ADBMS

Assignment

2025

Group Number: AL

Coursework Type: Group Assignment

Team Leader: M.H.M.P.D.Herath

E-mail: mpdherath@students.nsbm.ac.lk

Library Management System

Section 1

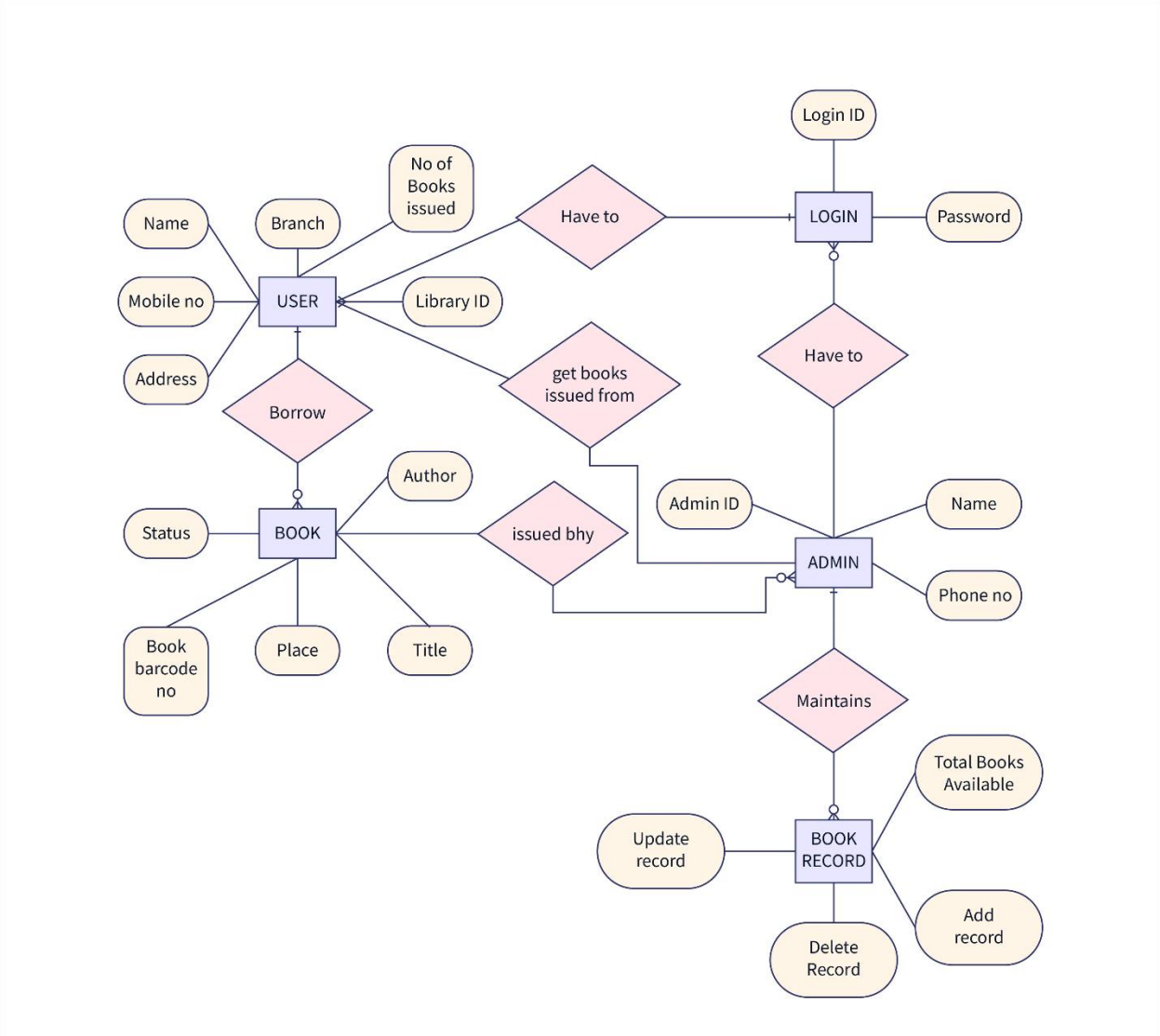
System Scenario

The Library Management System is designed to facilitate the checking out and managing of books in an orderly manner. The system mandates every user to log in utilizing their specific credentials. Following successful authentication, a user may access the library services available under their account, which comprises personal information such as their name, branch, address, contact number, and Library ID. Each user is able to borrow books from the library, and the system tracks the number of books borrowed by the user at any given time.

Also, system contains extensive metadata for every book including title, author, place of publication, status (available or borrowed), and together with a unique barcode number which provides aid in identification. Users who check out books first have their transactions recorded and the record kept under the Administrator's name together with other relevant transfer details. Administrators are provided with an individual Admin ID which uniquely identifies them, together with their personal log in credentials which facilitates system control. Their role involves updating the book records in the library which enables them to add new books, edit older records, or delete obsolete records. All these operations are captured through a transaction on the book record entity that maintains the inventory of copies of each book.

The process of borrow is concerned with tracking the dates of book issued and book return in relation to the user, book and administrator. So the responsibility and tracking in the system is assured. The system is literally organized from the login, through the record of the books borrowed to the maintenance of the records which provides order and simplicity to the administrators and users of the library.

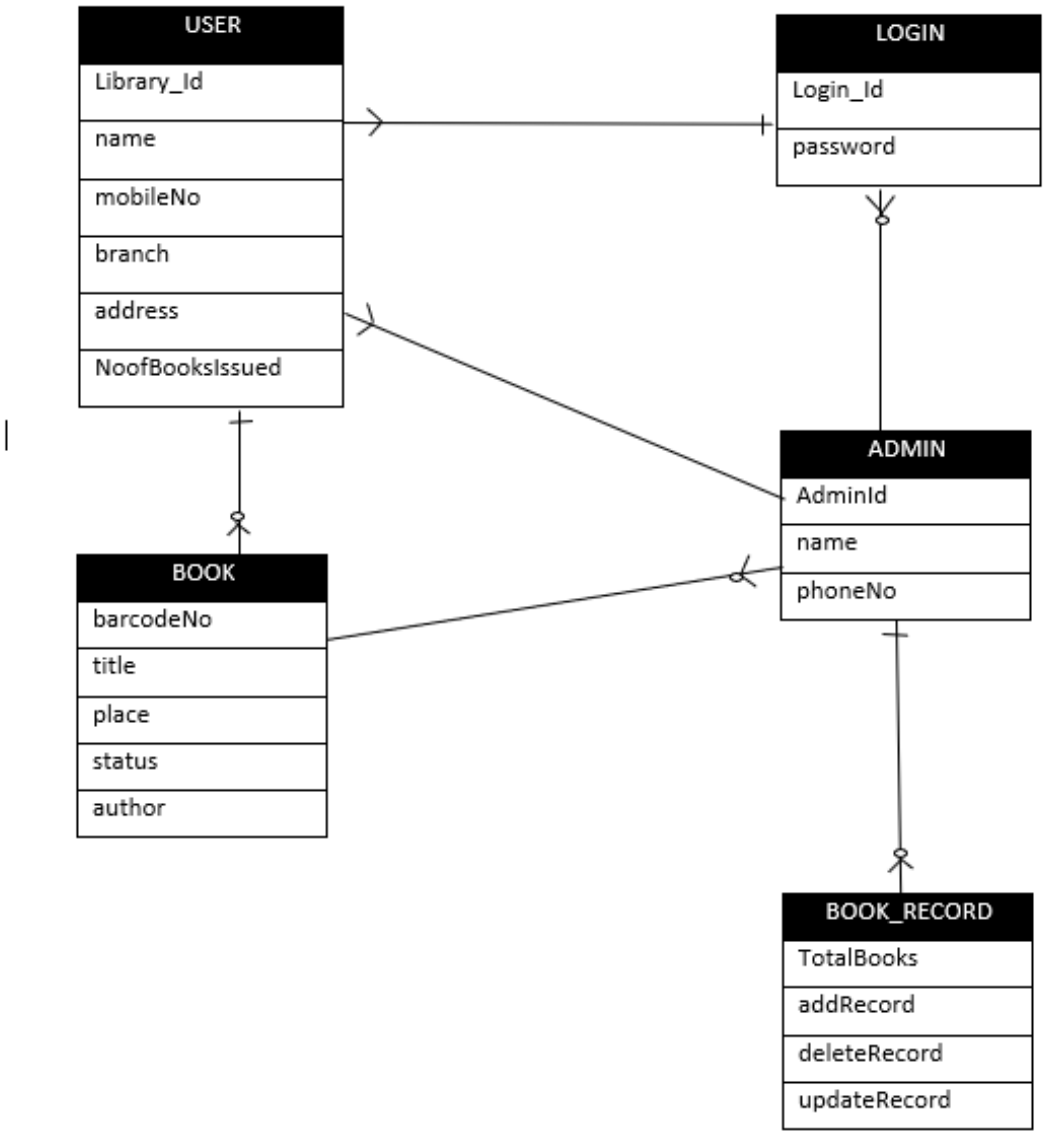
ER Diagram



📌 Assumptions

- Each book has a unique Barcode No.
- One User can borrow multiple books.
- A Book can only be borrowed by one user at a time.
- Admins can update the status of books.
- Admins and Users must log in to the system.
- Every borrow event is recorded with both the User and Admin involved.

Relational Mapping Diagram



Data Normalization Tables (up to 3NF)

USER
User ID (PK)
Branch
Mobile No
Address
Library ID
No_of_Books_Issued

BOOK
Book ID (PK)
Title
Author
Place
Status
Book Barcode No

ADMIN
Admin_ID (PK)
Name
Phone No

LOGIN
Login_ID (PK)
Password
Admin ID (FK)
User ID (FK)

BOOK_RECORD
Record ID (PK)
Book ID (FK)
Total Books Available
Update Record
Delete_Record
Add_Record

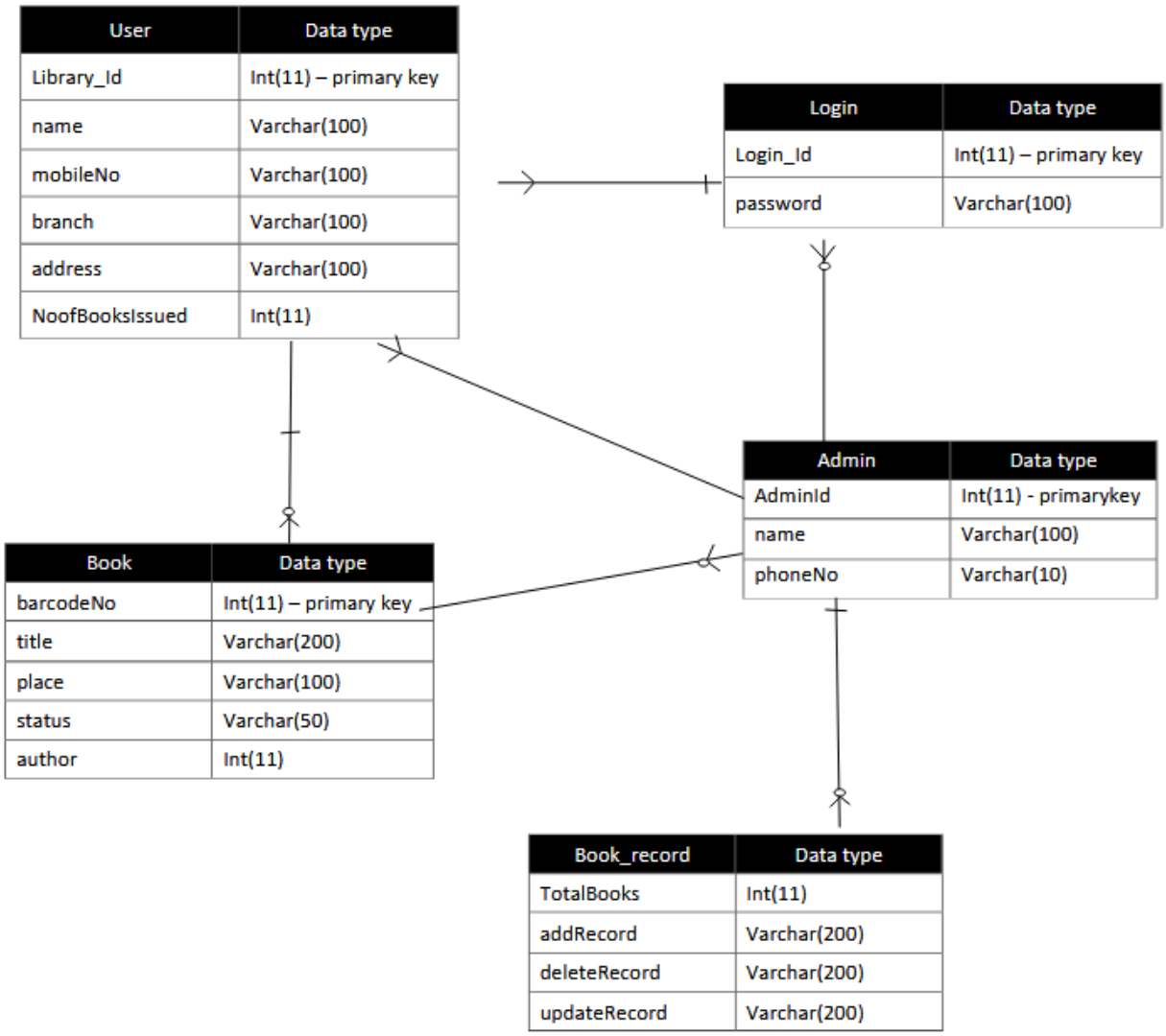
BORROW
Borrow ID (PK)
User ID (FK)
Book ID (FK)
Admin ID (FK)
Issue Date
Due_Date

Data Dictionary Summary

Table	Description
Users	Stores user details
Login	Stores login credentials
Admin	Stores admin details
Books	Stores book metadata
Borrow	Links users, books, and issuing admin
Book Record	Maintains record of book stock

Section 2

Database Diagram



Database Tables

The Library Management System database consists of well-structured tables to manage books, users, and library operations efficiently. Core tables such as books, authors, publishers, languages, and categories handle book-related metadata, while copies and ebooks track physical and digital availability. The users table, along with genders, faculties, semesters, and roles, stores member information and roles. Transactional activities like borrowing are managed by the deposits table. Additional tables such as libraries, news, cities, and sections enhance informational and administrative capabilities, ensuring smooth library operations.

Here are some tables screenshots that we created

```
Database changed
MariaDB [library]> show tables;
+--------------------+
| Tables_in_library |
+--------------------+
| admin              |
| book               |
| book_record        |
| borrow             |
| login              |
| user               |
+--------------------+
6 rows in set (0.001 sec)
```

```
MariaDB [library]> desc admin;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| admin_id | int(11)      | NO   | PRI | NULL    |       |
| name     | varchar(100) | YES  |     | NULL    |       |
| phone_no | varchar(20)  | YES  |     | NULL    |       |
| login_id | int(11)      | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.016 sec)

MariaDB [library]> desc book;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| book_barcode_no | int(11)      | NO   | PRI | NULL    |       |
| title           | varchar(200) | YES  |     | NULL    |       |
| author          | varchar(100) | YES  |     | NULL    |       |
| place           | varchar(100) | YES  |     | NULL    |       |
| status          | varchar(50)  | YES  |     | NULL    |       |
| admin_id        | int(11)      | YES  | MUL | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
6 rows in set (0.017 sec)
```

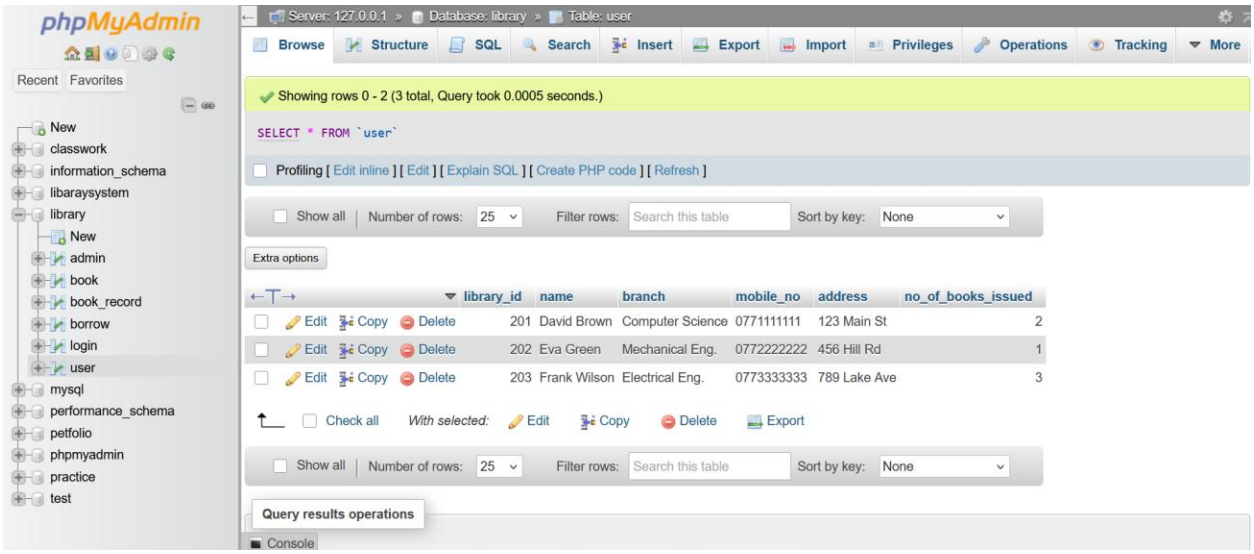
```
MariaDB [library]> desc book_record;
+----------------------+---------+------+-----+---------+----------------+
| Field                | Type    | Null | Key | Default | Extra          |
+----------------------+---------+------+-----+---------+----------------+
| record_id            | int(11) | NO   | PRI | NULL    | auto_increment |
| total_books_available | int(11) | YES  |     | NULL    |                |
| admin_id             | int(11) | YES  | MUL | NULL    |                |
+----------------------+---------+------+-----+---------+----------------+
3 rows in set (0.022 sec)

MariaDB [library]> desc borrow;
+----------------+---------+------+-----+---------+----------------+
| Field          | Type    | Null | Key | Default | Extra          |
+----------------+---------+------+-----+---------+----------------+
| borrow_id      | int(11) | NO   | PRI | NULL    | auto_increment |
| library_id     | int(11) | YES  | MUL | NULL    |                |
| book_barcode_no | int(11) | YES  | MUL | NULL    |                |
+----------------+---------+------+-----+---------+----------------+
3 rows in set (0.018 sec)
```

```
MariaDB [library]> desc login;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| login_id | int(11)      | NO   | PRI | NULL    |       |
| password | varchar(100) | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
2 rows in set (0.016 sec)

MariaDB [library]> desc user;
+--------------------+--------------+------+-----+---------+-------+
| Field              | Type         | Null | Key | Default | Extra |
+--------------------+--------------+------+-----+---------+-------+
| library_id         | int(11)      | NO   | PRI | NULL    |       |
| name               | varchar(100) | YES  |     | NULL    |       |
| branch             | varchar(100) | YES  |     | NULL    |       |
| mobile_no          | varchar(20)  | YES  |     | NULL    |       |
| address            | varchar(255) | YES  |     | NULL    |       |
| no_of_books_issued | int(11)      | YES  |     | NULL    |       |
+--------------------+--------------+------+-----+---------+-------+
6 rows in set (0.021 sec)
```

Here is our PHPMyAdmin panel



Section 3

Advanced Database Implementation

This section includes the implementation of Triggers, User-Defined Functions (UDFs), Views, and Stored Procedures in Microsoft SQL Server, based on the library management system database design. Each SQL feature helps enhance automation, data retrieval, and process handling within the system. Below are the details of the created components along with their respective SQL statements and screenshots.

🔁 1. Triggers

Triggers are special types of stored procedures that automatically execute when an event (INSERT, UPDATE, DELETE) occurs in a table. In this system, triggers are used to **automatically update the status of book copies** when they are issued or returned.

✅ Trigger 1: Automatically updates the status of a book copy to "Issued" when a new borrow record is inserted.

✅ Trigger 2: Automatically sets the status of a returned book copy to "Available" when the borrow record is deleted.

📸 ***Screenshot of Trigger SQL Code below***

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE TRIGGER trg_book_issued
    -> AFTER INSERT ON BORROW
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE BOOK
    ->     SET status = 'Issued'
    ->     WHERE book_barcode_no = NEW.book_barcode_no;
    -> END $$
Query OK, 0 rows affected (0.007 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> INSERT INTO BORROW (library_id, book_barcode_no)
    -> VALUES (201, 301);
Query OK, 1 row affected (0.007 sec)

MariaDB [library]> SELECT status FROM BOOK WHERE book_barcode_no = 301;
+---------+
| status |
+---------+
| Issued |
+---------+
1 row in set (0.000 sec)

MariaDB [library]>
```

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE TRIGGER trg_book_returned
    -> AFTER DELETE ON BORROW
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE BOOK
    ->     SET status = 'Available'
    ->     WHERE book_barcode_no = OLD.book_barcode_no;
    -> END $$
Query OK, 0 rows affected (0.006 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> DELETE FROM BORROW WHERE library_id = 201 AND book_barcode_no = 301;
Query OK, 1 row affected (0.007 sec)

MariaDB [library]> SELECT status FROM BOOK WHERE book_barcode_no = 301;
+-----------+
| status    |
+-----------+
| Available |
+-----------+
1 row in set (0.000 sec)

MariaDB [library]>
```

🧱 2. User-Defined Functions (UDFs)

User-defined functions return a single value based on the logic defined. These are used for **simplifying queries and encapsulating logic** that is frequently used across multiple queries or reports.

✅ Function 1: Returns the total number of books borrowed by a specific user (by library ID).

✅ Function 2: Returns the full name of an admin using their admin ID.

📸 *Screenshot of UDF SQL Code below*

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE FUNCTION get_borrowed_books_count(uid INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE total INT;
    ->     SELECT COUNT(*) INTO total FROM BORROW WHERE library_id = uid;
    ->     RETURN total;
    -> END $$
Query OK, 0 rows affected (0.007 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> SELECT get_borrowed_books_count(201) AS total_books_borrowed;
+----------------------+
| total_books_borrowed |
+----------------------+
|                    1 |
+----------------------+
1 row in set (0.006 sec)

MariaDB [library]>
```

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE FUNCTION get_admin_name(aid INT)
    -> RETURNS VARCHAR(100)
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE adminName VARCHAR(100);
    ->     SELECT name INTO adminName FROM ADMIN WHERE admin_id = aid;
    ->     RETURN adminName;
    -> END $$
Query OK, 0 rows affected (0.006 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> SELECT get_admin_name(101) AS admin_name;
+---------------+
| admin_name    |
+---------------+
| Alice Johnson |
+---------------+
1 row in set (0.007 sec)

MariaDB [library]>
```

🔍 3. Views

Views are virtual tables created by querying one or more tables. They help in **simplifying complex queries** and providing read-only access to data.

✅ **View 1: Displays details of borrowed books along with user names and book titles.**

✅ **View 2: Displays each admin and the total books available under their responsibility.**

📸 ***Screenshot of View SQL Code below***

```
MariaDB [library]> CREATE VIEW vw_borrowed_books AS
    -> SELECT
    ->     U.name AS user_name,
    ->     B.title AS book_title,
    ->     B.status,
    ->     B.place
    -> FROM
    ->     USER U
    -> JOIN
    ->     BORROW BR ON U.library_id = BR.library_id
    -> JOIN
    ->     BOOK B ON BR.book_barcode_no = B.book_barcode_no;
Query OK, 0 rows affected (0.006 sec)

MariaDB [library]> SELECT * FROM vw_borrowed_books;
+--------------+----------------------+-----------+---------+
| user_name    | book_title           | status    | place   |
+--------------+----------------------+-----------+---------+
| David Brown  | Data Structures      | Issued    | Shelf B |
| Eva Green    | C Programming        | Available | Shelf A |
| Frank Wilson | Digital Logic Design | Available | Shelf C |
+--------------+----------------------+-----------+---------+
3 rows in set (0.002 sec)

MariaDB [library]>
```

```
MariaDB [library]> CREATE VIEW vw_admin_book_count AS
    -> SELECT
    ->     A.name AS admin_name,
    ->     BR.total_books_available
    -> FROM
    ->     ADMIN A
    -> JOIN
    ->     BOOK_RECORD BR ON A.admin_id = BR.admin_id;
Query OK, 0 rows affected (0.005 sec)

MariaDB [library]> SELECT * FROM vw_admin_book_count;
+---------------+-----------------------+
| admin_name    | total_books_available |
+---------------+-----------------------+
| Alice Johnson |                   150 |
| Bob Smith     |                   230 |
| Clara Lewis   |                   175 |
+---------------+-----------------------+
3 rows in set (0.002 sec)

MariaDB [library]>
```

⚙️ 4. Stored Procedures

Stored procedures are precompiled SQL statements that can be reused to perform common operations. They are used to **standardize repetitive tasks like inserting new records.**

✅ **Procedure 1: Issues a book by inserting a borrow record and updating the book status.**

✅ **Procedure 2: Returns a book by deleting the borrow record and updating the book status to "Available"**

📸 ***Screenshot of Stored Procedure SQL Code below***

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE PROCEDURE sp_issue_book(
    ->     IN p_library_id INT,
    ->     IN p_book_barcode_no INT
    -> )
    -> BEGIN
    ->     INSERT INTO BORROW(library_id, book_barcode_no)
    ->     VALUES(p_library_id, p_book_barcode_no);
    ->
    ->     UPDATE BOOK
    ->     SET status = 'Issued'
    ->     WHERE book_barcode_no = p_book_barcode_no;
    -> END $$
Query OK, 0 rows affected (0.007 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> CALL sp_issue_book(201, 303);
Query OK, 2 rows affected (0.008 sec)

MariaDB [library]> SELECT book_barcode_no, title, status FROM BOOK WHERE book_barcode_no = 303;
+----------------+---------------------+--------+
| book_barcode_no | title               | status |
+----------------+---------------------+--------+
|             303 | Digital Logic Design | Issued |
+----------------+---------------------+--------+
1 row in set (0.001 sec)

MariaDB [library]>
```

```
MariaDB [library]> DELIMITER $$
MariaDB [library]>
MariaDB [library]> CREATE PROCEDURE sp_return_book(
    ->     IN p_library_id INT,
    ->     IN p_book_barcode_no INT
    -> )
    -> BEGIN
    ->     DELETE FROM BORROW
    ->     WHERE library_id = p_library_id AND book_barcode_no = p_book_barcode_no;
    ->
    ->     UPDATE BOOK
    ->     SET status = 'Available'
    ->     WHERE book_barcode_no = p_book_barcode_no;
    -> END $$
Query OK, 0 rows affected (0.006 sec)

MariaDB [library]>
MariaDB [library]> DELIMITER ;
MariaDB [library]> CALL sp_return_book(201, 303);
Query OK, 2 rows affected (0.005 sec)

MariaDB [library]> SELECT book_barcode_no, title, status FROM BOOK WHERE book_barcode_no = 303;
+-----------------+----------------------+-----------+
| book_barcode_no | title                | status    |
+-----------------+----------------------+-----------+
|             303 | Digital Logic Design | Available |
+-----------------+----------------------+-----------+
1 row in set (0.001 sec)

MariaDB [library]>
```

Section 4

👷 Application Demonstration

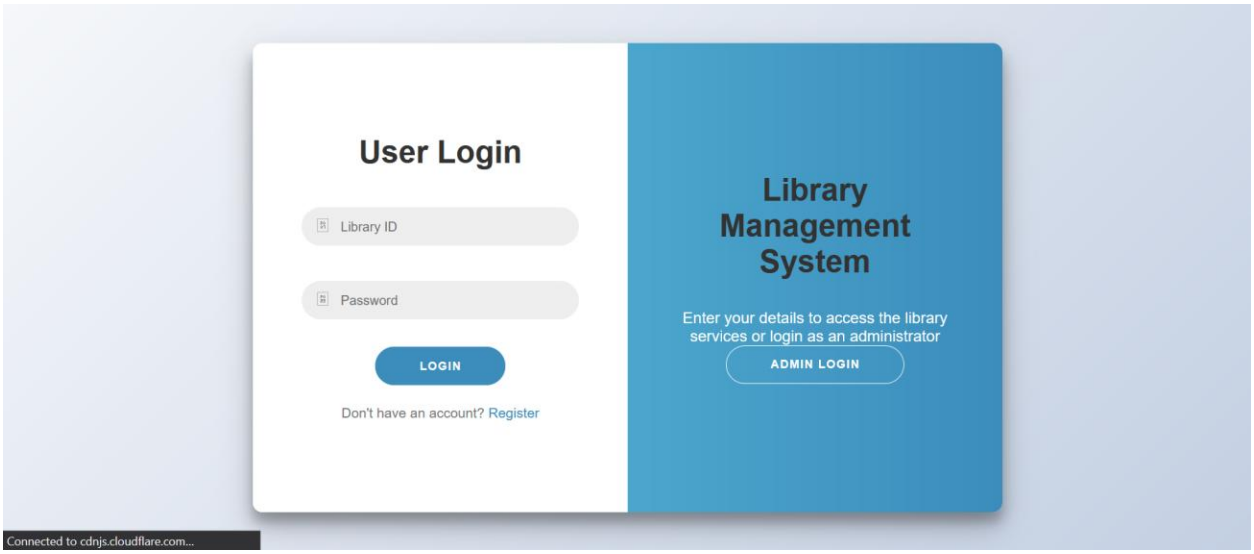
🛠️ **Source code : -**
https://drive.google.com/drive/fold

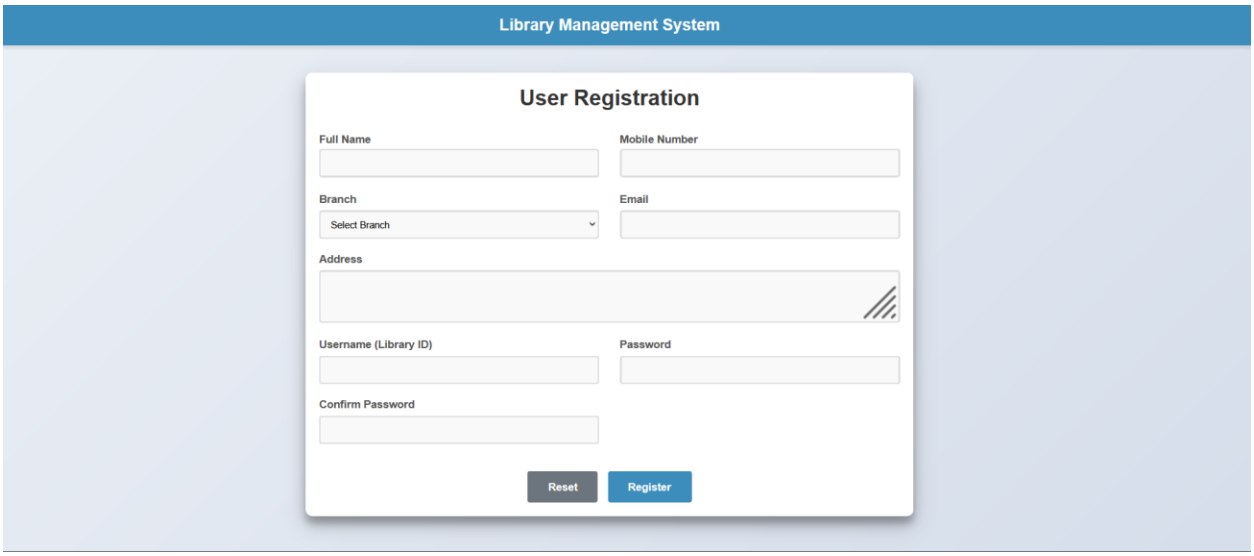
As part of the assignment requirement, I have developed a simple web-based application using the following technologies:

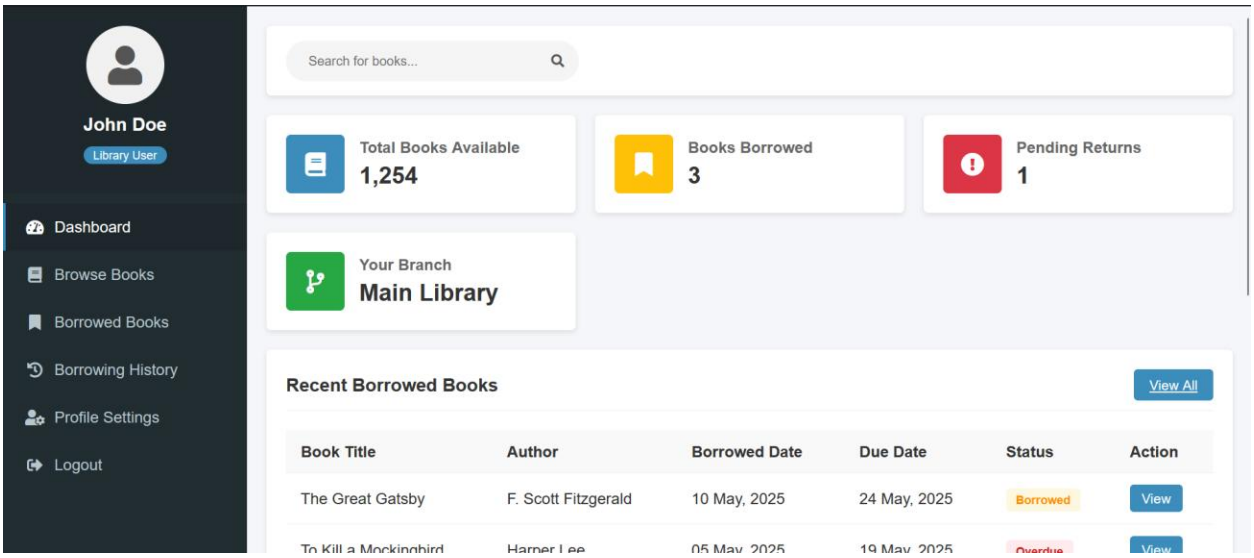
- **Frontend**: HTML, CSS, JavaScript
- **Backend**: PHP
- **Database**: MySQL (accessed via XAMPP)

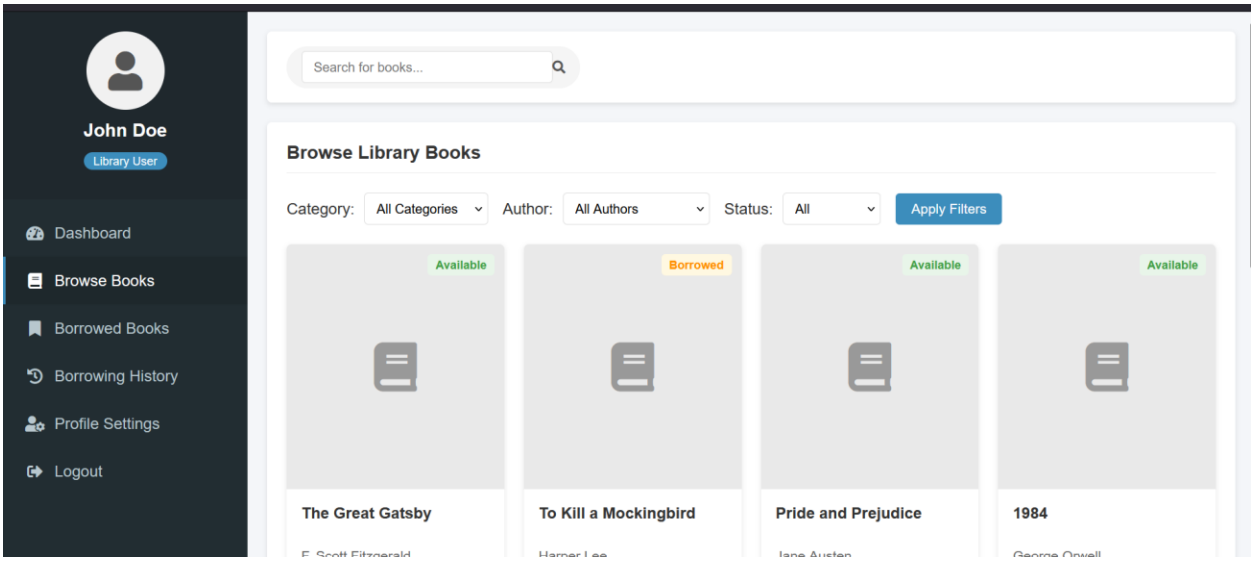
This application showcases a **single selected database operation** to demonstrate the integration of a user interface with a database system

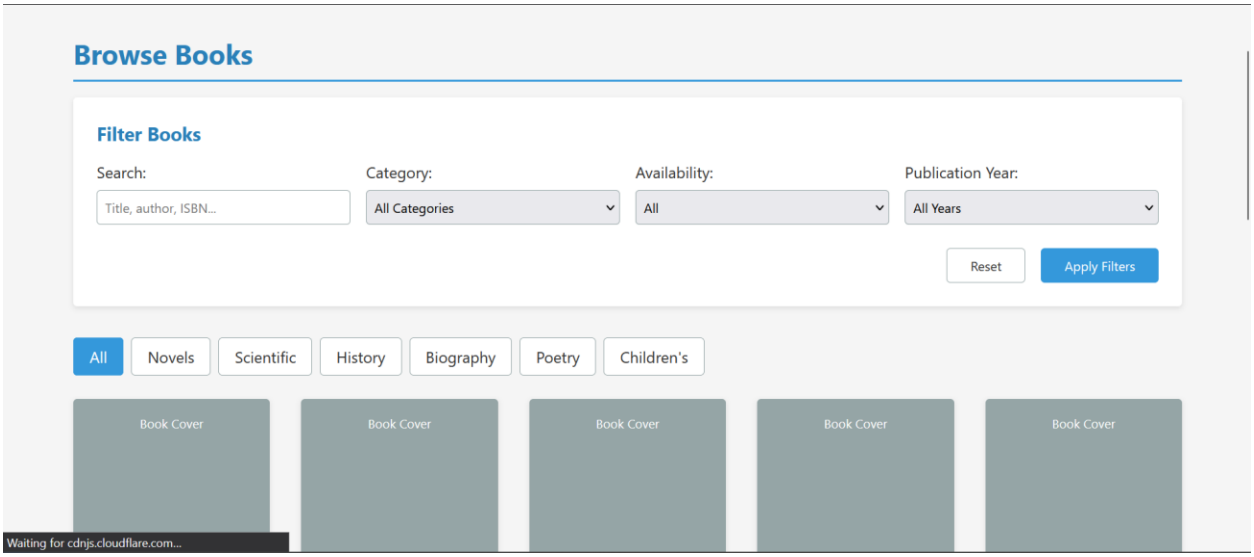
User view



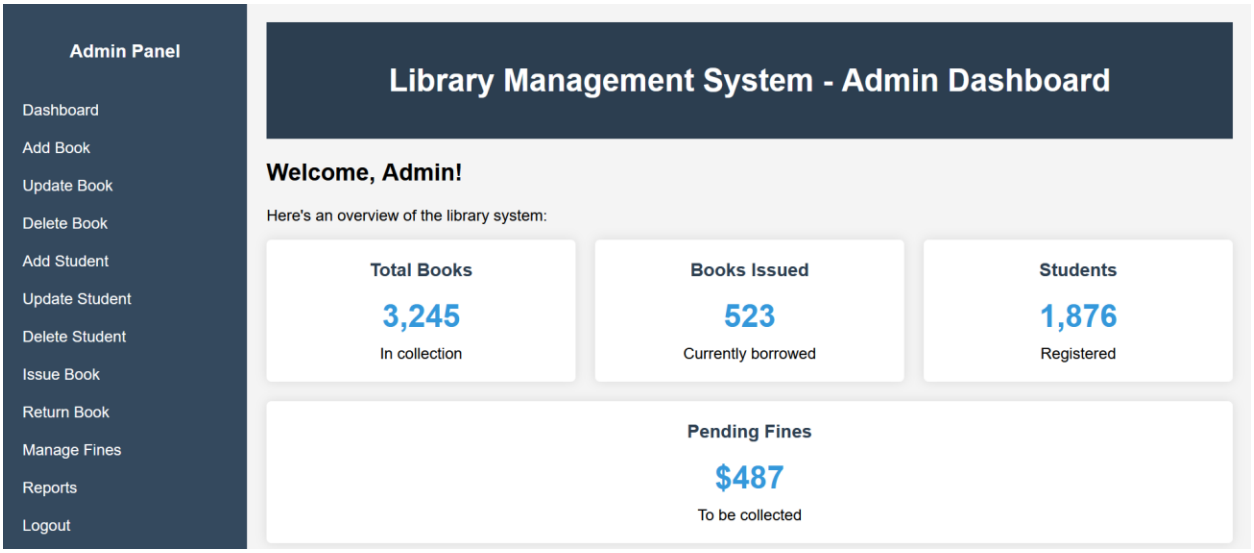

Library Management System
User Registration
Full Name
Mobile Number
Branch
Select Branch
Email
Address
Username (Library ID)
Password
Confirm Password
Reset
Register

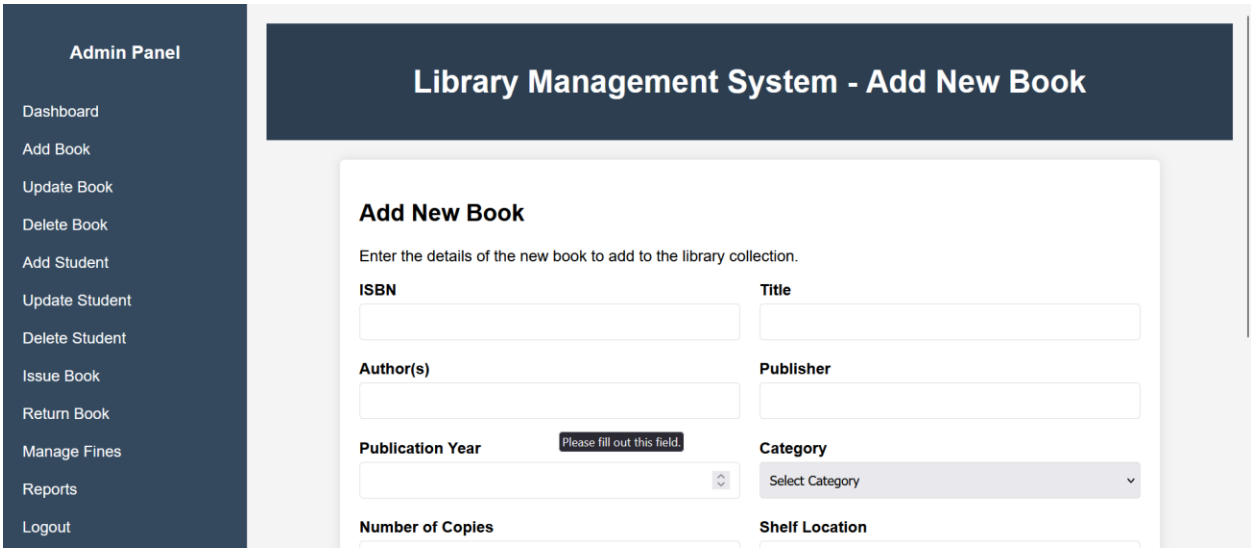

John Doe
Library User
Dashboard
Browse Books
Borrowed Books
Borrowing History
Profile Settings
Logout
Search for books...
Total Books Available
1,254
Books Borrowed
3
Pending Returns
1
Your Branch
Main Library
Recent Borrowed Books
View All
Book Title
Author
Borrowed Date
Due Date
Status
Action
The Great Gatsby
F. Scott Fitzgerald
10 May, 2025
24 May, 2025
Borrowed
View

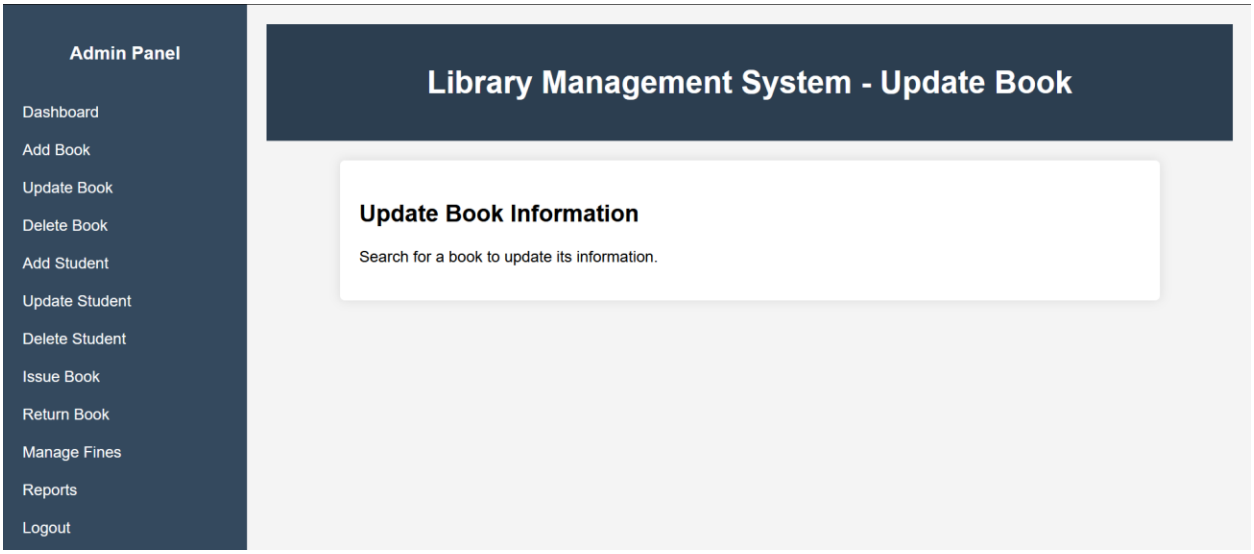

John Doe
Library User
Dashboard
Browse Books
Borrowed Books
Borrowing History
Profile Settings
Logout
Search for books...
Browse Library Books
Category:
All Categories
Author:
All Authors
Status:
All
Apply Filters
Available
Borrowed
Available
Available
The Great Gatsby
To Kill a Mockingbird
Pride and Prejudice
1984

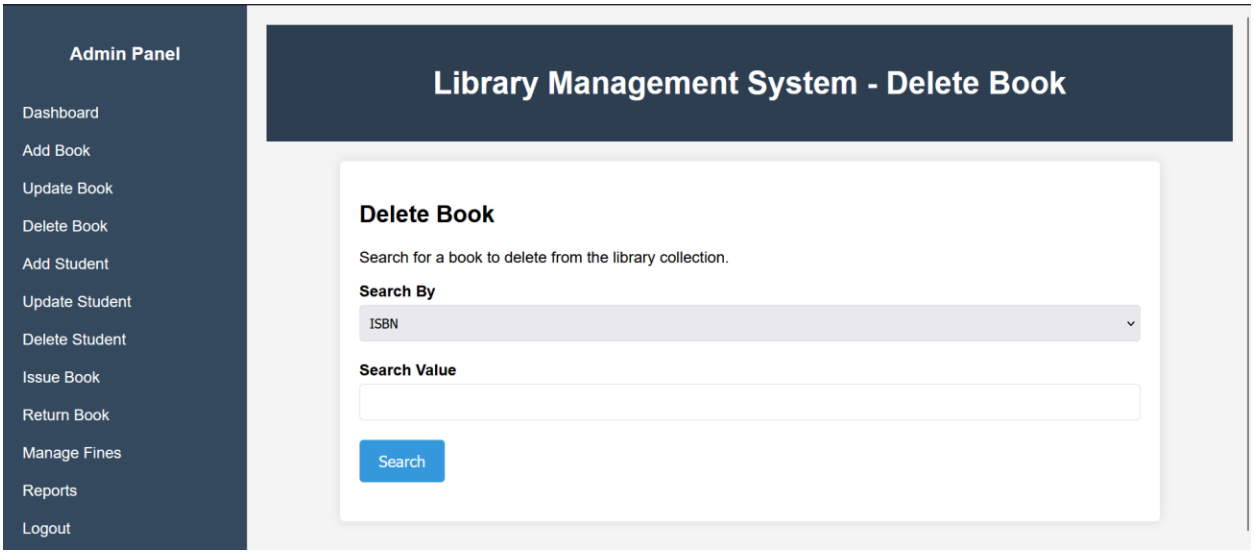

Browse Books
Filter Books
Search:
Title, author, ISBN...
Category:
All Categories
Availability:
All
Publication Year:
All Years
Reset
Apply Filters
All
Novels
Scientific
History
Biography
Poetry
Children's
Book Cover
Book Cover
Book Cover
Book Cover
Book Cover
Waiting for cdnjs.cloudflare.com...

Admin view


Admin Panel
Dashboard
Add Book
Update Book
Delete Book
Add Student
Update Student
Delete Student
Issue Book
Return Book
Manage Fines
Reports
Logout
Library Management System - Admin Dashboard
Welcome, Admin!
Here's an overview of the library system:
Total Books
3,245
In collection
Books Issued
523
Currently borrowed
Students
1,876
Registered
Pending Fines
$487
To be collected


Admin Panel
Dashboard
Add Book
Update Book
Delete Book
Add Student
Update Student
Delete Student
Issue Book
Return Book
Manage Fines
Reports
Logout
Library Management System - Add New Book
Add New Book
Enter the details of the new book to add to the library collection.
ISBN
Title
Author(s)
Publisher
Publication Year
Please fill out this field.
Category
Select Category
Number of Copies
Shelf Location



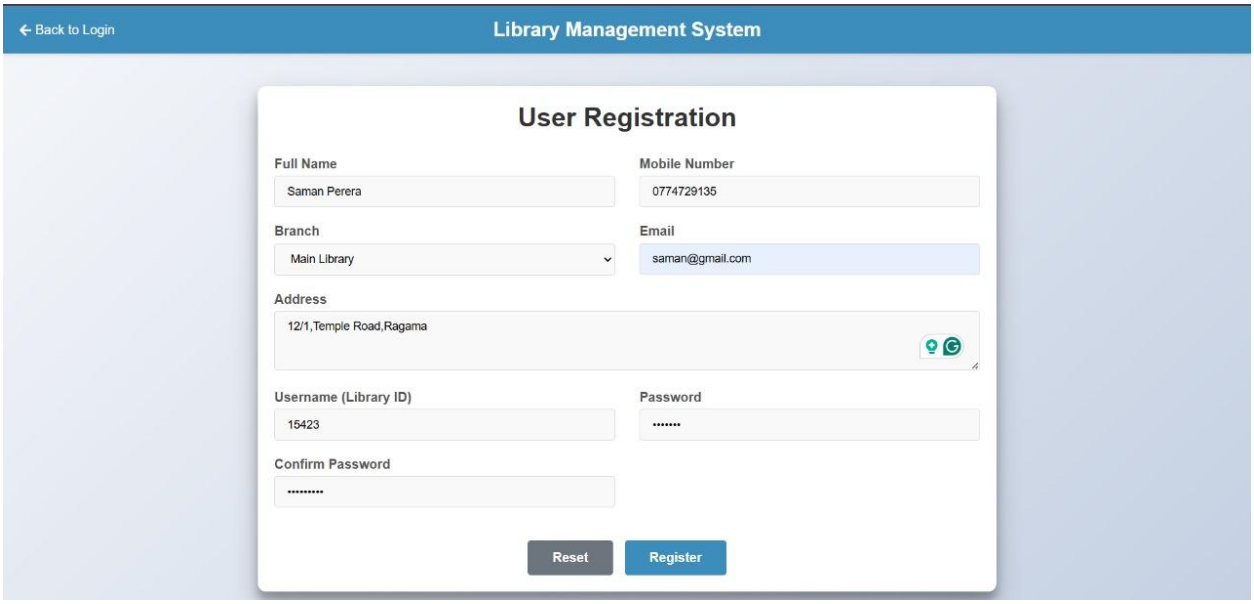


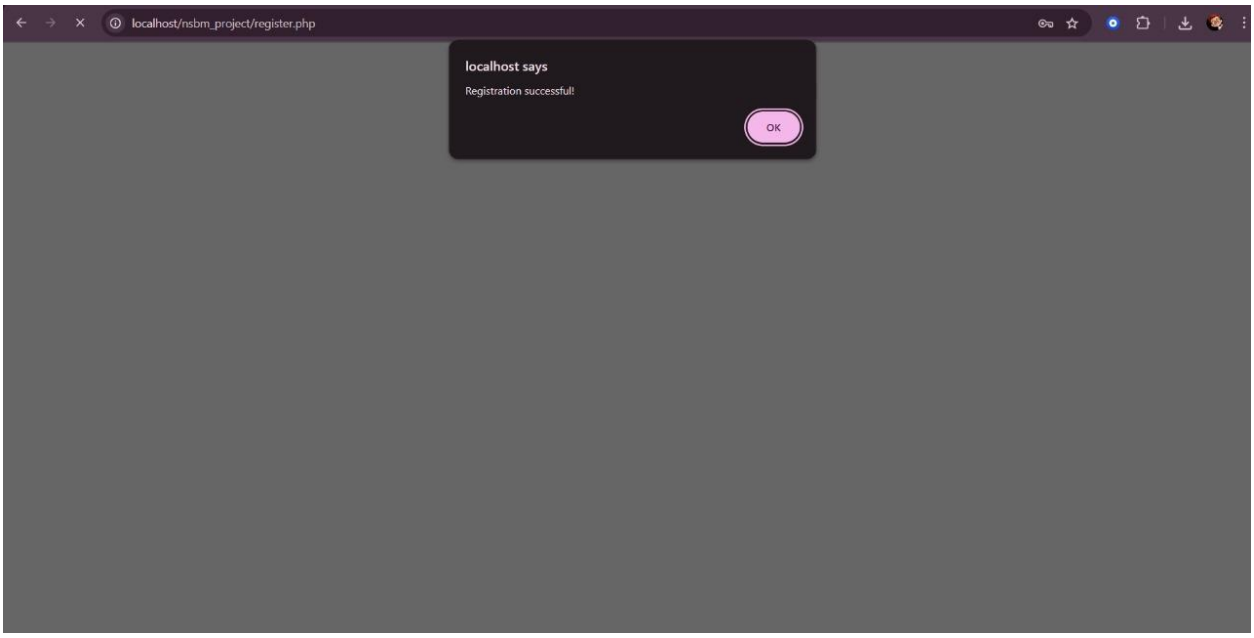
Below are the screenshots showing the different steps of the application, including:

1. **User Interface (Form/Page)**
2. **Submitted Data**
3. **Database Update (Verified via phpMyAdmin)**

🧪 Database Operation Demonstration

Selected Operation: [Insert Operation] - *Register a new user*






```
MariaDB [(none)]> use library;
Database changed
MariaDB [library]> select * from user;
+------------+----------+--------------+-------------+------------+----------------+-----------------------+--------------------+--------+
| Library_ID | Login_ID | Name         | Branch      | Mobile_No  | Email          | Address               | No_of_Books_Issued | Status |
+------------+----------+--------------+-------------+------------+----------------+-----------------------+--------------------+--------+
| 15423      |        3 | Saman Perera | Main Library | 0774729135 | saman@gmail.com | 12/1,Temple Road,Ragama |                  0 | Active |
+------------+----------+--------------+-------------+------------+----------------+-----------------------+--------------------+--------+
1 row in set (0.002 sec)
```

This operation was chosen to demonstrate how frontend inputs are captured, validated, and passed to the backend for processing, and how PHP interacts with the MySQL database to perform real-time data manipulation.

Using Power BI for Advanced Database Analytics and Business Intelligence

In this section, we demonstrate how to leverage **Power BI** to perform advanced analytics and Business Intelligence (BI) operations using the existing MySQL database. Power BI is a powerful data visualization tool that connects directly to databases, allowing users to create interactive reports and dashboards with minimal effort.

Step 1: Connecting Power BI to MySQL
We first established a connection between Power BI Desktop and the MySQL database (libaraysystem) using the MySQL ODBC connector. This connection enabled us to import tables such as books, authors, copies, and users into Power BI for analysis.

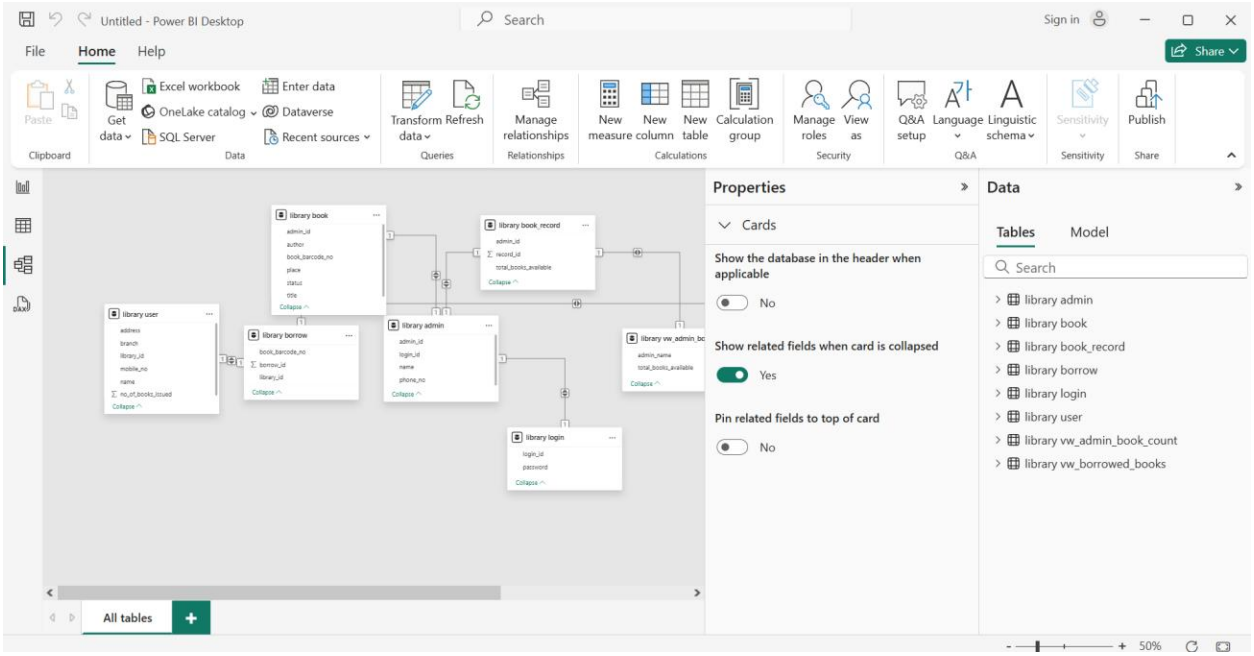
Step 2: Importing and Modeling Data
After importing the necessary tables, we verified and created relationships between them within Power BI's Model view. This step ensured that the data could be accurately related and aggregated across different entities, such as linking books to their authors and copies.

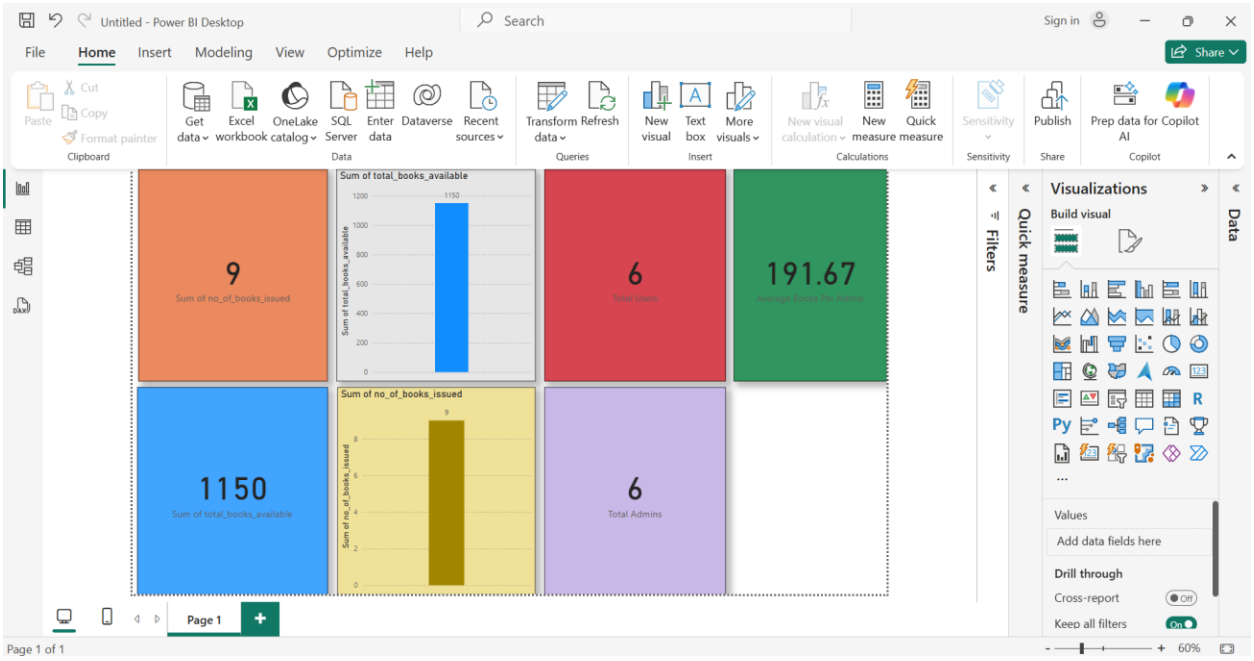
Step 3: Creating Visual Reports
We built multiple visualizations to gain insights from the database, including

- **Book Availability Report:** Displays the total number of available copies for each book.
- **User Distribution Report:** Shows active users categorized by their faculties.
- **Loan Trends:** Analyzes borrowing patterns over time, identifying peak periods of book issues.

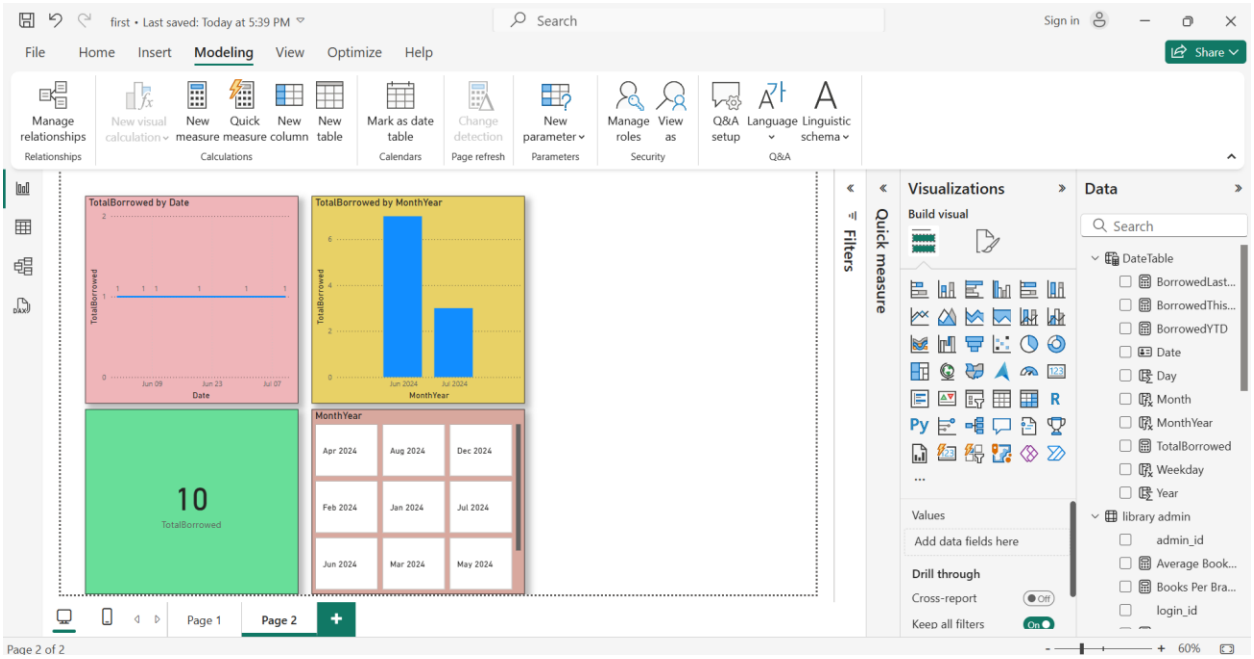
These interactive reports allow library administrators to monitor resource availability, user engagement, and borrowing trends effectively.

Step 4: Utilizing Advanced Analytics Features
Using Power BI's DAX language, we created calculated measures such as the count of available book copies. We also applied time intelligence functions to analyze data trends over different periods, enabling predictive insights and strategic decision-making.


Untitled - Power BI Desktop
Search
Sign in
File
Home
Help
Share
Properties
Cards
Show the database in the header when applicable
No
Show related fields when card is collapsed
Yes
Pin related fields to top of card
No
Data
Tables
Model
Search
library admin
library book
library book_record
library borrow
library login
library user
library vw_admin_book_count
library vw_borrowed_books
All tables
50%


Untitled - Power BI Desktop
Search
Sign in
File
Home
Insert
Modeling
View
Optimize
Help
Share
Clipboard
Data
Queries
Insert
Calculations
Sensitivity
Share
Copilot
9
Sum of no_of_books_issued
Sum of total_books_available
6
Total Users
191.67
1150
Sum of total_books_available
Sum of no_of_books_issued
6
Total Admins
Filters
Quick measure
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Data
Page 1
Page 1 of 1
60%

Advanced Analytics Features Using Power BI (Time Intelligence)



Overall, integrating Power BI with the MySQL database provides a robust solution for Business Intelligence, transforming raw data into meaningful information for better library management.

Conclusion

In this assignment, we successfully designed and implemented a comprehensive library management database system using MySQL. We created essential database objects including tables with well-defined relationships and constraints, ensuring data integrity and consistency.

Through the development of **triggers**, **user-defined functions**, **views**, and **stored procedures**, we enhanced the functionality, automation, and reusability of the database. These components allowed us to enforce business rules, simplify complex queries, and standardize repetitive operations.

Furthermore, we demonstrated the practical application of the database by developing a simple web-based interface using HTML, CSS, JavaScript, and PHP to interact with the system, showcasing key operations such as adding books and registering users.

To extend the system's capabilities into Business Intelligence, we connected the MySQL database with Power BI. This integration enabled us to create interactive dashboards and insightful reports, empowering stakeholders to make data-driven decisions regarding library resources, user engagement, and operational efficiency.

Overall, this project not only provided hands-on experience in database design and programming but also highlighted the importance of combining traditional database management with advanced analytics tools for effective business intelligence. This holistic approach ensures that the library system is not only functional but also strategically valuable.

Here is our team and tasks we done

Team Leader - M.H.M.P.D.Herath(29037)

Student Name	Registration Number	Tasks done
M.H.M.P.D.Herath	29037	Project Coordination and App Dashboard Design
L.D.D.Liyanaarachchi	29302	ER Diagram and App Registration Page
H.P.H.Dulshan	29315	Assumptions & Relational Mapping and App Profile Page
M.K.S.De.Silva	28302	Data Normalization and App Home Page
O.N.Samarasinghe	30658	Data Dictionary and App Validation Handling
R.P.T.G.Madushanka	28547	SQL Table Creation and App Login Page
R.P.S.G.Madushanka	28551	Sample Records & DB Diagram and App Record Display Page
R.R.Chandrasekara	28735	Triggers & Functions and App Transaction Page
S.B.M.I.P.Bandara	19925	Views & Procedures and Power BI integration