2025

Systems Design Document

LIBRARY MANAGEMENT SYSTEM

UNATHI OKHUE

Contents

1. Introduction

1.1. Project Overview

A Library Management System is a software application designed to manage library operations efficiently, by automating various tasks.

A Library Management System can do the following:

1. Manage books (searching, sorting and organizing books using a classification system)
2. Manage memberships (adding new members, tracking and collecting fines for materials overdue)
3. Book circulation (issue books and record issue dates, process book returns and check if books are overdue, allow renewal of books)
4. Report generation (generate inventory reports and various statistics with various variables)

1.2. Scope

System boundaries

The system's focus is to manage the books within the library. The system has no additional features for allowing access to e-books or managing library events, etc. The system does not integrate with external libraries or other systems and is restricted to one.

System limitations

1. **Data Accuracy:** inaccurate data that is beyond the systems data validation implementations can lead to error in reports and operations
2. **Security:** the system must have security measures that can help to protect sensitive data without fail.
3. **Scalability:** the system neds to be scaled to accommodate a growing number of members and books.
4. **Integration with other systems:** integrating the Library Management System with other systems and platforms may require additional development effort.

2. Overall System Description

2.1. General Perspective

The Library Management System is designed to automate and streamline various library operations.

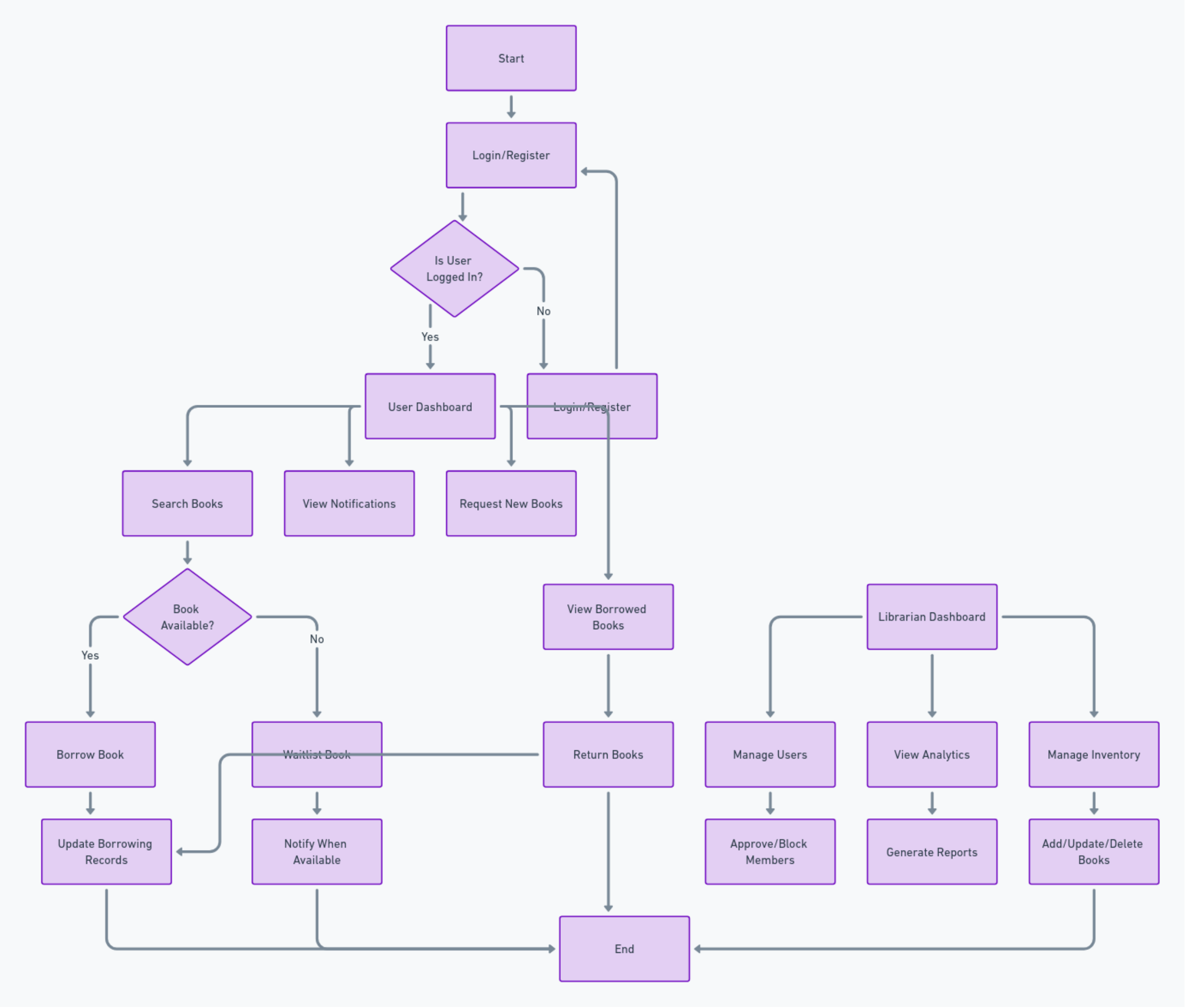
Key Components

User Interface	User friendly UI, including modules for adding/editing books and users, etc.
Database	Database to store all necessary information about books, members and transactions.
Business Logic	Logic which handles various tasks within the system. Tasks include the following: ○ Searching/ Updating/ Inserting books ○ Issuing and returning books ○ Calculating fines ○ Generating reports
Security	Solid security measures to protect sensitive data. Data encryption within the database.
Report	Generate various reports with stats, e.g.: overdue books, popular books, demographics.

2.2. Product Perspective

Functional Requirements	**Non-functional Requirements**
1. Book Management 2. Member Management 3. Circulation 4. Reports	1. Performance 2. Security 3. Reliability 4. Usability 5. Scalability 6. Maintainability

2.3. Process Perspective



2.4. Organizational Perspective

The system can be a crucial tool for libraries of all sizes. It streamlines operations, improves efficiency and enhances the overall library experience.

Key Roles and Responsibilities:

Librarians:

- Manage the library's collection
- Assist patrons with their queries
- Process book loans and returns
- Generate reports

IT Staff:

- Install and configure the LMS
- Maintain and update the system
- Troubleshoot technical issues
- Provide user training

How the system can play a role in other organizations

Schools and Universities:

- Manage textbooks and other learning materials
- Track student borrowing and return history

Archives:

- Catalogue and preserve historical documents and artifacts
- Manage access to archival materials

Rental Businesses:

- Manage inventory, rentals, and returns
- Track customer information and rental history

2.5. Interfaces

User interface

- GUI that allows librarians to interact with the system through menus, buttons and input fields.

Hardware interfaces (practical instances)

- Barcode scanners – scanning of books & member IDs
- RFID readers – used for automatic book check-in and check out
- Network devices – used for connecting the system to the internet and other devices

Software interfaces

- Database – used to store and retrieve information about books, members and transactions
- Operating system – system needs to be run on OS

2.6. Design Constraints

Technical Constraints

- **Hardware Limitations:** The system's performance may be limited by the available hardware resources, such as processor speed, memory, and storage capacity.
- **Software Limitations:** The choice of software and programming languages may influence the system's capabilities and performance.
- **Network Constraints:** Network speed and reliability can affect the system's responsiveness, especially for remote access and real-time updates.

Organizational Constraints:

- **Budget Constraints:** Limited budgets may restrict the scope of the system and the choice of technologies.
- **Time Constraints:** Tight deadlines may impact the development timeline and the inclusion of advanced features.
- **Staff Expertise:** The availability of skilled personnel may affect the system's development and maintenance.

User Constraints:

- **User Training:** Librarians and other users may require training to effectively use the system.
- **User Acceptance:** Users may have varying levels of technological literacy and may resist change.

Security Constraints:

- **Data Privacy:** The system must comply with data privacy regulations to protect sensitive user information.
- **Cybersecurity Threats:** The system must be protected against cyberattacks, such as hacking and data breaches.

3. Specific Design

3.1. Architectural Design

Component-Based Design

- **User Interface:**
  - Handles user interactions and displays information.
  - Includes modules for book search, member registration and report generation.

- **Business Logic:**
  - Encapsulates the core business rules and processes.
  - Manages book and member data, calculates fines and generates reports.

- **Data Access Layer:**
  - Interacts with the database to store and retrieve data.
  - Includes modules for connecting to the database, executing queries and handling data transactions.

Modular Design

The system can be further modularized into smaller, reusable components:

- **Book Module:** Handles book-related operations, such as adding, editing, deleting and searching for books.
- **Member Module:** Handles member-related operations, such as registration, updating information, and searching for members.
- **Circulation Module:** Manages book lending and return processes, including calculating fines.
- **Report Module:** Generates various reports, such as overdue books, popular books, and member activity.

By adopting a modular design, the system becomes more maintainable, flexible and easier to test.

3.2. Data Design

Database schema

Admin Table

Column Name	Data Type	Description
Username	VARCHAR (250)	Username of the system user, also determines the type of user
Password	VARCHAR (250)	Encrypted password for account protection
Blocked	BOOLEAN	Field that prevents the user from logging in if it returns true.

Books Table

Column Name	Data Type	Description
ISBN	VARCHAR (20)	Unique identifier of the book
Title	VARCHAR (100)	Title of the book
Author	VARCHAR (100)	Author of the book
Publication Year	INT	Year of publication
Publisher	VARCHAR (100)	Publisher of the book
Genre	VARCHAR (50)	Genre of the book
Quantity	INT	Number of copies available

Members Table

Column Name	Data Type	Description
Member ID	VARCHAR (255)	Unique identifier of the member
Name	VARCHAR (100)	Name of the member
Address	VARCHAR (255)	Address of the member
Contact Number	VARCHAR (20)	Contact number of the member
Email	VARCHAR (100)	Email address of the member

Loans Table

Column Name	Data Type	Description
Loan ID	VARCHAR (255)	Unique identifier of the loan
Member ID	VARCHAR (255)	Foreign key referencing the Members table
Book ISBN	VARCHAR (20)	Foreign key referencing the Books table
Issue Date	DATE	Date the book was issued
Return Date	DATE	Date the book is due
Due Date	DATE	Date the book was returned
Fine Amount	DECIMAL (10,2)	Amount of fine charged

3.3. User Interface Design

[See test plan document]

3.4. Software Design

Sequence Diagrams

A sequence diagram for issuing a book might involve the following actors and objects:

1. Librarian: Sends a request to the system.
2. System: Verifies the member's credentials and the book's availability.
3. Database: Retrieves book and member information.
4. System: Updates the book's status to "Issued" and records the issue date.
5. System: Informs the librarian of the successful issue.

Algorithm Design

1. Input: Member ID, Book ISBN

2. Check Member Validity:
   - Verify if the member is active and has no overdue books.

3. Check Book Availability:
   - Check if the book is available (not currently issued).

4. Issue Book:
   - If the book is available, update the book's status to "Issued" and record the issue date.
   - Calculate the due date.

5. Update Member Record:
   - Add the issued book to the member's record.

6. Print Receipt:
   - Generate a receipt with the book details, member information and due date.

7. Return Book:
   - Calculate any fines based on the due date and return date.
   - Update the book's status to "Available."
   - Remove the book from the member's record.

8. Print Receipt:
   - Generate a receipt with the book details, member information and any fines charged.

3.5. Security Design

Authentication and Authorization Mechanisms:

User Authentication:

- Implement strong password policies, requiring a combination of uppercase, lowercase letters, numbers, and special characters.
- Use password hashing techniques (e.g., bcrypt, Argon2) to store passwords securely.
- Consider two-factor authentication (2FA) for enhanced security.

Role-Based Access Control (RBAC):

- Assign different roles to users (e.g., librarian, admin) based on their privileges.
- Restrict access to sensitive data and functionalities based on user roles.
- Use a permission matrix to define which actions each role can perform.

Data Encryption:

- Data Encryption:
    - Encrypt sensitive data, such as member information and financial records, using strong encryption algorithms (e.g., AES).
    - Use secure key management practices to protect encryption keys.

- Secure Data Transmission:
    - Implement SSL/TLS encryption for secure communication between the client and server.
    - Use HTTPS to protect data transmitted over the network.

Access Control:

- Input Validation:
    - Validate user input to prevent malicious attacks like SQL injection and cross-site scripting (XSS).
    - Sanitize user input to remove harmful characters.

- Session Management:
    - Implement secure session management to prevent session hijacking.
    - Use session timeouts and regular session expiration.

- Logging and Monitoring:
    - Log user activities, system events, and security incidents.
    - Monitor system logs for suspicious activity and potential threats.

- Regular Security Audits:

- Conduct regular security audits to identify vulnerabilities and weaknesses.
- Keep the system and software up-to-date with the latest security patches.

3.6. Error Handling and Recovery

Error Handling Strategies:

- Input Validation:
  - Validate user input to ensure it is in the correct format and within acceptable ranges.
  - Reject invalid input and provide informative error messages.

- Data Validation:
  - Validate data before processing to prevent errors and inconsistencies.
  - Check for missing or invalid data and handle accordingly.

- Database Errors:
  - Implement error handling for database operations, such as connection failures, query errors, and data integrity issues.
  - Retry failed operations or log errors for later analysis.

- File I/O Errors:
  - Handle exceptions that may occur during file operations, such as file not found, permission denied, or disk errors.
  - Retry failed file operations or provide appropriate error messages.

- Network Errors:
  - Handle network connectivity issues, such as timeouts and connection failures.
  - Retry failed network operations or provide informative error messages.

Exception Handling Mechanisms

- Try-Catch-Finally:
  - Enclose code that may throw exceptions in a try block.

  - Handle specific exceptions in catch blocks.
  - Execute code in the finally block, regardless of whether an exception is thrown.

- Custom Exceptions:
  - Define custom exception classes to represent specific error conditions.
  - Throw custom exceptions to provide more specific error information.

- Logging:
  - Log errors and exceptions to a log file for analysis and debugging.
  - Include detailed information about the error, such as the timestamp, error message and stack trace.

3.7. Testing and Verification

[See test plan document]