Table of Contents

1 Introduction

No explanation is needed here. Only complete the subsections.

1.1 Purpose

This SRS defines the requirements for "MyLibrary", a robust and extensible Java Swing desktop application with a MySQL backend, designed for comprehensive management of books read or to be read by users.

The document is intended for our instructors, developers, testers, maintainers, and stakeholders, ensuring a shared, detailed understanding of functional and non-functional requirements, mapped directly to our actual codebase.

The purpose of the development of the project for the developers is to serve as a learning platform for mastering the best architectural patterns, deepening understanding of design patterns, and applying industry best practices. It aims to deliver a high-quality, extensible project that can be continuously improved and used as a case study for educational and professional purposes.

1.2 Scope

1.2.1 a) Software Product Identification

The software product to be produced is called “MyLibrary”.

1.2.2 b) What The Software Product Will Do

“MyLibrary” is a desktop application implemented in Java (using Swing for GUI) and MySQL for data storage. The software allows users to manage a personal or institutional library of books, capturing which books a user has read or wishes to read. The application provides a user-friendly interface for interacting with the library and supports multiple user types, each with different access rights.

Key features include:

- User authentication: Users log in with a username and password. There are two user types: Type-1 (admin/full access) and Type-2 (limited access).
- Book management: Users can add, edit, delete, and view books, supplying all relevant attributes (title, author, year, cover image, number of pages, description, read status, rating, comments, release date).
- Author management: The application manages author records, ensuring each book is linked to a single author and maintaining the authors list accordingly.
- User management: User credentials and types are stored and hashed securely in the database.
- Favorites and wish lists: Users can rate books as favorites, view unread books, view favorite authors, and manage wish lists with release notifications.
- Search and information display: The application provides robust search features for books and authors, as well as detailed information displays.
- Book cover images: The application displays cover images associated with each book.

- Notifications: Upon login, users are notified about books in their wish list that will be released within a week.

The application maintains data persistence using JDBC to interface with a relational database, ensuring data integrity and supporting concurrent usage.

1.2.3 c) The Goal Of The Application Of The Software Being Specified

1.2.3.1 For Stakeholders & Users

The main goal of "MyLibrary" is to provide an efficient, user-friendly, and comprehensive desktop application for managing and tracking books, reading habits, and author information for individual users or small organizations.

It aims to:

- Simplify the tracking of books read, books to read, and wish lists.
- Allow users to rate books, leave comments, and keep personal notes.
- Help users easily search for books and authors, manage favorites, and receive timely notifications related to new releases on their wish list.
- Equip administrators with the ability to manage the entire library, while also supporting user-level restrictions for regular users.
- Encourage organized reading habits and improve accessibility to personal or small-scale institutional libraries through a visually intuitive and robust desktop interface.

1.2.3.2 For Developers

The codebase of the application aims for the developers and code reviewers:

- be a practical educational project demonstrating software engineering principles, design patterns, best practices, database integration, and GUI design in Java.
- show the Onion Architecture and Domain-Driven Design (DDD) principles effectively to ensure that the system is extensible, maintainable, and team-friendly. The architecture should support agile development by promoting clear communication, transparency with stakeholders, and the use of well-defined design diagrams to guide accurate and efficient implementation.

1.3 Definitions, acronyms, and abbreviations

Term/Acronym/Abbreviation	Definition
SRS	Software Requirements Specification – the document outlining the functional and non-functional requirements of the software system.

MyLibrary	The name of the Java Swing desktop application specified in this document, designed for managing users' personal or institutional book libraries.
JDBC	Java Database Connectivity – a Java API for connecting and executing queries with a relational database such as MySQL.
MySQL	An open-source relational database management system used as the data storage in this project.
GUI	Graphical User Interface – a visual interface through which users interact with the application. In this context, implemented using Java Swing.
Swing	A Java toolkit for building graphical user interfaces, used for creating the application's windows, forms, and components.
User	An individual who interacts with the "MyLibrary" application. Users are divided into two types: Type-1 (Administrator/full access) and Type-2 (Limited access).
Admin / Administrator (Type-1 User)	A user with full access to all features of the application, including book and author management.
Type-2 User	A user with restricted access, able to use only selected functionalities as defined in the requirements.
Book	An entity in the system representing a literary work, with attributes such as id, title, authorId, year, numberOfPages, cover, about.
UserBookState	An entity in the system representing a state of book for a user with attributes such as read status, rating, list of comments, and release date.
Comment	An entity in the system representing a comment of user for a book with attributes such as value.
Author	An entity representing a book's creator, with attributes such as id, name, surname, and website.

Read Status	An attribute indicating the reading status of a book for the user: 1 (read), 2 (not read), 3 (wish to read).
Rating	An integer value (1–5) representing how much the user liked a book; 0 if not rated.
Wish List	A feature allowing users to track books they wish to read, including notification of upcoming releases.
Favorite Book	A book that a user has read and rated as 4 or 5.
Favorite Author	An author for whom the user has at least 3 books in their library.
LoginFrame	The initial window for user authentication.
Entity	In software, an object domain that has a distinct identity.
Aggregate	A group of related objects that belong together and are treated as one unit.
Domain	The core business logic and rules of the application.
Value Object	An object that represents a descriptive aspect of the domain with no conceptual identity.
Domain Exception	An exception that represents a business rule violation in the domain layer.
Mapper	A component that converts between object representations and other representations. This can be inputs of users in DTO's, entities, row representation objects, etc...
User Context	The current user's session and security context.
User Context Holder	A utility to manage and access the current user context.
Unit of Work	A pattern to group one or more operations (usually database operations) into a single transaction.
Encryptor	A component that encrypts sensitive data (e.g., passwords).
Decryptor	A component that decrypts sensitive data.

DTO	Data Transfer Object – a simple object used to transfer data between different layers of the application.
Repository	A class or component responsible for managing the persistence and retrieval of entities in the database.
Onion Architecture	A software architectural pattern emphasizing a separation of concerns, with layers such as domain, application, and infrastructure.
Domain-Driven Design (DDD)	A software design approach focused on modeling software based on the domain's real-world concepts and rules.
SQL	Structured Query Language – a language used for managing data in a relational database.
JFrame	A top-level container provided by Java Swing for building application windows.
Stakeholder	Any individual or group with an interest in the success and operation of the MyLibrary application (e.g., instructors).
IDE	Integrated Development Environment – a software suite (such as IntelliJ IDEA or NetBeans) used to develop, debug, and maintain code.
UML	Unified Modeling Language – a standardized way to visualize the design of a system, used in this project for diagrams.

1.4 References

The SRS and development phase used a lot of resources, they are:

1. Project assignment and requirements as provided by Asst. Prof. Dr. Deniz Özsoyeller.
2. Visual Paradigm UML Tool - https://www.visual-paradigm.com/
3. Oracle, Swing Tutorial, docs.oracle.com - https://docs.oracle.com/javase/tutorial/uiswing/
4. Jetbrains Intellij UI Designer: https://www.jetbrains.com/help/idea/design-gui-using-swing.html
5. Roman Glushach, Understanding Hexagonal, Clean, Onion, and Traditional Layered Architectures: A Deep Dive, Medium.com - https://romanglushach.medium.com/understanding-hexagonal-clean-onion-and-traditional-layered-architectures-a-deep-dive-c0f93b8a1b96

6. Ritesh Kapoor, Onion Architecture, Medium.com - https://medium.com/expedia-group-tech/onion-architecture-deed8a554423
7. OriginMaster, A Soft Introduction to Domain-Driven Design: From Theory to Java Code Implementation — Part 2, Medium.com - https://medium.com/@ygnhmt/a-soft-introduction-to-domain-driven-design-from-theory-to-java-code-implementation-part-2-5aa7e1cfef65
8. Techie's Spot, Understanding Java Streams: A Beginner's Guide, Medium.com - https://medium.com/@TechiesSpot/understanding-java-streams-a-beginners-guide-5b76fbe5df98
9. Arjun, Unit Of Work Pattern, Medium.com - https://medium.com/@arjunarora171997/unit-of-work-pattern-290cd46ffcf4
10. Prince Singh, Java 8's — Consumer, Predicate, Supplier, and Function., medium.com - https://medium.com/javarevisited/java-8s-consumer-predicate-supplier-and-function-bbc609a29ff9
11. Nayana Weligalla, Optional in Java 8, medium.com - https://medium.com/@nweligalla/optional-in-java-8-5fbf12c90bfe
12. oopdev, How can i achieve active(current) UserContext in some classes?, stackoverflow.com - https://stackoverflow.com/questions/6876679/how-can-i-achieve-activecurrent-usercontext-in-some-classes
13. Ramanamuttuana, Thread Local in Java, medium.com - https://medium.com/%40ramanamuttana/thread-local-in-java-4951fae6ee56
14. Alexander Obregon, Java Generics: Type Safety and Performance, medium.com - https://medium.com/@AlexanderObregon/java-generics-type-safety-and-performance-a0864ca72a5a
15. Refactoring Guru - https://refactoring.guru/
16. MySQL Documentation, 14.7 Date And Functions, dev.mysql.com, https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html

1.5 Overview

The document is organized to clearly reflect the structure and behavior of the system, guiding developers, testers, and stakeholders through every aspect of the project.

- Section 1 introduces the project by defining its purpose, scope, key terms, references, and this overview.
- Section 2 outlines the design and implementation constraints, including hardware, technology stack, tools, and any restrictions that influence development.
- Section 3 specifies the system's requirements, including detailed functional requirements, performance metrics, software attributes such as security and usability, and use case analysis. This section also includes the relationships between actors and use cases as well as the use case diagram.
- Section 4 describes behavioral models by presenting sequence diagrams for selected use cases, illustrating the dynamic interactions within the system.

- Section 5 presents the structural models of the application, notably the class diagram, which defines the static structure and relationships among system entities.
- Section 6 introduces process modeling through Data Flow Diagrams (DFD) to show how data moves through the system at different abstraction levels.
- Section 7 details the Graphical User Interfaces (GUIs), explaining how users interact with the application and what functionalities are presented through the GUI forms.
- Section 8 concludes the document by summarizing lessons learned during development and proposing potential future improvements and extensions of the system.

The organization of this SRS aims to ensure clarity, traceability, and completeness, supporting a shared understanding among all parties involved in the development and evolution of MyLibrary.

2 Design and Implementation Constraints

2.1 Programming Language

All components of the application are developed using Java 17. This version was selected due to its support for modern language features and its status as a long-term support (LTS) release.

2.2 Build Tool

The project uses Apache Maven for dependency management, build automation, and packaging.

2.3 IDE

Development is primarily carried out using IntelliJ IDEA (either Ultimate or Community Edition). User interfaces (GUIs) are designed with IntelliJ's built-in GUI Designer.

2.4 Operating System

Windows 11 was used as the primary operating system during development.

2.5 Database

The application uses MySQL 8.0 for data storage. The version is inferred from the JDBC driver and is compatible with JDBC Connector 8.0.32.

For data insertion, querying, and validation, two tools were used:

- MySQL Workbench 8.0.42
- Intellij Database Option

To improve development speed and demo performance, Docker was utilized to containerize MySQL. It was used in combination with WSL 2.1.5, as required by Docker on Windows.

2.6 Persistence Layer

Java Database Connectivity (JDBC) is used for direct interaction with the relational database. All database connections and operations are handled manually via the java.sql package.

2.7 GUI Library

The graphical user interface is built using Java Swing, a standard Java library for developing cross-platform desktop applications.

2.8 Modeling Tools

All UML diagrams and DFDs were prepared using Visual Paradigm 17.1, enabling IEEE-compliant documentation and providing clear traceability for system design.

2.9 Reporting Tools

To enable real-time team collaboration and centralized documentation, Google Docs was used throughout the project.

2.10 Platform Constraint

The application is designed as a desktop-only solution, not web-based. It runs on client machines with at least:

- 2 GB RAM
- Dual-core processor
- Java 17 installed
- Stable internet or Docker MySQL network access for database interaction

2.11 Security Constraint

User passwords are securely stored using hashing algorithms (e.g., SHA-256 or bcrypt-like implementations). The Java Cryptography API (javax.crypto) is used for encryption. No plain-text password is stored in the database.

2.12 Architecture Constraint

The project follows Onion Architecture, ensuring a strict separation between domain, application, infrastructure, and presentation layers. All business logic resides in the domain layer. Given the relatively small domain, layer slicing was not implemented, but parallel development across layers was maintained.

2.13 Domain Driven Design (DDD)

DDD principles are applied extensively. All entities, value objects, aggregates, and domain exceptions are located in the domain layer.

Due to gaps in the initial requirement specification, DDD was implemented following a thorough analysis of the functional requirements to ensure a robust and logically consistent domain model.

2.14 Concurrency Constraint

The system is expected to support up to 100 concurrent users without noticeable degradation in performance.

2.15 Exception Handling Constraint

Exceptions originating from the domain or application layers are converted into user-friendly messages at the presentation layer. These messages are displayed in dialog boxes to ensure consistent and informative feedback.

2.16 Validation Rules

All user input is validated both in the presentation layer (GUI) and in the application/domain layers. This enforces business rules and ensures integrity even if GUI-level validation is bypassed.

3 Specific Requirements

No explanation is needed here. Only complete the subsections.

3.1 Functional Requirements

3.1.1. F1. Book Management (Admin Only)

- F1.1 The system shall allow an admin user to add a new book through a form.
- F1.2 The system shall check if the author (by name and surname) exists when adding a book; if absent, the system shall create a new author entry.
- F1.3 The system shall automatically increment and assign book id and author id for new entries.
- F1.4 The system shall perform the insertion of a book and a new author as a single action.
- F1.5 The system shall allow an admin user to delete a book by specifying its book Id.
- F1.6 The system shall delete the author if the deleted book was their only book.
- F1.7 The system shall cascade delete all associated user book states and comments when a book is deleted as it will not exist.
- F1.8 The system shall allow an admin user to edit and update all attributes of a book, including changing the associated author.
- F1.9 The system shall delete any author who becomes orphaned (i.e., has no books) as a result of editing or deleting a book.

3.1.2. F2. Author Management

- F2.1 The system shall allow users to search for authors by name (exact search).
    - As multiple authors can be found, in case of found multiple authors, a set of authors will be shown.
- F2.2 The system shall display all author fields, including the auto-generated website if website information is not known, when showing author information.

3.1.3. F3. UserBookState Management

- F3.1 The system shall allow users to mark their reading status for a book as read, unread, or wish to be read, independently from other users (This was a missing requirement, without it a type 2 user would not be able to generate data changes to be able to see lists of unread books, etc.).
- F3.2 The system shall allow users to set a release date only when marking a book as wish to be read; otherwise, release date shall be null.
- F3.3 The system shall restrict book ratings to 1–5 for books marked as read, and a rating of 0 (or more as project requirements document not constraints it) for unread or wish statuses.
- F3.4 The system shall allow users to add, update, or delete (multiline as project requirements not constraint it) comments for each user-book combination; comments may be or not.
- F3.7 The system shall cascade delete user book states and associated comments if a book or user is deleted for correct integrity.

3.1.4. F4. Favorites & Lists

- F4.1 The system shall display all books that a user has marked as read and rated either 4 or 5 as Favorite Books.
- F4.2 The system will display as Favorite Authors those authors who have at least three books in the user's library (since this is an unclear requirement, books marked as unread, that is, marked as read or wish to be read, will be accepted).
- F4.3 The system shall display all books for which the user's status is unread (or books without user's status as is the same as not have been read).
- F4.4 The system shall display all books on the user's wishlist with a set releaseDate.
- F4.5 The system shall display the cover image of a book by retrieving the cover path.

3.1.5. F5. Notifications

- F5.1 The system shall notify the users of type 1 (admin) at login of all books on their Wishlist with a read status of wish to be read with a release date within the next 7 days.

3.1.6. F6. Login

- F6.1 The system shall present a login where users must enter their username and password.
- F6.2 The system shall securely store user passwords using encryption or a secure hash function.
- F6.3 The system shall validate entered credentials against the encrypted/hashed password in the system.
- F6.4 The system shall determine the user type upon successful login and display the available functionalities for the type of user.

3.2 Performance Requirements

3.2.1 Maximum Concurrent Users Over Single Application Instance

The system shall support at maximum 100 concurrent users operating the application on different machines without any degradation in response time greater than 500 milliseconds for key operations such as login, book search, or comment retrieval.

3.2.2 Query Performance

The system shall respond to user-triggered queries such as searching books or authors and loading favorites within 1 second, even when the total number of books and authors stored exceeds 10,000 entries.

3.2.3 Application Startup Time

The application shall fully load and display the login application within 3 seconds on systems with at least 2GB RAM and a minimum of 2 cores processor, assuming a stable MySQL connection is available.

3.3 Software System Attributes

3.3.1 Reliability

The system shall ensure that user data is consistently stored and retrieved without loss or corruption, even during unexpected application shutdowns or system crashes. All critical operations such as book insertions, deletions, and updates are wrapped in transactions to ensure atomicity and rollback on failure.

3.3.2 Availability

The application shall be available to users at all times as long as their system is operational and a stable MySQL connection is active. MySQL-Workbench or Docker-WSL are configured to auto-start MySQL containers to minimize startup dependency issues and maximize uptime during demos or usage.

3.3.3 Security

The system shall store user credentials using secure hashing algorithms (e.g., SHA-256 via javax.crypto) to prevent unauthorized access. All access to restricted operations (e.g., book creation or deletion) is enforced at the application layer based on user type. Sensitive data is never stored in plain text.

3.3.4 Maintainability

The project follows the Onion Architecture and Domain-Driven Design (DDD) principles, separating domain logic from infrastructure, making the system easier to test, refactor, and extend. Code is modularized across well-defined layers, enabling teams to make changes in parallel without conflict.

3.3.5 Usability

The user interface is designed using Java Swing with tab-based navigation, combo boxes, dynamic forms, and immediate feedback through dialogs. Admin and regular user views are clearly separated, and validations provide clear messages to guide user input. UI components are labeled, responsive, and follow consistent interaction patterns.

3.4 Use Case Analysis

No explanation is needed here. Only complete the subsections.

3.4.1 Actors

3.4.1.1 Type-1 User

3.4.1.1.1 Actor Name

Type-1 user (admin).

3.4.1.1.2 Description

The Type-1 user is a privileged user with full access rights within the system. After a successful login, this user can utilize all system functionalities and perform all operations, including modifying data. This role typically represents a system administrator, librarian, or authorized staff member. The user has both read and write privileges over the system's data.

3.4.1.1.3 Responsibilities

1. Logs into the system using a valid username and password.
2. Adds new books to the system.
3. Updates or deletes existing book records.
4. Adds new authors or associates existing authors with books.
5. Views and edits book details.
6. Searches for authors by name and displays their information.
7. Lists all unread books in the user's library.
8. Views favorite books (read and rated 4 or 5).
9. Views favorite authors who have at least three books in the user's library (read or wish to be read).
10. Displays the cover image of a selected book.
11. Rates books on a scale from 1 to 5 based on personal preferences.
12. Set books as read, non read, or wish to be read with a release date.
13. Writes personal comments about the content or opinion of a book.

14. Receives notifications about wish-listed books that will be released within one week.

15. Performs select, update , and delete operations on the database.

3.4.1.2 Type-2 User

3.4.1.2.1 Actor Name

Type-2 user.

3.4.1.2.2 Description

The Type-2 user is a standard user with limited access rights. After logging into the system, this user can only view or query existing data but cannot perform any modifications. This role typically represents end users or regular library members who use the application to browse books and authors.

3.4.1.2.3 Responsibilities

1. Logs into the system using a valid username and password.

2. Views the details of books stored in the library.

3. Searches for authors by name and views their details.

4. Lists all unread books in the user's library.

5. Views favorite books that have been read and rated 4 or 5.

6. Views favorite authors who have at least three books in the user’s collection (read or wish to be read).

7. Displays the cover image of a selected book.

8. Executes read-only operations (SELECT queries) to retrieve data.

9. Cannot perform any insert, update, or delete operations on the database, except for user book status personalization.

10. Set books as read, non read, or wish to be read with a release date.

11. Rates books on a scale from 1 to 5 based on personal preference.

12. Writes personal comments about the content or opinion of a book.

13. The user can update the book status as read, not read, or wish to be read.

3.4.2 Scenarios

3.4.2.1 Display Book Information

After successfully logging into the system, the Type-1 user (admin) navigates to the “My Books” tab. In this section, a table displays the list of all books stored in the library database, including key fields such as book ID, title, author, publication year, page count, and cover path. The user scans through the table to identify the book whose details they want to view.

Once the desired book is identified, the user proceeds to the “Book Details” section below. In the “Book ID” input field, they enter the ID corresponding to the selected book and then click the “Display” button. This triggers a request to the system to fetch detailed information about that specific book from the database.

The system retrieves the book’s information based on the provided ID and automatically populates the corresponding fields. These fields include the title, author's name and surname, year of publication, number of pages, a brief description, and the cover image path. All these values are displayed in a read-only format for verification.

Additionally, if a valid cover path exists, the book’s image is rendered and shown in the interface. This allows the admin to visually confirm the accuracy of the data and, if necessary, proceed to update or manage the book’s record through the relevant controls available on the screen.

3.4.2.2 Edit Book Information By Book Id

After logging in, the admin user (Type-1) navigates to the “Manage Books” tab. In the "Book Operations" section, they see a field labeled "Book ID" and two buttons: "Delete Book" and "Display/Edit Book". The user enters the ID of the book they wish to update into the input field and clicks the "Display/Edit Book" button.

The system retrieves the book details from the database using the provided ID and automatically fills the fields in the "Create New Book" form above (such as Author Name, Title, Year, Pages, About, and Cover Path) with the existing information of the selected book.

The admin checks the book information and updates any fields if necessary. For example, they might fix a spelling error in the title, adjust the page count, or update the path of the cover image. This allows the user to correct or improve previously stored data before saving the changes.

Once the edits are complete, the admin clicks the "Create Book" button again, which functions as an update in this context. The system validates the inputs and then updates the corresponding book record in the database. A confirmation message is shown to indicate the successful update.

3.4.3 Use Case Forms

3.4.3.1 Complete Use Cases

3.4.3.1.1 Add a Book

Complete Use Case Form #1 Use Case Name	Add a Book
Participating Actors	User (Type-1)
Description	The user adds a new book to their library. If the author does not exist in the system, the user also creates a new author record.
Trigger	The user clicks the “Add Book” button after filling out the book form.
Preconditions	The user must be logged in as a Type-1 user. All required book fields must be filled
Normal Course (Flow Of Events)	1. User enters book details (title, year, pages, read, rating, etc.) via GUI. 2. User enters author name and surname. 3. System searches the authors table for a matching name-surname. 4. If found: a. System retrieves authorId. 5. Else: a. System creates new author with a new authorId. 6. System creates a new book record using a new bookId and associates it with the determined authorId. 7. The system displays a successful operation message.

Post conditions	1. A new record is inserted into the books table. 2. If needed, a new record is also inserted into the authors table.
Exceptions	1. Database connection failure → error message shown. 2. Missing required fields → input validation error. 3. Duplicate book title for same author → optional check and warning.

3.4.3.1.2 Login

Complete Use Case Form #2 Use Case Name	Login
Participating Actors	User
Description	User logs in to access application features based on their user type.
Trigger	The user clicks the “Login” button after entering username and password.
Preconditions	1. The database must contain the user credentials. 2. The user must not already be logged in.
Normal Course (Flow Of Events)	1. User enters username and password.

	2. System queries the userInfo table for a match. 3. If match is found: a) System checks userType. b) If userType =1, all tabs shown. c) If userType = 2, all tabs except manage books and wishlist notification ara not shown. 4. loginFrame is closed.
Post conditions	User enters the main screen with access according to user type.
Exceptions	1. Wrong username/password → error dialog shown. 2. Database unavailable → error message shown. 3. Empty fields → validation warning.

3.4.3.1.3 Display Favorite Authors

Complete Use Case Form #3 Use Case Name	Display Favorite Authors
Participating Actors	User (Type-1 or Type-2)
Description	The system shows authors for whom the user has read or setted as wishing to read at least 3 of their books.
Trigger	The user clicks the “Show Favorite Authors” button.
Preconditions	1. The user must be logged in.

	2. The user must have rated books stored in the database.
Normal Course (Flow Of Events)	1. User initiates the request via GUI. 2. System retrieves all books of the current user. 3. System groups books by authorId and counts books with read=1 or read=3. 4. System selects authors who have at least 3 such books. 5. System fetches those authors from authors table. 6. Author list is displayed in the GUI (e.g., in JTable).
Post conditions	A list of qualifying authors is shown to the user.
Exceptions	1. No qualifying authors → message "No favorite authors found." 2. SQL error → error message shown. 3. User has no books → fallback message.

3.4.3.2 Basic Use Cases

3.4.3.2.1 Delete a Book

Basic Use Case Form #1 Use case name	Delete a Book
Participating actors	User (Type-1)
Description	The user deletes a book from their library. If the author has no other books left, the author is also removed.
Trigger	User enters bookId and clicks "Delete Book".

Preconditions	1. User must be logged in as Type-1. 2. bookId must be valid and exist in the database.

3.4.3.2.2 Search Author By Name

Basic Use Case Form #3 Use case name	Search Author by Name
Participating actors	User (Type-1 or Type-2)
Description	Searches for an author in the database using name.
Trigger	User enters author name, then clicks “Search”.
Preconditions	1. User must be logged in. 2. Author name must be provided.

3.4.3.2.3 Display Favorite Books

Basic Use Case Form #4 Use case name	Display Favorite Books
Participating actors	User (Type-1 or Type-2)
Description	Shows a list of books rated 4 or 5 and marked as read by the user.
Trigger	User clicks “Show Favorite Books”.
Preconditions	User must be logged in. Some books must have read=1 and rating ≥ 4.

3.4.3.2.4 Display Unread Books

Basic Use Case Form #5	Display Unread Books

Use case name	
Participating actors	User (Type-1 or Type-2)
Description	Lists all books marked as not read (read=2) or not having data related to the user as is equal to not read.
Trigger	User clicks “Show Unread Books”.
Preconditions	User must be logged in. There must be books with read=2.

3.4.3.2.5 Notify Upcoming Releases

Basic Use Case Form #6 Use case name	Notify Upcoming Releases
Participating actors	User (Type-1)
Description	On application startup, the user is notified about books on their wish list (read=3) that will be released within a week.
Trigger	Application starts and user logs in.
Preconditions	User must be logged in. There must be wish-list books with upcoming release dates.

3.4.4 Relationships among Actors and Use Cases

Explain the relationship between each actor and the use case in your use case diagram.

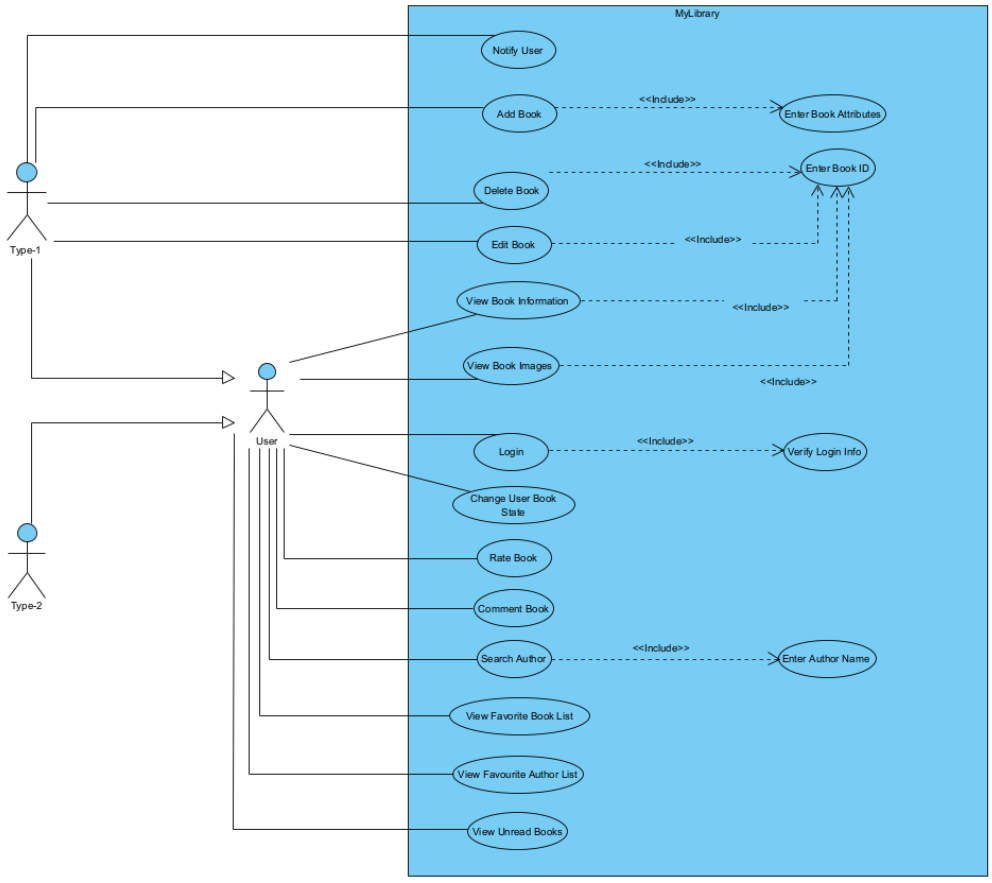
Type-1 User:

- Notify User: System notifies user on wishisted books.
- Add Book: Admin is able to add a book to the system. Requires book attributes to be entered. Shown by includes relation.
- Delete Book: Admin deletes a book. Book is removed from the system.
- Edit Book: Admin changes attributes of a book. Changed attributes reflect the system.

Type-2 User(Includes Type-1 User):

- Login: Enters account information. Login information needs to be verified by the system. Shown by includes relation.
- View Book Information: Book details are shown to the user by the system. Requires a specific book ID to be entered. Shown by includes relation.
- View Book Images: Related book cover is shown to the user. Requires a specific book ID to be entered. Shown by includes relation.
- Change User Book State: Changes book state into read, not read or wishes to read.
- Rate Book: Rates book out of 5 stars.
- Comment Book: Leave a comment to the book.
- Search Author: Searches for an author. Requires author name to be entered. Shown by includes relation.
- View Favourite Book List: User is shown a list of books that he/she added to favourites.
- View Favourite Author List: User is shown a list of authors that he/she added to favourites.
- View Unread Books: User is shown a list of books that he/she hasn't read.

3.4.5 Use Case Diagram



This Use case diagram displays a high level representation of MyLibrary application. There are two different types of users: Type-1 represents Admin user and Type-2 represents regular user. Both users interact with the system. Admin users are able to interact with the system using a wider range of

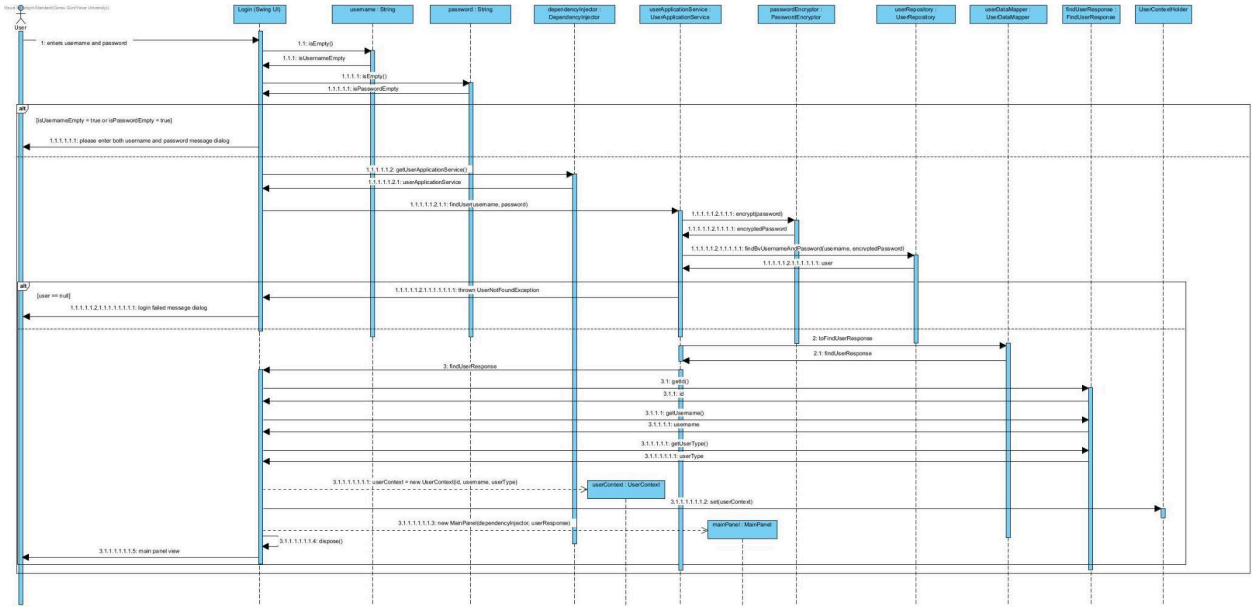
features. Admin users differ from regular users by being able to add, editing and updating books. They are also able to get notified by the system about the wishlisted books. Regular users are not able to add, edit, update books and they do not get notified. Both users have to login. They do not have to explicitly indicate their account type. System automatically recognises the account type based on login information. Both users can view book information and view book image. These interactions cannot directly occur. They require users to enter a book ID which is shown in the diagram with the include relation. Both users can change book state as read, not read and wish to read. They can rate the book out of 5 and leave comments about the book. They can search for an author that requires the user to enter the author name. Both users can view their favourite book list, favourite author list and unread books list.

4 Behavioral Models

No explanation is needed here. Only complete the subsection.

4.1 Sequence Diagram

4.1.1 Login



You can clearly see the interaction between the actor and the application in the image.

The presentation layer acts as the controller. Login, which is the GUI, is located in this layer. It obtains the username and password from the actor. It validates if they are empty. If they are, a dialog box will be sent with a message asking to enter both values correctly. If not, the operation will continue.

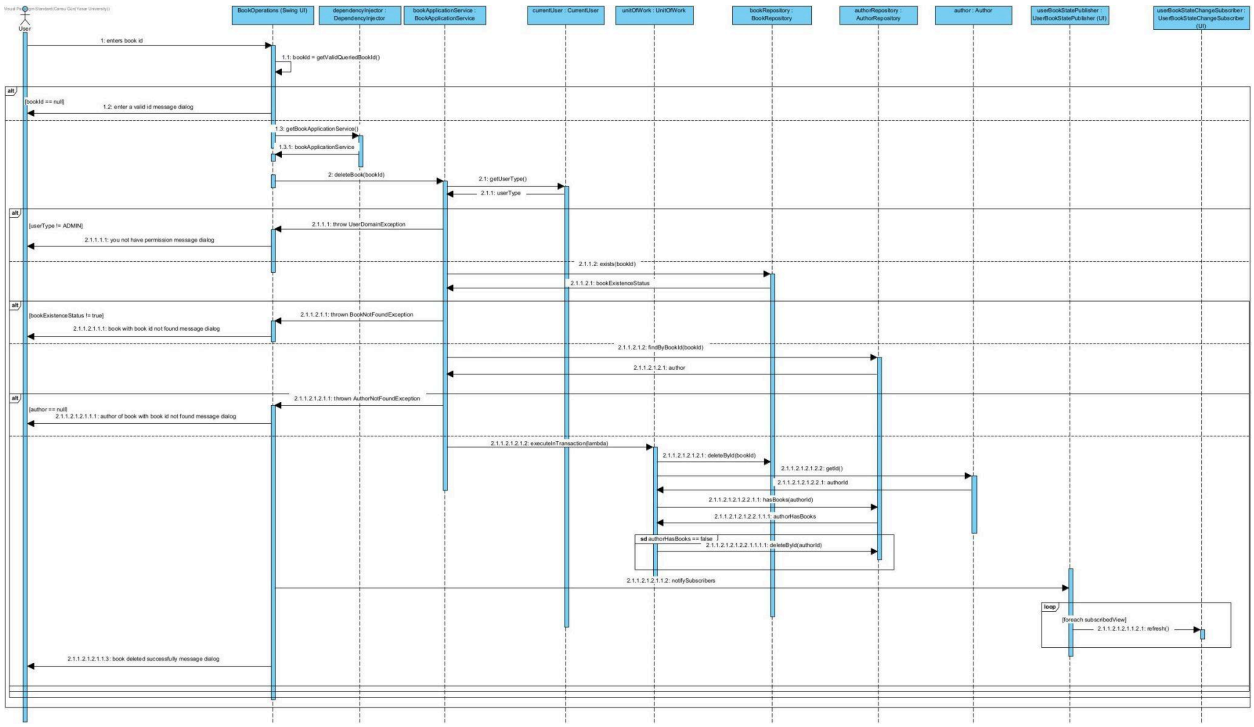
The presentation layer uses the dependencyInjector, which contains all the application layer services ready (initialized and with their dependencies). The GUI obtains the userApplicationService from the dependencyInjector and uses the findUser method with the username and password inputs.

This application service uses a passwordEncryptor to encrypt the user's password, since passwords are stored encrypted. The user is found via the userRepository within the userApplicationService. If a null user is returned, the user will be notified via a dialog box that a user with the entered credentials does not exist.

The GUI will extract the id, userType, and username from the returned data and create a userContext to place within the userContextHolder (thus maintaining the current user information, saving us from having to keep scattered or incomplete information and thus avoiding unnecessary queries throughout the application).

The mainPanel will be created with the data obtained from the application layer (userApplicationService) and the dependencyInjector. It will be disposed to the current login window. The client will see the main panel.

4.1.2 Delete A Book Use Case Diagram



You can clearly see the interaction between the actor and the application in the image.

The presentation layer acts as the controller. Within this layer is BookOperations, which is the GUI. It obtains the book ID from the actor. It validates whether it is a correct number, clearing it, and attempting to convert the input to a number. If not, it will remain null. Then, based on a condition, if it is null, a dialog box with an incorrect ID message will be sent to the user. If not, the operation will continue.

The presentation layer uses the dependencyInjector, which contains all the application layer services ready (initialized and with their dependencies). The GUI obtains the bookApplicationService from the dependencyInjector and uses the deleteBook method with the input already converted to a number.

This application service obtains the base user type as currentUser, which, behind the scenes in the infrastructure, uses the contextHolder, which is already initialized thanks to the user's login action. A check is made to see if the user type is admin or not. Based on this, the action will proceed or a dialog box will be sent to the user warning them that they do not have permission for the action.

If the action continues, it will be checked if the workbook exists; if not, the user will be notified of the situation via a dialog box.

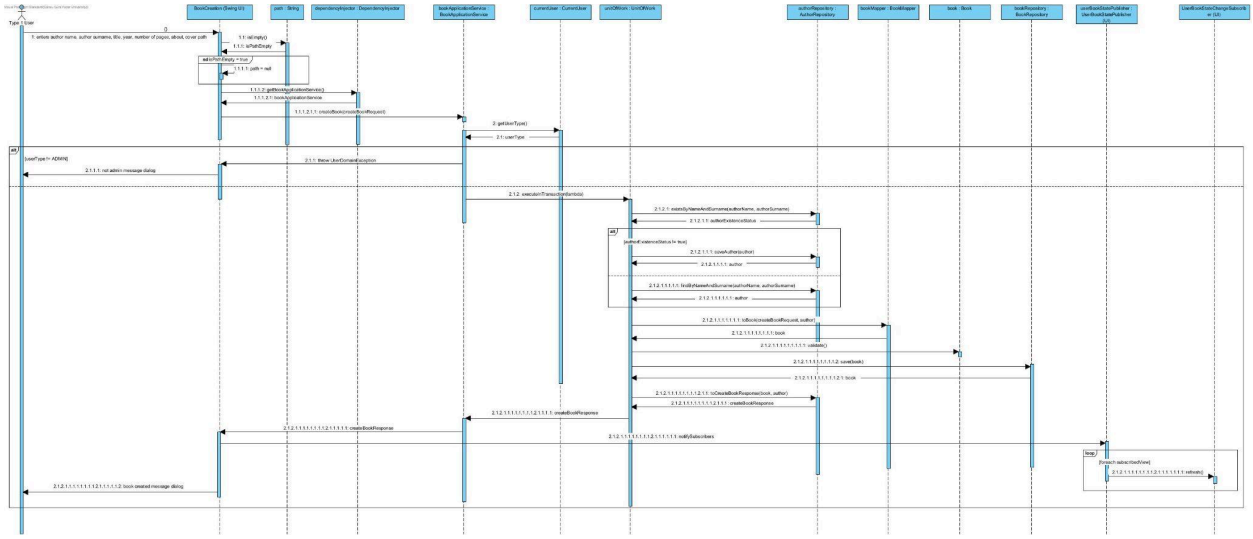
Before proceeding to delete the workbook, the author of the workbook to be deleted will be obtained. Since the author could have been deleted at that moment, it will be checked whether the author obtained is null or not. Based on this, the action will proceed or the user will be notified of the situation via a dialog box.

Continuing with the action, the unitOfWork will be used, which helps us execute atomic transactions. Within it, the workbook will be deleted. After that, the transaction will check to see if the author has any books linked to it. If not, the author will also be deleted within the same transaction.

The application layer service doesn't return any value because deletion shouldn't delete anything, and it propagates exceptions in the event of a failure. The current GUI will notify the subscribed GUIs via the userBookStatePublisher. This is clearly the Observer Design Pattern. Screens that need to refresh their data, such as unreadBooks, will refresh their data.

Finally, the user will be notified that the book has been deleted via a dialog box.

4.1.3 Insert A Book Use Case Diagram



You can clearly see the interaction between the actor and the application in the image.

The presentation layer acts as the controller. BookCreation, which is the GUI, is located in this layer. It obtains the author name, author surname, title, year, number of pages, about, and cover path from the actor. Check if the path is empty, so that instead of using an empty String, you can use null, since the bookApplicationService service located in the application layer requires it.

The dependencyInjector will be used to obtain the bookApplicationService, since not only it but all the application layer services are initialized and ready to be used.

The createBook method will be called with all the information obtained from the user converted into a CreateBookRequest (via the builder pattern).

The BookApplicationService already knows which user the operation is targeting thanks to the currentUser (since behind the scenes it uses the contextHolder of the infrastructure/security layer). It will check the current user type; if it is not type 1 (admin), it will reject the action, and at the end, a dialog box will be displayed to the user stating that they do not have permission for the action.

If the operation continues, the bookApplicationService will use the unitOfWork to execute an atomic transaction. Within the transaction, it will check if an author exists with the first and last name obtained from the actor. If none exists, an author will be created with the authorRepository; if not, the existing author will be fetched from that repository. The request and author data will be converted to a book entity, and that identity will be validated, which contains the domain-specific validations. If a validation error occurs, an error will be propagated to the actor, thanks to the GUI, which will display the error with a dialog box. If there are no problems with the validation, the book will be saved, the saved book will be obtained, converted to a response type expected by the presentation layer using the bookDataMapper, and the data will be returned. After passing the data through the layers, the presentation layer will obtain it, and before notifying the user of the creation of the base book in a dialog box, the userBookStatePublisher will be notified to the subscribed UIs of the newly generated data. The subscribed UIs will refresh, thus displaying updated data.

Finally, BookOperations will notify the actor with a dialog box that the operation was successfully applied.

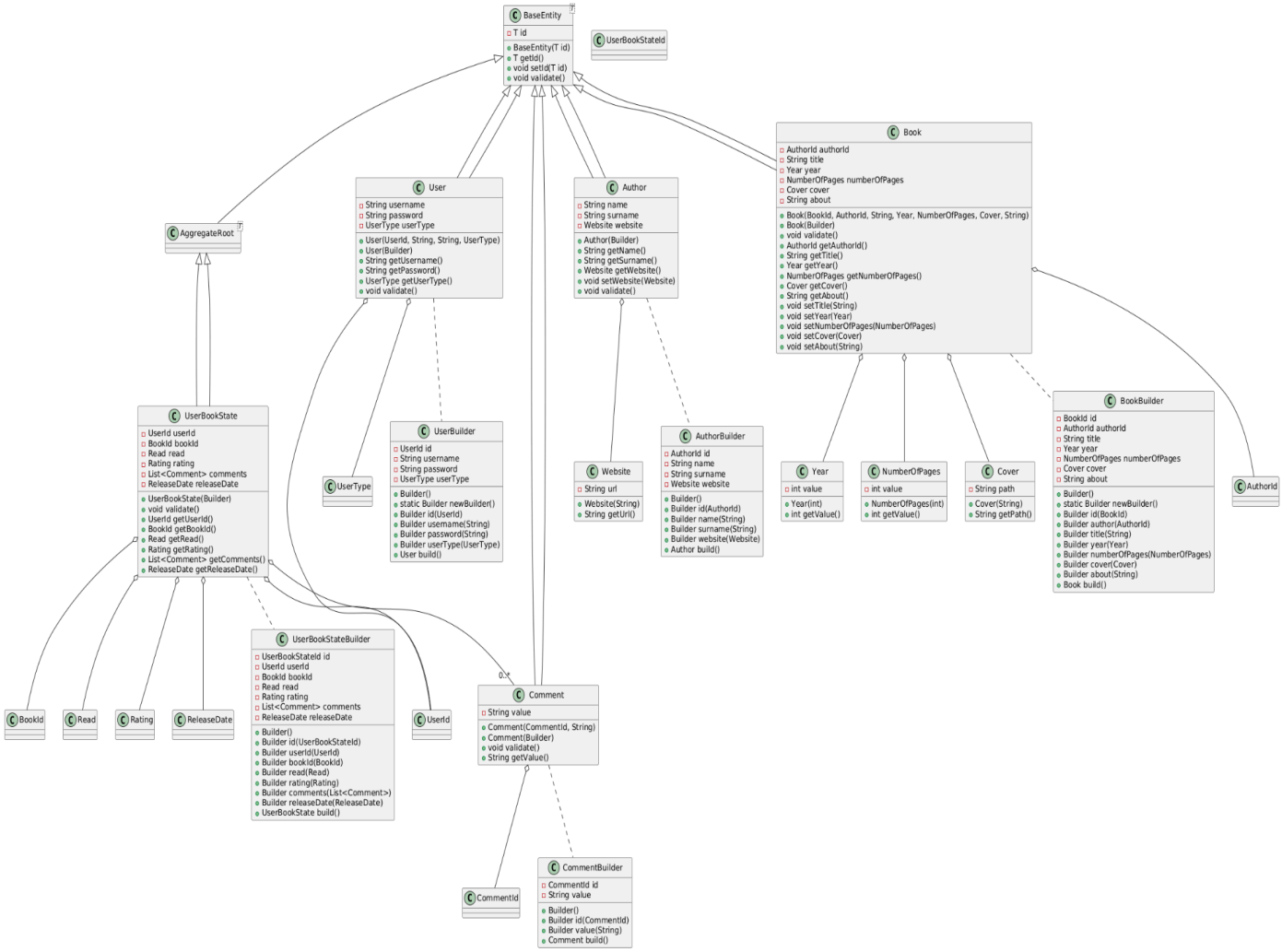
5 Structural Models

No explanation is needed here. Only complete the subsection.

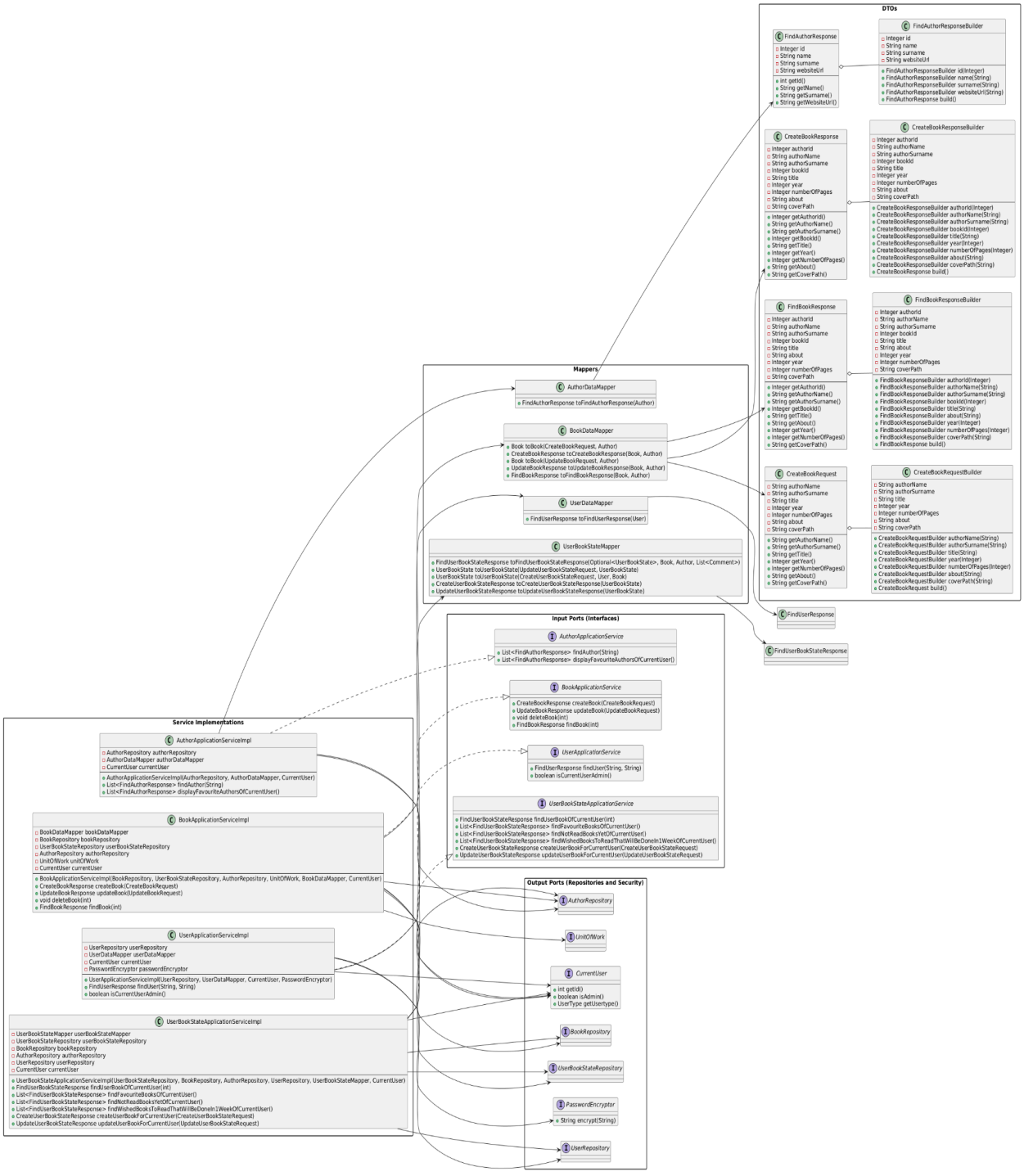
5.1 Class Diagram

As the count of classes is too big (93), we will show each class diagram of each layer of the project.

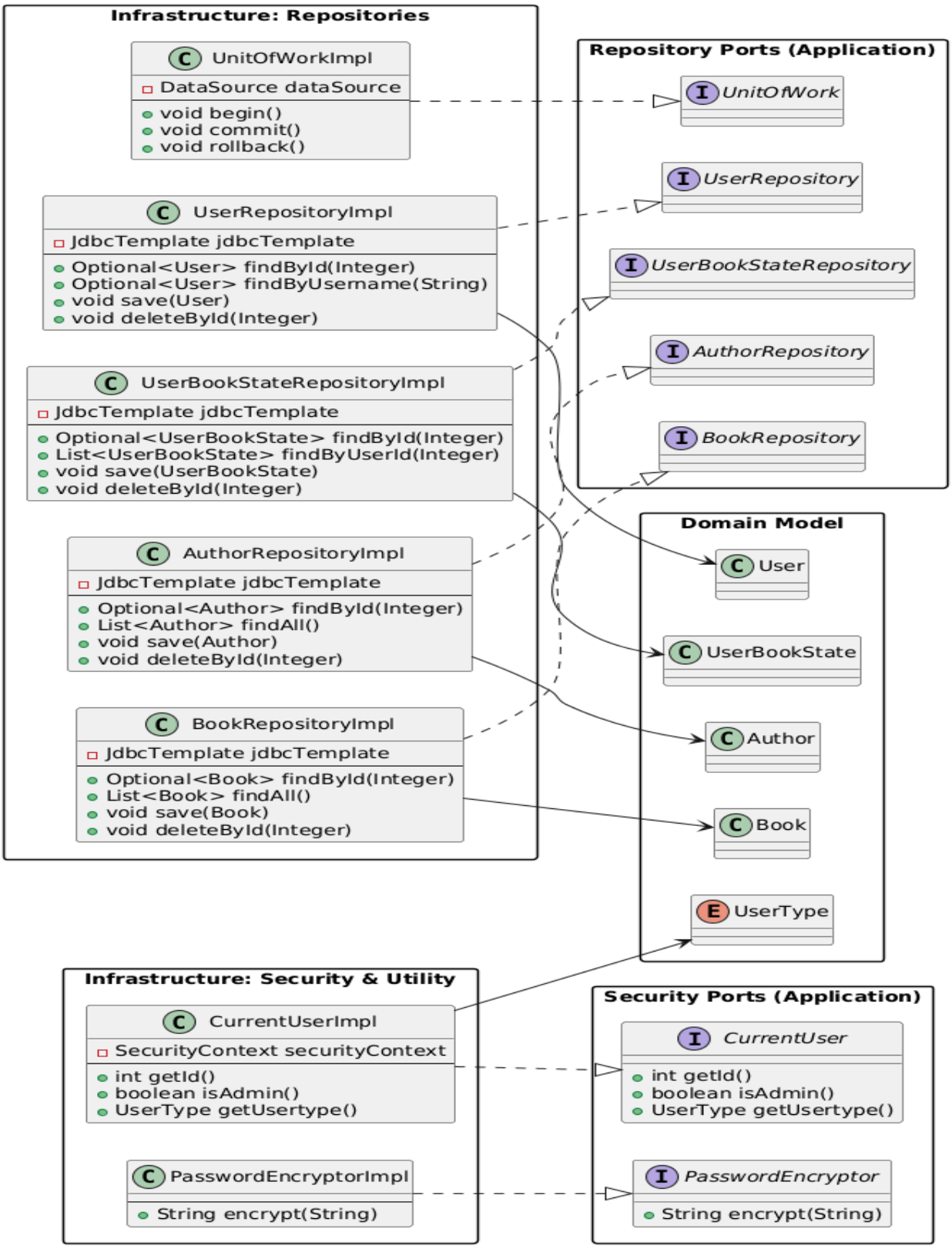
5.1.1 Domain Layer



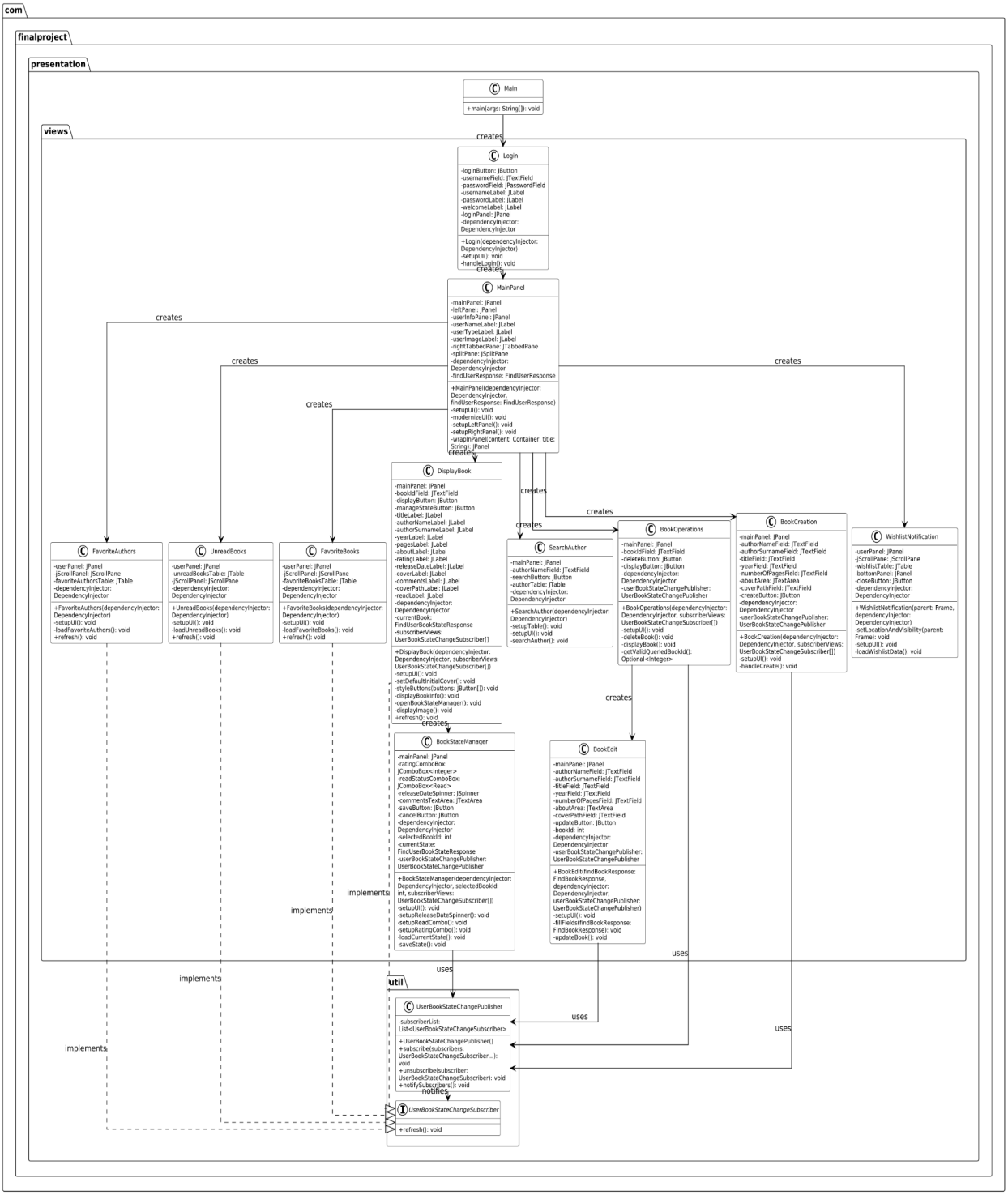
5.1.2 Application Layer



5.1.3 Infrastructure Layer



5.1.4 Presentation Layer

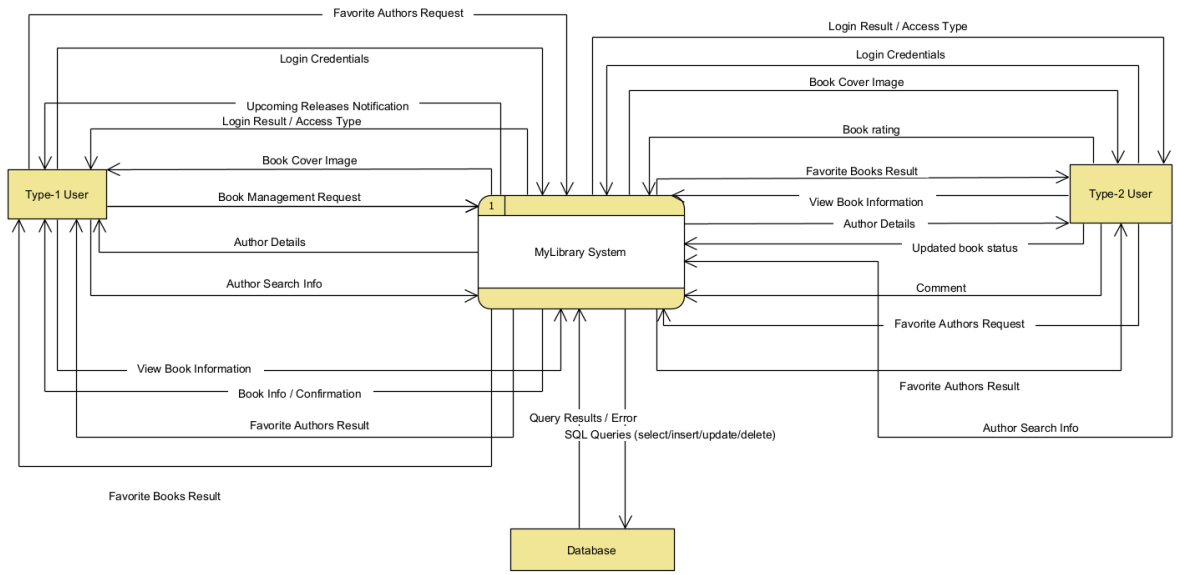


6 Process Modeling

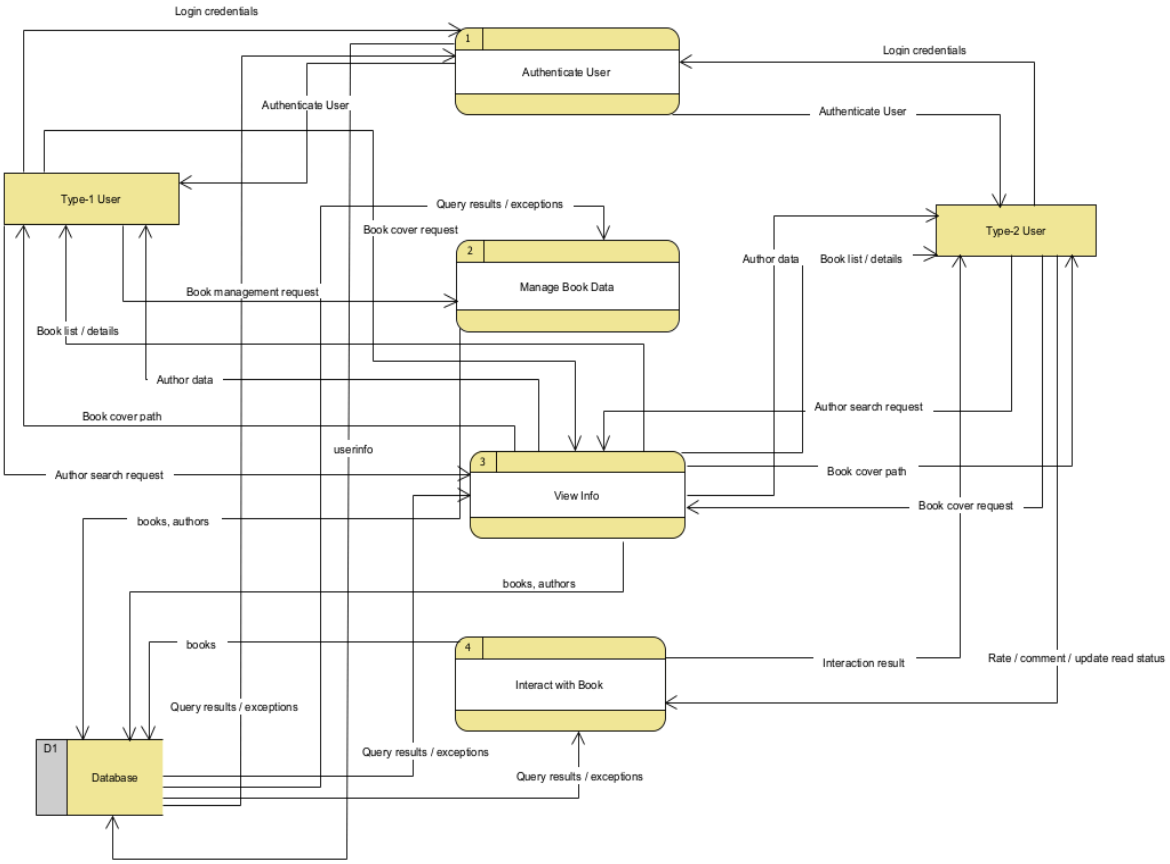
No explanation is needed here. Only complete the subsection.

6.1 Data Flow Diagram (DFD)

6.1.1 DFD Context Diagram

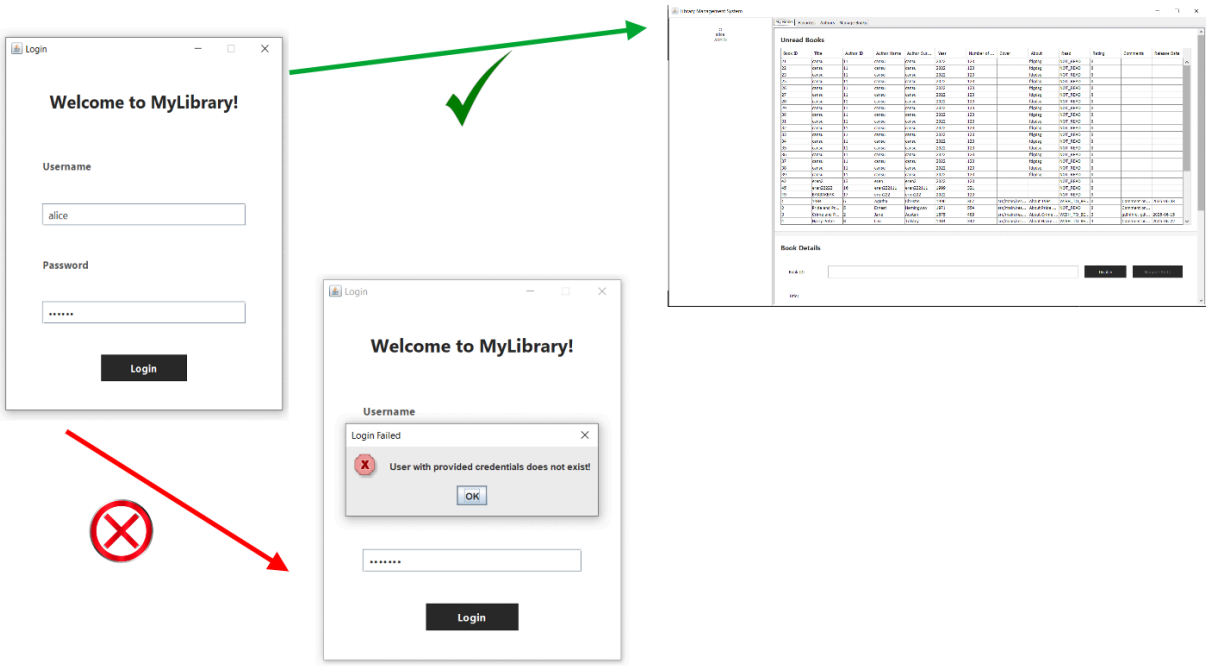


6.1.2 DFD Level 0



7 Graphical User Interface(s) (GUIs)

7.1 Login (Login.form)



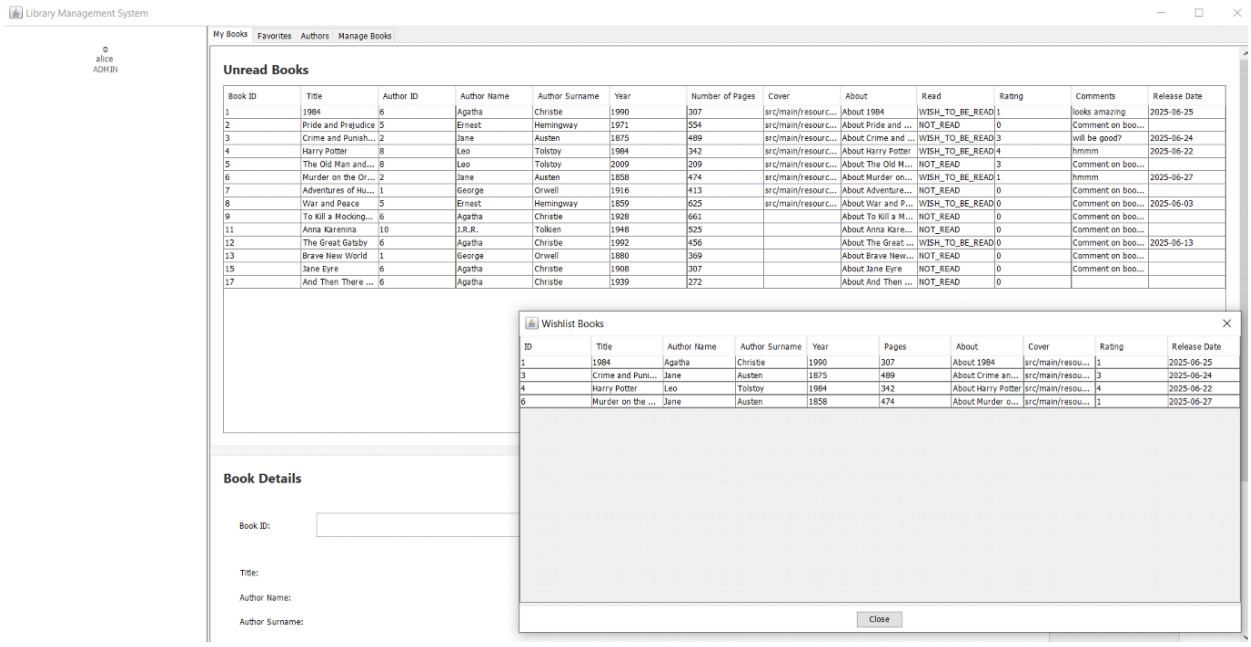
On the left, you can see the application's main window, the login window. The user's username and password will be requested here.

When you log in, or press enter while in the password text field, the application layer will be used to obtain the information of the logged-in user. The application layer hashes the entered password with the help of the infrastructure security layer. If the user does not exist, a domain user not exist exception will be thrown, which will be displayed in the presentation layer with a dialog box as shown in the image below. If the user exists, the current window will close and the main panel window will open.

7.2 MainPanel (MainPanel.form)

The application's core window will appear here, displaying all the available features for the current user (based on whether they are type 1 (admin) or type 2 (normal)) based on the top tabs. The window is multi-tab. The current user's information is displayed on the left.

7.2.1 For Admin (Type-1 User) Display



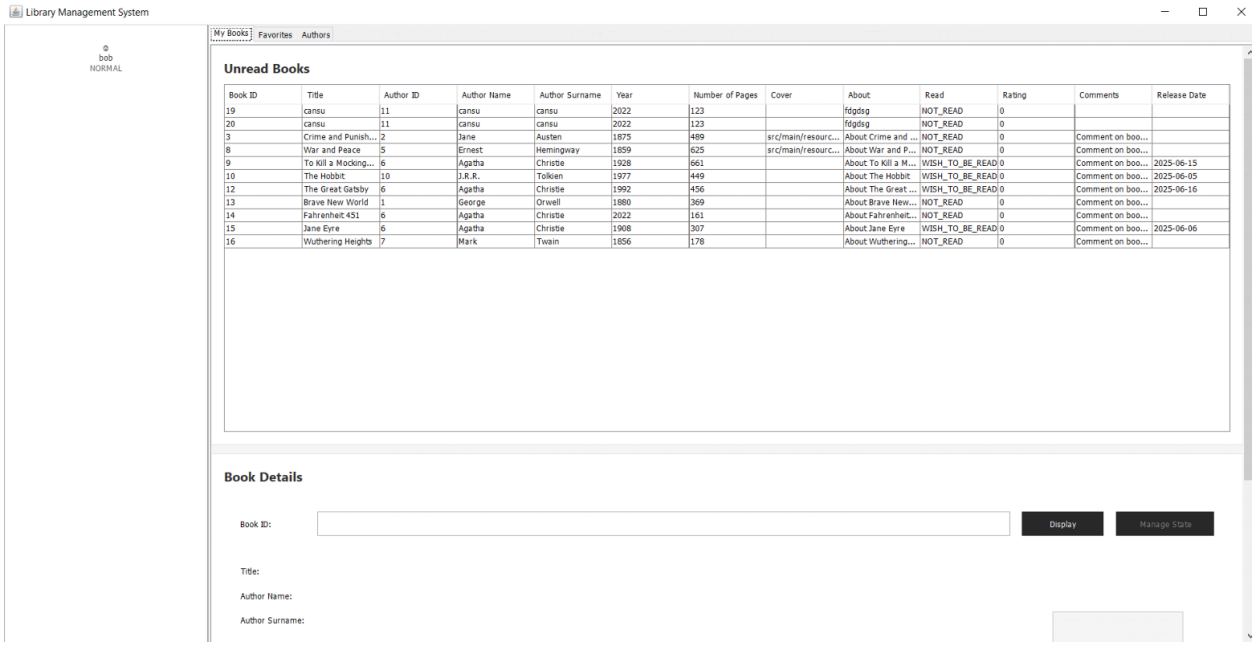
A type 1 user (admin) will have all the features shown in the image. They will have the tabs My Books, Favorites, Authors, and Manage Books;

- The My Books tab will encapsulate the features for displaying unread books, displaying book information (cover image, book and user book state values), and managing user book state (update/insert).
- The Favorites tab will encapsulate the functionality for favorite books and favorite authors.
- The Authors tab will encapsulate the functionality for searching for authors by name.
- The Manage Book tab will encapsulate the functionality for inserting, updating, and deleting books.

Also note that at the beginning of the main panel, if the current user is type 1 (admin), a window will also be displayed with the wishlist notification information.

All of these GUIs and features will be discussed in more detail in later sections.

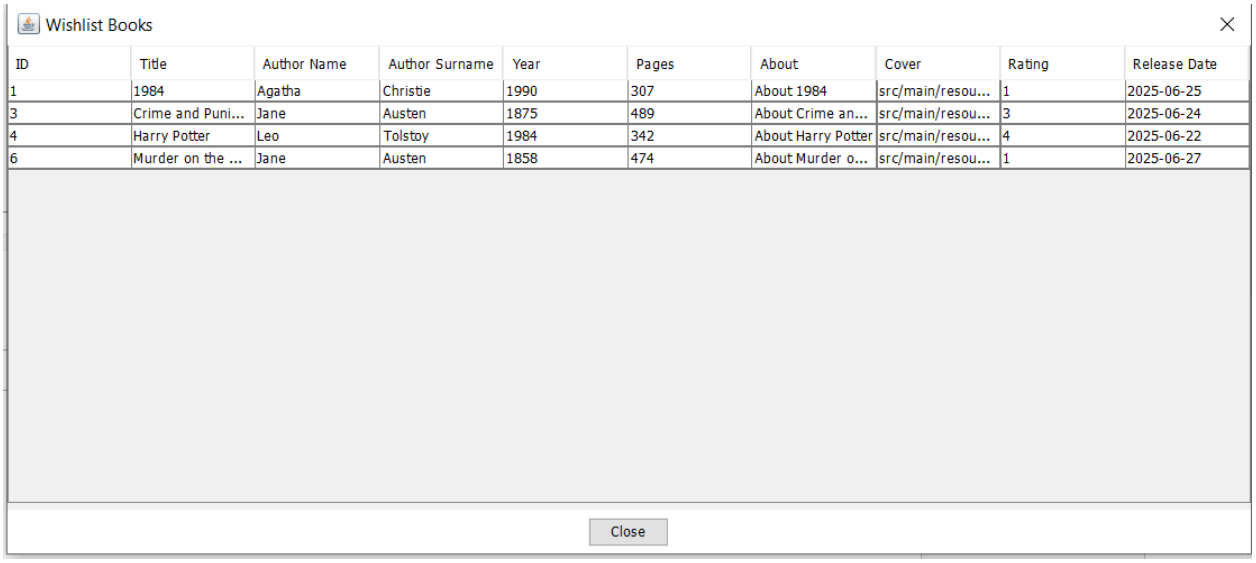
7.2.2 For Normal User (Type-2 User) Display



Here, it will be almost the same as for type 1 users (admin). The only difference is that the wishlist notification and the manage books tab will not be displayed.

It's worth noting that we don't just hide functionality with the GUI; thanks to the modern approaches and architecture we selected, we also validate any user action based on its type in each service at the application layer. That is, if a programmatic action related to manage books were to be performed, and the current user is not an admin (this is differentiated by the methods implemented in the infrastructure security layer, where we have a user context holder and rely on the current information used), then domain exceptions will be thrown, such as an invalid user for that action.

7.3 Wishlist Notification (WishlistNotification.form)



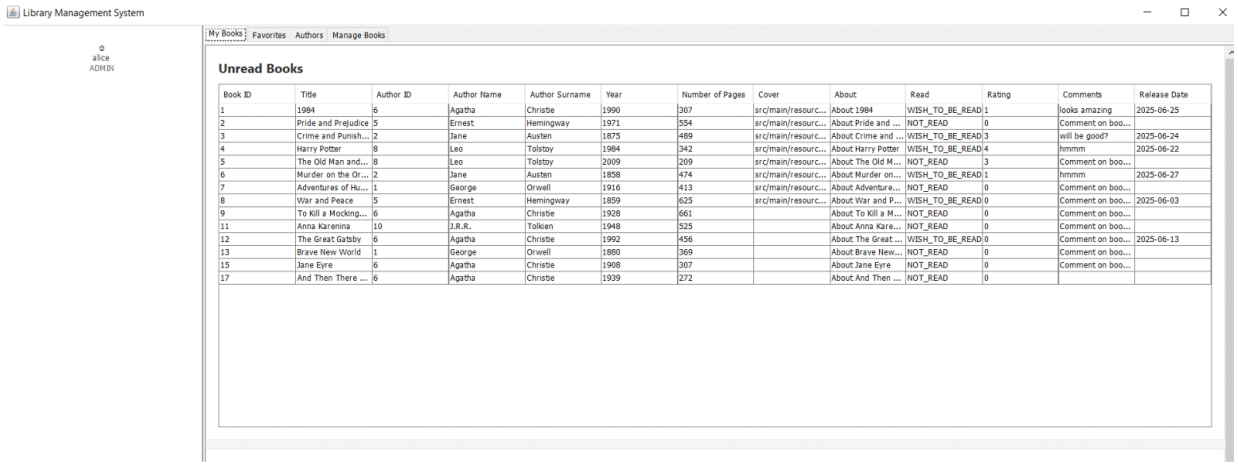
This feature is only available in both the GUI and the application layer (as it validates the current user's type) for type 1 (admin) users. The GUI will not display this feature because if the current user is not type 1, the window will not be displayed after login.

The view will only be displayed in the lower left corner of the main panel after logging in for type 1 users (admin). Books with a wish to be read status and a release date between the current date and 7 days after the current date will be displayed here.

The following information about the books and user book status will be displayed:

- ID
- Title
- Author name
- Author surname
- Year
- Number of pages
- About
- Cover
- Rating
- Release date

7.4 Unread Books (UnreadBooks.form)

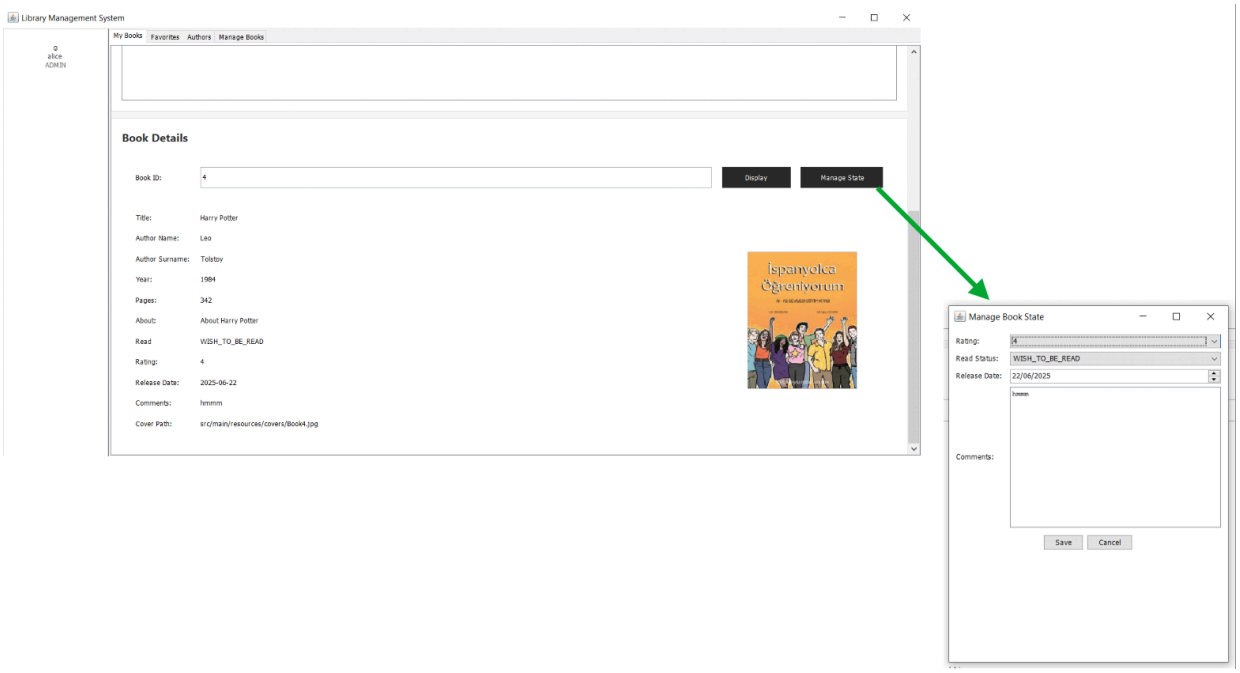


Book ID	Title	Author ID	Author Name	Author Surname	Year	Number of Pages	Cover	About	Read	Rating	Comments	Release Date
1	1984	6	Agatha	Christie	1990	307	src/main/resourc...	About 1984	WISH_TO_BE_READ	1	looks amazing	2025-06-25
2	Pride and Prejudice	5	Ernest	Hemingway	1971	554	src/main/resourc...	About Pride and ...	NOT_READ	0	Comment on boo...	
3	Crime and Punish...	2	Jane	Austen	1875	489	src/main/resourc...	About Crime and ...	WISH_TO_BE_READ	3	will be good?	2025-06-24
4	Harry Potter	8	Leo	Tolstoy	1984	342	src/main/resourc...	About Harry Potter	WISH_TO_BE_READ	4	hmmm	2025-06-22
5	The Old Man and...	8	Leo	Tolstoy	2009	209	src/main/resourc...	About The Old M...	NOT_READ	3	Comment on boo...	
6	Murder on the Or...	2	Jane	Austen	1858	474	src/main/resourc...	About Murder on...	WISH_TO_BE_READ	1	hmmm	2025-06-27
7	Adventures of Hu...	1	George	Orwell	1916	413	src/main/resourc...	About Adventure...	NOT_READ	0	Comment on boo...	
8	War and Peace	5	Ernest	Hemingway	1859	625	src/main/resourc...	About War and P...	WISH_TO_BE_READ	0	Comment on boo...	2025-06-03
9	To Kill a Mocking...	6	Agatha	Christie	1928	661		About To Kill a M...	NOT_READ	0	Comment on boo...	
11	Anna Karenina	10	J.R.R.	Tolkien	1948	525		About Anna Kare...	NOT_READ	0	Comment on boo...	
12	The Great Gatsby	6	Agatha	Christie	1992	456		About The Great ...	WISH_TO_BE_READ	0	Comment on boo...	2025-06-13
13	Brave New World	1	George	Orwell	1880	369		About Brave New...	NOT_READ	0	Comment on boo...	
15	Jane Eyre	6	Agatha	Christie	1908	307		About Jane Eyre	NOT_READ	0	Comment on boo...	
17	And Then There ...	6	Agatha	Christie	1939	272		About And Then ...	NOT_READ	0		

Here, information about the user's unread books will be displayed. There's a neat trick here: since inserting unread user book state records for all users isn't very accurate with each book insert, instead of that, to efficiently query, the application layer, along with the infrastructure persistence layer, collects the unread user book state records and directly collects the books that don't have user book state records for the user. The default values for read status, rating, comments, and release date are not read, 0, empty, and empty.

Information about the books and user book state are displayed: book ID, title, author ID, author name, author surname, year, number of pages, cover, about, read, rating, comments, and release date. Since comments can have more than one value, comments are displayed as a single value by joining the values with a comma.

7.5 Book Details (DisplayBook.form)



In the image, you can see an example of viewing the details of a book. Note that here we actually show not only the values corresponding to the user book state, but also values for the book and the book's author. The following values are displayed:

- Title
- Author name
- Author surname
- Year
- Number of pages
- About
- Read status
- Rating
- Release date
- Comments
- Cover path

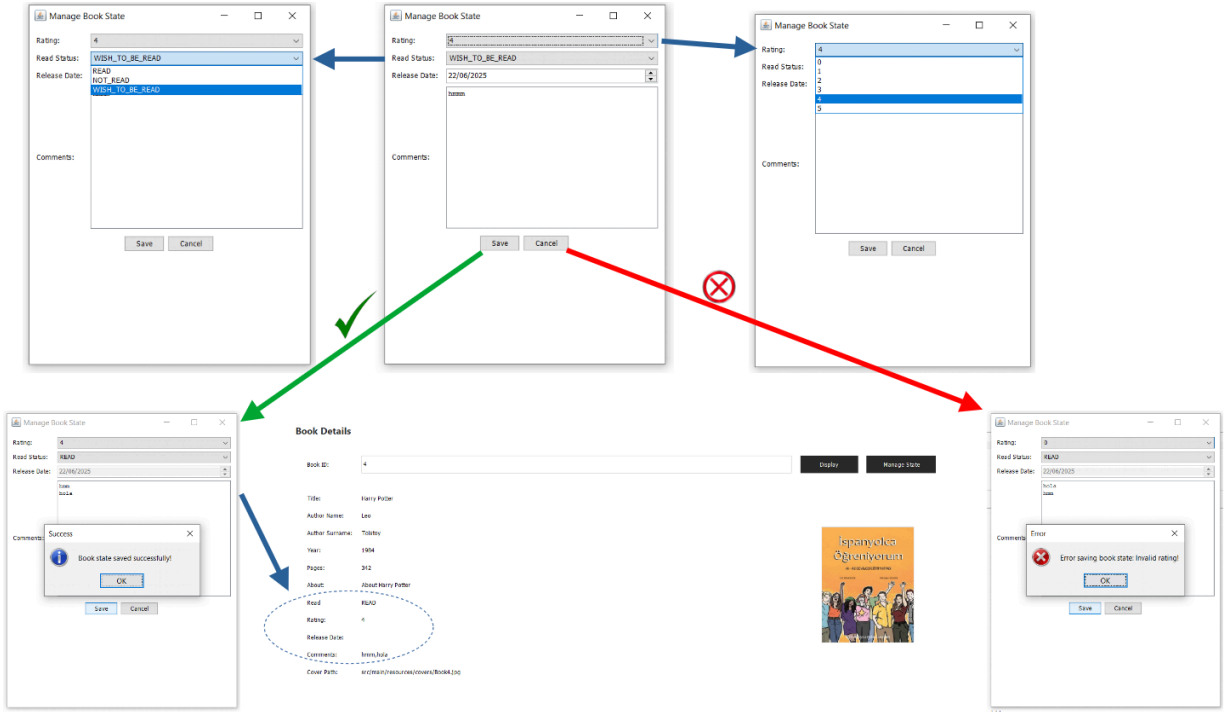
On the right, the image is shown (if it can be displayed, that is, if the file path exists, the photo will be shown; if not, nothing).

Note that there may be no user information for the corresponding book; therefore, the application layer will send default data, which would be not read, 0, nothing, nothing for read status, rating, release date, and comments.

If there are comments, they will all be displayed (as there may be many) as a single value joined with a comma.

On the right you can see that clicking the Manage Book State button will open the Manage Book State window.

7.6 Manage User Book State (BookStateManager.form)



Look at the top-center of the image. This is the main window. Here we select the rating and reading status values as a combo box. The release date value can be selected either via the keyboard or via the spinner.

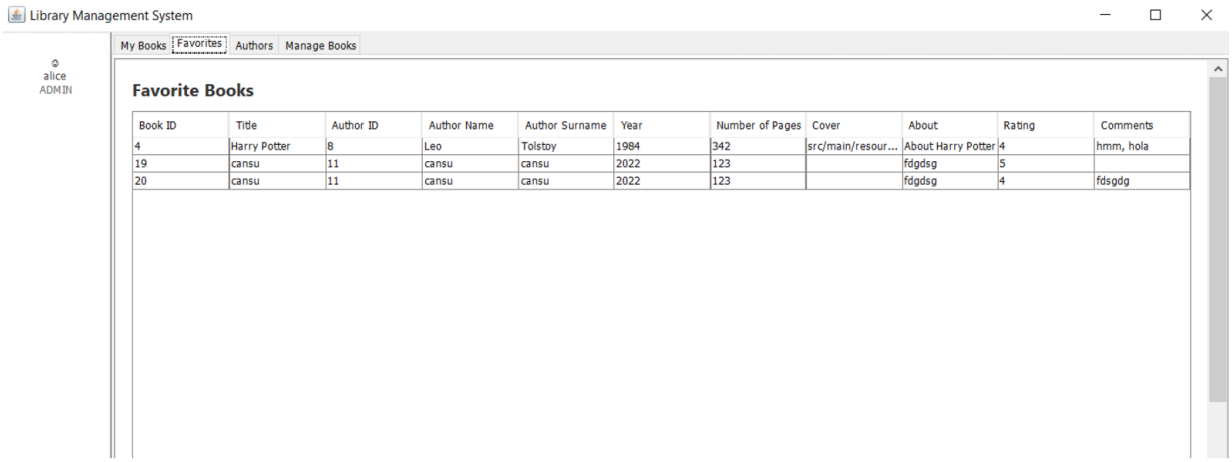
The top left of the image shows how to select the read status. The top right of the image shows how to select the rating. Remember that the rating can be 0, but not in all cases. This is why the selection is left in the combo box. At the end of the action, it will be validated to see if the base value for the selected read status can be selected. If the read status is selected as wish to be read, the date will be selected by the GUI. If another status is selected, the release date will be disabled so it cannot be set. Regardless, whatever happens, the application layer validates all selected values as business rules.

In the lower-left part of the image, you can see an example of correct value insertion. The presentation layer always keeps the data up to date. If it notices that there is no existing user data selection, it inserts the user book state with the application layer; if not, it updates the user book state with the application layer. At the end of the example in the lower-left part of the image, you can see how the new information is propagated to the book details view of the current user book state thanks to the observer pattern.

The lower-right part of the image shows how it is invalid to enter the pair of 0, read as rating, and read status, since the project requirements do not accept this pair of values. The reflected exception comes

from the application layer, which throws a domain exception, and the presentation layer displays the exception message with a dialog box.

7.7 Favorite Books (FavoriteBooks.form)



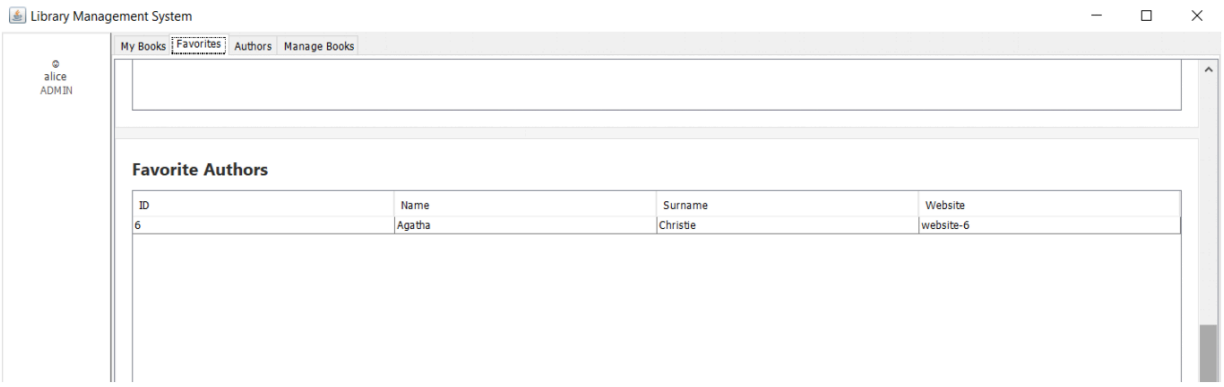
In the view, we display books that have a read status and a rating of 4 or 5.

Note that we don't display the read status column in the information. Since the request only requires that books have a read status, it's irrelevant to display that column.

The book information displayed includes book ID, title, author ID, author name, author surname, year, number of pages, cover path, about, rating, and comments.

Remember that comments can be more than one, so we join them with a comma to display them as a single value.

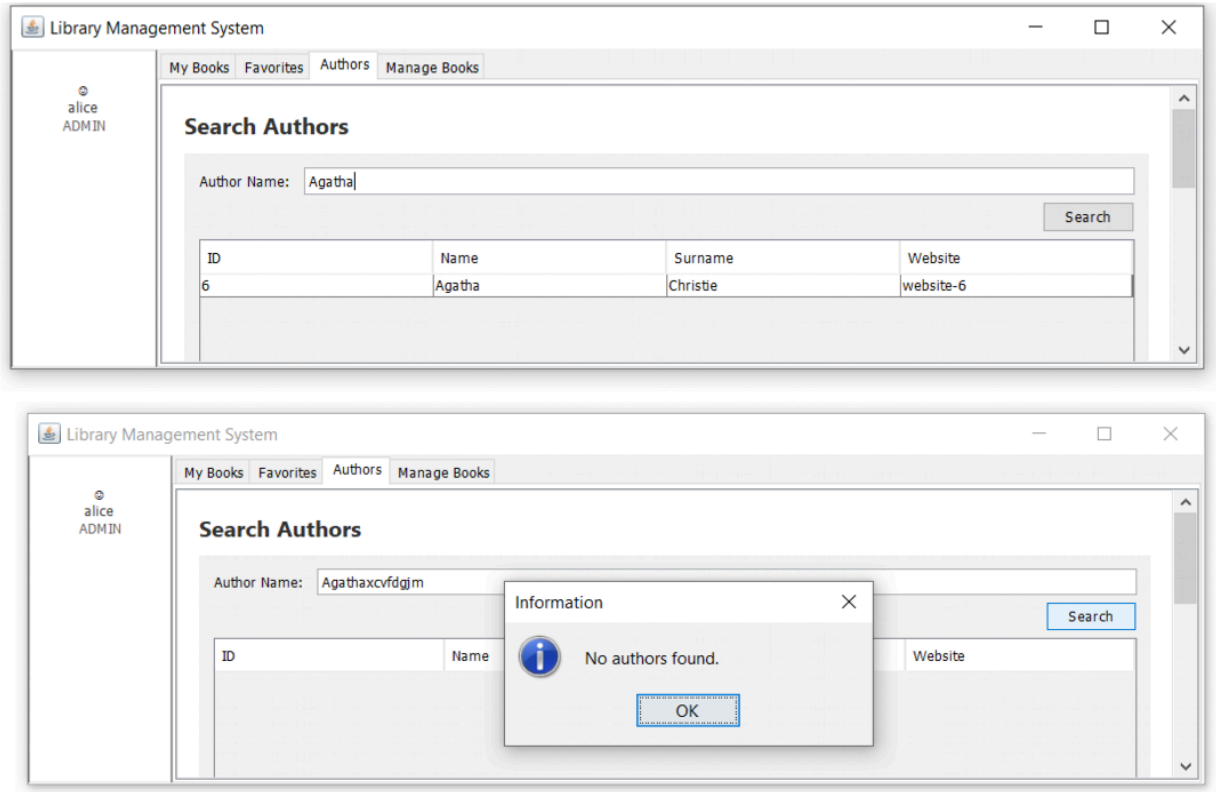
7.8 Favorite Authors (FavoriteAuthors.form)



In the view, we show the favorite authors (those who have at least 3 books in the current user's library, which means having them as read and wish to be read).

As author information, we display their ID, first name, last name, and website.

7.9 Search Author (SearchAuthor.form)

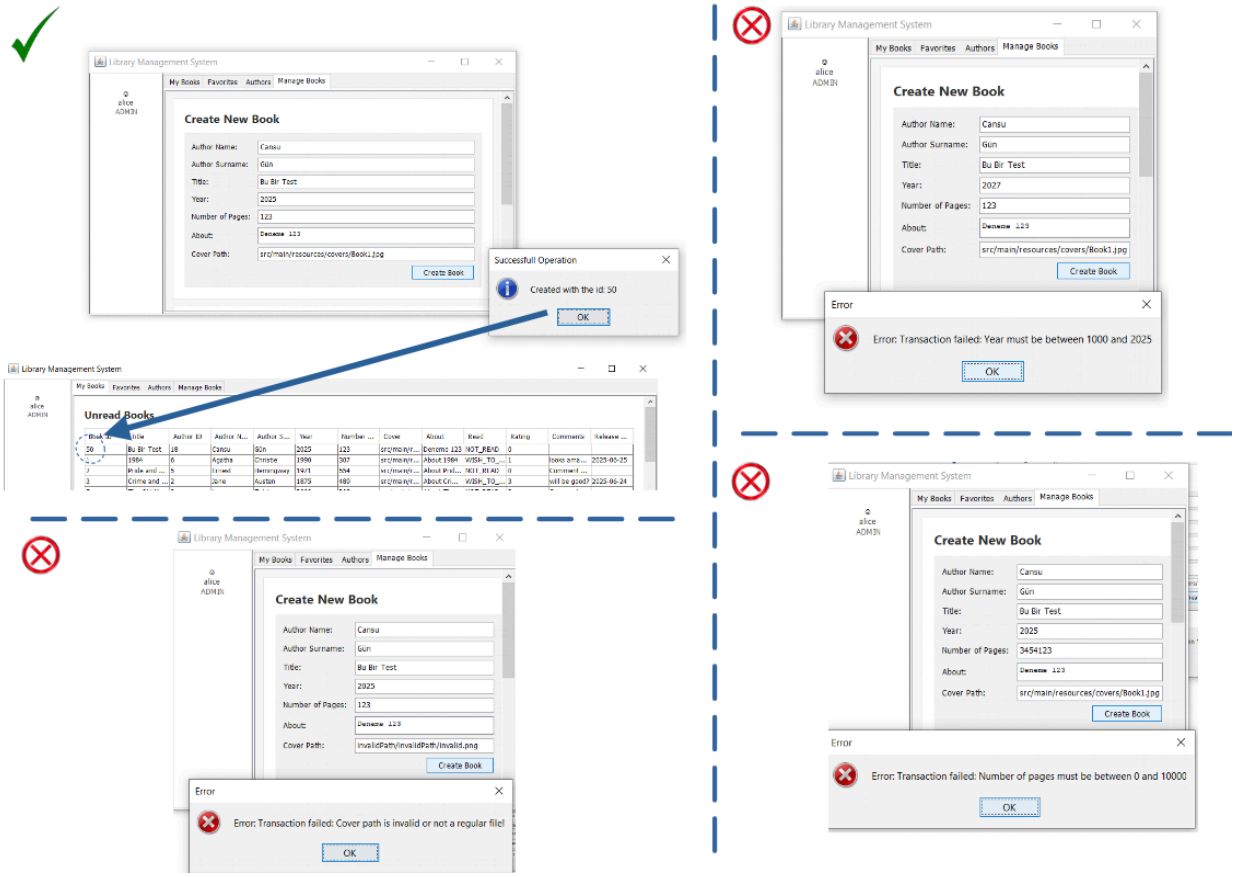


The top part of the image shows a successful search example. An author will be searched for by name (exactly, based on the requirements), and the result will be reflected in the data table.

As author information, we display their ID, first name, last name, and website.

In the bottom part of the image, you can see an example of a search with no results. The application layer throws an author domain not found exception, and the presentation layer reflects this with a dialog box.

7.10 Create New Book (BookCreation.form)



This feature is only available in both the GUI and the application layer (since it validates the current user type) for type 1 (admin) users.

Above left, you can see a correct insertion example. Here, we accept the following information from the client:

- Author's first name
- Author's last name
- Title
- Year
- Number of pages
- About
- Cover path

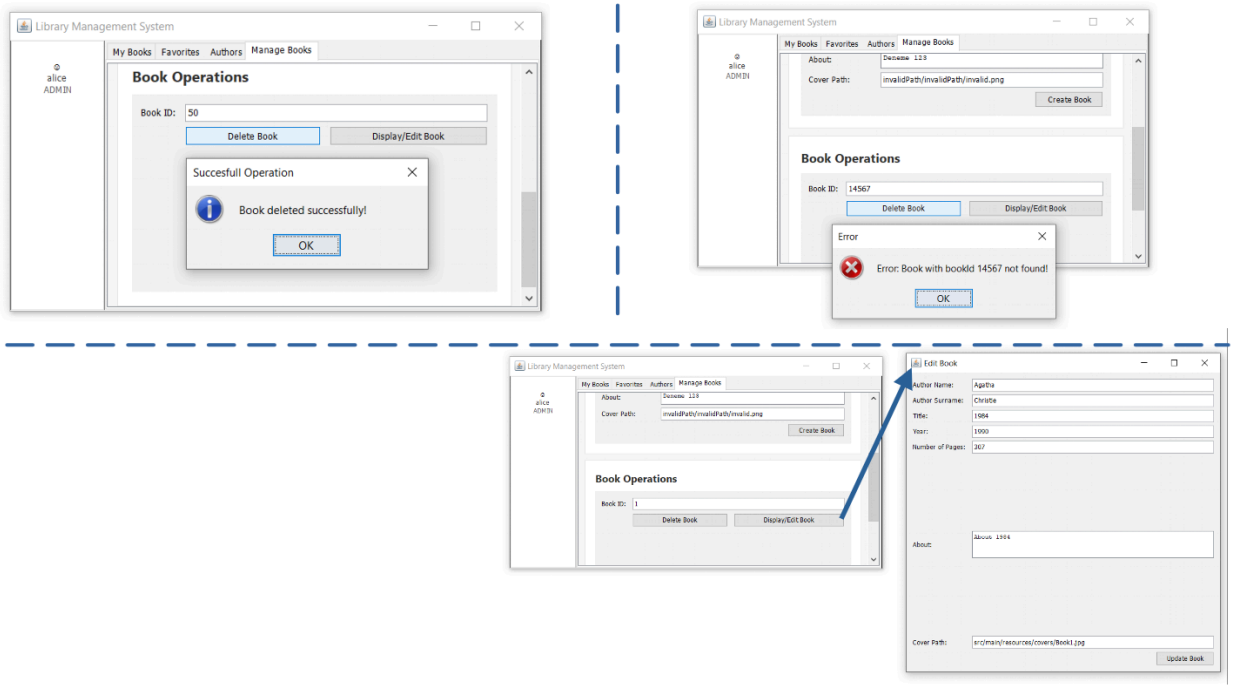
If an author with this information doesn't exist, it will be created and used for the inserted book. If it exists, it will be reused.

When an insertion occurs, the observer design pattern is used to propagate a refresh to the unread books view. This will reflect the newly inserted book as unread in the view.

The other parts of the image show examples of validations. They are domain validations; since they are domain-based, we can easily protect the book's logic. Which is;

- Enter a correct year (no larger than the current system year nor smaller than 1000 (this is an appropriate validation choice)
- Enter a correct page count (not empty, larger than 10000, nor smaller than 0)
- Validate the cover path (if it's not empty, validate the file exists and that it's not a folder)

7.11 Book Operations (BookOperations.form)



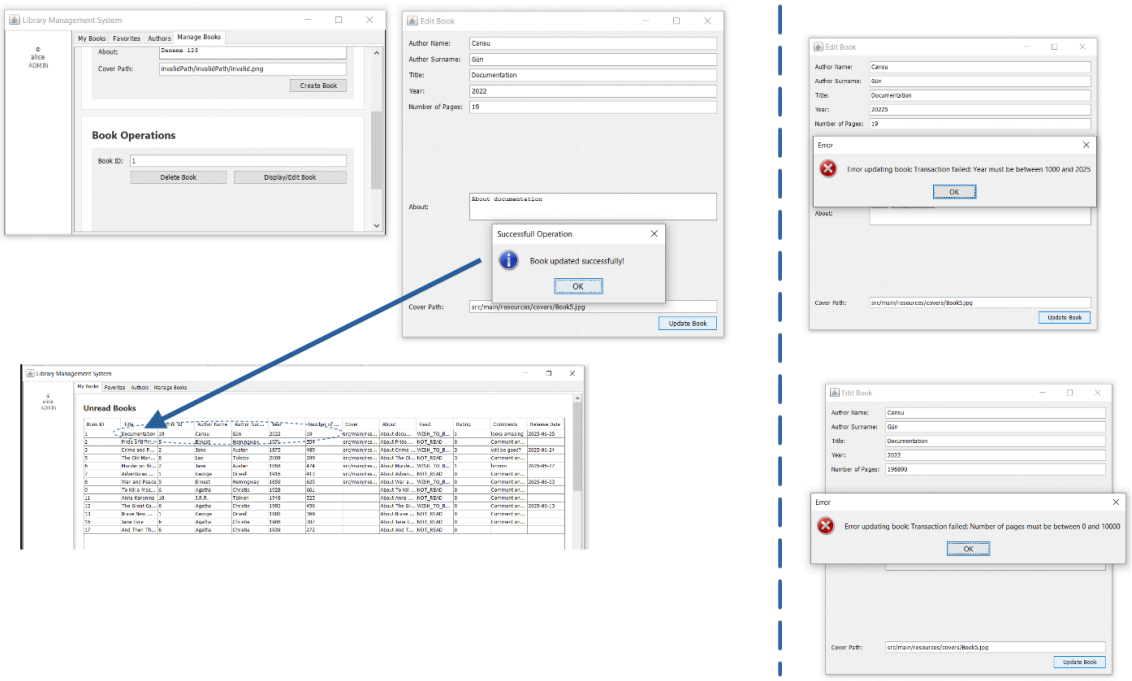
This feature is only available in both the GUI and the application layer (since it validates the current user type) for type 1 (admin) users. The GUI will not display this functionality because if the current user is not type 1, the Manage Books tab will not be displayed.

Above left, you can see a correct deletion example. To delete a book, it must exist. When a deletion occurs, a cascade of user book states and their respective comments occurs at the database level.

On the right, you can see the example of the fundamental validation: the existence of the book is a domain exception by the application layer. Here, we see the content of the exception as a dialog box.

At the bottom of the image you can see that pressing the display/edit book button will open the book editing view.

7.12 Book Edit (BookEdit.form)



This feature is only available in both the GUI and the application layer (since it validates the current user type) for type 1 (admin) users. The GUI will not display this functionality because if the current user is not type 1, the Manage Books tab will not be displayed.

Above left, you can see a correct editing example. Here, we accept the following information from the client:

- Author's first name
- Author's last name
- Title
- Year
- Number of pages
- About
- Cover path

Pay attention to the author section. Here, we use the same logic as when inserting and deleting books. If the author doesn't exist, it will be created and used for the inserted book. If it exists, it will be reused. The book's previous author will be taken into account. If the previous author is detected as an orphan (with no current related books), then the same logic that applies to deleting a book will be applied. Since there are no other references to books, the author will be deleted.

Note that at the end of the example in the upper-left part of the image, it is shown that the refresh propagation is resolved with the observer design pattern. In the photo, you can see that the changes were reflected for the unread books view.

On the right of the image, you can see examples of validations, which are the same validations applied when inserting a book. These are domain validations; since they are domain-based, we easily protect a book's logic. Which is;

- Enter a correct year (no larger than the current system year nor smaller than 1000 (this is an appropriate validation choice)
- Enter a correct page count (not empty, larger than 10000, nor smaller than 0)
- Validate the cover path (if it's not empty, validate the file exists and that it's not a folder)

8 Conclusion and Future Work

The project has been perfect for:

- Delve deeper into the requirements analysis, design, implementation, and testing phases (for the functional testing part).
- We have been able to apply what we learned in the last class labs (JDBC, hashing, Swing).
- understanding of waterfall, and due to development errors and understanding of requirements, the logic of agile
- Thanks to the project, we have delved deeper into the development of layered and onion architecture. We have been able to orchestrate multiple persistence actions and manage atomic actions through the use of transactions.
- Thanks to the use of onion architecture, without slicing logic, based on layers, and abstraction, we have been able to work in parallel across layers across our team.
- We have been able to delve deeper into design patterns for efficient use of database connections (singleton pattern) and correct rendering propagation of views (observer pattern).

The project will continue as a personal project for use in the portfolio, although it will also be used as a case study, to apply new technologies and architectures (spring boot, react, client server architecture with decoupled frontend) thanks to the plugable onion-style logic (isolation and abstraction), and to delve deeper into domain driver design, such as using identity factories, domain events, and expanding the project domain and using context maps.

In the future, such functionalities and improvements could be added;

- User Registration and Profile Management
- Advanced Search and Filtering (multi criterias of authors and books)
- Data export of data tables (export data as excel)
- Recommendation system of books for users based on their favorites and readed books pattern
    - Maybe with basic patterns, maybe with machine learning or deep learning
- UI themes as dark theme
- Image uploading for book covers
- Manual author insertion and update operations as current book operations
- İmprove the user experience of comments part
- Optimization of resources as connection pools of database connections for persistence layer
- Use of NoSQL databases for accelerate slow insertion operations as they are more fast in insertion cases
- Optimization of queries