Functional requirements

1. User management

1.1 Registration & Authentication

- The system shall allow a new user to register using a email, username, and password.
- The system shall allow registered users to log in using their username and password
- The system shall allow users to log out.

1.2 Profile Management

- The system shall allow a logged-in user to view their profile.
- The system shall allow a logged-in user to update their profile information.
- The system shall allow a user to delete their account.

- The system must validate the email and password
- The system must maintain session management to track logged-in users.

2. Book listings

2.1 CRUD Operations

- The system shall allow a logged-in user to create a book listing with the following fields: title, author, year of publication, genre, condition, and transaction type (Sell, Rent, Exchange, Giveaway).
- The system shall allow the owner of a book listing to update its details.
- The system shall allow the owner of a book listing to delete the listing.
- The system shall display a book listing's availability status (Available or Unavailable).

2.2 Transaction-specific Details

- The system shall require a price when creating a book listing for sale.
- The system shall require a rental duration when creating a book listing for rent.
- The system shall allow a book to be marked as exchangeable for exchange-type transactions.
- The system shall allow a book to be marked as free for giveaway-type transactions.

3. Requests / Transactions

3.1 Requesting a Book

- The system shall allow a logged-in user to request a book from another user.

3.2 Handling Requests

- The system shall allow the book owner to accept or decline a request.
- The system shall mark the book as unavailable once a request is accepted.
- The system shall allow the requesting user to view the status of their requests (Pending, Accepted, Declined).

4. Search and Filtering

4.1 Search

- The system shall allow users to search for books by title.
- The system shall allow users to search for books by author.

4.2 Filtering

- The system shall allow users to filter books by transaction type (Sell, Rent, Exchange, Giveaway).
- The system shall allow users to filter books by condition (New, Used, etc.).

- The system shall allow users to filter books by year of publication.
- The system shall allow users to filter books by genre.
- The system shall allow users to combine multiple filters for more precise results.

5. Nice-to-Have Features

- The system shall allow users to create book bundles containing multiple books in a single listing.
- The system shall allow users to create a wishlist of books they want.
- The system shall notify users if a book matching their wishlist becomes available.
- The system shall maintain a history of past transactions for each user.
- The system shall allow users to review books and other users.

Non-Functional requirements

- The system shall process API requests (e.g., creating or fetching book listings, handling requests) within 2 seconds under normal load.
- The system shall support at least 50 concurrent API clients without significant performance degradation.
- The system shall store user passwords securely using hashing.
- The system shall restrict access to API endpoints based on user authentication and authorization.
- The system shall validate all input data to prevent injection attacks (SQL injection, etc.).
- The system shall handle API errors gracefully, returning proper HTTP status codes.
- The system shall maintain consistent data state for books, users, and transactions even in case of concurrent requests.
- The system shall ensure that each book listing is associated with a valid user account.
- The system shall maintain accurate transaction status for books based on requests.
- The system shall enforce constraints like unique email per user and required fields for books.

Use cases

Use case 1

Name: User Registration
Actor: Guest (not logged in)
Description: Allows a new user to create an account in the system.
Precondition: User does not have an account.
Postcondition: User account exists in the system.

Main scenario:

1. User sends registration data (username, email, password) to the system.
2. System validates that the email is unique and password meets requirements.
3. System hashes the password and saves the new user in the database.
4. System confirms successful registration.

Alternative Flow:

1a. Input is invalid
   1. System returns validation errors.

2a. Email is already registered
   1. System returns an error.

Use case 2
Name: User Login
Actor: Registered User
Description: Allows a user to authenticate and access protected API endpoints.
Precondition: User has a valid account.
Postcondition: User is authenticated and can access protected endpoints.

Main Scenario:
1. User sends login request with email and password.
2. System retrieves the user by email.
3. System checks password against stored password.
4. If valid, system generates a session token or JWT for the user.

Alternative Flow:
1a. Email does not exist or password is incorrect
    1. System returns an error.

Use Case 3
Name: Create Book Listing
Actor: Logged-in User
Description: Allows a user to post a book for sale, rent, exchange, or giveaway.
Precondition: User is authenticated.
Postcondition: Book is listed and visible for search/filter.

Main Scenario:
1. User sends book details (title, author, year, genre, condition, transaction type) to the system.
2. System validates required fields and transaction-specific details (price, rental duration, etc.).
3. System saves the book listing and marks it as Available.

Alternative Flow:
2a. Validation of fields fails
    1. system returns errors.

Use Case 4
Name: Request Book
Actor: Logged-in User
Description: Allows a user to request a book from another user.
Precondition: User is authenticated; the book is Available.
Postcondition: Book request exists in the system with Pending status.

Main Scenario:
1. User sends a request specifying the listing ID of the book they want.
2. System validates that the book exists and belongs to another user.
3. System creates a request record with status Pending.

Alternative Flow:
2a. The book is already unavailable
    1. system rejects the request

Use Case 5
Name: Handle Book Request
Actor: Book Owner (Logged-in User)
Description: Allows a book owner to accept or decline a request for their book.
Precondition: User is authenticated and owns the book.
Postcondition: Book status and request status are updated accordingly.

Main Scenario:
1. Owner retrieves all requests for their books.
2. Owner chooses to accept or decline a request.
3. If accepted, system marks the book as Unavailable and updates request status to Accepted.
4. If declined, system updates request status to Declined.

Alternative Flow:
2a. The user is not the book owner
    1. system denies the action

Use Case 6
Name: Search & Filter Books
Actor: Logged-in User or Guest
Description: Allows users to find books based on criteria.
Precondition: Books exist in the system.
Postcondition: User receives a list of books matching criteria.

Main Scenario:
1. User sends search/filter parameters (title, author, genre, year, condition, transaction type).
2. System queries the database and returns a list of matching books.

Alternative Flow:
2a. No books match
1. system returns an empty list.

Objects, classes and relationships
Key objects:
- User - id, username, email, password
- Book - id, title, author, year, genre
- BookListing - id, book, owner, condition, transactionType, price, rentalDuration, status
- BookRequest - id, listing, requester, status, timestamp

Relationships:
- User-BookListing: one-to-many: one user can have multiple listings, listing belongs to one user
- Book-BookListing: one-to-many: one book can appear in multiple listings.
- User-BookRequest: one-to-many: one user can make multiple requests
- BookListing-BookRequest: one-to-many: one listing can have multiple requests.