RIET | RAJASTHAN INSTITUTE OF ENGINEERING AND TECHNOLOGY

Established in year 2000- RTU QIV RANKED A

Approved by AICTE and Affiliated to Rajasthan Technical University, Kota

Lab Manual

(6CS4-24) Mobile Application Development Lab Semester: VI

Branch: Computer Science & Engineering

Session 2020-2021

Faculty:

Dinesh Swami (Assistant Professor)

Department of Computer Science & Engineering

RIET, JAIPUR

LAB ETHICS

DO's

1. Please switch off the Mobile/Cell phone before entering Lab.
2. Enter the Lab with complete source code and data.
3. Check whether all peripheral are available at your desktop before proceeding for program.
4. Intimate the lab In charge whenever you are incompatible in using the system or in case software get corrupted/ infected by virus.
5. Arrange all the peripheral and seats before leaving the lab.
6. Properly shutdown the system before leaving the lab.
7. Keep the bag outside in the racks.
8. Enter the lab on time and leave at proper time.
9. Maintain the decorum of the lab.
10. Utilize lab hours in the corresponding experiment.
11. Get your Cd / Pendrive checked by lab In charge before using it in the lab.

Don'ts

1. No one is allowed to bring storage devices like Pan Drive /Floppy etc. in the lab.
2. Don't mishandle the system.
3. Don't leave the system on standing for long
4. Don't bring any external material in the lab.
5. Don't make noise in the lab.
6. Don't bring the mobile in the lab. If extremely necessary then keep ringers off.
7. Don't enter in the lab without permission of lab Incharge.
8. Don't litter in the lab.
9. Don't delete or make any modification in system files.
10. Don't carry any lab equipments outside the lab

INSTRUCTIONS

BEFORE ENTERING IN THE LAB

- All the students are supposed to prepare the theory regarding the next program.
- Students are supposed to bring the practical file and the lab copy.
- Previous programs should be written in the practical file.
- Algorithm of the current program should be written in the lab copy.
- Any student not following these instructions will be denied entry in the lab.

WHILE WORKING IN THE LAB

- Adhere to experimental schedule as instructed by the lab incharge.
- Get the previously executed program signed by the instructor.
- Get the output of the current program checked by the instructor in the lab copy.
- Each student should work on his/her assigned computer at each turn of the lab.
- Take responsibility of valuable accessories.
- Concentrate on the assigned practical and do not play games.
- If anyone caught red handed carrying any equipment of the lab, then he will have to face serious consequences.

List of Experiments

S.No.	Name of Experiment
1	To develop a Simple Android Application that uses GUI components, Font and Colors
2	To develop a Simple Android Application that uses Layout Managers and Event Listeners.
3	Design simple GUI application with activity and intents e.g. calculator
4	Develop an application that makes use of RSS Feed.
5	Write an application that draws basic graphical primitives on the screen
6	Create an android app for database creation using SQlite Database
7	Develop a native application that uses GPS location information
8	Implement an application that writes data on the SD card.
9	Design a StopWatch application
10	Create an application to play a video in android

Introduction to Android

Android Studio is a new and fully integrated development environment, which has been recently launched by Google for the Android operating system. It has been designed to provide new tools for app development and to provide an alternative to Eclipse, currently the most widely used IDE. When you begin a new project in Android studio, the project's structure will appear with almost all the files held within the SDK directory, this switch to a Gradle based management system offers an even greater flexibility to the build process.

Android Studio allows you to see any visual changes you make to your app in real-time, and you can also see how it will look on a number of different Android devices, each with different configurations and resolutions, simultaneously. Another feature in Android Studio are the new tools for the packing and labelling of code. These let you keep on top of your project when dealing with large amounts of code. The programme also uses a drag & drop system to move the components throughout the user interface.

The programme will also help you to localize your apps, giving you a visual way to keep programming while controlling the flow of the application.

What else does Android Studio offer?

- A robust and straight forward development environment.

- An easy way to test performance on other types of device.

- Wizards and templates for common elements found in all Android programming.

- A full-featured editor with lots of extra tools to speed up the development of your applications.

Android UI Controls: There are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

- **TextView::** This control is used to display text to the user.
- **EditText::** EditText is a predefined subclass of TextView that includes rich editing capabilities.
- **AutoCompleteTextView**:: The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
- **Button::** A push-button that can be pressed, or clicked, by the user to perform an action.
- **ImageButton::** AbsoluteLayout enables you to specify the exact location of its children.
- **CheckBox::** An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.

- **ToggleButton::** An on/off button with a light indicator.
- **RadioButton::** The RadioButton has two states: either checked or unchecked.
- **Spinner**::A drop-down list that allows users to select one value from a set.

Android Event Handling : Events are a useful way to collect data about a user's interaction with interactive components of your app, like button presses or screen touch etc. The Android framework maintains an event queue into which events are placed as they occur and then each event is removed from the queue on a first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements. There are following three concepts related to Android Event Management:

- **Event Listeners:** The View class is mainly involved in building up a Android GUI, same View class provides a number of Event Listeners. The Event Listener is the object that receives notification when an event happens.

- **Event Listeners Registration:** Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

- **Event Handlers:** When an event happens and we have registered and event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	onClickListener() This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	onLongClickListener() This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	onFocusChangeListener() This is called when the widget loses its focus i.e. user go away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	onTouchListener() This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	onMenuItemClickListener() This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.

Download Android Studio

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can download Android Studio from the Android Studio homepage, where you'll also find the traditional SDKs with Android Studio's command-line tools. Before downloading Android Studio, make sure your platform meets the following requirements:

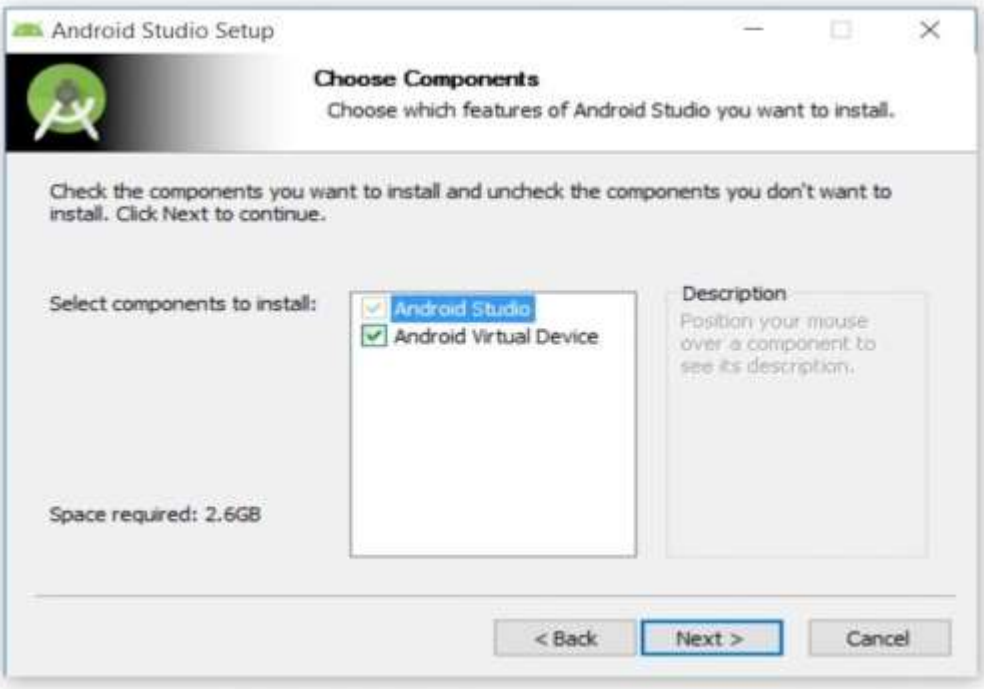
Windows requirements

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

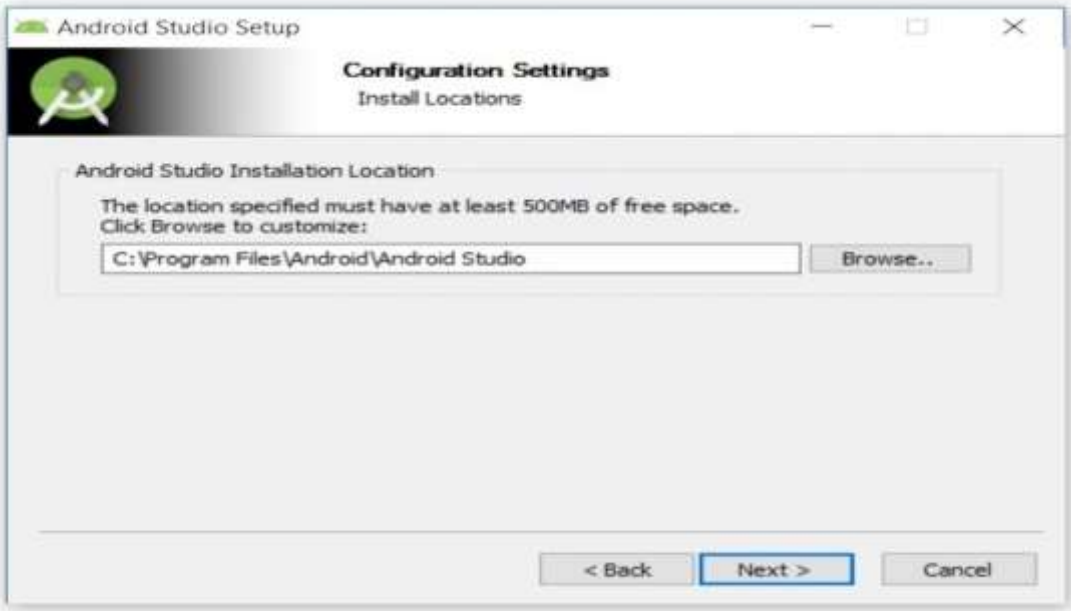
Installing Android Studio on 64-bit Windows 10

Launch android-studio-ide-181.5056338-windows.exe to start the installation process.

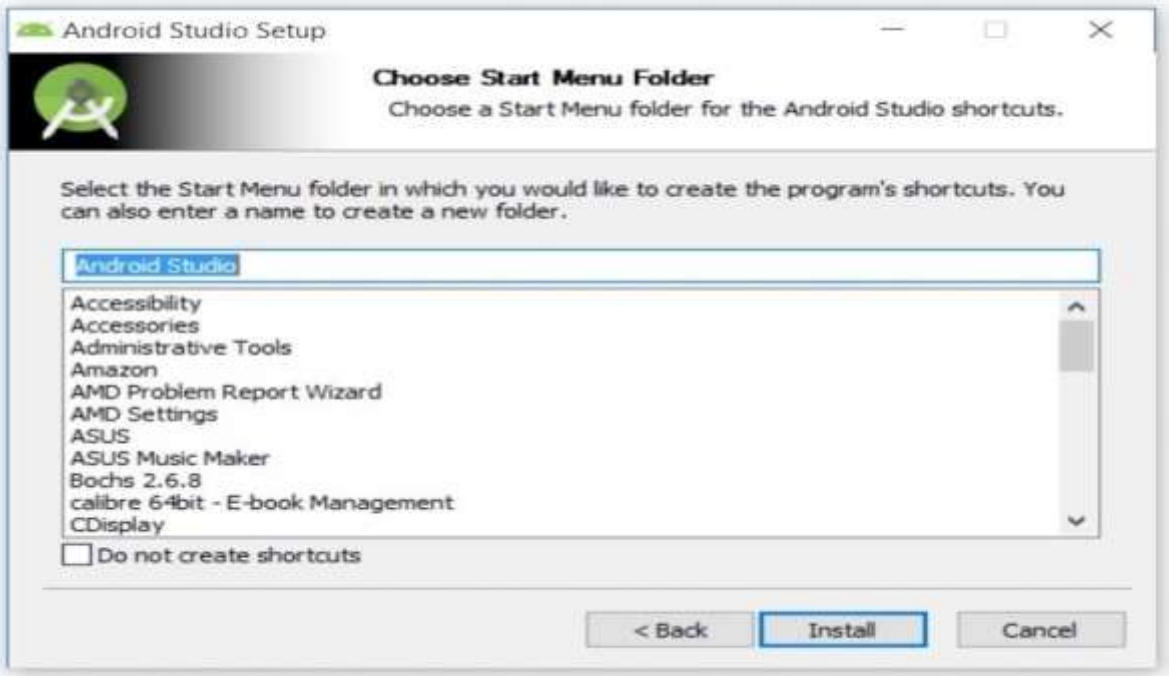
Clicking Next took to the following panel, which provides the option to decline installing an Android Virtual Device (AVD).



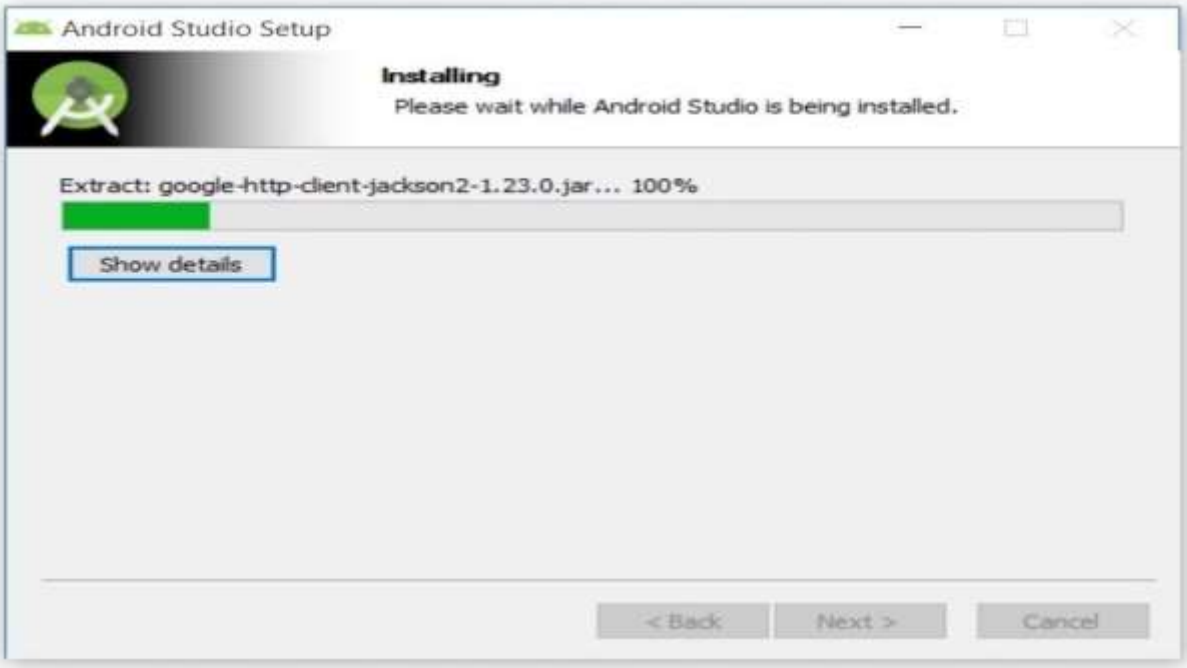
Keep the default settings. After clicking Next, Configuration Settings panel will be open, where you have to choose where to install Android Studio.



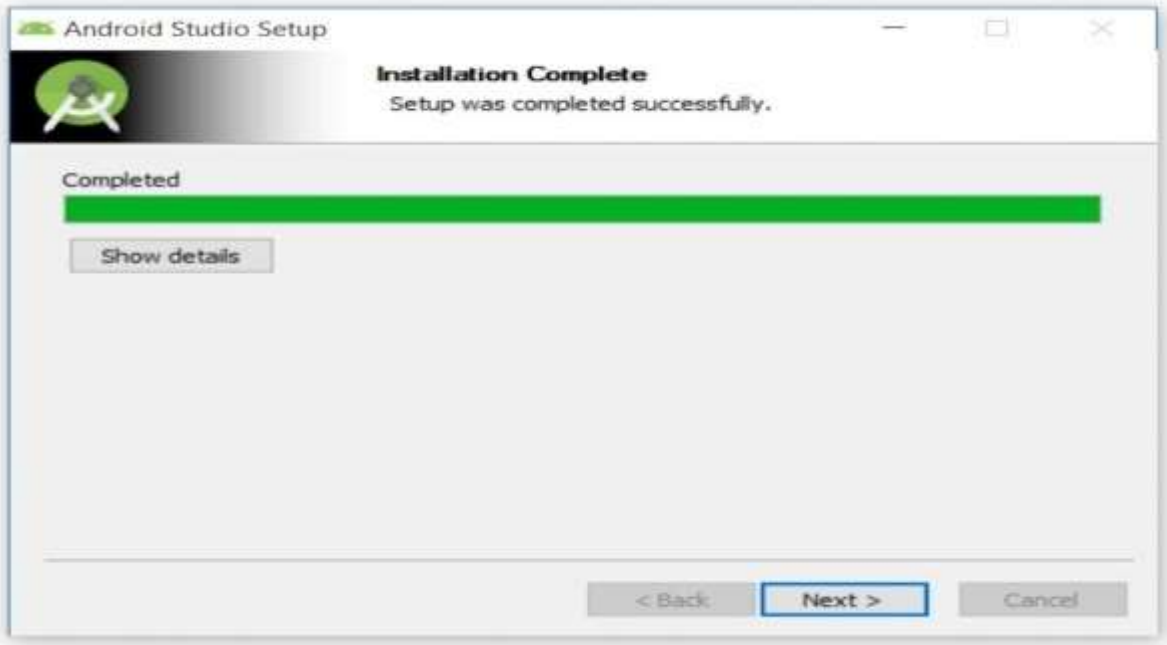
Keep the default installation location and click Next, this will open Choose Start Menu Folder panel.



Keep the default setting and click Install. The following Installing panel appeared:



Clicking Show details causes the names of files being installed and other activities to be displayed. When installation finished, the Installation Complete panel appeared.

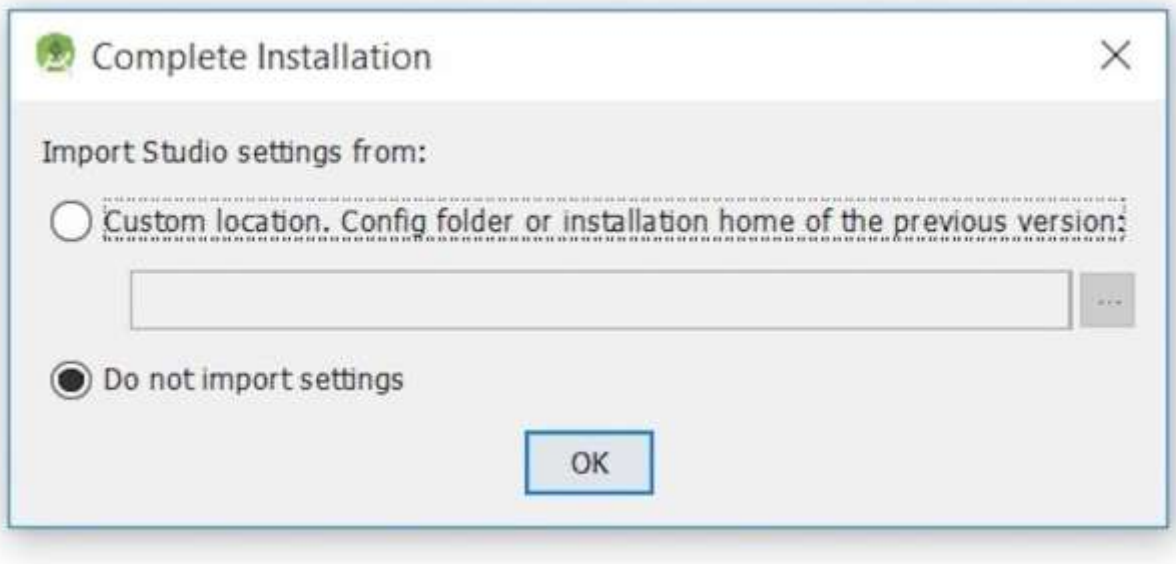


After clicking Next, the installer presented the Completing Android Studio Setup panel.

To complete the installation, I left the Start Android Studio box checked and clicked Finish.

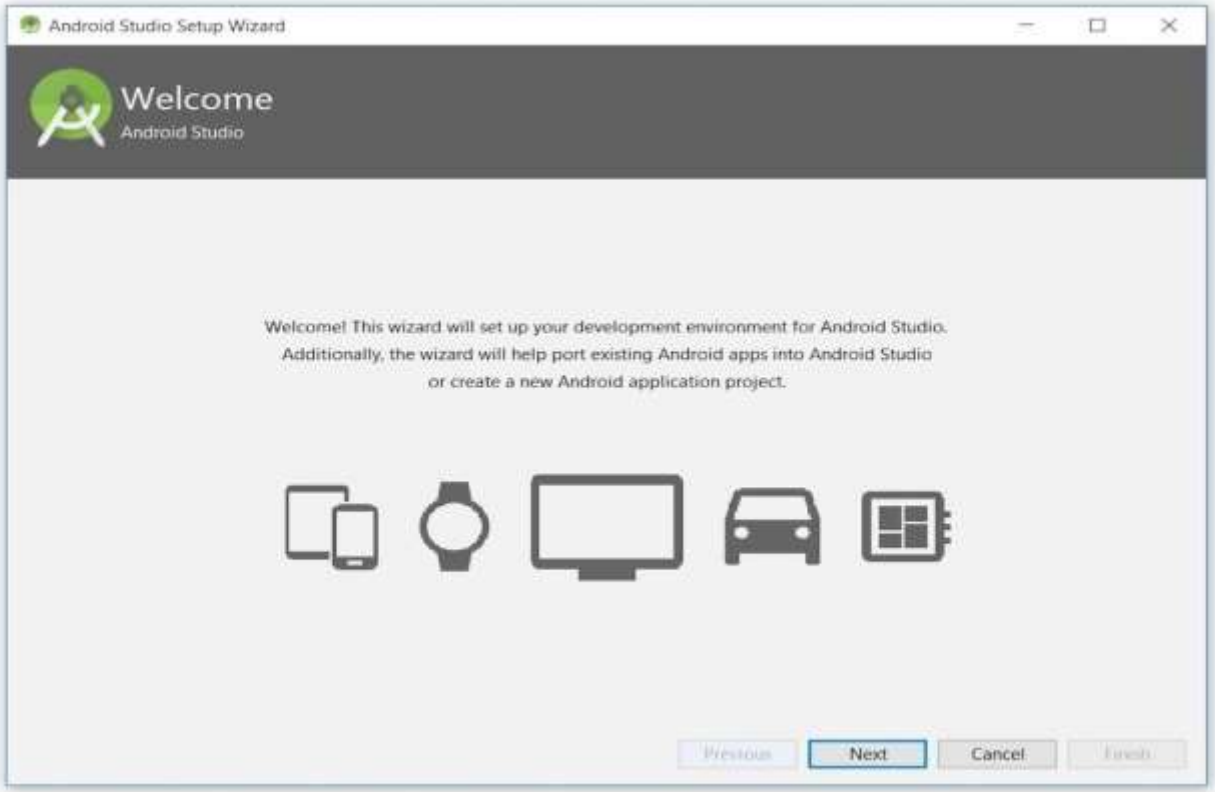
Running Android Studio

The first time Android Studio runs, it presents a Complete Installation dialog box that offers the option of importing settings from a previous installation.

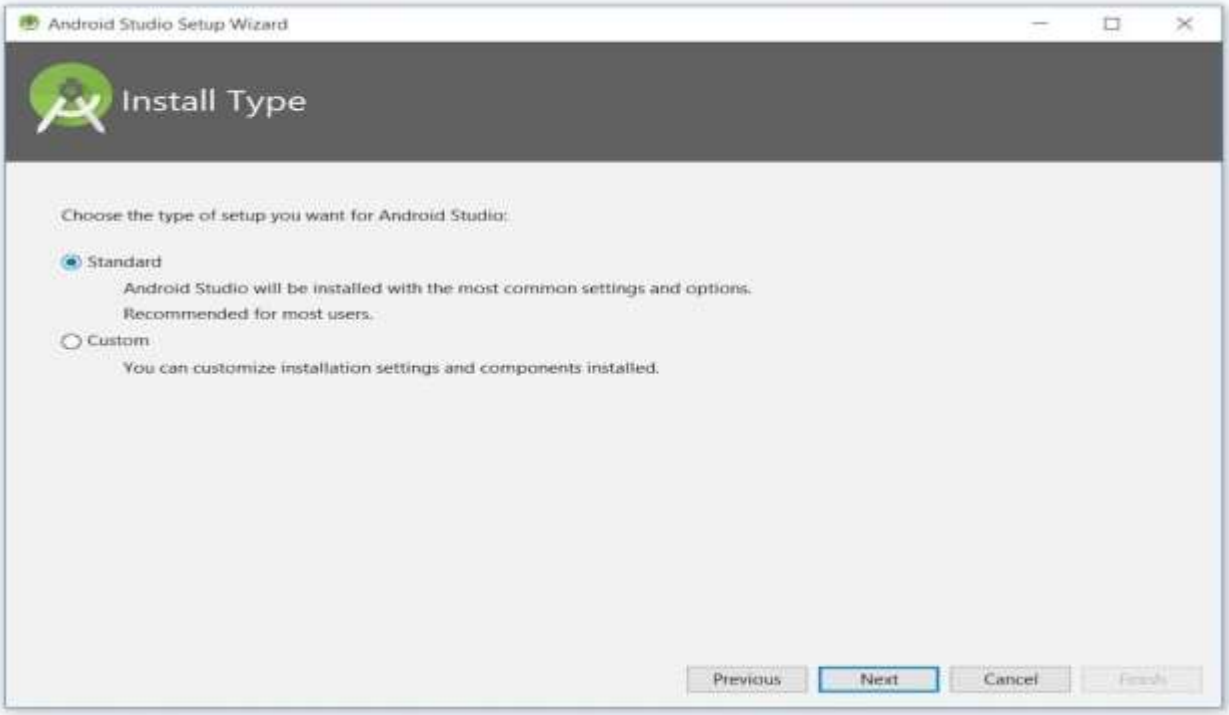


Choose not to import settings (the default selection) and clickOK, and the following splash screen will be displayed:

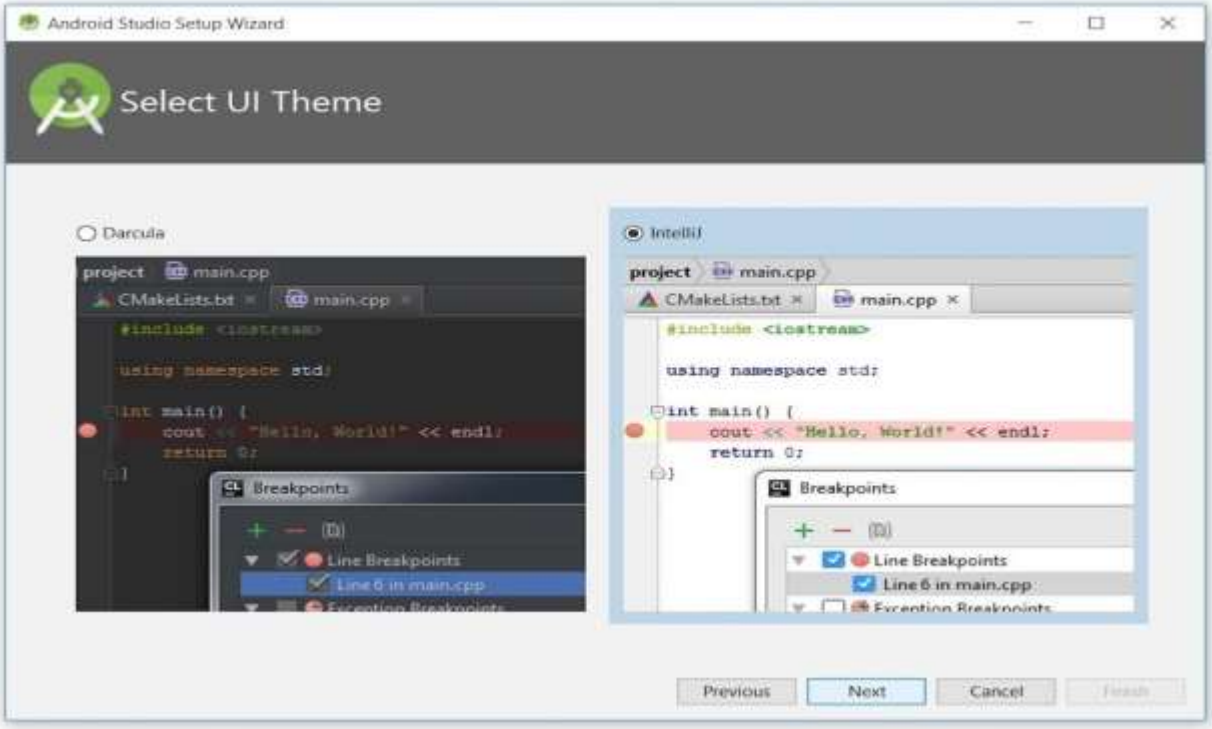
At this point, Android Studio presented the following Android Studio Setup Wizard dialog box:



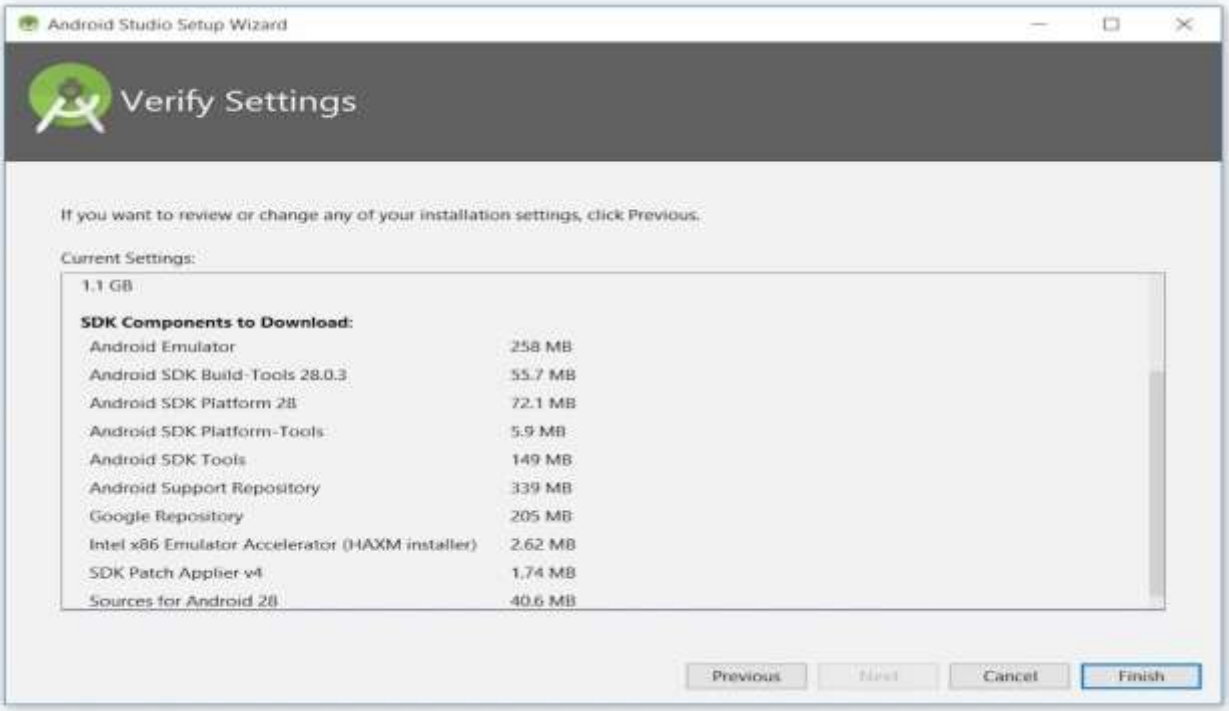
Click Next, and the wizard invited you to select an installation type. Keep the default standard setting.



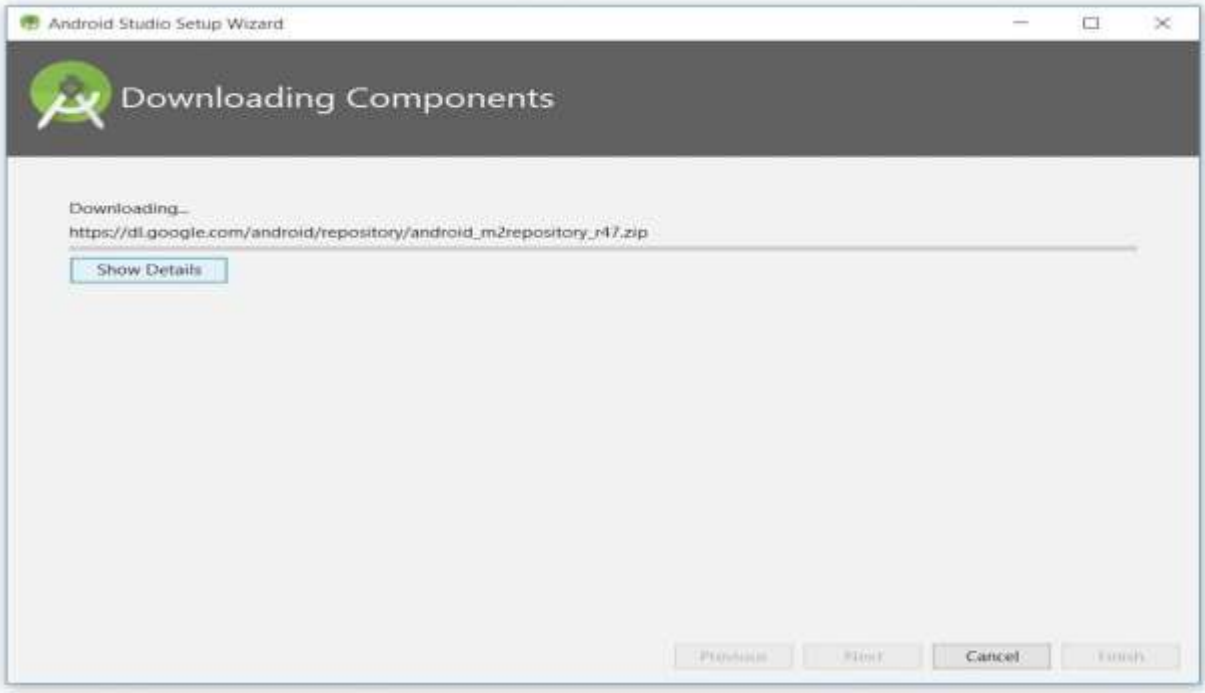
You will be directed to the page to choose a user interface theme.



Keep the default IntelliJ setting and click Next. Android Studio next provided the opportunity to verify settings.



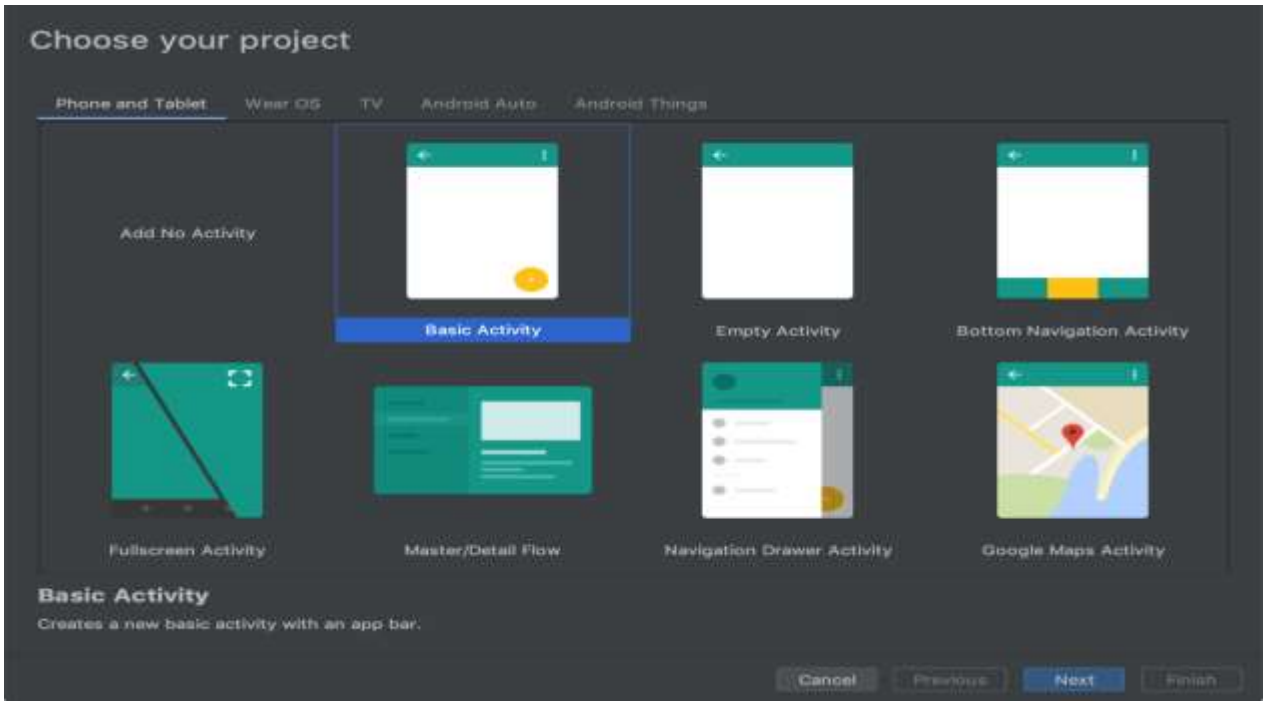
Click Finish and Android Studio began the process of downloading SDK components.



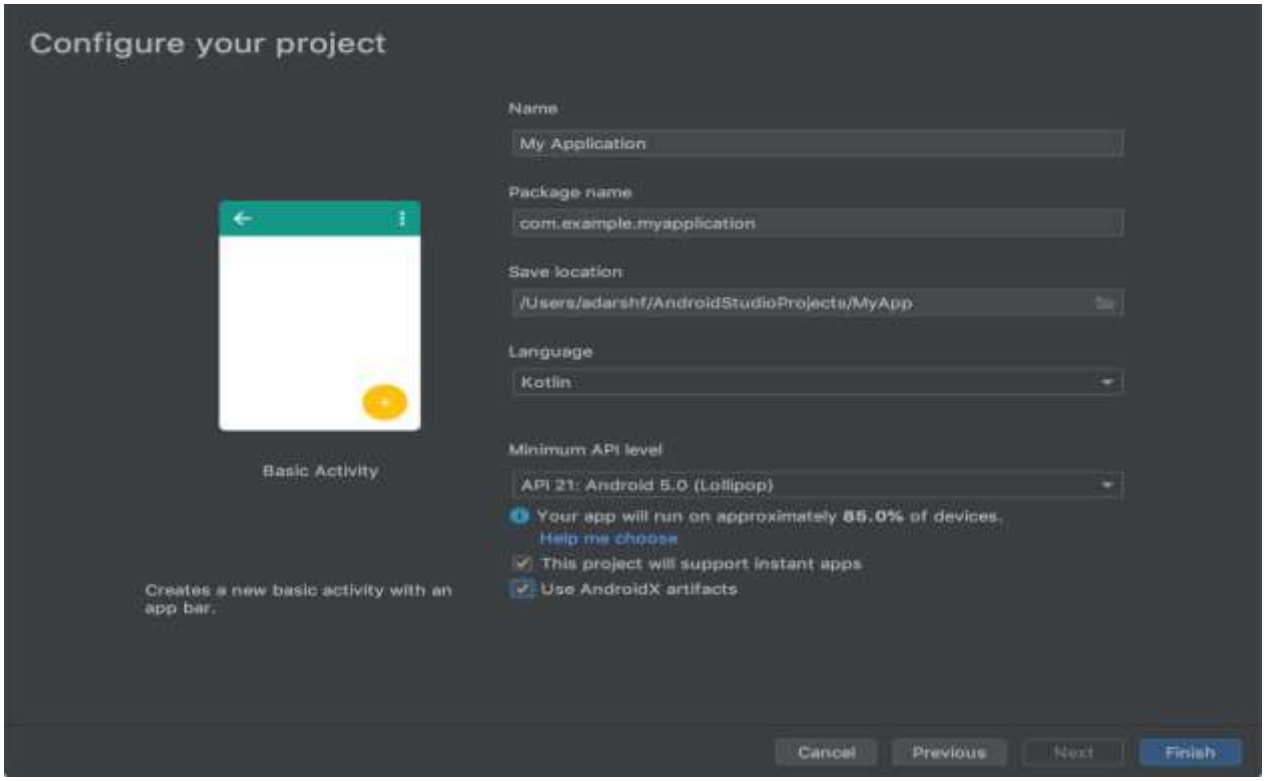
Finally, Click Finish to complete the wizard. The Welcome to Android Studio dialog box appeared.

Starting a new project

click Start a new Android Studio project. Android Studio will respond with the Create New Project dialog box shown in Figure



After you make a selection, click Next.



Select language as Java, name your activity and package and click on finish. This will launch the basic activity project.

RUN Application

To run your application in android studio you have two options.

1. either use a real device and connect it via USB cable .and run the application on it

2. or create a AVD in android(Tools->AVD manager-> create virtual device) and run application on this emulator.

Experiment-1

Aim: To develop a Simple Android Application that uses GUI components, Font and Colors

Procedure:

Creating a New project:

- 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.
- 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.

Code for 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">
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="30dp"
    android:gravity="center"
    android:text="Hello World!"
    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 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="Change color"
```

```
        android:textSize="25sp" />
</LinearLayout>
```

- **So now the designing part is completed.**

Java Coding for the Android Application:

- Click on app -> java -> com.example.exno1 -> MainActivity.

MainActivity.java

```
package com.example.exno1;

import android.graphics.Color;
import android.support.v7.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= (TextView) findViewById(R.id.textView);
Button b1= (Button) findViewById(R.id.button1);
b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        t.setTextSize(font);
        font = font + 5;
        if (font == 50)
            font = 30;
    }
});
Button b2= (Button) findViewById(R.id.button2);
b2.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;
      }
    });
  }
}
```

Experiment-2

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

Procedure:

Creating a New project:

- 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.
- It will take some time to build and load the project.
- After completion it will look as given below.

Creating Second Activity for the Android Application:

- Click on File -> New -> Activity -> Empty Activity.
- Type the Activity Name as Main2Activity and click Finish button.
- Thus Second Activity For the application is created.

Designing layout for the Android Application:

Designing Layout for Main Activity:

- Click on app -> res -> layout -> activity_main.xml.

Code for 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=".Login">
   <LinearLayout
      android:layout_width="match_paren
      t" android:layout_height="150dp">
   <TextView
      android:id="@+id/tv"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Login_details"
      android:gravity="center"
      android:textSize="30dp"
      android:layout_margin="30dp"/>
    </LinearLayout>
   <GridLayout
      android:id="@+id/gridl
      "
      android:layout_height="match_parent"
```

```
android:layout_width="match_parent"
android:columnCount="2"
android:rowCount="3"
android:layout_marginTop="100dp"
android:layout_marginBottom="200dp">
<TextView
   android:id="@+id/utext
   "
   android:layout_width="wrap_content
   "
   android:layout_height="wrap_conte
   nt" android:text="username"
   android:layout_margin="20dp"
   android:layout_row="0"
   android:layout_column="0"
   android:textSize="20sp"/>
<TextView
   android:id="@+id/ptext
   "
   android:layout_width="wrap_content
   "
   android:layout_height="wrap_conte
   nt" android:text="password"
   android:layout_margin="20dp"
   android:layout_marginTop="5dp"
   android:layout_row="1"
```

```
    android:layout_column="0"
    android:textSize="20sp"/>
<EditText
    android:id="@+id/uet
    "
    android:layout_width="wrap_content"
    android:layout_height="wrap_content
    " android:layout_margin="20dp"
    android:layout_row="0"
    android:layout_column="1"
    android:hint="type username here"
    android:inputType="textWebEditText"
    />
<EditText
    android:id="@+id/pet
    "
    android:layout_width="wrap_content
    "
    android:layout_height="wrap_conte
    nt" android:layout_margin="20dp"
    android:layout_row="1"
    android:layout_column="1"
    android:hint="type password here"
    android:inputType="textPassword"/>
```

```
    <Button
        android:layout_width="wrap_content
        "
        android:layout_height="wrap_conte
        nt" android:id="@+id/button"
        android:layout_marginLeft="30d
        p" android:layout_row="2"
        android:layout_column="0"
        android:text="submit details"/>
  <Button
     android:layout_width="wrap_content"
     android:layout_height="wrap_content
     " android:id="@+id/button1"
     android:layout_marginLeft="70dp"
     android:layout_row="2"
     android:layout_column="1"
     android:text="reset" />
  </GridLayout>
</RelativeLayout>
```

- So now the designing part of Main Activity is completed

Designing Layout for Second Activity:

- Click on app -> res -> layout -> activity_second.xml.

Code for Activity_second.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_pare
nt" tools:context=".Main2Activity">
<Button
    android:layout_width="wrap_content
    "
    android:layout_height="wrap_conten
    t" android:id="@+id/bt"
    android:layout_gravity="center"
    android:text="LOGOUT"/>

<TextView
    android:layout_width="wrap_content
    "
    android:layout_height="wrap_conten
    t" android:id="@+id/tv1"
    android:layout_margin="40dp"
    android:text="new text"
    android:textSize="30sp"/>
</LinearLayout>
```

- So now the designing part of Second Activity is also completed

Java Coding for the Android

Application: <u>Java Coding for Login:</u>

- Click on app -> java -> com.garima.login -> Login.

Code for Login.java:

```
package com.garima.login;
import
androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import
android.widget.EditText;
import android.widget.Toast;
public class Login extends
   AppCompatActivity { EditText
   username,password;
   Button l_button,r_button;

   @Override
   protected void onCreate(Bundle
      savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      username=(EditText)findViewById(R.id.uet);
      password=(EditText)findViewById(R.id.pet);
      l_button=(Button)findViewById(R.id.button);
      r_button=(Button)findViewById(R.id.button1);
```

```
l_button.setOnClickListener(new View.OnClickListener()
  { @Override
  public void onClick(View v) {
    if(username.getText().toString().equals("admin")&&
password.getText().toString().equals("1234"))
    {
      Toast t=
      Toast.makeText(Login.this,"redirecting",Toast.LENGTH_SHORT);
      t.show();
      Intent i=new Intent(Login.this,Main2Activity.class);
      startActivity(i);
    }
    else {
      Toast.makeText(Login.this, "User and Password is not correct",
          Toast.LENGTH_SHORT).show();
    }
  }
});
r_button.setOnClickListener(new
  View.OnClickListener() { @Override
  public void onClick(View
    v) {
    username.setText(null);
    password.setText(null);
    //finish();
  }
```

```
    });
  }
}
```

- So now the Coding part of Main Activity is completed.

Java Coding for Second Activity:

- Click on app -> java -> com.garima.login -> Main2Activity.

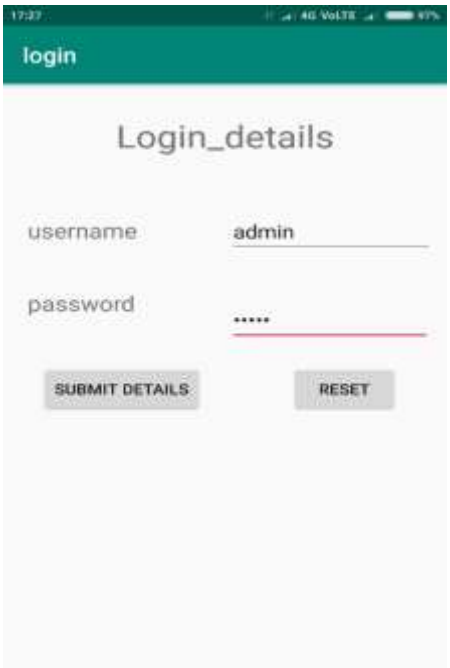
Code for Main2Activity.java:

```
package com.garima.login;
import
androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import
android.widget.TextView;
public class Main2Activity extends
  AppCompatActivity { TextView t1;
  Button b1;

  @Override
  protected void onCreate(Bundle
    savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    t1=(TextView)findViewById(R.id.tv1);
```

```
        Intent i1=getIntent();
        t1.setText("hello in second
        page");
        b1=(Button)findViewById(R.id.b
        t);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i=new Intent(Main2Activity.this,Login.class);
                startActivity(i);
                finish();
            }
        });
    }
}
```

- So now the Coding part of Second Activity is also completed.
- Now run the application to see the output.

OUTPUT



Experiment-3

Design simple GUI application with activity and intents e.g. calculator

Procedure:

Creating a New project:

- Open Android Stdio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as “ex.no.3″ , select the Minimum SDK and select language as Java click Finish.

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_main.xml.

Code for 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>
```

So now the designing part is completed.

Java Coding for the Android Application:

Click on app -> java -> com.example.exno3 -> MainActivity. And write the following code

Code for MainActivity.java:

```
package com.example.devang.exno3;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements OnClickListener
{
    //Defining the Views
    EditText Num1;
    EditText Num2;
    Button Add;
    Button Sub;
    Button Mul;
    Button Div;
    TextView Result;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Referring the Views
        Num1 = (EditText) findViewById(R.id.editText1);
        Num2 = (EditText) findViewById(R.id.editText2);
        Add = (Button) findViewById(R.id.Add);
```

```
        Sub = (Button) findViewById(R.id.Sub);
        Mul = (Button) findViewById(R.id.Mul);
        Div = (Button) findViewById(R.id.Div);
        Result = (TextView) findViewById(R.id.textView);

        // set a listener
        Add.setOnClickListener(this);
        Sub.setOnClickListener(this);
        Mul.setOnClickListener(this);
        Div.setOnClickListener(this);
    }
        @Override
    public void onClick (View v)
    {
        float num1 = 0;
        float num2 = 0;
        float 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
// write operation into oper, we will use it later for output
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);
    }
}
```

- So now the Coding part is also completed.
- Now run the application to see the output.

Output::


91% 11:05
ex.no.3
10
5
+
-
*
/
10.0 + 5.0 = 15.0

Experiment-4

Aim: Develop an application that makes use of RSS Feed.

Procedure:

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as "ex.no.6″, select the Minimum SDK and select language as Java then click Finish.

Designing layout for the Android Application:

Click on app -> res -> layout -> activity_main.xml

Code for Activity_main.xml:

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

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

   android:layout_width="fill_parent"

   android:layout_height="fill_parent"

   android:orientation="vertical" >

    <ListView

      android:id="@+id/listView"

      android:layout_width="match_parent"

      android:layout_height="wrap_content" />

 </LinearLayout>
```

So now the designing part is completed.

Adding permissions in Manifest for the Android Application:

- Click on app -> manifests -> AndroidManifest.xml
- Now include the INTERNET permissions in the AndroidManifest.xml file as shown below

Code for AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.exno6" >
     <uses-permission android:name="android.permission.INTERNET"/>
     <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                 <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>


</manifest>
```

So now the Permissions are added in the Manifest.

Java Coding for the Android Application:

Click on app -> java -> com.example.exno6 -> MainActivity.

Code for MainActivity.java:

```
package com.example.exno6;
```

```
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ListActivity
{
    List headlines;
    List links;
    @Override
    protected void onCreate(Bundle savedInstanceState)
```

```
{
    super.onCreate(savedInstanceState);
    new MyAsyncTask().execute();
}
 class MyAsyncTask extends AsyncTask<Object,Void,ArrayAdapter>
{
    @Override
    protected ArrayAdapter doInBackground(Object[] params)
    {
        headlines = new ArrayList();
        links = new ArrayList();
        try
        {
            URL url = new URL("https://codingconnect.net/feed");
            XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
            factory.setNamespaceAware(false);
            XmlPullParser xpp = factory.newPullParser();

            // We will get the XML from an input stream
            xpp.setInput(getInputStream(url), "UTF_8");
            boolean insideItem = false;

            // Returns the type of current event: START_TAG, END_TAG, etc..
            int eventType = xpp.getEventType();
            while (eventType != XmlPullParser.END_DOCUMENT)
```

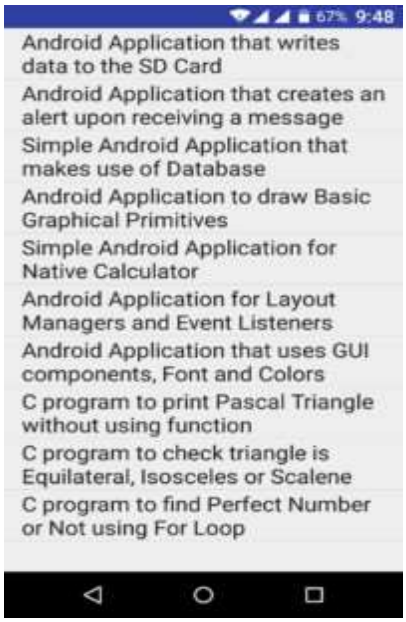
```
        {
            if (eventType == XmlPullParser.START_TAG)
            {
                if (xpp.getName().equalsIgnoreCase("item"))
                {
                    insideItem = true;
                }
                else if (xpp.getName().equalsIgnoreCase("title"))
                {
                    if (insideItem)
                        headlines.add(xpp.nextText()); //extract the headline
                }
                else if (xpp.getName().equalsIgnoreCase("link"))
                {
                    if (insideItem)
                        links.add(xpp.nextText()); //extract the link of article
                }
            }
            else if(eventType==XmlPullParser.END_TAG &&
xpp.getName().equalsIgnoreCase("item"))
            {
                insideItem=false;
            }
            eventType = xpp.next(); //move to next element
        }
```

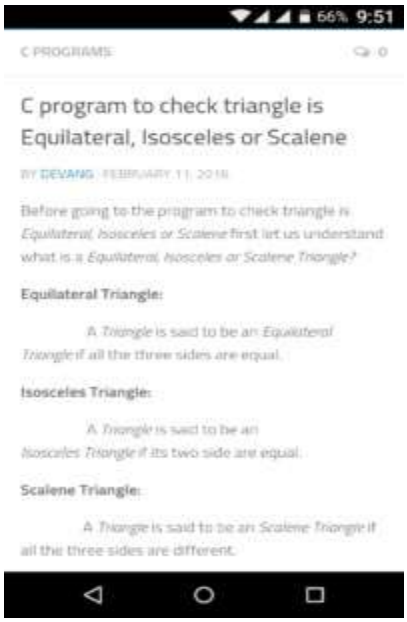
```
            }
            catch (MalformedURLException e)
            {
                e.printStackTrace();
            }
            catch (XmlPullParserException e)
            {
                e.printStackTrace();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
            return null;
        }
        protected void onPostExecute(ArrayAdapter adapter)
        {
            adapter = new ArrayAdapter(MainActivity.this, android.R.layout.simple_list_item_1,
headlines);
            setListAdapter(adapter);
        }
    }
     @Override
    protected void onListItemClick(ListView l, View v, int position, long id)
```

```
    {
        Uri uri = Uri.parse((links.get(position)).toString());
        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
        startActivity(intent);
    }

    public InputStream getInputStream(URL url)
    {
        try
        {
            return url.openConnection().getInputStream();
        }
        catch (IOException e)
        {
            return null;
        }
    }
}
```

- So now the Coding part is also completed.
- Now run the application to see the output.

Output


67% 9:48
Android Application that writes data to the SD Card
Android Application that creates an alert upon receiving a message
Simple Android Application that makes use of Database
Android Application to draw Basic Graphical Primitives
Simple Android Application for Native Calculator
Android Application for Layout Managers and Event Listeners
Android Application that uses GUI components, Font and Colors
C program to print Pascal Triangle without using function
C program to check triangle is Equilateral, Isosceles or Scalene
C program to find Perfect Number or Not using For Loop


66% 9:51
C PROGRAMS
C program to check triangle is Equilateral, Isosceles or Scalene
Before going to the program to check triangle is Equilateral, Isosceles or Scalene first let us understand what is a Equilateral, Isosceles or Scalene Triangle?
Equilateral Triangle:
A Triangle is said to be an Equilateral Triangle if all the three sides are equal.
Isosceles Triangle:
A Triangle is said to be an Isosceles Triangle if its two side are equal.
Scalene Triangle:
A Triangle is said to be an Scalene Triangle if all the three sides are different.

Experiment-5

Aim: Write an application that draws basic graphical primitives on the screen

Procedure:

Creating a New project:

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

Designing layout for the Android Application:

Click on app -> res -> layout -> activity_main.xml and write the following code.

Code for 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>
```

So now the designing part is completed.

Java Coding for the Android Application:

Click on app -> java -> com.example.exno5 -> MainActivity.

Code for MainActivity.java:

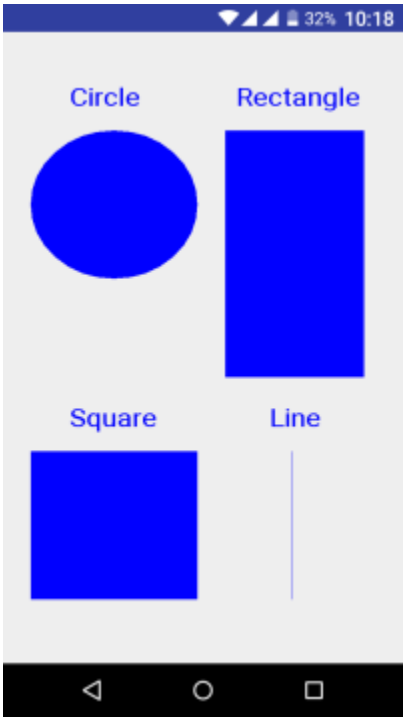
```
package com.example.exno5;
```

```
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 set its color & TextSize
        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setTextSize(50);

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

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

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

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

- So now the Coding part is also completed.
- Now run the application to see the output.

Output:



Experiment-6

Aim: Create an android app for database creation using SQlite Database

Procedure:

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as “databaseactivity ″, select the Minimum SDK and select language as Java then click Finish

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_main.xml and write the following code

Code for 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>
```

So now the designing part is completed.

Java Coding for the Android Application:

Click on app -> java -> com.garima.databaseactivity -> MainActivity and write the following code.

Code for MainActivity.java:

```
package com.garima.databaseactivity;
```

```
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;
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Rollno=(EditText)findViewById(R.id.Rollno);
      Name=(EditText)findViewById(R.id.Name);
```

```
        Marks=(EditText)findViewById(R.id.Marks);
        Insert=(Button)findViewById(R.id.Insert);
        Delete=(Button)findViewById(R.id.Delete);
        Update=(Button)findViewById(R.id.Update);
        View=(Button)findViewById(R.id.View);
        ViewAll=(Button)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)
    {
        // Inserting a record to the Student table
        if(view==Insert)
        {
            // Checking for empty fields
            if(Rollno.getText().toString().trim().length()==0||
```

```
            Name.getText().toString().trim().length()==0||
            Marks.getText().toString().trim().length()==0)
        {
           showMessage("Error", "Please enter all values");
           return;
        }
        db.execSQL("INSERT INTO student
VALUES('"+Rollno.getText()+"','"+Name.getText()+
            "','"+Marks.getText()+"');");
        showMessage("Success", "Record added");
        clearText();
     }
     // Deleting a record from the Student table
     if(view==Delete)
     {
        // Checking for empty roll number
        if(Rollno.getText().toString().trim().length()==0)
        {
           showMessage("Error", "Please enter Rollno");
           return;
        }
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
        if(c.moveToFirst())
        {
           db.execSQL("DELETE FROM student WHERE rollno='"+Rollno.getText()+"'");
```

```
            showMessage("Success", "Record Deleted");
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
        }
        clearText();
    }
    // Updating a record in the Student table
    if(view==Update)
    {
        // Checking for empty roll number
        if(Rollno.getText().toString().trim().length()==0)
        {
            showMessage("Error", "Please enter Rollno");
            return;
        }
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
        if(c.moveToFirst()) {
            db.execSQL("UPDATE student SET name='" + Name.getText() + "',marks='" +
Marks.getText() +
                    "' WHERE rollno='"+Rollno.getText()+"'");
            showMessage("Success", "Record Modified");
        }
        else {
```

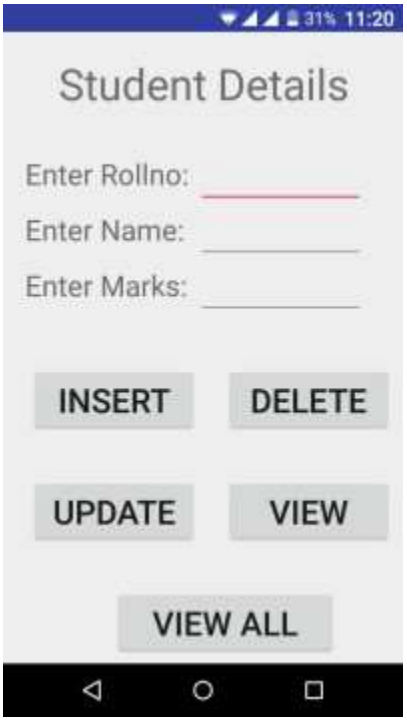
```
            showMessage("Error", "Invalid Rollno");
        }
        clearText();
    }
    // Display a record from the Student table
    if(view==View)
    {
        // Checking for empty roll number
        if(Rollno.getText().toString().trim().length()==0)
        {
            showMessage("Error", "Please enter Rollno");
            return;
        }
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
        if(c.moveToFirst())
        {
            Name.setText(c.getString(1));
            Marks.setText(c.getString(2));
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
            clearText();
        }
```

```
    }
    // Displaying all the records
    if(view==ViewAll)
    {
        Cursor c=db.rawQuery("SELECT * FROM student", null);
        if(c.getCount()==0)
        {
            showMessage("Error", "No records found");
            return;
        }
        StringBuffer buffer=new StringBuffer();
        while(c.moveToNext())
        {
            buffer.append("Rollno: "+c.getString(0)+"\n");
            buffer.append("Name: "+c.getString(1)+"\n");
            buffer.append("Marks: "+c.getString(2)+"\n\n");
        }
        showMessage("Student Details", buffer.toString());
    }
}
public void showMessage(String title,String message)
{
    Builder builder=new Builder(this);
    builder.setCancelable(true);
    builder.setTitle(title);
```

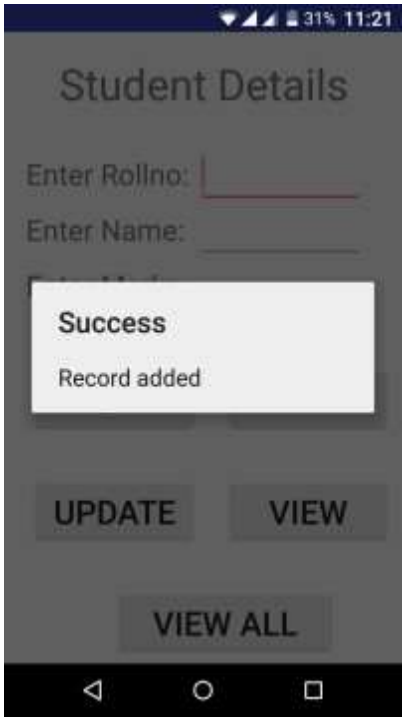
```
        builder.setMessage(message);
        builder.show();
    }
    public void clearText()
    {
        Rollno.setText("");
        Name.setText("");
        Marks.setText("");
        Rollno.requestFocus();
    }
}
```

- So now the Coding part is also completed.
- Now run the application to see the output.

Output::




31% 11:21
Student Details
Enter Rollno: 1
Enter Name: devang
Enter Marks: 95
INSERT
DELETE
UPDATE
VIEW
VIEW ALL


31% 11:21
Student Details
Enter Rollno:
Enter Name:
Success
Record added
UPDATE
VIEW
VIEW ALL


31% 11:22
Student Details
Student Details
Rollno: 1
Name: devang
Marks: 95
Rollno: 2
Name: akhil
Marks: 90
Rollno: 3
Name: abdul
Marks: 80
VIEW ALL

Experiment-7

Aim: Develop a native application that uses GPS location information

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Map Activity and click Next.
- Then type the Application name as "MapsActivity", select the Minimum SDK and select language as Java then click Finish

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_maps.xml and write the following code

Activity_maps.xml

```
<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:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="hello" />
</RelativeLayout>
```

So now the designing part is completed.

Adding permissions in Manifest for the Android Application:

- Click on app -> manifests -> AndroidManifest.xml

- Now include the ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, INTERNET permissions in the AndroidManifest.xml file as shown below

Code for AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.garima.mapactivity"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17"        />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.garima.mapactivity.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
  </application>
</manifest>
```

So now the Permissions are added in the Manifest.

Java Coding for the Android Application:

- Click on app -> java -> com.garima.mapactivity -> MainActivity.

Code for MainActivity.java:

```
package com.garima.mapactivity;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import  android.location.LocationListener;
import android.location.LocationManager;
import android.widget.TextView;
import android.util.Log;

public class MainActivity extends Activity implements LocationListener{
   protected LocationManager locationManager;
   protected LocationListener locationListener;
   protected Context context;
   TextView txtLat;
```

```
    String lat;
    String provider;
    protected String latitude,longitude;
    protected boolean gps_enabled,network_enabled;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        txtLat = (TextView) findViewById(R.id.textview1);
        locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
    }
    @Override
    public void onLocationChanged(Location location) {
        txtLat = (TextView) findViewById(R.id.textview1);
        txtLat.setText("Latitude:" + location.getLatitude() + ", Longitude:" +
location.getLongitude());
    }
    @Override
    public void onProviderDisabled(String provider) {
        Log.d("Latitude","disable");
    }
    @Override
    public void onProviderEnabled(String provider) {
        Log.d("Latitude","enable");
```

```
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        Log.d("Latitude","status");
    }
}
```

Output:

Experiment-8

Aim: Implement an application that writes data on the SD card.

Procedure:

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as "ex.no.8", select the Minimum SDK and select language as Java then click Finish

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_main.xml and write the following code

Code for 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:layout_margin="20dp"
    android:orientation="vertical">
     <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:textSize="30dp" />
     <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Write Data"
    android:textSize="30dp" />
  <Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Read data"
    android:textSize="30dp" />
  <Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Clear"
    android:textSize="30dp" />
</LinearLayout>
```

So now the designing part is completed.

Adding permissions in Manifest for the Android Application:

- Click on app -> manifests -> AndroidManifest.xml
- Now include the WRITE_EXTERNAL_STORAGE permissions in the AndroidManifest.xml file as shown below

Code for AndroidManifest.xml:

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.exno9" >
     <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
     <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                 <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

So now the Permissions are added in the Manifest.

Java Coding for the Android Application:

- Click on app -> java -> com.example.exno8 -> MainActivity.

Code for MainActivity.java:

```
package com.example.exno8;
 import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
 public class MainActivity extends AppCompatActivity
{
    EditText e1;
    Button write,read,clear;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         e1= (EditText) findViewById(R.id.editText);
        write= (Button) findViewById(R.id.button);
        read= (Button) findViewById(R.id.button2);
        clear= (Button) findViewById(R.id.button3);

        write.setOnClickListener(new View.OnClickListener()
        {
```

```
        @Override
        public void onClick(View v)
        {
            String message=e1.getText().toString();
            try
            {
                File f=new File("/sdcard/myfile.txt");
                f.createNewFile();
                FileOutputStream fout=new FileOutputStream(f);
                fout.write(message.getBytes());
                fout.close();
                Toast.makeText(getBaseContext(),"Data Written in
SDCARD",Toast.LENGTH_LONG).show();
            }
            catch (Exception e)
            {
                Toast.makeText(getBaseContext(),e.getMessage(),Toast.LENGTH_LONG).show();
            }
        }
    });


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

```
        {
            String message;
            String buf = "";
            try
            {
                File f = new File("/sdcard/myfile.txt");
                FileInputStream fin = new FileInputStream(f);
                BufferedReader br = new BufferedReader(new InputStreamReader(fin));
                while ((message = br.readLine()) != null)
                {
                    buf += message;
                }
                e1.setText(buf);
                br.close();
                fin.close();
                Toast.makeText(getBaseContext(),"Data Recived from
SDCARD",Toast.LENGTH_LONG).show();
            }
            catch (Exception e)
            {
                Toast.makeText(getBaseContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
            }
        }
    });
```

```
        clear.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                e1.setText("");
            }
        });
    }
}
```

So now the Coding part is also completed.

Now run the application to see the output.

OUTPUT:


56% 12:32
ex.no.9
hello!!!
WRITE DATA
READ DATA
CLEAR
Data Written in SDCARD

Experiment-9

Aim: Design a StopWatch application

Procedure:

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as "stopwatch", select the Minimum SDK and select language as Java then click Finish

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_main.xml and write the following code

Code for 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">
   <TextView
      android:layout_width="wrap_content
      "
      android:layout_height="wrap_conte
      nt" android:text="Hello World!"
      android:id="@+id/tv1"
      android:layout_marginTop="50dp"
      android:layout_marginLeft="150dp"
      android:gravity="center"
```

```
   android:textSize="30sp"/>
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/b1"
   android:text="START"
   android:layout_marginTop="200dp"
   android:layout_marginLeft="150dp"
   android:onClick="onClickStart"/>

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/b2"
   android:text="STOP"
   android:layout_marginTop="300dp"
   android:layout_marginLeft="150dp"
   android:onClick="onClickStop"/>
<Button
   android:layout_width="wrap_content
   "
   android:layout_height="wrap_conte
   nt" android:id="@+id/b3"
   android:text="RESET"
   android:layout_marginTop="400dp"
   android:layout_marginLeft="150dp"
```

```
    android:onClick="onClickReset"/>
</RelativeLayout>
```

So now the designing part is completed.

Java Coding for the Android Application:

Click on app -> java -> com.garima.stopwatch -> MainActivity.

Code for MainActivity.java:

```
package com.garima.stopwatch;
import androidx.appcompat.app.AppCompatActivity;

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

public class MainActivity extends
  AppCompatActivity { private int seconds=0;
  boolean
  running,wasRunning;
  TextView txtv;
  Handler hl;

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

```
if(savedInstanceState!=null)
```

```
    {
        seconds=savedInstanceState.getInt("seconds");
        running=savedInstanceState.getBoolean("running");
        wasRunning=savedInstanceState.getBoolean("wasRun
        ning");
    }
    runTimer();
}
protected void onPause() {

    super.onPause();
    wasRunning=runnin
    g; running=false;
}

protected void onResume()
{
    super.onResume
    ();
    if(wasRunning)
        running=true;

}
@Override
public void onSaveInstanceState(Bundle savedInstancestate)
{
    savedInstancestate.putInt("seconds",seconds);
```

```
    savedInstancestate.putBoolean("running",running);
    savedInstancestate.putBoolean("wasRunning",wasRun
    ning);
}
public void onClickStart(View v)
{
    running=true;
}
public void onClickStop(View v)
{
    running=false;
}
public void onClickReset(View v)
{
    running=fals
    e;
    seconds=0;
}
private void runTimer() {
    txtv=(TextView)findViewById(R.id.tv1);
    hl=new Handler();
    hl.post(new
        Runnable() {
        @Override
        public void run() {
            int hours=seconds/3600;
            int
```

```
minutes=(seconds%3600)/60;
int secs=seconds%60;
```

```
            String
            time=String.format("%d:%02d:%02d",hours,minutes,secs);
            txtv.setText(time);
            if(running)
            {
               seconds++;
            }
           hl.postDelayed(this,100) ;
         }
      });
   }
}
```

OUTPUT

Experiment-10

Aim: Create an application to play video in android.

Procedure:

Creating a New project:

- Open Android Studio and then click on File -> New -> New project.
- Then select the Empty Activity and click Next.
- Then type the Application name as "videoact", select the Minimum SDK and select language as Java then click Finish

Designing layout for the Android Application:

- Click on app -> res -> layout -> activity_main.xml and write the following code

Code for Activity_main.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:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.garima.videoact.MainActivity">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <Button
            android:text="Play"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
        android:id="@+id/btnPlay"
        android:onClick="playVideo" />
    <VideoView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/videoView" />
  </LinearLayout>
</RelativeLayout>
```

So now the designing part is completed.

Java Coding for the Android Application:

Click on app -> java -> com.garima.videoact -> MainActivity.

Code for MainActivity.java:

```
package com.garima.videoact;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.MediaController;
import android.widget.VideoView;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
  //create class reference
  VideoView vid;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        vid = (VideoView)findViewById(R.id.videoView);
    }
    public void playVideo(View v) {
        MediaController m = new MediaController(this);
        vid.setMediaController(m);
        String path = "android.resource://com.garima.videoact/"+R.raw.gulabo;
        Uri u = Uri.parse(path);
        vid.setVideoURI(u);
        vid.start();
    }
}
```

Viva Voice

What is Android?

Android is a stack of software for mobile devices which includes an Operating System, middleware and some key applications. The application executes within its own process and its own instance of Dalvik Virtual Machine.

What is the Android SDK?

To develop a mobile application, Android developers require some tools and this requirement is satisfied by "Android SDK" which is a set of tools that are used for developing or writing apps. It has a Graphical User Interface that emulates the Android environment. This emulator acts like an actual mobile device on which the developers write their code and then debug/test the same code to check if anything is wrong.

What are the different versions of Android OS that you remember?

Given below are the various versions of Android.

Version	Name
Android 8.0	Oreo
Android 7.0 – 7.1.2	Nougat
Android 6 – 6.0.1	Marshmallow
Android 5 – 5.1.1	Lollipop
Android 4.4 – 4.4.4	KitKat
Android 4.1 – 4.3	Jelly Bean
Android 4.0-4.0.4	Ice Cream Sandwich

What is the difference between Mobile Application Testing and Mobile Testing?

Mobile app testing is the testing of applications on a device which mainly focuses on functions and features of the application. And Mobile Testing is the testing of the actual mobile device and focuses on mobile features like Call, SMS, Contacts, Media Player, inbuilt browsers, etc.

Name the languages supported for Android development.

Java is the widely used language for Android development. It also supports Kotlin and when used with Android SDK, it improves the performance speed too.

What are the advantages of the Android Operating System?

It is open-source and platform-independent. It supports various technologies like Bluetooth, Wi-Fi, etc

Which components are necessary for a New Android project?

Whenever a new Android project is created, the below components are required:

manifest: It contains an XML file.
build/: It contains build output.
src/: It contains the code and resource files.
res/: It contains bitmap images, UI Strings and XML Layout i.e. all non-code resources.
assets/: It contains a file that should be compiled into a .apk file.

Provide the important core components of Android.

The core components of Android operating systems are:

- Activity
- Intents
- Services
- Content Provider
- Fragment

Explain briefly – what is meant by Activities?

Activities are the part of the mobile app which the user can see and interact with. For Example, if you open an SMS app which has multiple activities like create new SMS, add a contact from the address book, write the content in the SMS body, send SMS to the selected contact, etc.Activity keeps a track of the following:

- Keeps track of what a user is currently looking for in an app.
- Keeps a track of previously used processes, so that the user can switch between ongoing process and previous process.
- It helps to kill the processes so that the user can return to their previous state

An activity is implemented as a subclass of Activity class as shown below:

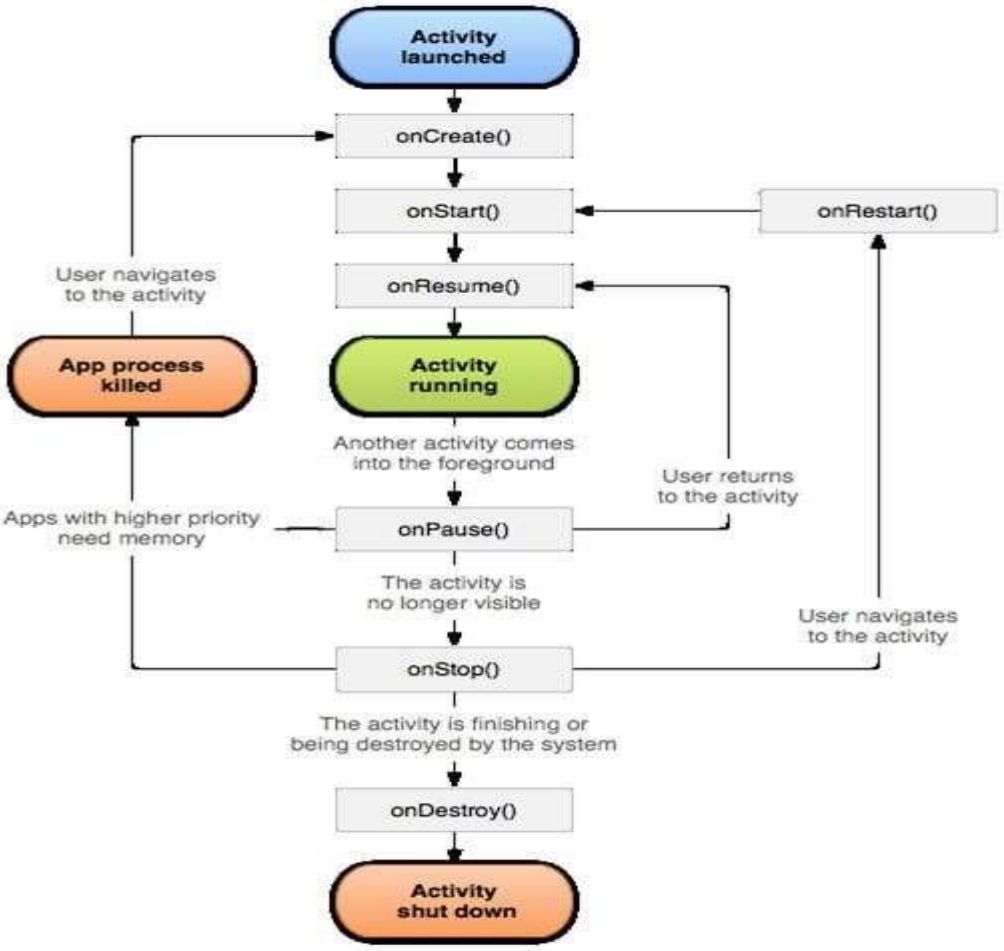
```
Public class MyActivity extends Activity
{
}
```

Explain Activity Lifecycle briefly.

When a user interacts with the app and moves here and there, out of the app, returns to the app, etc. During all this process "Activity" instances also move in the different stages in their lifecycle.

There are seven different states like – onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop(), and onDestroy(). These are termed as a 'callback'. Android system invokes these callbacks to know that the state has been changed.

The below-given diagram describes the Activity Lifecycle:



What is an Intent?

Android has an Intent class when the user has to navigate from one activity to another. Intent displays notifications from the device to the user and then the user can respond to the notification if required.

Given below are the two types:

- Implicit Intents
- Explicit Intents

Explain Implicit and Explicit Intents.

Implicit Intent calls the system components while explicit Intents invoke the Activity class.

What is the importance of setting up permission in app development?

Once the permissions are set for the app development, then the data and code are restricted to the authorized users only. If the code is kept without any restriction or if it is accessible to anyone then there are chances of compromise of code which results in defect leakage.

What is .apk extension in Android?

It is a default file format that is used by the Android Operating System. Application Package Kit (APK) is used for the installation of mobile apps. The .apk contains resource file, certificate, manifest file, and other code. APK files are archive files in the zip format with .apk extension.

What is the database used for the Android platform?

SQLite is the database that is used for the Android platform. It is an open-source, serverless database.

What is ANR in Android?

ANR stands for Application Not Responding. It is a notification or pop-up displayed by the Android platform whenever the application is performing too many functions at a time and if it is suddenly not responding for a long time to the user action.

Which are the dialog boxes supported by the Android platform?

Android supports four types of dialog boxes:

- **AlertDialog**: It has a maximum of 3 buttons and sometimes AlertDialog includes checkboxes and Radio buttons to select the element.
- **ProgressDialog**: It displays the progress bar or wheels.
- **TimePickerDialog**: Using this dialog box, a user selects the Time.
- **DatePickerDialog**: Using this dialog box, a user selects the Date

What is ADB?

Answer: Android Debug Bridge (ADB) is a command-line tool that performs shell commands. ADB is used for direct communication between the emulator ports. It gives direct control of the communication between the emulator instances to the developer.

What is ActivityCreator?

ActivityCreator is a batch file and shell script which was used to create a new Android project. It is now replaced by the "Create New Project" in Android SDK.

What is Orientation?

Answer: Orientation is the key feature in Smartphones nowadays. It has the ability to rotate the screen between Horizontal or Vertical mode.

Android supports two types of screen Orientations as mentioned below:

- **Portrait**: When your device is vertically aligned.
- **Landscape**: When your device is horizontally aligned.

setOrientation() is a method using which you can set a screen alignments. HORIZONTAL and VERTICAL are two values that can be set in the setOrientation() method. Whenever there is a change in the display orientation i.e. from Horizontal to Vertical or vice versa then onCreate() method of the Activity gets fired.

Basically, when the orientation of the Android mobile device gets changed then the current activity gets destroyed and then the same activity is recreated in the new display orientation. Android developers define the orientation in the AndroidManifest.xml file.

What is AIDL?

In the Android platform, there are remote methods that facilitate the use of methods from one program to another. To create and implement the remote methods the first step is to define the communication interface in AIDL.

AIDL stands for Android Interface Definition Language. It facilitates communication between the client and the service. It also communicates the information through inter-process communication. For communication between processes, the data is broken down into chunks which are easily understandable by the Android platform.

What are the data types supported by AIDL?
Data Types supported by AIDL are as follows:

- String
- List
- Map
- charSequence
- Java data types such as INT, Long, Char, Boolean, etc

Explain the AndroidManifest.xml file and why do you need this?
Every application must have an AndroidManifest.xml file in the root directory. It contains information about your app and provides the same to the Android system.
The information includes the package name, Android components such as Activity, Services, Broadcast Receivers, Content Providers, etc. Every Android system must have this information before running any app code.

AndroidManifest.xml file performs the following tasks:

- It provides a name to the Java package and this name is a unique identifier for the application.
- It describes the various components of the application which include Activity, Services, Content Providers, etc. Also, it defines the classes which implement these components.
- It is responsible to protect the application and it declares the permission for accessing the protected part of the app.
- It also declares the Android API which is going to be used by the application.
- It contains the library file details which are used and linked to the application.

What all devices have you worked on?
There are many mobile devices available in the market with different operating systems.
Specifically, I have worked on Android, Windows, Symbian, iPhone, etc

Which tools are used for debugging on the Android platform?
To understand the cause of the failure or cause of any issue, debugging is important. On the Android platform **Android Monitor.bat** utility is used while on the iOS platform, iPhone Configuration utility is used for debugging purposes.
There are different tools for debugging which include – Android DDMS, Android Debug Bridge, iOS simulator, Debugging from Eclipse with ADT, Remote debugging on Android with Chrome, etc.

Which scenario can test only on real devices but not on an emulator?
Emulators are used for performing similar kinds of testing which is performed on the real devices. Basically, emulators are used as a replacement for real devices as sometimes real

devices are not available for testing, the use of real mobile devices for testing purposes is costlier at times.
But there are few scenarios that cannot be tested using emulator, these can be tested only using real devices. These scenarios are interrupted scenarios i.e. message, phone call interruption while using the app, low battery, Bluetooth, memory card mount and unmount, etc.

How do you troubleshoot the android application which is crashing frequently?
Given below are the few steps that we need to follow while troubleshooting the crashing issue:

- **Free up memory space**: There is only limited space available on mobile devices for mobile apps. To avoid crashing issues or memory-related issues, you need to first check the memory space.
- **Clear app data usage**: You can clear the app data using the Application Manager under "Settings". This will clear the cache memory and allow some free space to install another app or it will boost up your current app.
- **Memory Management**: Some apps run perfectly on one type of mobile device but the same app may not work on another type of device as for such devices the processing power, memory management, and CPU speed is different. For any app to run properly on any type of mobile device, you should manage the memory on the device.
- **Compatibility issue**: It is always not possible to test mobile apps on all mobile devices, browsers, operating systems, etc. So you need to test your mobile app on as many mobile devices as you can in order to avoid any compatibility issue.

How do you find memory leaks in the mobile app on the Android platform?
Android Studio is using Android Device Manager (ADM), this ADM is used to detect the memory leaks in the Android platform.
When you open ADM in the Android Studio then on the left-hand side of the ADM, you will find your device or emulator in which a heap sign will be displayed. When you are running any mobile app then you will see the heap size, memory analysis and other statistics displayed on it.

What is DDMS?
Android Studio has debugging tools known as DDMS i.e. Dalvik Debug Monitor Server.
It has wide debugging features which include:

- Port forwarding services.
- Screen capture on the device.
- Thread and Heap information.
- Incoming call and SMS spoofing.
- Logcat
- Radio state information.
- Location data spoofing.

DDMS is integrated with the Android studio. To launch the DDMS, you need to open the Android Device Monitor (ADM) first and then click on the DDMS menu button. Once DDMS is launched, then on the left-hand side the list of connected devices is displayed along with the processes which are running on each device.

With the help of DDMS, you can debug both on real devices and emulators.

What are the different data storage options available on the Android platform?

Android platform provides a wide range of data storage options. These options must be used based on the need such as data is secure and used with permission only or can be accessed publicly.

Below is the list of data storage options on the Android platform:

- **SharedPreference**: It stores data in XML files. It is the simplest way to store private data in the key-value pair.
- **SQLite**: It stores structured data in the private database.
- **Internal Storage**: It stores data in the device file system and any other app cannot read this data.
- **External Storage**: Data is stored in the file system but it is accessible to all apps in the device

Explain Sensors in Android.

Android-enabled devices have built-in Sensors that measure Orientation, Motion and other conditions.

These sensors provide data with high accuracy, which will help to monitor the positioning and movement of the device. Some of the sensors are hardware-based and few are software-based.

There are three categories of sensors as mentioned below:

- **Motion Sensors**: These sensors measure the rotational & acceleration forces and it includes gravity sensors, rotational vector sensors, accelerometers, etc.
- **Environmental Sensors**: It measures air temperature, pressure, humidity, etc.
- **Position Sensors**: It measures the physical position of the device and includes orientation sensors and magnetometers.

There are four types of Java Classes as shown below:

- Sensor Manager
- Sensor
- SensorEvent
- SensorEventListener

What do ADT stands for?

ADT stands for Android development tool,This is useful to develop the applications and test the applications.

What are the tools are placed in An Android SDK?

Android SDK collaborated with Android Emulator, DDMS(Dalvik Debug Monitoring Services), AAPT(Android Asset Packaging tool) and ADB(Android debug bridge)

What is viewGroup in android?

View group is a collection of views and other child views, it is an invisible part and the base class for layouts.

What are the notifications available in android?

Toast Notification − It will show a pop up message on the surface of the window

Status Bar Notification − It will show notifications on status bar

Dialogue Notification − It is an activity related notification.

What is container in android?

The container holds objects,widgets,labels,fields,icons,buttons.etc.

What is an Adapter in android?

The Adapter is used to create child views to represent the parent view items.

Where layouts are placed in android?

In The Layout folder, layouts are placed as XML files

What are the exceptions available in android?

InflateException, Surface.OutOfResourceException, SurfaceHolder.BadSurfaceTypeException, and WindowManager.BadTokenException

Why can't you run java byte code on Android?

Android uses DVM (Dalvik Virtual Machine) rather using JVM(Java Virtual Machine), if we want, we can get access to .jar file as a library.

How to launch an activity in android?

Using with intent, we can launch an activity.

```
Intent intent = new Intent(this, MyTestActivity.class);

    startActivity(intent);
```

What is fragment in android?

Fragment is a piece of activity, if you want to do turn your application 360 degrees, you can do this by fragment.

Text Books & References

1. https://developer.android.com/studio

2.“Head First Android Development: A Brain-Friendly Guide” by Dawn Griffiths & David Griffiths O’Relliy publication

3.“Android Programming for Beginners” by John Horton