# Project Based Evaluation

Backend Engineering Project

Report Semester-V (Batch-2023)

# Library Management System

**Supervised By:**

Dr. Salil Bharany

**Submitted By:**

Komal , 2310992042

Madhu, 2310992056

Mehak , 2310992067

Diksha,2310992592

**Department of Computer Science and Engineering**
**Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab**

# INDEX

## Abstract

The Library Management System is a full-stack web-based application developed to automate and efficiently manage library operations in a digital environment. The system aims to replace traditional manual record-keeping methods with a centralized, secure, and user-friendly platform. It is designed using a React.js frontend and a Node.js with Express.js backend, following a RESTful architecture to ensure smooth communication between the client and server.

The application is structured into publicly accessible pages and secured, role-based dashboards. Public links allow users to browse general library information and view available books without authentication. To protect sensitive operations and data, secure login mechanisms are implemented using JWT (JSON Web Tokens) along with bcrypt-based password hashing, ensuring confidentiality and integrity of user credentials.

The system supports role-based access control, differentiating functionalities for Admin and Student users. The Admin dashboard provides full administrative privileges, including the ability to add, edit, and delete library members, add new books, update book details, and remove books from the system. This allows efficient management of library resources and user records. The Student dashboard, on the other hand, provides limited access, allowing students to view available books and access their personal dashboard, ensuring secure and controlled interaction with the system.

The backend services are responsible for handling business logic, API requests, authentication, and database operations. MongoDB is used as the database management system, offering scalable and flexible storage of data related to books, users, and transactions. Middleware is used to validate requests, protect routes, and maintain system reliability.

This project demonstrates the practical implementation of full-stack web development concepts, including frontend design, backend API development, authentication and authorization, database integration, and secure data handling. The Library Management System enhances operational efficiency, minimizes manual errors, improves data security, and provides a reliable digital solution suitable for modern educational institutions and libraries.

# 1. Introduction

The rapid growth of information technology has transformed the way organizations manage data and operations. Libraries, which traditionally rely on manual record-keeping and physical registers, often face challenges such as data redundancy, human errors, and inefficient retrieval of information. To address these limitations, the Library Management System has been developed as a full-stack web application that automates library processes and provides a secure, efficient, and user-friendly digital solution.

This system is designed using modern web technologies, integrating frontend and backend components to ensure seamless interaction between users and the system. The project focuses on role-based access, data security, and efficient management of library resources.

## 1.1 Background

Traditionally, libraries maintain records of books, members, and transactions using manual registers or basic software tools. Such methods are time-consuming, prone to errors, and difficult to manage as the number of users and resources increases. Searching for books, maintaining member details, and tracking issued or returned books become inefficient in manual systems.

With the advancement of web technologies, there is a growing need for automated and centralized systems that can manage large volumes of data securely and efficiently. The Library Management System has been developed to overcome these limitations by providing a digital platform that simplifies library operations and enhances accessibility for both administrators and students.

## 1.2 Objectives

The primary objectives of the Library Management System are:

- To automate the management of library resources and user records.
- To provide role-based access control for administrators and students.
- To enable administrators to add, edit, and delete members and books efficiently.

- To allow students to view available books and access their personal dashboards.
- To ensure secure authentication and authorization using modern security techniques.
- To reduce manual effort, errors, and redundancy in library operations.
- To provide a scalable and maintainable system for future enhancements.

## 1.3 Significance

The Library Management System plays a significant role in improving the efficiency and reliability of library operations. By digitizing records and automating processes, the system minimizes human errors and saves time. It enhances data security through authentication and authorization mechanisms and ensures that sensitive operations are accessible only to authorized users.

This project also serves as a practical implementation of full-stack web development concepts, allowing students to gain real-world experience in frontend development, backend API creation, database integration, and secure system design. The system is suitable for academic institutions and small to medium-sized libraries seeking a modern digital solution.

## 1.4 Scope of the Project

The scope of this project includes the design and implementation of a web-based library management system with the following functionalities:

- Public access to view general library information and available books.
- Secure login for Admin and Student users.
- Admin functionalities such as managing members and books.
- Student functionalities such as viewing available books and personal dashboard access.
- Secure backend APIs for handling data operations.
- Database integration for storing and managing library data.

The project does not include online payment systems or advanced recommendation engines, but it provides a strong foundation for future expansion.

## 1.5 Technologies used

The technologies used in the development of the Library Management System are:

- Frontend: React.js, HTML, CSS, JavaScript
- Backend: Node.js, Express.js
- Database: MongoDB
- Authentication: JWT (JSON Web Tokens)
- Security: bcrypt for password hashing
- API Communication: RESTful APIs, Axios
- Development Tools: Visual Studio Code, Git

These technologies were chosen to ensure performance, scalability, and security.

## 1.6 Challenges Encountered

During the development of the project, several challenges were encountered:

- Implementing secure authentication and role-based authorization.
- Protecting routes for Admin and Student dashboards.
- Managing state and API communication between frontend and backend.
- Designing a user-friendly interface while maintaining security.
- Handling errors and validations efficiently.

These challenges were addressed through proper planning, testing, and the use of middleware and secure coding practices.

## 1.7 Methodology

The project follows a modular and iterative development methodology. Initially, requirements were analyzed and system functionalities were defined. The frontend and backend were developed as separate modules to ensure clear separation of concerns. RESTful APIs were created for data communication, and authentication mechanisms were implemented to secure access.

Testing was conducted at each stage to identify and fix issues early. The system was continuously refined based on testing outcomes to improve performance, security, and usability.

## 1.8 Expected Outcomes

The expected outcomes of the Library Management System are:

- A fully functional and secure web-based library management application.
- Improved efficiency and accuracy in managing library resources.
- Secure and role-based access for Admin and Student users.
- Reduced manual workload and operational errors.
- A scalable system that can be enhanced with additional features in the future.

# 2. Problem Definition and Requirements

## 2.1 Problem Statement

In today's university environments, students frequently encounter situations where they require specific resources for a short period — such as a DSLR camera for a project, a tripod for a video shoot, or a textbook just before an exam. At the same time, these resources already exist within the campus community, owned by fellow students who are not utilizing them at that moment. However, due to the absence of a structured platform, these needs often go unmet, leading to unnecessary purchases, wasted resources, and missed opportunities for collaboration. To enable secure and real-time communication between users within the platform while maintaining message persistence and data integrity.

## 2.2 Software Requirements

The software requirements include tools and technologies needed for development, deployment, and execution of the system.

**Frontend Software**

- React.js
- HTML5
- CSS3
- JavaScript
- Axios (for API communication)

**Backend Software**

- Node.js
- Express.js
- RESTful APIs

**Database**

- MongoDB
- Mongoose (Object Data Modeling)

**File Upload & Processing**

- Multer (for file uploads)
- XLSX / Excel parsing library (for bulk student data upload)

**Security & Authentication**

- JWT (JSON Web Tokens)
- bcrypt (for password hashing)
- Middleware for route protection

**Containerization & Deployment**

- Docker
- Docker Compose (optional, for multi-container setup)

**Development Tools**

- Visual Studio Code
- Git & GitHub
- Postman (API testing)
- Web Browser (Chrome / Firefox)

**Operating System**

- Windows / Linux / macOS

## 2.3 Hardware Requirements

The hardware requirements specify the minimum system configuration required for development, deployment, and usage.

**For Development & Deployment**

- Processor: Intel Core i5 or equivalent
- RAM: Minimum 8 GB
- Storage: Minimum 50 GB free disk space
- Internet Connection: Required for Docker images, dependencies, and deployment

**For End Users**

- Desktop / Laptop / Mobile device
- Modern web browser
- Stable internet connection

## 2.4 Data Sets

The Library Management System manages structured datasets securely stored in a MongoDB database. These datasets are validated and processed efficiently, including support for bulk uploads.

**1. Books Dataset**

Book ID, Title, Author, Category, ISBN, Total Copies, Available Copies, Availability Status.

**2. Members (Students) Dataset**

Student ID, Name, Email, Phone Number, Role (Student), Account Status. **Note:** The system supports bulk student addition via Excel file upload, where records are automatically validated and inserted into the database.

**3. Admin Dataset**

Admin ID, Name, Email, Encrypted Password, Role (Admin).

**4. Transaction Dataset**

Transaction ID, Student ID, Book ID, Issue Date, Return Date, Transaction Status.

**5. Authentication Dataset**

User Credentials, Hashed Passwords, JWT Tokens, Login Sessions.

# 3. Proposed Design / Methodology

## 3.1 System Overview

The Library Management System is a web-based, full-stack application designed to manage library operations digitally. The system provides both public access and secure, role-based dashboards for administrators and students. Public users can view general information and available books, while protected dashboards are accessible only to authenticated users.

The system supports secure authentication, book and member management, transaction handling, and bulk student registration through Excel file uploads. The application is containerized and deployed using Docker, ensuring consistency across development and production environments.

## 3.2 Architecture

**1.Presentation Layer (Frontend)**

- Developed using React.js
- Provides interactive user interfaces for public pages, Admin dashboard, and Student dashboard
- Communicates with backend services using RESTful APIs

**2. Application Layer (Backend)**

- Developed using **Node.js and Express.js**
- Handles business logic, authentication, authorization, and data validation
- Processes Excel file uploads for bulk student addition
- Protects routes using middleware and JWT-based authentication

**3. Data Layer (Database)**

- Uses MongoDB for storing books, users, and transaction data
- Ensures data consistency and scalability

- This layered architecture enhances maintainability, security, and scalability of the system.

## 3.3 System Components

The major components of the Library Management System are:

**1. Frontend Component**

- Public pages for browsing available books
- Admin dashboard for managing books and members
- Student dashboard for viewing personal information and available books
- State management and API integration using React hooks

**2. Backend Component**

- RESTful API services
- Authentication and authorization mechanisms
- Excel file upload and parsing module
- Input validation and error handling

**3. Database Component**

- Collections for books, students, admins, and transactions
- Secure storage of encrypted passwords and authentication tokens

**4. Security Component**

- JWT for session management
- bcrypt for password encryption
- Middleware for route protection and role validation

**5. Deployment Component**

- Docker containers for frontend, backend, and database
- Environment configuration using Docker

## 3.4 Methodology

The project follows an iterative and modular development methodology, allowing continuous improvement and testing at each stage. The methodology includes the following phases:

1. **Requirement Analysis** – Identifying functional and non-functional requirements.
2. **System Design** – Designing architecture, database schema, and user roles.
3. **Implementation** – Developing frontend and backend modules independently.
4. **Integration** – Connecting frontend with backend APIs.

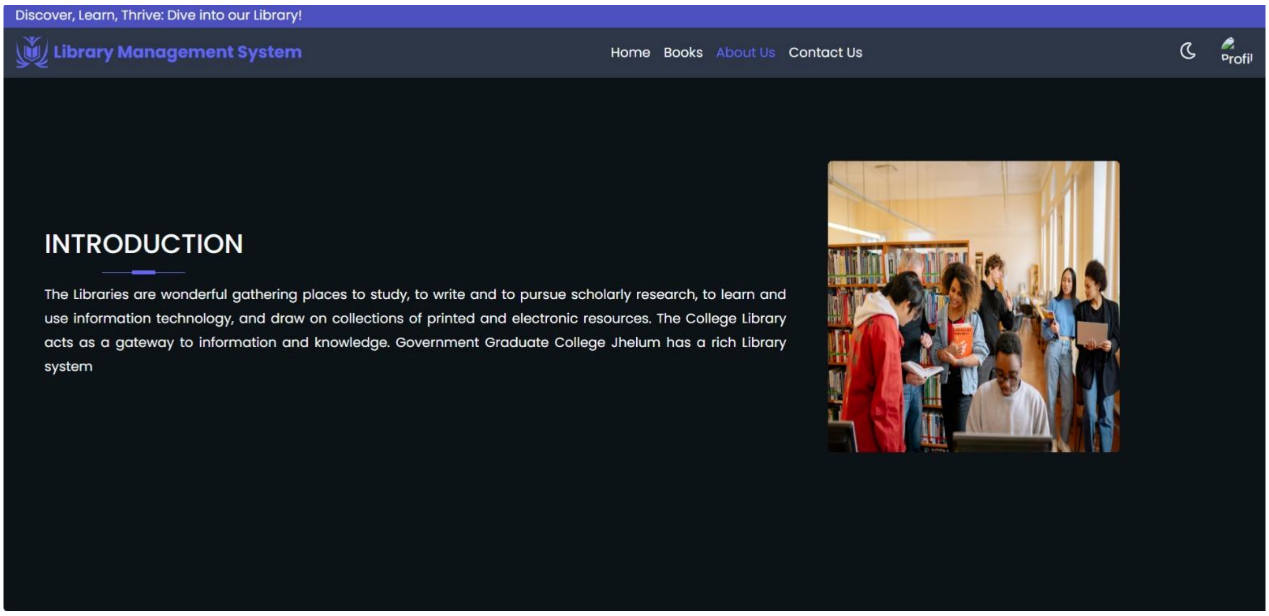
5. **Testing** – Validating functionality, security, and performance.
6. **Deployment** – Containerizing the application using Docker.

This approach ensures flexibility, reduced development risks, and improved software quality.

## 3.5 Deployment Strategy

- The Library Management System is deployed using Docker, which provides a containerized environment for consistent execution across platforms.
- **Deployment Process**
  - Separate Docker containers are created for:
  - Frontend (React.js)
  - Backend (Node.js & Express.js)
  - Database (MongoDB)
  - Environment variables are managed securely within containers
  - Containers can be orchestrated using Docker Compose
- **Advantages of Docker Deployment**
  - Platform independence
  - Simplified setup and maintenance
  - Easy scalability and updates
  - Improved reliability and performance
- This deployment strategy ensures that the system can be easily deployed, scaled, and maintained in real-world environments.

## 4. Results


Discover, Learn, Thrive: Dive into our Library!

Library Management System

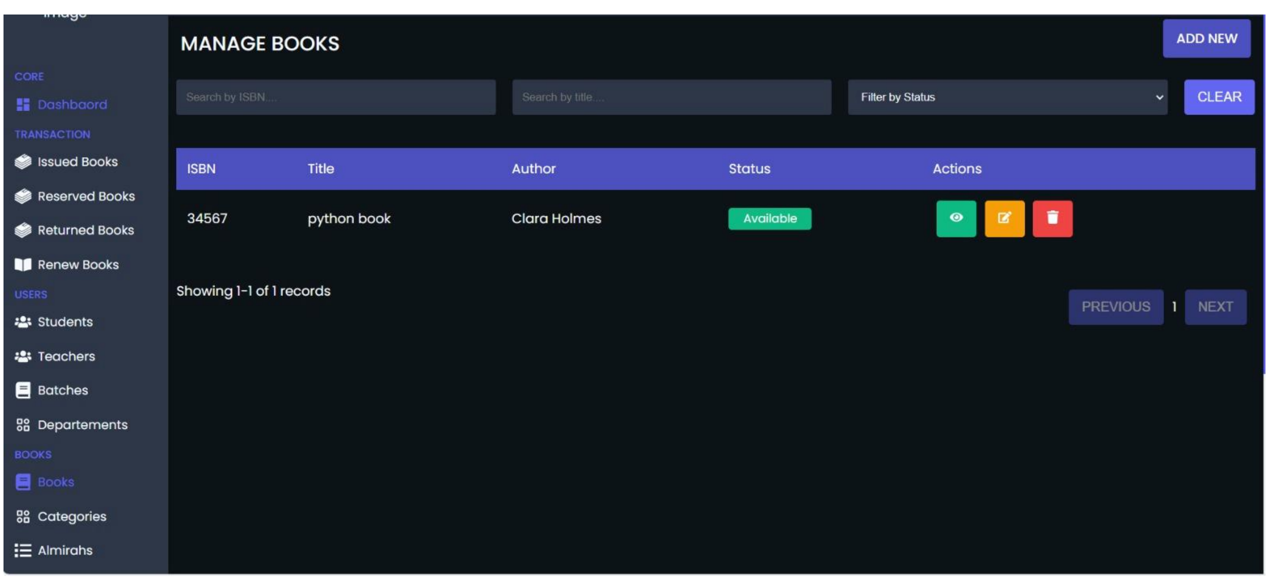
Home Books About Us Contact Us

Profil

INTRODUCTION

The Libraries are wonderful gathering places to study, to write and to pursue scholarly research, to learn and use information technology, and draw on collections of printed and electronic resources. The College Library acts as a gateway to information and knowledge. Government Graduate College Jhelum has a rich Library system

Fig 4.1:About us



MANAGE BOOKS

ADD NEW

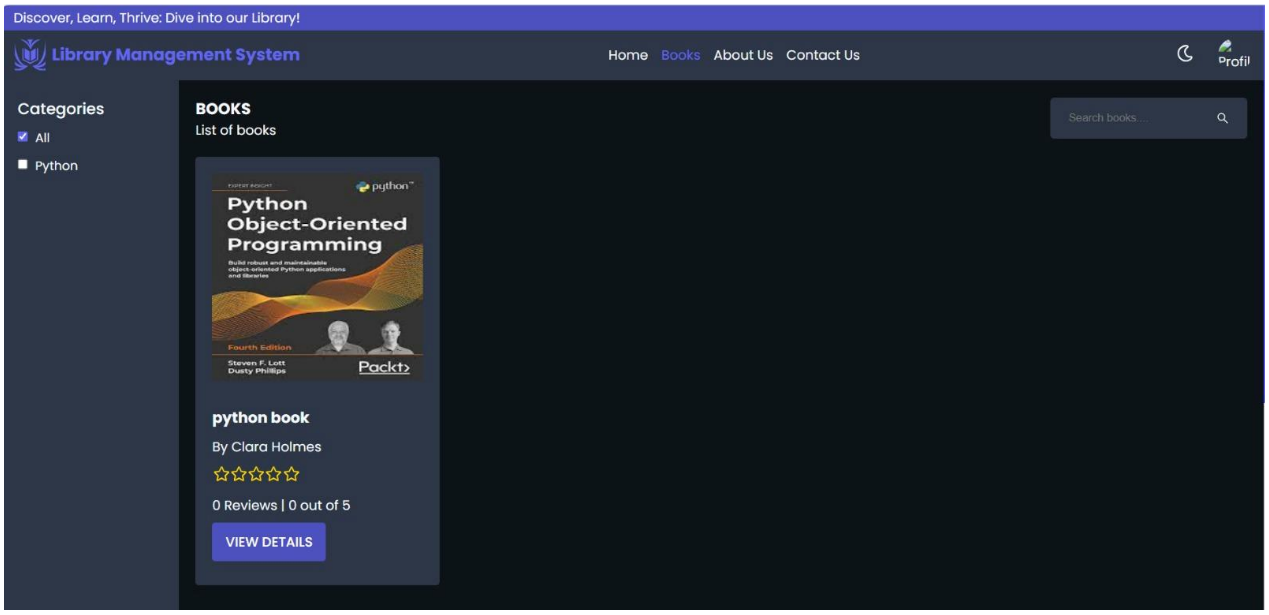
Search by ISBN... | Search by title... | Filter by Status | CLEAR

| ISBN | Title | Author | Status | Actions |
|---|---|---|---|---|
| 34567 | python book | Clara Holmes | Available | |

Showing 1-1 of 1 records

PREVIOUS 1 NEXT

Fig 4.2: Manage Books


Discover, Learn, Thrive: Dive into our Library!

Library Management System

Home Books About Us Contact Us

Categories

- [x] All
- [ ] Python

BOOKS

List of books

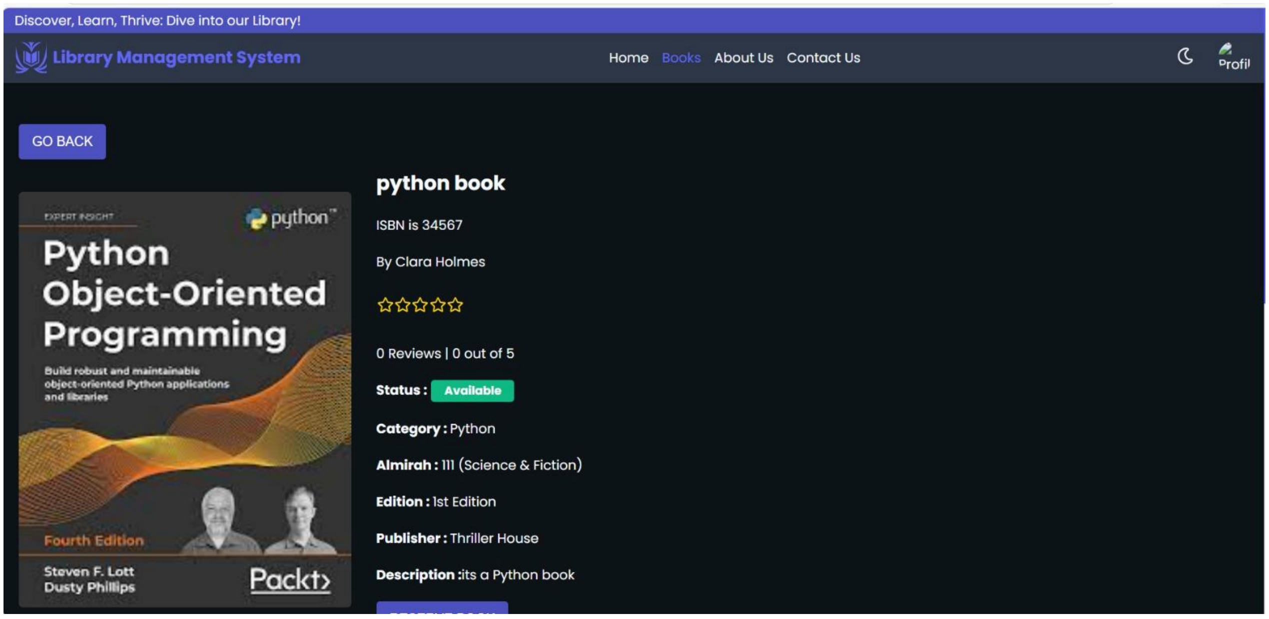
python book

By Clara Holmes

0 Reviews | 0 out of 5

VIEW DETAILS

Fig 4.3:Categories


Discover, Learn, Thrive: Dive into our Library!

Library Management System

Home Books About Us Contact Us

GO BACK

python book

ISBN is 34567

By Clara Holmes

0 Reviews | 0 out of 5

**Status :** Available

**Category :** Python

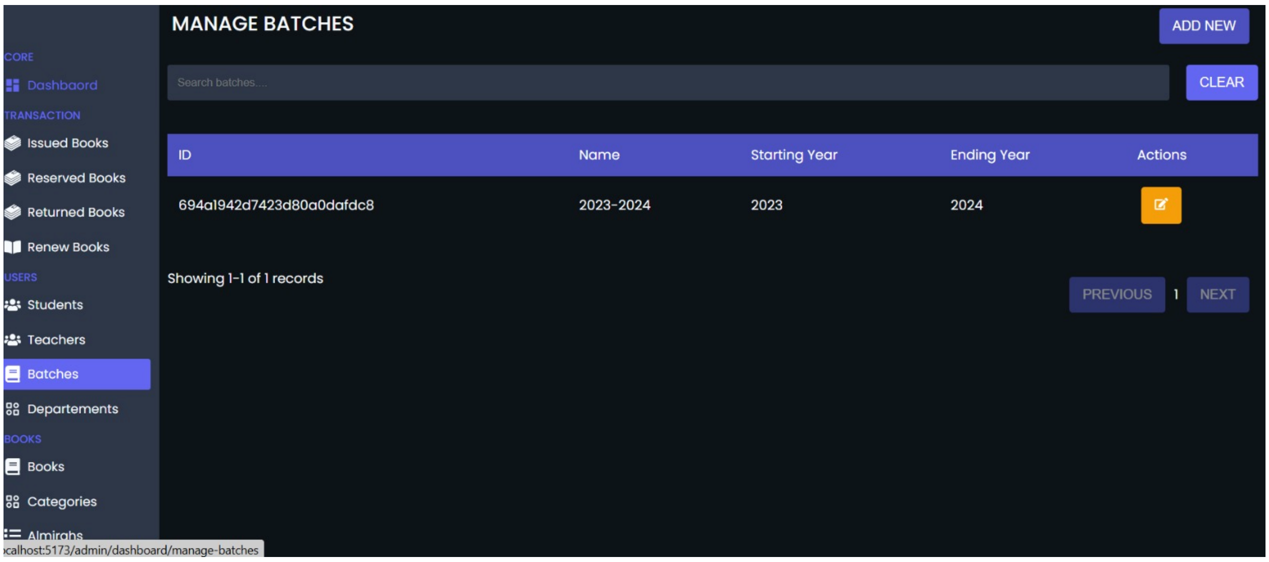
**Almirah :** 111 (Science & Fiction)

**Edition :** 1st Edition

**Publisher :** Thriller House

**Description :** its a Python book

Fig 4.4:Book Description

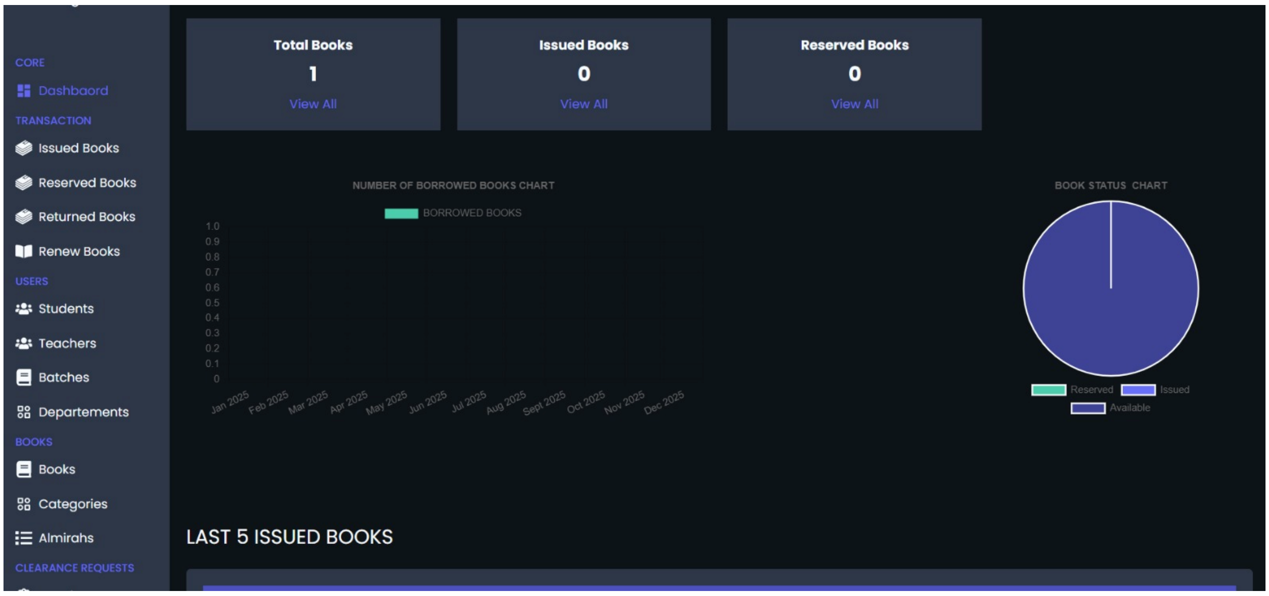


MANAGE BATCHES

| ID | Name | Starting Year | Ending Year | Actions |
|---|---|---|---|---|
| 694a1942d7423d80a0dafdc8 | 2023-2024 | 2023 | 2024 | |

Showing 1-1 of 1 records

Fig 4.5:Managing Batches

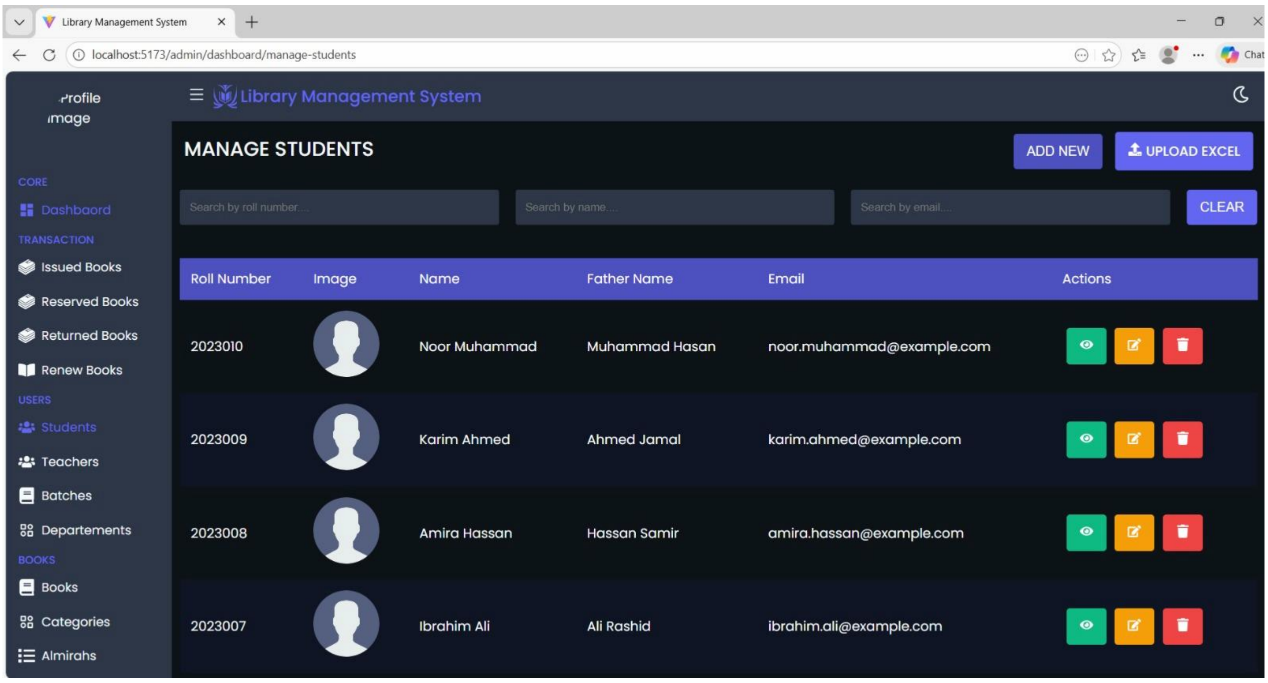


Total Books 1 View All

Issued Books 0 View All

Reserved Books 0 View All

Fig 4.6:Dashboard



| Roll Number | Image | Name | Father Name | Email | Actions |
|---|---|---|---|---|---|
| 2023010 | | Noor Muhammad | Muhammad Hasan | noor.muhammad@example.com | |
| 2023009 | | Karim Ahmed | Ahmed Jamal | karim.ahmed@example.com | |
| 2023008 | | Amira Hassan | Hassan Samir | amira.hassan@example.com | |
| 2023007 | | Ibrahim Ali | Ali Rashid | ibrahim.ali@example.com | |

**Fig 4.7:Adding Multiple Students**

## 5. References

- Node.js Documentation – https://nodejs.org/docs
- Express.js Documentation – https://expressjs.com
- React Documentation – https://react.dev
- MongoDB Documentation – https://www.mongodb.com/docs
- Mongoose ODM Documentation – https://mongoosejs.com
- JWT Authentication Guide – https://jwt.io/introduction
- bcrypt.js Documentation – https://www.npmjs.com/package/bcryptjs
- RESTful API Design Guidelines – https://restfulapi.net