Library Management System Design Document

1. Introduction

The Library Management System (LMS) is designed to automate and streamline library operations, including book search, borrower management, fines calculation, and check-in/check-out operations.

2. System Overview

The system comprises several modules: Book Management, Borrower Management, Book Loan, Fines Management, and Search. It is built using Flask,

3. Architecture

The architecture of the Library Management System (LMS) is designed as a web-based application using React for the frontend, Flask for the backend server, and MySQL as the database. The system follows a three-tier architecture, comprising the presentation layer, business logic layer, and data access layer. The presentation layer is implemented using React, providing a responsive and user-friendly graphical user interface (GUI) for librarians to interact with the system. React's component-based structure allows for modularity and ease of maintenance.

The backend server, implemented with Flask, acts as the business logic layer, handling various functionalities such as book search, availability, book loans, borrower management, and fines. Flask serves as a RESTful API, facilitating communication between the frontend and the database. The server processes requests from the React frontend, executes the necessary business logic, and interacts with the MySQL database to retrieve or update data.

MySQL is utilized as the relational database management system to store and manage the LMS data. The database schema includes tables such as BOOK, BOOK_AUTHORS, BORROWER, BOOK_LOANS, FINES, and AUTHORS, adhering to the specified requirements. The data access layer involves SQL queries and transactions to ensure proper retrieval and manipulation of data. The use of MySQL enables efficient storage, retrieval, and management of library-related information.

The system's overall design focuses on modularity, scalability, and adherence to functional requirements. React provides an intuitive and responsive user interface, Flask handles the business logic seamlessly, and MySQL ensures robust data storage and retrieval. This architecture allows for easy maintenance, future enhancements, and effective communication between the frontend and backend components. The combination of React, Flask, and MySQL forms a well-integrated solution for an efficient and user-friendly Library Management System.

4. Data Models

- **Book Model:** Represents books with attributes like ISBN, title, and authors.
- **Authors Model:** Represents book authors.
- **Borrower Model:** Represents library borrowers with a unique card ID.
- **BookLoan Model:** Represents book loans, tracks due dates, and check-in/out status.
- **Fines Model:** Represents fines associated with overdue books.

5. Features

5.1. Book Search

- Search books by author, ISBN, or title and display availability

5.2. Check-in and Check-out

- Check-in and check-out books, enforcing borrowing limits (maximum of 3 books) and limits related to undue fines (if fines are generated for borrower).
- Check-out books which are available. If the book is checked out it is set as unavailable
- If there are existing fines, then the user cannot checkout any more books.
-

5.3. Borrower Management

- Add borrower information (Name, SSN, Address, Phone No).
- Borrowers with duplicate SSNs are not allowed
- Generates and displays new ID for borrowers.

5.4. Fines Management

- Calculate fines for late books (0.25 cents for each day past due date).
- Update fines to update to the latest fines for all books.
- Allow payment of fines searched using borrower ID and display paid and unpaid fines.
- Fines can be paid, preventing further actions until cleared.

6. Check-in and Check-out

- Books are checked in with fines calculated based on due dates.
- Borrowing limits are enforced during check-out.
- Due date is 14 days past the current date.

7. Search and Browse

- Users can search for books based on various criteria.
- User-friendly interfaces for browsing books.

8. Borrower Management

- Borrower information can be added, and existing borrowers are added using scripts.
- Automatic generation of unique borrower IDs using a custom class method following the format: ID______ - 6 characters).

9. Technology Stack

- Flask (Backend)
- React (Frontend)
- MySQL (Database)

10. Deployment

- Deployment using runserver.
- Database server, local MySQL instance.

11. Conclusion

The Library Management System aims to provide an efficient, user-friendly solution for managing library operations, leveraging Flask's capabilities and best practices.