**BFL – Best Friend Library – Analysis Document**

|  |  |  |
| --- | --- | --- |
| **Students** |  | **Supervisors** |
| Ana-Maria Patriche – 302976  Cristina Aurelia Matei –354776  Elina Grumbina – 355013  Karina Rubahova – 354565  Mara Ioana Statie – 354536 |  | Michael Viuff  Ole Ildsgaard Hougaard |

**Software Engineering**

**Semester 2**

**01.04.2025**

Table of contents

[Summary 3](#_Toc198751118)

[Functional requirements 4](#_Toc198751119)

[Non-functional requirements 4](#_Toc198751120)

[Use case diagram 5](#_Toc198751121)

[Use case description 6](#_Toc198751122)

[**Log in** 6](#_Toc198751123)

[**Manage personal information** 7](#_Toc198751124)

[**Add book** 8](#_Toc198751125)

[**Lend book** 10](#_Toc198751126)

[**Mark book as returned** 11](#_Toc198751127)

[**Search and View book information** 13](#_Toc198751128)

[**Remove Book** 14](#_Toc198751129)

[**Add to waiting list** 15](#_Toc198751130)

[**Remove from waiting list** 17](#_Toc198751131)

[**View Friends' Profiles** 18](#_Toc198751132)

[**Notes - leave book comments** 19](#_Toc198751133)

[**Mark Book as Read** 19](#_Toc198751134)

[**Manage eBook sharing** 20](#_Toc198751135)

[**View activity reports** 21](#_Toc198751136)

[**Edit book** 21](#_Toc198751137)

[Relation between requirements and use cases 23](#_Toc198751138)

[Domain model 24](#_Toc198751139)

# 

# Summary

BFL (Best Friend Library) is a client-server system that will help the GirlyPops friend group centralize and coordinate their individual book collections. The system is interested in dealing with physical/eBook ownership, making borrowing easy, and reducing conflicts by keeping proper records of loans, returns, and the state of the books.

Key goals:

* Centralized control of book availability, ownership, and history of loans.
* Private LAN access to ensure confidentiality among the friend group.
* Simplified eBook sharing via personal cloud links (no hosted storage).

# Functional requirements

1. As a user I want to log into my personal user account when I access the server so that I can track my information in the system.
2. As a user, I want to edit my personal details from my user account, so that I can keep the relevant information up to date.
3. As a user, I want to add a book to the library with details (title, author, ISBN, genre, owner, eBook link, photos, personal reviews) so that I can easily show and borrow it to other users.
4. As a user, I want to edit a book from the library so that I keep the collection updated.
5. As a user, I want to delete a book from my library so that I keep the collection updated.
6. As a user, I want to view a book’s details, status (available/borrowed) and its loan history (borrower, lender, dates, condition notes) so that I can learn all I want about it.
7. As a user, I want to record that a book I own was borrowed by another user so that I keep track of my books’ location.
8. As a user, I want to confirm a book I own was returned so that a book cannot be marked as returned to me without my knowledge.
9. As a user, I want to add myself to a waiting list for a book so that I can borrow it.
10. As a user, I want to remove myself from a waiting list that I’ve added myself to so that I can avoid unnecessary delays if I changed my mind about waiting to borrow a book.
11. As a user, I want to see reports of recent activity in the app, so that I know what I and other users are doing.
12. As a user, I want to view friends’ profiles to see their owned books, read books and borrowed books lists, and active loans, so that we can talk about them.
13. As a user, I want to leave comments (reviews, impressions) on a book I read, so that I can share my opinion with other users.
14. As a user, I want to mark a book as read so that other users can see I read it.
15. As a user, I want to download eBooks from other users’ libraries so that I can store them locally for offline use.
16. As a user, I want to share eBook links so that other users can store them on their devices for offline use.
17. As a user, I want to search a book in the database, so that I can see which users have copies of it.

# Non-functional requirements

1. The system must be coded in Java and JavaFx.
2. The system must store the book and user data in databases.

# Use case diagram

A screenshot of a diagram

AI-generated content may be incorrect.

Figure Use case diagram

# Use case description

|  |  |
| --- | --- |
| **Use case** | **Log in** |
| **Summary** | User can log into their account. |
| **Actor** | User |
| **Precondition** | Client app installed  Network connectivity available  User has an account and has set up their password and unique username |
| **Postcondition** | Success: User is authenticated and connected; main UI is displayed  Failure: User remains on Log in window with error feedback |
| **Base sequence** | * + - 1. User launches the client application       2. System connects client application to server [ALT3]       3. Client displays Login window with * Username field * Password field * “Log in” button * “Cancel” button   + - 1. User enters a valid username and password       2. User selects “Log in”       3. Client validates the input fields [ALT1]       4. Client sends a Login request with the credentials to the server       5. Server validates credentials against the user database [ALT2,3]       6. Server returns a ‘success’ response       7. Client displays a ‘success’ message and displays the main application interface |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step, user may cancel and end use case  [ALT1] At step 6, if username or password fields are blank   * System displays “Please input your username and password” * Return to step 4   [ALT2] At step 8, if credentials don’t match   * Server responds with “Invalid credentials” * Client application displays the error * Return to step 4   [ALT3] At step 2 or 7, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | 1 |

|  |  |
| --- | --- |
| **Use case** | **Manage personal information** |
| **Summary** | User can modify their personal information |
| **Actor** | User |
| **Precondition** | Network connectivity available  User exists in the BFL Library user database  User is authenticated and connected to the server  Main application interface is displayed |
| **Postcondition** | Success: User’s information in the database is updated. Client displays the updated information.  Failure: No changes are saved; User remains on User information edit page with error feedback |
| **Base sequence** | User connects to the BFL server with their credentials [ALT3]  System displays the main application interface homepage with   * “My library” button * “My account” button * “Find book” button * “Find user” button * “Notifications log” text field   User selects “My account” [ALT3]  System displays the user’s information page with personal information fields (avatar, username, birthday, etc) and   * “Return” button (arrow icon) * “Edit” button (gear icon) * “Read books” button * “Lent books” button * “My library” button * “Borrowed books” button   User selects “Edit” (gear) button  System displays Edit User Information form, populated with the current values for:   * Username * Name * Birthday * Email * Telephone number * Address * More info * Add image   Empty fields for:   * Current password (required) * New password (optional)   And   * “Return” (arrow icon) button * “Save” button   User modifies the information in one or more fields, fills “Current password” field (optionally enters a new password in the “New password” field)  User clicks “Submit”  System validates the input fields [ALT1]  System sends a User information change request to the server with the information [ALT3]  System validates password against the user database[ALT2]  System validates changes  Server updates the User information in the user database and sends a ‘success’ response  System displays a “success” message  User returns to the User’s information page |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step between 1 and 8, user may cancel and end use case  [ALT1] At step 9, if the “Current Password” or other required fields are empty   * System displays an error message “Please fill all required\* fields” * Return to step 7   [ALT2] At step 11 or 12, if the changes or the current password are not validated   * Server responds with relevant error message (“Password incorrect”, “Username must not contain spaces”, etc) * User corrects errors and re-enters current password (return to step 6)   [ALT3] At step 1, 3 or 10, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | 2 |

|  |  |
| --- | --- |
| **Use case** | **Add book** |
| **Summary** | User can add books to the library. |
| **Actor** | User. |
| **Precondition** | Network connectivity available  User is authenticated and connected to the BFL server  Main application interface is displayed |
| **Postcondition** | Success: Book data is saved, updated, or removed in the central system and shown for all users.  Failure: No changes are saved; User remains on Add/Edit book page with error feedback |
| **Base sequence** | User selects “My library”  Client sends a request to display the user’s owned books list to the server [ALT3]  Server returns the user’s personal library table  Client displays the user’s personal library list and   * “Return” (arrow icon) button * “Add new book” button * “Remove book” button   User clicks “Add new book” button  Client displays Add/Edit Book Information form with book information empty fields (title, author, genre, etc) and   * “Return” (arrow icon) button * “Add photo” button * “Submit” button   User clicks “Submit” button  System validates the input fields [ALT1]  System sends an AddBook request to the server with the information [ALT3]  System validates changes [ALT2]  Server adds the book to the BFL library and updates the library information and sends a ‘success’ response  System displays a “success” message  Client returns to User Library page |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step between 1 and 7, user may cancel and end use case.  [ALT1] At step 7, if any of the required fields are empty   * System displays an error message “Please fill all required\* fields” * Return to step 6   [ALT2] At step 10, if the changes are not validated   * Server responds with relevant error message (“ISBN must not contain spaces”, etc) * User corrects errors and re-enters current password (return to step 7)   [ALT3] At step 2 or 9, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | Admin rights are not required for managing personal book entries. 3 |

|  |  |
| --- | --- |
| **Use case** | **Lend book** |
| **Summary** | A user lends a book to another user. The system records the loan and updates the book’s status. |
| **Actor** | Book Owner (User). |
| **Precondition** | Network connectivity available  Active user is authenticated and connected to the BFL server  The borrower is a user of the BFL system and has previously added themselves to the Waiting List for the book  The book exists in the system, belongs to the active user and is marked as "available"  Main application interface is displayed |
| **Postcondition** | Success: The book’s status changes to "borrowed to [borrower user name]"; a loan record is created with borrower details and loan date, recoded in the book’s History log and displayed in the Notifications log.  Failure: No changes are saved; User remains on the book’s information page |
| **Base sequence** | 1. User selects „Find book” [ALT3] 2. Client displays the Find Book Page with search and filter options, a partial BFL library list and  * “Return” button (arrow icon) * „Search” button (magnifier icon)  1. User selects search criteria for the book by owner/title/author/ISBN. 2. User clicks „Search” (magnifier icon) button [ALT1] 3. Client sends search request to the server library database [ALT3] 4. Server returns a list of books fitting the search criteria 5. Client displays the search results list 6. User selects the book they intend to lend 7. Client displays Book Information page with relevant Book information fields (photo, title, author, genre, etc) and  * “Return” button (arrow icon) * “Edit” button (gear icon) * “Waiting list” button * “History” button * “Lend book” button * “Add note” button * “Book was returned” button * “Read” toggle  1. User clicks „Lend book” button 2. Client displays the „Waiting List” page and prompts the user to select the borrowing user from the list or names 3. User selects the borrowing user and confirms the choice bly clicking the „Confirm” button 4. Client sends a Borrowing request with the required information [ALT3] 5. Server validates the information [ALT2] 6. Server updates the relevant book fields in the BFL Library database, logs the activity in the Notifications log and sends a ‚success’ response 7. Client displays a ‚success’ message 8. Client returns to Book Information page |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step between 1 and 12, user may cancel and end use case.  [ALT1] At step 4, if all the search fields are empty   * System displays an error message “Please select a search criteria” * Return to step 3   [ALT2] At step 14, if the information is not validated   * Server responds with relevant error message (“This book cannot be lent”, “This book is already borrowed”, “User is already borrowing this book”, etc.) * User corrects errors and tries again (return to step 10)   [ALT3] At step 1, 5 or 13, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | .Admin rights are not required for managing personal book entries. 7 |

|  |  |
| --- | --- |
| **Use case** | **Mark book as returned** |
| **Summary** | Book owner confirms the return of a borrowed book. The system updates the book’s status and logs any comments about the book's condition. |
| **Actor** | Book Owner (User) |
| **Precondition** | Network connectivity is available  User is authenticated and connected to the BFL server.  The book exists in the system, belongs to the active user and has a „borrowed” status  The active user has found the book and the book’s Information page is displayed |
| **Postcondition** | - Success: Book status is updated to “available”. Loan record is updated to include the return date and any optional comments.  - Failure: No changes are made; user remains on the current page with appropriate error message. |
| **Base sequence** | 1. Client displays Book Information page with relevant Book information fields (photo, title, author, genre, etc) and  * “Return” button (arrow icon) * “Edit” button (gear icon) * “Waiting list” button * “History” button * “Lend book” button * “Add note” button * “Book was returned” button * “Read” toggle  1. User clicks „Book was returned” button 2. Client displays a prompt with an editable text field, a “return” (arrow icon) button and a “Confirm” button asking for optional comments on the book's condition. 3. User enters comments (if any) and clicks the “Confirm” button. [ALT1] 4. Client sends a Returned book update request to the server with the relevant information [ALT3] 5. Server validates the information [ALT2] 6. Server updates the corresponding loan record and the BFL library book database with the information (return date, comments, status, etc). 7. Server sends a ‚success’ response to the client [ALT3] 8. Client displays a ‚success’ message 9. Client returns to ‚Book information’ page |
| **Alternate sequence (branch or**  **exception)** | \*[ALT0] At any step between 1 and 4, the user may cancel the process.  - No changes are saved; user returns to the previous page.  [ALT1] At step 4, if the book is returned in a damaged state.  - User enters a comment describing the damage.  - Proceed to step 5.  [ALT2] At step 2, if information is not validated by the server, a descriptive error message is displayed by the client („Book is already available”, „You are not the owner of this book”, etc)  [ALT3] At step 5 or 8, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | 8 |

|  |  |
| --- | --- |
| **Use case** | **Search and View book information** |
| **Summary** | User can view book information (own or others’), including a complete loan history for a selected book (borrower details, loan dates, and condition notes). |
| **Actor** | User. |
| **Precondition** | User is authenticated and connected to the BFL system.  The selected book exists in the database.  Main application interface is displayed |
| **Postcondition** | Success: System displays the book’s loan history in a readable, chronological format.  Failure: No data is shown; user remains on the current page with an appropriate error message. |
| **Base sequence** | 1. User selects „Find book” [ALT3] 2. System displays the Find Book Page with search and filter options, a partial BFL library list and  * “Return” button (arrow icon) * „Search” button (magnifier icon)  1. User selects search criteria for the book by owner/title/author/ISBN. 2. User clicks „Search” (magnifier icon) button [ALT1] 3. Client sends search request to the server library database [ALT3] 4. Server returns a list of books fitting the search criteria 5. Client displays the search results list 6. User selects the book they intend to view [ALT2] 7. Client displays Book Information page with relevant Book information fields (photo, title, author, genre, etc) and  * “Return” button (arrow icon) * “Waiting list” button * “Add note” button * “History” button * “Read” toggle   If user owns the book   * “Edit” button (gear icon) * “Lend book” button * “Book was returned” button   If user does not own the book   * “Download link” button (active if a link is available) * “Add me to waiting list” button  1. User clicks “History” or “Waiting list” button [ALT2] 2. Client displays the “Log history” or the “Waiting List” page  * Log history is displayed in chronological order, showing: * Borrower * Borrow and return dates * Notes about book condition (if any) * „Return” (arrow icon) button * Waiting list is displayed in chronological order, showing: * User name * Date (when the user was added to waiting list) * “Remove me from this waiting list” button * “Return” (arrow icon) button |
| **Alternate sequence (branch or**  **exception)** | \*[ALT0] At any step, the user may cancel and exit the use case.   * No changes are saved; user returns to the previous page.   [ALT1] At step 4, if all the search fields are empty   * System displays an error message “Please select at least one search criteria” * Return to step 3   [ALT2] At step 8 or 11, if the selected book information does not exist or has been deleted:   * System displays: “Information not found.” * User may retry or return to step 8 and select another book to view   [ALT3] At step 1 or 5, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | 6, 17 |

|  |  |
| --- | --- |
| **Use case** | **Remove Book** |
| **Summary** | The user (book owner) deletes it from their personal library to keep their collection up to date. |
| **Actor** | User (Book Owner) |
| **Precondition** | Network connection is available  User is authenticated and connected to the BFL system.  The book exists in the system and has been found  The user is the owner of the selected book. |
| **Postcondition** | Success: The book is removed from the system database and the owner’s library. Any related data (e.g., waiting lists, loan records) is updated accordingly.  Failure: No changes are made. |
| **Base sequence** | 1. Client displays the main application interface. 2. User navigates to “My Library” section. 3. Client displays a list of books owned by the user, a “Return” (arrow icon) button, an “Add New Book” button, a “View Book information” and a “Remove” button 4. User selects a book to remove. 5. User clicks the “Remove” button. 6. Client displays a confirmation dialog. 7. User confirms the action. 8. Client sends a Remove book request to the server [ALT2] 9. Server validates the command [ALT1] 10. Server deletes the book from the database and the user’s library. 11. Server updates related data (e.g., clears waiting lists, loan history). 12. Server sends a ‘success’ message indicating successful removal. [ALT2] 13. Client displays ‘success’ message and returns to “My library” page |
| **Alternate sequence (branch or**  **exception)** | \*[ALT0] At any step, the user may cancel and exit the use case.   * No changes are saved; user returns to the previous page.   [ALT1] At step 9, if the server does not validate the request   * Server sends a relevant ‚error’ message * Client displays the ‚error’ message * User may try again (return to step 4)   [ALT2] At step 8 or 12, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | 5 |

|  |  |
| --- | --- |
| **Use case** | **Add to waiting list** |
| **Summary** | A user adds themselves to the waiting list for a book they want to borrow. |
| **Actor** | User |
| **Precondition** | Network connection is available  User is authenticated and connected to the BFL system.  User does not own the book.  The desired book information is displayed in the interface (has been found)  The selected book is currently borrowed or available |
| **Postcondition** | Success: The user is added to the waiting list for the selected book.  Failure: No change is made; user is notified of the reason. |
| **Base sequence** | 1. System displays the Book information page, showing:  * Book information (title, author, cover, status, etc) * “Read” toggle * “Return” (Arrow icon) button * “Waiting list” button * “History” button * “Add note” button * “Add me to waiting list” button  1. User clicks the “Add me to waiting list” button. 2. Client prompts the user for confirmation 3. User confirms the action 4. Client checks if the user is already on the list. [ALT1] 5. Client sends “Add to Waiting List” request to server [ALT3] 6. Server validates the request [ALT2] 7. Server adds the user to the waiting list for the selected book and updates the relevant field in the database 8. Server returns a ‘success’ message [ALT3] 9. Client displays the ‘success’ message and returns to Book information page |
| **Alternate sequence (branch or**  **exception)** | \*[ALT0] At any step between 1 and 2, the user may cancel and exit the use case.   * No changes are saved; user returns to the previous page.   [ALT1] At step 5, if the user is already on the waiting list:   * System displays a popup message: “Already on the list.” * User is returned to the book detail page   [ALT2] At step 7, if the server does not validate the request   * Server sends a relevant ‚error’ message * Client displays the ‚error’ message * User may try again (return to step 2)   [ALT3] At step 6 or 9, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later” * User can close the application window |
| **Note** | Users may be on multiple waiting lists simultaneously. The system automatically notifies users when their turn comes up. 9 |

|  |  |
| --- | --- |
| **Use case** | **Remove from waiting list** |
| **Summary** | The user removes themselves from the waiting list of a book they no longer wish to borrow. |
| **Actor** | User |
| **Precondition** | User is authenticated and connected to the BFL system.  User is currently on the waiting list for the selected book.  The desired book exists in the BFL Library database, does not belong to the user  The user has found the book and the book’s Information page is displayed |
| **Postcondition** | Success: User is removed from the book’s waiting list.  Failure: No change is made; user is notified of the reason |
| **Base sequence** | 1. System displays the Book information page, showing:  * Book information (title, author, cover, status, etc) * “Read” toggle * “Return” (Arrow icon) button * “Waiting list” button * “History” button * “Add note” button * “Add me to waiting list” button  1. User clicks the “Waiting list” button. 2. Client displays the “Waiting List” page with a chronological list of users and the date they added themselves to the list, a “Return” (arrow icon) button and a “Remove me from this waiting list” button 3. User clicks “Remove me from this waiting list” button 4. Client prompts the user for confirmation 5. User confirms the action 6. Client checks if the user is on the list. [ALT1] 7. Client sends “Remove from Waiting List” request to server [ALT3] 8. Server validates the request [ALT2] 9. Server removes the user from the waiting list for the selected book and updates the relevant field in the database 10. Server returns a ‘success’ message [ALT3] 11. Client displays the ‘success’ message and returns to Book information page |
| **Alternate sequence (branch or**  **exception)** | \*[ALT0] At any step, the user may cancel and exit the use case.   * No changes are made; user returns to the previous page.   [ALT1] At step 7, if the user is not on the waiting list:   * „You are not on this list” error message is displayed * User is returned to the Waiting List page   [ALT2] At step 9, if the server does not validate the request   * Server sends a relevant ‚error’ message * Client displays the ‚error’ message * User may try again (return to step 4)   [ALT3] At step 8 or 11, if the connection fails   * Client application displays “Unable to reach server. Please check your connection or try again later”   User can close the application window |
| **Note** | 10 |

|  |  |
| --- | --- |
| **Use case** | **View Friends' Profiles** |
| **Summary** | User views another user's profile including their book collections and reading activity. |
| **Actor** | User |
| **Precondition** | - User is authenticated  - Friend exists in system |
| **Postcondition** | 1.Friend's profile displayed 2.Error message shown |
| **Base sequence** | 1. User selects "Find user" 2. Searches for friend 3. Selects friend from results 4. System displays profile with:  - user avatar, user information + more info - Owned books – My Library - Reading history – Read List - Currently borrowed books – Lent books list  - Borrowed books list |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step, user may cancel and end use case  [ALT1]: Friend not found [ALT2]: Connection issues |
| **Note** | 12 |

|  |  |
| --- | --- |
| **Use case** | **Notes - leave book comments** |
| **Summary** | User adds reviews or impressions about a book they've read. |
| **Actor** | User |
| **Precondition** | - User is authenticated - Book exists in system - User has marked book as read |
| **Postcondition** | 1. Comment added to book's record 2. Comment not saved |
| **Base sequence** | 1. User views book information  2. Clicks "Add note"  3. Enters comment  4. Submits comment  5. System saves comment |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step, user may cancel and end use case  [ALT1]: Empty comment - prompt to enter text [ALT2]: Server error during save |
| **Note** | 13 |

|  |  |
| --- | --- |
| **Use case** | **Mark Book as Read** |
| **Summary** | User marks a book as read to share this status with others. |
| **Actor** | User |
| **Precondition** | - User is authenticated - Book exists in system  - User has access to the book (owned/borrowed) |
| **Postcondition** | 1. Book marked as read in user's profile 2. Book is added to the user’s "Read Books" list 3. Status not updated |
| **Base sequence** | 1. User views book information 2. Toggles "Read" switch 3. System updates reading status 4. Confirmation displayed |
| **Alternate sequence (branch or**  **exception)** | [\*ALT0] At any step, user may cancel and end use case  [ALT1]: Connection fails - retry option |
| **Note** | 14 |

|  |  |
| --- | --- |
| **Use case** | **Manage eBook sharing** |
| **Summary** | User views eBook details and either downloads the PDF or manages sharing options. |
| **Actor** | User |
| **Precondition** | User is logged in  - eBook exists in the system  - Book information window is open |
| **Postcondition** | eBook is downloaded or share link is generated |
| **Base sequence** | 1. From Book Information window, user sees:  "Get PDF" button (if eBook)  2. For eBook download:  a. User clicks "Get PDF"  b. System shows download confirmation dialog  c. User confirms  d. System initiates PDF download  e. Progress bar appears during download   3. For sharing:   a. Owner clicks "Share Link"   b. System generates unique URL   c. Shows share options:   - Copy link   - Send via email   d. Owner selects option |
| **Alternate sequence (branch or**  **exception)** | [ALT1] Download fails:  - Shows "Download failed - try again"   - Offers "Retry" button   [ALT2] No PDF available:   - "Get PDF" button appears disabled  - Shows tooltip "No eBook available" on hover |
| **Note** | 15,16 |

|  |  |
| --- | --- |
| **Use case** | **View activity reports** |
| **Summary** | User sees recent system activities like new books, loans, returns, and comments from friends. |
| **Actor** | User |
| **Precondition** | - User is authenticated  - At least one friend exists in system |
| **Postcondition** | Timeline report displayed with newest first |
| **Base sequence** | 1. User clicks "Activity Feed" tab 2. System queries Log History table  3. System renders each entry with:  - Actor (user who performed action)  - Action type (icon + text: "added book", "left review", etc.)  - Timestamp - Linked item (clickable book title) 4. User can scroll through feed |
| **Alternate sequence (branch or**  **exception)** | [ALT1] No activities:  - Shows "No recent activity" placeholder  [ALT2] New notifications:  - Highlights unread entries in blue |
| **Note** | 11 |

|  |  |
| --- | --- |
| **Use case** | **Edit book** |
| **Summary** | Book owner modifies details of their book in the system.  "Edit" button appears only to book owners |
| **Actor** | User |
| **Precondition** | - User is logged in and owner of the book  - Book Information window is open  - "Edit" button is visible |
| **Postcondition** | Book details are updated across the system |
| **Base sequence** | 1. From Book Information window, owner clicks "Edit"  2. System opens Edit Book overlay showing:  - Editable fields with current values:  \* Title  \* Author  \* ISBN  \* Genre (dropdown)  \* eBook link (if eBook)  \* Type, Year, Language  \* Publishing year \* Cover image (with "Change" button)  - Action buttons:   \* "Save Changes"   \* "Cancel"   \* "Delete Book"  3. Owner makes changes to fields  4. Owner clicks "Save Changes"  5. System validates inputs [ALT1]  6. System confirms "Book updated successfully"  7. Returns to updated Book Information window |
| **Alternate sequence (branch or**  **exception)** | [ALT1] Validation fails:  - Highlights invalid fields in red  - Shows tooltips:  \* "Title cannot be empty"  \* "Invalid ISBN format"  [ALT2] Owner cancels:  - Discards changes without confirmation  - Returns to Book Information window  [ALT3] Owner deletes book:  - Triggers Remove Book use case |
| **Note** | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Relation between requirements and use cases Use cases | Covered requirements |  |  |  |  |
| Log in | 1 |  |  |  |  |
| Manage book information | 2 |  |  |  |  |
| Add book | 3 |  |  |  |  |
| Edit book | 4 |  |  |  |  |
| Remove Book | 5 |  |  |  |  |
| Search and View book information | 6,17 |  |  |  |  |
| Lend book | 7 |  |  |  |  |
| Mark book as returned | 8 |  |  |  |  |
| Manage eBook sharing | 9,10 |  |  |  |  |
| View activity reports | 11 |  |  |  |  |
| View Friends' Profiles | 12 |  |  |  |  |
| Notes - leave book comments | 13 |  |  |  |  |
| Mark Book as Read | 14 |  |  |  |  |
| Manage eBook sharing | 15,16 |  |  |  |  |
|  |  |  |  |  |  |

# Domain model

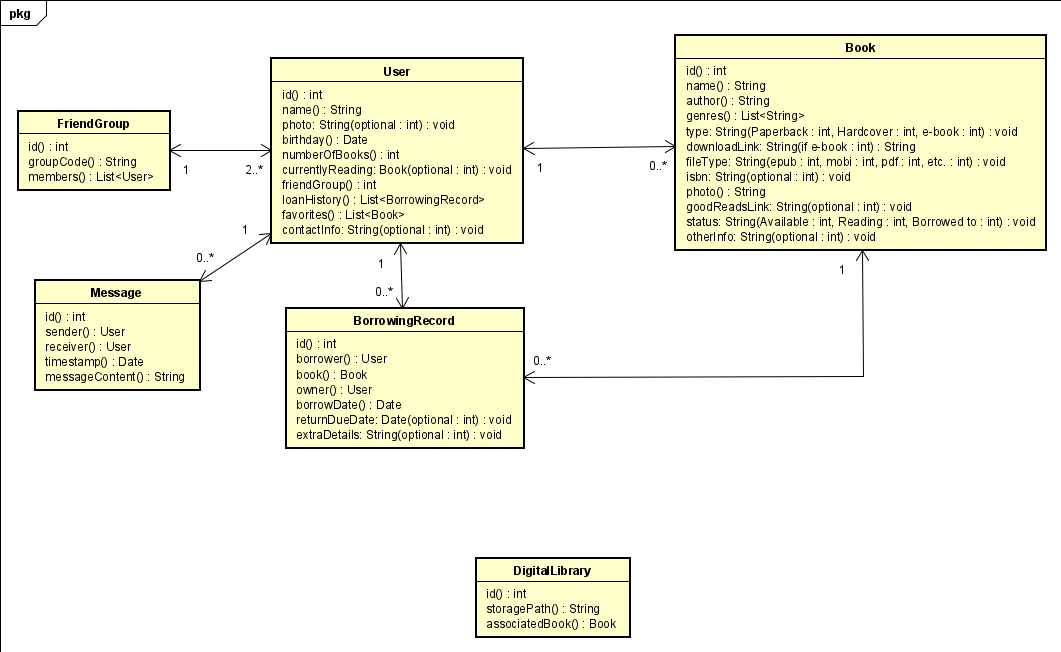


Figure 2 Domain Model