# FIT2107 Assignment 3 - Individual Report

**Apoorv Kansal, akan57, 27821455**

- *What bugs, if any, did you find with your unit tests? Please be brief.*

  Some bugs were very minor such as validation issues with input parameters. These were taken care of quickly as the bugs were very obvious. For instance, we found bugs with minimum word count and top t amount of words input parameters where there were not working as expected around their boundary cases. By inspecting the specification, we were able to fix these bugs quickly. However, there were some other complicated bugs such as not producing correct frequencies for the correct date range. This required us to do more extensive debugging with various date ranges inputted.

- *Did you think your testing strategy resulted in adequate tests? Please explain why or why not.*

  I believe that the various combination of test strategies applied (eg. domain testing, category partitioning, branch coverage, statement coverage etc.) allowed us to produce a wide array of varying test cases. Having multiple strategies allowed us to think laterally and from different angles which gave us more test cases. As a result, the test cases produced were very valuable for the future with continuous integration. For instance, the input parameters were tested aggressively to ensure that clean input data was used when communicating with Twitter. Now, if the program changes significantly on the internal side (i.e parsing tweets), the use of these unit tests combined with continuous integration will ensure that the future programmers will not have to concern themselves about validation issues with input parameters.

  In contrast, I do believe that we can still have more test cases from each strategy for the future. Given the time constraints and the complexity of the program, we certainly have a large number of useful test cases, but I believe we can add more and think more critically about potentially hidden and very difficult bugs.

- *Do you think it works better for the programmer responsible for code to write the unit tests for it, or for somebody else to do it? Explain why.*

  I believe that both the programmer responsible for the code and somebody else should write unit tests. The code writer can easily write test cases using whitebox testing as he/she has extensive knowledge of their source code. However, being the code writer can lead to potentially low quality test cases (due to lack of motivation after finishing

code) or inaccurate tests as they are biased to the their code and potentially not test to a standard that meets the specification precisely. As a result of these issues, another engineer could come in and create test cases. They will have an unbiased approach to writing test cases and can focus more on testing to a standard that meets the specification (eg. through black box testing). However, an outsider writing test cases will not have extensive knowledge of the source code so this is where the code writer can support the outsider's shortcomings. Hence, having both engineers write test cases will ensure more rigorous testing is applied. They essentially complement each other and make up for each other's downfalls when it comes to writing test cases.

- *How long did writing mock code take? Was it a major component of the overall effort?*

  It took roughly a day where we spent time researching about mocking and Python's mock library. After truly understanding how mocking works in unit tests, we modified our test cases that used Twitter's API to mock any network access. This was a major component of the overall effort because we had to think critically about how to mock Twitter's API (eg. which objects and functions to mock and what should their mocked expected behaviour be) and also think carefully of what to test.

- *What would you do differently next time if faced with a similar task?*

  I would first study mocking libraries more extensively and also study the Tweepy library more. This would allow us to easily generate test cases without being concerned about the fine technicalities behind mocking and the usage of Tweepy.

  On a more general case, I would revise the various testing strategies and pick up on their advantages and disadvantages. This will allow me to select the most appropriate testing strategy in various scenarios. I would also create a much more detailed test plan like in Assignment 2 and get it reviewed by teammates so that there is increased confidence across the team about the tests. Having other review the test cases can ensure a more critical analysis is applied and allow others to think of other test cases as they might have different thinking angles.