CUHK(SZ) Library Management System

HUO Yu 122090203

November 17, 2024

1 Overall Project Description

The CUHK(SZ) Library Management System is a comprehensive solution designed to modernize and streamline library operations at The Chinese University of Hong Kong, Shenzhen (CUHK(SZ)). The system integrates both physical and digital resources, enabling efficient management of books, user accounts, loans, holds, and fines. It caters to the needs of two primary user roles: **librarians** and **patrons** (students or general readers).

The system aims to:

- Provide seamless search and access to library resources.
- Enhance data analytics for informed decision-making.
- Simplify inventory tracking and catalog management.
- Improve user experience through an intuitive interface.

By leveraging modern technologies such as Python Flask for the backend and Vue.js for the frontend, the system ensures scalability, security, and ease of maintenance.

2 Requirements Analysis

2.1 User Roles

The system defines two distinct user roles:

1. **Librarian**: Responsible for managing library resources, users, and generating reports.
2. **Patron**: Library users who can search for books, borrow, place holds, and manage their accounts.

2.2 Functional Requirements

The system implements the following key functions, satisfying more than the minimum requirement of 10 different functions:

2.2.1 For Patrons

1. **User Registration**: Patrons can register for a new library account.
2. **User Login**: Secure authentication to access patron functionalities.
3. **Search Books**: Search for books by title or author.
4. **View Book Details**: Access detailed information about a specific book.
5. **Borrow Book**: Borrow available books.

6. **Place Hold**: Place a hold on books that are not currently available.
7. **View Loans**: View currently borrowed books and their due dates.
8. **Return Book**: Return borrowed books.
9. **Pay Fines**: Pay any outstanding fines.
10. **View Fines**: View details of fines incurred.

2.2.2 For Librarians

1. **User Login**: Secure authentication to access librarian functionalities.
2. **Add New Book**: Add new books to the library catalog.
3. **Edit Book Details**: Update information of existing books.
4. **Delete Book**: Remove books from the catalog.
5. **Manage Users**: View, edit, or delete patron accounts.
6. **Generate Reports**: Access analytics reports on library operations.
7. **View Overdue Loans**: Monitor loans that are overdue.
8. **View Unpaid Fines**: Keep track of fines that have not been paid.
9. **View Popular Books**: Analyze the most frequently borrowed books.
10. **Dashboard Access**: Access a comprehensive overview of library statistics.

2.3 Non-Functional Requirements

- **Security**: Secure user authentication and authorization.
- **Usability**: Intuitive user interface for both patrons and librarians.
- **Scalability**: Ability to handle an increasing number of users and resources.
- **Maintainability**: Clean code structure for ease of updates and maintenance.

3 Database and SQL Design

3.1 Database Schema

The database is designed using SQLite for simplicity, but it can be scaled to other relational databases like PostgreSQL or MySQL. The schema includes the following tables:

3.1. User Table

Stores information about both patrons and librarians.

Column	Data Type	Constraints
id	Integer	Primary Key, Auto-increment
username	String	Unique, Not Null
password	String	Not Null (hashed)
name	String	Not Null
role	String	Not Null ('patron' or 'librarian')

email	String	Not Null
phone	String	Not Null

3.2. Book Table

Contains details about books in the library.

Column	Data Type	Constraints
id	Integer	Primary Key, Auto-increment
title	String	Not Null
author	String	Not Null
isbn	String	Unique, Not Null
publisher	String	Not Null
publication_year	Integer	Not Null
copies_available	Integer	Not Null
total_copies	Integer	Not Null
location	String	Not Null

3.3. Loan Table

Tracks the borrowing of books by patrons.

Column	Data Type	Constraints
id	Integer	Primary Key, Auto-increment
user_id	Integer	Foreign Key to User.id, Not Null
book_id	Integer	Foreign Key to Book.id, Not Null
loan_date	DateTime	Default to current timestamp
due_date	DateTime	Not Null
return_date	DateTime	Nullable

3.4. Hold Table

Records holds placed by patrons on books.

Column	Data Type	Constraints
id	Integer	Primary Key, Auto-increment
user_id	Integer	Foreign Key to User.id, Not Null
book_id	Integer	Foreign Key to Book.id, Not Null
hold_date	DateTime	Default to current timestamp

3.5. Fine Table

Keeps track of fines incurred by patrons.

Column	Data Type	Constraints
id	Integer	Primary Key, Auto-increment
user_id	Integer	Foreign Key to User.id, Not Null
amount	Float	Not Null
paid	Boolean	Default False
description	String	Not Null

3.2 Relationships

- **User and Loan**: One-to-Many relationship (a user can have multiple loans).
- **Book and Loan**: One-to-Many relationship (a book can be loaned multiple times).
- **User and Hold**: One-to-Many relationship (a user can place multiple holds).
- **Book and Hold**: One-to-Many relationship (a book can have multiple holds).
- **User and Fine**: One-to-Many relationship (a user can have multiple fines).

3.3 Constraints

- **Primary Keys**: All tables have a primary key named `id`.
- **Foreign Keys**: Foreign keys (`user_id`, `book_id`) ensure referential integrity between tables.
- **Unique Constraints**:
  - **User Table**: `username` must be unique.
  - **Book Table**: `isbn` must be unique.

3.4 SQL Queries

Examples of SQL queries used:

User Login

```
SELECT * FROM User WHERE username = ?;
```

Search Books

```
SELECT * FROM Book WHERE title LIKE ? OR author LIKE ?;
```

Borrow Book

```
INSERT INTO Loan (user_id, book_id, due_date) VALUES (?, ?, ?);
UPDATE Book SET copies_available = copies_available - 1 WHERE id = ?;
```

Return Book

```
UPDATE Loan SET return_date = ? WHERE id = ?;
UPDATE Book SET copies_available = copies_available + 1 WHERE id = ?;
```

4 Front and Back-End Design and Implementation

4.1 Back-End Design (Flask)

4.1.1 Structure

- **app.py**: Initializes the Flask app and configures extensions.
- **models.py**: Defines database models using SQLAlchemy ORM.
- **routes.py**: Contains API endpoints and business logic.

- **database.py**: Configures the database connection.
- **config.py**: Stores configuration variables.
- **seed.py**: Populates the database with initial data.
- **requirements.txt**: Lists Python dependencies.

4.1.2 Key Technologies

- **Flask**: Lightweight web framework for Python.
- **Flask-JWT-Extended**: Handles authentication using JWT tokens.
- **SQLAlchemy**: ORM for database interactions.
- **Marshmallow**: Used for input validation and serialization.
- **Bcrypt**: For password hashing.

4.1.3 API Endpoints

- **/api/register**: User registration.
- **/api/login**: User login.
- **/api/books**: GET for listing books, POST for adding a new book.
- **/api/books/<book_id>**: GET, PUT, DELETE for specific book operations.
- **/api/dashboard**: Retrieves user-specific data like loans and fines.
- **/api/borrow/<book_id>**: Borrow a book.
- **/api/return/<loan_id>**: Return a borrowed book.
- **/api/hold/<book_id>**: Place a hold on a book.
- **/api/users**: GET for listing users (librarian only).
- **/api/users/<user_id>**: GET, PUT, DELETE for user management.
- **/api/payfine/<fine_id>**: Pay a fine.
- **/api/reports**: Generate various reports (librarian only).

4.2 Front-End Design (Vue.js)

4.2.1 Structure

- **main.js**: Initializes Vue app and integrates plugins.
- **App.vue**: Root component with navigation and router view.
- **axios.js**: Configures Axios for HTTP requests with authentication headers.
- **router/**: Defines routes and navigation guards.
- **store/**: Vuex store for state management.
- **components/**: Contains Vue components for different views.

4.2.2 Key Technologies

- **Vue.js 3**: Progressive JavaScript framework for building user interfaces.
- **Vue Router**: Manages application routing.
- **Vuex**: State management pattern and library for Vue.js applications.
- **Vuetify**: Material Design component framework.
- **Axios**: Promise-based HTTP client for the browser.

4.2.3 Components

- **UserLogin.vue**: Login interface.
- **UserRegister.vue**: Registration form.
- **SearchBooks.vue**: Search and display books.
- **BookDetails.vue**: Detailed view of a book with options to borrow or place a hold.
- **UserDashboard.vue**: Displays user loans and fines.
- **LibrarianDashboard.vue**: Overview of library statistics and management options.
- **AddBook.vue**: Form to add a new book to the catalog.
- **EditBook.vue**: Form to edit existing book details.
- **ManageUsers.vue**: List and manage patron accounts.
- **EditUser.vue**: Form to edit patron information.
- **ManageBooks.vue**: List and manage books in the catalog.

5 How to Use the System

5.1 Prerequisites

- **Backend**: Python 3.x, virtual environment, and required packages from `requirements.txt`.
- **Frontend**: Node.js and npm.

5.2 Setup Instructions

5.2.1 Backend

1. **Install Dependencies**:

```
pip install -r requirements.txt
```

2. **Initialize Database**:

```
python app.py
```

3. **Seed Database**:

```
python seed.py
```

4. **Run the Server**:

```
python app.py
```

5.2.2 Frontend

1. **Navigate to Frontend Directory**:

```
cd frontend

```

2. **Install Dependencies**:

```
npm install

```

3. **Run the Frontend**:

```
npm run serve

```

5.3 User Instructions

5.3.1 For Patrons

1. **Register**: Navigate to the registration page and fill in the required details.
2. **Login**: Use your credentials to log in.
3. **Search Books**: Use the search bar to find books by title or author.
4. **View Book Details**: Click on a book to see more information.
5. **Borrow Book**: If available, click the "Borrow Book" button.
6. **Place Hold**: If not available, place a hold to reserve the book.
7. **View Loans and Fines**: Access your dashboard to see current loans and any fines.
8. **Return Book**: Click the "Return" button next to a borrowed book in your dashboard.
9. **Pay Fines**: Use the "Pay" button next to any outstanding fines.

5.3.2 For Librarians

1. **Login**: Use your librarian credentials to log in.
2. **Dashboard**: Access library statistics and reports.
3. **Add New Book**: Navigate to "Add New Book" and fill in the book details.
4. **Manage Books**: Edit or delete existing books from the catalog.
5. **Manage Users**: View, edit, or delete patron accounts.
6. **Generate Reports**: View overdue loans, unpaid fines, and popular books.

6 Implementation Details Table

Below is a table mapping the system requirements to the functions and procedures in the codebase:

Requirement	**Function/Component**	**Description**
User Registration	`Register` (Backend), `UserRegister.vue` (Frontend)	Handles new user registrations, validates input, and stores hashed passwords.
User Login	`Login` (Backend), `UserLogin.vue` (Frontend)	Authenticates users using JWT tokens, validates credentials.
Search Books	`BookList` (Backend), `SearchBooks.vue` (Frontend)	Allows users to search for books by title or author using query parameters.
View Book Details	`BookDetail` (Backend), `BookDetails.vue` (Frontend)	Retrieves detailed information about a specific book.
Borrow Book	`BorrowBook` (Backend), `BookDetails.vue` (Frontend)	Enables patrons to borrow available books, updates inventory, and records loans.
Place Hold	`PlaceHold` (Backend), `BookDetails.vue` (Frontend)	Allows patrons to place holds on unavailable books.
View Loans	`UserDashboard` (Backend), `UserDashboard.vue` (Frontend)	Displays current loans and due dates for the logged-in patron.
Return Book	`ReturnBook` (Backend), `UserDashboard.vue` (Frontend)	Processes the return of borrowed books, updates inventory, calculates fines if overdue.
Pay Fines	`PayFine` (Backend), `UserDashboard.vue` (Frontend)	Allows patrons to pay outstanding fines.
View Fines	`UserDashboard` (Backend), `UserDashboard.vue` (Frontend)	Shows details of fines incurred by the patron.
Add New Book	`BookList` (POST method, Backend), `AddBook.vue` (Frontend)	Librarians can add new books to the catalog, validates input data.
Edit Book Details	`BookDetail` (PUT method, Backend), `EditBook.vue` (Frontend)	Librarians can edit existing book information.
Delete Book	`BookDetail` (DELETE method, Backend), `ManageBooks.vue` (Frontend)	Librarians can remove books from the catalog.
Manage Users	`UserList`, `UserDetail` (Backend), `ManageUsers.vue`, `EditUser.vue` (Frontend)	Librarians can view, edit, or delete patron accounts.
Generate Reports	`GenerateReports` (Backend), `LibrarianDashboard.vue` (Frontend)	Provides analytics such as total books, loans, fines, overdue loans, and popular books.
Dashboard Access	`UserDashboard`, `GenerateReports` (Backend), `UserDashboard.vue`, `LibrarianDashboard.vue` (Frontend)	Provides an overview of relevant information for both patrons and librarians based on their roles.

7 Conclusion

The CUHK(SZ) Library Management System streamlines modern library operations with features like book borrowing, holds, fines management, and reporting. Its scalable, secure architecture ensures efficiency and user satisfaction.