PROJECT REPORT

On

LIBRARY MANAGEMENT SYSTEM

Submitted in partial fulfilment of the requirement for the

Course BEE (22CS026) of

COMPUTER SCIENCE AND ENGINEERING

B.E. Batch-2022

in

June -2024

Supervised By:

Mr. Venkatesh Moyya

Chitkara University

Submitted By
Name: Sai madhav bhalla
Roll No: 2210990760
Name: Saksham dogra

Roll no.:2210990760

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHITKARA UNIVERSITY

PUNJAB

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as the part of the curriculum.

We are thankful to "Mr. Venkatesh Moyya" for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation to "Dr. Parneet Kaur" who provided her valuable suggestions and precious time in accomplishing our integrated project report.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we Shared our day-to day experience and received lots of suggestions that improve our quality of work.

Name:Sai Madhav Bhalla

Roll No.:2210990760

INDEX

1. ABSTRACT

The Library Management System is a comprehensive web-based application designed to automate and streamline the core operations of a library. Built using modern technologies, this system offers a user-friendly interface for managing library resources and student records efficiently. It provides a secure admin login that grants access to a centralized dashboard, where administrators can assign books to students, track borrowed books, and manage book returns.

At its core, the system leverages MongoDB as the backend database to ensure reliable and scalable storage of student and book information. The admin dashboard simplifies key library operations: books can be easily assigned to students, with a real-time summary of their borrowing history displayed. Additionally, the system maintains a Borrowed Books List that details the issue and return dates for each borrowed book, ensuring transparency in book circulation. The Return Books feature allows for easy tracking and processing of returns, with a search function that retrieves a student's borrowing history based on their ID, offering both single and bulk return options.

In addition to book assignment and return functionalities, the system includes modules for Student Management and Book Management. Admins can add new students and books, with dynamic tables displaying all relevant data for easy reference. This ensures that both the student roster and the library's book inventory remain up to date.

By using this system, libraries can significantly reduce the time and effort spent on administrative tasks. Manual tracking of books and student records is replaced by an automated process, minimizing errors and improving efficiency. The system's logout functionality also ensures secure access, preventing unauthorized usage.

In conclusion, this Library Management System enhances the way libraries function, providing a practical solution for book assignment, borrowing, and return processes. The system's design ensures scalability, allowing libraries to grow their collections and user base without compromising performance. Through its implementation, library staff can focus more on providing quality service to users, while the system handles the complex backend operations.

2. INTRODUCTION

2.1 Background

Libraries have long been the cornerstone of knowledge, providing access to books, research materials, and other resources. Traditionally, libraries have relied on manual processes to manage their inventory, record student and user information, and track book loans. Librarians would physically manage records, leading to labor-intensive work and frequent errors in maintaining the accuracy of student records, borrowed books, and due dates.

The Library Management System is a comprehensive software application designed to modernize the operations of a library. It enables librarians or administrators to efficiently manage books, student records, and the overall borrowing process. Built using MongoDB as the database backend, this system ensures robust data storage and fast retrieval of information, while the frontend provides an intuitive user interface for ease of use.

Using MongoDB for data storage and a web-based interface for ease of use, this **Library Management System** enables administrators to handle core library functions such as book assignments, tracking borrowed books, and managing returns with just a few clicks. The system ensures that librarians and administrators can focus on service quality while the software handles data storage and retrieval, simplifying the complexities of library management.

2.2 Problem Statement

Managing library operations manually presents significant challenges in terms of efficiency, accuracy, and scalability. Libraries that rely on traditional, paper-based record-keeping systems often encounter issues such as difficulty tracking borrowed books, student details, and return dates. This manual process is time-consuming, error-prone, and inefficient, especially as the number of students and books grows. Moreover, there is a high risk of data loss or misplacement, making it hard to maintain accurate records. Librarians face difficulties in retrieving information about which books are borrowed, by whom, and when they are due for return, resulting in delays and mismanagement. Additionally, managing an up-to-date inventory of books and student records becomes cumbersome without a digital solution, hindering the library's ability to operate smoothly.

To address these issues, there is a need for a Library Management System that can automate book assignment, track borrowing activity, and streamline the return process. Such a system would improve the efficiency of daily operations, provide real-time access to accurate data, and eliminate the risks associated with manual record-keeping.

The solution must also be scalable to handle growing collections and student populations while ensuring data security and ease of access for administrators.

3. SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

3.1 Methods

- **Requirement Gathering**:

Understanding the key functionalities such as student management, book inventory management, book assignment, return, and login authentication.

- **System Design**:

Designing the database structure in MongoDB, including collections for students, books, and borrowed books.

Designing the user interface using HTML, CSS, and JavaScript, with EJS used for templating the dynamic parts of the application.

- **Backend Development**:

Setting up the Nod e.js server using Express.js for routing and handling requests.

Creating RESTful APIs to interact with the MongoDB database for CRUD (Create, Read, Update, Delete) operations.

- **Frontend Development**:

Creating the HTML views for the admin login, dashboard, student management, book assignment, borrowed book list and returning list using EJS.

- **Authentication and Security**:

Implementing secure admin login using session management.

- **Testing and Debugging**:

Running the system in a local environment to ensure that all components (login, book assignment, borrowed list, return books, etc.) work seamlessly.

Unit testing for each feature and integration testing for overall system functionality.

3.2 Programming/Working Environment

The **Library Management System** is developed using the following tools and technologies:

- **Programming Languages**: JavaScript (Node.js for backend, HTML/CSS/JavaScript for frontend)
- **Frameworks**: Express.js for the backend, and EJS for frontend rendering.
- **Database**: MongoDB, using the Mongoose ODM for easy interaction with the database.
- **Development Environment**:
  - o Node.js runtime to execute backend code.

- o MongoDB installed locally or on a cloud server (such as MongoDB Atlas).
- o A text editor or IDE (such as Visual Studio Code or Notepad++) for coding.
- o Git (optional) for version control and collaboration.

The project environment consists of setting up a Node.js development server to handle backend logic and a MongoDB instance for database storage.

3.3 Requirements to run the application

- **Software Installations:**

Node.js: Install the latest version from nodejs.org.

MongoDB: Either install MongoDB locally or use MongoDB Atlas for cloud-based database management.

Install necessary dependencies using npm:

npm install express mongoose body-parser express-session ejs

Setting up the Database:

Create a MongoDB database with collections for library, admin, students, books,

borrowed books and returning.books.

Configure MongoDB connection strings in the Node.js backend code.

Running the Application:

Start the MongoDB service (if running locally).

Launch the Node.js server by navigating to the project folder and running:

node app.js

Open a browser and go to http://localhost:3000 (or the assigned port) to access the application.

4. DATABASE ANALYSING, DESIGN AND IMPLEMENTATION

4.1 Database Analysing:

The Library Management System requires a well-structured database to efficiently store and manage data related to students, books, borrowing activities, and administrative operations. MongoDB is chosen for this project due to its flexibility and scalability in handling large datasets, which are typical in library systems. The database needs to support CRUD (Create, Read, Update, Delete) operations for the following key entities:

- **Admin:** The administrator who logs in to manage the library.
- **Students:** Users who borrow books.
- **Books:** The collection of available books.
- **Borrowed Books:** A record of books assigned to students, along with issue and return details.
- **Returned Books:** A record of returned books and fine if any.

4.2 Database Design:

The database design involves creating collections (similar to tables in relational databases) for each entity. MongoDB being a NoSQL database, stores data in a flexible, document-based format (JSON), allowing for easy schema changes and data handling.

1. Admin Collection

This collection will store the admin login credentials. Since this is sensitive information, the password will be stored in a hashed format.

Field	Type	Description
_id	ObjectId	Auto-generated unique admin ID
username	String	Admin username
password	String	Hashed password for admin login

2. Students Collection

This collection stores details of each student in the library system. Each student can borrow multiple books.

Field	Type	Description
_id	ObjectId	Auto-generated unique student ID
student_id	String	Custom student ID roll no. (e.g., "S12345")
name	String	Student's full name
email	String	Student's email address
phone_no	String	Student's phone number
department	String	Department name (e.g., "Computer Science")

session	String	Student session or batch (e.g., "2021-2024")
student_status	String	Status of student (Active/Inactive)
registeredDate	Date	Date when the student was added

3. Books Collection

This collection stores the library's inventory of books. Each book is identified by a unique book_id.

Field	**Type**	**Description**
_id	ObjectId	Auto-generated unique book ID
book_id	String	Unique book identifier (e.g., "B1001")
title	String	Book title
author	String	Author's name
isbn	String	International Standard Book Number
language	String	Language of the book
cat_id	ObjectId	Reference to the category collection
total_copies	Number	Total number of copies
available_copies	Number	Number of copies currently available
book_status	String	Book status (Available/Unavailable)

4. BorrowedBooks Collection

This collection tracks the books borrowed by students. Each record links a student with a borrowed book and includes issue and due dates.

Field	**Type**	**Description**
_id	ObjectId	Auto-generated unique borrow ID
borrow_id	String	Unique borrow record ID
student_id	ObjectId	Reference to the student
book_id	ObjectId	Reference to the book
issue_date	Date	Date the book was issued
due_date	Date	Date by which the book must be returned

5. ReturnBooks Collection

This collection keeps a record of the books returned by students. It tracks return dates and any fines incurred.

Field	Type	Description
_id	ObjectId	Auto-generated unique return ID
return_id	String	Unique return record ID
borrow_id	ObjectId	Reference to the borrowed record
return_date	Date	Date when the book was returned
fine	Number	Fine for late return (if any)

6. Categories Collection

This collection categorizes the books in the library. Each category is identified by a cat_id.

Field	Type	Description
_id	ObjectId	Auto-generated unique category ID
cat_id	String	Unique category identifier
name	String	Category name (e.g., "Fiction")

4.3 Database relationship:

Students and **Books** have a **one-to-many** relationship through the **BorrowedBooks** collection, as a student can borrow multiple books.

BorrowedBooks collection serves as a join table, linking a student to one or more books, along with the dates they were borrowed and returned.

The relationships in MongoDB are established using references (ObjectId), where the BorrowedBooks collection contains references to the Students and Books collections. This approach allows for efficient querying and management of book borrowing activities.

4.4 Database implementation:

- **Install MongoDB:**
  Install MongoDB locally or set up a cloud-based MongoDB instance (such as MongoDB Atlas).

- **Create the Collections:**
  Define collections for Admin, Students, Books, and BorrowedBooks.
- **Insert Sample Data:**
  Insert sample data into the collections for testing.
- **Implementing CRUD Operations:**
  The backend Node.js server uses Mongoose to handle interactions with the MongoDB database. This provides an easy way to perform CRUD operations on the collections.
  Create (Add new student/book, record borrowing details).
  Read (Fetch details of students, books, and borrowed books).
  Update (Modify student/book details, update borrow/return status).
  Delete (Remove student/book records if necessary)

5. PROGRAM'S STRUCTURE ANALYSING AND GUI CONSTRUCTING

5.1 Program Structure Analysis:

1. Main Program (index.js)
- Sets up the server, defines routes, and configures middleware.
- Listens on port 8080 to serve the application.
- Core libraries: express, body-parser, express-session, mongodb, path, etc.

2. Routing Files (/routes)
- Each functional module (students, books, borrowing) has dedicated routing logic.
- Example routes:
  - GET /students - Retrieve student list and render to EJS.
  - POST /books/add - Add a new book to the database.
  - POST /borrow/add - Record a borrowed book.

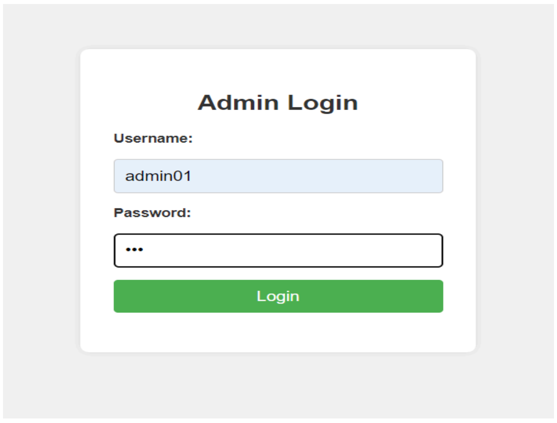
3. Views (/views)
- Contains EJS templates that render the user interface.
  - login.ejs: Admin login page.
  - dashboard.ejs: Displays main menu after login.
  - students.ejs, books.ejs: Pages to view and manage students and books.
  - borrow.ejs: Borrowing form and list.
  - return.ejs: Return book form and list.

4. Public Folder (/public)
- Holds static assets (CSS, images, JavaScript files).
- Provides styling for the front-end (e.g., layout for the dashboard, form styling, table views).

5. Database Configuration (database.js)
- Contains MongoDB connection setup, and allows access to various collections.

5.2 GUI (Graphical User Interface) Construction:

- **Admin Login Page**


Admin Login

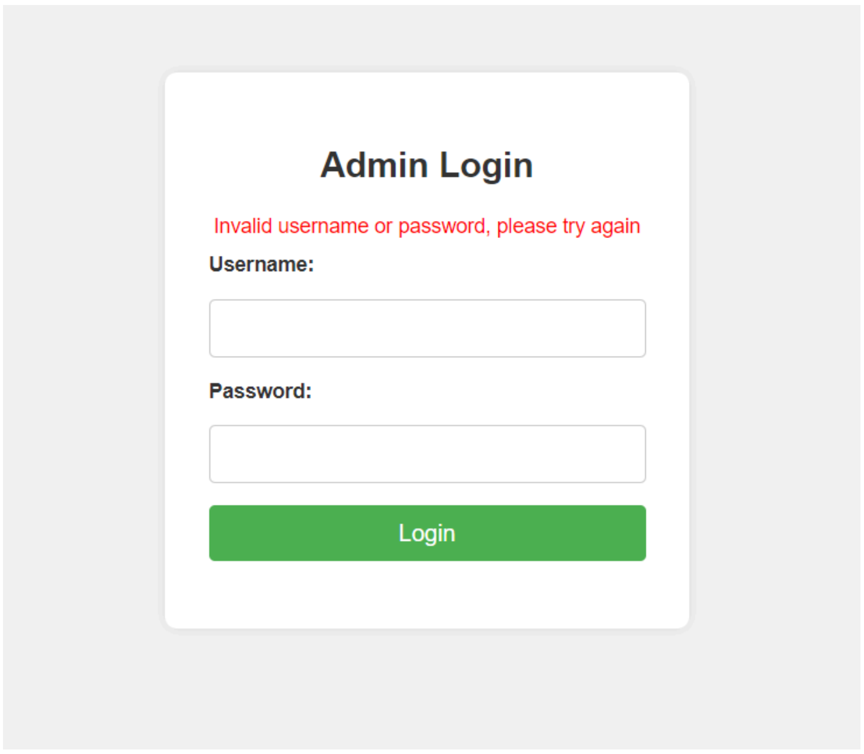
Username:

admin01

Password:

•••

- **If invalid username or password**

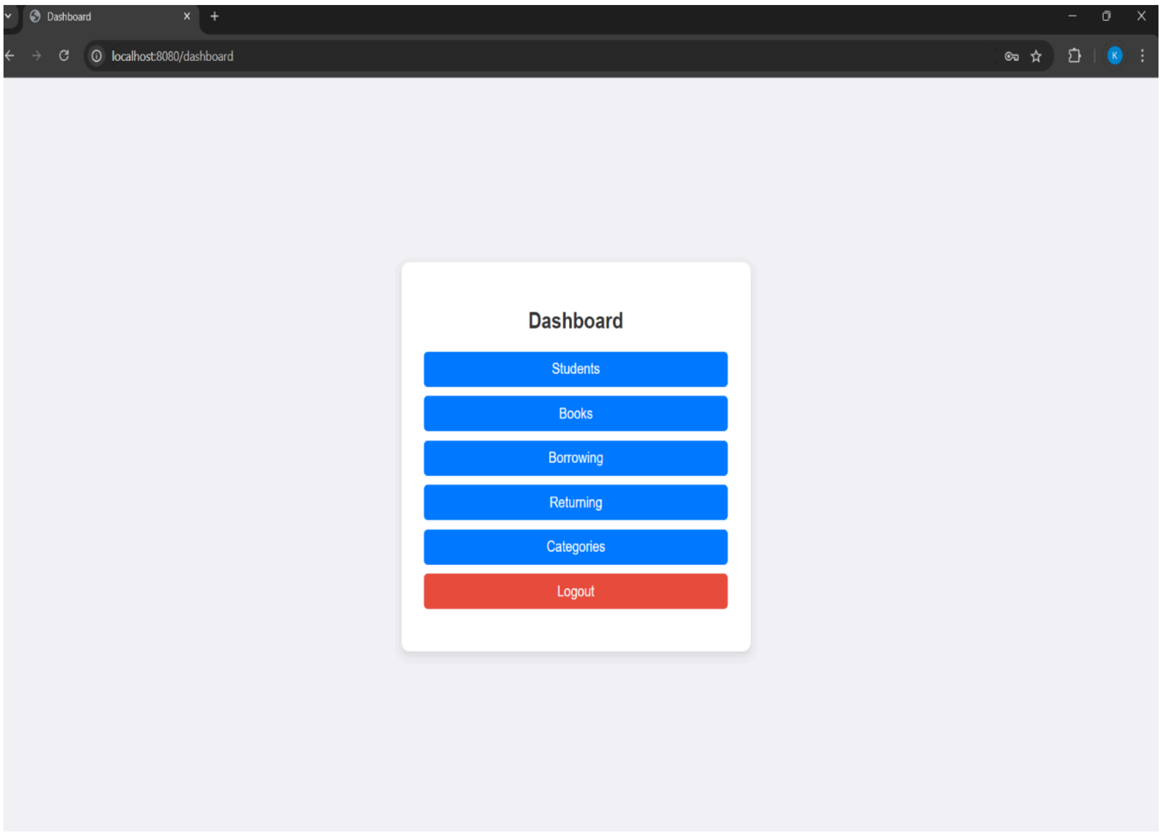

Admin Login

Invalid username or password, please try again

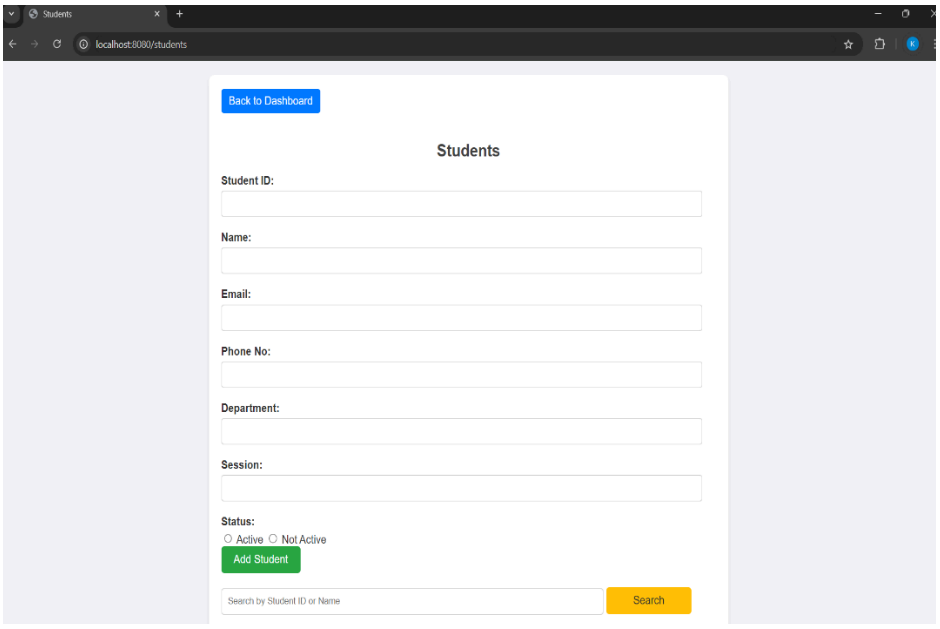
Username:

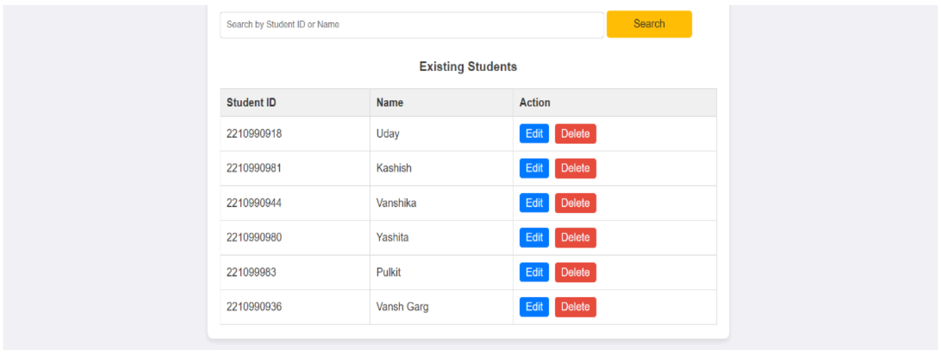
Password:

- **Dashboard**



- **Add Student and Existing Student Table**



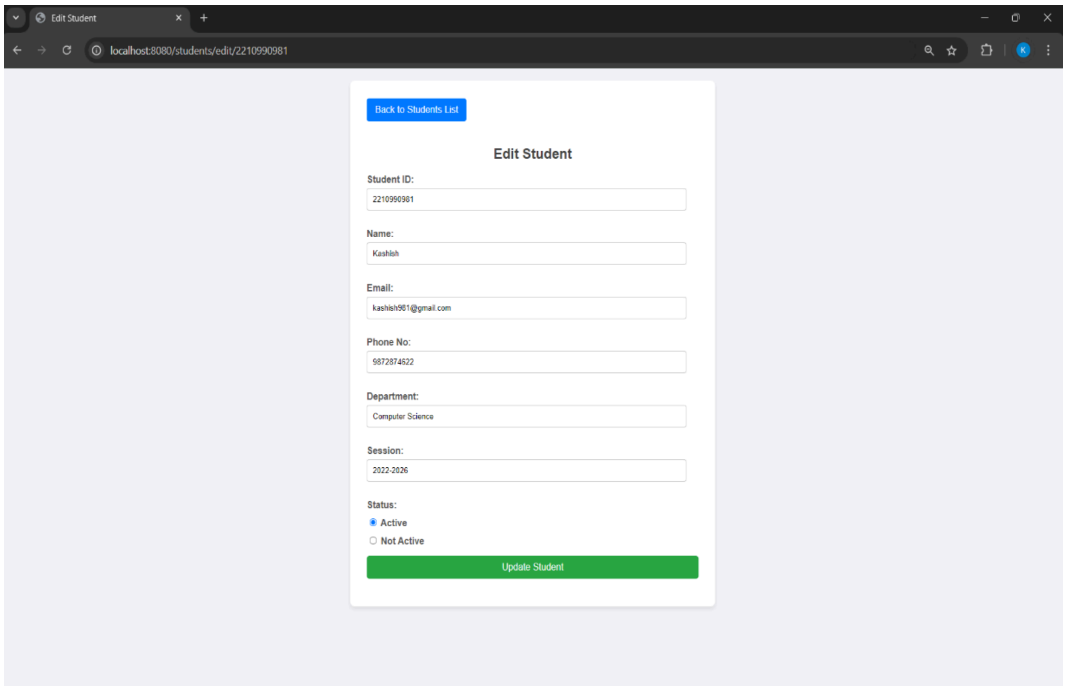

Search by Student ID or Name

Search

Existing Students

Student ID	Name	Action
2210990918	Uday	Edit Delete
2210990981	Kashish	Edit Delete
2210990944	Vanshika	Edit Delete
2210990980	Yashita	Edit Delete
221099983	Pulkit	Edit Delete
2210990936	Vansh Garg	Edit Delete

- **Edit Student details**


Edit Student

localhost:8080/students/edit/2210990981

Back to Students List

Edit Student

Student ID: 2210990981

Name: Kashish

Email: kashish981@gmail.com

Phone No: 9872874622

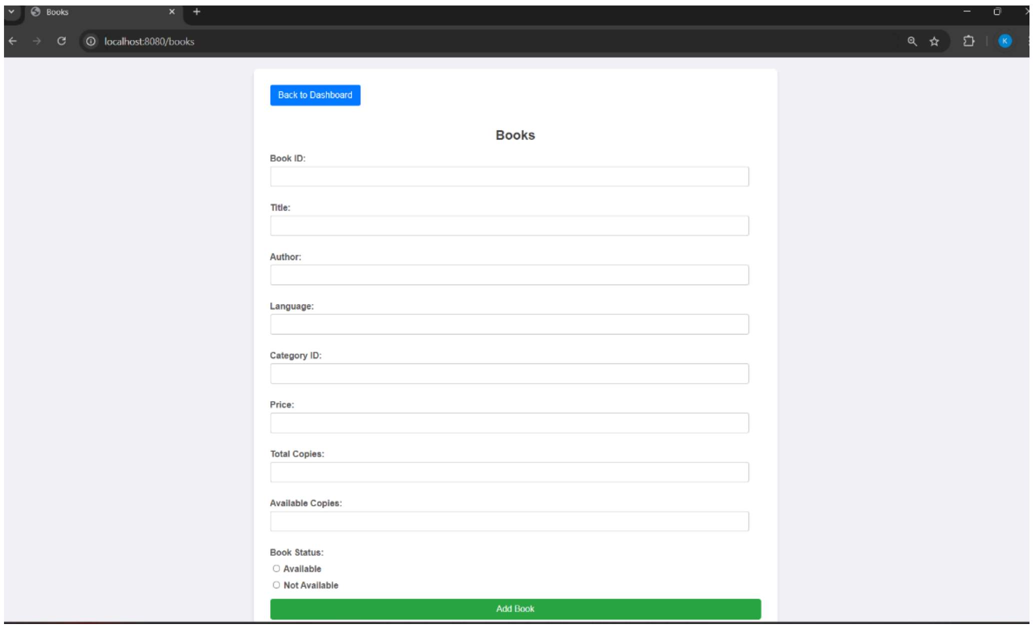
Department: Computer Science

Session: 2022-2026

Status:
[✓ Active]
Not Active

Update Student

- **Add Book and Existing Book details**


Books

localhost:8080/books

Back to Dashboard

Books

Book ID:

Title:

Author:

Language:

Category ID:

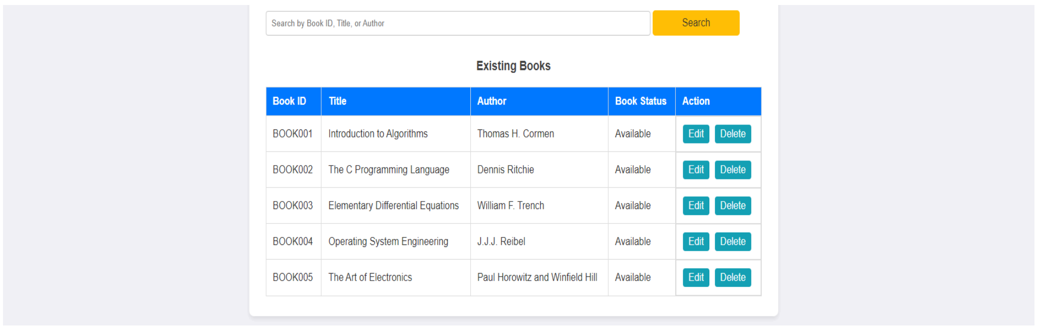
Price:

Total Copies:

Available Copies:

Book Status:
Available
Not Available

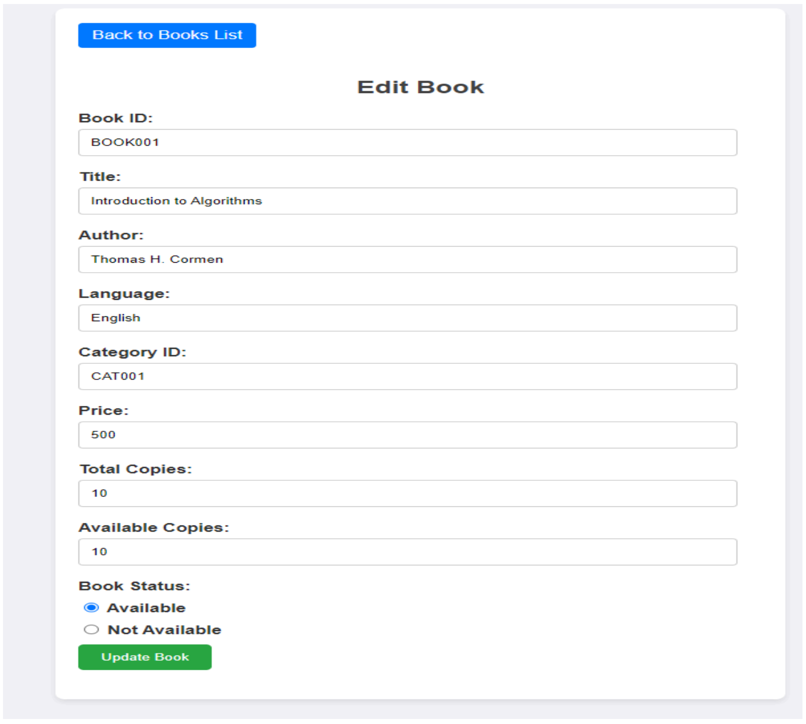
Add Book


Search by Book ID, Title, or Author

Existing Books

Book ID	Title	Author	Book Status	Action
BOOK001	Introduction to Algorithms	Thomas H. Cormen	Available	Edit Delete
BOOK002	The C Programming Language	Dennis Ritchie	Available	Edit Delete
BOOK003	Elementary Differential Equations	William F. Trench	Available	Edit Delete
BOOK004	Operating System Engineering	J.J.J. Reibel	Available	Edit Delete
BOOK005	The Art of Electronics	Paul Horowitz and Winfield Hill	Available	Edit Delete

- **Edit Book details**



Edit Book

Book ID: BOOK001

Title: Introduction to Algorithms

Author: Thomas H. Cormen

Language: English

Category ID: CAT001

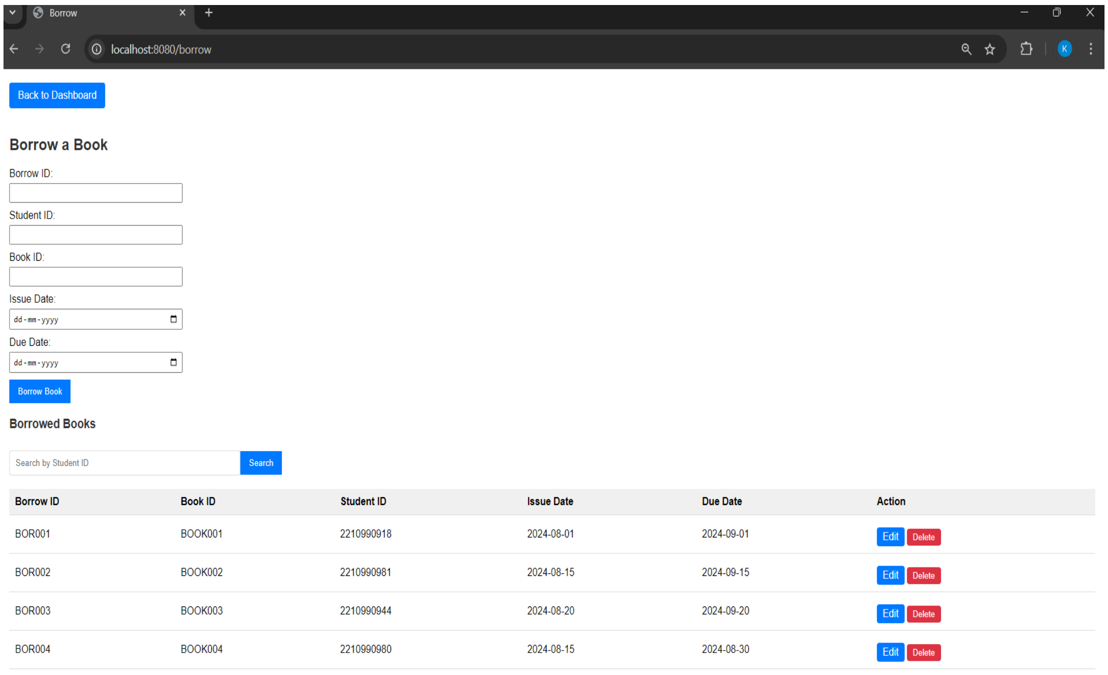
Price: 500

Total Copies: 10

Available Copies: 10

Book Status:
[✓ Available]
Not Available

- **Borrow Book**



Borrow a Book

Borrow ID:

Student ID:

Book ID:

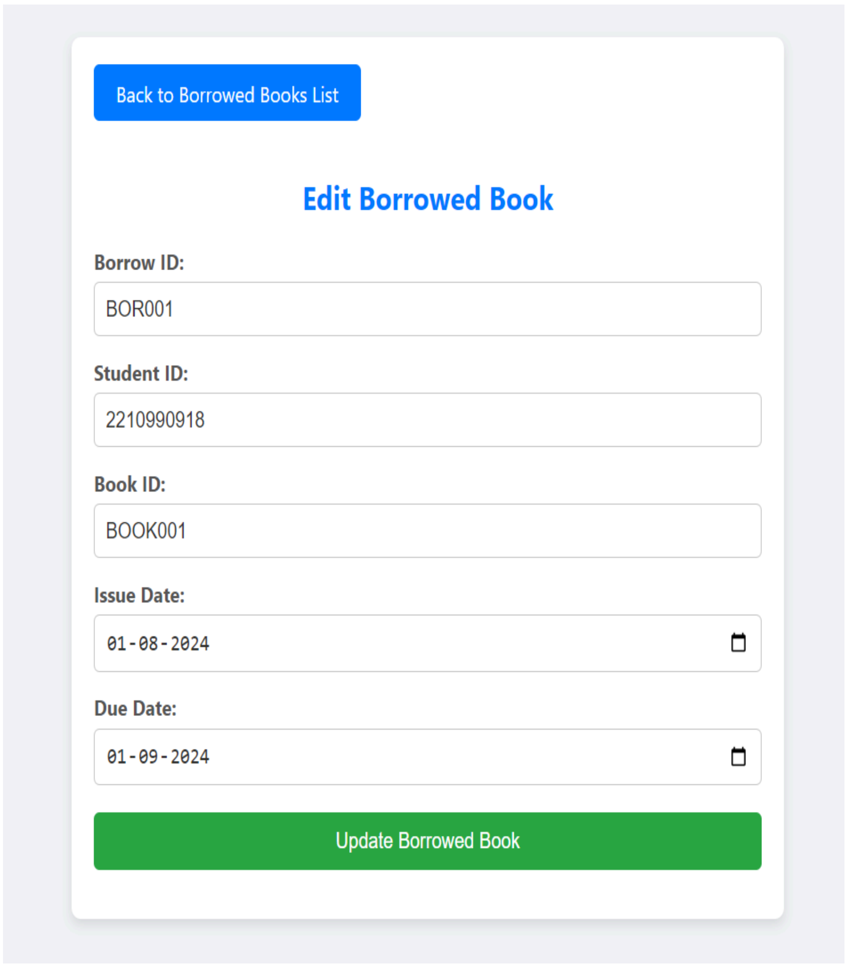
Issue Date: dd-mm-yyyy

Due Date: dd-mm-yyyy

Borrowed Books

Borrow ID	Book ID	Student ID	Issue Date	Due Date	Action
BOR001	BOOK001	2210990918	2024-08-01	2024-09-01	Edit Delete
BOR002	BOOK002	2210990981	2024-08-15	2024-09-15	Edit Delete
BOR003	BOOK003	2210990944	2024-08-20	2024-09-20	Edit Delete
BOR004	BOOK004	2210990980	2024-08-15	2024-08-30	Edit Delete

- **Edit Borrow Book details**



Edit Borrowed Book

Borrow ID: BOR001

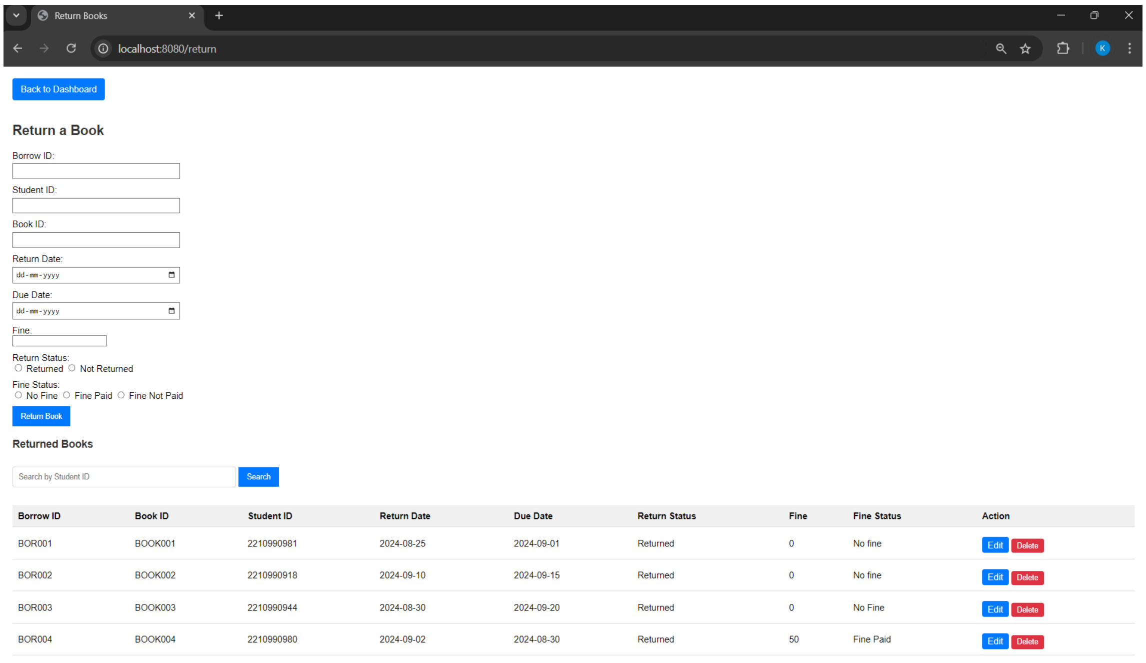
Student ID: 2210990918

Book ID: BOOK001

Issue Date: 01-08-2024

Due Date: 01-09-2024

- **Return Book**



Return a Book

Borrow ID:

Student ID:

Book ID:

Return Date: dd-mm-yyyy

Due Date: dd-mm-yyyy

Fine:

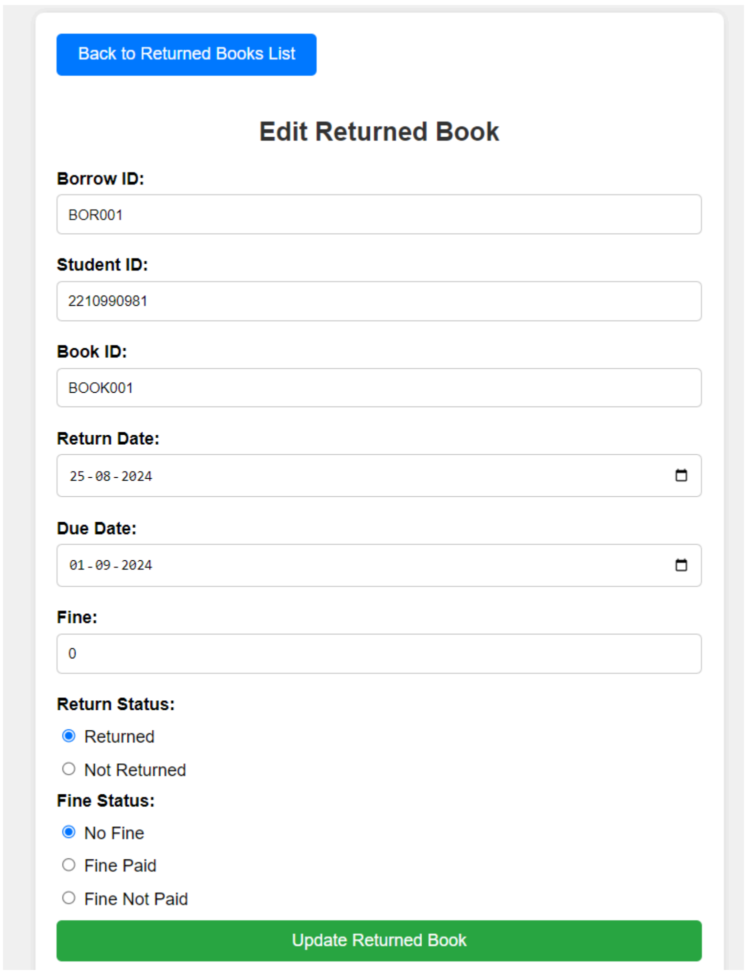
Return Status: ○ Returned ○ Not Returned

Fine Status: ○ No Fine ○ Fine Paid ○ Fine Not Paid

Returned Books

Borrow ID	Book ID	Student ID	Return Date	Due Date	Return Status	Fine	Fine Status	Action
BOR001	BOOK001	2210990981	2024-08-25	2024-09-01	Returned	0	No fine	Edit Delete
BOR002	BOOK002	2210990918	2024-09-10	2024-09-15	Returned	0	No fine	Edit Delete
BOR003	BOOK003	2210990944	2024-08-30	2024-09-20	Returned	0	No Fine	Edit Delete
BOR004	BOOK004	2210990980	2024-09-02	2024-08-30	Returned	50	Fine Paid	Edit Delete

- **Edit Return book details**



Edit Returned Book

Borrow ID: BOR001

Student ID: 2210990981

Book ID: BOOK001

Return Date: 25-08-2024

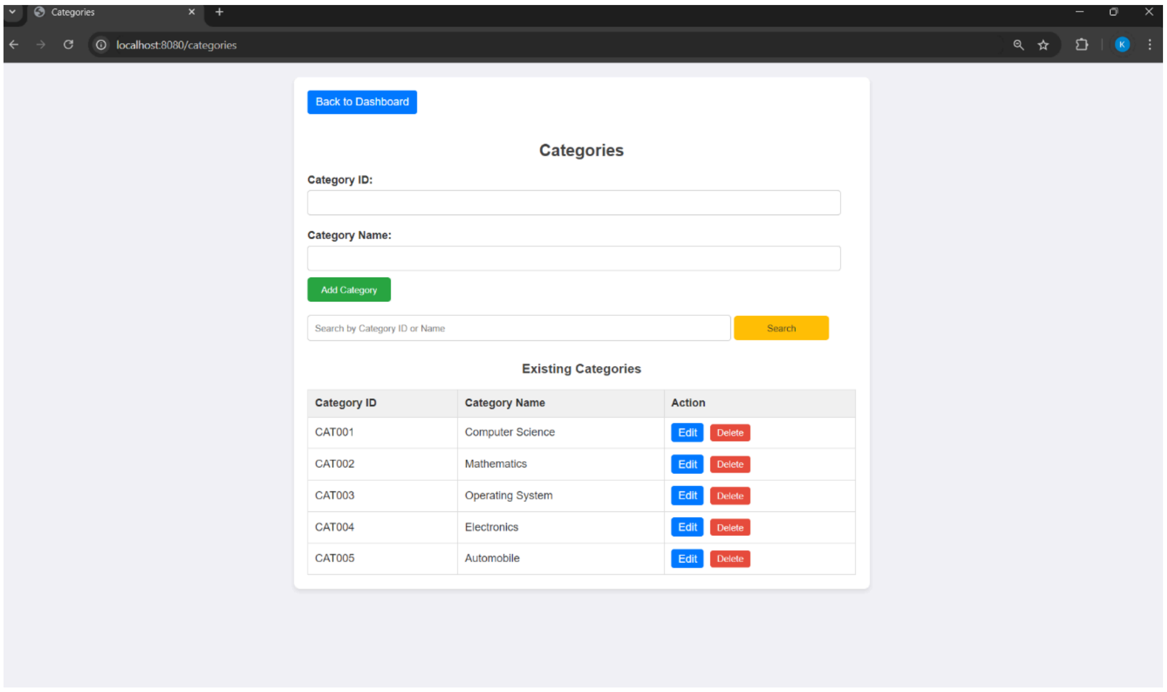
Due Date: 01-09-2024

Fine: 0

Return Status: [✓ Returned] Not Returned

Fine Status: [✓ No Fine] Fine Paid Fine Not Paid

- **Book Categories**



Categories

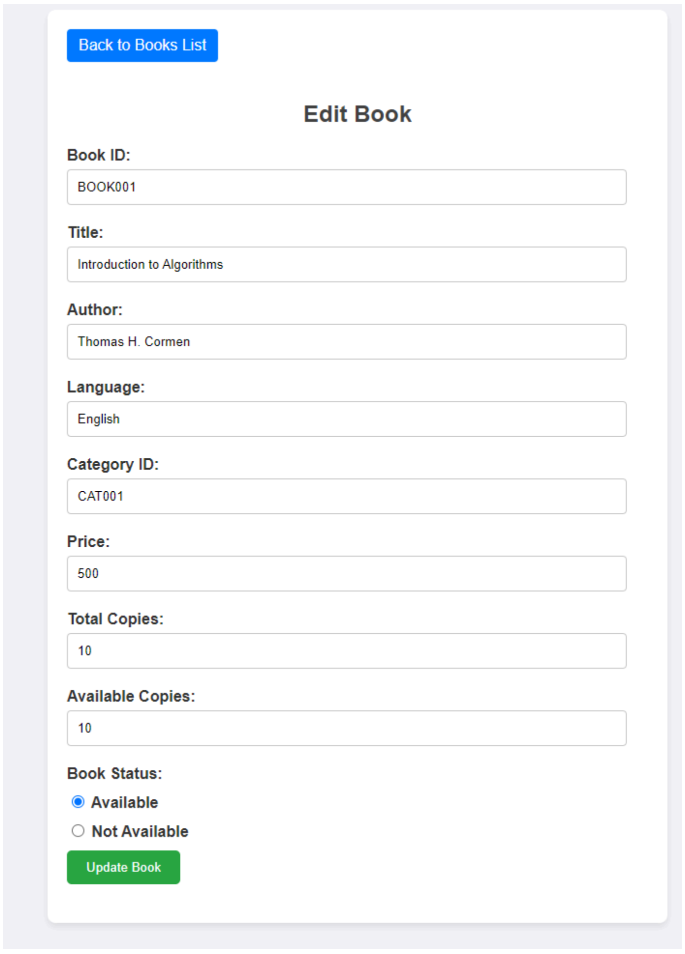
Category ID:

Category Name:

Existing Categories

Category ID	Category Name	Action
CAT001	Computer Science	Edit Delete
CAT002	Mathematics	Edit Delete
CAT003	Operating System	Edit Delete
CAT004	Electronics	Edit Delete
CAT005	Automobile	Edit Delete

- **Edit Categories**



Edit Book

Book ID: BOOK001

Title: Introduction to Algorithms

Author: Thomas H. Cormen

Language: English

Category ID: CAT001

Price: 500

Total Copies: 10

Available Copies: 10

Book Status:

[✓ Available]

Not Available

6. CONCLUSION

The **Library Management System (LMS)** developed using **Node.js**, **Express**, and **MongoDB** offers an efficient solution for managing library resources, including books, students, borrowing, and returning operations. The use of the **MVC (Model-View-Controller)** architecture ensures that the system is well-structured, with the **data layer** managed through MongoDB, the **business logic** handled by Express.js, and a dynamic **user interface** constructed using EJS templates.

The system is designed to streamline the management of library operations by providing an easy-to-use graphical interface for the admin to handle various tasks like adding or removing books, managing student records, and overseeing borrowing and returning activities. The seamless integration of **CRUD operations**, along with session-based user authentication, ensures that only authorized personnel can access sensitive areas of the system.

In conclusion, the Library Management System is a scalable and maintainable solution that meets the essential needs of library administration. It facilitates efficient library management, reduces manual effort, and helps improve the overall user experience for both library staff and students. The system's modular structure allows for future enhancements, such as adding advanced features like automated notifications for overdue books, detailed reporting, and user role management.

7. FUTURE SCOPE

The **Library Management System (LMS)** developed with **Node.js**, **Express**, and **MongoDB** can be expanded and enhanced to address evolving needs in library management and user experience. Some of the key areas for future development include:

1. **Advanced Search and Filtering**: Implementing a more sophisticated search mechanism with filters for books (by title, author, genre, availability, etc.) and students (by name, department, session) can make the system more user-friendly and efficient for large databases.
2. **Automated Notifications**: Adding functionality for automated email or SMS notifications to students about due dates, overdue books, fines, and reminders will enhance user engagement and ensure timely returns.
3. **Role-Based Access Control (RBAC)**: Introducing role-based access control would allow different users (e.g., librarians, admins, students) to have varying permissions, thereby increasing the security of the system. Admins could manage all aspects, while librarians and students would have restricted access based on their roles.

4. **Analytics and Reporting**: The addition of reporting features, such as detailed borrowing history, most borrowed books, overdue books, and student activity reports, can provide valuable insights for library management and help optimize resources.
5. **Mobile Application Integration**: Developing a mobile application or integrating mobile-friendly features using frameworks like React Native or Flutter could enhance accessibility, allowing users to check availability, borrow books, and manage their accounts directly from their smartphones.
6. **Digital Book Support**: Expanding the system to manage e-books or digital resources would provide students with access to online materials, making the system more versatile and adaptable to modern libraries.
7. **Recommendation System**: Implementing a recommendation system for books based on borrowing history or student preferences can offer a personalized experience, similar to how online platforms suggest content.
8. **Barcode/QR Code Scanning**: Integrating barcode or QR code scanning for book borrowing and returning processes would streamline operations and reduce human errors during manual data entry.
9. **Cloud Integration**: Moving the system to a cloud-based infrastructure will improve scalability, enabling the system to handle more data and users efficiently. It will also support remote access for staff and students from various locations.

8. REFERENCES

- **MongoDB Documentation:** https://www.mongodb.com/docs/
- **Express.js Documentation:** https://expressjs.com/
- **Node.js Documentation**: https://nodejs.org/docs/latest/api/
- **FreeCodeCamp:** https://www.freecodecamp.org/news/get-started-with-nodejs/
- **MDN Web Docs:**

  https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
- **W3Schools:** https://www.w3schools.com/nodejs/