**LIBRARY MANAGEMENT SYSTEM**

**A MINI PROJECT REPORT**

**Submitted by**

**Manimozhi I 220701160**

**Monica D 220701172**

**Nanditha N 220701182**

***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 thebonafide work of **“Manimozhi I (220701160),  Monica D(220701172), NANDITHA N(220701168)”** who carried out the project under my supervision.

**Submitted for the Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SIGNATURE                                                                  SIGNATURE**

**Dr.R.SABITHA                                               Ms. D. KALPANA**

**Professor and II Year Academic Head         Assistant Professor (SG)**

**Computer Science and Engineering,           Computer Science and Engineering**

**Rajalakshmi Engineering College                Rajalakshmi Engineering College**

**Thandalam, Chennai - 602 105                     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 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.

**1.3 MODULES**

* Admin module
* Borrower 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

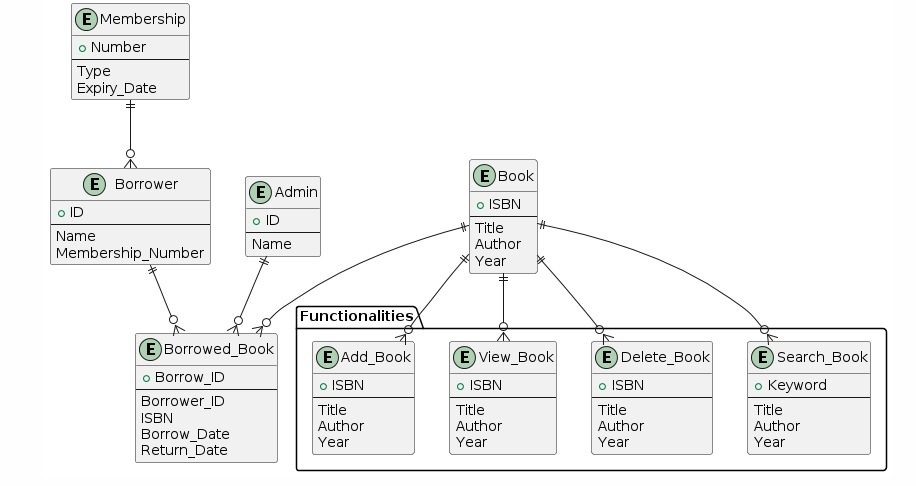
 - MySQL: Version 3 or higher

 - Python Libraries:

      - ‘tkinter’ for GUI development (included with Python)

      - ‘MySQL for database management (included with Python)

**3.3 ARCHITECTURE DIAGRAM:**

****

**3.4 ER DIAGRAM**

BOOKS

Contains

Library

**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="root",

password="Na#nDI!21?06\_#",

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', 'root', 'Na#nDI!21?06\_#', '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()

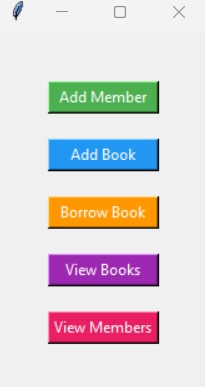
app = LibraryApp(root)

root.mainloop()

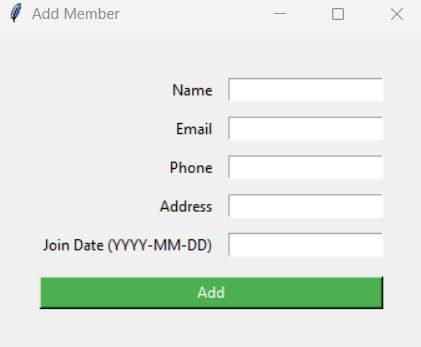
**CHAPTER -5**

**RESULTS AND DISCUSSION**

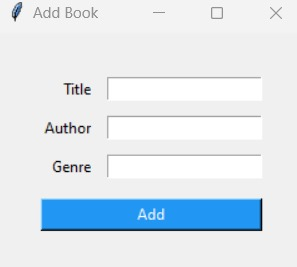
**LIBRARY MANAGEMENT SYSTEM:**

****

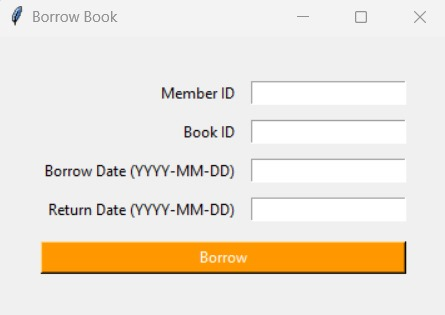
**MEMBER MODULE:**

****

**BOOK MODULE:**



**BORROW MODULE:**



**CHAPTER - 6**

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/)