Final Project: Library Management System

Emily Weed

Table of Contents

Customer Problem Statements & System Requirements

Problem Statement:

Public libraries are essential community resources, offering free access to various forms of media that can have a profound impact on people of all ages. While they play a crucial role in fostering literacy and lifelong learning, many public libraries continue to operate with outdated management systems. Both staff and patrons deserve a modern, streamlined system that is not only efficient but also user-friendly and accessible. A unified library management system that integrates essential tools into a single platform, improves functionality, and provides easy access to information would significantly enhance the overall library experience, boosting both efficiency and productivity.

Glossary of Terms:

- Patron: A library user, either one who borrows physical media from the library or uses a service provided by the library.
- Media Inventory: A database of all materials available through the library.
- Personalized Recommendation: Book and media suggestions tailored to the patron. Utilizes borrowed history to find related media.
- Reminders: Automated notifications when patrons have had a loan out for a certain period of time.

Functional Requirements:

No.	Priority	Description
REQ - 1	High	Supports cataloging of media in order to present data available to the user.
REQ - 2	High	Patrons will be able to search media inventory for a specific piece of media. Once searched, the system will display availability.
REQ - 3	Med	Automated overdue reminders to notify users of possible late fees.
REQ - 4	High	Allow library staff to change and/or update inventory or records.
REQ - 5	Med	Member account management such as loan history, suggestions, etc.
REQ - 6	Med	Overdue reminders

NonFunctional Requirements:

No.	Priority	Description
REQ - 1	Med	**Functionality:** System should be able to function regardless of platform.

REQ - 2	High	**Usability:** User interface should be simple and easy to use by staff and patrons.
REQ - 3	Med	**Reliability:** Should provide accurate information and update when necessary.
REQ - 4	High	**Performance:** Should be able to handle a large database with minimal to no issues.
REQ - 5	Med	**Supportability:** Easy to maintain with little-to-no upkeep. Should make everyone's job easier!

User-interface Requirements:

No.	Priority	Description
REQ - 1	High	Search and Filter
REQ - 2	Med	Member Account
REQ - 3	High	Inventory
REQ - 4	Med	Notification

Functional Requirement Specification

Stakeholders & Their Goals

Primary:

- Library Patron: Users who borrow/return media from the library. These are the people that interact with the library management system to search for media, check availability, and manage their account
    - Goals:
        1. Search for media
        2. Check availability
        3. Borrow and return
        4. Manage Account
- Library Staff: Employees of the library that are responsible for issuing, managing, and cataloging media. These individuals rely on the system to automate tasks.
    - Goals:
        1. Issue, renew, return
        2. Manage overdue procedures
        3. Update catalog

Secondary:

- Library Management: Users responsible for overseeing daily operations.
    - Goals:
        1. Manage staff accounts
        2. Oversee daily operations
        3. Generate reports
- System Admin.: Runs the backend of the library management system.
    - Goals:
        1. System settings
        2. Security
        3. Update system
- IT Support: Those who ensure the system is running smoothly.
    - Goals:
        1. Provide technical support
        2. Troubleshoot
        3. Hardware/software updates

Use Cases

Library Patron (12)

- Search: Patrons use a system to search the catalog for specific media. (2)
- Check Availability: Patrons use the library management system to check if certain media is available. (2)

- Borrow: Patrons use the library to borrow media. (2)
- Return: Patrons use the system to return previously borrowed media. (2)
- Renew: Patrons use the system to renew borrowed materials. (2)
- Manage Account: Patrons use the system to create and manage their personal accounts. (2)

Staff (9)

- Issue: Issue media to patrons. (2)
- Returns: Process returns and update availability. (2)
- Updates to Catalog: Add, remove, or update media. (3)
- Overdue Procedures: Enforce overdue procedures. (2)

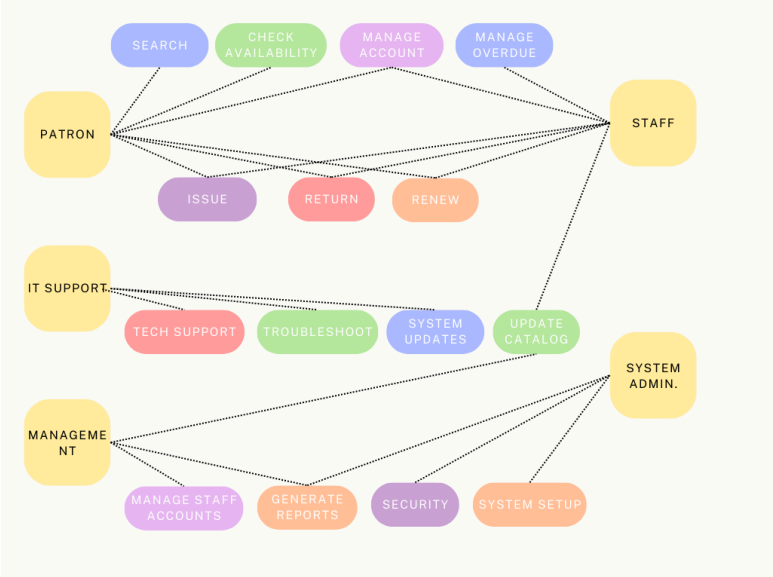
Management (7)

- Reports: Generate reports on library performance. (3)
- Manage Staff: Manage accounts and roles. (2)

Admin. (6)

- System Setup: Configure system settings in order to best fit the needs of the library. (3)
- Security: Ensure system security is under control. (3)

Use Case Diagram



Graph above shows all associations! The relationships will be listed down below:

- Borrow Media ← includes → Check Availability
- Return Media ← includes → Manage Overdue Procedures
- Generate Reports ← includes → Update Catalog
- Manage Account ← extends → Renew Media
- Issue Media ← extends → Renew Media
- Library Management ← generalizes → Library Staff
- Search for Media ← generalizes → Check Availability

Class Diagram

- LibrarySystem: Manages all system components
    - Attributes: systemID, systemName
    - Methods: manageUsers(), manageMedia(), generateReports()
- Media: Represents books, movies, etc…
    - Attributes: mediaID, title, author, category, availability
    - Methods: updateMedia(), checkAvailability()
- Patron: Information about account holders.
    - Attributes: patronID, name, contactInfo, accountStatus
    - Methods: borrowItem(), returnItem(), renewItem(), manageAccount()
- Staff: Library staff who manage the system.
    - Attributes: staffID, name, role
    - Methods: issueMedia(), processReturn()
- Management: Staff roles and additional managerial functions.
    - Attributes: managerID
    - Methods: generateReports(), manageStaff()
- Admin.: System administrators
    - Attributes:adminID, accessLevel
    - Methods: configureSystem(), manageSecurity()
- Loan: Tracks loans given out to patrons.
    - Attributes: loanID, mediaID, patronID, dueDate
    - Methods: calculateFine(), processReturn()

System Sequence Diagram

Use Case: Borrow Media

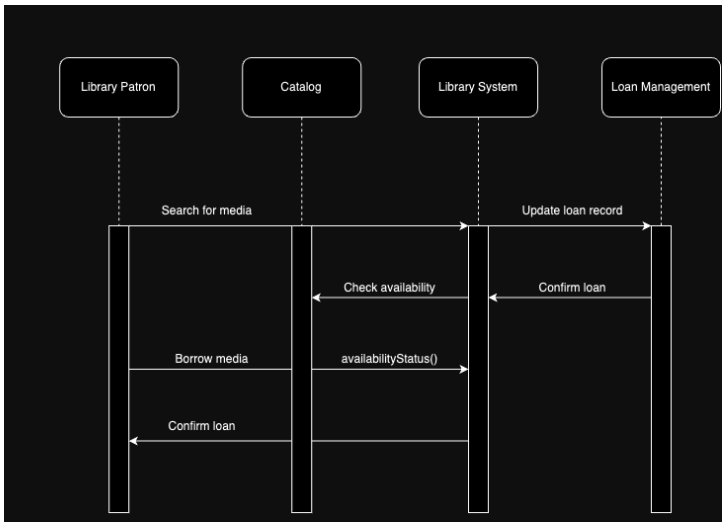
System Sequence

Actor: Library Patron

Objects: Library system, catalog, loan management

Steps:

1. The patron searches for their desired media in the library system.
2. The system checks the catalog to check availability.
3. The catalog returns availability status.
4. If the media is available, the patron requests to borrow the media.
5. The system processes the loan request.
6. The loan management system updates the loan records.
7. The system confirms the loan.



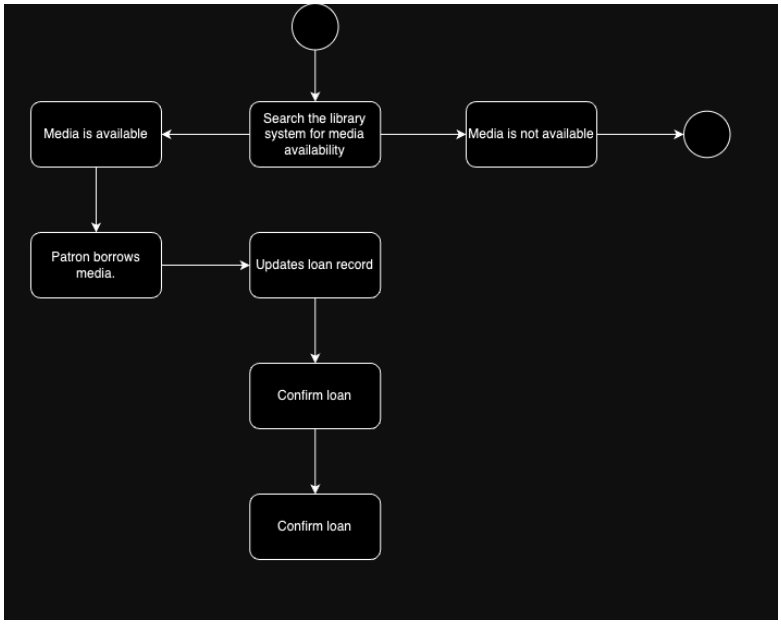
Activity Diagram

Initial: Patron searches for media.

Final: Confirmation that the media is borrowed.

Actions:

1. Patron searches for the media.
2. The system checks the catalog for its availability.
3. If the media is not available, the process ends.
4. If the media is available, the patron requests to borrow it.
5. The system processes the request.
6. Updates to the loan records.
7. Confirmation that the media is borrowed.



Use Case: Return Media

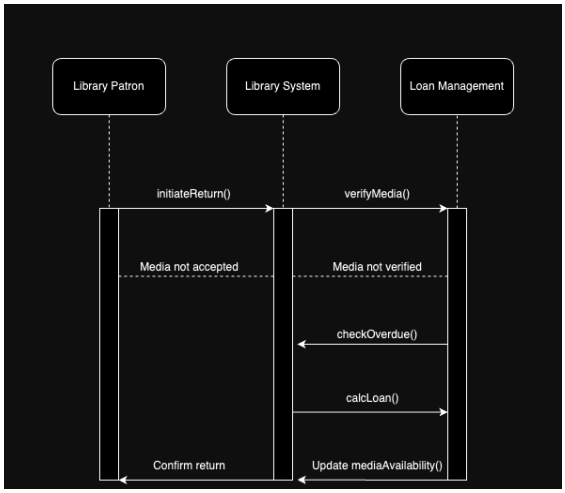
System Sequence

Actor: Library Patron

Objects: Library system, loan management

Steps:

1. The patron initiates the return process.
2. The system verifies the borrowed media.
3. The loan management system checks if the media is overdue.
4. If it is overdue, the system calculates a fine.
5. The loan management system updates the return status.
6. The catalog updates the media's availability.
7. The system confirms the return.



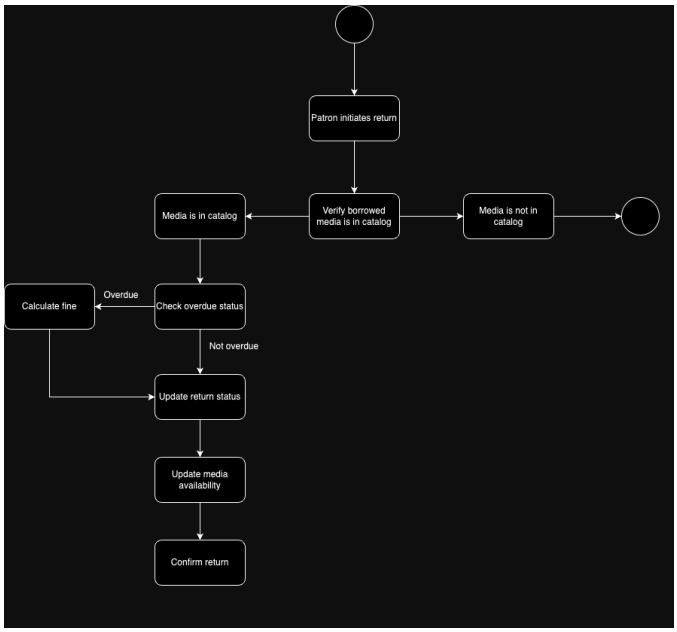
Activity Diagram

Initial: Patron initiates the return process

Final: Media is returned, and catalog is updated

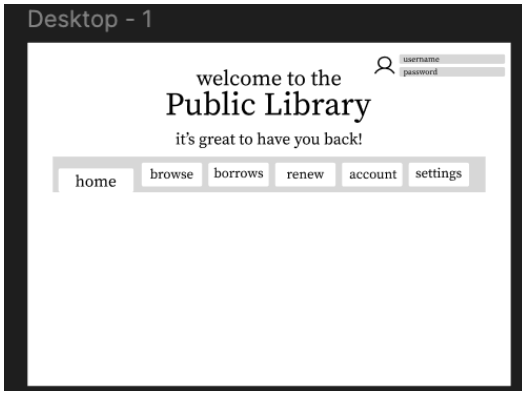
Actions:

1. Patron starts the return process.
2. The system verifies the media details.
3. Check if the media is overdue.
4. If overdue, calculate the fine.
5. Update the loan record.
6. Update the catalog to mark the media as available.
7. Confirm the return to the patron.



User Interface Specification

Main menu:



Use Case: Check Availability

Navigate to browse - Select media - Check availability

Use Case: Borrow

Select media - Click borrow

Use Case: Manage Account

Select account tab - Manage as needed

Use Case: Renew

Navigate to the renew tab (page will display all current borrows) - Renew selected title

Use Case: Search

Navigate to homepage - Search

Traceability Matrix

System Requirements

No.	Priority Weight (1-5)	Description
REQ1	5	Library Patrons can log in and log out from their accounts.
REQ2	5	Library Staff can log in and log out from their accounts.
REQ3	5	Library Patrons can search for media (books, movies, etc.) in the catalog.
REQ4	4	Library Patrons can check the availability of media items.
REQ5	5	Library Patrons can borrow and return media through the system.
REQ6	4	Library Patrons can renew borrowed media through the system.
REQ7	5	Library Staff can issue and return media for patrons.
REQ8	4	Library Staff can update the catalog by adding, modifying, or removing media items.
REQ9	3	The system should support overdue notifications and manage overdue procedures automatically.
REQ10	4	Library Management can generate reports on library usage and performance.
REQ11	4	Library Management can manage staff accounts and roles.
REQ12	3	System Admins can configure and update system settings.

Use Case

No.	Description
UC1	Login: To log in to a Library Patron or Library Staff account.
UC2	Search: To search for media items in the catalog.
UC3	Check Availability: To check if specific media items are available for borrowing.

UC4	Borrow: To borrow media items through the system.
UC5	Return: To return borrowed media items through the system.
UC6	Renew: To renew borrowed media items.
UC7	Issue: To issue media items to Library Patrons.
UC8	Update Catalog: To add, modify, or remove media items in the catalog.
UC9	Overdue Management: To notify patrons of overdue media and handle overdue procedures.
UC10	Generate Reports: To generate reports on library activity and performance.
UC11	Manage Accounts: To manage staff and patron accounts, including permissions and roles.
UC12	Configure System: To configure and update system settings for optimal operation.
UC13	Logout: To log out of a Library Patron or Library Staff account.

Traceability Matrix

System Requirement (REQ)	Related Use Cases (UC)
REQ1	UC1, UC13
REQ2	UC1, UC13
REQ3	UC2
REQ4	UC3
REQ5	UC4, UC5
REQ6	UC6
REQ7	UC7
REQ8	UC8
REQ9	UC9
REQ10	UC10
REQ11	UC11
REQ12	UC12

System Architecture & Design

The Book Burrow system is designed with a layered architecture to ensure modularity, scalability, and ease of maintenance. It comprises the following main layers:

1. Client Layer (Frontend)
    - Built using HTML, CSS, and basic JavaScript for rendering content.
    - Responsive and accessible design ensures usability across devices.
    - The frontend communicates with the server via HTTP requests for dynamic data updates.
2. Application Layer (Backend)
    - Implemented in PHP, the backend processes user requests, communicates with external APIs, and manages business logic.
    - Open Library API integration enables automated book data retrieval and recommendation generation.
    - Backend scripts handle login authentication, book searches, and dynamic content delivery.
3. Data Layer
    - While no internal database is currently used, the system fetches data from the Open Library API.
    - Data such as book titles, authors, publication years, and cover images are dynamically parsed and rendered on the frontend.
4. Automation Features
    - Automatic book recommendations based on predefined queries (e.g., “Harry Potter”).
    - API-based live search functionality reduces manual browsing effort.

System Flow:

1. User accesses the application via a web browser.
2. Requests (e.g., search queries) are sent to the PHP backend.
3. The backend interacts with the Open Library API, retrieves relevant data, and formats it into JSON.
4. Processed data is sent back to the frontend for rendering.

Algorithms & Data Structures

Algorithms:

1. Search Algorithm:
    - Implements a query-based search functionality using API calls to the Open Library API.
    - The algorithm parses user input and constructs an HTTP GET request to fetch relevant book data based on the search parameters.
    - Response data is filtered and formatted for display, showing book titles, authors, publication year, and cover images.
2. Data Fetching Algorithm:
    - Retrieves JSON data from the API endpoint.
    - Utilizes asynchronous methods to ensure a smooth user experience.
    - Includes error handling to manage failed requests or incomplete data.
3. Borrow/Hold System Logic:
    - Maintains a backend state to track book statuses (e.g., available, borrowed, on hold).
    - Ensures data integrity through checks to prevent duplicate or invalid status changes.

Data Structures:

1. Book Object:
    - A structured object containing:
        - Title
        - Author
        - ISBN
        - Status (Available/Borrowed/Hold)
    - Used for dynamic rendering and user interactions.
2. Status Dictionary:
    - Tracks the availability of books, keyed by unique identifiers (e.g., ISBN or database ID).
3. Search Result Array:
    - Temporarily stores search results for quick rendering and sorting.

User Interface Design & Implementation

Design Principles:

- Simplicity: The interface features a minimalistic design with intuitive navigation.
- Responsiveness: The layout adapts to different screen sizes for accessibility on desktops, tablets, and mobile devices.
- Accessibility: Includes ARIA labels for screen readers and keyboard navigability.

Key Components:

1. Search Bar:
    - Positioned at the top of the page.
    - Autocomplete functionality to suggest popular books or authors.
2. Results Section:
    - Grid layout displaying book covers, titles, and authors.
    - Hover effects to highlight clickable elements.
3. Action Buttons:
    - "Borrow" and "Hold" buttons with distinct colors (green and yellow, respectively).
    - Tooltip descriptions for additional guidance.
4. Notifications:
    - Pop-up messages confirming actions (e.g., "Book borrowed successfully").

Implementation:

- Frontend: HTML, CSS, and JavaScript.
- Backend: PHP for API integration and database management.
- Frameworks: Bootstrap for styling, and AJAX for dynamic content loading.

Design of Tests

Testing Objectives:

- Ensure seamless user interaction.
- Validate API integrations.
- Verify backend data integrity.

Testing Methods:

1. Unit Tests:
    - Test individual components like the search function, API calls, and data parsing logic.
    - Example: Verify that a search query returns the correct results.
2. Integration Tests:
    - Ensure smooth interactions between the frontend, backend, and external API.
    - Example: Confirm that borrowing a book updates its status both on the UI and in the database.
3. Usability Testing:
    - Test the interface with real users to identify pain points or confusion.
    - Example: Observe how easily users can search for and borrow books.
4. Performance Testing:
    - Test system responsiveness under load.
    - Example: Simulate multiple users searching simultaneously and evaluate performance.
5. Edge Cases:
    - Test unexpected or incorrect inputs.
    - Example: Entering a blank query or searching for a non-existent book.

Project Plan

Phases:

1. Planning:
    - Define requirements and scope (1 week).
    - Research APIs and tools to use.
2. Design:
    - Develop wireframes and UI mockups (2 weeks).
    - Plan backend architecture and database schema.
3. Implementation:
    - Build the frontend interface (2 weeks).
    - Integrate the Open Library API (1 week).
    - Develop backend logic for borrowing/holding functionality (2 weeks).
4. Testing:
    - Conduct unit, integration, and usability tests (2 weeks).
5. Deployment:
    - Set up hosting and deploy the application (1 week).
6. Documentation and Presentation:
    - Prepare user guide and presentation materials (1 week).

Timeline:

- Total duration: 10 weeks.