**MINISTRY OF EDUCATION AND TRAINING**

**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH-QUALITY TRAINING**



Final Project

Library Management System

Course: Database Management System

Period: 7 - 10

Lecturer: Nguyen Thanh Son

|  |  |
| --- | --- |
| **NAME** | **STUDENT’S ID** |
| Vo Minh Huy | 18110020 |
| Ho Dac Hieu | 18110014 |
| Le Trinh Hoang Phu | 18110037 |
| Dao Trung Nam | 18110031 |
| Nguyen Dinh Long | 18110027 |

*HCMC March, 2021  
Last update: March 28, 2021*

**SCORE**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Content | Presentation | Total |
| Point |  |  |  |

**REMARKS OF TEACHERS**

………………………………………………………………………………………………………………………………………………………………….

…………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………

Ho Chi Minh City, March 28th, 2021

Teacher’s score

(Signature and full name)

**Nguyen Thanh Son**

**THANK YOU**

In order to successfully complete this topic and this report, we would like to extend our sincere thanks to the lecturer, Dr. Nguyen Thanh Son, who directly supported us throughout the process of making the topic. We thank the teacher for giving advice from his practical experience to guide us in the right direction with the requirements of the selected topic, always answer questions and give suggestions and corrections. time to help us overcome our shortcomings and complete it well as well as on schedule.

We also would like to express our sincere thanks to the teachers in the

High Quality Education Department in general and the Information Technology industry in particular for their dedicated knowledge to help us have a foundation to make. This topic has created conditions for us to learn and perform well on the topic. Along with that, we would like to thank our classmates for providing useful information and knowledge to help us improve our topic.

The topic and report are made by us in a short time, with limited knowledge and many other limitations in terms of technical and experience in implementing a software project. Therefore, in the process of creating a topic with shortcomings is inevitable, we look forward to receiving valuable comments from the teachers to make our knowledge more complete and We can do even better next time.

We sincerely thank you.

At the end, we would like to wish all of you teachers, ladies and gentlemen, always having abundant health and more success in the career of growing people. Once again we sincerely thanks.

**SUBJECT DESCRIPTION OF THE SUBJECTS OF DATA BASIS MANAGEMENT**

Implementing student: **Vo Minh Huy** ID’s student: **18110020**

Implementing student: **Le Trinh Hoang Phu** ID’s student: **18110037**

Implementing student: **Ho Dac Hieu** ID’s student: **18110014**

Implementing student: **Dao Trung Nam** ID’s student: **18110031**

Implementing student: **Nguyen Dinh Long** ID’s student: **18110027**

Field: **Information Technology**

Name’s project: **Library Management System**

Instructor: **Dr. Nguyen Thanh Son**

**Tasks of the topic:** Building Library Management System to ensure the following functions:

1. Manage user information and books.

2. Manage your borrowing time.

3. Manage projects and work to be done in each project.

4. Manage user accounts.

**TABLE OF CONTENTS**

[**table Image list** 7](#_Toc72617741)

[**i.General** 8](#_Toc72617745)

[**iI.content** 12](#_Toc72617746)

[**iII.Interface design** 28](#_Toc72617747)

[**iV. conClude** 38](#_Toc72617748)

*All images which are presented in this report are listed here.*

|  |
| --- |
| **table Image list** |
|  |

|  |  |  |
| --- | --- | --- |
| Figure | Caption | Page |
| 1.1.2 | Database Diagram | 13 |
| 3.1 | Login Form Design | 29 |
| 3.2 | Main Form | 30 |
| 3.3 | Viewbook Form | 31 |
| 3.4 | Add Book Form | 32 |
| 3.5 | View Student Form | 33 |
| 3.6 | Add Student Form | 33 |
| 3.7 | View Teacher Form | 34 |
| 3.8 | Add Teacher Form | 35 |
| 3.9 | Lending Book Form | 36 |
| 3.10 | Returning Book Form | 37 |
| 3.11 | Update information Form | 38 |

**i.General**

**1. REASONS FOR CHOOSING THE PROJECT**

With the increase in the number of readers, better management of the library system is required. The Library management system focuses on improving the management of libraries in a city or town. What If you can check whether a book is available in the library through your phone or what if instead of having different library cards for different libraries you can just have one or you can reserve a book or issue a book from your phone sitting at your home. The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from a library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

The project help everyone to maintain the details of books and library members of different libraries. The main purpose of this project is to maintain an easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number). Moreover, the user can check all these features from their home.

**2. FUNCTIONS OF THE TOPIC**

* **FUNCTIONAL REQUIREMENTS**
* Register

+ The user will register or sign up.

+ The library management have to provide information about the name of library, address, phone, number, email, id.

+ Regular persons/ students have to provide details about their name of address, phone number, email id.

* Sign up

+ Input/ Output: Detail and confirm of registration status and a membership number and password will be generated and mailed to the user.

+ Processing: All details will be checked and will notify if an error occurs.

* Login

+ Input: Enter the membership number and password provided.

+ Output: User will be able to use the features of software.

* Manage books by users

+ Books issued.

+ Search.

+ Issues book.

+ Renew book.

+ Return the book to the library.

+ Reserve book.

+ Fines for book returns overdue.

* **NON FUNCTIONAL REQUIREMENS**
* Usability Requirement

+ The system shall allow the users to access the system from a common interface.

* Availability Requirement

+ The system should always be 100% available to the user and available 24 hours, a week.

* Efficiency Requirement

+ Mean Time to Repair (MTTR) - Even if the system fails, the system will be recovered back up within an hour or less.

* Accuracy

+ The system must provide accurate real-time information and relevant to various concurrent issues. The system should provide 100% access reliability.

* Performance Requirement

+ The information is refreshed depending upon whether some updates have occurred or not in the application. The system will reply members within not less than two seconds from the time of request. The response to the information would not take more than 5 seconds to appear on the screen. The system will be allowed to take longer when performing large processing jobs.

* Reliability Requirement

+ The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

**3. SYSTEM ANALYSIS AND DESIGN**

Having 3 levels of user:

* User module: In user module, user will check the availability of the books.

+ Issue book: book code, title, author's name, date of issue.

+ Reserve book: code book account, account name set, the name of the book, the code book, the book's title, author's name, date of issue, date of loan, date of payment, loan period.

+ Return book: book code, date of loan, date of payment, time of borrowing, fine.

+ Fine details: time to borrow, time to return books, fines.

* Library module:

+ Add new book: book code, title, author's name, date of issue, date of loan, date of payment, date of borrowing, account number, name, age of borrower, fine, quantity of the book.

+ Remove books

+ Update detail of book: book code, title, author's name, date of issue, date of loan, date to be paid, time of borrowing.

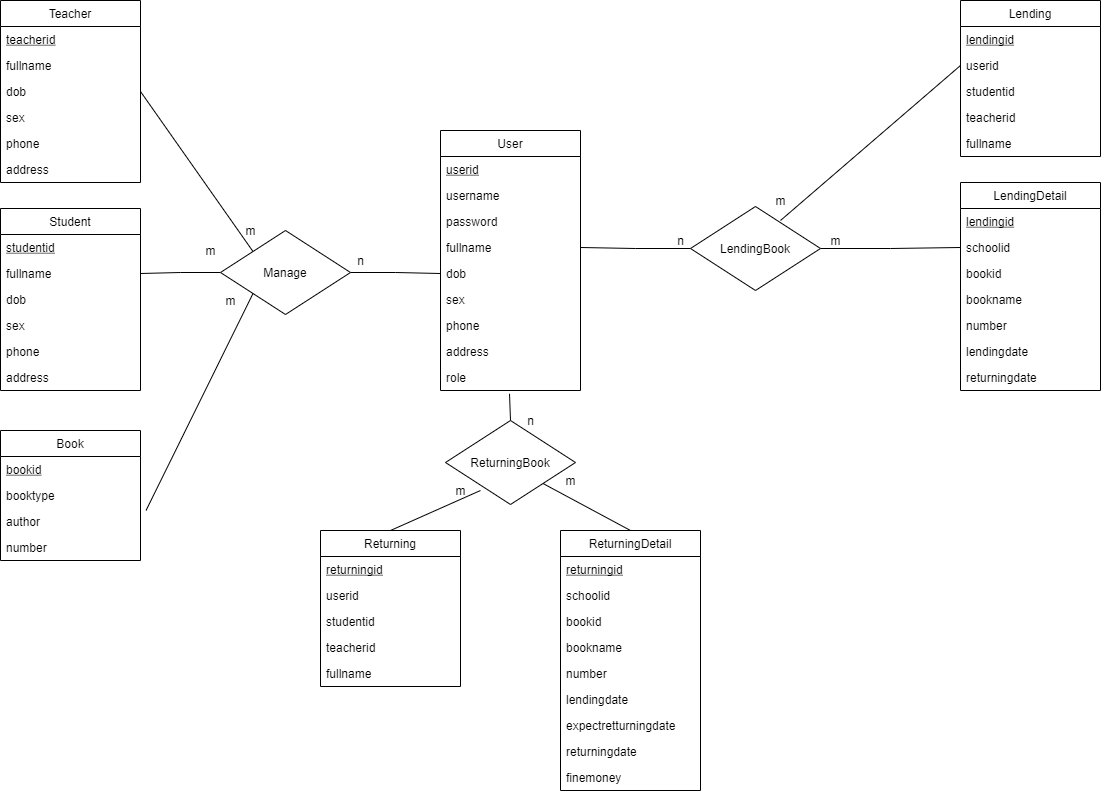
* Administration module:

+ Register user: user account code, registered account name, age, address.

+ Entry book details: book code, title, author's name, issue date.

+ Book issue: book code, title, date of issue.

**iI.content**

1. **DATABASE DESIGN**
   1. **Relational Model**

**User** (userid, username, password, fullname, dob, sex, phone, address, role)

**Teacher** (teacherid, fullname, dob, sex, phone, address)

**Student** (studentid, fullname, dob, sex, phone, address)

**Book** (bookid, booktype, author, number)

**Lending** (lendingid, userid, studentid, teacherid, fulllname)

**LendingDetail** (lendingid, schoolid, bookid, bookname, number, lendingdate, returningdate)

**Returning** (returningid, userid, studentid, teacherid, fullname)

**ReturningDetail** (returningid, schoolid, bookid, bookname, number, lendingdate, expectreturningdate, returningdate, finemoney)

* 1. **Create databases**
     1. **Description of Tables**

|  |  |  |
| --- | --- | --- |
| **No** | **Table’s name** | **Purpose** |
| 1 | users | -Store user’s individual information and login information |
| 2 | student | -Store student’s personal information |
| 3 | teacher | -Store teacher’s personal information |
| 4 | book | -Store book’s information in library system |
| 5 | lending | -Store id about people who are borrowing books |
| 6 | lendingdetail | -Contains user’s detail information about borrow/return datetime |
| 7 | returningdetail | -Contains user’s detail information about return datetime |
| 8 | returning | -Store user’s id on returning datetime. |

* + 1. **Database Diagram**

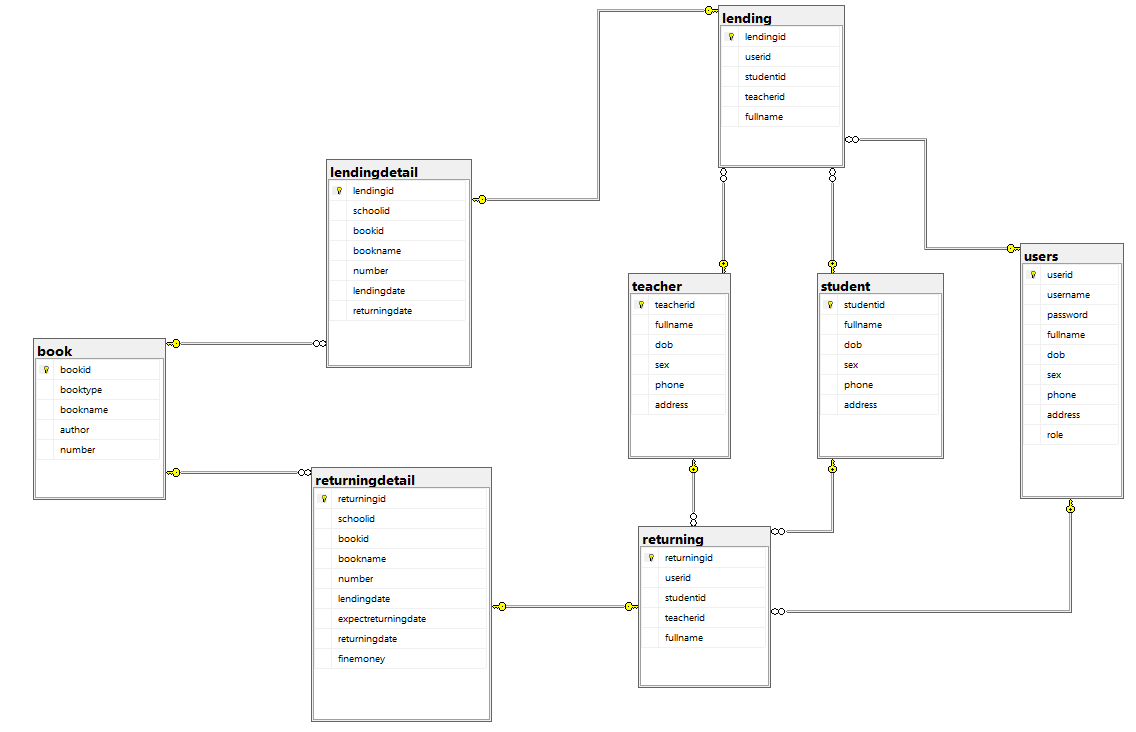


Figure 1.1.2 Database Diagram

1. **CREATE VIEW, STORED PROCEDURE, FUNCTION, DATA LEVEL TRIGGER**
   1. **Create the Views**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| **1** | ViewUsers\_ADMIN | CREATE VIEW ViewUsers\_ADMIN  AS  (  SELECT userid, username, password, fullname, dob, sex, phone, address, role  FROM users  where role = 'ADMIN'  ) |
| **2** | ViewUsers\_LIBRARIAN | CREATE VIEW ViewUsers\_LIBRARIAN  AS  (  SELECT userid, username, password, fullname, dob, sex, phone, address, role  FROM users  where role = 'LIBRARIAN'  ) |
| **3** | ViewUsers\_MANAGER | CREATE VIEW ViewUsers\_MANAGER  AS  (  SELECT userid, username, password, fullname, dob, sex, phone, address, role  FROM users  where role = 'MANAGER'  ) |
| **4** | ViewUsers\_FULLACCESSADMIN | CREATE VIEW ViewUsers\_FULLACCESSADMIN  AS  (  SELECT userid, username, password, fullname, dob, sex, phone, address, role  FROM users  where role = 'FULLACCESSADMIN'  ) |
| **5** | Select\_All\_Book | CREATE VIEW Select\_All\_Book  AS  (  select \*  from book  ) |
| **6** | Select\_All\_Teacher | CREATE VIEW Select\_All\_Teacher  AS  (  select \*  from teacher  ) |
| **7** | Select\_All\_Student | CREATE VIEW Select\_All\_Student  AS  (  select \*  from student  ) |
| **8** | Select\_Teacher\_BorrowBook | CREATE VIEW Select\_Teacher\_BorrowBook  AS  (  select \*  from lending  where teacherid is not null  ) |
| **9** | Select\_Student\_BorrowBook | CREATE VIEW Select\_Student\_BorrowBook  AS  (  select \*  from lending  where studentid is not null  ) |
| **10** | Select\_All\_LendingDetail | CREATE VIEW Select\_All\_LendingDetail  AS  (  select \*  from lendingdetail  ) |

* 1. **Create Store Procedure**
     1. **Store Proceduce for user**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| **1** | Add\_User | CREATE PROC Add\_User  @username varchar(15),  @password varchar(50), @fullname nvarchar(50),  @dob date, @sex bit, @phone int, @address nvarchar(100),  @role nchar(50)  AS  BEGIN  INSERT dbo.users(username, password, fullname, dob, sex, phone, address, role)  VALUES (@username, @password, @fullname, @dob, @sex, @phone, @address, @role)  END |
| **2** | Edit\_User | CREATE PROC Edit\_User  @userid int,  @password varchar(50), @fullname varchar(50),  @dob date, @sex bit, @phone int, @address nvarchar(100)  AS  BEGIN  UPDATE dbo.users  SET password = @password,  fullname = @fullname, dob = @dob, sex = @sex, phone = @phone, address = @address  WHERE userid = @userid  END |
| **3** | Edit\_User\_Admin | CREATE PROC Edit\_User\_Admin  @userid int, @fullname varchar(50),  @dob date, @sex bit, @phone int, @address nvarchar(100), @role nchar(50)  AS  BEGIN  UPDATE dbo.users  SET  fullname = @fullname, dob = @dob, sex = @sex, phone = @phone, address = @address, role = @role  WHERE userid = @userid  END |
| **4** | Delete\_User | CREATE PROC Delete\_User  @userid int  AS  BEGIN  DELETE FROM dbo.users  WHERE userid = @userid  END |
| **5** | Get\_User\_Info\_Login | CREATE PROC Get\_User\_Info\_Login  @username varchar(15), @password varchar(50), @role nchar(50)  AS  begin  select \*  from dbo.users  where username = @username and password = @password and role = @role  end |

* + 1. **Store Proceduce for Book**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| 1 | Add\_Book | --Add new book  CREATE PROC Add\_Book  @booktype varchar(15), @bookname nvarchar(50), @author nvarchar(50), @number int  as  begin  insert into dbo.book(booktype, bookname, author, number)  values (@booktype, @bookname, @author, @number)  end |
| 2 | Edit\_Book\_Detail | --Update Book  CREATE PROC Edit\_Book\_Detail  @bookid int, @booktype varchar(15), @bookname nvarchar(50), @author nvarchar(50), @number int  as  begin  update dbo.book  set booktype = @booktype, bookname = @bookname, author = @author, number = @number  where bookid = @bookid  end |
| 3 | Delete\_Book\_Detail | --Delete Book  CREATE PROC Delete\_Book\_Detail  @bookid int  as  begin  delete  from dbo.book  where bookid = @bookid  end |
| 4 | Add\_Book\_Borrowed | --Add book detail that student or teacher borrow  CREATE PROC Add\_Book\_Borrowed  @Schoolid int, @Bookid int, @Bookname nvarchar(50), @Number int, @Lendingdate date, @Returningdate date  as  begin  insert into lendingdetail(schoolid, bookid, bookname, number, lendingdate, returningdate)  values (@Schoolid, @Bookid, @Bookname, @Number, @Lendingdate, @Returningdate)  end |

* + 1. **Store Proceduce for Student**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| 1 | Add\_Student | --Add new student  CREATE PROC Add\_Student  @studentid int, @fullname nvarchar(50), @dob date, @sex bit, @phone int, @address nvarchar(100)  as  begin  insert into dbo.student(studentid, fullname, dob, sex, phone, address)  values (@studentid, @fullname, @dob, @sex, @phone, @address)  end |
| 2 | Edit\_Student\_Detail | --Update Student  CREATE PROC Edit\_Student\_Detail  @studentid int, @fullname nvarchar(50), @dob date, @sex bit, @phone int, @address nvarchar(100)  as  begin  update dbo.student  set fullname = @fullname, dob = @dob, sex = @sex, phone = @phone, address = @address  where studentid = @studentid  end |
| 3 | Delete\_Student\_Detail | --Delete Student  CREATE PROC Delete\_Student\_Detail  @studentid int  as  begin  delete  from dbo.student  where studentid = @studentid  end |

* + 1. **Store Proceduce for Teacher**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| 1 | Edit\_Teacher\_Detail | --Update Teacher  CREATE PROC Edit\_Teacher\_Detail  @teacherid int, @fullname nvarchar(50), @dob date, @sex bit, @phone int, @address nvarchar(100)  as  begin  update dbo.teacher  set fullname = @fullname, dob = @dob, sex = @sex, phone = @phone, address = @address  where teacherid = @teacherid  end |
| 2 | Delete\_Teacher\_Detail | --Delete Teacher  CREATE PROC Delete\_Teacher\_Detail  @teacherid int  as  begin  delete  from dbo.teacher  where teacherid = @teacherid  end |
| 3 | Add\_Teacher | --Add new teacher  CREATE PROC Add\_Teacher  @teacherid int, @fullname nvarchar(50), @dob date, @sex bit, @phone int, @address nvarchar(100)  as  begin  insert into dbo.teacher(teacherid, fullname, dob, sex, phone, address)  values (@teacherid, @fullname, @dob, @sex, @phone, @address)  end |

* + 1. **Store Proceduce for Lending Table**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| 1 | Add\_Teacher\_Lending | --Add teacher to lending table  CREATE PROC Add\_Teacher\_Lending  @Userid int, @Teacherid int, @Fullname nvarchar(50)  as  begin  insert into lending(userid, teacherid, fullname)  values (@Userid, @Teacherid, @Fullname)  end |
| 2 | Add\_Student\_Lending | --Add student to lending table  CREATE PROC Add\_Student\_Lending  @Userid int, @Studentid int, @Fullname nvarchar(50)  as  begin  insert into lending(userid, studentid, fullname)  values (@Userid, @Studentid, @Fullname)  end |
| 3 | Delete\_User\_Lending | --Delete user out of lending table  CREATE PROC Delete\_User\_Lending  @lendingid int  as  begin  delete  from lending  where lendingid = @lendingid  end |
| 4 | Delete\_Book\_Lending | --Delete book out of lending\_detail table  CREATE PROC Delete\_Book\_Lending  @lendingid int  as  begin  delete  from lendingdetail  where lendingid = @lendingid  end |
| 5 | To\_Returning\_Detail | --add book from lending\_detail to returning\_detail  CREATE PROC To\_Returning\_Detail  @Schoolid int, @Bookid int, @Bookname nvarchar(50), @Number int, @Lendingdate date, @Expectedreturningdate date, @Returningdate date, @Finemoney int  as  begin  insert into returningdetail(schoolid, bookid, bookname, number, lendingdate, expectreturningdate, returningdate, finemoney)  values (@Schoolid, @Bookid, @Bookname, @Number, @Lendingdate, @Expectedreturningdate, @Returningdate, @Finemoney)  end |

* 1. **Create Functions**
     1. **Function for user**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| **1** | CheckDuplicateUsername | create function CheckDuplicateUsername  (  @Username nvarchar(50)  )  returns int  as  begin  return(  select count(1)  from users  where password = @Username)  end |
| **2** | Search\_book | --Keyword search for table book  create function Search\_book  (  @Search\_key\_word nvarchar(100)  )  returns table  as return (  select \*  from book  where bookname like '%' + @Search\_key\_word + '%'  ) |
| **3** | Search\_student | --Keyword search for table student  create function Search\_student  (  @Search\_key\_word nvarchar(100)  )  returns table  as return (  select \*  from student  where studentid like '%' + @Search\_key\_word + '%'  ) |
| **4** | Search\_teacher | --Keyword search for table teacher  create function Search\_teacher  (  @Search\_key\_word nvarchar(100)  )  returns table  as return (  select \*  from teacher  where teacherid like '%' + @Search\_key\_word + '%'  ) |
| **5** | Search\_teacher\_lending | --Keyword serch for teacher in lending form  create function Search\_teacher\_lending  (  @teacherid varchar(50), @fullname nvarchar(50), @phone varchar(50)  )  returns table  as return (  select \*  from teacher  where teacherid like '%' + @teacherid +'%' and fullname like '%' + @fullname +'%' and phone like '%' + @phone +'%'  ) |
| **6** | Search\_student\_lending | --Keyword serch for student in lending form  create function Search\_student\_lending  (  @studentid varchar(50), @fullname nvarchar(50), @phone varchar(50)  )  returns table  as return (  select \*  from student  where studentid like '%' + @studentid +'%' and fullname like '%' + @fullname +'%' and phone like '%' + @phone +'%'  ) |
| **7** | Search\_book\_lending | --Keyword serch for book in lending form  create function Search\_book\_lending  (  @bookname varchar(50), @number varchar(50)  )  returns table  as return (  select \*  from book  where bookname like '%' + @bookname +'%' and number like '%' + @number +'%'  ) |
| **8** | Search\_teacher\_borrow | --Keyword search for teacher that borrow book  create function Search\_teacher\_borrow  (  @teacherid nvarchar(50), @fullname nvarchar(50)  )  returns table  as return (  select \*  from lending  where teacherid like '%' + @teacherid+'%' and fullname like '%' + @fullname +'%'  ) |
| **9** | Search\_student\_borrow | --Keyword search for student that borrow book  create function Search\_student\_borrow  (  @studentid nvarchar(50), @fullname nvarchar(50)  )  returns table  as return (  select \*  from lending  where studentid like '%' + @studentid +'%' and fullname like '%' + @fullname +'%'  ) |
| **10** | Search\_Book\_LendingDetail | --Keyword search for book in lending\_detail  create function Search\_Book\_LendingDetail  (  @bookname nvarchar(50), @number nvarchar(50)  )  returns table  as return (  select \*  from lendingdetail  where bookname like '%' + @bookname +'%' and number like '%' + @number +'%'  ) |

* 1. **Create Trigger**

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Code** |
| **1** | lending\_book | --Trigger to minus number of books when lending  create trigger lending\_book ON lendingdetail  after INSERT  as  declare @newbookid int, @totalnumber int  select @newbookid = bookid,@totalnumber =number  from lendingdetail  update book set number = number - @totalnumber where book.bookid = @newbookid; |
| **2** | return\_book | --Trigger to plus number of books when returning  create trigger return\_book ON returningdetail  after INSERT  as  declare @newbookid int, @totalnumber int  select @newbookid = bookid,@totalnumber =number  from returningdetail  update book set number = number + @totalnumber where book.bookid = @newbookid; |

**iII.Interface design**

1. **Login Form**

Login form use to sign in main form of library management system.

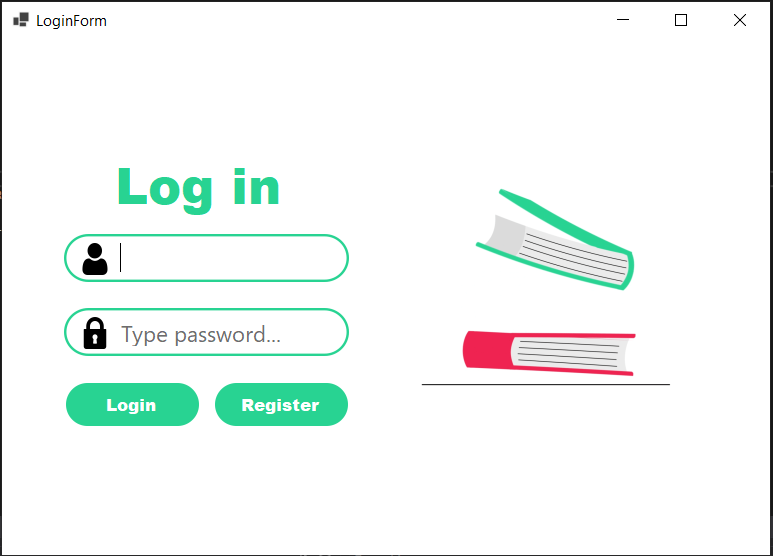


Figure 3.1 Login Form Design

1. **Main Form**

The Main Form is the main page for navigating to the management system functions. In addition, the homepage also helps us to quickly view quick functions in the system such as view teachers, students and add teachers / students.

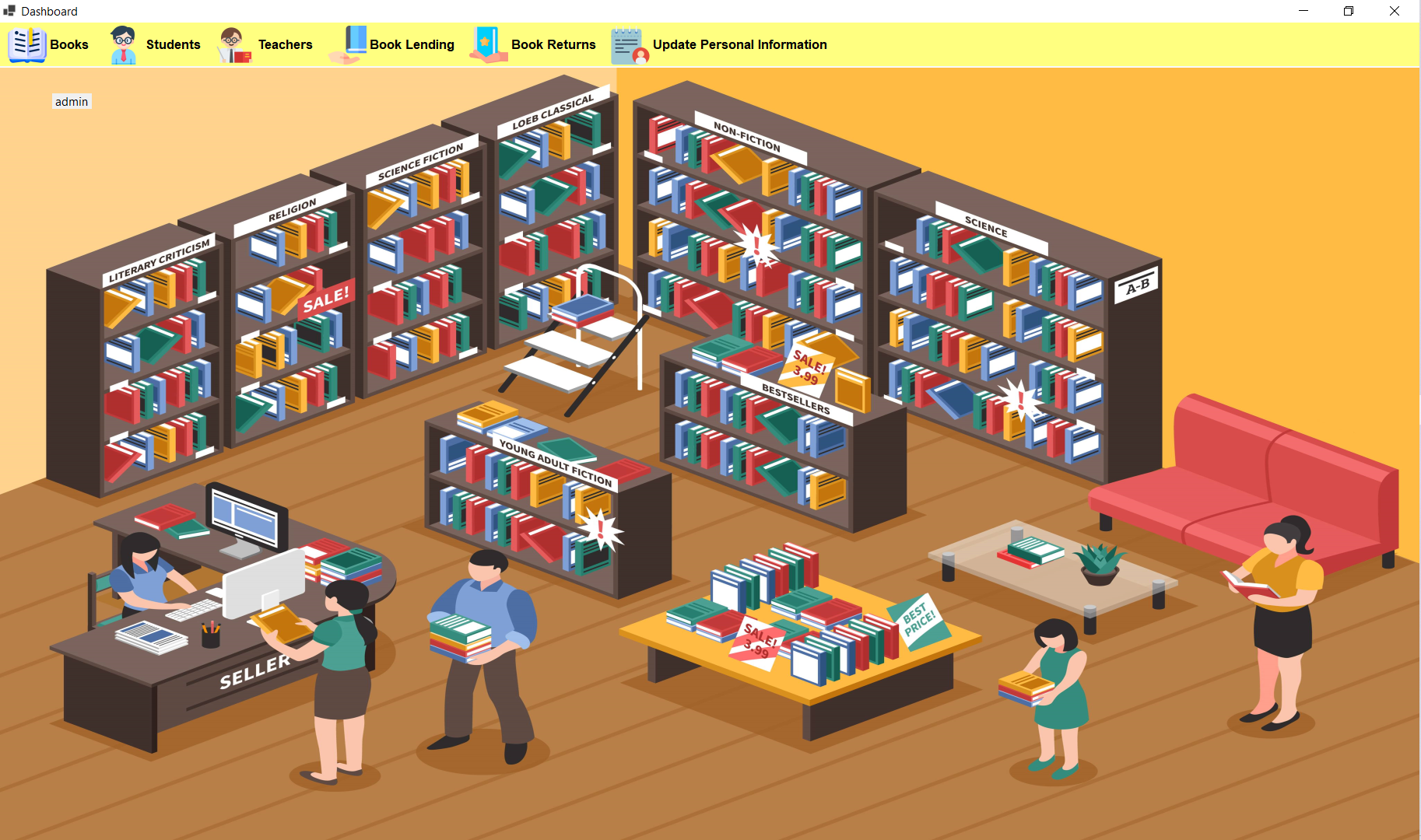


Figure 3.2.Main Form

1. **Viewbook Form**

The form displays information about all the books in the system such as id, type of book, name, author, quantity. Manager can edit and delete books.

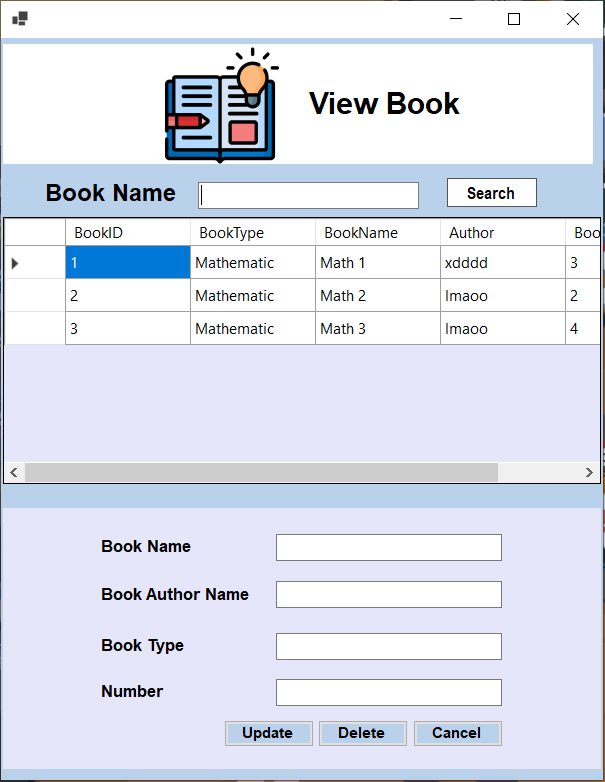


Figure 3.3. Viewbook Form

1. **Add Book Form**

Form used to add books to the system.

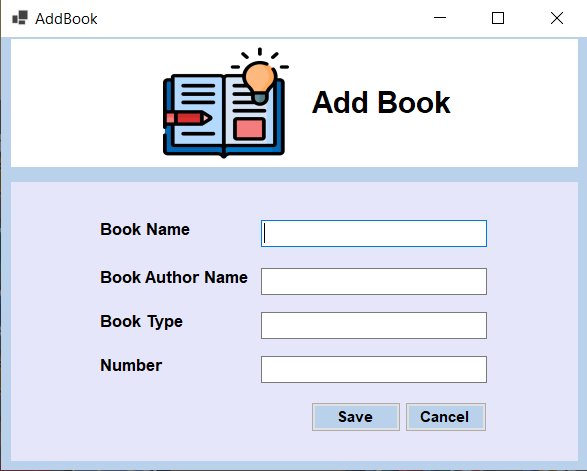


Figure 3.4. Add Book Form

1. **View Student Form**

The form shows us a list of students who have borrowed books, and can also search for students. You can also view, edit and delete the details of each student.

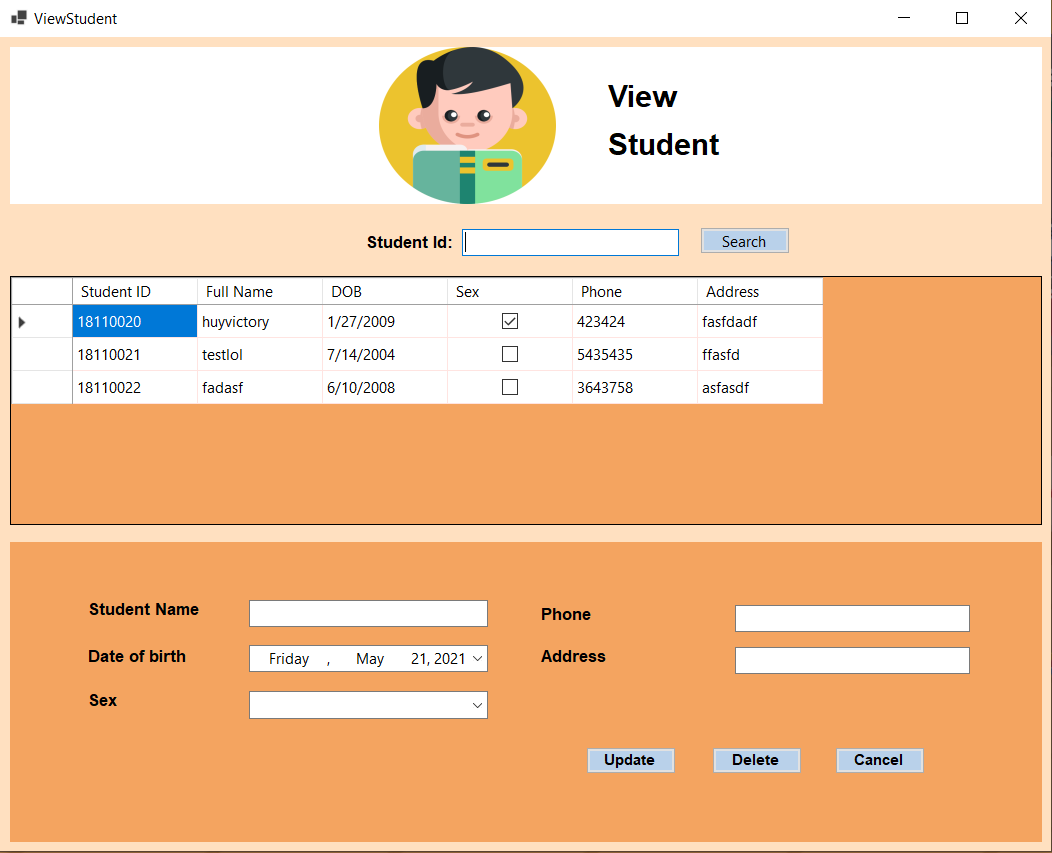


Figure 3.5. View Student Form

1. **Add Student Form**

Form used to add student to the system. And the student who has been added will be added to the list.

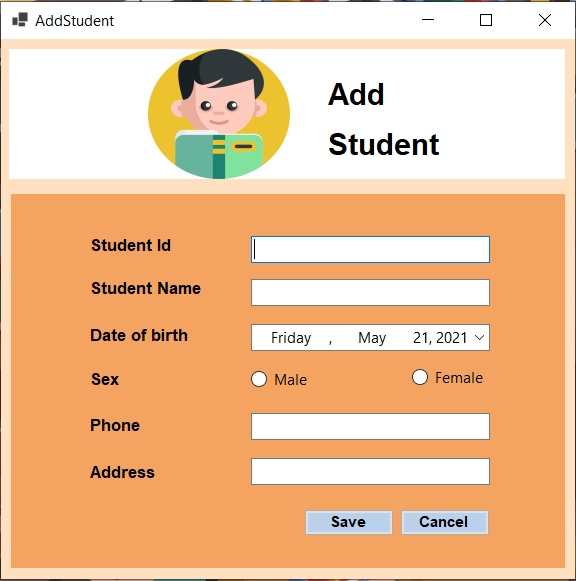


Figure 3.6..Add Student Form

1. **View Teacher Form**

The form shows us a list of teachers who have borrowed books, and can also search for teachers. You can also view, edit and delete the details of each teacher.

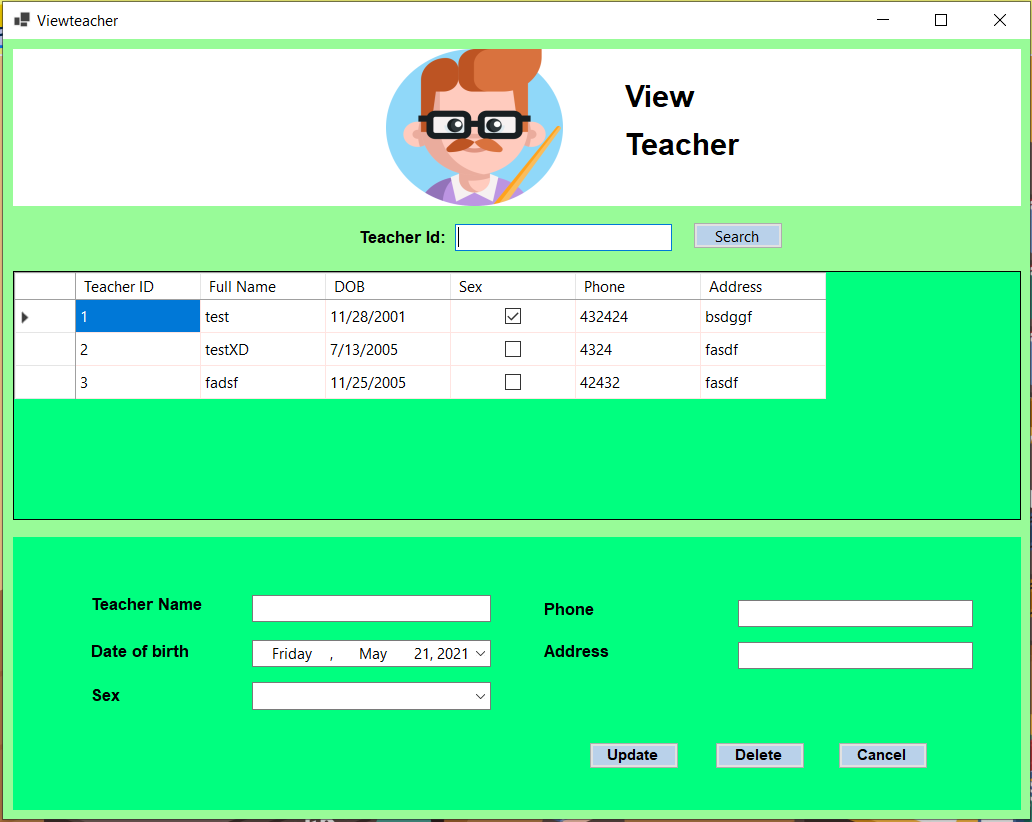


Figure 3.7. View Teacher Form

1. **Add Teacher Form**

Form used to add teacher to the system. And the teacher who has been added will be added to the list.



Figure 3.8. Add Teacher Form

1. **Lending Book Form**

The form displays the personal information of the lecturer who is borrowing the book and the types of books and information about the books that the lecturer has borrowed. You can search for teachers and borrowing books.

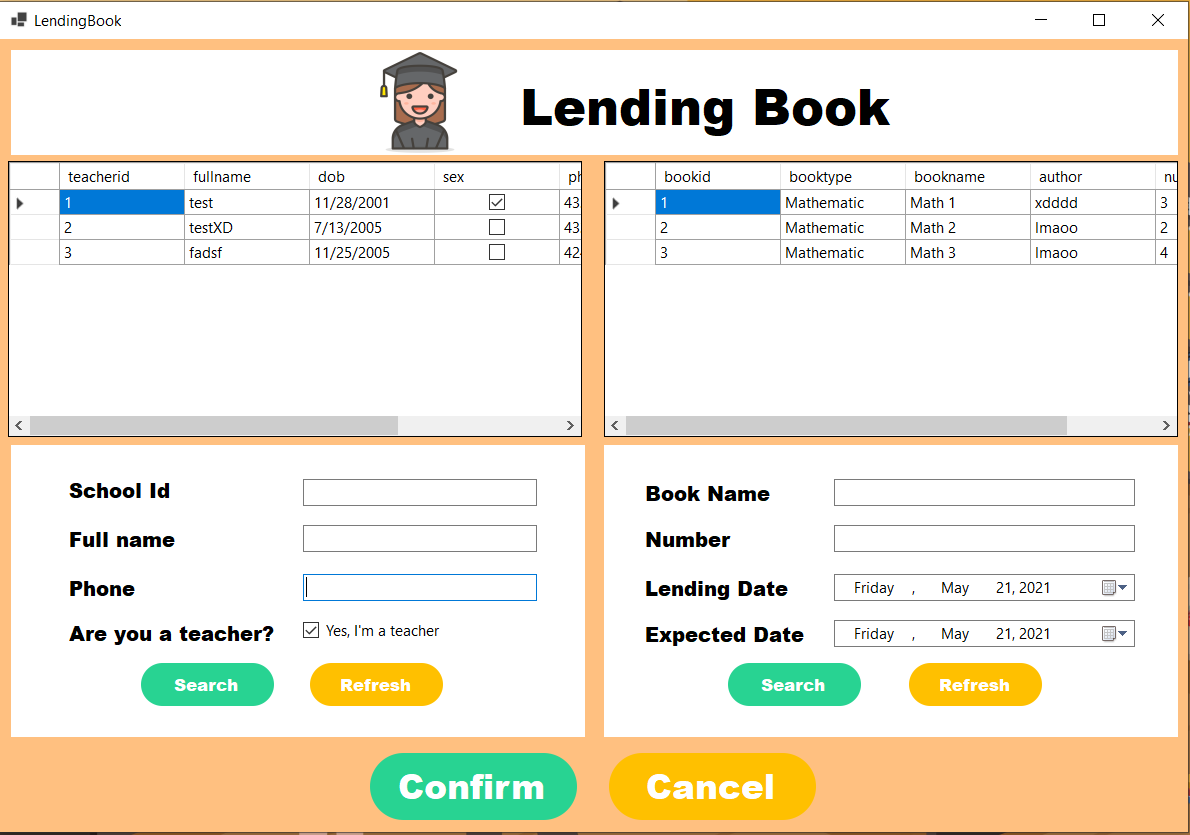


Figure 3.9. Lending Book Form

1. **Returning Book Form**

The form displays the code, the name of the lecturer and the student who borrowed the book. Along with that are the books that the lecturers and students have borrowed, the same period of borrowing and the current return. Finally, confirm the return of the book.

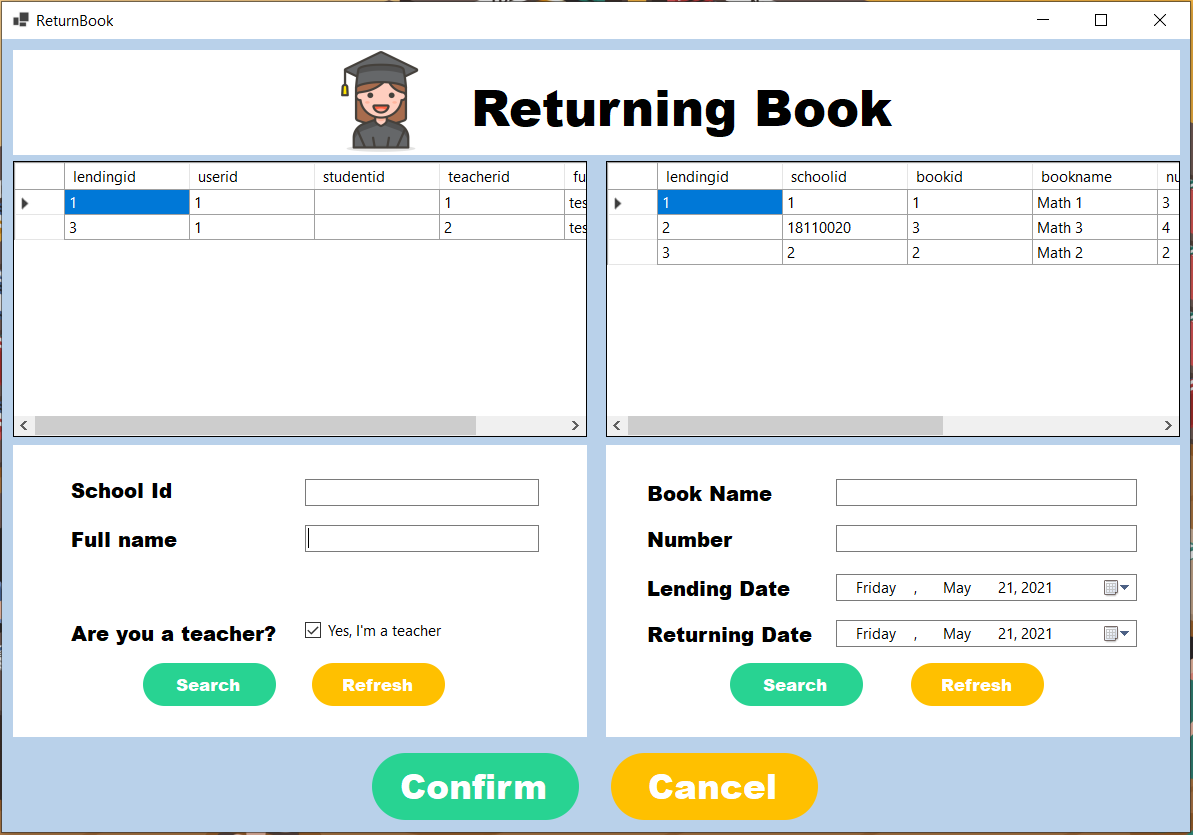


Figure 3.10. Returning Book Form

1. **Update Information Form**

Form used to edit account’s information that sign in to the system such as username, password, name, gender, date of birth, phone, address.

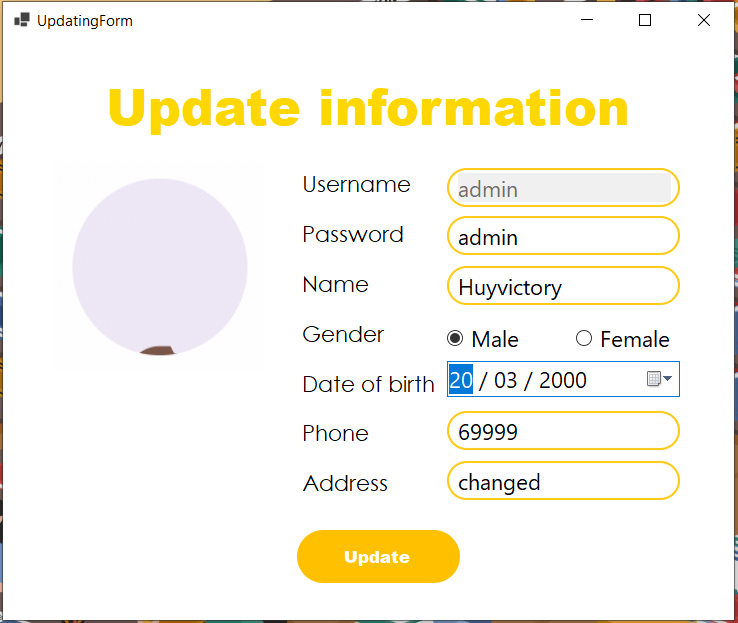


Figure 3.11.. Update information Form

**iV. conClude**

Basically, the team self-reviewed that the group's software had solved most of the requirements set out by the project and also added additional useful features to the program.

**Advantages:**

● Clean, intuitive interface, easy to approach, easy to get used to.

● The capacity is quite light.

● The program consumes very little system resources.

● The program runs stably, is tested many times and gives accurate results, no crashes, debug, except or lag during execution to ensure correct user requirements.

**Disadvantages:**

● The algorithm is not optimal when processing many large data streams (Sometimes processing is slow).

● The data cannot be found when it is lost.

**Toward Development:**

● Add attendance feature by QR code

● Optimized when handling large input data.

● Interface upgrade.

● Add a feature to search for an account when the password is lost

● Add account registration feature.

● Public database on the internet so that it can be accessed from many different machines.

**REFERENCES**

1. Nguyễn Thành Sơ, *Hệ quản trị cơ sở dữ liệu*, Đại học sư phạm kỹ thuật thành phố Hồ Chí Minh.
2. Tô Minh Phong, *Phân biệt mô hình 3 layer và MVC*, viblo.asia, https://viblo.asia/p/phan-biet-mo-hinh-3-layer-va-mvc-m68Z0mP2lkG, [20/11/2020].
3. <https://www.youtube.com/watch?v=4ToxRR-zlKs>