|  |
| --- |
| José Coronil Álvarez (joscoralv@alum.us.es)  Miguel García Vizcaíno (miggarviz@alum.us.es)  Javier Ignacio Milá de la Roca Dos Santos (javmildos@alum.us.es)  Emilio Manuel Vázquez Cruz (emivazcru@alum.us.es)  Group C1.028  5-16-2025 |

|  |
| --- |
| Acme AirNav Solutions, Inc. |
| **Testing Report** |
| https://github.com/Emilio-115/DP2-Acme-ANS |



Contents

[Executive Summary 2](#_Toc199148322)

[Revision Table 2](#_Toc199148323)

[Introduction 3](#_Toc199148324)

[Content 3](#_Toc199148325)

[Tests Developed 3](#_Toc199148326)

[Flight Assignments 3](#_Toc199148327)

[Activity Logs 4](#_Toc199148328)

[Test Coverage 5](#_Toc199148329)

[Flight Assignments 5](#_Toc199148330)

[Activity Logs 6](#_Toc199148331)

[Performance Analysis 6](#_Toc199148332)

[Before Indices 7](#_Toc199148333)

[After indices 8](#_Toc199148334)

[Comparison 9](#_Toc199148335)

[Conclusions 9](#_Toc199148336)

[Bibliography 9](#_Toc199148337)

# Executive Summary

This report documents the tests implemented for Student 3’s tasks. All required test cases were created and executed, covering both normal and unauthorized scenarios. Most features functioned as expected, and the few bugs identified were fixed. All reachable lines of code were covered by the tests. Performance was analyzed before and after adding indices, showing a slight slowdown.

# Revision Table

|  |  |  |
| --- | --- | --- |
| Revision number | Date | Description |
| 1 | 26/05/2025 | Initial version |

# Introduction

This document is the testing report for Student 1, tests cases, coverage and performance will be

# Content

Following the guidelines provided, safe and hack test cases were recorded for airline manager features including everything related to flights and legs management aiming for a high coverage, there is a performance test and a comparison between using or not indices.

## Tests Cases

The following is a list of the test cases implemented, grouped by feature:

### Flight

* **List-show.safe**:  
  These test checks that an airline manager can list only their flights, can show a single flight in detail it was tested for published and draft flights. Because of the simple nature of list where we don’t expose anything user related and the list of flights shown is based on the current logged in user there are no hack cases for list. No bugs were found.
* **show.hack**:  
  These tests were made to ensure safety on the show feature, making sure a airline manager can only see information about their own flights, there are tests like trying to access to others manager fight or non-existing flights. No bugs were found.
* **create.safe**, **update.safe**, **delete.safe**, **publish.safe**:  
  These tests ensure you can properly use all the airline manager legal features, this includes all validation related to flights and their respective legs, like publishing without legs or with not published legs, leaving all fields empty to ensure the backend did not fail, also tested all these features with big set of permutations ensuring expected behavior on every one of them. Some bugs were found like the system allowing flights with no legs or the system allowing the deletion of flights with published legs.
* **create.hack**:  
  This tests that, only airline managers can create flights, and that changing the id of the request to an existing flight does not modify it. This last test case was found during testing and solved effectively.
* **update.hack**, **delete.hack**, **publish.hack**:  
  This are test cases for updating , deleting and publishing flights under non legal situation, they are grouped because of their similarities. All of them check that an airline manager can’t perform these actions for others manager flights, non-existing flights, and their own flights already published. No bugs where found.

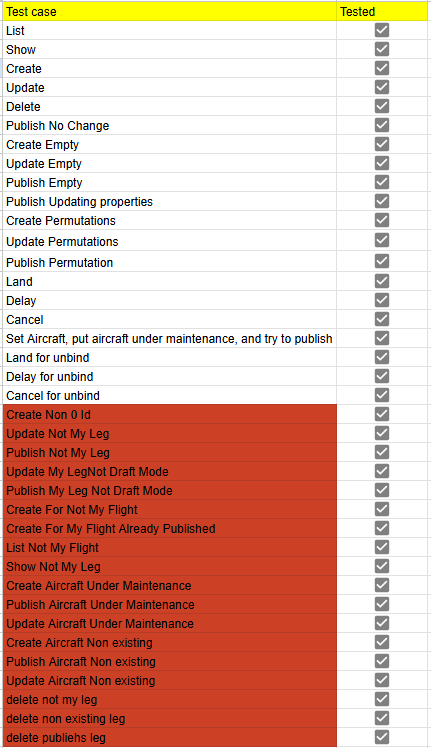
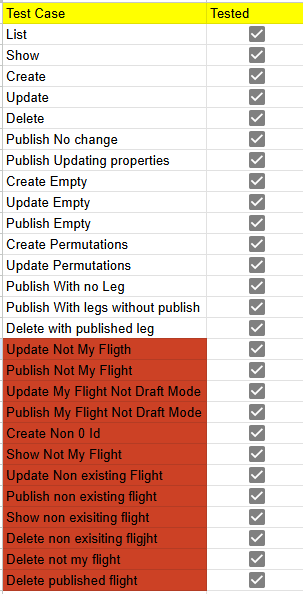
### Legs

* **list-show.safe**   
  These tests check that an airline manager can list legs of their flights, can show a single leg in detail it was tested for published and draft legs and published and draft flights. No bugs were found.
* **List-show.hack**:  
  These tests were made to ensure safety on the show and list features, making sure a airline manager can only see information about their own legs, there are tests like trying to access to others manager legs, others managers flights list of legs or non-existing legs. No bugs were found.
* **create.safe**, **update.safe**, **delete.safe**, **publish.safe**:  
  These tests ensure you can properly use all the airline manager legal features, this includes all validation related to legs, creating a leg in the past or leaving all fields empty to ensure the backend did not fail, also tested all these features with big set of permutations ensuring expected behavior on every one of them. Some changes were made to satisfy all requirements, the hardest one was about aircraft where an aircraft could be set to under maintenance while creating a leg with that same aircraft, the solution was setting it to null in frontend and making sure the airline manager solved it by changing the aircraft.
* **Create.hack, update.hack**, **delete.hack**, **publish.hack**:  
  This tests ensured the airline manager features related with legs were safe by checking different cases that should fail like creating legs for others manger flights, creating for publish flights or updating, deleting and publishing for others managers legs, non-existing legs and published legs. We also tested navigation hacking setting non-existing airports, non-existing aircrafts and aircrafts under maintenance. No bugs were found.
* **Extra.safe:**

These are extra tests made for improving test coverage, with specific cases like showing a past flight with a currently non available flight

## Test methodology

The test methodology followed was the one provided, but for making the process easier it was produced a test case checklist, where red tests were hacking tests, so we could have a record of what tests where done, what tests were not and what to rerecord when it was necessary, this helped not leaving behind any test case.



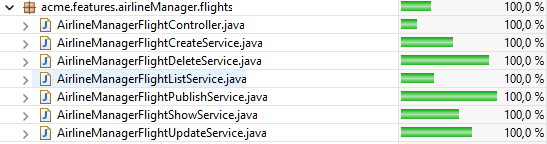
## Test Coverage

The following is a summary of the coverage achieved for all relevant files

### Flight

* **Validator** **(88,2%):**All the missing coverage comes from safety null checks that are never actually executed due to being an impossible branch of execution, all domain logic is 100% covered.
* **Services of features (100%)**

100% of coverage was successfully achieved in the all the services of the flight features





### Legs

* **Validator (92,7%):**  
  All the missing coverage comes from safety null checks that are never actually executed due to being an impossible branch of execution, all domain logic is 100% covered.
* **Controller**, **Delete Service**, **Update Service**, **List Service**, **Show Service**, **Publish Service (100%):**  
  100% of coverage was successfully achieved in these services of the leg features
* **Create Service (98,6%):**  
  The remaining coverage comes from the branch where it can’t find the flight in the database during the load, this is not possible because of it being checked before in the authorize.
* **Land Service, Delay Service, Cancel Service(99,2%)**

These services are not part of the requirements but it was the solution I got to the problem with the state of the leg, these have null checks that never fail so that’s the remaining coverage





## Performance Analysis

The performance of the project was analyzed according to the specifications both before and after indices were created for the relevant tables.

The times were recorded executing only the tests regarding flight crew member features. The exact same tests were run with both; the only difference is the indexing of the tables. The specific results can be seen on the Excel files in this folder.

### Before Indices

Before the indices, the testing resulted in a 95% confidence interval from 8.99ms to 10.36ms and a mean of 9.58ms. The costliest operation was cancelling a flight assignment, which in average took 33.39ms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Response time* | |  |  |  |  |
|  |  |  | *Interval (ms)* | 8.991419 | 10.3647 |
| Mean | 9.678059 |  | *Interval (s)* | 0.008991 | 0.010365 |
| Standard Error | 0.349847 |  |  |  |  |
| Median | 6.9195 |  |  |  |  |
| Mode | 2.1847 |  |  |  |  |
| Standard Deviation | 10.33677 |  |  |  |  |
| Sample Variance | 106.8488 |  |  |  |  |
| Kurtosis | 16.87884 |  |  |  |  |
| Skewness | 3.076939 |  |  |  |  |
| Range | 104.3643 |  |  |  |  |
| Minimum | 1.563 |  |  |  |  |
| Maximum | 105.9273 |  |  |  |  |
| Sum | 8448.946 |  |  |  |  |
| Count | 873 |  |  |  |  |
| Confidence Level (95.0%) | 0.68664 |  |  |  |  |

### After indices

After the indices, the interval was from 10.25ms to 11.95ms, with a mean of 10.99ms. The costliest operation was confirming a flight assignment, which in average took 36.72ms. Cancelling a flight assignment was a close second, taking 36.21ms in average.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Response time* | |  |  |  |  |
|  |  |  | *Interval (ms)* | 10.24744 | 11.95314 |
| Mean | 11.10029 |  | *Interval (s)* | 0.010247 | 0.011953 |
| Standard Error | 0.434532 |  |  |  |  |
| Median | 7.8506 |  |  |  |  |
| Mode | 2.3173 |  |  |  |  |
| Standard Deviation | 12.83892 |  |  |  |  |
| Sample Variance | 164.8379 |  |  |  |  |
| Kurtosis | 24.11654 |  |  |  |  |
| Skewness | 3.767839 |  |  |  |  |
| Range | 134.898 |  |  |  |  |
| Minimum | 1.5918 |  |  |  |  |
| Maximum | 136.4898 |  |  |  |  |
| Sum | 9690.553 |  |  |  |  |
| Count | 873 |  |  |  |  |
| Confidence Level (95.0%) | 0.85285 |  |  |  |  |

### Comparison

The two sample z-test for means resulted in a P(Z<=z) two-tail value of 0.0121, which is below significantly below the alpha value for the 95% confidence level (0.05). Thus, the means can be compared with confidence. And, since the mean response time is higher for the tests performed with indices, it can be concluded that adding the indices decreased performance.

This might be because of the small scale of the tests and the sample database resulting in the overhead cost of indices not being offset by the performance gains on querying the database.

|  |  |  |
| --- | --- | --- |
| z-Test: Two Sample for Means |  |  |
|  |  |  |
|  | *Before indices* | *After indices* |
| Mean | 9.588754702 | 10.98820539 |
| Known Variance | 106.84878 | 164.8379173 |
| Observations | 872 | 872 |
| Hypothesized Mean Difference | 0 |  |
| z | -2.507157719 |  |
| P(Z<=z) one-tail | 0.00608532 |  |
| z Critical one-tail | 1.644853627 |  |
| P(Z<=z) two-tail | 0.012170639 |  |
| z Critical two-tail | 1.959963985 |  |

# Conclusions

All required tests were written and executed successfully. The features behaved correctly in both valid and invalid scenarios. The tests helped identify bugs, which were subsequently fixed. Test coverage was achieved for all reachable lines in validators and services. Performance tests showed a small slowdown after indexing, likely due to the small database size not benefiting from optimization.

# Bibliography

Intentionally blank.