

***Mini project report on***

Exam Centre Management System

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**UE22CS351A – DBMS Project**

***Submitted by:***

|  |  |  |
| --- | --- | --- |
| **AAKANKSH SEELIN**  **ADITI ROOPESH MIRJI** | **PES2UG22CS003**  **PES2UG22CS032** |  |

Under the guidance of

**Dr. Suja**

|  |
| --- |
| PES University |

**AUG - DEC 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

**CERTIFICATE**

*This is to certify that the mini project entitled*

**University Fest Management System**

*is a bonafide work carried out by*

|  |  |
| --- | --- |
| **Aakanksh Seelin**  **Aditi Roopesh Mirji** | **PES2UG22CS003**  **PES2UG22CS032** |

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2024 – DEC. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

|  |  |
| --- | --- |
| Signature  Dr Suja |  |

**DECLARATION**

We hereby declare that the DBMS Project entitled **University Fest Management System** has been carried out by us under the guidance of **Dr Suja** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2024.

|  |  |  |
| --- | --- | --- |
| **Aakanksh Seelin**  **Aditi Roopesh Mirji** | **PES2UG22CS003**  **PES2UG22CS032** | **Aakanksh.S** |

**ABSTRACT**

The Exam Centre Management System is a comprehensive system that aims to streamline a university library operation, manage resources effectively and provide an interactive interface for users. This system aims to replace traditional methods with an efficient, user-friendly digital solution to simplify the day-to-day functioning of an exam centre. The Exam Management System project is a practical integration of Database Management Systems (DBMS) and Software Engineering concepts, designed to address the real-world challenges of managing a library. The primary focus of this project is to create a software application that manages library operations, with an emphasis on database management through CRUD (Create, Read, Update, Delete) operations. The system will be developed as a web-based application that interacts with a relational database.

The objectives of the project are:

• Implement and manage a relational database for storing and retrieving information related to books, members, and transactions.

• Enable seamless creation, reading, updating, and deletion of database entries directly from the application interface.

• Provide functionalities for recording, issuing, and returning books.

• Create a user-friendly interface that simplifies interactions with the database for both library staff and members.

• Ensure the integrity and security of data through verified access controls.

# 

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
|  | **INTRODUCTION** | **6** |
|  | **PROBLEM DEFINITION WITH USER REQUIREEMNT SPECIFICATIONS** | **7** |
|  | **LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED** | **10** |
|  | **ER MODEL** | **11** |
|  | **ER TO RELATIONAL MAPPING** | **12** |
|  | **DDL STATEMENTS** | **13** |
|  | **DML STATEMENTS (CRUD OPERATION SCREENSHOTS)** | **17** |
|  | **QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)** | **23** |
|  | **STORED PROCEDURE, FUNCTIONS AND TRIGGERS** | **27** |
|  | **FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)** | **35** |
| **REFERENCES/BIBLIOGRAPHY** | | **44** |
| **APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS** | | **45** |
|  | |  |

**1)INTRODUCTION**

The Exam Centre Management System will provide a user-friendly interface for managing a library’s book inventory, member registrations , and borrowing/returning activities. The system will allow library administrators to perform CRUD operations on the database tables that store information about books, members, and transactions. The system is a multi-user system which will be primarily used by library administrators , but can be accessed by members as well.

The features of the project are:

• Verified Login – Authorized access to the application. Admin, or existing members can seamlessly login to the application. Existing option to add a new member and create a new profile. Primary Login for Admins Only.

• Book Management- Operations to add, search, update, and delete book records in the database.

• Member registration and profile management- Features to manage member records, including registration, updating details, and deletion.

• Book loan, return, and reservation functionality- Track the status of each book. Note down date of borrowing and expected date of return.

• Overdue alerts and fine calculation- If any borrowed book is not returned within the expected date , then display an overdue alert and calculate the expected fine based on the number of exceeded days.

2) **PROBLEM DEFINITION WITH USER REQUIREEMNT SPECIFICATIONS**

The Exam Centre Management System is a comprehensive system that aims to streamline a university library operation, manage resources effectively and provide an interactive interface for users. This system aims to replace traditional methods with an efficient, user-friendly digital solution to simplify the day-to-day functioning of an exam centre.

1. **Functional Requirements:**

**1.1)Authentication**

* The system shall allow authorized users to login to the application.
* The system shall restrict access to administrative functionalities to only authorized admins.

**1.2)Book Management**

* The system shall allow administrators to add new book records to the database.
* The system shall allow administrators and members to view a list of all books in the library.
* The system shall allow administrators to update the details of existing book records.
* The system shall allow administrators to delete book records from the database.
* The system shall allow members to borrow and return books.

**1.3)Member Management**

* The system shall allow administrators to add new member records to the database.
* The system shall allow administrators to view a list of all members.
* The system shall allow administrators to update the details of existing member records.
* The system shall allow administrators to delete member records from the database

**1.4)Transaction Management**

* The system shall allow administrators to record the borrowing of a book by a member.
* The system shall allow administrators to record the return of a borrowed book.
* The system shall allow administrators to view a list of all borrowing and return transactions.
* The system shall allow admins to calculate fines in case of delay in return of book.

**2)Non-Functional Requirements:**

**2.1. Usability**

* The system shall provide a simple and user-friendly graphical user interface (GUI) that can be easily navigated by users with basic computer skills.

**2.2. Security**

* The system shall restrict access to administrative functionalities to authorized users only.
* The system shall require user authentication for access to the application.
* The system shall restrict access to the member portal to verified members only.

**2.3. Maintainability**

* The system shall be designed in a modular manner, allowing easy updates and maintenance of individual components.

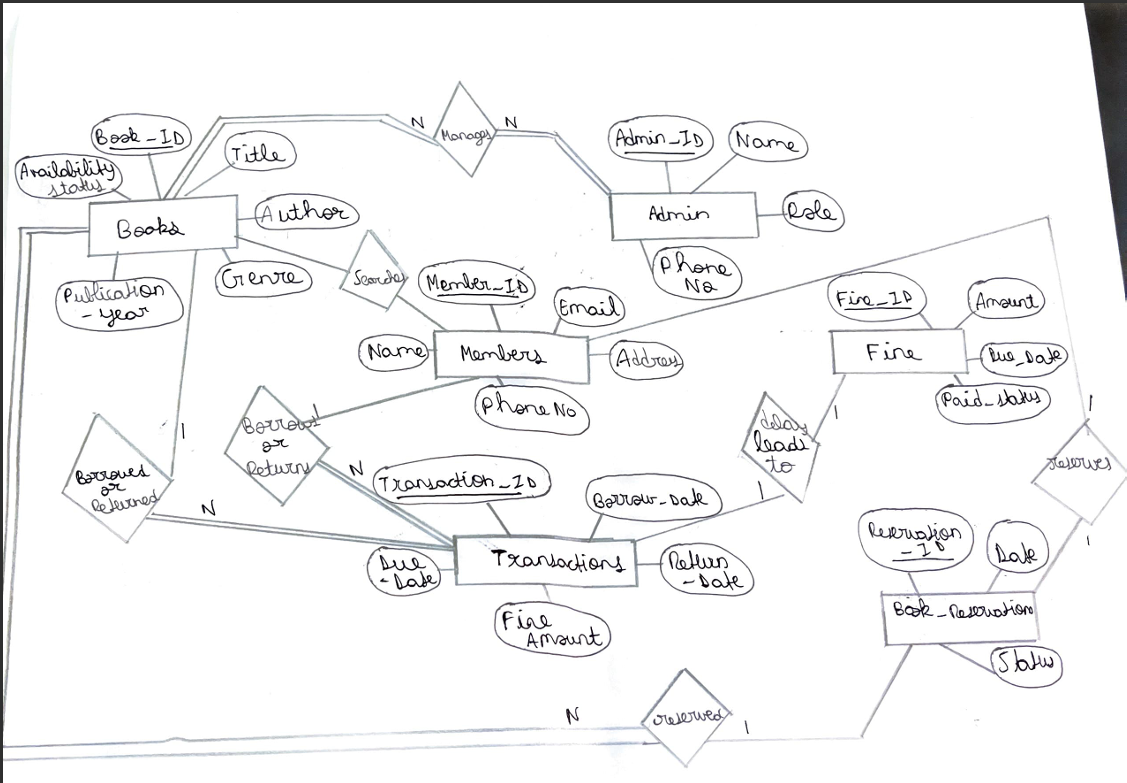
**2.4. Reliability**

* The system shall ensure data integrity and reliability, particularly during database transactions, to prevent data loss or corruption.

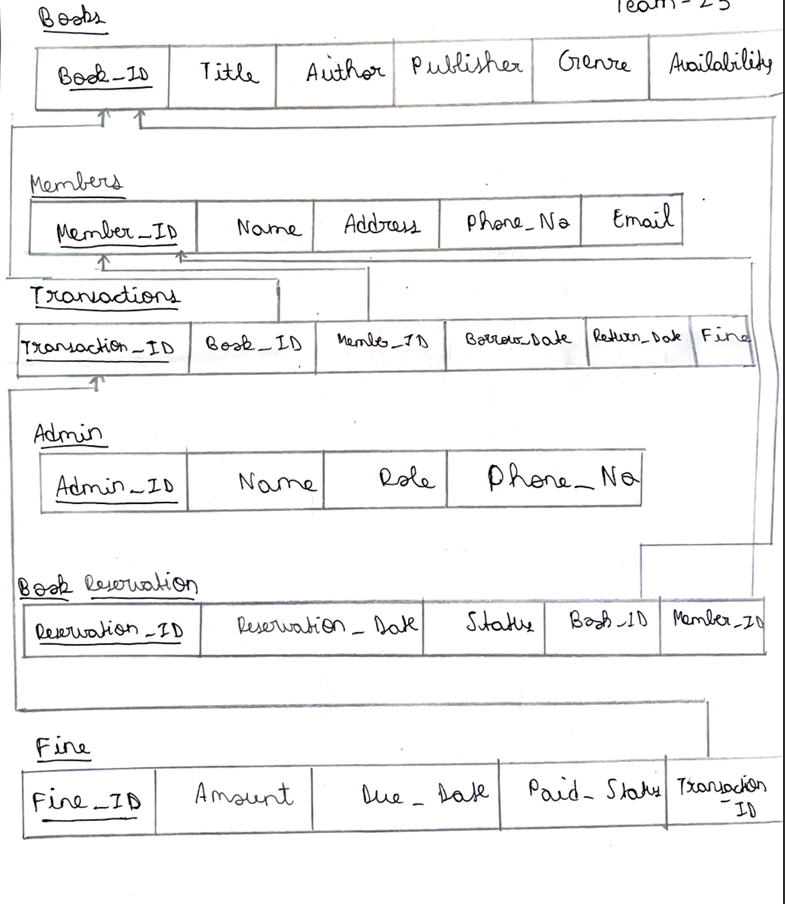
**3)LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED**

* **Planning Tool:** Google Sheets- for initial planning, task distribution, and creating simple timelines.
* **Design Tool:** smartdraw – for creating flowcharts, ER diagrams, and other design diagrams; Figma – for creating a wireframe of the UI if required.
* **Version Control:** Git & GitHub- for managing our source code, maintaining versions, and collaboration.
* **Development Tool:** Python (in VS Code) with Streamlit for quick iterations on the frontend; MySQL for backend development.
* **Bug Tracking:** GitHub Issues (for tracking bugs and feature requests as they emerge).
* **Testing Tool:** PyTest for unit testing in Python; Selenium for web application testing (if required).

**4) ER MODEL**

****

**5) ER TO RELATIONAL MAPPING**

****

**6)DDL STATEMENTS**

The DDL commands have been used in 27 different instances in the project.

**DROP DATABASE IF EXISTS lib\_mgmt;**

* This command drops the lib\_mgmt database if it already exists.

**CREATE DATABASE lib\_mgmt;**

* This command creates a new database named lib\_mgmt.

**USE lib\_mgmt;**

* This command sets the current database to lib\_mgmt.

**CREATE TABLE Authors (...)**

* This command creates a table named Authors with the specified columns and constraints.

**CREATE TABLE Categories (...)**

* This command creates a table named Categories with the specified columns and constraints.

**CREATE TABLE Books (...)**

* This command creates a table named Books with the specified columns and constraints.

**CREATE TABLE Administrators (...)**

* This command creates a table named Administrators with the specified columns and constraints.

**CREATE TABLE Members (...)**

* This command creates a table named Members with the specified columns and constraints.

**CREATE TABLE MemberTransactions (...)**

* This command creates a table named MemberTransactions with the specified columns and constraints.

**CREATE TABLE AdminTransactions (...)**

* This command creates a table named AdminTransactions with the specified columns and constraints.

**CREATE PROCEDURE CheckOutBook(...)**

* This command creates a stored procedure named CheckOutBook to handle the book checkout process.

**CREATE PROCEDURE ReturnBook(...)**

* This command creates a stored procedure named ReturnBook to handle the book return process.

**CREATE PROCEDURE BorrowBook(...)**

* This command creates a stored procedure named BorrowBook to handle the member book borrowing process.

**CREATE PROCEDURE ReturnBook(...)**

* This command creates a stored procedure named ReturnBook to handle the member book return process.

**CREATE OR REPLACE VIEW BookListView AS ...**

* This command creates a view named BookListView that provides a consolidated view of book information.

**CREATE OR REPLACE VIEW TransactionDetailsView AS ...**

* This command creates a view named TransactionDetailsView that provides a consolidated view of transaction details.

**CREATE PROCEDURE DeleteBook(...)**

* This command creates a stored procedure named DeleteBook to handle the deletion of books.

**CREATE TABLE MemberBorrowingSummary (...)**

* This command creates a table named MemberBorrowingSummary to store member borrowing statistics.

**CREATE TRIGGER after\_member\_insert ...**

* This command creates a trigger named after\_member\_insert that is executed after a new member is inserted into the Members table.

**CREATE TRIGGER after\_transaction\_insert ...**

* This command creates a trigger named after\_transaction\_insert that is executed after a new transaction is inserted into the MemberTransactions table.

**CREATE TRIGGER after\_transaction\_update ...**

* This command creates a trigger named after\_transaction\_update that is executed after a transaction in the MemberTransactions table is updated.

**CREATE TRIGGER before\_borrow\_check ...**

* This command creates a trigger named before\_borrow\_check that is executed before a new transaction is inserted into the MemberTransactions table.

**CREATE TABLE BookStatusLog (...)**

* This command creates a table named BookStatusLog to keep track of changes in book availability status.

**CREATE TRIGGER after\_book\_status\_change ...**

* This command creates a trigger named after\_book\_status\_change that is executed after a book's availability status is updated in the Books table.

**CREATE FUNCTION CalculateTotalFines(...)**

* This command creates a function named CalculateTotalFines to calculate the total fines for a member.

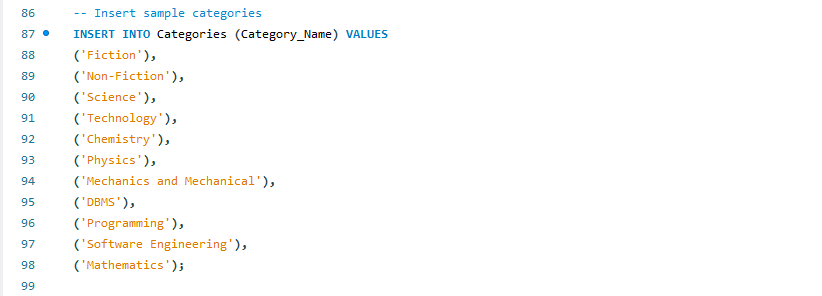
**CREATE FUNCTION GetBookAvailabilityDetails(...)**

* This command creates a function named GetBookAvailabilityDetails to get the availability status of a book with additional details.

**7.DML STATEMENTS**

1)**INSERT INTO Categories (Category\_Name) VALUES** (...)

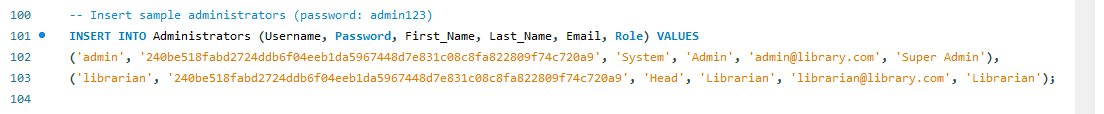
* This statement inserts sample category data into the Categories table.



2) **INSERT INTO Administrators (Username, Password, First\_Name, Last\_Name, Email, Role) VALUES (...)**

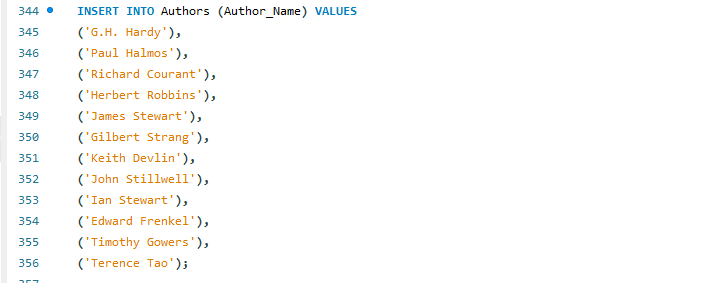
* This statement inserts sample administrator data into the

Administrators table.



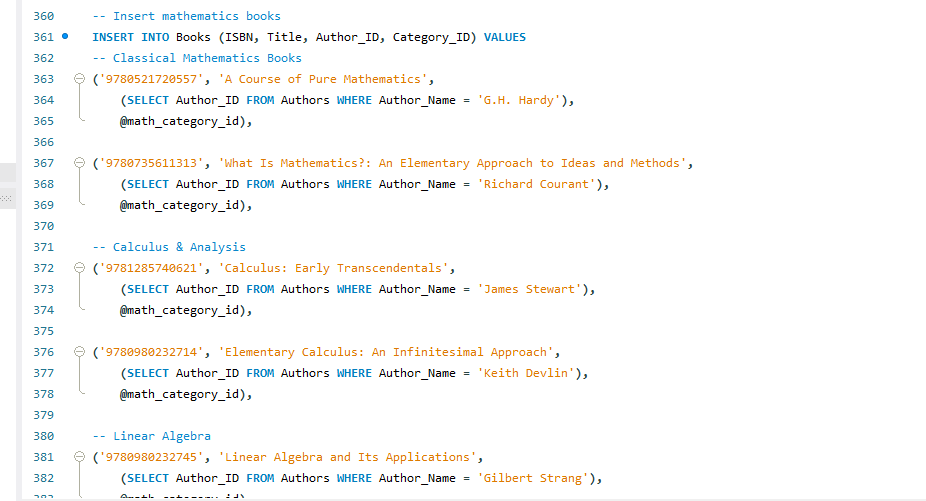
3) **INSERT INTO Authors (Author\_Name) VALUES (...)**

This statement inserts sample author data into the Authors table.



4) **INSERT INTO Books (ISBN, Title, Author\_ID, Category\_ID) VALUES (...)**

* This statement inserts sample book data into the Books table



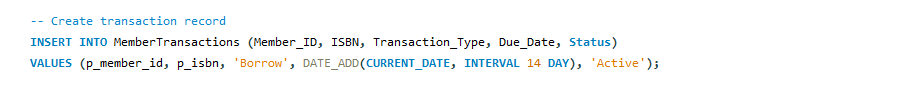
5) **INSERT INTO Members (Username, Password, First\_Name, Last\_Name, Email, Status) VALUES (...)**

* This statement inserts sample member data into the Members table



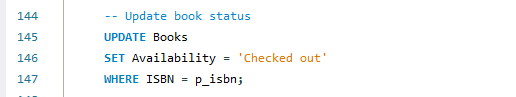
6) **INSERT INTO MemberTransactions (Member\_ID, ISBN, Transaction\_Type, Due\_Date, Status) VALUES (...)**

* This statement inserts sample member transaction data into the MemberTransactions table.



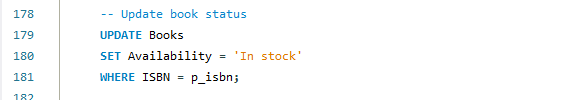
7) **UPDATE Books SET Availability = 'Checked out' WHERE ISBN = p\_isbn**

* This statement updates the availability status of a book in the Books table.



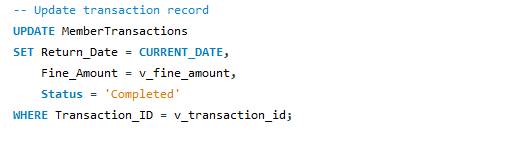
8) **UPDATE Books SET Availability = 'In stock' WHERE ISBN = p\_isbn**

* This statement updates the availability status of a book in the Books table.



9) **UPDATE MemberTransactions SET Return\_Date = CURRENT\_DATE, Fine\_Amount = v\_fine\_amount, Status = 'Completed' WHERE Transaction\_ID = v\_transaction\_id**

* This statement updates the member transaction record in the MemberTransactions table when a book is returned.



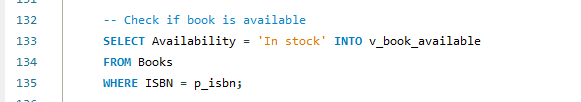
10) **DELETE FROM Books WHERE ISBN = p\_isbn**

* This statement deletes a book from the Books table.



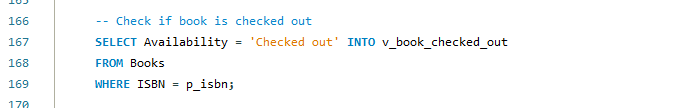
11) **SELECT Availability = 'In stock' INTO v\_book\_available FROM Books WHERE ISBN = p\_isbn;**

* This statement checks if the book with the given ISBN is available (in stock) and stores the result in the v\_book\_available variable. It's used in the CheckOutBook procedure to ensure the book is available before checking it out.



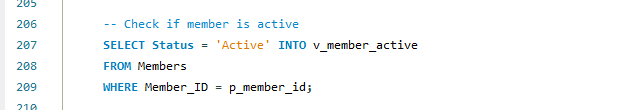
12) **SELECT Availability = 'Checked out' INTO v\_book\_checked\_out FROM Books WHERE ISBN = p\_isbn;**

* This statement checks if the book with the given ISBN is currently checked out and stores the result in the v\_book\_checked\_out variable. It's used in the ReturnBook procedure to ensure the book is currently checked out before allowing it to be returned.



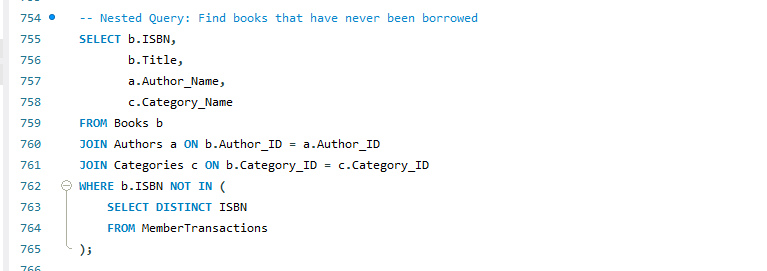
13) **SELECT Status = 'Active' INTO v\_member\_active FROM Members WHERE Member\_ID = p\_member\_id;**

* This statement checks if the member with the given ID has an active status and stores the result in the v\_member\_active variable. It's used in the BorrowBook procedure to ensure the member is active before allowing them to borrow a book.



**8) QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)**

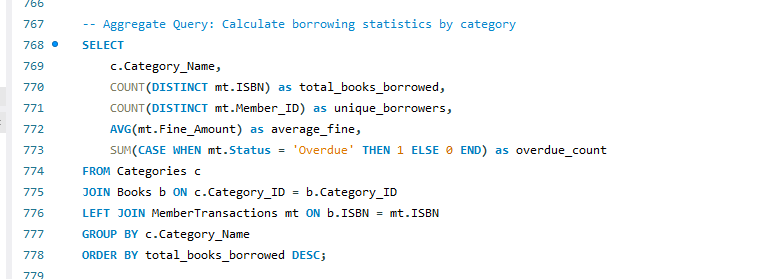
Nested JOIN Query:



This query finds the books that have never been borrowed by members.

* The main query selects the ISBN, Title, Author Name, and Category Name of the books.
* It joins the Books, Authors, and Categories tables to get the necessary information about the books.
* The WHERE clause uses a nested query to check for books that do not have any associated transactions in the MemberTransactions table.
* The nested query selects the distinct ISBNs from the MemberTransactions table. This gives us a list of all the books that have been borrowed at least once.
* The main query then filters the books to only include those whose ISBNs are not present in the nested query's result. This gives us the books that have never been borrowed.

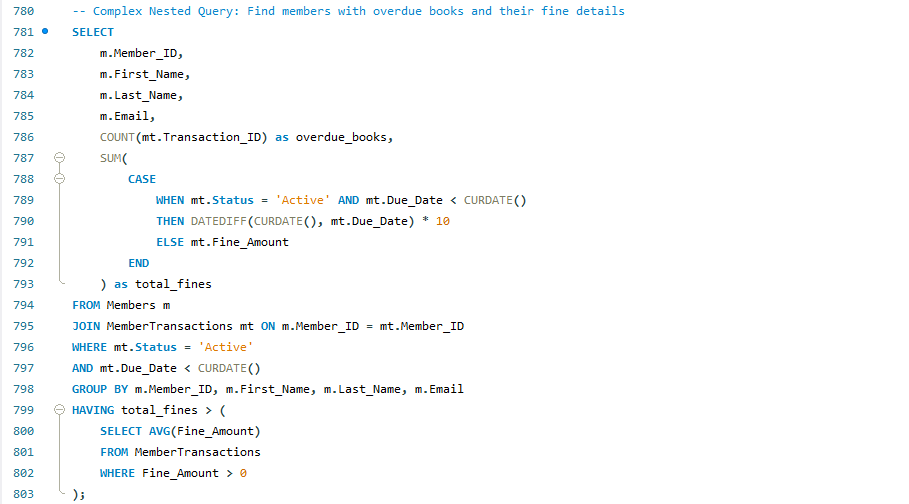
Aggregate Query:



This query calculates various borrowing statistics for each category of books.

* The main query selects the Category Name, total books borrowed, unique borrowers, average fine, and the count of overdue books.
* It joins the Categories, Books, and MemberTransactions tables to get the necessary data.
* The LEFT JOIN with the MemberTransactions table ensures that we include all categories, even if they don't have any associated transactions.
* The GROUP BY clause groups the results by the Category Name.
* For each category, it calculates:
  + total\_books\_borrowed: The count of distinct books borrowed (using DISTINCT mt.ISBN).
  + unique\_borrowers: The count of distinct members who have borrowed books (using DISTINCT mt.Member\_ID).
  + average\_fine: The average fine amount.
  + overdue\_count: The count of overdue transactions (using a CASE statement to check the Status column).
* The results are ordered by the total\_books\_borrowed column in descending order.

Complex Nested Query:



This complex nested query finds members who have overdue books and calculates the total fines they owe.

* The main query selects the Member ID, First Name, Last Name, Email, the count of overdue books, and the total fines owed.
* It joins the Members and MemberTransactions tables to get the necessary information about the members and their transactions.
* The WHERE clause filters to only include active transactions that are overdue.
* The GROUP BY clause groups the results by the member's identification columns (Member ID, First Name, Last Name, Email).
* The COUNT aggregate function is used to get the count of overdue books for each member.
* The SUM with a CASE statement is used to calculate the total fines owed by each member. If the transaction is active and overdue, it calculates the fine based on the number of days overdue . Otherwise, it simply uses the Fine\_Amount column.
* The HAVING clause further filters the results to only include members whose total fines are above the average fine amount.

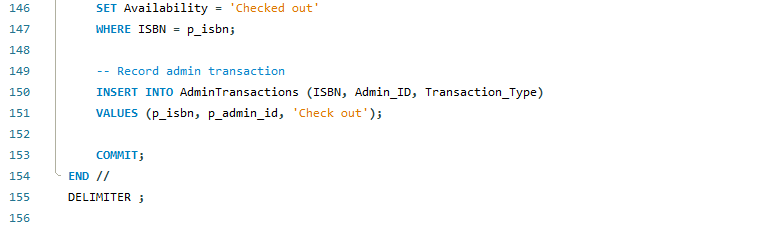
This query provides a comprehensive view of the members with overdue books and the associated fine details, allowing the library management system to identify and follow up with members who need to return overdue books and pay their fines.

**9) STORED PROCEDURE, FUNCTIONS AND TRIGGERS**

Stored Procedures

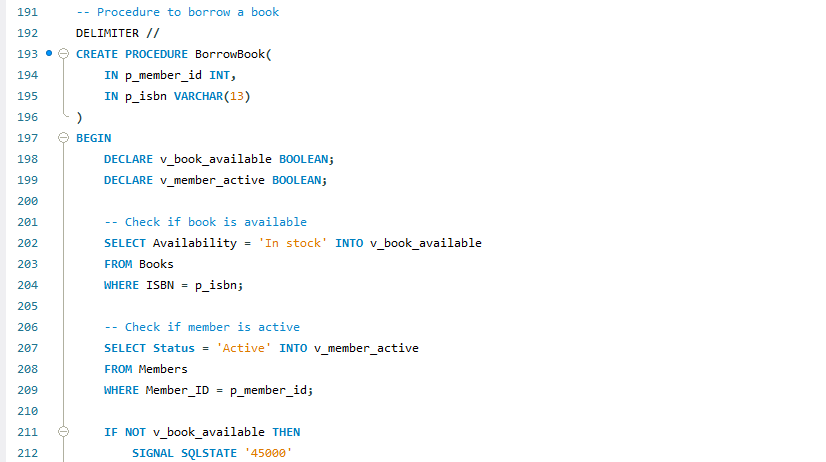
CheckOutBook



****

* Checks if the requested book is available for checkout.
* Updates the book's availability status to "Checked out".
* Records the checkout transaction in the AdminTransactions table

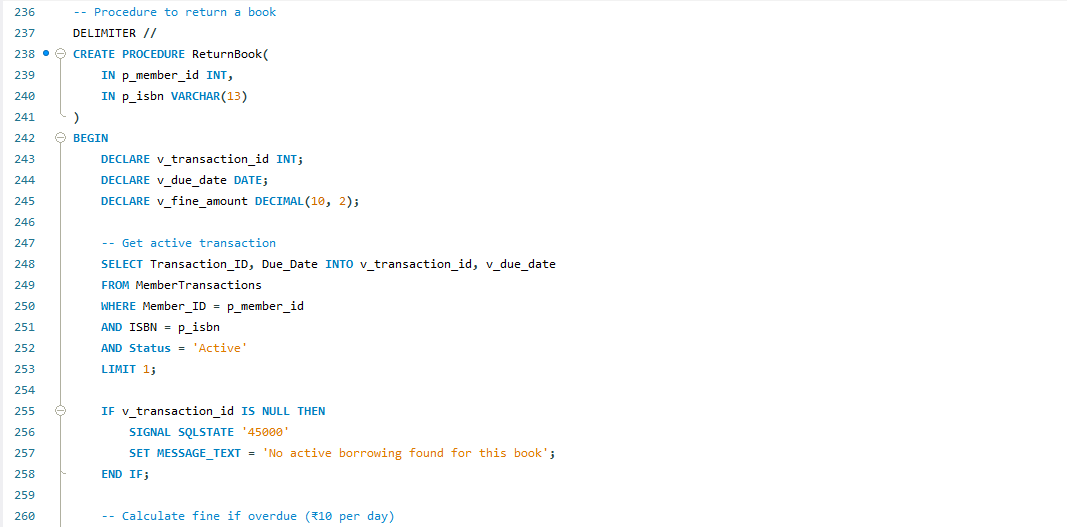
Borrow Book

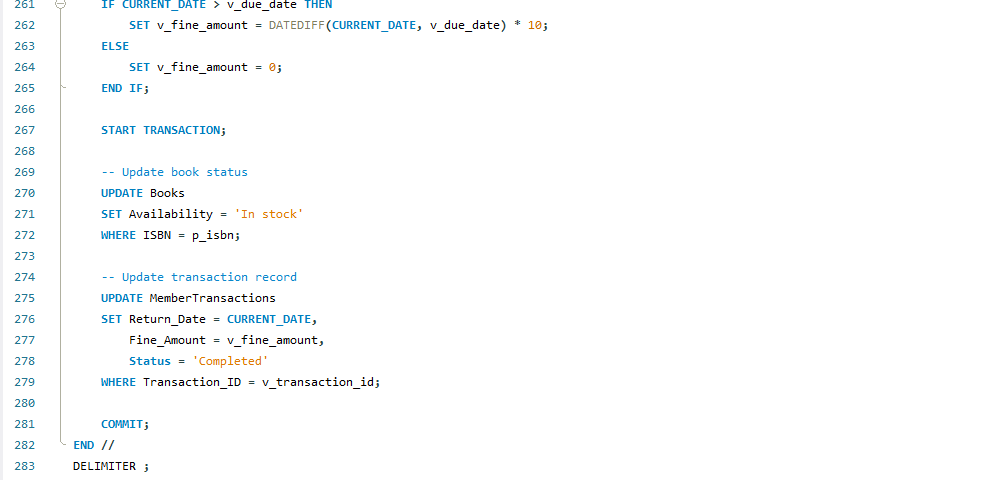




* Checks if the requested book is available and if the member's account is active.
* Updates the book's availability status to "Checked out".
* Creates a new transaction record in the MemberTransactions table with a due date 14 days from the current date.

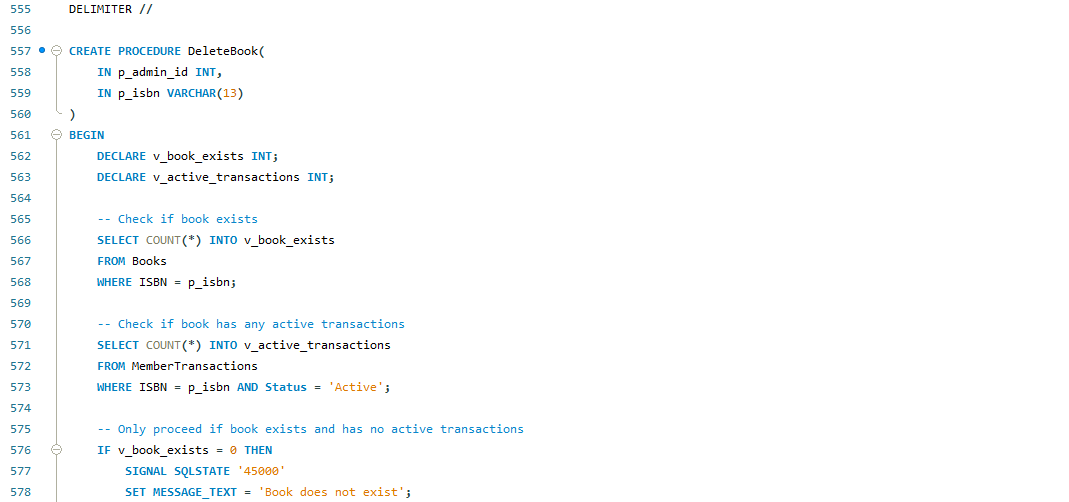
Return Book

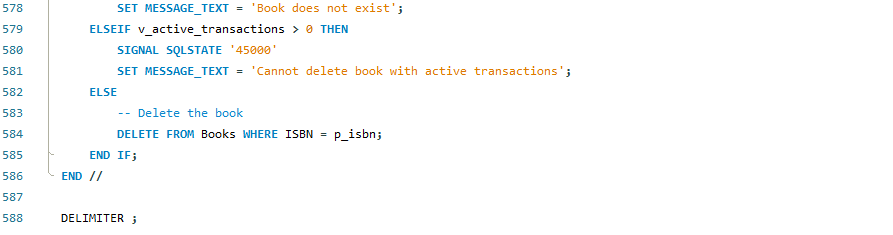




* Retrieves the active transaction for the given member and book.
* Calculates the fine amount based on the number of days the book is overdue.
* Updates the book's availability status to "In stock".
* Updates the transaction record in the MemberTransactions table with the return date and fine amount

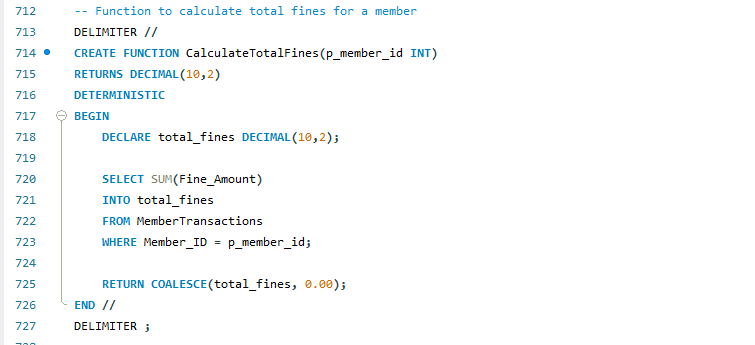
Delete Book





Functions

Calculate Total Fines



* Calculates the total fines owed by a given member by summing the Fine\_Amount column in the MemberTransactions table.

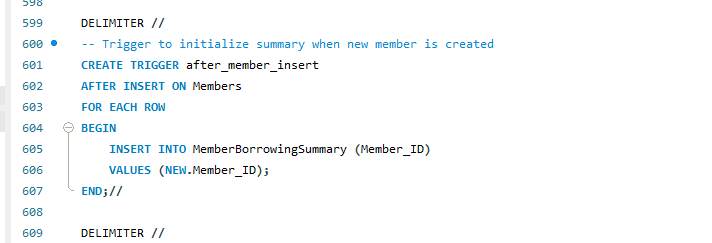
GetBookAvailabilityDetails:



* Retrieves the availability status of a given book, including the due date if the book is currently checked out.

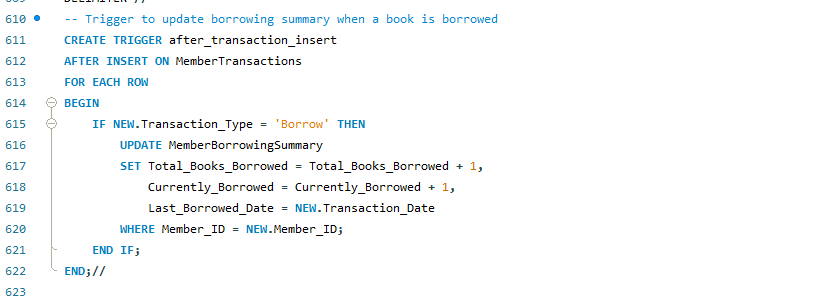
Triggers:

after\_member\_insert



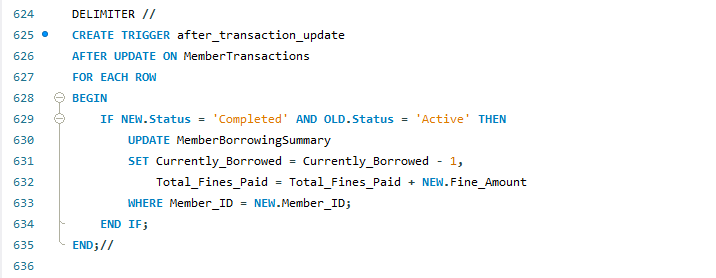
* Initializes a new record in the MemberBorrowingSummary table when a new member is created.

after\_transaction\_insert



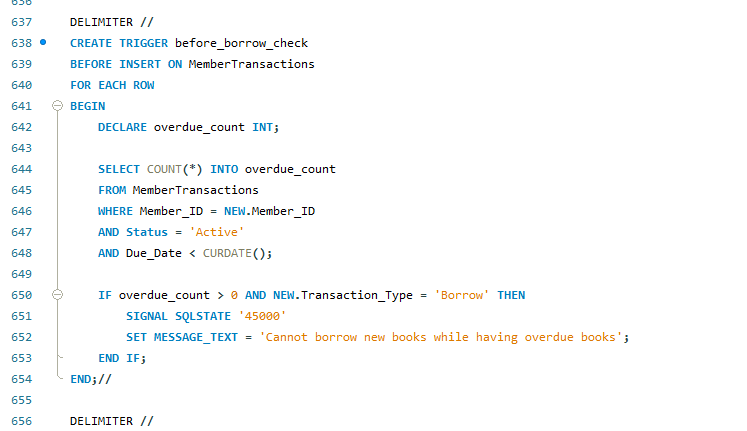
Updates the MemberBorrowingSummary table when a book is borrowed, incrementing the Total\_Books\_Borrowed and Currently\_Borrowed columns.

after\_transaction\_update



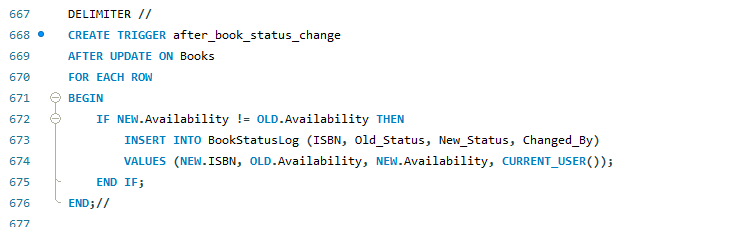
* Updates the MemberBorrowingSummary table when a book is returned, decrementing the Currently\_Borrowed column and adding the fine amount to the Total\_Fines\_Paid column.

before\_borrow\_check



* Checks if the member has any overdue books before allowing a new borrow transaction.
* Throws an error if the member has overdue books

after\_book\_status\_change



* Logs changes in the book's availability status in the BookStatusLog table.

**9) FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)**

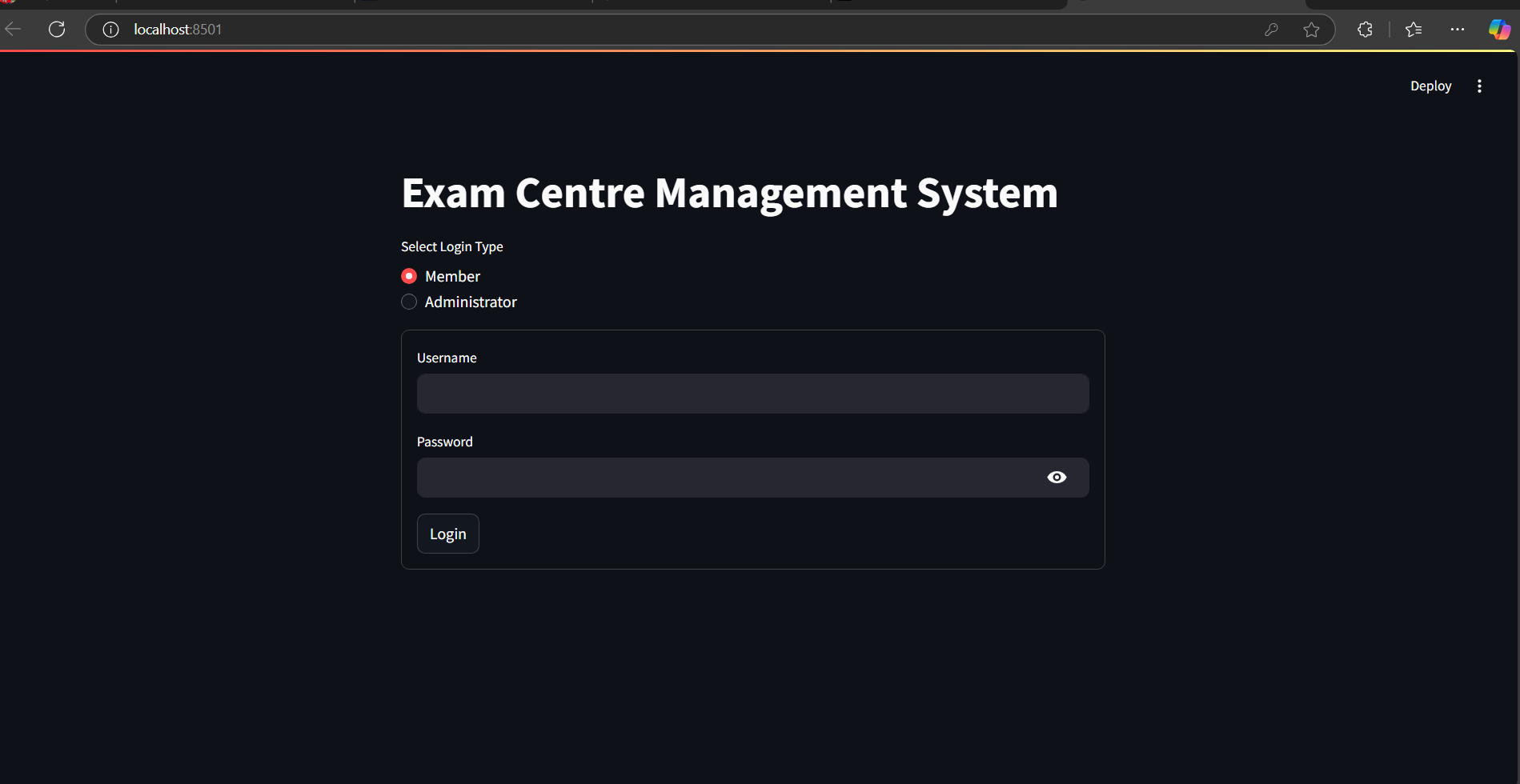
The frontend has been developed using Streamlit, a Python framework for creating web applications.

1. Main Structure:

* The application uses a single-page architecture managed by main() function
* It maintains user state using Streamlit's session state (st.session\_state)
* Two main portals: Admin Portal and Member Portal
* Login page as the entry point.

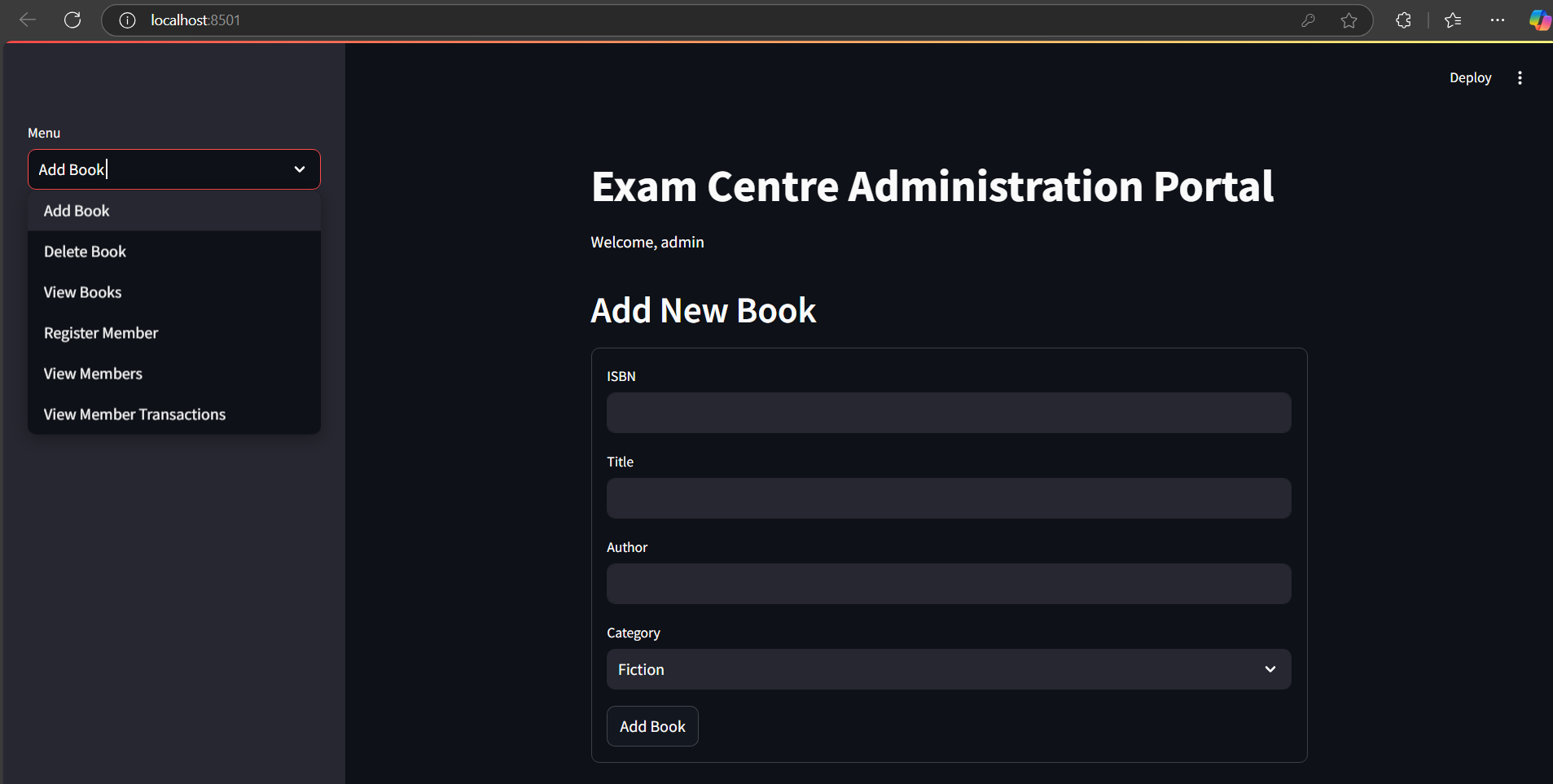
2.Login Page (login\_page()):

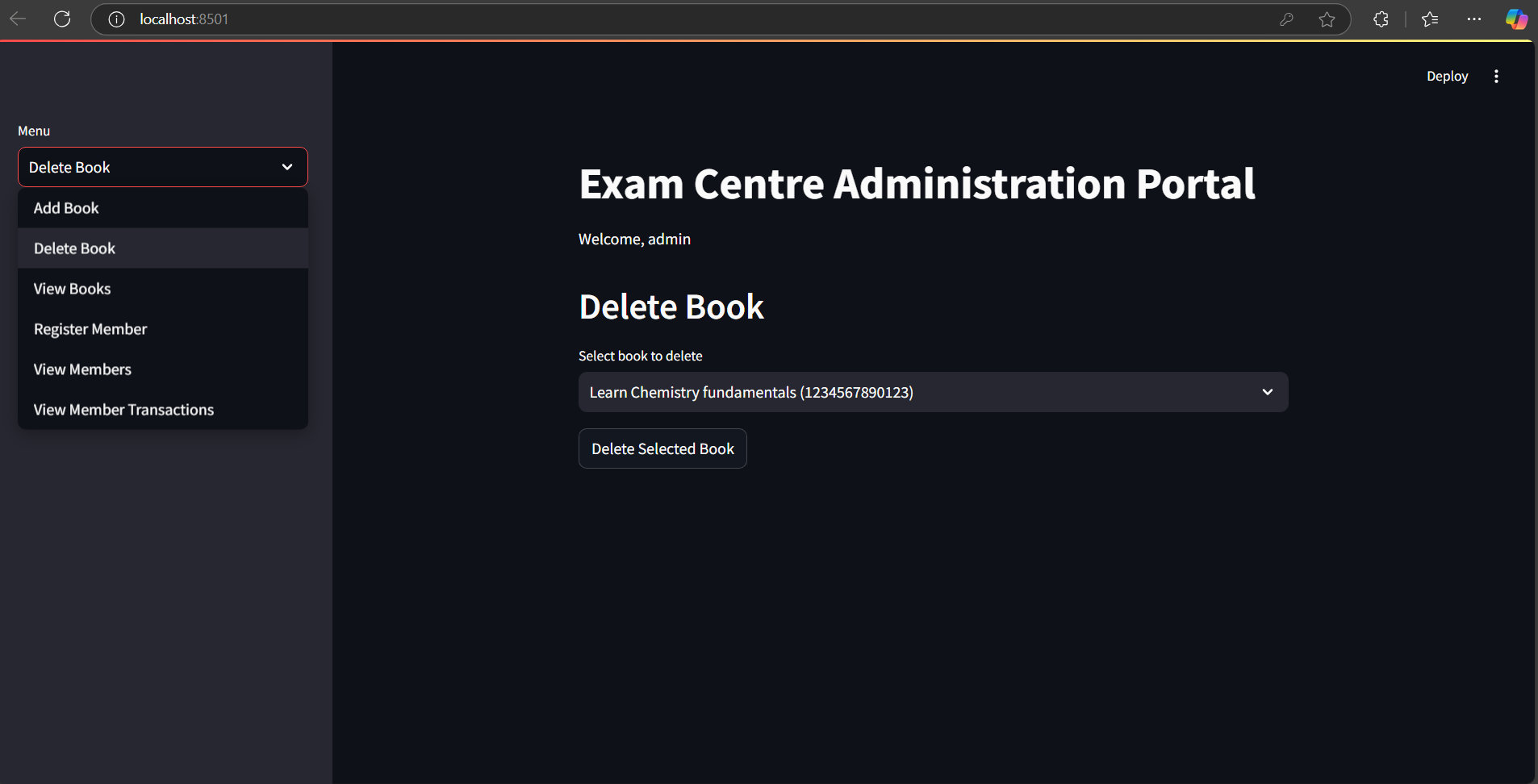
* Clean title "Exam Centre Management System"
* Radio button to select user type (Member/Administrator)
* Simple form with:
  + Username input
  + Password input (masked)
  + Login button
* Success/Error messages for login attempts

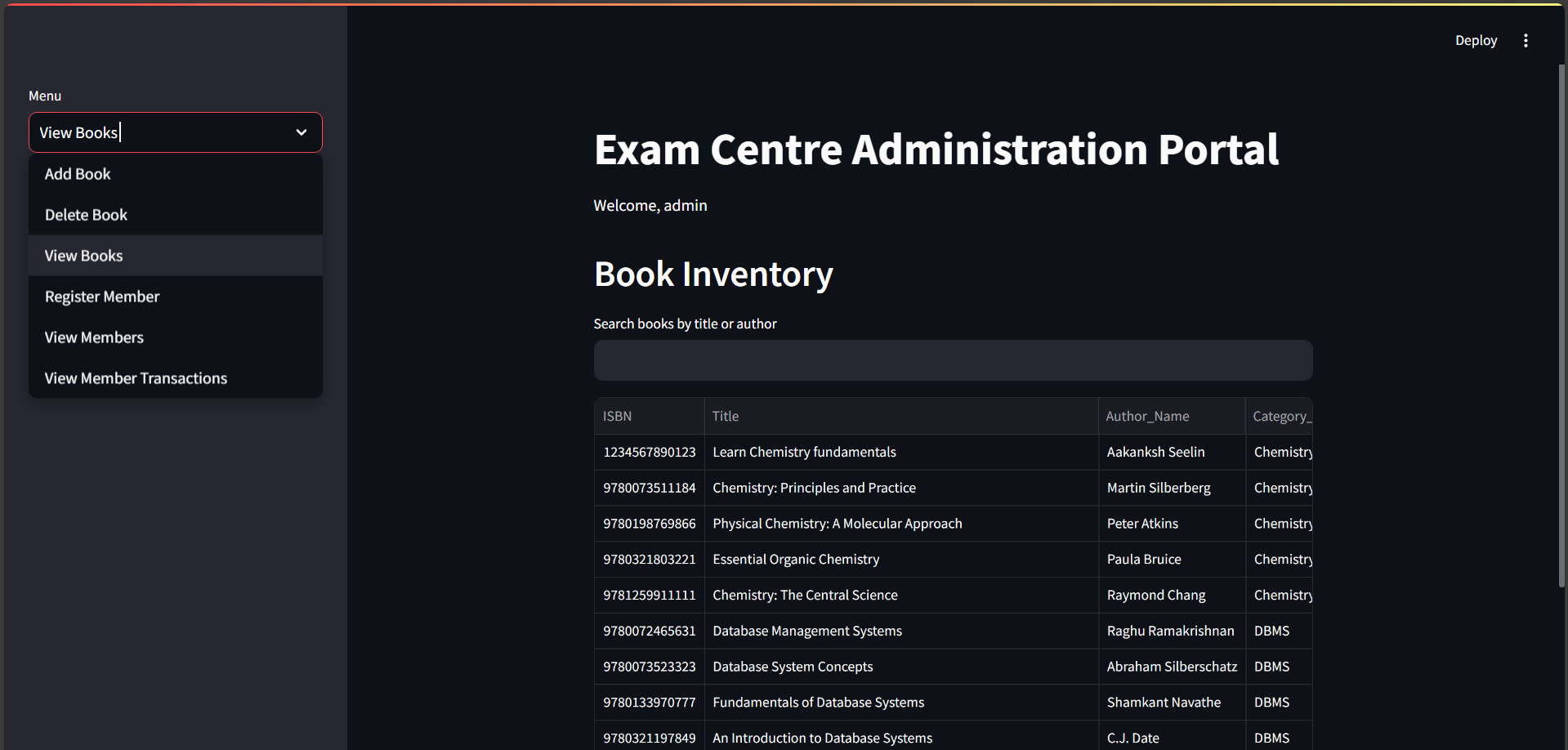
****

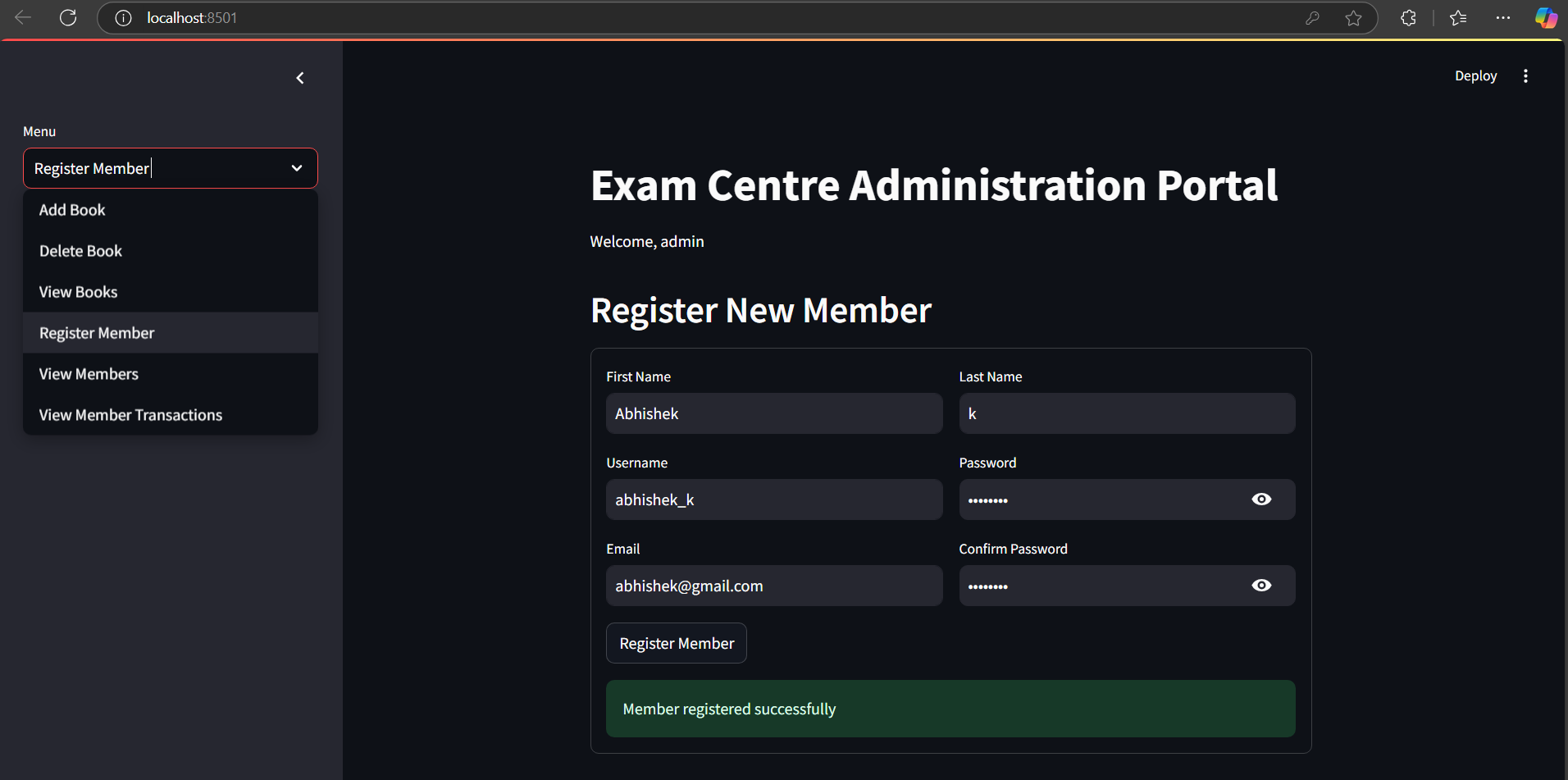
3Admin Portal (admin\_portal()):

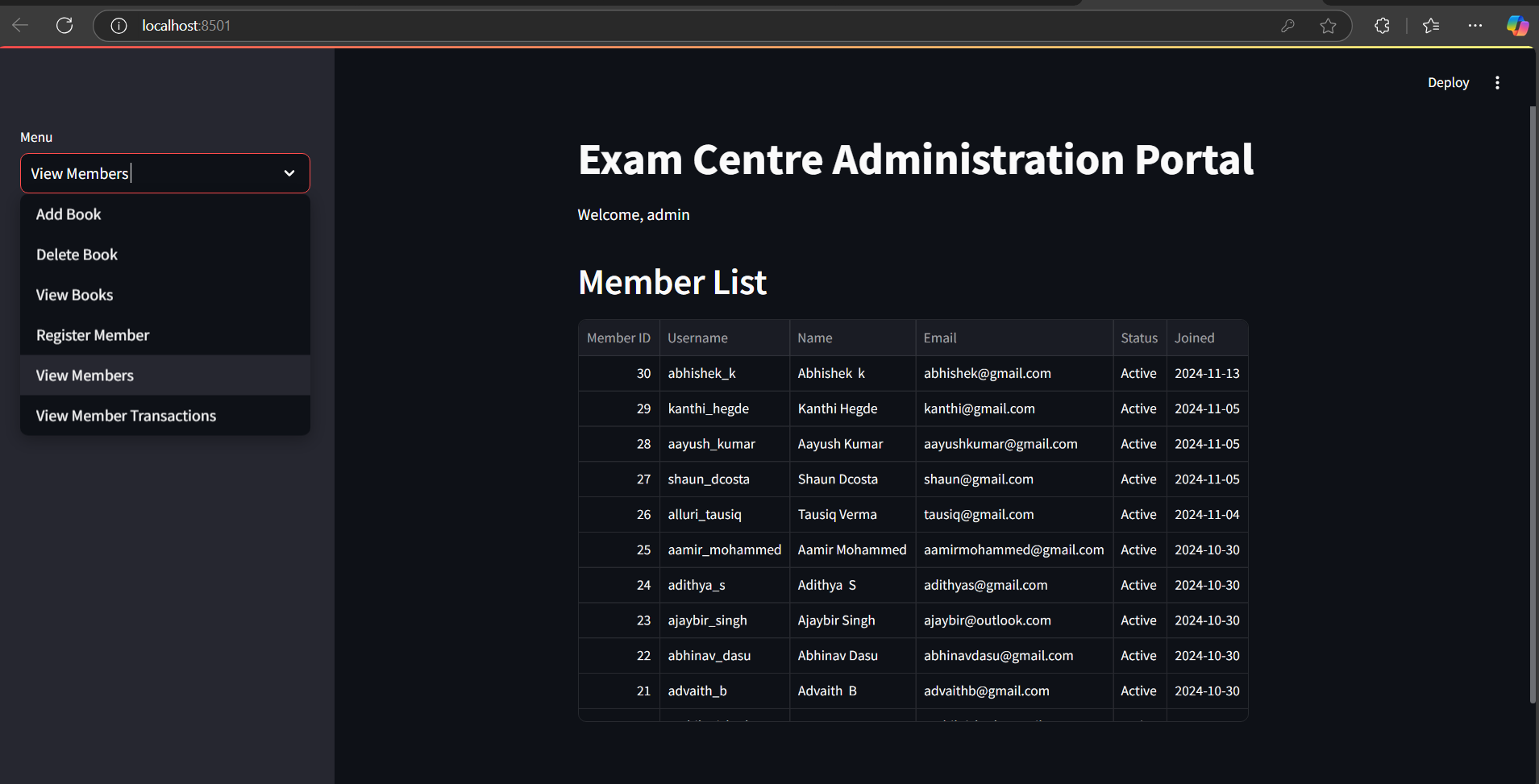
* Sidebar menu with options:
  + Add Book
  + Delete Book
  + View Books
  + Register Member
  + View Members
  + View Member Transactions
* Each menu option has its own section with relevant forms and tables
* Logout button in sidebar

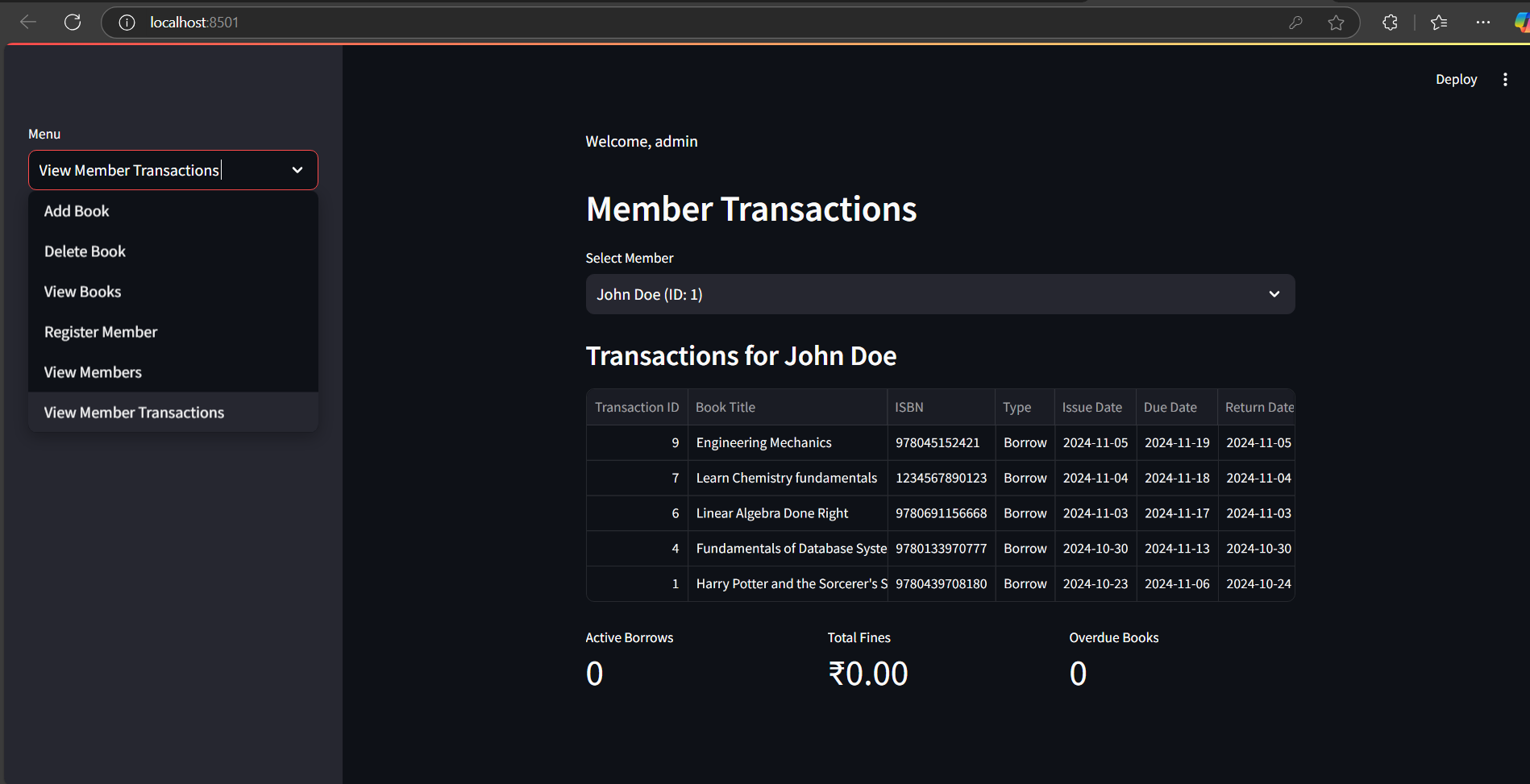






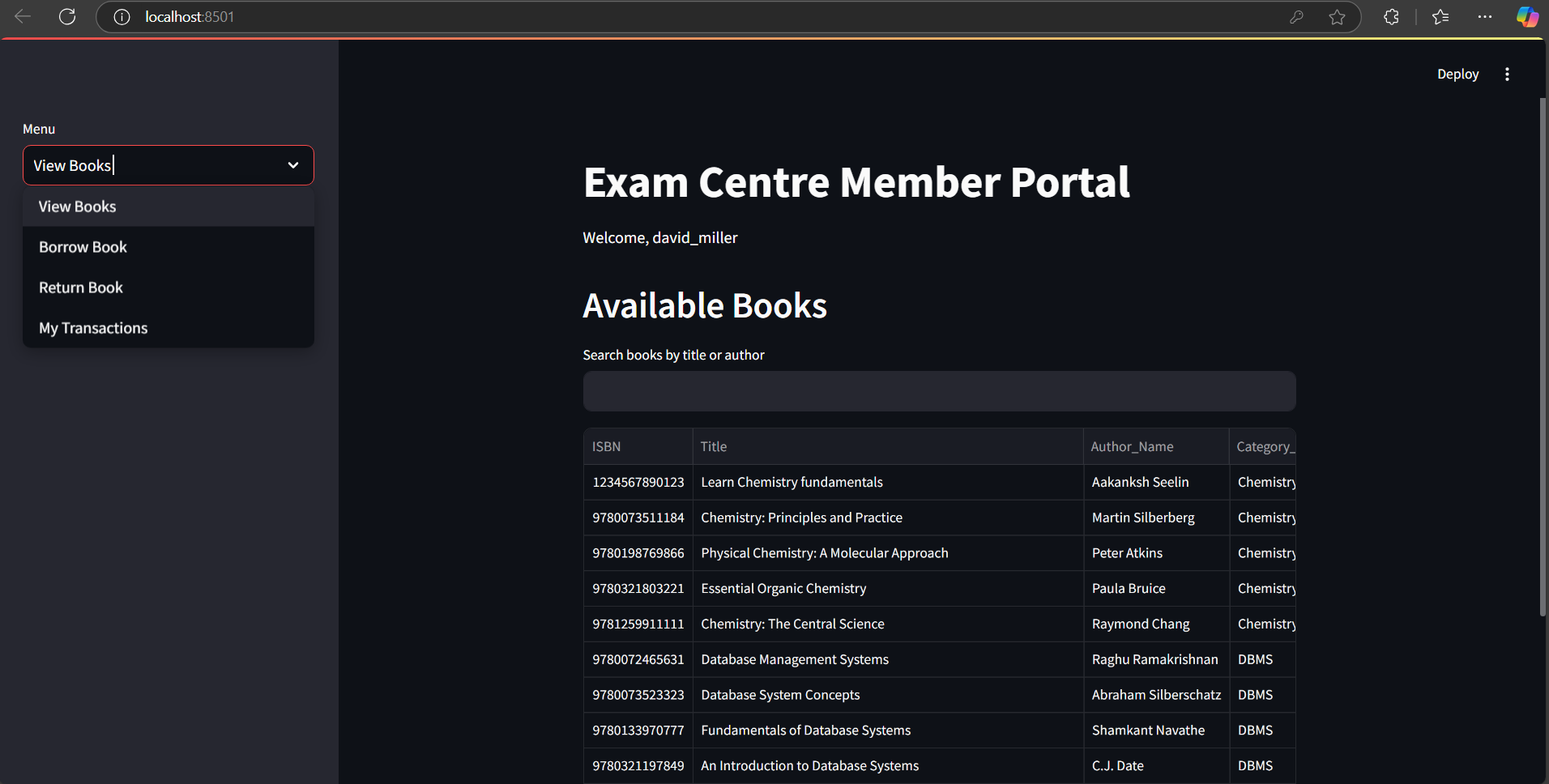


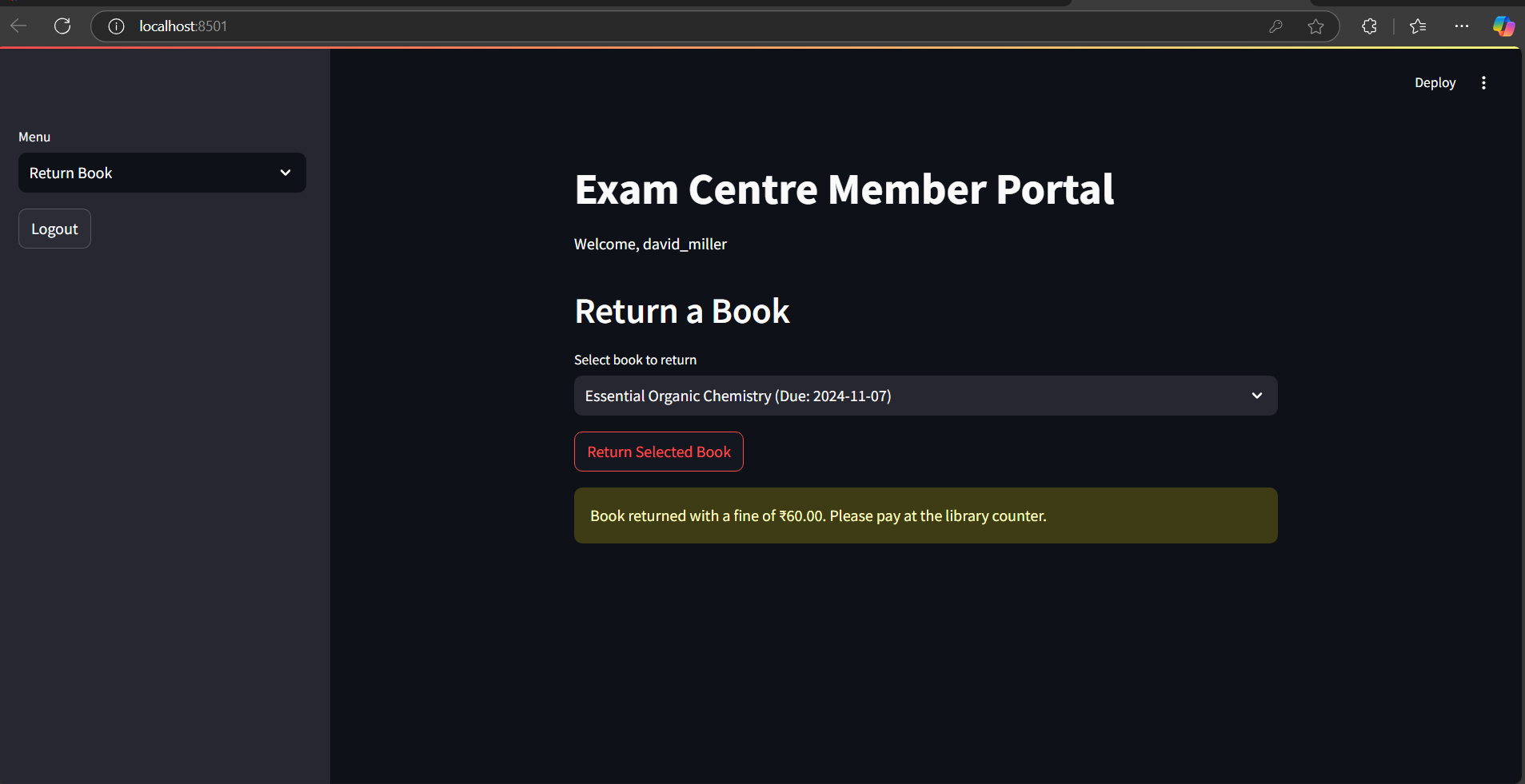
****

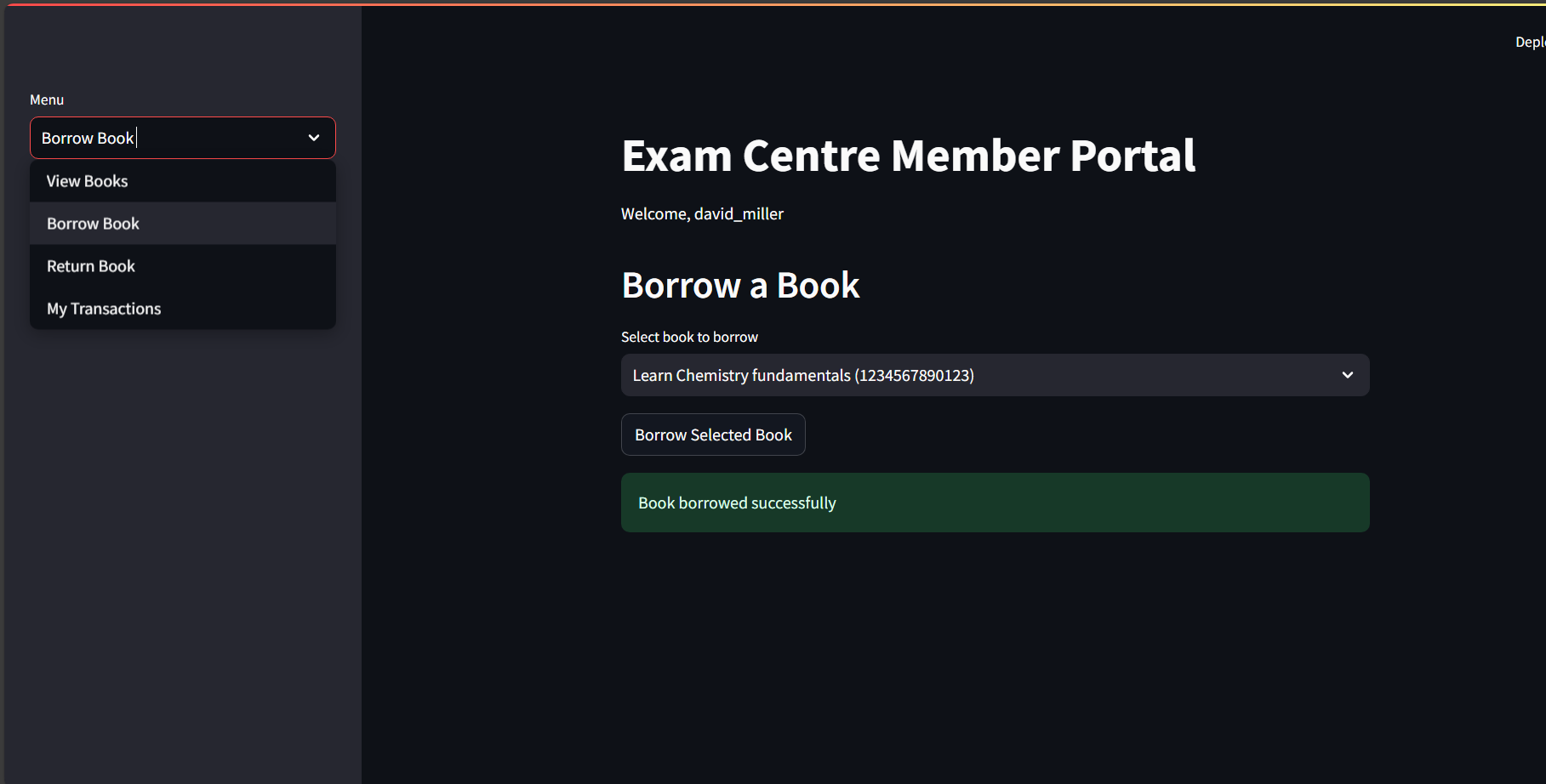
****

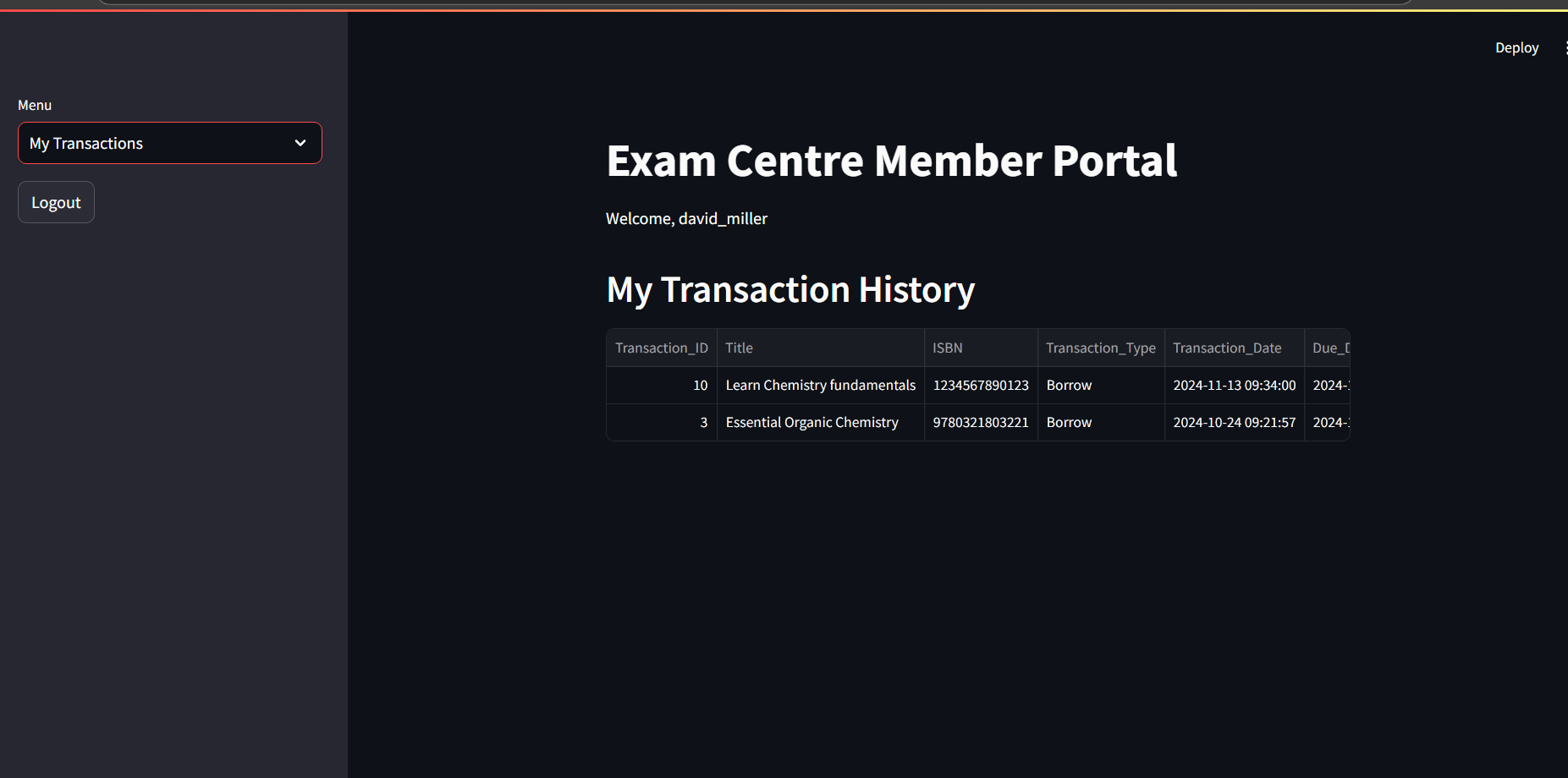
1. Member Portal (member\_portal()):

* Sidebar menu with options:
  + View Books
  + Borrow Book
  + Return Book
  + My Transactions
* Each section has appropriate forms and data displays
* Logout button in sidebar

****

****

****

****

1. Key UI Features:

* Responsive layout with columns for form organization
* Interactive data tables with search functionality
* Clear success/error messages for user feedback
* Metrics display for statistics
* Dropdown menus for selections
* Form validation and error handling

1. Data Presentation:

* Books displayed in tabular format with search capability
* Member transactions shown with formatted dates and currency
* Status indicators for books (In stock/Borrowed)
* Fine amounts displayed in Indian Rupees (₹)

1. Navigation:

* Clear hierarchical menu structure
* Sidebar for main navigation
* Logical grouping of related functions
* Easy logout access

This frontend design follows several good practices:

* Clean and intuitive interface
* Consistent layout across different sections
* Clear feedback for user actions
* Proper form validation
* Organized menu structure
* Responsive design elements
* Clear data presentation

**REFERENCES/BIBLIOGRAPHY**

* Database Management System principles and design
* Software Engineering concepts and practices.
* SQL commands and syntax.
* Github

[**SQL Syntax**](https://www.sqltutorial.org/sql-syntax/)

[**Database Design in DBMS - GeeksforGeeks**](https://www.geeksforgeeks.org/database-design-in-dbms/)

[**Software Engineering Tutorial - GeeksforGeeks**](https://www.geeksforgeeks.org/software-engineering/)

**APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

* SQL: Structured Query Language
* CRUD: Create, Read, Update, Delete
* EMS: Exam Centre Management System
* UI: User Interface
* Admin: Administrator of the EMS
* Member: A registered user of the library who can borrow books
* ISBN: International Standard Book Number