UNIT 4

Mobile Application Development

MSC CS

GAC-CBE

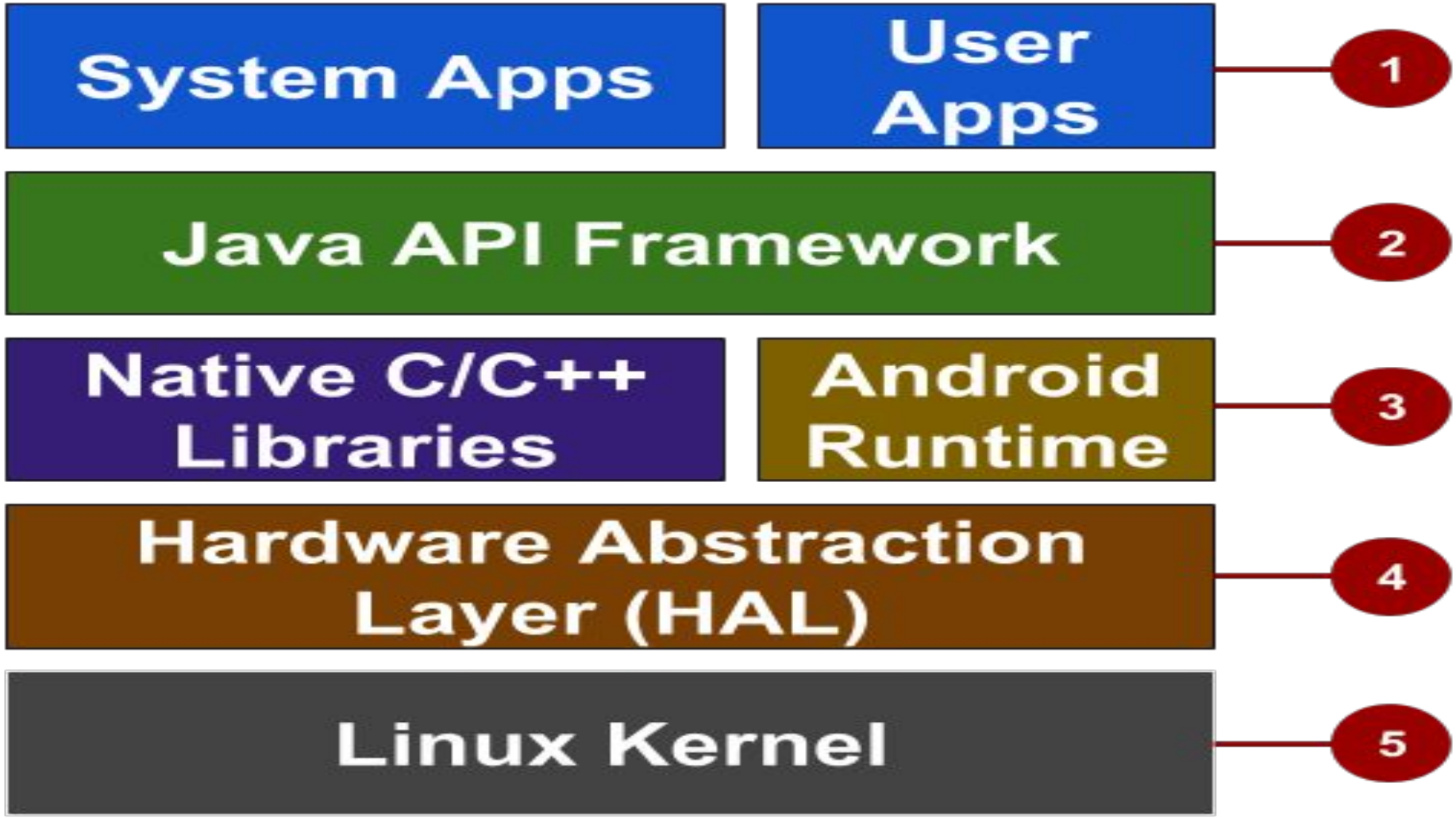
Contents

Android Introduction

- An operating system and programming platform for mobile phones and other mobile devices, such as tablets.
- Can run on many different devices from many different manufacturers.
- Includes a software development kit (SDK) to write original code and assemble software modules to create apps for Android
- Provides a marketplace to distribute apps.
- Represents an *ecosystem* for mobile apps.

Android Architecture

Android is an open source, Linux-based software stack created for a wide array of devices and form factors


System Apps
User Apps
1
Java API Framework
2
Native C/C++ Libraries
Android Runtime
3
Hardware Abstraction Layer (HAL)
4
Linux Kernel
5

Android Architecture

APPS

User apps live at this level, along with core system apps for email, SMS messaging, calendars, internet browsing, and contacts.

Java API framework

All features for Android development, such as UI components, resource management, and lifecycle management, are available through application programming interfaces (APIs).

Android Architecture

Libraries and Android Runtime

Each app runs in its own process, with its own instance of the Android runtime. Android includes a set of core runtime libraries that provide most of the functionality of the Java programming language. Many core Android system components and services are built from native code that require native libraries written in C and C++ which are available to apps through the Java API framework

Hardware Abstraction Layer

Provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.

Android Architecture

5. Linux Kernel

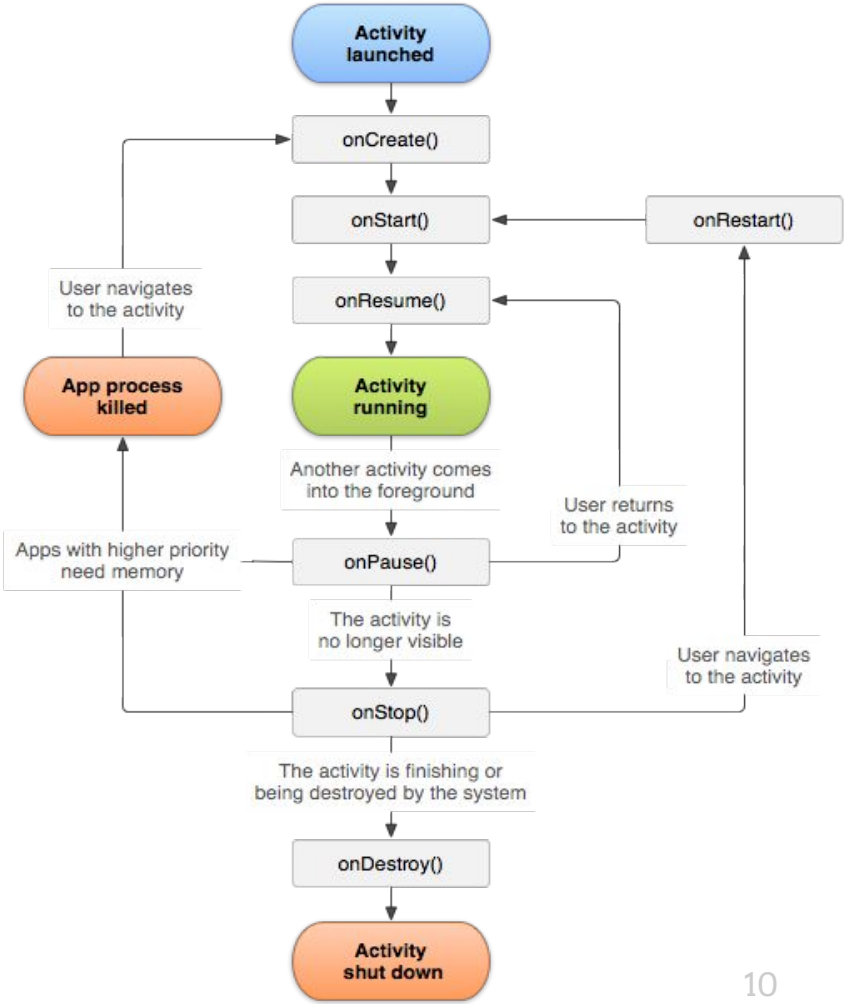
The foundation of the Android platform. The layers above the Linux kernel rely on the Linux kernel for threading, low-level memory management, and other underlying functionality. Enables Android to take advantage of Linux-based security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

Application Life Cycle

Intent facilitates navigation from one activity to another

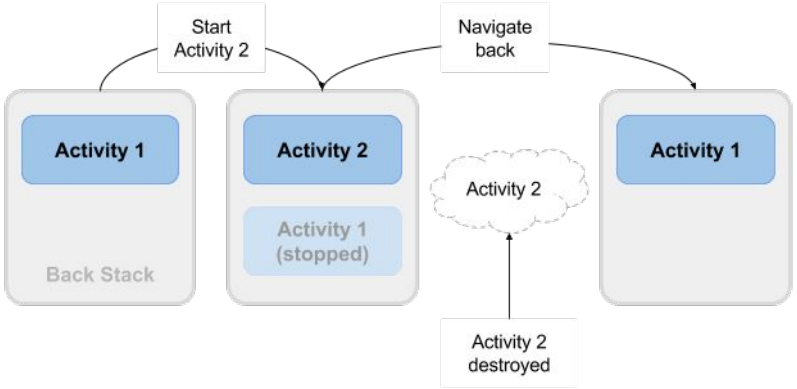
Activity Lifecycle

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as an activity enters a new state.



Application LifeCycle

- When user start an app, the app's main activity ("Activity 1") is started, comes to the foreground, and receives the user focus.
- When user start a second activity ("Activity 2"), a new activity is created and started, and the main activity is stopped.
- When use done with the Activity 2 and navigate back, Activity 1 resumes. Activity 2 stops and is no longer needed.
- If the user doesn't resume Activity 2, the system eventually destroys it

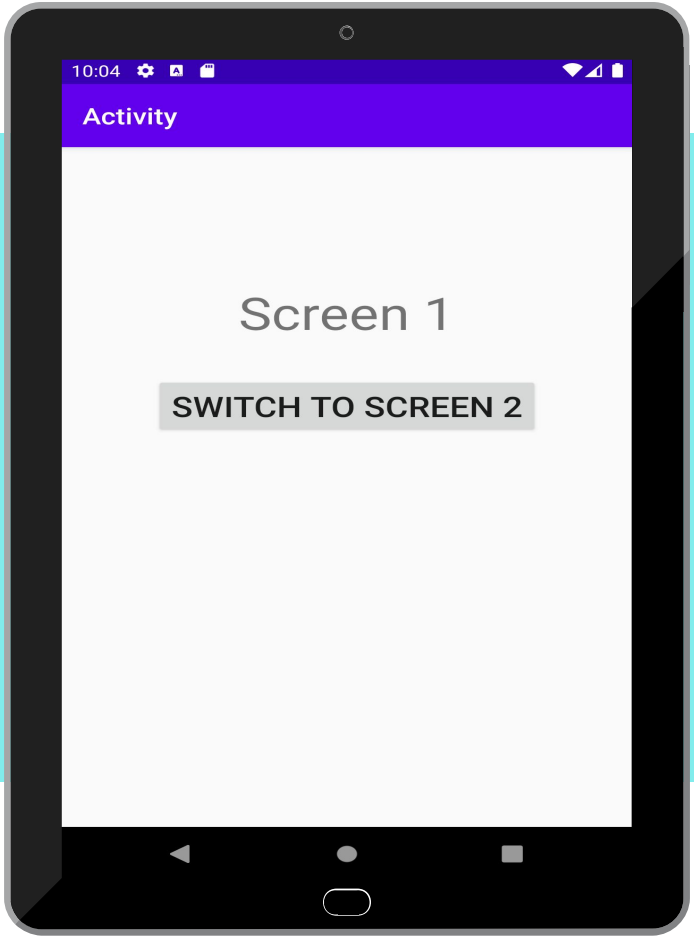


Intent

Intent facilitates navigation from one activity to another

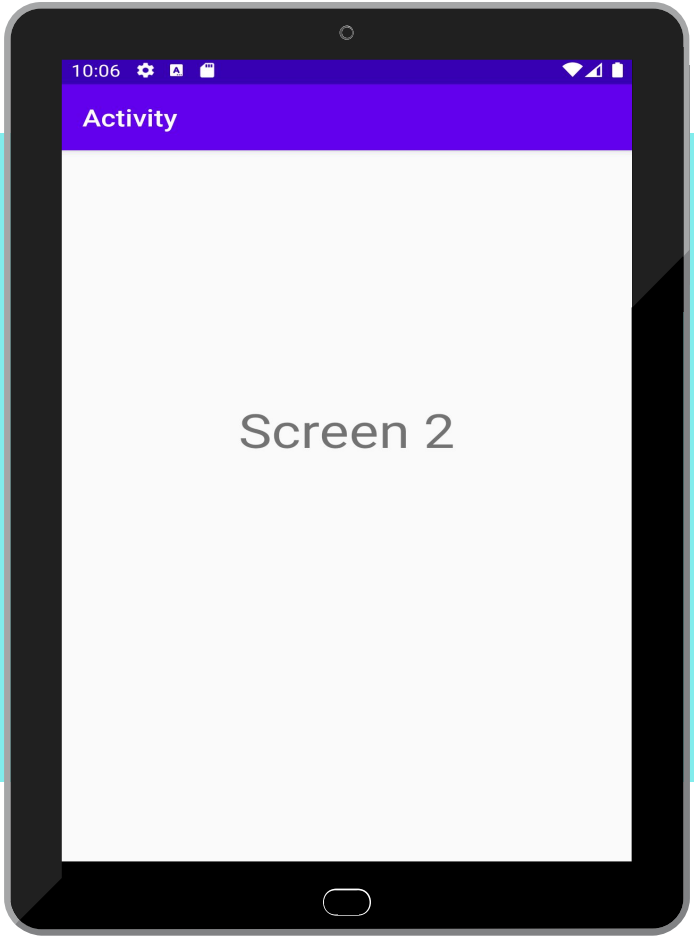
Explicit Intent

Present in the same application helps navigating from one activity to another. Eg Throwing a ball to particular person and asking him to catch it.



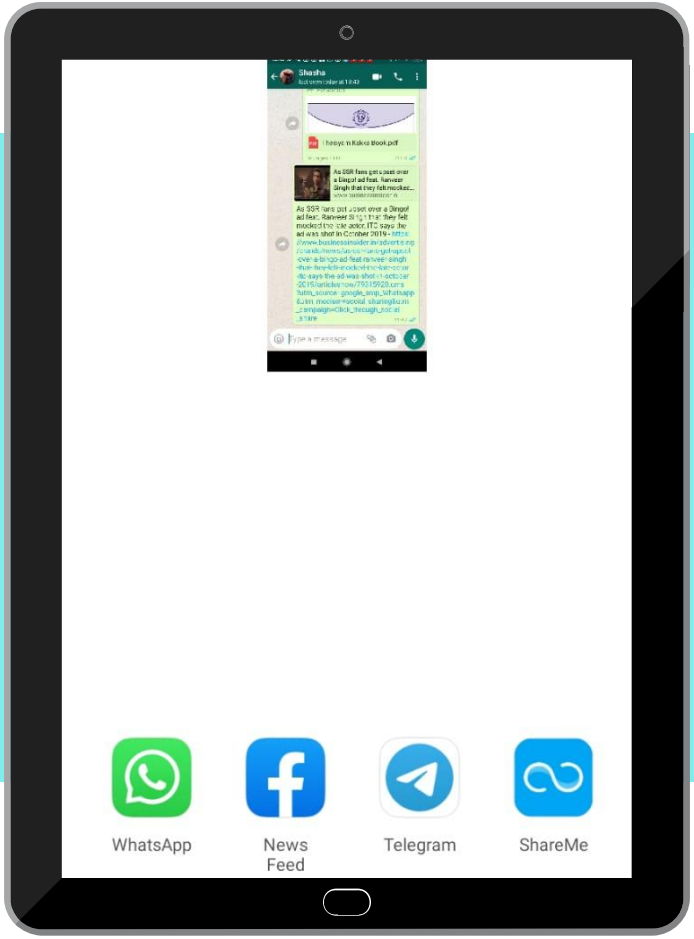
MainActivity

In Main Activity a text Screen1 is added. A Button SWITCH TO SCREEN2 is also added. When the Button is clicked it will move to SecondActivity with the help of Explicit Intents



SecondActivity

It has a text named as Screen2.This activity is associated with a different xml file eg. activity_second.xml



Implicit Intent

Not Specific to particular application. When the user wants to open a link he has 4 choices namely Whatsapp, News Feed, Telegram and ShareMe and he can choose one among them. Eg Throwing a ball and anyone can catch it.

UI Design

1

ViewGroup and Layouts

One or more views can be grouped together into one GroupView. It includes Layouts.

2

Unit of Measurement

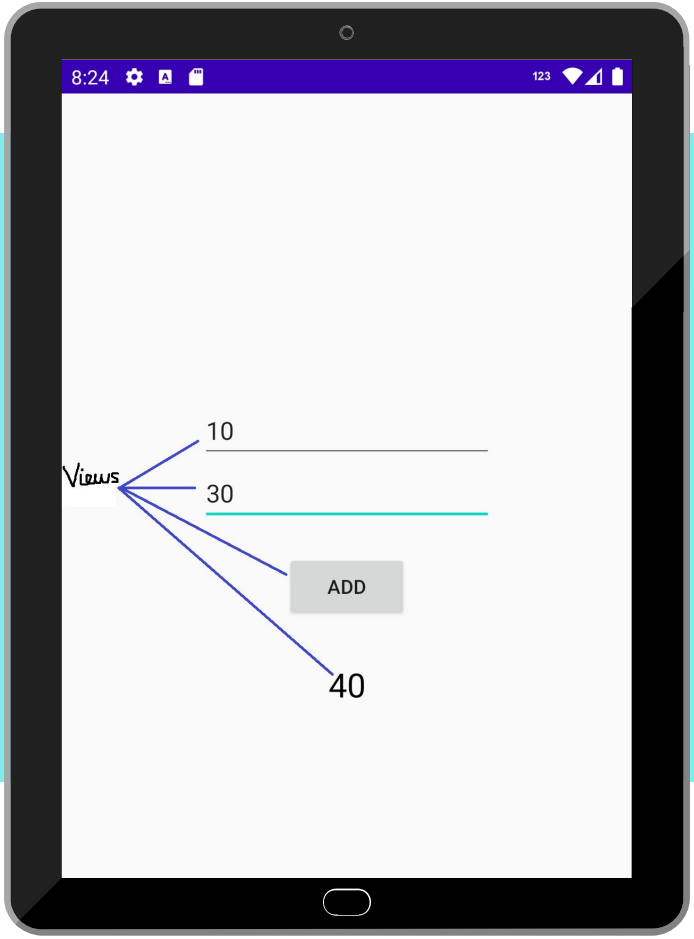
Specifying the size of an element on an Android.

- Density-independent Pixels (dp): for Views
- Scale-independent Pixels (sp): for text

3

Screen Densities

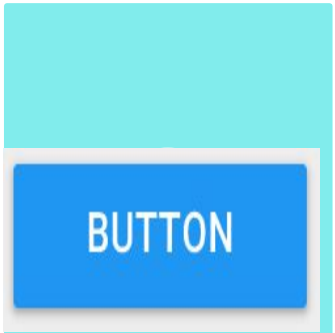
Can be low, medium, high and extra high.



View

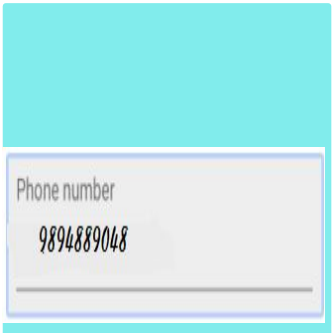
Every User Interface Element is a View.

View Classes Examples



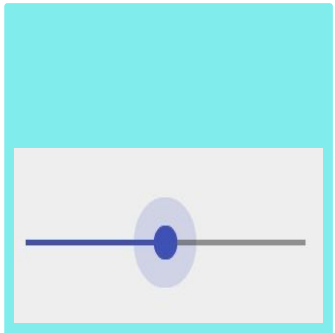
Button

Button class



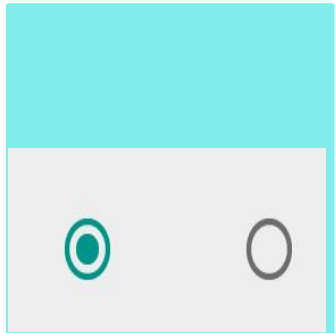
EditText

EditText class



Slider

Slider class



RadioButton

RadioButton class

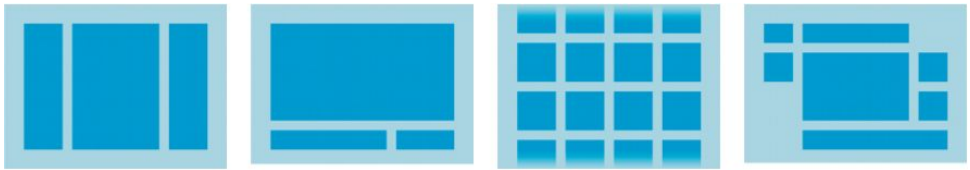
ViewGroup

Viewelements can be grouped inside a ViewGroup, which acts as a container. The relationship is parent-child, in which the *parent* is a ViewGroup, and the *child* is a View or another ViewGroup.

Layouts

- subclasses of ViewGroup
- contain child views
- can be in a row, column, grid, table, absolute

Types of Layout



LinearLayout
A group of child View elements positioned and aligned horizontally or vertically.

RelativeLayout
A group of child View elements in which each element is positioned and aligned relative to other elements within the ViewGroup.

TableLayout
A group of child View elements arranged into rows and columns.

GridLayout
A group that places its child View elements in a rectangular grid that can be scrolled

ConstraintLayout
A group of child View elements using constraints, edges, and guidelines to control how the elements are positioned relative to other elements in the layout. Easy to click and drag View elements in the layout editor.

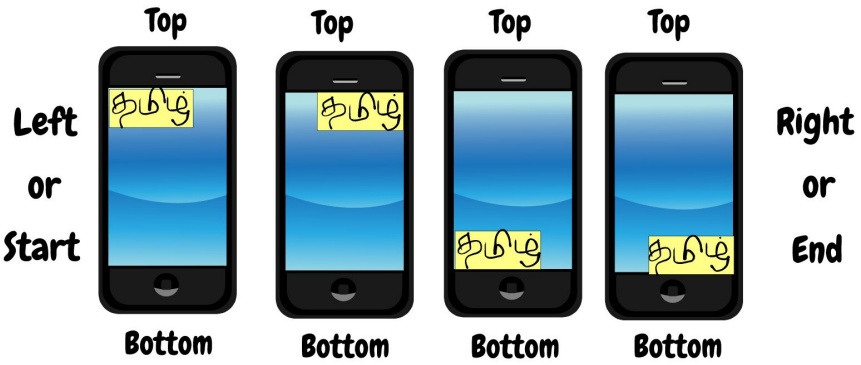
LinearLayout

The layout orientation is either horizontal or vertical. The important attributes are orientation, Weight,Width,Weight, Margin and Padding. The small figure shows that all three TextView occupies same space with the help of weight attribute

RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).

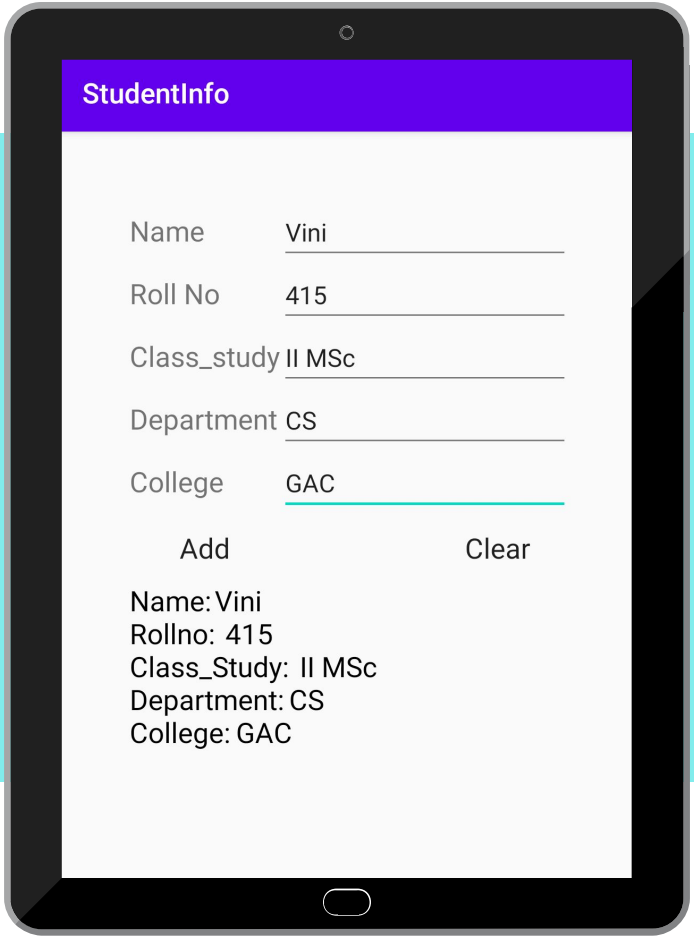
Relative to Parent

The Blue color screen is the Parent view. In the first image the child text Tamil is to the left and top of the parent. In the second image the child text Tamil is to the right and top of the parent. In the third one the child text Tamil is to the bottom and left of the parent and in the last one the child text is at the Bottom and right of the parent



Relative to Siblings

The first image shows the Parent view. The second image shows the child view inside the parent. In the third image to the right of the child image there are two other child views or siblings



TableLayout

Name TextView and EditText to enter Name is kept in a TableRow. TextView Rollno and the corresponding EditText are placed in the second row. Class_study and corresponding EditText is placed in the third row. Department along with EditText is placed in the fourth row. College and its associated EditText is placed in the fifth row. The table contains 5 rows and two columns.

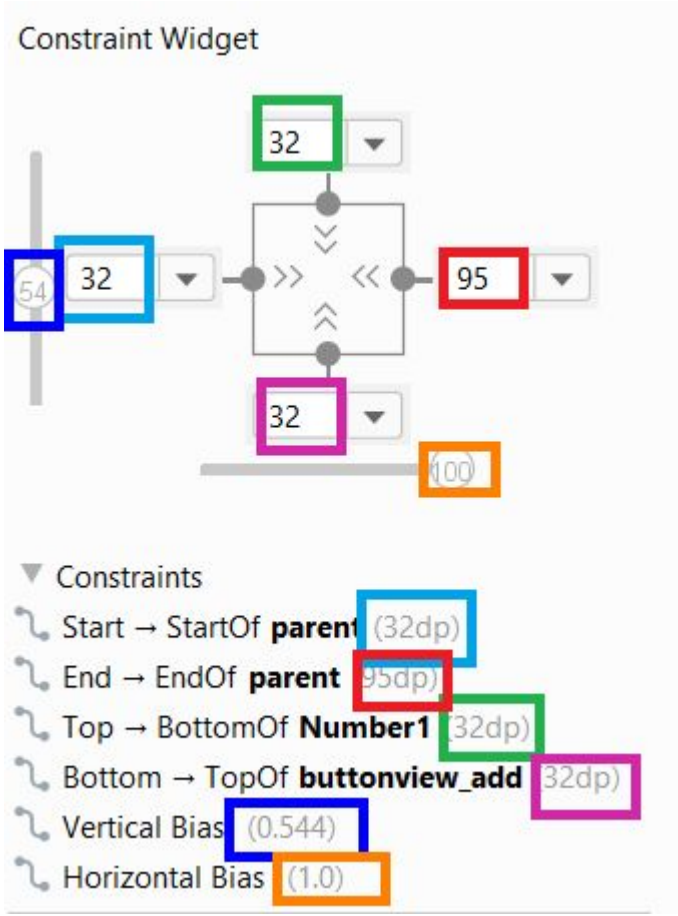
FrameLayout

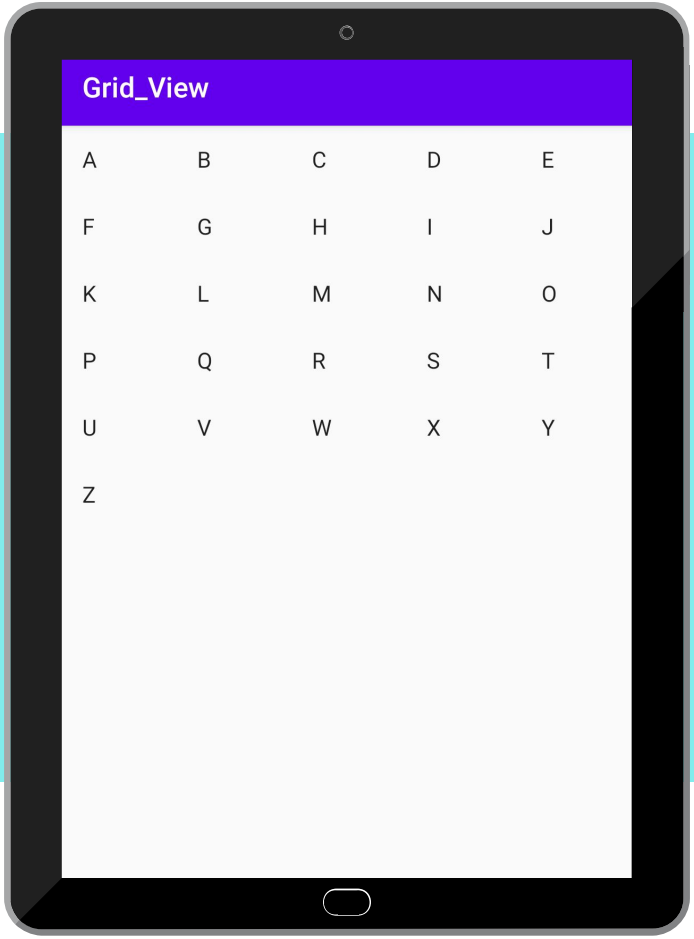
Another Exciting Layout is the frame layout. It works like a stack. The most recent view is placed at the top.

Over the rainbow image the fish image is placed and over the fish image the text :"Swim With Me" is placed

ConstraintLayout

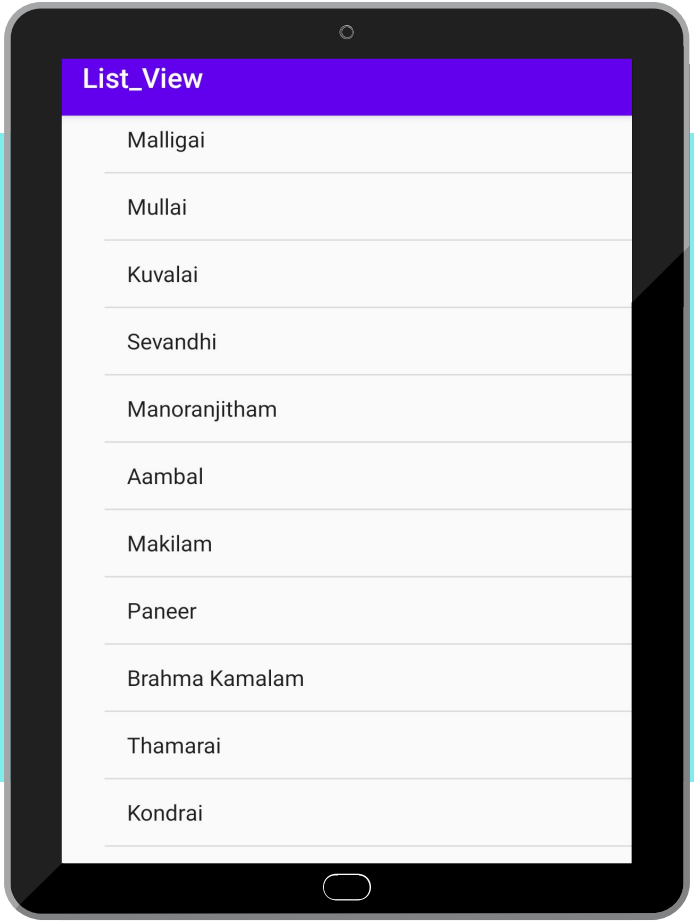
- The Constraint widget shows the six constraints Start, End, Top, Bottom, Vertical Bias, and Horizontal Bias included for EditText that is immediately reflected in code.
- The constraints included can be deleted as well





GridView

A view group that display items in two dimensional scrolling grid (rows and columns). The grid items are automatically inserted to the layout using a ListAdapter. Users can select any grid item by clicking on it. It is scrollable by default.



ListView

- A ViewGroup that displays a vertically-scrollable collection of views, where each view is positioned immediately below the previous view in the list

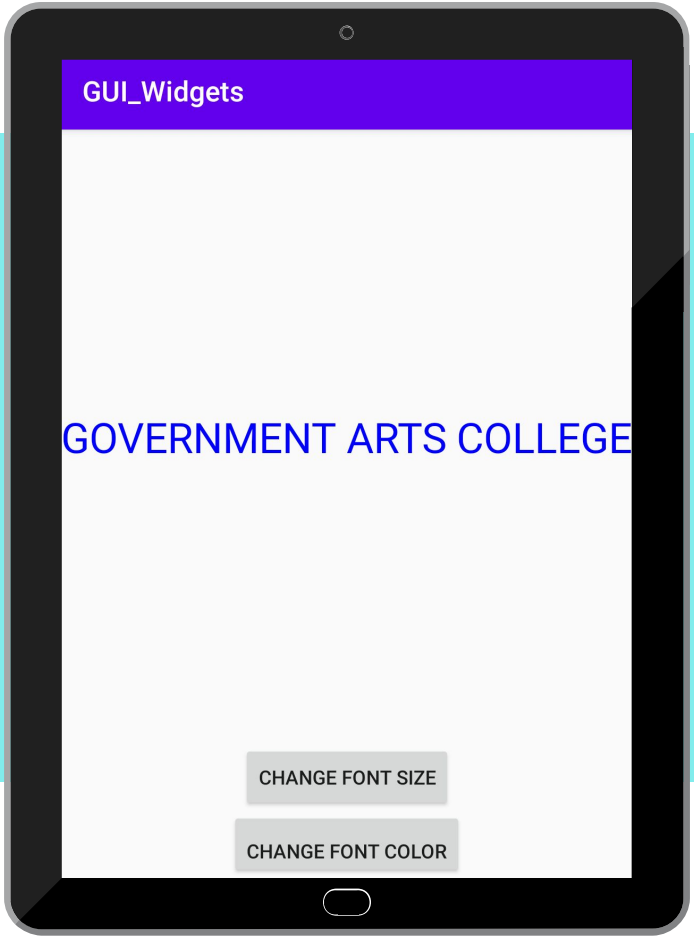
- The ListView displays the names of different flowers. When the name is clicked The name of the corresponding flower will display as a Toast message,

Widgets

Represents a reusable portion of app's UI

Widgets

A miniature app view that appears on the Android home screen and can be updated periodically with new data. App widgets display small amounts of information or perform simple functions such as showing the time, summarizing the day's calendar events, or controlling music playback.



Widget

This widget will change the font size and color of the text GOVERNMENT ARTS COLLEGE

Fragment

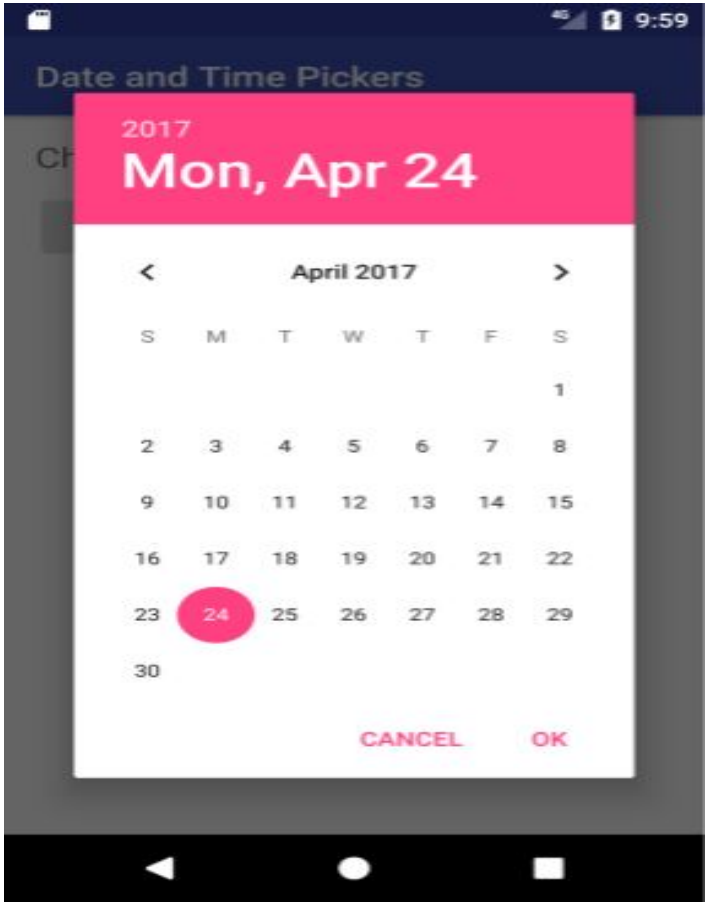
Represents a reusable portion of app's UI

Fragment

A self-contained component with its own user interface (UI) and lifecycle that can be reused in different parts of an app's UI.

Can be a *static* part of the UI of an Activity, which means that the Fragment remains on the screen during the entire lifecycle of the Activity. However, the UI of an Activity is more effective if it adds or removes the Fragment *dynamically* while the Activity is running.

Example of dynamic Fragment is the DatePicker which displays a dialog window floating on top of its Activity window when a user taps a button or an action occurs. The user can click OK or Cancel to close the Fragment.



Fragment

When the Application runs it shows this fragment with two buttons FRAGMENT ONE and FRAGMENT TWO and the displayed text as Fragment One

Fragment

When the Button FRAGMENT TWO is clicked the background changes along with the text.If FRAGMENT ONE is clicked the background changes again along with the text Fragment One

ViewPager

A layout manager that lets the user flip left and right through "pages" (screens) of content.

ViewPager

- often used in conjunction with `Fragment`, which is a convenient way to supply and manage the lifecycle of each "page". `ViewPager` also provides the ability to swipe "pages" horizontally.
- A `ViewPager` is used within the root layout to switch child screens. This provides the ability for the user to swipe from one child screen to another. Users are able to navigate to sibling screens by touching and dragging the screen horizontally in the direction of the desired adjacent screen.
- Swipe views are most appropriate where there is some similarity in content type among sibling pages, and when the number of siblings is relatively small.

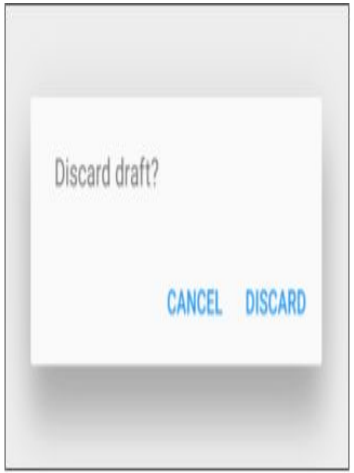
Dialog

Small window that prompts the user to make a decision or enter additional information.

Dialog

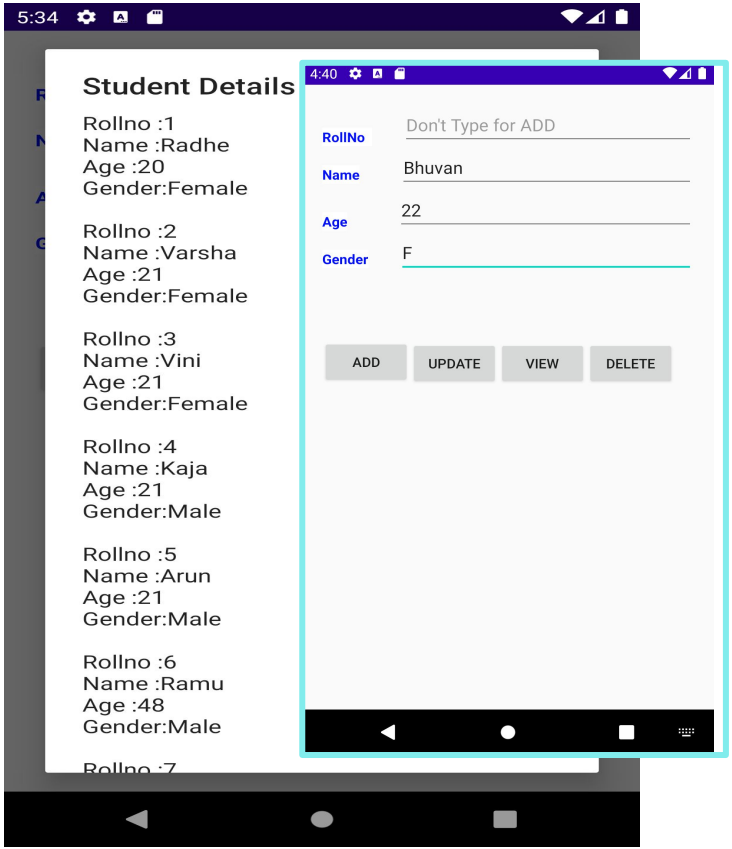
A *dialog* is a window that appears on top of the display or fills the display, interrupting the flow of `Activity`. Dialogs inform users about a specific task and may contain critical information, require decisions, or involve multiple tasks.

For example, an alert dialog might require the user to click **Continue** after reading it, or give the user a choice to agree with an action by clicking a positive button (such as **OK** or **Accept**), or to disagree by clicking a negative button (such as **Cancel**).



Alert Dialog

Student Details alert dialog appears on top of the display and fills it when the user clicks View Button. After seeing the Student Details, the User has to press Back Button to quit the Dialog and the output screen becomes visible.



Thanks!

Does anyone have any queries?

Mail to
buvann@gmail.com

For More Info Visit
https://sudarmigumarivu.blogspot.com

Credits

- https://slidesgo.com/
- https://google-developer-training.github.io/
- https://developer.android.com