Testing Techniques(3.1)

Functional Testing for requirement SR-1:
We will generate the payment and order info data with many edge cases, extreme cases as well as enough normal cases. Then we must ensure that every time the test generates data, at least 70% of the orders from the data that are valid must be delivered. This will ensure the requirement is satisfied. Because the following requirement is system-level, the following approach would be less expensive to design and execute than structural testing.

Pairwise combinatorial testing for requirement IR-2:
We will combine the attributes into two groups, payment info and order info and use pairwise combinatorial testing to efficiently cover all pairs without going exhaustive with all possible combinations of attributes. Because a single attribute failing within any group, will fail the group and in turn fail the entire test. Such an approach will reduce the number of tests to only four, and significantly cut the cost and time needed.

Testing Results (3.3)
The system-level requirement SR-1 was tested on five different days chosen randomly. For each day, the data included valid orders and invalid orders. At most forty of the orders were valid per day. For every single test, the drone simulation has successfully delivered at least 70% of the valid orders, meaning all tests have successfully passed. The integration-level requirement IR-2 was also tested on five different days and the data included both valid and invalid orders. Invalid orders were a combination of orders being invalid due to payment information or order information. The conducted tests have produced results which matched the expected output and therefore were successful as well.

Limitations, Gaps, Omissions (4.1)
Although the synthetic data was generated with a reasonable range of different cases, it might not necessarily reflect the real-world data and therefore there is no guarantee of delivering 70% of valid orders. Due to time constraints, it was decided to create data based on some rules and have a sufficient range of extreme, normal, and exceptional data. This could be handled by doing research and collecting real data about the orders the participating restaurants already get and testing the system on it. Another gap that has been identified is that the performance of the system on the average machine is unknown. Every machine can show a different performance due to the hardware, and because of the algorithms used the difference can be drastic. This happened due to a lack of machines to test on. This could be mediated by taking an average market server machine used by a similar system and testing how fast our system can perform.

Target Levels Discussion (4.4)
Although the testing has achieved >=70% of valid orders being delivered, this might be related to the data that is being generated. As proposed before, collecting sample data from the restaurants about the orders and using this collected real data in the testing will help verify the requirement and produce more accurate results. The performance metric of running under 60 seconds has been achieved, however, this can be improved by picking a range of mid-market machines and testing performed on them to understand a more realistic level of performance. Finally, to achieve 100% code coverage for the last requirement, the synthetic data generator specification must be changed

so that all possible combinations of payment information are tested. For example, currently, empty CVV is not being tested, since no such data exists, therefore code coverage is only at 97%.