***ILP Testing Approach Justification***

The approach that I have ultimately opted to go with is ***approach number 3*** in the “ILP Testing Approaches” document. This approach is essentially a combination of the ***proactive*** and ***reactive*** approaches already discussed. One thing worth noting is that, although approach number 5 (***DevOps continuous testing approach***) is in theory the most optimal approach, as mentioned in the earlier referenced document, it is not feasible for this project, and hence will not be considered as a viable option in this justification.

One key aspect where this approach outperforms the others is ***time/ flexibility/ cost***. With a ***proactive*** approach, there would be plenty of time being spent on testing in the early stages of development, and then further additional time being spent in parallel with the development process to refine the generated test suite. This could end up harming the quality of the end-product. On the other hand, leaving the testing process for too late, as is the case in the ***reactive*** approach, could mean that not enough time has been spent on the test suite, and hence the quality of the test suite would have to suffer, and there could also be undetected bugs/ issues in the end-product. Whilst using ***iterative approach*** allows for a more continuous timeline, where effort is being distributed more evenly, the main downside is having to break the task down into iterations of different sizes, which may sometimes result in time being wasted when not enough changes/ progress has been made. However, all these issues are accounted for with ***approach number 3***: the continuous timeline, without the need to break the development down into “iterations”, allows for testing to be conducted at various stages, bugs to be identified at suitable times, and all without damaging the quality of the end-product or the test-suite.

Another key aspect to be considered is ***resources*** being used. In the current example, the two main resources to account for are ***time*** and ***attention***, and since time has already been discussed, this paragraph will mainly focus on ***attention***. In the case of the ***proactive*** approach, most of my attention will be drawn to testing in the early stages, and hence there will be a heavy commitment of resources early on, with more overhead in the latter stages when refining the test suite. With a ***reactive*** approach, this attention is delayed until after the whole system is done; however, this could result in delays to the release of the end product, since bugs will be identified at a late stage, and, depending on the severity of the bug, may take a long time to fix. The ***iterative approach*** allows for better resource allocation, in that the attention constantly switches between development and testing; however, there is still significant overhead, since there could be plenty of small “iterations” triggering this switch between development and testing. With ***approach number 3***, there is once again added flexibility, meaning that resources are ultimately allocated in a much more optimal way for the developing team.

Having considered all these factors, as well as the ***risks*** (of losing quality on either the end-product or the test suite, or both), I believe my chose testing approach is justified in the context of my portfolio.