ERODE SENGUNTHAR
ENGINEERING COLLEGE

(APPROVED BY AICTE, NEW DELHI & PERMANENTLY AFFILIATED TO ANNA UNIVERSITY, CHENNAI.
ACCREDITED BY NBA, NEW DELHI, NAAC WITH GRADE "A" & IE(I),KOLKATA)

PERUNDURAI, ERODE – 638 057

An Autonomous Institution

BONAFIDE CERTIFICATE

Register No: .....

Certified that this is the Bonafide Record of Work Done By

Name of the Student :.....

Branch :.....

Lab Code/Name :.....

Year/Semester :.....

Faculty Incharge Head of the Department

Submitted for the End Semester Practical Examination held on .....

Internal Examiner *External Examiner*

Ex.No:01	Develop an Application to display Image and text
Date:	

Aim:

To develop an application to display image and text.

Procedure:

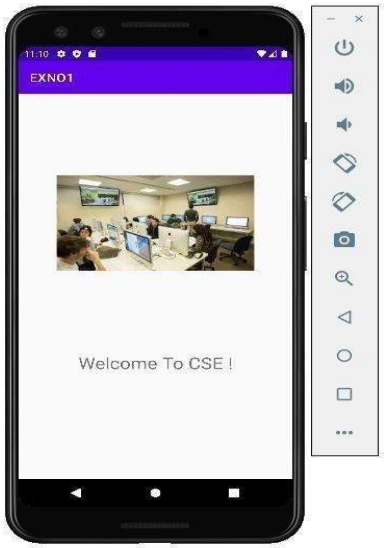
- Open Android Stdio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.1″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish

Program Coding:
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">


   <ImageView
      android:id="@+id/imageView2"
      android:layout_width="fill_paren
      t"
      android:layout_height="332dp"
      android:layout_margin="150px"
```

```
        android:layout_marginBottom="116  dp"

        app:layout_constraintBottom_toTopOf="@+id/editTextTextP
        er sonName2" app:srcCompat="@drawable/img1"
        tools:layout_editor_absoluteX="88dp" />

         <TextView
         android:id="@+id/editTextTextPersonNa
         me2" android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:gravity="center"
         android:text="Welcome To CSE !"
         android:textAlignment="center"
         android:textSize="75px"
         tools:layout_editor_absoluteX="154dp"
         tools:layout_editor_absoluteY="510dp"
        />
</LinearLayout>
```

Output:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application that uses to display the image and text is developed and executed successfully.

Ex.No:02	Develop a Simple Android Application that Uses Layout Managers and Event Listeners
Date:	

Aim:

To develop a Simple Android Application that uses Layout Managers and Event Listeners.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.2″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

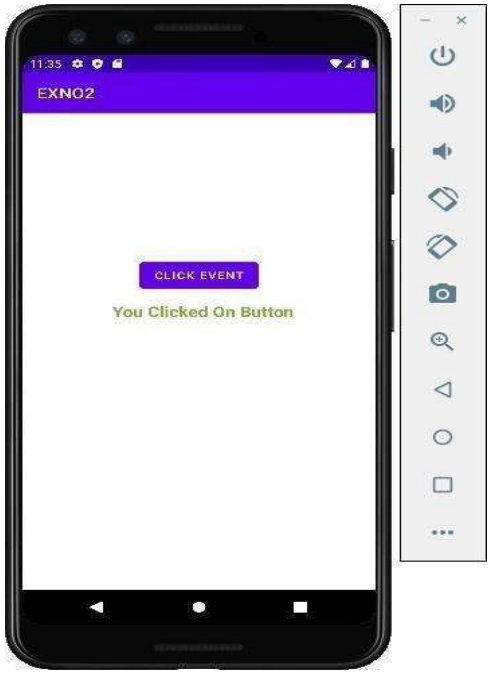
Activity_main.xml

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

   <Button
      android:id="@+id/btnClick"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Click Event"
      android:layout_marginTop="200dp"
      android:layout_marginLeft="130dp"
      android:textColor="#86AD33" />

   <TextView
      android:id="@+id/txtResult"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="100dp"
      android:textSize="20sp"
      android:textStyle="bold"
      android:layout_marginTop="12dp" />
</LinearLayout>
```

MainActivity.java

```
package com.example.exno2;

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

public class MainActivity extends AppCompatActivity {
    Button btn;
    TextView tView;

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

        btn = findViewById(R.id.btnClick);
        tView = findViewById(R.id.txtResult);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                tView.setText("You Clicked On Button");
            }
        });
    }
}
```

Output:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application that uses Layout Managers and Event Listeners is developed and executed successfully.

Ex.No:03	Develop a Simple Android Application that Draws Basic Graphical Primitives on the Screen
Date:	

Aim:

To develop a Simple Android Application that draws basic Graphical Primitives on the screen.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.3″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.xml:

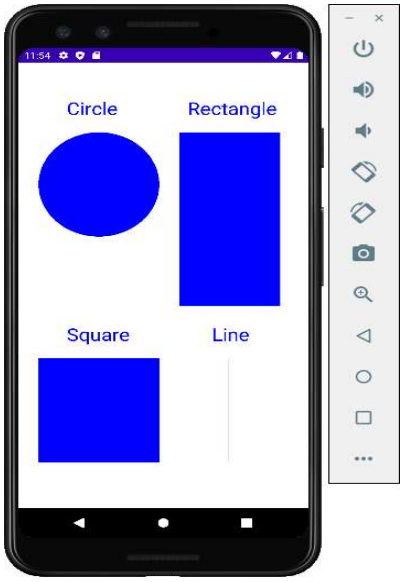
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <ImageView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/imageView" />
</RelativeLayout>
```

MainActivity.java:

```
package com.example.exno4;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.widget.ImageView;
```

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Creating a Bitmap
        Bitmap bg = Bitmap.createBitmap(720, 1280, Bitmap.Config.ARGB_8888);

        // Setting the Bitmap as background for the ImageView
        ImageView i = (ImageView) findViewById(R.id.imageView);
        i.setBackgroundDrawable(new BitmapDrawable(bg));

        // Creating the Canvas Object
        Canvas canvas = new Canvas(bg);

        // Creating the Paint Object and setting its color & text size
        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setTextSize(50);

        // Drawing a Rectangle
        canvas.drawText("Rectangle", 420, 150, paint);
        canvas.drawRect(400, 200, 650, 700, paint);

        // Drawing a Circle
        canvas.drawText("Circle", 120, 150, paint);
        canvas.drawCircle(200, 350, 150, paint);

        // Drawing a Square
        canvas.drawText("Square", 120, 800, paint);
        canvas.drawRect(50, 850, 350, 1150, paint);

        // Drawing a Line
        canvas.drawText("Line", 480, 800, paint);
        canvas.drawLine(520, 850, 520, 1150, paint);
    }
}
```

Output:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application that draws basic Graphical Primitive on the screen is developed and executed successfully.

Ex.No:04	Develop an Application Used for Changing Font
Date:	

Aim:

To develop an application Used for changing font size.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.4″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.xml:

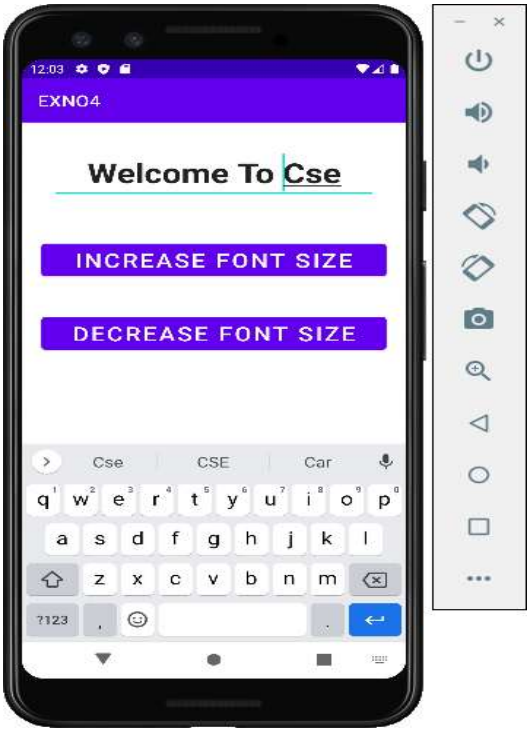
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <EditText
      android:id="@+id/textView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="30dp"
      android:gravity="center"
      android:text="Welcome To Cse"
      android:textSize="25sp"
      android:textStyle="bold" />

   <Button
      android:id="@+id/button1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="20dp"
      android:gravity="center"
      android:text="Increase Font size"
      android:textSize="25sp" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Decrease Font size"
        android:textSize="25sp" />
</LinearLayout>
```

MainActivity.java :

```
package com.example.exno4;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    int ch = 1;
    float font = 30;

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

        final TextView t = findViewById(R.id.textView);
        Button b1 = findViewById(R.id.button1);
        Button b2 = findViewById(R.id.button2);

        // Increase font size
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t.setTextSize(font);
                font += 5;
                if (font > 50) font = 30;
            }
        });

        // Decrease font size
        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t.setTextSize(font);
                font -= 5;
                if (font < 30) font = 30;}});}}
```

Output:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application that uses changing font size is developed and executed successfully.

Ex.No:05	Develop an Application Used for Changing Font Color
Date:	

Aim:

To develop an application Used for changing Color size.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.5″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.xml:

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

   <EditText
      android:id="@+id/textView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="30dp"
      android:gravity="center"
      android:text="Hello"
      android:textSize="25sp"
      android:textStyle="bold" />

   <Button
      android:id="@+id/button1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="20dp"
      android:gravity="center"
      android:text="Change Font Color"
      android:textSize="25sp" />
</LinearLayout>
```

MainActivity.java:

```
package com.example.exno5;
import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    int ch = 1;
    float font = 30;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final TextView t = findViewById(R.id.textView);
        Button b1 = findViewById(R.id.button1);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (ch) {
                    case 1:
                        t.setTextColor(Color.RED);
                        break;
                    case 2:
                        t.setTextColor(Color.GREEN);
                        break;
                    case 3:
                        t.setTextColor(Color.BLUE);
                        break;
                    case 4:
                        t.setTextColor(Color.CYAN);
                        break;
                    case 5:
                        t.setTextColor(Color.YELLOW);
                        break;
                    case 6:
                        t.setTextColor(Color.MAGENTA);
                        break;
                }
                ch++;
                if (ch == 7) ch = 1;
            }
        });
    }
}
```

Output:

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus, a Simple Android Application that uses changing font Color is developed and executed successfully.

Ex.No:06	Integrating Database with Mobile Apps
Date:	

Aim:

To connect the app with a database for real-time data storage, retrieval, and updates, ensuring smooth functionality.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.6″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.xml:

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

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="50dp"
      android:layout_y="20dp"
      android:text="Student Details"
      android:textSize="30sp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="20dp"
      android:layout_y="110dp"
      android:text="Enter Rollno:"
      android:textSize="20sp" />

   <EditText
      android:id="@+id/rollno"
      android:layout_width="150dp"
      android:layout_height="wrap_content"
      android:layout_x="175dp"
```

```
    android:layout_y="100dp"
    android:inputType="number"
    android:textSize="20sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="20dp"
    android:layout_y="160dp"
    android:text="Enter Name:"
    android:textSize="20sp" />

<EditText
    android:id="@+id/Name"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_x="175dp"
    android:layout_y="150dp"
    android:inputType="text"
    android:textSize="20sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="20dp"
    android:layout_y="210dp"
    android:text="Enter Marks:"
    android:textSize="20sp" />

<EditText
    android:id="@+id/Marks"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_x="175dp"
    android:layout_y="200dp"
    android:inputType="number"
    android:textSize="20sp" />

<Button
    android:id="@+id/Insert"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_x="25dp"
    android:layout_y="300dp"
    android:text="Insert"
    android:textSize="30dp" />

<Button
```

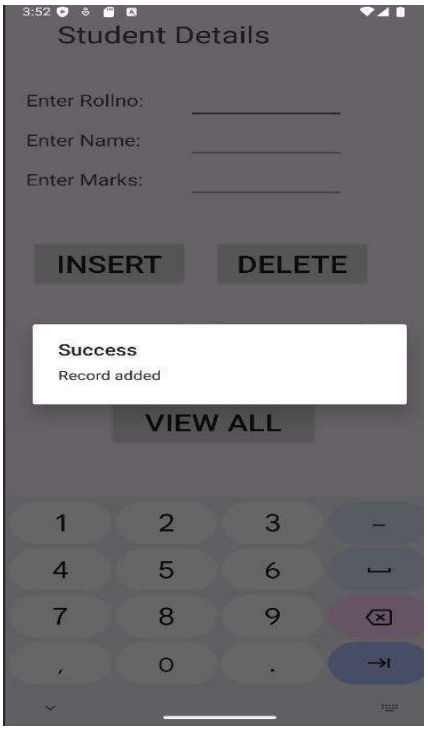
```
        android:id="@+id/Delete"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="200dp"
        android:layout_y="300dp"
        android:text="Delete"
        android:textSize="30dp" />

    <Button
        android:id="@+id/Update"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="25dp"
        android:layout_y="400dp"
        android:text="Update"
        android:textSize="30dp" />

    <Button
        android:id="@+id/View"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="200dp"
        android:layout_y="400dp"
        android:text="View"
        android:textSize="30dp" />

    <Button
        android:id="@+id/ViewAll"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_x="100dp"
        android:layout_y="500dp"
        android:text="View All"
        android:textSize="30dp" />

</AbsoluteLayout>
```

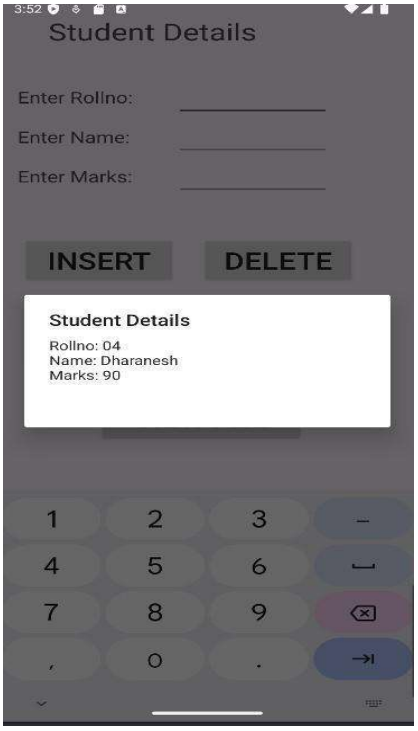
MainActivity.java:

```
package com.example.ex7;
import android.app.Activity;
import android.app.AlertDialog.Builder;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity implements OnClickListener {
    EditText Rollno, Name, Marks;
    Button Insert, Delete, Update, View, ViewAll;
    SQLiteDatabase db;

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

        Rollno = findViewById(R.id.rollno);
        Name = findViewById(R.id.Name);
        Marks = findViewById(R.id.Marks);
        Insert = findViewById(R.id.Insert);
        Delete = findViewById(R.id.Delete);
        Update = findViewById(R.id.Update);
        View = findViewById(R.id.View);
        ViewAll = findViewById(R.id.ViewAll);

        Insert.setOnClickListener(this);
        Delete.setOnClickListener(this);
        Update.setOnClickListener(this);
        View.setOnClickListener(this);
        ViewAll.setOnClickListener(this);

        // Creating database and table
        db = openOrCreateDatabase("StudentDB", Context.MODE_PRIVATE, null);
        db.execSQL("CREATE TABLE IF NOT EXISTS student(rollno VARCHAR, name VARCHAR,
marks VARCHAR);");
    }

    public void onClick(View view) {
        if (view == Insert) {
            if (Rollno.getText().toString().trim().isEmpty() ||
                Name.getText().toString().trim().isEmpty() ||
```

```
            Marks.getText().toString().trim().isEmpty()) {
            showMessage("Error", "Please enter all values");
            return;
        }
        db.execSQL("INSERT INTO student VALUES('" + Rollno.getText() + "','" + Name.getText()
+ "','" + Marks.getText() + "');");
        showMessage("Success", "Record added");
        clearText();
    }

    // Similar formatting applies to Delete, Update, View, ViewAll sections...
  }

  public void showMessage(String title, String message) {
    Builder builder = new Builder(this);
    builder.setCancelable(true);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.show();
  }

  public void clearText() {
    Rollno.setText("");
    Name.setText("");
    Marks.setText("");
    Rollno.requestFocus();
  }
}
```

Output:


3:52
Student Details
Enter Rollno:
Enter Name:
Enter Marks:
INSERT
DELETE
Success
Record added
VIEW ALL
1 2 3
4 5 6
7 8 9
, 0 .


3:52
Student Details
Enter Rollno:
Enter Name:
Enter Marks:
INSERT
DELETE
Student Details
Rollno: 04
Name: Dharanesh
Marks: 90
1 2 3
4 5 6
7 8 9
, 0 .

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus Android Application that creates an Integrating Database with Mobile Apps is developed and executed successfully.

Ex.No:07	Integrating Database with Mobile Apps
Date:	

Aim:

To Develop an Android Application that displaying images using multithreading

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.7″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.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">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/img2" />

</RelativeLayout>
```

MainActivity.java:

```
package com.example.ex8;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

import java.io.InputStream;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    private ImageView imageView;

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

        imageView = findViewById(R.id.imageView);
        new LoadImageTask().execute("https://example.com/image.jpg");
    }

    private class LoadImageTask extends AsyncTask<String, Void, Bitmap> {
        @Override
        protected Bitmap doInBackground(String... params) {
            String url = params[0];
            Bitmap bitmap = null;

            try {
                InputStream inputStream = new URL(url).openStream();
                bitmap = BitmapFactory.decodeStream(inputStream);
            } catch (Exception e) {
                e.printStackTrace();
            }

            return bitmap;
        }

        @Override
        protected void onPostExecute(Bitmap bitmap) {
            if (bitmap != null) {
                imageView.setImageBitmap(bitmap);
            }
        }
    }
}
```

Output:

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application that displaying images using multithreading is developed and executed successfully.

Ex.No:08	Create an Dialer Application
Date:	

Aim:

To develop a Simple Android Application to Create an dialer application.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “ex.no.8″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

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.CALL_PHONE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

``

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center">

    <TextView
        android:id="@+id/phone_number_display"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_marginBottom="24dp"
        android:layout_marginTop="120dp"
        android:gravity="center"
        android:text=""
        android:textSize="50sp" />

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:columnCount="3"
        android:layout_marginBottom="24dp">

        <!-- Number buttons -->
        <Button android:id="@+id/button1" android:layout_width="80dp"
android:layout_height="80dp" android:text="1" />
        <Button android:id="@+id/button2" android:layout_width="80dp"
android:layout_height="80dp" android:text="2" />
        <Button android:id="@+id/button3" android:layout_width="80dp"
android:layout_height="80dp" android:text="3" />
        <Button android:id="@+id/button4" android:layout_width="80dp"
android:layout_height="80dp" android:text="4" />
        <Button android:id="@+id/button5" android:layout_width="80dp"
android:layout_height="80dp" android:text="5" />
        <Button android:id="@+id/button6" android:layout_width="80dp"
android:layout_height="80dp" android:text="6" />
        <Button android:id="@+id/button7" android:layout_width="80dp"
android:layout_height="80dp" android:text="7" />
        <Button android:id="@+id/button8" android:layout_width="80dp"
android:layout_height="80dp" android:text="8" />
        <Button android:id="@+id/button9" android:layout_width="80dp"
android:layout_height="80dp" android:text="9" />
        <Button android:id="@+id/button_star" android:layout_width="80dp"
android:layout_height="80dp" android:text="*" />
```

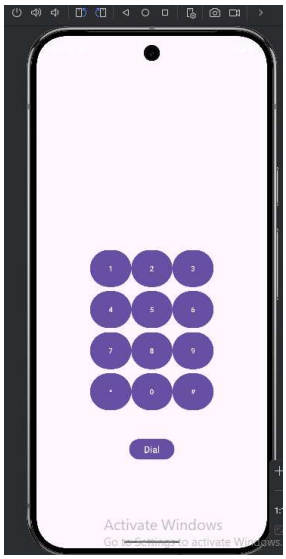
```
        <Button android:id="@+id/button0" android:layout_width="80dp"
android:layout_height="80dp" android:text="0" />
        <Button android:id="@+id/button_hash" android:layout_width="80dp"
android:layout_height="80dp" android:text="#" />
    </GridLayout>

    <Button
        android:id="@+id/dial_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dial"
        android:textSize="18sp"
        android:layout_marginTop="24dp" />
</LinearLayout>
```

MainActivity.java:

```
package com.example.myapplication;

import android.Manifest;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {
    private TextView phoneNumberDisplay;
    private StringBuilder phoneNumber = new StringBuilder();

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

        phoneNumberDisplay = findViewById(R.id.phone_number_display);

        // Define all number buttons
        int[] buttonIds = {R.id.button1, R.id.button2, R.id.button3, R.id.button4, R.id.button5,
                R.id.button6, R.id.button7, R.id.button8, R.id.button9, R.id.button0,
                R.id.button_star, R.id.button_hash};
```

```
        for (int id : buttonIds) {
            Button button = findViewById(id);
            button.setOnClickListener(v -> {
                phoneNumber.append(button.getText().toString());
                phoneNumberDisplay.setText(phoneNumber.toString());
            });
        }

        // Dial button action
        Button dialButton = findViewById(R.id.dial_button);
        dialButton.setOnClickListener(v -> dialPhoneNumber());
    }

    private void dialPhoneNumber() {
        String number = phoneNumber.toString();
        if (!number.isEmpty()) {
            if (ContextCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE)
                    != android.content.pm.PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.CALL_PHONE}, 1);
            } else {
                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + number));
                startActivity(callIntent);
            }
        } else {
            Toast.makeText(this, "Please enter a phone number", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Output:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application is developed and Create an dialer application executed successfully.

Ex.No:09	Creating A Weather App
Date:	

Aim:

To develop a android Application creating a Weather App.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as “exno10″and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.
- It will take some time to build and load the project.
- After completion it will look as given below. Designing layout for the Android Application:
- Click on app -> res -> layout -> activity_main.xml.
- Now click on Text as shown below.
- Then delete the code which is there and type the code as given below.

Program Coding:

Activity_main.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@drawable/background"
    android:padding="10dp">

    <!-- City Name -->
    <TextView
        android:id="@+id/cityNameText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:fontFamily="@font/urbanist"
        android:text="City"
```

```
    android:textColor="@color/white"
    android:textSize="36sp"
    android:textStyle="bold" />

<!-- Temperature -->
<TextView
    android:id="@+id/TemperatureText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"
    android:fontFamily="@font/urbanist"
    android:text="25°"
    android:textColor="#FFBF00"
    android:textSize="60sp"
    android:textStyle="bold"
    android:layout_below="@id/cityNameText" />

<!-- Details Layout -->
<LinearLayout
    android:id="@+id/detailsLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_below="@id/TemperatureText"
    android:background="@drawable/background2">

    <!-- Humidity Layout -->
    <LinearLayout
        android:id="@+id/humidityLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="0.5"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/humidityIcon"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:layout_gravity="center"
            android:src="@drawable/humidity" />

        <TextView
            android:id="@+id/humidityText"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="60%"
            android:textSize="16sp"
            android:textStyle="bold"
            android:layout_gravity="center"
            android:textColor="@color/white"
            android:fontFamily="@font/urbanist" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Humidity"
            android:textSize="16sp"
            android:layout_gravity="center"
            android:textColor="@color/white"
            android:fontFamily="@font/urbanist" />
    </LinearLayout>

    <!-- Wind Layout -->
    <LinearLayout
        android:id="@+id/windLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="0.5"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/windIcon"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:layout_gravity="center"
            android:src="@drawable/wind" />

        <TextView
            android:id="@+id/windText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="10 km/hr"
            android:textSize="16sp"
            android:textStyle="bold"
            android:layout_gravity="center"
            android:textColor="@color/white"
            android:fontFamily="@font/urbanist" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:text="Wind"
            android:textSize="16sp"
            android:layout_gravity="center"
            android:textColor="@color/white"
            android:fontFamily="@font/urbanist" />
    </LinearLayout>
</LinearLayout>

<!-- Weather Icon -->
<ImageView
    android:id="@+id/weatherIcon"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_below="@id/detailsLayout"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:elevation="12dp"
    android:src="@drawable/ic_01d"
    android:contentDescription="Weather Icon" />

<!-- Weather Description -->
<TextView
    android:id="@+id/descriptionText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/weatherIcon"
    android:layout_centerHorizontal="true"
    android:fontFamily="@font/urbanist"
    android:text="Sunny"
    android:textSize="28sp"
    android:textColor="@color/white" />

<!-- City Name Input -->
<EditText
    android:id="@+id/cityNameInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/descriptionText"
    android:padding="12dp"
    android:textSize="24sp"
    android:textColor="#EFEFEF"
    android:textColorHint="#BFBFBF"
    android:gravity="center"
    android:fontFamily="@font/urbanist"
    android:layout_marginTop="20dp"
    android:hint="Enter City Name" />

<!-- Fetch Weather Button -->
```

```
    <Button
        android:id="@+id/fetchWeatherButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/cityNameInput"
        android:backgroundTint="#2B3A67"
        android:fontFamily="@font/urbanist"
        android:text="Change City"
        android:textColor="#FFF"
        android:textSize="20sp" />
</RelativeLayout>
```

Build.gradle.kts(:app):

```
implementation("com.squareup.okhttp3:okhttp:4.9.3")
```

MainActivity.java:

```
package com.example.weatherer;
import androidx.appcompat.app.AppCompatActivity; import
android.os.Bundle;
import android.view.View; import
android.widget.Button; import
android.widget.EditText;
import android.widget.ImageView; import
android.widget.TextView;

import org.json.JSONException; import
org.json.JSONObject;

import okhttp3.OkHttpClient;
import okhttp3.Request; import
okhttp3.Response;

import java.io.IOException;
import java.util.concurrent.ExecutorService; import
java.util.concurrent.Executors;

public class MainActivity extends AppCompatActivity {
    private TextView cityNameText, temperatureText, humidityText, descriptionText,
```

```
windText;
    private ImageView weatherIcon;
    private Button refreshButton; private
    EditText cityNameInput;
    private static final String API_KEY = "YOUR_API_KEY";

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

        cityNameText = findViewById(R.id.cityNameText);
        temperatureText = findViewById(R.id.TemperatureText);
        humidityText = findViewById(R.id.humidityText); windText=
        findViewById(R.id.windText);  descriptionText =
        findViewById(R.id.descriptionText); weatherIcon =
        findViewById(R.id.weatherIcon); refreshButton  =
        findViewById(R.id.fetchWeatherButton); cityNameInput =
        findViewById(R.id.cityNameInput);

        refreshButton.setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View view) {
                String cityName = cityNameInput.getText().toString();
                if(!cityName.isEmpty())
                {
                    FetchWeatherData(cityName);
                }
                else
                {
                  cityNameInput.setError("Please enter a city name");
              }
            }
        });




        FetchWeatherData("Mumbai");
    }

    private void FetchWeatherData(String cityName) {
```

```
        String url = "https:api.openweathermap.org/data/2.5/weather?q="+cityName + "&appid="+
API_KEY + "&units=metric";
        ExecutorService executorService = Executors.newSingleThreadExecutor();
        executorService.execute(() ->
                {
                    OkHttpClient client = new OkHttpClient();
                    Request request = new Request.Builder().url(url).build(); try {
                        Response response = client.newCall(request).execute(); String
                        result = response.body().string(); runOnUiThread(() ->
                        updateUI(result));
                    } catch (IOException e)
                    {
                        e.printStackTrace();;
                    }
                }
        );
    }

    private void updateUI(String result)
    {
        if(result != null)
        {
            try {
                JSONObject jsonObject = new JSONObject(result); JSONObject main
                = jsonObject.getJSONObject("main"); double temperature =
                main.getDouble("temp");

 double humidity = main.getDouble("humidity"); double
                windSpeed =
jsonObject.getJSONObject("wind").getDouble("speed");

                String description =
jsonObject.getJSONArray("weather").getJSONObject(0).getString("description"
);
                String iconCode =
jsonObject.getJSONArray("weather").getJSONObject(0).getString("icon"
);
                String resourceName = "ic_" + iconCode;
                int resId = getResources().getIdentifier(resourceName, "drawable", getPackageName());
                weatherIcon.setImageResource(resId);

                cityNameText.setText(jsonObject.getString("name"));
```

```
                temperatureText.setText(String.format("%.0f°",
                temperature));
                humidityText.setText(String.format("%.0f%%", humidity));
                windText.setText(String.format("%.0f km/h", windSpeed));
                descriptionText.setText(description);
            } catch (JSONException e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

Add this at 4th line of AndroidManifest.Xml:

<uses-permission android:name="android.permission.INTERNET"/> Add 2 images as humidity & wind in drawables

GET_API_KEY:

- 1.Go to open Weather
- 2.login/create your account
- 3.get one weather api
- 4.copy the keys from My api keys
- 5.paste it in activity_main at your api keys

Output:

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a simple Application to Check weather has been created and executed Successfully.

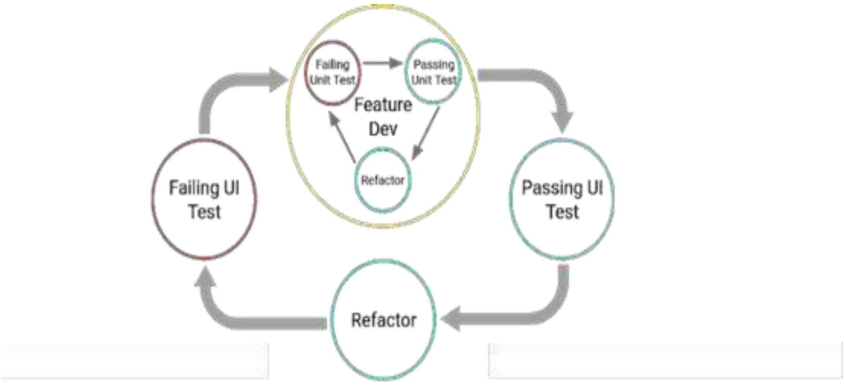
Ex.No:10	Testing Mobile Applications
Date:	

Organize your code for testing:

As your app expands, you might find it necessary to fetch data from a server, interact with the device's sensors, access local storage, or render complex user interfaces. The versatility of your app demands a comprehensive testing strategy.

Create and test code iteratively:

When developing a feature iteratively, you start by either writing a new test or by adding cases and assertions to an existing unit test. The test fails at first because the feature isn't implemented yet. It's important to consider the units of responsibility that emerge as you design the new feature. For each unit, you write a corresponding unit test. Your unit tests should nearly exhaust all possible interactions with the unit, including standard interactions, invalid inputs, and cases where resources aren't available. Take advantage of Jetpack libraries whenever possible; when you use these well-tested libraries, you can focus on validating behavior that's specific to your app.



The full workflow, as shown in Figure 1, contains a behavior that's specific to your appseries of nested, iterative cycles where a long, slow, UI-driven cycle tests the integration of code units. You test the units themselves using shorter, faster

development cycles. This set of cycles continues until your app satisfies every use case.

View your app as a series of modules:

To make your code easier to test, develop your code in termsof modules, where each module represents a specific task that users complete within your app. This perspective contrasts the stack-based view of an app that typically contains layers representing the UI, business logic, and data.

For example, a "task list" app might have modules for creating tasks, viewing statistics about completed tasks, and taking photographs to associate with a particular task. Such a modular architecture also helps you keep unrelated classes decoupled and provides a natural structure for assigning ownership within your development team.

It's important to set well-defined boundaries around each module, and to create new modules as your app grows in scale and complexity. Each module should have only one area of focus, and the APIs that allow for inter-module communication should be consistent. To make it easier and quicker to test these inter-module interactions, consider creating fake implementations of your modules. In your tests, the real implementation of one module can call the fake implementation of the other module.

As you create a new module, however, don't be too dogmatic about making it full-featured right away. It's OK for a particular module to not have one or more layers of the app's stack.

Configure your test environment:

When setting up your environment and dependencies for creating tests in your app, follow the best practices described in this section.

Organize test directories based on execution environment:

A typical project in Android Studio contains two directories in which you place tests. Organize your tests as follows:

- The androidTest directory should contain the tests that run on real or virtual devices. Such tests include integration tests, end-to-end tests, and

other tests where the JVM alone cannot validate your app's functionality.

- The test directory should contain the tests that run on your local machine, such as unit tests.

Consider tradeoffs of running tests on different types of devices:

When running your tests on a device, you can choose among the following types:

- Real device
- Virtual device (such as the Android Studio)
- Simulated device (such as Roboelectric)

Real devices offer the highest fidelity but also take the most time to run your tests. Simulated devices, on the other hand, provide improved test speed at the cost of lower fidelity. The platform's improvements in binary resources and realistic loopers, however, allow simulated devices to produce more realistic results.

Virtual devices offer a balance of fidelity and speed. When you use virtual devices to test, use to minimize setup time in between tests.

Consider whether to use test doubles:

When creating tests, you have the option of creating real objects or *test doubles*, such as fake objects or mock objects. Generally, using real objects in your tests is better than using test doubles, especially when the object under test satisfies one of the following conditions:

- The object is a data object.
- The object cannot function unless it communicates with the real object version of a dependency. A good example is an event callback handler.
- It's hard to replicate the object's communication with a dependency. A good example is a SQL database handler, where an in-memory database provides more robust tests than fakes of database results.

In particular, mocking instances of types that you don't own usually leads to brittle tests that work only when you've understood the complexities of someone else's implementation of that type. Use such mocks only as a last resort. It's OK to

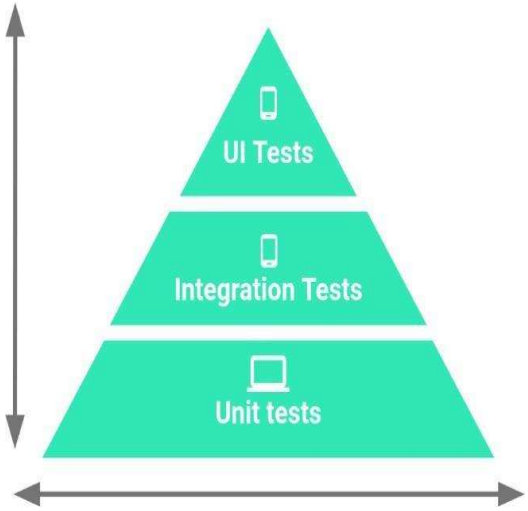
mock your own objects, but keep in mind that mocks annotated using SPY provide more fidelity than mocks that stub out all functionality within a class.

However, it's better to create fake or even mock objects if your tests try to perform the following types of operations on a real object:

- Long operations, such as processing a large file.
- Non-hermetic actions, such as connecting to an arbitrary open port.
- Hard-to-create configurations.

Write your tests:

After you've configured your testing environment, it's time to write tests that evaluate your app's functionality. This section describes how to write small, medium, and large tests.Levels of the Testing Pyramid:



The Testing Pyramid, showing the three categories of tests that you shouldinclude

in your app's test suite. The Testing Pyramid, shown in Figure, illustrates how

your app should include the three categories of tests: small, medium, and large

:

- Small tests are unit tests that validate your app's behavior one class at a time.
- Medium tests are integration tests that validate either interactions between levels of the stack within a module, or interactions between related modules.

- Large tests are end-to-end tests that validate user journeys spanning multiple modules of your app.

Local unit tests:

Use the AndroidX Test APIs whenever possible so that your unit tests can run on a device or emulator. For tests that always run on a JVM-powered development machine, you can use Roboelectric.

Robolectric simulates the runtime for Android 4.1 (API level 16) or higher and provides community-maintained fakes called *shadows*. This functionality allows you to test code that depends on the framework without needing to use an emulator or mock objects. Robolectric supports the following aspects of the Android platform:

- Component lifecycles
- Event loops
- All resources

Instrumented unit tests:

You can run instrumented unit tests on a physical device or emulator. This form oftesting involves significantly slower execution times than local unit tests, however, so it's best to rely on this method only when it's essential to evaluate your app's behavior against actual device hardware.

When running instrumented tests, AndroidX Test makes use of the following threads:

- The main thread, also known as the "UI thread" or the "activity thread", where UI interactions and activity lifecycle events occur.
- The instrumentation thread, where most of your tests run. When your test suite begins, the AndroidJUnitTest class starts this thread.

If you need a test to execute on the main thread, annotate it using UiThreadTest.

Write medium tests:

In addition to testing each unit of your app by running small tests, you should validate your app's behavior from the module level. To do so, write medium tests, which are integration tests that validate the collaboration and interaction of a group of units.

Use your app's structure and the following examples of medium tests (in order of increasing scope) to define the best way to represent groups of units in your app:

- Interactions between a view and view model, such as testinga Fragment object, validating layout XML, or evaluating the data- binding logic of a ViewModel object.
- Tests in your app's repository layer, which verify that your different data sources and data access objects (DAOs) interact as expected.
- Vertical slices of your app, testing interactions on a particular screen. Such a test verifies the interactions throughout the layers of your app's stack.
- Multi-fragment tests that evaluate a specific area of your app. Unlike the other types of medium tests mentioned in this list, this type of testusually requires a real device because the interaction under test involves multiple UI elements.

To carry out these tests, do the following:

1. Use methods from the Espresso Intents library. To simplify the information that you're passing into these tests, use fakes and stubbing.
2. Combine your use of and Truth-based assertions to verify the captured intents.

Run UI tests:

The class defines an instrumentation-based test runner that lets you run JUnit 3- or JUnit 4- style test classes on Android devices. The test runner facilitates loading your test package and the app under test onto a device oremulator, running your tests, and reporting the results.

To further increase these tests' reliability, use Android Test Orchestrator, which runs each UI test in its own sandbox. This architecture reduces shared state between tests and isolates app crashes on a per-test basis. For more information about the benefits that Android Test Orchestrator provides as you test your app, see the Android Test Orchestrator guide.

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus the study of Testing the Android application have been done successfully.

CONTENT BEYOND SYLLABUS

Ex.No:11	Simple Android Application for Native Calculator
Date:	

Aim:

To develop a Simple Android Application for Native Calculator.

Procedure:

- Open Android Studio and then click on File -> New -> New project.
- Then type the Application name as "ex.no.11″ and click Next.
- Then select the Minimum SDK as shown below and click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.

Program Coding:

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_margin="20dp">

   <LinearLayout
      android:id="@+id/linearLayout1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="20dp">

      <EditText
         android:id="@+id/editText1"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:inputType="numberDecimal"
         android:textSize="20sp" />

      <EditText
         android:id="@+id/editText2"
         android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="numberDecimal"
        android:textSize="20sp" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp">

        <Button
            android:id="@+id/Add"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="+"
            android:textSize="30sp" />

        <Button
            android:id="@+id/Sub"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="-"
            android:textSize="30sp" />

        <Button
            android:id="@+id/Mul"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="*"
            android:textSize="30sp" />

        <Button
            android:id="@+id/Div"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="/"
            android:textSize="30sp" />
    </LinearLayout>

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

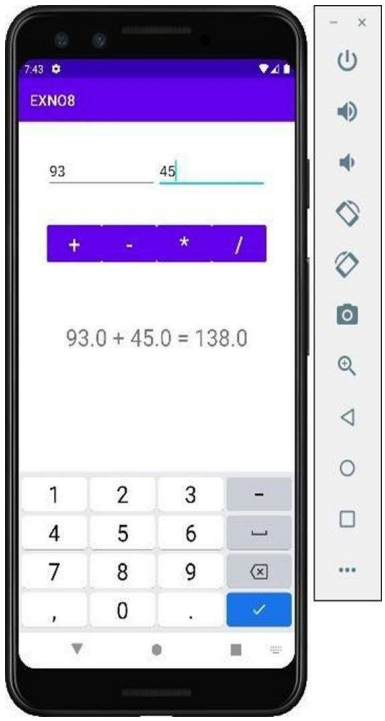
```
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:text="Answer is"
        android:textSize="30sp"
        android:gravity="center" />
</LinearLayout>
```

MainActivity.java:

```
package com.example.exo8;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // Defining the Views
    EditText Num1, Num2;
    Button Add, Sub, Mul, Div;
    TextView Result;

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

        // Referring the Views
        Num1 = findViewById(R.id.editText1);
        Num2 = findViewById(R.id.editText2);
        Add = findViewById(R.id.Add);
        Sub = findViewById(R.id.Sub);
        Mul = findViewById(R.id.Mul);
        Div = findViewById(R.id.Div);
        Result = findViewById(R.id.textView);

        // Set listeners
        Add.setOnClickListener(this);
        Sub.setOnClickListener(this);
        Mul.setOnClickListener(this);
        Div.setOnClickListener(this);
    }

    @Override
```

```
    public void onClick(View v) {
        float num1 = 0, num2 = 0, result = 0;
        String oper = "";

        // Check if the fields are empty
        if (TextUtils.isEmpty(Num1.getText().toString()) ||
TextUtils.isEmpty(Num2.getText().toString())) {
            return;
        }

        // Read EditText and fill variables with numbers
        num1 = Float.parseFloat(Num1.getText().toString());
        num2 = Float.parseFloat(Num2.getText().toString());

        // Defines the button that has been clicked and performs the corresponding operation
        switch (v.getId()) {
            case R.id.Add:
                oper = "+";
                result = num1 + num2;
                break;
            case R.id.Sub:
                oper = "-";
                result = num1 - num2;
                break;
            case R.id.Mul:
                oper = "*";
                result = num1 * num2;
                break;
            case R.id.Div:
                oper = "/";
                result = num1 / num2;
                break;
            default:
                break;
        }

        // Form the output line
        Result.setText(num1 + " " + oper + " " + num2 + " = " + result);
    }
}
```

OUTPUT:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

Result:

Thus a Simple Android Application for Native Calculator is developed and executed successfully.