**Project Name: Library Management System with Database Using Python**

**Objective**

The objective of this project is to create a comprehensive library management system that facilitates the management of students and books within a library. The system allows for:

* Creation and management of student and book records.
* Issuance and return of books.
* Displaying and modifying existing records.
* Ensuring that the database reflects the current status of books and student records.

**Summary**

The project uses Python and MySQL to implement a library management system with the following features:

* **Database Schema**: The system interacts with a MySQL database consisting of three main tables: students, books, and issued\_books.
  + students: Stores student records including roll number, name, and branch.
  + books: Stores book records including book ID, title, author, and availability status.
  + issued\_books: Tracks the issuance of books to students with issue and return dates.

**Key Functions**

1. **Initialize Database**:
   * Creates tables if they don't exist and sets up the database schema.
2. **Student Management**:
   * **Create Student**: Adds a new student record.
   * **Display All Students**: Lists all students and their details.
   * **Display Specific Student**: Shows details for a specific student based on roll number.
   * **Modify Student**: Updates student details.
   * **Delete Student**: Removes a student record.
3. **Book Management**:
   * **Create Book**: Adds a new book record.
   * **Display All Books**: Lists all books and their availability status.
   * **Display Specific Book**: Shows details for a specific book based on book ID.
   * **Modify Book**: Updates book details.
   * **Delete Book**: Removes a book record.
4. **Book Issuance and Return**:
   * **Issue Book**: Records the issuance of a book to a student and updates book availability.
   * **Deposit Book**: Records the return of a book by a student and updates book availability.
5. **Menu System**:
   * Provides a user interface for interacting with the system, including administration options and main operations.

**Results**

* **Database Schema**: The database is successfully initialized with tables for students, books, and issued books. The schema ensures referential integrity with foreign key constraints.
* **Student and Book Management**: The system effectively manages student and book records with functionality for creating, modifying, displaying, and deleting records.
* **Book Issuance and Return**: The system tracks the issuance and return of books, updating their availability status accordingly.
* **User Interface**: The menu-driven interface allows users to perform various operations easily and navigate between different functionalities.

**Conclusion**

The Library Management System is a robust solution for managing library operations, including student and book records, and tracking book issuance and returns. The use of Python and MySQL ensures a reliable and efficient system with a user-friendly interface. The system provides comprehensive features to manage library resources effectively, making it a valuable tool for educational institutions or libraries.

**Code :**

import mysql.connector

from datetime import date

# Define the database connection parameters

DB\_CONFIG = {

'user': 'root',

'password': 'Aditya@8624',

'host': 'localhost',

'database': 'library'

}

# Function to initialize the database

def initialize\_db():

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS students (

roll\_no VARCHAR(255) PRIMARY KEY,

name VARCHAR(255),

branch VARCHAR(255)

)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS books (

book\_id VARCHAR(255) PRIMARY KEY,

title VARCHAR(255),

author VARCHAR(255),

available INT

)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS issued\_books (

student\_roll\_no VARCHAR(255),

book\_id VARCHAR(255),

issue\_date DATE,

return\_date DATE,

FOREIGN KEY(student\_roll\_no) REFERENCES students(roll\_no),

FOREIGN KEY(book\_id) REFERENCES books(book\_id)

)''')

conn.commit()

conn.close()

# Function to create student record

def create\_student():

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

print("\nEnter student details:")

roll\_no = input("Roll No: ")

name = input("Name: ")

branch = input("Branch: ")

cursor.execute("INSERT INTO students (roll\_no, name, branch) VALUES (%s, %s, %s)", (roll\_no, name, branch))

conn.commit()

conn.close()

print(f"\nStudent Record for Roll No {roll\_no} created successfully.")

# Function to display all students

def display\_all\_students():

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students")

students = cursor.fetchall()

print("\nAll Students:")

for student in students:

print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")

print(f"Total Students: {len(students)}")

conn.close()

# Function to display specific student record

def display\_specific\_student():

roll\_no = input("\nEnter Roll No of the student to display: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students WHERE roll\_no = %s", (roll\_no,))

student = cursor.fetchone()

if student:

print("\nStudent Details:")

print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")

else:

print(f"\nStudent with Roll No {roll\_no} not found.")

conn.close()

# Function to modify student record

def modify\_student():

roll\_no = input("\nEnter Roll No of the student to modify: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students WHERE roll\_no = %s", (roll\_no,))

student = cursor.fetchone()

if student:

print("\nCurrent Student Details:")

print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")

print("\nEnter new details:")

name = input("Name (leave blank to keep current): ").strip()

branch = input("Branch (leave blank to keep current): ").strip()

if name:

cursor.execute("UPDATE students SET name = %s WHERE roll\_no = %s", (name, roll\_no))

if branch:

cursor.execute("UPDATE students SET branch = %s WHERE roll\_no = %s", (branch, roll\_no))

conn.commit()

print(f"\nStudent Record for Roll No {roll\_no} modified successfully.")

else:

print(f"\nStudent with Roll No {roll\_no} not found.")

conn.close()

# Function to delete student record

def delete\_student():

roll\_no = input("\nEnter Roll No of the student to delete: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("DELETE FROM students WHERE roll\_no = %s", (roll\_no,))

conn.commit()

if cursor.rowcount > 0:

print(f"\nStudent Record for Roll No {roll\_no} deleted successfully.")

else:

print(f"\nStudent with Roll No {roll\_no} not found.")

conn.close()

# Function to create book

def create\_book():

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

print("\nEnter book details:")

book\_id = input("Book ID: ")

title = input("Title: ")

author = input("Author: ")

cursor.execute("INSERT INTO books (book\_id, title, author, available) VALUES (%s, %s, %s, 1)", (book\_id, title, author))

conn.commit()

conn.close()

print(f"\nBook Record for Book ID {book\_id} created successfully.")

# Function to display all books

def display\_all\_books():

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM books")

books = cursor.fetchall()

print("\nAll Books:")

for book in books:

status = "Available" if book[3] == 1 else "Issued"

print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}, Status: {status}")

print(f"Total Books: {len(books)}")

conn.close()

# Function to display specific book record

def display\_specific\_book():

book\_id = input("\nEnter Book ID of the book to display: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM books WHERE book\_id = %s", (book\_id,))

book = cursor.fetchone()

if book:

status = "Available" if book[3] == 1 else "Issued"

print("\nBook Details:")

print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}, Status: {status}")

else:

print(f"\nBook with Book ID {book\_id} not found.")

conn.close()

# Function to modify book record

def modify\_book():

book\_id = input("\nEnter Book ID of the book to modify: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM books WHERE book\_id = %s", (book\_id,))

book = cursor.fetchone()

if book:

print("\nCurrent Book Details:")

print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}")

print("\nEnter new details:")

title = input("Title (leave blank to keep current): ").strip()

author = input("Author (leave blank to keep current): ").strip()

if title:

cursor.execute("UPDATE books SET title = %s WHERE book\_id = %s", (title, book\_id))

if author:

cursor.execute("UPDATE books SET author = %s WHERE book\_id = %s", (author, book\_id))

conn.commit()

print(f"\nBook Record for Book ID {book\_id} modified successfully.")

else:

print(f"\nBook with Book ID {book\_id} not found.")

conn.close()

# Function to delete book record

def delete\_book():

book\_id = input("\nEnter Book ID of the book to delete: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("DELETE FROM books WHERE book\_id = %s", (book\_id,))

conn.commit()

if cursor.rowcount > 0:

print(f"\nBook Record for Book ID {book\_id} deleted successfully.")

else:

print(f"\nBook with Book ID {book\_id} not found.")

conn.close()

# Function to issue a book

def issue\_book():

student\_roll\_no = input("\nEnter Student Roll No: ")

book\_id = input("Enter Book ID: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM students WHERE roll\_no = %s", (student\_roll\_no,))

student\_exists = cursor.fetchone() is not None

cursor.execute("SELECT \* FROM books WHERE book\_id = %s", (book\_id,))

book = cursor.fetchone()

book\_exists = book is not None

book\_available = book\_exists and book[3] == 1

if not student\_exists:

print(f"\nStudent with Roll No {student\_roll\_no} not found.")

conn.close()

return

if not book\_exists:

print(f"\nBook with Book ID {book\_id} not found.")

conn.close()

return

if not book\_available:

print(f"\nBook with Book ID {book\_id} is already issued.")

conn.close()

return

cursor.execute("INSERT INTO issued\_books (student\_roll\_no, book\_id, issue\_date, return\_date) VALUES (%s, %s, %s, %s)",

(student\_roll\_no, book\_id, date.today(), None))

cursor.execute("UPDATE books SET available = 0 WHERE book\_id = %s", (book\_id,))

conn.commit()

conn.close()

print(f"\nBook ID {book\_id} issued to Student Roll No {student\_roll\_no} successfully.")

# Function to deposit a book

def deposit\_book():

student\_roll\_no = input("\nEnter Student Roll No: ")

book\_id = input("Enter Book ID: ")

conn = mysql.connector.connect(\*\*DB\_CONFIG)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM issued\_books WHERE student\_roll\_no = %s AND book\_id = %s AND return\_date IS NULL",

(student\_roll\_no, book\_id))

issued\_book = cursor.fetchone()

if not issued\_book:

print(f"\nNo record found for Book ID {book\_id} issued to Student Roll No {student\_roll\_no}.")

conn.close()

return

cursor.execute("UPDATE issued\_books SET return\_date = %s WHERE student\_roll\_no = %s AND book\_id = %s AND return\_date IS NULL",

(date.today(), student\_roll\_no, book\_id))

cursor.execute("UPDATE books SET available = 1 WHERE book\_id = %s", (book\_id,))

conn.commit()

conn.close()

print(f"\nBook ID {book\_id} returned by Student Roll No {student\_roll\_no} successfully.")

# Main function to display menu and process user input

def main():

initialize\_db()

while True:

print("\n\*\*\*\*\* LIBRARY MANAGEMENT SYSTEM \*\*\*\*\*")

print("1. BOOK ISSUE")

print("2. BOOK DEPOSIT")

print("3. ADMINISTRATION MENU")

print("4. EXIT")

choice = input("\nEnter your choice: ")

if choice == '1':

issue\_book()

elif choice == '2':

deposit\_book()

elif choice == '3':

print("\n\*\*\*\*\* ADMINISTRATION MENU \*\*\*\*\*")

print("a. CREATE STUDENT RECORD")

print("b. DISPLAY ALL STUDENTS RECORD")

print("c. DISPLAY SPECIFIC STUDENT RECORD")

print("d. MODIFY STUDENT RECORD")

print("e. DELETE STUDENT RECORD")

print("f. CREATE BOOK")

print("g. DISPLAY ALL BOOKS")

print("h. DISPLAY SPECIFIC BOOK")

print("i. MODIFY BOOK RECORD")

print("j. DELETE BOOK RECORD")

print("k. BACK TO MAIN MENU")

admin\_choice = input("\nEnter your choice: ")

if admin\_choice == 'a':

create\_student()

elif admin\_choice == 'b':

display\_all\_students()

elif admin\_choice == 'c':

display\_specific\_student()

elif admin\_choice == 'd':

modify\_student()

elif admin\_choice == 'e':

delete\_student()

elif admin\_choice == 'f':

create\_book()

elif admin\_choice == 'g':

display\_all\_books()

elif admin\_choice == 'h':

display\_specific\_book()

elif admin\_choice == 'i':

modify\_book()

elif admin\_choice == 'j':

delete\_book()

elif admin\_choice == 'k':

continue

else:

print("\nInvalid choice! Please enter a valid option (a-k).")

elif choice == '4':

print("\nThank you for using the Library Management System. Goodbye!")

break

else:

print("\nInvalid choice! Please enter a valid option (1-4).")

# Entry point of the program

if \_\_name\_\_ == "\_\_main\_\_":

main()