TESTING EVALUATION AND SUMMARY

Requirement 1: The system shall fetch all data required by the drone from a REST server. The system shall also verify all data fetched from the API.

Gaps and omissions in testing: After reviewing the test results, I have noticed a few gaps in the tests. Firstly, validation of the format of the data fetched was not executed. This was due to a lack of resources and access to the REST server. Thus, I was unable to construct the scaffolding required to simulate data in other formats within the REST server. Secondly, the test suites generated for URL verification were not exhaustive. This implies the system might halt execution if it encounters URL instances which were not accounted for.

Target coverage and performance levels: Currently, I have executed systematic functional tests and structural tests. Firstly, the target performance level for the functional tests would be for the system to pass all test cases executed. This helps ensure the minimum correctness and robustness of the system. Secondly, the target branch coverage and method coverage would be a minimum of 70%. I am observing the branch coverage instead of path coverage as path coverage could provide false confidence. Since there are twice as many branches compared to paths in a system, a 70% branch coverage subsumes a 70% path coverage.

Evaluation of results: *Please refer to the "test results logging" document for a detailed list of test suites executed and code coverage calculated.*

From the results of the executed tests, I observed that most test cases in my test suites for systematic functional testing have passed. This shows that most boundary cases included in the test cases have been caught. Although one test case, whose URL input includes white spaces, failed, it was caught in the second test suite, as it was leading to an unresponsive REST server. This shows that the systematic functional testing has been effective, and I have some confidence in the system. For structural tests, it was observed that the branch and method coverage for the entire system was low. However, since this is just an integration test, only the classes and methods involved would be executed. Thus, looking at the coverage for only RestClient and CentralArea classes, I managed to achieve 100% branch and method coverage.

Actions to achieve target levels: From this, I can conclude that the testing approach has suitably met the adequacy criteria. However, this does not guarantee the modules involved are bug-free. Using code coverage as a measurement is good to discover missing test suites, but it is only a proxy for the thoroughness or adequacy of existing test suites. It is easy to improve code coverage without improving a test suite. Hence, with more resources, more exhaustive test cases for URL, REST server and data validation could be executed. Though this would increase the cost and time required and would have to be scheduled into the project plan earlier on for future projects.

Requirement 2: The system shall validate all orders received before generating a flight path for the drone.

Gaps and omissions in testing: There are a few gaps in the testing approach for this requirement, and this is due to unrepresentative synthetic data. The synthetic order and restaurant data used for testing simulate real-world instances as closely as possible. However, synthetic data does not capture all real-world instances. Unpredicted boundary cases would be missed by the system and could cause the system to crash if released. For example, this scenario could be observed when validating credit card numbers and CVVs. Currently, the system is only able to validate against MasterCards and VISA16s. These credit cards have 16-digit credit card numbers and 3-digit CVVs. With more time and resources, a larger variety of credit card types could be validated by the system.

Target coverage and performance levels: Currently, category partition tests, systematic functional tests and structural tests have been executed to validate this requirement. Firstly, the target performance level for the category partition and systematic functional tests for the system would be to pass all test cases. This helps ensure the correctness of the system and the safety of customers. Secondly, the target branch coverage and method coverage for structural tests would be a minimum of 85%. This requirement concerns the security of a customer's transactional details, hence why the adequacy criteria for structural testing is relatively stricter.

Evaluation of results: *Please refer to the "test results logging" document for a detailed list of test suites executed and code coverage calculated.*

From the results above, I observed that all edge cases presented in the test suites have been successfully detected by the system. This increases confidence in the system's robustness. Additionally, the structural tests have revealed test cases which were missed in a few test suites. Although the code coverage appears to be low, this is due to various unused methods and classes, as it is an integration test and only parts of the system will be used. Additionally, some branch statements not covered included checking for null objects, which I was unable to test due to limited resources. Thus, after recalculating the total code coverage for the relevant classes, a method coverage of 93% and a branch coverage of 88.5% were achieved. Bearing all the mentioned considerations in mind, all testing approaches satisfy the adequacy criteria and I can be confident in the correctness of the functionalities tested in this stage of the development.

Actions to achieve target levels: Although the adequacy criteria have been satisfied, there are still improvements to be made to the test suites. With my current lack of resources, I am unable to generate accurate synthetic data. This could be tackled in later phases of DevOps after monitoring the performance of the released system. This implies that throughout the development of the system as the order validation functionality increases, the regression testing approach can be taken as test suites will have to be executed repeatedly. This process could be automated to help decrease the cost and time required, along with avoiding human errors with testing.

Requirement 3: The mean time for the system to generate flight paths for all orders in a day and produce the resulting files in JSON format shall not exceed 60 seconds.

Gaps and omissions in testing: There are numerous omissions in the testing approach for this requirement. Currently, the response speed of the system has been verified under performance tests. Even so, it has only been tested for the predicted average users each day, which gives insufficient confidence in the system. Verification for other performance aspects like varying the interoperability, throughput, and environment of the system was not executed. Completely verifying these aspects have high significance, as it allows developers to locate bottlenecks within the system and measure stability under peak application usage. It also allows developers to better understand the limits of the instrumentation and tools used for the system. Side note: the testing for this requirement is executed under the assumption that tests verifying the system's functionality have been executed.

Target coverage and performance levels: The performance test will be evaluated based on a pass-or-fail basis, dependent on the system's execution time. The target performance level for this test involves the system completing execution within 60 seconds for all orders in a day.

Evaluation of results: *Please refer to the "test results logging" document for a detailed list of test suites executed and code coverage calculated.*

From the results, I observed that at predicted user volumes, the system can satisfy the adequacy criteria of completing execution in under 60 seconds. Although it performed well in the current tests, I would argue that it is unsuited for deployment. The performance tests executed are not thorough enough and would not accurately reflect the system environment in real-world settings. Thus, I conclude that more rigorous performance tests have to be executed before the system appropriately satisfies the adequacy criteria.

Actions to achieve target levels: Various types of performance tests must be executed to achieve a satisfactory level of performance adequacy. They have not been executed due to the lack of resources. However, assuming access to better tools and resources is available, the following tests would be executed:

- Endurance tests: this helps verify the system's performance at high levels over a long time period. This could reveal potential memory leaks and eliminates new issues resulting from a heavy load on the system over extended periods of time.
- Stress tests: this helps verify if the system can perform under peak conditions and identifies how the system responds correspondingly.
- Load tests: this helps evaluate the system's performance under a significantly higher number of users and reveals the bottlenecks in a system.

These tests require extensive simulation of the execution environment and require significantly more resources compared to other testing approaches. Thus, these tests would only be executed when required, like after completing the integration of two modules.

Additional testing processes:

Below are further testing approaches that would be implemented if there were sufficient resources and tools available:

- Acceptance testing: Currently, I am unable to execute this test approach, as the system is still incomplete and could not be tested by users. However, after the system's completion, I would perform alpha-beta testing as an acceptance test within the School of Informatics. This allows me to validate the system against user expectations to obtain insights into improving user experiences.
- Compliance testing: Only a portion of the system's functionalities are completed, thus I am currently unable to take this testing approach. However, it is important for compliance testing to be executed, as it validates that the system adheres to any regulatory bodies. This is a strong concern as the system has various safety and privacy regulations that must be satisfied.
- Penetration testing: The system contains transactional information of the customers, which is sensitive information and must be protected. Thus, penetration testing must be executed to ensure the security of the system is not vulnerable to cyber-attacks. Penetration testing helps validate existing security measures and identifies any frameworks that require improvements.

Note: all tests mentioned are relatively more expansive to perform, thus they will only be done whenever deemed necessary. For example, penetration testing will be done on an annual basis due to the high costs and risks it incurres.