Test Planning Document

**Test planning for flight control:**

Requirement 1: The drone must be able to navigate from point A to point B following the restrictions of:

1. It must not cross “no fly zones”
2. Should not cross the Central Area boundary multiple times during a single trip
3. Must be able to keep in track of battery
4. The path it creates should be sensible

Chapter 3 notes that we should handle hard testing and verification problems by partitioning the criteria of the test.

In this case I’ll split them up to:

1.

- Testing if it crosses “no fly zones”

- The process doesn’t pass 1 second per restaurant

- Testing the interaction with the Central Area boundary

- Testing and evaluating to path plan the software creates for the drone

2.

- Testing the interaction with out of battery situations

For the first part, because we already have a algorithm to create a flight path and it can be seen from the geojson.io website, I will test the functionality of the 3 criteria by checking every case the drone could travel multiple times (let’s say 5 times each), Dominoes -> Appleton Tower, Appleton Tower -> Dominoes, and etc. This could check all the possible combinations of travel to look for violations of the rules inside the algorithm. Also I could be able to check the time it takes for the process to terminate. The vulnerability of this testing method however would be that it needs me to physically check the path of the drone using the geojson.io website as well as measuring the time. As the whole validation process is not done automatically, there could be errors that I miss.

The second part will be tested out by creating tests to drain the battery by making the drone fly multiple trips. The important aspect for the test would be to find out what the software does when near the battery limit of the drone.

Requirement 2: The software must be able to process the order and give a final cost of the delivery following the restrictions of

1. Should be able to accurately send out an error for the order validation problem it encountered
2. Should be able to detect which payment issue it’s facing

The testing would include testing different orders with distinct validation issues to check if the software can successfully filter out the problems properly. Also to insure that the orders that should pass don’t get stopped by the program, I would also want to go through a good number of valid orders to make sure these orders passed. Potential omissions could be that because there are a huge number of orders to validate, some mistakes could be made by me when checking if the order is valid or not. Also, because the program is only logging out one error for each order (if there is one), if the order has multiple errors, the test wouldn’t be able to catch everything.

**Scaffolding and Instrumentation**

Requirement 1: I will use the runtime measure tool as instrumentation in java to test the run time of the path finding algorithm and see if it meets expectations. I think this would be the one and only making it the most adequate way of testing the runtime of the software. Since we can’t really check the geojson output files using the testing methods in java, scaffolding won’t be necessary for this section.

Requirement 2: For this requirement, scaffolding will be needed to create some kind of simulator for the software so that it automatically feeds orders from the REST server, could just use orders from specific days inside the program.

Evaluation of Instrumentation: The instrumentation that will be used for requirement 1 seems good enough because the runtime is a measurable factor of the code and it must be done by some sort of instrumentation. However, depending on which time check function I use the accuracy of the runtime could vary.

**Process and Risk**

Requirement 2 should be finished first before doing requirement 1, because the orders have to be verified first before the drone does any travelling. The risks of requirement 1 would be that during the testing process, most of the testing has to be validated by me by using the geojson website. This of course is a process that could be time consuming and could delay the exhaustive tests of the path finding algorithm. Also for checking run time, there would be a difference between systems because they would have different processing speeds. Testing requirement 2 also holds risks because I would have to create manual errors to feed to the test, which is also very time consuming and the testing amount could be insufficient to reflect real life use.