

BOOK STORE MANAGEMENT SYSTEM
A MINI PROJECT REPORT

Submitted by

JANANI KUMARESH	230701122
HARINI M	230701101

In partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023 - 24

BONAFIDE CERTIFICATE

Certified that this project report **“BOOK STORE MANAGEMENT SYSTEM”** is
the bonafide work of **“JANANI KUMARESH (230701122), HARINI M (230701101)”**

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs.Divya.M
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam,Chennai 602105

INTERNAL EXAMINER

SIGNATURE

Mrs.Raghu.G
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam,Chennai 602105

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

ABSTRACT:

A Book Store Management System is a digital solution designed to streamline the operations of a bookstore. It facilitates the management of inventory, sales, customer data, and other essential business processes, offering both online and offline interfaces for enhanced usability. This system allows bookstore staff to efficiently track books in stock, monitor sales, and manage supplier information, reducing the risk of overstock or stockouts. It also provides customer relationship management features, including customer purchase history and preferences, to support personalized service and targeted marketing efforts.

The system typically includes modules for inventory management, billing and invoicing, customer management, supplier management, and reporting. By automating tasks such as updating stock levels, processing transactions, and generating financial reports, the Book Store Management System increases productivity and minimizes human errors. For online bookstores, it often integrates with e-commerce platforms, enabling seamless online ordering and payment processing. Additionally, real-time analytics allow for data-driven decisions, which can improve profitability and customer satisfaction. Overall, a Book Store Management System is essential for modern bookstores, supporting them in becoming more efficient, customer-centric, and competitive in the digital age.

A Book Store Management System is a comprehensive software solution aimed at simplifying the day-to-day operations of a bookstore. This system automates key functions, including inventory control, customer data management, sales tracking, and supplier management. By centralizing information, it provides staff with quick access to book availability, customer preferences, and supplier details, resulting in improved customer service and operational efficiency. The system's analytical tools help store owners make informed decisions regarding inventory and marketing strategies, reducing manual errors and operational costs. This solution is an essential tool for bookstores looking to streamline operations and enhance customer experience in a competitive market.

TABLE OF CONTENTS

Chapter 1

1 INTRODUCTION

- 1.1 INTRODUCTION -----
- 1.2 OBJECTIVES -----
- 1.3 MODULES -----

Chapter 2

2 SURVEY OF TECHNOLOGIES

- 2.1 SOFTWARE DESCRIPTION -----
- 2.2 LANGUAGES -----
 - 2.2.1 JAVA SWING -----
 - 2.2.2 MYSQL-----
 - 2.2.3 JDBC-----
 - 2.2.4 APACHE NETBEANS-----

Chapter 3

3 REQUIREMENTS AND ANALYSIS

- 3.1 REQUIREMENT SPECIFICATION -----
- 3.1.1 FUNCTIONAL REQUIREMENTS----- 3.1.2
- NON FUNCTIONAL REQUIREMENTS----- 3.2
- HARDWARE AND SOFTWARE REQUIREMENTS ----- 3.3
- ARCHITECTURE DIAGRAM ----- 3.4 ER
- DIAGRAM ----- 3.5
- NORMALIZATION

Chapter 4

4 PROGRAM CODE

- 4.1 PROGRAM CODE-----

Chapter 5

5 RESULTS AND DISCUSSION

- 5.1 RESULTS AND DISCUSSION -----

Chapter 6

6 CONCLUSION _____

6.1 CONCLUSION-----

Chapter 7

7 REFERENCES _____

7.1 REFERENCES-----

1.1 INTRODUCTION

A Book Store Management System is an integrated software solution designed to streamline and automate various aspects of bookstore operations. In the traditional setting, bookstores face numerous challenges, such as managing extensive inventories, keeping track of customer purchases, handling sales transactions, managing suppliers, and ensuring that popular titles are always in stock. Manually handling these tasks can lead to inefficiencies, increased operational costs, and customer dissatisfaction due to stock shortages or slow service.

With a Book Store Management System, bookstores can efficiently manage their day-to-day activities while minimizing errors and reducing manual workload. This system provides functionalities for inventory management, sales processing, billing, customer and supplier management, and reporting. Additionally, many systems come with advanced features such as real-time stock updates, automated reorder alerts, and data analytics that enable bookstore owners to make data-driven decisions and optimize their business.

The system's inventory management module allows bookstore staff to quickly check stock levels, organize books by categories, and track book availability across different sections of the store. Sales and billing modules simplify the checkout process, supporting faster transactions and accurate records. Customer relationship management (CRM) features help bookstores track customer preferences, purchase history, and contact details, enabling them to offer personalized recommendations and promotions.

For bookstores operating in the digital age, a Book Store Management System often includes integration with e-commerce platforms. This feature allows bookstores to manage both in-store and online sales, reach a wider customer base, and handle payments seamlessly. Additionally, data analytics tools built into the system offer valuable insights into sales trends, high-demand genres, and seasonal customer preferences, which can help bookstore managers refine their inventory and marketing strategies.

In summary, a Book Store Management System is essential for modern bookstores aiming to improve efficiency, enhance customer satisfaction, and remain competitive. By automating repetitive tasks, ensuring accurate data handling, and providing valuable business insights, this system is a vital asset for bookstores of all sizes.

1.2 OBJECTIVES

- **Provide Data-Driven Insights:** Generate reports on sales trends and customer behavior to inform inventory and marketing decisions.
- **Streamline Inventory and Stock Management:** Automate tracking of books in stock, preventing shortages and overstock by managing inventory efficiently.
- **Simplify Sales and Billing:** Speed up transactions and reduce errors with integrated point-of-sale (POS) and billing features for a smoother checkout process.
- **Enhance Customer Relationship Management (CRM):** Track customer preferences and purchase history for personalized recommendations and loyalty programs.

1.3 MODULES

1. Inventory Management Module

Tracks and manages book inventory, including titles, authors, genres, quantities, and pricing. It automates stock updates, reorders, categorization, and alerts for low stock levels. **2. Sales and Billing Module**

Handles sales transactions, billing, invoicing, and receipts. It includes a point-of-sale (POS) system for quick checkouts and accurate record-keeping of all sales. **3. Customer Relationship Management (CRM) Module** Manages customer data, including personal details, purchase history, preferences, and loyalty programs. This module supports personalized recommendations and targeted promotions.

4. Supplier and Order Management Module

Manages supplier information, purchase orders, and deliveries. It automates ordering processes, tracks supplier relationships, and ensures timely restocking of books.

5. Reporting and Analytics Module

Provides detailed reports and analytics on sales trends, inventory status, customer behavior, and financial performance, supporting data-driven decision-making.

6. User Management and Security Module

Controls access to the system based on user roles, ensuring data security. It allows admins to set permissions, monitor usage, and protect sensitive data.

7. Online Sales and E-commerce Integration Module

For bookstores with online presence, this module integrates with e-commerce platforms, managing online orders, payments, and inventory synchronization between in-store and online channels.

8. Finance and Accounting Module

Tracks sales, revenue, expenses, and profits. This module assists with bookkeeping, financial analysis, and report generation for financial management.

9. Marketing and Promotions Module

Manages promotions, discounts, and campaigns, tracking their effectiveness and reaching targeted customers with offers and loyalty rewards.

These modules work together to provide a comprehensive solution for efficient bookstore management.

1. Java Swing as Frontend

Rich User Interface Components: Java Swing provides a comprehensive set of GUI components such as buttons, labels, tables, and more, allowing developers to create robust desktop applications.

Customizable Look and Feel: Swing supports a pluggable look-and-feel architecture, enabling customization of the UI to match specific design requirements.

Event Handling: Offers a well-defined event-handling mechanism, making it easy to handle user actions like clicks, selections, and inputs.

Lightweight Components: Swing components are lightweight compared to their predecessors (AWT), which ensures better performance and flexibility in the UI design.

2. MySQL as Backend

Relational Database Management: MySQL is an open-source RDBMS that efficiently handles large amounts of structured data and supports SQL (Structured Query Language) for database operations.

Scalability and Reliability: It is suitable for small to large-scale applications due to its scalability and consistent performance.

Strong Security Features: MySQL includes data encryption, user authentication, and privilege management, making it a secure choice for data storage.

Cross-Platform Compatibility: MySQL runs on various operating systems like Windows, macOS, and Linux, providing flexibility in deployment.

3.)Apache NetBeans

Comprehensive Development Environment: NetBeans is an open-source IDE that supports multiple programming languages, with strong tools for Java development such as syntax highlighting, code completion, and debugging.

Integrated GUI Builder: The IDE comes with a drag-and-drop GUI builder for designing Java Swing interfaces easily and intuitively.

Project Management: Supports Maven and Gradle for project management, which is beneficial for handling dependencies and project structure in complex applications.

Built-in Tools for Database Interaction: NetBeans has built-in support for connecting to databases, making it easier to preview and modify database tables directly from the IDE.

4. JDBC Connectivity for DBMS Integration Standard API for Java: JDBC (Java Database Connectivity) is a standard Java API for connecting and executing queries with databases, providing a common interface for different DBMS. **Establishing Connection:** JDBC enables establishing a connection with databases through drivers, allowing seamless communication between Java applications and MySQL databases. **Data Retrieval and Manipulation:** Supports executing SQL queries, retrieving results, and performing database operations like insert, update, and delete from Java programs. **Exception Handling:** JDBC provides structured exception handling to manage database connectivity errors, ensuring more robust and error-resistant code.

Chapter 3

REQUIREMENTS AND ANALYSIS

Functional Requirements

Java Swing (Frontend)

User Interaction: The application should provide a user-friendly interface that allows users to interact through buttons, forms, and other UI components.

Dynamic Data Display: The application should display data dynamically from the database in components like tables and labels.

Event Handling: The application must handle user inputs effectively, triggering appropriate events such as button clicks or text input processing.

Navigation: Provide clear navigation elements for moving between different windows or panels within the application.

MySQL (Backend)

Data Storage: The backend must support the storage and retrieval of structured data for the application.

CRUD Operations: The database should allow Create, Read, Update, and Delete operations to be performed by the application.

Data Integrity: Ensure the database maintains data accuracy and consistency, especially during concurrent user operations.

Secure Data Access: Implement user authentication and authorization mechanisms to manage access to sensitive data.

Apache NetBeans

IDE Features: Use the drag-and-drop GUI builder to streamline the development process of Java Swing interfaces.

Debugging and Testing: Utilize built-in debugging tools for testing the application and ensuring it works as expected.

Version Control: Integrate version control systems like Git for source code management directly within the IDE.

Database Tools: Leverage the built-in database connectivity features for efficient integration and testing of SQL queries.

JDBC Connectivity

Database Connection: Establish a reliable connection between the Java application and MySQL database using JDBC.

Query Execution: Ensure the application can execute SQL commands through JDBC for data manipulation and retrieval.

Error Handling: Implement robust exception handling for connection issues or query errors to enhance application stability.

Prepared Statements: Use prepared statements for executing parameterized queries to prevent SQL injection attacks.

Non-Functional Requirements

Java Swing (Frontend)

Performance: The UI must be responsive and minimize latency during user interactions.

Usability: The application should provide an intuitive interface, making it easy for users to understand and navigate.

Scalability: The UI design should accommodate future enhancements, such as additional features or components.

Accessibility: Ensure the application adheres to basic accessibility standards to be usable by individuals with disabilities.

MySQL (Backend)

Reliability: The database must be available and reliable to handle data requests without unexpected downtime.

Scalability: The database structure should be designed to handle growing data volumes without impacting performance.

Security: Implement strong data encryption, backup strategies, and user privilege management to secure the database.

Maintainability: Ensure that the database schema is easy to maintain and modify when requirements change.

Apache NetBeans

Ease of Use: The IDE should support an efficient development process through user-friendly navigation and features.

Compatibility: Must be compatible with various plugins and tools that can enhance productivity and integrate with project needs.

Performance: NetBeans should provide smooth performance during the coding, building, and deployment processes.

Resource Usage: Ensure that the IDE operates efficiently without excessive resource consumption on the development machine.

JDBC Connectivity

Performance: The connection between Java and the database should be fast and reliable for real-time data operations.

Scalability: JDBC should be capable of handling multiple database connections and queries as the application scales.

Security: Use secure methods for database connection, ensuring encrypted data transfer and authentication.

Error Tolerance: The system should handle errors gracefully without crashing, providing clear feedback for debugging purposes.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

Development Machine (PC/Laptop)

Processor: Minimum dual-core processor (Intel Core i3 or equivalent); recommended quad-core processor (Intel Core i5 or higher)

RAM: Minimum 4 GB; recommended 8 GB or more for better multitasking and performance

Storage: At least 500 GB HDD; recommended SSD for faster data access and program execution

Display: 14-inch or larger screen with a resolution of 1366 x 768 pixels or higher for better visualization of the GUI

Graphics: Integrated graphics card; a discrete GPU is optional but can improve rendering during GUI development

Network: Stable internet connection for software downloads and updates

Server (for Database Deployment)

Processor: Quad-core processor or higher

RAM: 8 GB or more for handling concurrent database queries

Storage: SSD with a minimum of 500 GB for database storage; capacity should scale with expected data volume

Backup Storage: Additional external or cloud-based storage for data backup

Software Requirements

Operating System

Development Machine: Windows 10/11, macOS, or a Linux distribution (e.g., Ubuntu)

Server: Windows Server or Linux server distribution (e.g., Ubuntu Server, CentOS)

Development Tools

IDE: Apache NetBeans IDE (latest stable version)

JDK: Java Development Kit (JDK) 17 or later for Java programming

JDBC Driver: MySQL Connector/J for JDBC connectivity

Database Management

Database Server: MySQL Server (version 8.0 or later) for backend storage

Database Management Tools: MySQL Workbench or phpMyAdmin for database management and query testing

Libraries and Dependencies

Java Swing Libraries: Included in the JDK for building GUI applications

JDBC API: Integrated as part of the standard Java API; ensure the proper MySQL Connector/J is included in the project's classpath

Optional Tools

Version Control System: Git for version control, with GitHub or GitLab for repository hosting

Build Tools: Maven or Gradle for project dependency management and build automation

Virtualization Software: Docker (optional) for creating a containerized database instance for testing

Security Tools

Antivirus/Firewall: Reliable antivirus software to ensure a secure development environment

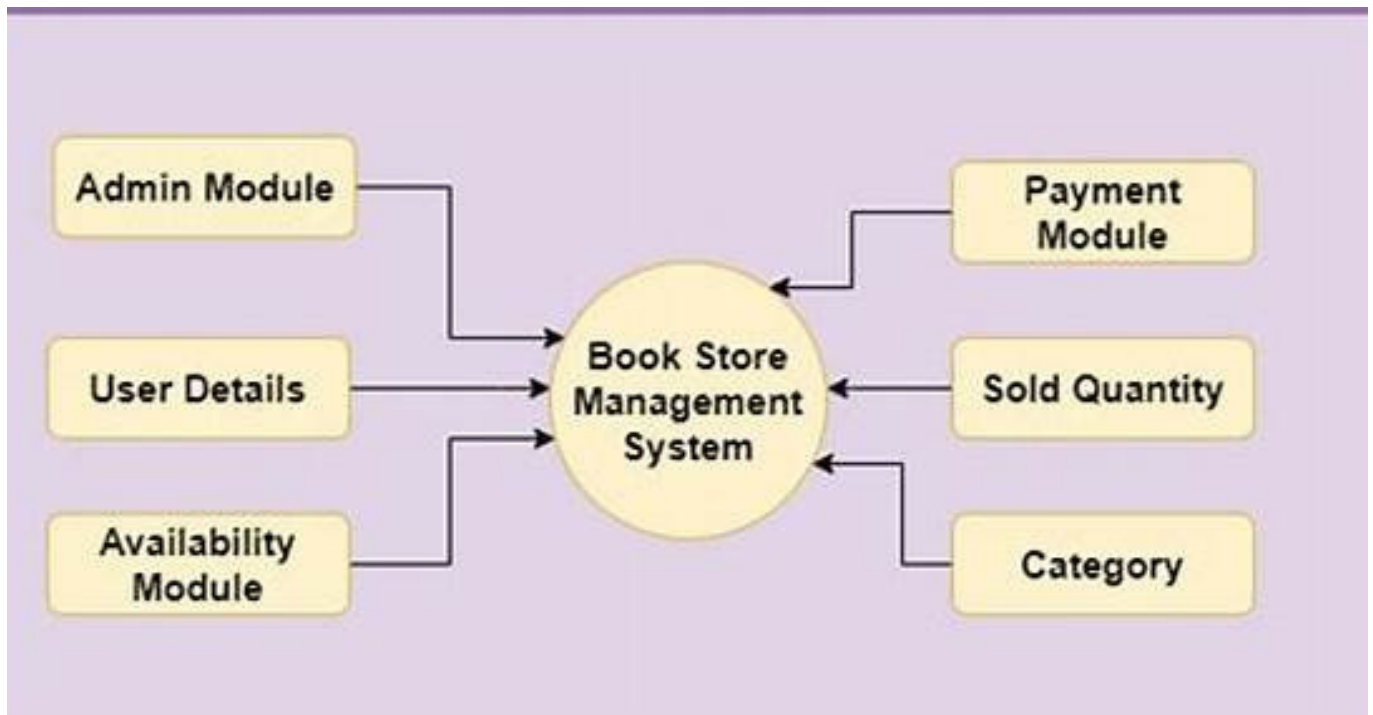
Data Encryption Software: Tools for securing data, if additional encryption is needed

Other Requirements

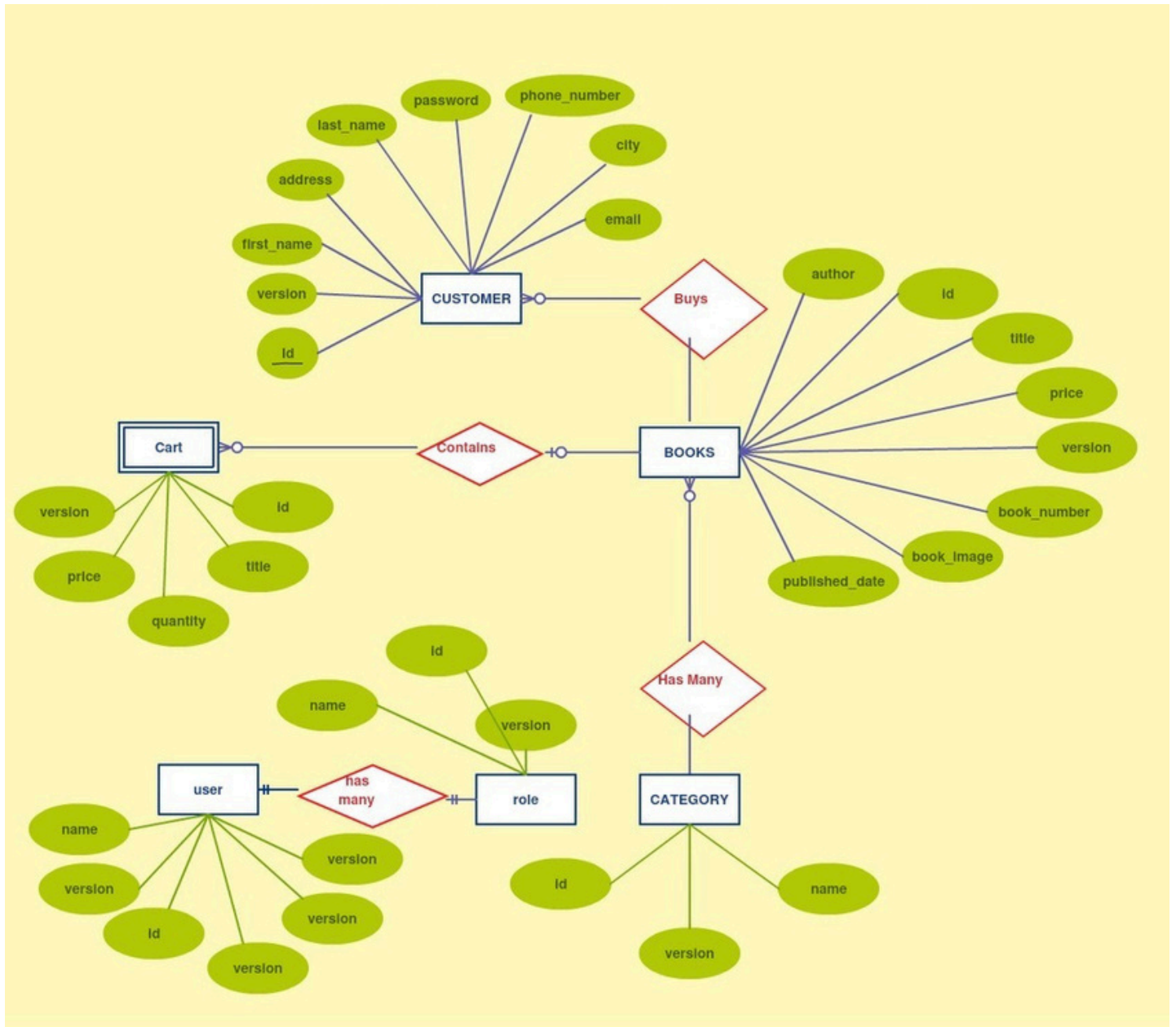
Web Browser: For accessing online resources, database administration tools, and testing web-based MySQL interfaces

PDF Reader: For reading documentation and manuals

3.3 ARCHITECTURE DIAGRAM A visual diagram that provides an overall view of the Book Store management system, identifying the external entities that interact with the system and the major data flows between these entities and the system.



3.4 ER DIAGRAM



3.5 NORMALISATION Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The steps to normalize a database table for Book Store Management System are as follows.

Raw Database

Attribute	Datatype	Example Value
customer_name	VARCHAR(50)	JOHN
customer_address	VARCHAR(50)	No:14, Raman Street, T-Nagar, Chennai
customer_phone_no	NUMBER(10,2)	9785044002
Address_landmark	VARCHAR(50)	Near Grace Stores
Type_of_address	VARCHAR(50)	Residence or Office
Payment_Mode	VARCHAR(50)	Card/Gpay/COD
Name of the book	VARCHAR(50)	Harry Potter
Author_of_the_book	VARCHAR(50)	JK ROWLING
Quantity	VARCCHAR(25)	02

First Normal Form (1NF)

To achieve 1NF, we need to ensure that each cell contains only a single value and each record is unique. We'll separate the blood_types and phone number column into individual rows

customer_name	customer_address	customer_phone no	address_landmark	type_of_address	payment_mode	name of the book	author of the book	quantity
JOHN	NY	9958036958	Grace stores	Residence	COD	Harry P	JK R	02
JOHN	NY	9958036958	Grace stores	Residence	COD	Harry P	JK R	02
JANE	SG	9632512598	Bay beach	Office	Card	The sellout	Paul B	03
JANE	SG	9632512598	Bay beach	Office	Card	The sellout	Paul B	03
.....								

Second Normal Form (2NF) with Phone Numbers

To achieve 2NF, we need to ensure that all non-key attributes are fully functionally dependent on the primary key. We'll decompose the table into multiple tables to remove partial dependencies. We also include phone numbers in the respective tables.

Donor's Table

Customer_id	Customer_name	Customer_add	Customer_landmark	Customer_phone
1	John	NY	Grace Stores	9958036958
2	Jane	SG	Bay beach	9632512598
.....				

Customer Table

Customer_id	Customer_name	Customer_address	Customer_phone
1	Alice	LA	456372890
2	Bob	NY	890674524
.....			

1. DASHBOARD

```
private void B5ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Add_Staff staff=new Add_Staff();  
    staff.setVisible(true);  
}  
  
private void B3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Remove_Books brooks=new Remove_Books();  
    brooks.setVisible(true);  
}  
  
private void B1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Books_Available books=new Books_Available();  
    books.setVisible(true);  
}  
  
private void B2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Add_Books add=new Add_Books();  
    add.setVisible(true);  
}  
  
private void B4ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Staff_Details staff=new Staff_Details();  
    staff.setVisible(true);  
}  
  
private void B7ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Edit_Admin edit=new Edit_Admin();  
    edit.setVisible(true);  
}  
  
private void B6ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Remove_Staff staff=new Remove_Staff();  
    staff.setVisible(true);  
}
```

2. BOOKS AVAILABLE

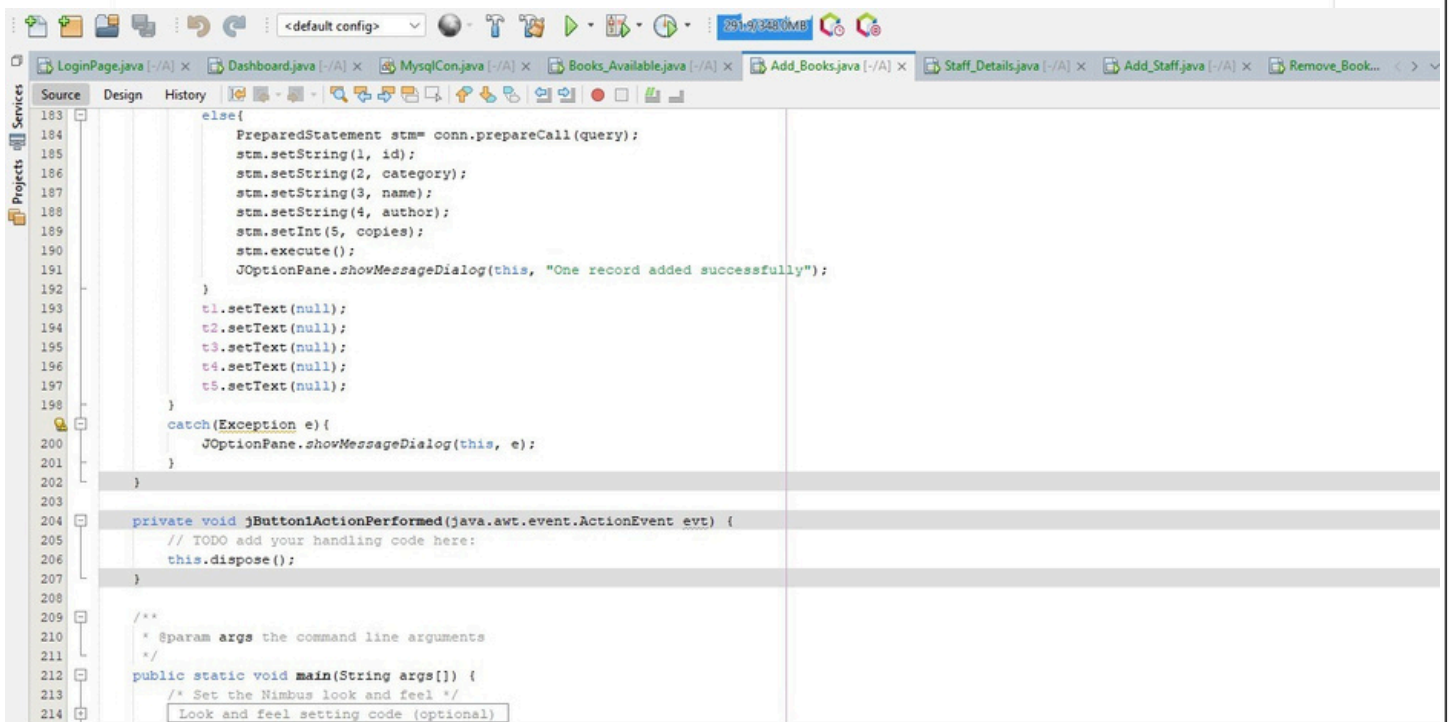
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    DefaultTableModel model= (DefaultTableModel)jTable1.getModel();  
    String url="jdbc:mysql://localhost/library";  
    String user="root";  
    String pwd="Harini@2005";  
    String query="select * from books;";  
    try{  
        Connection conn= DriverManager.getConnection(url,user,pwd);  
        Statement stm=conn.createStatement();  
        ResultSet rs= stm.executeQuery(query);  
        while(rs.next())  
        {  
            String bookid=rs.getString("BOOK_ID");  
            String category=rs.getString("CATEGORY");  
            String name=rs.getString("NAME");  
            String author=rs.getString("AUTHOR");  
            int copies=rs.getInt("COPIES");  
            model.addRow(new Object[] {bookid,category,name,author,copies});  
        }  
        rs.close();  
        stm.close();  
    }  
    catch(Exception e){  
        JOptionPane.showMessageDialog(this, e.getMessage());  
    }  
}
```

3. STAFF DETAILS

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String url="jdbc:mysql://localhost/library";
    String user="root";
    String pwd="Harini@2005";
    String query="insert into books values(?, ?, ?, ?, ?)";
    String id = t1.getText();
    String category=t2.getText();
    String name=t3.getText();
    String author=t4.getText();
    int copies=Integer.parseInt(t5.getText());
    String checkquery="update books set copies=copies+"+copies+" where name='"+name+"' and category='"+category+"' and author='"+author+"'";
    try{
        Connection conn= DriverManager.getConnection(url,user,pwd);
        Statement stmt=conn.createStatement();
        int rows=stmt.executeUpdate(checkquery);
        if(rows>0){
            JOptionPane.showMessageDialog(this, "One record added successfully");
        }
        else{
            PreparedStatement stm= conn.prepareStatement(query);
            stm.setString(1, id);
            stm.setString(2, category);
            stm.setString(3, name);
            stm.setString(4, author);
            stm.setInt(5, copies);
            stm.execute();
            JOptionPane.showMessageDialog(this, "One record added successfully");
        }
        t1.setText(null);
        t2.setText(null);
        t3.setText(null);
        t4.setText(null);
        t5.setText(null);
    }
}

```

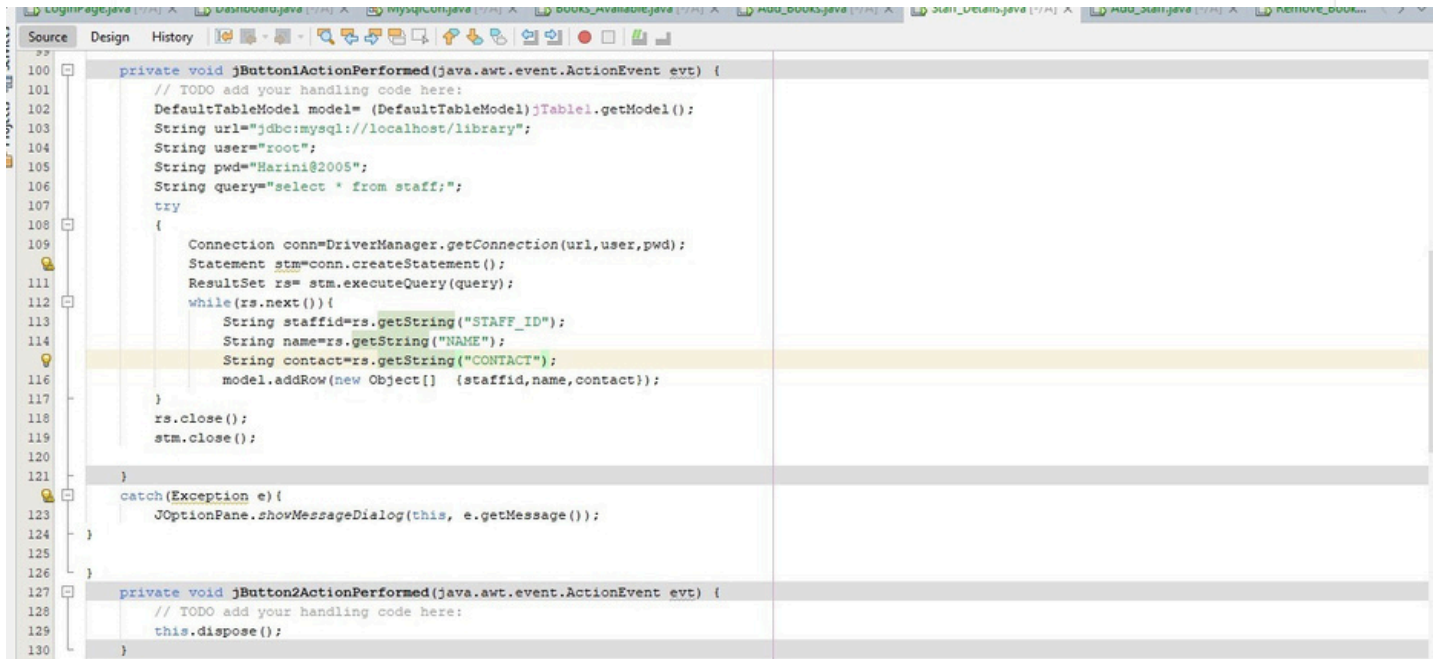


```

183     else{
184         PreparedStatement stm= conn.prepareStatement(query);
185         stm.setString(1, id);
186         stm.setString(2, category);
187         stm.setString(3, name);
188         stm.setString(4, author);
189         stm.setInt(5, copies);
190         stm.execute();
191         JOptionPane.showMessageDialog(this, "One record added successfully");
192     }
193     t1.setText(null);
194     t2.setText(null);
195     t3.setText(null);
196     t4.setText(null);
197     t5.setText(null);
198 }
199
200 catch(Exception e){
201     JOptionPane.showMessageDialog(this, e);
202 }
203
204 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
205     // TODO add your handling code here:
206     this.dispose();
207 }
208
209 /**
210  * @param args the command line arguments
211  */
212 public static void main(String args[]) {
213     /* Set the Nimbus look and feel */
214     LookAndFeel.setLookAndFeel(new NimbusLookAndFeel());
215 }

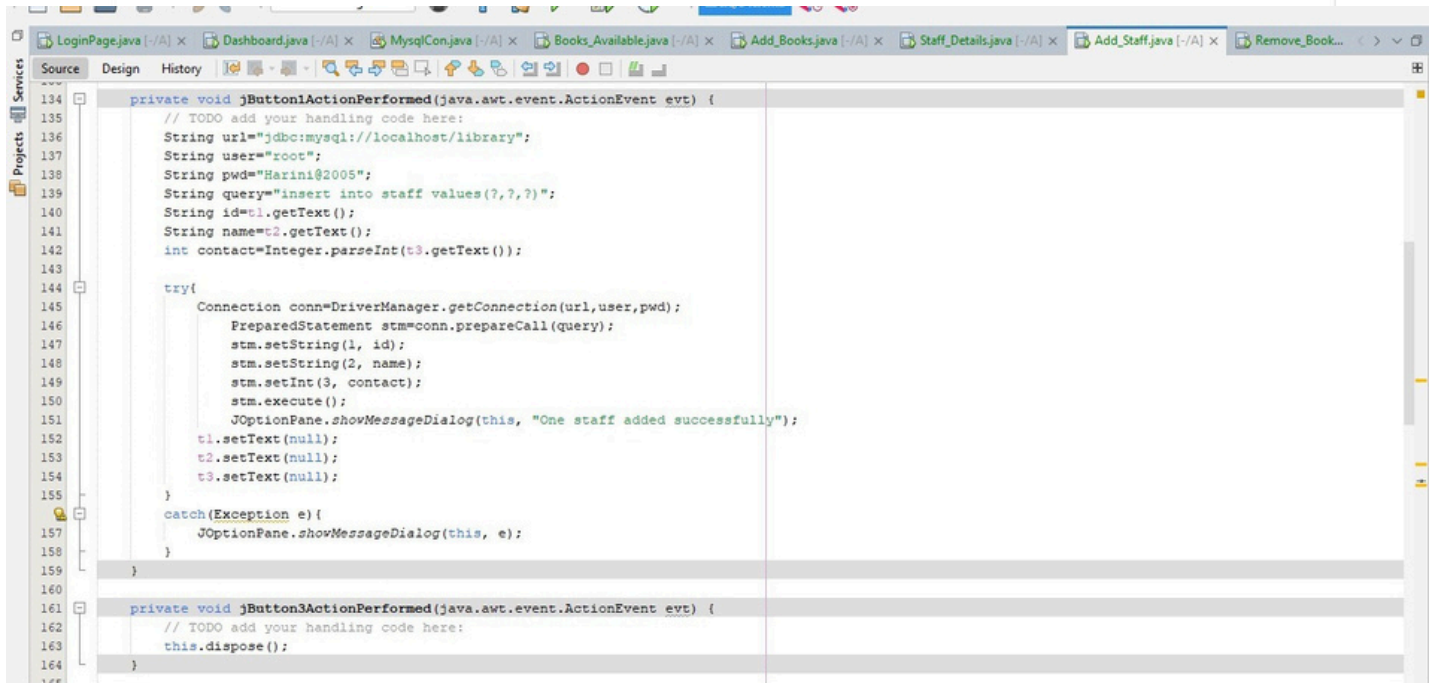
```

4.ADD BOOKS



```
100 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
101     // TODO add your handling code here:
102     DefaultTableModel model= (DefaultTableModel)jTable1.getModel();
103     String url="jdbc:mysql://localhost/library";
104     String user="root";
105     String pwd="Harini@2005";
106     String query="select * from staff;";
107     try
108     {
109         Connection conn=DriverManager.getConnection(url,user,pwd);
110         Statement stmt=conn.createStatement();
111         ResultSet rs= stmt.executeQuery(query);
112         while(rs.next()){
113             String staffid=rs.getString("STAFF_ID");
114             String name=rs.getString("NAME");
115             String contact=rs.getString("CONTACT");
116             model.addRow(new Object[] {staffid,name,contact});
117         }
118         rs.close();
119         stmt.close();
120     }
121     catch(Exception e){
122         JOptionPane.showMessageDialog(this, e.getMessage());
123     }
124 }
125
126 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
127     // TODO add your handling code here:
128     this.dispose();
129 }
130 }
```

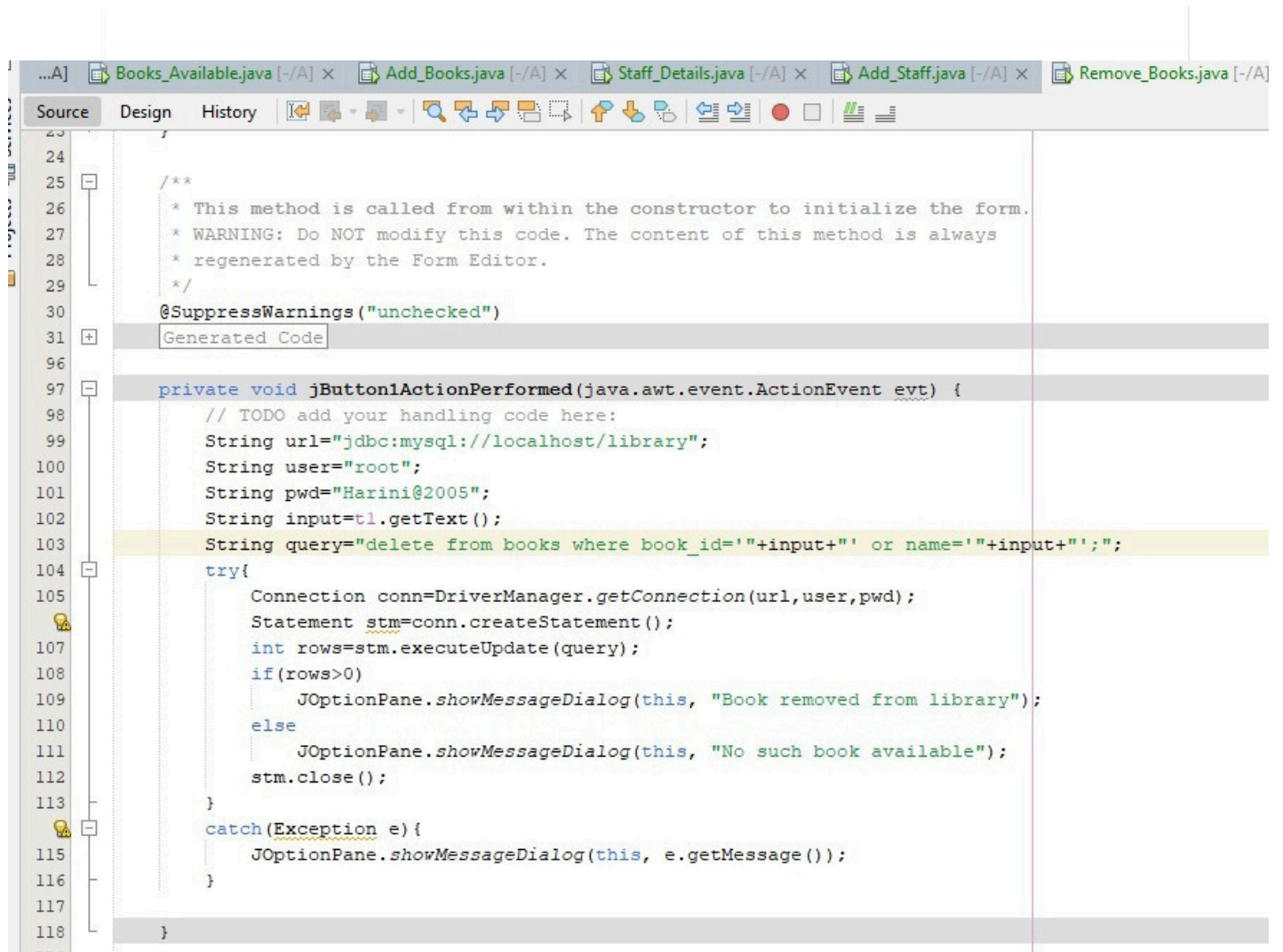

5.ADD STAFF



The screenshot shows an IDE with several open files. The active file is `Add_Staff.java`, which contains the following Java code:

```
134 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
135     // TODO add your handling code here:  
136     String url="jdbc:mysql://localhost/library";  
137     String user="root";  
138     String pwd="Harini@2005";  
139     String query="insert into staff values (?, ?, ?)";  
140     String id=t1.getText();  
141     String name=t2.getText();  
142     int contact=Integer.parseInt(t3.getText());  
143  
144     try{  
145         Connection conn=DriverManager.getConnection(url,user,pwd);  
146         PreparedStatement stm=conn.prepareStatement(query);  
147         stm.setString(1, id);  
148         stm.setString(2, name);  
149         stm.setInt(3, contact);  
150         stm.execute();  
151         JOptionPane.showMessageDialog(this, "One staff added successfully");  
152         t1.setText(null);  
153         t2.setText(null);  
154         t3.setText(null);  
155     }  
156     catch(Exception e){  
157         JOptionPane.showMessageDialog(this, e);  
158     }  
159 }  
160  
161 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
162     // TODO add your handling code here:  
163     this.dispose();  
164 }  
165
```

6.REMOVE BOOKS

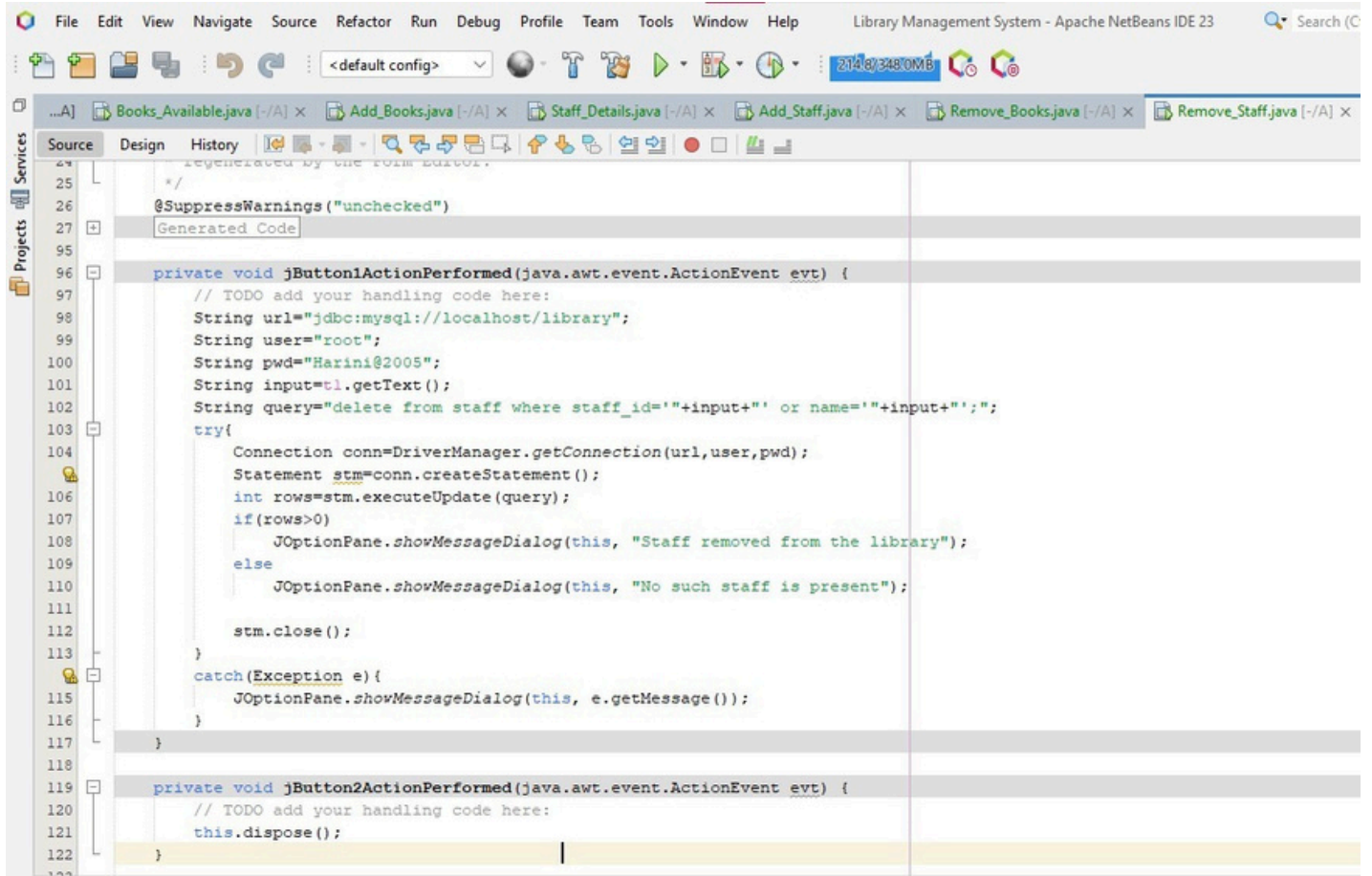


```

23
24
25 /**
26  * This method is called from within the constructor to initialize the form.
27  * WARNING: Do NOT modify this code. The content of this method is always
28  * regenerated by the Form Editor.
29  */
30 @SuppressWarnings("unchecked")
31 Generated Code
96
97 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
98     // TODO add your handling code here:
99     String url="jdbc:mysql://localhost/library";
100     String user="root";
101     String pwd="Harini@2005";
102     String input=t1.getText();
103     String query="delete from books where book_id='"+input+"' or name='"+input+"'";
104     try{
105         Connection conn=DriverManager.getConnection(url,user,pwd);
106         Statement stm=conn.createStatement();
107         int rows=stm.executeUpdate(query);
108         if(rows>0)
109             JOptionPane.showMessageDialog(this, "Book removed from library");
110         else
111             JOptionPane.showMessageDialog(this, "No such book available");
112         stm.close();
113     }
114     catch(Exception e){
115         JOptionPane.showMessageDialog(this, e.getMessage());
116     }
117
118 }

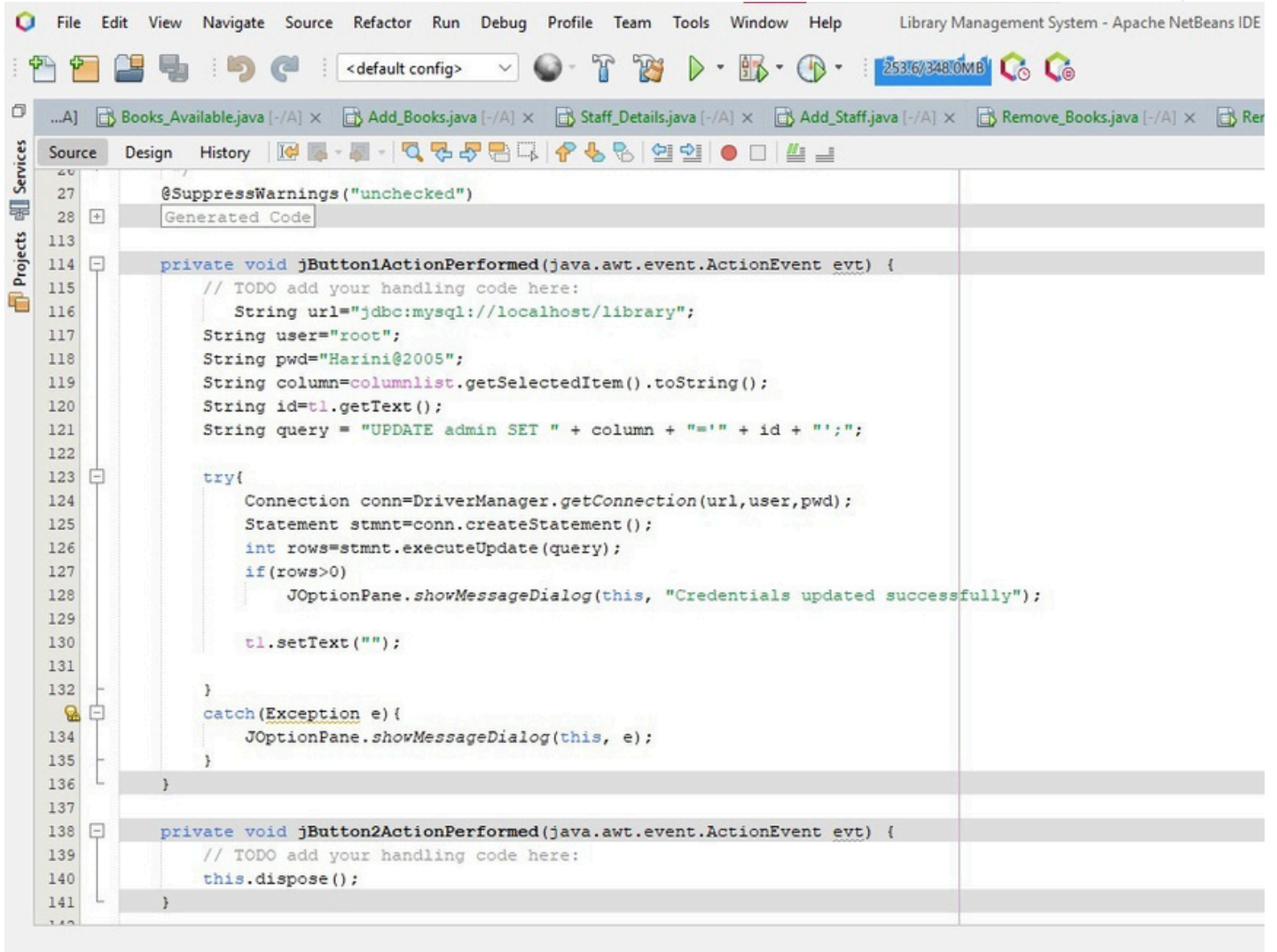
```

7.REMOVE STAFF



```
24  //regenerated by the form editor.
25  */
26  @SuppressWarnings("unchecked")
27  Generated Code
95
96  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
97      // TODO add your handling code here:
98      String url="jdbc:mysql://localhost/library";
99      String user="root";
100     String pwd="Harini@2005";
101     String input=t1.getText();
102     String query="delete from staff where staff_id='"+input+"' or name='"+input+"'";
103     try{
104         Connection conn=DriverManager.getConnection(url,user,pwd);
105         Statement stm=conn.createStatement();
106         int rows=stm.executeUpdate(query);
107         if(rows>0)
108             JOptionPane.showMessageDialog(this, "Staff removed from the library");
109         else
110             JOptionPane.showMessageDialog(this, "No such staff is present");
111
112         stm.close();
113     }
114     catch(Exception e){
115         JOptionPane.showMessageDialog(this, e.getMessage());
116     }
117 }
118
119 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
120     // TODO add your handling code here:
121     this.dispose();
122 }
```


8.EDIT ADMIN



```
20
21
22 @SuppressWarnings("unchecked")
23 Generated Code
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
115     // TODO add your handling code here:
116     String url="jdbc:mysql://localhost/library";
117     String user="root";
118     String pwd="Harini@2005";
119     String column=columnlist.getSelectedItem().toString();
120     String id=t1.getText();
121     String query = "UPDATE admin SET " + column + "=" + id + ";";
122
123     try{
124         Connection conn=DriverManager.getConnection(url,user,pwd);
125         Statement stmt=conn.createStatement();
126         int rows=stmt.executeUpdate(query);
127         if(rows>0)
128             JOptionPane.showMessageDialog(this, "Credentials updated successfully");
129
130         t1.setText("");
131
132     }
133     catch(Exception e){
134         JOptionPane.showMessageDialog(this, e);
135     }
136 }
137
138 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
139     // TODO add your handling code here:
140     this.dispose();
141 }
142
```

DASHBOARD

BOOKSTORE MANAGEMENT SYSTEM**Dashboard**

BOOKS AVAILABLE

STAFF DETAILS

ADD BOOKS

ADD STAFF

REMOVE BOOKS

REMOVE STAFF

EDIT ADMIN



The screenshot shows a window titled "LOGIN" with a light gray background. It contains two input fields: "USERNAME" with the text "ronald@123" and "PASSWORD" with masked characters "*****". Below the fields is a "LOGIN" button. The window has standard Windows-style title bar controls (minimize, maximize, close) in the top right corner.

LOGIN

USERNAME ronald@123

PASSWORD *****

LOGIN

BOOKS AVAILABLE

BOOK_ID	CATEGORY	NAME	AUTHOR	COPIES

FETCH

BACK

Message:

Send Message

ADD BOOKS

BOOK ID

CATEGORY

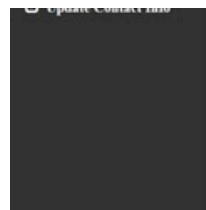
NAME

AUTHOR

COPIES

ADD

CANCEL



STAFF DETAILS

STAFF ID	NAME	CONTACT

FETCH**EXIT**

ADD STAFF

STAFF ID

NAME

CONTACT

ADD

CANCEL

REMOVE BOOKS

ENTER BOOK ID OR BOOK NAME TO DELETE

DELETE

CANCEL

REMOVE STAFF

ENTER STAFF ID OR STAFF NAME TO DELETE

DELETE

CANCEL

EDIT THE CONTENT

User_id



ENTER THE VALUE TO BE UPDATED

UPDATE

CANCEL

[6:50]

In conclusion, the Bookstore Management System developed using Java Swing for the front end, MySQL as the backend, and NetBeans as the development environment successfully demonstrates an effective approach to managing bookstore operations. The application provides an intuitive and interactive user interface for users to handle various tasks, such as managing books, customer details, sales transactions, and inventory tracking.

The integration between Java Swing and MySQL through JDBC, facilitated by NetBeans, has proven efficient and scalable, allowing seamless data management and ensuring data persistence. This system not only automates manual tasks, improving efficiency and accuracy, but also enhances data accessibility and integrity by storing information in a structured database. Additionally, the modular design and object-oriented approach make it adaptable for future expansion, such as adding more features, integrating with other systems, or deploying it on the web.

Overall, the system achieves its objectives, providing a reliable solution to simplify bookstore management processes, making it a valuable tool for small to medium-sized bookstores.

This conclusion wraps up the project by summarizing its key points, benefits, and future possibilities, aligning with typical academic or project report expectations.

7.1 REFERENCES

[1] GitHub: Library Management System by MuhammadNoman76

[2] GitHub: Library Management Project by Aman1905

[3] Edureka: S tep-by-step guide for Library Management System