Learning Outcome 3 – Apply a Wide Variety of Testing Techniques and Compute Test Coverage and Yield

3.1 Range of Techniques

A diverse set of testing techniques was applied to ensure correctness, robustness, and resilience across the drone delivery system. These included:

- Unit Testing: Core modules such as DroneAvailabilityService, PathfindingService, ValidationService, and RegionService were tested using JUnit 5. These tests validated isolated logic, including edge cases in filtering logic, path computation, and region validation.
- Integration Testing: Spring Boot's MockMvc was used to verify that controllers and services worked together seamlessly. For instance, integration tests on /calcDeliveryPath ensured that validation, region checking, and delivery planning were coordinated correctly.
- Boundary Testing: Fields such as maxMoves, payload capacity, and coordinate ranges were tested for minimum, maximum, and just-inside/outside thresholds.
- Negative Testing: Invalid delivery requests (e.g., missing fields, malformed coordinates, empty region lists) were tested to confirm graceful error handling.
- Exploratory Testing: Used to detect edge behaviors such as delivery requests that fall on the edge of restricted zones or dispatches using all available drones simultaneously.
- Regression Testing: After each functional update, full test suites were rerun to confirm the absence of regressions, ensuring consistent system behavior over time.

3.2 Evaluation Criteria for the Adequacy of the Testing

Adequacy was evaluated using structural and behavioral criteria:

- Coverage Metrics: JaCoCo instrumentation provided instruction and branch coverage reports. The final test suite achieved ~60% instruction coverage and 38% branch coverage. While not exhaustive, this confirmed broad logic execution across the application.
- Risk-Based Targeting: High-risk features (e.g., no-fly zone avoidance, TSP delivery planning, drone availability filters) were prioritized to ensure correct operation under constraints.
- Service-Specific Metrics: Services like RegionService, PathfindingService, and DroneAvailabilityService had higher coverage due to their isolated logic, complexity, and critical role in route correctness and dispatch filtering.

The adequacy of testing was reinforced by the system's ability to reject malformed inputs, pass high-volume test runs, and maintain deterministic output through functional changes.

3.3 Results of Testing

All 143 tests passed successfully in the final build. These included:

- Unit tests across core modules
- Integration tests on all major REST endpoints
- Edge case and invalid input coverage

JaCoCo reports showed that modules like DroneAvailabilityService had full coverage of basic filters, while RegionService had minor gaps in multi-condition handling. Pathfinding logic was robustly exercised, particularly through tests like calculatePath_aroundRestriction and calculatePath_multipleRestrictions, which validated A* algorithm behavior under no-fly zone constraints.

Behavioral correctness was validated via assertion-based verification and inspection of resulting JSON payloads. REST responses were validated for both 200 OK and 4xx errors under malformed inputs.

3.4 Evaluation of the Results

The testing process revealed several strengths:

- **Strong Fault Rejection:** Invalid inputs (e.g., malformed GeoJSON, out-of-bounds coordinates) were reliably rejected.
- **High Test Stability:** Tests produced deterministic results across runs, indicating a stable logic base.
- **Service-Level Confidence:** High-frequency components like availability checking and region validation were thoroughly tested with consistent pass rates.

Limitations:

- Instruction/branch coverage remained below target (goal was 70%/50%)
- No concurrency testing was performed
- Mutation or property-based testing not implemented due to scope

Despite these limits, the testing outcomes demonstrated that the software meets correctness, resilience, and input validation goals. Structural coverage and service-level test depth affirmed the success of applied testing techniques.

This document details the range, adequacy, and results of testing conducted on the ILP drone system. It reflects a comprehensive use of automated testing techniques, validates coverage depth, and provides rationale for gaps and strengths. Supporting evidence is available in the GitHub repository and Evidence Document.