CAPRA Report — Design Proposals

Visual comparison of section layouts

Automated Proposal Generator

February 25, 2026

Contents

1 Summary

1.1 Proposal A — Document Overview only (clean text)

Summary

This document presents a software engineering project for a **library management system** designed to handle book cataloging, user registration, and loan operations. The system is architected using a layered approach with service, persistence, and UI components, implemented in Java with a relational database backend. The project demonstrates a structured approach to requirements analysis, system design, and testing, with comprehensive use-case modeling and architectural documentation.

1.1 Proposal B — Document Overview + inline stats

Summary

This document presents a software engineering project for a **library management system** designed to handle book cataloging, user registration, and loan operations. The system is architected using a layered approach with service, persistence, and UI components, implemented in Java with a relational database backend. The project demonstrates a structured approach to requirements analysis, system design, and testing, with comprehensive use-case modeling and architectural documentation.

Issues found: **6** (HIGH: 0, MEDIUM: 6, LOW: 0) — Features: 5/7 present

1.1 Proposal C — Metrics box + Document Overview

Summary

Audit Overview

Total issues: **6** — HIGH: **0** MEDIUM: **6** LOW: **0**

Features: 5 present / 2 partial / 0 absent (avg. coverage: **84%**)

This document presents a software engineering project for a **library management system** designed to handle book cataloging, user registration, and loan operations. The system is architected using a layered approach with service, persistence, and UI components, implemented in Java with a relational database backend.

2 Architecture Analysis (new section)

2.1 Proposal A — LLM description + description list

Architecture Analysis

The system follows a **layered architecture** with three main tiers: a JavaFX-based presentation layer (controllers), a service layer (business logic), and a DAO-based persistence layer backed by MySQL. Session management uses a singleton pattern via `Sessione` class. The architecture employs MVC for the UI and DAO pattern for data access.

ARCH-001 — medium

Problem: Service layer partially mixes business logic with UI concerns. Some service methods reference JavaFX alert dialogs directly, breaking separation of concerns.

Suggestion: Move all alert/dialog logic to the controller layer. Service methods should return result objects or throw exceptions; controllers translate those into UI feedback.

ARCH-002 — medium

Problem: Session management via singleton contradicts stated support for concurrent user access. The `Sessione` class holds a single user reference, making multi-session scenarios impossible.

Suggestion: Either clarify that the system is single-user (desktop app) and remove the concurrent access claim, or redesign session management to support multiple sessions (e.g., session tokens).

2.1 Proposal B — LLM description + itemize compact

Architecture Analysis

The system follows a **layered architecture** with three main tiers: a JavaFX-based presentation layer (controllers), a service layer (business logic), and a DAO-based persistence layer backed by MySQL. Session management uses a singleton pattern via `Sessione` class.

- **ARCH-001** [MEDIUM] — Service layer partially mixes business logic with UI concerns. Some service methods reference JavaFX alert dialogs directly.

  Suggestion: Move all alert/dialog logic to the controller layer.

- **ARCH-002** [MEDIUM] — Session management via singleton contradicts stated concurrent access support.

  Suggestion: Clarify single-user scope or redesign session management.

2.1 Proposal C — LLM description + compact table + details

Architecture Analysis

The system follows a **layered architecture** with three main tiers: a JavaFX-based presentation layer, a service layer, and a DAO-based persistence layer backed by MySQL.

ID	Severity	Problem
ARCH-001	MEDIUM	Service layer mixes logic with UI concerns
ARCH-002	MEDIUM	Session mgmt contradicts concurrency claim

ARCH-001 Some service methods reference JavaFX alert dialogs directly, breaking separation of concerns. **Suggestion:** Move dialog logic to controllers; services should return results or throw exceptions.

ARCH-002 The `Sessione` singleton holds a single user reference, making concurrent sessions impossible. **Suggestion:** Clarify single-user scope or redesign with session tokens.

3 Use Case Analysis

3.1 Proposal A — Only UCs with problems, others listed at end

Use Case Analysis

The document describes **17 use cases**, of which 4 have structured templates. 2 use cases present issues requiring attention.

UC-2 — Effettua prestito **Problem:** Business rule on max 3 concurrent loans not reflected in use case template. The alternative course does not describe what happens when the limit is exceeded.
Suggestion: Update UC-2 to explicitly state the 3-loan limit and describe the rejection flow in the alternative course.

UC-3 — Aggiunge libro **Problem:** Actor "Bibliotecario" inconsistent with "Biblioteca" used in UC diagram. This creates ambiguity about who performs the operation.
Suggestion: Unify actor naming across all diagrams, templates, and controller references.

Use cases without issues: UC-1, UC-4, UC-5, UC-6, UC-7, UC-8, UC-9, UC-10, UC-11, UC-12, UC-13, UC-14, UC-15, UC-16, UC-17

3.1 Proposal B — Split with/without template, problems inline

Use Case Analysis

The document describes **17 use cases**.

Use Cases with Structured Template

UC-1 — Registrazione ✓ No issues identified.

UC-2 — Effettua prestito

MEDIUM Max loan constraint missing
Problem: The alternative course does not describe the 3-loan limit enforcement.
Suggestion: Add the constraint and rejection flow to the alternative course.

UC-3 — Aggiunge libro

MEDIUM Actor naming inconsistency
Problem: "Bibliotecario" vs "Biblioteca" across diagrams and text.
Suggestion: Unify actor naming throughout the document.

UC-4 — Conclude Prestito ✓ No issues identified.

Use Cases without Structured Template

UC-5 through UC-17 are referenced in the use-case diagram but lack a formal template. No additional issues were identified beyond the absence of structured descriptions.

3.1 Proposal C — Only UCs with problems, compact list at end

Use Case Analysis

The document describes **17 use cases** (4 with structured template).

UC-2 — Effettua prestito
▷ Business rule on max loans not in template
→ Add constraint to alternative course

UC-3 — Aggiunge libro
▷ Actor inconsistency: "Bibliotecario" vs "Biblioteca"
→ Unify actor naming across diagrams and templates

The remaining 15 use cases do not present significant issues: UC-1, UC-4, UC-5, UC-6, UC-7, UC-8, UC-9, UC-10, UC-11, UC-12, UC-13, UC-14, UC-15, UC-16, UC-17

4 Testing Analysis

4.1 Proposal A — Strategy description + list of uncovered UCs

Testing Analysis

Testing Strategy

The project uses **JUnit** for unit tests and **TestFX** for UI tests. Unit tests cover core business operations (loan reservations, cancellations, ISBN retrieval, availability checks). UI tests verify login flows, field validation, and book reservation interactions. Error scenarios include invalid logins, field validation failures, unavailable books, and exceeded loan limits.

Missing Test Coverage

The following use cases lack explicit test coverage:

- UC-3 — Aggiunge libro: no unit or UI test for adding books
- UC-4 — Conclude Prestito: no test for loan conclusion
- UC-6 — Modifica account: no test for profile update
- UC-7 — Elimina account: no test for account deletion
- UC-8 — Visualizza catalogo: browsing/search not tested
- UC-13 — Effettua commento su opera: comment creation untested
- UC-14 — Effettua commento su volume: volume comments untested
- UC-15 — Visualizza commenti: comment listing untested
- UC-16 — Modifica libro: book editing untested
- UC-17 — Rimuove libro: book removal untested

Suggestion: Add at least one test per uncovered UC, starting from the most critical business operations (UC-3, UC-4).

4.1 Proposal B — Strategy + compact mapping table

Testing Analysis

Testing Strategy

The project uses **JUnit** for unit tests and **TestFX** for UI tests. Coverage focuses on loan management operations and user registration/login.

Test Coverage Map

UC	Name	Test	Test Methods
UC-1	Registrazione	✓	testEmailNonValidaMostraErrore
UC-2	Effettua prestito	✓	testPrenotaEAnnullaPrestito
UC-3	Aggiunge libro	×	—
UC-4	Conclude Prestito	×	—
UC-5	Login	✓	testLoginInvalido, testLoginValido

UC-6	Modifica account	×	—
UC-7	Elimina account	×	—
	[...remaining UCs...]		

Suggestion: Prioritize tests for UC-3 and UC-4 (core library operations).

4.1 Proposal C — Strategy + single grouped issue

Testing Analysis

Testing Strategy

The project uses **JUnit** for unit tests and **TestFX** for UI tests. Unit tests verify core business operations. UI tests cover login and registration validation.

Test Issues

TST-001 — medium — 10 use cases lack test coverage

Problem: The document describes tests only for UC-1, UC-2, UC-5, UC-9, UC-10, UC-11, UC-12. The remaining 10 use cases (UC-3, UC-4, UC-6, UC-7, UC-8, UC-13, UC-14, UC-15, UC-16, UC-17) have no associated test cases.

Suggestion: Create a traceability table mapping each UC to its tests. Implement at least one test per critical use case, prioritizing UC-3 (Aggiunge libro) and UC-4 (Conclude Prestito).

5 Traceability Analysis

5.1 Proposal A — Description list with status + problem/suggestion

Traceability Analysis

This section analyzes traceability between use cases, their design implementation, and test coverage.
Of **17** traced use cases: 7 fully covered, 10 with gaps.

Traceability Gaps

UC-3 — Aggiunge libro
Design: ✓ Test: ×

Gap: No unit or UI tests documented for adding a new book to the catalog.
Suggestion: Add a JUnit test for `LibroService.aggiungiLibro()` verifying the complete book addition flow.

UC-4 — Conclude Prestito
Design: ✓ Test: ×

Gap: No explicit test for loan conclusion.
Suggestion: Add a test verifying `PrestitoService` loan completion and status update.

UC-6 — Modifica account
Design: ✓ Test: ×

Gap: No tests for profile update via `ProfiloController`.
Suggestion: Add unit test for `UtenteService.modificaUtente()`.

[...remaining 7 gaps follow same pattern...]

Fully Traced Use Cases

The following use cases have complete traceability (design + test coverage):

- UC-1 — Registrazione
- UC-2 — Effettua prestito
- UC-5 — Login
- UC-9 — Verifica disponibilità opera
- UC-10 — Recupero ISBN
- UC-11 — Cancella prestito
- UC-12 — Prolunga prestito

5.1 Proposal B — Compact table (✓/×) + gap summary

Traceability Analysis

UC	Name	Design	Test
UC-1	Registrazione	✓	✓
UC-2	Effettua prestito	✓	✓
UC-3	Aggiunge libro	✓	×
UC-4	Conclude Prestito	✓	×
UC-5	Login	✓	✓
UC-6	Modifica account	✓	×
UC-7	Elimina account	✓	×
UC-8	Visualizza catalogo	✓	×
UC-9	Verifica disponibilità opera	✓	✓
UC-10	Recupero ISBN	✓	✓
UC-11	Cancella prestito	✓	✓
UC-12	Prolunga prestito	✓	✓
UC-13	Commento su opera	✓	×
UC-14	Commento su volume	✓	×
UC-15	Visualizza commenti	✓	×
UC-16	Modifica libro	✓	×
UC-17	Rimuove libro	✓	×

Summary: All 17 use cases have design references. 10 lack test coverage.
Suggestion: Prioritize adding tests for UC-3, UC-4 (core operations), then UC-6, UC-7 (account management).

5.1 Proposal C — Grouped by gap type

Traceability Analysis

Of 17 use cases: all have design references, 7 have test coverage.

Missing Test Coverage (10 use cases)

UC	Name	Suggested Test
UC-3	Aggiunge libro	Test `aggiungiLibro()` flow
UC-4	Conclude Prestito	Test loan completion
UC-6	Modifica account	Test profile update
UC-7	Elimina account	Test account deletion
UC-8	Visualizza catalogo	Test catalog browsing
UC-13	Commento su opera	Test comment creation
UC-14	Commento su volume	Test volume comment
UC-15	Visualizza commenti	Test comment listing
UC-16	Modifica libro	Test book editing
UC-17	Rimuove libro	Test book removal

Suggestion: Create a traceability matrix and add at least one test per critical UC.

Fully Covered

UC-1, UC-2, UC-5, UC-9, UC-10, UC-11, UC-12

6 Missing Features

6.1 Proposal A — Description list with Problem/Suggestion

Missing Features

5 of 7 features are fully present. 2 require attention.

Unit testing framework — partial (60%)

Problem: Tests hit the real database with no mocking framework. No explicit separation between unit and integration tests. Dependency isolation is missing.

Suggestion: Introduce Mockito for unit test isolation. Add a brief section distinguishing unit tests (mocked dependencies) from integration tests (real DB).

Use Case Documentation — partial (60%)

Problem: Pre-conditions and post-conditions are not specified for any use case. Relationships between use cases (`«include»`) are shown in diagrams but not explained in text.

Suggestion: Add pre/post-conditions to each UC template. Add a paragraph explaining UC relationships.

6.1 Proposal B — Summary table + details below

Missing Features

Feature	Status	Coverage
Unit testing framework	PARTIAL	60%
Use Case Documentation	PARTIAL	60%
5 other features	PRESENT	100%

Unit testing: Tests lack dependency isolation (no mocking). → Introduce Mockito and clarify test categories.

Use Cases: Pre/post-conditions missing from templates. → Add conditions and explain `«include»` relationships.

6.1 Proposal C — Compact itemize

Missing Features

5 of 7 expected features are fully present. 2 need improvement:

- **Unit testing framework** (60%): Tests lack dependency isolation (no mocking). → Separate unit from integration tests and introduce Mockito.

- **Use Case Documentation** (60%): Pre/post-conditions missing. → Add them to each UC template for improved traceability.

7 Action Plan

7.1 Proposal A — Table with priority, effort, impact

Action Plan

The following actions are recommended in order of priority.

#	Issue	Action	Effort	Impact
1	ARCH-001	Separate UI logic from services	Medium	Medium
2	ARCH-002	Clarify session management approach	Medium	Medium
3	REQ-002	Add missing UC templates	Low	Medium
4	TST-001	Create UC → Test traceability	Low	Medium
5	REQ-003	Reflect max-loan rule in UC-2	Low	Medium

Effort: estimated time to implement the fix. **Impact:** expected improvement on document quality.

7.1 Proposal B — Numbered list (no table)

Action Plan

1. Separate UI concerns from the service layer (ARCH-001) — Medium effort, Medium impact
2. Clarify session management approach (ARCH-002) — Medium effort, Medium impact
3. Add templates for missing use cases (REQ-002) — Low effort, Medium impact
4. Create UC → Test traceability table (TST-001) — Low effort, Medium impact
5. Reflect max-loan rule in UC-2 (REQ-003) — Low effort, Medium impact

7.1 Proposal C — Remove action plan entirely

Action Plan

[This section would be removed entirely. All suggestions are already provided inline within each section (Architecture, Use Cases, Testing, Traceability). Keeping a separate Action Plan section creates redundancy.]