



**TRIBHUVAN UNIVERSITY  
INSTITUTE OF ENGINEERING  
PULCHOWK CAMPUS**

**LIBRARY MANAGEMENT SYSTEM**

A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND  
COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE PRACTICAL COURSE ON  
OBJECT ORIENTED PROGRAMMING [CT 451]

**Submitted by:**

Kshitiz Pandey	(PUL076BEI019)
Pramesh Shrestha	(PUL076BEI025)
Rahul Shrestha	(PUL076BEI028)
Sandesh Sitaula	(PUL076BEI035)

**Submitted to:**

Department of Electronics and Computer Engineering, Pulchowk Campus  
Institute of Engineering, Tribhuvan University  
Lalitpur, Nepal

## **ABSTRACT**

Library management system is a project which aims in developing a computerized system to maintain all the daily works of library. This project has many features which are generally not available in normal library management systems like facility of student login. It also has a facility of admin login through which the admin can monitor the whole system. It has also a facility where student after logging in their accounts can see list of books issued and its issue date and return date and also the students can request the librarian to add new books. The admin after logging into his account can generate various reports such as student report, issue report, and book report. Overall, this project of ours is being developed to help the students as well as staffs of library to maintain the library in the best way possible and also reduce the human efforts.

## **ACKNOWLEDGEMENT**

We extend our sincere and heartfelt thanks to our esteemed guide and respected teacher Mr. Bikal Adhikari, for providing us with the right guidance and advice at the crucial junctures and for showing us the right way. We would like to thank the other faculty members also, at this occasion. As well as our friends and family for the support and encouragement they have given us during the course of our work. Last but not the least, we would like to thank stackoverflow and those Qt discussion forum without whom the project wouldn't have seen the light of the day.

## TABLE OF CONTENTS

FORMAT OF COVER PAGE	I
ABSTRACT	II
ACKNOWLEDGEMENT	III
TABLE OF CONTENTS	IV
LIST OF FIGURES	VII
LIST OF TABLES	VIII
Chapter One - INTRODUCTION	1
1.1. Background of Project	1
1.2. Objectives	1
1.3. Limitations	2
Chapter Two – SYSTEM ANALYSIS	3
2.1. Requirement Specification	3
2.2. Feasibility	8
Chapter Three: SYSTEM DESIGN	9
3.1 Table Design	9
3.2. Algorithm	11
3.2.1. Algorithm for Login	11
3.2.2. Algorithm for Check-In	11
3.2.3. Algorithm of Renew	12
3.2.4. Algorithm for Check-Out	12
3.2.5. Algorithm for Add page	13

3.2.6. Algorithm for Edit/Delete	13
3.2.7. Algorithm for student page	14
3.3. Flowchart	14
3.3.1. Flowchart for Login	14
3.3.2. Flowchart for Admin Check-Ins	15
3.3.3. Flowchart for Renew	15
3.3.4. Flowchart for Check-Out	16
3.3.5. Flowchart for Add page	16
3.3.6. Flowchart for Edit/Delete	17
3.3.7. Flowchart for Student page	17
3.4. UML Diagrams	18
Chapter Four: SYSTEM TESTING	20
4.1. Test for Admin Module	20
4.2. Test for Student Module	21
Chapter Five: IMPLEMENTATION AND CODING	22
5.1. Overview	22
5.2. The database	23
5.3. Coding of the project	25

Chapter Six: RESULTS AND DISCUSSION	34
Chapter Six: CONCLUSIONS	35
REFERENCES	36
APPENDIX A (Header files)	37
APPENDIX B (Source files)	57
APPENDIX C (JSON file)	98

## **LIST OF FIGURES**

3.3.1.1. Flowchart for Login

3.3.1.2. Flowchart for Admin Check-Ins

3.3.1.3. Flowchart for Renew

3.3.1.4. Flowchart for Check-Out

3.3.1.5. Flowchart for Add page

3.3.1.6. Flowchart for Edit/Delete

3.3.1.7. Flowchart for Student page

3.3.2. UML Diagram

5.1. Overview

5.2. The database

5.3.A. The login page

5.3.a. Student detail page

5.3.b. Issue detail page

5.3.c. Search book page

5.4. Check-In page

5.5. Renew book page

5.6. Check-Out page

5.7. Add new data page

5.8. New book data

5.9. New student data

5.10. Edit/Delete data page

## **LIST OF TABLES**

- 3.1.1. Book table to keep book information
- 3.1.2. Student table to keep student information
- 3.1.3. Admin table to keep admin information
- 3.1.4. Issue books table for book issue record
- 3.1.5. Student login table



## **Chapter One: INTRODUCTION**

Starting the project, let us discuss firstly about the project introduction in brief. This chapter gives an overview about the aim, objectives, background and limitation of the system.

### **1.1. BACKGROUND OF PROJECT**

Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian(admin) to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students, etc. Books and student records are also included in this system which would keep track of the students using the library and also a detailed description about the books contained in the library. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used. Along with it, it greatly reduces the cost.

In addition, report module is also included in Library Management System. If user's position is admin, the user is able to generate different kinds of reports like lists of students registered, list of books, issue and return reports. All these modules are able to help librarian to manage the library with conveniently and in a more efficient way as compared to library systems which are not computerized.

## **1.2. OBJECTIVES**

The project aims and objectives that will be achieved after completion of this project are discussed in this sub-chapter. The aims and objectives are as follows:

1. A separate column for digital library.
2. Students' page where student can find books issued by him/her.
3. Students' don't need to remember the issued/return date and fine/
4. A search column to search availability of books.
5. Admin can easily check details of book and students.
6. Reduces risk of data loss and damage and also cost friendly.
7. Easy to use and requires no advanced hardware/software.

## **1.3. LIMITATIONS**

Below are the limitations of the project described point-wise:

1. It is not completely user-friendly as UX aspect is yet to be optimized.
2. Bar-code system isn't used, every detail has to be manually typed.
3. If ID/password is forgotten, one has to contact college administration.
4. If issue detail is entered wrong, the student might get in trouble while returning books.

## Chapter Two: SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of the Library management System along with the software requirement specification for it and also the comparison between this project system and existing system. The complete detailed description of the system requirement is given before the developing process is carried out. Also, the advantages of the new system over the existing system are also explained.

### 2.1. Requirement Specification

#### 2.1.1 GENERAL DESCRIPTION

**Project Description:** Library Management System is a computerized system which helps student and admin (librarian) to manage the library activity in electronic format. It is really useful for both of the users in many ways. Firstly, it reduces the risk of paper work such as file loss, file damage and time consumption. Also, It can help librarian to manage the transaction or record more effectively and properly. For students, it is easier to check the issued books timely and in detailed manner without worrying about the late-fees.

**Problem Statement:** The problems occurred before having computerized system includes:

1. File loss:
  - ➔ When computerized system wasn't implemented, there was always a risk of losing file. Sometimes due to simple human error, there may be a loss or wrong entry of records.
2. File damage:
  - ➔ When a computerized system was not introduced, file damage was always a serious problem. Due to some accidents like spilling of water by someone on file or also due to rats, records weren't safe.

3. Difficult to search record:
  - ➔ Due to the availability of large number of books and data, it was always hard to search for a specific book or record.
4. Space consuming:
  - ➔ A large space for physical storage of file and records was needed to keep all the data.
5. Cost consuming:
  - ➔ As papers aren't needed to store records physically, it drastically reduced the cost and expenses of libraries.

#### **2.1.2. SYSTEM OBJECTIVES**

1. Improvement in control and performance:
  - ➔ The system is developed to overcome the current issues and problems of library.  
The system can add user, validate user and is also bug free.
2. Reduce cost:
  - ➔ After computerized system is implemented, less human force will be required to maintain the library, thus reducing the overall cost.
3. Save time:
  - ➔ Librarian is able to search and manage record by using few clicks of mouse and few search keywords, thus saving his/her valuable time.
4. No risk of file loss/damage:
  - ➔ Every record is stored in computer in soft copy so the chance of file corruption or loss is less. Even if the system gets damaged, it can be easily retrieved.

## **2.1.3 SYSTEM REQUIREMENTS**

### **2.1.3.1 Product Requirements:**

#### **1. Efficiency Requirement:**

- ➔ When a library management system will be implemented, both librarian and student will have easy access to search books and book transaction will be very faster.

#### **2. Reliability Requirement:**

- ➔ The system should accurately perform student registration, student validation, book transaction and search.

#### **3. Usability Requirement:**

- ➔ The system is designed for a user-friendly environment so that students and librarians of library can perform various tasks easily and in an effective way.

### **2.1.3.2 Functional Requirements:**

#### **1. User Login:**

##### **❖ Description of feature:**

- ➔ This feature used by both student and admin to login into the system. They are required to enter their unique user id and password to enter into the system. The user id and password will be verified and if id or password is invalid, the user is not allowed to enter the system.

##### **❖ Functional requirements:**

- ⇒ Unique user id and password should be provided to users when registered.
- ⇒ The system should only allow user with valid id and password to enter into the system.
- ⇒ The system should perform authorization process which decides what user level can access to.
- ⇒ The user should be able to logout after they are done using the system.

## **2. Register New User:**

- ❖ Description of feature:

- ➔ This feature can be performed by all users to register new user to create account. They will be directed to their respective page after login.

- ❖ Functional requirements:

- ⇒ System must be able to verify information.
- ⇒ System must be able to add new information according to user.
- ⇒ System must be able to delete information if information is wrong.

## **3. Register New Book:**

- ❖ Description of feature:

- ➔ This feature allows to add new books to the system available in library.

- ❖ Functional requirements:

- ⇒ System must be able to verify information.
- ⇒ System must be able to modify data easily.
- ⇒ System must be able to identify if two books have same book id.

## **4. Issue and Return Books:**

- ❖ Description of feature:

- ➔ This feature allows to issue books and also view reports of book issued. The issue date will be used automatically according to pc's date.

- ❖ Functional requirements:

- ⇒ System must be able to enter issue information.
- ⇒ System must be able to update number of books.
- ⇒ System must be able to search if book is available or not before issuing book.
- ⇒ System should be able to enter issue and return date information.
- ⇒ System should be able to alert if same book is already issued.

## **5. Renew Books:**

- ❖ Description of feature:

- ➔ This feature is used to renew the date of issued books to avoid late-fees. If the books are renewed within deadline of return date, the user isn't fined of late-fees.

- ❖ Functional requirements:

- ⇒ System must be able to add date from the issued date of specific book.
- ⇒ System must be able to increase fine amount if books aren't renewed in time.

## **6. Search User/Book:**

- ❖ Description of feature:

- ➔ This feature is used to search users/books available in the system. We can search users/books using their unique id.

- ❖ Functional requirements:

- ⇒ System must be able to search the database based on select search type.
- ⇒ System must be able to filter book based on keyword entered.
- ⇒ System must be able to show the filtered book in table view.
- ⇒ System must be able to edit/delete data of users and books easily.

### **2.1.4 Software and Hardware requirements:**

- ➔ This section describes the software and hardware requirements of the system.

#### **1. Software Requirements:**

- ❖ Operating system:

- ➔ Windows 10 is used as the operating system as it is stable and supports more features and is more user friendly.

- ❖ Development tools and Programming language:
  - ➔ QT is used as a tool to write code and for front-end design. C++ is used for writing whole code. Since QT was used it can be compiled in cross platform. However, from few tests it was observed that the software works best in Linux.

## **2. Hardware Requirements:**

- ➔ Intel core duo can be used as a processor as it is faster and reliable, and we can run our pc for longtime. By using this processor, we can keep on developing our project without any worries.
- ➔ 1 gb ram can be used as it will provide enough reading and writing capabilities.

## **2.2. Feasibility**

Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. If the positive impact is greater than negative, it is considered as feasible. The feasibility study can be performed as:

### **2.2.1. Technical Feasibility:**

We can definitely say that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

### **2.2.2. Economic Feasibility:**

Development of this application is highly economically feasible. The only thing is to be done is making an environment for the development with an effective supervision. If we do so, we can attain the maximum usability of the corresponding resources. Even after the development, the system will be in a condition where much resources won't be needed.



## Chapter Three: SYSTEM DESIGN

In this chapter, we will discuss about the designs and steps involved in the project. Also, we describe about the way we used in the project to store the data and record in the system. The detailed description of the datatypes used in code and the flowcharts diagram is given before carrying out the development process.

### 3.5. TABLE DESIGN

Below are the tables used to manage the information in the system:

#### 3.5.1. BOOK TABLE TO KEEP BOOK INFORMATION:

Field	Data type	Default
ID	QString	N/A
Name	QString	N/A
Author	QString	N/A
Publication year	QString	N/A

#### 3.5.2. STUDENT TABLE TO KEEP STUDENT INFORMATION:

Field	Data type	Default
ID	QString	N/A
Year	QString	N/A
Part	QString	N/A
Course	QString	N/A
Access Level	QString	Student

3.5.3. ADMIN TABLE TO KEEP ADMIN INFORMATION:

Field	Data type	Default
Access Level	QString	Admin

3.5.4. ISSUE BOOKS TABLE FOR BOOK ISSUE RECORD:

Field	Data type	Default
Issued date	QString	Null
Issued by	QString	Null

3.5.5. STUDENT LOGIN TABLE:

Field	Data type	Default
ID	QString	N/A
Password	QString	N/A
Username	QString	N/A

### **3.6. Algorithms:**

#### **3.2.1. Algorithm for Login:**

Step 1 – Start.

Step 2 – Login window opens. Enter username: name, Enter password: pass

Step 3 - If username exist in table: If password is valid: //login successful

Another window opens To enter new node

Step 4 – Admin/Student page opens if login is successful. If not then GOTO step 2

Step 5 - If task complete then GOTO logout

Step 6 - End

#### **3.2.2. Algorithm for Check-In:**

Step1 - Start

Step2 - If login== successful // admin page opens //

Step3 - GOTO check-ins

Step4 - Enter inside ‘student’ or ‘book’ for details

Step5 - If student: enter id== valid; search== issue book ,

Step6 - If id==not valid;search== goto step5

Step7 - Book limit==check: availability== successfully issued

Step8 - If book limit==check: not available == cannot be issued , GOTO step 3.

Step9 – End

### **3.2.3. Algorithm of Renew:**

Step1 - Start

Step2 - Login to admin page.

Step3 - GOTO renewal page

Step4 - Student ID: //searched // ==valid; details of student and book is displayed.

If student ID or admin ID :// searched == Not valid; redirected to step 3

Step5 - Click on 'Renew' button ==renewed; If and only If the book has been issued by the same student

step6 - If not: 'ERROR' is displayed.

step7 - End

### **3.2.4. Algorithm for Check-Out:**

Step1 - Start

Step2 - Login to admin page

Step3 - Click to 'check-out' button

Step4 - Enter student ID: //searched // ==valid; details of student and book is displayed.

If student ID or admin ID :// searched == Not valid; redirected to step 3

Step5 - Click 'check-out' button

Step6 - If student== issued book : remove book or

If student==Not issued book : Display; 'ERROR'

Step7 - Display 'Removed successfully'

Step8 - End

### **3.2.5. Algorithm for Add page:**

Step1 - Start

Step2 - Login to admin page

Step3 - Click 'ADD' button

Step4 - If 'Add book' :Enter details of Book== valid and 'Add student' : Enter details of student== valid ; Display ' Registered successfully'

Step5 - If details==Not valid: GOTO step4

Step6 - If details==valid : Redirect to admin page to add pages

Step7 - End

### **3.2.6. Algorithm for Edit/Delete:**

Step1 - Start

Step2 - Login to admin page

Step3 - Click 'Edit or Delete' button

Step4 - Enter student ID: //searched // ==valid; details of student and book is displayed.

If student ID or admin ID :// searched == Not valid; redirected to step 3

Step5 - If entered details== valid : click 'EDIT DATA' or If entered details== Not valid : redirected to enter the details again

Step6 - If edited data== valid: redirected to student ID/ Book page

Step7 - If student Id and Book: 'Delete' // If Yes: redirected to step3 or If No: redirected to step4

Step8 - End

### 3.1.7. Algorithm for student page:

Step1- Start

Step2 - Login to student page

Step3 - Click 'Display' button :Displays student's details

Step4 - Click 'Issue Details' button :Displays issued book details

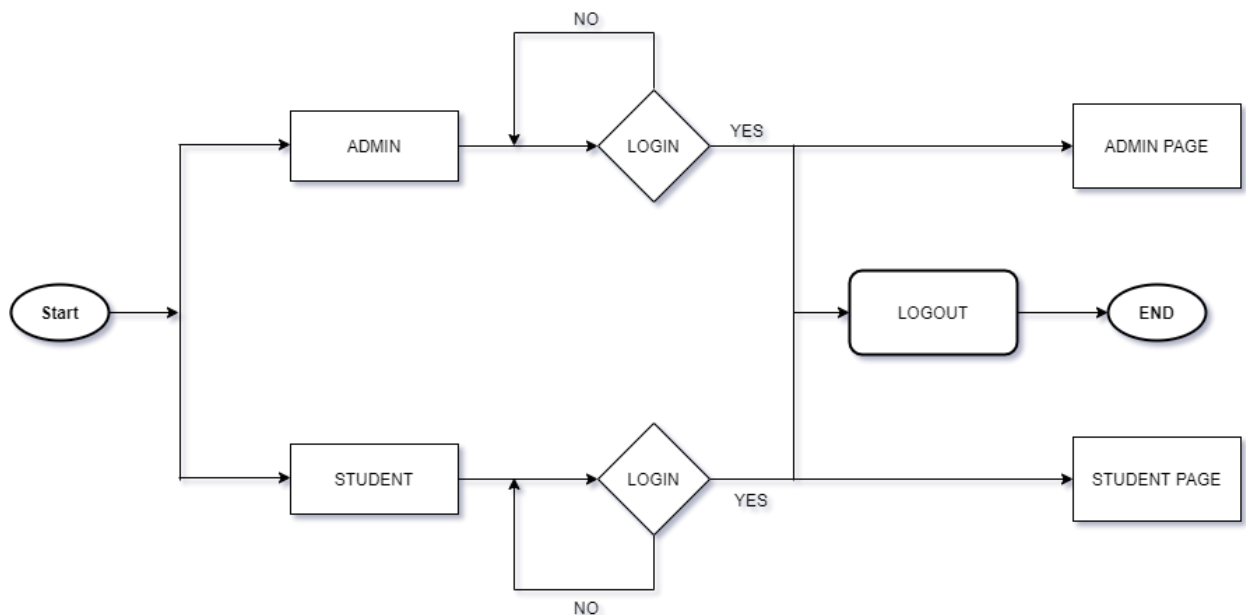
Step5 - Click 'Search Books' : Enter Book name; //search// If Book searched==valid:

Displays details of book or If Book searched==Invalid: redirected to step4

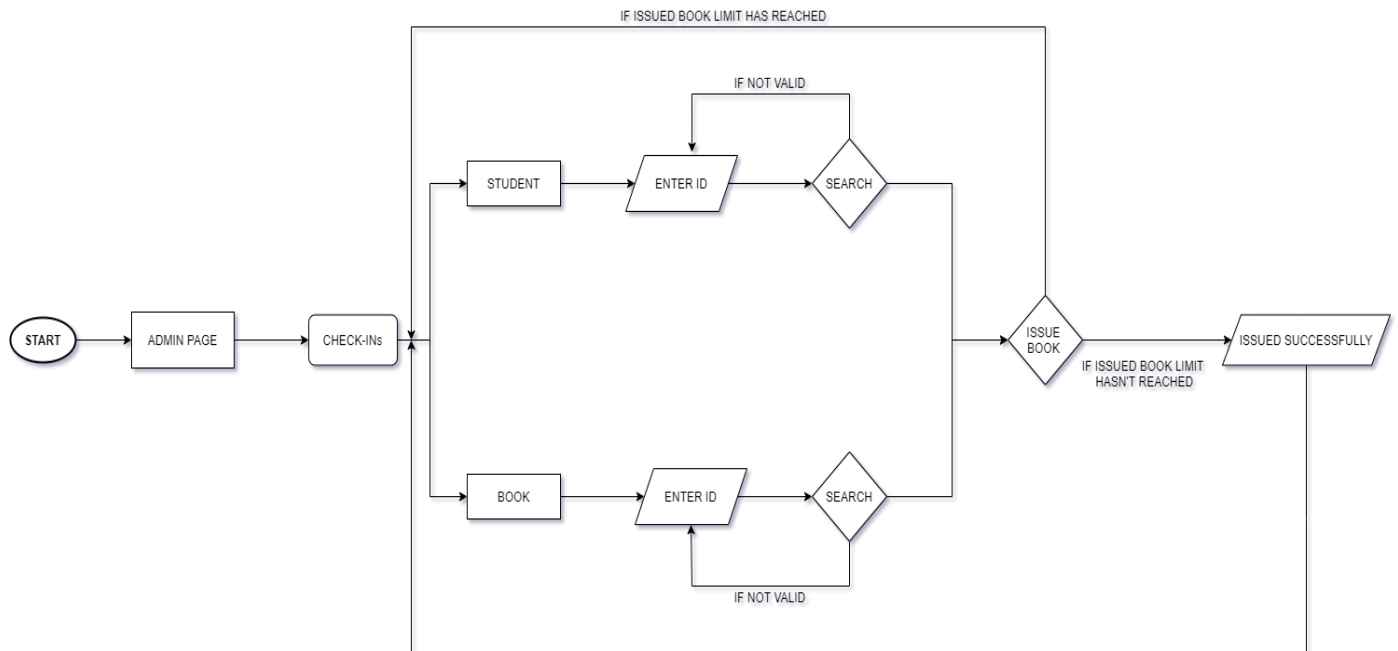
Step6 - End

## 3.7. Flowchart:

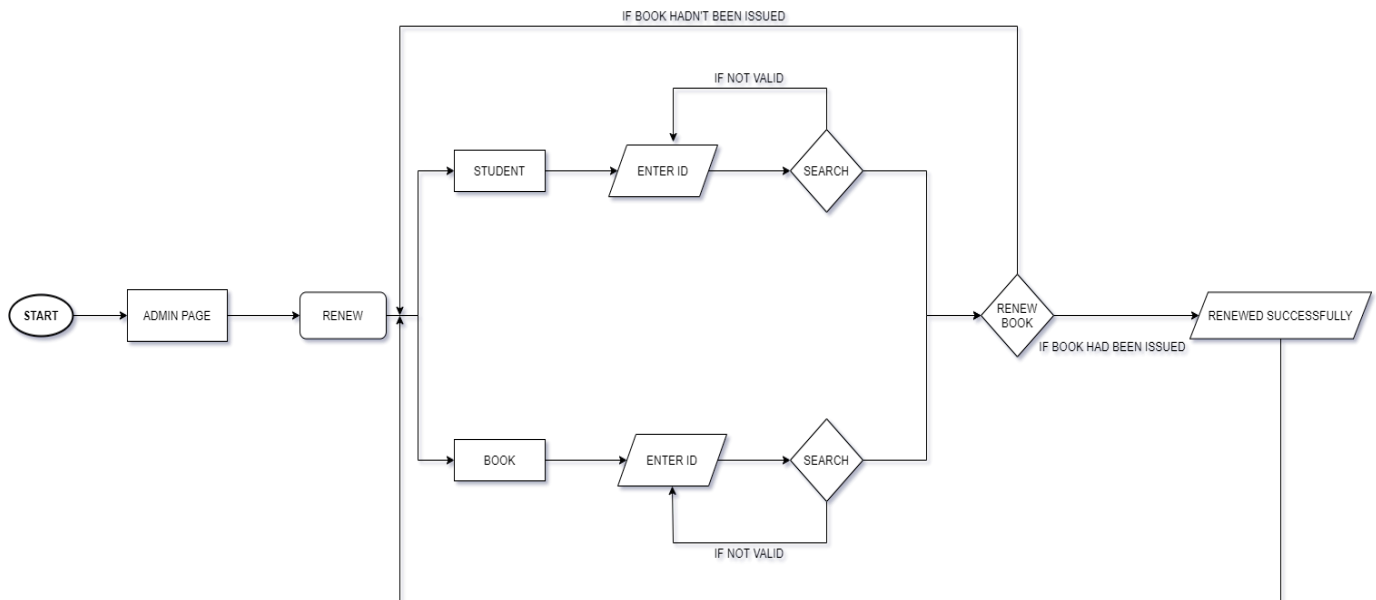
### 3.7.1. Flowchart for Login:



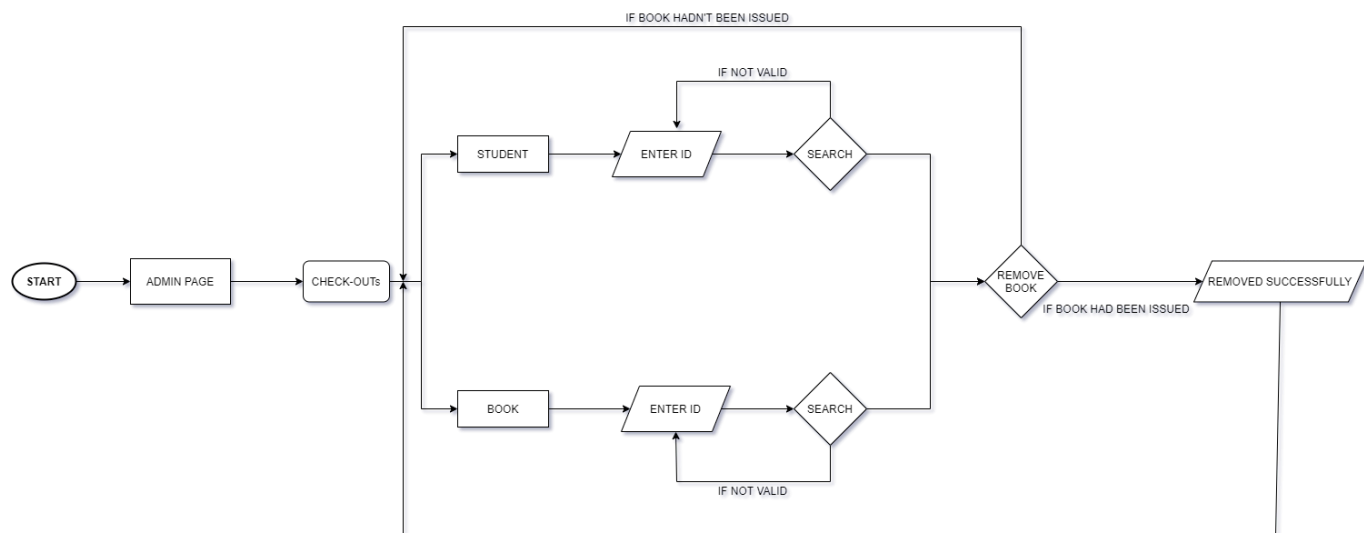
### 3.7.2. Flowchart for Admin Check-Ins:



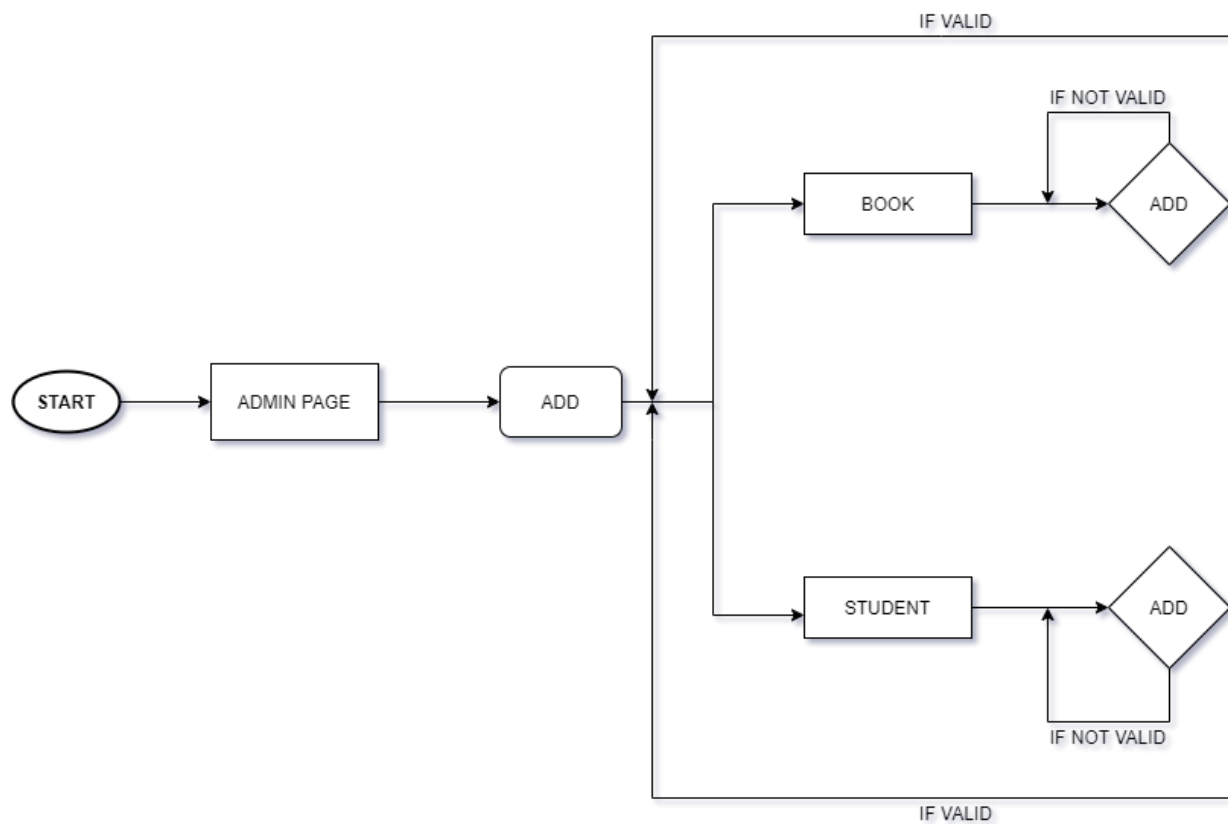
### 3.7.3. Flowchart for Renew:



### 3.7.4. Flowchart for Check-Out:

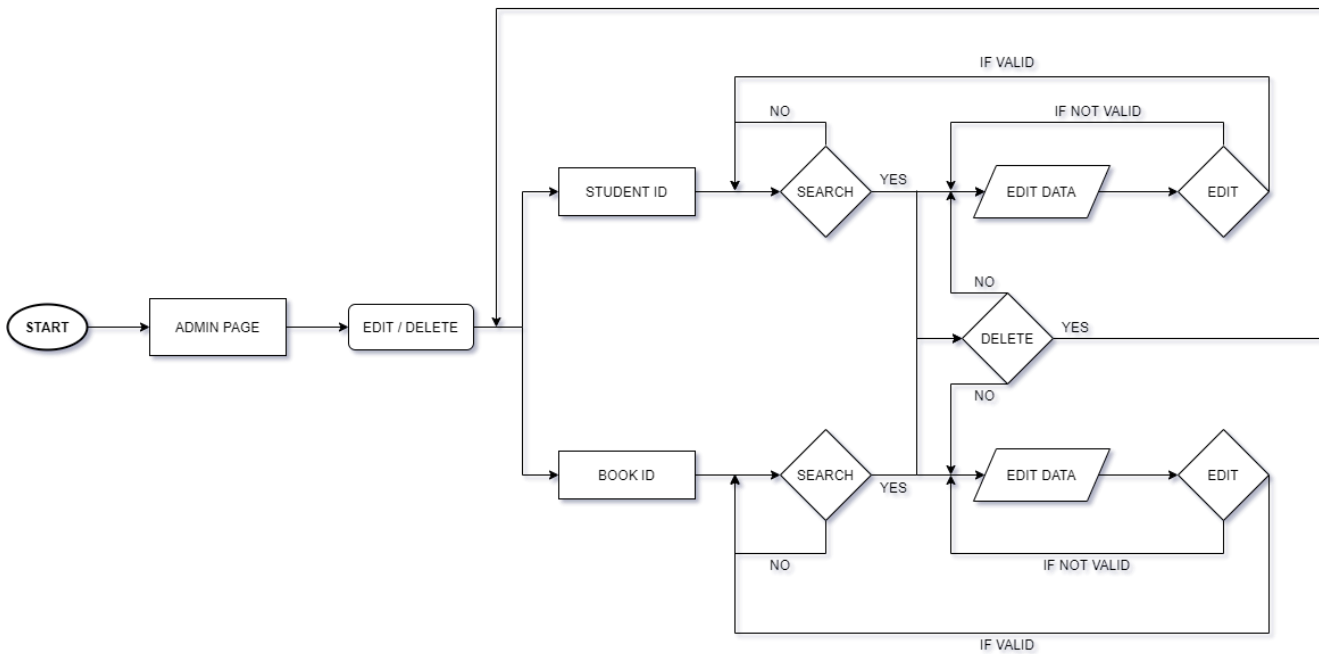


### 3.7.5. Flowchart for Add page:

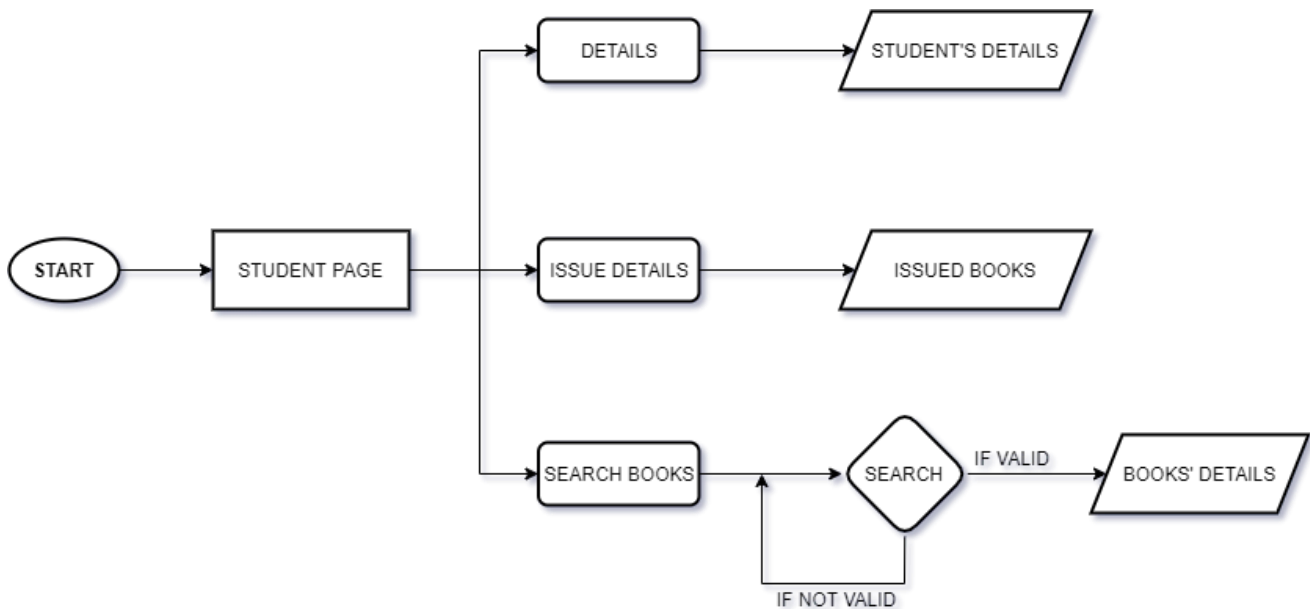




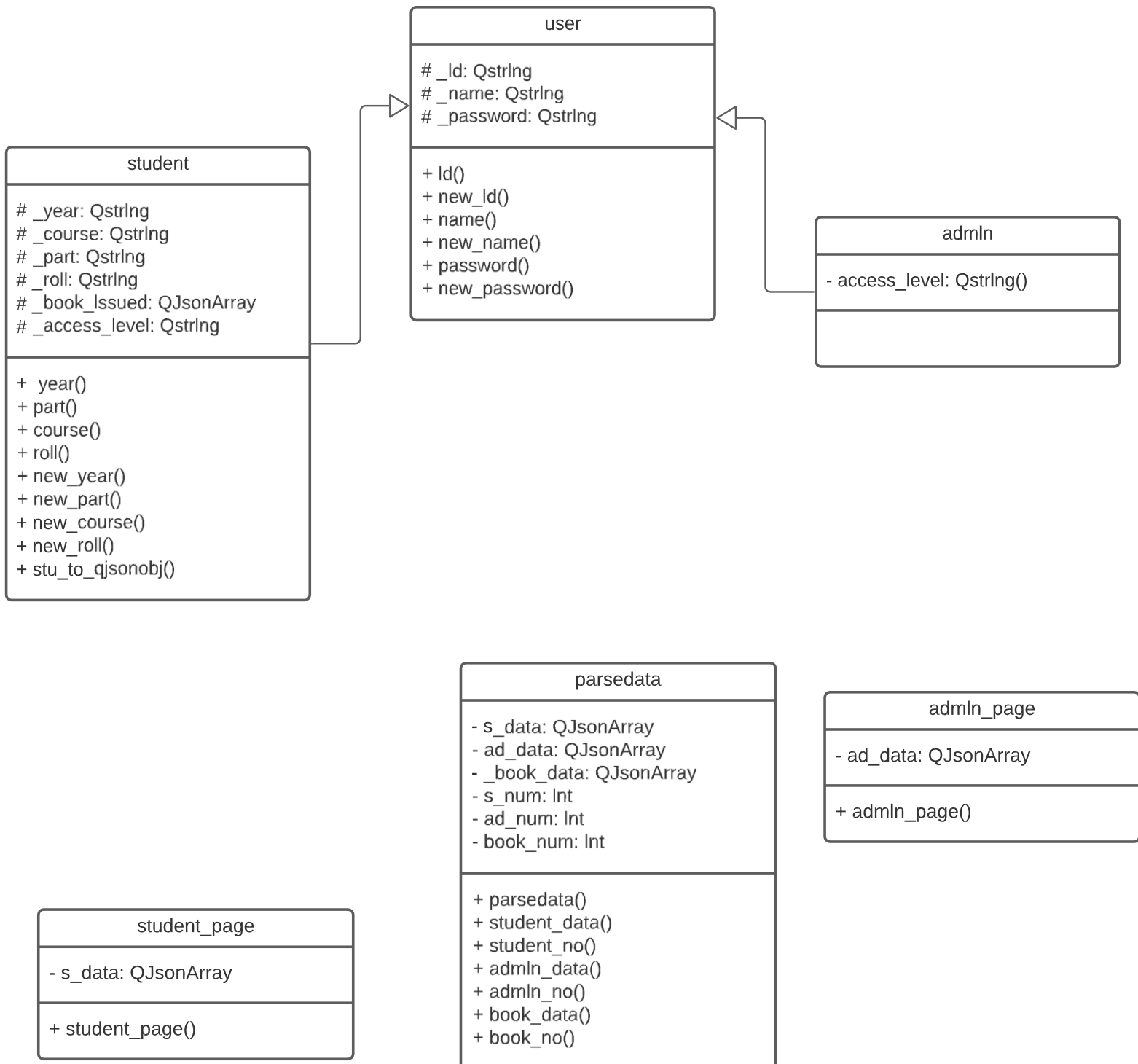
### 3.7.6. Flowchart for Edit/Delete:



### 3.7.7. Flowchart for Student page:



### 3.8. UML Diagram:



book
#_id: QString #_name: QString #_author: QString #_pub_year: QString #_issued_by: QString #_issued_date: QString
+ id() + name() + author() + pub_year() + issued_by() + issued_date() + new_id() + new_name() + new_author() + new_pub_year() + new_issued_by() + new_issued_date() + book_to_qjsonobj()

date
- _today: QString - _price: double - _deadline_day: int
+ date() + today() + price() + deadline_day() + add_deadline() + today_Qdate()

## Chapter Four: SYSTEM TESTING

The aim of the system testing process was to determine all defects in our project. The program was subjected to a set of test inputs and various observations were made and based on those observations, it will be decided whether the program behaves as expected or not. Our Project went through levels of testing.

Unit testing is undertaken when a module has been created and successfully reviewed. In order to test a single module, we need to provide a complete environment. Unit testing was done on each and every module of our system.

### **4.1. Test For the admin module**

**4.1.1. Testing admin login form** - This form is used for login of the system. In this we enter the ID and password of admin if both are correct, admin page will open. If any of data is wrong it will get redirected back to the login page and again ask for ID and password.

**4.1.2. Book Issue form** – Here, books are issued for specific students using their id. The first page of the admin page is the book issue page. To issue a book, admin has to search student's id first and if the id is valid and book limit hasn't reached, the student can issue the book. The clear button clears the input and output data on the page.

**4.1.3. Renew** – The books issued by students are renewed here. The book id is searched then deadline for the book is increased. If the books weren't issued or id used is not valid, then an error alert pops out then redirects it to the page again. Clear data button clears the input and output data on the page.

- 4.1.4. Book Return form** – The books to be returned is searched using ID. The student's details are shown along with the book. If the id isn't valid, error alert will pop up and redirect it to the page. Clear data button clears the input and output data on the page.
- 4.1.5. Student / Book account addition** - In this section, the admin can verify student details from student academic info and then only add student details to main library database. Also, for book addition, admin has to enter details of the book available in the library. It contains add button, if user clicks add button, data will be added.
- 4.1.6. Edit / Delete** – In this form, we can search information of student and book. Firstly, we enter id of either book or student which we want to search. By clicking on the search button, we get directed to next page where the details are shown where we can edit and delete the information as per our wish.

## **4.2. Test for Student module**

- 4.2.1. Test for Student login Form** - This form is used for login of Student. In this we enter student id and password, if all these are correct student page will open. If any of data is wrong it will get redirected back to the login page and again ask for id and password.
- 4.2.2. Details** – After the successful login, the landing page is Student's Details page where the details of students kept are shown properly. Here, the student's name, ID, the course he/she has been pursuing along with the year and part is shown.
- 4.2.3. Issue Details** – In this page, the overall issued book of the student is shown in a tabular form with book id, book name, issued date, return date and fine. Also, the total fine is calculated below which is to be paid.

**4.2.4. Search Book** – We can search book available in the library using the book name. The best matched search input is shown below the button in a tabular form with its id, name, author and if the book is available or not. The books can be updated by admin if new books are added or the books are available.

## Chapter Five: IMPLEMENTATION AND CODING

### 5.1. Overview:

For a better understanding of how the program works, understanding what files there are and how they are managed would be a good start.

Firstly, we have our main directory named LibraryMS with sub-directories img and JSON.

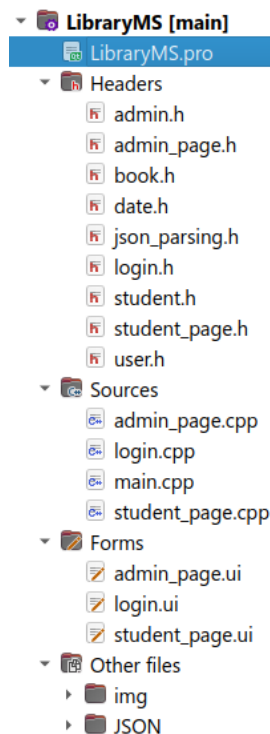
The classification made in the adjacent figure is done by the Qt creator and is not a directory in itself.

So in the header files, we have the class declaration, where all the classes we use are declared, which are all discussed in chapter 4.

Next, we have source code. The .cpp files contain all the class definitions which were declared in the header files. These files are directly linked with corresponding .ui files

The .ui files are the XML files that were created using Qt Designer, a convenient way to design the front-end. It is to be noted that this project was mainly focused on the back-end so it lacks the aspects of responsiveness.

Then there are the directories img and JSON which contains the image files and the JSON file used in the project.



## 5.2. The database

Before understanding the code it is better to first understand how the data used in the program is stored.

For the data storing we have used JSON format. Now JSON is not a Database Management System(DMS) but a data format. However, here we have used JSON as a form of a database which may not be ideal for Library Management software.

There are three JSON files used here, namely admin\_data.json, book\_data.json, and student\_data.json.\*

\*The content of the files are listed in Appendix C

Here is an example of how the data is stored in the JSON files. The whole data is an array of objects. The objects then consist of data and the corresponding key to access it.

The JSON files are parsed in the header file json\_parsing.h\*\*

\*\* The content of json\_parsing.h is listed in Appendix A

```
1  [{
2      "id": "admin0",
3      "name": "admin0",
4      "password": "admin",
5      "access_level": "admin"
6  },
7  {
8      "id": "admin1",
9      "name": "admin1",
10     "password": "admin",
11     "access_level": "admin"
12 },
13 {
14     "id": "",
15     "name": "Admin void",
16     "password": "",
17     "access_level": "admin"
18 }
19 ]
20
```

However, for a better understanding of how the code for parsing JSON works here is a short snippet without error handling.

For the parsing, we will be using the following libraries:

```
#include <QFile>
```

```
#include <QByteArray>
```

```
#include <QJsonDocument>
```

```
#include <QJsonArray>
```

```
#include <QJsonObject>
```

Now for the code:

```
QFile ad_file("../LibraryMS/JSON/admin_data.json");    //links the file location
ad_file.open(QIODevice::ReadOnly | QIODevice::Text)    //opens the file as ReadOnly
QByteArray rawData = ad_file.readAll();                //Reads the whole file and stores it
//Initialize the whole file as JSON
QJsonDocument ad_doc(QJsonDocument::fromJson(rawData));

//Since the file data is one array this line initializes the data as an array
QJsonArray ad_json = ad_doc.array();

//if the whole file data is one big object then this line initializes it as an object:
QJsonArray ad_json = ad_doc.object();
```

Now if we were access data and suppose access the ID of the first data we can do so with:

```
QString id = d_data.at(0).toObject()["id"].toString();
```

Here, the .at() is for the index of the array,

.toObject() inializes the data as an object,

["id"] is to access the data with key named "id"

And .toString() initalizes the value in the "id" key as string.

And with a similar concept, we can now access any data in the JSON files.



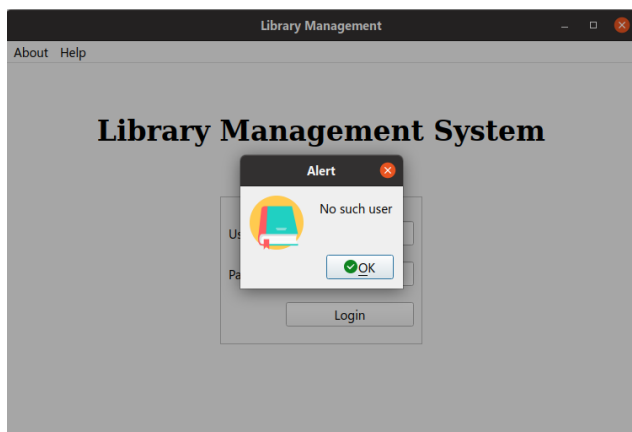
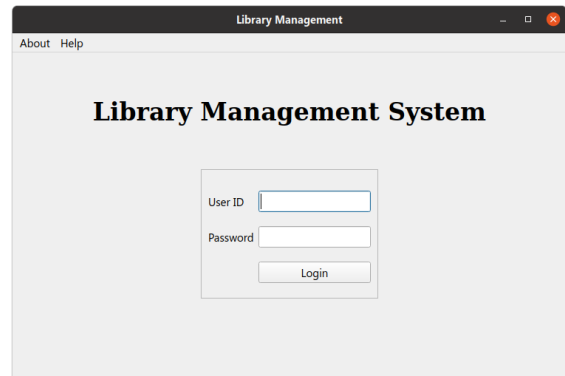
### 5.3. Coding of the project:

Since the software has different parts and files. We shall be discussing them on a page basis starting with the login page. Do note all the actual codes are listed in Appendix A and Appendix B

#### A. The login page

The login page takes two input User ID and Password, which is pretty much self-explanatory of what it does.

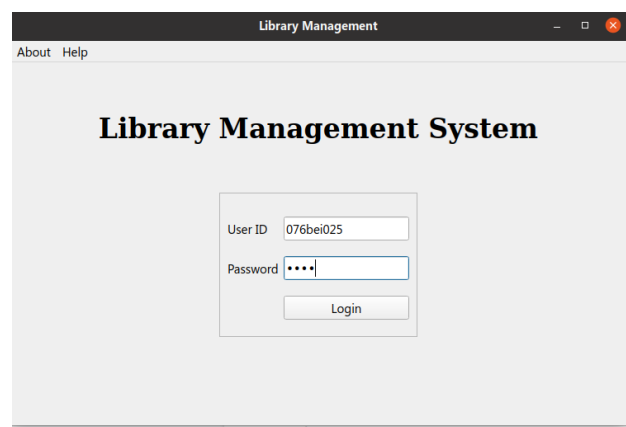
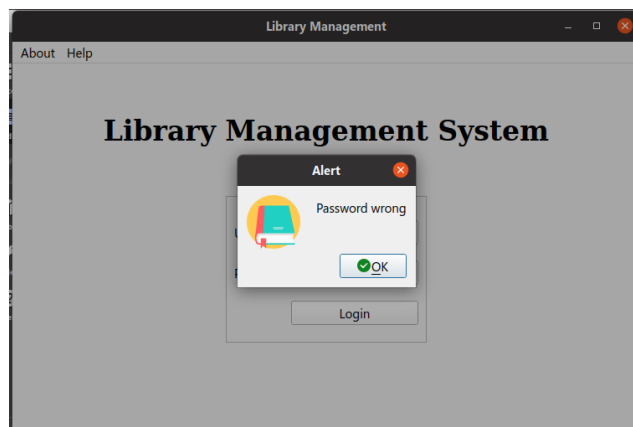
As for how it works is firstly it reads the data from the lineEdit object of the user-id then iterates through the whole admin\_data.json and if the data exists then it checks the password of that user with the lineEdit object of the password, and if it doesn't then it does the same process with student\_data.json.



Now there are a few failsafe for the errors which may occur:

For instance when the JSON file couldn't be opened or parsed or is empty. Similarly when the user id doesn't exist or if the password is wrong then it throws an error message.

And if the ID and password are correct it then opens the admin page or student as page depending on the user info.



## B. The student page

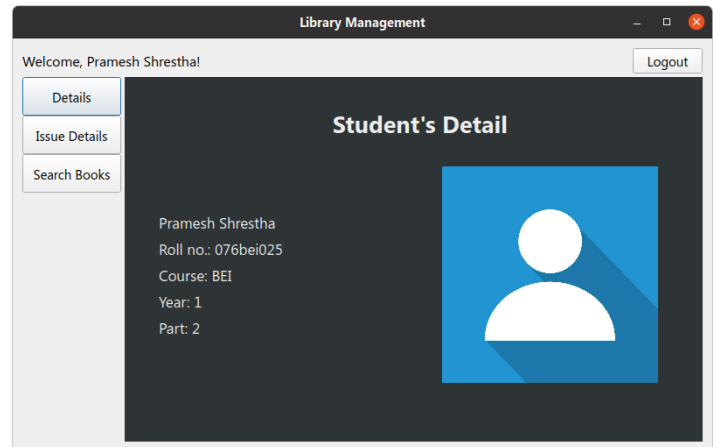
The student page consists of three different stack widgets namely:

### a. Detail page

The detail page consists of the student's info such as name, roll, course, and such.

Since these data must be displayed as soon as the student page opens, all these values are set on the constructor of the page\*

\*see appendix B

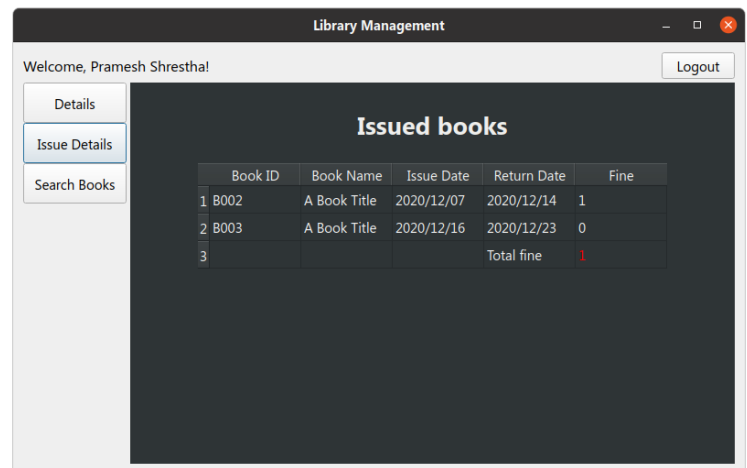


### b. Issue detail page

This page consists of all the book details the student has issued along with the fine from the due books.

For this to work we have used the QDate library to calculate the date which is included from date.h header file.

The header file, date.h also consists of a few const variables that determine the fine per day and the submission date

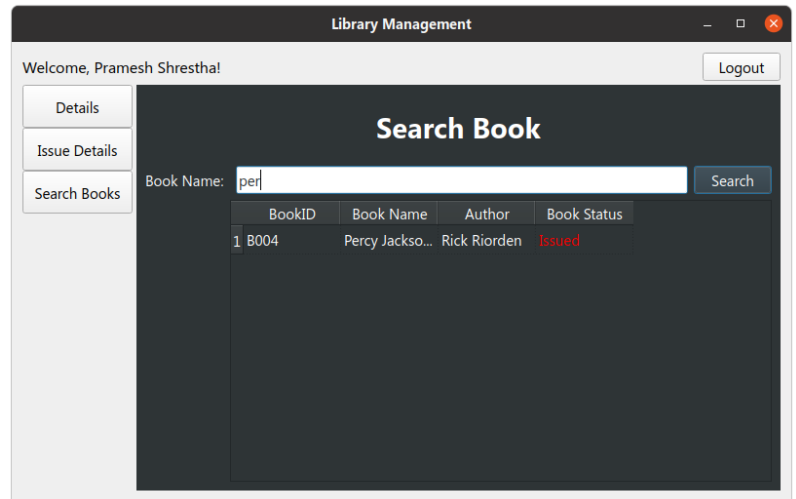


### c. Search book page

This page is meant for the students to search for the books and check if the book is available or not.

The searching algorithm for this is quite simple. First, the input from line edit is converted to lower case to prevent the unmatching of the name due to case sensitivity, Then the input keyword is converted to `std::String` to utilize the library and use one of its unique method which `QString` doesn't have i.e. `.find()`. `find()` method takes a string as an argument and searches if the string has those arguments or not.

With this concept, the whole `book_data.json` is iterated to find the books having those keywords inputted by the user and displays it.



## C. The admin page

The admin page is the core part of the software. It includes adding new student/book data, issuing book, renew it, checkout and so, which is discussed below

### a. Check-in page

This page is accessible to the admin only and is used to issue books to the student.

Firstly the admin searches the student's detail and then book detail with their respective id, which after confirmation issues the book to the student.

There are a few failsafe used here.

If the user id doesn't exist, it returns an error.

The screenshot shows a web application window titled 'Library Management'. On the left is a sidebar with buttons: 'Check-Ins', 'Renew', 'Check-Outs', 'Add', and 'Edit/Delete'. The main area is titled 'BOOK ISSUE' and contains two search sections. The first section is for the student, with fields for 'Student ID' (076bei028) and 'Name' (Rahul Shrestha), and a 'Search' button. The second section is for the book, with fields for 'Book ID' (B009), 'Name' (The subtle art of not giving a fck), 'Author' (Mark Manson), and 'Issued Status' (Not Issued), with a 'Search' button. At the bottom are 'Issue Book' and 'Clear' buttons. A 'Logout' button is in the top right corner.

Likewise, if the user has already issued 7 books, the user can't issue any more books. Similarly, for the book, it throws an error if the book is already issued by someone or if it doesn't exist.

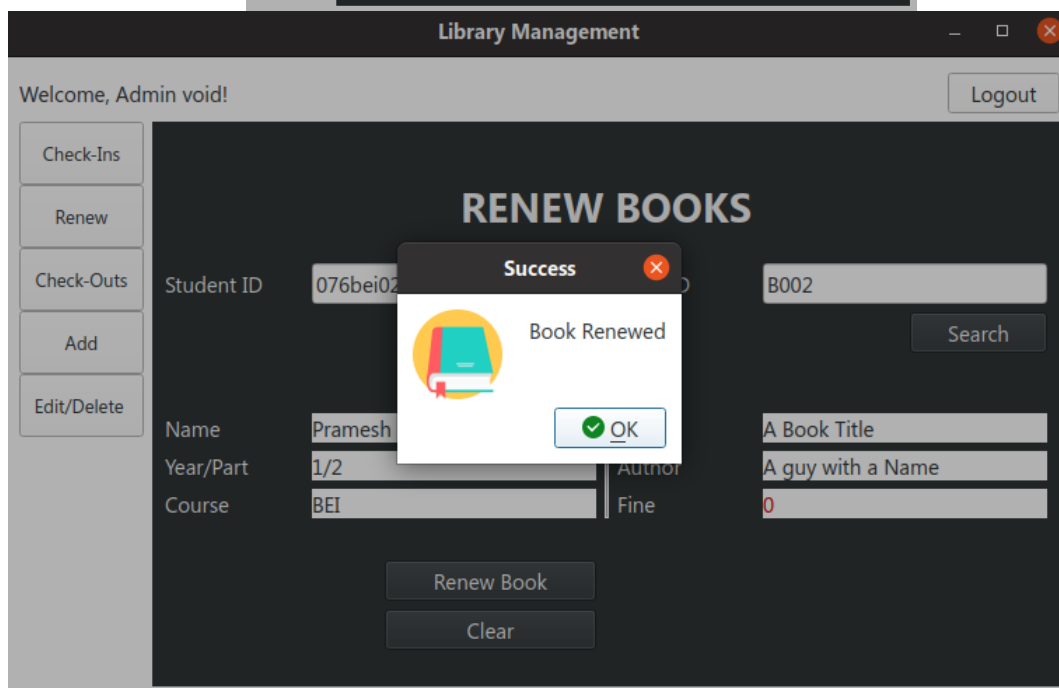
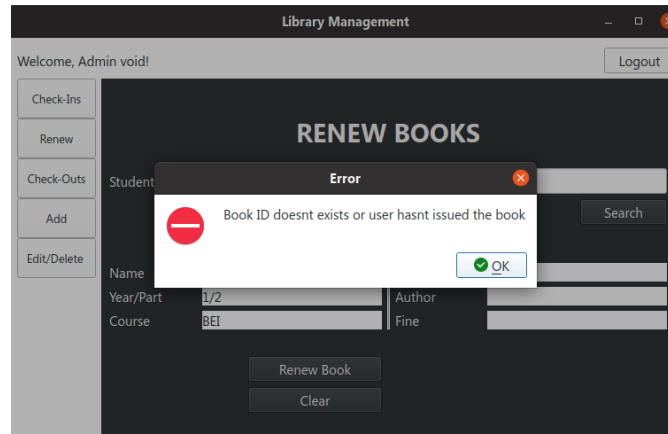
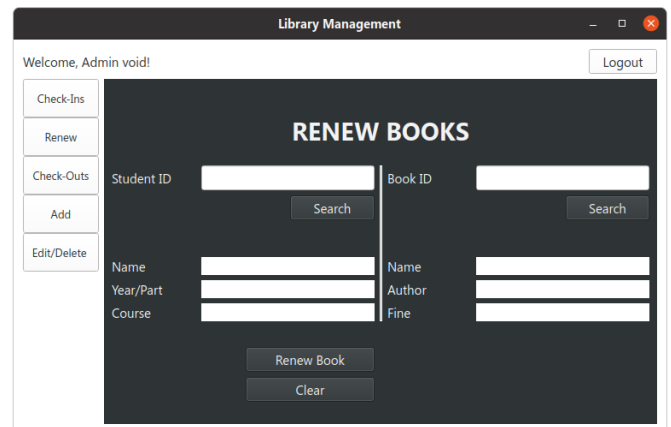
This screenshot is similar to the previous one, but it includes a 'Success' modal dialog box in the center. The dialog has a green checkmark icon and the text 'Issued successfully' with an 'OK' button. The background form is slightly dimmed.

## b. Renew book page

Similar to the check-in page the renew page firstly requires the admin to first enter the student's id and book id.

There is an extra label to display the fine from over-due if any.

Also, there is one more additional failsafe, which is if the book isn't issued by the user then it throws an error



### c. Check-out page

Similarly, the check-out page has all the failsafe from the renew page with the only difference being that on confirmation it checks the book out.

The screenshot shows a web application window titled "Library Management". At the top, it says "Welcome, Admin void!" and has a "Logout" button. On the left is a sidebar with navigation links: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area is titled "Check-out Books" and contains two search sections. The first section has input fields for "Student ID" (containing "076bei025") and "Book ID" (containing "b002"), each with a "Search" button. Below these are two columns of form fields. The left column has "Name" (Pramesh Shrestha), "Year/Part" (1/2), and "Course" (BEI). The right column has "Name" (A Book Title), "Author" (A guy with a Name), and "Fine" (0). At the bottom of the form are two buttons: "Check out book" and "Clear".

#### d. Add new data page

Through Add new data page we can add a new student or book data by choosing the radio options.

Upon choosing the option it sends the user to another stack widget page as the add page further consists of links to two other stack widgets:

The screenshot shows a web application window titled "Library Management". The header bar is dark grey with the title and window controls. Below the header, a light grey sidebar contains a list of menu items: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area has a dark grey background. At the top left of this area, it says "Welcome, Admin void!" and a "Logout" button is at the top right. The main heading is "Add Books/User". Below this, there are two radio button options: "New Book" (which is selected) and "New student". At the bottom of these options is a grey "Add" button.

#### i. New Book data:

The screenshot shows a web application window titled "Library Management". The header bar is dark grey with the title and window controls. Below the header, a light grey sidebar contains a list of menu items: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area has a dark grey background. At the top left of this area, it says "Welcome, Admin void!" and a "Logout" button is at the top right. The main heading is "Add New Book". Below this, there are four input fields labeled "ID", "Name", "Author", and "Pub Year". At the bottom right of these fields is a grey "Add" button.

ii. **New Student data:**

The screenshot shows a web application window titled "Library Management". At the top left, it says "Welcome, Admin void!". At the top right, there is a "Logout" button. On the left side, there is a vertical menu with buttons: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area has a dark background with the title "Add new Student" in large white text. Below the title, there are six input fields labeled "ID", "Name", "Year", "Part", "Course", and "Password". At the bottom right of the form, there is an "Add" button.

e. **Edit/Delete data page**

Similar to the add new data page, this page is also linked to two other stack widget for student data and book data

The screenshot shows a web application window titled "Library Management". At the top left, it says "Welcome, Admin void!". At the top right, there is a "Logout" button. On the left side, there is a vertical menu with buttons: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area has a dark background with the title "Edit Data" in large white text. Below the title, there are two input fields labeled "Enter Student ID" and "Enter Book ID". To the right of each input field is a "Search" button.



### i. Edit/delete student data

Once the admin enters the student's id, if it exists, it displays all the data of it in the lineEdit which can be then edited.

The failsafe here is, if the new id entered already exists then it throws an error.

The screenshot shows a web application window titled "Library Management". At the top, it says "Welcome, Admin void!" and has a "Logout" button. On the left, there is a sidebar with buttons: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area is titled "Edit Student Data" and contains a form with the following fields: Name (Sandesh sitaula), ID (0768EI035), Course (bei), Year (1), Part (2), and Password (1234). At the bottom of the form are "Edit" and "Delete" buttons.

### ii. Edit/delete book data

The edit book data is likewise similar to the edit student data besides the fact that it is the books.

The screenshot shows a web application window titled "Library Management". At the top, it says "Welcome, Admin void!" and has a "Logout" button. On the left, there is a sidebar with buttons: "Check-Ins", "Renew", "Check-Outs", "Add", and "Edit/Delete". The main content area is titled "Edit Book Data" and contains a form with the following fields: ID (B009), Name (The subtle art of not giving a fck), Author (Mark Manson), and Publication year (2016). At the bottom of the form are "Edit" and "Delete" buttons.

## **Chapter SIX: RESULTS AND DISCUSSION**

While making the project, we gained a lot of knowledge and experience. From the brainstorming of the ideas to the documentation, each moment was worth an experience that made us understand and learn a lot of new things and understand the class materials in-depth.

As we did the project in Qt, there was a lot for us to learn. Especially of how to interact with GUI based programs as till now we had but been learning in CLI based.

In the making of the software, we learned how to think in ways more than one. To think of errors that may occur and take measures to prevent that from occurring. We also learned that bugs are the only sole reason for a programmer's frustration and suicidal tendency and we learned to face it head-on.

## **Chapter SEVEN: CONCLUSIONS**

On the completion of the project, we learned that there will always be room for improvements. As at the end of the project, we did build a working Library management system however since it is still in its testing phase there are many bugs that are yet to be squashed as well as many features to add.

Even still, we did learn a lot from this project. May it be teamwork and problem handling, there was always something to take from those experiences and knowledge. So all in all, it was a really fruitful project for us.

## REFERENCES

1. Qt documentation: <https://doc.qt.io/qt-5/>
2. JSON parsing: [http://erickveil.github.io/2016/04/06/How-To-Manipulate-JSON-With-C++-and-Qt.html?fbclid=IwAR1aNXSK0WLV87fostDQG1sliHkuv\\_sHxjTn0VFQO9JFOv\\_RokqoczveNDk](http://erickveil.github.io/2016/04/06/How-To-Manipulate-JSON-With-C++-and-Qt.html?fbclid=IwAR1aNXSK0WLV87fostDQG1sliHkuv_sHxjTn0VFQO9JFOv_RokqoczveNDk)

## APPENDIX A (Header files)

<https://github.com/pramesh025/LibraryMS>

### Json\_parsing.h

```
#ifndef JSON_PARSING_H
```

```
#define JSON_PARSING_H
```

```
#include <QFile>
```

```
#include <QByteArray>
```

```
#include <QJsonDocument>
```

```
#include <QJsonArray>
```

```
#include <QJsonObject>
```

```
#include <QMessageBox>
```

```
class parsedata{
```

```
    QJsonArray s_data;
```

```
    QJsonArray ad_data;
```

```
    QJsonArray _book_data;
```

```
    int s_num;
```

```
    int ad_num;
```

```
    int book_num;
```

```
public:
```

```
    parsedata(){
```

```
        //parsing student json data
```

```
        QFile s_file("../LibraryMS/JSON/student_data.json");
```

```
        if (!s_file.open(QIODevice::ReadOnly | QIODevice::Text)) {
```

```
            QMessageBox::critical(nullptr, "Error", "Error in parsing student data");
```

```
        }
```

```
        QByteArray rawData = s_file.readAll();
```

```

// Parse document
QJsonDocument doc(QJsonDocument::fromJson(rawData));
if(doc.isEmpty()){
    QMessageBox::critical(nullptr,"Error","Student data empty");
}
// Get JSON object
QJsonArray json = doc.array();
int num=json.count()-1;
s_data = json;
s_num = num;
s_file.close();
//parsing admin json data
QFile ad_file("../LibraryMS/JSON/admin_data.json");
if (!ad_file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    QMessageBox::critical(nullptr,"Error", "Error in parsing admin data");
}
rawData = ad_file.readAll();
// Parse document
QJsonDocument ad_doc(QJsonDocument::fromJson(rawData));
if(ad_doc.isEmpty()){
    QMessageBox::critical(nullptr,"Error","Admin data empty");
}
// Get JSON object
QJsonArray ad_json = ad_doc.array();
num=ad_json.count()-1;
ad_data = ad_json;
ad_num = num;
ad_file.close();

//parsing book json data

```

```

QFile book_file("../LibraryMS/JSON/book_data.json");

if (!book_file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    QMessageBox::critical(nullptr, "Error", "Error in parsing book data");
}

rawData = book_file.readAll();

// Parse document

QJsonDocument book_doc(QJsonDocument::fromJson(rawData));

if(book_doc.isEmpty()){
    QMessageBox::critical(nullptr, "Error", "book data empty");
}

// Get JSON object

QJsonArray book_json = book_doc.array();

num=book_json.count()-1;

_book_data = book_json;

book_num = num;

book_file.close();
}

QJsonArray student_data(){
    return s_data;
}

int student_no(){
    return s_num;
}

QJsonArray admin_data(){
    return ad_data;
}

int admin_no(){
    return ad_num;
}

QJsonArray book_data(){

```

```

        return _book_data;
    }
    int book_no(){
        return book_num;
    }

};

#endif // JSON_PARSING_H

```

## date.h

```

#ifndef DATE_H
#define DATE_H

#include<QString>
#include<QDate>
#include<iostream>
#include<string>

class date{
    QString _today;

    double _price =0.5; //price for late turn in (per day)
    int _deadline_day=7; //deadline after ${deadline_day} from the issue day
public:
    date(){
        QDate temp = QDate::currentDate();
        _today=QString::number((temp.year()*100+temp.month()*100+temp.day()));
    }

    QString today(){

```



```
        return _today;
    }
    double price(){
        return _price;
    }
    int deadline_day(){
        return _deadline_day;
    }
    QDate add_deadline(QDate input){
        return input.addDays(_deadline_day);
    }
    QDate today_Qdate(){
        return QDate::currentDate();
    }

};

#endif // DATE_H
```

## user.h

```
#ifndef USER_H
#define USER_H

#include<QString>

class user{
protected:
    QString _id;
    QString _name;
    QString _password;
public:
    QString id(){
        return _id;
    }
    void new_id(QString new_id){
        _id=new_id;
    }
    QString name(){
        return _name;
    }
    void new_name(QString new_name){
        _name=new_name;
    }
    QString password(){
        return _password;
    }
    void new_password(QString new_password){
        _password=new_password;
    }
}
```

```
};
```

```
#endif // USER_H
```

## student.h

```
#ifndef STUDENT_H
#define STUDENT_H

#include<QString>
#include<QJsonArray>
#include<QJsonObject>

#include"user.h"

class student:public user{
protected:
    QString _year;
    QString _course;
    QString _part;
    QString _roll;
    QJsonArray _book_issued;
    QString _access_level = "student";
public:
    QString year(){
        return _year;
    }
    QString part(){
        return _part;
    }
    QString course(){
        return _course;
    }
    QString roll(){
        return _roll;
    }
}
```

```
}
```

```
void new_year(QString new_year){  
    _year=new_year;  
}
```

```
void new_part(QString new_part){  
    _part=new_part;  
}
```

```
void new_course(QString new_course){  
    _course=new_course;  
}
```

```
void new_roll(QString new_roll){  
    _roll=new_roll;  
}
```

```
JsonObject stu_to_qjsonobj(){  
    JsonObject Jtemp;  
    Jtemp.insert("id",_id);  
    Jtemp.insert("name",_name);  
    Jtemp.insert("password",_password);  
    Jtemp.insert("year",_year);  
    Jtemp.insert("part",_part);  
    Jtemp.insert("course",_course);  
    Jtemp.insert("roll",_roll);  
    Jtemp.insert("book_issued",_book_issued);  
    Jtemp.insert("access_level",_access_level);  
    return Jtemp;  
}
```

```
};
```

```
#endif // STUDENT_H
```

## **admin.h**

```
#ifndef ADMIN_H
#define ADMIN_H

#include<QString>
#include"user.h"

class admin:public user{
    QString access_level = "admin";
};

#endif // ADMIN_H
```

## **book.h**

```
#ifndef BOOK_H
#define BOOK_H

#include<QString>
#include<QJsonArray>
#include<QJsonObject>

class book{
protected:
    QString _id;
    QString _name;
    QString _author;
```

```

QString _pub_year;

QString _issued_by="NULL";

QString _issued_date="NULL";

public:

    QString id(){

        return _id;

    }

    QString name(){

        return _name;

    }

    QString author(){

        return _author;

    }

    QString pub_year(){

        return _pub_year;

    }

    QString issued_by(){

        return _issued_by;

    }

    QString issued_date(){

        return _issued_date;

    }


    void new_id(QString new_id){

        _id=new_id;

    }

    void new_name(QString new_name){

        _name=new_name;

    }

    void new_author(QString new_author){

```



```

        _author=new_author;
    }
    void new_pub_year(QString new_pub_year){
        _pub_year=new_pub_year;
    }
    void new_issued_by(QString new_issued_by){
        _issued_by=new_issued_by;
    }
    void new_issued_date(QString new_issued_date){
        _issued_date=new_issued_date;
    }

    QJsonObject book_to_qjsonobj(){
        QJsonObject Jtemp;
        Jtemp.insert("id",_id);
        Jtemp.insert("name",_name);
        Jtemp.insert("author",_author);
        Jtemp.insert("pub_year",_pub_year);
        Jtemp.insert("issued_by",_issued_by);
        Jtemp.insert("issued_date",_issued_date);
        return Jtemp;
    }

};

#endif // BOOK_H

```



## login\_page.h

```
#ifndef LOGIN_H
#define LOGIN_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class login; }
QT_END_NAMESPACE

class login : public QMainWindow
{
    Q_OBJECT

public:
    login(QWidget *parent = nullptr);
    ~login();

private slots:
    void on_pushButton_login_clicked();

private:
    Ui::login *ui;
};
#endif // LOGIN_H
```

## admin\_page.h

```
#ifndef ADMIN_PAGE_H
```

```

#define ADMIN_PAGE_H

#include <QWidget>
#include <QDialog>
#include <QJsonObject>

namespace Ui {
class admin_page;
}

class admin_page : public QWidget
{
    QJsonObject ad_data;
    Q_OBJECT

public:
    explicit admin_page(QJsonObject,QWidget *parent = nullptr);
    ~admin_page();

private slots:
    void on_pushButton_logout_clicked();

    void on_pushButton_checkin_clicked();

    void on_pushButton_renew_clicked();

    void on_pushButton_checkouts_clicked();

    void on_pushButton_add_clicked();

```

```
void on_pushButton_edit_clicked();

void on_pushButton_add_2_clicked();

void on_pushButton_addConfirm_clicked();

void on_pushButton_addnewbook_clicked();

void on_pushButton_edit_stuID_SEARCH_clicked();

void on_pushButton_edit_confirm_clicked();

void on_pushButton_delete_data_clicked();

void on_pushButton_edit_bookID_search_clicked();

void on_pushButton_edit_book_clicked();

void on_pushButton_delete_book_clicked();

void on_pushButton_issue_stID_search_clicked();

void on_pushButton_issue_bID_search_clicked();

void on_pushButton_issue_clicked();

void on_pushButton_clear_clicked();

void on_pushButton_issue_stID_search_2_clicked();
```

```
void on_pushButton_issue_bID_search_2_clicked();

void on_pushButton_Renew_clicked();

void on_pushButton_clear_2_clicked();

void on_pushButton_out_stID_search_3_clicked();

void on_pushButton_out_bID_search_3_clicked();

void on_pushButton_out_clicked();

void on_pushButton_clear_3_clicked();

private:
    Ui::admin_page *ui;
};

#endif // ADMIN_PAGE_H
```

## student\_page.h

```
#ifndef STUDENT_PAGE_H
```

```
#define STUDENT_PAGE_H
```

```
#include <QDialog>
```

```
#include <QJsonObject>
```

```
namespace Ui {
```

```
class student_page;
```

```
}
```

```
class student_page : public QDialog
```

```
{
```

```
    QJsonObject s_data;
```

```
    Q_OBJECT
```

```
public:
```

```
    explicit student_page(QJsonObject, QWidget *parent = nullptr);
```

```
    ~student_page();
```

```
private slots:
```

```
    void on_pushButton_clicked();
```

```
    void on_pushButton_2_clicked();
```

```
    void on_pushButton_3_clicked();
```

```
    void on_pushButton_4_clicked();
```

```
void on_pushButton_search_clicked();

private:
    Ui::student_page *ui;
};

#endif // STUDENT_PAGE_H
```



## APPENDIX B (Source files):

### main.cpp

```
#include "login.h"
// #include "student_page.h"
// #include "admin_page.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    a.setWindowIcon(QIcon("../LibraryMS/img/icon.png"));
    login w;
    // student_page w;
    w.show();
    return a.exec();
}
```

### login.cpp

```
#include <iostream>
#include <QDebug>

#include "login.h"
#include "json_parsing.h"
#include "ui_login.h"
```

```

#include "student_page.h"
#include "admin_page.h"

login::login(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::login)
{
    ui->setupUi(this);
}

login::~login()
{
    delete ui;
}

void login::on_pushButton_login_clicked()
{
    QString entered_username = ui->lineEdit_id->text().toLower();
    QString entered_password = ui->lineEdit_password->text();
    parsedata *check_json = new parsedata;
    QJsonArray stu_data = check_json->student_data();
    // qDebug()<<stu_data;
    QJsonArray ad_data = check_json->admin_data();
    int isdata=0;
    for(int i=0;i<=check_json->admin_no();i++){
        if(entered_username==ad_data.at(i).toObject()["id"].toString() &&
ad_data.at(i).toObject()["access_level"].toString()=="admin"){
            qDebug()<< ad_data.at(i).toObject()["id"].toString();
            isdata=1;

```

```

        if(entered_password==ad_data.at(i).toObject()["password"].toString()){
            hide();
            admin_page *ad_page = new admin_page(ad_data.at(i).toObject(),nullptr);
            ad_page->setWindowFlags(Qt::Window);
            ad_page->show();
//            qDebug()<<"Password correct!!"<<Qt::endl<<"Thou shall enter!!!!";
        }
        else{
            QMessageBox::about(this,"Alert","Password wrong");
        }
        break;
    }
}

for(int i=0;i<=check_json->student_no();i++){
    if(entered_username==stu_data.at(i).toObject()["id"].toString().toLowerCase() &&
    stu_data.at(i).toObject()["access_level"].toString()=="student"){
//        qDebug()<< stu_data.at(i).toObject()["id"].toString();
        isdata=1;
        if(entered_password==stu_data.at(i).toObject()["password"].toString()){
            hide();
            student_page *s_page = new student_page(stu_data.at(i).toObject(),nullptr);
            s_page->setWindowFlags(Qt::Window);
            s_page->show();
//            qDebug()<<"Password correct!!"<<Qt::endl<<"Thou shall enter!!!!";
        }
        else{
            QMessageBox::about(this,"Alert","Password wrong");
        }
        break;
    }
}

```

```
}  
if(!isdata){  
    QMessageBox::about(this,"Alert","No such user");  
}  
}
```

## student\_page.cpp

```
#include<QDebug>
#include<QDate>
#include<string>
#include<QPixmap>

#include "student_page.h"
#include "ui_student_page.h"
#include "login.h"
#include "json_parsing.h"
#include "date.h"

student_page::student_page(QJsonObject s_data, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::student_page)
{
    ui->setUpUi(this);
    student_page::s_data=s_data;
    ui->label_4->setText("Welcome, "+student_page::s_data["name"].toString()+"!");
    ui->label_detail_name->setText(student_page::s_data["name"].toString());
    ui->label_detail_course->setText("Course: "+student_page::s_data["course"].toString());
    ui->label_detail_roll->setText("Roll no.: "+student_page::s_data["roll"].toString());
    ui->label_detail_year->setText("Year: "+student_page::s_data["year"].toString());
    ui->label_detail_part->setText("Part: "+student_page::s_data["part"].toString());
    QPixmap pic("../LibraryMS/img/default_profile.jpg");
    int height=ui->label_profile->height();
    ui->label_profile->setPixmap(pic.scaled(height,height,Qt::KeepAspectRatio));
    ui->stackedWidget->setCurrentIndex(0);
}
```

```
student_page::~student_page()
```

```
{  
    delete ui;  
}
```

```
void student_page::on_pushButton_clicked()
```

```
{  
    ui->stackedWidget->setCurrentIndex(0);  
}
```

```
/*-----  
* -----  
* ----- Issue-detail_page -----  
* -----  
* -----*/
```

```
void student_page::on_pushButton_2_clicked()
```

```
{  
    ui->stackedWidget->setCurrentIndex(1);  
    parsedata *json = new parsedata;  
    QJsonArray book_json = json->book_data();  
    QJsonObject s_json= student_page::s_data;  
    int fine=0;  
    // qDebug()<<"so were did it crash?";  
    ui->tableWidget_issuedBooks->setRowCount(1);  
    for(int j=0;j<=s_json["book_issued"].toArray().count();j++){  
        for(int i=0;i<=json->book_no();i++){
```

```

if(s_json["book_issued"].toArray().at(j).toString().toLowerCase()==book_json.at(i).toObject()["id"].toString().toLowerCase()){

    QTableWidgetItem *temp_id = new QTableWidgetItem;
    temp_id->setText(book_json.at(i).toObject()["id"].toString());
    ui->tableWidget_issuedBooks->setItem(j, 0, temp_id);

    QTableWidgetItem *temp_name = new QTableWidgetItem;
    temp_name->setText(book_json.at(i).toObject()["name"].toString());
    ui->tableWidget_issuedBooks->setItem(j, 1, temp_name);

    date date;
    QDate Today=date.today_Qdate();
    QDate iss_Date=QDate::fromString(book_json.at(i).toObject()["issued_date"].toString(),
"yyyyMMdd");
    QDate return_date=date.add_deadline(iss_Date);
    int day= return_date.daysTo(Today);

    QTableWidgetItem *temp_issued = new QTableWidgetItem;
    temp_issued->setText(iss_Date.toString("yyyy/MM/dd"));
    ui->tableWidget_issuedBooks->setItem(j, 2, temp_issued);

    QTableWidgetItem *temp_return = new QTableWidgetItem;
    temp_return->setText(return_date.toString("yyyy/MM/dd"));
    ui->tableWidget_issuedBooks->setItem(j, 3, temp_return);

    QTableWidgetItem *temp_fine = new QTableWidgetItem;
    if(day>0){

```

```

        temp_fine->setText(QString::number(day*date.price()));
        fine+=day*date.price();
    }
    else{
        temp_fine->setText("0");
    }
    ui->tableWidget_issuedBooks->setItem(j,4,temp_fine);

//         qDebug()<<"so it had crashed here!!!!!!!!!!!!!!";
    ui->tableWidget_issuedBooks->insertRow(ui->tableWidget_issuedBooks->rowCount());
    break;
}
}
}
QTableWidgetItem *temp_fine_ = new QTableWidgetItem;
temp_fine_->setText("Total fine");
ui->tableWidget_issuedBooks->setItem(s_json["book_issued"].toArray().count(), 3, temp_fine_);

QTableWidgetItem *temp_fine_val = new QTableWidgetItem;
temp_fine_val->setText(QString::number(fine));
ui->tableWidget_issuedBooks->setItem(s_json["book_issued"].toArray().count(), 4, temp_fine_val);
ui->tableWidget_issuedBooks->item(s_json["book_issued"].toArray().count(), 4)-
>setTextColor(Qt::red);
}

```



```

/*-----
* ----
* ----          book_search_page          ----
* ----
* -----*/

void student_page::on_pushButton_3_clicked()
{
    ui->stackedWidget->setCurrentIndex(2);
}

void student_page::on_pushButton_search_clicked()
{
    std::string input = ui->lineEdit_search_words->text().toLower().toStdString();
    parsedata *json = new parsedata;
    QJsonArray book_json = json->book_data();

    ui->tableWidget_show_result->setRowCount(0);
    int row=0;
    for(int i=0;i<=json->book_no();i++){
        std::string temp_bname=book_json.at(i).toObject()["name"].toString().toLower().toStdString();
        //    qDebug()<<ui->lineEdit_search_words->text();
        //    qDebug()<<book_json.at(i).toObject()["name"].toString().toLower();
        if(temp_bname.find(input)!=std::string::npos){
            //    qDebug()<<temp_bname.find(input);
            ui->tableWidget_show_result->insertRow(ui->tableWidget_show_result->rowCount());
            QTableWidgetItem *temp_B_id = new QTableWidgetItem;
            temp_B_id->setText(book_json.at(i).toObject()["id"].toString());
            ui->tableWidget_show_result->setItem(row, 0, temp_B_id);
        }
    }
}

```

```

QTableWidgetItem *temp_B_name = new QTableWidgetItem;
temp_B_name->setText(book_json.at(i).toObject()["name"].toString());
ui->tableWidget_show_result->setItem(row, 1, temp_B_name);

QTableWidgetItem *temp_B_author = new QTableWidgetItem;
temp_B_author->setText(book_json.at(i).toObject()["author"].toString());
ui->tableWidget_show_result->setItem(row, 2, temp_B_author);

QTableWidgetItem *temp_B_status = new QTableWidgetItem;
if(book_json.at(i).toObject()["issued_by"].toString()=="NULL"){
    temp_B_status->setText("Not issued");
    ui->tableWidget_show_result->setItem(row, 3, temp_B_status);
}
else{
    temp_B_status->setText("Issued");
    ui->tableWidget_show_result->setItem(row, 3, temp_B_status);
    ui->tableWidget_show_result->item(row,3)->setTextColor(Qt::red);
}
row++;
}
}
ui->tableWidget_show_result->removeRow(ui->tableWidget_show_result->rowCount()+1);
}

```

```

/*-----
* -----
* ----- logout_button -----
* -----
* -----*/

```

```

void student_page::on_pushButton_4_clicked()
{
    hide();
    login *login_page = new login();
    login_page->show();
}

```

## admin\_page.cpp

```
#include <QDebug>
```

```
#include "admin_page.h"
```

```
#include "ui_admin_page.h"
```

```
#include "login.h"
```

```
#include "admin.h"
```

```
#include "student.h"
```

```
#include "book.h"
```

```
#include "json_parsing.h"
```

```
#include "date.h"
```

```
admin_page::admin_page(QJsonObject ad_data,QWidget *parent) :
```

```
    QWidget(parent),
```

```
    ui(new Ui::admin_page)
```

```
{
```

```
    ui->setupUi(this);
```

```
    admin_page::ad_data=ad_data;
```

```
    ui->label_welcome->setText("Welcome, "+admin_page::ad_data["name"].toString()+"!");
```

```
    ui->stackedWidget->setCurrentIndex(0);
```

```
}
```

```
admin_page::~admin_page()
```

```
{
```

```
    delete ui;
```

```
}
```

```
void admin_page::on_pushButton_logout_clicked()
```

```
{  
    hide();  
    login *login_page = new login();  
    login_page->show();  
}
```

```
/*-----  
* -----  
* ----- Checked in page -----  
* -----  
* -----*/
```

```
void admin_page::on_pushButton_checkin_clicked()
```

```
{  
    ui->stackedWidget->setCurrentIndex(0);  
    //clean that sht  
    ui->label_issue_BookName->setText("");  
    ui->label_issue_BookAuthor->setText("");  
    ui->label_issue_BookStatus->setText("");  
}
```

```
QString issue_stu_ID;
```

```
bool can_issue=false;
```

```
QString issue_book_ID;
```

```
void admin_page::on_pushButton_issue_stID_search_clicked()
```

```
{  
    parsedata *json = new parsedata;  
    QJsonArray stu_json = json->student_data();  
    QString input_st=ui->lineEdit_issue_stID_search->text();
```

```

//clean that sht part 2

ui->label_issue_stuName->setText("");
ui->label_issue_stuYear->setText("");
ui->label_issue_stuCourse->setText("");
ui->label_issue_stuIssued->setText("");

bool is_data=false;

for(int i=0;i<=json->student_no();i++){

    if(stu_json.at(i).toObject()["id"].toString().toLowerCase()==input_st.toLowerCase()){

        is_data=true;

        ui->label_issue_stuName->setText(stu_json.at(i).toObject()["name"].toString());

        ui->label_issue_stuYear->
>setText(stu_json.at(i).toObject()["year"].toString()+"/"+stu_json.at(i).toObject()["part"].toString());

        ui->label_issue_stuCourse->setText(stu_json.at(i).toObject()["course"].toString());

        ui->label_issue_stuIssued->
>setText(QString::number(stu_json.at(i).toObject()["book_issued"].toArray().count()));

        if(stu_json.at(i).toObject()["book_issued"].toArray().count()>=7)

            QMessageBox::critical(this,"Error","Book limit reached");

        else{

            issue_stu_ID=input_st;

            can_issue=true;

        }

        break;

    }

}

if(is_data==false)

    QMessageBox::critical(this,"Error","Student ID doesnt exists");

}

void admin_page::on_pushButton_issue_bID_search_clicked()

{

    parsedata *json = new parsedata;

```

```

QJsonArray book_json = json->book_data();
QString input_book=ui->lineEdit_issue_bID_search->text();
//clean that sht part 3
ui->label_issue_BookName->setText("");
ui->label_issue_BookAuthor->setText("");
ui->label_issue_BookStatus->setText("");
bool is_data=false;
for(int i=0;i<=json->book_no();i++){
    if(book_json.at(i).toObject()["id"].toString().toLowerCase()==input_book.toLowerCase()){
        is_data=true;
        ui->label_issue_BookName->setText(book_json.at(i).toObject()["name"].toString());
        ui->label_issue_BookAuthor->setText(book_json.at(i).toObject()["author"].toString());

        if(book_json.at(i).toObject()["issued_by"].toString()=="NULL"){
            ui->label_issue_BookStatus->setText("Not Issued");
            issue_book_ID=input_book;
            can_issue=true;
        }
        else{
            ui->label_issue_BookStatus->setText("Already Issued");
            QMessageBox::critical(this,"Error","Book ALready issued");
        }
        break;
    }
}
if(is_data==false)
    QMessageBox::critical(this,"Error","Book ID doesnt exists");
}

void admin_page::on_pushButton_issue_clicked()

```

```

{
    if(can_issue==false){
        QMessageBox::critical(this,"Error","Cannot issue");
    }
    else{
        date today;
        bool success = true;
        parsedata *json = new parsedata;
        QJsonArray book_json = json->book_data();
        QJsonArray stu_json = json->student_data();
        for(int i=0;i<=json->book_no();i++){
            if(issue_book_ID.toLower()==book_json.at(i).toObject()["id"].toString().toLowerCase()){
                QJsonObject temp = book_json.at(i).toObject();
                book_json.removeAt(i);
                temp.remove("issued_by");
                temp.remove("issued_date");
                temp.insert("issued_by",issue_stu_ID);
                temp.insert("issued_date",today.today());
                book_json.append(temp);
                QFile book_file("../LibraryMS/JSON/book_data.json");
                if (!book_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
                    QMessageBox::critical(nullptr,"Error", "Error in parsing book data");
                }
                QJsonDocument doc(book_json);
                if(!book_file.write(doc.toJson())){
                    QMessageBox::critical(nullptr,"Error", "Book Database couldnt be accessed");
                    success=false;
                }
                book_file.close();
            }
        }
        //      qDebug()<<book_json;
    }
}

```



```

        break;
    }
}

if(success){
    for(int i=0;i<=json->student_no();i++){
        if(issue_stu_ID.toLower()==stu_json.at(i).toObject()["id"].toString().toLowerCase()){
            QJsonObject tempOBJ =stu_json.at(i).toObject();
            stu_json.removeAt(i);
            QJsonArray tempAr = tempOBJ["book_issued"].toArray();
            tempAr.append(issue_book_ID);
            tempOBJ.remove("book_issued");
            tempOBJ.insert("book_issued",tempAr);
            stu_json.append(tempOBJ);
            QFile stu_file("../LibraryMS/JSON/student_data.json");
            if (!stu_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
                QMessageBox::critical(nullptr,"Error", "Error in parsing student data");
            }
            QJsonDocument doc(stu_json);
            if(!stu_file.write(doc.toJson())){
                QMessageBox::critical(nullptr,"Error", "Student Database couldnt be accessed");
                success = false;
            }
            else{
                QMessageBox::about(nullptr,"Success","Issued successfully");
                on_pushButton_clear_clicked();
            }
            stu_file.close();
            //      qDebug()<<stu_json;
            break;
        }
    }
}

```

```

    }
}
}

void admin_page::on_pushButton_clear_clicked()
{
    //Jesus Christ!!!! CLEAN IT ALL!!!
    issue_stu_ID = " ";
    can_issue=false;
    issue_book_ID = " ";
    ui->label_issue_BookName->setText("");
    ui->label_issue_BookAuthor->setText("");
    ui->label_issue_BookStatus->setText("");
    ui->label_issue_stuName->setText("");
    ui->label_issue_stuYear->setText("");
    ui->label_issue_stuCourse->setText("");
    ui->label_issue_stuIssued->setText("");
    foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
        le->clear();
    }
}

```

```

/*-----

```

```

* -----
* ----- Renew page -----
* -----
* -----*/

void admin_page::on_pushButton_renew_clicked()
{
    ui->stackedWidget->setCurrentIndex(1);
}

QJsonObject renew_stu_data;
bool can_renew = false;
void admin_page::on_pushButton_issue_stID_search_2_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray stu_json = json->student_data();
    QString input_st=ui->lineEdit_renew_stID_search_2->text();
    //clean that sht
    ui->label_renew_stuName_2->setText("");
    ui->label_renew_stuYear_2->setText("");
    ui->label_renew_stuCourse_2->setText("");
    bool is_data=false;
    for(int i=0;i<=json->student_no();i++){
        if(stu_json.at(i).toObject()["id"].toString().toLowerCase()==input_st.toLowerCase()){
            is_data=true;
            ui->label_renew_stuName_2->setText(stu_json.at(i).toObject()["name"].toString());
            ui->label_renew_stuYear_2->
>setText(stu_json.at(i).toObject()["year"].toString()+"/"+stu_json.at(i).toObject()["part"].toString());
            ui->label_renew_stuCourse_2->setText(stu_json.at(i).toObject()["course"].toString());
            renew_stu_data=stu_json.at(i).toObject();

```

```

        break;
    }
}

if(is_data==false)
    QMessageBox::critical(this,"Error","Student ID doesnt exists");
}

QString renew_book_ID;

void admin_page::on_pushButton_issue_bID_search_2_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray book_json = json->book_data();
    QString input_book=ui->lineEdit_renew_bID_search_2->text();
    //clean that sht part 3
    ui->label_renew_BookName_2->setText("");
    ui->label_renew_BookAuthor_2->setText("");
    ui->label_renew_BookFine->setText("");
    bool is_data=false;
    for(int i=0;i<=renew_stu_data["book_issued"].toArray().count();i++){
        if(renew_stu_data["book_issued"].toArray().at(i).toString().toLowerCase()==input_book.toLowerCase()){
            for(int j=0;j<=json->book_no();j++){
                if(input_book.toLowerCase()==book_json.at(j).toObject()["id"].toString().toLowerCase()){
                    is_data=true;
                    can_renew=true;
                    renew_book_ID = input_book;
                    date date;
                    ui->label_renew_BookName_2->setText(book_json.at(j).toObject()["name"].toString());
                    ui->label_renew_BookAuthor_2->setText(book_json.at(j).toObject()["author"].toString());
                    QDate iss_Date=QDate::fromString(book_json.at(j).toObject()["issued_date"].toString(),
"yyyyMMdd").addDays(date.deadline_day());

```

```

        QDate today(QDate::currentDate());
        int day= iss_Date.daysTo(today);
//         qDebug()<<iss_Date.toString("yyyy.MM.dd");
//         qDebug()<<today.toString("yyyy.MM.dd");
        if(day-date.deadline_day(>0)

            ui->label_renew_BookFine->setText(QString::number(day*date.price()));

        else

            ui->label_renew_BookFine->setText(QString::number(0));

            break;

        }

    }

    break;

}

}

if(is_data==false)

    QMessageBox::critical(this,"Error","Book ID doesnt exists or user hasnt issued the book");

}

void admin_page::on_pushButton_Renew_clicked()

{

    if(can_renew){

        parsedata *json = new parsedata;

        date today;

        QJsonArray book_json = json->book_data();

//         qDebug()<<"WE ARE HERE";

        for(int i=0;i<=json->book_no();i++){

            if(renew_book_ID.toLower()==book_json.at(i).toObject()["id"].toString().toLower()){

                QJsonObject temp = book_json.at(i).toObject();

                book_json.removeAt(i);

                temp.remove("issued_date");

```

```

temp.insert("issued_date",today.today());
book_json.append(temp);
QFile book_file("../LibraryMS/JSON/book_data.json");
if (!book_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
    QMessageBox::critical(nullptr,"Error", "Error in parsing book data");
}
QJsonDocument doc(book_json);
if(!book_file.write(doc.toJson())){
    QMessageBox::critical(nullptr,"Error", "Book Database couldnt be accessed");
}
else{
    QMessageBox::about(this,"Success","Book Renewed");
    on_pushButton_clear_2_clicked();
}
book_file.close();
//      qDebug()<<book_json;
break;
}
}
}
else{
    QMessageBox::critical(this,"Error","Renew error\nNo data");
}
}

void admin_page::on_pushButton_clear_2_clicked()
{
    renew_book_ID=" ";
    renew_stu_data = { };
    can_renew = false;

```

```

ui->label_renew_BookAuthor_2->setText("");
ui->label_renew_BookName_2->setText("");
ui->label_renew_BookFine->setText("");
ui->label_renew_stuCourse_2->setText("");
ui->label_renew_stuName_2->setText("");
ui->label_renew_stuYear_2->setText("");
ui->label_issue_stuIssued->setText("");
foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
    le->clear();
}
}

```

```

/*-----
* ----
* ----          Checked out page          ----
* ----
* ----
* -----*/

```

```

void admin_page::on_pushButton_checkouts_clicked()
{
    ui->stackedWidget->setCurrentIndex(2);
}

QJsonObject checkout_stu_data;

```

```

bool can_checkout = false;

void admin_page::on_pushButton_out_stID_search_3_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray stu_json = json->student_data();
    QString input_st=ui->lineEdit_out_stID_search_3->text();
    //clean that sht
    ui->label_out_stuName_3->setText("");
    ui->label_out_stuYear_3->setText("");
    ui->label_out_stuCourse_3->setText("");
    bool is_data=false;
    for(int i=0;i<=json->student_no();i++){
        if(stu_json.at(i).toObject()["id"].toString().toLowerCase()==input_st.toLowerCase()){
            is_data=true;
            ui->label_out_stuName_3->setText(stu_json.at(i).toObject()["name"].toString());
            ui->label_out_stuYear_3->
>setText(stu_json.at(i).toObject()["year"].toString()+"/"+stu_json.at(i).toObject()["part"].toString());
            ui->label_out_stuCourse_3->setText(stu_json.at(i).toObject()["course"].toString());
            checkout_stu_data=stu_json.at(i).toObject();
            break;
        }
    }
    if(is_data==false)
        QMessageBox::critical(this,"Error","Student ID doesnt exists");
}

QJsonObject checkout_book_data;

void admin_page::on_pushButton_out_bID_search_3_clicked()
{
    parsedata *json = new parsedata;

```



```

QJsonArray book_json = json->book_data();
QString input_book=ui->lineEdit_out_bID_search_3->text();
//clean that sht part 3
ui->label_out_BookName_3->setText("");
ui->label_out_BookAuthor_3->setText("");
ui->label_out_BookFine_2->setText("");
bool is_data=false;
for(int i=0;i<=checkout_stu_data["book_issued"].toArray().count();i++){
    if(checkout_stu_data["book_issued"].toArray().at(i).toString().toLowerCase()==input_book.toLowerCase()){
        for(int j=0;j<=json->book_no();j++){
            if(input_book.toLowerCase()==book_json.at(j).toObject()["id"].toString().toLowerCase()){
                is_data=true;
                can_checkout=true;
                checkout_book_data = book_json.at(j).toObject();
                date date;
                ui->label_out_BookName_3->setText(book_json.at(j).toObject()["name"].toString());
                ui->label_out_BookAuthor_3->setText(book_json.at(j).toObject()["author"].toString());
                QDate iss_Date=QDate::fromString(book_json.at(j).toObject()["issued_date"].toString(),
"yyyyMMdd").addDays(date.deadline_day());
                QDate today(QDate::currentDate());
                int day= iss_Date.daysTo(today);
//                qDebug()<<iss_Date.toString("yyyy.MM.dd");
//                qDebug()<<today.toString("yyyy.MM.dd");

                if(day-date.deadline_day()>0)
                    ui->label_out_BookFine_2->setText(QString::number(day*date.price()));
                else
                    ui->label_out_BookFine_2->setText(QString::number(0));
                break;
            }
        }
    }
}

```

```

        }
        break;
    }
}

if(is_data==false)
    QMessageBox::critical(this,"Error","Book ID doesnt exists or user hasnt issued the book");
}

void admin_page::on_pushButton_out_clicked()
{
    if(can_checkout){
        parsedata *json = new parsedata;
        date today;
        bool success = false;

        QJsonArray book_json = json->book_data();
        QJsonArray stu_json = json->student_data();
//    changing student data
        QJsonObject tempObj = checkout_stu_data;
        QJsonArray tempArr = tempObj["book_issued"].toArray();
        for(int i=0;i<=tempArr.count();i++){
            if(tempArr.at(i).toString().toLowerCase()==checkout_book_data["id"].toString().toLowerCase()){
                tempArr.removeAt(i);
                break;
            }
        }

        tempObj.remove("book_issued");
        tempObj.insert("book_issued",tempArr);
//    qDebug()<<tempObj;
//remove previous data and add new from stu_json
        for(int i=0;i<=json->student_no();i++){

```

```

        if(stu_json.at(i).toObject()["id"].toString().toLowerCase()==tempObj["id"].toString().toLowerCase()){
            stu_json.removeAt(i);
            stu_json.append(tempObj);
            break;
        }
    }
    QFile stu_file("../LibraryMS/JSON/student_data.json");
    if (!stu_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QMessageBox::critical(nullptr,"Error", "Error in parsing student data");
    }
    QJsonDocument doc(stu_json);
    if(!stu_file.write(doc.toJson())){
        QMessageBox::critical(nullptr,"Error", "Student Database couldnt be accessed");
    }
    else{
        success = true;
    }
    stu_file.close();
    //for the book
    if(success){
        checkout_book_data.remove("issued_by");
        checkout_book_data.remove("issued_date");
        checkout_book_data.insert("issued_by","NULL");
        checkout_book_data.insert("issued_date","NULL");
        //remove previus and add new data in book_json
        for(int i=0;i<=json->book_no();i++){

if(checkout_book_data["id"].toString().toLowerCase()==book_json.at(i).toObject()["id"].toString().toLowerCase())
{
            book_json.removeAt(i);
            book_json.append(checkout_book_data);

```

```

        break;
    }
}

QFile book_file("../LibraryMS/JSON/book_data.json");
if (!book_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
    QMessageBox::critical(nullptr, "Error", "Error in parsing book data");
}

QJsonDocument doc(book_json);
if (!book_file.write(doc.toJson())){
    QMessageBox::critical(nullptr, "Error", "Book Database couldnt be accessed");
}
else{
    QMessageBox::about(nullptr, "Success", "Checkout successfully");
    on_pushButton_clear_3_clicked();
}
book_file.close();
}

}

else{
    QMessageBox::critical(this, "Error", "checkout error\nNo data");
}
}

void admin_page::on_pushButton_clear_3_clicked()
{
    checkout_stu_data = {};
    can_checkout=false;
    checkout_book_data = {};
    ui->label_out_BookName_3->setText("");

```

```

ui->label_out_BookAuthor_3->setText("");
ui->label_out_BookFine_2->setText("");
ui->label_out_stuName_3->setText("");
ui->label_out_stuYear_3->setText("");
ui->label_out_stuCourse_3->setText("");
foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
    le->clear();
}
}

```

```

/*-----
* ----
* ----          add new page          ----
* ----
* -----*/

```

```

void admin_page::on_pushButton_add_clicked()
{
    ui->stackedWidget->setCurrentIndex(3);
}

```

```

void admin_page::on_pushButton_add_2_clicked()
{

```

```

foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
    le->clear();
}

if(ui->radioButton_addstud->isChecked())
    ui->stackedWidget->setCurrentIndex(5);
else if(ui->radioButton_addbook->isChecked())
    ui->stackedWidget->setCurrentIndex(6);
}

/*-----
* -----          add new student page          -----
* -----*/

void admin_page::on_pushButton_addConfirm_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray s_json= json->student_data();
    //checks for pre-existing data
    bool data_already_exists = false;
    for(int i=0;i<=json->student_no();i++){
        if(ui->lineEdit_addID->text()==s_json.at(i).toObject()["id"].toString()){
            QMessageBox::critical(nullptr,"Error","ID already exists");
            data_already_exists=true;
        }
    }
}

//      add new student
if(data_already_exists==false){
    student new_stu;
    new_stu.new_id(ui->lineEdit_addID->text());
    //  qDebug()<<new_stu.id();
    new_stu.new_name(ui->lineEdit_addName->text());
    new_stu.new_password(ui->lineEdit_addPass->text());
}

```

```

new_stu.new_course(ui->lineEdit_addCourse->text());
new_stu.new_year(ui->lineEdit_addYear->text());
new_stu.new_part(ui->lineEdit_addPart->text());
new_stu.new_roll(ui->lineEdit_addID->text());
QJsonObject new_stu_obj = new_stu.stu_to_qjsonobj();

s_json.append(new_stu_obj);
// qDebug()<<s_json;
QFile s_file("../LibraryMS/JSON/student_data.json");
if (!s_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
    QMessageBox::critical(nullptr,"Error", "Error in parsing student data");
}
QJsonDocument doc(s_json);
if(!s_file.write(doc.toJson()))
    QMessageBox::critical(nullptr,"Error", "Data could not be added");
else{
    QMessageBox::about(nullptr,"Success","Data added successfully");
    ui->stackedWidget->setCurrentIndex(3);
}
s_file.close();
}
}

/*-----
* -----          add new book page          -----
* -----*/

void admin_page::on_pushButton_addnewbook_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray b_json= json->book_data();

```

```

//checks for pre-existing data
bool data_already_exists = false;
for(int i=0;i<=json->book_no();i++){
    if(ui->lineEdit_addBookID->text()==b_json.at(i).toObject()["id"].toString()){
        QMessageBox::critical(nullptr,"error","ID already exists");
        data_already_exists=true;
    }
}

if(data_already_exists==false){
    book new_book;
    new_book.new_id(ui->lineEdit_addBookID->text());
    new_book.new_name(ui->lineEdit_addBookName->text());
    new_book.new_author(ui->lineEdit_addBookAuthor->text());
    new_book.new_pub_year(ui->lineEdit_addBookYear->text());

    QJsonObject new_book_obj = new_book.book_to_qjsonobj();

    b_json.append(new_book_obj);
//   qDebug()<<s_json;
    QFile b_file("../LibraryMS/JSON/book_data.json");
    if (!b_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QMessageBox::critical(nullptr,"Error", "Error in parsing book data");
    }
    QJsonDocument doc(b_json);
    if(!b_file.write(doc.toJson()))
        QMessageBox::critical(nullptr,"Error", "Data could not be added");
    else{
        QMessageBox::about(nullptr,"Success", "Data added successfully");
        b_file.close();
    }
}

```



```

        ui->stackedWidget->setCurrentIndex(3);
    }

}

}

/*-----
* ----
* ----          edit/delete page          ----
* ----
* -----*/

void admin_page::on_pushButton_edit_clicked()
{
    ui->stackedWidget->setCurrentIndex(4);
}

/*-----
* ----          edit/delete student data page          ----
* -----*/

QString edit_stud_id_temp;

void admin_page::on_pushButton_edit_stuID_SEARCH_clicked()
{
    QString input_id = ui->lineEdit_stu_id_edit->text().toLowerCase();
    // qDebug()<<input_id;

```

```

parsedata temp;

QJsonArray s_temp=temp.student_data();

bool is_data = false;

for(int i=0;i<=temp.student_no();i++){
    if(s_temp.at(i).toObject()["id"].toString().toLowerCase()==input_id){
//        qDebug()<<s_temp.at(i).toObject()["id"].toString();
        is_data=true;

        edit_stud_id_temp=s_temp.at(i).toObject()["id"].toString();

        ui->stackedWidget->setCurrentIndex(7);

        ui->lineEdit_edit_id->setText(s_temp.at(i).toObject()["id"].toString());
        ui->lineEdit_edit_name->setText(s_temp.at(i).toObject()["name"].toString());
        ui->lineEdit_edit_year->setText(s_temp.at(i).toObject()["year"].toString());
        ui->lineEdit_edit_part->setText(s_temp.at(i).toObject()["part"].toString());
        ui->lineEdit_edit_password->setText(s_temp.at(i).toObject()["password"].toString());
        ui->lineEdit_edit_course->setText(s_temp.at(i).toObject()["course"].toString());
    }
}

if(is_data==false){
    QMessageBox::critical(nullptr,"Error","Student data doesnt exist");
}
}

void admin_page::on_pushButton_edit_confirm_clicked()
{
    parsedata *json = new parsedata;

    QJsonArray s_json= json->student_data();

    //checks for pre-existing data

    bool data_already_exists = false;

    for(int i=0;i<=json->student_no();i++){

```

```

        if(ui->lineEdit_edit_id->text().toLower()==s_json.at(i).toObject()["id"].toString() && ui-
>lineEdit_edit_id->text().toLower()!=edit_stud_id_temp.toLower()){

            QMessageBox::critical(nullptr,"error", "ID already exists");

            data_already_exists=true;

        }
    }

    if(data_already_exists==false){
//        qDebug()<<"we are in";

        student new_stu;

        new_stu.new_id(ui->lineEdit_edit_id->text());
        new_stu.new_name(ui->lineEdit_edit_name->text());
        new_stu.new_password(ui->lineEdit_edit_password->text());
        new_stu.new_course(ui->lineEdit_edit_course->text());
        new_stu.new_year(ui->lineEdit_edit_year->text());
        new_stu.new_part(ui->lineEdit_edit_part->text());
        new_stu.new_roll(ui->lineEdit_edit_id->text());

        QJsonObject new_stu_obj = new_stu.stu_to_qjsonobj();
        for(int i=0;i<=json->student_no();i++){

            if(edit_stud_id_temp==s_json.at(i).toObject()["id"].toString()){

                s_json.removeAt(i);

                break;

            }

        }

//        qDebug()<<s_json;

        s_json.append(new_stu_obj);

//        qDebug()<<s_json;

        QFile s_file("../LibraryMS/JSON/student_data.json");

        if (!s_file.open(QIODevice::WriteOnly | QIODevice::Text)) {

            QMessageBox::critical(nullptr,"Error", "Error in parsing student data");

        }
    }

```

```

QJsonDocument doc(s_json);
if(!s_file.write(doc.toJson()))
    QMessageBox::critical(nullptr, "Error", "Data could not be edited");
else{
    QMessageBox::about(nullptr, "Success", "Data edited successfully");
    foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
        le->clear();
    }
    ui->stackedWidget->setCurrentIndex(4);
}
s_file.close();
}

}

void admin_page::on_pushButton_delete_data_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray s_json= json->student_data();
    for(int i=0;i<=json->student_no();i++){
        if(edit_stud_id_temp==s_json.at(i).toObject()["id"].toString()){
            s_json.removeAt(i);
            break;
        }
    }
}

// qDebug()<<s_json;

QMessageBox::StandardButton confirm;

confirm = QMessageBox::question(this, "Confirm", "Are you sure you want to delete?\nID:
"+edit_stud_id_temp,

```

```

        QMessageBox::Yes|QMessageBox::No);
if(confirm == QMessageBox::Yes){
    QFile s_file("../LibraryMS/JSON/student_data.json");
    if (!s_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QMessageBox::critical(nullptr,"Error", "Error in parsing student data");
    }
    QJsonDocument doc(s_json);
    if(!s_file.write(doc.toJson()))
        QMessageBox::critical(nullptr,"Error", "Data could not be deleted");
    else{
        QMessageBox::about(nullptr,"Success", "Data deleted successfully");
        foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
            le->clear();
        }
        ui->stackedWidget->setCurrentIndex(4);
    }
    s_file.close();
}
}

/*-----
* -----          edit/delete book data page          -----
* -----*/

QString edit_book_id_temp;
void admin_page::on_pushButton_edit_bookID_search_clicked()
{
    QString input_id = ui->lineEdit_book_id_edit->text().toLower();
    // qDebug()<<input_id;
    parsedata temp;
    QJsonArray book_temp=temp.book_data();
    bool is_data = false;

```

```

for(int i=0;i<=temp.book_no();i++){
    if(book_temp.at(i).toObject()["id"].toString().toLowerCase()==input_id){
//        qDebug()<<book_temp.at(i).toObject()["id"].toString();
        is_data=true;
        edit_book_id_temp=book_temp.at(i).toObject()["id"].toString();
        ui->stackedWidget->setCurrentIndex(8);
        ui->lineEdit_edit_book_id->setText(book_temp.at(i).toObject()["id"].toString());
        ui->lineEdit_edit_book_name->setText(book_temp.at(i).toObject()["name"].toString());
        ui->lineEdit_edit_book_year->setText(book_temp.at(i).toObject()["pub_year"].toString());
        ui->lineEdit_edit_book_author->setText(book_temp.at(i).toObject()["author"].toString());
    }
}
if(is_data==false){
    QMessageBox::critical(nullptr,"Error","book data doesnt exist");
}
}

void admin_page::on_pushButton_edit_book_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray book_json= json->book_data();
    //checks for pre-existing data
    bool data_already_exists = false;
    for(int i=0;i<=json->book_no();i++){
        if(ui->lineEdit_edit_id->text().toLowerCase()==book_json.at(i).toObject()["id"].toString() && ui->lineEdit_edit_id->text().toLowerCase()!=edit_book_id_temp.toLowerCase()){
            QMessageBox::critical(nullptr,"error","ID already exists");
            data_already_exists=true;
        }
    }
}

```

```

if(data_already_exists==false){
//    qDebug()<<"we are in";
    book new_book;
    new_book.new_id(ui->lineEdit_edit_book_id->text());
    new_book.new_name(ui->lineEdit_edit_book_name->text());
    new_book.new_author(ui->lineEdit_edit_book_author->text());
    new_book.new_pub_year(ui->lineEdit_edit_book_year->text());
    QJsonObject new_book_obj = new_book.book_to_qjsonobj();
    for(int i=0;i<=json->book_no();i++){
        if(edit_book_id_temp==book_json.at(i).toObject()["id"].toString()){
            book_json.removeAt(i);
            break;
        }
    }
//    qDebug()<<book_json;
    book_json.append(new_book_obj);
//    qDebug()<<book_json;
    QFile s_file("../LibraryMS/JSON/book_data.json");
    if (!s_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QMessageBox::critical(nullptr,"Error", "Error in parsing book data");
    }
    QJsonDocument doc(book_json);
    if(!s_file.write(doc.toJson()))
        QMessageBox::critical(nullptr,"Error", "Data could not be edited");
    else{
        QMessageBox::about(nullptr,"Success","Data edited successfully");
        foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
            le->clear();
        }
        ui->stackedWidget->setCurrentIndex(4);
    }
}

```

```

    }
    s_file.close();
}
}

void admin_page::on_pushButton_delete_book_clicked()
{
    parsedata *json = new parsedata;
    QJsonArray book_json= json->book_data();
    for(int i=0;i<=json->book_no();i++){
        if(edit_book_id_temp==book_json.at(i).toObject()["id"].toString()){
            book_json.removeAt(i);
            break;
        }
    }
}

// qDebug()<<book_json;

QMessageBox::StandardButton confirm;

confirm = QMessageBox::question(this, "Confirm", "Are you sure you want to delete?\nID:
"+edit_book_id_temp,

                                QMessageBox::Yes|QMessageBox::No);

if(confirm == QMessageBox::Yes){
    QFile book_file("../LibraryMS/JSON/book_data.json");
    if (!book_file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        QMessageBox::critical(nullptr,"Error", "Error in parsing book data");
    }
    QJsonDocument doc(book_json);
    if(!book_file.write(doc.toJson()))
        QMessageBox::critical(nullptr,"Error", "Data could not be deleted");
    else{

```



```
QMessageBox::about(nullptr, "Success", "Data deleted successfully");  
foreach(QLineEdit* le, findChildren<QLineEdit*>()) {  
    le->clear();  
}  
ui->stackedWidget->setCurrentIndex(4);  
}  
book_file.close();  
}  
}
```

## APPENDIX C (JSON file)

### Student\_data.json

```
[
  {
    "access_level": "student",
    "book_issued": [
    ],
    "course": "BEI",
    "id": "076bei026",
    "name": "Prashant Karn",
    "part": "2",
    "password": "1234",
    "roll": "076bei026",
    "year": "1 "
  },
  {
    "access_level": "student",
    "book_issued": [
    ],
    "course": "BEI",
    "id": "076bei024",
    "name": "Prakriti Timalisina",
    "part": "2",
    "password": "1234",
    "roll": "076bei024",
    "year": "1 "
  },
  {
    "access_level": "student",
```

```
"book_issued": [  
  ],  
  "course": "BEI",  
  "id": "076bei019",  
  "name": "kshitiz pandey",  
  "part": "2",  
  "password": "1234",  
  "roll": "076bei019",  
  "year": "1"  
},  
{  
  "access_level": "student",  
  "book_issued": [  
  ],  
  "course": "BEI",  
  "id": "076BEI041",  
  "name": "Dikshya parajuli",  
  "part": "2",  
  "password": "1234",  
  "roll": "076BEI041",  
  "year": "1"  
},  
{  
  "access_level": "student",  
  "book_issued": [  
  ],  
  "course": "bei",  
  "id": "076BEI035",  
  "name": "Sandesh sitaula",  
  "part": "2",
```

```

    "password": "1234",
    "roll": "076BEI035",
    "year": "1 "
  },
  {
    "access_level": "student",
    "book_issued": [
      "b004",
      "B005",
      "B006",
      "B001",
      "B007"
    ],
    "course": "BXX",
    "id": "",
    "name": "The Void",
    "part": "2",
    "password": "",
    "roll": "",
    "year": "1 "
  },
  {
    "access_level": "student",
    "book_issued": [
      "B002",
      "B003"
    ],
    "course": "BEI",
    "id": "076bei025",
    "name": "Pramesh Shrestha",

```

```
"part": "2",
"password": "1234",
"roll": "076bei025",
"year": "1"
},
{
  "access_level": "student",
  "book_issued": [
    "B009"
  ],
  "course": "BEI",
  "id": "076bei028",
  "name": "Rahul Shrestha",
  "part": "2",
  "password": "12345",
  "roll": "076bei028",
  "year": "1"
}
]
```

### admin\_data.json

```
[{
  "id": "admin0",
  "name": "admin0",
  "password": "admin",
  "access_level": "admin"
},
{
  "id": "admin1",
  "name": "admin1",
  "password": "admin",
  "access_level": "admin"
},
{
  "id": "",
  "name": "Admin void",
  "password": "",
  "access_level": "admin"
}
]
```

### Book\_data.json

```
[
{
  "author": "Rick Riorden",
  "id": "B004",
  "issued_by": "",
  "issued_date": "20201214",
```

```

    "name": "Percy Jackson and the lightning theid",
    "pub_year": "2005"
  },
  {
    "author": "J. K. Rowling",
    "id": "B005",
    "issued_by": "",
    "issued_date": "20201201",
    "name": "Harry Potter and the philosopher's stone",
    "pub_year": "1997"
  },
  {
    "author": "mathematician",
    "id": "B006",
    "issued_by": "",
    "issued_date": "20201114",
    "name": "Mathematics",
    "pub_year": "2000"
  },
  {
    "author": "english guy",
    "id": "B008",
    "issued_by": " ",
    "issued_date": "20201213",
    "name": "ENGLISH",
    "pub_year": "2009"
  },
  {
    "author": "A guy with a Name",
    "id": "B001",

```

```

    "issued_by": "",
    "issued_date": "20201215",
    "name": "A Book Title",
    "pub_year": "1430"
  },
  {
    "author": "SCIENCE GUY",
    "id": "B007",
    "issued_by": "",
    "issued_date": "20201215",
    "name": "SCIENCE",
    "pub_year": "1998"
  },
  {
    "author": "A guy with a Name",
    "id": "B003",
    "issued_by": "076BEI025",
    "issued_date": "20201216",
    "name": "A Book Title",
    "pub_year": "1420"
  },
  {
    "author": "Mark Manson",
    "id": "B009",
    "issued_by": "076bei028",
    "issued_date": "20201216",
    "name": "The subtle art of not giving a fck",
    "pub_year": "2016"
  },
  {

```



```
"author": "A guy with a Name",  
"id": "B002",  
"issued_by": "076BEI025",  
"issued_date": "20201216",  
"name": "A Book Title",  
"pub_year": "1420"  
}  
]
```