Software Testing 2025/2026 – Test Planning (LO 2)

The requirements defined in LO1 considered in this plan are:

- R1.1 - The REST service must validate all orders submitted through POST requests and return a result
- R1.2 - The service must calculate and return a valid flight path between the order's restaurant and Appleton Tower
- R2.1 - Route calculation must complete in fewer than 60 seconds

Priority and Prerequisites

R1.1

This is a high-priority functional requirement, as failure in order validation could lead to:

- Incorrect charges being applied to users
- Invalid definitions or amounts of pizzas being accepted
- Security and robustness issues with card details and JSON structures

Thus, a large amount of the testing effort should be directed towards this.

The validation logic is well-partitioned into discrete rules (e.g. valid card numbers, restaurant consistency). This supports exhaustive and systematic validation, individual testing before combining multiple validation types, and restricts the problem for simplicity.

Prerequisites include:

- A defined order JSON structure
- Defined validation error codes

R1.2

This is a high-priority functional requirement, as failure in route calculation could lead to:

- Drones entering no-fly zones
- Drones leaving the central area
- Other incorrect or unsafe navigation

Given the safety implications, a large amount of testing effort should be allocated to this requirement.

R1.2 is easily partitioned into independently-testable constraints – including step size and direction, goal nearness, geographic bounds, no-fly zone avoidance, central area consistency and valid start and end conditions. This allows for systematic testing and stronger fault isolation.

The problem can be restricted in testing using simple synthetic maps with limited no-fly zones and fixed restaurant locations.

Prerequisites include:

- Defined representation of zones e.g. no-fly, central
- Known restaurant and goal locations
- Defined movement rules for step size and direction

R2.1

This is a measurable quality attribute relating to system performance. While lower priority than safety and functional requirements, failure to meet this requirement could cause:

- Poor user experience

- System timeouts

Thus, a smaller amount of testing effort should be allocated to this.

This will primarily occur at a system level, as R2.1 cannot be meaningfully tested without a full route calculation implementation.

Prerequisites include:

- A complete route calculation implementation
- Representative test data for restaurant and goal locations, no-fly and central zones

Scaffolding and Instrumentation

R1.1

Instrumentation will be included to detail which validation rules cause an order to be invalid (present in the validated order structure), increasing visibility and making failures more easily diagnosable.

Scaffolding includes:

- Synthetic order generation to test individual and combined validation rules
- Mock restaurant and menu data

Redundancy in fault detection is included in this approach, through both expected test case output and the error codes being stored in the validated order structure.

R1.2

Instrumentation will be included to check and detail if any points in the path violate zone or step distance/direction constraints, in both path generation and the final result. This allows visibility into path generation behaviour, not just the final result.

Scaffolding includes:

- Synthetic map generator for configuration of no-fly zones, central area(s) and key locations
- An independent route validator to check geometric constraints

This supports reproducible testing, as well as early automated and systematic testing both before and after full system integration.

R2.1

Instrumentation will include timing probes either side of path calculation, logging of map and path complexity (e.g. number of no-fly zones, central area(s), nodes visited) and export of results for later analysis.

Scaffolding includes:

- A batch runner for executing multiple route calculations
- Synthetic map generator for increasing map complexity

This allows verification of meeting the timing threshold, and deeper diagnostic analysis of why.

Process and Risk

For this system, the Extreme Programming (XP) lifecycle process will be used for testing

R1.1

This requirement will be primarily placed in the unit testing stage of the XP process, as the module and its validation rules can be tested independently, and its logic is well-partitioned.

Risks in this requirement include:

- Late changes to JSON schema can invalidate existing tests
    - Mitigation: CI with automated regression testing to detect errors upon changes being made
- Validation rules may be underspecified, causing acceptance of invalid orders
    - Mitigation: explicitly specify validation partitions, create exhaustive unit test coverage for each rule

R1.2

This requirement will be primarily placed in the unit testing stage, as the path generation behaviour can be tested independently with synthetic data. It will be partially placed into acceptance testing regarding its integration with data endpoints for restaurants, map zones, etc.

Risks in this requirement include:

- Increasing pathing algorithm complexity causing errors in implementation
    - Mitigation: validate route independently after path generation
- Constructing diverse synthetic maps can take significant time, reducing test cases explored
    - Mitigation: create reusable map generators to parameterise map complexity and goal distance

R2.1

This requirement will be primarily placed in acceptance testing, as it is a system-wide quality requirement and cannot be meaningfully tested at a unit level.

Risks in this requirement include:

- The typical system load is undefined e.g. map complexity, start distance from goal, causing performance issues in real use
    - Mitigation: test performance over a wide range of map complexities and distances
- Variability in hardware causes variability in performance, invalidating performance testing
    - Mitigation: use a range of fixed benchmark environments in testing to ensure consistency