

## Unit 2

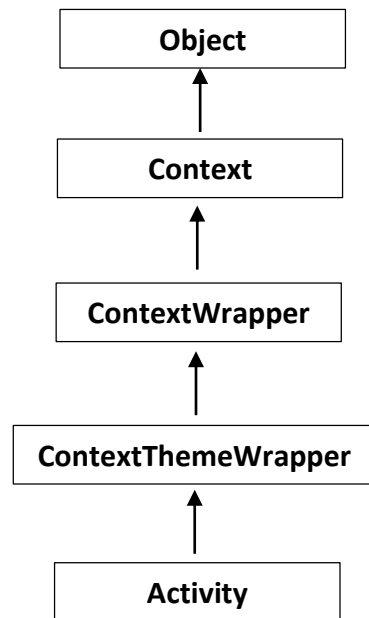
### User Interface Design

#### Activity:

An activity is each and every single screen in android. It is same as Frame or Window in Java.

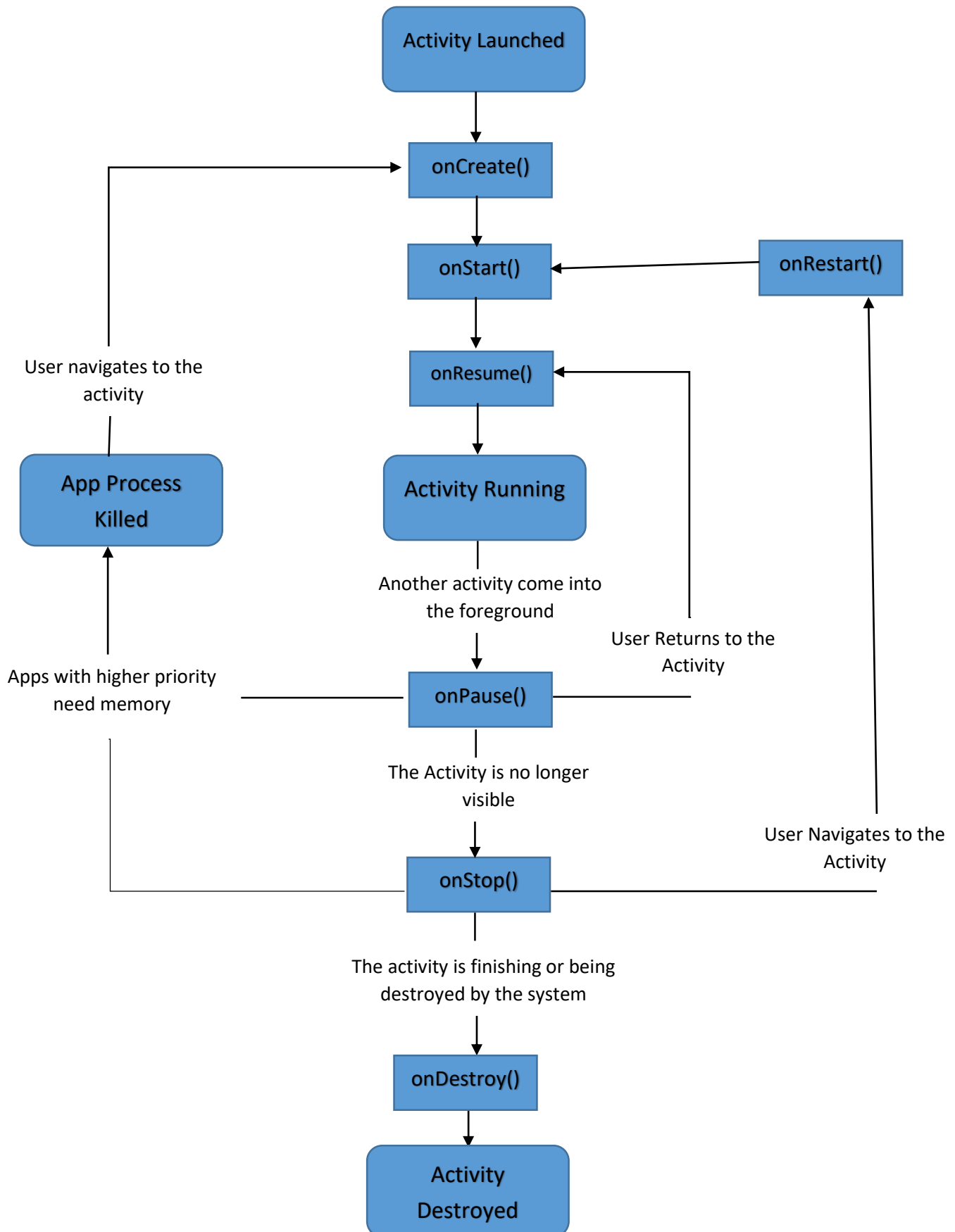
Using activity, we can insert elements and place UI elements in a single screen.

Android Activity is the subclass of ContextThemeWrapper class. Below is the complete class hierarchy for Activity.



Android Activity Life Cycle controlled by total of 7 methods of android.app.Activity class.

## Activity Life Cycle



The following callbacks, or events, are defined by the Activity class. It's not required that you use every callback method. But it's crucial that you comprehend each one and put those into practice to make sure your app operates as consumers would anticipate.

Sr. No.	Callback & Description
1	<b>onCreate()</b> This is the first callback and called when the activity is first created.
2	<b>onStart()</b> This callback is called when the activity becomes visible to the user.
3	<b>onResume()</b> This is called when the user starts interacting with the application.
4	<b>onPause()</b> The paused activity does not receive user input and cannot execute any code and called when the c activity is being resumed.
5	<b>onStop()</b> This callback is called when the activity is no longer visible.
6	<b>onDestroy()</b> This callback is called before the activity is destroyed by the system.
7	<b>onRestart()</b> This callback is called when the activity restarts after stopping it.

### Android Activity Lifecycle Example

File: MainActivity.java

```
package example.javatpoint.com.activitylifecycle;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.util.Log;
```

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

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        Log.d("lifecycle", "onCreate invoked");
```

```
    }
```

```
    @Override
```

```
    protected void onStart() {
```

```

    super.onStart();
    Log.d("lifecycle","onStart invoked");
}
@Override
protected void onResume() {
    super.onResume();
    Log.d("lifecycle","onResume invoked");
}
@Override
protected void onPause() {
    super.onPause();
    Log.d("lifecycle","onPause invoked");
}
@Override
protected void onStop() {
    super.onStop();
    Log.d("lifecycle","onStop invoked");
}
@Override
protected void onRestart() {
    super.onRestart();
    Log.d("lifecycle","onRestart invoked");
}
@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d("lifecycle","onDestroy invoked");
}
}

```

### Android Fragments:

Fragments are the part of activity or we can say it's a sub-activity. This helps to represent multiple screens in any activity.

Fragment lifecycle gets affected by activity lifecycle as fragments are embedded in activity only.

Just like Activity Manager, Fragment Manager is responsible for making interactions between fragment objects.

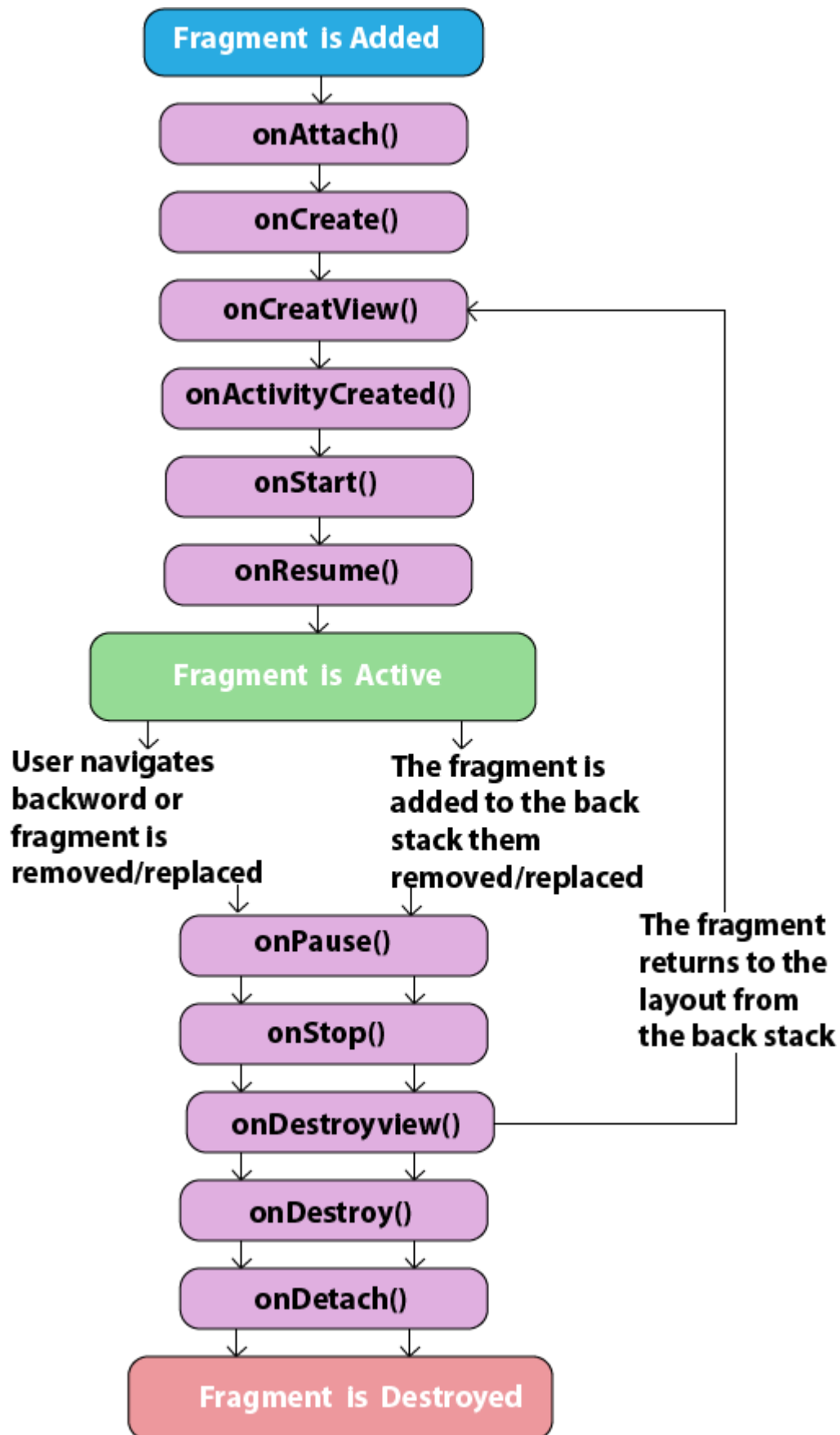
### Types of Fragments:

1. **Single Fragment:** Only one view displays on screen.
2. **List Fragment:** Fragments display in a list view from which user can select their desired sub-activity or fragments. We can consider Gmail as an example for list fragment display.
3. **Fragment Transaction:** User can make transition among multiple fragments at run time. User can switch between fragments by switching on tabs.

### Fragment LifeCycle:

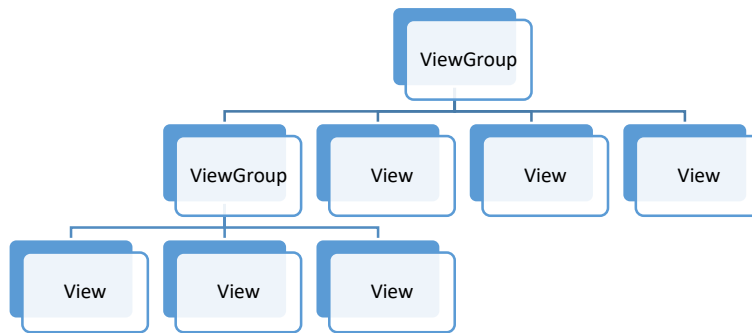
The following callbacks, or events, are defined by the Fragment class. It's not required that you use every callback method. But it's crucial that you comprehend each one and put those into practice to make sure your app operates as consumers would anticipate.

Sr. No.	Callback & Description
1	<b>onAttach()</b> Called when fragment is attached with activity.
2	<b>onCreate()</b> It is used to initialize the fragment.
3	<b>onCreateView()</b> Creates and returns view hierarchy.
4	<b>onActivityCreated(Bundle)</b> It is invoked after the completion of onCreate() method.
5	<b>onViewStateRestored(Bundle)</b> It provides information to the fragment that all the saved state of fragment view hierarchy has been restored.
6	<b>onStart()</b> Makes the fragment visible.
7	<b>onResume()</b> Makes the fragment interactive.
8	<b>onPause()</b> It is called when fragment is no longer interactive.
9	<b>onStop()</b> It is called when fragment is no longer visible.
10	<b>onDestroyView()</b> It allows the fragment to clean up resources.
11	<b>onDestroy()</b> It allows the fragment to do final clean-up of fragment state.
12	<b>onDetach()</b> It is called immediately prior to the fragment no longer being associated with its activity.



## Android Layouts:

Layout is a view or interface which we design for our app, such as for an Activity. The elements in the layout built using a hierarchy of View and ViewGroup. View is a place which user sees and using which user interacts with application. ViewGroup is an invisible container which holds our view and other view groups, as shown in below Fig:



## Types of Layouts

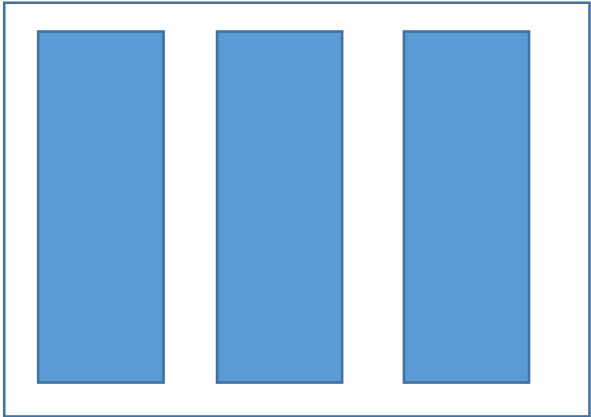
S.No.	Layout & Description
1.	<b>Linear Layout</b> This layout aligns all the elements in a line, i.e, Horizontal or Vertical
2.	<b>Relative Layout</b> This layout shows all the elements at its relative position.
3.	<b>Table Layout</b> This layout group views in rows and columns.
4.	<b>Absolute Layout</b> This layout allows developer to give the exact location of elements.
5.	<b>Frame Layout</b> This layout works as a container for our view/ elements.
6.	<b>List View</b> This layout displays elements in a list form and make it as scrollable items.
7.	<b>Grid View</b> This layout displays elements in a form of grid and make it as scrollable elements.

1. **Linear Layout:** Linear Layout is a view group which aligns all the elements in a single direction, horizontally or vertically. You can specify direction using **orientation** attribute (android:orientation).

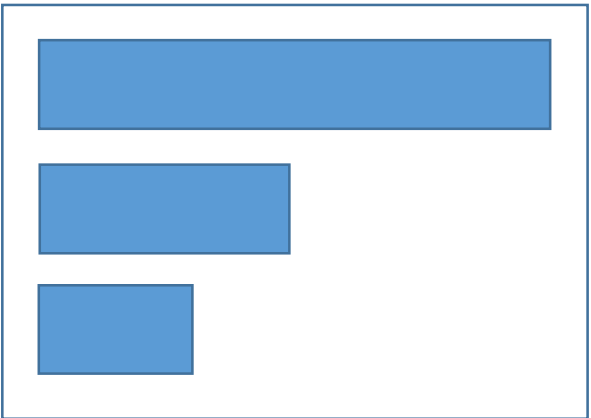
**<LinearLayout>**

**</LinearLayout>**

Horizontal Linear View



Vertical Linear View



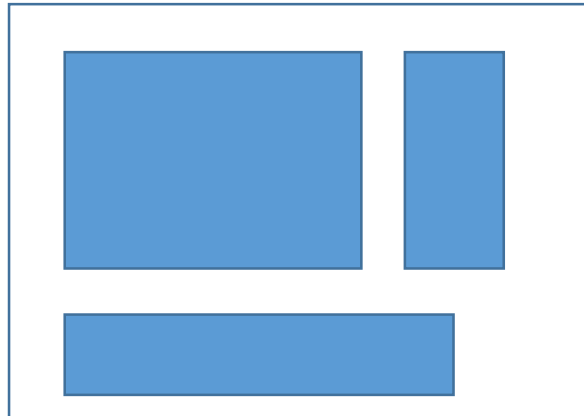


2. **Relative Layout:** Relative Layout is a layout that holds the elements at their relative position. A RelativeLayout is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance. If you find yourself using several nested LinearLayout groups, you may be able to replace them with a single RelativeLayout.

`<LinearLayout>`

`</LinearLayout>`

Relative Layout



3. **Table Layout:** Table Layout displays elements in rows and columns. It displays all children in rows and columns with no borders between rows and columns or for each cell. For defining rows we use “`<TableRow>` `</TableRow>`” tag.

`<TableLayout>`

`<TableRow>`

`<EditText>``</EditText>`

`<EditText>``</EditText>`

`</TableRow>`

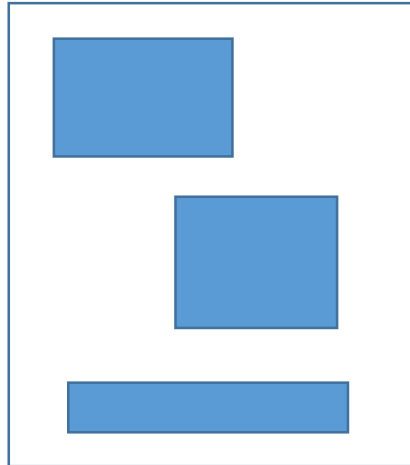
`</TableLayout>`

Table Layout



4. **Absolute Layout:** Absolute Layout allows developer to specify exact location, i.e., X and Y coordinates for its elements with the respect to the top-left corner at the top of the layout. This layout is flexible and harder to maintain in different sizes of screen, that's why it is highly not recommended.

**Absolute Layout**

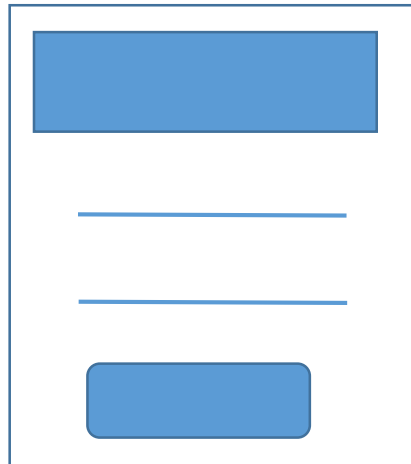


5. **Frame Layout:** A ViewGroup subclass called Android Framelayout is used to define the arrangement of several views stacked on top of one another to create the appearance of a single view screen. In general, FrameLayout is only a way to display a single view by blocking a specific section of the screen.

**<FrameLayout>**

**</FrameLayout>**

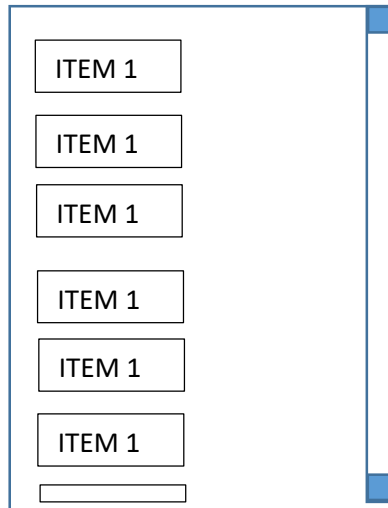
**Frame Layout**



6. **List View:** List view displays elements in the form of list with scrollable items. As subclasses of AdapterView, ListView and GridView can be filled with data by attaching them to an Adapter, which builds a View for each data entry and pulls data from an external source.

**Example:**

```
ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,StringArray);  
ListView listView = (ListView) findViewById(R.id.listview);  
listView.setAdapter(adapter);
```

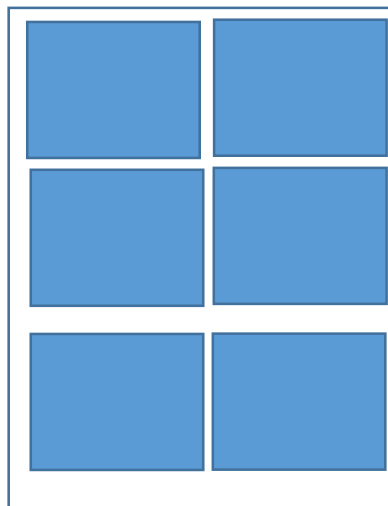


7. **Grid View:** Grid view displays elements in form of grid with scrollable items. One kind of AdapterView that shows things in a scrolling, two-dimensional grid is called a GridView. This grid layout inserts items from either an array or a database. This data is displayed using the adapter, and GridView is joined with the adapter using the setAdapter() method. The major job of the GridView adapter is to retrieve data from an array or database and add it to the relevant item that will be shown in GridView. The GridView structure looks like this. Both Java and Kotlin programming languages will be used to develop this project for Android.

**<GridView>**

**</GridView>**

**Grid View**



### Layout Attributes:

S.No.	Attribute and It's Description
1	<b>android:id</b> This is the ID which uniquely identifies the view.
2	<b>android:layout_width</b> This is the width of the layout.
3	<b>android:layout_height</b> This is the height of the layout
4	<b>android:layout_marginTop</b> This is the extra space on the top side of the layout.
5	<b>android:layout_marginBottom</b> This is the extra space on the bottom side of the layout.
6	<b>android:layout_marginLeft</b> This is the extra space on the left side of the layout.
7	<b>android:layout_marginRight</b> This is the extra space on the right side of the layout.
8	<b>android:layout_x</b> This specifies the x-coordinate of the layout.
9	<b>android:layout_y</b> This specifies the y-coordinate of the layout.
10	<b>android:layout_width</b> This is the width of the layout.
11	<b>android:paddingLeft</b> This is the left padding filled for the layout.
12	<b>android:paddingRight</b> This is the right padding filled for the layout.
13	<b>android:paddingTop</b> This is the top padding filled for the layout.
14	<b>android:paddingBottom</b> This is the bottom padding filled for the layout.

### UI (User Interface) Elements:

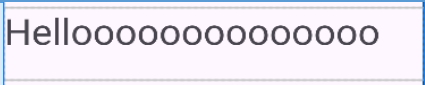

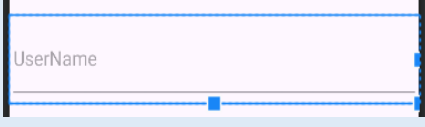
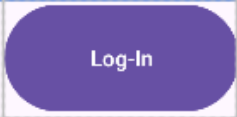
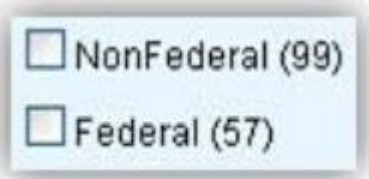

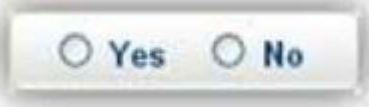

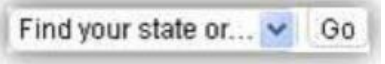
A **View** is a place which draws something on screen using which user can interact with and **ViewGroup** is a container that holds views to define the layout of user interface.

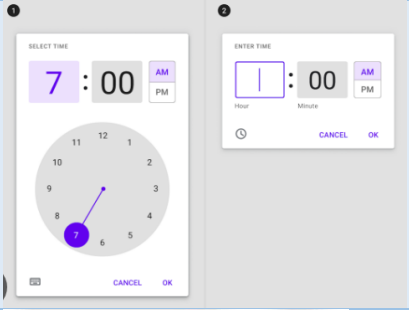

When creating your interface, make an effort to select interface pieces that are predictable and consistent. Users have grown accustomed to particular features functioning in a certain way, whether they realize it or not. Choosing to employ such elements when it makes sense will contribute to task completion, efficiency, and satisfaction.

Interface Elements includes:

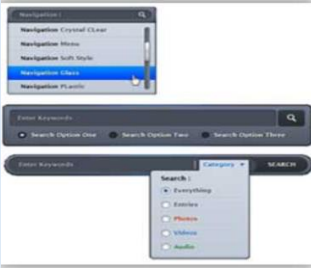


1. **Input Controls:** checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date field

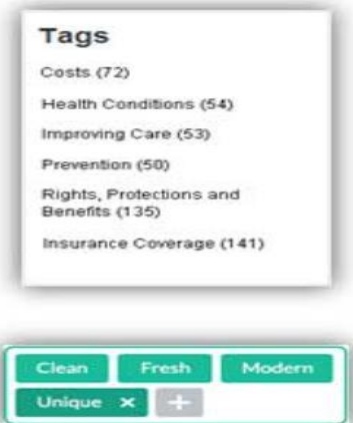



### Input Controls:

S.No	Input Control & It's Description	Examples
1	<b>TextView</b> This control is used to display text to the user.	
2	<b>EditText</b> EditText is a predefined subclass of TextView that includes rich editing capabilities.	
3	<b>AutoCompleteTextView</b> The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.	
4	<b>Button</b> A push-button that can be pressed, or clicked, by the user to perform an action.	
5	<b>ImageButton</b> An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.	
6	<b>CheckBox</b> An on/off switch that can be toggled by the user. You should use check box when presenting users with a group of selectable options that are not mutually exclusive.	
7	<b>ToggleButton</b> An on/off button with a light indicator.	
8	<b>RadioButton</b> The RadioButton has two states: either checked or unchecked.	
9	<b>RadioGroup</b> A RadioGroup is used to group together one or more RadioButtons.	
10	<b>Spinner/DropDown</b> A drop-down list that allows users to select one value from a set.	

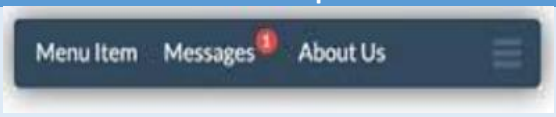
11	<b>TimePicker</b> The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.	
12	<b>DatePicker</b> The DatePicker view enables users to select a date of the day.	


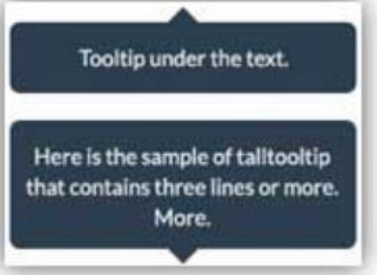
## 2. Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons.

S. No.	Input Control & It's Description	Examples
1	<b>Search Field</b> A search box allows users to enter a keyword or phrase (query) and submit it to search the index with the intention of getting back the most relevant results. Typically search fields are single-line text boxes and are often accompanied by a search button.	
2	<b>Breadcrumb</b> Breadcrumbs allow users to identify their current location within the system by providing a clickable trail of proceeding pages to navigate by.	
3	<b>Pagination</b> Pagination divides content up between pages, and allows users to skip between pages or go in order through the content.	

4	<b>Tags</b> Tags allow users to find content in the same category. Some tagging systems also allow users to apply their own tags to content by entering them into the system.	 <p>The top example shows a 'Tags' section with a list of categories and their counts: Costs (72), Health Conditions (54), Improving Care (53), Prevention (50), Rights, Protections and Benefits (135), and Insurance Coverage (141). The bottom example shows a set of filter buttons: 'Clean', 'Fresh', 'Modern', 'Unique' (with an 'x' to remove), and a '+' button to add more.</p>
5	<b>Sliders</b> A slider, also known as a track bar, allows users to set or adjust a value. When the user changes the value, it does not change the format of the interface or other info on the screen.	 <p>The example shows three sliders. The top one has a percentage scale from 0% to 100% with a handle at 45%. The middle one has a percentage scale from 0% to 100% with a handle at 20%. The bottom one has a numerical scale from 0 to 500 with a handle at 100.</p>
6	<b>Icons</b> An icon is a simplified image serving as an intuitive symbol that is used to help users to navigate the system. Typically, icons are hyperlinked.	 <p>The example shows a grid of 12 icons representing different system functions: a calendar, a document, a camera, a folder, a file, a person, a group of people, a clock, a gear, a list, a grid, and a window.</p>
7	<b>Image Carousel</b> Image carousels allow users to browse through a set of items and make a selection of one if they so choose. Typically, the images are hyperlinked.	 <p>The example shows a carousel with five placeholder images. It has left and right navigation arrows and a progress indicator at the bottom consisting of five dots, with the first dot being filled.</p>

3. **Information Components:** tooltips, icons, progress bar, notifications, message boxes, modal windows

S. No.	Elements & It's Description	Examples
1	<b>Notifications</b> A notification is an update message that announces something new for the user to see. Notifications are typically used to indicate items such as, the successful completion of a task, or an error or warning message.	 <p>The example shows a dark blue notification bar with three items: 'Menu Item', 'Messages' (with a red dot indicating a new message), and 'About Us'. There is a hamburger menu icon on the right.</p>

<p><b>2</b></p>	<p><b>Progress Bars</b></p> <p>A progress bar indicates where a user is as they advance through a series of steps in a process. Typically, progress bars are not clickable.</p>	
<p><b>3</b></p>	<p><b>Tool Tips</b></p> <p>A tooltip allows a user to see hints when they hover over an item indicating the name or purpose of the item.</p>	
<p><b>4</b></p>	<p><b>Message Boxes</b></p> <p>A message box is a small window that provides information to users and requires them to take an action before they can move forward.</p>	