ASSIGNMENT 1

LIBRARY MANAGEMENT SYSTEM

NAME: Ushba Fatima

CMS ID: 467212

Class: BESE 14-B

INTRODUCTION

1. <u>**Project Overview:**</u>

   The Library Management System is a Java-based application developed to streamline library operations and enhance the management of student and book records. This system provides a graphical user interface (GUI) for librarians to efficiently manage library resources. It facilitates tasks such as adding students, adding books, searching books, displaying books, issuing books, and returning books. The application is integrated with a MySQL database to store and retrieve data, ensuring data consistency and reliability.

2. <u>**Purpose and Objectives:**</u>

   The Library Management System is designed to modernize and optimize the operations within a library environment. Its primary purpose is to simplify the management of student and book records while providing librarians with a user-friendly interface. By automating various tasks, this application aims to enhance the efficiency and effectiveness of library operations, ultimately improving the overall library experience for both librarians and patrons.

   a. **Automation:** Automate tasks related to adding, searching, issuing, and returning books, reducing manual effort and time consumption.

   b. **Streamlined Operations**: Provide a streamlined process for managing student and book records, ensuring data accuracy and consistency.

   c. **User-Friendly Interface**: Offer a user-friendly interface for librarians to interact with the system easily and efficiently.

SYSTEM DESIGN

1. Classes:

The following major classes are utilized in the Library Management System:

a. Book:

Represents a book entity with attributes such as book ID, title, author, genre, availability status, issue date, due date, and user ID. The private attributes are accessed through getters/setters.

Attributes:

- **bookID:** Identifier for the book.
- **title:** Title of the book.
- **author:** Author of the book.
- **genre:** Genre of the book.
- **available:** Availability status of the book.
- **issueDate:** Date when the book was issued.
- **dueDate:** Due date for returning the book.
- **userID:** ID of the user who borrowed the book.

```
                 Book

- bookID: String
- title: String
- author: String
- genre: String
- available: String
- issueDate: String
- dueDate: String
- userID: String


+ getBookID(): String
+ getBookTitle(): String
+ getAuthor(): String
+ getGenre(): String
+ getIssueDate(): String
+ getDueDate(): String
+ getUserID(): String
+ getAvailability(): String
+ setAvailability(available: String): void
+ setBookID(bookID: String): void
+ setBookTitle(title: String): void
+ setAuthor(author: String): void
+ setGenre(genre: String): void
+ setUserID(userID: String): void
+ setDueDate(dueDate: String): void
+ setIssueDate(issueDate: String):
void
```

b. User:

Represents a user entity, typically a student, with attributes including user ID, name, contact number, and course name. The private attributes are accessed through getters/setters.

Attributes:

- **userID:** Unique identifier for the user.
- **name:** Name of the user.
- **contactNo:** Contact number of the user.
- **courseName:** Course in which the user is enrolled.

```
                        User
  ─────────────────────────────────────────
  - userID: String
  - name: String
  - contactNo: String
  - courseName: String
  ─────────────────────────────────────────
  + getUserID(): String
  + getUserName(): String
  + getContactNo(): String
  + getCourseName(): String
  + setUserID(userID: String): void
  + setUserName(name: String): void
  + setContactNo(contactNo: String):
  void
  + setCourseName(courseName:
  String): void
```
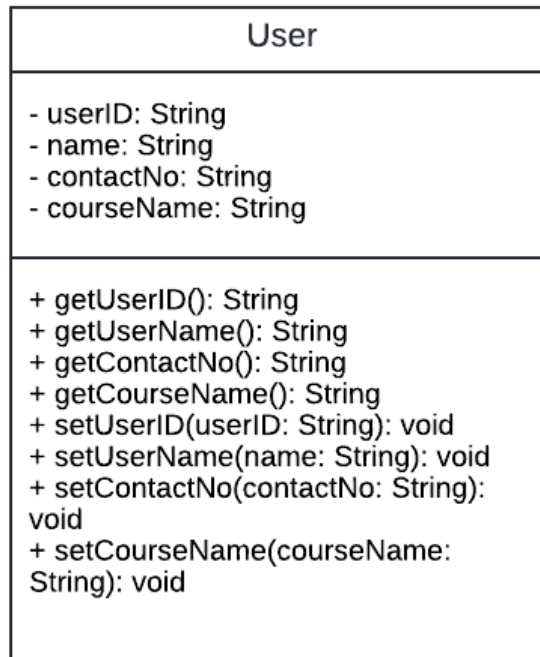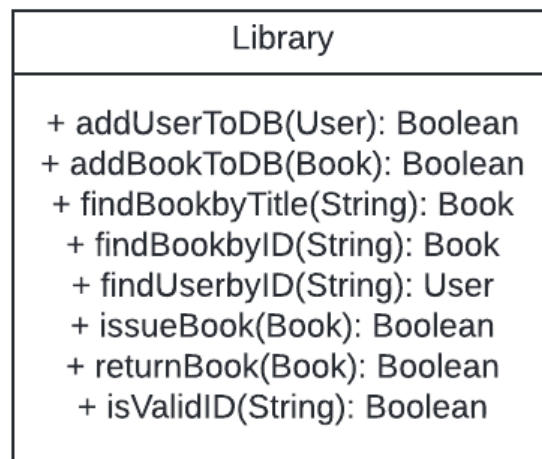
c. **Library:**

Serves as the core class responsible for implementing various functionalities of the library management system.

Contains methods for adding users and books to the database, finding books and users by their attributes, issuing and returning books, and validating IDs.

Methods:

- **addUserToDB(User user):** Adds a new user to the database. Returns a boolean indicating success status of adding a user.
- **addBookToDB(Book book):** Adds a new book to the database. Returns a boolean indicating success status of adding a book.
- **findBookbyTitle(String title):** Finds a book in the database by its title and returns it.
- **findBookbyID(String ID):** Finds a book in the database by its ID and returns it.
- **findUserbyID(String ID):** Finds a user in the database by their ID and returns it.
- **issueBook(Book book):** Issues a book to a user. Returns Boolean indicating the success status of issuing a book.
- **returnBook(Book book):** Returns a book to the library that was issued earlier. Returns Boolean indicating the success status of returning a book.
- **isValidID(String ID):** Validates the format of an ID. Returns a Boolean to validate the ID

Library
+ addUserToDB(User): Boolean + addBookToDB(Book): Boolean + findBookbyTitle(String): Book + findBookbyID(String): Book + findUserbyID(String): User + issueBook(Book): Boolean + returnBook(Book): Boolean + isValidID(String): Boolean

2. Database:

MySQL serves as the backend database management system for the library management application. The database is named "library" and consists of two tables: "student" and "book".

a. Student Table:

Manages student data.

Columns:

- **studentName**: Name of the student (varchar(30)).
- **studentID**: Unique identifier for the student (varchar(20), Primary Key).
- **contact**: Contact number of the student (varchar(20)).
- **course**: Course enrolled by the student (varchar(10)).

b. Book Table:

Handles book-related information.

Columns:

- **bookTitle**: Title of the book (varchar(50)).
- **bookID**: Unique identifier for the book (varchar(20), Primary Key).
- **author**: Author of the book (varchar(50)).
- **genre**: Genre of the book (varchar(30)).
- **availability**: Availability status of the book (varchar(20)).
- **issueDate**: Date when the book was issued (varchar(60)).
- **dueDate**: Due date for returning the book (varchar(60)).
- **studentID**: ID of the student who borrowed the book (varchar(20)).

c. Database Connectivity:

JDBC (Java Database Connectivity) is utilized to establish a connection to the MySQL database. The Connect class contains the logic for establishing this connection. This configuration ensures seamless interaction between the application and the MySQL database for effective data storage and retrieval. The following command is ru for establishing a connection.

```
Connectioncon=DriverManager.getConnection("jdbc:mysql://localhost:3306/library", "root", "MySQL@2773778");
```

IMPLEMENTATION

1. Software Requirements:

a. **Java Development Kit (JDK)**

b. **Eclipse IDE with WindowBuilder plugin**

Windows Builder 1.13.0: [Eclipse downloads - Select a mirror | The Eclipse Foundation](#)

c. **MySQL database**

[MySQL :: MySQL Community Downloads](#) (My SQL Installer for Windows)

d. **MySQL Connector/J 8.3.0**

[MySQL :: Download Connector/J](#) (Platform Independent)

e. **rs2xml.jar file for populating JTables**

[rs2xml.jar - Google Drive](#)

2. Key Features:

a. **User Authentication:** Admins can log in using predefined credentials to access the application.

b. **User Management:** Add new users to the database, including their name, ID, contact number, and course.

c. **Book Management:** Add new books to the database, including details like title, ID, author, genre, and availability.

d. **Search Functionality:** Search for books by title.

e. **Display Books:** View a list of all books stored in the database.

f. **Book Issuing:** Issue books to users by updating their due dates, issue dates, and availability status.

g. **Book Returning:** Return books by updating their availability status and removing user details.

h. **GUI Interface:** Utilize a user-friendly graphical interface created using the WindowBuilder plugin for Eclipse, featuring drag-and-drop tools and action buttons for seamless interaction.

i. **Login Page Integration:** Begin the program execution with a login page to authenticate users and grant access to the entire application.

GRAPHICAL USER INTERFACE

The graphical user interface (GUI) of this application is developed using the Windows Builder tool in Eclipse, which offers a convenient drag-and-drop interface for designing user interfaces. The GUI consists of various windows, each serving a specific purpose in the library management system.

The program starts with the loginPage, the entry point to the system. Users log in here to access the library management features. From the homePage, users can navigate to different windows and perform tasks. The loginPage controls access to the entire program, ensuring security.

 Here's an overview of the GUI components and their functionality:

3. Login Page:

This window prompts the user to enter their credentials to log in as an administrator. It typically includes fields for entering the username and password, along with a "Login" button to authenticate.
Admin credentials:
a. Username: Admin1 Password: library123
b. Username: Admin2 Password: library456
c. Username: Admin3 Password: library789

Screenshot:
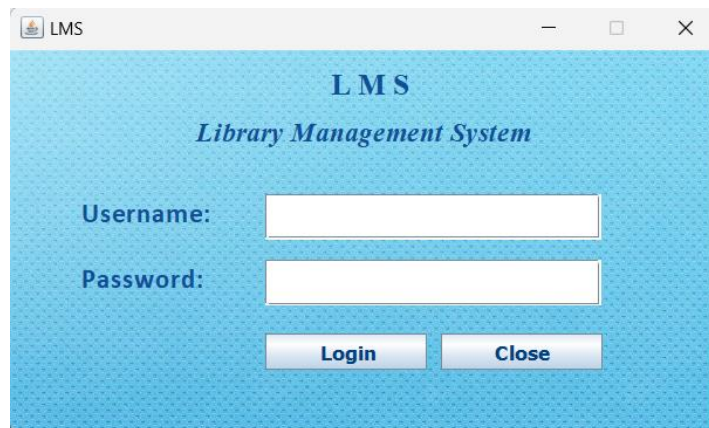
4. Home Page:

Upon successful login, the user is directed to the homePage, which serves as the main interface for accessing different functionalities of the library management system.
It contains buttons representing various actions or tasks that the user can perform, such as adding a new user, adding a new book, searching for books, displaying books, issuing books, and returning books. Each button or tab is associated with a specific window. The logout button leads back to the login window

5. Add User Window:

This window allows the librarian to add a new user to the library database. It typically includes fields for entering the user's name, ID, contact number, and course details. A "Add User" button is provided to add the user to the database.

6. **Add Book Window:**

   Similar to the Add User window, this window enables the librarian to add a new book to the library database. It includes fields for entering the book's title, ID, author, genre, and availability status. A "Add Book" button allows the librarian to add the book to the database.

7. **Search Book Window:**

   This window allows the librarian to search for books in the database based on their title. It typically includes a search field where the librarian can enter the title of the book they want to search for. A "Search" button triggers the search operation, displaying the matching results.

8. **Display Books Window:**

   This window displays a list of all the books available in the library database in table format.

9. <u>Issue Book Window:</u>

This window facilitates the process of issuing a book to a library user. It includes fields for entering the book ID and the user ID to whom the book is being issued. A "Issue" button allows the librarian to complete the book issuing process.

10. <u>Return Book Window:</u>

Similar to the Issue Book window, this window is used to process the return of a book by a library user. It includes fields for entering the book ID and the user ID returning the book. A "Return" button triggers the book return process.

Overall, the GUI is designed to provide user-friendly interface so that the librarian can navigate through different operations easily.

TESTING

During the testing phase of the library management system, evaluations are conducted to assess its functionality and performance. Various scenarios are explored to ensure the reliability and effectiveness of the software. These tests cover aspects such as login functionality, addition of users and books, search capabilities, and book issuing and return processes. Additionally, error handling mechanisms are thoroughly tested to ensure proper handling of unexpected situations.

1. <u>Login Page</u>

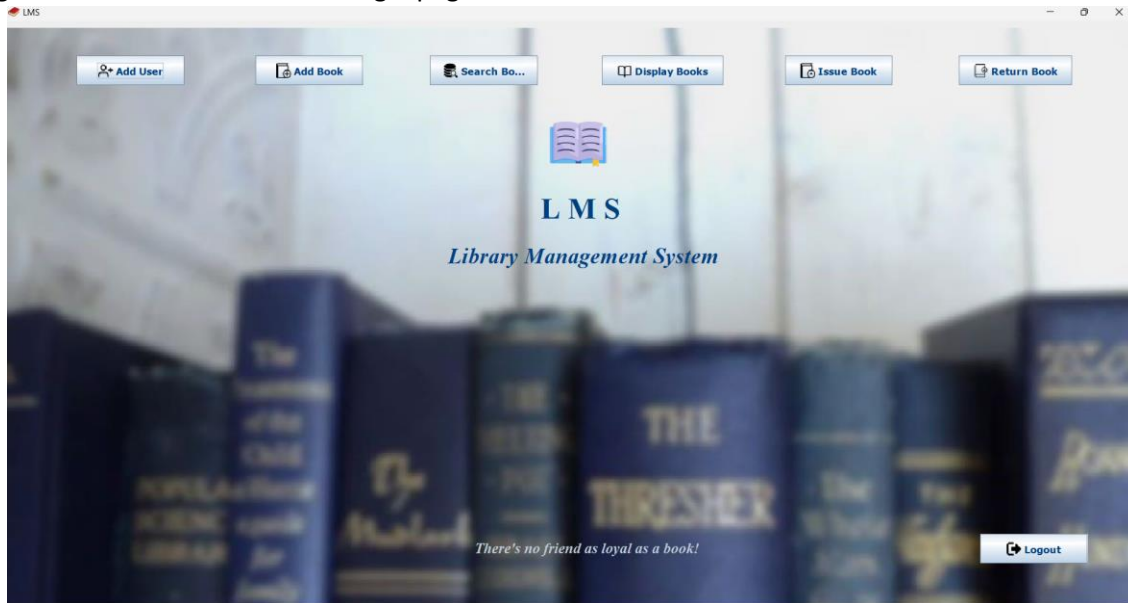a. Entering correct credentials leads to the home page.
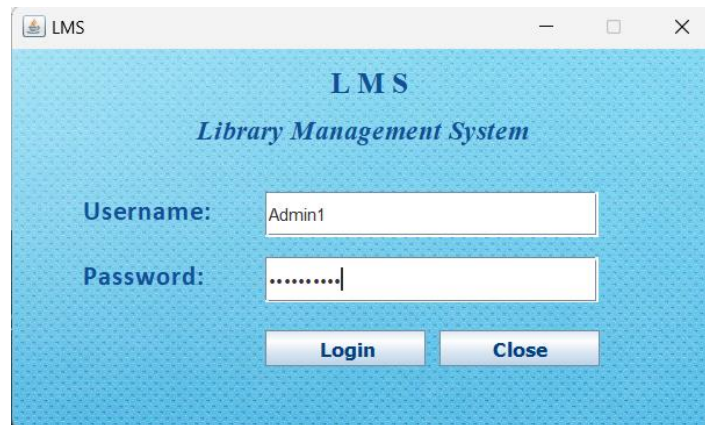
b. Entering wring credentials displays error message.
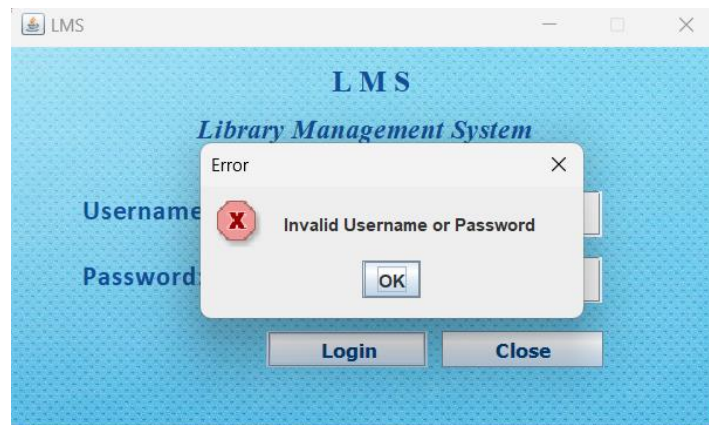
2. Home Page:

It is ensured that all buttons on the homepage lead to the correct windows and functionalities and the logout button leads back to the login page.

3. Add User Window:

a. Verify that users can be added successfully in the database without any errors.
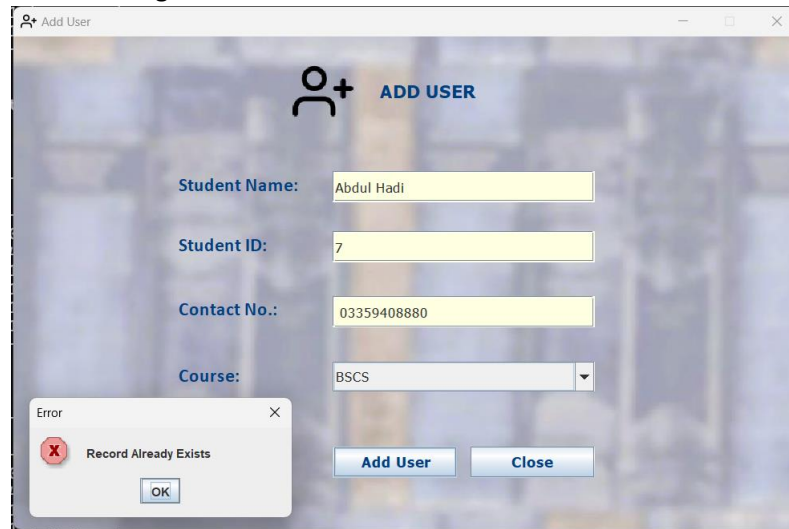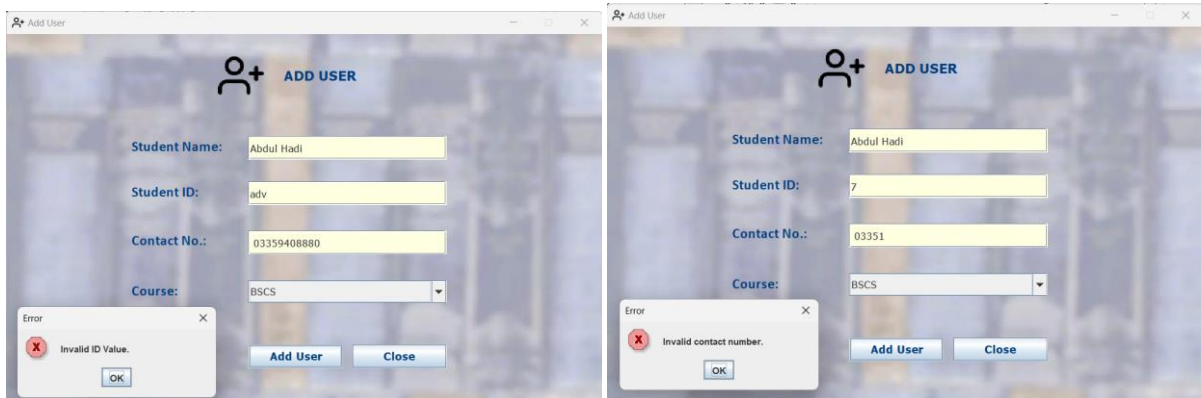
b. Check if the system properly handles cases where a user with the same ID already exists, displaying an appropriate error message.

c. Validate the handling of invalid inputs such as an invalid ID, name, or contact number, ensuring that the system displays relevant error messages.

4. Add Book Window

   a. Successfully adding books to database


```
mysql> select * from book;
+----------------------+--------+------------------+-------------------+--------------+-----------+----------+-----------+
| bookTitle            | bookID | author           | genre             | availability | issueDate | dueDate  | studentID |
+----------------------+--------+------------------+-------------------+--------------+-----------+----------+-----------+
| Java- How to Program | 1      | Paul & Dietel    | Education         | Available    | NULL      | NULL     | NULL      |
| Harry Potter         | 2      | JK Rowling       | Fantasy           | Available    | NULL      | NULL     | NULL      |
| The Hunger Games     | 3      | Suzanne Collins  | Fiction           | Available    | NULL      | NULL     | NULL      |
| War and Peace        | 4      | Leo Tolstoy      | Historical Fiction| Available    | NULL      | NULL     | NULL      |
| Pride and Prejudice  | 5      | Jane Austen      | Romance           | Available    | NULL      | NULL     | NULL      |
+----------------------+--------+------------------+-------------------+--------------+-----------+----------+-----------+
```

   b. Checking if the record already exists based on book ID.

   c. Checking for valid ID value:

5. Searching Book Window:

a. Successful searching of book by title from database on pressing the search button.

b. Error message if book not found in the database.

6. Display Books:

Simply displays the table of books stored in the database.

7. Issue Book:

a. Successful issuing of a book.
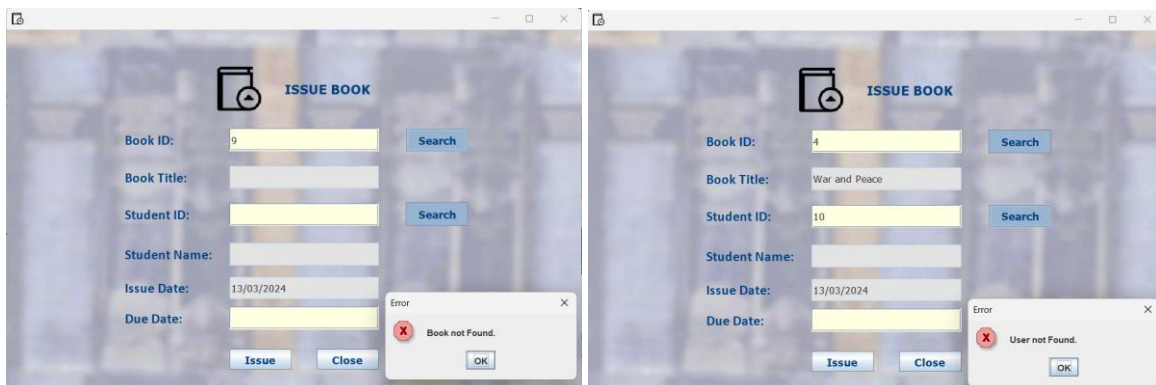

```
mysql> select * from book;
+-----------------------+--------+-----------------+--------------------+---------------+-----------+------------+----------+
| bookTitle             | bookID | author          | genre              | availability  | issueDate | dueDate    | studentID|
+-----------------------+--------+-----------------+--------------------+---------------+-----------+------------+----------+
| Java- How to Program  | 1      | Paul & Dietel   | Education          | Available     | NULL      | NULL       | NULL     |
| Harry Potter          | 2      | JK Rowling      | Fantasy            | Available     | NULL      | NULL       | NULL     |
| The Hunger Games      | 3      | Suzanne Collins | Fiction            | Available     | NULL      | NULL       | NULL     |
| War and Peace         | 4      | Leo Tolstoy     | Historical Fiction | Available     | NULL      | NULL       | NULL     |
| Pride and Prejudice   | 5      | Jane Austen     | Romance            | Not Available | 13/03/2024| 13/05/2024 | 2        |
+-----------------------+--------+-----------------+--------------------+---------------+-----------+------------+----------+
```
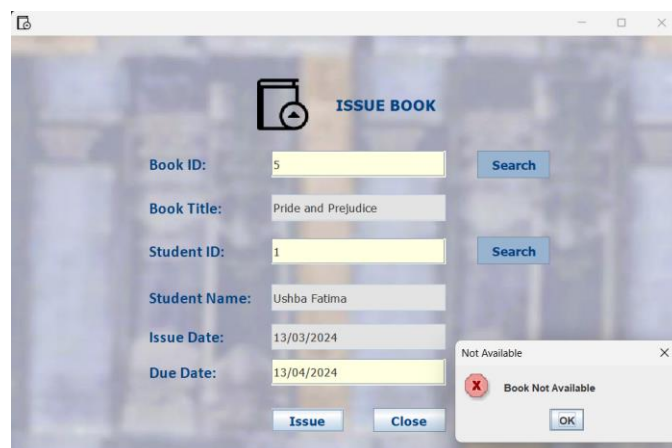
b. Error if book/user data is not found in database

c. Error is the book has already been issued

d. Input Validation:

8. Return Book Window:
a. Successful returning of book


```
mysql> select * from book;
+----------------------+--------+-----------------+--------------------+--------------+-----------+---------+-----------+
| bookTitle            | bookID | author          | genre              | availability | issueDate | dueDate | studentID |
+----------------------+--------+-----------------+--------------------+--------------+-----------+---------+-----------+
| Java- How to Program | 1      | Paul & Dietel   | Education          | Available    | NULL      | NULL    | NULL      |
| Harry Potter         | 2      | JK Rowling      | Fantasy            | Available    | NULL      | NULL    | NULL      |
| The Hunger Games     | 3      | Suzanne Collins | Fiction            | Available    | NULL      | NULL    | NULL      |
| War and Peace        | 4      | Leo Tolstoy     | Historical Fiction | Available    | NULL      | NULL    | NULL      |
| Pride and Prejudice  | 5      | Jane Austen     | Romance            | Available    | NULL      | NULL    | NULL      |
+----------------------+--------+-----------------+--------------------+--------------+-----------+---------+-----------+
```

b. Error message if book is already available.

c. Input Validation:

1. <u>Book Class:</u>

```java
package libraryManagementSystem;

// Class representing a Book object
class Book {
    // Private fields to store book
attributes
    private String bookID;
    private String title;
    private String author;
    private String genre;
    private String available;
    private String issueDate;
    private String dueDate;
    private String userID;

    // Default constructor
    public Book() {
    }

    // Getter methods to retrieve the
values of private fields

    public String getBookID() {
        return bookID;
    }

    public String getBookTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public String getGenre() {
        return genre;
    }

    public String getIssueDate() {
        return issueDate;
    }

    public String getDueDate() {
        return dueDate;
    }

    public String getUserID() {
        return userID;
    }

    public String getAvailability() {
        return available;
    }

    // Setter methods to modify the values
of private fields

    public void setAvailability(String
available) {
        this.available = available;
    }

    public void setBookID(String bookID) {
        this.bookID = bookID;
    }

    public void setBookTitle(String title)
{
        this.title = title;
    }

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

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public void setUserID(String userID) {
        this.userID = userID;
    }

    public void setDueDate(String dueDate)
{
        this.dueDate = dueDate;
    }

    public void setIssueDate(String
issueDate) {
        this.issueDate = issueDate;
    }
}
```

2. User Class

```java
package libraryManagementSystem;

// Class representing a User object
public class User {
    // User Attributes
    private String userID;
    private String name;
    private String contactNo;
    private String courseName;

    // Constructor
    public User() {

    }

    // Getter methods to retrieve the values of private fields

    public String getUserID() {
        return userID;
    }

    public String getUserName() {
        return name;
    }

    public String getContactNo() {
        return contactNo;
    }

    public String getCourseName() {
        return courseName;
    }


    // Setter methods to modify the values of private fields

    public void setUserID(String userID) {
        this.userID = userID;
    }

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

    public void setContactNo(String contactNo) {
        this.contactNo = contactNo;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }

}
```

3. Library Class:

```java
package libraryManagementSystem;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Library {

    static Connection con =
Connect.ConnectToDB();
    static PreparedStatement pst = null;
    static ResultSet rs = null;

    public Library() {

    }

    /******** ADD USER **********/
    public static Boolean
addUserToDB(User user) {
        try {
            // Prepare SQL
statement to insert user details into
the database
            pst =
con.prepareStatement("INSERT INTO
STUDENT
(studentName,studentID,contact,course)
VALUES (?,?,?,?)");

            // Set values for the
parameters in the SQL statement using
user details
            pst.setString(1,
user.getUserName());
            pst.setString(2,
user.getUserID());
            pst.setString(3,
user.getContactNo());
            pst.setString(4,
user.getCourseName());

            // Execute the SQL
statement to insert the user into the
database
            pst.executeUpdate();

            // Return true to
indicate successful addition of user to
the database
            return true;

        } catch (SQLException e1) {
            e1.printStackTrace();

            // Return false to indicate unsuccessful
addition of user to the database
            return false;
        }
    }

    /***************** ADD USER
*****************/

    /***************** ADD BOOK
*****************/
    public static Boolean
addBookToDB(Book book) {
        try {
            // Prepare SQL
statement to insert book details into
the database
            pst =
con.prepareStatement(
                        "INSERT
INTO BOOK
(bookTitle,bookID,author,genre,
availability) VALUES (?,?,?,?,?)");

            // Set values for the
parameters in the SQL statement using
book details
            pst.setString(1,
book.getBookTitle());
            pst.setString(2,
book.getBookID());
            pst.setString(3,
book.getAuthor());
            pst.setString(4,
book.getGenre());
            pst.setString(5,
book.getAvailability());

            // Execute the SQL
statement to insert the book into the
database
            pst.executeUpdate();

            // Return true to
indicate successful addition of book to
the database
            return true;
        } catch (SQLException e1) {
            e1.printStackTrace();
            return false;
        }
    }

    /***************** ADD BOOK
*****************/
```

```java
        /***************** FIND BOOK
******************/
    // Find Book by Title
    public static Book
findBookbyTitle(String title) {
        // Create a new Book object
        Book book = new Book();

        try {
            // Prepare SQL
statement to select book from database
using book title
            pst =
con.prepareStatement("SELECT * FROM BOOK
WHERE bookTitle=?");
            pst.setString(1,
title);
            rs =
pst.executeQuery();

            // If a book with the
specified title exists in the database
            if (rs.next()) {
                // Set the
book details retrieved from the database

    book.setBookTitle(rs.getString("book
Title"));

    book.setBookID(rs.getString("bookID"
));

    book.setAuthor(rs.getString("author"
));

    book.setGenre(rs.getString("genre"))
;

    book.setAvailability(rs.getString("a
vailability"));

    book.setDueDate(rs.getString("dueDat
e"));

    book.setIssueDate(rs.getString("issu
eDate"));
            } else {
                // If no book
with the specified title is found,
return null
                return null;
            }

        } catch (SQLException e1) {
            e1.printStackTrace();
            return null;
        }

        // Return the Book object
with the retrieved details
        return book;
    }

    // Find book by name
    public static Book
findBookbyID(String ID) {
        // Create a new Book object
        Book book = new Book();

        try {
            // Prepare SQL
statement to select book from database
using book ID
            pst =
con.prepareStatement("SELECT * FROM BOOK
WHERE bookID=?");
            pst.setString(1, ID);
            rs =
pst.executeQuery();

            // If a book with the
specified ID exists in the database
            if (rs.next()) {
                // Set the
book details retrieved from the database

    book.setBookTitle(rs.getString("book
Title"));

    book.setBookID(rs.getString("bookID"
));

    book.setAuthor(rs.getString("author"
));

    book.setGenre(rs.getString("genre"))
;

    book.setUserID(rs.getString("student
ID")); // Assuming you have a method to
set the user ID who borrowed

                // the book

    book.setAvailability(rs.getString("a
vailability"));

    book.setDueDate(rs.getString("dueDat
e"));

    book.setIssueDate(rs.getString("issu
eDate"));
```

```
                } else {
                        return null;
                }

        } catch (SQLException e1) {
                e1.printStackTrace();
                return null;
        }

        // Return the Book object
with the retrieved details
        return book;
    }
    /**************** FIND BOOK
******************/

    /*********** FIND USER
**************/
    public static User
findUserbyID(String ID) {
        // Create a new User object
        User user = new User();

        try {
                // Prepare SQL
statement to select user from database
using user ID
                pst =
con.prepareStatement("SELECT * FROM
STUDENT WHERE studentID=?");
                pst.setString(1, ID);
                rs =
pst.executeQuery();

                // If the user with
the specified ID exists in the database
                if (rs.next()) {
                        // Set the
user details retrieved from the database

    user.setUserName(rs.getString("stude
ntName"));

    user.setUserID(rs.getString("student
ID"));

    user.setContactNo(rs.getString("cont
act"));

    user.setCourseName(rs.getString("cou
rse"));
                } else {
                        // If no user
with the specified ID is found, return
null
                        return null;
                }
```

```
        } catch (SQLException e1) {
                e1.printStackTrace();
                return null;
        }

        // Return the User object
with the retrieved details
        return user;
    }
    /*********** FIND USER
**************/

    /**************** ISSUE BOOK
*****************/
    public static Boolean issueBook(Book
book) {
        try {
                // Updates book
details in the database based on the
provided information that
                // has been populated
in the object book
                pst =
con.prepareStatement("UPDATE BOOK SET
ISSUEDATE=?,DUEDATE=?,STUDENTID=? WHERE
BOOKID=?");
                pst.setString(1,
book.getIssueDate());
                pst.setString(2,
book.getDueDate());
                pst.setString(3,
book.getUserID());
                pst.setString(4,
book.getBookID());

                int rowsAffected =
pst.executeUpdate(); // Use
executeUpdate() for UPDATE queries
                // If update was
successful
                if (rowsAffected > 0)
{

    book.setAvailability("Not
Available");

                        // Update the
availability of the book
                        pst =
con.prepareStatement("UPDATE BOOK SET
AVAILABILITY=? WHERE BOOKID=?");

    pst.setString(1, "Not Available");

    pst.setString(2, book.getBookID());
```

```java
        pst.executeUpdate(); // Execute the
update query for availability

                        // Update the
book object's availability

        book.setAvailability("Not
Available");

                        return true;
// Indicate that the book was
successfully issued

                } else
                        // Book could
not be issued
                        return false;

        } catch (SQLException e1) {
                // TODO Auto-
generated catch block\

                e1.printStackTrace();
                return null;
        }
    }
    /*************** ISSUE BOOK
****************/

    /*************** RETURN BOOK
****************/
    public static Boolean
returnBook(Book book) {
        // Get the book ID
        String bookID =
book.getBookID();

        try {
                // Prepare SQL
statement to select book from database
using book ID
                pst =
con.prepareStatement("SELECT * FROM BOOK
WHERE bookID=?");
                pst.setString(1,
bookID);
                rs =
pst.executeQuery();

                // If the book exists
in the database
                if (rs.next()) {
                        // Find the
book by its ID
                        book =
findBookbyID(book.getBookID());

                // Prepare SQL
statement to update book details in the
database
                        pst =
con.prepareStatement(

        "UPDATE BOOK SET AVAILABILITY=?,
ISSUEDATE=?, DUEDATE=?,STUDENTID=? WHERE
BOOKID=?");

        pst.setString(1, "Available");

        pst.setString(2, null);

        pst.setString(3, null);

        pst.setString(4, null);

        pst.setString(5, bookID);

                        // Execute the
update query and get the number of rows
affected
                                int
rowsAffected = pst.executeUpdate(); //
Use executeUpdate() for UPDATE queries

                                // If the
update was successful
                                if
(rowsAffected > 0) {
                                        //
Update the book object's availability

        book.setAvailability("Available");
                                        //
Return true to indicate successful book
return
                                        return
true;
                                } else
                                        //
Return false to indicate unsuccessful
book return
                                        return
false;

                }

        } catch (SQLException e1) {

                // Return false to
indicate unsuccessful book return
                e1.printStackTrace();
                return false;
        }
```

```
            return null;

    }
    /*************** RETURN BOOK
*****************/

    // Validate ID Value
    public static Boolean
isValidID(String ID) {
        try {
            int idValue =
Integer.parseInt(ID);
            // Check if ID is a
positive number
            if (idValue <= 0) {
                return false;
// Return false method if validation
fails
                }
            } catch
(NumberFormatException ex) {
                // Handle invalid ID
format

                return false; //
Return false if validation fails
            }
            return true;

    }

}
```

4. <u>Connect Class for JDBC:</u>

```java
package libraryManagementSystem;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Connect {
    static Connection con = null;

    // Method to establish connection to the database
    public static Connection ConnectToDB() {
        try {
            // Establish connection to the MySQL database named "library" on
localhost at port 3306
            // with username "root" and password "MySQL@2773778"
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/library", "root",
"MySQL@2773778");
        } catch (SQLException ex) {
            // Log any SQLException that occurs during the connection attempt
            Logger.getLogger(Connect.class.getName()).log(Level.SEVERE, null,
ex);
        }
        return con; // Return the established connection (may be null if
connection fails)
    }
}
```

5. Login Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import net.miginfocom.swing.MigLayout;
import java.awt.Font;
import javax.swing.SwingConstants;
import java.awt.SystemColor;
import javax.swing.ImageIcon;
import java.awt.Color;
import java.awt.Toolkit;

public class loginPage extends
JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField
usernameField;
    private JPasswordField
passwordField;

    /**
     * Launch the application.
     */
    /*************************
MAIN METHOD
***************************/
    public static void main(String[]
args) {
        EventQueue.invokeLater(new
Runnable() {
            public void run() {
                try {
                            loginPage frame
= new loginPage();

frame.setVisible(true);
                } catch (Exception
e) {

e.printStackTrace();
                }
            }
        });
    }
    /*************************
MAIN METHOD
***************************/

    /**
     * Create the frame.
     */
    public loginPage() {

        // GUI SETUP
        setResizable(false);
        setTitle("LMS");

    setIconImage(Toolkit.getDefaultT
oolkit().getImage("C:\\Users\\Dell\\
Desktop\\book.png"));

setDefaultCloseOperation(JFrame.EXIT
_ON_CLOSE);
        setBounds(500, 250, 500,
300);
        contentPane = new JPanel();

contentPane.setBackground(SystemColo
r.inactiveCaption);
        contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        /***************** LABELS
********************/
        JLabel txtLogin = new
JLabel("L M S");
        txtLogin.setForeground(new
Color(19, 79, 151));
```

```java
txtLogin.setHorizontalAlignment(Swin
gConstants.CENTER);
        txtLogin.setFont(new
Font("Times New Roman", Font.BOLD,
20));
        txtLogin.setBounds(174, 10,
144, 31);
        contentPane.add(txtLogin);

        JLabel txtUsername = new
JLabel("Username:");

txtUsername.setForeground(new
Color(19, 79, 151));
        txtUsername.setFont(new
Font("Calibri", Font.BOLD, 19));
        txtUsername.setBounds(49,
100, 116, 31);

contentPane.add(txtUsername);

        JLabel txtPassword = new
JLabel("Password:");

txtPassword.setForeground(new
Color(19, 79, 151));
        txtPassword.setFont(new
Font("Calibri", Font.BOLD, 19));
        txtPassword.setBounds(49,
146, 105, 28);

contentPane.add(txtPassword);

        JLabel title = new
JLabel("Library Management System");
        title.setForeground(new
Color(19, 79, 151));

title.setHorizontalAlignment(SwingCo
nstants.CENTER);
        title.setFont(new
Font("Times New Roman", Font.BOLD |
Font.ITALIC, 19));
        title.setBounds(113, 41,
256, 32);
        contentPane.add(title);
        /****************** LABELS
********************/

/***************
TEXT/PASSWORD FIELDS
******************/
        usernameField = new
JTextField();
        usernameField.setBounds(175,
99, 229, 31);

contentPane.add(usernameField);

usernameField.setColumns(10);


        passwordField = new
JPasswordField();
        passwordField.setBounds(175,
144, 229, 31);

contentPane.add(passwordField);
        /***************
TEXT/PASSWORD FIELDS
******************/

        /*********************
BUTTONS ************************/
        // Close Button
        JButton closeButton = new
JButton("Close");

closeButton.setForeground(new
Color(0, 64, 128));
        closeButton.setFont(new
Font("Tahoma", Font.BOLD, 14));
        closeButton.setBounds(295,
194, 110, 25);
        // Program Execution
terminates on close button

closeButton.addActionListener(new
ActionListener() {
        public void
actionPerformed(ActionEvent e) {
                System.exit(ABORT);
        }
    });

contentPane.add(closeButton);

        // Login Button
        JButton loginButton = new
JButton("Login");
```

```java
		loginButton.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\login.png"));

loginButton.setForeground(new
Color(0, 64, 128));
		loginButton.setFont(new
Font("Tahoma", Font.BOLD, 14));
		loginButton.setBounds(175,
194, 110, 25);

loginButton.addActionListener(new
ActionListener() {
			public void
actionPerformed(ActionEvent e) {
				// Get the entered
username and password
				String username =
usernameField.getText();
				char[] passwordChars =
passwordField.getPassword();
				String password = new
String(passwordChars);

				// Perform
authentication or validation here
				// Check if the
entered username and password match
any of the predefined credentials
				if
(username.equals("Admin1") &&
password.equals("library123") ||

username.equals("Admin2") &&
password.equals("library456") ||

username.equals("Admin3") &&
password.equals("library789")) {

					// If the
credentials match, open the home
page
					setVisible(false);
					new
homePage().setVisible(true);
				} else {
```

```java
					// If the
credentials don't match, display an
error message

JOptionPane.showMessageDialog(null,
"Invalid Username or Password",
"Error", JOptionPane.ERROR_MESSAGE);
				}
			}
		});

contentPane.add(loginButton);

		/********************
BUTTONS ************************/


		/*****************
BACKGROUND IMAGE
********************/
		JLabel lblNewLabel = new
JLabel("");

lblNewLabel.setHorizontalAlignment(S
wingConstants.CENTER);
		lblNewLabel.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\book.png"));
		lblNewLabel.setBounds(164, -
21, 70, 87);

contentPane.add(lblNewLabel);

		JLabel background = new
JLabel("");
		background.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\R (1).jpg"));
		background.setBounds(-13, -
18, 522, 302);
		contentPane.add(background);
		/*****************
BACKGROUND IMAGE
********************/
	}
}
```

6. Home Page:

```java
package libraryManagementSystem;

import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Toolkit;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.ImageIcon;
import java.awt.Font;
import java.awt.Color;
import javax.swing.SwingConstants;
import java.awt.SystemColor;

public class homePage extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;

    /**
     * Launch the application.
     */
    /***************************
    MAIN METHOD
    ***************************/
    public static void main(String[] args) {
            EventQueue.invokeLater(new Runnable() {
                    public void run() {
                            try {

    homePage frame = new homePage();

    frame.setVisible(true);
                            } catch (Exception e) {

    e.printStackTrace();
                            }
                    }
```

```java
            });
    }

    /***************************
    MAIN METHOD
    ***************************/

    /**
     * Create the frame.
     */
    public homePage() {
            // GUI SETUP
            setTitle("LMS");

    setIconImage(Toolkit.getDefaultToolkit()

    .getImage("C:\\Users\\Dell\\Desktop\\Library Management
    System\\Images\\book.png"));

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setExtendedState(JFrame.MAXIMIZED_BOTH);
            Dimension screenSize =
    Toolkit.getDefaultToolkit().getScreenSize();
            int screenWidth =
    screenSize.width;
            int screenHeight =
    screenSize.height;

            int width = 1100;
            int height = 700;

            int x = (screenWidth -
    width) / 2;
            int y = (screenHeight -
    height) / 2;

            setBounds(1, 1,
    screenWidth, screenHeight);
            contentPane = new
    JPanel();
            contentPane.setBorder(new
    EmptyBorder(5, 5, 5, 5));
```

```java
        setContentPane(contentPane);

        contentPane.setLayout(null);

        /********************
BUTTONS ************************/

        // Each button opens
corresponding window when pressed
        JButton addUserButton =
new JButton("Add User");
        addUserButton
                        .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\adduser.png"));

    addUserButton.setForeground(new
Color(0, 64, 128));

    addUserButton.addActionListener(
new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
addUserWindow().setVisible(true);
                }
        });
        addUserButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    addUserButton.setBounds(95, 48,
146, 40);

    contentPane.add(addUserButton);

        JButton searchBookButton =
new JButton("Search Book");

    searchBookButton.setForeground(n
ew Color(0, 64, 128));
        searchBookButton
                        .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\datamining.png"));

    searchBookButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    searchBookButton.addActionListen
er(new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
searchBookWindow().setVisible(true);
                }
        });

    searchBookButton.setBounds(575,
48, 154, 40);

    contentPane.add(searchBookButton
);

        JButton addBookButton =
new JButton("Add Book");

    addBookButton.setForeground(new
Color(0, 64, 128));
        addBookButton
                        .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\addbook.png"));

    addBookButton.addActionListener(
new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
addBookWindow().setVisible(true);
                }
        });
        addBookButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    addBookButton.setBounds(338, 48,
146, 40);

    contentPane.add(addBookButton);

        JButton displayBooksButton
= new JButton("Display Books");
        displayBooksButton
                        .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\display.png"));
```

```java
    displayBooksButton.setForeground
(new Color(0, 64, 128));

    displayBooksButton.addActionList
ener(new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
displayBooksWindow().setVisible(true
);
                }
        });
    displayBooksButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    displayBooksButton.setBounds(808
, 48, 165, 40);

    contentPane.add(displayBooksButt
on);

        JButton issueBookButton =
new JButton("Issue Book");
        issueBookButton
                .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\issuebook.png"));

    issueBookButton.setForeground(ne
w Color(0, 64, 128));

    issueBookButton.addActionListene
r(new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
issueBookWindow().setVisible(true);
                }
        });
    issueBookButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    issueBookButton.setBounds(1057,
48, 146, 40);

    contentPane.add(issueBookButton)
;

        JButton returnBookButton =
new JButton("Return Book");

    returnBookButton.setForeground(n
ew Color(0, 64, 128));

    returnBookButton.addActionListen
er(new ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        new
returnBookWindow().setVisible(true);
                }
        });
        returnBookButton
                .setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\return.png"));

    returnBookButton.setFont(new
Font("Tahoma", Font.BOLD, 14));

    returnBookButton.setBounds(1293,
48, 154, 40);

    contentPane.add(returnBookButton
);

        JButton logoutButton = new
JButton("Logout");

    logoutButton.setForeground(new
Color(0, 64, 128));
        logoutButton.setFont(new
Font("Tahoma", Font.BOLD, 14));
        logoutButton.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\logout.png"));

    logoutButton.addActionListener(n
ew ActionListener() {
                public void
actionPerformed(ActionEvent e) {

    setVisible(false);
                        new
loginPage().setVisible(true);
                }
```

```java
        });

    logoutButton.setBounds(1322,
696, 146, 40);

    contentPane.add(logoutButton);
        /********************
BUTTONS ***********************/

        /******************
BACKGROUND **********************/
        JLabel lblTItle = new
JLabel("L M S");
        lblTItle.setForeground(new
Color(0, 64, 128));

    lblTItle.setHorizontalAlignment(
SwingConstants.CENTER);
        lblTItle.setFont(new
Font("Times New Roman", Font.BOLD,
40));
        lblTItle.setBounds(492,
210, 574, 90);
        contentPane.add(lblTItle);

        JLabel bookimg = new
JLabel("");

    bookimg.setHorizontalAlignment(S
wingConstants.CENTER);
        bookimg.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\LMSbook.png"));
        bookimg.setBounds(703,
114, 146, 110);
        contentPane.add(bookimg);

        JLabel lblTitle1 = new
JLabel("Library Management System");

    lblTitle1.setForeground(new
Color(0, 64, 128));

    lblTitle1.setHorizontalAlignment
(SwingConstants.CENTER);
        lblTitle1.setFont(new
Font("Times New Roman", Font.BOLD |
Font.ITALIC, 30));
        lblTitle1.setBounds(510,
296, 546, 46);

    contentPane.add(lblTitle1);

        JLabel lblQuote = new
JLabel("There's no friend as loyal
as a book!");

    lblQuote.setForeground(SystemCol
or.inactiveCaption);
        lblQuote.setFont(new
Font("Times New Roman", Font.BOLD |
Font.ITALIC, 20));

    lblQuote.setHorizontalAlignment(
SwingConstants.CENTER);
        lblQuote.setBounds(510,
696, 555, 40);
        contentPane.add(lblQuote);

        JLabel background = new
JLabel("");
        background.setIcon(
                new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\bluebackground.jpg")
);
        background.setBounds(0,
10, 1565, 893);

    contentPane.add(background);
    }
}
```

7. Add User Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.Toolkit;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import java.awt.Font;
import javax.swing.DefaultComboBoxModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Color;
import javax.swing.SwingConstants;
import java.awt.SystemColor;


public class addUserWindow extends
JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField
studentNameField;
    private JTextField
studentIDField;
    private JTextField contactField;

    /**
     * Launch the application.
     */

    /******************** MAIN
METHOD **************************/
    public static void main(String[]
args) {
        EventQueue.invokeLater(new
Runnable() {
            public void run() {
                try {
                    addUserWindow frame = new
addUserWindow();

                    frame.setVisible(true);
                                } catch
(Exception e) {

                    e.printStackTrace();
                                }
                        }
                });
        }

    /********************** MAIN
METHOD **************************/

    /**
     * Create the frame.
     */
    public addUserWindow() {

            // Window GUI
            setResizable(false);
            setTitle("Add User");

        setIconImage(Toolkit.getDefaultT
oolkit()

        .getImage("C:\\Users\\Dell\\Desk
top\\Library Management
System\\Images\\adduser.png"));

        setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
            setBounds(350, 150, 800,
530);
            contentPane = new
JPanel();
            contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));


        setContentPane(contentPane);

        contentPane.setLayout(null);

            /******************
LABELS *******************/
```

```java
        JLabel lblID = new
JLabel("Student ID:");
            lblID.setForeground(new
Color(0, 64, 128));
            lblID.setFont(new
Font("Calibri", Font.BOLD, 19));
            lblID.setBounds(165, 191,
169, 43);
            contentPane.add(lblID);

        JLabel lblStudentName =
new JLabel("Student Name:");

        lblStudentName.setForeground(new
Color(0, 64, 128));
            lblStudentName.setFont(new
Font("Calibri", Font.BOLD, 19));

        lblStudentName.setBounds(165,
131, 158, 43);

        contentPane.add(lblStudentName);

        JLabel lblcontact = new
JLabel("Contact No.:");

        lblcontact.setForeground(new
Color(0, 64, 128));
            lblcontact.setFont(new
Font("Calibri", Font.BOLD, 19));
            lblcontact.setBounds(165,
255, 130, 43);

        contentPane.add(lblcontact);

        JLabel lblcourse = new
JLabel("Course:");

        lblcourse.setForeground(new
Color(0, 64, 128));
            lblcourse.setFont(new
Font("Calibri", Font.BOLD, 19));
            lblcourse.setBounds(165,
321, 90, 43);

        contentPane.add(lblcourse);

        JLabel lblAddUser = new
JLabel("   ADD USER");

        lblAddUser.setForeground(new
Color(0, 64, 128));
            lblAddUser.setFont(new
Font("Tahoma", Font.BOLD, 18));

    lblAddUser.setHorizontalAlignmen
t(SwingConstants.CENTER);
            lblAddUser.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\\Li
brary Management
System\\Images\\adduser1.png"));
            lblAddUser.setBounds(160,
10, 420, 90);

        contentPane.add(lblAddUser);
        /****************
LABELS ********************/

        /**************** TEXT
FIELDS *****************/
            studentNameField = new
JTextField();

        studentNameField.setBackground(S
ystemColor.info);

        studentNameField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

        studentNameField.setBounds(319,
135, 261, 31);

        contentPane.add(studentNameField
);

        studentNameField.setColumns(10);

        studentIDField = new
JTextField();

        studentIDField.setBackground(Sys
temColor.info);
            studentIDField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

        studentIDField.setBounds(319,
195, 261, 31);

        contentPane.add(studentIDField);
```

```java
        studentIDField.setColumns(10);

            contactField = new
JTextField();

        contactField.setBackground(Syste
mColor.info);
            contactField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

        contactField.setBounds(319, 260,
261, 31);

        contentPane.add(contactField);

        contactField.setColumns(10);
            /**************** TEXT
FIELDS ******************/

            /**************** COMBO
BOX ******************/
            JComboBox courseBox = new
JComboBox();

        courseBox.setBackground(SystemCo
lor.control);
            courseBox.setFont(new
Font("Tahoma", Font.PLAIN, 14));
            courseBox.setModel(new
DefaultComboBoxModel(new String[] {
"BESE", "BSDS", "BSCS", "BEE" }));

        courseBox.setMaximumRowCount(4);
            courseBox.setBounds(319,
325, 261, 30);

        contentPane.add(courseBox);
            /**************** COMBO
BOX ******************/

            /**************** BUTTONS
******************/
            JButton btnAddUser = new
JButton("Add User");

        btnAddUser.setForeground(new
Color(0, 64, 128));
            btnAddUser.setFont(new
Font("Tahoma", Font.BOLD, 15));

            // Giving action to add
user button

        btnAddUser.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {

                    // Retrieving
the user information from the text
fields
                    String
studentName =
studentNameField.getText();
                    String
studentID = studentIDField.getText();
                    String
contactNo = contactField.getText();
                    String course
=
courseBox.getSelectedItem().toString();

                    // Removing
nay spaces in the contact number or ID.
                    contactNo =
contactNo.replace(" ", "");
                    studentID =
studentID.replace(" ", "");


    /********************** ERROR
HANDLING ************************/

                    // Checking
if all fields have been filled
                    if
(studentName.isEmpty() ||
studentID.isEmpty() ||
contactNo.isEmpty() ||
course.isEmpty()) {

    JOptionPane.showMessageDialog(nu
ll, "Please Fill in All FIelds",
"Error",

    JOptionPane.ERROR_MESSAGE);
                    } else {
                        //
Perform additional validations
```

```java
                            // Validate ID (it should be numeric)
                            if (!Library.isValidID(studentID)) {

        JOptionPane.showMessageDialog(null, "Invalid ID Value.", "Error",
JOptionPane.ERROR_MESSAGE);

        return;
                            }

                            // Validate contact number (it should be numeric)
                            try {

        // Converts the contact number string to a long value

        long contactValue = Long.parseLong(contactNo);

        // Check if contact number is a positive number and has a valid length

        if (contactValue <= 0 || contactNo.length() != 11) {

        JOptionPane.showMessageDialog(null, "Invalid contact number.", "Error",

        JOptionPane.ERROR_MESSAGE);

        return; // Exit method if validation fails

        }
                            }
catch (NumberFormatException ex) {

        // Handle invalid contact number, if it has characters other than digits

        JOptionPane.showMessageDialog(null, "Contact number must be a numeric value.", "Error",

        JOptionPane.ERROR_MESSAGE);

        return; // Exit method if validation fails
                            }

                            // Validation of Student name
                            try {

        // Check if students name contains any numbers

        if (studentName.matches(".*\\d.*")) {

        throw new IllegalArgumentException();

        }

                            }
catch (IllegalArgumentException e1) {

        // Display error message dialog box

        JOptionPane.showMessageDialog(null,

            "Error: Username is not valid. Please provide a valid username.", "Error",

        JOptionPane.ERROR_MESSAGE);

        return;
                            }

                            }

    /********************* ERROR HANDLING ***********************/
```

```java
                        // Creating
an Object of the User as per the
provided details
                        User userX =
new User();

    userX.setUserName(studentName);

    userX.setUserID(studentID);

    userX.setContactNo(contactNo);

    userX.setCourseName(course);

                        // Adding
user to the database
                        if
(Library.addUserToDB(userX)) {

    JOptionPane.showMessageDialog(ro
otPane, "Record Saved", "Saved",
JOptionPane.INFORMATION_MESSAGE);

    clear();
                        } else
                                //
Error message if the student already
exists based in the Student ID

    JOptionPane.showMessageDialog(nu
ll, "Record Already Exists", "Error",
JOptionPane.ERROR_MESSAGE);

                    }

            });
            btnAddUser.setBounds(319,
410, 124, 31);

    contentPane.add(btnAddUser);

        // Close button to dispose
the window
            JButton btnClose = new
JButton("Close");

    btnClose.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent e) {
                            dispose();
                    }
            });
            btnClose.setForeground(new
Color(0, 64, 128));
            btnClose.setFont(new
Font("Tahoma", Font.BOLD, 15));
            btnClose.setBounds(456,
410, 124, 31);
            contentPane.add(btnClose);
            /**************** BUTTONS
********************/


            /**************
BACKGROUND *****************/
            JLabel background = new
JLabel("");
            background.setIcon(
                        new
ImageIcon("C:\\Users\\Dell\\Desktop\\Li
brary Management
System\\Images\\windowbackground.jpg"))
;
            background.setBounds(0, 0,
801, 505);

    contentPane.add(background);
            /**************
BACKGROUND *****************/
    }

    // Method to clear all fields
    public void clear() {

    studentIDField.setText("");

    studentNameField.setText("");
            contactField.setText("");

    }
}
```

8. Add Book Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.Toolkit;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.SwingConstants;
import java.awt.Font;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
import java.awt.SystemColor;

public class addBookWindow extends
JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField bookIDField;
    private JTextField authorField;
    private JTextField genreField;
    private JLabel lblBookTItle;
    private JTextField
bookNameField;
    private JLabel lblBackground;

    /**
     * Launch the application.
     */

    /*************************
MAIN METHOD
*****************************/
    public static void main(String[]
args) {
            EventQueue.invokeLater(new
Runnable() {

            public void run() {
                    try {

    addBookWindow frame = new
addBookWindow();

    frame.setVisible(true);
                            } catch
(Exception e) {

    e.printStackTrace();
                            }
                    }
            });
    }
    /**************************
MAIN METHOD
****************************/

    /**
     * Create the frame.
     */
    public addBookWindow() {
            // GUI Setup

    setIconImage(Toolkit.getDefaultT
oolkit().getImage("C:\\Users\\Dell\\
Desktop\\Library Management
System\\Images\\addbook.png"));

    setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
            setBounds(350, 150, 800,
530);
            contentPane = new
JPanel();
            contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));


    setContentPane(contentPane);

    contentPane.setLayout(null);

            /******************
LABELS *********************/
            JLabel lblTitle = new
JLabel("   ADD BOOK");
            lblTitle.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
```

```
\Library Management
System\\Images\\addbook1.png"));
        lblTitle.setForeground(new
Color(0, 64, 128));
        lblTitle.setFont(new
Font("Tahoma", Font.BOLD, 18));

    lblTitle.setHorizontalAlignment(
SwingConstants.CENTER);
        lblTitle.setBounds(165,
35, 415, 69);
        contentPane.add(lblTitle);

        lblBookTItle = new
JLabel("Book Title:");
        lblBookTItle.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblBookTItle.setForeground(new
Color(0, 64, 128));

    lblBookTItle.setBounds(165, 138,
158, 43);

    contentPane.add(lblBookTItle);

        JLabel lblID = new
JLabel("Book ID:");
        lblID.setForeground(new
Color(0, 64, 128));
        lblID.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblID.setBounds(165, 201,
90, 31);
        contentPane.add(lblID);

        JLabel lblAuthor = new
JLabel("Book Author:");

    lblAuthor.setForeground(new
Color(0, 64, 128));
        lblAuthor.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblAuthor.setBounds(165,
254, 129, 38);

    contentPane.add(lblAuthor);

        JLabel lblGenre = new
JLabel("Genre");

        lblGenre.setForeground(new
Color(0, 64, 128));
        lblGenre.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblGenre.setBounds(165,
316, 129, 31);
        contentPane.add(lblGenre);
        /******************
LABELS *********************/

        /************* TEXT
FIELDS *********************/
        bookIDField = new
JTextField();
        bookIDField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookIDField.setBackground(System
Color.info);
        bookIDField.setBounds(319,
200, 261, 31);

    contentPane.add(bookIDField);

    bookIDField.setColumns(10);

        authorField = new
JTextField();
        authorField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    authorField.setBackground(System
Color.info);
        authorField.setBounds(319,
257, 261,31);

    contentPane.add(authorField);

    authorField.setColumns(10);

        genreField = new
JTextField();
        genreField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    genreField.setBackground(SystemC
olor.info);
        genreField.setBounds(319,
315, 261, 31);

    contentPane.add(genreField);
```

```java
        genreField.setColumns(10);

        bookNameField = new
JTextField();
        bookNameField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookNameField.setBackground(Syst
emColor.info);

    bookNameField.setBounds(319,
144, 261, 31);

    contentPane.add(bookNameField);

    bookNameField.setColumns(10);
        /************** TEXT
FIELDS *********************/

        /**************** BUTTONS
*********************/
        // Add Book Button
        JButton btnAdddBook = new
JButton("Add Book");

        // Button Actions

    btnAdddBook.addActionListener(ne
w ActionListener() {
                public void
actionPerformed(ActionEvent e) {

                    // Retrieving
the user information from the text
fields
                    String
bookName = bookNameField.getText();
                    String bookID
= bookIDField.getText();
                    String author
= authorField.getText();
                    String genre=
genreField.getText();

                    // Removing
nay spaces in the book ID.

    bookID=bookID.replace(" ", "");


    /*************************
ERROR HANDLING
************************/
                    if
(bookName.isEmpty() ||
bookID.isEmpty() || author.isEmpty()
|| genre.isEmpty()) {

    JOptionPane.showMessageDialog(nu
ll, "Please Fill in All FIelds",
"Error",

    JOptionPane.ERROR_MESSAGE);
                    } else {
                        //
Perform additional validations

                        //
Validate ID (it should be numeric)

    if(!Library.isValidID(bookID)) {

    JOptionPane.showMessageDialog(nu
ll, "Invalid ID Value.", "Error",

    JOptionPane.ERROR_MESSAGE);

    return;
                        }

                        try {
                            //
Check if genre and author's name
contains any numbers
                            if
(author.matches(".*\\d.*")||genre.ma
tches(".*\\d.*")) {

throw new
IllegalArgumentException();
                            }

                        } catch (
IllegalArgumentException e1) {
                            //
Display error message dialog box for
invalid author's name

JOptionPane.showMessageDialog(null,
```

```java
            "Error: Genre/Author Name is not
valid.", "Error",
JOptionPane.ERROR_MESSAGE);
                            return;
                        }

                        //
Creating an Object of the Book as
per the provided details
                            Book
bookX = new Book();

    bookX.setBookTitle(bookName);

    bookX.setBookID(bookID);

    bookX.setAuthor(author);

    bookX.setGenre(genre);

    bookX.setAvailability("Available
");

                        //
Adding book to the database
                            if
(Library.addBookToDB(bookX)) {

    JOptionPane.showMessageDialog(ro
otPane, "Record Saved", "Saved",
JOptionPane.INFORMATION_MESSAGE);

    clear();
                        }
                        //
Error message if the Book already
exists based on the Book ID
                            else

    JOptionPane.showMessageDialog(nu
ll, "Record Already Exists",
"Error", JOptionPane.ERROR_MESSAGE);


                    }
                }
            });

    btnAdddBook.setForeground(new
Color(0, 64, 128));

        btnAdddBook.setFont(new
Font("Tahoma", Font.BOLD, 15));
        btnAdddBook.setBounds(319,
410, 124, 31);

    contentPane.add(btnAdddBook);

        // Dispose Window on close
button
        JButton btnClose = new
JButton("Close");

    btnClose.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                    dispose();

    setVisible(false);
                }
        });
        btnClose.setFont(new
Font("Tahoma", Font.BOLD, 15));
        btnClose.setForeground(new
Color(0, 64, 128));
        btnClose.setBounds(456,
410, 124, 31);
        contentPane.add(btnClose);
        /*************** BUTTONS
*********************/


        /**************
BACKGROUND ******************/
        lblBackground = new
JLabel("");
        lblBackground.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\windowbackground.jpg
"));
        lblBackground.setBounds(0,
0, 786, 493);

    contentPane.add(lblBackground);
        /**************
BACKGROUND ******************/
    }


    public void clear() {
        bookIDField.setText("");
```

```java
            bookNameField.setText("");
            authorField.setText("");
            genreField.setText("");
```

9. Search Book Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.Toolkit;
import javax.swing.SwingConstants;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JToggleButton;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
import java.awt.SystemColor;

public class searchBookWindow
extends JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField
bookTitleField;
    private JTextField bookIDField;
    private JTextField
bookAuthorField;
    private JTextField
bookGenreField;
    private JTextField
availabilityField;

    /**
```

```java
    }
}

     * Launch the application.
     */

    /*************** MAIN METHOD
********************/
    public static void main(String[]
args) {
            EventQueue.invokeLater(new
Runnable() {
                    public void run() {
                            try {

    searchBookWindow frame = new
searchBookWindow();

    frame.setVisible(true);
                            } catch
(Exception e) {

    e.printStackTrace();
                            }
                    }
            });
    }
    /*************** MAIN METHOD
********************/

    /**
     * Create the frame.
     */
    public searchBookWindow() {

            // GUI SETUP

    setIconImage(Toolkit.getDefaultT
oolkit()

    .getImage("C:\\Users\\Dell\\Desk
top\\Library Management
System\\Images\\datamining.png"));

    setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
            setBounds(350, 150, 800,
530);
```

```java
        contentPane = new
JPanel();
        contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));


    setContentPane(contentPane);

    contentPane.setLayout(null);

        /***************** LABELS
*****************/
        JLabel lblSearchBook = new
JLabel("   SEARCH BOOK");
        lblSearchBook.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\search.png"));

    lblSearchBook.setForeground(new
Color(0, 64, 128));
        lblSearchBook.setFont(new
Font("Tahoma", Font.BOLD, 18));

    lblSearchBook.setHorizontalAlign
ment(SwingConstants.CENTER);

    lblSearchBook.setBounds(278, 44,
239, 66);

    contentPane.add(lblSearchBook);

        JLabel lblBookTitle = new
JLabel("Book Title:");

    lblBookTitle.setForeground(new
Color(0, 64, 128));
        lblBookTitle.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblBookTitle.setBounds(169, 144,
123, 24);

    contentPane.add(lblBookTitle);

        JLabel lblBookID = new
JLabel("Book ID:");

    lblBookID.setForeground(new
Color(0, 64, 128));

        lblBookID.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblBookID.setBounds(169,
202, 103, 24);

    contentPane.add(lblBookID);

        JLabel lblAuthor = new
JLabel("Book Author:");

    lblAuthor.setForeground(new
Color(0, 64, 128));
        lblAuthor.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblAuthor.setBounds(169,
260, 123, 24);

    contentPane.add(lblAuthor);

        JLabel lblGenre = new
JLabel("Genre:");
        lblGenre.setForeground(new
Color(0, 64, 128));
        lblGenre.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblGenre.setBounds(169,
316, 123, 35);
        contentPane.add(lblGenre);

        JLabel lblAvailability =
new JLabel("Availability");

    lblAvailability.setForeground(ne
w Color(0, 64, 128));

    lblAvailability.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblAvailability.setBounds(169,
372, 103, 35);

    contentPane.add(lblAvailability)
;
        /***************** LABELS
*****************/

        /************* TEXT
FIELDS ***************/
        bookTitleField = new
JTextField();
```

```java
        bookTitleField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookTitleField.setBackground(Sys
temColor.info);

    bookTitleField.setBounds(302,
140, 255, 31);

    contentPane.add(bookTitleField);

    bookTitleField.setColumns(10);
        bookIDField = new
JTextField();
        bookIDField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookIDField.setBackground(System
Color.controlHighlight);

    bookIDField.setEditable(false);
        bookIDField.setBounds(302,
198, 255, 31);

    contentPane.add(bookIDField);

    bookIDField.setColumns(10);

        bookAuthorField = new
JTextField();

    bookAuthorField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookAuthorField.setBackground(Sy
stemColor.controlHighlight);

    bookAuthorField.setEditable(fals
e);

    bookAuthorField.setBounds(302,
256, 255, 31);

    contentPane.add(bookAuthorField)
;

    bookAuthorField.setColumns(10);

        bookGenreField = new
JTextField();

        bookGenreField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookGenreField.setBackground(Sys
temColor.controlHighlight);

    bookGenreField.setEditable(false
);

    bookGenreField.setBounds(302,
317, 255, 31);

    contentPane.add(bookGenreField);

    bookGenreField.setColumns(10);

        availabilityField = new
JTextField();

    availabilityField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    availabilityField.setBackground(
SystemColor.controlHighlight);

    availabilityField.setEditable(fa
lse);

    availabilityField.setBounds(302,
373, 255, 31);

    contentPane.add(availabilityFiel
d);

    availabilityField.setColumns(10)
;
        /************** TEXT
FIELDS ***************/

        /************** BUTTONS
****************/
        JButton btnSearch = new
JButton("Search");

    btnSearch.setBackground(new
Color(176, 196, 222));
        btnSearch.setFont(new
Font("Tahoma", Font.BOLD, 15));

    btnSearch.addActionListener(new
ActionListener() {
```

```java
					public void
actionPerformed(ActionEvent e) {
						String
bookName = bookTitleField.getText();

						// Checks if
book name has been entered or not
						if
(bookName.isEmpty()) {

	JOptionPane.showMessageDialog(nu
ll, "Please Add Book Name", "Error",
JOptionPane.ERROR_MESSAGE);
						} else {
							// If
the book name field is not empty,
proceed to search for the book
							//
Search for the book by its title
							Book
foundBook = new Book();

	foundBook =
Library.findBookbyTitle(bookName);

							//
Check if the book is found
							if
(foundBook != null) {

	// If the book is found,
populate the fields with the book
details

	bookIDField.setText(foundBook.ge
tBookID());

	bookAuthorField.setText(foundBoo
k.getAuthor());

	bookGenreField.setText(foundBook
.getGenre());

	availabilityField.setText(foundB
ook.getAvailability());
							} else
{

	// If the book is not found,
display an error message

	JOptionPane.showMessageDialog(nu
ll, "Book not Found.", "Error",
JOptionPane.ERROR_MESSAGE);
							}
						}
					}
				});
				btnSearch.setBounds(598,
138, 103, 31);

				// Reset button is used to
clear all text fields
				JButton btnReset = new
JButton("Reset");
				btnReset.setForeground(new
Color(0, 64, 128));
				btnReset.setFont(new
Font("Tahoma", Font.BOLD, 15));

	btnReset.addActionListener(new
ActionListener() {
					public void
actionPerformed(ActionEvent e) {
						clear();
					}
				});
				btnReset.setBounds(302,
433, 103, 31);
				contentPane.add(btnReset);

				// CLose button to dispose
the window
				JButton btnClose = new
JButton("Close");
				btnClose.setForeground(new
Color(0, 64, 128));
				btnClose.setFont(new
Font("Tahoma", Font.BOLD, 15));

	btnClose.addActionListener(new
ActionListener() {
					public void
actionPerformed(ActionEvent e) {
						dispose();
					}
				});
				btnClose.setBounds(454,
433, 103, 31);
				contentPane.add(btnClose);
```

```
        contentPane.add(btnSearch);
                /************** BUTTONS
****************/

                /***************
BACKGROUND *************/
                JLabel background = new
JLabel("");
                background.setIcon(
                        new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\windowbackground.jpg
"));
                background.setBounds(0, 0,
786, 493);

        contentPane.add(background);
```

10.   Display Books Page:

```
package libraryManagementSystem;

import java.awt.EventQueue;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import
javax.swing.border.EmptyBorder;
import java.awt.Toolkit;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.SwingConstants;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JTable;
import
javax.swing.table.DefaultTableModel;

import net.proteanit.sql.DbUtils;

public class displayBooksWindow
extends JFrame {
```

```
                /****************
BACKGROUND *************/
    }

    // Method to Clear all Fields
    public void clear() {

    bookTitleField.setText("");
            bookIDField.setText("");

    bookAuthorField.setText("");

    bookGenreField.setText("");

    availabilityField.setText("");
    }
}
```

```
    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;

    // Establish connection with
database
    Connection con =
Connect.ConnectToDB();
    PreparedStatement pst=null;
    ResultSet rs=null;
    private JTable table_book;
    private JScrollPane scrollPane;


    /**
     * Launch the application.
     */

    /*************   MAIN METHOD
****************/
    public static void main(String[]
args) {
            EventQueue.invokeLater(new
Runnable() {
                    public void run() {
                            try {

    displayBooksWindow frame = new
displayBooksWindow();
```

```java
            frame.setVisible(true);
                        } catch
(Exception e) {

    e.printStackTrace();
                        }
                    }
            });
    }
    /*************** MAIN METHOD
****************/

    // Function to update the table
by taking information from the
database book
    private void UpdateTable() {
        try {
            // Prepare the SQL query
to select data from the book table
            pst =
con.prepareStatement("SELECT
bookTitle, bookID, author, genre,
availability FROM book");

            // Execute the query and
retrieve the ResultSet
            rs = pst.executeQuery();

            // Update the table_book
JTable with the fetched data

table_book.setModel(DbUtils.resultSe
tToTableModel(rs));

        } catch (SQLException e) {
            // Handle SQLException

JOptionPane.showMessageDialog(rootPa
ne, "", "",
JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        }
    }


    /**
     * Create the frame.
     */
    public displayBooksWindow() {

        // GUI SETUP

    setIconImage(Toolkit.getDefaultT
oolkit().getImage("C:\\Users\\Dell\\
Desktop\\Library Management
System\\Images\\display.png"));

    setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
        setBounds(350, 150, 800,
530);
        contentPane = new
JPanel();
        contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));


    setContentPane(contentPane);

    contentPane.setLayout(null);

        /**************** LABELS
***************/
        JLabel lblTitle = new
JLabel("    DISPLAY BOOKS");
        lblTitle.setForeground(new
Color(0, 64, 128));
        lblTitle.setFont(new
Font("Tahoma", Font.BOLD, 18));
        lblTitle.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\displaybook1.png"));

    lblTitle.setHorizontalAlignment(
SwingConstants.CENTER);
        lblTitle.setBounds(243,
39, 284, 71);
        contentPane.add(lblTitle);

        JLabel lblBookTitle = new
JLabel("Book Title");
        lblBookTitle.setFont(new
Font("Tahoma", Font.BOLD, 12));

    lblBookTitle.setBounds(107, 125,
68, 13);

    contentPane.add(lblBookTitle);
```

```java
        JLabel lblBookID = new
JLabel("Book ID");
        lblBookID.setFont(new
Font("Tahoma", Font.BOLD, 12));
        lblBookID.setBounds(230,
125, 63, 13);

    contentPane.add(lblBookID);

        JLabel lblAuthor = new
JLabel("Author");
        lblAuthor.setFont(new
Font("Tahoma", Font.BOLD, 12));
        lblAuthor.setBounds(369,
125, 62, 13);

    contentPane.add(lblAuthor);

        JLabel lblGenre = new
JLabel("Genre");
        lblGenre.setFont(new
Font("Tahoma", Font.BOLD, 12));
        lblGenre.setBounds(493,
125, 68, 13);
        contentPane.add(lblGenre);

        JLabel lblAvailability =
new JLabel("Availability");

    lblAvailability.setFont(new
Font("Tahoma", Font.BOLD, 12));

    lblAvailability.setHorizontalAli
gnment(SwingConstants.CENTER);

    lblAvailability.setBounds(598,
125, 76, 13);

    contentPane.add(lblAvailability)
;
        /**************** LABELS
***************/

        //SCROLL PANE
        scrollPane = new
JScrollPane();
        scrollPane.setBounds(76,
148, 628, 221);

    contentPane.add(scrollPane);


        /***************** TABLE
*******************/
        table_book = new JTable();

    scrollPane.setColumnHeaderView(t
able_book);
        table_book.setModel(new
DefaultTableModel(
            new Object[][] {
                {"Java- How
to Program", "1", "Paul & Dietel",
"Education", "Available"},
                {"Harry
Potter", "2", "JK Rowling",
"Fantasy", "Available"},
                {"The Hunger
Games", "3", "Suzanne Collins",
"Fiction", "Available"},
                {"Pride and
Prejudice", "5", "Jane Austen",
"Romance", "Available"},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
                {null, null, null,
null, null},
```

```java
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                    {null, null, null,
null, null},
                                },
                new String[] {
                        "bookTitle",
"bookID", "author", "genre",
"availability"
                }
        ));

    table_book.getColumnModel().getC
olumn(0).setPreferredWidth(125);

    table_book.getColumnModel().getC
olumn(1).setPreferredWidth(55);

    table_book.getColumnModel().getC
olumn(2).setPreferredWidth(100);
            /***************** TABLE
*******************/


            /***************
BACKGROUND  ****************/
            JLabel lblBackground = new
JLabel("");
            lblBackground.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\\Library
Management
System\\Images\\windowbackground.jpg"));
            lblBackground.setBounds(0, 0,
786, 512);
            contentPane.add(lblBackground);
            /*************** BACKGROUND
****************/

            // Call the update table method
to fill the rows of the table
            UpdateTable();

    }
}
```

11. Issue Book Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import java.awt.Toolkit;
import java.awt.Font;
import java.awt.Color;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.awt.event.ActionEvent;
import java.awt.SystemColor;

public class issueBookWindow extends
JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField
bookNameField;
    private JTextField bookIDField;
    private JTextField
studentIDField;
    private JTextField
issueDateField;
    private JTextField dueDateField;
    private JTextField
studentNameField;

    /**
     * Launch the application.
     */

    /************** MAIN METHOD
**************/
    public static void main(String[]
args) {

            EventQueue.invokeLater(new
Runnable() {
                    public void run() {
                            try {

    issueBookWindow frame = new
issueBookWindow();

    frame.setVisible(true);
                            } catch
(Exception e) {

    e.printStackTrace();
                            }
                    }
            });
    }

    /************** MAIN METHOD
**************/
    /**
     * Create the frame.
     */

    public issueBookWindow() {
            // GUI SETUP

    setIconImage(Toolkit.getDefaultT
oolkit()

    .getImage("C:\\Users\\Dell\\Desk
top\\Library Management
System\\Images\\issuebook.png"));

    setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
            setBounds(350, 150, 800,
530);
            contentPane = new
JPanel();
            contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));


    setContentPane(contentPane);
```

```java
        contentPane.setLayout(null);

        /*************** LABELS
***************/
        JLabel lblTitle = new
JLabel("    ISSUE BOOK");

    lblTitle.setHorizontalAlignment(
SwingConstants.CENTER);
        lblTitle.setForeground(new
Color(0, 64, 128));
        lblTitle.setFont(new
Font("Tahoma", Font.BOLD, 18));
        lblTitle.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\issuebook1.png"));
        lblTitle.setBounds(178,
40, 431, 83);
        contentPane.add(lblTitle);

        JLabel lblBookTitle = new
JLabel("Book Title:");

    lblBookTitle.setForeground(new
Color(0, 64, 128));
        lblBookTitle.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblBookTitle.setBounds(162, 193,
102, 24);

    contentPane.add(lblBookTitle);

        JLabel lblBookID = new
JLabel("Book ID:");

    lblBookID.setForeground(new
Color(0, 64, 128));
        lblBookID.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblBookID.setBounds(162,
142, 102, 24);

    contentPane.add(lblBookID);

        JLabel lblStudentID = new
JLabel("Student ID:");

    lblStudentID.setForeground(new
Color(0, 64, 128));
        lblStudentID.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblStudentID.setBounds(162, 240,
102, 32);

    contentPane.add(lblStudentID);

        JLabel lblIssueDate = new
JLabel("Issue Date:");

    lblIssueDate.setForeground(new
Color(0, 64, 128));
        lblIssueDate.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblIssueDate.setBounds(162, 344,
102, 24);

    contentPane.add(lblIssueDate);

        JLabel lblDueDate = new
JLabel("Due Date:");

    lblDueDate.setForeground(new
Color(0, 64, 128));
        lblDueDate.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblDueDate.setBounds(162,
385, 102, 24);

    contentPane.add(lblDueDate);

        JLabel lblStudentName =
new JLabel("Student Name:");

    lblStudentName.setForeground(new
Color(0, 64, 128));
        lblStudentName.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblStudentName.setBounds(162,
294, 133, 32);

    contentPane.add(lblStudentName);
        /*************** LABELS
***************/
```

```java
        /*************** TEXT
FIELDS ****************/
        bookNameField = new
JTextField();

    bookNameField.setEditable(false)
;
        bookNameField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookNameField.setBackground(Syst
emColor.controlHighlight);

    bookNameField.setBounds(305,
186, 204, 31);

    contentPane.add(bookNameField);

    bookNameField.setColumns(10);

        bookIDField = new
JTextField();
        bookIDField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    bookIDField.setBackground(System
Color.info);
        bookIDField.setBounds(305,
135, 204, 31);

    contentPane.add(bookIDField);

    bookIDField.setColumns(10);

        studentIDField = new
JTextField();
        studentIDField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    studentIDField.setBackground(Sys
temColor.info);

    studentIDField.setBounds(305,
237, 204, 31);

    contentPane.add(studentIDField);

    studentIDField.setColumns(10);

        studentNameField = new
JTextField();

    studentNameField.setEditable(fal
se);

    studentNameField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    studentNameField.setColumns(10);

    studentNameField.setBackground(S
ystemColor.controlHighlight);

    studentNameField.setBounds(305,
291, 204, 31);

    contentPane.add(studentNameField
);

        issueDateField = new
JTextField();

    issueDateField.setBackground(Sys
temColor.controlHighlight);
        issueDateField.setFont(new
Font("Tahoma", Font.PLAIN, 14));

    issueDateField.setEditable(false
);

    issueDateField.setBounds(305,
337, 204, 31);

    contentPane.add(issueDateField);

    issueDateField.setColumns(10);

        // Gets current date in
the format dd/MM/yyyy
        SimpleDateFormat date =
new SimpleDateFormat("dd/MM/yyyy");
        Date d = new Date();

    issueDateField.setText(date.form
at(d));

        dueDateField = new
JTextField();
        dueDateField.setFont(new
Font("Tahoma", Font.PLAIN, 14));
```

```java
        dueDateField.setBackground(Syste
mColor.info);

        dueDateField.setBounds(305, 378,
204, 31);

        contentPane.add(dueDateField);

        dueDateField.setColumns(10);
            /*************** TEXT
FIELDS ****************/

            /***************** BUTTONS
*****************/
            // Close button to dispose
window
            JButton btnClose = new
JButton("Close");
            btnClose.setFont(new
Font("Tahoma", Font.BOLD, 15));
            btnClose.setForeground(new
Color(0, 64, 128));

        btnClose.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                    dispose();
                }

        });
            btnClose.setBounds(424,
436, 85, 28);
            contentPane.add(btnClose);

        // Search Button 1 is used
to get Book name when Book ID is
entered
            JButton btnSearch1 = new
JButton("Search");

        btnSearch1.setBackground(SystemC
olor.activeCaption);

        btnSearch1.setForeground(new
Color(0, 64, 128));
            btnSearch1.setFont(new
Font("Tahoma", Font.BOLD, 15));

        btnSearch1.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                    String bookID
= bookIDField.getText();
                    // Checks if
ID has been provided
                    if
(bookID.isEmpty()) {

        JOptionPane.showMessageDialog(nu
ll, "Please Add Book ID", "Error",
JOptionPane.ERROR_MESSAGE);

        return;
                    }
                    // Input
validation
                    if
(!Library.isValidID(bookID)) {

        JOptionPane.showMessageDialog(nu
ll, "Invalid ID Value.", "Error",
JOptionPane.ERROR_MESSAGE);

        return;
                    }

                    // Create a
new Book object to store the found
book
                        Book
foundBook = new Book();

                    // Search for
the book by its ID using the
Library.findBookbyID() method
                        foundBook =
Library.findBookbyID(bookID);

                    // Check if
the book is found
                    if (foundBook
!= null) {
                        // If
the book is found, set the text of
the bookNameField to the book's
title
```

```java
            bookNameField.setText(foundBook.
getBookTitle());
                        } else {
                                // If
the book is not found, display an
error message

    JOptionPane.showMessageDialog(nu
ll, "Book not Found.", "Error",
JOptionPane.ERROR_MESSAGE);
                        }

                }

        });
        btnSearch1.setBounds(546,
135, 85, 32);

    contentPane.add(btnSearch1);

        // Search Button 2 is used
to get Student name when student id
is entered
        JButton btnSearch2 = new
JButton("Search");

    btnSearch2.setBackground(SystemC
olor.activeCaption);

    btnSearch2.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        String
studentID =
studentIDField.getText();
                        // Checks if
student ID has been entered or not
                        if
(studentID.isEmpty()) {

    JOptionPane.showMessageDialog(nu
ll, "Please Add Student ID",
"Error", JOptionPane.ERROR_MESSAGE);
                        } else {
                                //
Validates Student ID
                                if
(!Library.isValidID(studentID)) {

    JOptionPane.showMessageDialog(nu
ll, "Invalid ID Value.", "Error",
JOptionPane.ERROR_MESSAGE);

    return;
                                }

                                //
Creates a User object and searches
for it in the Database
                                User
foundUser = new User();

    foundUser =
Library.findUserbyID(studentID);

                                //
Check if the user is found
                                if
(foundUser != null) {

    // If the user is found, set the
text of the studentNameField to the
user's name

    studentNameField.setText(foundUs
er.getUserName());
                                } else
{

    // If the user is not found,
display an error message

    JOptionPane.showMessageDialog(nu
ll, "User not Found.", "Error",
JOptionPane.ERROR_MESSAGE);
                                }
                        }
                }
        });

    btnSearch2.setForeground(new
Color(0, 64, 128));
        btnSearch2.setFont(new
Font("Tahoma", Font.BOLD, 15));
        btnSearch2.setBounds(546,
236, 85, 32);

    contentPane.add(btnSearch2);
```

```java
        // Issue Book Button
        JButton btnIssue = new
JButton("Issue");
        btnIssue.setForeground(new
Color(0, 64, 128));
        btnIssue.setFont(new
Font("Tahoma", Font.BOLD, 15));

    btnIssue.addActionListener(new
ActionListener() {
                public void
actionPerformed(ActionEvent e) {
                        String
bookName = bookNameField.getText();
                        String bookID
= bookIDField.getText();
                        String
studentName =
studentNameField.getText();
                        String
studentID =
studentIDField.getText();
                        String
issueDate =
issueDateField.getText();
                        String
dueDate = dueDateField.getText();

                        // Checking
if all fields have been filled
                        if
(bookName.isEmpty() ||
issueDate.isEmpty() ||
studentName.isEmpty() ||
dueDate.isEmpty()) {

    JOptionPane.showMessageDialog(nu
ll, "Please Fill in All Fields",
"Error",

    JOptionPane.ERROR_MESSAGE);

    return;
                        } else {
                                //
Checking if correct due date has
been entered
                                try {

    // Checks if due date has any
alphabets

    if (dueDate.matches(".*[a-zA-
Z].*")) {

    JOptionPane.showMessageDialog(nu
ll, "Invalid Due Date", "Error",
JOptionPane.ERROR_MESSAGE);

    return;

    }

                                }
catch (IllegalArgumentException e2)
{

    JOptionPane.showMessageDialog(nu
ll, "", "Error",
JOptionPane.ERROR_MESSAGE);

    return;
                                }
                        }

                        // Checking
if the book is available or not
                        boolean
available = false;
                        Book
foundBook = new Book();
                        foundBook =
Library.findBookbyID(bookIDField.get
Text());

                        if (foundBook
!= null) {
                                String
availability =
foundBook.getAvailability();

    available =
(availability.equals("Available")) ?
true : false;

                        } else {

    JOptionPane.showMessageDialog(nu
ll, "Book Availability not Found.",
"Error",

    JOptionPane.ERROR_MESSAGE);
```

```java
                return;
            }
            // Issue book
if book is avaialable
            if
(available) {
                // If
the book is available, create a new
Book object to issue
                Book
bookToIssue = new Book();

                // Set
the details of the book to be issued

    bookToIssue.setBookTitle(bookName);

    bookToIssue.setBookID(bookID);

    bookToIssue.setUserID(studentID);

    bookToIssue.setIssueDate(issueDate);

    bookToIssue.setDueDate(dueDate);

                //
Attempt to issue the book using the
Library class method
                if
(Library.issueBook(bookToIssue)) {

    // Display a success message

    JOptionPane.showMessageDialog(rootPane, "Book Issued", "",
JOptionPane.INFORMATION_MESSAGE);

    clear();
                } else
{

    // Display an error message

    JOptionPane.showMessageDialog(rootPane, "Failed to issue book", "",
JOptionPane.ERROR_MESSAGE);

                return;
                }
            } else {
                //
Display a message indicating that
the book is not available

    JOptionPane.showMessageDialog(rootPane, "Book Not Available", "Not
Available",

    JOptionPane.ERROR_MESSAGE);
                }

            }

        });
        btnIssue.setBounds(305,
436, 85, 28);
        contentPane.add(btnIssue);
        /***************** BUTTONS
*****************/

        JLabel lblBackGround = new
JLabel("");
        lblBackGround.setIcon(
                    new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\windowbackground.jpg
"));
        lblBackGround.setBounds(0,
0, 786, 493);

    contentPane.add(lblBackGround);
    }

    // Method to clear all fields
    public void clear() {

        bookIDField.setText("");
        bookNameField.setText("");

    studentIDField.setText("");
        dueDateField.setText("");

    studentNameField.setText("");

    }
}
```

12. Return Book Page:

```java
package libraryManagementSystem;

import java.awt.EventQueue;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.Toolkit;
import javax.swing.SwingConstants;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
import java.awt.SystemColor;

public class returnBookWindow
extends JFrame {

    private static final long
serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField bookIDField;
    private JTextField
bookNameField;
    private JTextField
studentIDField;
    private JTextField dueDateField;
    private JTextField
issueDateField;

    /**
     * Launch the application.
     */

    /*************** MAIN METHOD
***************/
    public static void main(String[]
args) {
```

```java
        EventQueue.invokeLater(new
Runnable() {
            public void run() {
                try {

    returnBookWindow frame = new
returnBookWindow();

    frame.setVisible(true);
                } catch
(Exception e) {

    e.printStackTrace();
                }
            }
        });
    }
    /*************** MAIN METHOD
***************/

    /**
     * Create the frame.
     */
    public returnBookWindow() {
        // GUI SETUP

    setIconImage(Toolkit.getDefaultT
oolkit().getImage("C:\\Users\\Dell\\
Desktop\\Library Management
System\\Images\\return.png"));

    setDefaultCloseOperation(JFrame.
DISPOSE_ON_CLOSE);
        setBounds(350, 150, 800,
530);
        contentPane = new
JPanel();
        contentPane.setBorder(new
EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);

    contentPane.setLayout(null);

        /**************** LABELS
***************/
        JLabel lblTitle = new
JLabel("    RETURN BOOK");
```

```java
        lblTitle.setIcon(new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\returnbook1.png"));
        lblTitle.setForeground(new
Color(0, 64, 128));
        lblTitle.setFont(new
Font("Tahoma", Font.BOLD, 18));

    lblTitle.setHorizontalAlignment(
SwingConstants.CENTER);
        lblTitle.setBounds(186,
33, 395, 73);
        contentPane.add(lblTitle);

        JLabel lblStudentName =
new JLabel("Student ID:");

    lblStudentName.setForeground(new
Color(0, 64, 128));
        lblStudentName.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblStudentName.setBounds(186,
241, 128, 21);

    contentPane.add(lblStudentName);

        JLabel lblIssueDate = new
JLabel("Issue Date:");

    lblIssueDate.setForeground(new
Color(0, 64, 128));
        lblIssueDate.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblIssueDate.setBounds(186, 298,
113, 21);

    contentPane.add(lblIssueDate);

        JLabel lblDueDate = new
JLabel("Due Date:");

    lblDueDate.setForeground(new
Color(0, 64, 128));
        lblDueDate.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblDueDate.setBounds(186,
355, 113, 21);

    contentPane.add(lblDueDate);

        JLabel lblBookID = new
JLabel("Book ID:");

    lblBookID.setForeground(new
Color(0, 64, 128));
        lblBookID.setFont(new
Font("Calibri", Font.BOLD, 19));
        lblBookID.setBounds(186,
141, 102, 24);

    contentPane.add(lblBookID);

        JLabel lblBookTitle = new
JLabel("Book Title:");

    lblBookTitle.setForeground(new
Color(0, 64, 128));
        lblBookTitle.setFont(new
Font("Calibri", Font.BOLD, 19));

    lblBookTitle.setBounds(186, 189,
102, 24);

    contentPane.add(lblBookTitle);

        /**************** LABELS
***************/

        /**************** TEXT
FIELDS ***************/
        bookIDField = new
JTextField();

    bookIDField.setBackground(System
Color.info);
        bookIDField.setBounds(319,
137, 230, 31);

    contentPane.add(bookIDField);

    bookIDField.setColumns(10);

        bookNameField = new
JTextField();

    bookNameField.setBackground(Syst
emColor.controlHighlight);
```

```java
    bookNameField.setEditable(false)
;

    bookNameField.setBounds(319,
185, 230, 31);

    contentPane.add(bookNameField);

    bookNameField.setColumns(10);

        studentIDField = new
JTextField();

    studentIDField.setBackground(Sys
temColor.controlHighlight);

    studentIDField.setEditable(false
);

    studentIDField.setBounds(319,
235, 230, 31);

    contentPane.add(studentIDField);

    studentIDField.setColumns(10);

        dueDateField = new
JTextField();

    dueDateField.setBackground(Syste
mColor.controlHighlight);

    dueDateField.setEditable(false);

    dueDateField.setBounds(319, 349,
230, 31);

    contentPane.add(dueDateField);

    dueDateField.setColumns(10);

        issueDateField = new
JTextField();

    issueDateField.setBackground(Sys
temColor.controlHighlight);

    issueDateField.setEditable(false
);

    issueDateField.setBounds(319,
292, 230, 31);

    contentPane.add(issueDateField);

    issueDateField.setColumns(10);
        /*************** TEXT
FIELDS ***************/

        /***************** BUTTONS
****************/
        JButton btnReturn = new
JButton("Return");

    btnReturn.setForeground(new
Color(0, 64, 128));
        btnReturn.setFont(new
Font("Tahoma", Font.BOLD, 15));

    btnReturn.addActionListener(new
ActionListener() {
            public void
actionPerformed(ActionEvent e) {
                // Get the book ID
entered by the user
                String bookID =
bookIDField.getText();

                // Check if the
book ID is empty
                if
(bookID.isEmpty()) {

JOptionPane.showMessageDialog(rootPa
ne, "Please Enter Book ID", "Error",
JOptionPane.ERROR_MESSAGE);
                }

                // Create a new
Book object to represent the book to
be returned
                Book bookToReturn
= new Book();

bookToReturn.setBookID(bookID);

                // Attempt to
return the book using the Library
class method
```

```java
                if
(Library.returnBook(bookToReturn)) {
                        // Display a
success message

JOptionPane.showMessageDialog(rootPa
ne, "Book Returned", "",
JOptionPane.INFORMATION_MESSAGE);
                        clear();
                    } else {
                        // Display an
error message

JOptionPane.showMessageDialog(rootPa
ne, "Failed to return book", "",
JOptionPane.ERROR_MESSAGE);
                        return;
                    }
                }
            });

            btnReturn.setBounds(319,
416, 93, 31);

    contentPane.add(btnReturn);

            // Dispose Window on Close
button
            JButton btnClose = new
JButton("Close");
            btnClose.setFont(new
Font("Tahoma", Font.BOLD, 15));
            btnClose.setForeground(new
Color(0, 64, 128));

    btnClose.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent e) {
                        dispose();
                    }
            });
            btnClose.setBounds(464,
416, 85, 31);
            contentPane.add(btnClose);

            // Search Button
            JButton btnSearch = new
JButton("Search");

    btnSearch.setBackground(SystemCo
lor.activeCaption);
            btnSearch.setFont(new
Font("Tahoma", Font.BOLD, 15));

    btnSearch.setForeground(new
Color(0, 64, 128));

    btnSearch.addActionListener(new
ActionListener() {
                    public void
actionPerformed(ActionEvent e) {
                        // Get the book
ID entered by the user
                        String bookID =
bookIDField.getText();

                        // Check if the
book ID is empty
                        if
(bookID.isEmpty()) {

JOptionPane.showMessageDialog(null,
"Please Add Book ID", "Error",
JOptionPane.ERROR_MESSAGE);
                            return;
                        }

                        // Validate the
book ID (ensure it is numeric)
                        if
(!Library.isValidID(bookID)) {

JOptionPane.showMessageDialog(null,
"Invalid ID Value.", "Error",
JOptionPane.ERROR_MESSAGE);
                            clear();
                            return;
                        }

                        Boolean
available = false;

                        // Create a new
Book object to store the found book
details
                        Book foundBook
= new Book();

foundBook.setBookID(bookID);
```

```java
                    // Retrieve
book details from the Library using
the book ID
                    foundBook =
Library.findBookbyID(bookIDField.get
Text());

                    // Check if the
book was found
                    if (foundBook
!= null) {
                            // If
found, populate the fields with the
book details

bookNameField.setText(foundBook.getB
ookTitle());

studentIDField.setText(foundBook.get
UserID());

issueDateField.setText(foundBook.get
IssueDate());

dueDateField.setText(foundBook.getDu
eDate());

                            // Check if
the book is available for issuance
                            available =
foundBook.getAvailability().equals("
Available");
                    } else {
                            // If the
book was not found, display an error
message

JOptionPane.showMessageDialog(null,
"Book not Found.", "Error",
JOptionPane.ERROR_MESSAGE);
                            clear();
                            return;
                    }

                    // If the book
is available for issuance
                    if (available)
{

JOptionPane.showMessageDialog(rootPa
ne, "Book was not Issued", "",
JOptionPane.ERROR_MESSAGE);
                            clear();
                            return;
                    }
            }

        });
        btnSearch.setBounds(579,
135, 93, 31);

    contentPane.add(btnSearch);
        /**************** BUTTONS
*****************/

        /****************
BACKGROUND **************/
        JLabel background = new
JLabel("");
        background.setIcon(
                    new
ImageIcon("C:\\Users\\Dell\\Desktop\
\Library Management
System\\Images\\windowbackground.jpg
"));
        background.setBounds(0, 0,
786, 493);

    contentPane.add(background);
        /****************
BACKGROUND **************/
    }

    // Clear all Fields
    public void clear() {
        bookIDField.setText("");

    studentIDField.setText("");
        bookNameField.setText("");
        dueDateField.setText("");

    issueDateField.setText("");
    }
}
```

CONCLUSION

The Library Management System, developed in Java with Eclipse and MySQL integration, offers an efficient solution for librarians. With its user-friendly GUI and essential functionalities like adding users and books, searching, issuing, and returning books, it streamlines library operations. Testing ensures reliability, and error handling notifies users of any issues.

1. References:

a. Utilize resources like Geek for Geeks for comprehensive study materials.

b. Explore YouTube tutorials for step-by-step guidance on setting up libraries and learning SQL basics.

c. Leverage GitHub for a wide range of ideas and solutions to enhance and expand the functionality of the system.