LO2 - Design and Implement Comprehensive Test Plans with Instrumented Code:

2.1 - Design of the Test Plan and Tests Performed

Purpose of the Test Plan

The purpose of the test plan is to design a testing strategy for the ILP Medical Drone Service simulation. The plan focuses on validating safety, correctness, and performance properties of the system while remaining feasible within limited project resources. Rather than attempting exhaustive coverage of all requirements, the plan deliberately selects a small set of high-risk, representative requirements to allow deeper analysis and more rigorous testing. The full test plan can be referred to at **'docs/testplan.pdf'**.

The selected requirements are:

- **SR1 - No-Fly Zone Compliance** (safety-critical)
- **CR1 - Positioning Accuracy** (functional correctness)
- **CR6 - Path Validity** (complete validation) (functional correctness, system integrity)
- **PR1 - Pathfinding Response Time** (performance)

These requirements span different requirement types and testing levels (unit, integration, system), enabling the test plan to exercise a wide range of testing techniques.

Tests Designed for Each Requirement:

SR1 - No-Fly Zone Compliance

SR1 is safety-critical and has direct regulatory implications. As a result, it is prioritised early in the lifecycle.

- **Static Verification**: Code inspection is used to verify that no-fly zone enforcement logic is correctly implemented before dynamic testing begins.
- **Integration Testing**: Dynamic tests focus on the interaction between pathfinding and no-fly zone validation logic.
- **Systematic Functional Techniques**:
    - Boundary value testing around no-fly zone edges
    - Negative testing to ensure paths that enter restricted zones are rejected
    - Representative combinatorial testing to cover key transitions without test explosion

Synthetic flight paths are generated using input scaffolding to exercise both typical and edge-case scenarios.

CR1 - Positioning Accuracy

CR1 addresses the correctness of core navigation computations and is well-suited to deterministic testing.

- **Unit Testing**:
    - Deterministic unit tests with synthetic coordinates
    - Boundary value analysis around the 0.00015° tolerance

    - Exact expected-value oracles for distance and proximity calculations
- **Integration Testing**:
    - Verification that positioning logic remains accurate when combined with navigation and control components
- **Limited Validation**:
    - Scenario-based validation in the simulator to confirm operational plausibility

The test plan follows a Test-Driven Development (TDD) approach, enabling early fault detection and rapid localisation of defects.

CR6 - Path Validity

CR6 ensures that all generated paths are fully valid, meaning they:

- Start and end at valid locations
- Do not violate no-fly zones (overlaps with SR1)
- Do not exceed maximum path length or move limits
- Remain within defined operational boundaries
- Are continuous and reachable (no gaps or illegal transitions)

Because CR6 cuts across multiple subsystems, it is primarily tested at integration and system level.

- **Integration Testing:**
    - Validation that generated paths respect all geometric, safety, and operational constraints
    - Tests combine pathfinding, safety validation, and configuration limits
- **Negative and Robustness Testing:**
    - Invalid paths (e.g. exceeding maxMoves, unreachable destinations, invalid start/end points) are intentionally generated and verified to be rejected
    - Confirms graceful failure and correct error reporting
- **System-Level End-to-End Testing:**
    - Full mission scenarios verify that valid paths are consistently produced across complete delivery workflows

CR6 acts as a correctness “umbrella” requirement, ensuring that compliance with SR1 and CR1 results in globally valid system behaviour rather than isolated correctness.

PR1 - Pathfinding Response Time

PR1 is evaluated at the system level, as meaningful performance measurements require a fully integrated environment.

- **System-Level Performance Testing**:
    - Deterministic microbenchmarks using synthetic workloads
    - Timed integration tests for representative pathfinding scenarios
- **Instrumentation Support**:
    - Lightweight timing counters measure algorithm response time
    - Logging supports later analysis of performance regressions

Because the workloads are synthetic, these tests provide indicative performance estimates rather than statistical guarantees across all real-world conditions.

Construction of the Test Plan

The test plan is constructed around an incremental, TDD-inspired lifecycle. Scaffolding and instrumentation are implemented early to support repeated automated testing. Unit and integration tests are executed as soon as corresponding components become available, while system-level testing is deferred until core functionality is stable.

This structure ensures that high-risk defects are detected early and that later testing phases are not dominated by avoidable low-level faults.

2.2 - Evaluation of the Quality of the Test Plan

The quality of the test plan is assessed in terms of coverage, risk-awareness, adaptability, and effectiveness in verifying the system's critical requirements.

Comprehensive Coverage:
The test plan covers multiple levels of testing: unit, integration, and system, to validate both functional and non-functional requirements. Unit tests target deterministic components in isolation, such as distance calculations (CR1: Positioning Accuracy). Integration tests verify subsystem interactions, including coordination between pathfinding and no-fly zone validation (SR1: No-Fly Zone Compliance) and order validation across modules. System tests assess end-to-end workflows and overall performance, including multi-order batch pathfinding scenarios and compliance with maximum path length and drone capacity restrictions (PR1: Pathfinding Response Time) and checking for global validity in generated outputs (CR6: Path Validity).

Edge Case and Negative Testing:
The plan explicitly tests boundary conditions and high-risk scenarios. For SR1, flight paths that intersect polygon vertices or no-fly zone edges are included. CR1 is validated using synthetic coordinates near boundary limits to confirm accuracy within 0.00015°. PR1 is tested under synthetic high workloads to measure response times for complex routing. Negative testing ensures invalid orders or unreachable locations are handled gracefully without causing system crashes.

Adaptability and Iterative Refinement:
The plan follows an incremental, TDD-inspired approach. As development introduced new features, such as multi-drop deliveries or temperature-restricted items, tests were adapted and additional synthetic scenarios created. This iterative refinement ensures that the plan remains aligned with evolving requirements and supports early defect detection.

Strengths:

- Focused testing on high-priority requirements allows early identification of critical defects.
- Multi-level testing ensures both isolated logic and end-to-end system behaviour are validated.
- Boundary and negative testing increase confidence in safety-critical and correctness-related functionality.
- Flexible, iterative design allows the plan to evolve alongside system development.

Limitations:

- **Synthetic/Generated Workloads:** Performance and pathfinding tests rely on simulated data, which may not fully replicate real-world conditions, such as dynamic obstacles or unexpected order distributions
- **Limited Concurrency and Stress Testing:** The current plan does not extensively cover high-load or multi-drone simultaneous operations, potentially missing performance bottlenecks
- **External Dependency Simulation:** Mocked services (like the ILP REST API) may not capture real-world delays, errors, or edge-case responses, limiting the assessment of integration robustness
- **Security Coverage:** Explicit security testing, such as validation against malicious inputs or data integrity breaches, is limited in scope
- **Resource and Time Constraints:** Due to prioritisation, only a subset of requirements are deeply tested, leaving less critical features with reduced verification coverage

Overall, the test plan provides strong coverage for the system's most critical requirements and includes mechanisms for early defect detection, edge-case testing, and iterative adaptation. However, reliance on synthetic data, limited concurrency testing, and reduced security evaluation are important limitations that should be addressed in future testing phases to improve confidence in real-world deployment.

2.3 - Instrumentation of the code

Instrumentation was implemented to support debugging, verification, and performance measurement during testing. It was tailored for the selected high-priority requirements (SR1, CR1, PR1) to ensure accurate, traceable test results.

Logging and Diagnostic Output:

- **Order Validation Service:** Outputs diagnostic messages including validation failures, error codes, and status of individual orders. This supports unit and integration testing, helping identify incorrect handling of invalid or edge-case orders.
- **Position Service and Pathfinding Modules:** Logs computed positions, path waypoints, and distances to ensure CR1 (Positioning Accuracy) and SR1 (No-Fly Zone Compliance) are enforced. Waypoints near restricted zones are logged to verify that no illegal paths are generated.

Assertions:

- Assertions are applied in key methods to enforce expected behaviour, e.g., ensuring the total weight of orders does not exceed drone capacity and path length does not exceed maxMoves.
- CR1-specific assertions validate that calculated positions are within the 0.00015° tolerance of intended delivery points.

Error Handling and Reporting:

- Instrumentation ensures meaningful, actionable error messages for both internal debugging and API consumers (UR1: API Error Message Clarity).

- For SR1, any violation of no-fly zones triggers logging with coordinates, timestamp, and violated zone, enabling rapid identification and debugging of boundary or edge cases.

Performance Counters and Metrics:

- For PR1, timers measure pathfinding duration under varied synthetic workloads to assess compliance with response time requirements
- Repeated runs log statistics (average, maximum, and variance) to identify potential performance bottlenecks or regressions

2.4 – Evaluation of the Instrumentation:

The instrumentation implemented was generally effective in supporting test execution, debugging, and validation, but there are both strengths and areas for improvement.

Strengths:

- Facilitated early detection of defects in unit and integration tests, particularly for pathfinding, position calculations, and order validation
- Enabled precise verification of high-risk requirements, including SR1: No-Fly Zone Compliance, CR1: Positioning Accuracy, and PR1: Pathfinding Response Time
- Performance metrics were collected using synthetic workloads to help detect potential performance regressions
- Detailed logging improved failure traceability, enabling faster identification of defects

Limitations and Areas for Improvement:

- **Limited Real-World Representativeness:** Synthetic test scenarios may not fully reflect real-world conditions. Future work could improve realism by logging and analysing more varied or unpredictable scenarios
- **Performance Overhead:** Extensive logging and assertions slightly increase execution time, which could affect measured response times. Sampling or selective logging may mitigate this
- **Enhanced Metrics Needed:** Current instrumentation focuses on correctness. Additional metrics such as resource usage (memory, CPU) or latency would improve performance analysis for PR1
- **Extended Error Context:** While error messages provide coordinates and status, further context, such as cumulative path statistics or prior decision points, would improve debugging efficiency

The instrumentation is sufficient for verifying functional, safety, and performance requirements within the constraints of the current simulation. Early scaffolding and logging enable iterative testing and validation while highlighting defects before system-level integration. As development progresses, enhancements to performance monitoring, logging, and real-world scenario simulation would further strengthen instrumentation usefulness.