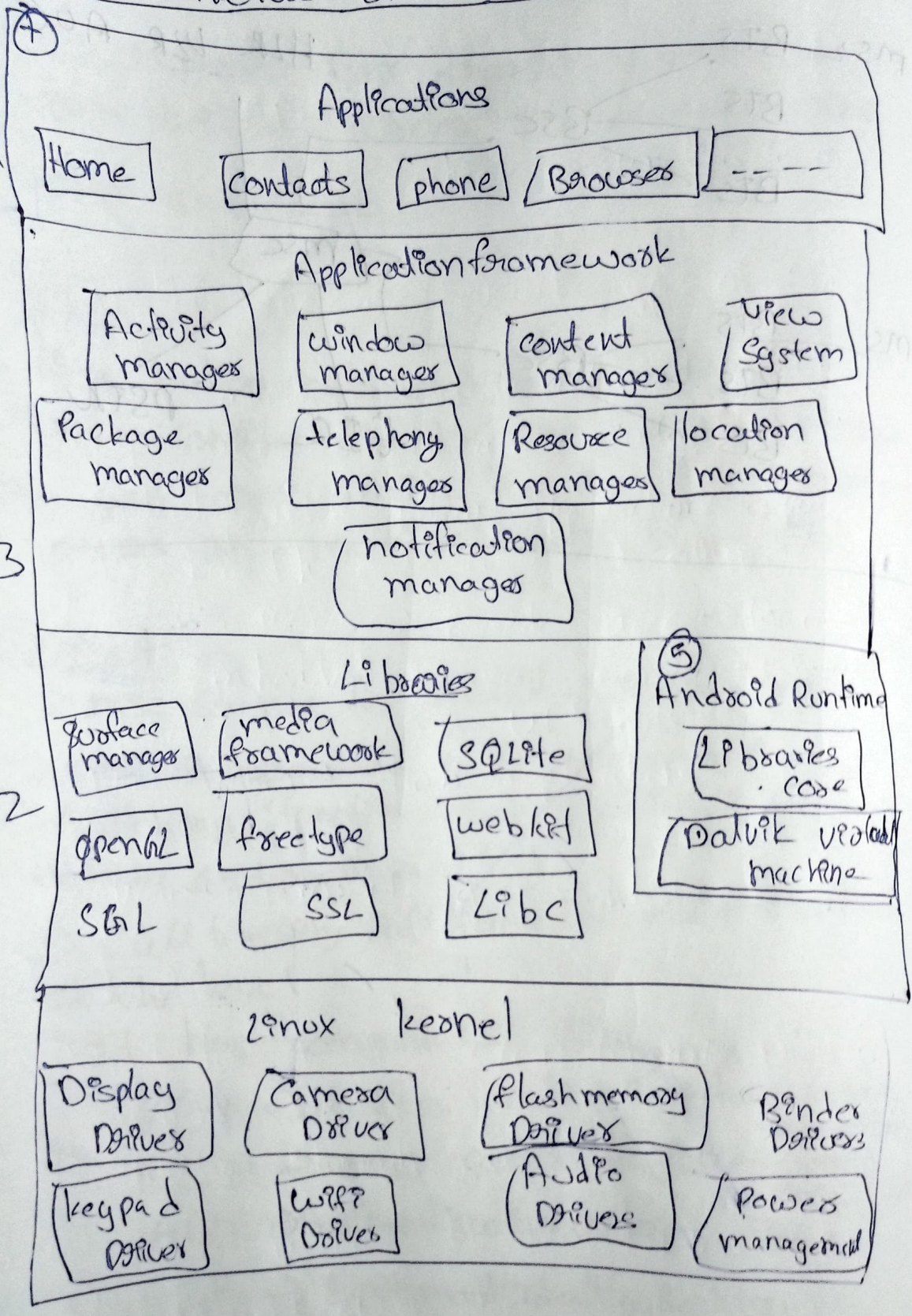
Architecture of Android



Android Applica

features of Android

- Beautiful UI
- Connectivity
- Storage.
- media support
- messaging
- web browser
- multi-touch
- wifi
- bluetooth
- multi languages
- multi-tasking.

Android Applications

- Android Apps are usually developed by the Java language using Android software Development kit.
- Ones developed, Android App can be packaged and sold out either through a store such as google play, App Store, Amazon Appstore.
- Android powers hundred of millions of mobile devices in more than 190 countries around the world
- Every day one million new Android devices are activated world wide.

Application frame work

~~Developers~~

- It is a API Interface
- the main work of the frame work is to manage the Application
- In this there are several types of frame works
- Activity manager -> controlls all aspets of the application lifecycle and activity.
- Content providers: Allows applications to publish and share the data with each other
- Resource manager: provides access to non-code-Embedded resources such as strings, color, setting and UI layouts
- Notifications: Allows applications to display notifications
- View System: It is a user interface of application

Libraries:

- Includes a set of C/C++ libraries used by components
- Interface through java
- Surface manages - handling UI windows
- 2D and 3D graphics
- media, SQLite, Browser Engine

Runtime :

Core libraries

- provide most of the functions available in the core libraries of the Java language.
  - Data structures
  - file Access
  - network Access
  - Graphics.

Dalvik virtual machine

- It's provide a Environment to the Android application.
- It is a virtual machine and used to run android applications

Linux kernel

- It is used to manage the hardware and software.
- It is also called as interface b/w hardware and Software.
- It is used to manage, camera, Display, Audio, keypad, power
- Security, process, network,

GSM Architecture:-

→ GSM stands for Global System for Mobile Communication.

→ GSM is an open and digital cellular technology used for mobile communication

→ It uses 4 different frequency bands of 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz.

→ It uses the combination of FDMA [Frequency Division Multiple Access] and TDMA [Time Division Multiple Access].

→ GSM is having 4 different sizes of cells are used in GSM

⇒ Macro cell :- Used in Base station antenna is installed.

⇒ Micro cell :- antenna height is less than the average roof level

⇒ Pico :- small cells diameter for few meters.

⇒ Umbrella :- It cover the shadow and region.

Features of GSM

→ Supports international roaming

→ Clear voice clarity

→ Ability to support multiple handled devices

→ frequency efficiency.

→ Low powered handled devices.

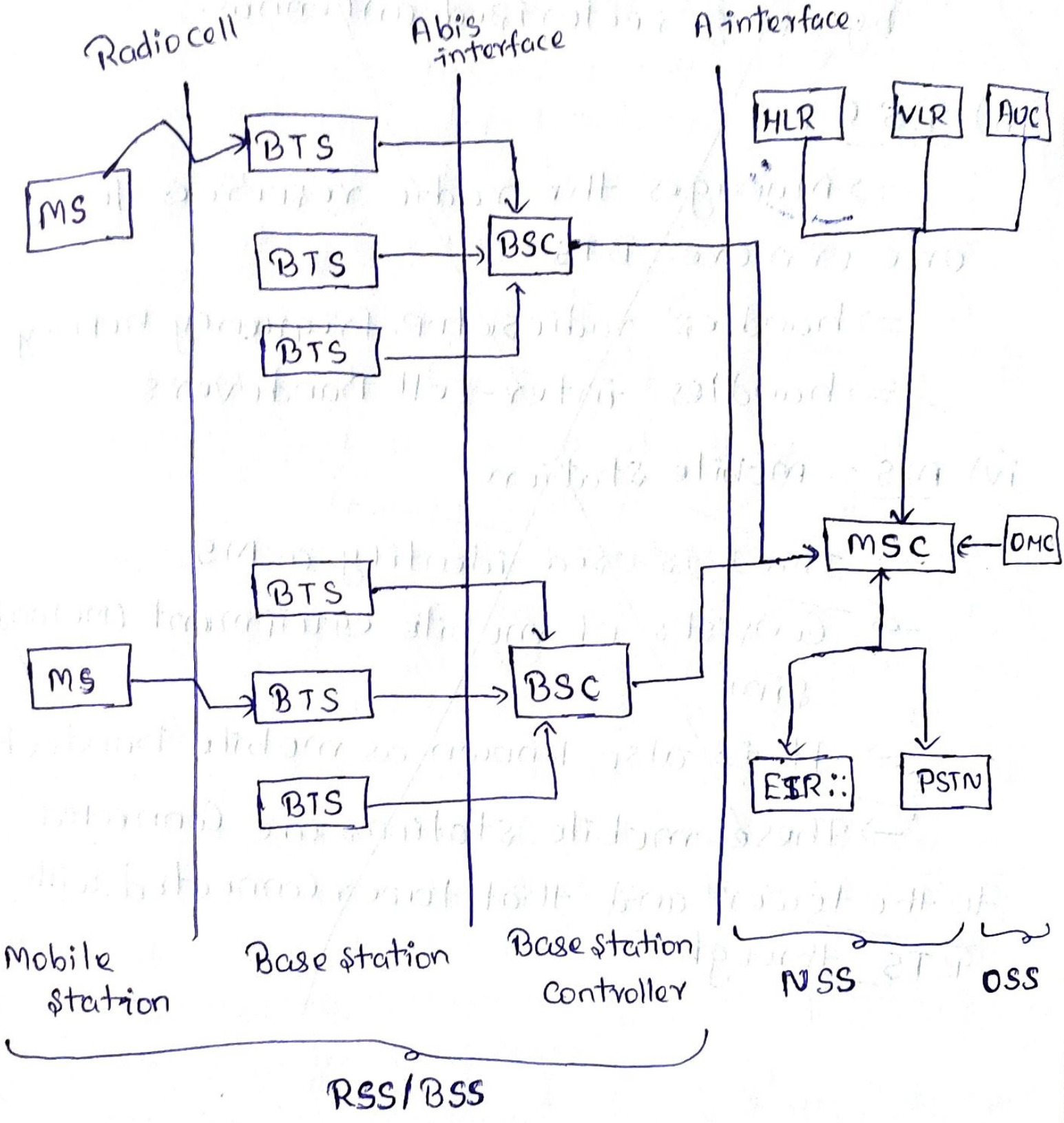
→ ease of accessing the network

→ GSM network can be boardly divided into 3 parts.

=> Radio/Base station subsystem (BSS)

=> Network switching subsystem (NSS)

=> The operation support subsystem (OSS)



1. MS :-

→ MS stands for Mobile station.

→ MS comprises = Mobile Equipment (ME) + Subscriber Identity Module (SIM)

→ Mobile stations are connected to the tower and that tower connected with BTS through Transceiver (TRX)

2. BTS :-

BTS stands for Base transceiver station, which facilitates wireless communication b/w user equipment and a network. Every tower has BTS.

3. BSC :-

Base station Controller. BSC has multiple BTS. You can consider the BSC as a local exchange of your area which has multiple towers and multiple towers have BTS.

4. MSC :-

→ MSC stands for mobile switching center.

→ MSC is associated with communication switching functions such as call setup, call release and routing.

→ Call tracing, Call forwarding all functions are Performed at the MSC level.

→ MSC having components like VLR, HLR, AUC, EIR and PSTN.

* VLR :-

→ VLR stands for Visitor Location Register

→ VLR is a database which contains the exact location of all mobile subscribes currently present in the service area of MSC.

→ HLR :-

⇒ Home Location Register

⇒ HLR is a database containing pertinent data regarding subscribers authorized to use a GSM network. If you purchase SIM card from in the HLR.

⇒ HLR is like a home which contains all data like your ID Proof, which plan you are taking, which caller tune you are using, etc...

→ AUC :-

⇒ AUC stands for Authentication center.

⇒ AUC authenticates the mobile subscriber that wants to connect in the network.

→ EIR :-

⇒ EIR stands Equipment Identity Register.

⇒ EIR is a database that keeps the record of all allowed or banned in the network.

⇒ If you are banned in the network then you can't enter the network, and you can't make the calls.

3. OMC :-

OMC stands for Operation Maintenance center.

→ OMC monitor and maintain the performance of each MS, BSC and MSC within a GSM system.

1. Air Interface :-

→ Air Interface is also known as UM interface

→ Interface b/w MS and BTS

2. Abis interface :-

→ It is a Bss internal interfacing linking with BTS and BSC

3. A interface :-

→ It Provides Communication b/w BSS and MSC

Advantages :-

* Compatibility
* Security
* Roaming
* Wide range of features

Disadvantages :-

* Data transfer speed
* Limited capacity.
* Network Congestion.

Q5. UI Controls / Widgets

Input controls are the interactive components in your app's user interface

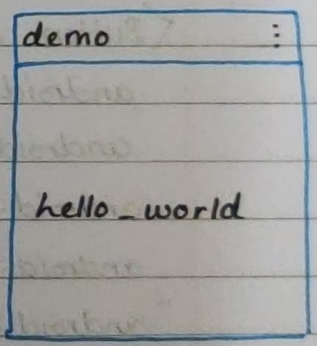
Some examples are as follows →

- TextView Control →

displays text to user and optionally allows them to edit it. It is a complete text editor, however the basic class is configured to not allow editing

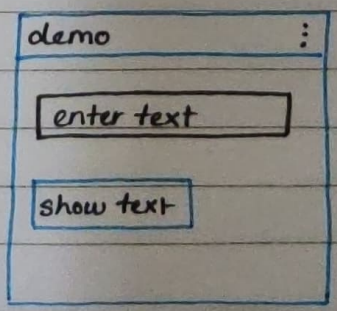
Attributes -

android:id → The id which uniquely identifies the control
android:fontFamily → Font Family for the text
android:inputType → The type of ~~text~~ data being placed in the text field (eg. Date, Time, Phone, Password etc.)
android:text → text to display
android:textColor → color of the text
android:textSize → size of the text



- Edit Text Control →

It is an overlay over TextView that configures itself to be editable



- Button Control →

It is a push button that can be pressed or clicked by the user to perform an action

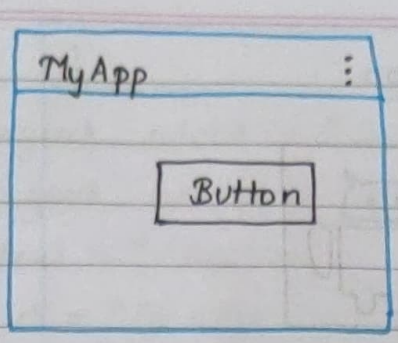
Attributes →

android : text → This is the text displayed
android : editable → This specifies that the TextView has an input method.
android : autoText → specifies that TextView has a text input method and automatically corrects some common spelling errors.
android : drawableBottom → This is the drawable to be drawn below the text
android : drawableRight → This is the drawable to be drawn to the right of the text.

XML Code →

```
<Button android:layout_width = "wrap_content"
  android : layout_height = "wrap_content"
  android : text = "Button" android : id = "@+id/button"
  android: layout_ alignTop = "@+id/ editText"
  android: layout_alignLeft = "@+id/textView1"
  android: layout_alignStart = "@+id/textView1"
  android: layout_alignRight = "@+id/editText"
  android: layout_alignEnd = "@+id/editText" />
```

Output →



- ImageButton →

It shows a button with an image instead of text, that can be pressed or clicked by the user.

Attributes →

android: baseline → this is the offset of the baseline within this view

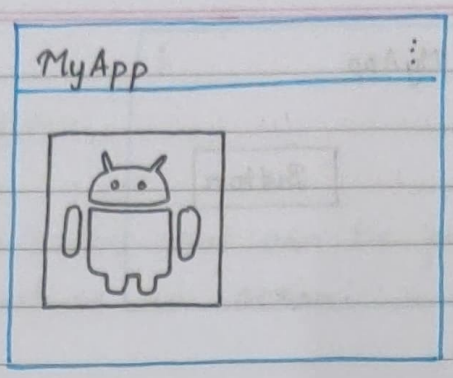
android: cropToPadding → If True, the image will be cropped to fit within its padding

android: src → This sets a drawable as the content of this ImageView.

XML code →

```
<ImageButton
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/imageButton"
  android:layout_centerVertical="true"
  android:layout_centerHorizontal="true"
  android:src="@drawable/abc" />
```

Output →



- ToggleButton Control →

It displays checked / unchecked states as a button
It is basically an on/off button with a light indicator.

Attributes →

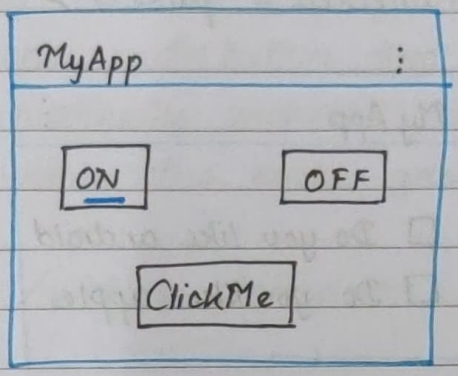
android : textOff → the text for the button when it is not checked
android : textOn → the text for the button when it is checked.

XML code →

```
<ToggleButton
  android : layout_width = "wrap-content"
  android : layout_height = "wrap-content"
  android : text = "On"
  android : id = "@+id/toggleButton1"
  android : checked = "true" />
<ToggleButton
  android : layout_width = "wrap-content"
  android : layout_width = "wrap-content"
  android : text = "Off"
  android : id = "@+id/toggleButton2"
  android : checked = "false" />
```

```
<Button
  android : layout_width = "wrap_content"
  android : layout_height = "wrap_content"
  android : text = "ClickMe"
  android : id = "@+id/button2" />
```

Output →



- **CheckBox Control →**

It is an on/off switch that can be toggled by the user
It is used to present the users with a group of selectable options that are not mutually exclusive. (ie. more than one can be selected)

Attributes →

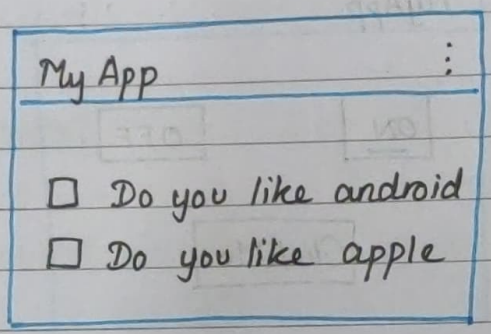
android : text → This is the text to be displayed
android : checked → Specifies if the checkbox is checked or not.

XML code →

```
<CheckBox
  android : layout_width = "wrap_content"
  android : layout_height = "wrap_content"
  android : text = "Do you like android"
  android : id = "@+id/checkBox1"
  android : checked = "false" />
```

```
<CheckBox
    android : layout_width = " wrap_content"
    android : layout_height= " wrap_content"
    android : text = " Do yo like apple"
    android : id = "@+id/checkButton2"
    android : checked = "false"/>
```

Output →



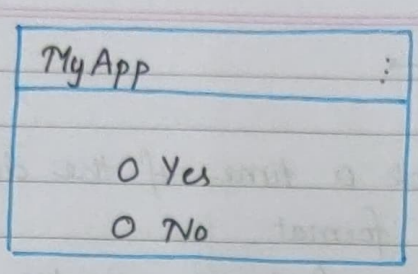
- Radio Button Control →

It allows users to select one option from a set, it has two states, checked / unchecked

XML code →

```
<RadioGroup>
    <RadioButton
        android : text = " Yes"
        android : id = "@+id/radioButton1"
        android : checked = " false" />
    <RadioButton
        android : text = "No"
        android : id = " @+id/radioButton2"
        android : checked = " false "/>
</RadioGroup>
```

Output →

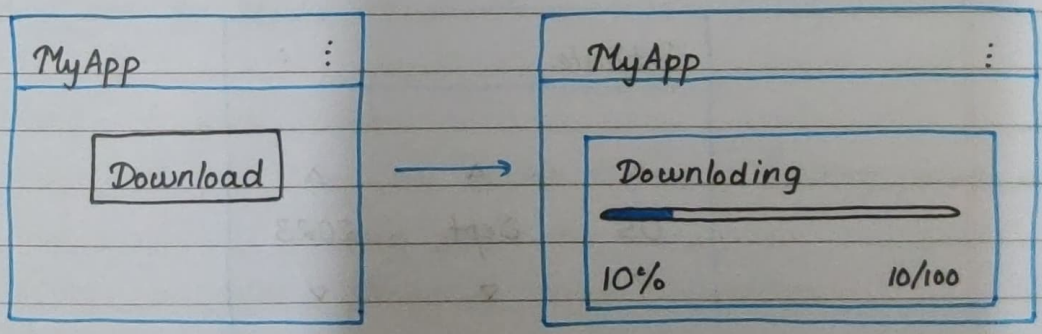


The RadioGroup class is used for a set of radio Buttons. If we check one radio Button that belongs to a radio Group, it automatically unchecks any previously checked radio button within the same group.

Other controls / widgets include →

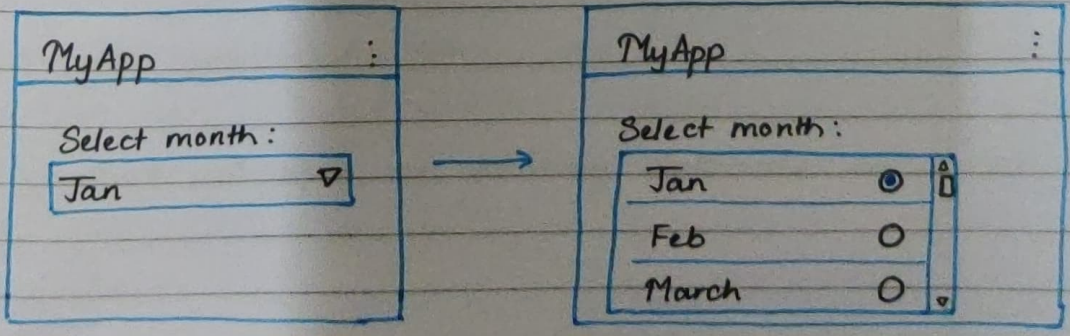
- Progress Bar Control →

Used to show the progress of a task
eg. when you are uploading or downloading something from the internet, it shows the progress of upload or download to the user



- Spinner Control →

Allows you to select from a drop down menu



6. UI Layouts

- The basic building block for user interface is a View object which is created from the View class
- It occupies a rectangular area on the screen and is responsible for drawing and event handling.
- View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

UI Layouts (con…)

- The ViewGroup is a subclass of View and provides invisible container that hold other Views or other ViewGroups and define their layout properties.
- At third level we have different layouts which are subclasses of ViewGroup class
- A typical layout defines the visual structure for an Android user interface.

UI Layouts (con…)

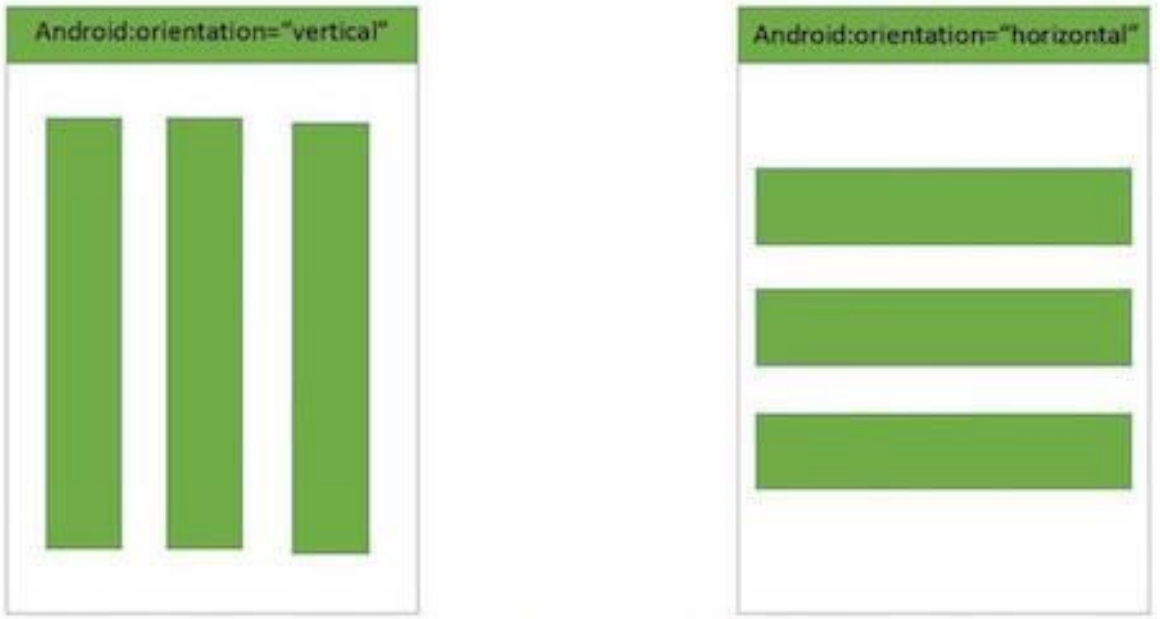
- To declare the layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.
- A layout may contain any type of widgets such as buttons, labels, textboxes, and so on.

Layout Types

- Linear Layout
- Relative Layout
- Table Layout
- Absolute Layout
- Frame Layout
- List View
- Grid View

Linear Layout

- Linear Layout is a view group that aligns all children in either vertically or horizontally.



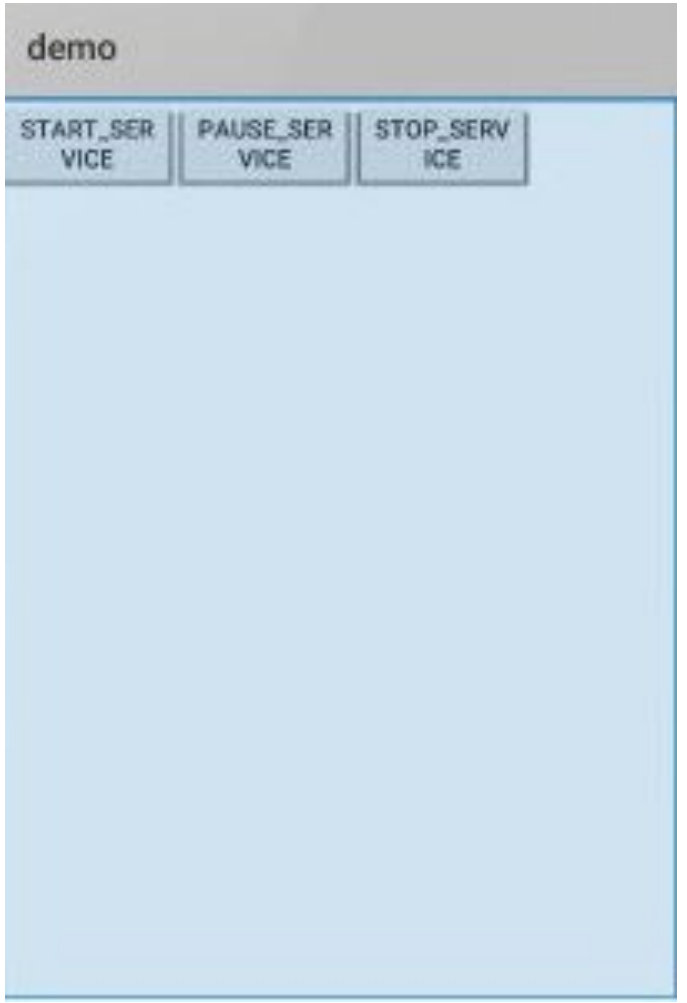
LINEAR LAYOUT

Example

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/
   res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
<!-- More GUI components go here -->
</LinearLayout>
```

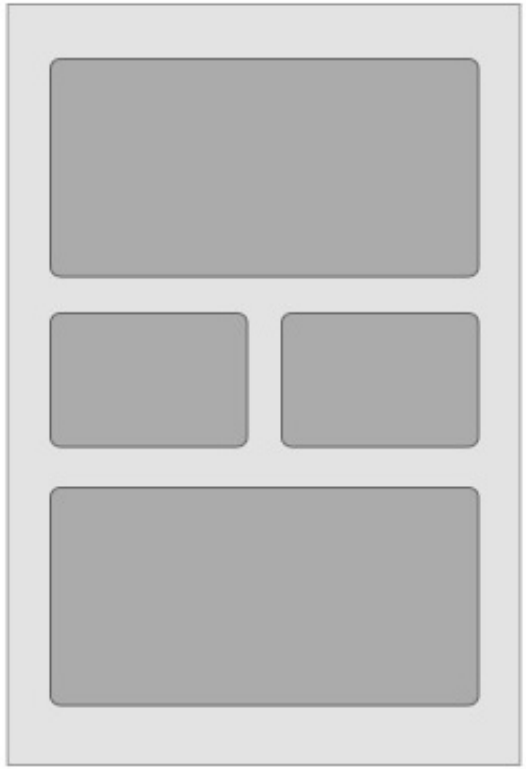
Output


demo
START_SERVICE
PAUSE_SERVICE
STOP_SERVICE


demo
START_SER VICE
PAUSE_SER VICE
STOP_SERV ICE

Relative Layout

- Relative Layout enables you to specify how child views are positioned relative to each other.
- The position of each view can be specified as relative to sibling elements or relative to the parent.



RELATIVE LAYOUT

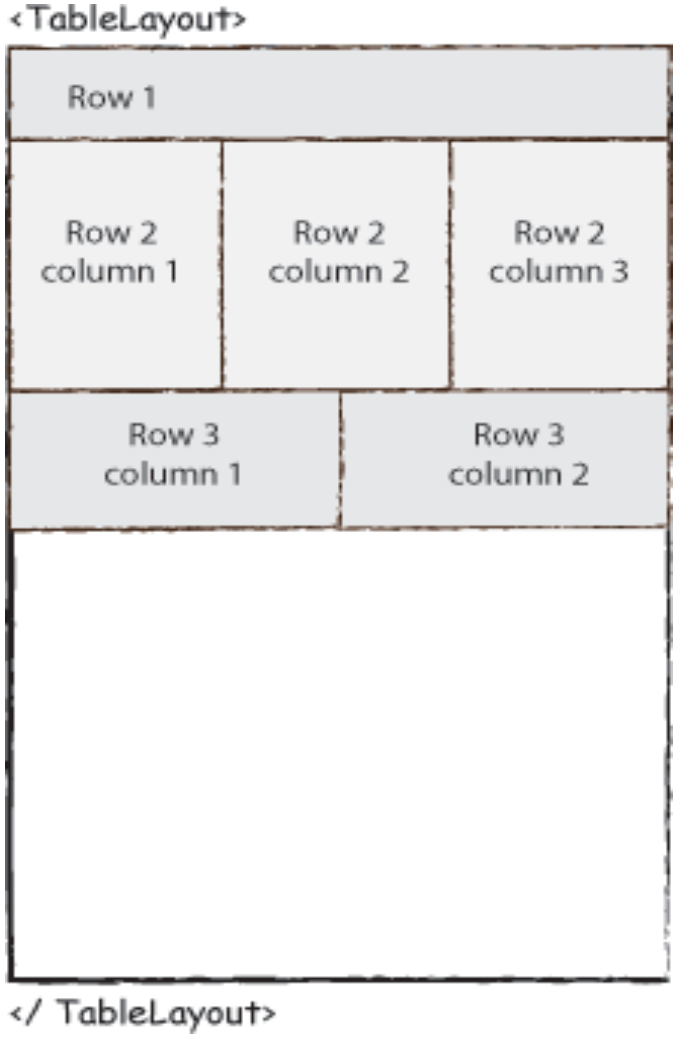
Example

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:paddingLeft="16dp"
   android:paddingRight="16dp" >
<!-- More GUI components go here -->
</RelativeLayout>
```

Output

Table Layout

- TableLayout going to be arranged groups of views into rows and columns.
- Use the <TableRow> element to build a row in the table.
- Each row has zero or more cells; each cell can hold one View object
- It don't display border lines for their rows, columns, or cells.



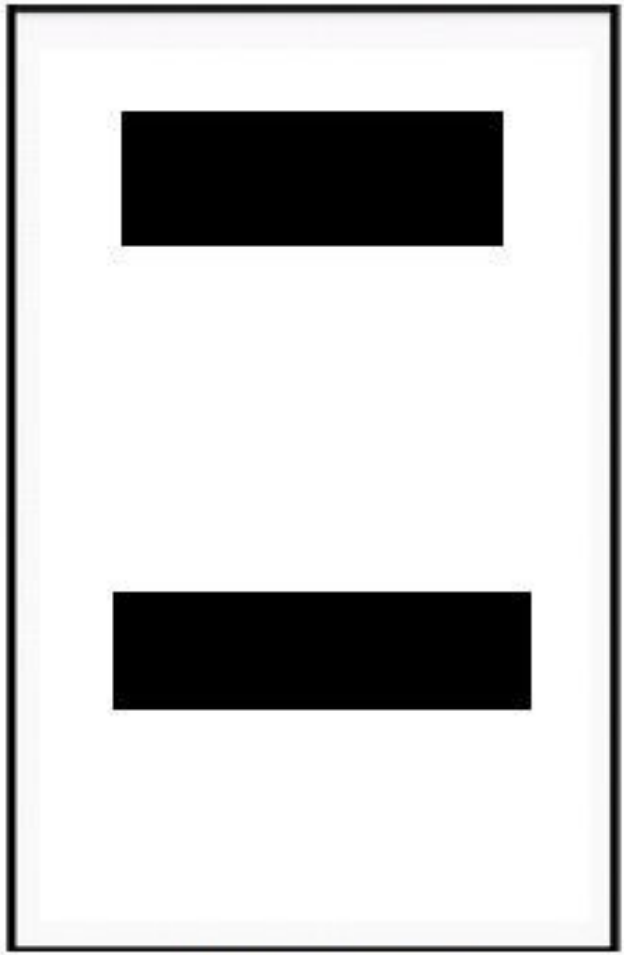
Example

```
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/and
   roid"
  android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <TableRow
         android:layout_width="fill_parent"
         android:layout_height="fill_parent">
         <!-- More GUI components go here -->
   </TableRow>
   <!-- More Table rows go here -->
</TableLayout>
```

Output

Absolute Layout

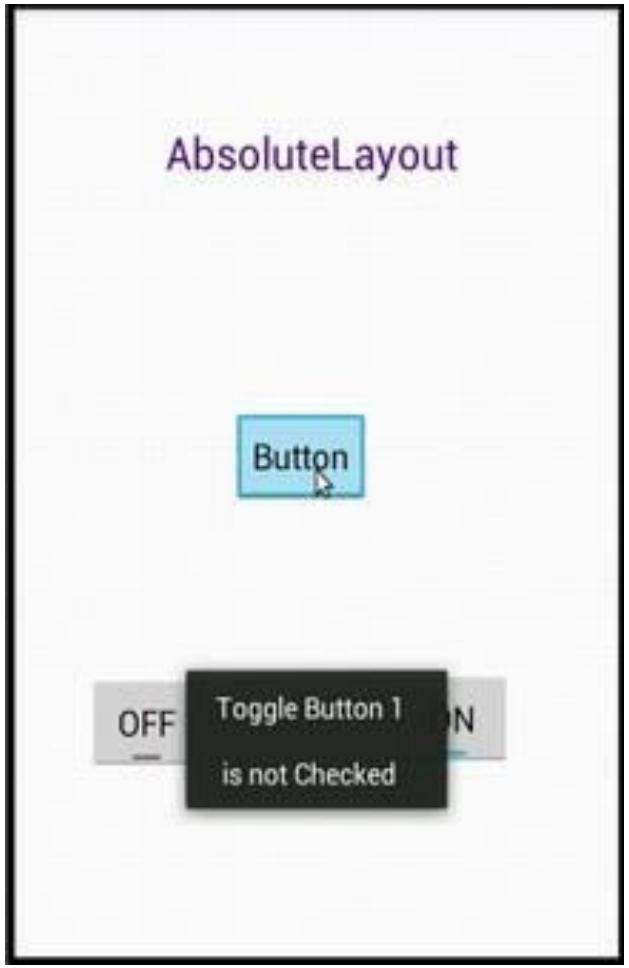
- Absolute Layout lets you specify exact locations (x/y coordinates) of its children.
- Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.



Example

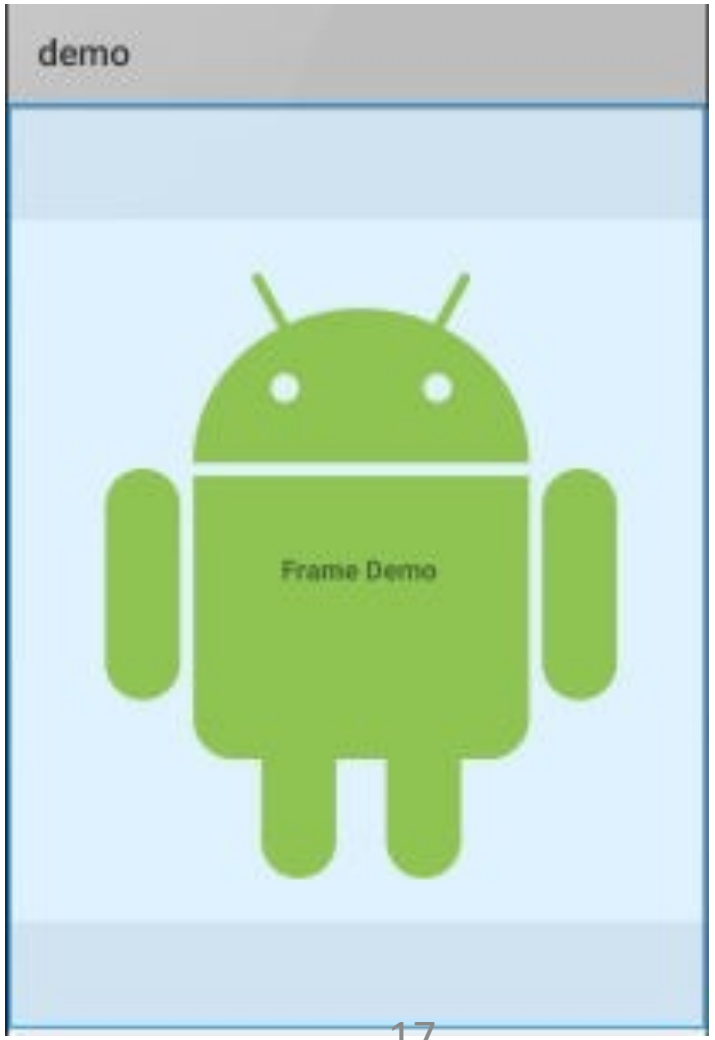
```
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text="OK"
            android:layout_x="50px"
            android:layout_y="361px" />
<!-- More GUI components go here -->
</AbsoluteLayout>
```

Output



Frame Layout

- Frame Layout is designed to block out an area on the screen to display a single item.
- Generally, Frame Layout should be used to hold a single child view.
- We can add multiple children to a FrameLayout and control their position by assigning gravity to each child, using the android:layout_gravity a ttribute.



Example

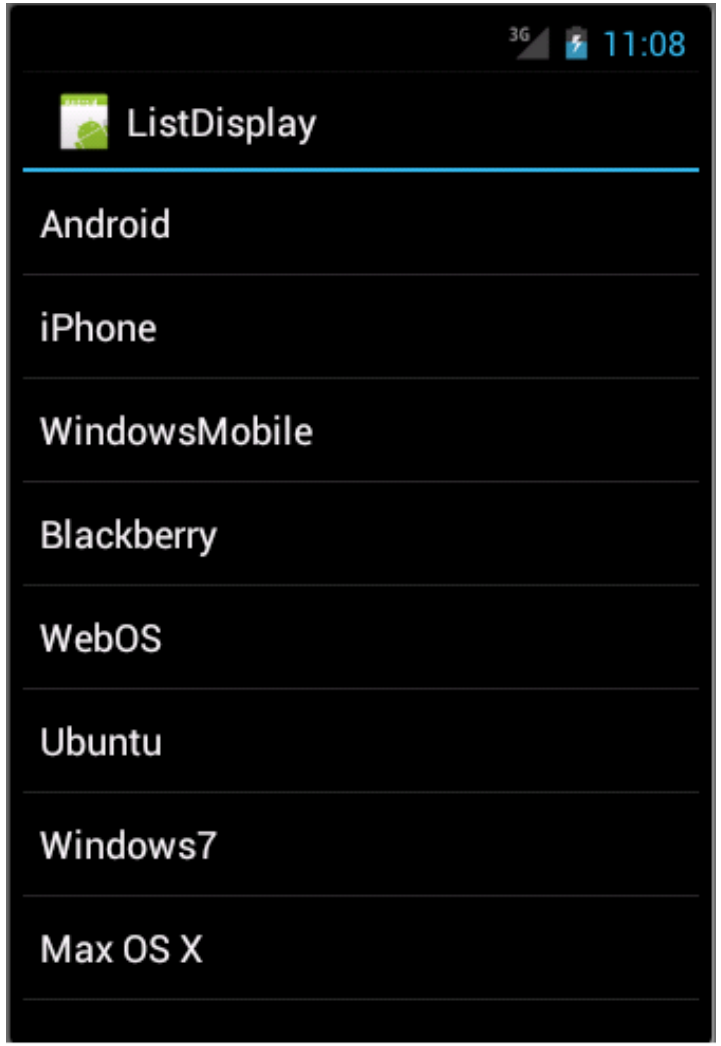
```
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <!-- More GUI components go here -->
</FrameLayout>
```

Output

FRAME LAYOUT

List View

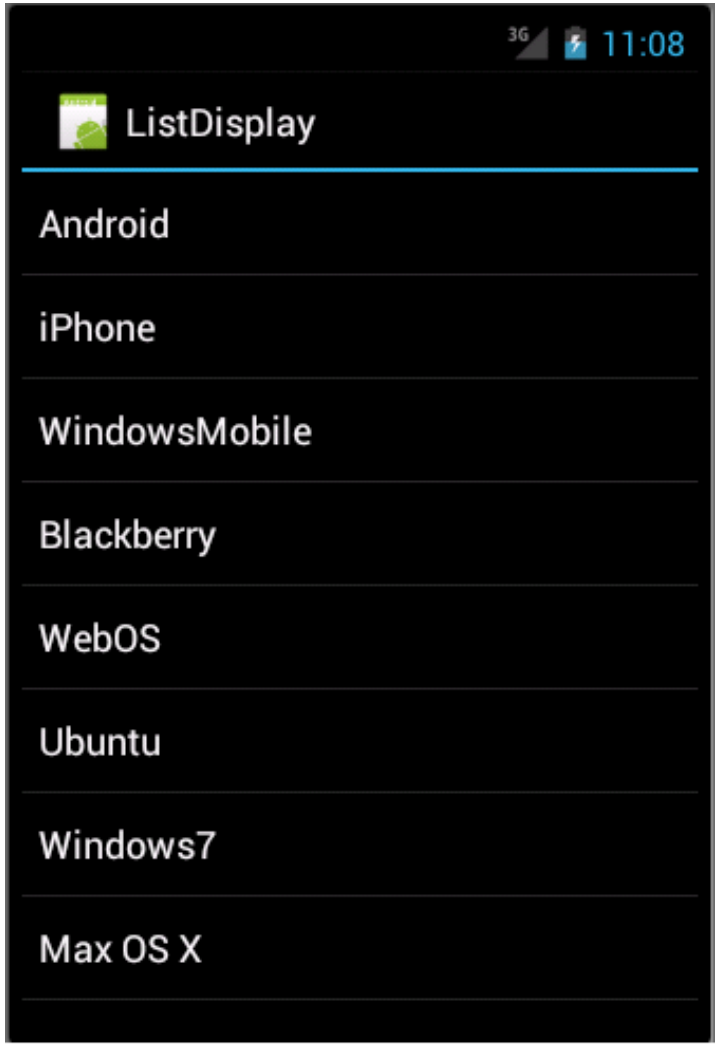
- List View is a view which groups several items and display them in vertical scrollable list.
- The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.



Example

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/and
    roid" xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ListActivity" >
        <ListView android:id="@+id/mobile_list"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" >
        </ListView>
</LinearLayout>
```

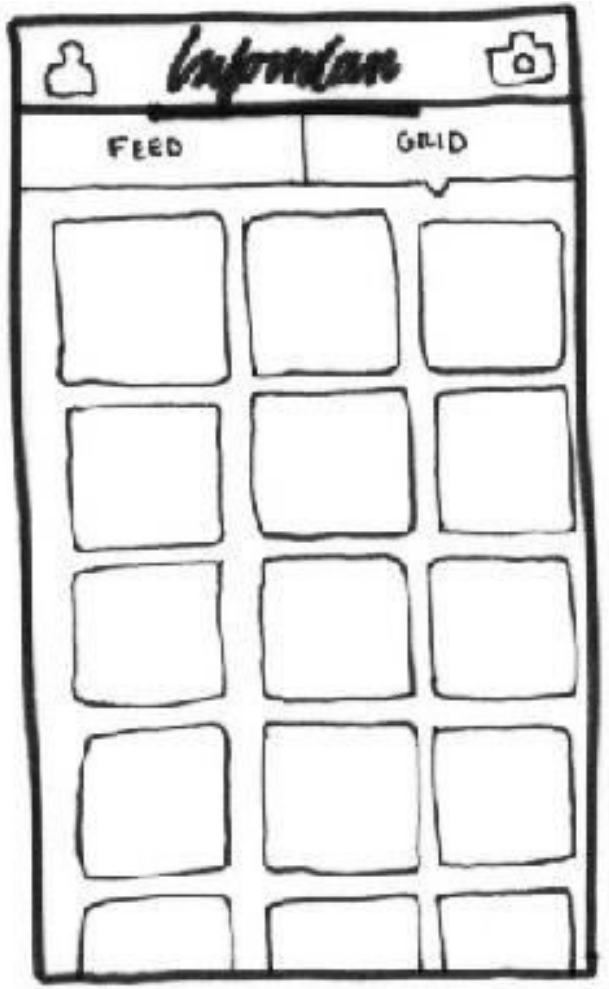
Output



LIST VIEW

Grid View

- Grid View shows items in two-dimensional scrolling grid (rows & columns)
- The grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter



GRID VIEW

Example

```
<?xml version="1.0" encoding="utf-8"?>
<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```

Output

Button Control

- A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.