

MOBILE APPLICATION DEVELOPEMENT LABORATORY

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

Bachelor of Technology

In

Information Technology



Submitted by :

Prabhnoor kaur (1905378)

Submitted to :

Prof. Ranjodh Kaur

**Guru Nanak Dev Engineering College,
Ludhiana-141006**

Table of Contents

| | | |
|-------|--|----|
| 1 | Android Development Environment: To study design aspects of development environment like Android, iOS. | 1 |
| 1.1 | Android | 1 |
| 1.2 | iOS | 2 |
| 2 | Android Development Environment: To setup Android studio2 and study its basic components. | 4 |
| 3 | Android User Interface Design: To study various XML files needed for interface design. | 10 |
| 4 | Android User Interface Design: To implement different type of layouts like relative, grid, linear and table | 14 |
| 4.1 | XML File | 14 |
| 5 | Apps Interactivity in Android: To incorporate element of interactivity using Android Fragment and Intent Class. | 16 |
| 5.1 | XML Files | 16 |
| 6 | Persistent Data Storage: To perform database connectivity of android app using SQLite | 24 |
| 7 | Android Services and Threads:To implement the concept of multithreading using Android Service class. | 34 |
| 7.0.1 | XML Code | 34 |
| 7.0.2 | Java Code | 34 |
| 7.0.3 | Output Screens: | 37 |
| 8 | Android Security and Debugging: To implement concept of permission and perform request for permission to access different hardware components of mobile. | 38 |
| 8.0.1 | Output Screens | 40 |
| 9 | Android Security and Debugging: To perform debugging and testing of android app using tools like Logcat, Android debug bridge, DDMS. | 42 |

1 Android Development Environment: To study design aspects of development environment like Android, iOS.

1.1 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. Following is the list of software's you will need before you start your Android application programming.

- Java JDK5 or later version
- Android Studio

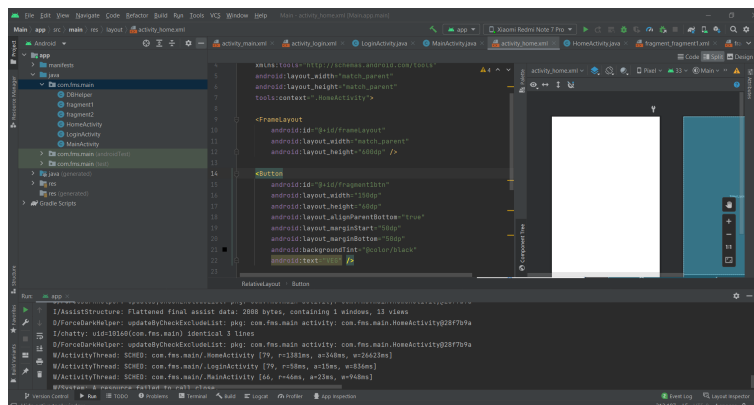


Figure 1.1: Android Studio

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

1.2 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.

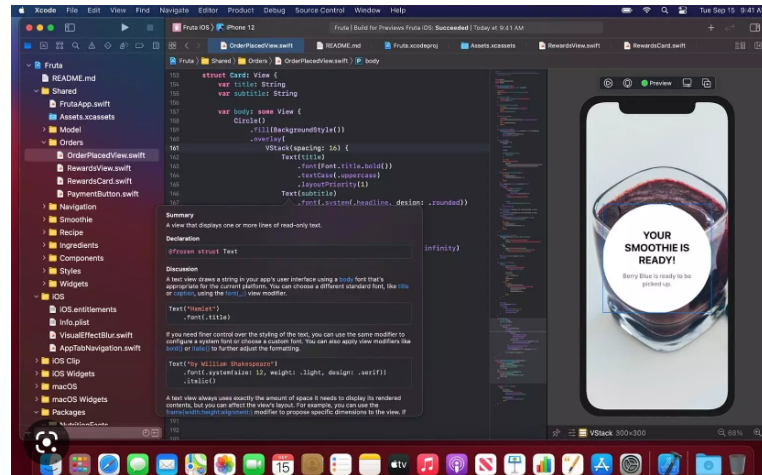


Figure 1.2: xcode iOS IDE

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.

One of the major advantages of iOS app development is the extensive collection of developer resources available to you. Because of the standardization, functionality and consistency of iOS app development, Apple is able to release native APIs and libraries as kits that are stable, feature-rich and easy to use. You can use these iOS SDKs to seamlessly integrate your app into Apple’s existing infrastructure.

For example, if you’re working on an app controller for a smart toaster oven, you can use HomeKit to standardize the communication between the toaster and the phone. There are kits for game development (such as SpriteKit, GameplayKit and ReplayKit), health apps, maps, cameras, as well as Siri, Apple’s virtual assistant.

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

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

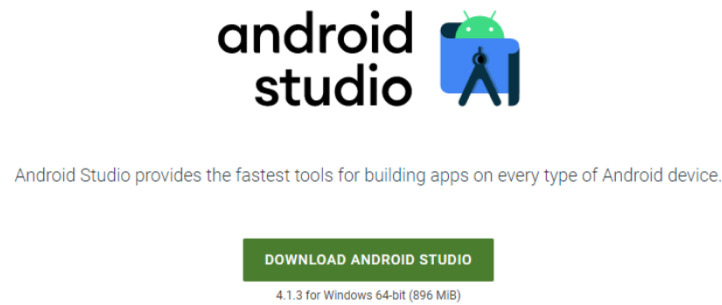


Figure 2.1: Download Android Studio Button.

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

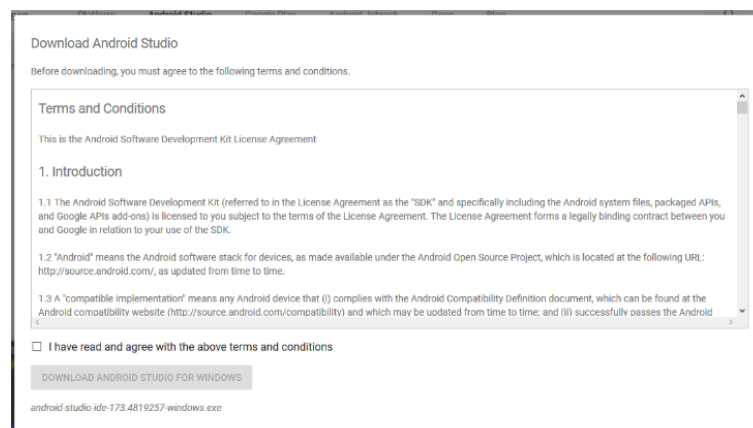


Figure 2.2: Terms and Conditions

Click on the Save file button in the appeared prompt box and the file will start downloading.

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



Figure 2.3: *Terms and Conditions*

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.

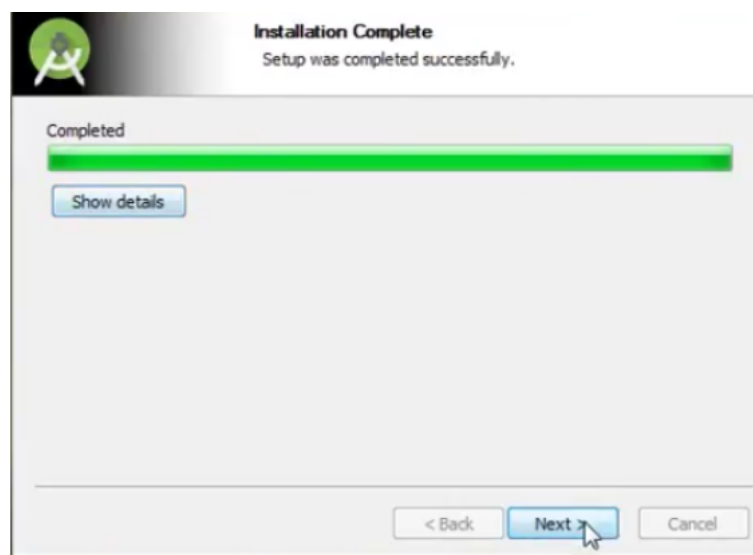


Figure 2.4: *Terms and Conditions*

Click on next.



Figure 2.5: *Terms and Conditions*

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’. Click on OK.
6. This will start the Android Studio.



Figure 2.6: *Terms and Conditions*

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

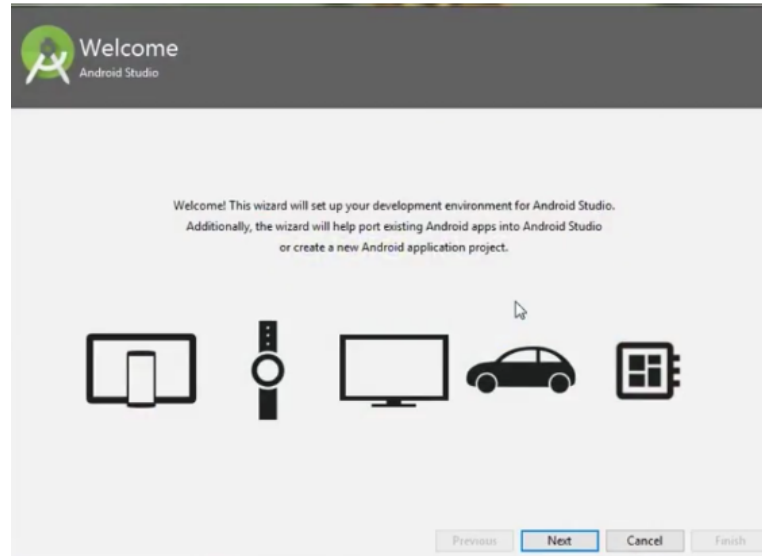


Figure 2.7: *Welcome dialog box*

Click on Next.

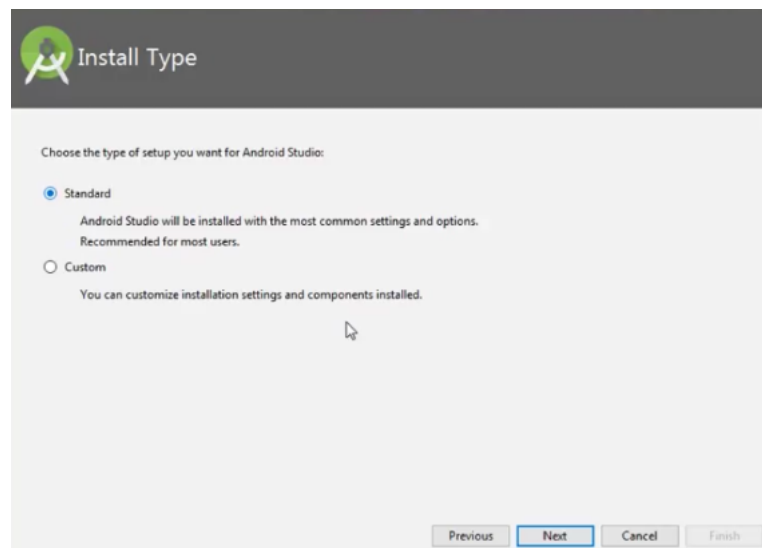


Figure 2.8: *Choose setup*

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.

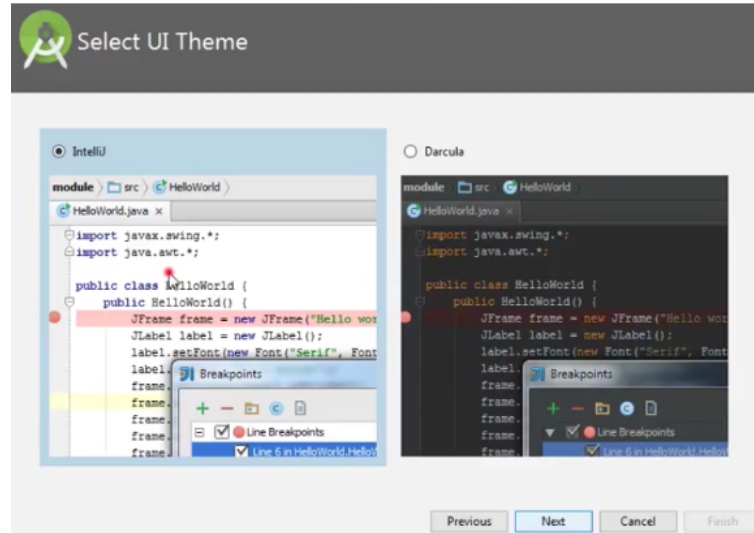


Figure 2.9: Choose Theme

Click on Next.

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

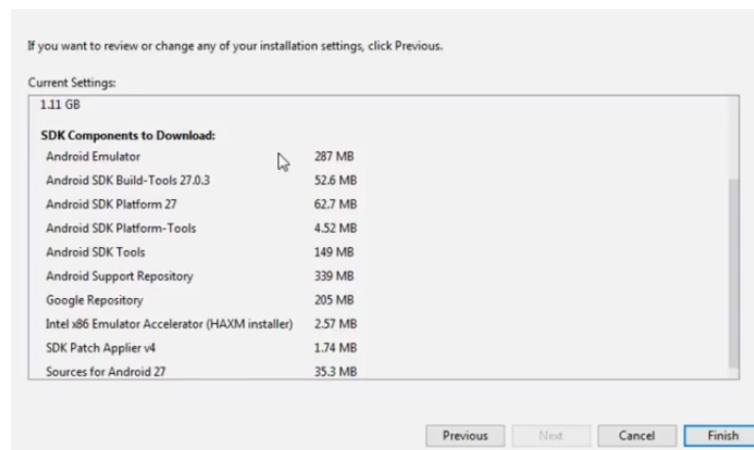


Figure 2.10: SDK components

Click on Finish.

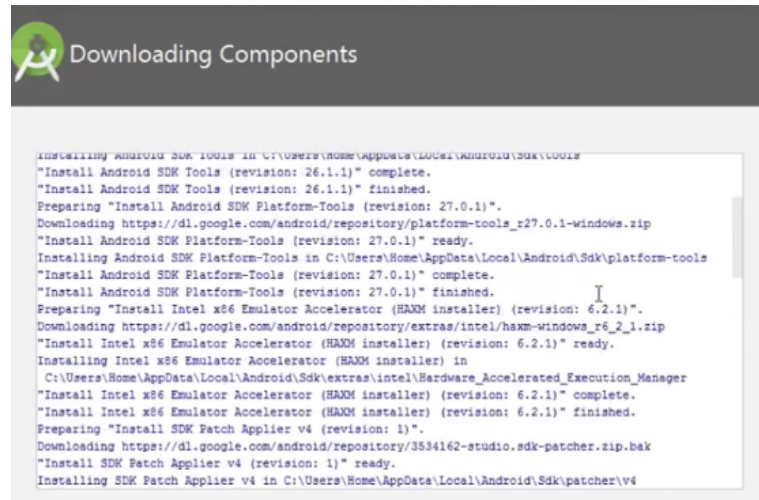


Figure 2.11: Downloading SDK components

Components begin to download let it complete.

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

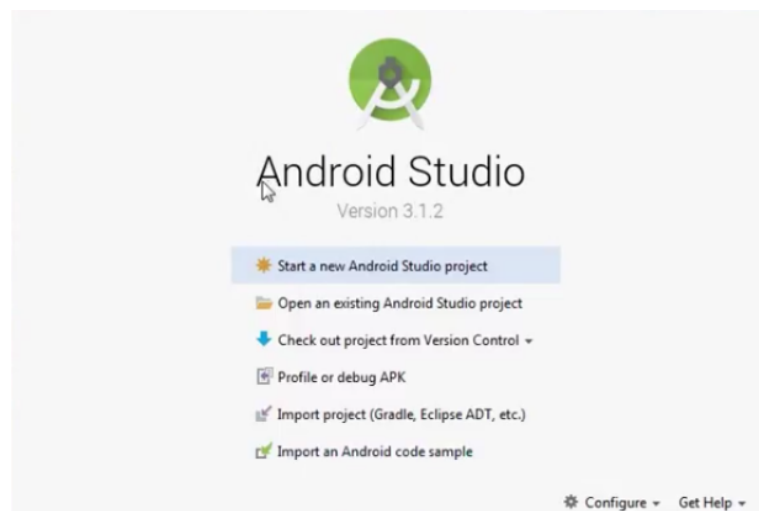


Figure 2.12: Start a new Android Studio project

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.

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.

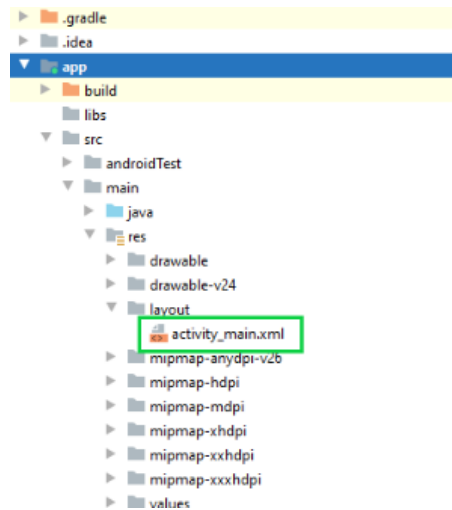


Figure 3.1: Location of Layout XML files

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.

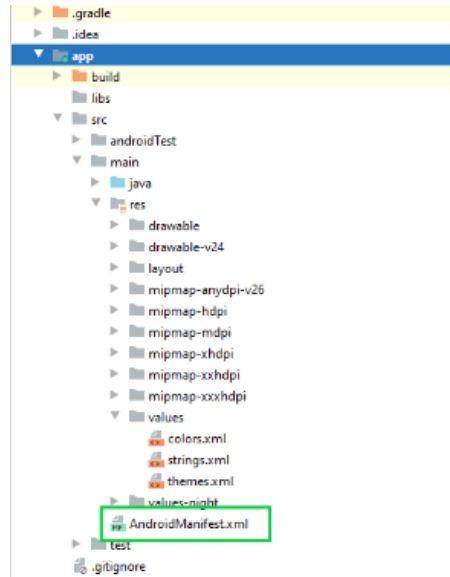


Figure 3.2: Location of AndroidManifest.xml

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.

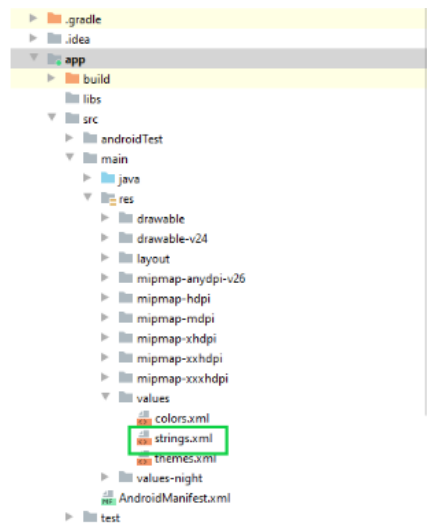


Figure 3.3: Location of strings.xml

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.

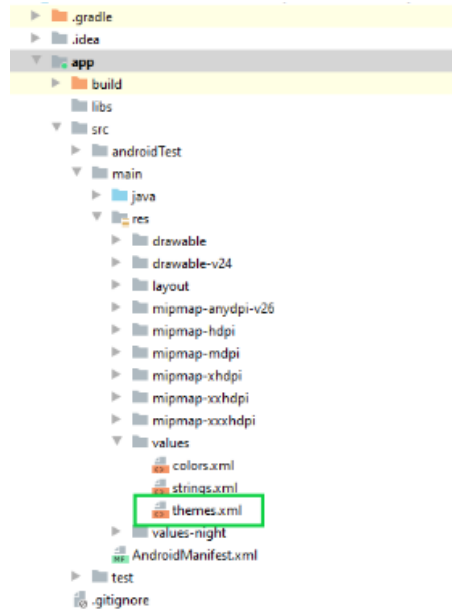


Figure 3.4: Location of themes.xml

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.

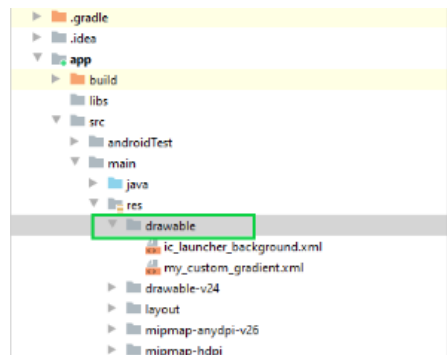


Figure 3.5: Location of Drawable XML

6. **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.

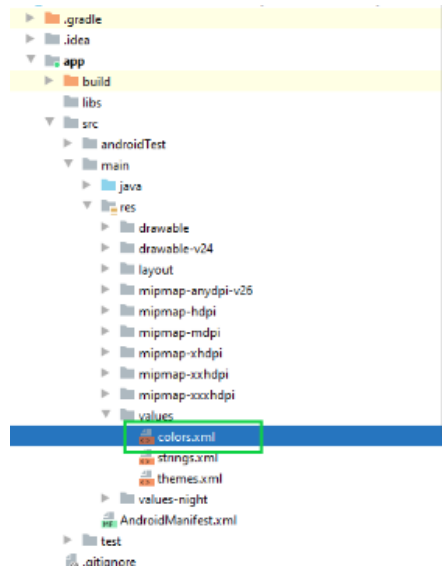


Figure 3.6: Location of colors.xml

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.

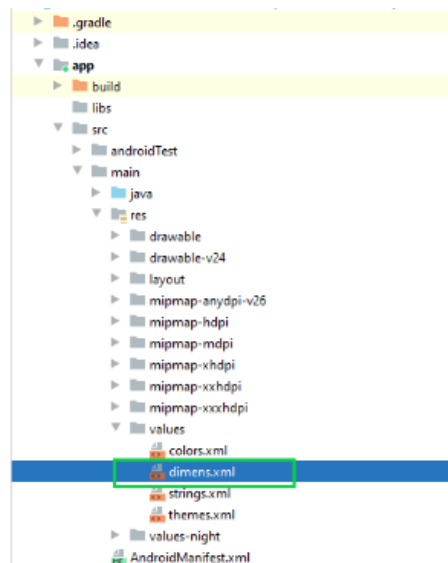


Figure 3.7: Location of dimens.xml

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

4.1 XML File

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#BECDD28"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="match_parent"
        android:layout_height="289dp"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="100dp"
        android:layout_marginRight="20dp"
        app:srcCompat="@drawable/dish2" />

    <LinearLayout
        android:layout_width="422dp"
        android:layout_height="157dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
            android:layout_width="216dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="120dp"
            android:layout_marginTop="50dp"
            android:layout_marginRight="50dp"
            android:layout_weight="1"
            android:text="Chinese"
            android:textSize="34sp" />

    </LinearLayout>

    <EditText
```



```

        android:id="@+id/editTextTextPersonName4"
        android:layout_width="192dp"
        android:layout_height="88dp"
        android:layout_marginStart="110dp"
        android:layout_weight="1"
        android:ems="10"
        android:hint="Add new Dish!"
        android:inputType="textPersonName" />

<Button
    android:id="@+id/button3"
    android:layout_width="117dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="140dp"
    android:backgroundTint="#1F6E22"
    android:text="Add Dish" />

<Button
    android:id="@+id/button2"
    android:layout_height="wrap_content"
    android:backgroundTint="#EC2517"
    android:text="Decide" />

</LinearLayout>

```



Figure 4.1: Multi Layout App

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

5.1 XML Files

1. activity_main4.xml:

```
?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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="#B388FF"
    android:backgroundTint="#FFD180"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="600dp"
        android:background="#A7FFEB"
        android:backgroundTint="#A7FFEB" />

    <Button
        android:id="@+id/fragment1btn"
        android:layout_width="150dp"
        android:layout_height="60dp"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="50dp"
        android:layout_marginEnd="30dp"
        android:layout_marginRight="20dp"
        android:layout_marginBottom="50dp"
        android:backgroundTint="@color/purple_200"
        android:text="Fragment_1" />

    <Button
        android:id="@+id/fragment2btn"
        android:layout_width="150dp"
        android:layout_height="60dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="50dp"
        android:layout_marginBottom="50dp"
```

```

        android:backgroundTint="@color/teal_700"
        android:text="Fragment_2" />

```

```

</RelativeLayout>

```

2. fragment_1.xml:-

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EA80FC"
    tools:context=".fragment1">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginTop="50dp"
        android:text="Fragment 1"
        android:textColor="@color/black"
        android:textSize="30dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/dataFrom2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:text=""
        android:textColor="@color/black"
        android:textSize="26dp" />

    <EditText
        android:id="@+id/fragment1Data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="26dp"
        android:layout_marginTop="300dp"
        android:ems="10"
        android:hint="Data to Fragment 2"
        android:inputType="textPersonName"
        android:minHeight="48dp" />

```

```

<Button
    android:id="@+id/fragment1btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/fragment1Data"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="78dp"
    android:backgroundTint="@color/black"
    android:text="Send Data to Fragment 2" />

</RelativeLayout>

```

3. fragment_2.xml:-

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#009688"
    tools:context=".fragment2">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"
        android:text="Fragment 2"
        android:textColor="@color/black"
        android:textSize="30dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/dataFrom1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:text=""
        android:textColor="@color/black"
        android:textSize="26dp" />

```

```

<EditText
    android:id="@+id/fragment2Data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginHorizontal="26dp"
    android:layout_marginTop="300dp"
    android:ems="10"
    android:hint="Data□to□Fragment□1"
    android:inputType="textPersonName"
    android:minHeight="48dp" />

<Button
    android:id="@+id/fragment2btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/fragment2Data"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="80dp"
    android:backgroundTint="@color/black"
    android:text="Send□Data□to□Fragment□1" />

</RelativeLayout>

```

4. fragment_1.java:-

```

package com.fms.fragment;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentResultListener;

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

public class fragment1 extends Fragment {

    View view;

```

```

@Override
public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {

    // Inflate the layout for this fragment
    view = inflater.inflate(R.layout.fragment_fragment1,
        container, false);

    Button button = view.findViewById(R.id.fragment1btn);

    getParentFragmentManager().setFragmentResultListener
        ("dataFrom2", this, new FragmentResultListener() {
        @Override
        public void onFragmentResult(@NonNull String requestKey,
            @NonNull Bundle result) {

            String data = result.getString("df2");
            TextView textView = view.findViewById(R.id.dataFrom2);
            textView.setText(data);

        }
    });

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

            EditText editText = view.findViewById(R.id.fragment1Data);
            Bundle result = new Bundle();
            result.putString("df1", editText.getText().toString());
            getParentFragmentManager().setFragmentResult("dataFrom1",
                result);
            editText.setText("");
        }
    });

    return view;
}
}

```

5. fragment_2.java:-

```

package com.fms.fragment;

```

```

import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentResultListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class fragment2 extends Fragment {

    View view;
    @Override
    public View onCreateView(LayoutInflater inflater,
    ViewGroup container,
    Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        view = inflater.inflate(R.layout.fragment_fragment2,
        container, false);

        Button button = view.findViewById(R.id.fragment2btn);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                EditText editText = view.findViewById(R.id.fragment2Data);
                Bundle result = new Bundle();
                result.putString("df2", editText.getText().toString());
                getParentFragmentManager().setFragmentResult("dataFrom2", result);
                editText.setText("");

            }
        });

        getParentFragmentManager().setFragmentResultListener
        ("dataFrom1", this, new FragmentResultListener() {
            @Override
            public void onFragmentResult(@NonNull String requestKey,
            @NonNull Bundle result) {

                String data = result.getString("df1");

```

```

TextView textView = view.findViewById(R.id.dataFrom1);
textView.setText(data);

}
});

return view;
    }
}

```

OUTPUTS:



Figure 5.1: Fragnmet 1



Figure 5.2: Fragment 2

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

DBHelper.java

```
package com.fms.sqlite;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) {
        super(context, "Userdata.db", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase DB) {
        DB.execSQL("create Table Userdetails(name TEXT primary key, contact TEXT, dob TEXT)");
    }
    @Override
    public void onUpgrade(SQLiteDatabase DB, int i, int ii) {
        DB.execSQL("drop Table if exists Userdetails");
    }
    public Boolean insertuserdata(String name, String contact, String dob)
    {
        SQLiteDatabase DB = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("name", name);
        contentValues.put("contact", contact);
        contentValues.put("dob", dob);
        long result=DB.insert("Userdetails", null, contentValues);
        if(result==-1){
            return false;
        }else{
            return true;
        }
    }
    public Boolean updateuserdata(String name, String contact, String dob)
    {
        SQLiteDatabase DB = this.getWritableDatabase();
```

```

ContentValues contentValues = new ContentValues();
contentValues.put("contact", contact);
contentValues.put("dob", dob);
Cursor cursor = DB.rawQuery("Select * from Userdetails
where name = ?", new String[]{name});
if (cursor.getCount() > 0) {
long result = DB.update("Userdetails", contentValues,
"name=?", new String[]{name});
if (result == -1) {
return false;
} else {
return true;
}
} else {
return false;
}
}

public Boolean deletedata (String name)
{
SQLiteDatabase DB = this.getWritableDatabase();
Cursor cursor = DB.rawQuery("Select * from Userdetails
where name = ?", new String[]{name});
if (cursor.getCount() > 0) {
long result = DB.delete("Userdetails", "name=?", new
String[]{name});
if (result == -1) {
return false;
} else {
return true;
}
} else {
return false;
}
}

public Cursor getdata ()
{
SQLiteDatabase DB = this.getWritableDatabase();
Cursor cursor = DB.rawQuery("Select * from Userdetails", null);
return cursor;
}
}

```

MainActivity.java

```

package com.fms.sqlite;

```

```

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    EditText name, contact, dob;
    Button insert, update, delete, view;
    DBHelper DB;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        name = findViewById(R.id.name);
        contact = findViewById(R.id.contact);
        dob = findViewById(R.id.dob);
        insert = findViewById(R.id.btnInsert);
        update = findViewById(R.id.btnUpdate);
        delete = findViewById(R.id.btnDelete);
        view = findViewById(R.id.btnView);
        DB = new DBHelper(this);
        // SQLiteDatabase db = DB.getReadableDatabase()
        insert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String nameTXT = name.getText().toString();
                String contactTXT = contact.getText().toString();
                String dobTXT = dob.getText().toString();

                Boolean checkinsertdata = DB.insertuserdata(nameTXT,
                    contactTXT, dobTXT);
                if(checkinsertdata==true)
                    Toast.makeText(MainActivity.this, "New Entry Inserted",
                        Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(MainActivity.this, "New Entry Not Inserted",
                        Toast.LENGTH_SHORT).show();
            }
        });
        update.setOnClickListener(new View.OnClickListener() {

```

```

@Override
public void onClick(View view) {
    String nameTXT = name.getText().toString();
    String contactTXT = contact.getText().toString();
    String dobTXT = dob.getText().toString();

    Boolean checkupdatedata = DB.updateuserdata(nameTXT,
        contactTXT, dobTXT);
    if(checkupdatedata==true)
        Toast.makeText(MainActivity.this, "Entry Updated",
            Toast.LENGTH_SHORT).show();
    else
        Toast.makeText(MainActivity.this, "New Entry Not Updated",
            Toast.LENGTH_SHORT).show();
}

delete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String nameTXT = name.getText().toString();
        Boolean checkdeletedata = DB.deletedata(nameTXT);
        if(checkdeletedata==true)
            Toast.makeText(MainActivity.this, "Entry Deleted",
                Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(MainActivity.this, "Entry Not Deleted",
                Toast.LENGTH_SHORT).show();
    }
});

view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Cursor res = DB.getdata();
        if(res.getCount()==0){
            Toast.makeText(MainActivity.this, "No Entry Exists",
                Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append("Name :"+res.getString(0)+"\n");
            buffer.append("Contact :"+res.getString(1)+"\n");
            buffer.append("Date of Birth :"+res.getString(2)+"\n\n");
        }
    }
});

```

```

AlertDialog.Builder builder = new AlertDialog.Builder
(MainActivity.this);
builder.setCancelable(true);
builder.setTitle("User Entries");
builder.setMessage(buffer.toString());
builder.show();
}
});
}}

```

Activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#D5DCDC"
android:backgroundTint="#F6F6FA"
android:orientation="vertical"
tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="662dp"
        android:background="#FAF7F7">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingTop="15dp">

            <ImageView
                android:id="@+id/imageView5"
                android:layout_width="match_parent"
                android:layout_height="155dp"
                app:srcCompat="@drawable/spoyify" />

            <View
                android:id="@+id/divider2"
                android:layout_width="match_parent"
                android:layout_height="1dp"
                android:layout_marginTop="20dp"
                android:background="?android:attr/listDivider" />

```

```

<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="50dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="40dp"
    android:text="SIGN UP WITH FACEBOOK "
    app:cornerRadius="23pt" />

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true">

    <TextView
        android:id="@+id/tvText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:text="or" />

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_centerVertical="true"
        android:layout_marginLeft="16dp"
        android:layout_toLeftOf="@id/tvText"
        android:background="?android:attr/listDivider" />

/>

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_centerVertical="true"
        android:layout_marginRight="16dp"
        android:layout_toRightOf="@id/tvText"
        android:background="?android:attr/listDivider" />

/>

```

```
</RelativeLayout>
```

```
<TextView
```

```
    android:id="@+id/textView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="70dp"
    android:backgroundTint="#121111"
    android:text="Sign up with your email address"
    android:textColor="#070707"
    android:textStyle="bold" />
```

```
<EditText
```

```
    android:id="@+id/editTextTextEmailAddress2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/image_border"
    android:ems="10"
    android:hint="Email "
```

```
    android:includeFontPadding="true"
    android:inputType="textEmailAddress"
    android:minHeight="48dp"
    android:textSize="16sp" />
```

```
<EditText
```

```
    android:id="@+id/editTextTextEmailAddress3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/image_border"
    android:ems="10"
    android:hint="Confirm Email"
    android:inputType="textEmailAddress"
```

```
    android:minHeight="48dp"
    android:textSize="16sp" />
```



```

<EditText
    android:id="@+id/editTextTextPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"

    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/image_border"
    android:ems="10"

    android:hint="Password"

    android:inputType="textPassword"
    android:minHeight="48dp"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    android:textSize="16sp" />

<EditText
    android:id="@+id/editTextTextPersonName2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/image_border"
    android:ems="10"
    android:hint="What should we call you?"
    android:inputType="textPersonName"
    android:minHeight="48dp"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    android:textSize="16sp" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="20dp"
    android:text="Date of Birth:" />

```

```

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="25dp"
            android:layout_marginEnd="25dp"
            android:orientation="horizontal">

            <Spinner
                android:id="@+id/spinner3"
                android:layout_width="200dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                tools:ignore="TouchTargetSizeCheck,
                SpeakableTextPresentCheck,SpeakableTextPresentCheck"

            <Spinner
                android:id="@+id/spinner1"
                android:layout_width="100dp"
                android:layout_height="match_parent"
                tools:ignore="TouchTargetSizeCheck,
                SpeakableTextPresentCheck,
                SpeakableTextPresentCheck" />

            <EditText
                android:id="@+id/editTextTextPersonName"
                android:layout_width="wrap_content"
                android:layout_height="48dp"
                android:layout_weight="1"
                android:ems="10"
                android:hint="Year"
                android:inputType="textPersonName" />
        </LinearLayout>

    </LinearLayout>
</ScrollView>


<Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Sign up" />

</LinearLayout>

```

Output:

spotify



SIGN UP WITH FACEBOOK

or

Sign up with your email address

Email

Confirm Email

Password

What should we call you?

Date of Birth:

Year

SIGN UP

Figure 6.1: Spotify App

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"?>
<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:gravity="center_horizontal"
    android:layout_marginTop="100dp"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/edit_query"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter_string" />
    <Button
        android:id="@+id/click"
        android:layout_marginTop="50dp"
        style="@style/Base.TextAppearance.AppCompat.Widget.
        Button.Borderless.Colored"
        android:layout_width="wrap_content"
        android:background="#c1c1c1"
        android:textColor="#FFF"
        android:layout_height="wrap_content"
        android:text="Button" />
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

7.0.2 Java Code

```

package com.fms.pass_check;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    EditText edit_query;
    TextView textView;
    TextView text1;
    boolean twice = false;
    Thread t = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        edit_query = findViewById(R.id.edit_query);
        textView = findViewById(R.id.text);
        text1 = findViewById(R.id.text1);
        findViewById(R.id.click).setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View v) {
                runthread();
                runthread1();
            }
        });
    }

    private void runthread1() {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                text1.setText("Hi_▯Prabhnoor");
            }
        });
    }
}

```

```

        });
    }

    private void runthread() {
        twice = true;
        if (twice) {
            final String s1 = edit_query.getText().toString();
            t = new Thread(new Runnable() {
                @Override
                public void run() {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            textView.setText(t.getName());
                            twice = false;
                        }
                    });
                }
            });
            t.start();
            t.setName(s1);
            t.setPriority(Thread.MAX_PRIORITY);
        }
    }
}

```

7.0.3 Output Screens:

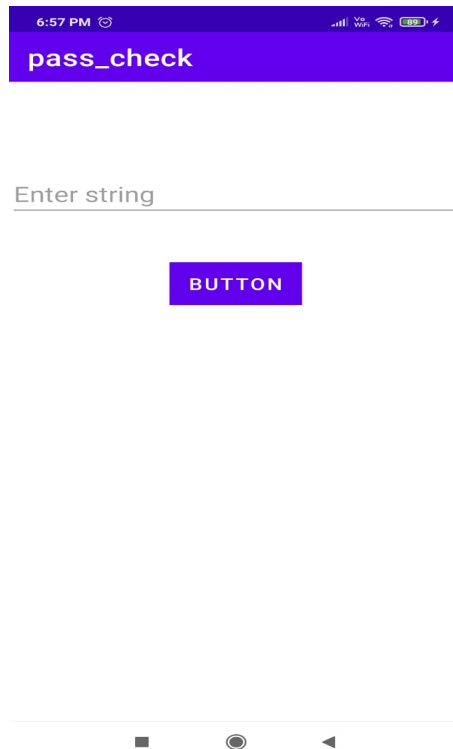


Figure 7.1: Main Screen

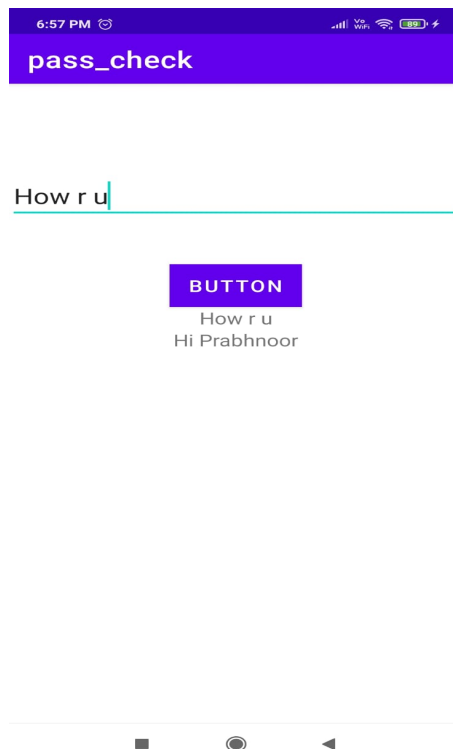


Figure 7.2: Thread Screen

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
    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
    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.

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. `MainActivity.java` code:

```
@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

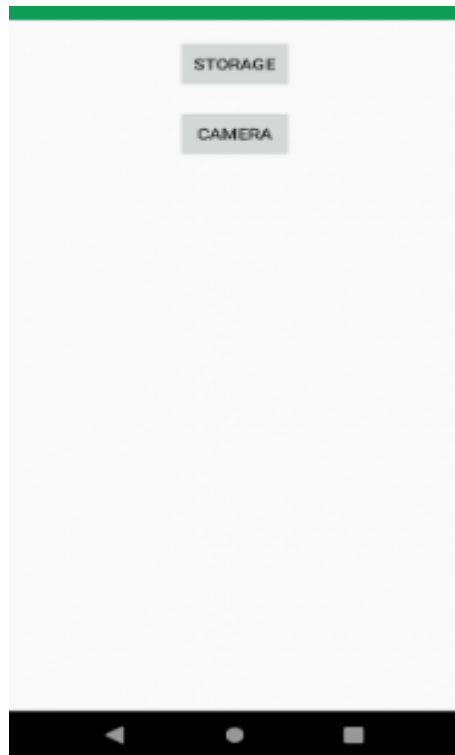


Figure 8.1: On starting the application

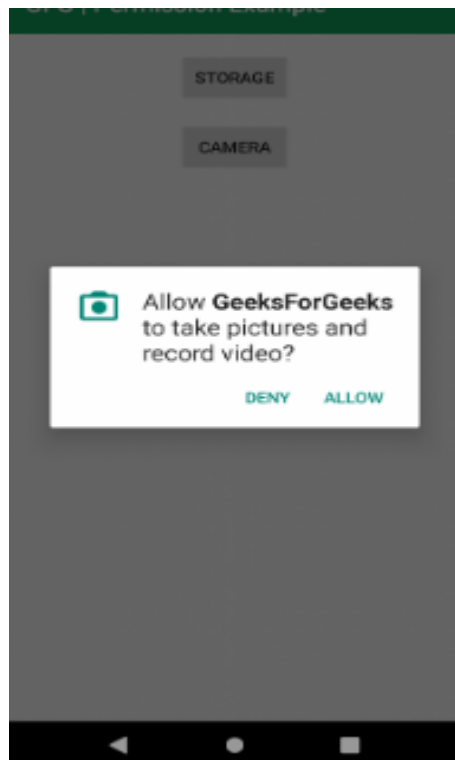


Figure 8.2: On clicking the camera button for the first time

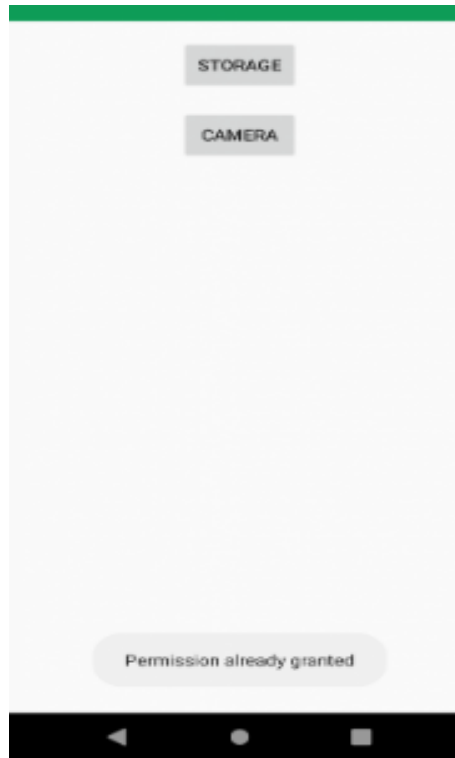


Figure 8.3: On Granting the permission

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. Logcat Window is going to help you to debug the output by collecting and viewing all the messages that your emulator throws. So, this is a very useful component for the app development because this Logcat dumps a lot of system messages and these messages are actually thrown by the emulator. This means, that when you run your app in your emulator, then you will see a lot of messages which include all the information, all the verbose messages, and all the errors that you are getting in your application.

Java file

```
public class MainActivity extends AppCompatActivity {

    EditText username, password, repassword;
    Button signup, signin;
    DBHelper DB;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.v("MainActivity1", "Verbose message");
        Log.d("Check", "Successfull");
        Log.i("Odd", "Visible");

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Output in logcat window:

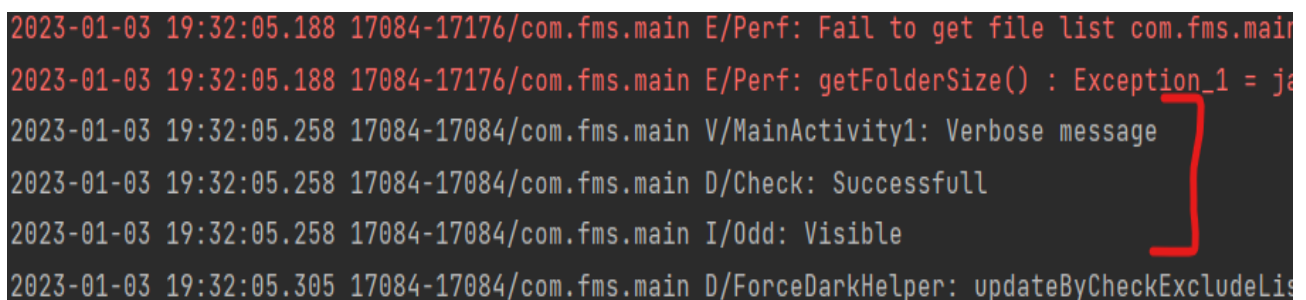
A screenshot of the Android Studio Logcat window. It displays a list of log messages. The messages are color-coded: red for errors, green for verbose, blue for debug, and grey for info. A red bracket on the right side of the screen groups three specific messages: a verbose message, a debug message, and an info message. The messages are as follows:
2023-01-03 19:32:05.188 17084-17176/com.fms.main E/Perf: Fail to get file list com.fms.main
2023-01-03 19:32:05.188 17084-17176/com.fms.main E/Perf: getFolderSize() : Exception_1 = java.io.IOException: Permission denied
2023-01-03 19:32:05.258 17084-17084/com.fms.main V/MainActivity1: Verbose message
2023-01-03 19:32:05.258 17084-17084/com.fms.main D/Check: Successfull
2023-01-03 19:32:05.258 17084-17084/com.fms.main I/Odd: Visible
2023-01-03 19:32:05.305 17084-17084/com.fms.main D/ForceDarkHelper: updateByCheckExcludeList

Figure 9.1: Custom Messages