

Project Title: Library management System

Course: Advanced Database Systems Design

Team Members:

Sushanth Malliboina(811343775)

Jaya Krishna Gudla(811347333)

Date of Submission: May 7, 2025

1. Executive Summary

Library Management System (LMS) is an internet-based application that is utilized to effectively manage the different activities of a library, such as book stock management, tracking borrowers, and user interactions. It offers a single digital platform to make easier the completion of manual library work, lessening the need for manual record handling and enhancing overall efficiency of operations. This project utilizes modern web tech such as Flask (Python) for server-side programming, SQLite for persistent data storage, Bootstrap for responsive UI, and Jinja2 for dynamic template rendering to provide a full-stack, production-level solution.

The LMS aims to solve generic library management problems by providing an extensible, secure, and simple-to-use system supporting role-based access, real-time update of data, and resource conservation. The LMS aims to facilitate key library operations like cataloging of books, maintenance of users, and transaction processing and preserve data integrity and system integrity. The system is installable on diverse server platforms, and this makes the system highly scalable and deployable for any sized library.

The major goals of this project are:

- Simplifying book cataloging and inventory management processes.
- Simplifying borrowing and return for the end-user.
- Facilitating role-based access for secure and controlled processing of data.
- Delivering an extensible and scalable platform to be deployed in different server environments.
- Minimizing operational overhead and enhancing data accuracy with automation.
- Improving overall user experience with easy interfaces and streamlined workflow.

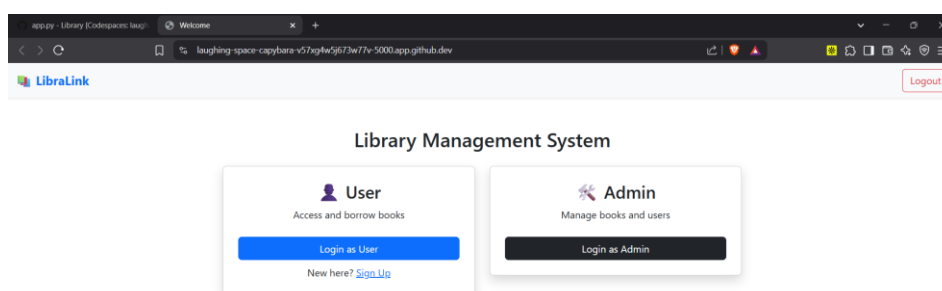


Figure: Welcome Page

Key Features:

- **User and Admin Authentication**
- **Book Management (Add, Update, Delete, View)**
- **Borrowing and Return Management**
- **User Dashboard** for personalized views
- **Admin Dashboard** for centralized control

This project was developed as part of the Advanced Database Systems Design course and is an experimental integration of frontend and backend technology in the service of ordinary practical applications.

2. Problem Statement

Ancient library management systems are usually marred by countless inefficiencies which limit their capability in the new study and research context. Those systems usually function based on piecemeal data storage, manual registration, and one-step processes, thereby resulting in limitations of keeping correct book inventories, monitoring user usage, and accessing resources timely. This absence of centralization may lead to misplaced files, non-standard data entry, and delay in marking the status of borrowed or returned materials, thereby lowering overall working efficiency.

Library Management System (LMS) specifically solves these problems by providing a centralized, automated system with book inventory counting, borrower tracking, and user interaction in one unified system. It is designed to enhance precision of information, minimize manual workload, and generally enhance the general user experience through the delivery of real-time access to critical information. This model not only minimizes operation overhead but also provides administrators and users with a natural, transparent experience in accessing library materials.

Stakeholders are:

- Library administrators who catalog books, handle inventories, monitor day-to-day library activity, and ensure data integrity.
- Library users (researchers, students, and employees) who borrow, return, and retrieve information regarding available resources, with a streamlined, self-service experience.

Through resolving these fundamental concerns, the LMS aims to make traditional libraries modern, technology-integrated places for more effective learning, research, and sharing of knowledge.

3. System Overview

System Architecture:

The three main layers of the system architecture are:

- **Frontend (Presentation Layer)** – HTML/CSS templates for the UI, embellished with Bootstrap for responsive look.

- **Backend (Application Layer)** – Flask routes to process user requests, authentication, and business rules.

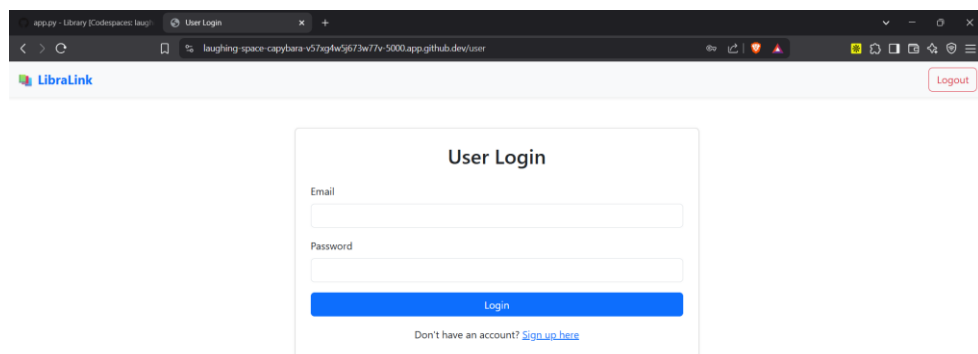
- **Database (Storage Layer)** – SQLite for database management of book, user, and transaction data.

Technologies Used:

- Flask (Python) for routing and API processing
- SQLite for lightweight database management
- Bootstrap for responsive, clean UI design
- Jinja2 for template rendering

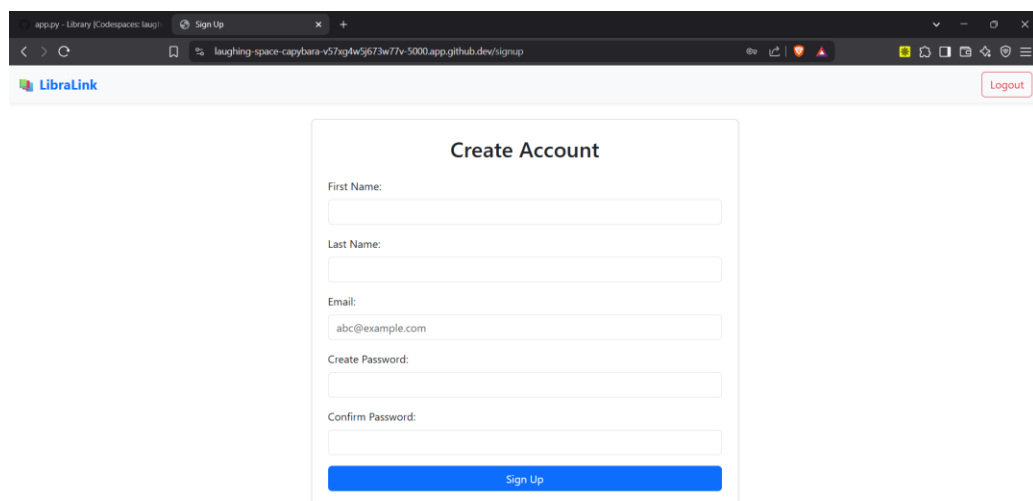
Subsystems:

- **User Authentication** – Secure login and registration with password encryption.



The screenshot shows a web browser window with the title 'User Login'. The address bar shows the URL 'laughing-space-capybara-v57q4w5673w77v-5000.app.github.dev/user'. The page has a 'LibraLink' logo on the left and a 'Logout' button on the right. The main content area is a white box with the title 'User Login'. It contains two input fields: 'Email' and 'Password'. Below these fields is a blue 'Login' button. At the bottom of the box, there is a link: 'Don't have an account? [sign up here](#)'.

Figure: User Sign In page



The screenshot shows a web browser window with the title 'Sign Up'. The address bar shows the URL 'laughing-space-capybara-v57q4w5673w77v-5000.app.github.dev/signup'. The page has a 'LibraLink' logo on the left and a 'Logout' button on the right. The main content area is a white box with the title 'Create Account'. It contains five input fields: 'First Name:', 'Last Name:', 'Email:' (with the value 'abc@example.com'), 'Create Password:', and 'Confirm Password:'. Below these fields is a blue 'Sign Up' button.

Figure: User Signup page

Admin Login

Email:

Password:

Login

Figure: Admin login page

- **Book Management** – Full CRUD operations on book records, including title, author, genre, and availability status.

Welcome, Jaya Krishna!

Search by title or author

Search

My Borrowed Books

Available Books

Title	Author	Price	Stock	Borrow
God is gamer	Ravi subramaniam	\$56.0	38	Borrow
The Alchemist	Paulo Coelho	\$3.59	9	Borrow
Atomic Habits	James Clear	\$5.99	6	Borrow
Sapiens	Yuval Noah Harari	\$7.19	6	Borrow
To Kill a Mockingbird	Harper Lee	\$4.79	7	Borrow
1984	George Orwell	\$4.19	3	Borrow
The Subtle Art of Not Giving a F*ck	Mark Manson	\$5.39	9	Borrow
A Thousand Splendid Suns	Khaled Hosseini	\$4.79	6	Borrow

Figure: User ‘Jaya Krishna’ Home page for Read Operation

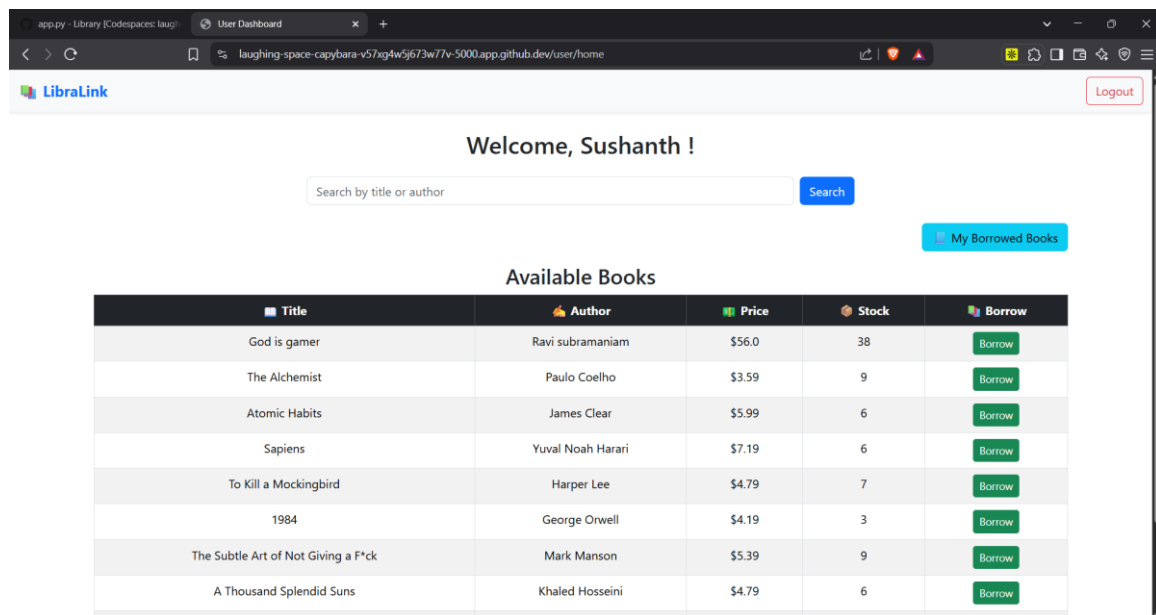


Figure: User 'Sushanth' Home page for **Read** Operation

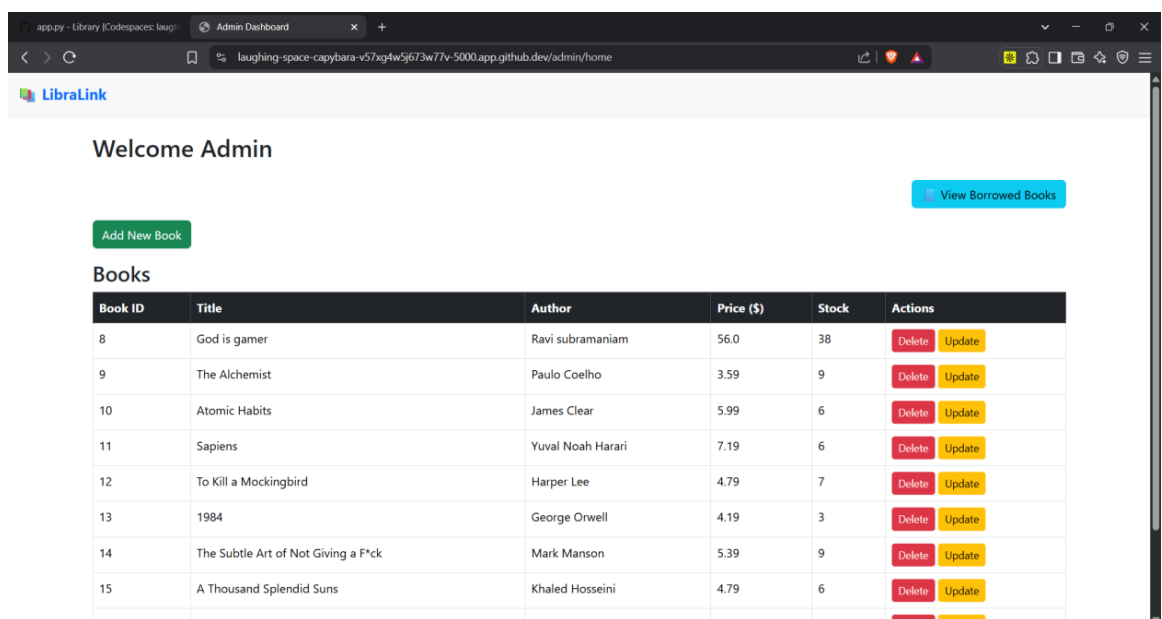


Figure: Admin page for **Delete, Update** operation

Add a New Book

Title:

Author:

Price (\$):

Stock:

Figure: Admin page for **Create** operation

- **Borrowing System** – Efficient management of book borrowing and return transactions, including due date tracking and overdue alerts

My Borrowed Books

Title	Author	Borrow Date
1	God is gamer	Ravi subramaniam
2	Atomic Habits	James Clear
3	Atomic Habits	James Clear
4	God is gamer	Ravi subramaniam
5	God is gamer	Ravi subramaniam
6	God is gamer	Ravi subramaniam
7	God is gamer	Ravi subramaniam
8	God is gamer	Ravi subramaniam
9	God is gamer	Ravi subramaniam
10	1984	George Orwell
11	1984	George Orwell
12	God is gamer	Ravi subramaniam
13	God is gamer	Ravi subramaniam

Figure: Borrowed Books list

- **Admin Dashboard** – Centralized control for library staff, including user management and transaction oversight.

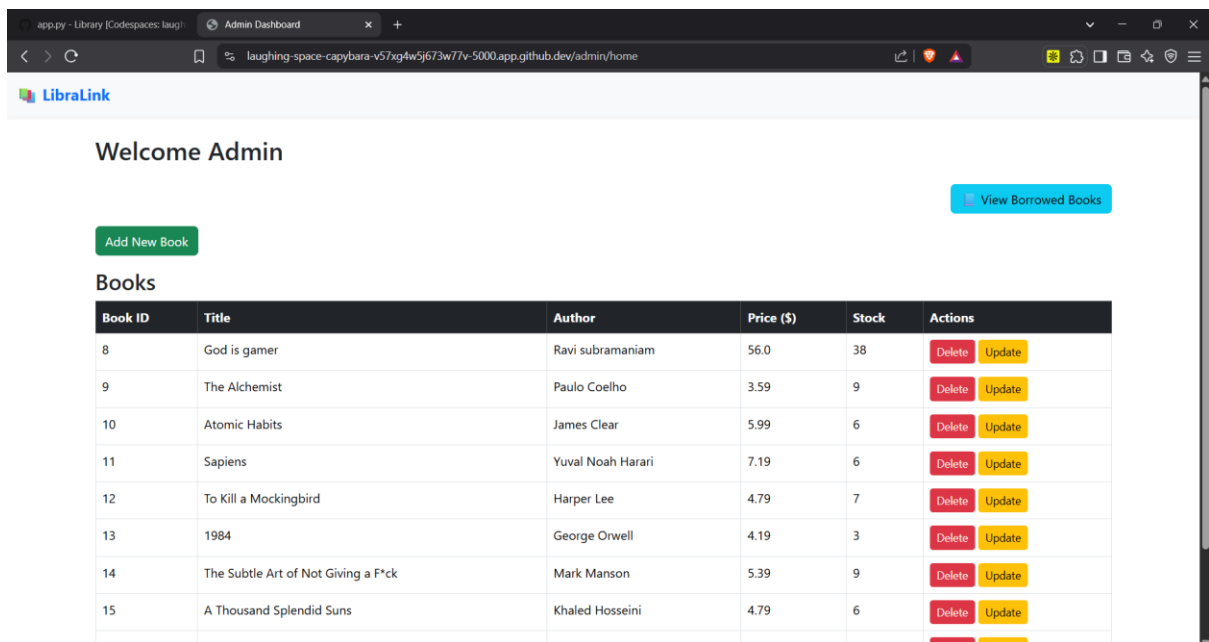


Figure: Admin Home page with **CRUD** operations

4. Development Process

Timeline:

- Week 1: Initial setup of requirements gathering, project planning, and database schema design.
- Week 2: Setup of Flask, routing definitions, and initial setup of basic user authentication.
- Week 3: Setup of frontend template integration, responsive UI, and data binding.
- Week 4: Setup of core features such as book management (CRUD operations) and transaction tracking.
- Week 5: Testing, debugging, and deployment of the final project.

Team Workflow:

The project employed an Agile-based workflow, with frequent meetings to exchange progress, allocate tasks, and solve issues. Everyone in the team had their specific roles to fill in order to maintain an equal task allocation and effective collaboration.

Roles:

- Jaya Krishna: Flask backend configuration, database schema design, user authentication, and deployment of the project.
- Sushanth Malliboina: Frontend integration, CRUD functionality for managing books, transaction tracking, and UI optimization.

Tools:

- GitHub for collaboration and version control.
- SQLite for database management.
- Flask for server-side logic and API management.

5.Implementation Details

Major Database Tables:

- \tuser (id, username, password, email, is_admin)
- \tbook (id, title, author, genre, publication_date, availability)
- \ttransaction (id, book_id, user_id, borrowed_date, return_date, status)

Code Structure:

- \t/app.py (Main application logic and route definitions)
- \t/templates/ (HTML templates for UI)
- \t/static/ (CSS and JS files)
- \t/setup_database.py (Database initialization and seeding script)

6. Testing and Validation

The project underwent extensive testing to ensure reliable performance and data integrity. Testing included both unit tests for individual components and system tests for end-to-end workflows, covering:

- User authentication and session management.
- Book addition, update, and deletion processes.
- Borrowing and return transaction flows.

Example Test Case:

Title: Book Checkout Flow Input: Valid user credentials and book selection. Expected: Book status updated to 'borrowed' and transaction logged successfully. Actual: System behavior as expected, confirming the accuracy of the checkout process.

7. Results and Evaluation

The Library Management System successfully achieved the following outcomes:

- Simplified and automated library operations.
- Enhanced data accuracy and record management.
- Improved user experience through a responsive and intuitive UI.
- Real-time transaction updates for efficient book tracking.

Evaluation Criteria:

- Book stock was monitored properly, with real-time borrow and return updates.
- User session management and authentication were secure and functioning as required.
- Data integrity and consistency were ensured in all CRUD operations.
- Transaction logs were properly generated for borrow and return operations.
- Backend and frontend integration ensured smooth user experience without data loss.

Retrospective:

- Flask enabled simple backend route management and session handling.
- Bootstrap enabled ease of frontend development with responsive UI.
- SQLite offered stable and efficient data storage without the requirement of external database servers.
- Jinja2 templates utilized made dynamic content rendering simple, eliminating duplicated code.
- The project had an ideal balance between scalability and usability and was therefore appropriate for small- to medium-sized libraries.

8. Future Work and Recommendations

Possible future projects of this project are:

- User borrowing history-based book suggestion algorithms implemented.
- Imposing fine and penalty control for overdue books.
- Utilize support for handling multimedia and e-book files.
- Hosting the system on a cloud server for wider accessibility.

9. User Manual

Installation Instructions:

- **Clone the repository:** git clone
- **Install dependencies:** pip install flask
- **Run the server:** python app.py
- **Access the application at** <http://127.0.0.1:5000>

10. Appendices

Codebase available on GitHub : <https://github.com/JayaKrishna21/Library>

Database Dump: library.db

Sample Data:

- **Title:** "Atomic Habits"
- **Author:** James Clear
- **Price:** \$5.99
- **Stock:** 7
- **Borrow**

11. Personal Contributions

Jaya Krishna Gudla

- Led the initial setup of the Flask backend, including route definitions and session management.
- Designed the SQLite database schema, ensuring efficient data storage and retrieval.
- Implemented user authentication and role-based access control.
- Integrated Bootstrap for responsive UI design and template styling.
- Conducted final testing and handled deployment.

Sushanth Malliboina

- Focused on integrating frontend templates with backend logic.
- Developed CRUD operations for book management and user transactions.
- Implemented the borrowing and return management modules.
- Assisted in database optimization and data integrity checks.
- Managed GitHub repository and coordinated version control.