UNIVERSITI TEKNOLOGI MALAYSIA

OBJECT-ORIENTED PROGRAMMING (SECJ2154)

SEMESTER 1 2023/2024

GROUP PROJECT

LIBRARY MANAGEMENT SYSTEM

AINA NURAIN BINTI MOHD AROFF (B23CS0022)

ANUSHKA ROSHNI SIVAKUMAR (B23CS0026)

DANESH MUTHU KRISNAN (B23CS0034)

MUHAMAD AFIQ IRFAN BIN MOHAMAD IZAM (B23CS0046)

2 SECRH/2 SECBH

SECTION 01

Lecturer:

MADAM LIZAWATI MI YUSUF

15 JANUARY 2024

SECTION A: PROJECT DESCRIPTION

Problem Statement and Project Overview:

Although online libraries have been the new trend, many physical libraries still maintain their libraries in the old-school way. As times change, many of these libraries have shown interest in embracing technology to help assist in everyday tasks such as maintaining the library database as well as keeping track of borrowed books instead of the traditional library cards. One such library is Otter's Library, a traditional library embracing technology.

Mr. Otter has been facing difficulties in keeping track of the books available in the store as well as locating them for any members who want to borrow them. This increases the time spent in assisting each member making his process slow and less efficient. Besides that, Mr. Otter has also been finding it hard to keep track of the books that have been borrowed and books that have not been returned as all his current records are manual and need a long time to be found. Overall his library is struggling to complete simple day-to-day tasks that can be easily overcome with a simple system.

Our team has planned to develop a system named Otter's Library Management System. This system is a simple and efficient Java application that can be used to complete simple day-to-day tasks of the library. This system is designed to be used by all those who work as librarians in the Library. The system mainly focuses on keeping an organized record of books borrowed and books available to increase the efficiency of managing a library and its tasks as well as improving the overall customer experience.

Objective and Scope:

The main objectives of the system are:-

1. To aid in the borrowing process as well as keep track of borrowed books and their return dates
2. To enable librarians and members to search for books and survey the number of copies available
3. To implement a digital database of the books available and to efficiently manage them
4. To ensure the security as well as integrity of the online library database

The scope of this project will mainly focus on providing a simple platform for Otter's Library to manage and facilitate everyday tasks of the library in a much more efficient way. This system can provide information regarding the members, their borrowed books, return dates, and other books available in the library. This system is available as a Java application and does not require an internet connection.

Work Flow:

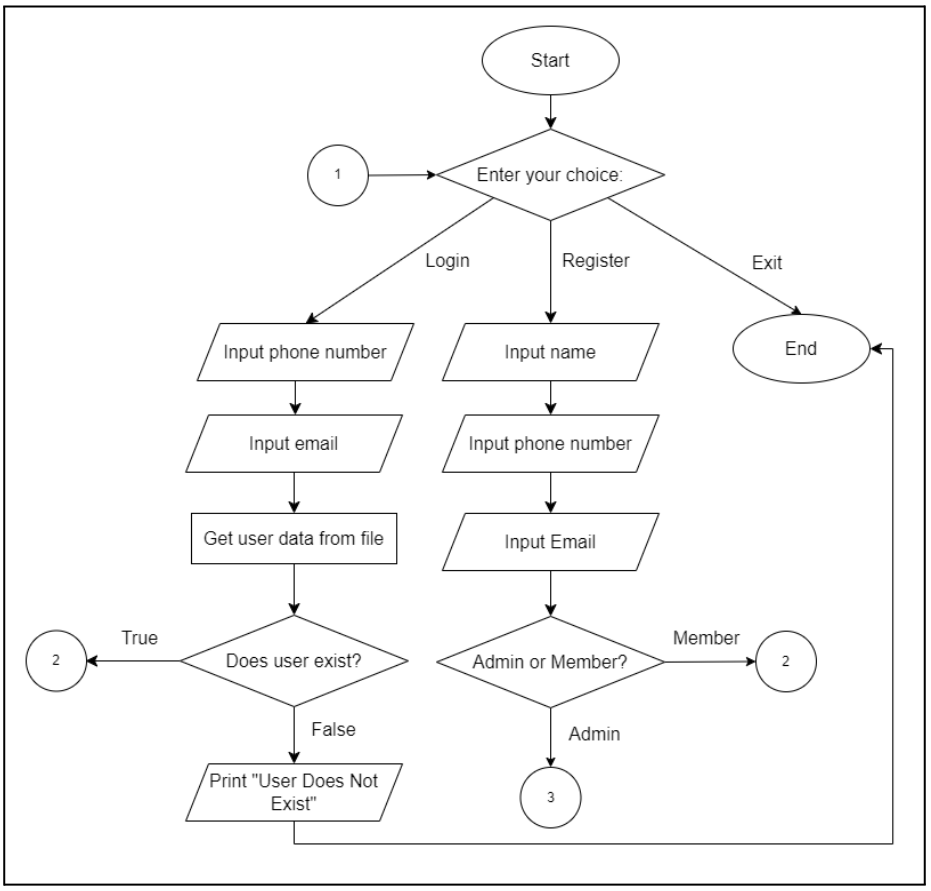


Figure 1.1: Flowchart for Main Menu

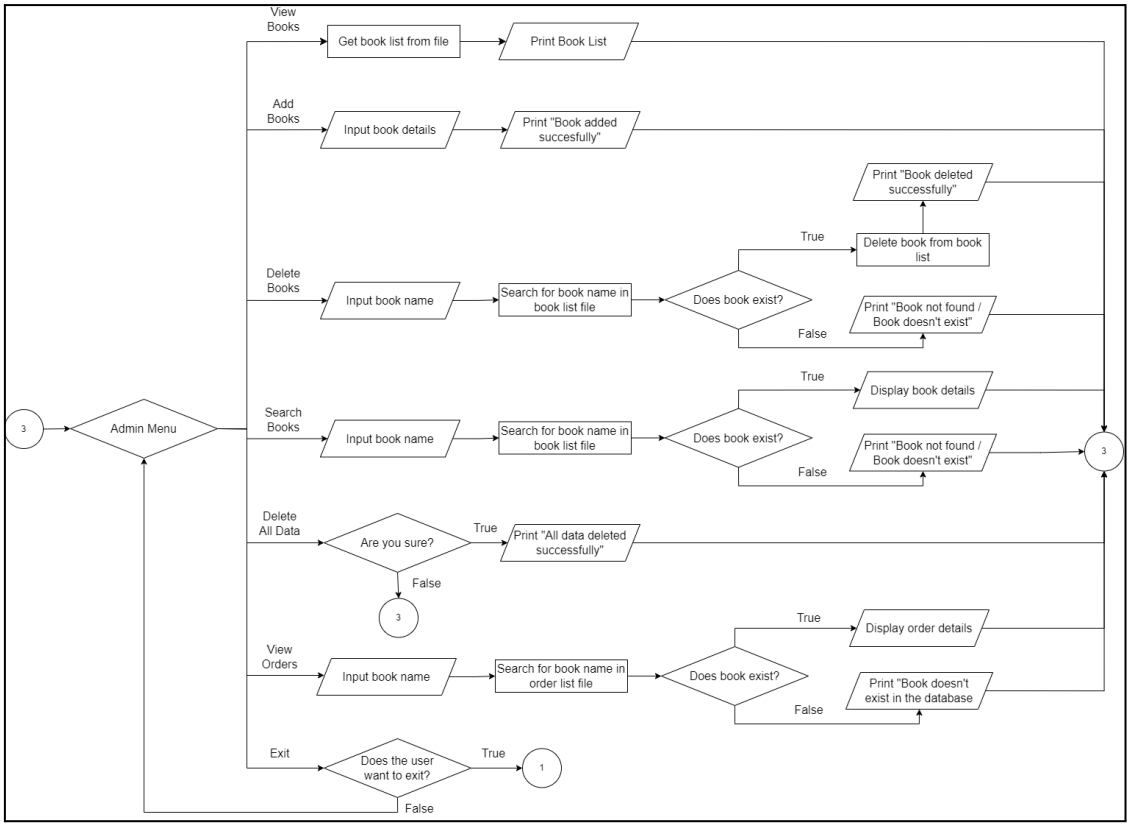


Figure 1.2: Flowchart for Admin

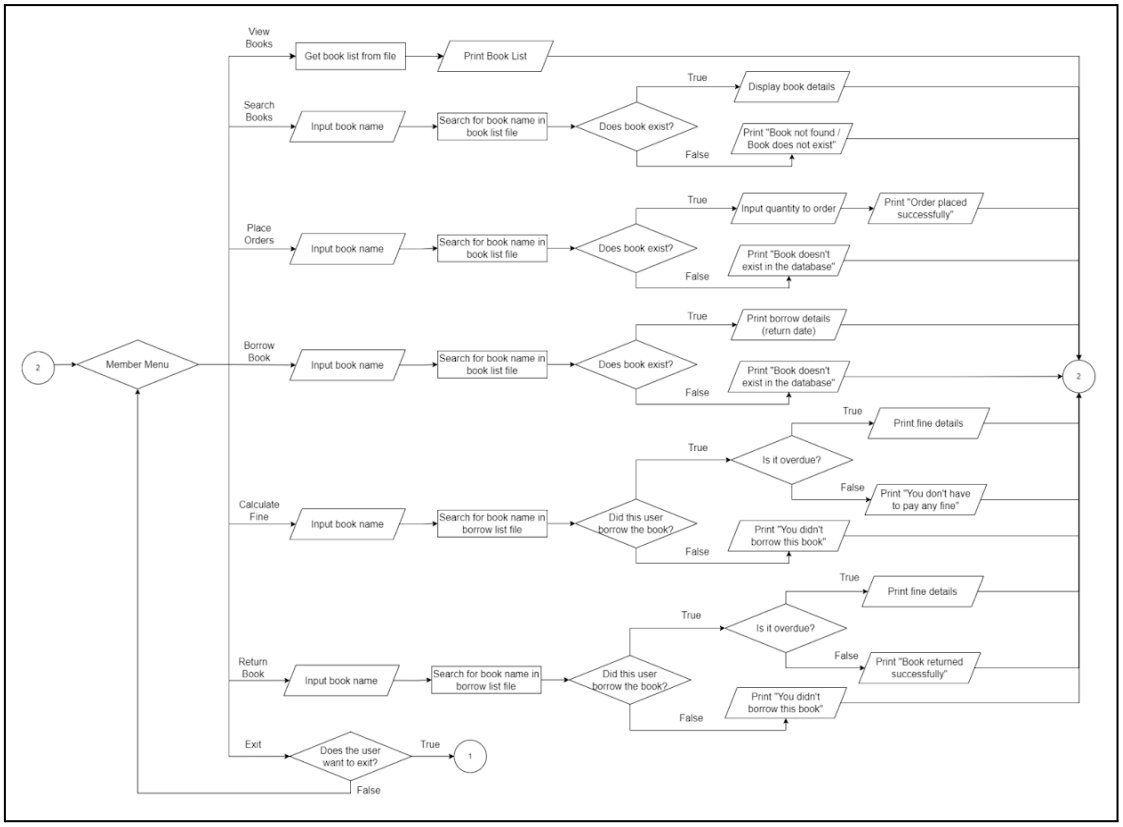


Figure 1.3: Flowchart for Member

OO Concepts:

Encapsulation and Data Hiding

```
20      public Book getBook() {
21          return this.book;
22      }
23
24      public User getUser() {
25          return this.user;
26      }
27
28      public double getPrice() {
29          return this.price;
30      }
31
32      public int getQuantity() {
33          return this.quantity;
34      }
```

Figure 1.4: Extract from Borrowing.java showing Encapsulation

```
package Library;
public class Book {

    private String title;
    private String author;
    private String publisher;
    private String isbn;
    private String status;
    private int qty;
    private double price;
    private int brwCoppies;
```

Figure 1.5: Extract from Book.java showing Data Hiding

```
public class Database {

    private ArrayList<User> users = new ArrayList<User>();
    private ArrayList<String> userName = new ArrayList<String>();
    private ArrayList<Book> books = new ArrayList<Book>();
    private ArrayList<String> bookName = new ArrayList<String>();
    private ArrayList<Order> orders = new ArrayList<Order>();
    private ArrayList<Borrowing> borrowings = new ArrayList<Borrowing>();
```

Figure 1.6: Extract from Book.java showing Data Hiding

Encapsulation and Data Hiding are two of the most prominent key concepts when it comes to Object Oriented Programming. Encapsulation combines both the data and behaviour into one package while also hiding the implementation process from the user. This is especially useful as it maximises reusability. Data Hiding is somewhat similar as it helps in protecting attributes from any accidental corruption which may be caused by modification of other users which may lead to "breaking" the code.

From the extracts, it is shown that we have implemented both Encapsulation, using accessors to retrieve the value of private variables within a class object, and Data Hiding in which we set a few attributes to be private to ensure the safety of a class' member.

Association (Aggregation and Composition)

```
Scanner scanner = new Scanner(System.in);
public void oper(Database database,User user){
    Book book = new Book();
    System.out.println(x:"Enter Book Name");
    String name = scanner.nextLine();

    if(database.getBook(name) > -1){
        System.out.println(x:"Book Already Exist\n");
        user.menu(database, user);
        return;
```

Figure 1.7: Extract from AddBook.java showing Association

```
public void oper(Database database,User user){

    System.out.println(x:"Enter Book You Wanna borrow : ");
    String bookName = s.nextLine();

    int i = database.getBook(bookName);
```

Figure 1.8: Extract from BorrowBook.java showing Association

```java
public void oper(Database database,User user){

    System.out.print(s:"Enter book name to delete : ");
    String bookName = s.nextLine();


    int i = database.getBook(bookName);
    if (i>-1){
        database.deleteBook(i);
        System.out.println(x:"Book deleted successfully\n");
    }else{
        System.out.println(x:"Book not found / Book Doesn't exist\n");
    }
```

Figure 1.9: Extract from DeleteBook.java showing Composition

Association essentially means the general relationship between two classes. In the world of Object Oriented Programming, especially Java, an Association is usually represented as a data field in a class. This allows objects to call methods in other objects. Association also takes the form of two special forms which are Aggregation and Composition where the relationship is viewed in a more detailed manner.

Aggregation is usually described as a data field in the aggregated class. Whereas if the object is exclusively owned by an aggregate object, the bond between the both object and the aggregated object is referred to as Composition.

Inheritance and Polymorphism

```java
public class Members extends User {

    public Members(){};

    public Members(String name){
        super(name);
        this.operations = new IOOperation[]{
            new ViewBook(),
            new Search(),
            new PlaceOrder(),
            new BorrowBook(),
            new CalculateFine(),
            new ReturnBook(),
            new Exit()

        };
    }

    public Members(String name, String email, String phoneNumber){
        super(name, email, phoneNumber);
        this.operations = new IOOperation[]{
            new ViewBook(),
            new Search(),
            new PlaceOrder(),
            new BorrowBook(),
            new CalculateFine(),
            new ReturnBook(),
            new Exit()
```

Figure 1.10: Extract from Members.java showing Inheritance

```java
public class BorrowBook implements IOOperation{
    Scanner s = new Scanner(System.in);
    public void oper(Database database,User user){

        System.out.println(x:"Enter Book You Wanna borrow : ");
        String bookName = s.nextLine();
```

Figure 1.11: Extract from BorrowBook.java showing Polymorphism

Inheritance, usually denoted by ‘extends’ in Java, is used to inherit characteristics of a ‘superclass’ to a ‘subclass’. To be more specific, ‘superclass’ is a general class while ‘subclass’ is a more specialised class. The latter inherits the methods and instance variables of its ‘superclass’. Meanwhile Polymorphism, usually denoted by ‘implements’, is when the same method is used but concludes in different behaviours. This may occur when the same method is invoked on different objects.

Exception Handling

```
public Database(){
    if (!folder.exists()){
        folder.mkdir();
    }
    if (!usersfile.exists()){
        try{
            usersfile.createNewFile();
        }catch (Exception e){}
    }
    if (!bookfile.exists()){
        try{
            bookfile.createNewFile();
        }catch (Exception e){}
    }
    if (!ordersfile.exists()){
        try{
            ordersfile.createNewFile();
        }catch (Exception e){}
    }
    if (!borrowingsfile.exists()){
        try{
            borrowingsfile.createNewFile();
        }catch (Exception e){}
    }
    getUsers();
    getBooks();
    getOrders();
    getBorrowings();
}
```

Figure 1.12: Extract from Database.java showing Exception Handling

Exception Handling simply handles the process of responding to instances that are usually bad or unforeseen. It is used to ensure a system crash does not occur. Shown in the extract of our Database.java, to ensure each one of our text file to store data exists even after a complete system wipe, we used the Exception Handling concept to handle cases in which the required files are not missing and to replace them if the event of one of the files are deleted were to occur.

SECTION B: CLASS DIAGRAMS

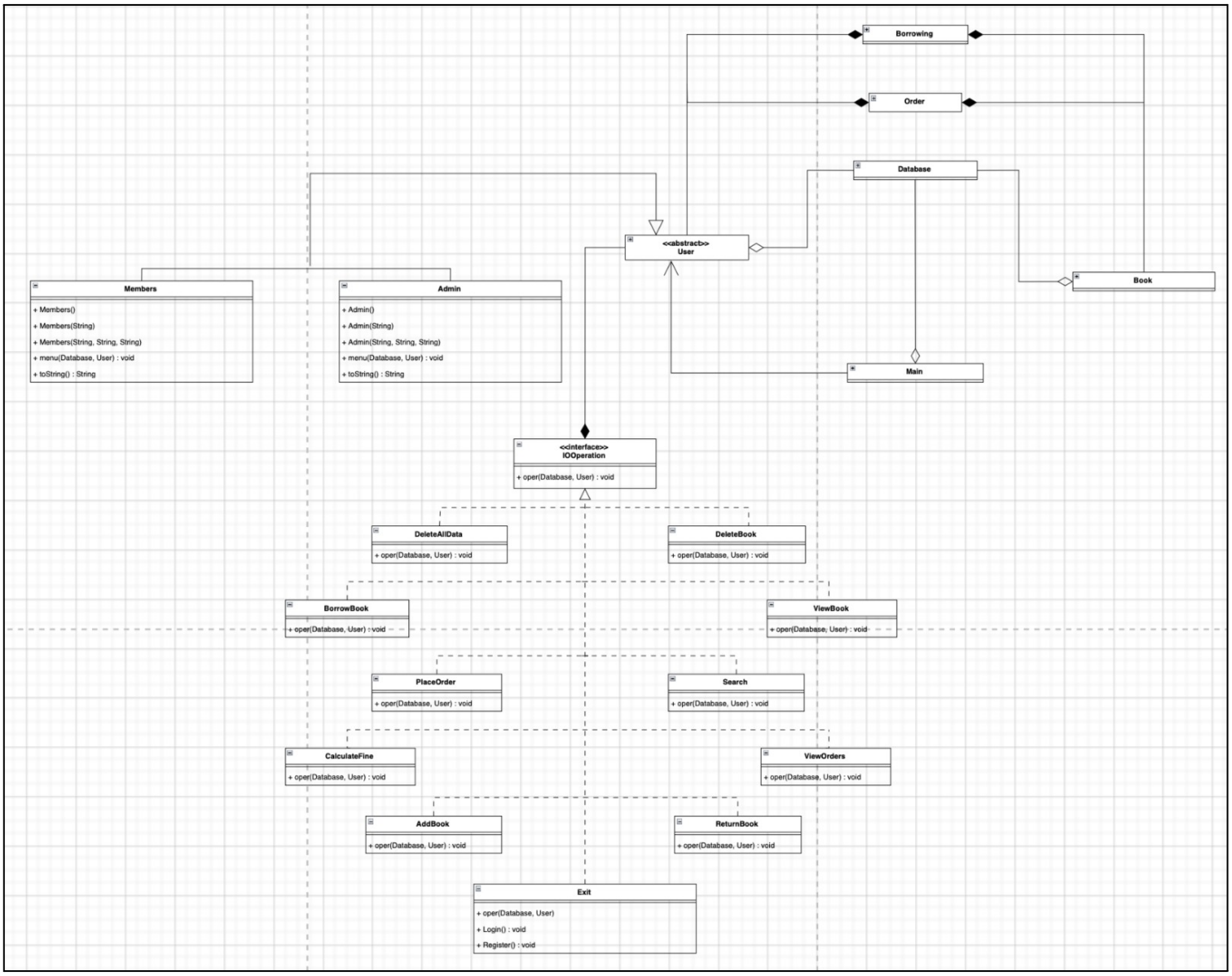


Figure 1.13: Class Diagram of Library Management System

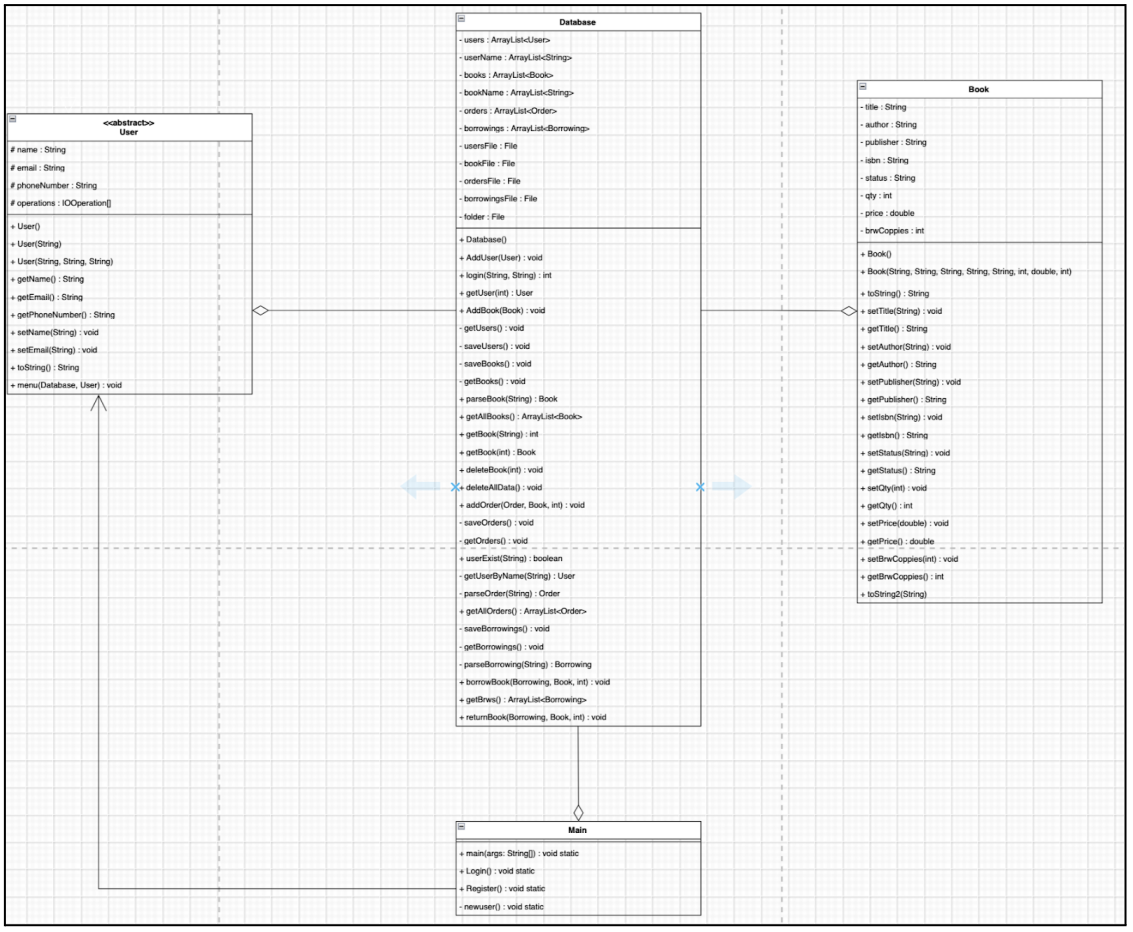


Figure 1.14: The Main Classes derived from the Class Diagram

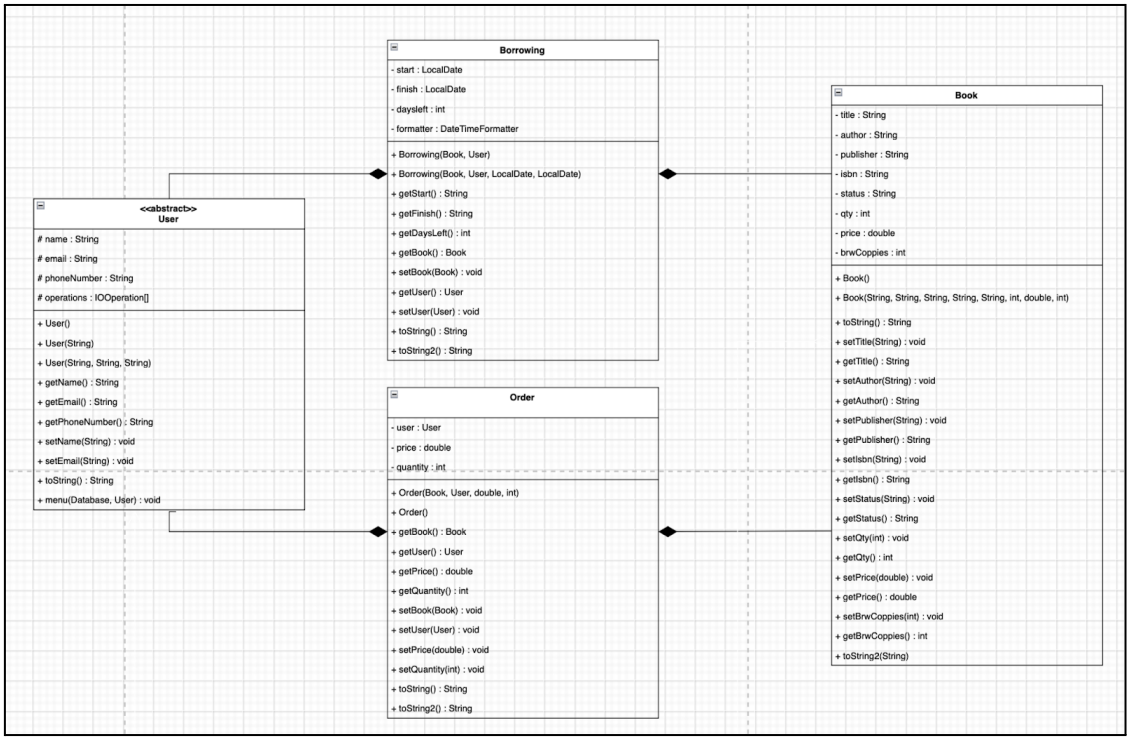


Figure 1.14: The Association Classes derived from the Class Diagram

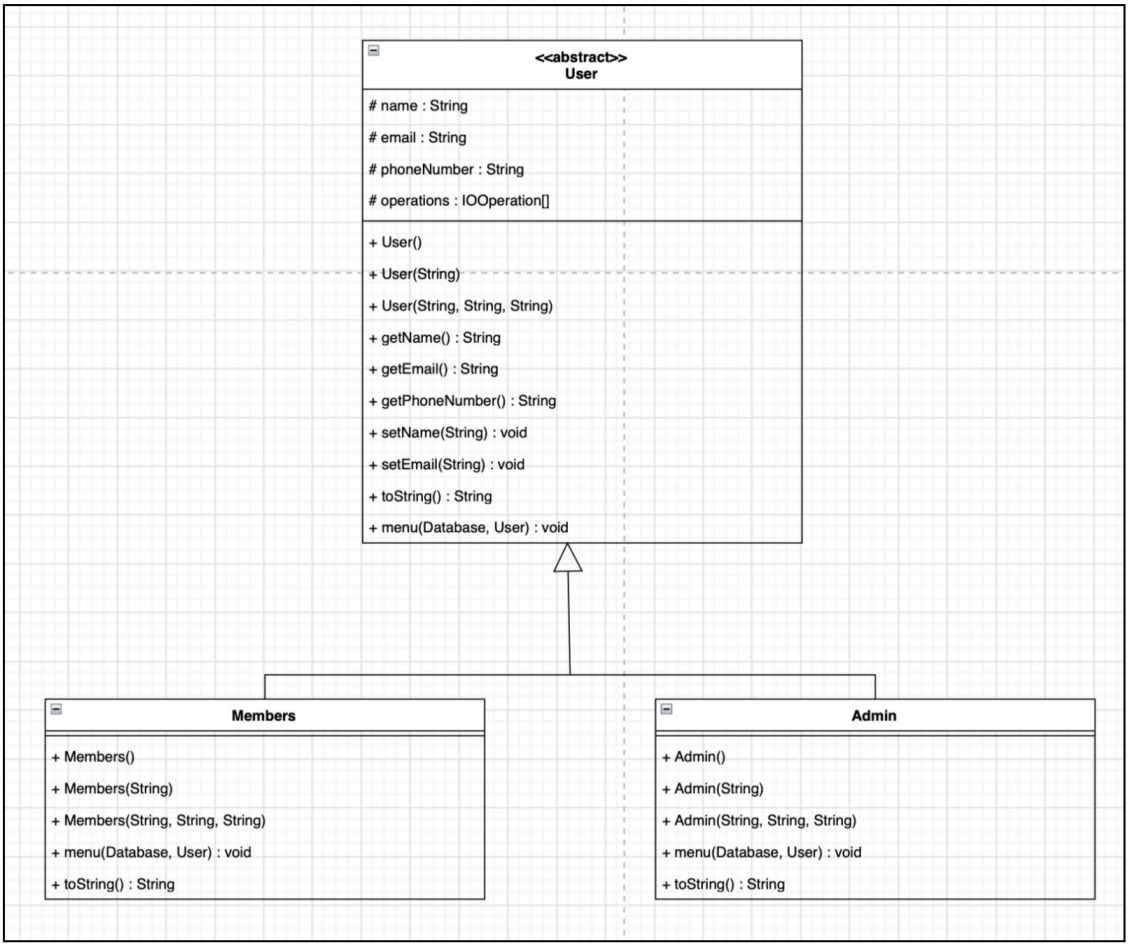


Figure 1.15: The Inheritance Classes derived from the Class Diagram

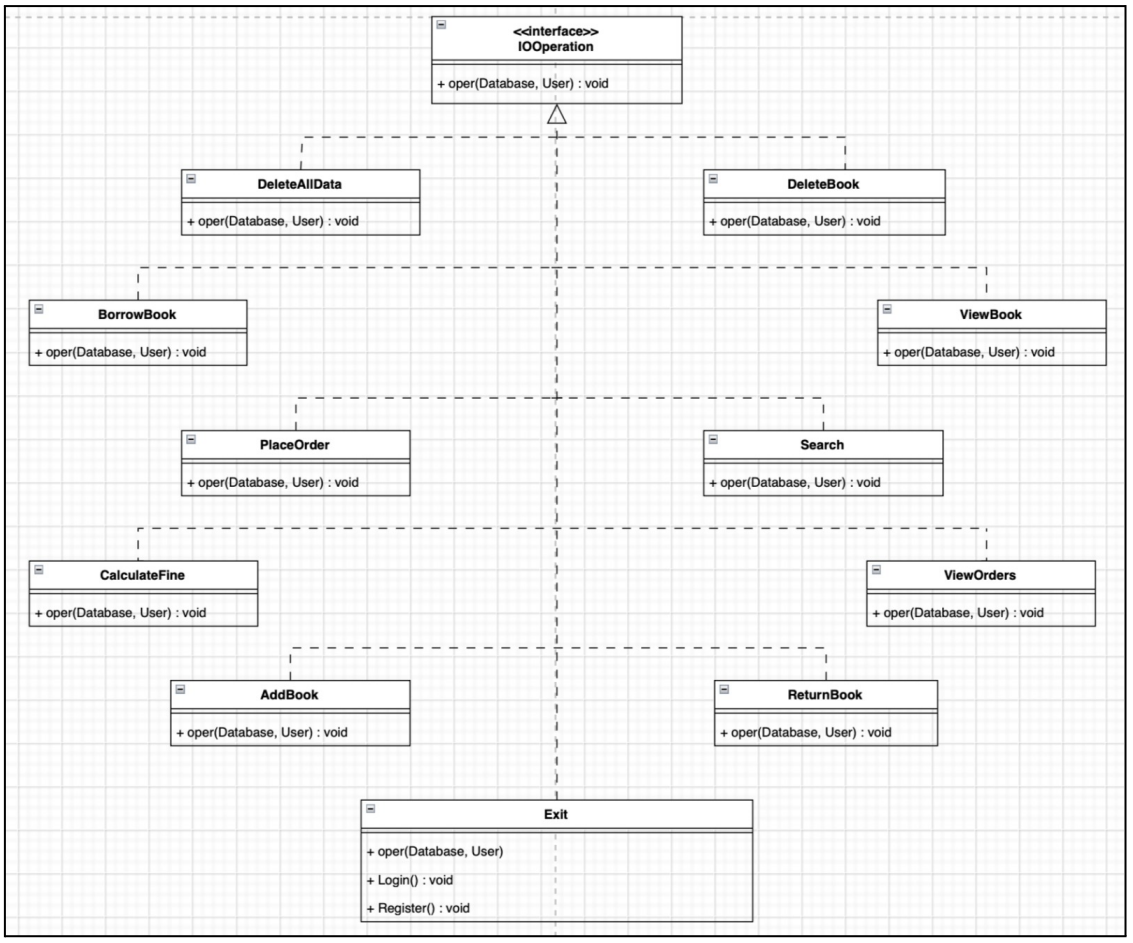


Figure 1.15: The Polymorphism Classes derived from the Class Diagram

Class Main:

Methods	Type	Description
Main (args: String[])	void static	To display menu and get input from user
Login()	void static	To get phoneNumber and email input to login.
Register()	void static	The get registration information input

Class Database:

Attributes	Type	Description
users	ArrayList<User>	Stores User
userName	ArrayList<String>	Username of the user
books	ArrayList<Book>	Stores Book
bookName	ArrayList<String>	Name of the book
orders	ArrayList<ORder>	Order of book
borrowings	ArrayLisr<Borrowing>	Borrowing of the book
usersFile	File	Stores User information
bookFile	File	Stores Book information
ordersFile	File	Stores Order information
borrowingsFile	File	Stores information on the borrows
folder	File	Create folder path if path doesnt exist
Methods	**Type**	**Description**
Database()		Creates files, if text file doesnt exist
AddUser(User)	void	Adds user to the database
login(String,String)	int	To login using saved credentials
getUser(int)	User	To read user data in the text file

AddBook(Book)	void	To add book in the database
getUsers()	void	To read user data in the text file
saveUsers	void	To save user information
saveBooks	void	To save book information
getBooks	void	To read book data in the text file
parseBook(String)	Book	To add book data in the text file
getAllBooks()	ArrayList<Book>	To retrieve all book data stored in database
getBook(String)	int	To read book title
getBook(int)	Book	To read book data
deleteBook(int)	void	To remove book data in the text file
deleteAllData()	void	To remove all data in database
addOrder(Order, Book, int)	void	To add order of books
saveOrders()	void	To save order information
getOrders()	void	To read order infromation
userExist(String)	boolean	To check if user exists in database
getUserByName(String)	User	To retrieve user by name
parseOrder(String)	Order	To add order data in the text file
saveBorrowings()	void	
getBorrowings()	void	To read borrow data from text file
parseBorrowing(String)	Borrowing	To add borrow data in the text file
borrowBorrow(Borrowing, Book, int)	void	To borrow book

getBrws()	ArrayList<Borrowing>	To return the number of borrows
returnBook(Borrowing, Book, int)	void	To return the borrowed book

Class Book:

Attributes		Description
title	String	Book title
author	String	Author of book
publisher	String	Publisher of Book
isbn	String	ISBN code of the Book
status	String	Status of the borrow ie: available/unavailable
qty	int	Quantity of book
price	double	Price of book
brwCoppies	int	Number of borrowed copies
Methods		**Description**
Book()		Default constructor
Book(String, String, String, String, String, int, double, int)		Rewrite data based on input
toString()	String	To change data to string
setTitle(String)	void	To rewrite title based on input
getTitle()	String	To read book title
setAuthor(String)	void	To rewrite author based on input
getAuthor()	String	To read author
setPublisher(String)	void	To rewrite publisher based on input
getPublisher()	String	To read publisher

setIsbn(String)	void	To rewrite ISBN based on input
getIsbn()	String	To read ISBN
setStatus(String)	void	To rewrite status based on input
getStatus()	String	To read status
setQty(int)	void	To rewrite quantity based on input
getQty()	int	To read quantity
setPrice(double)	void	To rewrite price based on input
getPrice()	double	To read price
setBrwCoppies(int)	void	To rewrite number of borrow copies based on input
getBrwCoppies()	int	To read number of borrow copies
toString2(String)		To show output

Class User:

Attributes	Type	Description
name	String	Name of User
email	String	Email of User
phoneNumber	String	Phone Number of User
operations	IOOperation[]	To conduct all functions
Methods		**Description**
User()		Default constructor
User(String)		To rewrite data of name
User(String, String, String)		To rewrite data of name, email and phone number based on input
getName()	String	To read name

getEmail()	String	To read email
getPhoneNumber()	String	To read phone number
setName(String)	void	To rewrite name based on input
setEmail(String)	void	To rewrite email based on input
toString()	String	To print output
menu(Database,User)	void	Will show menu based on user type

Borrowing Class:

Attributes	Type	Description
start	LocalDate	Data of Borrow
finish	LocalDate	Return Data of Borrow
daysleft	int	The number of days left for return the book
formatter	DateTimeFormatter	To format the Date type into (DD-MM-YYYY)
Methods	**Type**	**Description**
Borrowing(Book, User)		Default constructor
Borrowing(Book, User, LocalDate, LocalDate)		To rewrite data of name
getStart()	String	To rewrite data of name, email and phone number based on input
getFinish()	String	To read name
getDaysLeft()	int	To read email
getBook()	Book	To read phone number
setBook(Book)	void	To rewrite name based on input
getUser()	User	To rewrite email based on input
setUser(User)	void	To print output

toString()	String	Will show menu based on user type
toString2()	String	

Order Class:

Attributes	Type	Description
book	Book	Data of Borrow
user	User	Return Data of Borrow
price	double	The number of days left for return the book
quantity	int	To format the Date type into (DD-MM-YYYY)
Methods		**Description**
Order(Book, User, double, int)		Default constructor
Order()		To rewrite data of book, user, price and quantity
getBook()	Book	To read book information
getUser()	User	To read user information
getPrice()	double	To read price of book
getQuantity()	int	To read quantity of book
setBook(Book)	void	To rewrite book information based on input
setUser(User)	void	To rewrite user information based on input
setPrice(double)	void	To rewrite price based on input
setQuantity(int)	void	To rewrite quantity of book based on input
toString()	String	To print output
toString2()	String	To print output

Members Class:

Methods	Type	Description
Members()		Default Constructor
Members(String)		To recall the Member privileges with one parenthesis
Members(String, String, String)		To recall the Member privileges with three parenthesis
menu (Database, User)	void	Abstract menu for Members
toString()	String	To print the output

Admin Class:

Methods	Type	Description
Admin()		Default Constructor that initalize array operations with instance of classes that implements IOOperation interface
Admin(String Name)		Constructor that take name as argument and initialize the operations array
Admin(String Name,String Email,String phoneNumber)		Constructor that take name,email & phone number as argument to initialize the following fields in the user superlass and operations array
menu(Database database,User user)	void	Abstract menu for Admin
toString()	String	Method that return string with “” separator and

		append the string admin to indicate the user type

IOOperation Class:

Methods	Type	Description
oper(Database database,User user)	void	An interface class that is defined by Database Class Object and User Class Object that specifically implement user design based on the user type

DeleteAllData Class:

Methods	Type	Description
oper(Database database, User user)	void	Class Implement delete all data from the database

DeleteBook Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implementing that delete book from database class

BorrowBook Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implementing that borrow book from the bookList

PlaceOrder Class:

Methods	Type	Description
operr(Database database, User user)	Void	Class implementing that placeOrder from the book class

CalculateFine Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implementing that calculateFine from the Book.text file

AddBook Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implementing that add book into the database

Exit Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implementing exit function from the user privileges menu
Login()	void	class Login() to relog into the system
Register()	void	class Register() to register into the system

ReturnBook Class:

Methods	Type	Description
operr(Database database, User user)	void	Class the implemented to return book from the borrowing

ViewOrders Class:

Methods	Type	Description
operr(Database database, User user)	void	Class implemented to show view orders from Orders text file in database

Search Class:

Methods	Type	Description
operr(Database database, User user)	void	Class used to search books by using book tittle

ViewBook Class:

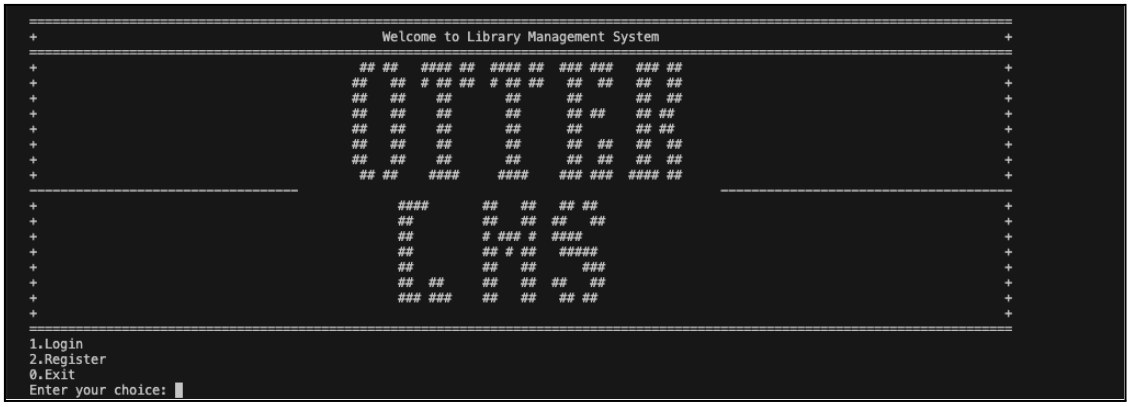
Methods	Type	Description
operr(Database database, User user)	void	Class used to view all books

DeleteBook Class:

Methods	Type	Description
operr(Database database, User user)	void	Class used to delete by getting tittle as input

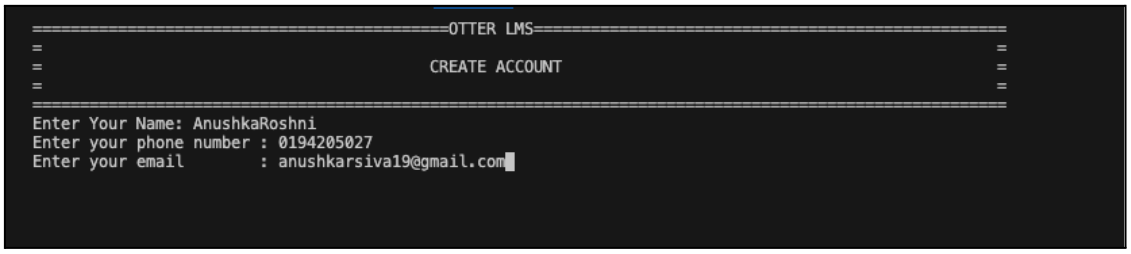
SECTION C: SOURCE CODE AND USER MANUAL

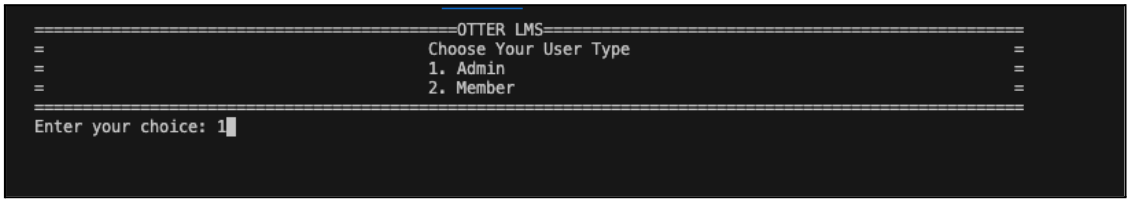
Main Screen:



This screen is our main screen. The user will have to either log in or register to enter the management system. We have two types of users which are Admin and Member.

Register:



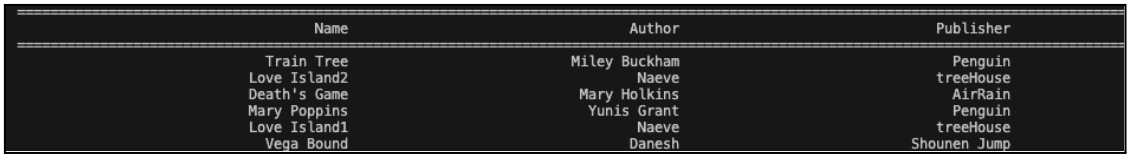


If it's a first-time user they will register. By entering number 2, they will be asked to enter their name, phone number, and email. They will then have to register either as an Admin (enter 1) or Member (enter 2). They will then be automatically logged into the system. The general menu will display based on the user type.

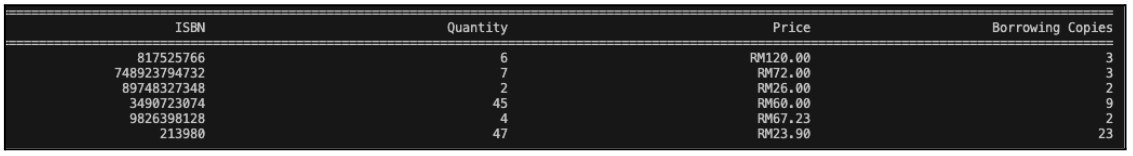
Login:

If it's an existing user, they can log in by entering 1. They will be prompted to enter their phone number and email to log in and the menu will be displayed. The menu shown in this diagram is the Member's Menu.

View Books:



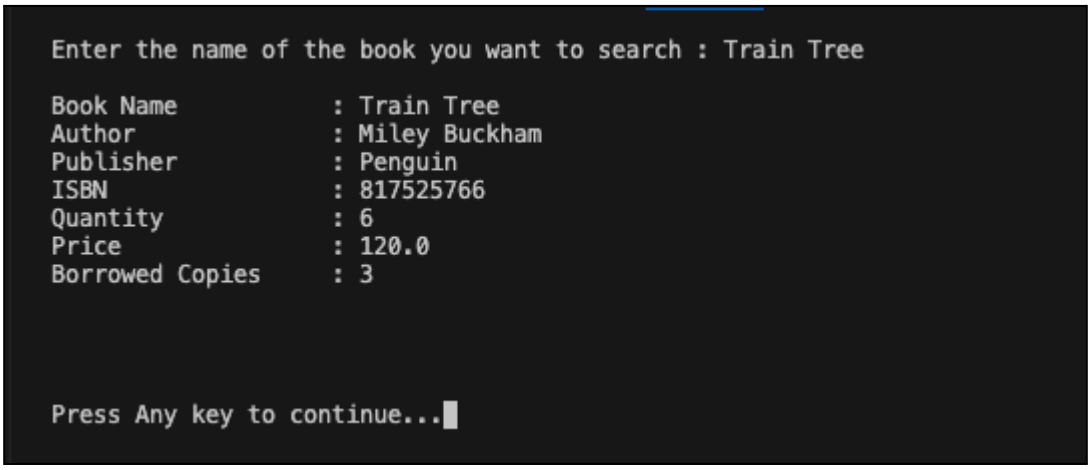
Name	Author	Publisher
Train Tree	Miley Buckham	Penguin
Love Island2	Naeve	treeHouse
Death's Game	Mary Holkins	AirRain
Mary Poppins	Yunis Grant	Penguin
Love Island1	Naeve	treeHouse
Vega Bound	Danesh	Shounen Jump



ISBN	Quantity	Price	Borrowing Copies
817525766	6	RM120.00	3
748923794732	7	RM72.00	3
89748327348	2	RM26.00	2
3490723074	45	RM60.00	9
9826398128	4	RM67.23	2
213980	47	RM23.90	23

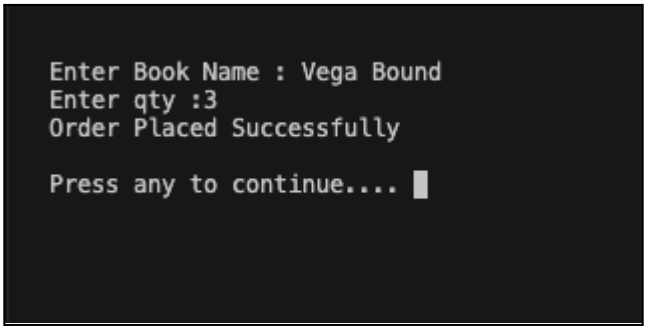
The view book function is accessible to both Admin and Member. They can access it by entering 1. This will then show the list of books available.

Search Books:



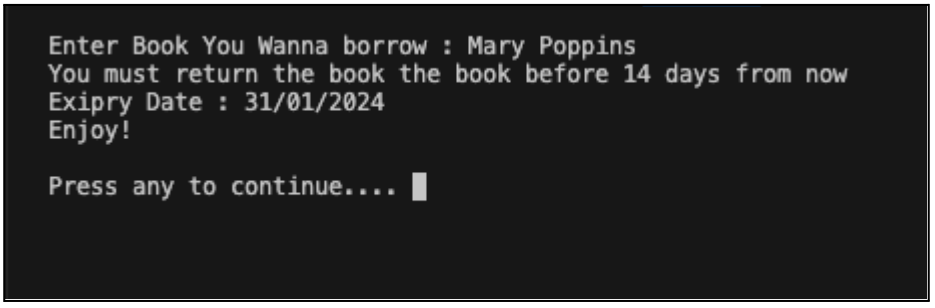
The search book function is accessible to both Admin and Member. They can access it by entering 2 (Member) or entering 4 (Admin). They will then be prompted to enter the name of the book they want to search. This will then display the book information if the book exists in the library database.

Place Orders:

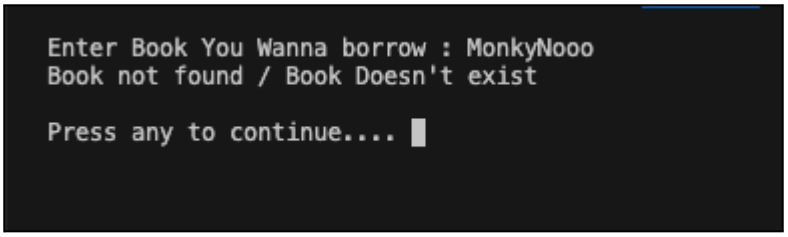


The place order function is accessible for Member only. They can access it by entering 3. They will then be prompted to enter the book name and how many books they would like to place an order on.

Borrow Book:



The borrow book function is accessible for Member only. They can access it by entering 4. They will then be prompted to enter the title of the book they want to borrow and it will display the return date.

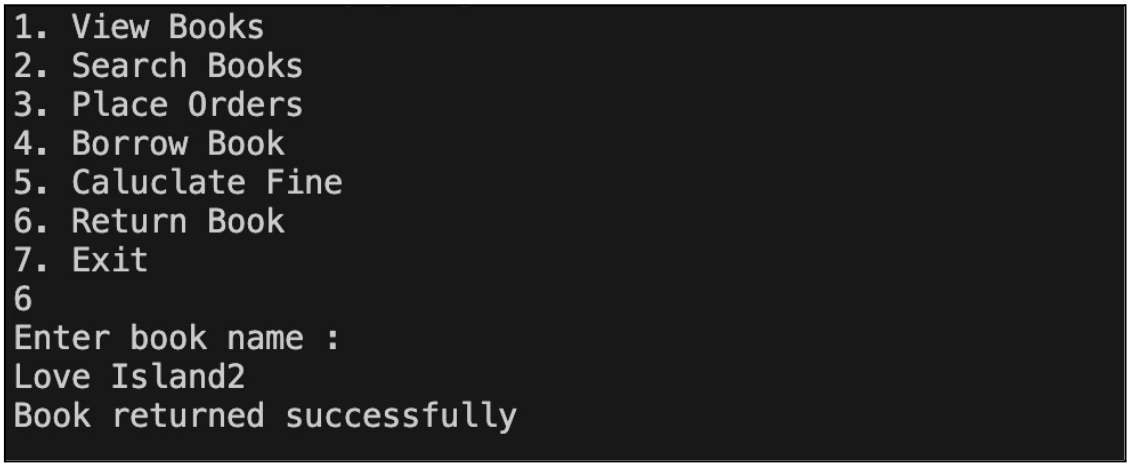


If the book does not exist it will show as the diagram above.

Calculate Fine:

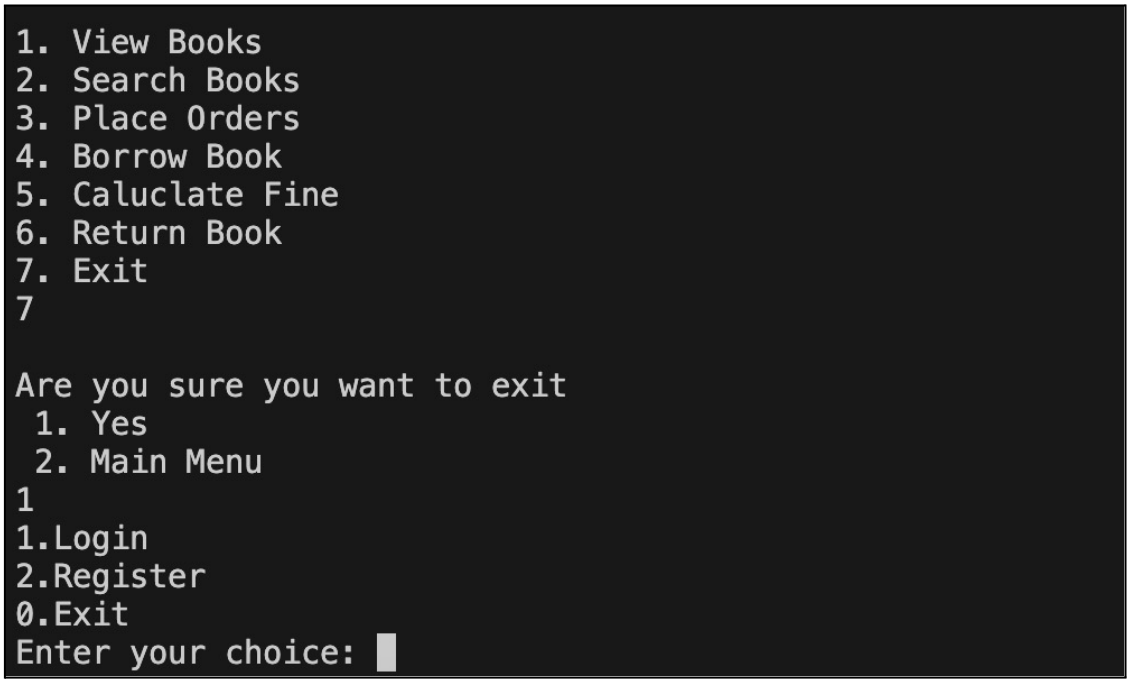
The calculate fine function is accessible for Member only. They can access it by entering 5. The user will be prompted to enter the book name and if a fine is to be paid, it will show the amount. If not, they will be a similar output as the diagram above.

Return Book:



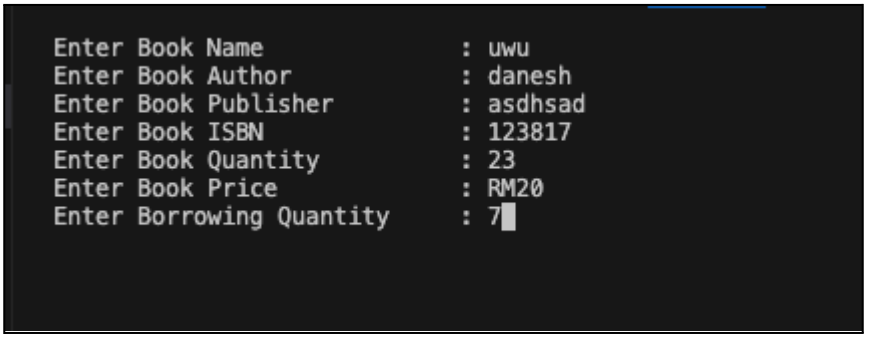
The return book function is accessible for Member only. They can access it by entering 6. They will be prompted to enter the book name which they wish to return.

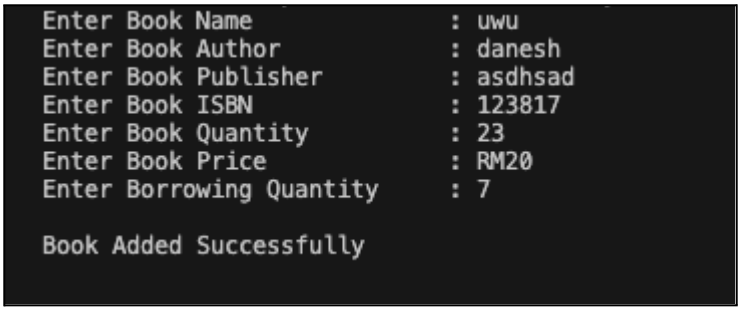
Exit:



The exit function is accessible for both Admin and Member. They can access it by entering 7. This will then show a prompt if the user really wants to exit. By entering 1 they will be brought back to the main page and be logged out automatically. By entering 2, they will be shown the menu based on the user type.

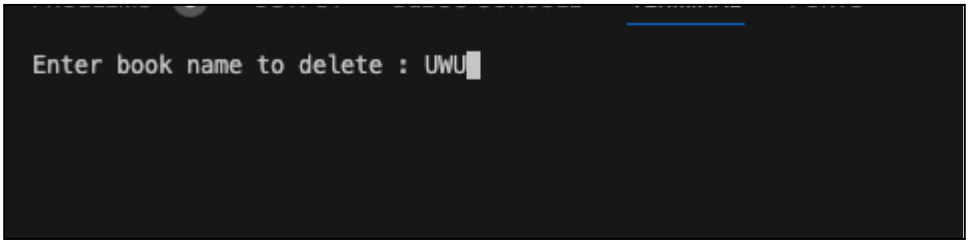
Add Book:

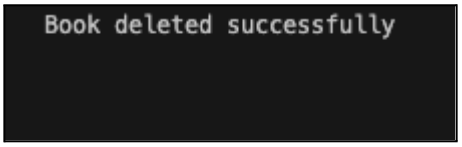




This function is accessible to Admin only. They can access it by clicking on 2. They will then be prompted to enter a book name, an author, a publisher, an ISBN, book quantity, price and borrowing quantity. A confirmation message will then be displayed once the book is added successfully.

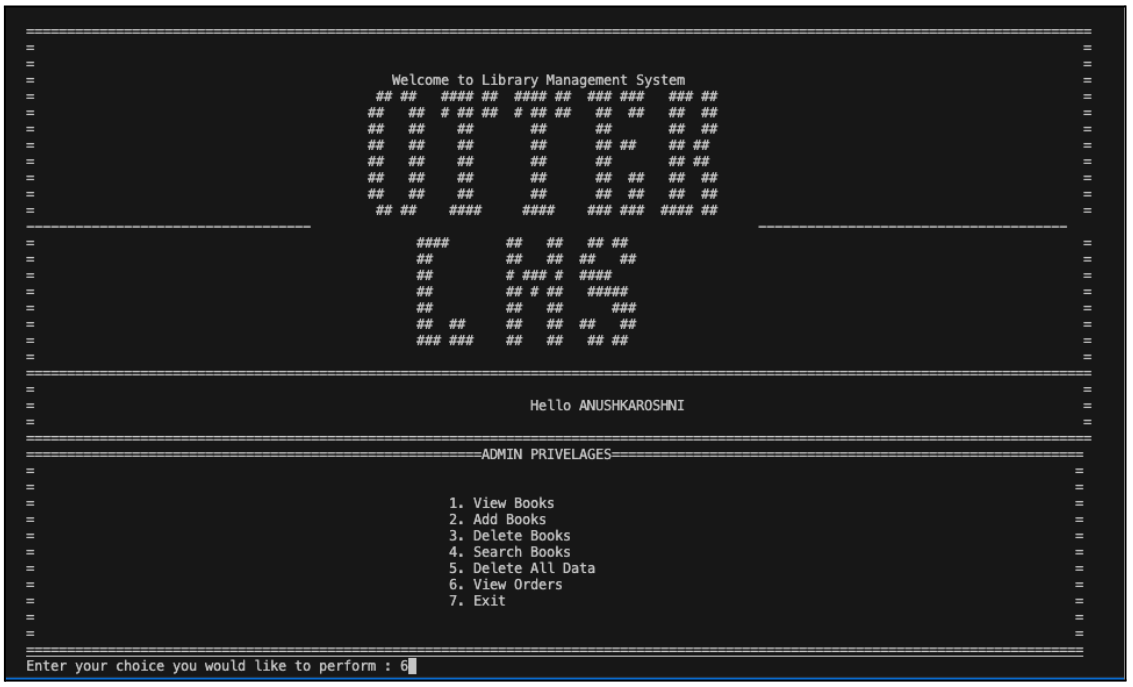
Delete Book:

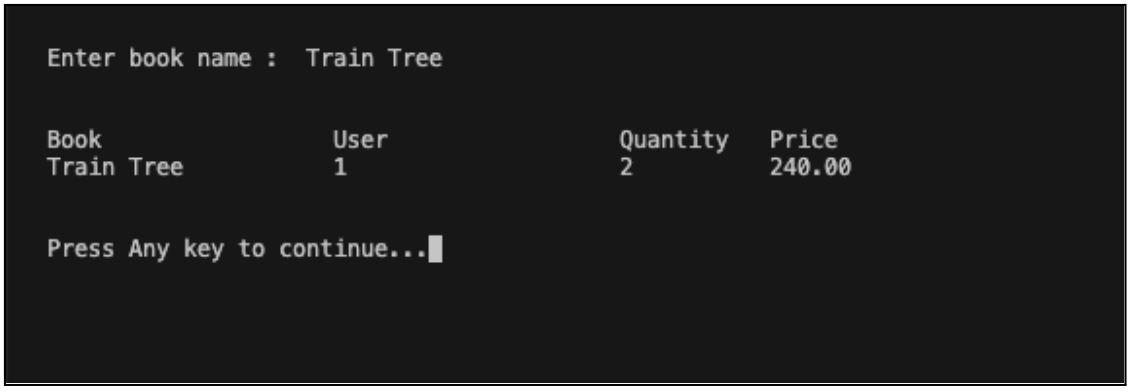




This function is accessible to admin only. They can access it by clicking on 3. They will then be prompted to enter a book name, and if the book exists in the library it will be deleted and a confirmation message will be displayed. They can then view the updated book list by clicking on 1 in the Admin menu.

View Order:





This function is accessible to Admin only. They can access it by clicking on 6. They will then be prompted to enter a book name and any order details for that book will be displayed.

Source Code

AddBook.java

```
package Library;

import java.util.Scanner;

public class AddBook implements IOOperation {

   Scanner scanner = new Scanner(System.in);
   public void oper(Database database,User user){
       Book book = new Book();
       System.out.print("Enter Book Name              : ");
       String name = scanner.nextLine();

       if(database.getBook(name) > -1){
           System.out.println("Book Already Exist\n");
           user.menu(database, user);
           return;

       }else{
           book.setTitle(name);
           System.out.print("Enter Book Author            : ");
           book.setAuthor(scanner.nextLine());
           System.out.print("Enter Book Publisher         : ");
           book.setPublisher(scanner.nextLine());
           System.out.print("Enter Book ISBN              : ");
           book.setIsbn(scanner.nextLine());
           System.out.print("Enter Book Quantity          : ");
           book.setQty(scanner.nextInt());
           System.out.print("Enter Book Price             : RM");
           book.setPrice(scanner.nextDouble());
           System.out.print("Enter Borrowing Quantity     : ");
       book.setBrwCoppies(scanner.nextInt());
           database.AddBook(book);
           System.out.println("\nBook Added Successfully\n");

           user.menu(database, user);
       }

   }
}
```

Admin.java

```
package Library;
import java.util.*;

public class Admin extends User{



   public Admin(){};

   public Admin(String name){
       super(name);
       this.operations = new IOOperation[]{
           new ViewBook(),
           new AddBook(),
           new DeleteBook(),
           new Search(),
           new DeleteAllData(),
           new ViewOrders(),
           new Exit()

       };
   }

   public Admin(String name, String email, String phoneNumber){
       super(name, email, phoneNumber);
       this.operations = new IOOperation[]{
           new ViewBook(),
           new AddBook(),
           new DeleteBook(),
           new Search(),
           new DeleteAllData(),
           new ViewOrders(),
           new Exit()

       };
   }

   @Override
   public void menu(Database database,User user){

       System.out.println("\n\n");
```

```
System.out.println("============================================================
============================================================================
=");
		System.out.println("=
=");
		System.out.println("=
=");
		System.out.println("=
Welcome to Library Management System
=");
		System.out.println("=                                          ## ##
#### ##  #### ##  ### ###   ### ##
=");
		System.out.println("=                                        ##
##  # ## ##  # ## ##   ##  ##   ##  ##
=");
		System.out.println("=                                        ##
##    ##       ##      ##       ##  ##
=");
		System.out.println("=                                        ##
##    ##       ##      ## ##    ## ##
=");
		System.out.println("=                                        ##
##    ##       ##      ##       ## ##
=");
		System.out.println("=                                        ##
##    ##       ##      ##  ##   ##  ##
=");
		System.out.println("=                                        ##
##    ##       ##      ##  ##   ##  ##
=");
		System.out.println("=                                         ## ##
####     ####    ### ###  #### ##
=");
		System.out.println("-----------------------------------
---------------------------------------");
		System.out.println("=
####       ##   ##   ## ##
=");
		System.out.println("=
##         ##   ##  ##   ##
=");
		System.out.println("=
##         # ### #  ####
```

```
=");
        System.out.println("=
##          ## # ##    #####
=");
        System.out.println("=
##          ##   ##      ###
=");
        System.out.println("=
##  ##      ##   ##  ##   ##
=");
        System.out.println("=
### ###     ##   ##   ## ##
=");
        System.out.println("=
=");

System.out.println("============================================================
============================================================================
=");
        System.out.println("=
=");
        System.out.printf("=
Hello %-61s =%n", this.name.toUpperCase());
        System.out.println("=
=");

System.out.println("============================================================
============================================================================
=");




System.out.println("============================================================
=ADMIN
PRIVELAGES==================================================================");
        System.out.println("=
=");
        System.out.println("=
=");
        System.out.println("=
1. View Books
=");
        System.out.println("=
2. Add Books
```

```
=");
        System.out.println("=
3. Delete Books
=");
        System.out.println("=
4. Search Books
=");
        System.out.println("=
5. Delete All Data
=");
        System.out.println("=
6. View Orders
=");
        System.out.println("=
7. Exit
=");
        System.out.println("=
=");
        System.out.println("=
=");

System.out.println("============================================================
================================================================================
");


        System.out.print("Enter your choice you would like to perform : ");
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();

        System.out.print("Press any to continue.... ");
        Scanner pause = new Scanner(System.in);
        pause.nextLine();
        System.out.print("\033[H\033[2J");
            System.out.flush();

        this.operations[n-1].oper(database,user);
        scanner.close();
        pause.close();


    }

    public String toString(){
```

```
        return name + "<N/>" + email + "<N/>" + phoneNumber + "<N/>" +
"Admin";
    }

}
```

Book.java

```
package Library;
public class Book {

   private String title;
   private String author;
   private String publisher;
   private String isbn;
   private String status;
   private int qty;
   private double price;
   private int brwCoppies;


   public Book(){
   };

   public Book(String title, String author, String publisher, String isbn,
String status, int qty, double price, int brwCoppies) {
       this.title = title;
       this.author = author;
       this.publisher = publisher;
       this.isbn = isbn;
       this.status = status;
       this.qty = qty;
       this.price = price;
       this.brwCoppies = brwCoppies;
   }

   public String toString(){
       String text =   String.format("%-20s: %s%n", "Book Name",
this.title) +
                       String.format("%-20s: %s%n", "Author", this.author)
+
                       String.format("%-20s: %s%n", "Publisher",
this.publisher) +
                       String.format("%-20s: %s%n", "ISBN", this.isbn) +
                       String.format("%-20s: %s%n", "Quantity",
String.valueOf(qty)) +
                       String.format("%-20s: %s%n", "Price",
String.valueOf(price)) +
                       String.format("%-20s: %s%n", "Borrowed Copies",
String.valueOf(brwCoppies));
```

```
        return text;
    }

    public void setTitle(String title){
        this.title = title;
    }
    public String getTitle(){
        return this.title;
    }

    public void setAuthor(String author){
        this.author = author;
    }
    public String getAuthor(){
        return this.author;
    }

    public void setPublisher(String publisher){
        this.publisher = publisher;
    }
    public String getPublisher(){
        return this.publisher;
    }

    public void setIsbn(String isbn){
        this.isbn = isbn;
    }
    public String getIsbn(){
        return this.isbn;
    }

    public void setStatus(String status){
        this.status = status;
    }
    public String getStatus(){
        return this.status;
    }

    public void setQty(int qty){
        this.qty = qty;
    }

    public int getQty(){
```

```
        return this.qty;
    }

    public void setPrice(double price){
        this.price = price;
    }

    public double getPrice(){
        return this.price;
    }

    public void setBrwCoppies(int brwCoppies){
        this.brwCoppies = brwCoppies;
    }

    public int getBrwCoppies(){
        return this.brwCoppies;
    }

    public String toString2(){
        String text = title + "<N/>" + author + "<N/>" + publisher + "<N/>"
+ isbn + "<N/>" + String.valueOf(qty) + "<N/>" + String.valueOf(price) +
"<N/>" + String.valueOf(brwCoppies);

        return text;
    }



}
```

BorrowBook.java

```
package Library;

import java.util.Scanner;

public class BorrowBook implements IOOperation{
   Scanner s = new Scanner(System.in);
   public void oper(Database database,User user){

       System.out.print("Enter Book You Wanna borrow : ");
       String bookName = s.nextLine();

       int i = database.getBook(bookName);

       if (i>-1){
           Book book = database.getBook(i);
           boolean n = true;
           for(Borrowing b  : database.getBrws()){
               if(b.getBook().getTitle().equals(bookName) &&
b.getUser().getName().equals(user.getName())){
                   n = false;
                   System.out.println("You already borrowed this book\n");
               }
           }

           if(n){
               if(book.getBrwCoppies()>1){
                   Borrowing borrowing = new Borrowing(book,user);
                   book.setBrwCoppies(book.getBrwCoppies()-1);
                   database.borrowBook(borrowing, book, i);
                   System.out.println("You must return the book the book
before 14 days from now \n" + "Exipry Date : " + borrowing.getFinish() +
"\nEnjoy!\n");
               }else{
                   System.out.println("Sorry, No available copies for this
book\n");
               }
           }

       }else{
           System.out.println("Book not found / Book Doesn't exist\n");
       }
```

```
        System.out.print("Press any to continue.... ");
        Scanner pause = new Scanner(System.in);
        pause.nextLine();
        System.out.print("\033[H\033[2J");
            System.out.flush();

        user.menu(database, user);

    }
}
```

Borrowing.java

```
package Library;
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;


public class Borrowing {
   LocalDate start;
   LocalDate finish;

   int daysleft;
   Book book;
   User user;

   DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");


   public Borrowing(Book book, User user) {
       start  = LocalDate.now();
       finish = start.plusDays(14);
       daysleft = Period.between(start, finish).getDays();
       this.book = book;
       this.user = user;
   }

   public Borrowing(LocalDate start, LocalDate finish,Book book,User user)
{
       this.start = start;
       this.finish = finish;
       this.daysleft = Period.between(finish,LocalDate.now()).getDays();
       this.book = book;
       this.user = user;
   }


   public String getStart(){
       return formatter.format(start);
   }

   public String getFinish(){
       return formatter.format(finish);
   }
```

```
    public int getDaysleft(){
        return Period.between(finish,LocalDate.now()).getDays();
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book){
        this.book = book;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user){
        this.user = user;
    }

    public String toString(){
        return "Borrowing time : " + start+"\nExpiry Date : " + finish +
"\nDays left : " + daysleft;
    }

    public String toString2(){
        return getStart() + "<N/>" + getFinish() + "<N/>" + getDaysleft() +
"<N/>" + book.getTitle() + "<N/>" + user.getName()+"<N/>";
    }


}
```

CalculateFine.java

```
package Library;
import java.util.Scanner;
public class CalculateFine implements IOOperation {

   Scanner scanner = new Scanner(System.in);
   public void oper(Database database,User user){
       System.out.println("Enter book name : ");
       String bookName = scanner.nextLine();

       boolean g = true;

       for(Borrowing b : database.getBrws()){
           if(b.getBook().getTitle().equals(bookName) &&
b.getUser().getName().equals(user.getName())){
               if(b.getDaysleft()>0){
                   System.out.println("You are late\n" + "You have to pay "
+ b.getDaysleft()*50 + " as fine");
               }else{
                   System.out.println("You dont have to pay any fine");
               }
               g= false;
               break;
           }
       }
       user.menu(database, user);

       if(g){
           System.out.println("You didn't borrow this book");
       }



   }
}
```

Database.java

```
package Library;
import java.util.ArrayList;


import java.io.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
public class Database {

   private  ArrayList<User> users = new ArrayList<User>();
   private ArrayList<String> userName = new ArrayList<String>();
   private ArrayList<Book> books = new ArrayList<Book>();
   private ArrayList<String> bookName = new ArrayList<String>();
   private ArrayList<Order> orders = new ArrayList<Order>();
   private ArrayList<Borrowing> borrowings = new ArrayList<Borrowing>();


   private File usersfile = new File("src/data/Users.txt");
   private File bookfile = new File("src/data/Books.txt");
   private File ordersfile = new File("src/data/Orders.txt");
   private File borrowingsfile = new File("src/data/Borrowings.txt");
   private File folder = new File("src/data");

   public Database(){
       if (!folder.exists()){
           folder.mkdir();
       }
       if (!usersfile.exists()){
           try{
               usersfile.createNewFile();
           }catch (Exception e){}
       }
       if (!bookfile.exists()){
           try{
               bookfile.createNewFile();
           }catch (Exception e){}
       }
       if (!ordersfile.exists()){
           try{
               ordersfile.createNewFile();
           }catch (Exception e){}
       }
```

```
        if (!borrowingsfile.exists()){
            try{
                borrowingsfile.createNewFile();
            }catch (Exception e){}
        }
        getUsers();
        getBooks();
        getOrders();
        getBorrowings();
    }

    public void AddUser(User m){
        users.add(m);
        userName.add(m.getName());
        saveUsers();
    }

    public int login(String phoneNumber,String email){
        int n = -1;
        for(User s : users){
            if(s.getPhoneNumber().equals(phoneNumber) &&
s.getEmail().equals(email)){
                n = users.indexOf(s);
                break;
            }
        }
        return n;
    }

    public User getUser(int n){
        return  users.get(n);
    }


    public void AddBook(Book b){
        books.add(b);
        bookName.add(b.getTitle());
        saveBooks();
    }

    private void getUsers(){
        String text1 = "";

        try{
```

```
            BufferedReader br1 = new BufferedReader(new
FileReader(usersfile));
            String s1;
            while((s1 = br1.readLine()) != null){
                text1 = text1 + s1;
            }
            br1.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }

        if(!text1.matches("") || !text1 .isEmpty()){
            String[] a1 = text1.split("<NewUser/>");
            for(String s : a1){
                String[] a2 = s.split("<N/>");
                if(a2[3].equals("Admin")){
                    User user = new Admin(a2[0],a2[1],a2[2]);
                    users.add(user);
                    userName.add(user.getName());

                }else{
                    User user = new Members(a2[0],a2[1],a2[2]);
                    users.add(user);
                    userName.add(user.getName());
                }
            }
        }

    }

    private void saveUsers(){
        String text1 = "";
        for(User user : users){
           text1 = text1 + user.toString() + "<NewUser/>\n";
        }
        try{
            PrintWriter pw = new PrintWriter(new FileWriter(usersfile));
            pw.println(text1);
            pw.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }
    }
```

```
   private void saveBooks(){
       String text1 = "";
       for(Book book : books){
          text1 = text1 + book.toString2() + "<NewBook/>\n";
       }
       try{
           PrintWriter pw = new PrintWriter(new FileWriter(bookfile));
           pw.println(text1);
           pw.close();
       }catch(Exception e){
           System.err.println(e.toString());
       }
   }

   private void getBooks(){
       String text1 = "";

       try{
           BufferedReader br1 = new BufferedReader(new
FileReader(bookfile));
           String s1;
           while((s1 = br1.readLine()) != null){
               text1 = text1 + s1;
           }
           br1.close();
       }catch(Exception e){
           System.err.println(e.toString());
       }

       if(!text1.matches("") || !text1 .isEmpty()){
           String[] a1 = text1.split("<NewBook/>");
           for(String s : a1){
               Book book = parseBook(s);
               books.add(book);
               bookName.add(book.getTitle());

           }
       }

   }

   public Book parseBook(String s){
       String[] a = s.split("<N/>");
       Book book = new Book();
```

```
        book.setTitle(a[0]);
        book.setAuthor(a[1]);
        book.setPublisher(a[2]);
        book.setIsbn(a[3]);
        book.setQty(Integer.parseInt(a[4]));
        book.setPrice(Double.parseDouble(a[5]));
        book.setBrwCoppies(Integer.parseInt(a[6]));
        return book;
    }

    public ArrayList<Book> getAllBooks(){
        return books;
    }

    public int getBook(String tittle){
        int i = -1;
        for(Book book : books){
            if(book.getTitle().equals(tittle)){
                i = books.indexOf(book);
            }
        }
        return i;
    }

    public Book getBook(int i){
        return books.get(i);
    }

    public void deleteBook(int n){
        books.remove(n);
        bookName.remove(n);
        saveBooks();
    }

    public void deleteAllData(){
        if (usersfile.exists()){
            try{
                usersfile.delete();
            }catch (Exception e){}
        }
        if (bookfile.exists()){
            try{
                bookfile.delete();
            }catch (Exception e){}
```

```
        }
        if (ordersfile.exists()){
            try{
                ordersfile.delete();
            }catch (Exception e){}
        }
        if (borrowingsfile.exists()){
            try{
                borrowingsfile.delete();
            }catch (Exception e){}
        }
    }

    public void addOrder(Order order,Book book,int bookindex){
        orders.add(order);
        books.set(bookindex, book);
        saveOrders();
        saveBooks();
    }

    private void saveOrders(){
        String text1 = "";
        for(Order order : orders){
           text1 = text1 + order.toString2() + "<NewOrder/>\n";
        }
        try{
            PrintWriter pw = new PrintWriter(new FileWriter(ordersfile));
            pw.println(text1);
            pw.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }
    }

    private void getOrders(){
        String text1 = "";

        try{
            BufferedReader br1 = new BufferedReader(new
FileReader(ordersfile));
            String s1;
            while((s1 = br1.readLine()) != null){
                text1 = text1 + s1;
            }
```

```
            br1.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }

        if(!text1.matches("") || !text1 .isEmpty()){
            String[] a1 = text1.split("<NewOrder/>");
            for(String s : a1){
                Order order = parseOrder(s);
                orders.add(order);

            }
        }

    }


    public boolean userExist(String name ){
        boolean f = false;
        for(User user : users){
            if(user.getName().toLowerCase().equals(name.toLowerCase())){
                break;
            }
        }

        return f;

    }

    private User getUserByName(String name){
        User u = new Members("");
        for(User user : users){
            if(user.getName().equals(name)){
                u = user;
                break;
            }
        }
        return u;
    }

    private Order parseOrder(String s ){
        String[] a = s.split("<N/>");
        Order order = new
Order(books.get(getBook(a[0])),getUserByName(a[1]),Double.parseDouble(a[2])
```

```
,Integer.parseInt(a[3]));
        return order;

    }

    public ArrayList<Order> getAllOrders(){
        return orders;
    }

    private void saveBorrowings(){
        String text1 = "";
        for(Borrowing borrowing : borrowings){
           text1 = text1 + borrowing.toString2() + "<NewBorrowing/>\n";
        }
        try{
            PrintWriter pw = new PrintWriter(new
FileWriter(borrowingsfile));
            pw.println(text1);
            pw.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }
    }

    private void getBorrowings(){
        String text1 = "";

        try{
            BufferedReader br1 = new BufferedReader(new
FileReader(borrowingsfile));
            String s1;
            while((s1 = br1.readLine()) != null){
                text1 = text1 + s1;
            }
            br1.close();
        }catch(Exception e){
            System.err.println(e.toString());
        }

        if(!text1.matches("") || !text1 .isEmpty()){
            String[] a1 = text1.split("<NewBorrowing/>");
            for(String s : a1){
                Borrowing borrowing = parseBorrowing(s);
                borrowings.add(borrowing);
```

```
            }
        }

    }

    private Borrowing parseBorrowing(String s ){
        String[] a = s.split("<N/>");
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate start = LocalDate.parse(a[0],formatter);
        LocalDate finish = LocalDate.parse(a[1],formatter);
        Book book = getBook(getBook(a[3]));
        User user = getUserByName(a[4]);
        Borrowing brw = new Borrowing(start,finish,book, user);
        return brw;
    }

    public void borrowBook(Borrowing brw, Book book, int bookindex){
        borrowings.add(brw);
        books.set(bookindex, book);
        saveBorrowings();
        saveBooks();
    }

    public ArrayList<Borrowing> getBrws(){
        return borrowings;
    }

    public void returnBook(Borrowing b, Book book, int bookindex){
        borrowings.remove(b);
        books.set(bookindex, book);
        saveBorrowings();
        saveBooks();
    }


}
```

DeleteAllData.java

```
package Library;
import java.util.*;
public class DeleteAllData implements IOOperation{
   public void oper(Database database,User user){

       System.out.println("\nAre you sure you want to delete all data?\n 1.
Continue\n 2. Main Menu");
       Scanner s = new Scanner(System.in);
       int n = s.nextInt();
       if(n==1){
           database.deleteAllData();
           System.out.println("All data deleted successfully");
       }
       else{
           user.menu(database, user);
       }


       s.close();



   }
}
```

DeleteBook.java

```
package Library;
import java.util.Scanner;
public class DeleteBook implements IOOperation {

   Scanner s = new Scanner(System.in);

   public void oper(Database database,User user){

       System.out.print("Enter book name to delete : ");
       String bookName = s.nextLine();


       int i = database.getBook(bookName);
       if (i>-1){
           database.deleteBook(i);
           System.out.println("Book deleted successfully\n");
       }else{
           System.out.println("Book not found / Book Doesn't exist\n");
       }

       user.menu(database, user);


   }
}
```

Exit.java

```
package Library;
import java.util.Scanner;


public class Exit implements IOOperation {
   Scanner scanner = new Scanner(System.in);
   Database database;
   User user;
   public void oper(Database database,User user){
       this.database = database;
       this.user = user;
       System.out.println("\nAre you sure you want to exit\n 1. Yes\n 2.
Main Menu");
       int n = scanner.nextInt();
       System.out.print("\033[H\033[2J");
           System.out.flush();
       if(n==1){
           System.out.println("1.Login");
           System.out.println("2.Register");
           System.out.println("0.Exit");
           System.out.print("Enter your choice: ");
           int num  = scanner.nextInt();

           switch(num){
               case 1:
                   Login();
                   break;
               case 2:
                   Register();
                   break;
               case 0:
                   System.out.println("Exit");
                   break;
               default:
                   System.out.println("Invalid choice");
                   break;
           }
       }
       else{
           user.menu(database, user);
       }
   }
```

```
   public void Login(){
       System.out.println("Login");
       System.out.print("Enter your Phone Number : ");
       String phoneNumber = scanner.next();
       System.out.print("Enter your Email: ");
       String email = scanner.next();
       int n = database.login(phoneNumber, email);
       if(n!= -1){
           User user = database.getUser(n);
           user.menu( database, user);
       }else{
           System.out.println("User Does not exist");
       }
   }

   public void Register(){

System.out.println("=============================================OTTER
LMS================================================");
       System.out.println("=
=");
       System.out.println("=                                                CREATE
ACCOUNT                                          =");
       System.out.println("=
=");

System.out.println("============================================================
==============================================");
       System.out.print("Enter Your Name: ");
       String name = scanner.next();
       if(database.userExist(name)){
           System.out.println("This User/Username is already exist
!.....");
           System.out.println("Please try again,Thank You.....");
           Register();
       }
       System.out.print("Enter your phone number : ");
       String phoneNumber = scanner.next();
       System.out.print("Enter your email        : ");
       String email = scanner.next();
```

```
        System.out.print("\033[H\033[2J");
        System.out.flush();



System.out.println("=============================================OTTER
LMS================================================");
        System.out.println("=                                                  Choose
Your User Type                                    =");
        System.out.println("=                                                  1.
Admin                                                =");
        System.out.println("=                                                  2.
Member                                               =");

System.out.println("=====================================================
================================================");


        int n2 = scanner.nextInt();
        User user;
        if(n2==1){
            user = new Admin(name, email, phoneNumber);

        }
        else{
            user = new Members(name, email, phoneNumber);
        }
        database.AddUser(user);
        System.out.print("\033[H\033[2J");
        System.out.flush();
        user.menu(database, user);




    }


}
```

IOOperation.java

```
package Library;
public interface IOOperation {
   public void oper(Database database,User user);
}
```

Main.java

```
package Library;
import java.util.Scanner;

public class Main{

   private static Scanner scanner = new Scanner(System.in);
   static Database database;
   public static void main(String[] args) {

       database = new Database();



System.out.println("============================================================
================================================================================")
;
       System.out.println("+
Welcome to Library Management System
+");

System.out.println("============================================================
================================================================================")
;
       System.out.println("+                                          ## ##
#### ##  #### ##  ### ###   ### ##
+");
       System.out.println("+                                        ##
##  # ## ##  # ## ##   ##  ##   ##  ##
+");
       System.out.println("+                                        ##
##    ##       ##      ##       ##  ##
+");
```

```
        System.out.println("+                                                  ##
##    ##       ##      ## ##    ## ##
+");
        System.out.println("+                                                  ##
##    ##       ##      ##       ## ##
+");
        System.out.println("+                                                  ##
##    ##       ##      ##  ##   ##  ##
+");
        System.out.println("+                                                  ##
##    ##       ##      ##  ##   ##  ##
+");
        System.out.println("+                                                   ## ##
####     ####    ### ###  #### ##
+");
        System.out.println("-----------------------------------
-------------------------------------");
        System.out.println("+
####       ##   ##   ## ##
+");
        System.out.println("+
##         ##   ##  ##   ##
+");
        System.out.println("+
##         # ### #  ####
+");
        System.out.println("+
##         ## # ##   #####
+");
        System.out.println("+
##         ##   ##      ###
+");
        System.out.println("+
##  ##     ##   ##  ##   ##
+");
        System.out.println("+
### ###    ##   ##   ## ##
+");
        System.out.println("+
+");

System.out.println("==========================================================
============================================================================")
;
```

```
            int num;
            System.out.println("1.Login");
            System.out.println("2.Register");
            System.out.println("0.Exit");
            System.out.print("Enter your choice: ");
            num  = scanner.nextInt();

            System.out.print("\033[H\033[2J");
            System.out.flush();

            switch(num){
                case 1:
                    Login();
                    break;
                case 2:
                    Register();
                    break;
                case 0:
                    System.out.println("Exit");
                    break;
                default:
                    System.out.println("Invalid choice");
                    break;
            }
            scanner.close();



    }

    public static void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }


    public static void Login(){

System.out.println("============================================OTTER
LMS================================================");
        System.out.println("=
=");
        System.out.println("=
```

```
LOGIN                                                    =");
        System.out.println("=
=");

System.out.println("==========================================================
================================================");
        System.out.print("Enter your Phone Number : ");
        String phoneNumber = scanner.next();
        System.out.print("Enter your Email: ");
        String email = scanner.next();

        System.out.print("\033[H\033[2J");
        System.out.flush();

        int n = database.login(phoneNumber, email);
        if(n!= -1){
            User user = database.getUser(n);
            user.menu( database, user);
        }else{
            System.out.println("User Does not exist");
        }
    }



    public static void Register(){

System.out.println("=============================================OTTER
LMS================================================");
        System.out.println("=
=");
        System.out.println("=                                                CREATE
ACCOUNT                                          =");
        System.out.println("=
=");

System.out.println("==========================================================
================================================");
        System.out.print("Enter Your Name: ");
        String name = scanner.next();
        if(database.userExist(name)){
            System.out.println("This User/Username is already exist
!.....");
            System.out.println("Please try again,Thank You.....");
```

```
            Register();
        }
        System.out.print("Enter your phone number : ");
        String phoneNumber = scanner.next();
        System.out.print("Enter your email        : ");
        String email = scanner.next();
        System.out.print("\033[H\033[2J");
        System.out.flush();



System.out.println("=============================================OTTER
LMS=================================================");
        System.out.println("=                                                Choose
Your User Type                                    =");
        System.out.println("=                                                1.
Admin                                               =");
        System.out.println("=                                                2.
Member                                              =");

System.out.println("==========================================================
================================================");
        System.out.print("Enter your choice: ");
        int n2 = scanner.nextInt();
        User user;
        if(n2==1){
            user = new Admin(name, email, phoneNumber);

        }
        else{
            user = new Members(name, email, phoneNumber);
        }
        database.AddUser(user);
        System.out.print("\033[H\033[2J");
        System.out.flush();
        user.menu(database, user);




    }
```

```
}
```

Members.java

```
package Library;
import java.util.Scanner;

public class Members extends User {

   public Members(){};

   public Members(String name){
       super(name);
       this.operations = new IOOperation[]{
           new ViewBook(),
           new Search(),
           new PlaceOrder(),
           new BorrowBook(),
           new CalculateFine(),
           new ReturnBook(),
           new Exit()

       };
   }

   public Members(String name, String email, String phoneNumber){
       super(name, email, phoneNumber);
       this.operations = new IOOperation[]{
           new ViewBook(),
           new Search(),
           new PlaceOrder(),
           new BorrowBook(),
           new CalculateFine(),
           new ReturnBook(),
           new Exit()

       };
   }
```

```
   @Override
   public void menu(Database database,User user){

       System.out.println("\n\n");

System.out.println("============================================================
============================================================================
=");
       System.out.println("=
=");
       System.out.println("=
=");
       System.out.println("=
Welcome to Library Management System
=");
       System.out.println("=                                          ## ##
#### ##  #### ##  ### ###   ### ##
=");
       System.out.println("=                                        ##
##  # ## ##  # ## ##   ##  ##   ##  ##
=");
       System.out.println("=                                        ##
##    ##       ##      ##       ##  ##
=");
       System.out.println("=                                        ##
##    ##       ##      ## ##    ## ##
=");
       System.out.println("=                                        ##
##    ##       ##      ##       ## ##
=");
       System.out.println("=                                        ##
##    ##       ##      ##  ##   ##  ##
=");
       System.out.println("=                                        ##
##    ##       ##      ##  ##   ##  ##
=");
       System.out.println("=                                         ## ##
####     ####    ### ###  #### ##
=");
       System.out.println("----------------------------------
-------------------------------------");
       System.out.println("=
####       ##   ##   ## ##
```

```
=");
        System.out.println("=
##          ##    ##   ##    ##
=");
        System.out.println("=
##          # ### #   ####
=");
        System.out.println("=
##          ## # ##    #####
=");
        System.out.println("=
##          ##    ##       ###
=");
        System.out.println("=
##  ##      ##    ##   ##    ##
=");
        System.out.println("=
### ###     ##    ##    ## ##
=");
        System.out.println("=
=");

System.out.println("============================================================
============================================================================
=");
        System.out.println("=
=");
        System.out.printf("=
Hello %-61s =%n", this.name.toUpperCase());
        System.out.println("=
=");

System.out.println("============================================================
============================================================================
=");



System.out.println("============================================================
=MEMBER
PRIVELAGES==================================================================");
        System.out.println("=
=");
        System.out.println("=
1. View Books
```

```
=");
        System.out.println("=
2. Search Books
=");
        System.out.println("=
3. Place Orders
=");
        System.out.println("=
4. Borrow Book
=");
        System.out.println("=
5. Caluclate Fine
=");
        System.out.println("=
6. Return Book
=");
        System.out.println("=
7. Exit
=");
        System.out.println("=
=");
        System.out.println("=
=");

System.out.println("============================================================
============================================================================
=");



        System.out.print("Enter your choice you woud like to perform : ");
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();

        System.out.print("Press any to continue.... ");
        Scanner pause = new Scanner(System.in);
        pause.nextLine();

        System.out.print("\033[H\033[2J");
            System.out.flush();

        this.operations[n-1].oper(database,user);
        scanner.close();
        pause.close();
```

```
    }
    public String toString(){
        return name + "<N/>" + email + "<N/>" + phoneNumber + "<N/>" +
"Member";
    }

}
```

Order.java

```
package Library;

public class Order {
   private Book book;
   private User user;
   private double price;
   private int quantity;

   public Order(Book book, User user, double price, int quantity) {
       this.book = book;
       this.user = user;
       this.price = price;
       this.quantity = quantity;
   }

   public Order(){

   }

   public Book getBook() {
       return this.book;
   }

   public User getUser() {
       return this.user;
   }

   public double getPrice() {
       return this.price;
   }

   public int getQuantity() {
       return this.quantity;
   }

   public void setBook(Book book) {
       this.book = book;
   }

   public void setUser(User user) {
       this.user = user;
   }
```

```
    public void setPrice(double price) {
        this.price = price;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String toString(){
        return "Book name : " + book.getTitle() + "\n" + "User name : " +
user.getName() + "\n" + "Price : " + String.valueOf(price) + "\n" +
"Quantity : " + String.valueOf(quantity) + "\n";
    }

    public String toString2(){
        return book.getTitle() + "<N/>" + user.getName() + "<N/>" +
String.valueOf(price) + "<N/>" + String.valueOf(quantity);
    }

}
```

PlaceOrder.java

```
package Library;

import java.util.Scanner;

public class PlaceOrder implements IOOperation{

   Scanner scanner = new Scanner(System.in);
   public void oper(Database database,User user){


       Order order = new Order();
       System.out.print("\nEnter Book Name : ");
       String bookName = scanner.nextLine();

       int i = database.getBook(bookName);

       if(i<=-1){
           System.out.println("Book doesn't exist in the database");
       }else{
           Book book = database.getBook(i);
           order.setBook(book);
           order.setUser(user);
           System.out.print("Enter qty :");
           int qty = scanner.nextInt();
           order.setQuantity(qty);
           order.setPrice(book.getPrice()*qty);

           int bookindex = database.getBook(book.getTitle());

           book.setQty(book.getQty()-qty);
           database.addOrder(order,book,bookindex);
           System.out.println("Order Placed Successfully\n");


       }

       System.out.print("Press any to continue.... ");
       Scanner pause = new Scanner(System.in);
       pause.nextLine();
       System.out.print("\033[H\033[2J");
           System.out.flush();
```

```
        user.menu(database, user);



    }


}
```

ReturnBook.java

```
package Library;
import java.util.Scanner;
public class ReturnBook implements IOOperation {

   Scanner scanner = new Scanner(System.in);
   public void oper(Database database,User user){
       System.out.print("Enter book name : ");
       String bookName = scanner.nextLine();

       if(!database.getBrws().isEmpty()){
           for(Borrowing b : database.getBrws()){
               if(b.getBook().getTitle().matches(bookName) &&
b.getUser().getName().matches(user.getName())){
                   Book book = b.getBook();
                   int i = database.getAllBooks().indexOf(book);
                   if(b.getDaysleft()>0){
                       System.out.println("You are late " + "You have to
pay" + Math.abs(b.getDaysleft()*50) + "as fine");
                   }
                   book.setBrwCoppies(book.getBrwCoppies()+1);
                   database.returnBook(b, book, i);
                   System.out.println("Book returned successfully\n");
                   break;
               }else{
                   System.out.println("You didn't borrow this book\n");
               }
           }
       }else{
           System.out.println("You didn't borrow any book\n");
       }

       System.out.print("Press any to continue.... ");
       Scanner pause = new Scanner(System.in);
       pause.nextLine();
       System.out.print("\033[H\033[2J");
           System.out.flush();

       user.menu(database, user);

   }
}
```

Search.java

```
package Library;
import java.util.*;
public class Search implements IOOperation{

   Scanner scanner = new Scanner(System.in);
   public void oper(Database database,User user){
       System.out.print("Enter the name of the book you want to search :
");
       String name = scanner.nextLine();

       int i = database.getBook(name);
       if (i>-1){
           System.out.println("\n" + database.getBook(i).toString()+"\n");
           System.out.print("\n\nPress Any key to continue...");
           scanner.nextLine(); // This will pause the program


       }else{
           System.out.println("Book not found / Book Doesn't exist\n");
           System.out.print("\n\nPress Any key to continue...");
           scanner.nextLine(); // This will pause the program

       }



       user.menu(database, user);


   }
}
```

User.java

```
package Library;
public abstract class User{
   protected String name;
   protected String email;
   protected String phoneNumber;
   protected IOOperation[] operations;

   public User(){};

   public User(String name){
       this.name = name;
   }

   public User(String name, String email, String phoneNumber){
       this.name = name;
       this.email = email;
       this.phoneNumber = phoneNumber;
   }

   public String getName(){
       return this.name;
   }

   public String getEmail(){
       return this.email;
   }

   public String getPhoneNumber(){
       return this.phoneNumber;
   }

   public void setName(String name){
       this.name = name;
   }

   public void setEmail(String email){
       this.email = email;
   }

   abstract  public String toString();

   abstract public void menu(Database database,User user);
```

```
}
```

ViewBook.java

```
package Library;
import java.util.ArrayList;
import java.util.Scanner;
public class ViewBook implements IOOperation{

   Scanner scanner = new Scanner(System.in);

   public void oper(Database database,User user){
       ArrayList<Book> books = database.getAllBooks();
       System.out.println("=".repeat(280));
       printCentered("Name", "Author", "Publisher", "ISBN", "Quantity",
"Price", "Borrowing Copies");
       System.out.println("=".repeat(280));

       for(Book b : books){
           printCentered(b.getTitle(), b.getAuthor(), b.getPublisher(),
b.getIsbn(), String.valueOf(b.getQty()), String.format("RM%.2f",
b.getPrice()), String.valueOf(b.getBrwCoppies()));
       }


       System.out.print("Press Any key to continue...");
       scanner.nextLine();
       user.menu(database, user);


   }
     private void printCentered(String... texts) {
        for (String text : texts) {
            System.out.printf("%20s%20s", "", text);
        }
        System.out.println();
    }

}
```

ViewOrders.java

```
package Library;
import java.util.Scanner;

public class ViewOrders implements IOOperation {

   Scanner s = new Scanner(System.in);

   public void oper(Database database,User user){
       System.out.print("\nEnter book name :  ");
       String bookName = s.nextLine();

       int i = database.getBook(bookName);
       if(i>-1){
           System.out.printf("\n\n%-20s %-20s %-10s %-10s%n", "Book",
"User", "Quantity", "Price");
           for (Order order : database.getAllOrders()) {
               if(order.getBook().getTitle().equals(bookName)){
                   System.out.printf("%-20s %-20s %-10d %-10.2f%n",
                   order.getBook().getTitle(),
                   order.getUser().getName(),
                   order.getQuantity(),
                   order.getPrice());
               }
               System.out.print("\n\nPress Any key to continue...");
               s.nextLine(); // This will pause the program
           }
           System.out.println();
       }else{
           System.out.println("Book doesn't exist in the database\n");
           System.out.print("Press Any key to continue...");
           s.nextLine(); // This will pause the program


       }

       user.menu(database, user);

   }
}
```

SECTION D: TASK DISTRIBUTION

Category	**Task**	**Person-In-Charge**
Coding Implementation	AddBook.java	Afiq Irfan
	Admin.java	Afiq Irfan
	Book.java	Afiq Irfan
	BorrowBook.java	Afiq Irfan
	Borrowing.java	Afiq Irfan
	CalculateFine.java	Danesh Muthu Krisnan
	Database.java	Danesh Muthu Krisnan
	DeleteAllData.java	Aina Nurain
	DeleteBook.java	Anushka Roshni
	Exit.java	Anushka Roshni
	IOOperation.java	Aina Nurain
	Main.java	Danesh Muthu Krisnan
	Members.java	Anushka Roshni
	Order.java	Anushka Roshni
	PlaceOrder.java	Danesh Muthu Krisnan
	ReturnBook.java	Danesh Muthu Krisnan
	Search.java	Aina Nurain
	User.java	Anushka Roshni
	ViewBook.java	Aina Nurain

	ViewOrders.java	Aina Nurain
Others (Diagrams, Reports and Slides)	Introduction	Anushka Roshni
	Flowchart Workflow	Aina Nurain
	OO Concept Explanation	Afiq Irfan / Danesh Muthu Krisnan
	User Manual	Danesh Muthu Krisnan/ Anushka Roshni / Aina Nurain
	Presentation Slide	Afiq Irfan