**Project Overview**

**LibManage** is a command-line based Library Management System designed to streamline and digitize the core operations of a library. It allows librarians or admins to efficiently manage books, members, book issues and returns, with an integrated fine calculation system for overdue books. The project focuses on building a robust backend using Python and MySQL to provide a real-world learning experience in database handling and backend logic implementation.

The system is designed to be simple to use via the terminal, without any frontend interface, making it ideal for practicing Python, database design, CRUD operations, and real-world problem-solving without the overhead of UI development.

## Core Features

1. **Login and Authentication**
   * Secure login system for admins to access the platform.
2. **Book Management**
   * Add, update, view, search (by title, author, or genre), and delete books from the system.
3. **Member Management**
   * Register new members, view member list, avoid duplicate registrations, and delete member records.
4. **Book Issue System**
   * Issue books to registered members, assign due date (15 days from issue date), and prevent re-issuing an already issued book.
5. **Book Return System**
   * Record book return, auto-calculate fines (Rs. 5/day after due date), and update book availability.
6. **Fine Calculation**
   * Fine is automatically calculated based on delay in return and shown during return.
7. **(Planned) Reports and Analytics**
   * Track most borrowed books, overdue returns, and active members.
8. **(Planned) Logging and CSV Export**
   * Maintain transaction logs and export data for analysis/reporting.

## Technologies Used

* **Programming Language:** Python
* **Database:** MySQL
* **Database Connector:** mysql-connector-python (for database operations)
* **Development Environment:** VS Code / PyCharm / Terminal
* **Version Control:** Git (for tracking progress, optional)
* **Optional:** CSV module for export feature

## Database Schema

1. Admin

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| username | VARCHAR (PK) | Unique admin username |
| email | VARCHAR (UNIQUE) | Admin email ID |
| password | VARCHAR | Hashed password |

1. Books

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| book\_id | INT (PK) | Unique identifier for each book |
| title | VARCHAR | Book title |
| author | VARCHAR | Book author |
| genre | VARCHAR | Book genre/category |
| quantity | INT | Number of copies available |

1. Members

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| member\_id | INT (PK) | Unique ID for each member |
| name | VARCHAR | Member's full name |
| email | VARCHAR (UNIQUE) | Member's email address |
| mobile\_no | VARCHAR | Member's contact number |
| date\_registered | DATE | Date of registration |

1. BookIssue

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| book\_id | INT (FK) | Refers to Books.book\_id |
| member\_id | INT (FK) | Refers to Members.member\_id |
| issue\_date | DATE | Date when book was issued |
| due\_date | DATE | Calculated as issue date + 15 days |
| return\_date | DATE (nullable) | Actual return date (if returned) |
| fine | INT | Fine amount (if applicable) |
| status | VARCHAR | 'Issued' or 'Returned' |

### **DAY-1**

### 🔧 **Work Done:**

* Implemented secure **Admin Login** functionality.
* Integrated **MySQL authentication** using username-password matching.
* Set a **limit of 3 login attempts** to enhance security.
* Used the **colorama** Python library to add **colored terminal output** for better user experience.
* Refined user feedback messages with appropriate color codes for:
  + Input prompts
  + Successful login
  + Wrong password
  + Invalid admin
  + Login attempt warnings

### ✅ **Features Implemented Today:**

* Admin authentication with attempt limits
* Colored CLI feedback using colorama

## Module Documentation: book.py

### 📌 Function: addbook()

This module is responsible for adding books to the Library Management System's MySQL database. It checks for duplicates based on **book title and author** and updates the quantity if the book already exists, preventing redundant entries.

### 🧩 Functionality Overview

#### 🔧 Inputs:

* **Book Title**
* **Author(s)** (comma-separated if more than one)
* **Genre**
* **Publisher**
* **Quantity**
* **Year of Publication**

#### 🛠️ Actions Performed:

1. Connects to the MySQL database via getconnection() from main.py.
2. Checks if a book with the **same title and author** already exists (case and space insensitive).
3. If exists: **updates quantity**.
4. If not: **inserts as new entry**.

### ✅ Output:

* Book Added Successfully! – if inserted as a new entry.
* Book quantity updated successfully! – if an existing entry was updated.

## 🔍 Key Features:

* **Duplicate Detection**: Uses normalized (lower() + strip()) title and author for precise matching.
* **Safe Execution**: Uses parameterized queries to prevent SQL injection.
* **Error Handling**: Catches and prints MySQL exceptions.

## 🚧 Challenges Faced & Solutions

| **Challenge** | **Description** | **Resolution** |
| --- | --- | --- |
| ❌ Multiple primary keys error | Occurred when trying to re-define AUTO\_INCREMENT and PRIMARY KEY on a column that already had it. | Fixed by removing extra PK declarations or checking schema first. |
| ❌ Duplicate entries being inserted | Even if the same book was entered again, a new record was added. | Used LOWER(TRIM(...)) in both SQL and input values to normalize data for accurate matching. |
| ❌ Incorrect detection of existing books | SQL check returned empty result due to case/space mismatch. | Debug print added to log fetched results. Then, query was fixed with normalization logic. |
| ❌ Update not happening | The old logic used += quantity, but user needed to **replace** quantity with the new one. | Changed logic to UPDATE books SET quantity = %s rather than incrementing. |
| ❌ Debugging was tedious | Multiple manual test runs needed to ensure match logic works | Added DEBUG: print statement to see exactly what the SELECT query returns. |

## 🖥️ Example Session:

shell

CopyEdit

You are adding a book

Enter the book name or title: Introduction to Algorithms

Enter the book author (if more than one authors, separate by commas): Thomas H.Cormen

Enter the genre of book: Computer Science

Enter the publisher: MIT Press

Enter the quantity: 10

Enter the year of publication of book: 2009

Book quantity updated successfully!

## 📂 Table Structure Assumed:

sql

CopyEdit

CREATE TABLE books (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255),

author VARCHAR(255),

genre VARCHAR(100),

quantity INT,

publisher VARCHAR(255),

year\_of\_publication YEAR

);

## 📦 Dependencies:

* colorama: For colored console output.
* mysql.connector: For database connectivity.
* main.getconnection(): Custom function to get a database connection.