

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

## Testing report D04



Degree in Computer Engineering – Software Engineering

Design and Testing II.

Academic year 2024 – 2025

Laboratory group C1.005		
Alphabetised authors	Role	Role description
Artero Bellido Manuel – manartbel@alum.us.es	Tester	Does formal testing and writes reports
Calderón Rodríguez, Manuel María -mancalrod@alum.us.es	Manager	Makes plans, creates and supervises tasks, initialises the repository and writes reports
González Benito, Claudio – clagonben@alum.us.es	Developer	Creates development configuration, customises the starter, implements features, does informal testing and writes reports
Márquez Gutiérrez, José Manuel – josmargut@alum.us.es	Operator	Creates deployment configurations, deploys the application, keeps the application running and writes reports
Ramos Vargas, Alba – albramvar1@alum.us.es	Developer	Creates development configuration, customises the starter, implements features, does informal testing and writes reports

## Versioning

Date	Version	Description
10/05/2025	v1.0.0	Creation of the report

## Table of contents

Versioning .....	2
Executive summary.....	<b>Error! Bookmark not defined.</b>
Introduction.....	<b>Error! Bookmark not defined.</b>
Contents .....	<b>Error! Bookmark not defined.</b>
Managerial Requirement 2 .....	<b>Error! Bookmark not defined.</b>
Problems.....	<b>Error! Bookmark not defined.</b>
Conclusions .....	<b>Error! Bookmark not defined.</b>
Validation .....	<b>Error! Bookmark not defined.</b>
Conclusions .....	<b>Error! Bookmark not defined.</b>
References .....	<b>Error! Bookmark not defined.</b>

## Executive Summary

The purpose of this report is to study the testing done on the project, specifying test cases, their expected and real results, the coverage gotten and the comparison of performance once indexes were added.

Due to the formatting decisions of this report, a PDF version will be added to ensure that the report is readable.

## Introduction

In this report we will specify the test cases produced, that is, list, show, create, update, delete and publish for both feature modules, flight assignment (usually referred to as simply 'assignment') and activity log (usually referred to as 'log').

I will expose what each test case is supposed to test in general and in particular, specifying validation constraints and expected exception. In addition, the coverage provided by these tests will be discussed.

Finally, the second chapter of this report will study the performance, considering three distinct cases that were considering worth studying.

## Functional testing

The generic cases that will be considered are:

- List: the user must receive a list of the entity in question.
- Show: the user must be shown all relevant information of the entity, this includes all properties and a singular property of any linked entity that's considered the most representative. In this case, the property chosen of legs is their flight code and the property of crew members are their identifier code, as it is an alternative primary key for both entities.
- Create: the user must be able to create an entity following some constraints.
- Update: the user must be able to update any preexisting entity following the same constraints as the create proceeding.
- Delete: the user must be able to delete the entity if the requirements allow it.
- Publish: the user must be able to publish the entity if the requirements allow it.

In addition, there will be two kind of tests considered, 'safe' cases and 'hack' cases. The former comprise only 'legal' actions and cannot ever get any exception; the latter's purpose is to get exception, specifically 500 Access not authorized exception, in their majority.

## Specific Test cases

### Flight Assignment

Test case		Expected result	Bug detection
List	SAFE	Show three different lists: <ul style="list-style-type: none"><li>- All assignments</li><li>- Future assignments</li><li>- Past assignments</li></ul> Future and past assignments will be classified considering the leg's scheduled departure and arrival. For users registered as lead attendants of a specific leg, all assignments of that leg will be shown, regardless of the crew member assigned. For users not registered as lead attendants, only their assignment will be shown.	No bugs detected
	HACK	When trying to access the list by another realm, there should be a not authorized exception. Seeing the list of another user can never be done as the system uses the user's id to give the list of assignments, the user does not provide any hackable information other than the user account, what is out of our purvey.	No bugs detected
Show	SAFE	Show a user's assignment, accessing it through the lists specified above. Lead attendants are shown all assignments of the leg they are in charge of and can access the edit buttons (delete, update and publish). Not lead attendants will be shown only their assignment and will have no buttons to interact with, as they are not authorized to do so.	No bugs detected
	HACK	The system should provide a 500 Access not authorized exception in the following cases: <ul style="list-style-type: none"><li>- A user tries to access an assignment that is not theirs</li><li>- A user tries to access an object that is not theirs, regardless of the class</li><li>- A user does not specify an id to be shown</li><li>- A user tries to be shown an object with an id not compatible (i.e. a string id)</li></ul>	No bugs detected

Create	SAFE	<p>A user can create an assignment in these two specific cases:</p> <ul style="list-style-type: none"> <li>- The user is the lead attendant of the leg given</li> <li>- The user is creating an assignment with a leg with no assignments and a lead attendant duty<sup>1</sup></li> </ul> <p>The validation considered should be:</p> <ul style="list-style-type: none"> <li>- Only the remarks value can be null</li> <li>- There can be only one lead attendant, pilot and co-pilot by leg</li> <li>- The last update should be a past moment before the leg's scheduled departure</li> <li>- The leg given must be from the same airline the user is an employee of</li> <li>- The assignee must be from the same airline the user is and must be available</li> </ul>	Due to the order of validation, some constraint messages are difficult to be shown. However, all cases are covered.
	HACK	<p>There should be a invalid value validation message for the following hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a Duty that does not exist.</li> <li>- Selecting a leg or a crew member with a not numeric id.</li> </ul> <p>There should be a 500 Access not authorized message for the following possible hacking attempts:</p> <ul style="list-style-type: none"> <li>- A user not registered as a crew member tries to access the create form</li> <li>- Selecting a leg or a crew member with a not provided id.</li> </ul>	No bugs detected
Update	SAFE	<p>A user registered as the leg attendant of the leg can update the assignments with the following constraints and validations:</p> <ul style="list-style-type: none"> <li>- Only the remarks value can be null</li> <li>- There can be only one lead attendant, pilot and co-pilot by leg</li> <li>- The last update should be a past moment before the leg's scheduled departure</li> <li>- The leg cannot be changed</li> <li>- The assignee must be from the same airline the user is and must be available</li> </ul>	When giving an invalid value and attempting to update the entity, the system returns a read-only form, although the unbind method specifies read-only to false.

<sup>1</sup> For more information, see the analysis report

Delete	HACK	<p>There should be an invalid value validation message for the following hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a Duty or Status that does not exist.</li> <li>- Selecting a leg or a crew member with a not numeric id.</li> </ul> <p>There should be a 500 Access not authorized message for the following possible hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a leg or a crew member with a not provided id.</li> <li>- A non lead attendant tries to update their assignment</li> <li>- A user tries to access an assignment that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to update</li> <li>- A user tries to update an object with id not compatible (i.e. a string id)</li> </ul>	No bugs detected
	SAFE	A lead attendant can delete any assignments from the specific leg. In case they try to delete their own assignment as lead attendant, they will receive a validation message telling them that they are not as of that moment allowed because there are other assignments for that leg.	No bugs detected
	HACK	<p>The system should provide a 500 Access not authorized exception in the following cases:</p> <ul style="list-style-type: none"> <li>- A non lead attendant tries to delete their assignment</li> <li>- A lead attendant tries to delete a published assignment</li> <li>- A user tries to access an assignment that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to delete</li> <li>- A user tries to delete an object with an id not compatible (i.e. a string id)</li> </ul>	No bugs detected
Publish	SAFE	A lead attendant can publish any assignment for a leg that is already published.	No bugs detected

HACK

The system should provide a 500 Access not authorized exception in the following cases:

- A non lead attendant tries to publish their assignment
- A lead attendant tries to publish a published assignment
- A user tries to access an assignment that is not theirs
- A user tries to access an object that is not theirs, regardless of the class
- A user does not specify an id to publish
- A user tries to publish an object with an id not compatible (i.e. a string id)

No bugs detected



## Activity Log

Test case		Expected result	Bug detection
List	SAFE	Show one list with all activity logs registered by the user.	No bugs detected
	HACK	When trying to access the list by another realm, there should be a not authorized exception.	No bugs detected
Show	SAFE	Show a user's logs, accessing it through the list specified above. The action buttons must be shown only if the action in question is considered 'legal'.	No bugs detected
	HACK	<p>The system should provide a 500 Access not authorized exception in the following cases:</p> <ul style="list-style-type: none"> <li>- A user tries to access a log that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to be shown</li> <li>- A user tries to be shown an object with an id not compatible (i.e. a string id)</li> </ul>	No bugs detected
Create	SAFE	<p>A user can create a log considering the following validation constraints:</p> <ul style="list-style-type: none"> <li>- No value can be null</li> <li>- The incident type must be below 50 characters</li> <li>- The description must be below 255 characters</li> <li>- The registration moment must be in the past but after the leg scheduled arrival</li> <li>- The leg given must be from an assignment the user has</li> <li>- The system must find the assignment and link it to the log, considering the leg given and the registered user</li> </ul>	Because of the constraint of the legs and how this entity depends on the leg's scheduled arrival, no logs can be created in this system as of now <sup>2</sup>

<sup>2</sup> For more information, check the analysis report

Update	HACK	<p>There should be an invalid value validation message for the following hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a leg with a not numeric id.</li> </ul> <p>There should be a 500 Access not authorized message for the following possible hacking attempts:</p> <ul style="list-style-type: none"> <li>- A user not registered as a crew member tries to access the create form</li> <li>- Selecting a leg with a not provided id.</li> </ul>	No bugs detected
	SAFE	<p>A user registered as a crew member can update their logs with the following validations:</p> <ul style="list-style-type: none"> <li>- No value can be null</li> <li>- The incident type must be below 50 characters</li> <li>- The description must be below 255 characters</li> <li>- The registration moment must be in the past but after the leg scheduled arrival</li> <li>- The leg given must be from an assignment the user has</li> <li>- The system must find the assignment and link it to the log, considering the leg given and the registered user</li> </ul>	As explained in the create bugs, no log in the system can actually be updated.
	HACK	<p>There should be an invalid value validation message for the following hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a leg with a not numeric id.</li> </ul> <p>There should be a 500 Access not authorized message for the following possible hacking attempts:</p> <ul style="list-style-type: none"> <li>- Selecting a leg with a not provided id.</li> <li>- The user is trying to update a published log.</li> <li>- A user tries to access a log that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to be updated</li> <li>- A user tries to update an object with id not compatible (i.e. a string id)</li> </ul>	No bugs detected
Delete	SAFE	A crew member can delete any of their logs	No bugs detected

Publish	HACK	<p>The system should provide a 500 Access not authorized exception in the following cases:</p> <ul style="list-style-type: none"> <li>- A lead attendant tries to delete a published log</li> <li>- A user tries to access a log that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to be deleted</li> <li>- A user tries to delete an object with an id not compatible (i.e. a string id)</li> </ul>	No bugs detected
	SAFE	A crew member can publish any of their logs for an assignment that is already published.	No bugs detected
	HACK	<p>The system should provide a 500 Access not authorized exception in the following cases:</p> <ul style="list-style-type: none"> <li>- A crew member tries to publish a published log</li> <li>- A user tries to access a log that is not theirs</li> <li>- A user tries to access an object that is not theirs, regardless of the class</li> <li>- A user does not specify an id to publish</li> <li>- A user tries to publish an object with an id not compatible (i.e. a string id)</li> </ul>	No bugs detected

## Coverage

The coverage for the flight assignment features is as follows:

























acme.features.flight_crew.flight_assignment		82.4 %	2,067	440	2,507
CrewFlightAssignmentCreateService.java		73.4 %	515	187	702
CrewFlightAssignmentCreateService		73.4 %	515	187	702
• validate(FlightAssignment)		53.5 %	166	144	310
• authorise()		100.0 %	92	0	92
• bind(FlightAssignment)		100.0 %	30	0	30
• load()		100.0 %	13	0	13
• perform(FlightAssignment)		100.0 %	6	0	6
• unbind(FlightAssignment)		100.0 %	137	0	137
CrewFlightAssignmentUpdateService.java		73.7 %	497	177	674
CrewFlightAssignmentUpdateService		73.7 %	497	177	674
• validate(FlightAssignment)		49.2 %	119	123	242
• authorise()		95.1 %	136	7	143
• perform(FlightAssignment)		0.0 %	0	6	6
• bind(FlightAssignment)		100.0 %	30	0	30
• load()		100.0 %	18	0	18
• unbind(FlightAssignment)		100.0 %	135	0	135
> CrewFlightAssignmentListCompletedService.java		55.5 %	81	65	146
> CrewFlightAssignmentPublishService.java		92.0 %	126	11	137
> CrewFlightAssignmentController.java		100.0 %	47	0	47
> CrewFlightAssignmentDeleteService.java		100.0 %	206	0	206
> CrewFlightAssignmentListPlannedService.java		100.0 %	146	0	146
> CrewFlightAssignmentListService.java		100.0 %	134	0	134
> CrewFlightAssignmentShowService.java		100.0 %	315	0	315

Figure 1. Flight Assignment features coverage

The objective was to manage a 90% of coverage. Although most modules have a coverage of 100%, the validate methods for the create and update methods are really low, even below 50% in the case of the update module, as can be seen in the image above.

acme.features.flight_crew.activity_log	77.2 %	1,097	324	1,421
CrewActivityLogUpdateService.java	71.1 %	300	122	422
CrewActivityLogUpdateService	71.1 %	300	122	422
validate(ActivityLog)	36.8 %	67	115	182
perform(ActivityLog)	0.0 %	0	6	6
authorise()	100.0 %	108	0	108
bind(ActivityLog)	100.0 %	26	0	26
load()	100.0 %	18	0	18
unbind(ActivityLog)	100.0 %	64	0	64
CrewActivityLogCreateService.java	68.8 %	236	107	343
CrewActivityLogCreateService	68.8 %	236	107	343
validate(ActivityLog)	36.7 %	58	100	158
perform(ActivityLog)	0.0 %	0	6	6
authorise()	100.0 %	51	0	51
bind(ActivityLog)	100.0 %	26	0	26
load()	100.0 %	13	0	13
unbind(ActivityLog)	100.0 %	71	0	71
CrewActivityLogPublishService.java	63.9 %	117	66	183
CrewActivityLogDeleteService.java	79.7 %	110	28	138
CrewActivityLogShowService.java	99.4 %	174	1	175
CrewActivityLogController.java	100.0 %	35	0	35
CrewActivityLogListService.java	100.0 %	125	0	125

Figure 2. Activity Log feature coverage

In this module, the difference in coverage is even more stark than in the flight assignment module, managing below 37% in the validate methods of both create and update services.

This low coverage can be both a result of test cases not exhaustive enough, however, I, as the developer, am more inclined to attribute these low values to an over-protectiveness of the code, having created consideration for cases that, as the tests show, can be excessive but that I, as the one who has written the code, consider necessary when statically revising the validations.

## Performance testing

Although the annexes specify that the performance must be compared between two computers, I decided to consider two cases to study.

The first case is the difference in performance of the same computer considering if it is charging or if it is not. I consider this case interesting because I generally do not like to work with the computer charging and I did notice the difference while working in this project, so I consider it interesting enough to analyze. The second case is the indexes case, necessary for the completion of the mandatory requirements of the projects.

## First study case: charging vs not charging

For this case there are the following values:

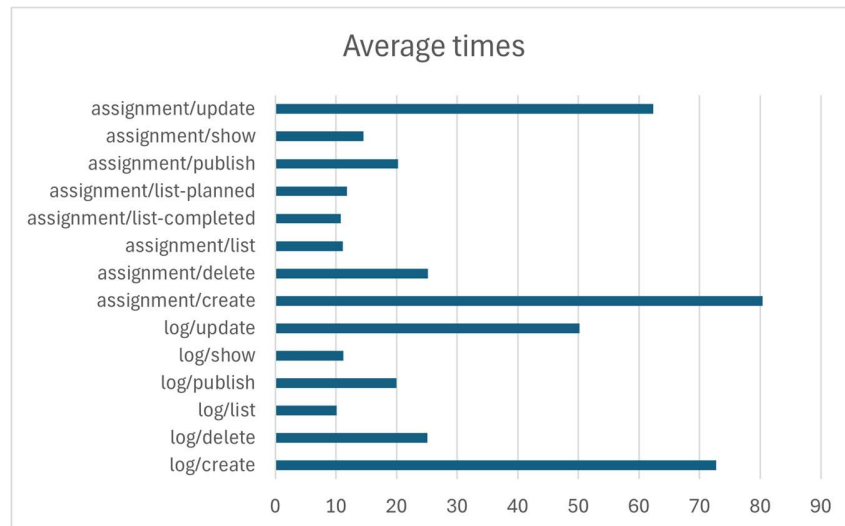


Figure 3. Average replay times

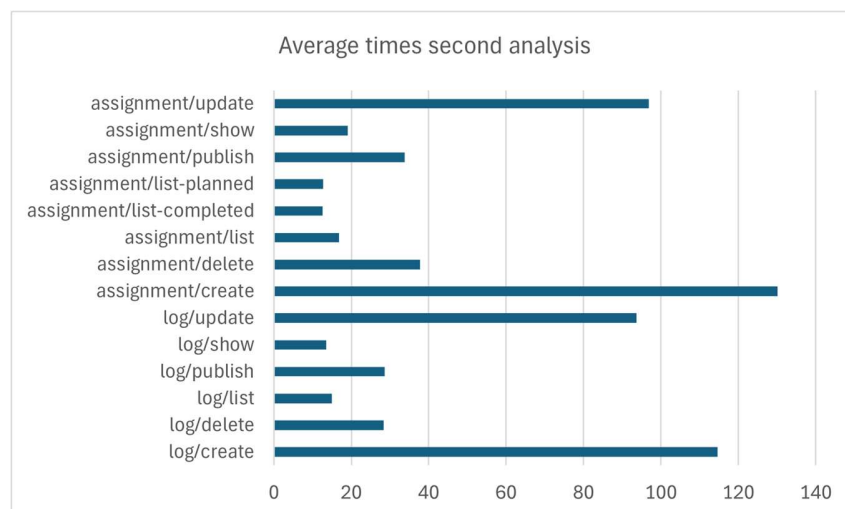


Figure 4. Average replay times when not charging

As we can see, the difference is not all that notable in the fastest cases, that are the list and show cases, but accumulate with the slower cases, such as update but specially create. The create case is something less than 2 times the time while charging, having an approximate value of 80 and 72 while charging but rising to about 135 and 118 when not.

If we see the means, that are 30.41547744 and 36.27183165 respectively, they are not that different but noticeable enough that while working with the system, the user notices them.

## Second study case: no indexes vs indexes

The study of the usefulness of indexes was mandatory for the deliverable, however there have been some curious values that I would like to discuss.

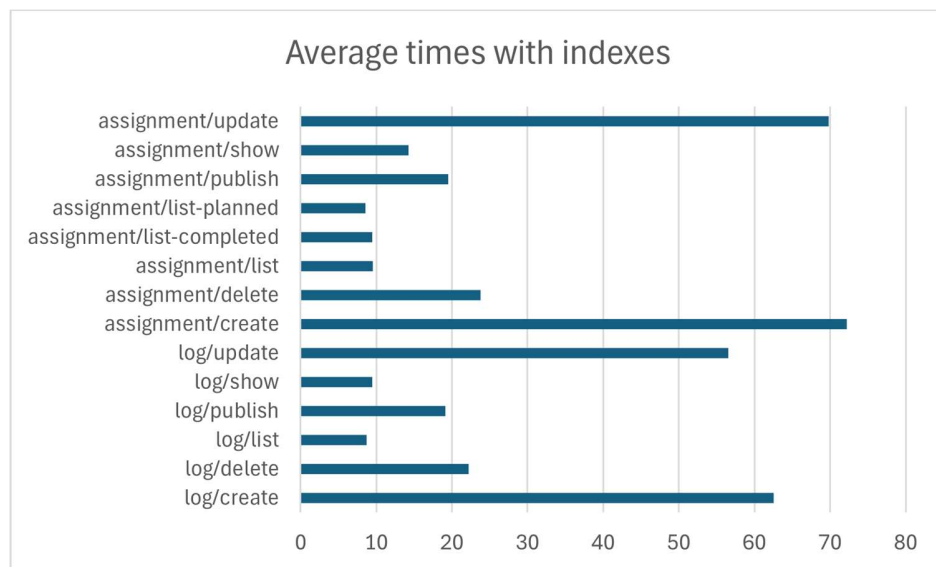


Figure 5. Average replay times with indexes

The slowest test cases, the create cases, are noticeably lower, managing to reduce almost by 10 the value of assignment/create. However, possibly less noticeable is the rise of update cases.

request-path	no indexes	indexes	change (percentage)
averages			
/flight-crew/activity-log/create Average	72.70516	62.57494	86.06671108
/flight-crew/activity-log/delete Average	25.06595	22.19306667	88.53870157
/flight-crew/activity-log/list Average	10.15636667	8.677216667	85.43622884
/flight-crew/activity-log/publish Average	20.01905	19.107425	95.44621248
/flight-crew/activity-log/show Average	11.2068875	9.48528125	84.63796259
/flight-crew/activity-log/update Average	50.18966667	56.538875	112.6504294
/flight-crew/flight-assignment/create Average	80.382885	72.24526	89.87642083
/flight-crew/flight-assignment/delete Average	25.17367778	23.76347778	94.3981169
/flight-crew/flight-assignment/list Average	11.15209677	9.503616129	85.21819996
/flight-crew/flight-assignment/list-completed Average	10.821	9.4595	87.41798355
/flight-crew/flight-assignment/list-planned Average	11.807	8.566	72.5501821
/flight-crew/flight-assignment/publish Average	20.226425	19.53459167	96.57955702
/flight-crew/flight-assignment/show Average	14.58364545	14.29110909	97.9940793
/flight-crew/flight-assignment/update Average	62.32687333	69.84210667	112.0577737

Figure 6. Average times direct comparison

As we can see in (Figure 6), most cases perform better with indexes, around a 12% decrease in time. However, the update cases suffer a 12% uptime, the same percentage but inversed. The mean of all percentages is around 92%, so, overall, the change is positive.

<b>z-test</b>		
	<i>no indexes (16.2162)</i>	<i>indexes (10.9009)</i>
Mean	28.28827072	27.55030497
Known Variance	28.22194066	27.45882473
Observations	181	181
Hypothesized Mean Difference	0	
z	1.330524552	
P(Z<=z) one-tail	0.091672751	
z Critical one-tail	1.644853627	
P(Z<=z) two-tail	0.183345503	
z Critical two-tail	1.959963985	

Figure 7. Z-test study

In our performance evaluation, we have obtained a value of **0.183345503** for an alpha of **0.05**. According to our course methodology, because our p-value is in the  $(\alpha, 1]$  interval and not in the  $[0, \alpha)$  one, the betterment is not significant.

However, due to the reduction of the mean, even if it is minimal, I have decided to keep the indexes as part of the code.

## Conclusions

In this report, we have specified the test cases, their objective and their result and we have analyzed the performance in three distinct cases.



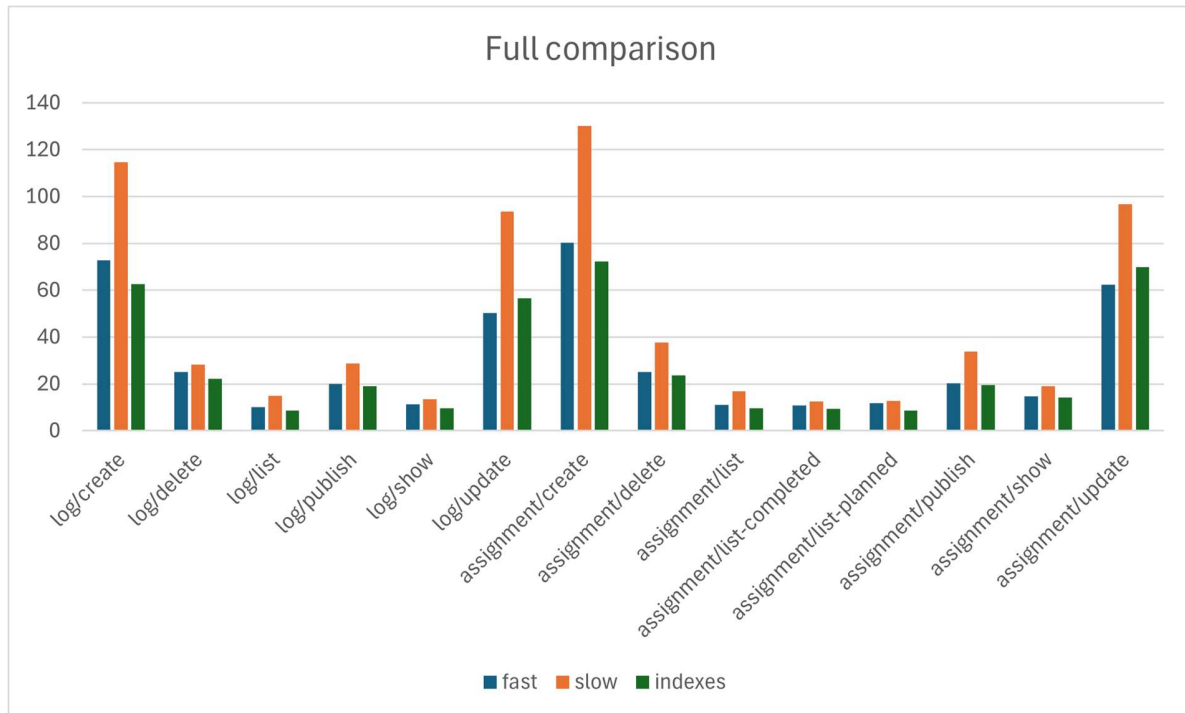


Figure 7. Full performance comparison

By this comparison, we have found that the optimal working conditions for our project and my computer is to run it while charging and with indexes.

## Bibliography