



Rizvi College of Engineering  
Department of Computer Engineering  
SKL-OOP (JAVA) Mini Project Report  
on

# Library management system

**Submitted**

by

<b>Danish Khan</b>	<b>201P027</b>
<b>Utsav Kuntalwad</b>	<b>201P049</b>
<b>Mayur Kyatham</b>	<b>201P013</b>



University of Mumbai (2020 –20201)

## Abstract

Now a day, through the advancement of modern technology, there are a lot of fast and reliable alternatives for research. However, library still plays a vital role on the students and researcher's life. Library is still considered the most accurate place for information. Undeniably, people especially those who are not having internet connections, and even electricity, rely solely on books. Libraries also supply information not found on World Wide Web. Library still remain the cheapest and the most accessible place for research. Gathering of information still plays a very important role when it comes to gathering of information.

This system is being conceptualized in order for the librarian to access all the books that was barrowed by the student in the school. And we also know that now days in this generation we are used to live with technology and we implement this system.

The Library Management System is an application for assisting a librarian in managing a book library in a university. The system would provide basic set of features to add/update members, add/update books, and manage check in specifications for the systems based on the client's statement of need.

Library management system is a typical management Information system (MIS), its Development include the establishment and maintenance of back-end database and front-end application development aspects. For the former require the establishment of data consistency and integrity of the strong data security and good libraries. As for the latter requires the application fully functional, easy to use and so on.

## *Certificate*

This is to certify that the mini project report entitled “Library management system” has been submitted by Danish Khan, Utsav Kuntalwad, Mayur Kyatham under the guidance of Prof. Shaikh Mohd Ashfaque in partial fulfillment of the requirement for the award of 2<sup>nd</sup> year Engineering in Computer Engineering from University of Mumbai.

Certified By

Prof. Shaikh Mohd Ashfaque

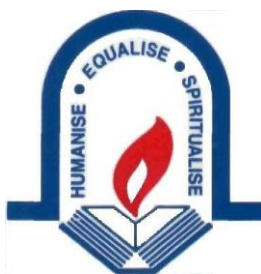
Project Guide

Prof. Shiburaj.Pappu

Head of Department

Dr. Varsha Shah

Principal



Department of Computer Engineering

**Rizvi College of Engineering,**

Off Carter Road, Bandra(W), Mumbai-400050

# Index

<b>Topic</b>	<b>Page no.</b>
Abstract	2
Table of contents	4
Project Description	5 – 8
Class Diagram	9
Software & Hardware Requirements	10
Algorithm	11 – 12
Project Code	13 - 15
Result / Output	16 – 18
Conclusion	19
References	20

## PROJECT DESCRIPTION

Online Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simple. The Online Library Management has been designed to computerize and automate the operations performed over the information about the members, book issues and returns and all other operations. This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced Library Management System in which we can perform all CRUD operations, in addition to advanced search, book issuing, Serialization, and Deserialization. CRUD is an acronym for CREATE, READ, UPDATE and DELETE which are basic functions of persistent storage. CRUD operations can use forms or an interface view to retrieve and return data from a database. Reads the table records based on the primary key within the input parameter. Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

The different concepts used here are classes and objects , collections, array list , searching characters and substring in a string and serialization and deserialization. Then moving to the project we have breakdown our work in to 6 tasks and we have layed our roadmap through this tasks and performed out project so talking about the tasks Create a blueprint to store Book details (getting started ) Implement searching for books based on their name or authors name (addition and deletion of book details ) Emulate admin login session via the command line (searching) Implement the functionality to issue a book for a user (issue books)

Implement serialization and deserialization to store the data in encrypted format in files (serialization and deserialization using files)

Implementing GUI framework using JFrame of applets library

The project will be a typical command line application in Java.

We'll be accepting data (book and user details) to be stored.

We'll be adding the functionality to retrieve book details.

The admin login session will be emulated via the command line. Only during the admin login session can a user be issued a book.

All of the data will be stored in encrypted format in text files to escape the initial volatile nature of the application.

# Task 1 : **Getting started**

Project is going to be a command line application in Java. Like most typical command line applications, it'll contain a switch case that will display a set of options to the user to perform the actions and make necessary queries.

We'll start off by creating an object for storing details of books and also adding some code to our `public static void main(String[] args)`.

1. *Created a class `Book`, for addition of book details.*
2. *Added required fields such as `bookId`, `bookName`, `writerName`, `price` and `quantity`.*
3. *In our main code, we used `Java Collections` to efficiently store the accepted book details*

Here we used concepts such as :

- `Classed and Objects`
  - `Collection in java`
  - `ArrayList in java`
- 

# Task 2 : **Addition and Deletion of book details**

Here we'll be adding the functionality to add/delete book details

- Here we should be able to accept details for a specific book and store the same in an object of type `Book`.
- Every time we add the details of a new book in our already present collection (preferably `ArrayList`), generate a unique `bookId` for the same.
- Accept a query for deleting book details and perform the same. We should check whether the book details requested to be deleted are present or not.
- Here we should have an attribute called `quantity` for each object of type `Book`. This attribute can come in handy when we want to reduce the quantity of a specific book.
- We should be able to check whether the number to be subtracted is actually lesser or equal to the already present quantity or not.

## Task 3 : **Searching**

**We should be able to search for a particular book based on its name or the writer's name**

- Search for book details by bookName. If found, display the details or display Book Absent.
- Search for book details by writerName. If found, display the details or display Writer Absent.
- Add the ability to search for a subsequence of bookName or writerName.
- The search functionality should not be case sensitive.

**The concept used here is : Searching characters and substring in a String**

## Task 4 : **Issue Book**

**In this task, we'll be implementing the functionality for users to issue a book. Only admins will be allowed to issue books to users**

- In order to issue a book, we'll need a user. For the same, create a User entity with fields userId (should be auto-generated), password, role (Admin/User).
- Create a class IssueBook having attributes bookId and userId. The objects of this class will basically be created whenever a user issues a book.
- This should accept various user details from the command line.
- In order to issue a book for a User, emulate a login session for Admin in the command line. On issuing, add the details as an object of type IssueBook to an ArrayList of the same type.

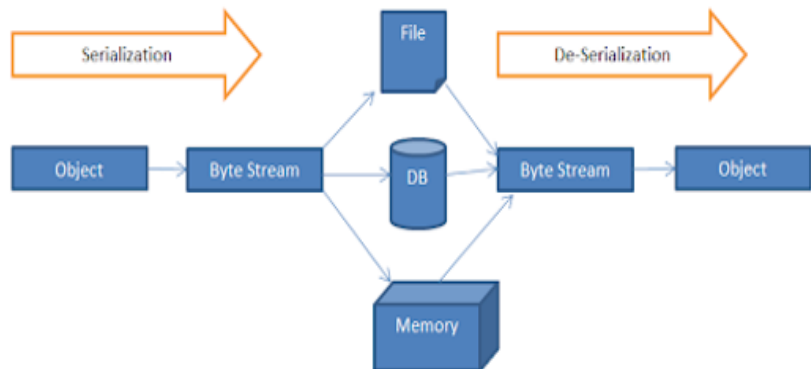


## Task 5 : **Serialization and Deserialization using Files**

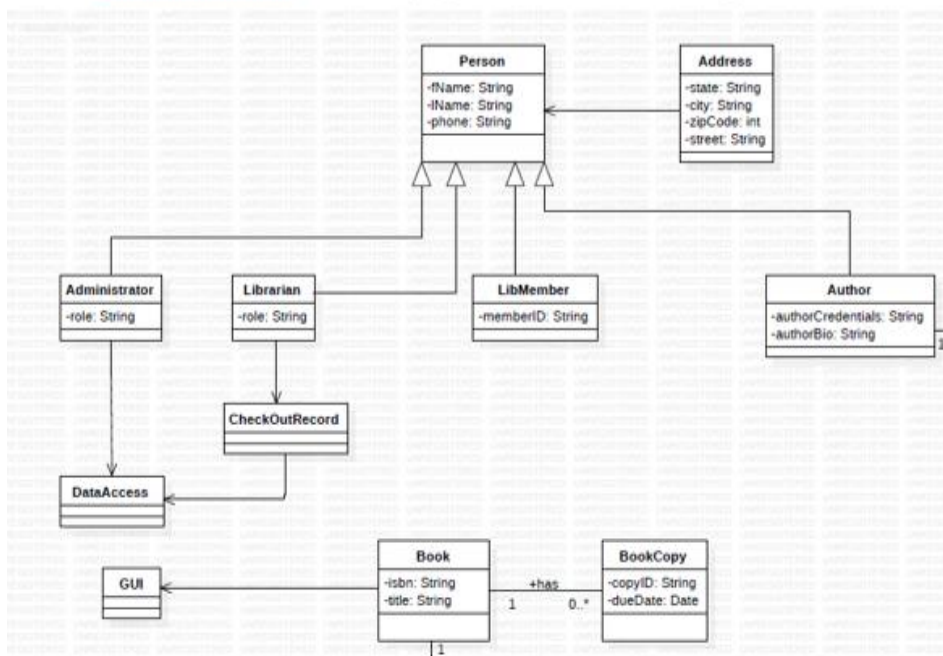
Now our main program should run fine. It should be able to add and remove book details. Add users and admins and enable admin login sessions to issue books for users. But if we stop our program execution, all the data gets deleted. That means that we'll need a way to store our data so that we can access and manipulate it at a later time.

- Create three files: userDetails.txt, bookDetails.txt and issueBook.txt.
- All the data should be stored in these files in an encrypted format.
- Utilize the concept of serialization to store the data to the files.
- Utilize the concept of deserialization to access the data from the files.

The concept we used in this task is :  
**Serialization and Deserialization**



## Task 6: **Implementing GUI framework using JFrame of applets library**



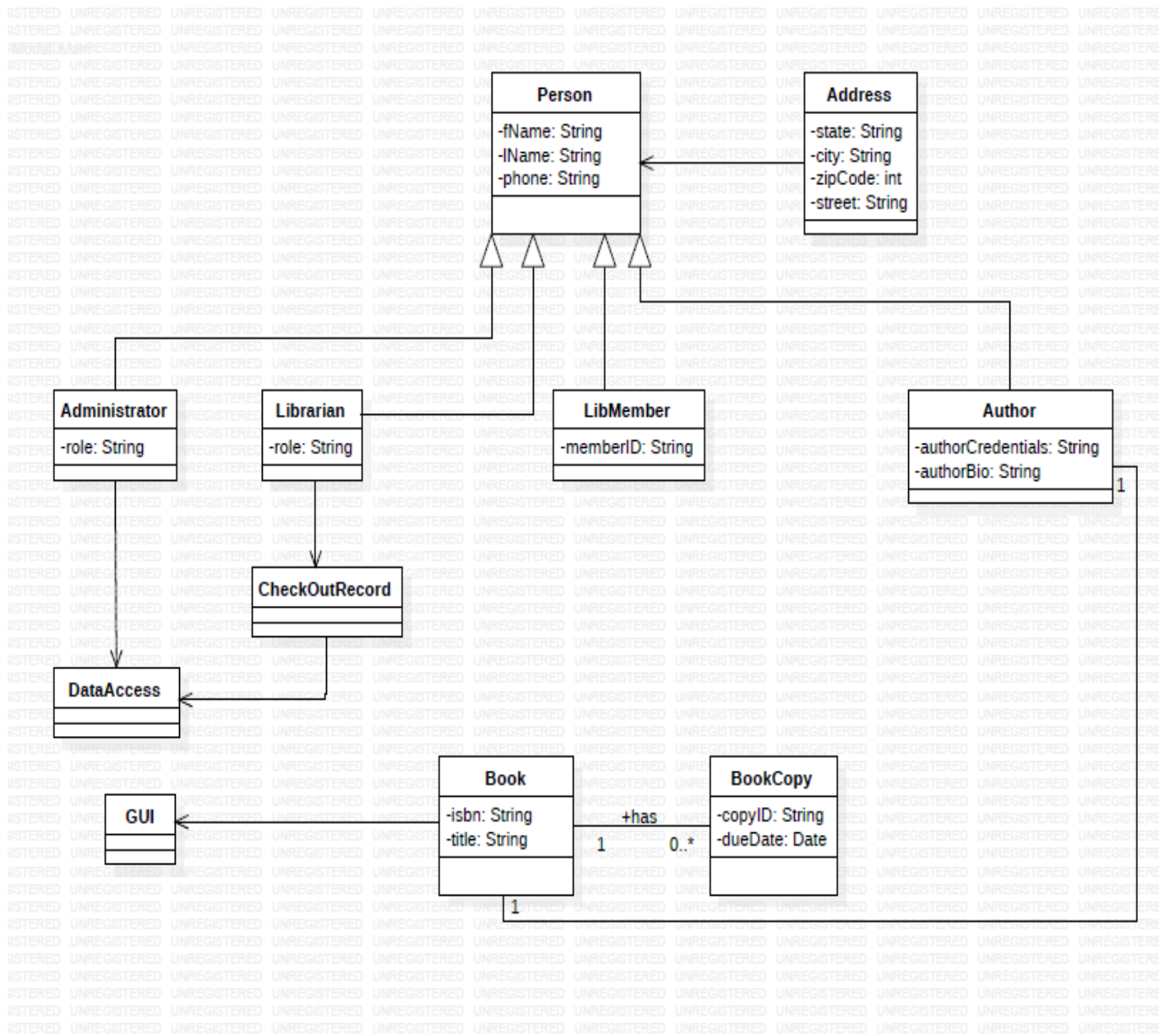
JApplet — a class that enables applets to use Swing components.

JApplet is a subclass of java.applet.Applet, which is covered in the Java Applets trail. JApplet is a top level swing container each swing applet has a root pane. The most noticeable effects of the root panes presence are support for adding a menu bar and the need to use a content pane

Swing components should be created queried and manipulated on the event dispatching thread but browsers don't involve applet "milestone" methods from that thread.



# CLASS DIAGRAM



# SOFTWARE & HARDWARE REQUIREMENTS

## **Required Software:**

Necessary Software's/Libraries are:

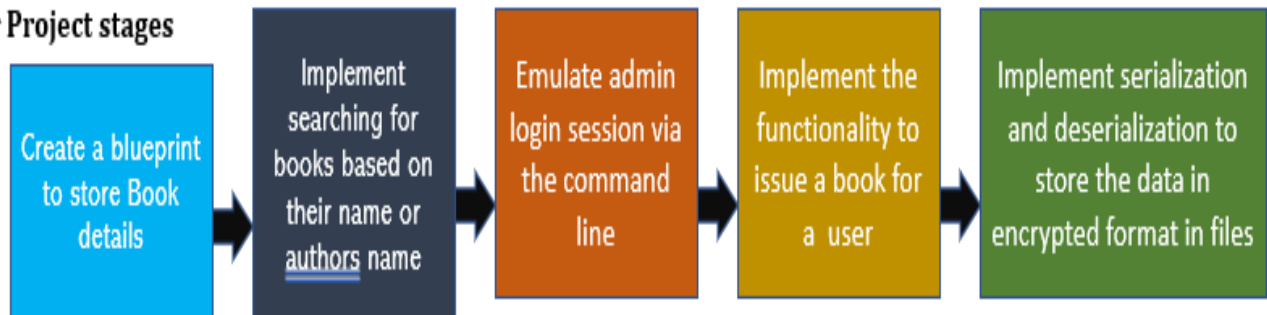
- Software framework library
- Eclipse IDE

## **Required Hardware:**

- Processor (CPU) with 2 gigahertz (GHz) frequency or above
- A minimum of 4 GB of RAM
- A minimum of 50 GB of available space on the hard disk
- Internet Connection Broadband (high-speed) Internet connection with a speed of 4 Mbps or higher
- Windows system with 16 GB RAM

# ALGORITHM

## ❖ Project stages



```
package business;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Main {
    public Main() {
        SystemController controller = new SystemController(); // is a statement that determines whether the other statements will be executed or not.

        try {
            controller.savetoBookList(books);

            controller.getBooks().forEach((e)->{
                System.out.println(e.getIsbn()); // for each book it will get ISBN number
            });

        } catch (Exception e) {
            e.printStackTrace(); // TODO Auto-generated catch block
        }
    }

    // adding predefined members
    List<User> testUsers = Arrays.asList(
        new User("lusam", "Luswata Samuel", "12345", UserType.ADMIN),
        new User("niyo", "Niyoshuti Moses", "12345", UserType.LIBRARAIN),
        new User("zebro", "Zinash Negga Zebro", "12345", UserType.LIBRARAIN),
        new User("kedi", "Kedi Emmanuel", "12345", UserType.BOTH));
}
```

```

List<LibraryMember> listOfMembers = new ArrayList<>(Arrays.asList(
    new LibraryMember("Samuel", "Luswata", "0752816800", "610096"),
    new LibraryMember("Zinash", "Negga", "0773927100", "610095"),
    new LibraryMember("Kedi", "Edgar", "0776621606", "610132"),
    new LibraryMember("Niyonshuti", "Moses", "0782242462", "108886")
));

List<Author> listOfAuthors = new ArrayList<>(Arrays.asList(
    new Author("Samuel", "Luswata", "0752816800", "610096", "He is wonderful"),
    new Author("Zinash", "Negga", "0773927100", "610095", "He is dhdhdhdhddd")
));

Book b1 = new Book("HSUWYET3638", "Intro To JAVA", 21);
Book b2 = new Book("JU274DHFY44", "Intro To ALGO", 7);
Book b3 = new Book("DBVUDU38WNF", "Intro To DS", 21);
Book b4 = new Book("BVYDHEY37DB", "Intro To SOFTWARE", 7);
Book b5 = new Book("E03DBFYEJND", "Intro To PROGRAMMING", 21);
Book b6 = new Book("DGEEICNIENBD", "Intro To MPP", 7);
BookCopy bc1 = new BookCopy("281736", b1);
BookCopy bc2 = new BookCopy("46764", b1);
BookCopy bc3 = new BookCopy("45345", b1);
BookCopy bc4 = new BookCopy("3464", b2);
BookCopy bc5 = new BookCopy("34677", b2);
BookCopy bc6 = new BookCopy("78964", b2);
BookCopy bc7 = new BookCopy("3346", b3);
BookCopy bc8 = new BookCopy("23467", b3);
BookCopy bc9 = new BookCopy("34689", b3);
BookCopy bc10 = new BookCopy("372824", b4);
BookCopy bc11 = new BookCopy("35678", b4);
BookCopy bc12 = new BookCopy("98765", b4);
BookCopy bc13 = new BookCopy("346742", b5);

BookCopy bc4 = new BookCopy("3464", b2);
BookCopy bc5 = new BookCopy("34677", b2);
BookCopy bc6 = new BookCopy("78964", b2);
BookCopy bc7 = new BookCopy("3346", b3);
BookCopy bc8 = new BookCopy("23467", b3);
BookCopy bc9 = new BookCopy("34689", b3);
BookCopy bc10 = new BookCopy("372824", b4);
BookCopy bc11 = new BookCopy("35678", b4);
BookCopy bc12 = new BookCopy("98765", b4);
BookCopy bc13 = new BookCopy("346742", b5);
BookCopy bc14 = new BookCopy("35790", b5);
BookCopy bc15 = new BookCopy("96434", b5);
BookCopy bc16 = new BookCopy("46764", b6);
BookCopy bc17 = new BookCopy("345790", b6);
BookCopy bc18 = new BookCopy("64223", b6);

List<Book> books = Arrays.asList(b1, b2, b3, b4, b5);

public static void main(String[] args) {
    new Main();
}
}

```

## PROJECT CODE

<https://github.com/danishkhanbx/Library-Management-System>

---

```
package business;

import java.io.Serializable;

// adding address of given person and serializing it
public class Address implements Serializable {

    private static final long serialVersionUID = 5728643884736556443L;
    private String state;
    private String city;
    private String zipCode;
    private String street;

    public Address(String state, String city, String street, String zipCode){
        setState(state);
        setCity(city);
        setStreet(street);
        setZipCode(zipCode);
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getZipCode() {
        return zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(street+"\n"+ city + ", " + state + ", " + zipCode);
        return sb.toString();
    }

}
```

---

```
package business;

import java.io.Serializable;
import java.time.LocalDate;

// entering details before taking the book
public class CheckoutRecordEntry implements Serializable {

    private static final long serialVersionUID = 8447984829265265909L;
    private BookCopy bookCopy;
    private LocalDate checkoutDate;
    private LocalDate dueDate;
    private LibraryMember libMember;

    public CheckoutRecordEntry(BookCopy bookCopy, LibraryMember libMember){
        this.bookCopy = bookCopy;
        this.checkoutDate = LocalDate.now();
        this.dueDate = checkoutDate.plusDays(bookCopy.getBook().getMaxDays());
        this.libMember = libMember;
        libMember.addCheckout(this);
    }

    public String toString() {
        return "MemberID: " + libMember.getMemeberID() +
            "\n" +
            "CopyID: " + bookCopy.getCopyID() +
            "\n" +
            "Book ISBN: " + bookCopy.getBook().getIsbn() +
            "\n" +
            "CheckOutDate: " + checkoutDate +
            "\n" +
            "DueDate: " + dueDate +
            "\n";
    }
}
```

---

```

package dataaccess;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.HashMap;
import java.util.List;

import business.Author;
import business.Book;
import business.LibraryMember;
import business.User;

// accessing all the Data and storing it in dataaccess.storage package respective to theirs files
public class DataAccessFacade implements DataAccess {

    public static final String OUTPUT_DIR = System.getProperty("user.dir")
        + "\\src\\dataaccess\\storage";
    public static final String DATE_PATTERN = "MM/dd/yyyy";

    public static void loadUserMap(List<User> userList) {
        HashMap<String, User> users = new HashMap<>();
        userList.forEach((user) -> {
            users.put(user.getUserID() ,user);
        });
        saveToStorage(StorageType.USERS, users);
    }

    public static void loadAuthorsMap(List<Author> authorsList) {
        HashMap<String, Author> authors = new HashMap<>();
        authorsList.forEach((author) -> {
            authors.put(author.getAuthorCredentials() ,author);
        });
        saveToStorage(StorageType.AUTHORS, authors);
    }

    public static void loadMembersMap(List<LibraryMember> memberList) {
        HashMap<String, LibraryMember> members = new HashMap<>();
        memberList.forEach((member) -> {
            members.put(member.getMemeberID() , member);
        });
        saveToStorage(StorageType.MEMBERS, members);
    }

    public static void loadBooksMap(List<Book> bookList) {
        HashMap<String, Book> books = new HashMap<>();
        bookList.forEach((book) -> {

```



## RESULT / OUTPUT

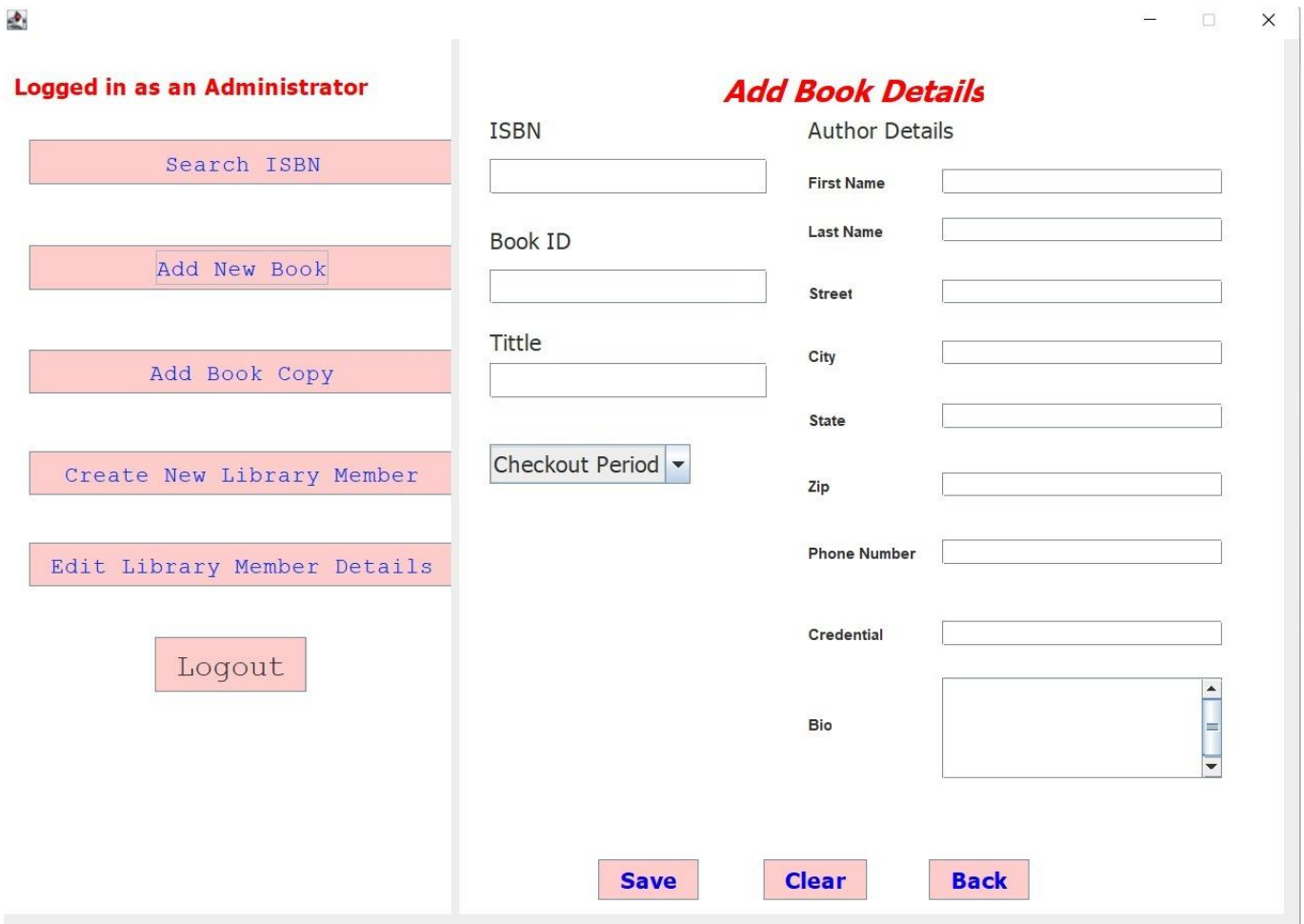


A login window with a title bar containing a small icon, a minus button, a maximize button, and a close button. The window has a light gray background. It contains two text input fields: one for 'Username' and one for 'Password'. Below these fields are two buttons: 'Login' and 'Exit'.

Username

Password

Login Exit



An 'Add Book Details' window with a title bar containing a small icon, a minus button, a maximize button, and a close button. The window has a light gray background. On the left side, there is a sidebar with a red header 'Logged in as an Administrator' and five buttons: 'Search ISBN', 'Add New Book', 'Add Book Copy', 'Create New Library Member', and 'Edit Library Member Details'. At the bottom of the sidebar is a 'Logout' button. The main area has a red header 'Add Book Details'. It contains several input fields: 'ISBN', 'Book ID', 'Title', 'First Name', 'Last Name', 'Street', 'City', 'State', 'Zip', 'Phone Number', 'Credential', and 'Bio'. There is also a 'Checkout Period' dropdown menu. At the bottom of the main area are three buttons: 'Save', 'Clear', and 'Back'.

**Logged in as an Administrator**

Search ISBN

Add New Book

Add Book Copy

Create New Library Member

Edit Library Member Details

Logout

**Add Book Details**

ISBN

Book ID

Title

Checkout Period

Author Details

First Name

Last Name

Street

City

State

Zip

Phone Number

Credential

Bio

Save Clear Back



**Logged in as an Administrator**

Search ISBN

Add New Book

Add Book Copy

Create New Library Member

Edit Library Member Details


Logout

Enter ISBN

Search

Clear

Back



**Logged in as an Administrator**

Search ISBN

Add New Book

Add Book Copy

Create New Library Member

Edit Library Member Details

Logout

Enter ISBN

Book ID

Add

Clear

Back

Logged in as an Administrator

Search ISBN

Add New Book

Add Book Copy

Create New Library Member

Edit Library Member Details

Logout

Enter Library Member Details

Member ID

First Name

Last Name

Street

City

State

zip

Telephone Number

Save

Clear

Back

Logged in as an Administrator

Search ISBN

Add New Book

Add Book Copy

Create New Library Member

Edit Library Member Details

Logout

Enter Member ID

Search

Clear

Back

## CONCLUSION

A Library Management System Project in Java is one of the various systems developed that has numerous functionalities that meet the current requirements of the present-day library system. To enhance it, we can add features like RFID, SMS to remind users of the return date, and others. Java has many features that can be explored to create such wonderful programs.

After we have completed the project we are sure the problems in the existing system would overcome. The "LIBRARY MANAGEMENT SYSTEM" process made computerized to reduce human errors and to increase the efficiency. The main focus of this project is to lessen human efforts. The maintenance of the records is made efficient, as all the records are stored in the ACCESS database, through which data can be retrieved easily. The navigation control is provided in all the forms to navigate through the large amount of records. If the numbers of records are very large then user has to just type in the search string and user gets the results immediately. The editing is also made simpler. The user has to just type in the required field and press the update button to update the desired field. The Books and Students are given a particular unique id no. So that they can be accessed correctly and without errors. Our main aim of the project is to get the correct information about a particular student and books available in the library. The problems, which existed in the earlier system, have been removed to a large extent. And it is expected that this project will go a long way in satisfying users requirements. The computerization of the Library Management will not only improves the efficiency but will also reduce human stress thereby indirectly improving human recourses.

## REFERENCES

1. <https://www.mitrais.com/news-updates/step-by-step-making-a-simple-crud-application-using-java-servlet-jsp/>
2. <https://www.edureka.co/blog/library-management-system-project-in-java>
3. <https://www.javatpoint.com/serialization-in-java>
4. <https://www.geeksforgeeks.org/serialization-in-java/>
5. <https://snyk.io/blog/serialization-and-deserialization-in-java/>
6. <https://www.geeksforgeeks.org/classes-objects-java/>
7. <https://www.geeksforgeeks.org/collections-in-java-2/>
8. <https://www.geeksforgeeks.org/arraylist-in-java/>
9. <https://www.geeksforgeeks.org/searching-for-characters-and-substring-in-a-string-in-java/>