

BLOOD DONATION MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

JAYASURYA.L (2303811710421066)

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 the bonafide work of **JAYASURYA.L(2303811710421066)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 03/12/2024

CGB1201-JAVA PROGRAMMING
Mr.MANJAMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

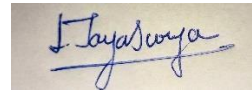
CGB1201-JAVA PROGRAMMING
EXTERNAL EXAMINER
ASSISTANT PROFESSOR

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

A handwritten signature in blue ink, reading "Jayasurya", with a horizontal line underneath.

JAYASURYA.L

Place: Samayapuram

Date: 03/12/2024

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 **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.)**, 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 application supports a fully interactive user interface with options to register a donor, schedule a donation, display the donor list, and view blood inventory. When a donor registers, the system stores the information in a list, and the donor names are automatically sorted alphabetically to maintain an organized database. A donor's details are displayed in a formatted manner, ensuring clarity and ease of access.

The donor registration form includes various input fields, such as text fields for name, age, contact details, and donation date, along with a gender selection using checkboxes. The user is prompted to fill in the required fields, and the system ensures that the inputs are valid. If the input is incorrect or incomplete, appropriate error messages are displayed, prompting the user to correct the information.

The schedule donation option provides users with a simple interface to schedule their next donation. Once a donation is scheduled, a message is displayed confirming the scheduled appointment. Additionally, there is an option to view the complete list of registered donors. The donor list is presented in a text area, and users can easily scroll through the names and details of all registered individuals. The donor list is updated in real-time as new donors are added.

The blood inventory management feature, while currently a placeholder, is designed to be expanded in the future to manage the actual blood stock levels. For now, it displays a message indicating that the feature has not been implemented yet. However, the design accommodates easy future integration for tracking the inventory of various blood groups.

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 application designed to streamline donor registration, scheduling, and inventory management. It offers an intuitive GUI for efficient data handling.	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 NO.	TITLE	PAGE NO.
	ABSTRACT	VIII
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	MODULE DESCRIPTION	4
	3.1 Product Module	4
	3.2 Donar Management Module	4
	3.3 Blood Bank Module	4
	3.4 GUI Module	4
	3.5 Main Module	5
4	CONCLUSION & FUTURE SCOPE	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	13
	REFERENCES	15

CHAPTER 1

INTRODUCTION

1.1 Objective

The Blood Donation Management System is an innovative application designed to streamline the management of blood donation processes and inventory in blood banks. This system empowers individuals to register as donors, schedule donations, and maintain comprehensive records of their donation history. Blood banks, on the other hand, gain access to advanced tools to monitor inventory levels, track donor information, and forecast blood demand. The application ensures transparency by providing real-time updates on blood stock availability, donor status, and donation schedules. By simplifying complex operations, the system reduces administrative overhead and enhances efficiency. Integrated alerts and notifications keep donors informed about upcoming donation drives and encourage timely participation. Additionally, the platform offers analytics for blood banks, enabling better resource allocation and planning to meet critical demand.

1.2 Overview

The **Blood Donation Management System** is a Java-based application utilizing AWT for managing blood donors and donation activities. It includes a Donor class to store details such as name, age, blood group, gender, contact, last donation date, and date of birth, with features for sorting donors alphabetically. The Blood Bank class maintains an organized donor list and facilitates efficient data management. The GUI provides an interactive main menu, allowing users to register donors, schedule donations, view donor lists, and explore inventory options. The registration process validates donor eligibility, ensuring compliance with criteria like a minimum 90-day interval since the last donation.

1.3 Java Programming Concepts

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

Classes and Objects: The Donor and BloodBank classes encapsulate donor details and blood bank operations.

Encapsulation: Private fields in the Donor class with public getters and methods ensure controlled access.

Inheritance: Although not directly used, the application structure adheres to OOP principles for scalability.

☐ **Interfaces and Abstract Classes:**

Comparable Interface: The Donor class implements Comparable<Donor> to facilitate sorting by name using the compareTo method.

☐ **Event-Driven Programming:**

The application uses AWT (Abstract Window Toolkit) to create a GUI and handle user interactions with ActionListener events.

☐ **Error Handling:**

Exception Handling: Input validation and parsing errors, such as invalid date formats, are managed using try-catch blocks.

☐ **Collections Framework:**

ArrayList: The BloodBank class stores donors in a dynamic list.

☐ **Java AWT for GUI Development:**

Components like Frame, Button, TextField, Label, Checkbox, and TextArea build an interactive user interface.

☐ **String Manipulation:**

Data like blood group input is normalized using string methods (e.g., toUpperCase).

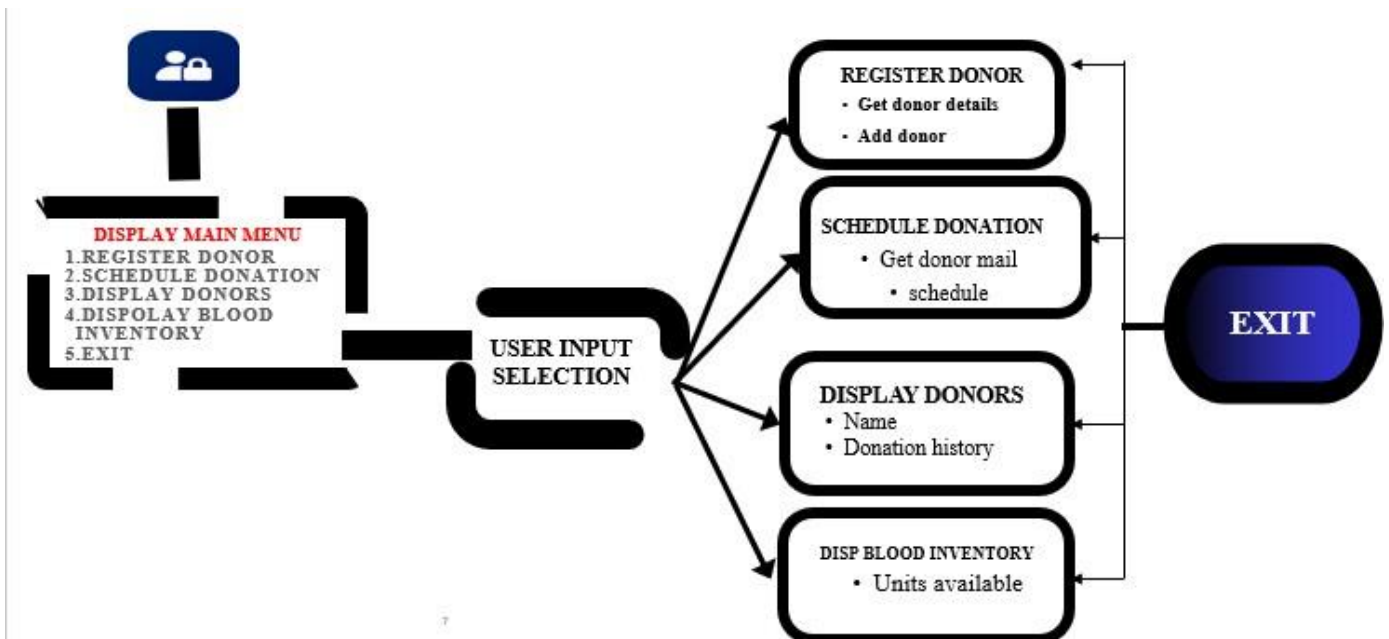
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The Blood Donation Management System focuses on developing an efficient, user-friendly platform to streamline donor registration, scheduling, and blood inventory management. The system will allow individuals to register as donors, verify eligibility based on donation history, and schedule donations seamlessly. Blood banks can maintain an organized database of donors, track inventory, and generate reports for better resource allocation. A robust GUI built using Java AWT will provide intuitive navigation and real-time updates. Advanced features like automated notifications, analytics, and integrated inventory tracking will be incorporated to enhance functionality. The system ensures transparency, reliability, and scalability, ultimately supporting life-saving blood donation initiatives.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Product Module:

Represents each product in the inventory with attributes: id, name, quantity, and price. Implements encapsulation by keeping fields private and providing public getters and setters for controlled access. Includes the Serializable interface to enable saving objects to a file. The toString method provides a readable string representation of product details for display.

3.2 Donor Management Module:

Handles the creation and storage of donor information using the Donor class. Includes attributes like name, age, blood group, gender, contact details, date of birth, and last donation date. Implements sorting functionality using the Comparable interface for alphabetical listing of donors.

3.3 Blood Bank Module:

Manages the database of donors using the BloodBank class, which utilizes a dynamic list (ArrayList) to store and retrieve donor data. Supports adding donors and viewing the donor list with organized data handling.

3.4 GUI Module:

Added Built using Java AWT components such as Frame, Button, TextField, Label, TextArea, and Checkbox to create an interactive interface. Provides functionalities like donor registration, scheduling donations, viewing donor lists, and navigating the main menu.

3.5 Main Modules:

Built Registration Module:

Collects and validates donor information through a registration form.

Ensures eligibility by checking donation intervals using date validation.

Scheduling Module:

Displays scheduling confirmation and guides users on the donation process.

Display Module:

Shows donor information in a formatted manner using TextArea.

Placeholder functionality for blood inventory display to be implemented later.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Blood Donation Management System is an efficient and user-friendly platform for streamlining blood donation processes, developed using Java programming principles. The code demonstrates a well-structured approach with clear modularity, incorporating donor management, registration, scheduling, and display functionalities. The system effectively uses object-oriented programming concepts, including encapsulation and interface implementation, ensuring organized and scalable data handling. The GUI, designed with AWT, provides an intuitive interface for seamless interaction, allowing users to register, schedule donations, and view donor lists with ease. Error handling ensures data validation, while sorting and real-time updates enhance operational efficiency. Although the blood inventory management is a placeholder, the current implementation establishes a strong foundation for future expansions. By automating key processes and promoting transparency, the system minimizes manual effort and supports blood banks in maintaining a steady supply of resources. With advanced features like notifications and analytics planned, the project is a vital step toward modernizing blood donation initiatives, ensuring reliability and accessibility for life-saving efforts.

4.2 FUTURE SCOPE

The Blood Donation Management System has significant potential for future enhancements to improve its functionality and reach. Key advancements include implementing automated notifications to remind donors of their next eligible donation date and alert blood banks about low inventory levels. A real-time inventory management module can be added to dynamically track available blood types and expiry dates. Advanced analytics can predict blood demand trends, optimize collection drives, and identify high-need areas. Expanding the application to mobile and web platforms will enhance accessibility, while integration with healthcare systems can provide real-time access to donor databases for hospitals and emergency services. Secure cloud storage will ensure data protection and accessibility. Additional features like donor engagement programs, multi-language support, AI-based donor-recipient matching, and integration with wearable devices for health monitoring will further modernize the system, fostering a more efficient, reliable, and inclusive platform for blood donation initiatives.

APPEDDIX A (SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.List;

class Donor implements Comparable<Donor> {
    private String name;
    private int age;
    private String bloodGroup;
    private String contactNumber;
    private String lastDonationDate;
    private String gender;
    private String dateOfBirth;

    public Donor(String name, int age, String bloodGroup, String contactNumber, String
lastDonationDate, String gender, String dateOfBirth) {
        this.name = name;
        this.age = age;
        this.bloodGroup = bloodGroup;
        this.contactNumber = contactNumber;
        this.lastDonationDate = lastDonationDate;
        this.gender = gender;
        this.dateOfBirth = dateOfBirth;
    }

    public String getName() {
        return name;
    }

    @Override
    public int compareTo(Donor other) {
        return this.name.compareToIgnoreCase(other.name);
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Age: " + age + ", Blood Group: " + bloodGroup +
            ", Gender: " + gender + ", Contact: " + contactNumber +
            ", Last Donation: " + lastDonationDate + ", DOB: " + dateOfBirth;
```

```
}  
}
```

```
class BloodBank {  
    private List<Donor> donors = new ArrayList<>();  
  
    public void addDonor(Donor donor) {  
        donors.add(donor);  
        Collections.sort(donors); // Maintain donors in alphabetical order  
    }  
  
    public List<Donor> getDonors() {  
        return donors;  
    }  
}
```

```
public class OnlineBloodDonorManagementAWT {  
    private static BloodBank bloodBank = new BloodBank();  
    private static TextArea displayArea;  
    private static TextField nameField, ageField, bloodGroupField, contactField, lastDonationField,  
    dobField;  
    private static CheckboxGroup genderGroup;  
    private static Checkbox maleCheckbox, femaleCheckbox;  
  
    public static void main(String[] args) {  
        displayMainMenu();  
    }  
}
```

```
private static void displayMainMenu() {  
    Frame frame = new Frame("Online Blood Donor Management");  
    frame.setLayout(new GridLayout(6, 1, 10, 10));  
  
    Label menuLabel = new Label("DISPLAY MAIN MENU", Label.CENTER);  
    menuLabel.setFont(new Font("Arial", Font.BOLD, 16));  
  
    Button registerButton = new Button("1. Register Donor");  
    Button scheduleDonationButton = new Button("2. Schedule Donation");  
    Button displayDonorsButton = new Button("3. Display Donors");  
    Button displayInventoryButton = new Button("4. Display Blood Inventory");  
    Button exitButton = new Button("5. Exit");  
  
    frame.add(menuLabel);  
    frame.add(registerButton);  
    frame.add(scheduleDonationButton);  
    frame.add(displayDonorsButton);  
    frame.add(displayInventoryButton);  
    frame.add(exitButton);  
}
```

```

registerButton.addActionListener(e -> {
    frame.dispose();
    displayRegisterForm();
});

scheduleDonationButton.addActionListener(e -> {
    frame.dispose();
    displayScheduleDonation();
});

displayDonorsButton.addActionListener(e -> {
    frame.dispose();
    displayDonorList();
});

displayInventoryButton.addActionListener(e -> {
    frame.dispose();
    displayBloodInventory();
});

exitButton.addActionListener(e -> System.exit(0));

frame.setSize(400, 300);
frame.setVisible(true);
}

private static void displayRegisterForm() {
    Frame frame = new Frame("Register Donor");

    frame.setLayout(new GridLayout(10, 2, 10, 10));

    Label nameLabel = new Label("Name:");
    Label ageLabel = new Label("Age:");
    Label bloodGroupLabel = new Label("Blood Group:");
    Label contactLabel = new Label("Contact Number:");
    Label lastDonationLabel = new Label("Last Donation Date (dd/MM/yyyy):");
    Label genderLabel = new Label("Gender:");
    Label dobLabel = new Label("Date of Birth (dd/MM/yyyy):");

    nameField = new TextField();
    ageField = new TextField();
    bloodGroupField = new TextField();
    contactField = new TextField();
    lastDonationField = new TextField();
    dobField = new TextField();

```

```

genderGroup = new CheckboxGroup();
maleCheckbox = new Checkbox("Male", genderGroup, false);
femaleCheckbox = new Checkbox("Female", genderGroup, false);

Button registerDonorButton = new Button("Register Donor");
Button backButton = new Button("Back");

displayArea = new TextArea();

frame.add(nameLabel);
frame.add(nameField);

frame.add(ageLabel);
frame.add(ageField);

frame.add(bloodGroupLabel);
frame.add(bloodGroupField);

frame.add(contactLabel);
frame.add(contactField);

frame.add(lastDonationLabel);
frame.add(lastDonationField);

frame.add(genderLabel);
Panel genderPanel = new Panel();
genderPanel.add(maleCheckbox);
genderPanel.add(femaleCheckbox);
frame.add(genderPanel);

frame.add(dobLabel);
frame.add(dobField);

frame.add(registerDonorButton);
frame.add(backButton);

registerDonorButton.addActionListener(e -> registerDonor());
backButton.addActionListener(e -> {
    frame.dispose();
    displayMainMenu();
});

frame.setSize(600, 500);
frame.setVisible(true);
}

private static void registerDonor() {

```

```

try {
    String name = nameField.getText();
    int age = Integer.parseInt(ageField.getText());
    String bloodGroup = bloodGroupField.getText().toUpperCase();
    String contactNumber = contactField.getText();
    String lastDonationDate = lastDonationField.getText();
    String gender = genderGroup.getSelectedCheckbox() == maleCheckbox ? "Male" :
"Female";
    String dob = dobField.getText();

    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
    Date lastDonation = sdf.parse(lastDonationDate);
    Date currentDate = new Date();

    if ((currentDate.getTime() - lastDonation.getTime()) / (1000 * 60 * 60 * 24) < 90) {
        displayArea.setText("Last donation must be at least 90 days ago.");
        return;
    }

    Donor donor = new Donor(name, age, bloodGroup, contactNumber, lastDonationDate,
gender, dob);
    bloodBank.addDonor(donor);
    displayArea.setText("Donor registered successfully!");
} catch (Exception e) {
    displayArea.setText("Invalid input! Please check all fields.");
}
}

private static void displayScheduleDonation() {
    Frame frame = new Frame("Schedule Donation");
    frame.setLayout(new GridLayout(2, 1, 10, 10));

    Label infoLabel = new Label("Donation scheduled. Please visit the blood bank.",
Label.CENTER);
    Button backButton = new Button("Back");

    frame.add(infoLabel);
    frame.add(backButton);

    backButton.addActionListener(e -> {
        frame.dispose();
        displayMainMenu();
    });

    frame.setSize(400, 200);
    frame.setVisible(true);
}

```

```

private static void displayDonorList() {
    Frame frame = new Frame("Donor List");
    frame.setLayout(new BorderLayout());

    TextArea donorList = new TextArea();
    donorList.setEditable(false);

    for (Donor donor : bloodBank.getDonors()) {
        donorList.append(donor.toString() + "\n");
    }

    Button backButton = new Button("Back");
    backButton.addActionListener(e -> {
        frame.dispose();
        displayMainMenu();
    });

    frame.add(donorList, BorderLayout.CENTER);
    frame.add(backButton, BorderLayout.SOUTH);

    frame.setSize(500, 400);
    frame.setVisible(true);
}

private static void displayBloodInventory() {
    Frame frame = new Frame("Blood Inventory");
    frame.setLayout(new BorderLayout());

    TextArea inventoryList = new TextArea();
    inventoryList.setEditable(false);

    inventoryList.append("Blood inventory management is not yet implemented.\n");

    Button backButton = new Button("Back");
    backButton.addActionListener(e -> {
        frame.dispose();
        displayMainMenu();
    });

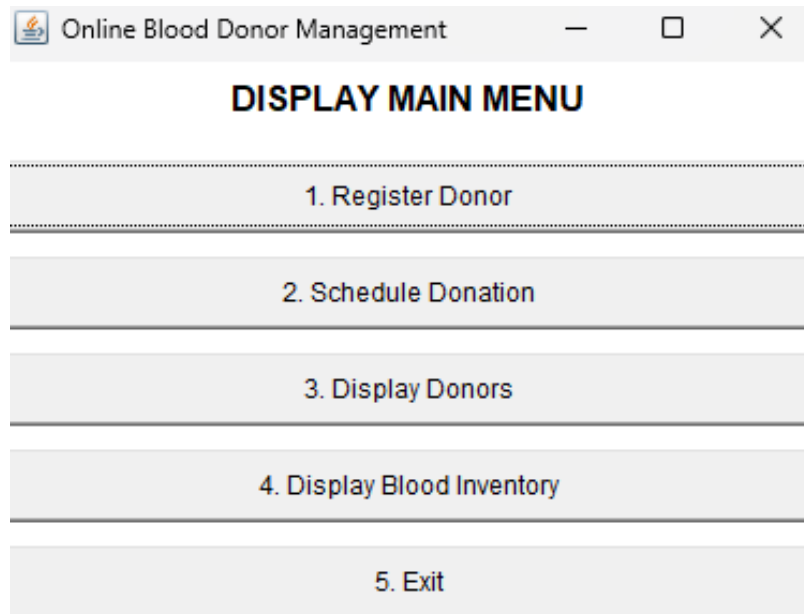
    frame.add(inventoryList, BorderLayout.CENTER);
    frame.add(backButton, BorderLayout.SOUTH);

    frame.setSize(400, 300);
    frame.setVisible(true);
}
}

```

APPENDIX B

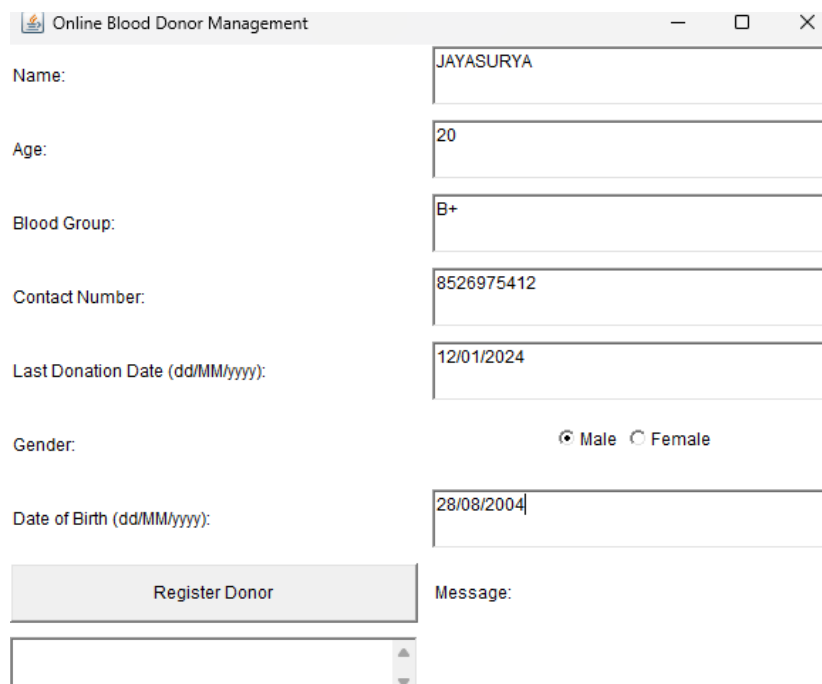
(SCREENSHOTS)



Online Blood Donor Management

DISPLAY MAIN MENU

1. Register Donor
2. Schedule Donation
3. Display Donors
4. Display Blood Inventory
5. Exit



Online Blood Donor Management

Name: JAYASURYA

Age: 20

Blood Group: B+

Contact Number: 8526975412

Last Donation Date (dd/MM/yyyy): 12/01/2024

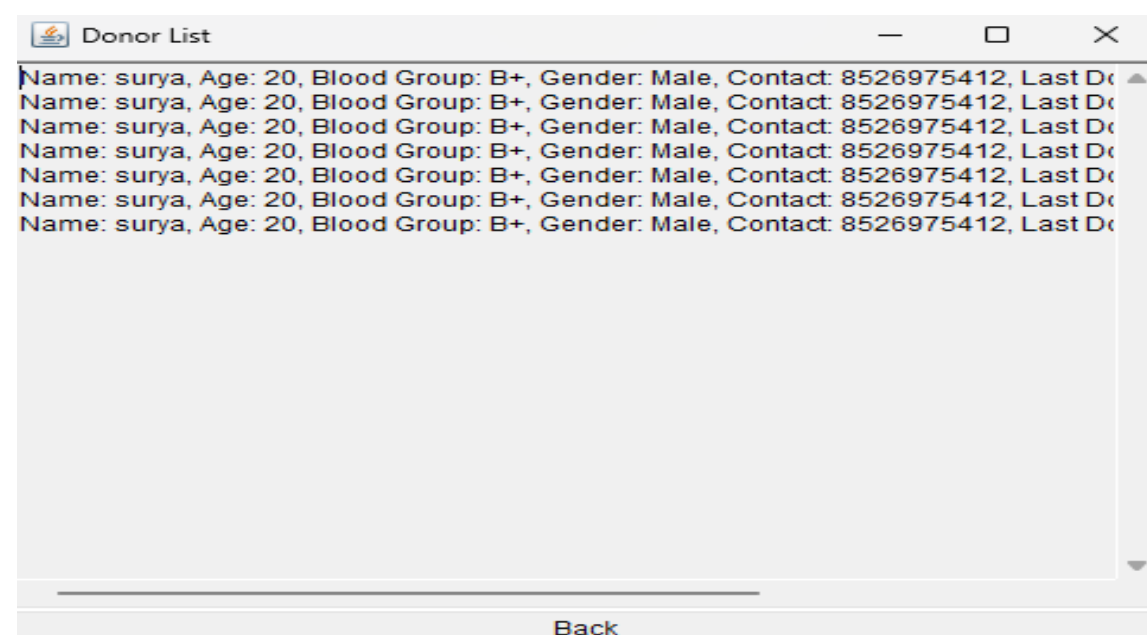
Gender: ☒ Male ☐ Female

Date of Birth (dd/MM/yyyy): 28/08/2004

Message:



Donation scheduled. Please visit the blood bank.



REFERENCES

❖ **Books:**

Head First Java by Kathy Sierra and Bert Bates.

Java: The Complete Reference by Herbert Schildt.

❖ **Websites:**

Oracle Java Documentation - Tutorials on AWT, serialization, and multithreading.

GeeksforGeeks - Examples of file handling and synchronization.

Baeldung - Guides on Java serialization and file handling.