**ACCESSIBILITY IN ANDROID WITH MATERIAL DESIGN – CODE IN MOBILE**

**HUMCOM1**

**FEBRUARY 25, 2021**

**Accessibility in Android With Material Design — Code-In Mobile**

# As part of Material Design Guidelines, Android added help for developers and designers to improve accessibility in Android apps. The topic of accessibility has become very important in web apps, this because indexing in search engines is improved when apps follow accessibility best practices. As this has not been enforced in mobile applications so much, we often forget it is key for inclusion and user experience. Accessibility in Android is very important and we will learn about it in this post.

# **Accessibility vs. Usability**

Before diving into the details of accessibility in Android, I would like to clarify the distinction between accessibility and usability. Those two terms are usually used interchangeably, but they refer to different things.

* Accessibility: This is concerned with equivalent user experience for people with disabilities. If we implement this correctly, they can interact with tools without problems.
* Usability: Is related to User Experience (UX). This is for all our users, not specifically for users with disabilities. Sadly, not all usability practices are inclusive with users that need accessibility.

For a broader explanation on this topic, take a look at [this](https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/) article from the W3C.

One important thing to note is that disabilities are not necessarily permanent conditions. If for example, one of your users breaks an arm, you want to help them keep using your product while they go through recovery.

# **Accessibility in Android and Google**

Google has several guidelines regarding accessibility, Android as an open-sourced product from them is not behind. If you want to have a look at the Material Design Accessibility Guide, you can access it [here](https://material.io/design/usability/accessibility.html).

Android bases its accessibility guide in three main principles:

## Clarity

This is not just a topic of accessibility. In general, UX practices from the early days have been moved to clear interfaces. This can mean several things, but let’s look at three clarity examples:

1. Things are spread out enough through the screen so that elements don’t overlap.
2. Labels in buttons have a meaning that resembles the actions they execute.
3. Visual hierarchy, so that your eyes are guided by the content depending on what you should focus on.

## Robustness

I will take the definition of Cambridge dictionary for this:

The quality of being strong, and healthy or unlikely to break or fail

<https://dictionary.cambridge.org/us/dictionary/english/robustness>

When talking about apps, the last three words are key: unlikely to **break or fail**. In this case, a robust app is unlikely to break when users require accessibility features, and they don’t fail to users with these needs. This means incorporating enough tools to fulfill these objectives. Luckily, most things are already included in the Android OS, and we just need to follow the guides.

## Specificity

This refers to the platform you are using. If you are developing apps for Android phones, you usually think of input methods things like touch screen, keyboard, and maybe a mouse; but you should also look at things like input through voice commands.

# **Accessibility for Android developers**

What we have discussed until now, is the base of accessibility in design, interface, and development. But let’s see how do developers implement this in order to make apps accessible.

The first general recommendation is to add a content description to all XML elements.

android:contentDescription="@string/descriptionOfElement"

There is only one exception to this, TextView, does not need a description.

Also, note that sometimes you just add elements for spacing or a purpose not related to visual content. In this case, you should set the contentDescription, but with a value of null.

## Accessibility in Buttons

Buttons are an important part of any mobile application. I have often heard people complain that it is hard to use their phones due to “fat fingers”. For this reason, and to help users with disabilities, it is important to consider the **target size**of a button.

Android recommends a minimum of 48dp x 48dp for any button you are using. This can be achieved not only by the body of the button itself but through padding around the button. For example, you can have a button with a width of 32dp, and padding of 8dp to each side.

<Button   
 android:layout\_width ="wrap\_content"   
 android:layout\_height ="wrap\_content"   
 android:minWidth ="32dp"   
 android:paddingRight="8dp"   
 android:paddingLeft="8dp"   
 ... />

## Accessibility in Edit Text

**1. Focusable elements**

<EditText   
 android:id="@+id/accessibility\_edit\_text"   
 android:inputType="textPassword"   
 android:layout\_height="wrap\_content"   
 android:layout\_width="wrap\_content"   
 android:focusable="true"> </EditText>

In the above code, the way of adding accessibility is through the last property. When you use your phone on a daily basis, the normal way of accessing an edit text is through touch, which is automatically handled. By adding the focusable you are telling accessibility devices that this is an editable item and if navigating with a keyboard or another device, the user can interact (focus) on it.

**2. Hints** It is also recommended to add hints in texts that are editable. These hints should have examples of valid input so that users can follow along.

**3. Associate each EditText with a Label**

I have seen many apps, even the ones I developed, that use the hint of an EditText as the label for it. The problem with this is that when the user inputs the text she no longer knows what the field was for.

It is encouraged to always pair up each EditText with a Label. Android studio suggests this every time you are creating a layout, it is marked as a warning, not an error. To clarify the relationship you should use the **android:labelFor** property.

<TextView   
 android:id="@+id/thisLabelDescribesTheEditText"   
 android:labelFor="@+id/thisIsTheEditText" .../> <EditText   
 android:id="@+id/thisIsTheEditText" ... />

Another important thing to remember is that the text of each label should be unique, otherwise, the user will not know to which one are you referring. For example, for a user with a screen reader, the position of the elements visually doesn’t give a hint and they will not know the difference between elements with the same name.

# **Accessible Layouts**

Similar to EditTexts that are focusable through input methods, view groups can also be focused. For screen readers, it is good to know which elements are grouped, that way when reading content, they can mention all the elements inside a layout as part of the same group. You can define this with the code below.

<LinearLayout   
 android:id="@+id/this\_is\_a\_container"   
 android:screenReaderFocusable="true" ...>

# **Wrapping Up**

These are the most important lines of code for Accessibility in Android. As you may see, it is not a lot of code to add, but have you considered the big difference it could make for some users? I hope you add this to your next project!

**REFERENCE :**

**https://medium.com/swlh/accessibility-in-android-with-material-design-code-in-mobile-d2d1df54a0b6**