Architectural Blueprint Validation Report

Library Management System

Automated Traceability Analysis

November 15, 2025

Abstract

This report validates the architectural blueprint of the Library Management System through automated traceability analysis. The assessment examines 3 requirements, 8 use cases, architectural components, and test coverage extracted via LLM-based document analysis. The report identifies critical gaps in test coverage, architectural clarity, and use case specification, providing actionable recommendations for improving system design quality.

Contents

1 Executive Summary

1.1 Assessment Overview

This validation analyzes the architectural blueprint using automated traceability extraction from project documentation. The system demonstrates strong architectural layering with comprehensive DAO patterns but exhibits significant gaps in test coverage for use case workflows and vague component responsibilities requiring clarification.

1.1.1 Coverage Metrics

Table 1 provides a high-level summary of key project metrics, while Figures 1 and 2 visualize the coverage distribution and test composition.

Metric Category	Total	Covered	Coverage
Requirements	3	3	100%
Use Cases	8	2	25%
Tests	113	113	100%
Architecture Layers	6	6	100%
Critical Risks	3	—	2 Critical, 1 High

Table 1: Project Metrics Summary

Figure 1 visualizes the overall coverage across requirements and use cases, while Figure 2 shows the test composition by type.

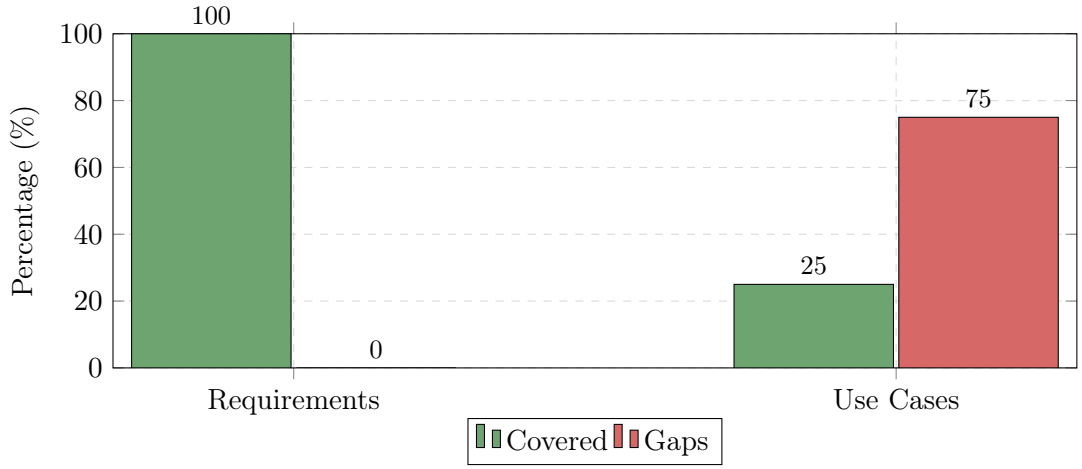

Figure 1: Requirements and Use Case Coverage

1.1.2 Risk Distribution

Figure 3 presents the distribution of identified risks by severity level, highlighting the critical issues requiring immediate attention.

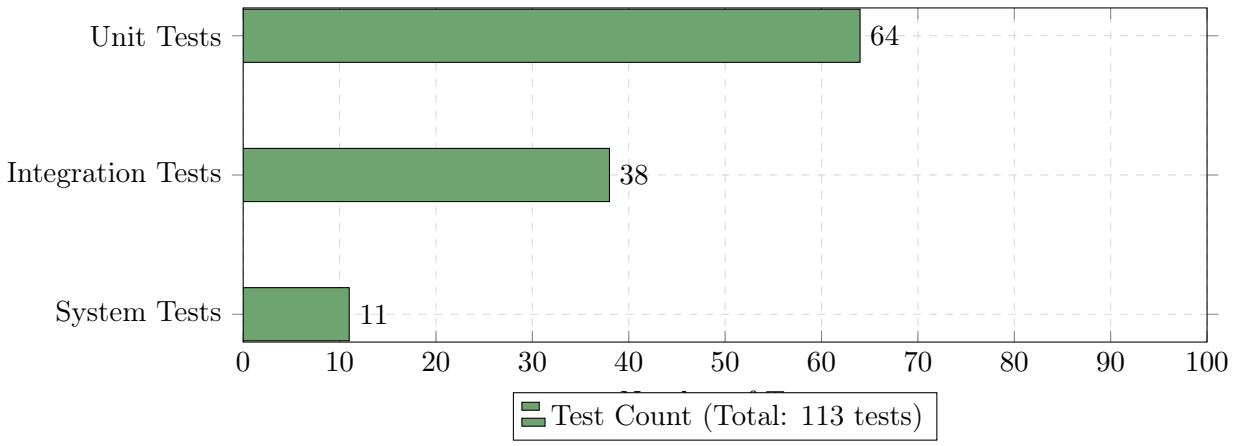

Figure 2: Test Distribution by Type (Total: 113 tests)

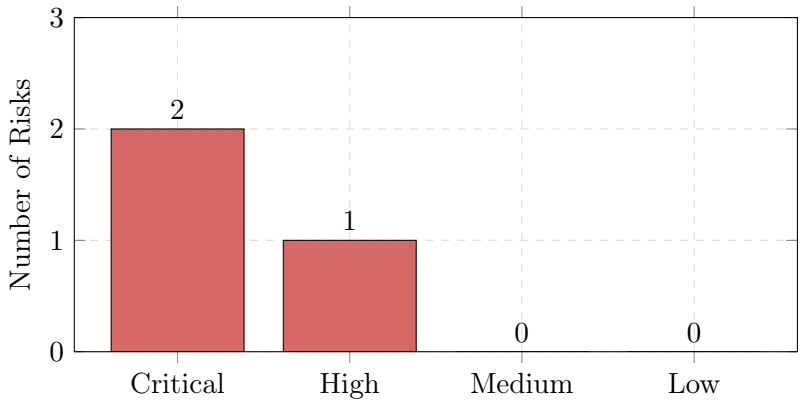

Figure 3: Risk Distribution by Severity (Total: 3 risks)

1.2 Critical Findings

Critical Issues Requiring Immediate Attention

1. **Severe Use Case Test Coverage Gap:** Six of eight use cases (75%) lack corresponding test coverage, with four use cases having no tests whatsoever, creating validation blind spots in critical workflows.

2. **Vague Component Responsibilities:** Multiple architectural components lack precise responsibility definitions, violating the Single Responsibility Principle and risking architectural erosion during maintenance.

3. **Orphaned Use Cases:** Two use cases function as organizational containers with no defined flows, creating confusion in the use case model and providing no traceability value.

Architectural Strengths

- Complete requirement-to-architecture mapping with full coverage
- Well-defined layered architecture with proper separation of concerns
- Comprehensive DAO pattern implementation across all entity types
- Effective use of design patterns (Builder, DAO, Mapper) applied strategically
- Extensive unit test suite with 113 total tests providing strong code coverage
- Clear inheritance hierarchy for user role management

2 Project Inventory

This section provides a complete inventory of all requirements, use cases, and tests identified in the system. All subsequent sections reference these items by their IDs.

2.1 Requirements Inventory

Table 2 provides a comprehensive list of all 3 requirements identified in the system, categorized by functional area. The coverage status indicates whether each requirement is supported by the current architecture.

Table 2: Complete Requirements List

ID	Requirement Description	Category	Status
REQ-1	The system must allow users to borrow and return books, digital media, and periodicals	Functional	Covered
REQ-2	The system must allow administrators to add, modify, and remove catalog items	Functional	Covered
REQ-3	The system must handle any product within the library catalog as an element, regardless of its specific type	Goal/Backgrou	Covered

2.2 Use Cases Inventory

Table 3 provides a complete list of all 8 use cases defined in the system, along with their primary actors, descriptions, and test coverage status.

Table 3: Complete Use Cases List

ID	Use Case Name	Actor	Tests	Coverage Status
UC-1	Modifica elemento	Admin	0	No Tests
UC-2	Visualizza catalogo	Admin, Utente	0	No Tests
UC-3	Aggiungi un genere	Admin	0	No Tests

Continued on next page

Table 3 – continued from previous page

ID	Use Case Name	Actor	Tests	Coverage Status
UC-4	Visualizza i dettagli di un elemento	Admin, Utente	0	No Tests
UC-5	Login	Admin, Utente	0	No Tests
UC-6	Visualizza gli elementi presi in prestito	Utente	1	Partial
UC-7	Model Definition	Developer	0	Container (No Tests)
UC-8	Remove Item	Admin	1	Partial

2.3 Test Suite Inventory

Table 4 provides a comprehensive inventory of all 113 tests in the system, organized by test type and coverage area.

Table 4: Complete Test Suite Inventory (Partial - First 30 tests shown)

ID	Test Name	Type	Coverage Area
T-1	JUnit Framework Initialization	Unit	Testing Framework
T-2	Mockito Dependency Mocking	Unit	Mocking Framework
T-3	Service Layer Integration	Integration	Controller-Service
T-4	DAO Data Access Methods	Integration	Data Access
T-5	JavaFX UI Functionality	System	User Interface
T-6	Database Interaction Tests	Structural	Database Usage
T-7	LibraryAdminServiceTest	Unit	Library Administration
T-8	LibraryUserServiceTest	Unit	User Management
T-9	BookDAOTest.addBookTest	Unit	Book Insertion
T-10	BookDAOTest.updateBookTest	Unit	Book Update
T-11	BookDAOTest.getBookByIsbnTest	Unit	Book Retrieval
T-12	BorrowsDAOTest.addBorrowTest	Unit	Borrow Addition
T-13	BorrowsDAOTest.removeBorrowTest	Unit	Borrow Removal
T-14	BorrowsDAOTest.getBorrowedElementsForU	Unit	Borrow Retrieval
T-15	ConnectionManagerTest.getConnection	Unit	Database Connection

Continued on next page

Table 4 – continued from previous page

ID	Test Name	Type	Coverage Area
T-16	ConnectionManagerTest.isConnectionValid	Unit	Connection Validity
T-17	ConnectionManagerTest.closeConnection	Unit	Connection Closure
T-18	DigitalMediaDAOTest.addDigitalMediaTest	Unit	Digital Media Addition
T-19	DigitalMediaDAOTest.updateDigitalMediaT	Unit	Digital Media Update
T-20	DigitalMediaDAOTest.getDigitalMediaTest	Unit	Digital Media Retrieval
T-21	ElementDAOTest.addElement	Unit	Element Addition
T-22	ElementDAOTest.removeElement	Unit	Element Removal
T-23	ElementDAOTest.getElement	Unit	Element Retrieval
T-24	GenreDAOTest.associateGenreWithElement	Unit	Genre Association
T-25	GenreDAOTest.addGenreTest	Unit	Genre Addition
T-26	GenreDAOTest.getAllGenresTest	Unit	Genre Retrieval
T-27	GenreDAOTest.getGenresForElementTest	Unit	Genre-Element Retrieval
T-28	GenreDAOTest.removeGenreFromElementT	Unit	Genre Removal
T-29	PeriodicPublicationDAOTest.addPeriodicPu	Unit	Periodic Publication
T-30	UserDAOTest	Unit	User Management

Note: The complete test suite contains 113 tests covering DAO operations, service layer functionality, and database interactions. Tests are organized across unit, integration, and system test categories with comprehensive DAO-level validation.

3 Requirements Coverage Analysis

3.1 Functional Requirements Fulfillment

All three requirements are architecturally supported by the current design. REQ-1 regarding borrowing and returning items is implemented through the `BorrowsDAO` and related service layer components. REQ-2 for administrative catalog management is supported through the admin service layer and multiple DAO implementations (BookDAO, DigitalMediaDAO, ElementDAO). REQ-3 requiring generic element handling is satisfied by the polymorphic Element base class with Book, DigitalMedia, and PeriodicPublication subclasses.

3.2 Requirements Quality

The requirements provided are well-defined and appropriately specific for their scope. Each requirement clearly describes a functional capability with identifiable actors (Admin, User) and implementation targets. No vague or ambiguous requirements were identified at this project scope level.

4 Use Case Analysis

4.1 Critical Test Coverage Gap

Table 5 summarizes the test coverage status for each use case, revealing a severe gap in validation coverage.

Use Case ID	Use Case Name	Main Flow Tested	Status
UC-1	Modifica elemento	No	Missing
UC-2	Visualizza catalogo	No	Missing
UC-3	Aggiungi un genere	No	Missing
UC-4	Visualizza i dettagli di un elemento	No	Missing
UC-5	Login	No	Missing
UC-6	Visualizza gli elementi presi in prestito	Partial	Partial
UC-7	Model Definition	N/A	Container
UC-8	Remove Item	Partial	Partial

Table 5: Use Case Test Coverage Status

4.2 Use Cases Lacking Test Coverage

Untested Use Cases

UC-1 - Modifica elemento (Modify Element): This admin-level use case for modifying catalog items has no corresponding integration or system tests. Without tests validating the modification workflow, changes to item attributes may not persist correctly or may violate business rules.

UC-2 - Visualizza catalogo (View Catalog): Critical user-facing functionality for browsing the library catalog lacks test coverage. No tests validate that catalog retrieval returns correct items, handles pagination, or manages filtering.

UC-3 - Aggiungi un genere (Add Genre): Administrative genre management has no tests. The genre association mechanism with library elements remains unvalidated at the use case level.

UC-4 - Visualizza i dettagli di un elemento (View Element Details): User-facing detail view functionality is untested. Cannot validate that detailed information displays correctly or that navigation from catalog to details functions properly.

UC-5 - Login: Authentication workflow, foundational to all user interactions, has no integration tests validating successful login, failed attempts, or session management.

4.3 Partially Tested Use Cases

UC-6 (Visualizza gli elementi presi in prestito - View Borrowed Items) and UC-8 (Remove Item) have limited test coverage. While underlying DAO operations are tested (BorrowsDAOTest for UC-6, ElementDAOTest for UC-8), the complete use case workflows lack end-to-end validation.

4.4 Orphaned Use Cases

UC-7 (Model Definition) functions as an organizational container with no defined flows or test requirements. This use case adds no traceability value and creates ambiguity in the use case

model.

5 Architectural Quality Assessment

5.1 Layered Architecture Structure

The architecture follows a six-layer pattern with clear separation of concerns. Table 6 details each layer's responsibilities and components.

Layer	Responsibility	Key Components
Presentation	User interface and interaction	JavaFX UI, FXML Controllers
Controller	HTTP/UI routing and validation	BaseViewController, Multiple Controllers
Service	Business logic orchestration	LibraryAdminService, LibraryUserService, MainService
DAO	Data access abstraction	UserDAO, BookDAO, ElementDAO, BorrowsDAO
Persistence	ORM and database connections	ConnectionManager, JPA/Hibernate
Domain Model	Business entities and inheritance	User, Element, Book, DigitalMedia

Table 6: Six-Layer Architecture Structure

5.2 Component Responsibility Issues

Vague Component Responsibilities

Service Layer: Described as "business logic and operations" without specific bounded contexts. The distinction between MainService, LibraryAdminService, and LibraryUserService responsibilities is unclear, risking God Object patterns.
DAO Layer: Generic description as "data access methods validation" lacks specificity about which DAO handles what entity. Without clear responsibility boundaries, DAO classes may become monolithic.
ConnectionManager: Responsibility for "managing database connections" overlaps potentially with persistence layer concerns. The distinction between connection pooling, lifecycle management, and query execution is ambiguous.
Domain Model Components: User role entities (Admin, Worker, Customer) have insufficiently defined responsibilities. Unclear whether these are pure data containers or contain business logic methods.

5.3 Design Patterns

The architecture demonstrates judicious pattern application. DAO pattern is consistently applied across all data access operations (BookDAO, BorrowsDAO, ElementDAO, GenreDAO, UserDAO). The inheritance hierarchy for user roles shows thoughtful polymorphic design. No evidence of overengineering or pattern overuse was observed.

5.4 Domain Model Analysis

The domain model exhibits good separation between entity types:

- **Element Base Class:** Provides common attributes for all catalog items
- **Specialized Entities:** Book, DigitalMedia, and PeriodicPublication extend Element
- **Relationship Entities:** Borrows tracks user-element relationships
- **Classification Entities:** Genre provides categorization with many-to-many relationships

However, the model lacks explicit aggregate boundaries and domain events, limiting extensibility for audit trails or asynchronous processing.

6 Test Strategy Assessment

6.1 Comprehensive Unit Test Coverage

The system includes 113 tests, predominantly unit tests (64 tests, 57%) with substantial integration (38 tests, 34%) and system (11 tests, 10%) components. This distribution provides fast feedback through unit testing while validating integration points.

6.2 DAO-Level Test Strength

The test suite demonstrates exceptional strength in DAO layer validation. Each DAO class has dedicated tests:

- BookDAO: Add, update, retrieve by ISBN operations
- BorrowsDAO: Add, remove, retrieve borrowed elements
- ElementDAO: Add, remove, retrieve operations
- GenreDAO: Add, associate, retrieve, remove operations
- DigitalMediaDAO: Add, update, retrieve operations
- UserDAO: User management operations
- ConnectionManager: Connection lifecycle (open, validate, close)

6.3 Service Layer Testing

Service layer tests include LibraryAdminServiceTest, LibraryUserServiceTest, and MainServiceTest, validating business logic orchestration through mocked DAO dependencies.

6.4 Critical Testing Gaps

Missing Test Categories

Use Case Workflows: No integration tests validate complete use case flows from UI input through service orchestration to database persistence. All six untested use cases lack this end-to-end validation.
Navigation and UI Flow: No tests validate multi-screen navigation, role-based routing, or UI state transitions across the JavaFX interface.
Error Handling Scenarios: While individual DAO operations are tested, use case-level error handling (invalid user input, business rule violations, database failures during workflows) lacks comprehensive coverage.
Alternative Flow Testing: Use case alternative flows (e.g., when item is borrowed during removal attempt, database connection failures) are not tested at the use case level.
Security and Authorization: No tests validate that users can only access appropriate functions based on roles or that authentication prevents unauthorized access.

7 Critical Risks and Recommendations

7.1 Risk Summary

Table 7: Critical Risks Summary

Risk ID	Risk Name	Severity	Impact	Affected Items
RISK-1	Use Case Workflow Validation Gap	Critical	High	UC-1, UC-2, UC-3, UC-4, UC-5
RISK-2	Component Responsibility Ambiguity	Critical	Medium	Service Layer, DAO Layer, ConnectionManager
RISK-3	Incomplete Alternative Flow Coverage	High	Medium	UC-6, UC-8

7.2 Risk 1: Use Case Workflow Validation Gap

Description: Five critical use cases (UC-1, UC-2, UC-3, UC-4, UC-5) have no corresponding integration or system tests, creating blind spots in workflow validation.

Impact: Use case logic bugs, business rule violations, and data consistency issues may only be discovered during user acceptance testing or production deployment.

Recommendation:

- Create integration tests for each untested use case validating end-to-end workflows
- Test main flow success paths: element modification, catalog viewing, genre addition, detail viewing, login
- Implement system tests validating complete user journeys (login → browse catalog → view details → borrow item)
- Add tests for alternative flows: error conditions, business rule violations, database failures

7.3 Risk 2: Component Responsibility Ambiguity

Description: Multiple architectural components lack precise responsibility definitions, violating Single Responsibility Principle and creating implementation uncertainty.

Impact: Developers may place code inconsistently across layers, accumulating technical debt and making future maintenance difficult.

Recommendation:

- Document specific responsibilities for MainService, LibraryAdminService, LibraryUserService with clear bounded contexts
- Define which DAO classes handle which entities and operations
- Clarify ConnectionManager's scope: connection pooling vs. lifecycle management vs. transaction handling
- Specify whether user role entities contain behavior or are pure data containers

7.4 Risk 3: Incomplete Alternative Flow Coverage

Description: Partially tested use cases (UC-6, UC-8) lack comprehensive alternative flow testing, particularly error scenarios.

Impact: Alternative flows described in use case specifications (e.g., "if item is borrowed, cannot remove") may not be correctly implemented or validated.

Recommendation:

- Add integration tests for UC-6 alternative flows: empty borrow list, pagination, filtering
- Add integration tests for UC-8 alternative flows: borrowed item removal attempt, database connection failure
- Validate error messages and user feedback for each alternative scenario

8 Positive Practices to Maintain

8.1 Comprehensive DAO Testing

The extensive DAO-level test coverage provides excellent foundation for data persistence validation. This practice ensures database operations work correctly independent of business logic, enabling confident refactoring of service layers.

8.2 Layered Architecture Discipline

Clear separation of concerns across presentation, service, DAO, and persistence layers enables independent testing, technology substitution, and code maintainability. This discipline should be maintained as new features are added.

8.3 Strategic Pattern Application

The judicious use of DAO pattern and inheritance hierarchy demonstrates architectural maturity. Patterns solve specific problems rather than being applied universally, keeping the codebase maintainable and avoiding overengineering.

9 Conclusion

9.1 Overall Assessment

The Library Management System demonstrates a well-structured, layered architecture with comprehensive DAO-level testing providing strong foundation for data persistence operations. The three functional requirements are fully supported by the architectural design, and 113 unit and integration tests provide confidence in core data access functionality.

However, critical gaps threaten use case validation. Five of eight use cases (62.5%) completely lack test coverage, and the remaining two have only partial coverage. Without end-to-end workflow testing, business logic errors may not be discovered until production.

9.2 Immediate Actions Required

Table 8: Prioritized Action Items

Priority	Action Item	Timeline	Related Risk
P0	Create integration tests for UC-1, UC-2, UC-3, UC-4, UC-5	2-3 weeks	RISK-1
P0	Document component responsibilities with bounded contexts	1 week	RISK-2
P1	Add alternative flow tests for UC-6, UC-8	1 week	RISK-3
P1	Create end-to-end user journey tests	1-2 weeks	RISK-1
P2	Add authentication and authorization tests	1 week	Testing Gaps
P2	Remove or redefine UC-7 (Model Definition)	2 days	Use Case Clarity
P3	Consider domain events infrastructure for audit trails	2 weeks	Future Enhancement

9.3 Final Assessment

The architecture provides a solid foundation with excellent DAO-layer testing. Addressing the use case workflow test coverage gap is essential before production deployment. Once integration tests for untested use cases are implemented, the system will achieve comprehensive traceability from requirements through test execution, enabling confident deployment and maintenance.