NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

OBJECT-ORIENTED PROGRAMMING
CS-212
SEMESTER PROJECT REPORT

Instructor:

Dr. Ahsan Saadat

Submitted by:

S. No.	Name	Reg. No.
1	Muhammad Bilal	389994
2	Abdul Arham	374696
3	Muhammad Ashhub Ali	380078

Table of Contents

LIBRARY MANAGEMENT SYSTEM

Introduction

Concept

Library automation means the unmanned electronic software that is equiped with the state-of-the-art technological infrastructure to manage library house-keeping functions and user services. It is an enterprise resource planning system, a **specialized management software**, which includes database management, data processing, storage, networking, etc. that helps keep track of several library functions.

These include members' profile and subscription, keeping a ledger of books to check the issues and balance, classifying and cataloguing materials, managing orders, unpaid and paid bills, invoicing, etc. It also keeps a database where new books are acquired by the library keeping in mind the current demand or requests from patrons. A library management system automatically keeps a record of the books and the due dates in relation to the borrowed books. Each member has a specific identification code that the software may maintain for monitoring.

This novel software has enabled libraries to connect and access different library collections and structural changes in the preservation and storage of materials. The internet has resulted in a paradigm shift in the functioning of libraries. With the help of the internet, members can login to their online library accounts to pre-order books and access the virtual database.

Our Project intends to offer a dynamic and complete framework that exhibits and fulfils the minimum requirements that are needed in such a system, as well as to impart features that not only add to the aesthetic of the project, but also increase its usability in a business paradigm.

Overview of LMS:

Our Library Management System provides a platform that can be implemented in a relevant setting i.e. any library, to allow the digitization of the library, and ensure ease of connectivity. It imparts features that are extremely useful. The aforementioned features are essential in any MS. We have particularly chosen to introduce and incorporate dynamics that elevate the attractiveness of our system, enticing any interested party to cash in on such a useful system.

Our system can be divided into three major components that work in unison i.e. an interface for the Administration aka the master control, a UI for the employees with unique functionalities and a separate UI for the library members.

The whole Library Management System is controlled by the Master Accounts that are created in the Admin block of the system. Any Admin account has the prowess to:

i. Add new employee accounts
ii. Modify employee accounts
iii. Delete employee accounts, in case they get fired etc.
iv. Add member accounts
v. Modify member accounts
vi. Delete member accounts
vii. Track-record of both member and employee accounts.

In this way, the "overseer" of the whole system is the Admin, and in a real-life scenario, this part of the system would be handed over to the Manager of the library. In this way, the manager would be able to keep in check of all members and employees, including the member's attendance and the issuance of books by the members.

As mentioned above, only the Admin account has the power to create new accounts. Thus, a student wishing to become a member of the library can only do so after a formal request to the Librarian. The Librarian would receive the student's credentials, and on its basis, create a unique member account that will be allocated to an address in the database, and all its information will be stored there. The details of the member account would then be shared by the Librarian with the student, after which the student can access their account, and issue books amongst other capabilities.

In addition to this, the LMS contains a plethora of features. These, as well as the basic implementation of our Library Management System, are discussed in the next section.

Basic Implementations & Features:

1. Database

In today's world, almost everything is connected digitally. So it would be very effective for libraries to use a management system to keep a record of all sorts of information like books, members, and staff. A key component of our project would be to use databases for real-time storage of all this data. Using MySQL, the data will be stored in the form of tables and blocks in a database and accessed through the JDBC (Java Database Connectivity) Library.

In general, user information such as Name, ID, Books Issued, Current Status, fines/ charges will be stored. Whereas, Employee Databases will include employee types (Librarian, Assistant, Technician, Janitors, etc.), their attendance, performance, and rank. For books, the number of copies, damaged items (if any), current availability, new arrivals, etc., will be saved in these databases.

2. Graphical User Interface

We intend to use and implement Java Swing Library for the outlook of our System. The interface created will allow the users (both the staff and the members) to interact freely with the application. We will set up navigation bars to access each page/ window. These pages or windows include but are not limited to Homepage, Admin Login, Library Stock, User Login, and Comments/ FAQs Section. To make it more appealing, we will add simple animations such as those enabled on hovering, etc.

3. Admin Controls

A core component of our application will be Serial Controls for the Administration (Librarian, in general). These controls will enable the Admin to cancel, renew, provide new membership, or fine a customer. The Librarian will have access to all the Library Stock (inflow as well as outflow).

4. Dynamic Search

This feature will allow the users to search for a book by entering either the Author Name, Book Title, Genre, or ISBN Number.

5. Request a Book

If a certain book is not available in the Library Stock, the user can request for new books, which will be conveyed to the Management to order in the future. The priority of each book will be proportional to its demand by the users.

6. Book Suggestion

The originality and uniqueness of a product is what sells it. To cater to the masses, the system must be able to adapt and show content based upon a particular user's preference. Therefore, our application will suggest books to each user depending on the genre they frequently issue, or books read by other users with similar tastes. This will ensure that redundancy and the static nature of the system is reduced, as the application will be personalized according to the likes and dislikes of the user.

7. Upload Profile Picture

Our application serves to be a place for students to be able to take advantage of the advancements of technology. At the same time, we intend to make it a place where the user can modify certain aspects to their own likability. Thus, the Library Management System will also enable the user to upload their profile picture to their account. The picture will then be displayed along with other account details.

Need for a Library Management System

The 21st century is a period of digitization, where any company or organization that fails to adapt to the constantly changing atmosphere of the world would most certainly be left behind. Automation and Machine-learning algorithms have made most works and jobs that were previously considered labor-intensive, completely human-free. In today's world, if any entity fails to familiarize itself with the current advancements in software and hardware, as well as being unable to identify, implement and take advantage of the latest devices available, it would result in a guaranteed downfall from the start. Thus, the use of software to implement operations that were previously considered to be solely hand-based, is not a gimmick anymore, but rather a necessity with these changing times. With that being said, a Library, home to a plethora of books, novels, research papers and articles, is one such place that cannot prosper and be maintained without the implementation of a **Library Management System.** This system would most certainly hasten most, if not all, processes that previously took due time, in the context of the management of a library.

Most libraries in Pakistan still use outdated software that has limited functionality. They're using the same systems that were introduced back when the libraries were inaugurated. Although this isn't true for all libraries, there are a countable number in Pakistan, and thus the aforementioned would certainly benefit from our endeavor of developing a state-of-the-art Library Management System, with peak functionality and usage.

There are countless problems that are faced by outdated LMS(s). Some noteworthy are:

1. Hierarchical distribution of employees is not done i.e., every employee on the system usually has the same access controls and modification capabilities.
2. Information about all the books in the library is stored in a separate database, rarely linked to the management system.
3. The System cannot host a large number of clients, and thus lag issues are common.
4. Employee System and Student System is usually made separately, without any integration. Communication between the Librarian and the members of the library through the system is thus a hassle.
5. The system on the member end is usually obsolete, with only static pages on book information. The members aren't given any capability to issue books through the system.

Thus, these among other reasons are why an advanced and modern Library Management System is a need of the hour. The LMS that we have so intricately constructed intends to fix and improve upon most, if not all, the problems and

inadequacies faced by outdated software. Our Management System intends to provide a framework that is not only user-friendly but also has maximum functionality. Furthermore, we intend to make it available at a market-competitive and economical rate, so that a large pool of companies can benefit from our efforts.

Scope

The aim was to build a complete system that simulates the workings and logistics of a generic library. Our program will have a plethora of dimensions which include, but are not limited to:

1. An Employee Management System
2. A Library Membership System
3. Library Stock and Book Collection System

To start, we intended to make our project such that it intricately mimics the daily occurrences inside a library, using samples and templates. Our goal is to make it workable, such that it can be implemented and used by any public or private library in the region, e.g., NUST's Main Library.

We made our best effort to broaden our horizons, and after extensive research, we were able to conclude that the following missing features are a must in an LMS:

1. Admin Master Control
2. Employee Management Control
3. Member/Student Control
4. Dynamic Search
5. Issue a Book feature

Thus, we were able to add all these necessary features into our System. Some supplementary features that we intended to add, and could possibly add in the future, are:

1. Chat-box feature
2. Online Employee Support for members
3. Direct E-book access.

However, it must be noted that our System has indeed implemented not only the basic requirements, but also most features that cater to the users of the System in such a way that their experience of it is highly productive and fruitful. The aforementioned "missed" features are merely "gimmicks" that serve less functionality. However, if given time, we would certainly implement these as well.

Code Overview

1. File Handling and Data Storage

Just like any other management system, the storage of data and records to keep track of all the transactions and processes is highly essential. Thus, the storage of data and records in our Library Management System is a key component of our project, and thus was a point of focus as well, that took a good percentage of our time to carve it to perfection.

The options that we had available to us were the use of either files or databases. For a large-scale project such as ours, the use of text files is cumbersome as it would require the creation and the manipulation, and overriding of multiple files all at once, which would put burden on the processor and the system, and thus, would result in a slower execution of our software. Therefore, to increase efficiency, we have decided to employ the use of a database. Specifically, we employed the use of MySQL to make our database.

Overview of MySQL:

The MySQL server provides a database management system with querying and connectivity capabilities, as well as the ability to have excellent data structure and integration with many different platforms. It can handle large databases reliably and quickly in high-demanding production environments. The MySQL server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases.

The MySQL server works in a client and server system. This system includes a multiple-threaded SQL server that supports varied backends, different client programs and libraries, administrative tools, and many application programming interfaces (API)s.

Available Options

i. Sequential Text Files
ii. Serialized Files
iii. Random-Access Files
iv. CSV Files
v. Databases

Preferred Option

MySQL Database

2. Graphical User Interface API

The use of a GUI is essential to our project, as it makes up the majority of our front-end code. Our project aims to provide a user-friendly and productive experience for both the students and the employees; therefore, the use of a seamless Graphical User Interface is required. A GUI ensures that the end-product appears aesthetically pleasing to the User, as well as allowing easier command inputs that the user is accustomed to, instead of e.g. command-line arguments. In this way, the development of the GUI is key to the success and popularity of our product.

There are three primary attributes of the GUI environment that can be attractive to various users.

The typical attributes of the GUI-based system are:

i. **The use of a pointing device, such as a mouse; and the use of this device to point and pick from a menu or list, or to "click" (select or de-select) items** in a dialog box. The most apparent advantage of this attribute is less typing. Perhaps more important is a learning advantage that comes from having the user options displayed in the menus and scroll bars. The menus, lists, and dialog boxes, usually associated with the mouse approach may provide a clearer indication of the choices available to the user than other inputting techniques.

ii. **The viewing of project schedule outputs in graphical formats.** Typically, project management software users will wish to see the project data in both tabular (spreadsheet) formats and graphically. GUIs can provide a vastly superior presentation of the graphic formats. In addition, some GUI-based systems also offer user control of font types and sizing, which can help to improve textual presentations, as well.

iii. **The ability to input tasks node-by-node and to link (denote precedence) the tasks with the pointing device.**

Available Options

1. Abstract Window Toolkit
2. Java Swing
3. JavaFX
4. Android Studio

Preferred Option

Java Swing

Use of OOP concepts

1. Inheritance in Java

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance is a core concept that we have applied **extensively** throughout our project. Without the use of inheritance, our project would not be able to be completed in the time allocated to us. This concept ensured that the need to rewrite code was eliminated, as it allowed us to reuse code snippets further down in our hierarchy wherever required. Any time that we needed to reduce redundancy, we would simply extend a subclass from its superclass and get the job done much more quickly. In our project, there are more than 6 separate hierarchies forming more than 50 classes. One such hierarchy is shown below:

```
public class Person {
    private Image profilePic;
    private String name, username, password, gender, email, phone;
    private int age;
```

```
public final class Employee extends Person{
    private String jobTitle;
```

```
public final class Member extends Person{
    private String book1, book2, book3, book1_genre, book2_genre, book3_genre;
```

Figure 1: Inheritance In OOP

In the same way, multiple hierarchies have been used by us to efficiently execute the code, and make it concise. It must be noted that the rest of the hierarchies are much longer and extensive, and the example shown is just for reference.

2. Method overriding

Method overriding occurs when a method in the subclass has the same name and signature as a method in its superclass.

In such cases, the version of the called overridden method will always refer to the method defined by the subclass.

As our code consists of multiple hierarchies with a highly distributed tree structure, the use of method overriding has been virtually exhausted. In every other subclass, we have made the use of this phenomenon. An example of it is shown below:

```
public class LabelDateFormatter  extends JFormattedTextField.AbstractFormatter {
    private String datePatern = "dd/MM/yyyy";

    private SimpleDateFormat dateFormatter = new SimpleDateFormat(datePatern);

    @Override
    public Object stringToValue(String text) throws ParseException {
        return dateFormatter.parseObject(text);
    }

    @Override
    public String valueToString(Object value) throws ParseException {
        if (value != null) {
            Calendar cal = (Calendar) value;
            return dateFormatter.format(cal.getTime());
        }

        return "";
    }
}
```

Figure 2: Method Overriding

This particular example overrides the paint() method i.e. the method of JPanel, to give it a specific usage.

In a similar way, this concept has been implemented in a number of places in our project.

3. Abstract classes, methods and polymorphism

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic.

In our project, we have used the concept of abstract classes as well as polymorphism in various places. This use has enabled us to achieve a higher level of abstraction in our project, and let us let the compiler decide upon runtime which methods to call. This has made our job as Java developers much easier. An example of an abstract class that we have implemented in our code is given below:

CheckCredentials

The first abstract class that we have created is CheckCredentials. It has two String data fields, i.e., username and password. It has its respective setters and getters, and finally it has an abstract method username_in_database() which is a public method that returns a Boolean value. The code snippet for it is shown below:

This Abstract class is extended by two concrete subclasses i.e. CheckMemberCredentials and CheckEmployeeCredentials. The former concrete class's implemented method checks the Member table while the latter checks the Employee's table. As these are the same methods with slightly different functionality, an abstract class makes the code less redundant. Their code snippet is shown below:

```
import ...

public class CheckMemberCredentials extends CheckCredentials{

    CheckMemberCredentials(String username) { this.setUsername(username); }

    public boolean username_in_database(){
        boolean username_in_db = false;
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/library?useSSL=false",
            Statement stm = con.createStatement();
            ResultSet rs = stm.executeQuery( sql: "select * from members");
            while (rs.next()){
                if (rs.getString( columnIndex: 1).equals(this.getUsername())){
                    username_in_db = true;
                    this.setPassword(rs.getString( columnIndex: 2));
                }
            }
        }catch (Exception e){
            e.printStackTrace();
        }
        return username_in_db;
    }

}
```

Figure 3: Extension of Abstract Class CheckCredentials

```java
package com;

import ...

public class CheckEmployeeCredentials extends CheckCredentials{
    CheckEmployeeCredentials(String username) { this.setUsername(username); }
    @Override
    public boolean username_in_database() {
        boolean username_in_db = false;
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/library?useSSL=false",
            Statement stm = con.createStatement();
            ResultSet rs = stm.executeQuery( sql: "select * from employees");
            while (rs.next()){
                if (rs.getString( columnIndex: 1).equals(this.getUsername())){
                    username_in_db = true;
                    this.setPassword(rs.getString( columnIndex: 3));
                }
            }
        }catch (Exception e){
            e.printStackTrace();
        }
        return username_in_db;
    }
}
```

Figure 4: Another Class Implementing methods in the Abstract CheckCredentials Class

The same two classes' objects are created in AdminMain and are called polymorphically. Both objects are stored in a variable of their abstract superclass and their method is called and is polymorphically determined. This is shown below:

```java
CheckCredentials checkMemberCredentials = new CheckMemberCredentials(username_login_textfield.getText());
CheckCredentials checkEmployeeCredentials = new CheckEmployeeCredentials(username_login_textfield.getText()
```

```java
boolean it_is_member = checkMemberCredentials.username_in_database();
if (it_is_member){
    if (String.valueOf(password_login_textfield.getPassword()).equals(checkMemberCr
        System.out.println("It is a member.");
    }else{
        new IncorrectPassword();
    }
}else{
    boolean it_is_employee = checkEmployeeCredentials.username_in_database();
    if (it_is_employee){
```

Figure 5: Polymorphism in Abstract Classes

4. Final methods and classes

Final methods and classes do not allow further inheritance. This allows us to make our code more secure from external interference. Among these Final Classes and Methods, few of the examples are:

```
public final class Employee extends Person{
```

```
public final class Member extends Person{
```

Figure 6: Example of Final Classes

5. Interfaces

An interface is an abstract type used to specify the behavior of a class. An interface cannot be instantiated, but instead can only be implemented by classes. Essentially, an interface forms a contract with the classes that implements the said interface, making it obligatory for these classes to override the methods named in the interface. All methods declared in the interface are abstract. Similarly, all variables declared are also static and final. In this way, an interface lets us connect two or more unrelated classes.

We have employed the use of interfaces as well in our project. An example of an interface that we have implemented is given below:

BLOBConverter

The interface that we have implemented is called BlobConvetter. It is a simple interface that easily enhances our code reusability. BlobConverter is an interface with a single abstract method convert_file_to_blob(File file, String username).

```
public interface BlobConverter {
    void convert_file_to_blob(File file, String username);
}
```

Figure 7: BlobConverter Interface

It is being implemented by the classes UpdateMemberProfilePic and UpdateEmployeeProfilePic.

```
public class UpdateMemberProfilePic extends Component implements BlobConverter{
```

```
public class UpdateEmployeeProfilePic extends Component implements BlobConverter{
```

Figure 8: Implementation of BlobConverter Interface

6. Exception Handling

As exception handling is a core concept, it is quite obvious that any self-sufficient project cannot be presented without compensating for its weak points. Exception handling takes these weak points and handles them.

Our project, like any other worthy project, is filled to the brim with Exception handling. Every corner of the code has been accounted for, and no err of any kind is shown. Some examples of the usage of Exception Handling is shown below:

```
public void searchEmployees(String input) {
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/library?useSSL=false",
                user: "root",  password: "abdularham123");
        Statement stm = con.createStatement();
        ResultSet rs = stm.executeQuery( sql: "select * from employees");
        employees = new ArrayList<>();
        while (rs.next()){
            if(rs.getString( columnIndex: 1).toLowerCase().contains(input.toLowerCase()) ||
                    rs.getString( columnIndex: 2).toLowerCase().contains(input.toLowerCase())){
                String username = rs.getString( columnIndex: 1);
                String name = rs.getString( columnIndex: 2);
                String password = rs.getString( columnIndex: 3);
                String gender = rs.getString( columnIndex: 4);
                int age = rs.getInt( columnIndex: 5);
                String jobTitle = rs.getString( columnIndex: 6);
                String email = rs.getString( columnIndex: 7);
                String phoneNumber = rs.getString( columnIndex: 8);
                ImageIcon profile = new ImageIcon(rs.getBlob( columnIndex: 9).getBytes( pos: 1l,
                        (int)rs.getBlob( columnIndex: 3).length()));
                Image profilePic = profile.getImage().getScaledInstance( width: 120,  height: 170, Image.SCALE_DEFAULT);

                employees.add(new Employee(profilePic, name, username, password, gender, email, phoneNumber, age,
                        jobTitle));
            }
        }
        con.close();
```

Figure 9: Exception Hnadling when accessing Databases

7. Flow Control

Flow Control determines the order in which the program is executed. Benefitting from OOP and various decision-making statements, we can access any part of the program as desired by us. One such example is that of the MoreOptionsPanel which provides the user with various buttons to navigate around.

```
public class MoreOptionsPanel extends JPanel implements ActionListener {
    JButton home, employees, members, catalogue, logout, login, member_home;
```

Figure 10: MoreOptionsPanel Class

The panel is visible by pressing the optionsButton at the top left corner of each panel.

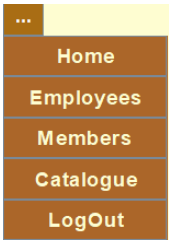

Figure 11: Available Options for Admin Account

Other examples include but are not limited to if-else statements. In the snippet below, every time the text in the search bar is changed, the display panel is refreshed and new panels are added according to the entered text. It can also be noted that two different types of panels are added for both the Member Accounts and Admin Accounts using the integer control.

```
public void updateSearch() throws IOException {
    bookDisplay.removeAll();
    bookDisplay.updateUI();
    if (!searchbar.getText().equals("")){
        DynamicSearch data = new DynamicSearch();
        data.fetch_books(searchbar.getText());
        for (Book book : data.books) {
            if (control == 0) {
                AdminBookPanel bookPanel = new AdminBookPanel(book);
                bookPanel.setAlignmentX(CENTER_ALIGNMENT);
                bookDisplay.add(Box.createRigidArea(new Dimension( width: 400,  height: 20)));
                bookDisplay.add(bookPanel);
            } else {
                BookPanel bookPanel = new BookPanel(book);
                bookPanel.setAlignmentX(CENTER_ALIGNMENT);
                bookDisplay.add(Box.createRigidArea(new Dimension( width: 400,  height: 20)));
                bookDisplay.add(bookPanel);
            }
```

Figure 12: Using If-Else Statement for Flow Control in Dynamic Search

Graphical User Interface

The GUI, as mentioned above, that we have used is **Swing**. Its concepts and libraries are thoroughly used, and its full capabilities will be explained in our presentation. The Color Palette that we have decided to use is shown below:

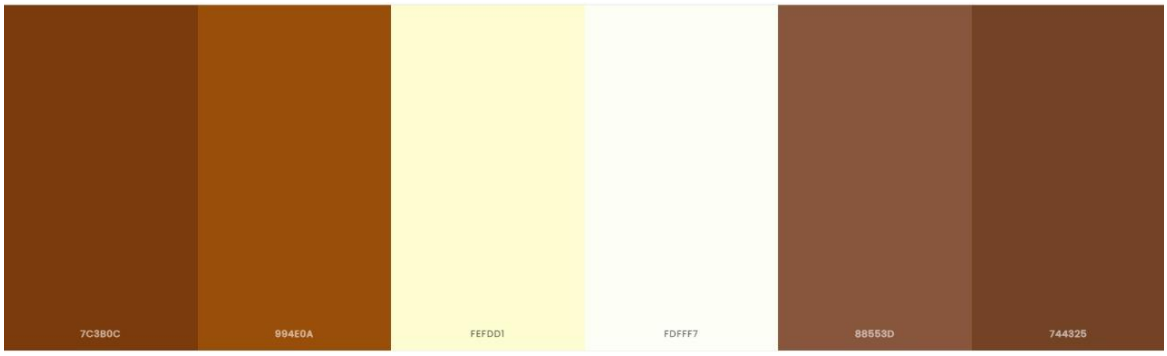

Figure 13: Color Pallet Used

The color palette shown above is akin to a natural setting in a library. The different shades of brown simulate a library environment in a digital setting. This makes the

user experience feel much more natural, and makes it seem as if they are in an actual library.

We created our own GUI Components that extend from the already existing components. For example, RoundedPanel extends from the JPanel class and is used to create panels that have rounded corner. It takes the Layout, corner radius, and Color as Constructor parameter to avoid assigning these traits to the Components formed individually.

```
public class RoundedPanel extends JPanel
{
    private Color backgroundColor;
    private int cornerRadius = 15;

    public RoundedPanel(LayoutManager layout, int radius) {
        super(layout);
        cornerRadius = radius;
    }

    public RoundedPanel(LayoutManager layout, int radius, Color bgColor) {
        super(layout);
        cornerRadius = radius;
        backgroundColor = bgColor;
    }

    public RoundedPanel(int radius) {
        super();
        cornerRadius = radius;
    }

    public RoundedPanel(int radius, Color bgColor) {
        super();
        cornerRadius = radius;
        backgroundColor = bgColor;
    }
```

Figure 14: RoundedPanel Class

It overrides the paintComponent() method to draw these curved corners according to the desired radius.

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Dimension arcs = new Dimension(cornerRadius, cornerRadius);
    int width = getWidth();
    int height = getHeight();
    Graphics2D graphics = (Graphics2D) g;
    graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

    //Draws the rounded panel with borders.
    if (backgroundColor != null) {
        graphics.setColor(backgroundColor);
    } else {
        graphics.setColor(getBackground());
    }
    graphics.fillRoundRect( x: 0,  y: 0,  width: width-1,  height: height-1, arcs.width, arcs.height); //pa
    graphics.setColor(getForeground());
    graphics.drawRoundRect( x: 0,  y: 0,  width: width-1,  height: height-1, arcs.width, arcs.height); //pa
}
```

Figure 15: Overriden paintComponent() Method

File Manipulation and Databases

For data storage, we have employed the use of a database. In particular, we have opted for the use of MySQL.

The MySQL server provides a database management system with querying and connectivity capabilities, as well as the ability to have excellent data structure and integration with many different platforms. It can handle large databases reliably and quickly in high-demanding production environments. The MySQL server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases.

The MySQL server works in a client and server system. This system includes a multiple-threaded SQL server that supports various back ends, different client programs and libraries, administrative tools, and many application programming interfaces (API)s.

The directories in the Database are as follows:

Figure 16: Directories in Database

Some working directories are shown as follows. The employee table stores all relevant data fields necessary for the Employee Class in the program.

username	name	password	gender	age	job	email	phone_number	profile_pic
m_ashali	Muhammad Ashhub Ali	SEECS@123	Male	18	Librarian	aam@gmail.com	+92 312 9394 050	BLOB

Figure 17: Employee Table

Similarly, the Member Data stores the Member credentials along with the books issued, and the genres of each book.

	username	password	name	age	member_profile_pic
▶	a_arhamx	SEECS@123	Abdul Arham	18	BLOB

book_issued_1	book_issued_1_genre
The Architect's Apprentice	Novel, Histor, Fiction

gender	email_address	phone_number
Male	a.arham@gmail.com	a.arham@gmail.com

Figure 18: Member Table

The data is stored for the books as follows:

isbn	name	cover	author	publisher	genre	damage_level	issued_by
97802410	The Architect's Apprentice	BLOB	Elif Shafak	Penguin Books	Novel, Histor, Fiction	5	a_arhamx
44631078	To Kill A Mockingbird	BLOB	Harper Lee	J. B. Lippincott & Co.	Fiction, Coming-of-age, South Gothic	5	No One
74327356	The Great Gatsby	BLOB	F. Scott Fitzgerald	Charles Scribner's Sons	Tragedy, Novel, Fiction	5	No One
61122416	The Alchemist	BLOB	Paulo Coelho	HarperTorch	Quest, Adventure, Fantasy	5	No One
1594480001	The Kite Runner	BLOB	Khaled Hosseini	Riverhead Books	Historical fiction, Drama, Classic	5	No One
375842209	The Book Thief	BLOB	Markus Zusak	Alfred A. Knopf	Novel, Young Adult, Fiction	5	No One
812911612	Animal Farm	BLOB	George Orwell	Secker and Warburg	Allegory, Satire, Fable	5	No One

Figure 19: Book Table

The attendance sheet for the Employees marks their attendance corresponding to the dates.

	dates	attendances
▶	01/07/2022	,shahbaz@employee.library: P
	02/07/2022	,shahbaz@employee.library: P

Figure 20: Attendance Table

One example of the many lines of code used to create these tables is:

```
CREATE TABLE books (
    isbn int,
    name varchar(255),
    cover BLOB,
    author varchar(255),
    publisher varchar(255),
    genre varchar(255),
    damage_level int,
    issued_by varchar(255))
```

Figure 21: Database Code in MySQL

Sample Outputs

The following screenshots demonstrate the output of the program. The login page has a welcome message that moves across the screen.

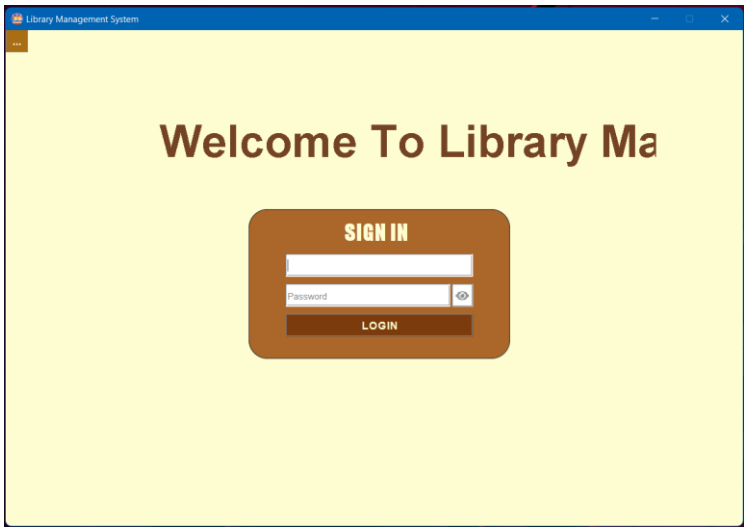

Figure 22: SignIn Page

After the Administrator enters the correct login credentials, they are prompted to the Home Page that displays the Profile Pic, Name, and any messages or requests from the Members. To the left is the options pane that can be accessed by the button at the top left corner.

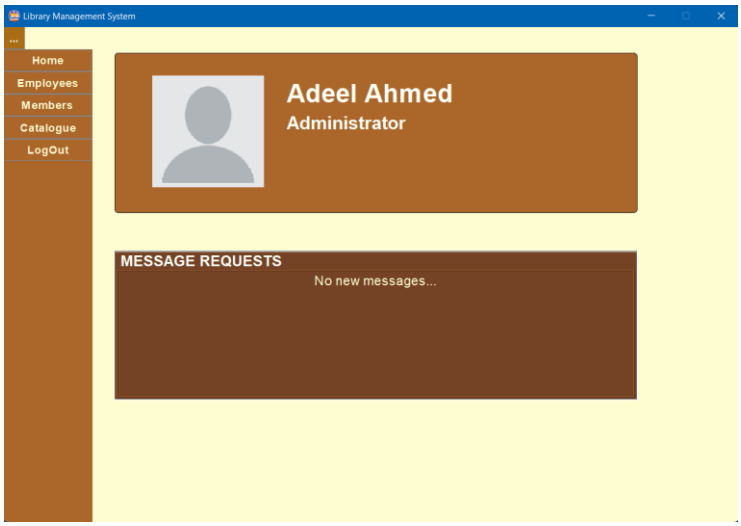

Figure 23: Admin Home Page

Through the options pane, the Administrator or Librarian can access the Employees Panel where they can search for employees as well as remove or ad them.

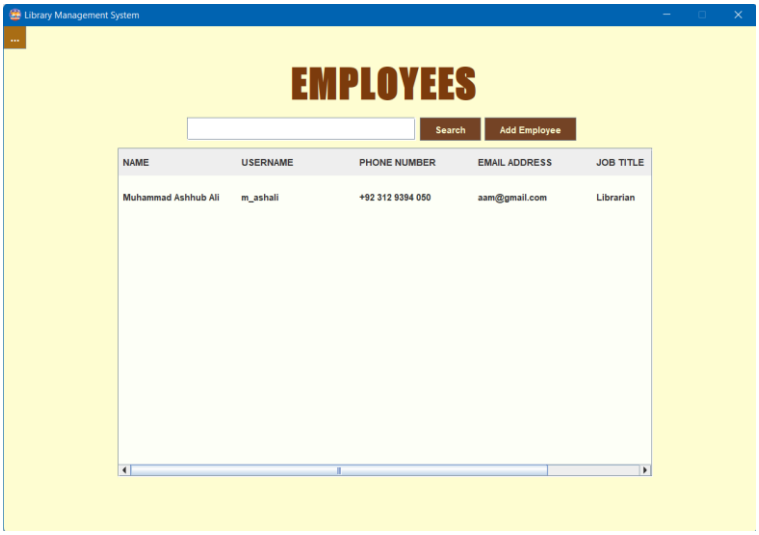

Figure 24: View Employees

If we wish to add a new Employee, we simply fill in the fields, and enter the Add Button.

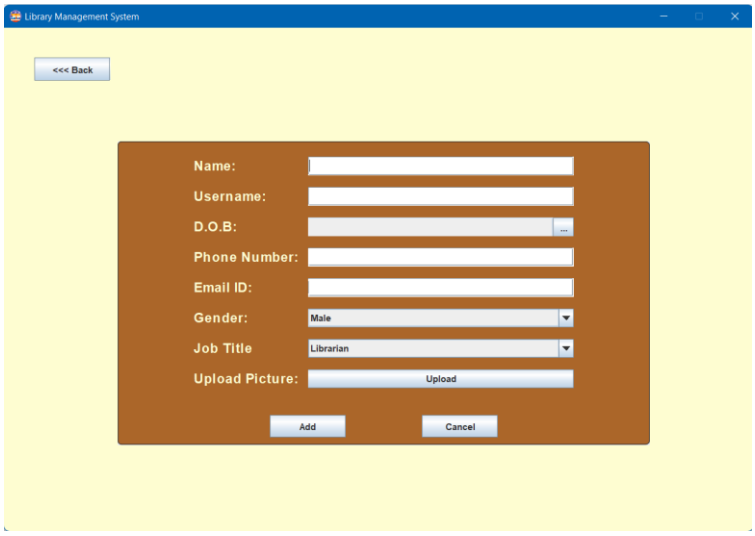

Figure 25: Fields required to Add Employee

Similarly, the Administrator Account can view and add members in the system.

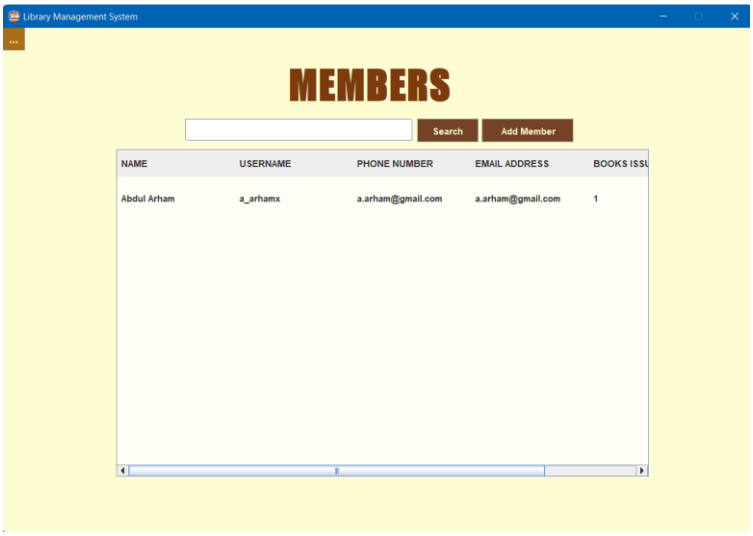

Figure 26: View Members

The DatePicker option displays a Calendar to enter the Date of Birth which is then used to compute the age of the Employee or Member.

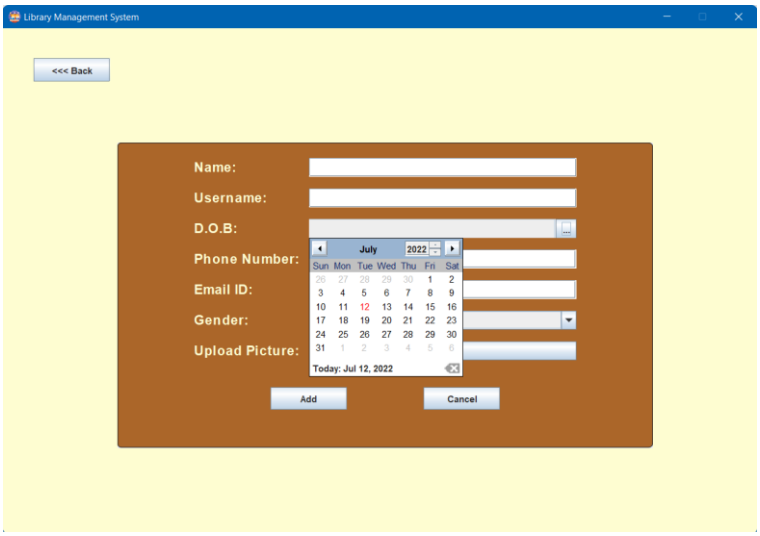

Figure 27: Add Member

The user can view the books and search them by clicking the Catalogue Button in the options pane. Every time the text is changed, the panel refreshes to display the results matching the search.

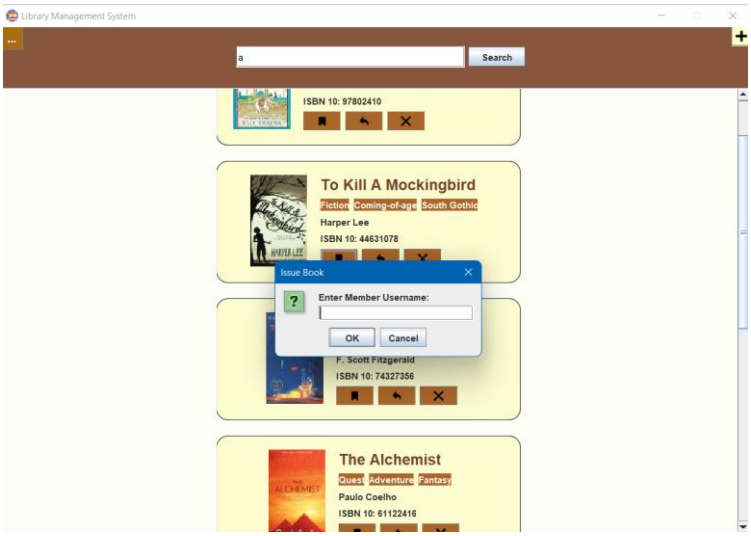

Figure 28: Issuing a Book

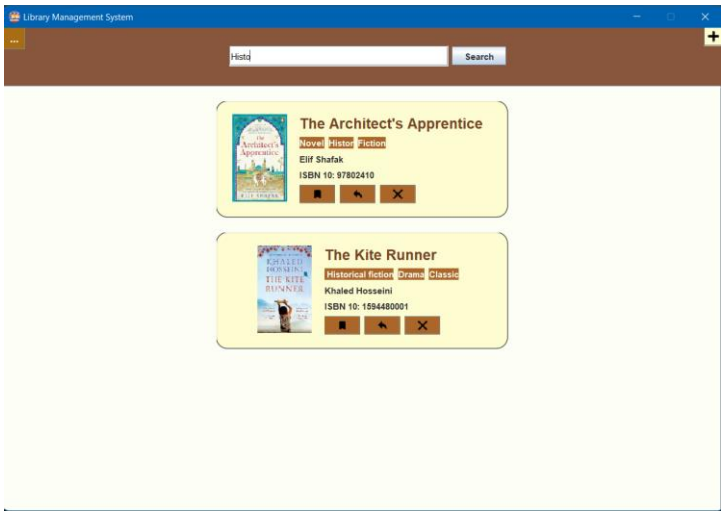

Figure 29: Narrow Down Search