IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

Library Management System

Vinayak Sharad Ighave, Deepa Hishobe, Akshay Jadhav
Department of Science and Technology
Bachelor of Computer Applications (BCA)
JSPM University, Pune, Maharashtra, India

Abstract: *Managing a library manually is harder than it looks. Between keeping track of hundreds of books, remembering who borrowed what, calculating fines, and maintaining membership records, library staff end up spending most of their time on repetitive administrative work rather than actually helping patrons. This paper talks about why that's a problem and what we built to fix it.*

We designed and developed a Library Management System (LMS) — a software application that handles all the day-to-day operations of a library automatically. The system covers everything from adding and searching books, to registering members, issuing and returning copies, tracking overdue items, and generating reports. We built it using a layered architecture and a relational database, and we tested it thoroughly to make sure it works well under real conditions.

The results were encouraging. Search time dropped by about 80%, admin workload fell by over 65%, and the people who used the system during testing rated it positively in terms of ease of use. We believe this kind of system can make a genuine difference for libraries of all sizes.

Keywords: Library Management System, Automation, Database, Circulation, Information Retrieval, Software Design

I. INTRODUCTION

Think about the last time you visited a library. If it was a well-run one, chances are you found the book you wanted quickly, checked it out without much trouble, and left satisfied. But behind that smooth experience, there is usually a lot of work going on — cataloguing new arrivals, processing returns, chasing overdue borrowers, and keeping membership records up to date. When all of that happens on paper or in spreadsheets, it becomes a serious bottleneck.

Libraries in schools, colleges, and communities across India still rely heavily on manual processes. This leads to problems: books go missing, fines are miscalculated, members show up for books that are already out, and staff spend hours doing work that a computer could handle in seconds. We noticed this firsthand at our institution and decided to do something about it.

Our goal was straightforward — build a system that automates the routine tasks so that librarians can focus on things that actually need a human touch, like helping students find the right resources or organising events. The system we built is not just a digital version of the old register book. It is a full-fledged application with a clean interface, a proper database, role-based access, automated notifications, and reporting tools.

II. PROBLEM STATEMENT & OBJECTIVES

During our initial analysis, we identified several recurring pain points in traditional library setups:

- Books frequently go missing or are misplaced because there is no real-time tracking of who has what.
- Overdue fines are either not calculated at all or done inconsistently, leading to revenue loss and patron confusion.
- Searching for a specific book in a large collection takes too long when done manually.
- There is no easy way to know at a glance how many copies of a title are available right now.
- Generating reports for institutional records is a tedious, error-prone process.

DOI: 10.48175/IJARSCT-35346

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

With these problems in mind, we set out to build a system that would:

- Automate the entire book issue, return, and renewal cycle.
- Maintain accurate member records and track borrowing history.
- Calculate fines automatically and maintain payment records.
- Allow fast, flexible searching of the catalogue.
- Generate meaningful reports that help library administrators make better decisions.

III. SYSTEM DESIGN

3.1 Architecture

We followed a three-tier architecture, which keeps the application clean and easy to maintain. The presentation layer handles what the user sees — the web interface built with React.js. The business logic layer handles all the rules and processing using Node.js with Express. The data layer manages everything stored in the database, which runs on PostgreSQL.

This separation matters in practice. If we ever want to change how the search results look, we only touch the frontend. If we need to update how fines are calculated, we update the business logic and nothing else breaks. It also makes the system easier to scale as the library grows.

3.2 Key Modules

Module	What It Does
Catalogue Management	Add, edit, search, and organise books. Supports bulk CSV import and barcode/QR label printing.
Member Management	Register patrons, manage membership tiers, track borrowing history and contact details.
Circulation	Handle book issue, return, renewal, and reservation with business rule enforcement.
Fine & Payment	Auto-calculate overdue fines, record payments, and print receipts.
Notifications	Send email alerts for due dates, overdue reminders, and reservation availability.
Reports & Analytics	Generate circulation stats, inventory summaries, and member activity reports in PDF/Excel.

3.3 Database Design

We designed the database in third normal form (3NF) to avoid redundancy and keep data consistent. The core tables are: Books, Authors, Publishers, Members, Transactions, Reservations, and Users. Every book copy is tracked individually, which means the system always knows not just that a library owns three copies of a title, but exactly where each copy is and what its current status is.

We also added a full-text search index on the books table using PostgreSQL's built-in tsvector support. This allows patrons to search by any combination of title, author, or keyword and get results almost instantly, even in a catalogue with hundreds of thousands of entries.

IV. IMPLEMENTATION HIGHLIGHTS

We used React.js for the frontend because it makes it easy to build interactive interfaces that respond quickly to user actions without reloading the page. The backend runs on Node.js with Express, which handles multiple requests

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

simultaneously without slowing down. All database interactions go through the Sequelize ORM, which prevents SQL injection by design and makes it easier to manage schema changes over time.

Security was something we took seriously from the start rather than adding at the end. Passwords are hashed with bcrypt before storage. All API sessions use JWT tokens with short expiry times. Role-based access ensures that a student patron cannot access admin functions, and an ordinary librarian cannot change system configuration. Every write action in the system is recorded in an audit log, so there is always a clear trail of who did what and when.

For performance, we introduced a Redis caching layer. Frequently searched catalogue queries are stored in memory for a short period, which dramatically reduces the load on the database during busy hours — say, when hundreds of students are all searching for the same textbook before an exam.

V. TESTING & RESULTS

We tested the system in several ways. Unit tests covered individual functions and business rules. Integration tests validated that all the modules worked properly together. We also ran load tests using Apache JMeter, simulating up to 500 simultaneous users, to check how the system performed under pressure.

The numbers came back better than we expected. Even with 500 concurrent users, the average response time stayed under 2.2 seconds. Search results for a 500,000-record catalogue came back in just over a second. We ran the system continuously for 30 days during our pilot test and recorded 99.87% uptime.

We also asked 15 real users — a mix of librarians and students — to use the system and give feedback. The average usability score (measured using the standard System Usability Scale) came out at 81.6 out of 100, which falls in the "Good" range. The feedback that stood out most: people appreciated how easy the search was to use and how clearly the issue/return process was laid out.

What We Measured	Target	Result
Search response time (500K records)	< 2.0 s	1.12 s ✓
System uptime over 30 days	≥ 99.5%	99.87% ✓
Usability score (SUS)	≥ 75	81.6 ✓
Admin workload reduction	≥ 50%	65.2% ✓
Fine calculation accuracy	100%	100% ✓

VI. CONCLUSION

Building this system taught us a lot about what it actually takes to solve a real problem with software. The technical side — choosing the right architecture, designing the database, writing clean code — is important, but so is understanding the actual workflow of the people who will use it. We spent time talking to librarians, observing how they worked, and making sure the system fit their process rather than forcing them to change everything they were used to.

The Library Management System we built does what it set out to do. It saves time, reduces errors, and makes the library experience better for everyone involved. It is not perfect — there are features we would like to add, and there are edge cases we are still refining — but it is a solid foundation that a real institution could deploy and benefit from.

Going forward, we want to add a mobile app so students can check availability and reserve books from their phones. We are also interested in experimenting with a recommendation feature that suggests books based on what a patron has borrowed in the past. RFID integration for self-checkout is another direction we see as practical and impactful. For now, though, we are happy with what this project achieved and grateful for everything it taught us along the way.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, May 2026
Impact Factor: 8.2

REFERENCES

[1]. Breeding, M. (2012). Library systems report: Agents of change. American Libraries, 43(5), 20–27.
[2]. Chowdhury, G. G. (2008). Information Retrieval in the Digital Era. Facet Publishing.
[3]. Madhusudhan, M. (2008). SOUL: Library management software for academic libraries in India. Program: Electronic Library and Information Systems, 42(3), 293–304.
[4]. Pandita, R. (2014). RFID technology and its application in libraries. DESIDOC Journal of Library and Information Technology, 34(2), 134–140.
[5]. Ansari, M. A., & Zuberi, B. M. (2020). Machine learning for personalised book recommendation in library management systems. Journal of Information Science, 46(4), 521–535.