



MAJOR PROJECT REPORT
CAP680: PROGRAMMING IN JAVA-LABORATORY

SUBMITTED TO
Lovely Professional University
in partial fulfilment of the requirements for the award of degree of
Master of Computer Applications

Submitted by

Vyshnav P

LOVELY FACULTY OF COMPUTER APPLICATION
LOVELY PROFESSIONAL UNIVERSITY
PUNJAB
(April 2023)

| Table of Content | | |
|------------------|-------------------------------------|-------------------|
| <i>Sr. No.</i> | <i>Content</i> | <i>Page Index</i> |
| 1 | Introduction about project | 1-2 |
| 2 | Project Modules and its description | 2-5 |
| 3 | Coding & Design | 5-54 |
| 4 | Screenshot output | 55-59 |

ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude and profound appreciation to Jaswinder Singh, esteemed faculty at our prestigious institute, for his unwavering guidance, unwavering support, and invaluable contributions throughout the arduous journey of developing our major project, "Digital Library."

Mr. Singh's unparalleled expertise and profound knowledge in the realm of digital libraries have been instrumental in not only shaping our project but also ensuring its resounding success. His profound insights, astute observations, and thoughtful suggestions have propelled us towards achieving our goals and surpassing our own expectations.

Furthermore, we would like to express our deep indebtedness to Mr. Singh for his immeasurable patience, unfaltering encouragement, and steadfast understanding during every stage of this project. His unwavering commitment to our growth and development has been a constant source of inspiration and motivation.

We would also like to acknowledge his exceptional dedication and availability, as he consistently went above and beyond his role as a mentor to provide us with continuous guidance and mentorship. Mr. Singh's approachability, responsiveness, and willingness to address our concerns and queries have been instrumental in fostering a nurturing learning environment.

Lastly, we would like to express our gratitude to the institute for providing us with the opportunity to work under the mentorship of such an exceptional individual. It is through their support and belief in our abilities that we were able to undertake this challenging project and emerge triumphant.

In conclusion, we cannot emphasize enough the immense impact Mr. Jaswinder Singh has had on our project and our personal growth. His exceptional guidance, support, and expertise have been instrumental in shaping our understanding and proficiency in the field of digital libraries. We consider ourselves fortunate to have had the privilege of working with him and will forever be grateful for his unwavering commitment to our success.

Once again, we extend our sincerest gratitude to Mr. Singh, without whom the successful completion of our major project would not have been possible.

INTRODUCTION

Overview of the Project:

- This digital library is a platform that allows users to access and search through a vast collection of digital resources such as books and admin, librarian can edit the books by name and quantity as well the database for users. In this project, we have developed a digital library system using Java programming language and NetBeans Integrated Development Environment (IDE).
- The main objective of this project is to create a user-friendly digital library system that enables users to search, borrow, and return digital resources. The system will also provide administrators with the ability to manage the library's resources, users, and borrowing policies.
- The system will consist of a graphical user interface (GUI) that enables users to search the library's resources by title, author, subject, and keyword. Users will also be able to view the availability status of resources and request to borrow them.
- Administrators will have access to additional functionalities such as adding new resources to the library, updating existing resources, managing user accounts, and setting borrowing policies.
- Overall, the digital library system project using Java and NetBeans IDE has provided us an excellent opportunity to improve our programming skills, database management, and GUI development.

OBJECTIVE OF PROJECT

The main objective of the Digital Library project is to design and develop a system that provides an efficient and user-friendly platform for managing digital resources such as books. The following are the specific objectives of the project:

- To create a database of digital resources that can be searched and pre-register by users.
- To provide a user-friendly interface for Admin, Librarian and Users to search and access digital resources.
- To implement a borrowing and returning system for users to borrow and return digital resources.
- To provide administrators with the ability to manage the library's resources, users, and borrowing policies.
- To ensure that the system is secure and protected from unauthorized access.
- To ensure that the system is scalable and can accommodate future expansions and upgrades.

MODULE SPECIFICATION

1. Admin:

The Admin module of the Digital Library system is designed to provide authorized personnel with access to a range of functionalities to manage the library's resources and users. The admin module includes the following features:

- a) **Add and Remove Books:** The Admin can add new books to the library's database by entering the book's title, author, publication date, and other relevant details. The admin can also remove books from the database when necessary.
- b) **Manage Book Quantity:** The Admin can update the quantity of books in the library's inventory. This feature enables the admin to keep track of the availability of each book and ensure that there are enough copies for users to borrow.
- c) **Manage Librarian and User Accounts:** The Admin has the ability to create new Librarian accounts and manage existing accounts. The admin can also create new User accounts, update User information, and manage User borrowing privileges.
- d) **Issue and Submit Books:** The Admin can issue books to Users when requested and track the status of each book. The admin can also receive books from Users when they are returned and update the inventory accordingly.
- e) **View Graphical Analysis of Books by Quantity:** The Admin can view graphical analyses of the library's resources, including books, by their quantity. This feature provides a visual representation of the library's inventory and helps the admin make informed decisions about managing the library's resources.

Overall, the Admin module of the Digital Library system provides a comprehensive set of functionalities to manage the library's resources and users effectively. The module is designed with security and scalability in mind, ensuring that authorized personnel can access the system's features safely and efficiently.

2. Librarian:

The Librarian module of the Digital Library system is designed to provide authorized personnel with access to functionalities to manage the borrowing and returning of books. The Librarian module includes the following features:

- a) **Issue and Submit Books:** The Librarian can issue books to Users when requested and track the status of each book. The Librarian can also receive books from Users when they are returned and update the inventory accordingly.
- b) **Manage Book Quantity:** The Librarian can update the quantity of books in the library's inventory. This feature enables the Librarian to keep track of the availability of each book and ensure that there are enough copies for users to borrow.
- c) **Approve Pre-Registration:** The Librarian has the ability to approve pre-registration requests made by Users for a specific book. Once the Librarian approves the request, the User can borrow the book without further approval.
- d) **View Borrowing History:** The Librarian can view the borrowing history of Users, including the books borrowed and the return dates. This feature allows the Librarian to monitor the borrowing patterns of Users and ensure that the library's resources are being used efficiently.
- e) **Generate Reports:** The Librarian can generate reports on the library's resources, including books borrowed, returned, and overdue. This feature provides the Librarian with insights into the library's usage patterns and enables them to make informed decisions about managing the library's resources.
- f) **Feature: OTP verification while sign up.** It is important to note that the Librarian does not have permission to add new Librarian accounts to the system. Their access is restricted to managing the borrowing and returning of books and related functionalities.

Overall, the Librarian module of the Digital Library system provides a comprehensive set of functionalities to manage the borrowing and returning of books. The module is designed with security and scalability in mind, ensuring that authorized personnel can access the system's features safely and efficiently.

3. User:

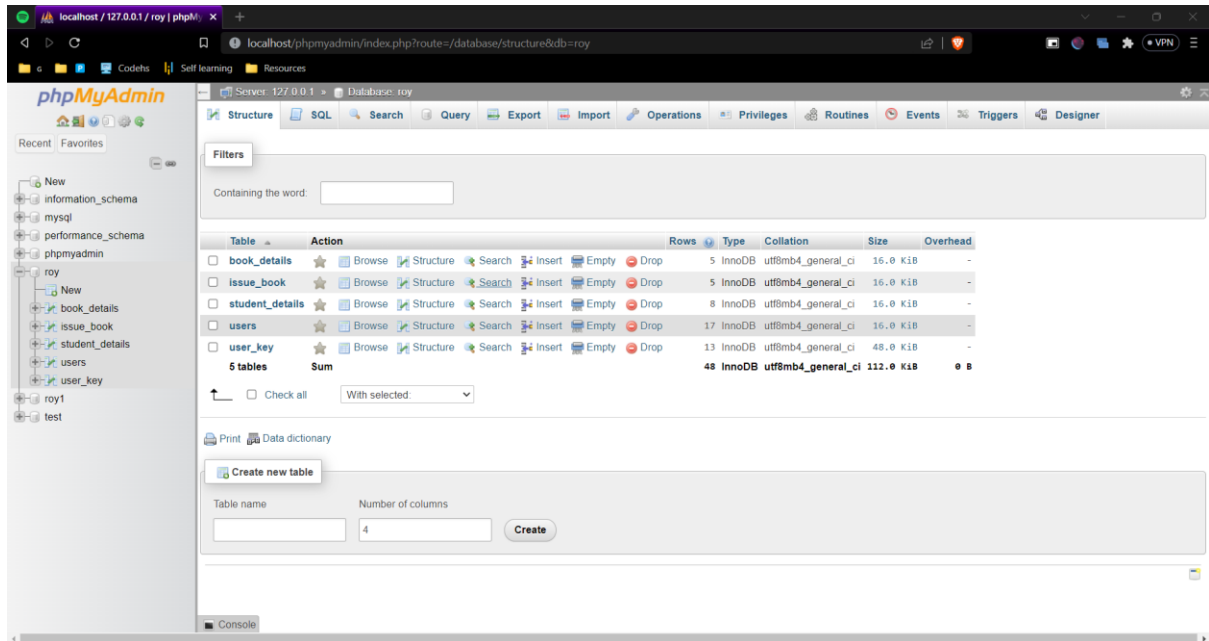
The User module of the Digital Library system is designed to provide users with access to functionalities to register, search for books, and pre-register for them. The User module includes the following features:

- a) Register: The User can register themselves with the library by providing their personal information, including their name, email, and contact details. This feature enables the User to access the library's resources and borrowing privileges.
- b) Search Books: The User can search for books in the library's inventory by entering relevant keywords such as author, title, or subject. This feature allows the User to find the books they need quickly and efficiently.
- c) Pre-Register for Books: The User can pre-register for a book that is currently borrowed by another User. This feature enables the User to reserve the book and receive approval from the Librarian for borrowing it once it is returned.
- d) Talk with Chatbot: The User has the ability to interact with a Chatbot for assistance. The Chatbot can answer common questions related to borrowing, returning, and renewing books, as well as provide recommendations for books based on the User's interests.
- e) View Available Books: The User can view the books that are currently available for borrowing. This feature enables the User to see what resources are immediately accessible to them.

Overall, the User module of the Digital Library system provides a range of functionalities to make the borrowing and returning of books efficient and accessible for Users. The module is designed to be user-friendly and easy to navigate, allowing Users to find the resources they need quickly and easily.

Design & Code

1. Database Schema:



The database schema for the Digital Library system includes the following primary keys and foreign keys:

We have used jar file MySQL.

- a. Primary Keys:
 - i. book_details table: book_id
 - ii. issue_book table: id
 - iii. student_details table: student_id
 - iv. users table: id
 - v. users_key table: key_id
- b. Foreign Keys:
 - i. users_key table: book_id (references book_details table: book_id)
 - ii. users_key table: student_id (references student_details table: student_id)

```
2. package OG;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.SQLException;
6. public class BACKENDCONNECTION {
7.     static Connection con = null;
8.     public static Connection getConnection() {
9.         try{
10.             Class.forName("com.mysql.cj.jdbc.Driver");
```

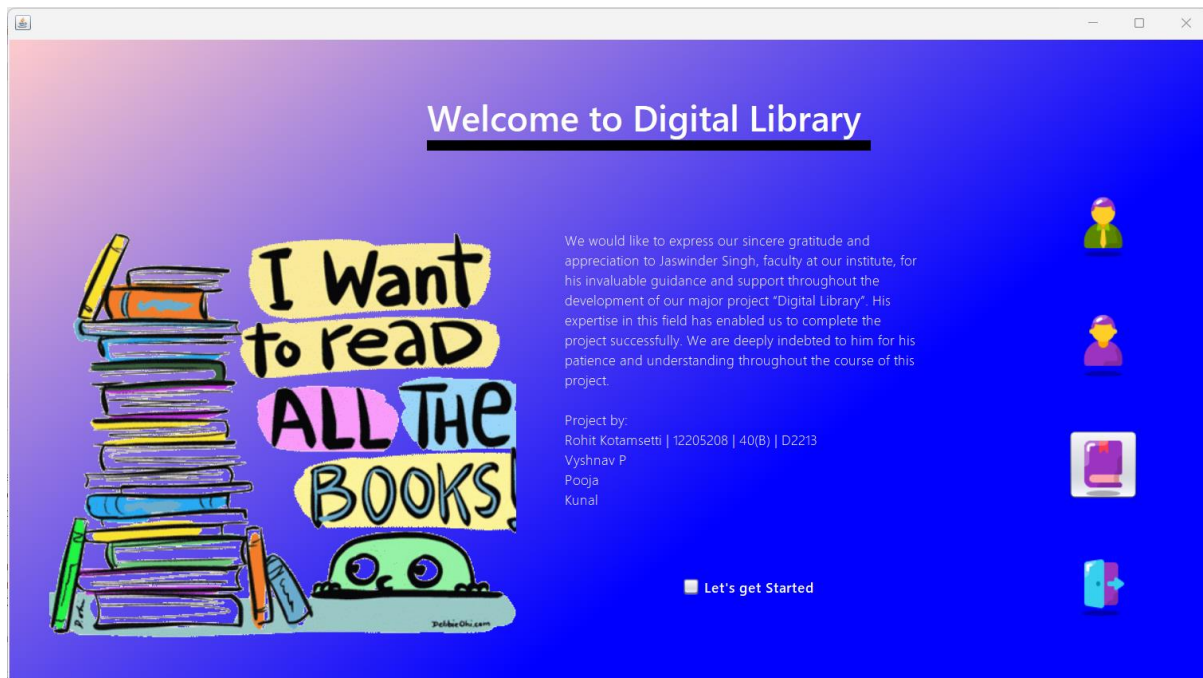


```

11.         con =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/roy","root","")
    );
12.     }
13.         catch(ClassNotFoundException | SQLException e){
14.             }return con;
15.     }
16. }

```

17. Home:



The homepage of the Digital Library system is designed to provide Users with a clear and intuitive interface to access the various modules of the system. The homepage includes the following features:

- **Checkbox:** The homepage includes a checkbox that Users can select to indicate whether they are an Admin/Librarian or a regular User. This feature enables the system to direct Users to the appropriate module based on their authorization level.
- **Admin/Librarian Button:** If the User selects the Admin/Librarian checkbox, they will see a button that allows them to access the Admin/Librarian module. This button enables authorized personnel to manage the library's resources, including adding and removing books, managing book quantities, approving pre-registration requests, and generating reports.

```

➤ if(start.isSelected()){
➤     digital.Loginpage l = new digital.Loginpage();
➤     l.show();
➤     dispose();
➤ }
➤ else
➤ JOOptionPane.showMessageDialog(this, "Please checkmark");
➤

```

- User Button: If the User selects the User checkbox, they will see a button that allows them to access the User module. This button enables Users to register with the library, search for books, pre-register for books, and interact with a Chatbot for assistance.

```
➤ if(start.isSelected()){
➤     Login l = new Login();
➤     l.show();
➤     dispose();
➤ }
➤ else
➤     JOptionPane.showMessageDialog(this, "Please checkmark");
➤
```

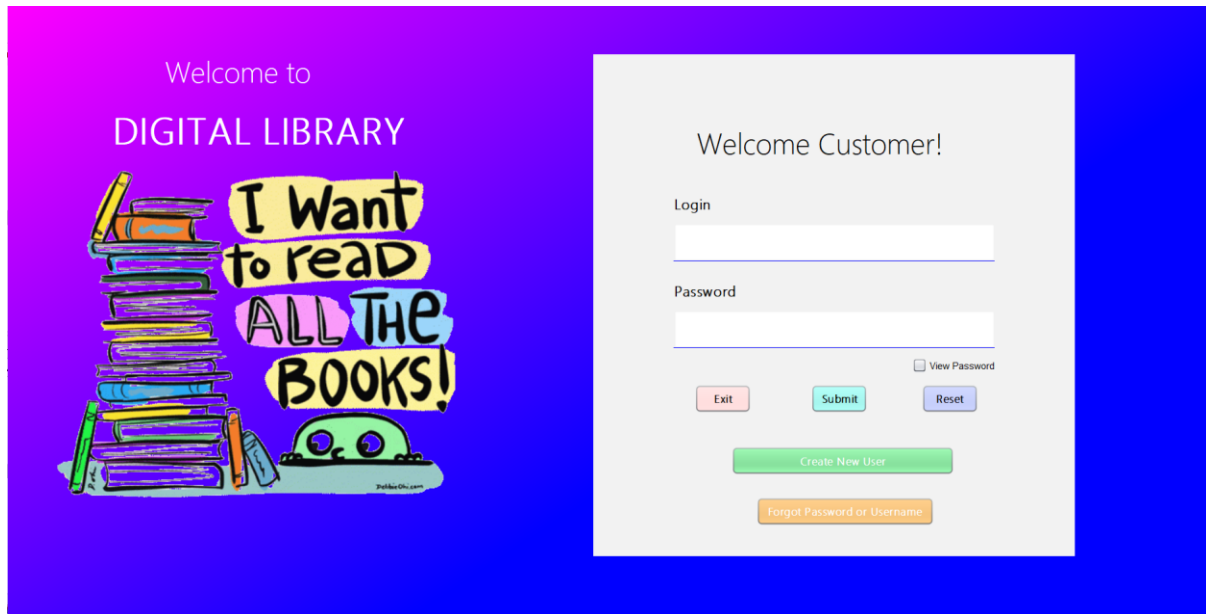
- View Report Button: The homepage includes a button that allows authorized personnel to generate reports on the library's resources. This feature provides insights into the usage patterns of the library's resources and enables the library to make informed decisions about managing its resources.

```
if(start.isSelected()){
try{
File file = new
File("C:\\Users\\kotar\\Documents\\NetBeansProjects\\Customer\\src\\Rep
ort\\12205208.pdf");
if(file.exists()){
if(Desktop.isDesktopSupported()){
Desktop.getDesktop().open(file);
}else {
JOptionPane.showMessageDialog(this,"Unsupported");
} }
}catch(HeadlessException | IOException){}
} else JOptionPane.showMessageDialog(this, "Please checkmark");
```

- Exit Button: The homepage includes an Exit button that allows Users to exit the system. This feature ensures that Users can easily and quickly exit the system when they have completed their tasks.

Overall, the homepage of the Digital Library system is designed to provide a clear and user-friendly interface for Users to access the various modules of the system. The homepage is designed with security and scalability in mind, ensuring that authorized personnel can access the system's features safely and efficiently.

4. User:



➤ Gradient used KGradientpanel.jar

Validation for login:

```
public boolean valid(){
    String name = username.getText();
    String pass = password.getText();
    if(name.equals("") || pass.equals("")){
        JOptionPane.showMessageDialog(this, "Please recheck Username Password");
        return false;
    }
    return true;
}
```

➤ Validation with database:

```
if(valid()){
    String name =username.getText();
    String pass =password.getText();
    try {
        Connection con =BACKENDCONNECTION.getConnection();
        String sql ="select * from users where email=? and password=? and
status='user'";
        PreparedStatement ps =con.prepareStatement(sql);
        ps.setString(1, name);
        ps.setString(2, pass);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```

ResultSet rs= ps.executeQuery();
if(rs.next()){
    user l = new user();
    l.show();
    dispose();
}
else{
    JOptionPane.showMessageDialog(this, "Your password/username is wrong
please check again");
}
} catch (HeadlessException | SQLException e) {
    System.err.println(e);
}}

```

➤ Validation for Database:

```

String query = "INSERT INTO users
(email,password,name,contact,sissueDate,details,status) VALUES (?, ?, ?, ?, ?,
?, 'user')";
try ( Connection con = BACKENDCONNECTION.getConnection();
PreparedStatement stmt = con.prepareStatement(query)) {
    stmt.setString(1, email);
    stmt.setString(2, password);
    stmt.setString(3, firstName);
    stmt.setString(4, phone);
}

```

```

stmt.setDate(5, sissueDate);
stmt.setString(6, details);
int rowsInserted = stmt.executeUpdate();
if (rowsInserted > 0) {
System.out.println("A new customer was inserted successfully!");
}
} catch (SQLException ex) {
System.out.println("An error occurred while inserting the customer: " +
ex.getMessage());
}
String query1 = "INSERT INTO student_details (name) VALUES (?)";
try ( Connection con = BACKENDCONNECTION.getConnection();
PreparedStatement stmt = con.prepareStatement(query1)) {
stmt.setString(1, firstName);
int rowsInserted = stmt.executeUpdate();
if (rowsInserted > 0) {
System.out.println("A new customer was inserted successfully!");
}
} catch (SQLException ex) {
System.out.println("An error occurred while inserting the customer: " +
ex.getMessage());
}
}

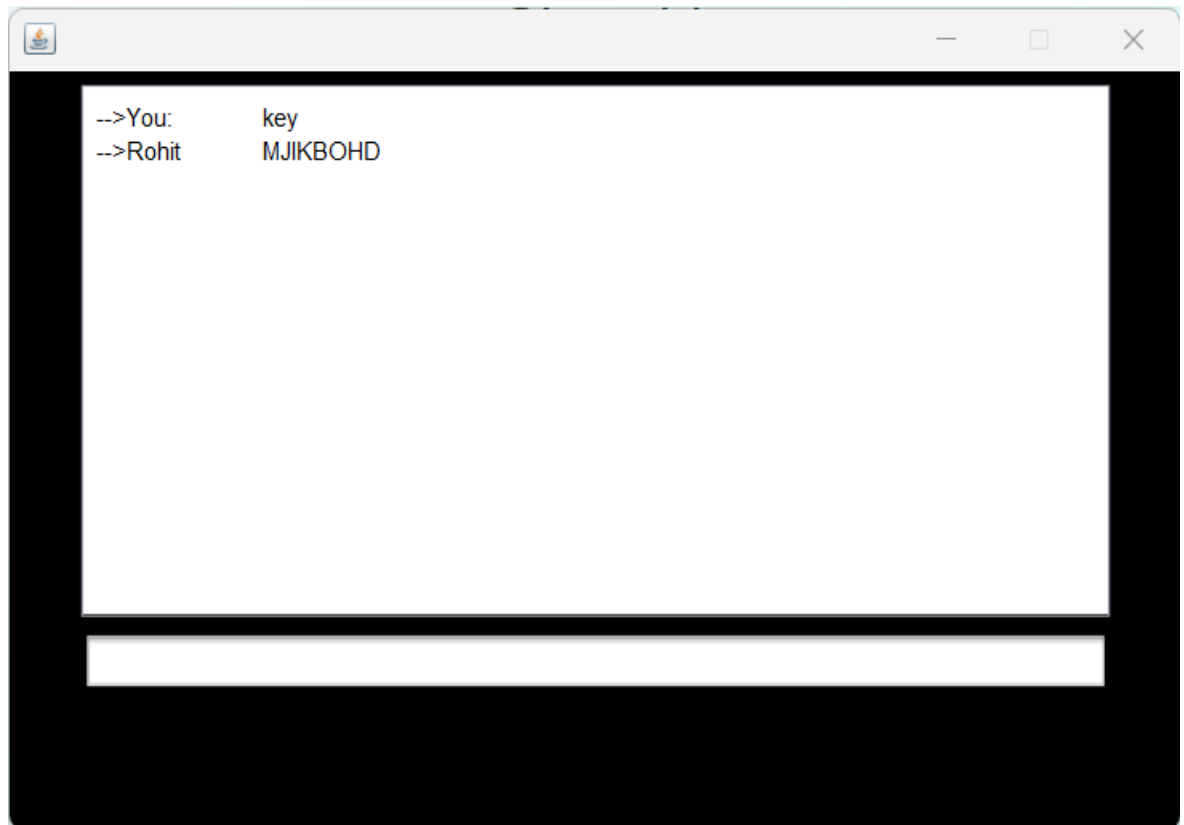
```

➤ Other minor checks

```

└ String phone = u_ph.getText();
if(phone.matches("[0-9]*$") && phone.length() == 10){
u_ph.setBackground(Color.green);
}
else
u_ph.setBackground(Color.red);
char c = evt.getKeyChar();
    if(!Character.isAlphabetic(c)){
        evt.consume();
    }
char c = evt.getKeyChar();
    if(!Character.isDigit(c)){
        evt.consume();
    }
}

```



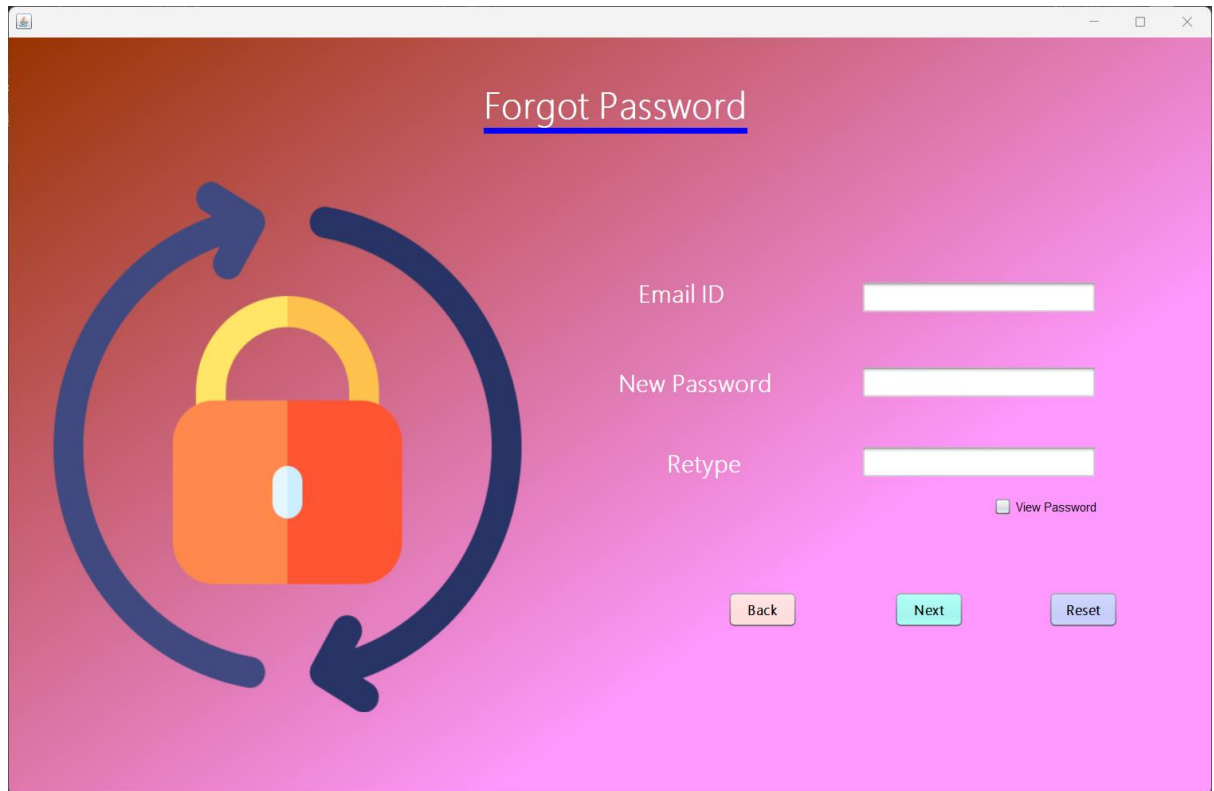
➤ Chatbot:

```
String[][] chatBot={
    {"Key","key","KEY"},
    {"IO89TC87","A75FKLS0","785TY98D","J0A0V0A0","MJIKBOHD","hQ7804yv","c8VveG
8c","(Rohit is unavailable)"},
};
public static void main(String[] args){
    new ChatBot();
}
public ChatBot(){
    setSize(600,400);
    setResizable(false);
    dialog.setEditable(false);
    input.addKeyListener(this);
    p.add(scroll);
    p.add(input);
    p.setBackground(new Color(0,0,0));
    add(p);
    setVisible(true);
}
public void keyPressed(KeyEvent e){
    if(e.getKeyCode()==KeyEvent.VK_ENTER){
        input.setEditable(false);
        String quote=input.getText();
        input.setText("");
    }
}
```

```

addText("-->You:\t"+quote);
quote.trim();
while(
quote.charAt(quote.length()-1)=='!' ||
quote.charAt(quote.length()-1)=='.' ||
quote.charAt(quote.length()-1)=='?'
){
quote=quote.substring(0,quote.length()-1);
}
quote.trim();
byte response=0;
int j=0;
while(response==0){
if(inArray(quote.toLowerCase(),chatBot[j*2])){
response=2;
int r=(int)Math.floor(Math.random()*chatBot[(j*2)+1].length);
addText("\n-->Rohit\t"+chatBot[(j*2)+1][r]);
}
j++;
(j*2==chatBot.length-1 && response==0){
response=1;
}
}
if(response==1){
r=(int)Math.floor(Math.random()*chatBot[chatBot.length-1].length);
addText("\n-->Rohit\t"+chatBot[chatBot.length-1][r]);
}
addText("\n");
}
}
public void keyReleased(KeyEvent e){
if(e.getKeyCode()==KeyEvent.VK_ENTER){
input.setEditable(true);
}
}
public void keyTyped(KeyEvent e){}
public void addText(String str){
dialog.setText(dialog.getText()+str);
}
public boolean inArray(String in,String[] str){
boolean match=false;
for(int i=0;i<str.length;i++){
if(str[i].equals(in)){
match=true;
}
}
return match;
}
}

```



➤ Validation for password:

```
public boolean valid(){
    String Email = email.getText();
    String Password = password.getText();
    String Password2 = password1.getText();
    if(Email.equals("") || !Password.endsWith(Password2) ||
    !Password2.equals(Password)){
        JOptionPane.showMessageDialog(this, "Invalid");
        return false;
    }
    return true;
}
```

➤ Database updating:

```
public void reset(){
    String email = this.email.getText();
    String password = this.password.getText();
    String query = "UPDATE users SET password = ? WHERE email = ?";
    try (Connection con = BACKENDCONNECTION.getConnection();
        PreparedStatement stmt = con.prepareStatement(query)) {
        stmt.setString(1, password);
        stmt.setString(2, email);
    }
```



```

int rowsUpdated = stmt.executeUpdate();
if (rowsUpdated > 0) {
    System.out.println("Password updated successfully!");
} else {
    System.out.println("No customer found with the specified email.");
}
} catch (Exception e) {
    System.out.println("An error occurred while updating the password: " +
e.getMessage());
}
}
//ActionPerformed
if(valid()){
    reset();
    JOptionPane.showMessageDialog(this, "Updated");
}
else{
    JOptionPane.showMessageDialog(this, "Invalid");
}
}

```



Database connection and validation:

```

private boolean isBookAvailable(String bookId) {
    Connection connection = BACKENDCONNECTION.getConnection();
    try {
        String query = "SELECT Quantity FROM book_details WHERE book_id = ?";
        try (PreparedStatement statement = connection.prepareStatement(query))
        {
            statement.setString(1, bookId);

```

```

        try (ResultSet resultSet = statement.executeQuery()) {
            if (resultSet.next()) {
                int quantity1 = resultSet.getInt("Quantity");
                if (quantity1 <= 0) {
                    System.out.println("No books available.");
                    return false;
                }
            } else {
                System.out.println("Book not found.");
                return false;
            }
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
    return true;
}

private void insertUserKey(String bookId, String studentId, String keyValue,
String keyStatus) {
    Connection connection = BACKENDCONNECTION.getConnection();
    try {
        String query = "INSERT INTO user_key (book_id, student_id,
key_value, key_status) VALUES (?, ?, ?, ?)";
        try (PreparedStatement statement =
connection.prepareStatement(query)) {
            statement.setString(1, bookId);
            statement.setString(2, studentId);
            statement.setString(3, keyValue);
            statement.setString(4, keyStatus);
            int rowsAffected = statement.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Data inserted successfully into
user_key table.");
            } else {
                System.out.println("Failed to insert data into user_key
table.");
            }
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}

private void updateBookDetails(String bookId) {
    Connection connection = BACKENDCONNECTION.getConnection();
    try {
        String query = "UPDATE book_details SET Quantity = Quantity - 1
WHERE book_id IN (SELECT book_id FROM user_key WHERE key_status = 'pending')";

```

```

        try (PreparedStatement statement =
connection.prepareStatement(query)) {
            int rowsAffected = statement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Quantity decremented successfully in
book_details table.");
            } else {
                System.out.println("Failed to decrement Quantity in
book_details table.");
            }
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}

public void setdata() {
    Statement st = null;
    PreparedStatement ps = null;
    String today = LocalDate.now().toString();
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        st = con.createStatement();
        ResultSet resultSet = st.executeQuery("select count(*) from
book_details");
        if (resultSet.next()) {
            int count = resultSet.getInt(1);
            count_book.setText(String.valueOf(count));
        }

    } catch (SQLException e) {
    }
}

public void setBookDetailstotable() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/roy", "root", "");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from book_details");
        while (rs.next()) {
            String booktableId = rs.getString("book_id");
            String booktableName = rs.getString("book_name");
            String Author = rs.getString("Author");
            String Quantity = rs.getString("Quantity");

            Object[] obj = {booktableId, booktableName, Author, Quantity};

```

```

        model = (DefaultTableModel) tbl_book1.getModel();
        model.addRow(obj);
    }
} catch (ClassNotFoundException | SQLException e) {
}
}

public void setStudentDetailstotable() {

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/roy", "root", "");
        Statement st = con.createStatement();
        String use = t_name.getText().toLowerCase();
        System.out.println(use);
        ResultSet rs = st.executeQuery("select * from student_details
where name='user'");
        while (rs.next()) {
            String studenttableId = rs.getString("student_id");
            String studenttableName = rs.getString("name");
            String coursed = rs.getString("course");
            String Quantity = rs.getString("branch");

            Object[] obj = {studenttableId, studenttableName, coursed,
Quantity};

            model = (DefaultTableModel) tbl_student.getModel();
            model.addRow(obj);

        }
    } catch (ClassNotFoundException | SQLException e) {
    }
}

```

Only : 27 sec 4

Book

Student_id

Email

Password

id

Database connectivity and validation:

```
private boolean isBookAvailable(String bookId) {
    Connection connection = BACKENDCONNECTION.getConnection();
    try {
        String query = "SELECT Quantity FROM book_details WHERE book_id = ?";
        try (PreparedStatement statement = connection.prepareStatement(query))
        {
            statement.setString(1, bookId);
            try (ResultSet resultSet = statement.executeQuery()) {
                if (resultSet.next()) {
                    int quantity = resultSet.getInt("Quantity");
                    if (quantity <= 0) {
                        System.out.println("No books available.");
                        return false;
                    }
                } else {
                    System.out.println("Book not found.");
                    return false;
                }
            }
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}
```

```

        return true;
    }
    private void insertUserKey(String bookId, String studentId, String keyValue,
String keyStatus) {
        Connection connection =BACKENDCONNECTION.getConnection();
        try {
            String query = "INSERT INTO user_key (book_id, student_id,
key_value, key_status) VALUES (?, ?, ?, ?)";
            try (PreparedStatement statement =
connection.prepareStatement(query)) {
                statement.setString(1, bookId);
                statement.setString(2, studentId);
                statement.setString(3, keyValue);
                statement.setString(4, keyStatus);

                int rowsAffected = statement.executeUpdate();

                if (rowsAffected > 0) {
                    System.out.println("Data inserted successfully into
user_key table.");
                } else {
                    System.out.println("Failed to insert data into user_key
table.");
                }
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }

    private void updateBookDetails(String bookId) {
        Connection connection =BACKENDCONNECTION.getConnection();
        try {
            String query = "UPDATE book_details SET Quantity = Quantity - 1
WHERE book_id IN (SELECT book_id FROM user_key WHERE key_status = 'pending')";
            try (PreparedStatement statement =
connection.prepareStatement(query)) {
                int rowsAffected = statement.executeUpdate();

                if (rowsAffected > 0) {
                    System.out.println("Quantity decremented successfully in
book_details table.");
                } else {
                    System.out.println("Failed to decrement Quantity in
book_details table.");
                }
            }
        } catch (SQLException e) {

```

```

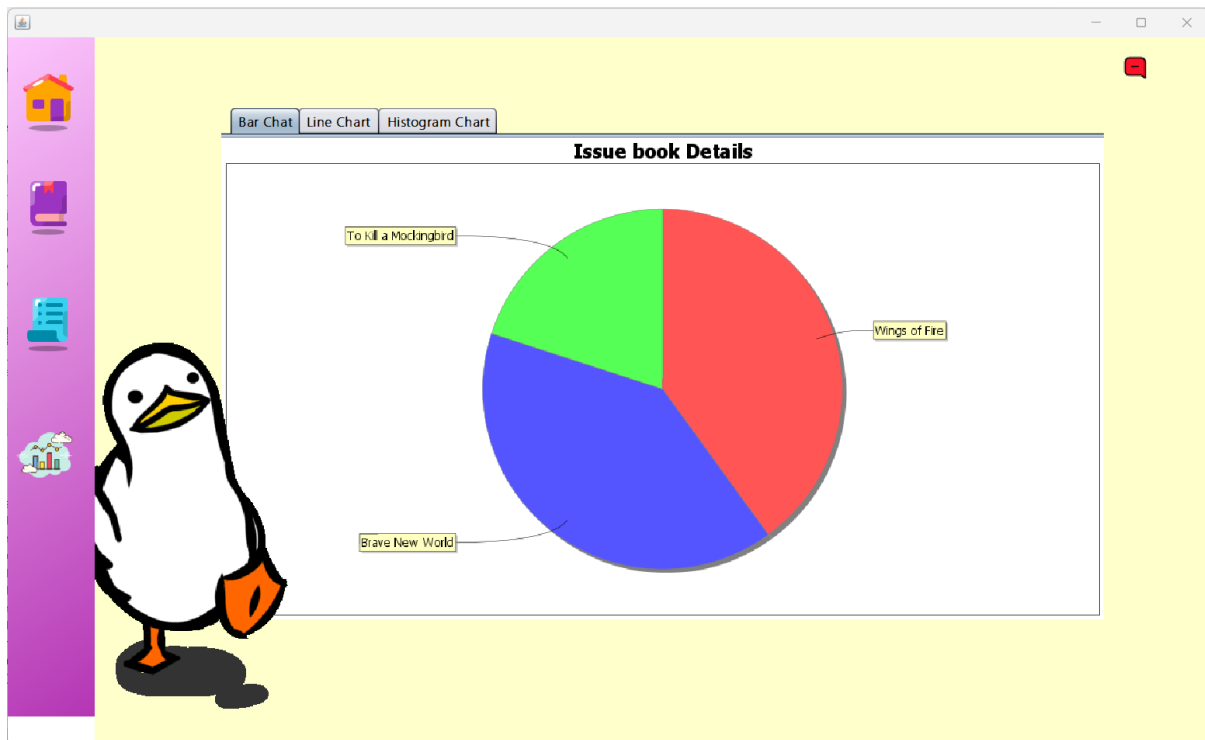
        System.out.println(e);
    }

    }

    if(i<20){
        try {
            Connection con =OG.BACKENDCONNECTION.getConnection();
            String sql ="select * from users where email=? and password=? and
status='user'";
            PreparedStatement ps =con.prepareStatement(sql);
            ps.setString(1, eml);
            ps.setString(2, pwd);
            ResultSet rs= ps.executeQuery();
            if(rs.next()){
                if (isBookAvailable(bookId)) {
                    String keyValue = generateRandomKey();
                    a.setText(keyValue);
                    insertUserKey(bookId, studentId, keyValue, keyStatus);
                    updateBookDetails(bookId);
                }
                if(i<10){
                    user l = new user();
                    l.show();
                    dispose();
                }

            }
            else{
                JOptionPane.showMessageDialog(this, "Your password/username is
wrong please check again");
            }
        } catch (HeadlessException | SQLException e) {
            System.err.println(e);
        }
    }
    else{
        user l = new user();
        l.show();
        dispose();
    }
}

```



```
public void showHistogramChart() {
    // create dataset
    HistogramDataset dataset = new HistogramDataset();
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "SELECT book_name, COUNT(*) AS book_count FROM
book_details where book_name";
        Statement st = con.createStatement();
        ResultSet resultSet = st.executeQuery(sql);
        while (resultSet.next()) {
            dataset.addSeries(resultSet.getString("book_count"), new
double[]{resultSet.getDouble("quantity")}, 10);
        }
    } catch (SQLException e) {
    }
    JFreeChart chart = ChartFactory.createHistogram("book Details", "Book
Name", "book Count", dataset, PlotOrientation.VERTICAL, false, true, false);
    chart.setBackgroundPaint(Color.white);
    ChartPanel chartPanel = new ChartPanel(chart);
    panelHistogramChart.removeAll();
    panelHistogramChart.add(chartPanel, BorderLayout.CENTER);
    panelHistogramChart.validate();
}

public void showLineChart() {
    DefaultCategoryDataset lineDataset = new DefaultCategoryDataset();
    try {
        Connection con = BACKENDCONNECTION.getConnection();
```



```

        String sql = "SELECT (*)book_details, issue_count FROM book_details
where Quantity=? ";
        Statement st = con.createStatement();
        ResultSet resultSet = st.executeQuery(sql);
        while (resultSet.next()) {
            lineDataset.setValue(resultSet.getInt("issue_count"), "Issues",
resultSet.getString("Quantity "));
        }
    } catch (SQLException e) {
    }

    JFreeChart lineChart = ChartFactory.createLineChart("Issue Book Summary",
"Month", "Issue Count", lineDataset, PlotOrientation.VERTICAL, true, true,
false);

    lineChart.getCategoryPlot().setBackgroundPaint(Color.white);
    lineChart.getCategoryPlot().setDomainGridlinePaint(Color.black);
    lineChart.getCategoryPlot().setRangeGridlinePaint(Color.black);
    JFreeChart chart =
ChartFactory.createBarChart("contribution","monthly","amount",
        dataset, PlotOrientation.VERTICAL, false,true,false);

    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //categoryPlot.setRangeGridlinePaint(Color.BLUE);
    categoryPlot.setBackgroundPaint(Color.WHITE);
    BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
    Color clr3 = new Color(204,0,51);
    renderer.setSeriesPaint(0, clr3);

    ChartPanel barpChartPanel = new ChartPanel(chart);
    panellineChart.removeAll();
    panellineChart.add(barpChartPanel, BorderLayout.CENTER);
    panellineChart.validate();
}

public void showPieChart(){

    //create dataset
    DefaultPieDataset barDataset = new DefaultPieDataset( );

    try {
        Connection con =BACKENDCONNECTION.getConnection();
        String sql ="SELECT book_name,COUNT(*) AS issue_count FROM
issue_book group by book_id";
        Statement st =con.createStatement();
        ResultSet resultSet =st.executeQuery(sql);
        while (resultSet.next()) {
            barDataset.setValue(resultSet.getString("book_name"),
Double.valueOf(resultSet.getDouble("issue_count")));

```

```

    }
    } catch (SQLException e) {
    }
    JFreeChart piechart = ChartFactory.createPieChart("Issue book
Details", barDataset, false, true, false);
    PiePlot piePlot = (PiePlot) piechart.getPlot();

    //changing pie chart blocks colors
    piePlot.setSectionPaint("a", new Color(255, 255, 102));
    piePlot.setSectionPaint("b", new Color(102, 255, 102));
    piePlot.setSectionPaint("c", new Color(255, 102, 153));
    piePlot.setSectionPaint("d", new Color(0, 204, 204));

    piePlot.setBackgroundPaint(Color.white);

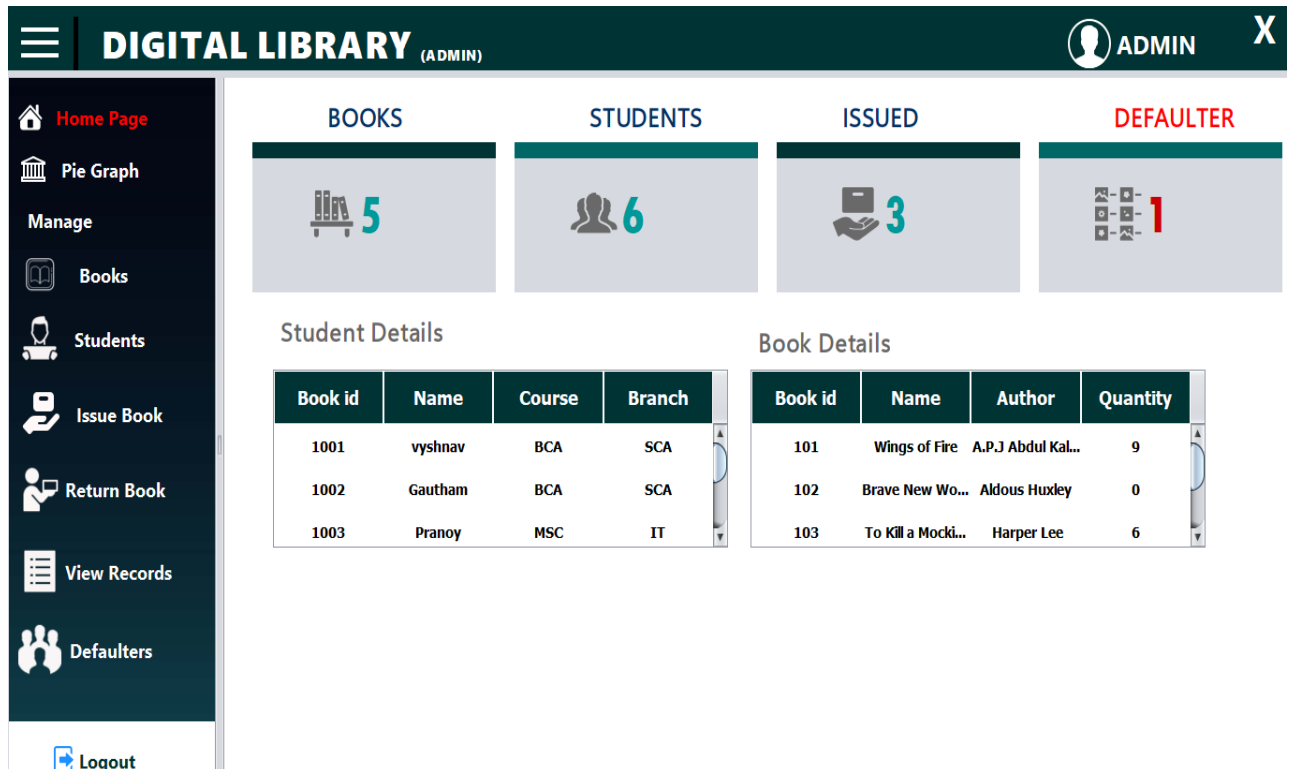
    ChartPanel barChartPanel = new ChartPanel(piechart);
    panelBarChart.removeAll();
    panelBarChart.add(barChartPanel, BorderLayout.CENTER);
    panelBarChart.validate();
}

```

[Admin](#)

[Home page](#)

The homepage for the library system consists of several components. The main focus is on the information related to books, students, issued books, and defaulters. Additionally, there are two tables displaying information about individual students and books.



```
//Calling username from login
public void y(String user) {
    t_name.setText(user);
}

//setting values to the count in homepage
public void setdata() {
    Statement st = null;
    PreparedStatement ps = null;
    String today = LocalDate.now().toString();
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        st = con.createStatement();
        ResultSet resultSet = st.executeQuery("select count(*) from
book_details");
        if (resultSet.next()) {
            int count = resultSet.getInt(1);
            count_book.setText(String.valueOf(count));
        }
    }
}
```

```

        ResultSet rz = st.executeQuery("select count(*) from
student_details");
        if (rz.next()) {
            int count = rz.getInt(1);
            count_stuent.setText(String.valueOf(count));

        }
        ResultSet rc = st.executeQuery("SELECT COUNT(*) FROM issue_book
WHERE status = 'Pending'");
        if (rc.next()) {
            int count = rc.getInt(1);
            count_issued.setText(String.valueOf(count));
        }

        PreparedStatement phs = con.prepareStatement("SELECT COUNT(*) FROM
issue_book WHERE due_date < ? AND status = ?");
        phs.setString(1, today);
        phs.setString(2, "pending");
        ResultSet ras = phs.executeQuery();
        if (ras.next()) {
            int counts = ras.getInt(1);
            count_defulter.setText(String.valueOf(counts));
        }

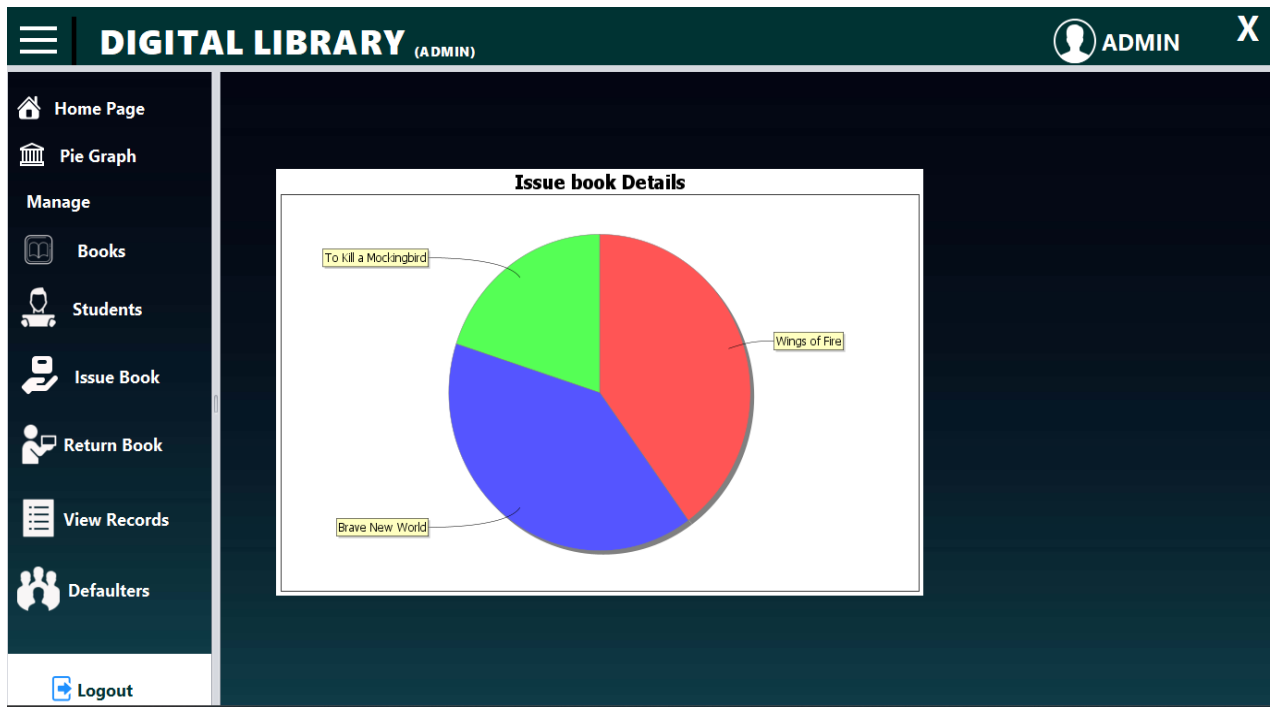
    } catch (SQLException e) {

    }
}

```

Pie-graph

A pie graph depicting the "Issue Book" data provides a visual representation of the distribution of books issued from the library. This graph showcases the proportion of issued books based on specific titles.



```
public void showPieChart() {

    //create dataset
    DefaultPieDataset barDataset = new DefaultPieDataset();

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "SELECT book_name,COUNT(*) AS issue_count FROM issue_book
group by book_id";
        Statement st = con.createStatement();
        ResultSet resultSet = st.executeQuery(sql);
        while (resultSet.next()) {
            barDataset.setValue(resultSet.getString("book_name"),
Double.valueOf(resultSet.getDouble("issue_count")));
        }
    } catch (SQLException e) {
    }

    //create charts
    JFreeChart piechart = ChartFactory.createPieChart("Issue book Details",
barDataset, false, true, false);

    PiePlot piePlot = (PiePlot) piechart.getPlot();

    //changing pie chart blocks colors
    piePlot.setSectionPaint("a", new Color(255, 255, 102));
    piePlot.setSectionPaint("b", new Color(102, 255, 102));
    piePlot.setSectionPaint("c", new Color(255, 102, 153));
    piePlot.setSectionPaint("d", new Color(0, 204, 204));
}
```

```
piePlot.setBackgroundPaint(Color.white);

//create chartPanel to display chart(graph)
ChartPanel pieChartPanel = new ChartPanel(piechart);
ChartPanel barChartPanel = new ChartPanel(piechart);
Admin_piechart1.removeAll();
Admin_piechart1.add(barChartPanel, BorderLayout.CENTER);
Admin_piechart1.validate();
}
```

Books

This section provides information about the library's book collection. It includes details such as book titles, authors, genres, and availability status. Admin can search for specific books or browse through different categories and change the details.

DIGITAL LIBRARY

(ADMIN)

ADMIN

X

Home Page

Pie Graph

Manage

Books

Students

Issue Book

Return Book

View Records

Defaulters

Logout

Enter Book ID

101

Enter Book Name

Wings of Fire

Author Name

A.P.J Abdul Kalam

Quantity

9

UPDATE

ADD

DELETE

Manage Books

Back

| Book id | Name | Author | Quantity |
|---------|-----------------------|-------------------|----------|
| 101 | Wings of Fire | A.P.J Abdul Kalam | 9 |
| 102 | Brave New World | Aldous Huxley | 0 |
| 103 | To Kill a Mockingbird | Harper Lee | 6 |
| 104 | Wings of Fire | A.P.J Abdul Kalam | 10 |
| 105 | d | d | 0 |

```
public boolean addbook() {
    boolean isadded = false;
    if (txt_bookid.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter a book ID!");
        return false;
    }
    try {
        bookId = Integer.parseInt(txt_bookid.getText());
    } catch (NumberFormatException e) {
```

```

        JOptionPane.showMessageDialog(null, "Please enter a valid book ID!");
        return false;
    }

    // Validate bookName
    if (txt_bookname.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter a book name!");
        return false;
    }
    bookName = txt_bookname.getText();

    // Validate author
    if (txt_authorname.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter an author name!");
        return false;
    }
    author = txt_authorname.getText();

    // Validate quantity
    if (txt_quantity.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter a quantity!");
        return false;
    }
    try {
        quantity = Integer.parseInt(txt_quantity.getText());
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Please enter a valid quantity!");
        return false;
    }

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "insert into book_details values(?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookId);
        ps.setString(2, bookName);
        ps.setString(3, author);
        ps.setInt(4, quantity);
        int rowCount = ps.executeUpdate();
        isadded = rowCount > 0;
    } catch (SQLException e) {
    }
    return isadded;
}

//clear table
public void clearTable() {

```

```

        DefaultTableModel tablemodel = (DefaultTableModel) tbl_book2.getModel();
        tablemodel.setRowCount(0);
    }

    public boolean updatebook() {
        boolean isupdated = false;
        bookId = Integer.parseInt(txt_bookid.getText());
        bookName = txt_bookname.getText();
        author = txt_authorname.getText();
        quantity = Integer.parseInt(txt_quantity.getText());
        try {
            Connection con = BACKENDCONNECTION.getConnection();
            String sql = "update book_details set book_name = ?,Author =
?,Quantity = ? where book_id = ?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(4, bookId);
            ps.setString(1, bookName);
            ps.setString(2, author);
            ps.setInt(3, quantity);
            int rowCount = ps.executeUpdate();
            isupdated = rowCount > 0;
        } catch (SQLException e) {
        }
        return isupdated;
    }

    public boolean Deletebook() {
        boolean isDeleted = false;
        bookId = Integer.parseInt(txt_bookid.getText());
        try {
            Connection con = BACKENDCONNECTION.getConnection();
            String sql = "Delete from book_details where book_id = ?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, bookId);
            int rowCount = ps.executeUpdate();
            isDeleted = rowCount > 0;
        } catch (SQLException e) {
        }
        return isDeleted;
    }
}

```

Add Students

The student section contains a database of registered library users. Each student entry includes essential details like student names, identification numbers, and

contact information. This information helps in managing student accounts and tracking their activities within the library system.

Home Page

Pie Graph

Manage

Books

Students

Issue Book

Return Book

View Records

Defaulters

Logout

DIGITAL LIBRARY

(ADMIN)

ADMIN

Enter Book ID

1001

Enter Student Name

vyshnav

Select Course

BCA

Select Branch

SCA

ADD

UPDATE

DELETE

Manage Students

Back

| student id | Name | course | Branch |
|------------|---------|--------|--------|
| 1001 | vyshnav | BCA | SCA |
| 1002 | Gautham | BCA | SCA |
| 1003 | Pranoy | MSC | IT |
| 1004 | Roshit | MSC | SCA |
| 1005 | fghj | MSC | SCA |
| 24355 | dfdfhgh | BCA | CSE |

```
public void setStudentDetailstotable() {

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/roy", "root", "");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from student_details");
        while (rs.next()) {
            String studenttableId = rs.getString("student_id");
            String studenttableName = rs.getString("name");
            String coursed = rs.getString("course");
            String Quantity = rs.getString("branch");

            Object[] obj = {studenttableId, studenttableName, coursed,
Quantity};
            model = (DefaultTableModel) tbl_student.getModel();
            model.addRow(obj);
            model = (DefaultTableModel) tbl_student1.getModel();
            model.addRow(obj);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

    }
    } catch (ClassNotFoundException | SQLException e) {
    }
}

public boolean addstudent() {
    boolean isadded = false;

    try {
        String studentName = txt_stuentname.getText();
        if (!studentName.matches("^[a-zA-Z\\s]+$")) {
            JOptionPane.showMessageDialog(null, "Invalid Student Name!");
            return false;
        }

        String course = combo_course.getSelectedItem().toString();
        if (course.equals("")) {
            JOptionPane.showMessageDialog(null, "Please select a Course!");
            return false;
        }

        String branch = combo_branch.getSelectedItem().toString();
        if (branch.equals("")) {
            JOptionPane.showMessageDialog(null, "Please select a Branch!");
            return false;
        }

        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "insert into student_details values(?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, studentID);
        ps.setString(2, studentName);
        ps.setString(3, course);
        ps.setString(4, branch);
        int rowCount = ps.executeUpdate();
        isadded = rowCount > 0;
    } catch (SQLException e) {
    }
    return isadded;
}

//clear table
public void clearstudentTable() {
    DefaultTableModel tablemodel = (DefaultTableModel) tbl_student.getModel();
    tablemodel.setRowCount(0);
}

```

```

public boolean updatestudent() {
    boolean isupdated = false;
    studentID = Integer.parseInt(txt_studentid.getText());
    studentName = txt_stuentname.getText();
    course = combo_course.getSelectedItem().toString();
    branch = combo_branch.getSelectedItem().toString();
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "update student_details set name = ?,course = ?,branch =
? where student_id = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(4, studentID);
        ps.setString(1, studentName);
        ps.setString(2, course);
        ps.setString(3, branch);
        int rowCount = ps.executeUpdate();
        isupdated = rowCount > 0;
    } catch (SQLException e) {
    }
    return isupdated;
}

public boolean Deletestudent() {
    boolean isDeleted = false;
    studentID = Integer.parseInt(txt_studentid.getText());
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "Delete from student_details where student_id = ?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, studentID);
        int rowCount = ps.executeUpdate();
        isDeleted = rowCount > 0;
    } catch (SQLException e) {
    }
    return isDeleted;
}

```

Issue Books

This section displays information about books that have been borrowed by students. It includes details about the student who borrowed the book, the due date for returning the book, and any relevant transaction history. It helps both librarians and students keep track of borrowed materials and ensures timely returns.

DIGITAL LIBRARY (ADMIN)

ADMIN

Home Page
 Pie Graph
 Manage
 Books
 Students
 Issue Book
 Return Book
 View Records
 Defaulters
 Logout

Student Details

student id: 1002

Student Name : Gautham

Course : BCA

Branch : SCA

Issue Book

Book Id : 101

Student Id : 1002

Issue Date :

Due Date :

ISSUE BOOK

Book Details

Book id: 101

Book Name : Wings of Fire

Author : A.P.J Abdul Kalam

Quantity : 9

```

public void getbookdetails() {
    int bookId = Integer.parseInt(issue_bookid.getText());

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "select * from book_details where book_id= ? ";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookId);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            lbl_bookid.setText(rs.getString("book_id"));
            lbl_bookname.setText(rs.getString("book_name"));
            lbl_author.setText(rs.getString("Author"));
            lbl_quantity.setText(rs.getString("Quantity"));
            lbl_errorbook.setText("");
        } else {
            lbl_errorbook.setText("invalid book id");
        }
    } catch (SQLException e) {
    }
}

public void getstudentdetails() {
    int studentId = Integer.parseInt(issue_studentid.getText());
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "select * from student_details where student_id= ? ";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, studentId);
        ResultSet rs = ps.executeQuery();
    }
}

```

```

        if (rs.next()) {
            lbl_studentid.setText(rs.getString("student_id"));
            lbl_studentname.setText(rs.getString("name"));
            lbl_course.setText(rs.getString("course"));
            lbl_branch.setText(rs.getString("branch"));
            lbl_errorstudent.setText("");

        } else {
            lbl_errorstudent.setText("invalid student id");
        }
    } catch (SQLException e) {
    }
}

//insert book details

public boolean issuebook() {
    boolean isinserted = false;
    int bookIda = Integer.parseInt(issue_bookid.getText());
    int studentida = Integer.parseInt(issue_studentid.getText());
    String book_name = lbl_bookname.getText();
    String student_name = lbl_studentname.getText();
    Date uIssueDate = date_issuedate.getDate();
    Date udueDate = date_duedate.getDate();
    long l1 = uIssueDate.getTime();
    long l2 = udueDate.getTime();
    java.sql.Date sissueDate = new java.sql.Date(l1);
    java.sql.Date sudueDate = new java.sql.Date(l2);

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "insert into issue_book (book_id,book_name,student_id,"
            + "name,issue_date,due_date,status)values(?,?,?,?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookIda);
        ps.setString(2, book_name);
        ps.setInt(3, studentida);
        ps.setString(4, student_name);
        ps.setDate(5, sissueDate);
        ps.setDate(6, sudueDate);
        ps.setString(7, "pending");
        int rc = ps.executeUpdate();
        isinserted = rc > 0;
    } catch (SQLException e) {
    }
    return isinserted;
}

```

```

//update book count
public void updatecount() {

    int bookIda = Integer.parseInt(issue_bookid.getText());
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "update book_details set Quantity = Quantity - 1 where
book_id =?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookIda);
        int rc = ps.executeUpdate();
        if (rc > 0) {
            JOptionPane.showMessageDialog(this, "updated");
            int initailcount = Integer.parseInt(lbl_quantity.getText());
            lbl_quantity.setText(Integer.toString(initailcount - 1));
        } else {
            JOptionPane.showMessageDialog(this, "impossible");
        }
    } catch (HeadlessException | NumberFormatException | SQLException e) {
    }
}

//checking wether already allocated or not
public boolean isAlreadyissued() {
    boolean isAlreadyissued = false;
    int bookIda = Integer.parseInt(issue_bookid.getText());
    int studentid = Integer.parseInt(issue_studentid.getText());

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "select * from issue_book where book_id =? and
student_id=? and status=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookIda);
        ps.setInt(2, studentid);
        ps.setString(3, "pending");
        ResultSet rc = ps.executeQuery();
        isAlreadyissued = rc.next();

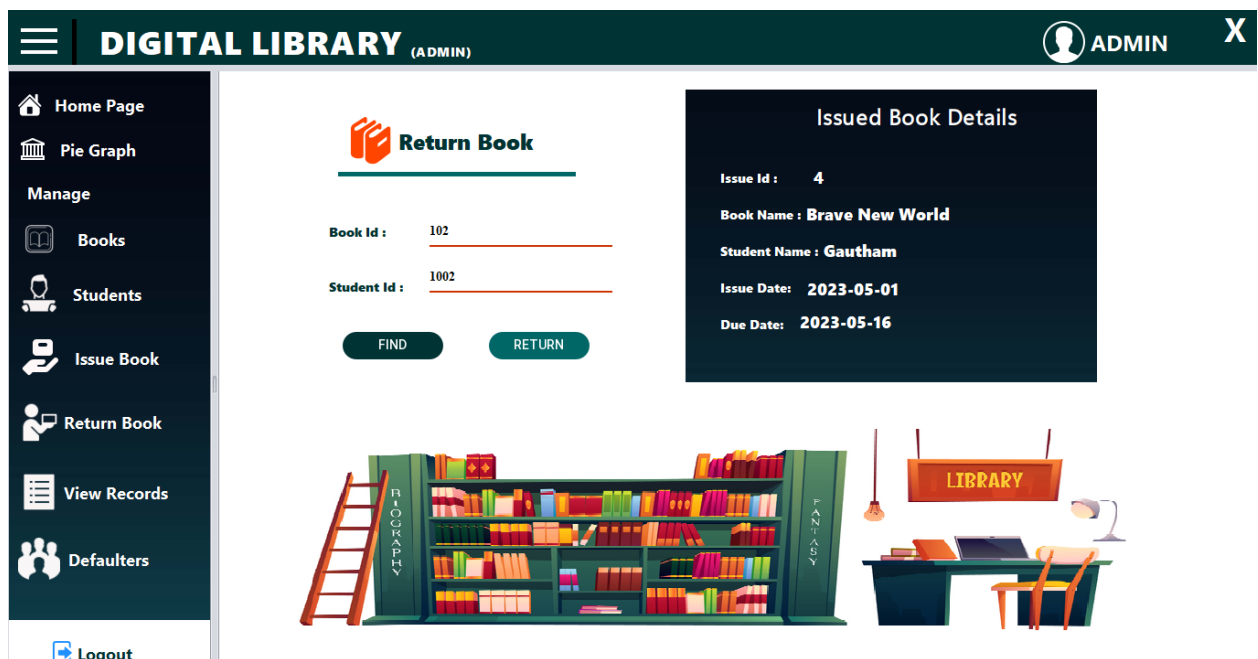
    } catch (SQLException e) {
    }

    return isAlreadyissued;
}

```

Return Books

The "Return Book" module in a library application is a feature that enables librarians to manage and track the return of borrowed books by students. The module typically has a user-friendly interface where librarians can interact with the system to process book returns. It may include fields or search options to help librarians locate the specific book being returned. Once the book is identified, the system updates its status to reflect that it has been returned. This may involve changing the availability status from "pending" to "returned" in the library's database or system.



```
public void Returndetails() {  
  
    int bookissueid = Integer.parseInt(return_bookid.getText());  
    int studentissueid = Integer.parseInt(return_student_id.getText());  
    try {  
        Connection con = BACKENDCONNECTION.getConnection();  
        String sql = "select * from issue_book where book_id = ? and  
student_id =? and status =? ";  
        PreparedStatement ps = con.prepareStatement(sql);  
        ps.setInt(1, bookissueid);  
        ps.setInt(2, studentissueid);  
        ps.setString(3, "pending");  
        ResultSet rs = ps.executeQuery();  
        if (rs.next()) {
```

```

        issued_id.setText(rs.getString("id"));
        issued_book_name.setText(rs.getString("book_name"));
        issued_student_name.setText(rs.getString("name"));
        issued_date.setText(rs.getString("issue_date"));
        issued_due_date.setText(rs.getString("due_date"));
        issued_errors.setText("");
    } else {
        issued_errors.setText("");
        issued_id.setText("");
        issued_book_name.setText("");
        issued_student_name.setText("");
        issued_date.setText("");
        issued_due_date.setText("");
        issued_errors.setText("No Record Found");
    }
} catch (SQLException e) {
}
}

// to return book
public boolean Returnbook() {

    boolean isreturned = false;
    int bookissueid = Integer.parseInt(return_bookid.getText());
    int studentissueid = Integer.parseInt(return_student_id.getText());

    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "update issue_book set status=? where book_id=? and student_id=? and status=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, "returned");
        ps.setInt(2, bookissueid);
        ps.setInt(3, studentissueid);
        ps.setString(4, "pending");
        int rc = ps.executeUpdate();
        isreturned = rc > 0;
    } catch (SQLException e) {
    }
    return isreturned;
}

public void updatereturnCount() {

    int bookissueid = Integer.parseInt(return_bookid.getText());
    try {

```



```

        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "update book_details set Quantity = Quantity + 1 where
book_id =?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, bookissueid);
        ps.executeUpdate();

    } catch (HeadlessException | NumberFormatException | SQLException e) {
    }
}

```

View Records

The librarian module serves as a comprehensive tool for librarians to effectively manage library resources, track book circulation, handle student records, and ensure smooth operations within the library system.

DIGITAL LIBRARY (ADMIN)

ADMIN

X

Home Page

Pie Graph

Manage

Books

Students

Issue Book

Return Book

View Records

Defaulters

Logout

VIEW RECORDS

SEARCH

From Date : May 3, 2023

To Date : May 3, 2023

ALL

SEARCH

| ID | BOOK | STUDENT | ISSUE DATE | DUE DATE | STATUS |
|----|---------------------|----------|------------|------------|----------|
| 1 | Wings of Fire | user | 2023-05-02 | 2023-05-09 | returned |
| 2 | Brave New Wo... | user | 2023-05-02 | 2023-05-09 | returned |
| 3 | To Kill a Mockin... | user | 2023-05-02 | 2023-05-03 | pending |
| 4 | Brave New Wo... | Gautham | 2023-05-01 | 2023-05-16 | pending |
| 5 | Wings of Fire | Gauthamc | 2023-05-02 | 2023-05-16 | pending |

```
public void GetValueFromDatabase() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/roy", "root", "");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from issue_book");
        while (rs.next()) {
            String tablebookid = rs.getString("id");
            String tablebookname = rs.getString("book_name");
            String tablestudentname = rs.getString("name");
        }
    }
}
```

```

        String tableissuedate = rs.getString("issue_date");
        String tablwduedate = rs.getString("due_date");
        String tablestatus = rs.getString("status");

        Object[] obj = {tablebookid, tablebookname, tablestudentname,
tableissuedate, tablwduedate, tablestatus};
        model = (DefaultTableModel) t.getModel();
        model.addRow(obj);
    }
} catch (ClassNotFoundException | SQLException e) {
}
}

public void clearviewTable() {
    DefaultTableModel tablemodel = (DefaultTableModel) t.getModel();
    tablemodel.setRowCount(0);
}

//to fetch the record by date

public void datesearch() {

    Date uFromdate = date_from_view.getDate();
    Date uTodate = date_to_view.getDate();
    long l1 = uFromdate.getTime();
    long l2 = uTodate.getTime();
    java.sql.Date Fromdate = new java.sql.Date(l1);
    java.sql.Date Todate = new java.sql.Date(l2);
    try {
        Connection con = BACKENDCONNECTION.getConnection();
        String sql = "select * from issue_book where issue_date BETWEEN ? and
?";

        PreparedStatement ps = con.prepareStatement(sql);
        ps.setDate(1, Fromdate);
        ps.setDate(2, Todate);

        ResultSet rs = ps.executeQuery();

        if (rs.next() == false) {

            JOptionPane.showMessageDialog(this, "no record found");
        } else {
            while (rs.next()) {
                String tablebookid = rs.getString("id");
                String tablebookname = rs.getString("book_name");
                String tablestudentname = rs.getString("name");
                String tableissuedate = rs.getString("issue_date");
                String tablwduedate = rs.getString("due_date");
            }
        }
    }
}

```

```

        String tablestatus = rs.getString("status");

        Object[] obj = {tablebookid, tablebookname, tablestudentname,
tableissuedate, tablwduedate, tablestatus};
        model = (DefaultTableModel) t.getModel();
        model.addRow(obj);
    }
}

} catch (SQLException e) {

}

}

```

Defaulters list

The defaulters section lists students who have not returned their borrowed books by the due date. It highlights the names of the defaulting students, the books they have yet to return, and any applicable fines or penalties. This feature aids librarians in identifying and managing overdue book returns.

DIGITAL LIBRARY (ADMIN)
 ADMIN

Home Page
 Pie Graph

Manage

Books
 Students
 Issue Book
 Return Book
 View Records
 Defaulters

Logout

DEFAULTERS LIST

| ID | BOOK | STUDENT | ISSUE DATE | DUE DATE | STATUS |
|----|-----------------------|---------|------------|------------|---------|
| 3 | To Kill a Mockingbird | user | 2023-05-02 | 2023-05-03 | pending |

```
public void Defaultterlist() {
    long l = System.currentTimeMillis();
    java.sql.Date todaysdate = new java.sql.Date(l);
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/roy", "root", "");
        PreparedStatement ps = con.prepareStatement("select * from issue_book
where due_date < ? and status = ? ");
        ps.setDate(1, (java.sql.Date) todaysdate);
        ps.setString(2, "pending");
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            String tablebookid = rs.getString("id");
            String tablebookname = rs.getString("book_name");
            String tablestudentname = rs.getString("name");
            String tableissuedate = rs.getString("issue_date");
            String tablwduedate = rs.getString("due_date");
            String tablestatus = rs.getString("status");

            Object[] obj = {tablebookid, tablebookname, tablestudentname,
tableissuedate, tablwduedate, tablestatus};
            model = (DefaultTableModel) Dl.getModel();
            model.addRow(obj);
        }
    } catch (ClassNotFoundException | SQLException e) {
    }
}
```

LIBRARIAN

☰
POOJA
X

Home Page

Pie Graph

Students

Issue Book

Return Book

Defaulters

DIGITAL LIBRARY (LIBRARIAN)

BOOKS

5

STUDENTS

6

ISSUED

3

DEFAULTER

1

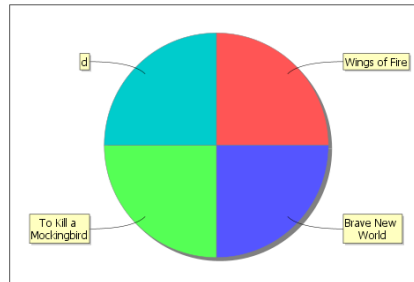
Student Details

| Book id | Name | Course | Branch |
|---------|----------|--------|--------|
| 1001 | vysnav | BCA | SCA |
| 1002 | Gauthamc | BCA | SCA |
| 1003 | Pranoy | MSC | IT |

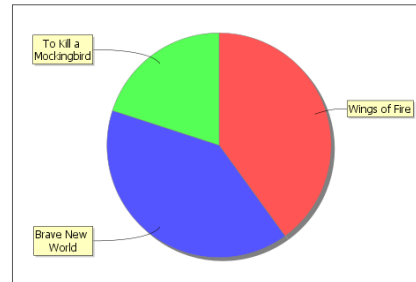
Book Details

| Book id | Name | Author | Quantity |
|---------|--------------------|--------------------|----------|
| 101 | Wings of Fire | A.P.J Abdul Kal... | 9 |
| 102 | Brave New Wo... | Aldous Huxley | 0 |
| 103 | To Kill a Mocki... | Harper Lee | 6 |

Book Details



Issue book Details



Manage Students

[Back](#)

| student id | Name | course | Branch |
|------------|----------|--------|--------|
| 1001 | vyshnav | BCA | SCA |
| 1002 | Gauthamc | BCA | SCA |
| 1003 | Pranoy | MSC | IT |
| 1004 | Roshit | MSC | SCA |
| 1005 | fghj | MSC | SCA |
| 24355 | dfdhfgh | BCA | CSE |

DIGITAL LIBRARY

(LIBRARIAN)

POOJA

X

Home Page

Pie Graph

Students

Issue Book

Return Book

Defaulters

Logout

Return Book

Book Id :

Enter Book Id

Student Id :

Enter Student Id

FIND

RETURN

Issued Book Details

Issue Id :

Book Name :

Student Name :

Issue Date:

Due Date:

DIGITAL LIBRARY

(LIBRARIAN)

POOJA

X

Home Page

Pie Graph

Students

Issue Book

Return Book

Defaulters

Logout

Student Details

student id: 1002

Student Name : Gauthamc

Course : BCA

Branch : SCA

Issue Book

Book Id :

102

Student Id :

1002

Issue Date :

May 9, 2023

Due Date :

May 6, 2023

ISSUEBOOK


Book Details

Book id: 102

Book Name : Brave New World

Author : Aldous Huxley

Quantity : 0



Thank you
for upgrading to Digital

Sign Up


| | | | |
|----------------------|----------------------|------------------------|--------------------------|
| First Name* | <input type="text"/> | DOB* | <input type="text"/> |
| Last Name* | <input type="text"/> | Password* | <input type="password"/> |
| Phone* | <input type="text"/> | Tell us about yourself | <input type="text"/> |
| Current Address | <input type="text"/> | | |
| Residential Address* | <input type="text"/> | | |
| | Email* | <input type="text"/> | |
| | Captcha* | <input type="text"/> | |

NOTE
Press NEXT and ask for KEY


GENERATE KEY

Back Clear Submit

28°C
Mostly cloudy



ENG
IN 22:47
11-05-2023



```
-->You:      key
-->Rohit    MJIKBOHD
```


Forgot Password

Email ID

New Password

Retype

☐ View Password

DIGITAL LIBRARY

(USER)

USER

X

Home Page

Graph

Pre Register

View Records

Logout

Book Details

| Book id | Name | Author | Quantity |
|---------|-----------------------|-------------------|----------|
| 101 | Wings of Fire | A.P.J Abdul Kalam | 4 |
| 102 | Brave New World | Aldous Huxley | 0 |
| 103 | To Kill a Mockingbird | Harper Lee | -1 |
| 104 | Wings of Fire | A.P.J Abdul Kalam | 7 |
| 105 | d | d | 0 |

BOOKS

5

Only : 27 sec 4

Book

Student_id

Email

Password

id

