Queens University Belfast

Android Studio

CSC3054 / CSC7054

Calculator App Tutorial

Building the App's GUI

The `Tip Calculator App` calculates and displays possible tips for a restaurant bill. As you enter each digit of a bill amount by touching the numeric keypad, the app calculates and displays the tip amount and total bill (bill amount + tip) for a 15% tip and a custom tip percentage (18% by default). You can specify a custom tip percentage from 0%-30% by moving the `SeekBar` *thumb* – this updates the custom percentage shown and displays the custom tip and total. The keypad may differ based on your AVD's or device's Android version, or based on whether you've installed and selected a custom keyboard on your device.

`GridLayout` Introduction

This app uses `GridLayout` to arrange views into five *rows* and two *columns*. Each cell in a `GridLayout` can be empty or can hold one or more views, including layouts that contain other views. Views can span multiple roes or columns. You can specify a `GridLayout's` number of rows and columns in the `activity_main.xml`.

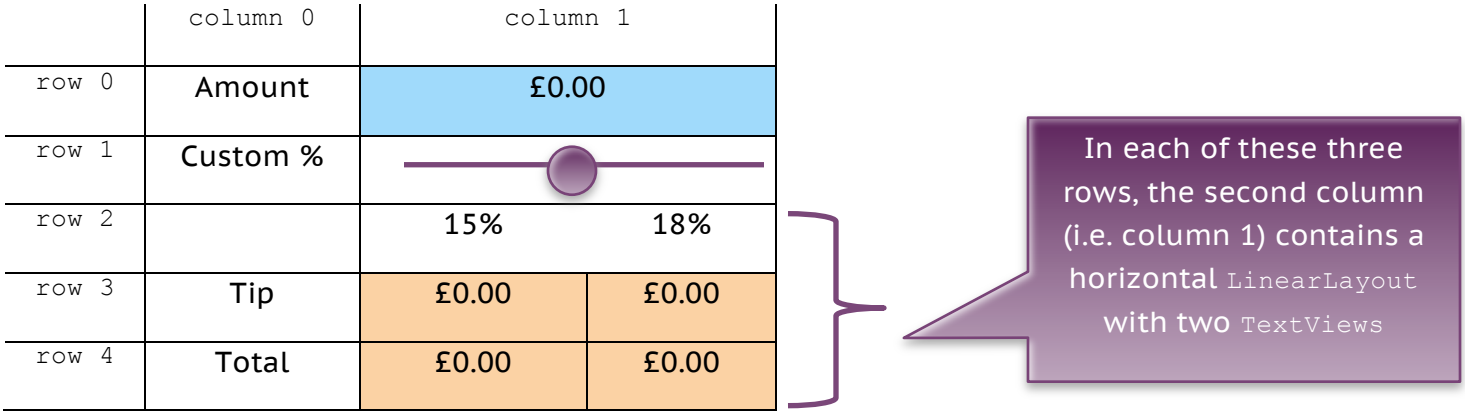

FIGURE 1 TIP CALCULATORS GUI'S GRIDLAYOUT LABELED BY ITS ROWS AND COLUMNS

Each row's height is determined by the tallest view in that row. Similarly, the width of a column is defined by the widest view in that column. By default, views are added to a row from left to right. You can specify the exact row and column in which the view is to be placed.

Figure 2 shows the views' `Id` attribute values. For clarity, the naming convention is to use the view's class name in the view's `Id` attribute and Java variable name.

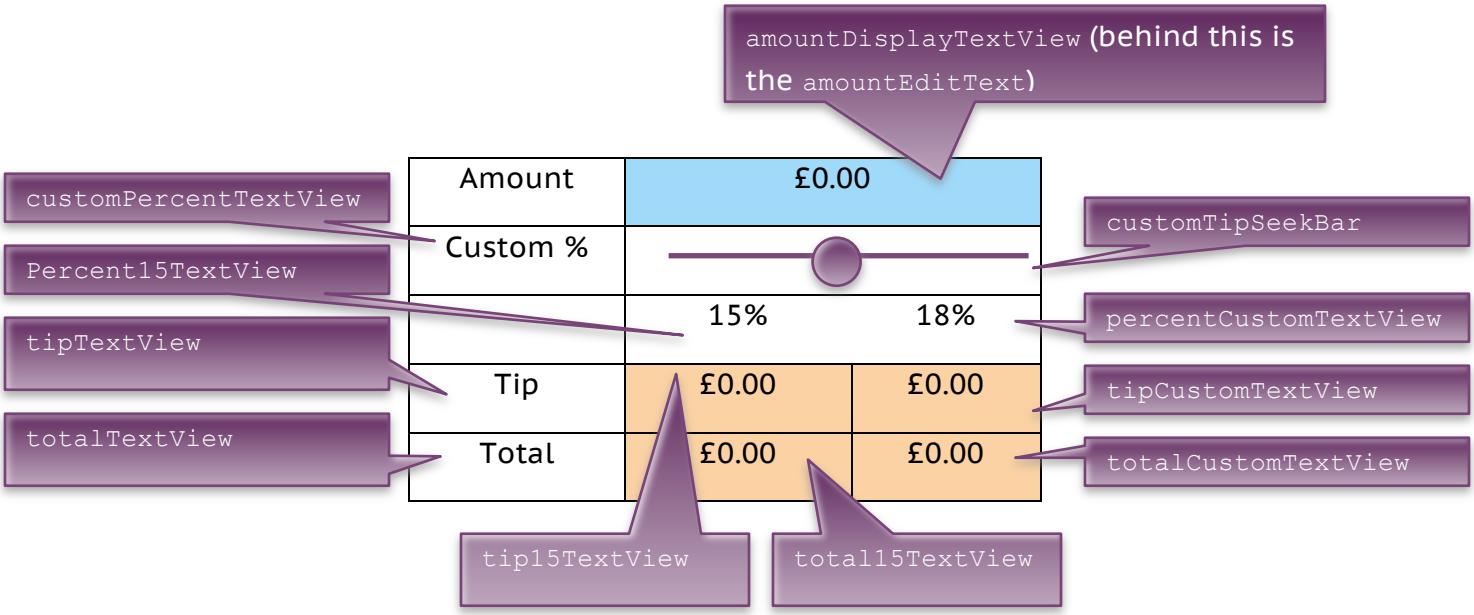

FIGURE 2 - TIP CALCULATOR GUI'S COMPONENTS WITH THEIR ID ATTRIBUTE VALUES

In the right column of the first row there are actually two components in the same grid cell - the `amountDisplayTextView` is hiding the `amountEditText` that receives the user input. The user's input is restricted to integer digits so that the user cannot enter invalid input. However the user will see the bill amount as a currency value.

Figure 3 shows the Ids of the three horizontal LinearLayouts in the GridLayout's right column.

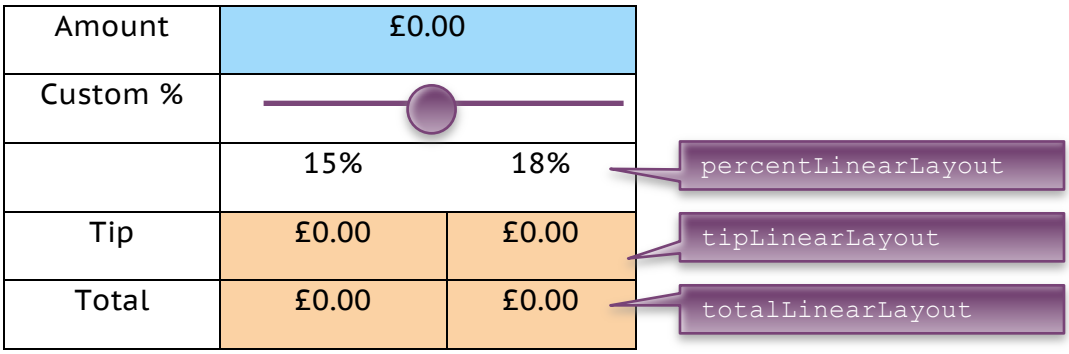

FIGURE 3 TIP CALCULATOR GUI'S LINEARLAYOUTS WITH THEIR ID PROPERTY VALUES

EXERCISE 1 – Creating the `TipCalculator` Project

Step 1: Create a New Android Project in Android Studio

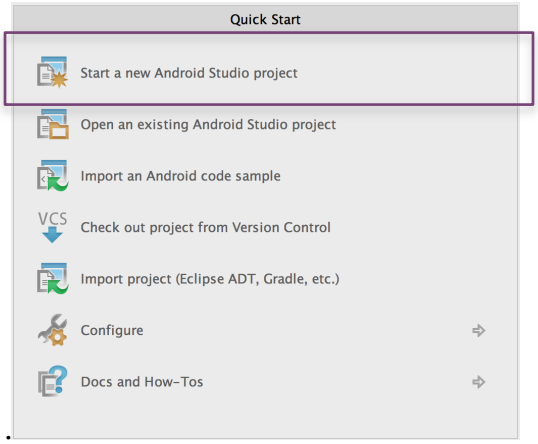

FIGURE 4 QUICK START OPTIONS

Step 2: Call the application `TipCalculator` and store it is an appropriate location.

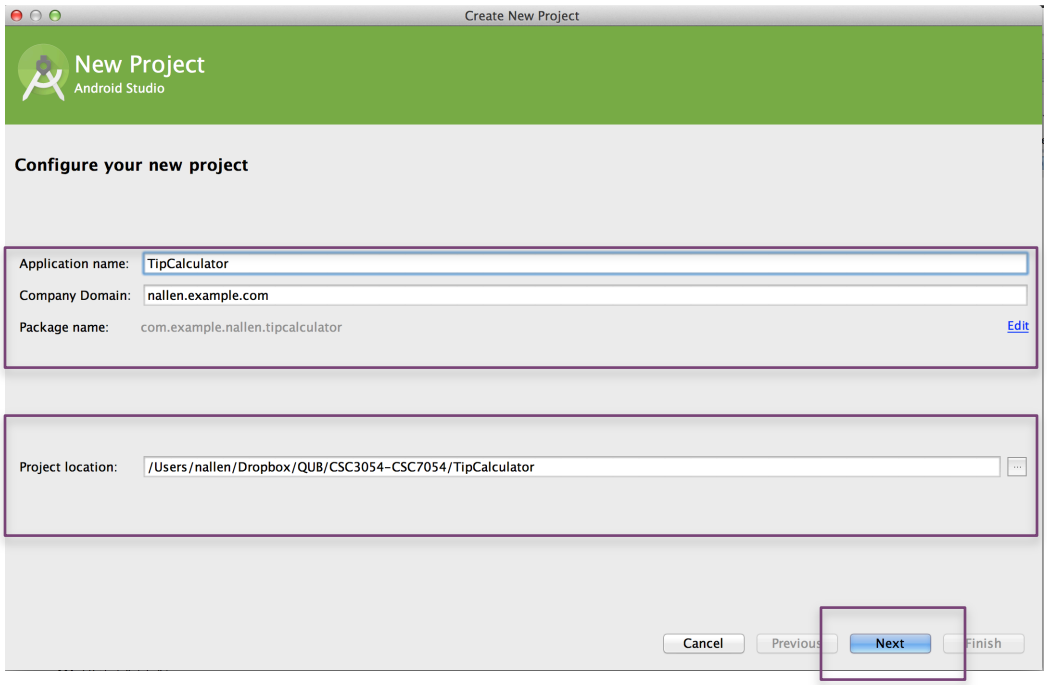

FIGURE 5- NEW PROJECT DIALOGUE

Step 3: Accept the default values as specified and press `Next`.

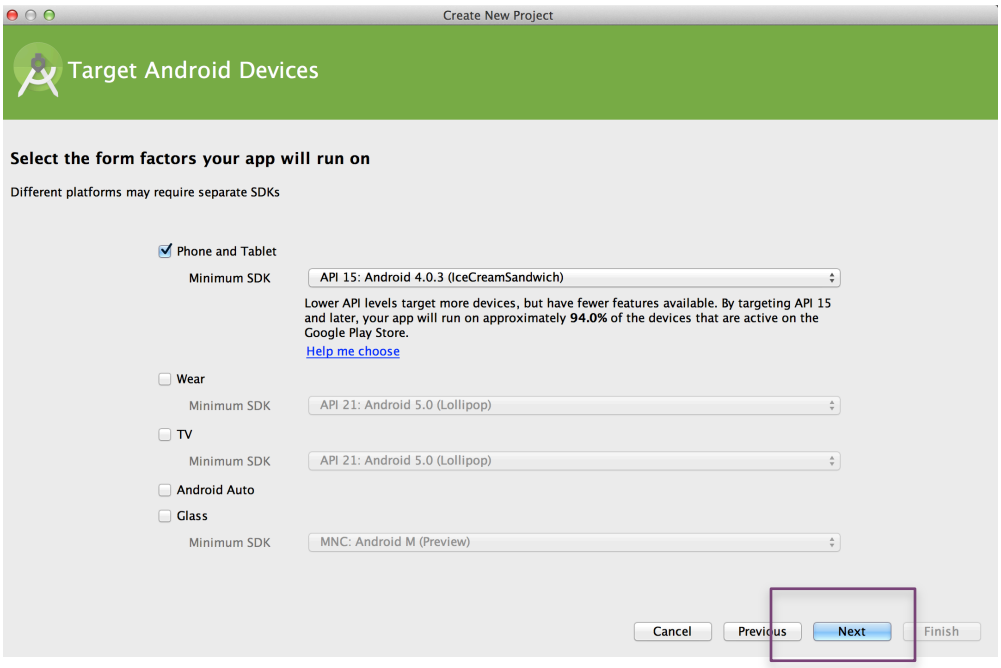

FIGURE 6 - TARGET ANDROID DEVICES DIALOGUE

Step 4: Choose a `Blank Activity`

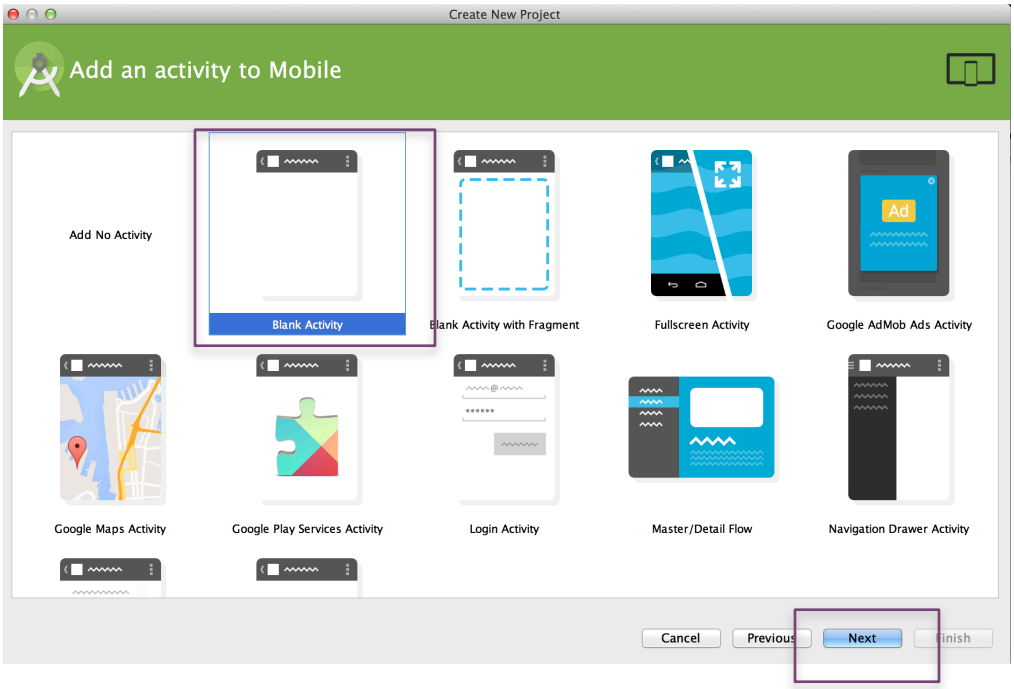

FIGURE 7- ADD AN ACTIVITY TO MOBILE DIALOGUE

Step 5: Customize the Activity and press `Finish`.

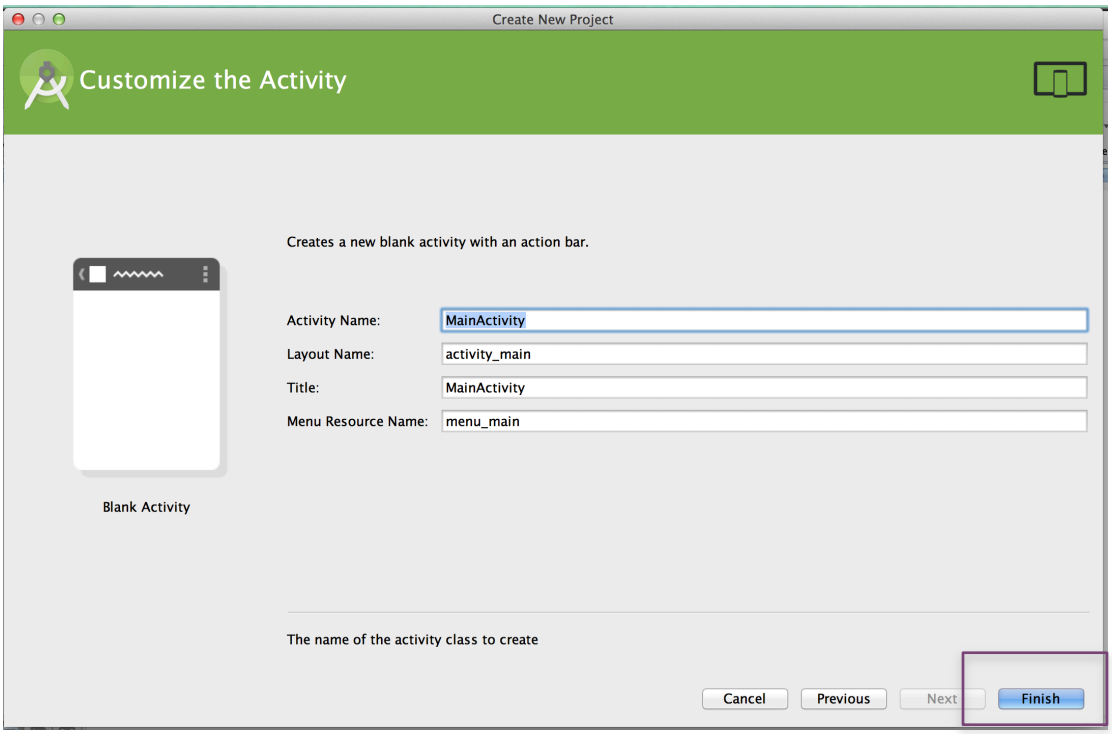

FIGURE 8 - CUSTOMIZE THE ACTIVITY DIALOGUE

Step 6: Your project should open up in Android Studio as shown in Figure 9.

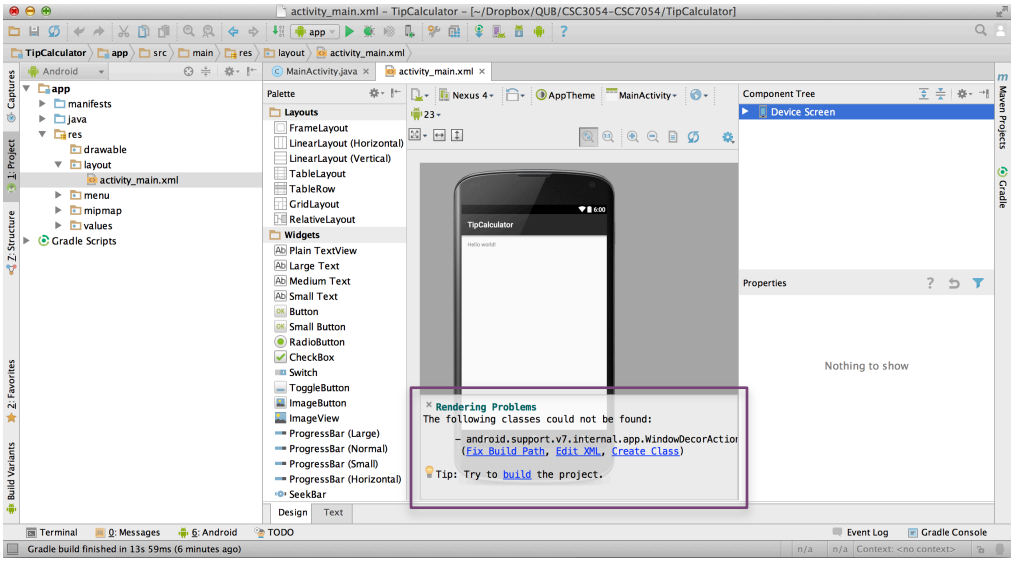

FIGURE 9 - RENDERING ISSUE

You may note that an error message comes up stating that there was a rendering problem. Ignore this error at the moment it will be discussed in more detail at a later stage in the tutorials.

EXERCISE 2 - Changing to a `GridLayout`

The default layout out in a Blank App's GUI is `relativeLayout`, however this app will use `GridLayout`. First, go to the XML view of `main_activity.xml` by clicking on the Text Tab at the bottom of the window as shown in Figure 11.

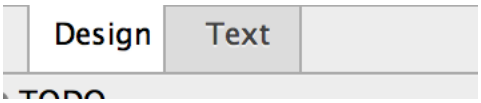

FIGURE 10 - SWITCHING TO THE XML

The `main_activity.xml` will be displayed. Change the layout from `RelativeLayout` to `GridLayout` and delete the auto generated `TextView` as shown in Figures 12 & 13.

FIGURE 11 - RELATIVELAYOUT

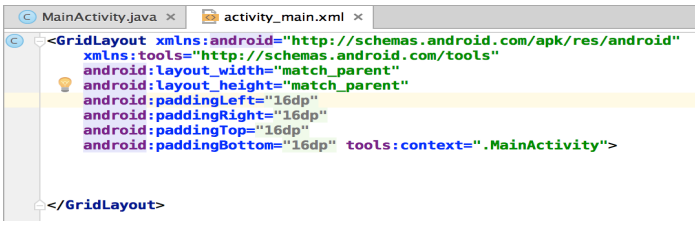

FIGURE 12 - GRIDLAYOUT

As there are two columns in this app, set the `columnCount` to `2`. By default, there are no margins – spaces that separate views – around a `GridLayout's` cells. Set the `GridLayout's useDefaultMargin` property to true to indicate that the `GridLayout` should place margins around its cells. By default, the `GridLayout` uses the recommended gap between views (`8dp`).

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:columnCount="2"
    android:useDefaultMargins="true">
</GridLayout>
```

EXERCISE 3 – Adding the `TextViews`, `EditText`, `SeekBar` and `LinearLayouts`

The design for the GUI is represented in Figure 14. In order for this GUI to be designed a total number of 16 views needs to be added to the canvas as shown in Figure 15.

Amount	£0.00	
Custom %		
	15%	18%
Tip	£0.00	£0.00
Total	£0.00	£0.00

FIGURE 13 - TIP CALCULATOR GUI DESIGN

	Column 0	**Column 1**
Row 0	`amountTextView`	`amountDisplayTextView` & `amountEditText`
Row 1	`customePercenttextView`	`customTipSeekBar`
Row 2		`LinearLayout` (Horizontal) `percent15TextView` | `percentCustomTextView`
Row 3	`tiptextView`	`LinearLayout` (Horizontal) `tip15TextView` | `tipCustomTextView`
Row 4	`totalTextView`	`LinearLayout`(Horizontal) `total15TextView` | `totalCustomTextView`

FIGURE 14 - PLACEMENT OF VIEWS

As a view is added to the XML, immediately set it's `Id` attribute to the `Id` that is specified in Figure 15.

Step1: Adding Views to the First Row

The first row (row 0) consists of the `amountTextVew` in the first column (column 0). The `amountEditText` is behind the `amountDisplayTextView` in the second column (column 1). Each time you drop a view or layout onto the `GridLayout`, the view is placed in the layout's next open cell, unless you specify otherwise by setting the view's `Row` and `Column` attributes.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/amountTextView" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/amountEditText" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/amountDisplayTextView"
    android:layout_row="0"
    android:layout_column="1" />
```

Please note: When the second `TextView` is added to this row, it is initially placed in the first column of the `GridLayout's` second row. To place it in the second column of the `GridLayout's` first row, set the `amountDisplayTextView` `Row` and `Column` attributes to the values `0` and `1` respectively.

Step 2: Adding Views to the Second Row

This row has a `TextView` and a `SeekBar`. Add the following code to you XML.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:id="@+id/customPercentTextView"
    android:text="Medium Text"
  />

<SeekBar
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/customeTipSeekBar" />
```

Step 3: Adding Views to the Third Row

This row has a `LinearLayout` set to horizontal along with two `TextViews`. Add the following code to your XML.

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/percentLinearLayout"
    android:layout_column="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:id="@+id/percent15TextView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:id="@+id/percentCustomTextView" />
</LinearLayout>
```

Step 4: Adding Views to the Fourth Row

This row has a `TextView` and `LinearLayout` set to horizontal along with two `TextViews`. Add the following code to your XML.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/tipTextView" />

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tipLinearLayout"
    android:layout_column="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:id="@+id/tip15TextView" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:id="@+id/tipCustomTextView" />

</LinearLayout>
```

Step 5: Adding Views to the Fifth Row

This step is the same as the step above. This row has the following views:

- `TextView` (called `totalTextView`)
- `LinearLayout` (called `totalLinearLayout`) set to horizontal
- The `LinearLayout` has two `TextViews (total15TextView & totalCustomTextView)`.

What your App should look like

Your app should now look similar to Figure 16.

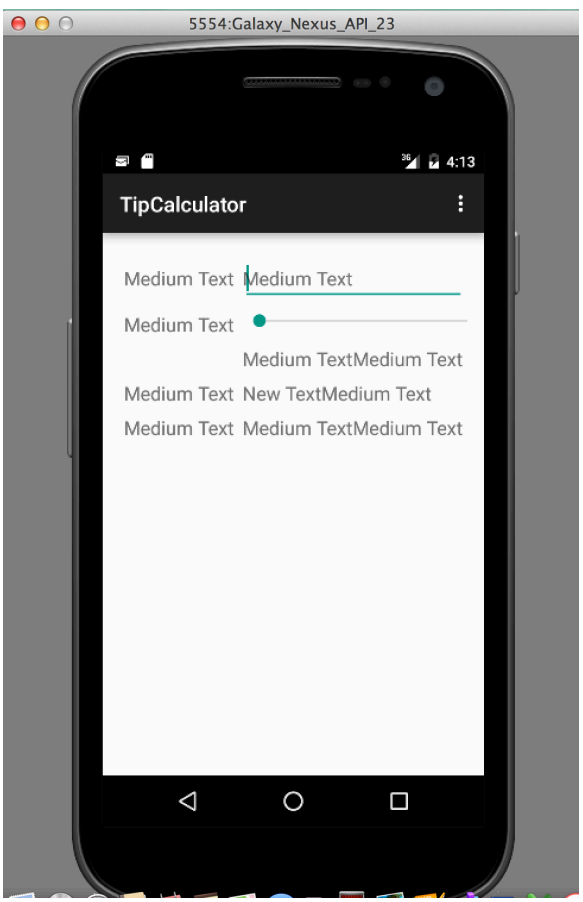

FIGURE 15 - VIEW COMPONENTS

EXERCISE 4 – Customizing the Views to Complete the Design

In this exercise each of the views that you added in the previous section will be customized to complete the design for the app. Several string and Dimension resources will also be added. Literal string values should be placed in the `strings.xml` file. Similarly, literal numeric values that specify view dimensions (e.g. widths, heights and spacing) should be placed in the `dimens.xml` resource file.

Step 1: Specifying Literal Text

Specify the literal text for the `amountTextView`, `customPecentTextView`, `percent15TextView`, `percentCustomTextView`, `tipTextView` and `totalTextView`.

Select the `strings.xml` file from the project explorer. Double click it and the `strings.xml` file should open up as shown in Figure 17.

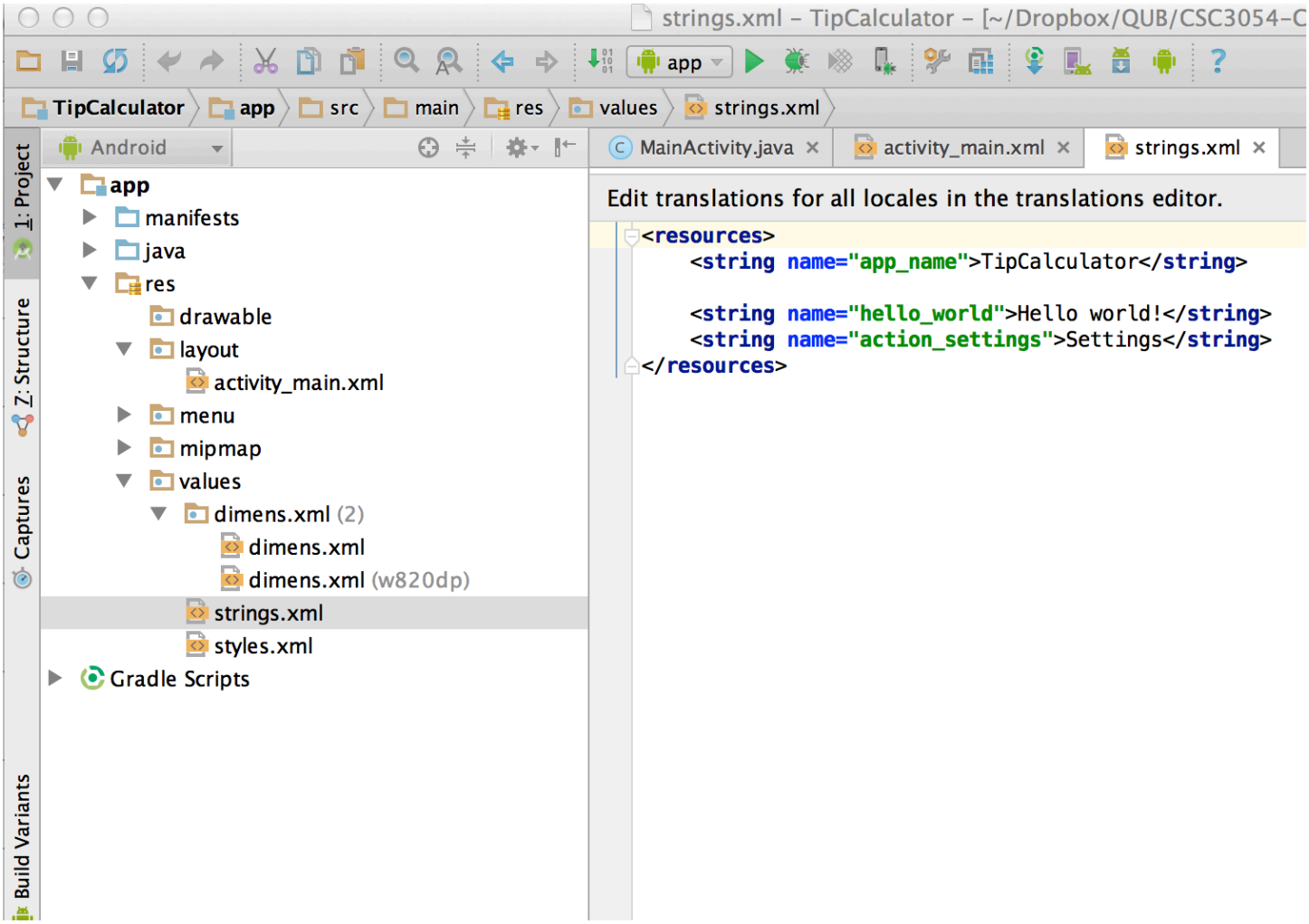

FIGURE 16 - OPENING THE STRING.XML FILE

In the `strings.xml` add the following XML code:

```
<string name="custom_tip_percentage">Custom % </string>
```

Once added, repeat this step using the values shown in Figure 18.

String name =	Actual String
amount	Amount
fifteen_percent	15%
eighteen_percent	18%
tip	Tip
total	Total

FIGURE 17 - STRING VALUES

Step 2: Specifying Literal Dimensions

Select the `dimens.xml` file from the project explorer. Double click it and the `dimens.xml` file should open up as shown in Figure 19.

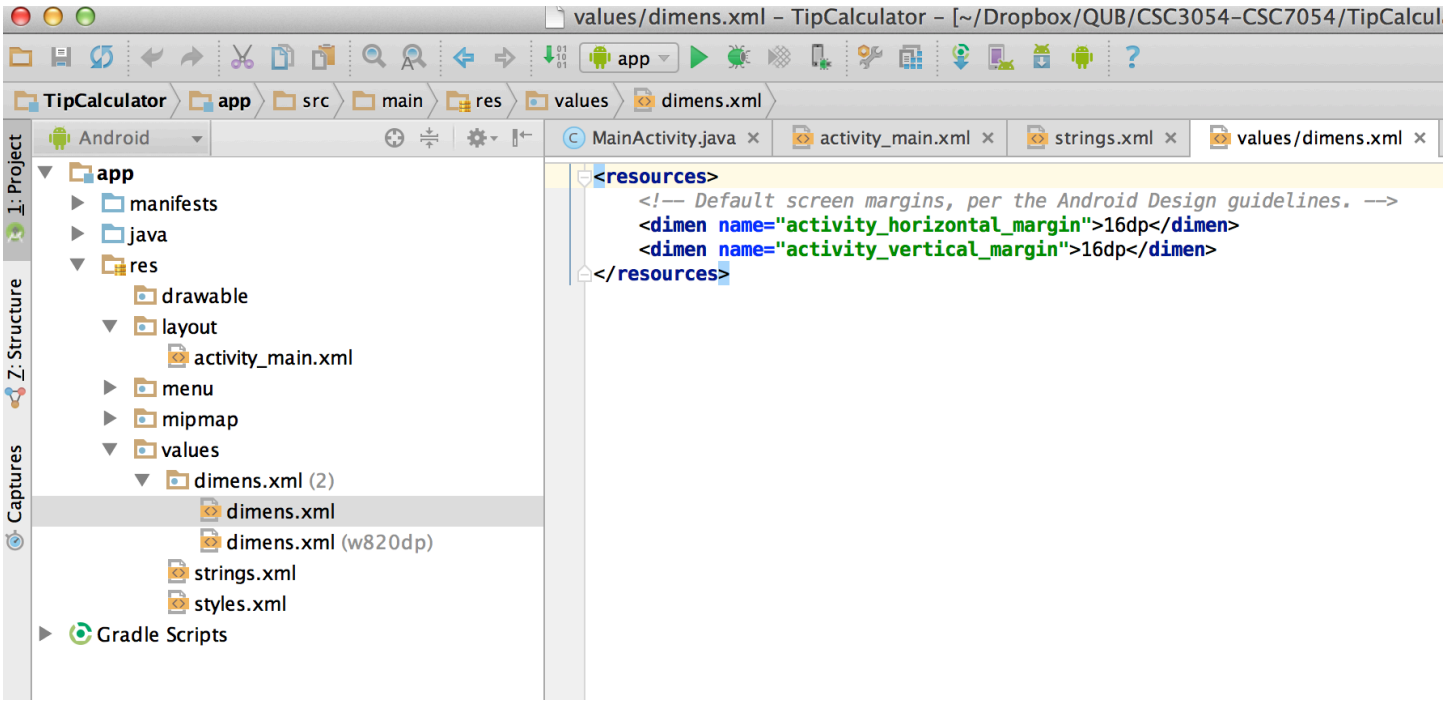

FIGURE 18 - DIMENSION.XML FILE

In the `dimens.xml` add the following XML code:

```
    <dimen name="activity_horizontal_margin">16dp</dimen>
```

Once added, repeat this step using the values shown in Figure 20.

dimen name =	dp
activity_vertical_margin	16dp
textview_padding	8dp
textview_margin	8dp

FIGURE 19 - DIMENSION VALUES

Step 3: Right Aligning the `TextVews` in the Left Column

In this app, each of the left columns `TextViews` are right aligned for the `amountTextView`, `customPercentTextView`, `tipTextView` and `totalTextView` add the following XML code:

```
android:gravity="right"
```

Step 4: Configuring the `amountEditText`

In the final app, the `amountEditText` is hidden behind the `amountDisplayTextView` and is configured to allow only digits to be entered by the user. Add the following XML code to the `amountEditText`:

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:id="@+id/amountEditText"
    android:layout_row="0"
    android:layout_column="1"
    android:digits="0123456789"
    android:maxLength="6"
     />
```

Notes on the attributes:

Attribute	Description
`android:layout_width="wrap_content"` `android:layout_height="wrap_content"`	This indicates that the `EditText` should be just large enough to fit its contents, including any padding
`android:digits="0123456789"`	This allows only digits to be entered, even though the numeric keypad contains minus (-), comma(,), period(.) and space buttons.
`android:maxLength="6"`	This restricts the bill amount to a maximum of six digits - so the largest supported bill amount is 9999.99.

Step 5: Configuring the `amountDisplayTextView`

To complete the formatting of the `amountDisplayTextView`, select it and set the following:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:id="@+id/amountDisplayTextView"
    android:layout_row="0"
    android:layout_column="1"
    android:layout_gravity="fill_horizontal"
    android:background="@android:color/holo_blue_bright"
    android:padding="@dimen/textview_padding" />
```

Notes on the attributes

Attribute	Description
`android:layout_gravity="fill_horizontal"`	This indicates that the `TextView` should occupy all remaining horizontal space in this `GridLayout` row.
`android:background="@android:color/holo_blue_bright"`	This is one of several predefined colours (each starts with `@android:color`) in Android's `Holo` theme
`android:padding="@dimen/textview_padding"`	This is the padding around the `TextView`. A View's padding property specifies space on all sides of the View's content.

Step 6: Configuring the `customPercentTextView`

Notice that the `customPercentTextView` is aligned with the top of the `customTipSeekBar's` thumb. This looks better if vertically centered. To do this, add the following to the XML code:

```
android:layout_gravity="center_vertical|right"
```

The vertical bar (`|`) character is used to separate multiple `Gravity` values – in this case indicating that the `TextView` should be right aligned and centered vertically within the grid cell.

Step 8: Configuring the `customTipSeekBar`

By default, a `SeekBar's` range is `0` to `100` and its current value is indicated by its `Progress` attribute. This app allows custom tip percentages from `0` to `30` and specifies a default of `18`. Set the `SeekBar's Max` attribute to `30` and the `Progress` attribute to `18`.

```
<SeekBar
    android:layout_width="257dp"
    android:layout_height="wrap_content"
    android:id="@+id/customTipSeekBar"
    android:layout_row="1"
    android:layout_column="1"
    android:max="100"
    android:progress="18" />
```

Step 9: Configuring the `percent15TextView` and `percentCustomTextView`

Add the following code to the `activity_main.xml` file:

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:gravity="fill_horizontal"
    android:layout_height="wrap_content"
    android:layout_row="2"
    android:layout_column="1"
    android:id="@+id/percentLinearLayout"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/fifteen_percent"
        android:id="@+id/percent15TextView"
        android:layout_weight="0.3"
        android:gravity="center"
         />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/eighteen_percent"
        android:id="@+id/percentCustomTextView"
        android:layout_weight="0.3"
        android:gravity="center" />

</LinearLayout>
```

Recall that `GridLayout` does not allow you to specify how a view should be sized relative to other views in a given row. This is why the `percent15TextView` and `percentCustomTextView` in a `LinearLayout`, which does allow proportional sizing.

Notes on attributes

Attribute	Description
`android:layout_weight="0.3"`	A view's `layout _weight` (in certain layouts, such as `LinearLayout`) specifies the view's relative importance with respect to other views in the layout. By default, all views have a `Weight` of `0`. In this layout, Weight is set to `0.3` for `percent15TextView` and `percentCustomTextView` – this indicates that they have equal importance, so they should be sized equally.
`android:gravity="fill_horizontal"`	By default, when `percentLinearLayout` was added to the `GridLayout`, its layout `Gravity` property was set to `fill_horizontal`, so the layout occupied the remaining space in the third row.
`android:gravity="center"`	When the `LinearLayout` is stretched to fill the rest of the row, the `TextViews` each occupy half of the `LinearLayout's` width. Each `TextView` should `center` its text therefore the `Gravity` attribute should be set to center - this specifies the `TextView's` text alignment, whereas the `layout_gravity` property specifies how a view aligns with respect to the layout.s

Step 10: Configuring the `tip15TextView`, `tipCustomTextView`, `total15TextView` and `totalCustomTextView`

To finalize these four `TextViews`, update the XML code so that it includes the following:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="@string/tip"
    android:id="@+id/tipTextView"
    android:layout_row="3"
    android:layout_column="0"
    android:gravity="right" />

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="259dp"
    android:layout_height="wrap_content"
    android:layout_row="3"
    android:layout_column="1"
    android:id="@+id/tipLinearLayout">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/tip15TextView"
        android:layout_weight="1"
        android:padding="@dimen/textview_padding"
        android:gravity="center"
        android:layout_gravity="center_horizontal"
        android:password="false"
        android:layout_marginRight="@dimen/textview_margin"
        android:background="@android:color/holo_orange_light" />

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/tipCustomTextView"
        android:layout_weight="1"
        android:padding="@dimen/textview_padding"
        android:gravity="center"
        android:background="@android:color/holo_orange_light" />
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="@string/total"
    android:id="@+id/totalTextView"
    android:layout_row="4"
    android:layout_column="0"
    android:gravity="right"
    android:layout_gravity="center_vertical" />
```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="259dp"
    android:layout_height="wrap_content"
    android:layout_row="4"
    android:layout_column="1"
    android:id="@+id/linearLayout" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/total15TextView"
        android:layout_weight="1"
        android:padding="@dimen/textview_padding"
        android:gravity="center"
        android:layout_gravity="center_horizontal"
        android:password="false"
        android:layout_marginRight="@dimen/textview_margin"
        android:background="@android:color/holo_orange_light" />

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/totalCustomTextView"
        android:layout_weight="1"
        android:padding="@dimen/textview_padding"
        android:gravity="center"
        android:background="@android:color/holo_orange_light" />
</LinearLayout>
```

Notice that there's no horizontal space between the `TextViews` in the `tipLinearLayout` and `totalLinearLayout`. To fix this, an `8dp` right margin for the `tip15TextView` and `total15TextView` is specified.

Step 11: Vertically Centering the `tipTextView` and TotalTextView

To vertically center `tipTextView`, the `GridLayout` centers this component vertically in the remaining space from the fifth row to the bottom of the screen. To fix this problem, add a `Space` view into the XML.

```
<Space
    android:layout_width="20px"
    android:layout_height="20px"
    android:layout_row="5"
    android:layout_column="0" />
```

This creates a sixth row that occupies the rest of the screen. As its name implies a `Space` view occupies space in a GUI. The GUI should now look like Figure 21.

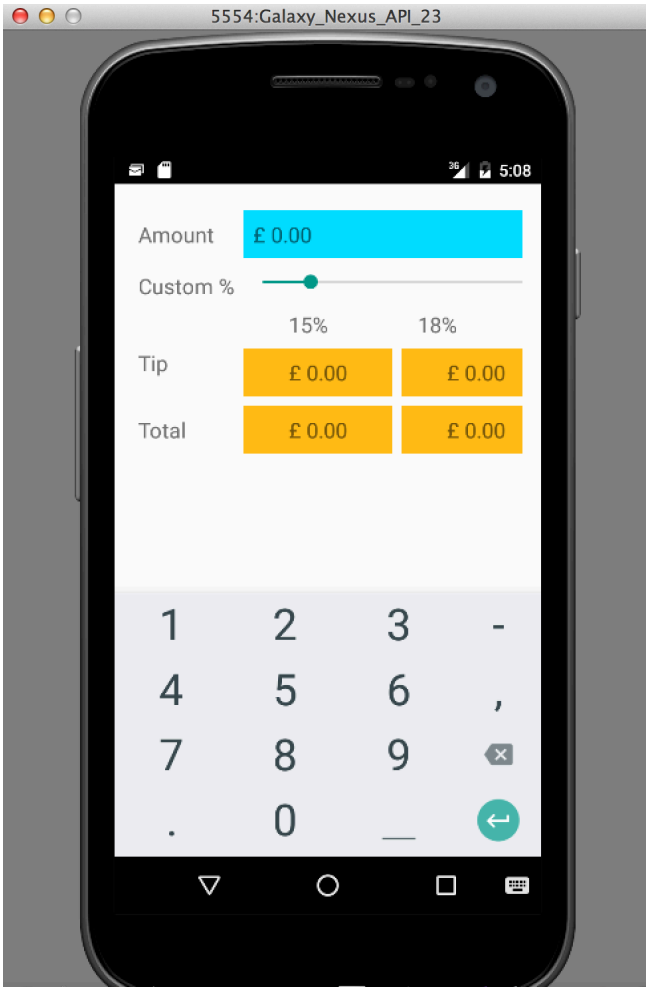

FIGURE 20 - FINISHED GUI