Test Planning Document

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

Priority and prerequisites: This requirement is a dependability quality, as it affects the correctness and robustness of the system. The requirement is significant for the viability of the system. Thus, it is categorized as a high priority, and it is necessary to invest a relatively large amount of resources to verify it. The system interfaces with the external REST API and other modules and classes, so this would be classified under integration level testing.

Testing and scaffolding: The partition principle suggests that to help ensure correctness, I should decompose requirements into smaller units. Hence, systematic partition testing, a form of functional testing, could be used to test each category of the input space and the boundaries between them. Structural testing would also be used to verify the completeness of the test cases created. This would require various scaffolding to simulate the REST API to properly verify the interface between the system and the server. Some data for the simulation would be necessary, for example, the coordinates of the no-fly zones, a list of orders, a list of participating restaurants, and more. Scaffolding that simulates boundary cases and invalid inputs would also be required, as it's suggested in chapter 10 of Y&P that failures often lie at the boundaries of input space. This would also help ensure more bugs are caught and the system is robust. However, this may involve some effort that has to be scheduled. Thus, this form of testing should be carried out early during the planning and creation phase of the project.

Process and risk: From chapter 20 of the Y&P textbook, there could be a schedule risk. As the requirement is essential for the development of other modules of the system, the critical dependence and critical path are increased. This increases the length of the project schedule and could also affect the quality of the development. Firstly, this could be mitigated by providing early warnings and feedback to shorten the critical path and increase the code quality. Secondly, scheduling design, code and test suites can increase code quality as well but should only be done during scheduled inspection times to ensure there is sufficient time scheduled for system development.

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

Priority and pre-requisites: This requirement is a high priority requirement. It ensures the correctness of the system by removing deliveries with invalid transaction details or pizza orders. It also ensures the safety of a customer. For example, people with dietary restrictions or allergies would not get their orders mixed up. Thus, its testing is categorized as a high priority and is necessary to invest a decent amount of resources into validating and verifying it. This requirement calls for various interactions between the system's modules and thus could be verified with integration level testing.

Testing and scaffolding: From the partition and restriction principles in Chapter 3 of Y&P, it is suggested to divide the integration level test into unit level tests. This includes validating and verifying each order. For example, each order must have valid transactional details, valid pizza orders and more. Thus, firstly, I will use category partition testing for unit testing each order's validation criteria. Secondly, I could use pairwise combinatorial testing to verify the combined validation for an order. Both testing techniques help fold the input space and decrease the amount

of scaffolding required. As seen in this case, only some stubs are needed. For example, to simulate the data on participating restaurants and customer orders.

Once the unit level testing has been completed, I will use functional testing on the integration level to validate the interfacing between the orders module and the REST API. This would include verifying the total number of valid and invalid orders detected on orders in a single day taken from the server. The integration test would require the same scaffolding used in previous tests and would not require any drivers to be added. Additionally, structural testing will be used alongside both test levels to ensure test cases ran cover most paths in the system and more bugs are caught early on. As Chapter 4 of the Y&P textbook suggests, testing should be performed early in development. Thus, I would ideally begin testing throughout the planning and creation phases of the DevOps process.

Process and risks: Many synthetic data are required as scaffolding for testing this requirement. There is a risk of it being unrepresentative of actual data, which could cause the system's design to perform poorly. This is likely to occur, as outliers and boundary cases are difficult to predict and produce synthetically. Thus, it is necessary to plan various validation activities to ensure all invalid orders are caught.

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.

Priority and pre-requisites: This is a system level non-functional requirement with medium priority. Though it is still important for the system to have a reasonable response time, it does not affect the basic correctness and functionality of the system. Hence, the testing for this requirement is categorized under a lower priority and less effort will be invested into validating it.

Testing and scaffolding: This requirement is a measurable attribute of the code, so a means to measure it will be required. Additionally, it can only be tested after the completion of the system, so it will occur in the creation and verification phase of the DevOps cycle. To validate this requirement, some synthetic data will be required to simulate the drone's environment. For validation, logging the time of each delivery could be a viable way of assessing the response time. This suggests that various tasks must be scheduled for the testing, including the following:

- Generating synthetic data to test the outcome of the system.
- Building some scaffolding to simulate the hardware of the actual drone so early tests are possible on synthetic data.
- Designing the logging system to capture real performance when tested.

From above, it is necessary to have various scaffolding for testing. Firstly, some sort of simulation for the hardware is required to test the software, which must be scheduled early on. Secondly, having synthetic data for the simulation would require some effort that has to be scheduled as well. Finally, the system test would include combining all modules and interfaces and using synthetic data to observe its response.

Process and risk: There are risks in the development and test execution for this requirement. Firstly, since a completed system is required for testing, inadequate unit tests and analysis could be executed before this stage of the plan. I could mitigate this issue by scheduling consistent inspections of the design, code, and test suites to ensure the code quality remains above a set standard. Secondly, as much scaffolding is required for testing the system, resulting in costs may exceed the planned amount. This could be resolved by minimizing the parts that require the full system to be executed and increasing intermediate feedback on the system.