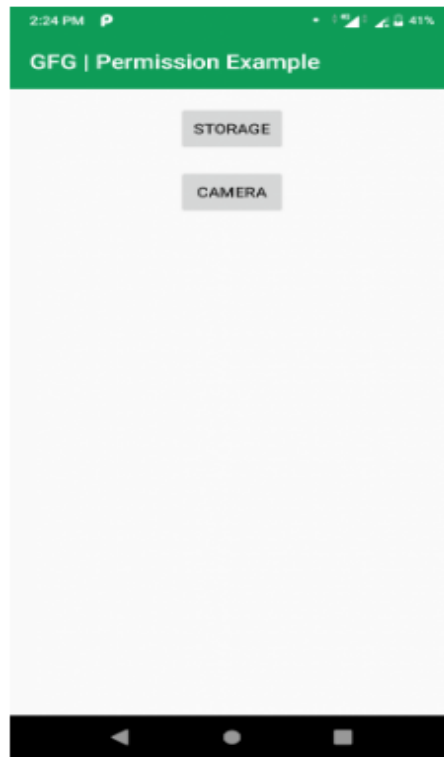


Figure 8.1: Main Activity

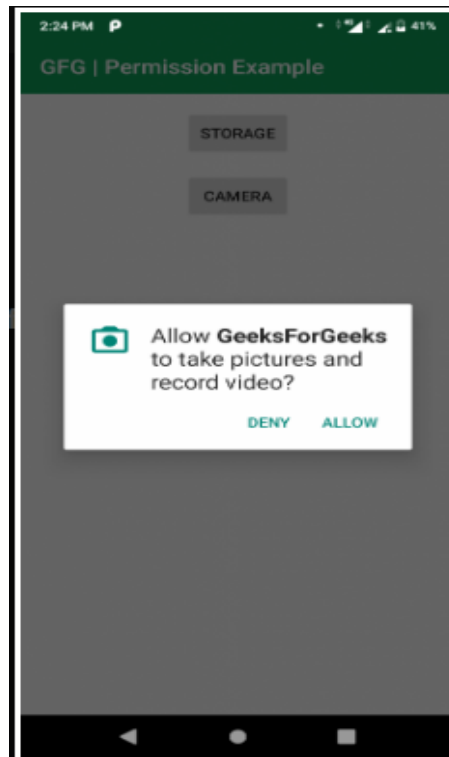


Figure 8.2: Permission dialog box



Figure 8.3: Toast message after permission

9 Android Security and Debugging: To perform debugging and testing of android app using tools like Logcat, Android debug bridge, DDMS

Logcat is an important tool in Android studio. Logcat helps developers to debug the code when an application is not running or has crashed. Steps to run Logact in Android Studio:-

1. Create new project in android studio and select empty activity.

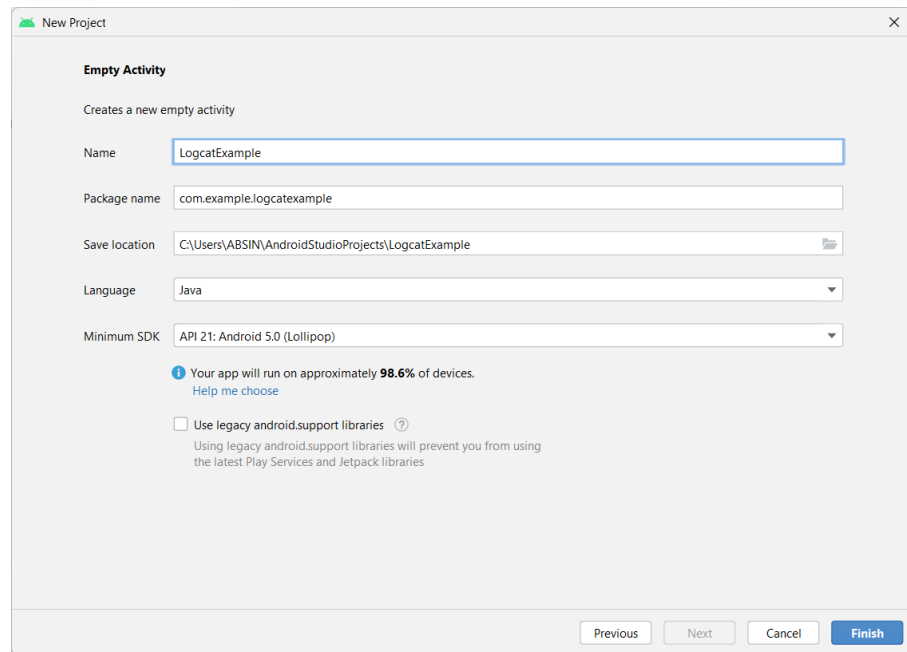


Figure 9.1: Create New Project

2. Now go to activity_main.xml. By default we get a textview so just set the id of this textview.

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Now go to MainActivity.java. Create a Textview object and same time initialize with the help of findViewById() method.

```
package com.example.logcatexample;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    TextView textView=findViewById(R.id.textview);
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

4. Now connect any physical or virtual device to android studio and click on Run button. The application is installed however immediately crashed.
5. Open Logcat and again click on run button. In logcat some line of code is generated so we have to find out the line that tells us which line of code contains error. Sometimes we need to scroll up in locat window to find out the error. Examine the below picture carefully and an orange color arrow indicate an error line.

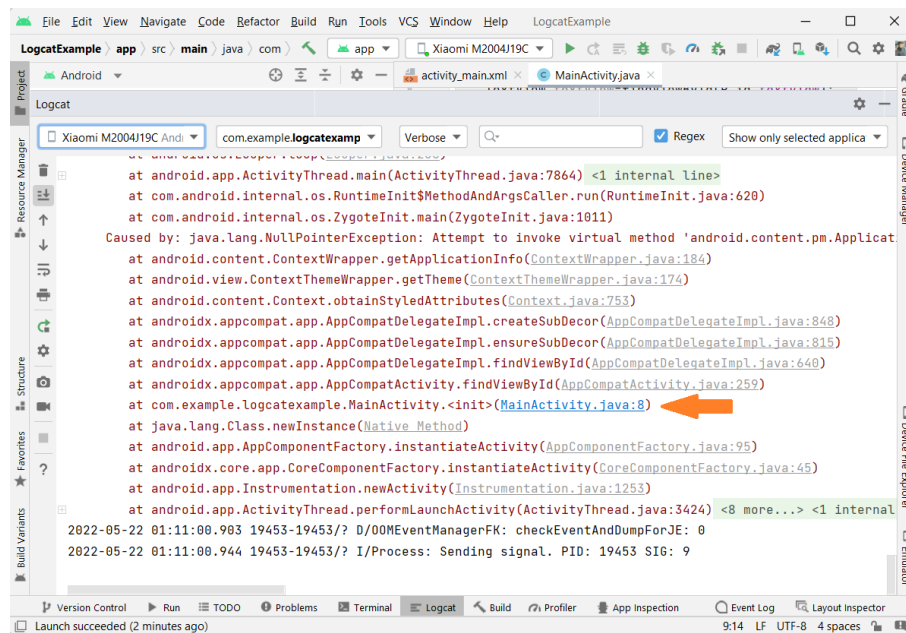


Figure 9.2: Logcat window with error

6. Go to MainActivity.java and added the below code inside onCreate() method.

```
Log.v("MainActivity1","Inside onCreate()");
Log.d("MainActivity2","Inside onCreate()");
Log.i("MainActivity3","Inside onCreate()");
Log.w("MainActivity4","Inside onCreate()");
Log.e("MainActivity5","Inside onCreate()");
```

7. Now run the application and open logcat window. In logcat we have a search option where we can check easily if message is printed or not by writing TAG name in search option.

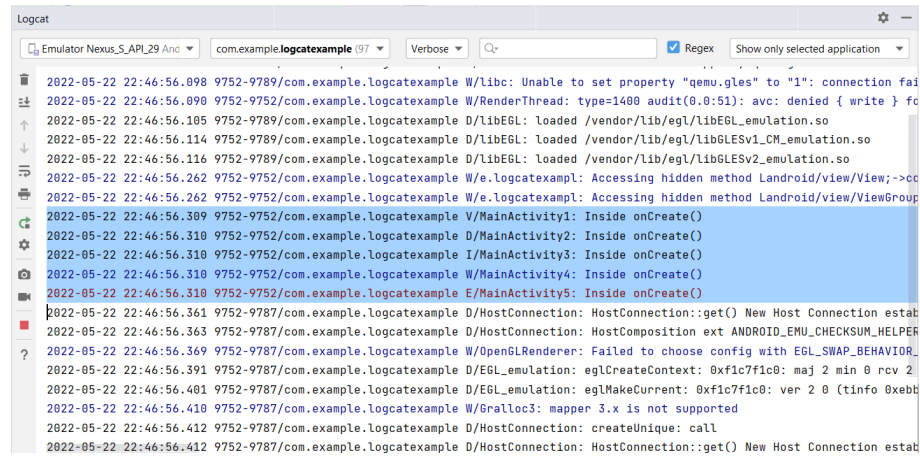


Figure 9.3: Logcat with custom message