Ain Shams University

Faculty of Engineering

Electrical Department

CSE 321: Software Engineering

Software Project Documentation

Library Management System

Submitted by:

Amr El-Sersy	1600942
Mahmoud Osama Mahmoud Soliman	1601294
Mahmoud Mohamed Anwar	1601323
Marwan Mahmoud Badawy	1601346
Mostafa Ahmed Mohamed Mohamed	1601397
Mostafa Amgad Ahmed Saeed	1601401
Mostafa Gamal Zakaria	1601405

Submitted to:

Dr. Gamal Abdel Shafy

M. Ahmed Mostafa

Abstract:

In this project, we created an online library management system. It’s a whole new experience in the library world: software built to handle the primary housekeeping functions of a library. It provides several services for different users including students, librarians and publishers. A user can create a library account, include his personal and professional information, access his navigation history and be presented with some recommended books based on his favorites. Also, a student can borrow books online, he can search for his target using any information he has (name, type, author, version …etc.), if the book is available he can borrow it for as long as he want considering the cost is weekly incremental (we offer the first week for free for new users), all he has to do is enter the expected return date and our server calculates the estimated cost and display it for him; if the book is not available, he can put on a request for it, the librarian will get it and he will try and provide it; or our system can provide him with similar books to choose from. When the student returns the book, he must first get the verification from the librarian about the book integrity before he gets the bill. In addition, our system provides publishers the opportunity to upload new books and get paid according to their value which is determined first by the librarian; they can also review the status of their book and see how many viewers they reached. It’s the complete package for library management.

Table of content:

1. INTRODUCTION

1.1 Purpose

The goal of this document is to provide support information on our software. It will attempt to explain the functionality of the program and the features it provides.
This document is intended for:
- Developers who can review project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it.
- Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
- End users of this application who wish to read about what this project can do.

1.2 List of definitions

- UML diagrams: Unified Modeling Language, a graphical language to represent data.

- IDE: Integrated Development Environment, a software application that provides comprehensive facilities to computer programmers for software development.

1.3 Scope

"El Ray2 Library System" is our product name. It is used to automate libraries for more efficient and more organized transactions. It separates the system's human actors and make each one of them deal with a computer, this eliminates wasted time and reduces probability of conflicts or misunderstanding.

1.4 Overview

The document will start with some general description for the software, then proceeds to the system requirements, followed by several UML diagrams, and then it finishes with the user interface design and system testing.

2. General description

2.1 Product Perspective

Other systems like the publishing company providing the book(s), the delivery company that the user may use in case of wanting a physical copy of the book(s).

2.2 General Capabilities

The main capabilities of the system is to facilitate the process of taking out & returning books for the client, as well as searching for the available books in the system, and providing them to the user through simple steps.

2.3 General Constrains

The user must confirm the chosen book he wants to issue or return after confirming his name and password, with restrictions on the amount of books issued to each user and seeing if the user has delayed returns which can lead to restriction from issuing more books in the future.

2.4 User Characteristics

The user who uses this software is interested in accessing books easily and professionally.

2.5 Environment Description

The user using a system on a computer platform to choose to borrow or return a specified book chosen by him from the offered items.

2.6 Assumptions and Dependencies

The user only chooses his desired book, he enters his name and password and his request (borrow or return) is to be stored in the system and may be accessed by the librarian (admin) to keep track of user.

2.7 Other resources needed

This system needs a data base that saves all the book data and user data and an internet connection with the publishing companies providing the books.

3- SYSTEM REQUIREMENTS

3.1- Functional Requirements

1. User can create a library account.
2. User can include his personal and professional information
3. User can access his navigation history and be presented with recommended books based on his favorites.
4. Each user account has a cash balance for transactions.
5. Student can search for a book using any information he has (name, type, price range, author …etc.).
6. 4. Student can borrow books online.
7. Student enters the expected return date of borrowed book and the estimated cost is displayed for him.
8. User can put on a request for the book if it’s not available and can be provided by similar books.
9. Librarian will get the requested book from an external library.
10. When the student returns the book, he must first get the verification from the librarian about the book integrity before he gets the bill.
11. Publisher can upload new books and get paid according to their value.
12. Publishers can review the status of their books and see how many viewers the book reached.
13. An informative and easy GUI is supported.

3.2- Non-Functional Requirements

- **Product requirements:**
  The application execution speed is fast as it executes in 100ms. It takes memory size of 1MB. The system should have fast response time as it responds in 10ms after failure and it should tolerate common types of faults (such as signup faults and login faults)
- **Organizational requirements :**
  It is written in C++ using QT IDE. The delivery time is on January 1st. It should conform to IEEE and ISO standards.
- **External requirements:**
  The system should conform to all applicable local and international laws.

4. Use-Case Diagram

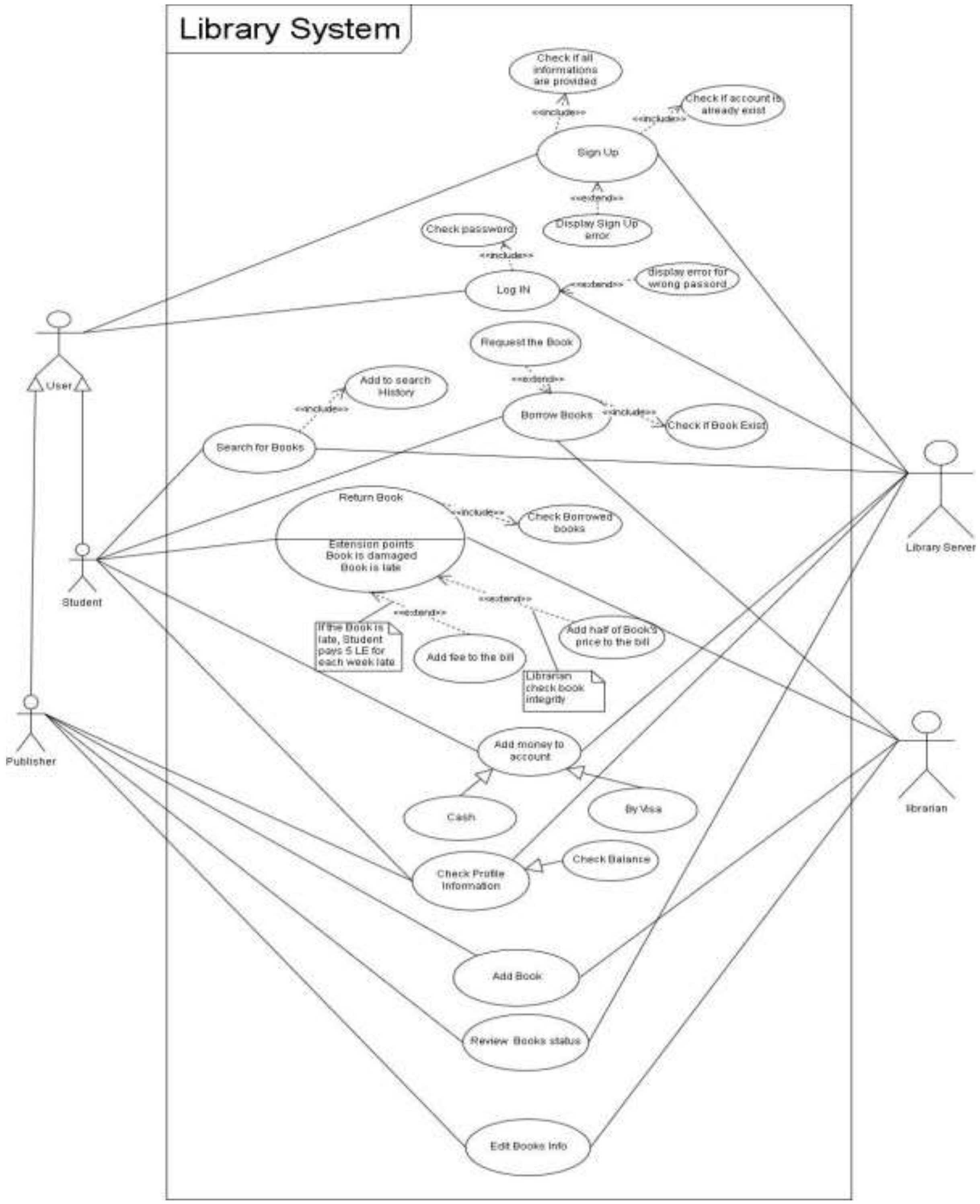


Figure 1 Use case diagram

5. Narrative Description of Use Cases

Use case name	**Book details entry**
Related requirements	3
Goal in context	User enters details of book
Preconditions	Correct program password is entered
Successful End condition	Book is added to system
Failed End condition	Book not added to system
Primary actors	User
Secondary actors	Admin
Trigger	Book added to system and database is modified
Included cases	Book ID Author Book Name
Main flow	User enters program password User wants to enter a book User fills fields Book is added to the system Data base is modified
Extension	-

Use case name	**Sign Up**
Related requirements	1
Goal in context	Create new user account
Preconditions	-
Successful End condition	Account successfully created
Failed End condition	Failed to create an account
Primary actors	User
Secondary actors	Librarian server
Trigger	User presses Sign up button
Included cases	- - -
Main flow	Step / Action 1 User enters Name 2 User enters password 3 System confirms password 4 User enters E-mail 5 User select account type 6 User account is created 7 Data base is modified
Extension	-

Use case name	**Log In**
Related requirements	1
Goal in context	Log in to existing account
Preconditions	-
Successful End condition	Logged in successfully
Failed End condition	Failed to Log in
Primary actors	User
Secondary actors	Librarian server
Trigger	User presses Log in button
Included cases	- - -
Main flow	Step Action 1 User presses Log in 2 User enters Name 3 User enters password 4 System verify entered name and password 5 Information are returned as verified from data base 6 User Logged in to the account
Extension	-

Use case name	**Search Books**
Related requirements	6
Goal in context	User search for desired book
Preconditions	-
Successful End condition	User found the desired book
Failed End condition	Book does not exist on the system
Primary actors	Student
Secondary actors	Librarian server
Trigger	User presses on Search bar
Included cases	- - -
Main flow	Step Action 1 Student presses on search bar 2 Student enters Book's name 3 server check book's name 4 Book's name is returned as exist from data base 5 Book's name appear in search bar 6 Book's name is saved as searched book in data base for this specific user
Extension	-

Use case name	**Borrow Book**
Related requirements	4
Goal in context	User can borrow desired book and request for non-existence book
Preconditions	-
Successful End condition	Use found book and borrow it
Failed End condition	Book does not exist on the system
Primary actors	Student
Secondary actors	Librarian
Trigger	Student presses borrow button
Included cases	- - -
Main flow	Step Action 1 Student presses on borrow button 2 Student enters Book's name 3 Student enters Book's Type 3 server check book's name 4 Book's name is returned as exist in data base 5 Student borrow book from librarian 6 Data base is modified
Extension	-

Use case name	**Return Book**
Related requirements	5
Goal in context	Student returns book back to library
Preconditions	-
Successful End condition	Student return it in expected return date
Failed End condition	Student has returned book late
Primary actors	Student
Secondary actors	Librarian
Trigger	Student asks librarian to return book
Included cases	- - -
Main flow	Step / Action
	1 Student Asks librarian to return book
	2 Student log in to system
	3 Student enters borrowed book's ID
	3 system check's student name
	4 Book's ID is returned as Right borrowed book from data base for student name
	5 Student Enters today's date
	6 bill is calculated
	7 bill is taken from Student's credit
	8 Student return Book to librarian
	9 Data base is modified
Extension	-

Use case name	**Add money to account**
Related requirements	5
Goal in context	Student add money to his credit
Preconditions	-
Successful End condition	Money add successfully
Failed End condition	Failed to add money as student entered wrong information
Primary actors	Student
Secondary actors	Librarian server
Trigger	Student Log in
Included cases	- - -
Main flow	Step Action 1 Student Log in to system successfully 2 Student presses Edit account 3 Student presses cash amount 4 Student Select desired amount of money 5 Student select desired way to add money 6 money is add to the account 7 Data base is modified
Extension	-

Use case name	**Check profile information**
Related requirements	5
Goal in context	Student and publisher can check and modify profile information
Preconditions	-
Successful End condition	-
Failed End condition	-
Primary actors	User
Secondary actors	Librarian server
Trigger	User log in
Included cases	- - -
Main flow	Step — Action 1 — User Log in to system successfully 2 — User presses view profile
Extension	-

Use case name	**Add book**
Related requirements	11
Goal in context	Publisher can upload new books to the system
Preconditions	-
Successful End condition	Publisher logged in successfully ad upload book successfully
Failed End condition	Failed to upload book
Primary actors	Publisher
Secondary actors	Librarian
Trigger	Publisher logged in
Included cases	- - -
Main flow	Step — Action 1 — Publisher Log in to system successfully 2 — publisher presses upload Book 3 — Publisher enters book’s Name, type and price 4 — Publisher gives book to librarian 5 — Data base is modified
Extension	-

Use case name	**Add book**
Related requirements	11
Goal in context	Publisher can upload new books to the system
Preconditions	-
Successful End condition	Publisher logged in successfully ad upload book successfully
Failed End condition	Failed to upload book
Primary actors	Publisher
Secondary actors	Librarian
Trigger	Publisher logged in
Included cases	- - -
Main flow	Step — Action 1 — Publisher Log in to system successfully 2 — publisher presses upload Book 3 — Publisher enters book's Name, type and price 4 — Publisher gives book to librarian 5 — Data base is modified
Extension	-

Use case name	**Review book status**
Related requirements	12
Goal in context	Publisher review book status , borrowed times and searched times
Preconditions	-
Successful End condition	Publisher review book status successfully
Failed End condition	Failed
Primary actors	Publisher
Secondary actors	Librarian
Trigger	Publisher logged in
Included cases	- - -
Main flow	Step Action 1 Publisher Log in to system successfully 2 publisher presses Review book status 3 Publisher enters book’s Name 4 book’s status is returned from data base
Extension	-

Use case name	**Edit book’s information**	
Related requirements	12	
Goal in context	Publisher can update book’s name , type or price	
Preconditions	-	
Successful End condition	Publisher updated book’s information	
Failed End condition	Failed -----???????	
Primary actors	Publisher	
Secondary actors	Librarian	
Trigger	Publisher logged in	
Included cases	- - -	
Main flow	Step	Action
	1	Publisher Log in to system successfully
	2	publisher presses update book button
	3	Publisher enters book’s Name
	4	book’s name is returned as exist from data base
	5	Publisher select what to update
	6	Publisher enters new information
	7	Data base is modified
Extension	-	

6.REQUIREMENTS VALIDATION

6.1 Requirements Traceability Matrix

REQ ID	REQ.1	REQ.2	REQ.3	REQ.4	REQ.5	REQ.6	REQ.7	REQ.8	REQ.9	REQ.10	REQ.11	REQ.12	REQ.13
REQ.01													R
REQ.02	D			R									R
REQ.03	D				D	D							
REQ.04	D	R											
REQ.05	D		D	R		D	R						
REQ.06	D						D	D					
REQ.07	D			D		R							
REQ.08	D		R						D				
REQ.09	D							D					
REQ.10	D			R		R	D						
REQ.11	D			D					R				
REQ.12	D		D		R						D		
REQ.13													

6.2 Source Traceability Matrix

REQ ID	User	Student	Publisher	Librarian	Developer
REQ.01	✓				
REQ.02	✓				
REQ.03	✓				
REQ.04	✓				✓
REQ.05		✓			
REQ.06		✓			
REQ.07		✓			
REQ.08		✓			
REQ.09				✓	
REQ.10				✓	
REQ.11			✓		
REQ.12			✓		
REQ.13					✓

7. CLASS MODEL

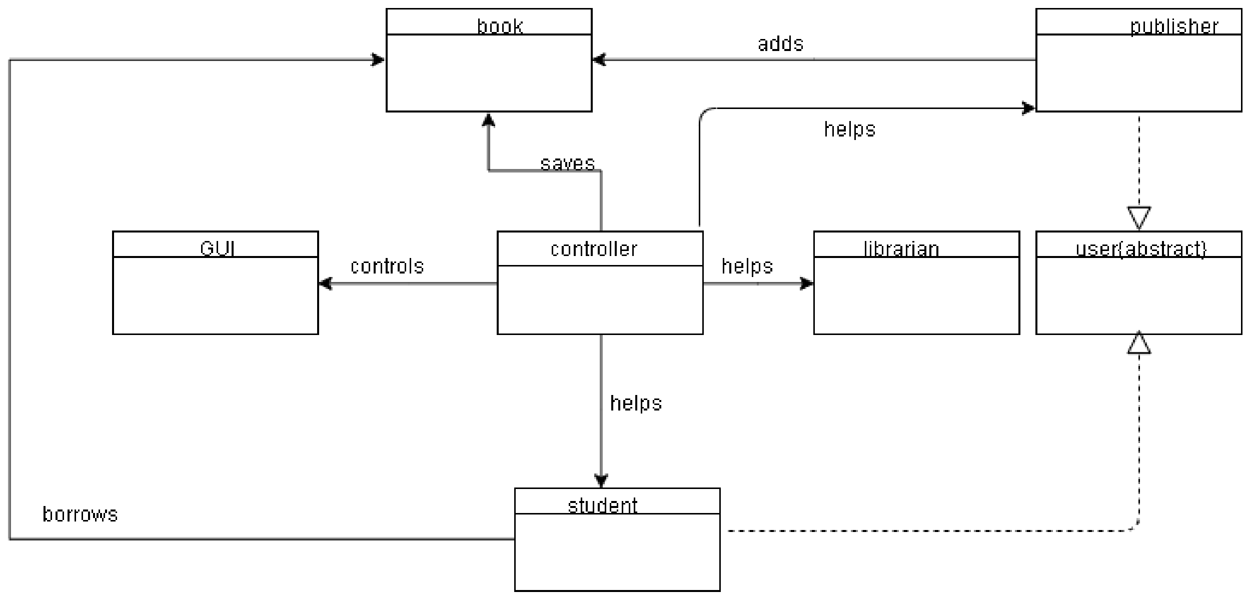


Figure 2 class model

CLASS **GUI**
RESPONSIBILITY 1. Contains book details 2. Contains user's details 3. Searching for books 4. borrowing and returning books 5. Showing book issue status 6. Deleting books 7. Reset the program 8. Access the user details to confirm 9. Access the book details to confirm 10. The books and user will be saved in the database
COLLABORATION Class **System**

CLASS **Book**
RESPONSIBILITY 1. Contains the basic fields for every book 2. Contains method to get book name, get book ID and get book author. 3. Contains method to get the book status (borrowed, available, unavailable) 4. Helps create new items
COLLABORATION 1. Class **student** 2. Class **publisher** 3. Class **controller**

User **{abstract}**
RESPONSIBILITY 1. Contains the fields for the user identification 2. Provide incomplete functions 3. Contains extra method to get the user details.
COLLABORATION 1. Class **publisher** 2. Class **student**

CLASS **librarian**
RESPONSIBILITY 1. Access book and user data through system 2. can delete books 3. check books conditions after return
COLLABORATION 1. Class **system** 2. Class **Book**

CLASS **controller**
RESPONSIBILITY 1. Save data entered by user 2. Update the database of user info 3. Update the book database and save it
COLLABORATION 1. Class **App** 2. Class **User**

8. State Diagram

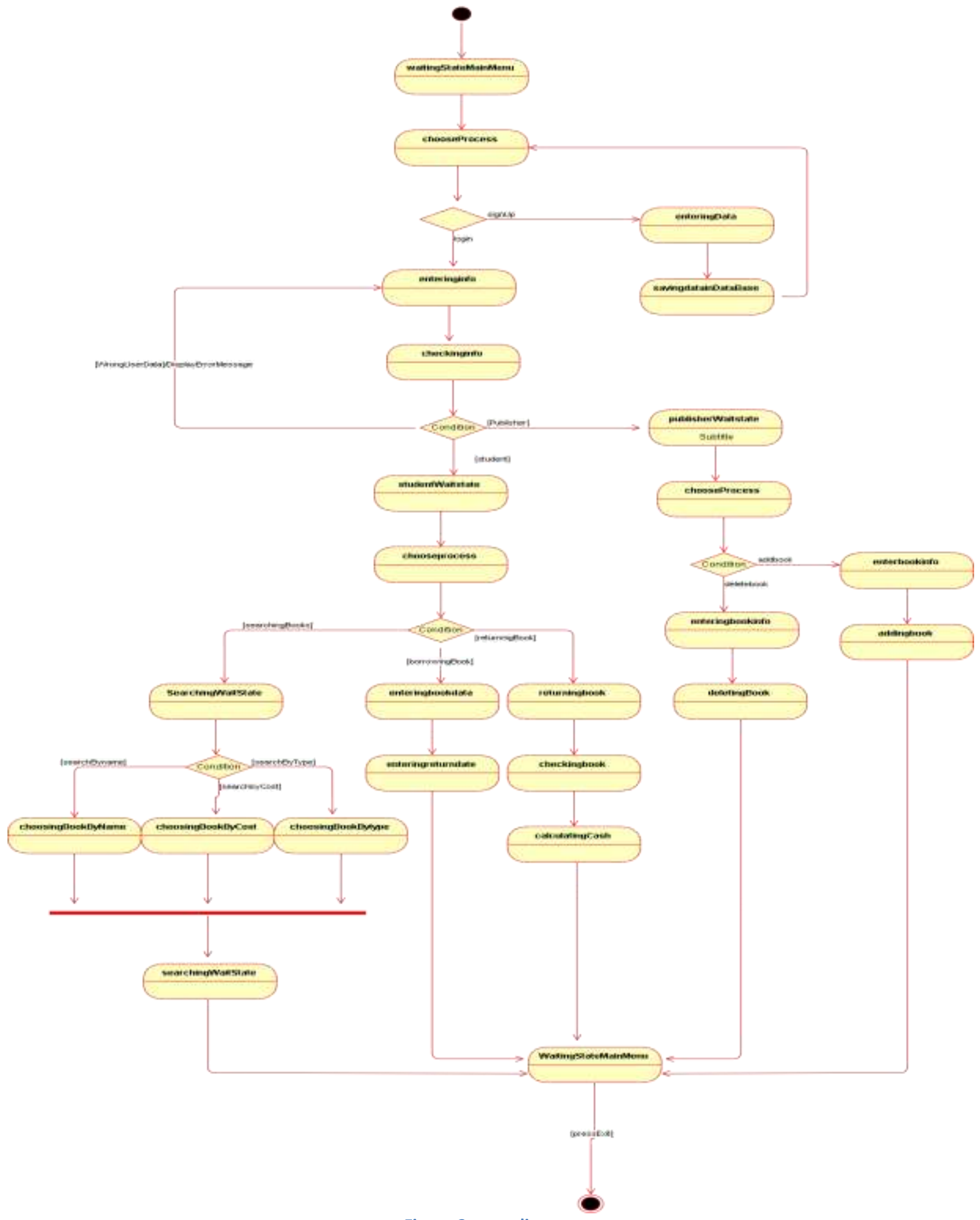
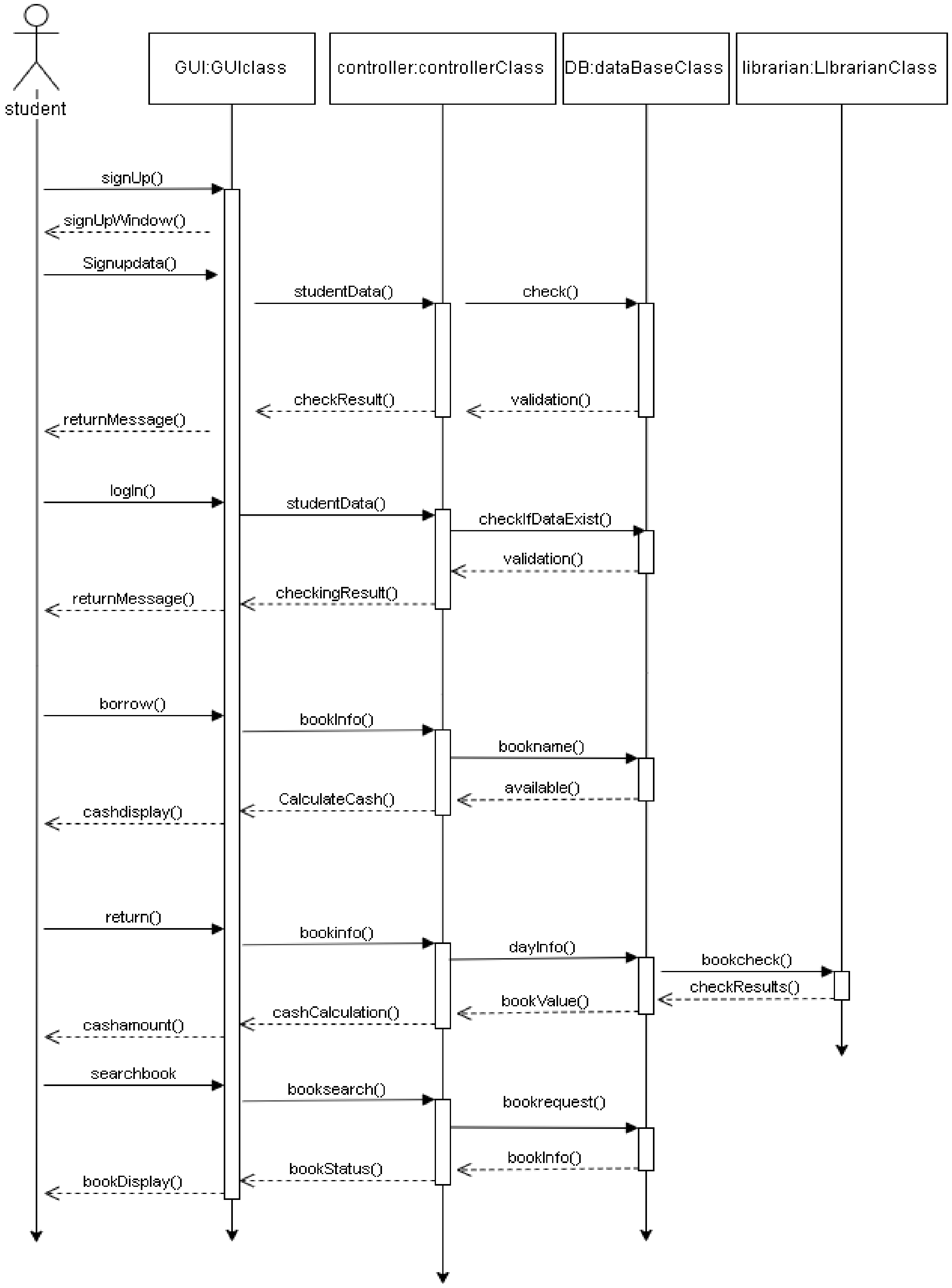


Figure 2 state diagram

9. Interaction Diagram



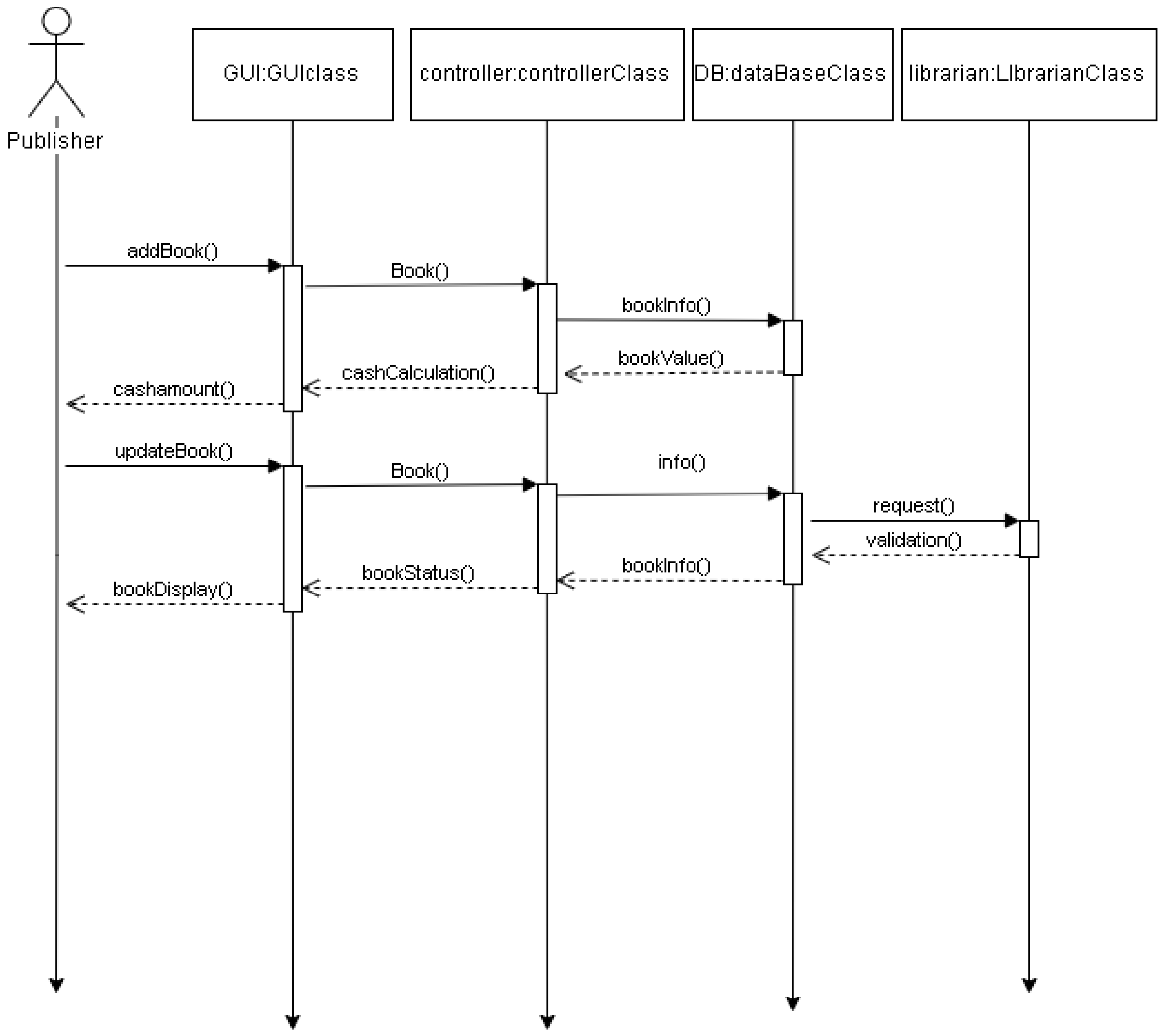


Figure 3 interaction diagram

10. DETAILED CLASS DIAGRAM

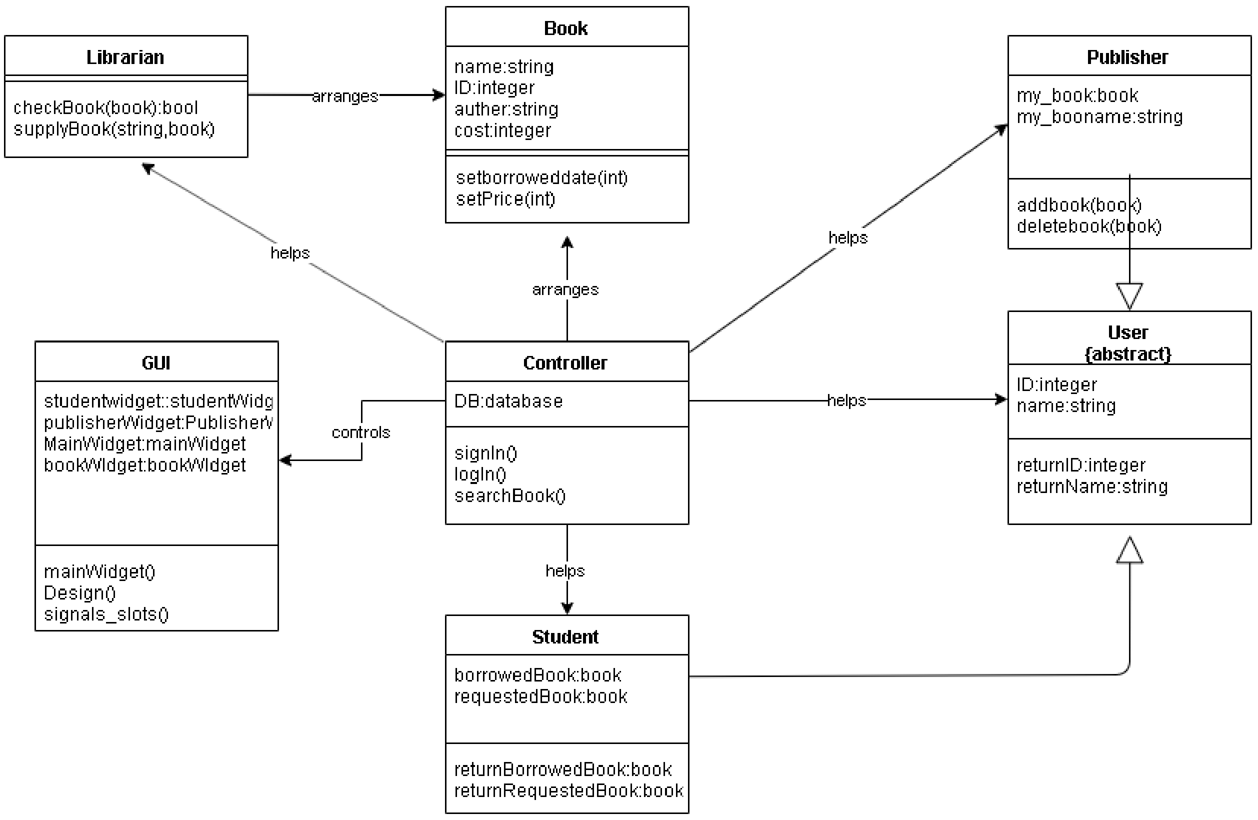
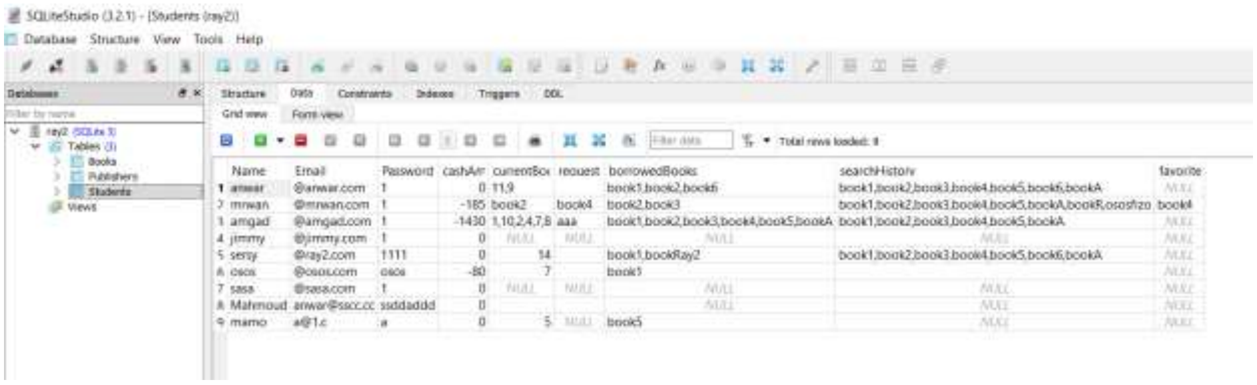


Figure 4 detailed class diagram

11. Data Model Design

11.1 Student Table

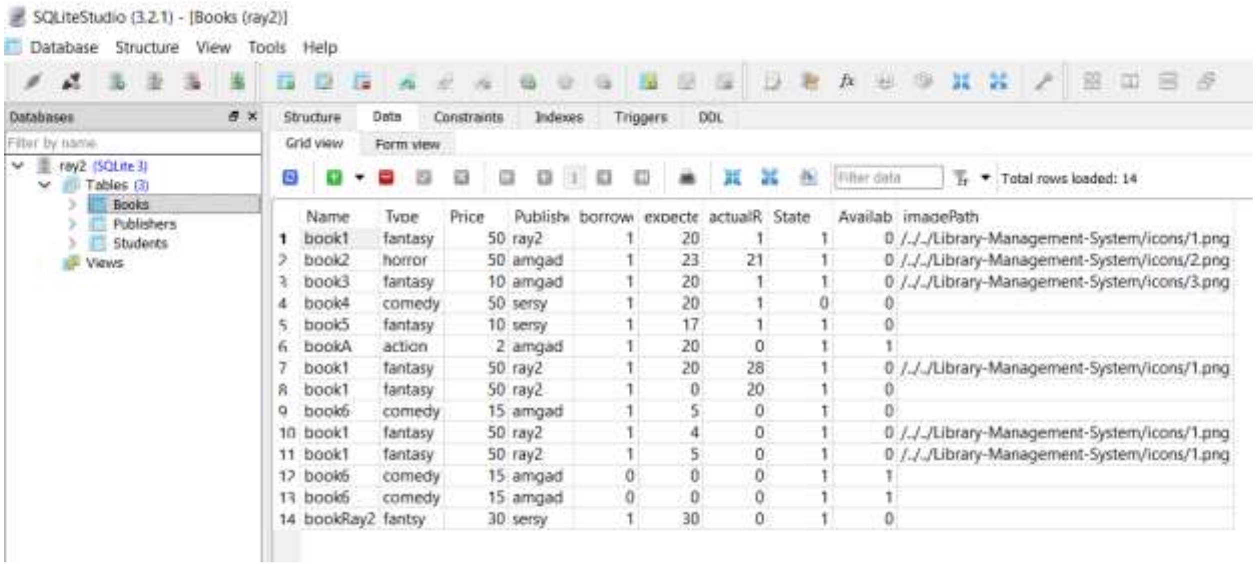


	Name	Email	Password	cashAm	currentBo	request	borrowedBooks	searchHistory	favorite
1	anwar	@anwar.com	1	0	11,9		book1,book2,book6	book1,book2,book3,book4,book5,book6,bookA	NULL
2	mrwan	@mrwan.com	1	-185	book2	book4	book2,book3	book1,book2,book3,book4,book5,bookA,bookR,ososfizo	book4
3	amgad	@amgad.com	1	-1430	1,10,2,4,7,8	aaa	book1,book2,book3,book4,book5,bookA	book1,book2,book3,book4,book5,bookA	NULL
4	jimmy	@jimmy.com	1	0	NULL	NULL	NULL	NULL	NULL
5	sersy	@ray2.com	1111	0	14		book1,bookRay2	book1,book2,book3,book4,book5,book6,bookA	NULL
6	osos	@osos.com	osos	-80	7		book1		NULL
7	sasa	@sasa.com	1	0	NULL	NULL	NULL	NULL	NULL
8	Mahmoud	anwar@sscc.cc	ssddaddd	0			NULL	NULL	NULL
9	mamo	a@1.c	a	0	5	NULL	book5	NULL	NULL

Attributes : Name,Email,Password ,currentBooks,requests,borrowedBooks and searchHistory

Key : studentID

11.2 Books Table:

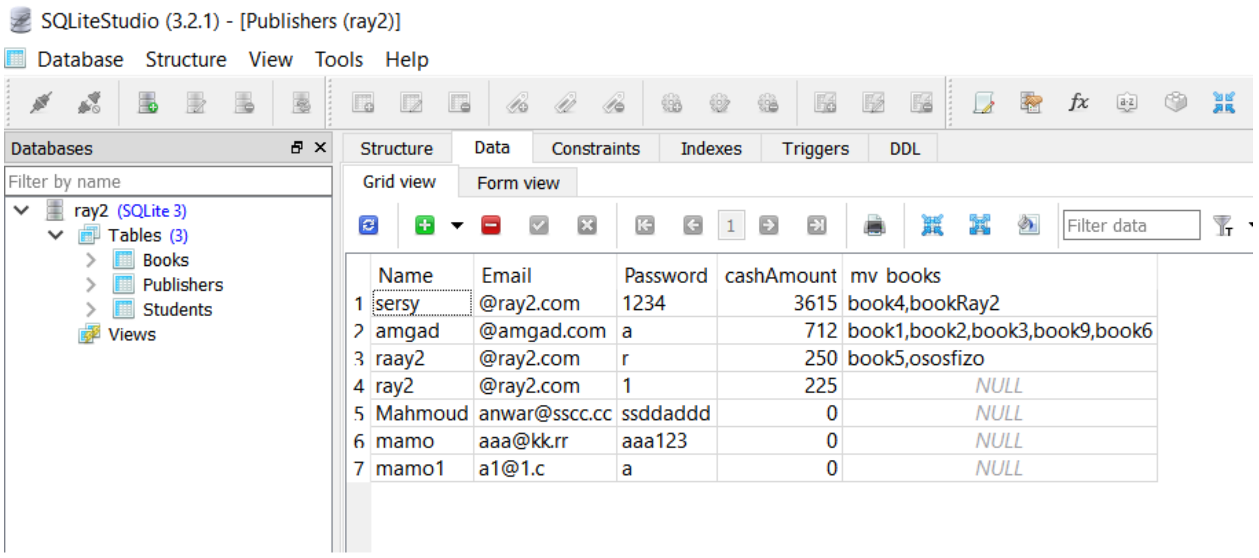


	Name	Type	Price	Publish	borrow	expecte	actualR	State	Availab	imagePath
1	book1	fantasy	50	ray2	1	20	1	1	0	/../../Library-Management-System/icons/1.png
2	book2	horror	50	amgad	1	23	21	1	0	/../../Library-Management-System/icons/2.png
3	book3	fantasy	10	amgad	1	20	1	1	0	/../../Library-Management-System/icons/3.png
4	book4	comedy	50	sersy	1	20	1	0	0	
5	book5	fantasy	10	sersy	1	17	1	1	0	
6	bookA	action	2	amgad	1	20	0	1	1	
7	book1	fantasy	50	ray2	1	20	28	1	0	/../../Library-Management-System/icons/1.png
8	book1	fantasy	50	ray2	1	0	20	1	0	
9	book6	comedy	15	amgad	1	5	0	1	0	
10	book1	fantasy	50	ray2	1	4	0	1	0	/../../Library-Management-System/icons/1.png
11	book1	fantasy	50	ray2	1	5	0	1	0	/../../Library-Management-System/icons/1.png
12	book6	comedy	15	amgad	0	0	0	1	1	
13	book6	comedy	15	amgad	0	0	0	1	1	
14	bookRay2	fantsy	30	sersy	1	30	0	1	0	

Attributes : Name, Type, Price, Publisher, borrowed, exectedreturnDate, actualReturnDate, state and available

Key : bookID

11.3 Publisher Table:


SQLiteStudio (3.2.1) - [Publishers (ray2)]
Database Structure View Tools Help
Databases
Filter by name
ray2 (SQLite 3)
Tables (3)
Books
Publishers
Students
Views
Structure Data Constraints Indexes Triggers DDL
Grid view Form view
Filter data

	Name	Email	Password	cashAmount	mv books
1	sersy	@ray2.com	1234	3615	book4,bookRay2
2	amgad	@amgad.com	a	712	book1,book2,book3,book9,book6
3	raay2	@ray2.com	r	250	book5,ososfizo
4	ray2	@ray2.com	1	225	NULL
5	Mahmoud	anwar@sscc.cc	ssddaddd	0	NULL
6	mamo	aaa@kk.rr	aaa123	0	NULL
7	mamo1	a1@1.c	a	0	NULL

Attributes : Name,Email,Password ,CashAmountandmyBooks

Key : PblisherID

12. User Interface Design

12.1 Human Factors in UI Design

- Limited short-term memory
    - People can instantaneously remember about 7 pieces of information, so added 6 tabs in the main menu.
    - The menu is sorted starting the frequently needed tabs (book details entry, search book details).
    - The least needed taps (about, exit) are in the very bottom of the menu.

- People make mistakes
    - There are multiple inputs (passwords, search data, etc…) needed from the user which can lead to several mistakes.
    - The system error message is very simple which shows up without any annoying sounds which can increase stress and, hence, the likelihood of more mistakes increases.

- People are different
  The system interface is simple and within the capabilities of the average user.

- People have different interaction preferences.

12.2 UI Styles used in user interface

- Menu selection :
    - The system index contains only book navigation menu.
    - Some other tools in the system use menus as well, but with limited options to avoid complexity.
    - Menu selection is perfect to prevent user errors, as no typing required.

- Form Fill-in :

Simple data entry used in the details entry and search sections, or for some required passwords.

- Natural Language :
  Natural Language used is accessible to the average user.

12.3 Action done by the user that changes the user interface:

- Password entry:
  - Leads to the main user interface.
  - Successful login will allow access to the main menu.

- Choosing from the main navigation menu:
  - Leads to the system services (adding and searching for books, admin menu and issue & return section, etc…).

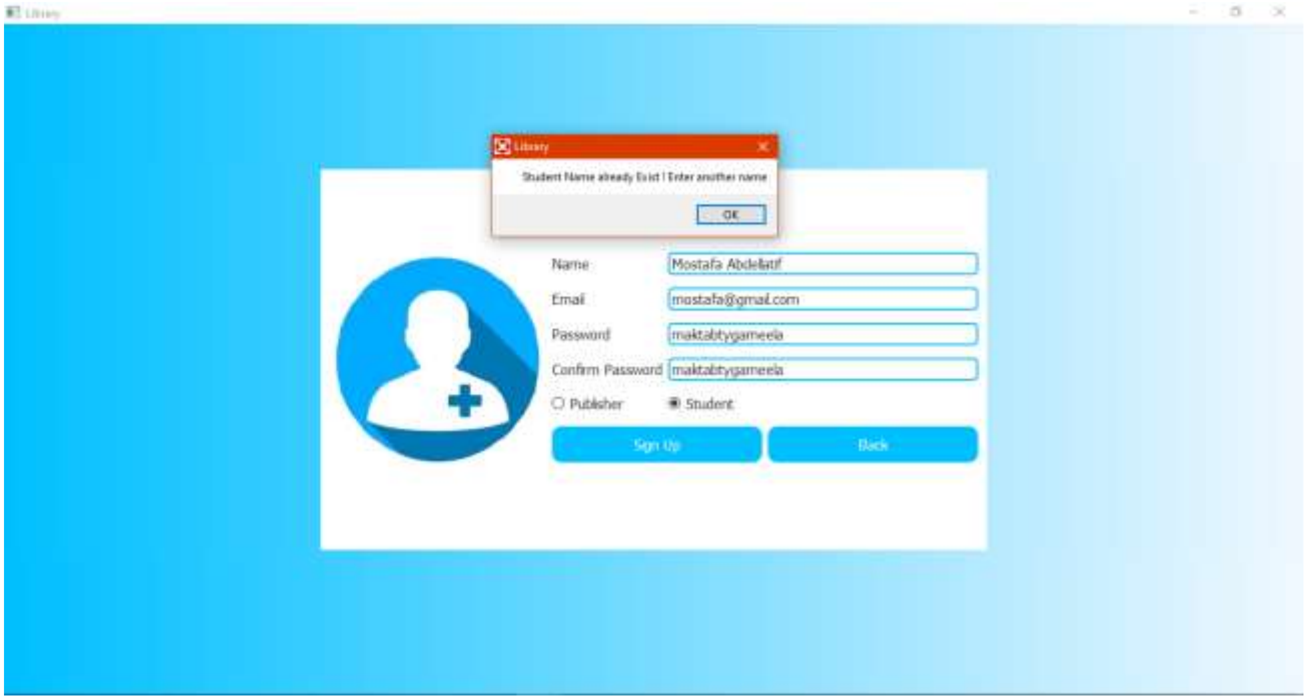
- Exit:
  Closes the program.

12.4 Interface Summary:

- The system is very easy to use and learn.
- All the services are close from the system start point.
- Very high speed of operation.
- Only 1 color is used mainly its color code: 00BFFF.
- Interface provides meaningful feedback when errors occur.

1. Signup errors.

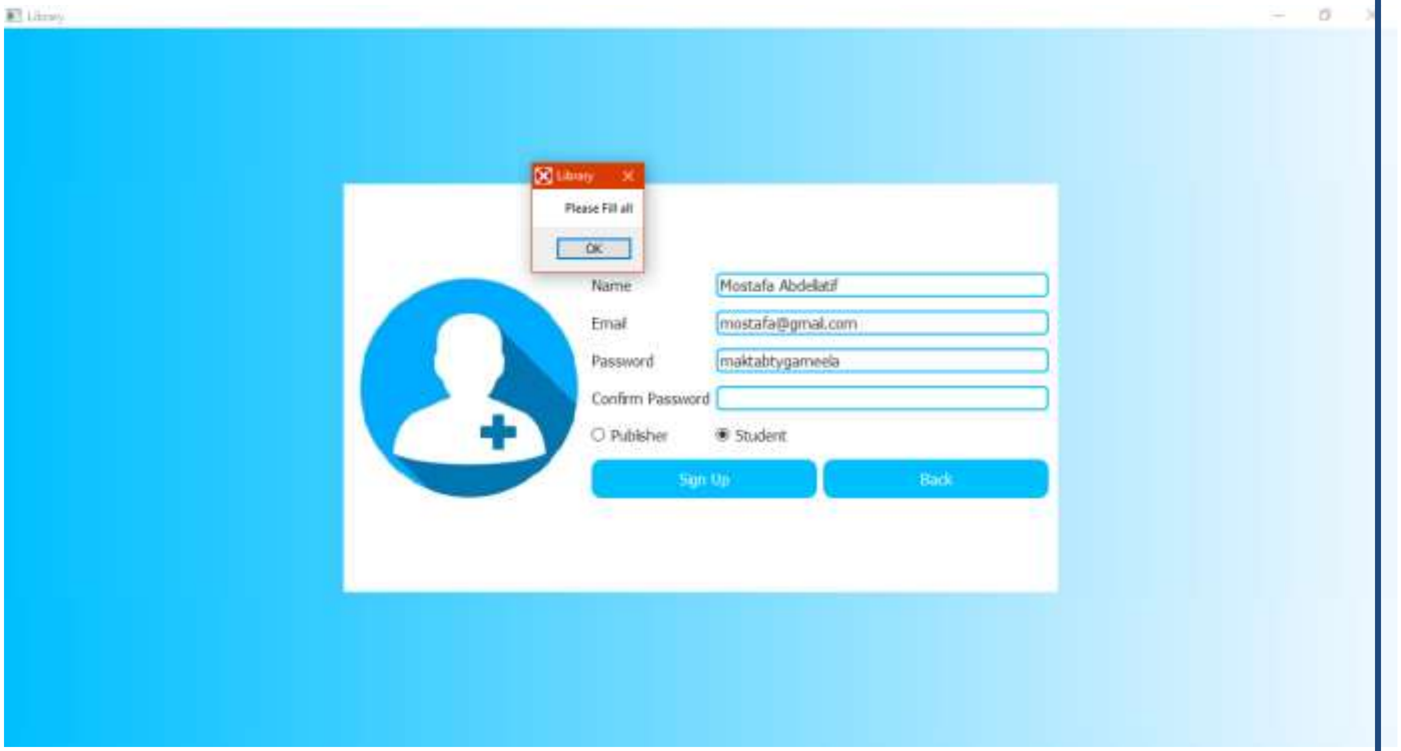
1.1. Enter an existing user name.



Student Name already Exist ! Enter another name

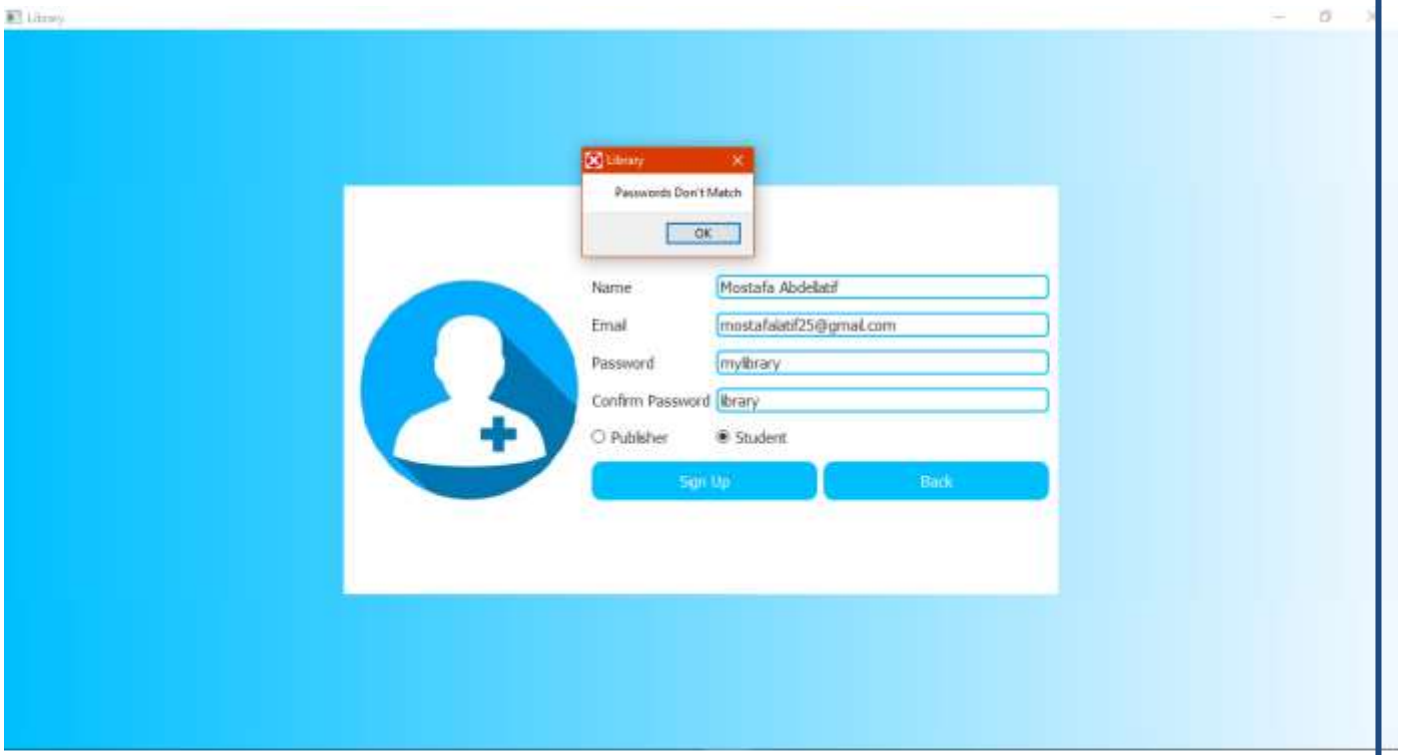
Name: Mostafa Abdellatif
Email: mostafa@gmail.com
Password: maktabtygameela
Confirm Password: maktabtygameela
Publisher [✓ Student]

1.2. Not filling all data fields.



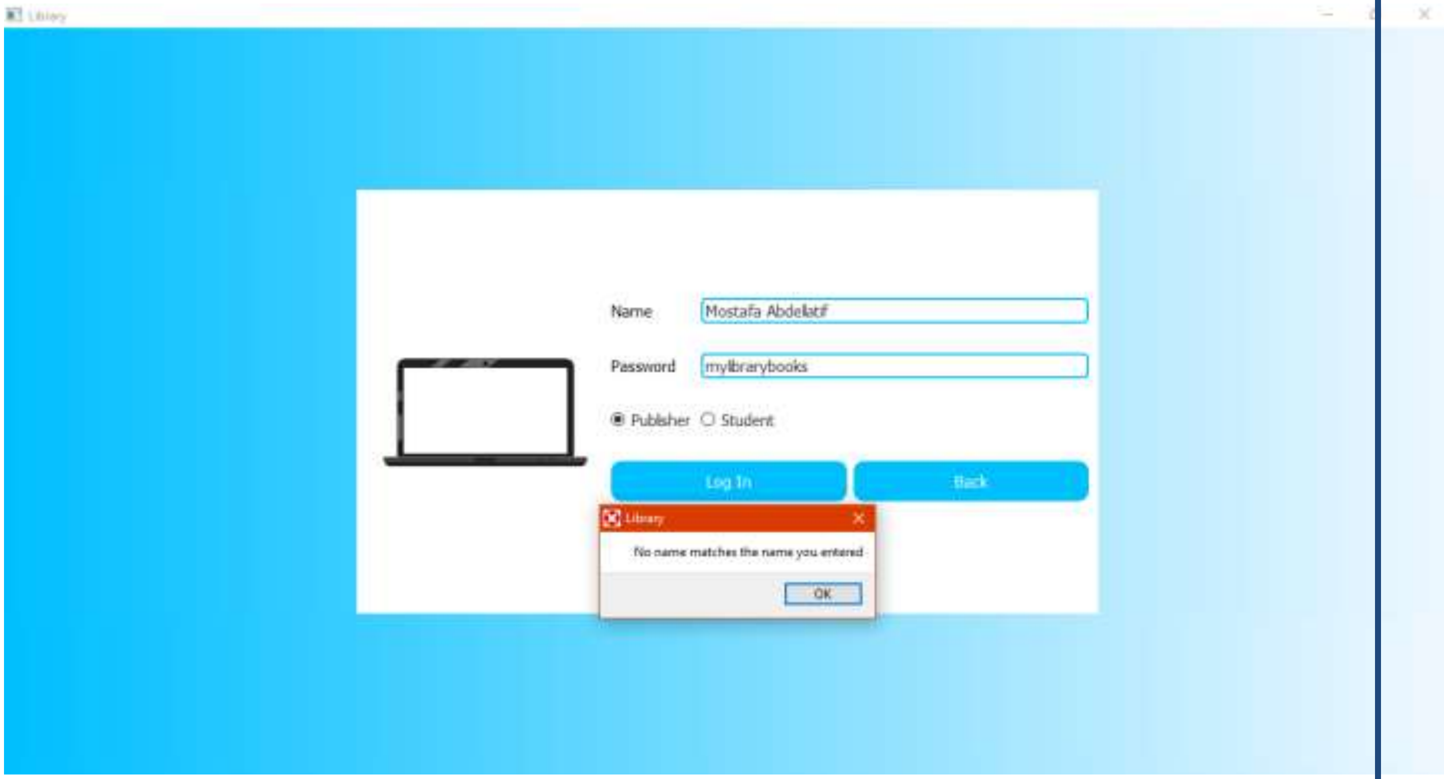
Name: Mostafa Abdellatif
Email: mostafa@gmal.com
Password: maktabtygameela
Confirm Password:
○ Publisher ● Student

1.3. Password confirm doesn't match the entered one.



2. Logging in errors.

2.1. Entering wrong name or choosing wrong category.

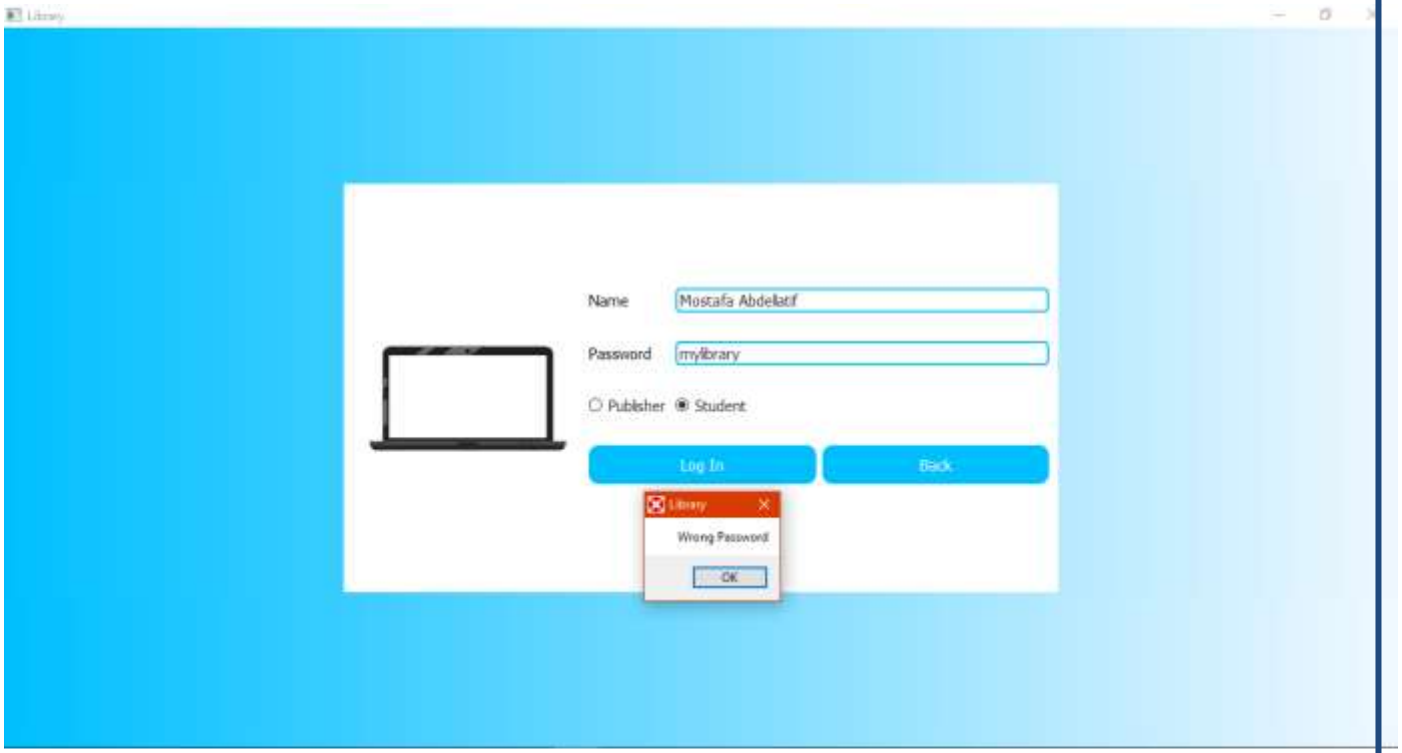


Mostafa Abdellatif

mylibrarybooks

No name matches the name you entered

2.2. Entering wrong password.



Name: Mostafa Abdellatif

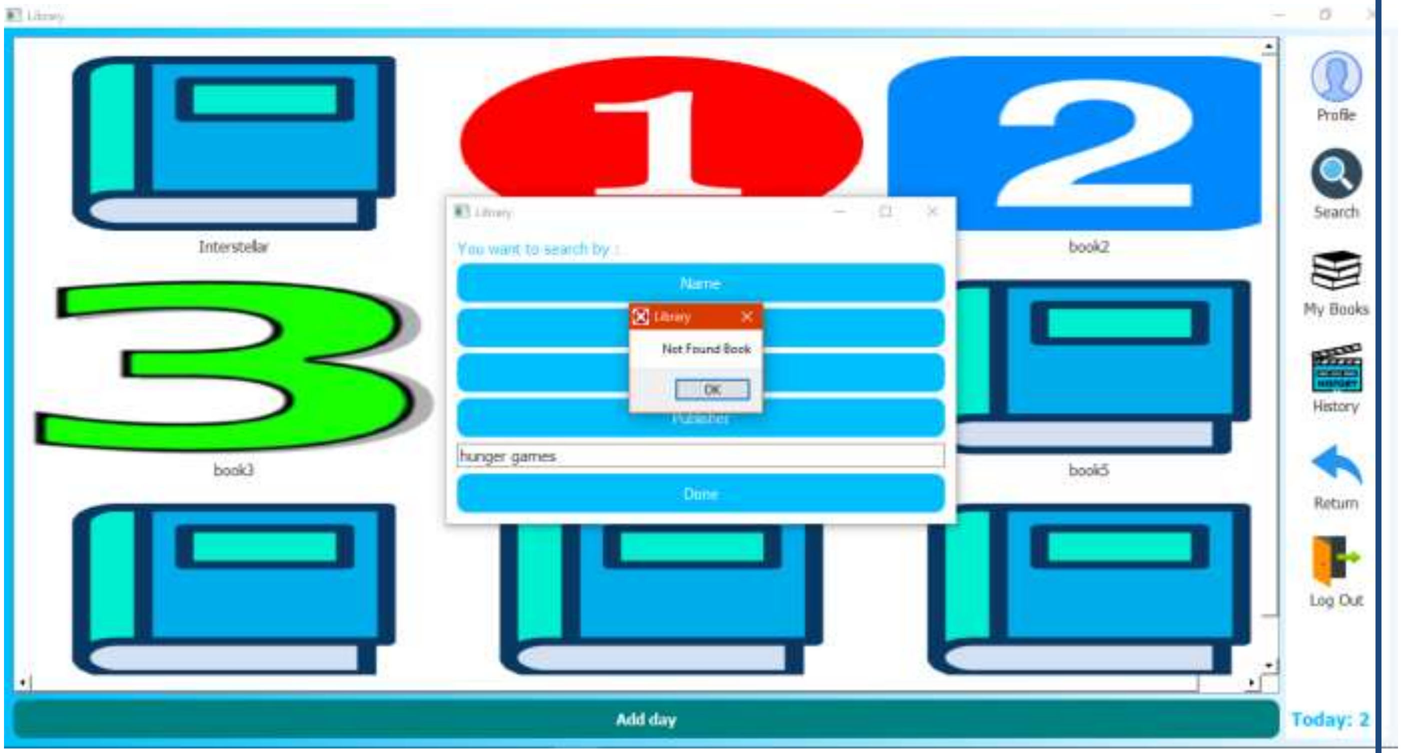
Password: mylibrary

Publisher [✓ Student]

Wrong Password

3. Errors in student mode.

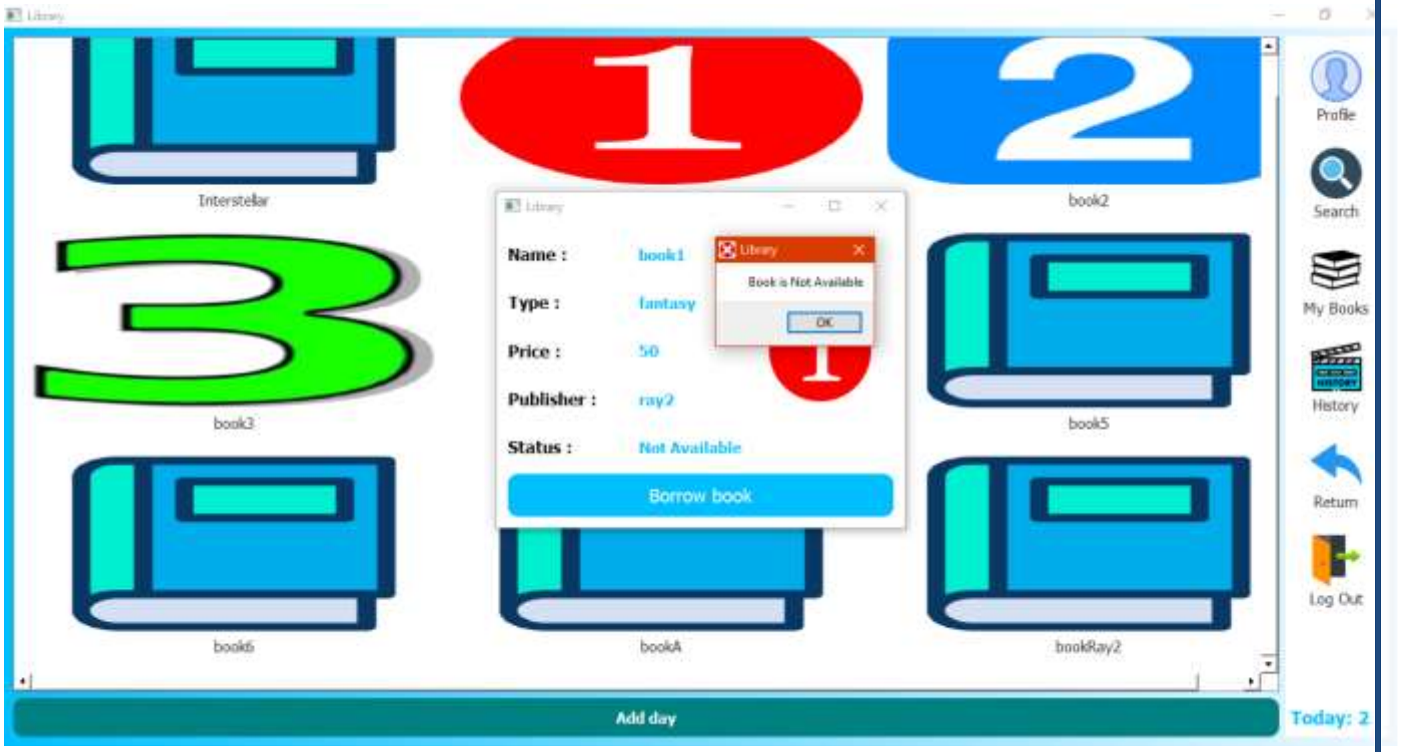
3.1. Searching for a not existing book in library either by name, type, price or publishers name.



hunger games

Not Found Book

3.2. Borrowing un available books.



Name : book1

Type : fantasy

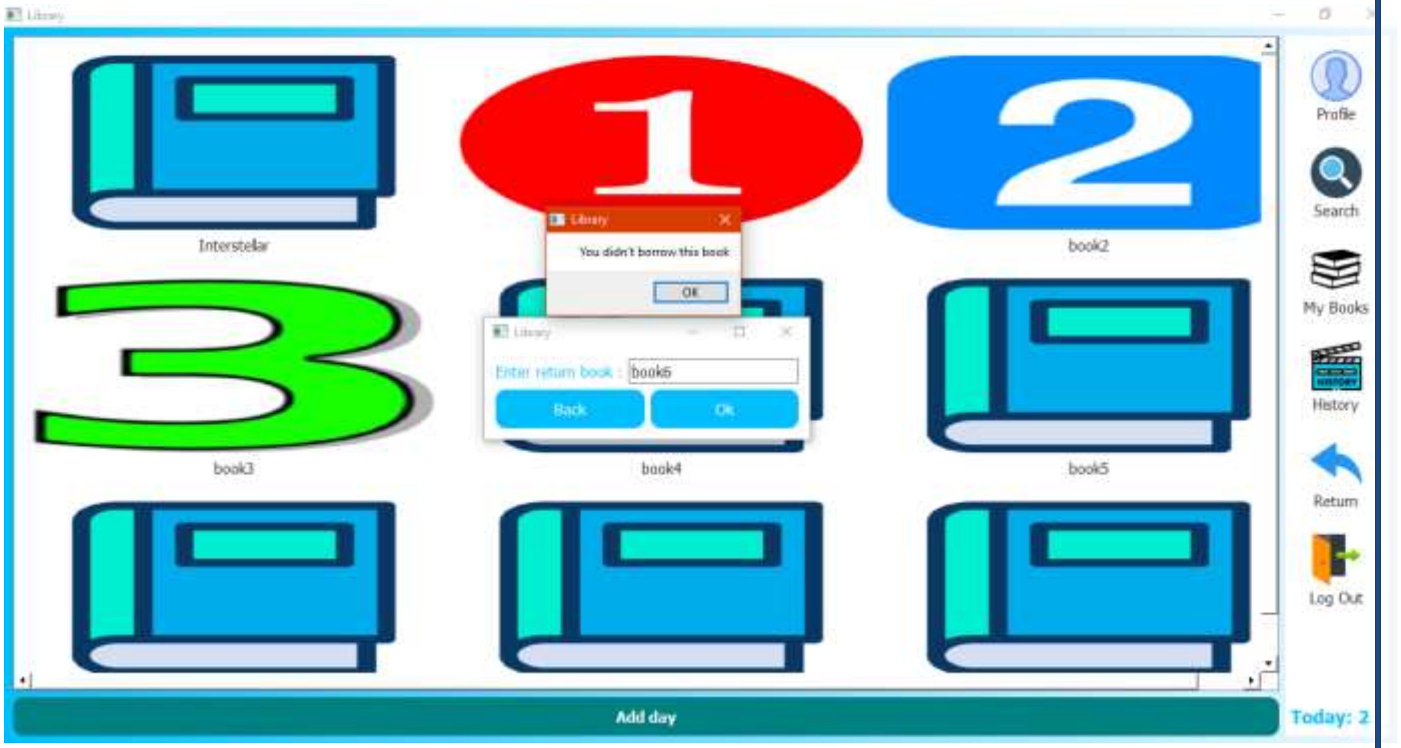
Price : 50

Publisher : ray2

Status : Not Available

Book is Not Available

3.3. Returning un borrowed books.

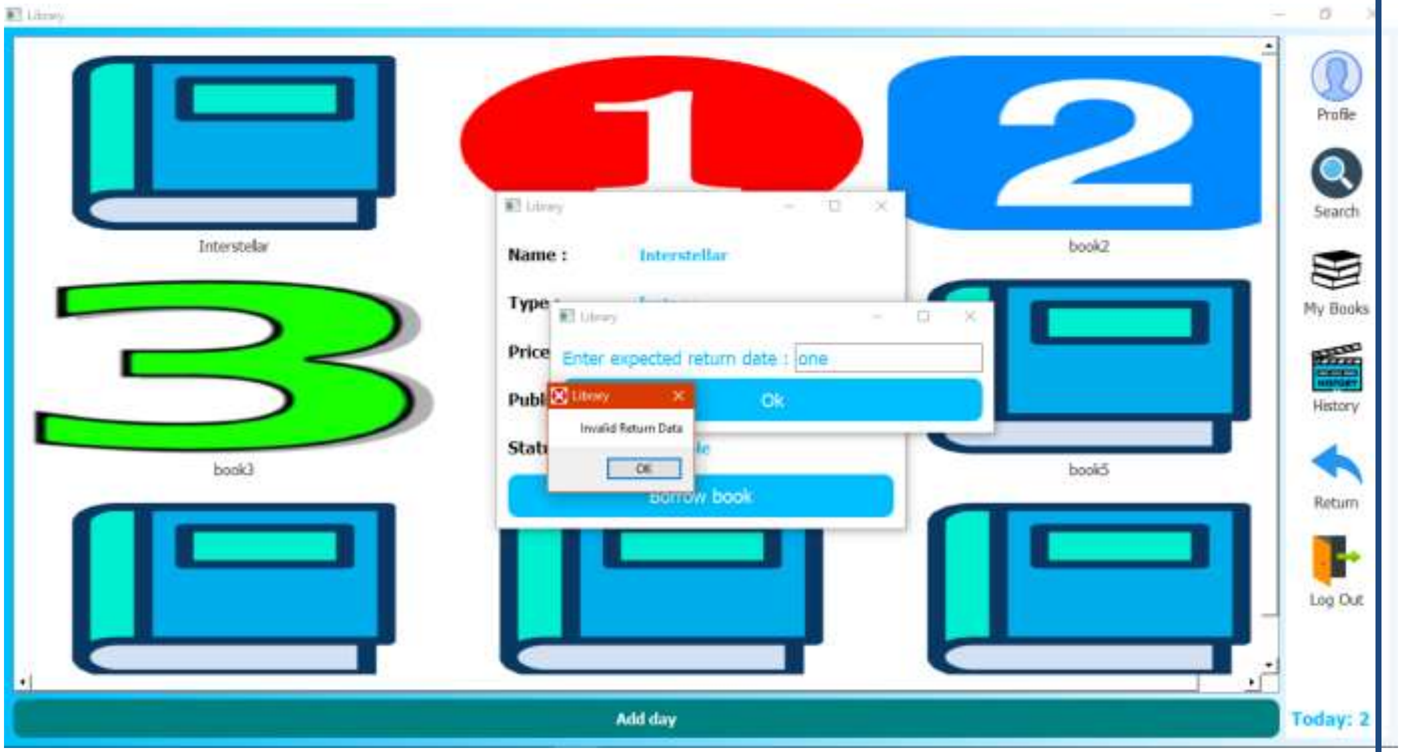


You didn't borrow this book

OK

Enter return book : book6

3.4. Entering return date in words.



Name : Interstellar

Enter expected return date : one

Invalid Return Data

OK

13. CLIENT-OBJECT RELATION DIAGRAM

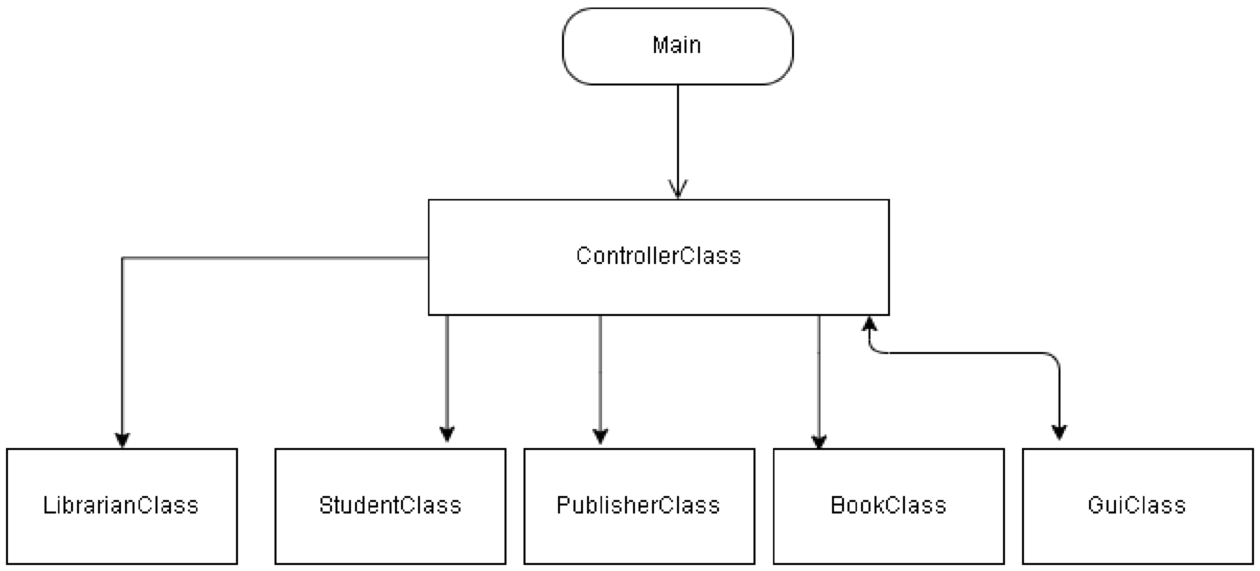


Figure 5 CLIENT-OBJECTS RELATION DIAGRAM

14. DETAILED DESIGN

Controller Class

```
void sign_up(name,email,password,choice)
{
   if ( user is student )
   {
      student:: set student Name ;
      student:: set student Password;
      student:: set student Email;
      if ( student user name found in data base )
      {
         print "Student Name already Exist ! Enter another name" ;
                        Re-enter the user name ;
      }
                else
                {
                        print "You signed up successfully" ;
                }
```

```
        }
    else
    {
        publisher:: set publisher Name;
        publisher:: set publisher Password;
        publisher:: set publisher Email;

        if ( publisher user name found in data base )
        {
            print " publisher Name already Exist ! Enter another name" ;
                            Re-enter the user name ;
        }
                    else
                    {
                            print "You signed up successfully" ;
                    }
    }
}
void log_in(name,password,choice)
{
    if(user is student)
    {
        if (user name not found in data base students table)
        {
            print "No name matches the name you entered";
        }
        else if( user name found but password not correct)
        {
            print "Wrong Password" ;
        }
        else if( user name found and password matches )
        {

            print "You've logged in successfully";
        }
    }
    else if(user is publisher)
    {
                    if (user name not found in data base publisher table)
        {
            print "No name matches the name you entered";
        }
        else if( user name found but password not correct)
        {
            print "Wrong Password" ;
        }
        else if( user name found and password matches )
        {

            print "You've logged in successfully";
        }
    }
}
void publisherLoggedIn(pubName)
{
    while(TRUE)
    {
        if( publisher want to Upload book )
        {
            print "Enter Book Name : ";
            enter name ;
            print"Enter Book Type : ";
            enter type;
            print "Enter Book Price : ";
            enter price;
            db :: save book's name & price & type in the book data base table ;
        }
        else if(publisher want to Update book)
        {
            enter book name ;
            db :: load book from data base by searching by book name ;
            if(publisher want to update Name )
                        {
                                enter book new name;
                                change book name ;
                        }
            if(publisher want to update type)
                        {
                                enter book new type;
                                change book type ;
                        }
            if(publisher want to update price)
                        {
                                enter book new price;
                                change book price ;
                        }
            db :: save book with changes;
        }
        else if( publisher want to view his published books)
        {
            db :: load vector of publisher book in data base publisher table ;
            print "books names in the vector" ;
```

```
                }
        else if(publisher want to view his cash)
                {
                        db ::load publisher cash fro data base ;
                        print "You Have x L.E.";
        }
                else if(publisher want to view his profile)
                {
                        db :: load Publisher;
                        print publisher name ,email,cash;
                }
        else if( publisher want to edit  his profile)
        {

           db:: load Publisher from data base ;
           if(publisher want to change name )
                                change name ;
                        else if(publisher want to change pass)
                                change pass ;
                        else if(publisher want to change email)
                                change email ;
                        else if(publisher want to add cash)
                                add cash;
           db:: save publisher ;
                }
        else if(publisher want to Log Out) controller :: log out;
        else
                        print "Wrong Choice. " ;
     }
  }
  void Upload_book(pubName ,name ,type,price)
  {
     Book book1;
     book :: set book Name;
     book :: set book Price(price);
     book :: set Type(type);
     book :: set book PublisherName(pubName);
     book :: set book State(1);
     book :: set book Availability(1);
     if(book name found in data base )
     {
        print "book already exist! ";
     }
     db :: save book ;
          db :: add cash amount to publisher
     print "Book saved successfully and cash added to your account <3";
```

```
}
void searchBookByName(BookNameOrID,string stuName)
{
    check for available books
    db :: load Book from data base;
    if (not found )
                    get any unavailable book to just show it to user and know that there is no
available books
          print bookInfo ;
    db :: adding book to search history of the student;
 }
void searchBookByType(type , stuName)
{
    db :: gets all book of this type ;
}

void searchBookByPrice(price,stuName)
{
          db :: gets all book of this price ;
}

void searchBookByPub(string pub, string stuName)
{
   db :: gets all book with this publisher name ;
}

void borrowBook(bookName,stuName,expectedReturnDate)
{
    if(book not found in data base)
    {
       print "This book is not available";

    }
    else {
        db:: load Book;
        Book :: set Borrowed Date;
        Book :: set Expected Return Date;
        db :: add Borrowed Book ID to student table ;
        book :: reset Availability  ;
        print " You borrowed the book successfully, your book's ID is x ";
        print " You will pay :  ceil((expectedReturnDate-borrowDate)/7.0)*bookprice ";
        print " Warning 1 : if you return the book late, you'll pay a fee of 5$ for each week
late";
        print " Warning 2 : if you return the book damaged, you'll pay a fee of half the
book's price";
    }
```

```
}
void returnBook(bookName,stuName)
{
    db :: load student from db ;
    search for book name in borrowed books of student ;
    if(book not found)
        print "You didn't borrow this book";
    else
    {
        calculate the bill by actual return data;
        if(actual Borrowed Period more than expected Borrowed Period or book damaged )
        {
                        increase fees;
        }
        b:: set Availability;
        b:: set State;
        db:: save book ;
    }
}
void getBookInfo(BookNameOrID)
{
    db :: load Book from data base ;
    if(book not found)
    {
        print "Not Found Book";
    }
    show book Info ;
}
void getSearchHistory( stuName)
{
    db :: load student by his name from data base ;
    show his searched Books;
}
void updatePublisher( pubName,name,pass,email,cashAmount)
{
    db :: load publisher from data base ;
    publisher :: set Name;
    publisher ::set Password;
    publisher :: set Email;
    publisher ::set Cash;
    db :: save publisher;
}
void updateStudent(stuName,name, pass,email,cashAmount)
{
    db :: load student  from data base ;
    student :: set Name;
    student :: set Password;
    student :: set Email;
    student :: set Cash;
    db :: save student;
}
void studentLoggedIn(stuName)
{
    int choice;
    while(TRUE)
    {
        if(Search )
        {
            while(TRUE){
                if(Name)
                {
                    enter book name ;
                    if(not found) print " Your book is not available :( ";
                    else
                                                {
                                                          db :: load book by name;
                       book :: show book info
                    }
                    db:: add book to Search History of this student
                                       }
                else if(type)
                {
                    enter book type ;
                    if(not found) print " Your book is not available :( ";
                    else
                                                {
                                                          db :: load book by type;
                        book :: show book info
                    }
                    db:: add book to Search History of this student
                }
                else if(publisher)
                {
                    enter book publisher name ;
                    if(not found) print " Your book is not available :( ";
                    else
                                                {
                                                          db :: load book by publisher name ;
                        book :: show book info
                    }
                    db:: add book to Search History of this student
                }
                else if(price)
                {
                                                enter book price ;
                    if(not found) print " Your book is not available :( ";
                    else
                                                {
                                                          db :: load book by price ;
                        book :: show book info
                    }
                    db:: add book to Search History of this student
                }
                else if (exit)
                                                break;
                else
                                                print " Wrong choice , rakkez b2a mate2refnash";
            }
        }
        else if(Borrow)
        {
            controller :: borrow book;
        }
        else if(Return book)
        {
                            controller :: return book ;
                     }
        else if(view search history)
        {
            db :: load student;
                            student :: show book history ;
        }
        else if(view favorite books)
        {
            db ::  load student;
                            student :: show favorite history ;
        }
        else if(view his borrowed books)
        {
            db :: load student;
                            student :: show his borrowed books;
        }
        else if(edit profile)
        {
            db:: load student ;
            if(Name){
                                       enter name ;
                                       student :: set student name ;
                    }
          if(password)
                    {
                              enter password ;
                              student :: set student password ;
                    }
          if(email ){
                              enter email ;
                              student :: set student email ;
                    }
                    if(cash)
                    {
                              enter cash ;
                              student :: set student cash ;
                    }
                    db:: update Student ;

        }
        else if(Log Out)
          break;
        else
          print "Enter correct choice" ;
    }

}
```

Book Class

```
void setName( book name)
{
   Book :: set name of the book   ;
}
void setPrice(book price)
{
   Book :: set price of the book;
}
void setType(Type)
{
   Book :: set type of the book;
}
void setPublisher(myPublisher)
{
        Book :: set the publisher of the book;
}
void setImagePath(path)
{
   Book :: set path of book image;
}
void setState(State)
{
   Book :: set the state of the book;
}
void setAvailability(Availability)
{
   Book :: set the existence  of the book;
}
void setBorrowedDate(date)
{
   Book :: set the borrow date of the book ;
}
void setExpectedReturnDate(date)
{
   Book :: set the expected return date of the book ;

void setActualReturnDate(date)
{
   Book :: set return of the book;
}

void setRowId(Rowid)
{
   Book :: set ID of the book according to data base table of books;
}
string getRowId()
{
   Book :: returns ID of the book in data base table of books  ;
}
string getName()
{
   Book :: returns the name of the book;
}
int getPrice()
{
   Book :: returns price of the book;
}
string getType()
{
   Book :: returns the type of the book;
}
Publisher* getPublisher()
{
   Book :: returns the publisher of the book class ;
}
string getImagePath()
{
   Book :: returns image path of the book;
}
bool getState()
{
   Book :: checks the state of the book;
}

bool getAvailability()
{
   Book :: checks the availability of the book;
}
int getBorrowedDate()
{
   Book :: returns the borrow date class of the book;
}
int getExpectedReturnDate()
{
   Book :: returns the expected return date class of the book;
}
int getActualReturnDate()
{
    Book :: returns the actual return date class of the book;
}

void showInfo()
{
        print book name ;
        print book price ;
        print book state ;
        print book type;
        print book publisher ;
}
```

Student Class

```
void addSearchHistory(vector<string> x)
{
        push back the book name in the vector  ;
}

void addFavoriteBooks(vector<string> x)
{
        push back the book name in the vector  ;
```

```
}

void Student::showInfo()
{
          print student  name& his borrowed books & his search history
                    & his favorite Books & his current borrow book & his requested books ;
}

void setCurrentBook(name)
{
   set student current book ;
}

void Student::setRequestedBook(string name)
{
   set student requested book ;
}

void addBorrowedBooks(vector<string> x)
{
   push borrowed book in the vector ;
}
string getCurrentBookName()
{
   gets current Book name;
}

vector<Book> getCurrentBookVector(DataBase *db)
{
   load all current books from data base ;
}

string getRequestedBookName()
{
   get requested Book name;
}

vector<string> getSearchHistory()
{
   gets student search History names;
}
User:
User (name , email , password)
{
  set Name  ;
  set Email ;
```

```
   set Password  ;
}
void addCash(cash)
{
    add new cash to the old cash ;
}

void setCash(cash)
{
    change the all cash of the user to new value;
}
string getName()
{
    gets user Name ;
}
string getEmail()
{
    gets user email ;
}
string getPassword()
{
    gets user password ;
}

int getCash()
{
    gets cash amount  ;
}

void User::showInfo()
{
          print user's Name ;
          print user's Email ;
          print user's cash amount ;
}
LibrarianClass:


Book supplyRequestedBook(stuName,db)
{
    db :: load Requested Book Names of student ;
    book :: set book Name ;
    print "Ya librarian bsha, enter book's type : ";
    enter book type ;
    print "w ma3lesh kman, enter book's price : ";
    enter book price;
```

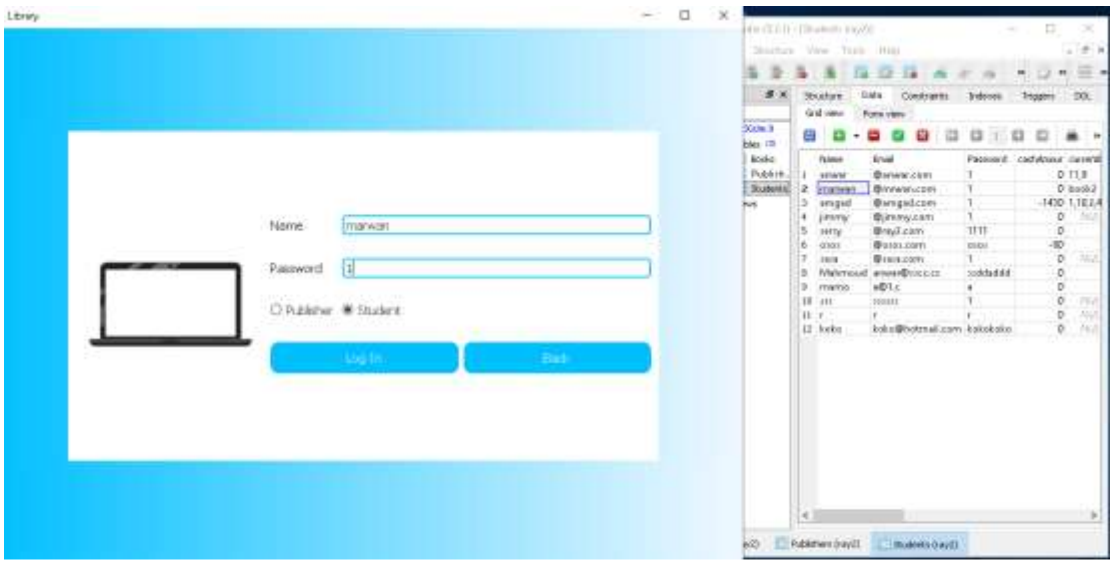
```
        db :: save Book;
    student :: remove requested books ;
}

void libBookState(int libBookState)
{

    set bookState;
}
```

15. TESTING

Top-Down Component interaction Testing:

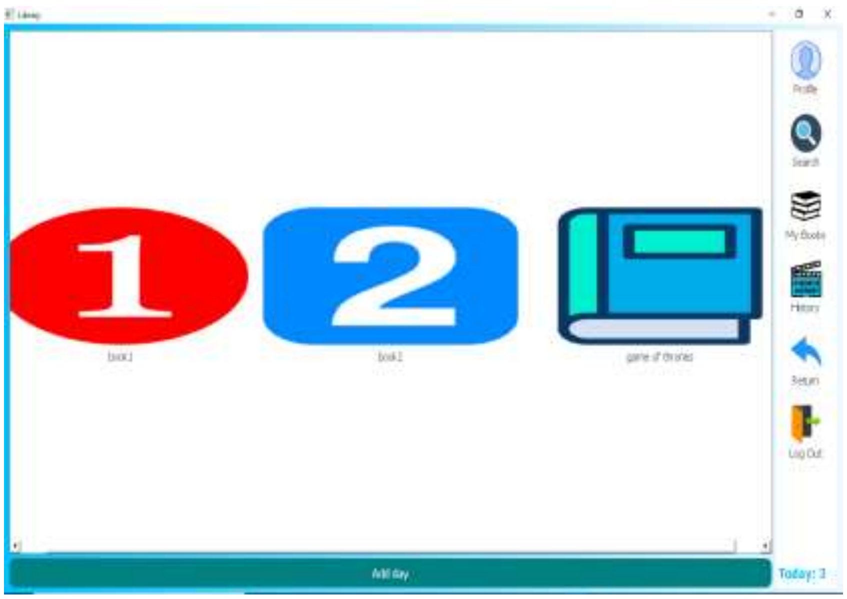
- Involves building a system from its components and testing it for problems that arise from component interactions.
- Testing the program starting the index screen in random sequence.

1. **<u>Testing Student log in:</u>**
   Showing existing data in database.



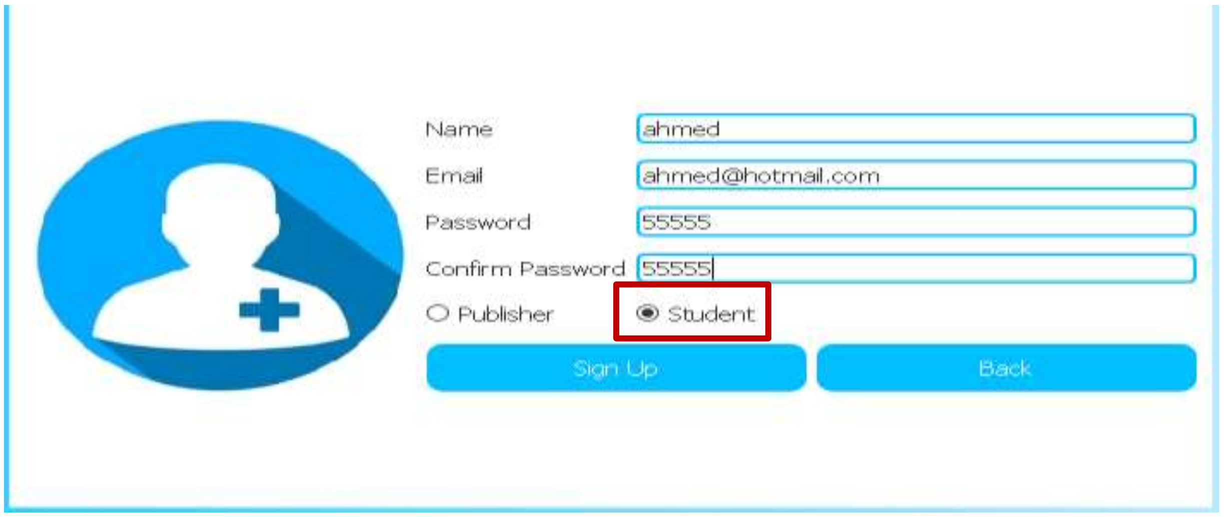
Name: marwan
Password: 1
( ) Publisher (● Student)

	Name	Email	Password	cashAmount	current
1	anwar	@anwar.com	1	0	11,8
2	marwan	@marwan.com	1	0	book2
3	amgad	@amgad.com	1	-1400	1,10,2,4
4	jimmy	@jimmy.com	1	0	NULL
5	jerry	@ray3.com	1111	0	
6	oooo	@oooo.com	oooo	-80	
7	rara	@rara.com	1	0	NULL
8	Mahmoud	anwar@cccc.cc	[illegible]	0	
9	mamo	a@1.c	a	0	
10	zzz	[illegible]	1	0	NULL
11	r	r	r	0	NULL
12	koko	koko@hotmail.com	kokokoko	0	NULL



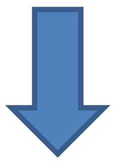
2. **<u>Student Sign up:</u>**

Showing adding data in database.



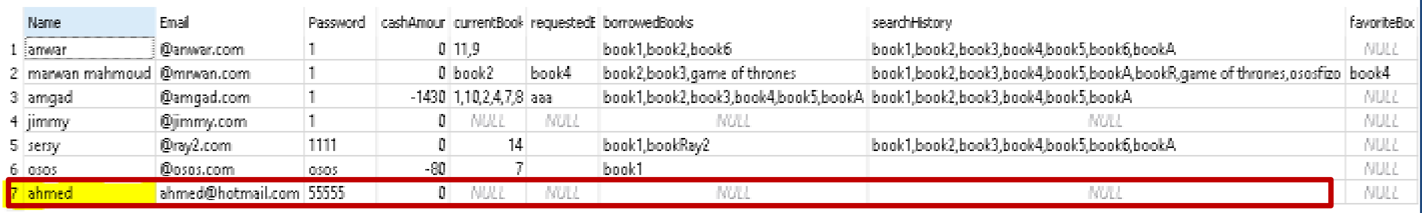
Name	ahmed
Email	ahmed@hotmail.com
Password	55555
Confirm Password	55555

○ Publisher [✓ Student]



After signing up successfully data

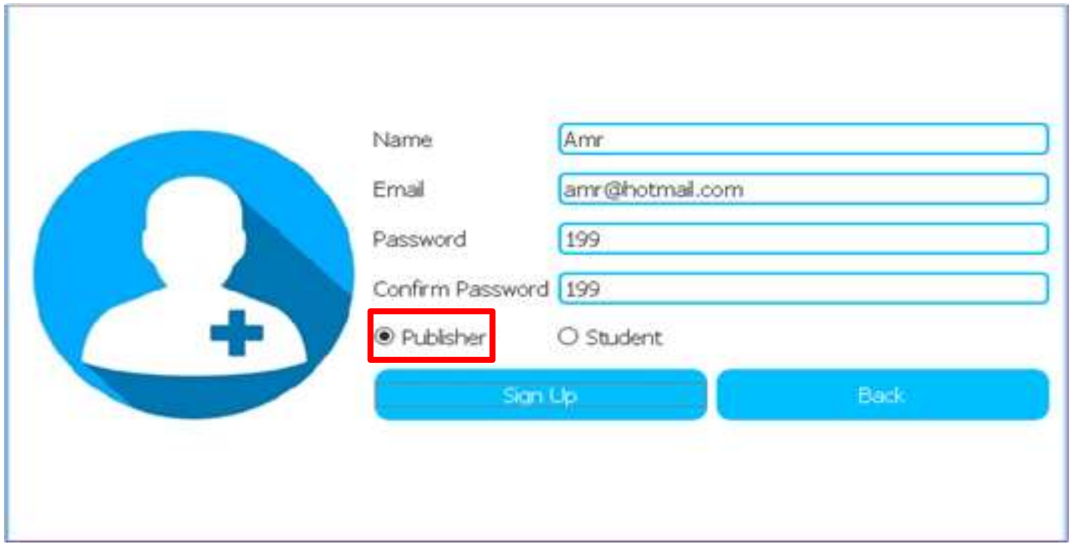
Is added to database.



	Name	Email	Password	cashAmour	currentBook	requestedE	borrowedBooks	searchHistory	favoriteBo
1	anwar	@anwar.com	1	0	11,9		book1,book2,book6	book1,book2,book3,book4,book5,book6,bookA	NULL
2	marwan mahmoud	@mrwan.com	1	0	book2	book4	book2,book3,game of thrones	book1,book2,book3,book4,book5,bookA,bookR,game of thrones,ososfizo	book4
3	amgad	@amgad.com	1	-1430	1,10,2,4,7,8	aaa	book1,book2,book3,book4,book5,bookA	book1,book2,book3,book4,book5,bookA	NULL
4	jimmy	@jimmy.com	1	0	NULL	NULL	NULL	NULL	NULL
5	sersy	@ray2.com	1111	0	14		book1,bookRay2	book1,book2,book3,book4,book5,book6,bookA	NULL
6	osos	@osos.com	osos	-80	7		book1		NULL
7	ahmed	ahmed@hotmail.com	55555	0	NULL	NULL	NULL	NULL	NULL

3. Publisher Signup:

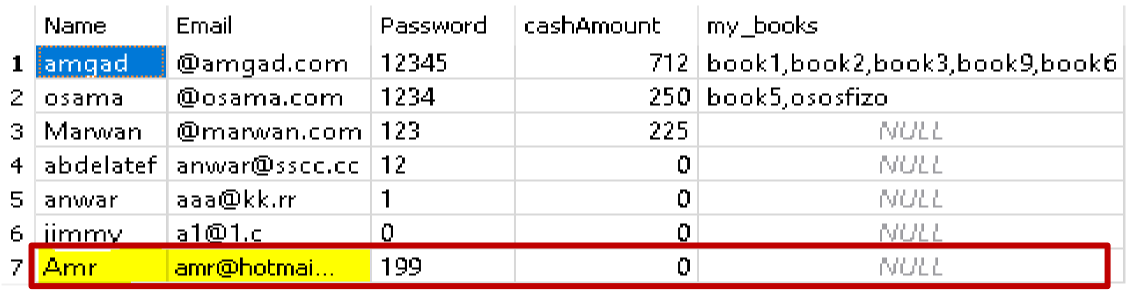
Publisher can sign up by choosing publisher
In signup screen.



Name	Amr
Email	amr@hotmail.com
Password	199
Confirm Password	199

[✓ Publisher] Student

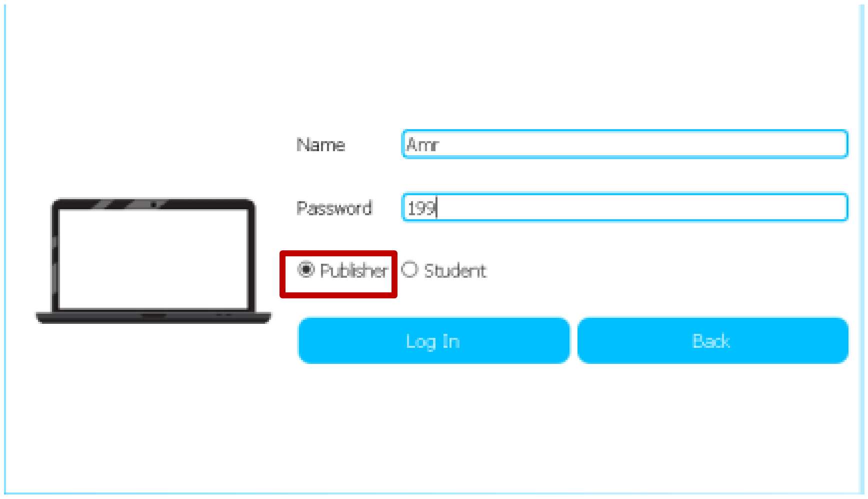
After signing up data is add to database.



	Name	Email	Password	cashAmount	my_books
1	amgad	@amgad.com	12345	712	book1,book2,book3,book9,book6
2	osama	@osama.com	1234	250	book5,ososfizo
3	Marwan	@marwan.com	123	225	NULL
4	abdelatef	anwar@sscc.cc	12	0	NULL
5	anwar	aaa@kk.rr	1	0	NULL
6	jimmy	a1@1.c	0	0	NULL
7	Amr	amr@hotmai...	199	0	NULL

4. Publisher login

Publisher can login by choosing publisher in login window as shown.



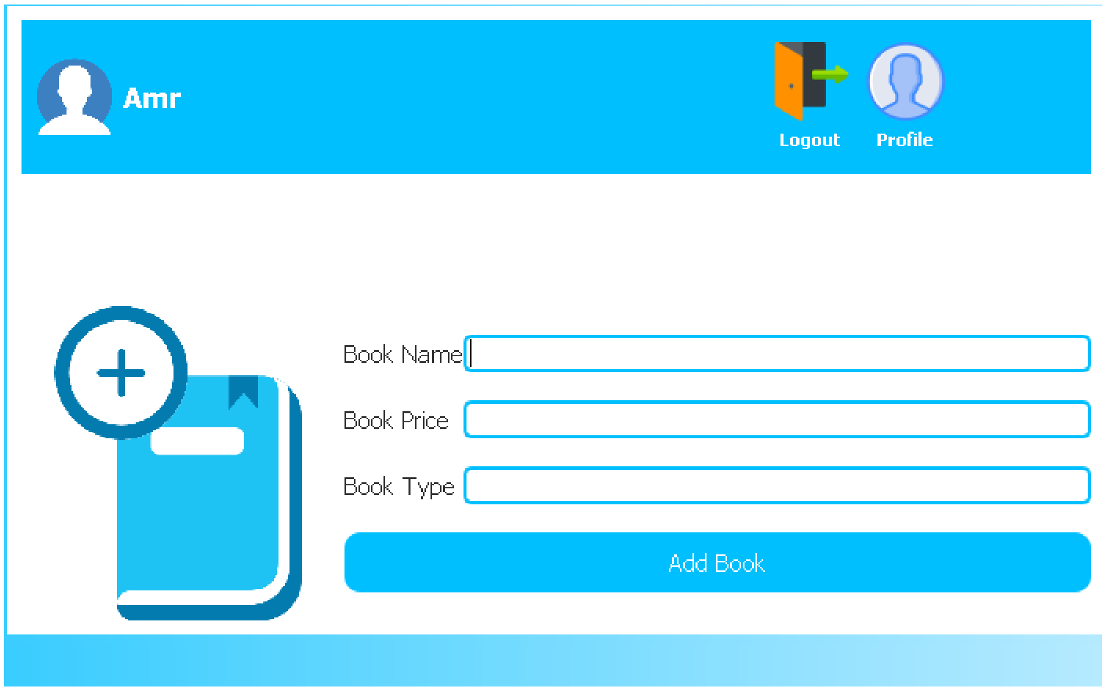
Name: Amr

Password: 199

[✓ Publisher] Student

Login Successfully

.



Amr

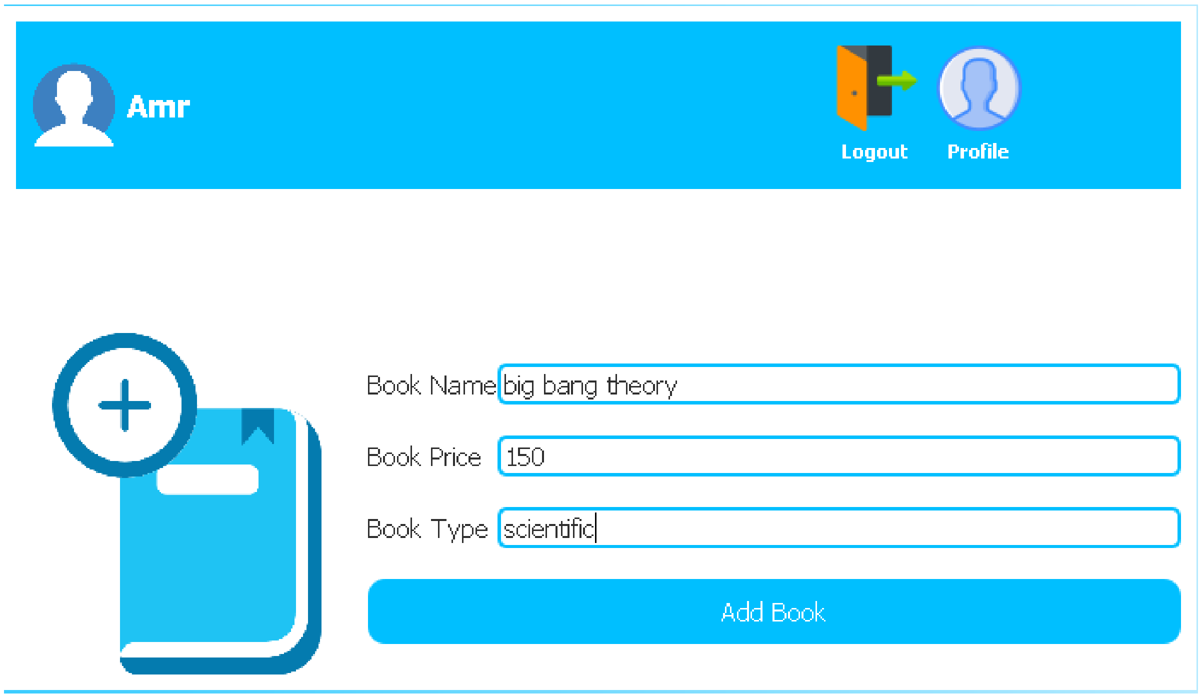
Book Name:

Book Price:

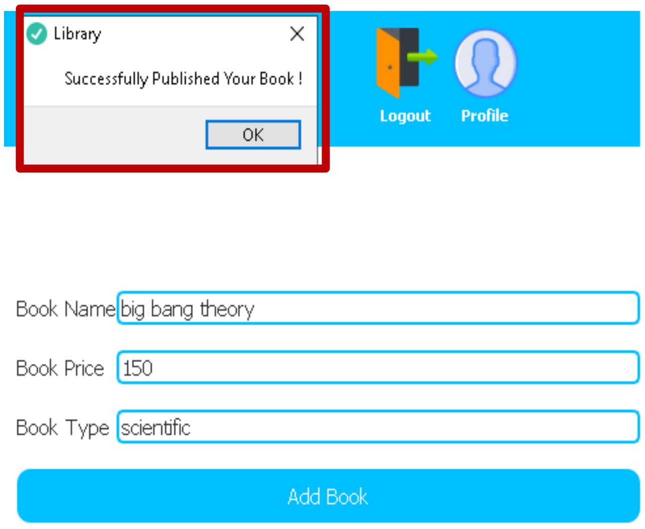
Book Type:

Profile Window Is opened so he can add books, Edit his profile data.

5. Publisher add book:
Publisher can add new books by entering name, price and the type of the book.



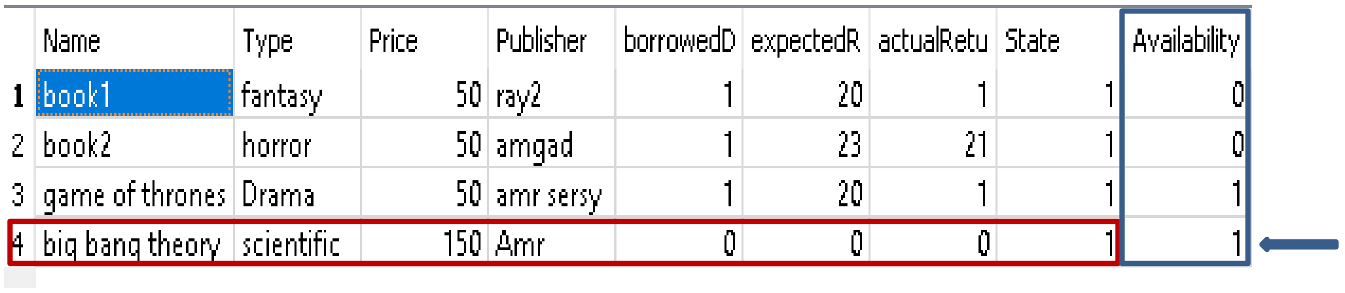
Book Name: big bang theory
Book Price: 150
Book Type: scientific



Successfully Published Your Book !

Book Name: big bang theory
Book Price: 150
Book Type: scientific

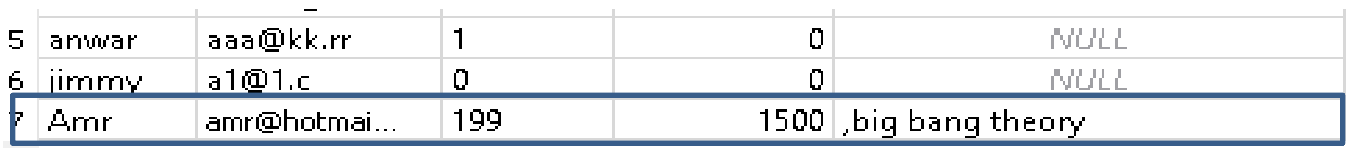
After pressing add book, the book Is added to database.



	Name	Type	Price	Publisher	borrowedD	expectedR	actualRetu	State	Availability
1	book1	fantasy	50	ray2	1	20	1	1	0
2	book2	horror	50	amgad	1	23	21	1	0
3	game of thrones	Drama	50	amr sersy	1	20	1	1	1
4	big bang theory	scientific	150	Amr	0	0	0	1	1

Availability of the book becomes 1.

The book is then updated in database for the Publisher.



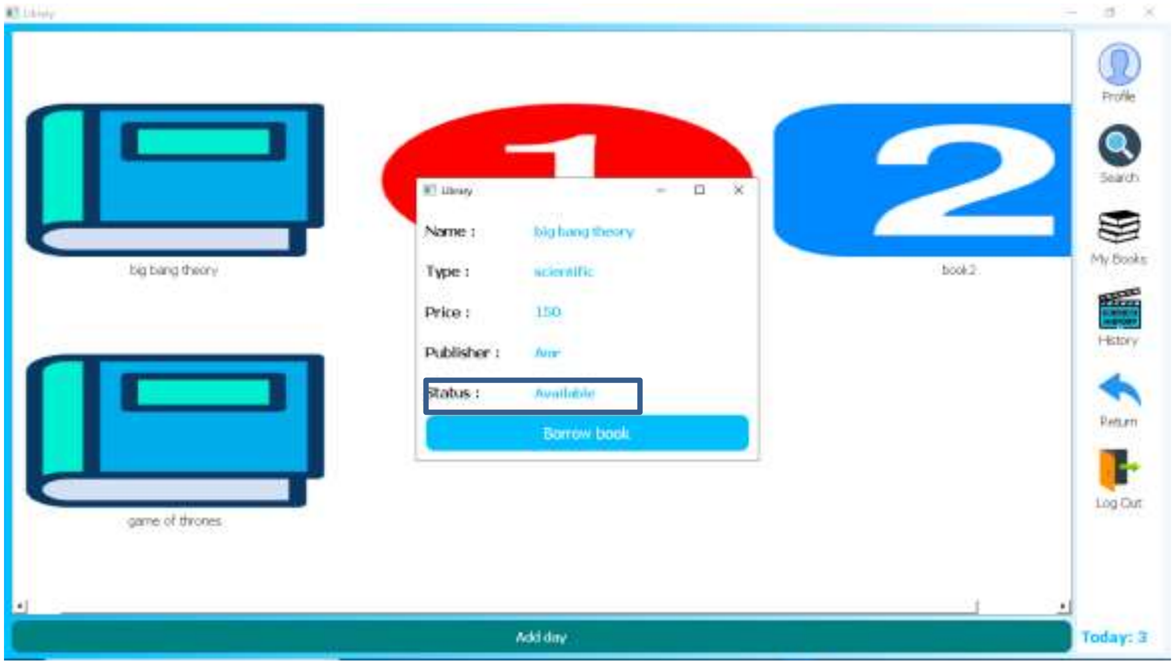
5	anwar	aaa@kk.rr	1	0	NULL
6	jimmy	a1@1.c	0	0	NULL
7	Amr	amr@hotmai...	199	1500	,big bang theory

6. Student Borrow book

Student can borrow book by entering its name and see if it is available or not.

-The book is available now in the library.

-status of the book shows if it is available or not.



Name : big bang theory

Type : scientific

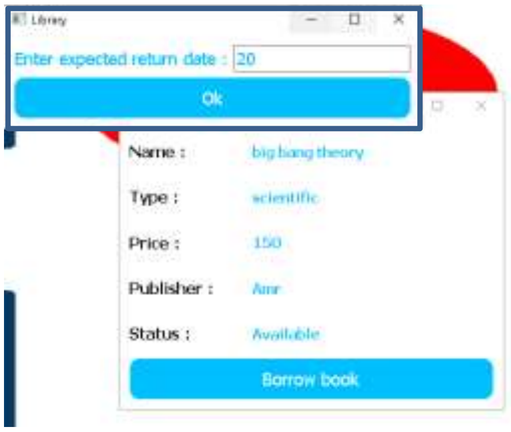
Price : 150

Publisher : Amr

Status : Available

After pressing borrow book Student should

Enter return date.



Enter expected return date : 20

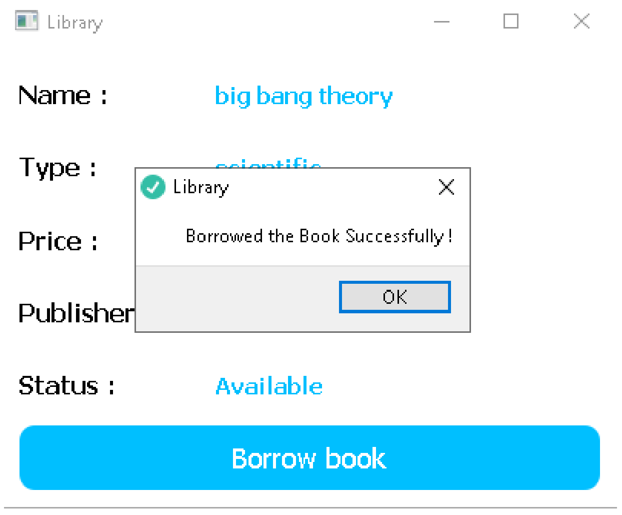
Name : big bang theory

Type : scientific

Price : 150

Publisher : Amr

Status : Available


Library

Name : big bang theory

Type : scientific

Price :

Publisher

Status : Available

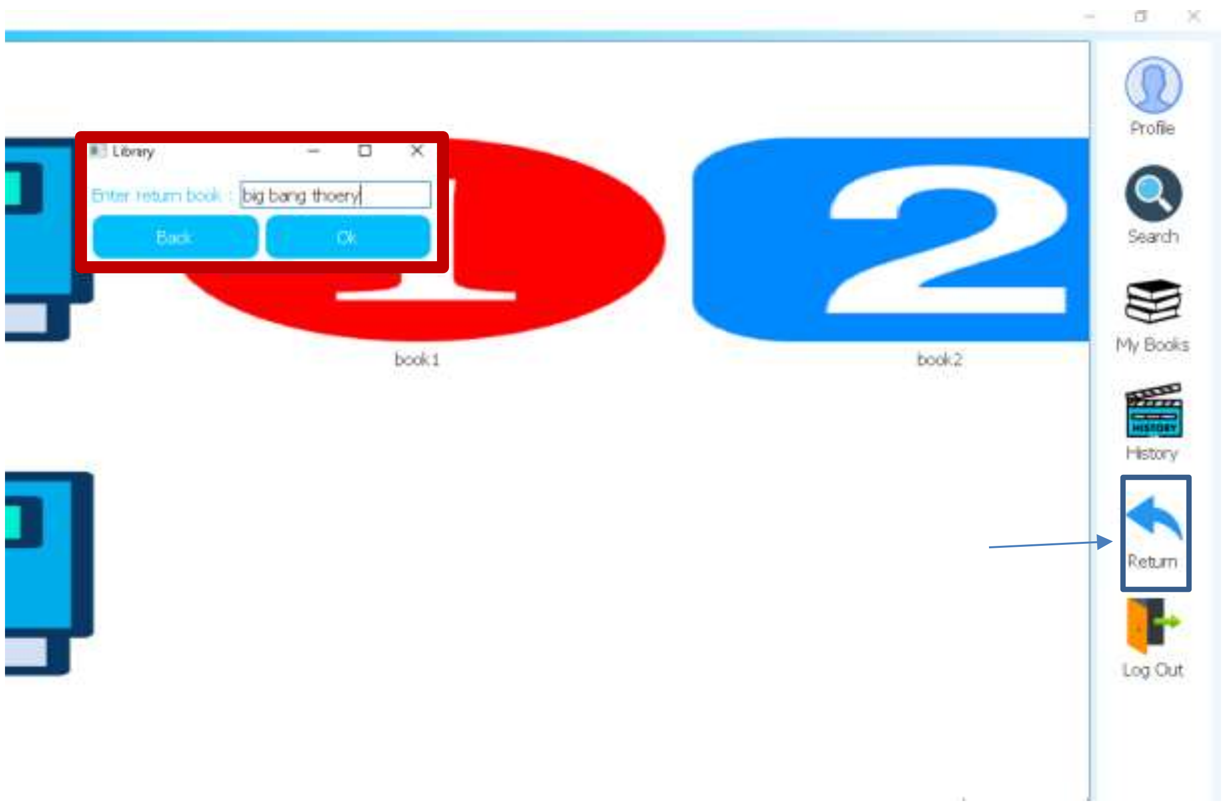
Library

Borrowed the Book Successfully !

After entering expected return date borrowed successes and availability of the book become 0 updated in the database.

7. Student return book.

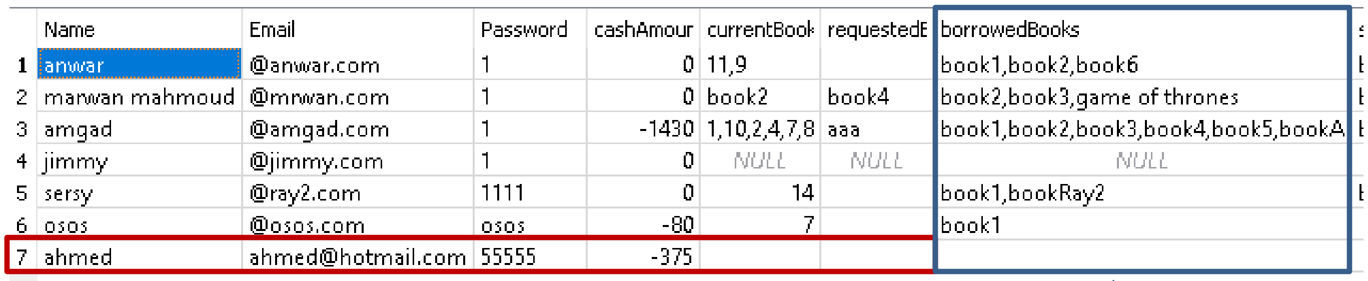
Student can return borrowed book by clicking return and entering borrowed book's name.



After pressing ok, it shows message with

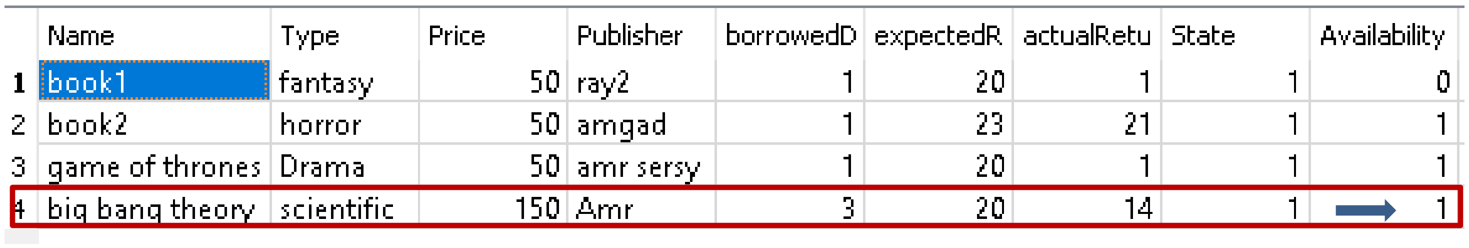
The bill, and book is removed from

Student's borrowed books



	Name	Email	Password	cashAmour	currentBook	requestedE	borrowedBooks	s
1	anwar	@anwar.com	1	0	11,9		book1,book2,book6	b
2	marwan mahmoud	@mrwan.com	1	0	book2	book4	book2,book3,game of thrones	b
3	amgad	@amgad.com	1	-1430	1,10,2,4,7,8	aaa	book1,book2,book3,book4,book5,bookA	b
4	jimmy	@jimmy.com	1	0	NULL	NULL	NULL	
5	sersy	@ray2.com	1111	0	14		book1,bookRay2	b
6	osos	@osos.com	osos	-80	7		book1	
7	ahmed	ahmed@hotmail.com	55555	-375				

Book availability in database become



	Name	Type	Price	Publisher	borrowedD	expectedR	actualRetu	State	Availability
1	book1	fantasy	50	ray2	1	20	1	1	0
2	book2	horror	50	amgad	1	23	21	1	1
3	game of thrones	Drama	50	amr sersy	1	20	1	1	1
4	big bang theory	scientific	150	Amr	3	20	14	1	1

Testing summary:

Module Testing

Module test went successfully on the security password check.

- Login was successful when entering the correct password.
- Login failed while entering any other password.

Integration Testing

Integration testing went through for 2 scenarios.

- Adding a new book to the library.
- Displaying the available books.

System Testing

System successfully checked to fulfill the meant purpose.

16. Estimated Project Cost

16.1 Estimated Project Cost

- The library system consists of 4K lines of code.
- According to cocomo 2 early design models, **PM = A × Size^B × M**
- A is a constant of 2.94
- Size is 4KLs
- B= 1.1
- Multipliers table

Effort Adjustment Factors

RCPX	**1.6**
RUSE	**1.5**
PDIF	**1.5**
PREX	**1.6**
PERS	**1.6**
SCED	**1.5**
FCIL	**1.8**

Estimated multipliers table to find M

PM = 330 Person-month

Post-Architecture Stage:

The estimation process is the same as those in the early design stage. The only difference is that it has more EAF to estimate.

16.2 Estimated Project Cost:

- 1 project manager(Mostafa Amgad) (5 thousand pounds)
- 2 developers (AmrElsersy and Mostafa Gamal)(3 thousand pounds forr each one)
- Hardware and software (MostafaAhmed)(3 thousand pounds for each one)
- 2 testers (Marwan and Mahmoud Anwar) (3 thousand pounds for each one)
- Designer (Mahmoud Osama)(10 thousand pounds)

Effort Overhead:

- Office rent (1 thousand pounds)
- Drinks and launch (1 thousand pounds)
- Heating and electricity (1 thousand pounds)

Travel & Transports:

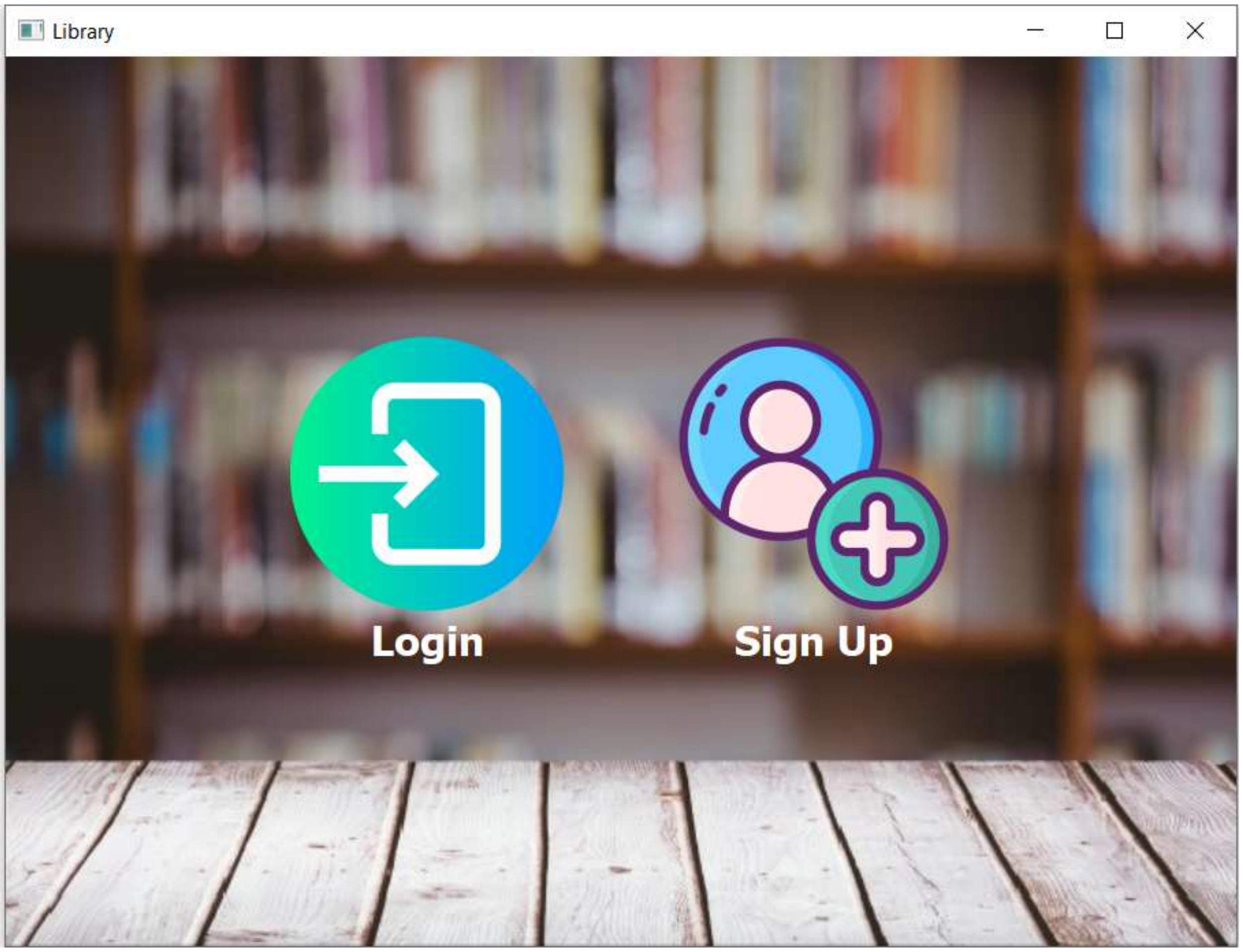
- 3 developers (250 pounds each)

Total:

- Estimated cost is 10, 2 Million pounds.
- Selling price is 11 Million.

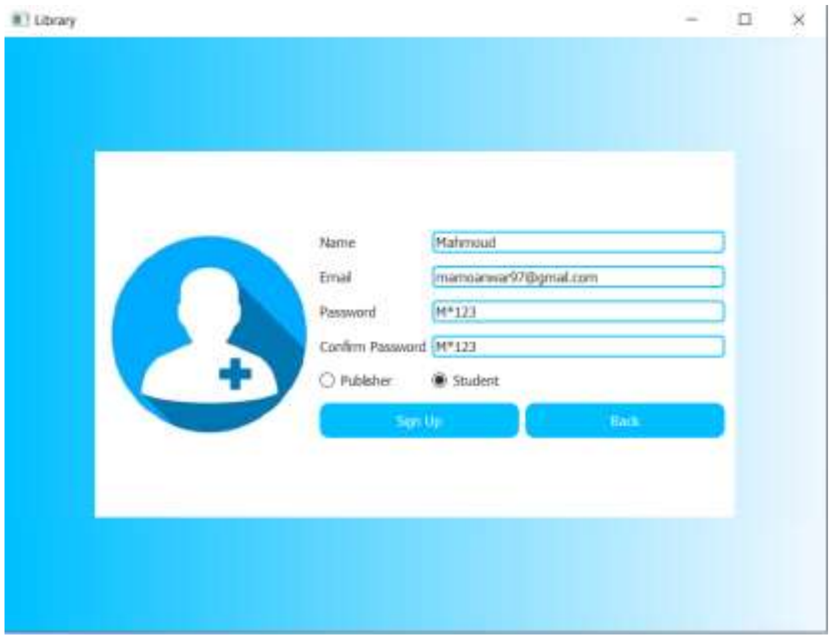
17. User guide.

Welcome To the library system



Registration:

IF you are a new user you can Sign up then you can create a Student or Publisher account:

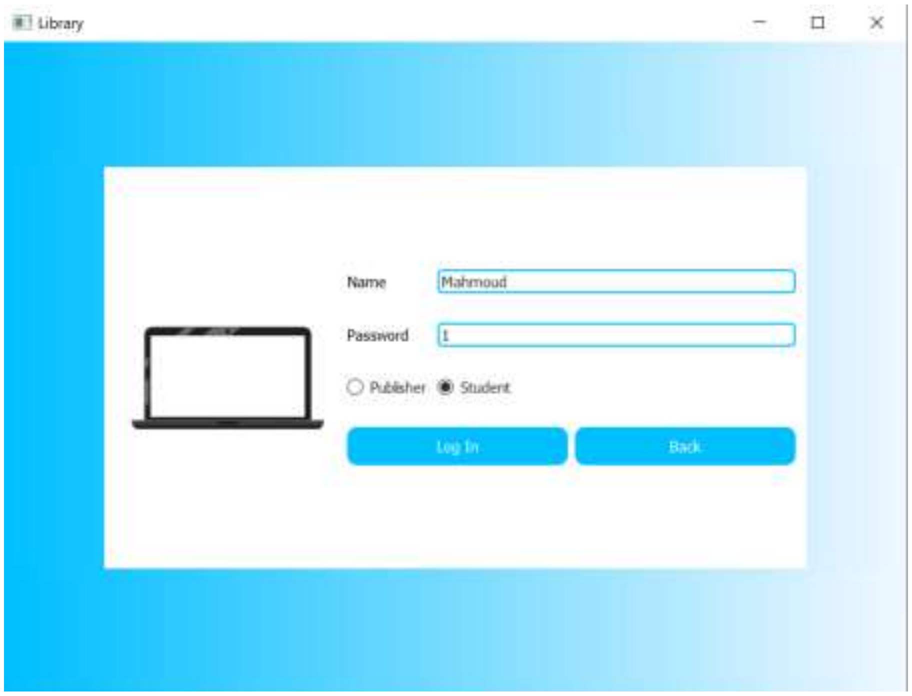


Name	Mahmoud
Email	mamoanwar97@gmail.com
Password	M*123
Confirm Password	M*123

Publisher [✓ Student]

Figure 3 Sign up form

IF you already have an account, then you can log in with your mail and password, and we shall say Welcome Home :D



Name	Mahmoud
Password	1

Publisher [✓ Student]

Figure 4 Log in form

Publisher Account:

Welcome to Publisher account, after login/sign up as Publisher this window appear where you can:

1. Publish (Add) a new book.
2. Edit profile
3. Log out

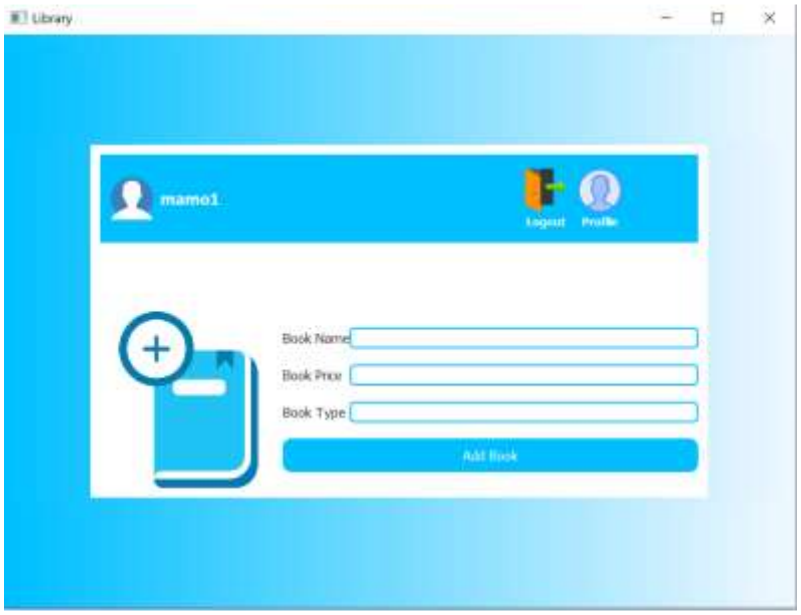
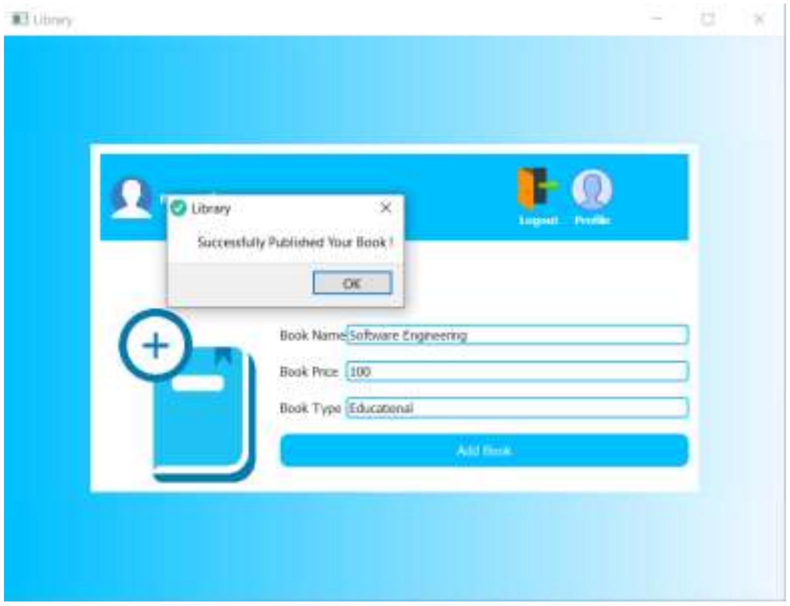


Figure 5 Publisher Account

Add book:

To add a new book:

1. Fill all the information about the book
2. Click add Book
3. A confirmation Message will appear if you successfully done every step (As in Figure 4).



Library

Successfully Published Your Book !

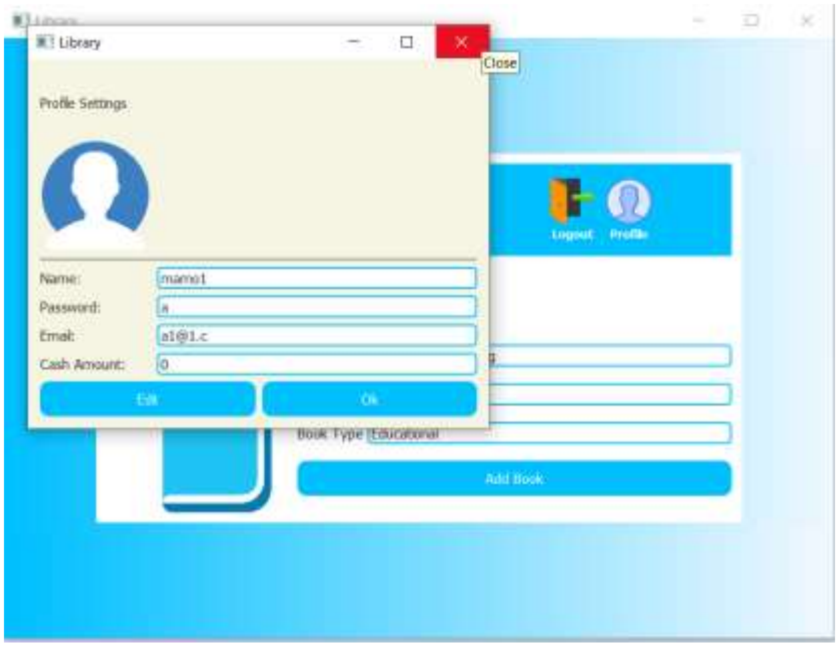
Book Name: Software Engineering
Book Price: 100
Book Type: Educational

Figure 6 Add a new book

Edit profile:

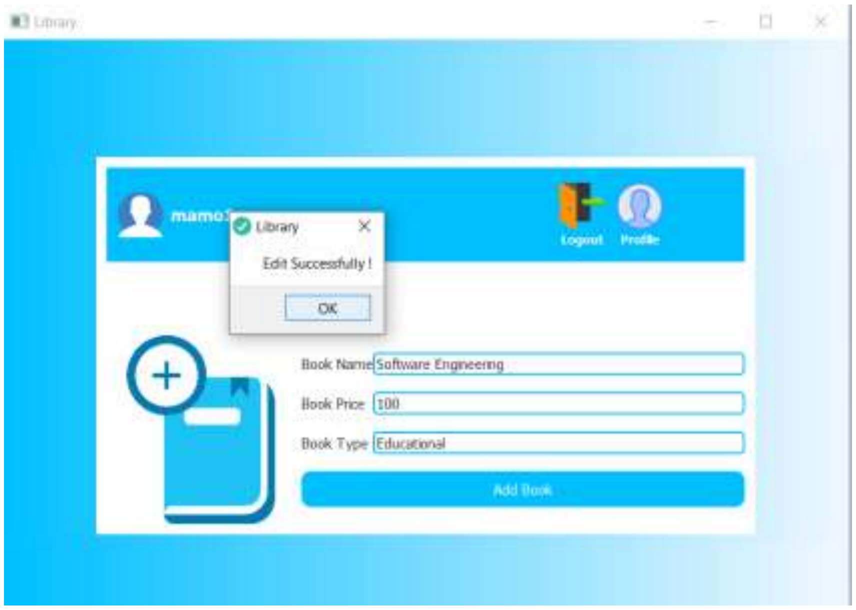
To edit your publisher profile:

1. Click Profile button.
2. A new window will appear with profile information (As in Figure 6).
3. To edit it, choose Edit.
4. Re-fill the new info about the publisher profile.
5. Press okay to save the information.
6. A confirmation message will appear till you that everything went right.



Name: mamo1
Password: a
Email: a1@1.c
Cash Amount: 0
Book Type Educational

Figure 8 Profile Widget



Edit Successfully !

Book Name Software Engineering
Book Price 100
Book Type Educational

Figure 7 Successfully updated profile information

Student Account:

Welcome to Student account, after login/sign up as Student this window appear where you can:

1. Edit profile.
2. Borrow a book.
3. Return a book.
4. Search for a book.
5. Show borrowed books.
6. Show Search history.
7. Log out

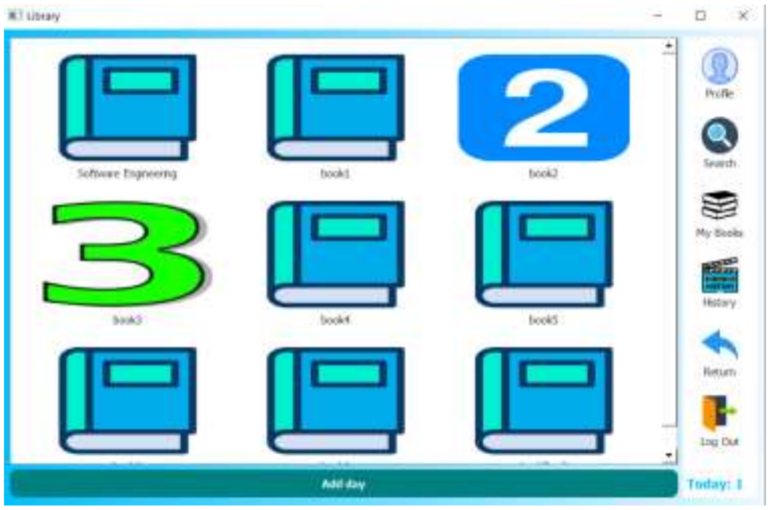
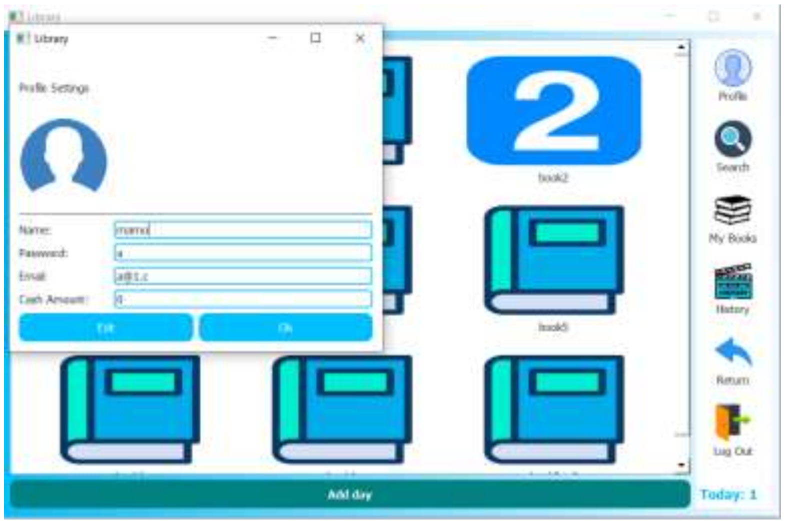


Figure 9 Student account

Edit profile:

To edit profile:

1. Click Profile button.
2. A new window will appear with profile information (As in Figure 8).
3. To edit it, choose Edit.
4. Re-fill the new info about the publisher profile.
5. Press okay to save the information.
6. A successful message will appear when everything goes right.



Name: mamo
Password: a
Email: a@1.c
Cash Amount: 0

Figure 10 Edit profile Information

Search:

To Search for a book:

1. Click on the search icon.
2. Search widget will appear (As in Figure 9).
3. You can search for a book by its Name, Type, Price, Publisher
4. After you choose the Search filter the information of targeted book will appear (As in Figure 10).
5. You can click on borrow book if you want the book
6. Or just click done if you got the info you need

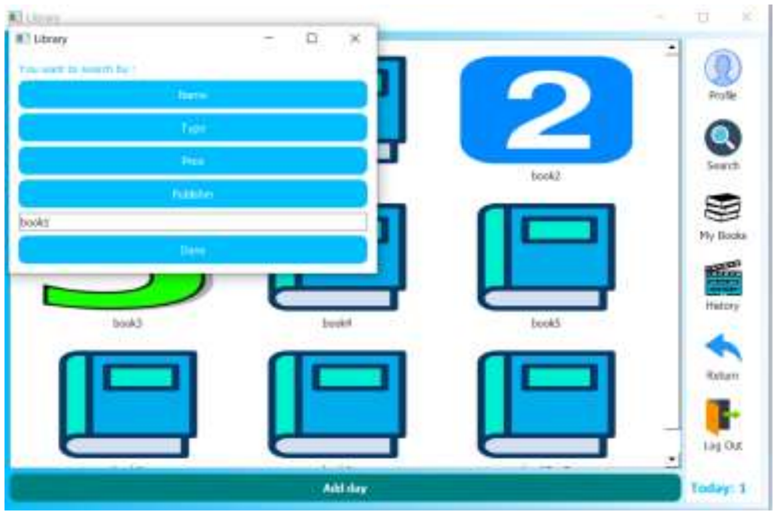
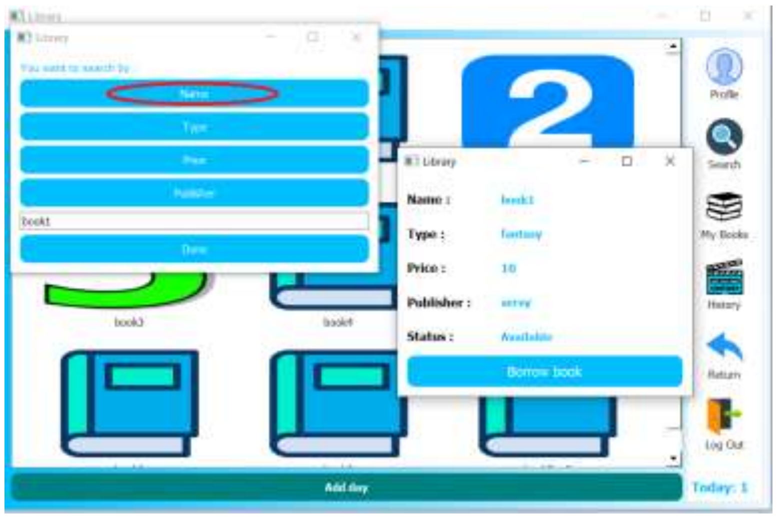


Figure 11 Search widget



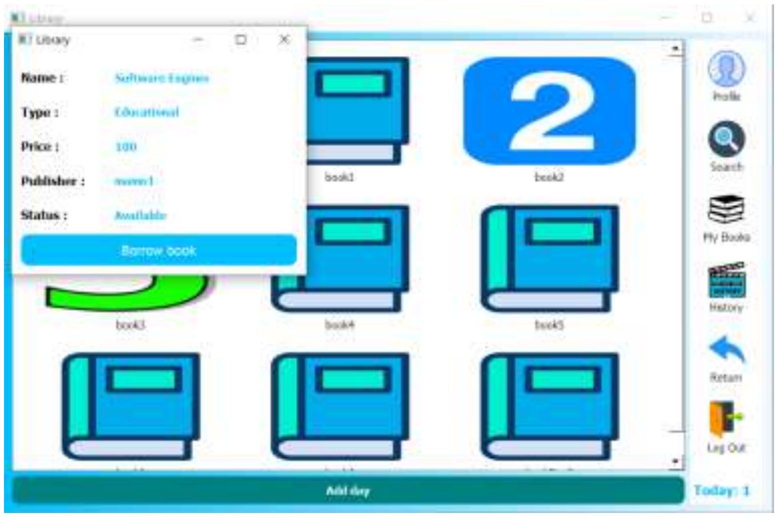
Name : book1
Type : fantasy
Price : 10
Publisher : [illegible]
Status : Available

Figure 12 Filter Search by book name

Borrow book:

To borrow a book:

1. You can search for it by clicking on search icon or just choose the book you want to borrow.
2. A new window with the book information will appear (As in Figure 11).
3. If the book status is not available then we are sorry you can Request it or Just wait until a new batch arrive, or someone return his used book.
4. If the book status is Available then you can Borrow it, click on borrow book.
5. A new window will appear asking for the return date (As in Figure 12).
6. Then press okay, then we hope a happy reading for you and wish that you enjoy the book.



Name : Software Engineer

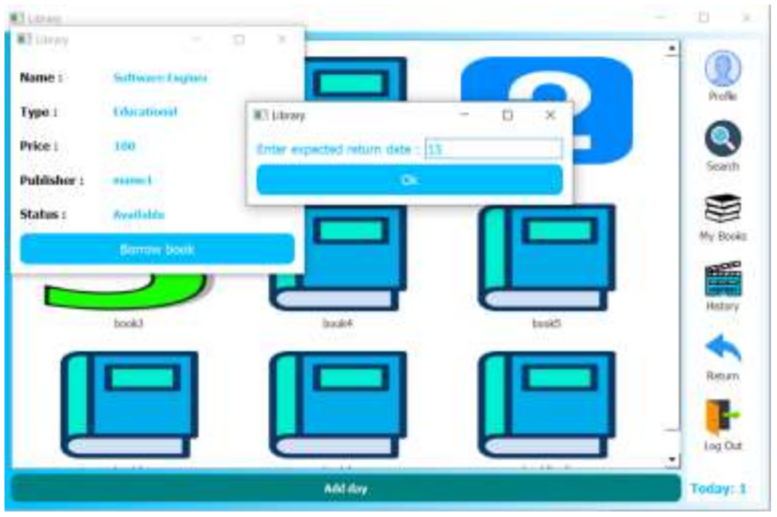
Type : Educational

Price : 100

Publisher : name1

Status : Available

Figure 13 Book Information widget



Name : Software Engineer

Type : Educational

Price : 100

Publisher : name1

Status : Available

Enter expected return date : 15

Figure 14 Borrow an available book

Return book:

To return a Borrowed book:

1. Click on return icon.
2. A Return widget will appear (As in Figure 13).
3. Specify the name of the book you want to return then press ok
4. If the book in a bad condition, you will pay extra insurance fees.

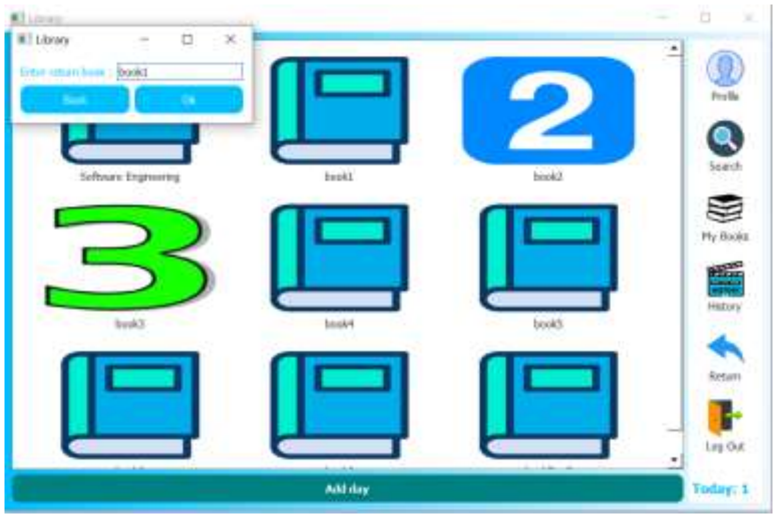
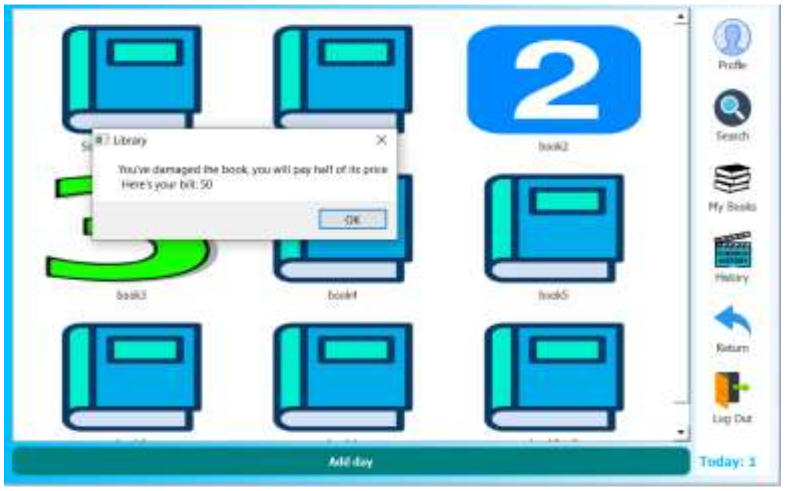


Figure 13 Return book widget



Library

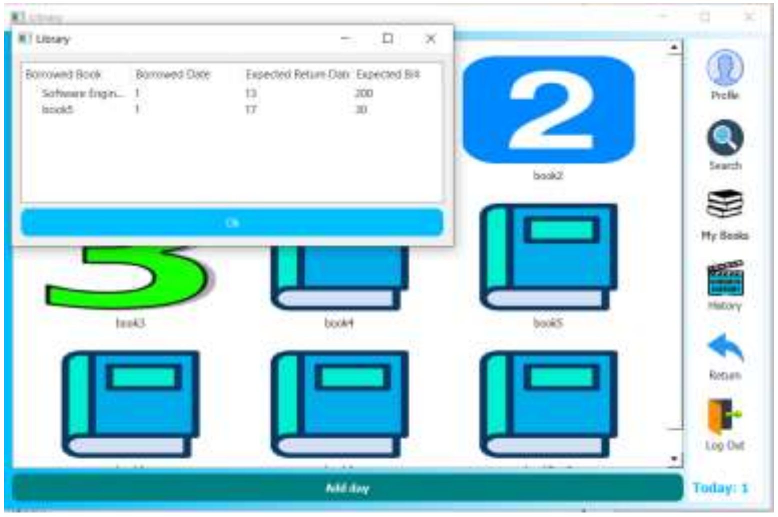
You've damaged the book, you will pay half of its price
Here's your bill: 50

Figure 14 Extra fees for Damaged books

Borrowed books:

To Show your Borrowed books log:

1. Click on My books from the right panel
2. A Table of information about all the borrowed books you have at that moment (As in Figure 15).



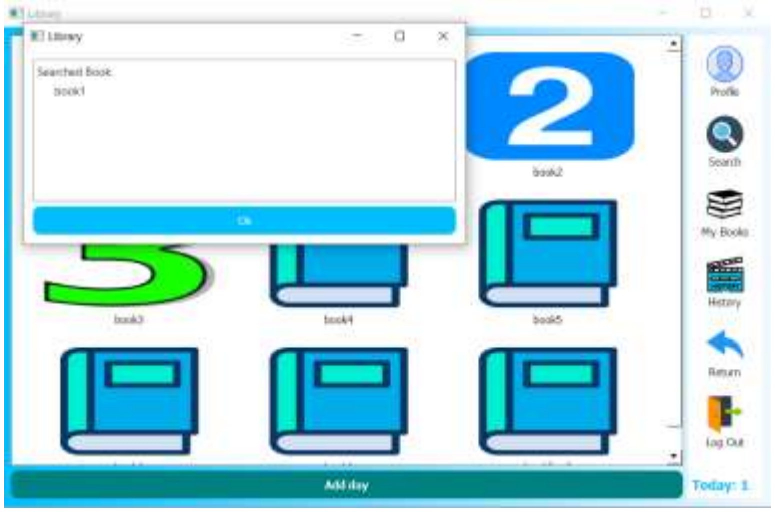
Borrowed Book	Borrowed Date	Expected Return Date	Expected Bill
Software Engin...	1	13	200
book5	1	17	30

Figure 15 Borrowed books Widget

Search history:

To Show your Search History:

3. Click on History from the right panel
4. A List of names of the books that you searched for will appear (As in Figure 16).



Searched Book
book1

Figure 16 Search History Widget