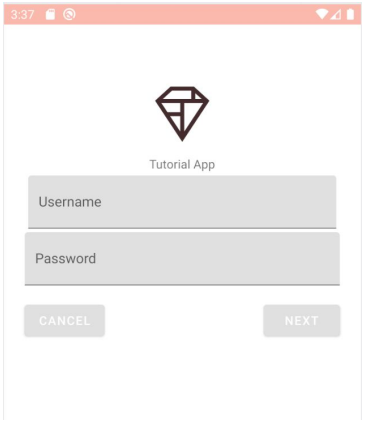
Material Design for Android

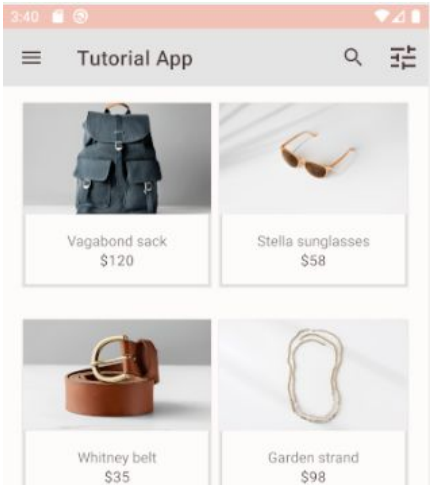
In this tutorial, we will showcase how Material Design Components and Principles can be used to create not only beautiful apps but also great experiences. This tutorial will not focus on the specifics of the underlying implementation but only on the code changes needed to achieve the desired design outcomes.
This tutorial is based on the documentation provided over at https://material.io/develop/android and will be a guideline to the webinar performed by Diogo Amores, Francisco Ferreira and Maria Inês Roseiro.

In order to start, download the code over at the link below and open it as an existing project in Android Studio.
https://github.com/manofrancisco/TutorialCM

First, we will take a look at the app. Running the App the first time will show a login screen like this:



You can login by just placing 3 letters in the password input and clicking next. That should show you this:



At this point, you should look at the code and explore.

This tutorial will work on the basis of replacing parts of the code that are commented. This will allow us to compare the differences between the parts of code and what each change does. Each part of this tutorial will have an ID and the files that need to be changed. Searching for the ID on the files will show you the relevant parts for that section of the tutorial.

App Theme

A1 - styles.xml

Go to the *styles.xml* file and look at the applied style. In this section, you do not need to change anything in the code. You can see that in this part we are just setting the colors defined which will be used later. You should also pay attention to the parent field. This means that it is extending or changing the parent style.
The theme that is used is defined in the *AndroidManifest.xml* file in the property **android:theme="@style/Theme.Tutorial"**.
This theme looks like this:

```
<style name="Theme.Tutorial" parent="Theme.MaterialComponents.Light.NoActionBar">
  <item name="colorPrimary">@color/colorPrimary</item>
  <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
  <item name="colorAccent">@color/colorAccent</item>
  <item name="android:textColorPrimary">@color/textColorPrimary</item>
</style>
```

Login Screen

In this section, we will show how we can make the login page prettier and also make it so it follows more of the Material Design principles. In this part, we will look at the layout file for this page, login_fragment.xml

App Name

L1 - login_fragment.xml

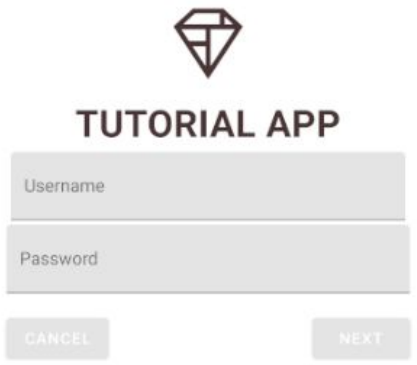
To observe the changes in this section we must go to the layout file and search for the ID or look for a TextView. Now you uncomment the commented part and delete the part that was not commented previously. You should look at the differences between what was removed and what was added.

added	*removed*
android:textAppearance="@style/TextAppearance.Shrine.Title"	-

In this part, we only changed the text appearance. If we go to the styles.xml file we can see that the defined text style looks like this:

```
<style name="TextAppearance.Shrine.Title"
parent="TextAppearance.MaterialComponents.Headline4">
  <item name="textAllCaps">true</item>
  <item name="android:textStyle">bold</item>
  <item name="android:textColor">?android:attr/textColorPrimary</item>
</style>
```

This changes the text to all caps, makes the text bold, and sets the color of the text. The login screen should now look like this:



“Next” Button

L2 - login_fragment.xml

Currently, the next button should look like this:

In order to make it more noticeable, we go to the layout file and find the part for this button. After uncommenting we can compare the differences:

added	*removed*
style="@style/Widget.Shrine.Button"	-

Once again only a style property was set. This style, defined on styles.xml, looks like this:

```
<style name="Widget.Shrine.Button"
parent="Widget.MaterialComponents.Button">
  <item name="android:textColor">?android:attr/textColorPrimary</item>
  <item
name="shapeAppearanceOverlay">@style/ButtonAppearance</item>
</style>
```

You should notice that the shape of the button is defined by calling a different style, that is defined as this:

```
<style name="ButtonAppearance">
  <item name="cornerFamily">rounded</item>
  <item name="cornerFamilyTopRight">cut</item>
  <item name="cornerFamilyBottomRight">cut</item>
  <item name="cornerSizeTopRight">16dp</item>
  <item name="cornerSizeBottomRight">16dp</item>
  <item name="cornerSizeTopLeft">4dp</item>
  <item name="cornerSizeBottomLeft">4dp</item>
</style>
```

The next button should now look like this:

“Cancel” Button

L3 - login_fragment.xml

In this section, there are bigger differences between what was previously set and what should now be set.

added	*removed*
android:id="@+id/cancel_button" style="@style/Widget.Shrine.Button.TextButton" *android:layout_toStartOf="@id/next_button"* *android:layout_toLeftOf="@id/next_button"*	-

In this part it sets the button position relative to the “Next Button” and also defines its style.

The style looks like this:

```
<style name="Widget.Shrine.Button.TextButton"
parent="Widget.MaterialComponents.Button.TextButton">
  <item
name="android:textColor">?android:attr/textColorPrimary</item>
</style>
```

This style on defines the Text Color, however it extends TextButton which is a different type of button defined by Material Design. This is because the Cancel button is

a secondary button, therefore should be a TextButton, so that the “Next Button” is more prominent.

The “Cancel” Button should look like this:

Username and Password Text Inputs

L4 - login_fragment.xml

In this section the text inputs will be modified, once again look in the login layout file and uncomment the relevant sections.

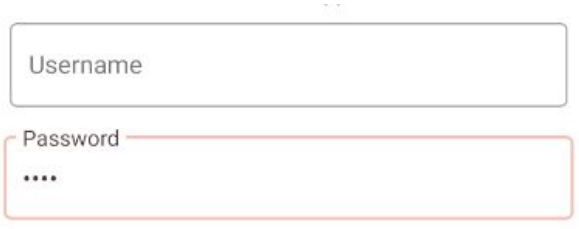
added	*removed*
style="@style/Widget.Shrine.TextInputLayout"	-

Once again only a style is added, the styling looks like this:

```
<style name="Widget.Shrine.TextInputLayout"
parent="Widget.MaterialComponents.TextInputLayout.OutlinedBox">
  <item
name="hintTextAppearance">@style/TextAppearance.Shrine.TextInputLayout.HintText</item>
  <item name="hintTextColor">@color/textColorPrimary</item>
  <item name="android:paddingBottom">8dp</item>
  <item name="boxStrokeColor">@color/textInputOutlineColor</item>
</style>

<style name="TextAppearance.Shrine.TextInputLayout.HintText"
parent="TextAppearance.MaterialComponents.Subtitle2">
  <item name="android:textColor">?android:attr/textColorPrimary</item>
</style>
```

This style sets the text color, hint color, and the color of the outline of the text box. It should now look like this:



Products Screen

In this section, we will use several layout files since this page is composed of different elements:

1. The toolbar
2. The Product Grid
3. The Product Card
4. The Menu Button and its view

Toolbar

P1 - product_grid_fragment.xml

In the toolbar we go to the product grid layout and perform the changes.

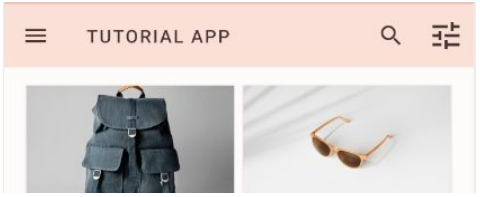
added	*removed*
style="@style/Widget.Shrine.Toolbar"	-

Once again we only specify a style, which looks like this:

```
<style name="Widget.Shrine.Toolbar" parent="Widget.AppCompat.Toolbar">
  <item name="android:background">?attr/colorAccent</item>
  <item name="android:theme">@style/Theme.Tutorial</item>
  <item name="popupTheme">@style/ThemeOverlay.AppCompat.Light</item>
  <item name="titleTextAppearance">@style/TextAppearance.Shrine.Toolbar</item>
</style>

<style name="TextAppearance.Shrine.Toolbar"
parent="TextAppearance.MaterialComponents.Button">
  <item name="android:textSize">16sp</item>
</style>
```

This style specifies which theme to use, which is the Base App Theme that we defined previously and looked at at the beginning.
The page should now look like this:



Product Grid

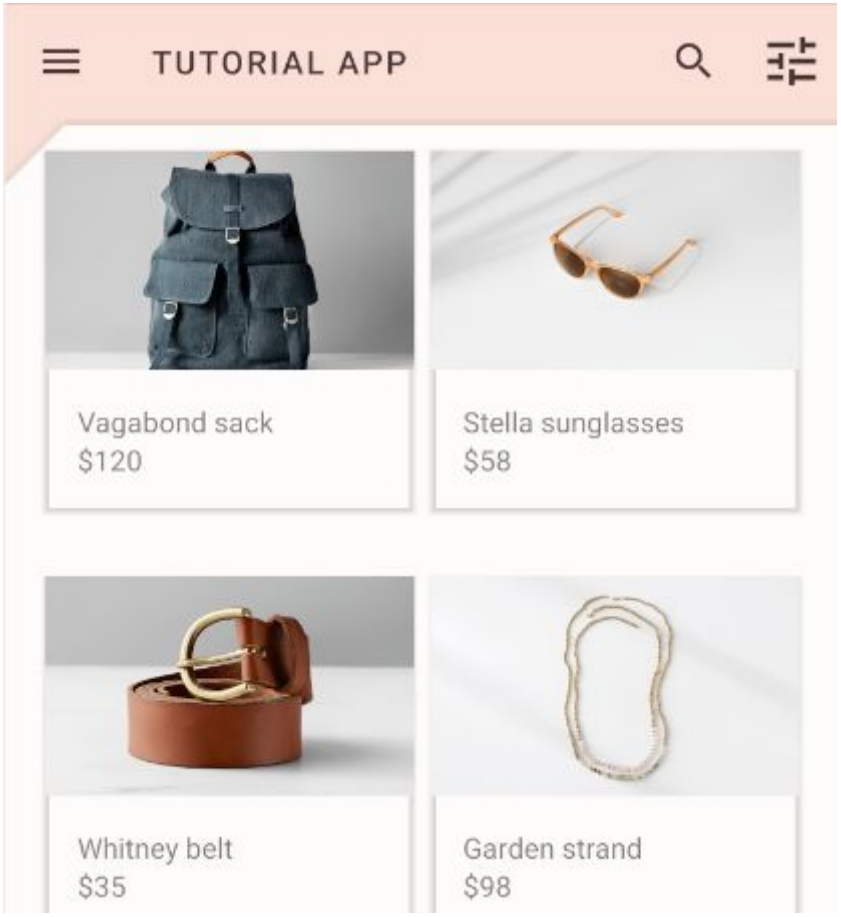
P2 - product_grid_fragment.xml,ProductGridFragment.java

In this part we will do something new, we will add a custom shape. In order to imprint a shape like this we head to the java file, and in the function onCreateView we will draw the shape of the Product Grid. The shape is defined at *shr_product_grid_background_shape.xml.*

After that we will still add some styling to change the elevation of the grid, so that the shape draw has a bigger shadow:

added	***removed***
android:elevation="8dp"	-

The app should now look like this:



Product Card Text

P3 - product_card.xml

In this section we change the product card text values, both in the product title and in the product price.
For the product title we have the following changes:

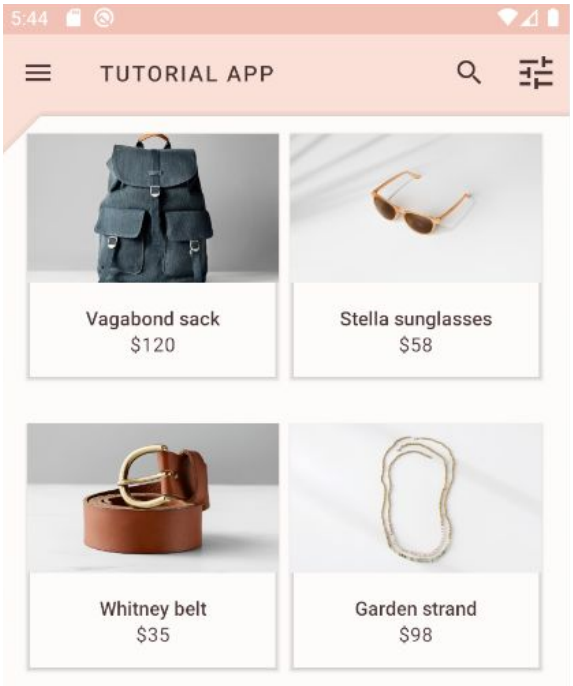
added	*removed*
android:textAlignment="center" android:textAppearance="?attr/textAppearanceSubtitle2" android:gravity="center_horizontal"	-

It centers the text and changes the text appearance.
For the product price the changes are almost similar:

added	*removed*
android:textAlignment="center" android:textAppearance="?attr/textAppearanceBody2" android:gravity="center_horizontal"	-

The only difference is the text appearance that is defined.

The cards should now look like this:



Product Card Layout

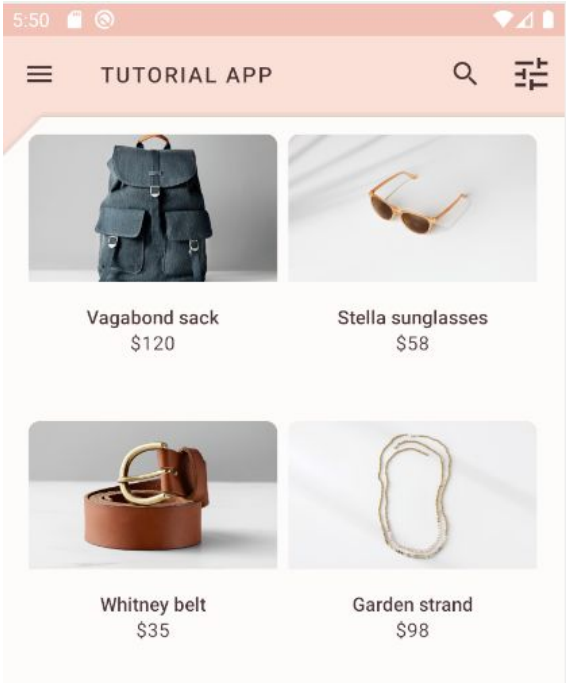
P4 - product_card.xml

In this section we edit the look of the product card layout. After uncommenting the following changes were made:

added	*removed*
app:cardElevation="0dp" app:cardCornerRadius="8dp"	app:cardElevation="2dp" app:cardCornerRadius="0dp"

This changes the elevation of the cards in comparison to the grid and makes the corners round.

At this point the app should look like this:



Menu Button

P5 - ProductGridFragment.java

In this section, we just uncomment the code on the java file and we can observe the animation.