Insertando imagen...

ACME-SF

G1.007

**Analysis report S05-D03**

23/04/2024



# Cover

|  |  |
| --- | --- |
| **Repository:** <https://github.com/Pablo-Caballero-Maria/Acme-One-24.1.0-C1.07> | |
| **Student #1**  **ID:** 31878881F  **UVUS:**  pabcabmar3  **Name:** Caballero María, Pablo **Roles:** manager, developer | **Student #2**  **ID Number:**49034820Q  **UVUS:** mararnmon  **Name:** Arnáiz Montero, Marco Antonio  **Roles:** developer, operator |
| **Student #3**  **ID Number:** 77865211E  **UVUS:**  alfalolan  **Name:** Alonso Lanzarán, Alfonso Luis  **Roles:** developer, tester | **Student #4**  **ID Number:** 53932912M  **UVUS:** albsanmim  **Name:** Sánchez Mimbrero, Alberto  **Roles:** developer |
| **Student #5**  **ID Number:** 48123111G  **UVUS:** juagarcar4  **Name:** Garcia Carballo, Juan  **Roles:** developer, analyst |  |

Table of contents

[Cover 2](#_Toc1812869520)

[Executive summary 3](#_Toc2137473466)

[Revision table 3](#_Toc561794051)

[Introduction 3](#_Toc368742189)

[Contents 4](#_Toc108401824)

[Conclusions 6](#_Toc853269074)

[Bibliography 6](#_Toc1196555261)

# Executive summary

This document outlines Functional Requirements of the third deliverable, related to operations that auditors can perform on code audits and records. Included operations are listing, showing details, creating, updating, and deleting both audits and code records.

A significant technical aspect discussed is the implementation of the "mark" attribute, essential for determining whether an audit can be published, requiring at least a "C" grade.

# Revision table

|  |  |  |
| --- | --- | --- |
| Number | Date(dd/mm/yyyy) | Description |
| 1.0 | 23/04/2024 | Document done in its entirety, reviewed by peers. No major errors were found. |

# Introduction This analysis addresses crucial aspects such as the creation, update, display, and deletion of audits and records, emphasizing the importance of functionality and effective data management in the system. Additionally, the implications and proposed solutions for handling the 'mark' attribute are discussed, which plays an essential role in determining the publishability of code audits.

# Contents

* Student #5. Functional Requirements 6: Operations by auditors on code audits

• List the code audits that they have created.

• Show the details of their code audits.

• Create, update, or delete their code audits. Code audits can be updated or deleted as long as they have not been published. For a code audit to be published, the mark must be, at least, “C”.

* + Problem presented
  + To implement the 'mark' attribute, it was stated in the previous deliverable that it would be calculated in the service. However, there are several alternatives for how to implement it
  + Proposed Solutions:

1. Create a 'Mark' property in the Code Audit entity, which would be initialized as null (when there is no associated Audit Record to the Code Audit). Each time an Audit Record is added or modified, this 'Mark' property would be recalculated.

As it is a property stored in the database, it is not necessary to recalculate it every time it is displayed. However, this approach significantly increases the complexity during the editing and creation of Audit Records, because it would be necessary to recalculate the mark each time an Audit Record is updated or created. This could lead to unexpected bugs in the code."

1. Calculate the mark each time it is to be displayed. This would result in a simpler implementation and less potential for bugs in the functionality. Additionally, it is not necessary to calculate the mark each time an Audit Record is modified. However, on the other hand, it would be necessary to recalculate the mark each time it is to be shown to the user
   * Selected alternative:

Following the instructions given in the "Tutorials" forum, we will decide to opt for the second alternative and we will calculate the mark each time it needs to be displayed.

Link to the forum where the problem was discussed:

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_85092_1&nav=discussion_board&conf_id=_405265_1&forum_id=_234042_1&message_id=_416134_1>

* Student #5. Functional Requirements 6 & 7:

6. Operations by auditors on code audits:

List the code audits that they have created.

Show the details of their code audits.

Create, update, or delete their code audits. Code audits can be updated or deleted as long as they have not been published. For a code audit to be published, the mark must be, at least, “C”.

7. Operations by auditors on audit records:

List the audit records in their code audits.

Show the details of their audit records.

Create and publish an audit record.

Update or delete an audit record as long as it is not published.

* + Problem presented

The requirement does not clarify whether all audit records of a code audit must be published before the code audit itself can be published. Initially, there is nothing to indicate this necessity in the requirement, but given the type of relationship between the entities (composition), it would be understandable to implement this restriction. Additionally, this leads to another restriction: if all audit records associated with a code audit must be published in order to publish the code audit, this would mean that it is not allowed to create audit records for a code audit that has already been published.

* + Proposed Solutions:

1. Not to implement these two restrictions. On one hand, they are two validations we can avoid including, which reduces our workload. On the other hand, we could face application maintenance issues if users publish an unfinished code analysis, mistakenly believing it is complete due to the lack of indication in the application, which would require technical assistance from the team.
2. Implement the two restrictions. With this choice, we would be reaffirming the nature of the relationship between the two entities and avoiding the previously mentioned problems.
   * Selected alternative:

Following the instructions given in the "Tutorials" forum, we will decide to opt for the second alternative, we would be avoiding the previously mentioned problems.

Link to the forum where the problem was discussed: <https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_85092_1&nav=discussion_board&conf_id=_405265_1&forum_id=_234042_1&message_id=_416338_1>

NOTE: This conversation was not carried out by any of the members of the group but has been taken into account to select an alternative.

# Conclusions

After detailed analysis, it was decided to implement the real-time calculation of the "mark" attribute each time it is required for display, rather than keeping it constantly updated in the database. This decision facilitates a less complex and more robust implementation against potential errors, while meeting the established functional requirements.

Additionally, it was decided to implement restrictions that ensure all audit records of a code audit are published before the audit itself can be published, reinforcing the integrity of the information and the overall audit process. These strategic decisions highlight the team's adaptability to the project's needs and its ability to effectively solve problems, ensuring the success and quality of the ACME-SF system.

# Bibliography

Public Forum “Tutorials”: <https://ev.us.es/webapps/discussionboard/do/forum?action=list_threads&course_id=_85092_1&nav=discussion_board&conf_id=_405265_1&forum_id=_234042_1>