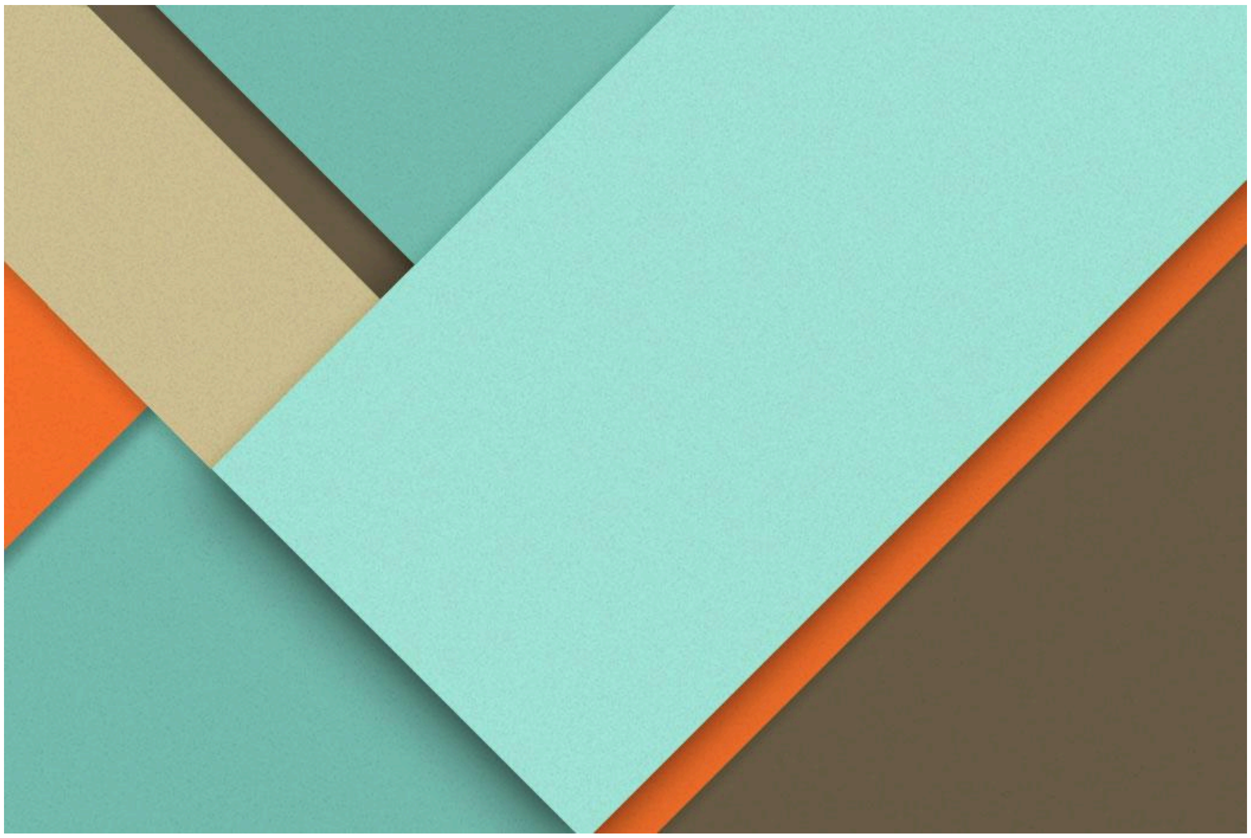


Library Management System

27.12.2024

—

By V.Jalagandaiswaran

Overview

This **Library Management System** project is a console-based application designed to manage books, users, and transactions in a library. Here's what you can do with this project and the kind of output you can expect when running it:

Use Of the Project

1. **Book Management**:
    - **Add Books**: Add new books with unique IDs, titles, and authors to the library.
    - **Delete Books**: Remove books from the library database.
    - **Update Book Status**: Automatically updates the status of a book to "issued" when it is borrowed and back to "available" when it is returned.

2. **User Management**:
    - **Register Users**: Create user accounts by registering with a username and password.
    - **Login**: Authenticate users by verifying their credentials.

3. **Transaction Management**:

- **Issue Books**: Allow users to borrow books if they are available. Records the issue date and sets the book's status to "issued."
- **Return Books**: Process book returns by recording the return date and updating the book's status back to "available."

4. **Record Keeping**:

- All actions (e.g., issuing, returning books) are stored in the database. This ensures the library has a history of which user borrowed or returned a specific book.

CODE:

```
import sqlite3
import datetime

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

# Create tables
cursor.execute('''CREATE TABLE IF NOT EXISTS books (
                    book_id TEXT PRIMARY KEY,
                    title TEXT,
                    author TEXT,
                    status TEXT
                  )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS users (
                    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT UNIQUE,
                    password TEXT
                  )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS transactions (
                    transaction_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER,
                    book_id TEXT,
                    issue_date TEXT,
                    return_date TEXT,
            FOREIGN KEY(user_id) REFERENCES users(user_id),
            FOREIGN KEY(book_id) REFERENCES books(book_id)
          )''')

conn.commit()

# Functions for managing books
def add_book(book_id, title, author):
    cursor.execute("SELECT * FROM books WHERE book_id = ?", (book_id,))
    if cursor.fetchone():
        print(f"Book with ID {book_id} already exists.")
        return
    cursor.execute("INSERT INTO books (book_id, title, author, status) VALUES (?, ?, ?,
?)", (book_id, title, author, "available"))
    conn.commit()
    print(f"Book {title} added successfully!")

def delete_book(book_id):
    cursor.execute("DELETE FROM books WHERE book_id = ?", (book_id,))
    conn.commit()

def update_book_status(book_id, status):
    cursor.execute("UPDATE books SET status = ? WHERE book_id = ?", (status, book_id))
    conn.commit()

# Functions for user login and registration
def register_user(username, password):
    try:
        cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)",
(username, password))
```

```
        conn.commit()
        print("User registered successfully!")
    except sqlite3.IntegrityError:
        print("Username already exists. Please choose a different one.")

def login_user(username, password):
    cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?",
(username, password))
    user = cursor.fetchone()
    if user:
        return user[0]  # Return user_id
    else:
        print("Invalid username or password")
        return None

# Functions for issuing and returning books
def issue_book(user_id, book_id):
    cursor.execute("SELECT status FROM books WHERE book_id = ?", (book_id,))
    book = cursor.fetchone()
    if not book:
        print("Book not found.")
        return
    if book[0] == "available":
        issue_date = datetime.date.today().strftime('%Y-%m-%d')
        cursor.execute("INSERT INTO transactions (user_id, book_id, issue_date,
return_date) VALUES (?, ?, ?, ?)",
                       (user_id, book_id, issue_date, None))
        update_book_status(book_id, "issued")
        conn.commit()
        print("Book issued successfully!")
    else:
        print("Book is already issued.")

def return_book(transaction_id):
    cursor.execute("SELECT book_id FROM transactions WHERE transaction_id = ?",
(transaction_id,))
    transaction = cursor.fetchone()
    if not transaction:
        print("Invalid transaction ID.")
        return
    book_id = transaction[0]
    return_date = datetime.date.today().strftime('%Y-%m-%d')
    cursor.execute("UPDATE transactions SET return_date = ? WHERE transaction_id =
?", (return_date, transaction_id))
    update_book_status(book_id, "available")
    conn.commit()
    print("Book returned successfully!")

# Function to add multiple books
def add_books():
    books = [
        ("H1H1", "Harry Potter and the Philosopher's Stone", "J.K. Rowling"),
        ("H2H2", "Feathers of Fire", "Dr. APJ Abdul Kalam"),
        ("M3M3", "Mahabharatha", "Vedavyasa"),
        ("R4R4", "Srimath Ramayanam", "Valmiki"),
        ("B5B5", "Srimath Bagvatham", "Vedavyasa"),
        ("L6R6", "Lord of the Rings", "J.R.R. Tolkien"),
        ("S7S7", "Sundarakanda", "Valmiki"),
        ("H3H3", "Harry Potter and the Chamber of Secrets", "J.K. Rowling"),
        ("H4H4", "Harry Potter and the Prisoner of Azkaban", "J.K. Rowling"),
        ("H5H5", "Harry Potter and the Goblet of Fire", "J.K. Rowling"),
        ("H6H6", "Harry Potter and the Order of the Phoenix", "J.K. Rowling"),
        ("H7H7", "Harry Potter and the Half-Blood Prince", "J.K. Rowling"),
        ("H8H8", "Harry Potter and the Deathly Hallows", "J.K. Rowling"),
    ]

    for book_id, title, author in books:
        add_book(book_id, title, author)

# User Interface and Main Program
def main():
    print("Welcome to the Library Management System")
    add_books()  # Automatically add the books when the program starts
    while True:
        print("\n1. Register\n2. Login\n3. Exit")
        choice = input("Enter your choice: ").strip().lower()
        if choice in ['1', 'register']:
            username = input("Enter username: ")
            password = input("Enter password: ")
            register_user(username, password)
        elif choice in ['2', 'login']:
            username = input("Enter username: ")
            password = input("Enter password: ")
            user_id = login_user(username, password)
            if user_id:
                while True:
                    print("\n1. Add Book\n2. Delete Book\n3. Issue Book\n4. Return Book\n5.
Logout")
                    user_choice = input("Enter your choice: ").strip().lower()
                    if user_choice in ['1', 'add book']:
                        book_id = input("Enter book ID: ")
                        title = input("Enter book title: ")
                        author = input("Enter book author: ")
                        add_book(book_id, title, author)
                    elif user_choice in ['2', 'delete book']:
                        book_id = input("Enter book ID to delete: ")
                        delete_book(book_id)
                    elif user_choice in ['3', 'issue book']:
                        book_id = input("Enter book ID to issue: ")
                        issue_book(user_id, book_id)
                    elif user_choice in ['4', 'return book']:
                        transaction_id = input("Enter transaction ID to return: ")
                        return_book(transaction_id)
                    elif user_choice in ['5', 'logout']:
                        break
                    else:
                        print("Invalid choice. Please try again.")
        elif choice in ['3', 'exit']:
            print("Thank you for using the Library Management System. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Expected Outputs

Startup:

When you start the program, it automatically initializes a set of predefined books into the database (if they don't already exist). You'll see messages indicating the initialization:

SQL

```
Welcome to the Library Management System
Book with ID H1H1 already exists.
Book with ID H2H2 already exists.
Book with ID H8H8 already exists.
```

Main Menu:

You'll see options for interacting with the system:

```
1. Register
2. Login
3. Exit
Enter your choice:
```

Register a New User:

If you choose to register, you'll provide a username and password:

```
Enter username: john_doe
Enter password: password123
User registered successfully!
```

If the username is already taken:

```
Username already exists. Please choose a different one.
```

Login as a User:

If you log in with valid credentials:

```
Enter username: john_doe
Enter password: password123
Logged in successfully!
```

If the credentials are invalid:

```
Invalid username or password
```

Logged-In User Menu:

Once logged in, you'll have access to these options:

```
1. Add Book
2. Delete Book
3. Issue Book
4. Return Book
5. Logout
Enter your choice:
```

1.Add a New Book:

When adding a book:

```
Enter book ID: L9L9
Enter book title: The Great Gatsby
Enter book author: F. Scott Fitzgerald
Book The Great Gatsby added successfully!
```

If the book ID already exists:

```
Book with ID L9L9 already exists.
```

2. Delete a Book:

When deleting a book:

```
Enter book ID to delete: L9L9
Book deleted successfully!
```

3.Issue a Book:

When issuing a book:

```
Enter book ID to issue: H1H1
Book issued successfully!
```

If the book is already issued:

```
Book is already issued.
```

If the book ID doesn’t exist:

```
Book not found.
```

4.Return a Book:

When returning a book:

```
Enter transaction ID to return: 1
Book returned successfully
```

If the transaction ID doesn’t exist:

```
Invalid transaction ID.
```

5.Logout:

```
You have been logged out.
```

Exit the Program:

If you choose to exit:

2.Library GUI Extension

To incorporate the additional features, I'll upgrade the project to include the following functionalities:

1. **Graphical User Interface (GUI)** using tkinter.
2. **Search Functionality** to find books by title or author.
3. **Overdue Notifications** for late book returns.
4. **Reports** to show borrowing trends or most popular books.

Code:

```
import sqlite3

import datetime

from tkinter import Tk, Label, Entry, Button, Listbox, END,
messagebox


# Database setup

conn = sqlite3.connect('library.db')

cursor = conn.cursor()
```

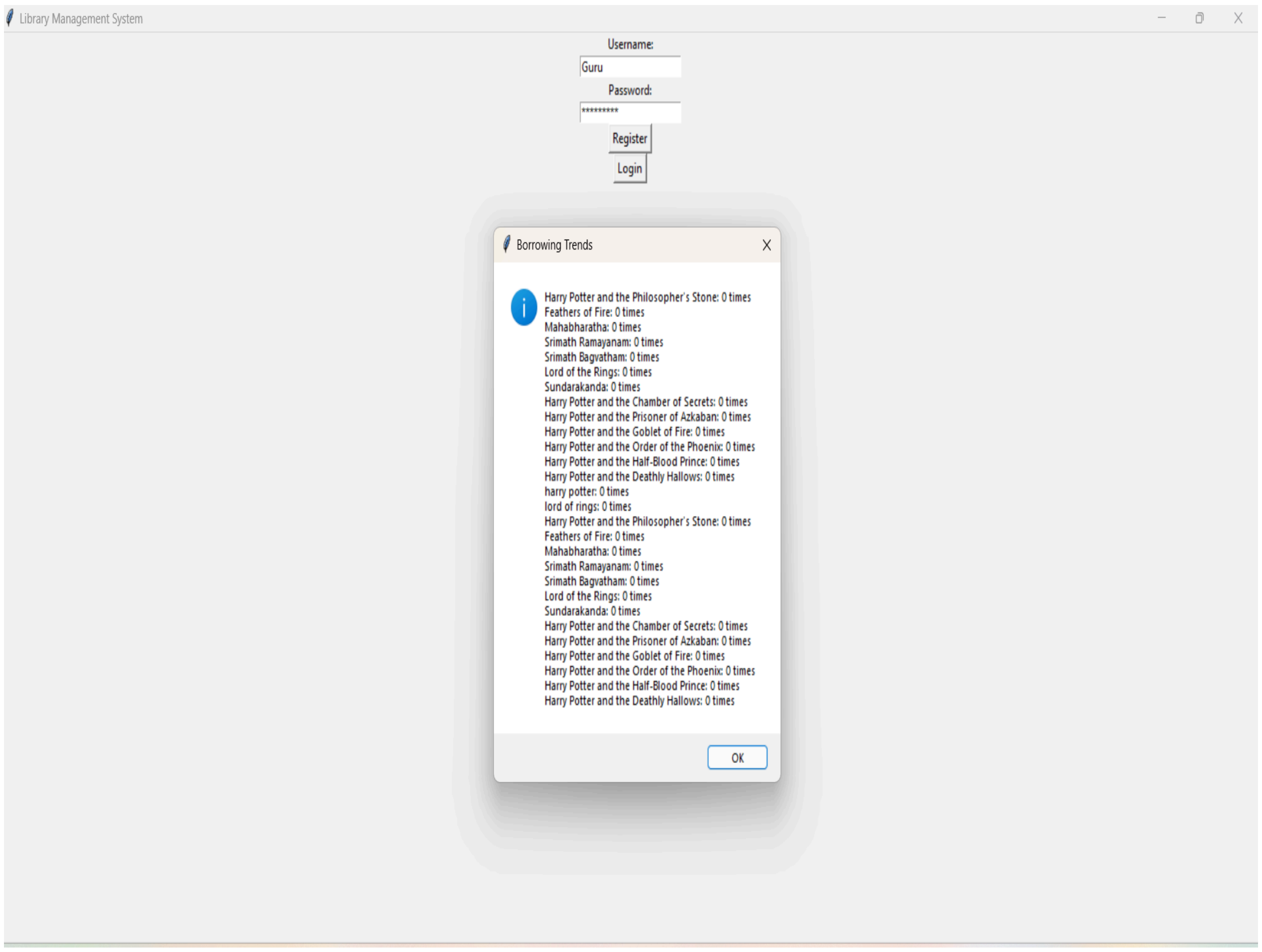
```
# Create tables
cursor.execute('''CREATE TABLE IF NOT EXISTS books (
                    book_id TEXT PRIMARY KEY,
                    title TEXT,
                    author TEXT,
                    status TEXT
                 )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS users (
                    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT UNIQUE,
                    password TEXT
                 )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS transactions (
                    transaction_id INTEGER PRIMARY KEY
AUTOINCREMENT,
                    user_id INTEGER,
                    book_id TEXT,
                    issue_date TEXT,
                    return_date TEXT,
                    FOREIGN KEY(user_id) REFERENCES
users(user_id),
                    FOREIGN KEY(book_id) REFERENCES
books(book_id)
                 )''')

conn.commit()
```

```
# Add sample books (if not already present)
def add_sample_books():
    books = [
        ("H1H1", "Harry Potter and the Philosopher's Stone",
"J.K. Rowling"),
        ("H2H2", "Feathers of Fire", "Dr. APJ Abdul Kalam"),
        ("M3M3", "Mahabharatha", "Vedavyasa"),
        ("R4R4", "Srimath Ramayanam", "Valmiki"),
        ("B5B5", "Srimath Bagvatham", "Vedavyasa"),
        ("L6R6", "Lord of the Rings", "J.R.R. Tolkien"),
        ("H3H3", "Harry Potter and the Chamber of Secrets", "J.K.
Rowling")
    ]
    for book_id, title, author in books:
        try:
            cursor.execute("INSERT INTO books (book_id, title,
author, status) VALUES (?, ?, ?, ?)",
                           (book_id, title, author, "available"))
            conn.commit()
        except sqlite3.IntegrityError:
            pass  # Ignore duplicates

add_sample_books()

# GUI Functions
def register_user():
    username = entry_username.get()
    password = entry_password.get()
    try:
        cursor.execute("INSERT INTO users (username, password) 
VALUES (?, ?)", (username, password))
        conn.commit()
        messagebox.showinfo("Success", "User registered 
successfully!")
    except sqlite3.IntegrityError:
        messagebox.showerror("Error", "Username already exists!")


def login_user():
    username = entry_username.get()
    password = entry_password.get()
    cursor.execute("SELECT * FROM users WHERE username = ? AND 
password = ?", (username, password))
    user = cursor.fetchone()
    if user:
        messagebox.showinfo("Success", "Logged in successfully!")
        user_dashboard(user[0])
    else:
        messagebox.showerror("Error", "Invalid username or 
password")


def search_books():
    query = entry_search.get()
    cursor.execute("SELECT * FROM books WHERE title LIKE ? OR 
author LIKE ?", (f"%{query}%", f"%{query}%"))
    results = cursor.fetchall()
    listbox_results.delete(0, END)
    for book in results:
        listbox_results.insert(END, f"{book[1]} by {book[2]} 
({book[0]}) - {book[3]}")
```

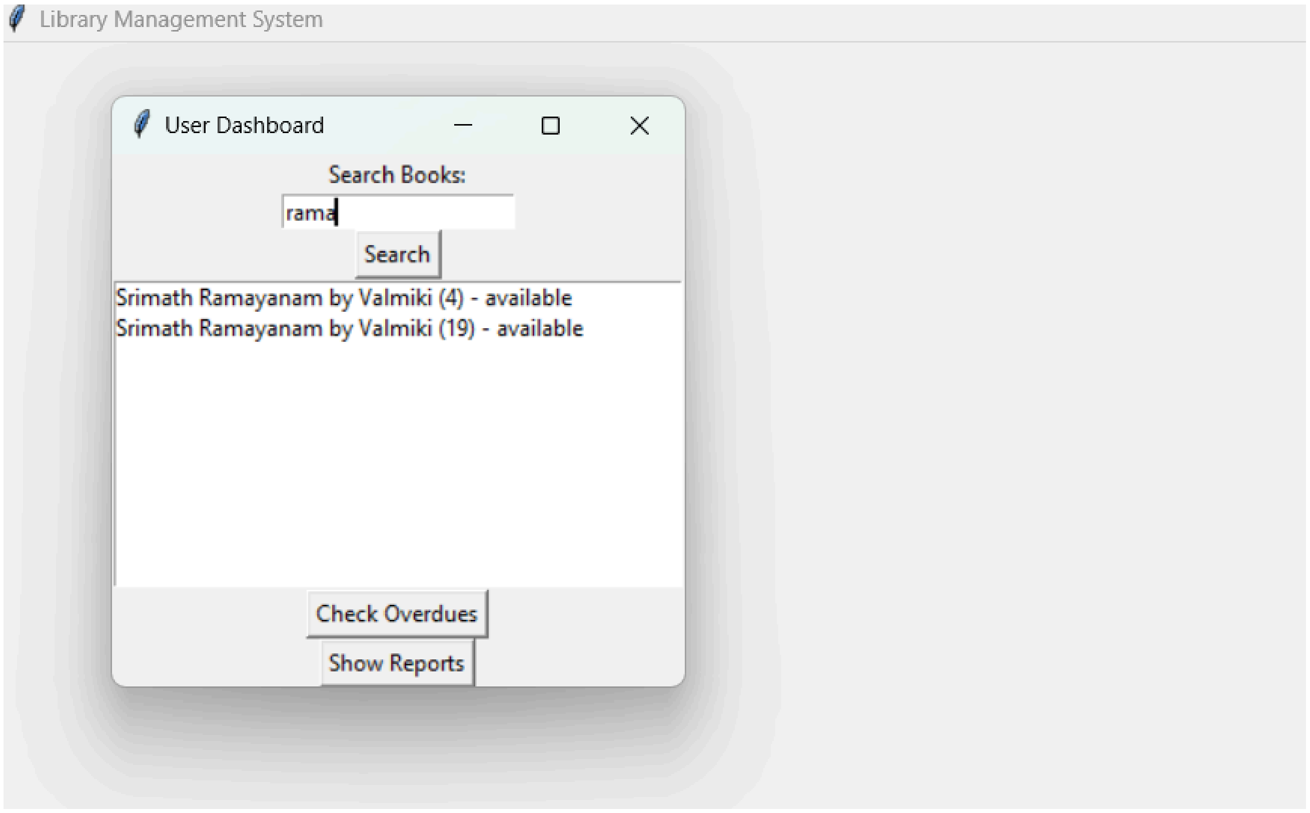
```
# Overdue Notifications
def check_overdues():
    today = datetime.date.today()
    overdue_books = []
    cursor.execute("SELECT t.transaction_id, b.title, 
t.issue_date FROM transactions t "
                   "JOIN books b ON t.book_id = b.book_id "
                   "WHERE t.return_date IS NULL")
    transactions = cursor.fetchall()
    for transaction_id, title, issue_date in transactions:
        issue_date_obj = datetime.datetime.strptime(issue_date, 
'%Y-%m-%d').date()
        if (today - issue_date_obj).days > 14:  # Assuming 14 
days as due period
            overdue_books.append((transaction_id, title))

    if overdue_books:
        messagebox.showwarning("Overdue Books", ", 
".join(f"{title}" for _, title in overdue_books))
    else:
        messagebox.showinfo("Overdue Books", "No overdue books.")

# Reports

def show_reports():
    cursor.execute("SELECT b.title, COUNT(t.transaction_id) AS 
borrow_count FROM books b "
                   "LEFT JOIN transactions t ON b.book_id = 
t.book_id "
                   "GROUP BY b.book_id ORDER BY borrow_count 
DESC")
```

```
    results = cursor.fetchall()
    report_text = "\n".join(f"{title}: {count} times" for title,
count in results)
    messagebox.showinfo("Borrowing Trends", report_text if
report_text else "No borrowing trends available.")

# User Dashboard
def user_dashboard(user_id):
    dashboard = Tk()
    dashboard.title("User Dashboard")

    Label(dashboard, text="Search Books:").pack()
    global entry_search
    entry_search = Entry(dashboard)
    entry_search.pack()

    Button(dashboard, text="Search", command=search_books).pack()

    global listbox_results
    listbox_results = Listbox(dashboard, width=50, height=10)
    listbox_results.pack()

    Button(dashboard, text="Check Overdues",
command=check_overdues).pack()
    Button(dashboard, text="Show Reports",
command=show_reports).pack()

    dashboard.mainloop()

# Main Application
```

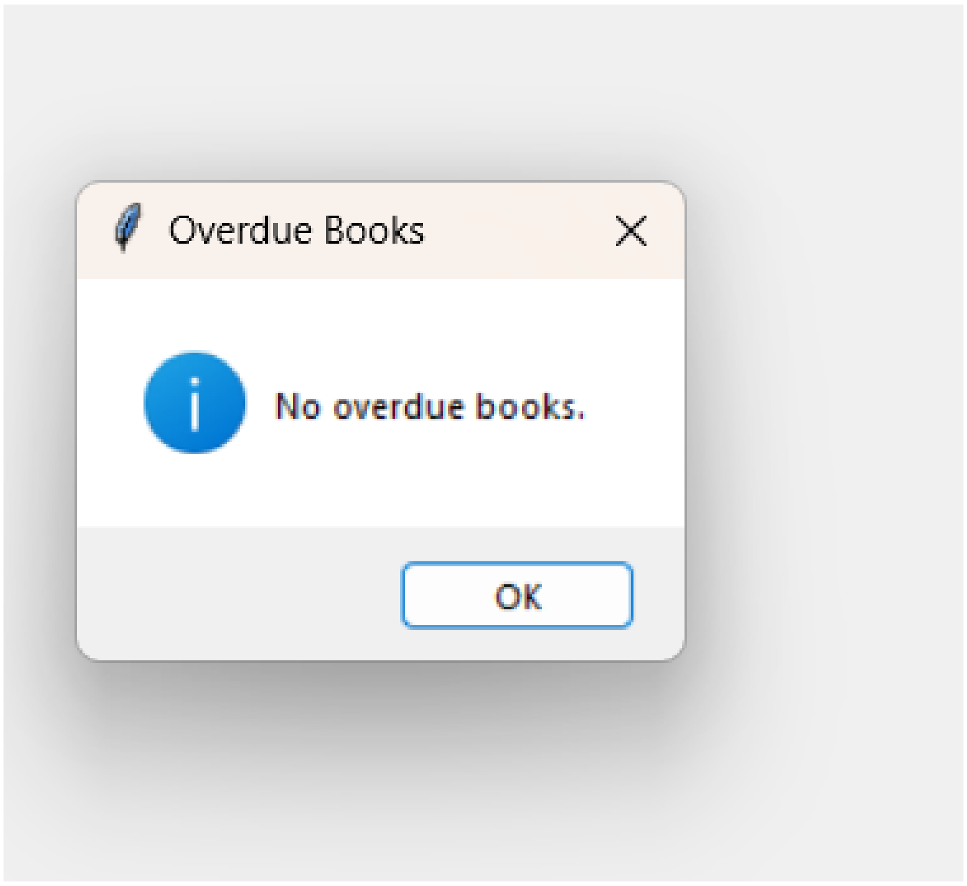
```
app = Tk()
app.title("Library Management System")

Label(app, text="Username:").pack()
entry_username = Entry(app)
entry_username.pack()

Label(app, text="Password:").pack()
entry_password = Entry(app, show="*")
entry_password.pack()

Button(app, text="Register", command=register_user).pack()
Button(app, text="Login", command=login_user).pack()

app.mainloop()
```



```
Harry Potter and the Philosopher's Stone: 0 times
Feathers of Fire: 0 times
Mahabharatha: 0 times
Srimath Ramayanam: 0 times
Srimath Bagvatham: 0 times
Lord of the Rings: 0 times
Sundarakanda: 0 times
Harry Potter and the Chamber of Secrets: 0 times
Harry Potter and the Prisoner of Azkaban: 0 times
Harry Potter and the Goblet of Fire: 0 times
Harry Potter and the Order of the Phoenix: 0 times
Harry Potter and the Half-Blood Prince: 0 times
Harry Potter and the Deathly Hallows: 0 times
harry potter: 0 times
lord of rings: 0 times
Harry Potter and the Philosopher's Stone: 0 times
Feathers of Fire: 0 times
Mahabharatha: 0 times
Srimath Ramayanam: 0 times
Srimath Bagvatham: 0 times
Lord of the Rings: 0 times
Sundarakanda: 0 times
Harry Potter and the Chamber of Secrets: 0 times
Harry Potter and the Prisoner of Azkaban: 0 times
Harry Potter and the Goblet of Fire: 0 times
Harry Potter and the Order of the Phoenix: 0 times
Harry Potter and the Half-Blood Prince: 0 times
Harry Potter and the Deathly Hallows: 0 times
```



Srimath Ramayanam by Valmiki (4) - available
Srimath Ramayanam by Valmiki (19) - available



No overdue books.

Practical Applications:

1. **Small Libraries**: Manage books and users in a simple library without requiring complex systems.
2. **Personal Book Collections**: Track borrowed and returned books from a personal library or collection.
3. **Educational Purpose**: A great project to understand Python, SQLite, and basic database operations.

```
Thank you for using the Library Management System. Goodbye!
```