-----------------------------------------Material design - introduced in Android Lollipop, API 21 ----------------------------------------------

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices.it is comprehensive guide line for

create app More attractive UI, preformance , flexible, page animaction, good design and invoaction.

Like- App bar, ToolBar, recycalview, cartview,floating button.

#1. App bar --

The app bar, formerly known as the action bar in Android, is a special kind of toolbar that’s used for branding, navigation, search, and actions

#2. Toolbars --

it is the spiritual successor of the ActionBar. It's a ViewGroup that can be placed anywhere in your XML layouts. ToolBar's appearance and behavior

can be more easily customized than the ActionBar.ToolBar is implemented in the android.support.v7.widget.Toolbar class.. One of the biggest

advantages of using the Toolbar widget is that you can place the view anywhere within your layout

@2.0 There are two ways to use Toolbar:

a> Use a Toolbar as an Action Bar when you want to use the existing ActionBar facilities (such as menu inflation and selection, ActionBarDrawerToggle,

and so on) but want to have more control over its appearance.

b> Use a standalone Toolbar when you want to use the pattern in your app for situations that an Action Bar would not support; for example, showing

multiple toolbars on the screen, spanning only part of the width, and so on.

@2.1 . ToolBar vs ActionBar --

1. ToolBar is a View included in a layout like any other View, Multiple distinct ToolBar elements can be defined within a single activity,

As a regular View, the toolbar is easier to position, animate and control

2. ActionBar continues to work and if all you need is a static bar at the top that can host icons and a back button, then you can safely continue to use ActionBar

@2.2 how to make n use

1. create layout and put/ add Toolbar.

2. in your activity - find toolbar and setsupportactionbar

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

#3 Styling and Theme -

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary">@color/colorPrimary</item>

<item name="colorPrimaryDark">@color/colorPrimaryDark</item>

<item name="colorAccent">@color/colorAccent</item>

</style>

<style name="ToolbarTheme" parent="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

<!-- android:textColorPrimary is the color of the title text in the Toolbar -->

<item name="android:textColorPrimary">@android:color/holo\_blue\_light</item>

<!-- actionMenuTextColor is the color of the text of action (menu) items -->

<item name="actionMenuTextColor">@android:color/holo\_green\_light</item>

<!-- Tints the input fields like checkboxes and text fields -->

<item name="colorAccent">@color/cursorAccent</item>

<!-- Applies to views in their normal state. -->

<item name="colorControlNormal">@color/controlNormal</item>

<!-- Applies to views in their activated state (i.e checked or switches) -->

<item name="colorControlActivated">@color/controlActivated</item>

<!-- Applied to framework control highlights (i.e ripples or list selectors) -->

<item name="colorControlHighlight">@color/controlActivated</item>

<!-- Enable these below if you want clicking icons to trigger a ripple effect -->

<!--

<item name="selectableItemBackground">?android:selectableItemBackground</item>

<item name="selectableItemBackgroundBorderless">?android:selectableItemBackground</item>

-->

</style>

<!-- This configures the styles for the title within the Toolbar -->

<style name="Toolbar.TitleText" parent="TextAppearance.Widget.AppCompat.Toolbar.Title">

<item name="android:textSize">21sp</item>

<item name="android:textStyle">italic</item>

</style>

#4 CoordinatorLayout

The CoordinatorLayout can be used to create floating effects using the layout\_anchor and layout\_gravity attributes. CoordinatorLayout is used to

create scrolling and "floating" effects within a layout. it is use as a container for a specific interaction with one or more child views. By

specifying Behaviors for child views of a CoordinatorLayout you can provide many different interactions within a single parent and those views can

also interact with one another.The CoordinatorLayout is a super-powered FrameLayout . CoordinatorLayout is its ability to coordinate the animations

and transitions of the views within it

#5 Cards

CardView extends the FrameLayout class and lets you show information inside cards that have a consistent look across the platform. CardView widgets

can have shadows and rounded corners.To create a card with a shadow, use the card\_view:cardElevation attribute. The CardView UI component shows

information inside cards. We can customise its corners, the elevation and so on.These cards will be the rows of RecyclerView. A FrameLayout with a rounded corner background and shadow.

#6 RecyclerView

The RecyclerView is a more advanced and more flexible version of the ListView.This widget is a container for displaying large data sets that can be scrolled very efficiently and smoothness by maintaining a limited number of views. To use the [RecyclerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html) widget, you have to specify an adapter and a layout manager. To create an adapter, extend the [RecyclerView.Adapter](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter.html) class.

@6.1 - providing:

Layout managers for positioning items

Default animations for common item operations, such as removal or addition of items