# LIBRARY MANAGEMENT SYSTEM

**A MINI PROJECT REPORT**

**Submitted by**

**Manimozhi I 220701160**

**Nanditha N 220701182**

**Monica D 220701172**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**RAJALAKSHMI ENGINEERING COLLEGE**

**THANDALAM**

**CHENNAI-602105**

**2024-2025**

# BONAFIDE CERTIFICATE

Certified that this project report **" LIBRARY MANAGEMENT SYSTEM"** is the bonafide work of **" Nanditha N(220701182), Manimozhi I (220701160), Monica D(220701172)"** who carried out the project under my supervision.

**Submitted for the Practical Examination held on _____**


**SIGNATURE**

**Dr.R.SABITHA**

**Professor and II Year Academic Head,**

**Computer Science and Engineering,**

**Rajalakshmi Engineering College,**

**Thandalam, Chennai - 602 105**

**SIGNATURE**

**Ms. D. KALPANA**

**Assistant Professor (SG),**

**Computer Science and Engineering,**

**Rajalakshmi Engineering College,**

**Thandalam, Chennai-602 105**

# ABSTRACT

The Library Management System is a software solution designed to streamline the management of a library's resources, including books, members, and borrowing activities. This system provides an intuitive graphical user interface (GUI) built using the Tkinter library in Python, featuring pastel colors for a visually appealing look.

The system allows librarians to perform various tasks efficiently, such as adding, updating, searching, and deleting book records. Additionally, librarians can manage member information, including names, emails, and phone numbers. The system also facilitates the management of borrowing activities, enabling librarians to track book borrowings by members, along with relevant dates.

# TABLE OF CONTENTS

**1. INTRODUCTION**

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

**2. SURVEY OF TECHNOLOGIES**

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES

2.2.1 SQL

2.2.2 PYTHON

2.2.3 TKINTER

**3. REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

**4. PROGRAM CODE**

5. **RESULTS AND DISCUSSION**

**6. CONCLUSION**

**7. REFERENCES**

# CHAPTER 1

## 1.1 INTRODUCTION

A Library Management System (LMS) is a comprehensive solution designed to facilitate the efficient management of a library's operations. This system provides an intuitive interface that allows users to easily add, view, and delete records related to various entities such as books, members, publishers, borrowers, and categories. The centralized approach not only simplifies the data management process but also enhances the accuracy and accessibility of information. This system facilitates the organization, management, and usage of library resources efficiently.

## 1.2 EXISTING SYSTEM

The existing library management system relies heavily on manual processes and outdated technology, resulting in inefficiencies and inaccuracies. The current system involves physical card catalogs for book indexing and tracking, which requires considerable time and effort from both library staff and patrons. Inventory management is done manually, leading to potential errors in book availability and status. Additionally, there is no centralized database for storing and retrieving member information, making the process of lending and returning books cumbersome and time-consuming. Communication with patrons regarding due dates and overdue books is also handled manually, often resulting in delays and missed notifications. Overall, the existing system is labor-intensive, prone to human error, and unable to meet the growing demands of our library users efficiently.

## 1.3 PROPOSED SYSTEM

The proposed Library Management System is designed to streamline and enhance the efficiency of library operations through a comprehensive, modular approach. The system incorporates several key modules: add members, add books, view members, view books, and borrow books. The 'Add Members' module facilitates the registration of new library users by capturing their personal details and generating unique member IDs. Similarly, the 'Add Books' module enables the entry of new books into the library’s catalog, including relevant

information such as title, author, genre, and availability status. The 'View Members' and 'View Books' modules provide a user-friendly interface to search and display detailed records of members and books respectively, ensuring quick access to necessary information. Finally, the 'Borrow Books' module manages the lending process, allowing members to borrow books by updating their status and tracking due dates to ensure timely returns. This systematic approach ensures that all library operations are efficiently managed, providing a seamless experience for both library staff and members.

## 1.4 OBJECTIVES

The key objective of a Library Management System is to streamline library operations by automating tasks such as cataloging, circulation, and inventory management, thereby improving resource management and ensuring data accuracy. It aims to enhance the user experience through an intuitive interface that allows easy access to and management of library resources, facilitating informed decision-making with detailed reports and analytics. Additionally, the LMS ensures data security and privacy, supports scalability to accommodate library growth, and fosters efficient communication and collaboration among staff and users. This comprehensive approach optimizes library functionality and accessibility.

## 1.5 MODULES

- ✓ Add books module
- ✓ Add members module
- ✓ Borrow book module
- ✓ View book module
- ✓ View members module

# CHAPTER 2

## 2.1 SOFTWARE DESCRIPTION

**Visual studio code:**

Visual Studio Code combines a simple source code editor with advanced development tools such as IntelliSense code completion and debugging.

## 2.2 LANGUAGES

**1. Python:**

It is used to script application logic, manage database activities, and integrate various components.

**2. Tkinter:**

Tkinter is the official Python interface for the Tk GUI toolkit. It is used to create the application's graphical user interface (GUI).

**3. MySQL:**

Structured Query Language (SQL) is a specialized programming language for managing relational database data. It allows users to store, manipulate, and retrieve data efficiently in databases like MySQL, SQL Server, Oracle, and more.

# REQUIREMENT AND ANALYSIS

## 3.1 REQUIREMENT SPECIFICATION:

**Book Management:**

- Add new books to the system, including details such as title, author, publication year, and ISBN.
- Update existing book records with revised information.
- Delete books from the system when necessary.
- Search for books based on title, author, publication year, or ISBN.
- View a list of all books in the library.

**Member Management:**

- Add new members to the system, including details such as name, email, and phone number.
- Update existing member records with revised information.
- Delete members from the system when necessary.
- Search for members based on name, email, or phone number.
- View a list of all library members.

**Borrowing Management:**

- Record book borrowings by members, including the book ID, member ID, borrow date, and return date.
- Update borrowing records with revised information, such as return dates.
- Delete borrowing records from the system when books are returned.
- Search for borrowing records based on book ID, member ID, borrow date, or return date.
- View a list of all borrowing activities.

**User Interface:**

- The system should have a user-friendly and intuitive graphical user interface (GUI) to facilitate easy navigation and operation.
- GUI elements should be properly labeled and organized to enhance usability.

**Performance:**

- The system should perform efficiently, with fast response times for database operations and GUI interactions.
- Database queries should be optimized for speed, especially when handling large volumes of data.

**Security:**

- User authentication and authorization mechanisms should be implemented to restrict access to sensitive functionalities, such as adding, updating, and deleting records.
- Data encryption should be used to secure sensitive information stored in the database.

**Reliability:**

- The system should be robust and reliable, with error handling mechanisms in place to gracefully handle unexpected situations, such as database connection failures or input validation errors.
- Data integrity should be maintained to prevent data corruption or loss.

**Scalability:**

- The system should be designed to accommodate future growth, with support for adding additional books, members, and borrowing activities without significant performance degradation.

**Database Interface:**

- The system should interface with a relational database management system (RDBMS) to store and retrieve library data.
- MySQL or another suitable RDBMS can be used for database storage.

**GUI Interface:**

- The system should provide a graphical user interface (GUI) for users to interact with the application.
- Tkinter, a standard GUI toolkit for Python, can be used to develop the GUI.

**Constraints:**

- The system must be developed using the Python programming language.
- The system must be compatible with MySQL or another suitable RDBMS for database storage.
- The system should be platform-independent and run on major operating systems such as Windows, macOS, and Linux.
- This requirement specification outlines the functional and non-functional requirements, system interfaces, and constraints of the Library Management System, providing a comprehensive overview of the system's scope and objectives.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

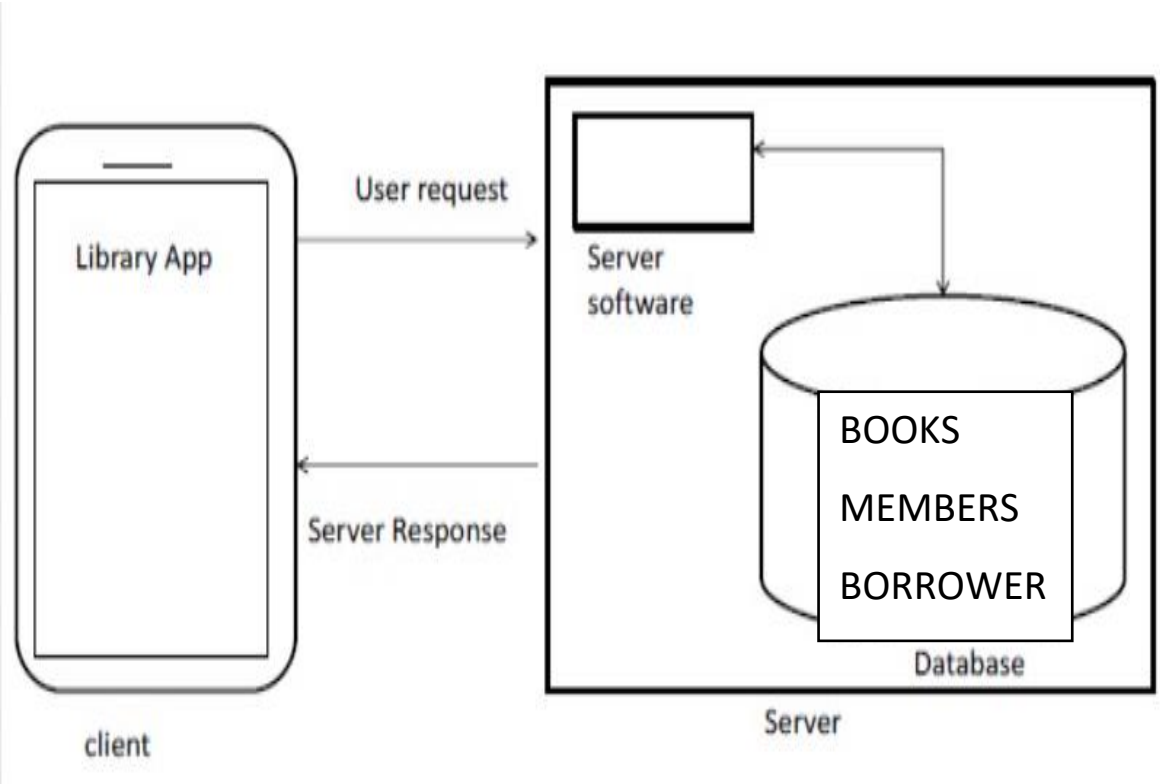
**Hardware Requirements :**

- Processor: 1 GHz or faster processor
- RAM: 2 GB or more
- Storage: At least 500 MB of available disk space
- Display: Minimum resolution of 1024x768
- Input Devices: Keyboard and mouse

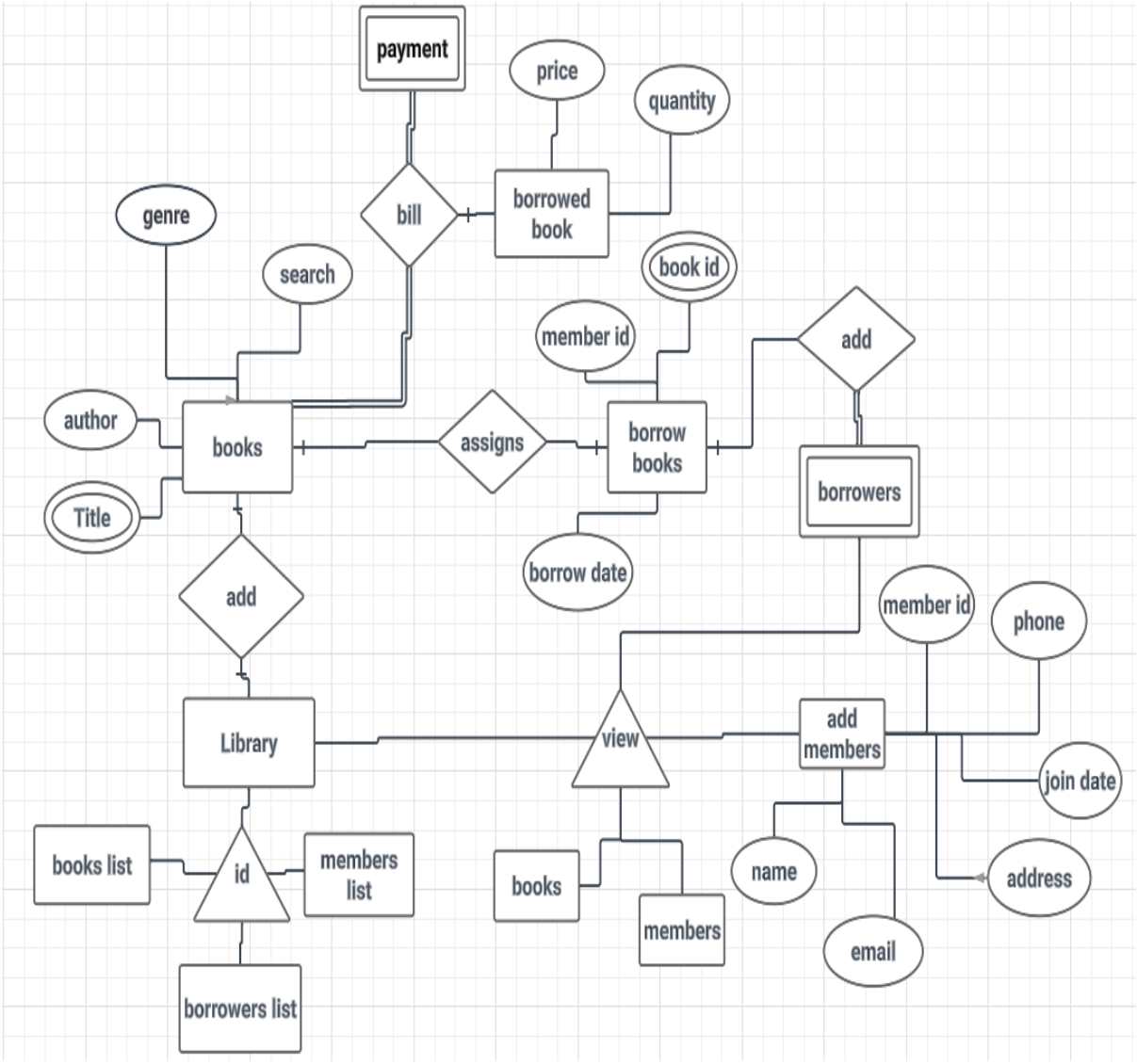
**Software Requirements:**

- Operating System: Windows 7 or later, macOS, or Linux
- Python: Version 3.6 or higher
- MySQL: Version 3 or higher
- Python Libraries:
    - ‘tkinter’ for GUI development (included with Python)
    - ‘MySQL for database management (included with Python)

## 3.3 ARCHITECTURE DIAGRAM:



## 3.4 ER DIAGRAM:



# CHAPTER-4

## PROGRAM CODE

**BACK END CODE:**

```
import mysql.connector
class LibraryDB:
    def _init_(self, host, user, password, database):
        self.conn = mysql.connector.connect(
         host="localhost",
         user="user_name",
         password="password",
         database="library"
        )
        self.cursor = self.conn.cursor()
 def add_member(self, name, email, phone, address, join_date):
        query = "INSERT INTO members (name, email, phone, address,

     join_date) VALUES (%s, %s, %s, %s, %s)"
     self.cursor.execute(query, (name, email, phone, address,join_date))
     self.conn.commit()
```

```
    def add_book(self, title, author, genre):
        query = "INSERT INTO books (title, author,  genre)
        VALUES (%s, %s, %s)"
        self.cursor.execute(query, (title, author, genre))
        self.conn.commit()
def borrow_book(self, member_id, book_id,
borrow_date, return_date):
        query = "INSERT INTO borrows (member_id,
       book_id,     borrow_date, return_date) VALUES(
      %s, %s,%s,%s)"
    self.cursor.execute(query, (member_id,
    book_id,   borrow_date,return_date))
    self.conn.commit()


    def get_books(self):
        query = "SELECT * FROM books"
         self.cursor.execute(query)
         return self.cursor.fetchall()
def get_members(self):
        query = "SELECT * FROM members"
        self.cursor.execute(query)
```

```
        return self.cursor.fetchall()

    def close(self):
        self.cursor.close()
        self.conn.close()
```

**FRONT END CODE:**

```
import tkinter as tk
from tkinter import messagebox
from library_db import LibraryDB
class LibraryApp:
    def _init_(self, root):
        self.db = LibraryDB('localhost', 'user_name',
        'password ',   'library')
       self.root = root
       self.root.title("Library Management System")
       self.root.configure(bg="#f0f0f0")
       self.create_widgets()
  def create_widgets(self):
        frame = tk.Frame(self.root, bg="#f0f0f0", padx=20, pady=20)
        frame.pack(padx=10, pady=10)
```

```
tk.Button(frame, text="Add Member", command=self.add_member_window, bg="#4CAF50", fg="white").grid(row=0, column=0, pady=10, padx=10, sticky="ew")
tk.Button(frame, text="Add Book", command=self.add_book_window, bg="#2196F3", fg="white").grid(row=1, column=0, pady=10, padx=10, sticky="ew")
```

```
tk.Button(frame, text="Borrow Book", command=self.borrow_book_window, bg="#FF9800", fg="white").grid(row=2, column=0, pady=10, padx=10, sticky="ew")

tk.Button(frame, text="View Books", command=self.view_books, bg="#9C27B0", fg="white").grid(row=3, column=0, pady=10, padx=10, sticky="ew")

tk.Button(frame, text="View Members", command=self.view_members, bg="#E91E63", fg="white").grid(row=4, column=0, pady=10, padx=10, sticky="ew")

def add_member_window(self):

    self.new_window = tk.Toplevel(self.root)

    self.app = AddMemberWindow(self.new_window, self.db)

 def add_book_window(self):

    self.new_window = tk.Toplevel(self.root)

    self.app = AddBookWindow(self.new_window, self.db)

 def borrow_book_window(self):

    self.new_window = tk.Toplevel(self.root)

    self.app = BorrowBookWindow(self.new_window, self.db)

def view_books(self):

    books = self.db.get_books()

    for book in books:

    print(book)

 def view_members(self):

    members = self.db.get_members()

    for member in members:

    print(member)
```

```
class AddMemberWindow:
    def _init_(self, root, db):
        self.db = db
        self.root = root
        self.root.title("Add Member")
        self.root.configure(bg="#f0f0f0")
        self.create_widgets()
def create_widgets(self):
        frame = tk.Frame(self.root, bg="#f0f0f0", padx=20, pady=20)
        frame.pack(padx=10, pady=10)
        tk.Label(frame, text="Name", bg="#f0f0f0").grid(
        row=0,  column=0, pady=5, padx=5, sticky="e")
        self.name_entry = tk.Entry(frame)
        self.name_entry.grid(row=0, column=1, pady=5,
        padx=5,  sticky="ew")
       tk.Label(frame, text="Email", bg="#f0f0f0").grid(
       row=1,   column=0, pady=5, padx=5, sticky="e")
       self.email_entry = tk.Entry(frame)
        self.email_entry.grid(row=1, column=1, pady=5,
        padx=5,  sticky="ew")
       tk.Label(frame, text="Phone", bg="#f0f0f0").grid(row=2,
```

```
 column=0, pady=5, padx=5, sticky="e")
  self.phone_entry = tk.Entry(frame)
self.phone_entry.grid(row=2, column=1, pady=5,
padx=5,  sticky="ew")
tk.Label(frame, text="Address", bg="#f0f0f0").grid(
row=3,  column=0, pady=5, padx=5, sticky="e")
self.address_entry = tk.Entry(frame)
self.address_entry.grid(row=3, column=1,
pady=5, padx=5,sticky="ew")
tk.Label(frame, text="Join Date (
YYYY-MM-DD)",   bg="#f0f0f0").grid(row=4,
column=0,   pady=5, padx=5, sticky="e")
self.join_date_entry = tk.Entry(frame)
self.join_date_entry.grid(row=4, column=1,
pady=5,   padx=5,sticky="ew")
tk.Button(frame, text="Add", command=
self.add_member,  bg="#4CAF50", fg="white").grid(
row=5,  column=0, columnspan=2, pady=10,
 padx=5, sticky="ew")
```

```
def add_member(self):
        name = self.name_entry.get()
        email = self.email_entry.get()
        phone = self.phone_entry.get()
        address = self.address_entry.get()
        join_date = self.join_date_entry.get()
       self.db.add_member(name, email, phone, address, join_date)
       messagebox.showinfo("Success", "Member
       added   successfully!")
       self.root.destroy()
class AddBookWindow:
    def _init_(self, root, db):
        self.db = db
        self.root = root
        self.root.title("Add Book")
        self.root.configure(bg="#f0f0f0")
        self.create_widgets()

    def create_widgets(self):
        frame = tk.Frame(self.root, bg="#f0f0f0", padx=20, pady=20)
        frame.pack(padx=10, pady=10)
```

```
        tk.Label(frame, text="Title", bg="#f0f0f0").grid(
        row=0, column=0, pady=5, padx=5, sticky="e")
       self.title_entry = tk.Entry(frame)
       self.title_entry.grid(row=0, column=1, pady=5,
       padx=5, sticky="ew")
       tk.Label(frame, text="Author", bg="#f0f0f0").grid(
      row=1, column=0, pady=5, padx=5, sticky="e")
     self.author_entry = tk.Entry(frame)
     self.author_entry.grid(row=1, column=1, pady=5,
    padx=5, sticky="ew")
   tk.Label(frame, text="Genre", bg="#f0f0f0").grid(
   row=2, column=0, pady=5, padx=5, sticky="e")
   self.genre_entry = tk.Entry(frame)
   self.genre_entry.grid(row=2, column=1, pady=5,
   padx=5, sticky="ew")
   tk.Button(frame, text="Add", command=
  self.add_book,    bg="#2196F3", fg="white").grid(
  row=3, column=0, columnspan=2, pady=10, padx=5, sticky="ew")
def add_book(self):
        title = self.title_entry.get()
        author = self.author_entry.get()
```

```
        genre = self.genre_entry.get()
        self.db.add_book(title, author, genre)
        messagebox.showinfo("Success", "Book added successfully!")
        self.root.destroy()
class BorrowBookWindow:
    def _init_(self, root, db):
        self.db = db
        self.root = root
        self.root.title("Borrow Book")
        self.root.configure(bg="#f0f0f0")
        self.create_widgets()

    def create_widgets(self):
        frame = tk.Frame(self.root, bg="#f0f0f0", padx=20, pady=20)
        frame.pack(padx=10, pady=10)
        tk.Label(frame, text="Member ID", bg="#f0f0f0").grid(
        row=0, column=0, pady=5, padx=5, sticky="e")
        self.member_id_entry = tk.Entry(frame)
        self.member_id_entry.grid(row=0, column=1, pady=5,
        padx=5, sticky="ew")
        tk.Label(frame, text="Book ID", bg="#f0f0f0").grid (
```

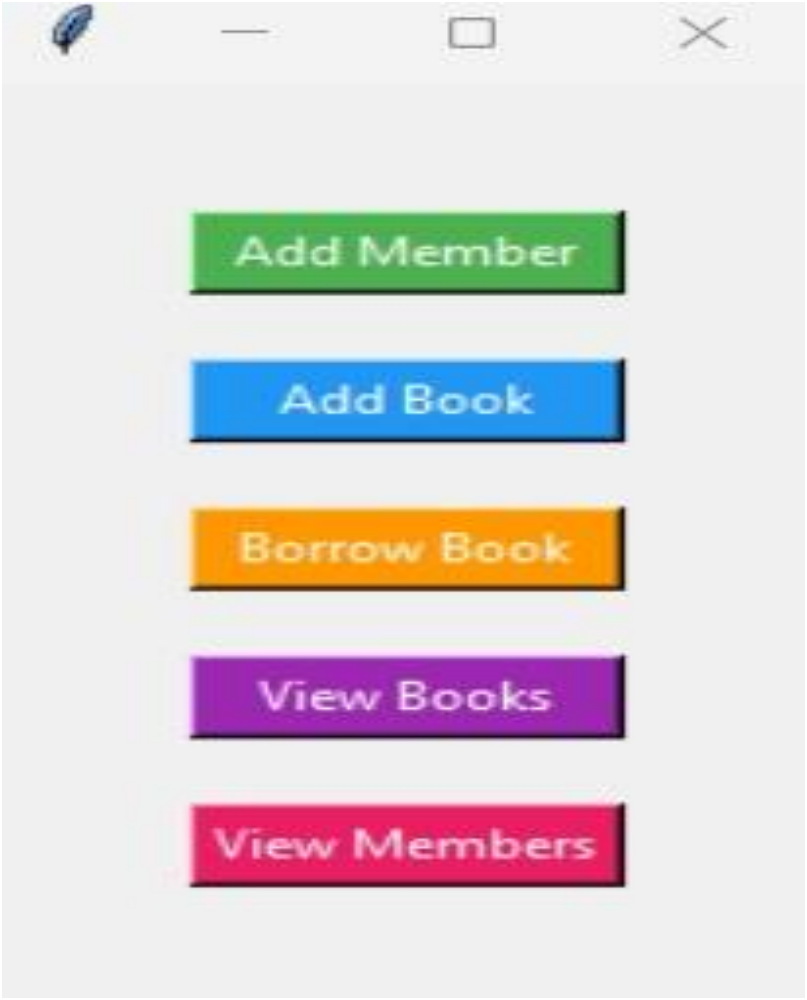
```
        row=1, column=0, pady=5, padx=5, sticky="e")
        self.book_id_entry = tk.Entry(frame)
        self.book_id_entry.grid(row=1, column=1, pady=5,
        padx=5, sticky="ew")
        tk.Label(frame, text="Borrow Date (
        YYYY-MM-DD)", bg="#f0f0f0").grid(row=2, column=0,
        pady=5, padx=5, sticky="e")
self.borrow_date_entry = tk.Entry(frame) self.borrow_date_entry.grid(row=2, column=1, pady=5, padx=5, sticky="ew")
tk.Label(frame, text="Return Date (YYYY-MM-DD)", bg="#f0f0f0").grid(row=3, column=0, pady=5, padx=5, sticky="e")
        self.return_date_entry = tk.Entry(frame)
        self.return_date_entry.grid(row=3, column=1, pady=5,
        padx=5,  sticky="ew")
        tk.Button(frame, text="Borrow", command=
        self.borrow_book, bg="#FF9800", fg="white").grid(
        row=4, column=0, columnspan=2, pady=10, padx=5, sticky="ew")
    def borrow_book(self):
        member_id = self.member_id_entry.get()
        book_id = self.book_id_entry.get()
        borrow_date = self.borrow_date_entry.get()
        return_date = self.return_date_entry.get()
```

```
        self.db.borrow_book(member_id, book_id,
        borrow_date, return_date)



messagebox.showinfo("Success", "Book borrowed successfully!")
self.root.destroy()
if _name_ == "_main_":
    root = tk.Tk()
    app = LibraryApp(root)
    root.mainloop()
```

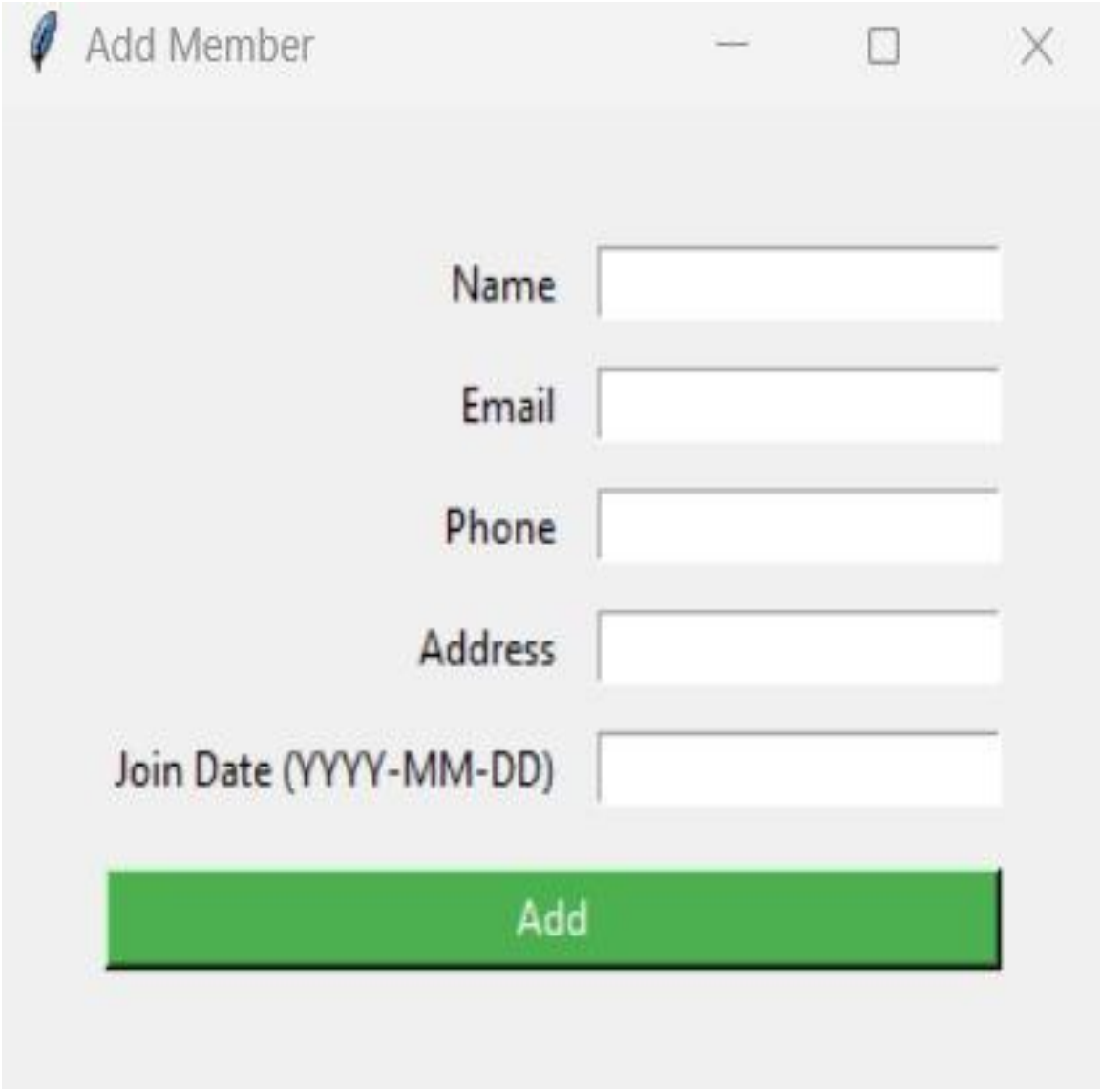
# CHAPTER -5

## RESULTS AND DISCUSSION

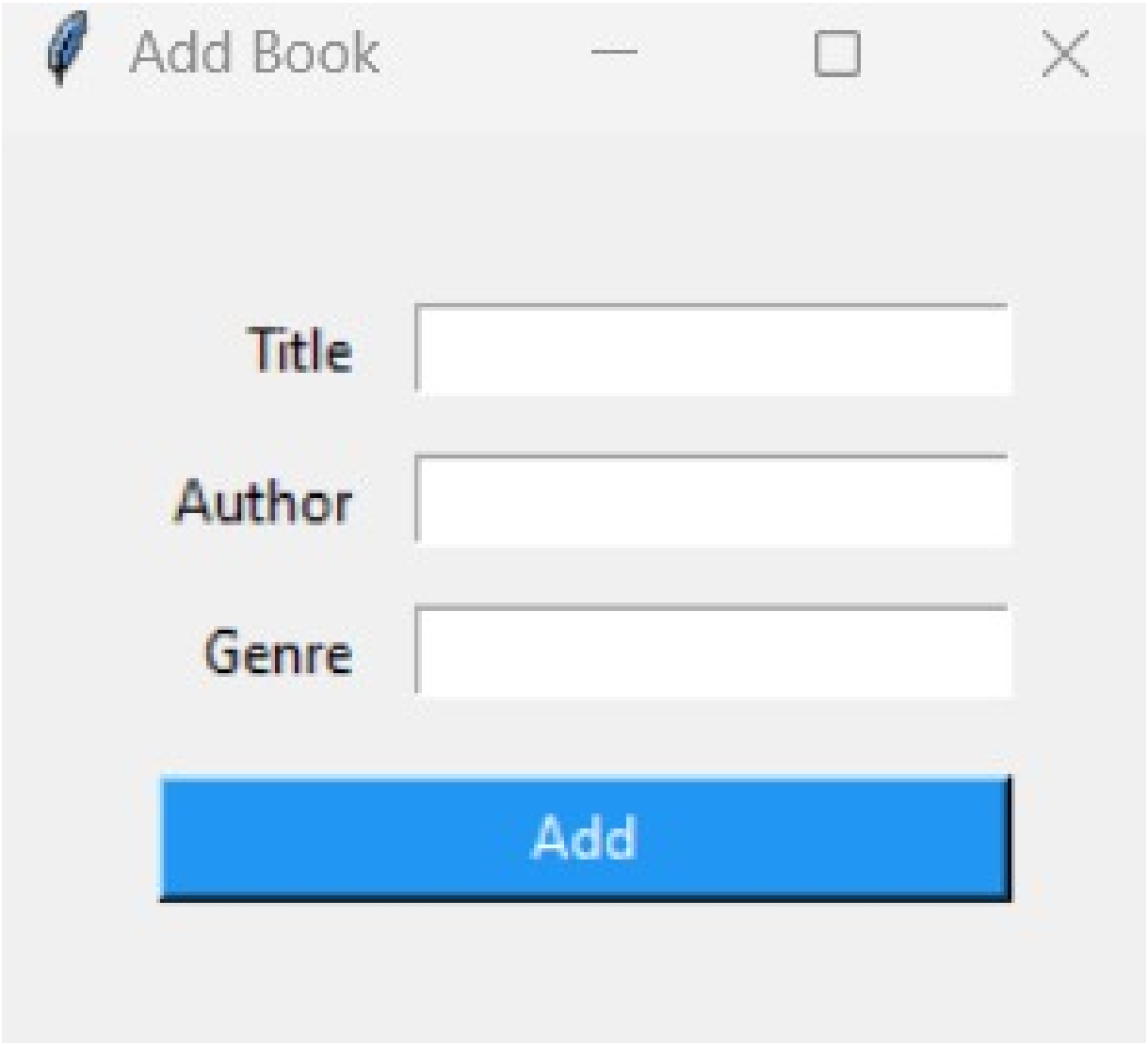
**LIBRARY MANAGEMENT SYSTEM:**



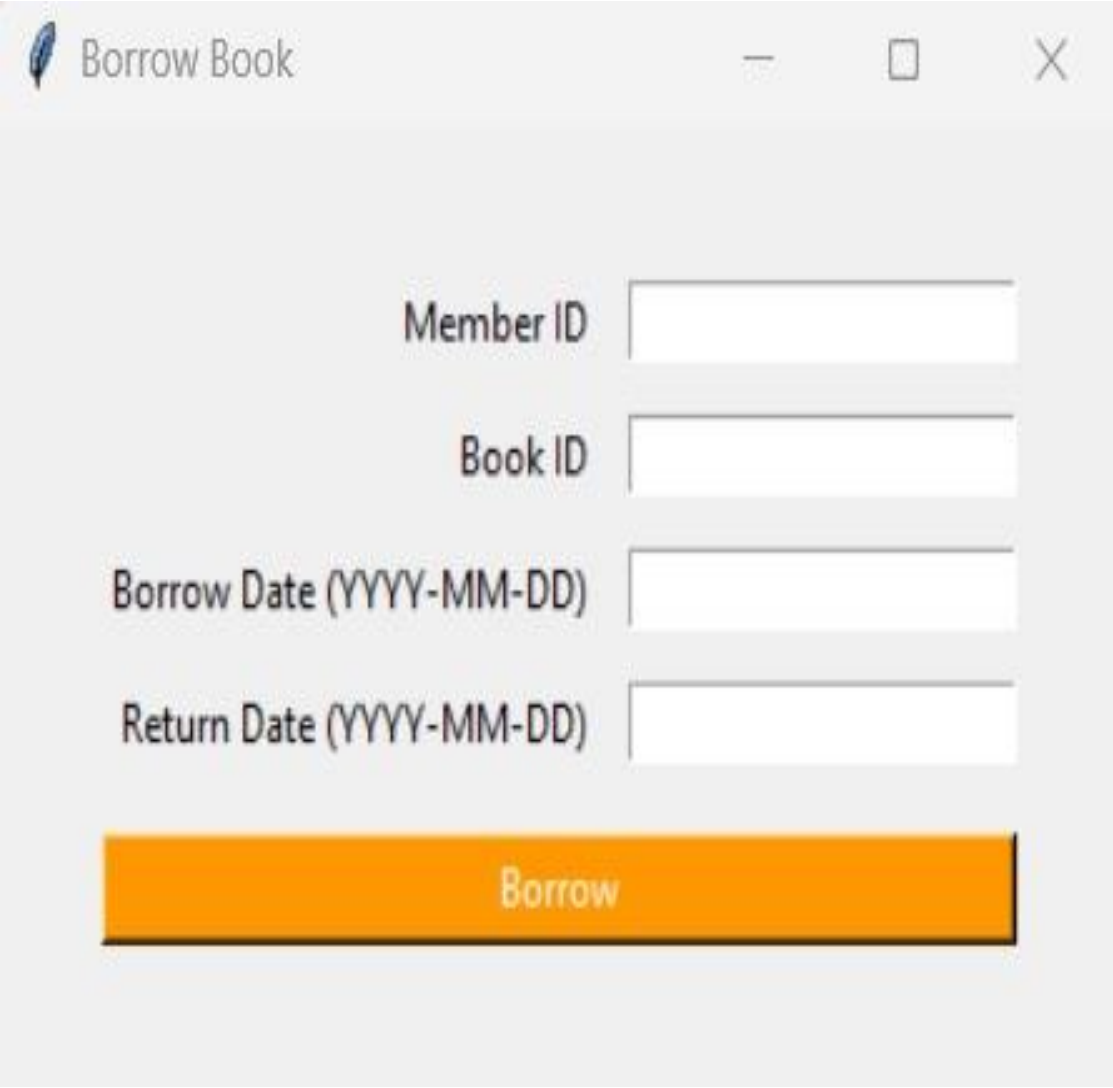
**MEMBER MODULE:**



**BOOK MODULE:**



**BORROW MODULE:**




## CHAPTER-6

## 6.1 CONCLUSION:

In conclusion, the Library Management System (LMS) is a vital tool for modern libraries to efficiently organize and manage their resources. By automating various tasks such as book management, member management, and borrowing activities, the LMS enhances the overall productivity of librarians and improves the user experience for library patrons.

Throughout this project, we have designed a comprehensive system with a user-friendly graphical interface, leveraging the Tkinter library in Python. The system allows librarians to perform essential functions such as adding, updating, searching, and deleting records for books, members, and borrowers. Additionally, it provides functionalities to track borrowing activities, ensuring accurate records of book loans and returns.

With its intuitive interface and robust functionalities, the LMS not only simplifies day-to-day library operations but also contributes to better organization, increased efficiency, and enhanced service delivery. Moreover, by adhering to stringent security measures and implementing error handling mechanisms, the system ensures data integrity and reliability.

Overall, the Library Management System presented here serves as a valuable tool for libraries of all sizes, enabling them to optimize their resources, streamline workflows, and provide an exceptional experience for library users. Through continuous refinement and adaptation to evolving requirements, the LMS can further contribute to the advancement of library services and the promotion of knowledge dissemination in the community.

# CHAPTER-7

## 7.1 REFERENCES


1. https://docs.python.org/3/ [https://docs.python.org/3/](https://docs.python.org/3/)

2. https://docs.python.org/3/library/tkinter.html

3. MySQL Documentation: MySQL. (n.d.). MySQL Documentation. Retrieved from [https://www. MySQL.org/docs.html](https://www.sqlite.org/docs.html)

4. https://www.skoolbeep.com/blog/library-management-system/

5. https://realpython.com/python-gui-tkinter/

6. https://www.w3schools.com/mysql/mysql_rdbms.asp

7. Database System Concepts: Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill.

8. GeeksforGeeks: Various authors. (n.d.). GeeksforGeeks. Retrieved from [https://www.geeksforgeeks.org/](https://www.geeksforgeeks.org/)

9. W3Schools: W3Schools. (n.d.). SQL Tutorial. Retrieved from [https://www.w3schools.com/sql/](https://www.w3schools.com/sql/)