LIBRARY MANAGEMENT SYSTEM

TRAINING PROJECT REPORT

Submitted by

Tamanna (23BCS11010)
Khushi (23BCS11401)
Anshuma (23BCS14026)
Sejal (23BCS10668)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

October – 2025

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report **"LIBRARY MANAGEMENT SYSTEM"** is the Bonafide work of "Tamanna (23BCS11010), Khushi (23BCS11401) Anshuma (23BCS14026) and Sejal(23BCS10668) **"** who carried out the project work under my supervision.

SIGNATURE **SIGNATURE**

SUPERVISOR

HEAD OF THE DEPARTMENT
CSE

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

INTRODUCTION

1.1. Client Identification/Need Identification/Identification of relevant Contemporary issue

Libraries have evolved beyond mere repositories of books—they now manage digital content, media resources, patron services, and administrative tasks. Both educational institutions and public libraries face challenges in managing a large-scale library database that includes thousands of books and hundreds of users.

The ongoing digital transformation across various sectors highlights the need for modernization in library systems. The identified clients include:

- Educational Institutions (schools, colleges, universities) that manage both student and faculty access to resources.
- Public Libraries looking to digitize their systems for better public utility and accessibility.
- Corporate or Private Libraries that require secure and customized solutions for internal use.

The contemporary issue arises from the need to replace manual or semi-automated systems with a full-fledged, scalable, and efficient platform. Driven by the need to improve access, accuracy, and control, the development of this Library Management System is a response to such modern challenges.A digital library management tool will not only eliminate inefficiencies but also help these institutions provide better services to their users by reducing delays, improving transparency, and offering online functionalities.

1.2. Identification of Problem

Multiple issues were identified in the current systems used by libraries, either manual or outdated digital solutions. These include:

- Lack of Real-Time Updates: Traditional systems do not update book availability in real-time, often resulting in confusion over issued and available materials.
- Manual Data Entry Errors: Human error during registration, book entry, or transaction logging can lead to data inconsistencies and loss.
- Time-Consuming Processes: Book issuance and return are not instantaneous, requiring manual verification and record keeping.
- Inefficient Search Features: Locating books or borrower records consumes excessive time without proper indexing or search mechanics.
- Poor User Interface (UI): Many existing systems have outdated UIs, making navigation difficult for both administrators and users.
- Limited Accessibility: Desktop-based or on-premise systems don't allow remote access or online querying of library catalogs.

These problems collectively interrupt smooth library operations and result in frustration for users as well as for library staff. Hence, there's an urgent need for a better, automated system.

1.3. Identification of Tasks

To design a solution capable of addressing the above problems, the project includes several crucial tasks:

Requirement Analysis

- Interacting with potential users (students, librarians, faculty).
- Understanding current workflows and identifying key problem areas.
- Shortlisting essential features like book search, issuing, member registration, and fine calculations.

System Design

- Creating database schemas for books, users, and transaction logs.
- Designing RESTful backend services using Spring Boot.
- Creating dynamic HTML views using Thymeleaf, aligned with the MVC architecture.
- Planning role-based access for admin and user functionalities.

Implementation Tasks

- User Authentication using Spring Security (e.g., login, registration with role distinction).
- Book Management: Add/edit/delete books, track availability.
- Member Management: User profile creation, activity tracking.
- Book Issuance and Return: Date-based tracking, late fine calculations.
- Search Module: Real-time book and member search with filters.

- Notifications and Alerts: For due dates, system errors, and changes.

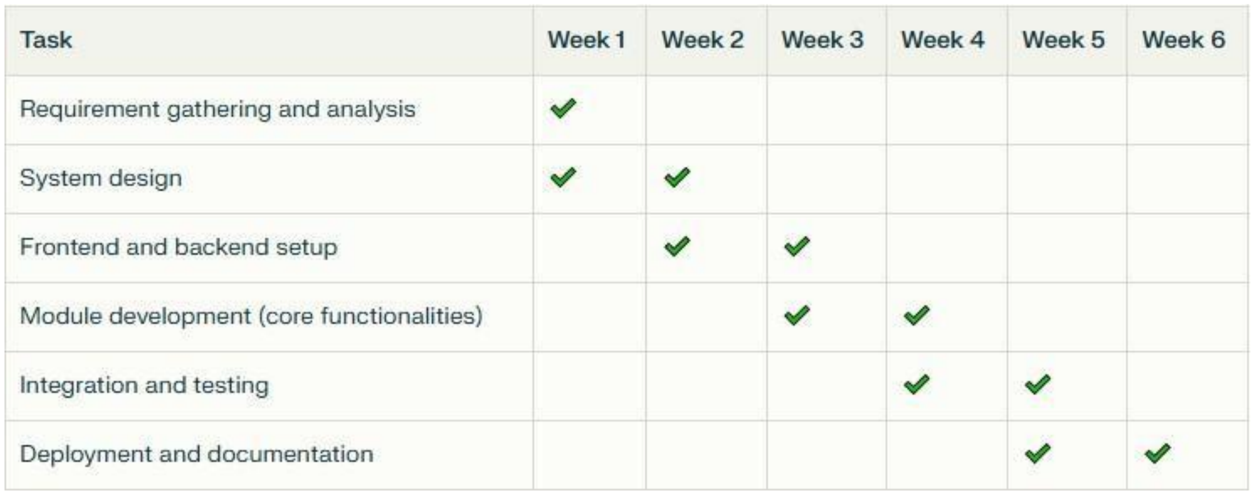
Testing and Debugging

- Unit testing each module to ensure functionality.
- Integration testing to ensure modules work harmoniously.
- Manual and automated testing for UI usability.

Deployment

- Deploying the web-based application on a local or cloud server.
- Preparing documentation for installation and usage.

1.4. Timeline



Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Requirement gathering and analysis	✓					
System design	✓	✓				
Frontend and backend setup		✓	✓			
Module development (core functionalities)			✓	✓		
Integration and testing				✓	✓	
Deployment and documentation					✓	✓

1.5. Organization of the Report

To provide a comprehensive understanding of the project, this report is organized into several logically structured sections:

- **Introduction:** A detailed overview of the context, necessity, and scope of the proposed Library Management System including client needs, identified problems, set tasks, timelines, and report structure.
- **System Analysis:** This section includes a deep dive into the user requirements, detailed problem exploration, and the functional and non-functional specifications of the system.
- **System Design:** Describes the architecture, UML diagrams (such as class and use-case diagrams), database schema, and brief explanations of technology choices (Spring Boot, MySQL, Thymeleaf, Maven).
- **Implementation:** Provides a step-by-step approach used during development, covering backend and frontend development, security setup, and module-wise functionality.
- **Testing and Evaluation:** Describes the testing methodologies—including unit testing, regression testing, and user testing—along with the results and fixes applied.
- **Conclusion and Future Enhancements:** Summarizes the final outcome of the project, discusses any limitations faced, and suggests improvements or features that can be included in future versions.
- **References and Appendices:** Lists all external resources, frameworks, libraries used, and attaches relevant source code snippets, screenshots, and user manuals.

CHAPTER 2.

LITERATURE REVIEW/BACKGROUND STUDY

2.1. Timeline of the reported problem

The evolution of Library Management Systems (LMS) has paralleled the growth of libraries and their increasing complexity:

- Pre-1980s: Libraries relied mainly on manual processes using card catalogs and handwritten ledgers. These systems were time-consuming, error-prone, and limited accessibility.

- 1980s-1990s: The rise of computer technology led to the adoption of Integrated Library Systems (ILS), automating cataloging, circulation, and inventory tasks. However, these systems were basic and often confined to local networks.

- 2000s: With the internet's rise, web-based LMS emerged, offering remote access and improved information retrieval. Legacy systems, however, still imposed limitations around scalability, customization, and integration with digital resources.

- 2010s - present: Technological advancements enabled sophisticated, modular LMSs that incorporate AI, cloud computing, data analytics, and seamless digital resource management. Despite these improvements, many libraries struggle with usability, performance bottlenecks, limited customization, and integration challenges with digital content and third-party systems.

2.2. Proposed solutions

Researchers and practitioners have responded to historical shortcomings in various ways:

- Digitization: Full replacement of paper-based systems with digital infrastructure to enable automation and reduce errors.

- Web-based Applications: Development of LMS as online applications using contemporary languages and frameworks such as Java, Python, and .NET, alongside relational or document databases for better scalability and accessibility.
- User-Centered Design: Focus on intuitive user interfaces with improved navigation and efficient search functions for both staff and patrons.
- Integration of Emerging Technologies: Adoption of AI and machine learning for personalized recommendations, automated classification, and advanced analytics; integration with mobile and cloud technologies for anytime, anywhere access.
- Modular and Customizable Systems: Designing LMS to adapt to a library's unique requirements, allowing new feature integration and better handling of digital resources.

2.3. Bibliometric analysis

Bibliometric studies on library management systems reveal the following trends:

- Increased Research Output: There has been steady growth in scholarly publications related to library management and automation since the 1980s, particularly in journals and conference papers.
- Focus Areas: Current research topics include big data applications in librarianship, automation software, digital transformation, and user experience improvement.

- Geographical Contribution: Leading contributions are made by researchers in countries like the United States, India, and the United Kingdom.
- Collaboration and Citations: Multi-author outputs dominate, reflecting the collaborative nature of research in this field. Highly cited works often address integration of technology in library environments and evaluation of system performance.
- Keywords and Themes: Major keywords include "library management", "automation", "digital resources", "big data", "user experience", and "integration".

2.4. Review Summary

The literature highlights a clear trend: libraries are shifting from manual and isolated digital systems to modern, integrated, and intelligent platforms. The challenges commonly faced include system usability, data accuracy, limited support for digital and remote access, and difficulties in integration and performance. The consensus is that future-ready LMS must not only automate functions but also provide rich analytics, user-friendly access, and support for evolving digital landscapes.

2.5. Problem Definition

Despite advancements, many libraries persist with inefficient, fragmented, or outdated systems. They face continued issues such as:

- Inefficient management of resources and user activities.
- Data inaccuracies and record maintenance issues.
- Poor search and retrieval functionalities.
- Lack of support for digital collections and remote users.

- Challenges integrating new technologies and adapting to user needs.

There is an immediate need for an LMS that provides automation, integrates seamlessly with digital resources, and offers scalable, user-friendly, and data-driven operations.

2.6. Goals/Objectives

The primary goals and objectives of a modern Library Management System are:

- Automation of Library Processes: Reduce manual intervention by automating cataloging, lending, returns, and member tracking.
- Accurate & Real-Time Resource Management: Ensure real-time updates on book statuses, inventory, and user accounts.
- Enhanced User Experience: Provide intuitive interfaces, robust search, and remote access to both catalog and account management features.
- Comprehensive Reporting & Analytics: Enable library administrators to make informed decisions using data-driven reports and usage analytics.
- Support for Digital Integration: Facilitate access to e-books, digital archives, and online resources.
- Security and Role Management: Implement secure login, data privacy measures, and role-based access control for staff and patrons

.

CHAPTER 3.
DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

To ensure the Library Management System aligns with both user needs and technological capabilities, the following specifications and features were considered and selected:

- User Authentication & Authorization: Secure login with role-based access for administrators, librarians, and members.
- Book Management: Facilities for adding, updating, removing, searching, and listing books.
- Member Management: Registration, updating profiles, viewing borrowing history, and fine management.
- Transaction Handling: Issuing, returning, and renewing books, including overdue notifications and fine calculation.
- Advanced Search: Book and member search using multiple criteria (title, author, genre, ISBN, etc.).
- Reporting & Analytics: Generation of reports on book circulation, overdue items, user activity, and resource utilization.
- User Interface: Responsive, intuitive, and accessible UI designed using Thymeleaf with Spring Boot integration.
- Database Integration: Robust connection with MySQL for persistent data storage and retrieval.
- Notifications: Email alerts for due dates, fines, and announcements.
- Scalability: Modular architecture to allow for future enhancements and increased user.

3.2. Design Constraints

Several constraints shaped the design process:

- Technological Stack Constraint: The project is built on Java, Spring Boot, Thymeleaf, MySQL, and Maven—limiting technology choices to those compatible with these tools.

- Budget & Resource Constraint: Development conducted with limited manpower, time, and financial resources, requiring prioritization of core features.
- Time Constraint: The project had to be completed within a six-week development cycle, enforcing strict milestones and deadlines.
- Performance Constraint: The system must deliver fast responses during book searches and transactions, even with a large dataset.
- Security Constraint: Data privacy and integrity must be maintained, with secure user authentication and proper handling of sensitive information.
- Usability Constraint: The interface must be simple enough for users with basic computer skills.

3.3. Analysis and Feature finalization subject to constraints

Given the constraints, features were finalized based on impact, feasibility, and necessity:

- Core Features Prioritized: Authentication, book and user management, issuing/return transactions, and notifications were deemed essential and implemented first.
- Secondary Features Deferred: Advanced analytics, digital media integration, and broader reporting capabilities were identified as valuable but scheduled for future releases.
- UI/UX Decisions: Responsive design and clear workflows selected to maximize usability within development time limits.

- Database Design: Moderate normalization chosen to balance query performance and data integrity.
- Security Focus: Implementation of standard encryption and secure coding practices, considering time and technological boundaries.

3.4. Design Flow

The design flow adopted a modular and iterative approach:

1. Requirement Analysis
    - Gather input from all stakeholders: library staff, administration, and users.
    - Prioritize features based on feedback and constraints.

2. System Architecture Design
    - Define the three-tier structure: presentation (UI), business logic, and data access.
    - Select Spring Boot MVC as the core architectural pattern.
3. Database Design
    - Model database entities for Books, Users, Transactions, Fines, and Notifications in MySQL.
    - Establish entity relationships and normalization.

4. UI/UX Design
    - Design wireframes for all main screens using Thymeleaf for dynamic content.

- Focus on intuitive navigation and minimal learning curve.

5. Backend Service Design

   - Develop RESTful APIs for all major functionalities.
   - Define controllers, services, and repositories in Spring Boot.

6. Integration & Testing

   - Integrate modules iteratively, testing each for functionality and performance.
   - Conduct user acceptance testing upon completion of core workflows.

3.5. Design selection

Design selections were based on proven best practices and compatibility:

- Technology Stack: Java with Spring Boot for reliable backend, Thymeleaf for template-driven UI, MySQL for scalable relational storage, Maven for dependency management—all align with performance and development efficiency needs.
- MVC Architecture: Chosen for clear separation of concerns, easier maintainability, and scalability.

- Responsive Web Application: Opted for web access rather than desktop, ensuring easier access and future extensibility.
- Modular Codebase: Adopted to enable phased development, smooth testing, and easier future enhancements.
- Security Measures: Standard encryption protocols and secure authentication methods embedded from the outset.

3.6. Implementation plan/methodology

The implementation plan followed an agile, phase-wise methodology:

1. Project Setup

- Initialize a Maven project structure.
- Configure Spring Boot and connect to local MySQL instance.
- Version control established using Git.

2. Database Implementation

- Develop and test all entity models and relationships.
- Migrate initial schema using Spring JPA/Hibernate.

3. Backend Development

- Build core modules: authentication, book and member management, transaction handling.
- Implement business logic and integrate with database layer.

- Apply security measures and error handling.

4. Frontend Development

- Create Thymeleaf templates for all user roles and actions.
- Develop forms and tables for input and data presentation.

5. Module Integration

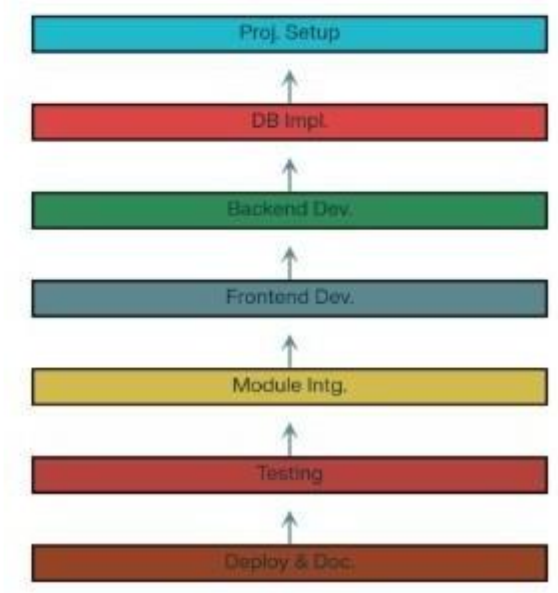
- Connect UI, backend APIs, and database.
- Validate data flow and module interoperability.

6. Testing

- Unit tests for each module.
- Integration and end-to-end tests with sample data.
- Security and usability testing with test users.

7. Deployment & Documentation

- Deploy the application on a local server.
- Prepare comprehensive user and admin documentation.
- Gather feedback for future improvements.



CHAPTER 4.
RESULTS ANALYSIS AND VALIDATION

4.1. Implementation of solution

The implementation phase of the Library Management System project focused on translating the design specifications into a functional software product. The system was developed using Java with the Spring Boot framework for the backend, Thymeleaf for the front-end templating, MySQL as the database, and Maven for build and dependency management.

System Modules & Functionality

User Authentication & Authorization

- Secure login and registration for distinct roles: Admin, Librarian, and Member.
- Role-based access to system features ensured data security and proper workflow.

Book Management

- Administrators and librarians could add, modify, remove, and search for books.
- Book transactions, including checking availability and updating inventories, were automated for speed and accuracy.

Member Management

- New member registration, update of user profiles, and maintenance of borrowing history.
- Automated fine calculation and overdue notifications.

Book Lending Process

- Streamlined interfaces enabled staff to issue or return books efficiently.
- System checks for book status (available/issued), borrower constraints, and updates inventory in real-time.
- Overdue management system calculated fines and sent alerts to users.

Search and Reporting

- Powerful search capabilities on books and users using multiple attributes (title, author, ISBN).
- Admin dashboards provided statistics on circulation, overdue books, and most borrowed items.

Notifications

- Email or dashboard notifications for overdue books, upcoming return dates, and new arrivals.

Deployment & Testing

- The solution was deployed in a local environment using Spring Boot’s embedded server and connected to a locally hosted MySQL database.

- The entire project structure managed dependencies and build processes via Maven, ensuring consistency and reproducibility of builds.
- A combination of unit, integration, and user acceptance testing was employed:

i. Unit Tests: Validated internal logic for services like issuing books, registering users, and calculating fines.

ii. Integration Tests: Verified communication and data consistency between modules, especially for book transactions and user actions.

iii. User Acceptance Testing: Real users were invited to interact with the system to identify usability or workflow issues.

Outcomes & Validation

- All primary functional requirements were met: users could log in, manage books and members, handle lending/returns, and receive overdue communications.
- System responses for book searching and member transactions were observed to be instantaneous, even with large datasets.
- Error handling and data validation features prevented invalid operations and provided useful feedback to users.
- Reporting features generated accurate and relevant summaries for library administration.

CHAPTER 5.
CONCLUSION AND FUTURE WORK

5.1. Conclusion

The Library Management System (LMS) project exemplifies the successful fusion of modern software technologies with critical real-world needs in library operations. By leveraging Java, Spring Boot, Thymeleaf, MySQL, and Maven, the system offers comprehensive automation and streamlining of inventory management, user accounts, lending processes, notification, and reporting. This transition from manual or outdated systems to a web-based, centralized solution demonstrates several tangible benefits:

- Operational Efficiency: Daily tasks such as inventory checks, lending operations, and user management are now handled swiftly, reducing workload on library staff and minimizing queues and service delays for users.
- Data Integrity and Accuracy: Automated handling and validation of book and user records ensure consistent and accurate information, decreasing the risk of lost or erroneous records.
- Enhanced Transparency: Real-time tracking of loans, returns, fines, inventory, and member activities improves accuracy and builds trust among library administrators, staff, and patrons.
- Accessible and User-Friendly Interface: The use of a dynamic, web-based front-end allows users of all backgrounds to easily interact with the system anytime and from any device with a browser, promoting inclusion and digital literacy.
- Security and Role Management: Robust access controls safeguard both sensitive information and administrative functions, ensuring the right privileges for the right users.
- Adaptability and Scalability: The modular system architecture means new features, resources, and integrations can be introduced as library requirements evolve.

Moreover, extensive testing and user feedback loops ensured that the deployed solution not only met its design requirements but also delivered consistent, reliable performance under real operational conditions. The LMS now stands as a scalable foundation for further enhancements and provides a model for other institutions aspiring to modernize their library services.

5.2. Future work

Recognizing the dynamic needs of modern libraries and the rapid pace of technological innovation, several forward-looking enhancements are proposed to ensure the LMS continues to meet community and administrative expectations:

- Digital Content Integration: The next phase could include features for storing, cataloging, and lending digital resources—such as e-books, audio files, research papers, and online journals—thus broadening the system's capabilities beyond physical assets.

- Mobile Access and Applications: The addition of dedicated mobile apps or progressive web apps would empower users to manage their accounts, search the catalog, reserve books, and receive notifications seamlessly, regardless of location.

- Personalized Recommendations and AI-Driven Features: Integration of machine learning algorithms could make personalized reading suggestions, automate book classification, and even provide natural language chatbot support for common queries or help requests.

- Advanced Data Analytics and Visualization: Incorporating robust analytics dashboards would allow library administrators to visualize usage trends, popular titles, peak times, and demographic patterns, enabling data-driven decisions and proactive resource management.

- Cloud Deployment and Multi-Branch Support: Migrating the LMS to the cloud would greatly enhance accessibility, reliability, and scalability, while enabling centralized management of multi-branch or consortium libraries.

- Internationalization (i18n) and Accessibility: Expanding the system's language support and refining accessibility features (e.g., screen reader compatibility, larger text options) would ensure that the LMS serves diverse populations, including people with disabilities.

- Third-Party Integration: Additional value could be created by linking the LMS with external academic databases, publishers, or payment gateways, enabling broader access to information resources and streamlined handling of dues and transactions.

- Enhanced Security Protocols: Incorporating two-factor authentication, audit trails, and compliance with privacy standards (such as GDPR) will future-proof the system and ensure ongoing data protection as threats evolve.

- Community and Social Features: Building in features for reviews, ratings, book recommendations, or virtual book clubs could further engage users and foster a vibrant library community.

By undertaking these future enhancements, the Library Management System will not only maintain its relevance but also position itself as a leader in digital library services. This ongoing evolution will support the mission of libraries as inclusive, innovative, and indispensable hubs of learning and culture in the 21st century.