RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BLOOD DONATION MANAGEMENT SYSTEM

MINI PROJECT REPORT

SUBMITTED BY

230701019 AKASH N

230701023 AKSHAYA BALAJI N

In partial fulfilment for the award of the degree

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING
COLLEGE(AUTONOMOUS)

THANDALAM

CHENNAI-602105

BONAFIDE CERTIFICATE

Certified that this project report

"BLOOD DONATION MANAGEMENT SYSTEM"

is the bonafide work of

"AKASH N(230701019),

AKSHAYA BALAJI N(230701023)"

who carried out the project work under my supervision.

Submitted for the Practical Examination held on 23.11.2024_____

Signature of faculty in-charge

ABSTRACT

Blood DONATION Management System Project Overview

The Blood Management System automates the processes involved in blood donation and distribution. This system provides essential information such as donor details, blood availability, and requests for blood, allowing efficient data storage and retrieval related to blood donation and transfusion. The system enables the recording of donor information when a donation is made and facilitates tracking the inventory of blood products. We use a MySQL database for data storage and retrieval, ensuring a scalable and efficient management system.

Key

- **Donors**
- **Blood Banks**
- **Patients/Requestors**

Each component accesses a database schema with corresponding tables.

Language Used

- Java Core

Concept Used

- Swing

IDE Used

- NetBeans

Database Used

- MySQL

CONTENTS

CHAPTERS PAGE NO

CHAPTER 1: INTRODUCTION

The Blood Management System aims to streamline the process of blood donation and transfusion, ensuring that blood banks can efficiently manage their inventory and donors can easily offer their blood. This system not only helps in maintaining accurate records of donors and blood availability but also ensures that patients receive timely blood transfusions. The introduction of this system reflects the commitment to leveraging technology in the healthcare sector to improve efficiency and responsiveness.

1.1 Problem Definition

This project automates the processes involved in blood management. The system can provide crucial information such as donor details, blood type availability, and blood requests. It allows users to add records when a donation occurs and to update or delete records based on the requirements of donors and patients. The use of a MySQL database allows for seamless storage and retrieval of data, ensuring that any changes in the frontend are accurately reflected in the backend.

1.2 Need

The Blood Management System addresses the need for electronic management of blood records to enhance accuracy, reliability, and efficiency while minimizing human error. In a world where blood donation is critical, it is essential to maintain up-to-date records regarding the availability of blood types and donor information. The healthcare sector is under constant pressure to manage resources effectively, and the Blood Management System aims to provide a robust structure capable of handling numerous records.

Key Factors for Development:

- **Faster System**
- **Accuracy**
- **Reliability**
- **Informative**
- **Easy management of donations and requests from anywhere**

CHAPTER 2: REQUIREMENTS

2.1 Software Requirement Specifications

- **Operating System**: ○ Windows 10
- **Frontend Software**:
  - **Programming Language**: Java
  - **Integrated Development Environment (IDE)**: NetBeans 8.2 ○ **Java Development Kit**: JDK 8
- **Backend Software**:
  - **Database Management System**: MySQL
- **Server**:
  - Local or remote server for hosting the MySQL database
- **Documentation**:
  - User manuals for end-users
  - Technical documentation for developers and system administrators

CHAPTER 3: ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) diagram is a specialized graphic representation that illustrates the interrelationships between various entities in a database. ER diagrams typically use specific symbols to represent three key types of information:

- **Boxes**: Represent entities
- **Diamonds**: Represent relationships
- **Ovals**: Represent attributes

In the context of the Blood Management System, the ER model serves to outline the data items relevant to the system and the relationships between them. This approach is effective for modeling critical components of the blood management process, such as donor information, blood inventory, and patient requests.

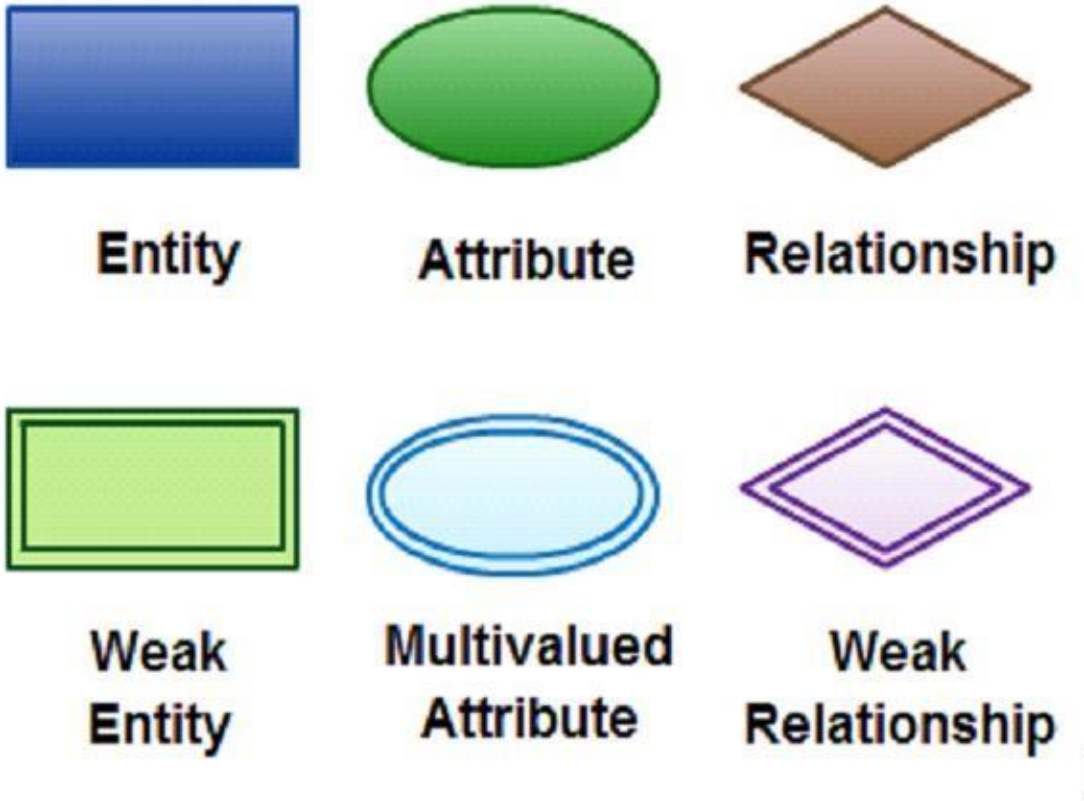
The primary focus of ER modeling is to create a conceptual schema from the user's perspective, ensuring that the model is independent of any specific database implementation. This flexibility allows for a clear understanding of the system's data architecture.

Key Entities and Relationships

1. **Donor**
    a. Attributes: Donor ID, Name, Age, Blood Type, Contact Information, Donation History
2. **Blood Bank**
    a. Attributes: Blood Bank ID, Name, Location, Contact Information, Inventory
3. **Patient**
    a. Attributes: Patient ID, Name, Age, Blood Type, Contact Information, Medical History
4. **Donation**
    a. Attributes: Donation ID, Donor ID, Blood Bank ID, Date of Donation, Amount Collected
5. **Request**
    a. Attributes: Request ID, Patient ID, Blood Type Needed, Quantity, Date of Request, Status

Relationships

- **Donor to Donation**: A donor can make multiple donations (One-to-Many).
- **Blood Bank to Donation**: A blood bank can receive multiple donations (One-toMany).
- **Patient to Request**: A patient can make multiple blood requests (One-to-Many).
- **Blood Bank to Request**: A blood bank can fulfill multiple requests (One-to-Many)



CHAPTER 4: DATABASE SCHEMA

A database schema is the structural framework that represents the logical view of an entire database. It defines the various entities within the system and the relationships among them. In the context of the Blood Management System, the schema outlines how data is organized and the associations between different data elements. This framework formulates all the constraints that are necessary for maintaining data integrity.

Key Components of the Database Schema

1. **Entities**: The main components of the schema, representing real-world objects or concepts.
    a. **Donor**
    b. **Blood Bank**
    c. **Patient**
    d. **Donation**
    e. **Request**

2.Attributes: The data elements associated with each entity.

- **Donor**:
    - Donor ID (Primary Key)
    - Name ○ Age ○ Blood Type ○ Contact Information
    - Donation History

Donation:

- Donation ID (Primary Key)
- Donor ID (Foreign Key)
- Blood Bank ID (Foreign Key)
- Date of Donation

Relationships: Define how entities interact with each other.

- A **Donor** can make multiple **Donations** (One-to-Many).
- A **Blood Bank** can receive multiple **Donations** (One-to-Many).
- A **Patient** can place multiple **Requests** for blood (One-to-Many).
- A **Blood Bank** can fulfill multiple **Requests** (One-to-Many).

CHAPTER 5: IMPLEMENTATION

5.1 Backend Implementation

MySQL

MySQL is an open-source relational database management system (RDBMS) widely used for managing and organizing data in structured formats. In the context of the Blood Management System, MySQL serves as the backbone for data storage, retrieval, and management.

A relational database organizes data into one or more tables, where data types may be related to each other through defined relationships. This structured approach ensures that data is stored efficiently and can be accessed and manipulated with ease.

Key Features of MySQL for the Blood Management System:

1. **Data Organization**: MySQL allows for the creation of multiple tables to store various entities such as Donors, Blood Banks, Patients, Donations, and Requests. Each table can define relationships with others, ensuring data integrity and coherence.

2. **SQL Language**: Structured Query Language (SQL) is the programming language used to interact with the database. Through SQL, developers can:
    a. Create tables and define their schemas.
    b. Insert, update, and delete records as needed.
    c. Query the database to retrieve specific information (e.g., available blood types, donor details).

3. **User Management**: MySQL provides robust user management capabilities, allowing administrators to control access to the database. Different roles can be defined (e.g., admin, staff), each with specific permissions, ensuring that sensitive data is protected.

4. **Network Access**: MySQL supports networked applications, allowing remote access to the database. This is particularly useful for blood banks and hospitals that may need to share data securely across locations.

5. **Data Integrity and Backup**: MySQL includes features to maintain database integrity, such as transactions and constraints. Additionally, it supports regular backups to prevent data loss and ensure continuity in case of system failures.

```
CREATE TABLE cancellation (  cancellation_id VARCHAR(10 ), donor_id VARCHAR(10),
cancellation_date DATE,      reason VARCHAR(100),  FOREIGN KEY (donor_id) REFERENCES
donor(donor_id));

CREATE TABLE blood_bank (  blood_bank_id VARCHAR(10), name VARCHAR(50), location
VARCHAR(100),   contact_info VARCHAR(100));

CREATE TABLE login (  username VARCHAR(20),   password VARCHAR(20));

CREATE TABLE patient (    patient_id VARCHAR(10),   name VARCHAR(50), age INT, blood_type
VARCHAR(5),   contact_info VARCHAR(100), medical_history TEXT) ); CREATE TABLE donor (
donor_id VARCHAR(10),    name VARCHAR(50),  age INT,  blood_type VARCHAR(5),
contact_info VARCHAR(100),    address VARCHAR(100) );
```

5.2 Frontend Implementation

Java Core

Core Java is the foundation of the Java programming language, used for creating generalpurpose applications. In the context of the Blood Management System, Core Java enables the development of a standalone application that provides a user-friendly interface for interacting with the system. Core Java encompasses various concepts such as Swing, AWT, threading, and collections, which are essential for building robust applications.

Swing

Swing is a powerful GUI widget toolkit for Java, part of Oracle's Java Foundation Classes (JFC). It provides an API for developing graphical user interfaces (GUIs) in Java applications. Swing is particularly well-suited for the Blood Management System due to its flexibility and rich set of components, which enhance the user experience.

Key Features of Swing for the Blood Management System:

1. **Cross-Platform Look and Feel**: Swing supports a pluggable look and feel, allowing the application to maintain a consistent interface across different operating systems. This ensures that users have a familiar experience, regardless of the platform they are using.

2. **Event Handling**: Swing provides a robust event-handling model that allows developers to define actions in response to user interactions. This capability is essential for ensuring that the Blood Management System responds efficiently to user input, such as clicking buttons or selecting items from lists.
3. **Customization and Flexibility**: Swing components can be easily customized to meet the specific needs of the Blood Management System, allowing for a tailored user experience

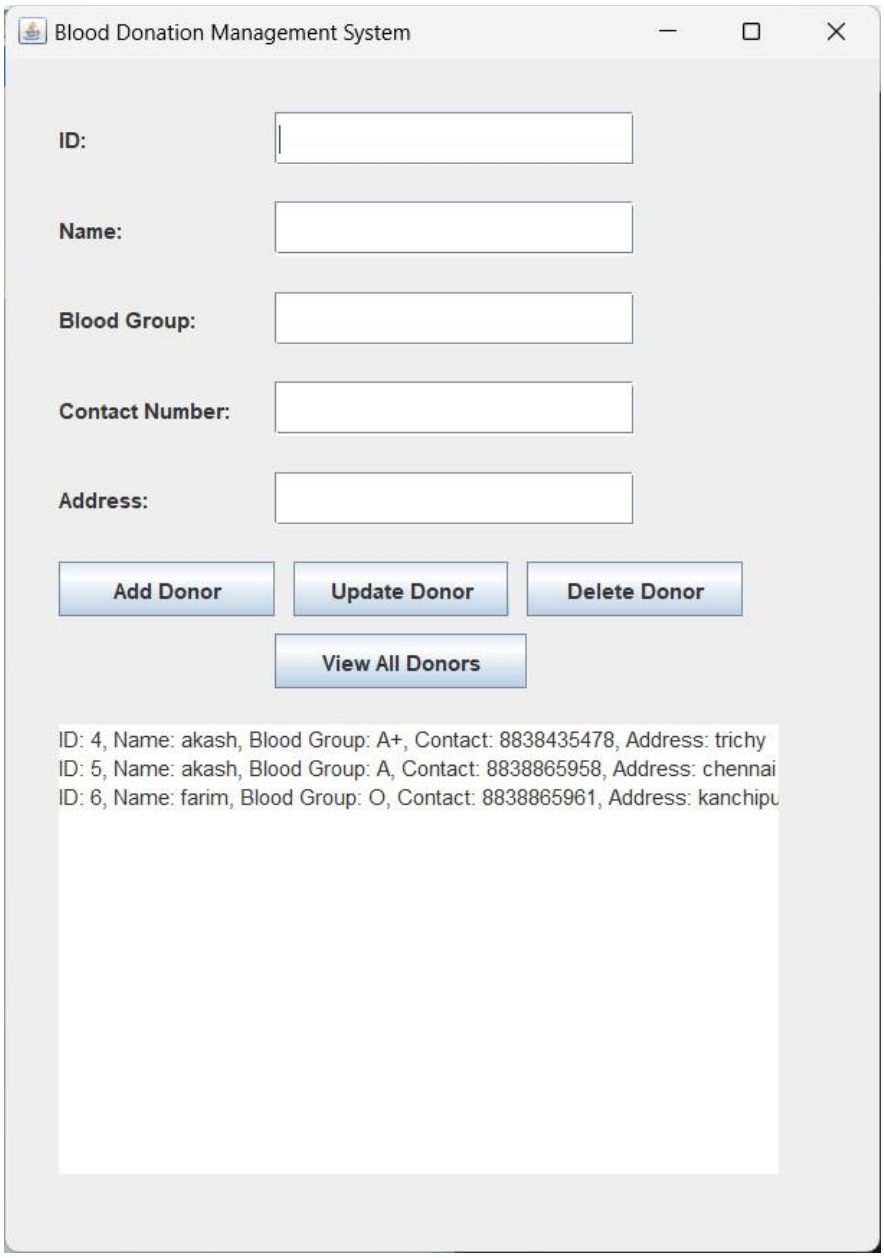
```
package blood.management.system;
```

```
import net.proteanit.sql.DbUtils;
import java.awt.*; import
javax.swing.*;
import java.awt.event.*;

public class Mainframe extends JFrame {     public
static void main(String[] args) {
      new Mainframe().setVisible(true);
   }

   public Mainframe() {
      super("BLOOD MANAGEMENT SYSTEM");
initialize();
   }

   private void initialize() {
setForeground(Color.CYAN);
      setLayout(null);

      JLabel backgroundLabel = new JLabel("");
      backgroundLabel.setIcon(new
ImageIcon(ClassLoader.getSystemResource("blood/management/system/icon/ba
ckground.jpg")));
      backgroundLabel.setBounds(0, 0, 1920, 990);        add(backgroundLabel);

     JLabel welcomeLabel = new JLabel("WELCOME TO BLOOD MANAGEMENT SYSTEM");
      welcomeLabel.setForeground(Color.BLUE);
      welcomeLabel.setFont(new Font("Tahoma", Font.PLAIN, 36));
welcomeLabel.setBounds(500, 60, 1000, 55);
      backgroundLabel.add(welcomeLabel);

      JMenuBar menuBar = new JMenuBar();
setJMenuBar(menuBar);

      JMenu donorMenu = new JMenu("DONOR MANAGEMENT");
      donorMenu.setForeground(Color.BLUE);
menuBar.add(donorMenu);
```

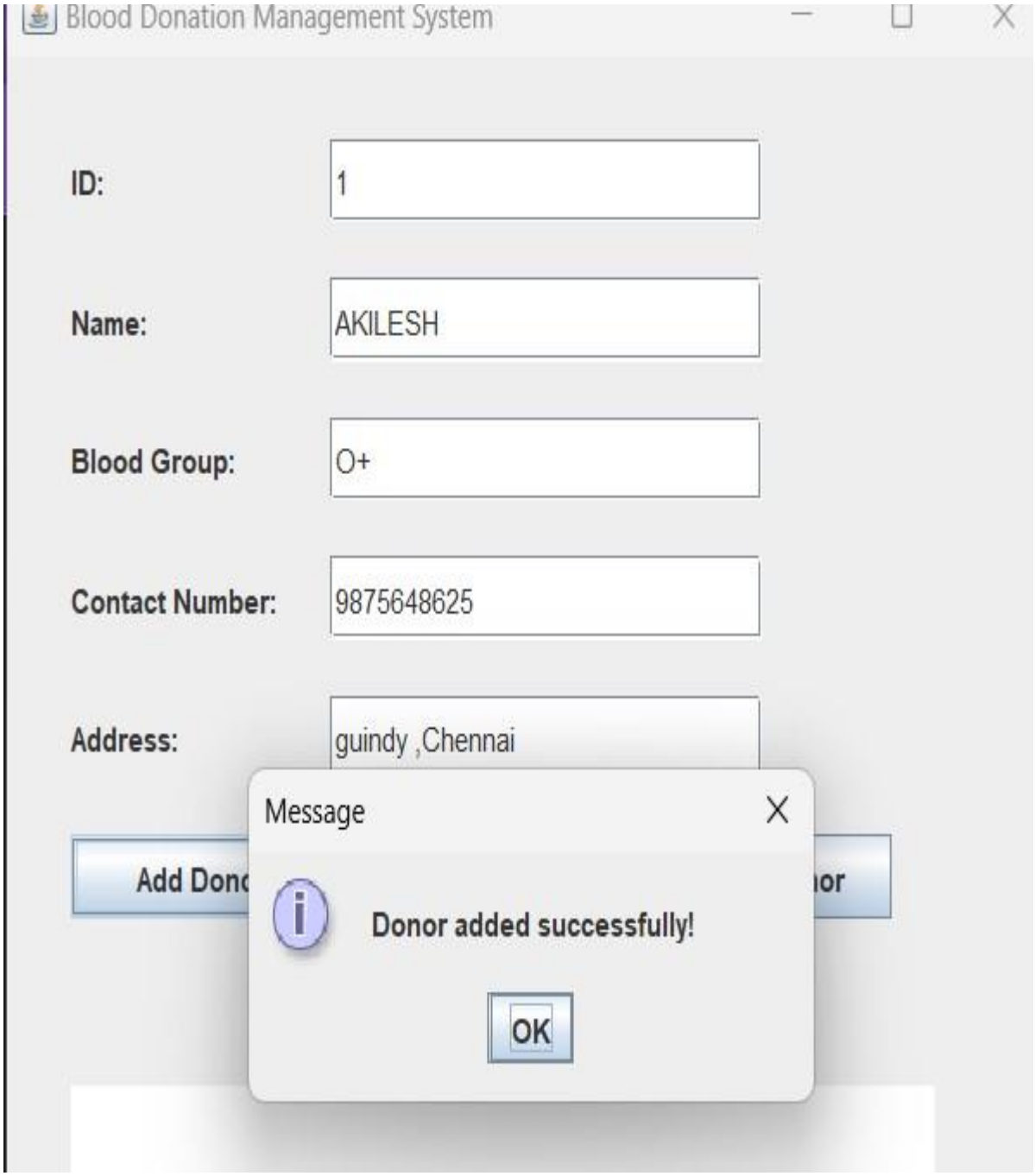
```
        JMenuItem addDonor = new JMenuItem("ADD DONOR");
donorMenu.add(addDonor);
        JMenuItem viewDonor = new JMenuItem("VIEW DONORS");
        donorMenu.add(viewDonor);

        JMenu patientMenu = new JMenu("PATIENT MANAGEMENT");
        patientMenu.setForeground(Color.RED);
menuBar.add(patientMenu);

        JMenuItem addPatient = new JMenuItem("ADD PATIENT");
patientMenu.add(addPatient);
        JMenuItem viewPatients = new JMenuItem("VIEW PATIENTS");
patientMenu.add(viewPatients);

        JMenu donationMenu = new JMenu("DONATION MANAGEMENT");
        donationMenu.setForeground(Color.GREEN);        menuBar.add(donationMenu);

        JMenuItem recordDonation = new JMenuItem("RECORD DONATION");
donationMenu.add(recordDonation);
        JMenuItem viewDonations = new JMenuItem("VIEW DONATIONS");
donationMenu.add(viewDonations);

        JMenu requestMenu = new JMenu("REQUEST MANAGEMENT");
requestMenu.setForeground(Color.MAGENTA);
        menuBar.add(requestMenu);

        JMenuItem addRequest = new JMenuItem("ADD REQUEST");
requestMenu.add(addRequest);
        JMenuItem viewRequests = new JMenuItem("VIEW REQUESTS");
requestMenu.add(viewRequests);

        // Action Listeners
        addDonor.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ae) {
            new Add_Donor(); // Assuming you have a class for adding donors
          }
        });
```

```
        viewDonor.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new View_Donors(); // Assuming you have a class for viewing donors
            }
        });

        addPatient.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new Add_Patient(); // Assuming you have a class for adding patients
            }
        });

        viewPatients.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new View_Patients(); // Assuming you have a class for viewing patients
            }
        });

        recordDonation.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new Record_Donation(); // Assuming you have a class for recording donations
            }
        });

        viewDonations.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new View_Donations(); // Assuming you have a class for viewing donations
            }
        });

        addRequest.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new Add_Request(); // Assuming you have a class for adding requests
            }
        });
```

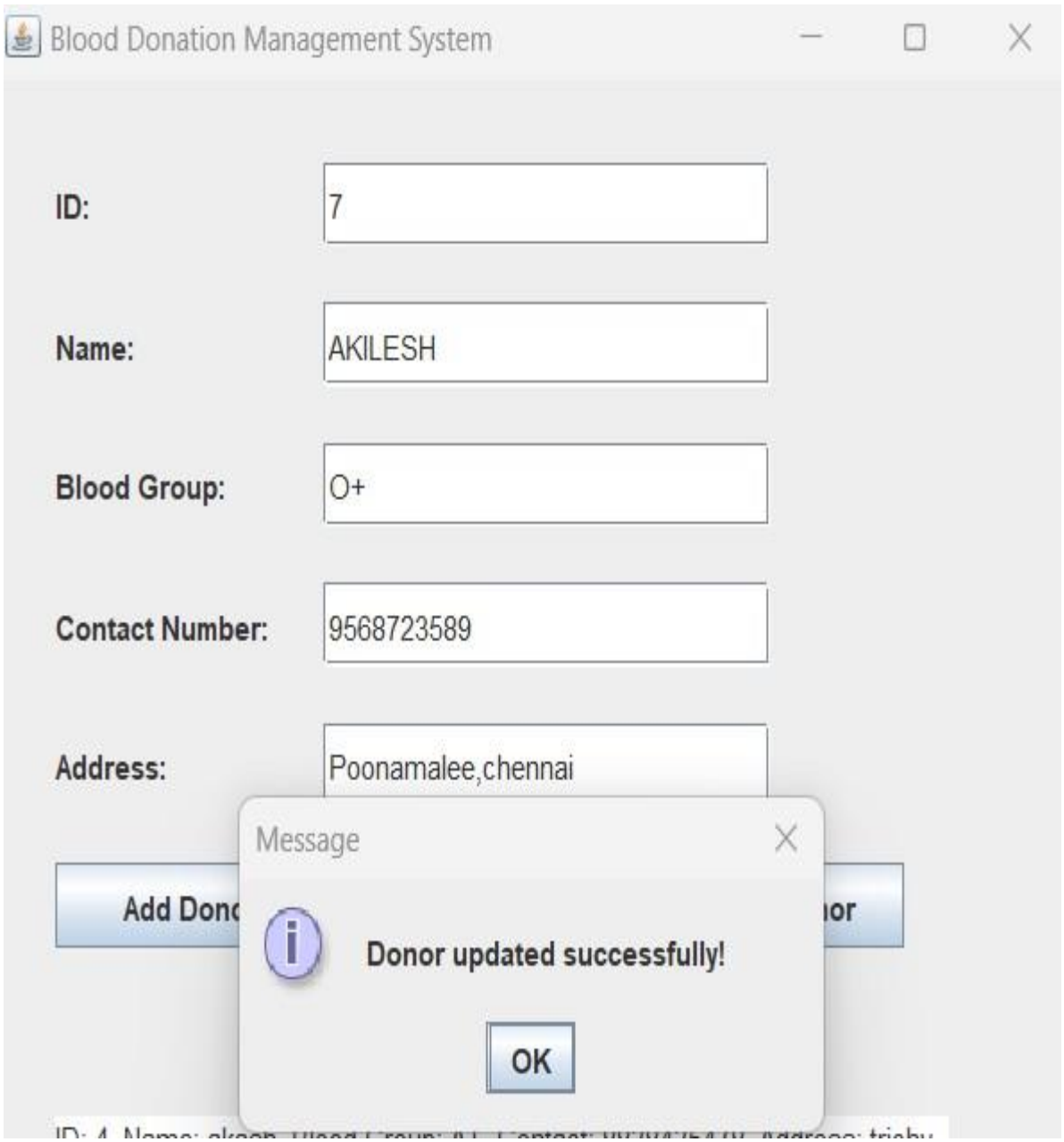
```
        viewRequests.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
                new View_Requests(); // Assuming you have a class for viewing requests
            }
        });

        setSize(1950, 1090);
    setVisible(true);
      }
    }
```


Blood Donation Management System
ID:
Name:
Blood Group:
Contact Number:
Address:
Add Donor
Update Donor
Delete Donor
View All Donors
ID: 4, Name: akash, Blood Group: A+, Contact: 8838435478, Address: trichy
ID: 5, Name: akash, Blood Group: A, Contact: 8838865958, Address: chennai
ID: 6, Name: farim, Blood Group: O, Contact: 8838865961, Address: kanchipu

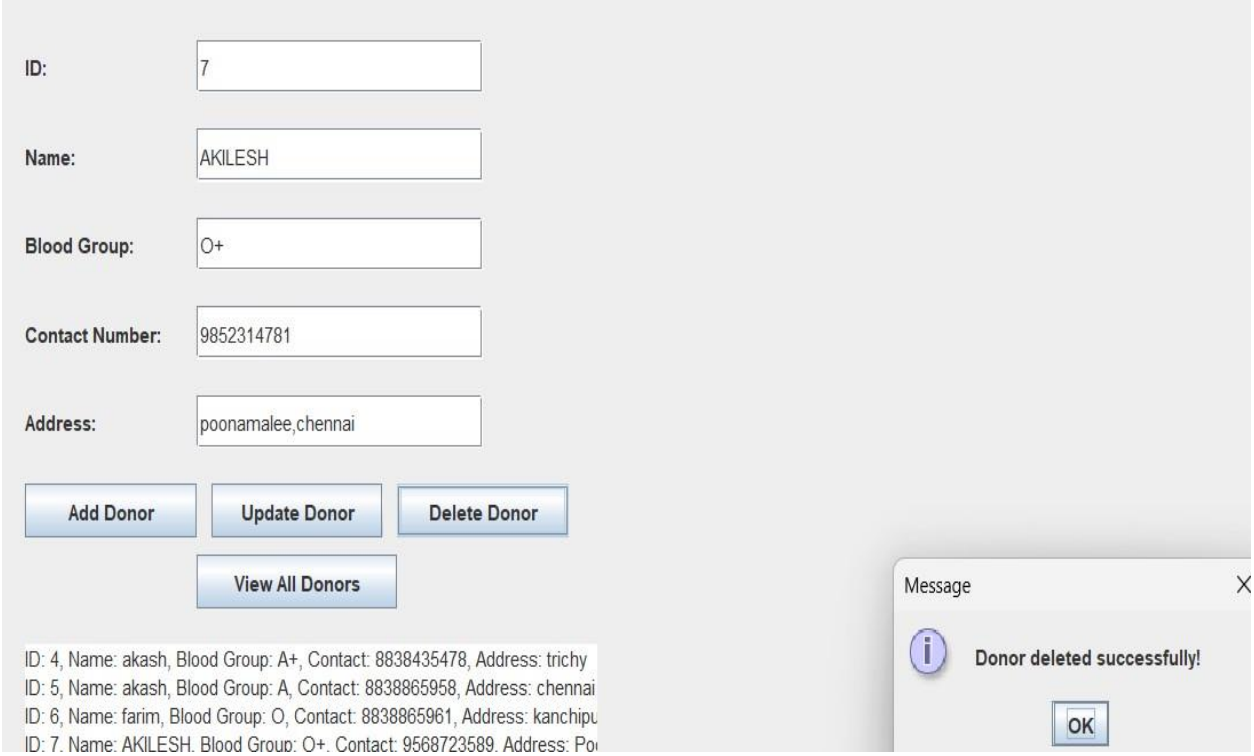
ADD DONOR;



ADDRESS IS WRONG ,SO UPDATE DONOR ADDRESS;



DELETE DONOR



CONCLUSION

In conclusion, the Blood Management System is an invaluable tool for efficiently managing blood inventory, availability, and donor records, thus significantly streamlining the blood management process. This system not only enhances the organization of donor data and blood stock but also improves the overall operational effectiveness of blood banks and healthcare facilities. By providing real-time tracking of blood inventory, facilitating seamless blood donations, and ensuring proper compatibility checks during transfusions, the system meets the essential needs of both healthcare providers and donors.

Moreover, with automated record-keeping, appointment scheduling, and easy access to detailed reports, the Blood Management System reduces the administrative burden on staff, allowing them to focus on providing quality care and maintaining a safe and reliable blood supply. This system plays a crucial role in creating a well-organized, transparent, and efficient blood management process that meets the demands of modern healthcare systems while ensuring the availability of lifesaving resources when needed.

REFERENCE: https://developers.openshift.com/database/mysql.html

Web References- https://youtu.be/UbIIFLsEeiM