BLOOD DONATION MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

ENIYAA A(2303811710422040)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

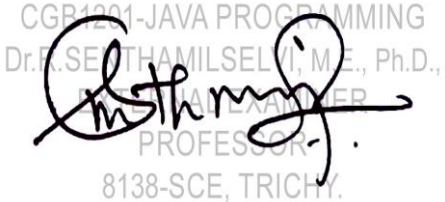
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"BLOOD DONATION MANAGEMENT SYSTEM"** is bonafide work of ENI**YAA A (2303811710422040)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. M. Saravanan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024



INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"BLOOD DONATION MANAGEMENT SYSTEM"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

.

Signature

ENIYAA A

Place: Samayapuram

Date:02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Blood Donation Management System is a Java-based desktop application designed to streamline the management of donor registration, history tracking, and blood inventory. Utilizing Swing and AWT for its graphical user interface, the application allows users to register donors by entering details such as name, age, and blood group. Input validation ensures the accuracy and completeness of data. Donor information is stored in an organized list, and a history section displays detailed records for easy reference. The system also tracks blood inventory categorized by blood groups, providing real-time stock updates stored in a hashmap for efficient data handling.This project demonstrates the practical use of object-oriented programming through the implementation of a Donor class, ensuring modularity and extensibility. By integrating core Java concepts with user-friendly design, the application enhances the efficiency and accessibility of blood donation processes. It serves as a valuable tool for blood banks and donation centers, enabling centralized management of donor and inventory information while fostering a streamlined approach to blood donation activities.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Blood Donation Management System is a Java-based desktop application that simplifies donor registration, history tracking, and blood inventory management. Built with Swing and AWT, it features a user-friendly interface for registering donors, validating inputs, and displaying donor details alongside real-time blood stock updates. Using object-oriented principles and efficient data structures like hashmaps, the system ensures modularity and streamlined operations, making it a valuable tool for blood banks and donation centers to enhance efficiency and accessibility in managing blood donation activities.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Objective

The objective of the Blood Donation Management System is to create an efficient and interactive platform that simplifies the management of blood donation processes. This system is designed to register donors by collecting essential information such as name, age, and blood group, ensuring data accuracy through input validation. It also maintains a centralized record of donor history, enabling easy tracking and retrieval of information. Additionally, the system incorporates a blood inventory management feature to monitor the availability of various blood groups in real-time, helping blood banks maintain optimal stock levels. By leveraging Java's graphical user interface components, the application offers a user-friendly experience, making it accessible to a wide range of users. The system focuses on enhancing the operational efficiency of blood banks and donation centers, streamlining donor registration, improving inventory visibility, and ensuring the reliability of blood donation activities. Through its modular design and robust data handling, the application serves as a valuable tool for addressing the organizational challenges associated with blood donation.

1.2. Overview

The Blood Donation Management System is a Java-based desktop application developed using Swing and AWT to efficiently manage the donor registration process, blood inventory tracking, and donor history. The system enables users to register new blood donors by entering their name, age, and blood group, while ensuring that all input data is validated for correctness and completeness. Donor information is stored in a list, and users can view a detailed history of all registered donors, including their personal details and blood group.In addition to managing donor records, the system also tracks the blood inventory, categorizing blood by its group and displaying real-time stock levels. The blood inventory is stored using a hashmap to ensure quick updates and retrieval. The application also incorporates error handling, providing users with clear feedback if incorrect or incomplete data is entered. By integrating these features into a single platform, the Blood Donation Management System enhances the efficiency of blood donation management, making it a useful tool for blood banks and donation centers.

1.3 Java Programming Concepts

The basic concepts of Object-Oriented Programming (OOP) are:

- **1. Object-Oriented Programming (OOP):** A programming paradigm that uses objects and classes to structure software, promoting code reusability, modularity, and encapsulation.
- **2. Data Structures:** Organizing and storing data efficiently in structures like arrays, lists, and maps to enable quick access, modification, and storage.
- **3. Event Handling:** A mechanism in programming that allows a program to respond to user actions, such as clicks or keystrokes, by triggering specific functions.
- **4. GUI Programming:** The process of designing graphical user interfaces that allow users to interact with software through visual elements like buttons, text fields, and menus.
- **5. Exception Handling:** A technique for managing runtime errors by catching and handling exceptions to prevent the program from crashing.
- **7. Data Validation:** The process of checking if the data entered by users meets the required format and constraints before processing it.
- **8. String Manipulation:** The process of modifying, parsing, or formatting strings (text) to suit specific needs in a program.

Project related concepts:

- **Object-Oriented Programming (OOP)**

  The system is built around object-oriented principles, with the Donor class encapsulating the donor's details like name, age, and blood group. This approach ensures that data is securely handled and can be easily accessed or modified through methods, promoting modularity and reusability in code.

- **Data Structures**

  The use of an ArrayList allows dynamic storage of donors, enabling easy addition and removal of records. A HashMap is utilized to efficiently manage blood inventory by mapping blood groups to their respective stock levels, ensuring quick lookups and updates.

- **Exception Handling**

  Exception handling is employed to manage potential errors during user input. For instance, if a user enters an invalid age, the system catches the NumberFormatException and

displays a friendly error message, ensuring smooth user interaction without crashing the program.

- **Event Handling**

  Event handling is implemented using ActionListener, making the system responsive to user actions like button clicks. When a user clicks the "Register Donor" or "View Donor History" button, appropriate actions are triggered, allowing for dynamic interaction with the interface.

- **GUI Programming**

  Swing and AWT are used to create a user-friendly graphical interface, with components like buttons, text fields, and lists. GridBagLayout is employed for flexible, organized positioning of these components, ensuring a neat and intuitive layout for the user.

- **Access Modifiers**

  Access modifiers like private are used to restrict direct access to critical data in the Donor class. This ensures data security and allows for controlled access to the data, promoting better encapsulation and code maintainability

- **String Manipulation**

  String manipulation is used to format and display donor information clearly in the user interface. For instance, donor details are concatenated into readable strings, which are then shown in the donor history area or the donor list, providing users with easily understandable data.

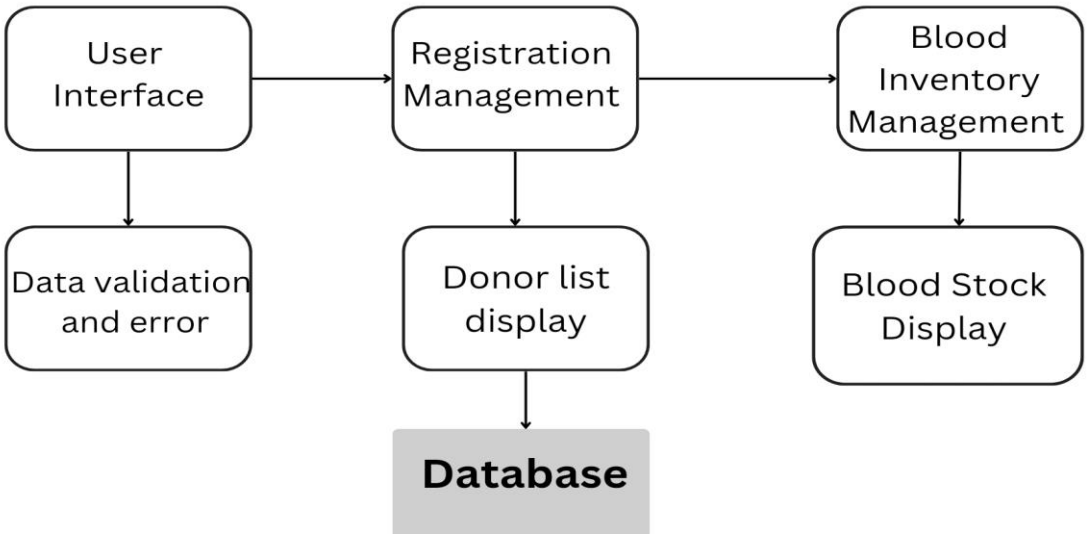
CHAPTER 2

PROJECT METHODOLOGY

2.1. Proposed Work

The proposed Blood Donation Management System will streamline donor registration and blood inventory tracking for blood banks. It will allow users to register donors, store their details, and manage blood stock by categorizing blood groups. A user-friendly GUI will be developed using Java Swing and AWT to ensure easy interaction. Data validation and error handling will be implemented to maintain data integrity and system stability. The system will be scalable, allowing future additions like donation history tracking and reporting features. Overall, it aims to improve blood donation management and support better practices in blood donation centers.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 User Interface Module

The user interface is developed using Java Swing and AWT libraries to provide an intuitive and interactive graphical environment. It includes components like text fields, buttons, labels, and lists, which allow users to register donors, view donor history, and manage blood inventory. The layout is managed using GridBagLayout, ensuring a flexible and organized design. This module is critical for facilitating user interactions with the system and ensuring ease of use for non-technical users.

3.2 Donor Registration Module

This module handles the process of adding new donors to the system. It collects essential information such as name, age, and blood group, validates the data for completeness and correctness, and stores it in an ArrayList. If the input is invalid, it uses exception handling to notify the user with appropriate error messages. This module ensures the accuracy and security of donor information, making it foundational for managing records.

3.3 Blood Inventory Management Module

This module maintains the stock levels of different blood groups using a HashMap. Each blood group is mapped to its respective quantity, which can be updated based on donations. It provides real-time data on blood availability, enabling efficient inventory tracking and management.

3.4. Data Validation and Error Handling Module

This module ensures that user inputs meet required constraints, such as non-empty fields and valid numeric age entries. If an error occurs, such as invalid data, it uses try-catch blocks to catch exceptions and provide user-friendly feedback via pop-up messages.

3.5Module 5 Heading

This module retrieves and displays the list of registered donors along with their details, including name, age, and blood group. It uses Java Swing components like JList and JTextArea for a clear and organized presentation of donor data.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Blood Donation Management System is a software solution aimed at streamlining donor registration, blood inventory tracking, and donor history management. It provides a user-friendly GUI built with Java Swing and AWT, ensuring seamless interaction. The system allows users to register donors by collecting their name, age, and blood group while validating inputs to maintain data integrity. Blood inventory is efficiently managed using a HashMap, with real-time updates categorized by blood groups. An ArrayList stores donor data, ensuring easy access and manipulation. The system incorporates robust data validation and error handling to prevent disruptions due to incorrect inputs. Donor details and inventory information are clearly displayed, enhancing operational efficiency. Designed for scalability, the system can include future features like donation tracking and report generation. It addresses operational challenges faced by blood banks and ensures better resource allocation. The application promotes efficient blood donation management, ensuring timely availability of resources. Overall, it contributes significantly to life-saving efforts by supporting well-organized donation practices.

4.2 FUTURE SCOPE

The Blood Donation Management System offers significant potential for future enhancements to improve its utility and efficiency. Integrating with a centralized database can enable real-time synchronization of donor and inventory data across multiple blood banks, ensuring seamless resource sharing. Features like donation history tracking, automated reporting, and advanced analytics can provide deeper insights and streamline operations. Additionally, mobile and web integration can allow donors to register remotely, receive reminders, and check eligibility for donations. AI algorithms could predict blood demand, ensuring optimal inventory levels, while IoT integration can automate stock updates directly from storage units.

REFERENCES

BOOKS :

- Java: A Beginner's Guide by Herbert Schildt – For understanding Java programming basics and advanced concepts.

- Effective Java by Joshua Bloch – For learning best practices in Java programming and design.

- Core Java Volume I - Fundamentals by Cay S. Horstmann and Gary Cornell – For mastering Java fundamentals and GUI development.

- Head First Java by Kathy Sierra and Bert Bates – For an engaging introduction to Java concepts, including Swing and event handling.

ONLINE RESOURCES :

- GeeksforGeeks – Tutorials and examples for Java development, data structures, and algorithms.

- W3Schools Java Tutorials – For learning Java programming concepts with practical examples.

- TutorialsPoint – Comprehensive guides on Java, Swing, and AWT.

- Stack Overflow – For troubleshooting and exploring solutions to common Java development issues.

APPENDIX A
(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.ArrayList;
import java.util.HashMap;
public class BloodDonationSystem extends JFrame {
    // Defining the fields for registration and donor history
    private JTextField nameField, ageField, bloodGroupField;
    private JButton registerButton, viewButton;
    private JList<String> donorList;
    private JTextArea donorHistoryArea;
    private DefaultListModel<String> donorListModel;
     private static final ArrayList<Donor> donors = new ArrayList<>();
    private static final HashMap<String, Integer> bloodInventory = new HashMap<>();
     public BloodDonationSystem() {
        setTitle("Blood Donation Management System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        // Initializing components
        nameField = new JTextField(15);
        ageField = new JTextField(15);
        bloodGroupField = new JTextField(15);
       registerButton = new JButton("Register Donor");
       viewButton = new JButton("View Donor History");
       donorListModel = new DefaultListModel<>();
       donorList = new JList<>(donorListModel);
       donorHistoryArea = new JTextArea(10, 30);
       donorHistoryArea.setEditable(false);
      // Adding components to the frame with alignment
        gbc.insets = new Insets(5, 5, 5, 5); // Padding
        gbc.anchor = GridBagConstraints.WEST;
       gbc.gridx = 0;
        gbc.gridy = 0;
      add(new JLabel("Name:"), gbc);
       gbc.gridx = 1;
```

```
    add(nameField, gbc);
    gbc.gridx = 0;
    gbc.gridy = 1;
    add(new JLabel("Age:"), gbc);
    gbc.gridx = 1;
    add(ageField, gbc)
    gbc.gridx = 0;
    gbc.gridy = 2;
    add(new JLabel("Blood Group:"), gbc);
    gbc.gridx = 1;
    add(bloodGroupField, gbc);
    gbc.gridx = 0;
    gbc.gridy = 3;
    add(registerButton, gbc);
    gbc.gridx = 1;
    add(viewButton, gbc);
    gbc.gridx = 0;
    gbc.gridy = 4;
    gbc.gridwidth = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    add(new JLabel("Registered Donors:"), gbc);
    gbc.gridy = 5;
    add(new JScrollPane(donorList), gbc);
    gbc.gridy = 6;
    add(new JLabel("Donor History:"), gbc);
    gbc.gridy = 7;
    add(new JScrollPane(donorHistoryArea), gbc);
    // Adding action listeners
    registerButton.addActionListener(e -> registerDonor());
    viewButton.addActionListener(e -> viewDonorHistory());
   // Initializing the blood inventory
    bloodInventory.put("A+", 10);
    bloodInventory.put("O+", 8);
    bloodInventory.put("B+", 6);
    bloodInventory.put("AB+", 4);
     pack();
    setLocationRelativeTo(null);
    setVisible(true); }

private void registerDonor() {
```

```
        String name = nameField.getText();
        String ageText = ageField.getText();
        String bloodGroup = bloodGroupField.getText();
        if (name.isEmpty() || ageText.isEmpty() || bloodGroup.isEmpty()) {
            showMessage("Please fill in all fields.");
            return;
        }
       try {
            int age = Integer.parseInt(ageText);

            Donor donor = new Donor(name, age, bloodGroup);
            donors.add(donor);
            donorListModel.addElement(name + " (" + bloodGroup + ")");
            nameField.setText("");
            ageField.setText("");
            bloodGroupField.setText("");
             showMessage("Donor registered successfully!");
        } catch (NumberFormatException e) {
            showMessage("Invalid age. Please enter a valid number.");
        }
    }
        private void viewDonorHistory() {
        donorHistoryArea.setText("");
        for (Donor donor : donors) {
            donorHistoryArea.append(donor.toString() + "\n");
        }
        donorHistoryArea.append("\nBlood Inventory:\n");
        for (String bloodGroup : bloodInventory.keySet()) {
            donorHistoryArea.append(bloodGroup + ": " + bloodInventory.get(bloodGroup) + "
units\n");
        }
    }

    private void showMessage(String message) {
        JOptionPane.showMessageDialog(this, message);
    }
    public static void main(String[] args) {
        new BloodDonationSystem();
    }
}
```

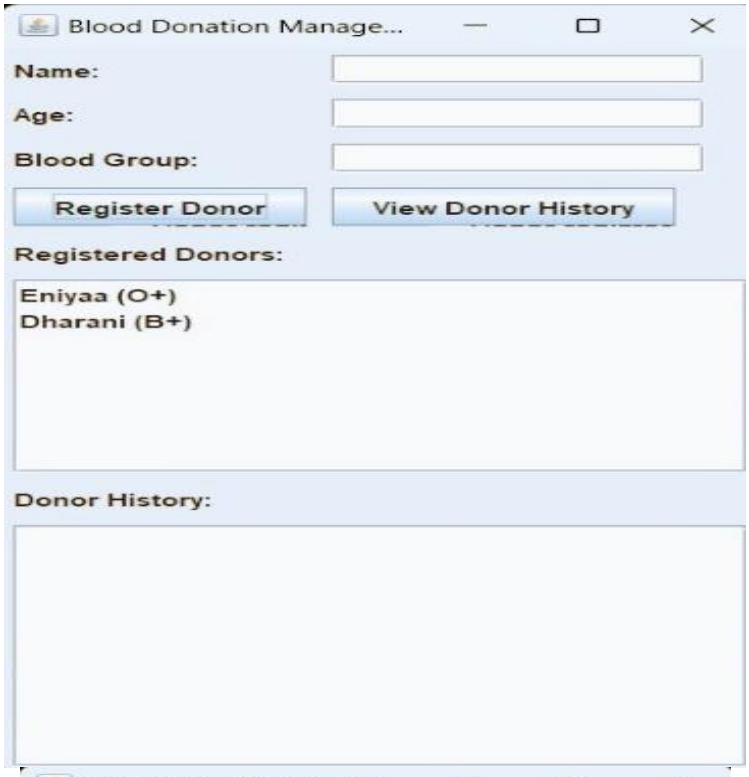
```
class Donor {
    private final String name;
    private final int age;
    private final String bloodGroup;
    public Donor(String name, int age, String bloodGroup) {
        this.name = name;
        this.age = age;
        this.bloodGroup = bloodGroup;
    }

    public String getName() {
        return name;
    }

    public String getBloodGroup() {
        return bloodGroup;
    }

    @Override
    public String toString() {
        return name + " (" + bloodGroup + ") - Age: " + age;
    }
}
```

APPENDIX B(SCREENSHOTS)


Blood Donation Manage...
Name:
Age:
Blood Group:
Register Donor
View Donor History
Registered Donors:
Eniyaa (O+)
Dharani (B+)
Donor History:


Blood Donation Manage...
Name:
Age:
Blood Group:
Register Donor
View Donor History
Registered Donors:
Eniyaa (O+)
Dharani (B+)
Donor History:
Eniyaa (O+) - Age: 19
Dharani (B+) - Age: 20
Blood Inventory:
B+: 6 units
A+: 10 units
AB+: 4 units
O+: 8 units