

**BLOOD BANK MANAGEMENT SYSTEM**

# CS23333 – Object Oriented Programming using Java Project Report

*Submitted by*

**SWETHA A - 231001226**

**HARSHINI T - 231001059**

*Of*

**BACHELOR OFTECHNOLOGY**

*In*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE**

**NOVEMBER-2024**

# BONAFIDECERTIFICATE

Certified that this project titled “BLOOD BANK MANAGEMENT SYSTEM” is the Bonafide work of “ **SWETHA A (231001226) ,HARSHINI T(231001059)”**who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.PVALARMATHI**  **HEAD OFTHE DEPARTMENT**  **DEPARTMENT OF INFORMATION TECHNOLOGY**  **RAJALAKSHMI ENGINEERING COLLEGE THANDALAM**  **CHENNAI-602105** | **SIGNATURE:** |

This project is submitted for CS23333 – Object Oriented Programming using Java held on

## INTERNAL EXAMINAR EXTERNAL EXAMINAR

**Mr.Narayana K.E**

**Assistant Professor,**

**Information Technology**

RAJALAKSHMI ENGINEERING COLLEGE

(Autonomous),

THANDALAM, CHENNAI-602105

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| **1.1** | **ABSTRACT** | 5 |
| **1.2** | **INTRODUCTION** | 5 |
| **1.3** | **PURPOSE** | 5 |
| **1.4** | **SCOPE** **OF THE PROJECT** | 6 |
| **1.5** | **SOFTWARE SPECIFICATION** | 6 |
| **1.6** | **OVERALL** **DESCRIPTION** | 7 |
| **2** | **SYSTEM FLOW DIAGRAM** | **13** |
| **2.1** | **CASE DIAGRAM** | 14 |
| **2.2** | **ENTITY RELATIONAL DIAGRAM** | 15 |
| **3** | **MODULEDESCRIPTIONFORBBMS** | 15 |
| **3.1** | **TOOLS/PLATFORM** | 16 |
| **3.2** | **IMPLEMENTATION** | 17 |
| **3.3** | **PROJECTSETUP&CONFIGURATION** | 17 |
| **4** | **INPUT IMAGES** | 17 |
| **4.1** | **DESIGN** | 18 |
| **4.2** | **DATA DESIGN** | 18 |
| **4.3** | **CODING&CONNECTION IN JAVA** | 20 |
| **4.3.1** | **ADDING DONAR DETAILS** | 39 |

## 1. Abstract

The Blood Bank Management System is a software application designed to streamline and automate the processes involved in blood donation, storage, and distribution. The primary aim of this system is to improve the efficiency and accuracy of managing blood inventories, ensuring that blood is available when needed, while also maintaining a database of donors, recipients, and blood types. This system is developed using Java, incorporating both graphical user interface (GUI) and back-end functionality. The system allows blood donors to register, update their details, and schedule donation appointments

## 2. Introduction

The development of the Blood Bank Management System aims to address the challenges faced by blood banks in managing large volumes of data, ensuring the safety and quality of blood, and improving accessibility. By leveraging Java’s object-oriented principles, this project ensures that the system is modular, maintainable, and scalable, providing a robust solution for modern blood bank operations. Additionally, the system offers a user-friendly graphical interface that makes it easy for users to navigate and interact with the application.

## 3. Purpose

Automate blood donation and inventory management to reduce manual errors and improve efficiency.

Ensure accurate tracking of donor information, blood types, and stock levels.

Provide timely blood supply by monitoring and managing blood availability and distribution.

Engage donors by offering a platform for registration, history tracking, and event notifications.

Generate reports for inventory, donor activity, and blood distribution for better decision-making.

**4. Scope of the project**

* Allows donors to register their personal details and medical history.
* Provides an interface for updating donor records and viewing donation history.
* Alerts donors when they are eligible to donate again based on blood donation policies.
* **Blood Inventory Management:**
* Tracks blood donations received and updates inventory levels based on donations.
* Monitors expiration dates of blood units to ensure only safe and viable blood is distributed.
* Helps the system administrator track the stock of different blood types.

## 5. Software Requirement Specification

### Introduction

This system provides an intuitive interface for both donors and administrators to enhance operational efficiency, accuracy, and transparency in blood bank operations. It helps blood banks efficiently track donor information, manage blood stocks, and ensure timely delivery to hospitals.

### Product Scope

The Blood Bank Management System automates the processes of blood donation, inventory tracking, and distribution. It enables efficient management of donor information, blood stock, and donation schedules, while providing secure data handling. The system also includes reporting features for performance monitoring and decision-making.

### References andAcknowledgement

**Online course**:

1.Oracle, Java Documentation. Available at: <https://docs.oracle.com/javase/8/docs/>

2.W3Schools, Java Tutorials. Available at: https://www.w3schools.com/java/

**Books:**

* Sommerville, I. (2011). *Software Engineering* (9th ed.). Boston: Addison-Wesley.
* Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.

**1.Acknowledgements:**

* + We would like to express our sincere gratitude to our project supervisor for their guidance and support throughout the development of this project.
  + Special thanks to the developers, designers, and contributors whose work and research were foundational in shaping the system's architecture and functionality.
  + We also acknowledge the importance of online resources, textbooks, and Java documentation for providing the essential knowledge and tools required for successful project implementation

### Overall Description

The **Blood Bank Management System** is a comprehensive software solution designed to automate and manage the various functions of a blood bank, from donor registration and inventory management to blood distribution. Built using Java, the system aims to streamline the management of blood donations, track blood stock levels, and ensure efficient distribution to hospitals in need. The use of Java ensures that the system is scalable, secure, and can be easily maintained. The system aims to optimize the management of blood donations, inventory, and distribution, contributing to more effective healthcare services and ensuring a reliable blood supply for those in need.

### Product Perspective

The system integrates with a relational database (such as MySQL, SQLite, or similar) to store critical data such as donor details, blood types, donation history, and blood inventory. The database provides the foundation for data management, ensuring secure, structured storage and retrieval of information. The Blood Bank Management System is designed to function as an independent application on a local network or a single machine. It can be used by blood banks of any size, from small clinics to larger regional facilities.

### Product Functionality:

**Donor Registration and Profile Management:**

Donorscan update their profile information as needed.

**Eligibility Check:**

* Automatically checks donor eligibility based on donation history (e.g., the time interval between donations) and notifies the donor when they are eligible to donate again.

**Expiration Monitoring:**

* The system monitors blood expiration dates and alerts administrators when blood units are nearing expiration, ensuring that expired blood is discarded.

**Blood Request Handling:**

* Hospitals or healthcare facilities can submit blood requests based on urgent needs or scheduled surgeries.

**Inventory Reports:**

* Generate detailed reports on current blood inventory levels, including the amount of each blood type available and near expiration.

**Admin Role:**

Full access to all system features, including managing donors, blood inventory, donation scheduling, and generating reports.

* **Donor Role:**

Donors can access their profile, donation history, and schedule appointments, but have no access to system administration functions.

* **Staff Role:**

Staff members can manage blood inventory, process blood requests, and help with scheduling donation camps but have limited access to sensitive data.

### User and Characteristics:

Qualification: Users should be comfortable reading and understanding English, as the system interface and documentation will primarily be in English.

Experience:Familiarity with medical or healthcare processes, such as blood donation or inventory management, is advantageous.

Experience in interacting with computer systems and online platforms is beneficial but not mandatory for basic users.

### Operating Environment

#### Hardware Requirements

**Processor:** Intel i3 or higher (or equivalent AMD processor).

**Operating System:** Windows 8, 10, or 11.

**Processor Speed:** Minimum 2.0 GHz.

**RAM:** Minimum 4GB.

**Hard Disk:** Minimum 500GB of storage available for system files and data.

#### Software Requirements

**Database:** MySQL (used for data storage and management).

**Frontend Technology:** Java Swing or JavaFX (for the user interface).

**Backend Technology:** Java (with JDBC for database connectivity).

**Web Server (optional):** Apache Tomcat (for running the JSP interface, if required).

#### Constraints

* The system is designed to handle moderate volumes of data and users, suitable for typical blood bank operations. It will effectively manage donor records, blood stock levels, and blood distribution without performance issues under normal usage conditions.

### User Interface

The **Blood Bank Management System** offers an intuitive, menu-driven interface with the following features:

* **Register:**
  + Allows new users (donors) to register by entering their personal information, blood type, and contact details.
* **Login:**
  + Existing users (donors, administrators) can log in securely with their credentials to access their respective accounts and perform actions.
* **Donor Dashboard:**
  + Donors can view their donation history, schedule appointments, and see eligibility for future donations.
* **Blood Inventory Management:**
  + Administrators can access the inventory page to track the available blood types, quantities, and expiration dates.
* **Order/Request Blood:**
  + Hospitals or healthcare facilities can view available blood types and place requests for blood based on their needs.
* **Reports and Analytics:**
  + Admins can generate reports on blood donations, inventory levels, and donor statistics for better decision-making.

### Hardware Interface:

### Screen Resolution:

### The system is optimized to function properly with a minimum screen resolution of 640 x 480, though higher resolutions are recommended for better display.

### Operating System Compatibility:

### Compatible with any version of Windows 8, 10, or 11 for smooth system operation and user experience.

### Software Interface:

### Operating System:

### MS-Windows (8, 10, 11) is required to run the Blood Bank Management System.

### Frontend Technology:

### JSP (Java Server Pages) is used for designing the user interface, allowing for a dynamic, responsive web interface.

### Backend Technology:

### Java is used for the backend processing and business logic of the system, ensuring robust and scalable operations.

### Database:

### MySQL is used for data storage, managing donor records, blood inventory, and transaction logs securely and efficiently

**Functional Requirements:**

**1. User Registration and Authentication:**

* **User Registration:**
  + Allow users (donors, administrators, and hospital staff) to register with their details such as name, email, phone number, blood type, address, and medical history for donors.
* **Authentication:**
  + Provide secure login/logout functionality with encrypted passwords, ensuring that only authorized users can access the system.

**2. Donor Management:**

* **Donor Profile Management:**
  + Allow donors to update their profile information (contact details, medical history, eligibility status for donation).
* **Donation History:**
  + Track the donation history for each donor, including blood type, donation dates, and eligibility for future donations.

**3. Blood Inventory Management:**

* **Inventory Management:**
  + Track and display blood types, quantities, and expiration dates of donated blood in real-time.

**4. Blood Request and Distribution:**

* **Blood Request:**
  + Allow hospitals and healthcare facilities to request blood by specifying required blood type and quantity.
* **Blood Distribution:**
  + Admins can approve and track blood distribution to healthcare facilities and hospitals based on available inventory.

**5. Reports and Analytics:**

* **Generate Reports:**
  + Admins should be able to generate real-time reports, such as donor participation, blood inventory levels, donation trends, and blood usage.
* **Analytics:**
  + Analyze donation trends, blood stock usage, and demand from hospitals to optimize donation drives and inventory management.

**6. Admin Functions:**

* **Manage Donors:**
  + Admins can add, update, or remove donor records, ensuring that donor data is up-to-date and accurate.

### ****Non-functional Requirements:****

#### ****1. Performance:****

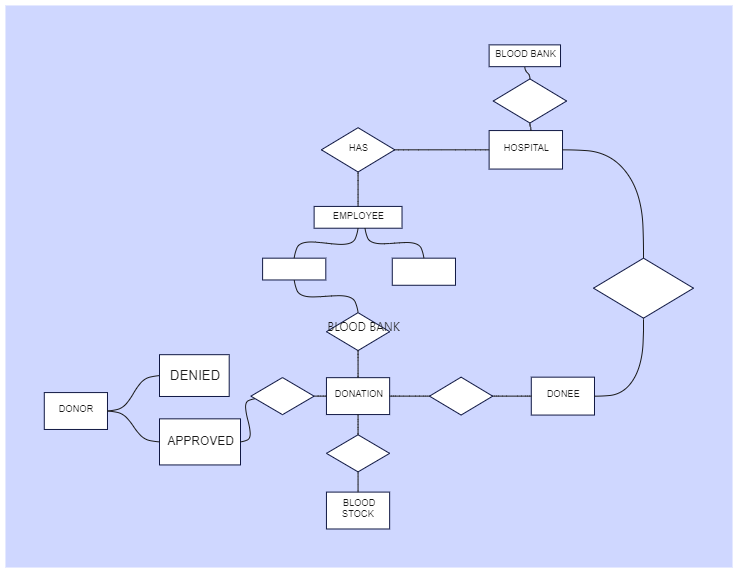
* The system should load quickly (within 2-3 seconds) and perform efficiently even when handling a moderate volume of blood donations, donor registrations, and hospital requests.

#### ****2. Scalability:****

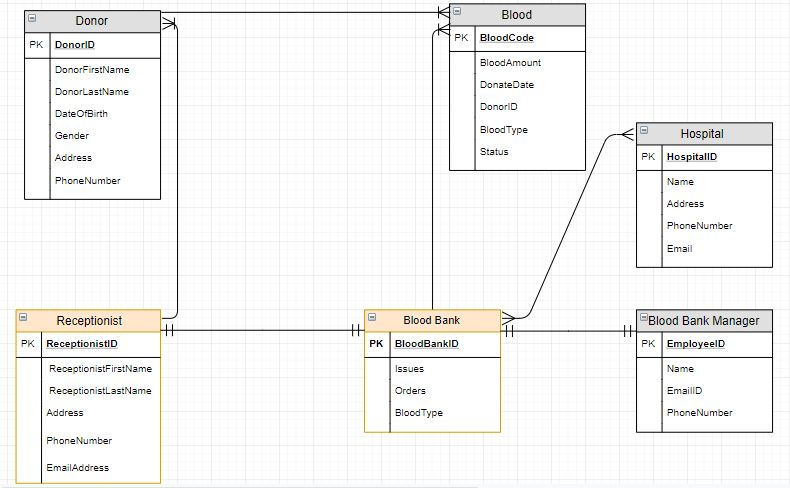
* The system should be able to handle increasing numbers of users, blood donations, and requests from healthcare facilities without performance degradation.

## 2.SYSTEM FLOW DIAGRAM

**Figure 2.1 Use Case Diagrams**

****

### Figure 2.2 Entity Relationship Diagram



**Figure 2.3 Data-flow diagram**

**3.Module Description for Blood Bank Management System:**

User Management: Handles user registration, login, profile management, and authentication for donors, admins, and hospital staff

## Blood Inventory Management: Manages tracking, updating, and monitoring of blood stocks, including blood type, quantity, and expiration dates.

## Donation Management: Oversees the scheduling of blood donations, donor eligibility checks, and logging of donation history.

## Blood Request & Distribution Management: Allows hospitals to request blood, and admins to validate, approve, and distribute blood.

## Reporting and Analytics: Generates reports and provides analytics on blood donations, inventory levels, and hospital blood requests.

## Admin Dashboard: Provides admins with tools to manage users, track inventory, process blood requests, and generate reports.

## 3.1 TOOLS /PLATFORM

**Programming Language:** Java for backend development.

**IDE:** Eclipse, IntelliJ IDEA, or NetBeans for Java development.

**Database Management:** MySQL for relational database management.

**Frontend Technology:** JSP for dynamic web page creation.

**Web Server:** Apache Tomcat for serving Java-based web applications.

**Frameworks:** Spring Framework for backend development, Hibernate ORM for database object-relational mapping.

**Version Control:** Git for version control, GitHub/GitLab for remote code repositories and collaboration.

**Project Management:** JIRA, Trello, or Asana for task management and project tracking.

**Testing Tools:** JUnit for unit testing, Selenium for automated functional testing.

**Deployment Platforms:** AWS, Heroku, or DigitalOcean for cloud hosting and deployment.

**Security Tools:** SSL/TLS for secure data transmission, OWASP ZAP for vulnerability scanning.

**Documentation Tools:** Swagger or Postman for API documentation and testing, Confluence for team documentation.

**User Interface Design Tools:** Figma, Adobe XD, or Sketch for designing wireframes and UI mockups.

**Communication Tools:** Slack for team collaboration and communication.

**Continuous Integration/Continuous Deployment (CI/CD):** Jenkins or CircleCI for automating testing, integration, and deployment processes.

**Containerization:** Docker for containerizing the application for easier deployment and scalability.

**Monitoring Tools:** Prometheus or New Relic for monitoring the application’s performance and health in production.

**Backup and Recovery:** AWS RDS Automated Backups, or MySQL Workbench for database backup and restoration.

**Search Functionality:** Apache Solr or Elasticsearch for integrating efficient search functionalities into the system.

**Analytics Tools:** Google Analytics for tracking user activity and behavior on the platform.

**3.2 IMPLEMENTATION**

Implementation for Blood Bank Management System

The implementation of the Blood Bank Management System involves building a complete system that allows users (donors, hospitals, and admins) to manage blood donations, requests, inventory, and more. Below is a structured approach for implementing the system:

**3.3 Project Setup & Environment Configuration**

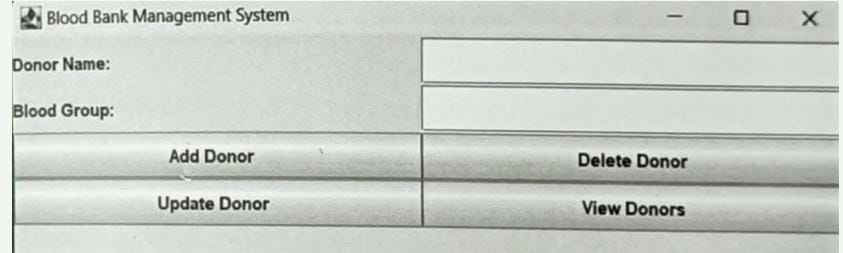
* **Tools Used:**
  + Java: Backend development using Java for business logic and APIs.
  + MySQL: Setting up the relational database to manage donor details, inventory, blood requests, and admin data.
  + JSP & Servlets: For building the frontend and backend web pages.
  + Apache Tomcat: For hosting and running the web application.

### ****Backend Development (Java & Spring Framework)****

* **Steps:**
  + **Set up the Spring Framework** to handle backend logic and business processes.
  + Develop **Java classes** to implement core functionalities, including:

**User Authentication & Authorization**: Implement login, logout, and session management using Spring Security

**4. INPUT IMAGES:**



### 4.1 Design



### 4.2 DATA DESIGN:

**1)Donor Table:**

It stores the details of blood donors, including personal information and their donation history.

* **Table Name:** donors
* **Columns:**
  + donor\_id (Primary Key): A unique identifier for each donor.
  + name: The full name of the donor.
  + email: The email address of the donor.
  + phone: Contact number of the donor.
  + blood\_type: The blood type of the donor (e.g., A+, O-, etc.).
  + last\_donation\_date: The date of the most recent blood donation.
  + eligible\_for\_donation: A flag (Boolean) indicating if the donor is eligible to donate blood again.

**2)Blood Inventory Table**

The **Blood Inventory Table** holds data on the blood available at the blood bank, including quantities, types, and expiration dates.

**Table Name:**

**Blood\_inventory:**

* **Columns:**
  + inventory\_id (Primary Key): A unique identifier for each blood stock record.
  + blood\_type: The type of blood (A+, B-, O+, etc.).
  + quantity: The amount of blood (usually in liters or pints).
  + donation\_date: The date when the blood was donated.
  + expiry\_date: The date when the blood expires.
  + status: A status indicating whether the blood is available, expired, or used.

**3) Blood Request Table**

The **Blood Request Table** manages requests made by hospitals or healthcare providers for specific blood types and quantities.

* **Table Name:** blood\_requests
* **Columns:**
  + request\_id (Primary Key): A unique identifier for each request.
  + hospital\_name: The name of the hospital or healthcare facility requesting blood.
  + blood\_type: The type of blood requested (e.g., A+, O-, etc.).
  + quantity\_requested: The amount of blood requested.
  + request\_date: The date on which the request was made.
  + status: The status of the request (e.g., Pending, Fulfilled, Cancelled).

**4) Admin Table**

The **Admin Table** stores information related to the system administrators who manage the blood bank operations.

* **Table Name:** admins
* **Columns:**
  + admin\_id (Primary Key): A unique identifier for each admin.
  + name: The name of the admin.
  + email: The admin's email address.
  + password: The encrypted password for secure login.
  + role: The role of the admin (e.g., Super Admin, Inventory Manager, Request Manager).

**4.3 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);

}

});

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();

}

}

}

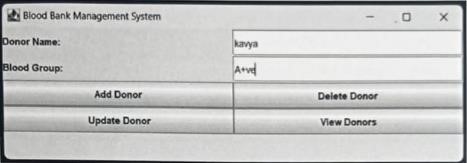
public static void main(String[] args) { SwingUtilities.invokeLater(() -> new updateDonar());

}

}

**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

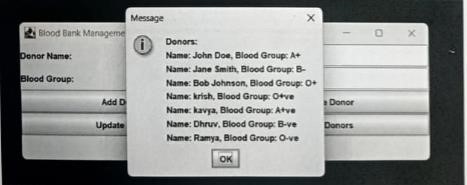
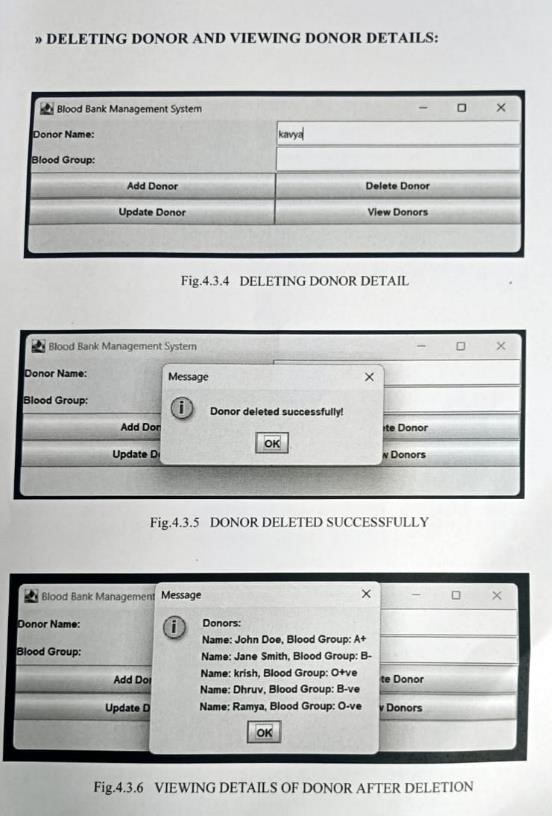


FIG.4.3.3 VIEWING DETAILS OF DONOR AFTER ADDITION



#### 4.4 IMPLEMENTATIONS:

To implement a **Blood Bank Management System (BBMS)** in **Java**, we will go through the steps of **setting up the database**, creating the necessary **Java classes**, and implementing the **backend logic** for managing donors, recipients, inventory, and transactions.

Here is a full implementation of the **Blood Bank Management System** in Java using **JDBC** for database interaction. This implementation will cover essential features such as:

Donor Management

Blood Inventory Management

Recipient Management

Transaction (Transfusion) Management

**Steps for Implementation**

1. **Set up Database**:
   * Install and set up a relational database.
   * Create the required tables as shown in the schema.
2. **Back-End Development**:
   * Implement the business logic and APIs for donor registration, blood inventory management, and transfusions.
3. **Front-End Development**:
   * Create a simple web interface for users to interact with the system.
   * Use JavaScript (AJAX/fetch) to interact with back-end APIs.
4. **Testing**:
   * Test the system for bugs and ensure all features work correctly, like registering donors, updating inventory, and processing transactions.
5. **Deployment**:
   * Deploy the application on a server using a platform like **Heroku**, **AWS**, or **Azure**.
   * Make sure the database is accessible and secure.

**5.CONCLUSION:**

The Blood Bank Management System (BBMS) provides an efficient and organized solution for managing blood donations, blood inventory, and transfusions. It simplifies the process of registering blood donors, tracking blood types and quantities in the inventory, managing recipient requests, and recording transfusion transactions.

By implementing a database-driven approach using Java and MySQL (or any relational database), this system allows for seamless tracking, updating, and reporting of critical information, ensuring the efficient operation of a blood bank**.**

**6.REFERENCE:**

**Books:**

1. [Head First Java by Kathy Sierra and Bert Bates](https://www.amazon.com/Head-First-Java-Kathy-Sierra/dp/0596009208)
   * Beginner-friendly Java programming guide.
2. [Effective Java by Joshua Bloch](https://www.amazon.com/Effective-Java-3rd-Joshua-Bloch/dp/0134685997)
   * Best practices for writing clean, efficient, and maintainable Java code.
3. [Java: The Complete Reference by Herbert Schildt](https://www.amazon.com/Java-Complete-Reference-10th/dp/1260440230)
   * A comprehensive guide to Java for deep understanding of language and libraries.

### ****Online Resources:****

1. [Oracle Java Tutorials](https://docs.oracle.com/javase/tutorial/)
   * Official guide to learning Java programming and core libraries.
2. **GeeksforGeeks Java Tutorials**
   * Tutorials on Java programming, JDBC, and collections.
3. **W3Schools MySQL Tutorial**
   * Beginner-friendly tutorial on MySQL for designing and managing databases.
4. **Java Code Geeks JDBC Tutorials**
   * Guide to Java Database Connectivity (JDBC) for database operations.