Fall 2025 ByteBuilder Java - Final Project

Library Book Borrowing System

Team: Hoeu Tiangkimhong, Song Kimhor, Khoy Morngdyza, and Seng Mengkheang

Repository: https://github.com/tkimhong/Library-Management-System

---

Overview

The Library Book Borrowing System helps school libraries manage users, books, and borrowing activities. Librarians can register users, update book details, and search the catalog by title, author, genre, or ISBN. The system handles book checkouts with a 14-day borrowing period, processes returns, and automatically detects overdue books. It enforces a 3-book limit per user and confirms only active users can borrow.

Purpose

Manual library management leads to errors, lost records, and inefficient tracking. Without automation, librarians struggle to monitor book availability, enforce borrowing limits, and track overdue items consistently.

Our system solves this by automating these tasks. It tracks book availability in real-time, enforces the 3-book limit automatically, and flags overdue books without manual checking. This reduces librarian workload, minimizes errors, and creates a fair borrowing process for students.

Solution

Architecture

We built the system using Java with three main packages:

- `models/` - Data classes (User, Book, BorrowRecord)

- `services/` - Business logic (UserService, BookService, BorrowingService, RulesEngine)
- `ui/` - Console interface (MainMenu)

This separation let each team member work independently while maintaining clean integration.

What Each Module Does

User Management (Mengkheang)

- Add, view, edit, and deactivate users
- Track name, student ID, email, and active status
- Only active users can borrow books

Book Management (Dyza & Tiangkimhong)

- Add, view, update, and remove books
- Track title, author, genre, ISBN, and copy counts
- Search by title, author, genre, or ISBN
- Automatic copy tracking when books are borrowed/returned

Borrowing System (Kimhor)

- Process book checkouts and returns
- Create borrow records with due dates (14 days)
- Track which books are currently borrowed
- Update book availability automatically

Rules Engine (Tiangkimhong)

- Validate user eligibility (active status + under 3-book limit)
- Check book availability
- Calculate due dates
- Detect and mark overdue books

Main Menu (Tiangkimhong)

Console interface with menus for:

1. User management
2. Book management
3. Borrowing operations
4. Overdue book reports

Technical Decisions

- **ArrayList storage** - Simple in-memory data storage (no persistence needed for this project)
- **LocalDate for dates** - Java's date class handles due date calculations and overdue checking
- **Centralized validation** - RulesEngine handles all business rules in one place
- **Modular design** - Each module works independently, making testing and integration easier

---

Reflection

Main Challenges

Team Coordination: As one but all were freshmen, we needed frequent check-ins and clear documentation. Some members needed extra guidance on Java basics, which took additional time for code reviews.

Module Dependencies: The borrowing system couldn't start until User and Book models were done. We learned to identify critical path items early in planning.

Time Management: Balancing this with other coursework was tough, especially with the Dec 5th and Dec 19th deadlines approaching.

New Concepts: Freshmen members were learning OOP, service architecture, and date/time handling with LocalDate while building the project.

Key Lessons

Modular Architecture: Separating code into layers made development manageable. Each person could focus on their module without breaking other parts.

Git is Necessary: Version control with regular commits prevented last-minute integration nightmares.

Test Early: Testing each method as we wrote it caught bugs early instead of dealing with everything at the end.

Ask for Help: When someone got stuck, asking early saved way more time than working alone.

What We Learned Individually

- **Mengkheang**: Built confidence in CRUD operations and designing foundational data models
- **Dyza**: Learned to implement multiple search strategies (exact vs partial matching)
- **Kimhor**: Practiced coordinating between multiple modules and working with date-based logic
- **Tiangkimhong**: Developed leadership skills while implementing the most complex module and integrating everything

---

Conclusion

We successfully built a functional library management system that automates user tracking, book inventory, borrowing operations, and overdue detection. The project met its objectives: 3-book limit enforcement, 14-day borrowing periods, automatic overdue flagging, and a complete user interface.

Beyond the code, we learned valuable lessons about teamwork, communication, and managing dependencies. Working with mixed experience levels taught us patience and knowledge sharing.

The experience of planning, implementing, and integrating a complete system prepared us well for future software projects.