Engineering and Scientific International Journal (ESIJ)
Volume 13, Issue 2, April – June 2026
ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

Smart Library Management System with Automated Borrowing Workflow, Penalty Computation and AI based Recommendation using Django Framework

P. Soma Sekhar [*1], B. Shireeshai[2], Dr. S. Usharani[3]

[1]*Student, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[2]*Associate Professor, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[3]*Professor & Head, Department of Applications, Viswam Engineering College, Andhra Pradesh, India*
Email id: [2]shireeshabusireddyurl@gmail.com

Abstract — The increasing demand for efficient information management in educational institutions has highlighted the limitations of traditional library systems, which rely heavily on manual processes for cataloguing, borrowing, and record maintenance. This paper presents a Smart Library Management System, a web-based solution designed to automate and optimize library operations while enhancing accessibility and user experience. The system is developed using the Django framework with Python as the backend language and SQLite for data persistence. The frontend interface is built using HTML5, CSS3, JavaScript, and Bootstrap, ensuring a responsive and user-friendly design. The system supports dual-role functionality, enabling library members to browse books, request borrowing, access digital resources, and receive notifications, while librarians manage inventory, approve requests, track loans, and handle overdue penalties. A key feature of the system is the automated borrowing workflow, which includes real-time book availability tracking, request approval mechanisms, and penalty computation for overdue returns. Additionally, an optional AI-powered recommendation module utilizing the Google Gemini API enhances user engagement by suggesting relevant books based on borrowing history. Experimental evaluation demonstrates improved operational efficiency, reduced administrative workload, and enhanced user satisfaction compared to traditional systems. The proposed system offers a scalable, cost-effective, and deployable solution for modern educational libraries.

Keywords — *Library Management System; Django; Automation; Borrowing Workflow; Penalty Calculation; AI Recommendation; Web Application.*

1. Introduction

Libraries play a crucial role in academic and research environments by providing access to knowledge resources. However, many institutional libraries continue to rely on manual or semi-automated systems that limit operational efficiency and user convenience. Traditional systems involve paper-based catalogues, manual record-keeping, and in-person interactions for borrowing and returning books. These processes are time-consuming, prone to human error, and lack real-time accessibility. Users often face challenges in checking book availability remotely, while librarians struggle with maintaining accurate records and managing overdue penalties.

With the advancement of web technologies and database systems, there is a significant opportunity to modernize library operations through digital transformation. A web-based Library Management System can automate workflows, enable remote access, and provide real-time updates, thereby improving service quality. The proposed Smart Library Management System addresses these challenges by integrating automated request processing, digital resource access, and intelligent notification mechanisms. The inclusion of AI-based recommendation features further enhance user engagement and resource utilization.

2. Literature survey

Library management systems have evolved from manual record-keeping to digital catalogues and integrated management platforms. Early systems focused on digitizing inventory but lacked workflow automation and user interaction capabilities. Modern systems such as Koha provide comprehensive features but often require complex deployment and maintenance.

Database-driven systems using relational databases like SQLite have improved data management but still lack intelligent automation. Recent advancements incorporate Artificial Intelligence to enhance recommendation systems.

APIs such as Google Gemini enable personalized suggestions based on user behavior, improving engagement. However, most existing systems do not integrate automation, digital resource access, and AI capabilities into a single lightweight platform. The proposed system bridges this gap by providing a unified, scalable solution.

DOI: 10.30726/esij/v13.i2.2026.1320032

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

3. System Architecture and Design

3.1 System Architecture

The Smart Library Management System follows a three-tier architecture combined with Django's Model-View-Template (MVT) design pattern, ensuring modularity, scalability, and maintainability. The system follows a three-layer architecture consisting of presentation, application, and data layers. The presentation layer manages user interaction using HTML5, CSS3, Bootstrap, and JavaScript, providing interfaces such as user and librarian dashboards, book catalog, PDF viewer, and notifications. The application layer is implemented using the Django framework, where components like views.py, urls.py, and admin.py handle request processing, routing, authentication, book transactions, penalty calculation, and notifications. The data layer uses SQLite for structured storage of book records, user details, borrow history, and requests. Additionally, PDF files are stored in the filesystem, ensuring efficient data management and accessibility.

3.2 System Flow Diagram

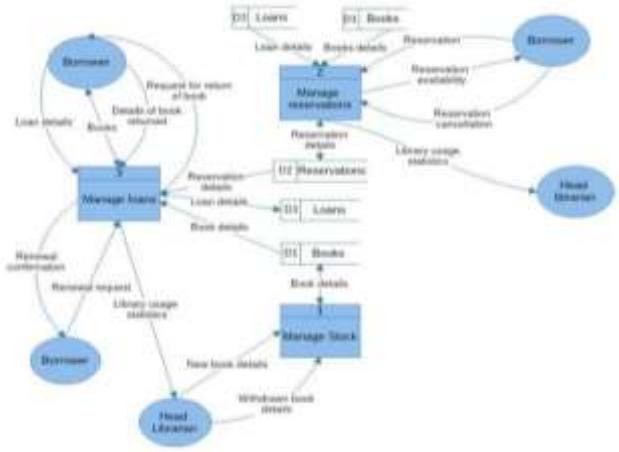

Fig. 1: Library System DFD Level 1

The system operates through two interconnected workflows:
User Workflow
1. User logs in
2. Browses book catalogue
3. Selects book
4. Sends request
5. Receives notification
6. Waits for approval
7. Borrows book
8. Returns book (penalty if late)

Librarian Workflow
- Admin logs in
- Views pending requests
- Approves or rejects request
- System creates borrow record
- Tracks active loans
- Records return
- System calculates penalty

System Processing Flow
- Request → Approval → Borrow Record
- Return → Penalty Calculation
- Events → Notifications

3.3 UML Diagrams

3.3.1 Use Case Diagram

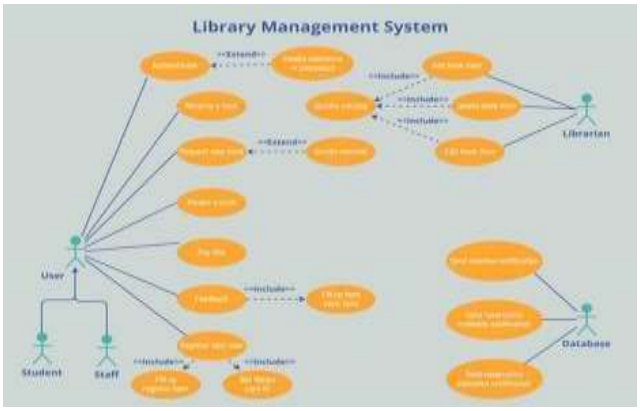

Fig. 2: Use case Diagram Library Management system

Actors
- Library Member (User)
- Librarian (Admin)

User Use Cases
- Register/Login
- Browse Books
- Request Book
- View Notifications
- Track Borrowing History
- Read PDF Book

Librarian Use Cases
- Add Book
- Manage Inventory
- Approve/Reject Requests
- Track Loans
- Record Returns

3.3.2 Class Diagram

Core Classes
- Book
- BookRequest

DOI: 10.30726/esij/v13.i2.2026.1320032

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

- BorrowRecord
- Notification
- User

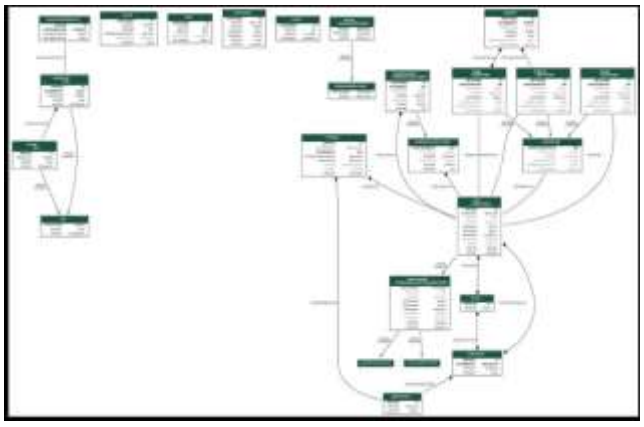

Fig. 3: Class diagram

Relationships
- User → BookRequest (One-to-Many)
- User → BorrowRecord (One-to-Many)
- Book → BookRequest (One-to-Many)
- Book → BorrowRecord (One-to-Many)
- User → Notification (One-to-Many)

4. Implementation

4.1 System Modules

The Smart Library Management System is implemented using a modular architecture within the Django framework. Each module is designed to automate a specific functionality, ensuring efficient workflow execution and system scalability.

❖ User Management Module handles:
- User registration and login
- Authentication and session management
- Role-based access (User / Librarian)

Django's built-in authentication ensures secure password handling and user validation.

❖ Book Management Module responsible for:
- Adding and updating book records
- Managing book inventory
- Uploading PDF resources

Books are stored with metadata such as title, author, category, and availability status.

❖ Book Request Module handles:
- Book request submission by users
- Request tracking
- Approval/rejection workflow

Each request is linked to both the user and the selected book.

❖ Borrowing Management Module manages:
- Creation of borrow records
- Tracking borrowed books
- Monitoring due dates

This module ensures accurate tracking of issued books.

❖ Return and Penalty Module handles:
- Book return process
- Late return detection
- Automatic penalty calculation

Penalty is calculated based on:
- Number of overdue days
- Predefined fine rate

❖ Notification Module provides:
- Alerts for request status
- Notifications for approvals
- Reminders for due dates

Improves user engagement and system usability.

❖ Digital Library Module enables:
- Viewing PDF books
- Accessing digital content

Files are served from the system storage.

❖ AI Recommendation Module (Optional)

This module integrates AI APIs such as Google Gemini to:
- Suggest books based on user history
- Enhance reading experience

4.2 Detailed Implementation

4.2.1 Request and Approval Workflow

The borrowing workflow is implemented as follows:
- User submits book request
- Request stored in database
- Librarian reviews request
- Approval triggers borrow record creation
- Notification sent to user

4.2.2 Borrow Record Management

Each borrow record contains:
- User ID
- Book ID
- Borrow date
- Due date
- Return status

This ensures complete tracking of borrowing activities.

DOI: 10.30726/esij/v13.i2.2026.1320032

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

4.2.3 Penalty Calculation Logic

Penalty is computed based on overdue duration:
Penalty=Overdue Days×Daily FinePenalty = \text{Overdue Days}\times\text{Daily Fine}Penalty=Overdue Days×Daily Fine
Overdue Days = Return Date – Due Date
This logic is implemented within the backend using Python.

4.2.4 Database Implementation

The system uses SQLite with the following tables:
- User
- Book
- BookRequest
- BorrowRecord
- Notification

Django ORM ensures efficient data operations and relationship handling.

4.2.5 Notification System

Notifications are generated for:
- Request approval/rejection
- Due date reminders
- Return confirmation

Displayed in user dashboard.

5. Results and performance analysis

5.1 Performance Analysis

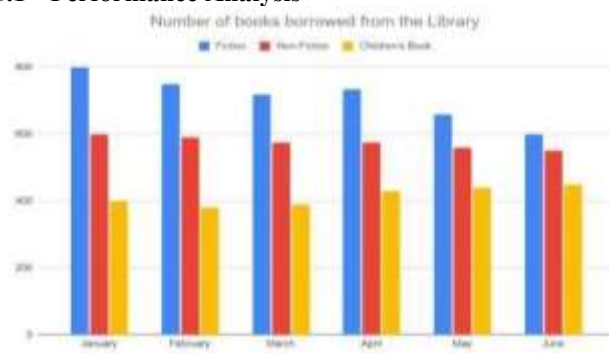

Fig. 5: Bar Diagram

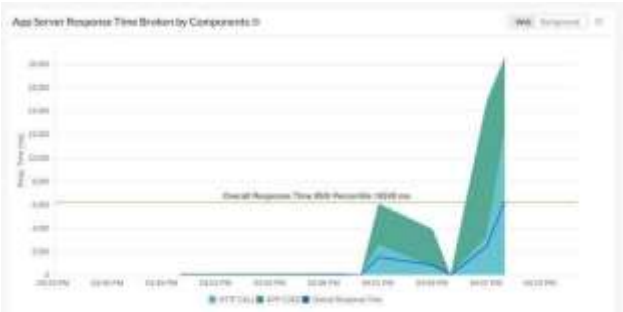

Fig. 6: Detailed Analysis

The system was tested using simulated library operations to evaluate performance and efficiency.
Observations
- Fast Processing: Request approval and borrow operations are instantaneous
- Accurate Tracking: No data inconsistency in borrow records
- Efficient Workflow: Automated processes reduce manual effort

5.2 Analytical Results

Borrowing Trends
- Peak borrowing during academic sessions
- Increased usage during exam periods

Overdue Analysis
- Percentage of delayed returns identified
- Helps improve policy decisions

System Efficiency
- Reduced manual workload
- Faster processing compared to traditional systems

User Engagement
- Increased usage of digital library features
- Positive feedback on notification system

5.3 Discussion

The Smart Library Management System demonstrates significant improvements over traditional systems.
Advantages
- Automation of workflows
- Real-time tracking
- Reduced administrative workload
- Improved user experience

Limitations
- Limited scalability (SQLite)
- Basic AI recommendation
- No mobile application

Despite limitations, the system is highly effective for academic institutions.

6. Conclusion

The proposed Smart Library Management System effectively addresses the limitations of traditional library operations by introducing a fully automated, web-based solution that enhances efficiency, accuracy, and user accessibility. By leveraging modern web technologies and structured database management, the system successfully digitizes key library functions such as book cataloguing, borrowing, returning, and record maintenance. The implementation using the Django framework ensures a

DOI: 10.30726/esij/v13.i2.2026.1320032

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

modular and scalable architecture, while the use of SQLite provides a lightweight yet efficient data storage solution suitable for small to medium-scale deployments. The automated borrowing workflow, combined with real-time notifications and penalty computation, significantly reduces manual effort and administrative overhead for librarians. The inclusion of a digital library module enables users to access PDF resources remotely, improving knowledge accessibility. Furthermore, the optional integration of AI-based recommendation features using Google Gemini enhances user engagement by providing personalized reading suggestions. Experimental results indicate that the system improves operational efficiency, minimizes human errors, and enhances user satisfaction compared to traditional methods. The system demonstrates reliability, usability, and practical applicability in real-world academic environments. Overall, the Smart Library Management System provides a cost-effective, scalable, and user-friendly solution that aligns with the digital transformation needs of modern educational institutions.

7. Future Enhancements

The Smart Library Management System provides a strong foundation for digital automation, but several enhancements can further improve its performance and scalability. Advanced recommendation systems using collaborative and content-based filtering can deliver personalized suggestions. Migrating from SQLite to scalable databases like MySQL or PostgreSQL with caching (Redis) will improve performance. Cloud deployment using platforms such as AWS or Azure ensures high availability and scalability. Mobile application development for Android and iOS can enhance accessibility with push notifications. Integration of RFID and barcode technologies enables faster transactions and automated inventory tracking. Real-time notifications through WebSockets, email, and SMS improve communication. Expanding digital content to include e-books and audiobooks transforms the system into a learning platform. Security can be strengthened with multi-factor authentication and encryption. Integration with academic systems and advanced analytics will support better decision-making and continuous system improvement.

.

References

[1] Django Software Foundation, *Django Documentation*. [Online]. Available: https://docs.djangoproject.com/. Accessed: Jul. 7, 2026.
[2] SQLite Consortium, *SQLite Documentation*. [Online]. Available: https://www.sqlite.org/docs.html.
[3] Google, *Gemini API Documentation*. [Online]. Available: https://ai.google.dev/gemini-api/docs.
[4] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
[5] Bootstrap Team, *Bootstrap Documentation*. [Online]. Available: https://getbootstrap.com/docs/.
[6] Python Software Foundation, *Python Documentation*. [Online]. Available: https://docs.python.org/3/.
[7] PostgreSQL Global Development Group, *PostgreSQL Documentation*. [Online]. Available: https://www. postgresql. org/docs/.
[8] Amazon Web Services, *AWS Documentation*. [Online]. Available: https://docs.aws.amazon.com/.
[9] Docker Inc., *Docker Documentation*. [Online]. Available: https://docs.docker.com/.
[10] The Kubernetes Authors, *Kubernetes Documentation*. [Online]. Available: https://kubernetes.io/docs/.
[11] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*, 3rd ed. Hoboken, NJ, USA: Wiley, 2010.
[12] M. K. Buckland, "Barcode technology in libraries," *Library Hi Tech*, vol. 8, no. 3, pp. 7–15, 1990.
[13] A. Breeding, "Library management systems: Current trends and future directions," *Computers in Libraries*, vol. 42, no. 5, pp. 12–18, 2022.
[14] X. Liu, Y. Wang, and J. Zhang, "Digital library systems: Architecture, technologies and challenges," *Journal of Information Science*, vol. 47, no. 6, pp. 755–770, Dec. 2021.
[15] IEEE Computer Society, *IEEE Standard for Software and System Test Documentation*, IEEE Std 829-2008, Jul. 2008.

DOI: **10.30726/esij/v13.i2.2026.1320032**