Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Advanced Sciences

Programme	:	**B.Sc Mathematics and Computing**
Course Title	:	**DATA STRUCTURES**
Course Code	:	**UCSE252P**
Slot	:	**M6+N7+U7**

Title: LIBRARY MANAGEMENT SYSTEM(LMS)

TEAM MEMBERS:

- **S.SANTHOSH 22BMC1009**
- **KEERTHI MENON 22BMC1006**
- **SHINY JASPER NIKKITHA 22BMC1007**

Faculty: Mrs. NATHEZHTHA T

Sign:

Date: 28.09.2023

ABSTRACT:

A library management system is used to keep and monitor library records, including the number of books, information about the books, student membership information, information about book borrowing and book returns, etc. When compared to manual library management systems, it makes book tracking and mistake reduction easier, which aids in appropriate administration and overall library operation. This study describes the transition of traditional libraries to digital ones. In this paper, the code contains the programming language, Python with Tkinter module (to create GUI application), as the front-end to represent the library system to the two users, namely administrator and student. Structured Query Language (SQL) is used as the back-end to store the information in the form of tables in a database. Using the administrator's information, a table is generated that shows the specifics of the books that are currently on hand at the library. Additionally, there are tables with details on the books that each student has checked out from the library so that the administration can keep track of it. LMS is made to be user-friendly so the administrator may easily activate the system without professional assistance. Since all data is stored in and retrieved from a SQL database, it is extremely safe. As a result, our solution offers a fresh perspective on how to build up a digital library. Additionally, the paper incorporates a bar graph indicating the total number of books and the books that are now available for this paper, and it also allows students the opportunity to write a book review.

INTRODUCTION:

A library is a collection of books, as well as perhaps other resources and media, that is available for use by both its members and members of affiliated institutions.Through books, novels, and encyclopaedias, libraries provide information and amusement. It is a location for lifelong learning and has innumerable more resources that would be expensive or impossible to get elsewhere.Libraries offer knowledge and entertainment through books, novels, and encyclopedia. It is a place for lifelong learning and contains countless other resources which would be otherwise difficult to find or afford.

An integrated library system (ILS), also called a library management system (LMS), is an enterprise resource planning (ERP) tool for libraries that keeps track of the materials possessed, orders placed, bills paid, and borrowers. Prior to computerization,each library task was completed physically and independently. Manually cataloging and indexing sources was done by catalogers using the card catalogue system, which retained all bibliographic data on a single index card.

The local bailiffs were responsible for collecting fines. Using clue cards, which were kept at the circulation desk, users manually checked out books by writing their names on the cards.When the University of Texas started employing a punch card system to control library circulation in 1936, the first signs of mechanization were made. While the punch card system made loan monitoring more effective, other library tasks were unaffected by this change, and library services were still far from being integrated. The next big innovation came with the advent of MARC standards in the 1960s, which coincided with the growth of computer technologies – library automation was born[1]. The 1970s were characterised by advancements in telecommunications and computer storage[2].

These developments led to the emergence of integrated library management systems (ILS), sometimes known as "turnkey systems on microcomputers" [3]. Through Online Public Access Catalog (OPAC is an online bibliography of a library collection that is available to the public) and other online web-based portals, Integrated Library System (ILS) started to enable users to engage more actively with their libraries as a result of the development of the Internet throughout the 1990s and into the 2000s. Users could sign into their library accounts to renew or reserve books and authenticate themselves for access to online databases that the library had registered to. The admin is able to observe the quantity of books accessible by means of the application of the bar graph. Similarly, student reviews contribute to the selection of books for students. Furthermore, our paper incorporates Graphical-User Interface viewing and submitting reviews are new capabilities for students, while book availability visualization is a new tool for administrators. Using tkinter module in python to create a better user-friendly interface for users to interact with LMS. Use of buttons and labels to interact with users efficiently and create GUI application. In order to acquire improved comprehension of the book, students can select to read or borrow it based on the opinions and reviews provided by other students. For the purpose of helping develop book reviews that will inspire students to read the books, students can share their thoughts and ideas about the books they have read and evaluate the books. After selecting the SUBMIT option, students submit their reviews. Book availability visualization is done using a bar graph.Y-axis represents the number of books and X-axis represents borrowed books and available books. This graph facilitates keeping track of books and provides a visual representation of the library's collection.

Key Features:

- **User Types:**

Admin: Administrators have access to features such as adding books to the library, printing the list of available books, removing books, and adding new students.

Student: Students can search for books, borrow books, return books, visualize book availability through graphs, submit reviews, and view book reviews.

- **Book and Library Classes:**

The project employs object-oriented programming with Python, defining two key classes: Book and Library.
The Book class represents individual books in the library, storing information such as the book's name, author, and ISBN.
The Library class manages collections of books and student records. It includes methods to add books, remove books, list books, search for books, add students, issue books to students, and return books.

- **Graphical User Interface (GUI):**

The GUI is created using tkinter, a Python library for building desktop applications with graphical elements.
It provides a user-friendly interface for both administrators and students to interact with the system.
Database Integration:

The project uses a MySQL database to store and manage data.
SQL queries are executed to create tables for book records and student details, insert data, delete records, and select information.
The database allows for persistent storage of library resources and student information.

Functionalities:

- **Admin Functionalities:**

Adding Books: Admins can add new books to the library by providing book details such as name, author, and ISBN.
Printing Books: Admins can view and print the list of available books in the library.
Removing Books: Admins can remove books from the library by specifying the book name.
Adding Students: Admins can add new students to the system, providing student names, registration numbers, and passwords.

- **Student Functionalities:**

Searching for Books: Students can search for books by entering the book's name and check if it's available.
Borrowing and Returning Books: Students can borrow books from the library and return them.
Visualizing Book Availability: Students can view graphical representations of book availability in the library.
Submitting Reviews: Students can submit reviews for books they've read, including a text review and a rating.
Viewing Reviews: Students can view reviews submitted by others for specific books.

- **User Authentication:**

The system authenticates users based on their user type (Admin or Student) and password.

METHODOLOGY:

1. Database Setup:

Set up a MySQL database connection using module(Mysql.connector) in python with the following specifications:
Host: localhost
User: root
Port: 3136
Password: SMDS
Database: library

2. Class Definitions:

Define two classes, Book and Library, to organize and manage data in the from of arrays:
Book class represents individual books.
Library class manages books and student records.

3. Library Functions:

Create functions within the Library class to perform library operations:

add_book: Add books to the library.
remove_book: Remove books from the library and the database.
list_books: List all available books in the database.
search_book: Search for books in the library by name.
add_student: Add student records to the library.
issue_book: Allow students to borrow books and update records.
return_book: Enable students to return books and update records.

4. GUI Development:

Develop a graphical user interface (GUI) using Tkinter for both administrators and students.

5. Admin Login:

Implement admin login with the credentials "SKS" and "DSBMC" for authentication.

6. Admin Functionalities:

Admins can perform the following tasks:
Add books to the library and the database.
Print the list of books in the library.

Remove books from the library and the database.
Add students to the library's records.

7. Student Login:

Implement student login using registration numbers and passwords.

8. Student Functionalities:

Students can use the following functionalities:
Search for books in the library.
Borrow books and update their records.
Return books and update their records.

9. Database Interaction:

Ensure proper interaction between the GUI and the MySQL database by executing SQL queries for data retrieval, updates, and deletions.

10. User Experience:
Design an intuitive GUI, validate user inputs, and provide clear feedback after each action to enhance the user experience.

Benefits:

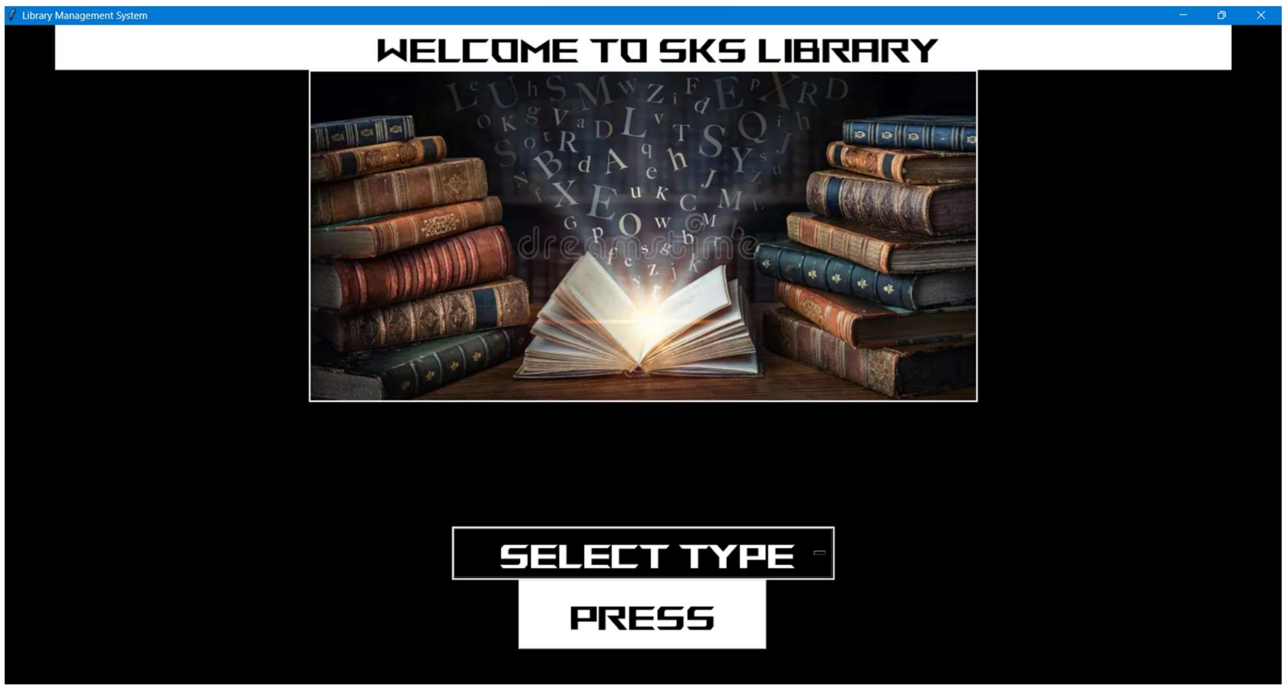
Efficient Management: The system streamlines library operations, making it easier for administrators to manage books and students.
Accessibility: Students can quickly check book availability and submit reviews.
Data Persistence: All book records and student information are stored in a MySQL database, ensuring data integrity.

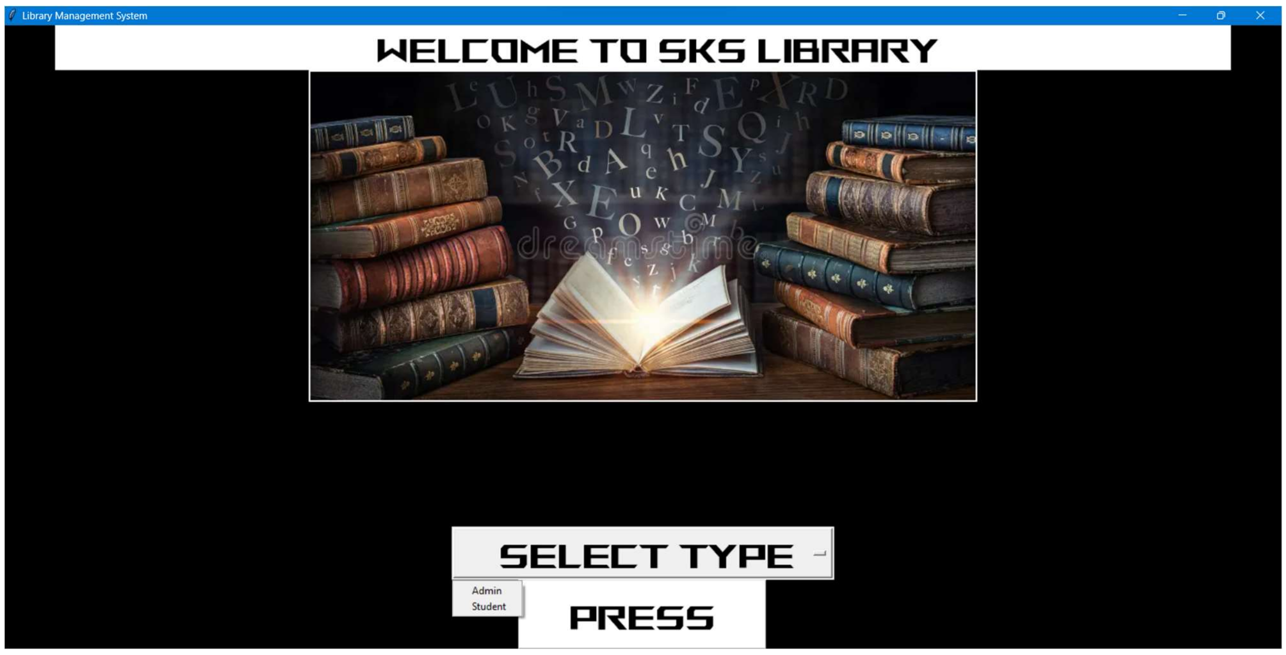
CODE:

```
import mysql.connector
import tkinter as tk
from tkinter import messagebox
from tkinter import *
import matplotlib.pyplot as plt

mydb=mysql.connector.connect(host="localhost",
user="root", password='SMDS',port='3136',
database='library', autocommit=True)
mc=mydb.cursor()

class Book:
    def __init__(self, name, author, isbn):
        self.name = name
        self.author = author
        self.isbn = isbn

class Library:
    def __init__(self):
        self.books = []
        self.student_records = []

    def add_book(self, book):
        self.books.append(book)

    def remove_book(self, book_name):
        for book in self.books:
            if book.name == book_name:
                self.books.remove(book)
                return True
        return False

    def remove_books(self, book_name):
        for book in self.books:
            if book.name == book_name:
                self.books.remove(book)

    def list_books(self):
        for book in self.books:
            print(book.name, book.author, book.isbn)

    def search_book(self, book_name):
        for book in self.books:
            if book.name == book_name:
                return book
        return None

    def add_student(self, student_name, student_id,
passw):
        self.student_records.append({
            "student_name": student_name,
            "student_id": student_id,
            "student_pwd": passw
        })

    def issue_book(self, student_id, book_name):
        for student_record in self.student_records:
            if student_record["student_id"] ==
student_id:
                for book in self.books:
                    if book.name == book_name:
                        if book not in
student_record["books_issued"]:

student_record["books_issued"].append(book)
                            self.books.remove(book)
                            return True
        return False

    def return_book(self, student_id, book_name):
        for student_record in self.student_records:
            if student_record["student_id"] ==
student_id:
                for book in
student_record["books_issued"]:
                    if book.name == book_name:

student_record["books_issued"].remove(book)
                        self.books.append(book)
                        return True
        return False

library = Library()
J = 0

def add_book_window():
    add_book_window = tk.Toplevel(root)
    add_book_window.title("Add Book - SKS LIBRARY")
    add_book_window.geometry("1920x1080")
    add_book_window.configure(bg="black")

    library_label = tk.Label(add_book_window, text="
SKS LIBRARY                             ", font=("ROG
FONTS", 30),bg="black",fg="white")
    library_label.pack()

    def add_book():
        bname = book_name_entry.get()
        aname = author_name_entry.get()
        bid = book_id_entry.get()
        mc.execute(
            "CREATE TABLE IF NOT EXISTS
add_book(BOOK_NAME varchar(20), AUTHORS_NAME
varchar(20), BOOK_ID varchar(20))")
        mc.execute("INSERT INTO add_book
VALUES('{}','{}','{}');".format(bname, aname, bid))
        library.add_book(Book(bname, aname, bid))
        messagebox.showinfo("Success", "Book added to
the library.")
        add_book_window.destroy()

    book_name_label = tk.Label(add_book_window,
text="Book Name:",bg="black",fg="white")
    book_name_label.pack()
    book_name_entry = tk.Entry(add_book_window)
    book_name_entry.pack()

    author_name_label = tk.Label(add_book_window,
text="Author Name:",bg="black",fg="white")
    author_name_label.pack()
    author_name_entry = tk.Entry(add_book_window)
    author_name_entry.pack()

    book_id_label = tk.Label(add_book_window, text="Book
ID:",bg="black",fg="white")
    book_id_label.pack()
    book_id_entry = tk.Entry(add_book_window)
    book_id_entry.pack()

    add_button = tk.Button(add_book_window, text="Add
Book",bg="black",fg="white", command=add_book)
    add_button.pack()

def print_books():
    mc.execute("SELECT * FROM add_book")
    data = mc.fetchall()


    print_books_window = tk.Toplevel(root)
    print_books_window.title("Print Books - SKS
LIBRARY")
    print_books_window.geometry("1920x1080")
    print_books_window.configure(bg="black")
    library_label = tk.Label(print_books_window, text="
SKS LIBRARY                             ", font=("ROG
FONTS", 30),bg="black",fg="white")
    library_label.pack()

    books_listbox =
tk.Listbox(print_books_window,bg="black",fg="white",font
=("TIMES NEW ROMAN", 16))
    books_listbox.pack(fill=tk.BOTH, expand=True)

    for book in data:
        books_listbox.insert(tk.END, book)

def remove_book_window():
    remove_book_window = tk.Toplevel(root)
    remove_book_window.title("Remove Book - SKS
LIBRARY")
    remove_book_window.geometry("1920x1080")
    remove_book_window.configure(bg="black")

    library_label = tk.Label(remove_book_window, text="
SKS LIBRARY                             ", font=("ROG
FONTS", 30),bg="black",fg="white")
    library_label.pack()

    def remove_book():
        bname = book_name_entry.get()

        mc.execute("DELETE FROM add_book WHERE BOOK_NAME
= ('{}');".format(bname))
        if library.remove_book(bname):
            messagebox.showinfo("Success", f"{bname} has
been removed from the library.")
        else:
            messagebox.showinfo("Error", f"{bname} not
found in the library.")
        remove_book_window.destroy()

    book_name_label = tk.Label(remove_book_window,
text="Book Name:",bg="black",fg="white")
    book_name_label.pack()
    book_name_entry = tk.Entry(remove_book_window)
    book_name_entry.pack()
    remove_button = tk.Button(remove_book_window,
text="Remove Book",bg="black",fg="white",
command=remove_book)
    remove_button.pack()

def add_student_window():
    add_student_window = tk.Toplevel(root)
    add_student_window.title("Add Student - SKS
LIBRARY")
    add_student_window.geometry("1920x1080")
    add_student_window.configure(bg="black")

    library_label = tk.Label(add_student_window, text="
SKS LIBRARY                             ", font=("ROG
FONTS", 30),bg="black",fg="white")
    library_label.pack()

    def add_student():
        sname = student_name_entry.get()
        reg_no = reg_no_entry.get()
        pwd = password_entry.get()

        mc.execute(
            "CREATE TABLE IF NOT EXISTS
student_details(STUDENT_NAME varchar(20),
REGISTRATION_NUM varchar(20), PWD varchar(20))")
        mc.execute("INSERT INTO student_details
VALUES('{}','{}','{}');".format(sname, reg_no, pwd))
        mc.execute("create table if not exists
"+reg_no+"(BOOK_NAME varchar(20))")
        library.add_student(sname, reg_no, pwd)
        messagebox.showinfo("Success", "Student added.")
        add_student_window.destroy()
    head_label = tk.Label(add_student_window,
text="\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n",bg="black",fg="whi
te")
    head_label.pack()
    student_name_label = tk.Label(add_student_window,
text="Student Name:",bg="black",fg="white")
    student_name_label.pack()
    student_name_entry = tk.Entry(add_student_window)
    student_name_entry.pack()
    reg_no_label = tk.Label(add_student_window,
text="Registration Number:",bg="black",fg="white")
    reg_no_label.pack()
    reg_no_entry = tk.Entry(add_student_window)
    reg_no_entry.pack()

    password_label = tk.Label(add_student_window,
text="Password:",bg="black",fg="white")
    password_label.pack()
    password_entry = tk.Entry(add_student_window,
show="*")
    password_entry.pack()

    add_button = tk.Button(add_student_window, text="Add
Student",bg="black",fg="white",command=add_student)
    add_button.pack()
def visualize_book_availability_admin():
    mc.execute("SELECT COUNT(*) FROM add_book")
    available_books = mc.fetchone()[0]

    mc.execute("SELECT REGISTRATION_NUM FROM
student_details")
    student_usernames = [row[0] for row in
mc.fetchall()]

    borrowed_books = 0
    for username in student_usernames:
        mc.execute("SELECT COUNT(*) FROM {}"
.format(username))
        borrowed_books += mc.fetchone()[0]+1

    labels = ['Available Books', 'Borrowed Books']
    values = [available_books, borrowed_books]

    plt.bar(labels, values)
    plt.xlabel('Book Status')
    plt.ylabel('Number of Books')
    plt.title('Book Availability')
    plt.show()


def admin_menu():
    root.withdraw()
    admin_login_window = tk.Toplevel(root)
    admin_login_window.title("Admin Login")
    admin_login_window.geometry("1920x1080")
    admin_login_frame = tk.Frame(admin_login_window,
bg="black")
    admin_login_frame.pack(expand=True, fill="both")

    label1 = tk.Label(admin_login_frame, text="  SKS
Library Portal \n", font=("ROG FONTS", 51),fg="white",
bg="black")
    label1.pack()


    admin_user_label = tk.Label(admin_login_frame,
text="\n\n", bg="black",font=("Helvetica",30))
    admin_user_label.pack()

    admin_username_label = tk.Label(admin_login_frame,
text="Username:",
bg="black",fg="white",font=("Helvetica",30))
    admin_username_label.pack()
    admin_username_entry = tk.Entry(admin_login_frame,
width=50, font=('Arial 24'))
    admin_username_entry.pack()

    admin_password_label = tk.Label(admin_login_frame,
text="Password:",
bg="black",fg="white",font=("Helvetica",30))
    admin_password_label.pack()
    admin_password_entry = tk.Entry(admin_login_frame,
width=50, font=('Arial 24'), show="*")
    admin_password_entry.pack()

    admin_login_button = tk.Button(admin_login_frame,
text="Login",
command=lambda:admin_login(admin_username_entry,admin_pa
ssword_entry),
bg="black",fg="white",font=("Helvetica",30))
    admin_login_button.pack()

def
admin_login(admin_username_entry,admin_password_entry):
    username = admin_username_entry.get()
    password = admin_password_entry.get()
    if username == "SKS" and password == "DSBMC":
        admin_menu_window = tk.Toplevel(root)
        admin_menu_window.geometry("1920x1080")
        admin_menu_window.title("Admin Menu - SKS
LIBRARY")
        admin_menu_window.configure(bg="black")
        library_label = tk.Label(admin_menu_window,
text="                                SKS LIBRARY
", font=("ROG FONTS", 30),bg="BLACK",fg="white")
        library_label.pack()

        head_label = tk.Label(admin_menu_window,
text="\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n",bg="black")
        head_label.pack()

        add_book_button = tk.Button(admin_menu_window,
text="1.Add Books",bg="black",fg="white",
command=add_book_window, font=("BOOKMAN OLD STYLE", 30))
        add_book_button.pack()

        print_books_button =
tk.Button(admin_menu_window, text="2.Print Books In The
Library",bg="black",fg="white", command=print_books,
font=("BOOKMAN OLD STYLE", 30))
        print_books_button.pack()

        remove_book_button =
tk.Button(admin_menu_window, text="3.Remove
Books",bg="black",fg="white",
command=remove_book_window, font=("BOOKMAN OLD STYLE",
30))
        remove_book_button.pack()

        add_student_button =
tk.Button(admin_menu_window, text="4.Add
Student",bg="black",fg="white",
command=add_student_window, font=("BOOKMAN OLD STYLE",
30))
        add_student_button.pack()

        visualize_availability_button =
tk.Button(admin_menu_window, text="5.Visualize Book
Availability",bg="black",fg="white",
command=visualize_book_availability_admin,
font=("BOOKMAN OLD STYLE", 30))
        visualize_availability_button.pack()

def student_login_window():
    student_login_window = tk.Toplevel(root)
    student_login_window.title("Student Login")
    student_login_window.geometry("1920x1080")
    student_login_window.configure(bg="black")
    library_label = tk.Label(student_login_window,
text="                                SKS LIBRARY
", font=("ROG FONTS", 30),bg="white",fg="black")
    library_label.pack()

    student_user_label = tk.Label(student_login_window,
text="\n\n\n", bg="black",font=("Helvetica",30))
    student_user_label.pack()

    student_username_label =
tk.Label(student_login_window, text="Username
(Registration Number):",
bg="black",fg="white",font=("Helvetica",30))
    student_username_label.pack()
    student_username_entry =
tk.Entry(student_login_window, width=50, font=('Arial
24'))
    student_username_entry.pack()

    student_password_label =
tk.Label(student_login_window, text="Password:",
bg="black",fg="white",font=("Helvetica",30))
    student_password_label.pack()
    student_password_entry =
tk.Entry(student_login_window, width=50, font=('Arial
24'), show="*")
    student_password_entry.pack()

    student_login_button =
tk.Button(student_login_window, text="Login",
command=lambda: student_login(student_username_entry,
student_password_entry),fg="white", bg="black")
    student_login_button.pack()
```

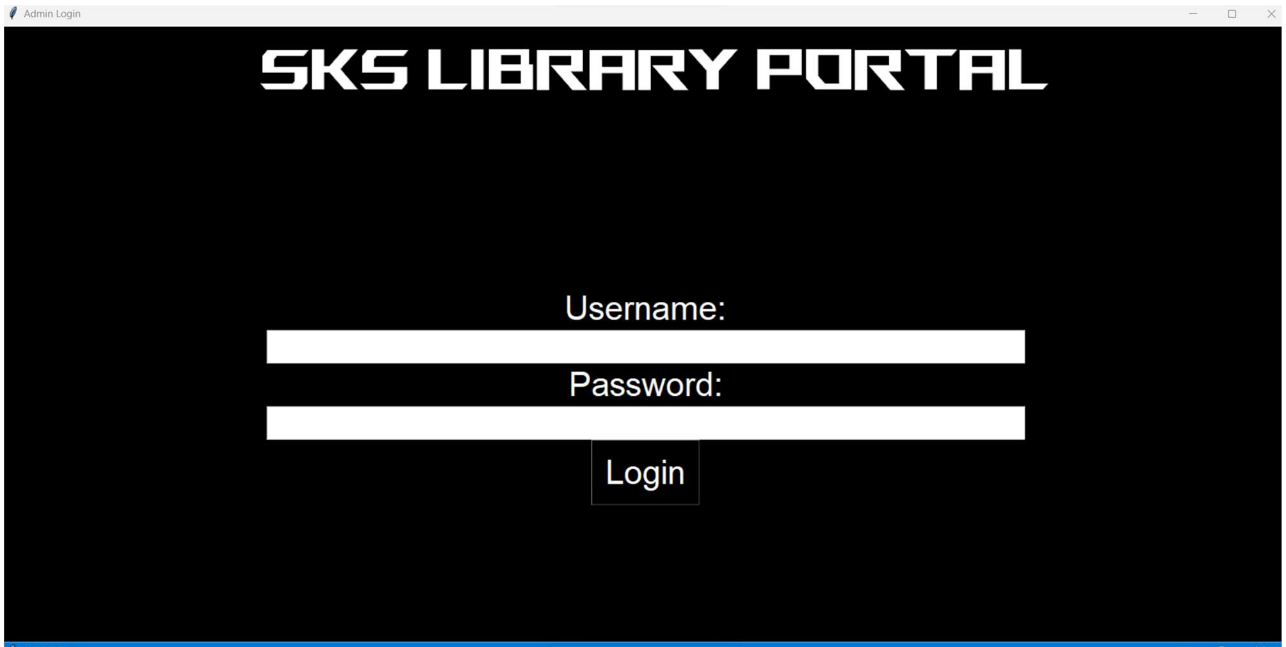
```
def student_login(student_username_entry,
student_password_entry):
    username = student_username_entry.get()
    password = student_password_entry.get()

    mc.execute("SELECT PWD FROM student_details WHERE
REGISTRATION_NUM = %s", (username,))
    stored_password = mc.fetchone()

    if stored_password and password ==
stored_password[0]:
        messagebox.showinfo("Login Successful",
"Welcome, Student!")
        student_menu(username)
    else:
        messagebox.showerror("Login Error", "Invalid
username or password.")
def submit_review(username):
    def submit():
        book_name = book_name_entry.get()
        review_text = review_text_entry.get()
        rating = rating_entry.get()

        if not book_name or not review_text or not
rating:
            messagebox.showerror("Error", "Please fill
in all fields.")
        else:
            mc.execute("CREATE TABLE IF NOT EXISTS
book_reviews(BOOK_NAME varchar(20),USERNAME
varchar(20),REVIEW_TEXT varchar(20),RATING INT(10))")
            mc.execute("INSERT INTO book_reviews
(book_name, username, review_text, rating) VALUES (%s,
%s, %s, %s)",
                       (book_name, username,
review_text, rating))
            mydb.commit()
            messagebox.showinfo("Success", "Review
submitted successfully.")
            review_window.destroy()

    review_window = tk.Toplevel(root)
    review_window.title("Submit Review - SKS LIBRARY")
    review_window.geometry("400x300")
    review_window.configure(bg="black")
    book_name_label = tk.Label(review_window, text="Book
Name:",bg="black",fg="white")
    book_name_label.pack()
    book_name_entry = tk.Entry(review_window)
    book_name_entry.pack()

    review_text_label = tk.Label(review_window,
text="Review Text:",bg="black",fg="white")
    review_text_label.pack()
    review_text_entry = tk.Entry(review_window)
    review_text_entry.pack()

    rating_label = tk.Label(review_window,
text="Rating:",bg="black",fg="white")
    rating_label.pack()
    rating_entry = tk.Entry(review_window)
    rating_entry.pack()

    submit_button = tk.Button(review_window,
text="Submit", command=submit)
    submit_button.pack()
def view_reviews(username):
    view_reviews_window = tk.Toplevel(root)
    view_reviews_window.title("View Reviews - SKS
LIBRARY")
    view_reviews_window.geometry("400x300")
    view_reviews_window.configure(bg="black")
    book_name_label = tk.Label(view_reviews_window,
text="Book Name:",bg="black",fg="white")
    book_name_label.pack()
    book_name_entry = tk.Entry(view_reviews_window)
    book_name_entry.pack()

    def show_reviews():
        book_name = book_name_entry.get()
        if not book_name:
            messagebox.showerror("Error", "Please enter
a book name.")
        else:
            mc.execute("SELECT review_text, rating FROM
book_reviews WHERE book_name = %s", (book_name,))
            reviews = mc.fetchall()
            if not reviews:
                messagebox.showinfo("No Reviews", "No
reviews found for this book.")
            else:
                review_text = "\n\n".join([f"Review:
{review[0]}\nRating: {review[1]}" for review in
reviews])
                messagebox.showinfo("Reviews for " +
book_name, review_text)

    show_button = tk.Button(view_reviews_window,
text="Show Reviews",
command=show_reviews,bg="black",fg="white")
    show_button.pack()

def student_menu(username):
    student_menu_window = tk.Toplevel(root)
    student_menu_window.title("Student Menu - SKS
LIBRARY")
    student_menu_window.configure(bg="black")
    student_menu_window.geometry("1920x1080")
    library_label = tk.Label(student_menu_window, text="
SKS LIBRARY                             ", font=("ROG
FONTS", 30),bg="black",fg="white")
    library_label.pack()

    #search_books_button =
tk.Button(student_menu_window, text="Search Books",
command=lambda:search_books)
    #search_books_button.pack()

    def search_books():
        search_label = tk.Label(student_menu_window,
text="Enter the book name to search:",
bg="black",fg="white", font=("BOOKMAN OLD STYLE", 30))
        search_label.pack()
        search_book_entry =
tk.Entry(student_menu_window)
        search_book_entry.pack()

        search_button1 = tk.Button(student_menu_window,
text="Search", command=lambda:
search_book(search_book_entry,
result_label),bg="black",fg="white", font=("BOOKMAN OLD
STYLE", 30))
        search_button1.pack()

        result_label = tk.Label(student_menu_window,
text="", bg="black",fg="white")
        result_label.pack()

    def search_book(search_book_entry, result_label):
        sb = search_book_entry.get()
        mc.execute("SELECT BOOK_NAME FROM add_book")
        sc1 = mc.fetchall()
        for sc2 in sc1:
            if sb in sc2:
                result_label.config(text="Yes, the book
is available.", font=("BOOKMAN OLD STYLE", 30))
                return
        result_label.config(text="No, the book is not
available.", font=("BOOKMAN OLD STYLE", 30))
    #search_label = tk.Label(student_menu_window,
text="Enter the book name to search:",bg="black",
font=("BOOKMAN OLD STYLE", 30))
    #search_label.pack()
    #search_book_entry = tk.Entry(student_menu_window)
    #search_book_entry.pack()

    def get_book_window():
        book1_name_label = tk.Label(student_menu_window,
text="Book Name:", bg="black",fg="white")
        book1_name_label.pack()
        book1_name_entry = tk.Entry(student_menu_window)
        book1_name_entry.pack()

        search_button2 = tk.Button(student_menu_window,
text="Borrow",
command=lambda:get_book(book1_name_entry,remove_button),
bg="black",fg="white", font=("BOOKMAN OLD STYLE", 30))
        search_button2.pack()

        remove_button = tk.Label(student_menu_window,
text="", bg="black",fg="white")
        remove_button.pack()

    def get_book(book1_name_entry,remove_button):
        bname = book1_name_entry.get()

        mc.execute("DELETE FROM add_book WHERE BOOK_NAME
= ('{}');".format(bname))
        mc.execute("insert into "+username+"
values('{}');".format(bname))
        if library.remove_book(bname):
            remove_button.config(text="Error the book
hasn't been received by the student.", font=("BOOKMAN
OLD STYLE", 30))
            return
            #messagebox.showinfo("Success", f"{bname}
has been received by the student.")
        #else:
        remove_button.config(text="Success Book has been
Issued To The Student.", font=("BOOKMAN OLD STYLE", 30))
            #messagebox.showinfo("Error", f"{bname} not
found in the library.")



    def return_book_window():
        book_name1_label = tk.Label(student_menu_window,
text="Book Name:",bg="black",fg="white")
        book_name1_label.pack()
        book_name1_entry = tk.Entry(student_menu_window)
        book_name1_entry.pack()

        author1_name_label =
tk.Label(student_menu_window, text="Author
Name:",fg="white",bg="black")
        author1_name_label.pack()
        author1_name_entry =
tk.Entry(student_menu_window)
        author1_name_entry.pack()

        book1_id_label = tk.Label(student_menu_window,
text="Book ID:",fg="white",bg="black")
        book1_id_label.pack()
        book1_id_entry = tk.Entry(student_menu_window)
        book1_id_entry.pack()
        add_button1 = tk.Button(student_menu_window,
text="Add Book",bg="black",fg="white",
command=lambda:return_book(book_name1_entry,author1_name
_entry,book1_id_entry))
        add_button1.pack()
    def
return_book(book_name1_entry,author1_name_entry,book1_id
_entry):
        bname = book_name1_entry.get()
        aname = author1_name_entry.get()
        bid = book1_id_entry.get()
        mc.execute("INSERT INTO add_book
VALUES('{}','{}','{}');".format(bname, aname, bid))
        mc.execute("DELETE FROM "+username+" WHERE
BOOK_NAME = ('{}');".format(bname))
        library.add_book(Book(bname, aname, bid))
        messagebox.showinfo("Success", "Book returned to
the library.")

    '''def visualize_book_availability():
        mc.execute("SELECT COUNT(*) FROM add_book")
        available_books = mc.fetchone()[0]

        mc.execute("SELECT COUNT(*) FROM {}"
.format(username))
        borrowed_books = mc.fetchone()[0]

        labels = ['Available Books', 'Borrowed Books']
        values = [available_books, borrowed_books]

        plt.bar(labels, values)
        plt.xlabel('Book Status')
        plt.ylabel('Number of Books')
        plt.title('Book Availability')
        plt.show()'''


    def submit_review_window():
        submit_review(username)

    def view_reviews_window():
        view_reviews(username)
    search_button = tk.Button(student_menu_window, 
text="1.Search Books",bg="black",fg="white", 
command=search_books, font=("BOOKMAN OLD STYLE", 30))
    search_button.pack()

    print_books_button = tk.Button(student_menu_window, 
text="2.Print 
Books",bg="black",fg="white",command=print_books, 
font=("BOOKMAN OLD STYLE", 30))
    print_books_button.pack()

    get_button = tk.Button(student_menu_window, 
text="3.To Borrow Books",bg="black",fg="white", 
command=get_book_window, font=("BOOKMAN OLD STYLE", 30))
    get_button.pack()

    return_button = tk.Button(student_menu_window, 
text="4.To Return Books",bg="black",fg="white", 
command=return_book_window, font=("BOOKMAN OLD STYLE", 
30))
    return_button.pack()

    #visualize_button = tk.Button(student_menu_window, 
text="5.Graphical History",bg="black", 
command=visualize_book_availability, font=("BOOKMAN OLD 
STYLE", 30))
    #visualize_button.pack()

    submit_review_button = 
tk.Button(student_menu_window, text="6.Submit 
Review",bg="black",fg="white", 
command=submit_review_window, font=("BOOKMAN OLD STYLE", 
30))
    submit_review_button.pack()

    view_reviews_button = tk.Button(student_menu_window, 
text="7.View Reviews",bg="black",fg="white", 
command=view_reviews_window, font=("BOOKMAN OLD STYLE", 
30))
    view_reviews_button.pack()

def select_user_type():
    global J
    user_type = user_type_var.get()
    if user_type == "Admin":
        J = 1
        admin_menu()
    elif user_type == "Student":
        student_login_window()
    else:
        messagebox.showerror("User Type Error", "Invalid
user type selected.")

root = tk.Tk()
root.title("Library Management System")
root.configure(bg="black")
library_label = tk.Label(root, text="
WELCOME TO SKS LIBRARY                             ",
font=("ROG FONTS", 30),bg="white",fg="black")
library_label.pack()
bg = PhotoImage(file = "D:\old-book-flying-letters-
magic-light-background-bookshelf-library-ancient-books-
as-symbol-knowledge-history-218640948.png")
root.geometry("1920x1080")
library_label = tk.Label(root, text="", font=("ROG
FONTS", 30),bg="white",fg="white",image=bg)
library_label.pack()

user_type_label = tk.Label(root, text=" \nSelect User
Type:\n",bg="black",font=("ROG FONTS", 30))
user_type_label.pack()

user_type_var = tk.StringVar(root)
user_type_var.set("    Select Type ")
user_type_optionmenu = tk.OptionMenu(root,
user_type_var, "Admin", "Student")
user_type_optionmenu.config(bg="black",
fg="WHITE",font=("ROG FONTS", 30))
user_type_optionmenu.pack()

select_user_type_button = tk.Button(root, text="   PRESS
",bg="white", font=("ROG FONTS", 30),
command=select_user_type)
select_user_type_button.pack()

root.mainloop()
```

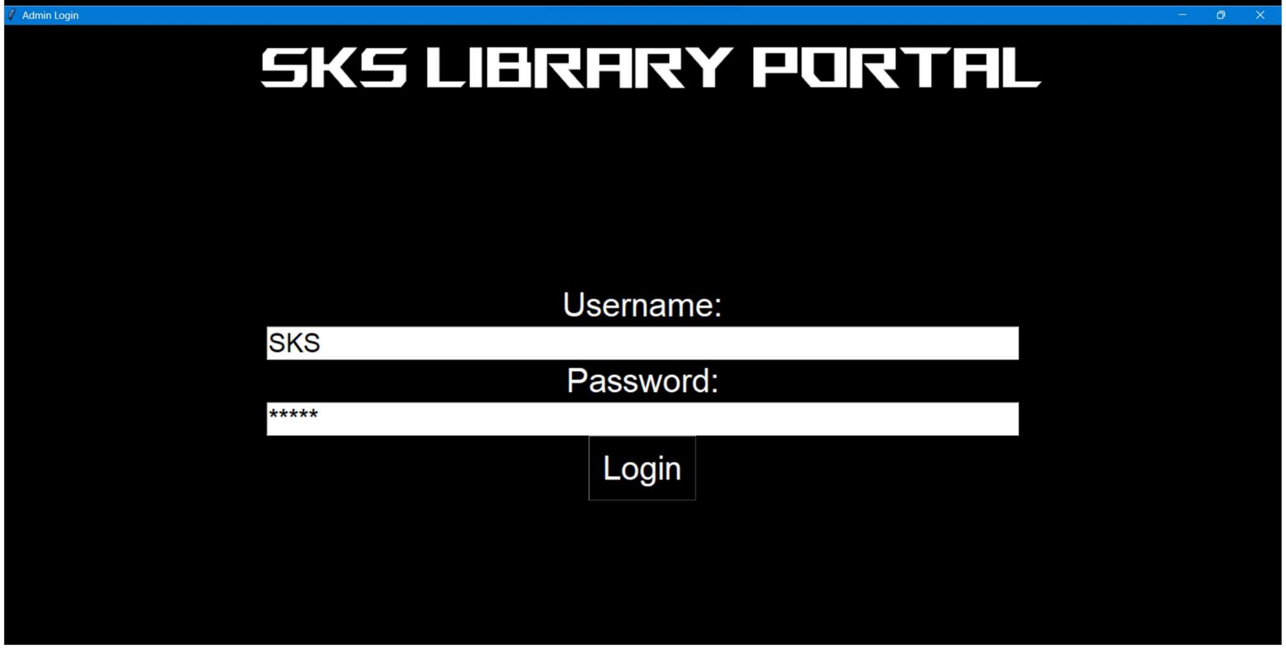
OUTPUT:

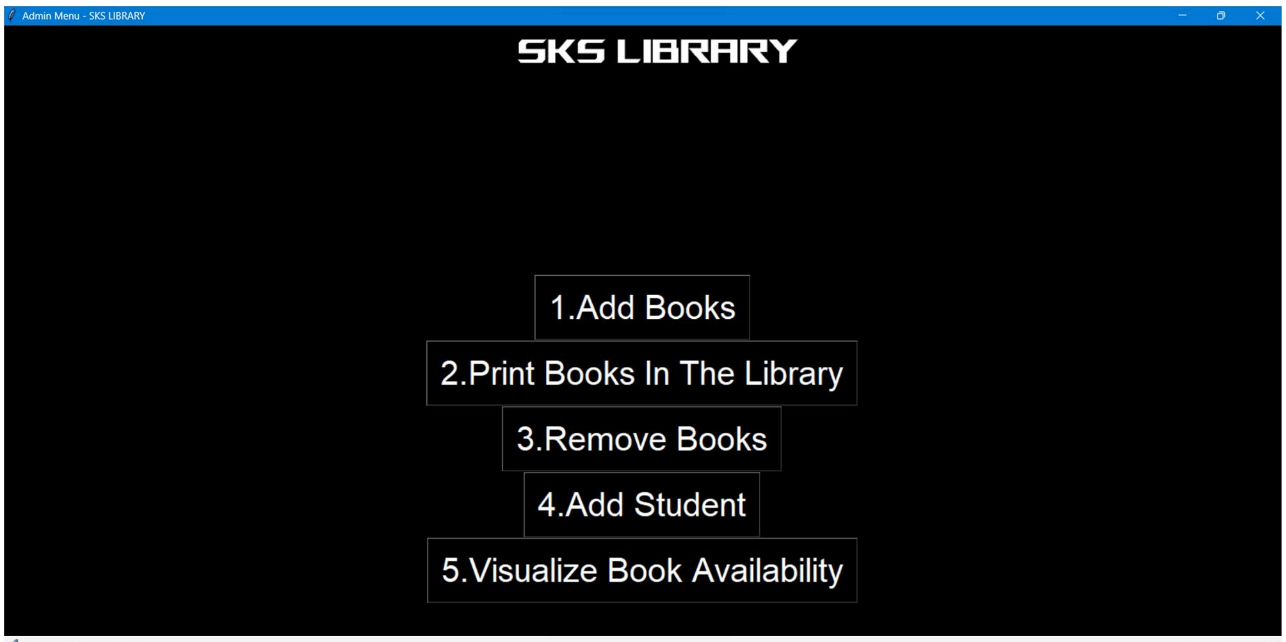

Library Management System
WELCOME TO SKS LIBRARY
SELECT TYPE
PRESS

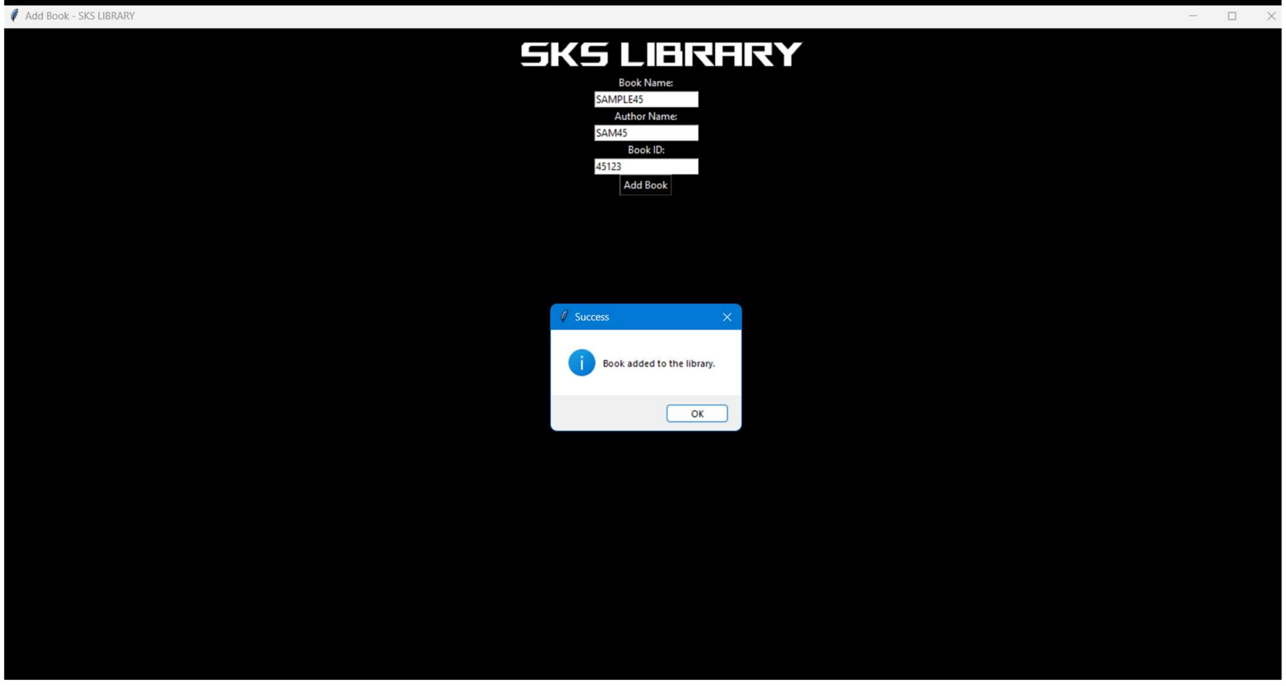

Library Management System
WELCOME TO SKS LIBRARY
SELECT TYPE
Admin
Student
PRESS

ADMIN LOGIN AND THEIR FEATURES:







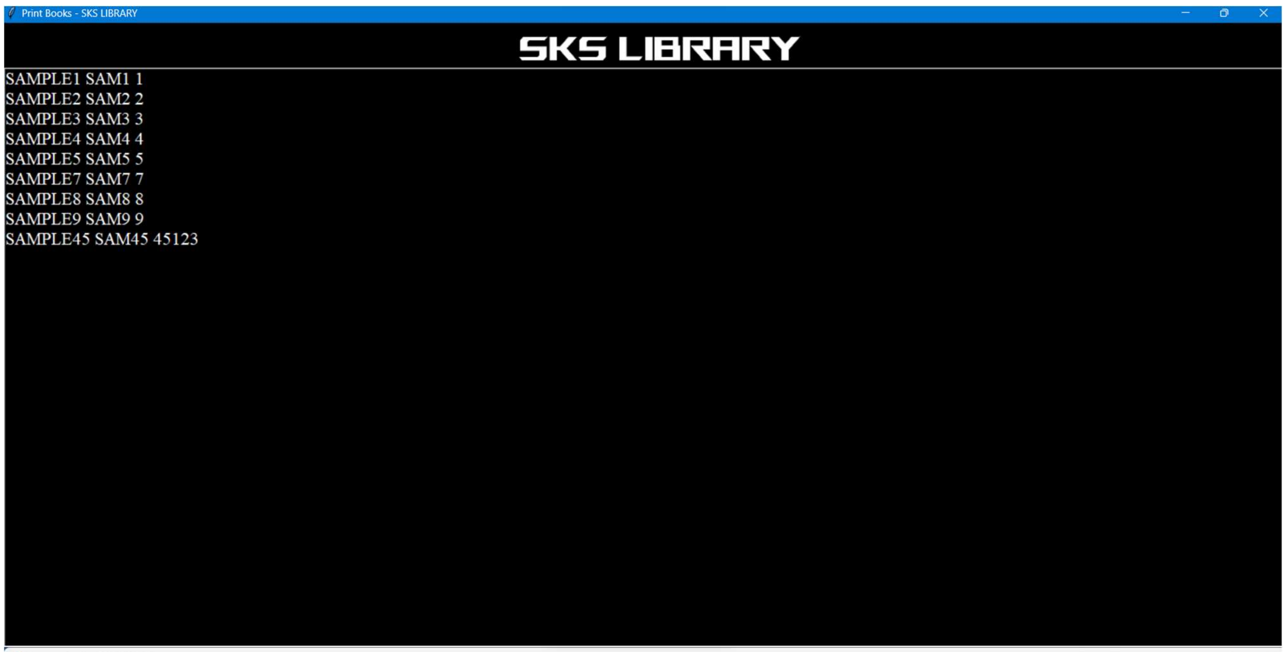


Book Name: SAMPLE45

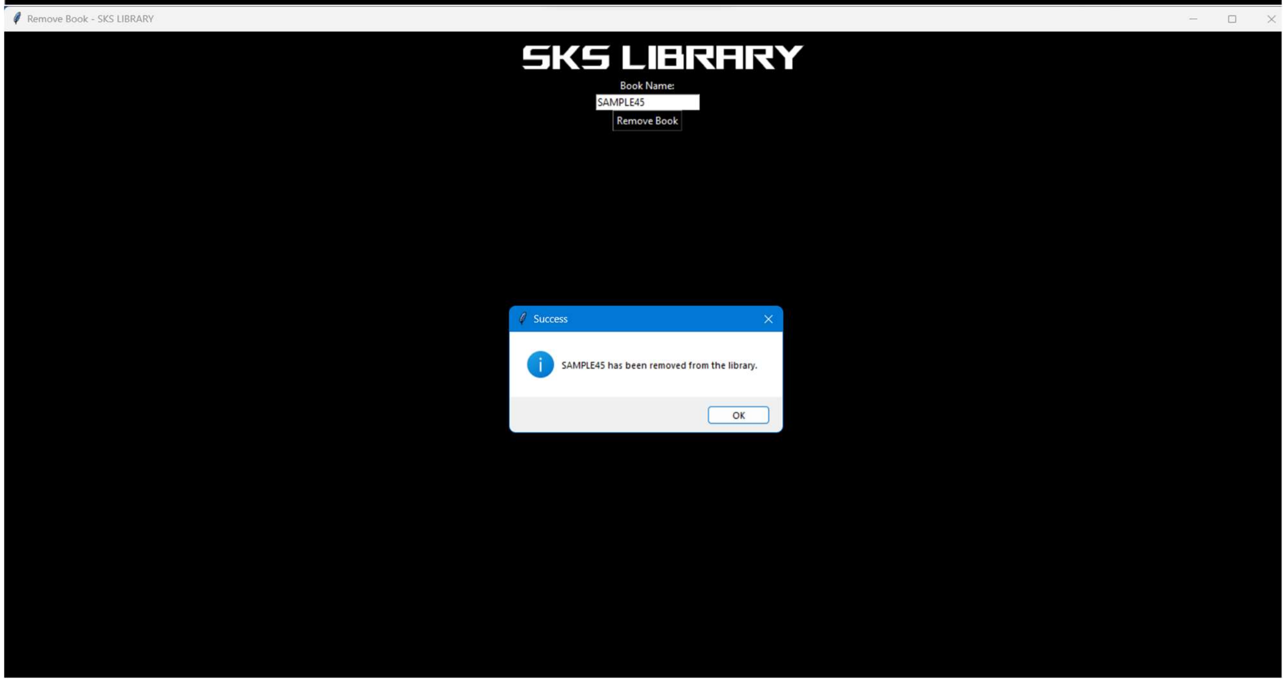
Author Name: SAM45

Book ID: 45123

Book added to the library.

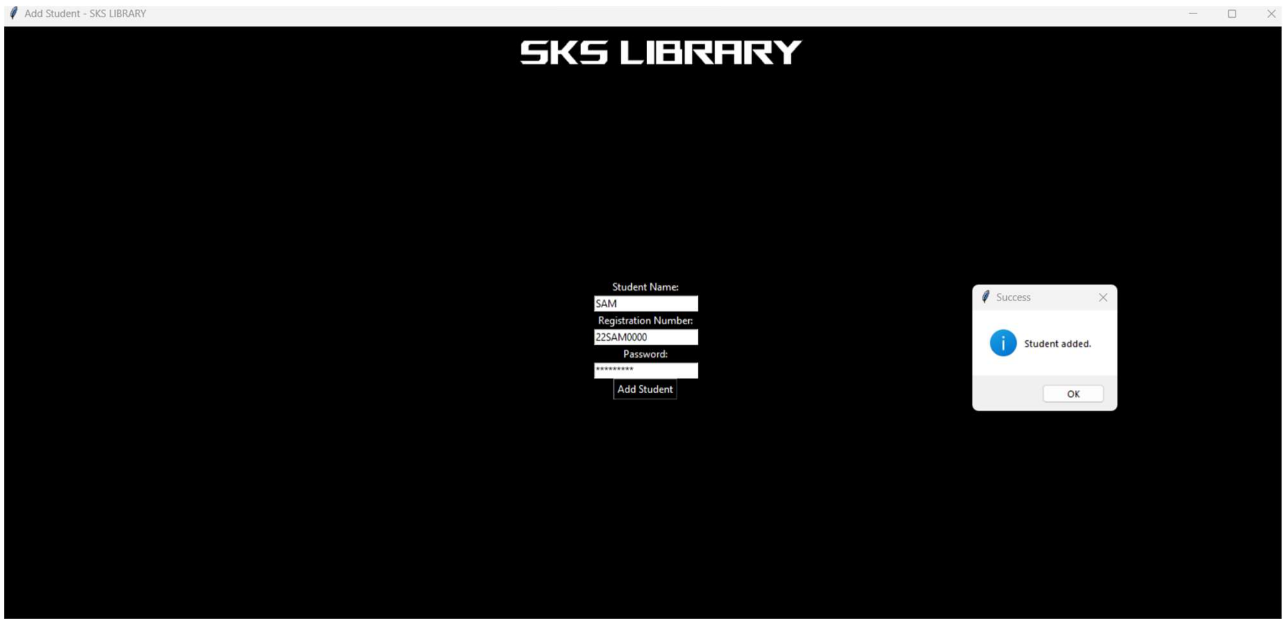


```
SAMPLE1 SAM1 1
SAMPLE2 SAM2 2
SAMPLE3 SAM3 3
SAMPLE4 SAM4 4
SAMPLE5 SAM5 5
SAMPLE7 SAM7 7
SAMPLE8 SAM8 8
SAMPLE9 SAM9 9
SAMPLE45 SAM45 45123
```



Book Name: SAMPLE45

SAMPLE45 has been removed from the library.

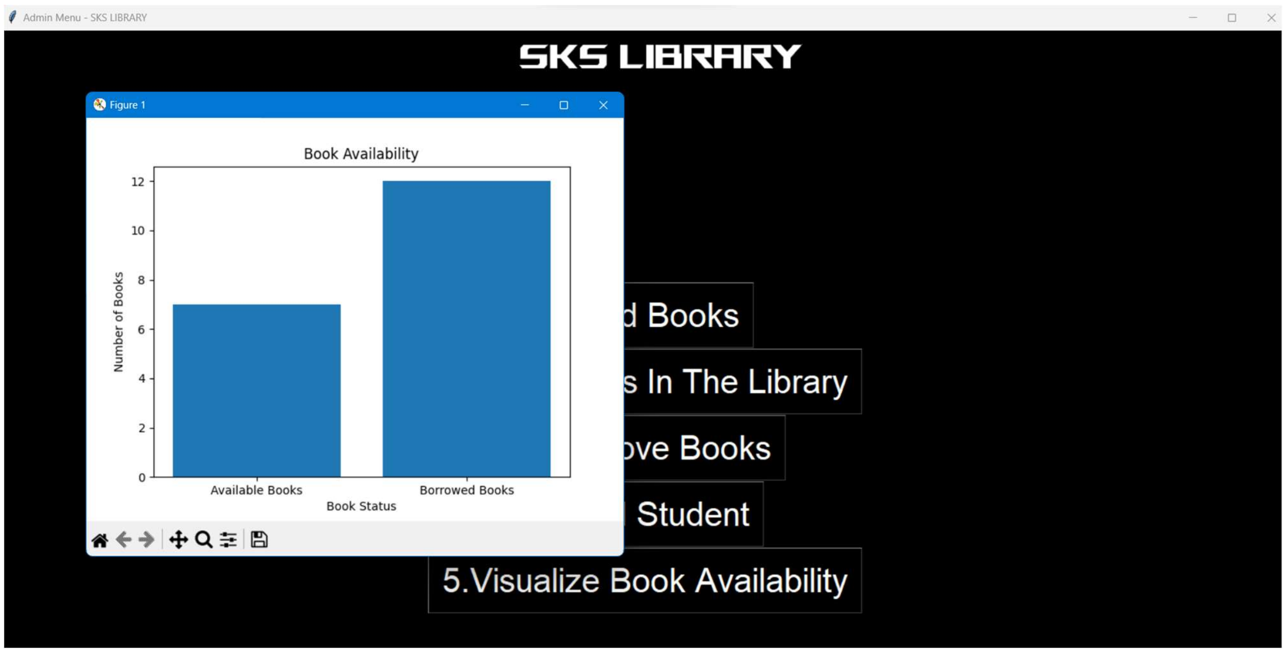

Add Student - SKS LIBRARY

SKS LIBRARY

Student Name:
SAM
Registration Number:
22SAM0000
Password:
****

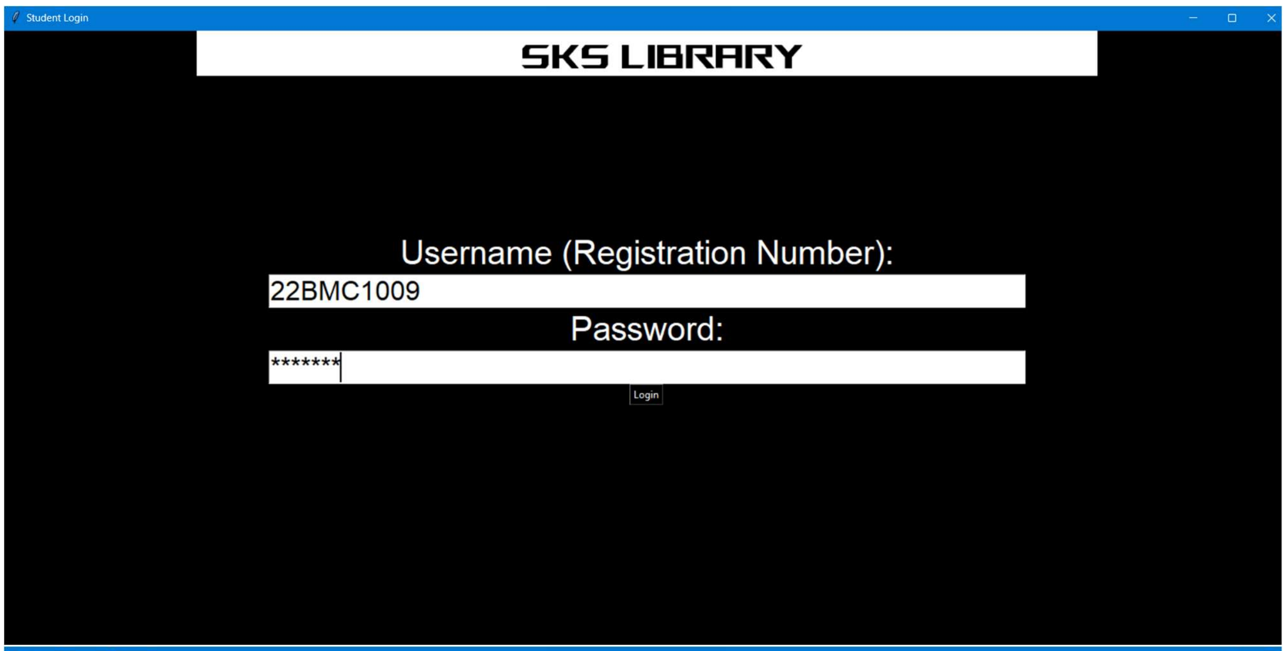
Success

Student added.


Admin Menu - SKS LIBRARY

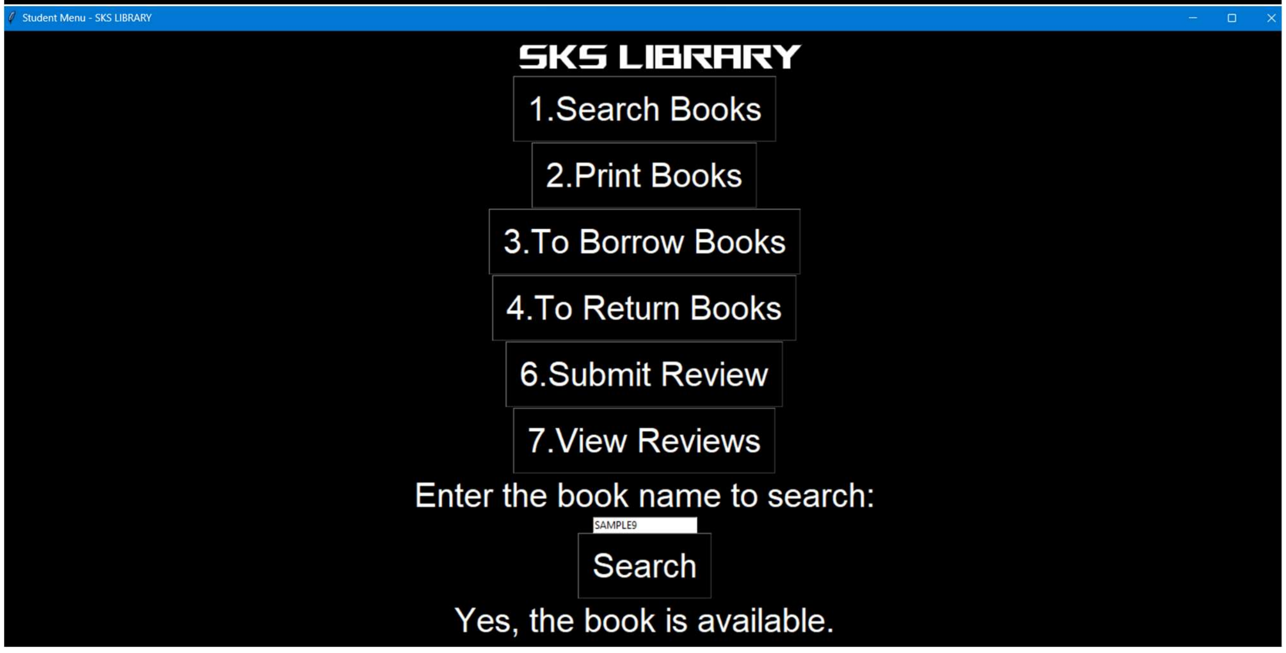
SKS LIBRARY

STUDENT LOGIN AND THEIR FEATURES:



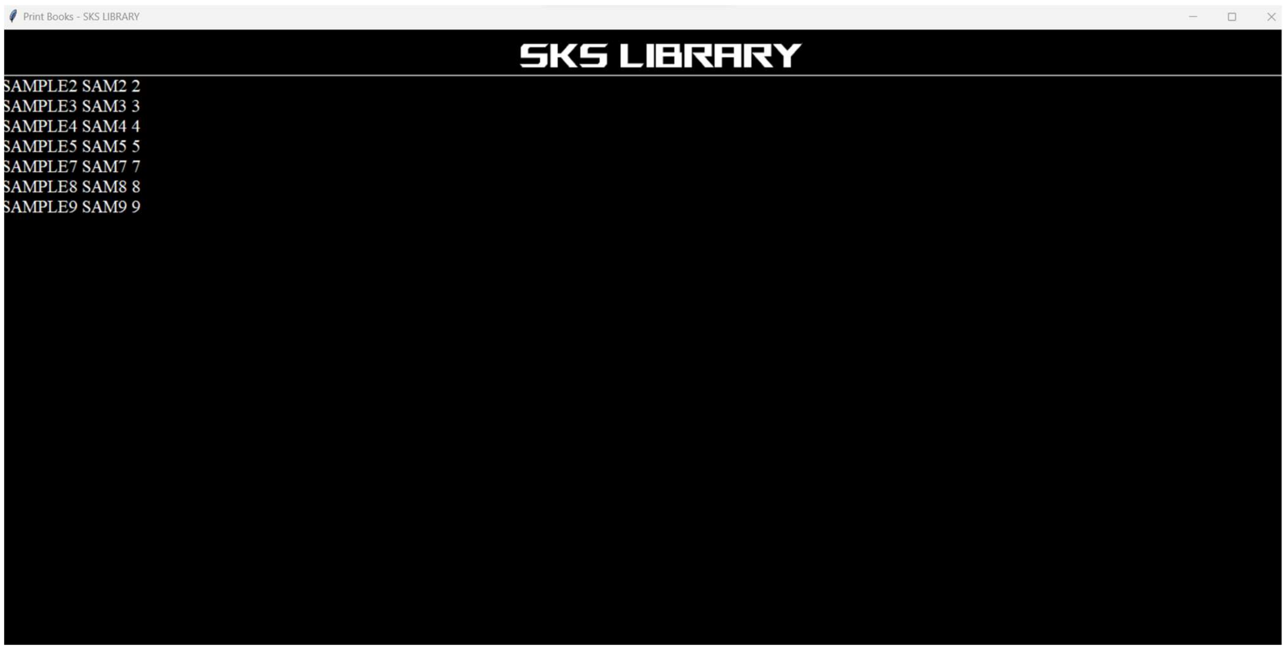
Username (Registration Number): 22BMC1009

Password: *******



Enter the book name to search: SAMPLE9

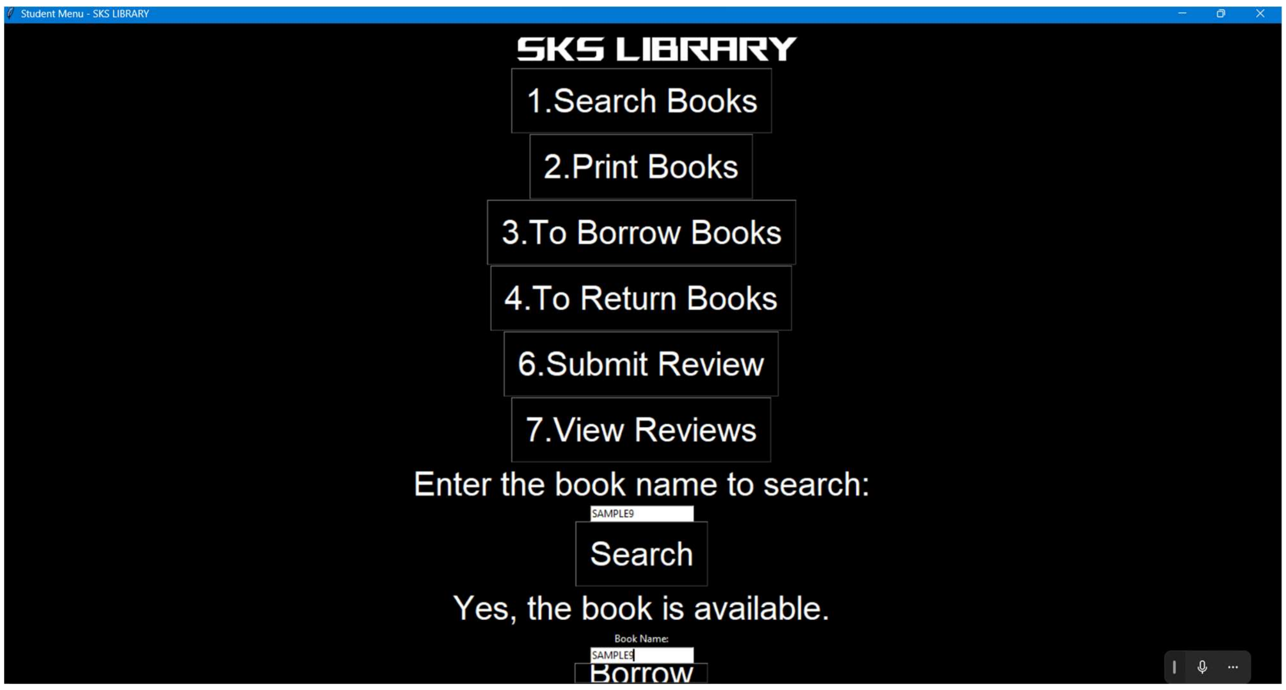
Yes, the book is available.



Print Books - SKS LIBRARY

SKS LIBRARY

```
SAMPLE2 SAM2 2
SAMPLE3 SAM3 3
SAMPLE4 SAM4 4
SAMPLE5 SAM5 5
SAMPLE7 SAM7 7
SAMPLE8 SAM8 8
SAMPLE9 SAM9 9
```



Student Menu - SKS LIBRARY

SKS LIBRARY

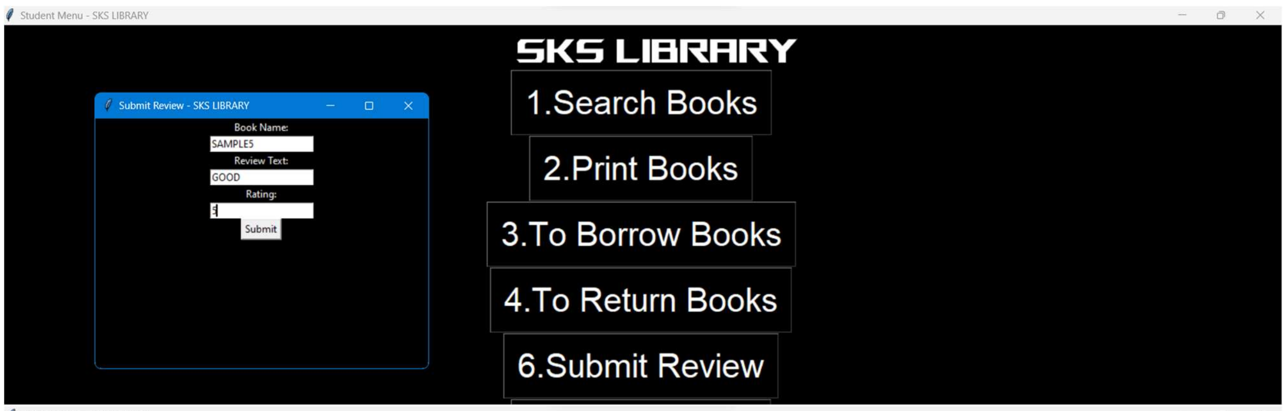
Enter the book name to search:

SAMPLE9

Yes, the book is available.

Book Name:

SAMPLE9



Student Menu - SKS LIBRARY

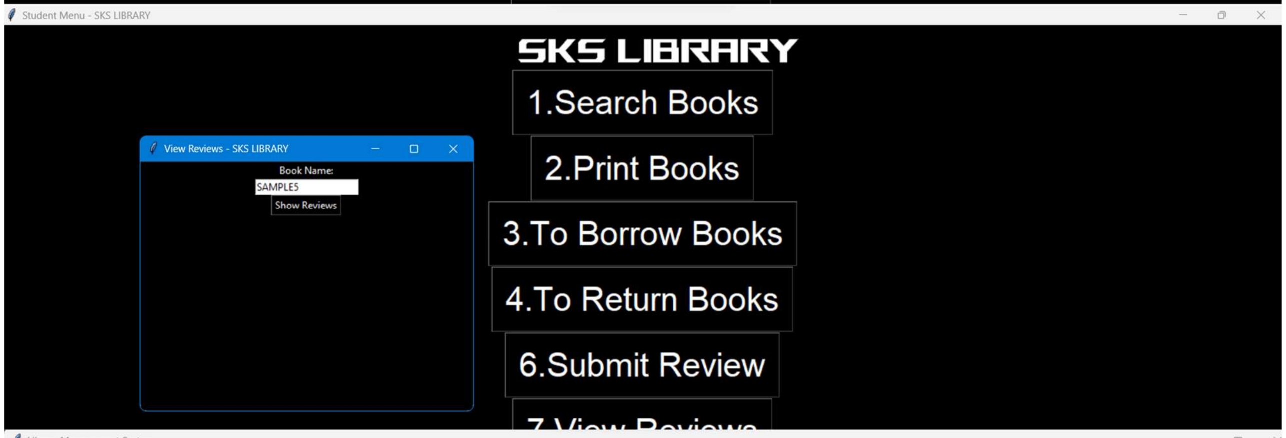
SKS LIBRARY

Submit Review - SKS LIBRARY

Book Name: SAMPLE5

Review Text: GOOD

Rating: 5

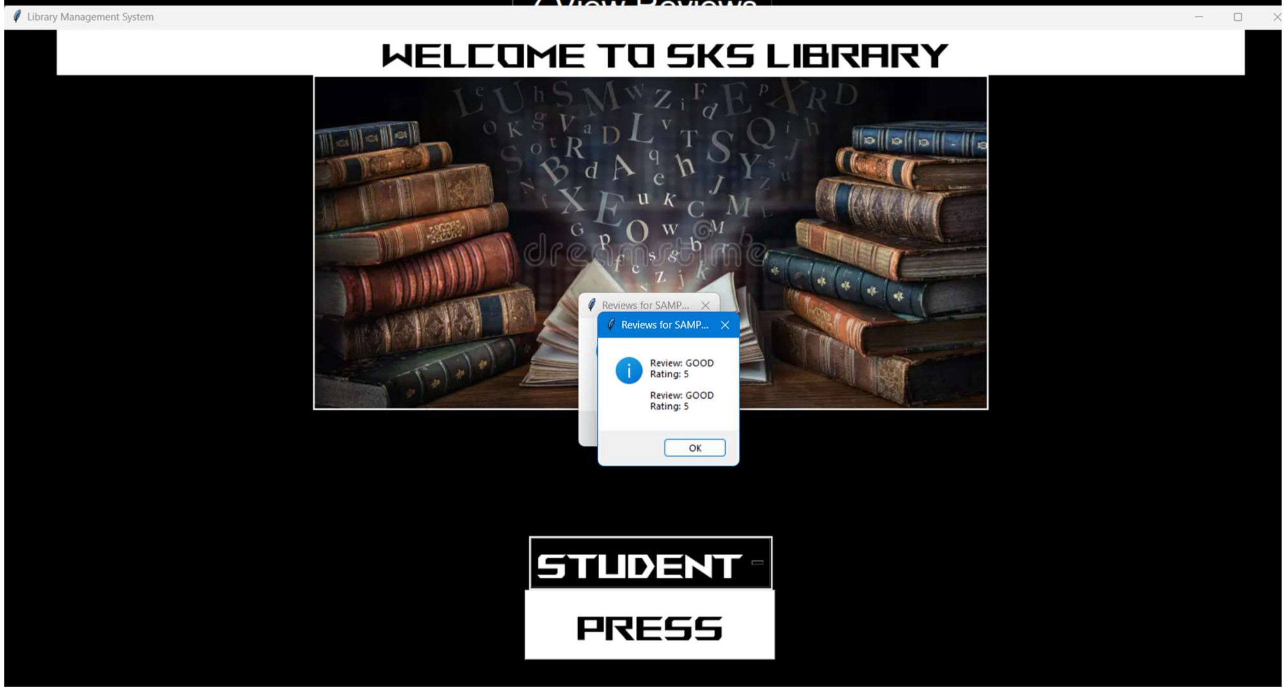


Student Menu - SKS LIBRARY

SKS LIBRARY

View Reviews - SKS LIBRARY

Book Name: SAMPLE5



Library Management System

WELCOME TO SKS LIBRARY

Reviews for SAMP...

Review: GOOD
Rating: 5

Review: GOOD
Rating: 5

CONCLUSION:

The library management system has been implemented successfully using GUI interface. This project facilitates the efficient working of a library with authenticated access. The output has been verified using the database where the tables for users are stored. The graph is used to find the number of books available in the library so that the admin members can monitor the total number of books. Further developments such as incorporation of fine for delayed return of books and amount based on date of return can be implemented in the same scenerio. Data structures namely, array to store values has been used for organizing the data of books and retrieving information regarding the same in a systematic manner.

REFERENCES:

1. Adamson, Veronica, et al. (2008). "JISC & SCONUL Library Management Systems Study" (PDF). (1 MB). Sheffield, UK: Sero Consulting. p. 51. Retrieved on 21 January 2009. "... a Library Management System (LMS or ILS 'Integrated Library System' in US parlance)." Some useful library automation software are: KOHA ,Greenstone ,Libsys, and granthlaya.

2. Wallace, Patricia M. (1991). Gary M. Pitkin (ed.). Library Systems Migration: An Introduction. Westport, CT: Meckler. pp. 1–7 [4]. ISBN 0-88736-738-0.

3.Kochtanek, Thomas R. (2002). "1 – The Evolution of LIS and Enabling Technologies". Library Information Systems: From Library Automation to Distributed Information Access Solutions. Westport, CT: Libraries Unlimited. p. 4. ISBN 1-59158-018-8.