LIBRARY MANAGEMENT SYSTEM

By:

Nada Alshamsi (05) | 2230001749

Khawla Almoalem (04) | 2230003156

Zahara Alsadah (07) | 2230004418

Elaf Alsharqi (03) | 2230003789

Jana Alharbi (07) | 2230004859

Rodynah Alghamdi (01) | 2230000655

GROUP 3

CS 321 - OBJECT ORIENTED PROGRAMMING (2)

COMPUTER SCIENCE

2024-2025

Table of Contents

Introduction:

In an era where information is readily accessible and the demand for efficient resource management is paramount, the need for a robust Library Management System (LMS) has never been more critical.

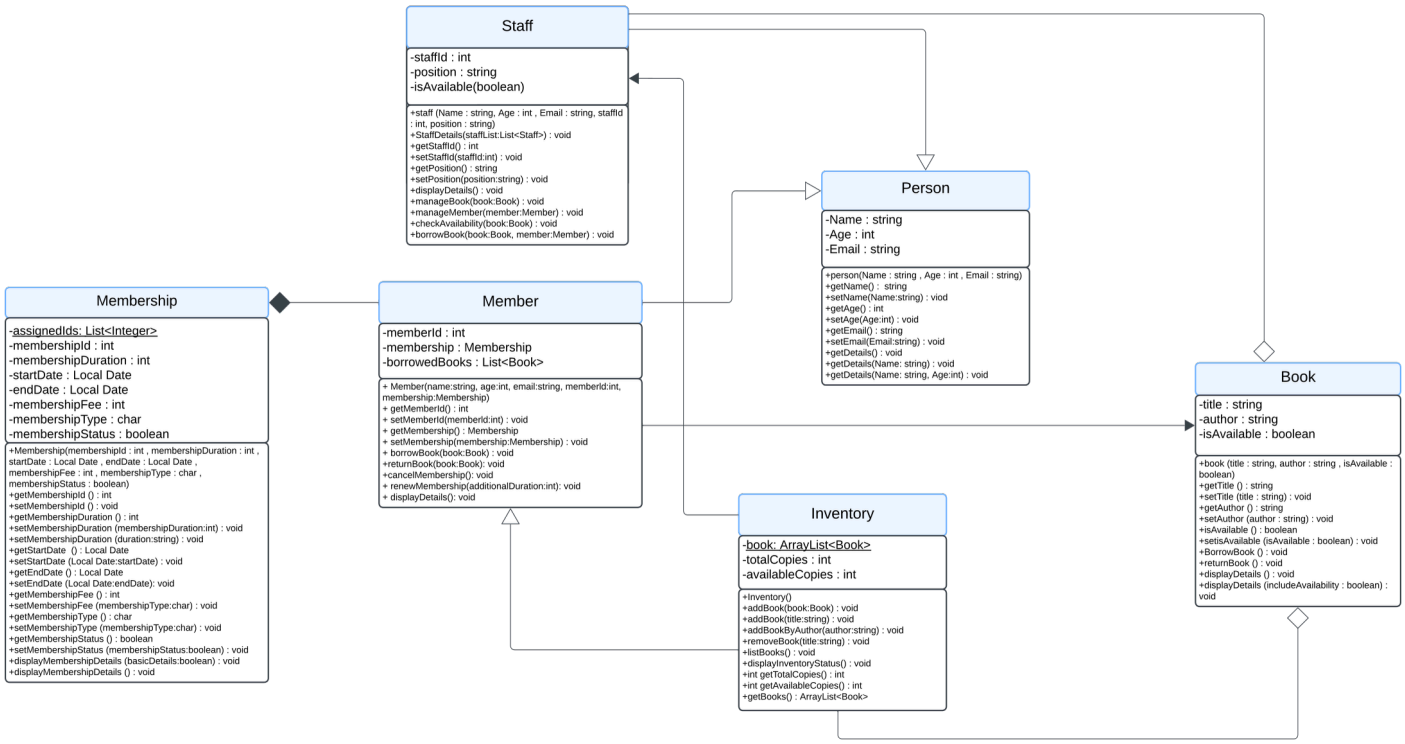
Our project aims to design a Library Management System that is user-friendly and GUI-based to ensure resource management processes become simple for library work, thus interfacing the user and improving operational efficiency.

The main idea of this project is the development of a complete software solution for the entire community of librarians, staff, and patrons. Our LMS would serve to smoothen interaction and keep things running by covering the essential functionalities of book cataloging, member registration, borrowing and returning processes, and inventory management.

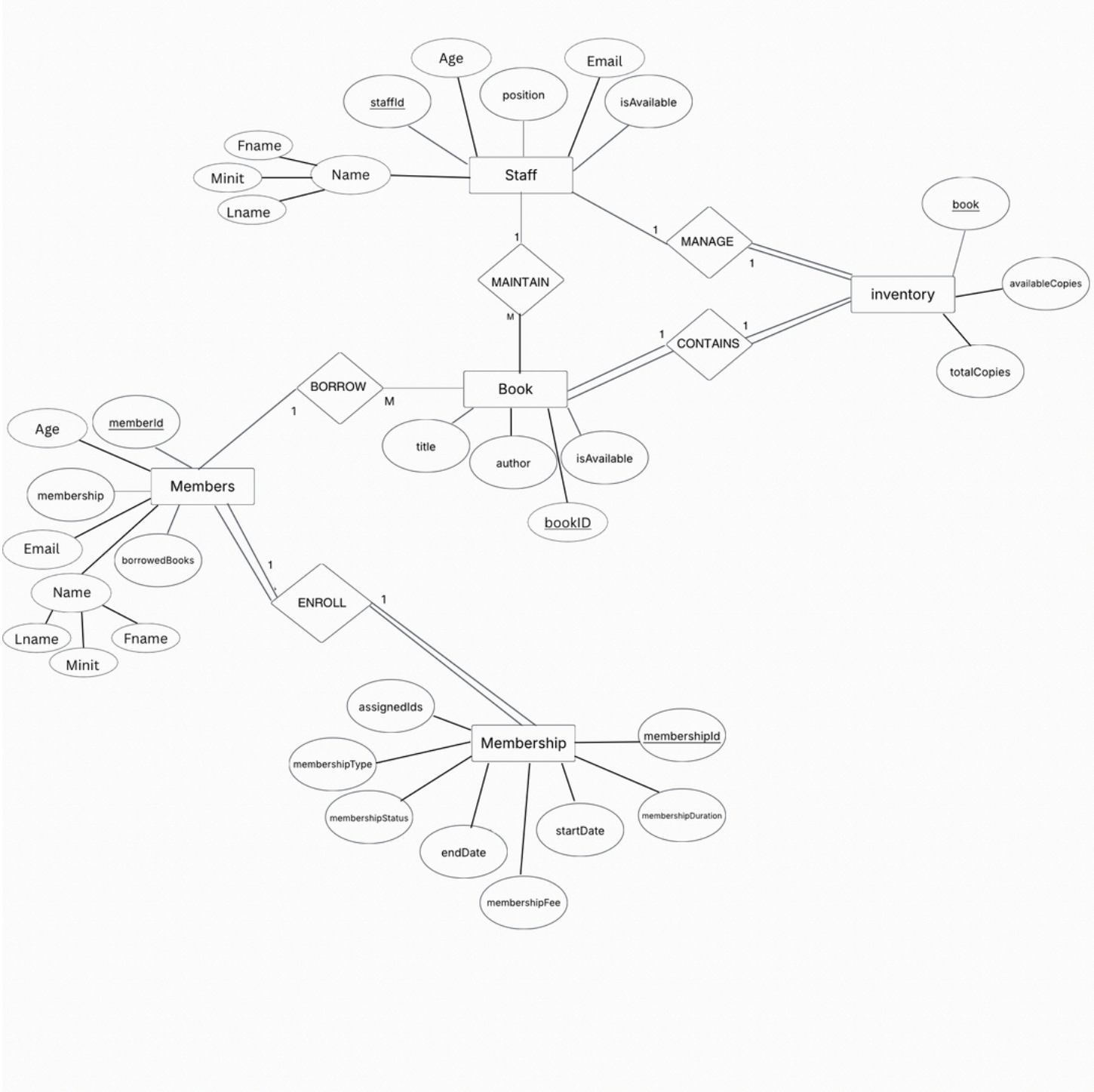
The GUI developed by us takes center stage regarding usability and accessibility, letting a user from every technically aware background navigate through the system easily. Not only does it allow the administration of library resources, but our LMS enhances user experience.

In this project, the library resource management system intends to address some of the challenges faced by libraries. Inventory tracking, user account management, are some of them. The application has been developed aimed at present and the future, using standard programming techniques and design principles.

UML:


Staff
-staffId : int
-position : string
-isAvailable(boolean)
+staff (Name : string, Age : int , Email : string, staffId : int, position : string)
+StaffDetails(staffList:List<Staff>) : void
+getStaffId() : int
+setStaffId(staffId:int) : void
+getPosition() : string
+setPosition(position:string) : void
+displayDetails() : void
+manageBook(book:Book) : void
+manageMember(member:Member) : void
+checkAvailability(book:Book) : void
+borrowBook(book:Book, member:Member) : void
Person
-Name : string
-Age : int
-Email : string
+person(Name : string , Age : int , Email : string)
+getName() : string
+setName(Name:string) : viod
+getAge() : int
+setAge(Age:int) : void
+getEmail() : string
+setEmail(Email:string) : void
+getDetails() : void
+getDetails(Name: string) : void
+getDetails(Name: string, Age:int) : void
Membership
-assignedIds: List<Integer>
-membershipId : int
-membershipDuration : int
-startDate : Local Date
-endDate : Local Date
-membershipFee : int
-membershipType : char
-membershipStatus : boolean
+Membership(membershipId : int , membershipDuration : int , startDate : Local Date , endDate : Local Date , membershipFee : int , membershipType : char , membershipStatus : boolean)
+getMembershipId () : int
+setMembershipId () : void
+getMembershipDuration () : int
+setMembershipDuration (membershipDuration:int) : void
+setMembershipDuration (duration:string) : void
+getStartDate () : Local Date
+setStartDate (Local Date:startDate) : void
+getEndDate () : Local Date
+setEndDate (Local Date:endDate): void
+getMembershipFee () : int
+setMembershipFee (membershipType:char) : void
+getMembershipType () : char
+setMembershipType (membershipType:char) : void
+getMembershipStatus () : boolean
+setMembershipStatus (membershipStatus:boolean) : void
+displayMembershipDetails (basicDetails:boolean) : void
+displayMembershipDetails () : void
Member
-memberId : int
-membership : Membership
-borrowedBooks : List<Book>
+ Member(name:string, age:int, email:string, memberId:int, membership:Membership)
+ getMemberId() : int
+ setMemberId(memberId:int) : void
+ getMembership() : Membership
+ setMembership(membership:Membership) : void
+ borrowBook(book:Book) : void
+returnBook(book:Book): void
+cancelMembership(): void
+ renewMembership(additionalDuration:int): void
+ displayDetails(): void
Book
-title : string
-author : string
-isAvailable : boolean
+book (title : string, author : string , isAvailable : boolean)
+getTitle () : string
+setTitle (title : string) : void
+getAuthor () : string
+setAuthor (author : string) : void
+isAvailable () : boolean
+setisAvailable (isAvailable : boolean) : void
+BorrowBook () : void
+returnBook () : void
+displayDetails () : void
+displayDetails (includeAvailability : boolean) : void
Inventory
-book: ArrayList<Book>
-totalCopies : int
-availableCopies : int
+Inventory()
+addBook(book:Book) : void
+addBook(title:string) : void
+addBookByAuthor(author:string) : void
+removeBook(title:string) : void
+listBooks() : void
+displayInventoryStatus() : void
+int getTotalCopies() : int
+int getAvailableCopies() : int
+getBooks() : ArrayList<Book>

ER Daigram:


Age
Email
staffId
position
isAvailable
Fname
Minit
Name
Lname
Staff
1
1
MANAGE
1
book
MAINTAIN
M
inventory
availableCopies
1
1
CONTAINS
totalCopies
BORROW
M
Book
1
Age
memberId
title
author
isAvailable
membership
Members
bookID
Email
borrowedBooks
1
Name
ENROLL
1
Lname
Fname
Minit
assignedIds
membershipId
Membership
membershipType
membershipStatus
endDate
startDate
membershipDuration
membershipFee

Program implementation and interface design:

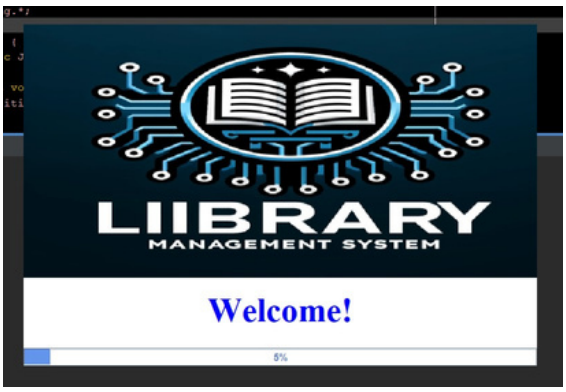


Fig.1
The home page of our GUI project features the Library management system's logo prominently displayed. Creating a welcome interface for users.

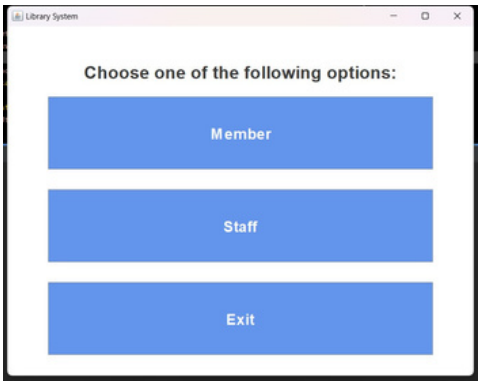


Fig.2
The menu bar include options for users to choose their role, the “Member” button for new and old members of the Library. The “Staff” button is for the admins to monitor and makes changes to the system. And it includes exit option for user convenience.

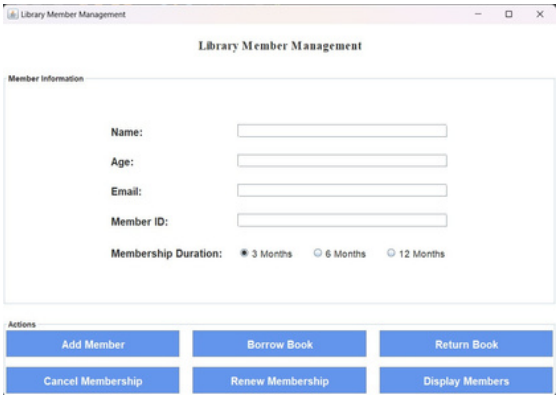


Fig.3
This interface simplifies member management tasks, making it easy for librarians to perform essential operations.

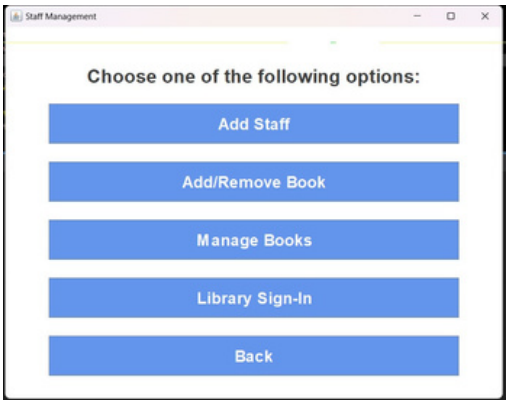


Fig.4
This menu bar provides only staff members with various management options, including adding staff, managing books (add/remove), monitoring library sign-ins, and return back to the main interface.

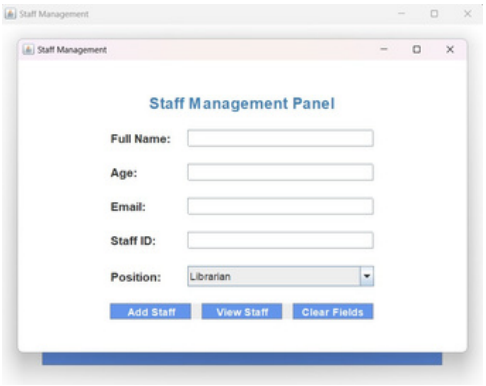


Fig.5
This interface streamlines staff management, allowing administrators to easily add, review, or update employee records.

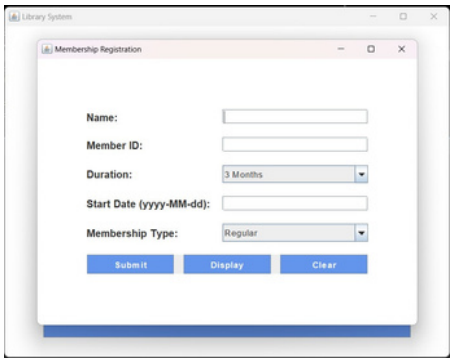


Fig.6
This interface allow user to select the duration and type of their membership. Options in this one also include the start date and membership type. Action buttons allow users to submit, display membership information, or clear all the fields.

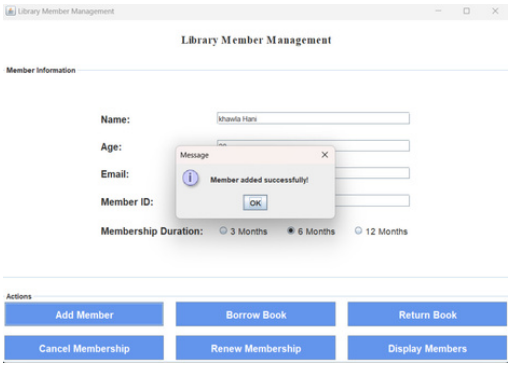


Fig.7

This interface is designed for registering new members. It includes fields for name, age, email, and membership duration and ID for the users to fill. After registering the action buttons allow users to choose either borrowing or returning books, or to make changes to the membership.

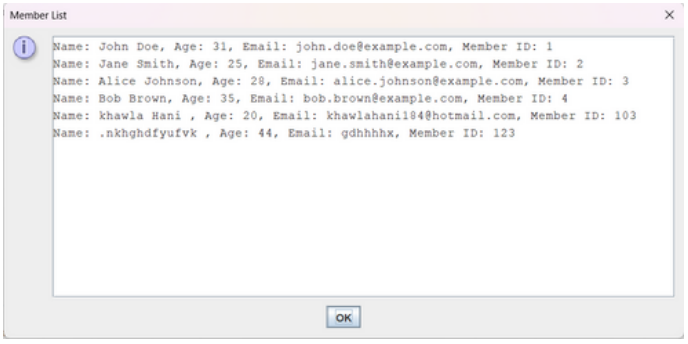


Fig.8

The “Display Members” button presents a list of all registered members, showing the details of the members including name, ID, age, and contact information.

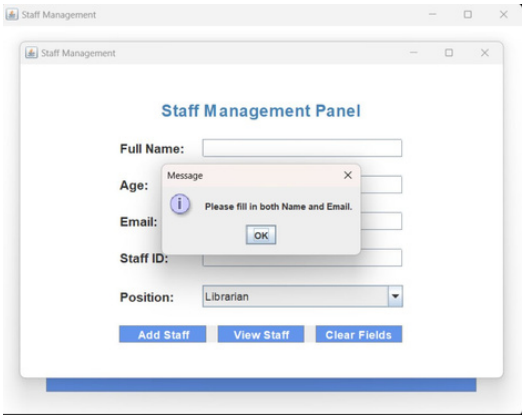


Fig.9

This interface allow the Admin to add new staff member. It includes fields for name, age, email, ID, and position of the staff. Action buttons allow users to add the staff, view staff information, or clear all the fields.

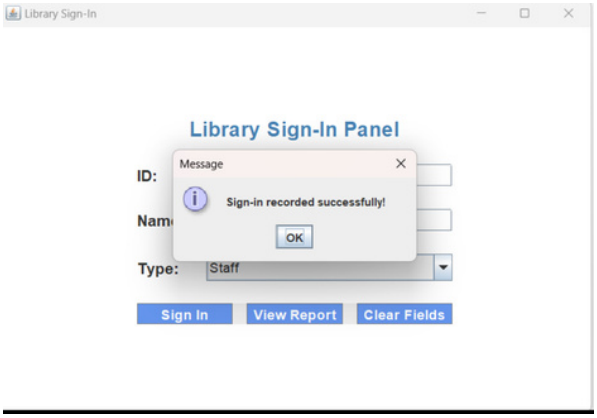


Fig.10

This interface allow the users to sign in if they are already registered. It only ask for user ID, name, and user type (staff/member). And it shows a message of success for the registered users.

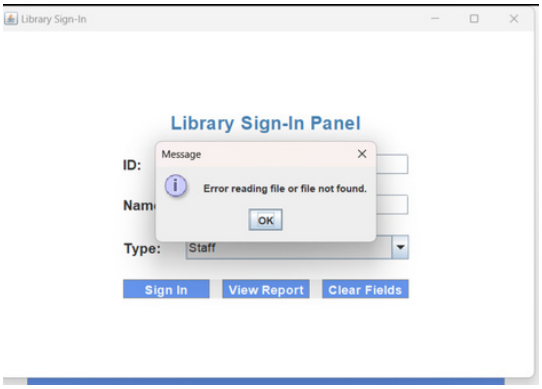


Fig.11

This Sign-In interface also shows another message if the user filled information incorrectly or if the user is not registered in the system.

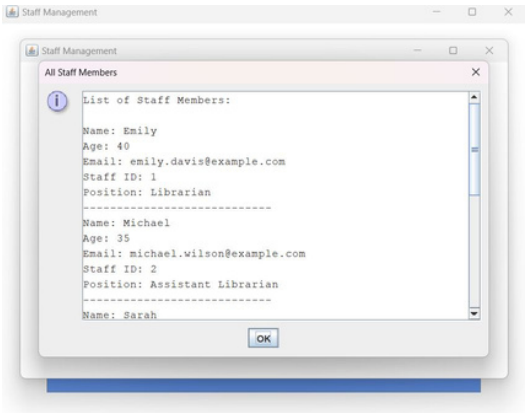


Fig.12

The “View staff” button presents a list of all registered Staff, showing the details of the Staff including name, ID, age, Email, and position.

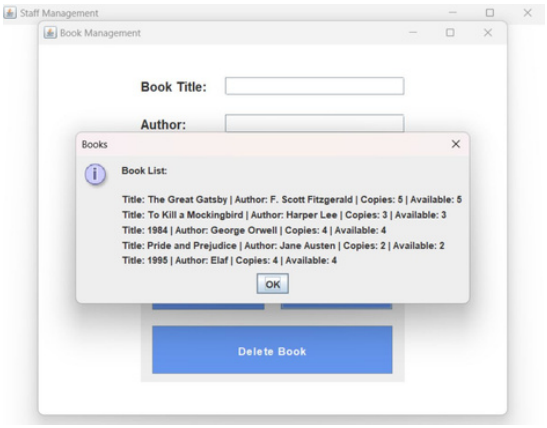


Fig.13

This Interface is for staff to search for books by its title and author. The action buttons allow the staff to add or delete the books. It also present a list of all books in the system showing the details of the books such as, title, author name, how many copies and its availability.

Challenges Faced During the Development of the Library Management System:

One of the main challenges we faced during the implementation of the Library Management System project was in several areas. These included the design of the database, where we had difficulty determining the appropriate tables and the relationships between them, especially when dealing with the interconnection of data between books, members, and borrowing operations. We also faced technical challenges in integrating Java with MySQL, particularly in the early stages when data was not being displayed or saved correctly. In addition, we encountered difficulties in designing the graphical user interfaces, especially in organizing components within the windows in a way that makes the system user-friendly and efficient. On the team level, we experienced challenges in coordinating and distributing tasks among team members, especially when merging different parts of the code and avoiding duplication or conflicting changes.

Challenges Faced During the Development of the Library Management System:

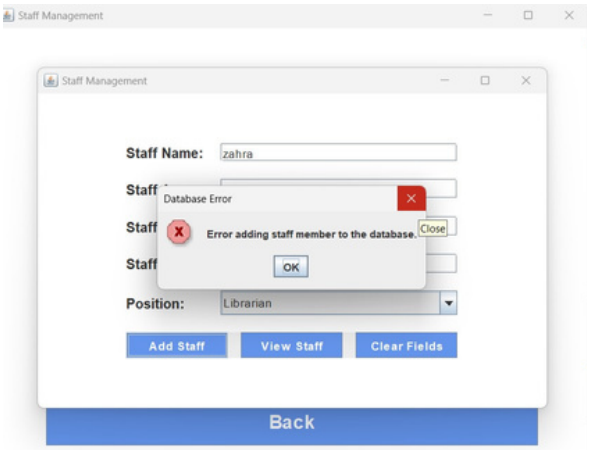


Fig.14: Staff Database Error

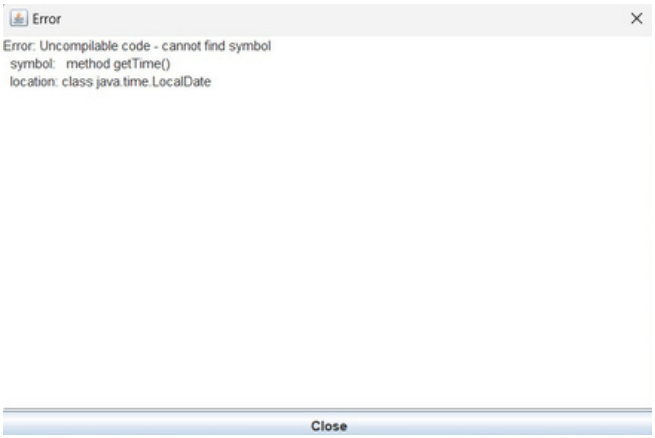


Fig.15: Compilation Error – Method Not Found

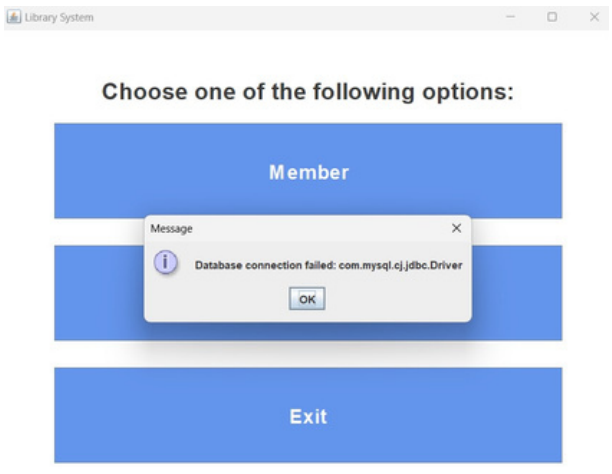


Fig.16: DBC Driver Not Found

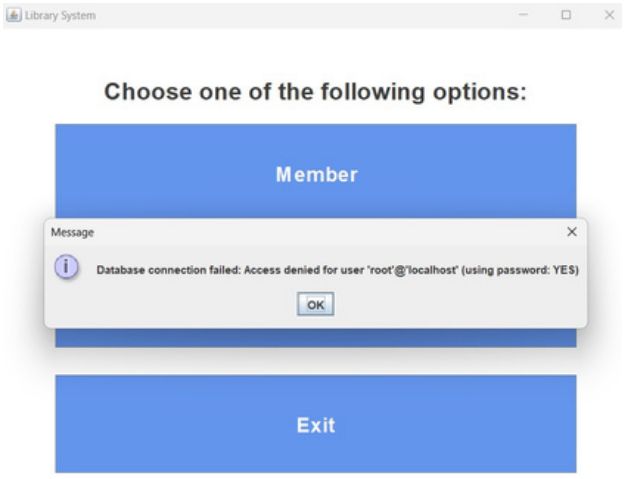


Fig.17: Database Access Denied (Wrong Credentials)

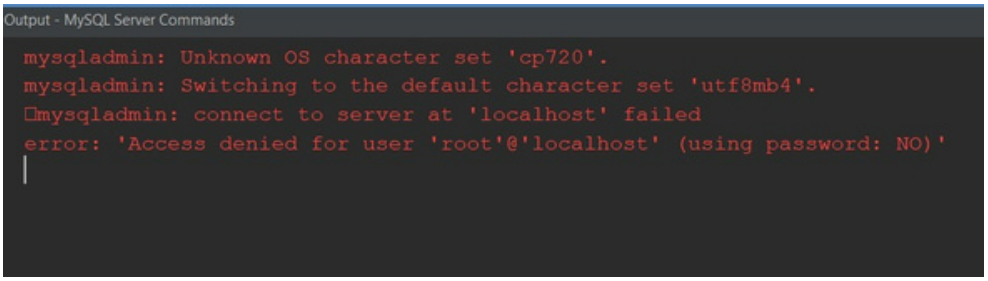


Fig.18: MySQL Login Error – Access Denied

Database connection:

This code is responsible for establishing a connection to a MySQL database called "Library_System3" using JDBC. It defines the URL of the database, the username, and the password needed to access it.

```java
package oop2;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {


    private static final String URL = "jdbc:mysql://localhost:3306/Library_System3?zeroDateTimeBehavior=CONVERT_TO_NULL";
    private static final String USER = "root";
    private static final String PASSWORD = "Khawla_Hani313";

    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            System.err.println("Connection failed: " + e.getMessage());
            return null;
        }
    }
}
```

Fig.19
code for connection to a MySQL database.

Some MySQL database :

```sql
CREATE TABLE staff (
    StaffID INT PRIMARY KEY,
    Fname VARCHAR(20) NOT NULL,
    minit VARCHAR(20),
    Lname VARCHAR(20) NOT NULL,
    age TINYINT,
    email VARCHAR(100),
    position VARCHAR(50),
    is_available BOOLEAN DEFAULT TRUE
);
```

Fig.20: Staff table

```sql
CREATE TABLE book (
    bookID INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    is_available BOOLEAN DEFAULT TRUE,
    StaffID INT,
    FOREIGN KEY (StaffID) REFERENCES staff(StaffID)
);
```

Fig.21: Book table

```sql
CREATE TABLE inventory (
    bookID INT PRIMARY KEY,
    total_copies INT NOT NULL,
    available_copies INT NOT NULL,
    FOREIGN KEY (bookID) REFERENCES book(bookID)
);
```

Fig.22: Inventory table

```sql
CREATE TABLE members (
    MemberID INT PRIMARY KEY,
    Fname VARCHAR(20) NOT NULL,
    minit VARCHAR(20),
    Lname VARCHAR(20) NOT NULL,
    age TINYINT NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);
```

Fig.23: Members table

```sql
CREATE TABLE Membership (
    MembershipID INT AUTO_INCREMENT PRIMARY KEY,
    MemberID INT NOT NULL,
    DurationMonths INT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    MembershipType ENUM('Regular', 'VIP', 'Premium') NOT NULL DEFAULT 'Regular',
    MembershipFee INT NOT NULL,
    MembershipStatus ENUM('Active', 'Cancelled') NOT NULL DEFAULT 'Active',
    FOREIGN KEY (MemberID) REFERENCES members(MemberID) ON DELETE CASCADE
);
```

Fig.24: Membership table

```sql
CREATE TABLE borrow_transactions (
    TransactionID INT AUTO_INCREMENT PRIMARY KEY,
    MemberID INT NOT NULL,
    BookID INT NOT NULL,
    BorrowDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (MemberID) REFERENCES members(MemberID) ON DELETE CASCADE,
    FOREIGN KEY (BookID) REFERENCES book(bookID) ON DELETE CASCADE
);
```

Fig.25: Borrow transactions

```sql
INSERT INTO book (bookID, title, author, is_available) VALUES
(1, 'The Great Gatsby', 'F. Scott Fitzgerald', TRUE),
(2, 'To Kill a Mockingbird', 'Harper Lee', TRUE),
(3, '1984', 'George Orwell', TRUE),
(4, 'Pride and Prejudice', 'Jane Austen', TRUE);
```

Fig.26: Insert statements for book table

```sql
INSERT INTO inventory (bookID, total_copies, available_copies) VALUES
(1, 5, 5),
(2, 3, 3),
(3, 4, 4),
(4, 2, 2);
```

Fig.27: Insert statements for inventory table

```sql
INSERT INTO members (MemberID, Fname, minit, Lname, age, email) VALUES
(1, 'John', 'A', 'Doe', 30, 'john.doe@example.com'),
(2, 'Jane', 'B', 'Smith', 25, 'jane.smith@example.com'),
(3, 'Alice', 'C', 'Johnson', 28, 'alice.johnson@example.com'),
(4, 'Bob', 'D', 'Brown', 35, 'bob.brown@example.com');
```

Fig.28: Insert statements for members table

```sql
INSERT INTO Membership (MemberID, DurationMonths, StartDate, EndDate, MembershipType, Members
(1, 12, '2023-01-01', '2024-01-01', 'Regular', 50, 'Active'),
(2, 6, '2023-06-01', '2023-12-01', 'VIP', 100, 'Active'),
(3, 12, '2023-03-01', '2024-03-01', 'Premium', 150, 'Active'),
(4, 3, '2023-09-01', '2023-12-01', 'Regular', 30, 'Cancelled');
```

Fig.29: Insert statements for membership table

```sql
INSERT INTO staff (StaffID, Fname, minit, Lname, age, email, position, is_available) VALUES
(1, 'Emily', 'E', 'Davis', 40, 'emily.davis@example.com', 'Librarian', TRUE),
(2, 'Michael', 'F', 'Wilson', 35, 'michael.wilson@example.com', 'Assistant Librarian', TRUE),
(3, 'Sarah', 'G', 'Taylor', 29, 'sarah.taylor@example.com', 'Library Manager', TRUE),
(4, 'David', 'H', 'Anderson', 45, 'david.anderson@example.com', 'IT Support', TRUE);
```

Fig.30: Insert statements for staff table

```sql
INSERT INTO borrow_transactions (MemberID, BookID, BorrowDate, ReturnDate) VALUES
(1, 1, '2023-10-01', NULL),
(2, 2, '2023-10-02', NULL),
(3, 3, '2023-10-03', NULL),
(4, 4, '2023-10-04', NULL);
```

Fig.31: Insert statements for borrow transactionstable

```sql
SELECT title FROM book WHERE is_available = TRUE
UNION
SELECT title FROM book WHERE author = 'Jane Austen';

SELECT DISTINCT MembershipType FROM Membership;

SELECT COUNT(*) AS TotalMembers FROM members;

SELECT * FROM members WHERE MemberID IN (1, 2, 3);

SELECT * FROM borrow_transactions WHERE BorrowDate BETWEEN '2023-10-01' AND '2023-10-04';

SELECT * FROM members WHERE email IS NULL;

SELECT * FROM book WHERE NOT is_available;

SELECT * FROM members WHERE email LIKE '%@example.com';

SELECT * FROM members WHERE MemberID IN (SELECT MemberID FROM Membership WHERE MembershipStatus = 'Active');

SELECT * FROM book WHERE bookID > ALL (SELECT bookID FROM book WHERE is_available = FALSE);

SELECT MembershipType, COUNT(*) AS Count FROM Membership GROUP BY MembershipType;

SELECT * FROM members ORDER BY age DESC;
```

Fig.32: SELECT query from some table

Conclusion:

The development of the Library Management System (LMS) presented in this project successfully achieved the objectives of creating a user-centered, GUI-based application for efficient library resource management. Utilizing Java and adhering to object-oriented programming principles, the system effectively facilitates the core operations of book cataloging, member registration, borrowing and returning processes, and inventory management.

The designed interface ensures seamless navigation and interaction, accommodating users with varying levels of technical expertise.

Future improvment:

- Adding security features such as password protection and role-based access control.
- Enhancing search functionality with advanced filtering and sorting options.
- Incorporating real-time notifications for overdue books and membership expirations.
- Improving the UI/UX by introducing more interactive and responsive design elements.

In conclusion, the project establishes a robust framework for a Library Management System capable of adapting to future technological advancements and the evolving needs of library users.