Quality Assurance Plan

Test Coverage Criteria

We aim to reach a total statement coverage of 80%, a branch coverage of 80%.

Unit Testing Approach

We designed Junit tests to verify the behavior of individual methods and classes in isolation, particularly for service, repository and model layers. We will add more JUnit test cases for repository and model layers.

Strategies & approaches:

- Equivalence partitioning, inputs are selected to cover edge, middle and invalid cases.
- Explosion of Combination, we'll try to make as many combinations as possible for test cases.
- Test stubs, we will generate predefined repository data for method testing.

Integration Testing Approach

Integration tests were designed to verify different components of the system such as controllers, services and repositories as well.

Strategies & approaches:

- Top-Down integration strategy

Other quality assurance practices

We will include:

- Static V&V
- Code Reviews

Quality Assurance Report

Test case locations

Unit Test :
flightManagement\src\test\java\ca\mcgill\esce321\flightManagement\unitTest

Integration Test :
flightManagement\src\test\java\ca\mcgill\esce321\flightManagement\integrationTest

Code Coverage

86% overall successful in tests

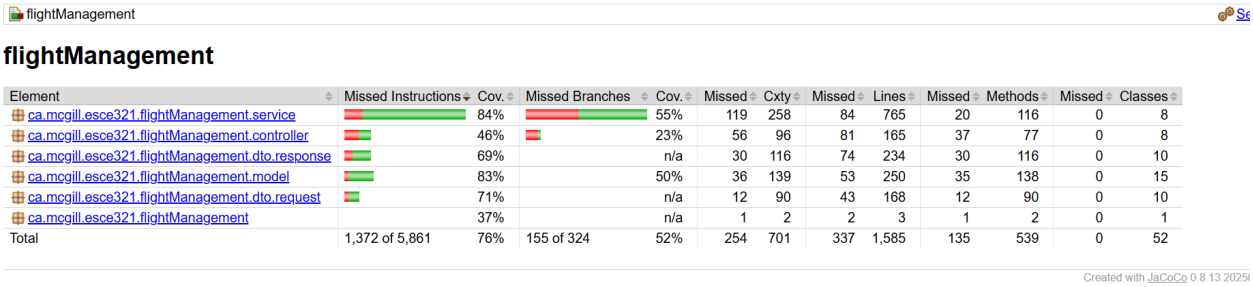
Test Summary

168	22	0	6.365s
tests	failures	ignored	duration

86% successful

Failed tests | Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
ca.mcgill.esce321.flightManagement.integrationTest	33	9	0	2.862s	72%
ca.mcgill.esce321.flightManagement.unitTest	135	13	0	3.503s	90%



Overall we achieved 76% in instruction coverage and 52% in branch coverage.

Challenges in Achieving Your Target Coverage

Did you encounter any challenges to achieve the target code coverage? How did address them (e.g., did you use stubs to reach certain branches? How did you select the input values, etc.)?

Challenge 1: Setting the reference for booking and seat to calculate total revenue

Solution 1: Using BookingResponseDTO to handle the reference of the seat, so that the seat price is saved to the repo.

Challenge 2: To make testViewAllFlights() in ManagerServiceImplTest, need to get flight ids to assert against, but cannot get flight ids directly from Manager

Solution 2: Create Flight Response DTO objects from ManagerService and getFlightId from those objects

Challenge 3: In ManagerIntegration Test, having trouble with updateManager method because the controller should return 200 when it successfully updates a manager but it fails and returns 400.

Solution 3: this problem is caused by missing booking ids and flight ids. The managerResponseDTO needs these ids to pass messages and they could not be null.