BLOOD DONATION MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

KEERTHANA M G 220701123

HAREESH S S 220701079

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institute

THANDALAM

CHENNAI-602105

2023-2024

BONAFIDE CERTIFICATE

Certified that this Project report "**BLOOD DONATION MANAGEMENT SYSTEM**" is the bonafide work of "**KEERTHANA MG (220701123)**" and "**HAREESH S S (220701079)**" who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Dr.R.SABITHA
Professor and Academic Head,
Computer Science and Engineering,
Rajalakshmi Engineering College (Autonomous),
Thandalam, Chennai - 602 105

SIGNATURE

Dr. G. DHARANI DEVI
Associate Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College, (Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Blood Donation Management System (BDMS) is a Java Based application with SQL database that serves as a vital bridge between blood donors, healthcare institutions and those in need of life-saving blood transfusions. With Java as the frontend and SQL as the backend, this system provides a robust and user-friendly platform for efficient blood donation operations. Leveraging the power of object-oriented programming in Java and the data storage capabilities of SQL, the DBMS project exemplifies a holistic approach to managing the entire blood donation process.

This comprehensive system empowers donors to register, schedule donations, and track their contributions, while healthcare facilities can seamlessly manage blood inventory, donor records, and distribution. By combining the flexibility of Java for the frontend and the reliability of SQL for data management, the DBMS ensures a scalable and maintainable solution for the blood donation community. Through this project, we aim to facilitate the altruistic act of blood donation and enhance the overall management of this critical resource, contributing to the well-being of our communities.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 BLOOD DONATION MANAGEMENT SYSTEM

The project blood bank management system is known to be a pilot project that is designed for the blood bank to gather blood from various sources and distribute it to the needy people who have high Requirements for it. The Software is designed to handle the daily transaction of the blood bank and search the details when required. It also helps to register the details of donors, blood collection details as well as blood issued reports. At any point of time the people who are in need can reach the donors through our search facility. By mobilizing people and organization who desire to make a difference in the lives of people in need. On the basis of humanity, everyone is welcome to register as a blood donor.

1.2 IMPLEMENTATION:

The **BLOOD DONATION MANAGEMENT SYSTEM** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL.**

1.3 PRODUCT SCOPE:

This application is built such a way that it suits for all type of blood bank in future.so every effort is taken to implement this project in this blood bank, on successful implementation in this blood bank, we can target other blood banks in the city. Main modules of the project: This project have the following modules, to manage all the requirements of the blood bank.

1. Blood donor details
2. Donor details
3. Recipient details
4. Blood collection details
5. Blood issued details
6. Stock details
7. Camp details
8. Reports

1.4 SYSTEM FEATURES:

❖ Donor database-blood group wise and area wise.

❖ Maintain and update unique donor identifications.

❖ Track and maintain all the donor types- voluntary, exchange and directed.

❖ Improved the Effectiveness and Efficiency of blood bank- faster response time and better control.

❖ Accurate database/Record management.

❖ Blood cross match and result storage facility.

❖ Rejected donor database for donor control and identifications.

❖ Blood transfusion related diseases control and prevention

1.5 OBJECTIVES

Blood Donation Management System is designed and suitable for several Blood Bank either operating as individual's organizations or part of organizations covers all blood banking process from donors recruitment, donor management, mobile session component preparation, screening covering all test, blood stock inventory maintenance, patient registration, cross matching, patient issues etc.

1.6 PRODUCT FUNCTION

CLASS OF USE CASES	USE CASES	DESCRIPTION
Use cases related to system authorization of system administrator	1.Login of admin. 2. Change password of admin.	1.Log admin into the system. 2.Change login password of the admin of the system
Use cases related to registration of donor.	1.Register the donor by himself. 2.Register the donor by system admin.	1.store personal, contact, medical details of donors. 2. store personal, contact, medical details of donors
Use cases related to system authorization of the donor.	1.Login of donor. 2.Change password of the donor.	1.Log donor into the system. 2.Change login password of the donors of the system.
Use cases related to change the registration details of donor.	1.Change personal, contact details by the donor himself. 2.Change personal, contact details by system admin.	1. Change personal and contact details of donors. 2. Change personal and contact details of donors.
Use cases related to withdraw names from the donor list.	1.withdraw reg. details by the donor. 2.withdraw reg. details by the admin.	1.Delete all details of an exact donors by themselves. 2. Delete all details of an exact donors by the system admin.

Use cases related to inform blood donation details	Send blood donation details to the relevant donors.	Inform the requirement of the blood group to donors who has same blood group.
Use cases related to replace the older HC Certificates.	Replace donors' HC Certificates	Override the help condition report details.

2. SURVEY OF TECHNOLOGIES

2.1 LANGUAGES

2.1.1 SQL:

The SQL Command Prompt is essential for executing SQL statements to manage the database, such as creating the necessary tables (**donors** and **donor_status**), inserting initial data, and performing database maintenance tasks. It allows direct interaction with the MySQL database to ensure the schema is correctly set up and facilitates debugging SQL-related issues during the project development.

2.1.2 JAVA:

Java files are crucial for the implementation of the Blood Donation Management System. They contain the main logic for the application, including the user interface created using Swing, and the interaction with the database through JDBC. These files manage the core functionalities such as donor registration, data retrieval, updates, and deletions, ensuring a seamless user experience in managing blood donation records.

3. REQUIREMENT AND ANALYSIS

3.1. REQUIREMENT SPECIFICATION

Java Development Kit, MySQL Server, MySQL Connector, JDBC Connectivity

3.2. SOFTWARE AND HARDWARE REQUIREMENT

3.2.1. HARDWARE REQUIREMENT:

PROCESSOR - Intel® core™ i5-6006U @ 2.00 GHz

RAM - 4GB

OS -Windows, Mac or Linux

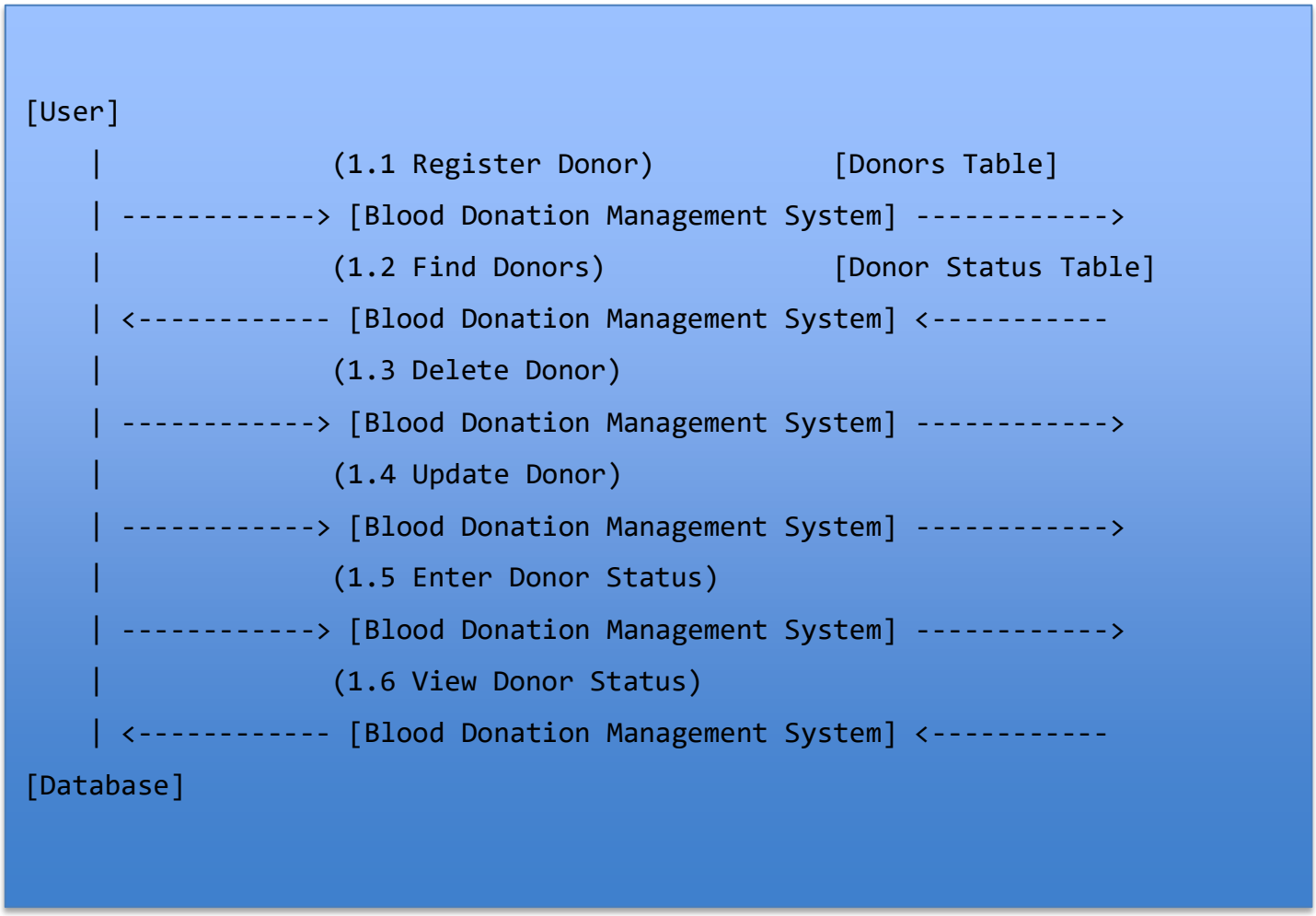
HARD DISK - 500 GB of free space

3.2.2. SOFTWARE REQUIREMENT:

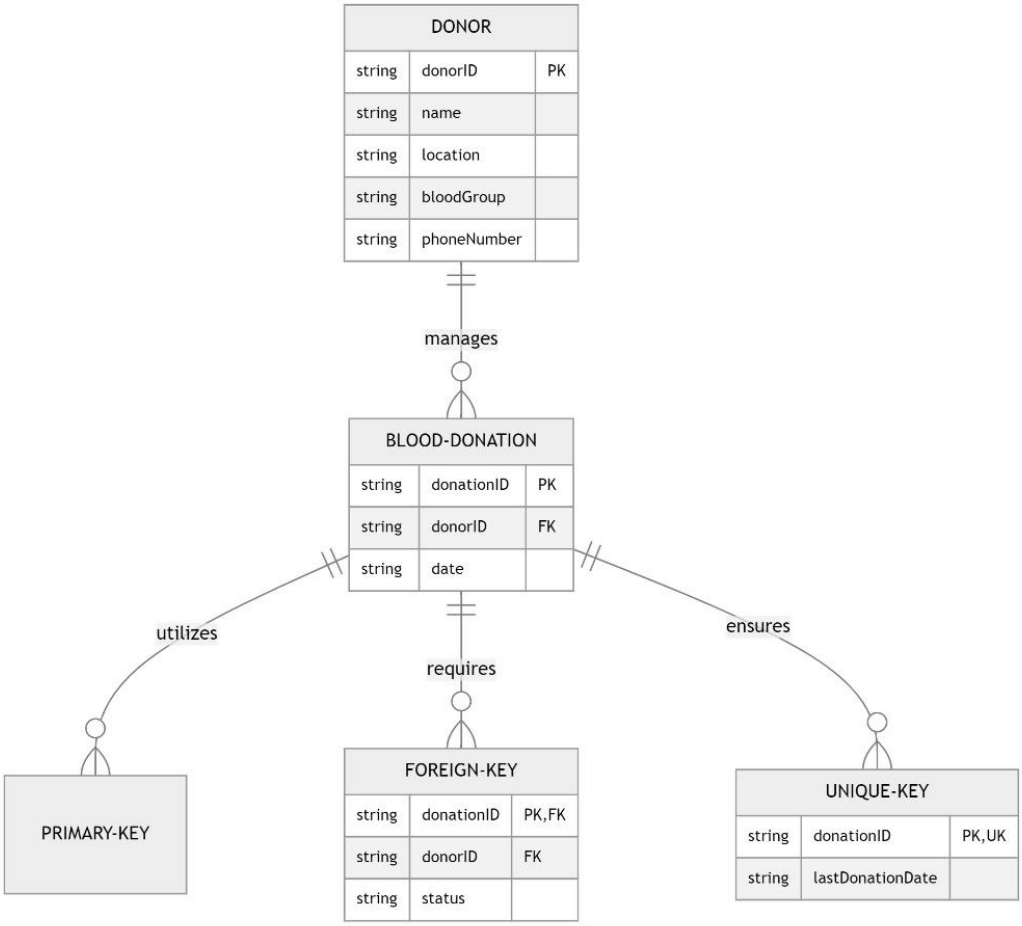
PROGRAMMING LANGUAGE: Java, MySQL

OPERATING SYSTEM : Windows, Mac or Linux

3.3. DATA FLOW DIAGRAM



3.4. ER DIAGRAM



3.5. NORMALISATION

Normalization is a process of organizing data in a database to reduce redundancy and improve data integrity. Here's how you can normalize the data for the Blood Donation Management System:

1st Normal Form (1NF):

In the first normal form, we ensure that each table has a primary key and that each column contains atomic values, i.e., each column contains only one value per row.

Donors Table

Primary Key: id

Columns:

id: INT, PRIMARY KEY, AUTO_INCREMENT

name: VARCHAR(255), NOT NULL

blood_group: VARCHAR(5), NOT NULL

location: VARCHAR(255), NOT NULL

mobile_number: VARCHAR(15), NOT NULL

Donor Status Table

Primary Key: id

Foreign Key: donor_id references donors(id)

Columns:

id: INT, PRIMARY KEY, AUTO_INCREMENT

donor_id: INT, NOT NULL

last_donation_month: VARCHAR(20), NOT NULL

eligibility: VARCHAR(10), NOT NULL

2nd Normal Form (2NF):

In the second normal form, we ensure that all non-key attributes are fully functional dependent on the primary key.

The Donors table is already in 2NF as all non-key attributes (name, blood_group, location, mobile_number) are fully dependent on the primary key (id).

The Donor Status table is also in 2NF as all non-key attributes (last_donation_month, eligibility) are fully dependent on the primary key (id).

3rd Normal Form (3NF):

In the third normal form, we ensure that there are no transitive dependencies, i.e., non-key attributes should not depend on other non-key attributes.

The Donors table is in 3NF as there are no transitive dependencies.

The Donor Status table is in 3NF as there are no transitive dependencies.

4. SOURCE CODE:

SQL CODE:

```
-- Create Donors Table
CREATE TABLE IF NOT EXISTS donors (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    blood_group VARCHAR(5) NOT NULL,
    location VARCHAR(255) NOT NULL,
    mobile_number VARCHAR(15) NOT NULL
);

-- Create Donor Status Table
CREATE TABLE IF NOT EXISTS donor_status (
    id INT AUTO_INCREMENT PRIMARY KEY,
    donor_id INT,
    last_donation_month VARCHAR(20),
    eligibility VARCHAR(20),
    FOREIGN KEY (donor_id) REFERENCES donors(id) ON DELETE CASCADE
);
```

JAVA CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;

public class blood_donate {

    private static Connection connection;

    public static void main(String[] args) {
        try {
            String jdbcURL = "jdbc:mysql://localhost:3306/blood_donation";
            String dbUsername = "root";
            String dbPassword = "0709";
            connection = DriverManager.getConnection(jdbcURL, dbUsername, dbPassword);

            createTableIfNotExists(connection);

            SwingUtilities.invokeLater(() -> createAndShowMainGUI());
```

```
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void createTableIfNotExists(Connection connection) throws SQLException {
    String createDonorsTableSQL = "CREATE TABLE IF NOT EXISTS donors ("
            + "id INT AUTO_INCREMENT PRIMARY KEY,"
            + "name VARCHAR(255) NOT NULL,"
            + "blood_group VARCHAR(5) NOT NULL,"
            + "location VARCHAR(255) NOT NULL,"
            + "mobile_number VARCHAR(15) NOT NULL"
            + ")";

    String createDonorStatusTableSQL = "CREATE TABLE IF NOT EXISTS donor_status ("
            + "id INT AUTO_INCREMENT PRIMARY KEY,"
            + "donor_id INT,"
            + "last_donation_month VARCHAR(20),"
            + "eligibility VARCHAR(20),"
            + "FOREIGN KEY (donor_id) REFERENCES donors(id) ON DELETE CASCADE"
            + ")";

    try (Statement statement = connection.createStatement()) {
        statement.executeUpdate(createDonorsTableSQL);
        statement.executeUpdate(createDonorStatusTableSQL);
    }
}

private static void createAndShowMainGUI() {
    JFrame frame = new JFrame("Blood Donation Management System");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(7, 1));

    JButton registerButton = new JButton("Register Donor");
    JButton findButton = new JButton("Find Donors by Blood Group");
    JButton deleteButton = new JButton("Delete Donor by Name");
    JButton updateButton = new JButton("Update Donor Information");
    JButton enterStatusButton = new JButton("Enter Donor Status");
    JButton viewStatusButton = new JButton("View Donor Status");
    JButton exitButton = new JButton("Exit");

    panel.add(registerButton);
    panel.add(findButton);
    panel.add(deleteButton);
    panel.add(updateButton);
    panel.add(enterStatusButton);
    panel.add(viewStatusButton);
    panel.add(exitButton);
```

```
        registerButton.addActionListener(e -> showRegisterDonorWindow());
        findButton.addActionListener(e -> showFindDonorsWindow());
        deleteButton.addActionListener(e -> showDeleteDonorWindow());
        updateButton.addActionListener(e -> showUpdateDonorWindow());
        enterStatusButton.addActionListener(e -> showEnterDonorStatusWindow());
        viewStatusButton.addActionListener(e -> showViewDonorStatusWindow());
        exitButton.addActionListener(e -> System.exit(0));

        frame.getContentPane().add(BorderLayout.CENTER, panel);
        frame.setVisible(true);
    }

    private static void showRegisterDonorWindow() {
        JFrame frame = new JFrame("Register Donor");
        frame.setSize(400, 300);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(5, 2));

        JTextField nameField = new JTextField();
        JTextField bloodGroupField = new JTextField();
        JTextField locationField = new JTextField();
        JTextField mobileNumberField = new JTextField();

        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Blood Group:"));
        panel.add(bloodGroupField);
        panel.add(new JLabel("Location:"));
        panel.add(locationField);
        panel.add(new JLabel("Mobile Number:"));
        panel.add(mobileNumberField);

        JButton registerButton = new JButton("Register");
        registerButton.addActionListener(e -> {
            registerDonor(nameField.getText(), bloodGroupField.getText(), locationField.getText(),
mobileNumberField.getText());
            frame.dispose();
        });

        panel.add(registerButton);

        frame.getContentPane().add(BorderLayout.CENTER, panel);
        frame.setVisible(true);
    }

    private static void showFindDonorsWindow() {
        JFrame frame = new JFrame("Find Donors by Blood Group");
        frame.setSize(300, 150);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(2, 2));
```

```
    JTextField bloodGroupField = new JTextField();

    panel.add(new JLabel("Blood Group:"));
    panel.add(bloodGroupField);

    JButton findButton = new JButton("Find");
    findButton.addActionListener(e -> {
        findDonorsByBloodGroup(bloodGroupField.getText());
        frame.dispose();
    });

    panel.add(findButton);

    frame.getContentPane().add(BorderLayout.CENTER, panel);
    frame.setVisible(true);
}

private static void showDeleteDonorWindow() {
    JFrame frame = new JFrame("Delete Donor by Name");
    frame.setSize(300, 150);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(2, 2));

    JTextField nameField = new JTextField();

    panel.add(new JLabel("Name:"));
    panel.add(nameField);

    JButton deleteButton = new JButton("Delete");
    deleteButton.addActionListener(e -> {
        deleteDonorByName(nameField.getText());
        frame.dispose();
    });

    panel.add(deleteButton);

    frame.getContentPane().add(BorderLayout.CENTER, panel);
    frame.setVisible(true);
}

private static void showUpdateDonorWindow() {
    JFrame frame = new JFrame("Update Donor Information");
    frame.setSize(400, 300);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(6, 2));

    JTextField donorNameField = new JTextField();
    JTextField newNameField = new JTextField();
    JTextField newBloodGroupField = new JTextField();
```

```
    JTextField newLocationField = new JTextField();
    JTextField newMobileNumberField = new JTextField();

    panel.add(new JLabel("Donor Name:"));
    panel.add(donorNameField);
    panel.add(new JLabel("New Name:"));
    panel.add(newNameField);
    panel.add(new JLabel("New Blood Group:"));
    panel.add(newBloodGroupField);
    panel.add(new JLabel("New Location:"));
    panel.add(newLocationField);
    panel.add(new JLabel("New Mobile Number:"));
    panel.add(newMobileNumberField);

    JButton updateButton = new JButton("Update");
    updateButton.addActionListener(e -> {
        updateDonorInformation(
                donorNameField.getText(),
                newNameField.getText(),
                newBloodGroupField.getText(),
                newLocationField.getText(),
                newMobileNumberField.getText()
        );
        frame.dispose();
    });

    panel.add(updateButton);

    frame.getContentPane().add(BorderLayout.CENTER, panel);
    frame.setVisible(true);
}

private static void showEnterDonorStatusWindow() {
    JFrame frame = new JFrame("Enter Donor Status");
    frame.setSize(400, 300);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(5, 2));

    JTextField nameField = new JTextField();
    JTextField bloodGroupField = new JTextField();
    JTextField lastDonationMonthField = new JTextField();
    JTextField eligibilityField = new JTextField();

    panel.add(new JLabel("Name:"));
    panel.add(nameField);
    panel.add(new JLabel("Blood Group:"));
    panel.add(bloodGroupField);
    panel.add(new JLabel("Last Donation Month:"));
    panel.add(lastDonationMonthField);
    panel.add(new JLabel("Eligibility:"));
    panel.add(eligibilityField);
```

```
        JButton saveButton = new JButton("Save");
        saveButton.addActionListener(e -> {
            enterDonorStatus(nameField.getText(), bloodGroupField.getText(),
lastDonationMonthField.getText(), eligibilityField.getText());
            frame.dispose();
        });

        panel.add(saveButton);

        frame.getContentPane().add(BorderLayout.CENTER, panel);
        frame.setVisible(true);
    }

    private static void showViewDonorStatusWindow() {
        JFrame frame = new JFrame("View Donor Status");
        frame.setSize(300, 150);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 2));

        JTextField nameField = new JTextField();
        JTextField bloodGroupField = new JTextField();

        panel.add(new JLabel("Name:"));
        panel.add(nameField);
        panel.add(new JLabel("Blood Group:"));
        panel.add(bloodGroupField);

        JButton viewButton = new JButton("View");
        viewButton.addActionListener(e -> {
            viewDonorStatus(nameField.getText(), bloodGroupField.getText());
            frame.dispose();
        });

        panel.add(viewButton);

        frame.getContentPane().add(BorderLayout.CENTER, panel);
        frame.setVisible(true);
    }

    private static void registerDonor(String name, String bloodGroup, String location, String
mobileNumber) {
        try {
            String insertSQL = "INSERT INTO donors (name, blood_group, location,
mobile_number) VALUES (?, ?, ?, ?)";
            try (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
                preparedStatement.setString(1, name);
                preparedStatement.setString(2, bloodGroup);
                preparedStatement.setString(3, location);
                preparedStatement.setString(4, mobileNumber);
```

```
            preparedStatement.executeUpdate();
            JOptionPane.showMessageDialog(null, "Donor registered successfully!");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private static void findDonorsByBloodGroup(String bloodGroup) {
    try {
        String selectSQL = "SELECT * FROM donors WHERE blood_group = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {
            preparedStatement.setString(1, bloodGroup);
            ResultSet resultSet = preparedStatement.executeQuery();

            StringBuilder result = new StringBuilder();
            if (resultSet.next()) {
                result.append("\nDonors with blood group ").append(bloodGroup).append(":\n");
                result.append(String.format("%-20s %-15s %-15s %-15s\n", "Name", "Blood
Group", "Location", "Mobile Number"));
                do {
                    result.append(String.format("%-20s %-15s %-15s %-15s\n",
                            resultSet.getString("name"),
                            resultSet.getString("blood_group"),
                            resultSet.getString("location"),
                            resultSet.getString("mobile_number")));
                } while (resultSet.next());
            } else {
                result.append("No donors found with blood group
").append(bloodGroup).append("\n");
            }
            JOptionPane.showMessageDialog(null, result.toString());
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private static void deleteDonorByName(String donorName) {
    try {
        String deleteSQL = "DELETE FROM donors WHERE name = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(deleteSQL)) {
            preparedStatement.setString(1, donorName);

            int affectedRows = preparedStatement.executeUpdate();

            if (affectedRows > 0) {
                JOptionPane.showMessageDialog(null, "Donor with name " + donorName + "
deleted successfully!");
            } else {
                JOptionPane.showMessageDialog(null, "No donor found with name " + donorName);
            }
```

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

    private static void updateDonorInformation(String donorName, String newName, String
newBloodGroup, String newLocation, String newMobileNumber) {
        try {
            String updateSQL = "UPDATE donors SET name=?, blood_group=?, location=?,
mobile_number=? WHERE name=?";
            try (PreparedStatement preparedStatement = connection.prepareStatement(updateSQL)) {
                preparedStatement.setString(1, newName);
                preparedStatement.setString(2, newBloodGroup);
                preparedStatement.setString(3, newLocation);
                preparedStatement.setString(4, newMobileNumber);
                preparedStatement.setString(5, donorName);

                int affectedRows = preparedStatement.executeUpdate();

                if (affectedRows > 0) {
                    JOptionPane.showMessageDialog(null, "Donor with name " + donorName + "
updated successfully!");
                } else {
                    JOptionPane.showMessageDialog(null, "No donor found with name " + donorName);
                }
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    private static void enterDonorStatus(String name, String bloodGroup, String
lastDonationMonth, String eligibility) {
        try {
            String selectSQL = "SELECT id FROM donors WHERE name = ? AND blood_group =
?";
            int donorId = -1;
            try (PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {
                preparedStatement.setString(1, name);
                preparedStatement.setString(2, bloodGroup);
                ResultSet resultSet = preparedStatement.executeQuery();
                if (resultSet.next()) {
                    donorId = resultSet.getInt("id");
                } else {
                    JOptionPane.showMessageDialog(null, "No donor found with the provided name and
blood group.");
                    return;
                }
            }

            String insertSQL = "INSERT INTO donor_status (donor_id, last_donation_month,
```

```
eligibility) VALUES (?, ?, ?)";
        try (PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
            preparedStatement.setInt(1, donorId);
            preparedStatement.setString(2, lastDonationMonth);
            preparedStatement.setString(3, eligibility);

            preparedStatement.executeUpdate();
            JOptionPane.showMessageDialog(null, "Donor status entered successfully!");
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
  }

  private static void viewDonorStatus(String name, String bloodGroup) {
    try {
        String selectSQL = "SELECT donors.name, donors.blood_group,
donor_status.last_donation_month, donor_status.eligibility "
                + "FROM donors JOIN donor_status ON donors.id = donor_status.donor_id "
                + "WHERE donors.name = ? AND donors.blood_group = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(selectSQL)) {
            preparedStatement.setString(1, name);
            preparedStatement.setString(2, bloodGroup);
            ResultSet resultSet = preparedStatement.executeQuery();

            StringBuilder result = new StringBuilder();
            if (resultSet.next()) {
                result.append("Donor Status:\n");
                result.append("Name: ").append(resultSet.getString("name")).append("\n");
                result.append("Blood Group:
").append(resultSet.getString("blood_group")).append("\n");
                result.append("Last Donation Month:
").append(resultSet.getString("last_donation_month")).append("\n");
                result.append("Eligibility: ").append(resultSet.getString("eligibility")).append("\n");
            } else {
                result.append("No status found for donor with the provided name and blood group.");
            }
            JOptionPane.showMessageDialog(null, result.toString());
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
  }
}
```

5. RESULTS AND DISCUSSION

The Blood Donation Management System delivers efficient donor registration and status tracking, ensuring a comprehensive database. Its user-friendly Java Swing interface facilitates easy navigation for users, offering functionalities like donor registration, status entry, and information retrieval. Adhering to normalization principles, the system maintains data integrity and minimizes redundancy, optimizing database performance. By leveraging JDBC interactions, it ensures seamless data storage, retrieval, and manipulation. Overall, the system promotes efficiency, accuracy, and reliability in blood donation operations, enhancing donor management and communication for better healthcare outcomes.

6. CONCLUSION

This program has been created successfully to create a BLOOD DONATION MANAGEMENT SYSTEM

7. REFERENCES

The below websites helped us in gaining more knowledge on the subject and in completing the project

https://www.scribd.com/document/408508426/Blood-Bank-Management-System-BBMS

https://www.academia.edu/37555053/SRS_on_Blood_Bank_Management_System