Task – Spike: Report 1

Student ID: 102567604
Name: Aaron Rickards

Goals:

The goal of this investigation was to investigate the effect that different libraries used to create graphs using data points or entries would have on the performance of the application. Other factors that were looked at was the capabilities, ease of use and is the library still able to be used correctly.

Tools and Resources Used

This section lists related software, tools, libraries, API's, and other resources used for this knowledge gap.

- Android Studio
- Assignment Sheet

-

Knowledge Gaps and Solutions

This section presents the listed knowledge gaps and their solutions with supporting images, screenshots and captions where appropriate/required.

Libraries being used:

The two libraries that I have chosen to compare are MPAndroidChart as well as GraphView. Both of these libraries give the ability to programmatically create graphs using either datapoints or entries using floats or doubles as the coordinates. They both offer a similar experience with the user being able to customise the graphs for what they are looking for however offer slight performance differences.

Implementation of libraries:

GraphView:
For the implementation of GraphView an implementation must be added to the dependencies section of gradle which can be found under the Gradle Scripts folder in build.gradle. The implementation must be made into dependencies section of the code and synced to be able to be used.

```
implementation 'com.jjoe64:graphview:4.2.2'
```

MPAndroidChart:
For the implementation of MPAndroidChart an implementation must be made to both the gradle.builds app and project files. In the project build.gradle the addition of the code below must be added to the repositories section of the code.

```
maven {url 'https://jitpack.io'}
```

In the gradle.build app file an implementation must be made simmilar to graphview. The code below must be added to the implementation section and synced for the library to be used

```
implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
```

Adding the Graphs:

To be able to see the graphs they must be added into a layout.xml to be displayed. To be able to do this an element needs to be added which can be done in the code section of the layout rather than design:

Graphview:

To add a graphview element to the screen the code below must be added into the layout file. It is noted that this is the main graph element that can be edited to change graphs rather than selecting between different elements

```
<com.jjoe64.graphview.GraphView
    android:id = "@+id/graphView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

MPAndroidChart:

To add a MPAndroidChart element to the screen the code below must be added into the layout file the same as graphview. Note that in this element the section after charts decides what graph the element will be rather then being changed using code.

```
<com.github.mikephil.charting.charts.LineChart
    android:id = "@+id/graphView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

Test Cases:

GraphView:

The code that I used to test graphviews performance was to get the element create a LineGraphSeries using getDataPoint method to return an array of DataPoints and add those data points to the graphview.

```
val array : Array<DataPoint> = arrayOf(
    DataPoint( x: 1.0, y: 0.0),
    DataPoint( x: 2.0, y: 1000.0),
    DataPoint( x: 3.0, y: 730.0),
    DataPoint( x: 4.0, y: 200.0),
    DataPoint( x: 5.0, y: 666.0),
    DataPoint( x: 6.0, y: 1200.0),
    DataPoint( x: 7.0, y: 10.0),
    DataPoint( x: 8.0, y: 60.0),
    DataPoint( x: 9.0,  y: 800.0)
)
```

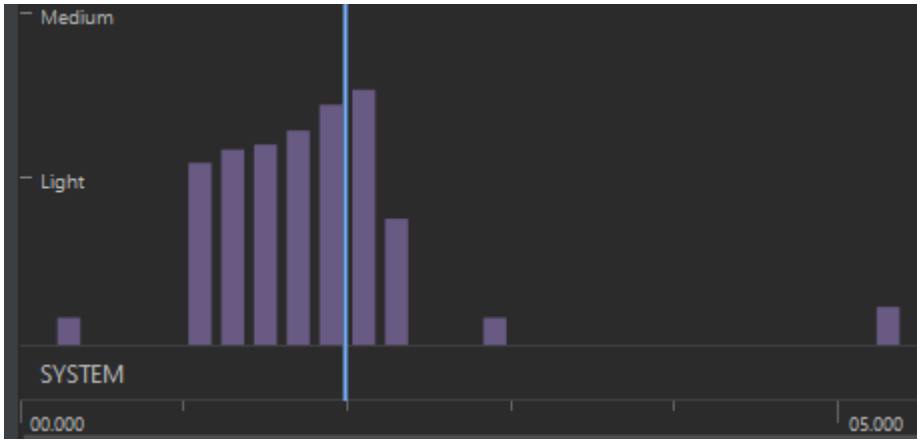
MPAndroidChart:

The process of adding data to MPAndroidChart is a bit more tedious then graphview having more steps which starts with getting the element by Id, next creating a LineDataSet using getEntryPoints which returns a MutableList of Entries aswell as pass in a name for the LineDataset, then creating a ArayList of ILineDateSet then adding the LineDataset to the Array list then creating a variable of LienData passing in the dataset and finally changing the data of MPAndroidChart by assigning it to the elements .data property.

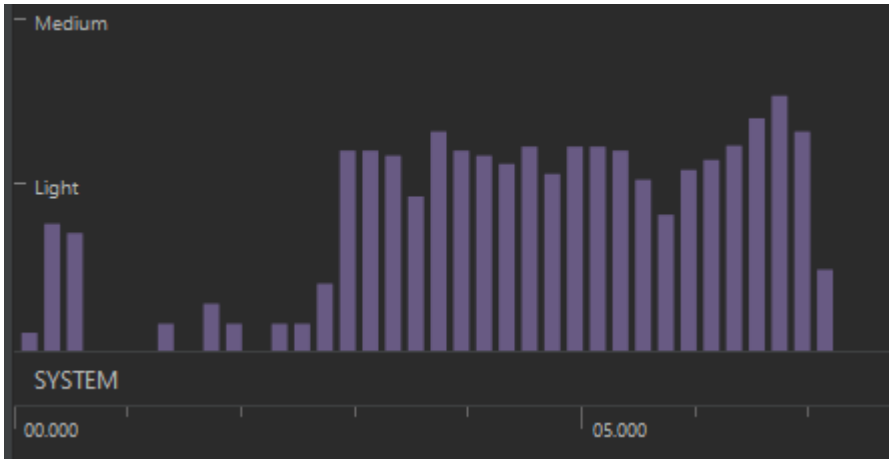
```
Entry( x: 1.0f, y: 0.0f),
Entry( x: 2.0f, y: 1000.0f),
Entry( x: 3.0f, y: 730.0f),
Entry( x: 4.0f, y: 200.0f),
Entry( x: 5.0f, y: 666.0f),
Entry( x: 6.0f, y: 1200.0f),
Entry( x: 7.0f, y: 10.0f),
Entry( x: 8.0f, y: 60.0f),
Entry( x: 9.0f,  y: 800.0f)
```

Performance:

The Library that I believed to have the best performance as indicated by the profiler would have to be the Graphview library which had minimal effect on the energy of the profiler. As shown by the data below the energy never goes above medium and only spikes for around a second or two before using no energy afterwards. For those reasons I believe GraphView to have the best performance.



Comapred to the energy use of MPAndroidChart which used similar amounts of energy by not going above medium, however, when compared to Graphview on the duration of use of energy you can see that then energy is consumed for about twice as long as Graphview being around 5 seconds.



Ease of use:

For ease of use I would have to give it to MPAndroidChart it allows the ability to easily change settings of the graph such as line colour, axis labels and colour of the series. You are also able to change the width of the line and the radius of each of the points of data.

```
lineDataSet.valueFormatter = myValueFormatter()
lineDataSet.lineWidth = 4.0f
lineDataSet.color = resources.getColor(R.color.green)
lineDataSet.setCircleColor(resources.getColor(R.color.gWhite))
lineDataSet.circleRadius = 5.0f
lineDataSet.valueTextSize = 14.0f
lineDataSet.valueTextColor = resources.getColor(R.color.gWhite)
return lineDataSet
```

Compared to graphview whose outdatted and hard to use which makes it hard to be able to change settings and customise your graph. This was the main reason that I did not use graphview was the inability to change most of the settings.

Overview and problems:

Overall, I found that graphview would have been the preferred library that I would of used however due to it being outdated and hard to be able to customise I went with MPAndroidChart which I was able to easily customise and was happy with the results. The main issue I had with both graphing libraries was the inability to add a date to the x axis to be able to track when an item was sold on the graphs. Besides that I found the library to work as intended for my project.

Git Hub