

UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University



LIBRARY MANAGEMENT SYSTEM

PROJECT REPORT

Submitted by:

PIYUSH SAURABH
UID: 24MCA20002 UID:24MCA20018

In partial fulfilment for the award of the degree of

MASTERS OF COMPUTER APPLICATIONS

IN

UNIVERSITY INSTITUTE OF COMPUTING (UIC)



CHANDIGARH UNIVERSITY, GHARUAN, MOHALI-140413,
PUNJAB

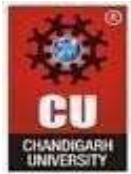


TABLE OF CONTENTS

1.	Abstract
2.	Introduction
3.	Objectives
4.	System Requirements
5.	System Design/Flow -Flow Diagram -Front-End Design -Database Design
6.	Implementation
7.	Screenshots
8.	Conclusion



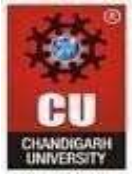
1. ABSTRACT

The Library Management System is a full-stack web application designed to streamline the management of book catalogs for libraries, educational institutions, and individuals. This system provides users with a comprehensive interface to add, view, and delete books, effectively simplifying catalog organization. The application leverages front-end technologies such as HTML, CSS, and JavaScript, combined with Bootstrap for a responsive and visually appealing interface that adjusts seamlessly across devices. JavaScript, along with jQuery, enhances user interaction through real-time updates and validations, providing a smooth and engaging experience.

On the server side, Node.js is utilized as a lightweight and scalable backend platform, efficiently handling data processing and API requests. This framework allows for asynchronous, event-driven architecture, ensuring that user requests are managed promptly without any delays or blocking. CRUD (Create, Read, Update, Delete) functionalities are implemented through RESTful API endpoints, allowing users to perform essential operations on book records. By sending and receiving JSON data between the frontend and backend, the system achieves dynamic interactions that make the catalog continuously up-to-date for the user.

Data persistence is managed through MySQL, a relational database known for its reliability, security, and ease of use in managing structured data. MySQL stores all book-related information, including the title, author, genre, and optional cover image for each book. This database solution not only supports efficient storage and retrieval of data but also allows for scalability, making it feasible to handle large catalogs with potentially thousands of book entries.

This Library Management System serves as a valuable digital tool for users seeking an organized way to handle book information. The integration of HTML, CSS, JavaScript, Node.js, and MySQL results in a cohesive system that maintains high performance and reliability while being easy to deploy. Future iterations of the application could include additional features like search and filtering options, user accounts, and data export functionalities, further enhancing the usability and practicality of the system in larger library or institutional settings. Overall, this project exemplifies a modern approach to full-stack development and provides a foundation for more advanced digital library management solutions.



2. INTRODUCTION

Library Management Systems play a vital role in organizing and cataloging book collections, making it easier to manage inventory and ensure that resources are available and accessible. With the increasing volume of resources in libraries, schools, and personal collections, a digital solution becomes essential. Traditionally, manual cataloging methods require significant time and effort, often resulting in inconsistencies and inefficiencies. A digital Library Management System addresses these issues by automating cataloging processes and providing tools for streamlined inventory management.

This project offers a modern, digital solution to manage book information effectively. Users can perform essential tasks, including adding, viewing, and deleting books, through an intuitive, user-friendly interface. The frontend is built with HTML and CSS to create a clean, visually appealing layout, while JavaScript enhances interactivity and responsiveness. The integration of jQuery supports dynamic, real-time interactions, allowing users to see updates to the book catalog instantly. The backend of this system, developed in Node.js, handles all requests and operations with efficiency and scalability. Node.js is particularly well-suited for this project as it allows for an asynchronous, non-blocking architecture, meaning user requests are managed without delays. This asynchronous capability enables the system to handle multiple users or requests simultaneously, making it suitable for library environments where multiple operations may occur concurrently.

For data management, the project uses MySQL, a relational database known for its reliability, security, and capacity to handle large datasets. MySQL enables structured data storage for each book entry, including details like title, author, genre, and an optional cover image. The use of MySQL ensures that the data remains secure, easy to manage, and readily accessible, while also supporting complex queries for potential future enhancements, like search and filtering functions. This system is more than just a storage tool; it also provides an organized, centralized catalog for library resources, simplifying workflows and reducing dependency on traditional cataloging methods. As a full-stack solution, the Library Management System demonstrates the effective integration of frontend and backend technologies, offering a cohesive user experience. Future improvements could include user accounts, personalized book lists, and search functionalities, which would allow for even more effective resource management. By implementing a reliable backend with Node.js and secure data storage with MySQL, this project creates a strong foundation for an efficient, scalable, and user-focused digital library system.

3. OBJECTIVES

- ❑ **Create a User-Friendly Interface:** Develop an easy-to-navigate interface for managing books.
- ❑ **Implement CRUD Operations:** Enable users to add, view, update, and delete book records.
- ❑ **Allow Image Uploads:** Enable users to upload and preview book cover images when adding books.
- ❑ **Ensure Secure Data Storage:** Use MySQL for reliable and secure storage of book information.
- ❑ **Handle Server Logic:** Use Node.js to manage server-side operations and respond to user requests efficiently.
- ❑ **Update Book List Dynamically:** Automatically refresh the book list on the frontend when books are added or removed.
- ❑ **Provide Search and Filter Options:** Allow users to search for books by title, author, or genre.
- ❑ **Ensure Responsive Design:** Make the application compatible with various devices like desktops and smartphones.
- ❑ **Provide User Feedback:** Display messages to inform users about the success or failure of their actions.
- ❑ **Plan for Future Enhancements:** Design the system to be easily expandable for future features like user accounts and reports.

4. SYSTEM REQUIREMENTS

Software Requirements

Frontend

- **HTML:** Used to create the structure of the web application, allowing for proper layout and content organization.
- **CSS:** Styles the application to make it visually appealing, customizing elements like colors and fonts.
- **JavaScript:** Adds interactivity to the application, enabling features like dynamic content updates and form validation.

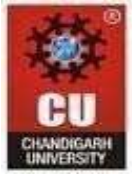
Backend

- **Node.js:** A JavaScript runtime that manages the server-side logic, enabling efficient handling of user requests and data processing.
- **MySQL Database:** A relational database used for securely storing book information, allowing quick data retrieval and management.

Libraries

- **jQuery:** A JavaScript library that simplifies HTML manipulation and AJAX operations, enhancing user experience by allowing smooth interactions.
- **Bootstrap:** A front-end framework that provides pre-designed components, making it easier to create a responsive layout that works well on different devices.

Browser Compatibility



- The application should work on modern web browsers like Google Chrome, Firefox, Edge, and Safari, ensuring accessibility for a wide range of users.

Hardware Requirements

Server or Local Environment

- **Node.js and MySQL Support:** The hardware must be capable of running Node.js and MySQL, whether on a local machine or a dedicated server.

Minimum Specifications

- **RAM:** At least 2 GB of RAM is recommended to ensure the application runs smoothly and can handle multiple users.
- **Storage:** A minimum of 500 MB of storage is needed for application files and database entries, allowing for future growth as more books are added.

Additional Considerations

- **Network Connection:** A reliable internet connection is important for both local development and deployment, enabling quick access to the application.
- **Backup Solutions:** Regular backups are recommended to prevent data loss and ensure easy recovery in case of issues

5. SYSTEM DESIGN/FLOW

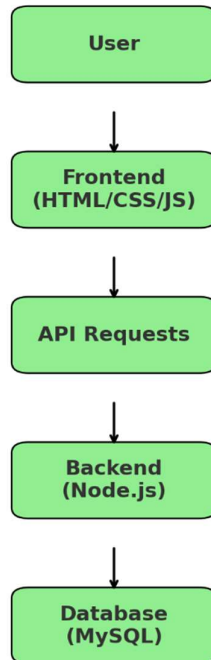
The Library Management System is thoughtfully designed to streamline the management of books, making it user-friendly and efficient. The system is structured into three main components: the frontend, the backend, and the database, each playing a critical role in ensuring the smooth operation of the application.

The system employs a client-server model, where the frontend is the visible part that users interact with, developed using HTML, CSS, and JavaScript. This creates a responsive and dynamic user interface, featuring an intuitive layout that allows users to easily navigate the system, view the list of books, and access the form for adding new books. The frontend utilizes AJAX functionality with jQuery to make asynchronous requests to the backend, enabling real-time updates without the need for page reloads. This enhances the user experience by providing immediate feedback on actions such as adding or deleting books.

On the other hand, the backend is implemented using Node.js and handles the application logic and processes user requests. Acting as an intermediary between the frontend and the database, the backend provides a set of RESTful API endpoints that allow the frontend to perform CRUD operations (Create, Read, Update, Delete) on book records. This separation of concerns enables clear communication and interaction between the two layers while ensuring that the backend processes data efficiently and sends appropriate responses back to the frontend.

FLOW DIAGRAM

Library Management System Flow Diagram



FRONT-END DESIGN

The front-end consists of:

1. **Book List Section:** Displays all books with titles, authors, genres, and cover images.
2. **Add Book Form:** A form where users can input book details and upload a cover image, with an option for preview.
3. **Message Display Area:** Displays success or error messages to provide user feedback.
4. **Responsive Design:** Bootstrap ensures compatibility across device screen

DATABASE DESIGN

The database design for the Library Management System is structured to efficiently manage and store information about books. It ensures data integrity and supports the application's requirements for adding, viewing, and managing book records.

SQL Commands

1. **Create Database:** This command checks for the existence of the library database and drops it if it exists. This allows for a fresh start when setting up the database.

```
DROP DATABASE IF EXISTS library;
```

```
CREATE DATABASE library;
```

2. **Use the Library Database:** This command sets the context to the newly created library database, allowing subsequent commands to operate within this database.

```
USE library;
```

3. **Create Books Table:** The books table is created to store information about each book in the library. The table includes the following fields:

- **id:** A unique identifier for each book, which automatically increments with each new entry.
- **title:** The title of the book, which is a required field.
- **author:** The author of the book, also a required field.
- **genre:** An optional field for the genre of the book.
- **image_path:** A required field that stores the path or URL to the book's cover image.
- **created_at:** A timestamp that records when the book entry was created, defaulting to the current time.

```
CREATE TABLE IF NOT EXISTS books (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(255) NOT NULL,  
    author VARCHAR(255) NOT NULL,  
    genre VARCHAR(100),  
    image_path VARCHAR(255) NOT NULL,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

4.Select All Books: This command retrieves all entries from the books table, allowing users to view the current collection of books in the library.

```
SELECT * FROM books;
```

6. IMPLEMENTATION

Backend (Node.js and MySQL)

The backend of the Library Management System is built using Node.js, which manages the server-side logic. MySQL is used as the database for storing book records. The backend offers an API with the following endpoints:

- **GET /books:** Retrieves all book records from the MySQL database and sends them as a JSON response to the frontend.
- **POST /books:** Accepts book details and an optional image file from the client, storing this information in the database along with the image path.
- **DELETE /books/**
: Deletes a specific book by its ID from the MySQL database when requested by the client.

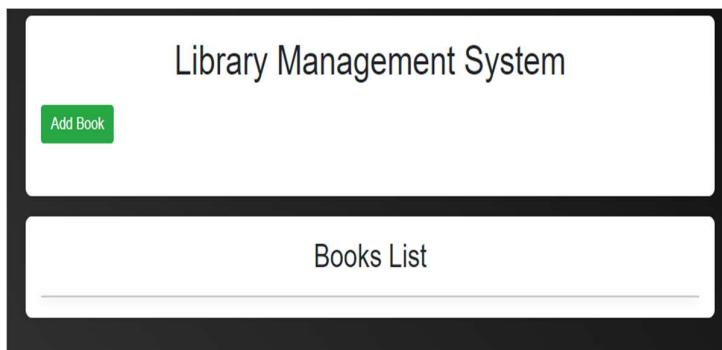
Frontend (HTML, CSS, JavaScript)

The frontend uses HTML, CSS, and JavaScript to create a responsive and interactive user interface. jQuery is employed for AJAX calls to communicate with the backend API, enabling real-time updates without refreshing the page:

- **Adding a Book:** A form collects book information and an optional image. Upon submission, JavaScript validates the input and sends the data to the server using AJAX.
 - **Displaying Books:** The `fetchBooks()` function retrieves all books from the server and displays them dynamically as cards on the webpage.
 - **Deleting a Book:** Each book card has a delete button that triggers a request to the backend to remove the corresponding book from the database.
-

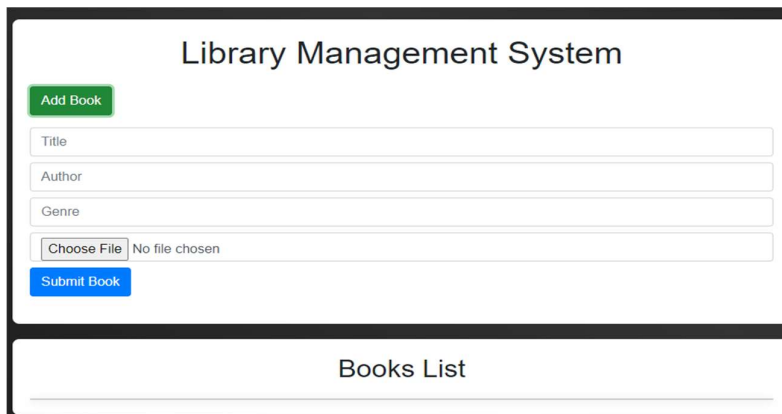
7. SCREENSHOTS OF PROJECT

1.View the Library Management System: The page displays the title, an "Add Book" button, and an empty "Books List."



The screenshot shows a web interface with a black border. At the top, the text "Library Management System" is centered. Below it, on the left, is a green button labeled "Add Book". At the bottom, there is a section titled "Books List" with a horizontal line underneath it, indicating an empty list.

2.Click "Add Book": When you click the "Add Book" button, a form appears to enter details.



The screenshot shows the same web interface as before, but with a form visible. The "Add Book" button is still present. Below it, there are four input fields: "Title", "Author", "Genre", and a file upload section with a "Choose File" button and the text "No file chosen". At the bottom of the form is a blue button labeled "Submit Book". The "Books List" section at the bottom remains empty.

3.Fill in the Book Details: Enter the title, author, genre, and optionally upload a file.

Library Management System

Add Book


Behold the Man

Michael moor

Literature

Choose File

bookcase-books-bookshelves-159711 - Copy.jpg



Submit Book

4.Submit the Book: Click "Submit Book" to add the book to the list.

5.View the Updated Books List: The new book will appear in the "Books List" section.

Library Management System

Add Book


Behold the Man

Michael moor

Literature

Choose File

bookcase-books-bookshelves-159711 - Copy.jpg



bookcase-books-bookshelves-159711 - Copy.jpg


Submit Book

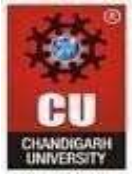
Books List

Behold the Man

by Michael moor Genre: Literature

Delete





8. CONCLUSION

The Library Management System developed in this project demonstrates a robust and efficient approach to managing book inventories. By utilizing a combination of HTML, CSS, JavaScript, Node.js, and MySQL, the system provides a user-friendly interface for seamless interaction with the database. Users can easily add, view, and delete books, ensuring a streamlined process for library management.

The integration of MySQL as the database backend guarantees secure and reliable data storage, while Node.js handles server-side logic, facilitating smooth communication between the frontend and the database. The application not only meets the core requirements of a library system but also enhances user experience through features such as image uploads and dynamic content updates.

This project has highlighted the importance of effective data management solutions in modern libraries. By adopting a digital approach, libraries can improve accessibility, streamline operations, and foster a more engaging environment for readers and researchers alike. Future enhancements could include user authentication, advanced search functionalities, and integration with external APIs to enrich the library's offerings further.