PRACTICAL 1

AIM: Introduction to Android, Introduction to Android Studio IDE, Application Fundamentals: Creating a Project, Android Components, Activities, Services, Content Providers, Broadcast Receivers, Interface overview, Creating Android Virtual device, USB debugging mode, Android Application Overview. Simple "Hello World" program.

Solution:

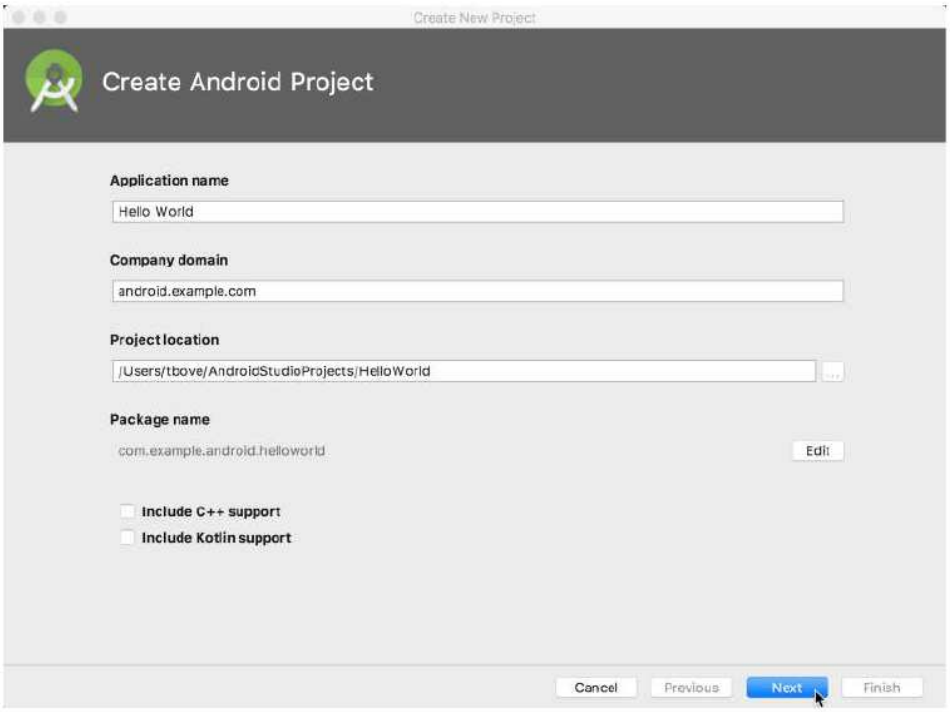
1.1 Creating a Project

1.1.1 In the Welcome to Android Studio window, click **Start a new Android Studio project.**

1.1.2 In the Create New Project window, enter the following values:

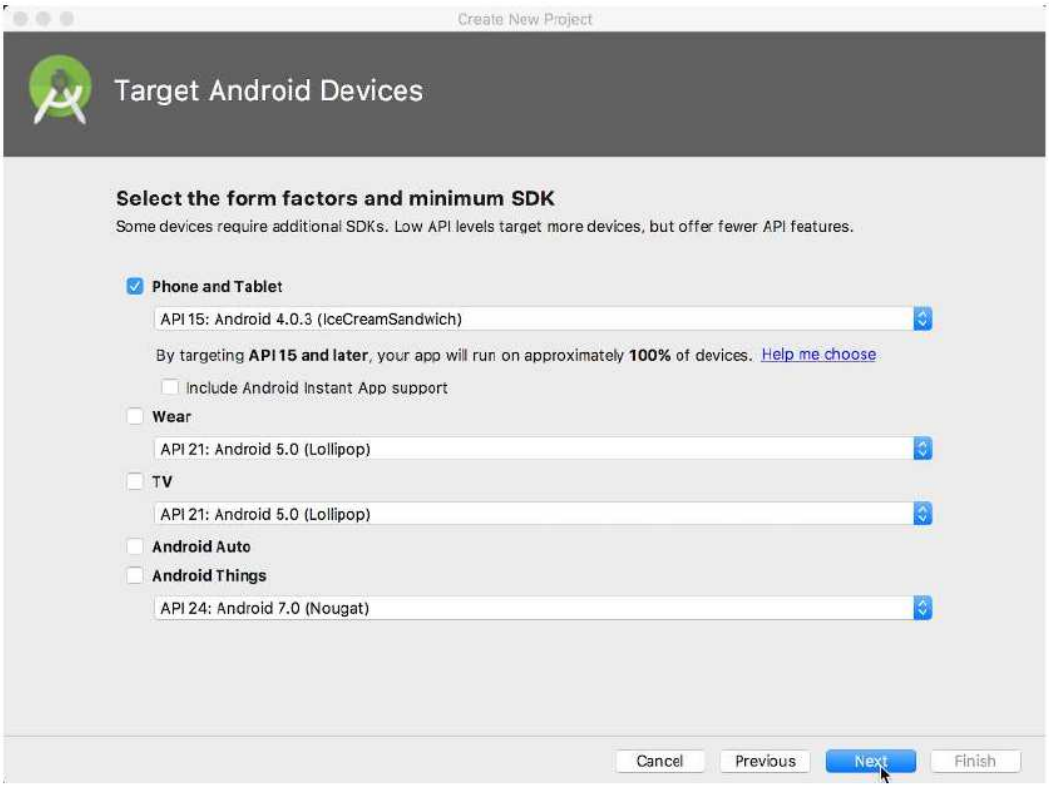
Application Name: " practical1helloworld "

Company Domain: "example.com"

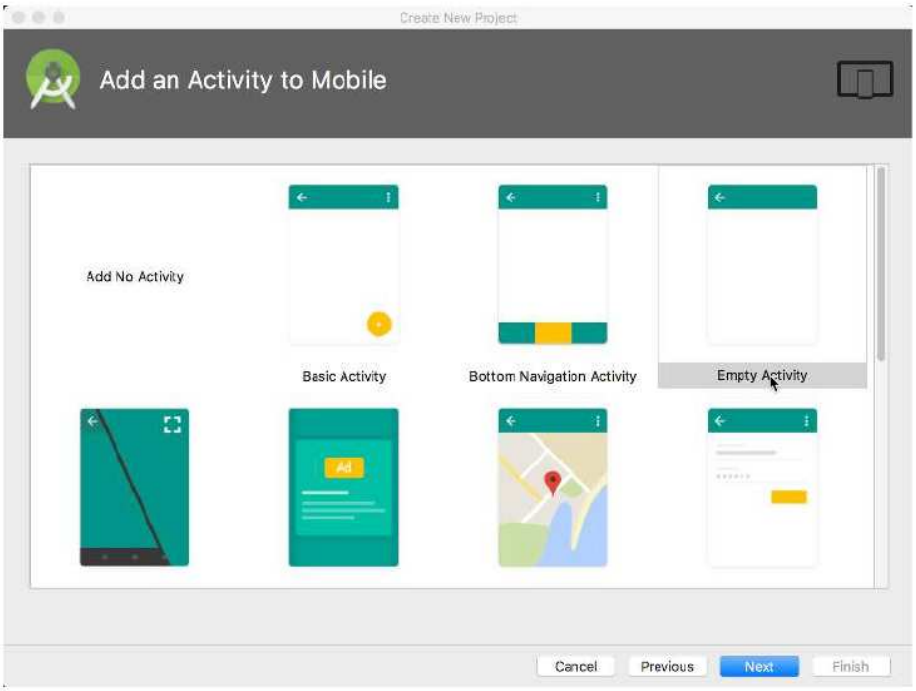


1.1.3 Leave unchecked the options to Include C++ support and Include Kotlin support, and click Next.

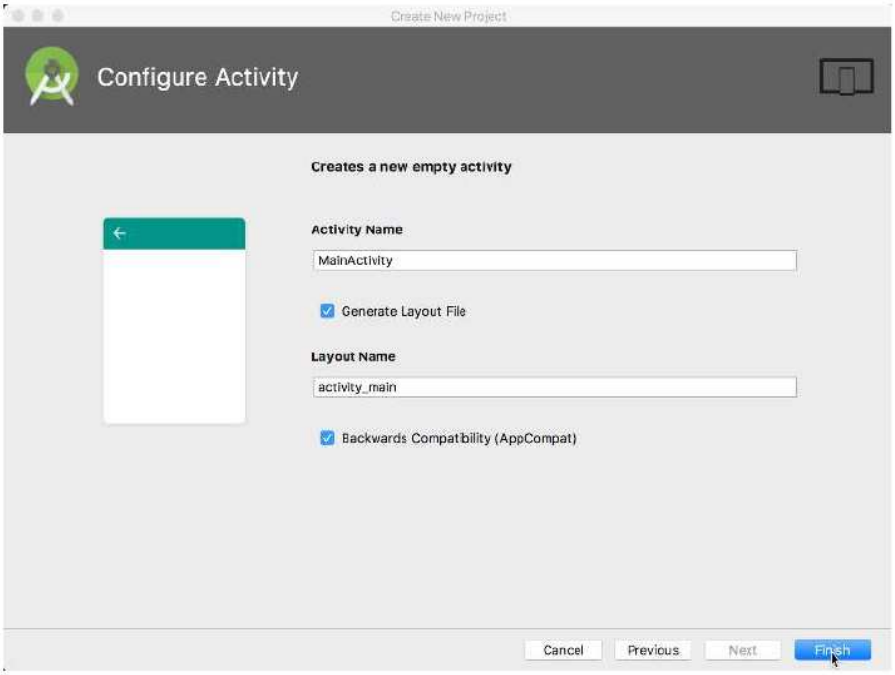
1.1.4 On the Target Android Devices screen, Phone and Tablet should be selected. Ensure that API 15: Android 4.0.3 IceCreamSandwich is set as the Minimum SDK; if it is not, use the popup menu to set it.



1.1.5 **The Add an Activity** window appears. An Activity is a single, focused thing that the user can do. It is a crucial component of any Android app. An Activity typically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, **choose Empty Activity** as shown below, and **click Next.**



1.1.6 The Configure Activity screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activity provided by the template is named MainActivity. You can change this if you want, but this lesson uses MainActivity.



1.1.7 Click Finish

1.2 Android Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application –Activities, Services, Content Providers, Broadcast Receivers

1.3 Activities

An activity represents a single screen with a user interface; in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of Activity class as follows –

```
class MainActivity : AppCompatActivity() {

}
```

1.4 Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different

application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of Service class as follows –

class MyService: Service(){}

1.5 Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
Class  MyExample: ContentProvider {

  public void onCreate(){}

}
```

1.6 Broadcast Receivers

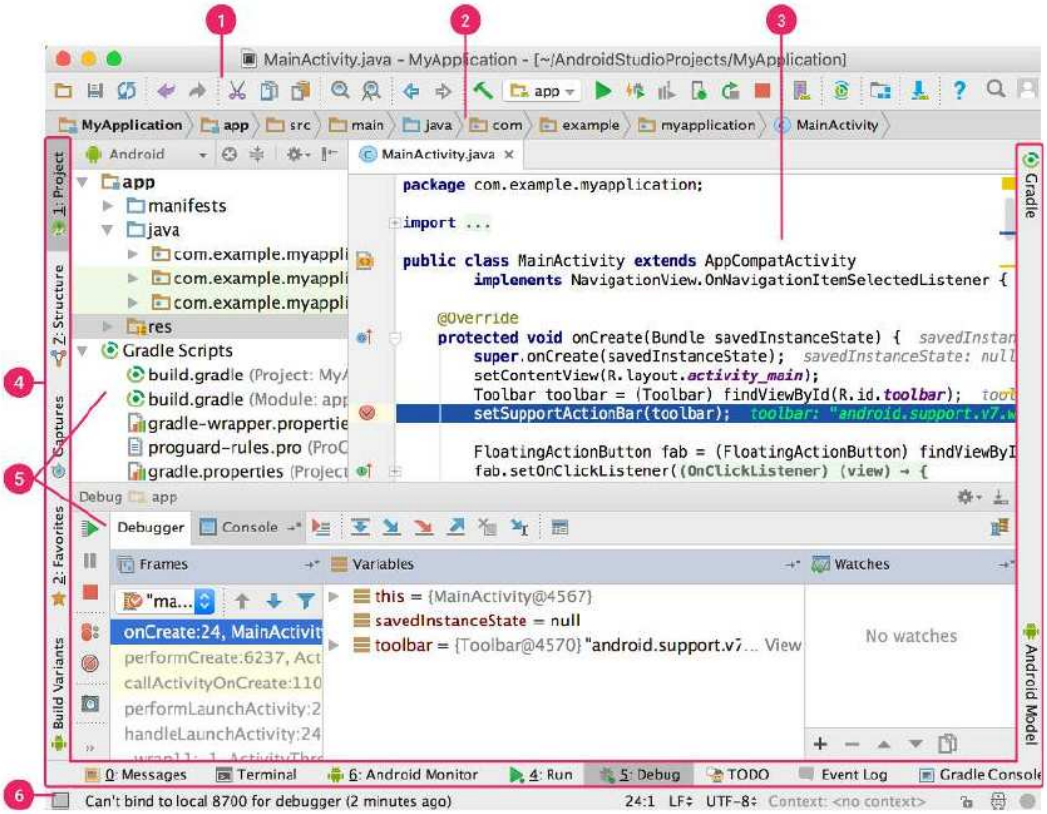
Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
class MyReceiver : BroadcastReceiver()  {

  public void onReceive(context,intent){}

}
```

1.7 Interface overview

The Android Studio main window is made up of several logical areas identified in below figure



1.7.1 The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

1.7.2. The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

1.7.3. The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

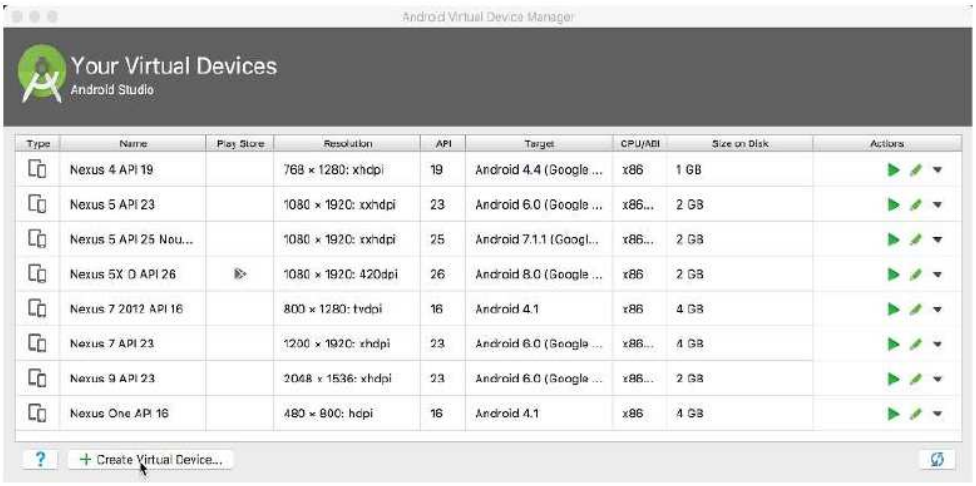
1.7.4. The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

1.7.5. The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

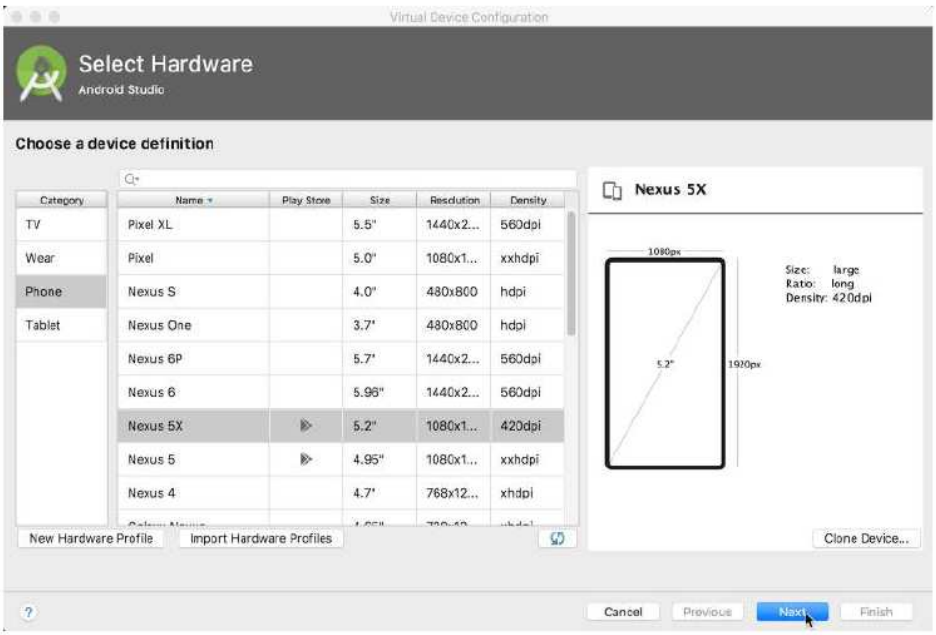
1.7.6. The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

1.8 Creating Android Virtual device (AVD)

In Android Studio, **select Tools > Android > AVD Manager**, or **click the AVD Manager icon** in the toolbar. The Your Virtual Devices screen appears. If you've already created virtual devices, the screen shows them (as shown in the figure below); otherwise you see a blank list.

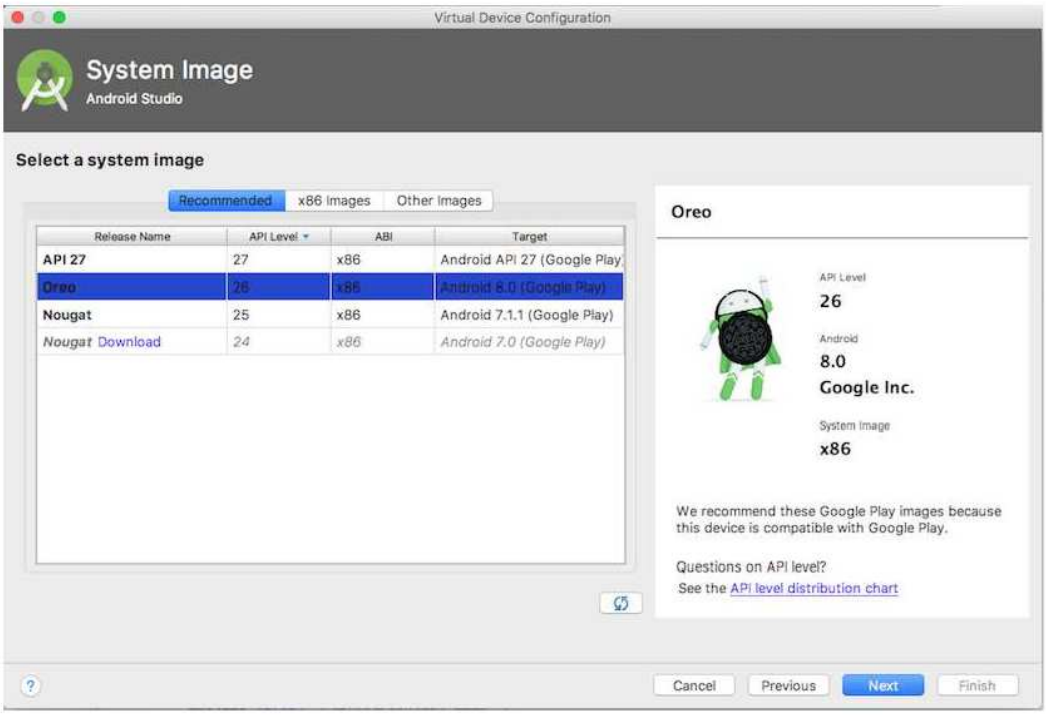


Click the +**Create Virtual Device**. The Select Hardware window appears showing a list of pre configured hardware devices. For each device, the table provides a column for its diagonal display **size (Size), screen resolution in pixels (Resolution), and pixel density (Density).**



Choose a device such **as Nexus 5x or Pixel XL**, and **click Next**. The System Image screen appears.

Click the **Recommended tab** if it is not already selected, and choose which version of the Android system to run on the virtual device (such as Oreo).



If a **Download** link is visible next to a system image you want to use, it is not installed yet. Click the link to **start the download**, and click **Finish** when it's done.

After choosing a **system image**, **click Next**. The Android Virtual Device (AVD) window appears. You can also change the name of the AVD. Check your configuration and **click Finish.**

1.9 Simple "Hello World" program.

1.9.1The Main Activity File

The main activity code is a Java file MainActivity.java. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application.

MainActivity.kt

```
package com.example.harshali.practical1helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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

1.9.2 The Manifest File

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.harshali.practical1helloworld">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      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>
```

1.9.3 The Strings File

strings.xml

```
<resources>
   <string name="app_name">Practical1(Helloworld)</string>
   <string name="hello_world">Hello World!...</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_main"> MainActivity </string>
</resources>
```

1.9.4 The Layout File

The activity_main.xml is a layout file available in res/layout directory, that is referenced by your application when building its interface.

activity_main.xml

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

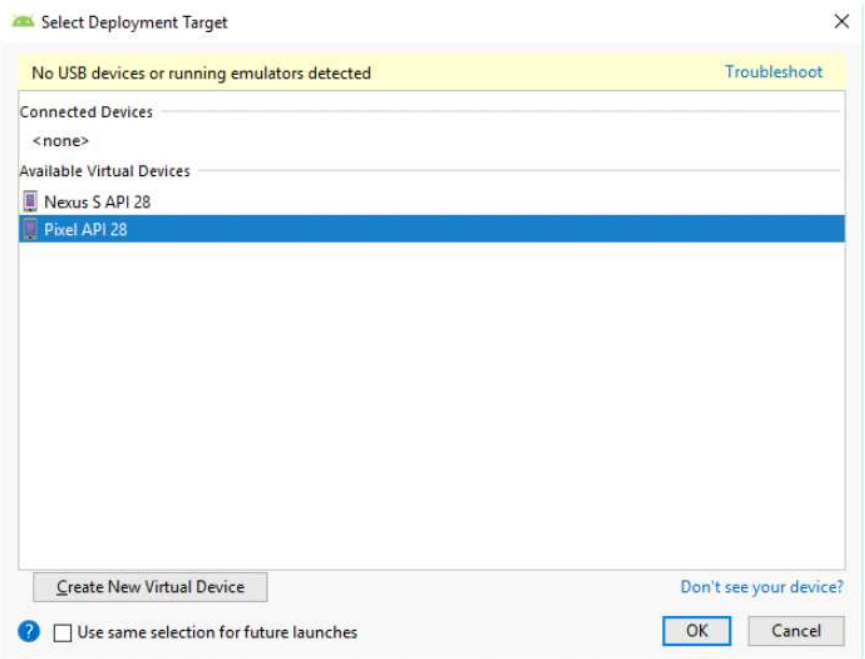
```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

1.10 Run the app on the virtual device

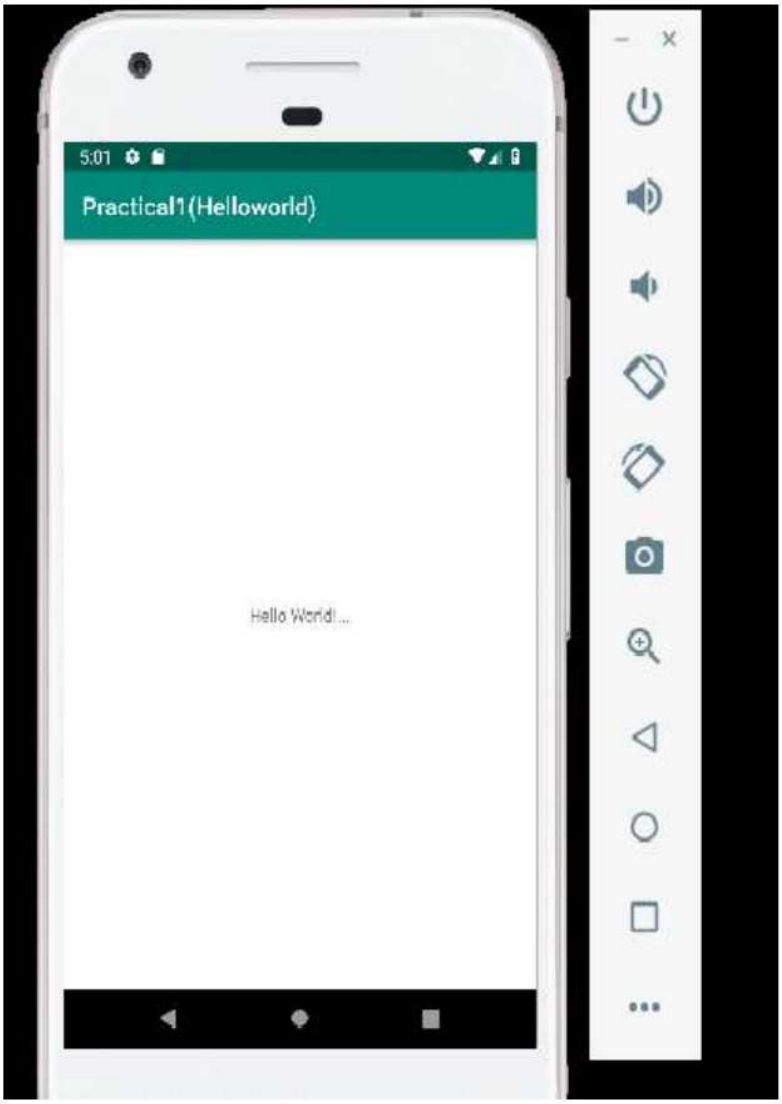
In this task, you will finally run your Hello World app.

In Android Studio, choose **Run > Run app** or **click the Run icon** in the toolbar.

The Select Deployment Target window, under Available Virtual Devices, **select the virtual device**, which you just created, **and click OK**.



OUTPUT:

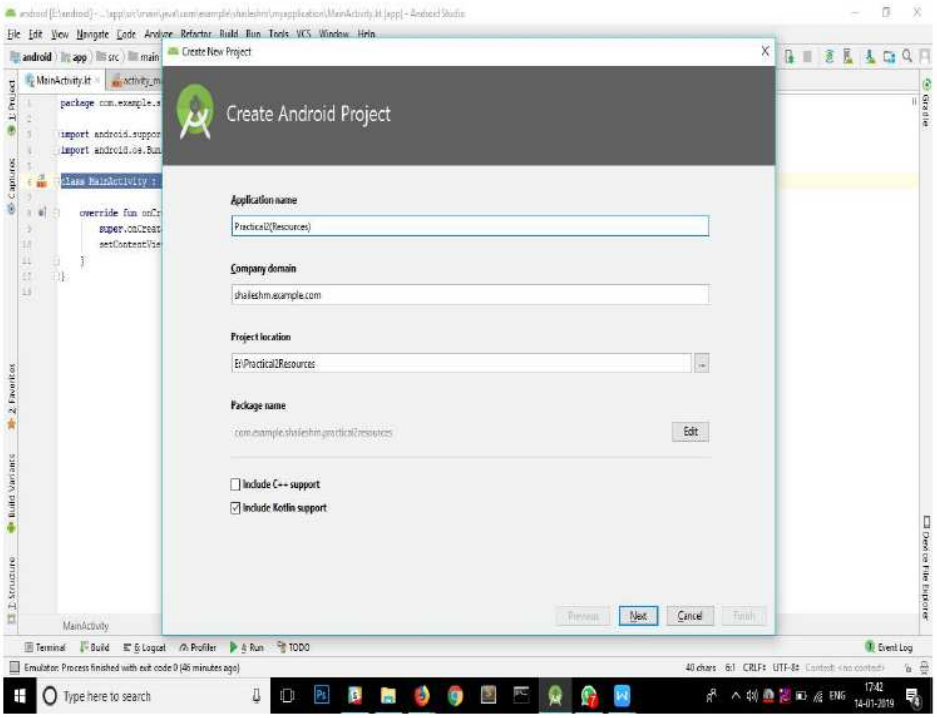


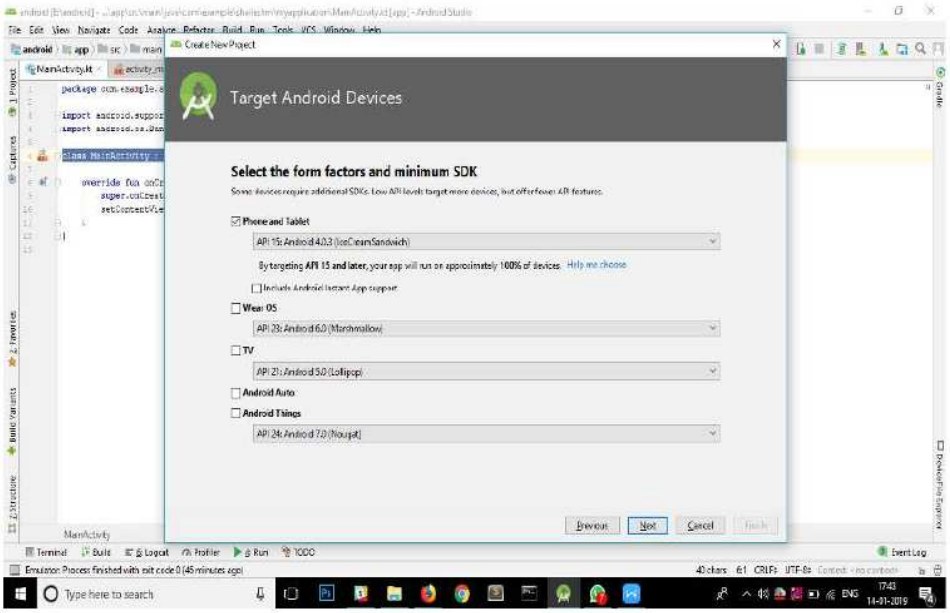
PRACTICAL 2

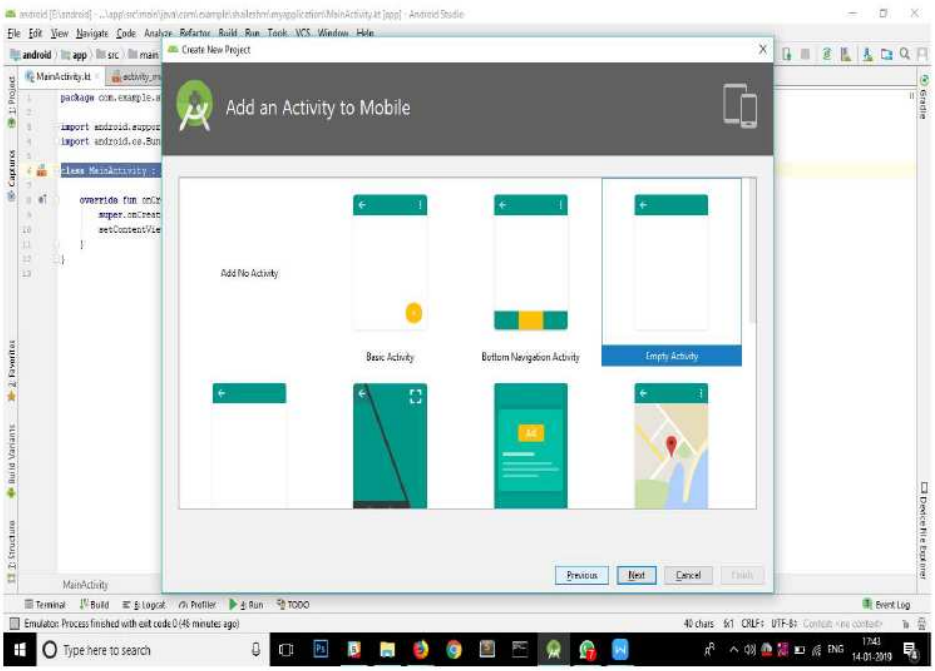
Programming Resources

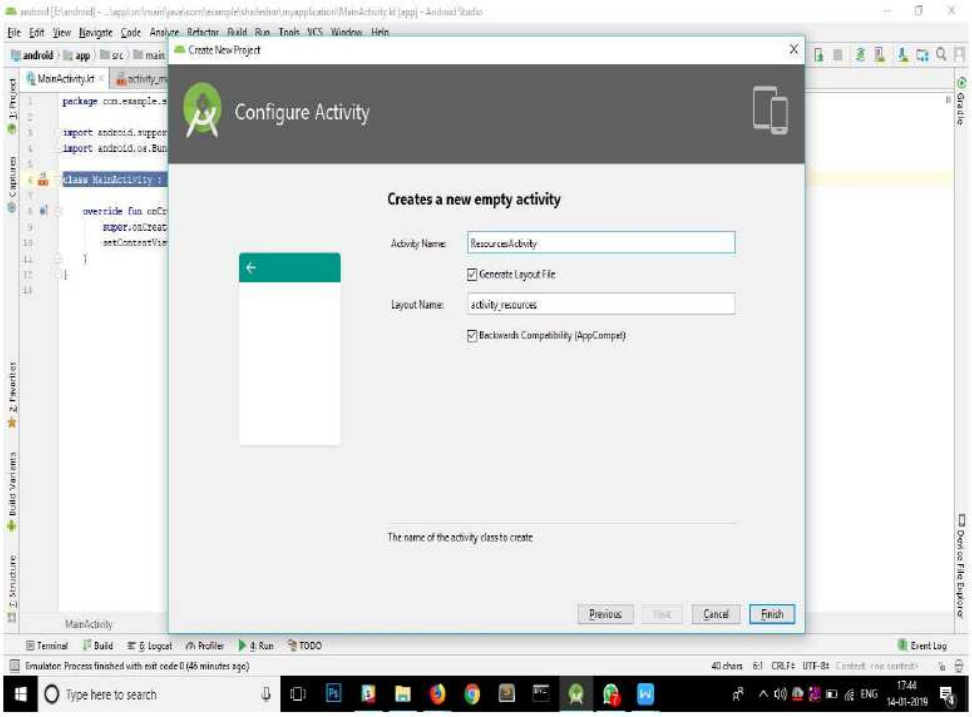
AIM: Android Resources: (Color, Theme, String, Drawable, Dimension, Image).

STEP 1:Create a new Project (give appropriate name)





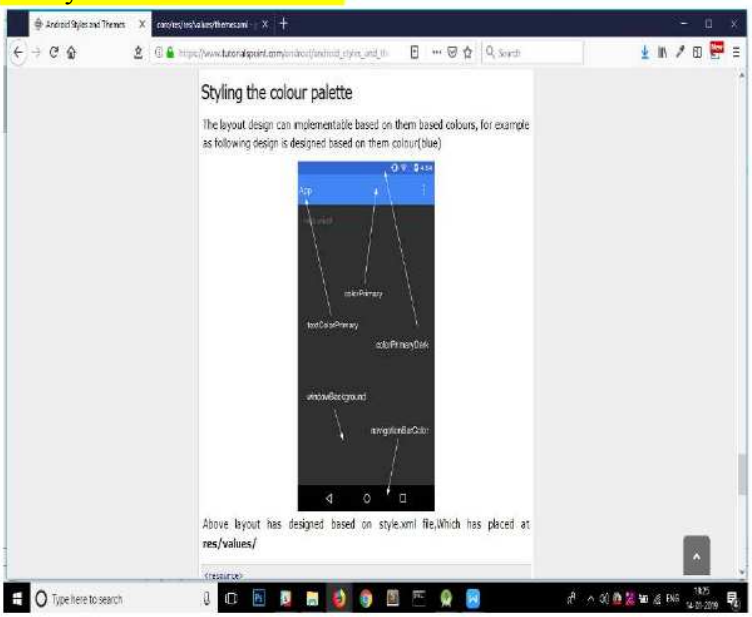




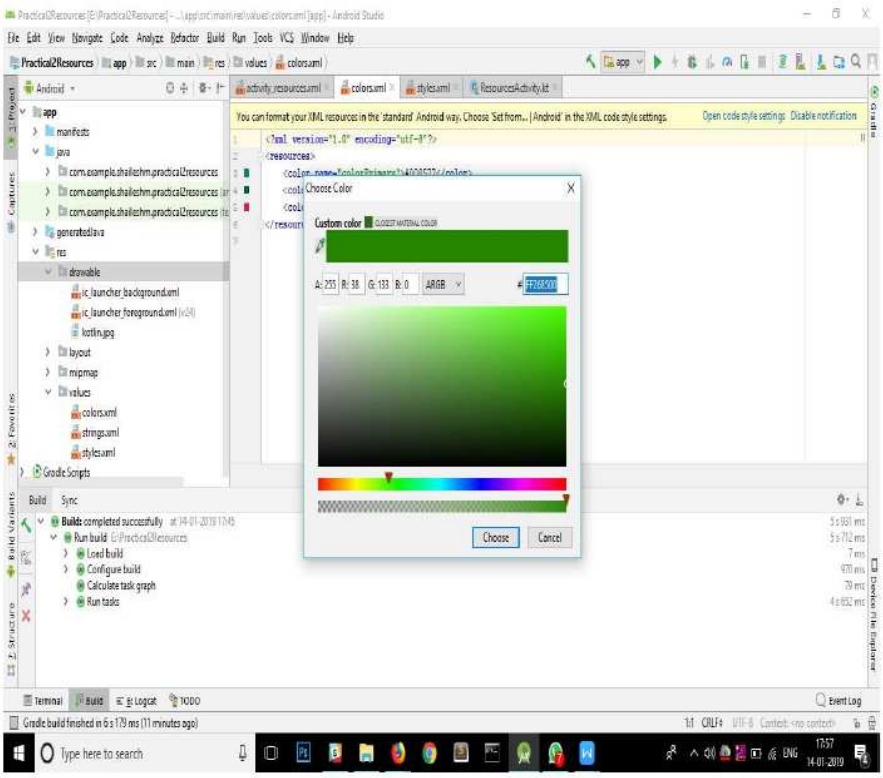
2.1 Styling the colour palette

The layout design can implementable based on them based colours, for example as following design is designed based on them colour(blue)

```
<resource>
  <style name="AppTheme" parent="android:Theme.Material">
     <item name ="android:color/primary">@color/primary</item>
     <item name ="android:color/primaryDark">@color/primary_dark</item>
     <item name ="android:colorAccent/primary">@color/accent</item>
  </style><resource>
```



Change the color shown in below figure in colors.xml

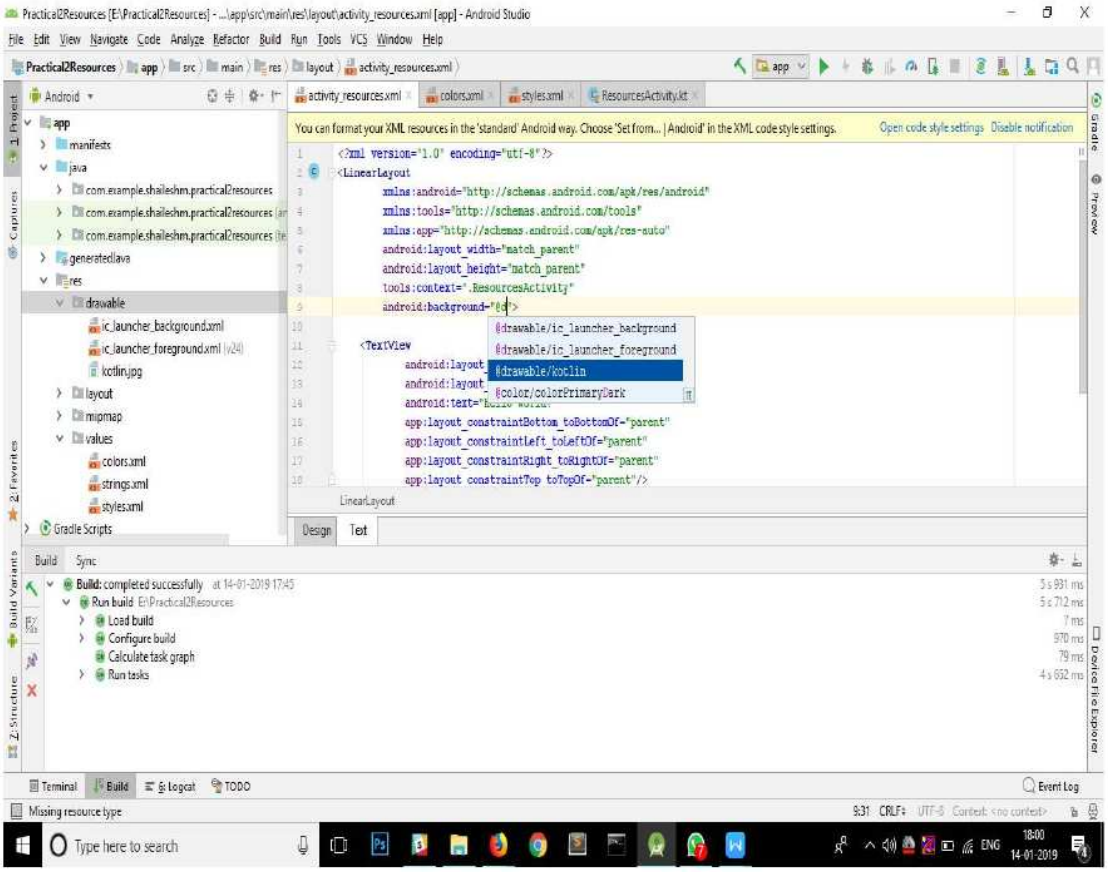


colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#268500</color>
    <color name="colorPrimaryDark">#57003e</color>
    <color name="colorAccent">#1bd8a2</color>
</resources>
```

2.2 Add the image in Drawable folder in below manner

Download any image and paste under drawable folder



2.3 activity_resources.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ResourcesActivity"
    android:background="@drawable/kotlin"> Note: drawble and image

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

```
        android:text="@string/Resources"          Note : String
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</LinearLayout>
```

2.4 strings.xml

```
<resources>
    <string name="app_name">Practical2(Resources)</string>
    <string name="Resources">Android Resources Example</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main"> ResourcesActivity </string>
</resources>
```

2.5 styles.xml (Themes and Dimensions)

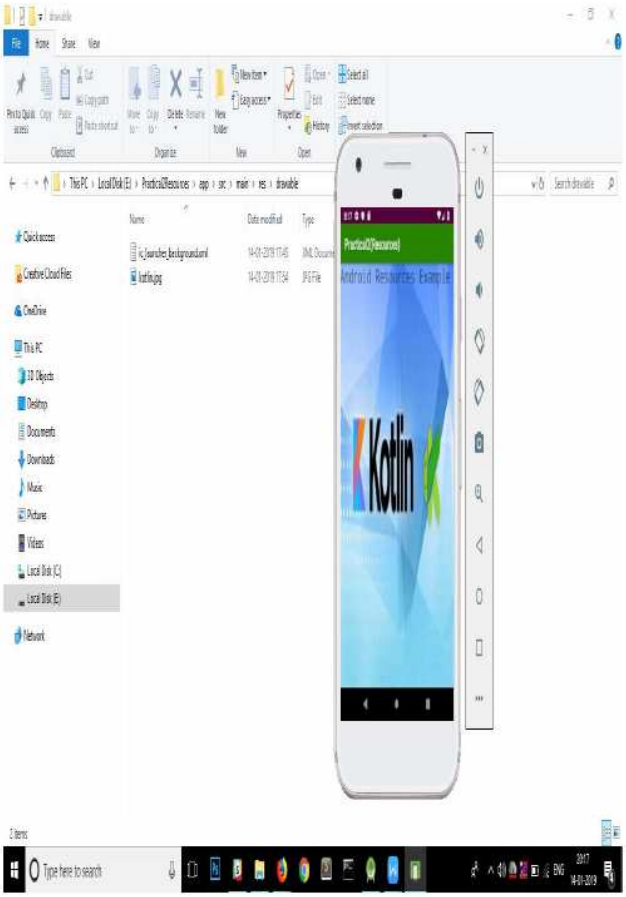
```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme"parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:capitalize">characters</item>
        <item name="android:typeface">monospace</item>
        <item name="android:textSize">12pt</item>

    </style>
</resources>
```

2.6 ResourcesActivity.kt

```
package com.example.harshali.practical2resources

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class ResourcesActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_resources)
```

```
    }
}
```

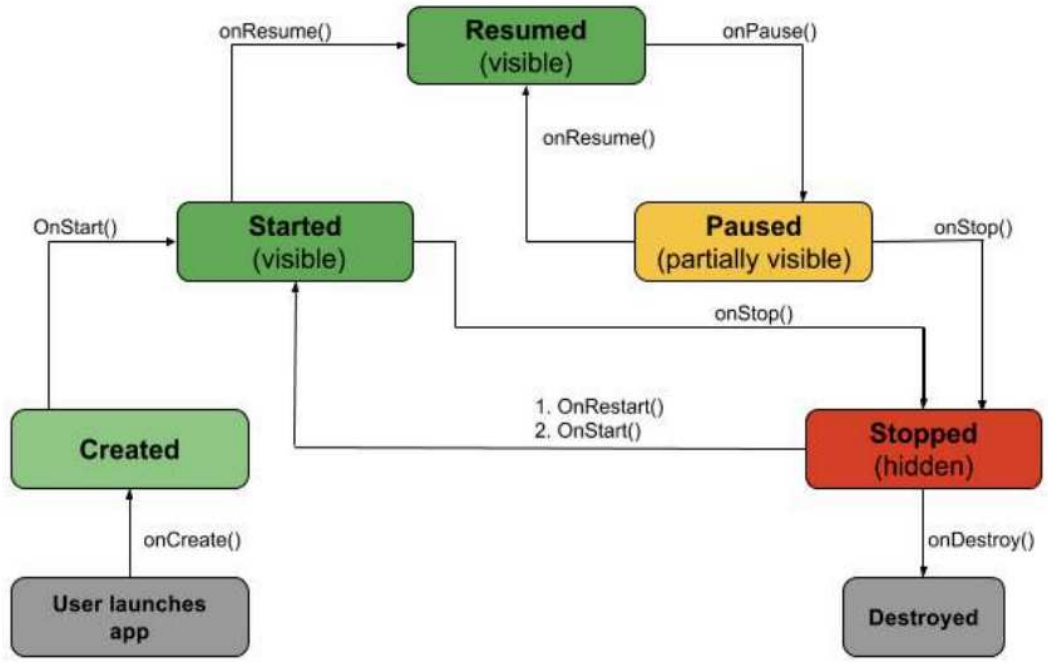
2.7 OUTPUT:



PRACTICAL 3
Programming Activities and fragments

3.A Activity Life Cycle, Activity methods, Multiple Activities,

Activity Lifecycle:



1. **onCreate():** Called by the OS when the activity is first created. This is where you initialize any UI elements or data objects. You also have the savedInstanceState of the activity that contains its previously saved state, and you can use it to recreate that state.

2. **onStart():** Just before presenting the user with an activity, this method is called. It's always followed by onResume(). In here, you generally should start UI animations, audio based content or anything else that requires the activity's contents to be on screen.
3. **onResume():** As an activity enters the foreground, this method is called. Here you have a good place to restart animations, update UI elements, restart camera previews, resume audio/video playback or initialize any components that you release during onPause().
4. **onPause():** This method is called before sliding into the background. Here you should stop any visuals or audio associated with the activity such as UI animations, music playback or the camera. This method is followed by onResume() if the activity returns to the foreground or by onStop() if it becomes hidden.

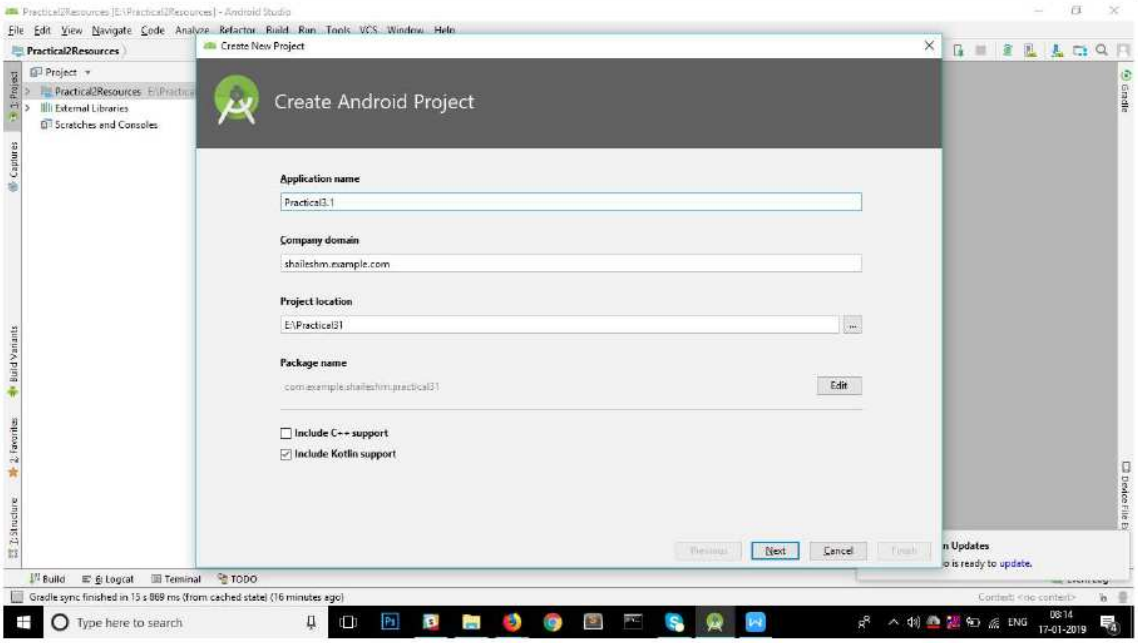
5. **onStop():** This method is called right after onPause(), when the activity is no longer visible to the user, and it's a good place to save data that you want to commit to the disk. It's followed by either onRestart(), if this activity is coming back to the foreground, or onDestroy() if it's being released from memory.

6. **onRestart():** Called after stopping an activity, but just before starting it again. It's always followed by onStart().

7. **onDestroy():** This is the final callback you'll receive from the OS before the activity is destroyed. You can trigger an activity's desctruction by calling finish(), or it can be triggered by the system when the system needs to recoup memory. If your activity includes any background threads or other long-running resources, destruction could

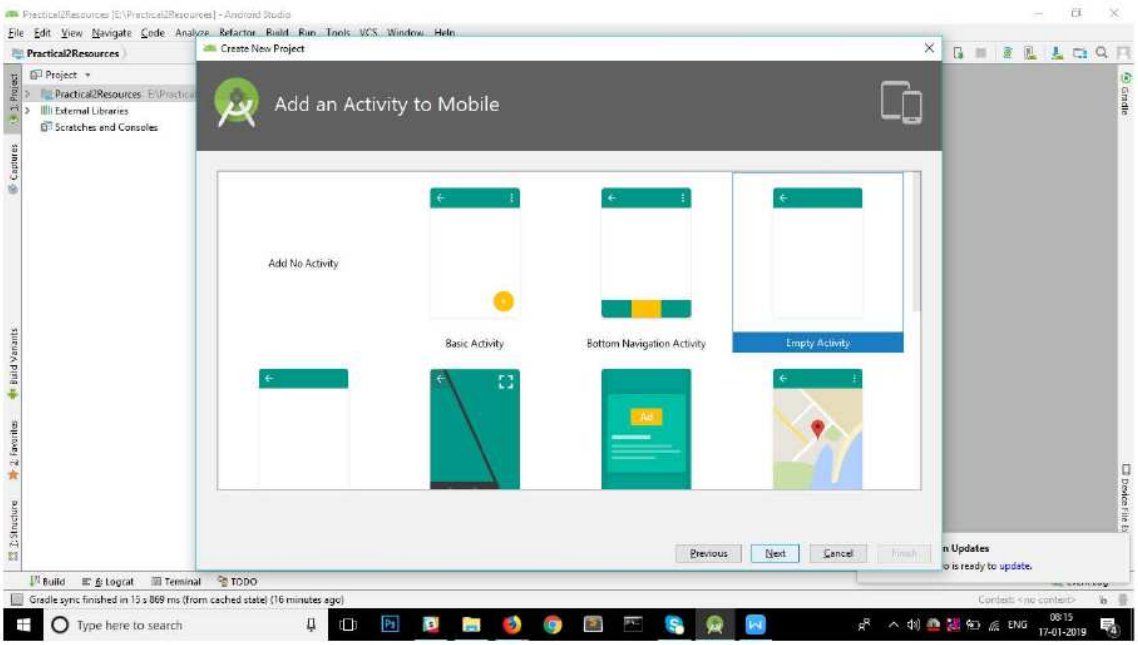
lead to a memory leak if they're not released, so you need to remember to stop these processes here as well.

3.2 Create a new project named Practical3.1

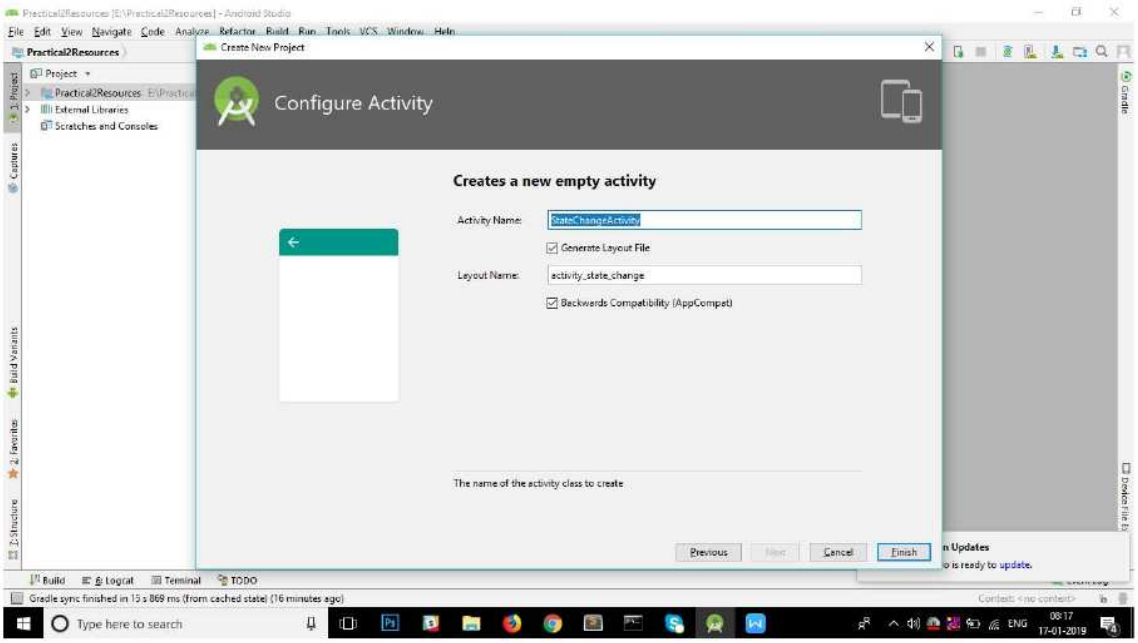


Select minimum form factor and SDK

Select empty activity



3.3 Assign name to activity StateChangeActivity



3.4 Activity_state_change.xml

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


  <TextView
      android:text="Android Life Cycle  Activities"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_alignParentTop="true"
      android:id="@+id/textView"
          android:layout_marginTop="146dp"
      android:layout_marginRight="142dp"
          android:textSize="30sp"
      android:textStyle="bold"
      android:gravity="top|center_horizontal"
          android:layout_gravity="fill"/>

 <Button
      android:text="Jump To Next Activity"
      android:layout_width="wrap_content"
```

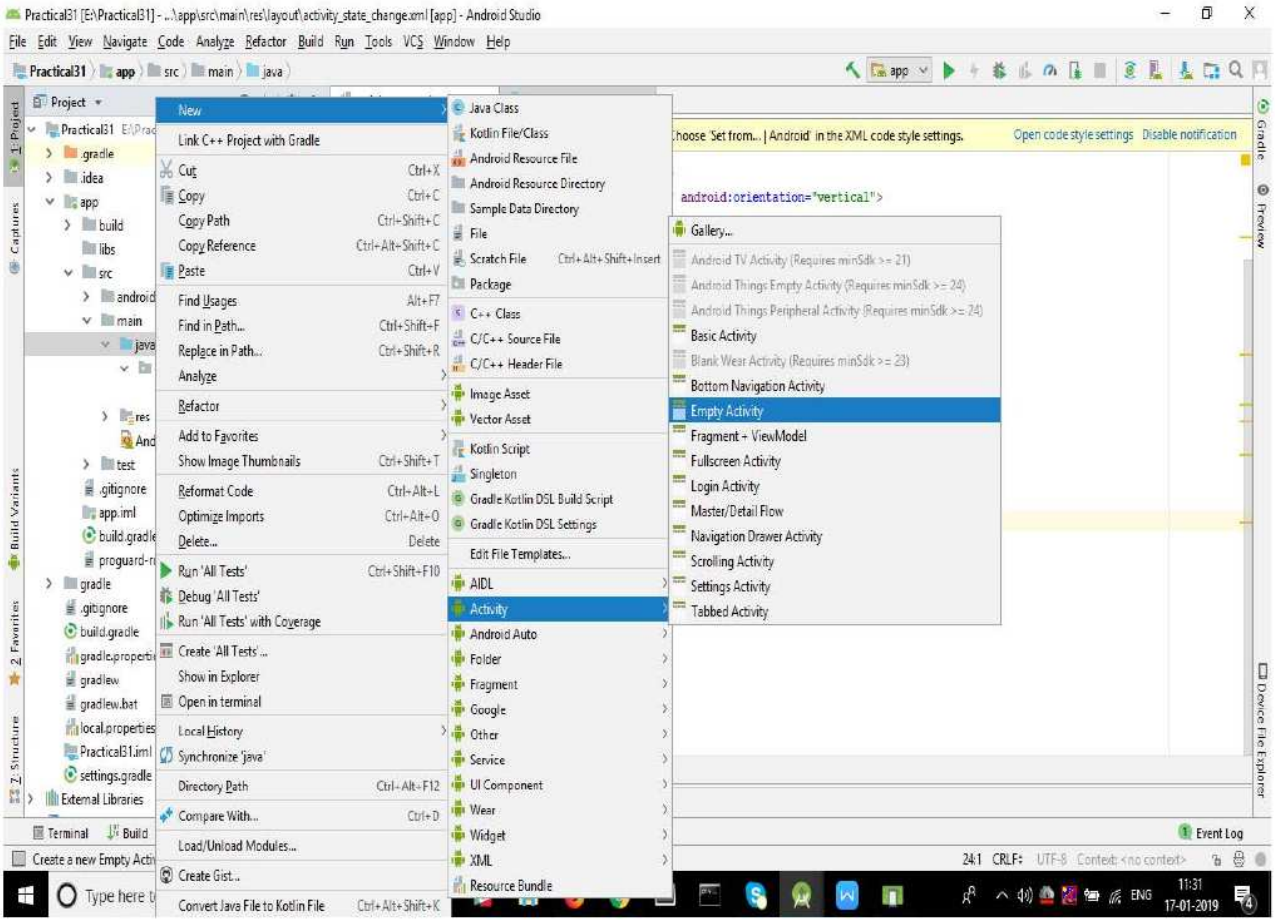
```
        android:layout_height="wrap_content"
        android:id="@+id/btn_next_activity"
        android:gravity="center"/>

</LinearLayout>
```

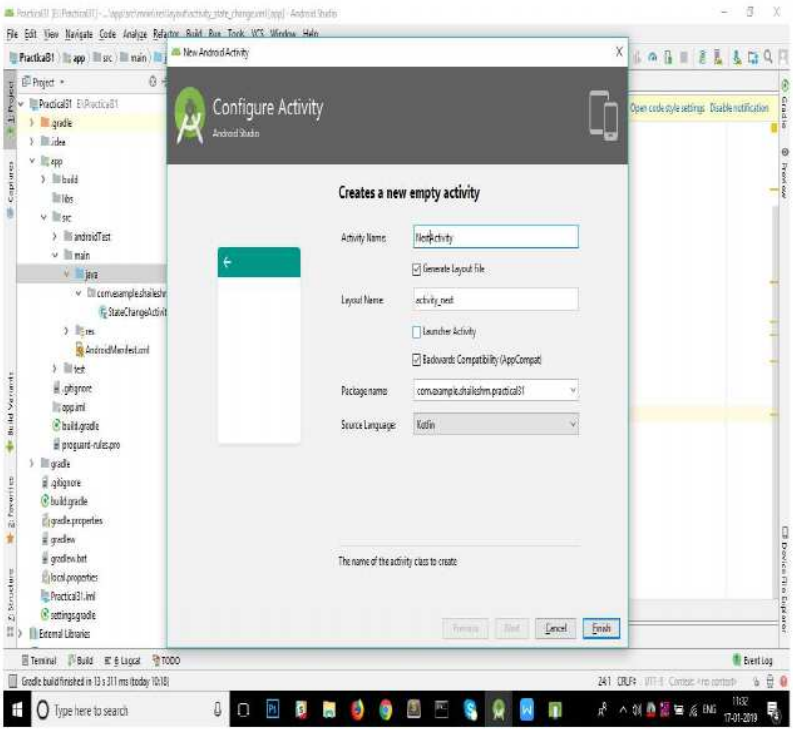
3.5 StateChangeActivity.kt

```
package com.example.harshali.practical31
import android.content.Context
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Button

class StateChangeActivity : AppCompatActivity() {
   internal var context: Context?= null

   val TAG = "Activity Status"
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_state_change)
      context = this@StateChangeActivity
      val btnNextActivity = findViewById(R.id.btn_next_activity) as Button
      btnNextActivity.setOnClickListener { view ->
         val intent = Intent(this, NextActivity::class.java)
         startActivity(intent);
         finish();
         Log.i(TAG, "onCreate")
      }
   }
   override fun onStart() {
      super.onStart()
      Log.i(TAG, "onStart")
   }

   override fun onResume() {
      super.onResume()
      Log.i(TAG, "onResume")
   }

   override fun onPause() {
      super.onPause()
      Log.i(TAG, "onPause")
   }

   override fun onStop() {
```

```
        super.onStop()
        Log.i(TAG, "onStop")
    }

    override fun onRestart() {
        super.onRestart()
        Log.i(TAG, "onRestart")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.i(TAG, "onDestroy")
    }

    override fun onSaveInstanceState(outState: Bundle?) {
        super.onSaveInstanceState(outState)
        Log.i(TAG, "onSaveInstanceState")
    }

    override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
        super.onRestoreInstanceState(savedInstanceState)
        Log.i(TAG, "onRestoreInstanceState")
    }
}
```

3.6 create another activity named (NextActivity)


Practical31 [E:\Practical31] - ..\app\src\main\res\layout\activity_state_change.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'All Tests' Ctrl+Shift+F10
Debug 'All Tests'
Run 'All Tests' with Coverage
Create 'All Tests'...
Show in Explorer
Open in terminal
Local History
Synchronize 'java'
Directory Path Ctrl+Alt+F12
Compare With... Ctrl+D
Load/Unload Modules...
Create Gist...
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Java Class
Kotlin File/Class
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Singleton
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Resource Bundle
Gallery...
Android TV Activity (Requires minSdk >= 21)
Android Things Empty Activity (Requires minSdk >= 24)
Android Things Peripheral Activity (Requires minSdk >= 24)
Basic Activity
Blank Wear Activity (Requires minSdk >= 23)
Bottom Navigation Activity
Empty Activity
Fragment + ViewModel
Fullscreen Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity
Tabbed Activity
android:orientation="vertical"
Event Log
17-01-2019


New Android Activity
Configure Activity
Creates a new empty activity
Activity Name:
Generate Layout File
Layout Name:
Launcher Activity
Backwards Compatibility (AppCompat)
Package name:
Source Language:
Kotlin
The name of the activity class to create
Cancel
Finish

NextActivity.kt

```
package com.example.harshali.practical31
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class NextActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_next)
   }
}
```

Activity_next.xml

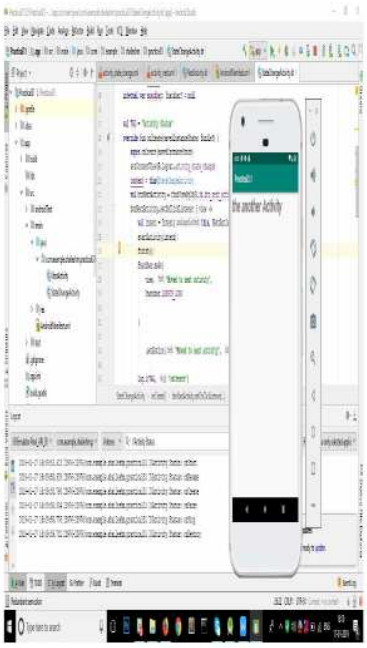
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      android:layout_width="match_parent"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

   <TextView
         android:text="the another Activity "
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:id="@+id/textView2"
         android:textSize="36sp"
         android:textStyle="bold|italic"/>
</android.support.constraint.ConstraintLayout>
```

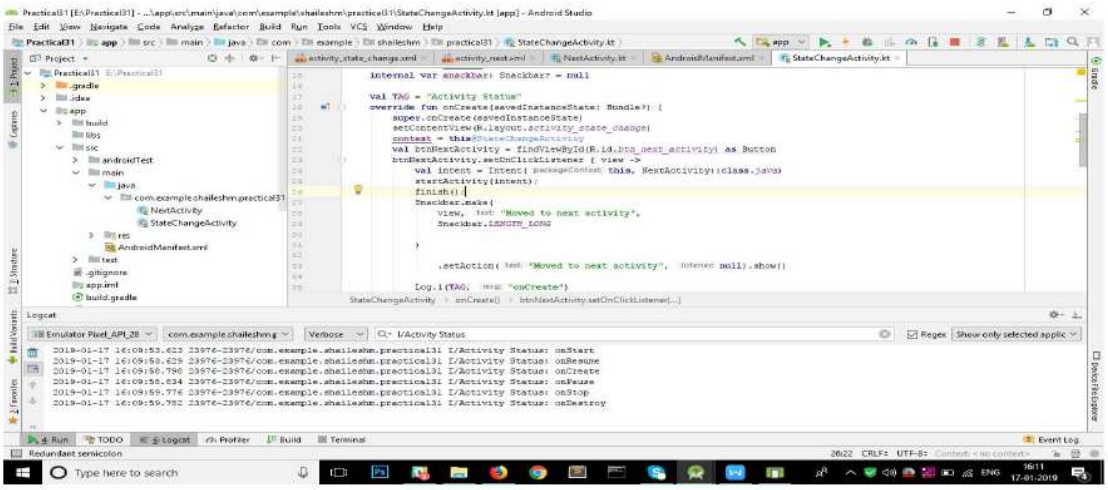
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.shaileshm.practical31">

   <application
         android:allowBackup="true"
         android:icon="@mipmap/ic_launcher"
         android:label="@string/app_name"
         android:roundIcon="@mipmap/ic_launcher_round"
         android:supportsRtl="true"
         android:theme="@style/AppTheme">
```

```
        <activity android:name=".StateChangeActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

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

</manifest>
```

OUTPUT





PRACTICAL 4
Programs related to different Layouts
Coordinate, Linear, Relative, Table, Absolute, Frame, List View, Grid View.

4.1 linear layout:

1. Create a new project **File -> New -> Android Project-->LinearActivity**
2. In Package Explorer right click on **res/layout** folder **"activity_linear.xml"**
3. Now open newly created xml file (in my case **"activity_linear.xml"**) and type the following code.

Activity_linear.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LinearActivity"
    android:orientation="vertical">
  <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Email:"
        android:padding="5dip"
        android:textSize="30sp"/>

  <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dip"/>

  <Button android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Login"
      android:textSize="30sp" android:id="@+id/btnlogin"/>
  <!-- Child linear layout with horizontal orientation -->
  <LinearLayout android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal"
          android:background="#2a2a2a"
          android:layout_marginTop="25dip">

    <TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="Home" android:padding="15dip"
```

```
            android:layout_weight="1"
            android:gravity="center"
                    android:textSize="30sp"
           android:textColor="@color/colorAccent"/>

    <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="About" android:padding="15dip"
            android:layout_weight="1"
            android:gravity="center" android:textSize="30sp"
android:textColor="@color/colorAccent"/>

  </LinearLayout>
</LinearLayout>
```

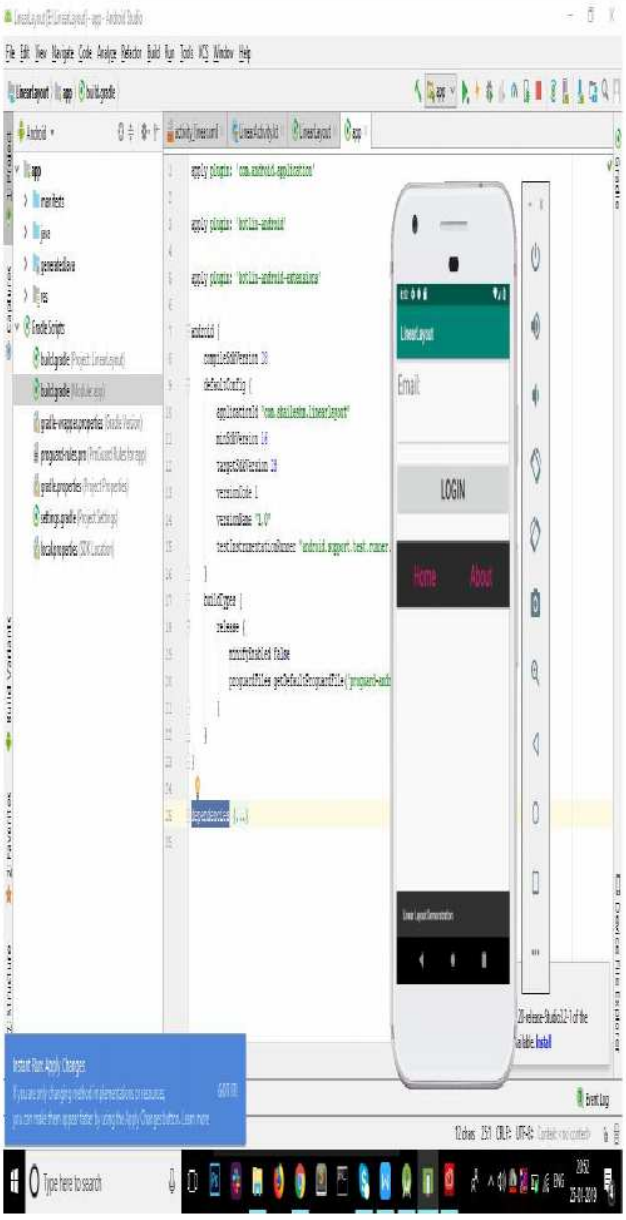
LinearActivity.kt

```
package com.harshali.linearlayout

import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.support.design.widget.Snackbar
class LinearActivity : AppCompatActivity() {
   internal var snackbar: Snackbar? = null
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_linear)
      val btnNextActivity = findViewById(R.id.btnlogin) as Button
      btnNextActivity.setOnClickListener { view ->

         Snackbar.make(
            view, "Linear Layout Demonstration",
            Snackbar.LENGTH_LONG
         )
            .setAction("Linear layout", null).show()
      }
   }
}
```

Add dependencies to gradle module

implementation **'com.android.support:design:28.0.0'**

OUTPUT:



4.2 Relative layout:

2. Create a new project **File -> New -> Android Project-->RelativeActivity**

2. In Package Explorer right click on **res/layout** folder

3. **"activity_relative.xml"**

3. Now open newly created xml file (in my case **"activity_relative.xml"**) and type the following code.

Activity_relative.xml

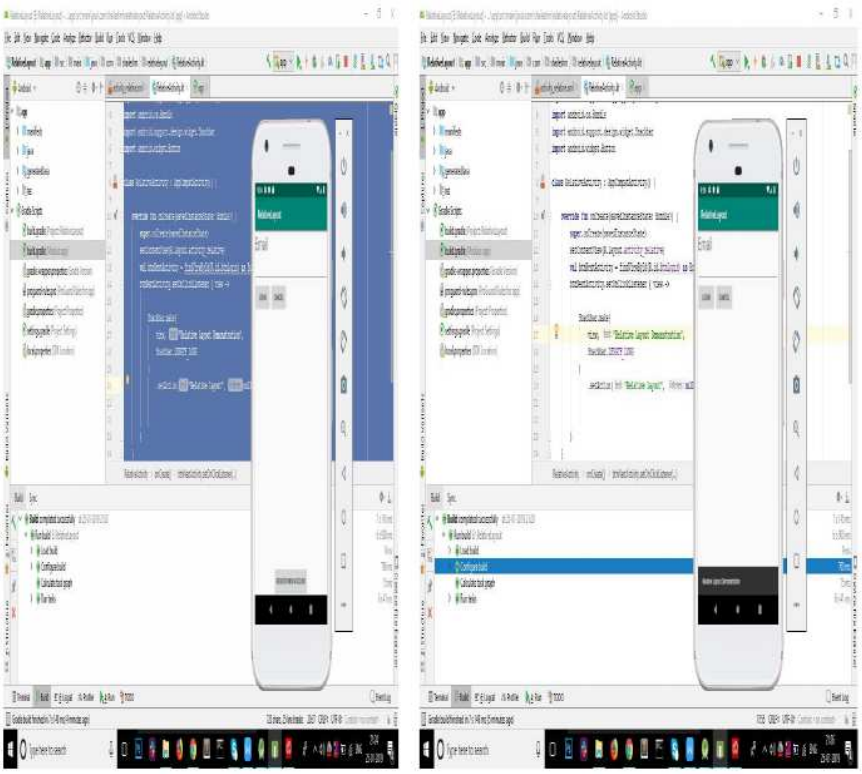
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

  <TextView android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Email"
        android:textSize="30dp"/>

  <EditText android:id="@+id/inputEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label" />

  <Button android:id="@+id/btnLogin"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/inputEmail"
      android:layout_alignParentLeft="true"
      android:layout_marginRight="10px"
      android:text="Login" />

  <Button android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_toRightOf="@id/btnLogin"
      android:layout_alignTop="@id/btnLogin"
      android:text="Cancel" />
```

```
   <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Register new Account"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

RelativeActivity.kt

```
package com.harshali.relativelayout
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.support.design.widget.Snackbar
import android.widget.Button

class RelativeActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_relative)
      val btnNextActivity = findViewById(R.id.btnLogin) as Button
      btnNextActivity.setOnClickListener { view ->

         Snackbar.make(
            view, "Relative Layout Demonstration",
            Snackbar.LENGTH_LONG
         )
            .setAction("Relative layout", null).show()
      }
   }
}
```

OUTPUT:



4.3 Table layout:

1. Create a new project **File -> New -> Android Project-->TableLayout**
2. In Package Explorer right click on **res/layout** folder **"activity_table.xml"**
3. Now open newly created xml file (in my case **"activity_table.xml"**) and type the following code

Activity_table.xml

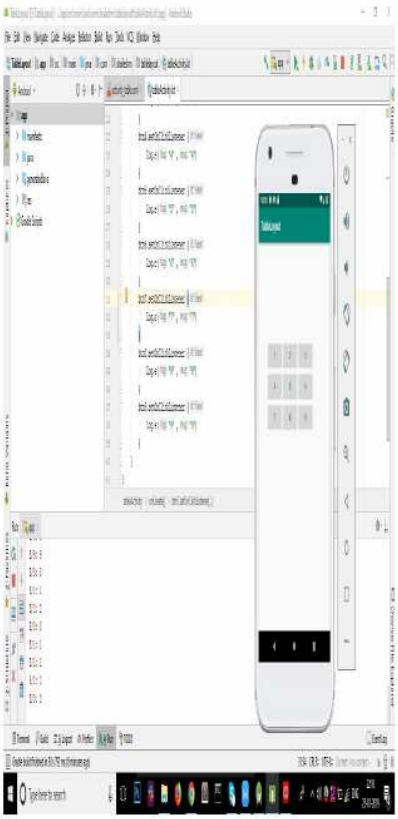
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".tableActivity">
   <TableLayout android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="50dp"
            android:layout_marginTop="150dp">
      <TableRow>
         <Button
               android:id="@+id/btn1"
               android:text="1"
               android:layout_gravity="center"
         />
```

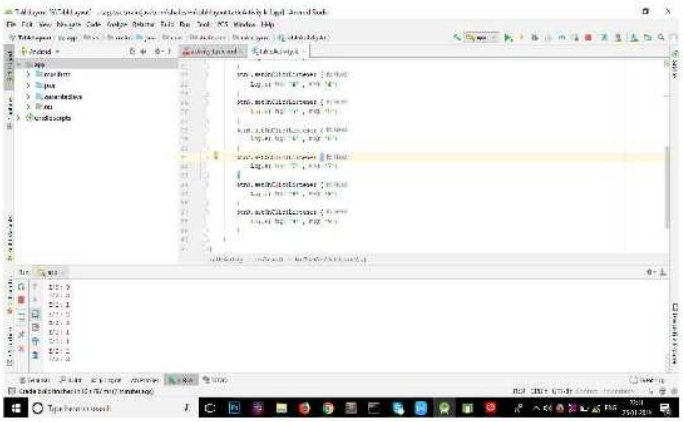
```
        <Button
            android:id="@+id/btn2"
            android:text="2"
            android:layout_gravity="center"
        />
        <Button
            android:id="@+id/btn3"
            android:text="3"
            android:layout_gravity="center"
        />
      </TableRow>
      <TableRow>
        <Button
            android:id="@+id/btn4"
            android:text="4"
            android:layout_gravity="center"
        />
        <Button
            android:id="@+id/btn5"
            android:text="5"
            android:layout_gravity="center"
        /><Button
          android:id="@+id/btn6"
          android:text="6"
          android:layout_gravity="center"
      />
      </TableRow>
      <TableRow>
        <Button
            android:id="@+id/btn7"
            android:text="7"
            android:layout_gravity="center"
        />
        <Button
            android:id="@+id/btn8"
            android:text="8"
            android:layout_gravity="center"
        /><Button
          android:id="@+id/btn9"
          android:text="9"
          android:layout_gravity="center"
      />
      </TableRow>
    </TableLayout>

</LinearLayout>
```

TableActivity.kt

```
package com.harshali.tablelayout

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import kotlinx.android.synthetic.main.activity_table.*

class tableActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_table)
        btn1.setOnClickListener {
            Log.e("1" ,"1")
        }
        btn2.setOnClickListener {
            Log.e("2" ,"2")
        }
        btn3.setOnClickListener {
            Log.e("3" ,"3")
        }
        btn4.setOnClickListener {
            Log.e("4" ,"4")
        }
        btn5.setOnClickListener {
            Log.e("5" ,"5")
        }
        btn6.setOnClickListener {
            Log.e("6" ,"6")
        }
        btn7.setOnClickListener {
            Log.e("7" ,"7")
        }
        btn8.setOnClickListener {
            Log.e("8" ,"8")
        }
        btn9.setOnClickListener {
            Log.e("9" ,"9")
        }
    }
}
```

OUTPUT:





4.4 Frame Layout

Step 1. Create new project "FrameLayout"

Step 2. Add following code in activity_frame.xml resource file

Here is base container FrameLayout then top of it is **<ImageView>** then top of its **<TextView>**

You can add you own image or download "android.jpg" and add in mipmap

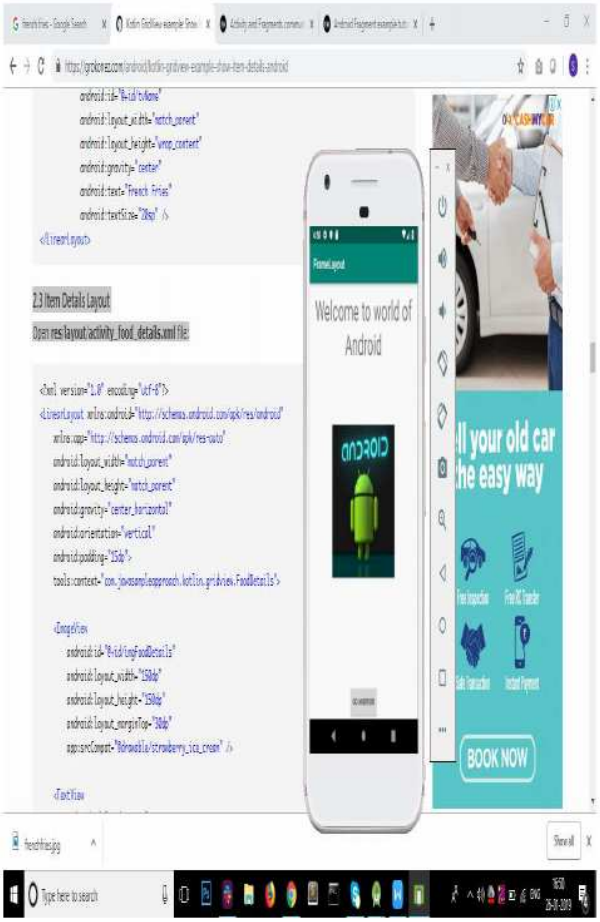
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/frame_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        tools:context=".FrameActivity"
        >
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitXY"
        android:src="@drawable/android"
android:layout_gravity="center_vertical|center"/>

    <TextView
        android:id="@+id/textview1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:gravity="top|center_horizontal"
        android:text="Welcome to world of Android"
        android:textSize="40dp"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="Go Android" android:layout_gravity="bottom|center"/>
</FrameLayout>
```

FrameActivity.kt

```
package com.shaileshm.framelayout

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class FrameActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_frame)
    }
}
```

Output:



4.5 ListView

Step 1 Create a project named ListViewDemo

Step 2 Under activity_main.xml take Button component

Step 3 Set **id attribute to btn** in following manner

Activity_main.xml

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

  <Button android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/btn"
      android:layout_marginTop="200dp"
      android:layout_marginLeft="10dp"
      android:textSize="30dp"

      android:text="android technology" android:gravity="center"/>
</LinearLayout>
```

Step 4 Create an additional mylist.xml file in layout folder which contains view components displayed in listview

Mylist.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">
    <ListView
      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=".list_view" android:entries="@array/array_technology">

  </ListView>
</LinearLayout>
```

Step 5 Now place the list of data in strings.xml file by creating string-array.

strings.xml

```
<resources>
  <string name="app_name">ListViewDemo</string>
  <string-array name="array_technology">
    <item>Android</item>
    <item>Java</item>
    <item>Php</item>
    <item>Hadoop</item>
    <item>Sap</item>
```

```
        <item>Python</item>
        <item>Ajax</item>
        <item>C++</item>
        <item>Ruby</item>
        <item>Rails</item>
        <item>.Net</item>
        <item>Perl</item>
    </string-array>
</resources>
```

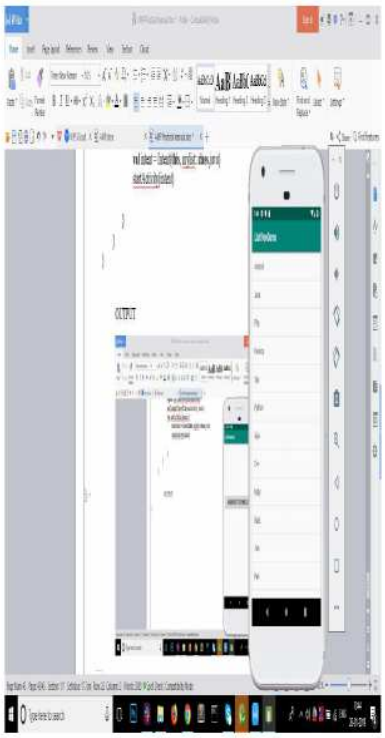
Mylist.kt

```
package com.harshali.listviewdemo

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class mylist : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_mylist)
    }
}
```

MainActivity.kt

```
package com.harshali.listviewdemo
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btn.setOnClickListener {
            val intent = Intent(this, mylist::class.java)
            startActivity(intent)
        }
    }
}
```

OUTPUT



4.6 GridLayout

4.6.1 Container Layout

Open **res/layout/activity_main.xml** file:

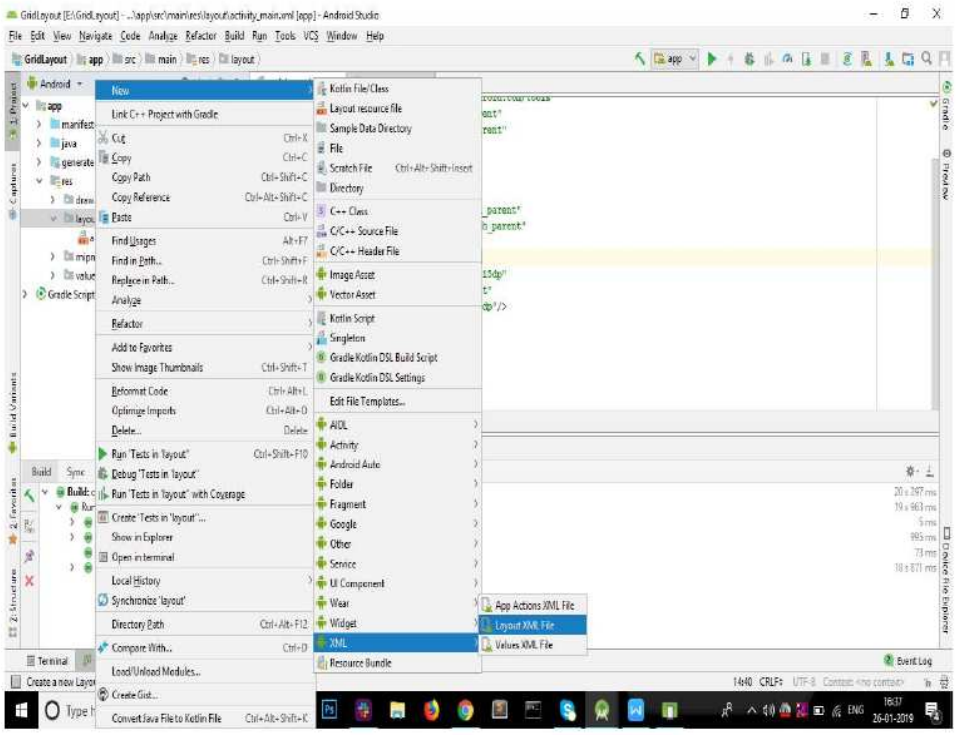
Activity_main.xml

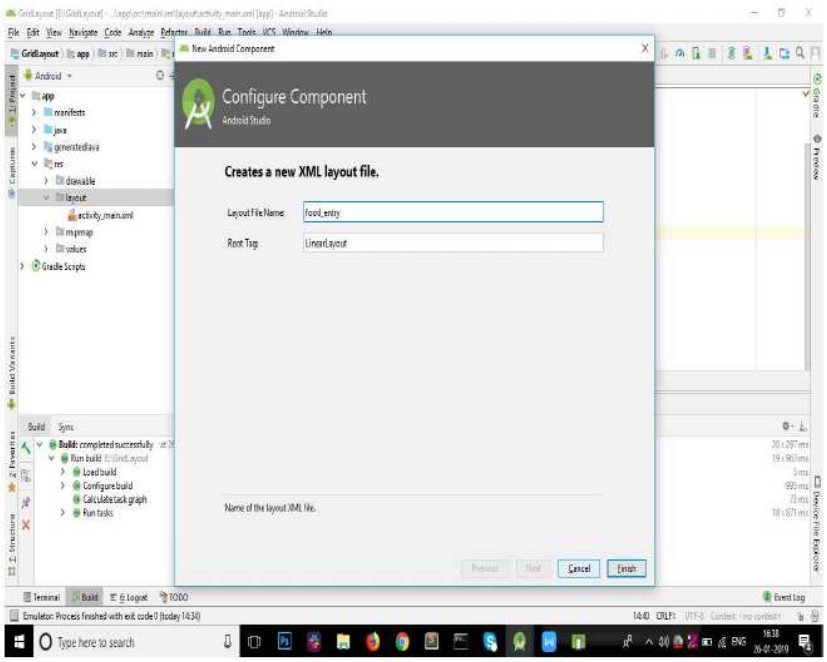
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
  <GridView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/gvFoods"
      android:columnWidth="150dp"
      android:horizontalSpacing="15dp"
      android:numColumns="auto_fit"
      android:verticalSpacing="15dp"/>
</LinearLayout>
```

4.6.2 Item Layout

Add **food_entry.xml** file to **res/layout** folder:



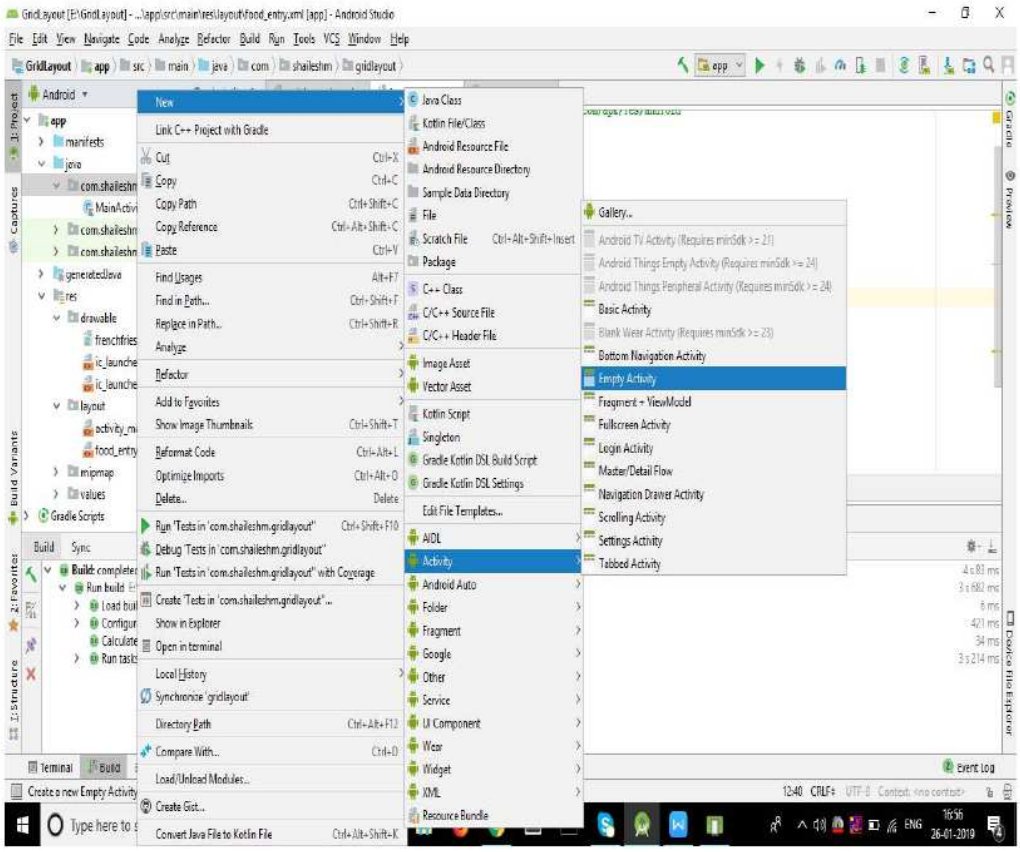


Food_entry.xml

```
<?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:background="#ddd"
        android:gravity="center"
        android:padding="15dp">
   <ImageView
       android:id="@+id/imgFood"
       android:layout_width="150dp"
       android:layout_height="120dp"
       android:src="@drawable/frenchfries"/>

   <TextView
       android:id="@+id/tvName"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:text="French Fries"
       android:textSize="20sp"/>

</LinearLayout>
```

4.6.3 Item Details Layout

Open **res/layout/activity_food_details.xml** file:



food_details.xml

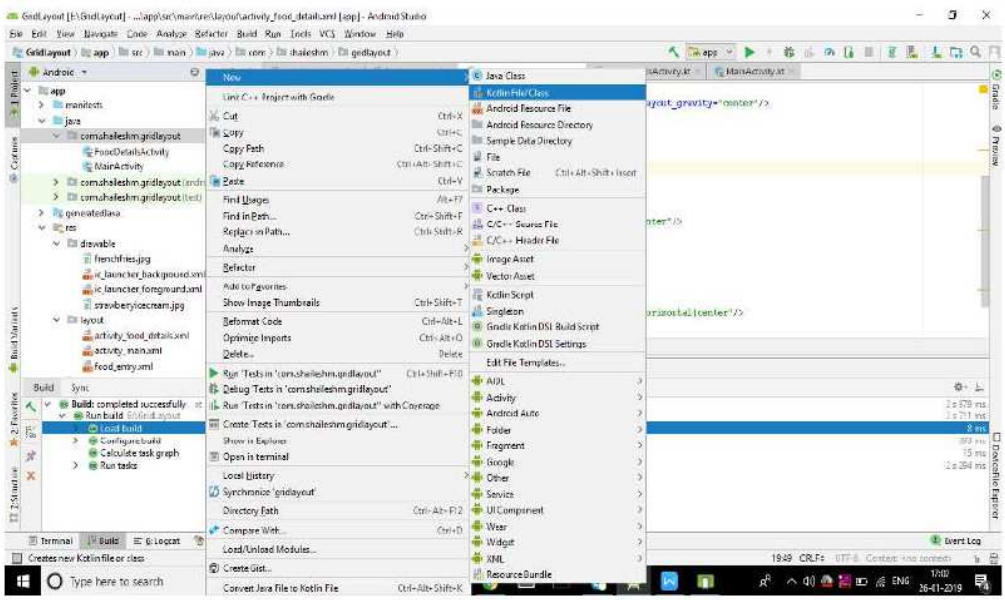
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FoodDetailsActivity">
  <ImageView
      android:id="@+id/imgFoodDetails"
      android:layout_width="150dp"
      android:layout_height="150dp"
      android:layout_marginTop="30dp"
      app:srcCompat="@drawable/strawberryicecream"
android:layout_gravity="center"/>

  <TextView
      android:id="@+id/tvName"
```

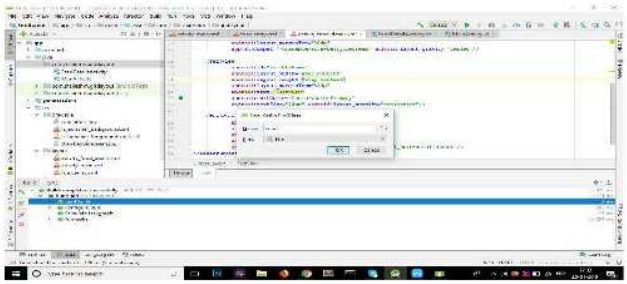
```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="TextView"
        android:textColor="@color/colorPrimary"
        android:textSize="24sp" android:layout_gravity="top|center"/>

    <TextView
        android:id="@+id/tvDetails"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="TextView" android:layout_gravity="fill_horizontal|center"/>
</LinearLayout>
```

4.6.3. Logic

4.6.3.1 Item Class

This class represents data of each Food Item:





Food.kt

```
package com.harshali.gridlayout

class Food {
    var name: String? = null
    var description: String? = null
    var image: Int? = null

    constructor(name: String, description: String, image: Int) {
        this.name = name
        this.description = description
        this.image = image
    }
}
```

4.6.3.2 Main Activity Class

We will:
– Create a BaseAdapter subclass.
– Set an instance of this class as data provider to the GridView.
– Return each cell's view from getView() on your adapter.
– Add OnClickListener for Item's image.

MainActivity.kt

```
package com.harshali.gridlayout

import android.content.Context
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.BaseAdapter
import kotlinx.android.synthetic.main.activity_main.*
```

```
import kotlinx.android.synthetic.main.food_entry.view.*

class MainActivity : AppCompatActivity() {
   var adapter:FoodAdapter? = null
   var foodsList = ArrayList<Food>()
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      // load foods
      foodsList.add(
         Food(
            "Coffee",
            "Coffee preparation is the process of turning coffee beans into a beverage. While the particular steps vary with the type of coffee and with the raw materials, the process includes four basic steps; raw coffee beans must be roasted, the roasted coffee beans must then be ground, the ground coffee must then be mixed with hot water for a certain time (brewed), and finally the liquid coffee must be separated from the used grounds.",
            R.drawable.coffeepot
         )
      )
      foodsList.add(
         Food(
            "Espersso",
            "Espresso's authentic formula is clear and basic, its proper execution a matter of training, experience and natural talent.  A jet of hot water at 88°-93°C (190°-200°F) passes under a pressure of nine or more atmospheres through a seven-gram (.25 oz) cake-like layer of ground and tamped coffee. Done right, the result is a concentrate of not more than 30 ml (one oz) of pure sensorial pleasure.",
            R.drawable.espresso
         )
      )
      foodsList.add(
         Food(
            "French Fires",
            "Heat a few inches of vegetable oil to 300 degrees F in a heavy pot. In 3 or 4 batches, fry the potatoes about 4 to 5 minutes per batch, or until soft. They should not be brown at all at this point-you just want to start the cooking process. Remove each batch and drain them on new, dry paper towels.",
            R.drawable.frenchfries
         )
      )
      foodsList.add(
         Food(
            "Honey",
            "While it is less likely that anyone would do this on their own if they are not a beekeeper, this might be useful for those who aspire to become one. Bees are really great and easy to keep, even in the urban environment! As Novella Carpenter calls them, bees are &amp;quot;gateway animal for urban farmers&amp;quot;. All you need is some space in the backyard/deck. The process of honey harvesting and extraction most likely happens on
```

```
a separate days.",
            R.drawable.honey
        )
    )
    foodsList.add(
        Food(
            "Strawberry",
            "Preparation. Coarsely mash strawberries with sugar, lemon juice, and salt
using a potato masher in a large bowl. Let stand, stirring and mashing occasionally, 10
minutes. Transfer half of strawberry mixture to a blender and purée with cream until
smooth. Freeze mixture in ice cream maker.",
            R.drawable.strawberryicecream
        )
    )
    foodsList.add(
        Food(
            "Sugar cubes",
            "Sugar cubes are extremely simple to make at home - all you need is sugar and
water. In addition to standard cubes, you can add color and flavor to add fun flair to a
tea party or another gathering. Learn how to make sugar cubes using two different
methods: using a pan in the oven or an ice cube tray you leave out overnight.",
            R.drawable.sugarcubes
        )
    )
    adapter = FoodAdapter(this, foodsList)
    gvFoods.adapter = adapter
}    class FoodAdapter : BaseAdapter {
    var foodsList = ArrayList<Food>()
    var context: Context? = null

    constructor(context: Context, foodsList: ArrayList<Food>) : super() {
        this.context = context
        this.foodsList = foodsList
    }

    override fun getCount(): Int {
        return foodsList.size
    }

    override fun getItem(position: Int): Any {
        return foodsList[position]
    }

    override fun getItemId(position: Int): Long {
        return position.toLong()
    }

    override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View
{
        val food = this.foodsList[position]
```

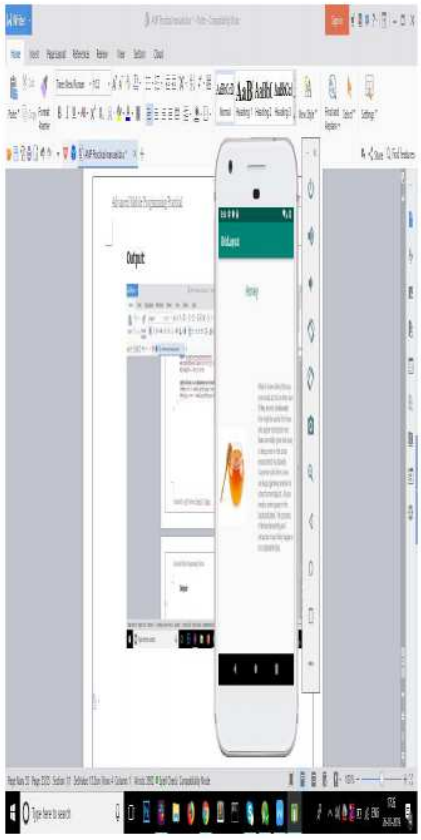
```
            var inflator =
context!!.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
            var foodView = inflator.inflate(R.layout.food_entry, null)
            foodView.imgFood.setOnClickListener {

                val intent = Intent(context, FoodDetailsActivity::class.java)
                intent.putExtra("name", food.name!!)
                intent.putExtra("description", food.description!!)
                intent.putExtra("image", food.image!!)
                context!!.startActivity(intent)
            }
            foodView.imgFood.setImageResource(food.image!!)
            foodView.tvName.text = food.name!!

            return foodView
        }
    }
}
```

4.6.3.2 Item Details Activity Class

FooddetailsActivity.kt

```
package com.harshali.gridlayout

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_food_details.*

class FoodDetailsActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_food_details)
        val bundle = intent.extras

        imgFoodDetails.setImageResource(bundle.getInt("image"))
        tvName.text = bundle.getString("name")
        tvDetails.text = bundle.getString("description")
    }
}
```

OUTPUT



PRACTICAL 5
Programming UI elements

Aim : Design App With UI:

5.1 Activity_main.xml

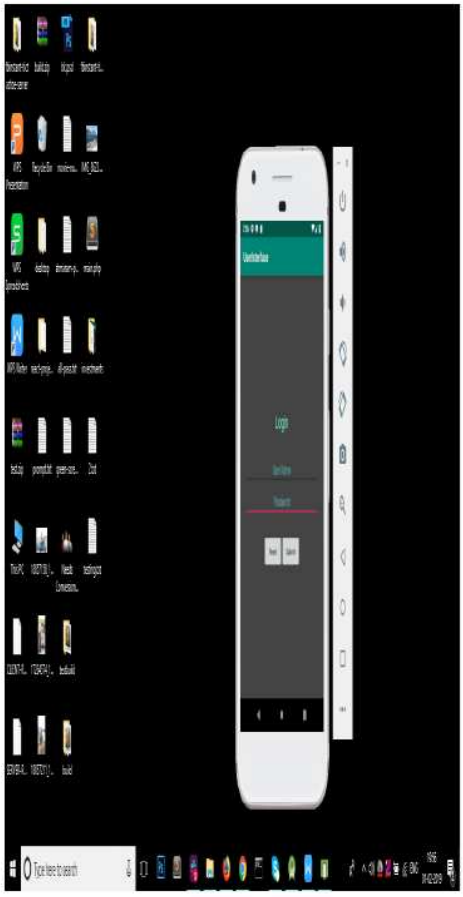
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

  <LinearLayout
      android:id="@+id/ll_main_layout"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:gravity="center"
      android:background="#444444"
      android:padding="25dp"
      android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25dp"
        android:textColor="#6dffbf"
        android:padding="30dp"
        android:text="Login"/>
```

```
    <EditText
        android:id="@+id/et_user_name"
        android:hint="User Name"
        android:textColor="#6bfff7"
        android:textColorHint="#52afaa"
        android:textAlignment="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <EditText
        android:id="@+id/et_password"
        android:hint="Password"
        android:textColor="#6bfff7"
        android:textColorHint="#52afaa"
        android:textAlignment="center"
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:padding="25dp"
        android:orientation="horizontal">
      <Button
          android:id="@+id/btn_reset"
          android:text="Reset"
          android:textAllCaps="false"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
      <Button
          android:id="@+id/btn_submit"
          android:text="Submit"
          android:textAllCaps="false"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
    </LinearLayout>
  </LinearLayout>

</LinearLayout>
```

5.2 MainActivity.kt

```
package com.shaileshm.userinterface

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast




class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // get reference to all views
        var et_user_name = findViewById(R.id.et_user_name) as EditText
        var et_password = findViewById(R.id.et_password) as EditText
        var btn_reset = findViewById(R.id.btn_reset) as Button
        var btn_submit = findViewById(R.id.btn_submit) as Button

        btn_reset.setOnClickListener {
            // clearing user_name and password edit text views on reset button click
            et_user_name.setText("")
            et_password.setText("")
        }

        // set on-click listener
        btn_submit.setOnClickListener {
            val user_name = et_user_name.text;
            val password = et_password.text;
            Toast.makeText(this@MainActivity, user_name,
Toast.LENGTH_LONG).show()

            // your code to validate the user_name and password combination
            // and verify the same

        }
    }
}
```

5.3 OUTPUT



PRACTICAL 6
Programming menus, dialog, dialog fragments

6.A dialog
6.A.1 activity_main.xml

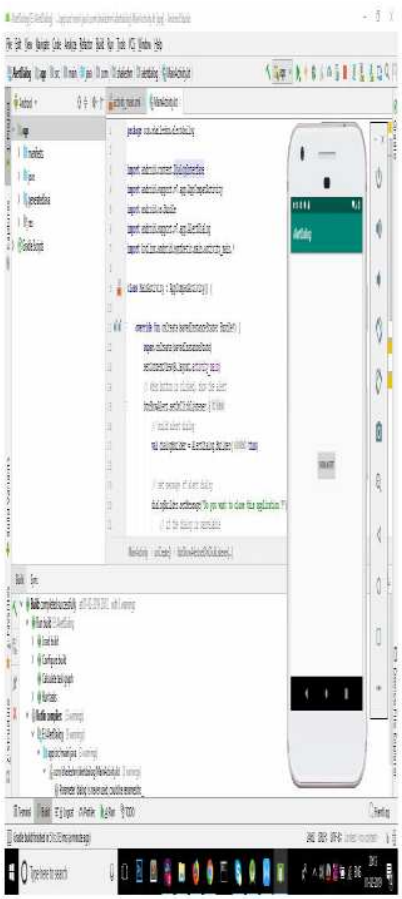
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

  <Button
      android:id="@+id/btnShowAlert"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Show Alert"
      android:gravity="center_vertical"
      />

</LinearLayout>
```

6.A.2 MainActivity.kt

```
package com.shaileshm.alertdailog

import android.content.DialogInterface
```

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.support.v7.app.AlertDialog
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // when button is clicked, show the alert
        btnShowAlert.setOnClickListener {
            // build alert dialog
            val dialogBuilder = AlertDialog.Builder(this)

            // set message of alert dialog
            dialogBuilder.setMessage("Do you want to close this application ?")
                // if the dialog is cancelable
                .setCancelable(false)
                // positive button text and action
                .setPositiveButton("Proceed", DialogInterface.OnClickListener {
                        dialog, id -> finish()
                })
                // negative button text and action
                .setNegativeButton("Cancel", DialogInterface.OnClickListener {
                        dialog, id -> dialog.cancel()
                })

            // create dialog box
            val alert = dialogBuilder.create()
            // set title for alert dialog box
            alert.setTitle("AlertDialogExample")
            // show alert dialog
            alert.show()
        }
    }
}
```

6.A.3 OUTPUT:



6.B menus

6.B.1 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/root_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="#f6ffe5"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="25dp"
android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
        android:textColor="#168ace"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:text="Tap an item from action bar menu."
    />
  </LinearLayout>
```

6.B.2 toolbar_menu.xml

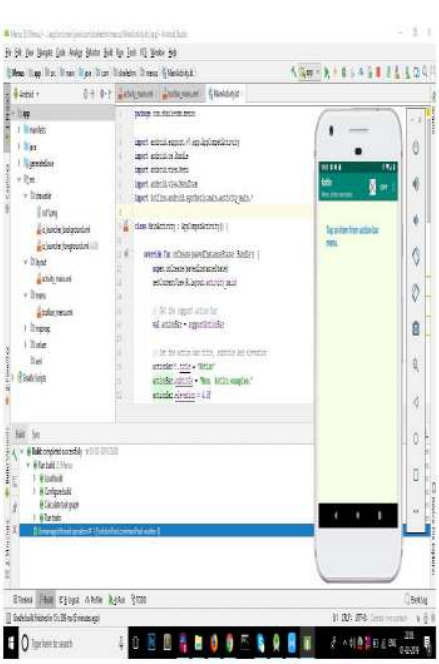
```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
      android:id="@+id/action_cut"
      android:title="Cut"
      app:showAsAction="ifRoom|withText"
      android:icon="@drawable/cut1"   />
  <item
      android:id="@+id/action_copy"
      android:title="Copy"
      app:showAsAction="always|withText"   />
  <item
      android:id="@+id/action_paste"
      android:title="Paste"
      app:showAsAction="ifRoom"   />
  <item
      android:id="@+id/action_new"
      android:title="New"
      app:showAsAction="ifRoom"   />
</menu>
```

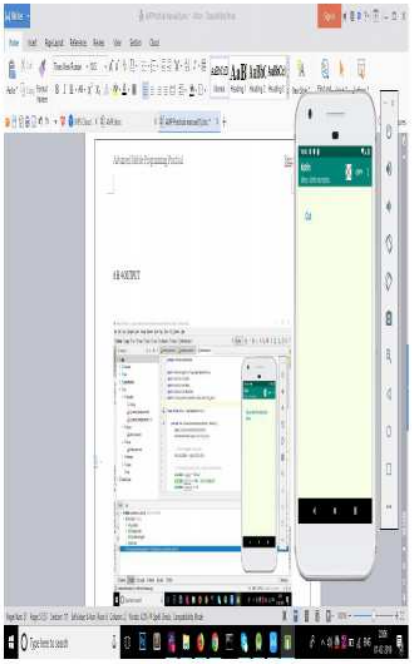
6.B.3 MainActivity.kt

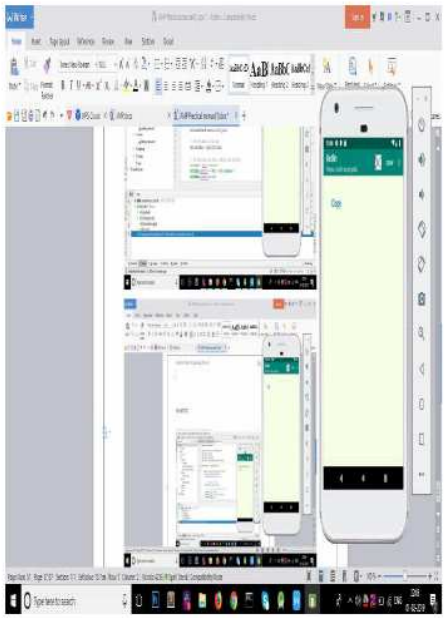
```
package com.shaileshm.menus

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

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

    // Get the support action bar
    val actionBar = supportActionBar
```

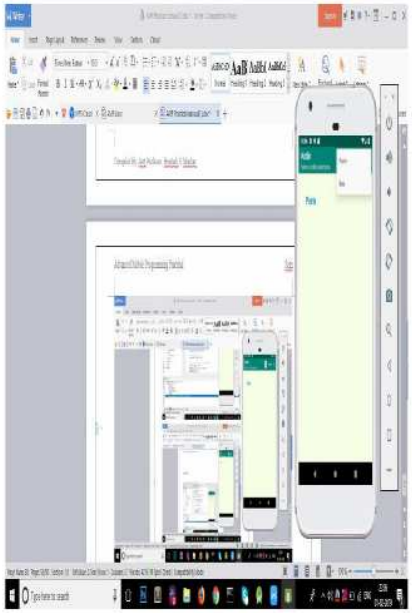
```
        // Set the action bar title, subtitle and elevation
        actionBar!!.title = "Kotlin"
        actionBar.subtitle = "Menu  kotlin examples."
        actionBar.elevation = 4.0F
    }
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        // Inflate the menu to use in the action bar
        val inflater = menuInflater
        inflater.inflate(R.menu.toolbar_menu, menu)
        return super.onCreateOptionsMenu(menu)
    }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        // Handle presses on the action bar menu items
        when (item.itemId) {
            R.id.action_cut -> {
                text_view.text = "Cut"
                return true
            }
            R.id.action_copy -> {
                text_view.text = "Copy"
                return true
            }
            R.id.action_paste -> {
                text_view.text = "Paste"
                return true
            }
            R.id.action_new -> {
                text_view.text = "New"
                return true
            }
        }
        return super.onOptionsItemSelected(item)
    }

}
```

6.B.4 OUTPUT









PRACTICAL 7
Programs on Intents, Events Listeners and Adapters

Note: Refer Grid layout code for Events Listeners, Adapters and for Intent GUI code

Practical 8
Programs on Services, notification and broadcast receivers

Step 1. Create an android app BroadcastDemo

Step2 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="com.example.student.internetconnection.MainActivity">
```

```
   <Button
      android:id="@+id/checkInternet"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:background="@color/colorPrimaryDark"
      android:text="Check Connection"
      android:textColor="#fff" />
</RelativeLayout>
```

Step 3 MainActivity.kt

```
package com.example.student.internetconnection

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button;
import android.widget.Toast;

class MainActivity : AppCompatActivity() {
   var button: Button? = null
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      button = findViewById<View>(R.id.checkInternet) as Button
      button!!.setOnClickListener(View.OnClickListener {
         //Check for Internet Connection first
         if (AppStatus.getInstance(applicationContext).isOnline) {

                        Toast.makeText(applicationContext, "WiFi/Mobile
Networks Connected!", Toast.LENGTH_SHORT).show()
         } else {

            Toast.makeText(applicationContext, "Ooops! No WiFi/Mobile
Networks Connected!", Toast.LENGTH_SHORT).show()
         }
      })
   }
}
```

Step 4 AppStatus.kt

```
package com.example.student.internetconnection

import android.annotation.SuppressLint
import android.content.Context
import android.net.ConnectivityManager
import android.net.NetworkInfo
import android.util.Log

class AppStatus {
   internal lateinit var connectivityManager: ConnectivityManager
   internal var wifiInfo: NetworkInfo? = null
   internal var mobileInfo: NetworkInfo? = null
   internal var connected = false

   val isOnline: Boolean
      @SuppressLint("MissingPermission")
      get() {
         try {
            connectivityManager =
context.getSystemService(Context.CONNECTIVITY_SERVICE) as
ConnectivityManager

            val networkInfo = connectivityManager.activeNetworkInfo
            connected = networkInfo != null && networkInfo.isAvailable &&
                  networkInfo.isConnected
            return connected

         } catch (e: Exception) {
            println("CheckConnectivity Exception: " + e.message)
            Log.v("connectivity", e.toString())
         }

         return connected
      }

   companion object {
```

```
    internal lateinit var context: Context
    /**
     * We use this class to determine if the application has been connected to
either WIFI Or Mobile
     * Network, before we make any network request to the server.
     *       *
     * The class uses two permission - INTERNET and ACCESS NETWORK
STATE, to determine the user's
     * connection stats
     */

    private val instance = AppStatus()

    fun getInstance(ctx: Context): AppStatus {
      context = ctx.applicationContext
      return instance
    }
  }
}
```

Step 5 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.student.internetconnection">
  <uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
```

```
        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>
```

Step 6 OUTPUT

PRACTICAL 9
AIM: Database Programming with SQLite

Compiled By: Astt Professor. Harshali S. Mankar

9.1 activity_main.xml:

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

  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="SQLite Tutorial - User Management"
      android:textSize="20dp"
      android:padding="10dp" />

  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
    <EditText
        android:id="@+id/edittext_userid"
        android:hint="User ID"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/edittext_name"
        android:hint="User Name"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/edittext_age"
        android:hint="User Age"
        android:gravity="center"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
  <Button
        android:id="@+id/button_add_user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="addUser"
        android:text="Add" />

  <Button
        android:id="@+id/button_delete_user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="deleteUser"
        android:text="Delete" />

  <Button
        android:id="@+id/button_show_all"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="showAllUsers"
        android:text="Show All" />
</LinearLayout>
<TextView
    android:id="@+id/textview_result"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<LinearLayout
    android:id="@+id/ll_entries"
    android:padding="15dp"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```

```
    </LinearLayout>
</LinearLayout>
```

9.2 UserModel.kt

```
package com.shaileshm.sqllitedemo

class UserModel(val userid: String, val name: String, val age: String)
```

9.3 DBContract.kt

```
package com.shaileshm.sqllitedemo

import android.provider.BaseColumns

object DBContract {

    /* Inner class that defines the table contents */
    class UserEntry : BaseColumns {
        companion object {
            val TABLE_NAME = "users"
            val COLUMN_USER_ID = "userid"
            val COLUMN_NAME = "name"
            val COLUMN_AGE = "age"
        }
    }
}
```

9.4 UsersDBHelper.kt

```
package com.shaileshm.sqllitedemo

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteConstraintException
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteException
import android.database.sqlite.SQLiteOpenHelper

import java.util.ArrayList
```

```
class UsersDBHelper(context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {
   override fun onCreate(db: SQLiteDatabase) {
      db.execSQL(SQL_CREATE_ENTRIES)
   }

   override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion:
Int) {
      // This database is only a cache for online data, so its upgrade policy is
      // to simply to discard the data and start over
      db.execSQL(SQL_DELETE_ENTRIES)
      onCreate(db)
   }

   override fun onDowngrade(db: SQLiteDatabase, oldVersion: Int,
newVersion: Int) {
      onUpgrade(db, oldVersion, newVersion)
   }

   @Throws(SQLiteConstraintException::class)
   fun insertUser(user: UserModel): Boolean {
      // Gets the data repository in write mode
      val db = writableDatabase

      // Create a new map of values, where column names are the keys
      val values = ContentValues()
      values.put(DBContract.UserEntry.COLUMN_USER_ID, user.userid)
      values.put(DBContract.UserEntry.COLUMN_NAME, user.name)
      values.put(DBContract.UserEntry.COLUMN_AGE, user.age)

      // Insert the new row, returning the primary key value of the new row
      val newRowId = db.insert(DBContract.UserEntry.TABLE_NAME, null,
values)

      return true
   }

   @Throws(SQLiteConstraintException::class)
   fun deleteUser(userid: String): Boolean {
      // Gets the data repository in write mode
      val db = writableDatabase
      // Define 'where' part of query.
      val selection = DBContract.UserEntry.COLUMN_USER_ID + " LIKE
```

```
?"
        // Specify arguments in placeholder order.
        val selectionArgs = arrayOf(userid)
        // Issue SQL statement.
        db.delete(DBContract.UserEntry.TABLE_NAME, selection,
selectionArgs)

        return true
    }

    fun readUser(userid: String): ArrayList<UserModel> {
        val users = ArrayList<UserModel>()
        val db = writableDatabase
        var cursor: Cursor? = null
        try {
            cursor = db.rawQuery("select * from " +
DBContract.UserEntry.TABLE_NAME + " WHERE " +
DBContract.UserEntry.COLUMN_USER_ID + "='" + userid + "'", null)
        } catch (e: SQLiteException) {
            // if table not yet present, create it
            db.execSQL(SQL_CREATE_ENTRIES)
            return ArrayList()
        }

        var name: String
        var age: String
        if (cursor!!.moveToFirst()) {
            while (cursor.isAfterLast == false) {
                name =
cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_N
AME))
                age =
cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_A
GE))

                users.add(UserModel(userid, name, age))
                cursor.moveToNext()
            }
        }
        return users
    }

    fun readAllUsers(): ArrayList<UserModel> {
```

```
        val users = ArrayList<UserModel>()
        val db = writableDatabase
        var cursor: Cursor? = null
        try {
            cursor = db.rawQuery("select * from " +
DBContract.UserEntry.TABLE_NAME, null)
        } catch (e: SQLiteException) {
            db.execSQL(SQL_CREATE_ENTRIES)
            return ArrayList()
        }

        var userid: String
        var name: String
        var age: String
        if (cursor!!.moveToFirst()) {
            while (cursor.isAfterLast == false) {
                userid =
cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_U
SER_ID))
                name =
cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_N
AME))
                age =
cursor.getString(cursor.getColumnIndex(DBContract.UserEntry.COLUMN_A
GE))

                users.add(UserModel(userid, name, age))
                cursor.moveToNext()
            }
        }
        return users
    }

    companion object {
        // If you change the database schema, you must increment the database
version.
        val DATABASE_VERSION = 1
        val DATABASE_NAME = "FeedReader.db"

        private val SQL_CREATE_ENTRIES =
            "CREATE TABLE " + DBContract.UserEntry.TABLE_NAME + " ("
+
                    DBContract.UserEntry.COLUMN_USER_ID + " TEXT
```

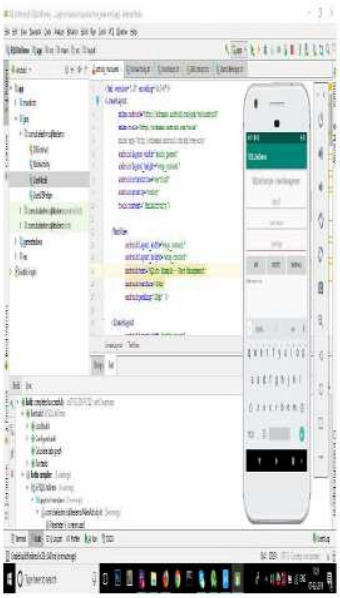
```
PRIMARY KEY," +
                DBContract.UserEntry.COLUMN_NAME + " TEXT," +
                DBContract.UserEntry.COLUMN_AGE + " TEXT)"

        private val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS "
+ DBContract.UserEntry.TABLE_NAME
    }

}
```

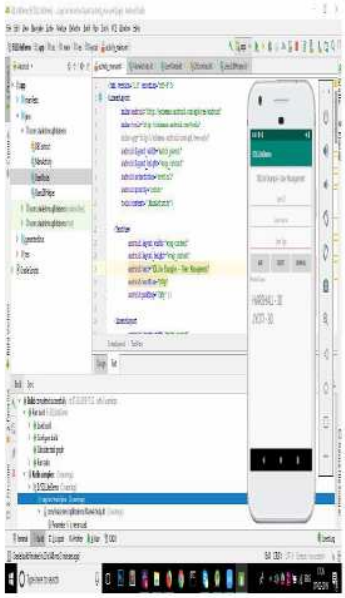
9.5 MainActivity.kt

```
package com.shaileshm.sqllitedemo

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.TextView
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    lateinit var usersDBHelper : UsersDBHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        usersDBHelper = UsersDBHelper(context = this)
    }

    fun addUser(v:View){
        var userid = this.edittext_userid.text.toString()
        var name = this.edittext_name.text.toString()
        var age = this.edittext_age.text.toString()
        var result = usersDBHelper.insertUser(UserModel(userid = userid,name =
name,age = age))
        //clear all edittext s
        this.edittext_age.setText("")
        this.edittext_name.setText("")
        this.edittext_userid.setText("")
        this.textview_result.text = "Added user : "+result
        this.ll_entries.removeAllViews()
    }
```

```
    fun deleteUser(v:View){
        var userid = this.edittext_userid.text.toString()
        val result = usersDBHelper.deleteUser(userid)
        this.textview_result.text = "Deleted user : "+result
        this.ll_entries.removeAllViews()
    }

    fun showAllUsers(v:View){
        var users = usersDBHelper.readAllUsers()
        this.ll_entries.removeAllViews()
        users.forEach {
            var tv_user = TextView(this)
            tv_user.textSize = 30F
            tv_user.text = it.name.toString() + " - " + it.age.toString()
            this.ll_entries.addView(tv_user)
        }
        this.textview_result.text = "Fetched " + users.size + " users"
    }
}
```

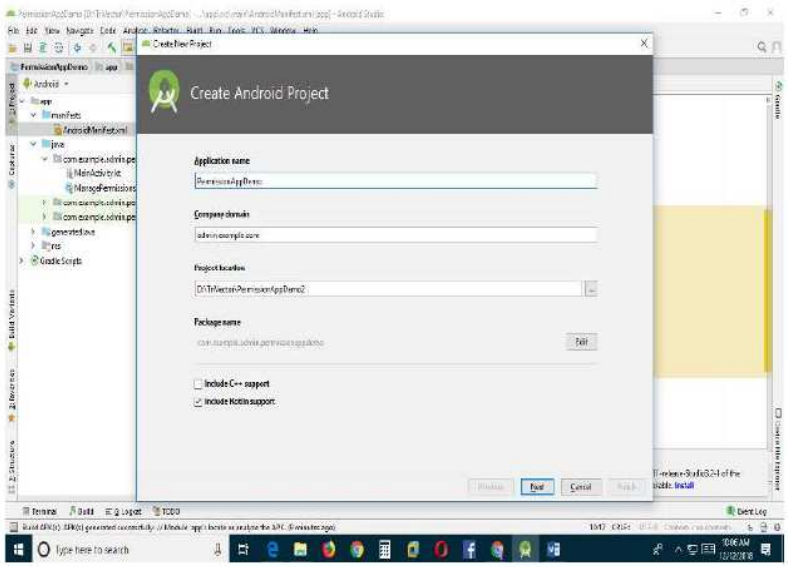
9.6 OUTOUT





PRACTICAL 10
Aim :Programming Security and permissions

Create a new project in android studio



An app must publicize the permissions it requires by including <uses-permission> tags in the app manifest.

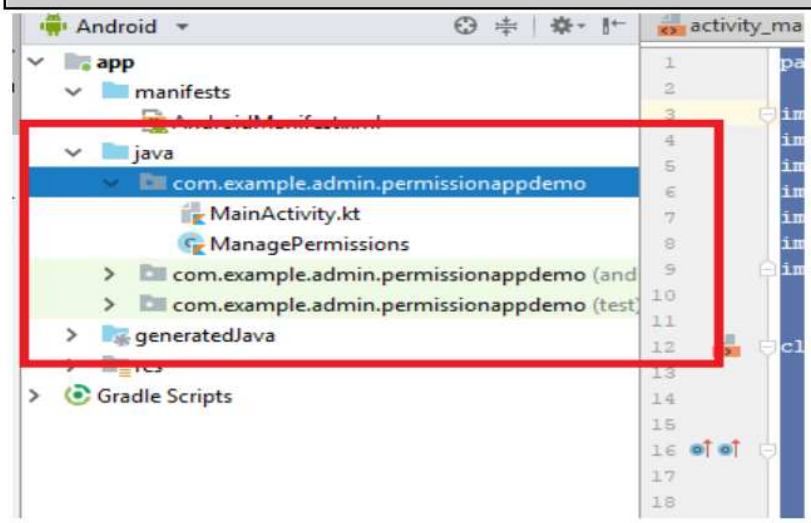
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
```

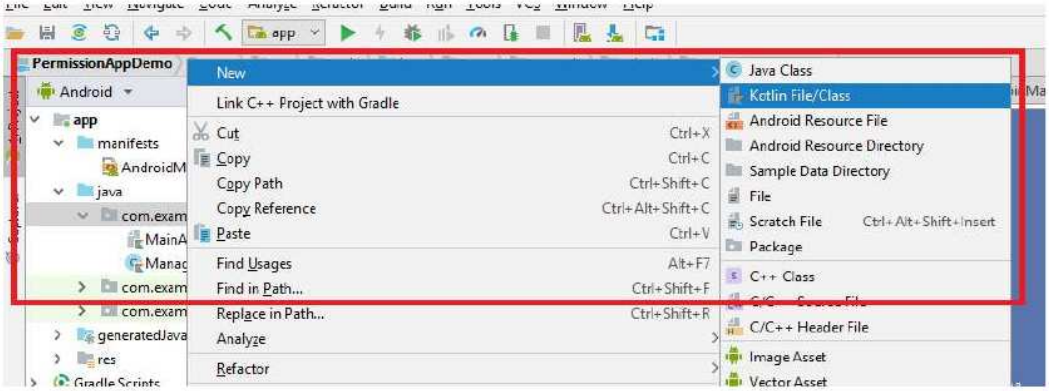
MainActivity.kt

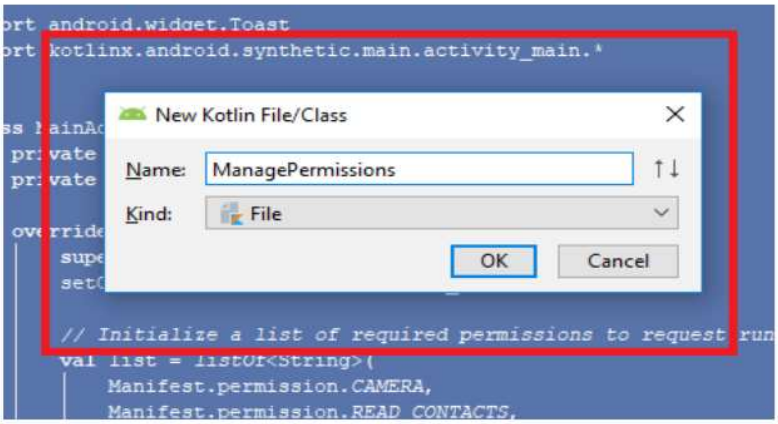
```
package com.shaileshm.permisiondemo
import android.content.Context
import android.os.Build
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*
class MainActivity : AppCompatActivity() {
   private val PermissionsRequestCode = 123
   private lateinit var managePermissions:ManagePermissions
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      // Initialize a list of required permissions to request runtime
      val list = listOf<String>(
         android.Manifest.permission.CAMERA,
         android.Manifest.permission.READ_CALENDAR,
        android.Manifest.permission.READ_CONTACTS,
       android.Manifest.permission.READ_EXTERNAL_STORAGE,
        android.Manifest.permission.SEND_SMS

      )

      // Initialize a new instance of ManagePermissions class
      managePermissions =
ManagePermissions(this,list,PermissionsRequestCode)

      // Button to check permissions states
      button.setOnClickListener{

         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
            managePermissions.checkPermissions()
      }
   }


   // Receive the permissions request result
   override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<String>,
                                           grantResults: IntArray) {
      when (requestCode) {
         PermissionsRequestCode ->{
```

```
            val isPermissionsGranted = managePermissions
                .processPermissionsResult(requestCode,permissions,grantResults)

            if(isPermissionsGranted){
                // Do the task now

                toast("Permissions granted.")
            }else{
                toast("Permissions denied.")
            }
            return
        }
    }
}
}


// Extension function to show toast message
fun Context.toast(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
    }
```

Create a New Kotlin Class








```
package com.shaileshm.permisiondemo

import android.app.Activity
import android.content.pm.PackageManager
import android.support.v4.app.ActivityCompat
import android.support.v4.content.ContextCompat
import android.support.v7.app.AlertDialog

import android.widget.Toast
class ManagePermissions(val activity: Activity,val list: List<String>,val
code:Int) {

   // Check permissions at runtime
   fun checkPermissions() {
      if (isPermissionsGranted() != PackageManager.PERMISSION_GRANTED)
{
         showAlert()
      } else {
         Toast.makeText(activity, "Permissions already granted",
            Toast.LENGTH_LONG).show()
```

```
        }
    }


    // Check permissions status
    private fun isPermissionsGranted(): Int {
        // PERMISSION_GRANTED : Constant Value: 0
        // PERMISSION_DENIED : Constant Value: -1
        var counter = 0;
        for (permission in list) {
            counter += ContextCompat.checkSelfPermission(activity, permission)
        }
        return counter
    }


    // Find the first denied permission
    private fun deniedPermission(): String {
        for (permission in list) {
            if (ContextCompat.checkSelfPermission(activity, permission)
                == PackageManager.PERMISSION_DENIED) return permission
        }
        return ""
    }


    // Show alert dialog to request permissions
    private fun showAlert() {
        val builder = AlertDialog.Builder(activity)
        builder.setTitle("Need permission(s)")
        builder.setMessage("Some permissions are required to do the task.")
        builder.setPositiveButton("OK", { dialog, which -> requestPermissions()
})
        builder.setNeutralButton("Cancel", null)
        val dialog = builder.create()
        dialog.show()
    }

    // Request the permissions at run time
    private fun requestPermissions() {
        val permission = deniedPermission()
        if (ActivityCompat.shouldShowRequestPermissionRationale(activity,
```

```
permission)) {
        // Show an explanation asynchronously
        Toast.makeText(activity, "Should show an explanation.",
            Toast.LENGTH_LONG).show()

    } else {
        ActivityCompat.requestPermissions(activity, list.toTypedArray(), code)
    }
  }

  // Process permissions result
  fun processPermissionsResult(requestCode: Int, permissions: Array<String>,
                               grantResults: IntArray): Boolean {
    var result = 0
    if (grantResults.isNotEmpty()) {
      for (item in grantResults) {
        result += item
      }
    }
    if (result == PackageManager.PERMISSION_GRANTED) return true
    return false
  }
}
```

Activity_main.xml

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

  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Permission Example Demo"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
```

```
    <Button
        android:text="Request for Permission"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
android:id="@+id/button"/>

</android.support.constraint.ConstraintLayout>
```

OUTOUT: