

VIETNAMESE – GERMAN UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER SCIENCE DEPARTMENT

JAVA PROJECT REPORT
LIBRARY MANAGEMENT
SYSTEM

Module 61CSE215: Object Oriented Programming in JAVA

1. Từ Trung Tín – 10421106
2. Cao Thắng Long – 10421089
3. Nguyễn Ngọc Hoàng Nam – 10421042

Lecturer: Dr. Tran Hong Ngoc

Binh Duong, WS2023

Versions

Date	Version	In charge	Description
18-October 23	V1.0	Hoang Nam, Thang Long, Trung Tin	Update Section I, II
20-October 23	V1.1	Thang Long, Trung Tin	Update Section II.2 and III
21-October 23	V1.2	Hoang Nam	Update Section II.2, III
22-October 23	V1.3	Hoang Nam Trung Tin	Update Section III
3-November 23	V1.4	Trung Tin	Update Section VI
4-November 23	V1.5	Thang Long, Trung Tin	Update Section III, IV, VI, VII, Appendix A
5-November 23	V1.6	Nam Nguyen	Update Sections III, IV, V
14-November 23	V1.7	Hoang Nam Thang Long Trung Tin	Update and complete the entire report

Abbreviation

LMS	Library Management System
IBDetails	Issue Book Details

List of Figures

Figure 1. Class Diagram of Library.....	10
---	----

List of Tables

Table 1: List of Objects.....	7
Table 2: Functions and Methods of Classes.....	8
Table 3: Details of Classes.....	11
Table X1: Data Access Control Table.....	21
Table X2: Method Access Control Table.....	25

Table of Contents

I. INTRODUCTION.....	6
II. CLASS ANALYSIS.....	7
III. CLASS DESIGN.....	9
IV. PACKAGE DESIGN.....	21
V.INTERFACE DESIGN.....	21
VI.ACCESS CONTROL.....	21
VII. ENCAPSULATION VS INHERITANCE VS POLYMORPHISM.....	29
VIII. EXPERIMENT.....	23
IX. CONCLUSION.....	28

I. INTRODUCTION

A library is a collection of sources of information and similar resources, made accessible to a defined community for reference or borrowing. Thus, the process of handling a library manually is very troublesome and clumsy. As regards this point of view, we plan to develop a computerized system for handling the activities of library management in a comprehensive way to lessen physical labor and reduce the complexity of the manual system.

This project is being developed to help the staff of the library to maintain the library in the best way possible and also reduce human efforts. There are several basic functions included in this project for the staff as follows:

Input Functions:

1. Add New Book: This function allows librarians to input information about newly acquired books, including title, author, ISBN, publication date, and location within the library.
2. User Registration: Enables librarians to input user information, including name, contact details, and library card number, for new members or to update existing user records.
3. Book Update: Library staff can use this function to input changes, additions, or deletions to the library books, including metadata updates for existing items.
4. Reserve or Hold Materials: Users can place a hold or reserve on a book that is currently checked out, and they will be notified when it becomes available.
5. Renew Materials: Users can renew borrowed materials online, extending the due date if no one else has requested the item.

Search Functions:

1. Keyword Search: Allows users to search the library catalog for books, articles, or other materials based on keywords, titles, authors, or subject categories.
2. User Account Lookup: Enables librarians to search for and retrieve user account information using library card numbers or names.

Output Functions:

1. Book Availability: Displays a specific book's availability status and location, allowing users to know whether a book is checked out or available for loan.

2. Generate Reports: Provides librarians with the capability to generate reports on various aspects of library management, such as circulation statistics, overdue items, or the most popular books.
3. Inventory Management: Helps library staff perform regular inventory checks and updates to ensure the accuracy of the catalog..

These functions assist in the efficient management and functioning of a library by providing users with access to and discovery of resources. Librarians may maintain the collection, help patrons, and provide reports for analysis and decision-making all at the same time.

II. CLASS ANALYSIS

1. Objects

No.	Object Name	State	Behaviors	Class
1	Student A	7682, Nguyen Thi A, abc12345, 106 Vo Thi Sau HCM, 07	isStudying	Student
2	Student B	2537, Cao Van B, 57 Nguyen Thi Minh Khai, 12	isStudying	Student
3	Student C	8745, Bui Van C, 4326, 52 Nguyen Van Tao	isStudying	Student
4	Book A	2311, Mike, Java, 15	isAvailable	Book
5	Book B	4212, Clara, Python, 40	isAvailable	Book

6	Book C	5341, Jacob, C#, 78	isAvailable	Book
---	--------	------------------------	-------------	------

Table1. List of Objects

2. Classes

No	Name	Functions	Method
1	LMS	printInfo() setUserName() setPassword()	-To print out the login information of the user
2	Material	printName()	-To print the name of the book
3	Book	addBook() updateBook() deleteBook() setBookID() setAuthorName setQuantity() setBookName()	-To add books, update and delete books in the managed book table. -To set ID, author name, quantity and name of the book
4	Book Return	returnBook() updateBookCountcontrol()	-To return book to the system -To update the quantity of books
5	Book Issue	getBookDetailscontrol() getStudentDetailscontrol() isAlreadyIssuedcontrol() getBookId() getBookName() getAuthor() getQuantity() getName() getId()	-To get book and student details. -To check book already loaned or not- -To get book ID, name, author, quantity. -To get student name, ID, course and branch.

		getCourse() getBranch()	
6	User	setId() setName() setEmail() setPhoneNo()	-To set student ID, name, email and phone number.
7	Student	setCourse() setBranch() addStudent() updateStudent() deleteStudent() isStudent()	-To set a course, branch for students. -To add, update and delete students. -To check if this is a student ID.
8	IBDetails	getIssueBookDetailscontrol(bookId, studentId)	-To get issue book details.

Table 2: Functions and Methods of Classes

III. CLASS DESIGN

1. Classes

Figure 2. Class Diagram of Library

No	Class	Instance Variable	Methods	Description
1	Book	public String author; public int bookId; public int quantity;	1. boolean addBook() 2. boolean updateBook() 3. boolean deleteBook() 4. void SetBookId(int id) 5. void setAuthorName(String an) 6. void setQuantity(int q)	This class is used to manage book and save data of books in the system

			7. void setBookName(String bn)	
2	Book Issue(extend of LMS abstract class)	private Book b; private Student s;	BookIssue getBookDetailscontrol(int bookId) BookIssue getStudentDetailscontrol(int studentId) boolean isAlreadyIssuedcontrol(int bookId, int studentId) int getBookId() String getBookName() String getAuthor() int getQuantity() String getName() int getId() String getCourse() String getBranch()	This class is used to manage loan stuff of the library system
3	Student(extend of User class)	public String course; public String branch;	void setCourse(String c) void setBranch(String b) boolean addStudent() boolean updateStudent() boolean deleteStudent() boolean isStudent(int userId)	This class is used to set information about student and used to manage registered students in the system.

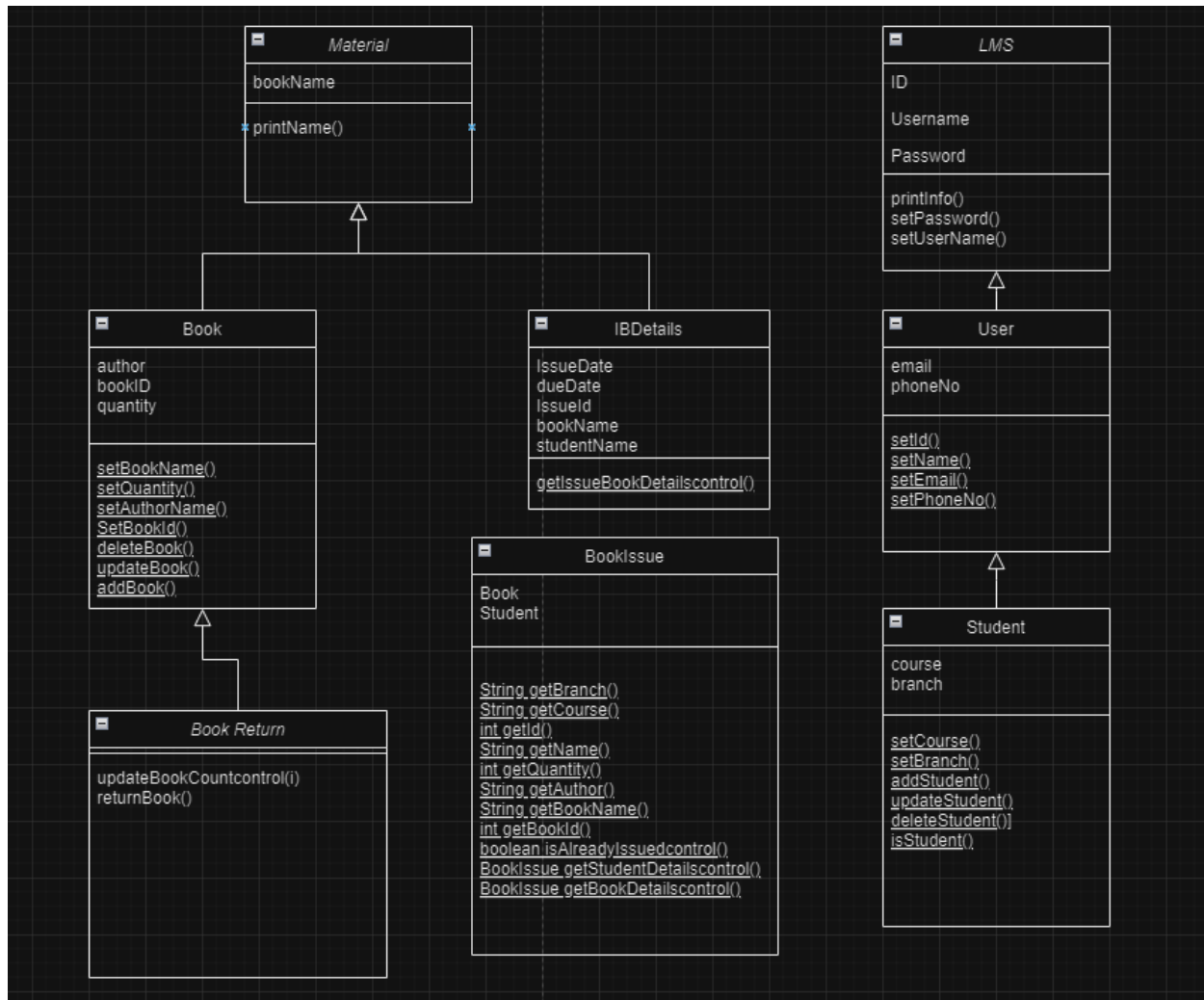
4	User(extend of LMS abstract class)	private String email; private String phoneNo;	void setId(int id) void setName(String n) void setEmail(String e) void setPhoneNo(String p)	This class is used to set information about user
5	IBDetails(extend of LMS abstract class)	Date IssueDate, dueDate; int IssueId; private String bookName; private String studentName;	getIssueBookDetailscontrol(int bookId, int studentId)	This class is used to manage all the details about book issues.
6	BookReturn(extend of Book class)		boolean returnBook(int studentId, int bookId) int updateBookCountcontrol(int bookId)	This class is used to manage return book

Table 3: Details of Classes

Abstract classes

No	Abstract Class	Instance Variable	Abstract Methods	Description
1	LMS	String id, username, password	printInfo()	This class is an abstract class that is used by subclass User. (See Figure 2.)
2	Material	String bookname	printName()	This class is an abstract class that is used by subclass Book, IBDetails. (See Figure 2.)

CLASS DIAGRAM



2. Some OOP techniques

2.1. Overloading method:

Constructor overloading in User, Student, Book, Book Issue, IBDetails class.

- **BookIssue()** constructor:

```
public BookIssue(String n, String a, int i, int q) {  
    b = new Book(n, a, i, q);  
}  
  
public BookIssue(String n, int i, String c, String b) {  
    s = new Student(n, i, c, b );  
}  
  
public BookIssue() {}
```

- IBDetails() constructor:

```
public IBDetails(int issueId, String bookName, String studentN  
    this.issueId = issueId;  
    this.bookName = bookName;  
    this.studentName = studentName;  
    this.issueDate = issueDate;  
    this.dueDate = dueDate;  
}  
  
public IBDetails()  
{  
    this.issueId = 0;  
    this.bookName = "";  
    this.studentName = "";  
    this.issueDate = null;  
    this.dueDate = null;  
}
```

- Student() constructor:

```
public Student() {  
    super();  
    this.course = "";  
    this.branch = "";  
}  
  
public Student(String n, int i, String c, String b) {  
    this.name = n;  
    this.id = i;  
    this.course = c;  
    this.branch = b;  
}
```

Figure 4: Student() constructor:

- User() constructor:

```
public User(String n, int i, String pwd, String e, String p) {  
    this.name = n;  
    this.id = i;  
    this.pwd = pwd;  
    this.email = e;  
    this.phoneNo = p;  
}  
  
public User() {  
    this.name = "";  
    this.id = 0;  
    this.pwd = "";  
    this.email = "";  
    this.phoneNo = "";  
}
```

- 3 Book() constructors:

```
public Book(String n, String a, int i, int q){  
    this.bookName = n;  
    this.author = a;  
    this.bookId = i;  
    this.quantity = q;  
}  
  
public Book(){  
    this.bookName = "";  
    this.author = "";  
    this.bookId = 0;  
    this.quantity = 0;  
}  
  
public Book(int i){  
    this.bookId = i;  
}
```

2.2. Overriding method:

In the below example, the User class extends the abstract LMS class, and the printInfo() method is overridden.

```
public class User extends LMS {  
  
    @Override  
    public void printInfo() {  
        super.printInfo();  
        System.out.println("Course: " + course);  
        System.out.println("Branch: " + branch);  
    }  
}
```

3. Inheritance

In this project, we use one kind of inheritance-related technique which is multilevel inheritance. Here are the codes to prove it:

```
public abstract class LMS {  
  
    public class User extends LMS {  
  
        public class Student extends User
```

In this code, we can see that Student inherits User which inherits the LMS abstract class which is a multilevel inheritance.

IV. Package Design

There are 3 different packages we use in this project which are Database, LMS, and GUI (more detailed in Appendix A)

- + The Database package contains all classes used to manage the data. They are called by the method to receive data from the corresponding class in the LMS package.
- + The LMS package contains all classes used to control the objects, receive the data from the GUI classes, and transfer the data to the corresponding Database classes by corresponding methods.
- + The GUI package contains all classes used to create the interface for users to interact with the system and send the data typed by users to the corresponding LMS class.

V. Interface Design

- The BookInterface:


```
public interface BookInterface {  
    // Method to add a book  
    boolean addBook(String bookName, String author, int bookId, int quantity);  
  
    // Method to update book information  
    boolean updateBook(String bookName, String author, int bookId, int quantity);  
  
    // Method to delete a book  
    boolean deleteBook(int bookId);  
  
    // Setter method for bookId  
    void setBookId(int bookId);  
  
    // Setter method for author  
    void setAuthorName(String author);  
  
    // Setter method for quantity  
    void setQuantity(int quantity);  
  
    // Setter method for bookName  
    void setBookName(String bookName);  
}
```

```
public class Book implements BookInterface {
```

- The StudentInterface:

```
public interface StudentInterface {  
    boolean addStudent(int id, String name, String course, String branch);  
  
    // Method to update student information  
    boolean updateStudent(int id, String name, String course, String branch);  
  
    // Method to delete a student  
    boolean deleteStudent(int id);  
  
    // Setter method for course  
    void setCourse(String course);  
  
    // Setter method for branch  
    void setBranch(String branch);  
}
```

```
public class Student extends User implements StudentInterface {
```

VI. Access Control

No	Data	Class	Modifier	Description
1	username, password, ID, name	LMS	public	● Can be accessed from outside the class where it is defined
2	bookName, author, bookId, quantity	Book	public	● Can be accessed from outside the class where it is defined
3	inherited from Book and Student	Book Issue	private	<ul style="list-style-type: none">● Book and Student of Book Issue can not be directly accessible from outside the class where it is defined. Only methods and members within the same class can access and modify this data.● Book and Student is sensitive and not intended for direct external access. This information is part of a library management system where confidential details about departments, including desk numbers, are managed.
4	IssueDate, dueDate, IssueId, bookName, studentName	IBDetails	private	● The issue date, due date, issue ID, name of the book, and name of the student of IBDetails can not be directly accessible from outside the class where it is

				<p>defined. Only methods and members within the same class can access and modify this data.</p> <ul style="list-style-type: none"> • The issue date, due date, issue ID, name of the book, and name of the student of IBDetails data are sensitive and not intended for direct external access. This information is part of a library management system where confidential details about departments, including desk numbers, are managed.
5	email, phoneNo	User	private	<ul style="list-style-type: none"> • The email and phone number of the User can not be directly accessible from outside the class where it is defined. Only methods and members within the same class can access and modify this data. • The email and phone number of User is sensitive and not intended for direct external access. This information is part of a library management system where confidential details about departments,

				including desk numbers, are managed.
6	course, branch	Student	public	• Can be accessed from outside the class where it is defined

Table X1: Data Access Control Table

No	Method	Type	Class	Modifier	Description
1	printInfo()	void	LMS	Public	To print all the username, password and id of the users.
2	addBook()	boolean	Book	Public	Adds a new book to the Book resource.
3	updateBook()	boolean	Book	Public	Updates information for an existing book.
4	deleteBook()	boolean	Book	Public	Deletes a book from the existing resource.
5	SetBookId()	void	Book	Public	Sets the ID for a given book.
6	setAuthorName()	void	Book	Public	Sets the author's name for a given book.
7	setQuantity()	void	Book	Public	Sets the number of available copies for a given book.
8	setBookName()	void	Book	Public	Sets the name of a given book.
9	getBookDetailscontrol()		Book Issue	Public	Retrieves details about books.
10	getStudentDetailscontrol()		Book Issue	Public	Retrieves details about students.

11	isAlreadyIssuedcontrol()	boolean	Book Issue	Public	Checks if a book has already been issued.
12	getBookId()	int	Book Issue	Public	Retrieves the ID of a book associated with a book issue.
13	getBookName()	String	Book Issue	Public	Retrieves the name of a book associated with a book issue.
14	getAuthor()	String	Book Issue	Public	Retrieves the author's name of a book associated with a book issue.
15	getQuantity()	int	Book Issue	Public	Retrieves the number of copies associated with a book issue.
16	getName()	String	Book Issue	Public	Retrieves the name of the student associated with a book issue.
17	getId()	int	Book Issue	Public	Retrieves the ID of the student associated with a book issue.
18	getCourse()	String	Book Issue	Public	Retrieves the course of the student associated with a book issue.
19	getBranch()	String	Book Issue	Public	Retrieves the branch of the student associated with a book issue.
20	getIssueBookDetailscontrol()		IBDetails	Public	Retrieves details related to issued books.
21	setId()	int	User	Public	Sets the ID for a given user.
21	setStudentName()	String	User	Public	Sets the name of a

					given student.
22	setCourse()	String	User	Public	Sets the course of a given student.
23	setBranch()	String	User	Public	Sets the branch of a given student.
24	addStudent()	boolean	User	Public	Adds a new student to the system.
25	updateStudent()	boolean	User	Public	Updates information for an existing student.
26	deleteStudent()	boolean	User	Public	Deletes a student from the system.
27	returnBook()	boolean	BookR eturn	Public	Handles the return of a book.
28	updateBookCountcontrol()	boolean	BookR eturn	Public	Update the copies quantity of a book

Table X2: Method Access Control Table

VII. Encapsulation vs Inheritance vs Polymorphism

1. Encapsulation

```

public class User {
    public String Name;
    public int Id;
    public String course;
    public String branch;

    private GUI.ManageStudents ms;
    private Database.MS_SQL mssql = new Database.MS_SQL();
    public User(String n, int i, String c, String b){
        this.Name = n;
        this.Id = i;
        this.course = c;
        this.branch = b;
    }
}

```

```
public class Student extends User {  
    public Student() {  
        super();  
    }  
  
    public Student(String n, int i, String c, String b) {  
        super( n, i, c, b);  
    }  
}
```

In this code Student class inherits the User class and since **encapsulation** is a primary attribute of OOP whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super.

-The keyword super in this code is used to call to superclass' constructor which is class User("super(n, i, c, b);")

2. Inheritance

In this project, we use one kind of inheritance-related technique which is multilevel inheritance. Here are the codes to prove it:

```
public abstract class LMS {  
    public class User extends LMS {  
        public class Student extends User
```

In this code, we can see that Student inherits User which inherits the LMS abstract class which is a multilevel inheritance.

3. Polymorphism

In the below example, the User class extends the LMS class, and the printInfo() method is overridden, this represents **run-time polymorphism** in Java.

```
public void printInfo()
{
    System.out.println("\nThe details are: \n");
    System.out.println("ID: " + id);
    System.out.println("Password: " + pwd);
};
}

public class User extends LMS {

@Override
public void printInfo() {
    super.printInfo();
    System.out.println("Course: " + course);
    System.out.println("Branch: " + branch);
}
}
```

VIII. Experiment

1. Environment and Tools

- a. **Environment:** We use 3 laptops for the project: 2 Lenovo Legion 5 and 1 Acer Nitro 16
RAM: 16gb
CPU: Ryzen 5, 7 over 5000 series
- b. **Tools:**
Libraries:
.JAR files: RSTableMetro.jar, RSFoto_v1.0.jar, RSCalendar.jar, RojeruSan.part1.jar,

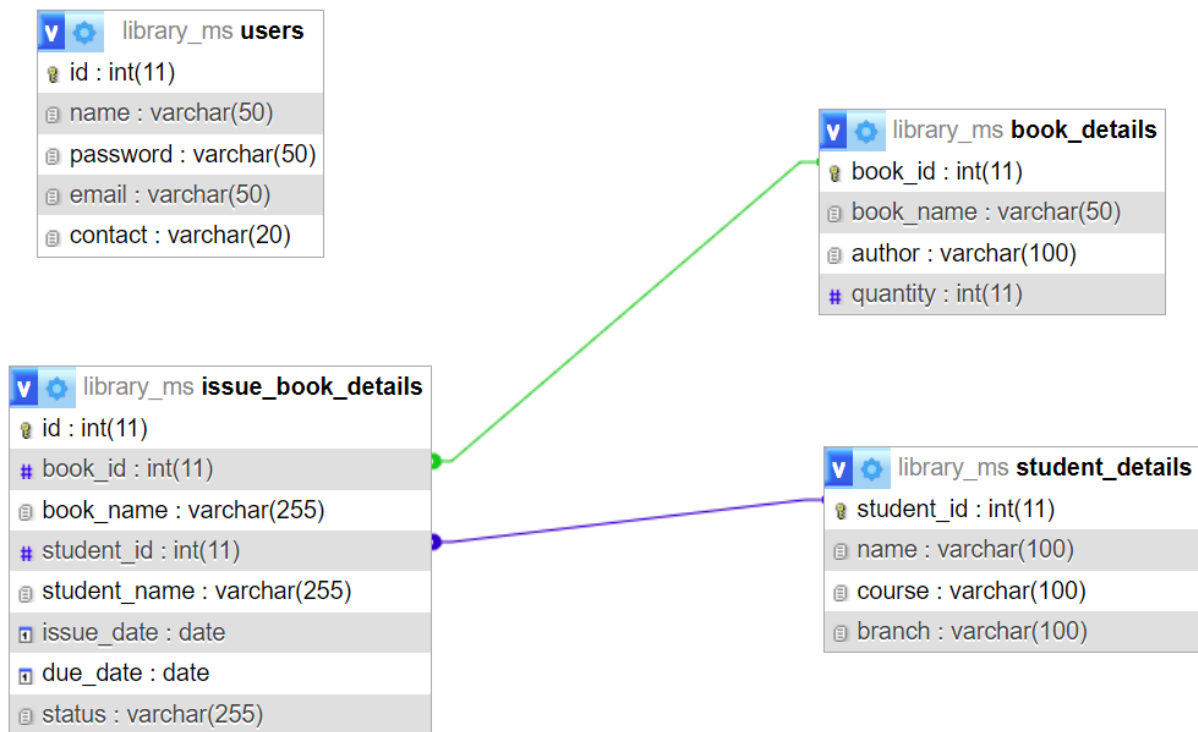
c.

2. Project functions

No	Name of the function	Describe
1	Login function	To login to the system
2	Sign up function	To sign up an account to use in the system
3	Dashboard in Home Page	To show number of books, students, issued books, defaulters and also show

		detail about books and student that are registered in the system
4	Manage issue book function	To register a book to loaned status and give the information about the person who loaned, loaned date and due date to the system.
5	Issue book detail dashboard	Show all loaned book details
6	Manage book function	To add, update and delete book information in the system
7	Manage student function	To add, update and delete student information in the system.
8	Return book function	To register a book to available status

3. Database (4 tables)



4. GUI (4 figures)

In the project we have a total of 10 GUIs to present the project and here is the list of the GUIs:

Order	Name
1	Sign up Page
2	Login Page
3	Home Page
4	Issue book Page
5	Issue book details Page
6	Return book Page
7	Book management Page
8	Student management Page
9	Defaulter list Page
10	View all records Page

- 4 main GUIs in the report:
1. Book management Page

Book ID	Name	Author	Quantity
220	Another Test	Bilal Sol	9
1402	Essential Calculus	James Stewart	2
2314	Test	Paulie The Platypus	124
3154	An Introduction To Formal L...	Peter Linz	3
3950	Linear Algebra And Applicatio...	Gilbert Strang	4
5432	Introduction To Algorithms	Thomas Cormen, Charles Leis...	1
7954	Probability And Statistics	Anthony Hayter	2

In this GUI, we design to perform 3 actions which are adding books, updating books, deleting books from the database and also showing the book data using the table. Books can be added, updated, and deleted by using 3 buttons ADD, UPDATE, and DELETE. For the add function, the admin must enter an ID for the book so it can separate from the others and fill out the remaining details which are book name, author name of the book, and quantity of books, and then press the ADD button. For the update function, the admin has to click on the row that needs to be altered in the table shown in the right panel and then simply enter the new information. For example, if the admin wants to change the quantity of Book ID 220 to 100, first he/ she simply click on the row that has the ID 220 in the table, then all the value of that book will automatically fill in the blanks in the left panel, next set the value of the quantity to 100, and finally press the UPDATE button. To delete a book from the database, similar to updating, the admin has to click on the row that needs to be deleted and then click on the DELETE button.

2. Issue Book Page

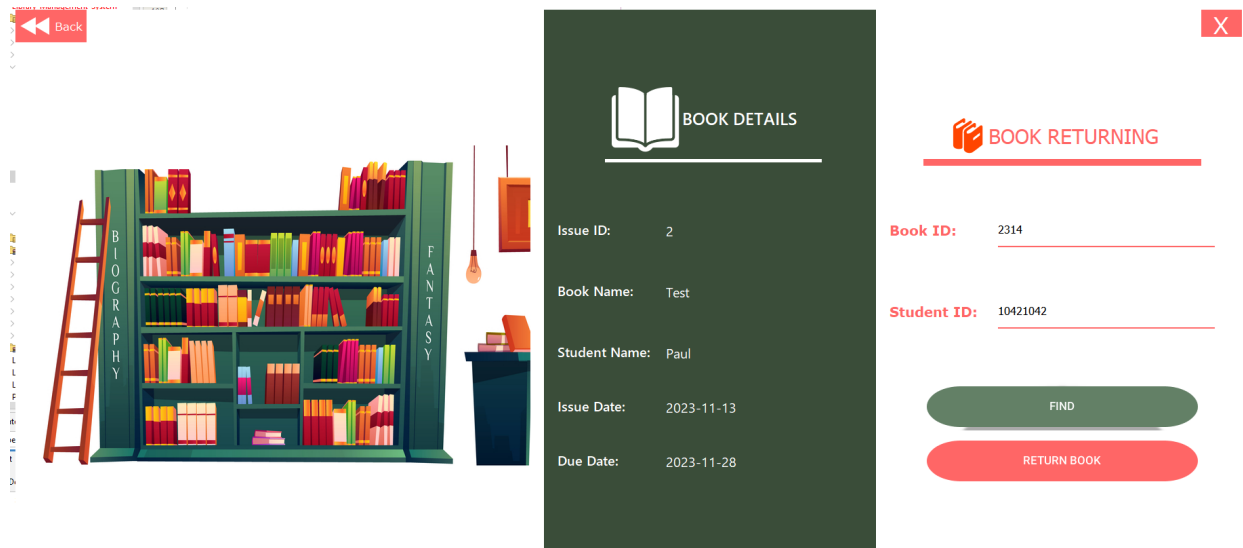
The screenshot displays the 'Issue Book Page' with three main panels:

- BOOK DETAILS (Left Panel):** Contains a 'Back' button and a book icon. Fields include: Book ID: 220, Book name: Another Test, Author: 8, and Quantity: Bill, Sol.
- STUDENT DETAILS (Middle Panel):** Contains a graduation cap icon. Fields include: Student ID: 10421042, Student name: Paul, Course: PHD, and Branch: CS.
- BOOK ISSUING (Right Panel):** Features a red header with a book icon and the text 'BOOK ISSUING'. It includes input fields for Book ID (220), Student Id (10421042), Issue Date (2023/11/30), and Due Date (2023/11/15), each with a calendar icon. A large red button at the bottom is labeled 'ISSUE BOOK'.

In this GUI, the admin can choose a book to be issued. For example, a student whose ID is 10421042 wants to borrow a book named “Another Test” has an ID 220, the admin has to enter the book ID and Student ID in the right panel, and all the Book and Student information will be shown in the Book and Student panel respectively right after

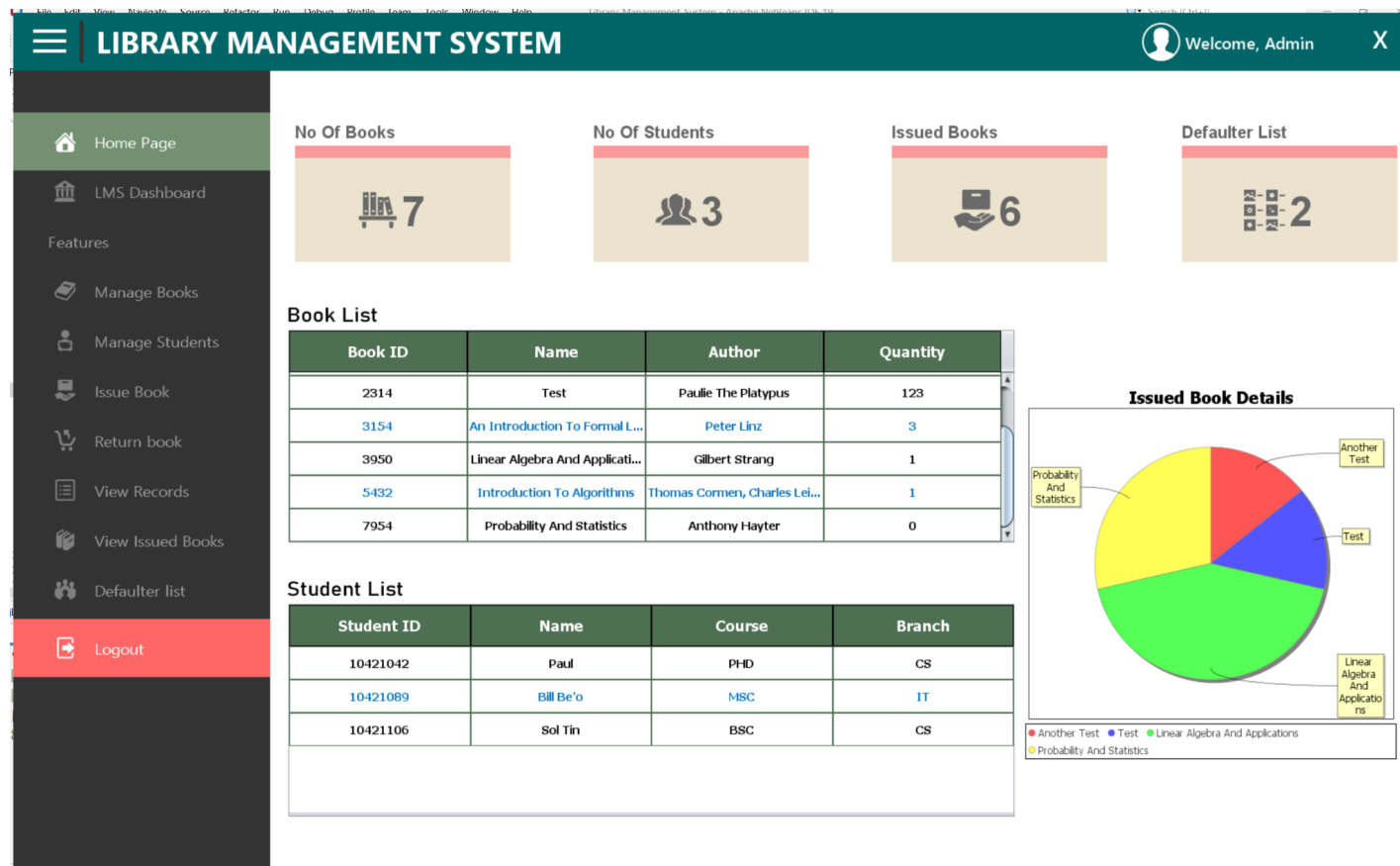
he filled in the 2 ID blanks. Then he simply chooses the Issue and Due Date using the calendar icon next to them before clicking on the **ISSUE BOOK** button. After the book is issued successfully, the quantity will automatically decrease by 1. If the quantity of the book is 0, the system will prevent the admin from issuing the book to the student.

3. Book Returning Page



In this GUI, we designed for the admin to manage the book returning. Whenever a student returns the book, the admin just needs to have the student ID and Book ID to set the book to available status and complete the loan for the student. For example, a student named Paul whose student ID is 10421042 returns the “Text” book whose ID is 2, the admin needs to enter to ID of the student into the Student ID text box and Book ID into the Book ID textbox and the press the **FIND** button to check if the loan is still not on due date so the admin just need to press to **RETURN BOOK** to complete the loan and if it is over the due date the student must pay the fee for returning late.

4. Home Page (Dashboard Page)



In this GUI, we design for the admin and employees of the library to manage all the things in the library. There are 4 boxes in the top middle of the dashboard showing the current number of books, students, issued books, and defaulters of the library. There are also 2 tables in the middle of the window to show all information about all the books and students that are currently in the library, and a pie chart to show the percentage of each issued book.


On the Features side, there are several navigations that the user can transmit to another page of the system. For example, you want to go to the View Records page to see all the records that have been made throughout this month then press the View Records button. Similar to the others, the names of other navigation buttons are named according to their function. The logout buttons will log out of the system and navigate to the Login tab.

IX. Conclusion

In advantages, the project can serve all minimum requirements for the library management system but still miss some practical function.

In our opinion, the deadline for a project is quite short so that we can not manage to fulfill all requirements and we can not finish the project perfectly. That led to some small issue in the coding section but in general we did our best to complete the project.

DUTY ROSTER

ID	Task	In Charge	Start	End	Note
1	- Design Class - Code Function - Report Section	Từ Trung Tín Nguyễn Ngọc Hoàng Nam Cao Thăng Long	18/10/2023	14/11/2023	We did tasks together because lack of members 

REFERENCE

<https://github.com/OSSpk/Library-Management-System-JAVA/tree/master>

Appendix

Package Hierarchy

