FIT4004 Assignment 3 – Dekel Pilli

Bugs found via unit tests

The automatic tests helped us identify:

- A chunk of code that was accidentally deleted
- A bug caused by accessing the time object, instead of the time of a Tweet object

These are likely to have been found anyway, even without automated tests.

More bugs were found in the unit tests than by the unit tests. This is likely due to our lack of experience with writing tests in comparison to developing a relatively simple system. Also, ad hoc testing often caught bugs before we even ran the automated tests.

Adequacy of tests

While our unit tests did not find many bugs in the code, I do believe it produced relatively comprehensive tests. Our black box testing alone achieved quite high coverage, and most of gaps were easily filled by our white box testing.

Some aspects of the program were difficult to test, though, which was a flaw. For example, testing that the graph was mapped correctly from the tweets was not automated. Additionally, bugs originating in __*main*__ would not be caught by the automated tests.

Role allocation

While there is some value in having one person develop the functionality and its tests, as they would already be familiar with the needs and the nature of the unit to be tested, we preferred to have fresh eyes develop the tests. While this probably causes the test writing process to take a bit more time, it does have the benefit of forcing us to perform a pseudo code-review in addition to the testing.

Mocking

Developing test for units where mocking was necessary was considerably more time consuming than the developing tests for other units. While implementing these mocks required a lot of effort, I believe that it was skewed by the fact that neither of us have had to implement mocking before. In future, it is likely that while mocking will require a lot of time, it would not be as difficult as it was this time.

Future changes

One of the things we found happened quite often is that we needed to refactor the base code in order to make it more testable. In future, ideally, I'd know how to create easily testable code.

Another change would be better planning of time in regards to making sure units have tests before the next unit is developed.

Lastly, I believe in future it would be beneficial to keep track of bugs more formally.

While not a change in approach, I also believe that if we did this in the future we wouldn't have as much difficulty with mocking, Travis and, more specifically, testing GUI elements with Travis. The latter caused 12 out of our last 16 builds to be considered a failure by Travis.

Time log

Person	Start	End	Total time (hours)	Item(s) worked on	Same Location	Total:
Both	12/05/2017 18:16	12/05/2017 19:53	1.62	Test strategy	FALSE	21.57
Dekel	13/05/2017 14:50	13/05/2017 15:10	0.33	Test strategy rationale	FALSE	
Lawrence	18/05/2017 18:52	18/05/2017 19:44	0.87	Test strategy rationale, Git/Travis setup	FALSE	
Lawrence	19/05/2017 16:10	19/05/2017 17:39	1.48	Travis setup, Studying Tweepy	FALSE	
Lawrence	23/05/2017 17:21	23/05/2017 19:35	2.23	Module development, Test writing	FALSE	
Lawrence	24/05/2017 14:24	24/05/2017 14:49	0.42	Test writing	FALSE	
Lawrence	24/05/2017 16:21	24/05/2017 16:52	0.52	Test writing	FALSE	
Dekel	24/05/2017 21:45	25/05/2017 00:00	2.25	Module development, readme	FALSE	
Dekel	25/05/2017 09:30	25/05/2017 12:00	2.50	Module development, Test writing	FALSE	
Both	25/05/2017 12:12	25/05/2017 20:02	7.83	Module development, Test writing	FALSE	
Dekel	25/05/2017 20:02	25/05/2017 20:34	0.53	Module development	FALSE	
Both	25/05/2017 20:34	25/05/2017 21:33	0.98	Module development, Test writing, Travis configuration	FALSE	

Also available through the following link:

https://docs.google.com/a/monash.edu/spreadsheets/d/1H0_gN89HxvYc8HkLyGRTV3GafIeII7qZtcxbBi0fK5I/edit?usp=sharing