Statement: UCLM's Library Management System

DELIVERY 1: THEORITICAL WORK

Subject:Software engineering

Group(2025/2026):A1-2

Degree:Computer Science engineering

Date:13-11-2025

Team members:

FRANCISCO JAVIER YÉBENES CAZALLAS
MOHAMED AMIN EL HARRADJI AOURY
MARTÍN GARCÍA NIETO RODRÍGUEZ
ALEJANDRO ORTEGA MORENO
JAVIER SOBRINO OCAÑA
INCI ÖCAL
ECE MINA ÖRENLER

Index:

1.Requirements:

Functional Requirements – Students

FR-EST-01: Search and browse the bibliographic catalog (books, journals, theses, digital resources) in general

FR-EST-02: Reserve library materials

FR-EST-03: Renew loans of library materials

FR-EST-04: Access general and specialized reading clubs

FR-EST-05: Access electronic resources (databases, e-books, etc) with appropriate rights management

FR-EST-06: view and manage loan history

FR-EST-07: Request the use and loan of specific hardware (e.g., laptops, desktop computers, tablets)

FR-EST-08: Search and browse bibliographic materials relevant to each enrolled course

FR-EST-09 Request the use of physical spaces such as meeting rooms or study areas

FR-EST-10: Access study groups organized by course or by topic of interest

Functional Requirements - Faculty and Researchers

FR-FAC-01: Extended access to restricted resources funded by research projects

FR-FAC-02: Request bibliometric studies of R&D activity for specific calls (e.g., applications for Research Assessment Exercises or ANECA

FR-FAC-03: Manage recommended reading lists for courses

FR-FAC-04: Manage bibliographic resources and subscriptions for research projects

FR-FAC-05: Create and manage study groups

FR-FAC-06: Access usage statistics for student resources or research group members

Functional Requirements – Library Staf

FR-LIB-01: Control reservations and renewals

FR-LIB-02: Generate operational reports on the use of hardware and electronic resources

FR-LIB-03: Create update and delete library user accounts

FR-LIB-04: Create update and delete bibliographic records

FR-LIB-05: Manage loans and returns

FR-LIB-06: Maintain inventory and track physical materials

Functional Requirements – System Administrators

FR-ADM-01: Generate global statistics and usage reports

FR-ADM-02: Manage system updates

FR-ADM-03: Configure system parameters (loan policies, schedules, etc)

FR-ADM-04: Manage roles and permissions

FR-ADM-05: Integrate with other university systems (e.g., academic system)

Non-Functional Requirements – Library System

NFR-01: The system shall provide access via web, desktop (Windows, Linux, macOS), and mobile (Android, iOS, iPadOS) applications.

NFR-02: The system shall follow a client-server architecture.

NFR-03: All data (users, loans, bibliographic resources) shall be stored in a centralized database.

NFR-04: The system shall enforce role-based access control for students, faculty, library staff, and administrators.

NFR-05: The system shall provide secure integration with the university and other academic systems for user and course synchronization.

NFR-06: The system shall deliver fast performance, with search results and pages loading in the fatest time possible under normal conditions.

NFR-07: The system shall provide clear notifications for loans, reservations, and resource availability.

2.Mapping Functional Requirements to uses cases

Each functional requirement has a corresponding use case. Since the system includes both a client application and a server application, each use case is implemented on both sides.

Use cases were also assigned a priority level (1 = High, 2 = Medium, 3 = Low). Priority decisions followed a CRUD-based logic, giving higher importance to operations necessary for creation or modification of data.

Effort estimation for the Client application:

REQ	UC	Priority	R	A	D	I	T
FR1	CUC1	1	4	4	3	5	5
FR2	CUC2	2	3	4	3	7	5
FR3	CUC3	2	2	4	5	6	6
FR4	CUC4	3	3	4	2	6	5
FR5	CUC5	2	3	3	4	7	5
FR6	CUC6	3	4	4	4	6	6
FR7	CUC7	3	2	1	4	7	7
FR8	CUC8	1	2	4	3	9	4
FR9	CUC9	2	3	6	2	6	3
FR10	CUC10	2	4	3	2	6	5
FR11	CUC11	3	2	4	3	9	7

FR12	CUC12	3	4	7	3	9	5
FR13	CUC13	2	3	6	4	6	8
FR14	CUC14	2	4	6	3	10	7
FR15	CUC15	1	5	5	4	7	4
FR16	CUC16	3	4	4	3	9	5
FR17	CUC17	2	3	8	4	7	6
FR18	CUC18	3	4	5	3	7	4
FR19	CUC19	1	2	4	5	10	9
FR20	CUC20	1	3	2	5	10	9
FR21	CUC21	2	4	5	3	9	6
FR22	CUC22	2	3	5	3	9	3
FR23	UC23	2	4	5	5	9	4
FR24	CUC24	2	4	4	4	6	4
FR25	CUC25	2	2	4	4	6	6
FR26	CUC26	2	3	3	4	9	7
FR27	CUC27	1	2	3	5	10	7

Effort estimation for the Server application:

REQ	UC	Priority	R	A	D	I	T
FR1	SUC1	1	4	4	3	5	5

FR2	SUC2	2	3	4	3	7	5
FR3	SUC3	2	2	4	5	6	6
FR4	SUC4	3	3	4	2	6	5
FR5	SUC5	2	3	3	4	7	5
FR6	SUC6	3	4	4	4	6	6
FR7	SUC7	3	2	1	4	7	7
FR8	SUC8	1	2	4	3	9	4
FR9	SUC9	2	3	6	2	6	3
FR10	SUC10	2	4	3	2	6	5
FR11	SUC11	3	2	4	3	9	7
FR12	SUC12	3	4	7	3	9	5
FR13	SUC13	2	3	6	4	6	8
FR14	SUC14	2	4	6	3	10	7
FR15	SUC15	1	5	5	4	7	4
FR16	SUC16	3	4	4	3	9	5
FR17	SUC17	2	3	8	4	7	6
FR18	SUC18	3	4	5	3	7	4
FR19	SUC19	1	2	4	5	10	9
FR20	SUC20	1	3	2	5	10	9
FR21	SUC21	2	4	5	3	9	6
FR22	SUC22	2	3	5	3	9	3
FR23	SUC23	2	4	5	5	9	4
FR24	SUC24	2	4	4	4	6	4

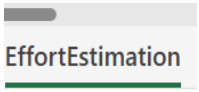
FR25	**SUC25**	**2**	**2**	**4**	**4**	**6**	**6**
FR26	**SUC26**	**2**	**3**	**3**	**4**	**9**	**7**
FR27	**SUC27**	**1**	**2**	**3**	**5**	**10**	**7**

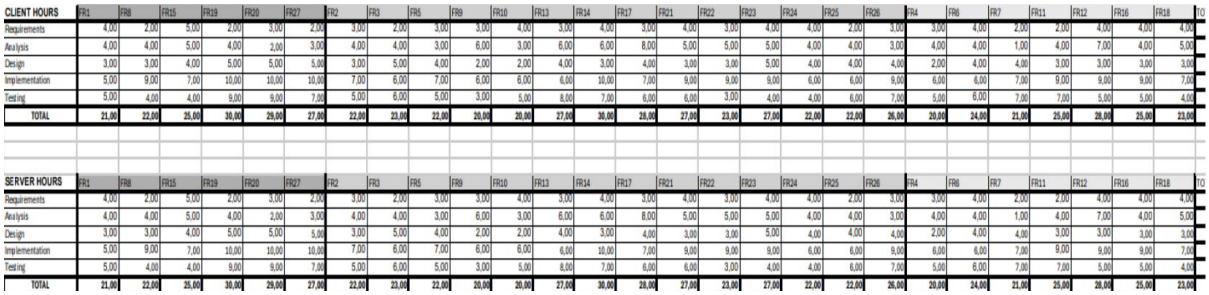
Priorities :

High Priority (1):
Core functionalities essential for the system to operate. These represent the minimum required capabilities for the system to fulfill its primary objectives and must be implemented in the initial development phase.

Medium Priority (2):
Important features that significantly enhance usability and system value. These support project goals and user expectations, but are not required for the initial basic operation.

Low Priority (3):
Non-critical features that contribute to completeness and improvement of the system. These can be scheduled for later phases as time and resources allow.

To view these tables in Excel, please refer to the Excel file located in the "Effort Estimation" section. 



CLIENT HOURS	FR1	FR8	FR15	FR19	FR20	FR27	FR2	FR3	FR5	FR9	FR10	FR13	FR14	FR17	FR21	FR22	FR23	FR24	FR25	FR26	FR4	FR6	FR7	FR11	FR12	FR16	FR18	TO
Requirements	4,00	2,00	5,00	2,00	3,00	2,00	3,00	2,00	3,00	3,00	4,00	3,00	4,00	3,00	4,00	3,00	4,00	4,00	2,00	3,00	3,00	4,00	2,00	2,00	4,00	4,00	4,00	
Analysis	4,00	4,00	5,00	4,00	2,00	3,00	4,00	4,00	3,00	6,00	3,00	6,00	6,00	8,00	5,00	5,00	5,00	4,00	4,00	3,00	4,00	4,00	1,00	4,00	7,00	4,00	5,00	
Design	3,00	3,00	4,00	5,00	5,00	5,00	3,00	5,00	4,00	2,00	2,00	4,00	3,00	4,00	3,00	3,00	5,00	4,00	4,00	4,00	2,00	4,00	4,00	3,00	3,00	3,00	3,00	
Implementation	5,00	9,00	7,00	10,00	10,00	10,00	7,00	6,00	7,00	6,00	6,00	6,00	10,00	7,00	9,00	9,00	9,00	6,00	6,00	9,00	6,00	6,00	7,00	9,00	9,00	9,00	7,00	
Testing	5,00	4,00	4,00	9,00	9,00	7,00	5,00	6,00	5,00	3,00	5,00	8,00	7,00	6,00	6,00	3,00	4,00	4,00	6,00	7,00	5,00	6,00	7,00	7,00	5,00	5,00	4,00	
TOTAL	21,00	22,00	25,00	30,00	29,00	27,00	22,00	23,00	22,00	20,00	20,00	27,00	30,00	28,00	27,00	23,00	27,00	22,00	22,00	26,00	20,00	24,00	21,00	25,00	28,00	25,00	23,00	
SERVER HOURS	FR1	FR8	FR15	FR19	FR20	FR27	FR2	FR3	FR5	FR9	FR10	FR13	FR14	FR17	FR21	FR22	FR23	FR24	FR25	FR26	FR4	FR6	FR7	FR11	FR12	FR16	FR18	TO
Requirements	4,00	2,00	5,00	2,00	3,00	2,00	3,00	2,00	3,00	3,00	4,00	3,00	4,00	3,00	4,00	3,00	4,00	4,00	2,00	3,00	3,00	4,00	2,00	2,00	4,00	4,00	4,00	
Analysis	4,00	4,00	5,00	4,00	2,00	3,00	4,00	4,00	3,00	6,00	3,00	6,00	6,00	8,00	5,00	5,00	5,00	4,00	4,00	3,00	4,00	4,00	1,00	4,00	7,00	4,00	5,00	
Design	3,00	3,00	4,00	5,00	5,00	5,00	3,00	5,00	4,00	2,00	2,00	4,00	3,00	4,00	3,00	3,00	5,00	4,00	4,00	4,00	2,00	4,00	4,00	3,00	3,00	3,00	3,00	
Implementation	5,00	9,00	7,00	10,00	10,00	10,00	7,00	6,00	7,00	6,00	6,00	6,00	10,00	7,00	9,00	9,00	9,00	6,00	6,00	9,00	6,00	6,00	7,00	9,00	9,00	9,00	7,00	
Testing	5,00	4,00	4,00	9,00	9,00	7,00	5,00	6,00	5,00	3,00	5,00	8,00	7,00	6,00	6,00	3,00	4,00	4,00	6,00	7,00	5,00	6,00	7,00	7,00	5,00	5,00	4,00	
TOTAL	21,00	22,00	25,00	30,00	29,00	27,00	22,00	23,00	22,00	20,00	20,00	27,00	30,00	28,00	27,00	23,00	27,00	22,00	22,00	26,00	20,00	24,00	21,00	25,00	28,00	25,00	23,00	

3.Mapping To Iterations

We are mapping both Client System Use Case (CUC#) and Server System Use Case (SUC#) of the FR# into one iteration

Iteration	Use Case
it1	CUC1, SUC1
it2	CUC8, SUC8
it3	CUC15, SUC15
it4	CUC19, SUC19
it5	CUC20, SUC20
it6	CUC27, SUC27
it7	CUC2, SUC2
it8	CUC3, SUC3
it9	CUC5, SUC5
it10	CUC9, SUC9
it11	CUC10, SUC10
it12	CUC13, SUC14
it13	CUC17, SUC17
it14	CUC21, SUC21
it15	CUC21, SUC21

it16	CUC22, SUC22
it17	CUC23, SUC23
it18	CUC24, SUC24
it19	CUC25, SUC25
it20	CUC26, SUC26
it21	CUC4, SUC4
it22	CUC6, SUC6
it23	CUC7, SUC7
it24	CUC11, SUC11
it25	CUC12, SUC12
it26	CUC16, SUC16
it27	CUC18, SUC18

RED: priority 1

Green: priority 2

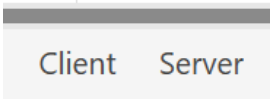
Blue:priority 3

4.PUD SRATEGY

A table was created for each strategy following the RADIT rubric presented in class, including calculations of total cost and total hours.

	ITERATION 1				
	R	A	D	I	T
Fran (Requirements and Analisys)	2	2			
Amin (Requirements and Analisys)	2	2			
Alejandro (Design and Implementations)			1	2	
Martin (Design and Implementations)			1	2	
Ece (Design and Implementations)			1	1	
Inci (Testing)					3
Javier (Testing)					2
Cost	1015				
Hours	21				

	ITERATION 2				
	R	A	D	I	T
Fran (Requirements and Analisys)	2	2			
Amin (Requirements and Analisys)	1	2			
Alejandro (Design and Implementations)			1	1	
Martin (Design and Implementations)			1	2	
Ece (Design and Implementations)			1	4	
Inci (Testing)					3
Javier (Testing)					2
Cost	965				
Hours	22				

For a complete view of all PUD strategy tables, please see the Excel file under the “Client and Server” section 

5.Cost Analysis

The project was divided into phases, and each phase had a specific number of team members assigned based on the type of work required:

- **Requirements Phase:** 2 workers
- **Analysis Phase:** 2 workers
- **Design Phase:** 3 workers

- **Implementation Phase:** 3 workers
- **Testing Phase:** 2 workers

Each phase has an associated hourly cost depending on the complexity and specialization required:

- Requirements work: **100 € per hour**
- Analysis work: **60 € per hour**
- Design work: **50 € per hour**
- Implementation work: **25 € per hour**
- Testing work: **20 € per hour**

The total cost for each phase was calculated by multiplying the estimated hours by the hourly cost and the number of workers assigned to that phase.

We created an excel file for this part:

TOTAL HOURS		PRICE PER HOUR	TOTAL PRICE
Requierments	172	100,00 €	17.200,00 €
Analisys	234	60,00 €	14.040,00 €
Desing	194	50,00 €	9.700,00 €
Implementation	414	25,00 €	10.350,00 €
Testing	304	20,00 €	6.080,00 €
Inception			1.500,00 €
Transition			2.000,00 €
TOTAL BUDGET			**60.870,00 €**

Taking into account a week has 40 workable hours, number of weeks can also be computed (for 1 worker).

Summary

- **Total Hours:** 1318 h

- **Total Budget:** 60.870,00 €

ELABORATION time (h)	308
CONSTRUCTION time(h	1010
TOTAL HOURS	1318
HOURS PER WEEK	40

6.Project Agenda

The first part, Inception, was named as it0, which is done in the Wiki Page

Elaboration Phase

After the Elaboration part, the objective is having the System Architecture. In our problem, this is having all Use Cases with Priority 1. This will last 6 iterations (it1 - it6)

It1	It2	It3	It4	It5	It6
CUC1	CUC8	CUC15	CUC19	CUC20	CUC27
SUC1	SUC8	SUC15	SUC19	SUC20	SUC27

Construction Phase

Then, in Construction phase, all the rest of use cases can be found. This will be the longest phase including 21 iterations (it7- it27)

It7	It8	It9	It10	It11	It12	It13	It14
CUC2	CUC3	CUC5	CUC9	CUC10	CUC13	CUC14	CUC17
SUC2	SUC3	SUC5	SUC9	SUC10	SUC13	SUC14	SUC17

It15	It16	It17	It18	It19	It20	It21
CUC21	CUC22	CUC23	CUC24	CUC25	CUC26	CUC4

SUC21	SUC22	SUC23	SUC24	SUC25	SUC26	SUC4

It22	**It23**	**It24**	**It25**	**It26**	**It27**
CUC6	CUC7	CUC11	CUC12	CUC16	CUC18
SUC6	SUC7	SUC11	SUC12	SUC16	SUC18

Transition Phase

Finally, in the Transition part, final version and deployment can be found.

7.Project calendar

Taking the previous information into account the project schedule would result like this:

Also you can see this file in the excel created in the section (Schedule).

8.Quality management

The quality management strategy defines the key quality characteristics that our Library Management System must meet to ensure long-term efficiency, user satisfaction, and system stability.

After analyzing the system's objectives and scope, we identified four main quality characteristics as the most relevant for our project: Security, Maintainability, Usability, and Reliability.

These characteristics were selected because they directly influence the system's performance, sustainability, and user experience.

1. Security

Justification:

Security is essential to protect the system from unauthorized access, data breaches, and malicious actions. Applying security by design ensures trust, data integrity, and system stability, safeguarding both users and institutional resources.

Non-Functional Requirements Related to Security:

- **Two-Step Authentication:** Users must verify their identity using more than one authentication method.
- **Data Encryption:** Sensitive data must be encrypted both during transmission and storage.
- **Backup and Recovery Mechanisms:** The system must perform automatic backups and provide recovery procedures in case of corruption or data loss.
- **Role-Based Access Control:** Permissions are assigned according to user roles (students, faculty, staff, administrators).

Impact on Project Planning:

Integrating advanced security mechanisms will increase development time and testing effort, as encryption, authentication, and access control

must be carefully implemented and validated. This may also involve additional costs for security tools and maintenance.

2. Maintainability

Justification:
Maintainability guarantees that the system can be easily updated, corrected, and extended throughout its lifecycle. A maintainable system simplifies future improvements, reduces long-term costs, and supports collaborative development.

Non-Functional Requirements Related to Maintainability:

- **Reusable Architecture:** System components should be designed for reuse in other functionalities.
- **Code Documentation:** The code must include clear comments, naming conventions, and developer documentation.
- **Version Control Management:** A version control system must be used to track changes and support team collaboration.
- **Modular Structure:** The system should be divided into independent modules to simplify maintenance and updates.

Impact on Project Planning:
Ensuring maintainability will require extra time during the design and coding phases to implement a modular structure and proper documentation. However, it will reduce future maintenance costs and facilitate scalability.

3. Usability

Justification:
Usability measures how easy and intuitive the system is for users. Since

students, faculty, and staff will interact with the system daily, providing a clear and accessible interface is vital to ensure efficiency and satisfaction.

Non-Functional Requirements Related to Usability:

- **Consistent User Interface:** A unified interface across all platforms (web, mobile, desktop).
- **Accessibility Compliance:** The interface must meet accessibility standards (WCAG) for users with disabilities.
- **Clear Navigation Structure:** Users should easily locate resources, reservations, and account details.
- **User Feedback and Guidance:** The system should provide meaningful messages for errors, confirmations, and help.

Impact on Project Planning:

Usability improvements require additional time for interface design, prototype testing, and user feedback sessions. This may increase development effort but ensures a more successful and user-friendly product.

4. Reliability

Justification:

Reliability ensures the system performs correctly and consistently over time, minimizing interruptions and failures. Given that the library system supports academic operations, reliability is crucial for continuous service availability.

Non-Functional Requirements Related to Reliability:

- **High Availability:** The system should remain accessible even during peak usage periods.
- **Fault Tolerance:** The system must recover gracefully from minor failures.

- **Performance Monitoring:** Ongoing monitoring should detect performance issues early.
- **Recovery Time Objective (RTO):** Services should be restored within an acceptable timeframe after failure.

Impact on Project Planning:

Ensuring reliability requires additional testing phases, such as stress tests, performance analysis, and backup validation. These activities may extend the testing schedule and slightly increase infrastructure costs but will significantly improve system robustness.

Conclusion

By incorporating **Security, Maintainability, Usability, and Reliability** as core quality characteristics, the Library Management System will achieve a high standard of performance, safety, and user satisfaction.

However, these characteristics also introduce **additional time, effort, and cost** into the project plan. Therefore, the development schedule should be **revised** to include:

- Security testing and validation phases.
- Dedicated iterations for documentation and modularization.
- Usability evaluations and user feedback sessions.
- Performance and reliability testing before deployment.

Implementing these adjustments ensures a realistic, sustainable, and high-quality software product aligned with user needs and institutional standards.

9.Configuration Management

Introduction:

The **purpose of this configuration plan** is to track the expected versioning with successive software development, as well as to define the plan's limitations and the responsibilities, authorities, policies, resources, and schedule we will have, ensuring product integrity, facilitating change control, and optimizing system control throughout its entire lifecycle.

The **scope of the configuration plan** encompasses all software components, client requirements documents, the development phase, source code, and system manuals for both the client and the server.

The configuration elements will consist of the system components developed in each iteration. Each component will correspond to an implemented module or functionality and will be managed through specific branches in the Git repository, following the GitFlow model. This ensures that each iteration produces controlled and traceable versions of the components that make up the final product.

Criteria for Identifying Configuration Elements:

Configuration management will be applied to all project artifacts, such as requirements documents, source code, metadata databases, scripts, and user manuals. Each configuration item will have a unique identifier and will be controlled through the Git repository, applying semantic versioning.

Limitations and Assumptions:

The main limitation is the client's financial resources and a fixed budget for development. Other limitations and assumptions include the work schedule, which is defined by 8-hour workdays, 40 hours per week with rest days on Saturdays and Sundays.

Due to the tight schedule, to meet the final delivery date, it is necessary to work on two Saturdays: the Saturday of week 1 (November 15, 2025) and the Saturday of week 3 (November 29, 2025).

Three components will be completed per week, with an exception in weeks 3 and 4 where two components will be worked on simultaneously.

Although an iteration does not necessarily constitute a component, our organizational planning has resulted in one component being completed per iteration.

The team will consist of 7 people (2 requirements specialists and analysts, 3 designers and implementers and 2 testers), with a multi-device approach to meet the client's requirements.

The system's construction will be completed on December 5th, and from that date, a transition phase will begin during which preliminary versions will be generated to validate the product's quality. From December 8rd to 11th, the *alpha version* will be tested to detect general errors; from December 12th to 17th, the *beta version*, which is more stable and focused on verifying the system's actual functionality, will be evaluated. Finally, the final version will be delivered on December 19th, ensuring that the software has been reviewed and refined through several iterations to guarantee its stability and compliance with the requirements.

Plan Responsibilities and Authorities:

The Configuration Management Authority is the project **coordinator**, responsible for verifying that configuration management activities are planned and executed, and also responsible for evaluating and approving configuration changes.

Furthermore, all members of the work team are responsible for ensuring compliance with this plan and guaranteeing the correct implementation of the software development of each component.

Scheduled:

The project's timeline will follow the same schedule defined in the Excel spreadsheet, with no changes to the planned dates. Each use case has an initial version, V1.0.0, representing its first complete implementation.

The **Semantic Versioning** standard will be applied for version control, ensuring consistency and traceability in subsequent software updates.

Component versioning table:

Component	*Construction*	*Date completed*
Component 1	v1.0.0	15/11/2025
Component 2	v1.0.0	15/11/2025
Component 3	v1.0.0	15/11/2025
Component 4	v1.0.0	21/11/2025
Component 5	v1.0.0	21/11/2025
Component 6	v1.0.0	21/11/2025
Component 7	v1.0.0	28/11/2025
Component 8	v1.0.0	28/11/2025
Component 9	v1.0.0	28/11/2025
Component 10	v1.0.0	2/12/2025
Component 11	v1.0.0	2/12/2025
Component 12	v1.0.0	2/12/2025
Component 13	v1.0.0	5/12/2025
Component 14	v1.0.0	5/12/2025
Component 15	Not implemented	-
Component 16	Not implemented	-
Component 17	Not implemented	-
Component 18	Not implemented	-
Component 19	Not implemented	-
Component 20	Not implemented	-
Component 21	Not implemented	-
Component 22	Not implemented	-
Component 23	Not implemented	-
Component 24	Not implemented	-
Component 25	Not implemented	-
Component 26	Not implemented	-
Component 27	Not implemented	-

CM Responsibilities:

For **version control**, the project will use the GitFlow model, which allows separating development work from stable code. This system will facilitate collaborative work and continuous integration through dedicated

branches for development, new features, testing, releases, and urgent fixes.

Before each release, a **configuration audit** will be performed to verify that all components meet the defined functional and non-functional requirements, that versions are correctly labelled, and that there are no inconsistencies in the repository.

Planned Activities, Agenda and Resources:

Configuration management will follow the same timeline established in the overall project plan. Iteration 0 will be dedicated to planning and defining requirements, while subsequent iterations will focus on the incremental development of the various system components, with configuration reviews and audits before each release.

Git will be used as the primary version control and collaboration tool. Git will allow for maintaining traceability of changes, recording code and documentation versions, and ensuring an organized repository structure in accordance with the GitFlow model.

CM Plan Maintenance:

This Configuration Management Plan will be a living document, subject to continuous updates as changes occur in requirements or the project structure. Periodic reviews will be conducted at the end of each iteration to adapt the plan to new needs or technical adjustments identified during development.

Furthermore, the plan will serve as a reference at the start of each new phase, ensuring that approved changes are feasible, properly documented, and maintain system integrity.