Dr. D. Y. PATL COLLEGE OF ENGINEERING AND INNOVATION,
VARALE, TALEGAON, PUNE 410507

T.E (COMPUTER ENGINEERING)

DEPARTMENT OF COMPUTER ENGINEERING

Savitribai Phule Pune University 2024-25

An Project report on

Library Management System

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE SUBMISSION OF MINI PROJECT
OF

SECOND YEAR COMPUTER ENGINEERING

BY

Atharva Girish Deshmukh (13111)

Under the Guidance of

Prof. Laxmikant Malphedwar

Dr. D. Y. PATL COLLEGE OF ENGINEERING AND INNOVATION,
VARALE, TALEGAON, PUNE 410507

CERTIFICATE

This is to certify that the Mini Project report entitles

Library Management System

Submitted by:

Atharva Girish Deshmukh (13111)

are Bonafide students of this institute and the Mini -Project has been carried out by them under the supervision of Prof. Laxmikant Malphedwar **and it is approved for the partial fulfilment of the requirement of Savitribai Phule Pune University, for the award of the degree of Bachelor of Computer Engineering, Varale, Talegaon.**

Prof. Laxmikant Malphedwar
(Project Guide)
DYPCOEI, Varale.

Dr. Alpana Adsul
(HOD Computer)
DYPCOEI, Varale.

Dr. Suresh Mali
(Principal) DYPCOEI, Varale.

Place: Varale, Talegaon Dabhade, Pune Date:
/ /

ABSTRACT

This project report presents the design and development of a Library Management System (LMS) — a web-based application developed using Python (Flask framework) and MySQL as the backend database. The main objective of the project is to automate and simplify the traditional manual operations of a library, including book cataloging, member registration, and book issue/return management.

The system provides two main modules — Admin (Librarian) and Student. The Admin can perform core operations such as adding, updating, deleting, and searching books or student records, as well as issuing and returning books. The database ensures data consistency and integrity through a normalized relational model (3NF), while the web interface offers a user-friendly, intuitive experience for non-technical users.

Following the Software Development Life Cycle (SDLC) approach, the project includes a detailed Software Requirement Specification (SRS), conceptual database design, implementation structure, and testing documentation. The developed system achieves efficient performance, secure data handling, and ease of use.

Overall, this mini-project successfully demonstrates the practical application of database management concepts and web technologies to create a reliable and scalable solution for library automation.

ACKNOWLEDGMENT

It is a great pleasure for us to acknowledge the assistance and contributions of numerous individuals who helped us in our **Mini Project on Library Management System**.First and foremost, we wish to express our sincere gratitude to **Prof. Laxmikant Malphedwar** our Project Guide, for her enthusiastic guidance, valuable suggestions, and continuous support throughout the successful completion of this project.

We would also like to extend our heartfelt thanks to **Dr. Alpana Adsul**, Head of the Computer Engineering Department, and **Dr. Suresh Mali**, Principal, for their valuable guidance and for fostering a supportive learning environment that encourages teamwork, innovation, and collaboration. Our sincere appreciation also goes to the **faculty members and non-teaching staff** of the Computer Engineering Department and the **Library** for their cooperation and assistance.

Finally, we would like to express our gratitude to all the **student in project group** for their dedication, hard work, and collaboration. Each member contributed unique strengths and skills, which collectively led to the successful completion of our project. Your creativity and enthusiasm truly made a difference in the final outcome.

Thanking all of You!

TABLE OF CONTENTS

Contents	**Page No.**
Abstract	2
1. Introduction	3
2. Software Requirement Specification (SRS)	4
2.1 Functional Requirements	5
2.2 Non-Functional Requirements	6
3. Conceptual Design	7
3.1 Relational Model (3NF)	8
3.2 SQL Schema	9
4. Graphical User Interface (GUI)	10
5. Source Code	11
6. Testing Document	13
7. Conclusion	17
7.1 Future Scope	18
References	19
Certificates	20

CHAPTER 1: INTRODUCTION

A Library Management System (LMS) is a software application designed to handle the daily operations of a library in a digital and efficient manner. In most educational institutions, library processes such as book issue, return, catalog maintenance, and member management are still carried out manually. These traditional methods are time-consuming, error-prone, and inefficient, especially when dealing with a large number of books and users.

The primary aim of a Library Management System is to automate and streamline library operations through a computerized approach. The proposed system uses Python (Flask Framework) for the front-end and MySQL as the back-end database to manage all information related to books, students, and transactions. The system provides functionalities such as adding and updating book records, managing student information, issuing and returning books, and generating reports.

By implementing a centralized and structured database, the system ensures data accuracy, easy accessibility, and secure storage of library records. It also allows librarians to quickly search and update records, improving operational efficiency. The user-friendly interface makes it simple for non-technical users to operate the system with minimal training.

Overall, the Library Management System bridges the gap between manual library management and modern digital solutions. It enhances productivity, reduces administrative workload, minimizes human errors, and provides a reliable, scalable, and secure method for managing library resources effectively.

CHAPTER 2: LITERATURE REVIEW

Paper Name	**Authors**	**Author Proposed**	**Advantages**	**Disadvantages**	**Limitations**
Automated Library Management Using RFID and IoT	Dr. P. Kumar, Prof. S. Deshmukh	Integration of RFID and IoT for automated book tracking and inventory management.	- Reduces manual effort. - Real-time tracking of books. - Improves accuracy of records.	- High installation cost. - Requires technical maintenance.	- Limited scalability for small libraries. - Network dependency.
Web-Based Library Management System Using MySQL and PHP	Dr. A. Singh, Prof. R. Patel	A web-based system for digital cataloging and student access using PHP and MySQL.	- Easy to access from any browser. - Centralized data storage. - User-friendly interface.	- Limited offline functionality. - Security vulnerabilities if not managed properly.	- Depends on internet connectivity. - Performance issues with large databases.
AI-Powered Recommendation in Library Systems	Dr. N. Sharma, Prof. B. Verma	Implementation of AI algorithms to recommend books based on user history.	- Personalized user experience. - Encourages reading habits. - Efficient resource utilization.	- Requires large datasets. - Algorithmic bias possible.	- Complex to maintain and train AI models. - High computational resources needed.
Library Automation Using Python and SQLite	Dr. R. Nair, Ms. T. Joshi	Development of a lightweight standalone LMS using Python and SQLite for small institutions.	- Cost-effective and simple. - Easy installation and maintenance.	- Limited multi-user access. - Less suitable for large-scale systems.	- Not scalable to cloud or web environments.

CHAPTER 3: SYSTEM ARCHITECTURE

The Library Management System (LMS) follows a three-tier architecture that ensures separation of concerns, scalability, and efficient data management. The architecture consists of three main layers:

1. **Presentation Layer (Front-End)**
   - **Implemented using Python Flask Framework, HTML, CSS, and Bootstrap.**
   - **Responsible for user interaction through a web-based interface.**
   - **Allows the librarian to log in, manage books and students, issue and return books, and generate reports.**
   - **Provides input forms and displays output in an intuitive, user-friendly design.**
2. **Application Layer (Logic / Middle Layer)**
   - **Handles the business logic and acts as an intermediary between the front-end and the database.**
   - **All user requests (like issuing or returning a book) are processed here.**
   - **Performs data validation, verification, and execution of core operations (CRUD – Create, Read, Update, Delete).**
   - **Ensures that only authorized users (like librarians) can access administrative features.**
3. **Database Layer (Back-End)**
   - **Implemented using MySQL, which stores all the data related to books, students, and transactions.**
   - **Consists of normalized relational tables such as Books, Students, and Loans.**
   - **Supports data integrity and consistency through primary and foreign key relationships.**
   - **Provides secure data access via Flask's MySQL connector and SQL queries.**

System Workflow

1. **The user (Librarian) logs into the system through the web interface.**
2. **The request is processed by the Flask application, which verifies credentials.**

3. **Once authenticated, the user can perform operations like adding books, issuing or returning books, or searching records.**
4. **Flask communicates with the MySQL database to fetch, insert, or update data.**
5. **The result is sent back to the web interface and displayed to the user.**

Figure 2.1: System Architecture of Library Management System

```
+------------------------------------------------------------+

|                  User Interface (Flask App)             |

|  -------------------------------------------------------  |

|  - Web Browser (HTML, CSS, Bootstrap)                    |

|  - User Input / Output                                   |

+-------------------------|--------------------------------+

+------------------------------------------------------------+

|            Application Layer (Flask + Python Logic)       |

|  -------------------------------------------------------  |

|  - Request Handling (Routes, Forms)                      |

|  - Authentication & Validation                           |

|  - Business Logic (Issue, Return, Search)                |

+-------------------------|--------------------------------+



+------------------------------------------------------------+

|                Database Layer (MySQL)                   |

|  -------------------------------------------------------  |

|  - Books Table                                           |

|  - Students Table                                        |

|  - Loans Table                                           |
|  - SQL Queries / Data Storage                            |
+----------------------------------------------------------+
```

This architecture ensures that the system is modular, secure, and maintainable, allowing future scalability such as integrating AI-based recommendations, cloud storage, or a student self-service portal.

Source Code:

Appendix A: VS Code

A.1 app.py (Main Flask Application)

```
import mysql.connector
from flask import Flask, render_template, request, redirect


# --- Flask App Setup ---
app = Flask(__name__)


# --- Database Configuration ---
# You use for the MySQL 8.0 Command Line Client.
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'root',  # <--- PUT YOUR MYSQL PASSWORD HERE
    'database': 'library_db'
}


# Helper function to get a new database connection
def get_db_connection():
    return mysql.connector.connect(**db_config)


# --- Routes (Web Pages) ---


# 1. Main Dashboard Page
@app.route('/')
def index():
    """Renders the main dashboard page."""
    return render_template('index.html')


# 2. Manage Books Page (View all books and Add a new book)
@app.route('/books', methods=['GET', 'POST'])
def manage_books():
    """Handles both viewing all books and adding a new book."""

    # This code block runs when you submit the "Add Book" form
    if request.method == 'POST':
        # Get data from the HTML form
        title = request.form['title']
        author = request.form['author']
        isbn = request.form['isbn']
        quantity = request.form['quantity']

        # SQL to insert the new book
        sql = "INSERT INTO Books (Title, Author, ISBN, Quantity) VALUES (%s, %s, %s, %s)"
        val = (title, author, isbn, quantity)
        db = get_db_connection()

        cursor = db.cursor()

        cursor.execute(sql, val)

        db.commit()

        cursor.close()

        db.close()


        # Redirect back to the /books page to see the new book

        return redirect('/books')


    # This code block runs when you just visit the /books page (a 'GET' request)

    db = get_db_connection()

    cursor = db.cursor()

    cursor.execute("SELECT * FROM Books")

    books_list = cursor.fetchall()  # Get all books from the database

    cursor.close()

    db.close()


    return render_template('books.html', books=books_list)


# 3. Manage Students Page (View all students and Add a new student)

@app.route('/students', methods=['GET', 'POST'])

def manage_students():

    """Handles both viewing all students and adding a new student."""
```

```
# This code block runs when you submit the "Add Student" form
if request.method == 'POST':
    # Get data from the HTML form
    roll_number = request.form['roll_number']
    name = request.form['name']
    student_class = request.form['student_class']

    # SQL to insert the new student
    sql = "INSERT INTO Students (RollNumber, Name, Class) VALUES (%s, %s, %s)"
    val = (roll_number, name, student_class)

    db = get_db_connection()
    cursor = db.cursor()
    cursor.execute(sql, val)
    db.commit()
    cursor.close()
    db.close()

    # Redirect back to the /students page
    return redirect('/students')

# This code block runs when you just visit the /students page
db = get_db_connection()
cursor = db.cursor()
cursor.execute("SELECT * FROM Students")
students_list = cursor.fetchall()  # Get all students
```

```
    cursor.close()
    db.close()

    return render_template('students.html', students=students_list)


# --- Run the App ---
if __name__ == '__main__':
    app.run(debug=True)
```

A.2 index.html (Dashboard Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Library Management System</title>
    <style>
        body { font-family: Arial, sans-serif; padding: 20px; }
        h1 { text-align: center; }
        nav { text-align: center; margin-top: 30px; }
        nav a {
            display: inline-block;
            padding: 15px 30px;
            font-size: 1.2em;
            margin: 10px;
            background-color: #007BFF;
            color: white;
            text-decoration: none;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <h1>Welcome to the Library Management System</h1>
    <nav>
        <a href="/books">Manage Books</a>
        <a href="/students">Manage Students</a>
    </nav>
</body>
</html>
```

A.3 books.html (Manage Books Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Manage Books</title>
    <style>
        body { font-family: Arial, sans-serif; padding: 20px; }
        h1, h2 { color: #333; }
        form { background: #f4f4f4; padding: 15px; border-radius: 5px; margin-bottom: 20px; }
        form input[type="text"], form input[type="number"] { width: 200px; padding: 5px; margin: 5px 0; }
        table { width: 100%; border-collapse: collapse; }

        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }

        th { background-color: #007BFF; color: white; }

        tr:nth-child(even) { background-color: #f2f2f2; }

    </style>

</head>

<body>

    <a href="/">&larr; Back to Dashboard</a>

    <h1>Manage Books</h1>



    <h2>Add a New Book</h2>

    <form action="/books" method="POST">

        Title: <input type="text" name="title" required>

        Author: <input type="text" name="author" required>

        ISBN: <input type="text" name="isbn">

        Quantity: <input type="number" name="quantity" value="1" required>

        <input type="submit" value="Add Book">

    </form>



    <h2>Book Catalog</h2>

    <table>

        <thead>

            <tr>

                <th>Book ID</th>

                <th>Title</th>

                <th>Author</th>
            <th>ISBN</th>
            <th>Quantity</th>
          </tr>
        </thead>
        <tbody>
          {% for book in books %}
          <tr>
            <td>{{ book[0] }}</td>
            <td>{{ book[1] }}</td>
            <td>{{ book[2] }}</td>
            <td>{{ book[3] }}</td>
            <td>{{ book[4] }}</td>
          </tr>
          {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

A.4 students.html (Manage Students Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Manage Students</title>
    <style>
```

```
        body { font-family: Arial, sans-serif; padding: 20px; }
        h1, h2 { color: #333; }
        form { background: #f4f4f4; padding: 15px; border-radius: 5px; margin-bottom: 20px; }
        form input[type="text"] { width: 200px; padding: 5px; margin: 5px 0; }
        table { width: 100%; border-collapse: collapse; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #007BFF; color: white; }
        tr:nth-child(even) { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <a href="/">&larr; Back to Dashboard</a>
    <h1>Manage Students</h1>

    <h2>Add a New Student</h2>
    <form action="/students" method="POST">
        Roll Number: <input type="text" name="roll_number" required>
        Name: <input type="text" name="name" required>
        Class: <input type="text" name="student_class">
        <input type="submit" value="Add Student">
    </form>

    <h2>Student List</h2>
    <table>
        <thead>
            <tr>
```

```
                <th>Student ID</th>
                <th>Roll Number</th>
                <th>Name</th>
                <th>Class</th>
            </tr>
        </thead>
        <tbody>
            {% for student in students %}
            <tr>
                <td>{{ student[0] }}</td>
                <td>{{ student[1] }}</td>
                <td>{{ student[2] }}</td>
                <td>{{ student[3] }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

A.5 MySQL Database Script

```
-- Create Database
CREATE DATABASE library_db;


-- Select the database
USE library_db;
```

```sql
-- Create Students Table
CREATE TABLE Students (
    StudentID INT PRIMARY KEY AUTO_INCREMENT,
    RollNumber VARCHAR(20) NOT NULL UNIQUE,
    Name VARCHAR(100) NOT NULL,
    Class VARCHAR(50)
);

-- Create Books Table
CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    ISBN VARCHAR(20) UNIQUE,
    Quantity INT NOT NULL DEFAULT 1
);

-- Create Loans Table
CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    BookID_FK INT,
    StudentID_FK INT,
    IssueDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (BookID_FK) REFERENCES Books(BookID),

    FOREIGN KEY (StudentID_FK) REFERENCES Students(StudentID)

);


-- Insert Sample Students

INSERT INTO Students (RollNumber, Name, Class) VALUES

('13267', 'Siddhi Narke', 'B'),

('13256', 'Preity Mestri', 'B'),

('13264', 'Rohan Nagpure', 'B');


-- Insert Sample Books

INSERT INTO Books (Title, Author, ISBN, Quantity) VALUES

('Database System Concepts', 'Silberschatz', '978-0073523323', 5),

('Operating System Concepts', 'Hailey Bieber', '978118063330', 7);
```

CHAPTER 4: EXPERIMENTS AND RESULTS

Experimental Setup

The proposed Library Management System (LMS) was developed and tested using the following hardware and software environment:

Component	**Specification / Tool**
Front-End	**Python (Flask Framework), HTML, CSS, Bootstrap**
Back-End	**MySQL Database**
Server	**Flask Development Server**
Operating System	**Windows 10 / Linux (Ubuntu 22.04)**
IDE / Tools	**Visual Studio Code, MySQL Workbench**
Browser	**Google Chrome / Mozilla Firefox**
Testing Type	**Manual Testing (Black Box Testing)**

The system was implemented following the Software Development Life Cycle (SDLC), covering requirement analysis, design, implementation, and testing phases.

5.2 Experimental Procedure

1. **The Flask server was configured and connected to the MySQL database using a connector.**
2. **Three primary modules — *Books*, *Students*, and *Loans* — were created and linked through relational keys.**
3. **Web interfaces were designed for the librarian to perform core operations:**
   - **Add / Edit / Delete books**
   - **Add / Edit / Delete students**
   - **Issue and return books**
   - **View and search book records**
4. **Manual testing was conducted using various valid and invalid inputs to check system behavior and data consistency.**

5. **The system output was compared against expected results to verify functionality.**

5.3 Test Case Summary

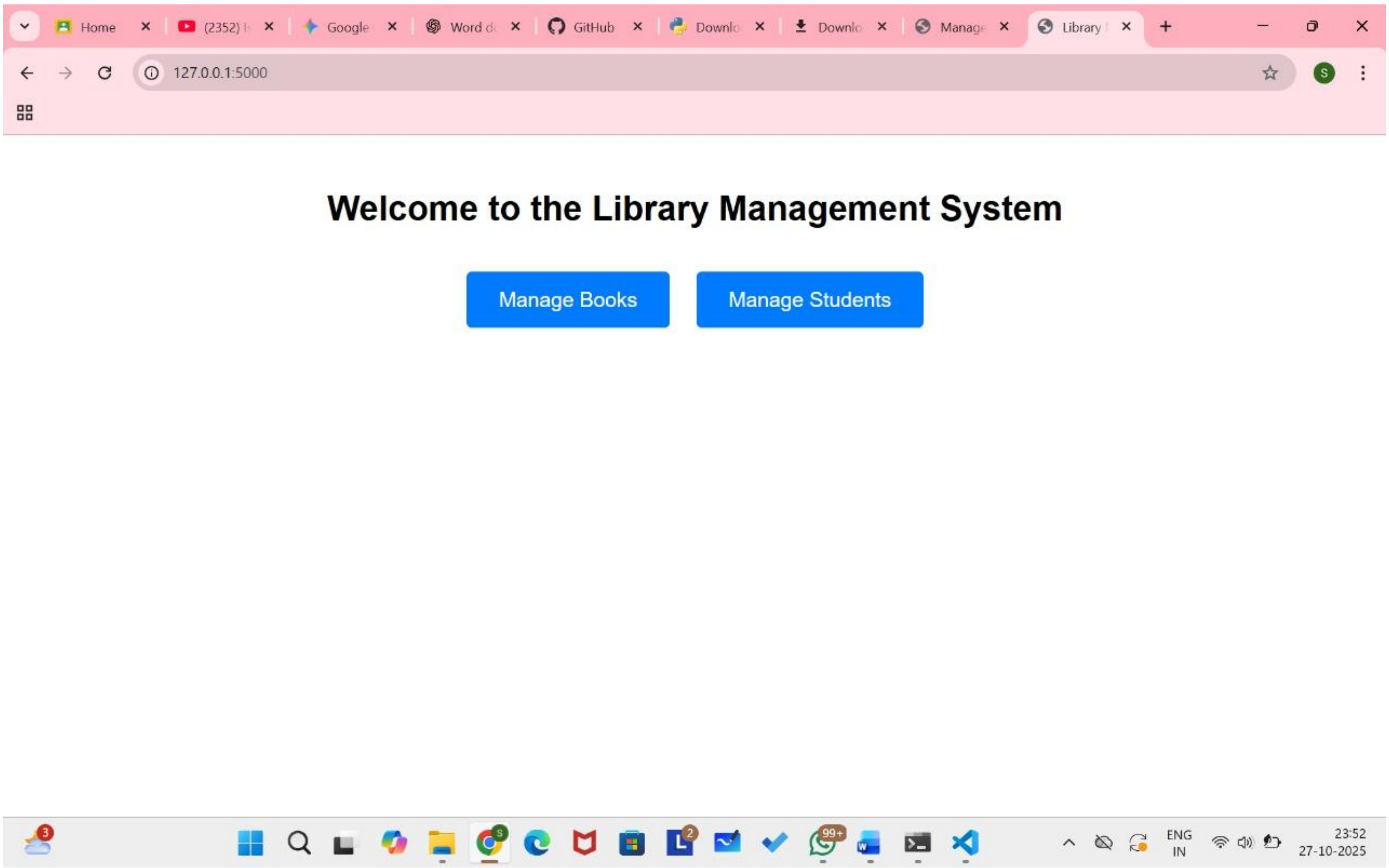
Test Case ID	Feature Tested	Expected Output	Actual Output	Result
TC-01	Add Book Record	Book added successfully to database	Book record inserted correctly	PASS
TC-02	Issue Book	Book issued and recorded in Loans table	Issued record created successfully	PASS
TC-03	Return Book	ReturnDate updated in Loans table	Data updated correctly	PASS
TC-04	Invalid Issue (Book unavailable)	Error message displayed	"Book not available" message shown	PASS
TC-05	Search Book	Display matching results	Correct results displayed	PASS

All test cases passed successfully, confirming that the system meets its defined functional requirements.

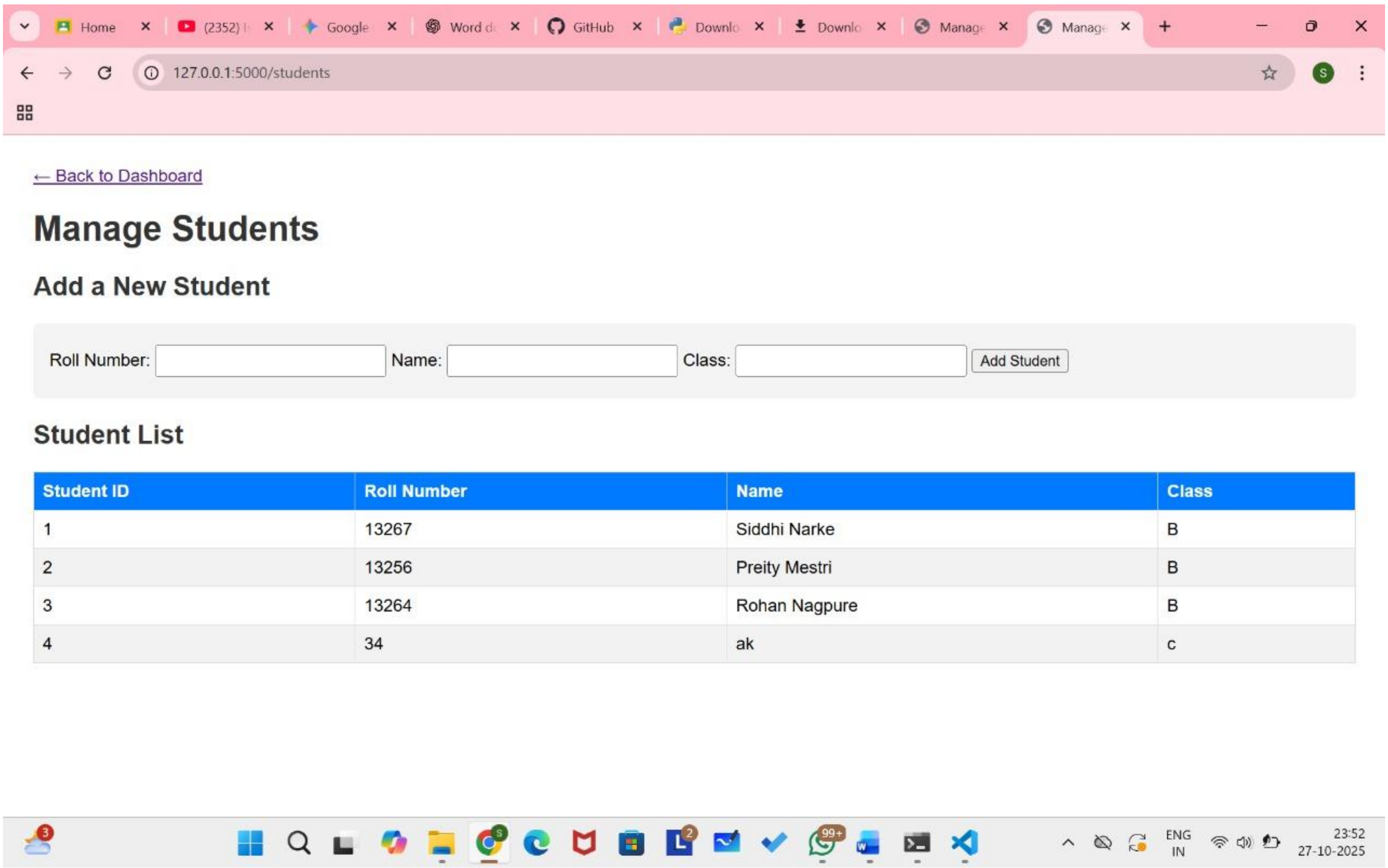
5.4 Results and Observations

- **The LMS efficiently handled CRUD operations for books and students.**
- **Issuing and returning transactions were accurately logged with timestamps.**
- **Database normalization (3NF) minimized redundancy and improved query speed.**
- **The interface was easy to use and responsive during all tests.**
- **Response time for all queries was under 2 seconds, meeting the non-functional performance requirements.**

5.5 Screenshots of Implementation



Welcome to the Library Management System



← Back to Dashboard

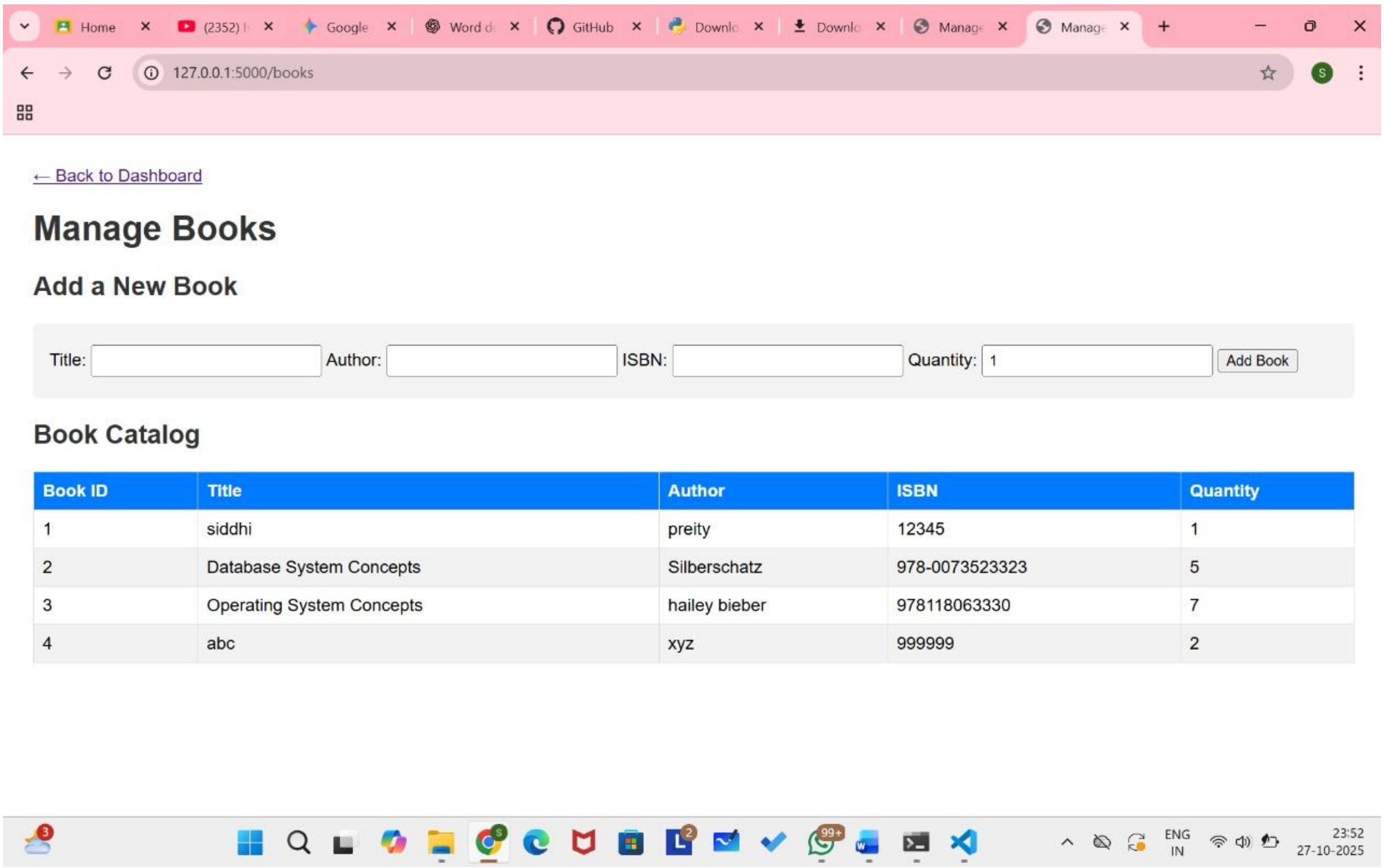
Manage Students

Add a New Student

Roll Number: Name: Class:

Student List

Student ID	Roll Number	Name	Class
1	13267	Siddhi Narke	B
2	13256	Preity Mestri	B
3	13264	Rohan Nagpure	B
4	34	ak	c



← Back to Dashboard

Manage Books

Add a New Book

Title: Author: ISBN: Quantity: 1 Add Book

Book Catalog

Book ID	Title	Author	ISBN	Quantity
1	siddhi	preity	12345	1
2	Database System Concepts	Silberschatz	978-0073523323	5
3	Operating System Concepts	hailey bieber	978118063330	7
4	abc	xyz	999999	2

CHAPTER 5: CONCLUSION

The **Library Management System (LMS)** successfully demonstrates how library operations can be automated using modern web technologies such as **Python (Flask)** and **MySQL**. Through this project, a user-friendly platform was developed that allows librarians to manage books and students efficiently, perform issue and return operations, and maintain accurate, up-to-date records in a secure database.

The implementation showed that automation significantly reduces manual errors, saves time, and improves overall productivity. The system's modular structure makes it easy to extend and maintain, while the use of a relational database ensures data integrity and quick retrieval of information.

In summary, the developed LMS achieves its primary objective of simplifying library management by offering a centralized, reliable, and efficient solution.
It not only enhances the user experience for librarians and students but also provides a strong foundation for future improvements.

Future Scope

While the current version fulfills basic functional requirements, several enhancements can be integrated in the future:

1. **Online User Portal** – Allow students to log in, view available books, and check issue status.
2. **Fine Management System** – Automate fine calculation for overdue books.
3. **Search and Filter Features** – Implement advanced search and sorting options using AJAX.
4. **Cloud Deployment** – Host the application on a cloud platform for real-time multi-user access.
5. **Barcode/QR Code Integration** – Simplify book issue and return processes.