Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

CSE 321: Software Engineering – 3rd Year CSE – 1st Semester 2019/2020

Library Management System

Abstract

Library Is a collection of sources of information and similar resources, made accessible to a defined community for reference or borrowing.Thus the process of handling a library manually is very troublesome and clumsy.As regards to this point of view, the computerized system for handling the activities of library management provides a comprehensive way to lessen physical labour, to reduce complexity of the manual system and soon. This project aims to design and implement a computerized library management system. Thus, this software is built to handle the primary functions of a library. Libraries rely on library management systems to manage and store book information electronically. The system helps both students and library managers to keep a constant track of all the books available in the library. It allows both the admin and the student to search for the desired book. It also allows libraries to keep a continuous check on the books issued and returned and even calculate fine.

The Library Management System will comprise of:

- Acquisition (allowing the library to add new books to its collection)
- Cataloging (classifying books by types and genres)
- Circulation (lending books to students and receiving them back)

Table of Contents

1. Introduction

Library management system is a project which aims in developing a computerized system to maintain all the daily work of the library. This project has many features which are generally not available in normal libraries like facility of admin login through which the admin can monitor the whole system. It also has a facility where students, after logging into the system, can see a list of books available in the library's collection and also the students can borrow and return books. The librarian after logging into his account, ie. admin account, can monitor and control every aspect of the library's operations. This project of ours is being developed to help the students as well as staff of the library to maintain the library in the best way possible and also reduce the human efforts.

1.1. Purpose

The purpose of this document is to convey all the necessary information needed to understand, run, and use the software.

Additionally, this document provides:

- Software design and modelling using UML notation.
- A specification of the application's functional and non-functional requirements.
- A definition of the application's capabilities.
- A description of the environment in which the application is expected to operate.

The document is intended to serve several groups of audiences:

First, it is anticipated that it will be used by the application designers. Designers will use the information recorded here as the basis for creating the application's design.

Secondly, the client for the project, the library manager in our case, is expected to review this document, as it will serve to establish a basis for agreement between the client and development team about the functionality to be provided by the application.

Finally, the application maintainers will review the document to clarity their understanding of what the application does.

1.2. List of Definitions

Acronym	Definition
LMS	Library management system
BDB	Book Database
SDB	Student Database
UI	User interface

1.3. Scope

The scope of the Library Management System project is as follows:

- ➢ To assist the staff in capturing the effort spent on their respective working areas.
- ➢ To utilize resources in an efficient manner by increasing their productivity through automation.
- ➢ The system generates types of information that can be used for various purposes.

The Library Management System will be PC-base that allows library users to search for books and view the available ones, and library staff members to manage the book inventory and corresponding databases.

1.4. Overview

This document provides the user with a comprehensive description of the software provided in the library management system, including system description, detailed design features, software GUI and user manual description. The document is split up into the following sections:

1. An introduction which briefly describes the purpose of this document and LMS.
2. General description of the system by using UML diagrams.
3. User manual to help the user to interact with the system.

2. General Description

2.1. Product Perspective

LMS is a stand-alone system used by the library manager, librarian, and students. The system is self-contained. However, it is possible to exchange data with other systems through external interface if required.

2.2. General Capabilities

The application will have the following capabilities:

- Library staff will be able to manage and have full control over the library's students' accounts.
- Library staff will be able to manage and have full control over the book inventory database.
- The application will provide search for books by title functionality.
- Students can easily borrow and return books, provided that certain limits aren't exceeded.
- The database is readily available for administrative purposes.

The project's client has determined that this application will provide the following benefits:

1. Provide additional flexibility and convenience to the library users.
2. Provide better reliability and security of the library information.
3. Provide a more productive environment for the library .
4. Reduce the cost of the library operations.
5. The availability of information at any time in any place.

2.3. General Constraints

This system is a Windows-based application, there will be a need to provide PC hardware connected to the system.

LMS can potentially have hundreds of users. Therefore, the system should be designed to be easy to use, providing help instructions, and appropriate error messages for invalid user inputs.

Security is important to library operation. Library users are allowed to use the LMS only for searching book records. Users should never be able to break into the system and to perform any modification.

Reliability is vital to library operation. The LMS should not suffer any data loss, even if the system fails for any reason. The data should be recovered and operations should re-run smoothly.

2.4. User Characteristics

The two types of user for the LMS are:

- Librarian
- Library User

The following table describes general users characteristics that will affect the functionality of the software product.

User	User Characteristics	Role
Librarian	Good understanding of library operations. Should receive a brief training on how to use the system	Responsible for library day-to-day operations. Must be able to manage the book and the student databases.
Student	Will not receive any formal training to use the system.	Search and view the library's book collection.

2.5. Environment Description

The project is a Windows-based environment.

2.6. Assumptions and Dependencies

The following is a list of assumptions and dependencies that would affect the software requirements if they turned out to be false:

- Users have a basic understanding of PC and Windows.
- The librarian should have some Excel skills in order to interact with the database.

2.7. Other resources needed

- Datetime Module that was useful for calculating late fees.
- Microsoft Excel was used as a database.
- Openpyxl Module that was useful for working with an excel database.
- PyQt5 Module that was used to create the GUI.

3. System Requirements

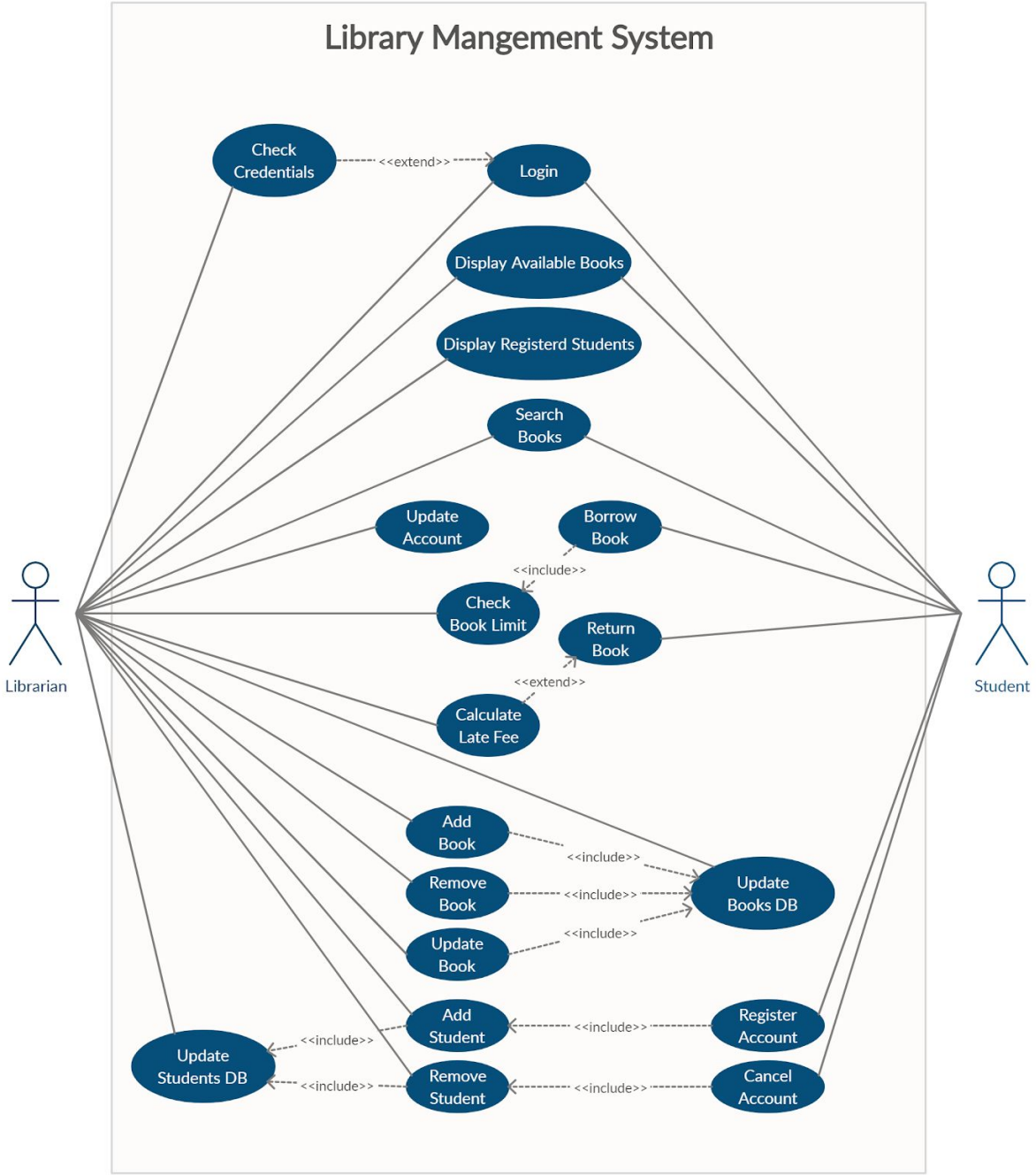
3.1. Functional Requirements

1. Librarians should be able to access admin functionality through a simple login system.
2. Librarians should be able to change their login credentials.
3. Librarians should be able to add new/remove books to/from the library's collection.
4. Librarians should be able to update existing books in the library's collection.
5. Books shall have a unique identification number and other details including a rack number which will help to physically locate it.
6. Librarians should be able to view all students that are registered in the library.
7. Librarians should be able to add/remove students from the library's database.
8. There should be a limit on how many books a student can borrow.
9. There should be a limit on how many days a student can borrow a book, after which the student must pay a fee.
10. Librarians and students shall be able to view the library's book collection.
11. Librarians and students should be able to search for books by title.
12. Students should be able to borrow and return books.
13. System should calculate fees for late returns.

3.2. Non-functional Requirements

1. System is to be implemented using Python 3.8.
2. Development timespan is 2 months.
3. In case of system failure, no data shall be lost.
4. Response time of the system shall not exceed 500ms.
5. System should follow the Style Guide for Python Code PEP8.
6. System should follow all applicable copyright laws and the Engineer's code of ethics.

4. Use-Case Diagram



5. Narrative Description of Use Cases

Use Case Name	Login
Related Requirements	FR-1
Goal in Context	Librarian can login into the system
Pre-condition	Librarian has opened LMS Interface
Successful End Condition	Librarian successfully logged into the system
Failed End Condition	Login failed
Primary Actor	Librarian
Secondary Actor	None
Trigger	Librarian clicked the login button on UI
Main Flow	1. Librarian opens the LMS program. 2. Librarian clicks on the login button. 3. Librarian enters his username and password. 4. Librarian successfully logs into the system.
Extensions	1. System checks entered credentials. 2. Successful end condition is achieved if the entered credentials match the actual ones. 3. Otherwise, the login fails.

Use Case Name	Add/Remove/Edit Book
Related Requirements	FR-3
Goal in Context	Librarian adds/removes/edits the required book to the BDB
Pre-condition	Librarian has sufficient information about the book
Successful End Condition	Book is added/removed/edited successfully
Failed End Condition	Book addition/removal/edit failed
Primary Actor	Librarian
Secondary Actor	None
Trigger	Add/Remove/Edit book button is clicked
Main Flow	1. Librarian opens the LMS program. 2. Librarian logs into the system. 3. Librarian enters book information. 4. Book is successfully added/removed/edited in the system.
Extensions	1. System checks if the entered information is sufficient and correct. 2. If so, the book will be added/removed/edited in the BDB 3. Otherwise, book addition/removal/edit will fail.

Use Case Name	Display Available Books
Related Requirements	FR-10
Goal in Context	View the library's entire book collection
Pre-condition	None
Successful End Condition	The user successfully viewed the available books in the library.
Failed End Condition	The user failed to view the available book in the library.
Primary Actor	Librarian/Student
Secondary Actor	None
Trigger	User clicked on the "view all" button in the books tab.
Main Flow	1. User opens the LMS program. 2. User logs into the system. 3. User clicks "view all". 4. The available books are successfully displayed.
Extensions	None

Use Case Name	Search Books
Related Requirements	FR-11
Goal in Context	Users are able to search for a book by its title.
Pre-condition	Book already exists in the library's collection.
Successful End Condition	User successfully found the required book.
Failed End Condition	User failed to find the required book
Primary Actor	Librarian/Student
Secondary Actor	None
Trigger	User clicks the “search” button.
Main Flow	1. User opens the LMS program. 2. User logs into the system. 3. User enters the book title. 4. Users click “search”. 5. The matched books are successfully displayed.
Exceptions	1. If the entered title doesn’t match any books in the BDB, no books will be displayed.

Use Case Name	Display Registered Students
Related Requirements	FR-6
Goal in Context	Librarians can view all registered students in the library.
Pre-condition	There are registered students.
Successful End Condition	Librarians successfully view the registered students in the library system.
Failed End Condition	Librarians fail to view the registered students in the library system.
Primary Actor	Librarian
Secondary Actor	None
Trigger	Librarian clicks the “view all” button in the students tab.
Main Flow	1. Librarian logs in successfully into the system. 2. Librarians access the SDB through the UI. 3. Librarian clicks the “view all” button. 4. The registered students are successfully displayed.
Exception	1. Librarian failed to login into the system. 2. There aren’t any registered students in the library.

Use Case Name	Update Account
Related Requirements	FR-2
Goal in Context	Librarian updates admin credentials
Pre-condition	None
Successful End Condition	Librarian successfully updates admin credentials
Failed End Condition	Librarian failed to update admin credentials
Primary Actor	Librarian
Secondary Actor	None
Trigger	Librarian clicks on “Change Admin Credentials”
Main Flow	1. Librarian logs in successfully into the system. 2. Librarian clicks on “Change Admin Credentials” 3. Librarian enters new credentials. 4. Credentials are updated successfully.
Exceptions	1. Librarian failed to login into the system.

Use Case Name	Add/Remove Student
Related Requirements	FR-7
Goal in Context	Librarian adds/removes a student in the SDB
Pre-condition	Librarian has sufficient information about the student.
Successful End Condition	Add/Removal is successful.
Failed End Condition	Add/Removal failed
Primary Actor	Librarian
Secondary Actor	Student
Trigger	Librarian clicks the “add/remove” button in students tab.
Main Flow	1. Librarian opens the LMS program. 2. Librarian logs into the system. 3. Librarian enters student information. 4. Students are successfully added/removed in the system.
Extensions	1. System checks if the entered information is sufficient and correct. 2. If so, the student will be added/removed in the SDB. 3. Otherwise, the student addition/removal will fail.

6. Requirements Validation

1. Functional Requirements

1.1. **Librarians should be able to access admin functionality through a simple login system.**

1.2. **Librarians should be able to add new/remove books to/from the library's collection.**

1.3. **Librarians should be able to update existing books in the library's collection.**

1.4. **Books shall have a unique identification number and other details including a rack number which will help to physically locate it.**

1.5. **Librarians should be able to add/remove students from the library's database.**

1.6. **There should be a limit on how many books a student can borrow.**

1.7. **There should be a limit on how many days a student can borrow a book, after which the student must pay a fee.**

1.8. **Librarians and students shall be able to view the library's book collection.**

1.9. **Students should be able to search for books by title.**

1.10. **Students should be able to borrow and return books.**

1.11. **System should calculate fees for late returns.**

2. Non-functional Requirements

2.1. **System is to be implemented using Python 3.8.**

2.2. **Development timespan is 2 months.**

2.3. **In case of system failure, no data shall be lost.**

2.4. **Response time of the system shall not exceed 500ms.**

2.5. **System should follow the Style Guide for Python Code PEP8.**

2.6. **System should follow all applicable copyright laws and Engineer's code of ethics.**

Requirements Traceability Matrix

	Req 1.1	Req 1.2	Req 1.3	Req 1.4	Req 1.5	Req 1.6	Req 1.7	Req 1.8	Req 1.9	Req 1.10	Req 1.11
Req 1.1		✓	✓		✓			✓			
Req 1.2	✓										
Req 1.3	✓										
Req 1.4											
Req 1.5	✓										
Req 1.6							✓				
Req 1.7						✓					✓
Req 1.8	✓										
Req 1.9										✓	
Req 1.10									✓		
Req 1.11							✓				

Source Traceability Matrix

	Students	Librarian	Developers	Sponsor	Project Manager	Executive Manager
Req 1.1		✓				
Req 1.2		✓				
Req 1.3		✓				
Req 1.4		✓				
Req 1.5	✓	✓				
Req 1.6	✓					
Req 1.7	✓					
Req 1.8	✓	✓				
Req 1.9	✓					
Req 1.10	✓					
Req 1.11		✓				
Req 2.1			✓		✓	
Req 2.2						✓
Req 2.3	✓	✓				
Req 2.4	✓	✓				
Req 2.5						✓
Req 2.6						✓

7. Class Model

Noun Extraction

Librarians should be able to access admin functionality through a simple login system.

Librarians should be able to add new/remove books to/from the library's collection.

Librarians should be able to update existing books in the library's collection.

Books shall have a title, an author, a unique identification number and other details including a rack number which will help to physically locate it.

Librarians should be able to add/remove students from the library's database.

There should be a limit on how many books a student can borrow.

There should be a limit on how many days a student can borrow a book, after which the student must pay a fee.

Librarians and students shall be able to view the library's book collection.

Students should be able to search for books by title.

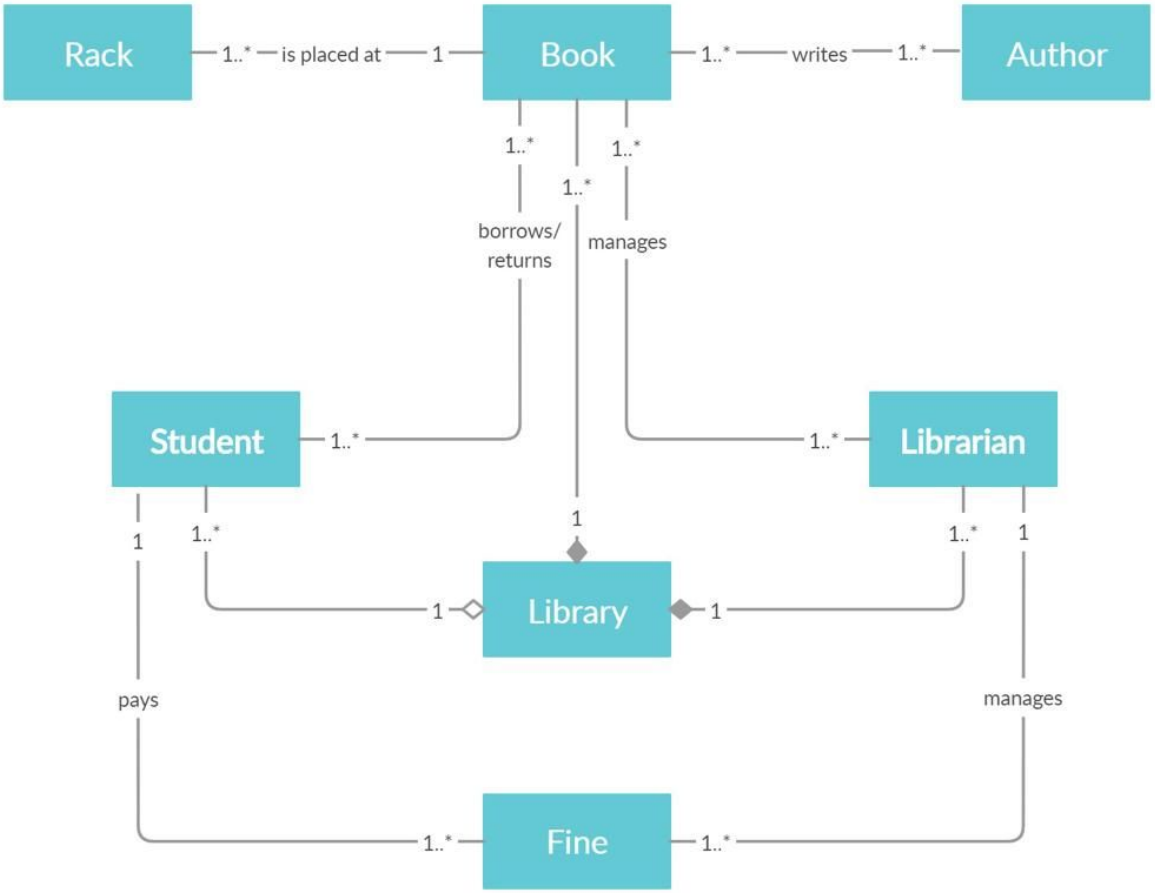
Students should be able to borrow and return books.

System should calculate fees for late returns.

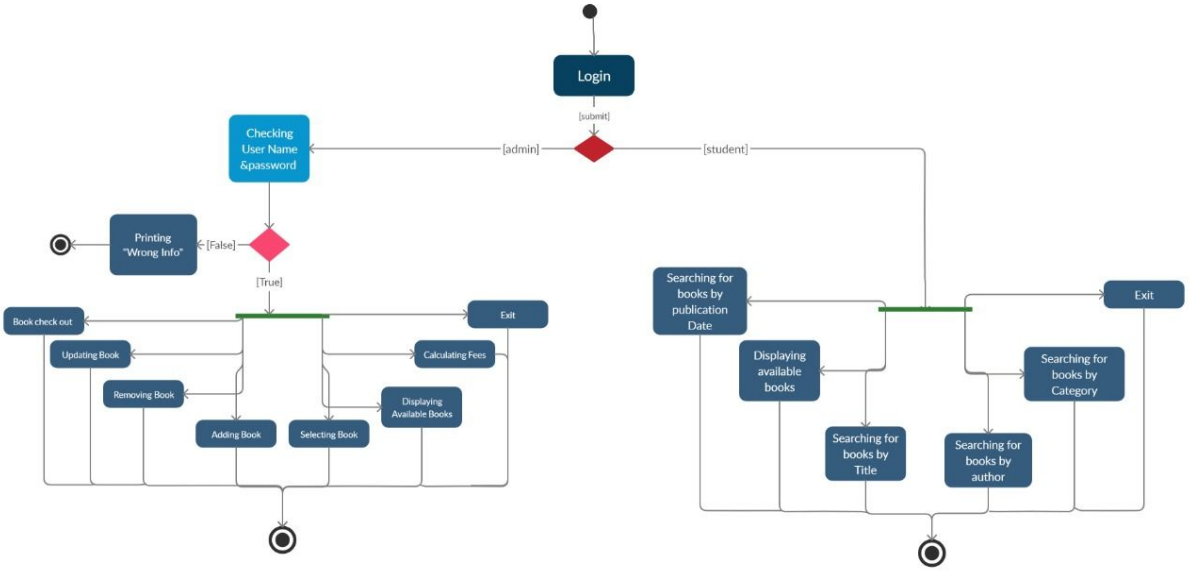
List of extracted classes:

1. Librarian
2. Books
3. Library
4. Rack
5. Students
6. Fee

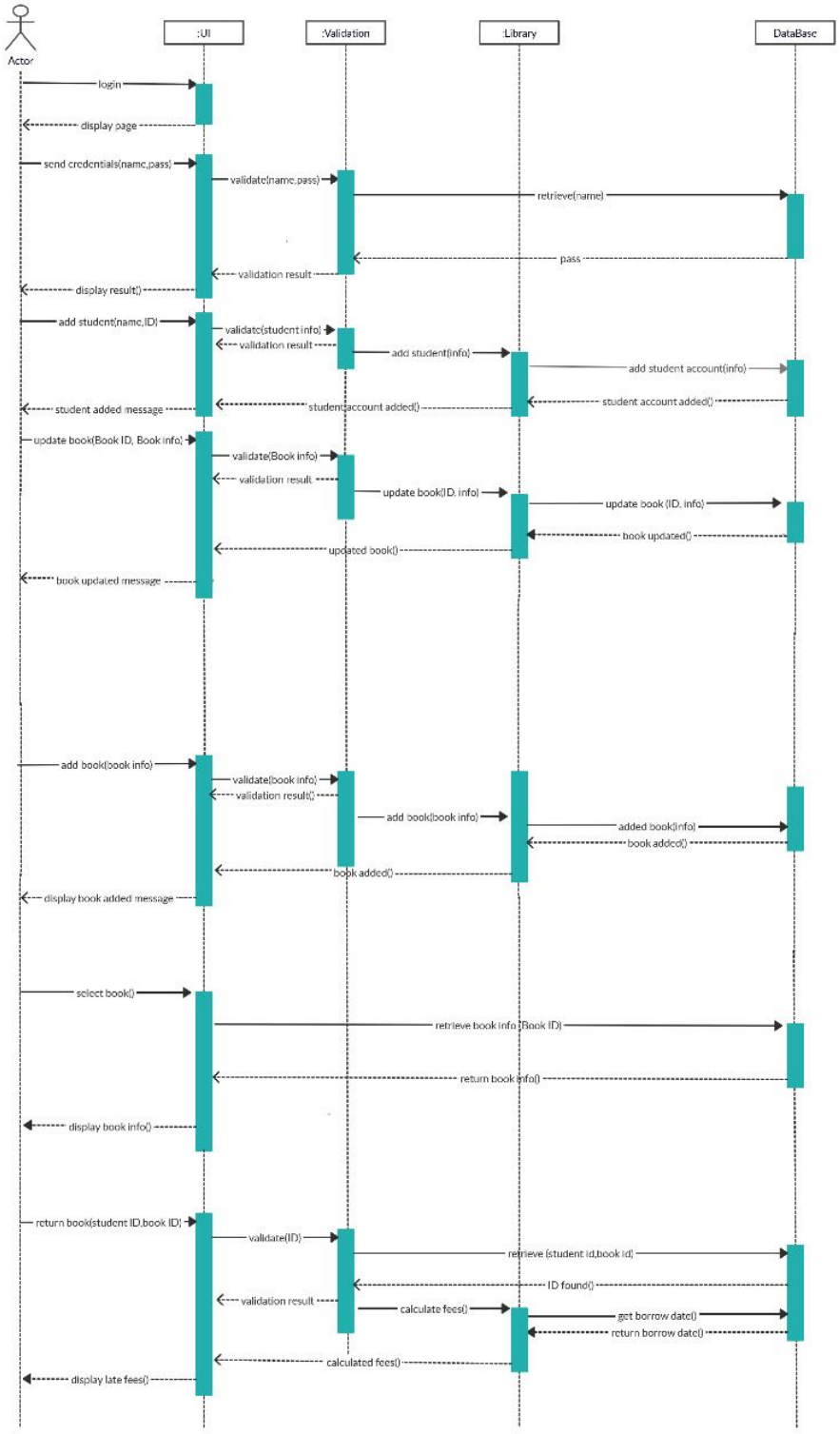
Classes	Relationships
Book is placed at rack	Association
Author writes book	Association
Student borrows/returns books	Association
Librarian manages books	Association
Librarian manages fine	Association
Student pays fine	Association
Library has books	Composition
Library has a librarian	Composition
Library has students	Aggregation



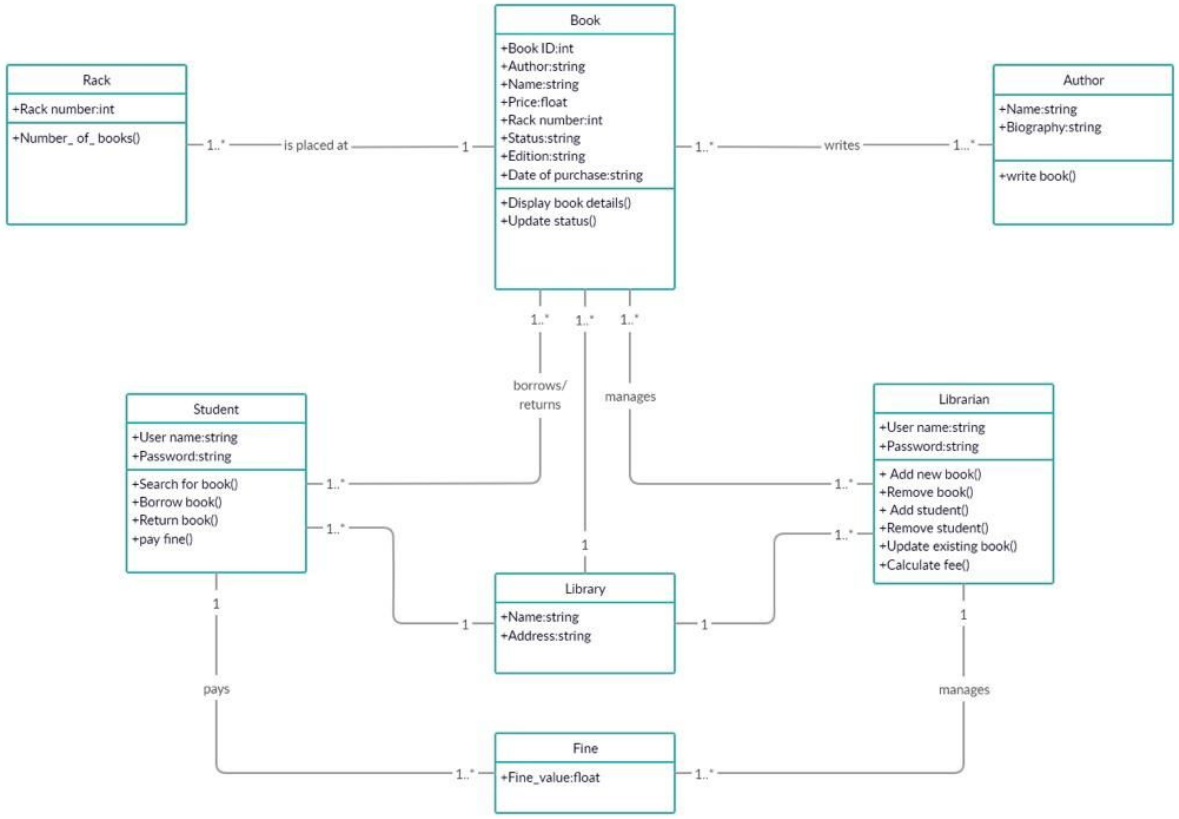
8. State Diagram



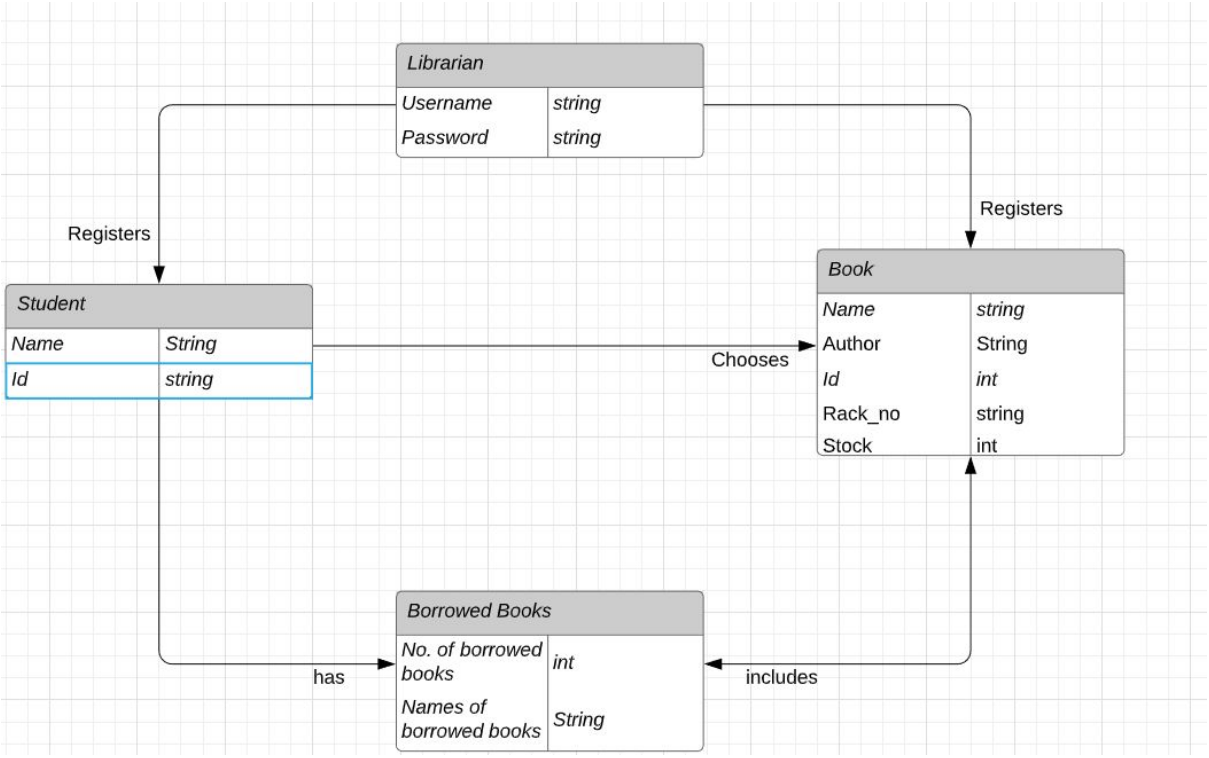
9. Interaction Diagram



10. Detailed Class Diagram



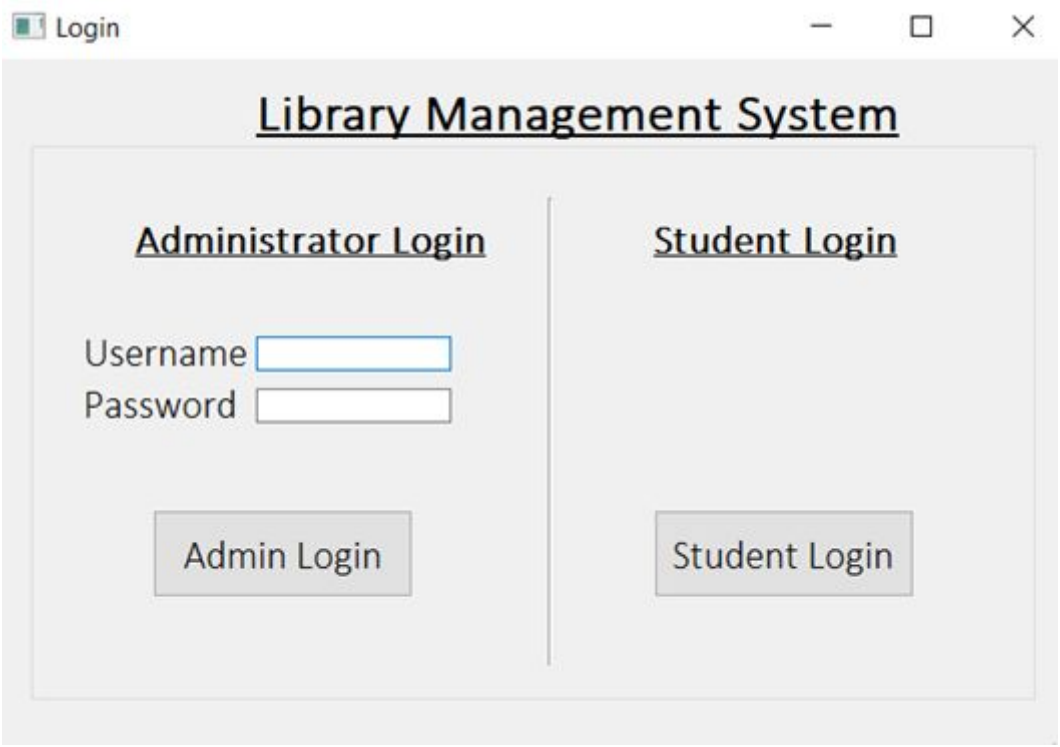
11. Data Model Design



12. User Interface Design

Form fill-in interaction style was used, for more intuitive, easy-to-use user interface.

Shown below is the Login Window that is shown to the user once the software is run. The user will have the option to login as a system administrator, provided they entered correct credentials, or they could simply login as students but have limited functionality,



There are two actions within this window:

- "Admin Login" changes the UI from "Login Window" to "Administrator Window".
- "Student Login" changes the UI from "Login Window" to "Student Window".

An option to go back to the "Login Window" is provided from any other window in the system.

Students Window

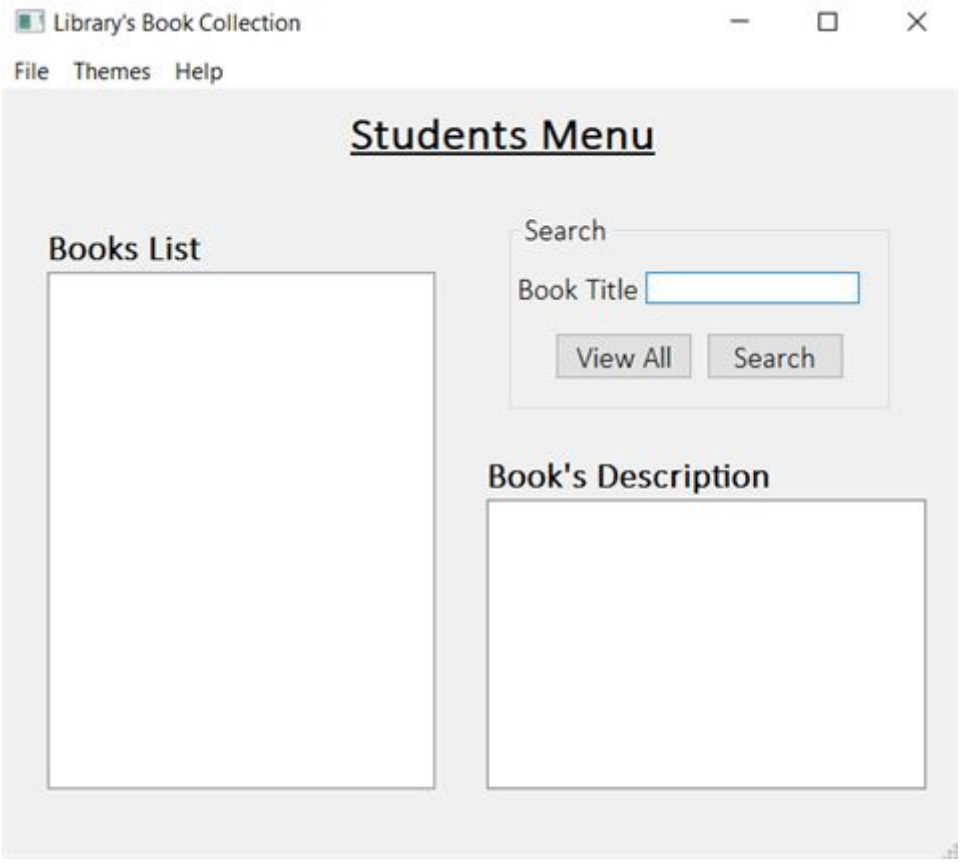
This window is freely accessed by any student. It allows students to:

- ➢ View all available books in the library's collection.
- ➢ Search the library's collection for any book by title.
- ➢ Select and view additional information about any book they pick, including where that book is physically located within the library.

The UI was designed to be intuitive, easy-to-use and aesthetically pleasing. It additionally has a menu-bar with the following tools:

- ➢ File Menu, which allows the students to go back to the login window or exit the system.
- ➢ Themes Menu, which allows the students to change the colour scheme of the system.
- ➢ Help Menu, which provides students with a brief description of what they can do with the system.

Additional help was implemented through the use of a status bar that shows insight into the functionality of each component they hover over.



Administrator Window

This window is only accessed by librarians who enter correct credentials upon login.

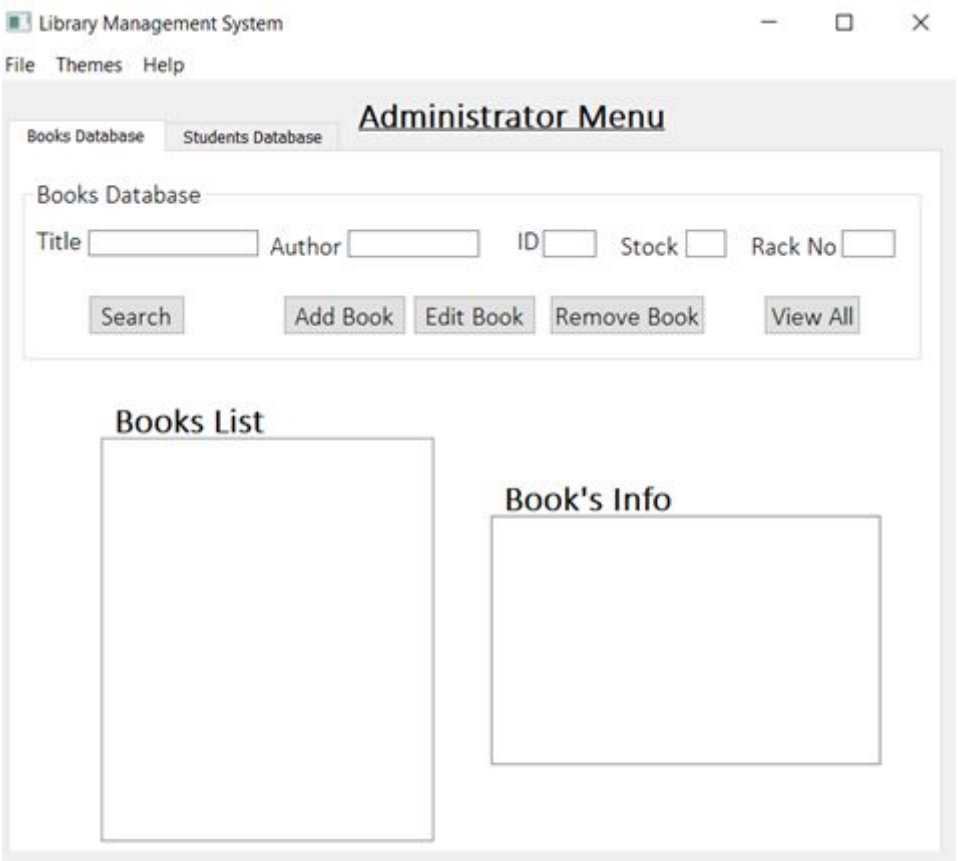
They're allowed to monitor and are given complete control over every aspect of the library's day-to-day operations.

The window is divided into two tabs:

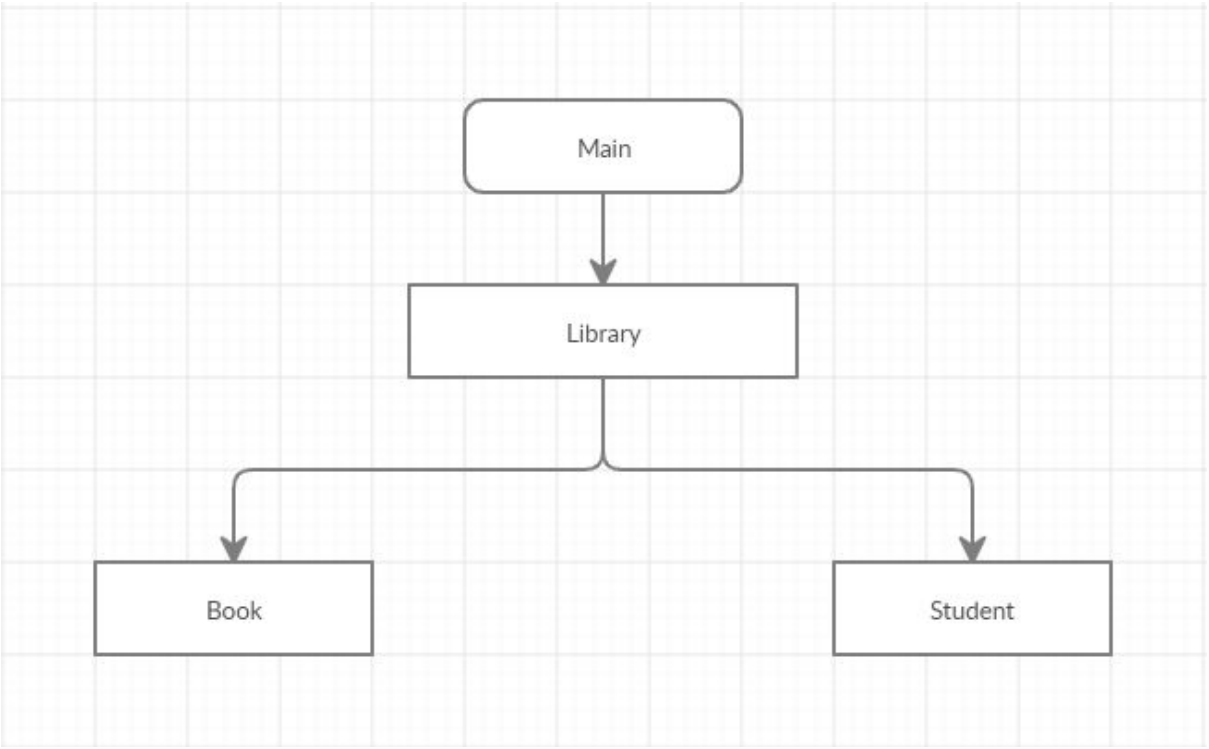
- BDB tab allows admins to monitor/control BDB by adding/removing/editing books in the library's collection.
- SDB tab allows admins to monitor/control SDB by registering/removing students to/from the library.

From the Menu-bar, they can access similar functionality as discussed previously, however some admin functionality was added to aid librarians with the circulation of books.

- File Menu now has a Lend/Return field which allows librarians to manage library's transactions.



13. Client-Object Relation Diagram



14. Detailed Design

ADD STUDENT

```
def add_student(student_list):
    student_name = input("Enter the student name")
    if student_name in student_list:
        print("sorry student_name already exists")
    else:
        student_list.append(student_name)
```

REMOVE STUDENT

```
def remove_students(student_list):
    student_name = input("Enter the student name")
    if student_name in student_list:
        student_list.remove(student_name)
    else:
        print(" The student doesn’t exist in SDB")
```

SEARCH STUDENT

```
def search_students(student_list):
    student_name = input("Enter the student name")
    if student_name in student_list:
        print(Info about the student)
    else:
        print("The name doesn't exist")
```

ADD BOOK

```
def add_book(books_list):
    book_name = input("Enter the book name")
    if book_name in books_list:
        print("Sorry the book already exists in the BDB")
    else:
        books_list.append(book_name)
```

REMOVE BOOK

```
def remove_book(books_list):
    book_name = input("Enter the book name")
    if book_name in books_list:
        books_list.remove(book name)
    else:
        print(" The book doesn't exist in BDB")
```

15. Testing

Testing was divided into three main sections:

- **Class/unit testing**
  - Where individual components/functions/classes were tested.
- **Integration testing**
  - Where classes' interfaces were tested with each other.
- **System testing**
  - Where the entire system was tested through the GUI.

Class Testing

1. **Book**

Class book has the following attributes:

- Id
- Title
- Author
- Stock
- Rack number

And the following methods:

- Getters and setters
- Print overriding
- Get info
- Update Book

Getters and setters were the first tested methods (which also tested correct instantiation) and the testing was straightforward.

Print Overriding Test

```
class Book(object):...


b1 = Book("1", "The Magicians", "Lev. Grossman", stock="4", rack_no="QA")
print(b1)
```

```
-----------------------------------------------
ID: 1
Title:  The Magicians
Author: Lev. Grossman

Process finished with exit code 0
```

get_info Test

```
class Book(object):...


b1 = Book("1", "The Magicians", "Lev. Grossman", stock="4", rack_no="QA")
print(b1.get_info())
```

```
ID: 1
Title:      The Magicians
Author:     Lev. Grossman
Stock:      4
Rack No:    QA

Process finished with exit code 0
```

Update Book Test

```
class Book(object):...


b1 = Book("1", "The Magicians", "Lev. Grossman", stock="4", rack_no="QA")
print(b1)
b1.update_book("Magicians", "Lev. G.", "2", stock="2", rack_no="QT")
print(b1)
```

```
-----------------------------------------------
ID: 1
Title:  The Magicians
Author: Lev. Grossman
-----------------------------------------------
ID: 2
Title:  Magicians
Author: Lev. G.

Process finished with exit code 0
```

2. Student

Again, getters and setters were the first tested methods (which also tested correct instantiation) and the testing was straightforward.

Print Overriding Test

```
class Student(object):...


s1 = Student("Michael", "16T0076")
print(s1)
```

```
Name:   Michael
ID:     16T0076

Process finished with exit code 0
```

Student class testing was hauled at this point and continued later on as part of integration testing. Moreover some database testing was done in order to ensure database functionality later on as part of integration and system testing.

Additionally, the following GUI classes were tested individually and the testing was mostly done in the GUI itself.

```
class LoginScreen(object):...

class StudentsMenu(object):...

class ChangeCredsWindow(object):...

class AdminWindow(object):...

class LendReturnWindow(object):...
```

Integration Testing

Student - Book Integration Test

```
class Book(object):...


class Student(object):...


b1 = Book("1", "The Magicians", "Lev. G.")
b2 = Book("2", "12 Rules for Life", "Jordan Peterson")

s1 = Student("Michael", "16T0076")

borrow_date = "1/1/2020"
s1.borrow_book(b1.get_title(), borrow_date="1/1/2020")
s1.borrow_book(b2.get_title(), borrow_date="1/1/2020")

print(s1.get_borrowed_books_list())

s1.return_book(b2.get_title())
print(s1.get_borrowed_books_list())
```

```
['The Magicians', '12 Rules for Life']
['The Magicians']

Process finished with exit code 0
```

Student - Book - Library Integration Test

```python
class Library(object):...


class Book(object):...


class Student(object):...


b1 = Book("1", "The Magicians", "Lev. G.")
b2 = Book("2", "12 Rules for Life", "Jordan Peterson")

s1 = Student("Michael", "16T0076")

LMS = Library([b1, b2], [s1], ["username", "password"])

LMS.display_available_books()
LMS.display_students()
```

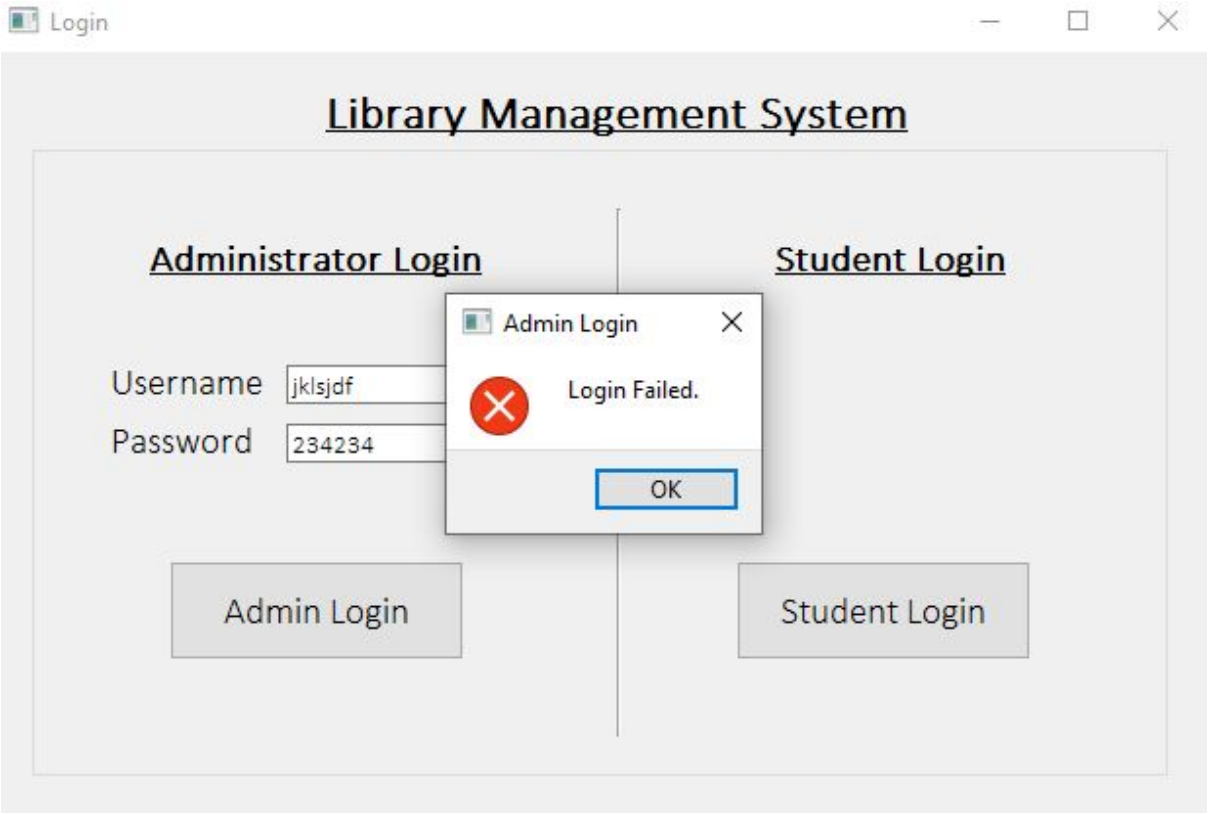
```
--------------------------------------------------
ID: 1
Title:  The Magicians
Author: Lev. G.
--------------------------------------------------
ID: 2
Title:  12 Rules for Life
Author: Jordan Peterson
--------------------------------------------------
--------------------------------------------------
Name:   Michael
ID:     16T0076
--------------------------------------------------

Process finished with exit code 0
```

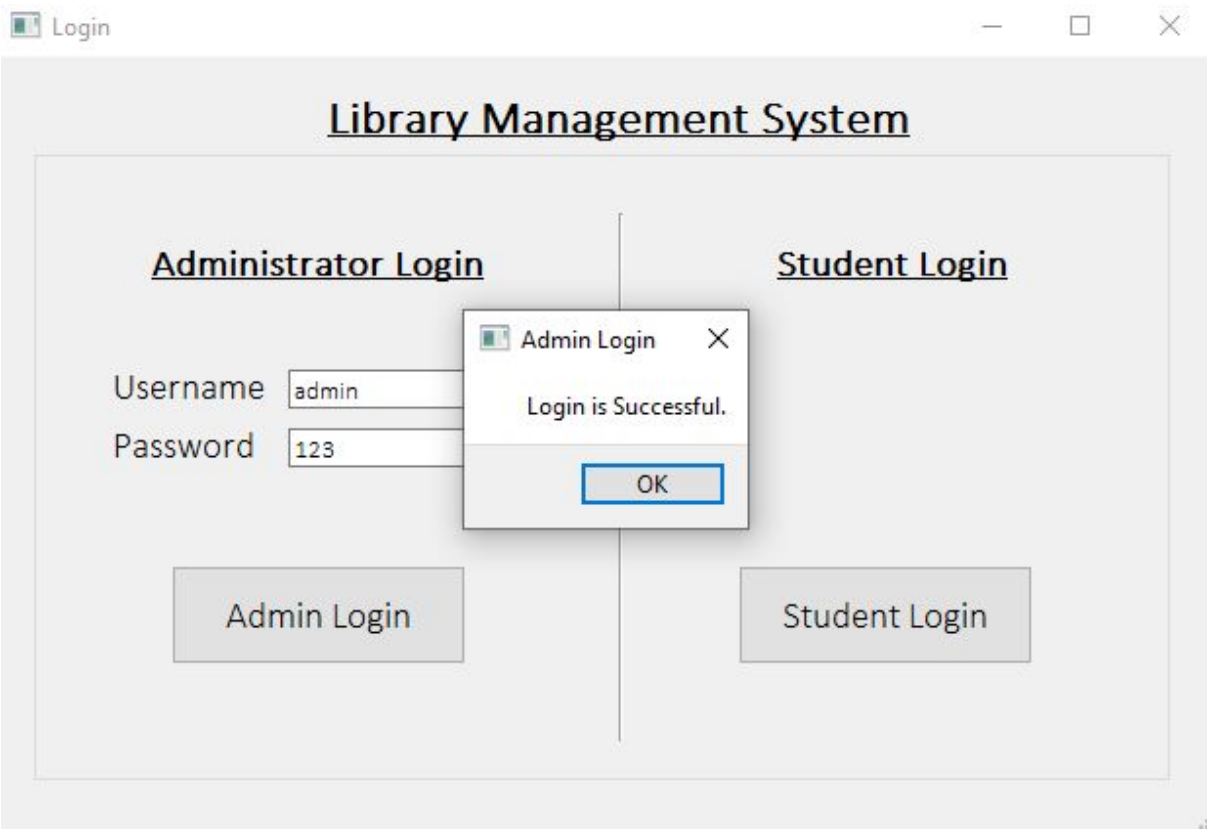
System Testing

The majority of our testing was done here to find and fix little bugs that couldn't be identified before. System testing was done through the GUI and after integrating the entire system.

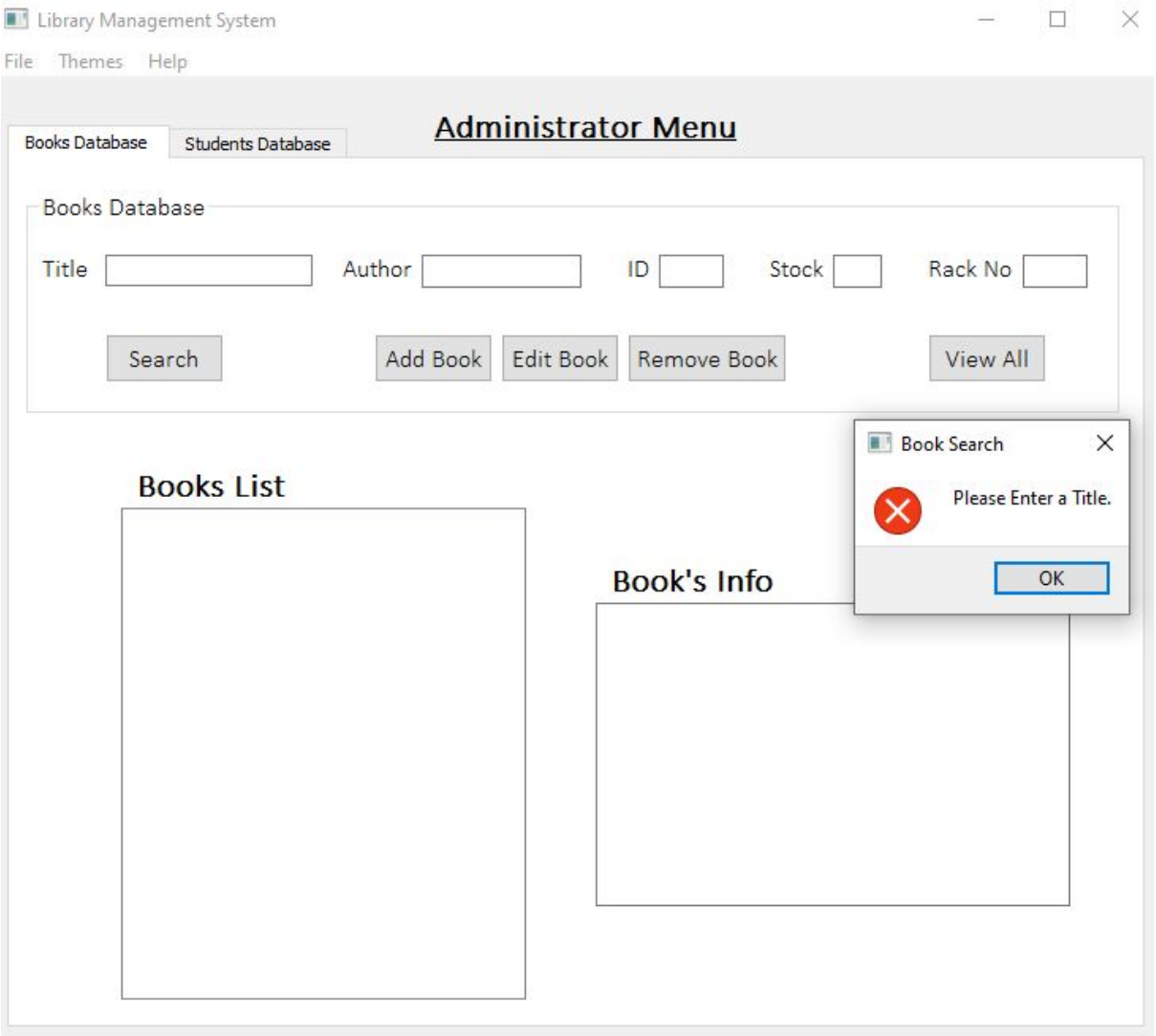
Failed Login Test



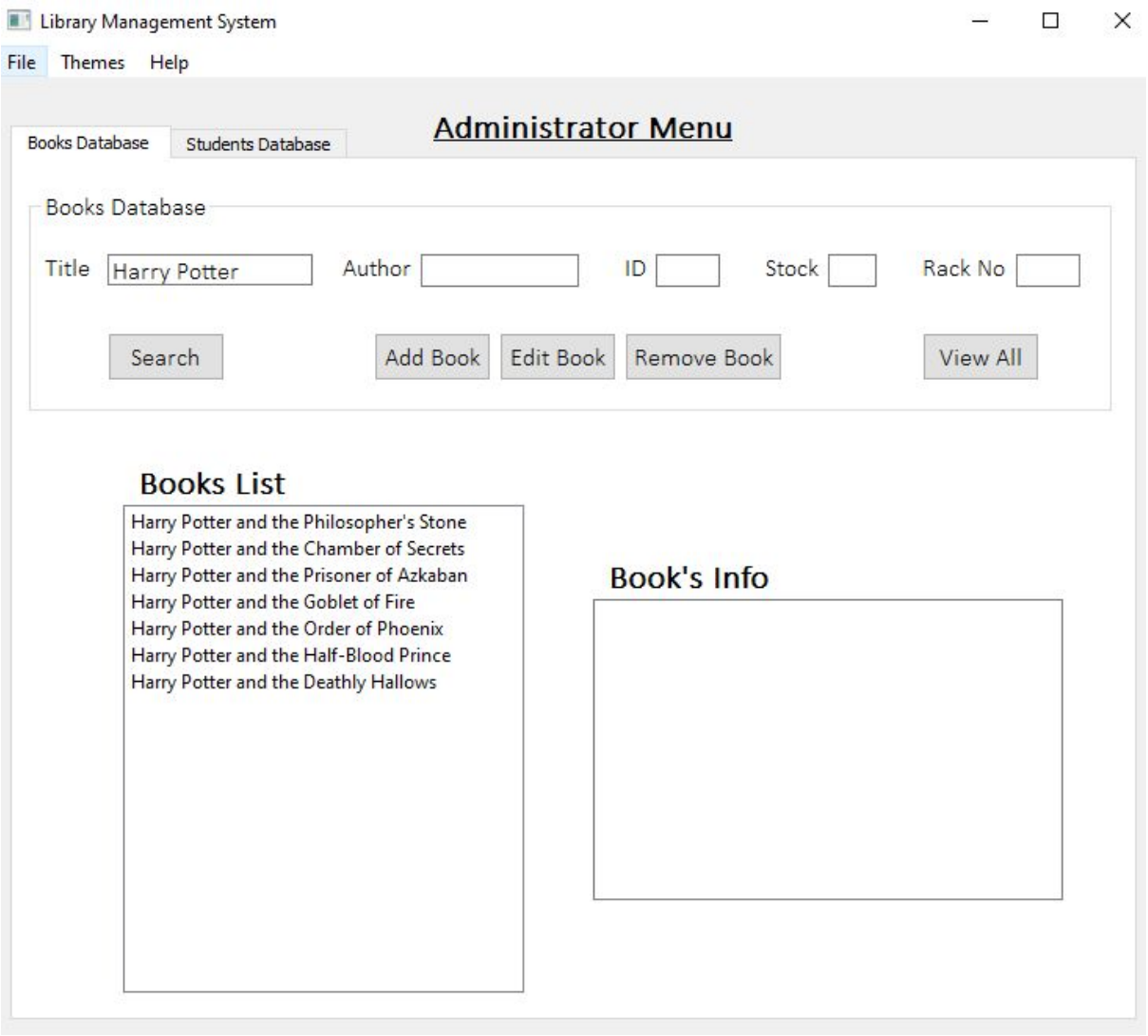
Successful Login Test



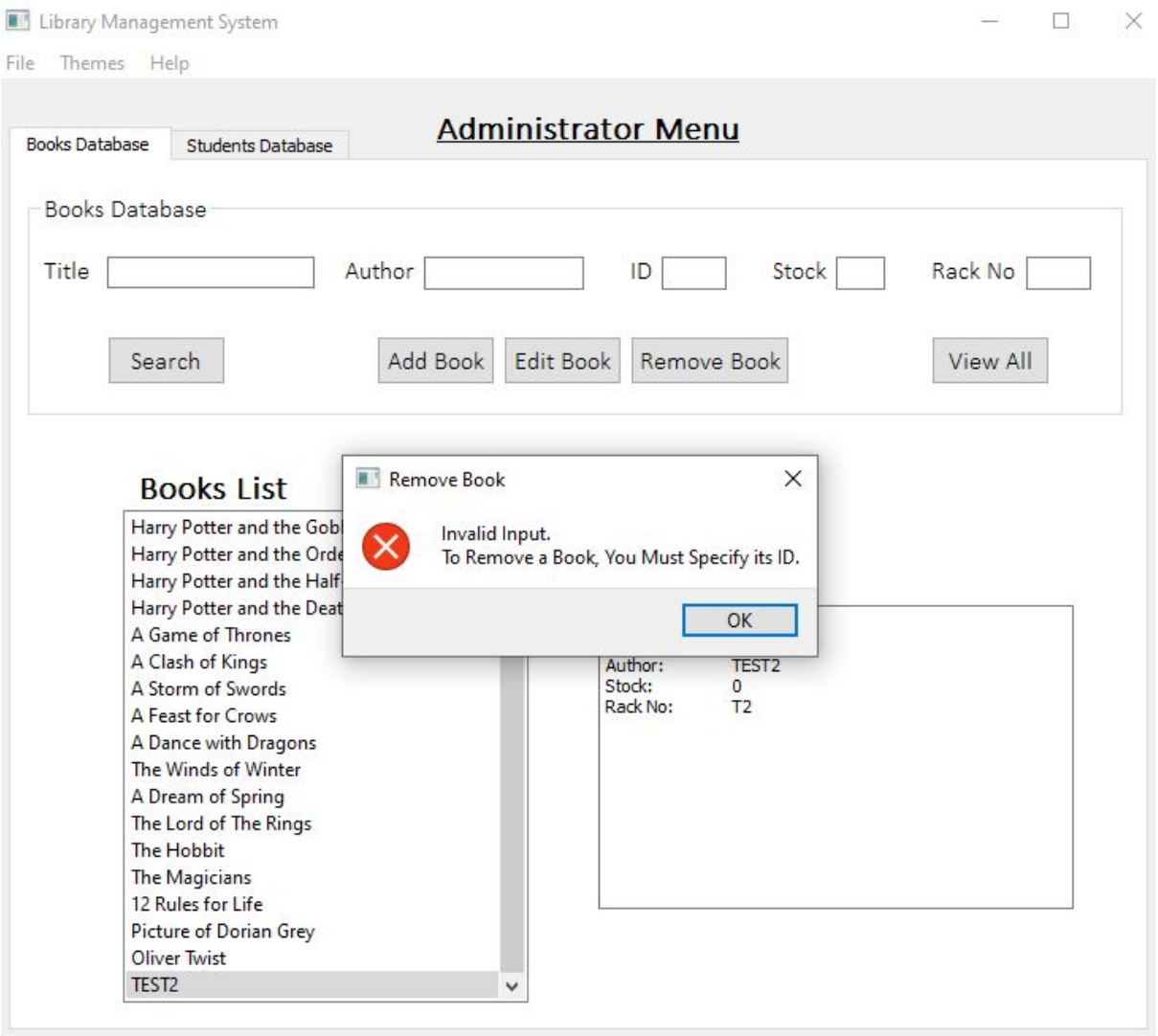
Search Fail Test



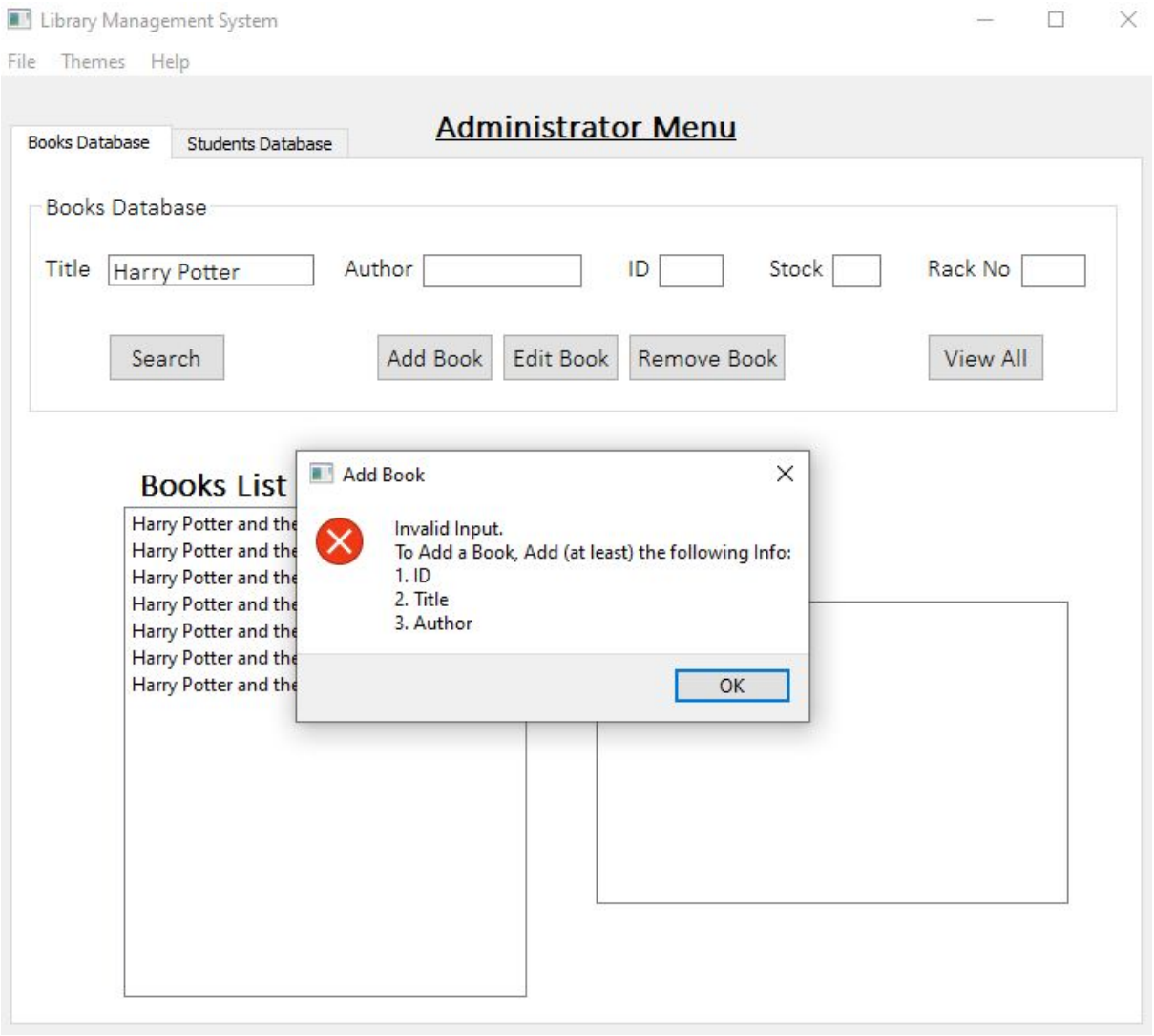
Search Successful Test



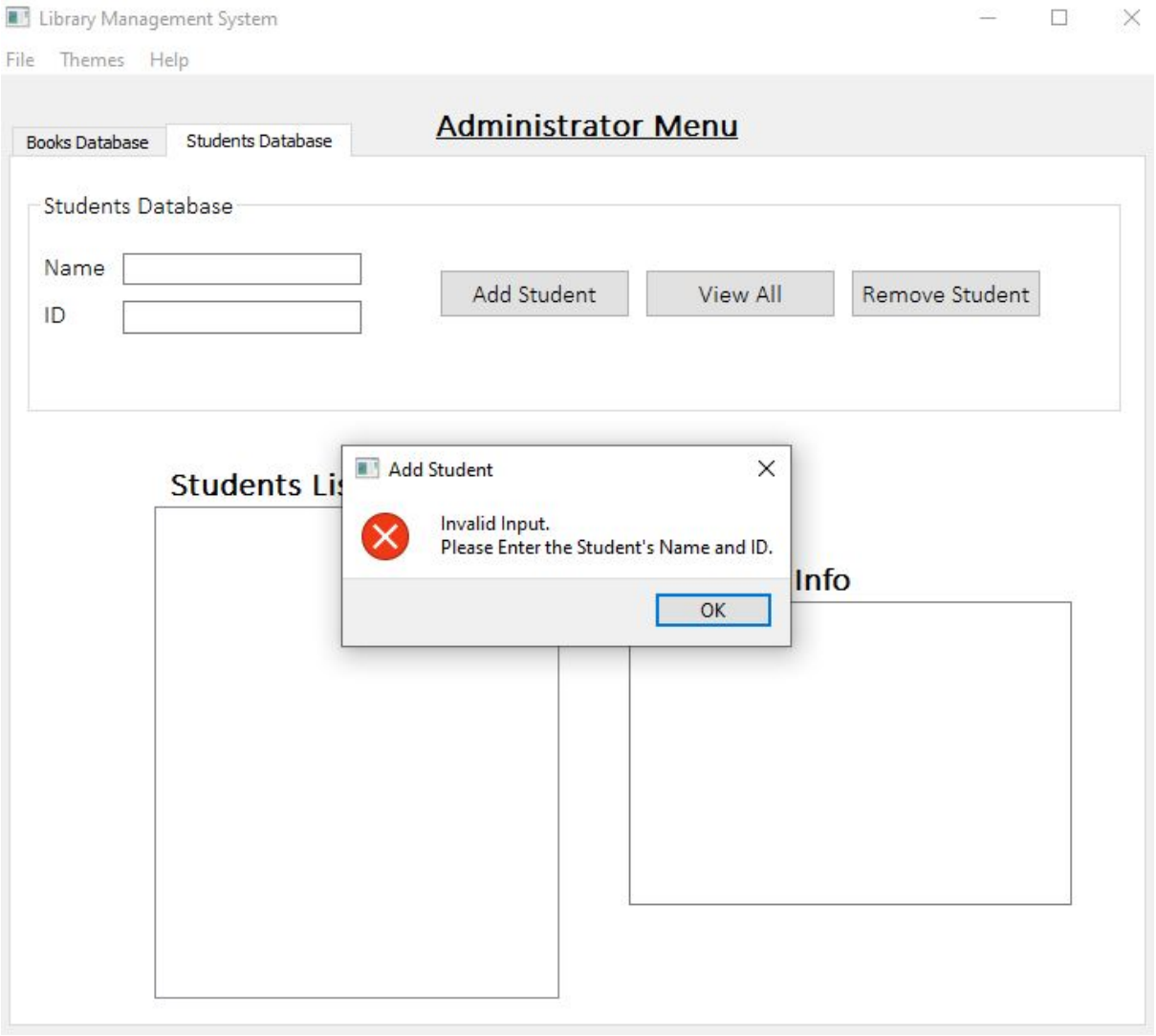
Remove Book Fail Test



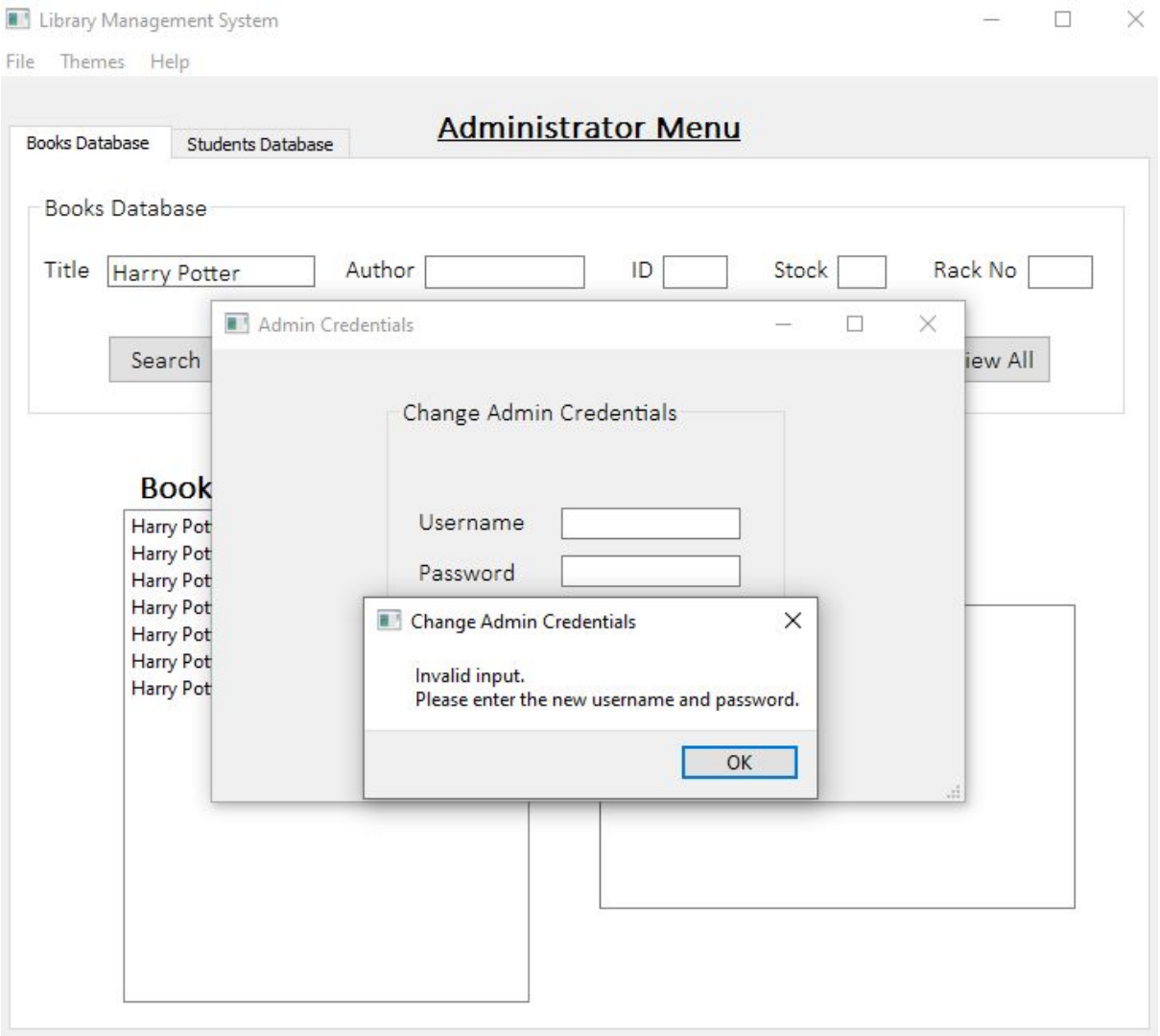
Add Book Fail Test



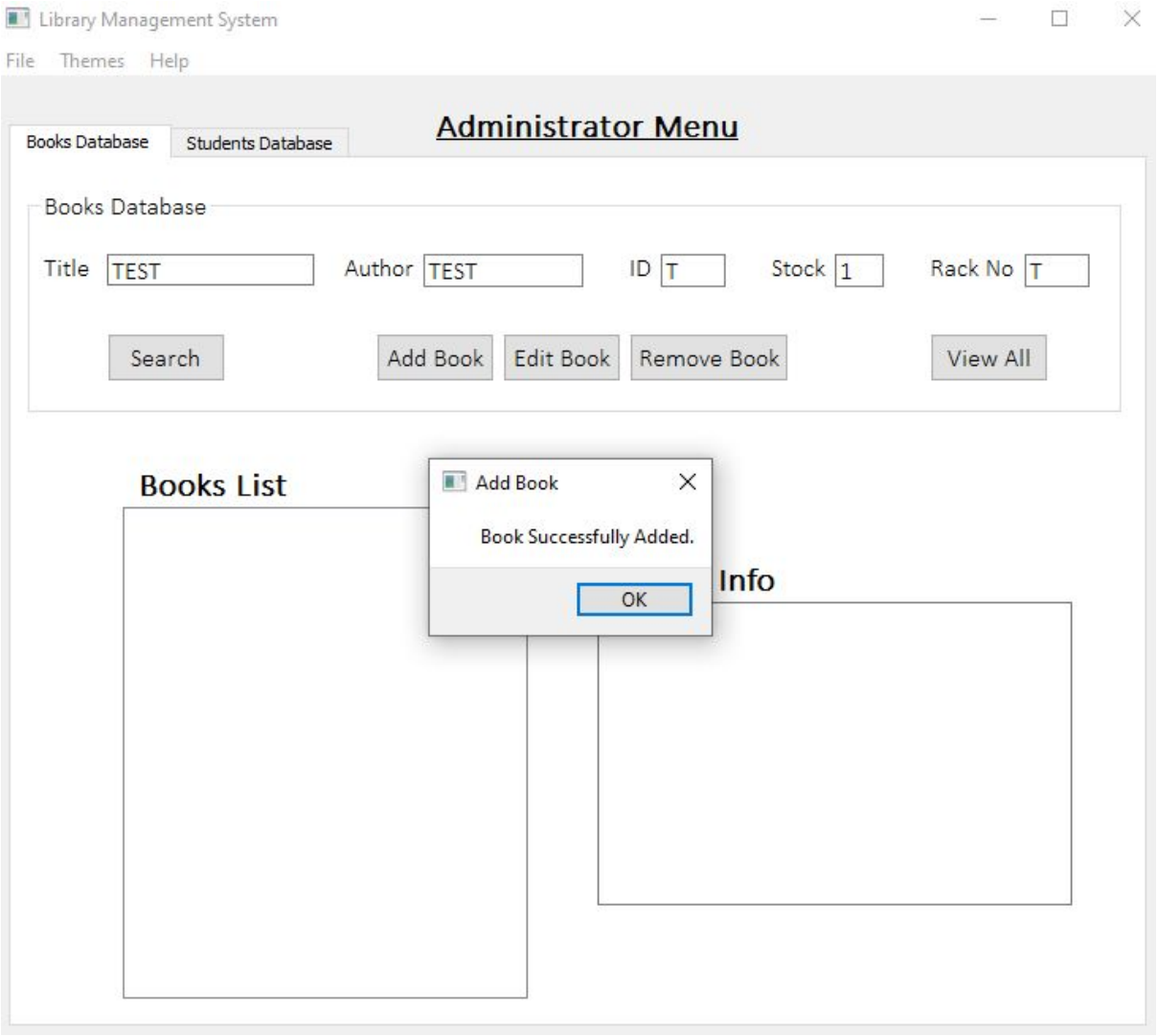
Add Student Fail Test

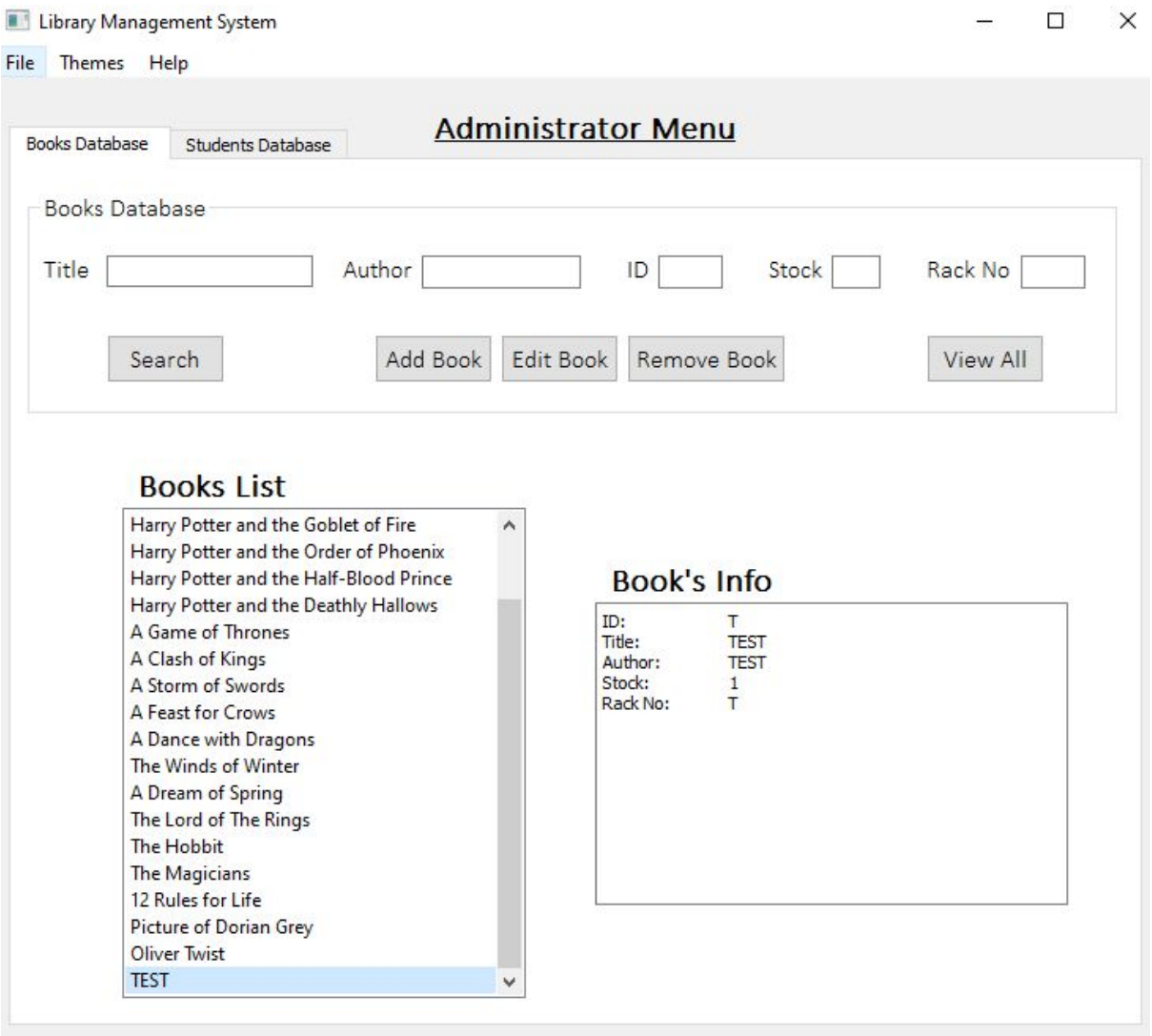


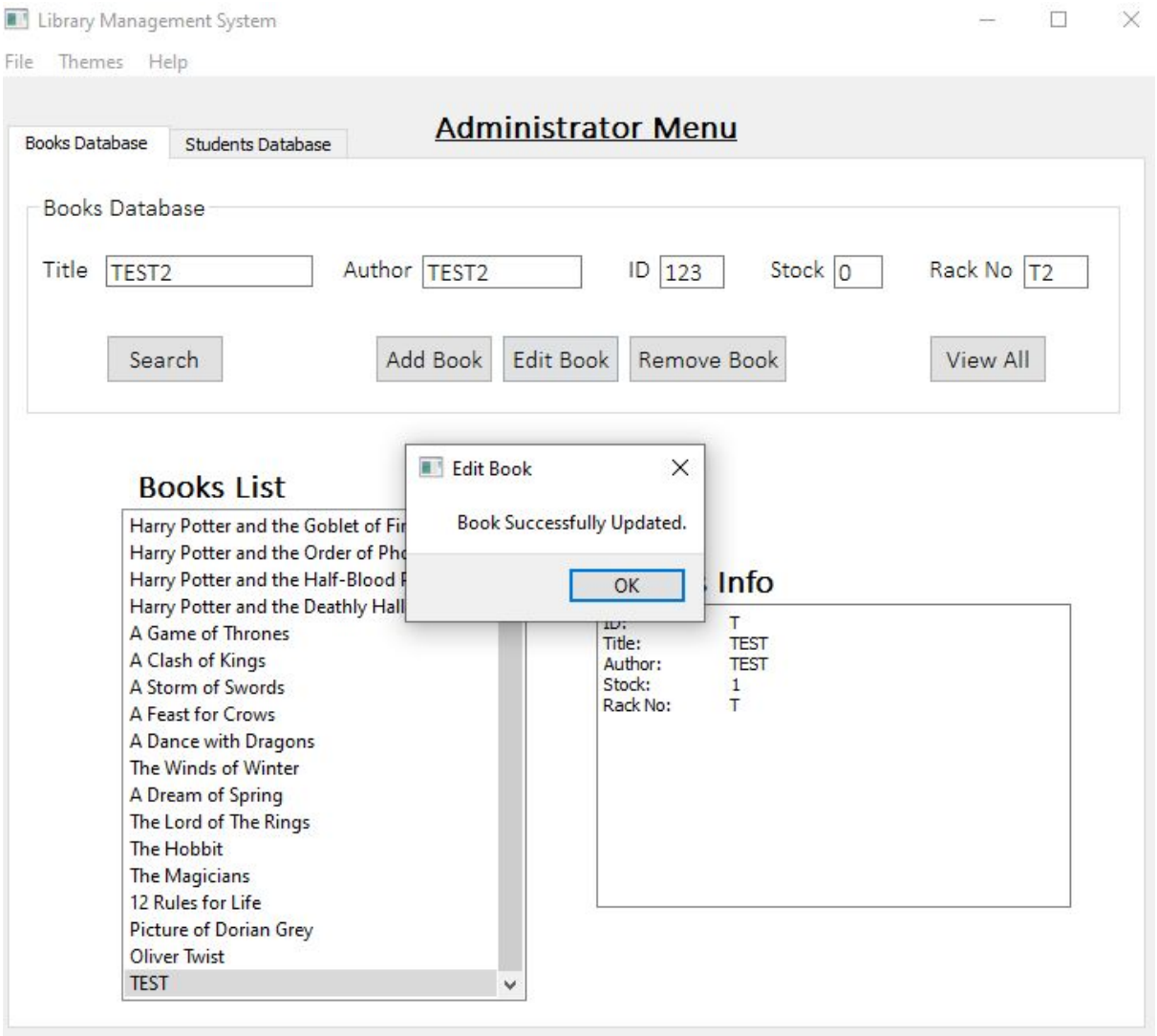
Change Credentials Fail Test

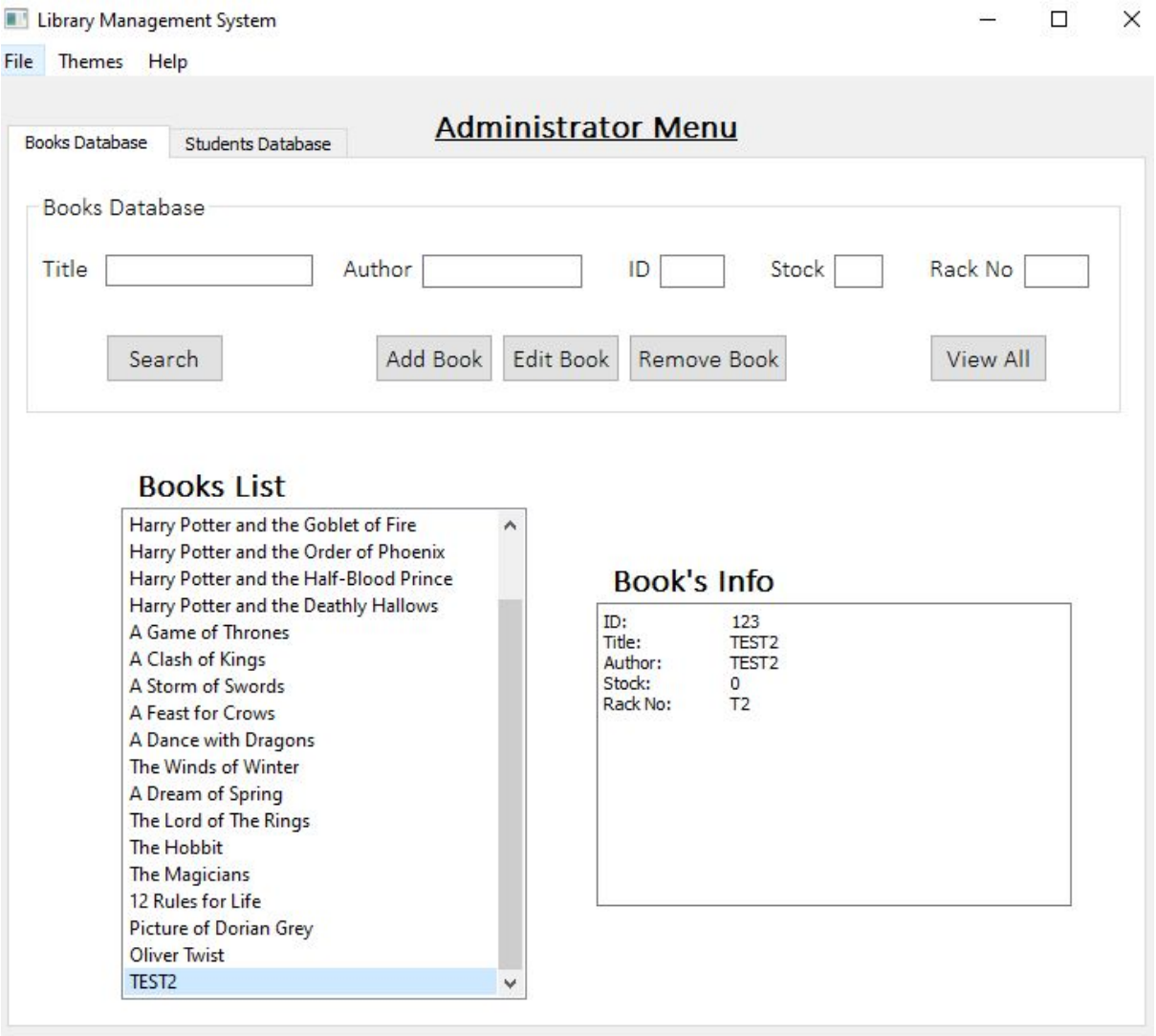


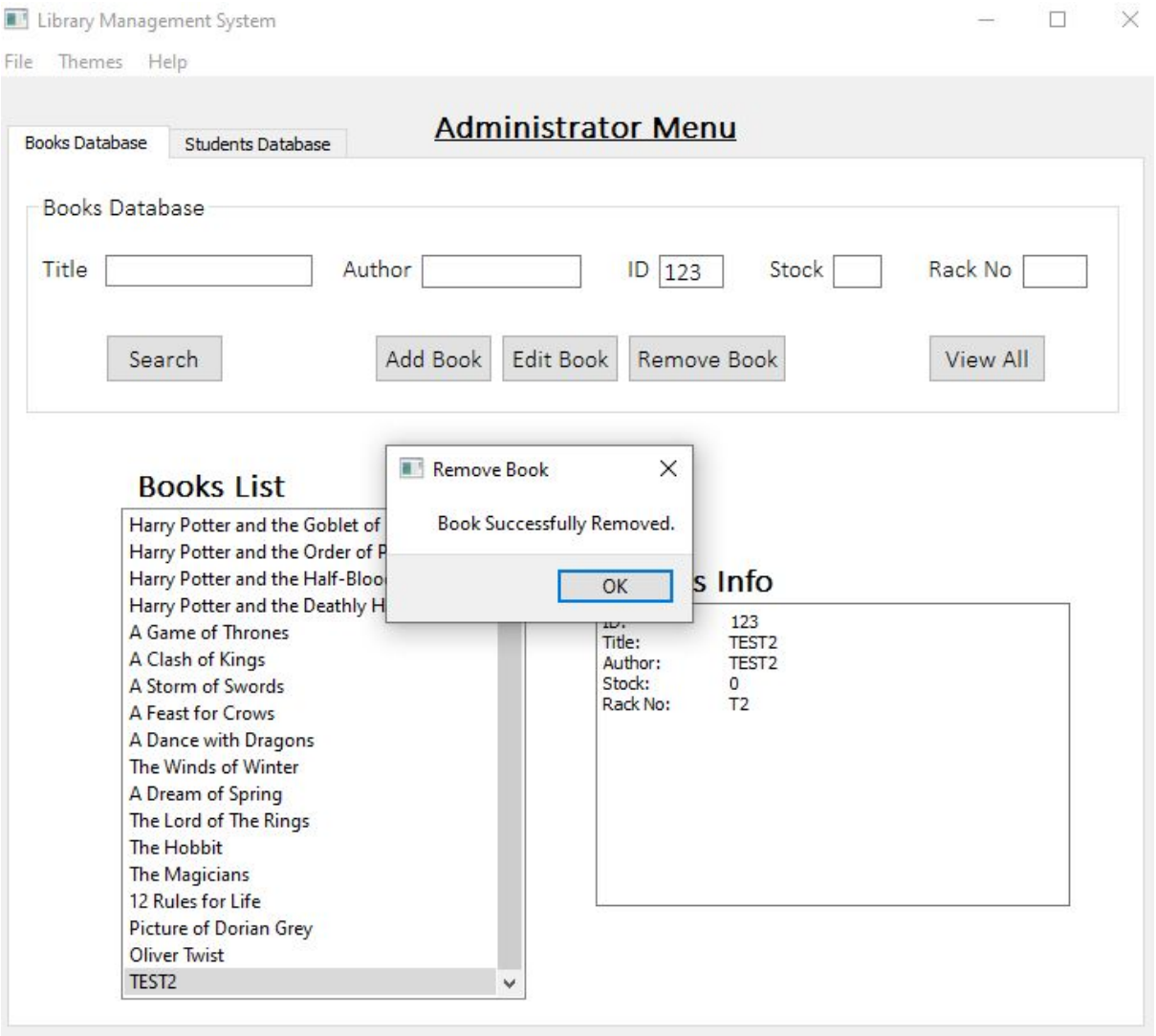
Success Test

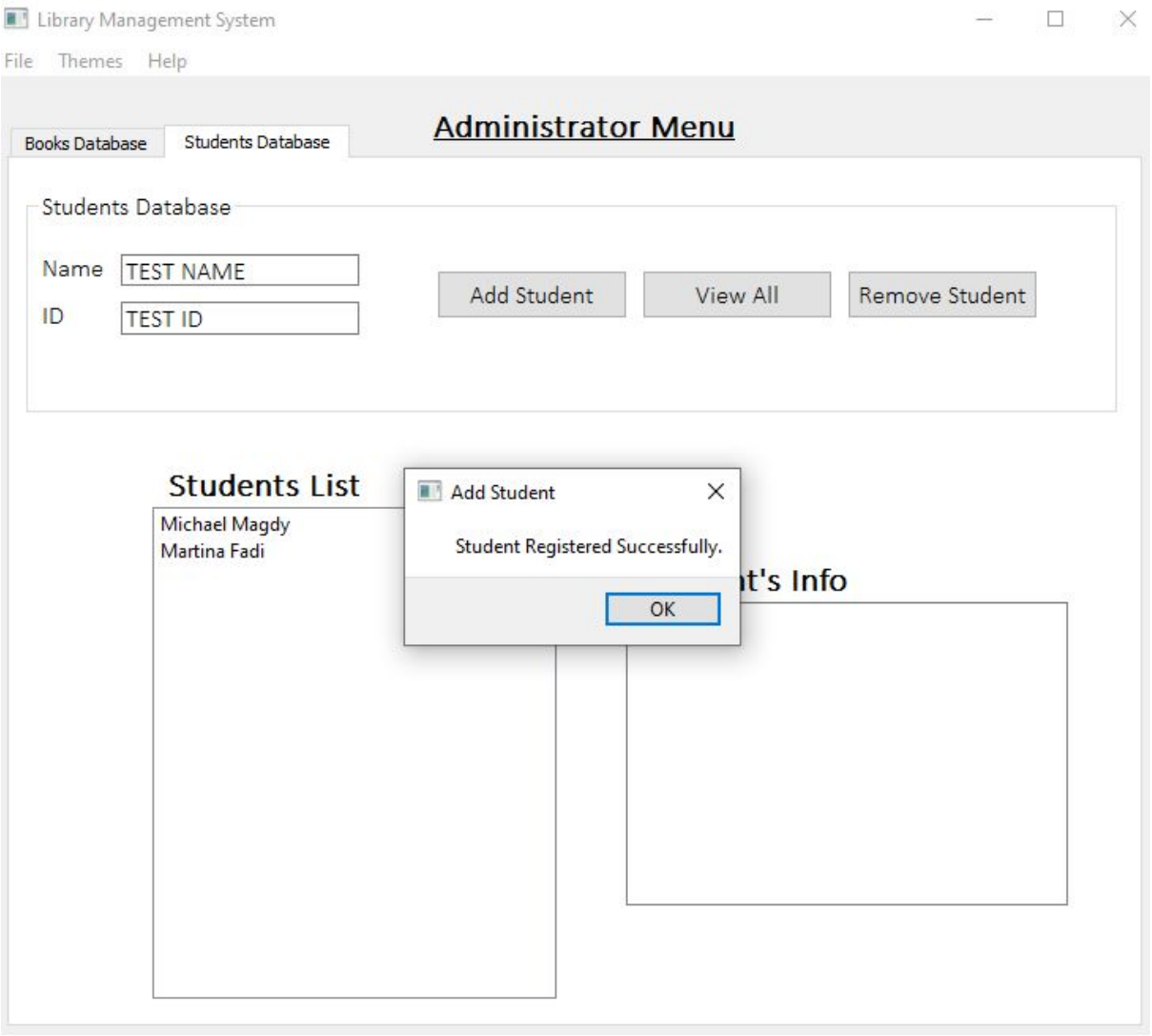


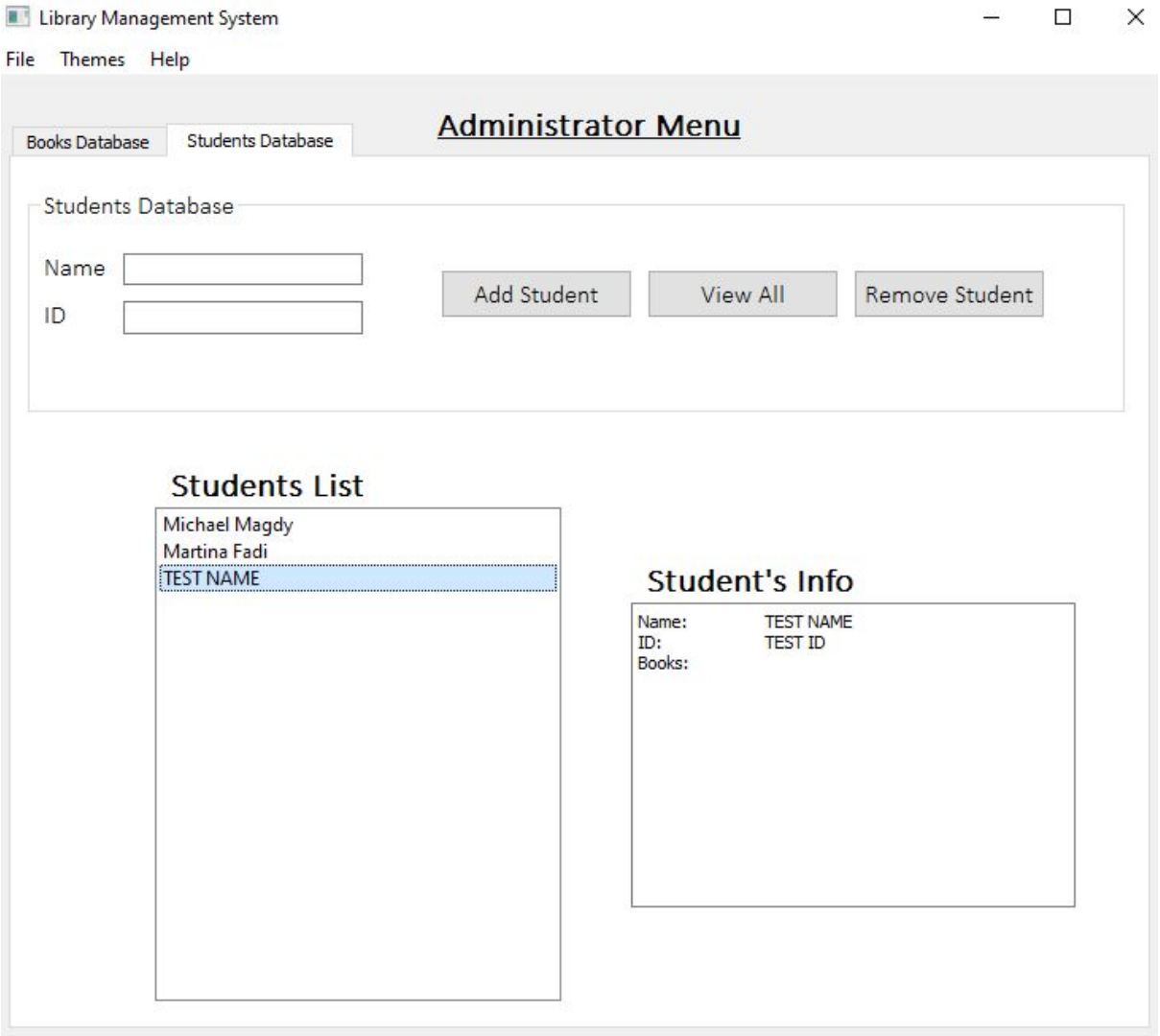

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Books Database
Title
Author
ID
Stock
Rack No
Search
Add Book
Edit Book
Remove Book
View All
Books List
Harry Potter and the Goblet of Fire
Harry Potter and the Order of Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows
A Game of Thrones
A Clash of Kings
A Storm of Swords
A Feast for Crows
A Dance with Dragons
The Winds of Winter
A Dream of Spring
The Lord of The Rings
The Hobbit
The Magicians
12 Rules for Life
Picture of Dorian Grey
Oliver Twist
TEST
Book's Info
ID: T
Title: TEST
Author: TEST
Stock: 1
Rack No: T

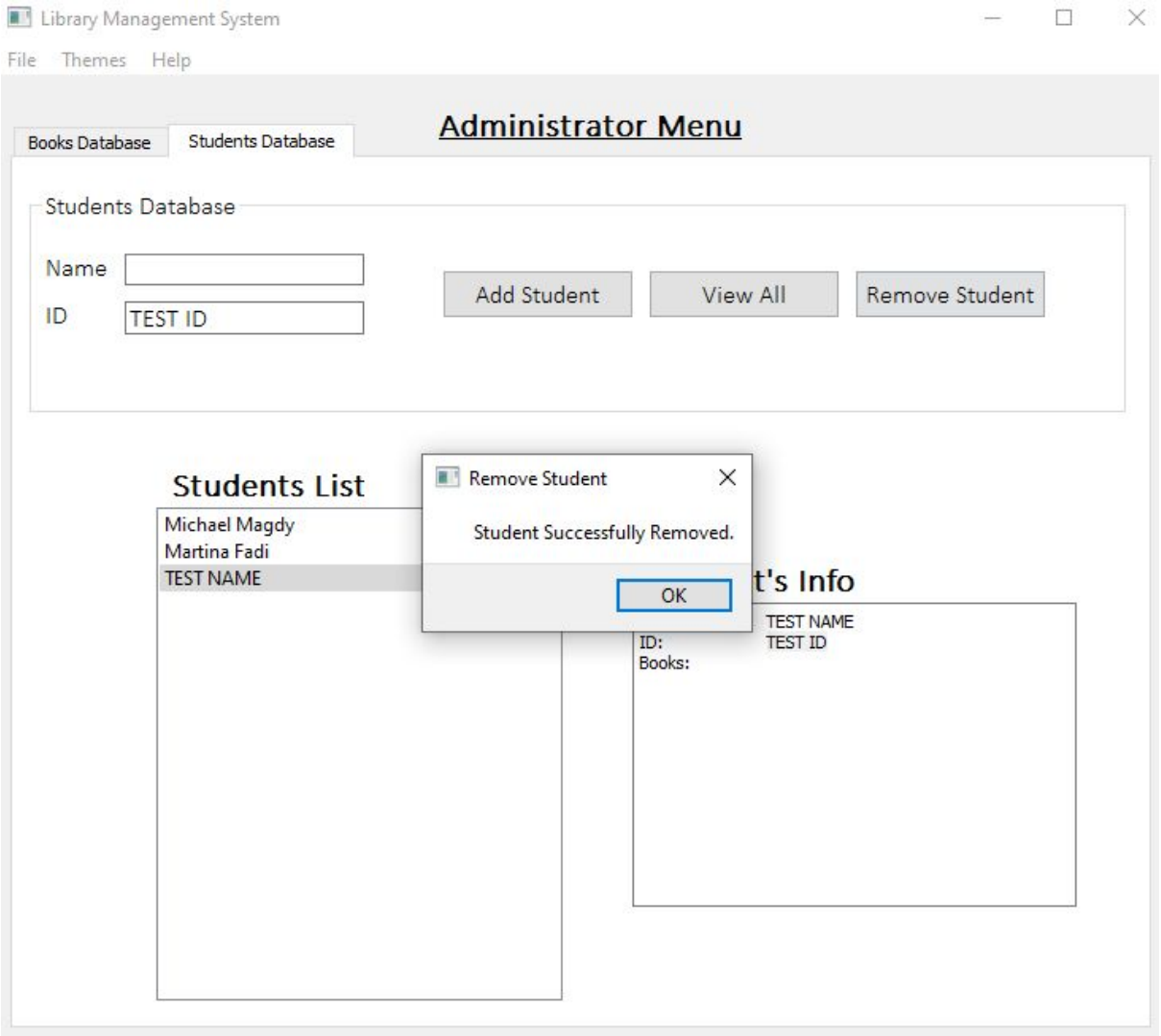

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Books Database
Title
TEST2
Author
TEST2
ID
123
Stock
0
Rack No
T2
Search
Add Book
Edit Book
Remove Book
View All
Books List
Harry Potter and the Goblet of Fir
Harry Potter and the Order of Phc
Harry Potter and the Half-Blood F
Harry Potter and the Deathly Hall
A Game of Thrones
A Clash of Kings
A Storm of Swords
A Feast for Crows
A Dance with Dragons
The Winds of Winter
A Dream of Spring
The Lord of The Rings
The Hobbit
The Magicians
12 Rules for Life
Picture of Dorian Grey
Oliver Twist
TEST
Edit Book
Book Successfully Updated.
OK
Info
T
Title:
TEST
Author:
TEST
Stock:
1
Rack No:
T

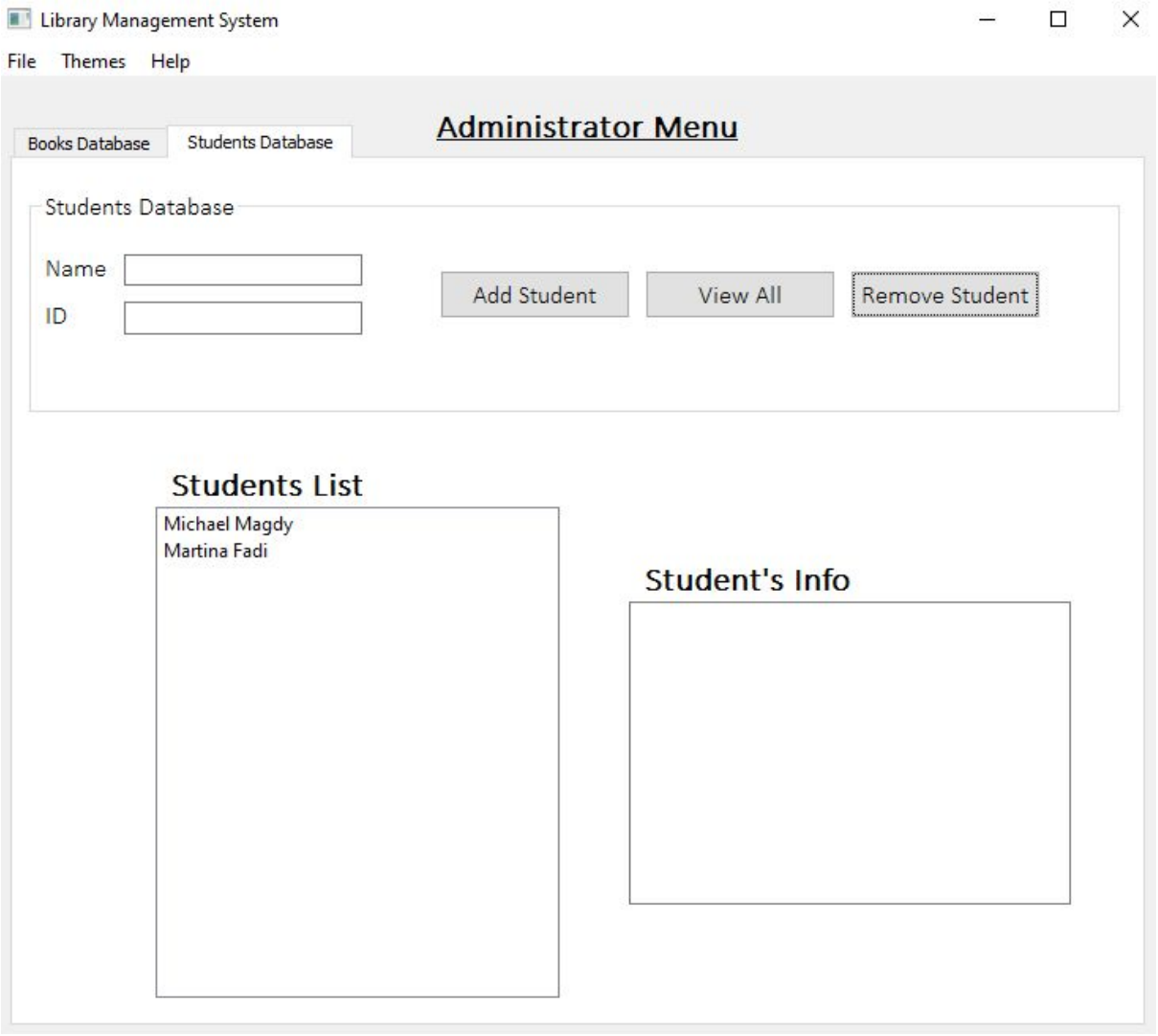

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Books Database
Title
Author
ID
Stock
Rack No
Search
Add Book
Edit Book
Remove Book
View All
Books List
Harry Potter and the Goblet of Fire
Harry Potter and the Order of Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows
A Game of Thrones
A Clash of Kings
A Storm of Swords
A Feast for Crows
A Dance with Dragons
The Winds of Winter
A Dream of Spring
The Lord of The Rings
The Hobbit
The Magicians
12 Rules for Life
Picture of Dorian Grey
Oliver Twist
TEST2
Book's Info
ID: 123
Title: TEST2
Author: TEST2
Stock: 0
Rack No: T2


Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Books Database
Title
Author
ID
123
Stock
Rack No
Search
Add Book
Edit Book
Remove Book
View All
Books List
Harry Potter and the Goblet of
Harry Potter and the Order of P
Harry Potter and the Half-Bloo
Harry Potter and the Deathly H
A Game of Thrones
A Clash of Kings
A Storm of Swords
A Feast for Crows
A Dance with Dragons
The Winds of Winter
A Dream of Spring
The Lord of The Rings
The Hobbit
The Magicians
12 Rules for Life
Picture of Dorian Grey
Oliver Twist
TEST2
Remove Book
Book Successfully Removed.
OK
s Info
123
Title:
TEST2
Author:
TEST2
Stock:
0
Rack No:
T2

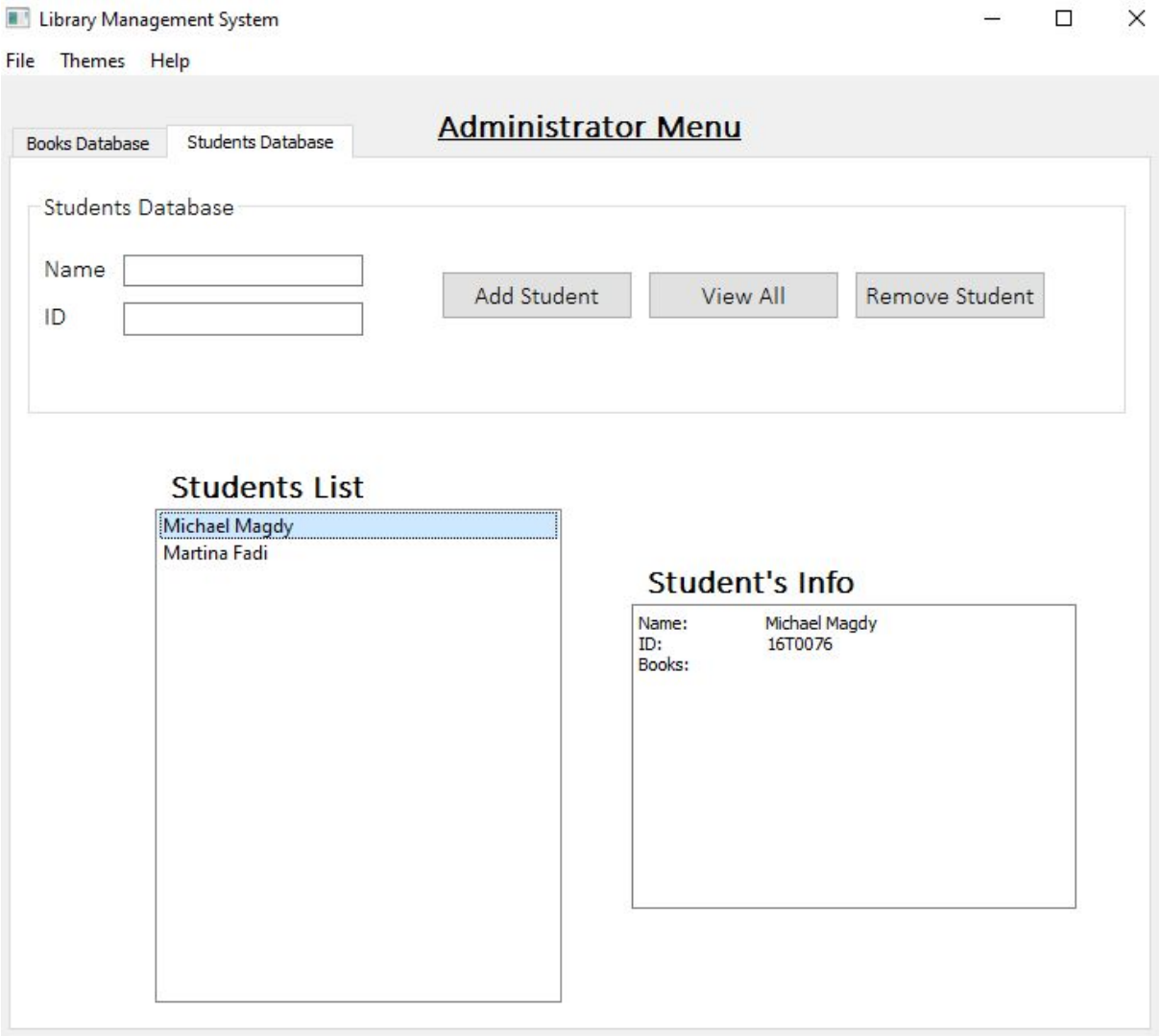

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Students Database
Name
TEST NAME
ID
TEST ID
Add Student
View All
Remove Student
Students List
Michael Magdy
Martina Fadi
Add Student
Student Registered Successfully.
OK
t's Info

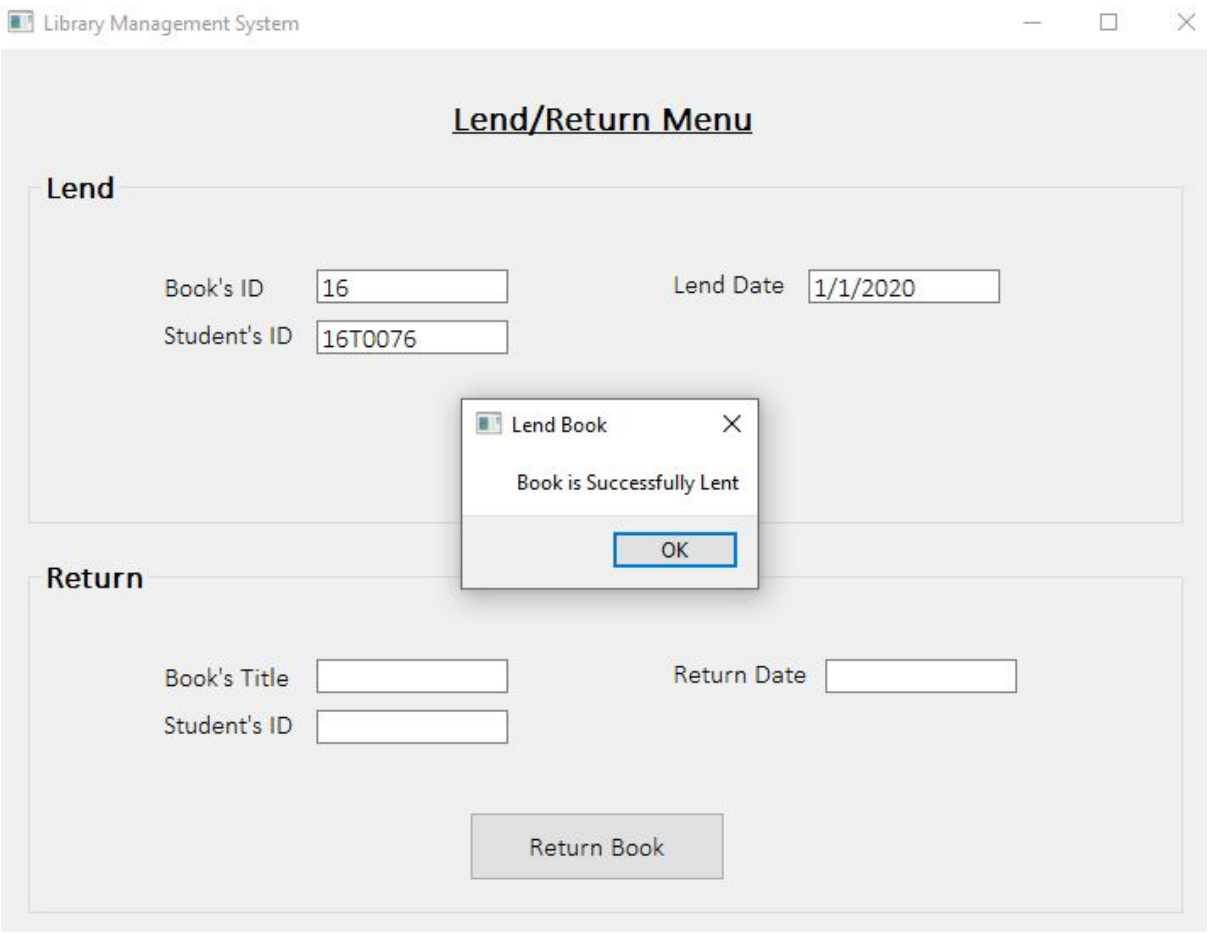

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Students Database
Name
ID
Add Student
View All
Remove Student
Students List
Michael Magdy
Martina Fadi
TEST NAME
Student's Info
Name:
TEST NAME
ID:
TEST ID
Books:

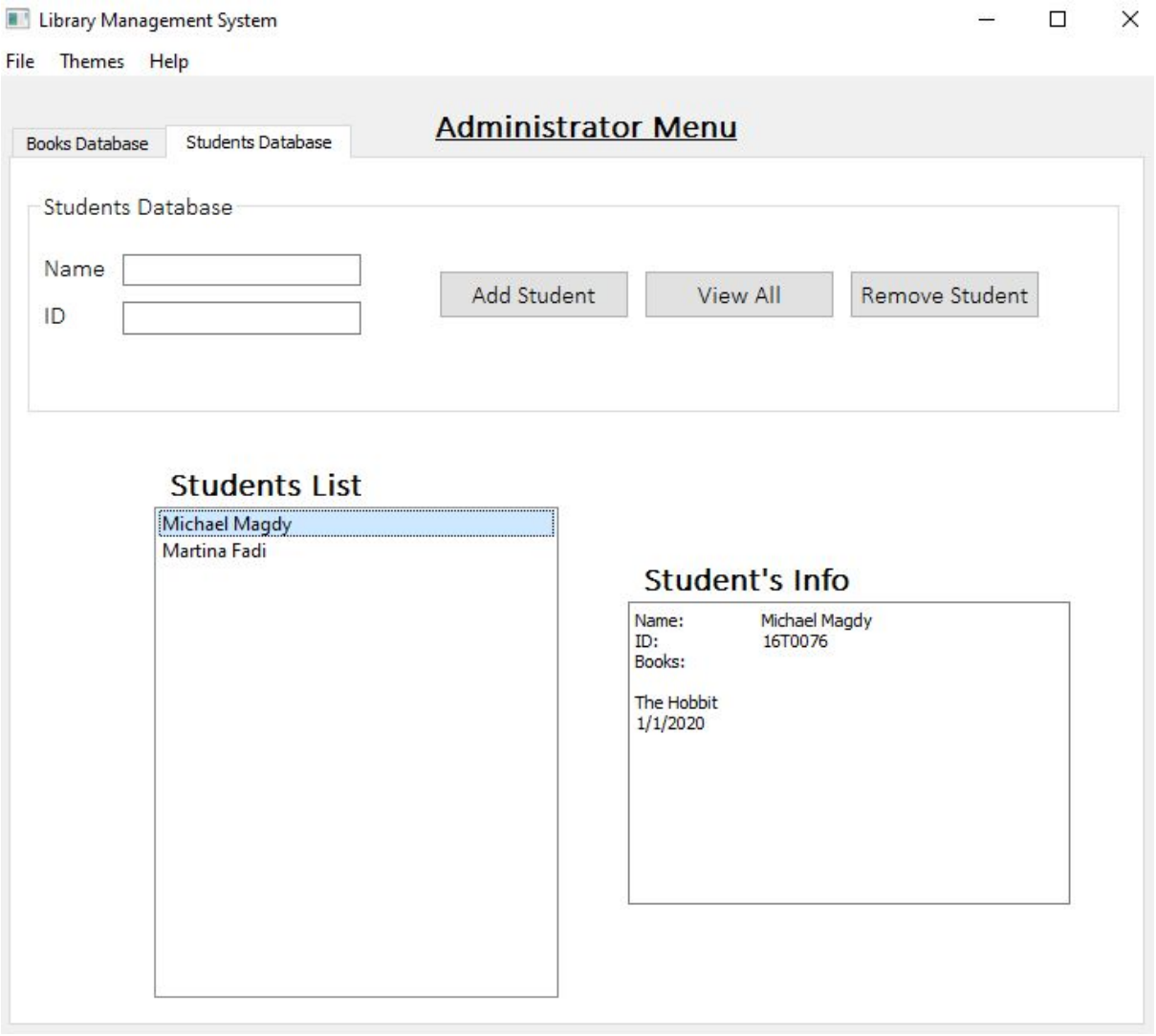

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Students Database
Name
ID
TEST ID
Add Student
View All
Remove Student
Students List
Michael Magdy
Martina Fadi
TEST NAME
Remove Student
Student Successfully Removed.
OK
t's Info
TEST NAME
TEST ID
ID:
Books:

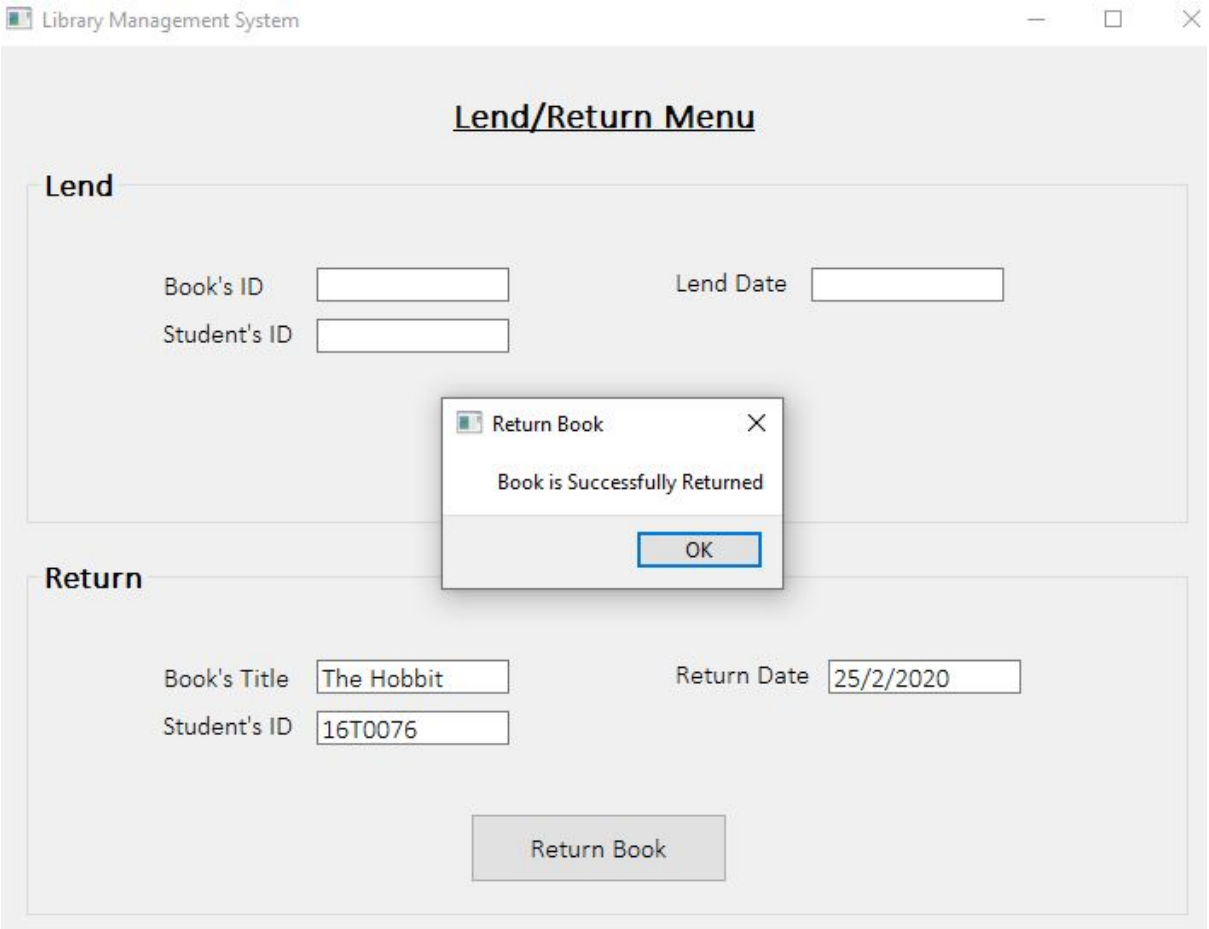

Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Students Database
Name
ID
Add Student
View All
Remove Student
Students List
Michael Magdy
Martina Fadi
Student's Info

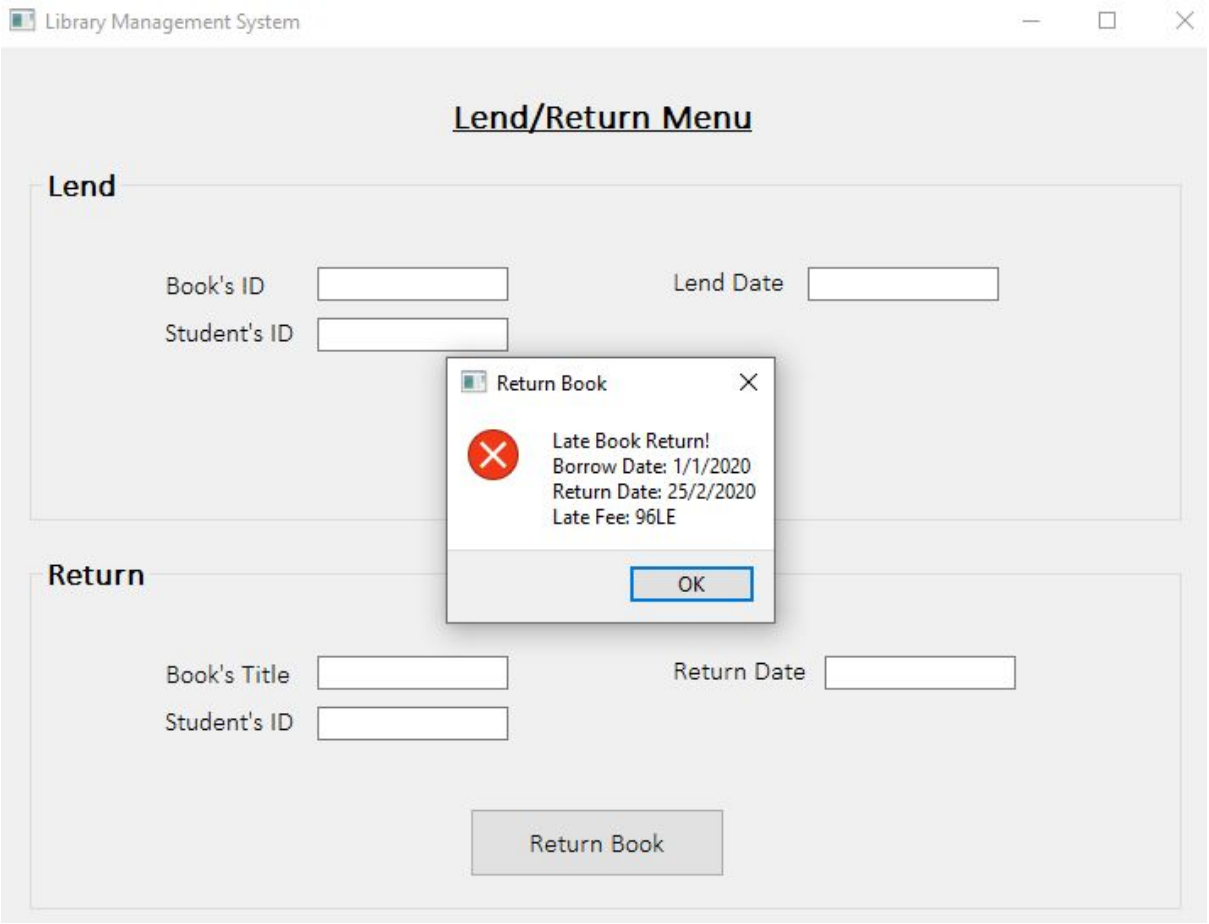
Lend/Return Testcase




Library Management System
Lend/Return Menu
Lend
Book's ID
16
Lend Date
1/1/2020
Student's ID
16T0076
Lend Book
Book is Successfully Lent
OK
Return
Book's Title
Return Date
Student's ID
Return Book


Library Management System
File
Themes
Help
Administrator Menu
Books Database
Students Database
Students Database
Name
ID
Add Student
View All
Remove Student
Students List
Michael Magdy
Martina Fadi
Student's Info
Name: Michael Magdy
ID: 16T0076
Books:
The Hobbit
1/1/2020


Library Management System
Lend/Return Menu
Lend
Book's ID
Student's ID
Lend Date
Return Book
Book is Successfully Returned
OK
Return
Book's Title
The Hobbit
Student's ID
16T0076
Return Date
25/2/2020
Return Book


Library Management System
Lend/Return Menu
Lend
Book's ID
Lend Date
Student's ID
Return Book
Late Book Return!
Borrow Date: 1/1/2020
Return Date: 25/2/2020
Late Fee: 96LE
OK
Return
Book's Title
Return Date
Student's ID
Return Book

16. Estimated Project Cost

In order to estimate the total cost of the project, we will use COCOMO-II model, there are several sub-models in COCOMO-II

such as:

- Application composition model: Used when software is composed from existing parts.
- Early design model: Used when requirements are available but design has not yet started.
- Post-architecture model: Used once the system architecture has been designed and more information about the system is available.

In our project we used the second and the third type only to estimate the total cost.

Early Design Model

Estimates can be made after the requirements have been agreed. Based on a standard formula for algorithmic models:

PM = A × Size^B × M

Where: A = 2.94

Size: (of the code) in terms (KLOC), and in our case it will be equal = 2.073.

B: It depends on 5 scale factors shown in the following table (each factor is rated takes values from 0 to 5):

Name of scale factor	Meaning	Rating in our project
Precedentedness	Reflects the previous experience of the organization with this type of project.	Low=4.96
Development flexibility	Reflects the degree of flexibility in the development process.	High=2.03
Architecture/risk resolution	Reflects the extent of risk analysis carried out.	Nominal=4.26
Team cohesion	Reflects how well the development team know each other and work together.	Very high=1.10

Process maturity	Reflects the process maturity of the organization.	Nominal=4.68

The formula is: B = (sumOfFactors/100) + 1.01

So that B will be equal = 1.1803.

M: Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform ... etc.

Name of multiplier	Refer to	Its value in our project (Rating)
RCPX	Product reliability and complexity	Nominal = 1
RUSE	The reuse required	High = 1.07
PDIF	Platform difficulty	Nominal = 1
PREX	Personal experience	Low = 4.96
PERS	Personal capability	High = 0.83
SCED	Required schedule	High = 1
FCIL	Team support facilities	High = 1.07

M = PERS×RCPX×RUSE×PDIF×PREX×FCIL×SCED = 4.71332.

So that, (Effort in person per month) PM will be equal

PM = 2.94 * (2.073^1.18) * 4.71332 = 32.7537.

We will assume that every engineer's salary per month will be 2k$.

So that the total cost will be equal = PM * salary=32.7537*2000 = 65.560K$.

Post-architecture Model

Name of multiplier	Refers to	Its value in our project (Rating)
RELY	Required software reliability	Nominal=1
DATA	Size of application database	Low=0.90
CPLX	Complexity of the product	Nominal=1
RUSE	The reuse required	High=1.07
DOCU	Extent of documentation required	High=1.11
TIME	Run-time performance constraints	Nominal=1
STOR	Memory constraints	Nominal=1
PVOL	Volatility of the virtual machine environment	Low=0.87
ACAP	Analyst capability	Nominal=1
PCON	Personal continuity	Nominal=1
PCAP	Programmer capability	High=0.88
PEXP	Programmer experience in project domain	High=0.91
LTEX	Language and tool experience	Nominal=1
AEXP	Analyst experience in project domain	Low=1.10
TOOL	Use of software tools	Low=1.09
SCED	Development schedule compression	High=1
SITE	Extent of multi-site working and quality of site communication	Nominal=1

By multiplying all the terms above, M=0.8929 and it will be multiplied by PREX (Personal experience) as a scale factor and as mentioned before, its rating is (Low=4.96).

So that total M will be equal 0.892*4.96=4.428

As before, A=2.94, B=1.18, Size=2.073.

So that, (Effort in person per month) PM will be equal = 2.94 * (2.073^1.18) * 4.428=30.771.

We will assume that every engineer's salary per month will be 2k$.

So that the total cost will be equal = PM * salary= 30.7710*2000= 61.154K$.

So that after calculating the total estimated cost by those 2 methods, we can assume that the total estimated cost of the project will be 65k$.

We use the following tables in order to calculate M(Multiplier) and Bin both methods:

These two tables are used to calculate M:

Drivers	Symbol	Scale Factors					
		Very Low	Low	Nominal	High	Very High	Extra High
PREC	SF1	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	SF2	5.07	4.05	3.04	2.03	1.01	0.00
RESL	SF3	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	SF4	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SF5	7.80	6.24	4.68	3.12	1.56	0.00
Drivers	Symbol	Effort Multiplier					
		Very Low	Low	Nominal	High	Very High	Extra High
Product Factors							
RELY	EM1	0.82	0.92	1.00	1.10	1.26	-
DATA	EM2	-	0.90	1.00	1.14	1.28	-
CPLX	EM3	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	EM4	-	0.95	1.00	1.07	1.15	1.24
DOCU	EM5	0.81	0.91	1.00	1.11	1.23	-
Platform Factors							
TIME	EM6	-	-	1.00	1.11	1.29	1.63
STOR	EM7	-	-	1.00	1.05	1.17	1.46
PVOL	EM8	-	0.87	1.00	1.15	1.30	-
Personnel Factors							
ACAP	EM9	1.42	1.22	1.00	0.85	0.71	-
PCAP	EM10	1.34	1.16	1.00	0.88	0.76	-
PCON	EM11	1.29	1.10	1.00	0.90	0.81	-
APEX	EM12	1.22	1.10	1.00	0.88	0.81	-
PLEX	EM13	1.19	1.12	1.00	0.91	0.85	-
LTEX	EM14	1.20	1.10	1.00	0.91	0.84	-
Project Factors							
TOOL	EM15	1.17	1.09	1.00	090	0.78	-
SITE	EM16	1.22	1.09	1.00	0.93	0.86	0.80
SCED	EM17	1.43	1.14	1.00	1.00	1.00	-

COCOMO II Post Architecture Effort Multipliers:17 cost drivers

Product attributes			
RELY	Required system reliability	DATA	Size of database used
CPLX	Complexity of system modules	RUSE	Required percentage of reusable components
DOCU	Extent of documentation required		
Computer attributes			
TIME	Execution time constraints	STOR	Memory constraints
PVOL	Volatility of development platform		
Personnel attributes			
ACAP	Capability of project analysts	PCAP	Programmer capability
PCON	Personnel continuity	AEXP	Analyst experience in project domain
PEXP	Programmer experience in project domain	LTEX	Language and tool experience
Project attributes			
TOOL	Use of software tools	SITE	Extent of multi-site working and quality of site communications
SCED	Development schedule compression		

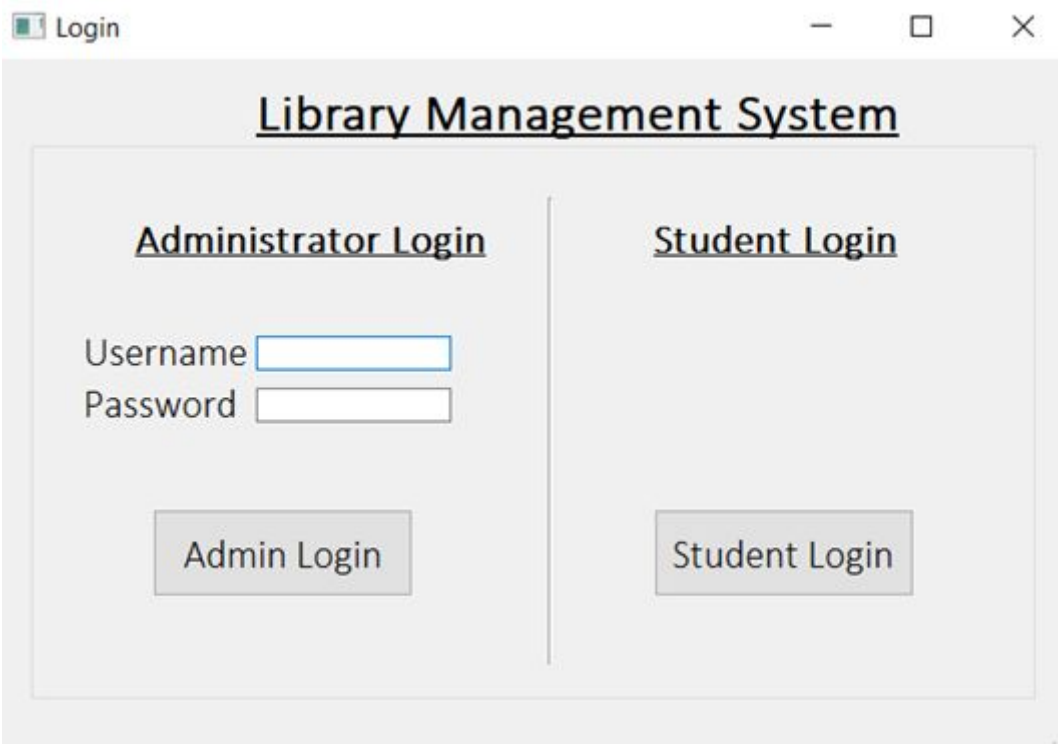
Software Engineering SW Cost Estimation Slide 81

This table is used to calculate B:

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

17. User Guide

Login Window



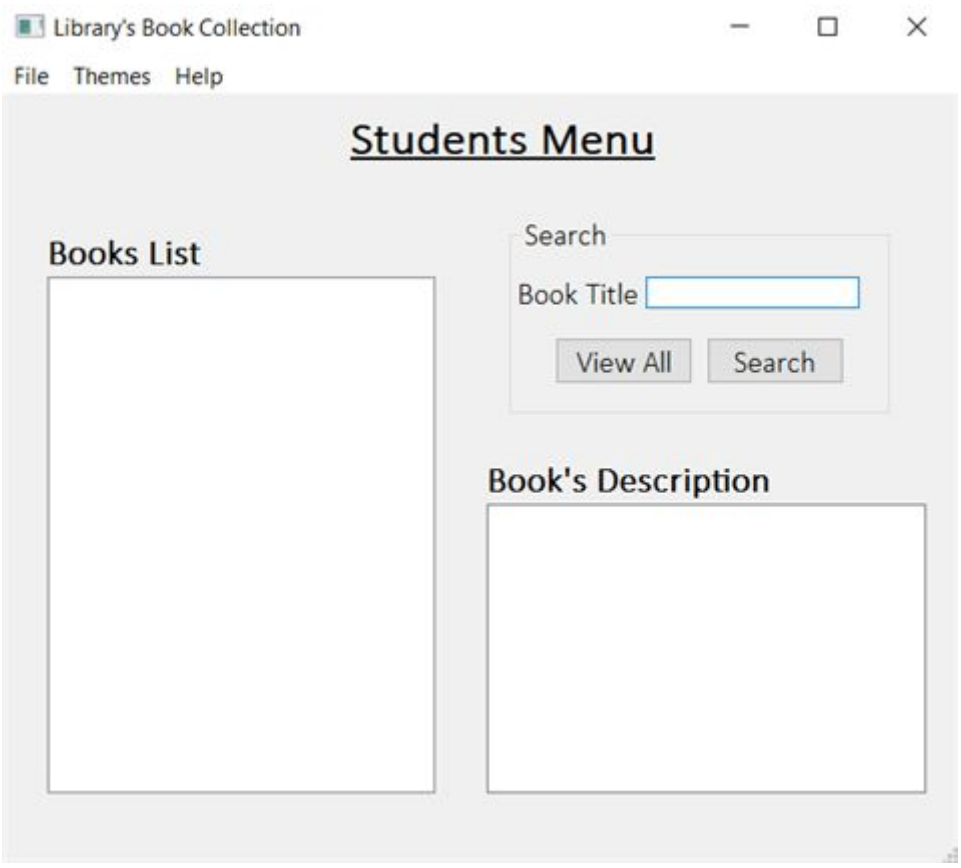
Button list:

- ➢ Admin login — Login as admin (must enter correct credentials).
- ➢ Student login — Login as a student (freely accessible to anyone).

User inputs:

- ➢ Username — Admin's Username.
- ➢ Password — Admin's Password.

Students Window



Button list

- **Search**
  - Search for a book by title.
- **View All**
  - Display all books in the library's collection.

User inputs

- **Book Title**
  - Title of the book that the user wishes to search for.

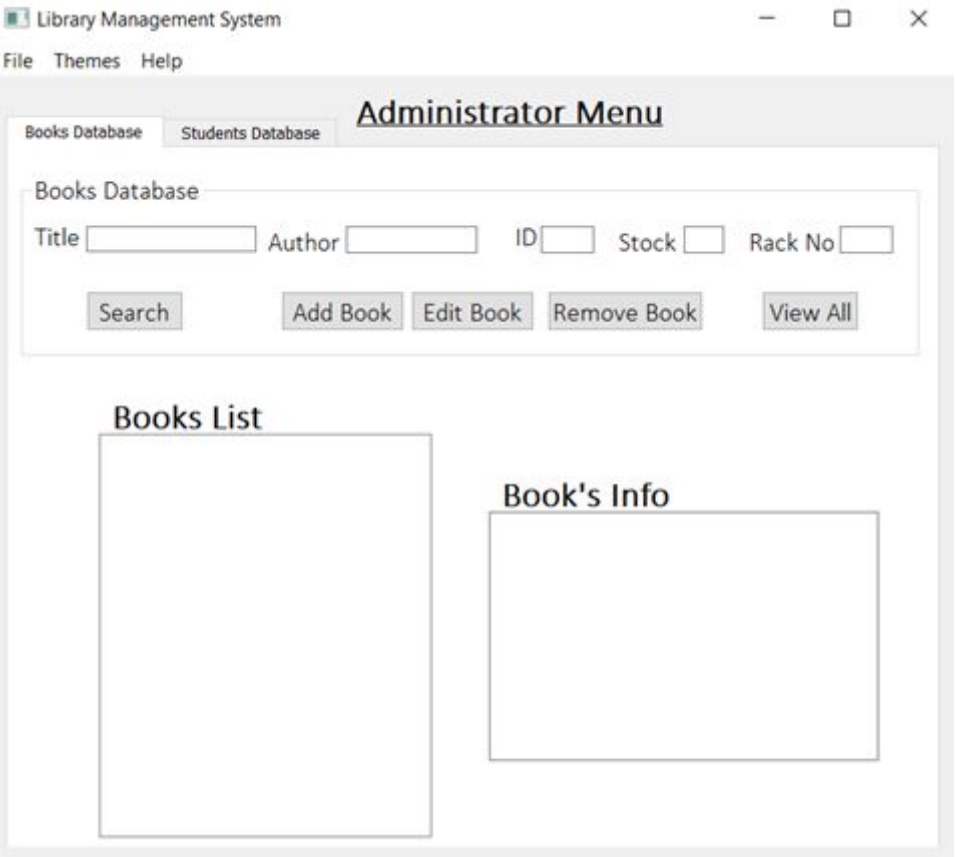
Output for user

- **Books List**
  - List of all books in the library, the user can select a book from the list to view additional information about it.
- **Book's Description**
  - Information about the selected book.

Administrator Window

Menu Bar Actions:

- File
  - **Lend/Return Books**
    - Open the library transactions menu.
  - **Change Admin Credentials**
    - Opens a menu allowing the admin to change their credentials.
  - **Logout**
    - Returns librarian to login window.
  - **Exit**
    - Exits the program.
- Themes
  - Allows librarians to change the colour scheme of the system.
- Help
  - Shows the librarian a brief overview of the software and how to use it.



As discussed previously, the administrator window is subdivided into two tabs.

Books Database Tab

Buttons:

- **Search**
  - Allows the librarian to search for a book in the database, provided they entered a title.
- **Add Book**
  - Allows the librarian to add a new book to the database, provided they entered sufficient information about the book.
- **Edit Book**
  - Allows the librarian to edit an already existing book in BDB by changing its title, ID, author, stock or rack no.
- **Remove Book**
  - Allows the librarian to remove a book from the database, provided they entered the book ID.
- **View All**
  - Displays all available books in the library's collection.

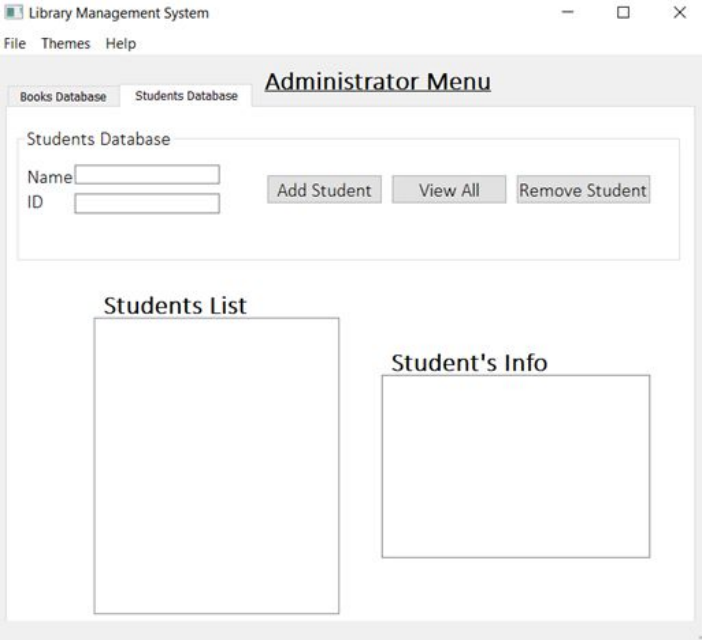
User Inputs:

- **Title**
  - Title of the book that the user wishes to add, edit, or search for.
- **Author**
  - Name of the book's author.
- **ID**
  - Book's unique identification number.
- **Stock**
  - Number of copies of any book in the library.
- **Rack no**
  - Physical location of the book in the library.

Output for user

- **Books List**
  - List of all books in the library, the user can select a book from the list to display additional information about it.
- **Book's Description**
  - Information about the selected book.

Students Database Tab



Button list

- **Add Student**
  - Allows the librarian to add a new student to the SDB.
- **View All**
  - Allows the librarian to view all students currently registered in the library.
- **Remove Student**
  - remove a student from the database.

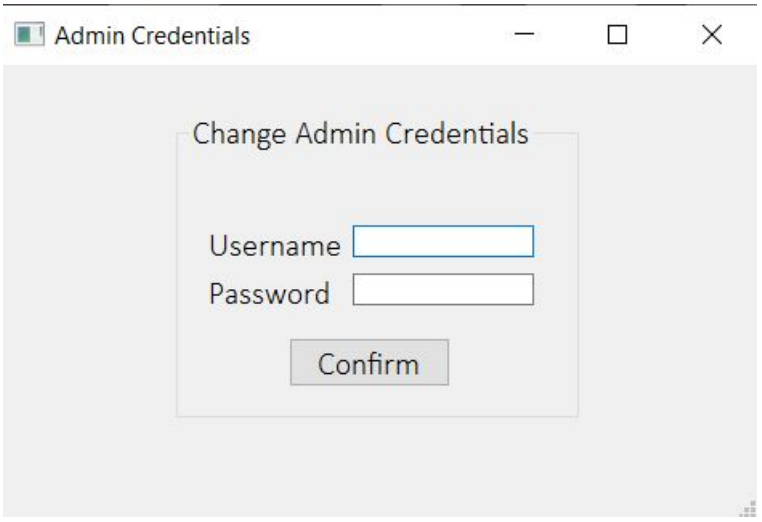
User inputs

- **Name**
  - Name of student the user wishes to add.
- **ID**
  - ID of student the user wishes to add/remove.

Output for user

- **Students List**
  - List of all students registered in the library, users can choose a student from the list to view additional information about them.
- **Students' Info**
  - Information about the selected student.

Change Admin Credentials Window



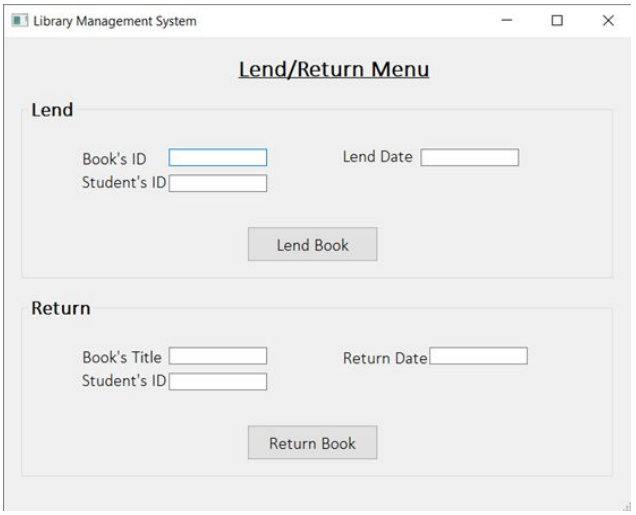
Button list

- **Confirm**
  - Confirm changing the old username and password to the new ones.

User inputs

- **Username**
  - Admin's new username.
- **Password**
  - Admin's new password.

Lend/Return Menu



Button list

- **Lend Book**
  - Stores in the database which student borrowed the book and the date, provided the student didn't exceed the borrowing limit.
- **Return Book**
  - Stores in the database that the student returned the book and calculates late fees if needed.

User inputs

Lend Section

- **Book's ID**
  - ID of the book that the student wants to borrow.
- **Lend Date**
  - Borrowing date (must be written in the form "dd/mm/yyyy")
- **Student's ID**
  - ID of the student who wants to borrow the book.

Return Section

- **Book's ID**
  - ID of the book that the student wants to return.
- **Return Date**
  - Return date (must be written in the form "dd/mm/yyyy").
- **Student's ID**
  - ID of the student who wants to return the Book.