Yaşar University
Spring, 2024 - 2025
SE 2232 - Software System Analysis
Final Project Report: Software Requirements Specifications Document (SRS)

Student Name:	Batuhan Salcan, Beril Filibelioğlu
Student No:	22070006040, 22070006042
Department Name:	Software Engineering
Course Section No:	2

This template is prepared based on the IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998).

Table of Contents (Do not change the Section Names!)

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the requirements of the desktop application “MyLibrary.” This document aims to clearly describe the functionalities, constraints, and behaviors of the system so that everyone involved in the project can understand what the system is supposed to do. It helps make sure that developers, testers, and users all have the same understanding of how the application will work.

1.2 Scope

a) The software product is named MyLibrary.

b) MyLibrary is a desktop application developed using Java. It helps users keep track of the books they have read, haven’t read yet, or plan to read.

c) The goal of this project is to provide an easy way for users to manage their personal book lists. The system supports two types of users: some can only view book details, while others can also add, update, or delete them.

1.3 Definitions, acronyms, and abbreviations

- **GUI** – Graphical User Interface
- **IDE** – Integrated Development Environment
- **JFrame** – A Java Swing component used for building GUI windows
- **MySQL** – A relational database management system used to store data
- **AdvancedReader** – A Type-1 user who has full access to all functionalities, including adding, deleting, and updating books
- **BasicReader** – A Type-2 user who can only view book and author information and access limited functionalities
- **read=1** – The user has read the book
- **read=2** – The user has not read the book
- **read=3** – The user wishes to read the book

1.4 References

- SE2232 Final Project Document – Yaşar University (Spring 2024–2025)
- Java 17 Documentation
- MySQL 8.0 Community
- NetBeans Documentation
- Visual Paradigm Help Resources

1.5 Overview

This document includes all the necessary details to describe how the MyLibrary desktop application will work. It starts with the purpose and scope of the system, then continues with important definitions and references. After that, the document presents the functional and non-functional requirements, the use case analysis, and the GUI design. Later sections include diagrams such as use case, sequence, class, and data flow diagrams. Finally, the report ends with a conclusion and suggestions for possible improvements in the future.

2 Design and Implementation Constraints

- The application will be developed in NetBeans, and the GUI will be built using Java Swing.
- All data will be stored in a MySQL database. The tables used will include: userinfo, books, and authors.
- Visual Paradigm should be used to create UML diagrams.
- The application will run offline, so no web integration is needed.
- It should be tested and used on a computer that has at least 4 GB of RAM and a dual-core processor.

3 Specific Requirements

3.1 Functional Requirements

1) The system shall allow readers to log in to the system by entering their username and password.

2) If the entered password is incorrect, the system shall display an error message and prompt the reader to enter the password again.

3) The system shall identify the user type after a successful login.

If the user is of type 1, the system shall display the AdvancedReader interface.

If the user is of type 2, the system shall display the BasicReader interface.

4) The system shall allow AdvancedReader to add a book to their personal library by entering title, year, number of pages, cover path, about, read status, rating, comments, release date and author name.

5) The system shall allow the AdvancedReader to select a read status when adding a book, where: read = 1 means the book has been read, read = 2 means the book has not been read and read = 3 means the user wishes to read the book.

6) The system shall check that whether the entered author exists in the authors table based on name and surname when the AdvancedReader adds a book.

7) The system shall assign a new authorId and insert a new record to the authors table if the author does not exist.

8) The system shall automatically increment the bookId by 1 for each new book added by any AdvancedReader.

9) The system shall allow AdvancedReader to delete a book from their personal library by entering the bookId.

10) The system shall display a confirmation message to the AdvancedReader after a deletion operation.

11) The system shall delete the corresponding author from the authors table if the deleted author has no other books in the user's personal library.

12) The system shall display the related information of the book to both BasicReader and AdvancedReader by entering bookId.

13) The system shall allow both BasicReader and AdvancedReader to search an author by name and then display author's information.

14) The system shall allow AdvancedReader to edit and update a book's information after displaying the information.

15) The system shall display the list of favorite books which are rated as 4 or 5 for both BasicReader and AdvancedReader.

16) The system shall display the list of favorite authors for both BasicReader and AdvancedReader, where an author is considered favorite if they have at least 3 books in the reader's personal library.

17) The system shall display the list of books that has not read yet for both BasicReader and AdvancedReader.

18) The system shall notify the AdvancedReader about the books that are on advanced reader's wish list and will be released within 1 week when the application is started.

19) The system shall display the cover image of a book to both BasicReader and AdvancedReader.

20) The system shall allow the AdvancedReader to rate a book from 1 to 5 only if the read status is 1 (read) otherwise, the rating shall be stored as 0.

21) The system shall allow the AdvancedReader to leave the comments field empty when adding or editing a book.

3.2 Performance Requirements

1. The system shall support at least 2 users accessing and performing actions (e.g., search, add, update) without any noticeable performance degradation. (Static – number of simultaneous users)

2. The system shall retrieve and display the book belonging to the id given within 2 seconds after clicking the "view book info" button.
(Dynamic – response time for user interaction)

3. The system shall be able to store at least 25 book records of user and when user wants to view them, shall display them within 2 seconds.

3.3 Software System Attributes

1. The system shall be portable and run on any operating system that supports Java Runtime Environment (JRE), including Windows, macOS**. (Maintainability)**

2. The login and main screens shall include labeled buttons, input fields with placeholder texts, and tooltips to guide the user through available actions. **(Usability)**

3. For each user action (add, delete, update), the system shall display a confirmation dialog if the operation succeeds, and an error dialog with a descriptive message if the operation fails. **(Reliability)**

4. BasicReader shall only see and access the following functions in the main interface: View Book Information, Search Author, View Favorite Books, View Favorite Authors, View Unread Books, and View Book Cover**. (Security)**

5. The system shall be modular and well-documented to allow future developers to easily understand, update, and extend individual components without affecting unrelated parts of the system. **(Maintainability)**

3.4 Use Case Analysis

3.4.1 Actors

AdvancedReader

An AdvancedReader is a Type-1 user who can perform all actions in the system. This includes adding, deleting, and updating books in their personal library. They have full access to all features and can fully manage their own book collection.

BasicReader

BasicReader is a Type-2 user who has limited access to the system. They can only view book and author information in their personal library. Additionally, they can view their favorite books, favorite authors, unread books, and receive notifications about upcoming wishlist books. BasicReaders cannot add, delete, or update books.

3.4.2 Scenarios

1)

Use Case Name: View book Details

Actors: AdvancedReader, BasicReader

Scenario:

When a AdvancedReader or BasicReader successfully logs into the system, they can browse through the list of books available in the library. When the reader decides to view more information about a specific book, they can see that with entering bookId. The system then retrieves and displays detailed information about the selected book, such as the book's title, author, publication year, number of pages, cover image, and any additional comments or descriptions provided. If the book has been rated, the system will display the rating as well. If there is no detailed information available for the selected book, the system will notify the reader with a message indicating that the book does not have any details available at the moment. After viewing the details, the reader can choose to view other books, search for a different book, or return to the main page to perform other actions.

2)

Use Case Name: Edit and Update Book Info

Actors: AdvancedReader

Scenario: AdvancedReader reviews the list of books in their personal library and notices an error in the details of a previously added book. To correct the mistake, AdvancedReader clicks the "Update Book" button. After this action, AdvancedReader is directed to an "Edit Book Form" that is pre-filled with the current book details(after writing id of the book). In this form, AdvancedReader updates the book's title, adjusts the page count, and enters a new description in the "About" section. Additionally, AdvancedReader updates the "Read" status and modifies the rating for the book within the acceptable range. After making the necessary updates, AdvancedReader clicks the "Update Book" button to confirm and save the changes. The system checks the data entered by AdvancedReader. After the update is successfully completed, the system displays a success message to AdvancedReader.Finally, the book list is updated on the interface, and AdvancedReader can view the updated information.

3.4.3 Use Case Forms

Complete Use Case Form 1

Use Case Name: Add a Book

Actors: AdvancedReader

Description: This use case describes how AdvancedReader adds a book to his/her personal library.

Trigger: AdvancedReader decides to add a book to their personal library.

Preconditions:

- The reader is logged in to the system as AdvancedReader (Type 1 user).

- The book datastore is available and online.

-The authors datastore is available and online.

Normal Flow:

1. AdvancedReader clicks to add a book option.
2. The system displays the book addition page.
3. AdvancedReader enters the book details.
4. The system checks if the author in the database based on author's name and surname.
5. The author exists in the database, system gets the authoId from the database.
6. The system increases bookId by 1.
7. The system adds the book to books table.
8. The system confirms that the book has been successfully added to the personal library and notifies the AdvancedReader.

Alternate Flow:

5.1 The author does not exists in the database.

1. The system assigns a new authorId to the author.
2. The system inserts a new record for the author into the authors database.

Post conditions:

-The new book is successfully added to the reader's personal library.

-If the author was not already in the database, a new author record has been created and linked to the book.

Exceptions:

E1: Invalid book details (occurs at step 2)

1. If any required fields (such as title, author name, etc.) are missing or incorrectly filled, the system displays an error message prompting the AdvancedReader to complete the form correctly.
2. The system prevents the book from being added until all the necessary details are provided

Complete Use Case Form 2

Use Case Name: Delete a Book

Actors: AdvancedReader

Description: This use case describes how AdvancedReader deletes a book from his/her personal library.

Trigger: AdvancedReader decides to delete a book to their personal library.

Preconditions:

- The reader is logged in to the system as AdvancedReader (Type 1 user).
- The book datastore is available and online.
- The authors datastore is available and online.
- The bookId of the book to be deleted exists in the reader's personal library.

Normal Flow:

1. AdvancedReader selects the delete book option.
2. The system displays the book deletion page.
3. AdvancedReader enters the bookId he/she wants to delete.
4. The system checks if the book exists in the AdvancedReader's personal library database.
5. If the book exists, the system deletes the book from the books table.
6. The system checks if the deleted book's author has other books in the reader's library.
7. If the author has no other books in the personal library, the system deletes the author record from the authors table.
8. The system confirms that the book has been successfully deleted from the personal library and notifies the AdvancedReader.

Alternate Flow:

A1: Author has other books in the library (occurs at step 7)

1. If the author has other books, the system does not delete the author record from the authors table.
2. The system confirms that the author is still present in the database and no changes are made to the authors table.

Postconditions:

- The selected book is successfully removed from the reader's personal library.
- If the author had no other books in the reader's library, the corresponding author record is deleted from the authors table.

Exceptions:

E1: Book does not exist in the personal library (occurs at step 3)

1. If the entered bookId is not valid or does not exists in the personal library, the system displays an error message to the AdvancedReader.

2. The system asks AdvancedReader to enter the bookId again.

Complete Use Case Form 3

Use Case Name: Search Book by Name

Actors: BasicReader, AdvancedReader

Description: This use case describes how the user searches for books in the library system.

Trigger: The user enters the book's name in the search bar and clicks “Search”.

Preconditions:

- The user is logged in as BasicReader.

- The search bar is available on the main interface.

- Books and authors are already stored in the database.

Normal Flow:

1. The user logged in as BasicReader.
2. The user types the book's name in the "Search" field on the main screen.
3. The user clicks the "Search" button.
4. The system performs a search in the database based on the book's name.
5. The system processes the query and returns a list of books.

Alternate Flow:

A1: User is AdvancedReader (occurs at step 1)

1.Same flow continues. The user has extended privileges.

Postconditions:

-The system displays the books information.

-BasicReader can only view the results, not make any changes.

-AdvancedReader may proceed to add new entries if desired.

Exceptions:

- If no books are found, the system shows the message: “No books found by this author.”

- If there is an issue with the database, the system displays an error message: “Error occurred during search. Please try again.”

Basic Use Case Forms

1)

Use case name: View Favorite Books

Actors: AdvancedReader, BasicReader

Description: This use case describes how reader view his/her favorite books. Favorite books are those that have been rated 4 or 5 stars.

Trigger: Reader selects view fovorite books option.

Preconditions:

- The reader is logged in to the system.
- The book datastore is available and online.
- The book is marked as “read=1” in database.
-The book in the database is rated as 4 or 5 stars. (These are considered as “favorite books”)
- The reader has books rated 4 or 5 stars in their personal library.

2)

Use case name: View Favorite Authors

Actors: AdvancedReader, BasicReader

Description: This use case describes how reader view their favorite authors. Favorite authors are those who have at least 3 books in the reader’s personal library.

Trigger: Reader selects view fovorite authors option.

Preconditions:

- The reader is logged in to the system.
- The book datastore is available and online.
- The author datastore is available and online.
- The reader has at least 3 books from an author in their personal library.

3)

Use case name: Notify Upcoming Wishlist Books

Actors: AdvancedReader

Description: This use case describes how the AdvancedReader is notified by the system about books in their wishlist (which means books marked as “read=3") that are going to be released in the next week.

Trigger: The AdvancedReader logs in to the application.

Preconditions:

- The reader is logged in to the system as AdvancedReader.
- The book datastore is available and online.
- The releaseDate attribute for books which are marked as “read=3" is set within the next week.

4)

Use case name: View Book Cover Image

Actors: AdvancedReader, BasicReader

Description: This use case describes how the reader views the book cover image by entering the id of a book. The system loads the image from the path stored in the cover attribute of the book in the database.

Trigger: The reader selects the view book cover image option.

Preconditions:

- The reader is logged in to the system.
- The book datastore is available and online.
- The cover attribute of the book in the database contains a valid path to the image.

5)

Use Case Name: Login

Actors: BasicReader, AdvancedReader

Description: This use case describes how the reader enters their username and password to access the system. Based on the credentials, the system grants access to either BasicReader or AdvancedReader privileges.

Trigger: The reader opens the application and clicks the “Login” button.

Preconditions:

- The reader has registered with valid login credentials.
- The system is operational and connected to the reader database.

6)

Use Case Name: Search Author by Name

Actors: BasicReader, AdvancedReader

Description: This use case describes how the reader searches for an author by name. Then the system retrieves the authors information and list of books written by that author.

Trigger: The reader enters the author's name in the search bar and clicks “Search”.

Preconditions:

- The reader is logged into the system.
- The database contains books with author information.

7)

Use Case Name: Edit and Update Book Info

Actors: AdvancedReader

Description: The AdvancedRreader modifies the details of an existing book in the system. Changes may include the title, author, or description.

Trigger: The reader selects a book and clicks the “Edit” button to update the book’s information.

Preconditions:

- The reader is logged in as a AdvancedReader.
- The book to be edited exists in the database.

8)

Use Case Name: View Unread Books

Actors: BasicReader, AdvancedReader

Description: The user views a list of books they have not yet read.

Trigger: The user clicks the “View Unread Books” button in the main interface.

Preconditions:

-The user is logged in.

-Books are tagged with “read=2” in the database.

9)

Use Case: View Book Details

Actors: BasicReader, AdvancedReader

Description: The user views detailed information about a selected book.

Trigger: The user clicks on the view book information button.

Preconditions:

-The user is logged in.

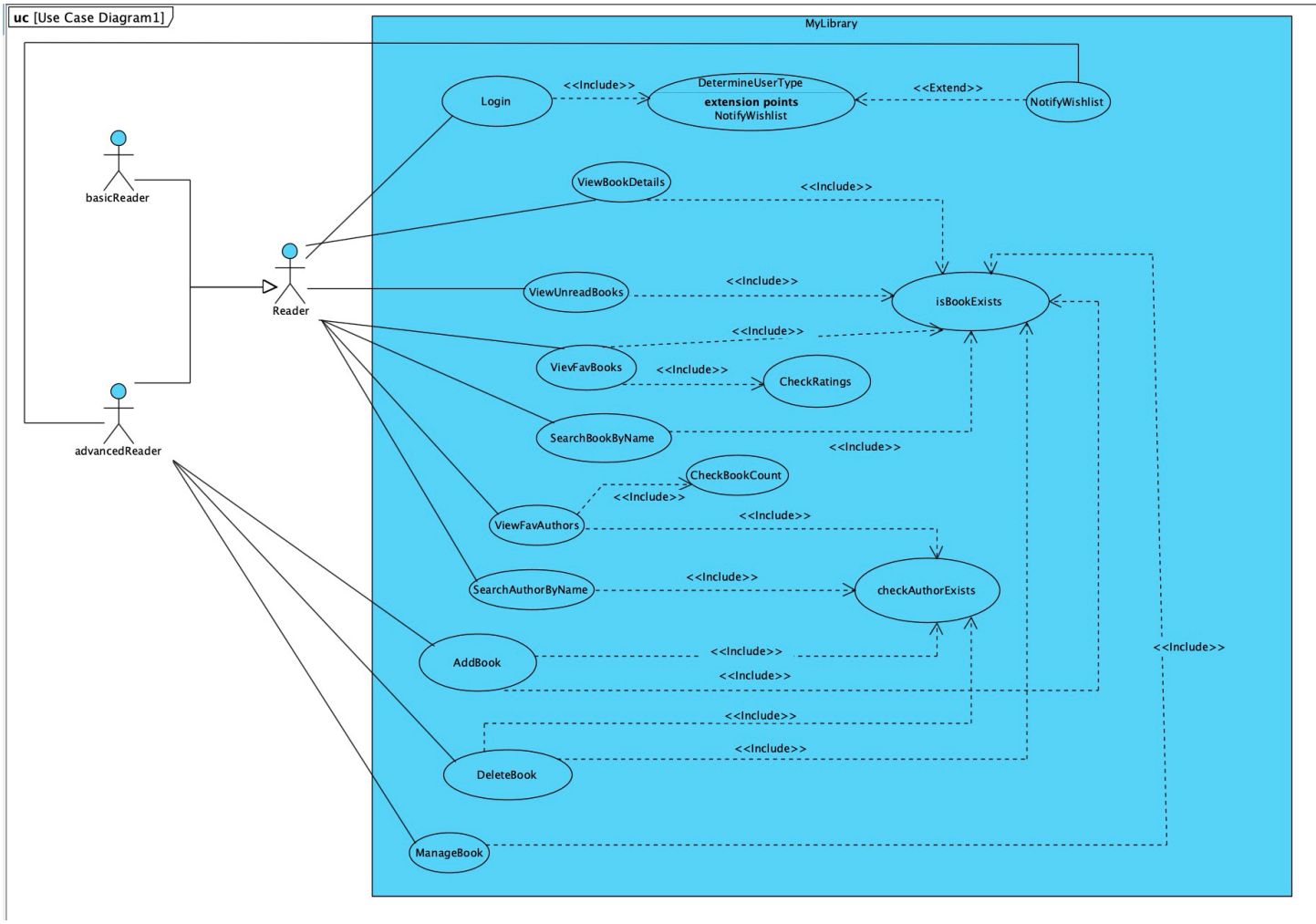
-Books are stored in the system.

3.4.4 Relationships among Actors and Use Cases

Both Advancedreader and BasicReader: Login, selects ViewBookDetails(this also means ViewBookCoverImage), selects ViewUnreadBooks, selects ViewFavBooks, selects SearchBookByName, selects ViewFavAuthors, selects SearchAuthorByname.

AdvancedReader: selects AddBook, selects DeleteBook, selects ManageBook, notified by system (NotifyWishlist).

3.4.5 Use Case Diagram



Login -> includes DetermineUserType : The system must first determine user type for successful login operation.

DetermineUserType <- extends NotifyWishlist : The system notifies AdvancedReader if there is an upcoming book that will release in one week.

ViewBookDetails -> includes isBookExist : For reader to view book details the system must check is the book exists in database.

ViewFavBook -> includes isBookExist : For reader to view favorite books the system must check is the book exists in database.

ViewFavBook -> includes CheckRatings: For reader to view favorite books the system must check is the book has rate 4 or 5 in the database.

SearchBookByName -> includes isBookExist : For reader to search book by name the system must check is the book exists in database.

SearchAuthorByName -> includes CheckAuthorExists : For reader to search author by name the system must check is the author exists in database.

ViewFavAuthor -> includes CheckAuthorExists: For reader to view favorite authors the system must check is the author exists in database.

ViewFavAuthor -> includes CheckBookCount: For reader to view favorite authors the system must check is the author has at least 3 books in the database.

AddBook -> includes isBookExist : For AdvancedReader to add book, the system must check is the book exists in database.

AddBook -> includes CheckAuthorExists: For AdvancedReader to add book, the system must check is the author exists in database. If the author is not exists in database, the system must add the new author to the database.

DeleteBook -> includes isBookExist : For AdvancedReader to delete book, the system must check is the book exists in database.

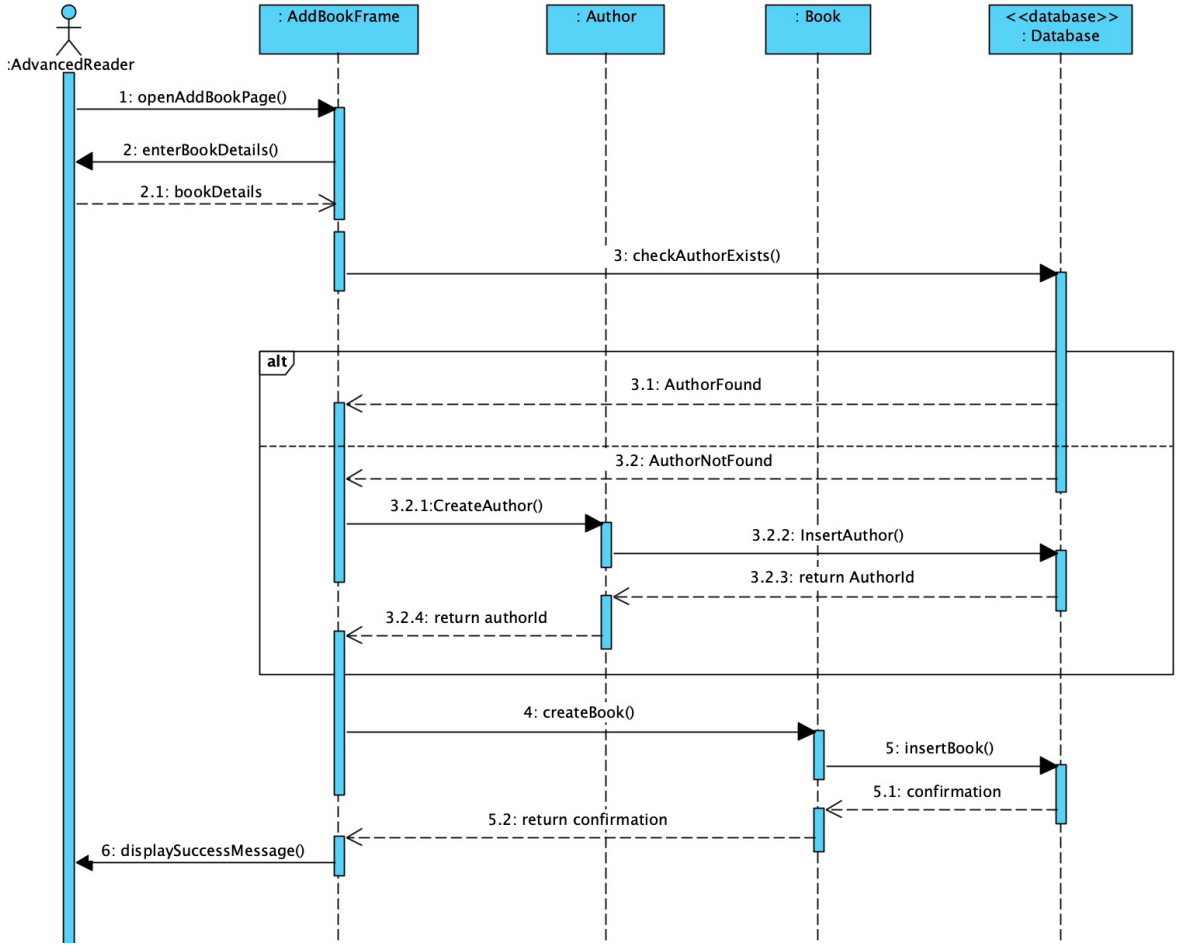
AddBook -> includes CheckAuthorExists: For AdvancedReader to delete book, the system must check author exists and has another books in database. If the author hasn't got any other books in database, the system must delete the author from database.

ManageBook -> includes isBookExist : For AdvancedReader to manage book, the system must check is the book exists in database.

4 Behavioral Models

4.1Sequence Diagram

1)Add book sequence diagram



openAddBookPage() : The user opens the page/interface to add a new book.

enterBookDetails() : The user enters all the required details about the book in the form.

checkAuthorExists() : The system checks the database to see if the author already exists.

AuthorFound : If the author is found in the database, the system proceeds using the existing authorId.

AuthorNotFound : If the author is not found, the system takes the alternative path to create a new author.

CreateAuthor() : System starts the process of creating a new author record.

InsertAuthor() : The new author is inserted into the authors table in the database.

return authorId : The database returns the newly created authorId .

return authorId : System receives the authorId.

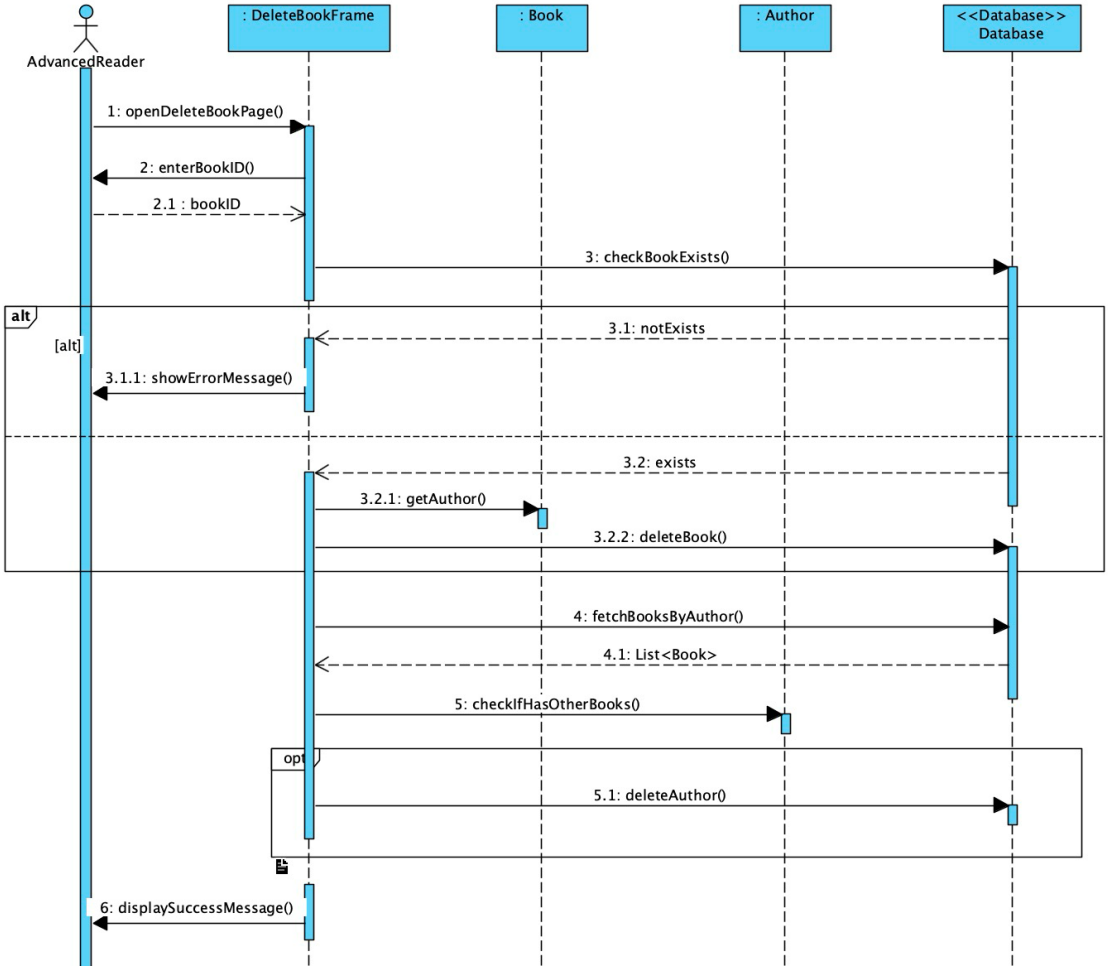
createBook() : System creates a new book object with the authorId and book details.

insertBook() : The new book is inserted into the books table in the database.

Confirmation : The database confirms successful insertion.

displaySuccessMessage() : The UI displays a success message to the user.

2)Delete book sequence diagram



openDeleteBookPage() : The user opens the page/interface to start the book deletion process.

enterBookID() : The user enters the ID of the book they want to delete.

checkBookExists() : The system checks in the database whether the entered book ID exists.

NotExists : Indicates that the book with the given ID was **not found** in the database.

showErrorMessage() : The system shows an error message to the user indicating the book was not found.

Exists : Confirms that the book exists in the database.

getAuthor() : Retrieves the author information associated with the deleted book.

deleteBook() : The system deletes the book record from the database.

fetchBooksByAuthor() : Queries the database to find other books by the same author.

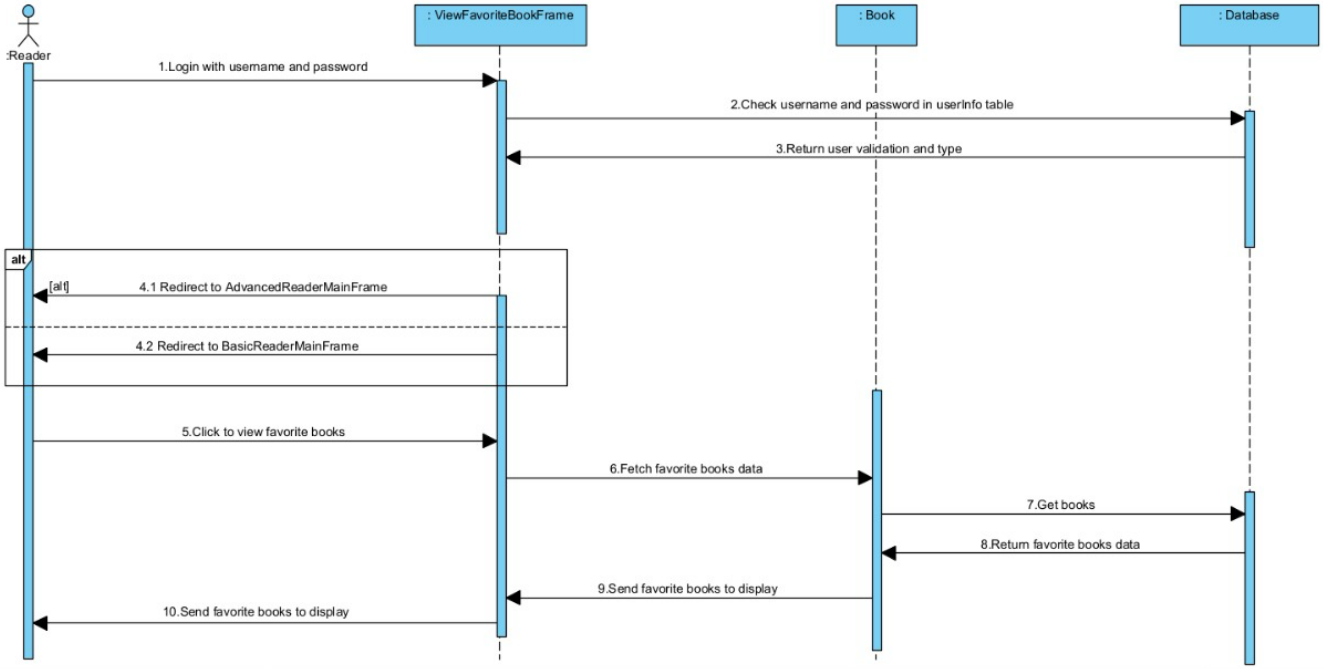
List<Book> : Returns a list of books written by the author.

checkIfHasOtherBooks() : Checks if the author has any other books besides the one being deleted.

deleteAuthor() : Deletes the author record from the database if no other books are associated.

showSuccessMessage() : Displays a success message confirming the book deletion.

3)View Favorite Books Sequence Diagram



1.Login with username and password Reader → ViewFavoriteBookFrame The user initiates the login process by entering their username and password into the login screen

2.Check username and password in userInfo table ViewFavoriteBookFrame → Database The system sends a request to the database to verify whether the provided username and password match any record in the userInfo table.

3.Return user validation and type Database → ViewFavoriteBookFrame The database returns a response indicating whether the login is successful and specifies the user type (AdvancedReader or BasicReader).

4.1 Redirect to AdvancedReaderMainFrame (if user is Type 1) Alternative Fragment (alt block) If the logged-in user is an AdvancedReader, the system redirects them to the AdvancedReaderMainFrame.

4.2 Redirect to BasicReaderMainFrame (if user is Type 2) Alternative Fragment (alt block) If the logged-in user is a BasicReader, the system redirects them to the BasicReaderMainFrame.

5.Click to view favorite books Reader → ViewFavoriteBookFrame The user clicks the option/button to view their list of favorite books.

6.Fetch favorite books data ViewFavoriteBookFrame → Book The GUI sends a request to the Book class or controller to retrieve the list of favorite books for the current user.

7.Get books Book → Database The Book class queries the database to obtain the favorite book records associated with the current user.

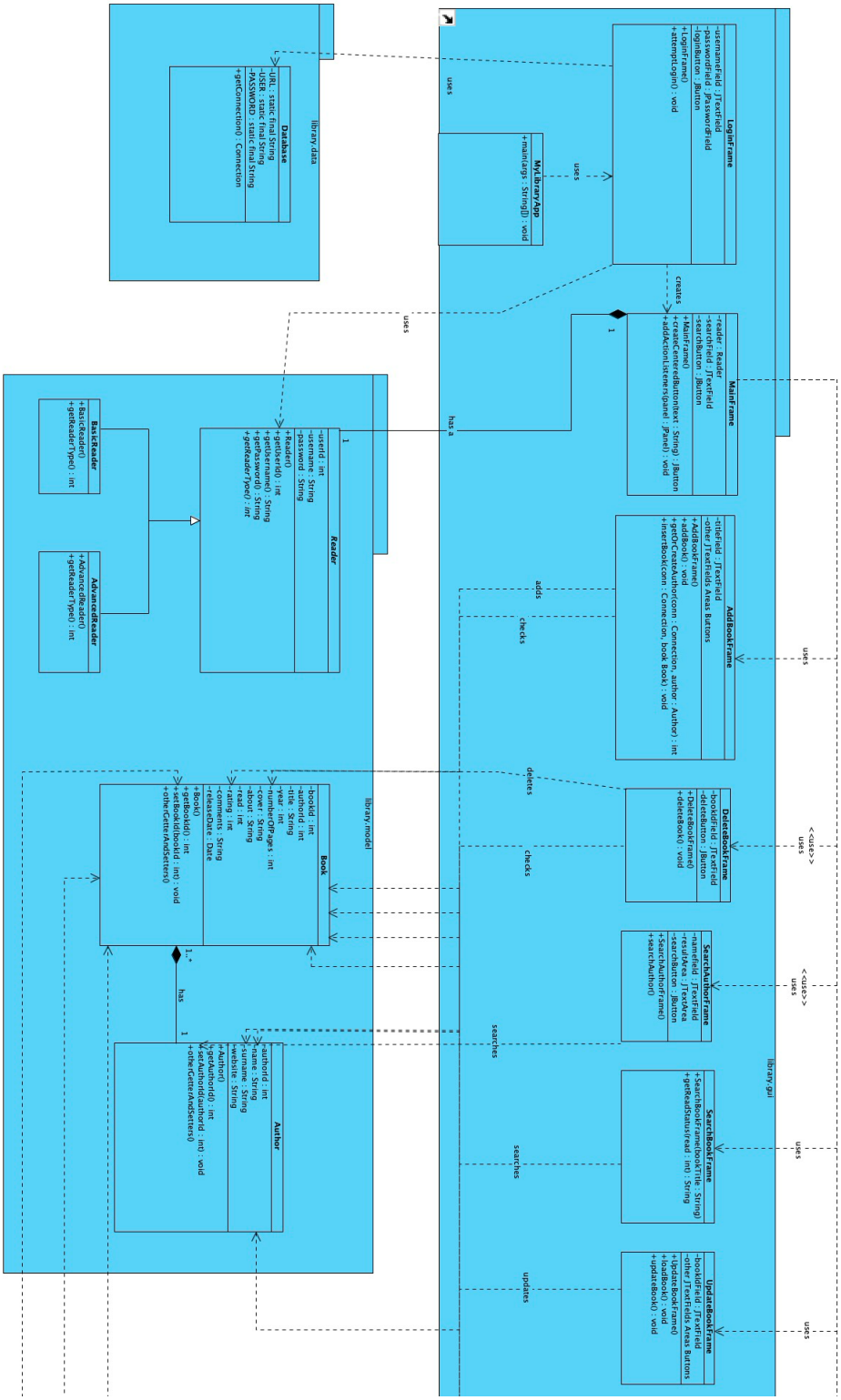
8.Return favorite books data Database → Book The database returns the relevant favorite book data to the Book class.

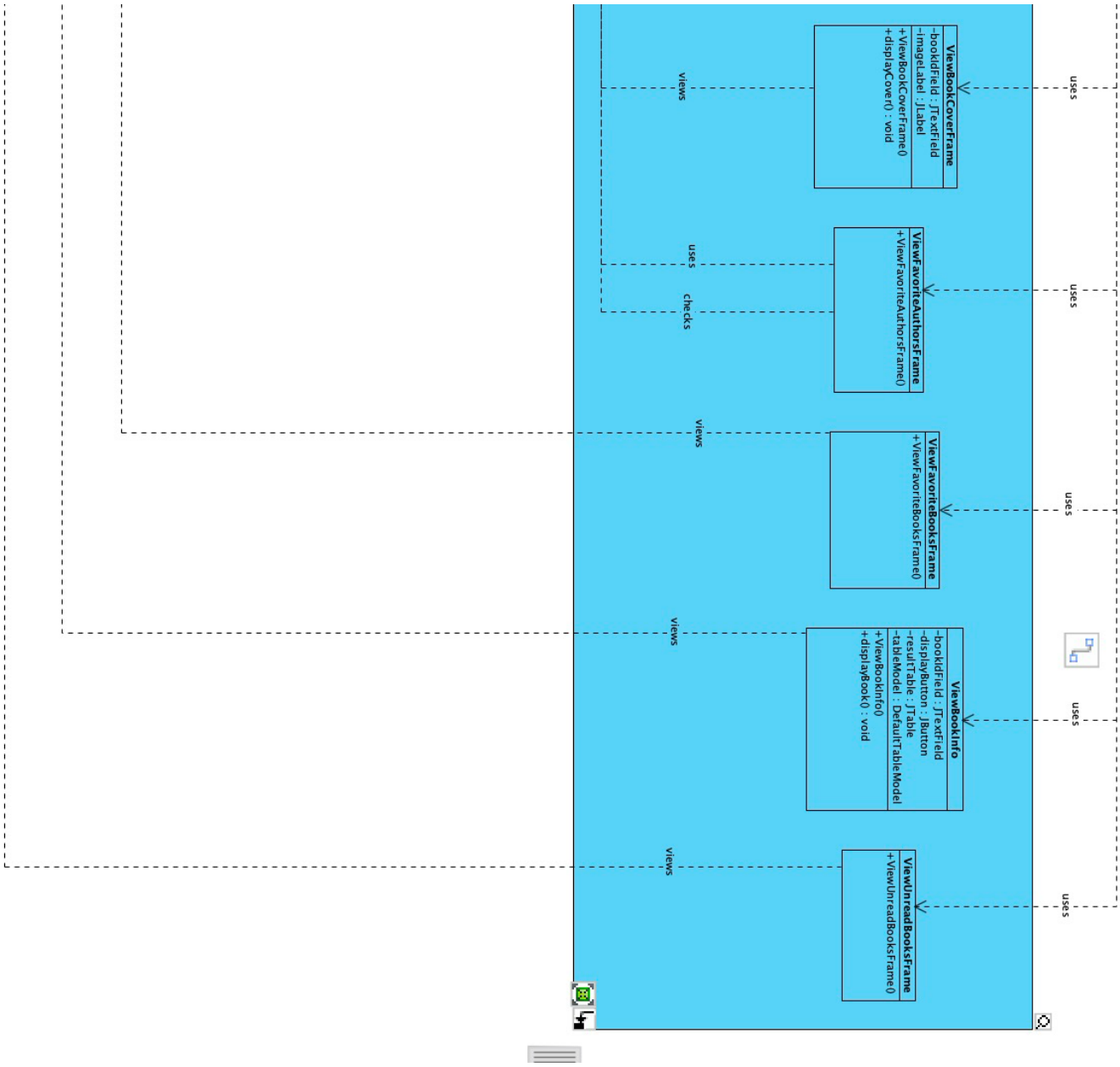
9.Send favorite books to display Book → ViewFavoriteBookFrame The Book class sends the retrieved list of favorite books to the GUI (ViewFavoriteBookFrame).

10.Send favorite books to display ViewFavoriteBookFrame → Reader Finally, the GUI displays the favorite books to the user on the screen.

5 Structural Models

5.1 Class Diagram





<u>The full photo of our class diagram is in our project files.</u>

The reason we used dependency is for classes uses other classes in their methods. Classes may exist without other classes because they don't use other classes in their attributes. They call a method that they used other classes when they need other classes.

The reason we used composition is mainframe used reader class in it's attribute. Mainframe class cannot exist without Reader class.

The reason we used composition between Book and Author classes is a book needs to written by an author. An author can write multiple books.

In code all frame classes connects to database and in database uses book or author. But we assumed in the class diagram that all frames directly uses book or author class.

APPLICATION ENTRY

1. MyLibraryApp

Uses → LoginFrame

Acts as the application entry point (contains main method).

Launches the login window.

LOGIN & SESSION CONTROL

2. LoginFrame

Creates → MainFrame

After successful login, it opens the MainFrame.

Uses → Database

Connects to the database to validate credentials.

Uses → Reader

Determines the reader type (Basic or Advanced) upon login and passes it to MainFrame.

MAIN INTERFACE

3. MainFrame

Has a → Reader

Maintains a reference to the logged-in reader (either BasicReader or AdvancedReader).

Uses → All GUI frames:

AddBookFrame,DeleteBookFrame,SearchAuthorFrame,SearchBookFrame,UpdateBookFrame,ViewBookCoverFrame,ViewBookInfo,ViewUnreadBooksFrame,ViewFavoriteBooksFrame,ViewFavoriteAuthorsFrame

GUI FRAMES (library.gui)

4. AddBookFrame

Adds → Book

Checks → Author

Checks whether the author exists; if not, it can add a new author.

5. DeleteBookFrame

Deletes → Book by ID

Checks → Author

6. SearchAuthorFrame

Searches → Author by name/surname

Checks → Book (number of books of author)

7. SearchBookFrame

Searches → Book by title

Uses → getReadStatus

8. UpdateBookFrame

Updates → Fields like read, rating, comments in Book

9. ViewBookCoverFrame

Views → Displays book cover (image path)

10. ViewBookInfo

Views → Full book details in a table format

11. ViewUnreadBooksFrame

Views → All unread books (based on read status)

12. ViewFavoriteBooksFrame

Views → Favorite books of the logged-in reader

13. ViewFavoriteAuthorsFrame

Views → Favorite authors

MODEL LAYER (library.model)

14. Reader *(abstract class)*

Used By → LoginFrame, MainFrame

Subclasses:

BasicReader

AdvancedReader

15. Book

Has → 1 Author (via authorId)

Used By → Most GUI frames (Add, Delete, View, Update, etc.)

16. Author

Has → 1..* Book instances

Used By → AddBookFrame,DeleteBookFrm, SearchAuthorFrame, ViewFavoriteAuthorsFrame

DATABASE CONNECTION (library.data)

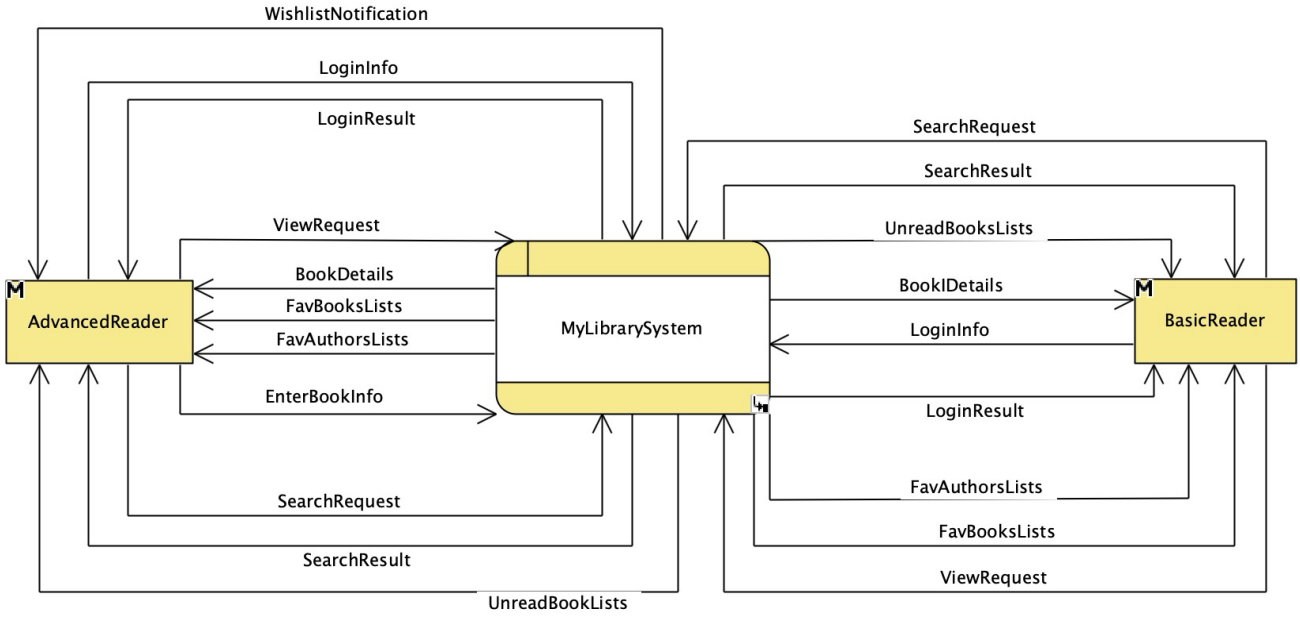
17. Database

Used By → LoginFrame

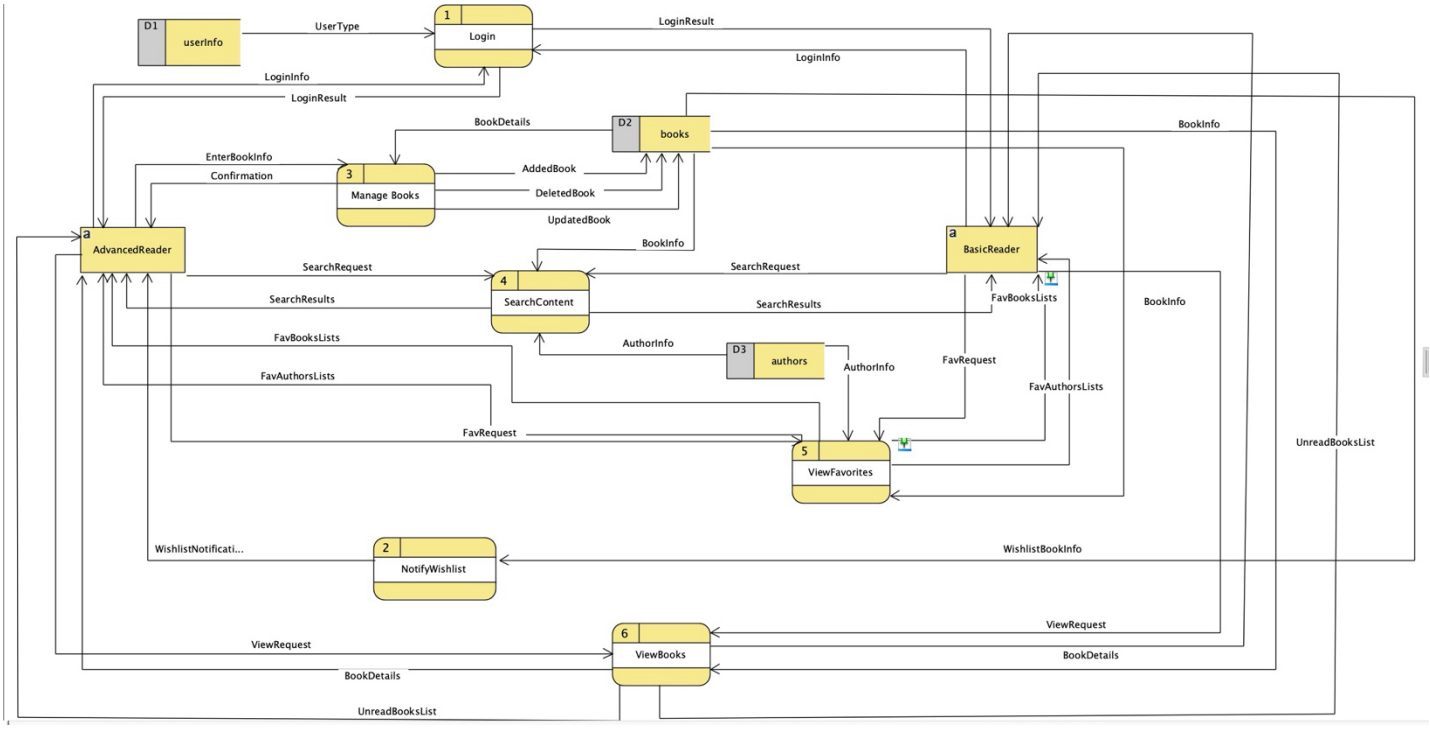
6 Process Modeling

6.1 Data Flow Diagram (DFD)

Context Diagram



Level 0 DFD



Process 1: Login

Use Case(s):

- Login

Data Flow Explanation:

Users (BasicReader or AdvancedReader) provide login credentials (username and password), which are sent to the system.

The system checks the userInfo data store to validate the credentials.

The system returns a login result indicating success or failure.

Process 2: Notify Wishlist

Use Case(s):

- Notify Upcoming Wishlist Books

Data Flow Explanation:

Triggered when an AdvancedReader logs in.

The system checks the books data store for books marked as read=3 (wishlist) with a releaseDate within the next 7 days.

If matches exist, a notification is shown.

Process 3: Manage Books

Use Case(s):

- Add a Book
- Update Book Info
- Delete a Book

Data Flow Explanation:

AdvancedReader submits book-related actions (add, edit, or delete).

- For adding: the system checks if the author exists in the authors store; if not, it creates a new record. Then the book is added to the books store.
- For editing: the book is retrieved and updated in books.
- For deleting: the book is removed from books, and if the author has no other books, the author is deleted from authors.

Process 4: Search Content

Use Case(s):

- Search Book by Name
- Search Author by Name

Data Flow Explanation:

Both BasicReader and AdvancedReader can enter search requests (book name or author name).

The system queries the books or authors data store and returns matching results.

Process 5: View Favorites

Use Case(s):

- View Favorite Books
- View Favorite Authors

Data Flow Explanation:

The user requests to view their favorite books (rated 4 or 5) or favorite authors (who have ≥3 books in the reader's library).

The system filters and retrieves the related records from books.

Process 6: View Books

Use Case(s):

- View Book Details
- View Book Cover Image
- View Unread Books

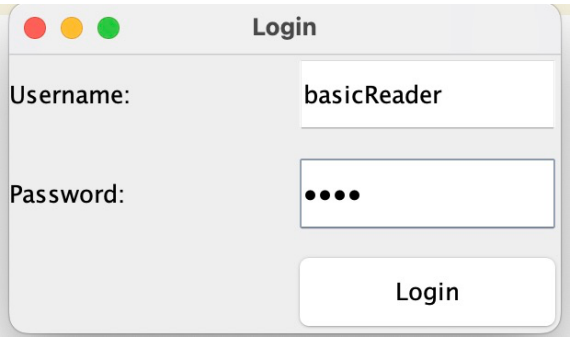
Data Flow Explanation:

The user requests to view a book's full details(includes cover image), or unread books.

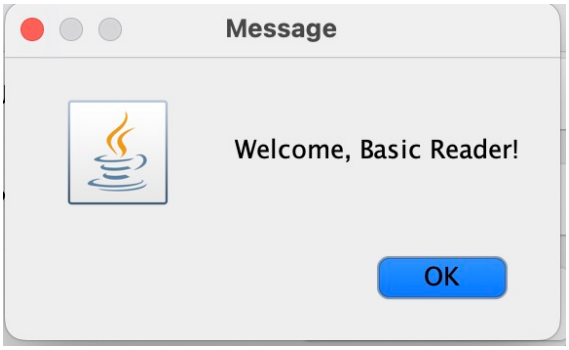
The system queries the books data store to retrieve detailed information, image path (cover), or books with read=2 status.

7 Graphical User Interface(s) (GUIs)

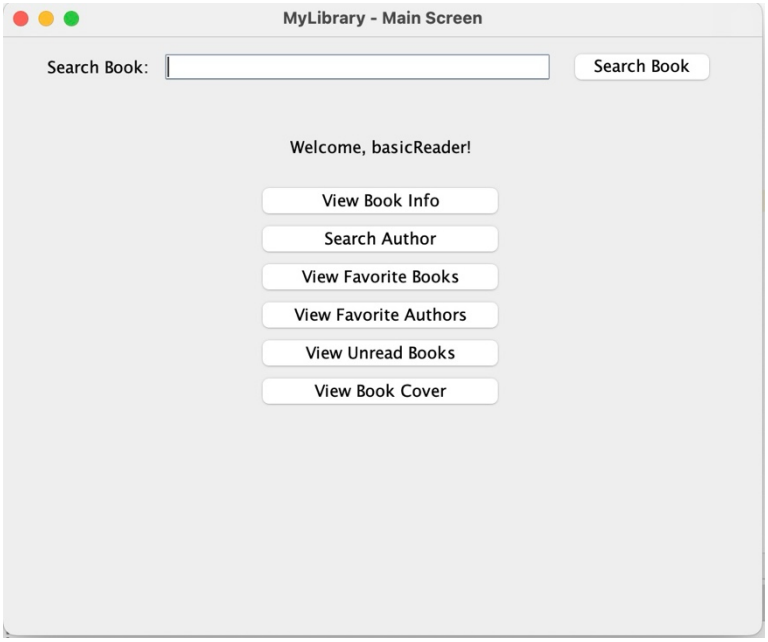
Login Frame For basicReader (type-2 user)



Login Confirmation

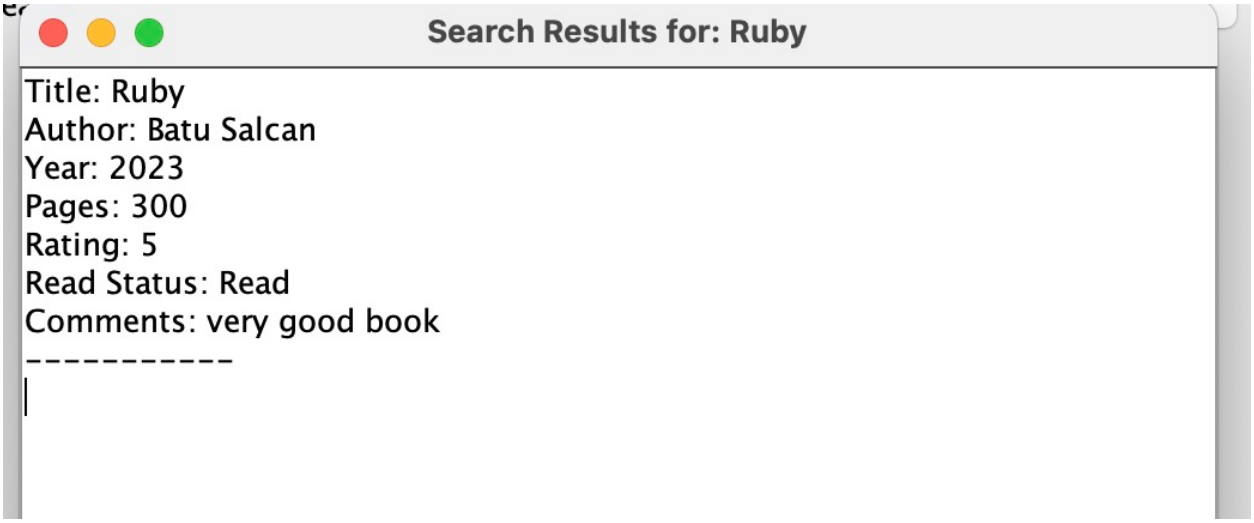


BasicReader Mainframe

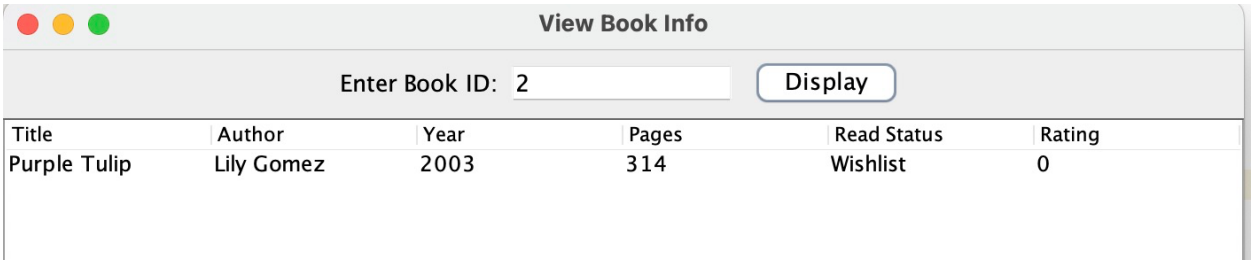


Search Book By Name

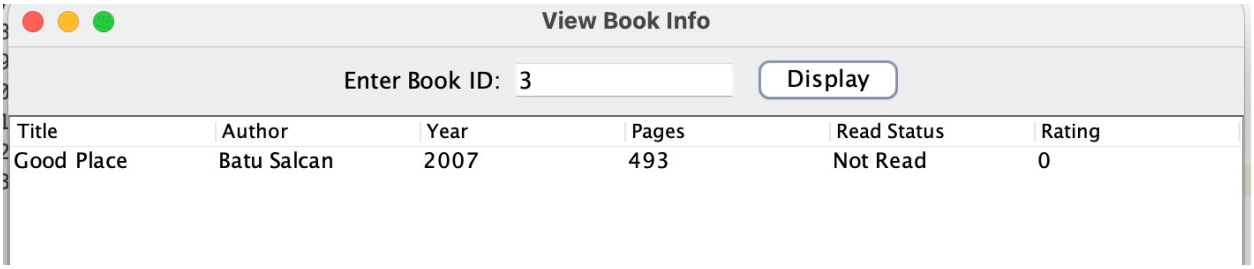
Search Book Result



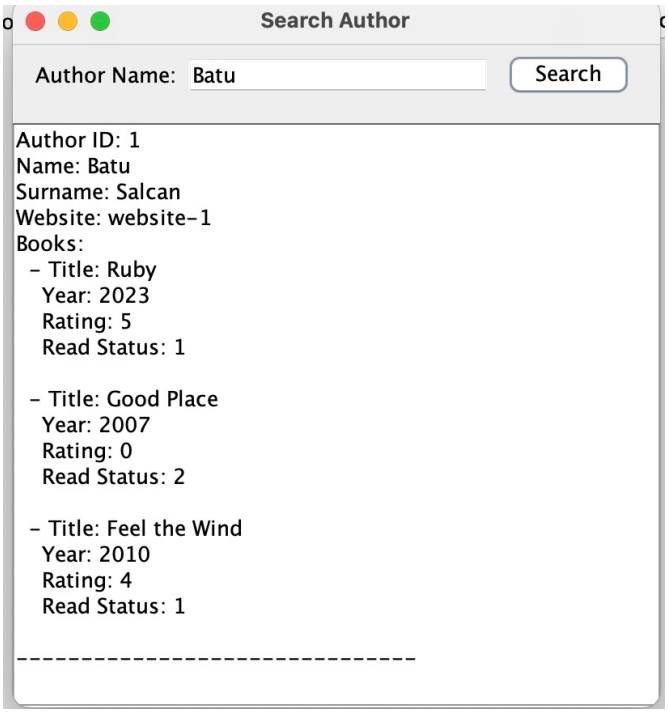
View Book Information With The Book Id: 2



View Book Information With The Book Id: 3

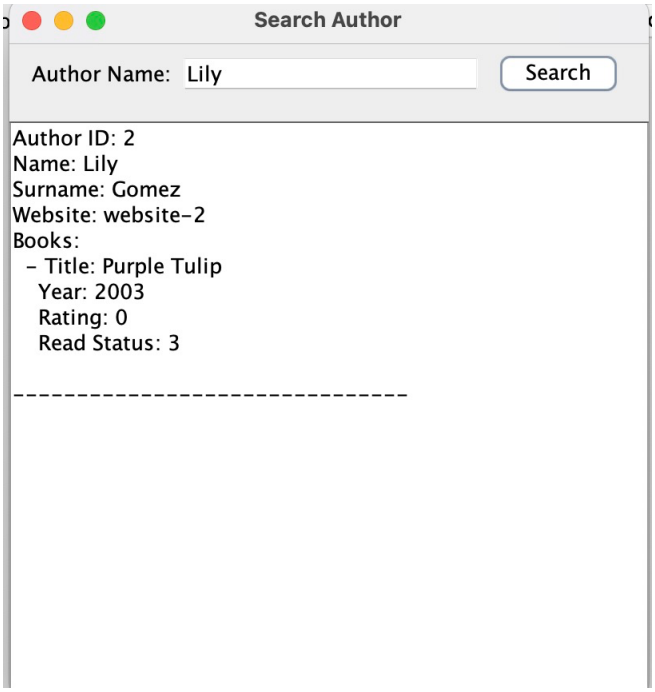


Search Author By Name Example 1 :

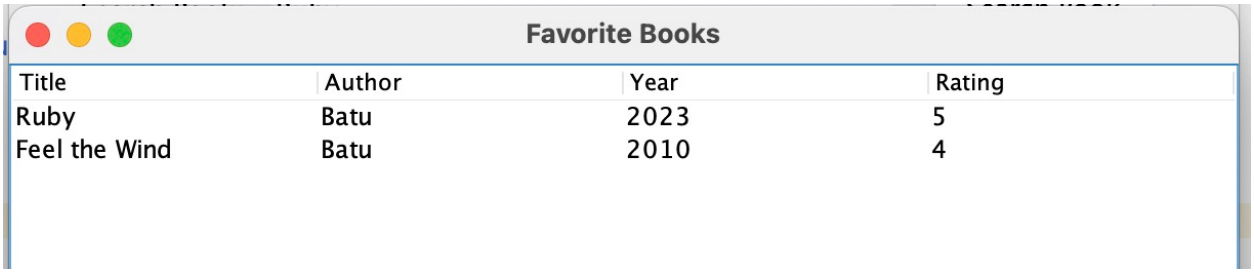


This frame has the author information and the books written by the author we searched.

Search Author By Name Example 2

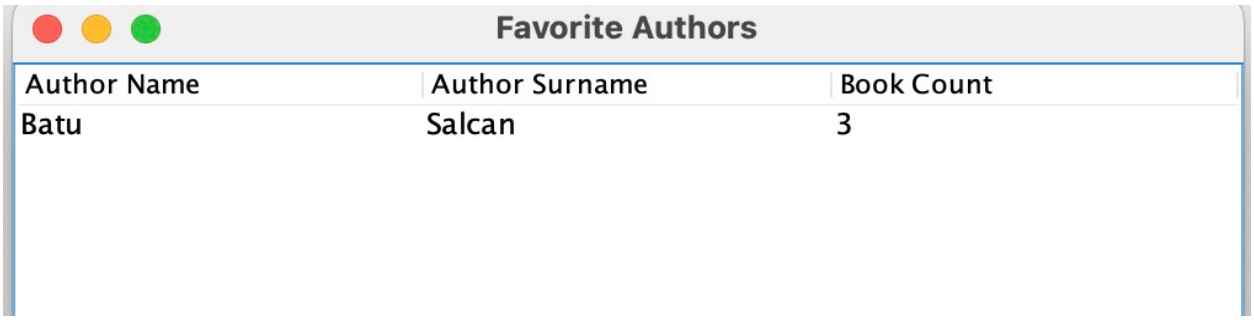


View Favorite Books



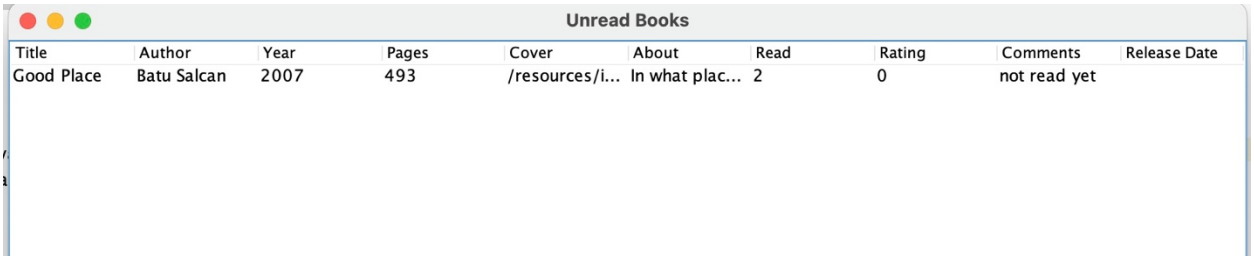
This frame retuns us the books rated as 4 or 5 in the MyLibrary Application.

View Favorite Authors



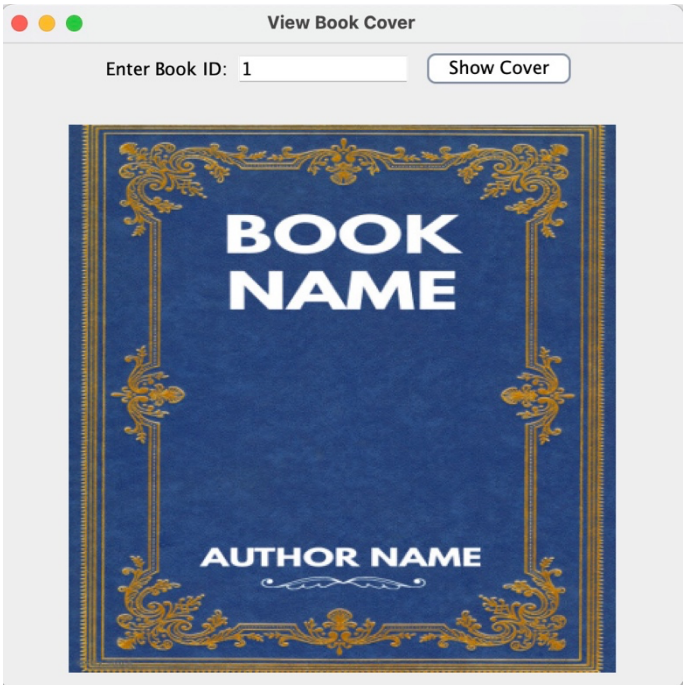
This frame returns us the authors who has at least 3 books in the MyLibrary Application.

View Unread Books

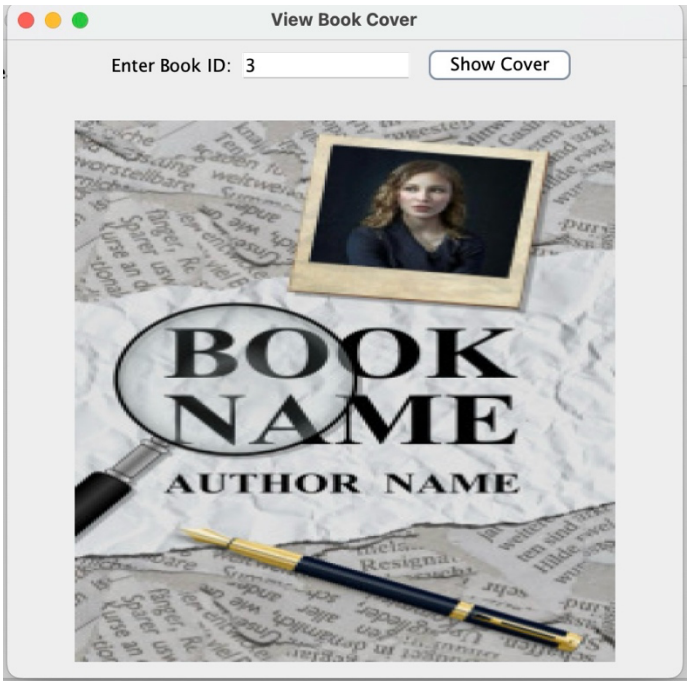


This Frame returns us the books marked as 2 in the MyLibrary Application.

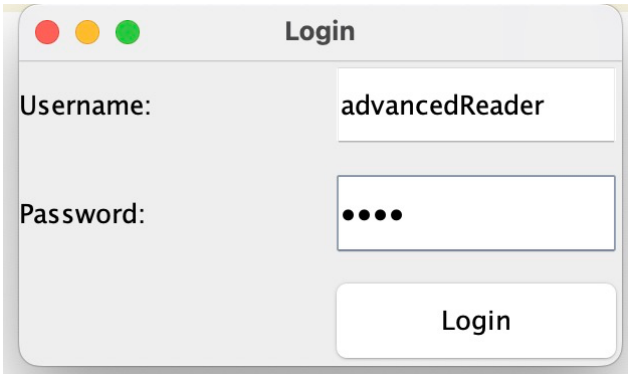
View Book Cover With The Id: 1



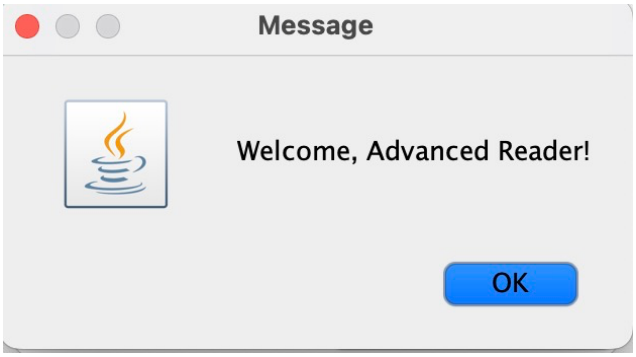
View Book Cover With The Id: 3



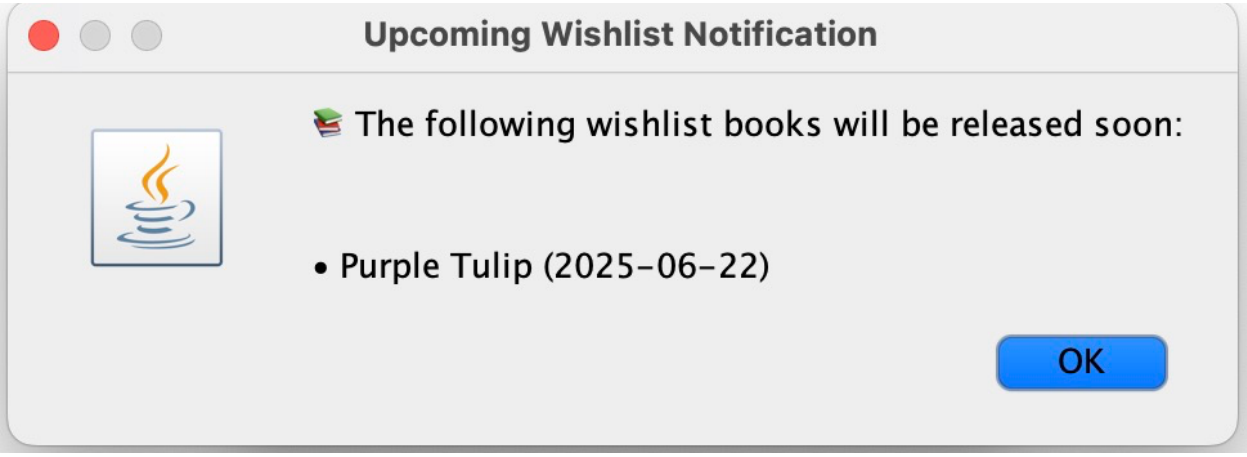
Login Frame For advancedReader (type-1 user)



Login Confirmation

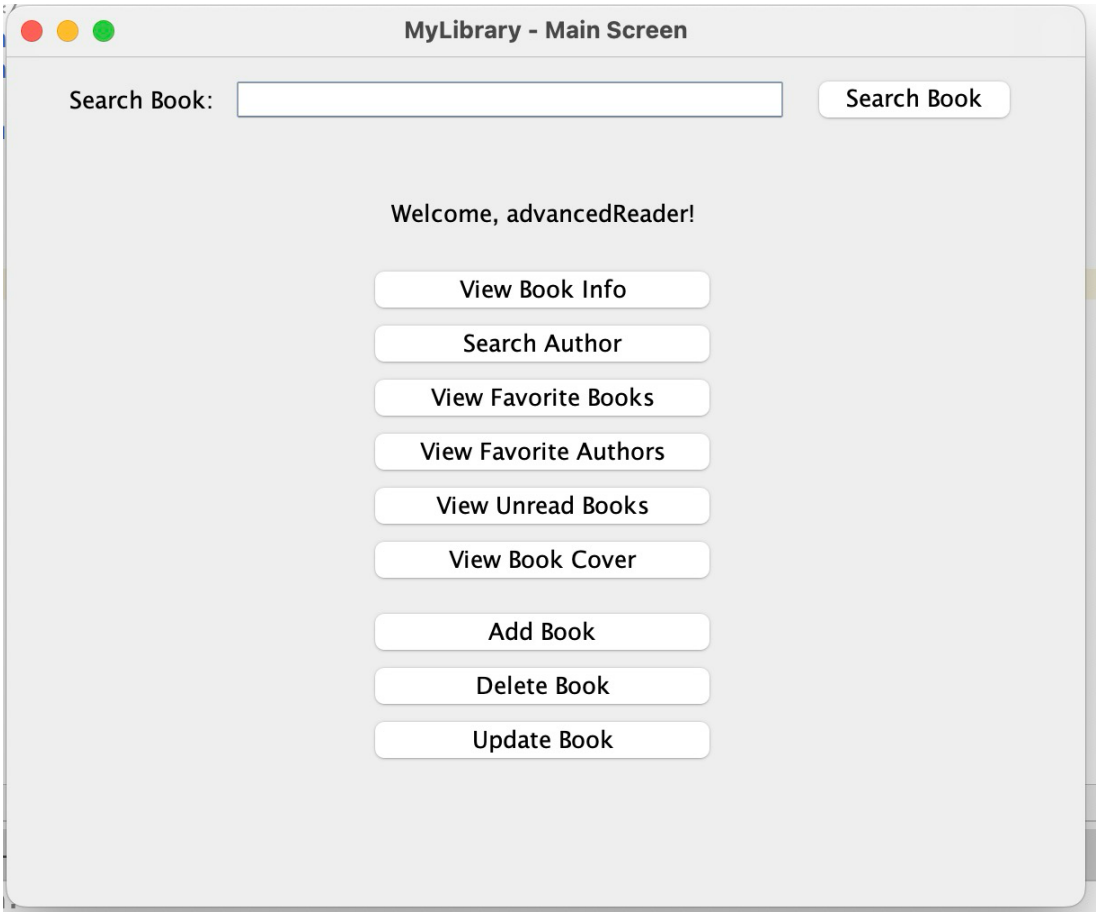


Upcoming Book Notification



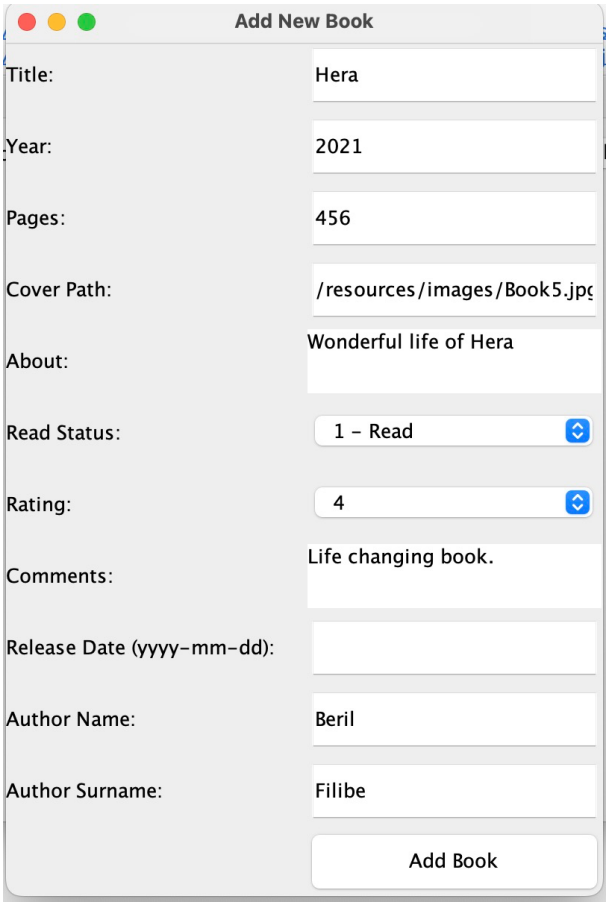
When the advancedrReader logged in to the application he/she recieves this notification if he/she has books marked as 3 and will released in 1 week in the system.

AdvancedReader Mainframe

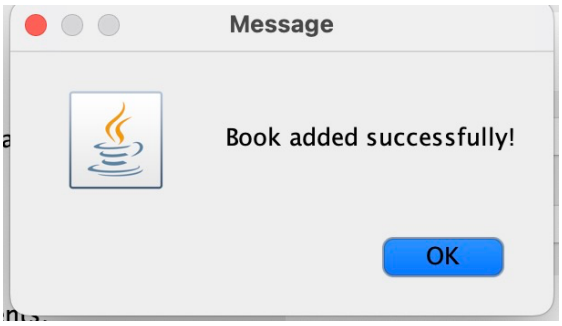


This mainframe is different than basicReader's mainframe. It additionally has "Add Book", "Delete Book" and "Update Book" buttons.

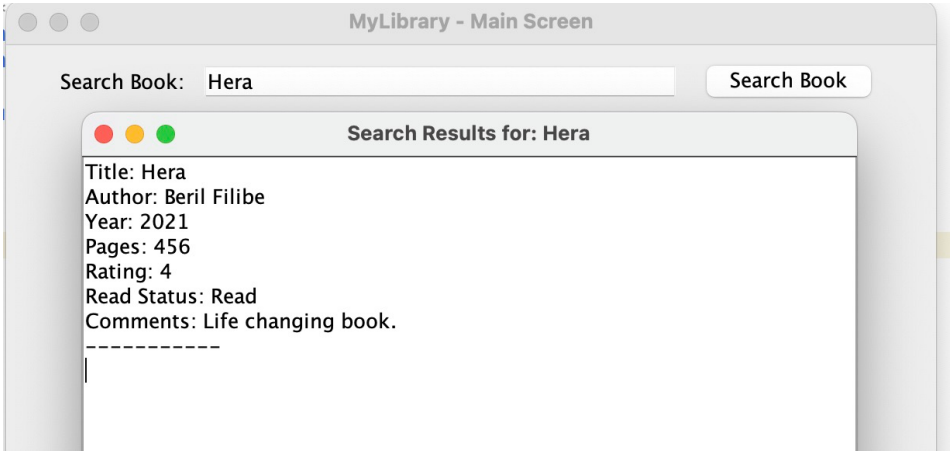
Add Book



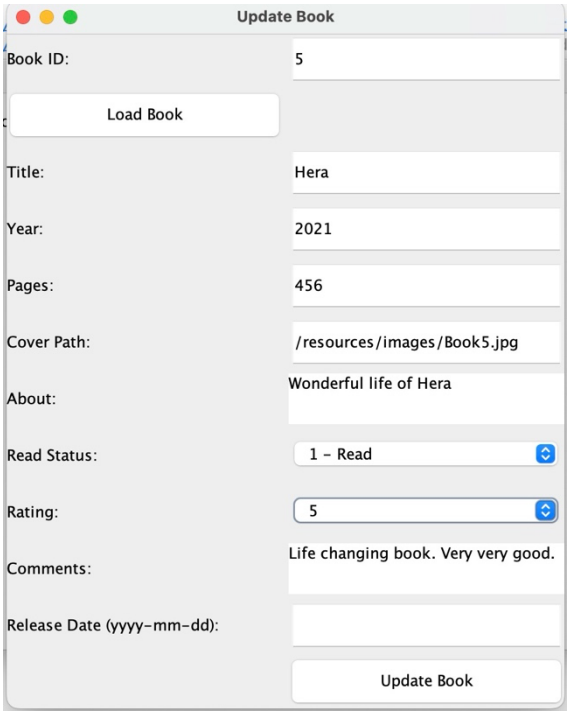
Add Book Confirmation Message



Search Added Book In The Search Bar

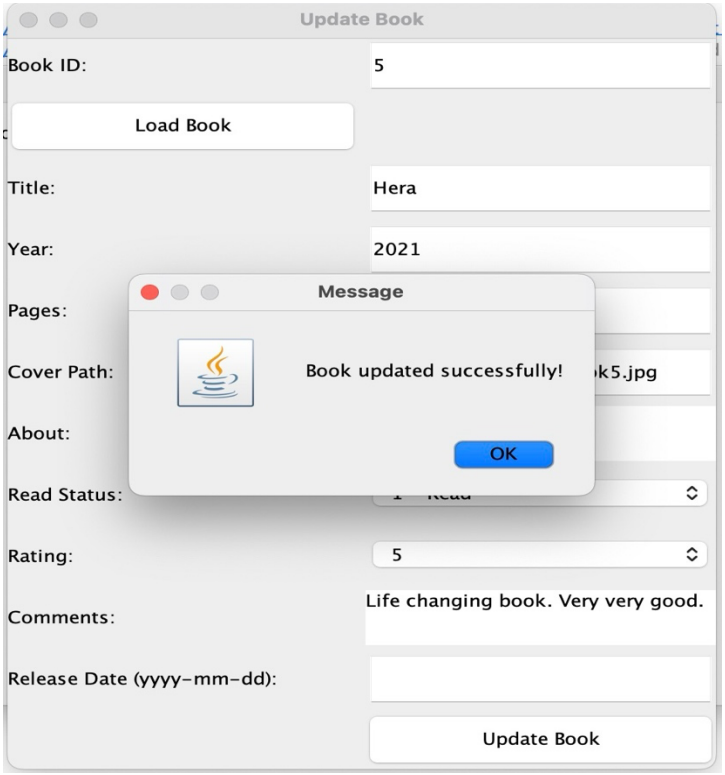


Update Book

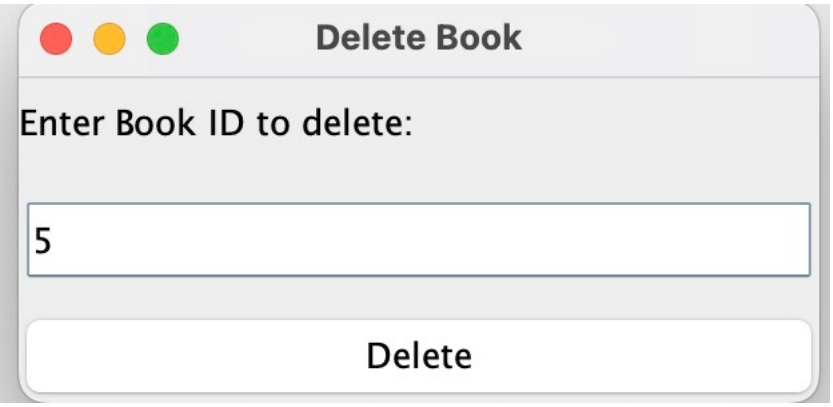


When advancedReader presses the “Update Book” button, he/she writes the bookId then presses “Load Book” button. When the button is pressed, the books information automatically displayed to the advancedReader by the system. Then advancedReader changes some information of the book. In this photo the rating of the book is changed from 4 to 5 and more comment added.

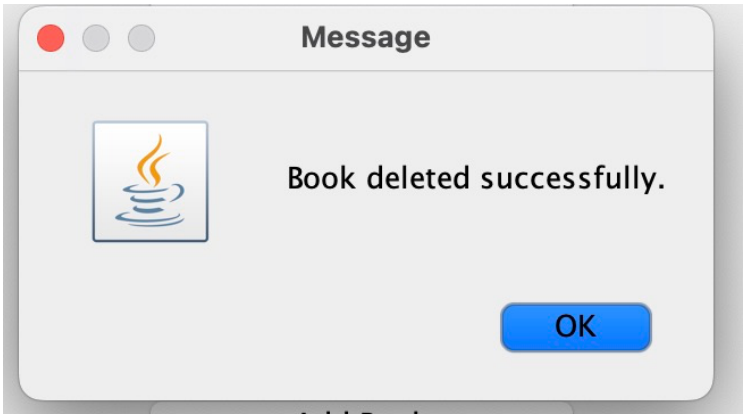
Update Book Confirmation Message



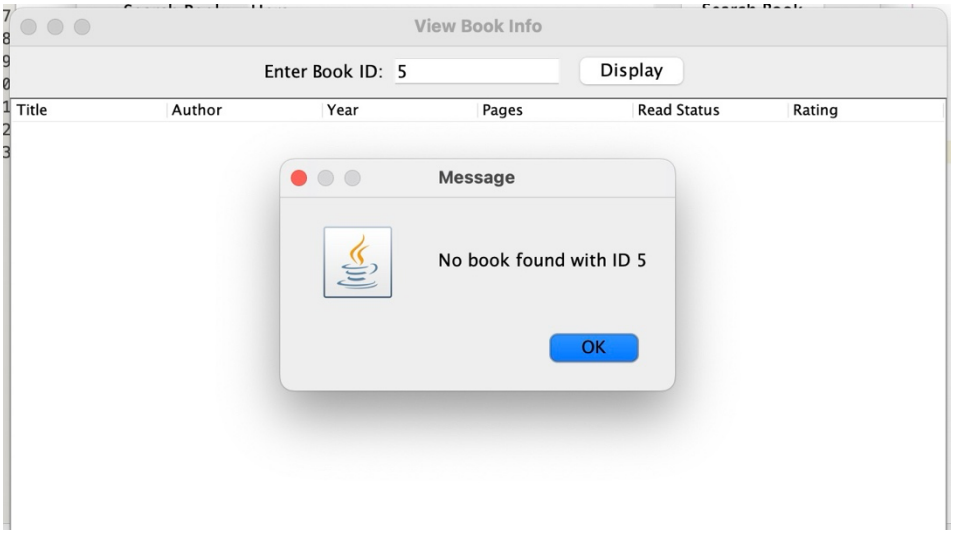
Delete Book



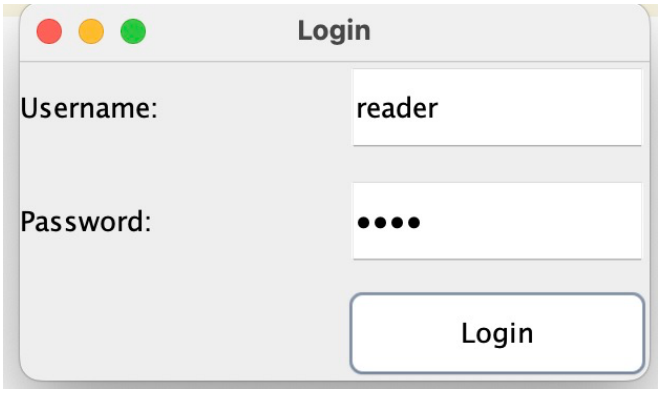
Delete Book Confirmation Message



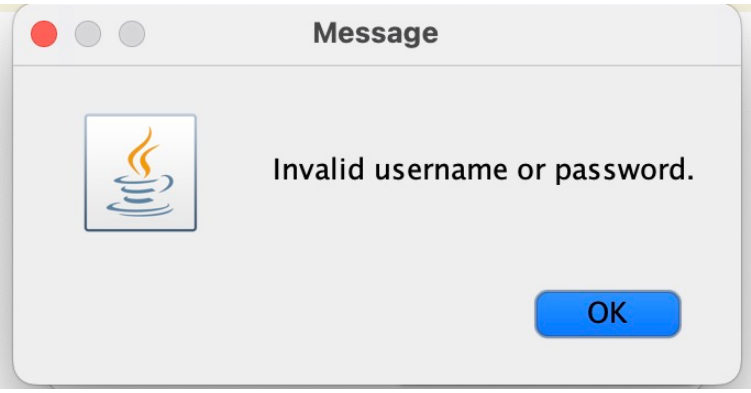
Book Deleted Proof



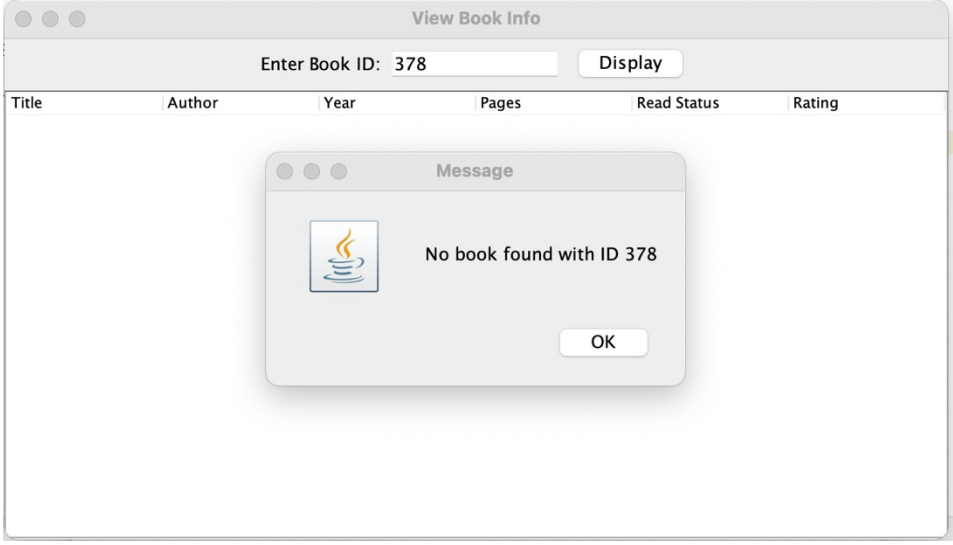
Try To Login With Invalid Username



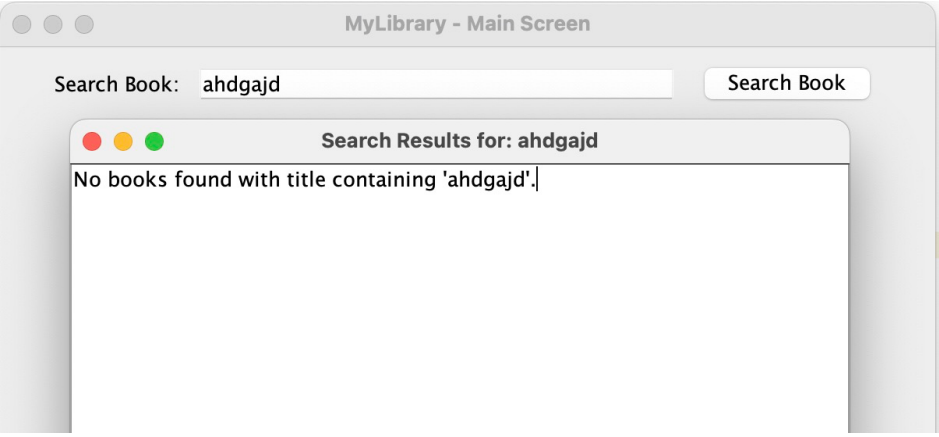
Login Error Message



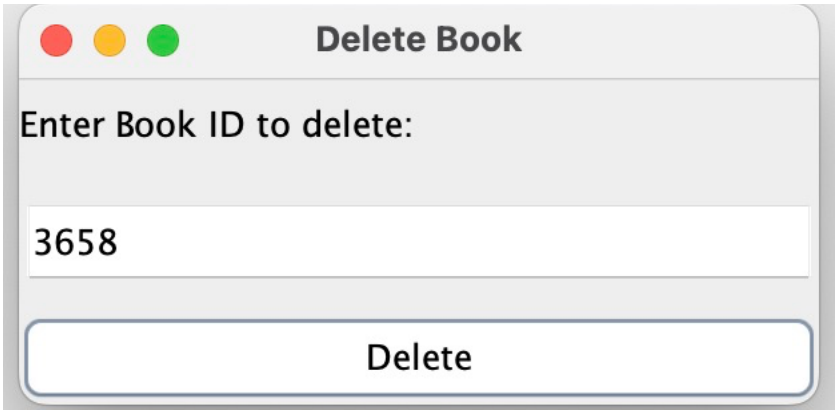
Try To View A Book With AN Invalid Id



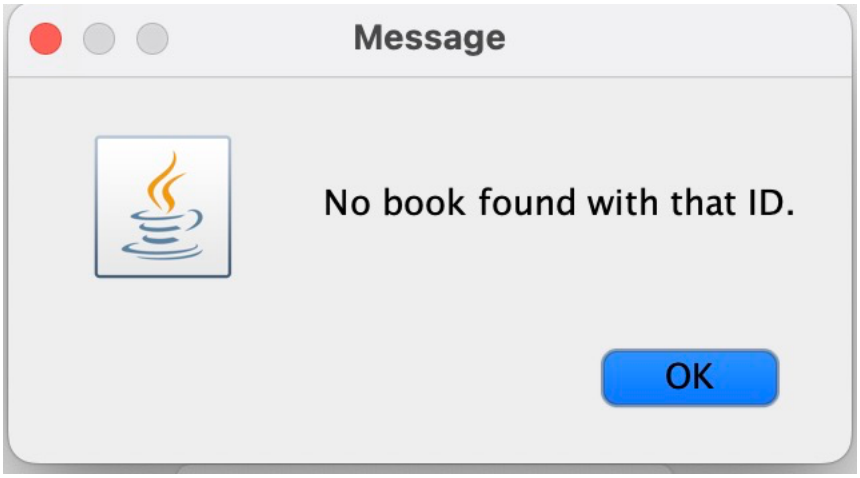
Try To Search Not Existed Book



Try To Delete Not Existing Book With Not Existing Id



Not Existing Book Deletion Error



8 Conclusion and Future Work

In conclusion, the "MyLibrary" desktop application successfully achieves its main goal of helping users manage their personal library of books through an intuitive and user-friendly interface. The application allows users to log in securely, add new books, edit or delete existing ones, search for books or authors, and view book details such as unread status, rating, and comments. The system supports both Basic and Advanced Readers, each with tailored functionality based on user type. The design follows object-oriented principles and integrates seamlessly with a MySQL database for persistent data storage.

Throughout the development process, great attention was given to proper class design, database connectivity, and graphical user interface layout. Each module was implemented and tested to ensure functional accuracy and stability.

As for future improvements, several features can be added to enhance the application's usability and functionality:

Book Recommendation System: Integrate a simple recommendation engine based on reading history and preferences.

Notifications: Improve notification features for upcoming releases or unread books in the wishlist.

GUI Enhancements: Improve the visual design and responsiveness of the interface using modern UI frameworks.

Admin Panel: Add an admin role for managing users and moderating content.

Multi-language Support: Implement internationalization to support different languages for global usability.

Search Filters and Sorting Options: Enhance search by adding filters (e.g., by category, rating, author) and allow sorting (e.g., by title, year).

Data Visualization Dashboard: Include charts and graphs to visualize reading habits, favorite genres, and stats.

PDF or eBook Reader Integration: Allow users to upload and read books directly within the application.

These future developments would provide a richer experience for users and make the MyLibrary system more scalable and adaptable for different user needs.