Library management system requirements documentation

Ikram Filali & Honoris Ndereyimana

1.5 -- 2025-05-30

This document follows the requirements documentation structure presented in the Handbook of requirements and business analysis.

CHANGELOG

Version	Date	Comment
1.0	2025-05-19	Initial draft setup by Honoris and Ikram.
1.1	2025-05-21	Ikram added Goals requirements.
1.2	2025-05-22	Ikram added System requirements.
1.3	2025-05-22	Honoris added github CI/CD workflow.
1.4	2023-08-27	Ikram added Project requirements.
1.5	2023-12-22	Merged all feature branches to main.

Table of Contents

Goals . . . 3
- G.1 Context and overall objectives . . . 3
- G.2 Current situation . . . 3
- G.3 Expected benefits . . . 3
- G.4 Functionality overview . . . 4
- G.5 High-level usage scenarios . . . 4
- G.6 Limitations and exclusions . . . 5
- G.7 Stakeholders and requirements sources . . . 5

Environment . . . 6
- E.1 Glossary . . . 6
- E.2 Components . . . 6
- E.3 Constraints . . . 6
- E.4 Assumptions . . . 7
- E.5 Effects . . . 7
- E.6 Invariants . . . 7

System . . . 8
- S.1 Components . . . 8
- S.2 Functionality . . . 8
- S.3 Interfaces . . . 10
- S.4 Detailed usage scenarios . . . 11
- S.5 Prioritization . . . 12
- S.6 Verification and acceptance criteria . . . 13

Project . . . 14
- P.1 Roles and Personnel □|□□ . . . 14
- P.2 Imposed technical choices . . . 14
- P.3 Schedule and Milestones □ . . . 15
- P.4 Tasks and Deliverables □ . . . 16
- P.5 Required technology elements . . . 17
- P.6 Risk and Mitigation Analysis □□ . . . 18
- P.7 Requirements process and report . . . 18

Goals

Goals are "needs of the target organization, which the system will address". While the development team is the principal user of the other books, the Goals book addresses a wider audience: essentially, all stakeholders (see Handbook).

It must contain enough information to provide a general sketch of the entire project. To this effect, chapter G.3 presents a short overview of the system and G.1 will typically include some key properties of the environment. As it addresses a wide readership, it should be clear and minimize the use of specialized technical terms. Together, G.1, G.2 and G.3 describe the rationale for the project. It is important to state these justifications explicitly. Typically, they are well understood at the start of the project, but management and priorities can change (see Handbook).

G.1 Context and overall objectives

High-level view of the project: organizational context and reason for building a system (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

Many libraries still use outdated systems that slow down daily tasks. This project defines the requirements for a simple, modern Library Management System (LMS) to improve how books and users are managed. The goal is to build a clear, adaptable foundation for future development.

G.1.1 The public library system is designed to simplify and organize the book lending process across different library branches, making it easier for both users and staff to manage borrowings efficiently.

G.1.2 The system should help users easily place holds, check out books, and manage their interactions with the library collection in a clear and user-friendly way.

G.1.3 The system ensures that books are made available to users while respecting fair borrowing rules, such as limits on reservations and returning books on time to avoid overdue situations.

G.2 Current situation

Current state of processes to be addressed by the project and the resulting system (see Handbook).

G.2.1 There is no centralized, modern system in place. The structure and features of the future Library Management System are still being defined.

G.3 Expected benefits

New processes, or improvements to existing processes, made possible by the

projectâ€™s results (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

G.3.1 Librarians can easily add new books to the system, including details like title, author, and number of copies. This helps new books become available to users more quickly.

G.3.2 Users can manage their loans, reservations, and account online, without needing to ask a librarian for help. This saves time for everyone.

G.3.3 Users get alerts when a reserved book becomes available, or when a due date is approaching. This avoids surprises and missed returns.

G.3.4 The system keeps track of books that are returned late. It can warn users and apply small penalties if needed, based on the library's rules.

G.3.5 Library rules (like how long you can borrow a book or how many renewals are allowed) are applied automatically by the system, so everyone is treated equally.

G.4 Functionality overview

Overview of the functions (behavior) of the system. Principal properties only (details are in the System book) (see Handbook).

G.4.1 Users can search for books by title, author, or genre. They can also use filters to find exactly what they're looking for faster.

G.4.2 Each user has their own page where they can see their current loans, reservations, due dates, and past activity.

G.4.3 Users can reserve books, even if they are currently borrowed. The system manages a waiting list and alerts users when it's their turn.

G.4.4 Users can renew their loans if no one else is waiting for the book.

G.4.5 Librarians can manage the catalog, add or remove books, monitor usage, and apply changes to rules or policies through a dedicated interface.

G.4.6 The system sends reminders before a book is due. If the return is late, it can apply the appropriate rules automatically.

G.4.7 The system knows where each copy of a book is (borrowed, reserved, or on the shelf).

G.5 High-level usage scenarios

Fundamental usage paths through the system (see Handbook).

G.5.1 Encourage More Visitors

G.5.2 Help Students Find What They Need

G.5.3 Smooth Experience for Regular Users

G.5.4 Better Book Circulation

G.6 Limitations and exclusions

Aspects that the system need not address (see Handbook).

G.6.1 The system will not handle digital content such as e-books, remote file access, or integration with online academic libraries. It is also not meant to support payments, subscriptions, or any kind of financial transaction.

G.6.2 Physical logistics like book shelving, RFID tracking, or managing multiple library branches are not part of this system.

G.7 Stakeholders and requirements sources

Groups of people who can affect the project or be affected by it, and other places to consider for information about the project and system (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

G.7.1 **Key Stakeholders** The main people concerned by the system are:

- **Library users**, who borrow and reserve books. Their needs include a simple interface, clear due dates, and notifications.
- **Librarians**, who manage the catalog, loans, and user accounts. They need efficient tools to save time on daily tasks.
- **Library Administrators** : They are responsible for setting the library rules and monitoring operations. Their needs guided the design of system configuration features and reporting functions.

G.7.2 To define the system's needs, we are relying on:

- Real-life usage of library systems we know or have observed.
- Feedback from students and staff who use university libraries.
- Documentation and templates from https://requirements.university to ensure alignment with PEGS methodology.

Environment

The Environment book describes the application domain and external context, physical or virtual (or a mix), in which the system will operate (see Handbook).

E.1 Glossary

Clear and precise definitions of all the vocabulary specific to the application domain, including technical terms, words from ordinary language used in a special meaning, and acronyms (see Handbook).

This chapter should not be empty (following the Glossary Principle_, p.27 of the Handbook).

Example of terms definition.

E.1.1 Terms

Book

Copy of a book with a copy number and an availability status.

Catalog

List of library books and their instance availability.

E.2 Components

List of elements of the environment that may affect or be affected by the system and project. Includes other systems to which the system must be interfaced (see Handbook).

E.2.1 **Physical Library Branches** Geographically separate locations where books are stored and borrowed. Each maintains its own copy inventory.

E.2.2 **Library Users (Patrons)** Individuals registered to borrow materials. Have borrowing limits based on membership type.

E.3 Constraints

Obligations and limits imposed on the project and system by the environment (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

E.3.1 A single physical book copy can only be borrowed by one patron at a time.

E.3.2 Patrons may not borrow new books if they have overdue items or reached their borrowing limit.

E.4 Assumptions

Properties of the environment that may be assumed, with the goal of facilitating the project and simplifying the system (see Handbook).

E.4.1 Each book copy has a physically scannable identifier (barcode/RFID)

E.5 Effects

Elements and properties of the environment that the system will affect (see Handbook).

E.5.1 Reduce manual record-keeping by librarians for loans/returns

E.6 Invariants

Properties of the environment that the systemâ€™s operation must preserve (see Handbook).

E.6.1 Total copies = Available copies + Borrowed copies + Reserved copies

System

The System book refines the Goal one by focusing on more detailed requirements about the system under development, mainly its constituents, behaviors and properties.

S.1 Components

Overall structure expressed by the list of major software and, if applicable, hardware parts (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

S.1.1 **User Interface :** The web-based interface used by librarians and users. It includes the home page, search bar, user dashboard, and admin panel. It allows interaction with the system in a simple and intuitive way.

S.1.2 **Catalog Management Module :** Handles the storage and organization of all books and media in the library. It supports searching, filtering, and classification by metadata (title, author, genre, status).

S.1.3 **Loan and Reservation Module :** Manages borrowing, returns, renewals, and reservations. It also applies rules (loan duration, renew limits) based on the user type and book status.

S.1.4 **User Management Module :** Stores user information and login credentials. It tracks their current loans, reservation history, and late returns. It also enforces borrowing restrictions if needed.

S.1.5 **Notification System :** Sends alerts to users about book availability, due dates, or overdue books via email or dashboard messages.

S.1.6 **Database :** Stores all data about users, books, reservations, and system configuration. Ensures consistency and quick access for all modules.

S.1.7 **Admin Tools :** Used by library staff to configure system rules, add new books, view reports, and manage the entire system efficiently.

S.1.8 **Authentication System :** Manages login and role-based access (user vs librarian). Can later be connected to an existing university identity provider.

S.2 Functionality

One section, S.2.n, for each of the components identified in S.1, describing the corresponding behaviors (functional and non-functional properties; see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

S.2.1 User Interface

The UI allows users and librarians to interact with the system. It should be simple, responsive, and accessible. Key behaviors include:

- Displaying available books with filters and search tools.
- Showing personalized dashboards (loans, reservations, alerts).
- Ensuring consistent display across devices and browsers.
- Guiding the user clearly through the reservation or return process.

S.2.2 Catalog Management Module

This module:

- Stores all book metadata (title, author, ISBN, status, etc.).
- Allows librarians to add, modify or delete books.
- Supports fast and flexible search (by title, author, genre).
- Updates availability in real-time when books are borrowed or returned.

Non-functional: Must respond to search queries in under 1 second for a database of 10,000+ entries.

S.2.3 Loan and Reservation Module

This module manages the entire lifecycle of a book transaction:

- Allows users to borrow books if available and within their loan limits.
- Lets users reserve a book already borrowed.
- Applies rules like loan duration or number of renewals.
- Cancels reservations if not picked up on time.

Non-functional: Ensures no double booking of the same book copy.

S.2.4 User Management Module

This module:

- Handles user registration, login, and roles.
- Keeps a history of user activity (reservations, loans, penalties).
- Enforces borrowing limitations.

Non-functional: Protects user data according to GDPR principles.

S.2.5 Notification System

This module:

- Sends automatic alerts before due dates.
- Notifies users when a reserved book is available or a loan is late.
- Allows users to manage their notification preferences (e.g., email only).

Non-functional: Ensures delivery within 5 minutes of event trigger.

S.2.6 Database

The database:

- Stores all persistent data (books, users, transactions).
- Is structured to allow fast queries and scalability.
- Supports regular backups and protects against data corruption.

Non-functional: Database must support 99.9% uptime and backup every 24h.

S.2.7 Admin Tools

This module:

- Provides staff with access to advanced tools for managing users, books, and policies.
- Generates statistics and reports (most borrowed books, late returns).
- Lets staff configure system behavior (loan limits, penalties).

Non-functional: Interface should be intuitive and require no technical knowledge.

S.2.8 Authentication System

This system:

- Differentiates between users and librarians with role-based access.
- Verifies credentials securely.
- Can later integrate with a central identity provider (e.g., university SSO).

Non-functional: Must comply with best practices for password encryption and access control.

S.3 Interfaces

How the system makes the functionality of S.2 available to the rest of the world, particularly user interfaces and program interfaces (APIs) (see Handbook).

S.3.1 Graphical User Interface (GUI)

At this stage, all interactions with the system are performed through a web-based user interface. This interface is designed to be:

- Accessible via standard web browsers.
- Divided into sections based on user roles (user vs. librarian).
- Responsive and easy to use, even for non-technical users.

The interface provides access to all key functionalities described in S.2: catalog search, reservations, account management, and administration.

S.3.2 Future API Integration (Planned)

Although no programmatic interface (API) is available in the current version, future development may include:

- A REST API for integration with external systems (e.g., university portals).
- Endpoints for retrieving book data, user status, or system statistics.

These additions would support automation, mobile applications, or third-party services.

S.4 Detailed usage scenarios

Examples of interaction between the environment (or human users) and the system: use cases, user stories (see Handbook).

S.4.1 User Story: Reserving a Book

As a student, I want to search for a book and reserve it online, so I can pick it up as soon as it becomes available.

→ The system shows the current status of the book.

→ If the book is borrowed, the system adds me to the reservation queue.

→ I receive an email notification when it's my turn to pick up the book.

S.4.2 User Story: Managing a Late Return

As a librarian, I want the system to automatically detect late returns and apply the corresponding rules, so I don't have to check each user manually.

→ The system flags the loan as overdue when the due date has passed.

→ The user receives an automatic reminder and warning by email.

→ If the book is still not returned, the system applies the penalty.

→ I can review and override the penalty if needed through the admin panel.

S.4.3 User Story: Managing My Account

As a regular user, I want to log in to my personal dashboard to check which books I've borrowed, when they are due, and if I can renew them.

→ I access my dashboard after logging in.

→ I see a list of current loans with due dates.

→ I click on a book to renew it, if allowed.

S.4.4 User Story: Adding a New Book

As a librarian, I want to add a new book to the catalog with all its details, so it can be borrowed by users.

→ I access the admin panel.

→ I fill in the book information (title, author, copies, status).

→ The book appears in the public catalog immediately.

S.4.5 User Story: Checking Availability Before Visiting

As a visitor, I want to check online if a book is available in the library, so I don't waste time coming for nothing.

→ I search the book by title on the public site.

→ I see that it's available and on which shelf.

→ I go to the library to borrow it.

S.5 Prioritization

Classification of the behaviors, interfaces and scenarios (S.2, S.3 and S.4) by their degree of criticality (see Handbook).

Element	Description	Priority
S.2.1 – User Interface	Main access point for users and librarians. Needed for all interactions.	Critical
S.2.2 – Catalog Management Module	Core of the system: handles book data and availability.	Critical
S.2.3 – Loan and Reservation Module	Manages borrowing and reservations. Central to system usage.	Critical
S.2.4 – User Management Module	Stores and controls user access and rights.	Important
S.2.5 – Notification System	Improves user experience but can be delayed or simplified in MVP.	Optional

Element	Description	Priority
S.2.6 – Database	Ensures persistent and consistent storage of data.	Critical
S.2.7 – Admin Tools	Support daily operations of the librarians.	Important
S.2.8 – Authentication System	Protects system access, especially for sensitive librarian functions.	Critical
S.3.1 – Graphical User Interface	Essential for using the system.	Critical
S.3.2 – Future API Integration	Not required in the first version but useful later.	Optional
S.4.1 – Reserving a Book	Core use case for users.	Critical
S.4.2 – Managing a Late Return	Important for enforcing library rules.	Important
S.4.3 – Managing My Account	Useful for user autonomy.	Important
S.4.4 – Adding a New Book	Necessary for catalog updates.	Critical
S.4.5 – Checking Availability Before Visiting	Improves experience but not strictly required.	Optional

S.6 Verification and acceptance criteria

Specification of the conditions under which an implementation will be deemed satisfactory (see Handbook).

S.6.1 **Acceptance Test 1 - Loan Enforcement** System must prevent checkout when:

```
a) Patron has 5+ active loans.
b) Patron has overdue items.
c) Book copy is already checked out.
```

Project

The Project book describes all the constraints and expectations not about the system itself, but about how to develop and produce it.

P.1 Roles and Personnel

Main responsibilities and required qualifications for the project team.

P.1.1 **Project Manager** Oversees the planning and progress of the project. Ensures deadlines are met and communication flows between team members. Should have skills in coordination, scheduling, and basic knowledge of library processes.

P.1.2 **Requirements Analyst**

Collects and structures requirements based on input from librarians, users, and institutional goals. Needs strong communication, analytical thinking, and basic understanding of library workflows.

P.1.3 **UX/UI Designer**

Designs interfaces that are intuitive for users and staff. Must understand accessibility, responsive design, and typical behaviors of library users.

P.1.4 **Full-Stack Developer**

Implements the system's functionalities across front-end and back-end. Requires skills in web development, database management, and integration of user-facing features with core logic.

P.1.5 **DevOps Engineer**

Sets up and maintains the infrastructure, automates deployment workflows, and ensures that changes can be integrated and delivered continuously. Should be skilled in cloud services, container technologies, and CI/CD tools.

P.1.6 **Librarian Consultant**

Provides domain knowledge: lending rules, cataloging standards, and user needs. Helps validate whether the system aligns with real library practices. Ideally someone with hands-on library experience.

P.1.7 **Requirements Editor / Technical Writer**

Maintains the AsciiDoc documents, ensures clarity and coherence, and prepares the final export to PDF. Requires writing skills and basic familiarity with version control systems like Git.

P.2 Imposed technical choices

Any a priori choices binding the project to specific tools, hardware, languages or

other technical parameters (see Handbook).

The project is bound to specific technologies and design principles that will guide the development of the Library Management System (LMS).

P.2.1 **Java 17 :** Chosen as the primary programming language for back-end development due to its robustness, strong typing, and ecosystem support.

P.2.2 **Spring Boot :** Main framework for building the application. It simplifies the creation of REST APIs, integrates well with databases, and supports dependency injection out of the box.

P.2.3 **PostgreSQL :** Relational database system used to store data about users, books, loans, and reservations. Offers strong ACID compliance and is open source.

P.2.4 **Docker :** Used to containerize the application for easier deployment and environment consistency across development, testing, and production.

P.2.5 **React.js :** Front-end library chosen to develop a responsive and user-friendly web interface for both users and librarians.

P.2.6 **Maven :** Build and dependency management tool for compiling, testing, and packaging the Java backend.

P.2.7 **JUnit 5 and Mockito :** Used for unit and integration testing of backend services to ensure reliability and test coverage.

P.2.8 **RESTful APIs :** All communication between front-end and back-end is based on REST principles for simplicity and interoperability.

P.2.9 **Git + GitHub :** Used for version control and collaboration. All code is stored and managed through GitHub with branching strategies and CI pipelines.

P.3 Schedule and Milestones

List of tasks to be carried out and their scheduling (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

This section outlines the major phases and expected milestones for the development of the Library Management System (LMS). Each milestone marks the delivery of a significant feature or stage of the project.

Milestone ID	Description	Target Date
M1	**Project Setup** Initialize Git repository, define technical stack (Java, Spring Boot, React, PostgreSQL), configure Docker, and prepare project documentation.	Week 1

Milestone ID	Description	Target Date
M2	**Domain Modeling and Architecture Design** Design the system's core domain model using DDD and define high-level architecture (Hexagonal + REST APIs). Create database schema and component structure.	Week 2
M3	**User & Authentication Module** Implement user registration, login, roles (user/librarian), and basic access control with JWT or session management.	Week 3
M4	**Catalog Management** Implement book catalog: CRUD for books, metadata management, search and filtering capabilities.	Week 5
M5	**Loan & Reservation System** Develop features to allow book borrowing, returning, reservation queues, overdue handling, and validation rules.	Week 7
M6	**Notification System** Set up email and in-app notifications for due dates, reservation availability, and late return warnings.	Week 8
M7	**Librarian Admin Dashboard** Create an admin panel for librarians to manage books, users, and lending policies.	Week 9
M8	**Front-end Integration** Develop and connect React-based UI with all back-end endpoints. Ensure responsiveness and role-based views.	Week 10
M9	**Testing & QA** Conduct unit, integration, and UI testing (JUnit, Mockito, Cypress). Run performance and usability tests.	Week 11
M10	**Final Deployment & Documentation** Deploy application using Docker. Deliver user manual, API documentation, and technical report.	Week 12

P.4 Tasks and Deliverables

Details of individual tasks listed under P.3 and their expected outcomes (see Handbook).

This chapter should not be empty (following the *Minimum Requirements Outcome Principle*, p.27 of the Handbook).

This section details the main tasks associated with each milestone from the project schedule and the corresponding deliverables.

Task ID	Description	Deliverable
T1.1	Set up GitHub repository, initialize folder structure, add README and project license.	Repository initialized with documentation baseline.
T1.2	Define the technical stack and tools (Java, Spring Boot, PostgreSQL, React, Docker, GitHub Actions).	Technical stack document in `/docs/`.
T2.1	Model domain entities (User, Book, Loan, Reservation) using DDD principles.	UML class diagram + Entity definitions in code.
T2.2	Design the system architecture (Hexagonal + REST + PostgreSQL + CI).	Architecture overview document (`architecture.adoc`).
T3.1	Implement user registration, authentication, and role-based access (JWT or sessions).	Working API for login/signup + user DB schema.
T4.1	Develop book catalog CRUD (create, read, update, delete).	REST endpoints for books + integration tests.
T4.2	Implement search and filter functionality.	Search API + basic UI component for catalog.
T5.1	Implement book borrowing and return workflow.	Loan controller, rules engine, unit tests.
T5.2	Manage reservations with FIFO queue and expiration logic.	Reservation service + edge-case tests.
T6.1	Add email notification service (due dates, reservations).	Mailer module + mock email tests.
T7.1	Build admin dashboard with views for managing users and books.	React UI pages for librarians + role restrictions.
T8.1	Connect React front-end to backend APIs using Axios or Fetch.	Front-end forms and data rendering working for MVP.
T9.1	Write and run JUnit and Mockito tests for backend.	Test coverage report (>80%) + CI status badge.
T9.2	Perform manual and automated UI tests (e.g., Cypress).	Usability test report + bug list (if any).
T10.1	Containerize application with Docker Compose (backend + frontend + DB).	`docker-compose.yml` + deployment instructions.
T10.2	Write user guide, API documentation, and installation manual.	`/docs/` folder with complete documentation in AsciiDoc or Markdown.

P.5 Required technology elements

External systems, hardware and software, expected to be necessary for building the system (see Handbook).

P.5.1 **Barcode Scanners** Hardware for quick book copy identification during check-in/check-out

P.6 Risk and Mitigation Analysis □□

Potential obstacles to meeting the schedule of P.4, and measures for adapting the plan if they do arise (see Handbook).

This section identifies potential risks that may affect the successful completion of the project, particularly with respect to the tasks and milestones in P.4, along with mitigation strategies.

Risk	Description	Mitigation Strategy
R1 – Technical Complexity	Integrating multiple technologies (Spring, React, PostgreSQL, Docker) may cause unexpected bugs or delays.	Begin with a small proof of concept for each component. Plan extra buffer time in early sprints.
R2 – Lack of familiarity with tools	Team members may lack experience with some parts of the stack.	Allocate time for guided onboarding, tutorials, and pair programming during the first two weeks.
R3 – Feature creep	New feature ideas might be added mid-project, risking delay or scope loss.	Stick to a defined MVP. Any additional features are moved to a backlog for post-M10 consideration.
R4 – Insufficient test coverage	Lack of automated tests could lead to regressions and unstable releases.	Define test tasks per milestone (P.4) and enforce CI checks on every commit.
R5 – Incomplete requirements	Some functional details might be misunderstood or missing at implementation time.	Validate requirements early with librarian feedback. Update documentation iteratively.
R6 – Integration delays	Back-end and front-end teams might fall out of sync during API integration.	Use API contracts (e.g., Swagger/OpenAPI) and mock servers to allow parallel development.
R7 – Team availability or dropout	Team members may become unavailable due to personal or academic reasons.	Distribute knowledge and responsibilities. Maintain clear documentation to ease handovers.
R8 – Deployment issues	Errors may arise during deployment or Docker configuration.	Test deployments regularly in staging environment using GitHub Actions.

P.7 Requirements process and report

Initially, description of what the requirements process will be; later, report on its steps (see Handbook).

P.7.1 **Process Steps:**

1. Stakeholder interviews with librarians

2. Observation of current manual processes