Systems Requirements Documentation - Library Management System
Final

Logan Young

Library Management System Proposal

Problem Statement

A library management system is used to manage the books in a library through the use of an automated system. Basically, the system tracks the status of books and other media throughout the checkout process (whether a book is checked out or available, in storage, reserved, etc.). It can be difficult to keep track of everything a larger-sized library might carry through typical analog filing systems, so automated management systems are used to keep everything organized.

System Objectives

The library management system (LMS) will contain the entire inventory of the library in a database, converting various properties of books and other media to attributes. It will also keep track of if a book is available or has been loaned out, if it's reserved, if it's damaged, etc. Its primary use is for managing the library's inventory, but there will also be a frontend to allow library members to check out books.

System Requirements

- Let library members check out books.
- Let library members reserve books.
- Let library members request a book from a library in the same network.
- Let library staff mark a book as returned.
- Let library staff create a new entry for a new book the library has received.
- Let library staff delete an entry for a book the library no longer possesses.
- Let library staff mark a user as a new library member.
- Let library staff adjust a book's properties if there's been a mistake in filing (e.g. wrong publish date, wrong author).
- Let library administrator mark a user as new library staff.
- Let library administrator perform various administrative tasks (not too sure what this might entail).

Typical Customers

- Library members (the people who check out the books)
- Library staff (the people who work at the library)
- Library administrator (the person in charge of the library)
- IT Staff (the people who work on the LMS, supposing they're separate from staff)

Project Planning and Development Approach

1. Software
    a. Frontend: React (or JavaScript if I can't learn React in time)
    b. Backend: Java
    c. Database: MySQL
2. Hardware
    a. PC (or device with an internet browser to access the web app)
    b. Small server in the library for the database.
3. Network
    a. [In all honesty, the LMS database and web app shouldn't be so intensive as to need high-speed networking, but it's good to future-proof.]
    b. Fiber-optic internet with ethernet integration to allow quick uploading and downloading from the server to the web app.

Development Plan

- Weeks 1-2
    - Conceptualize and complete the system proposal.
- Weeks 3-4
    - Determine connection between frontend, backend, and database and start construction.
- Weeks 5-7
    - Build database to organize books and their attributes, build frontend webpage for users to interact with, build backend to connect the two.
- Week 8
    - Test system with sample entries and actions and record demo for midterm.
- Weeks 9-11
    - Improve system with feedback from professor. Implement suggested features and bug fixes.
- Weeks 12-14
    - Write test cases and optionally add some cool last-minute features.
- Week 15
    - Test final system and record demo for final.

Problem Statements and System Requirements Document

Logan Young

Customer Problem Statement

Problem Statement

A **Library Management System** keeps track of all the goings-on in a library. It catalogs the entire inventory of the library, keeping track of each attribute of each piece of media. It also logs the library's members, its staff, and its administrator as users. If a member checks out a book the LMS will alter that book's checkout status to reflect that change. A member can also view what books are available and submit requests to hold a book. Other users have more privileges related to the system, such as being able to mark a book as unavailable manually or remove a member's records.

Glossary of Terms

LMS – (Library Management System) A tool to electronically manage a library.

User – An entity that uses the LMS to interact with the library.

Member – An LMS user that can only check out and return books.

Staff – An LMS user with elevated privileges related to library upkeep.

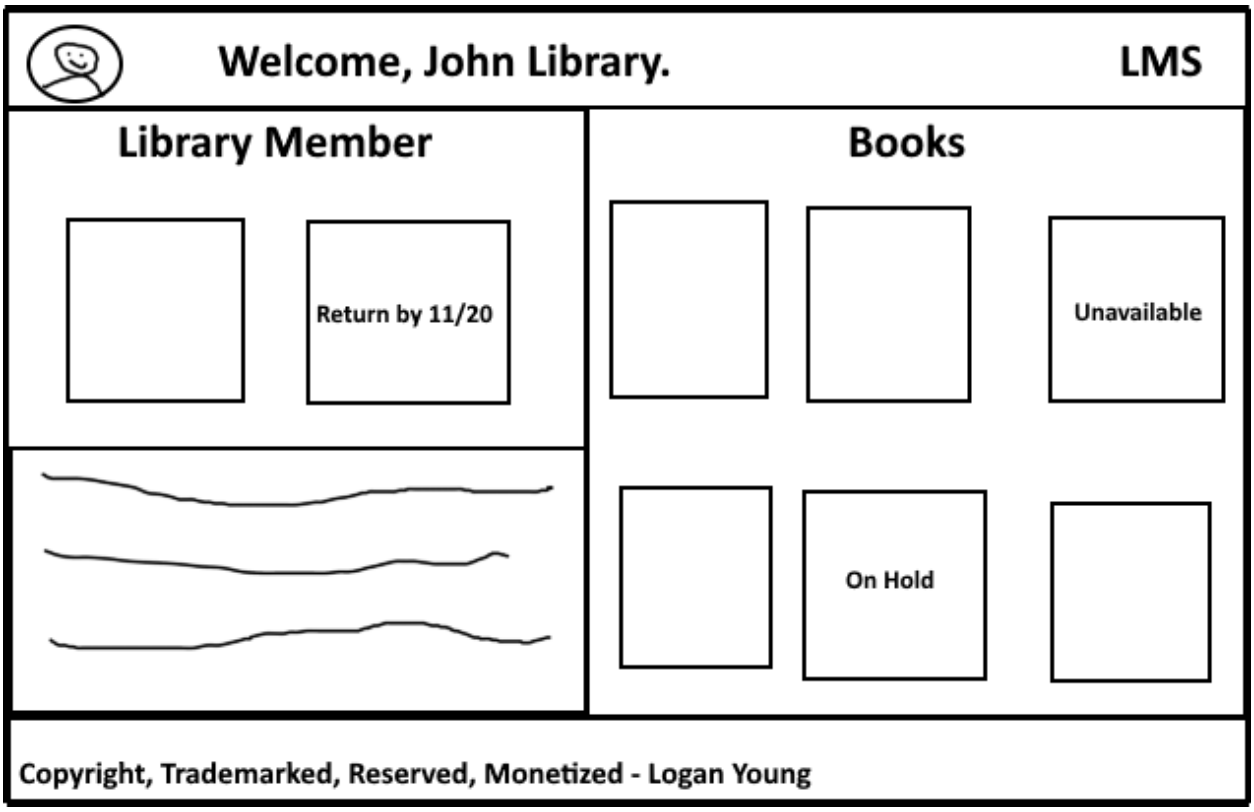
Administrator – LMS staff with further elevated privileges.

Checked-out – Status of a book after a user has reserved it. Unavailable for use by other users until returned.

Hold – Status of a book after a user has submitted a hold request. No user other than the one that requested the hold can check it out after it becomes available.

Attribute – Some quality of a book, such as its title or author.

Privilege – Specific groups of users can access more of the LMS than others.



System Requirements

Functional Requirements

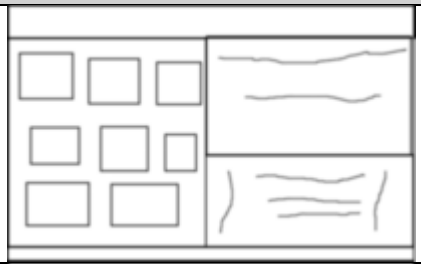
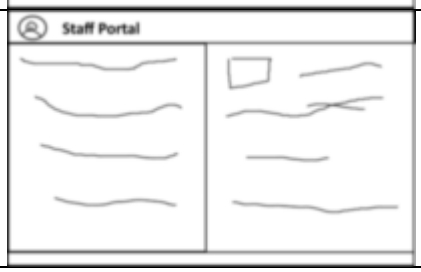
No.	Priority Weight	Description
REQ-1	High	The LMS should be able to store and display books.
REQ-2	High	Library members should be able to use the LMS to check out books.
REQ-3	Medium	Library members should be able to use the LMS to submit a hold request for a maximum of three books.
REQ-4	High	Library staff should be able to use the LMS to mark a book as unavailable.
REQ-5	High	Library members should be able to delete their account from the LMS.
REQ-6	Medium	Library staff should be able to delete a member's account using the LMS.
REQ-7	High	Library staff should be able to convert a new user into a member.
REQ-8	Medium	Library staff should be able to adjust a book's attributes if incorrect.
REQ-9	High	Library staff should be able to create a new entry in the LMS for a newly received book.
REQ-10	Low	Library members should be able to request a book from a different library in the same network.
REQ-11	Medium	Library administrator should be able to convert a new user into staff.
REQ-12	Low	Library administrator should have (nearly) complete access to the LMS.
REQ-13	Medium	Library members should be able to submit a suggestion for new books for the library.

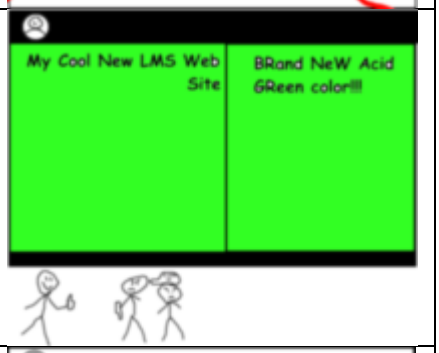
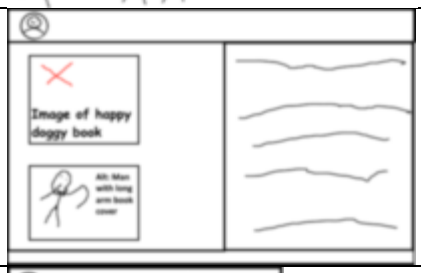
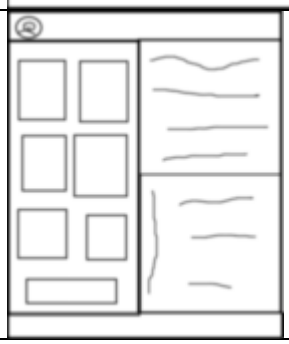
REQ-14	Medium	Library members should be able to create a new donated book entry with approval from staff.
REQ-15	High	LMS users should be able to easily search for a specific book using various filters and sorting techniques.

Nonfunctional Requirements

No.	Priority Weight	Description
N-REQ-1	High	The LMS should be scalable to include as many books as possible.
N-REQ-2	Medium	The LMS should be able to support at least 100 concurrent users.
N-REQ-3	High	The LMS front-end (HTML and CSS) should pass W3C validation with no errors.
N-REQ-4	High	The LMS front-end should pass all test cases for Google Chrome, Microsoft Edge and Mozilla Firefox.
N-REQ-5	Medium	All database processes should be completed within a maximum of 30 seconds.
N-REQ-6	High	LMS users should have password-protected accounts.
N-REQ-7	Medium	Front-end and back-end code should be written with readability and maintainability in mind.
N-REQ-8	Medium	Database should support placeholder entries for testing purposes that won't appear for users with insufficient privileges.
N-REQ-9	Low	Database should support different types of media rather than only books.
N-REQ-10	Medium	Full database backups should be performed weekly.
N-REQ-11	High	Frontend updates should occur only on Saturdays at 2:00 a.m. EST.

User Interface Requirements

No.	Priority Weight	Description	Graphic
UI-REQ-1	High	The website should be neat and organized.	
UI-REQ-2	High	Once logged in, each user should see specific options available to them according to their privileges.	

UI-REQ-3	Medium	No backend or database operations should be overtly visible to the user.	
UI-REQ-4	Medium	UI changes should be implemented based on consensus from users and council of IT department.	
UI-REQ-5	High	All essential UI components should be screen-readable (including alt text for images, default text for errors, transcripts for videos).	
UI-REQ-6	Low	UI components should scale based on monitor resolution.	

Plan of Work

- Weeks 1-2
    - Conceptualize and complete the system proposal.
- Weeks 3-4 (**We are here.**)
    - Determine connection between frontend, backend, and database and start construction.
        - **Going well so far. My backend has switched from Java to Node.js. I've set up the MySQL schema and started on the HTML for the site. Connecting all three is proving a little confusing, but I'll figure it out.**
- Weeks 5-7

    - Build database to organize books and their attributes, build frontend webpage for users to interact with, build backend to connect the two.
- Week 8
    - Test system with sample entries and actions and record demo for midterm.
- Weeks 9-11
    - Improve system with feedback from professor. Implement suggested features and bug fixes.
- Weeks 12-14
    - Write test cases and optionally add some cool last-minute features.
- Week 15
    - Test final system and record demo for final.

Functional Requirements Specification

Logan Young

Use Case Specifications

Stakeholders

1. Library staff – helpful tool to compile information on their library's stock and people who use the library
2. Library members – allows them to see what books are in stock and perform a variety of useful actions
3. Library administrators – allows them to perform administrative duties automatically and/or digitally that would otherwise be time-consuming
4. Library board of trustees – allows them to gauge interest in the library and better understand their goals
5. Local governments – a better functioning library helps to better educate the populace and the populace in turn improves their community and their government

Actors and Goals

Primary Actors:

1. Library member: This actor checks out books, marks books as returned, and places holds.
2. Library staff: This actor manages member accounts and book statuses.
3. Library administrator: This actor manages the entire system, holding the highest privileges.

Secondary Actors:

1. System: This stores information about library users, contents, etc. and performs operations.

Use Cases

Library Member (Total: 14)

1. Check out book: To check out a book from the library. (2)
2. Return book: To return a book to the library. (2)
3. Hold request: To request a hold on a book from the library. (1)
4. Procurement request: To request a book from a different library in local network. (3)
5. Delete self: To delete a user's own account from the LMS. (2)
6. Submit suggestion: To submit a suggestion for a new book to the library. (1)
7. Donate book: To donate a book to the library. (3)

Library Staff (Total: 11)

1. Hide book: To mark a book as unavailable. (1)
2. Delete account: To delete a member's account. (2)
3. Create account: To create a new account for a member. (3)
4. Adjust book: To adjust a book's attributes. (3)
5. Create book: To create a new book. (2)

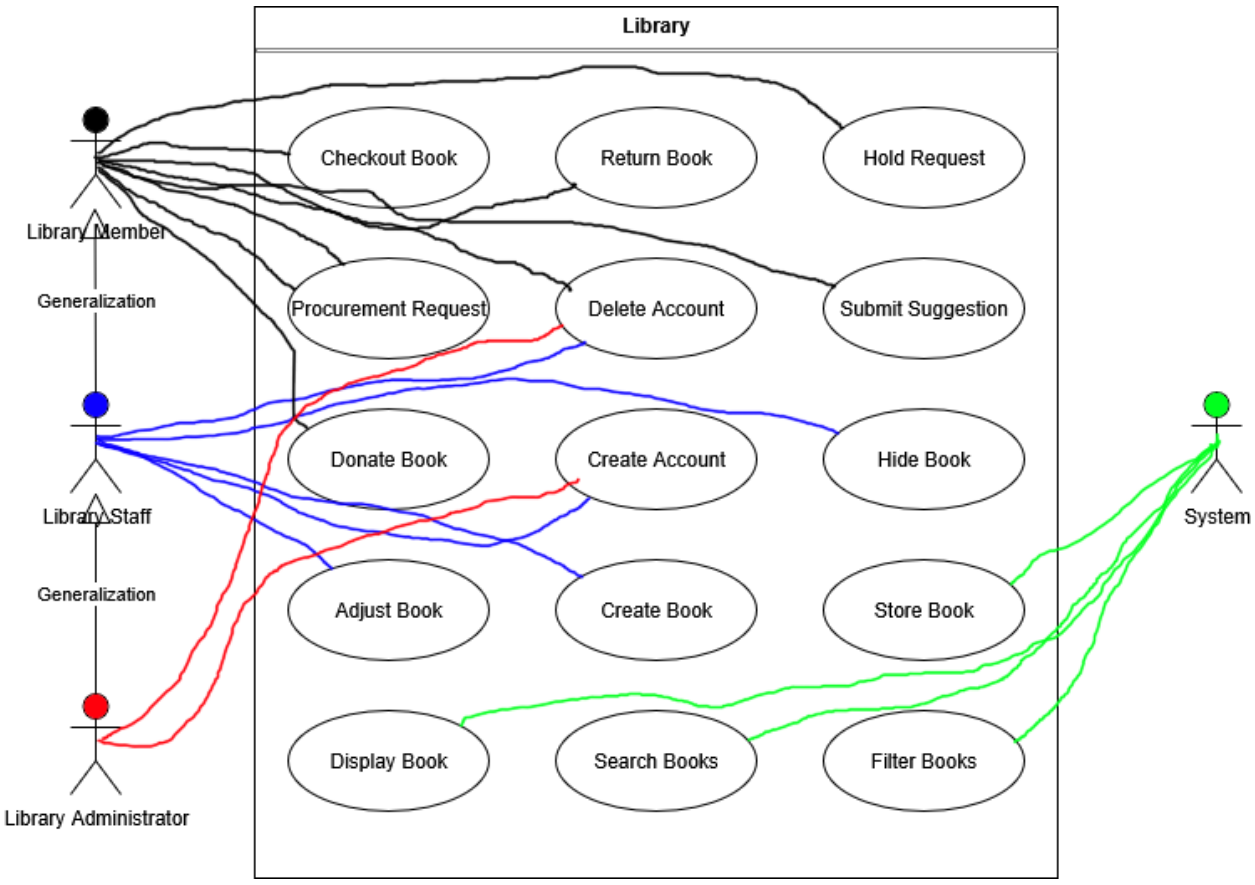
Library Administrator (Total: 4)

1. Create staff account: To convert a member account into a staff account. (2)
2. Delete account: To delete any level account (excepting administrator). (2)

System (Total: 14)

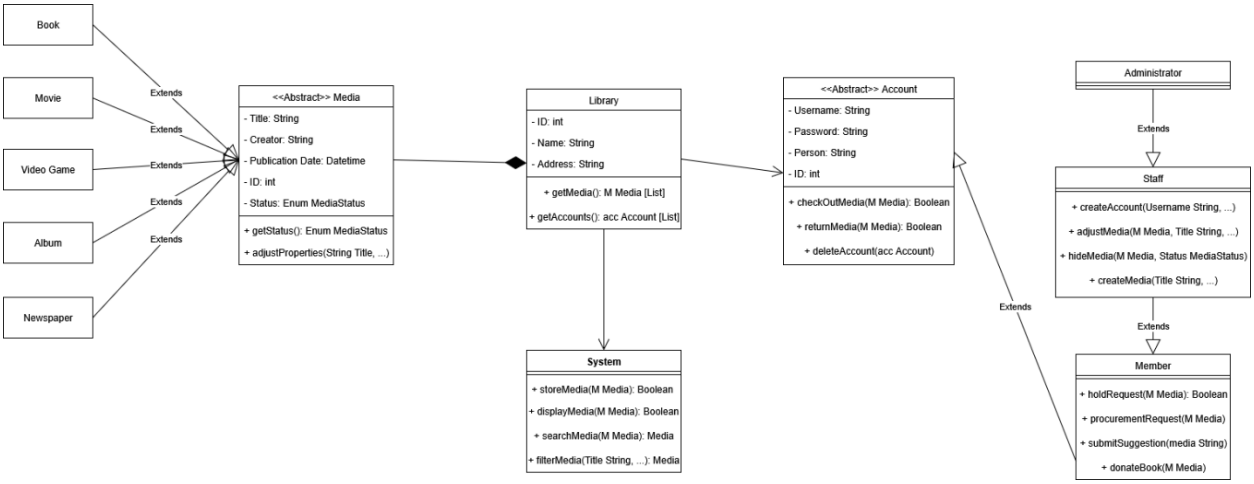
1. Store book: To store a new book in database. (3)
2. Display book: To display a new book from database. (3)
3. Search book: To search through database for a book matching criteria. (4)
4. Filter books: To filter database via matching criteria. (4)

Use Case Diagram



Class Specifications

Class Diagram



(This is a rather bare-bones class diagram. As I get further along with implementation, this is subject to change. I have a tendency to scope-creep all of my projects.)

Class Diagram Explanations

Media

Media is an abstract class since “media” isn’t itself a thing, and more of a category. It’s the template for things like books and movies. If this was a more in-depth project, each implementation of media would have its own attributes. (They still might; I haven’t made up my mind yet.) Media attributes are self-explanatory. Status refers to whether the media is “available”, “checked out”, “reserved”, or otherwise “unavailable”. It has a few basic methods right now, one to check the book’s status for the site and one to adjust properties via library staff.

Library

Library is the base for pretty much everything. It’s got a name, address, and ID since it’s planned to be able to request a piece of media from a networked library. You can get the list of Media and Accounts from the library. The accounts access the library, as well as the system.

System

The system has no attributes right now, but it does have several methods, mainly for use with the website (the user interface). The system can store media, display media, search for specific media, and filter media via provided criterion.

Account

Account is an abstract class like Media. There are a few basic attributes and methods that should be provided to all implementations. Unlike Media, there’s a hierarchy to things with Member at the bottom and Administrator at the top. Administrator can do anything that any account class below it can do.

Sequence and Activity Diagrams

Logan Young

System Sequence Diagrams

Hold Request

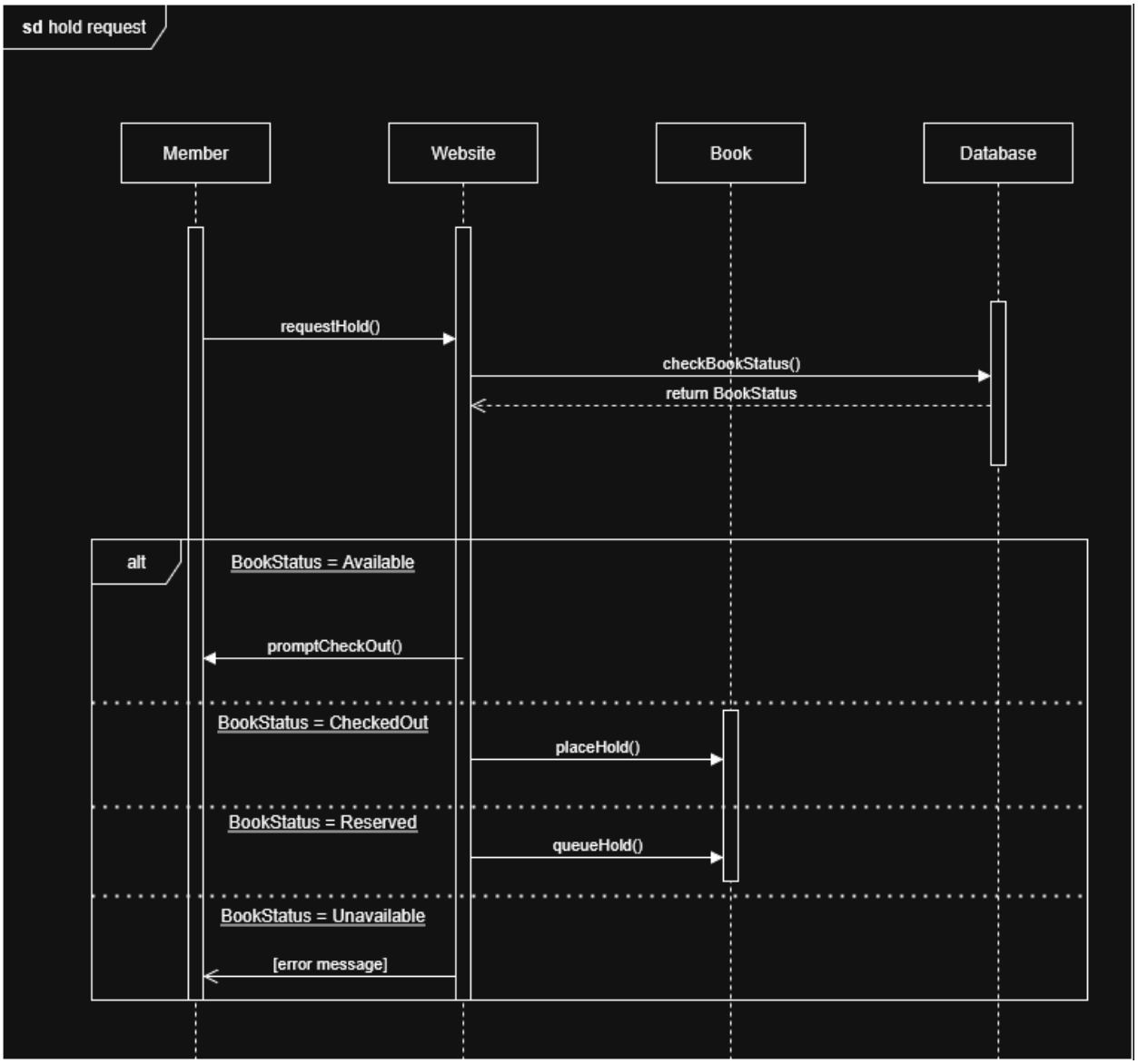
Actor and Objects

- Actor: **Member**
- Objects:
    - Website
    - Library Database
    - Media

Process

1. A library member requests a hold on a book they want to borrow.
2. The system checks the book's current state: if it's available, if it's checked out, if it's reserved by another member, or if it's unavailable.
3. If the book is available: the member receives a prompt to check out the book now.
4. If the book is checked out with no other holds: the hold is placed successfully.
5. If the book is reserved by at least one other member: the member is added to the hold queue.
6. If the book is unavailable: the hold fails and an error message is sent.

Diagram



Adjust Book

Actors and Objects

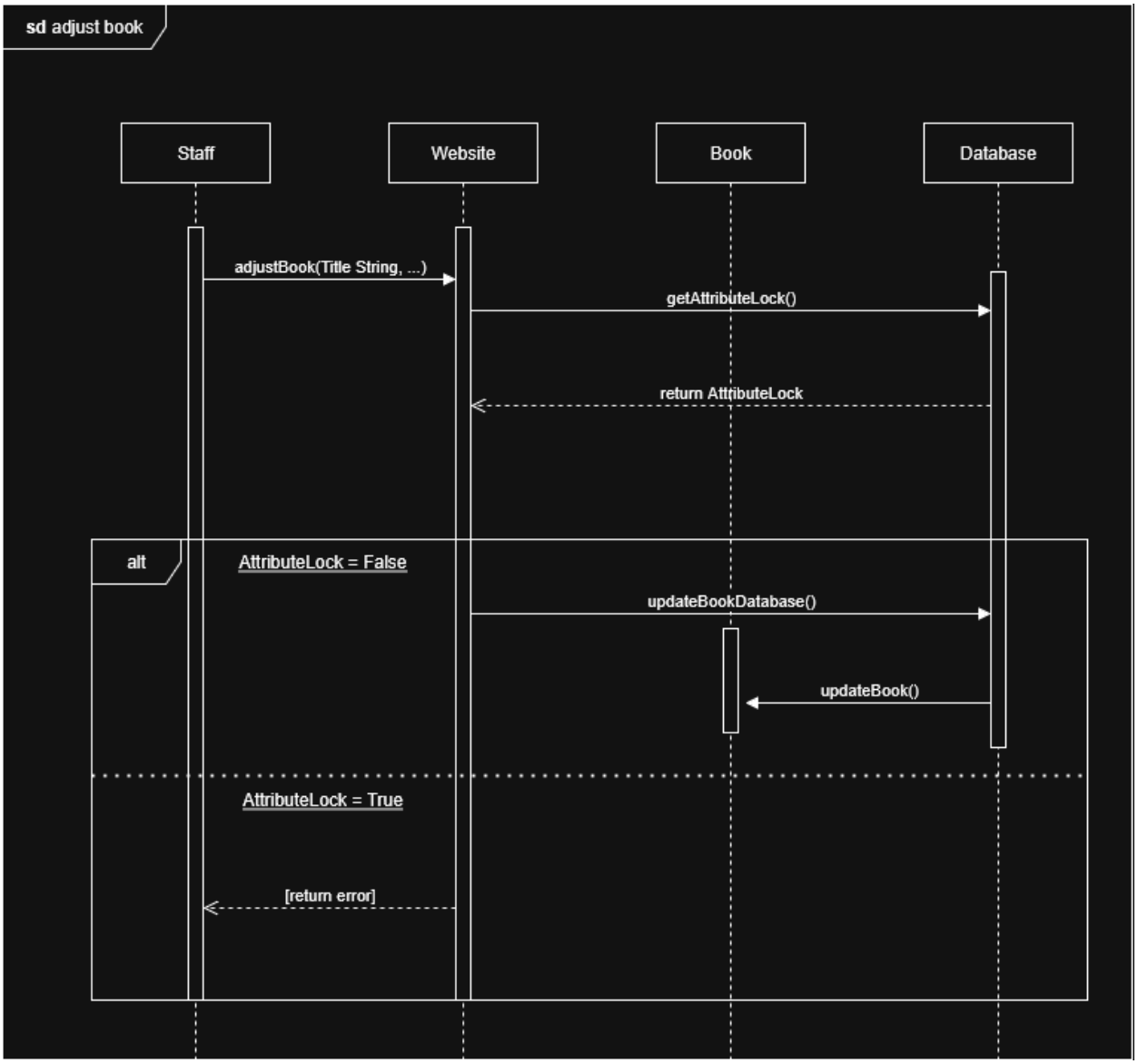
- Actor: **Staff**
- Objects:
    - Website
    - Library Database
    - Media

Process

1. Staff selects a book from website to alter its attributes.

2. Staff inputs changed attributes.
3. Website checks attributeLock status.
4. If attributeLock is false: Website send changed attributes to Database, and Database stores them.
5. If attributeLock is true: Website return error and process is halted.

Diagram



Activity Diagrams

Create Staff Account

States

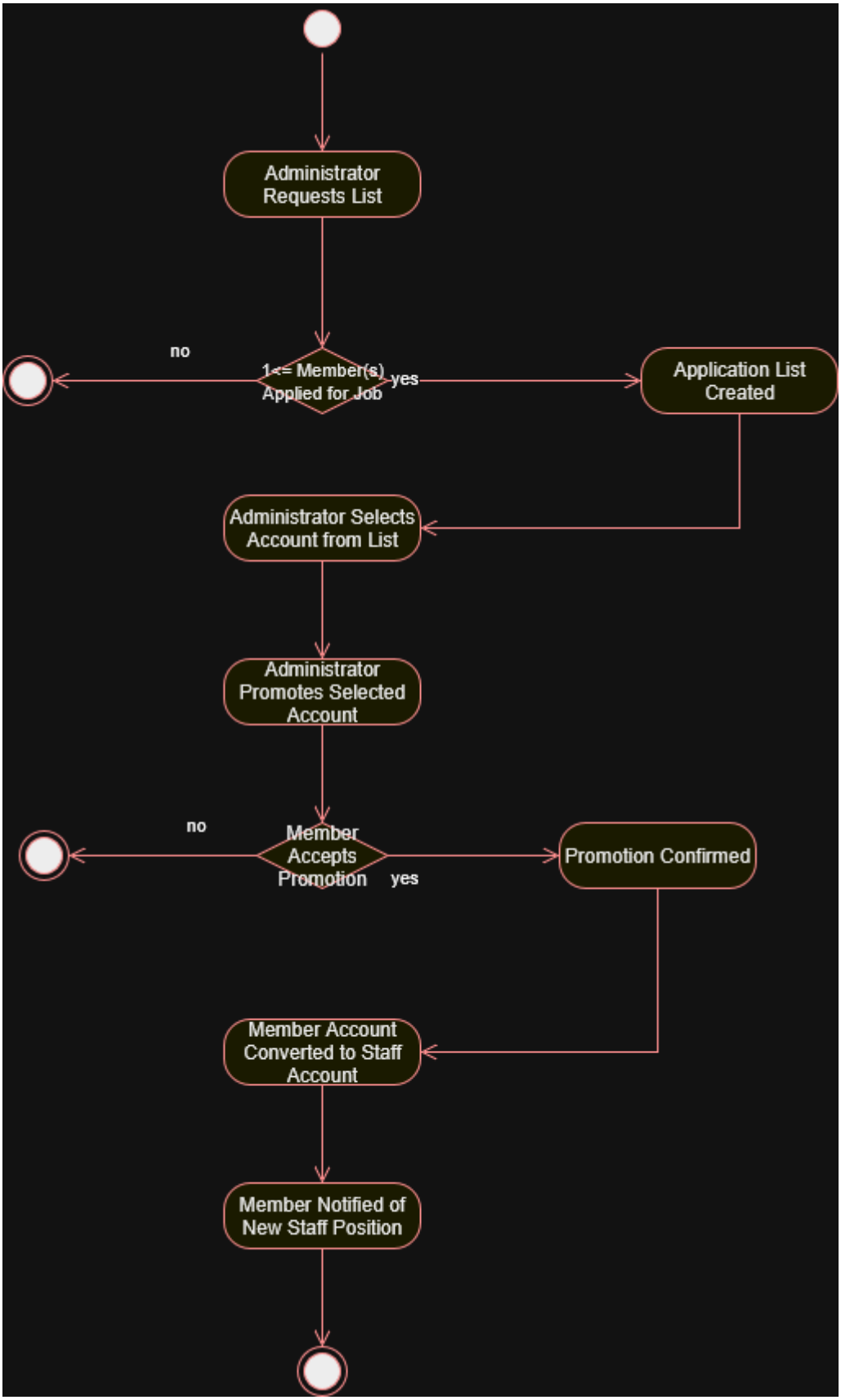
- Initial State: A member account is ready to be converted into a staff account.

- Final State:
    - No member accounts have sent in staff applications. Creation failed.
    - Member does not accept new staff position. Creation failed.
    - A member account becomes a staff account and gains access to its permissions.

Actions

1. The website retrieves a list of member accounts from the database that have applied for a staff position.
2. The Administrator selects a specific member account from the list.
3. The Administrator promotes the member account to a staff account.
4. The database converts the member account into a staff account.
5. The member is notified of their new position as a staff account.

Diagram


Administrator Requests List
no
1<= Member(s) Applied for Job
yes
Application List Created
Administrator Selects Account from List
Administrator Promotes Selected Account
no
Member Accepts Promotion
yes
Promotion Confirmed
Member Account Converted to Staff Account
Member Notified of New Staff Position

Filter Books

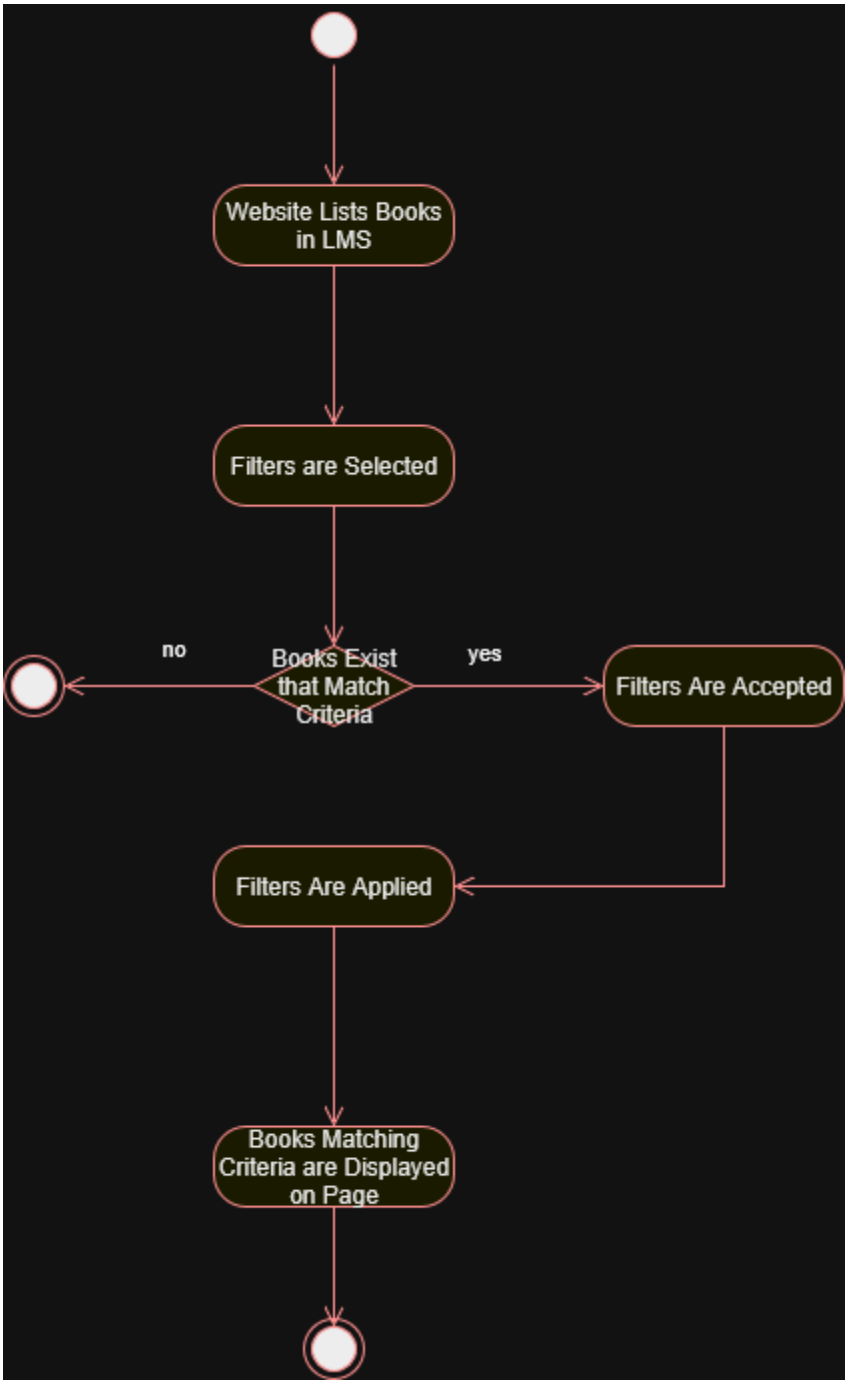
States

- Initial State: An account selects filters to apply to the book list on the website.
- Final State:
    - Books matching the filters selected are shown.
    - The page is empty to show that there are no books that match the criteria.

Actions

1. The website shows a list of books available at the library.
2. An account selects filters to apply to the list of books.
3. Filters are applied, selecting only the books that match the input criteria.
4. The books are shown on the website.

Diagram


Website Lists Books in LMS
Filters are Selected
Books Exist that Match Criteria
no
yes
Filters Are Accepted
Filters Are Applied
Books Matching Criteria are Displayed on Page

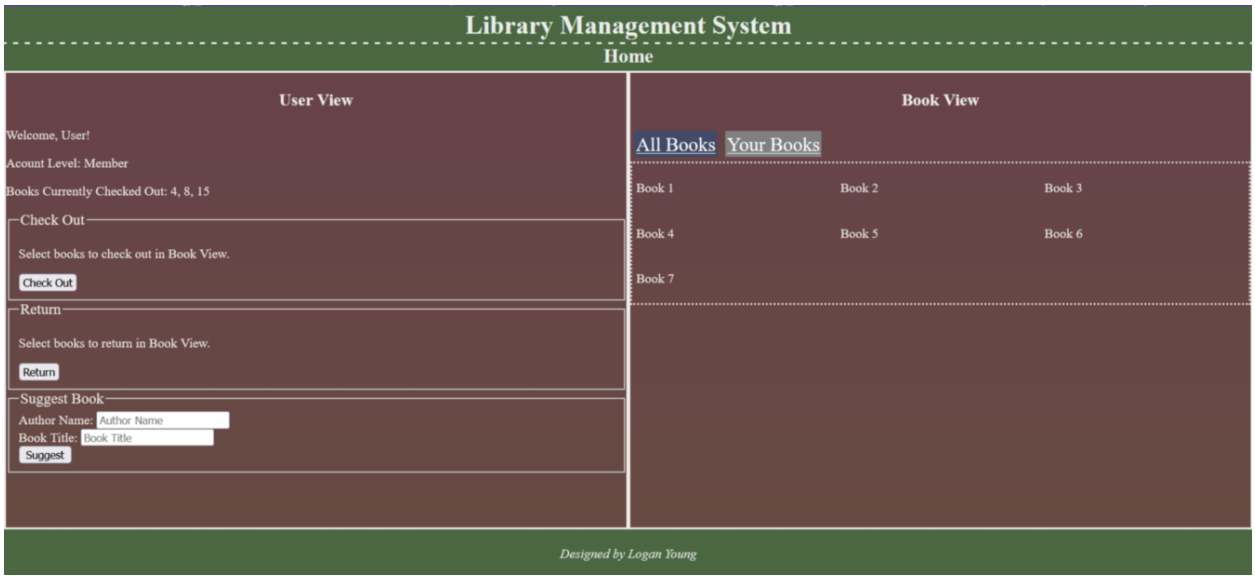
User Interface Specification

Logan Young

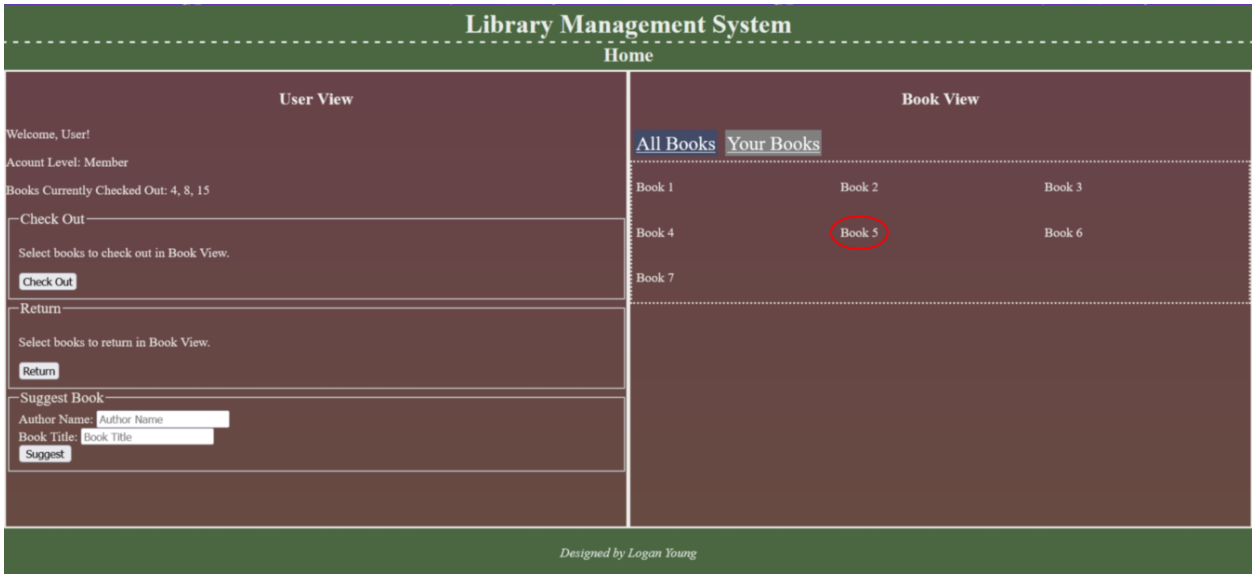
Preliminary Design

Use Case 1

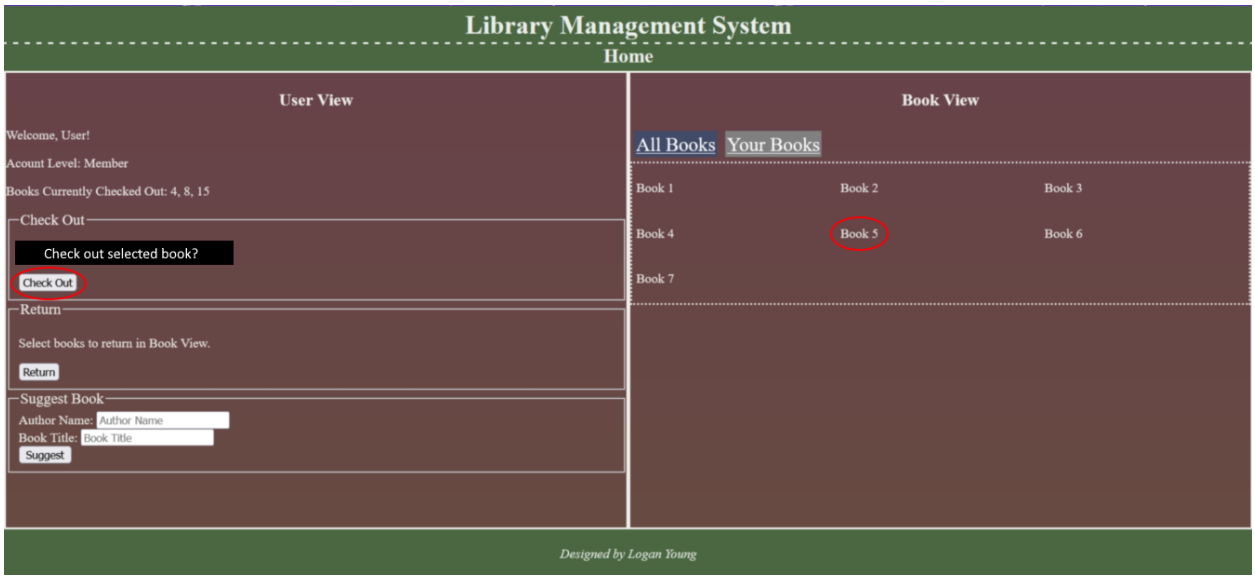
- A library member checks out a book.
  1. Home page.



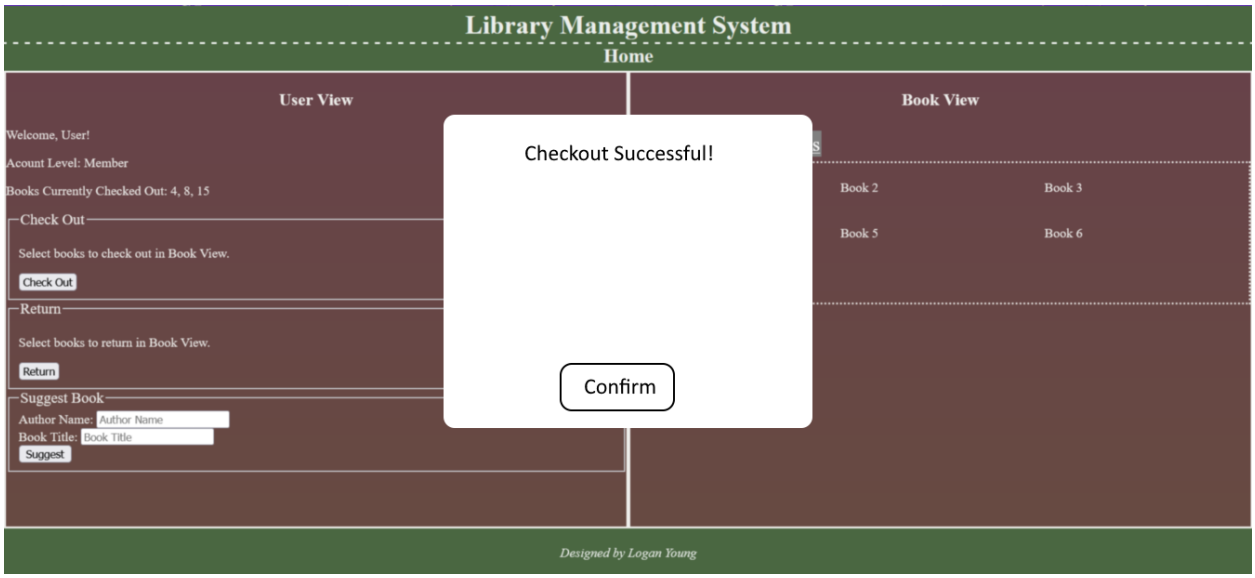
  2. Select book.



3. Select check out button.



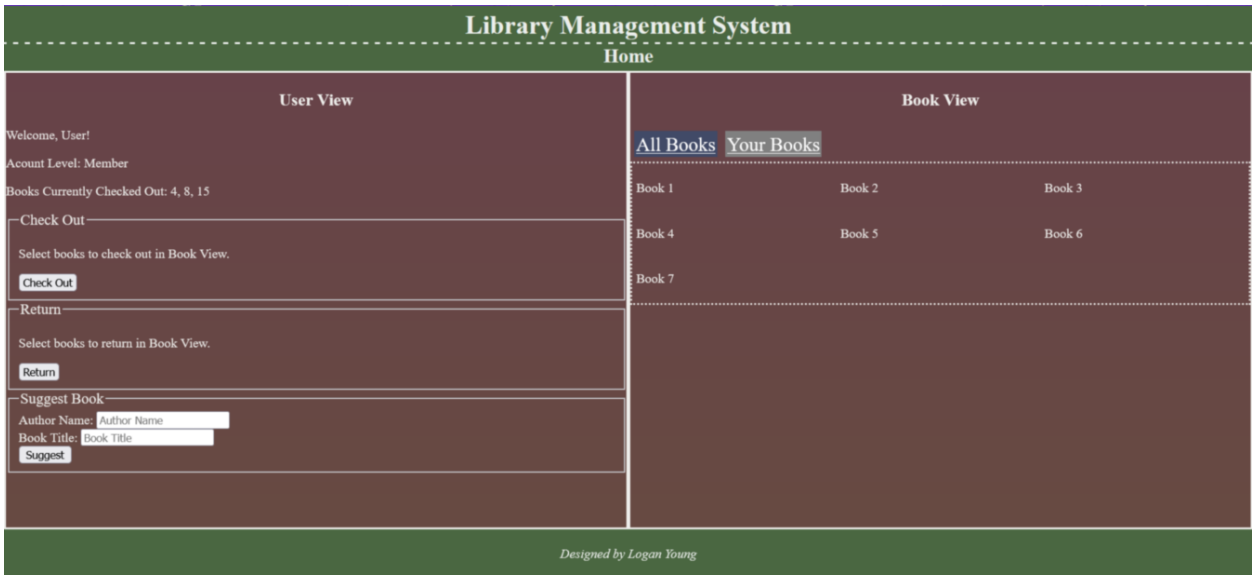
4. Confirmation.



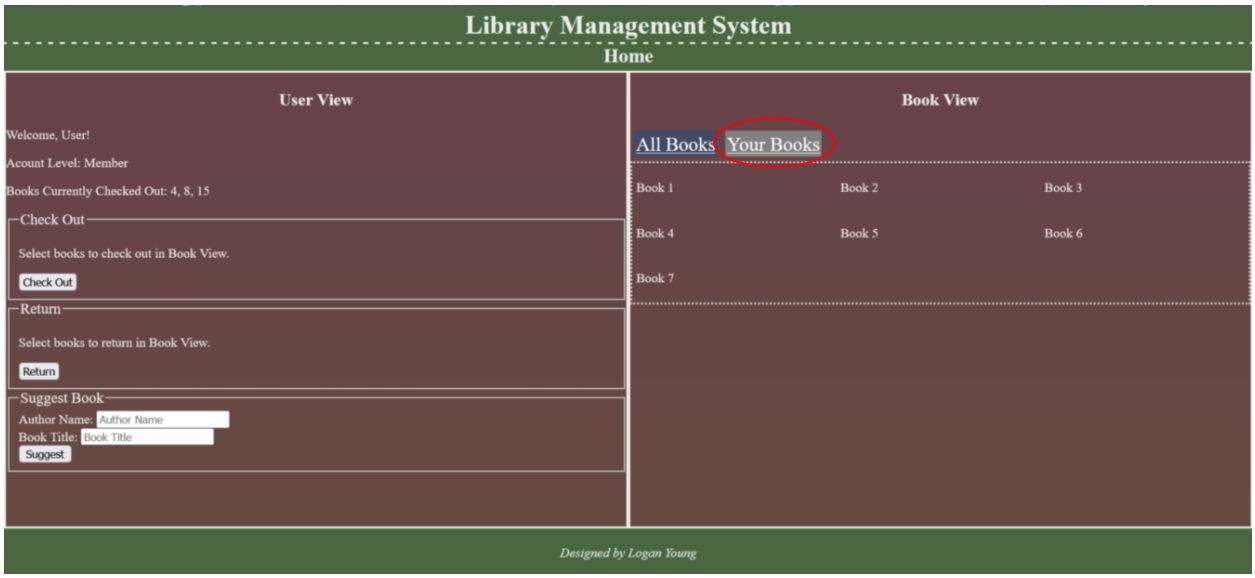
Use Case 2

- A library member returns a book and then views their current checked-out books.

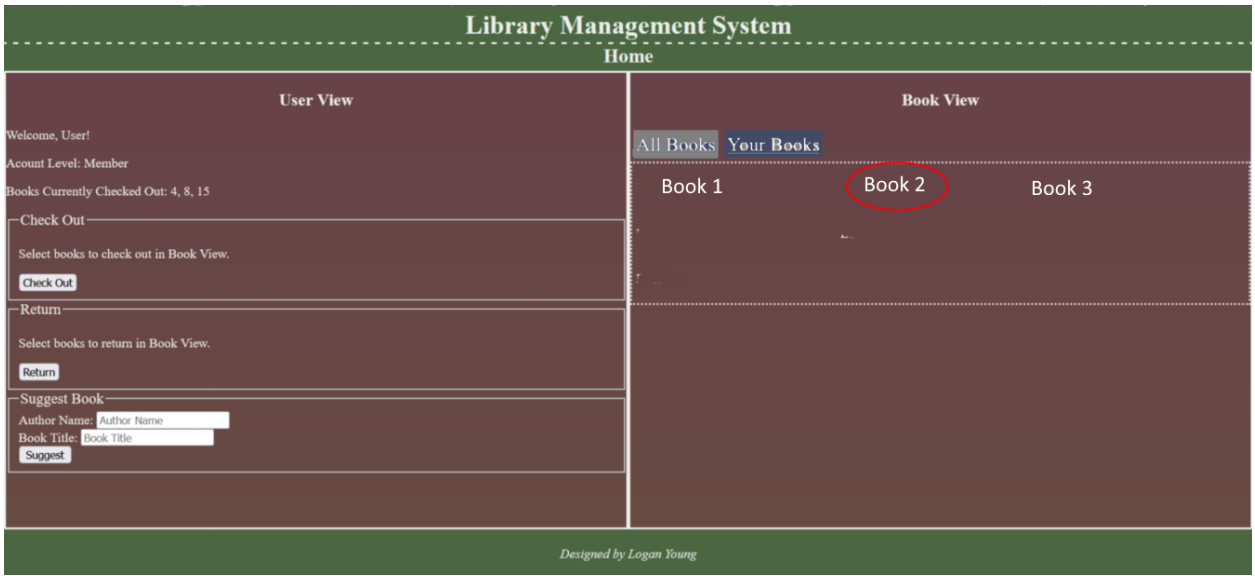
1. Home page.



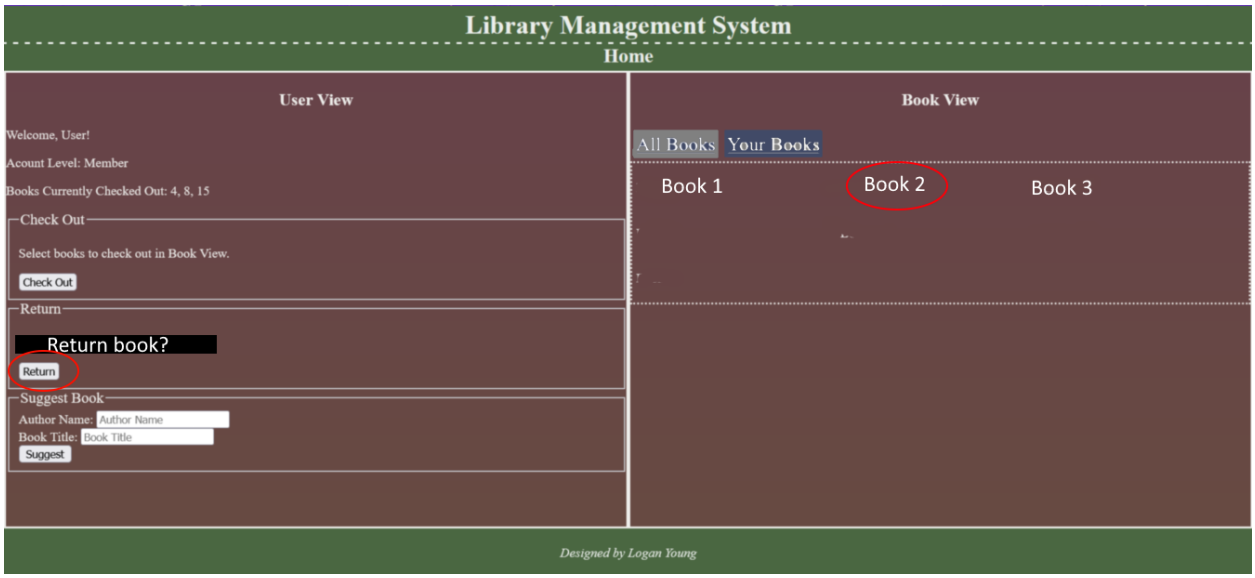
2. Select Your Books section.



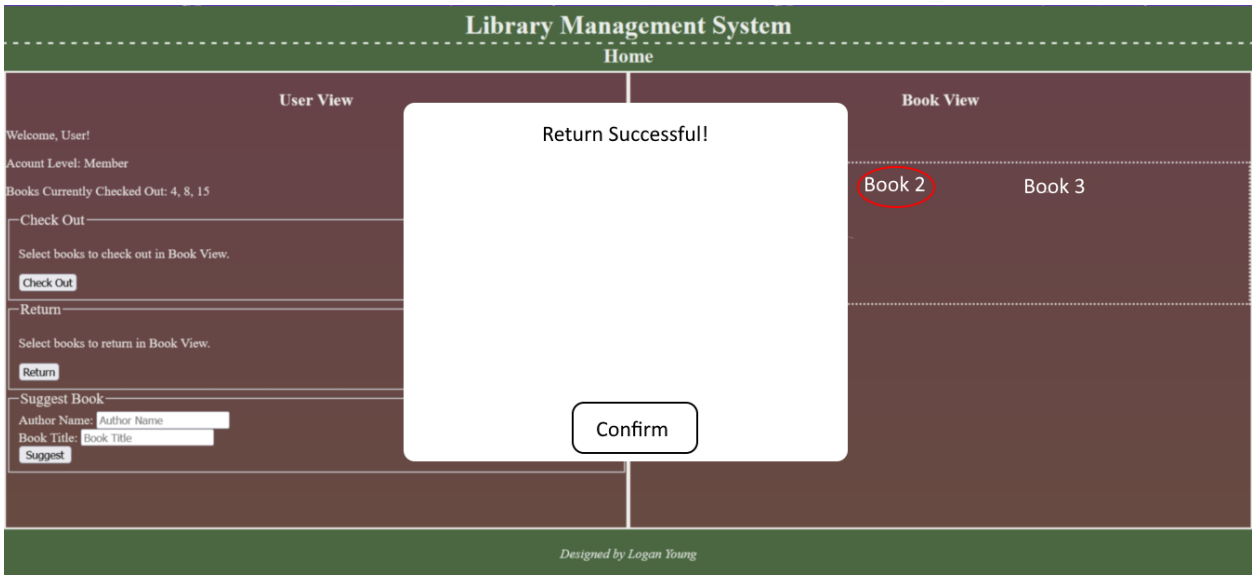
3. Select a book to return.



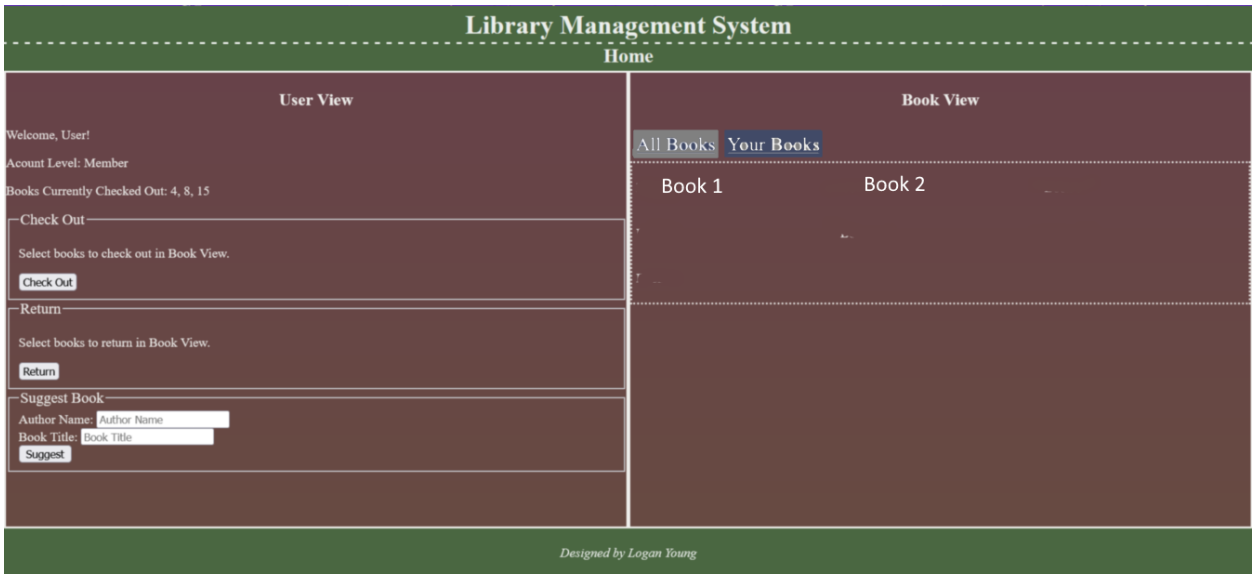
4. Click return button.



5. Confirmation.



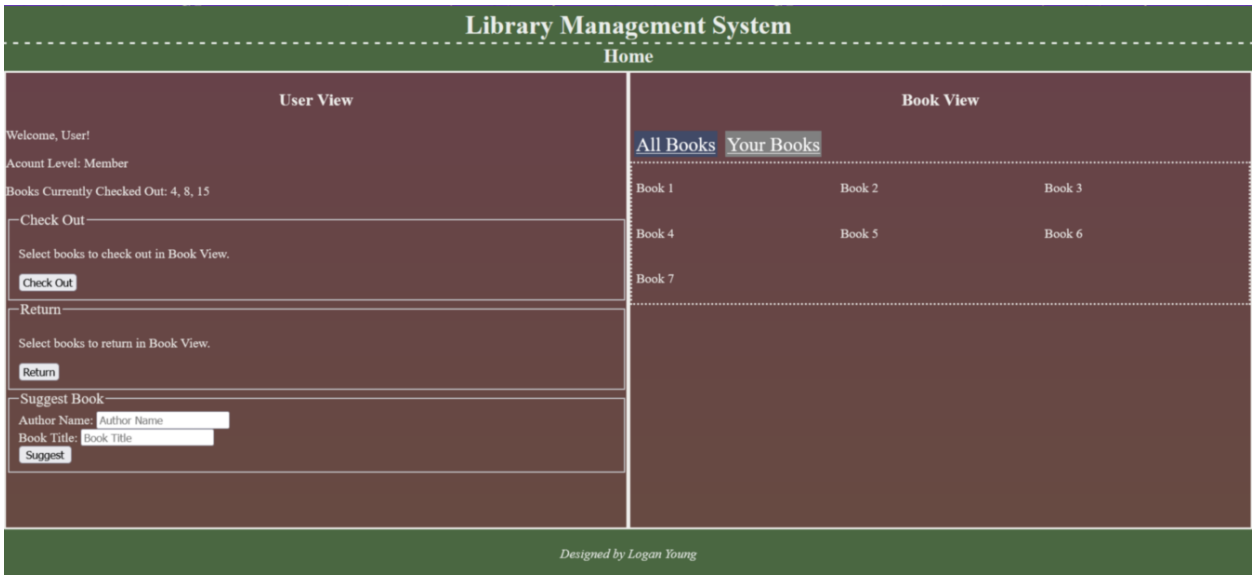
6. Return to book view.



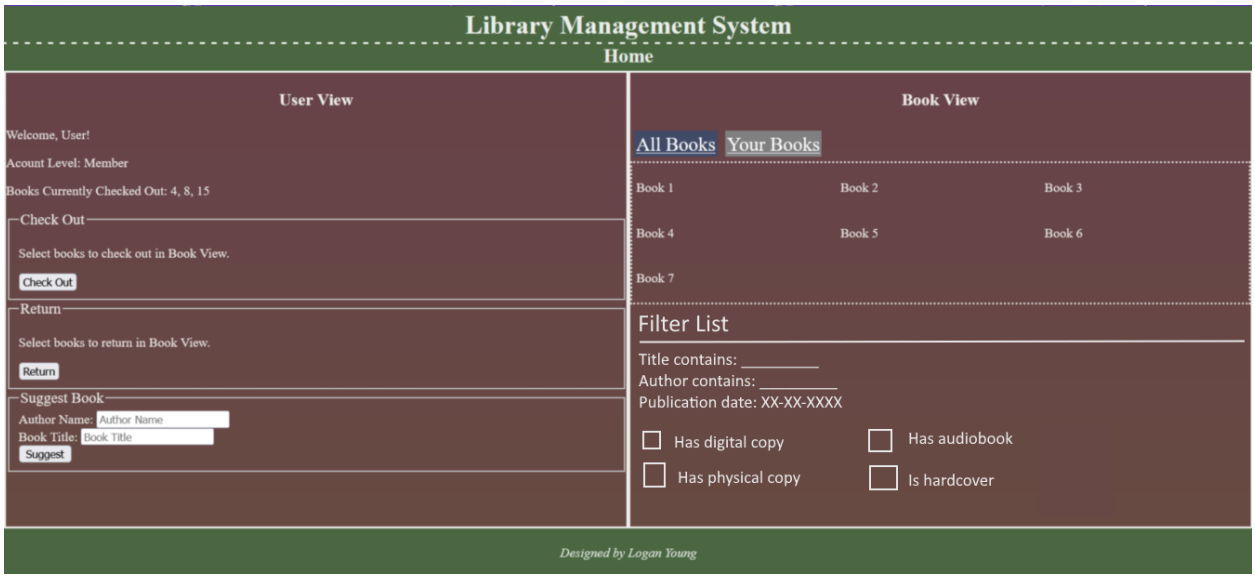
Use Case 3

- A user filters books by selected criteria.

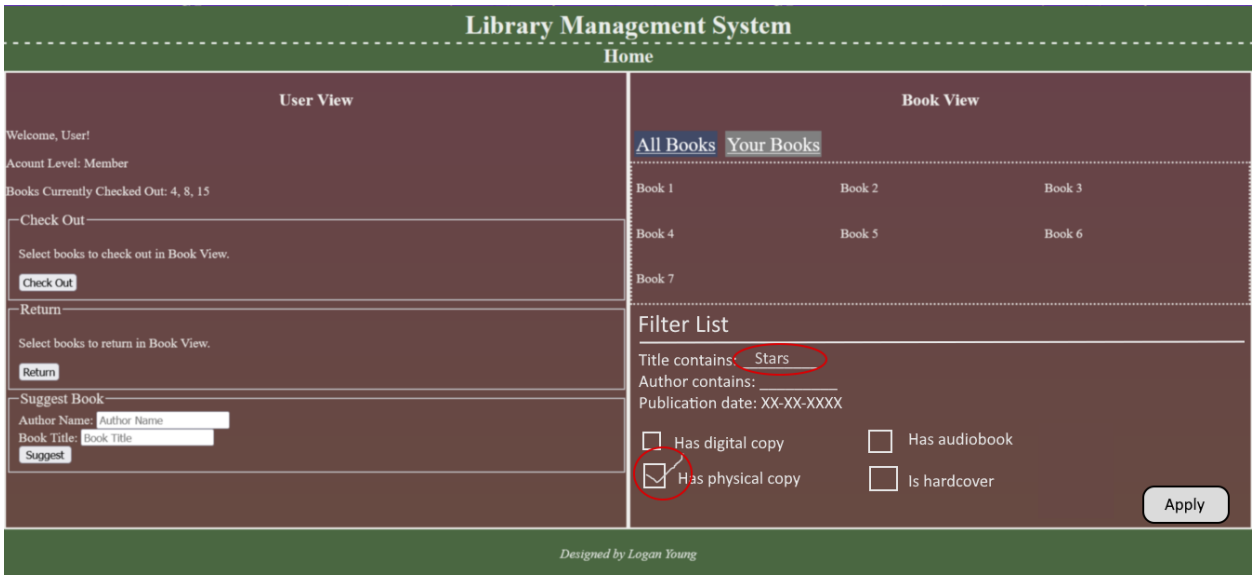
1. Home page.



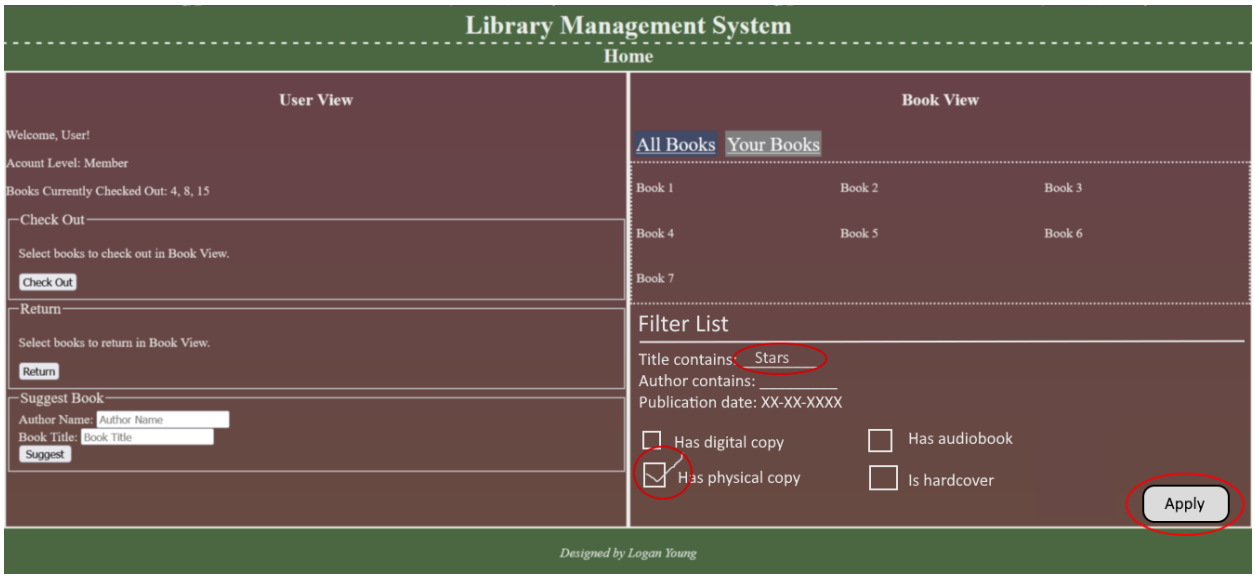
2. Select filter menu (will appear as expandable menu below Books sections).



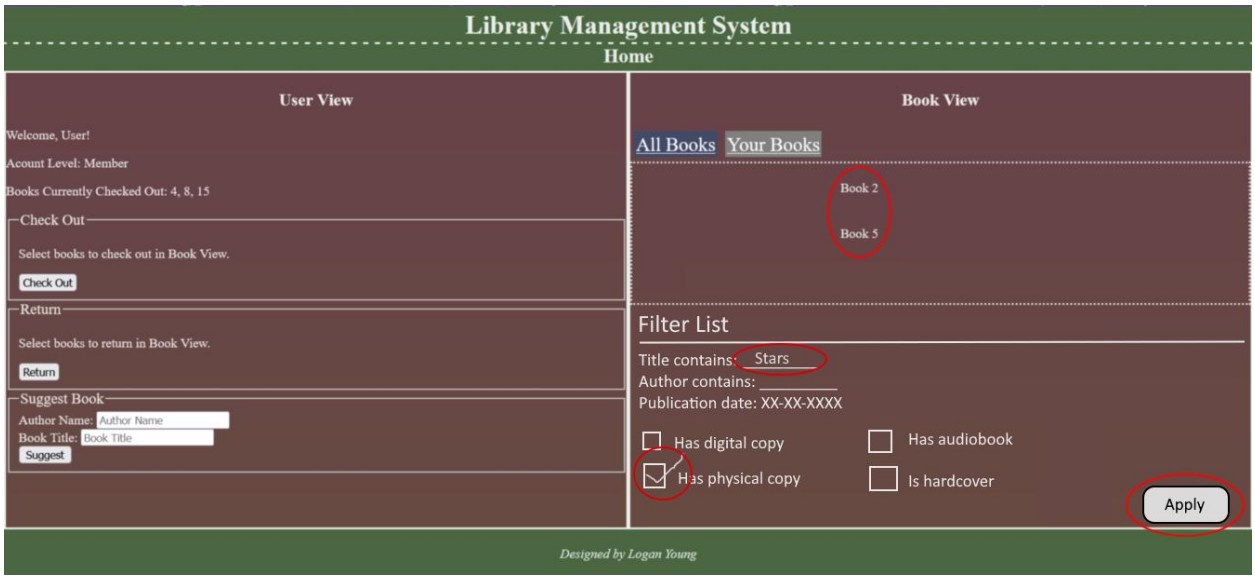
3. Choose filters.



4. Click apply.

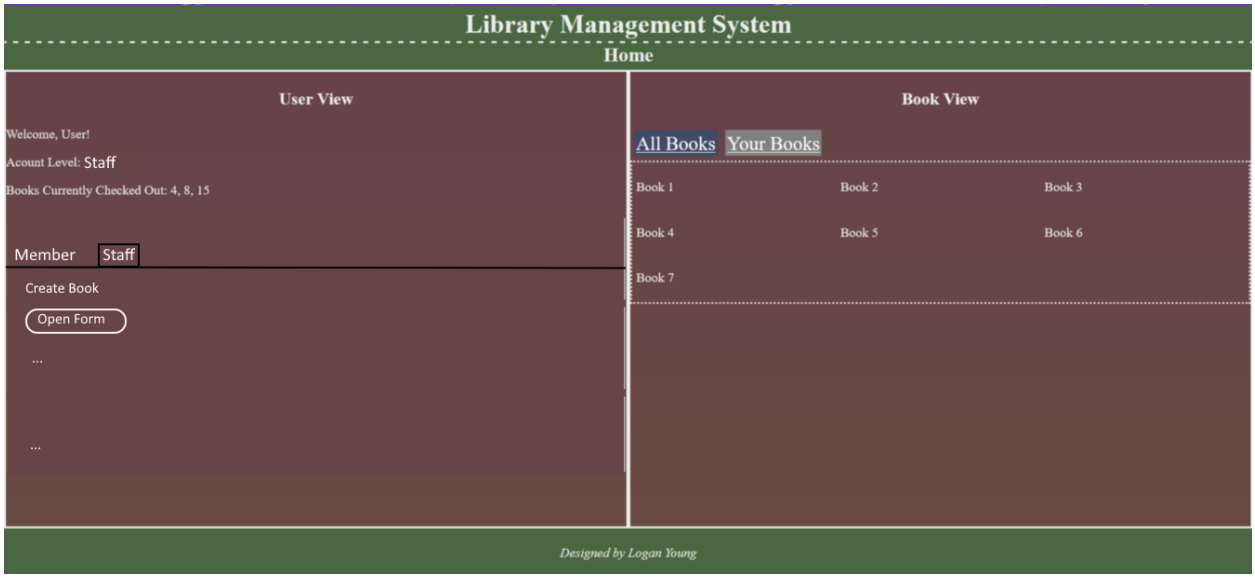


5. Filtered books.

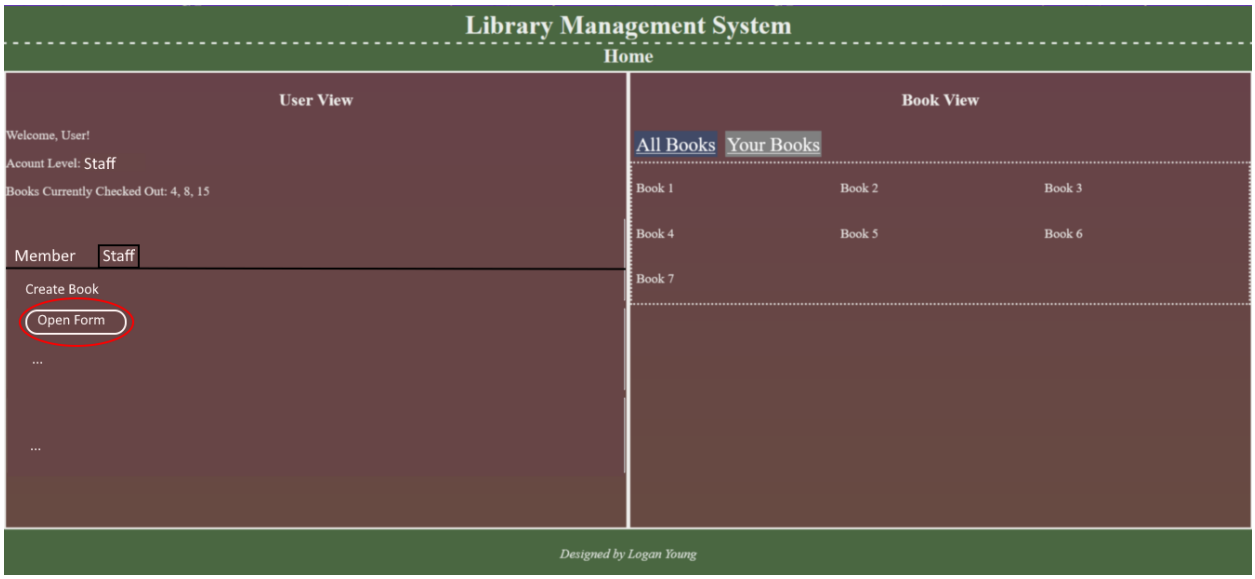


Use Case 4

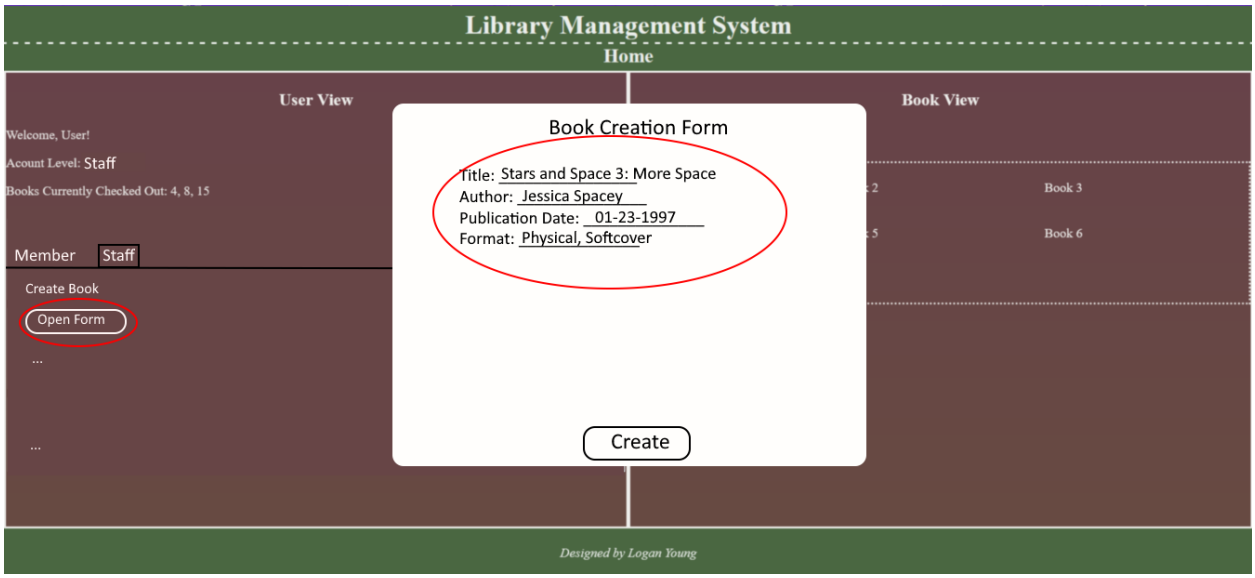
- A staff member adds a book.
  1. Home page.



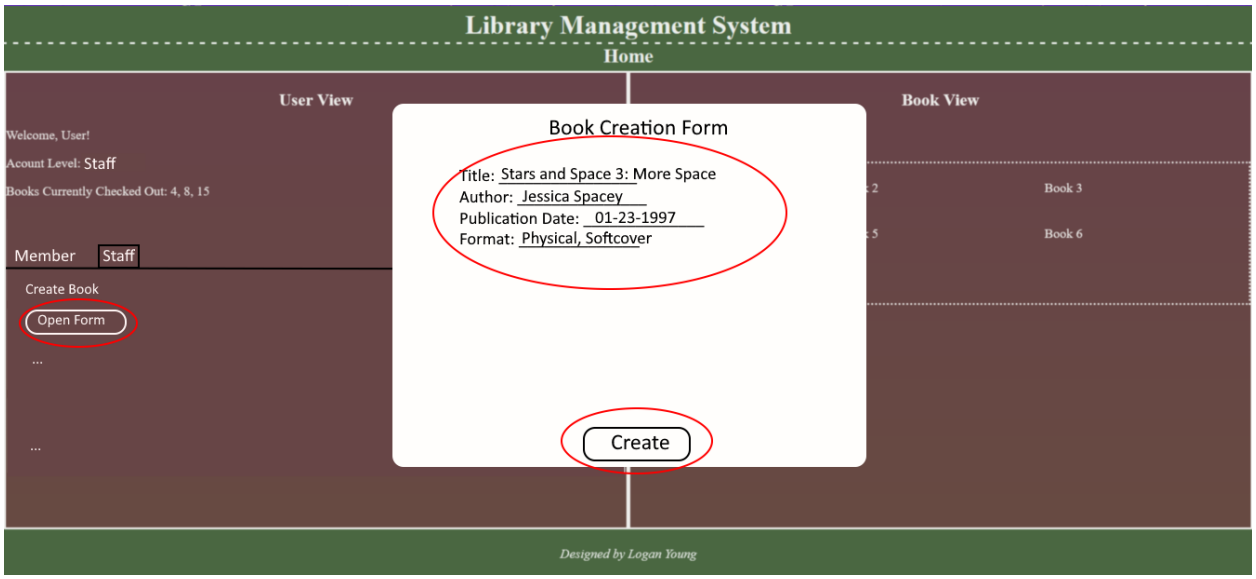
2. Select Open Form.



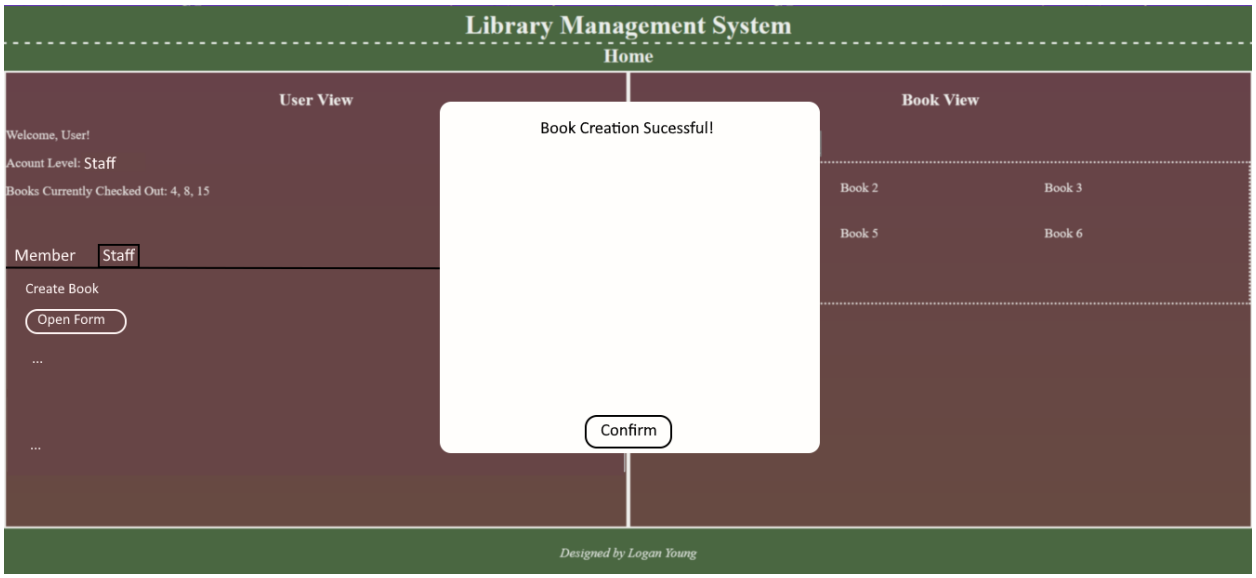
3. Fill out form for book creation.



4. Select Create.



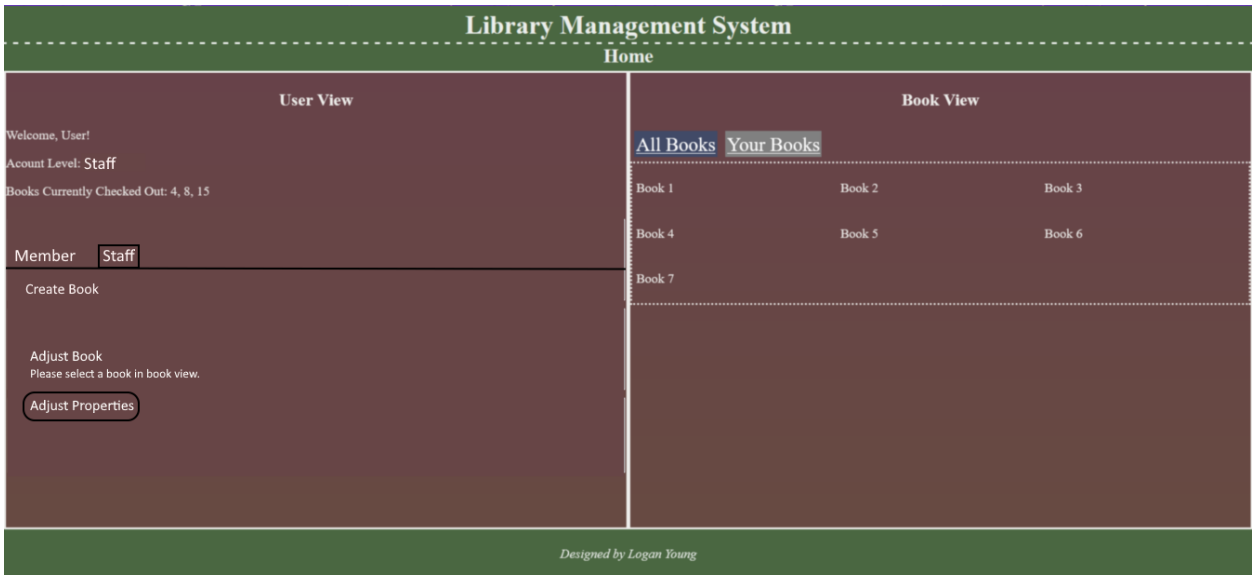
5. Confirmation.



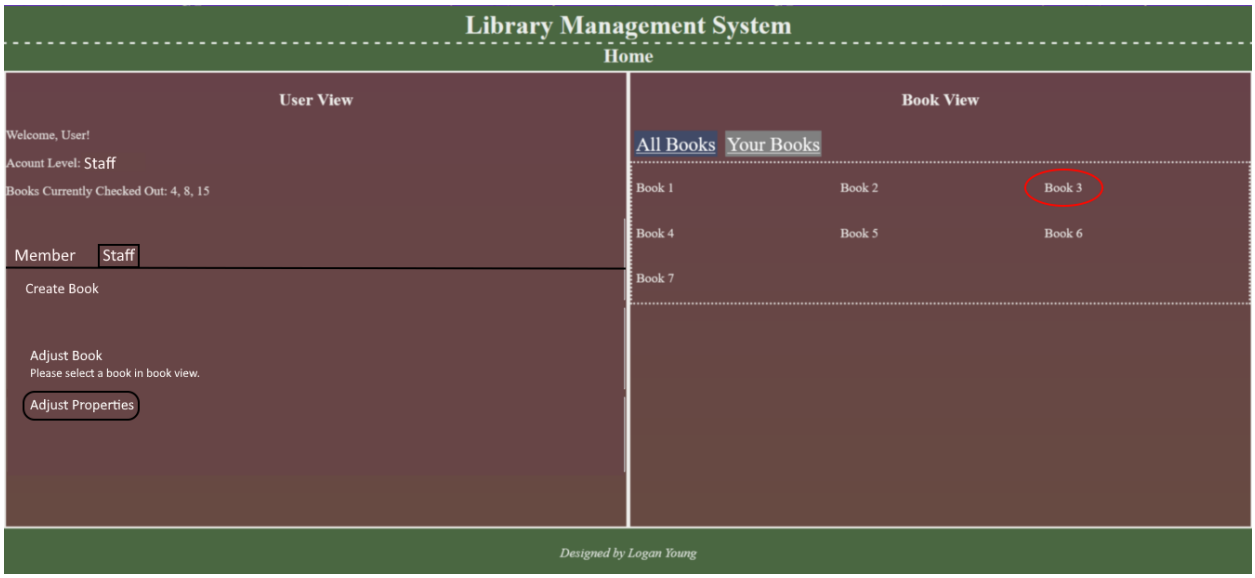
Use Case 5

- A staff member adjusts a book's properties.

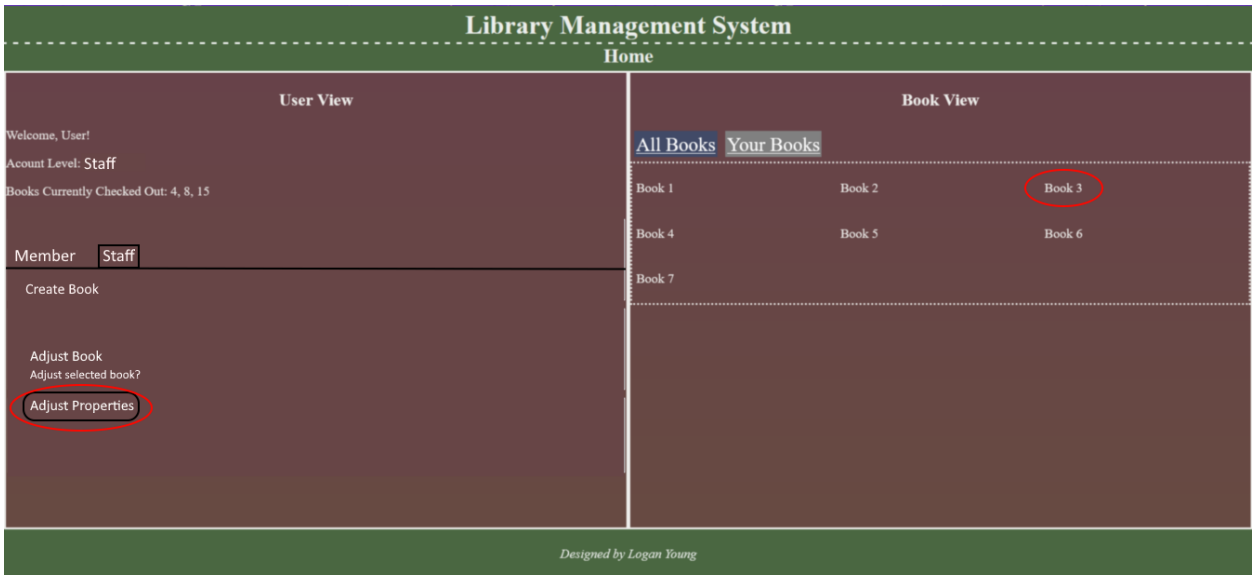
1. Home page.



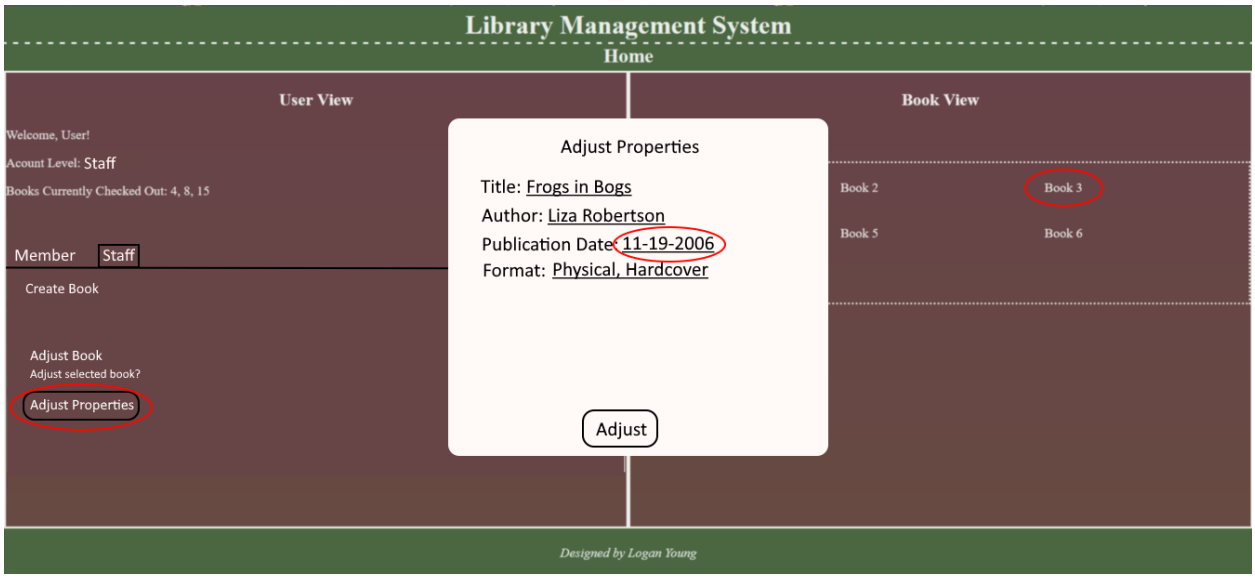
2. Select book.



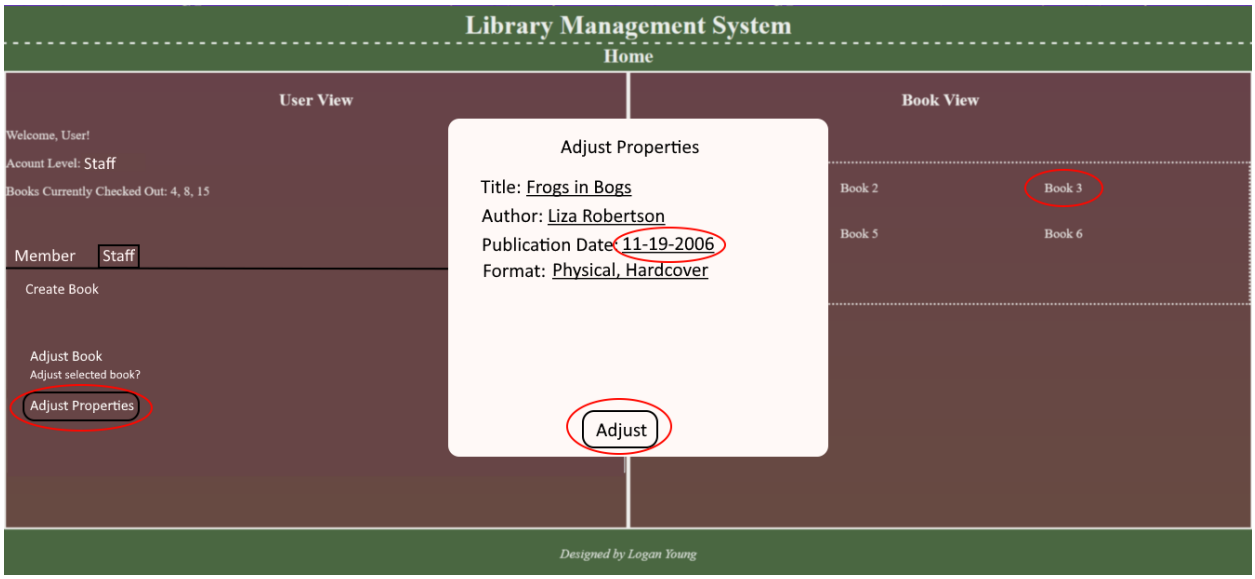
3. Click Adjust Properties.



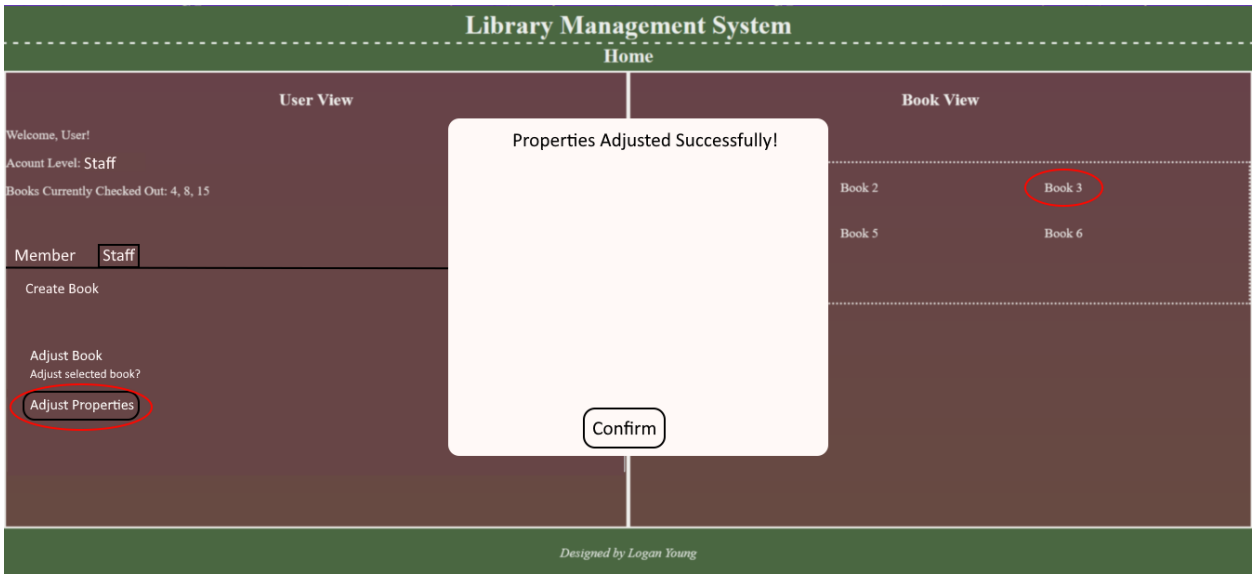
4. Adjust properties in form.



5. Click Adjust.



6. Confirmation.



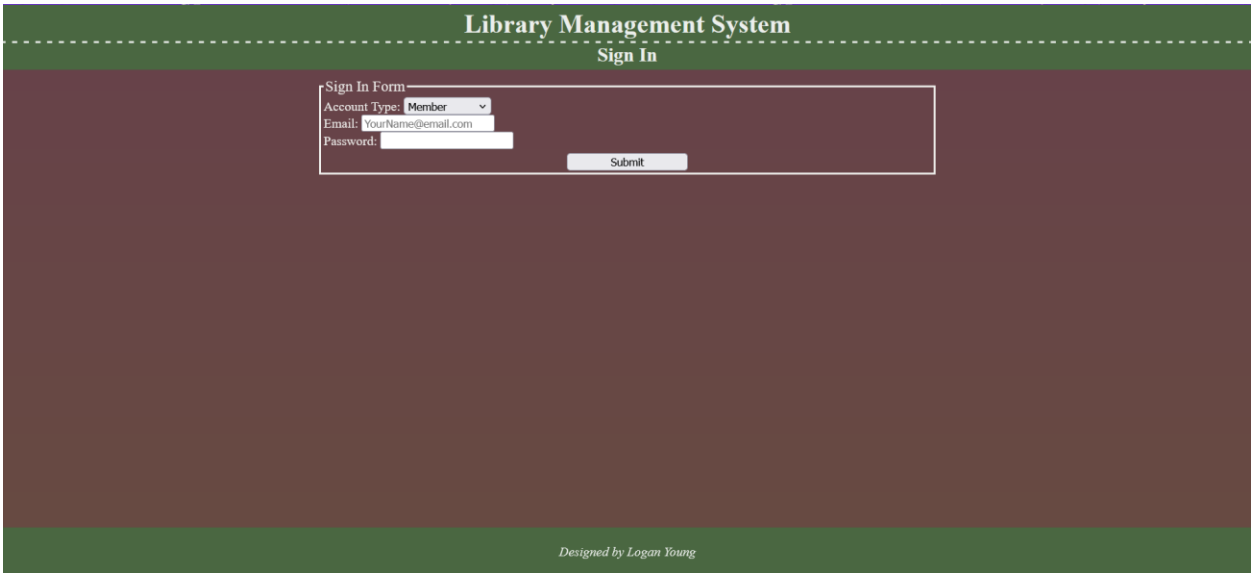
User Effort Estimation

Usage Scenario	Navigation	Clicks	Keystrokes
Member checks out a book	Home -> Book View -> User View -> Confirmation	3	0
Member returns a book and views current checked out books	Home -> Book View-> Confirmation -> Home -> Book View	5	0
Staff adjusts a book's properties	Home -> Book View -> User View -> Popup Form -> Confirmation	>=5	100>=x>=1

Staff adds a book	Home -> User View -> Popup Form -> Confirmation	>=4	100>=x>=10
User filters books	Home -> Book View	>=3	>=0

Additional Notes

- Please excuse the poor photoshopping done on top of my website screenshots.
- Like the sample example, each use case assumes that the user has already signed in. Below is an early work-in-progress example of the sign in form.



- The base home page and sign in page are currently being worked on, but they are fully coded in HTML and CSS. JavaScript and node.JS functionality are coming soon. Might support JavaScript frontend with jQuery as I progress.
- The appearance of the site and UI placement are subject to change, but the general functionality will remain the same.

Traceability Matrix and Updates

Logan Young

Updated Functional Requirements

No.	Priority Weight (1-5; 1: lowest)	Description
REQ-1	5	The LMS should be able to store and display books on the website.
REQ-2	5	Library members should be able to use the LMS to check out and return books.
REQ-3	3	Library members should be able to use the LMS to submit a hold request for a maximum of three books.
REQ-4	4	Library staff should be able to use the LMS to mark a book as unavailable.
REQ-5	3	Library members should be able to delete their account from the LMS.
REQ-6	4	Library staff should be able to delete a member's account using the LMS.
REQ-7	5	Library staff should be able to create a new member account for a user.
REQ-8	5	Library staff should be able to adjust a book's attributes if incorrect.
REQ-9	5	Library staff should be able to create a new entry in the LMS for a newly received book.
REQ-10	2	Library members should be able to request a book from a different library in the same network.
REQ-11	4	Library administrator should be able to convert a new user into staff.
REQ-12	4	Library administrator should be able to convert staff into members or delete accounts entirely.
REQ-13	2	Library members should be able to submit a suggestion for new books for the library.
REQ-14	1	Library members should be able to create a new donated book entry to be approved by staff.
REQ-15	5	LMS users should be able to easily search for a specific book using various filters and sorting techniques.
REQ-16	5	LMS users should be able to log in to personal accounts using email and passwords from the website and log out.
REQ-17	2	LMS users should be able to activate 2-factor authentication for their accounts from user settings.
REQ-18	3	Library administrators should be able to completely delete a book from the database.

Updated Use Cases

No.	Name	Description
UC-1	Log In	To let a user log in to their account.

UC-2	Log Out	To let a user log out of their account.
UC-3	Toggle 2FA	To let a user toggle the status of 2-factor authentication.
UC-4	Check Out	To allow a member to check out a book.
UC-5	Return	To allow a member to return a book.
UC-6	Hold Request	To allow a member to submit a hold request for a book.
UC-7	Procurement Request	To allow a user to submit a request for a book to a different library branch.
UC-8	Delete Account	To allow a user to delete an account.
UC-9	Create Account	To allow a user to create an account.
UC-10	Convert Account	To allow a user to downgrade or upgrade an account's permission level.
UC-11	Submit Suggestion	To submit a suggestion to the library for a new book.
UC-12	Donate Book	To allow a member to donate a book to the library.
UC-13	Create Book	To allow staff to create a new book.
UC-14	Adjust Book	To allow staff to adjust a book's properties.
UC-15	Hide Book	To allow staff to hide a book.
UC-16	Delete Book	To allow administrator to delete a book.
UC-17	Search Books	To allow a user to search for a specific book.
UC-18	Filter Books	To allow a user to filter books based on supplied criteria.

Traceability Matrix

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15	UC16	UC17	UC18
REQ1	5																		
REQ2	5	X			X	X													

REQ3	3	X					X												
REQ4	4	X													X				
REQ5	3	X							X										
REQ6	4	X							X										
REQ7	5	X								X									
REQ8	5	X												X					
REQ9	5	X											X						
REQ10	2	X						X											
REQ11	4	X									X								
REQ12	4	X							X	X	X								
REQ13	2	X										X							
REQ14	1	X											X						
REQ15	5	X																X	X
REQ16	5	X	X																
REQ17	2	X		X															
REQ18	3																X		
Max PW		5	5	2	5	5	3	2	4	5	4	2	1	5	5	4	3	5	5
Total PW		59	5	2	5	5	3	2	11	9	8	2	1	5	5	4	3	5	5

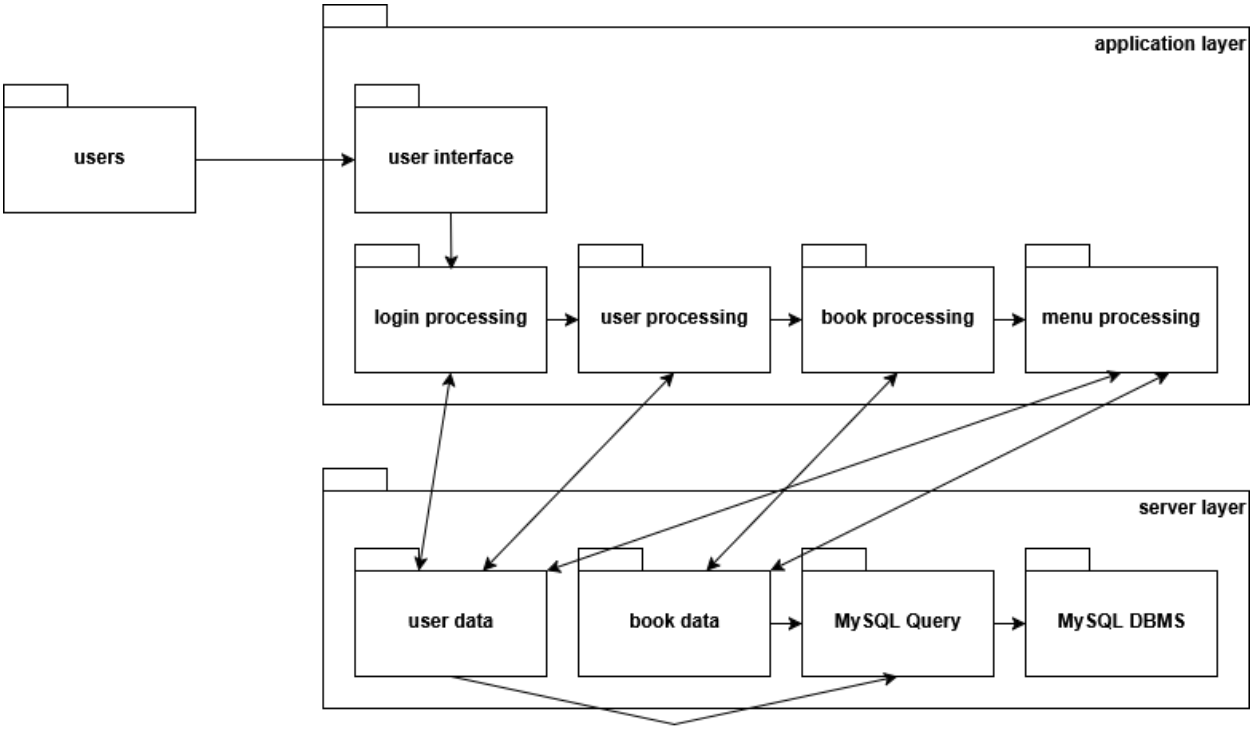
System Architecture and System Design

Architectural Styles

The Library Management System uses a client-server architectural style. The client is a computer operated by an end user, and the server is a computer solely used for running the website. The server would be located in the real library. I'm using a combination of HTML, CSS, and JS for the frontend; NodeJS and Express for the backend; and MySQL for the database. (I'm currently deciding on another language to send information from the client to the server.) The frontend displays the website and allows the user to perform some actions to solicit a response. The backend communicates between the frontend and the database. The database (MySQL) stores all information used for the application: user data, book data, etc. As of now, both the client and the server run on the same computer when you use the application, but in production, the client would be on a separate device from the server (which would be running from the library).

Identifying Subsystems

Shown below is a UML package diagram of the Library Management System. The application layer represents the frontend, while the server layer represents the backend and database. The process begins with the user accessing the user interface package. From there, the user interface package contains four processes (what could also be defined as subsystems): login processing, user processing, book processing, and menu processing. Login processing accesses user data in the server layer and controls how the user logs in and out. User processing accesses user data and controls the various aspects of user account details. Book processing accesses book data and controls all book operations. Menu processing accesses both user data and book data and controls UI displays and changes. The data in the server layer are modified after querying the MySQL database.



Mapping Subsystems to Hardware

All subsystems are run on one server computer and one client computer.

Persistent Data Storage

There are two main objects that are stored globally: users and books. Both of these are stored as relations in a MySQL database, saved to the server. Other secondary objects are also stored as relations, such as “suggested books”.

Network Protocol

There will only be one server computer.

Global Control Flow

Execution Orders

My system is event-driven, that is, users can perform different actions in different orders from other users. The LMS does this through forms and buttons. For example, one user could check out a book and then suggest a book, while another user could return a book with no further actions. In standard operation, a user would click the icon of a book in the “Book View” and then use the “User View” to check out that selected book.

Time Dependency

The LMS is event-driven, and my implementation does not depend on real time or any sort of timer. However, one potential feature is a “Book of the Day”, which would advertise a random book from the database each day. This would change in real time, most likely at midnight in the server’s time zone. Other than that, the only waiting is done in periods before the user inputs some action.

Concurrency

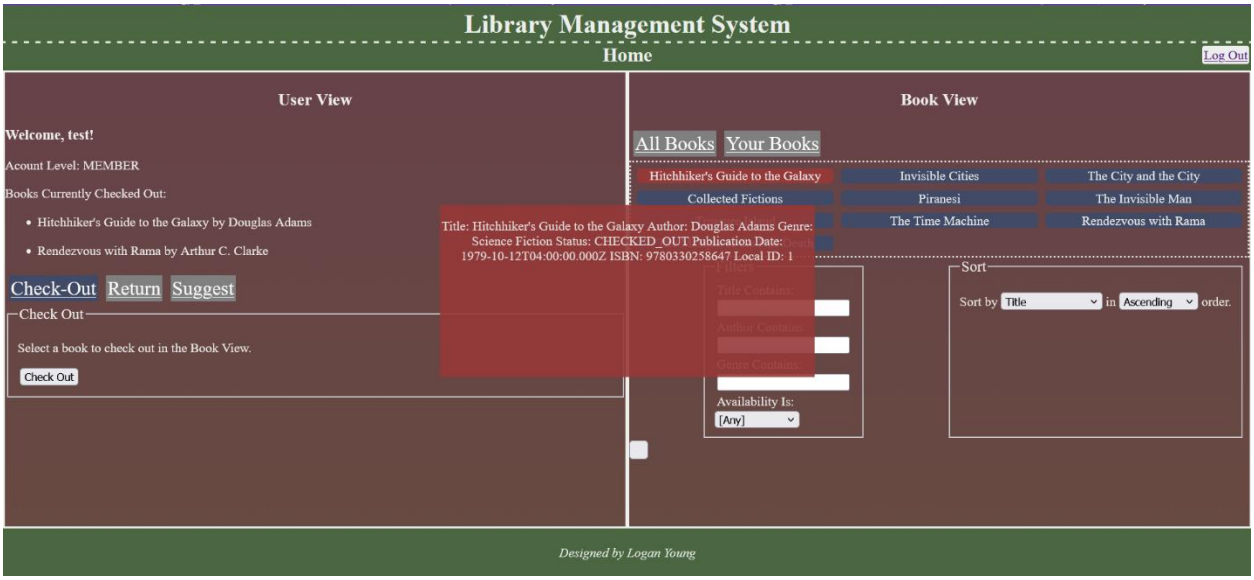
The LMS is single threaded.

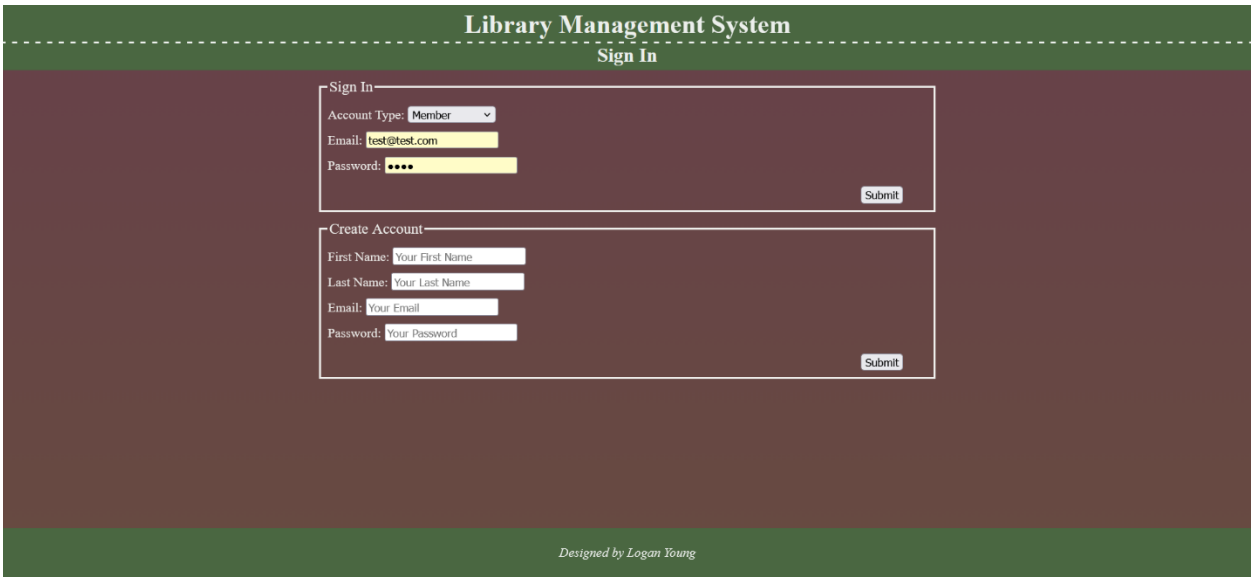
Hardware Requirements

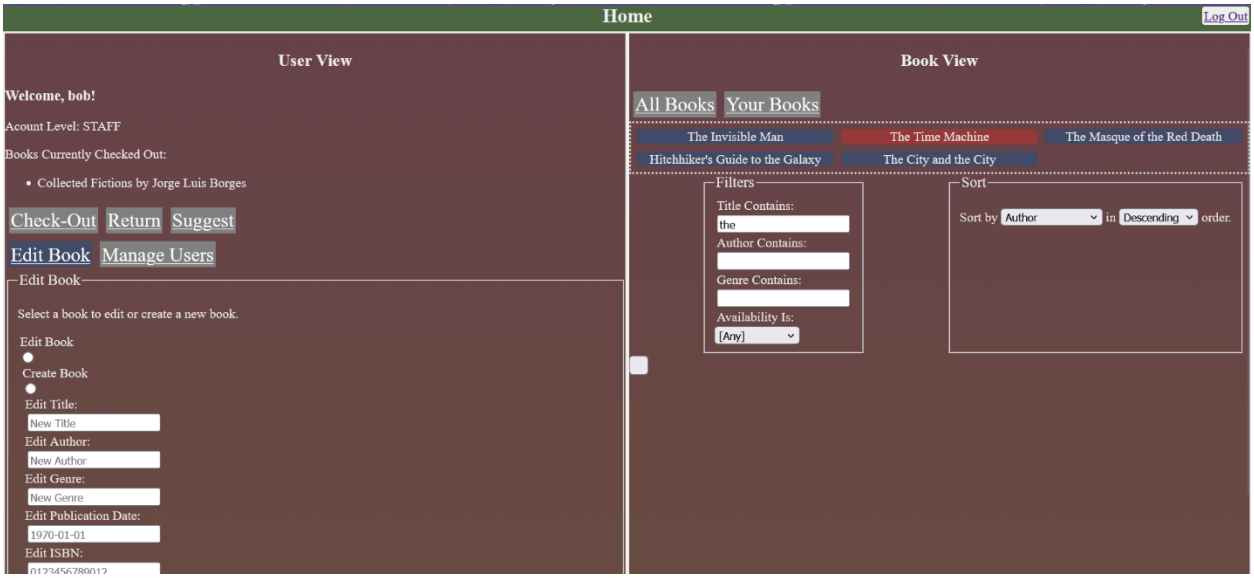
- Most components of the user interface are scrollable, so only a standard resolution of 640x480 pixels is required.
- The user needs a device with a user interface capable of connecting to the internet.
- 1 GB of memory should be sufficient for processing operations.
- 10 Mbps upload/download network speed should be sufficient to send and receive data from the LMS.

User Interface Design and Implementation

The final user interface was designed with ease of use in mind, not aesthetics. A dual-pane window is used for the main page so that user and book tasks are separated. For some tasks, the user can select a book in the book view and then perform some operation in the user view.


Library Management System
Home
Log Out
User View
Welcome, test!
Acount Level: MEMBER
Books Currently Checked Out:
• Hitchhiker's Guide to the Galaxy by Douglas Adams
• Rendezvous with Rama by Arthur C. Clarke
Check-Out Return Suggest
Check Out
Select a book to check out in the Book View.
Check Out
Book View
All Books Your Books
Hitchhiker's Guide to the Galaxy
Invisible Cities
The City and the City
Collected Fictions
Piranesi
The Invisible Man
The Time Machine
Rendezvous with Rama
Title: Hitchhiker's Guide to the Galaxy Author: Douglas Adams Genre: Science Fiction Status: CHECKED_OUT Publication Date: 1979-10-12T04:00:00.000Z ISBN: 9780330258647 Local ID: 1
Filters
Title Contains:
Author Contains:
Genre Contains:
Availability Is:
[Any]
Sort
Sort by Title in Ascending order.
Designed by Logan Young


Library Management System
Sign In
Sign In
Account Type: Member
Email: test@test.com
Password: ••••
Submit
Create Account
First Name: Your First Name
Last Name: Your Last Name
Email: Your Email
Password: Your Password
Submit
Designed by Logan Young


Home
Log Out
User View
Welcome, bob!
Acount Level: STAFF
Books Currently Checked Out:
• Collected Fictions by Jorge Luis Borges
Check-Out Return Suggest
Edit Book Manage Users
Edit Book
Select a book to edit or create a new book.
Edit Book
Create Book
Edit Title:
New Title
Edit Author:
New Author
Edit Genre:
New Genre
Edit Publication Date:
1970-01-01
Edit ISBN:
Book View
All Books Your Books
The Invisible Man
The Time Machine
The Masque of the Red Death
Hitchhiker's Guide to the Galaxy
The City and the City
Filters
Title Contains:
the
Author Contains:
Genre Contains:
Availability Is:
[Any]
Sort
Sort by Author in Descending order.

Design of Tests

I did not have enough time to work on the LMS, so there are no runnable unit or integration tests. All tests done so far have been through “desk testing” and the debug runs I perform through the code myself. There are also numerous error handlers in the code if something goes wrong during operation.

Project Plan

- Weeks 1-2
    - Conceptualize and complete the system proposal.
- Weeks 3-4
    - Determine connection between frontend, backend, and database and start construction.
- Weeks 5-7
    - Build database to organize books and their attributes, build frontend webpage for users to interact with, build backend to connect the two.
- Week 8
    - Test system with sample entries and actions and record demo for midterm.
- Weeks 9-11
    - Improve system with feedback from professor. Implement suggested features and bug fixes. Implement rest of features left out from demo.
- Weeks 12-14
    - Write test cases and optionally add some cool last-minute features.
- Week 15 **(We are here.)**
    - Test final system and record demo for final.