School of Computational Sciences and Artificial Intelligence

Fall 2024

BookX

By

Abstarct Team

Members:

Abdelrahman elkady

Kareem Adel

Rana Hazem

1. Introduction

1.1 Vision

Our purpose is to develop a book exchange system “**BookX**” that is designed to create an efficient, community-based platform for sharing books. Our system will be built mainly using Java and Spring Boot.

1.2 Problem Statement

Traditional methods of book exchanges are manual, inefficient, and lack visibility. There is no streamlined solution to connect individuals with the book they wish to exchange, leading to missed opportunities and wasted resources. This system aims to address these issues by automating the process and ensuring ease of use for all book owners.

1.3 Scope

The system will include:

- **User Features**: Registration, login, profile management, and the ability to list and search for books.
- **Search Features**: Searching for the book you want
- **Administrative Tools**: Functions for managing users and moderating content.

---

2. Functional Requirements

The following functionalities will be implemented in the Book Exchange System:

1. **User Registration and Login**
    - Users can register and log in using their email and password.
    - Password encryption and validation for secure authentication.
2. **Profile Management**
    - Users can update their personal information, including contact details and profile pictures.
3. **Book Listing**
    - Users can add his book to the system with details like title, author, condition, and genre.
4. **Book Browsing**
    - Users can browse the available book categorized by genre, condition, or location.
5. **Advanced Search**

    - Filters include book title, author, genre, condition, and proximity to the user's location.
6. **Matchmaking System**
    - Suggests potential exchanges by comparing user-uploaded books with their requested books.
7. **Exchange Management**
    - Users can accept, decline, or cancel exchange requests.
8. **User Ratings and Feedback**
    - Post-exchange feedback to ensure reliability and accountability.
9. **Administrative Dashboard**
    - Admins can manage users, moderate content, and resolve disputes.
10. **Reporting Mechanism**
    - Users can report inappropriate behavior or listings for admin review.
11. **Exchange History**
    - Users can view a history of their completed exchanges.

---

3. Supplementary Specs and Non-Functional Requirements

The following quality attributes and constraints will guide system design:

1. **Performance**
    - The system should support 50 concurrent users with a response time of under 2 seconds for most operations.
2. **Scalability**
    - The architecture should allow easy scaling to accommodate future growth in user base and data.
3. **Security**
    - Implement encryption for sensitive data (e.g., passwords).
4. **Data Integrity**
    - Prevent duplication and loss of data through transactional consistency in the database.
5. **Usability**
    - Design a clean and intuitive user interface suitable for a wide audience.
6. **Maintainability**
    - Codebase should be modular and well-documented to simplify updates and debugging.
7. **Compliance**
    - Follow applicable data protection and privacy laws (e.g., GDPR).
8. **Accessibility**

- Ensure usability for individuals with disabilities by adhering to WCAG standards.

---

4. Use Cases (fully Dressed):

UC1: User Registration and Login

Scope: User Registration and Login
Description: This use case describes the process of a user registering for a new account or logging into the platform using their email and password.
Level: User Goal

Stakeholders: Book owners, Admin

Primary Actor:
User

Supporting Actors:
Authentication System

Pre-Conditions:

- The user must have an active internet connection.
- The system must be operational.

Main Scenario (Login):

1. The user navigates to the login page.
2. The system prompts for email and password.
3. The user enters valid credentials.
4. The Authentication System validates the credentials.
5. The user is logged in successfully and redirected to the homepage.

Alternative Scenario (Forgot Password):

1. The user enters incorrect credentials three times.
2. The system locks the account temporarily and shows a "Forgot Password" button.
3. The user clicks "Forgot Password" and is prompted to enter their registered email.
4. The system sends a password reset link via email.
5. The user clicks the link, resets the password, and logs in with the new password.

Post-Conditions:

- The user is logged in.
- The user can access all system features based on their role.

UC2: Book Listing

Scope: Book Listing
Description: This use case describes the process of users adding books to the platform with detailed information.
Level: User Goal

Stakeholders: Book owners, Admin

Primary Actor:
User

Supporting Actors:
None

Pre-Conditions:

- The user must be logged in.
- The user has a book to list.

Main Scenario:

1. The user navigates to the "Add Book" section.
2. The system prompts the user to fill in book details (e.g., title, author, genre, condition).
3. The user uploads a photo of the book.
4. The system validates the entered details.
5. The book is successfully listed and made available for browsing by others.

Post-Conditions:

- The book is visible to other users on the platform.
- The listing is stored in the system database.

UC3: Book Search Browsing

Scope: Advanced Search and Book Browsing
Description: This use case describes the process of users searching for books using filters and browsing through available listings.

Stakeholders: Book owners.
Level: User Goal

Primary Actor:
User

Supporting Actors:
None

Pre-Conditions:

- The user must be logged in.
- The system must contain available book listings.

Main Scenario:

1. The user navigates to the "Search" or "Browse" section.
2. The system displays available books categorized by genre, condition, and location.
3. The user enters specific search criteria.
4. The system filters results based on the input and displays matching books.
5. The user views the book details or initiates an exchange request.

Post-Conditions:

- The user finds a book of interest or refines their search further.

UC4: Exchange Management

Use Case Name: Exchange Management
Description: This use case describes the process of users initiating and managing book exchanges.
Level: User Goal

Stakeholders: Book owners, Delivery Courier.

Primary Actor:
Requesting User

Supporting Actors:
Recipient User, Courier/Delivery System

Pre-Conditions:

- Both users must be logged in.
- Books must be listed in the system.

Main Scenario:

1. The requesting user views a book and selects "Request Exchange."
2. The system prompts the requesting user to select a book to offer in return.
3. The system sends the exchange request to the Recipient User.
4. The Recipient User reviews the request and either accepts or declines it.
5. If accepted, the system initiates delivery coordination with the Courier/Delivery System.

Alternative Scenario:

1. The recipient user declines the exchange request.
2. The system notifies the requesting user about the decline.
3. The requesting user can select another book to offer or cancel the exchange request.

Post-Conditions:

- If accepted, the exchange is processed and tracked.
- Both users provide feedback after the exchange.

UC5: Provide Feedback

Scope: Provide Feedback
Description: This use case describes the process of users leaving feedback for each other after completing a book exchange.
Level: User Goal

Stakeholders: Book owners, Admin, Customer Service representative.

Primary Actor:
Requesting User, Recipient User

Supporting Actors:
None

Pre-Conditions:

- A completed exchange exists between the users.
- The users are logged into the system.

Main Scenario:

1. The user navigates to the "History" section.
2. The system displays a list of completed exchanges.
3. The user selects a completed exchange.
4. The system prompts the user to provide feedback (rating and comment).
5. The user enters the feedback and submits it.
6. The system stores the feedback and updates the other user's profile with the new rating.
7. The system confirms that feedback has been submitted successfully.

Alternative Scenario:

1. The user attempts to provide feedback for an exchange that has not been completed.
2. The system displays a message: "Feedback can only be given for completed exchanges."

Post-Conditions:

- Feedback is saved and reflected in the recipient's profile.
- The user can view their submitted feedback in the "Exchange History" section.

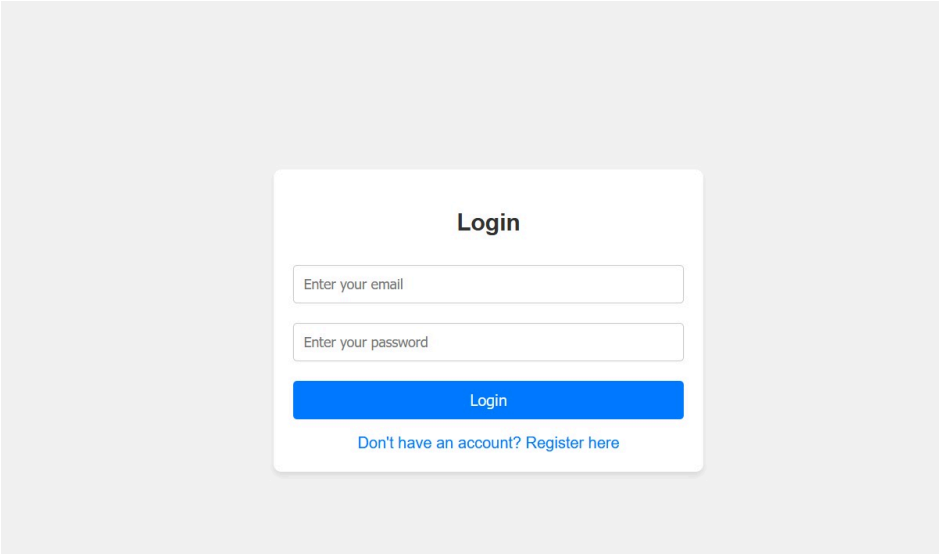
5. Design Patterns Used:

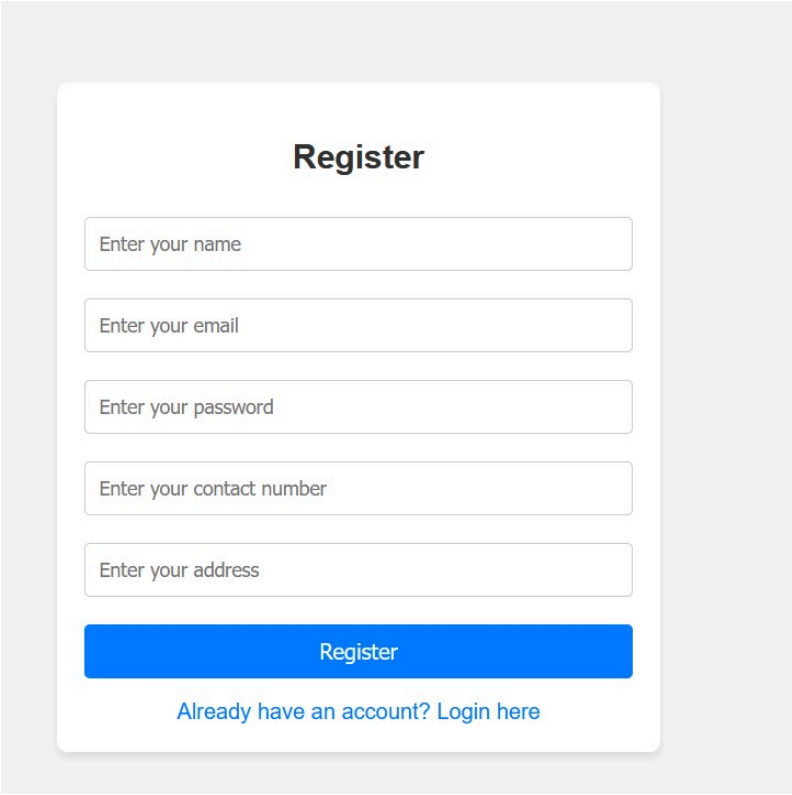
- **Singleton**: we should have one instance of a user at a time, used in class (User Manager).
- **Creator:** User Should Create the report, feedback, exchange. Used in class (User).
- **Information Expert:** the book should have all data needed and sufficient for the user about the book. Implemented in class (Book, Exchange).

6. Diagrams:

All diagrams including Domain Model, Class Diagram, etc… Can be found on our [Github Repository]

7. GUI Snapshots:





User Books

Book Title
Author: Author Name
Genre: Genre
Condition: Condition

BookX

Hello, User!
Email: user@example.com

Add a Book
Report a Book

Welcome to BookX. Your one-stop platform for book exchanges.

Search for books or authors...

Exchange History

Exchange 1
User 1
User 2
Book Title: Book 1
Details: Some details about the book and exchange.

Exchange 2
User 3
User 4
Book Title: Book 2
Details: Some details about the book and exchange.

Exchange 3
User 5
User 6
Book Title: Book 3
Details: Some details about the book and exchange.

Book List

Book Title 1
Author 1
Details about the book.
BookX

Book Title 2
Author 2
Details about the book.
BookX

Book Title 3
Author 3
Details about the book.
BookX

8. Conclusion

The **Book Exchange System** will provide a robust platform for users to exchange books seamlessly. By implementing advanced features such as matchmaking and search filters, coupled with a focus on usability and security, the system should address the challenges of traditional book exchange methods effectively.

9. Lessons learnt:

Lessons Learned from the OOAD Project

1. **Requirements Gathering**
   Clear identification of functional and non-functional requirements ensured the system met user needs effectively.

2. **Application of OOAD Principles**
   Design patterns (e.g., Singleton, Creator) and UML diagrams facilitated modular, maintainable, and scalable system design.

3. **Collaboration and Communication**
   Effective teamwork, version control (GitHub), and shared documentation improved project coordination.

4. **Transition from Design to Code**
   Mapping high-level designs to Java/Spring Boot highlighted the complexity of bridging analysis with implementation.

9. Task Distribution:

Kareem Adel:

Use Case Diagram, Fully dressed use cases, Functional requirements, Proposal, Activity diagram, Logical architecture, FrontEnd.

Rana Hazem:

Activity Diagram, Sequence Diagram, interaction, state machine, System sequence, OCL Constraints.

Abdelrahaman ElKady:

Class Diagram, Domain Model, FrontEnd, Spring Backend, ORM database and persistent layer, ER diagram.