2.1 Test Plan Document

The test plan was constructed in line with the lifecycles discussed in chapter 20, combining unit, integration and system-level testing.

Unit-level testing

Unit level tests were first developed for the following core requirements:

- Drone movement criteria
    - Drone should only move in 16 valid directions (0 degrees is East, with directions being in 22.5-degree intervals)
        - Check each move in a path generated by the A-star path finding algorithm, ensuring it is at one of the 16 valid angles.
    - Each drone movement should be a straight line of size 0.00015 degrees
        - Check each move in a path generated by the A-star path finding algorithm, ensuring it is of size 0.00015 degrees.
    - Drone path should not intersect no-fly zones
        - Check each point in the path is not within a no-fly zone.

These tests ensure correct behaviour of fundamental logic prior to integration.

Integration-level testing

Integration level tests were then developed, testing the integration of the requirements testing during the unit-level testing stage and integration of our system with external systems:

- Integration between drone movement units to give a valid 2D path
    - Ensure each movement of the drones is of the correct size and at a valid angle
    - Ensure A-star returns a valid path, avoiding no-fly zones
- Correct retrieval of data from the ILP REST service
    - Manually check data in ILP REST service
    - Compare data received from retrieval to the data stored in the REST service, ensure it matches.

System-level testing

System-level tests were planned to test the complete REST service:

- Performance tests to ensure all tasks complete within 30 seconds
    - Time each task to ensure it runs in under 30 seconds
- End to end tests to ensure valid JSON output
    - Inspect output of each task to ensure a valid JSON format

By prioritising unit-tests, this tests plan allows for errors to be found where they are cheapest to fix, before continuing to integration tests to ensure correct between internal components within the system, and interaction between the system and an external system, and finally moves to system testing to test functionality of the complete REST service and ensure good performance and robustness, and ensure it matches all requirements from system-level down to unit-level.