## Introduction

Mobile devices are for many daily tasks these days. The majority of the time is connecting with friends and family through messaging apps and calls on the default phone app. However, due to advances in technology mobiles can be used in other ways such as streaming videos, taking and storing images or running apps that have many functions and features. Moreover, all these functions take many resources from the memory, a percentage of your CPU and a certain amount of the phone's battery as the app takes energy to run. Henceforth this research assessment explores this, by changing parts of the code to compare differences in implementing icon images to a recycle view and how these different implementations change the performance.

## Method

The process of this experiment was done by doing the following steps. Firstly in the adapter class uncommenting one of these lines would display the image in different ways, such as the last line would display a small graduation hat symbol and another would display the number with a coloured background which would change when more numbers would be added, using the FAB button. Secondly, before running the app, changes would have to be made to the My unit class where the icon would either represent null or a method called drawicon(). Depending on the line of code icon would equal null or drawicon(). Lastly, the app was running for each line of code uncommented, and the performance was measured using a profiler that gives a percentage of CPU, amount of memory and level of energy used.

## Results

**Senerio 1 → Changes to a constant icon**

| Step\_ID | Evidence |
| --- | --- |
| S1 |  |
| S2 |  |
| S3 |  |

**Scenario 2** → Generated icon created on bind

| Step\_ID | Evidence |
| --- | --- |
| S1 |  |
| S2 |  |
| S3 |  |

**Scenario** **3** → Generated icon but created on initialisation

| Step\_ID | Evidence |
| --- | --- |
| S1 |  |
| S2 |  |
| S3 |  |

## Discussion

Out of all the scenarios tested scenario 1 is the best. It is evident this scenario uses 88.7 MB whereas scenario 2 uses 105.5 MB and scenario 3 uses 148.1 MB. Therefore, the lower the memory faster the ram and the faster the transfer of memory and hence the faster the application. Hence as the only change to code was the uncommenting of this line it can be concluded that the lower memory usage was due to this and as well over faster application use was due to this. The worst-performing application was scenario 3 and the slowest application as it used 148.1 MB. Which was the most memory used out of all the applications and there was more of a delay in loading of the application, which can be concluded to occur due to the higher memory usage, which may have occurred as the icon that was created by initialization by using a method. Therefore, more functionality would need to be used to transfer data. Ultimately it can be concluded that depending on how an icon was implemented in a recycle view. The amount of memory will be high or low. Uses a concurrent approach to loading a large number of data from a file, as concurrency enables the large processing of large amounts of data and its allocation to memory. This is done by multiple threads or tasks that can be run parallel to each other and data can be shared between them task or application. In addition, the data may be able to be processed and executed from different components in the mobile. Therefore, helps assist in the loading of a large number of items from a file which can be used in this application in the implementation of scenario 3.

## Conclusion

In conclusion, this paper was able to come to the conclusion that using initialization to generate an icon uses the most memory in the execution of the application, in addition, it can be concluded that using a concurrent approach may assist with processing the memory use.