

**BLOOD BANK MANAGEMENT SYSTEM**

For the Evaluation of

**Project Mode – CS23333- Object Oriented Programming using Java**

Submitted by

Harishini T (231001059) Swetha A (231001226)

Mini Project

November 2024

***Department of Information Technology***

**Rajalakshmi Engineering College, Thandalam**

# Bonafide Certificate

This is to certify that the Mini project work titled **“Blood Bank Management System**” done by **“Student Indhujha N (231001065), Jacitha R (231001066), Jayasri J (231001074)”** is a record of **bonafide** work carried out by him/her under my supervision as a part of Mini project for the **Course CS23332- Database Management System, Department of Information Technology, REC.**

**SUPERVISOR Head/IT**

**Mr.Narayanan Dr. Valarmathi**

**Date:**

# BLOOD BANK MANAGEMENT SYSTEM ABSTRACT

* The abstract of a Blood Bank Management System typically provides a concise overview of the system. It may include information about the system's purpose, features, and benefits.
* The goal of blood management is to ensure the safe and efficient use of the many resources involved in the complex process of blood component theory.
* "The Blood Bank Management System is a comprehensive software solution designed to streamline and enhance the operations of blood banks. This system facilitates efficient donor registration, inventory management, and blood distribution. With user-friendly interfaces, real- time tracking, and robust reporting capabilities, the Blood Bank Management System aims to improve overall blood bank functionality, ensuring timely access to blood units and promoting effective donor management."

## INDEX

|  |  |  |
| --- | --- | --- |
| **S. No** | **CHAPTER** | **PG. No.** |
| **1** | **INTRODUCTION** | **5** |
| **1.1** | **PROBLEM STATEMENT** | **5** |
| **1.2** | **OBJECTIVES OF THE PROJECT** | **6** |
| **1.3** | **ORGANIZATION OF THE REPORT** | **6** |
| **2** | **SYSTEM DESIGN** | **7** |
| **2.1** | **UML MODELING** | **8** |
| **2.1.1** | **USE CASE DIAGRAM/CLASS DIAGRAM** | **9** |
| **3** | **SYSTEM SPECIFICATION** | **11** |
| **3.1** | **SOFTWARE REQUIREMENTS** | **11** |
| **3.2** | **HARDWARE REQUIREMENTS** | **14** |
| **3.3** | **TOOLS /PLATFORMS PROCEDURE** | **14** |
| **4** | **IMPLEMENTATION** | **16** |
| **4.1** | **INPUT IMAGE** | **16** |
| **4.2** | **CODING** | **17** |
| **4.3** | **OUTPUT IMAGE** | **68** |
| **5** | **CONCLUSION** | **71** |

# 1. INTRODUCTION

A Blood Bank Management System in Java is a program that helps organize and streamline the activities of a blood bank. It allows users to efficiently manage donor information, track blood inventory, and facilitate the process of blood transfusions. Using Java, the system can provide a user-friendly interface for tasks such as adding donor details, recording blood donations, and monitoring available blood types. This helps in maintaining an organized and accessible record of blood-related activities, ensuring a more effective and responsive blood bank operation.

## 1.1 PROBLEM STATEMENT

The existing manual processes within blood banks often lead to inefficiencies, inaccuracies, and delays in managing the crucial resource of donated blood. The lack of a streamlined system for inventory tracking, donor management, and distribution hinders the ability to respond effectively to the dynamic demands of hospitals and healthcare facilities.

Developing a comprehensive Blood Bank Management System that automates and integrates key processes, including donor registration, inventory tracking, blood testing, and distribution. This system aims to enhance operational efficiency, improve data accuracy, and provide valuable insights for better decision-making within blood banks.

## 1.2 OBJECTIVE OF THE PROJECT

The objectives of a Blood Bank Management System generally revolve around efficiently managing blood inventory, ensuring the availability of safe blood, and facilitating smooth operations. The key objectives includes Inventory Management,

Donor Management, Blood Collection and Processing, Distribution and Transfusion, Quality Control and Testing. Appointment Scheduling and Reminders, Reporting and Analytics, Accessibility and Security, Compliance and Documentation

By achieving these objectives, a Blood Bank Management System aims to enhance the efficiency, safety, and accessibility of blood products while supporting the overall healthcare system.

## 1.3 ORGANIZATION OF THE REPORT

**Introduction:** Briefly introduce the purpose and scope of the Blood Bank Management System. Outline the key features and functionalities.

**System Design:** Explain structure and design of the database used in the system.

**System Requirements:** Explains hardware and software requirements for the system.

**System Architecture:** Describes the overall architecture of the system.

**User Interface and functionality:** Discuss the graphical user interface and how users interact with the system. Provide an overview of the main functionalities, such as donor registration and blood inventory management.

**Conclusion:** Summarize key points and emphasize the significance of the Blood Bank Management System.

# 2. SYSTEM DESIGN

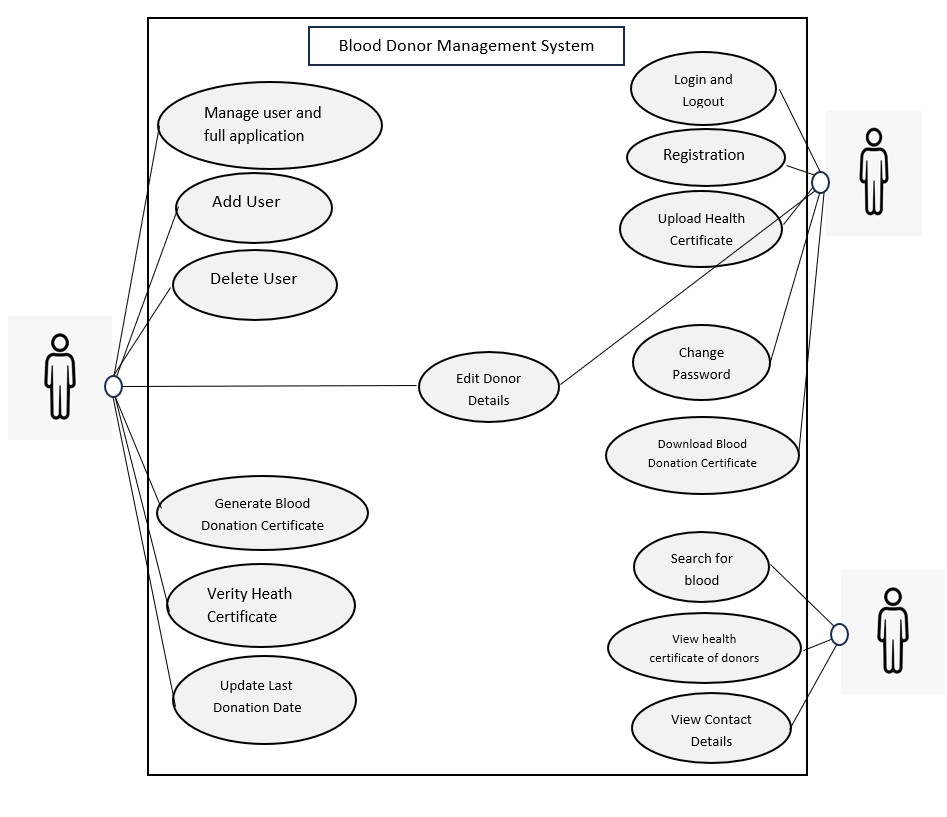
Designing a Blood bank management system involves various components.

Here's a high-level overview of the system design:

* Java-based GUI for user-friendly interactions.
* SQL database for donor, recipient, and inventory data.
* Java backend for business logic and JDBC for database connectivity.
* Donor registration, blood request, inventory management.
* User authentication, role-based access control, data encryption.
* Generate reports for donations, inventory, and metrics.
* Design for future growth, optimize performance.
* Connect with external services if necessary.
* Provide user and admin documentation.
* Implement robust error handling and logging.

It consists of UML model and Use case diagram or the class diagram. UML (Unified Modeling Language) modeling is a standardized visual language used in software engineering to represent and document system designs through a set of diagrams, facilitating communication and understanding among stakeholders during the software development process. UMI. diagrams include representations of classes, objects, relationships, and behaviors within a system A use case diagram in UML illustrates the interactions between various actors and a system, showcasing the different ways users (actors) can interact with the system to achieve specific goals or functionalities. It provides a high-level view of system functionalities and user interactions.

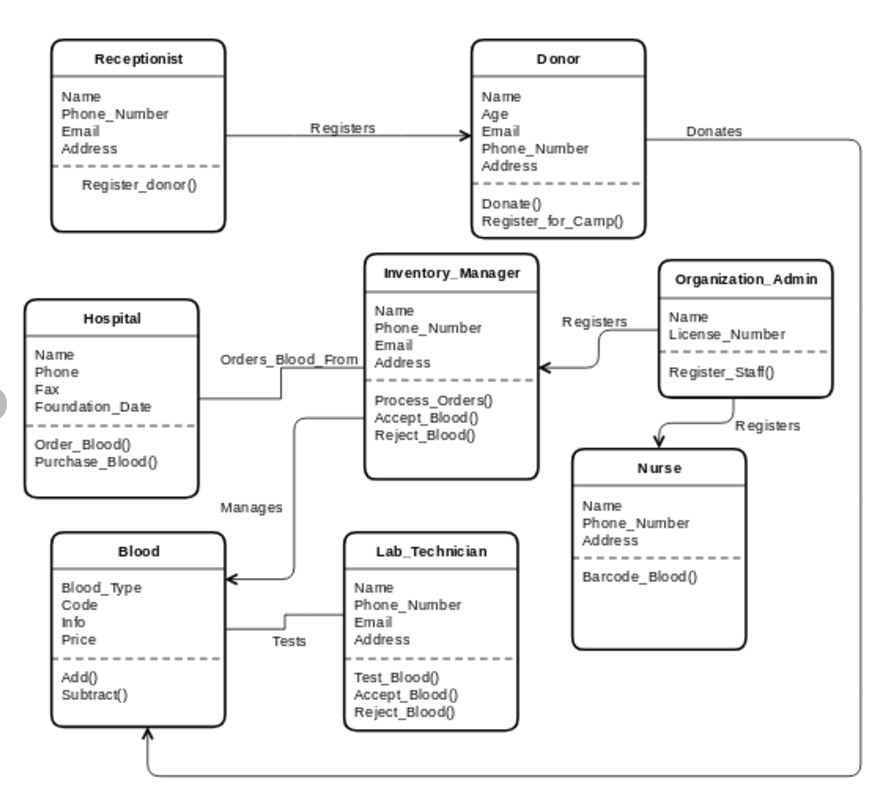
## 2.1 UML MODELING



### Fig.2.1.1 UML MODEL DIAGRAM

The above picture depicts the project's Sequence Diagram, which is a sort of interaction diagram since it describes how and in what order a set of items interacts with one another. A sequence diagram focuses on lifelines or processes and objects that coexist, and the messages transferred between them to complete a function before the lifeline terminates.

### 2.1.1 USE CASE DIAGRAM / CLASS DIAGRAM



# Fig.2.1.2 USE CASE DIAGRAM

A use case diagram for a Blood Bank Management System represents the interactions between different actors (users or external systems) and the system itself. This use case diagram provides a high-level overview of the interactions between actors and the Blood Bank Management System. Depending on the specific functionalities and requirements of your system, we might need to expand or customize this diagram further.

# 3. SYSTEM SPECIFICATION

A system specification succinctly outlines the functional and non-functional requirements of a software system, detailing its expected behavior and performance characteristics. It serves as a blueprint for system development and guides the design and implementation processes.

The system design of Blood Bank Management System facilitates efficient donor registration, blood inventory management, and donor history tracking. Users, including administrators and receptionists, interact with a user-friendly graphical interface for seamless operations. The system ensures secure handling of sensitive information and implements robust security measures. It features a centralized database design for effective data storage and retrieval. The Blood Bank Management System is designed to enhance the overall workflow of blood bank operations, providing a reliable and user-centric solution.

## 3.1 SOFTWARE REQUIREMENTS

The software requirements for a Blood Bank Management System encompass various components to ensure efficient management of blood donations, inventory, donors, and related processes. Here's an outline of the software requirements:

**Operating System:** The system should be compatible with commonly used operating systems such as Windows, Linux, or macOS.

**Database Management System (DBMS):** A robust database system to storeand manage donor information, blood inventory, transactional data, etc. Examples include MySQL, PostgreSQL, MongoDB, or Oracle.

**Programming Languages and Frameworks:** Selection of appropriate programming languages and frameworks for software development, such as:

**Backend development:** Java, Python, Node.js, NET, PHP, etc.

**Frontend development:** HTML, CSS, JavaScript, React, Angular etc.

**Web Server:** A web server software to host and manage the application, for example: Apache HTTP Server, Nginx, Microsoft Internet Information Services.

**Security Measures:** Encryption protocols for secure data transmission (SSL/TLS).

Access control mechanisms to restrict unauthorized access to sensitive information.

Regular security updates and patches to protect against vulnerabilities.

**User Interface (UI) Design Tools:** Tools and frameworks for designing a user- friendly interface. Prototyping tools like Adobe XD, Sketch, or Figma.

Frontend libraries for creating responsive and intuitive user interfaces.

**Testing Tools:** Software for quality assurance and testing purposes:

Automated testing frameworks (e.g., Selenium, Jest, JUnit). Debugging tools for identifying and fixing issues.

**Reporting and Analytics Tools:** Tools to generate reports and analyze data.

Reporting libraries or frameworks (e.g., Jasper Reports, Microsoft Power BI, Tableau).

Analytics tools to derive insights from blood inventory, donation patterns, etc.

**Integration and APIs:** Integration with other systems and APIs for functionalities like payment gateways for online donations. Integration with hospital systems for blood request and delivery.

**Backup and Recovery Systems:** Mechanisms to regularly backup data and protocols for disaster recovery to prevent data loss.

**Documentation and User Manuals:** Creation of comprehensive documentation and user manuals for system administrators, staff, and users to understand system functionalities and operations.

These software requirements form the foundation for developing a robust Blood Bank Management System, ensuring it meets the operational needs while maintaining security, usability, and compliance standards.

**FUNCTIONAL REQUIREMENTS:**

1. The system should allow for efficient donor registration, capturing donor information such as name, contact details, and blood type.
2. It must support blood inventory management, enabling the addition, removal, and tracking of blood units with details on blood type and quantity.
3. The system should facilitate donor history tracking, maintaining a record of donation dates, quantities, and relevant medical information.
4. Users, including administrators and receptionists, should have secure login access with role-based permissions for specific functionalities.
5. It should generate reports on blood inventory levels, donor statistics, and other relevant metrics to aid in decision-making and management.

**NON-FUNCTIONAL REQUIREMENTS:**

1. The system should ensure a high level of security, implementing encryption and access controls to protect sensitive donor and medical data.
2. It must provide a responsive and user-friendly interface, ensuring ease of use for administrators, receptionists, and other users.
3. The system should have high availability, minimizing downtime to ensure continuous access to critical blood bank functionalities.
4. It should be scalable, accommodating an increase in the volume of donor registrations, blood inventory, and overall system usage.
5. The system must comply with relevant regulatory standards and data protection laws to ensure ethical and legal use of donor and patient information.

## 3.2 HARDWARE REQUIREMENTS

The hardware requirements depend on factors such as the scale of the system, concurrent users, and the complexity of operations. The basic requirements are:

**Processor:** A modern multi-core processor (e.g., Intel Core i5 or equivalent) to handle Java-based application processing efficiently.

**Memory (RAM):** At least 4GB of RAM for smooth execution. If the system is expected to handle a larger dataset or simultaneous users, consider increasing the RAM accordingly.

**Storage:** Allocate sufficient hard disk space for the Java runtime environment, the application, and any associated files. A minimum of 20GB is a reasonable starting point.

**Network Connection:** A stable internet connection is recommended for potential updates, user authentication, and any network-related features.

**Display:** A monitor with a resolution suitable for Java GUI development, typically 1280x800 or higher.

**Input Devices**: Standard keyboard and mouse for development and data input.

These requirements may vary based on expected user load and the amount of data being processed. The scalability and potential future growth when determining hardware requirements is always considered.

## 3.3 TOOLS/PLATFORM

When developing a Blood Bank Management System using Java, several tools and platforms are used.

**Java SE (Standard Edition):**

The core Java platform provides the fundamental tools and libraries necessary for building Java applications. The latest version should be used.

**Integrated Development Environment (IDE):**

Choose a Java IDE for efficient development. This project is implemented using Eclipse IDE.

**Database Management System (DBMS):**

Use a reliable DBMS to handle the storage of data. MySQL database is used here. **Java Database Connectivity (JDBC):**

JDBC allows Java applications to interact with databases. It's a key component for database connectivity in Java projects.

**Java Swing for GUI:**

Select a graphical user interface (GUI) for creating the front end like Java Swing **Java Servlets and JSP (JavaServer Pages):**

For web-based applications, Java Servlets and JSP can be used to handle serverside logic and generate dynamic content.

**Spring Framework:**

Spring provides comprehensive support for building Java enterprise applications. It includes features for data access, transaction management, and more.

**Version Control System:**

Use a version control system such as Git to manage your source code.

# 4. IMPLEMENTATION

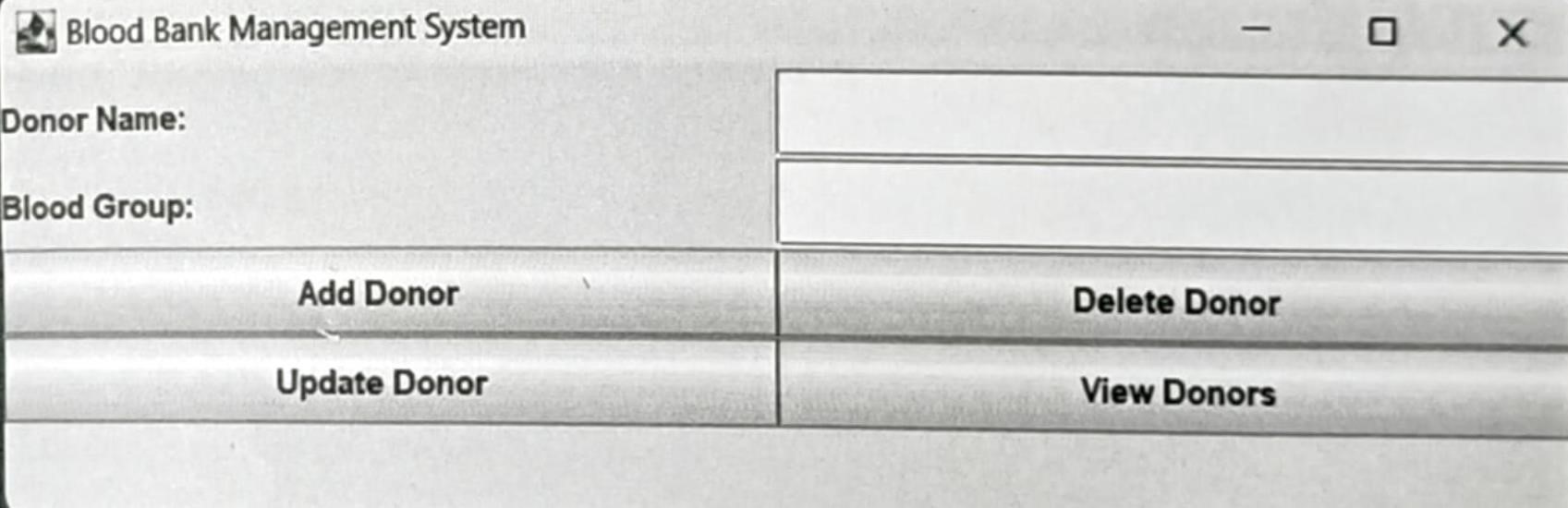
**The implementation of code includes:**

* Creation of a new java project in eclipse IDE
* Creation of a java class and write the code.
* Database connection details must be given in the code.
* Add MySQL JDBC Driver or the jar file to the projects build path
* Run the program and the program will open a GUI window with text fields, buttons, and labels.
* Add donors, delete donors, update donor information, and view donors using the respective buttons.

**Before running the code, we need to ensure:**

* A MySQL database named "bloodbank" set up.
* A table named "donors" with columns donor\_name and blood group in the "bloodbank" database.

## 4.1 INPUT IMAGES



### Fig.4.1.1 INPUT IMAGE

A window with two text fields (for donor name and blood group), a button to add donors, a button to delete donors, a button to view donors, a button to update donors, and labels for the text fields appears when the program is executed.

**4.2 CODING:**

**CONNECTION PROVIDER.JAVA**

package Project; import java.sql.\*;

public class ConnectionProvider

{

public static Connection getCon(){

try {

Class.forName("con.mysql.jdbc.Driver");

Connection

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms","root","de vi");

return con;

}

catch(Exception e) { return null;

}

}

}

**BLOOD GROUP SEARCH.JAVA**

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

public class BloodGroupSearch extends JFrame { private JTextField bloodGroupField; private JTextArea resultArea;

public BloodGroupSearch() { setTitle("Search Blood Group"); setSize(679, 506);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel bloodGroupLabel = new JLabel("Enter Blood Group:"); bloodGroupLabel.setFont(new Font("Tahoma", Font.BOLD, 15)); bloodGroupLabel.setBounds(34, 135, 174, 33); bloodGroupField = new JTextField(); bloodGroupField.setBounds(260, 122, 337, 59);

JButton searchButton = new JButton("Search"); searchButton.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\search1.png"))

;

searchButton.setFont(new Font("Tahoma", Font.BOLD, 14)); searchButton.setBounds(60, 227, 148, 44);

resultArea = new JTextArea(); resultArea.setBounds(34, 305, 574, 129); resultArea.setEditable(false);

getContentPane().setLayout(null); getContentPane().add(bloodGroupLabel); getContentPane().add(bloodGroupField); getContentPane().add(searchButton); getContentPane().add(resultArea);

JSeparator separator = new JSeparator(); separator.setBounds(0, 102, 691, 10); getContentPane().add(separator);

JLabel lblNewLabel = new JLabel("Search Donor Details"); lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 27)); lblNewLabel.setBounds(158, 41, 337, 33); getContentPane().add(lblNewLabel); JButton btnClose = new JButton("Close"); btnClose.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { setVisible(false);

}

});

btnClose.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

btnClose.setFont(new Font("Tahoma", Font.BOLD, 14)); btnClose.setBounds(306, 227, 148, 44); getContentPane().add(btnClose);

JLabel lblNewLabel\_1 = new JLabel("New label");

lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background image.png"));

lblNewLabel\_1.setBounds(0, 10, 667, 461); getContentPane().add(lblNewLabel\_1);

searchButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { searchBloodGroup();

}

});

setVisible(true);

}

private void searchBloodGroup() {

String bloodGroup = bloodGroupField.getText();

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

try {

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root", "devi");

String sql = "SELECT \* FROM Donors WHERE blood\_group = ?"; stmt = conn.prepareStatement(sql); stmt.setString(1, bloodGroup);

rs = stmt.executeQuery();

StringBuilder result = new StringBuilder();

while (rs.next()) {

int donorId = rs.getInt("donor\_id");

String donorName = rs.getString("donor\_name");

String donorEmail = rs.getString("donor\_email");

String donorBloodGroup = rs.getString("blood\_group");

result.append("Donor ID: ").append(donorId) .append(", Name: ").append(donorName)

.append(", Email: ").append(donorEmail)

.append(", Blood Group: ").append(donorBloodGroup)

.append("\n");

}

if (result.length() == 0) {

resultArea.setText("No donors found for blood group: " + bloodGroup);

} else {

resultArea.setText(result.toString());

}

} catch (SQLException ex) {

ex.printStackTrace();

resultArea.setText("Error: Unable to retrieve donors.");

} finally { try { if (rs != null) { rs.close();

}

if (stmt != null) { stmt.close();

}

if (conn != null) { conn.close();

}

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

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new BloodGroupSearch());

}

}

**DONOR DELETION FRAME.JAVA**

import javax.swing.\*; import java.awt.event.\*; import java.sql.\*; import java.awt.BorderLayout; import java.awt.Font;

public class DonorDeletionFrame extends JFrame { private JTextField donorIDField; private JButton deleteButton; private Connection connection; private PreparedStatement preparedStatement; private JLabel lblNewLabel; private JButton btnClose; private JLabel lblNewLabel\_1;

public DonorDeletionFrame() { setTitle("Delete Donor"); setSize(708, 505);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); getContentPane().setLayout(null); donorIDField = new JTextField(10); donorIDField.setBounds(247, 199, 207, 37); getContentPane().add(donorIDField); deleteButton = new JButton("Delete");

deleteButton.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete donor.png"));

deleteButton.setFont(new Font("Tahoma", Font.BOLD, 16)); deleteButton.setBounds(98, 333, 172, 66); getContentPane().add(deleteButton);

JLabel lblDonorId = new JLabel("Donor ID:"); lblDonorId.setFont(new Font("Tahoma", Font.BOLD, 15)); lblDonorId.setBounds(98, 193, 110, 45); getContentPane().add(lblDonorId);

lblNewLabel = new JLabel("Delete Donor Details"); lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 30)); lblNewLabel.setBounds(151, 52, 359, 37);0 getContentPane().add(lblNewLabel);

JSeparator separator = new JSeparator(); separator.setBounds(10, 121, 674, 44); getContentPane().add(separator); btnClose = new JButton("Close");

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

}

});

btnClose.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

btnClose.setFont(new Font("Tahoma", Font.BOLD, 16)); btnClose.setBounds(356, 333, 172, 66); getContentPane().add(btnClose);

lblNewLabel\_1 = new JLabel("New label");

lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background image.png"));

lblNewLabel\_1.setBounds(10, 10, 674, 448); getContentPane().add(lblNewLabel\_1);

deleteButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { deleteDonor();

}

});

setLocationRelativeTo(null);

// Connect to the database try {

connection =

DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root", "devi");

} catch (SQLException e) {

e.printStackTrace();

}

}

private void deleteDonor() {

try {

int donorID = Integer.parseInt(donorIDField.getText());

String deleteQuery = "DELETE FROM Donors WHERE donor\_id = ?";

preparedStatement = connection.prepareStatement(deleteQuery); preparedStatement.setInt(1, donorID);

int rowsAffected = preparedStatement.executeUpdate(); if (rowsAffected > 0) { JOptionPane.showMessageDialog(this, "Donor deleted successfully.");

} else {

JOptionPane.showMessageDialog(this, "No donor found with that ID.");

}

} catch (SQLException | NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());

}

}

public static void main(String[] args) {

// Ensure the database driver is loaded (e.g., for MySQL) try {

Class.forName("com.mysql.cj.jdbc.Driver"); } catch (ClassNotFoundException e) {

e.printStackTrace(); return;

}

SwingUtilities.invokeLater(() -> {

DonorDeletionFrame frame = new DonorDeletionFrame(); frame.setVisible(true);

});

}

}

**ADD NEW DONOR.JAVA**

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

import Project.ConnectionProvider; public class addNewDonar extends JFrame {

/\*\*

\* \*/

private static final long serialVersionUID = 1L; private JTextField nameField, emailField, bloodGroupField; private JButton submitButton; private JButton btnClose; private JLabel lblNewLabel; private JSeparator separator; private JSeparator separator\_1; private JLabel lblNewLabel\_1;

public addNewDonar() { setTitle("Donor Information"); setSize(708, 559);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel nameLabel = new JLabel("Name: ");

nameLabel.setFont(new Font("Tahoma", Font.BOLD, 16)); nameLabel.setBounds(18, 114, 171, 67); nameField = new JTextField();

nameField.setFont(new Font("Tahoma", Font.PLAIN, 16)); nameField.setBounds(320, 118, 364, 59); JLabel emailLabel = new JLabel("Email: ");

emailLabel.setFont(new Font("Tahoma", Font.BOLD, 16)); emailLabel.setBounds(18, 210, 228, 74); emailField = new JTextField();

emailField.setFont(new Font("Tahoma", Font.PLAIN, 16)); emailField.setBounds(320, 218, 364, 59);

JLabel bloodGroupLabel = new JLabel("Blood Group: "); bloodGroupLabel.setFont(new Font("Tahoma", Font.BOLD, 16)); bloodGroupLabel.setBounds(18, 309, 228, 74); bloodGroupField = new JTextField();

bloodGroupField.setFont(new Font("Tahoma", Font.PLAIN, 16)); bloodGroupField.setBounds(320, 317, 364, 59); submitButton = new JButton("Submit"); submitButton.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\save.png")); submitButton.setFont(new Font("Tahoma", Font.BOLD, 16)); submitButton.setBounds(40, 415, 196, 74); getContentPane().setLayout(null);

getContentPane().add(nameLabel); getContentPane().add(nameField); getContentPane().add(emailLabel); getContentPane().add(emailField); getContentPane().add(bloodGroupLabel); getContentPane().add(bloodGroupField); getContentPane().add(submitButton);

btnClose = new JButton("Close");

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

}

});

btnClose.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

btnClose.setFont(new Font("Tahoma", Font.BOLD, 16)); btnClose.setBounds(352, 415, 196, 74); getContentPane().add(btnClose);

separator = new JSeparator(); separator.setBounds(0, 103, 702, 38); getContentPane().add(separator); separator\_1 = new JSeparator(); separator\_1.setBounds(-113, 393, 702, 38); getContentPane().add(separator\_1);

lblNewLabel\_1 = new JLabel("ADD NEW DONAR"); lblNewLabel\_1.setFont(new Font("Algerian", Font.BOLD, 26)); lblNewLabel\_1.setBounds(172, 41, 330, 38); getContentPane().add(lblNewLabel\_1);

lblNewLabel = new JLabel("New label");

lblNewLabel.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background image.png"));

lblNewLabel.setBounds(0, 10, 694, 512); getContentPane().add(lblNewLabel);

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { saveDonorInformation();

}

});

setVisible(true);

}

private void saveDonorInformation() { String name = nameField.getText();

String email = emailField.getText();

String bloodGroup = bloodGroupField.getText()

Connection conn = null;

PreparedStatement stmt = null; try {

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root", "devi");

String sql = "INSERT INTO Donors (donor\_name, donor\_email, blood\_group) VALUES (?, ?, ?)"; stmt = conn.prepareStatement(sql); stmt.setString(1, name); stmt.setString(2, email); stmt.setString(3, bloodGroup);

stmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Donor information saved successfully.");

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

JOptionPane.showMessageDialog(this, "Error: Unable to save donor information."); } finally { try {

if (stmt != null) { stmt.close();

}

if (conn != null) { conn.close();

}

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

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new addNewDonar());

}

}

**ALL DONOR.JAVA**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*; import java.awt.event.\*; import java.sql.\*;

public class alldonar extends JFrame { private JTable donorTable; private DefaultTableModel tableModel;

public alldonar() { setTitle("Donor Details"); setSize(705, 518);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

tableModel = new DefaultTableModel(); donorTable = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(donorTable); scrollPane.setBounds(10, 71, 673, 326);

JButton printButton = new JButton("Print"); printButton.setBounds(131, 417, 112, 42);

printButton.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\print.png")); printButton.setFont(new Font("Tahoma", Font.BOLD, 14)); getContentPane().setLayout(null); getContentPane().add(scrollPane); getContentPane().add(printButton);

JLabel lblNewLabel = new JLabel("All Donor Details"); lblNewLabel.setBounds(200, 23, 344, 42);

lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 26)); getContentPane().add(lblNewLabel);

JButton printButton\_2 = new JButton("Close"); printButton\_2.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { setVisible(false);

}

});

printButton\_2.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

printButton\_2.setFont(new Font("Tahoma", Font.BOLD, 14)); printButton\_2.setBounds(368, 417, 112, 42); getContentPane().add(printButton\_2);

JLabel lblNewLabel\_1 = new JLabel("New label"); lblNewLabel\_1.setBounds(0, -29, 776, 531);

lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background image.png"));

getContentPane().add(lblNewLabel\_1);

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

donorTable.print(); // Prints the table content } catch (java.awt.print.PrinterException ex) { ex.printStackTrace();

}

}

});

populateDonorsTable();

setVisible(true);

}

private void populateDonorsTable() {

Connection conn = null; Statement stmt = null; try {

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms",

"root", "devi");

stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT donor\_id, donor\_name, donor\_email, blood\_group FROM Donors");

ResultSetMetaData metaData = rs.getMetaData(); int columnCount = metaData.getColumnCount();

for (int i = 1; i <= columnCount; i++) {

tableModel.addColumn(metaData.getColumnName(i));

}

while (rs.next()) {

Object[] row = new Object[columnCount]; for (int i = 1; i <= columnCount; i++) { row[i - 1] = rs.getObject(i);

}

tableModel.addRow(row);

}

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

} finally { try { if (stmt != null) { stmt.close();

} if (conn != null) { conn.close();

}

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

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new alldonar());

}

}

**HOME.JAVA**

import java.awt.EventQueue;

import javax.swing.JFrame; import javax.swing.JPanel; import javax.swing.border.EmptyBorder; import javax.swing.JMenuBar; import javax.swing.JMenu; import javax.swing.JMenuItem; import javax.swing.JOptionPane; import javax.swing.JCheckBoxMenuItem; import javax.swing.ImageIcon; import java.awt.event.ItemListener; import java.awt.event.ItemEvent; import java.awt.Font; import javax.swing.JLabel; import java.awt.event.ActionListener; import java.awt.event.ActionEvent; import java.awt.Color;

public class home extends JFrame {

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

/\*\*

\* Launch the application. \*/

public static void main(String[] args) {

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

try {

home frame = new home(); frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public home() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setBounds(100, 100, 1366, 853); contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JMenuBar menuBar = new JMenuBar(); menuBar.setBounds(0, 0, 1555, 71); contentPane.add(menuBar);

JMenu mnNewMenu = new JMenu("Donor");

mnNewMenu.setFont(new Font("SansSerif",

Font.BOLD, 16));

mnNewMenu.addItemListener(new ItemListener() { public void itemStateChanged(ItemEvent e) {

}

});

mnNewMenu.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Donor.png")); menuBar.add(mnNewMenu);

JMenuItem mntmNewMenuItem\_1 = new

JMenuItem("Add New");

mntmNewMenuItem\_1.addActionListener(new

ActionListener() {

public void actionPerformed(ActionEvent e) {

new addNewDonar().setVisible(true);

}

});

mntmNewMenuItem\_1.setFont(new Font("SansSerif",

Font.PLAIN, 14));

mntmNewMenuItem\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Add new.png"));

mnNewMenu.add(mntmNewMenuItem\_1);

JMenuItem mntmNewMenuItem = new

JMenuItem("Update");

|  |  |
| --- | --- |
| ActionListener() { | mntmNewMenuItem.addActionListener(new |
|  | public void actionPerformed(ActionEvent e) { |
|  | new updateDonar().setVisible(true); |
|  | } |
|  | }); |
| Font.PLAIN, 14)); | mntmNewMenuItem.setFont(new Font("SansSerif", |
|  | mntmNewMenuItem.setIcon(new |

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Update details.png"));

mnNewMenu.add(mntmNewMenuItem);

JMenu mnNewMenu\_1 = new JMenu("Search Blood

Donar");

mnNewMenu\_1.setFont(new Font("SansSerif",

Font.BOLD, 16));

mnNewMenu\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\search user.png"));

menuBar.add(mnNewMenu\_1);

JMenuItem mntmNewMenuItem\_4 = new

JMenuItem("Blood Group");

|  |  |
| --- | --- |
| ActionListener() { | mntmNewMenuItem\_4.addActionListener(new |
|  | public void actionPerformed(ActionEvent e) { |
|  | new BloodGroupSearch().setVisible(true); |
|  | } |
|  | }); |
| Font.PLAIN, 14)); | mntmNewMenuItem\_4.setFont(new Font("SansSerif", |
|  | mntmNewMenuItem\_4.setIcon(new |

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Blood group.png"));

mnNewMenu\_1.add(mntmNewMenuItem\_4);

JMenu mnNewMenu\_2 = new JMenu("Details"); mnNewMenu\_2.setFont(new Font("SansSerif",

Font.BOLD, 16));

mnNewMenu\_2.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\stock.png"));

menuBar.add(mnNewMenu\_2);

JMenuItem mntmNewMenuItem\_2\_1 = new

JMenuItem("All Donar Details");

mntmNewMenuItem\_2\_1.addActionListener(new

ActionListener() {

public void actionPerformed(ActionEvent e) { new alldonar().setVisible(true);

}

});

mntmNewMenuItem\_2\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Details.png"));

|  |  |
| --- | --- |
| Font.PLAIN, 14)); | mntmNewMenuItem\_2\_1.setFont(new Font("SansSerif", |
|  | mnNewMenu\_2.add(mntmNewMenuItem\_2\_1); |
|  |  |
|  | JMenu mnNewMenu\_3 = new JMenu("Delete Donar"); |
| Font.BOLD, 16)); | mnNewMenu\_3.setFont(new Font("SansSerif", |

mnNewMenu\_3.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete donor.png"));

menuBar.add(mnNewMenu\_3);

JMenuItem mntmNewMenuItem\_8 = new

JMenuItem("Delete Donar");

|  |  |
| --- | --- |
| ActionListener() { | mntmNewMenuItem\_8.addActionListener(new |
|  | public void actionPerformed(ActionEvent e) { |
|  | new DonorDeletionFrame().setVisible(true); |
|  | } |
|  |  |
|  | }); |
| Font.PLAIN, 14)); | mntmNewMenuItem\_8.setFont(new Font("SansSerif", |
|  | mntmNewMenuItem\_8.setIcon(new |

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete.png"));

mnNewMenu\_3.add(mntmNewMenuItem\_8);

JMenu mnNewMenu\_4 = new JMenu("Exit");

mnNewMenu\_4.setFont(new Font("SansSerif",

Font.BOLD, 16));

mnNewMenu\_4.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\exit.png")); menuBar.add(mnNewMenu\_4);

JMenuItem mntmNewMenuItem\_9 = new

JMenuItem("Logout");

mntmNewMenuItem\_9.addActionListener(new

ActionListener() {

public void actionPerformed(ActionEvent e) {

int

a=JOptionPane.showConfirmDialog(null,"Do you really want to logout","Select",JOptionPane.YES\_NO\_OPTION);

if(a==0)

{

setVisible(false);

new login().setVisible(true);

}

}

});

mntmNewMenuItem\_9.setFont(new Font("SansSerif",

Font.PLAIN, 14));

mntmNewMenuItem\_9.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Logout.png"));

mnNewMenu\_4.add(mntmNewMenuItem\_9);

JMenuItem mntmNewMenuItem\_10 = new

JMenuItem("Exit Application");

mntmNewMenuItem\_10.addActionListener(new

ActionListener() {

public void actionPerformed(ActionEvent e) {

int

a=JOptionPane.showConfirmDialog(null,"Do you really want to Close the

Application","Select",JOptionPane.YES\_NO\_OPTION);

if(a==0)

{

System.exit(0);

}

}

});

mntmNewMenuItem\_10.setFont(new Font("SansSerif",

Font.PLAIN, 14));

mntmNewMenuItem\_10.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

mnNewMenu\_4.add(mntmNewMenuItem\_10);

JLabel lblNewLabel\_2\_1 = new JLabel("New label"); lblNewLabel\_2\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\blood

10.jpg"));

lblNewLabel\_2\_1.setBounds(519, 231, 188, 163); contentPane.add(lblNewLabel\_2\_1);

JLabel lblNewLabel\_2\_1\_1\_1 = new JLabel("“Be the reason someone smiles today");

lblNewLabel\_2\_1\_1\_1.setForeground(new Color(255,

255, 255));

lblNewLabel\_2\_1\_1\_1.setFont(new Font("Algerian",

Font.PLAIN, 55));

lblNewLabel\_2\_1\_1\_1.setBounds(10, 462, 1332, 195); contentPane.add(lblNewLabel\_2\_1\_1\_1); JLabel lblNewLabel\_2 = new JLabel("New label"); lblNewLabel\_2.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\g5.jpg"));

lblNewLabel\_2.setBounds(769, 168, 413, 240); contentPane.add(lblNewLabel\_2); JLabel lblNewLabel\_2\_1\_1 = new JLabel("");

lblNewLabel\_2\_1\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\b19.jpg"));

lblNewLabel\_2\_1\_1.setBounds(351, 168, 372, 240); contentPane.add(lblNewLabel\_2\_1\_1);

JLabel lblNewLabel\_2\_1\_1\_1\_1 = new JLabel("Donate

Blood\"");

lblNewLabel\_2\_1\_1\_1\_1.setForeground(Color.WHITE);

lblNewLabel\_2\_1\_1\_1\_1.setFont(new Font("Algerian",

Font.PLAIN, 55));

lblNewLabel\_2\_1\_1\_1\_1.setBounds(905, 576, 1035,

163);

contentPane.add(lblNewLabel\_2\_1\_1\_1\_1);

JLabel lblNewLabel = new JLabel("New

label");

lblNewLabel.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\background1.j pg"));

lblNewLabel.setBounds(772, -163, 1943, 1170); contentPane.add(lblNewLabel);

JLabel lblNewLabel\_1 = new JLabel("New label"); lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\back 2.jpg"));

lblNewLabel\_1.setBounds(10, 10, 1906, 1170); contentPane.add(lblNewLabel\_1);

}

}

**LOGIN.JAVA**

import java.awt.EventQueue;

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

import javax.swing.JOptionPane;

import java.awt.Font; import javax.swing.ImageIcon; import java.awt.Color; import javax.swing.JTextField; import javax.swing.JPasswordField; import javax.swing.JButton; import java.awt.event.ActionListener; import java.awt.event.ActionEvent; import javax.swing.UIManager;

public class login extends JFrame {

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

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() { public void run() {

try {

login frame = new login(); frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public login() {

setBackground(new Color(240, 240, 240)); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setBounds(100, 100, 1487, 882); contentPane = new JPanel(); contentPane.setToolTipText(""); contentPane.setBorder(new EmptyBorder(5, 5, 5, 5)); setContentPane(contentPane); contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("UserName");

lblNewLabel.setFont(new Font("Serif", Font.BOLD,

28));

lblNewLabel.setBounds(571, 209, 126, 28); contentPane.add(lblNewLabel); textField = new JTextField();

textField.setFont(new Font("Serif", Font.PLAIN, 20)); textField.setBounds(758, 200, 268, 54); contentPane.add(textField); textField.setColumns(10);

JLabel lblNewLabel\_2 = new JLabel("password");

lblNewLabel\_2.setFont(new Font("Serif", Font.BOLD,

28));

lblNewLabel\_2.setBounds(571, 321, 126, 37); contentPane.add(lblNewLabel\_2); passwordField = new JPasswordField(); passwordField.setFont(new Font("Serif", Font.PLAIN,

20));

passwordField.setBounds(758, 319, 268, 48); contentPane.add(passwordField);

JButton btnNewButton = new JButton("login"); btnNewButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

if(textField.getText().equals("bbms")&&passwordField.getText

().equals("admin")) {

setVisible(false); new home().setVisible(true);

}

else

JOptionPane.showMessageDialog(null,"Incorrect Username or

Password");

}

});

btnNewButton.setFont(new Font("Serif", Font.BOLD,

26));

btnNewButton.setBounds(480, 492, 160, 62); contentPane.add(btnNewButton);

JButton btnNewButton\_1 = new JButton("close");

btnNewButton\_1.addActionListener(new

ActionListener() {

public void actionPerformed(ActionEvent e) {

int

a=JOptionPane.showConfirmDialog(null,"Do you really want to close

Application","Select",JOptionPane.YES\_NO\_OPTION);

if(a==0)

System.exit(0);

}

});

btnNewButton\_1.setFont(new Font("Serif", Font.BOLD,

26));

btnNewButton\_1.setBounds(920, 492, 168, 62); contentPane.add(btnNewButton\_1);

JLabel lblNewLabel\_1\_2 = new JLabel("\"Give the gift of life");

lblNewLabel\_1\_2.setForeground(new Color(255, 255,

255));

lblNewLabel\_1\_2.setFont(new Font("Algerian",

Font.PLAIN, 45));

lblNewLabel\_1\_2.setIcon(null); lblNewLabel\_1\_2.setBackground(new Color(240, 240,

240));

lblNewLabel\_1\_2.setBounds(304, 541, 500, 218); contentPane.add(lblNewLabel\_1\_2);

JLabel lblNewLabel\_1\_2\_1 = new JLabel("Donate

Blood\"");

lblNewLabel\_1\_2\_1.setForeground(Color.WHITE);

lblNewLabel\_1\_2\_1.setFont(new Font("Algerian",

Font.PLAIN, 45));

lblNewLabel\_1\_2\_1.setBackground(UIManager.getColor("Butt on.background"));

lblNewLabel\_1\_2\_1.setBounds(720, 627, 500, 218); contentPane.add(lblNewLabel\_1\_2\_1); JLabel lblNewLabel\_1\_1\_1 = new JLabel("");

lblNewLabel\_1\_1\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\b19.jpg"));

lblNewLabel\_1\_1\_1.setBackground(UIManager.getColor("Butt on.background"));

lblNewLabel\_1\_1\_1.setBounds(355, -137, 703, 794); contentPane.add(lblNewLabel\_1\_1\_1); JLabel lblNewLabel\_1\_1 = new JLabel("");

lblNewLabel\_1\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\back 3.jpg")); lblNewLabel\_1\_1.setBackground(UIManager.getColor("Button

.background"));

lblNewLabel\_1\_1.setBounds(786, -147, 843, 1065); contentPane.add(lblNewLabel\_1\_1); JLabel lblNewLabel\_1 = new JLabel("");

lblNewLabel\_1.setBackground(new Color(240, 240,

240));

lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\back 3.jpg"));

lblNewLabel\_1.setBounds(10, -73, 1186, 1065); contentPane.add(lblNewLabel\_1);

}

}

**UPDATE DONOR.JAVA**

import java.awt.EventQueue; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.SQLException;

import javax.swing.ImageIcon; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JOptionPane; import javax.swing.JPanel; import javax.swing.JTextField; import javax.swing.SwingUtilities; import javax.swing.border.EmptyBorder;

import Project.ConnectionProvider; import java.awt.Font; import javax.swing.JLabel; import javax.swing.JSeparator;

public class updateDonar extends JFrame {

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

/\*\*

* Launch the application.

\*/

/\*\*

* Create the frame.

\*/

private JButton updateButton;

private JTextField textField; private JLabel emailLabel; private JTextField textField\_1; private JLabel bloodGroupLabel; private JTextField textField\_2; private JLabel lblNewLabel\_1;

public updateDonar() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setBounds(100, 100, 710, 541); contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane); contentPane.setLayout(null); updateButton = new JButton("Update"); updateButton.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Details.png"));

// Change the path to your icon

updateButton.setFont(new Font("Tahoma", Font.BOLD,

16));

updateButton.setBounds(62, 423, 156, 68); getContentPane().add(updateButton);

JLabel nameLabel = new JLabel("Name: ");

nameLabel.setFont(new Font("Tahoma", Font.BOLD,

16));

nameLabel.setBounds(37, 83, 134, 111); contentPane.add(nameLabel);

textField = new JTextField();

textField.setFont(new Font("Tahoma", Font.PLAIN, 16)); textField.setBounds(299, 104, 372, 68); contentPane.add(textField);

emailLabel = new JLabel("Email: ");

emailLabel.setFont(new Font("Tahoma", Font.BOLD,

16));

emailLabel.setBounds(37, 197, 140, 111);

contentPane.add(emailLabel);

textField\_1 = new JTextField();

textField\_1.setFont(new Font("Tahoma", Font.PLAIN,

16));

textField\_1.setBounds(299, 218, 372, 68); contentPane.add(textField\_1);

bloodGroupLabel = new JLabel("Blood Group: ");

bloodGroupLabel.setFont(new Font("Tahoma",

Font.BOLD, 16));

bloodGroupLabel.setBounds(37, 302, 140, 111); contentPane.add(bloodGroupLabel);

textField\_2 = new JTextField();

textField\_2.setFont(new Font("Tahoma", Font.PLAIN,

16));

textField\_2.setBounds(299, 323, 372, 68); contentPane.add(textField\_2);

JButton btnClose = new JButton("Close"); btnClose.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));

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

} });

btnClose.setFont(new Font("Tahoma", Font.BOLD, 16)); btnClose.setBounds(363, 423, 169, 68); contentPane.add(btnClose);

JSeparator separator = new JSeparator(); separator.setBounds(10, 405, 721, 31); contentPane.add(separator);

JSeparator separator\_1 = new JSeparator(); separator\_1.setBounds(10, 83, 721, 21); contentPane.add(separator\_1);

JLabel lblNewLabel = new JLabel("UPDATE DONAR DETAILS");

lblNewLabel.setFont(new Font("Algerian", Font.BOLD,

28));

lblNewLabel.setBounds(163, 20, 350, 53);

contentPane.add(lblNewLabel);

lblNewLabel\_1 = new JLabel("New label"); lblNewLabel\_1.setIcon(new

ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background image.png"));

lblNewLabel\_1.setBounds(0, -99, 769, 707); contentPane.add(lblNewLabel\_1);

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) { updateDonorInformation();

}

});

setVisible(true);

}

private void updateDonorInformation() {

String name = textField.getText();

String email = textField\_1.getText();

String bloodGroup = textField\_2.getText();

Connection conn = null;

PreparedStatement stmt = null; try {

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root", "devi");

String sql = "UPDATE Donors SET donor\_email = ?, blood\_group = ? WHERE donor\_name = ?";

stmt = conn.prepareStatement(sql);

stmt.setString(1, email); stmt.setString(2, bloodGroup); stmt.setString(3, name);

int rowsUpdated = stmt.executeUpdate();

if (rowsUpdated > 0) {

JOptionPane.showMessageDialog(this, "Donor information updated successfully.");

} else {

JOptionPane.showMessageDialog(this, "Error: Unable to update donor information.");

}

}

catch (SQLException ex) { ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Error: Unable to update donor information.");

} finally { try {

if (stmt != null) { stmt.close();

}

if (conn != null) { conn.close();

}

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

}

}

}

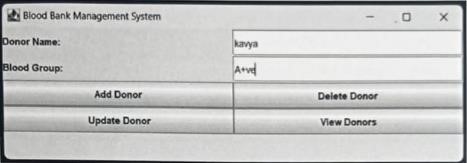
public static void main(String[] args) { SwingUtilities.invokeLater(() -> new updateDonar());

}

}

## 4.3 OUTPUT IMAGES

**ADDING DONOR AND VIEWING DONOR DETAILS:**



### FIG.4.3.1 ADDING DONOR DETAIL



FIG.4.3.2 DONOR DETAIL ADDED SUCCESSFULLY

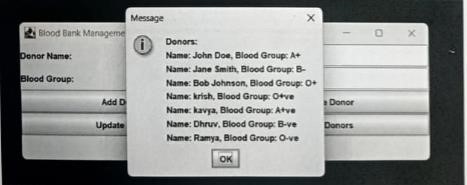
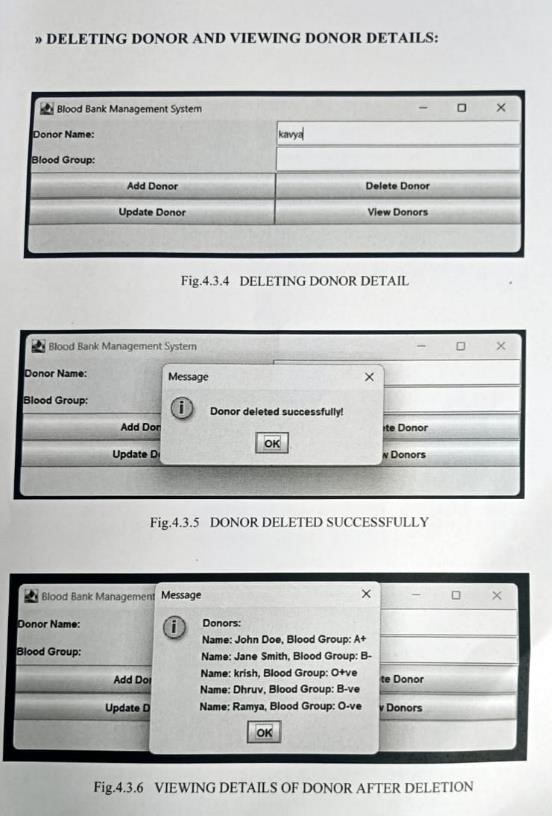
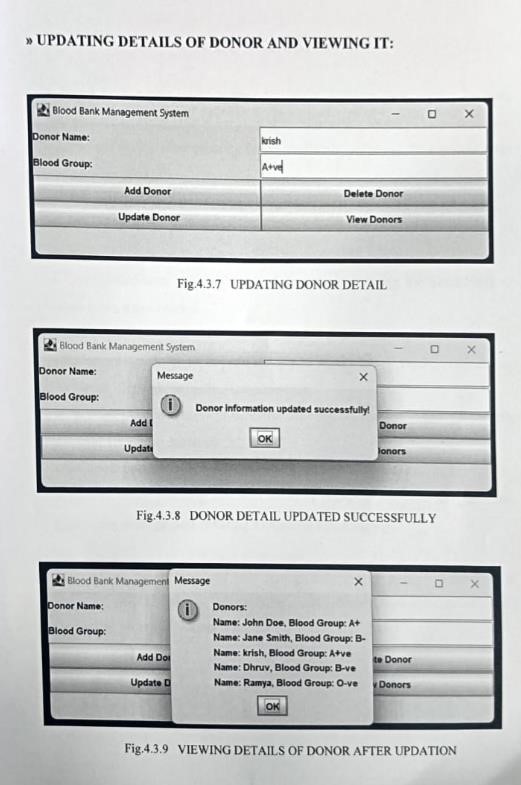


FIG.4.3.3 VIEWING DETAILS OF DONOR AFTER ADDITION





# 5. CONCLUSION

A blood bank management system facilitates efficient coordination between donors, recipients, and blood banks. It streamlines the process of blood donation, storage, and distribution, ensuring timely availability of blood. This system helps in:

Allows users to find blood donors and blood banks easily, especially in emergencies, by providing a centralized platform.

Enables real-time tracking of blood inventory, ensuring the availability of blood types when needed.

Enhancing the overall efficiency of the blood donation process.

Enhances the ability to respond quickly to urgent blood requirements during accidents, disasters, or medical emergencies.

Facilitates the management of donor information and maintaining a reliable database for future donations.

Enhances transparency and helps in building trust among donors and recipients.

Allows better planning and utilization of resources by predicting demand, preventing wastage, and ensuring a balanced blood supply.

In summary, an online blood bank management system plays a crucial role in ensuring the availability, accessibility, and efficient management of blood resources, ultimately saving lives.