Smart Library System

A

Minor Project

Submitted in partial fulfilment for the award of the degree of

BACHELORS OF TECHNOLOGY

IN

Computer Science Engineering

by

Nandkishore kushwah
BETN1CS22262 &

Devraj Thakur

BETN1CS22288 &

Vineet Rajput (BETN1CS22020) &

Pushpendra Singh Rajput (BETN1CS22271)

Department of Computer Science Engineering

School of Engineering and Technology
ITM UNIVERSITY, GWALIOR - 474026 MP, INDIA
(April 2025)

CERTIFICATE

This is to certify that the work titled "DFAT-An Digital Forensica Automate Tool for extraction from memory dump" submitted by "Nandkishore kushwah, (BETN1CS22262)" in partial fulfilment for the award of the degree of B.Tech. (CSE), ITM University, Gwalior has been carried out under my/our supervision.

To the best of my knowledge and belief, the dissertation

(ii) Is an original piece of work of Candidate.

(ii) It has duly been completed.

(iii) It is up to the standard both in respect of contents and language.

(iv) Free from plagiarism

(v) Work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Guide	Signature of Co-guide
Designation	Designation
Date	Date

DECLARATION

I with this declare that the work entitled “Title of the thesis” submitted to the Department of Computer Science Engineering, School of Engineering and Technology, ITM University, Gwalior (M.P.) is our work done under the supervision of pragya jain. The dissertation doesn’t contain any part which has been submitted for award of any degree either in this University or in any other University.

We further declare that the work is free from any plagiarism.

(Signature of the candidates)

(Name of the candidates)

(Roll Number of the candidates)

Verified by guide

Acknowledgement

Firstly, I thank Lord Almighty for making it possible for me to complete this work. The success and outcome of this project required a lot of guidance and assistance from many people, and I am incredibly privileged to have got it all along with the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank you ……..for providing us an opportunity to do the project work and giving us all support and guidance which made us complete the project duly.He/She took keen interest on our project work and guided us all along and provided all the necessary information for developing a good system.

I would not forget to remember HOD, Mentors,DEAN etc….. for his encouragement and more over for his timely support and guidance till the completion of our project work. I heartily thank our …… for their guidance and suggestions during this project work.

At last but not the least we would like to thank our parents who trusted us and helped us through the errand.

Abstract

The Smart Library Management System is a **web-based MERN application** designed to automate and streamline library operations. It provides a centralized platform for managing books, users, borrowing/returning transactions, and fine collection. The system eliminates manual record-keeping, reduces errors, and improves efficiency in handling large-scale library services.

Key features include:

- Admin management of books, students, and transactions
- Online book issue/return tracking with fine calculation
- Role-based dashboards for admin, librarian, and students
- Online fine payment and digital receipt generation
- AI-powered book recommendations for students

The system is developed using MongoDB, Express.js, React.js, and Node.js, ensuring secure data handling and real-time functionality. It enhances communication, transparency, and accessibility while providing a user-friendly interface. Ultimately, this project makes library operations more efficient and student-friendly.

Contents

Chapter 1

Introduction & Objectives

The rapid advancement of technology has transformed the way libraries operate, shifting from traditional manual record-keeping systems to modern digital solutions. A Smart Library Management System is designed to streamline library activities such as book issuance, returns, catalog management, fine collection, and user authentication. This project aims to develop a fully functional and user-friendly platform that serves both students and administrators by providing secure, reliable, and efficient management of library operations.

The objectives of the system include:

- Automating book borrowing and return processes to reduce manual errors.
- Providing students with an online interface to track issued books, check due dates, and pay fines.
- Assisting administrators in maintaining book records, managing students, and monitoring fine collections.
- Ensuring strong security using authentication and authorization mechanisms.
- Offering scalability so that the system can handle increasing numbers of users and resources.
- Integrating real-time notifications, online payment, and digital receipts for enhanced convenience.

By using modern web technologies, this system ensures scalability, performance, and security, making it suitable for academic institutions of different sizes.

1.1 Technologies Used

1.1.1 React.js (Frontend)

React.js is a widely used open-source JavaScript library developed by Facebook, primarily for building interactive and dynamic user interfaces.

- It enables the creation of reusable components, reducing development time and effort.
- The Virtual DOM ensures high performance by updating only the necessary parts of the web page.
- It supports features like state management, routing, and animations, ensuring a seamless and modern user experience.
- In this system, React.js is responsible for the student and admin dashboards, search functionalities, book listings, payment pages, and fine calculation display.

1.1.2 Node.js (Backend Runtime)

Node.js is a high-performance runtime environment built on the Chrome V8 JavaScript engine.

- It allows developers to use JavaScript on both frontend and backend, ensuring a consistent development experience.
- Its asynchronous, event-driven architecture makes it efficient in handling multiple user requests simultaneously.
- Node.js ensures real-time data updates, which is particularly useful for tracking issued books, overdue fines, and payment confirmations.
- It forms the backbone of the server-side logic, acting as a bridge between the database and the frontend.

1.1.3 Express.js (Server-Side Framework)

Express.js is a lightweight and flexible web application framework for Node.js.

- It simplifies backend development by offering middleware support, routing mechanisms, and HTTP request handling.
- In this project, Express.js is used to develop RESTful APIs that connect the frontend with the database.
- These APIs handle operations such as student login, book issue/return requests, fine calculation, and payment integration.
- Its modular nature ensures the backend remains well-structured and maintainable, reducing complexity in larger applications.

1.1.4 MongoDB (Database)

MongoDB is a NoSQL, document-oriented database designed for flexibility and scalability.

- Unlike traditional relational databases, it stores data in JSON-like documents, making it easier to work with dynamic and unstructured data.
- It is ideal for storing records such as student profiles, book catalogs, issued books, fine transactions, and digital receipts.
- MongoDB's high query performance ensures quick retrieval of data, improving the responsiveness of the system.
- Its scalability allows the system to handle growing amounts of data as more students, books, and transactions are added over time.

1.2 Additional Tools & Features

1.2.1 JWT Authentication

Security is a critical aspect of any library management system, especially when dealing with sensitive information such as student records and payment details. JSON Web Token (JWT) is used for authentication and authorization.

- Once a user logs in, the system generates a unique token that verifies identity for subsequent requests.

- This ensures that only authenticated students and admins can access restricted functionalities.
- JWT also prevents issues like session hijacking and enhances the overall security of the application.
- By using tokens, the system remains stateless, improving scalability and efficiency.

1.2.2 Razorpay/Stripe for Fine Payment

Managing fines is an essential part of library operations, and integrating an online payment gateway simplifies this process.

- Razorpay (widely used in India) and Stripe (global solution) are powerful payment gateways that support multiple methods like UPI, debit/credit cards, and net banking.
- Students can directly pay their fines online without physically visiting the library.
- After successful payment, the system automatically updates the database and generates a digital receipt for transparency and record-keeping.
- This feature improves convenience, reduces manual errors, and ensures financial transactions remain secure and traceable.

1.2.3 GitHub for Version Control

GitHub plays a crucial role in managing the project's source code and collaboration.

- It allows version control using Git, ensuring that changes in the codebase are tracked systematically.
- Developers can work collaboratively, contributing to different modules through branches and pull requests.
- In case of errors, GitHub provides the option to roll back to previous stable versions, ensuring project reliability.
- Additionally, GitHub acts as a cloud backup, preventing code loss and improving deployment workflows.

Chapter 2

Literature Survey

2.1 Review of Existing Systems

Several existing systems have been developed to address the challenges of managing library operations, focusing on automation, digitalization, and user convenience. A few notable contributions are summarized below:

- **Ramesh Kumar et al. (2018)** developed an RFID-based library management system that automated book issuing and returning. By integrating RFID tags with a central database, the system minimized manual errors, reduced waiting time, and improved overall operational efficiency [1].
- **Neha Sharma et al. (2019)** proposed an Android-based mobile library application that enabled students to search books, check availability, and reserve them online. This system enhanced accessibility and reduced dependency on manual catalog searches [2].
- **S. Gupta et al. (2020)** designed a cloud-based digital library management system where book records, member data, and transactions were stored on cloud servers. The system supported remote access, scalability, and improved data security compared to traditional on-premise solutions [3].
- **M. Hassan et al. (2021)** introduced an AI-powered recommendation system integrated with a digital library. It analyzed users' reading patterns and suggested relevant books, improving user engagement and promoting personalized learning [4].
- **Pooja Patel et al. (2022)** developed a web-based library management system with online fine payment integration using PayPal and Razorpay APIs. This system automated overdue fine calculation, allowed secure online transactions, and generated instant payment receipts for students [5].

2.2 Comparison of Features

The table below compares different types of library management systems with the proposed Smart Library Management System:

Feature	Manual System	Basic Digital Systems	Commercial Systems	Proposed System (MERN)
Book Issue/Return	Manual Register	Semi-Automated	Automated	Fully Automated (Online)
Fine Calculation	Manual	Manual Input	Automated	Auto + Integrated Online Payment
Authentication	None	Password Only	Basic Login	JWT-Based Secure Auth
Accessibility	Physical Only	Local Computer	Web-Based	Web + Cloud Support
Payment Integration	Not Available	Not Available	Rarely Integrated	Razorpay/Stripe Payment Gateway
Scalability	Very Low	Limited	Moderate	High (MongoDB + Node.js)
Cost	Very Low	Low	High (License Fees)	Moderate (Open-Source MERN)

This comparison shows that existing systems either lack modern features, are too costly, or are not scalable. The proposed MERN-based system bridges these gaps by integrating modern web technologies, online fine payment, secure JWT authentication, and scalability through a NoSQL database.

2.3 Technologies Used in Existing Systems

Different existing systems use different technological stacks depending on their purpose:

- **Koha** → Built on Perl, MySQL, Apache, and Linux. Reliable but outdated for cloud-native development.
- **Libsys** → Relies on proprietary frameworks with relational databases, offering less flexibility for customization.
- **NewGenLib** → Developed using Java, PostgreSQL, and Tomcat server. Requires heavy resources for deployment.
- **Basic Digital Tools** → Microsoft Excel/Access, or standalone relational databases, limited to local usage.

These systems, while functional, are not designed for modern requirements such as real-time cloud access, secure token-based authentication, online payment integration, and scalability for large datasets.

In contrast, the proposed Smart Library Management System leverages the MERN stack:

- **MongoDB** for scalable and flexible database management.
- **Express.js** for building structured backend APIs.
- **React.js** for a dynamic and interactive frontend.
- **Node.js** for a fast, event-driven backend runtime. Additionally, JWT authentication ensures strong security, and Razorpay/Stripe integration enables online fine payment with automatic receipt generation.

2.4 Summary of Literature Review

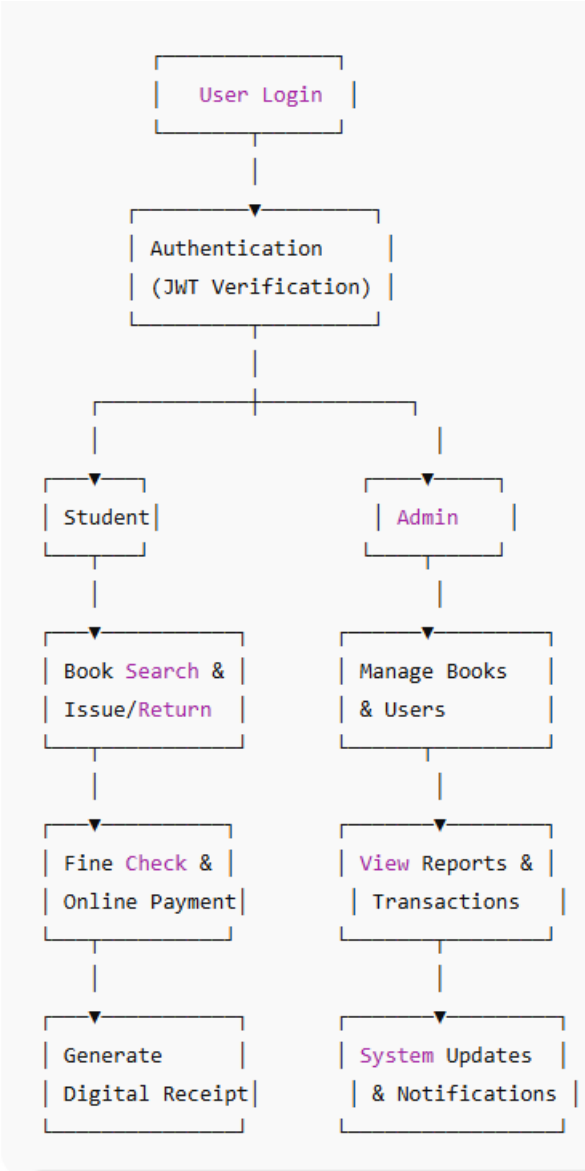
From the above review, the following conclusions can be drawn:

- Traditional manual systems are outdated and cannot meet modern needs.
- Basic digital systems improve efficiency but lack automation, security, and remote access.
- Commercial systems offer better features but are often costly, complex, and not future-proof.
- Most existing systems fail to integrate modern web technologies like JWT authentication, cloud scalability, or online payments.

The proposed Smart Library Management System overcomes these limitations by offering:

- Automation of library tasks (book issue/return, fine calculation).
- Security through JWT authentication.

2.5 Flow chart



Chapter 3

Research Gap & Proposed System

3.1 Research Gap Identified

Existing library management systems often face several challenges that limit their efficiency and scalability. The main research gaps identified are:

- **Limited Automation:** Most traditional systems lack AI-based recommendations for personalized book suggestions.
- **Manual Fine Management:** Fine calculation and payment are often handled manually, increasing errors and inefficiency.
- **Weak User Engagement:** Students cannot easily track their issued books, due dates, or get real-time notifications.
- **Lack of Smart Analytics:** Existing systems do not provide insights into book usage trends, most borrowed books, or resource optimization.
- **Outdated Architectures:** Many systems still rely on monolithic architectures instead of scalable web-based solutions like MERN with REST APIs.

Thus, there is a strong need for a smart, automated, scalable, and AI-driven Library Management System that improves user experience and operational efficiency.

3.2 Proposed System Architecture

The proposed Smart Library Management System is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) with AI-based features and RESTful API integration.

Key Components:

1. **Frontend (React.js):** Provides a user-friendly interface for students, librarians, and administrators.
2. **Backend (Node.js + Express.js):** Manages API endpoints, authentication, fine calculation, notifications, and AI book recommendation logic.

3. **Database (MongoDB):** Stores user profiles, books, transactions, fines, and activity logs.
4. **AI Recommendation Engine:** Suggests books based on borrowing history and user preferences.
5. **Payment Gateway Integration:** Enables secure online fine payments.
6. **Notification Service:** Sends reminders about due dates, fines, and new arrivals via email/SMS.

3.3 Functional Requirements

The system provides the following functional features:

- **Student Panel**
  - Search and borrow books.
  - View issued books, due dates, and fine details.
  - Pay fines online and download receipts.
  - Receive AI-powered book recommendations.
- **Admin Panel**
  - Manage books (add/update/delete).
  - Manage students and staff.
  - Track transactions and generate reports.
  - Approve/reject requests.
- **Librarian Panel**
  - Issue/return books.
  - Manage fines and generate receipts.
  - Monitor book availability and stock.

3.4 Non-Functional Requirements

The system also ensures the following quality attributes:

- **Scalability:** Supports large datasets of books and users.
- **Security:** Implements JWT-based authentication, password hashing, and role-based access control.
- **Performance:** Fast response through optimized queries and REST APIs.
- **Reliability:** Ensures data consistency and regular backups.
- **Usability:** Simple and interactive UI for all types of users.
- **Maintainability:** Follows modular MVC and REST design for easy updates.

3.5 D. Database Schema

1. Users Collection

- **user_id (PK)**
- name
- email
- password (hashed)
- role (student, librarian, admin)
- borrowed_books[] (list of book_ids)
- fines[] (list of fine_ids)

2. Books Collection

- **book_id (PK)**
- title
- author
- category

- ISBN
- availability_status (available/issued)
- added_date

3. Transactions Collection

- **transaction_id (PK)**
- user_id (FK → Users)
- book_id (FK → Books)
- issue_date
- due_date
- return_date
- fine_amount

4. Fines Collection

- **fine_id (PK)**
- user_id (FK → Users)
- transaction_id (FK → Transactions)
- amount
- status (paid/unpaid)
- payment_date

5. Notifications Collection

- **notification_id (PK)**
- user_id (FK → Users)
- message
- timestamp

- status (read/unread)

6. Complaints/Feedback Collection (Optional)

- **complaint_id (PK)**
- user_id (FK → Users)
- subject
- message
- status (open/resolved)
- timestamp

3.6 Model-View-Controller (MVC) Architecture

- **Model:** Represents the database layer (MongoDB collections for users, books, fines, transactions).
- **View:** React.js frontend where students, librarians, and admins interact.
- **Controller:** Express.js + Node.js server handling requests, applying business logic, and communicating with the database.

Example:

- Student clicks "Borrow Book" (View) → Request goes to Controller → Controller checks availability in Model (DB) → Updates database → Response sent back to View.

REST Architecture

The system follows RESTful API design for communication between frontend and backend.

- **GET /books** → Fetch list of books.
- **POST /books** → Add a new book.
- **PUT /books/:id** → Update book details.
- **DELETE /books/:id** → Delete a book.
- **POST /fines/pay** → Fine payment processing.

Advantages of REST in the System:

- Stateless and scalable communication.
- Platform-independent (can be integrated with mobile apps).
- Standardized CRUD operations for easy maintenance

Chapter-4

Results and Discussions

This chapter presents the actual outcomes of the **Smart Library Management System** implementation, highlighting technical details, developed modules, user interface snapshots, testing methodologies, and evaluation of performance and user satisfaction.

4.1 Implementation Details

The Smart Library System was practically designed and developed using the MERN stack, ensuring scalability, security, and user-friendliness. The following subsections describe the implementation in detail:

Technologies Used

- **Frontend:** React.js, HTML5, CSS3, JavaScript (Bootstrap & Material UI for styling)
- **Backend:** Node.js with Express.js (REST API-based architecture)
- **Database:** MongoDB (NoSQL document-based database)
- **APIs & Integrations:**
    - Nodemailer API – Email notifications (due date reminders, fines, new arrivals)
    - Payment Gateway API – Online fine payment
    - JWT Authentication – Secure login & session handling

Architecture Used

The system follows **a Three-tier MVC + REST Architecture:**

1. **Presentation Layer (React.js Frontend):** Provides an interactive user interface for students, librarians, and administrators.

2. **Application Layer (Node.js + Express.js):** Handles business logic, authentication, book recommendation engine, and API routing.
3. **Data Layer (MongoDB):** Stores user profiles, books, transactions, fines, and notifications.

Development Environment

- **Server:** Node.js runtime (Express.js framework)
- **Database Tool:** MongoDB Atlas / Compass
- **IDE & Tools:** VS Code, Postman for API testing
- **Version Control:** GitHub for source code management
- **Deployment:** Vercel (frontend), Render/Heroku (backend), MongoDB Atlas (cloud database)

Key Implemented Modules

1. **Admin Dashboard**
    - Manage books (add/update/delete).
    - View issued/returned transactions.
    - Generate analytical reports.
2. **Student Panel**
    - Search and borrow books.
    - View issued books, due dates, and fines.
    - Pay fines online and download receipts.
    - Get AI-based book recommendations.
3. **Librarian Panel**
    - Manage book inventory.
    - Issue/return books to students.

- Track late returns and fines.

4. **Notifications Module**
    - Email alerts for due dates, fines, and new arrivals.
    - In-app notifications for borrowing/returning updates.
5. **Login/Signup & Authentication**
    - Role-based authentication (student, librarian, admin).
    - Password hashing using bcrypt.
    - Secure session management with JWT.

4.2 Interface Screenshots

This subsection presents the user interface (UI) screenshots of the **Smart Library Management System** to demonstrate the functionality, design, and ease of use of the developed system. Each screenshot is captioned to highlight the module or feature being displayed.

1. Login and Signup Pages

Screenshots of the Login and Signup forms for different user roles (Admin, Student, Librarian).

- **Features:**
    - Secure authentication with JWT.
    - Role-based redirection (student → Student Dashboard, admin → Admin Panel, librarian → Librarian Panel).
    - Password hashing using bcrypt for security.

2. Admin Panel Interface

The Admin Dashboard provides control over book inventory, user management, and system monitoring.

- **Key Features:**
    - Add/Update/Delete books.
    - Manage users (students, librarians).
    - View and respond to complaints.
    - Generate reports and monitor system usage.

3. Student Panel

The Student Dash**board** allows students to interact with the library system.

- **Key Features:**
    - Search and borrow books.
    - View issued books and due dates.
    - Check fines and make online payments.
    - Submit complaints and feedback.
    - Receive AI-based book recommendations.

4. Librarian Panel

The **Librarian Panel** is designed for day-to-day library staff operations.

- **Key Features:**
    - Issue and return books.
    - Manage book stock and availability.

5. Notification System

The system provides real-time notifications through in-app alerts and email.

- **Key Features:**
    - Email reminders for due dates and fines.
    - Pop-up notifications for borrowing/return confirmations..

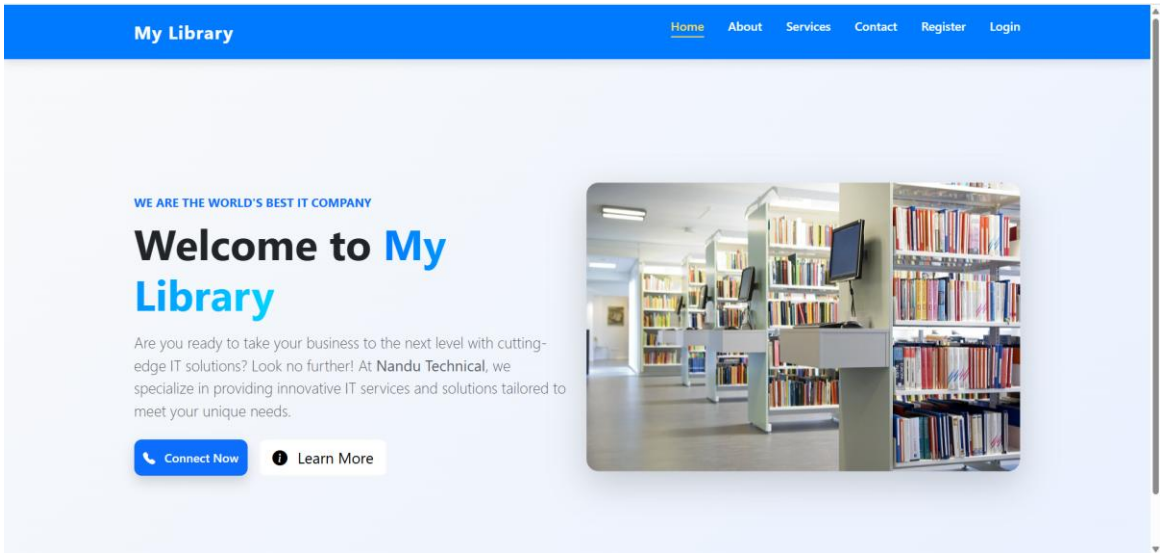


Figure: 4.2.1

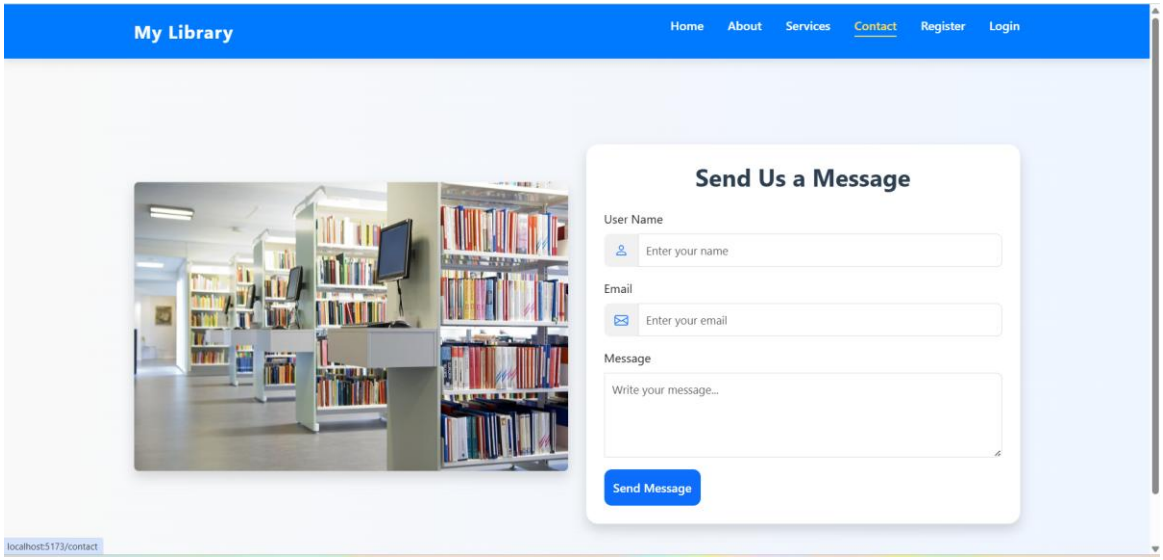


Figure:4.2.2

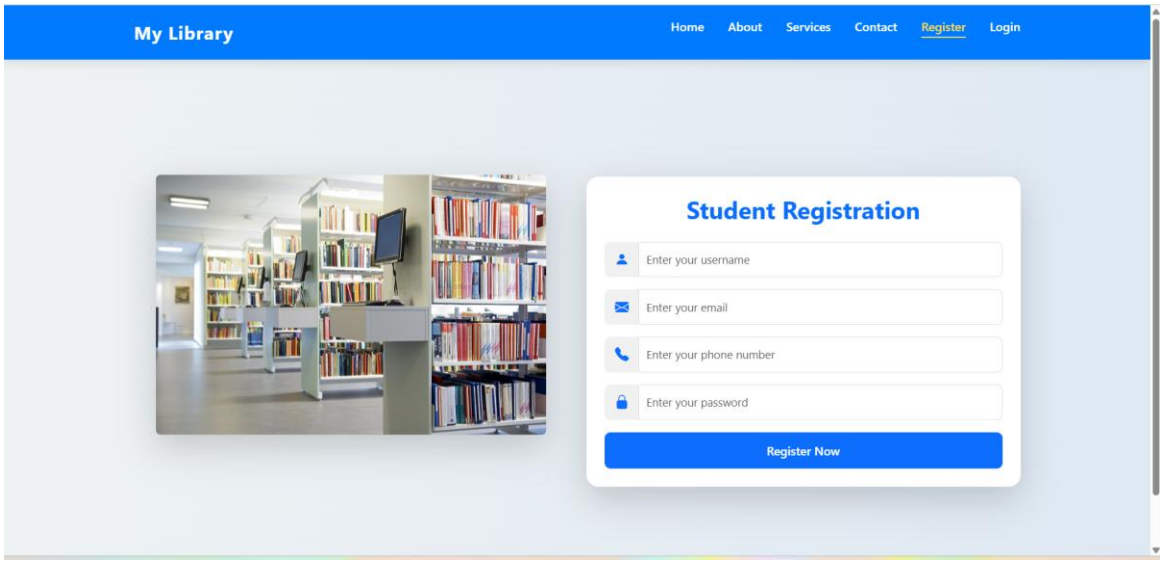


Figure:4.2.2

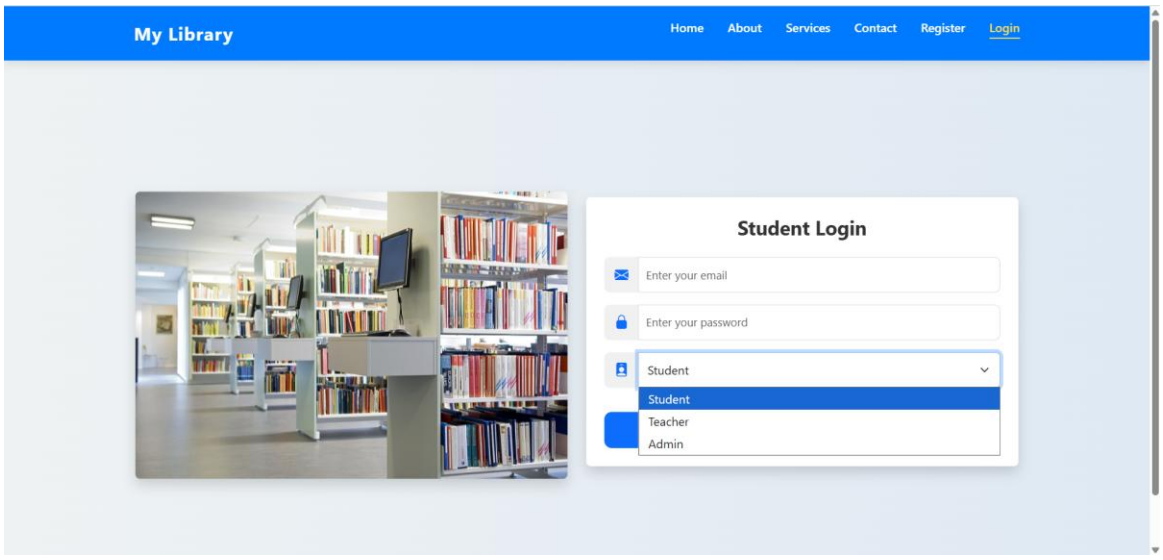


Figure:4.2.3

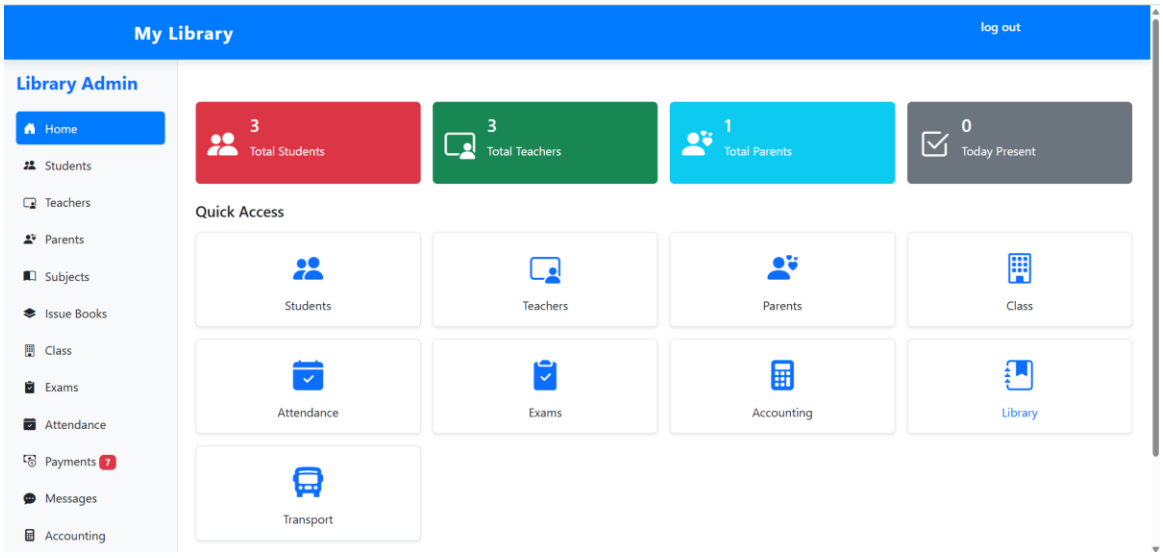


Figure :4.2.4

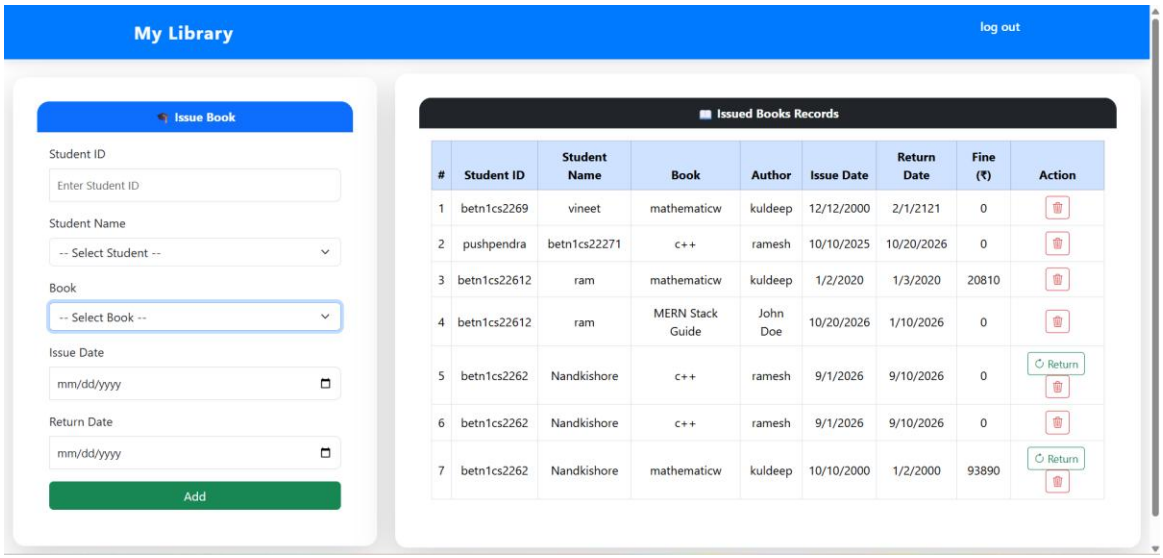


Figure:4.2.5

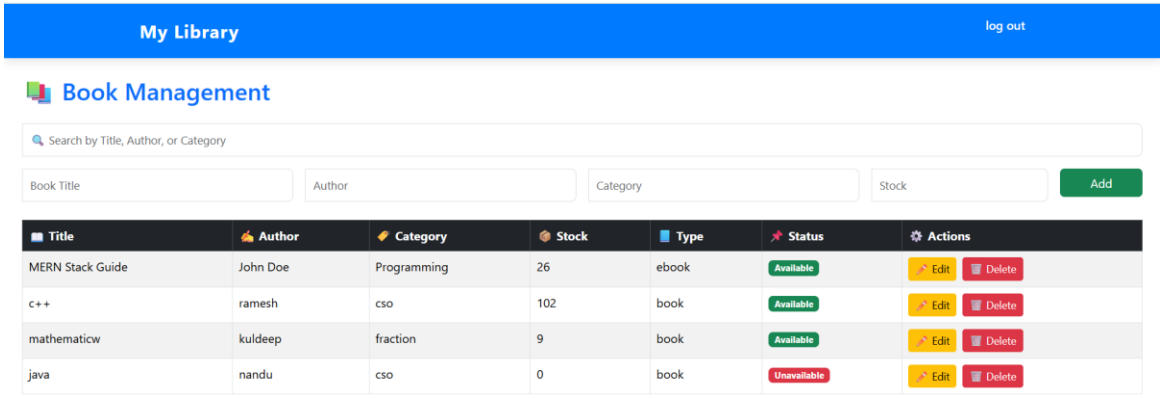


Figure: 4.4.6

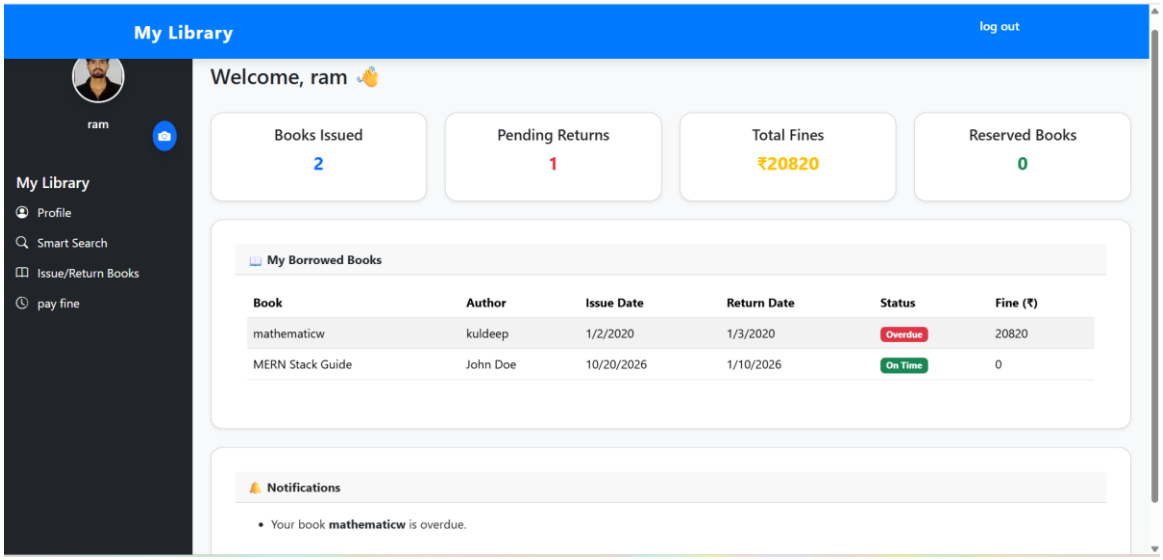


Figure:4.4.7

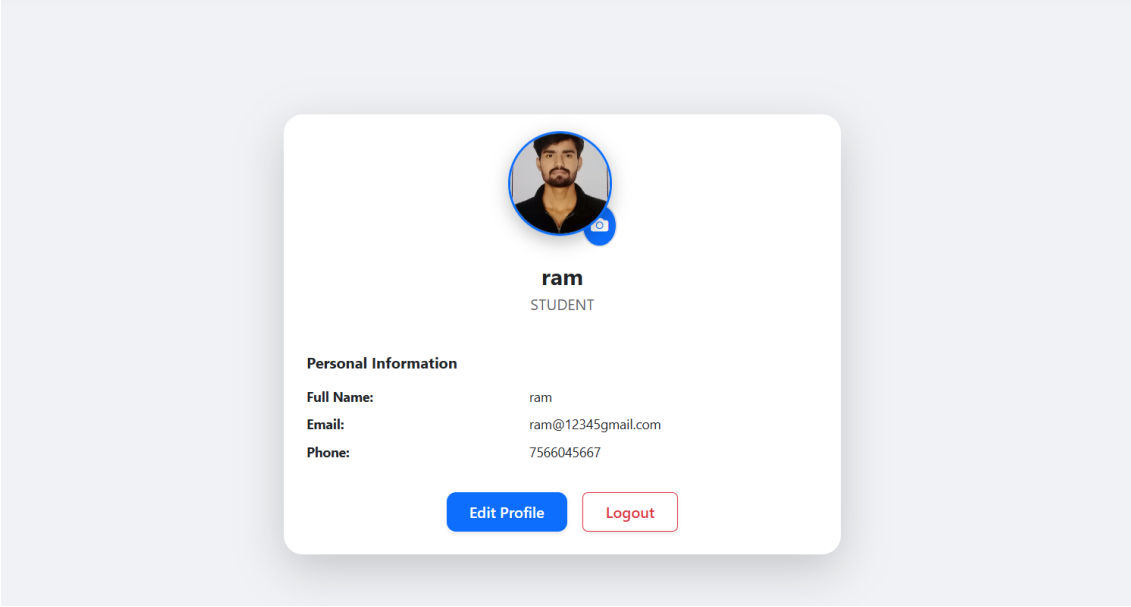

ram
STUDENT
Personal Information
Full Name:
ram
Email:
ram@12345gmail.com
Phone:
7566045667
Edit Profile
Logout

Figure:4.4.8

ID: STU123

Name: Nandkishore Kushwah

Email: nandkishore@example.com

Fine Details

Book	Due Date	Returned Date	Fine (₹)
Mathematics	01/09/2026	10/09/2026	50
C++ Programming	01/09/2026	10/09/2026	30

Total Fine: ₹80

Pay via UPI QR Code

UPI ID: example@upi

Pay Now

Figure:4.4.8

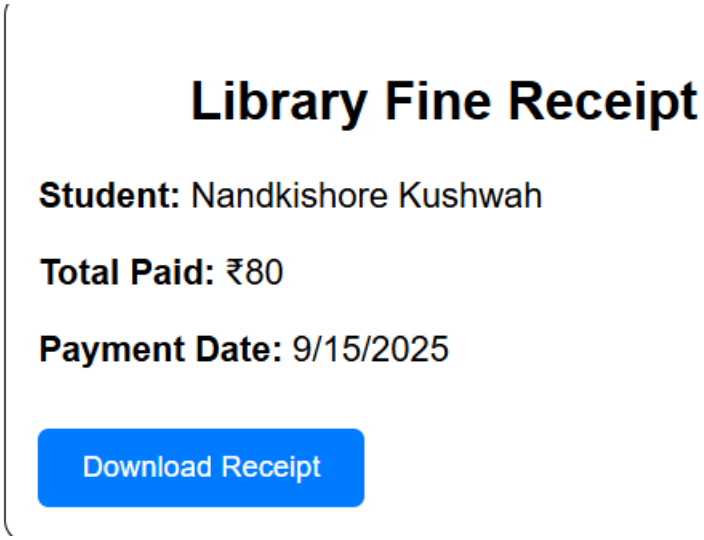

Library Fine Receipt
Student: Nandkishore Kushwah
Total Paid: ₹80
Payment Date: 9/15/2025
Download Receipt

Figure :4.4.9

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The Smart Library Management System was successfully implemented using the MERN stack (MongoDB, Express.js, React.js, Node.js) to digitize and automate traditional library tasks. The system provides a secure, scalable, and user-friendly platform for managing books, transactions, fines, and notifications.

Key achievements of the system include:

- **Efficient Book Management:** Admins and librarians can easily add, update, and track books.
- **Automated Fine Calculation:** The system automatically tracks overdue books and generates fines with online payment support.
- **User Engagement:** Students can view their borrowed books, due dates, receive recommendations, and get real-time notifications.
- **Scalability and Security:** Implemented REST APIs, JWT-based authentication, and password hashing to ensure secure multi-role access.
- **Improved Accessibility:** The system is web-based, making it accessible anytime and anywhere.

Overall, the project demonstrates how modern web technologies can improve library operations, enhance student satisfaction, and reduce manual workload.

5.2 Future Work

While the current system achieves its core objectives, several enhancements can be integrated to make it more powerful and user-friendly:

1. **Mobile Application (React Native):**
    - Develop a cross-platform mobile app for Android and iOS to improve accessibility.

2. **QR/Barcode Book Scanning:**
    - Integrate barcode/QR code scanning for faster book issue/return and inventory management.
3. **Integration with University ERP:**
    - Connect with existing ERP systems to synchronize student data, attendance, and library records.
4. **Advanced AI Recommendation Engine:**
    - Improve book recommendations using NLP and machine learning techniques (e.g., collaborative filtering, BERT-based models).
5. **Voice Assistant for Book Search:**
    - Add a voice-enabled assistant for hands-free searching and improved accessibility.
6. **Cloud Deployment (AWS/GCP):**
    - Deploy the system on cloud platforms for better performance, scalability, and real-time backups.

References

[1] Dhanalakshmi, R., & Radha, P. (2020). Smart Library Management System using IoT. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, Vol. 9 Issue 4.

[2] Gaurav Kumar, N., & Ashish, M. (2019). Web-Based Digital Library Management System. *International Journal of Computer Science and Mobile Computing (IJCSMC)*, Vol. 8 Issue 6.

[3] Ahmed, S., & Rahman, M. (2021). A Cloud-Based Library Management System for Academic Institutions. *International Journal of Emerging Trends in Engineering Research (IJETER)*, Vol. 9 Issue 8.

[4] Patel, R., & Singh, P. (2018). Library Automation and Management through Web Applications. *International Journal of Computer Applications (IJCA)*, Vol. 182 No. 16.

[5] MongoDB. (n.d.). MongoDB Documentation. Retrieved from https://www.mongodb.com/docs/

[6] ReactJS. (n.d.). React Official Documentation. Retrieved from https://react.dev/

[7] Node.js Foundation. (n.d.). Node.js Documentation. Retrieved from https://nodejs.org/en/docs/

[8] Express.js. (n.d.). Express Official Documentation. Retrieved from https://expressjs.com/

[9] W3Schools. (n.d.). MERN Stack Tutorial. Retrieved from https://www.w3schools.com/mern/

[10] GeeksforGeeks. (n.d.). MERN Stack Development – Introduction. Retrieved from https://www.geeksforgeeks.org/mern-stack/

[11] Bootstrap. (n.d.). Introduction to Bootstrap. Retrieved from https://getbootstrap.com/

[12] TutorialsPoint. (n.d.). RESTful Web Services Tutorial. Retrieved from https://www.tutorialspoint.com/restful/index.htm

[13] IEEE Xplore. (2020). An Intelligent Library Information System with Book Recommendation Features. Retrieved from https://ieeexplore.ieee.org/

[14] SpringerLink. (2021). Digital Libraries and Knowledge Organization. Retrieved from https://link.springer.com/

[15] ResearchGate. (2019). Design and Implementation of Smart Library System Using Web Technologies. Retrieved from https://www.researchgate.net/