DBMS MINI PROJECT: LIBRARY MANAGEMENT SYSTEM

Submitted by:

Anandikamathi R

Chris Glary.C

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

St. JOSEPH’S COLLEGE OF ENGINEERING

(An Autonomous Institution)

St. Joseph’s Group of Institutions

OMR, Chennai 600 119

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
1.	ABSTRACT	3
2.	INTRODUCTION	4
3.	PROGRAM	5 – 10
4.	RESULT	12 - 14
5.	CONCLUSION	15

ABSTRACT

The Library Management System (LMS) mini-project aims to design and implement a robust database system for efficiently managing the operations of a library. The system encompasses key features such as book management, member management, borrowing, returning, searching, and reporting functionalities .In this project, a relational database model is employed to represent various entities and their relationships within the library ecosystem. The entities include books, members, and borrowings, each with specific attributes capturing essential information. Through normalization techniques, the database schema is structured to minimize redundancy and ensure data integrity.The LMS enables library administrators to add, update, and remove books from the collection, facilitating seamless management of the library inventory. Concurrently, members can register, update their information, and borrow books, with the system automatically recording borrowing details and managing due dates. Fines for overdue books are calculated and enforced to encourage timely returns.

INTRODUCTION

The Library Management System (LMS) mini-project endeavors to create a comprehensive database system tailored to the needs of a modern library. With the advent of digital technologies, libraries are transitioning towards automated systems to streamline their operations and enhance user experience. The LMS serves as a digital solution, offering functionalities to efficiently manage the library's resources and facilitate seamless interaction between library staff and patrons.This project addresses the fundamental requirements of a library, including book management, member registration, borrowing, returning, searching, and reporting. By leveraging database management principles and techniques, the LMS aims to optimize data organization, accessibility, and reliability, thereby improving overall library management efficiency.The implementation of the LMS involves the design and development of a relational database model, where entities such as books, members, and borrowings are defined along with their respective attributes and relationships

Key Features

1. Book Management:

- Add new books with details like title, author, genre, publication year, ISBN, quantity, etc.
- Update book details.
- Remove books from the library database.

2. Member Management:

- Register new library members with their information like name, contact details, and membership ID.
- Update member details.
- Remove members from the system.

3.Borrowing and Returning:

- Allow members to borrow books.
- Record borrowing details such as the member who borrowed the book, the book borrowed, borrowing date, and due date.

- Handle book returns and update the database accordingly.
- Calculate fines for late returns, if applicable.

4. Search Functionality:

- Enable users to search for books by title, author, genre, ISBN, etc.
- Provide advanced search options like filtering by availability, publication year range, etc.

5. Reporting:

- Generate reports such as:
  - List of all books in the library.
  - List of books borrowed by a specific member.
  - List of overdue books.
  - Statistical analysis like most borrowed books, most active members, etc.

PROGRAM

```
import sqlite3
import datetime

# Connect to the database
conn = sqlite3.connect('library.db')
cursor = conn.cursor()

def create_tables():
    # Create tables if they don't exist
    cursor.execute('''CREATE TABLE IF NOT EXISTS books (
                    id INTEGER PRIMARY KEY,
                    title TEXT,
                    author TEXT,
                    publication_year INTEGER,
                    isbn TEXT,
                    quantity INTEGER
                )''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY,
                    name TEXT,
                    email TEXT
                )''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS transactions (
                    id INTEGER PRIMARY KEY,
                    user_id INTEGER,
                    book_id INTEGER,
                    transaction_date TEXT,
                    FOREIGN KEY (user_id) REFERENCES users(id),
                    FOREIGN KEY (book_id) REFERENCES books(id)
                 )''')
    conn.commit()

def add_book(title, author, publication_year, isbn, quantity):
    cursor.execute('''INSERT INTO books (title, author, 
publication_year, isbn, quantity) VALUES (?, ?, ?, ?, ?)''',
                 (title, author, publication_year, isbn, quantity))
    conn.commit()

def display_books():
    cursor.execute("SELECT * FROM books")
    books = cursor.fetchall()
    if books:
        print("ID\tTitle\tAuthor\tPublication 
Year\tISBN\tQuantity")
        for book in books:

print(f"{book[0]}\t{book[1]}\t{book[2]}\t{book[3]}\t{book[4]}\t{
book[5]}")
    else:
        print("No books available.")

def borrow_book(user_id, book_id):
    # Check if the book is available
    cursor.execute("SELECT quantity FROM books WHERE id=?", 
(book_id,))
    result = cursor.fetchone()
    if result and result[0] > 0:
        # Decrement the quantity of the book
        cursor.execute("UPDATE books SET quantity=quantity-1
WHERE id=?", (book_id,))
        # Record the transaction
        transaction_date = datetime.datetime.now().strftime("%Y-
%m-%d %H:%M:%S")
        cursor.execute("INSERT INTO transactions (user_id,
book_id, transaction_date) VALUES (?, ?, ?)",
                      (user_id, book_id, transaction_date))
        conn.commit()
        print("Book borrowed successfully.")
    else:
        print("Book not available.")
def return_book(user_id, book_id):
    # Check if the user has borrowed the book
    cursor.execute("SELECT * FROM transactions WHERE
user_id=? AND book_id=?", (user_id, book_id))
    result = cursor.fetchone()
    if result:
        # Increment the quantity of the book
        cursor.execute("UPDATE books SET quantity=quantity+1
WHERE id=?", (book_id,))
        # Delete the transaction record
        cursor.execute("DELETE FROM transactions WHERE id=?",
(result[0],))
        conn.commit()
        print("Book returned successfully.")
        else:
```

```
        print("You haven't borrowed this book.")

def list_borrowed_books(user_id):
    cursor.execute('''SELECT books.id, books.title, books.author
                      FROM books
                      JOIN transactions ON books.id =
transactions.book_id
                      WHERE transactions.user_id = ?''', (user_id,))
    borrowed_books = cursor.fetchall()
    if borrowed_books:
        print("ID\tTitle\tAuthor")
        for book in borrowed_books:
            print(f"{book[0]}\t{book[1]}\t{book[2]}")
    else:
        print("No books borrowed by this user.")
def close_connection():
    conn.close()

# Example usage
create_tables()

# Add some books
add_book("Book 1", "Author A", 2000, "1234567890", 5)
add_book("Book 2", "Author B", 2010, "0987654321", 3)
add_book("Book 3", "Author C", 2020, "1357924680", 2)

# Display all books
print("\nAll books in the library:")
display_books()
```

```
# Borrow a book
borrow_book(1, 1)

# List borrowed books for a user
print("\nBooks borrowed by User 1:")
list_borrowed_books(1)

# Return a book
return_book(1, 1)

# Display all books again
print("\nAll books in the library after returning a book:")
display_books()

close_connection()
```

RESULT

:

```
All books in the library:
ID     Title   Author  Publication Year       ISBN    Quantity
1      Book 1  Author A       2000    1234567890     5
2      Book 2  Author B       2010    0987654321     3
3      Book 3  Author C       2020    1357924680     2

Book borrowed successfully.

Books borrowed by User 1:
ID     Title   Author
1      Book 1  Author A

Book returned successfully.

All books in the library after returning a book:
ID     Title   Author  Publication Year       ISBN    Quantity
1      Book 1  Author A       2000    1234567890     6
2      Book 2  Author B       2010    0987654321     3
3      Book 3  Author C       2020    1357924680     2
```

CONCLUSION

In conclusion, the library management system project has successfully leveraged a database management system (DBMS) to streamline the organization's operations. Through careful design and implementation of tables and procedures, the system ensures efficient management of books, members, and borrowing records. Key features such as adding, updating, and removing books and members, along with borrowing and returning functionalities, have been effectively implemented. With a solid foundation in place, the system is poised for further enhancements and customization, promising continued efficiency and improved user experience for both administrators and members alike.