Development and Management of: CPP Library System

Spencer Barrett

011727157

Department of Computer Science, California State Polytechnic University Pomona

Dr. Ericsson Santana Marin

July 12, 2025

List of Content

List of Figures

Figures

List of Tables

Tables

REQUIREMENTS SPECIFICATION

Functional Requirements

Student Management:

Id	Requirement	Rationale
SM-01	The system should allow staff to create, read, update, and delete student records.	Managing student information is essential for loan processing.

Table 1.0

Book and Copy Management:

Id	Requirement	Rationale
BCM-01	The system should allow staff to create, read, update, and delete book records.	Keeping the catalog of books accurate and up to date.
BCM-02	The system should allow staff to create, read, update, and delete book copy records.	Managing individual copies and their status.

Table 1.1

Loan Processing:

Id	Requirement	Rationale
LP-01	The system should allow staff to create, read, update, and delete loan records.	Tracking borrowing transactions between students and the CPP library.
LP-02	The system should check for overdue books before approving a loan.	Enforcing library policy
LP-03	The system should prevent students from borrowing more than 5 books at once.	Maintain fairness, encourage returns, ensure availability of books.
LP-04	The system should restrict loan duration from exceeding 180 days.	Ensure books are circulated properly.
LP-05	The system should require all book copies associated with a loan to be returned together in one transaction	Enforce policy and ensure accurate returns handling.

Table 1.2

Book Search and Availability:

Id	Requirement	Rationale
BSA-01	The system should allow staff to search for books by title, author, or ISBN.	Improve book discovery and reduce time spent manually browsing.
BSA-02	The system should display all book copies' availability and due dates	Help staff and students understand what books are available

Table1.3

Receipts and Reports:

Id	Requirement	Rationale
RR-01	The system should display a loan receipt showing loan number, student name, borrowing date, due date, and borrowed items.	Confirm and summarize transactions
RR-02	The system shall generate loan reports filtered by students and loan period.	Help administrative tracking

Table 1.4

Assumptions

System Setup:

Id	Assumption	Rationale
SS-01	The system will be a desktop application with a simple GUI	Outlined in requirements to specifically use JavaFX
SS-02	No server or hosting necessary	Not mentioned in requirements

Table 2.0

Users:

Id	Assumption	Rationale
U-01	Only library staff intended to use.	Staff at base of all use-case diagrams
U-02	No authentication or login necessary.	Not mentioned in requirements

Table 2.1

System Operation:

Id	Assumption	Rationale
SO-01	Each book copy belongs to one book.	Each book may have multiple copies, but each copy is from a single book
SO-02	Each loan is tied to one student and includes specific book copies	Loans are created based on students borrowing a group of books at once
SO-03	All books in a loan must be returned together	Partial returns are not allowed

Table 2.2

Out of Scope:

Id	Assumption	Rationale
OOS-01	Book demand tracking and reservation systems will not be included	Listed as outside of scope in requirements
OOS-02	Fines/Charging mechanisms or late fees will not be included	Listed as outside of scope in requirements

Table 2.3

DESIGN

Domain Model

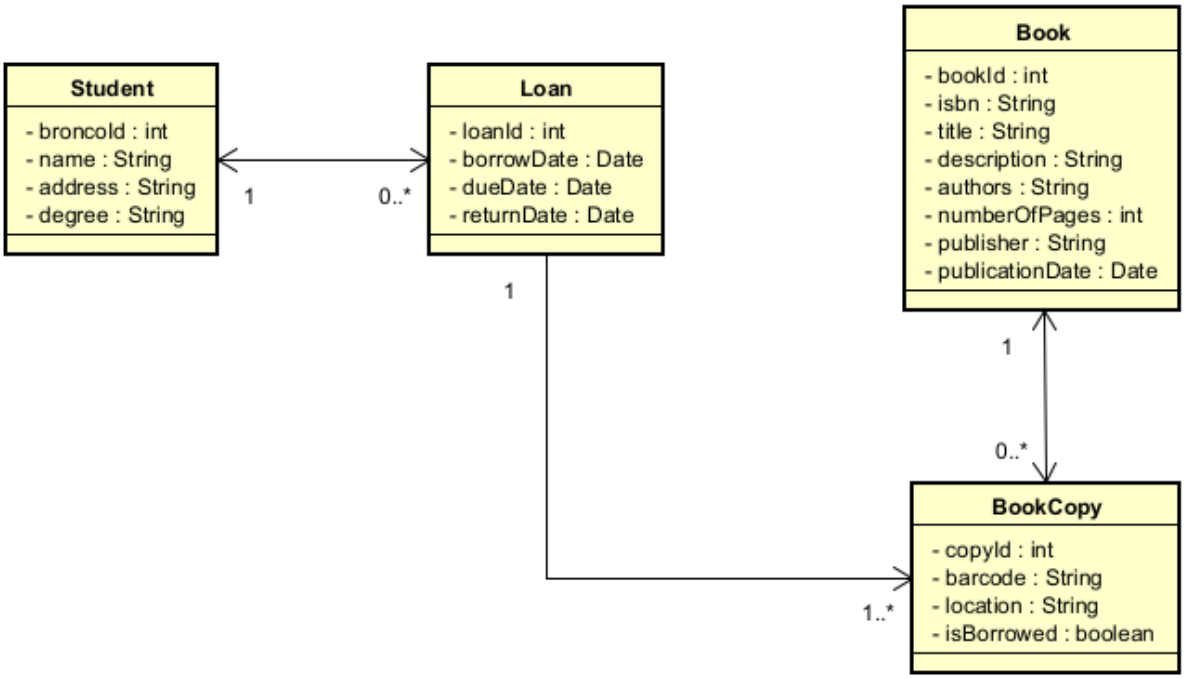

Figure 1.0

Object Oriented Model

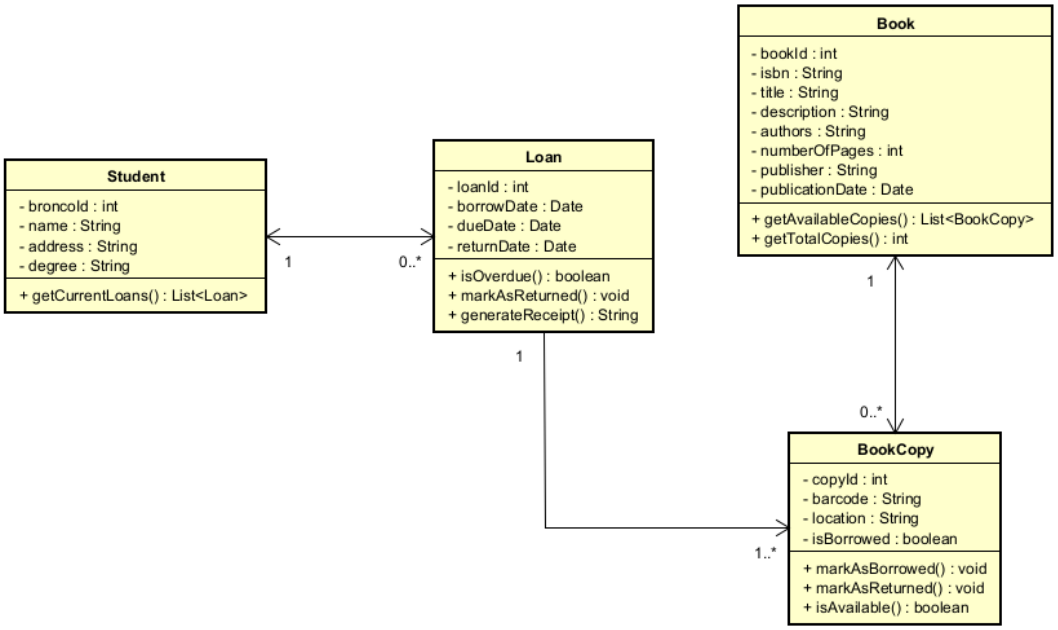

Figure 1.1

Data Logical Model

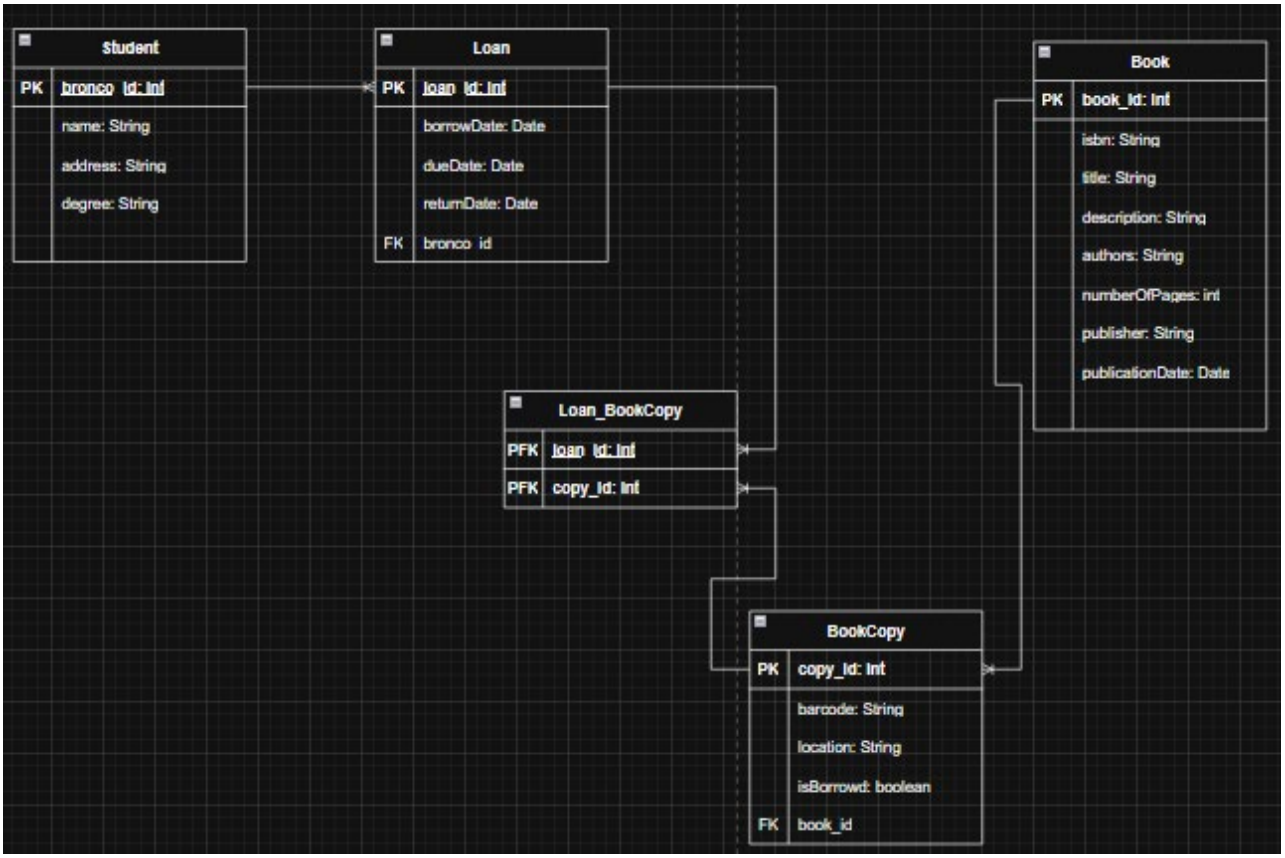

Figure 1.2

IMPLEMENTATION

GitHub Repository

https://github.com/spencer-barrett/cs3560_Project

This system was implemented using JavaFX for the GUI, PostgreSQL for the database, and Hibernate for the ORM (Object-Relational Mapping) framework. I separated this project using the MVC (model-view-controller) pattern. We have 3 key directories: controller, model, and service. My controller classes serve as a "middleman" between the UI and logic by handling actions and updating the view programmatically. The model classes are used to represent data and map to the database schema. Lastly, the service classes handle logic, validation, and database operations. We also have view files (fxml) which serve as the visual layer of the system for JavaFX UI.

I decided to implement this system with 5 screens: Welcome, Loan Management, Book Management, and Book Copy Management. On the welcome screen I thought it would be best to implement a navigation bar to easily navigate between the different screens. I also thought it would be important, as a visual aid, to highlight the current tabbed screen to additionally mark which screen the user was currently on.

- The Student Management screen is simple and has necessary CRUD capabilities displaying all current students with Bronco Id, Name, Address, and Degree. To create or update a student we must fill out all fields and select a button to proceed.
- The Loan Management screen is a bit more complex as we have to load from 3 different tables. At the top we have a combo box where we can select students from as well as date inputs for indicating the period of a book loan. On the left of the screen, we have a list of book copies the user can select from to add to a new loan so long as the copy is available. On the right side, we can track which books we have added to the loan so far which helps

act as a visual aid to the user. We also have a loan table which displays all the loans that have been created. Above the loan table we have our CRUD functions as well as a receipt display. Lastly, below the loan table we have a button to generate loans grouped by students and sorted by due date.

- The Book Management screen, like the Student Management screen, is pretty rudimentary and simply displays all the books with necessary details like: Book Id, ISBN, Title, Authors, Publisher, Pages, Publication Date, and Description. We have all the necessary CRUD capabilities and input boxes necessary for creating new books or updating information on current books.
- Lastly, the Book Copy Management screen contains a table of all copies belonging to the library system. Here, we can see whether a copy is available or is currently checked out. If it is already checked out, we can see when the book is due. Furthermore, we have the necessary CRUD capabilities. We can select from a combo box to choose which book we would like to either create a new copy or update from as well as inputs for barcode, location, and indicating availability status.

DISCUSSION

Although this project was challenging, I enjoyed it and think it was a valuable tool as a steppingstone towards becoming a well-rounded software engineer. Working with a user interface as a “frontend” as well as the underlying logic layers or a sort-of “backend” helped reinforce key concepts that we’ve learned throughout this course.

Learning a new technology like JavaFX to create desktop applications is something that I believe I will use again in the future on my own in personal projects to help expand my programming knowledge. Furthermore, I think exposure to tools like PostgreSQL in school and gaining as much hands-on experience as possible is invaluable as it is a technology I know is used heavily within the industry and opens the door to more real-world development.

References

Marin, E. (n.d.). Lecture 8: Graphical User Interfaces (GUI) – Basic Controls. CS 3560 – Object-Oriented Design and Programming. California State Polytechnic University, Pomona. https://www.cpp.edu/

Marin, E. (n.d.). Lecture 9: Architectural, Enterprise/Application, and Design Patterns. CS 3560 – Object-Oriented Design and Programming. California State Polytechnic University, Pomona. https://www.cpp.edu/