

**LIBRARY MANAGEMENT SYSTEM
A MINI-PROJECT REPORT**

Submitted by

PONSURABHI V

241901077

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project “**LIBRARY MANAGEMENT SYSTEM**” is the bonafide work of “**PONSURABHI V**” who carried out the project work under my supervision.

SIGNATURE

Dr. FOWZIA SIHANA

ASSISTANT PROFESSOR SS

Dept. of Computer Science and Engineering (Cyber Security)
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Library Management System is a database-driven application designed to streamline and automate library operations. This system enables users to view, issue, return, and search books efficiently while maintaining accurate records of members and transactions. Built using Python (CustomTkinter for GUI) and MySQL as the backend, this project demonstrates how databases can be integrated with real-world applications to ensure data consistency, accessibility, and reliability. The system includes functionalities such as viewing available books, issuing and returning books, searching for books, managing members, and tracking overdue items.

ACKNOWLEDGEMENT

I express my sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended by my Head Of The Department **Dr.BENEDICT JAYAPRAKASH NICHOLAS** for being ever supporting force during my project work .

I also extend my sincere and hearty thanks to my internal guide **Dr.FOWZIA SIHANA**, for her valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering (Cyber Security).

PONSURABHI V

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	3
1	INTRODUCTION	7
1.1	INTRODUCTION	7
1.2	SCOPE OF THE WORK	7
1.3	PROBLEM STATEMENT	7
1.4	AIM AND OBJECTIVES OF THE PROJECT	7
2	SYSTEM SPECIFICATIONS	8
2.1	HARDWARE SPECIFICATIONS	8
2.2	SOFTWARE SPECIFICATIONS	8
3	MODULE DESCRIPTION	19
4	CODING	10
5	SCREENSHOTS	25
6	CONCLUSION AND FUTURE ENHANCEMENT	29
	REFERENCES	30

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	INTRODUCTION PAGE	25
5.2	AVAILABLE BOOKS	25
5.3	VIEW MEMBERS	26
5.4	SEARCH BOOKS	26
5.5	ISSUE BOOK	27
5.6	RETURN BOOK	27
5.7	ERD	28

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Library Management System automates daily tasks like managing books, members, and transactions. Built with Python (CustomTkinter) and MySQL, it replaces manual records with a secure, efficient, and user-friendly interface for issuing, returning, and searching books..

1.2 SCOPE OF THE WORK

This project offers a complete solution for small to medium-sized libraries by digitalizing all records. It automates the management of books and members, enables quick searches, tracks issued and returned books, and minimizes manual errors. The system is scalable and can be upgraded with additional features like online access and fine calculation.

1.3 PROBLEM STATEMENT

Traditional library systems depend on manual recordkeeping, which often results in mismanagement, human errors, and time-consuming operations. A computerized system is necessary to store, update, and retrieve records efficiently and provide real-time data access.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main aim of this project is to design a user-friendly system that automates and simplifies the management of library operations. It securely stores and manages book and member information, automates the issuing, returning, and searching of books, maintains accurate transaction records, and improves overall efficiency while minimizing human errors.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10 or higher
Front - End	:	Python(CustomTkinter)
Back - End	:	MySql database
Language	:	Python 3
Libraries	:	Python libraries (mysql-connector, tkinter)
Tools used	:	MySQL Workbench, VS Code

CHAPTER 3

MODULE DESCRIPTION

This application is designed with two main modules that define how different types of users interact with the system. Instead of using a login window, the system operates through two access modes: **Administrator Mode** and **User Mode**. Each mode has its own set of responsibilities and functionalities, ensuring smooth operation and proper data handling.

1. Administrator Mode

This mode is designed for the project maintainer. The administrator can update, modify, or manage the data stored in the system. All database-related operations such as adding, deleting, and editing records are handled in this module.

2. User Mode

In this mode, the user interacts with the application by entering the number of symptoms. Based on the input, the system processes the data and displays the result in a structured table format.

CHAPTER 4

SAMPLE CODING

FRONTEND CODE

```
import customtkinter as ctk

from tkinter import messagebox

from backend import (
    get_available_books, get_issued_books, search_books,
    issue_book, return_book, get_members
)

def clear_frame(main_frame):
    for widget in main_frame.winfo_children():
        widget.destroy()

def display_table(parent, headers, rows):
```

```
table_frame = ctk.CTkScrollableFrame(parent, fg_color="white")
table_frame.pack(pady=10, padx=20, fill="both", expand=True)

for i, header in enumerate(headers):
    label = ctk.CTkLabel(table_frame, text=header, font=("Helvetica",
15, "bold"), text_color="#005f73")
    label.grid(row=0, column=i, padx=15, pady=10)

for r_index, row in enumerate(rows, start=1):
    for c_index, value in enumerate(row):
        label = ctk.CTkLabel(table_frame, text=value,
font=("Helvetica", 13))
        label.grid(row=r_index, column=c_index, padx=15, pady=6)

def create_card(parent, title, desc, button_text, command):
    frame = ctk.CTkFrame(parent, corner_radius=15, fg_color="white")
    frame.pack_propagate(False)
    frame.configure(width=280, height=180)
```

```
title_label = ctk.CTkLabel(frame, text=title, font=("Helvetica", 17,
"bold"), text_color="#005f73")

title_label.pack(pady=(15, 5))

desc_label = ctk.CTkLabel(frame, text=desc, font=("Helvetica", 13),
text_color="black", wraplength=220)

desc_label.pack(pady=(0, 10))

btn = ctk.CTkButton(frame, text=button_text, width=100, height=30,
fg_color="#0a9396",
hover_color="#007f7b", command=command)

btn.pack(pady=(5, 10))

return frame

def available_books_page(main_frame, back_to_home):
    clear_frame(main_frame)

    ctk.CTkLabel(main_frame, text=" Available Books",
```

```
font=("Helvetica", 22, "bold")).pack(pady=15)

rows = get_available_books()

display_table(main_frame, ["Book ID", "Title", "Author", "Genre",
"Year"], rows)

ctk.CTkButton(main_frame, text="⬅ Back to Home",
fg_color="#0a9396", command=back_to_home).pack(pady=20)

def issued_books_page(main_frame, back_to_home):
    clear_frame(main_frame)

    ctk.CTkLabel(main_frame, text=" Issued Books", font=("Helvetica",
22, "bold")).pack(pady=15)

    rows = get_issued_books()

    display_table(main_frame, ["Issue ID", "Book Title", "Member
Name", "Issue Date", "Due Date"], rows)

    ctk.CTkButton(main_frame, text="⬅ Back to Home",
fg_color="#0a9396", command=back_to_home).pack(pady=20)

def search_books_page(main_frame, back_to_home):
    clear_frame(main_frame)
```

```
ctk.CTkLabel(main_frame, text=" Search Books", font=("Helvetica",  
22, "bold")).pack(pady=10)
```

```
search_frame = ctk.CTkFrame(main_frame)
```

```
search_frame.pack(pady=10)
```

```
search_entry = ctk.CTkEntry(search_frame, width=300,  
placeholder_text="Enter title or author...")
```

```
search_entry.grid(row=0, column=0, padx=10)
```

```
result_frame = ctk.CTkFrame(main_frame)
```

```
result_frame.pack(fill="both", expand=True, pady=10)
```

```
def search_action():    for widget in
```

```
result_frame.winfo_children():
```

```
    widget.destroy()
```

```
    term = search_entry.get().strip()
```

```
    if not term:
```

```
    messagebox.showwarning("Empty", "Please enter a search
term.")

    return

rows = search_books(term)

display_table(result_frame, ["Book ID", "Title", "Author",
"Genre", "Year"], rows)

    ctk.CTkButton(search_frame, text="Search", fg_color="#0a9396",
command=search_action).grid(row=0, column=1, padx=10)

    ctk.CTkButton(main_frame, text="← Back to Home",
fg_color="#0a9396", command=back_to_home).pack(side="bottom",
pady=20)

def issue_book_page(main_frame, back_to_home):

    clear_frame(main_frame)

    ctk.CTkLabel(main_frame, text=" Issue Book", font=("Helvetica",
22, "bold")).pack(pady=15)

    frame = ctk.CTkFrame(main_frame)

    frame.pack(pady=20)
```

```
book_entry = ctk.CTkEntry(frame, width=200,
placeholder_text="Book ID")

book_entry.grid(row=0, column=0, padx=10)

member_entry = ctk.CTkEntry(frame, width=200,
placeholder_text="Member ID")

member_entry.grid(row=0, column=1, padx=10)

def issue_action():

    success, msg = issue_book(book_entry.get(), member_entry.get())

    if success:

        messagebox.showinfo("Success", msg)

    else:

        messagebox.showwarning("Error", msg)

    ctk.CTkButton(frame, text="Issue", fg_color="#0a9396",
command=issue_action).grid(row=1, column=0, columnspan=2,
pady=10)

    ctk.CTkButton(main_frame, text="← Back to Home",
fg_color="#0a9396", command=back_to_home).pack(side="bottom",
pady=20)
```

```
def return_book_page(main_frame, back_to_home):

    clear_frame(main_frame)

    ctk.CTkLabel(main_frame, text="➡ Return Book",
    font=("Helvetica", 22, "bold")).pack(pady=15)

    frame = ctk.CTkFrame(main_frame)

    frame.pack(pady=20)

    entry = ctk.CTkEntry(frame, width=300, placeholder_text="Enter
Issue ID")

    entry.grid(row=0, column=0, padx=10)

def return_action():

    success, msg = return_book(entry.get())

    if success:

        messagebox.showinfo("Success", msg)

    else:

        messagebox.showwarning("Error", msg)
```

```
    ctk.CTkButton(frame, text="Return", fg_color="#0a9396",
command=return_action).grid(row=0, column=1, padx=10)

    ctk.CTkButton(main_frame, text="← Back to Home",
fg_color="#0a9396", command=back_to_home).pack(side="bottom",
pady=20)

def view_members_page(main_frame, back_to_home):
    clear_frame(main_frame)

    ctk.CTkLabel(main_frame, text="View Members", font=("Helvetica",
22, "bold")).pack(pady=20)

    rows = get_members()

    if rows:
        display_table(main_frame, ["Member ID", "Name", "Email",
"Phone"], rows)

    else:
        ctk.CTkLabel(main_frame, text="No members found.",

font=("Helvetica", 14)).pack(pady=20)
```

```
    ctk.CTkButton(main_frame, text="← Back to Home",
fg_color="#0a9396", command=back_to_home).pack(side="bottom",
pady=20)
```

BACKEND CODE

```
import mysql.connector

def connect_db():

    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Angelin@2006",
        database="library_db"
    )

def get_available_books():

    conn = connect_db()

    cur = conn.cursor()

    cur.execute("""
```

```
SELECT book_id, title, author, genre, year
      FROM books
 WHERE book_id NOT IN (SELECT book_id FROM issued_books
 WHERE return_date IS NULL)
      """
rows = cur.fetchall()
conn.close()
return rows

def get_issued_books():
    conn = connect_db()
    cur = conn.cursor()
    cur.execute("""
        SELECT i.issue_id, b.title, m.name, i.issue_date, i.due_date
        FROM issued_books i
        JOIN books b ON i.book_id = b.book_id
        JOIN members m ON i.member_id = m.member_id
    """)
```

```
    WHERE i.return_date IS NULL
    """")
rows = cur.fetchall()
conn.close()
return rows

def search_books(term):
    conn = connect_db()
    cur = conn.cursor()
    cur.execute("""
        SELECT book_id, title, author, genre, year
        FROM books
        WHERE title LIKE %s OR author LIKE %s
    """, (f"%{term}%", f"%{term}%"))
    rows = cur.fetchall()
    conn.close()
    return rows
```

```
def issue_book(book_id, member_id):  
  
    conn = connect_db()  
  
    cur = conn.cursor()  
  
    cur.execute("SELECT * FROM issued_books WHERE book_id=%s  
AND return_date IS NULL", (book_id,))  
  
    if cur.fetchone():  
  
        conn.close()  
  
        return False, "Book already issued."  
  
    cur.execute("""  
        INSERT INTO issued_books (book_id, member_id, issue_date,  
        due_date)  
        VALUES (%s, %s, CURDATE(), DATE_ADD(CURDATE(),  
        INTERVAL 14 DAY))  
        """ , (book_id, member_id))  
  
    conn.commit()  
  
    conn.close()
```

```
return True, "Book issued successfully."  
  
def return_book(issue_id):  
  
    conn = connect_db()  
  
    cur = conn.cursor()  
  
    cur.execute("UPDATE issued_books SET return_date =  
    CURDATE() WHERE issue_id = %s AND return_date IS NULL",  
    (issue_id,))  
  
    conn.commit()  
  
    updated = cur.rowcount  
  
    conn.close()  
  
    if updated == 0:  
  
        return False, "Invalid or already returned Issue ID."  
  
    return True, "Book returned successfully."  
  
def get_members():  
  
    conn = connect_db()  
  
    cur = conn.cursor()
```

```
cur.execute("SELECT member_id, name, email, phone FROM  
members")
```

```
rows = cur.fetchall()
```

```
conn.close()
```

```
return rows
```

CHAPTER 5

SCREEN SHOTS

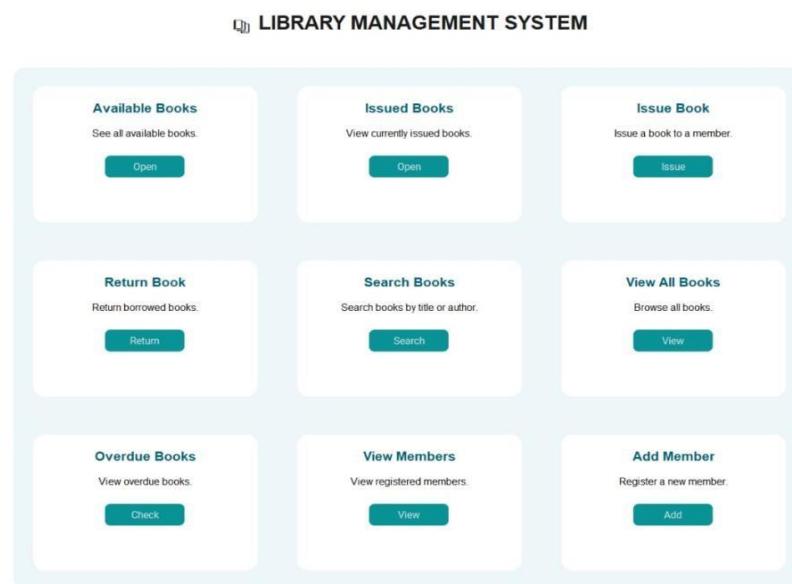


Fig 5.1 Introduction page

■ Available Books				
Book ID	Title	Author	Genre	Year
1	Harry Potter and the Sorcerer's Stone	J.K. Rowling		
2	Atomic Habits	James Clear		
3	The Alchemist	Paulo Coelho		
4	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	1997
5	Atomic Habits	James Clear	Self-help	2018
6	The Alchemist	Paulo Coelho	Fiction	1988
7	To Kill a Mockingbird	Harper Lee	Classic	1960
8	1984	George Orwell	Dystopian	1949
9	Pride and Prejudice	Jane Austen	Romance	1813
10	The Great Gatsby	F. Scott Fitzgerald	Classic	1925
11	The Hobbit	J.R.R. Tolkien	Fantasy	1937
12	The Catcher in the Rye	J.D. Salinger	Classic	1951
13	Becoming	Michelle Obama	Biography	2018
14	The Power of Habit	Charles Duhigg	Self-help	2012
15	The Da Vinci Code	Dan Brown	Thriller	2003
16	Think and Grow Rich	Napoleon Hill	Motivation	1937
17	Rich Dad Poor Dad	Robert Kiyosaki	Finance	1997

Fig 5.2 Available Books

View Members				
3 John john@gmail.com 9876873210				
4	Rebecca James	rebecca.james@yahoo.com	9123456789	
5	David Raj	david.raj@outlook.com	9876123456	
6	Aishwarya R	aishu.r@gmail.com	9988776655	
7	Michael Thomas	mthomas@gmail.com	9876001122	+ Add Member
8	Priya Sharma	priya.sharma@gmail.com	9012345678	<input type="text"/> Search <input type="button" value="Search"/>
9	Arjun Mehta	arjun.mehta@hotmail.com	9898989898	<input type="button" value="Edit Member"/>
10	Sneha Kapoor	snehakapoor@icloud.com	9700000001	
11	Rohit Verma	rohitverma@gmail.com	9822222222	
12	Ananya Iyer	ananya.iyer@gmail.com	9933445566	
13	Joseph Fernandez	joseph.f@outlook.com	9789789789	
14	Diya Pillai	diyapillai@gmail.com	9123456000	
15	Karan Singh	karansingh@gmail.com	9811111111	
16	Elena Grace	elenagrace@gmail.com	9555555555	
17	Vishal Kumar	vishal.kumar@gmail.com	9090909090	
18	sam	sam@gmail.com	234554322	
19	jash	jash@gmail.com	123456789	
20	Scarlett	scar@gmail.com	231432675	

[Back to Home](#)

Fig 5.3 View Members

Search Books				
Book ID	Title	Author	Genre	Year
1	Harry Potter and the Sorcerer's Stone	J.K. Rowling		
4	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	1997

Fig 5.4 Search Books

□ **Issued Books**

Issue ID	Book Title	Member Name	Issue Date	Due Date
4	Sapiens	John	2025-10-31	2025-11-14
5	The Alchemist	John	2025-11-04	2025-11-18

■ Issue a Book

Issue

Success

Book issued successfully.

OK

Fig 5.5 Issue Book

Issue ID	Book Title	Member Name	Issue Date	Due Date
5	The Alchemist	John	2025-11-04	2025-11-18

☒ Return a Book

Return

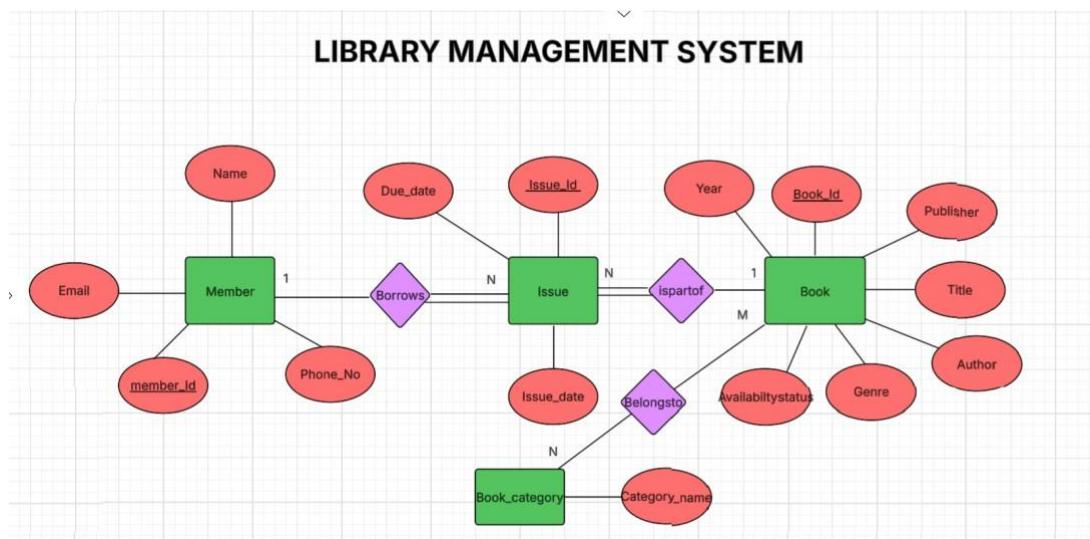
Success

Book returned successfully.

OK

Fig 5.6 Return Book

Entity Relationship Diagram:



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The Library Management System successfully demonstrates the use of database management principles in real-world applications. It helps the librarian to automate the data easily into computer. The main aim of this Database system is to reduce human efforts. The Books and the Students are given a particular unique Id no, so that they can be accessed correctly without any errors. The problems which existed earlier have been removed to a large extent with the help of this MYSQL database software. The computerization of the library management will not only improve the efficiency but will also reduce human stress thereby indirectly improving human resources. It simplifies library operations by automating manual tasks, reducing human errors, and improving efficiency. Future enhancements could include user authentication, fine calculation for overdue books, and integration with online library systems.

REFERENCES

1. https://www.researchgate.net/publication/378435639_Library_management_database_design_and_application
2. https://www.researchgate.net/publication/266650310_The_Design_and_Implementation_of_Database_on_Library_Management_Information_System
3. https://www.claudiusspress.com/assets/default/article/2023/05/15/article_1684158036.pdf
4. <https://www.geeksforgeeks.org/system-design/system-design-forlibrary-management/>
5. <https://www.scribd.com/document/639582554/PROJECTREPORT-ON-LIBRARY-DATABASE-MANAGEMENT-SYSTEM>