V V COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

LIBRARY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

C. MATHEW KURUSE (953421104025)

A. N. MOHAMED RIYAZ (953421104026)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

V V COLLEGE OF ENGINEERING, TISAIYANVILAI

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report **"LIBRARY MANAGEMENT SYSTEM"** is a bonafide work of **"C. MATHEW KURUSE (953421104025), A.N. MOHAMED RIYAZ (953421104026) "** who carried out final year project work under my supervision.

SIGNATURE

Dr.I.Muthulakshmi.,M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

Computer Science and Engineering,
V V College of Engineering,
Tisayainvilai-627 657.

SIGNATURE

Ms. M.Jancyrani Malli.,M.Tech.,

SUPERVISOR

Computer Science and Engineering,
V V College of Engineering,
Tisaiyanvilai-627 657.

Submitted for the viva-voce held on ……..

INTERNAL EXAMINER

EXTERNAL EXAMINAR

ACKNOWLEDGEMENT

First and foremost, we express our gratitude to God Almighty who has showered blessings on us by giving us the strength and determination to complete this project successfully.

Our sincere thanks to the **Management** for giving us good firmware facilities to complete the project. We thank our department faculty members for their support to complete this project.

We sincerely thank our **Principal Dr.P.Vanitha.,M.E,Ph.D.,** for her valuable guidance and support.

We would like to express our heartfelt gratitude to our Head of the Department **Dr.I.Muthulakshmi.,M.E.,Ph.D.,** for being a constant source of inspiration to us.

We are indebted to our project co-ordinator **Ms. M.Jancyrani Malli.,M.Tech.,** for providing moral support and motivation.

Our special thanks to our families and friends who have stood by us during the course of our project.

ABSTRACT

The library management system project aims to streamline the process of managing books and student records within a library setting. By providing a user-friendly graphical interface, users, including librarians and administrators, can efficiently perform various tasks such as issuing books to students and handling book returns. The system allows for easy searching and retrieval of book and student information using unique identifiers like book IDs and student IDs. Through integration with a database using JDBC, the system ensures data integrity and persistence, enabling seamless storage and retrieval of information. Key features include the ability to input, update, and delete book and student records, as well as issuing and returning books with automated date tracking. With its intuitive design and robust functionality, the library management system project aims to enhance the overall efficiency and effectiveness of library operations, ultimately benefiting both library staff and patrons. Additionally, the library management system project incorporates features to manage overdue books and fines, providing automated notifications to remind users of impending due dates and overdue items. This helps promote timely returns and ensures accountability among library patrons.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The library management system (LMS) project represents a comprehensive solution aimed at optimizing the management of books and user records within a library environment. This section serves as an introduction to the project, providing an overview of its aims, objectives, background, and the operational environment in which it functions.

1.1 PROJECT AIMS AND OBJECTIVES

The primary goal of the LMS project is to streamline and modernize the processes involved in managing library resources and user information. Key objectives include:

Streamlining book management processes: The LMS aims to simplify tasks such as book cataloging, tracking, and inventory management, ultimately improving the overall efficiency of library operations.

Enhancing user experience: By providing a user-friendly interface and intuitive features, the LMS seeks to enhance the experience of both library staff and patrons, making it easier to search for and access library resources.

Improving operational efficiency: Through automation and integration with databases and other systems, the LMS aims to reduce manual workload, minimize errors, and optimize resource allocation within the library.

Facilitating data-driven decision-making: By centralizing and organizing library data, the LMS enables librarians and administrators to analyze trends, track usage patterns, and make informed decisions regarding collection development and resource allocation.

1.2 BACKGROUND OF PROJECT

The development of the LMS project was prompted by several factors, including:

Existing challenges in manual book management systems: Traditional methods of cataloging and managing library resources are often time-consuming and prone to errors. The need for a more efficient and accurate solution led to the exploration of automated library management systems.

Technological advancements: The rapid advancements in information technology, particularly in areas such as database management, user interface design, and networking, presented opportunities to develop sophisticated library management solutions that could address the evolving needs of modern libraries.

Modernization in library operations: As libraries increasingly embrace digital resources and online services, there is a growing need for robust, flexible, and scalable management systems that can accommodate these changes while maintaining the core functions of traditional library services.

1.3 OPERATION ENVIRONMENT

The operational environment of the LMS encompasses various factors that influence its functionality and usability, including:

Physical setting of the library: This includes considerations such as the size, layout, and infrastructure of the library facility, as well as the availability of resources such as computers, scanners, and printers.

Target user base: The LMS caters to a diverse user base, including librarians, library staff, students, faculty, and other library patrons. As such, the system must be designed to accommodate the needs and preferences of different user groups.

Technological infrastructure: The LMS relies on a robust technological infrastructure, including hardware such as servers, computers, and networking equipment, as well as software components such as databases, operating systems, and application servers.

External factors: External factors such as regulatory requirements, institutional policies, and industry standards may impact the design, implementation, and operation of the LMS. Additionally, considerations such as data security, privacy, and accessibility must be taken into account to ensure compliance with relevant regulations and best practices.

CHAPTER 2

SYSTEM ANALYSIS

The system analysis phase constitutes a comprehensive investigation into the intricacies of the library management system (LMS), encompassing its requirements, functionalities, and limitations. This section scrutinizes diverse facets of the system, including the delineation of software requirements, a comparative analysis between the prevailing system and the proposed LMS, and an elucidation of the software tools harnessed throughout the development trajectory.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

The software requirement specification (SRS) serves as the cornerstone of the LMS, delineating both its functional and non-functional prerequisites, thereby furnishing a blueprint for the system's conception and maturation. The cardinal constituents of the SRS entail:

Functional Requirements: These delineate the gamut of the system's capabilities and functionalities, encompassing pivotal aspects such as user authentication, book cataloging, search and retrieval mechanisms, user management functionalities, reporting functionalities, and administrative tasks, ensuring a seamless orchestration of library operations.

Non-functional Requirements: These edify the quality attributes underpinning the system, embracing elements such as performance, reliability, security, usability, scalability, and maintainability. For instance,

the system is mandated to adeptly navigate a surfeit of concurrent users sans any performance compromises.

User Requirements: These epitomize the exigencies and anticipations of disparate user cohorts, spanning librarians, administrators, and patrons alike. User requisites may span a spectrum of features including an ergonomic user interface, personalized user profiles, and provisions for accessibility catering to users with disabilities.

2.2 EXISTING VS PROPOSED

This section compares the existing library management system with the proposed one, highlighting improvements:

Ease of Use: The proposed system offers a simpler interface and smoother workflows, making tasks like book retrieval easier.

Functionality: The new system introduces advanced features like better search options, integration with external databases, and automated notifications for overdue books.

Performance: It shows better response times, system uptime, and scalability, accommodating more users and data effectively.

Security: The new system includes stronger security measures like access control and encryption to protect data better.

2.3 SOFTWARE TOOL USED

The development of the LMS relies on a variety of software tools chosen to support its design, implementation, testing, and deployment. These tools include:

Integrated Development Environments (IDEs): Tools like Eclipse, NetBeans, and IntelliJ IDEA provide comprehensive support for coding, debugging, and testing, offering features like auto-completion and syntax highlighting.

Database Management Systems (DBMS): Platforms such as MySQL, PostgreSQL, and Oracle are essential for storing and managing the library's data, including book records, user information, and system configurations.

Version Control Systems (VCS): Systems like Git and Subversion facilitate collaborative development by enabling multiple developers to work on the same codebase simultaneously, supporting versioning, branching, and merging.

The development of the library management system (LMS) heavily relies on Java, Java Swing for the frontend, and SQL for the backend, along with additional tools to support the development process.

Java: Java serves as the primary programming language for developing the library management system. Java is chosen for its platform independence, robustness, and extensive standard libraries. It allows for the creation of modular, scalable, and maintainable code, making it well-suited

for large-scale software projects like the LMS. Additionally, Java's object-oriented nature facilitates code reusability and modularity, contributing to the system's overall flexibility and extensibility.

Java Swing: Java Swing is a set of graphical user interface (GUI) components and libraries for building desktop applications in Java. In the context of the LMS, Java Swing is utilized to create the frontend interface, including windows, dialogs, buttons, text fields, and other GUI elements. Java Swing provides a rich set of tools and layouts for designing intuitive and interactive user interfaces, enabling users to navigate, interact with, and manage library resources seamlessly. With Java Swing, developers can create platform-independent GUI applications that run smoothly on various operating systems without compromising performance or functionality.

SQL (Structured Query Language): SQL is used as the backend database language for storing, retrieving, and manipulating data in the library management system. SQL databases such as MySQL, PostgreSQL, or SQLite are commonly employed to manage the system's relational database schema, which includes tables for storing information about books, students, transactions, and other relevant entities. SQL enables efficient data querying, indexing, and transaction management, ensuring the integrity, consistency, and security of the library's data. Through SQL, developers can implement complex database operations such as CRUD (Create, Read, Update, Delete) operations, data aggregation, and joins, providing a solid foundation for the LMS's data management capabilities.

CHAPTER 3
SYSTEM DESIGN

3.1 TABLE DESIGN

The table design for the library management system (LMS) involves creating structured database tables to store information about accounts, books, students, issued books, and returned books. Each table is designed to efficiently organize and manage specific types of data, ensuring data integrity and facilitating seamless retrieval and manipulation of information. Below is the table design for the LMS:

1. Account Table:

- **`id`:** Primary key auto-incremented integer representing the account ID.
- **`username`:** VARCHAR(50) field storing the username of the account holder.
- **`name`:** VARCHAR(50) field storing the name of the account holder.
- **`password`:** VARCHAR(100) field storing the encrypted password of the account.
- **`sec_q`:** VARCHAR(100) field storing the security question for the account.
- **`sec_ans`:** VARCHAR(100) field storing the encrypted security answer for the account.

2. Book Table:

- **`book_id`:** Primary key auto-incremented integer representing the book ID.

- **`name`:** VARCHAR(100) field storing the name of the book.

- **`isbn`:** VARCHAR(50) field storing the ISBN (International Standard Book Number) of the book.

- **`publisher`:** VARCHAR(100) field storing the publisher of the book.

- **`edition`:** VARCHAR(50) field storing the edition of the book (if applicable).

- **`price`:** DECIMAL(10, 2) field storing the price of the book.

- **`pages`:** INT field storing the number of pages in the book.

3. Student Table:

- **`student_id`:** Primary key auto-incremented integer representing the student ID.

- **`name`:** VARCHAR(100) field storing the name of the student.

- **`father`:** VARCHAR(100) field storing the name of the student's father.

- **`course`:** VARCHAR(50) field storing the course enrolled by the student.

- **`branch`:** VARCHAR(50) field storing the branch of study of the student.

- **`year`:** VARCHAR(20) field storing the academic year of the student.

- **`semester`:** VARCHAR(20) field storing the semester of the student.

4. IssueBook Table:

- **`issue_id`:** Primary key auto-incremented integer representing the issue ID.

- **`book_id`:** Foreign key referencing the `book_id` in the Book table.

- **`student_id`:** Foreign key referencing the `student_id` in the Student table.
- **`bname`:** VARCHAR(255) field storing the name of the issued book.
- **`sname`:** VARCHAR(255) field storing the name of the student to whom the book is issued.
- **`course`:** VARCHAR(50) field storing the course of the student.
- **`branch`:** VARCHAR(50) field storing the branch of the student.
- **`dateOfIssue`:** DATE field storing the date when the book is issued.

5. ReturnBook Table:

- **`book_id`:** VARCHAR(20) field storing the ID of the returned book.
- **`student_id`:** VARCHAR(20) field storing the ID of the student returning the book.
- **`bname`:** VARCHAR(100) field storing the name of the returned book.
- **`sname`:** VARCHAR(100) field storing the name of the student returning the book.
- **`course`:** VARCHAR(50) field storing the course of the student.
- **`branch`:** VARCHAR(50) field storing the branch of the student.
- **`dateOfIssue`:** DATE field storing the date when the book was issued.
- **`dateOfReturn`:** DATE field storing the date when the book was returned.

These tables form the backbone of the LMS database, providing a structured framework for storing and managing crucial information related to accounts, books, students, and transactions within the library system.

3.2 UML Diagrams

In the system design phase of the library management system (LMS) project, Unified Modeling Language (UML) diagrams are instrumental in visualizing the architecture, structure, and interactions within the system. These diagrams serve as blueprints for developers and stakeholders, providing a clear understanding of the system's design and functionality. The following UML diagrams are created for the LMS:

3.2.1. Class Diagram:

The class diagram illustrates the static structure of the system by representing classes, attributes, methods, and their relationships. It depicts the entities involved in the LMS, such as books, students, accounts, and transactions, along with their attributes and associations.

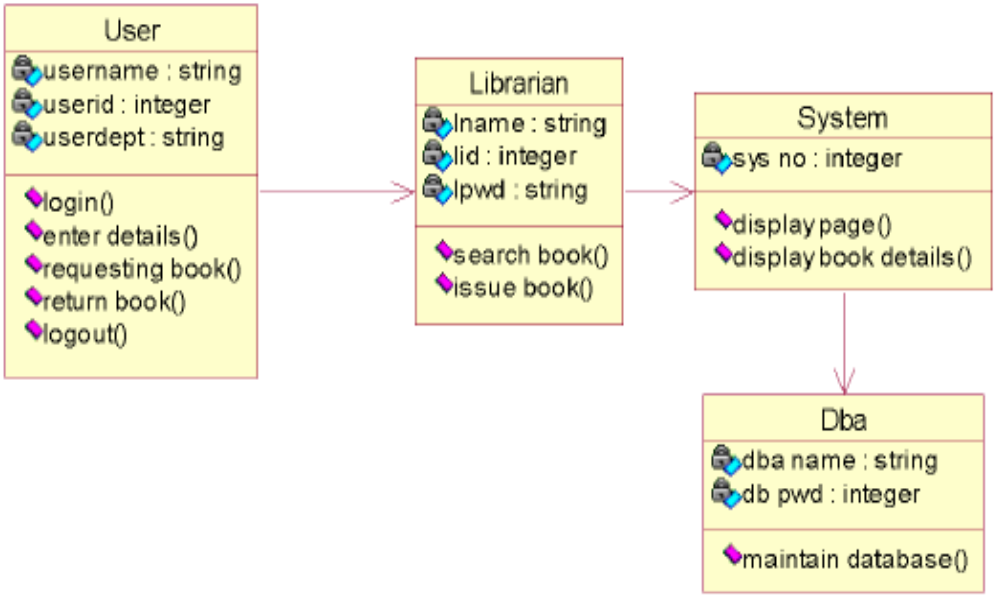


Figure 3.2.1 Class Diagram

3.2.2. Use Case Diagram:

The use case diagram identifies the system's functionalities from the user's perspective. It showcases various actors, such as librarians, administrators, and students, interacting with the system to perform tasks like book search, checkout, return, user management, and reporting. Each use case represents a specific functionality of the system.

Figure 3.2.2 Use Case Diagram

3.2.3. Sequence Diagram:

The sequence diagram visualizes the interactions between objects or components in a sequential manner, depicting the flow of messages exchanged during runtime. It illustrates how different system components collaborate to fulfill a specific use case scenario, such as issuing a book to a student or returning a book to the library.

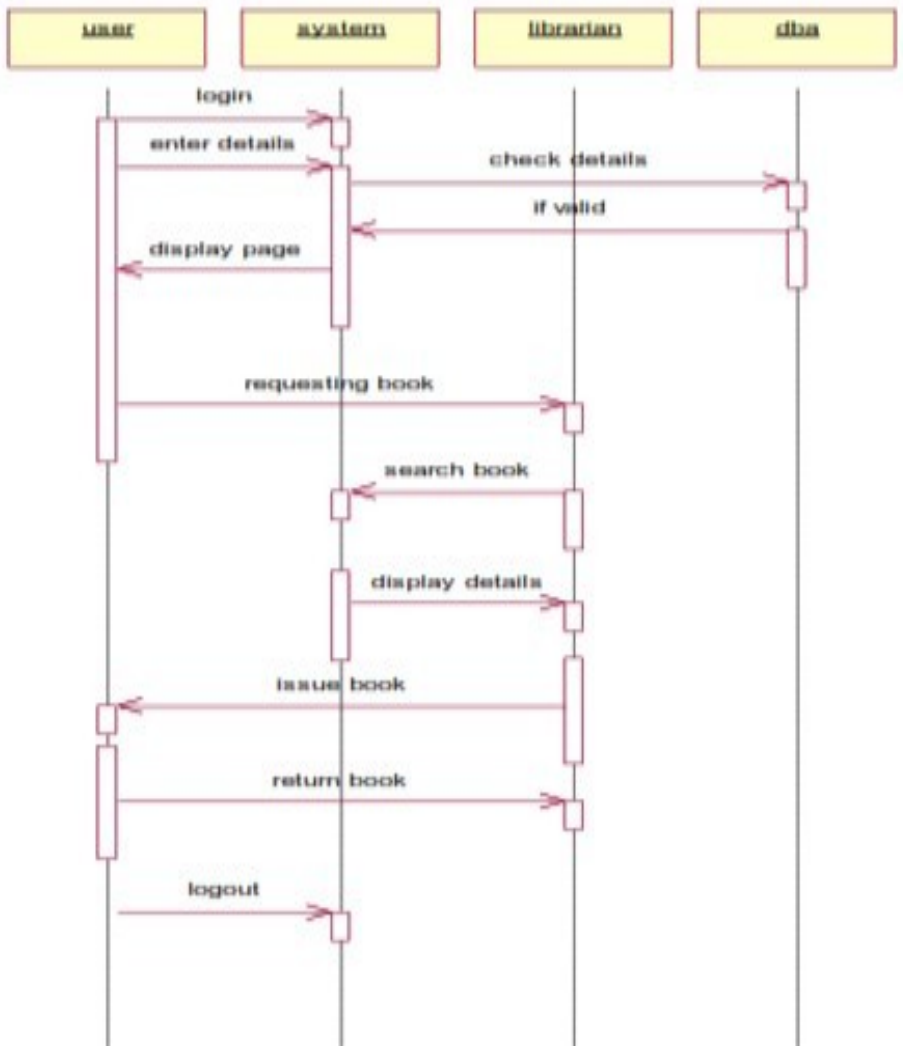


Figure 3.2.3 Sequence Diagram

3.2.4. Activity Diagram:

The activity diagram represents the workflow or business processes within the system, showcasing the sequence of activities, decision points, and transitions. It helps in understanding the control flow and logic of various operations performed by the system, such as user authentication, book cataloging, and transaction processing.

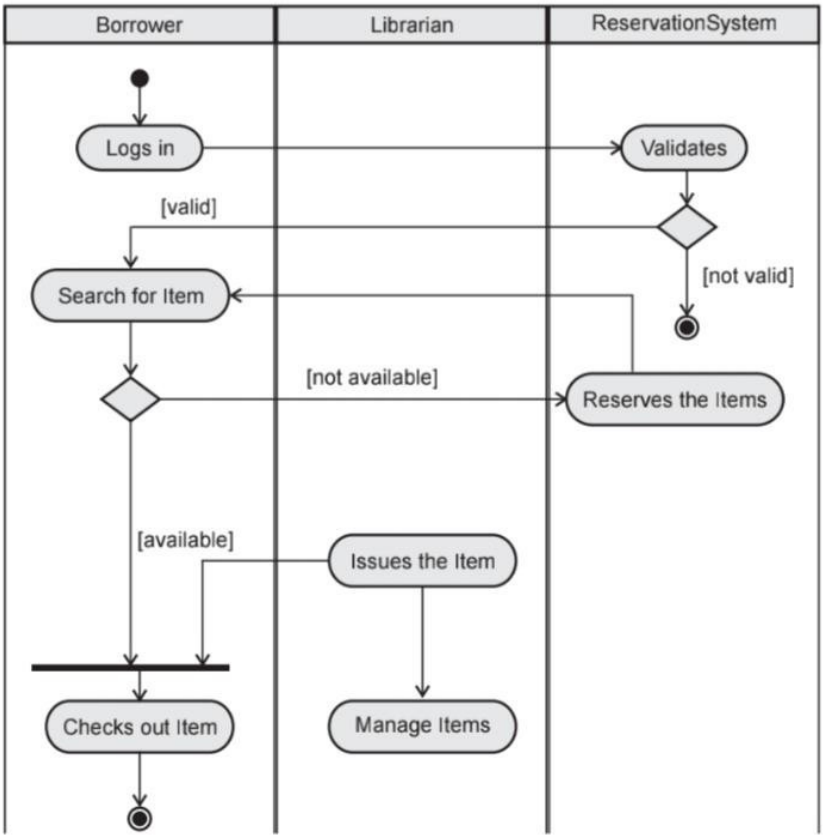


Figure 3.2.4 Activity Diagram

3.2.5. State Diagram:

The state diagram models the lifecycle of objects or entities within the system, depicting their different states and transitions triggered by events. It illustrates the behavior of entities like books, students, and transactions as they progress through various states, such as available, issued, overdue, and returned.

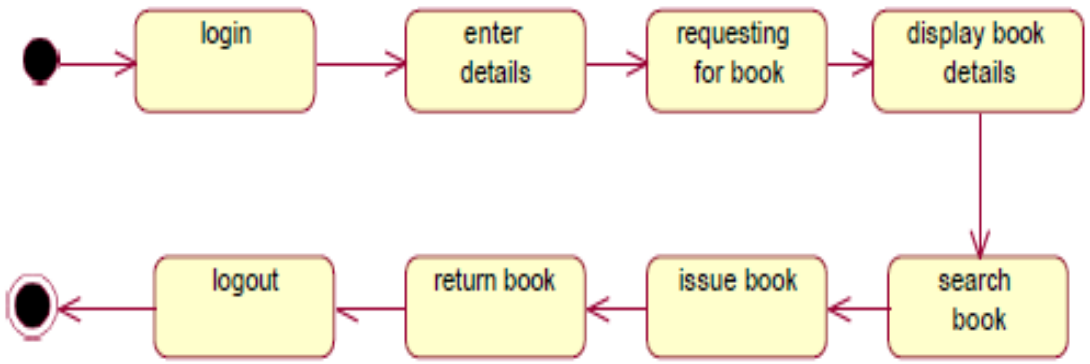


Figure 3.2.5 State Diagram

These UML diagrams collectively provide a comprehensive overview of the LMS design, enabling stakeholders to visualize system components, functionalities, interactions, and behaviors in a structured manner. They serve as invaluable tools for communication, analysis, and validation throughout the development lifecycle of the LMS project.

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 MODULE DESCRIPTION

The library management system (LMS) consists of several interconnected modules, each tailored to specific functions essential for efficient library operations. Here are the key modules within the LMS:

1. Admin Dashboard Module:

This module serves as the central control hub for administrators to oversee and manage various aspects of the LMS. It provides real-time visibility into system activities, user interactions, and performance metrics. Administrators can configure system settings, manage user permissions, and address issues from a single interface.

2. Book Management Module:

Responsible for cataloging, tracking, and organizing library resources, this module allows administrators to add, update, and delete book records. It includes functionalities for assigning unique identifiers (e.g., ISBN numbers), managing book metadata (e.g., title, author, publisher), and tracking book availability.

3. Search and Retrieval Module:

This module enables users to search for and retrieve books from the library's collection based on various criteria, such as title, author, or subject.

It provides a user-friendly interface for browsing search results, viewing detailed book information, and checking availability status.

4. Issue and Return Module:

Facilitating the borrowing and returning of books, this module allows administrators to issue books to users, record due dates, and track loan transactions. It also manages the processing of book returns, calculates overdue fines, and updates loan statuses.

5. Reporting and Analytics Module:

Equipped with tools for generating reports and analyzing library usage data, this module offers pre-defined report templates for metrics like book circulation, patron demographics, and overdue books. Advanced analytics features empower administrators to gain insights into usage patterns and make informed decisions about resource allocation.

6. Security and Access Control Module:

Enforcing security policies and access controls within the LMS, this module includes features such as role-based access control (RBAC), encryption mechanisms, and audit logging functionalities. It safeguards sensitive data and ensures compliance with regulatory requirements.

Each module serves a critical role in supporting library operations and delivering a seamless user experience. By collaborating and integrating functionalities, these modules collectively optimize resource management, streamline workflows, and enhance the overall efficiency of library services.

CHAPTER 5
SYSTEM TESTING

5.1 UNIT TESTING:

Unit testing is a crucial phase in the software development life cycle, focusing on testing individual units or components of the library management system (LMS) to ensure their functionality, reliability, and correctness. Each unit is tested in isolation, verifying that it performs as expected and meets the specified requirements. The unit testing process typically involves the following steps:

1. **Test Case Design:** Create tests covering different scenarios and functions, including typical and extreme cases, to ensure thorough testing.

2. **Test Environment Setup:** Prepare a test environment with tools like mock objects to simulate system behavior and dependencies.

3. **Test Execution:** Run the tests and compare actual outcomes with expected results, noting any discrepancies.

4. **Test Result Analysis:** Analyze test results to find reasons for failures and improve unit performance.

5. **Regression Testing:** Re-run tests to ensure changes don't introduce new defects, maintaining system integrity.

6. **Code Coverage Analysis:** Assess test effectiveness by measuring how much of the code is covered, aiming for high coverage to catch more issues.

7. **Documentation:** Keep detailed records of tests, results, and observations for future reference and maintenance.

Effective unit testing detects and fixes issues early, improving software quality and reducing maintenance costs. It also enhances code modularity and reusability, contributing to project success.

5.2 INTEGRATION TESTING:

Integration testing ensures that different parts of the library management system work well together. Here's how it works:

1.Strategy Selection: Decide how to combine modules based on system design.

2. Environment Setup: Create a test environment similar to the real one.

3.Test Planning: Design tests covering how modules interact and handle errors.

4.Component Integration: Combine modules step by step, checking if they fit together.

5.Data Exchange: Make sure data moves accurately between modules.

6.Interface Check: Test if interfaces work correctly.

7.Error Handling: See if the system handles mistakes well.

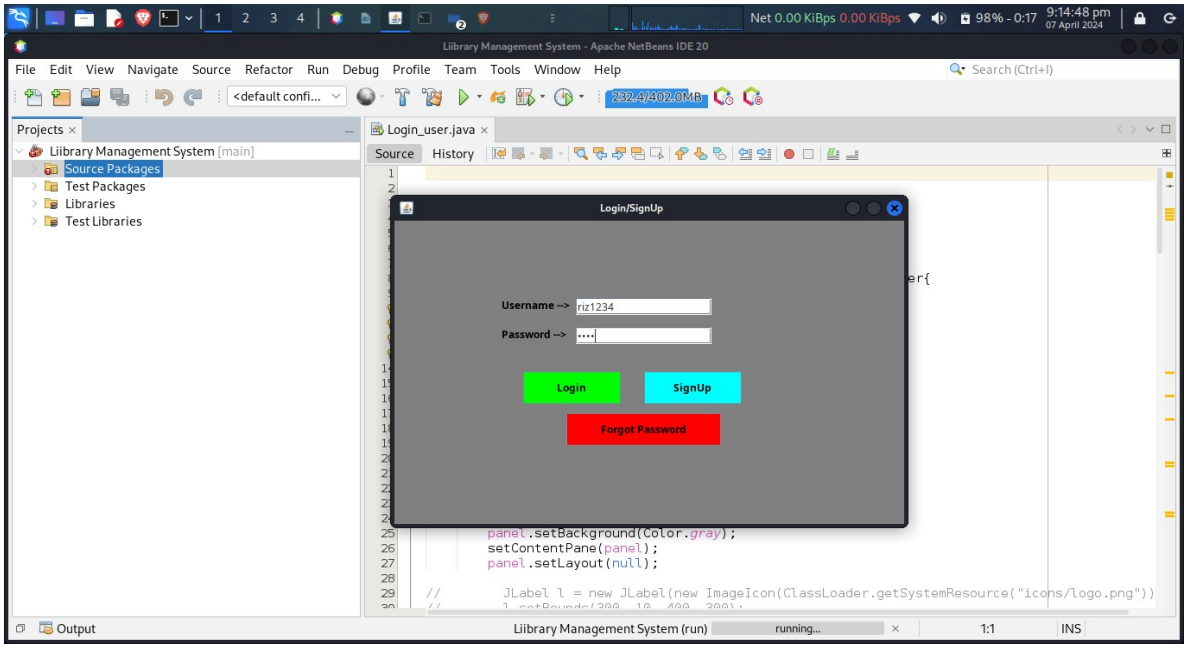
8.Test Run: Execute tests, record results, and fix any problems.

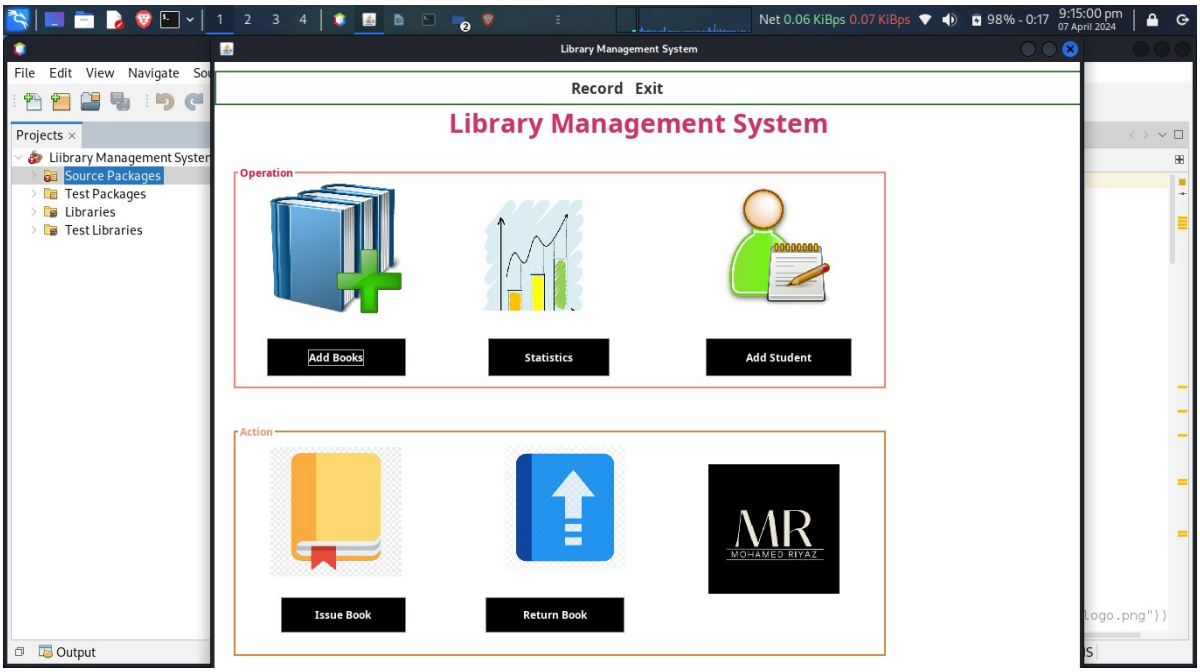
9.Regression Testing: Recheck everything after changes to ensure nothing breaks.

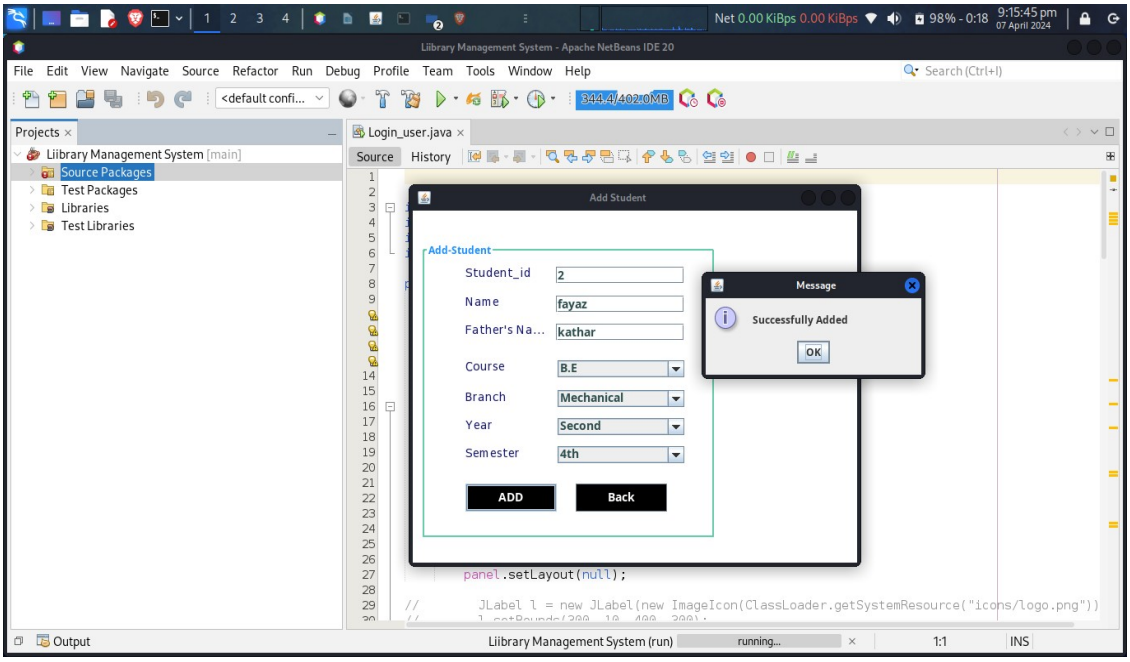
Integration testing guarantees the system works smoothly, meeting performance and reliability standards for success.

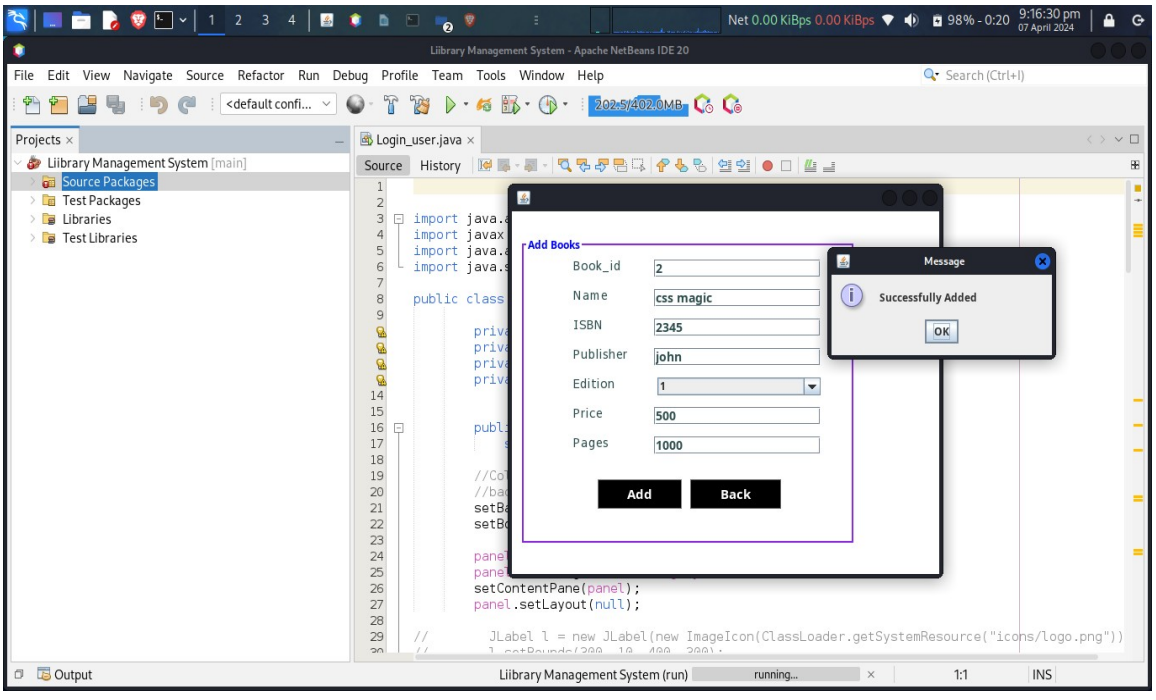
CHAPTER 6

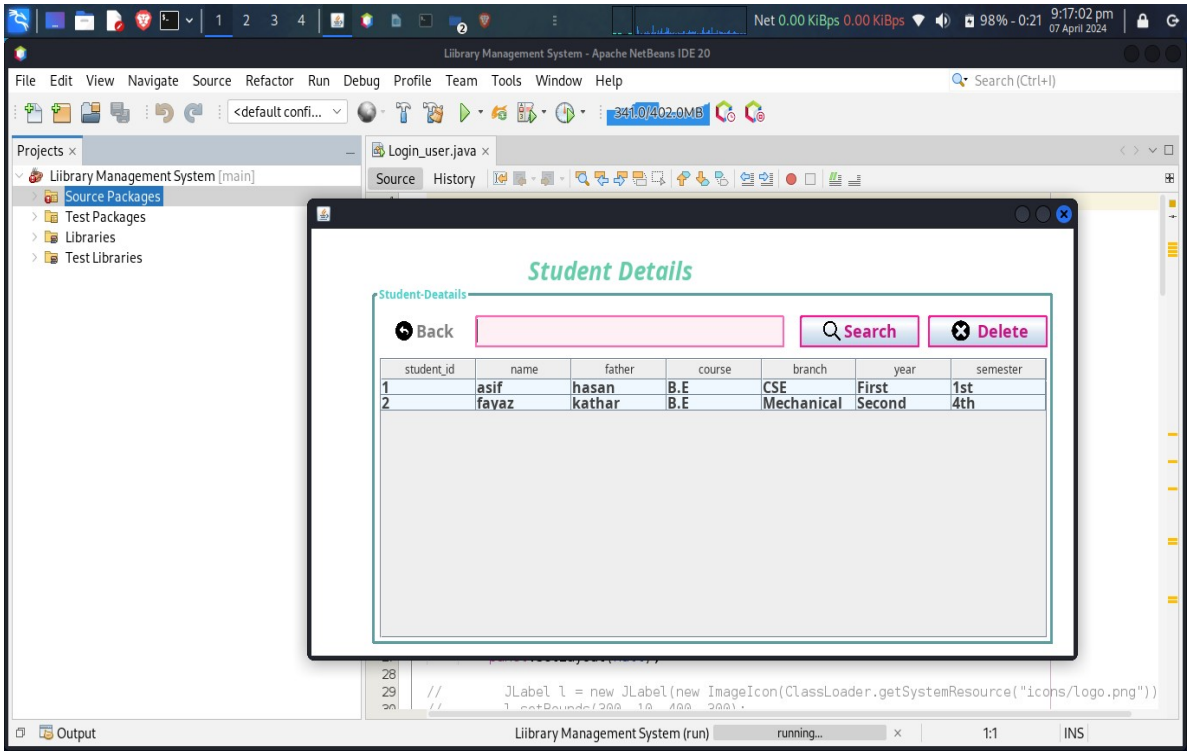
RESULTS

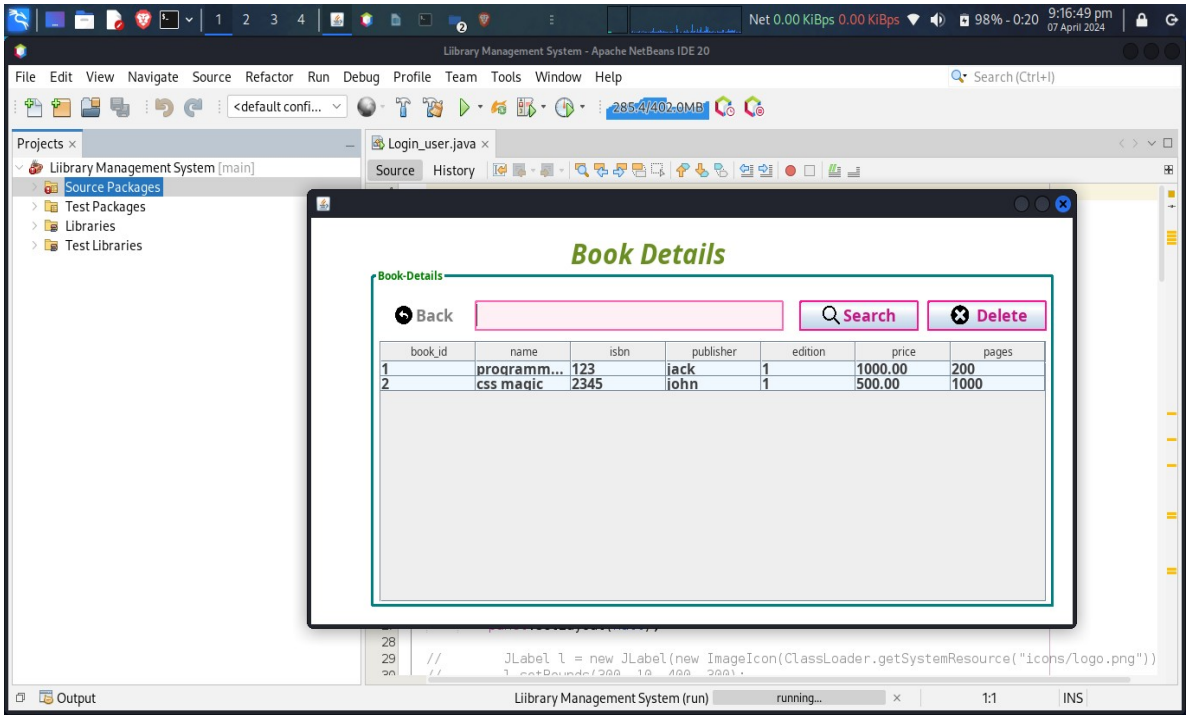


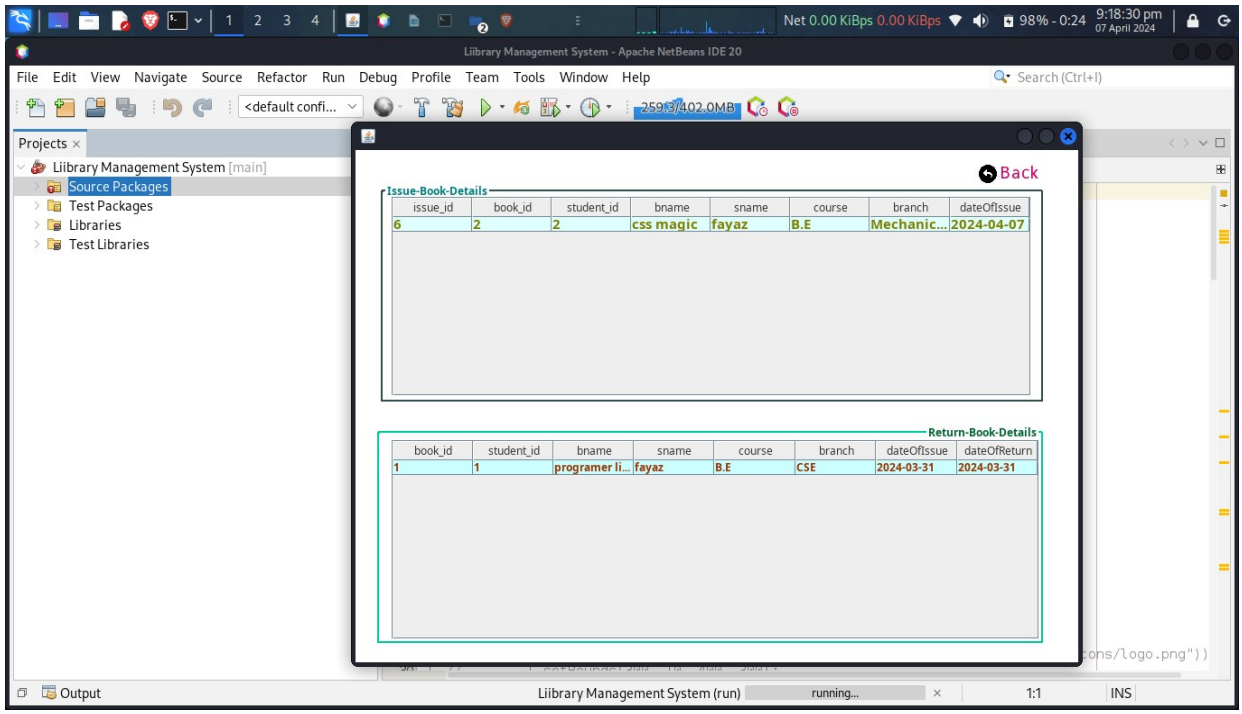



Liibrary Management System - Apache NetBeans IDE 20
Add Student
Add-Student
Student_id
2
Name
fayaz
Father's Na...
kathar
Course
B.E
Branch
Mechanical
Year
Second
Semester
4th
ADD
Back
Message
Successfully Added
OK


Liibrary Management System - Apache NetBeans IDE 20
Add Books
Book_id
2
Name
css magic
ISBN
2345
Publisher
john
Edition
1
Price
500
Pages
1000
Add
Back
Message
Successfully Added
OK


Liibrary Management System - Apache NetBeans IDE 20
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Liibrary Management System [main]
Source Packages
Test Packages
Libraries
Test Libraries
Login_user.java
Source History
Student Details
Student-Deatails
Back
Search
Delete
student_id name father course branch year semester
1 asif hasan B.E CSE First 1st
2 fayaz kathar B.E Mechanical Second 4th
Output
Liibrary Management System (run)
running...
1:1
INS


Liibrary Management System - Apache NetBeans IDE 20
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Liibrary Management System [main]
Source Packages
Test Packages
Libraries
Test Libraries
Login_user.java
Source History
Book Details
Book-Details
Back
Search
Delete
book_id name isbn publisher edition price pages
1 programm... 123 jack 1 1000.00 200
2 css magic 2345 john 1 500.00 1000
Output
Liibrary Management System (run)
running...
1:1
INS


Liibrary Management System - Apache NetBeans IDE 20
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Liibrary Management System [main]
Source Packages
Test Packages
Libraries
Test Libraries
Back
Issue-Book-Details
issue_id book_id student_id bname sname course branch dateOfIssue
6 2 2 css magic fayaz B.E Mechanic... 2024-04-07
Return-Book-Details
book_id student_id bname sname course branch dateOfIssue dateOfReturn
1 1 programer li... fayaz B.E CSE 2024-03-31 2024-03-31
Output
Liibrary Management System (run)
running...
1:1
INS

CHAPTER 7

CONCLUSION & FUTURE SCOPE

The development of the library management system (LMS) marks a significant step forward in enhancing library operations and user experience. By streamlining processes and ensuring data integrity, the project has overcome many challenges faced by manual systems. Looking ahead, there are opportunities for further improvement:

1. Reporting and Analytics: Implement advanced reporting to understand library usage and improve resource allocation.

2. Mobile App Development: Create a mobile app for users to access library services conveniently from their smartphones.

3. Integration with Digital Libraries: Connect the LMS with digital resources to expand access to e-books and multimedia content.

4. User Feedback and Improvement: Continuously gather feedback to enhance the LMS, ensuring it remains user-friendly and effective.

5. Accessibility: Ensure the LMS meets accessibility standards, making it usable for all users, including those with disabilities.

In conclusion, the LMS project lays the foundation for modernizing libraries and enhancing user services, with ongoing innovation driving its evolution as a vital resource for knowledge sharing and learning.

CHAPTER 8

CODING

8.1 Code For SQL Connection :

```
import java.sql.*;
public class conn{
    Connection c;
    Statement s;
    public conn(){
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
             c=DriverManager.getConnection("jdbc:mysql:///project2","root","12
34");
            s =c.createStatement();
        }catch(Exception e){
            System.out.println(e);
        }
    }
    public static void main(String args[])
    {
        new conn();
        System.out.println("Database Connected");
    }
}
```

8.2 Code For Login_user:

```
import java.awt.
import javax.swing.*;
import java.awt.event.*;
import java.sql.*;

public class Login_user extends JFrame implements ActionListener{

    private JPanel panel;
    private JTextField textField;
    private JPasswordField passwordField;
   private JButton b1,b2,b3;


    public Login_user() {
      super("Login/SignUp");

   //Color class --> Swing
   //background --> Jframe
    setBackground(Color.ORANGE);
   setBounds(600, 300, 600, 400);

   panel = new JPanel();
    panel.setBackground(Color.gray);
    setContentPane(panel);
    panel.setLayout(null);
```

```
  JLabel l1 = new JLabel("Username --> ");
  l1.setBounds(124, 89, 95, 24);
l1.setForeground(Color.black);
  panel.add(l1);
  JLabel l2 = new JLabel("Password --> ");
  l2.setBounds(124, 124, 95, 24);
l2.setForeground(Color.black);
  panel.add(l2);
  textField = new JTextField();
  textField.setBounds(210, 93, 157, 20);
  panel.add(textField);
  passwordField = new JPasswordField();
  passwordField.setBounds(210, 128, 157, 20);
  panel.add(passwordField);
  JLabel l3 = new JLabel("");
  l3.setBounds(377, 79, 46, 34);
  panel.add(l3);
  JLabel l4 = new JLabel("");
  l4.setBounds(377, 124, 46, 34);
  panel.add(l3);
  b1 = new JButton("Login");
  b1.addActionListener(this);
  b1.setForeground(Color.black);
  b1.setBackground(Color.green);
  b1.setBounds(149, 181, 113, 39);
  panel.add(b1);
```

```
  b2 = new JButton("SignUp");
    b2.addActionListener(this);
    b2.setForeground(Color.black);
    b2.setBackground(Color.CYAN);
    b2.setBounds(289, 181, 113, 39);
    panel.add(b2);
    b3 = new JButton("Forgot Password");
    b3.addActionListener(this);
  b3.setForeground(Color.black);
    b3.setBackground(Color.red);
    b3.setBounds(199, 231, 179, 39);
    panel.add(b3);
    }
  public void actionPerformed(ActionEvent ae){
      if(ae.getSource() == b1){
          Boolean status = false;
              try {
              conn con = new conn();
String sql = "select * from account where username=? and password=?";
              PreparedStatement st = con.c.prepareStatement(sql);

              st.setString(1, textField.getText());
              st.setString(2, passwordField.getText());

              ResultSet rs = st.executeQuery();
              if (rs.next()) {
```

```
                this.setVisible(false);
                new Home().setVisible(true);
            } else
            JOptionPane.showMessageDialog(null,"Invalid Login...!.");

            } catch (Exception e2) {
            e2.printStackTrace();
            }
    }
    if(ae.getSource() == b2){
        setVisible(false);
            Signup su = new Signup();
            su.setVisible(true);
    }
    if(ae.getSource() == b3){
        setVisible(false);
            Forgot forgot = new Forgot();
            forgot.setVisible(true);
    }
}

  public static void main(String[] args) {
        new Login_user().setVisible(true);
  }}
```

8.3 Code For AddBook :

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
public class AddBook extends JFrame implements ActionListener{
   private JPanel contentPane;
   private JTextField t1,t2,t3,t4,t5,t6;
   private JButton b1,b2;
   JComboBox comboBox;

   public static void main(String[] args) {
        new AddBook().setVisible(true);
   }
   public void random() {
      Random rd = new Random();
        t1.setText("" + rd.nextInt(1000 + 1));
   }
   public AddBook() {
      setBounds(600, 200, 518, 442);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
```

```
JLabel l1 = new JLabel("Name");
l1.setForeground(new Color(47, 79, 79));
l1.setFont(new Font("Tahoma", Font.BOLD, 14));
l1.setBounds(73, 84, 90, 22);
contentPane.add(l1);
JLabel l2 = new JLabel("ISBN");
l2.setForeground(new Color(47, 79, 79));
l2.setFont(new Font("Tahoma", Font.BOLD, 14));
l2.setBounds(73, 117, 90, 22);
contentPane.add(l2);
JLabel l3 = new JLabel("Publisher");
l3.setForeground(new Color(47, 79, 79));
l3.setFont(new Font("Tahoma", Font.BOLD, 14));
l3.setBounds(73, 150, 90, 22);
contentPane.add(l3);
JLabel l4 = new JLabel("Price");
l4.setForeground(new Color(47, 79, 79));
l4.setFont(new Font("Tahoma", Font.BOLD, 14));
l4.setBounds(73, 216, 90, 22);
contentPane.add(l4);
JLabel l5 = new JLabel("Pages");
l5.setForeground(new Color(47, 79, 79));
l5.setFont(new Font("Tahoma", Font.BOLD, 14));
l5.setBounds(73, 249, 90, 22);
contentPane.add(l5);
```

```
JLabel l6 = new JLabel("Book_id");
l6.setForeground(new Color(47, 79, 79));
l6.setFont(new Font("Tahoma", Font.BOLD, 14));
l6.setBounds(73, 51, 90, 22);
contentPane.add(l6);
JLabel l7 = new JLabel("Edition");
l7.setForeground(new Color(47, 79, 79));
l7.setFont(new Font("Tahoma", Font.BOLD, 14));
l7.setBounds(73, 183, 90, 22);
contentPane.add(l7);
t1 = new JTextField();
t1.setForeground(new Color(47, 79, 79));
t1.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
t1.setBounds(169, 54, 198, 20);
contentPane.add(t1);
t1.setColumns(10);
t2 = new JTextField();
t2.setForeground(new Color(47, 79, 79));
t2.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
t2.setColumns(10);
t2.setBounds(169, 87, 198, 20);
contentPane.add(t2);
t3 = new JTextField();
t3.setForeground(new Color(47, 79, 79));
t3.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
t3.setColumns(10);
```

```
        t3.setBounds(169, 120, 198, 20);
        contentPane.add(t3);
        t4 = new JTextField();
        t4.setForeground(new Color(47, 79, 79));
        t4.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t4.setColumns(10);
        t4.setBounds(169, 153, 198, 20);
        contentPane.add(t4);
      t5 = new JTextField();
        t5.setForeground(new Color(47, 79, 79));
        t5.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t5.setColumns(10);
        t5.setBounds(169, 219, 198, 20);
        contentPane.add(t5);
        t6 = new JTextField();
        t6.setForeground(new Color(47, 79, 79));
        t6.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t6.setColumns(10);
        t6.setBounds(169, 252, 198, 20);
        contentPane.add(t6);
        comboBox = new JComboBox();
        comboBox.setModel(new DefaultComboBoxModel(new String[]
{ "1", "2", "3", "4", "5", "6", "7", "8", "9" }));
        comboBox.setBounds(173, 186, 194, 20);
        contentPane.add(comboBox);
```

```
b1 = new JButton("Add");
  b1.addActionListener(this);
  b1.setBorder(new CompoundBorder(new LineBorder(new Color(128, 128, 128)), null));
b1.setFont(new Font("Trebuchet MS", Font.BOLD, 15));
  b1.setBounds(102, 300, 100, 33);
b1.setBackground(Color.BLACK);
b1.setForeground(Color.WHITE);
  contentPane.add(b1);
  b2 = new JButton("Back");
  b2.addActionListener(this);
  b2.setBorder(new CompoundBorder(new LineBorder(new Color(105, 105, 105)), null));
  b2.setFont(new Font("Trebuchet MS", Font.BOLD, 15));
  b2.setBounds(212, 300, 108, 33);
b2.setBackground(Color.BLACK);
b2.setForeground(Color.WHITE);
  contentPane.add(b2);
  JPanel panel = new Jpanel();
panel.setBorder(new TitledBorder(new LineBorder(new Color(138, 43, 226), 2), "Add Books", TitledBorder.LEADING,TitledBorder.TOP, null, new Color(0, 0, 255)));
  panel.setBounds(10, 29, 398, 344);
  contentPane.add(panel);
panel.setBackground(Color.WHITE);
contentPane.setBackground(Color.WHITE);
```

```
        random();
    }
    public void actionPerformed(ActionEvent ae){
       try{
          conn con = new conn();
          if(ae.getSource() == b1){
                String sql = "insert into book(book_id, name, isbn, publisher,
edition, price, pages) values(?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement st = con.c.prepareStatement(sql);
             // st.setInt(1, Integer.parseInt(textField.getText()));
                st.setString(1, t1.getText());
                st.setString(2, t2.getText());
                st.setString(3, t3.getText());
                st.setString(4, t4.getText());
                st.setString(5, (String) comboBox.getSelectedItem());
                st.setString(6, t5.getText());
                st.setString(7, t6.getText());
                int rs = st.executeUpdate();
                if (rs > 0)
                JOptionPane.showMessageDialog(null, "Successfully Added");
                else
                JOptionPane.showMessageDialog(null, "Error");
             t1.setText("");
             t2.setText("");
             t3.setText("");
             t4.setText("");
```

```
            t5.setText("");
            t6.setText("");
                st.close();
        }
        if(ae.getSource() == b2){
            this.setVisible(false);
                new Home().setVisible(true);
        }
        con.c.close();
      }catch(Exception e){
      }
   }}
```

8.4 Code For Add Student:

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
public class AddStudent extends JFrame implements ActionListener{
   private JPanel contentPane;
   private JTextField t1,t2,t3;
         private JComboBox comboBox, comboBox_1, comboBox_2, comboBox_3;
   JButton b1,b2;
   public static void main(String[] args) {
```

```
    new AddStudent().setVisible(true);
}
public void random() {
    Random rd = new Random();
      t1.setText("" + rd.nextInt(10000 + 1));
}
public AddStudent() {
    super("Add Student");
      setBounds(700, 200, 550, 450);
      contentPane = new JPanel();
      contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
      setContentPane(contentPane);
      contentPane.setLayout(null);
      JLabel l1 = new JLabel("Student_id");
      l1.setForeground(new Color(25, 25, 112));
      l1.setFont(new Font("Tahoma", Font.BOLD, 14));
      l1.setBounds(64, 63, 102, 22);
      contentPane.add(l1);
      JLabel l2 = new JLabel("Name");
      l2.setForeground(new Color(25, 25, 112));
      l2.setFont(new Font("Tahoma", Font.BOLD, 14));
      l2.setBounds(64, 97, 102, 22);
      contentPane.add(l2);
      JLabel l3 = new JLabel("Father's Name");
      l3.setForeground(new Color(25, 25, 112));
      l3.setFont(new Font("Tahoma", Font.BOLD, 14));
```

```
        l3.setBounds(64, 130, 102, 22);
        contentPane.add(l3);
        JLabel l4 = new JLabel("Branch");
        l4.setForeground(new Color(25, 25, 112));
        l4.setFont(new Font("Tahoma", Font.BOLD, 14));
        l4.setBounds(64, 209, 102, 22);
        contentPane.add(l4);
      JLabel l5 = new JLabel("Year");
        l5.setForeground(new Color(25, 25, 112));
        l5.setFont(new Font("Tahoma", Font.BOLD, 14));
        l5.setBounds(64, 242, 102, 22);
        contentPane.add(l5);
        JLabel l6 = new JLabel("Semester");
        l6.setForeground(new Color(25, 25, 112));
        l6.setFont(new Font("Tahoma", Font.BOLD, 14));
        l6.setBounds(64, 275, 102, 22);
        contentPane.add(l6);
        t1 = new JTextField();
//      t1.setEditable(false);
        t1.setForeground(new Color(47, 79, 79));
        t1.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t1.setBounds(174, 66, 156, 20);
        contentPane.add(t1);
        t1.setColumns(10);
        t2 = new JTextField();
        t2.setForeground(new Color(47, 79, 79));
```

```
        t2.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t2.setColumns(10);
        t2.setBounds(174, 100, 156, 20);
        contentPane.add(t2);
        t3 = new JTextField();
        t3.setForeground(new Color(47, 79, 79));
        t3.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        t3.setColumns(10);
        t3.setBounds(174, 133, 156, 20);
        contentPane.add(t3);
        comboBox = new JComboBox();
        comboBox.setModel(new DefaultComboBoxModel(new String[]
{ "Mechanical", "CSE", "IT", "Civil", "Automobile", "EEE", "Other" }));
        comboBox.setForeground(new Color(47, 79, 79));
        comboBox.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        comboBox.setBounds(176, 211, 154, 20);
        contentPane.add(comboBox);
        comboBox_1 = new JComboBox();
        comboBox_1.setModel(new DefaultComboBoxModel(new String[]
{ "First", "Second", "Third", "Four" }));
        comboBox_1.setForeground(new Color(47, 79, 79));
        comboBox_1.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        comboBox_1.setBounds(176, 244, 154, 20);
        contentPane.add(comboBox_1);
        comboBox_2 = new JComboBox();
        comboBox_2.setModel(
```

```
new DefaultComboBoxModel(new String[] { "1st", "2nd", "3rd", "4th",
"5th", "6th", "7th", "8th" }));
        comboBox_2.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        comboBox_2.setForeground(new Color(47, 79, 79));
        comboBox_2.setBounds(176, 277, 154, 20);
        contentPane.add(comboBox_2);
        b1 = new JButton("ADD");
        b1.addActionListener(this);
        b1.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        b1.setBounds(64, 321, 111, 33);
      b1.setBackground(Color.BLACK);
      b1.setForeground(Color.WHITE);
        contentPane.add(b1);
        b2 = new JButton("Back");
        b2.addActionListener(this);
        b2.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        b2.setBounds(198, 321, 111, 33);
      b2.setBackground(Color.BLACK);
      b2.setForeground(Color.WHITE);
        contentPane.add(b2);
        JLabel l7 = new JLabel("Course");
        l7.setForeground(new Color(25, 25, 112));
        l7.setFont(new Font("Tahoma", Font.BOLD, 14));
        l7.setBounds(64, 173, 102, 22);
        contentPane.add(l7);
```

```
        comboBox_3 = new JComboBox();
        comboBox_3.setModel(new DefaultComboBoxModel(
new String[] { "B.E", "B.Tech", "M.Tech", "MBA", "BBA", "BCA",
"B.Sc", "M.Sc", "B.Com", "M.Com" }));
        comboBox_3.setForeground(new Color(47, 79, 79));
        comboBox_3.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
        comboBox_3.setBounds(176, 176, 154, 20);
        contentPane.add(comboBox_3);
        JPanel panel = new JPanel();
        panel.setBorder(new TitledBorder(new LineBorder(new Color(102,
205, 170), 2, true), "Add-Student",
TitledBorder.LEADING, TitledBorder.TOP, null, new Color(30, 144,
255)));
        panel.setBackground(new Color(211, 211, 211));
        panel.setBounds(10, 38, 358, 348);
      contentPane.setBackground(Color.WHITE);
      panel.setBackground(Color.WHITE);
        contentPane.add(panel);
        random();
   }
   public void actionPerformed(ActionEvent ae){
      try{
         if(ae.getSource() == b1){
            try{
            conn con = new conn();
```

```
            String sql = "insert into student(student_id, name, father, course,
branch, year, semester) values(?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement st = con.c.prepareStatement(sql);
                st.setString(1, t1.getText());
                st.setString(2, t2.getText());
                st.setString(3, t3.getText());
                st.setString(4, (String) comboBox_3.getSelectedItem());
                st.setString(5, (String) comboBox.getSelectedItem());
                st.setString(6, (String) comboBox_1.getSelectedItem());
                st.setString(7, (String) comboBox_2.getSelectedItem());
                int i = st.executeUpdate();
                if (i > 0){
                JOptionPane.showMessageDialog(null, "Successfully Added");
                this.setVisible(false);
                new Home().setVisible(true);
            }
                else
                JOptionPane.showMessageDialog(null, "error");
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        if(ae.getSource() == b2){
            this.setVisible(false);
                new Home().setVisible(true);
        }
```

```
        }catch(Exception e){
        }
    }
}
```

8.5 Code For ReturnBook :

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import com.toedter.calendar.JDateChooser;
import java.awt.event.*;
import java.sql.*;
import java.text.SimpleDateFormat;
public class ReturnBook extends JFrame implements ActionListener{
    private JPanel contentPane;
    private JTextField textField;
    private JTextField textField_1;
    private JTextField textField_2;
    private JTextField textField_3;
    private JTextField textField_4;
    private JTextField textField_5;
    private JTextField textField_6;
    private JButton b1,b2,b3;
    private JDateChooser dateChooser;
    public static void main(String[] args) {
        new ReturnBook().setVisible(true);
```

```
}
public void delete() {
    try {
        conn con = new conn();
        String sql = "delete from issueBook where book_id=?";
        PreparedStatement st = con.c.prepareStatement(sql);
        st.setString(1, textField.getText());
        int i = st.executeUpdate();
        if (i > 0)
            JOptionPane.showConfirmDialog(null, "Book Returned");
        else
            JOptionPane.showMessageDialog(null, "error in Deleting");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e);
        e.printStackTrace();
    }
}
public ReturnBook() {
    setBounds(450, 300, 617, 363);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setBackground(Color.WHITE);
    contentPane.setLayout(null);
    JLabel lblNewLabel = new JLabel("Book_id");
    lblNewLabel.setForeground(new Color(0, 0, 0));
```

```
lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 14));
lblNewLabel.setBounds(52, 52, 87, 24);
contentPane.add(lblNewLabel);
JLabel lblStudentid = new JLabel("Student_id");
lblStudentid.setForeground(Color.BLACK);
lblStudentid.setFont(new Font("Tahoma", Font.BOLD, 14));
lblStudentid.setBounds(243, 52, 87, 24);
contentPane.add(lblStudentid);
JLabel lblBook = new JLabel("Book");
lblBook.setForeground(Color.BLACK);
lblBook.setFont(new Font("Tahoma", Font.BOLD, 14));
lblBook.setBounds(52, 98, 71, 24);
contentPane.add(lblBook);
JLabel lblName = new JLabel("Name");
lblName.setForeground(Color.BLACK);
lblName.setFont(new Font("Tahoma", Font.BOLD, 14));
lblName.setBounds(300, 98, 71, 24);
contentPane.add(lblName);
JLabel lblCourse = new JLabel("Course");
lblCourse.setForeground(Color.BLACK);
lblCourse.setFont(new Font("Tahoma", Font.BOLD, 14));
lblCourse.setBounds(52, 143, 87, 24);
contentPane.add(lblCourse);
JLabel lblBranch = new JLabel("Branch");
lblBranch.setForeground(Color.BLACK);
lblBranch.setFont(new Font("Tahoma", Font.BOLD, 14));
```

```
lblBranch.setBounds(303, 144, 68, 24);
contentPane.add(lblBranch);
JLabel lblDateOfIssue = new JLabel("Date of Issue");
lblDateOfIssue.setForeground(Color.BLACK);
lblDateOfIssue.setFont(new Font("Tahoma", Font.BOLD, 14));
lblDateOfIssue.setBounds(52, 188, 105, 29);
contentPane.add(lblDateOfIssue);
JLabel lblDateOfReturn = new JLabel("Date of Return");
lblDateOfReturn.setForeground(Color.BLACK);
lblDateOfReturn.setFont(new Font("Tahoma", Font.BOLD, 14));
lblDateOfReturn.setBounds(52, 234, 118, 29);
contentPane.add(lblDateOfReturn);
textField = new JTextField();
textField.setForeground(new Color(105, 105, 105));
textField.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
textField.setBounds(128, 56, 105, 20);
contentPane.add(textField);
textField.setColumns(10);
textField_1 = new JTextField();
textField_1.setForeground(new Color(105, 105, 105));
textField_1.setFont(new Font("Trebuchet MS", Font.BOLD, 14));
textField_1.setBounds(340, 56, 93, 20);
contentPane.add(textField_1);
textField_1.setColumns(10);
b1 = new JButton("Search");
b1.addActionListener(this);
```

```
b1.setBounds(443, 52, 105, 29);
b1.setBackground(Color.BLACK);
b1.setForeground(Color.WHITE);
contentPane.add(b1);
textField_2 = new JTextField();
textField_2.setEditable(false);
textField_2.setForeground(new Color(0, 100, 0));
textField_2.setFont(new Font("Trebuchet MS", Font.BOLD, 13));
textField_2.setBounds(128, 102, 162, 20);
contentPane.add(textField_2);
textField_2.setColumns(10);
textField_3 = new JTextField();
textField_3.setEditable(false);
textField_3.setForeground(new Color(0, 100, 0));
textField_3.setFont(new Font("Trebuchet MS", Font.BOLD, 13));
textField_3.setColumns(10);
textField_3.setBounds(369, 102, 179, 20);
contentPane.add(textField_3);
textField_4 = new JTextField();
textField_4.setEditable(false);
textField_4.setForeground(new Color(0, 100, 0));
textField_4.setFont(new Font("Trebuchet MS", Font.BOLD, 13));
textField_4.setColumns(10);
textField_4.setBounds(128, 147, 162, 20);
contentPane.add(textField_4);
```

```
textField_5 = new JTextField();
textField_5.setForeground(new Color(0, 100, 0));
textField_5.setFont(new Font("Trebuchet MS", Font.BOLD, 13));
textField_5.setEditable(false);
textField_5.setColumns(10);
textField_5.setBounds(369, 147, 179, 20);
contentPane.add(textField_5);
textField_6 = new JTextField();
textField_6.setForeground(new Color(0, 100, 0));
textField_6.setFont(new Font("Trebuchet MS", Font.BOLD, 13));
textField_6.setEditable(false);
textField_6.setColumns(10);
textField_6.setBounds(167, 194, 162, 20);
contentPane.add(textField_6);
dateChooser = new JDateChooser();
dateChooser.setBorder(new LineBorder(new Color(0, 0, 0), 0, true));
dateChooser.setBounds(167, 234, 172, 29);
contentPane.add(dateChooser);
b2 = new JButton("Return");
b2.addActionListener(this);
b2.setFont(new Font("Trebuchet MS", Font.BOLD, 15));
b2.setBorder(new LineBorder(new Color(0, 0, 0), 0, true));
b2.setBounds(369, 179, 149, 30);
b2.setBackground(Color.BLACK);
b2.setForeground(Color.WHITE);
contentPane.add(b2);
```

```
        b3 = new JButton("Back");
        b3.addActionListener(this);
        b3.setFont(new Font("Trebuchet MS", Font.BOLD, 15));
        b3.setBorder(new LineBorder(new Color(0, 0, 0), 0, true));
        b3.setBounds(369, 231, 149, 30);
        b3.setBackground(Color.BLACK);
        b3.setForeground(Color.WHITE);
        contentPane.add(b3);
        JPanel panel = new JPanel();
        panel.setBorder(new TitledBorder(new LineBorder(new Color(255, 69,
0), 2, true), "Return-Panel",
                TitledBorder.LEADING, TitledBorder.TOP, null, new Color(220,
20, 60)));
        panel.setBounds(10, 24, 569, 269);
        panel.setBackground(Color.WHITE);
        contentPane.add(panel);
    }
    public void actionPerformed(ActionEvent ae){
        try{
            conn con = new conn();
            if(ae.getSource() == b1){
                String sql = "select * from issueBook where student_id = ? and
book_id =?";
                try (PreparedStatement st = con.c.prepareStatement(sql)) {
                    st.setString(1, textField_1.getText());
                    st.setString(2, textField.getText());
```

```
                try (ResultSet rs = st.executeQuery()) {
                    if (rs.next()) {
                        textField_2.setText(rs.getString("bname"));
                        textField_3.setText(rs.getString("sname"));
                        textField_4.setText(rs.getString("course"));
                        textField_5.setText(rs.getString("branch"));
                        // Format the date before setting it to the text field
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String formattedDate = sdf.format(rs.getDate("dateOfIssue"));
                        textField_6.setText(formattedDate);
                    } else {
JOptionPane.showMessageDialog(null, "No records found for the given student ID and book ID");
                    }
                }
            } catch (SQLException e) {
 JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
                e.printStackTrace();
            }
        }
        if(ae.getSource() == b2){
              String sql = "insert into returnBook(book_id, student_id, bname, sname,course, branch, dateOfIssue, dateOfReturn) values(?, ?, ?, ?, ?, ?, ?, ?)";
            try {
                PreparedStatement st = con.c.prepareStatement(sql);
```

```
                st.setString(1, textField.getText());
                st.setString(2, textField_1.getText());
                st.setString(3, textField_2.getText());
                st.setString(4, textField_3.getText());
                st.setString(5, textField_4.getText());
                st.setString(6, textField_5.getText());
                st.setString(7, textField_6.getText());
  // Format the date before setting it to the prepared statement
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
     String formattedDate = sdf.format(dateChooser.getDate());
                st.setString(8, formattedDate);

                int i = st.executeUpdate();
                if (i > 0) {
                   JOptionPane.showMessageDialog(null, "Processing..");
                   delete();
                } else {
                   JOptionPane.showMessageDialog(null, "error");
                }
             } catch (SQLException e) {
                            JOptionPane.showMessageDialog(null,  "Error:  "  +
e.getMessage());
                e.printStackTrace();
             }
          }
          if(ae.getSource() == b3){
```

```
            this.setVisible(false);
            new Home().setVisible(true);
        }
    } catch(Exception e){
        JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
        e.printStackTrace();} }}
```

CHAPTER 9
REFERENCES

1. Oracle. "Java SE Documentation." Available online: https://docs.oracle.com/en/java/javase/index.html

2. Oracle. "Java Swing Documentation." Available online: https://docs.oracle.com/javase/tutorial/uiswing/index.html

3. MySQL Documentation. "MySQL 8.0 Reference Manual." Available online: https://dev.mysql.com/doc/refman/8.0/en/

4. W3Schools. "SQL Tutorial." Available online: https://www.w3schools.com/sql/

5. Baeldung. "Introduction to JDBC." Available online: https://www.baeldung.com/java-jdbc

6. GeeksforGeeks. "Introduction to Software Testing." Available online: https://www.geeksforgeeks.org/software-testing/

7. Baeldung. "Introduction to Integration Testing." Available online: https://www.baeldung.com/integration-testing