Mehran University of Engineering & Technology
Department of Computer Systems Engineering

CEP Project Report

Library Management System Using Streamlit Python and MySQL Workbench

Names: **Adeel Shah, M. Sameer, Mashaal Zulfikar**

Roll No.s: **21CS041, 21CS077, 21CS105**

Class: **21CS-1**

Submitted to: **Ma'am Zartasha Baloch**

DBMS 5th Semester 21CS Project

A Library Management System Using Streamlit Python & MySQL Workbench

Group Members

Muhammad Adeel Shah	Muhammad Sameer	M. Mashaal Zulfikar
21CS041	21CS077	21CS105

Introduction

The Library Management System (LMS) project aims to develop an efficient software solution to streamline library operations and enhance user experience for both librarians and patrons. This report provides a comprehensive overview of the project, including objectives, processes, system functionalities, and implementation details.

Objectives

The primary objectives of the Library Management System project are:

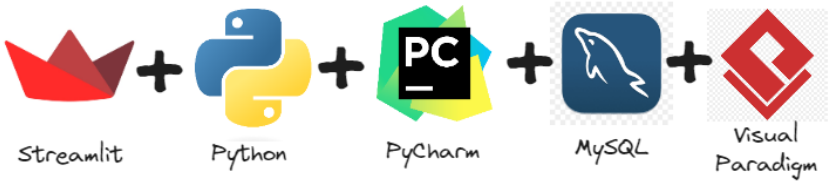
1. To automate and digitise library operations, including book cataloguing, borrowing, and return processes.
2. To provide an intuitive and user-friendly interface for librarians and patrons to access and manage library resources.
3. To enhance accessibility to library resources and improve overall efficiency in library management.
4. To generate reports and insights into library usage patterns for better decision-making and resource allocation.

In today's digital era, where libraries face challenges such as book management, user engagement, and resource accessibility, the Library Management System project endeavours to address these issues by leveraging technology to streamline operations and enhance user satisfaction.

Resources Used

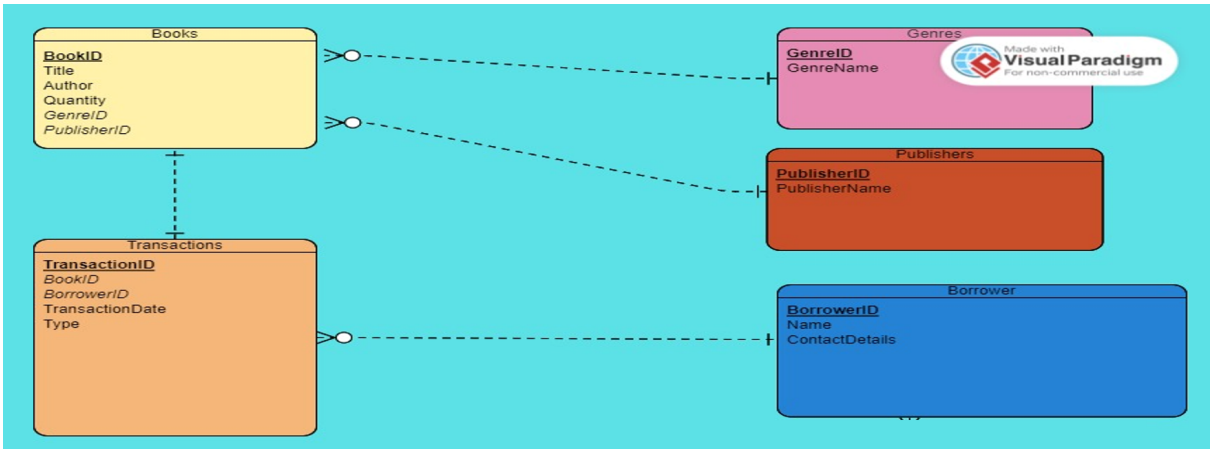
The following softwares/applications have been used to create the Library Management System:

- **Streamlit**: A frontend Python library for creating interactive web applications with simple scripts. Used as the interface of the library management system.
- **Python**: A high-level programming language used for frontend as well as backend logic, including user authentication, database interactions, and business logic in the library management system.
- **PyCharm IDE**: An integrated development environment (IDE) for Python development, providing features like code editing and debugging. Can be used to write and manage Python code in the project.
- **MySQL**: A relational database management system used for backend optimization and logic to store and manage data related to users, books, and borrowing history in the library management system.
- **Visual Paradigm**: A software design and modeling tool used for designing the database schema and creating visual representations of the system's architecture, aiding in planning and visualization before implementation.



ER Diagram

The following ER Diagram displays all the schemas and tables that have been created in this project with the aid of MySQL Workbench and Visual Paradigm. The ER model diagram illustrates the relationships between entities in the Library Management System database. It includes entities such as Books, Genres, Publishers, Borrowers, and Transactions, along with their attributes and relationships.



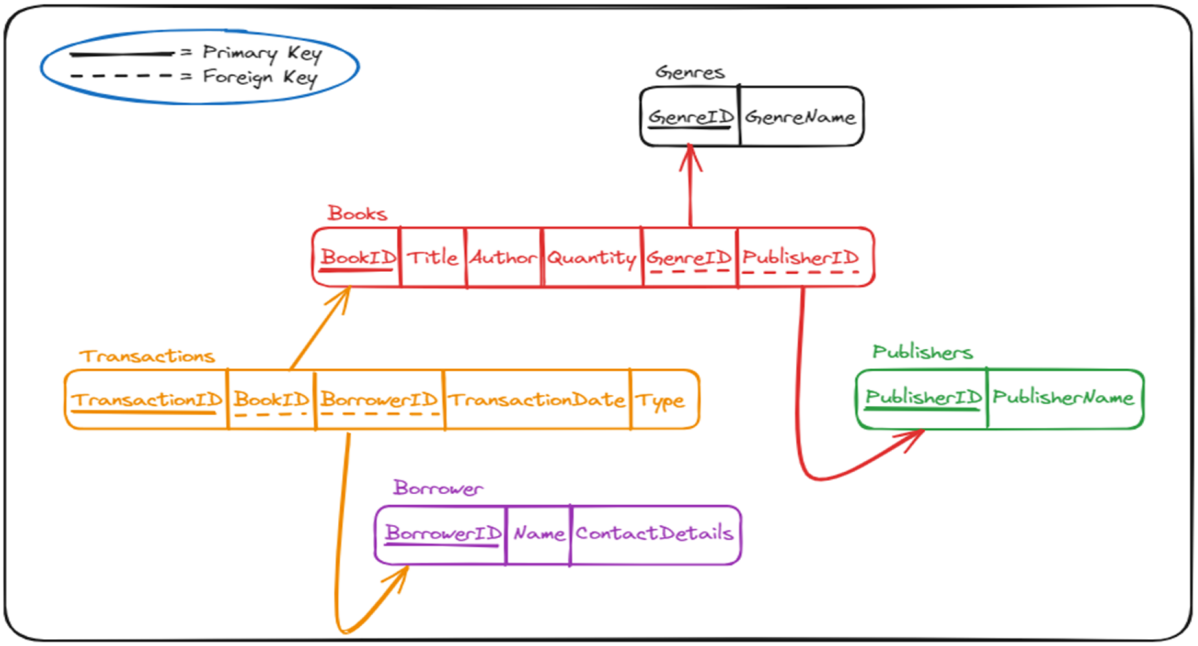
Here's a breakdown of the entity-relationships cardinality:

Title	Relation Type	Description
Books - Genres	Many to One	Multiple Books can share a same Genre. A single book can not consist of multiple Genres.
Books - Publishers	Many to One	Multiple Books can share a same Publisher. A single book can not consist of multiple Publishers,
Books - Transactions	One to Many	A single book may be involved in multiple transactions. Similarly, multiple transactions may involve a single particular book being borrowed or returned simultaneously.
Transactions - Borrowers	Many to One	Multiple Book Transactions can be made by a single person. A single particular transaction can not be done by multiple users.

ER Mapping

ER Mapping in the context of this project involves defining the relationships and structures within the Library Management System database. The interconnections between entities such as Books, Genres, Publishers, Borrowers, and Transactions are outlined. Each entity is associated with

attributes, including primary keys that uniquely identify records, and foreign keys that establish relationships between tables. This mapping elucidates the database schema, facilitating efficient management of library resources by ensuring data integrity and relational integrity.



Here's a breakdown of the relationships:

- **Books:** Books have a title, author, quantity, and a unique identifier **PRIMARY KEY**(BookID). They are also associated with a genre **FOREIGN KEY**(GenreID) and a publisher **FOREIGN KEY**(PublisherID).
- **Publishers:** Publishers have a name and a unique identifier **PRIMARY KEY**(PublisherID).
- **Borrowers:** Borrowers have a name, contact details, and a unique identifier **PRIMARY KEY**(BorrowerID). They can borrow books through transactions.
- **Transactions:** Transactions have a unique identifier **PRIMARY KEY**(TransactionID), a date (TransactionDate), and a type (Type). They link borrowers **FOREIGN KEY**(BorrowerID) to books **FOREIGN KEY**(BookID).
- **Genres:** Genres have a name (GenreName) and a unique identifier **PRIMARY KEY**(GenreID). They are associated with books **FOREIGN KEY**(BookID).

Catering to Diverse Stakeholders' Needs

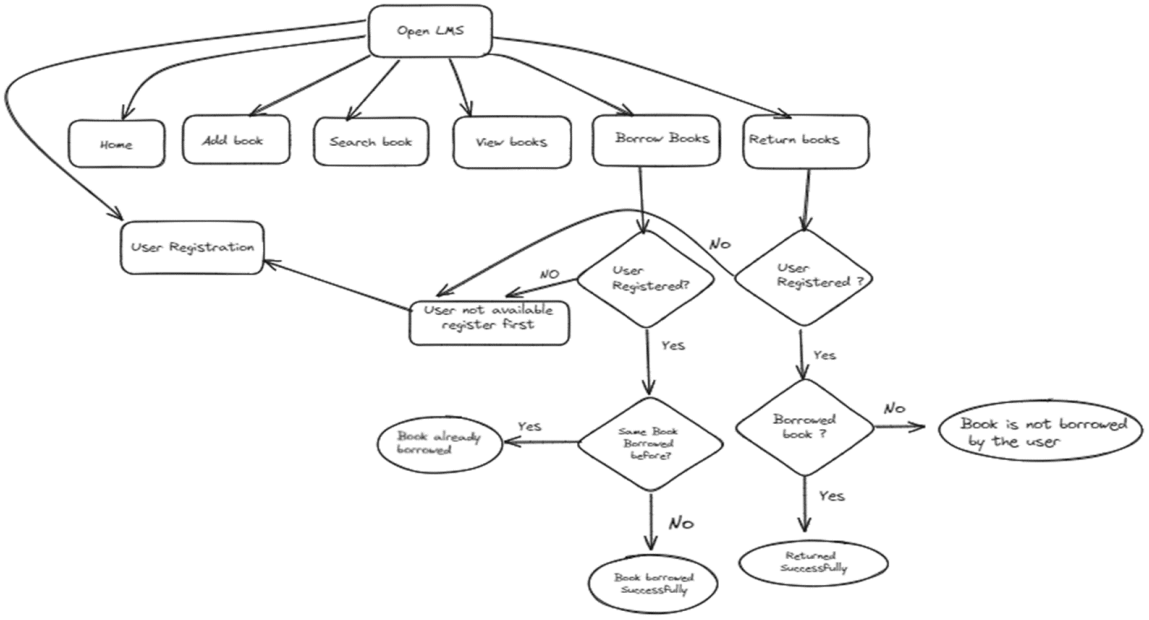
Our Library Management System (LMS) is created to improve library functions and user experiences for different stakeholders. It provides a wide array of features customized to meet the requirements of various users such as librarians, and students. With our system, stakeholders can easily perform tasks like managing book collections, borrowing, and returning books, all through a user-friendly interface.

Stakeholder Access and Features:

Stakeholder	Features/Accessibilities	Interface
Librarian	- Manage book collections - Handle borrower information - View transaction history	User-friendly dashboard with management controls
Student	- Search and view available books - Borrow and return books	Simple and intuitive interface with search and borrowing functionalities

This system ensures that each stakeholder has access to the necessary features and functionalities to perform their tasks efficiently while providing a seamless and enjoyable experience.

Execution Flowchart of the Library Management System



The flowchart of a library management system shows the different steps a user can take to interact with the system. The flowchart starts with a box labelled "Open LMS". The flowchart splits into six main paths: Home, Add Book, Search Book, View Books, Borrow Books, and Return Books.

The "Borrow Books" path leads to two decision one after the other in a sequence:

- "Some Book Borrowed before?": This decision checks whether the user has or has not borrowed any books before. The user then has the access to borrow.
- "Book borrowed by the user?": If the book has already been borrowed by the user, the system will not allow them to borrow it again.

The flowchart has a box labelled "Borrowed Successfully" in the end if the user borrows a book successfully.

Similarly, the flowchart has also illustrated the path of returning books labelled "Return Books" that follows the decisions "User Registered?", and "Borrowed Book?". This path would likely lead to a check for the specific book being returned and potentially an update to the book's availability status in the system. The flowchart would then conclude with a "Return Successful" box upon successful completion.

Analysing Library Management System Functionalities using DBMS Course Concepts

Our Library Management System is a cutting-edge software that modernizes library operations and enhances the library experience for both librarians and users. Our system offers a wide range of functionalities aimed at simplifying day-to-day tasks and improving access to library resources by providing librarians with actionable insights.

Understanding Backend MySQL Queries

Complete MySQL Queries File:
https://www.dropbox.com/scl/fi/kd2mohn75bfndbwx0r4ai/Queries.txt?rlkey=n8ztff5oked4×9sczw63d5glk&dl=0

Explanation of each part of the query:

1. **Database Creation and Table Definitions**:

- A database named `library` is created.
- Tables for `Books`, `Borrowers`, `Transactions`, `Genres`, and `Publishers` are created to store information about books, borrowers, transactions, genres, and publishers respectively.

2. **Normalization**:
   - The schema follows normalization principles by breaking down data into multiple related tables to avoid redundancy and maintain data integrity.
   - For example, the `Books` table contains information about books such as title, author, quantity, etc. while the `Genres` and `Publishers` tables store specific information about book genres and publishers respectively. This helps in avoiding data duplication and ensures consistency.
3. **Referential Integrity**:
   - Foreign key constraints are used to establish relationships between tables.
   - For instance, the `Transactions` table references the `BookID` from the `Books` table and the `BorrowerID` from the `Borrowers` table, ensuring that only valid book and borrower IDs can be inserted into the `Transactions` table.
4. **Entity Relationships**:
   - One-to-Many relationships are established between tables like `Books` and `BookCopies`, `Books` and `Transactions`, and `Borrowers` and `Transactions`.
   - For example, one book can have multiple copies (one-to-many relationship between `Books` and `BookCopies`), and one borrower can make multiple transactions (one-to-many relationship between `Borrowers` and `Transactions`).
5. **Data Integrity Constraints**:
   - Unique constraints are applied to columns such as `Title` and `Author` in the `Books` table to prevent duplicate entries.
6. **Data Insertion**:
   - Sample data is inserted into the `Genres` and `Publishers` tables to populate them with predefined values.
7. **User Management**:

- A `Users` table is created to manage user accounts with different access levels (admin, librarian, student).
- Sample data for admin, librarian, and student users are inserted into the `Users` table.

8. **Security**:
   - Passwords are stored securely in the `Users` table, although in a real-world scenario, it's recommended to use hashing and salting techniques for better security.
9. **Column Alteration**:
   - Additional columns are added to the `Borrowers` table to accommodate more information, such as `BookID`, `Type`, and `Noofbooks`.
   - Foreign key constraint (`fk_BookID`) is added to ensure referential integrity between the `BookID` column in the `Borrowers` table and the `BookID` column in the `Books` table.

Understanding Python Code

Complete Python Code File: https://www.dropbox.com/scl/fi/2q4yhqo6gjvuvu1wmecf6/main-python.txt?rlkey=59rcshl3xbt7utw4i3np9qry4&dl=0

Explnation of the Python Code:

1. **Imported Libraries**:
   - `streamlit`: A Python library used for building web applications with interactive elements.
   - `mysql.connector`: A Python driver for connecting to MySQL databases.
   - `pandas`: A powerful data manipulation library used for data analysis and manipulation.
2. **Database Connection**:
   - The code establishes a connection to the MySQL database named "library" using the provided credentials (host, user, password, database).
   - If the connection is successful, it prints "Connected to the database".
3. **Streamlit App**:

- The code sets up a Streamlit application with a title and a sidebar to select different options.
- The options include "Home", "Add Book", "View Books", "Search Books", "Borrow Book", "Return Book", "Borrowers List", "User Registration", and "Registered Users".
- Based on the selected option, different functionalities are displayed.

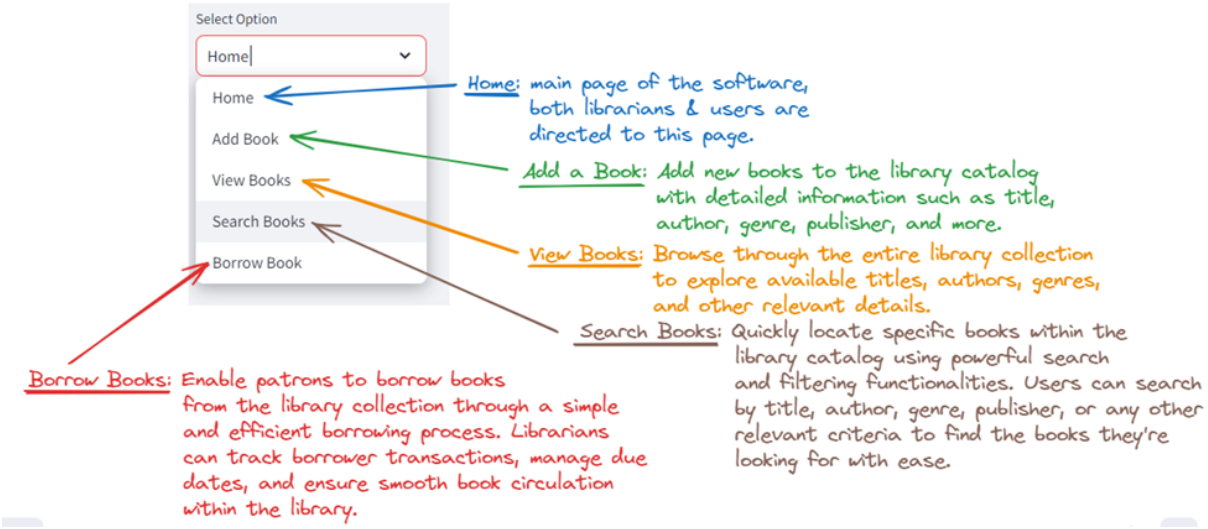
4. **Functions**:
   - `fetch_genres()`: Fetches genres data from the database.
   - `fetch_publishers()`: Fetches publishers data from the database.
   - `execute_query2()`: Executes SQL queries without returning any result.
   - `execute_query()`: Executes SQL queries and returns the result.
   - `add_book()`: Adds a new book to the database.
   - `registered_users()`: Displays registered users from the database.
   - `login()`: Handles user login functionality.
   - `user_registration()`: Handles user registration functionality.
   - `view_books()`: Displays books stored in the database.
   - `search_books()`: Searches for books based on provided criteria.
   - `return_book()`: Handles book return functionality.
5. **Streamlit UI**:
   - Each option in the sidebar corresponds to a specific functionality, such as adding a book, viewing books, searching books, borrowing and returning books, displaying borrowers list, registering users, and displaying registered users.
   - Depending on the selected option, appropriate UI elements are displayed using Streamlit's interactive widgets like text inputs, buttons, select boxes, etc.
6. **Database Operations**:
   - Various database operations are performed within the Streamlit app, such as adding, viewing, searching, and returning books, displaying borrowers list, registering users, etc.

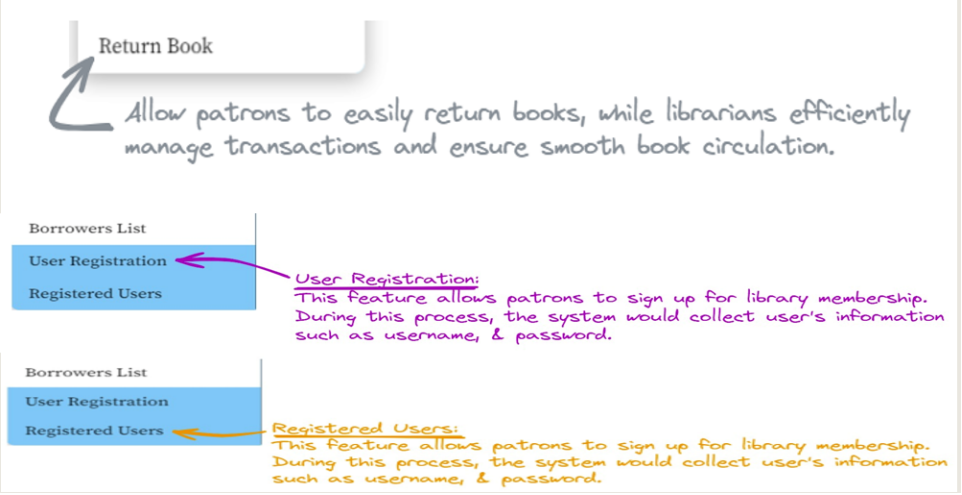
- SQL queries are executed to interact with the MySQL database using the `execute_query()` function.

7. **Closing Connection**:
   - Finally, the connection to the database is closed after the Streamlit app completes its execution.

Final Interface Result

The Navigation Bar:





Homepage:

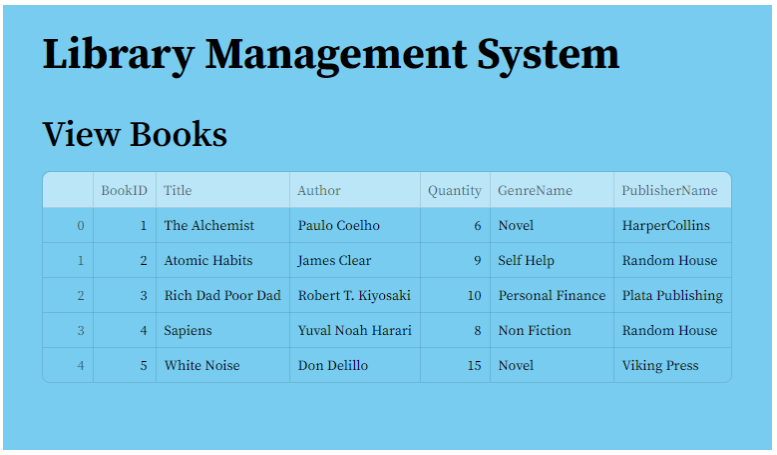
- Provides a welcoming message and an introduction to the library management system, includes an image related to libraries or books for visual appeal.

Add Book:

- Form to input book details: title, author, quantity, genre, and publisher.

- Dropdown menus for genre and publisher selection.
- Submit button to add the book to the database.
- Error handling for incomplete or incorrect input.

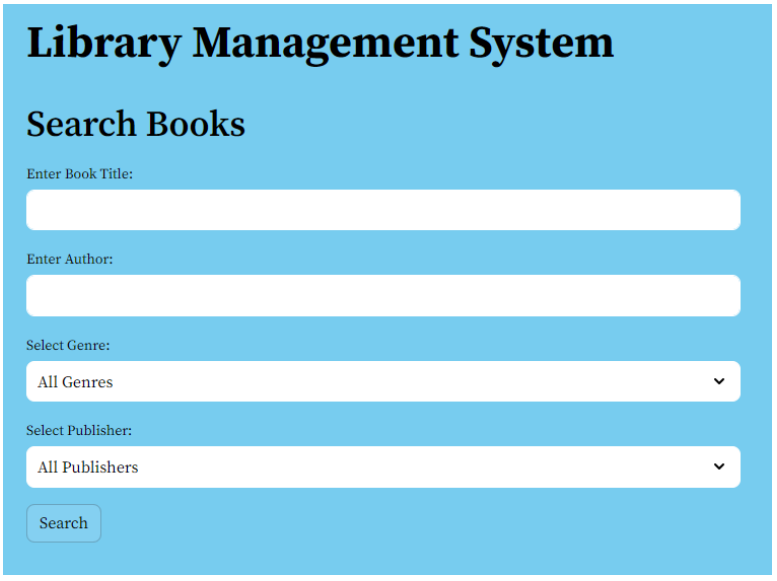
View Books:



	BookID	Title	Author	Quantity	GenreName	PublisherName
0	1	The Alchemist	Paulo Coelho	6	Novel	HarperCollins
1	2	Atomic Habits	James Clear	9	Self Help	Random House
2	3	Rich Dad Poor Dad	Robert T. Kiyosaki	10	Personal Finance	Plata Publishing
3	4	Sapiens	Yuval Noah Harari	8	Non Fiction	Random House
4	5	White Noise	Don Delillo	15	Novel	Viking Press

- Table displaying all books stored in the database.
- Columns for BookID, title, author, quantity, genre, and publisher.
- Pagination or scrolling for large book collections.
- Sorting options for different criteria.

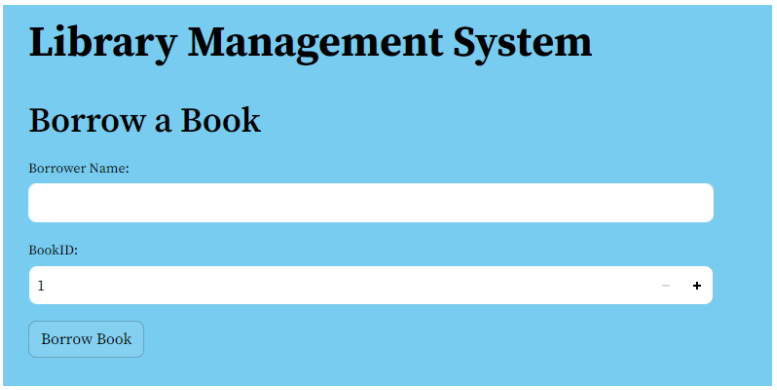
Search books:



- Input fields for search criteria: title, author, genre, and publisher.

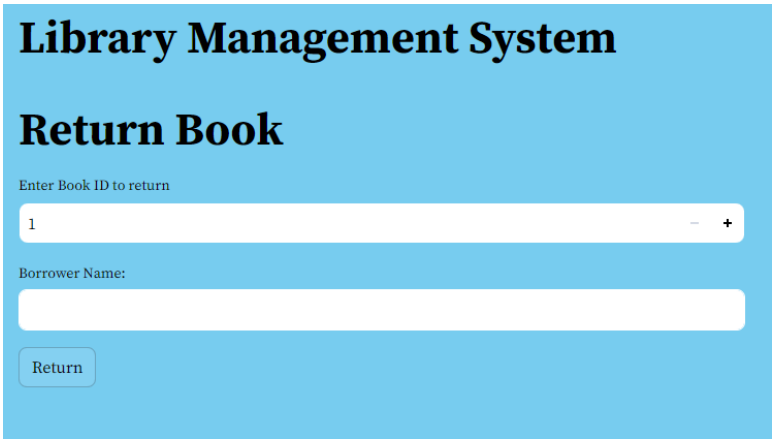
- Dropdowns for genre and publisher selection.
- Search button to initiate the search.
- Display of search results in a table format.

Borrow Books:



- Input fields for borrower's name and BookID.
- Button to submit borrowing request.
- Validation for borrower registration and book availability.
- Error messages for issues with borrowing process.

Return Book



- Input field for BookID of the book to return.
- Button to submit return request.
- Validation for borrowed book and user association.
- Confirmation message upon successful return.

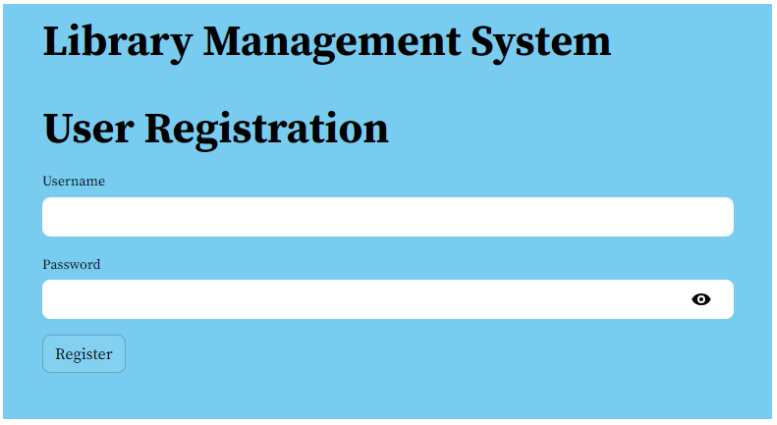
Borrowers List (1NF: Concept of Normalization):

	UserID	Name	BookName
0	1	adeel	White Noise
1	1	adeel	Sapiens
2	2	henry	Rich Dad Poor Dad
3	2	henry	Sapiens

- Table listing borrowers and number of books borrowed.
- Columns: BorrowerID, Name, No. of books borrowed.
- Each row represents a unique borrower, with count based on BorrowerID.

```
# (1NF: Normalization Concept)
query2 ="""
    SELECT Users.UserID, Borrowers.Name, books.Title
AS BookName
    FROM Borrowers
    JOIN Users ON Borrowers.Name = Users.Username
    JOIN books ON Borrowers.BookID = books.bookid;
    """
```

User Registration:



- Input fields for username and password.
- Dropdown for user type selection.
- Button to register user.
- Validation for unique username and non-empty fields.
- Success message upon successful registration.

Conclusion

The "Library Management System" project utilizes Streamlit, Python, and MySQL Workbench to create an intuitive web application for efficient library operations. Key features include book management (addition, viewing, searching), borrower management, user registration, and borrowing/returning books. The system adheres to normalization principles for data integrity and includes user-friendly interfaces for seamless navigation. Future enhancements may include user authentication, advanced search capabilities, and performance optimization. Overall, the project contributes to digitalizing libraries and enhancing user experiences.

Department of Computer Systems Engineering Mehran University of Engineering and Technology, Jamshoro			
Course: Database Management Systems (CS-353)			
Instructor	Dr. Zartasha Baloch	**Assignment Type**	Complex Engineering Problem
Semester	5th	**Year**	3rd
Submission Deadline	01-03-24	**Assessment Score**	10

Complex Engineering Problem - Characteristics		
1	Depth of knowledge Required	☑
2	Range of Conflicting Requirements	☐
3	Depth of Analysis Required	☑
4	Infrequently Encountered Issues Involved	☐
5	Beyond codes/standards of practice	☐
6	Diverse groups of stakeholders with widely varying needs involved	☑
7	Interdependence (high-level problems including many component parts/sub-problems)	☐
8	Have significant consequences in a range of contexts	☐
9	Judgment (Require judgment in decision making)	☐

Problem Description

Choose a real-world problem, draw the ER Diagram, apply ER-mapping rules to convert it into the relational model, normalize the relations, and follow the application development process. Make sure that the application should have five or more tables, using a suitable frontend tool. Application areas may include but are not limited to; health care, education, industry, transport, supply chain, business, human resource management, manufacturing, etc. Marking will be done on the following parameters:

1. Problem identification
2. ER Diagram
3. ER Mapping
4. Normalization
5. Implementation

Rubrics	Assessment					Marks
	Unacceptable (0)	Poor (2)	Acceptable (5)	Adequate (8)	Proficient (10)	
R1 Identification of constraints/requirements/demands/ research gap or challenges well defined	☐	☐	☐	☐	☐	
R2 Engineering knowledge (standards)	☐	☐	☐	☐	☐	

R3 Efficiency of the solution	☐	☐	☐	☐	☐
R4 Technical Writing	☐	☐	☐	☐	☐
Total Marks					

Rubrics

	Unacceptable	Poor	Acceptable	Adequate	Proficient	Score
R1 Problem/Requirement Identification	Problem not identified	Problem poorly identified	Problem is identified	Problem is defined adequately.	Problem is identified and analyzed in a well-defined manner.	
R2 Engineering knowledge (standards)	Can not apply engineering knowledge to the solution.	Has difficulty applying mathematics to the solution of complex engineering problems	Correctly applies basic sciences to the solution of complex engineering problems	Correctly applies engineering fundamentals to the solution of complex engineering problems	Correctly applies engineering specialization to the solution of complex engineering problems.	
R3 Efficiency of the solution	Solution does not meet requirements.	A difficult and inefficient solution.	A logical solution that is easy to follow but it is not the most efficient.	Solution is adequately efficient.	Solution is efficient, easy to understand, and maintain.	
R4 Technical Writing	The report is submitted but lacks solutions to major requirements.	The report submitted but not according to the requirements.	The requirements of report writing are not properly addressed.	Reports meets all prescribed requirements.	Reports meets all requirements, and it is prepared in original and corrective way to engage readers.	