DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

Subject Name: MAD LAB

Subject Code: 20CSP356

Submitted to:

Faculty name

Er. Kuldeep Kumar

E13820

Submitted by:

Name: Vikash Yadav

UID: 21BCS8093

Section: 20BCS_DM-719

Group: B

INDEX

S.NO	PROGRAM	EVALUATION				SIGN
		LW (12)	VV (8)	FW (10)	Total (30)	
1.	Installing and running applications on android studio.					
2.	To perform basic designing using android studio.					
3.	Create Application by Using Widgets					
4.	Create an application that takes the name from a text box and shows a hello message along with the name entered in text box when the user clicks the OK button.					
5.	Create an Android App using various controls such TextEdit, CheckBox, RadioButton, RadioGroup, etc.					
6.	Create SMS sending android application using XML and JAVA.					
7.	Creating the Application Choosing Options Radio Group					
8.	Create an Android App using fragments.					
9.	Implement building blocks for android application using different layouts					
10.	Design the Android application using Menus and action bar.					

Experiment: 1.1	
Student Name: Vikash Yadav	**UID: 21BCS8093**
Branch: CSE	**Section/Group:** 719/B
Semester: 6	**Date of Performance: 16.02.2023**
Subject Name: Mobile Application Development Lab	
Subject Code: 20CSP-356	

1. **Aim**: Installing and running applications on android studio.

2. **Objective:** Downloading and installing android studio in the system and setting up the environments & configurations for development.

3. **Script and Output:**

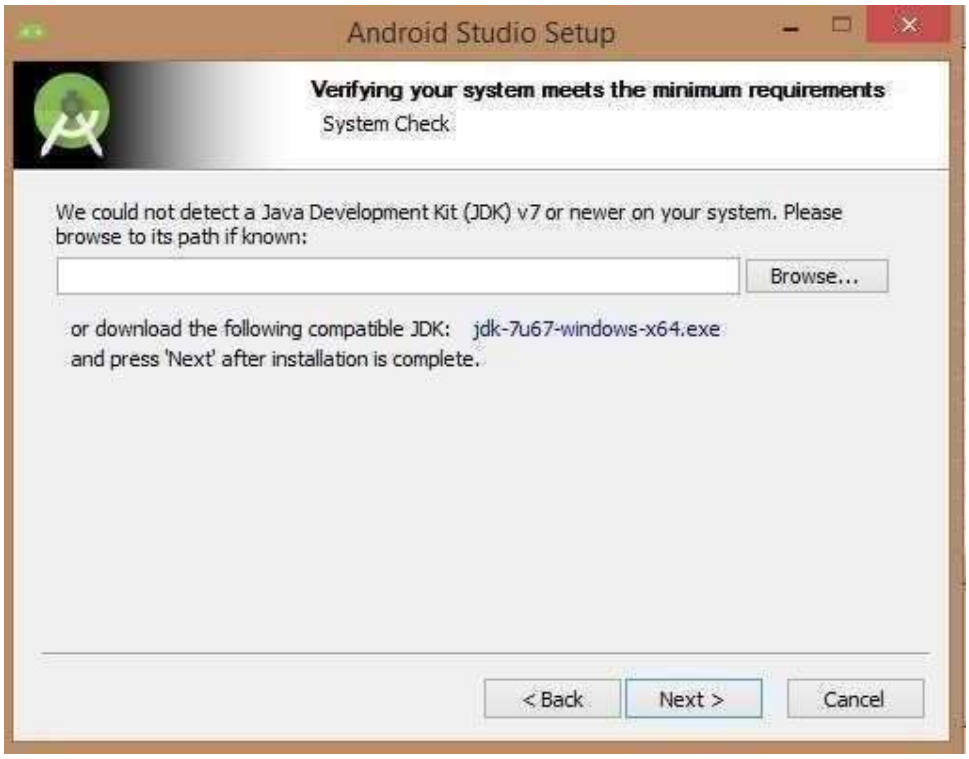
Step 1 - System Requirements

The required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.
Java JDK5 or later version
Java Runtime Environment (JRE)
6Android Studio

Step 2 - Setup Android Studio

Android Studio is the official IDE for android application development. It works based on IntelliJ IDEA, You can download the latest version of android studio from Android Studio 2.2 Download, If you are new to installing Android Studio on windows, you will find a file, which is named as android-studio-bundle-143.3101438-windows.exe.So just download and run on windows machine according to android studio wizard guideline.
If you are installing Android Studio on Mac or Linux, You can download the latest version from Download, or Android Studio Linux Download, check the instructions provided along with the downloaded file for Mac OS and Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 8.1 operating system. Installation
So let's launch Android Studio.exe, Make sure before launch Android Studio, Our Machine should require installed Java JDK. To install Java JDK, take a references of Android environment setup

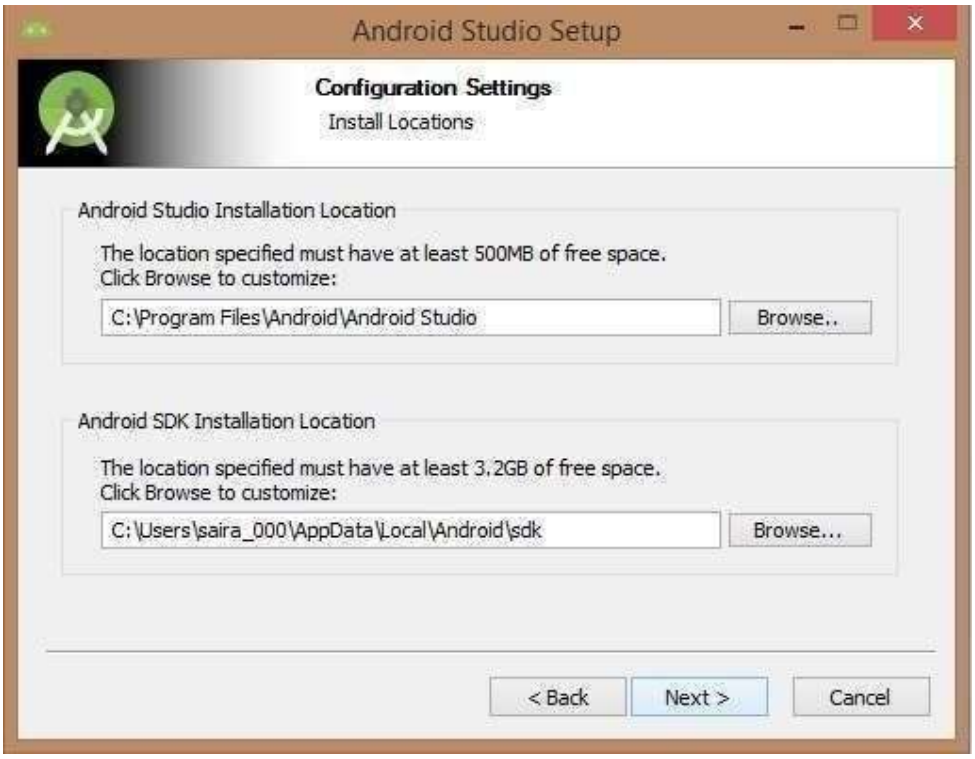
Once you launched Android Studio, its time to mention JDK7 path or later version in android studio installer.



Below the image initiating JDK to android SDK

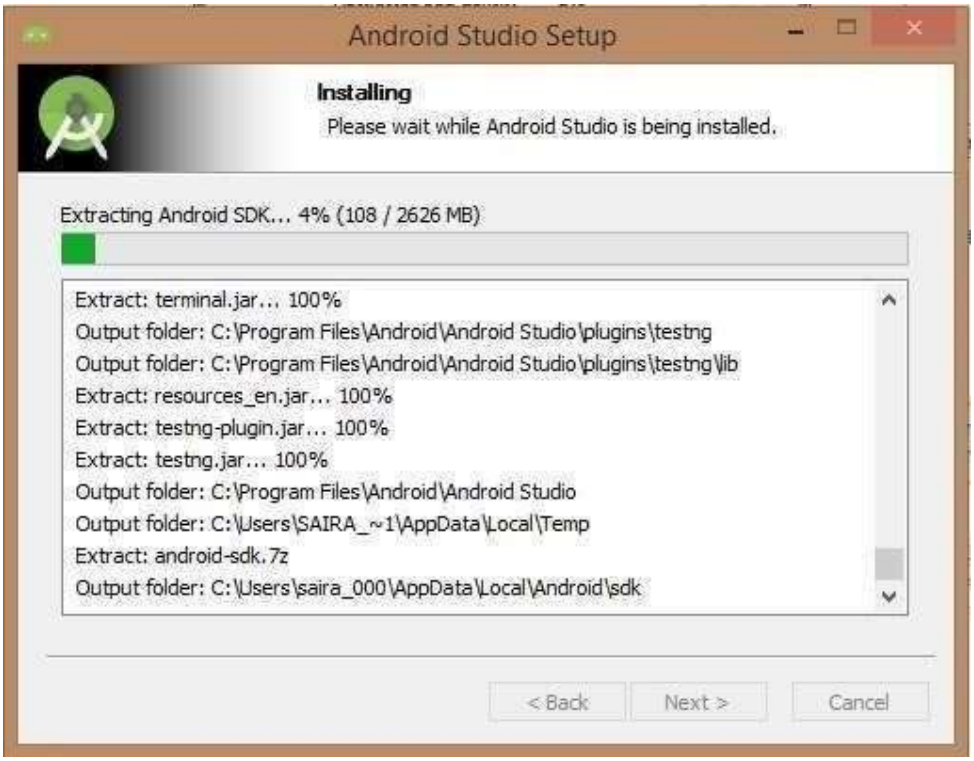
Need to check the components, which are required to create applications, below the image has selected Android Studio, Android SDK, Android Virtual Machine and performance (Intel chip).

Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.



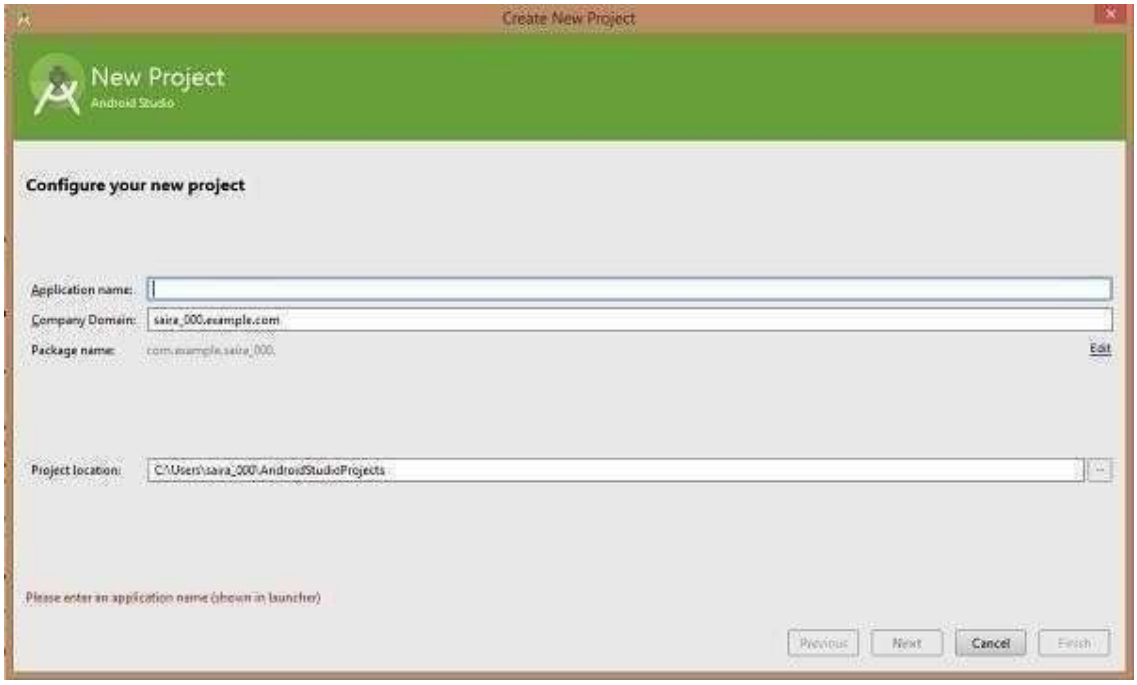
Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.

At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.

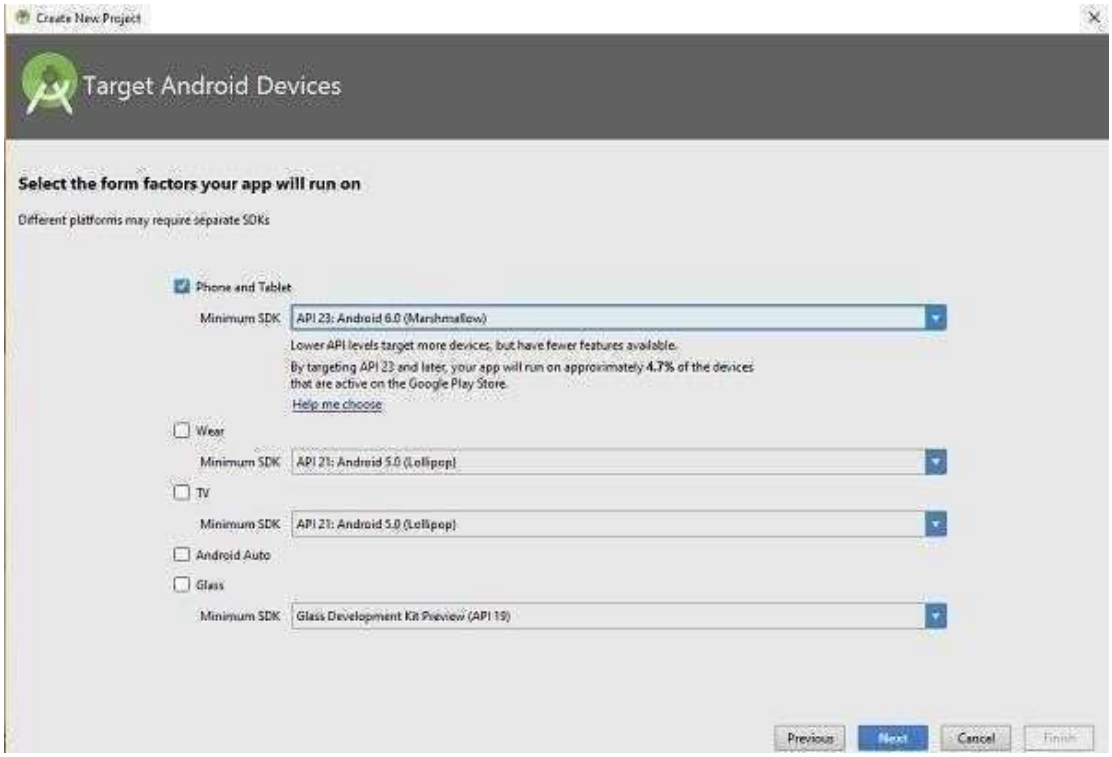


After done all above steps perfectly, you must get finish button and it going to be open android studio project with Welcome to android studio message as shown below

You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.

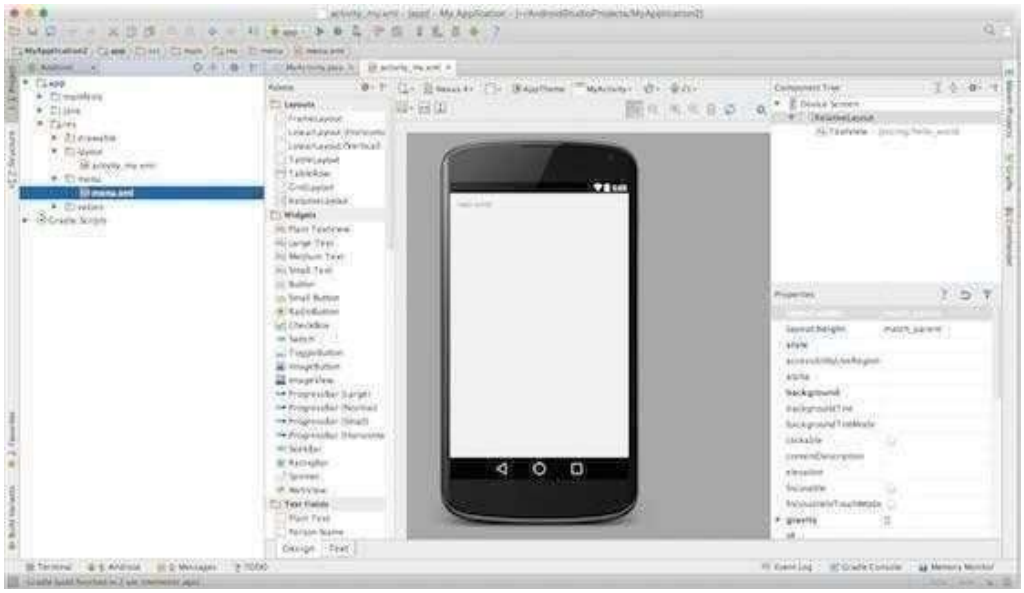


After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Marshmallow)



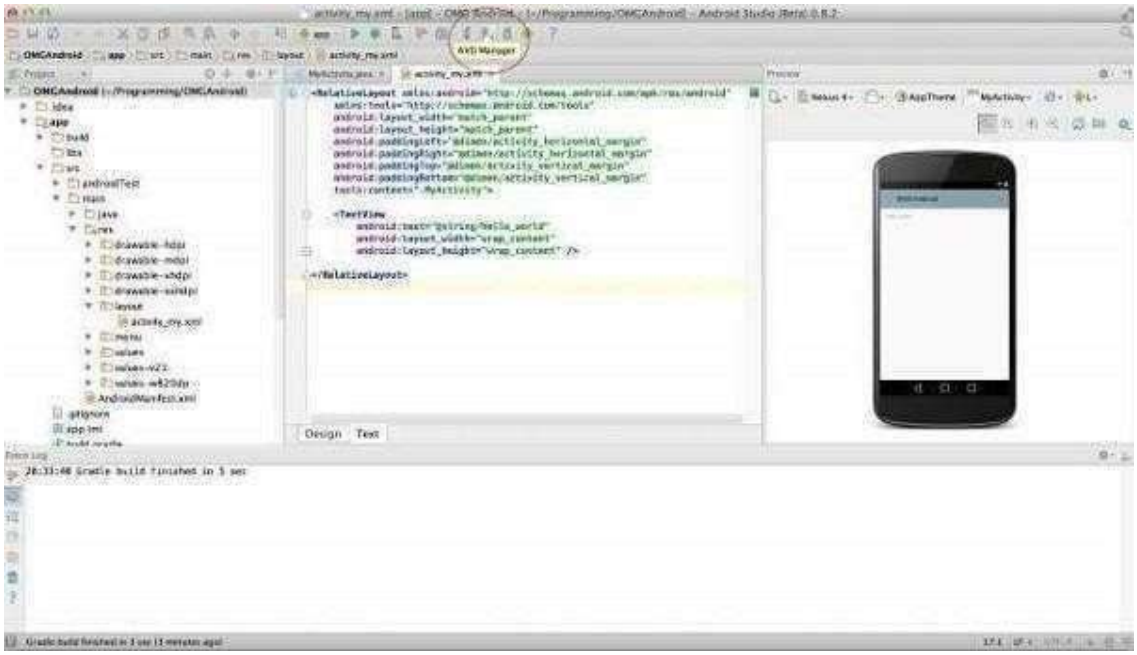
The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications

At the final stage it going to be open development tool to write the application code.



Step 3 - Create Android Virtual Device

To test your Android applications, you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager Clicking AVD_Manager icon as shown below



After Click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking Create new Virtual device button

If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re-start your machine and once you are done with this last step, you are ready to proceed for your first

Android example but before that we will see few more important concepts related to Android Application Development.

Experiment1.2

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: BE-CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 23/02/2023**
Subject Name: MAD Lab **Subject Code: 20CSP-356**

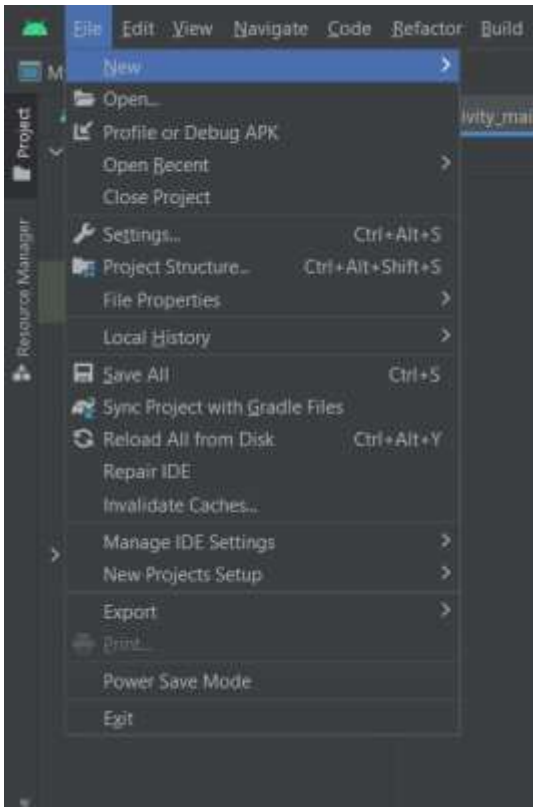
1. Aim:

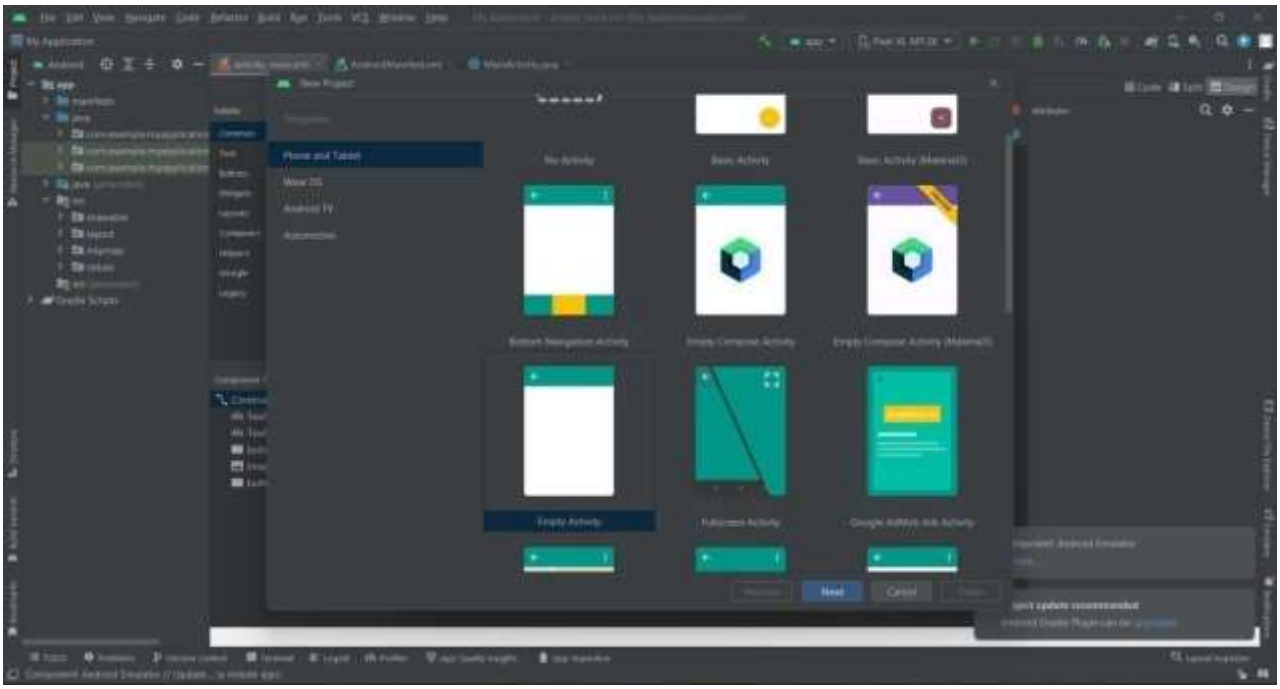
To perform basic designing using android studio.

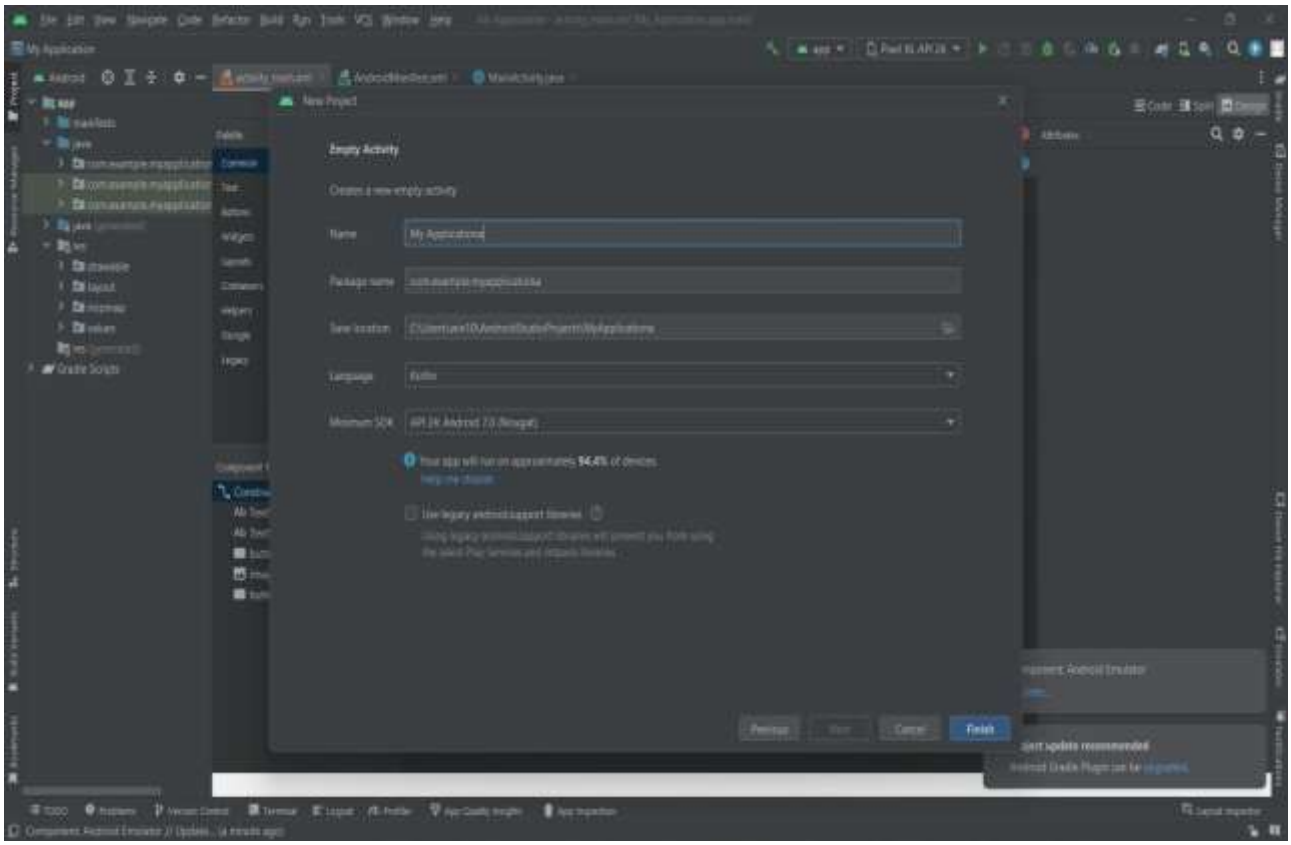
2. Objective:

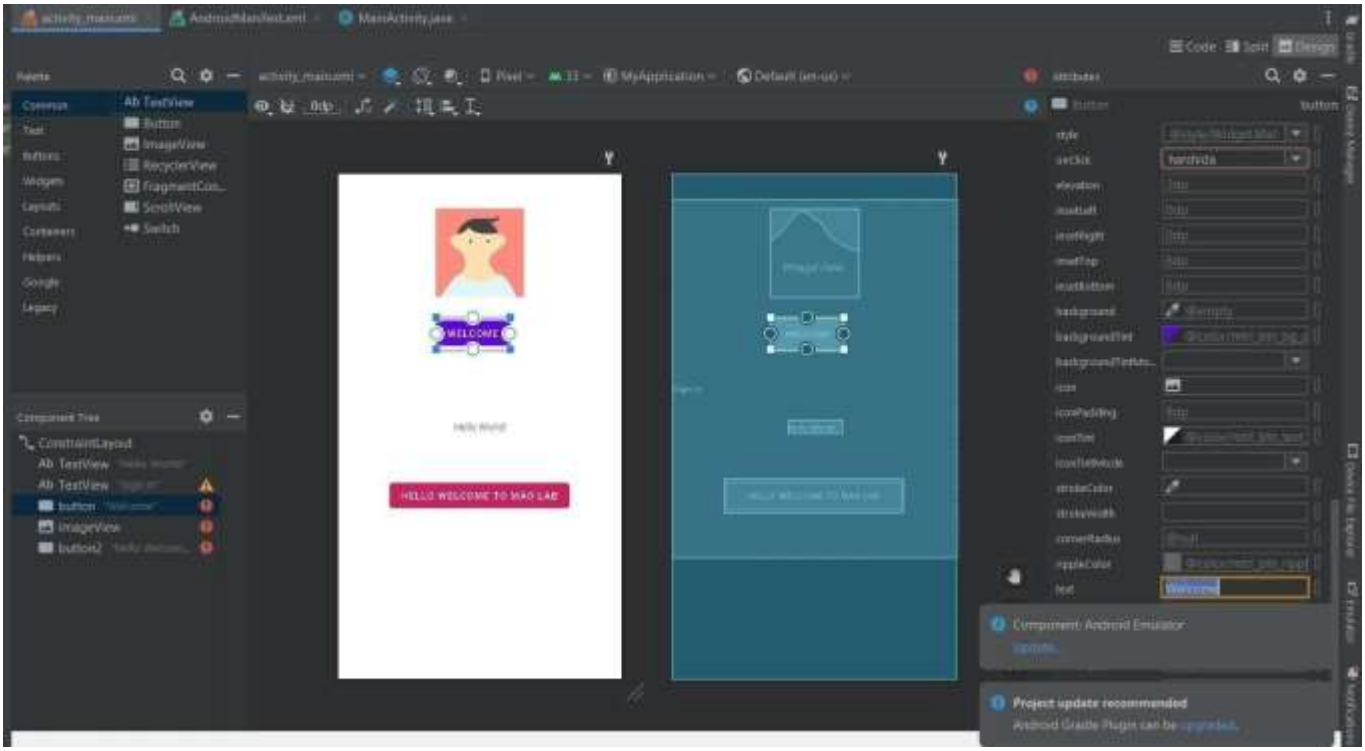
- The main objective is to learn designing android apps which are able to run onvarious mobile devices.
- To learn about gradle, xml structure, java code, resources and project in android studio.

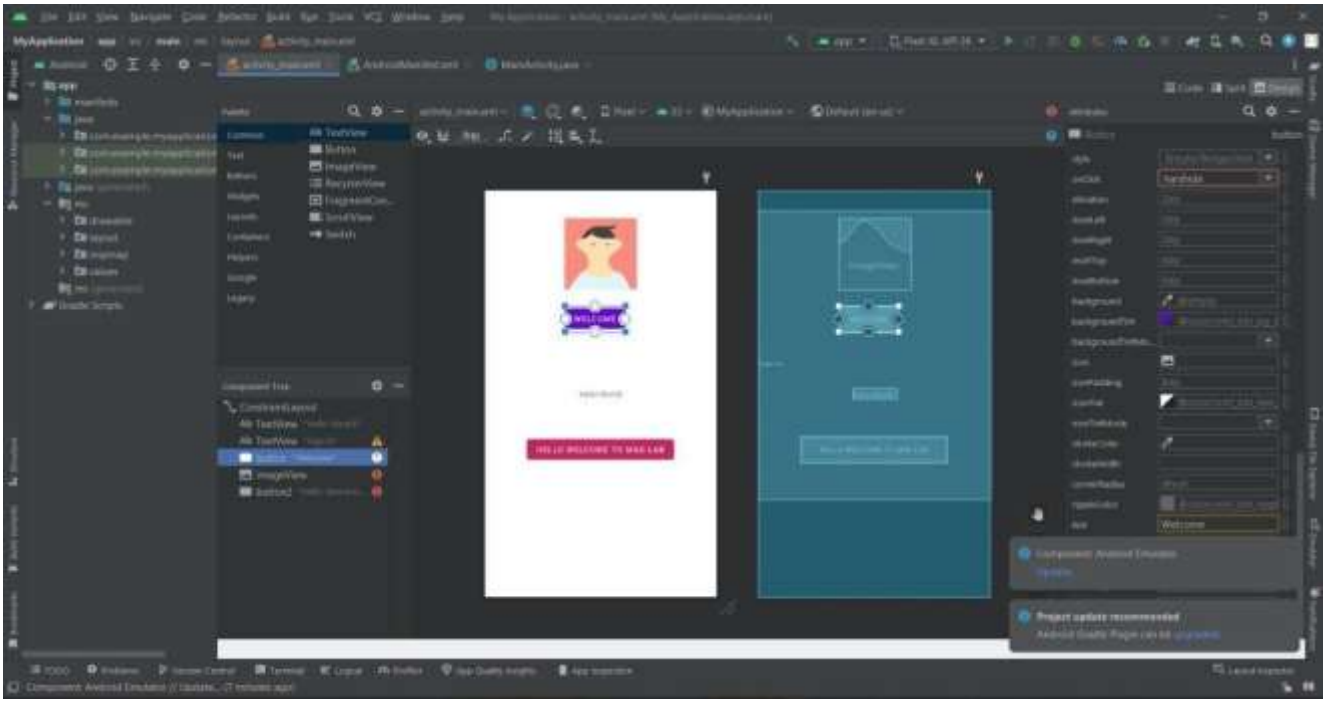
3. Code and Output:



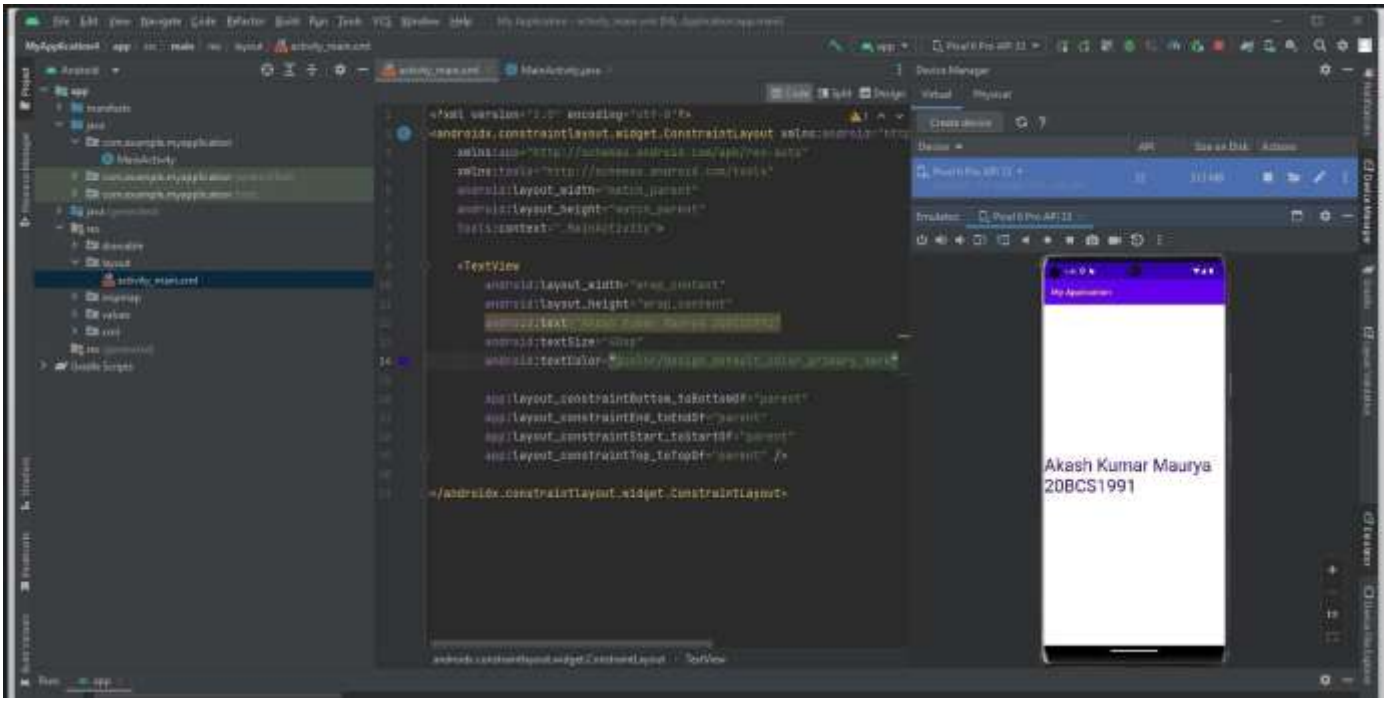





WELCOME
HELLO WELCOME TO MAD LAB


WELCOME
HELLO WELCOME TO MAD LAB

DEPARTMENT OF



Experiment - 3

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: CSE **Section/Group: 719 - B**
Semester: 6th

Subject Name: MOBILE APPLICATION DEVELOPMENT LAB

1. Aim/Overview of the practical:

Create Application by Using Widgets

Widgets are the micro-version of the application that consists of some functionality of the application that is displayed only on the **Home Screens** or the **Lock Screen**. For example, we see **Weather**, **Time**, **Google Search Bars** on the Home Screen, and **Face Lock**, **FingerprintLock** on the Lock Screen, which are some of the Widgets available on the device. Widgets come along with the Application when you install it or download it from the Web. Generally, phones come with a manufacturing configuration, but such elements can be adjusted by a user later in time. In this article, we prove how one can implement a basic widget for an Android App.

Steps for Creating a Basic Widget

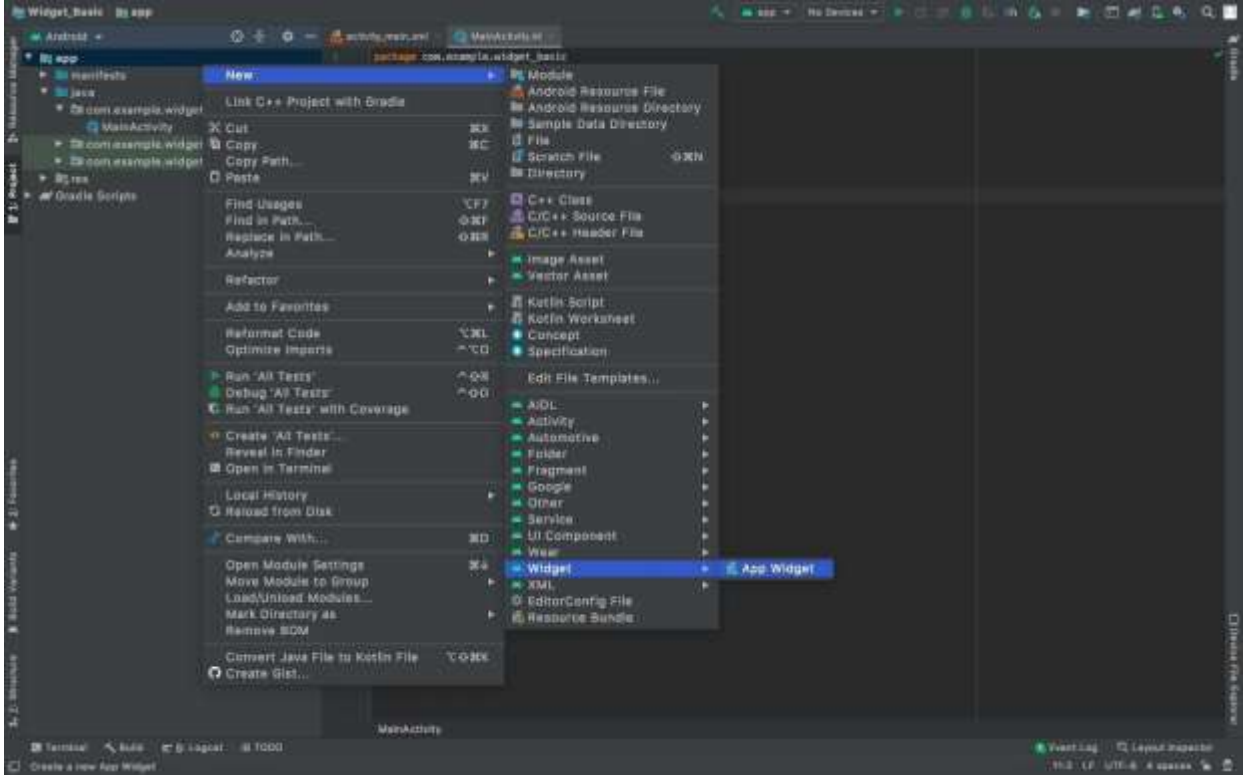
Step 1: Create a New

Project

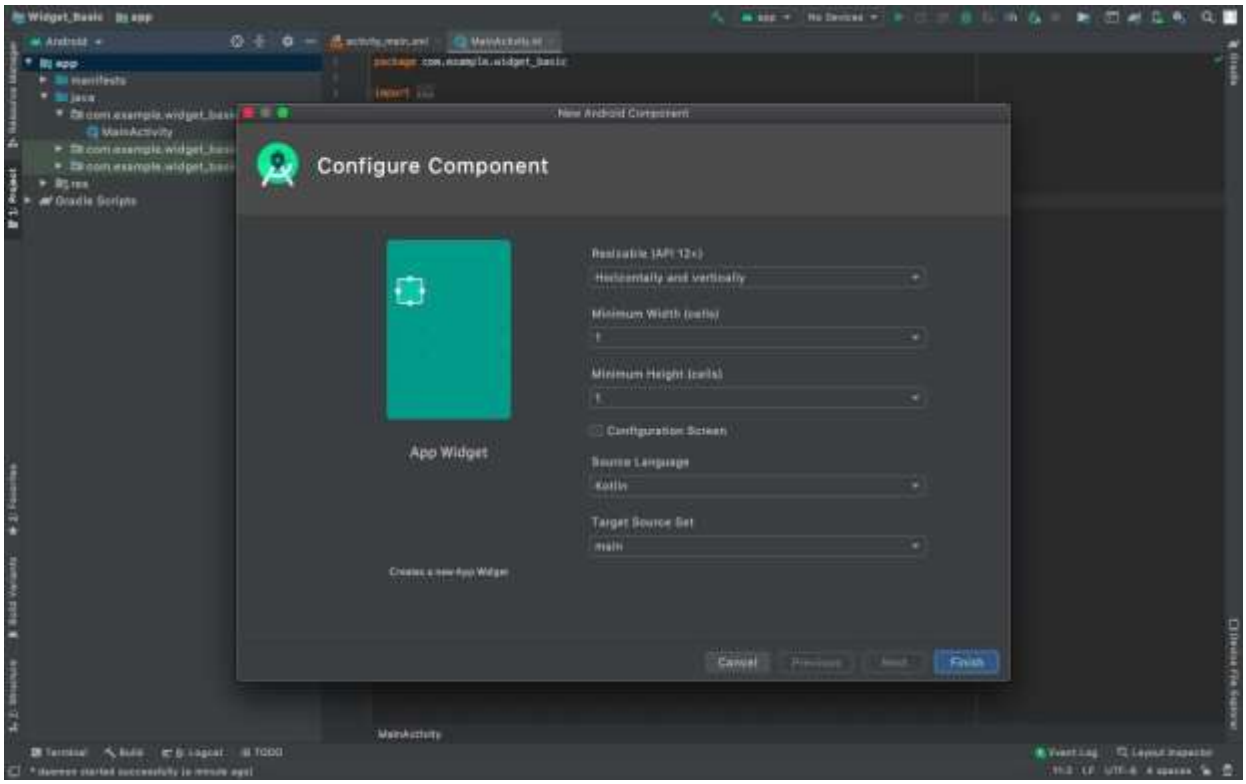
To create a new project in Android Studio please refer to **How to Create/Start a New Project in Android Studio**. We are implementing it for both **Java** and **Kotlin** languages.

Step 2: Add the App Widget to the Project

Right-Click on the **app**, move the cursor to **new**, find the "**Widget**" option at the end, select it.



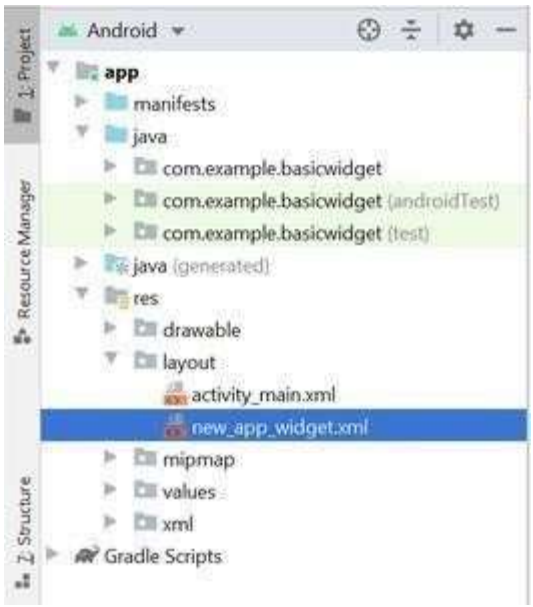
Specify the required properties for the widget such as **min. width** and **height**, config file and preferred language, etc., and go ahead. Files are automatically generated.



Step 3: Install and Run the Code

- Install and run the code on Android Virtual Device (AVD) or a personal device.
- Open the widget section of the phone, lookup for a widget with the Application name, select it, bring it to the home screen.
- Try changing the dimensions and we are done!

CODE IN NEW_APP_WIDGET.XML



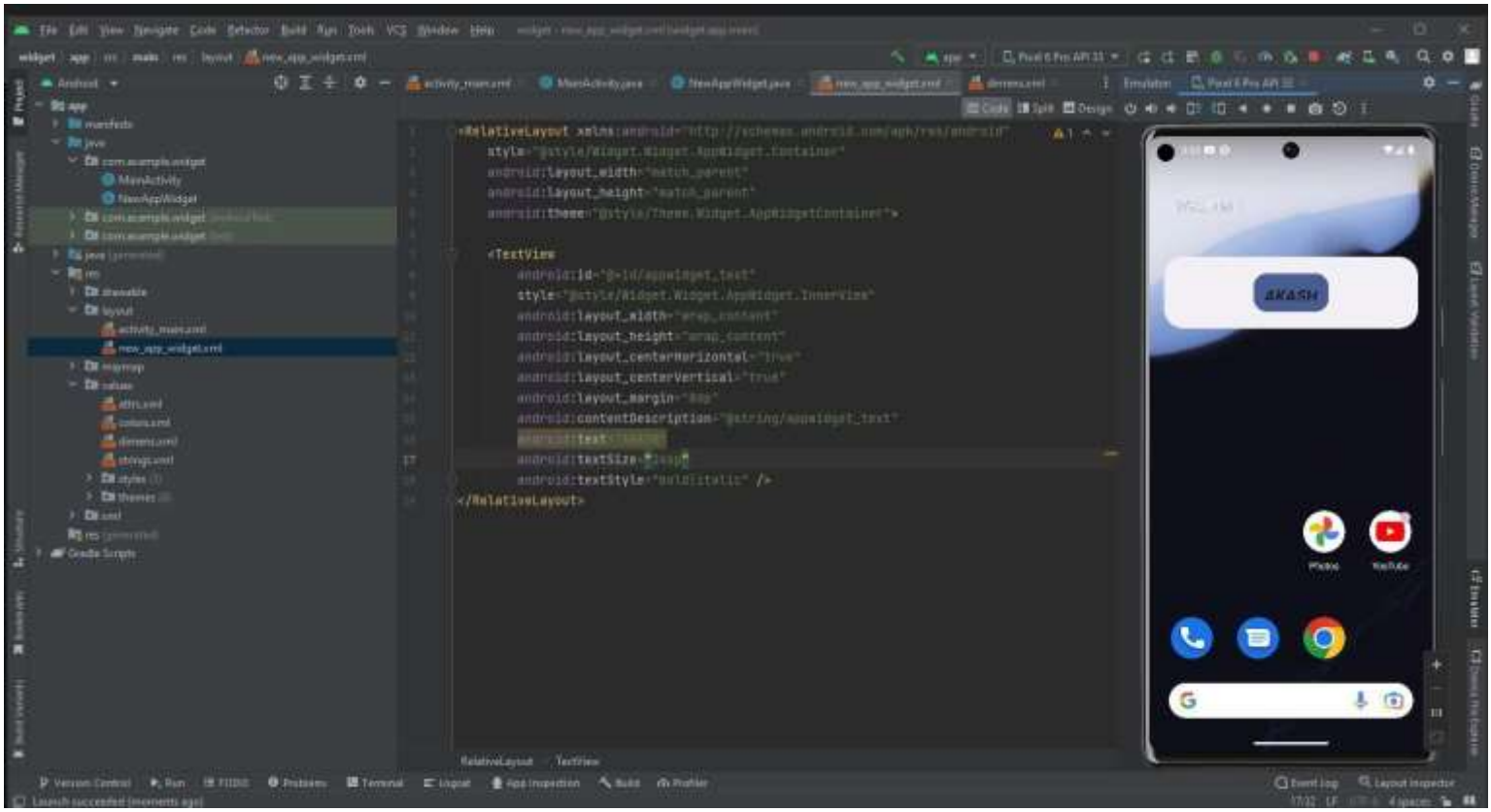

```
import android.appwidget.AppWidgetManager

import android.appwidget.AppWidgetProvider

import android.content.Context

import android.widget.RemoteViews


// Implementation of App Widget functionality.

class NewAppWidget : AppWidgetProvider() {

override fun onUpdate( context: Context,

appWidgetManager: AppWidgetManager,

appWidgetIds: IntArray

) {
```

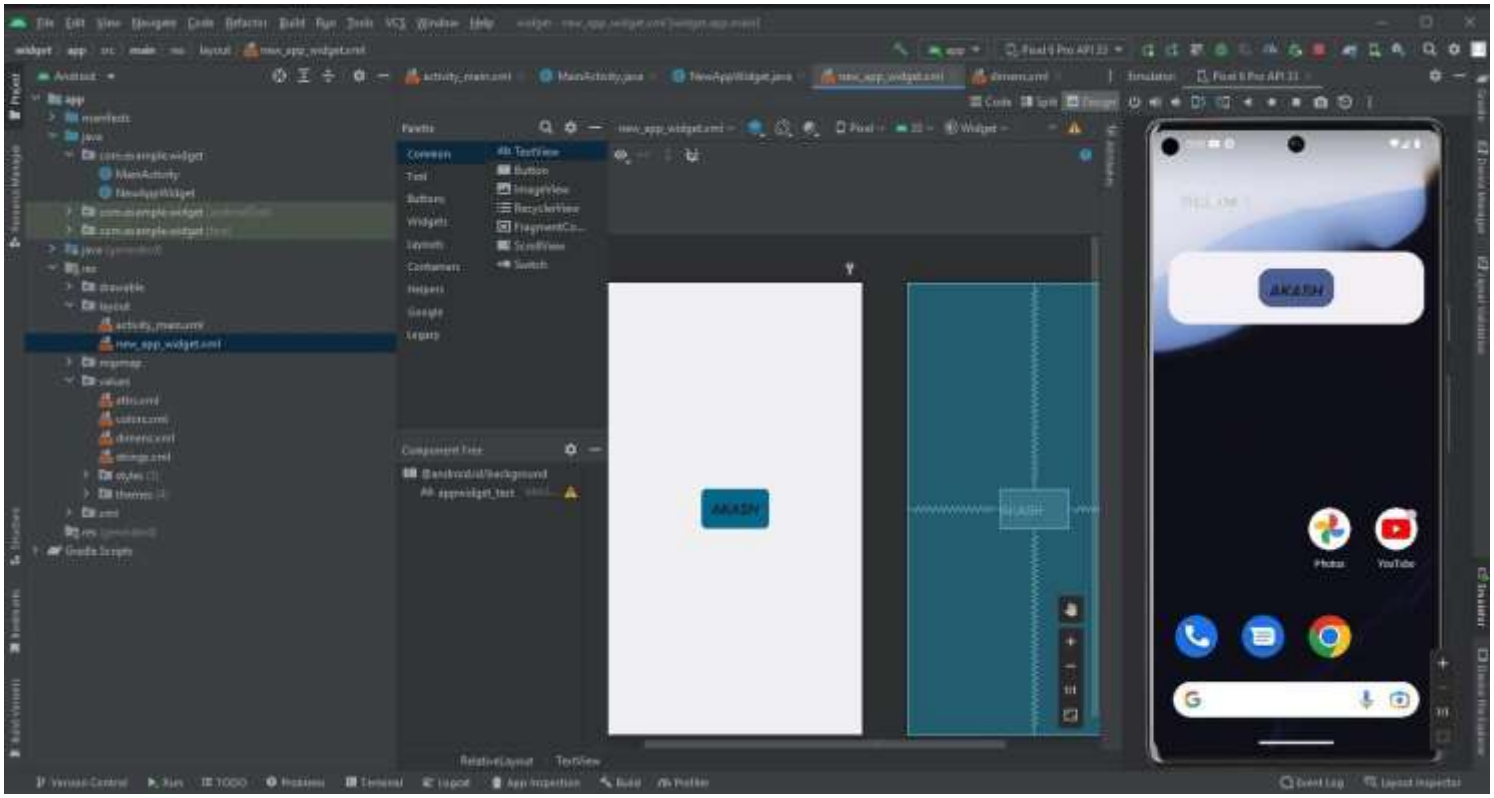
```
            // There may be multiple widgets active, so update all of them

            for (appWidgetId in appWidgetIds) { updateAppWidget(context,

            appWidgetManager, appWidgetId)

}
}

        // Enter relevant functionality for
// when the first widget is created
override fun onEnabled(context: Context) {

        }

        // Enter relevant functionality for
// when the last widget is disabled
override fun onDisabled(context: Context) {

        }

    }
internal fun updateAppWidget(
    context: Context,
    appWidgetManager:
    AppWidgetManager,
    appWidgetId: Int
    ) {
val widgetText = context.getString(R.string.appwidget_text)
        // Construct the RemoteViews object

        val views = RemoteViews(context.packageName, R.layout.new_app_widget)

        views.setTextViewText(R.id.appwidget_text, widgetText)


        // Instruct the widget manager to update the widget

        appWidgetManager.updateAppWidget(appWidgetId, views)

    }
```

XML CODE

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#09C"
    android:padding="@dimen/widget_margin">

    <TextView android:id="@+id/appwidget_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_margin="8dp"
        android:background="#09C"
        android:contentDescription="@string/appwi
        dget_text"
        android:text="AKASH"
        android:textColor="#ffffff" android:textSize="24sp"
        android:textStyle="bold|italic" />
</RelativeLayout>
```

WIDGET OUTPUT





Experiment -2.1

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 23.03.2023**
Subject Name: MAD LAB **Subject Code: 20CSP-356**

1. AIM: Create an application that takes the name from a text box and shows a hello message along with the name entered in text box when the user clicks the OK button.

2. Objective:

Understanding of the interactions between user interface and underlying application infrastructure.Creating the Application by using Text Edit control.

Steps:

1. Open Android Studio and create a new project with an Empty Activity.
2. Open the activity_main.xml file and add a TextView, an EditText, and a Button to the layout.
3. Set the properties of the TextView and EditText, such as ID, text, hint, and layout parameters.
4. Set the text property of the Button to "OK" or any other appropriate label.
5. Open the MainActivity.java file and define the Button click listener.
6. In the Button click listener, retrieve the value entered in the EditText.
7. Concatenate the retrieved value with the hello message, for example, "Hello, [name]!".
8. Set the text property of the TextView to the concatenated message.
9. Test the application by running it and entering different names in the EditText.

3. Program:

MainActivity.java package

com.example.messageintent; import

androidx.appcompat.app.AppCompatActivity;

```
import android.content.Intent; import
android.content.pm.PackageManager;
import        android.net.Uri;        import
android.os.Bundle;                    import
android.view.View;                    import
android.widget.Button;                import
android.widget.EditText;              import
android.widget.Toast;                 import
com.hbb20.CountryCodePicker;

public      class      MainActivity      extends
   AppCompatActivity {CountryCodePicker
   countryCodePicker;                    EditText
   phone,message;Button sendbtn;
   String messagestr,phonestr="";

   @Override protected void onCreate(Bundle
   savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      countryCodePicker=findViewById(R.id.countyCodePi
      c ker);phone = findViewById(R.id.phoneNo); message
      = findViewById(R.id.message); sendbtn =
      findViewById(R.id.sendbtn);


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

            messagestr = message.getText().toString(); phonestr=phone.getText().toString();

            if(!messagestr.isEmpty()     &&     !     phonestr.isEmpty()){
               countryCodePicker.registerCarrierNumberEditText(phone
               ); phonestr = countryCodePicker.getFullNumber();
if(isWhatappInstalled()){

                  Intent i = new Intent(Intent.ACTION_VIEW,
 Uri.parse("https://api.whatsapp.com/send?phone="+phonestr+"&text="+messagestr
                  )); startActivity(i); message.setText(""); phone.setText("");
               }else{
                  Toast.makeText(MainActivity.this,"whatsapp is not
 installed",Toast.LENGTH_SHORT).show();
               }

            }else{
               Toast.makeText(MainActivity.this,"please enter the Phone number and enter the
 text",Toast.LENGTH_LONG).show();
```

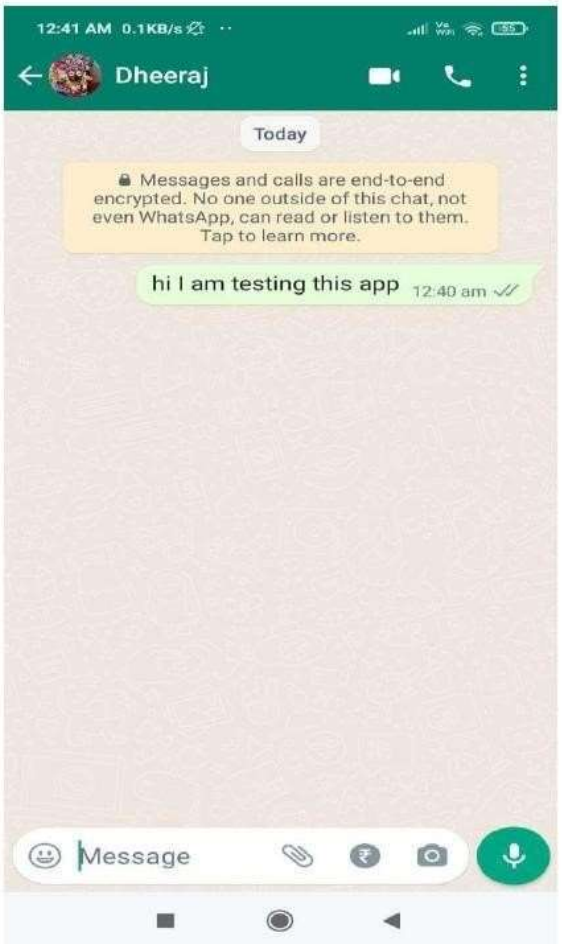
```
        }
    });

}
private boolean isWhatappInstalled(){
    PackageManager packageManager = getPackageManager();
    boolean whatsappInstalled;

    try{
        packageManager.getPackageInfo("com.whatsapp",PackageManager.GET_ACTIVITIES);
        whatsappInstalled = true;
    }catch(PackageManager.NameNotFoundException e){


        whatsappInstalled=false;
    }


    return whatsappInstalled;


}
```

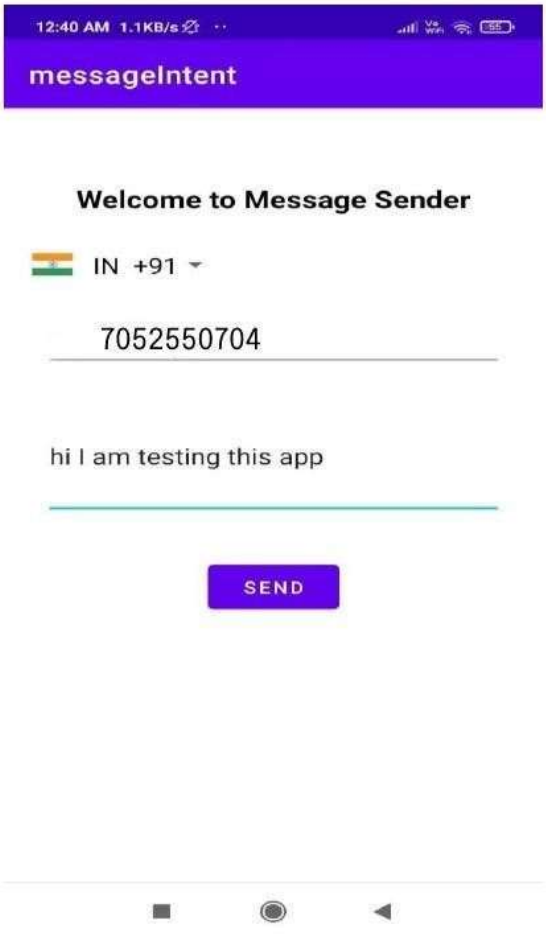
activity_main.xml

```
<?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:orientation="vertical"
    android:padding="10dp" tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="50dp"
        android:text="Welcome to Message Sender"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold" />

    <com.hbb20.CountryCodePicker
        android:id="@+id/countyCodePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp" />

    <EditText       android:id="@+id/phoneNo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="16dp
        "  android:layout_marginVertical="20dp"
        android:ems="10"     android:hint="Enter
        Phone" android:inputType="phone" />

    <EditText android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_marginHorizontal="16dp
        " android:ems="10"
        android:hint="Message"
        android:inputType="textMultiLine" />

    <Button android:id="@+id/sendbtn"
        android:layout_width="wrap_content
        "
        android:layout_height="wrap_content
```

```
" android:layout_gravity="center"
android:layout_marginTop="32dp"
android:text="send" />


</LinearLayout>
```

Output:





Experiment -2.2

Student Name: Vikash Yadav **UID: 21BCS8093**
Branch: CSE **Section/Group: 719/B**
Semester: 6th **Date of Performance: 06.04.2023**
Subject Name: MAD LAB **Subject Code: 20CSP-356**

1. AIM: Create an Android App using various controls such TextEdit, CheckBox, RadioButton, RadioGroup, etc.

2. Objective:

Checkbox belongs to android.widget.CheckBox class. Android Checkbox class is the subclass of Compound Button class. It is used in a place where user can select one or more than choices from a given list of choices. For example, selecting hobbies.

3. Steps/Program:

Create an Android App using Checkbox.

CheckBox belongs to android.widget.CheckBox class. Android CheckBox class is the subclass of CompoundButton class. It is generally used in a place where user can select one or more than choices from a given list of choices. For example, selecting hobbies.

public class CheckBox *extends* CompoundButton

it has two states – **checked** or **unchecked**.

Methods of CheckBox class

• ***public boolean isChecked():*** If CheckBox is in checked state then return true otherwise false. • ***public void setChecked(boolean status):*** It changes the state of the CheckBox.
Below is the code for an example where the user chooses its hobbies from the given list containing Painting, Reading, Singing and Cooking with the help of CheckBox.

1. Open Android Studio and create a new project by selecting "Start a new Android Studio project" from the welcome screen.
2. Enter your app name and choose your project location. Then, select the minimum SDK you want to target and click "Next".
3. Choose the activity you want to add to your project. For this example, select "Empty Activity" and click "Next".

4. In the "Configure your project" screen, you can change the default values or leave them as they are. Click "Finish" to create the project.

5. In the project navigator, open the layout file "activity_main.xml". This is where you will create the UI for your app.

6. Drag and drop the controls you want to use from the "Palette" onto the design view of the layout file. For this example, let's add a TextEdit, a CheckBox, a RadioButton, and a RadioGroup.

7. Configure the properties of each control by selecting it in the design view and editing its attributes in the "Properties" panel. For example, you can set the text for the TextEdit control, or the text and ID for the RadioButton control.

8. You can also add event handlers to the controls by selecting them in the design view and clicking on the "Events" tab in the "Properties" panel. For example, you can add an "onClick" handler to the CheckBox control.

9. Save the layout file and switch to the Java code for the activity by opening the file "MainActivity.java".

10. In the onCreate method, add code to find the controls you added to the layout file by calling the findViewById method and passing in the ID of each control.

11. You can also add event handlers for the controls in the Java code by defining methods that match the signature of the event handler and adding annotations to link them to the controls.

12. Build and run the app to see the controls in action. You can do this by clicking on the green "Run" button in the toolbar and selecting your device or emulator.

MainActivity.java

```
package com.example.exp5; import
androidx.appcompat.app.AppCompatActivity;
import android.view.View; import
android.os.Bundle; import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   CheckBox ch, ch1, ch2, ch3;

   @Override protected void onCreate(Bundle
   savedInstanceState) {
```

```
        super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
//              finding          checkbox
        ch=(CheckBox)findViewById(R.id.checkBox);
        ch1=(CheckBox)findViewById(R.id.checkBox2);
        ch2=(CheckBox)findViewById(R.id.checkBox3);
        ch3=(CheckBox)findViewById(R.id.checkBox4);



            btn press
//      Button btn = (Button) findViewById(R.id.button);

        btn.setOnClickListener( new
              View.OnClickListener() {
                 @Override
                 public void onClick(View v) {
                    String                    msg="";
                    if(ch.isChecked())   msg   =
                    msg + " Painting ";

                    if(ch1.isChecked()) msg =
                       msg + " Reading ";


                    if(ch2.isChecked())
                       msg = msg + " Singing ";


                    if(ch3.isChecked())
                       msg = msg + " Cooking ";
Toast.makeText(getApplicationContext(), msg.toString() + "are selected",
Toast.LENGTH_LONG).show();
                 }
              }
        );
    }
}
```

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"
    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="Select Hobbies"
        app:layout_constraintBottom_toTopOf="@+id/checkBox"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
        <CheckBox android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:text="Painting"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        />

    <CheckBox android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp" android:text="Reading"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox"
        />

    <CheckBox android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:text="Singing" app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkBox2"
    app:layout_constraintVertical_bias="0.142" />



<CheckBox android:id="@+id/checkBox4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cooking"
    app:layout_constraintBottom_toTopOf="@+id/button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/checkBox3
    " app:layout_constraintVertical_bias="0.109" />

<Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="400dp"
    android:text="Submit"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.503"
    app:layout_constraintStart_toStartOf="parent" />

<RadioGroup android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button
    " >

    <RadioButton android:id="@+id/radioButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="text1" />
```

```
        <RadioButton android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="text2" />

        <RadioButton android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="text3" />
    </RadioGroup>

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

Output:

Experiment 2.3

Student Name: Vikash Yadav

Branch: BE-CSE
Semester: 6

UID: 21BCS8093
Section/Group: 20BCS_DM-719(B)
Date of Performance: 27-04-2023
Subject Code: 20CSP-356

AIM:

Create SMS sending android application using XML and JAVA.

OBJECTIVE:

To design an android application by Using Various elements and send a SMS using the app when click on send button.

HARDWARE/SOFTWARE REQUIREMENT:

- Java JDK 5 or later version
- Java Runtime Environnent (JRE) 6 Android Studio
- Microsoft Windows 10
- 4 GB RAM minimum , 8 GB (recommended)
- 10GB of available disk space minimum, 20 GB recommended 1280 x 800 minimum screen resolution

INTRODUCTION:

SMS stands for Short Message Service and is another name for a text message. An SMS is generally sent from one mobile device to another over the cellular network. SMS is a textonly standard first formalized in 1985 in the Global System for Mobile Communications (GSM) standards.

STEPS TO CREATE APPLICATION:

Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. We are implementing it for both Java and Kotlin languages.

Add the Empty activity to the Project

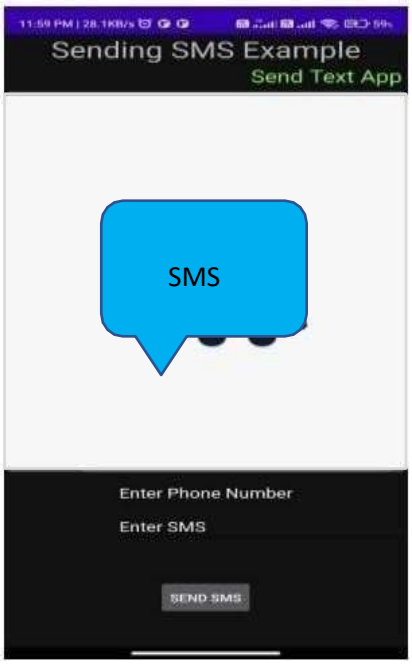
Right-Click on the app, move the cursor to new, find the "Empty Activity" option at the end, select it and proceed. Files are automatically generated.

Install and Run the Code

Install and run the code on Android Virtual Device (AVD) or a personal device.

Open the Apps of the phone, lookup for a new App with the Application name. ▪ Run the App

DESIGN OF THE APP :



CODE:

<u>MainActivity.java</u>

```
package com.example.myapplication;

import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends Activity { private static final
    int
    MY_PERMISSIONS_REQUEST_SEND_SMS =0 ;
```

```
Button sendBtn;
EditText txtphoneNo;
EditText txtMessage; String
phoneNo;
String message;

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

     sendBtn = (Button) findViewById(R.id.btnSendSMS);
    txtphoneNo = (EditText) findViewById(R.id.editText);
    txtMessage = (EditText) findViewById(R.id.editText2);

     sendBtn.setOnClickListener(new
    View.OnClickListener() { public void
    onClick(View view) { sendSMSMessage();
        }
  }); } protected void
sendSMSMessage() { phoneNo =
  txtphoneNo.getText().toString(); message
  =
  txtMessage.getText().toString();

if (ContextCompat.checkSelfPermission(this,
    Manifest.permission.SEND_SMS)
```

```
        != PackageManager.PERMISSION_GRANTED)
    { if
    (ActivityCompat.shouldShowRequestPermissionRat
    ionale(this, Manifest.permission.SEND_SMS)) {
     } else {


            ActivityCompat.requestPermissions( this,
            new
            String[]{Manifest.permission.SEND _SMS},
            MY_PERMISSIONS_REQUEST_S
            END_SMS);
     }



}
}
     @Override public void onRequestPermissionsResult(int
    requestCode,String permissions[], int[] grantResults) {
switch (requestCode) { case
    MY_PERMISSIONS_REQUEST_SE
    ND_SMS: { if (grantResults.length > 0
            && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNo, null, message,
            null, null);
             Toast.makeText(getApplicationContext(), "SMS sent.",
            Toast.LENGTH_LONG).show();
} else {
Toast.makeText(getApplicationContext(),
    "SMS faild, please try again.",
Toast.LENGTH_LONG).show(); return;
     }
```

```
        }
      }
    }
  }
```

AndroidManifest.xml

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

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

    <application android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extrac tion_rules"
        android:fullBackupContent="@xml/backup_rule
        s" android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication" tools:targetApi="31">
        <activity
          android:name=".MainActivi
          ty" android:exported="true">
          <intent-filter>
              <action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
          </activity>
        </application>
```

```
</manifest>
```

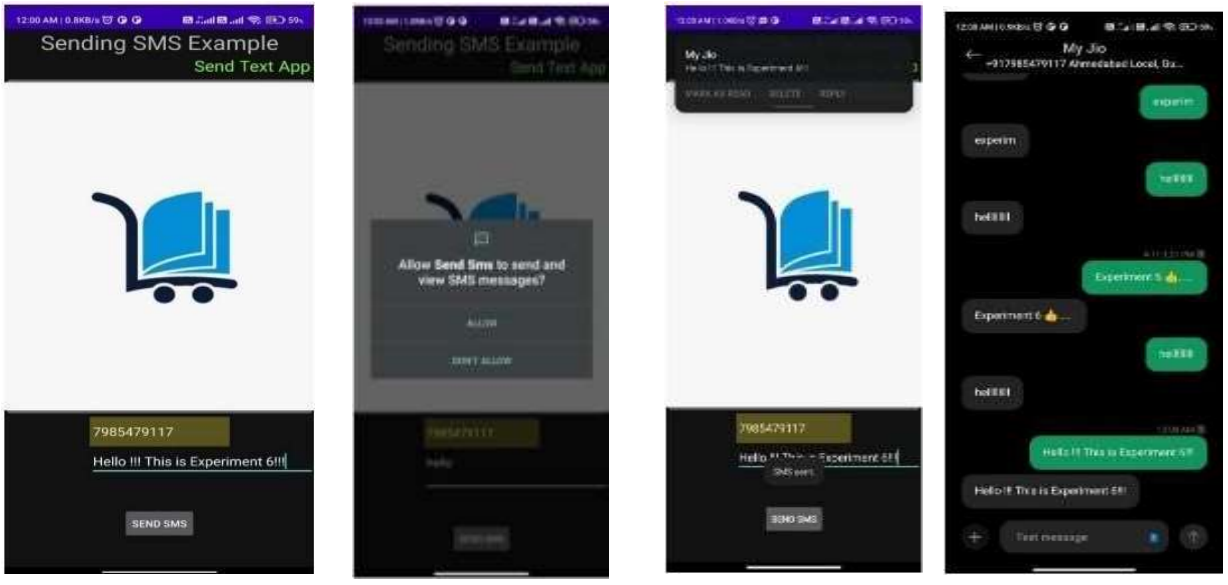
ActivityMain.xml

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

        <TextView android:id="@+id/textView1"
            android:layout_width="wrap_ content"
            android:layout_height="wrap_
            content" android:text="Sending SMS
            Example"
            android:layout_alignParentTo p="true"
            android:layout_centerHorizont
            al="true" android:textSize="30dp" />

        <TextView android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Send Text App "
            android:textColor="#ff87ff09" android:textSize="23dp"
            android:layout_below="@+id/textView
            1"
            android:layout_alignRight="@+id/ima geButton"
            android:layout_alignEnd="@+id/image
            Button" />

        <ImageButton android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageButton"
```

```
    android:src="@drawable/abc"
    android:layout_below="@+id/textView
    2"
    android:layout_centerHorizontal="true
    " />

<EditText android:layout_width="wrap_conten
  t"
  android:layout_height="wrap_conte nt"
  android:id="@+id/editText"
  android:hint="Enter Phone Number"
  android:phoneNumber="true"
  android:textColorHint="@color/abc_pr
  imary_text_material_dark"
  android:layout_below="@+id/imageButton"
  android:layout_centerHorizontal="true" />

<EditText
  android:layout_width="wrap_conten t"
  android:layout_height="wrap_conte nt"
  android:id="@+id/editText2"
  android:layout_below="@+id/editT
  ext"
  android:layout_alignLeft="@+id/edi tText"
  android:layout_alignStart="@+id/ed itText"
  android:textColorHint="@color/abc_primary_text_material_dark"
  android:layout_alignRight="@+id/imageButton"
  android:layout_alignEnd="@+id/imageButton"
   android:hint="Enter SMS" />

<Button
  android:layout_width="wrap_conte nt"
```

```
        android:layout_height="wrap_cont ent"
        android:text="Send Sms"
        android:id="@+id/btnSendSMS"
        android:layout_below="@+id/edit
        Text2"
        android:layout_centerHorizontal="t
        rue"
        android:layout_marginTop="48dp"
        />

</RelativeLayout>
```

FINAL OUTPUT:



<u>Learning outcomes (What I have learnt)</u>:

- To design an android application using different types of elements from palette in android studio.
- Learned about running application on android studio.

Experiment 2.4

Student Name: Vikash Yadav

Branch: BE-CSE
Semester: 6

UID: 21BCS8093
Section/Group: 20BCS_DM-719(B)
Date of Performance: 27-04-2023
Subject Code: 20CSP-356

AIM:

Creating the Application Choosing Options Radio Group

HARDWARE/SOFTWARE REQUIREMENT:

- Java JDK 5 or later version
- Java Runtime Environment (JRE) 6 Android Studio
- Microsoft Windows 10
- 4 GB RAM minimum , 8 GB (recommended)
- 10GB of available disk space minimum, 20 GB recommended 1280 x 800 minimum screen resolution

INTRODUCTION:

Android radio button is a widget that can have more than one option to choose from. The user can choose only one option at a time. Each option here refers to a radio button and all the options for the topic are together referred to as Radio Group. Hence, Radio Buttons are used inside a RadioGroup.

STEPS TO CREATE APPLICATION:

Step 1: First create a new Android Application. This will create an XML file "activity_main.xml" and a Java File "MainActivity.Java". Please refer the pre-requisites to learn more about this step.

Step 2: Open the "activity_main.xml" file and add the following widgets in a Relative Layout:

- A TextView to display the question message
- A RadioGroup to hold the option Radio Buttons which are the possible answers
- 4 RadioButtons to hold an answer each.
- A Submit and a Clear button to store the response.

Also, Assign the ID to each of the components along with other attributes as shown in the given image and the code below. The assigned ID on a component helps that component to be easily found and used in the Java files.

Syntax:

android:id="@+id/id_name" Here

the given IDs are as follows:

- RadioGroup: groupradio
- RadioButton1: radia_id1
- RadioButton2: radia_id2
- RadioButton3: radia_id3
- RadioButton4: radia_id4
- Submit Button: submit
- Clear Button: clear

This will make the UI of the Application.

Step 3: Now, after the UI, this step will create the Backend of Application. For this, open the "MainActivity.java" file and instantiate the components made in the XML file (Radio Group, Text View, Clear, and Submit Button) using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

Syntax: General

ComponentType object = (ComponentType)findViewById(R.id.IdOfTheComponent); Syntax: Components used

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

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

RadioGroup radioGroup = (RadioGroup)findViewById(R.id.groupradio);

Step 4: This step involves setting up the operations on the RadioGroup, RadioButtons, and the Submit and Clear Buttons.
These operations are as follows:

· Unset all the Radio Buttons initially as the default value. This is done by the following command:

radioGroup.clearCheck();

· Add the Listener on the RadioGroup. This will help to know whenever the user clicks on any Radio Button, and the further operation will be performed. The listener can be added as follows:

radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){} ·

Define the operations to be done when a radio button is clicked. This involves getting the specific radio button that has been clicked, using its id. Then this radio button gets set and the rest of the radio button is reset.

· Add the listener on Submit button and clear button. This will be used to check when the user clicks on the button. This is done as follows:

submit.setOnClickListener(new View.OnClickListener() {} clear.setOnClickListener(new View.OnClickListener() {}

· In the Submit Button Listener, set the operations to be performed. This involves displaying the marked answer in the form of Toast.

· In the Clear Button Listener, set the operations to be performed. This involves resetting all the radio buttons.

Step5: Now run the app and operate as follows:

· When the app is opened, it displays a question with 4 answers and a clear and submit button.
· When any answer is clicked, that radio button gets set.
· Clicking on any other radio button sets that one and resets the others.
· Clicking on Submit button displays the currently marked answer as a Toast.
· Clicking on Clear button resets all the radio buttons to their default state.

CODE:

✞ **MainActivity.java**

```
package com.example.sevenapp;
```

```
import      android.os.Bundle; import
android.widget.Button; import
android.widget.RadioButton; import
android.widget.RadioGroup; import
android.widget.Toast;

import      androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

      Button submitButton = findViewById(R.id.button);
      RadioGroup radioGroup = findViewById(R.id.radioGroup);

      submitButton.setOnClickListener(view -> { int selectedId
         = radioGroup.getCheckedRadioButtonId(); if (selectedId
         != -1) {
            RadioButton selectedRadioButton = findViewById(selectedId);
            String selectedText = selectedRadioButton.getText().toString();
            Toast.makeText(getApplicationContext(), "Selected: " + selectedText,
Toast.LENGTH_SHORT).show();
         } else {
            Toast.makeText(getApplicationContext(), "Please select an option",
Toast.LENGTH_SHORT).show();
         }
      });
   } }
```

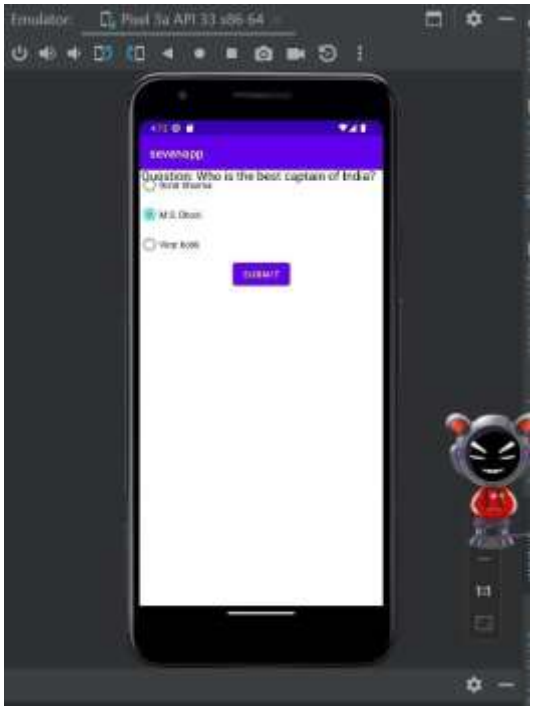
✞ **ActivityMain.xml**

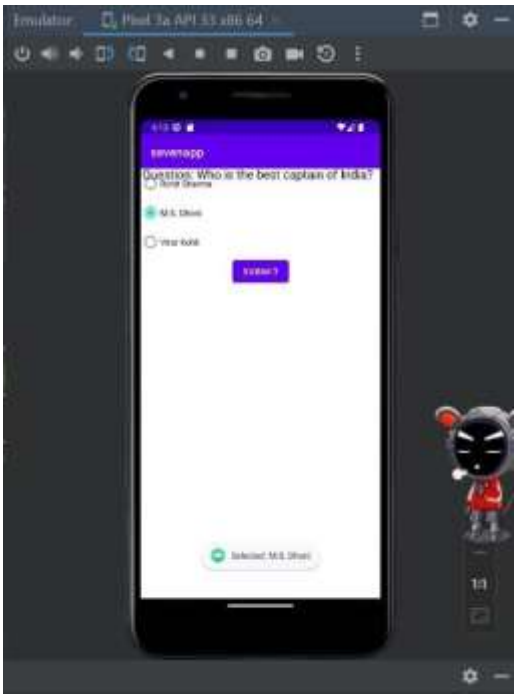
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

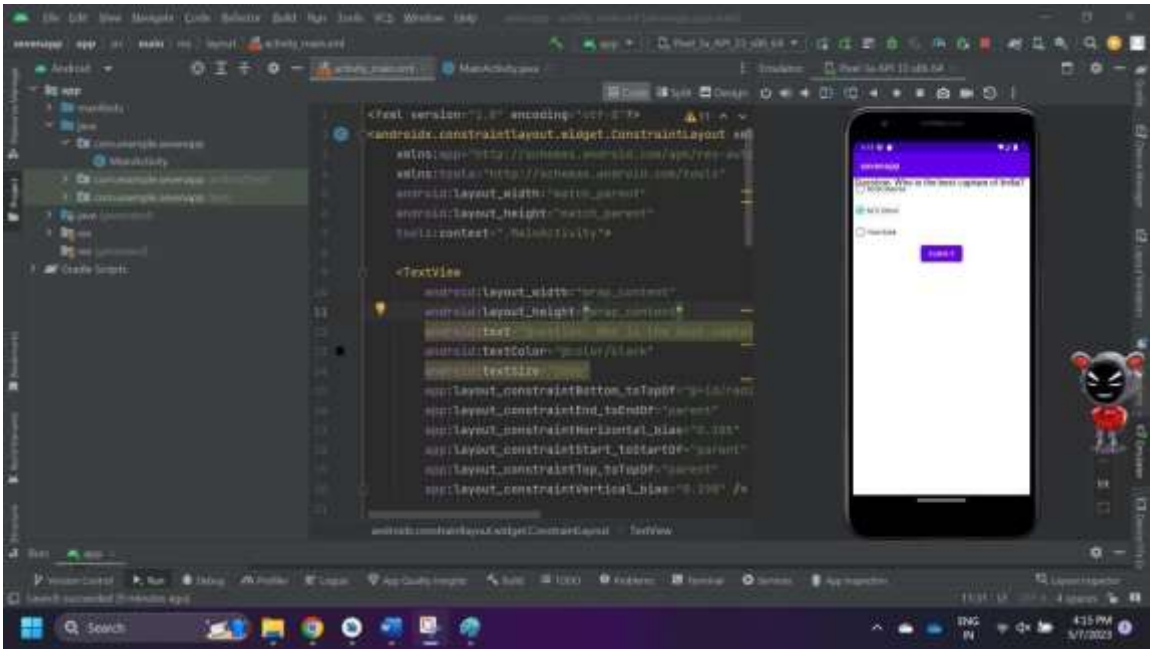
```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Question: Who is the best captain of India?"
    android:textColor="@color/black"
    android:textSize="20dp"
    app:layout_constraintBottom_toTopOf="@+id/radioGroup"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.185"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.198" />

<RadioGroup android:id="@+id/radioGroup"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    tools:ignore="DuplicateIds,MissingConstraints">

    <RadioButton android:id="@+id/radioButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Rohit Sharma" />

    <RadioButton android:id="@+id/radioButton2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="M.S. Dhoni" />

    <RadioButton android:id="@+id/radioButton3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Virat Kohli" />
```

```
</RadioGroup>

<Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content
    " android:text="Submit"
    app:layout_constraintTop_toBottomOf="@id/radioGroup"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintVertical_bias="0.0"
    tools:ignore="HardcodedText" />


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

FINAL OUTPUT:







<u>Learning outcomes (What I have learnt)</u>:

- To design an android application using different types of elements from palette in android studio.
- Learnt about running application on android studio.

Experiment 3.1

Student Name: Vikash Yadav	**UID:** 21BCS8093
	Section/Group: 20BCS_DM-719(B)
	Date of Performance: 04-05-2023
Branch: BE-CSE	**Subject Code:** 20CSP-356
Semester: 6	

AIM: Create an Android App using fragments.

SOFTWARE USED:

- Android Studio
- Android Emulator Pixel 6 Pro API 31 **CODE:**

MAINACTIVITY.JAVA

```java
package com.example.experiment7;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;

2 usages
public class MainActivity extends AppCompatActivity {
    2 usages
    Button firstFragmentBtn, secondFragmentBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        firstFragmentBtn = findViewById(R.id.fragment1btn);
        secondFragmentBtn = findViewById(R.id.fragment2btn);

        firstFragmentBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                replaceFragment(new fragment1());
            }
        });
```

```java
        secondFragmentBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                replaceFragment(new fragment2());
            }
        });
    }

    2 usages
    private void replaceFragment(Fragment fragment) {

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frameLayout,fragment);
        fragmentTransaction.commit();
    }
}
```

ACTIVITY.xml

```xml
<?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">

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

    <Button
        android:id="@+id/fragment1btn"
        android:layout_width="150dp"
        android:layout_height="60dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"

        android:layout_marginStart="20dp"
        android:layout_marginRight="30dp"
        android:layout_marginBottom="30dp"
        android:backgroundTint="@color/white"
        android:text="Fragment 1"
        android:textColor="@color/black" />
```

```
    <Button
        android:id="@+id/fragment2btn"
        android:layout_width="150dp"
        android:layout_height="60dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="30dp"
        android:layout_marginEnd="20dp"
        android:layout_marginRight="52dp"
        android:layout_marginBottom="30dp"
        android:backgroundTint="@color/white"
        android:text="Fragment 2"
        android:textColor="@color/black" />

</RelativeLayout>
```

FRAGMENT1.java

```java
package com.example.experiment7;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
2 usages
public class fragment1 extends Fragment {

    2 usages
    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, attachToRoot: false);
        return view;
    }
}
```

FRAGMENT1.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_200"
    tools:context=".fragment2">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:textSize="30dp"
        android:layout_centerInParent="true"
        android:textColor="@color/black"/>

</RelativeLayout>
```

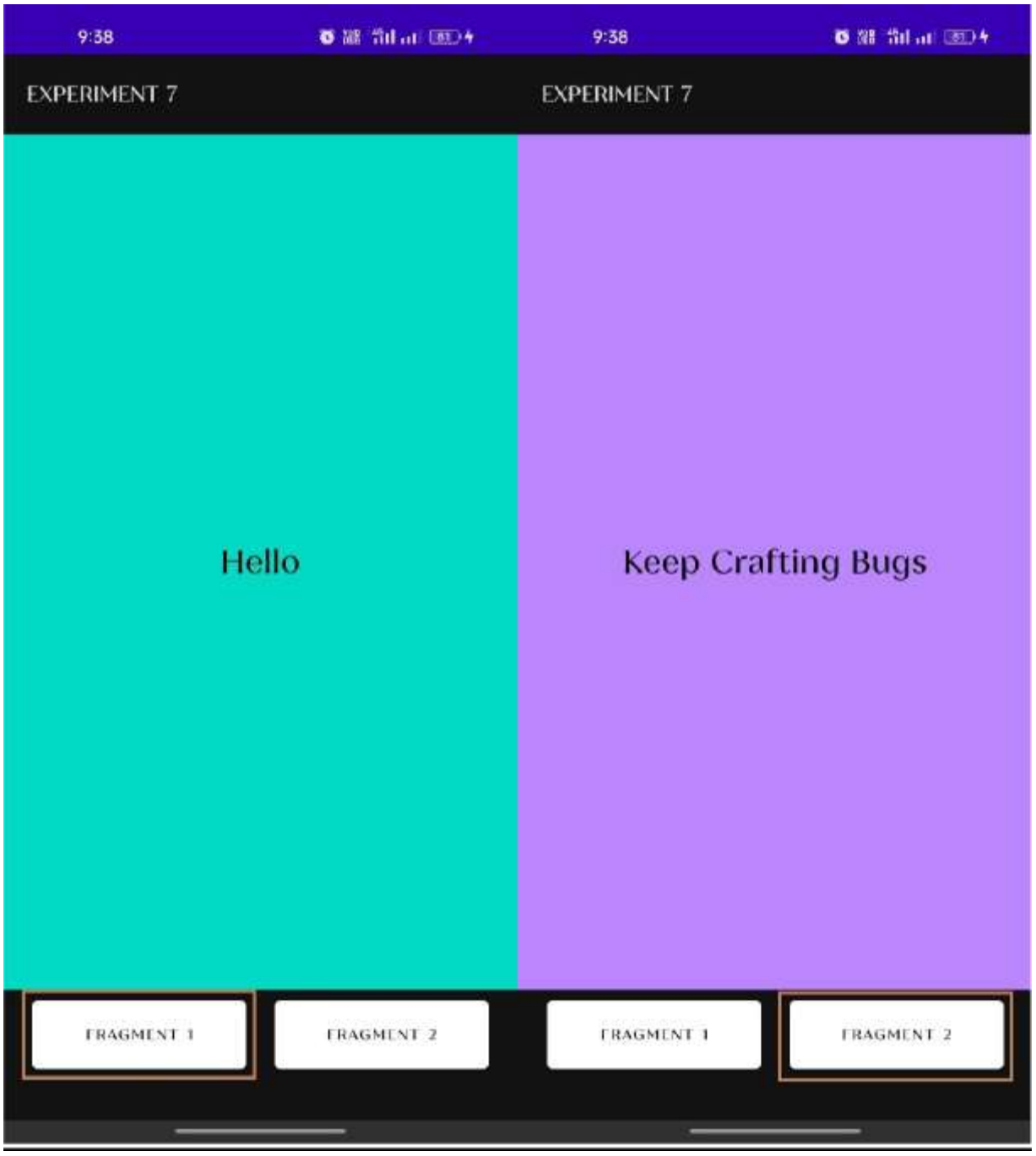
FRAGMENT2.java

```java
package com.example.experiment7;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
2 usages
public class fragment2 extends Fragment {

    2 usages
    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, attachToRoot: false);
        return view;
    }
}
```

FRAGMENT2.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/purple_200"
    tools:context=".fragment1">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Keep Crafting Bugs"
        android:textSize="30dp"
        android:layout_centerInParent="true"
        android:textColor="@color/black"/>

</RelativeLayout>
```

OUTPUT:



LEARNING OUTCOMES:

- Learnt about the use of fragment in android development
- Learnt about what are fragments and their purpose

Experiment 3.2

Student Name: Vikash Yadav	**UID:** 21BCS8093 **Section/Group:** 20BCS_DM-719(B) **Date of Performance:** 04-05-2023
Branch: BE-CSE **Semester:** 6	**Subject Code:** 20CSP-356

AIM:
Implement building blocks for android application using different layouts (such as linear, relative absolute)

OBJECTIVE:
To show the implementation of different types of layouts.

HARDWARE/SOFTWARE REQUIREMENT:

- Java JDK 5 or later version
- Java Runtime Environnent (JRE) 6 Android Studio
- Microsoft Windows 10
- 8 GB RAM minimum , 16 GB (recommended)
- 10GB of available disk space minimum, 20 GB recommended
- 1280 x 800 minimum screen resolution

INTRODUCTION:

- An Absolute layout allows you to specify the exact location i.e, X and Y coordinates of its children with respect to the origin at the top left corner of the layout.

- The relative layout is used to arrange the child views in a proper order which means arranging the child objects relative to each other.

- Android Linear layout is a view group subclass, used to provide child view elements one by one either in a particular direction either horizontally or vertically based on the orientation property.

STEPS TO CREATE APPLICATION:

- **Create a New Project**

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. We are implementing it for both Java and Kotlin languages.

- **Add the Empty activity to the Project**

Right-Click on the app, move the cursor to new, find the "Empty Activity" option at the end, select it and proceed. Files are automatically generated.

- **Install and Run the Code**

  - Install and Run the code on Android Virtual Device (AVD) or a personal device.
  - Open the Apps of the phone, lookup for a new App with the Application name.
  - Run the App

DESIGN OF THE APP : **[Absolute Layout]**

CODE:

- **Activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
    <AbsoluteLayout
        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="fill_parent"
        android:layout_height="fill_parent" tools:context=".MainActivity">

        <!--Setting up TextViews-->
        <TextView android:id="@+id/heading"
            android:textSize="25dp"
            android:textColor="@color/purple_200"
            android:textStyle="bold"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_x="420px"
            android:layout_y="500px" />

        <TextView android:id="@+id/subHeading"
            android:textSize="15dp"
            android:textColor="@color/teal_200"
```

```
        android:textStyle="italic"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="600px"
        android:layout_y="600px" />

    <Button android:text="Know More"
        android:textColor="@color/white"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content
        " android:layout_x="550px"
        android:layout_y="700px"/>

</AbsoluteLayout>
```

✞ **MainActivity.java**

```
package com.example.experiment8;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView heading, subHeading;

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

        // Referencing the TextViews heading = (TextView)
        findViewById(R.id.heading); subHeading = (TextView)
        findViewById(R.id.subHeading);

        // Setting text dynamically
```

```
    heading.setText("CRAFTING BUGS"); subHeading.setText("BY
    Vadik");
}}
```

DESIGN OF THE APP : **[Relative Layout]**

✞ **Activity_main.xml**

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

    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TOP LEFT"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"/>
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TOP RIGHT"
        android:layout_alignParentTop="true"
```

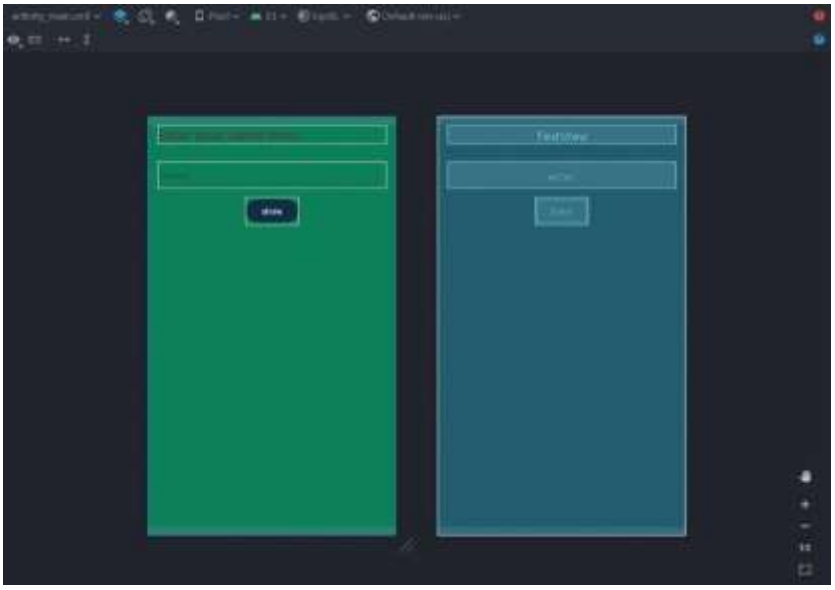
```
        android:layout_alignParentRight="true"/
        >
    <Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BOTTOM LEFT"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"/
        >
    <Button android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BOTTOM RIGHT"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"/
        >
    <Button android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_cont
        ent" android:text="MIDDLE "
        android:layout_centerVertical="tru
        e"
        android:layout_centerHorizontal="true"/>

  </RelativeLayout>
```

✞ **MainActivity.java**

```
package com.example.experiment8;

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

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

```
}
```

DESIGN OF THE APP : **[Linear Layout]**



✞ **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:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/green"
     tools:context=".MainActivity">


     <TextView android:id="@+id/txtVw"
        android:layout_width="match_parent"
        android:layout_height="wrap_content
        " android:layout_margin="16dp"
```

```
        android:text="Enter your name here:"
        android:textSize="24dp"
        android:textStyle="bold" />

    <EditText android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content
        " android:layout_margin="16dp"
        android:hint="Name"
        android:inputType="text"/>

    <Button android:id="@+id/showInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal
        " android:backgroundTint="@color/blue"
        android:text="show"
        android:textColor="@android:color/white" />


</LinearLayout>
```

✞ **MainActivity.kt**

```
package com.example.exp8l

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle import
android.widget.Button import
android.widget.EditText
import android.widget.TextView

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

```
    val showButton = findViewById<Button>(R.id.showInput)
  val editText = findViewById<EditText>(R.id.editText) val
  textView = findViewById<TextView>(R.id.txtVw) }
}
```

Learning outcomes (What I have learnt):

- To design an android application using and styling different layouts in android studio
- Learnt about running application on android studio.
- Learnt about how different layout outputs.
- Learnt how to implement of layouts in app.

Experiment 3.3

Student Name: Vikash Yadav
UID: 21BCS8093
Section/Group: 20BCS_DM-719(B)
Branch: BE-CSE
Date of Performance: 11-05-2023
Semester: 6
Subject Code: 20CSP-356

AIM:
Design the Android application using Menus and action bar.

OBJECTIVE:
To design an android application by using menu.

HARDWARE/SOFTWARE REQUIREMENT:

- Java JDK 5 or later version
- Java Runtime Environnent (JRE) 6 Android Studio
- Microsoft Windows 10
- 4 GB RAM minimum , 8 GB (recommended)
- 10GB of available disk space minimum, 20 GB recommended
- 1280 x 800 minimum screen resolution

INTRODUCTION:

The menu is a part of the User Interface (UI) component, used to handle some common functionality around the app. To utilize the menu, you should define it in a separate XML file and use that file in your app based on your requirements. You can also use menu APIs to represent user actions and other options in your app activities.

STEPS TO CREATE APPLICATION:

✞ Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. We are implementing it for both Java and Kotlin languages.

✞ Add the Empty activity to the Project

Right-Click on the app, move the cursor to new, find the “Empty Activity” option at the end, select it and proceed. Files are automatically generated.

✞ Install and Run the Code

- Install and Run the code on Android Virtual Device (AVD) or a personal device.
- Open the Apps of the phone, lookup for a new App with the Application name. Run the App

<u>DESIGN OF THE APP</u> :

[APP FRONTEND]

CODE:

- **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:orientation="vertical"
    android:layout_height="match_parent" tools:context=".MainActivity">

    <include layout="@layout/toolbar_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/
        >

</LinearLayout>
```

- **toolbar_layout.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"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#450899"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme"
        app:logo="@drawable/ic_home"
        app:title="Home"
```

```
        app:titleTextColor="#EDF8F7" />
</LinearLayout>
```

✞ **menu_item.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item app:showAsAction="always"
        android:icon="@drawable/ic_more_vert
        "
        android:title="">
        <menu>
            <item
                android:id="@+id/android"
                android:icon="@drawable/baseline_android_24"
                android:title="Android"
                app:showAsAction="never"/>
            <item
                android:id="@+id/profile"
                android:icon="@drawable/baseline_person_24"
                android:title="Profile"
                app:showAsAction="never"/>
            <item android:id="@+id/download"
                android:icon="@drawable/ic_download
                " android:title="Download"
                app:showAsAction="never"/>
            <item android:id="@+id/setting"
                android:icon="@drawable/ic_settings
                "
                android:title="Setting"
                app:showAsAction="never"/>
            <item
                android:id="@+id/exit"
                android:icon="@drawable/baseline_exit_to_app_24
                " android:title="Exit"
                app:showAsAction="never"/>
```

```
        </menu>
    </item>
  </menu>
```

✞ **MainActivity.java**

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.os.Bundle;
import android.view.Menu;


public class MainActivity extends AppCompatActivity {

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

        Toolbar tb = (Toolbar)findViewById(R.id.toolbar);

        setSupportActionBar(tb);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_item,menu);
        return true;
    }
}
```

Learning outcomes (What I have learnt):

- To design an android application using menus in android studio
- Learnt about running application on android studio.
- Learnt about how toolbar works