ANDROID TUTORIAL

L - 2

3rd Sem, MCA

L A B - 1

L1	1. Create a **Hello world** android application using android studio and compile it using android SDK. Run it on both different Android Virtual Devices (AVD) and your own mobile. 2. Create an android application to display the welcome message as a **toast**. (A) Default toast display (B) Toast on button click 3. Create an android application and understand the application **life-cycle** in different scenarios.

CONTENT

- **Mini Project details,**
- Resources,
- Android Components,
- Android Layout,
- UI components
  - Button,
  - TextView,
  - EditView.

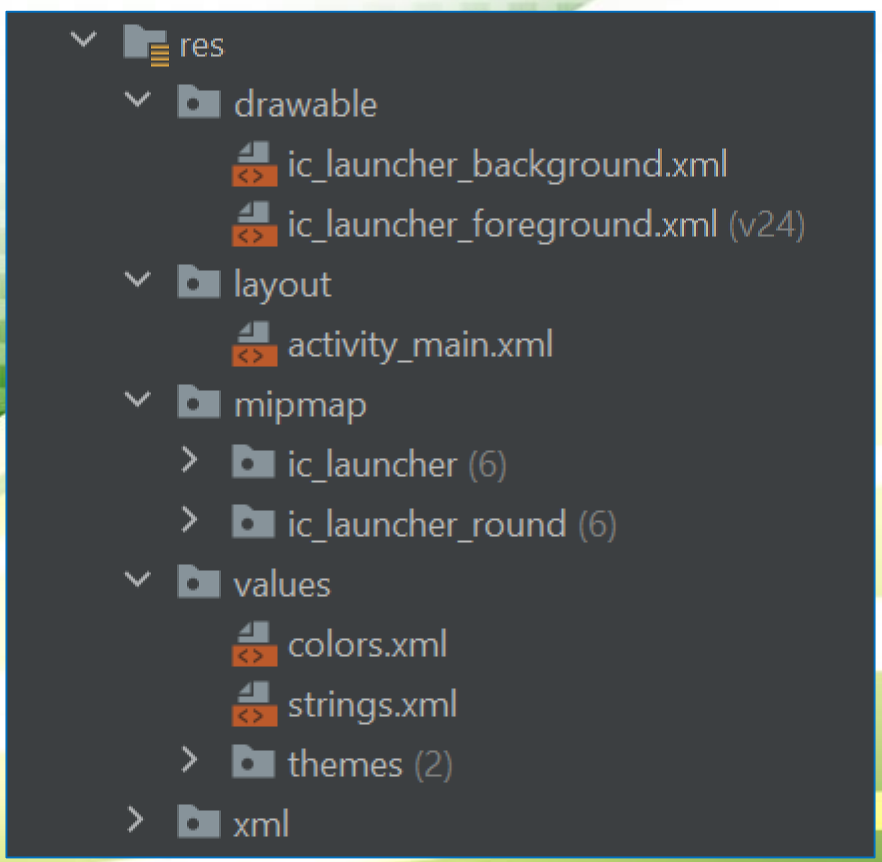
ANDROID MENIFEST

- Compulsorily required xml file for all the android application and located inside the root directory.
- *Contains information of app package*, including components of the application such as activities, services, broadcast receivers, content providers etc.
- **<manifest> :** root element; contains **package** attribute that describes package name of the activity class.
- **<application> :** subelement of manifest; includes namespace declaration. contains several subelements that declares the application component. Commonly used attributes: **icon**, **label**, **theme** etc.
- **<activity> :** subelement of application and represents an activity that must be defined in AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.
- **<intent-filter> :** sub-element of activity that describes type of intent to which activity, service or broadcast receiver can respond to.
- **<action> :** It adds an action for the intent-filter. The intent-filter must have at least one action element.
- **<category> :** It adds a category name to an intent-filter.

ANDROID UI COMPONENTS

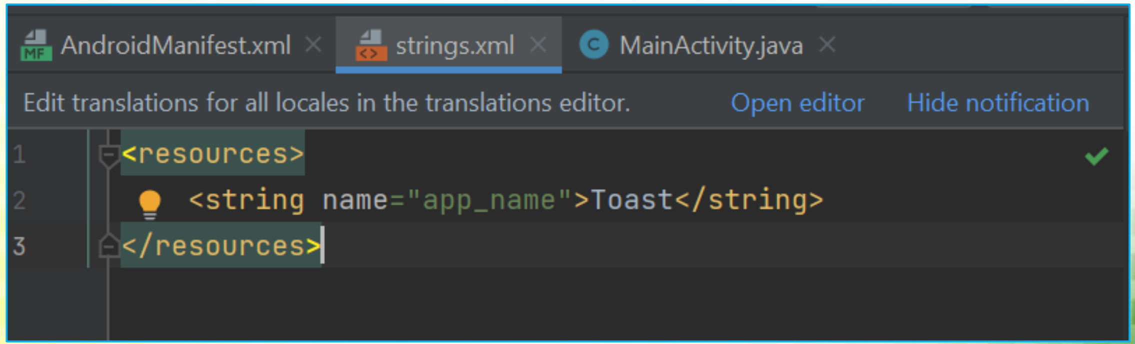
Resource Directory

- Layout
- String
- Color
- Drawable
- MipMap



```
res
  drawable
    ic_launcher_background.xml
    ic_launcher_foreground.xml (v24)
  layout
    activity_main.xml
  mipmap
    ic_launcher (6)
    ic_launcher_round (6)
  values
    colors.xml
    strings.xml
    themes (2)
  xml
```

ANDROID APP NAME



AndroidManifest.xml × | strings.xml × | MainActivity.java ×

Edit translations for all locales in the translations editor. Open editor Hide notification

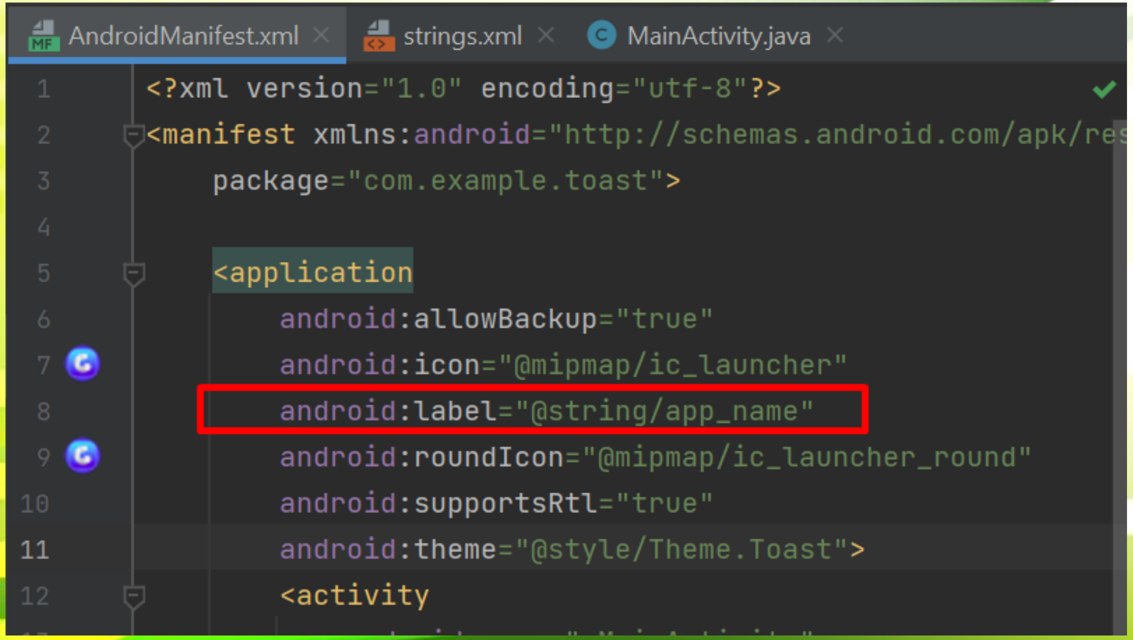
```xml
<resources>
    <string name="app_name">Toast</string>
</resources>
```

Step 1:

- **app > manifests > AndroidManifest.xml**
- change **android:label** field in app name

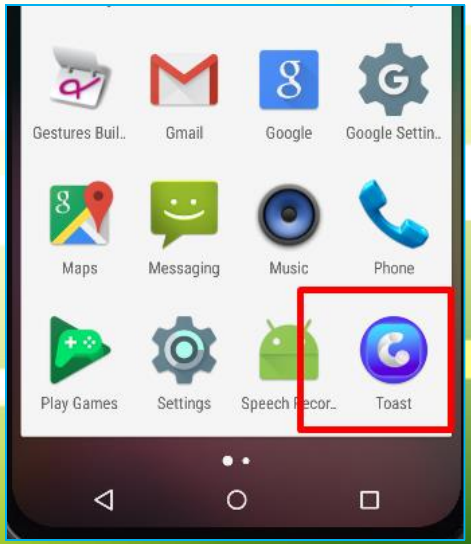
Step 2:

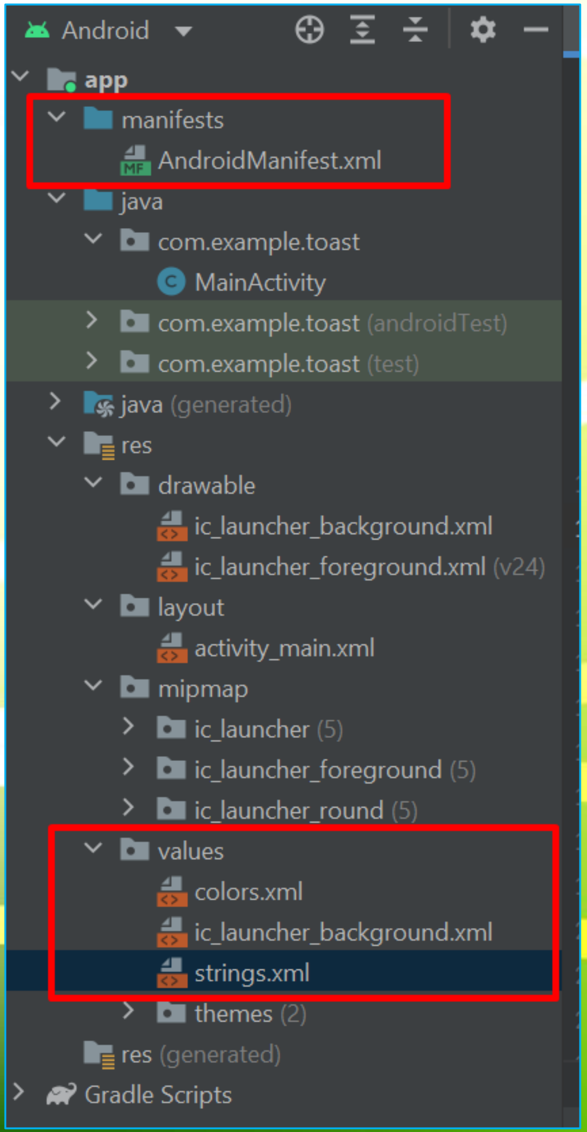
- **res > values > strings.xml** file
- change string resources value to **AppName.**



AndroidManifest.xml × | strings.xml × | MainActivity.java ×

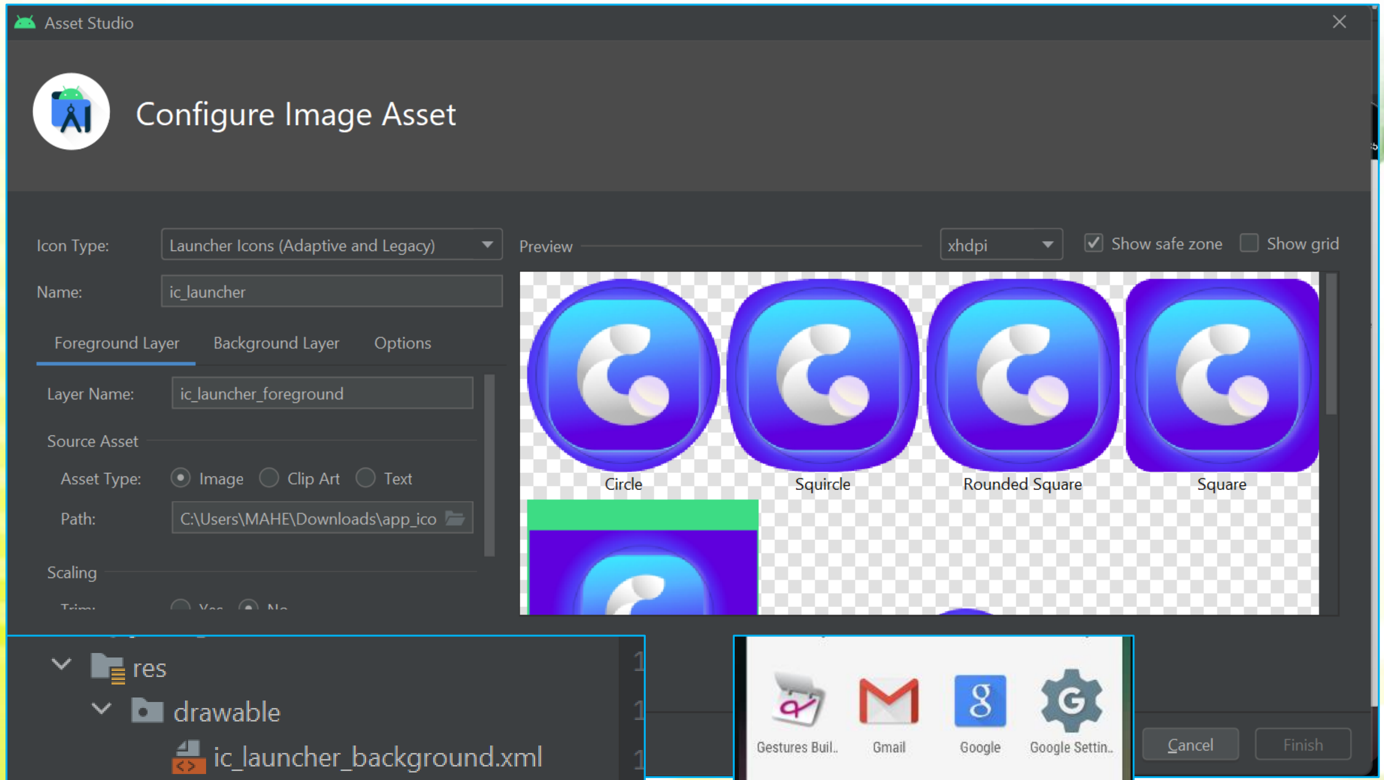
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res
    package="com.example.toast">

    <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/Theme.Toast">
        <activity
```





```
Android
app
  manifests
    AndroidManifest.xml
  java
    com.example.toast
      MainActivity
    com.example.toast (androidTest)
    com.example.toast (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
      activity_main.xml
    mipmap
      ic_launcher (5)
      ic_launcher_foreground (5)
      ic_launcher_round (5)
    values
      colors.xml
      ic_launcher_background.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
```

ANDROID APP ICON


Asset Studio

Configure Image Asset

Icon Type: Launcher Icons (Adaptive and Legacy)

Name: ic_launcher

Foreground Layer | Background Layer | Options

Layer Name: ic_launcher_foreground

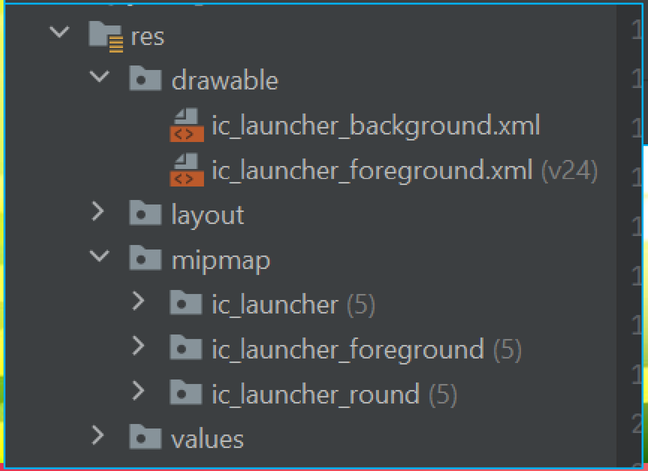
Source Asset

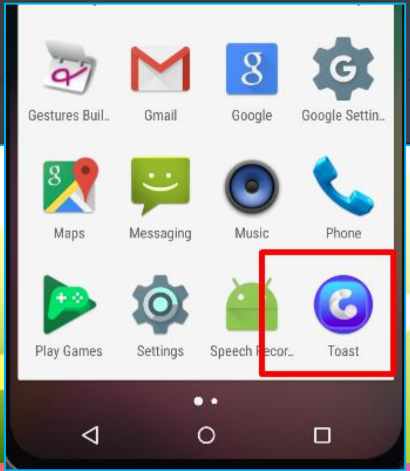
Asset Type: [✓ Image] Clip Art Text

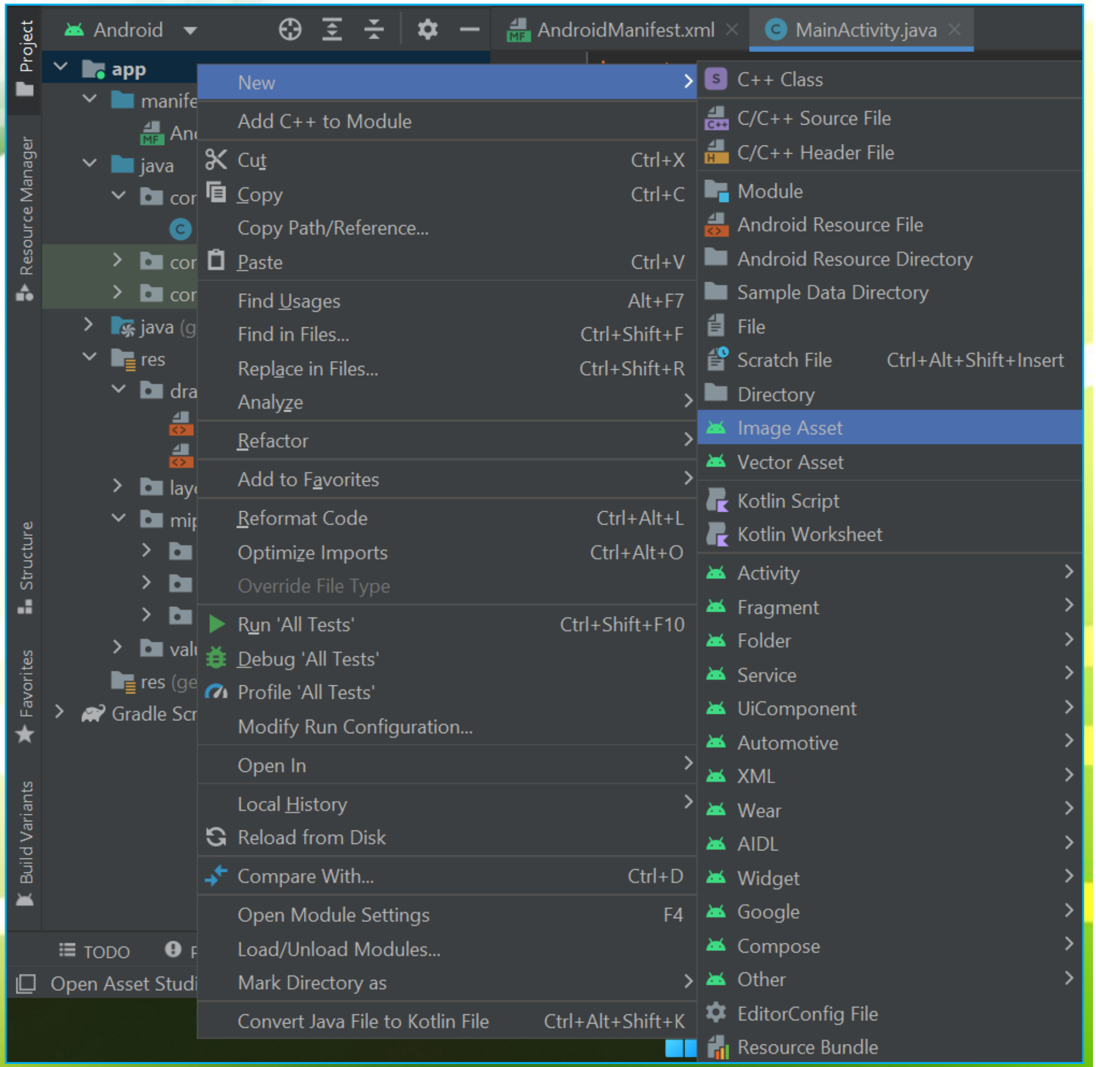
Path: C:\Users\MAHE\Downloads\app_ico

Scaling

Preview: xhdpi [✓ Show safe zone] Show grid


res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
mipmap
ic_launcher (5)
ic_launcher_foreground (5)
ic_launcher_round (5)
values





ANDROID COMPONENTS

- Essential building blocks (piece of code) of an Android application that has a well defined life cycle.
- These components are loosely coupled by application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

- **Activities:** They dictate the UI and handle user interaction to the smart phone screen.
- **Services:** They handle background processing associated with an application.
- **Broadcast Receivers:** They handle communication between Android OS and applications.
- **Content Providers:** They handle data and database management issues.
- **Manifest:** Manifest file plays an integral role as it provides the essential information about app to Android system, which the system must have before it can run any of the app's code.

ANDROID COMPONENTS

Activities

- An activity represents a single screen with a user interface,
- Activity performs actions on the screen.
- 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 −

```
public class MainActivity extends Activity {
}
```

ANDROID COMPONENTS

Services

- Service component runs in background to perform long-running operations.
- Example, a service might play music in background while user is in different application, or it might fetch data over network without blocking user interaction with an activity.
- Two types of services: local and remote.
    - Local service is accessed from within the application.
    - Remote service is accessed remotely from other applications running on same device.
- A service is implemented as a subclass of Service class as follows −

```
public class MyService extends Service {
}
```

ANDROID COMPONENTS

Content Providers

- Used to share data between the applications.
- Content provider component supplies data from one application to others on request.
- Such requests are handled by methods of ContentResolver class.
- Data may be stored in file system, database or somewhere else entirely.

- Implemented as subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends  ContentProvider {
   public void onCreate(){}
}
```

ANDROID COMPONENTS

Broadcast Receivers

- Broadcast Receivers simply respond to broadcast messages from other applications or from the system.
- Example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to device and is available for them to use.
  - Broadcast receiver who will intercept this communication and will initiate appropriate action.
- Subclass of BroadcastReceiver class; and each message is broadcasted as an Intent object.

```
public class MyReceiver  extends  BroadcastReceiver {
   public void onReceive(context,intent){}
}
```

ANDROID COMPONENTS

Fragments

- Represents a portion of user interface in an Activity.
- Fragments are like parts of activity. An activity can display one or more fragments on screen at same time.

Views

- UI elements that are drawn on-screen including buttons, lists forms etc. Anything that user see is a view.

Layouts

- View hierarchies that control screen format and appearance of the views.

Resources

- External elements, such as strings, constants and drawable pictures.

Manifest

- Configuration file for the application.
- It contains informations about activities, content providers, permissions etc.

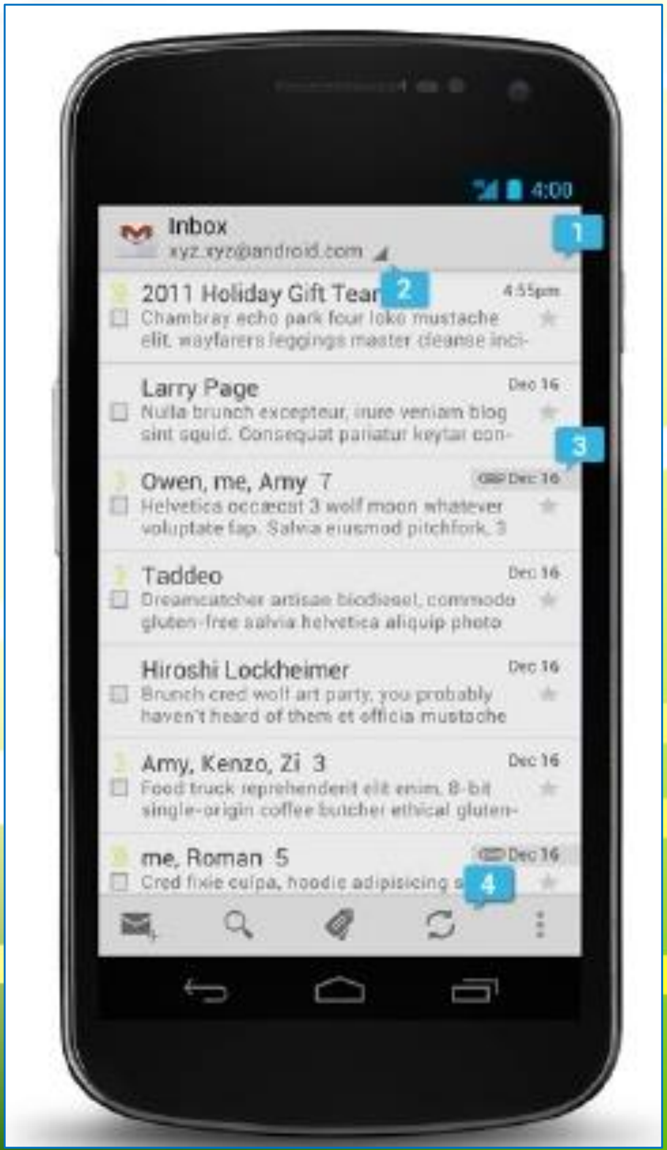
ANDROID COMPONENTS

Intents

- Messages wiring components together.
- Intent is used to invoke components. It is mainly used to:
  - Start the service
  - Launch an activity
  - Display a web page
  - Display a list of contacts
  - Broadcast a message
  - Dial a phone call etc.

ANDROID UI COMPONENTS

- Typical UI of any Android application consists of these components:
  1. Main Action Bar
  2. View Control
  3. Content Area
  4. Split Action Bar
- These components play major role while developing complex application.
- Another important factor that helps in customizing UI design would be view component.



2011 Holiday Gift Team — 4:55pm
Chambray echo park four loko mustache elit. wayfarers leggings master cleanse inci-

Larry Page — Dec 16
Nulla brunch excepteur, irure veniam blog sint squid. Consequat pariatur keytar con-

Owen, me, Amy 7 — Dec 16
Helvetica occaecat 3 wolf moon whatever voluptate fap. Salvia eiusmod pitchfork, 3

Taddeo — Dec 16
Dreamcatcher artisan biodiesel, commodo gluten-free salvia helvetica aliquip photo

Hiroshi Lockheimer — Dec 16
Brunch cred wolf art party, you probably haven't heard of them et officia mustache

Amy, Kenzo, Zi 3 — Dec 16
Food truck reprehenderit elit enim, 8-bit single-origin coffee butcher ethical gluten-

me, Roman 5 — Dec 16
Cred fixie culpa, hoodie adipisicing s

VIEWS & VIEWGROUP

- **View objects** are basic building blocks of UI elements in Android.
- A View is a simple rectangle box which response to user's actions.
- View refers to the android.view.View class, which is the base class of all UI classes.
- Examples are EditText, Button, CheckBox, etc..

- **ViewGroup** is the invisible container which holds View and ViewGroup.
- ViewGroup is the base class for the Layouts.
- Example, LinearLayout is the ViewGroup that contains Button(View), and other Layouts also.

LAYOUT

Defines user interface for an app or activity and it will hold UI elements that will appear to user.

- **Linear Layout:** view group that aligns all child components in single direction, vertically or horizontally.
- **Relative Layout:** view group that displays child views in relative positions.
- **Table Layout:** view that groups views into rows and columns.
- **Absolute Layout** enables to specify the exact location of its children.
- **Frame Layout:** placeholder on screen that can be used to display a single view.
- **Constraint Layout:** View group which allows to position and size widgets in a flexible way.
- **List View:** view group that displays a list of scrollable items.
- **Grid View:** View Group that displays items in a two-dimensional, scrollable grid.

LAYOUT

Linear layout

- used to place one element on each line.
- all elements will be placed in an orderly top-to-bottom fashion.
- very widely-used layout for creating forms on Android.
- Its a view group that aligns all children in a single direction, vertically or horizontally.

Absolute layout

- Developer can specify exact coordinates of each control to be placed.
- can give exact X and Y coordinates of each control.

LAYOUT

Relative layout

- Its a ViewGroup that displays child views in relative positions.
- Developer can specify position of elements in relation to other elements, or parent container.

Table layout

- Can create a table with rows and columns and place elements within them.
- Each row for one or more elements.
- Command <TableRow> to create a new table layout.

Frame layout

- Used to show one item on each screen.
- Placeholder on the screen that can be used to display a single view each.

LAYOUT ATTRIBUTES

- Each layout has a set of attributes which define the visual properties of that layout.
- Few common attributes among all layouts and few attributes which are specific to that layout.

Attribute	Description
android:id	It is used to uniquely identify the view and ViewGroups
android:layout_width	It is used to define the width for View and ViewGroup elements in a layout
android:layout_height	It is used to define the height for View and ViewGroup elements in a layout
android:layout_marginLeft	It is used to define the extra space in the left side for View and ViewGroup elements in a layout
android:layout_marginRight	It is used to define the extra space in right side for View and ViewGroup elements in layout
android:layout_marginTop	It is used to define the extra space on top for View and ViewGroup elements in layout
android:layout_marginBottom	It is used to define the extra space in the bottom side for View and ViewGroup elements in a layout
android:paddingLeft	It is used to define the left side padding for View and ViewGroup elements in layout files
android:paddingRight	It is used to define the right side padding for View and ViewGroup elements in layout files
android:paddingTop	It is used to define padding for View and ViewGroup elements in layout files on top side
android:paddingBottom	It is used to define the bottom side padding for View and ViewGroup elements in layout files
android:layout_gravity	It is used to define how child Views are positioned

UNITS OF MEASUREMENT

When specifying size of an element on an Android UI, following units of measurement are used.

Unit	Description
dp	Density Independent Pixel. 1dp is equivalent to one pixel on a 160dpi screen.
sp	Scale Independent Pixel. similar to dp but just that this is recommended for specifying font sizes.
pt	Point. A point is defined to be 1/72 of an inch.
px	Pixel. Corresponds to actual pixels on the screen

ANDROID UI COMPONENTS

TextView used to display text to the user.
EditText is a predefined subclass of TextView that includes rich editing capabilities.
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 can be pressed, or clicked, by the user to perform an action.
CheckBox An on/off switch that can be toggled by user.
ToggleButton An on/off button with a light indicator.
RadioButton two states: either checked or unchecked.
RadioGroup used to group together one or more RadioButtons.
ProgressBar provides visual feedback about some ongoing tasks, such as when performing a task in background.
Spinner A drop-down list that allows users to select one value from a set.
TimePicker enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
DatePicker enables users to select a date of the day.

ANDROID UI COMPONENTS

android:id="@+id/text_id"

```
<?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">

  <TextView android:id="@+id/text_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am a TextView" />
</LinearLayout>
```

```
TextView myText = (TextView) findViewById(R.id.text_id);
```

ANDROID UI COMPONENTS

- **android.widget.Button** is subclass of TextView class and CompoundButton is the subclass of Button class.
- There are different types of buttons in android : RadioButton, ToggleButton, CompoundButton etc.
- perform action on button by
  - calling listener on button or
  - adding onClick property of button in activity's xml file or
  - Implementing listerner method on class

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
button.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
            //code
          }
});
```

```
<Button
        android:onClick="methodName"
/>
```

ANDROID UI COMPONENTS

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

```
Button btn = (Button)findViewById(R.id.button1);
TextView textView11 = (TextView) findViewById(R.id.textView1);
EditText editText11 = (EditText) findViewById(R.id.editText1);
```

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String value1=editText11.getText().toString();
        textView11.setText(Integer.toString(“show value”));
}
```

L A B - 2

L2	4. Create an android application to input two numbers and display the sum on button click. a) Display result in toast & textview. b) Validate the sum action. c) Provide Reset button with its functionality. d) Customize the app name, and icon. e) Customize the UI component with color, font properties. f) Make use of String & Color resource files. g) Apply different layouts for this; Relative, Constraint and Linear.