Mobile & Wearable Computing - assignment 01
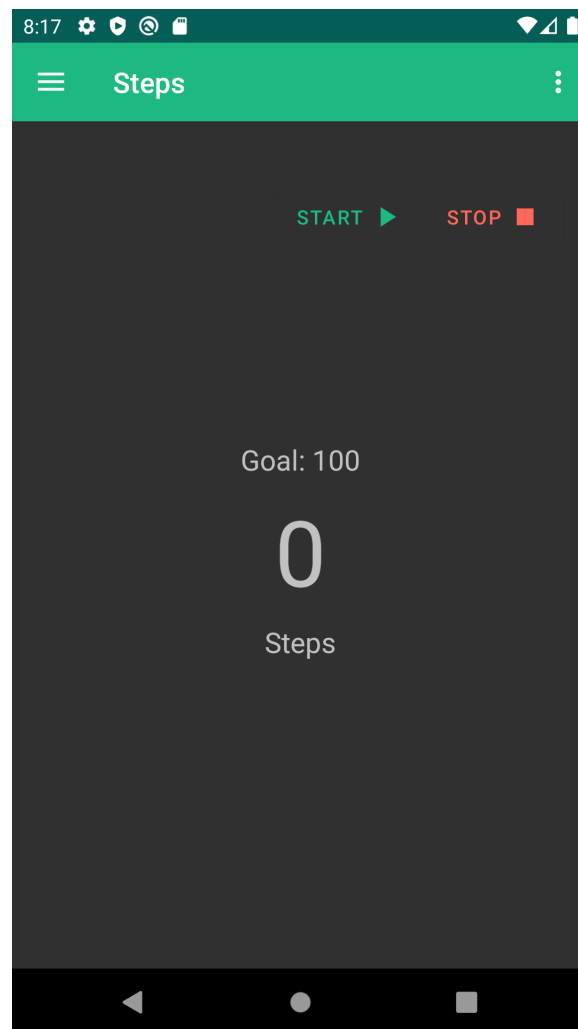
[GitHub Project](#)

Exercise 1 - Material Design

Screenshot of the home with the modified buttons

New app icon

To change the app icon I made a new Image Asset in res/drawable, replacing the old asset.
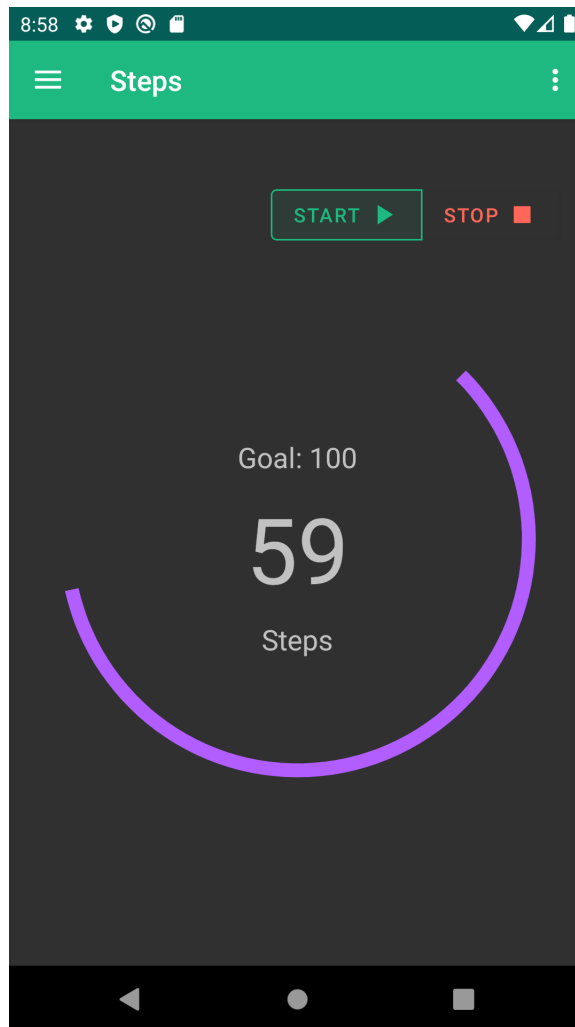

fragment_home.xml

```xml
<com.google.android.material.button.MaterialButton
    android:id="@+id/toggleStart"
    style="?attr/materialButtonOutlinedStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/toggle_start"
    android:textColor="@color/colorPrimary"
    // ic_button_start is a new xml file in res/drawable
    android:drawableRight="@drawable/ic_button_start"
    android:drawableTint="@color/colorPrimary"/>

<com.google.android.material.button.MaterialButton
    android:id="@+id/toggleStop"
    style="?attr/materialButtonOutlinedStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/toggle_stop"
    android:textColor="@color/colorStop"
    // ic_button_stop is a new xml file in res/drawable
    android:drawableRight="@drawable/ic_button_stop"
    android:drawableTint="@color/colorStop"/>
```

Exercise 2 - Step Counter

Updated circular progress bar

strings.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE resources [
  <!ENTITY goalValue "100">
]>


<resources>
  .....
  <string name="goal_value">&goalValue;</string>
  <string name="goal">Goal: &goalValue;</string>
  .....
</resources>
```

fragment_home.xml

I set the goal value as a new `ENTITY` in the string.xml file. I've done so to use the same value for the Goal string and to set the max value of the Progress Bar.

```xml
<ProgressBar
  // .....
  android:max="@string/goal_value"
  // The progress bar is rotated for a personal design
preference
  android:rotation="-45"
  // Style changed to make the progress bar determinate
  style="@style/Widget.AppCompat.ProgressBar.Horizontal"/
>
```

HomeFragment.java

```java
public class HomeFragment extends Fragment {
  // .....
  // variable to hold the step detector sensor
  private Sensor mSensorSTEP;
  // .....
  public View onCreateView(@NonNull LayoutInflater
inflater, ViewGroup container, Bundle savedInstanceState)
{
    // .....
```

```java
    // get the default step detector sensor
    mSensorSTEP =
mSensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR
);
    // .....
    @Override
    public void onButtonChecked(MaterialButtonToggleGroup
group, int checkedId, boolean isChecked) {
        if (group.getCheckedButtonId() == R.id.toggleStart)
{
            // .....
            // Check if the Step detector sensor exists
            if (mSensorSTEP != null) {
                mSensorManager.registerListener(listener,
mSensorSTEP, SensorManager.SENSOR_DELAY_NORMAL);
            }
            else {
                Toast.makeText(getContext(),
R.string.step_not_available, Toast.LENGTH_SHORT).show();
            }
}

class StepCounterListener implements SensorEventListener
{
  // .....
  // Step detector counter
  int mSTEPStepCounter = 0;
  // .....
  @Override
  public void onSensorChanged(SensorEvent event) {
    switch (event.sensor.getType()) {
        // .....
        case Sensor.TYPE_STEP_DETECTOR:
            countSteps(event.values[0]);
    }
  }
  // .....
  private void countSteps(float step) {
    mSTEPStepCounter += step;
    Log.d("STEP STEPS: ",
String.valueOf(mSTEPStepCounter));
    // updateView(mSTEPStepCounter);
  }
```

```java
    private void updateView(int stepCounter) {
        // update the text view

    stepsCountTextView.setText(String.valueOf(stepCounter));
        // update the ProgressBar
        stepsCountProgressBar.setProgress(stepCounter, true);
    }
}
```

I made a method to handle updating the view to avoid code repetition and be able to use either the ACC sensor or the STEP sensor.