MSCC-MD Repeat Assignment

Building an application that will produce telemetry traces of sensor output.

Assignment Information:

|  |  |
| --- | --- |
| **Course:** | MSCC |
| **Stage/Year:** | 1 |
| **Module:** | Mobile Development |
| **Semester:** | 1 |
| **Assignment:** | Repeat Assignment |
| **Date of Issue:** | TBC |
| **Assignment Deadline:** | See email informing you about repeat assignments |
| **Assignment Submission:** | See email informing you about repeat assignments |
| **Assignment Weighting:** | 50% |

Introduction:

**Note: read the whole assignment brief first before implementing it. The brief contains very important information.**

In this assignment you will be tasked with building an application capable of querying and showing the output of individual sensors on a telemetry trace over the last 5 seconds. A telemetry trace is simply a line graph and you will need to generate a custom view for this part. You will be required to display the data of the following sensors:

* scalar sensors: light sensor, proximity sensor
* vector sensors: gyroscope, linear acceleration

For the scalar sensors your custom view must draw a single trace. While for the vector sensors your custom view must draw three different coloured traces on the same graph (a trace for the X, Y, and Z values respectively). Note that for some of the sensors you will only have positive values but for others you will have both positive and negative values. Your custom view must also account for this by setting the zero line at the bottom of the graph (if positive only) or in the middle of the graph (if positive and negative)

For your UI you will need mechanisms to enable a user to switch between the sensors and also to start and stop telemetry traces. Your update rate for sensors should be set to the constant for UI delay.

**Note that you are not permitted to use libraries of any kind to produce the graphs. You must render the graphs yourself using mathematics and primitives.**

Submission and Penalties:

You are required to submit two separate components:

* A copy of your complete Android Studio project. The accepted archive formats are: zip, rar, 7z, tar.gz, tar.bz2, tar.xz. The use of any other archive format will incur a 10% penalty before grading.
* A PDF containing documentation of your code. **If you do not provide documentation your code will not be marked.** Copying and pasting code will not count as documentation.

There are also penalties you should be aware of:

* Code that fails to compile will incur a **30%** penalty before grading. At this stage you have zero excuse to produce non-compiling code. I should be able to compile and run without having to fix syntax errors.

**Please Note:** You should compile your code on a machine other than your own before

submission, to ensure that there are no errors.

* The use of libraries outside the SDK will incur a **20%** penalty before grading. You have all you need in the standard SDK. I shouldn't have to figure out how to install and use an external library to get your application to work.
* The standard late penalties will also apply

**Very important: Take note of the brackets listed below. These are meant to completed in order. If you skip a bracket or trigger one of the failing conditions the following brackets will not be considered for marking. You should be well capable of producing strong and generally robust software by now. For example, if there are six brackets and you fail the third one, then the fourth, fifth, and sixth milestones will not be marked. Documentation brackets will be treated independently of Coding brackets and are unaffected by bracket failures.**

Marks will also be removed for the presence of bugs anywhere in the code and this will incur a deduction between **1%** and **15%** depending on the severity. If you have enough of these bugs it is entirely possible that you may not score many marks overall despite answering all milestones. I want robust bug free code that also validates all user input to make sure it is sensible in nature.

Also note that the percentage listed after the bracket is the maximum mark you can obtain if you complete that many brackets without error.

Coding Brackets (70%):

|  |  |  |
| --- | --- | --- |
| **No.** | **%** | **Coding Brackets** |
| 1 | 10 | * Generate the shell of an application that contains a single activity with a single custom view * Add in controls for starting and stopping traces * Add in control for changing the sensor type |
| 2 | 20 | * Enable the controls for changing the sensor * Enable the controls for starting and stopping the traces * When traces are running keep the last 5 seconds worth of trace data in a datastructure of some description at all times. |
| 3 | 30 | * Generate a custom view for the single traces that has an upper limit of 600 lux for the light sensor and 20cm for the proximity sensor * You will fail this bracket if your drawing commands are not done relative to canvas.getWidth() and canvas.getHeight() |
| 4 | 40 | * Generate a custom view for the double traces that has a limit of 1G (positive or negative) for the linear accelerometer and a limit of 2 rads/s ) positive or negative for the gyroscope * You will fail this bracket if your drawing commands are not done relative to canvas.getWidth() and canvas.getHeight() |
| 5 | 50 | Modify the single trace to adapt the Y axis maximum value to the maximum value that is seen in the last 5 seconds of data |
| 6 | 60 | Modify the multi-trace to adapt the Y axis maximum and minimum to the absolute highest value seen in the last 5 seconds of data |
| 7 | 70 | Add in UI elements to display the current values of the trace your user is currently looking at |

Documentation Brackets (30%):

|  |  |  |
| --- | --- | --- |
| **No.** | **%** | **Documentation Brackets** |
| 1 | 15 | Document why you designed the UI the way you did. This should detail your choices in widget layout and position and how they make user interaction easier. Examples of what I am looking for are as follows:   * The button was placed at the bottom of the screen as it is a frequent action that the user's thumb must reach * The colour scheme that was chosen to avoid the main form of colour blindness and produce high contrast for the visually impaired.   You should also describe the mathematics used to draw your custom view. |
| 2 | 30 | Give a high-level description of every method in your Java code. You should also document the datastructures you have used and why they are used.  **Please Note:** that there should be no copying and pasting of code here. |