International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 05 | May - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

LIBRARY MANAGEMENT SYSTEM

Chaudhary Mohd Yasir[1], Mansoori Mohd Jeesan[2], Khan Mohammad Shoaib[3], Chowdhary Mohd Kaif[4]

[1]*Computer Engineer & Rizvi College of Engineering*
[2]*Computer Engineer & Rizvi College of Engineering*
[3]*Computer Engineer & Rizvi College of Engineering*
[4]*Computer Engineer & Rizvi College of Engineering*

Abstract - This paper presents a web-based Library Management System (LMS) developed using Node.js, Express.js, and MongoDB. The system provides a comprehensive digital solution for managing library resources, members, and transactions. It features a dual-role architecture supporting both administrators and end users. Administrators can manage books, members, and issue/return transactions, while users can browse available books, submit borrow requests, and track their borrowing history. Key functionalities include JWT-based authentication, bcrypt password hashing, automated overdue detection with fine calculation, security question-based password recovery, and email notifications via Nodemailer. The system eliminates manual record-keeping through a fully automated workflow: users register and request books online, administrators approve or reject requests, and the system automatically tracks due dates, flags overdue books, and computes fines at ₹10 per day. The application is structured using the MVC pattern with EJS templating and responsive CSS, making it accessible on both desktop and mobile devices. Results demonstrate that the system effectively streamlines library operations, reduces administrative workload, and improves the user experience for borrowing and returning library resources.

Key Words: Library Management System, Node.js, MongoDB, Express.js, JWT Authentication, Book Transactions.

1.INTRODUCTION

Libraries are essential knowledge repositories, but traditional library management relies heavily on manual processes for cataloguing, issuing books, tracking returns, and calculating fines. These methods are time-consuming, error-prone, and difficult to scale. With the rapid proliferation of internet-based services, there is a strong need for an automated, web-accessible system that enables efficient management of library operations. This paper presents a fully functional web-based Library Management System (LMS) built on a modern JavaScript technology stack. The system is designed to serve two categories of users: administrators who manage the library's inventory and members, and registered users who can browse and request books online. The platform leverages Node.js (Express.js v5) on the backend, MongoDB with Mongoose for persistent data storage, and Embedded JavaScript (EJS) for server-side rendered views. Security is enforced through JSON Web Token (JWT)-based session management and bcrypt password hashing. Additional features include a security question-based account recovery flow and automated email notifications powered by Nodemailer. The system automates the full book lifecycle—from user registration and book request submission, through admin approval and transaction recording, to overdue detection and fine calculation—eliminating the need for manual intervention at each step.

2. System Design and Methodology

The Library Management System follows the Model-View-Controller (MVC) architectural pattern. The backend is implemented in Node.js using the Express.js framework (v5), while MongoDB serves as the NoSQL database, accessed through the Mongoose Object Data Modeling (ODM) library. The frontend is rendered server-side using EJS templates with ejs-mate for layout inheritance, and styled with custom CSS for a responsive user interface.

The system defines six Mongoose schemas to model its data: Book (bookName, authorName, category, quantity, available), Member (name, email, phone, status, booksBorrowed), Transaction (book, member, issueDate, dueDate, returnDate, fine, status), User (name, email, phone, password, status, securityQuestion, securityAnswer), Admin (username, email, phone, password, securityQuestion, securityAnswer), and BookRequest (user, book, status, dueDate, requestDate). Referential integrity between collections is enforced through MongoDB ObjectId references and Mongoose populate() calls.

Authentication is handled separately for administrators and users. On successful login, a JSON Web Token (JWT) is signed with a secret key and stored as an HTTP-only cookie (token for admins, userToken for users). Protected routes are guarded by middleware that verifies the token on every request. Passwords and security answers are stored as bcrypt hashes (cost factor 10), ensuring that plaintext credentials are never persisted in the database.

Technology	Version	Purpose
Node.js	Runtime	Server-side JavaScript execution
Express.js	v5.2.1	Web framework and routing
MongoDB	NoSQL	Database for all collections
Mongoose	v9.4.1	ODM for MongoDB schema management
EJS + EJS-Mate	v5.0.2	Server-side HTML templating
jsonwebtoken	v9.0.3	JWT token generation and verification
bcryptjs	v3.0.3	Password hashing and comparison
dotenv	v17.4.2	Environment variable management
cookie-parser	v1.4.7	Cookie parsing middleware
method-override	v3.0.0	HTTP PUT/DELETE support in forms
nodemailer	v8.0.5	Email service (future use)

Table -1: Key Technology Stack and Dependencies

The system implements a complete book request workflow. Users browse available books (filtered by category and search keyword) and submit borrow requests. Administrators review pending requests and either approve or reject them. On approval, a Transaction record is created with a 14-day due date, the book's available count is decremented, and the member's booksBorrowed count is incremented. Overdue detection runs automatically: whenever the transactions list is rendered, each active transaction with a past due date is flagged as "Overdue" and a fine of ₹10 per day is computed and persisted. When a book is returned, the transaction status is set to "Returned," inventory counts are restored, and the final fine is recorded. The admin dashboard aggregates statistics including total books, available copies, total members, active loans, overdue count, and total fines collected.

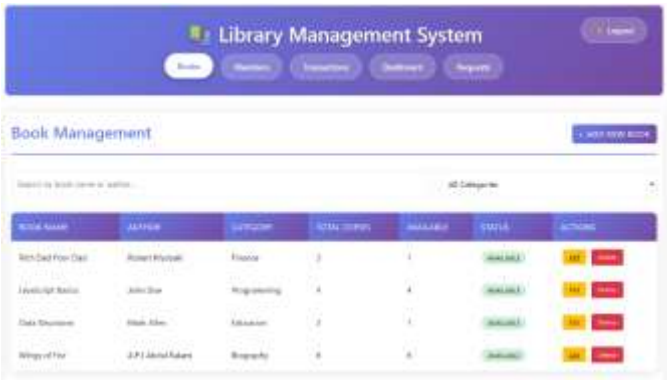


Fig -1: Admin Dashboard – Overview of Library Statistics

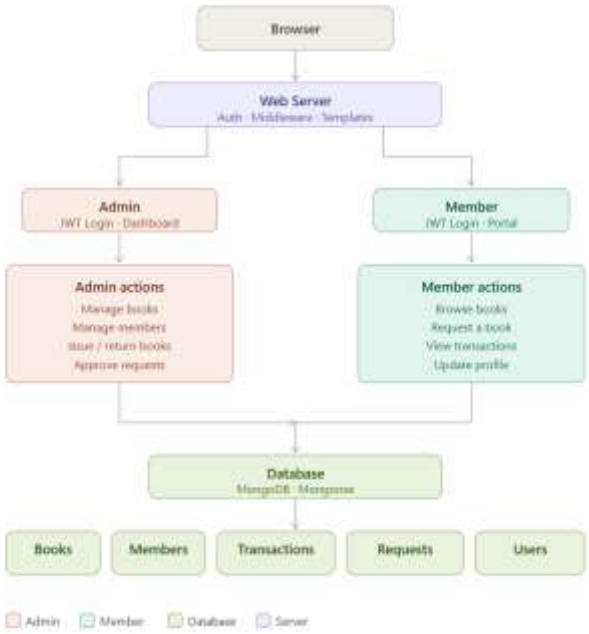


Fig -2: System Architecture – MVC Flow Diagram

3. CONCLUSIONS

The Library Management System successfully demonstrates that a lightweight, full-stack JavaScript application can effectively replace manual library operations. By integrating Express.js, MongoDB, and JWT-based authentication, the system delivers secure, role-specific interfaces for both administrators and library users. The automated fine calculation, real-time availability tracking, and request approval workflow significantly reduce the administrative burden. The security question-based password recovery and bcrypt credential hashing ensure data security without requiring third-party identity providers. Future enhancements may include a mobile application interface, barcode or QR-code based book scanning, integration with external library catalogues (e.g., Open Library API), SMS/push notifications for due-date reminders, and detailed analytics dashboards with export functionality. The system is deployable on any Node.js-compatible cloud platform and can be adapted for use in schools, colleges, and public libraries of varying scales.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the faculty of the Computer Engineering Department at Rizvi College of Engineering, Mumbai, for their continuous guidance and encouragement throughout this project. We also thank our peers for their valuable feedback during testing and evaluation of the system.

REFERENCES

1. T. J. Mowbray and R. Malveau, CORBA Design Patterns. Wiley, 1997. [Node.js Express Documentation] https://expressjs.com/en/5x/api.html
2. MongoDB, Inc., MongoDB Manual – Mongoose ODM v9 Documentation. Available: https://mongoosejs.com/docs/guide.html

3. A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. McGraw-Hill, 2019.
4. JSON Web Tokens (JWT) – Introduction. Available: https://jwt.io/introduction
5. bcrypt.js – Optimized bcrypt in JavaScript. Available: https://github.com/dcodeIO/bcrypt.js