Limitations of the Testing Processes

This document contains evidence that the limitations of the testing processes conducted for *R1.1.1* and *R2.1.1* were identified. It also evaluates how far the test results fell short of the target coverage/performance levels that were set and discusses possible ways to achieve those target levels.

R1.1.1

Limitations

- The functional requirement was tested at the segment level, meaning that scenarios where no viable end-to-end route exists are not catered to. For instance, if the delivery point lies inside a no-boundary region, then segment-level tests for the final few candidate steps will fail, but it is only through a full-route check that this specific problem can be identified.
- Moreover, while path coverage and statement coverage provide strong confidence that all paths in the flowchart were executed, and it Is easy to improve coverage (as was seen when more test cases were added after the first iteration), we cannot say definitively that the test suite is sensitive enough to detect faults in the decision logic itself.
- Also, although the tests cover each identified violation mode, they are not sufficiently rigorous to validate behaviour under floating-point precision and tolerance edge cases. In particular, scenarios where the candidate point or segment lies extremely close to a no-fly zone boundary may lead to incorrect classification due to rounding errors, or errors arising from comparing floating-point values using a fixed epsilon. This might lead to false positives or negatives.

Target Coverage

Since the requirement's core logic was reduced at the test planning stage to the validation of three constraints, and a model-based testing approach was used, it was straightforward to test each constraint and identify any untested cases. Therefore, the target adequacy levels were set as follows:

- **100% path coverage** across the segment-validation flowgraph (to ensure that each violation mode, as well as the valid path, is exercised)
- **100% statement coverage** for the top-level segment validation method and the relevant helper methods (for greater assurance that the implementation corresponding to each path is executed thoroughly during testing)

Comparison of achieved test levels against target

The initial test-suite derived from the flowchart achieved full path coverage as well as 100% statement coverage for the top-level segment validation logic. However, lower-level helper method (*pointOnSegment*) contained some unexecuted statements. This inspired additional test cases to close the gap and achieve 100% statement coverage across all methods involved.

This means the achieved adequacy levels in the second iteration of the tests meet the target levels defined in criterion *4.2*, increasing confidence in the correctness of the segment-level enforcement logic.

Steps to address limitations and achieve stronger target levels

- System-level testing would be necessary to exercise full route construction, as scenarios in which no viable end-to-end route exists require interactions between multiple components to ensure that appropriate responses are generated. A similar approach to earlier testing could be adopted, using flowcharts to model the route construction logic and designing requests that exercise all feasible route-formation outcomes and corresponding responses.
- Mutation testing to assess how sensitive the test suite is to faults in the decision logic.
- Including boundary-adjacent test cases (for e.g., coordinates differing by ±ε) and stress testing with randomly generated near-boundary segments to evaluate numerical stability.

R2.1.1

Limitations

- Performance testing was limited by assumptions of static and controlled inputs, whereas in practice, drone availability data would change dynamically as orders are created or completed.
- JMeter results are test-run specific, making it difficult to compare performance over time or correlate metrics with system behaviour. This surely limits confidence in long-running or production-level performance.
- Since each load configuration was executed once, the results cannot be used to compute confidence intervals for mean or p95 response time, reducing statistical confidence in repeatability.
- Moreover, the predefined load configurations for each test case cannot fully reflect real-world usage patterns, and the current results suggest that system reliability under load may be overestimated. In the graph, the average response time does not rise as would be expected at higher loads. So, it is impossible to adequately test performance on simulated loads given current resources.

Target Coverage

The requirement states that the mean response time must be less than 3 seconds under normal load. Therefore, the target performance level was defined as:

- Mean response time < 3 seconds under expected normal-load conditions.

For the resulting average response time to be considered as an appropriate metric, additional performance indicators were also considered:

- 0% error rate under normal load, and
- acceptable p95 response time, meaning that even the slowest requests still respond within a reasonable time.

Comparison of achieved test levels against target

During the first execution of TC-3, a concurrency issue was identified, resulting in an error rate of approximately 70%. This was traced to shared mutable state and was corrected. After the fix, all test cases achieved an error rate of 0%.

Across all test cases, the mean response time remained between 1.308s and 1.382s, indicating low variability and consistent typical performance across the tested load configurations. The 95th

percentile response time ranged from 1.815s to 2.164s, showing greater variation than the mean and highlighting occasional slower requests. TC-2 produced the highest p95 value, suggesting that performance may be sensitive to warm-up effects or unexpected background processes on the test machine.

Overall, the results indicate that the requirement target (mean response time < 3s under normal load) was achieved with a significant margin under the tested conditions. However, taking into account the limitations highlighted above, it was impossible to adequately conduct this performance testing.

Steps to address limitations and achieve stronger target levels

Target performance levels would require the system to remain stable even as its state evolves during operation to reflect dynamic changes in drone availability. To achieve this, we would need:

- a changing database state, for instance, with regular updates to drone availability,
- more realistic traffic models, including burst loads and uneven request arrival patterns. This could be obtained through beta testing or by conducting surveys targeting potential users of the system to gain insight into typical order patterns
- performance confidence that is not based on short test runs alone, but on observing acceptable response times and error rates over longer periods and across repeated executions.