**Initial requirements**

**Functional testing:**

* All error messages should display when needed
* All buttons should function accordingly
* Gallery should display all images taken by user and fill up all available slots. Screenshots that are taken after should not overwrite the images within the slots
* **Add your AR camera requirements here….**

**Non-functional testing:**

* Our database should be able to keep user password secure
* All screens within the application should be user-friendly
* Our database should be able to store over 100,000 user accounts
* AR camera should be compatible with majority android devices
* Our app should be able to handle multiple users at once
* **Add your AR camera requirements here….**

**Static and dynamic testing:**

* There should be no defects in any of the C# scripts
* All unit tests should produce no errors
* Integration tests should work successfully and produce no errors
* **Add your AR camera requirements here….**

**Quality assurance**

Our approach to quality assurance

To maintain the desired level of quality in our application, we came up with various scenarios that we tested for our functional and non-functional tests. These scenarios are based on areas that we didn’t really focus on as much, areas we didn’t develop well or areas that we were unsure about.

Another way we maintained the desired level of quality is by coming up with various defects that we tested for our static and dynamic tests. The aim of this was to find bugs within our C# scripts and compare expected outputs with our actual outputs to see if they were the same. This helped clean up our code, remove errors and prevent more bugs from being made.

How well our final system conforms to our initial requirements

From the results of our tests, we can see that some failed to work and meet our initial requirements. Our functional tests suffered a few failures. One of them was failing to display an error message within the login screen whenever the user inputted an invalid username and/or invalid password. We were able to fix this error at the end of it, thus justifying the importance of software testing as we were able to identify our failure, fix the failure and deem it as successful, but I digress. The other failure was in our profile page, where screenshots taken by the user within the AR camera would overwrite previously taken screenshots stored in the gallery. We expected the images to not overwrite. Unfortunately, this is something we could not fix as we did not have enough time.

Our non-functional tests also suffered a few failures. One of them failed our usability tests, where we tested to see if all our screens were user-friendly. We found that our create account screen forced users to input a lot of text, making it difficult to use. We fixed this issue by simply reducing the number of input fields. The other failure involved our compatibility test, where we found that top phones like Samsung Galaxy S7 Edge were not able to render the AR camera properly. We were able to fix the rendering issue for most Samsung phones, but there are other mobile devices that have the same issue, which we couldn’t entirely fix.

In some of our static tests, we were able to find defects within our C# scripts. Generally, it would be defects such as unused variables, dead code and variables with undefined values which we were able to easily fix. But in rare instances, we found string variables that we know would produce a NullPointerException when they are equal to null. These string variables query our MongoDB database for specific reasons (e.g. to find whether the inputted username and password exists in the database), and if it doesn’t find anything in the database, then it returns null. This is something that could be easily fixed with a try-catch block, but the query does not work when I use this. Hence, I had to leave this defect in the program.

Any changes made to initial requirements?

The only change I made to the initial requirements was in the non-functional testing section, where I initially stated that their should be a backup database. I found out that we would need to pay over twenty pounds to have a backup in MLab.com (where our Mongo database is stored.) Paying for resources is something we all agreed to not do, and as a result I erased this from the initial requirements.