Test Plan

High Priority Requirements

Requirement 1: Orders Must be Validated

- Test Planned- Functional Tests
- I plan to test every possible invalidity an order can have and then a valid order and then check if the system finds these invalid elements. The class works by taking an order and checking all of its variables, and changing the validation code of the order to whatever error it has or no error. Therefore my tests are going to check if the order validation code matches the expected one.
- Parts of order being tested:
    - Invalid card number
        - Card number with letters
        - Short card number
        - Long card number
        - Intended Output: Order Validation Code.CARD_NUMBER_INVALID
    - Invalid card expiry date
        - Short expiry date
        - Invalid format
        - Expired card
        - Letters in card expiry
        - Intended Output: Order Validation Code.EXPIRY_DATE_INVALID
    - Invalid card cvv
        - Short cvv
        - Long cvv
        - Letters in cvv
        - Intended Output: Order Validation Code.CVV_INVALID
    - Pizza ordered not on menus
        - Intended Output: Order Validation Code.PIZZA_NOT_DEFINED_INVALID
    - Pizzas ordered from multiple restaurants
        - Intended Output: Order Validation Code.PIZZA_FROM_MULTIPLE_RESTAURANTS
    - Pizza ordered from a closed restaurant
        - Intended Output: Order Validation Code.RESTAURANT_CLOSED
    - Order total incorrect
        - Intended Output: Order Validation Code.TOTAL_INCORRECT
    - No error
        - Intended Output: Order Validation Code.NO_ERROR
- Process and Risk - Inadequate data using to test the system can lead to errors not being discovered and then delay the rest of the process

Requirement 2: Drone route must be valid

- Tests planned - Functional, combinatorial and Manual Testing

- Firstly I plan to perform functional unit tests on the part of the system that checks the smaller parts necessary for the route planning. These are:
    - Distance from one point to another - here I will make two points with a known distance between them. The test passes if the value returned by the function is the distance expected, and otherwise the test fails
    - Two points close to each other - the specification states that two points are deemed close to each other if they are within 0.00014 degrees of each other. I will have two tests, one where the points are close to each other, and one where the points aren't. The tests pass if the function returns true and false respectively, and fail if this is not the case.
    - Point inside a region - here I will do combinatorial testing. I will make a few different regions that are different shapes, I plan on trying regions from triangles to hexagons. I will also create points that are inside, on edges and corners for some regions. The tests pass if the expected boolean is returned, and will fail otherwise.
    - Point's next position - the drone move must always be the same distance (0.00014 degrees) and therefore a function is required to calculate this next position. The function is given a point and an angle and using this calculates the next position. The test passes if the next position calculated by the function is the one I expected, and fails otherwise.
- Following this I plan to perform manual integration testing on the whole path finder function. To do this I will specifically pick points on a map that will put the path finder under stress and test all of its capabilities. I will get all the coordinates that the drone passes through between these points, and then put them into geojson.io and look at the path visually on a real map. The test fails if the path goes into a no-fly zone, leaves the central area after reentering it, doesn't start or end where intended. If these are all fine then the test is passed.
- Process and Risk - A risk with these tests are the time constraints. Designing the path finder and testing it effectively is a very time consuming process which may leave me with insufficient time to effectively finish and test the whole system.

Requirement 3: System must receive orders, restaurants, central area and no fly zones from an external Rest API

- Tests planned - Integration Tests
- To test the Rest API integration I plan on picking a date from the rest and then requesting all the data from that day. The tests will pass if the data obtained from the Rest API isn't null.
- Process and Risk - I have never got items from an external system before and therefore am unfamiliar with this process. This could lead to ineffective integration tests.

Requirement 4: System must run from command line with two inputs (date and website url) and must generate three output files

- Tests planned - System Tests

- To test the system I plan on running it from the command line as intended and then checking the output files to make sure they have the intended data within them. The test fails if the system doesn't run (first part of requirement unfulfilled) or if the output files are empty or have data in the wrong format within them. Otherwise the test passes.
- Process and Risk - If the unit and integration tests aren't good and don't test the system properly, the system can fail without knowing why

Requirement 5 - Application must run in less than 60 seconds

- Tests planned - Timed Testing
- I plan on running the program for 14 days on the rest server and then check the time the program takes to run on each of these days. The test passes if the system can run in under 60 seconds for every day, otherwise it fails.
- Process and Risk - This task is heavily reliant on the efficiency of the system. If the system isn't efficient then this could lead to this task failing and there being a very time consuming process of making the system more efficient.

<u>Low Priority Requirement</u>

Requirement 6 - System should run under different amounts of stress

- Tests planned - Load Testing
- I plan on running the program and giving it an increasing amount of orders and restaurants, finishing at 500 orders. The test passes if the system can still run in under 60 seconds with this amount of orders, otherwise it fails.
- Process and Risk - The system is designed to deal with a relatively similar number of orders every time, and therefore I am uncertain on how it may act under stress.