Major Project
On
Online Library Management System

A Major Project Report Submitted to SAGE University, Indore

Towards Partial fulfillment for the award of Bachelor of

Computer Application (BCA) degree

Guided by:

Prof. Anil Patidar

Submitted by:

Akshat Gangrade

22COA2BCA0087

Institute of Computer Application

SAGE University, Indore

www.sageuniversity

SAGE University, Indore

Approval Sheet

The project entitled **"Online Library Management System"** submitted by **Akshat Gangrade** approved as partial fulfillment for the award of the **BACHELOR OF COMPUTER APPLICATION (BCA)** by SAGE University, Indore.

Internal Examiner **External Examiner**

Date: **Date:**

SAGE University, Indore

CERTIFICATE

This is to certify that the project work entitled **"Online Library Management System"** has been carried out by **Akshat Gangrade** student of **BACHELOR OF COMPUTER APPLICATION** VI Semester under my supervision and guidance. He has submitted this project report towards partial fulfillment for the award of the **Bachelor of Computer Application** by **SAGE University**, Indore.

Dr. Maya Rathore

(Academic Coordinator)

Prof. Anil Patidar

(Guide)

Dr. Sanjay Dubey

(HOD)

Dr. Rashmi Yadav

(HOI)

SAGE University, Indore

RECOMMENDATION

The project entitled **"Online Library Management System**" submitted by **Akshat Gangrade** is a satisfactory account University, Indore Prof. Anil Patidar (Guide)of the bona fide work done under my supervision is recommended towards partial fulfillment for the award of the **Bachelor of Computer Application** by **SAGE University**, Indore.

Date:

Dr. Sanjay Dubey
(HOD)

Prof. Anil Patidar
(Guide)

Dr. Rashmi Yadav
(HOI)

SAGE University, Indore

ACKNOWLEDGEMENT

First and foremost, I would like to express my thankfulness towards **Mr. Anil Patidar** of **INSTITUTE OF COMPUTER APPLICATION** for extending all the facilities needed to carry out this work, I take pride in saying that I have successfully completed my project work under his/her able guidance. He was a major support to us throughout the project, being available at odd hours with his/her ideas, inspiration and encouragement. It is through his/her masterful guidance that I have been able to complete my project work.

I am also thankful to **Dr. Rashmi Yadav (HOI) and Dr. Sanjay Dubey (HOD)** for giving their guidance throughout the Project phase.

Akshat Gangrade

(STUDENT)

SAGE University, Indore

CANDIDATE DECLARATION

I hereby declare that the work which is being presented in this project report entitled **"Online Library Management System"** in partial fulfillment for the award of **Bachelor of Computer Application** is an authentic record of my own work carried out under the supervision and guidance of **Prof. Anil Patidar**, **SAGE University**, Indore.

I am fully responsible for the matter embodied in this report and it has not been submitted elsewhere for the award of any other degree.

Date: **Akshat Gangrade**

Table of Contents

1. INTRODUCTION

1.1 General Introduction:

From ancient times, knowledge has been preserved and shared by people across the world in written form. Initially, words were carved onto copper plates or stone using sharp tools to preserve information for future generations. As civilizations advanced, the invention of paper revolutionized the way knowledge was recorded. People began using ink and bird feathers to write manuscripts, making it easier to store and share information. However, preserving physical records such as books came with numerous challenges—pages could be torn, stolen, lost, or damaged by natural elements like water, fire, or pests such as termites and bookworms.

Historically, libraries were managed manually by a group of staff responsible for maintaining inventory, issuing books, and keeping handwritten logs. These traditional methods were labor-intensive and prone to human error. With the exponential growth of recorded information over the years, managing library operations manually became increasingly complex and inefficient.

In contrast, the 21st century has witnessed rapid digital transformation across every sector, including education and library sciences. The introduction of computerized systems has made it possible to organize, store, and retrieve data instantly. In today's digital world, information is just one click away, and the need for efficient digital solutions has become essential. Digitizing libraries not only reduces paperwork but also enhances data security, retrieval, and management.

The **Online Library Management System** project aims to overcome the limitations of manual systems. It enables both library staff and students to interact with the system in a transparent and accessible manner. Students can check book availability, view issue and return dates, and track fines for overdue returns. Meanwhile, librarians can manage book inventories, student profiles, and records efficiently.

The system includes two user roles:

- **Admin**: Responsible for adding, updating, and deleting books, registering students, and managing user alerts.
- **Student/user**: Required to register and log in before accessing services. If a book is not available for a physical issue, the student may be offered access to an online version of the book in PDF format.

1.2 Project Objectives

The primary objective of the **Library Management System** is to digitally manage all records related to books, students, and transactions in a library. It aims to replace outdated manual systems with a user-friendly, efficient, and secure digital interface that can be used by both administrators and students.

Key objectives include:

- To provide a robust platform to store and manage all data related to books, authors, and students.
- To enable students to easily check book availability, issue history, and pending returns or fines.
- To empower the library administrator with tools to add, delete, or modify book records and user profiles.
- To implement a secure login system that allows only authorized users to access or modify data.
- To provide an intuitive graphical user interface (GUI) built with Django and Python that ensures ease of use, even for non-technical users.
- To enhance data accuracy by reducing manual entry errors and to provide fast data retrieval capabilities.
- To facilitate the addition of future features like digital book uploads (PDF), return reminders, and fine calculation.
- To create a scalable and secure system that can handle growing numbers of users and books over time.

This application, developed using **Python (Django framework)** and **MySQL**, has a clean design, is simple to use, and supports full-stack web development principles. The project is built primarily for administrative use but includes essential features accessible to students for interaction and self-service.

1.3 Scope of the Project

The **Online Library Management System** is developed with the intent to streamline library operations within educational institutions such as colleges and universities. The scope of this project encompasses both administrative tasks and user interactions, creating a centralized platform that supports day-to-day library activities.

This system enables the management of books, authors, categories, and student registrations while also providing users with access to important information like book availability, issue/return dates, and overdue fines. The software is scalable and can be extended to support advanced features such as:

- E-book integration for remote access
- Fine automation based on due dates
- QR code or barcode scanning for book inventory
- Mobile app or SMS notifications

The project is primarily intended for use within **academic institutions**, but its architecture makes it suitable for adaptation to public libraries, private collections, and corporate knowledge centers as well. The system ensures improved **accuracy, efficiency, and user satisfaction** by replacing outdated manual systems with a modern digital interface.

1.4 Significance of the Project

In today's fast-paced educational environment, **time, accuracy, and accessibility** are key factors that influence the effectiveness of library services. The significance of this project lies in its ability to transform how libraries function—from a static storage facility to an interactive, responsive information hub.

Key reasons why this project holds academic and practical importance include:

- 🔍 **Transparency in Transactions**: Students can track their own borrowing history and dues, reducing confusion and disputes.
- ⏱️ **Time-saving for Librarians**: Automated record-keeping and reporting reduce manual workload and improve productivity.
- 📚 **Better Access to Resources**: Even if a physical copy of a book is unavailable, users may still access digital versions or request alerts when available.
- 💾 **Data Integrity and Backup**: Unlike physical records, digital data can be backed up and restored easily in case of failure or disaster.

2. Problem Statement / Abstract

Library Management System (LMS) is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simpler. The Library Management has been designed to computerize and automate the operations performed over the information about the members, book issues and returns and all other operations. This computerization of the library helps in many instances of its maintenance. It reduces the workload of management as most of the manual work done is reduced. With the advancement of technology, it is imperative to exalt all the systems in a user-friendly manner. The Library Management system acts as a tool to transform traditional libraries into digital libraries. In traditional libraries, the students/user has to search for books which are a hassle process and there is no proper maintenance of databases about issues/fines. The admin has to work allotted for arranging, sorting books in the book sales. At the same time, they have to check and monitor the lend/borrow book details with its fine. LMS will assist the admin to work easily. The LMS supports the admin to encounter all the issues concurrently. The users need not stand in a queue for a long period to return/borrow a book from the library. The single PC contains all the data in it. The admin has to assess the system and provide an entry in it. Through LMS the admin can find the book in the bookshelves. The LMS is designed with the basic features such as Admin can add/view/update/delete book. Once he/she ingress into the system they can modify any data in the database. The complete model is developed in PHP/MySQL language and is used to build the application.

3. Objectives

The primary objective of the Online Library Management System (OLMS) is to streamline and automate all the core functions of a traditional library using a web-based application. This system is designed to improve the management, efficiency, accessibility, and transparency of library operations for administrators, teachers, and students alike. The OLMS serves as a centralized platform where users can search for books, issue and return books, view notifications, and communicate with library staff — all with minimal manual effort and maximum ease.

Primary Objectives:

- **To Digitize Library Operations:**
  Replace outdated, manual methods of book issuance, return, and record-keeping with an automated, digital platform that reduces human error and administrative burden.

- **To Develop a Role-Based Access System:**
  Provide different levels of access and functionality based on user roles — Admin, Teacher, and Student — ensuring data integrity, personalized access, and a secure user experience.

- **To Simplify Book Management:**
  Allow administrators to easily add, update, categorize, and track books, including their availability, issue history, and ISBN details. The system should maintain accurate real-time stock.

- **To Enable Seamless Book Issuing and Returning:**
  Implement a robust system for tracking book issues and returns, including fine calculation (if applicable), due dates, and automatic updating of inventory levels.

- **To Improve Communication:**
  Integrate a notification and feedback mechanism between administrators, teachers, and students to enhance communication regarding library updates, overdue books, and leave requests.

Secondary Objectives:

- **To Provide Feedback and Support Features:**
  Allow students and teachers to submit feedback and suggestions regarding the library system or operations, enabling continuous improvement.

- **To Support Student and Teacher Leave Management:**
  Integrate a leave request module for both students and teachers, where the administrator can approve or reject requests within the system.

- **To Ensure System Security and Data Integrity:**
  Implement authentication mechanisms (login system) and input validations to protect sensitive data and prevent unauthorized access.

- **To Design an Intuitive, User-Friendly Interface:**
  Create a simple, responsive web interface using Django that enables users to navigate, search, and interact with the system effortlessly across devices.

- **To Offer Real-Time Data and Reports:**
  Provide administrators with real-time access to information such as book stock levels, user activity logs, issue/return history, and leave applications.

4. Hypothesis

In traditional academic environments, the management of a physical library involves manual entry, paperwork, delayed communication, and inefficient book tracking. This leads to frequent errors in data handling, book loss, and a time-consuming user experience for both students and library staff. As institutions grow, the need for automation and digital record-keeping becomes more pressing.

The central hypothesis of this project is:

> *"If an online, role-based library management system is implemented using web technologies (Python, Django, SQLite), then it will significantly improve the efficiency, accuracy, and user satisfaction in managing library operations when compared to conventional manual systems."*

🔍 Supporting Assumptions

1. **Manual systems are inefficient**
   Traditional libraries rely heavily on physical logs, registers, and human oversight, which are prone to errors and delays.

2. **Web-based systems are accessible and scalable**
   A web-based solution allows remote access, multi-user interaction, and can be updated or scaled easily as needed.

3. **Role-based design improves control**
   By dividing functionalities among Admin, Teacher, and Student roles, the system will provide clear boundaries and improve system security and usability.

4. **Role-based design improves control**

   By dividing functionalities among Admin, Teacher, and Student roles, the system will provide clear boundaries and improve system security and usability.

5. **Real-time data access reduces turnaround time**

   With automated notifications, instant book search, and inventory updates, both users and staff will experience faster operations.

6. **Digital systems promote eco-friendly practices**

   By eliminating paper-based records, the system will also contribute to environmental sustainability and efficient data storage.

✅ Hypothesis Testing Approach

To test the hypothesis, the following development and evaluation steps are taken:

- **System Development**: Build the system using Python (Django framework), implementing all core library functionalities.
- **User Testing**: Collect feedback from mock users (students, teachers, admin) on usability, speed, and accuracy.
- **Comparative Study**: Compare performance metrics (time taken for issuing/returning books, error rates, feedback satisfaction) between the manual and digital system.
- **Result Analysis**: Use this data to validate whether the new system meets or exceeds expectations in functionality and efficiency.

5. Methodology / Project Plan

The Online Library Management System was developed using a structured and systematic approach, adhering to standard software engineering practices. The development methodology followed in this project is the **Waterfall Model**, which is best suited for academic software projects that require clear, sequential development steps.

🔧 Development Methodology: Waterfall Model

The **Waterfall Model** is a linear and sequential approach to software development. It consists of distinct phases, each of which must be completed before moving on to the next. This method was chosen due to the clarity it offers in tracking project progress and ensuring completeness at every stage.

📌 Phases in the Waterfall Model used:

1. **Requirement Analysis:**
   - Collected requirements from potential users: students, teachers, and librarians.
   - Identified the core functionalities: user registration, book management, issuing/returning, notifications, feedback, and leave requests.
   - Separated functional and non-functional requirements.

2. **System Design**
   - Designed system architecture and database schema.
   - Created ER diagram, DFD (Data Flow Diagram), and Class Diagram.
   - Mapped out modules for Admin, Teacher, and Student.

3. Implementation:

- Developed the project using:
- **Backend**: Python (Django)
- **Frontend**: HTML, CSS, Bootstrap
- **Database**: SQLite
- Modular code structure using separate view files for admin, teacher, and student roles.

4. Testing

- Performed unit testing on each module (book issue, login system, leave request, etc.).
- Conducted integration testing after all modules were combined.
- Simulated real-time user interactions to identify bugs or logical flaws

5. Deployment

- Locally hosted the project using Django's built-in server.
- Demonstrated the system in a LAN environment for evaluation and user testing.
- Prepared the project for cloud deployment if needed in future scope.

6. Maintenance (Planned)

- Although not implemented in this version, provisions have been made for future updates, such as fine calculation, advanced search, and QR-code scanning.

Project Modules and Timeline

Phase	Module/Task	Time Allocation	Status
Phase 1	Requirement Gathering & Planning	1 Week	Completed
Phase 2	Design (UI/UX + DB Design + Diagrams)	1 Week	Completed
Phase 3	Admin Panel (User mgmt. + Books)	1 Week	Completed
Phase 4	Student Panel (Book search, feedback)	1 Week	Completed
Phase 5	Teacher Panel (Leave, notification)	1 Week	Completed
Phase 6	Integration + Testing	1 Week	Completed
Phase 7	Final Report, Deployment & Review	1 Week	In Progress

- **MODULES**
  Library management systems have two modules,

  1. ADMIN
  2. STUDENT

  ADMIN:
  - Admin can maintain the whole process of this application.
  - Admin can monitor all of the book detail and student detail also.
  - Admin can edit the book details. And they can search and view the student details**.**

  STUDENT:
  - Students can register themselves and after registration they will get a student ID.
  - Students can view issued books and book return date-time.
  - Students can also change their password. Students can also recover their own password.

MODULES DESCRIPTION:

ADMIN

1. Dashboard:

- Admin dashboard can view all the processes of admin.
- It will display the admin details also.

2. Add/update/ delete category:

- Admin can add/update/delete the category details.
- Category means the book will be considered as many types like mathematics, science etc**.**

3. Add/update/ delete author:

- Admin can add new author details.
- And also, the admin can update/delete the author details.

4. Add/update/ delete books:

- Admin can add new Book detail.
- Admin can also update /delete the existing book's details.

5. Update issue a new book to student:

- Admin can update the issue book details.
- And also update the details when students return the book.

6. View student details:

- Admin can view the student detail.
- Admin can search the student detail using their student id.

7. Change own password:

- Admin can change their own password.
- And also, they can change their details also.

STUDENT

1. Registration:

- Students can register their details in this portal.
- After Successful registration they can get the student id.
- Student id is used to get the books from the library, and also it is an
- identification for every student.

2. Login:

- Once you register, you can login in this portal.
- Students can login using their mail id and password.

3. View Books:

- After the successful login, students can view the books list.
- And they can view the return book detail
- And also, they can view the issue book details.

4. View profile:

- Students can view their own profile.
- Students can also change their own password.

6. Feasibility Study

Before initiating full-scale development of the Online Library Management System (OLMS), a detailed feasibility study was conducted to ensure the practicality, viability, and success potential of the project. This study evaluates whether the project is realistic in terms of available resources, time constraints, technical infrastructure, and organizational needs.

🔧 1. Technical Feasibility

This refers to the availability and suitability of the technology, tools, and resources required for the development and implementation of the system.

- The project is built using **Python (Django framework)** — a robust, secure, and scalable web development framework with strong community support.
- **SQLite** is used as the database, which is lightweight and well-integrated with Django for development and testing.
- Basic **HTML, CSS, and Bootstrap** are used for frontend design — all of which are easy to learn and implement.
- The system can be deployed on local servers or cloud platforms with minimal configuration.
- No high-end hardware or specialized software is required.

✅ **Conclusion**: The system is **technically feasible** using readily available tools and skills.

💰 2. Economic Feasibility

This evaluates whether the project can be developed and run within the available budget and whether the long-term benefits outweigh the costs.

- Development was done using free and open-source tools (Django, SQLite, Python, Bootstrap).
- There were no licensing costs, making it ideal for educational institutions or small libraries with limited budgets.
- Maintenance and future scaling can also be managed at a low cost due to the open-source nature of the stack.
- Hosting on free-tier platforms (e.g., Heroku, Vencel) can reduce infrastructure costs significantly if deployed online.

✅ **Conclusion: The project is economically feasible, especially for academic or non-profit library settings.**

👨‍💼 3. Operational Feasibility

This checks whether the system can function effectively in the real-world environment and be adopted by its intended users.

- The system is designed with role-based access for Admin, Teacher, and Student — each tailored to user needs.
- The UI is simple and intuitive, requiring no technical knowledge for basic operations.
- All core library operations — book management, issuance, return, notifications, and leave/feedback — are automated and accessible through a web browser.
- Staff and students can adapt to this system without altering their workflows drastically.

✅ Conclusion: The system is operationally feasible and user-ready.

📜 4. Legal Feasibility

- The project uses only open-source technologies, which are legally allowed for both development and deployment.
- It does not collect or expose any sensitive personal information that could violate data protection laws.
- Any future deployment can comply with data privacy regulations with simple SSL and user access controls.

✅ Conclusion: There are no legal barriers to development or deployment.

⏳ 5. Schedule Feasibility

- The entire project was planned and developed within a 6–8-week academic window, following the waterfall model.
- Each phase (Requirement gathering, Design, Implementation, Testing) was strictly time-bound and tracked.
- The project met all internal deadlines and allowed time for final testing, documentation, and presentation.

✅ Conclusion: The project is feasible within the given academic timeframe.

7. Functional Requirements

Functional requirements describe the **core features and operations** that the system must support to meet its intended purpose. These are based on user roles (Admin, Teacher, and Student) and the specific actions each can perform within the Online Library Management System.

✅ User Roles & Functionalities:

👤 Admin Functions-

1. **Admin Login**
   - Secure login with credentials to access the admin panel.

2. **Manage Books**
   - Add, update, delete book details.
   - Track issued and returned books.
   - Maintain book inventory (stock available vs. issued).

3. **Manage Users**
   - Add and manage teacher and student accounts.
   - Reset passwords or deactivate accounts.

4. **Issue & Return Books**
   - Issue books to students and teachers.
   - Accept returned books and update inventory.

5. **View Leave Requests**
   - View and process leave applications from students/teachers.

Teacher Functions-

1. **Teacher Login**
    - Access dashboard using valid credentials.

2. **View Books**
    - Search, filter, and view book listings.

3. **Request Book Issue/Return**
    - Send a request to the admin for issuing or returning books.

4. **Submit Feedback**
    - Provide feedback about the library or system.

5. **Apply for Leave**
    - Submit leave applications which are reviewed by the admin.

Student Functions

1. **Student Login**
    - Secure login for students to access their panel.

2. **Search & View Books**
    - View all available books with filters like subject, author, etc.

3. **Request Issue/Return**
    - Request books for borrowing or mark them for return.
4. **Give Feedback**
    - Submit suggestions or feedback about the system or library.
5. **Leave Application**
    - Apply for leaves (if integrated with attendance/log system).

📦 System Functionalities (General)

- Role-based access control (admin, teacher, student).
- Dashboard overview of book inventory and issued books.
- Notifications for due dates or book return.
- Automatic timestamp on book issue/return.

8. Non-Functional Requirements

Non-functional requirements define quality attributes of the system. These ensure the system is reliable, usable, secure, and performs well under expected loads.

⚙️ Performance Requirements

- System should load all pages within 3 seconds under normal usage.
- Should handle simultaneous login by multiple users without performance drop.

🔐 Security Requirements

- Secure login with encrypted password handling (Django default hashing).
- Admin-exclusive access to sensitive functions (add/delete books, manage users).
- Protection against SQL Injection and XSS (handled by Django ORM and templates).
- Sessions expire after inactivity for security.

🌐 Availability

- The system should be accessible 24/7 when deployed on a hosting service.
- Local version works without the internet after setup**.**

🎨 Usability

- Simple, clean UI with Bootstrap to ensure easy navigation.
- Users should be able to understand and use the system with minimal training.

🔄 Scalability

- Database and Django framework allow for scaling to thousands of users and books with minor upgrades.
- Can easily add new features like fines, QR scanning, or mobile support.

🛠️ Maintainability

The system has been designed following Django's Model-View-Template (MVT) architecture, which promotes separation of concerns and makes the application easier to maintain and debug.

- Modular code ensures that updates or bug fixes can be performed on specific components without affecting the entire system.
- Use of consistent naming conventions, comments, and documentation makes the system understandable for future developers or maintainers.

Additionally, database migrations and model changes are handled using Django's built-in migration framework, ensuring seamless upgrades and schema evolution.

🌍 Portability

The system is platform-independent and can run on any operating system that supports Python and MySQL.

- Whether deployed on Windows, Linux, or macOS, the system remains fully functional with minimal configuration changes.
- This ensures that educational institutions with different infrastructure setups can easily adopt the software without compatibility issues.

Virtual environments and package managers like pip further improve portability by ensuring all dependencies are consistently managed.

✅ Reliability

The Online Library Management System is built with reliability in mind to ensure that the system behaves as expected under normal and abnormal conditions.

- Critical operations such as issuing and returning books are validated through multiple layers including form validation, database constraints, and backend logic.
- Django's built-in exception handling and form validation reduce the chances of system crashes or corrupt data.
- In the event of an error, user-friendly error messages are displayed without exposing technical details.

🧩 Extensibility

The system has been structured in a way that new modules and features can be added with minimal disruption.

- Features like fine automation, QR code book scanning, SMS/email alerts, and API-based integrations can be introduced in future versions.
- The use of object-oriented programming allows for easy extension of classes and models without rewriting existing code.
- The admin panel is also easily customizable for future staff roles or access permissions.

This extensibility ensures the longevity and scalability of the system for evolving library needs.

🛟 Supportability

The use of popular, community-supported technologies such as Django, Python, and MySQL guarantees a high degree of supportability.

- Developers can easily find documentation, tutorials, and community forums for troubleshooting and improvements.

- The system follows standard web development practices, making it easier for other developers or institutions to adopt, support, and customize the application.

Furthermore, Django's compatibility with third-party packages and plugins ensures that the system can be integrated with modern tools such as cloud storage, analytics dashboards, or even mobile apps.

📈 Compliance and Best Practices

- The application follows general web development best practices such as session handling, CSRF protection, and input sanitization.

- Django's built-in protections against **SQL Injection**, **Cross-site Scripting (XSS)**, and **Cross-site Request Forgery (CSRF)** add another layer of compliance with standard web application security guidelines.

- Passwords are never stored in plain text; Django uses robust hashing algorithms such as PBKDF2 by default.

9. Software Requirements

This section lists the software components used in the development and deployment of the **Online Library Management System**. The stack is chosen to ensure reliability, scalability, and ease of development.

System Software:

Category	Requirement
Operating System	Windows 10/11, Linux, or macOS
Python Version	Python 3.13
Web Browser	Chrome, Firefox, Edge

Development Tools & Frameworks:

Tool/Software	Purpose
Python 3.13	Primary backend programming language
Django Framework	Web development framework (MTV architecture)
MySQL	Relational database for data persistence
HTML5	Markup language for structuring web content
CSS3	Styling and layout of frontend UI
Bootstrap	Responsive frontend design framework
JavaScript	Adds interactivity and dynamic behavior to UI
PyCharm	Code editor/IDE for development.
Git	Version control for collaborative development

📦 Python Libraries / Packages

Installed via pip:

- Django – Web framework
- MySQL client or MySQL – MySQL database connectors
- Django-crispy-forms – For enhanced form rendering (optional)

🛠️ Frontend Assets

The project utilizes static assets for better UI/UX:

- **CSS Libraries**: Bootstrap, Code mirror stylesheets
- **JavaScript Libraries**: Bootstrap Table, Alerts, Ace editor, background popups

These are located in the static/assets/ directories and are integrated into Django templates.

💡 Why These Tools?

- **Python 3.13**: Modern, efficient, and supported for long-term use.
- **Django**: Secure, batteries-included framework that simplifies development.
- **MySQL**: Robust and scalable RDBMS suitable for handling structured data.
- **HTML/CSS/JS**: Standard technologies for building and interacting with web frontends.
- **Bootstrap**: Allows for quick responsive layouts and styling without writing custom CSS.
- **PyCharm**: Lightweight editor with extensive plugin support for Django and Git.

9.1 SOFTWARE DESCRIPTION

INTRODUCTION TO PYTHON:

Python is a powerful, high-level, interpreted programming language known for its simple syntax and wide applicability across domains such as web development, data science, automation, and artificial intelligence. It was developed by Guido van Rossum and released in 1991. Python emphasizes code readability and supports multiple programming paradigms including procedural, object-oriented, and functional programming.

Python has become one of the most popular languages for web development due to its vast ecosystem of frameworks and libraries. In this project, Python version 3.13 was used, providing robust features like built-in data types, exception handling, and integration with web frameworks like Django.

Key features of Python include:

- Interpreted Language: Python code runs line by line, making debugging and development faster.
- Dynamic Typing: Variable types are determined at runtime.
- Extensive Standard Library: Includes modules for file handling, databases, regular expressions, and more.
- Cross-platform: Python code can run on Windows, Linux, and macOS without modification.

INTRODUCTION TO DJANGO:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Created to ease the development of complex, database-driven websites, Django takes care of much of the hassle of web development so that developers can focus on writing their apps without needing to reinvent the wheel.

Django is:

- Secure: Helps developers avoid common security mistakes.
- Scalable: Used in high-traffic applications.
- Flexible: Integrates easily with third-party APIs, databases, and user authentication systems.
- Component-Based: Follows the Model-View-Template (MVT) architectural pattern.

In this project, Django was used to:

- Design the database schema using Django ORM.
- Handle routing and URL configurations.
- Create dynamic views for both admin and student portals.
- Manage user authentication and access control.

INTRODUCTION TO MYSQL AND MYSQL WORKBENCH:

MySQL is a popular open-source relational database management system (RDBMS). It is known for its reliability, ease of use, and ability to handle large-scale data efficiently. MySQL is widely used in web applications due to its compatibility with multiple languages and platforms.

MySQL Workbench is a visual tool for database architects and developers. It allows designing, modeling, generating, and managing databases using a graphical user interface.

Advantages of MySQL:

- **Open Source and Free**: Under the GPL license.
- **High Performance**: Capable of handling millions of records.
- **Secure**: Offers strong data protection and access control.
- **Scalable**: Suitable for small to enterprise-level applications.

In this project, MySQL was used to store:

- User data (students/admins)
- Book records
- Author and category data
- Issued and returned book transactions

INTRODUCTION TO PYCHARM IDE:

PyCharm is a powerful integrated development environment (IDE) specifically built for Python and Django development. Developed by JetBrains, PyCharm provides code analysis, graphical debugging, unit testing, and support for Django-specific features.

Key features of PyCharm used in this project:

- **Intelligent Code Editor**: Auto-completion, syntax highlighting, and code suggestions.
- **Integrated Terminal and Git Support**: For version control and CLI operations.
- **Database Tools**: Ability to connect and query the MySQL database directly from the IDE.
- **Django Support**: Templates, models, and migrations are handled efficiently.

PyCharm significantly improved productivity by simplifying debugging, testing, and managing files across the entire Django project structure.

10. Hardware Requirements

To successfully develop, test, and run the Online Library Management System, the following hardware is required. These specifications are suitable for both development and deployment on a personal computer or a local server.

Minimum Requirements:

- Processor: Dual Core 2.0 GHz or higher
- RAM: 4 GB
- Hard Disk: 10 GB free space
- Monitor: 15-inch color monitor
- Keyboard: Standard keyboard
- Mouse: Optical mouse
- Internet Connection: Required for accessing web pages and installing dependencies.

Recommended Requirements:

- Processor: Intel i5 or equivalent
- RAM: 8 GB or more
- Hard Disk: 20 GB or more (SSD preferred)
- Display: Full HD resolution for better development experience
- Power Backup: UPS or inverter (optional but useful).

11. Data Flow Diagram

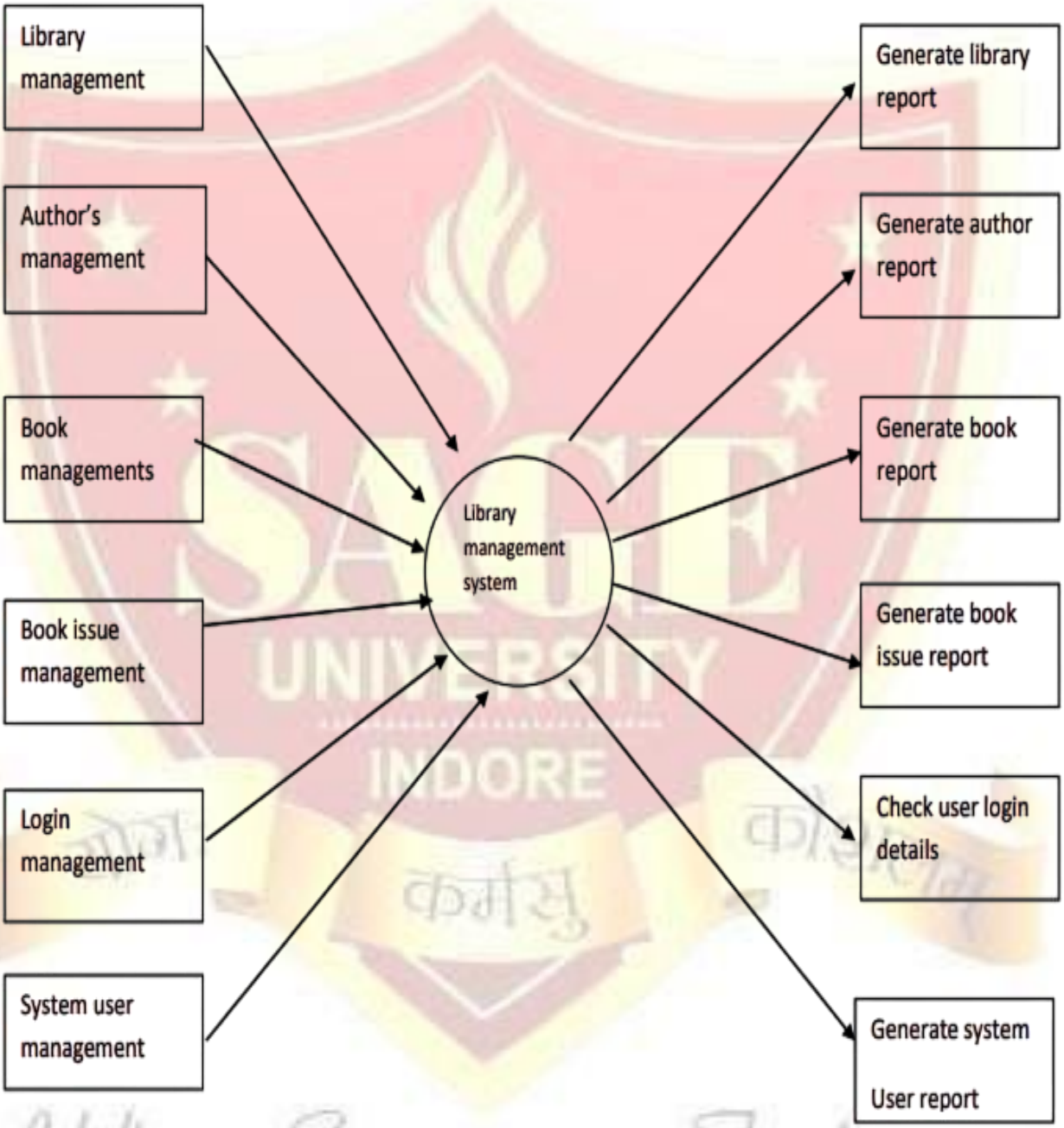

Fig 1: DFD Diagram

11.1 Introduction to DFD

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It visually illustrates how data is processed, stored, and transferred between components in the system. DFDs are essential for understanding the system architecture and breaking down its functionalities into manageable processes.

The DFD for the **Online Library Management System** captures the core processes like user login, book management, student registration, book issuance, and return. It shows how data moves between users (students/admin), the system, and the database.

11.2 Levels of DFD

a. Level 0 – Context Level DFD:
This is the highest abstraction of the system. It shows the entire system as a single process and highlights its interactions with external entities:

- **Admin** (manages books, students)
- **Student** (search, request, return books)
- **Database** (stores all records)

b. Level 1 – Detailed DFD:
Breaks the main system down into sub-processes:

1. **Login Module** – authenticates admin or student
2. **Book Management** – add/edit/delete books and categories
3. **Student Management** – register and manage student profiles
4. **Issue/Return Book** – handles lending and tracking of books
5. **Fine Management** – calculates and stores fine details

11.3 Description of Key DFD Components

- **Processes** (circles): Represent actions like “Issue Book” or “Register Student”
- **Data Stores** (open rectangles): Represent stored data like Student Records, Book List, or Issue Logs
- **Data Flows** (arrows): Represent how data moves between processes, users, and data stores
- **External Entities** (squares): Represent users or systems interacting with the application (e.g., Admin, Student)

11.4 Summary

The DFD is essential to understanding how the Online Library Management System handles key operations. It provides a clear, high-level view of:

- User interactions
- Backend processing
- Database flow

This structured representation helps developers and stakeholders identify data dependencies, validate requirements, and ensure all necessary functionalities are accounted for.

Flow Diagram

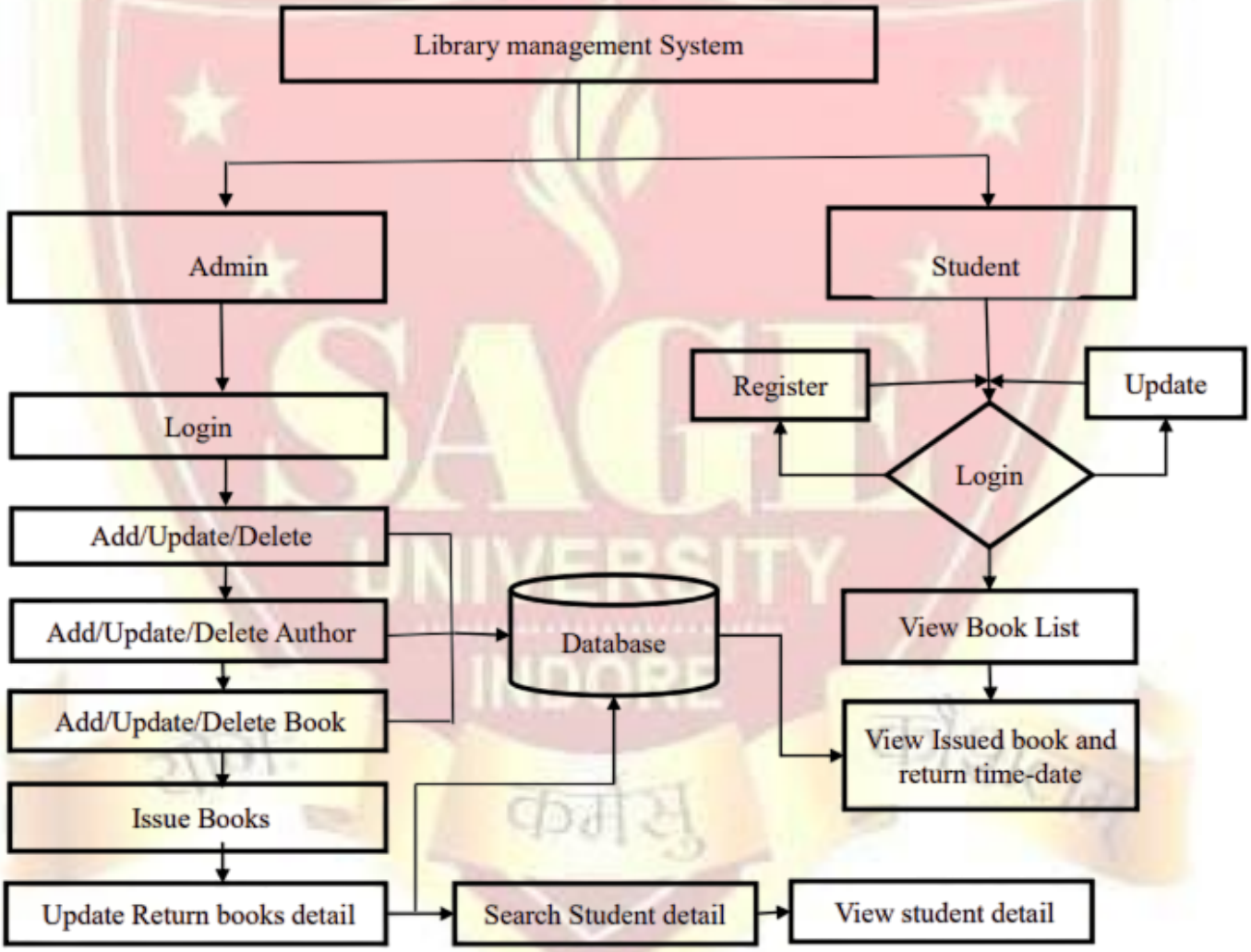

Fig 2: Flow Diagram

12. ER Diagram

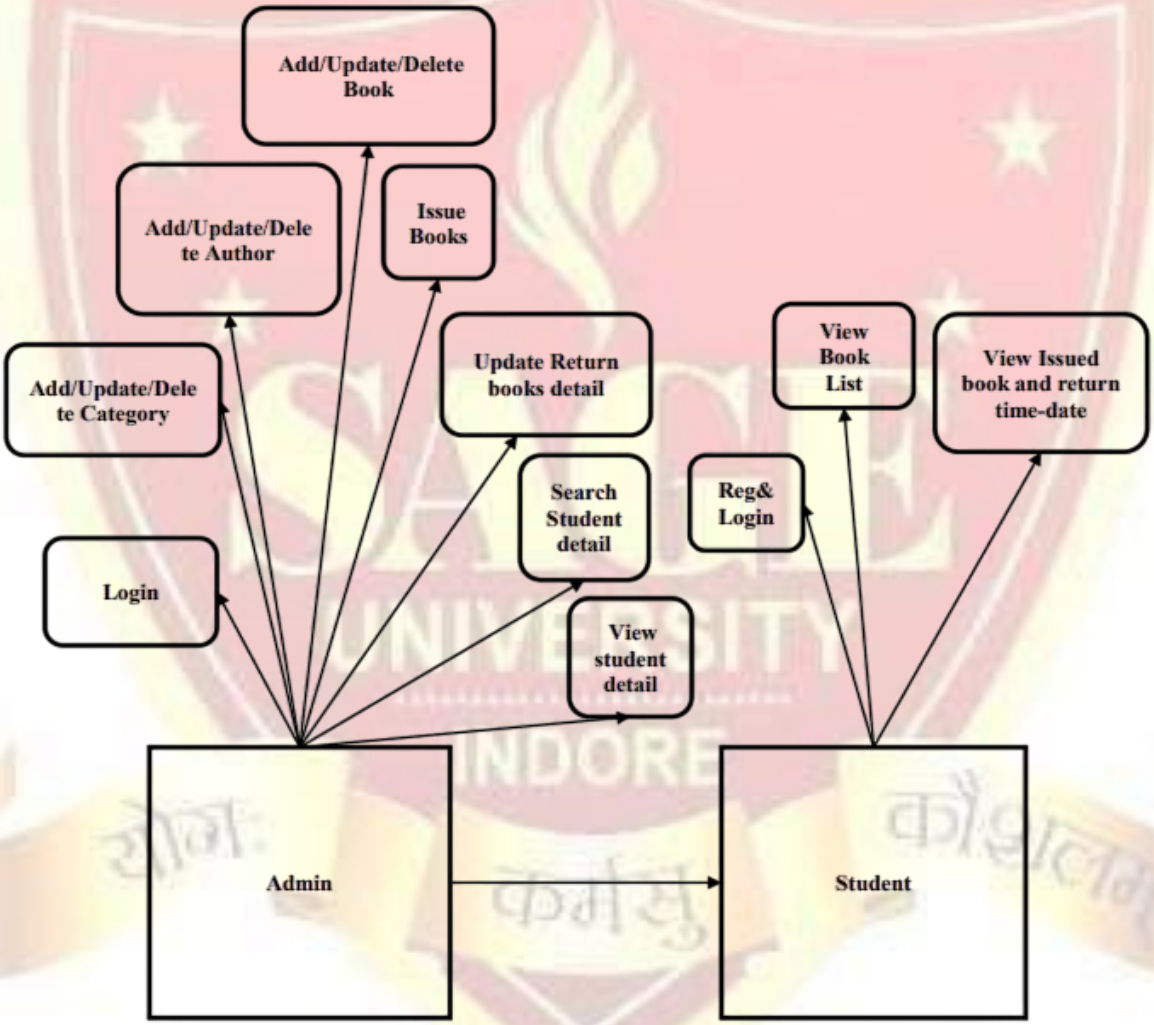

Fig 3: ER Diagram

12.1 Introduction to ER Diagram

An **Entity-Relationship (ER) Diagram** is a visual representation of the entities in a database and the relationships between them. It is used during database design to ensure logical data organization and integrity. For the **Online Library Management System**, the ER diagram maps out the core entities such as students, books, authors, and the transactions related to issuing and returning books.

12.2 Key Entities and Attributes

1. **CustomUser**
    - user_id (PK)
    - username
    - password
    - email
    - user_type (admin/student)
    - profile_pic

2. **Student**
    - student_id (PK)
    - admin_id (FK to CustomUser)
    - mobile_number
    - registration_date

3. **Book**
    - book_id (PK)
    - book_name
    - ISBN number
    - price
    - category_id (FK)
    - author_id (FK)
    - isIssued
    - book_image

4. **Category**
    - category_id (PK)
    - category_name

5. **Author**
    - author_id (PK)
    - author_name
6. **IssuedBookDetails**
    - issue_id (PK)
    - book_id (FK)
    - student_id (FK)
    - issue_date
    - return_date
    - return_status
    - fine

12.3 Relationships Between Entities

- One **CustomUser** (as student) has **one Student profile** (1:1)
- One **Category** can be assigned to **many Books** (1:N)
- One **Author** can write **many Books** (1:N)
- One **Book** can be **issued to many Students** over time (M:N) — handled by IssuedBookDetails
- One **Student** can have **many issued books** (1:N via IssuedBookDetails)

12.4 Summary

The ER diagram defines the structural design of the library system's database. It shows how users, books, and related data are organized and interrelated. This ensures:

- Accurate record-keeping
- Referential integrity via foreign keys
- Efficient query design

This ER model supports the fundamental operations of the Online Library Management System such as issuing books, managing users, and tracking returns and fines.

13. Use Case Diagram

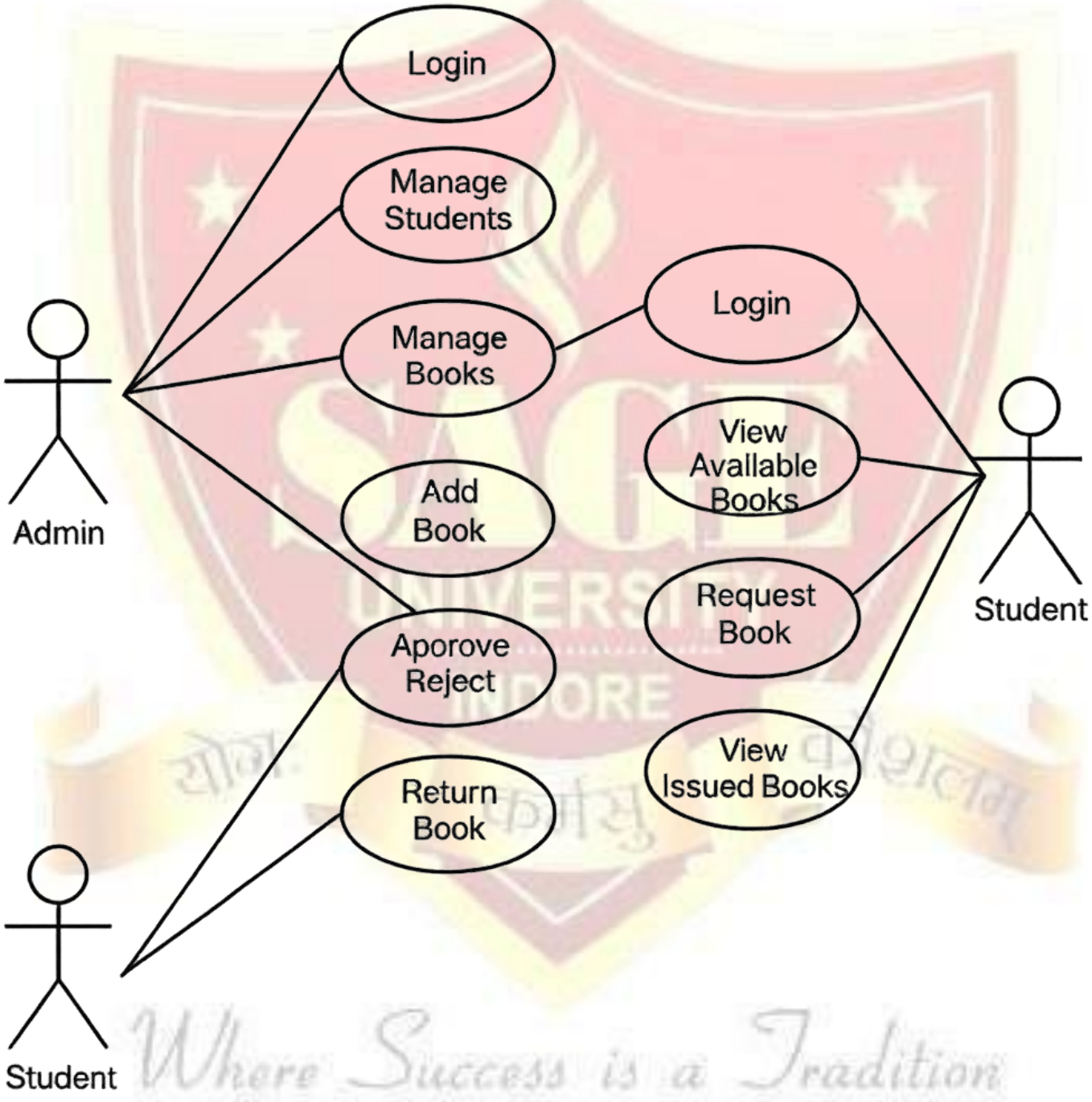

Fig 4: Use Case Diagram

13.1 Introduction to Use Case Diagram

A **Use Case Diagram** is a visual representation of the interactions between users (actors) and the system. It outlines what functionalities are available to different users and how they interact with those functionalities. This is crucial in understanding user behavior and identifying system requirements from a user-centric perspective.

For the **Online Library Management System**, the two main actors are:

- **Admin**
- **Student**

13.2 Main Use Cases

A. Admin Use Cases

- Login
- Manage Books (Add/Edit/Delete)
- Manage Categories
- Manage Authors
- Register Students
- View Student Profiles
- Issue Books
- Accept Returned Books
- Calculate Fines
- View Issued Book Records

B. Student Use Cases

- Login
- View Available Books
- Request to Issue Book
- View Issued Books
- Return Book
- View Fine

13.3 Actors

- **Admin**: Has full control over the system. Responsible for managing content, issuing books, and handling returns/fines.
- **Student**: Can view books, request/return books, and monitor their issued records.

13.4 Relationships

- **Association**: Direct connection between actor and the use case.
- **Include**: Some use cases depend on others (e.g., “Issue Book” includes “Check Book Availability”).
- **Extend**: Optional flows (e.g., “Calculate Fine” may extend “Return Book”).

13.5 Summary

The use case diagram clearly separates admin and student responsibilities. It ensures that all user interactions are captured for development, testing, and documentation. This diagram is useful in planning UI features, access permissions, and future enhancements.

14. Class Diagram

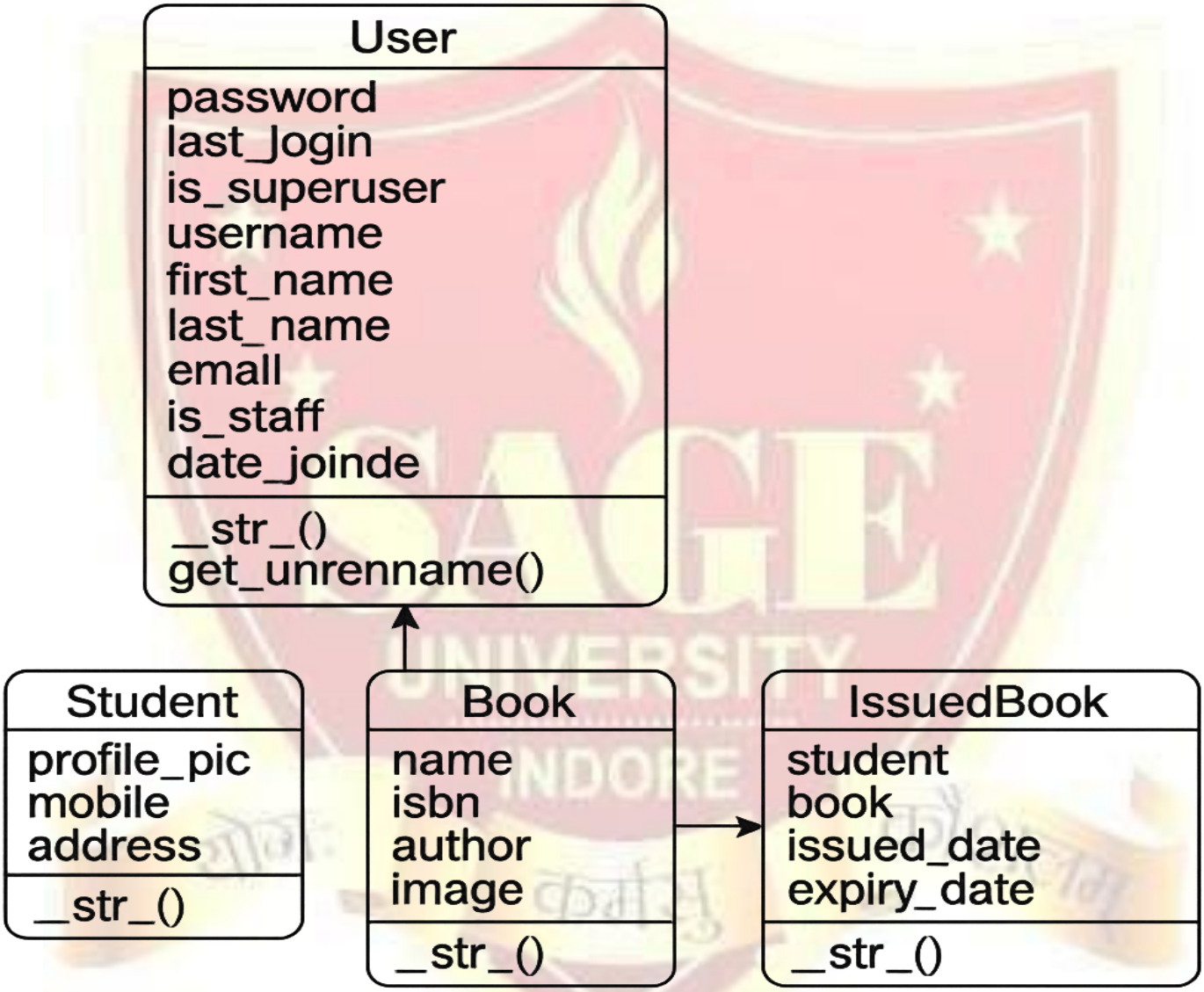

Fig 5: Class Diagram

14.1 Introduction to Class Diagram

A **Class Diagram** is a type of static structure diagram in Unified Modeling Language (UML) that describes the structure of a system by showing its classes, attributes, methods, and the relationships among objects.

In the **Online Library Management System**, the class diagram illustrates how the core components of the system—such as users, students, books, and issued book details—are modeled and related to each other in the object-oriented design.

14.2 Core Classes and Their Responsibilities

1. CustomUser

- Inherits from Django's AbstractUser
- Attributes: username, password, user_type (admin/student), profile_pic
- Purpose: Handles authentication and role-based access

2. Student

- Attributes: mobile_number, student_id, reg_date
- Relationships: One-to-one with CustomUser
- Purpose: Holds student-specific data and links to user account

3. Book

- Attributes: book_name, ISBN, price, image, isIssued
- Relationships: Many-to-one with Author and Category
- Purpose: Represents physical or digital books in the library

4. Author

- Attributes: author_name
- Purpose: Stores information about authors

5. Category

- Attributes: category_name
- Purpose: Groups books under genres or topics

.

6. IssuedBookDetails

- Attributes: issue_date, return_date, return_status, fine
- Relationships: Foreign keys to Book and Student
- Purpose: Tracks which book was issued to which student, with timestamps and penalties

14.3 Relationships

- **One-to-One**: CustomUser ↔ Student
- **One-to-Many**:
    - Category → Book
    - Author → Book
    - Student → IssuedBookDetails
    - Book → IssuedBookDetails
- **Inheritance**: Django handles CustomUser via AbstractUser

14.4 Summary

The class diagram provides a blueprint of how data is structured and connected in the system. It helps in:

- Designing the database schema
- Understanding object relationships
- Implementing features in Django models
- Planning class-based views or serializers (if using Django REST)

This diagram ensures the system is scalable, logically structured, and easy to maintain.

Architecture Diagram

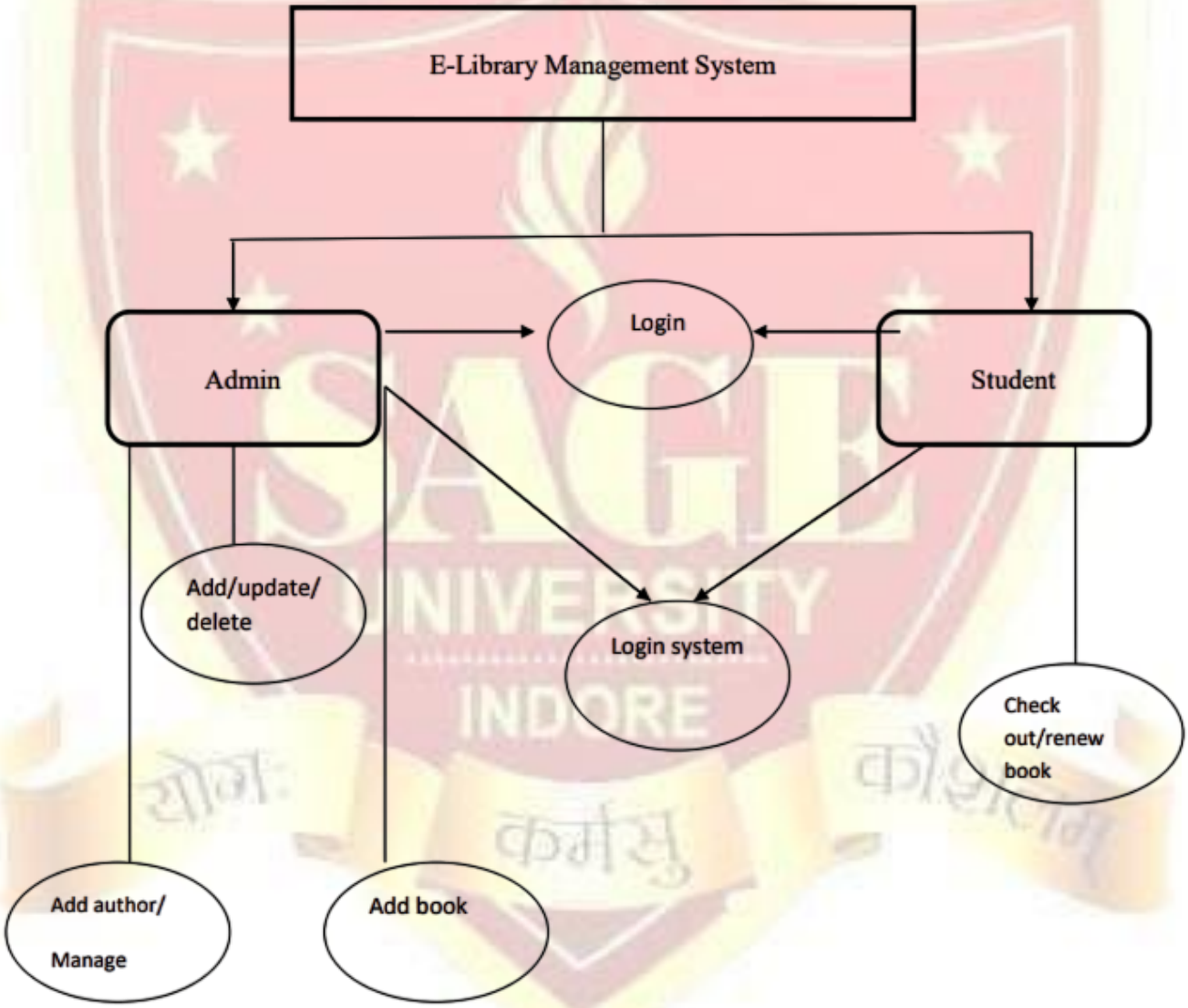

Fig 6: Architecture Diagram

15. Database Tables

1. Database Architecture Overview

The OLMS database is built on MySQL (MariaDB 10.4.32), which provides robust support for relational data management. The database schema consists of several interconnected tables that represent the core entities of the library management system. These tables are designed with appropriate relationships to ensure data consistency and support the various functionalities of the system.

2. Core Database Components

2.1 User Management

The database implements a custom user authentication system with role-based access control. The olmsapp_customuser table extends Django's built-in user model to accommodate different user types (administrators and students) with additional fields for profile management.

2.2 Library Resources

Books, authors, and categories form the foundation of the library resources. The normalized structure ensures that:

- Each book belongs to a specific category
- Each book is associated with a specific author
- Books can be tracked for availability status

2.3 Student Management

Student records are maintained separately from user accounts but are linked via foreign keys. This approach allows for detailed student information management while maintaining connection to authentication credentials.

2.4 Transaction Management

The system tracks all book issuance and return operations, including issue dates, return status, and any applicable fines. This enables comprehensive reporting and monitoring of library circulation.

3. Database Design Principles Applied

The database design follows these key principles:

1. **Normalization**: Tables are normalized to minimize redundancy and maintain data integrity
2. **Referential Integrity**: Foreign key constraints ensure data consistency across related tables
3. **Appropriate Indexing**: Primary and foreign keys are indexed for optimized query performance
4. **Audit Trails**: Timestamp fields track creation and modification dates of important records
5. **Data Validation**: Constraints such as NOT NULL and UNIQUE ensure data quality.

4. Database Schema Structure

The following section presents the detailed structure of each table in the database. For each table, we provide:

- Table name and description
- Column definitions including data types and constraints
- Primary and foreign key relationships
- Indexes and other relevant constraints

This documentation serves as a comprehensive reference for understanding the data model that powers the Online Library Management System.

Database Tables

1. olmsapp_book Table:

Table: olmsapp_book

Create Table: CREATE TABLE `olmsapp_book` (
`id` bigint NOT NULL AUTO_INCREMENT,
`bookname` varchar(200) NOT NULL,
`isbnnum` varchar(200) NOT NULL,
`price` varchar(200) NOT NULL,
`bookimage` varchar(100) DEFAULT NULL,
`created_at` datetime(6) NOT NULL,
`updated_at` datetime(6) NOT NULL,
`isIssued` varchar(50) NOT NULL,
`authid_id` bigint NOT NULL,
`catid_id` bigint NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `isbnnum` (`isbnnum`),
KEY `olmsapp_book_authid_id_92576cf6_fk_olmsapp_author_id` (`authid_id`),
KEY `olmsapp_book_catid_id_7aa39b2d_fk_olmsapp_category_id` (`catid_id`),
CONSTRAINT `olmsapp_book_authid_id_92576cf6_fk_olmsapp_author_id` FOREIGN KEY (`authid_id`) REFERENCES `olmsapp_author` (`id`),
CONSTRAINT `olmsapp_book_catid_id_7aa39b2d_fk_olmsapp_category_id` FOREIGN KEY (`catid_id`) REFERENCES `olmsapp_category` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

mysql> DESCRIBE olmsapp_book;

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
bookname	varchar(200)	NO		NULL	
isbnnum	varchar(200)	NO	UNI	NULL	
price	varchar(200)	NO		NULL	
bookimage	varchar(100)	YES		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
isIssued	varchar(50)	NO		NULL	
authid_id	bigint	NO	MUL	NULL	
catid_id	bigint	NO	MUL	NULL	

2. olmsapp_author Table:

```
SHOW CREATE TABLE olmsapp_author\G;
*************************** 1. row ***************************
       Table: olmsapp_author
Create Table: CREATE TABLE `olmsapp_author` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `authorname` varchar(200) NOT NULL,
  `created_at` datetime(6) NOT NULL,
  `updated_at` datetime(6) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

mysql> DESCRIBE olmsapp_author;

```
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| authorname | varchar(200) | NO   |     | NULL    |                |
| created_at | datetime(6)  | NO   |     | NULL    |                |
| updated_at | datetime(6)  | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

3. olmsapp_category Table:

```
Table: olmsapp_category
Create Table: CREATE TABLE `olmsapp_category` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `catname` varchar(200) NOT NULL,
  `status` varchar(200) NOT NULL,
  `created_at` datetime(6) NOT NULL,
  `updated_at` datetime(6) NOT NULL,
  PRIMARY KEY (`id`)
)
```

mysql> DESCRIBE olmsapp_category;

```
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| catname    | varchar(200) | NO   |     | NULL    |                |
| status     | varchar(200) | NO   |     | NULL    |                |
| created_at | datetime(6)  | NO   |     | NULL    |                |
| updated_at | datetime(6)  | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

5 rows in set (0.00 sec)

4. olmsapp_customuser Table:

Table: olmsapp_customuser
Create Table: CREATE TABLE `olmsapp_customuser` (

```
  `id` bigint NOT NULL AUTO_INCREMENT,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(150) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL,
  `user_type` int NOT NULL,
  `profile_pic` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

mysql> DESCRIBE olmsapp_customuser;

```
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| password     | varchar(128) | NO   |     | NULL    |                |
| last_login   | datetime(6)  | YES  |     | NULL    |                |
| is_superuser | tinyint(1)   | NO   |     | NULL    |                |
| username     | varchar(150) | NO   | UNI | NULL    |                |
| first_name   | varchar(150) | NO   |     | NULL    |                |
| last_name    | varchar(150) | NO   |     | NULL    |                |
| email        | varchar(254) | NO   |     | NULL    |                |
| is_staff     | tinyint(1)   | NO   |     | NULL    |                |
| is_active    | tinyint(1)   | NO   |     | NULL    |                |
| date_joined  | datetime(6)  | NO   |     | NULL    |                |
| user_type    | int          | NO   |     | NULL    |                |
| profile_pic  | varchar(100) | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

13 rows in set (0.00 sec)

5. olmsapp_student Table:

Table: olmsapp_student
Create Table: CREATE TABLE `olmsapp_student` (
`id` bigint NOT NULL AUTO_INCREMENT,
`mobilenumber` varchar(11) NOT NULL,
`studentid` varchar(50) NOT NULL,
`regdate_at` datetime(6) NOT NULL,
`updated_at` datetime(6) NOT NULL,
`admin_id` bigint NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `studentid` (`studentid`),
UNIQUE KEY `admin_id` (`admin_id`),
CONSTRAINT `olmsapp_student_admin_id_2000799a_fk_olmsapp_customuser_id` FOREIGN KEY (`admin_id`) REFERENCES `olmsapp_customuser` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)

mysql> DESCRIBE olmsapp_student;

```
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id           | bigint      | NO   | PRI | NULL    | auto_increment |
| mobilenumber | varchar(11) | NO   |     | NULL    |                |
| studentid    | varchar(50) | NO   | UNI | NULL    |                |
| regdate_at   | datetime(6) | NO   |     | NULL    |                |
| updated_at   | datetime(6) | NO   |     | NULL    |                |
| admin_id     | bigint      | NO   | UNI | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

6 rows in set (0.00 sec)

Data Table

mysql> SELECT * FROM olmsapp_student;

```
+----+--------------+-----------+----------------------------+----------------------------------------------+----------+
| id | mobilenumber | studentid | regdate_at                 | updated_at                                   |admin_id |
+----+--------------+-----------+----------------------------+----------------------------------------------+----------+
|  3 | 0000000000   | SS6435    | 2025-05-19 16:45:44.572019 | 2025-05-19 16:45:44.572040    | 4          |
+----+--------------+-----------+----------------------------+----------------------------------------------+----------+
```

6. olmsapp_issuebookdetails Table:

Table: olmsapp_issuedbookdetails

Create Table: CREATE TABLE `olmsapp_issuedbookdetails` (

```
  `id` bigint NOT NULL AUTO_INCREMENT,
  `issued_date` datetime(6) NOT NULL,
  `return_date` datetime(6) NOT NULL,
  `return_status` varchar(50) NOT NULL,
  `fine` decimal(10,2) NOT NULL,
  `book_id_id` bigint NOT NULL,
  `stud_id_id` bigint NOT NULL,
  PRIMARY KEY (`id`),
  KEY `olmsapp_issuedbookdetails_book_id_id_756580fa_fk_olmsapp_book_id` (`book_id_id`),
  KEY `olmsapp_issuedbookde_stud_id_id_ea19652e_fk_olmsapp_s` (`stud_id_id`),
  CONSTRAINT `olmsapp_issuedbookde_stud_id_id_ea19652e_fk_olmsapp_s` FOREIGN KEY
(`stud_id_id`) REFERENCES `olmsapp_student` (`id`),
  CONSTRAINT `olmsapp_issuedbookdetails_book_id_id_756580fa_fk_olmsapp_book_id` FOREIGN
KEY (`book_id_id`) REFERENCES `olmsapp_book` (`id`)
)
```

mysql> DESCRIBE olmsapp_issuedbookdetails;

```
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| issued_date  | datetime(6)  | NO   |     | NULL    |                |
| return_date  | datetime(6)  | NO   |     | NULL    |                |
| return_status| varchar(50)  | NO   |     | NULL    |                |
| fine         | decimal(10,2)| NO   |     | NULL    |                |
| book_id_id   | bigint       | NO   | MUL | NULL    |                |
| stud_id_id   | bigint       | NO   | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

7 rows in set (0.00 sec)

7. auth_group Table:

mysql> SELECT * FROM table_name;

ERROR 1146 (42S02): Table 'olmspythondb.table_name' doesn't exist

mysql> DESCRIBE auth_group;

```
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int         | NO   | PRI | NULL    | auto_increment |
| name  | varchar(150)| NO   | UNI | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
```

Table: auth_group

Create Table: CREATE TABLE `auth_group` (

```
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(150) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

1 row in set (0.00 sec)

8. **Tables in Database:**

mysql> SHOW TABLES;

```
+-----------------------------------+
| Tables_in_olmspythondb            |
+-----------------------------------+
| auth_group                        |
| auth_group_permissions            |
| auth_permission                   |
| django_admin_log                  |
| django_content_type               |
| django_migrations                 |
| django_session                    |
| olmsapp_author                    |
| olmsapp_book                      |
| olmsapp_category                  |
| olmsapp_customuser                |
| olmsapp_customuser_groups         |
| olmsapp_customuser_user_permissions |
| olmsapp_issuedbookdetails         |
| olmsapp_student                   |
+-----------------------------------+
```

16.Testing

Testing is a crucial phase in the software development life cycle that ensures the functionality, reliability, and performance of the system. For the Online Library Management System, various testing strategies were applied to validate that the system meets its requirements and functions correctly under different scenarios.

16.1 Types of Testing Performed

a. Unit Testing

Each model and view in the Django backend was tested independently to ensure they work as expected. Functions like __str__(), form validations, and model relationships were verified.

b. Integration Testing

After individual modules passed unit testing, their interactions were tested. Examples:

- Adding a book and assigning it to a category and author.
- Registering a student and issuing a book to them.

c. System Testing

The system was tested as a whole to confirm it behaves correctly in a production-like environment. This includes the complete user flow:

- Admin login
- Add books, authors, and categories
- Student registration
- Book issue and return processes

d. User Acceptance Testing (UAT)

The system was deployed and tested by potential users (students and librarians) to ensure it meets user needs. Feedback was collected and used for minor improvements.

16.2 Test Cases:

Test Case ID	Description	Input	Expected Output	Result
TC001	Admin Login	Valid username/password	Redirect to dashboard	Pass
TC002	Add Book	Valid book data	Book saved and listed	Pass
TC003	Add Student	Valid student info	Student profile created	Pass
TC004	Issue Book	Select valid book & student	Book status changed to issued	Pass
TC005	Return Book	Click return on issued book	Book status updated, fine calculated if any	Pass
TC006	Prevent Duplicate ISBN	Duplicate ISBN	Error message shown	Pass
TC007	View Book List	N/A	Display all available books	Pass
TC008	Profile Picture Upload	Valid image file	Profile image saved and displayed	Pass

16.3 Tools Used for Testing

- **Django's built-in test framework** (python manage.py test)
- **Manual UI testing** using browser for forms, navigation, and uploads
- **MySQL queries** to validate database entries post-operations

16.4 Bug Tracking and Fixing

Any issues found during testing were logged in a simple bug tracking spreadsheet and resolved iteratively. Most bugs were related to form validation and UI alignment.

17. Limitations

While the Online Library Management System is functional and serves its core purpose, there are several limitations in its current implementation:

1. No Fine Calculation Logic Based on Due Date

The system stores fine as a field but does not automatically calculate it based on overdue days. This needs to be manually updated.

2. Limited Role-Based Access

Only two user types are implemented: admin and student. There's no librarian role or additional access control layers like staff with limited privileges.

3. No Email or SMS Notifications

The system does not send automatic email or SMS alerts for:

- Book issue confirmations
- Return reminders
- Overdue fines

4. Static Book Categories

Book categories must be created manually by the admin. There's no recommendation or dynamic suggestion engine.

5. No Search Autocomplete or Filters

Students cannot use autocomplete, filters, or advanced search while looking for books. This affects usability, especially with large book datasets.

6. File Upload Restrictions

Although the system supports profile pictures and book images, it lacks:

- File size validation
- Format restrictions
- Image compression or optimization

7. Limited Reporting Features

There are no advanced analytics or reporting tools to show:

- Most borrowed books
- Top students
- Monthly issue/return summaries

8. Not Mobile-Responsive

The current frontend design does not offer a responsive layout optimized for tablets or mobile devices, which limits accessibility.

9. No Multi-Language Support

The system is designed only in English and does not support internationalization or localization features.

10. Minimal Security Enhancements

Password reset, CAPTCHA on login forms, and audit logging are not implemented. These are important for production-level security.

18. Future Scope

The Online Library Management System has strong potential for further development and improvement. Future enhancements can make the system more efficient, user-friendly, and scalable to meet the growing needs of modern educational institutions and digital libraries.

1. Fine Automation Based on Return Date

Implement logic to automatically calculate fines based on the number of overdue days and predefined rates. This would reduce manual work and improve accuracy.

2. Integration of Email and SMS Notifications

Enable notifications for:

- Book issuance and return
- Due date reminders
- Fine alerts
- New book arrivals

This would enhance communication with users and improve timely returns.

3. Mobile App Integration

Develop an Android and iOS app version of the system to improve accessibility and user engagement, allowing students to manage books on the go.

4. QR Code for Book Scanning

Use QR codes on books for quick issue/return actions using mobile scanning. This would simplify transactions and reduce manual entry errors.

5. Role-Based Access Control (RBAC)

Add more user roles like librarian, staff, or department admin with customizable permissions. This helps in delegating tasks and improving management.

6. Advanced Search and Filters

Introduce advanced filters, category-based browsing, and search suggestions to help users quickly find relevant books.

7. Analytics and Reports Dashboard

Include a reporting module with visual charts and data on:

- Most borrowed books
- Active users
- Fine collection reports
- Book demand trends

8. Support for E-books and Digital Resources

Enable the upload and access of e-books and PDF materials, allowing the library to support digital reading habits.

9. Multi-Language and Localization Support

Add support for multiple languages to make the system usable in regional and international contexts.

10. Cloud Deployment and Scalability

Host the application on cloud services (like AWS, Azure, or Google Cloud) to support larger institutions, with better performance, backups, and scalability.

11. Integration with Learning Management Systems (LMS)

As educational institutions continue to digitize their operations, integrating the library management system with existing LMS platforms like Moodle, Google Classroom, or Blackboard can streamline academic workflows. Students could access library services directly from their academic portals, enabling seamless interaction between learning resources and library records.

This would also allow automated updates for reading lists, syllabus-related book references, and direct links to e-resources. Faculty could monitor book usage trends to evaluate the impact of recommended readings.

12. Artificial Intelligence (AI) for Smart Recommendations

Future versions of the system could leverage AI and machine learning algorithms to analyze students' borrowing history, academic discipline, and popular titles to provide intelligent book recommendations. This personalized approach would not only enhance user experience but also promote underutilized books and authors.

Furthermore, predictive analytics could be used to anticipate demand for specific books based on semester schedules, exam periods, or newly added courses.

13. Integration with National and Institutional Libraries

By integrating with national digital library networks (like the National Digital Library of India), the system can broaden its reach and give users access to millions of external academic resources. This federated library access model would be especially useful for research scholars and postgraduate students seeking peer-reviewed or rare resources beyond their local library collection.

14. Real-Time Chat Support and Helpdesk

Adding a built-in chat system or chatbot can help users navigate the system and resolve common issues (e.g., "How do I renew a book?", "Where can I find journals?"). This AI-driven virtual assistant can improve student engagement and reduce the workload on library staff.

An optional human-help escalation system can also be built-in for issues requiring manual resolution.

15. Integration of Biometric or RFID Authentication

Security and personalization can be further enhanced by using biometric login (fingerprint or face recognition) or RFID-enabled smart cards for identity verification at library kiosks. This minimizes fraudulent usage and speeds up the process of book issuing and returning.

RFID-based automation can also be applied for real-time inventory tracking, self-checkout, and loss prevention.

19. Conclusion

The Online Library Management System successfully achieves its core objective of simplifying and automating the processes involved in managing a library. By digitizing tasks such as student registration, book categorization, book issuance, and return tracking, the system enhances both efficiency and accuracy in library operations.

This project demonstrates the practical application of software development concepts using the Django framework with a MySQL database. It provides a functional admin panel for managing books, authors, and users, while also offering a student portal to view and manage issued books. The inclusion of image support, user roles, and timestamp-based records adds to the overall reliability of the system.

Despite some limitations, the project lays a strong foundation for further development and can be scaled with features like automated fine calculation, mobile app integration, and analytics dashboards.

In summary, the system serves as a robust solution for small to medium-sized libraries and showcases the successful implementation of a full-stack web application in a real-world educational context.

The development of the Online Library Management System has been a comprehensive and rewarding journey that allowed me to apply a wide range of theoretical and practical knowledge in building a real-world application. The core purpose of this project was to design a web-based system that streamlines and automates traditional library processes such as book inventory management, student registration, and book issuance and return workflows. By digitizing these operations, the system not only improves accuracy and speed but also enhances accessibility for users, particularly students and administrators.

From a technical standpoint, the system was built using the **Django framework**, one of the most powerful and scalable Python-based web development platforms. Its built-in authentication system, ORM (Object Relational Mapper), and support for modular development were instrumental in ensuring a well-structured and maintainable codebase. The backend logic was carefully mapped using Django models, which connected seamlessly to a **MySQL database**. The database design was normalized, secure, and capable of handling relationships between users, students, books, authors, and issued book records.

One of the key achievements of this project is its **role-based user system**, allowing differentiated access for admins and students. Admins are empowered to manage the entire system—from adding book categories and authors to issuing books and handling returns—while students are given access to view, request, and return books assigned to them. This division not only enhances security but also ensures a user-friendly experience for both types of users.

The **frontend** was designed using Django templates along with HTML and Bootstrap for responsive design. The interface is clean, intuitive, and purpose-driven, making it easy for users to navigate the system. Features like image upload (for profile pictures and book covers), unique ISBN checks, and automatic date tracking further increase the reliability and professionalism of the system.

Despite its strengths, the project does have a few **limitations**, such as the absence of automated fine calculation based on overdue dates and the lack of features like SMS/email alerts. However, these limitations are acknowledged and have been outlined as part of the **future scope** of development. The project has been structured in such a way that these enhancements can be added modularly, without the need for major architectural changes.

On a personal level, this project served as a **major milestone** in my learning journey. It challenged me to think critically, solve problems independently, and apply programming concepts in a real-world context. I learned how to:

- Design and normalize a relational database
- Create and manage Django models and views
- Build secure login systems with custom user roles
- Handle file uploads and media management in web apps
- Debug and test applications using Django's built-in tools
- Create UML diagrams (Class, Use Case, DFD, ER) based on live code
- Understand how frontend and backend connect through routes and templates

I also became more comfortable with tools such as **VS Code, MySQL Workbench**, and Git, which are essential in modern development environments. Testing each module, documenting the flow, and fixing bugs helped me understand the **importance of clean code, testing, and maintainability**.

In summary, the Online Library Management System not only fulfills its intended purpose of digitizing library operations but also serves as a showcase of full-stack web development. It blends theory and practice, addresses a real-world problem, and offers a scalable foundation for future innovation. This project has helped me grow technically, creatively, and professionally—and has inspired me to continue exploring more complex software systems in the future.

Sample Code and Screenshots

1. Custom User Model (User Authentication Base)

This model extends Django's default user to add roles and profile pictures:

python
CopyEdit

```
from django.contrib.auth.models import AbstractUser
from django.db import models
class CustomUser(AbstractUser):
    USER = (
        (1, 'admin'),
        (2, 'students'),
    )
    user_type = models.IntegerField(choices=USER, default=1)
    profile_pic = models.ImageField(upload_to='media/profile_pic', blank=True, null=True)
```

Purpose:
Enables role-based access control and profile image storage for both students and admins.

2. Book Model (Core Library Asset)

Defines the structure of books in the system with relationships to category and author:

python
CopyEdit

```
class Book(models.Model):
    bookname = models.CharField(max_length=200)
    catid = models.ForeignKey(Category, on_delete=models.CASCADE)
    authid = models.ForeignKey(Author, on_delete=models.CASCADE)
    isbnnum = models.CharField(max_length=200, unique=True)
    price = models.CharField(max_length=200)
    bookimage = models.ImageField(upload_to='book_img/', blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    isIssued = models.CharField(max_length=50, default=None)

    def __str__(self):
        return self.bookname
```

Purpose:
Represents each book along with its author and category and includes image support.

3. Book Issue Tracking (IssuedBookDetails)

Tracks which student borrowed which book and includes fine and return status:

python
CopyEdit

```
class Issuedbookdetails(models.Model):
    book_id = models.ForeignKey(Book, on_delete=models.CASCADE)
    stud_id = models.ForeignKey(Student, on_delete=models.CASCADE)
    issued_date = models.DateTimeField(auto_now_add=True)
    return_date = models.DateTimeField(auto_now=True)
    return_status = models.CharField(max_length=50)
    fine = models.DecimalField(max_digits=10, decimal_places=2, default=0)
```

Purpose:
Core logic to track issued books, return status, and fines – essential for library operations.

4. Student Model (Extends User Profile with Student Info)

python
CopyEdit

```
class Student(models.Model):
    admin = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
    mobilenumber = models.CharField(max_length=11, default=None, blank=True)
    studentid = models.CharField(max_length=50, unique=True)
    regdate_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.studentid
```

Purpose:
This model extends Django's user system by linking each CustomUser to a Student profile. It stores student-specific details like:

- Unique student ID
- Mobile number
- Registration and update timestamps

The OneToOneField ensures that each student has exactly one user account, making it essential for managing user-specific library records and issuing books.

Screenshots of Project Dashboard

1. Library Management System Dashboard:

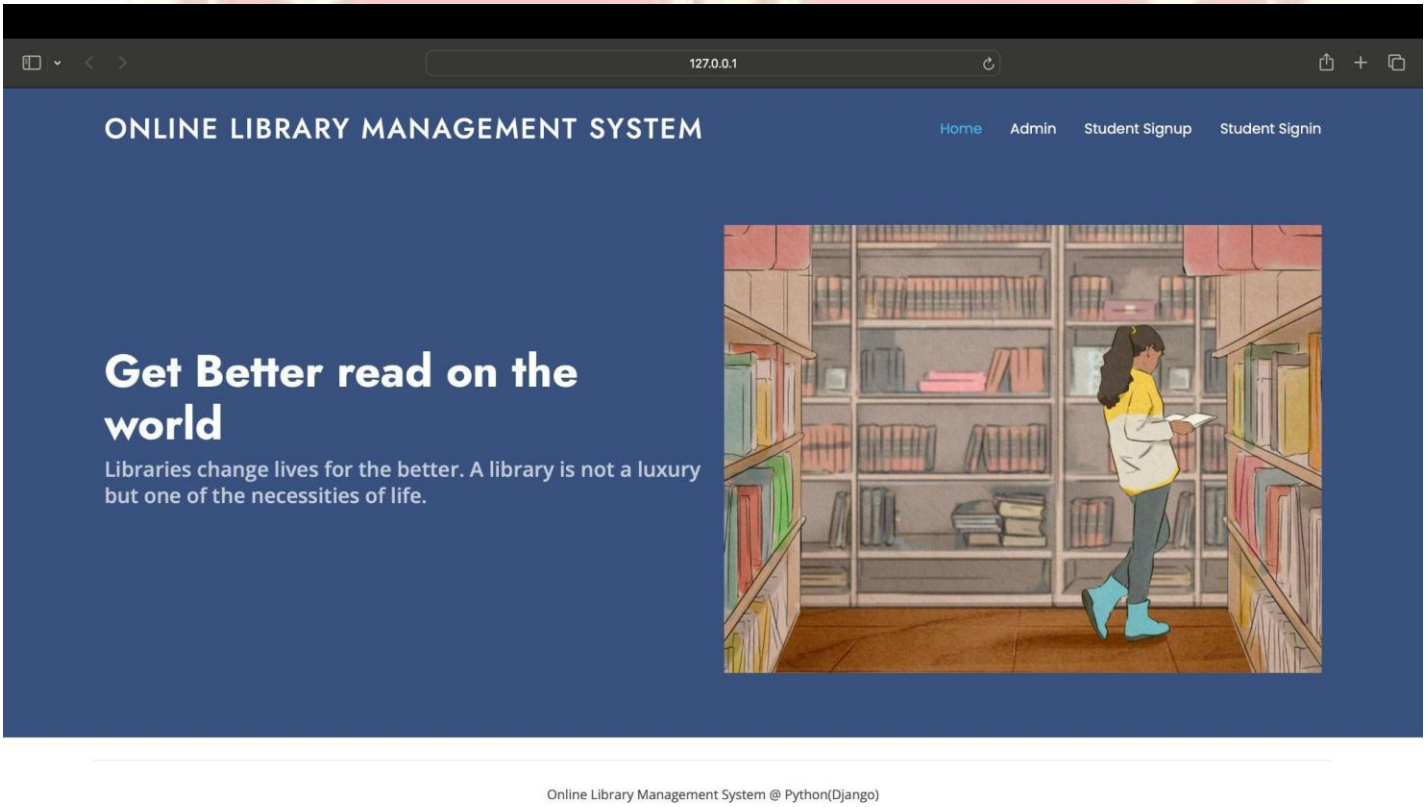

Fig 7: Main Dashboard

2. Admin Login Dashboard:

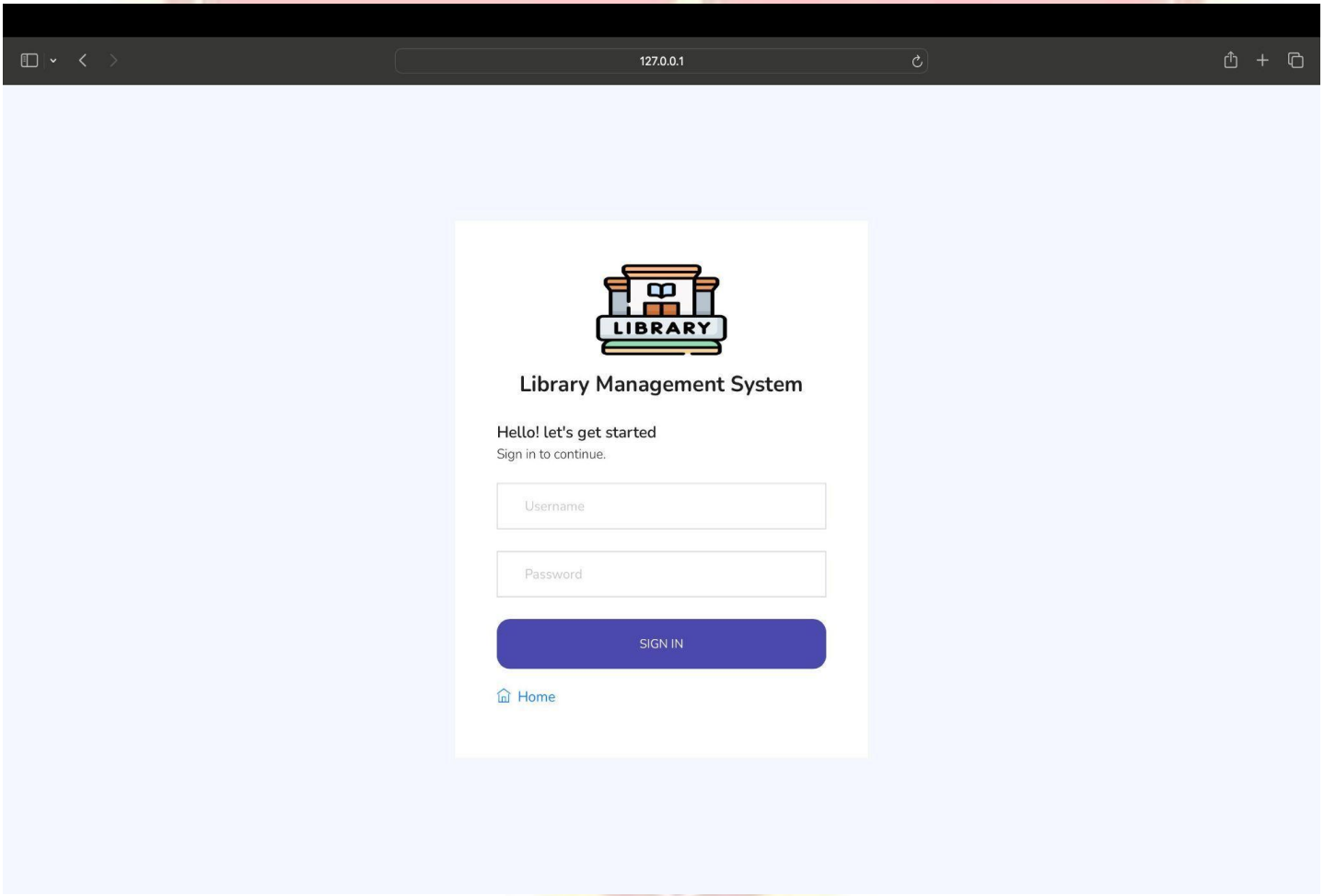

Fig 8: Admin Login Dashboard

3. Admin Dashboard:

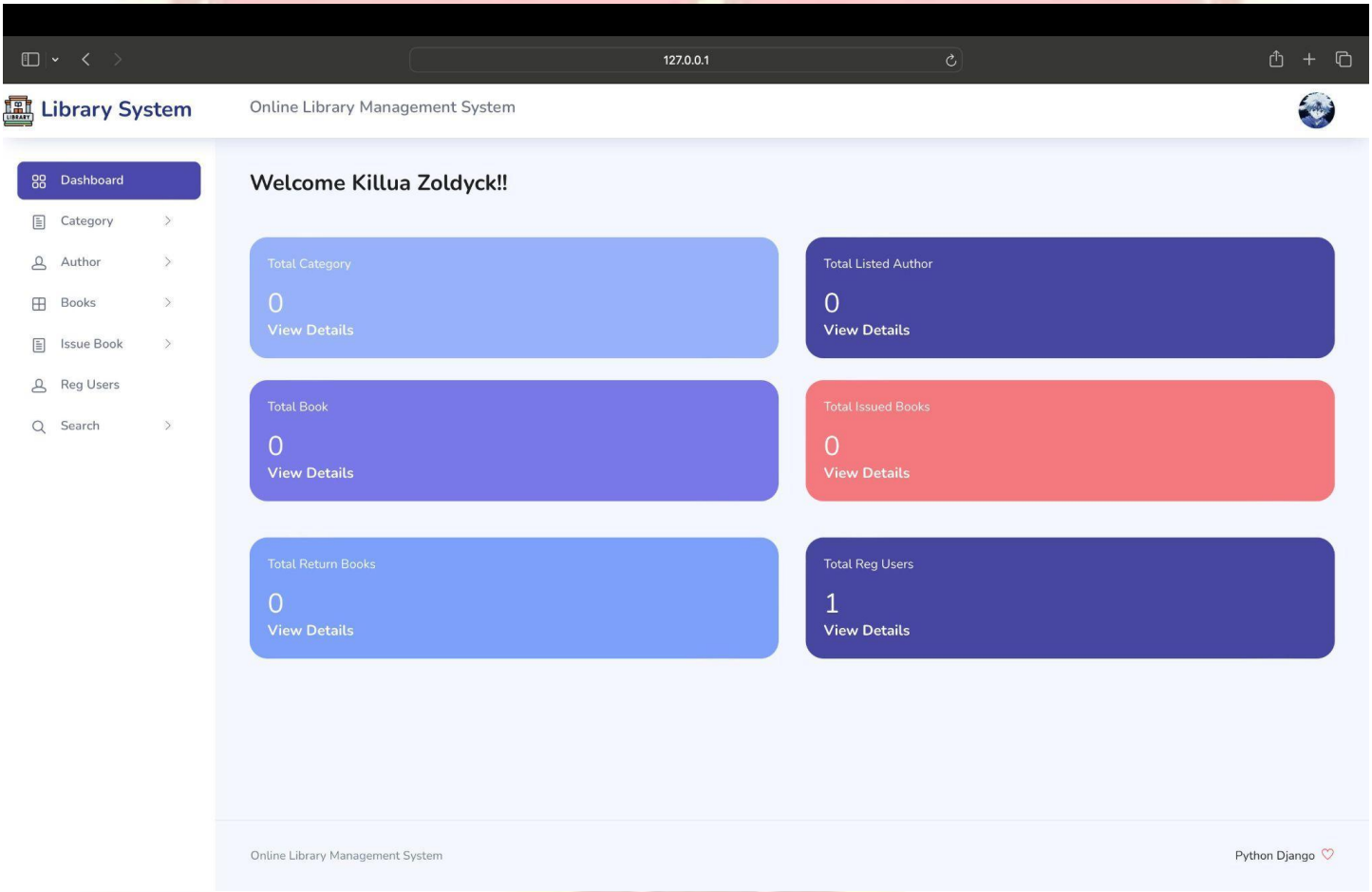

Fig 9: Admin Dashboard

4. Student Dashboard:

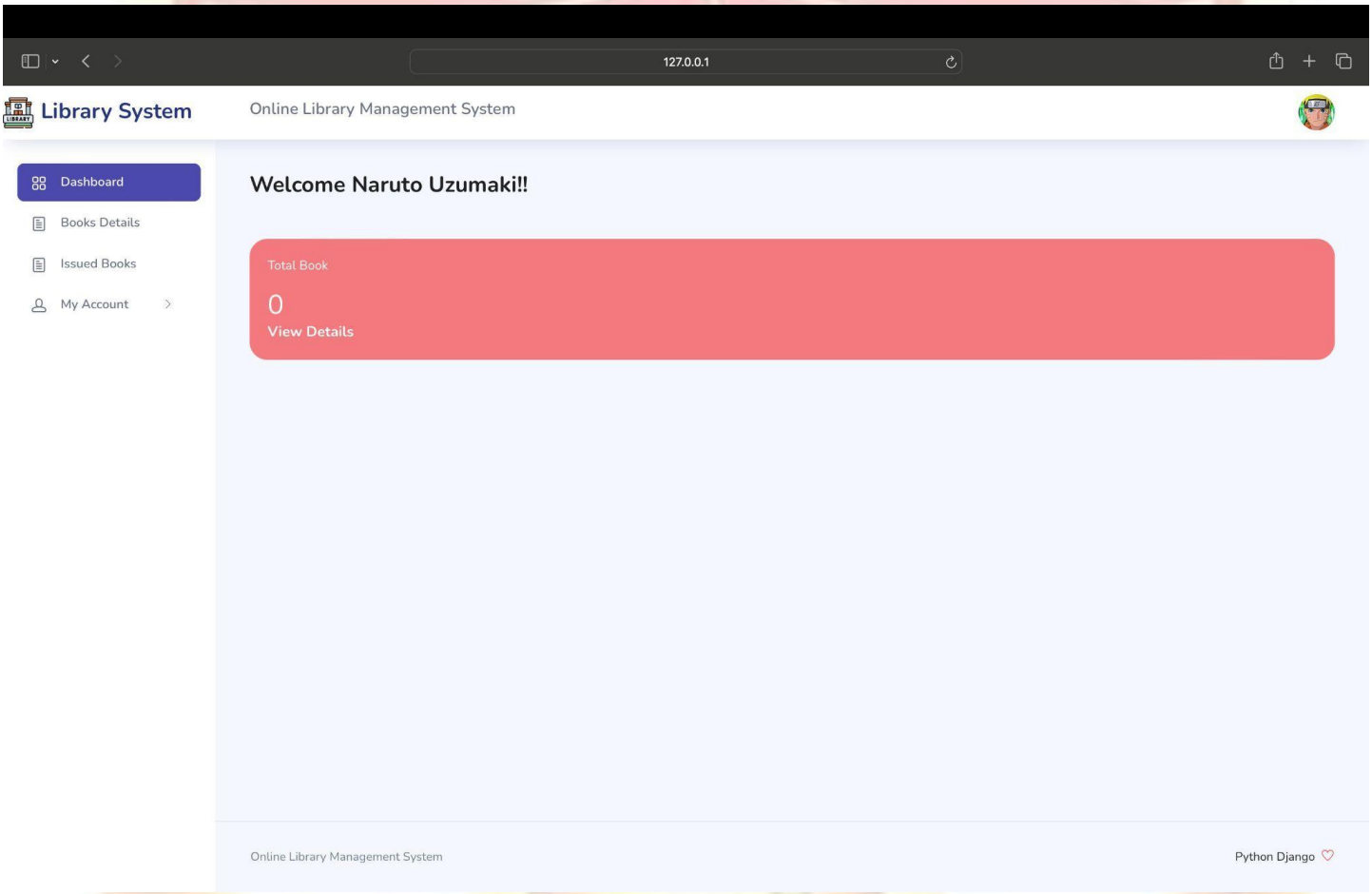

Fig 10: Student Dashboard

5. Library Book Details:

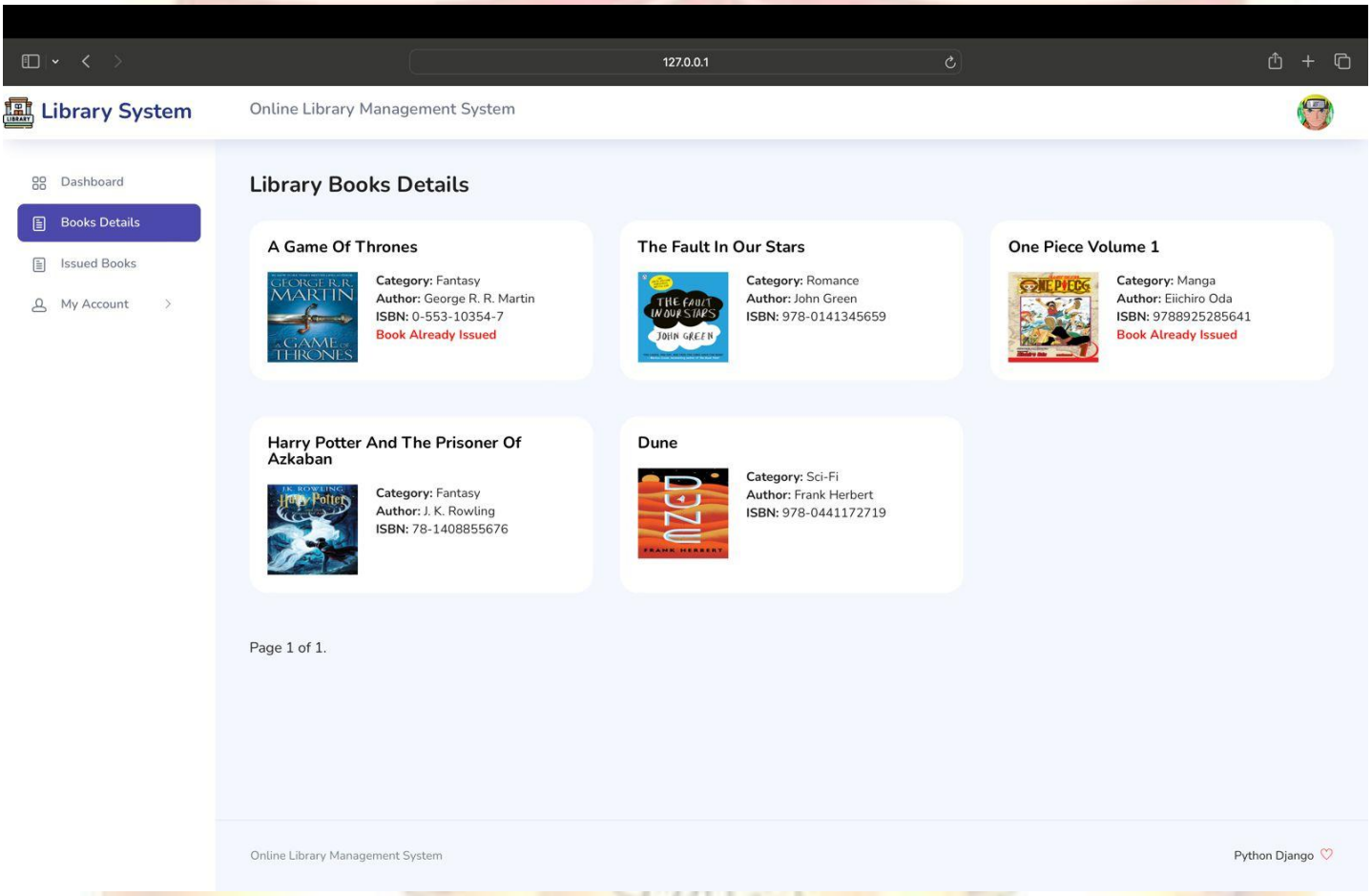

Fig 11: All Books Available in Library

6. Manage Issued Books

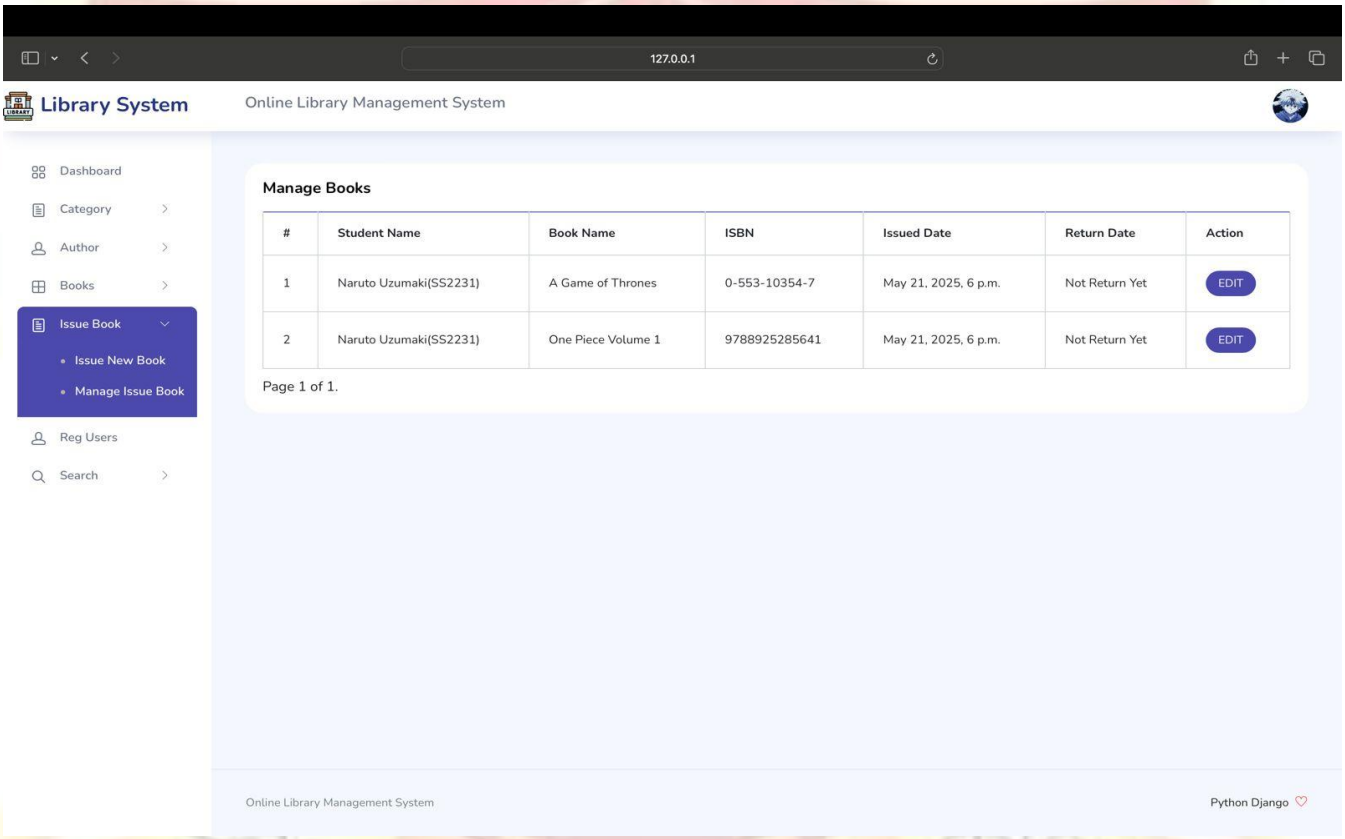

Fig 12: Issued Books List

7. Manage Books

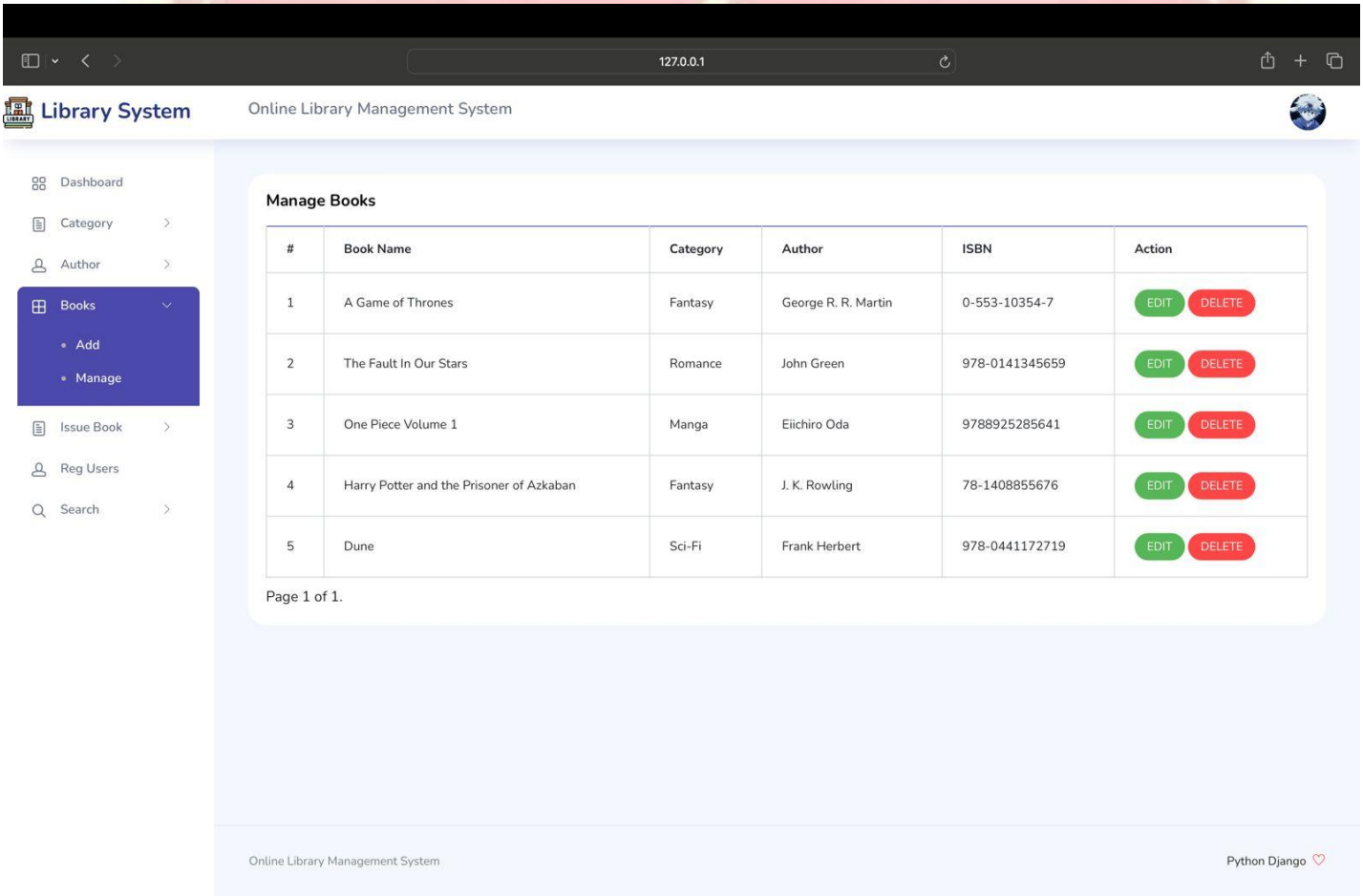

Fig 13: Manage Books of Library

8. Book Categories:

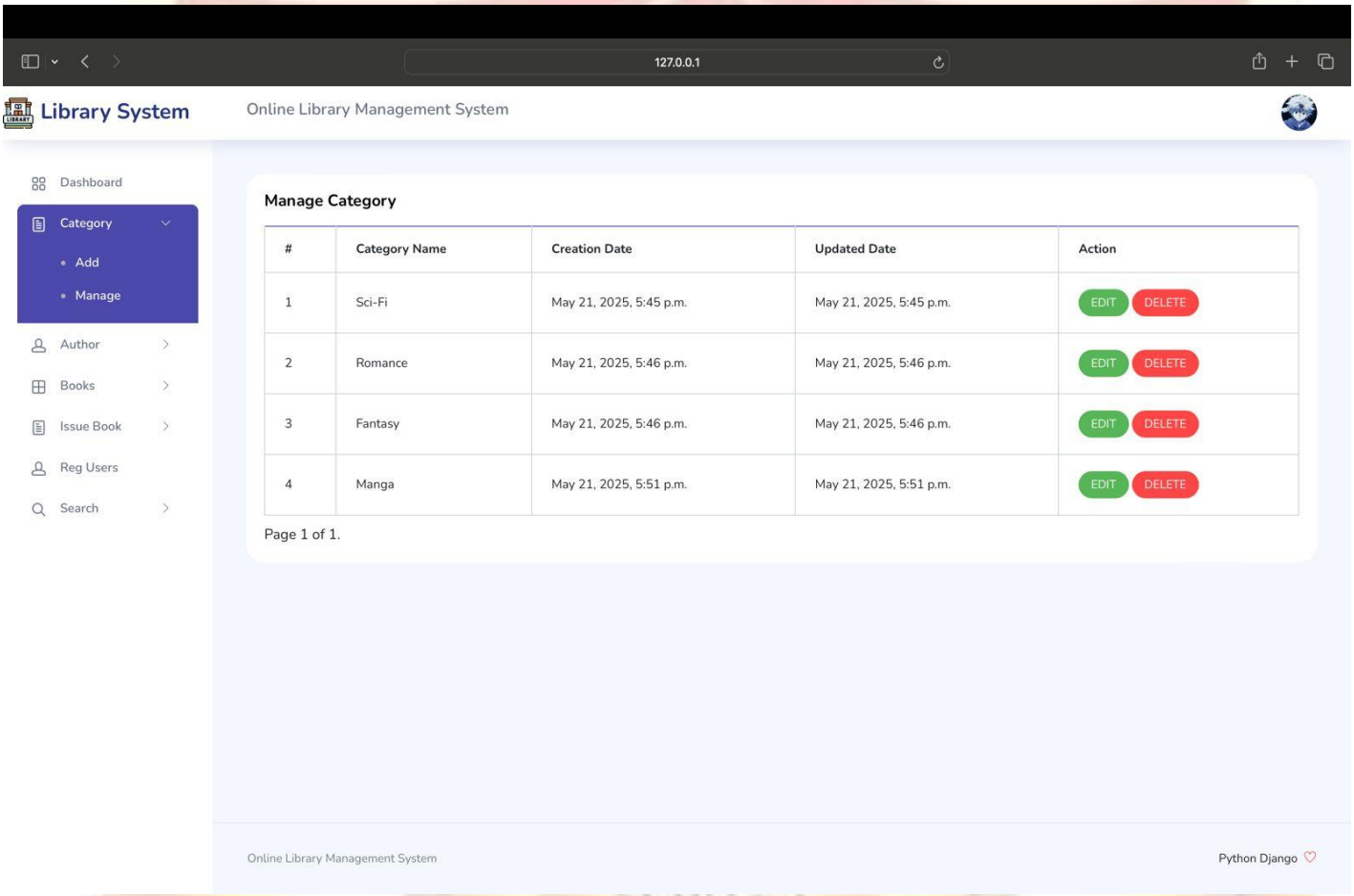

Fig 14: Book Categories of All Available Books

9. Issued Books

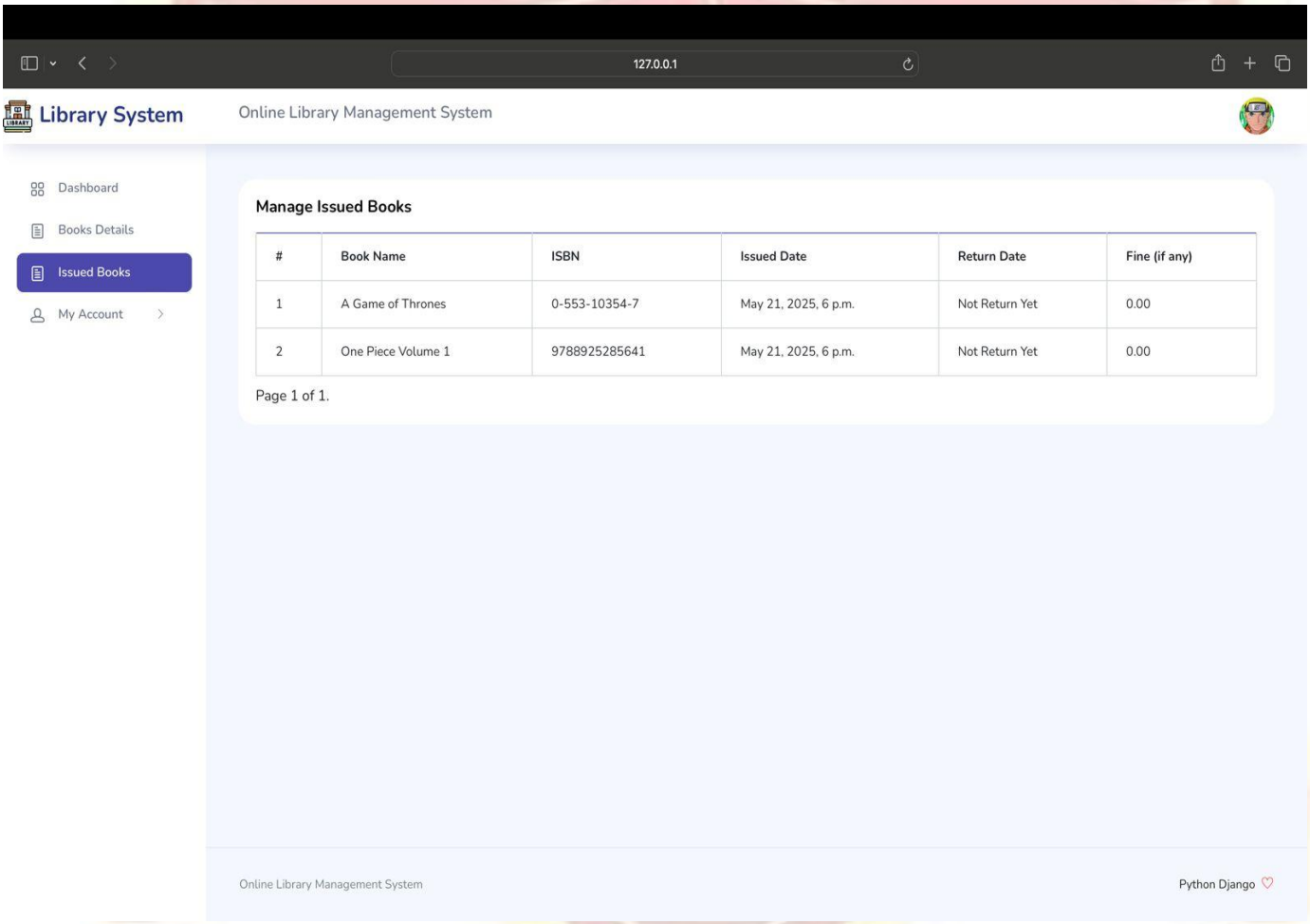

Fig 15: Issued Books List

10. Manage Author List:

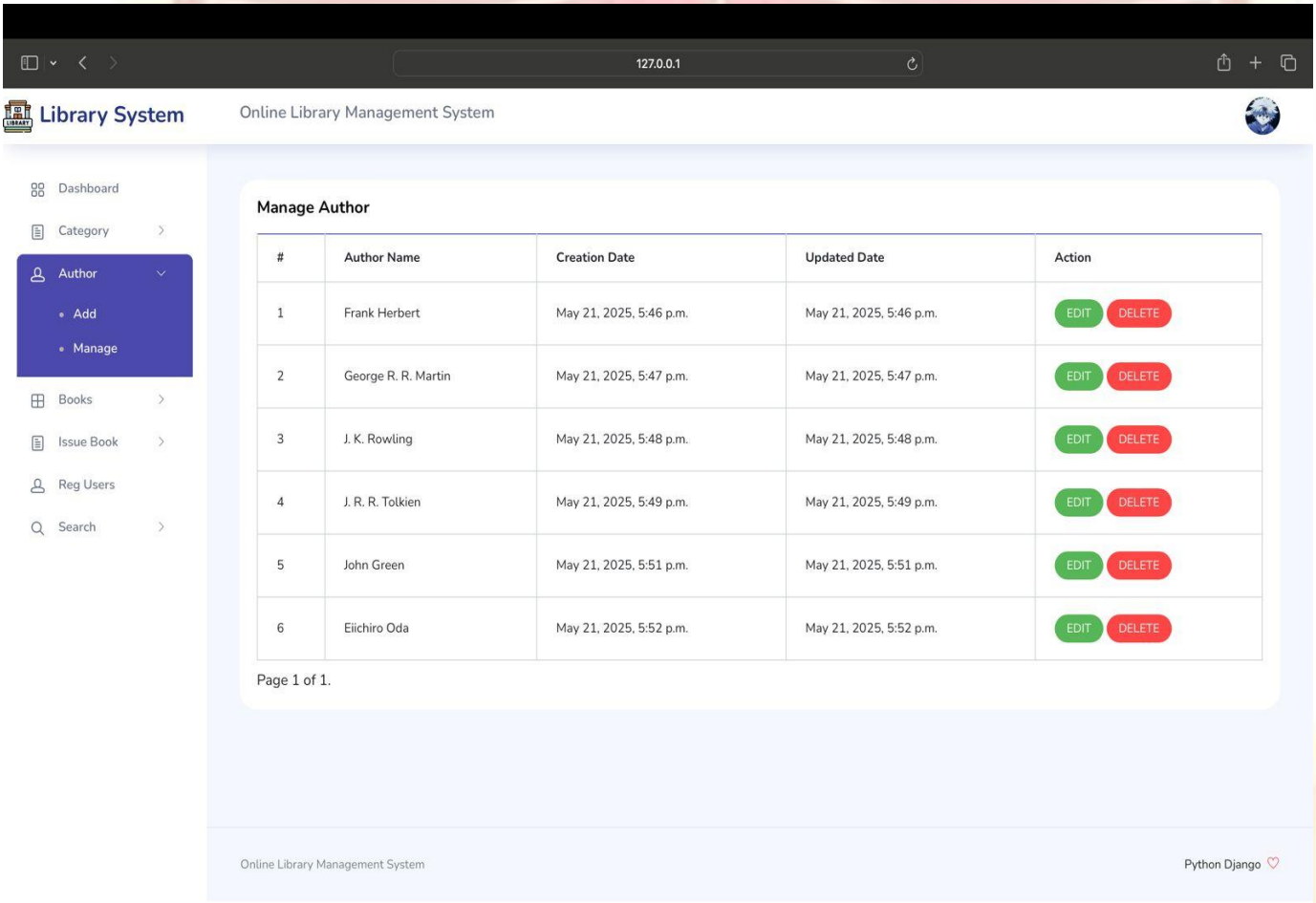

Fig 16: Author Management Dashboard

20. References

The following resources were consulted during the design, development, and documentation of the Online Library Management System:

1. Django Official Documentation
   https://docs.djangoproject.com/
   → Used for model creation, views, authentication, and admin panel setup.

2. MySQL Documentation
   https://dev.mysql.com/doc/
   → Referred to for database design, table relationships, and query structure.

3. W3Schools Django Tutorial
   https://www.w3schools.com/django/
   → For basic understanding of Django framework and template handling.

4. Stack Overflow
   https://stackoverflow.com/
   → Helped resolve errors, bugs, and implementation questions during development.

5. GitHub Django Projects
   https://github.com/
   → For structure ideas and understanding open-source Django-based library systems.

6. Python Official Documentation
   https://docs.python.org/3/
   → For core Python syntax, data types, and functions used in models and views.

7. UML Diagrams Reference
   https://www.visual-paradigm.com/
   → For designing and referencing class diagrams, ER diagrams, and use case diagrams.

8. Bootstrap & HTML/CSS Guides
   https://getbootstrap.com/
   → For frontend styling, form designs, and responsive layout handling.

9. Django Packages
   https://djangopackages.org/
   → To explore reusable components and plugins for Django projects.

Library Management System – Research References:

10. "Library Management System Using RFID and IoT"
    International Journal of Computer Applications (IJCA), 2019
    → Explores the use of automation through RFID and Internet of Things (IoT) in modern libraries.
    https://www.ijcaonline.org/archives/volume177/number6/raj-2019-ijca-918293.pdf

11. "Smart Library Management System Using Cloud and Machine Learning"
    IEEE International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020
    → Discusses the future scope of cloud-based library platforms with predictive analytics.
    IEEE Xplore
    80
12. "Design and Implementation of Digital Library Management System"
    International Journal of Computer Science and Mobile Computing (IJCSMC), 2017
    → Focuses on architecture, usability, and database management in digital library systems.
    http://www.ijcsmc.com/docs/papers/May2017/V6I5201799a35.pdf

13. "Library Automation: An Overview"
    DESIDOC Journal of Library & Information Technology, 2016
    → Surveys different library automation tools, including Koha and NewGenLib.
    https://publications.drdo.gov.in/ojs/index.php/djlit/article/view/10090

14. "Comparative Study of Library Management Systems: Koha vs. Proprietary Tools"
    International Journal of Library and Information Science, 2021
    → Compares open-source and commercial LMSs in terms of features, cost, and community support.