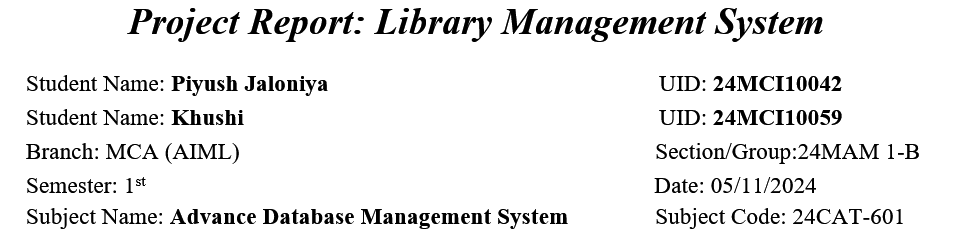
****

# Title of the Project:

## Library Management System

## Aim

The aim of this project is to develop a user-friendly Library Management System using Python and Tkinter to facilitate efficient management of library resources, including books and borrowers.

## Objectives

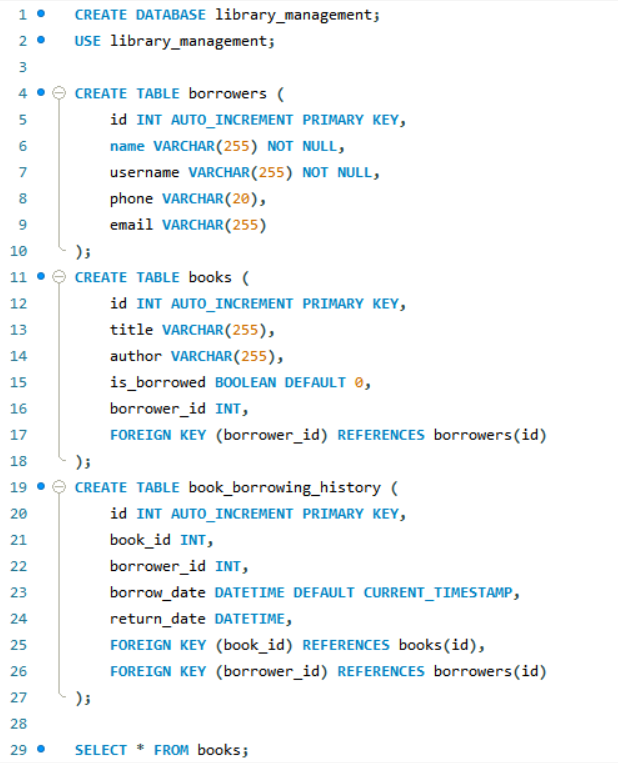
1. To create a database schema for managing books and borrower information.  
2. To develop a GUI application for adding, viewing, borrowing, and returning books.  
3. To implement database connectivity using MySQL to store and retrieve data.  
4. To ensure user-friendly interactions through informative message boxes and error handling.

## Tasks to be Done

1. Set up the MySQL database and create necessary tables for books and borrowers.  
2. Design the Tkinter GUI layout for user interactions.  
3. Implement functions to add, delete, borrow, and return books.  
4. Test the application for functionality and user experience.

## Code

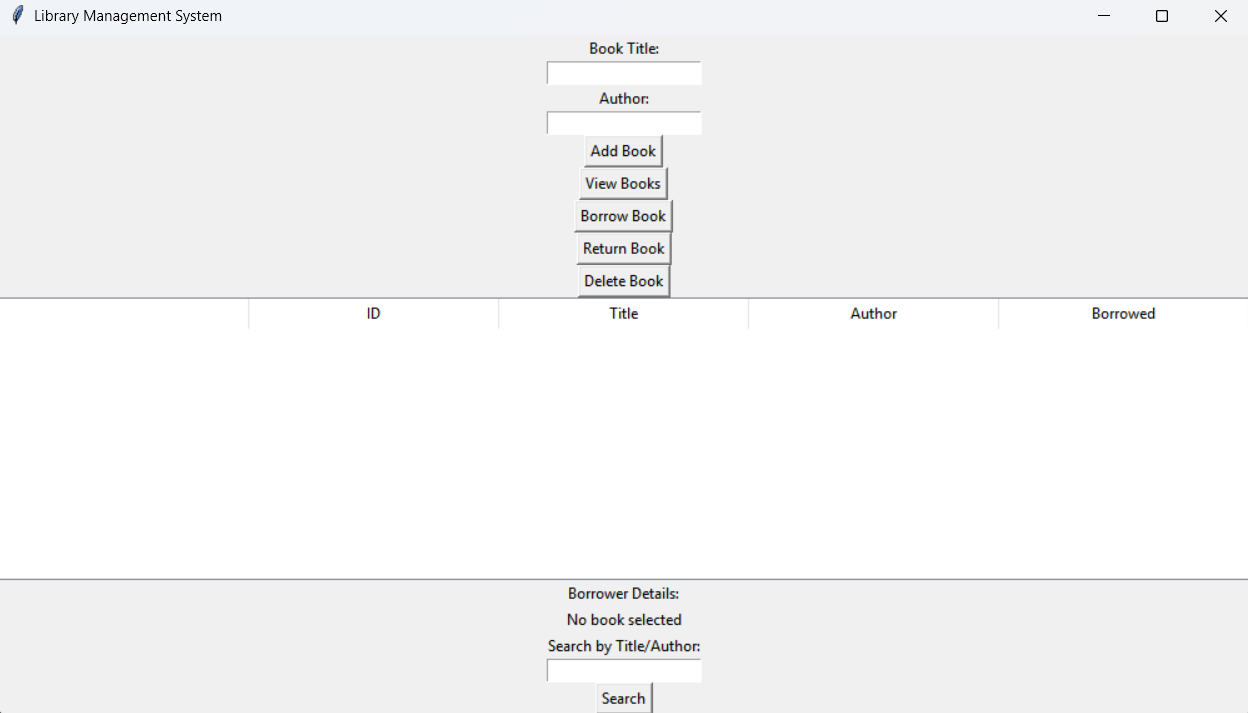
* **Database Query**



* **Python Code**

import tkinter as tk  
from tkinter import messagebox, ttk  
import mysql.connector  
  
# Database connection details  
db\_config = {  
 'user': 'root',   
 'password': 'piyush',   
 'host': 'localhost',  
 'database': 'library\_management'  
}  
  
# Database functions  
def add\_book(title, author):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('INSERT INTO books (title, author) VALUES (%s, %s)', (title, author))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
def delete\_book(book\_id):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('DELETE FROM books WHERE id = %s', (book\_id,))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
def view\_books():  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('SELECT \* FROM books')  
 rows = cursor.fetchall()  
 cursor.close()  
 conn.close()  
 return rows  
  
def get\_borrower\_details(borrower\_id):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('SELECT \* FROM borrowers WHERE id = %s', (borrower\_id,))  
 borrower = cursor.fetchone()  
 cursor.close()  
 conn.close()  
 return borrower  
  
def borrow\_book(name, username, phone, email, book\_id):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
  
 # Insert borrower details  
 cursor.execute('INSERT INTO borrowers (name, username, phone, email) VALUES (%s, %s, %s, %s)',   
 (name, username, phone, email))  
 borrower\_id = cursor.lastrowid  
  
 # Update book with borrower\_id  
 cursor.execute('UPDATE books SET is\_borrowed = 1, borrower\_id = %s WHERE id = %s', (borrower\_id, book\_id))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
def return\_book(book\_id):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('UPDATE books SET is\_borrowed = 0, borrower\_id = NULL WHERE id = %s', (book\_id,))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
# Tkinter GUI  
class LibraryApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Library Management System")  
  
 # Input fields for book info  
 self.title\_label = tk.Label(root, text="Book Title:")  
 self.title\_label.pack()  
 self.title\_entry = tk.Entry(root)  
 self.title\_entry.pack()  
  
 self.author\_label = tk.Label(root, text="Author:")  
 self.author\_label.pack()  
 self.author\_entry = tk.Entry(root)  
 self.author\_entry.pack()  
  
 self.add\_button = tk.Button(root, text="Add Book", command=self.add\_book)  
 self.add\_button.pack()  
  
 self.view\_button = tk.Button(root, text="View Books", command=self.view\_books)  
 self.view\_button.pack()  
  
 self.borrow\_button = tk.Button(root, text="Borrow Book", command=self.borrow\_book)  
 self.borrow\_button.pack()  
  
 self.return\_button = tk.Button(root, text="Return Book", command=self.return\_book)  
 self.return\_button.pack()  
  
 self.delete\_button = tk.Button(root, text="Delete Book", command=self.delete\_book)  
 self.delete\_button.pack()  
  
 self.books\_tree = ttk.Treeview(root, columns=("ID", "Title", "Author", "Borrowed"))  
 self.books\_tree.heading('#1', text='ID')  
 self.books\_tree.heading('#2', text='Title')  
 self.books\_tree.heading('#3', text='Author')  
 self.books\_tree.heading('#4', text='Borrowed')  
 self.books\_tree.pack()  
  
 self.borrower\_label = tk.Label(root, text="Borrower Details:")  
 self.borrower\_label.pack()  
  
 self.borrower\_info = tk.Label(root, text="No book selected")  
 self.borrower\_info.pack()  
  
 def add\_book(self):  
 title = self.title\_entry.get()  
 author = self.author\_entry.get()  
 if title and author:  
 add\_book(title, author)  
 messagebox.showinfo("Success", "Book added successfully!")  
 self.title\_entry.delete(0, tk.END)  
 self.author\_entry.delete(0, tk.END)  
 else:  
 messagebox.showwarning("Input Error", "Please enter both title and author.")  
  
 def view\_books(self):  
 for row in self.books\_tree.get\_children():  
 self.books\_tree.delete(row)  
 books = view\_books()  
 for book in books:  
 self.books\_tree.insert("", tk.END, values=book)  
  
 def borrow\_book(self):  
 selected\_item = self.books\_tree.selection()  
 if selected\_item:  
 book\_id = self.books\_tree.item(selected\_item, 'values')[0]  
  
 # Create the borrow window for user details  
 self.borrow\_window = tk.Toplevel(self.root)  
 self.borrow\_window.title("Borrow Book")  
  
 tk.Label(self.borrow\_window, text="Enter your Name:").pack()  
 self.name\_entry = tk.Entry(self.borrow\_window)  
 self.name\_entry.pack()  
  
 tk.Label(self.borrow\_window, text="Enter your Username:").pack()  
 self.username\_entry = tk.Entry(self.borrow\_window)  
 self.username\_entry.pack()  
  
 tk.Label(self.borrow\_window, text="Enter your Phone Number:").pack()  
 self.phone\_entry = tk.Entry(self.borrow\_window)  
 self.phone\_entry.pack()  
  
 tk.Label(self.borrow\_window, text="Enter your Email:").pack()  
 self.email\_entry = tk.Entry(self.borrow\_window)  
 self.email\_entry.pack()  
  
 borrow\_button = tk.Button(self.borrow\_window, text="Borrow Book", command=lambda: self.borrow\_confirm(book\_id))  
 borrow\_button.pack()  
  
 else:  
 messagebox.showwarning("Selection Error", "Please select a book to borrow.")  
  
 def borrow\_confirm(self, book\_id):  
 name = self.name\_entry.get()  
 username = self.username\_entry.get()  
 phone = self.phone\_entry.get()  
 email = self.email\_entry.get()  
  
 if name and username and phone and email:  
 borrow\_book(name, username, phone, email, book\_id)  
 messagebox.showinfo("Success", "Book borrowed successfully!")  
 self.view\_books()  
 self.borrow\_window.destroy() # Close the borrow window  
 self.show\_borrower\_details(book\_id) # Display borrower details  
 else:  
 messagebox.showwarning("Input Error", "Please enter all user details.")  
  
 def return\_book(self):  
 selected\_item = self.books\_tree.selection()  
 if selected\_item:  
 book\_id = self.books\_tree.item(selected\_item, 'values')[0]  
 return\_book(book\_id)  
 messagebox.showinfo("Success", "Book returned successfully!")  
 self.view\_books()  
 self.borrower\_info.config(text="No book selected")  
 else:  
 messagebox.showwarning("Selection Error", "Please select a book to return.")  
  
 def delete\_book(self):  
 selected\_item = self.books\_tree.selection()  
 if selected\_item:  
 book\_id = self.books\_tree.item(selected\_item, 'values')[0]  
 delete\_book(book\_id)  
 messagebox.showinfo("Success", "Book deleted successfully!")  
 self.view\_books()  
 else:  
 messagebox.showwarning("Selection Error", "Please select a book to delete.")  
  
 def show\_borrower\_details(self, book\_id):  
 conn = mysql.connector.connect(\*\*db\_config)  
 cursor = conn.cursor()  
 cursor.execute('SELECT borrower\_id FROM books WHERE id = %s', (book\_id,))  
 borrower\_id = cursor.fetchone()[0]  
  
 if borrower\_id:  
 borrower = get\_borrower\_details(borrower\_id)  
 borrower\_text = f"Name: {borrower[1]}  
Username: {borrower[2]}  
Phone: {borrower[3]}  
Email: {borrower[4]}"  
 self.borrower\_info.config(text=borrower\_text)  
 cursor.close()  
 conn.close()  
  
# Run the app  
root = tk.Tk()  
app = LibraryApp(root)  
root.mainloop()

## Output

.

## Learning Outcomes

* Gained practical experience in using Tkinter for GUI development in Python.
* Developed skills in integrating Python applications with MySQL databases.
* Understood the importance of user interface design for improving user experience.
* Learned how to handle user inputs and validate data in a real-world application.
* Enhanced problem-solving skills by implementing various functionalities in the library management context.

## Conclusions

The Library Management System successfully meets its objectives of simplifying library operations. Through this project, I have learned valuable programming and database management skills. The system can be further enhanced by adding features like search functionality, late fee calculation, and user authentication to improve overall user experience and efficiency.