EAST WEST UNIVERSITY

Department of CSE

LIBRARY MANAGEMENT SYSTEM

Submitted by

Name of Student:

MD. Shahrukh Hossain Shihab

Nushrat Jaben Aurnima

Shairin Akter Hashi

Sabera Zannat Sheba

Zihad Khan

Student Id:

2022-1-60-372

2022-2-60-146

2022-2-60-102

2022-2-60-095

2022-2-60-107

Course Title: Database System.

Course Code: CSE 302

Semester: Fall 2024

Submitted to

Md Sabbir Hossain

Lecturer, Department of Computer Science and Engineering

Table Of Contents

Executive Summary

In the modern digital age, the management of libraries has transitioned from manual, paper-based operations to streamlined, automated systems that offer convenience, speed, and enhanced efficiency. The Library Management System (LMS) is designed to modernize and optimize library operations, addressing the inefficiencies of traditional systems and meeting the needs of members, librarians, and administrators. The system offers functionalities such as online book requests and automated fine calculations for overdue returns. Users are categorized into three roles: Members, Librarians, and Admins, each with specific responsibilities and access levels.

Key functionalities include book inventory management, user profile management, overdue fine management. The LMS aims to enhance the overall user experience by providing intuitive interfaces, seamless workflows, and secure data handling. This summary outlines the design, functionality, and benefits of the LMS for efficient library operations.

Key Features

- **Online Book Requests**: Members request and return books online, reducing manual intervention.
- **Automated Inventory Management**: updates the database (add, update, delete).
- **Role-Based Access**: Provides specific functionalities to Members, Librarians, and Admins for secure and efficient management.
- **Secure Data Handling**: Ensures user and transaction data is protected from unauthorized access.

Functional Requirements

- ***Member Portal:***
  - Profile Management: Members can view and update their profiles.
  - Book Request and Return: Enables members request books and return online.
  - Request Tracking: Members can track the status of their requests.
  - Fine Management: Automatically calculates and displays fines for overdue books.
- ***Librarian Portal:***
  - Profile Management: Librarians can view their contact and professional details.
  - Book Inventory Management: Librarians can add, update, and remove books.
  - Process Requests: Librarians manage book requests, issue books, and set return deadlines.
- ***Admin Portal:***
  - Manage Users: Admins can add or remove librarians and view member details.
  - Oversee Transactions: Admins can monitor, and audit book requests, returns, and fines.
  - Generate Reports: Provides detailed report to user for fine.

Benefits

- **Efficiency**: Streamlines library operations, reducing manual work and errors.
- **User Convenience**: Facilitates remote access to library resources and services.
- **Cost-Effectiveness**: Minimizes operational costs by automating processes and reducing paper usage.
- **Scalability**: Supports growing user bases and expanding library resources.
- **Enhanced User Experience**: Offers intuitive interfaces and real-time updates for seamless operations.
- **Environmental Impact**: Reduces paper usage, contributing to eco-friendly practices.

Implementation Overview

- **Technology Stack**: Developed using PHP for the backend and a relational database for data storage.
- **Deployment**: Implemented in a controlled environment for testing and validation.
- **Testing**: Rigorous testing to ensure reliability and functionality across all modules.
- **Future Enhancements**:
    - Integration with external databases for inter-library collaboration.
    - Advanced analytics to optimize inventory and predict user preferences.
    - Mobile app development for enhanced accessibility.

The Library Management System is a transformative solution for libraries, ensuring efficient resource management, secure data handling, and user satisfaction. By leveraging modern technology, the LMS empowers libraries to meet the demands of the digital era.

Abstract

The goal of this project is to design and implement a Library Management System (LMS) that streamlines library operations and improves user experiences. The system provides distinct roles for three actors: Members, Librarians, and Administrators. Key functionalities include profile management, book inventory management, and fine handling. This report outlines the design, functionality, and implementation details of the LMS.

1. Introduction

Libraries play a vital role in knowledge dissemination and resource management. The increasing demand for digital solutions has necessitated the development of efficient systems to manage library operations. The LMS aims to automate manual processes, enhance operational efficiency, and provide a seamless experience for users.

2. Problem Statement

The current manual library system is inefficient, prone to errors, and unable to handle growing user demands. Common issues include tracking book request and return, managing user profiles, and handling fines for overdue books. A robust, scalable, and user-friendly digital solution is essential to address these challenges.

3. Search Question and Research Hypothesis

➢ ***Search Question***

"What are the most efficient ways to design a Library Management System that automates core library operations while ensuring user satisfaction and scalability?"

➢ ***Research Hypothesis***

A well-designed Library Management System that leverages role-based access control, real-time inventory management, and automated fine calculation can significantly enhance operational efficiency and user satisfaction.

➢ ***Objectives***

- Automate library operations.
- Provide role-based access and functionality for Members, Librarians, and Admins.
- Ensure accurate tracking of book inventory and user transactions.
- Enable efficient fine management for overdue books.
- Enhance user experience through intuitive interfaces and real-time updates.

4. Actors and Their Responsibilities

➢ ***4.1 Member***

- **View and Update Profile:** Members can manage their personal details.
- **View Existing Books:** Members can view the library's catalogue to find books.
- **Request Books:** Members can request specific books from the librarian.

- **Track Requests:** Members can view the status of their book requests and monitor issued books.
- **Return Books:** Members can return borrowed books.
- **Fine Management:** If books are returned after the due date, fines are automatically calculated and applied.
- **View Reports:** Members can view if they have any fine or not.

➢ *4.2 Librarian*

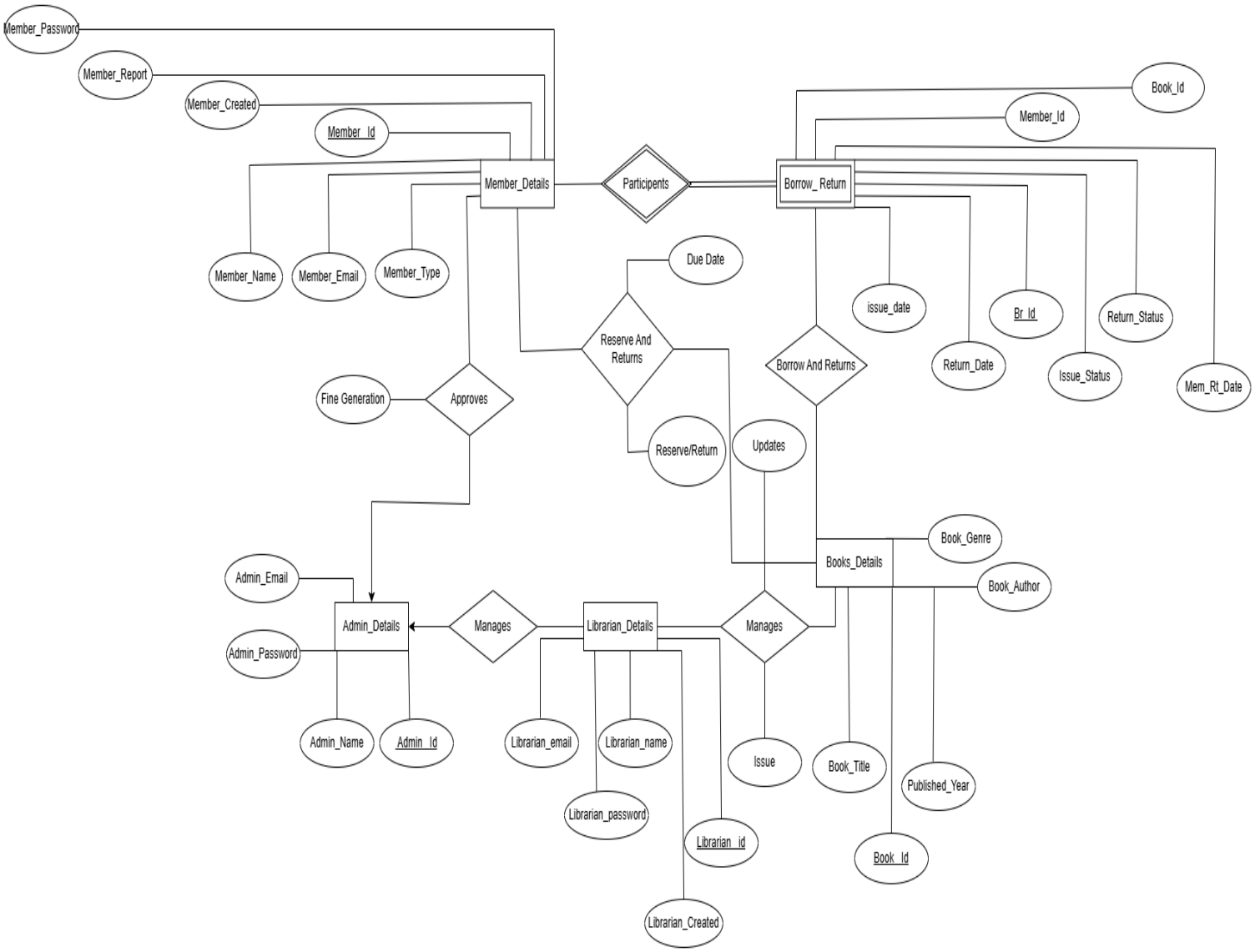
- **View Profile:** Librarians can view their contact and professional details.
- **Add Books:** Librarians can add new books to the library's inventory.
- **Manage Books:** Librarians can update or remove book records as needed.
- **Process Book Requests:** Librarians handle book requests, issue books, and assign return dates to members.

➢ *4.3 Admin*

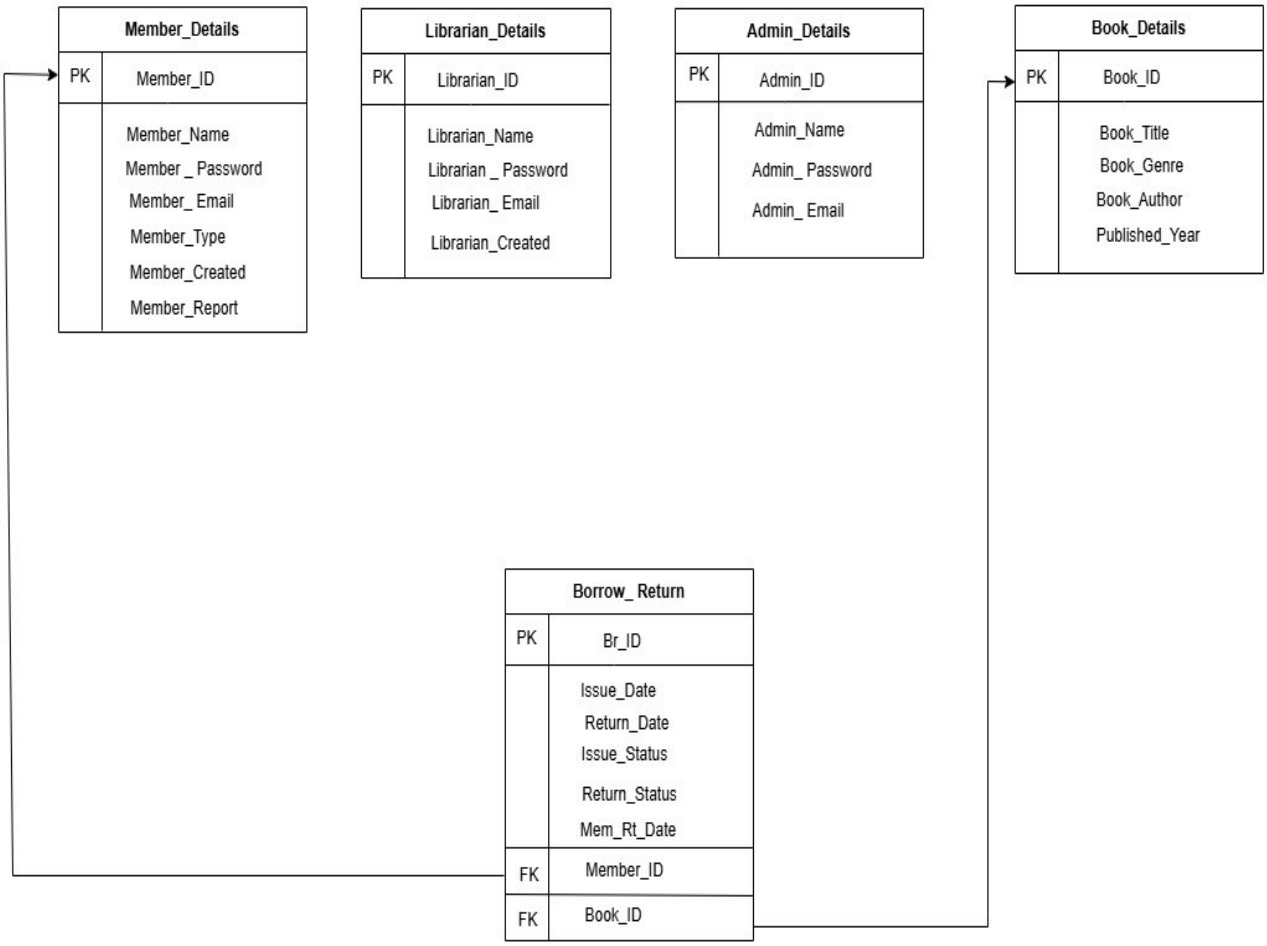
- **Add Librarian:** Admins can create new librarian accounts.
- **View and Delete Librarians:** Admins can manage the list of librarians, including account deletion.
- **View Member List:** Admins can access detailed records of library members.
- **Issue Fines:** Admins can generate reports for fines and they can also, oversee the entire fine management process.

5. System Design

➢ *5.1 Entity-Relationship (ER) Diagram*



➢ *5.2 Relational Schema Diagram*



6. Methodology

1. **Requirement Gathering:** Conducted stakeholder interviews to understand system requirements.
2. **System Design:** Developed ER diagrams, relational schemas.
3. **Implementation:** Built the system using PHP.
4. **Testing:** Performed rigorous testing to ensure functionality and reliability.
5. **Deployment:** Deployed the system in a controlled environment for final validation.

7. Results and Discussion

The Library Management System successfully automates the following:

- Streamlined book inventory management.
- Efficient role-based access control for Members, Librarians, and Admins.
- Real-time fine calculation and notification for overdue books.
- Enhanced user satisfaction through intuitive interfaces and seamless workflows.

8. Conclusion

The Library Management System addresses the inefficiencies of traditional library systems by offering a robust, scalable, and user-friendly solution. Future enhancements could include:

- Integration with external databases for inter-library collaborations.
- Advanced analytics to predict user preferences and optimize inventory.
- Mobile app development for increased accessibility.

9. SQL Queries

Below are the SQL queries used for setting up the Library Management System:

➢ ***9.1 Table Creation Queries***

- Member Details Table

```
CREATE TABLE member_details (
    member_id INT AUTO_INCREMENT PRIMARY KEY,
    member_name VARCHAR(100),
    member_password VARCHAR(255),
    member_email VARCHAR(100),
    member_type VARCHAR(100),
    member_created TIMESTAMP,
 member_report VARCHAR(2000)
);
```

- Librarian Details Table

```
CREATE TABLE librarian_details (
    librarian_id INT AUTO_INCREMENT PRIMARY KEY,
    librarian_name VARCHAR(100),
    librarian_password VARCHAR(255),
    librarian_email VARCHAR(100),
    librarian_created TIMESTAMP
);
```

- Admin Details Table

```
CREATE TABLE admin_details (
    admin_id INT AUTO_INCREMENT PRIMARY KEY,
    admin_name VARCHAR(100),
    admin_password VARCHAR(255),
    admin_email VARCHAR(100)
);
```

- Book Details Table

```
CREATE TABLE book_details (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    book_title VARCHAR(2000),
    book_genre VARCHAR(10),
    book_author VARCHAR(500),
    published_year YEAR
);
```

- Borrow and Return Table

```
    CREATE TABLE borrow return (
        br_id INT AUTO_INCREMENT PRIMARY KEY,
        issue_date DATE,
        return_date DATE,
        issue_status VARCHAR(100) DEFAULT 'Not Issued',
        return_status VARCHAR(100) DEFAULT 'Not Returned',
        mem_rt_date DATE,
        member_id INT NOT NULL,
        book_id INT NOT NULL,
        FOREIGN KEY (member_id) REFERENCES member_details(member_id) ON
    DELETE CASCADE,
    FOREIGN KEY (book_id) REFERENCES book_details(book_id) ON DELETE
CASCADE
    );
```

➢ *9.2 Database Connection (config.php)*

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "library_db";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

➢ *9.3 SQL's used in code*

- Member Dashboard/View Member Profile

```
public function fetchMemberDetails() {
        $sql = "SELECT * FROM member_details WHERE member_id = $this-
>memberId";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            $this->memberData = $result->fetch_assoc();
        } else {
            die("Member not found.");
        }
    }
```

- Member Log IN

```
$sql = "SELECT * FROM member_details WHERE member_name = '$member_name'";
```

- Update Member Class

```
 -- member class
     public function getMemberDetails($member_id) {
        $sql = "SELECT member_name, member_email, member_type,
member_password FROM member_details WHERE member_id = $member_id";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            return $result->fetch_assoc();
        } else {
            return null;
        }
    }

    public function updateMemberDetails($member_id, $name, $email, $type,
$password = null) {
        if ($password) {
            // Update with password
            $sql = "UPDATE member_details SET
                        member_name = '$name',
                        member_email = '$email',
```

```
                    member_type = '$type',
                    member_password = '$password'
                WHERE member_id = $member_id";
        } else {
            // Update without password
            $sql = "UPDATE member_details SET
                    member_name = '$name',
                    member_email = '$email',
                    member_type = '$type'
                WHERE member_id = $member_id";
        }
        return $this->conn->query($sql);
    }
```

- View Book List

```
public function getAllBooks() {
        $books = [];
        $sql = "SELECT book_id, book_title, book_genre, book_author,
published_year FROM book_details";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $books[] = $row;
            }
        }
        return $books;
    }
```

- Request Book

```
public function addBookRequest($memberId, $bookTitle, $authorName,
$publishedYear) {
        $sql = "INSERT INTO borrow_return (member_id, book_id,
issue_status, return_status)
                SELECT $memberId, book_id, 'Not Issued', 'Not Returned'
                FROM book_details
                WHERE book_title = '$bookTitle' AND book_author =
'$authorName' AND published_year = '$publishedYear'";

        if ($this->conn->query($sql)) {
            return "Request sent successfully!";
        } else {
            return "Error: " . $this->conn->error;
        }
    }


public function fetchMemberRequests($memberId) {
```

```
        $sql = "SELECT br.br_id, b.book_title, br.issue_status,
br.return_status, br.issue_date, br.return_date
                FROM borrow_return br
                LEFT JOIN book_details b ON br.book_id = b.book_id
                WHERE br.member_id = $memberId";
        return $this->conn->query($sql);
    }
}
```

- return book

```
 // Fetch books available for return
    public function getBooksToReturn() {
        $books = [];
        $sql = "SELECT br.br_id, b.book_title, br.issue_date
                FROM book_details b
                JOIN borrow_return br ON b.book_id = br.book_id
                WHERE br.member_id = $this->member_id AND br.return_status
= 'Not Returned'";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $books[] = $row;
            }
        }
        return $books;
    }

// Update the borrow_return table
    public function returnBook($transaction_id, $mem_rt_date) {
        $sql = "SELECT issue_date, return_date FROM borrow_return WHERE
br_id = $transaction_id";
        $result = $this->conn->query($sql);

        if ($result && $row = $result->fetch_assoc()) {
            $issue_date = $row['issue_date'];
            $due_date = $row['return_date'];

            if ($mem_rt_date > $issue_date) {
                // Always update mem_rt_date regardless of due date
                $update_sql = "UPDATE borrow_return
                               SET mem_rt_date = '$mem_rt_date',
return_status = 'Returned'
                               WHERE br_id = $transaction_id";
                if ($this->conn->query($update_sql)) {
                    if ($mem_rt_date > $due_date) {
                        return "Book returned successfully, but the due
date was exceeded!";
                    }
                    return "Book returned successfully!";
                } else {
                    return "Error updating return status.";
                }
```

```
            } else {
                // If the return date is before or on the issue date
                return "Return date must be after the issue date!";
            }
        }
        return "Invalid transaction ID.";
    }
}
```

- check issued books

```
  public function fetchIssuedBooks($memberId) {

        $sql = "SELECT br.br_id, b.book_title, br.issue_status,
br.return_status, br.issue_date, br.return_date
                FROM borrow_return br
                JOIN book_details b ON br.book_id = b.book_id
                WHERE br.member_id = $memberId AND br.issue_status =
'Issued'";
        return $this->conn->query($sql);
    }

-- view fine reports
// Fetch the member report
$query = "SELECT member_name, member_report FROM member_details WHERE
member_id = $member_id";
$result = $conn->query($query);

if ($result && $result->num_rows > 0) {
    $member = $result->fetch_assoc();
} else {
    die("No report found for this member.");
}
?>
```

- view librarian Dashboard

```
public function fetchLibrarianDetails() {
        $sql = "SELECT * FROM librarian_details WHERE librarian_id = $this-
>librarianId";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            $this->librarianData = $result->fetch_assoc();
        } else {
            die("Librarian not found.");
        }
    }
```

- Librarian Log in

```
 $sql = "SELECT * FROM librarian_details WHERE librarian_name =
'$librarian_name'";
```

- Update Librarian Profile

```
 --Librarian class
     // Method to fetch librarian details by ID
    public function getLibrarianDetails($librarian_id) {
        $sql = "SELECT librarian_name, librarian_email, librarian_created
FROM librarian_details WHERE librarian_id = $librarian_id";
        $result = $this->conn->query($sql);
        if ($result && $result->num_rows > 0) {
            return $result->fetch_assoc();
        } else {
            return null;
        }
    }

    // Method to update librarian details
    public function updateLibrarianDetails($librarian_id, $name, $email,
$password = null) {
        if ($password) {
            // Update with password
            $sql = "UPDATE librarian_details SET
                        librarian_name = '$name',
                        librarian_email = '$email',
                        librarian_password = '$password'
                    WHERE librarian_id = $librarian_id";
        } else {
            // Update without password
            $sql = "UPDATE librarian_details SET
                        librarian_name = '$name',
                        librarian_email = '$email'
                    WHERE librarian_id = $librarian_id";
        }
        return $this->conn->query($sql);
    }
}
?>
```

- Add books

```
public function addBook($title, $genre, $author, $year) {
        // Insert into the database
        $sql = "INSERT INTO book_details (book_title, book_genre,
book_author, published_year)
                VALUES ('$title', '$genre', '$author', $year)";

        if ($this->conn->query($sql)) {
            return "Book added successfully!";
        } else {
            return "Error: " . $this->conn->error;
        }
    }
}
```

- Transaction

```
public function fetchAllTransactions() {
        $sql = "SELECT br.br_id, br.issue_date, br.return_date,
br.issue_status, br.return_status,
                        m.member_name, b.book_title
                FROM borrow_return br
                JOIN member_details m ON br.member_id = m.member_id
                LEFT JOIN book_details b ON br.book_id = b.book_id";
        $result = $this->conn->query($sql);
        return $result;
    }

    public function updateReturnDate($transactionId, $returnDate) {
        $sql = "UPDATE borrow_return SET return_date = '$returnDate' WHERE
br_id = $transactionId";
        return $this->conn->query($sql);
    }

    public function updateIssueStatus($transactionId, $statusValue) {
        $sql = "UPDATE borrow_return SET issue_status = '$statusValue',
issue_date = CURDATE() WHERE br_id = $transactionId";
        return $this->conn->query($sql);
    }
```

- admin dashboard

```
$sql = "SELECT admin_name FROM admin_details WHERE admin_id = $this-
>adminId";
```

- admin view member

```
 // Fetch all members from the database
    public function getAllMembers() {
        $query = "SELECT member_id, member_name, member_email, member_type
FROM member_details";
        return $this->conn->query($query);
    }
```

- admin view librarian

```
public function getAllLibrarians() {
        $query = "SELECT librarian_id, librarian_name, librarian_email FROM
librarian_details";
        return $this->conn->query($query);
    }

    public function deleteLibrarian($librarianId) {
        // Finally, delete the librarian from librarian_details
        $query = "DELETE FROM librarian_details WHERE librarian_id =
$librarianId";
        return $this->conn->query($query);
    }
```

- Admin log in

```
$sql = "SELECT * FROM admin_details WHERE admin_name = '$admin_name'";
```

- Admin generate report

```
// Update the member_report column for the given member_id
        $sql = "UPDATE member_details SET member_report = '$report' WHERE
member_id = $memberId";
```

- Admin add librarian

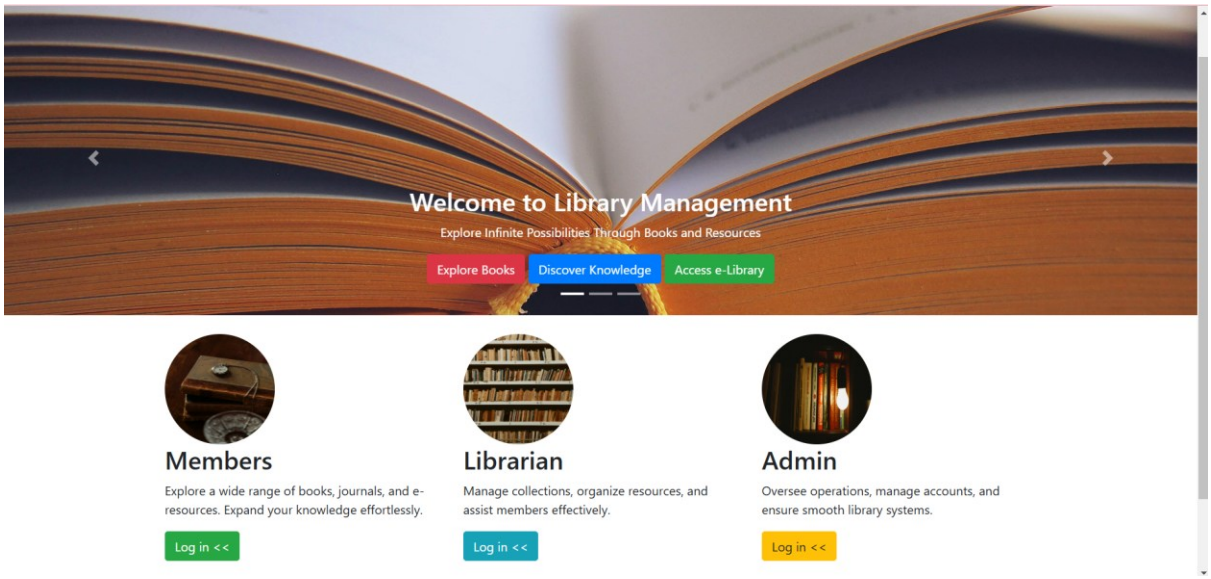
```
public function add($name, $email, $password) {

        $sql = "INSERT INTO librarian_details (librarian_name,
librarian_email, librarian_password)
                VALUES ('$name', '$email', '$password')";
```

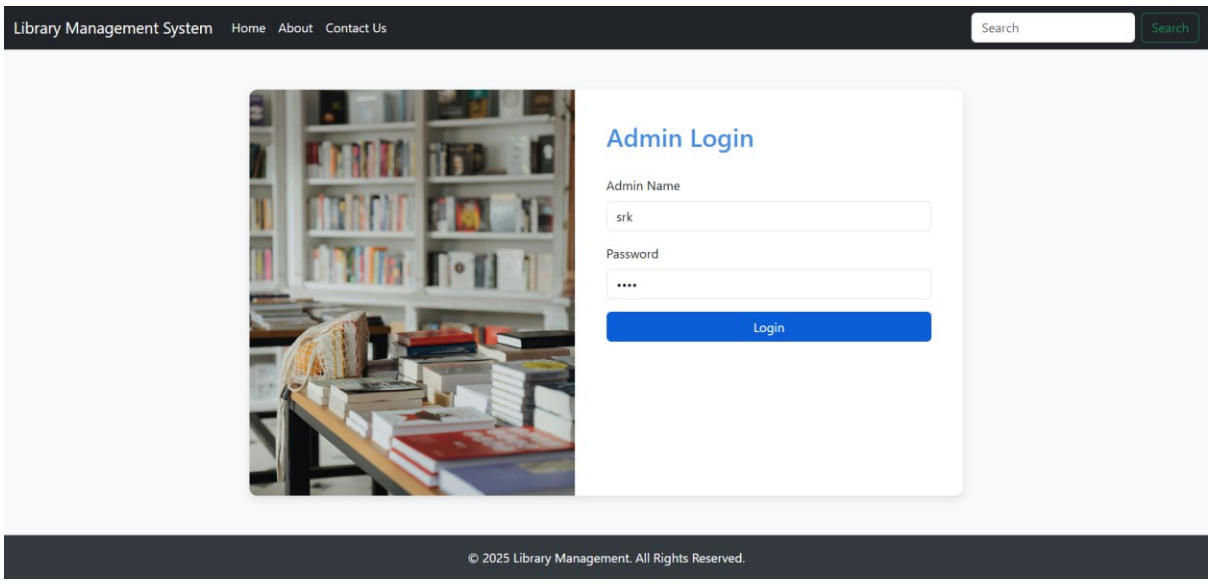
10. Appendix

Sample Screens

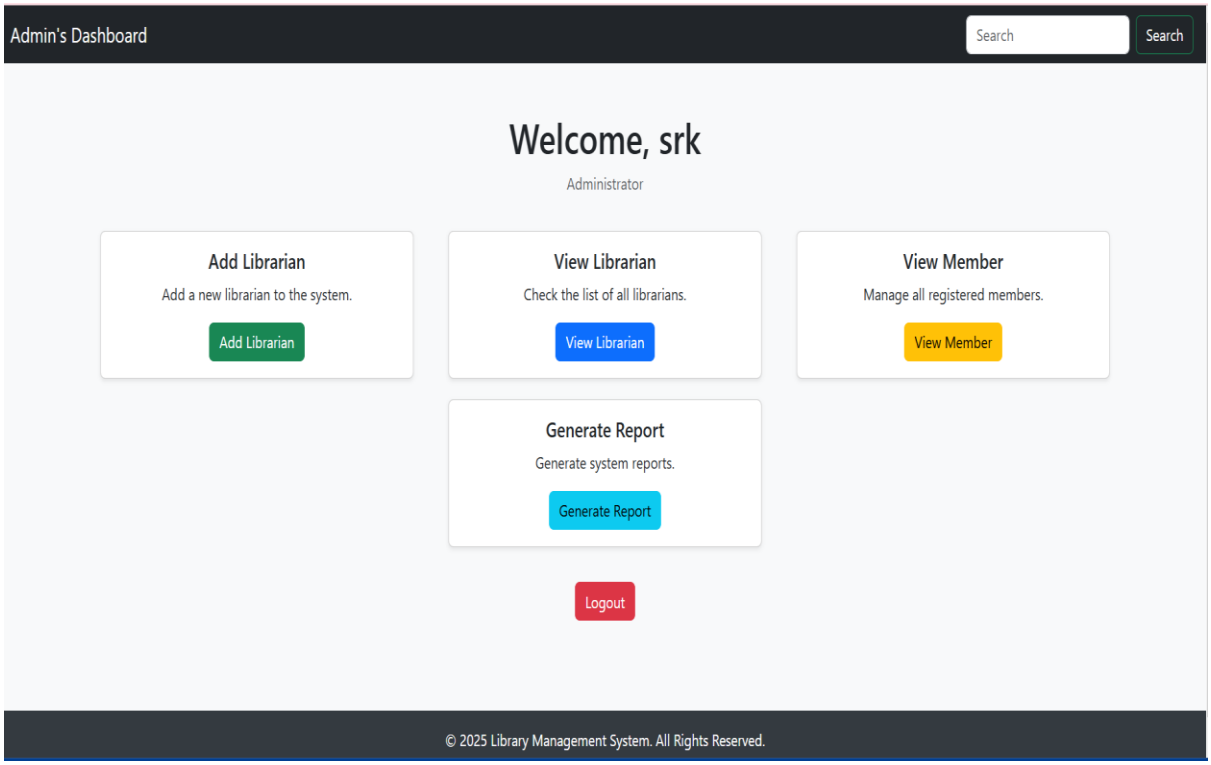
- ***Welcome page***



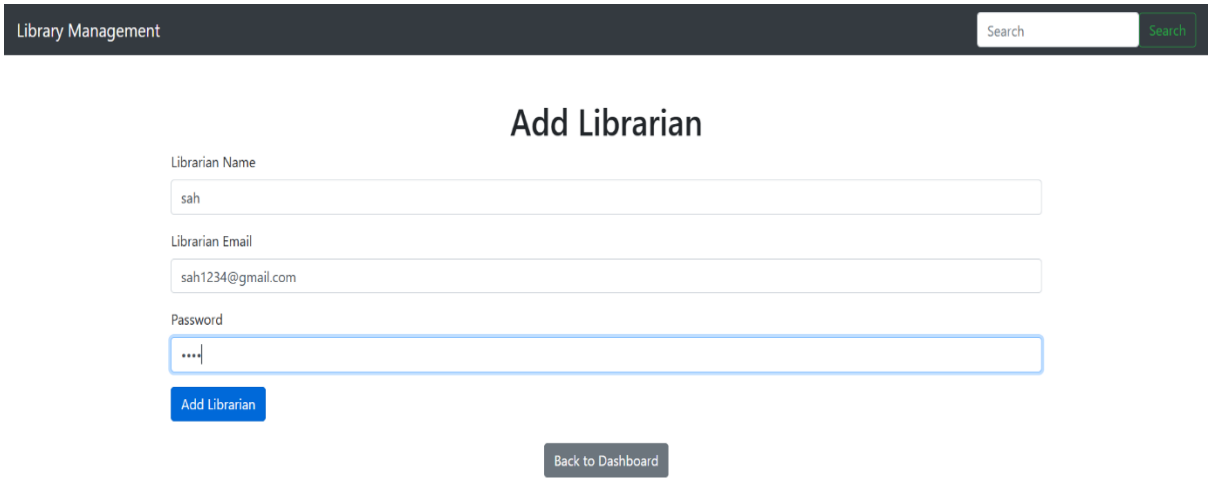
- ***Admin Login Page***



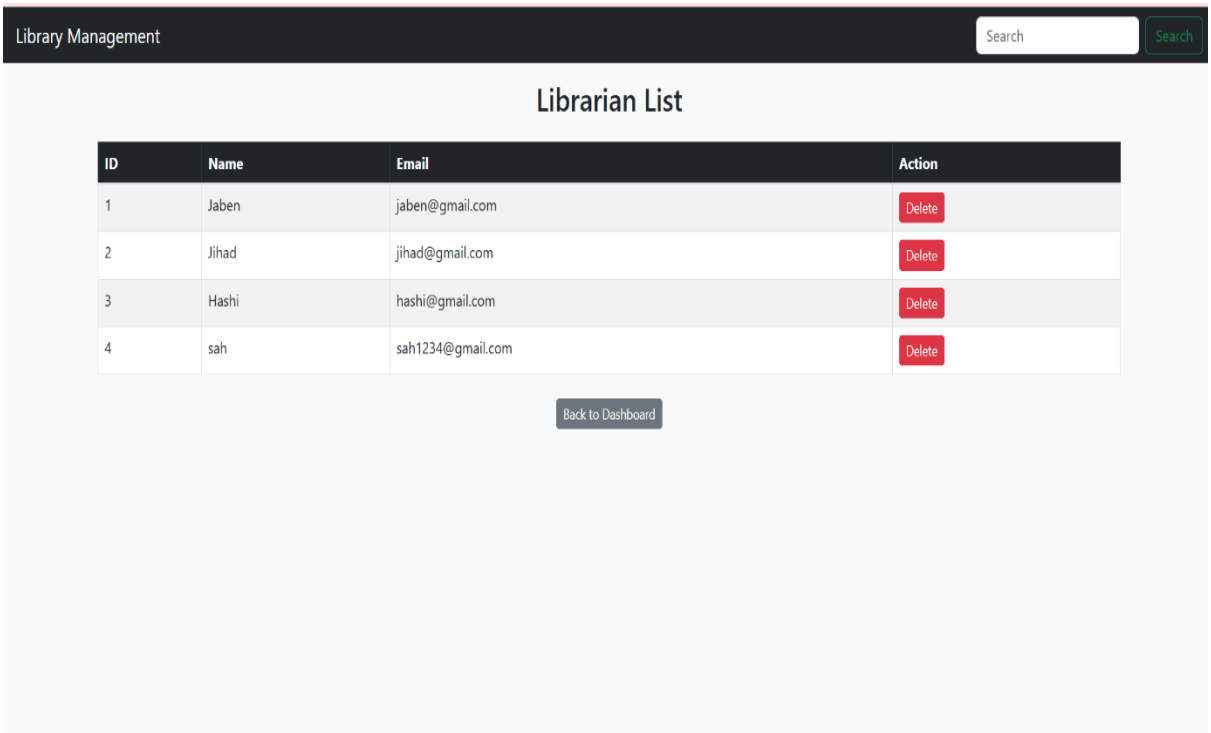
- ***Admin Dashboard***



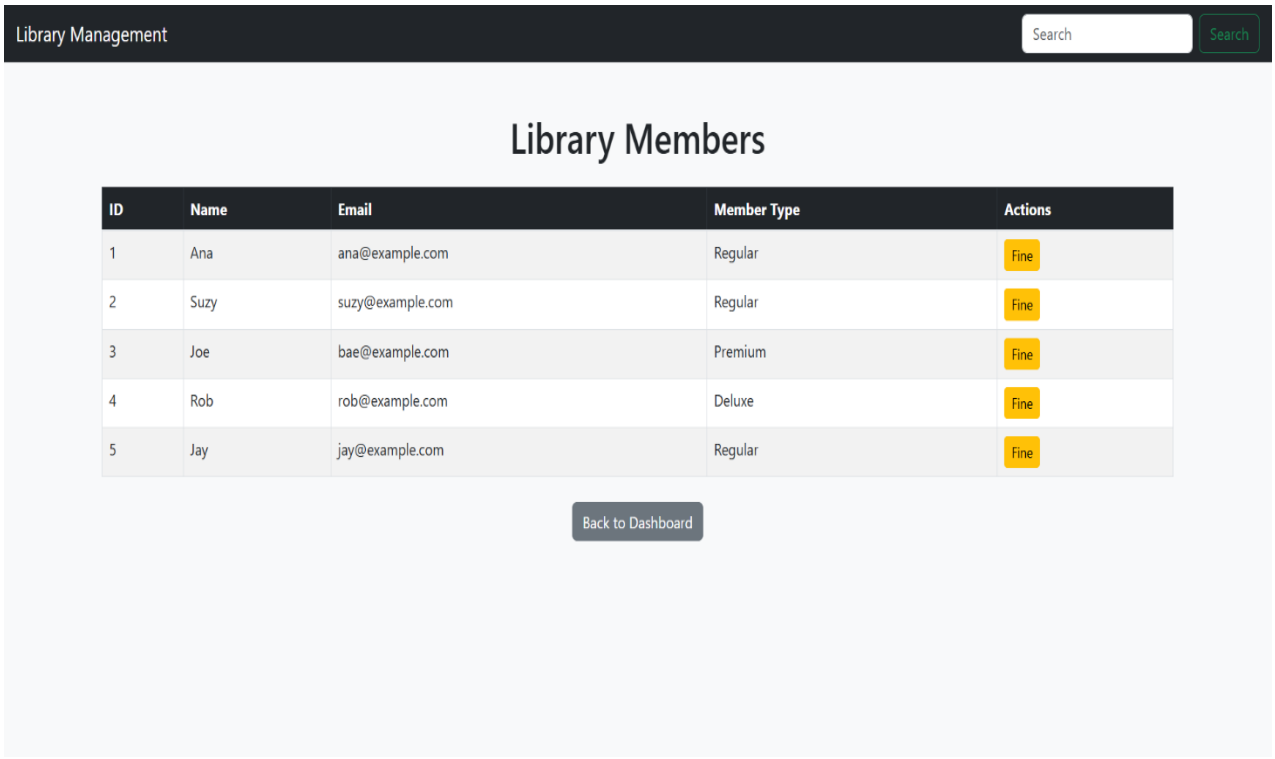
➢ ***Add Librarian***



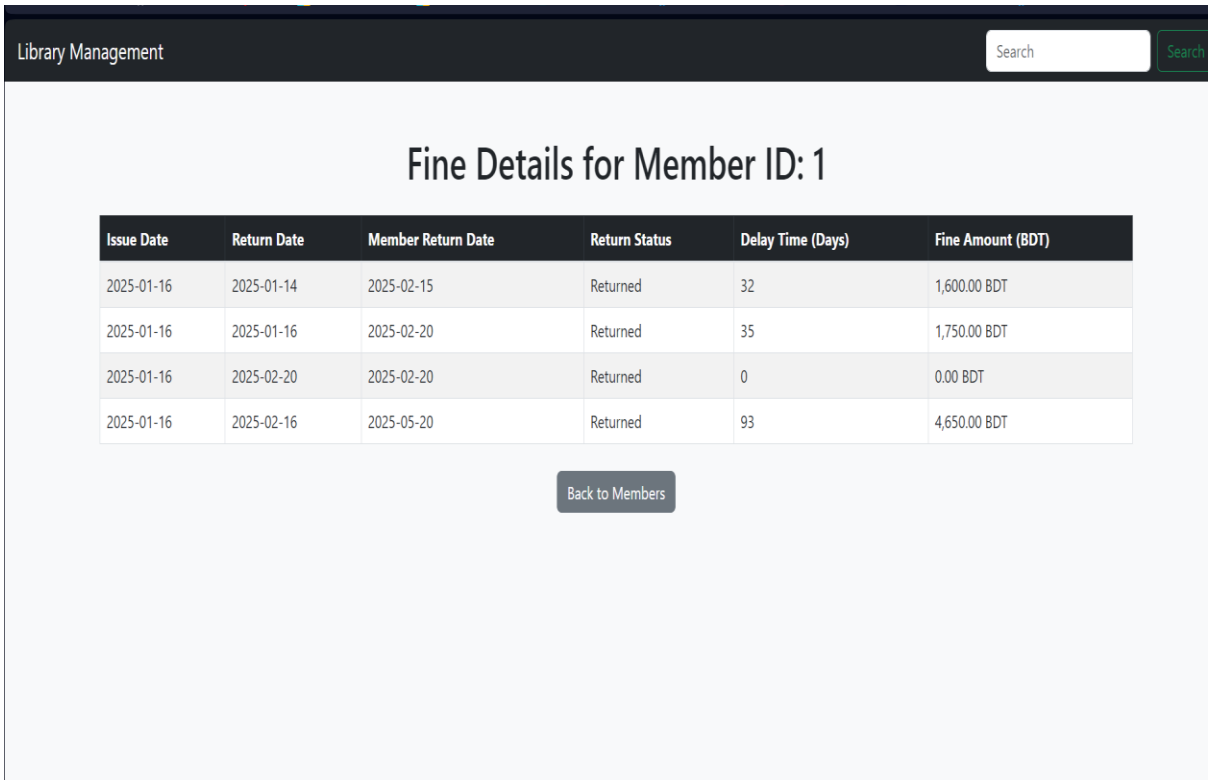
➢ ***View librarian***



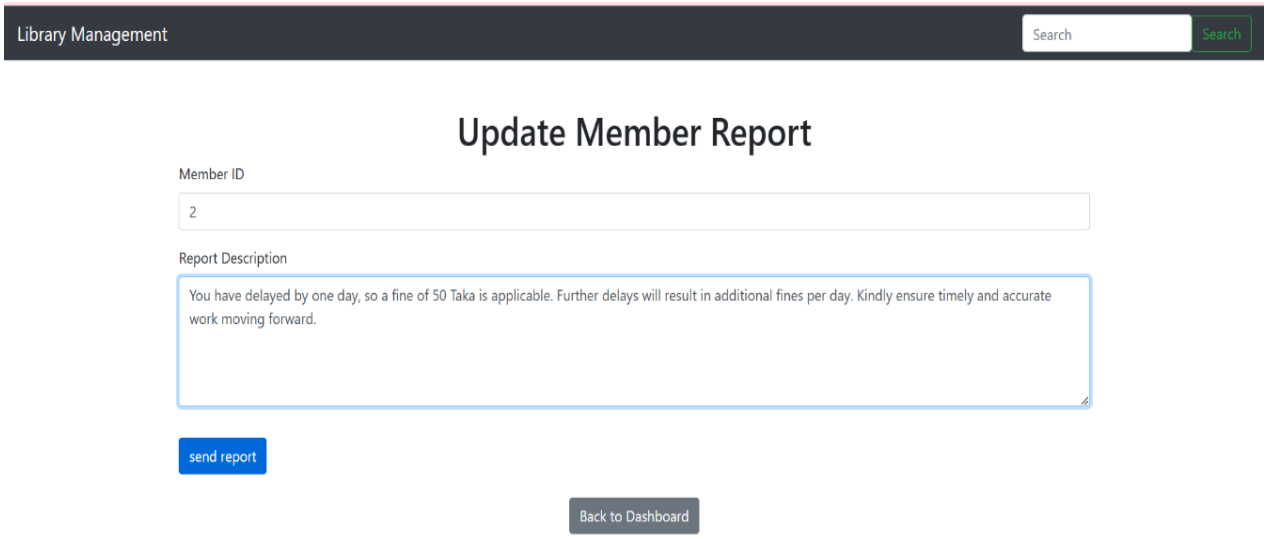
- ***View Members***



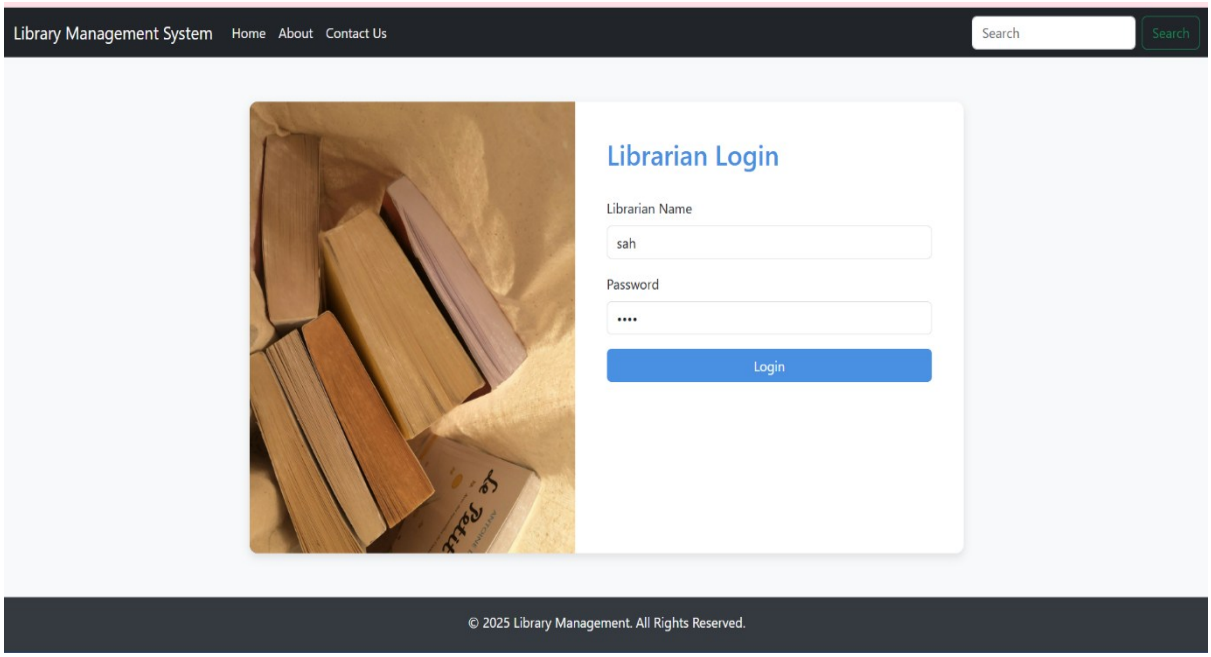
- ***Fine page***



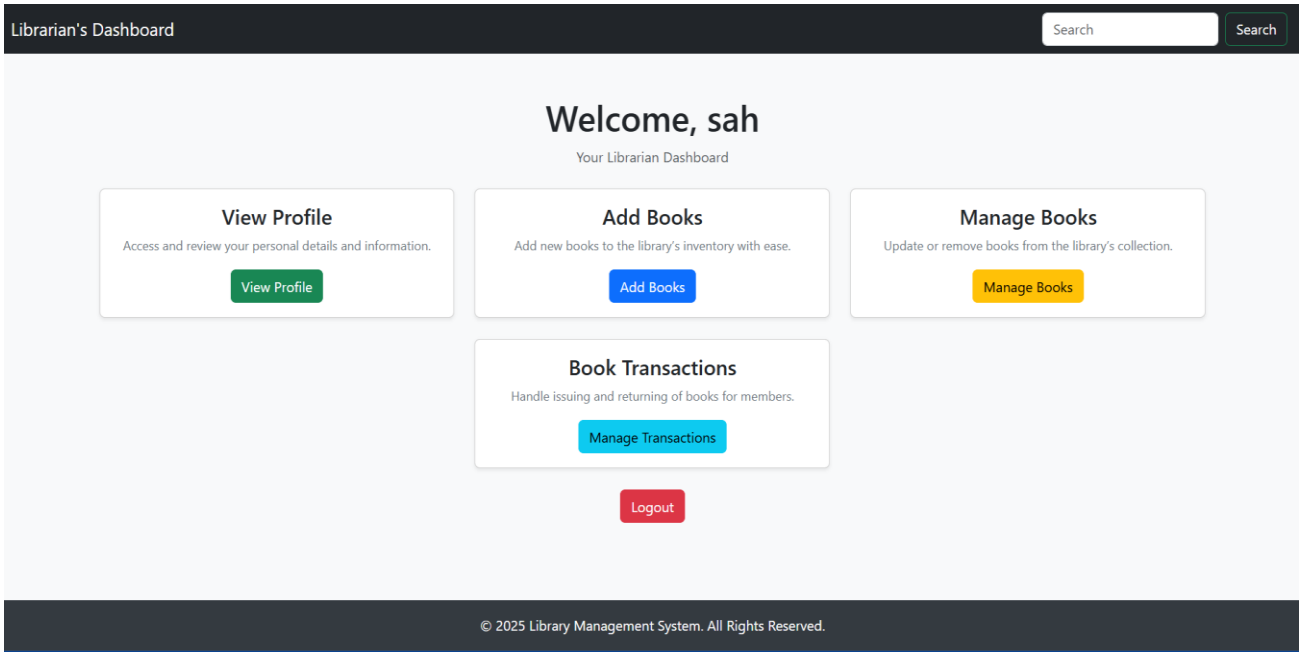
➢ ***Generate Report***



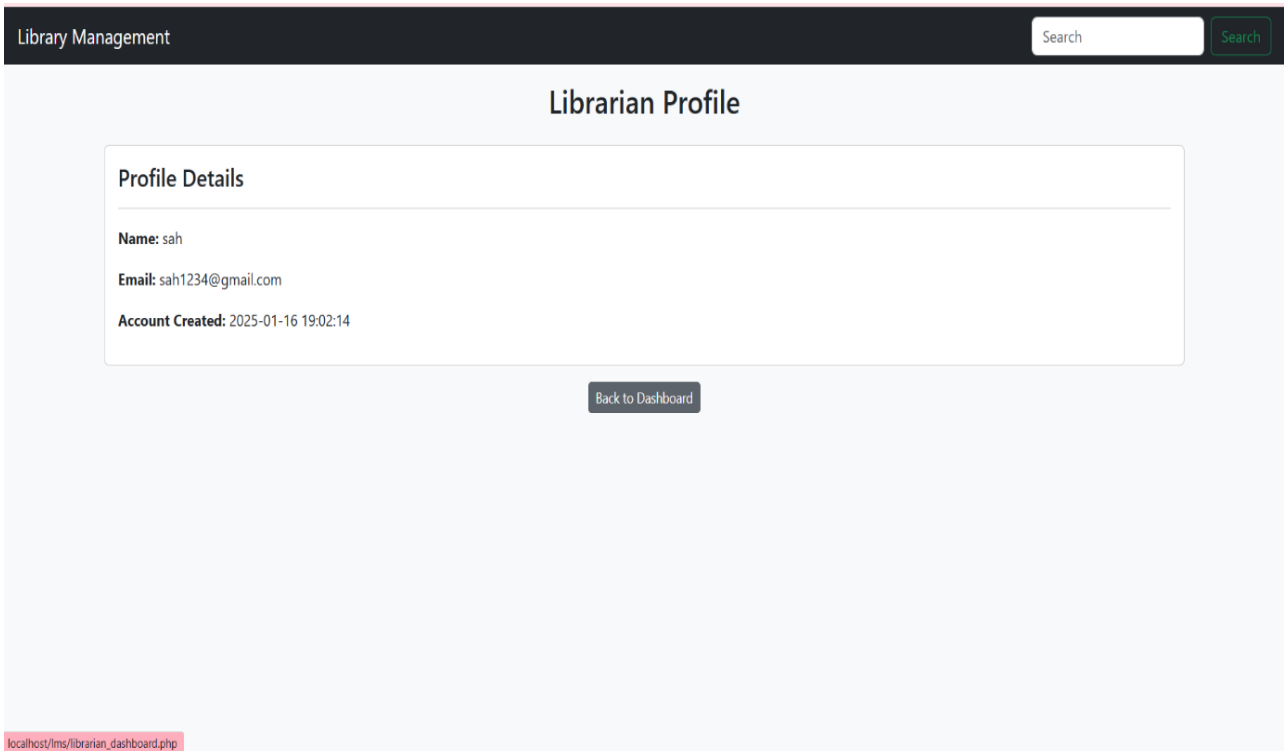
➢ ***Librarian Login Page***



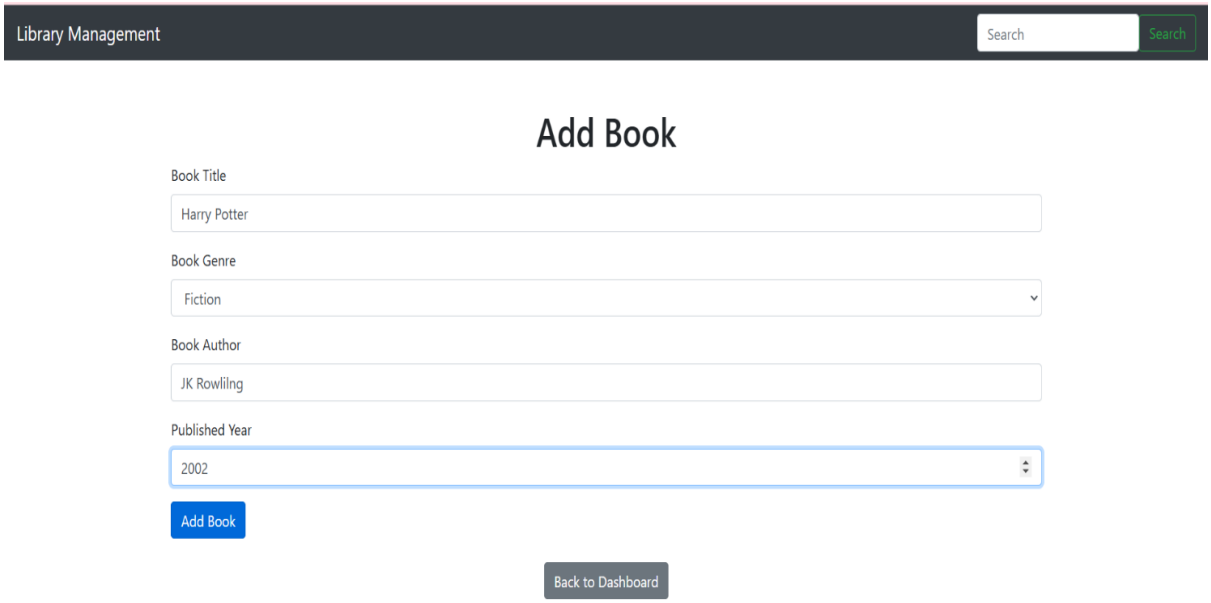
➢ ***Librarian Dashboard***



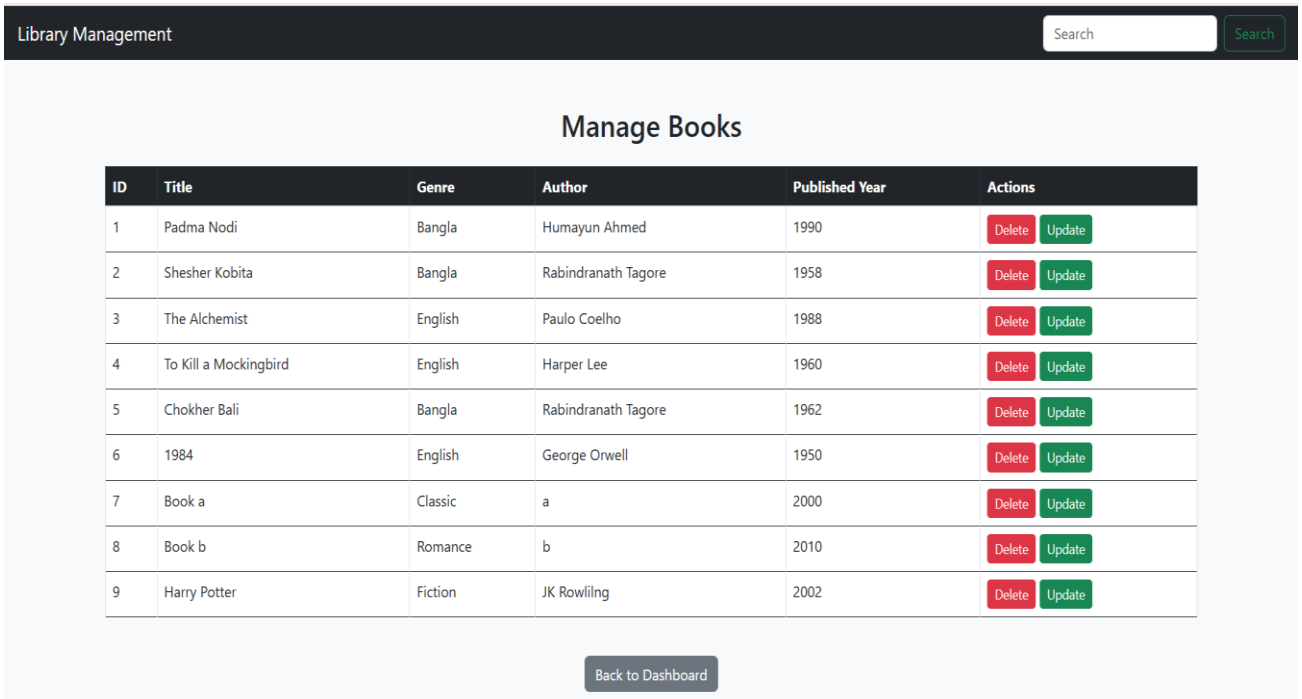
➢ ***View profile***



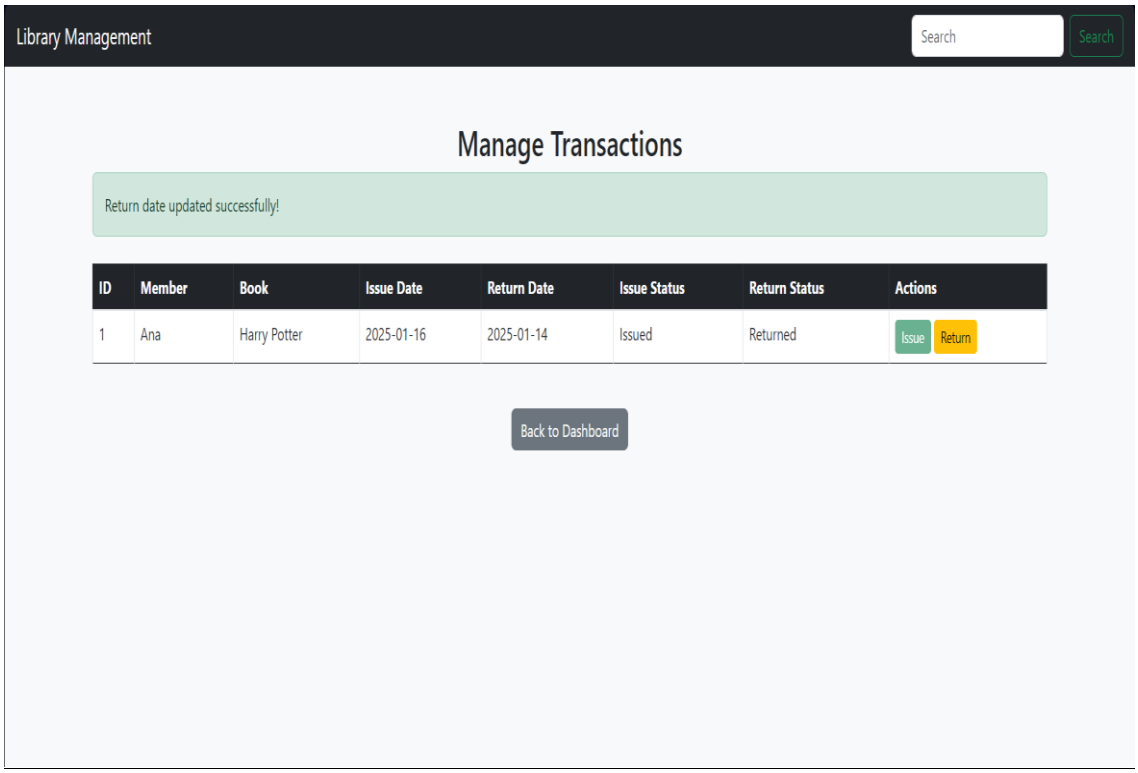
➢ ***Add Book***



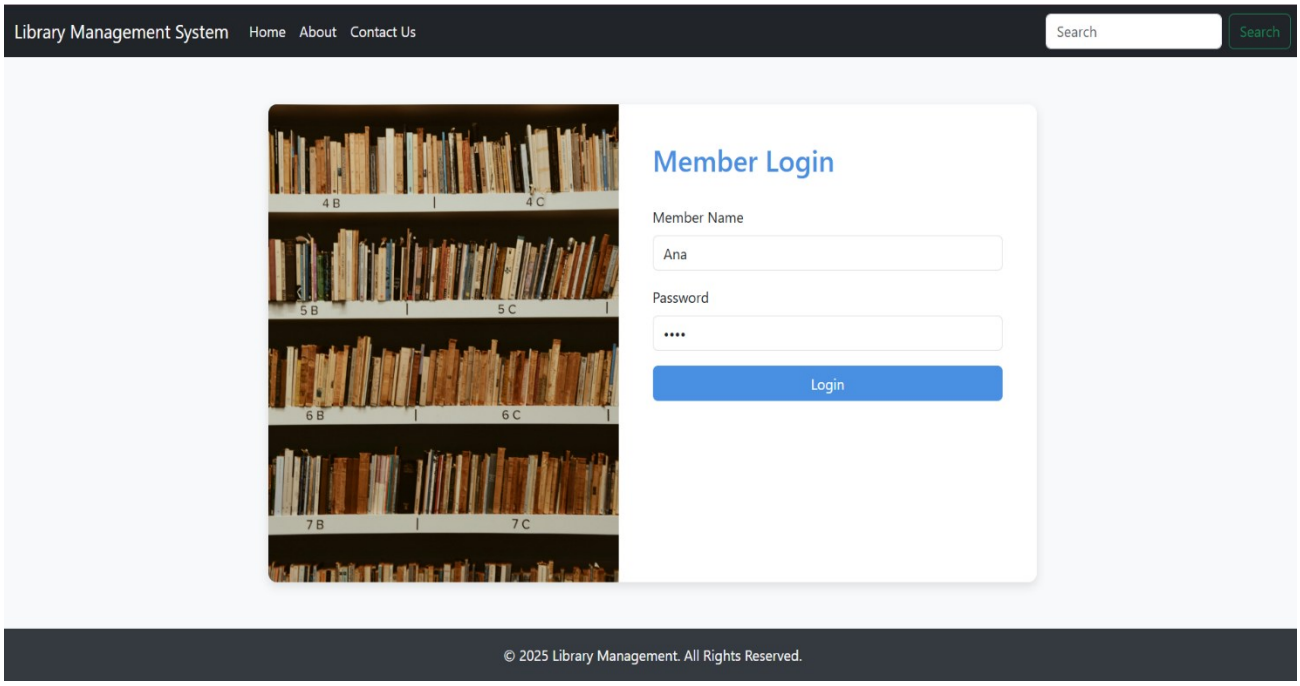
➢ ***Manage Books***



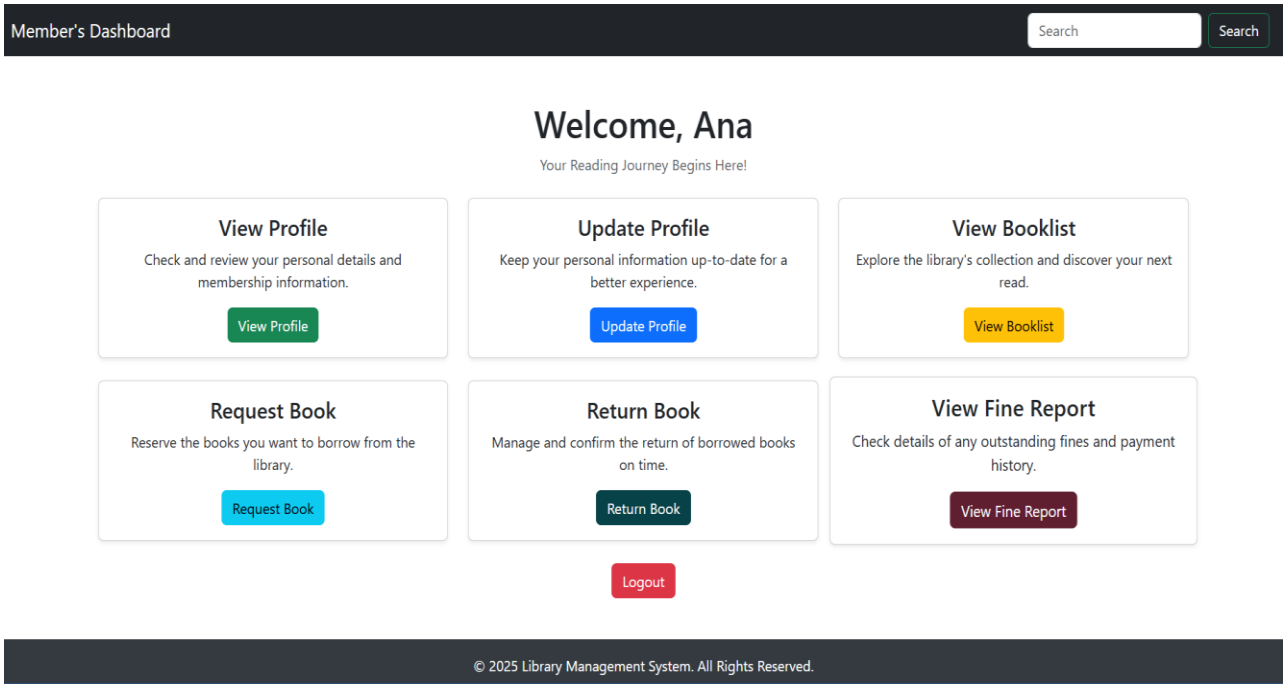
➢ ***Manage Transaction***



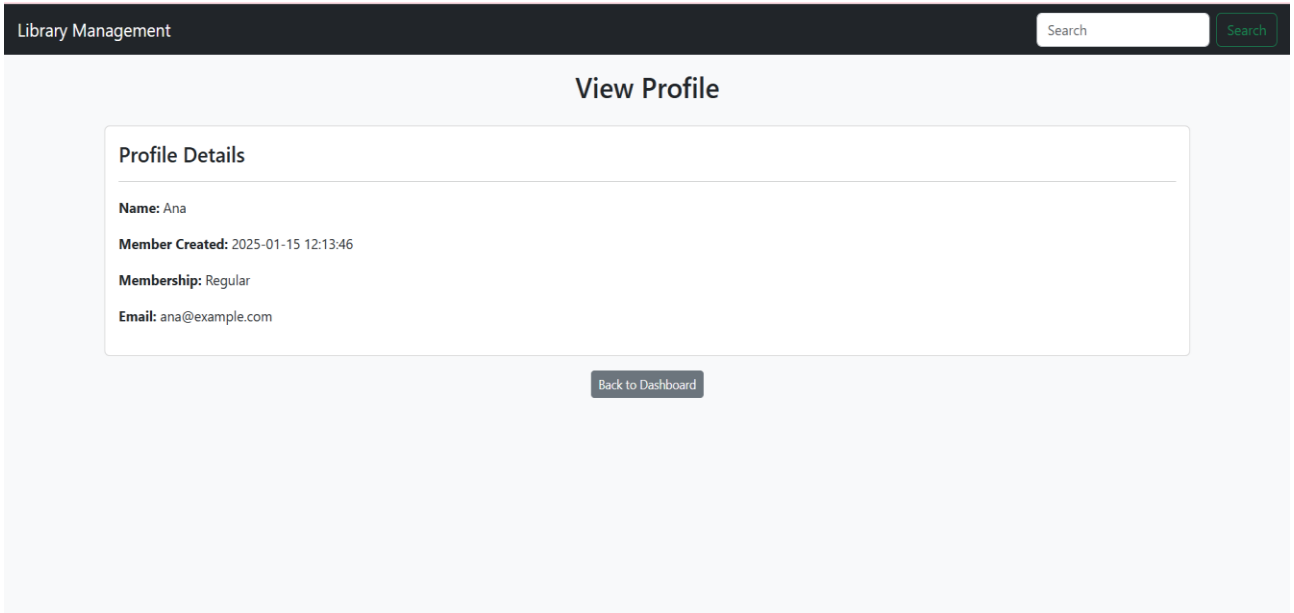
➢ ***Member Login Page***



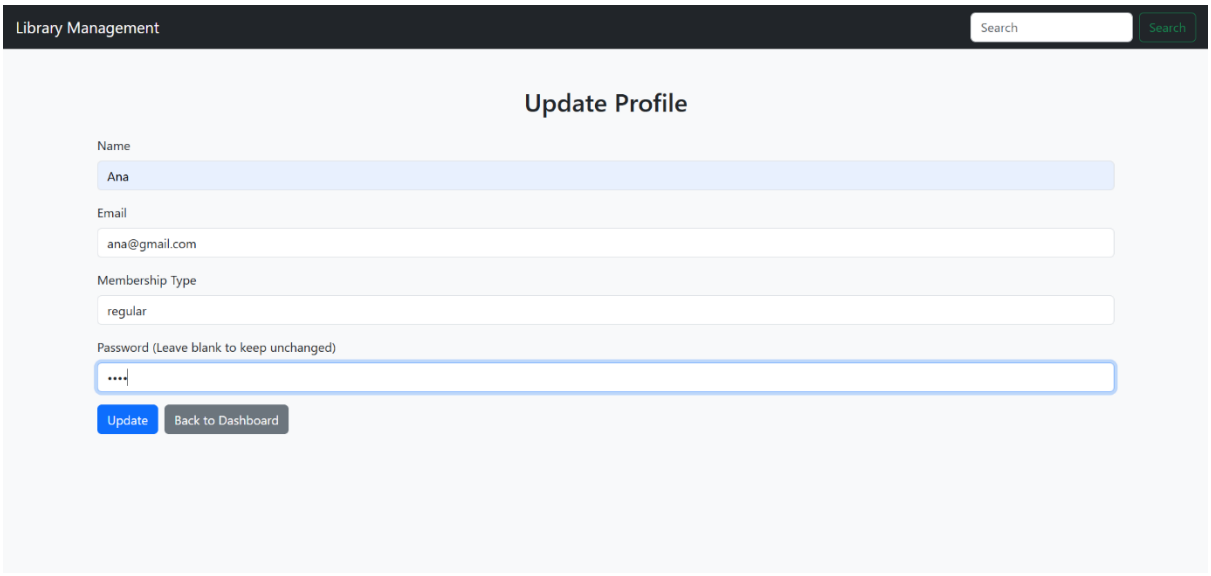
➢ ***Member Dashboard***



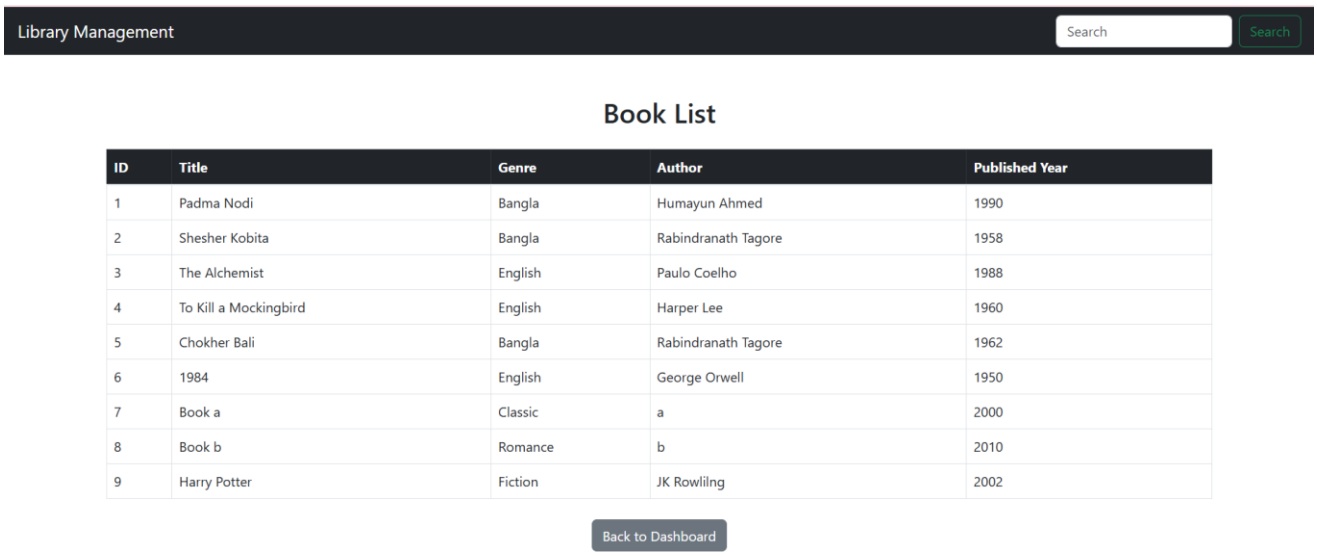
➢ ***View Profile***



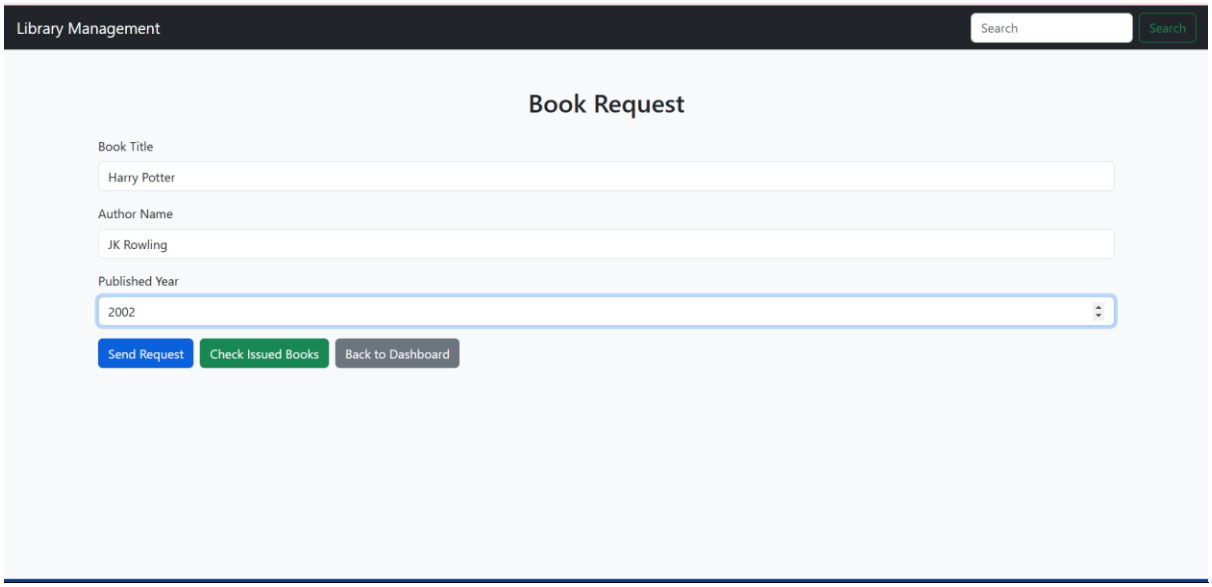
➢ *Update Profile*



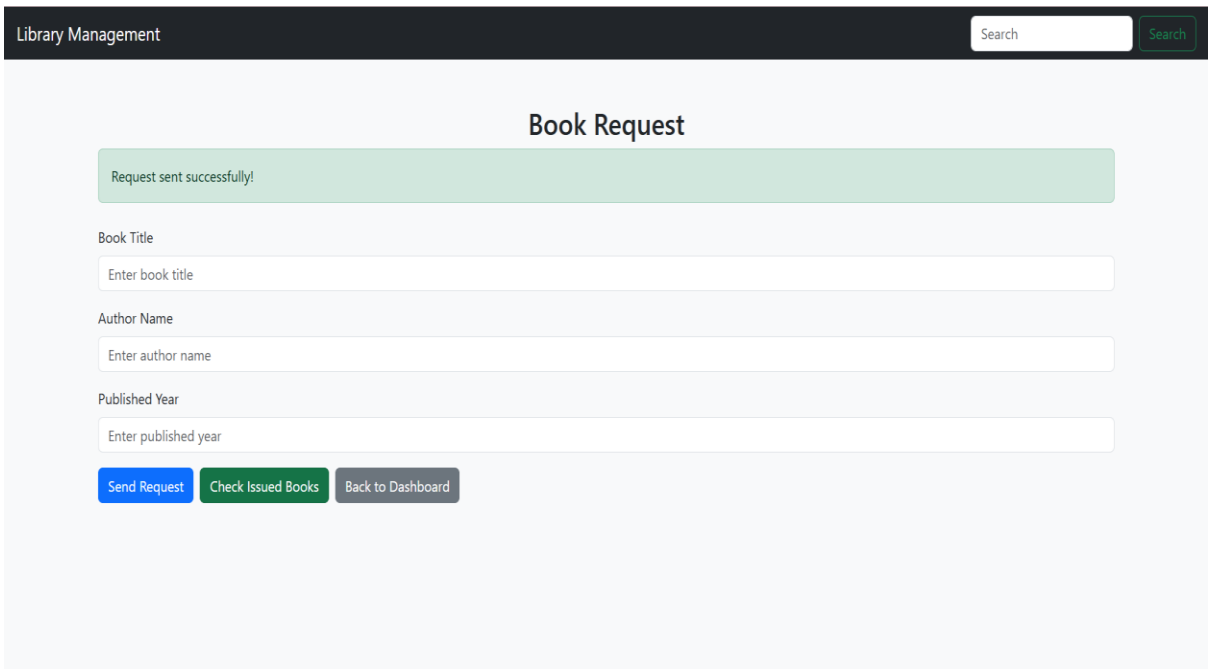
➢ *View Booklist*



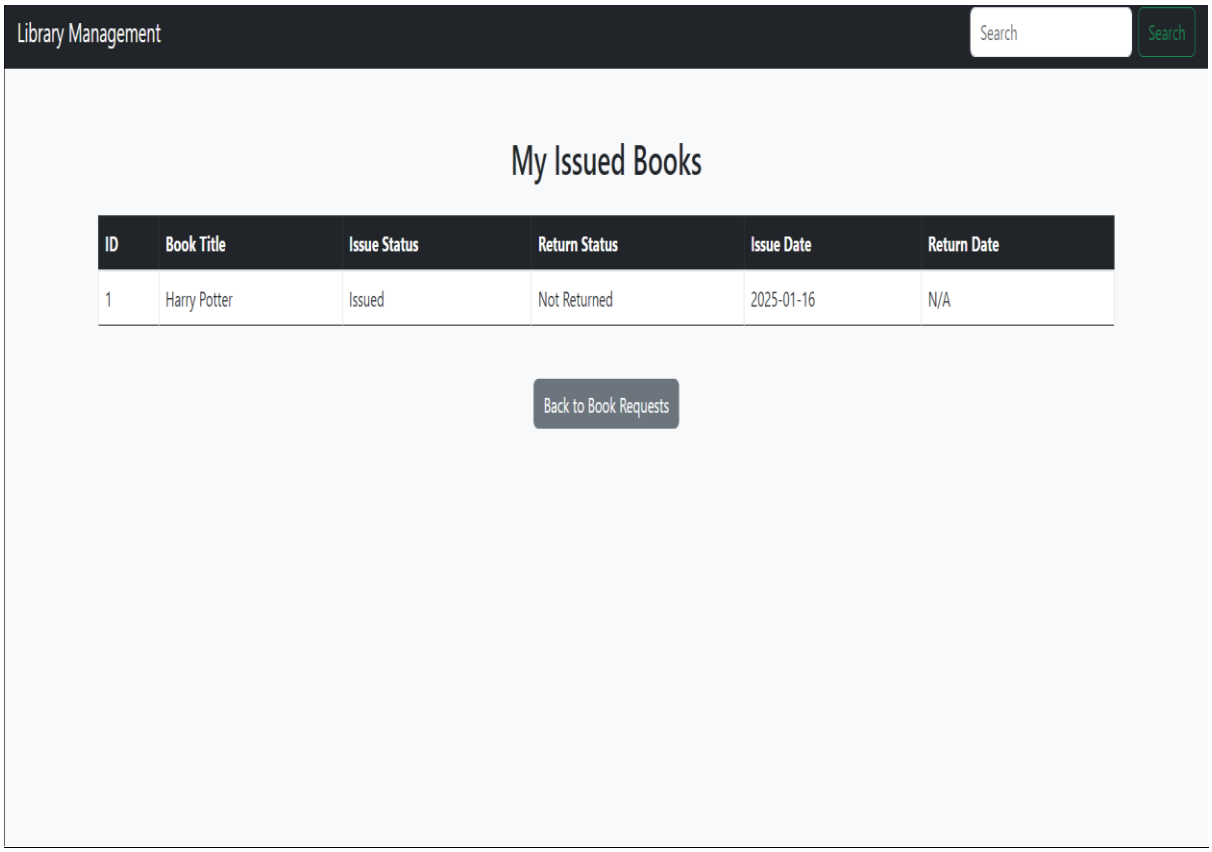
➢ ***Request Book***



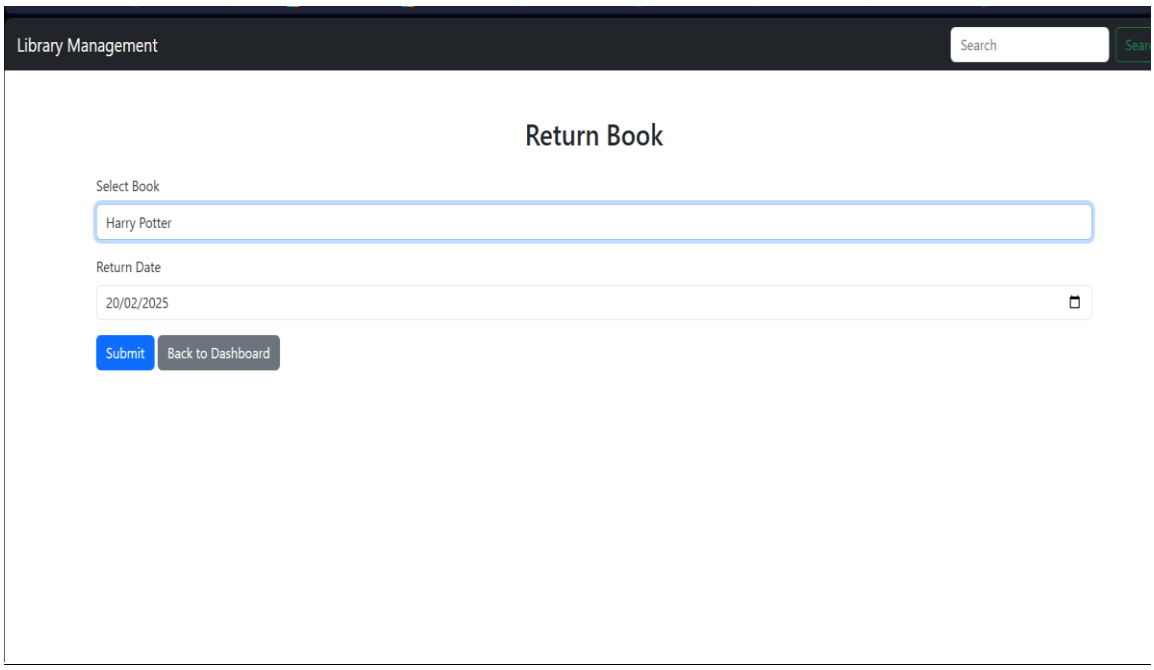
➢ ***Request Send***



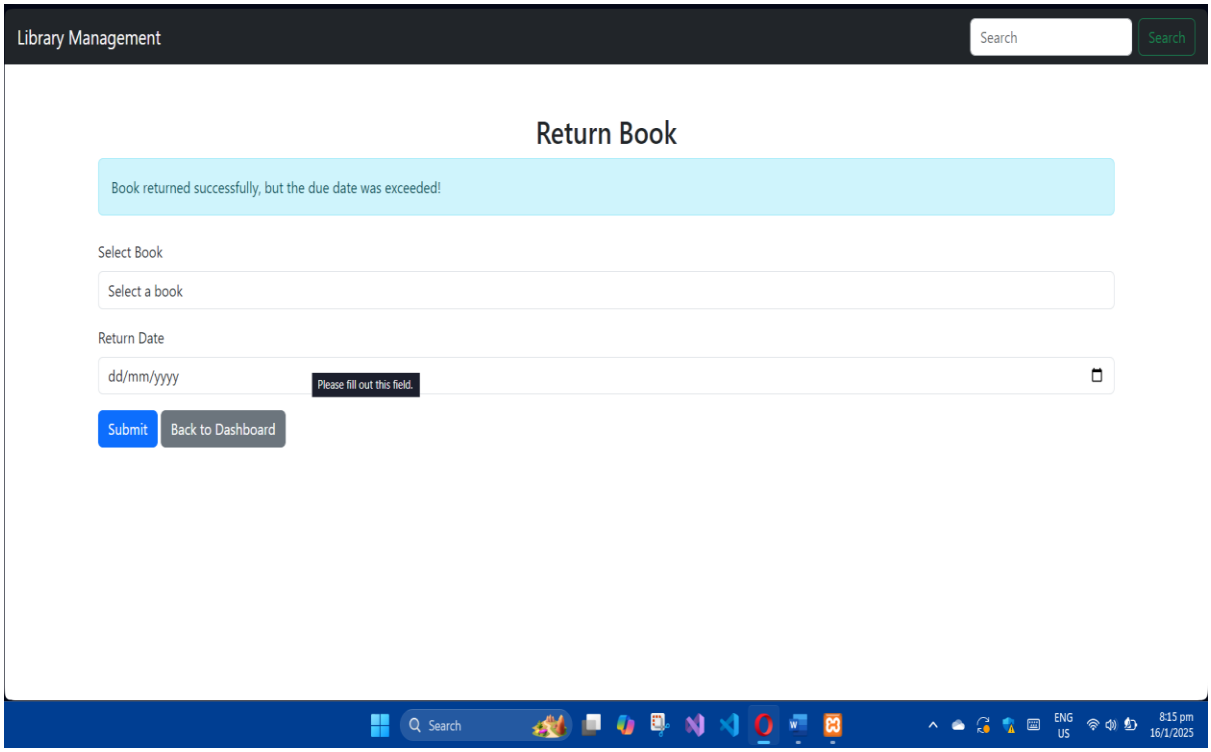
➢ ***Check Issued Books***



➢ ***Return Book***



➢ ***Book Returned After Due Date***



➢ ***View Fine Report***