**Module 4:**

Android UI components: Text Controls, Buttons, Widgets, Layouts, Containers

Task: Explore all the UI Controls and design a Student Registration Activity

Android UI Controls

There are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

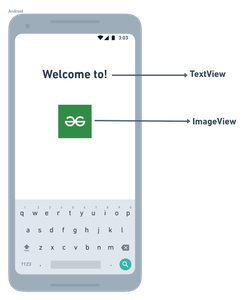
|  |  |
| --- | --- |
| **Sr.No.** | **UI Control & Description** |
| 1 | [**TextView**](https://www.tutorialspoint.com/android/android_textview_control.htm)  This control is used to display text to the user. |
| 2 | [**EditText**](https://www.tutorialspoint.com/android/android_edittext_control.htm)  EditText is a predefined subclass of TextView that includes rich editing capabilities. |
| 3 | [**AutoCompleteTextView**](https://www.tutorialspoint.com/android/android_autocompletetextview_control.htm)  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 | [**Button**](https://www.tutorialspoint.com/android/android_button_control.htm)  A push-button that can be pressed, or clicked, by the user to perform an action. |
| 5 | [**ImageButton**](https://www.tutorialspoint.com/android/android_imagebutton_control.htm)  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 | [**CheckBox**](https://www.tutorialspoint.com/android/android_checkbox_control.htm)  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 | [**ToggleButton**](https://www.tutorialspoint.com/android/android_togglebutton_control.htm)  An on/off button with a light indicator. |
| 8 | [**RadioButton**](https://www.tutorialspoint.com/android/android_radiobutton_control.htm)  The RadioButton has two states: either checked or unchecked. |
| 9 | [**RadioGroup**](https://www.tutorialspoint.com/android/android_radiogroup_control.htm)  A RadioGroup is used to group together one or more RadioButtons. |
| 10 | [**ProgressBar**](https://www.tutorialspoint.com/android/android_progressbar.htm)  The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background. |
| 11 | [**Spinner**](https://www.tutorialspoint.com/android/android_spinner_control.htm)  A drop-down list that allows users to select one value from a set. |
| 12 | [**TimePicker**](https://www.tutorialspoint.com/android/android_timepicker_control.htm)  The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode. |
| 13 | [**DatePicker**](https://www.tutorialspoint.com/android/android_datepicker_control.htm)  The DatePicker view enables users to select a date of the day. |

### **Android UI Layouts**

Android **Layout** is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. Generally, every application is a combination of View and ViewGroup. As we know, an android application contains a large number of activities and we can say each activity is one page of the application. So, each activity contains multiple user interface components and those components are the instances of the View and ViewGroup. All the elements in a layout are built using a hierarchy of **View**and **ViewGroup**objects.

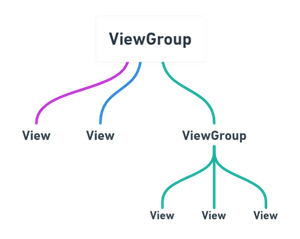
### **View**

A **View** is defined as the user interface which is used to create interactive UI components such as [TextView](https://www.geeksforgeeks.org/textview-widget-in-android-using-java-with-examples/), [ImageView](https://www.geeksforgeeks.org/imageview-in-android-with-example/), [EditText](https://www.geeksforgeeks.org/edittext-widget-in-android-using-java-with-examples/), [RadioButton](https://www.geeksforgeeks.org/radiobutton-in-kotlin/), etc., and is responsible for event handling and drawing. They are Generally Called Widgets.

[](https://whimsical.com/91txrXHfW3NywpD1qBXxhD)

*View*

A **ViewGroup** act as a base class for layouts and layouts parameters that hold other Views or ViewGroups and to define the layout properties. They are Generally Called layouts.

[](https://whimsical.com/viewgroup-8fCbKjTrG2rG76qpCiANX3)

*ViewGroup*

The Android framework will allow us to use UI elements or widgets in two ways:

* Use UI elements in the XML file
* Create elements in the Kotlin file dynamically

### **Types of Android Layout**

* **Android Linear Layout:**LinearLayout is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property.
* **Android Relative Layout:**RelativeLayout is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fix to the top of the parent).
* **Android Constraint Layout:**ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.
* **Android Frame Layout:**FrameLayout is a ViewGroup subclass, used to specify the position of View elements it contains on the top of each other to display only a single View inside the FrameLayout.
* **Android Table Layout:**TableLayout is a ViewGroup subclass, used to display the child View elements in rows and columns.
* **Android Web View:**WebView is a browser that is used to display the web pages in our activity layout.
* **Android ListView:**ListView is a ViewGroup, used to display scrollable lists of items in a single column.
* **Android Grid View:**GridView is a ViewGroup that is used to display a scrollable list of items in a grid view of rows and columns.

# **Android Login and Registration Screen Design**

Create a new android application using android studio and give names as **LoginExample**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Once we create an application, open **activity\_main.xml** file from **\res\layout** folder path and write the code like as shown below.

## activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <TextView  
        android:id="@+id/loginscrn"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="80dp"  
        android:text="Login"  
        android:textSize="25dp"  
        android:textStyle="bold"  
        android:layout\_gravity="center"/>  
    <TextView  
        android:id="@+id/fstTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:layout\_marginTop="20dp"  
        android:text="Email"/>  
    <EditText  
        android:id="@+id/txtEmail"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:ems="10"/>  
    <TextView  
        android:id="@+id/secTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Password"  
        android:layout\_marginLeft="100dp" />  
    <EditText  
        android:id="@+id/txtPwd"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:inputType="textPassword"  
        android:ems="10" />  
    <Button  
        android:id="@+id/btnLogin"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:text="Login" />  
    <TextView android:id="@+id/lnkRegister"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="40dp"  
        android:text="New to Tutlane? Register here"  
        android:gravity="center"  
        android:textSize="20dp"  
        android:textColor="#3F51B5"/>  
</LinearLayout>

Now we will create another layout resource file **registration.xml** in **\res\layout** path to allow new users to register in our application for that right click on your **layout** folder à Go to **New** à select **Layout Resource File** and give name as **registration.xml**.

Once we create a new layout resource file **registration.xml**, open it and write the code like as shown below

## registration.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <TextView  
        android:id="@+id/loginscrn"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="80dp"  
        android:text="Registration"  
        android:textSize="25dp"  
        android:textStyle="bold"  
        android:layout\_gravity="center"/>  
    <TextView  
        android:id="@+id/fstTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:layout\_marginTop="20dp"  
        android:text="Full Name"/>  
    <EditText  
        android:id="@+id/txtName"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:ems="10"/>  
    <TextView  
        android:id="@+id/secTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Email"  
        android:layout\_marginLeft="100dp" />  
    <EditText  
        android:id="@+id/txtEmail"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:ems="10" />  
    <TextView  
        android:id="@+id/thirdTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Password"  
        android:layout\_marginLeft="100dp" />  
    <EditText  
        android:id="@+id/txtPwd"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:inputType="textPassword"  
        android:ems="10" />  
    <Button  
        android:id="@+id/btnLogin"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginLeft="100dp"  
        android:text="Login" />  
    <TextView android:id="@+id/lnkLogin"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_marginTop="40dp"  
        android:text="Already Registered? Login here"  
        android:gravity="center"  
        android:textSize="20dp"  
        android:textColor="#3F51B5"  
        android:onClick="test"/>  
</LinearLayout>

Now open your main activity file **MainActivity.java** from **\java\com.tutlane.loginexample** path and write the code like as shown below

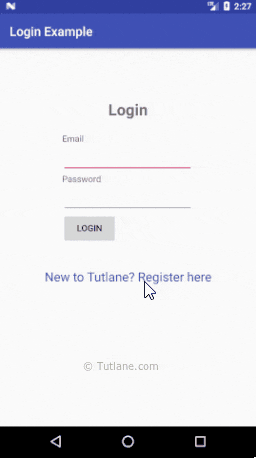
## MainActivity.java

package com.tutlane.loginexample;  
import android.content.Intent;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.text.method.LinkMovementMethod;  
import android.view.View;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        TextView register = (TextView)findViewById(R.id.lnkRegister);  
        register.setMovementMethod(LinkMovementMethod.getInstance());  
        register.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Intent intent = new Intent(MainActivity.this, RegistrationActivity.class);  
                startActivity(intent);  
            }  
        });  
    }  
}

If you observe the above code, whenever the user click on **register** link, we are redirecting the user from login screen to registration screen using “**RegistrationActivity**” for that create another [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) file **RegistrationActivity.java** in **\java\com.tutlane.loginexample** path.

## Output :

When we run the above example in the android emulator we will get a result like as shown below.



**Module 5:**

Material Design for Android: Material theme and widgets, Elevation shadows, Cards, Animations, Drawable

Task: Design the Student Registration Activity using Material Design for Android Components.

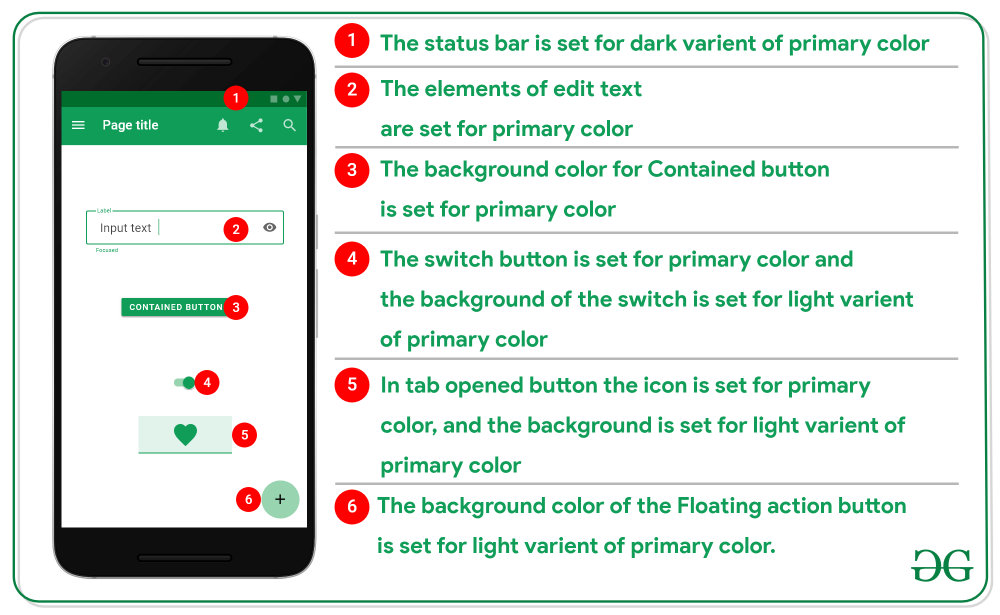
# **Introduction to Material Design in Android**:

Material Design Components (MDC Android) offers designers and developers a way to implement Material Design in their Android application. Developed by a core team of engineers and UX designers at Google, these components enable a reliable development workflow to build beautiful and functional Android applications. Material design in Android is one of the key features that attracts and engages the customer towards the application. This is a special type of design, which is guided by Google. So in this article, it has been introduced to the basic things that need to be considered before designing or developing any Materialistic Android Application.

### ***1. Colors and Theming***

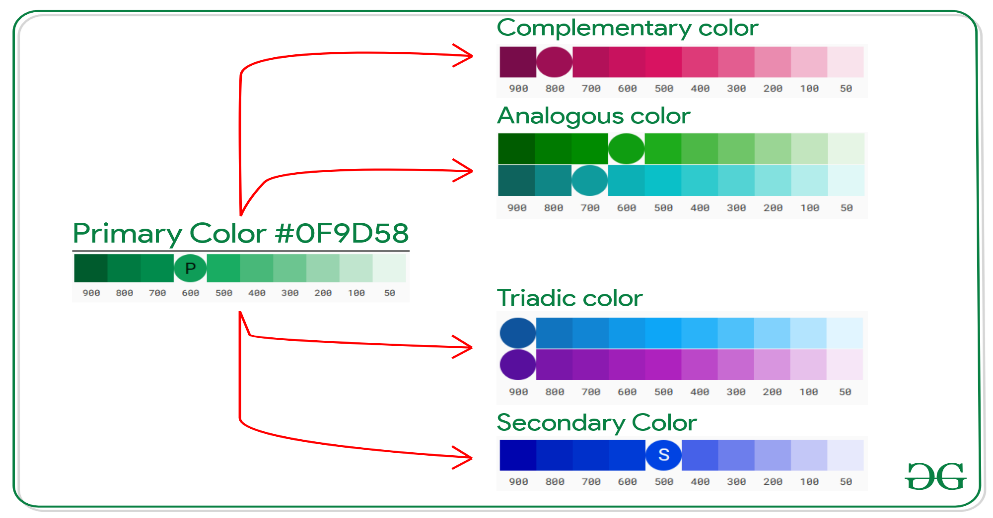
By choosing the right kind of color combination reflects the application’s brand and style. For example, the application’s main or primary color is the Green, then in the whole application, the green color will be frequently shown. Choosing the color for the application, there are three types of colors to be chosen for developing the android application.

* **Primary Color:**This color should be chosen very cautiously because this color is frequently visible in the application components like high emphasis buttons or the button ripple color, and also the top and bottom navigation bar.
* **Secondary Color:**This color should be chosen only when there is a low complexity level of the application. This color will be applied to those elements which need a little color accent like the background color for the Floating Action Buttons (FAB), sliders, toggle buttons, chips (Active State), progress bars, etc.
* **Light and Dark variants:**These colors are the variants of the primary color. The dark variant of the primary color is set for the status bar and the light variant of the primary color is set for the Floating action button, outline for the edit texts, and where the elements need some color accents the light variant of the primary colors will be set for them. Have a look at the following image when only the primary color is set for the application theme looks like.



**But according to the complexity of the application, there are various types of color can be chosen.**

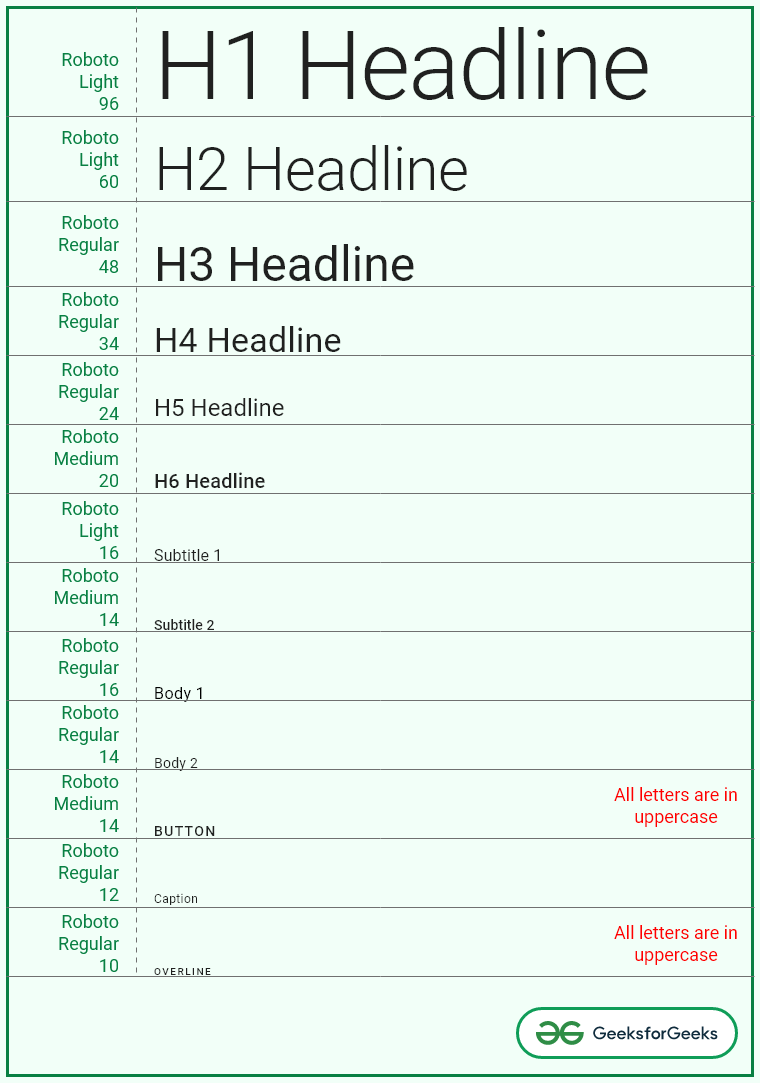
* Have a look at the following image, having selected the primary color, with that the **Complementary color** can be chosen or **Analogous color** can be chosen or **Triadic color** can be chosen or **Secondary color**can be chosen to meet the required style.



* To make color swatches of their own can visit [here](https://material.io/design/color/the-color-system.html#tools-for-picking-colors).

### ***2. Typography (Choosing the Right Font)***

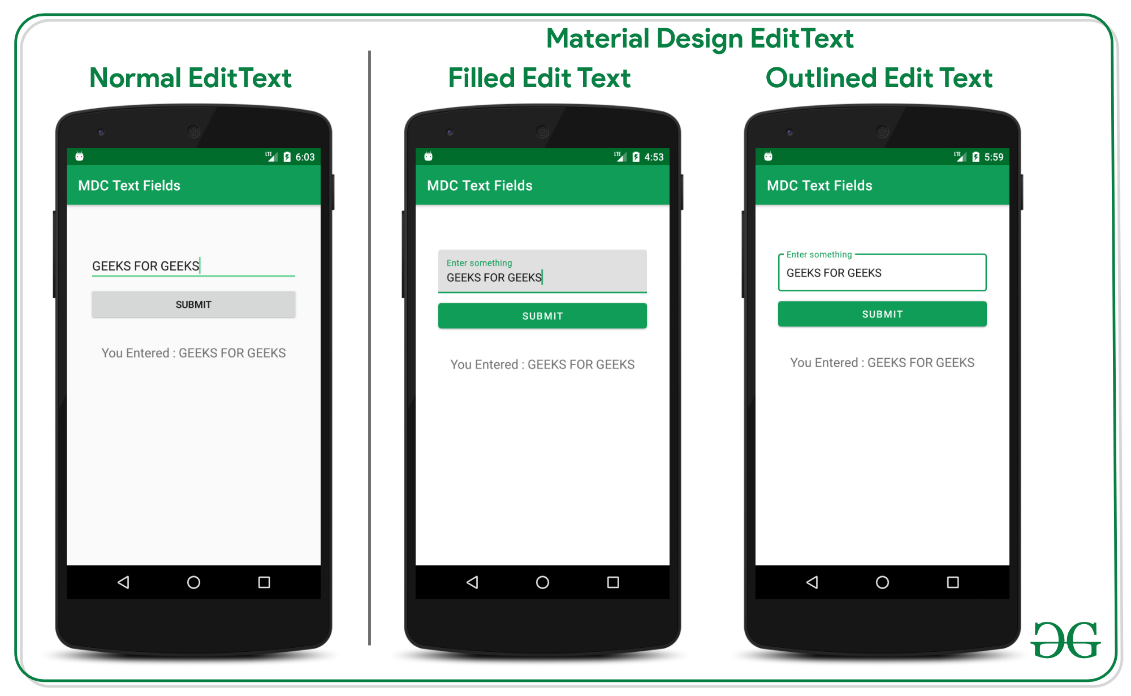
* In android, however, the ***Roboto***font meets all the requirements. But too, if the developer wants to customize the application more with the font, the font needs to be chosen where it has all its variants. The variants are the light face, regular face, medium face, and sometimes the dark face.
* Choosing the font from [Google Font](https://fonts.google.com/) is recommended. As it offers a variety of font families, and almost all the fonts have contained all the variants.
* There are some guidelines that need to be followed by having the font chosen. In this case, the ***Roboto*** is chosen for demonstration purposes only. Looking at the following image which is the type scale chart for applying the styles of the fonts for various contexts.
* The various contexts include captions, Body, Subtitles, Button, captions, etc.
* In the below image left side column contains the font is chosen, font style, and the font size. The second column contains the preview of the selected context of the font.



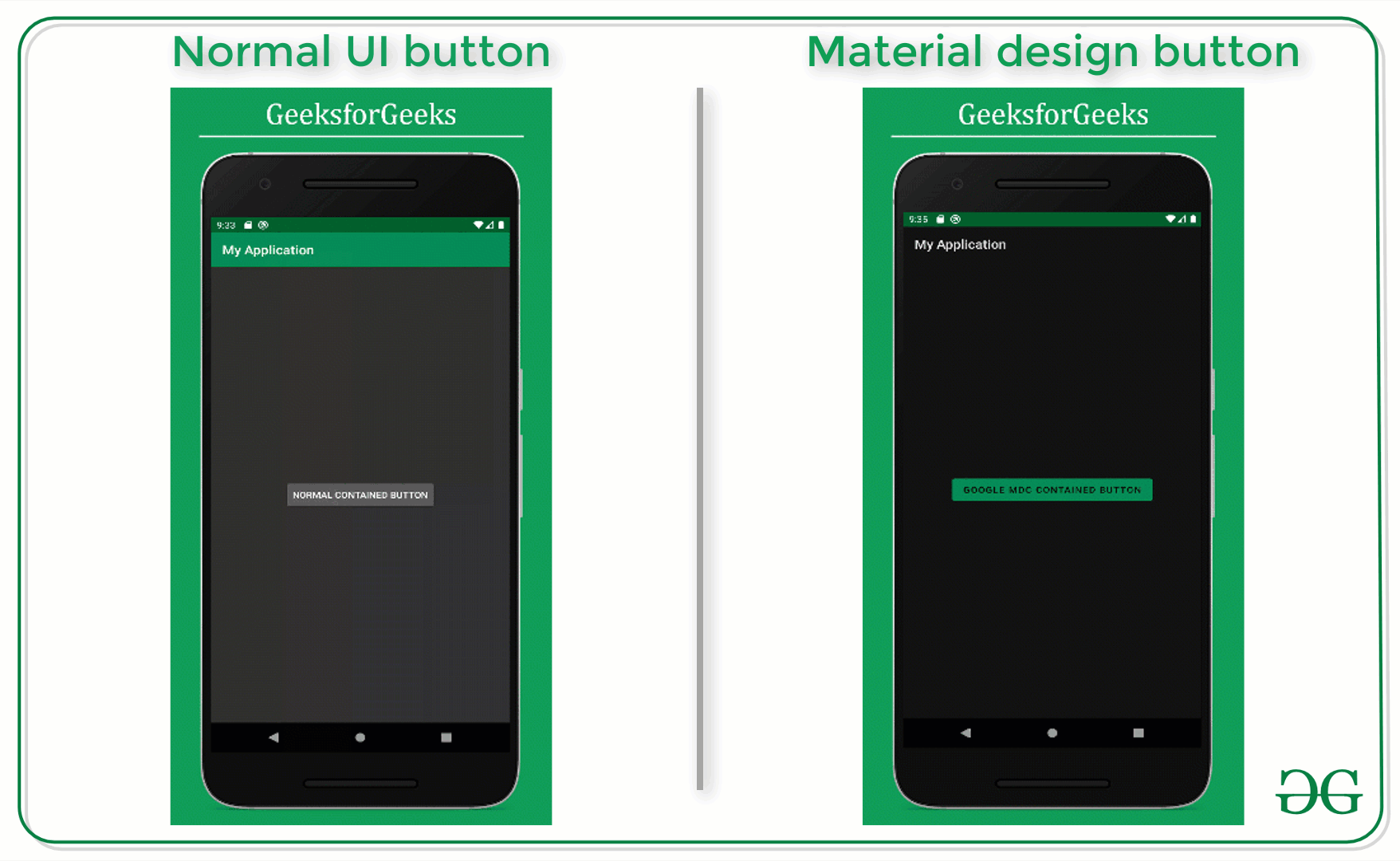
* To select the right kind of font for the application can visit [this](https://material.io/design/typography/the-type-system.html#type-scale).

### ***3. Material Design Components***

* Material design components are the components that allow a lot of features for the users and easy to implement for the developers.
* Have a look at the following image on how material design components stand out in terms of the customization, styling, and look, from the normal UI components.



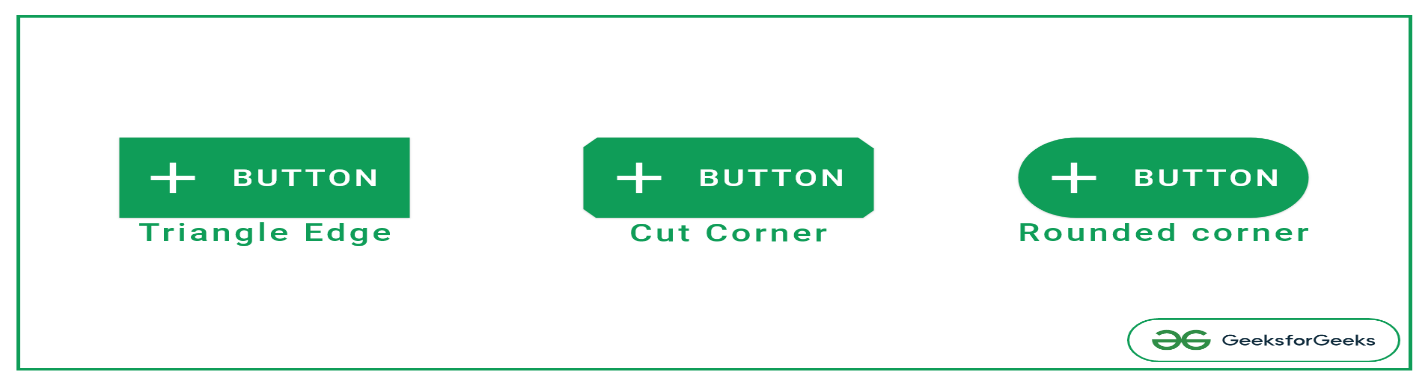
* The features offered by the material design components can be implemented in various contexts. One can notice in the above image how the normal button and edit text has adapted to the application’s theme and how the material design button has adapted to the application’s theme.
* These components can even adapt to the dark theme and change their styles when it is toggled by the user. Have look at the following image to differentiate the behaviors between the material design components and normal UI components.



### **4. Shaping the Components**

* In material design, there are three types of shaping methods.

1. Cut corner
2. Rounded corner
3. Triangle edge



* These methods are also can be applied for the material design buttons, text fields, chips, floating action buttons, cards, navigation bars, bottom sheets, etc.
* These features are available with the Material design components out of the proverbial box. Just need to add the dependency of the material design components and start implementing the styling for material design components.

1. Open the build.gradle file for your application.
2. Make sure that the repositories section includes Google's Maven Repository google(). For example:

 allprojects {

   repositories {

     google()

     jcenter()

  }

}

1. Add the library to the dependencies section:

 dependencies {

   // ...

   implementation 'com.google.android.material:material:<version>'

   // ...

}

Visit [Google's Maven Repository](https://maven.google.com/web/index.html#com.google.android.material:material) or [MVN Repository](https://mvnrepository.com/artifact/com.google.android.material/material) to find the latest version of the library.

#### New Namespace and AndroidX

If your app currently depends on the original Design Support Library, you can make use of the [Refactor to AndroidX…](https://developer.android.com/jetpack/androidx/migrate) option provided by Android Studio. Doing so will update your app's dependencies and code to use the newly packaged androidx and com.google.android.material libraries.

If you don't want to switch over to the new androidx and com.google.android.material packages yet, you can use Material Components via the com.android.support:design:28.0.0 dependency.

Note: You should not use the com.android.support and com.google.android.material dependencies in your app at the same time.

### 2. Compile your app with Android 10

In order to use Material Components for Android, and the latest versions of the Support Libraries, you will have to install Android Studio 3.5 or higher to build with Android 10, and update your app's compileSdkVersion to 29.

### 3. Ensure you are using **AppCompatActivity**

Using AppCompatActivity will ensure that all the components work correctly. If you are unable to extend from AppCompatActivity, update your activities to use AppCompatDelegate. This will enable the AppCompat versions of components to be inflated among other important things.

### 4. Change your app theme to inherit from a Material Components theme

Doing an app-wide migration by changing your app theme to inherit from a Material Components theme is the recommended approach. However, be sure to test thoroughly afterwards, as components in existing layouts may change their looks and behavior.

Note: If you **can't** change your theme, you can do one of the following:

* Inherit from one of our Material Components **Bridge** themes. See the [**Bridge Themes**](https://m2.material.io/develop/android/docs/getting-started#bridge-themes) section for more details.
* Continue to inherit from an AppCompat theme and add some new theme attributes to your theme. See the **[AppCompat Themes](https://m2.material.io/develop/android/docs/getting-started" \l "appcompat-themes" \o "" \t "_self)** section for more details.

#### **Material Components themes**

The following is the list of Material Components themes you can use to get the latest component styles and theme-level attributes.

* Theme.MaterialComponents
* Theme.MaterialComponents.NoActionBar
* Theme.MaterialComponents.Light
* Theme.MaterialComponents.Light.NoActionBar
* Theme.MaterialComponents.Light.DarkActionBar
* Theme.MaterialComponents.DayNight
* Theme.MaterialComponents.DayNight.NoActionBar
* Theme.MaterialComponents.DayNight.DarkActionBar

Update your app theme to inherit from one of these themes, e.g.:

<style name="Theme.MyApp" parent="Theme.MaterialComponents.DayNight">

   <!-- ... -->

</style>

For more information on how to set up theme-level attributes for your app, take a look at our [Theming](https://m2.material.io/develop/android/theming/theming-overview) guide, as well as our [Dark Theme](https://m2.material.io/develop/android/theming/dark) guide for why it's important to inherit from the DayNight theme.

Note: Using a Material Components theme enables a custom view inflater which replaces default components with their Material counterparts. Currently, this only replaces <Button> and <AutoCompleteTextView> XML components with [<MaterialButton>](https://www.github.com/material-components/material-components-android/tree/master/docs/components/Button.md) and [<MaterialAutoCompleteTextView>](https://github.com/material-components/material-components-android/tree/master/lib/java/com/google/android/material/textfield/MaterialAutoCompleteTextView.java), respectively.

#### **Bridge Themes**

If you cannot change your theme to inherit from a Material Components theme, you can inherit from a Material Components **Bridge** theme.

<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light.Bridge">

   <!-- ... -->

</style>

Both Theme.MaterialComponents and Theme.MaterialComponents.Light have .Bridge themes:

* Theme.MaterialComponents.Bridge
* Theme.MaterialComponents.Light.Bridge
* Theme.MaterialComponents.NoActionBar.Bridge
* Theme.MaterialComponents.Light.NoActionBar.Bridge
* Theme.MaterialComponents.Light.DarkActionBar.Bridge

Bridge themes inherit from AppCompat themes, but also define the new Material Components theme attributes for you. If you use a bridge theme, you can start using Material Design components without changing your app theme.

#### **AppCompat Themes**

You can also incrementally test new Material components without changing your app theme. This allows you to keep your existing layouts looking and behaving the same, while introducing new components to your layout one at a time.

However, you must add the following new theme attributes to your existing app theme, or you will encounter ThemeEnforcement errors:

<style name="Theme.MyApp" parent="Theme.AppCompat">

​

 <!-- Original AppCompat attributes. -->

 <item name="colorPrimary">@color/my\_app\_primary\_color</item>

 <item name="colorSecondary">@color/my\_app\_secondary\_color</item>

 <item name="android:colorBackground">@color/my\_app\_background\_color</item>

 <item name="colorError">@color/my\_app\_error\_color</item>

​

 <!-- New MaterialComponents attributes. -->

 <item name="colorPrimaryVariant">@color/my\_app\_primary\_variant\_color</item>

 <item name="colorSecondaryVariant">@color/my\_app\_secondary\_variant\_color</item>

 <item name="colorSurface">@color/my\_app\_surface\_color</item>

 <item name="colorOnPrimary">@color/my\_app\_color\_on\_primary</item>

 <item name="colorOnSecondary">@color/my\_app\_color\_on\_secondary</item>

 <item name="colorOnBackground">@color/my\_app\_color\_on\_background</item>

 <item name="colorOnError">@color/my\_app\_color\_on\_error</item>

 <item name="colorOnSurface">@color/my\_app\_color\_on\_surface</item>

 <item name="scrimBackground">@color/mtrl\_scrim\_color</item>

 <item name="textAppearanceHeadline1">@style/TextAppearance.MaterialComponents.Headline1</item>

 <item name="textAppearanceHeadline2">@style/TextAppearance.MaterialComponents.Headline2</item>

 <item name="textAppearanceHeadline3">@style/TextAppearance.MaterialComponents.Headline3</item>

 <item name="textAppearanceHeadline4">@style/TextAppearance.MaterialComponents.Headline4</item>

 <item name="textAppearanceHeadline5">@style/TextAppearance.MaterialComponents.Headline5</item>

 <item name="textAppearanceHeadline6">@style/TextAppearance.MaterialComponents.Headline6</item>

 <item name="textAppearanceSubtitle1">@style/TextAppearance.MaterialComponents.Subtitle1</item>

 <item name="textAppearanceSubtitle2">@style/TextAppearance.MaterialComponents.Subtitle2</item>

 <item name="textAppearanceBody1">@style/TextAppearance.MaterialComponents.Body1</item>

 <item name="textAppearanceBody2">@style/TextAppearance.MaterialComponents.Body2</item>

 <item name="textAppearanceCaption">@style/TextAppearance.MaterialComponents.Caption</item>

 <item name="textAppearanceButton">@style/TextAppearance.MaterialComponents.Button</item>

 <item name="textAppearanceOverline">@style/TextAppearance.MaterialComponents.Overline</item>

​

</style>

### 5. Add a Material component to your app

Take a look at our [documentation](https://m2.material.io/components/android) for the full list of available Material components. Each component's page has specific instructions on how to implement it in your app.

Let's use [text fields](https://www.github.com/material-components/material-components-android/tree/master/docs/components/TextField.md) as an example.

#### **Implementing a text field via XML**

The default [filled text field](https://m2.material.io/go/design-text-fields#filled-text-field) XML is defined as:

<com.google.android.material.textfield.TextInputLayout

   android:layout\_width="match\_parent"

   android:layout\_height="wrap\_content"

   android:hint="@string/textfield\_label">

​

 <com.google.android.material.textfield.TextInputEditText

     android:layout\_width="match\_parent"

     android:layout\_height="wrap\_content"/>

</com.google.android.material.textfield.TextInputLayout>

Note: If you are **not** using a theme that inherits from a Material Components theme, you will have to specify the text field style as well, via style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"

Other text field styles are also provided. For example, if you want an [outlined text field](https://m2.material.io/go/design-text-fields#outlined-text-field) in your layout, you can apply the Material Components outlined style to the text field in XML:

<com.google.android.material.textfield.TextInputLayout

   style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

   android:layout\_width="match\_parent"

   android:layout\_height="wrap\_content"

   android:hint="@string/textfield\_label">

​

 <com.google.android.material.textfield.TextInputEditText

     android:layout\_width="match\_parent"

     android:layout\_height="wrap\_content"/>

</com.google.android.material.textfield.TextInputLayout>