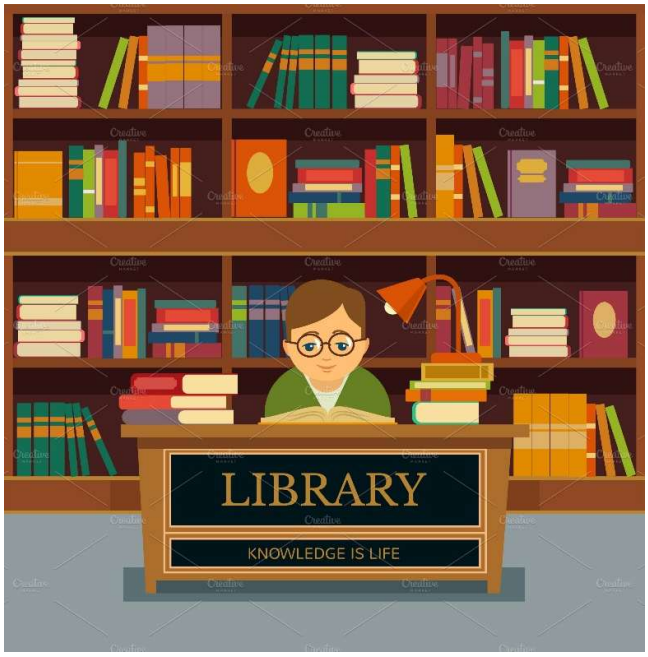
PM SHRI KENDRIYA VIDYALAYA NFC VIGYAN VIHAR

COMPUTER SCIENCE PROJECT

LIBRARY MANAGEMENT



DEVELOPED BY

SHAGUN DUBEY

XII E

28

SUBMITTED TO

POOJA KHARE

CERTIFICATE

This is to certify that SHAGUN DUBEY of class XII E of PM SHRI KENDRIYA VIDYALAYA NFC VIGYAN VIHAR has completed her project on "**SCHOOL MANAGEMENT SYSTEM**" under my supervision. She has shown great interest and sincerity in completing this project.

_____ _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I want to express my deepest gratitude to all those who have helped me complete this school project successfully.

I am extremely thankful to my project guide, Mrs. POOJA KHARE, my Computer Science teacher, for invaluable guidance, encouragement, and support throughout this project. She provided me with direction, reviewed my progress and results, and helped me clarify my doubts. I sincerely appreciate the time and effort she put into supervising my project.

I am grateful to our school Principal, Dr Gaya Ravidas for providing us the resources and facilities to enable our project work. She took a keen interest in my project and motivated me to do my best.

My special thanks to my family, they encouraged me at every step and helped me manage my time effectively. Their faith in my abilities inspired me to take on this project.

I could not have completed this project successfully without the support of all these people. I will always be indebted for everything they have done for me.

INDEX

INTRODUCTION TO THE LIBRARY MANAGEMENT SYSTEM

- A Library Management System is a software application designed to help manage library operations effectively.
- It integrates various modules for tasks like:
    1. Book Management: Adding, updating, and deleting book records.
    2. User Management: Registering and maintaining records of library users.
    3. Transaction Management: Issuing, returning, and tracking overdue books.
    4. Search Functionality: Allowing users to search for books by title, author, or year.
    5. Reports and Insights: Generating data about book popularity, availability, and user activity.
- By transitioning from manual operations to a digital system, libraries can meet the growing demand for efficient and accurate management.

AIM OF THE LIBRARY MANAGEMENT SYSTEM

The primary aim of a Library Management System is to simplify and automate library operations, making the library more efficient, organized, and user-friendly. Specifically, it aims to:

- Provide librarians and administrators with tools for efficient management.
- Improve user satisfaction by enabling fast and easy access to resources.
- Minimize errors in book allocation, returns, and inventory management.
- Track overdue items and facilitate reminders.
- Enable smooth registration and management of library members.

IMPLEMENTATION OF THE LIBRARY MANAGEMENT SYSTEM

The implementation of the Library Management System involves several steps:

1. **Database Creation**:

A MySQL database is created with tables for storing information about books, users, and transactions. The key tables include:

- Books: Contains details like Book ID, title, author, publication year, and availability.
- Users: Maintains records of library members, including User ID, name, and contact information.
- Transactions: Tracks book issues and returns with timestamps and user details.

2. **Development of Features**:

- Admin Features:
  - Add, update, or delete books.
  - View all books and users.
  - Register new users.
- Librarian Features:
  - Issue books to users.
  - Return books and update availability.
- User Features (optional for future expansion):
  - Search for books.
  - View borrowed books and due dates.

3. **Programming**:

- Frontend: Uses Python for the command-line interface to interact with the system.
- Backend: Python's mysql.connector module connects the program to the MySQL database for CRUD (Create, Read, Update, Delete) operations.

4. **System Security**:
   Password-protected interfaces (e.g., Admin password) ensure restricted access to critical operations.

5. **Testing and Deployment**:
   The system is tested with various scenarios like issuing a book that's not available or attempting to return a book already returned to ensure it handles edge cases effectively.

6. **Future Enhancements**:
   - Integration of a graphical user interface (GUI) for better usability.
   - Online catalog access for remote users.
   - SMS or email notifications for overdue books.

NEED FOR A LIBRARY MANAGEMENT SYSTEM

1. **Efficient Resource Management**
   Manual library systems are prone to errors and inefficiencies. A Library Management System (LMS) automates operations like issuing, returning, and inventory management to reduce human effort and error.

2. **Time-Saving**
   Searching for books, tracking due dates, and managing users manually can be time-consuming. An LMS provides instant access to information, saving time for both librarians and users.

3. **Transparency and Accountability**
   The system maintains a detailed record of all transactions—like books issued, returned, and overdue—ensuring transparency and accountability.

4. **Improved User Experience**
   Users can quickly find the availability of books, check due dates, or request a book, enhancing their experience.

5. **Better Inventory Management**
   An LMS helps keep track of available books, damaged copies, and the most popular books. This information is useful for stock updates and purchasing decisions.

#CODE#

```
import mysql.connector as pymysql
from datetime import datetime

password = None
db = None
C = None

def base_check():
    check = 0
    db = pymysql.connect(host="localhost", user="root", password=passwrd)
    cursor = db.cursor()
    cursor.execute('SHOW DATABASES')
    result = cursor.fetchall()
    for r in result:
        for i in r:
            if i == 'library':
                cursor.execute('USE library')
                check = 1
    if check != 1:
        create_database()

def table_check():
    db = pymysql.connect(host="localhost", user="root", password=passwrd)
    cursor = db.cursor()
```

```
    cursor.execute('SHOW DATABASES')
    result = cursor.fetchall()
    for r in result:
        for i in r:
            if i == 'library':
                cursor.execute('USE library')
                cursor.execute('SHOW TABLES')
                result = cursor.fetchall()
                if len(result) <= 2:
                    create_tables()
                else:
                    print('      Booting systems...')

def create_database():
    try:
        db = pymysql.connect(host="localhost", user="root", password=passwrd)
        cursor = db.cursor()
        cursor.execute("CREATE DATABASE IF NOT EXISTS library")
        db.commit()
        db.close()
        print("Database 'library' created successfully.")
    except pymysql.Error as e:
        print(f"Error creating database: {str(e)}")

def create_tables():
    try:
```

```
    db = pymysql.connect(host="localhost", user="root", password=passwrd,
database="library")
    cursor = db.cursor()

    cursor.execute("""
      CREATE TABLE IF NOT EXISTS books (
        BOOK_ID INT PRIMARY KEY,
        TITLE VARCHAR(255),
        AUTHOR VARCHAR(255),
        YEAR INT,
        AVAILABLE INT
      )
    """)

    cursor.execute("""
      CREATE TABLE IF NOT EXISTS users (
        USER_ID INT PRIMARY KEY,
        NAME VARCHAR(255),
        PHONE_NO VARCHAR(15)
      )
    """)

    cursor.execute("""
      CREATE TABLE IF NOT EXISTS transactions (
        TRANSACTION_ID INT AUTO_INCREMENT PRIMARY KEY,
        USER_ID INT,
        BOOK_ID INT,
```

```
            ISSUE_DATE DATE,
            RETURN_DATE DATE,
            FOREIGN KEY (USER_ID) REFERENCES users(USER_ID),
            FOREIGN KEY (BOOK_ID) REFERENCES books(BOOK_ID) ON DELETE CASCADE
        )
    """)

    db.commit()
    db.close()
    print("Tables 'books', 'users', and 'transactions' created successfully.")
  except pymysql.Error as e:
    print(f"Error creating tables: {str(e)}")

def QR():
  result = C.fetchall()
  for r in result:
    print(r)

def add_book():
  book_id = int(input("Enter Book ID: "))
  title = input("Enter Book Title: ")
  author = input("Enter Author: ")
  year = int(input("Enter Year of Publication: "))
  available = int(input("Enter Number of Available Copies: "))
  data = (book_id, title, author, year, available)
  sql = "INSERT INTO books (BOOK_ID, TITLE, AUTHOR, YEAR, AVAILABLE) VALUES (%s, %s,
%s, %s, %s)"
```

```
    try:
        C.execute(sql, data)
        db.commit()
        print('Book added successfully...')
    except pymysql.Error as e:
        print(f"Error adding book: {str(e)}")

def view_books():
    C.execute("SELECT * FROM books")
    QR()

def update_book():
    book_id = int(input("Enter Book ID to update: "))
    field = input("Enter field to update [TITLE, AUTHOR, YEAR, AVAILABLE]: ")
    new_value = input(f"Enter new value for {field}: ")
    if field in ['YEAR', 'AVAILABLE']:
        new_value = int(new_value)
    sql = f"UPDATE books SET {field} = %s WHERE BOOK_ID = %s"
    try:
        C.execute(sql, (new_value, book_id))
        db.commit()
        print('Book updated successfully...')
    except pymysql.Error as e:
        print(f"Error updating book: {str(e)}")

def delete_book():
    book_id = int(input("Enter Book ID to delete: "))
```

```
    try:
        sql_delete_transactions = "DELETE FROM transactions WHERE BOOK_ID = %s"
        C.execute(sql_delete_transactions, (book_id,))

        sql_delete_book = "DELETE FROM books WHERE BOOK_ID = %s"
        C.execute(sql_delete_book, (book_id,))

        db.commit()
        print('Book and related transactions deleted successfully...')
    except pymysql.Error as e:
        print(f"Error deleting book: {str(e)}")

def register_user():
    user_id = int(input("Enter User ID: "))
    name = input("Enter User Name: ")
    phone_no = input("Enter User Phone Number: ")
    data = (user_id, name, phone_no)
    sql = "INSERT INTO users (USER_ID, NAME, PHONE_NO) VALUES (%s, %s, %s)"
    try:
        C.execute(sql, data)
        db.commit()
        print('User registered successfully...')
    except pymysql.Error as e:
        print(f"Error registering user: {str(e)}")

def view_users():
    C.execute("SELECT * FROM users")
```

```
    QR()


def issue_book():
    user_id = int(input("Enter User ID: "))
    book_id = int(input("Enter Book ID: "))
    issue_date = datetime.now().date()
    sql_check = "SELECT AVAILABLE FROM books WHERE BOOK_ID = %s"
    C.execute(sql_check, (book_id,))
    result = C.fetchone()
    if result and result[0] > 0:
        sql_issue = "INSERT INTO transactions (USER_ID, BOOK_ID, ISSUE_DATE) VALUES (%s,
%s, %s)"
        try:
            C.execute(sql_issue, (user_id, book_id, issue_date))
            sql_update = "UPDATE books SET AVAILABLE = AVAILABLE - 1 WHERE BOOK_ID = %s"
            C.execute(sql_update, (book_id,))
            db.commit()
            print('Book issued successfully...')
        except pymysql.Error as e:
            print(f"Error issuing book: {str(e)}")
    else:
        print("Book not available.")


def return_book():
    user_id = int(input("Enter User ID: "))
    book_id = int(input("Enter Book ID: "))
    return_date = datetime.now().date()
```

```
    sql_update = "UPDATE transactions SET RETURN_DATE = %s WHERE USER_ID = %s AND
BOOK_ID = %s AND RETURN_DATE IS NULL"
    try:
        C.execute(sql_update, (return_date, user_id, book_id))
        sql_check = "SELECT ISSUE_DATE FROM transactions WHERE USER_ID = %s AND
BOOK_ID = %s AND RETURN_DATE = %s"
        C.execute(sql_check, (user_id, book_id, return_date))
        result = C.fetchone()
        if result:
            sql_book = "UPDATE books SET AVAILABLE = AVAILABLE + 1 WHERE BOOK_ID = %s"
            C.execute(sql_book, (book_id,))
            db.commit()
            print('Book returned successfully...')
        else:
            print("Transaction not found or already returned.")
    except pymysql.Error as e:
        print(f"Error returning book: {str(e)}")


def view_user_transactions():
    user_id = int(input("Enter User ID: "))
    try:
        sql_issued = """
            SELECT b.TITLE, t.ISSUE_DATE
            FROM transactions t
            JOIN books b ON t.BOOK_ID = b.BOOK_ID
            WHERE t.USER_ID = %s AND t.RETURN_DATE IS NULL
        """
```

```
    C.execute(sql_issued, (user_id,))
    issued_books = C.fetchall()

    sql_returned = """
        SELECT b.TITLE, t.ISSUE_DATE, t.RETURN_DATE
        FROM transactions t
        JOIN books b ON t.BOOK_ID = b.BOOK_ID
        WHERE t.USER_ID = %s AND t.RETURN_DATE IS NOT NULL
    """
    C.execute(sql_returned, (user_id,))
    returned_books = C.fetchall()

    print("\nBooks Currently Issued:")
    if issued_books:
        for book in issued_books:
            print(f"Title: {book[0]}, Issue Date: {book[1]}")
    else:
        print("No books currently issued.")

    print("\nBooks Returned:")
    if returned_books:
        for book in returned_books:
            print(f"Title: {book[0]}, Issue Date: {book[1]}, Return Date: {book[2]}")
    else:
        print("No books returned.")
except pymysql.Error as e:
    print(f"Error fetching user transactions: {str(e)}")
```

```
def main():
    global passwrd
    passwrd = input("Enter password for MySQL: ")

    base_check()
    table_check()

    global db, C
    db = pymysql.connect(host="localhost", user="root", password=passwrd,
database="library")
    C = db.cursor()

    while True:
        log = input("\nLogin as Admin (A) or Librarian (L). Press Q to quit: ").upper()
        if log == "A":
            p = input("Enter Admin Password: ")
            if p == 'admin123':
                print("Admin Login Successful!")
                while True:
                    menu = input('''\nAdd Book: AB, View Books: VB, Update Book: UB, Delete Book:
DB, Exit: E: ''').upper()
                    if menu == 'AB':
                        add_book()
                    elif menu == 'VB':
                        view_books()
                    elif menu == 'UB':
```

```
                update_book()
            elif menu == 'DB':
                delete_book()
            elif menu == 'E':
                break
        else:
            print("Incorrect Admin Password.")
    elif log == "L":
        print("Librarian Login Successful!")
        while True:
            menu = input('''\nRegister User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: ''').upper()
            if menu == 'RU':
                register_user()
            elif menu == 'VU':
                view_users()
            elif menu == 'IB':
                issue_book()
            elif menu == 'RB':
                return_book()
            elif menu == 'VT':
                view_user_transactions()
            elif menu == 'E':
                break
    elif log == "Q":
        db.commit()
        db.close()
```

```
        print("Goodbye!")
        break
    else:
        print("Invalid input. Please try again.")

if __name__ == '__main__':
    main()
```

MAIN LOGIN

```
Enter password for MySQL: k@ng@r00
Database 'library' created successfully.
Tables 'books', 'users', and 'transactions' created successfully.
```

ADMIN LOGIN

```
Login as Admin (A) or Librarian (L). Press Q to quit: a
Enter Admin Password: admin123
Admin Login Successful!
```

ADMIN FUNCTIONS

➢ ADD BOOKS

```
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: AB
Enter Book ID: 1
Enter Book Title: Pride & Prejudice
Enter Author: Jane Austen
Enter Year of Publication: 1813
Enter Number of Available Copies: 8
Book added successfully...
```

➢ VIEW BOOK

```
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: vb
(1, 'Pride & Prejudice', 'Jane Austen', 1813, 8)
(2, 'A Tale of Two Cities', 'Charles Dickens', 1859, 4)
(3, 'Lord of the Flies', 'William Golding', 1954, 9)
(4, 'The Alchemist', 'Paulo Coelho', 1988, 18)
(5, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 9)
```

➢ UPDATE BOOK

```
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: ub
Enter Book ID to update: 2
Enter field to update [TITLE, AUTHOR, YEAR, AVAILABLE]: available
Enter new value for available: 8
Book updated successfully...
```

```
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: vb
(1, 'Pride & Prejudice', 'Jane Austen', 1813, 8)
(2, 'A Tale of Two Cities', 'Charles Dickens', 1859, 8)
(3, 'Lord of the Flies', 'William Golding', 1954, 9)
(4, 'The Alchemist', 'Paulo Coelho', 1988, 18)
(5, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 9)
```

➢ DELETE BOOK

```
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: db
Enter Book ID to delete: 5
Book and related transactions deleted successfully...
Add Book: AB, View Books: VB, Update Book: UB, Delete Book: DB, Exit: E: vb
(1, 'Pride & Prejudice', 'Jane Austen', 1813, 8)
(2, 'A Tale of Two Cities', 'Charles Dickens', 1859, 8)
(3, 'Lord of the Flies', 'William Golding', 1954, 9)
(4, 'The Alchemist', 'Paulo Coelho', 1988, 18)
```

LIBRARIAN LOGIN

```
Login as Admin (A) or Librarian (L). Press Q to quit: l
Librarian Login Successful!
```

LIBRARIAN FUNCTIONS

➢REGISTER USER

```
Register User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: ru
Enter User ID: 1001
Enter User Name: Ajay
Enter User Phone Number: 9810556677
User registered successfully...
```

➢VIEW USER

```
Register User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: vu
(100, 'Gyatri', '9876543210')
(1001, 'Ajay', '9810556677')
(1002, 'mohini', '9810765656')
```

➢ISSUE BOOK

```
Register User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: ib
Enter User ID: 100
Enter Book ID: 4
Book issued successfully...
```

➢RETURN BOOK

```
Register User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: rb
Enter User ID: 100
Enter Book ID: 4
Book returned successfully...
```

➢ VIEW TRANSACTIONS

```
Register User: RU, View Users: VU, Issue Book: IB, Return Book: RB, View User Transactions: VT, Exit: E: vt
Enter User ID: 100

Books Currently Issued:
No books currently issued.

Books Returned:
Title: The Alchemist, Issue Date: 2024-11-26, Return Date: 2024-11-26
```

TABLES FORMED

```
mysql> select * from books;
+---------+----------------------+------------------+------+-----------+
| BOOK_ID | TITLE                | AUTHOR           | YEAR | AVAILABLE |
+---------+----------------------+------------------+------+-----------+
|       1 | Pride & Prejudice    | Jane Austen      | 1813 |         8 |
|       2 | A Tale of Two Cities | Charles Dickens  | 1859 |         8 |
|       3 | Lord of the Flies    | William Golding  | 1954 |         9 |
|       4 | The Alchemist        | Paulo Coelho     | 1988 |        18 |
+---------+----------------------+------------------+------+-----------+
4 rows in set (0.03 sec)

mysql> select * from transactions;
+----------------+---------+---------+------------+-------------+
| TRANSACTION_ID | USER_ID | BOOK_ID | ISSUE_DATE | RETURN_DATE |
+----------------+---------+---------+------------+-------------+
|              1 |     100 |       4 | 2024-11-26 | 2024-11-26  |
+----------------+---------+---------+------------+-------------+
1 row in set (0.01 sec)

mysql> select * from users;
+---------+--------+------------+
| USER_ID | NAME   | PHONE_NO   |
+---------+--------+------------+
|     100 | Gyatri | 9876543210 |
|    1001 | Ajay   | 9810556677 |
|    1002 | mohini | 9810765656 |
+---------+--------+------------+
3 rows in set (0.01 sec)
```

DESCRIPTION OF TABLES

```
mysql> DESC TRANSACTIONS;
+----------------+------+------+-----+---------+----------------+
| Field          | Type | Null | Key | Default | Extra          |
+----------------+------+------+-----+---------+----------------+
| TRANSACTION_ID | int  | NO   | PRI | NULL    | auto_increment |
| USER_ID        | int  | YES  | MUL | NULL    |                |
| BOOK_ID        | int  | YES  | MUL | NULL    |                |
| ISSUE_DATE     | date | YES  |     | NULL    |                |
| RETURN_DATE    | date | YES  |     | NULL    |                |
+----------------+------+------+-----+---------+----------------+
5 rows in set (0.20 sec)

mysql> DESC BOOKS;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| BOOK_ID   | int          | NO   | PRI | NULL    |       |
| TITLE     | varchar(255) | YES  |     | NULL    |       |
| AUTHOR    | varchar(255) | YES  |     | NULL    |       |
| YEAR      | int          | YES  |     | NULL    |       |
| AVAILABLE | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.06 sec)

mysql> DESC USERS;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| USER_ID  | int          | NO   | PRI | NULL    |       |
| NAME     | varchar(255) | YES  |     | NULL    |       |
| PHONE_NO | varchar(15)  | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

REQUIREMENTS (HARDWARE & SOFTWARE)

HARDWARE

- PC
- MOBILE PHONE

SOFTWARE

- PYTHON (latest version)
- MYSQL
- PYTHON CONNECTOR

BIBLIOGRAPHY

- Textbook
- Google
- YouTube
- Class notes