M.Kumarasamy
College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

BLOOD DONATION MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Ms. B. SHARMILADEVI M.E.,

Assistant Professor / EEE

Submitted By

LIBIKA S (927624BEE057)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

DECEMBER 2025

M.Kumarasamy
College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **“BLOOD DONATION MANAGEMENT SYSTEM ”** is the bonafide work of **LIBIKA S (927624BEE057)** who carried out the project work during the academic year 2025– 2026 under my supervision.

Signature	Signature
Ms. B. SHARMILADEVI M.E.,	**Dr. J. UmaM.E.,Ph.D.,**
SUPERVISOR,	**HEAD OF THE DEPARTMENT,**
Department of Electricaland ElectronicsEngineering,	Department of Electricaland ElectronicsEngineering,
M.Kumarasamy College of Engineering,	M.Kumarasamy College of Engineering,
Thalavapalayam, Karur -639 113.	Thalavapalayam, Karur -639 113.

VISIONANDMISSIONOFTHEINSTITUTION

VISION

- ✓ Toemergeasaleaderamongthetopinstitutionsinthefieldoftechnicaleducation

MISSION

- ✓ ProducesmarttechnocratswithempiricalknowledgewhocansurmounttheglobalChallenges.
- ✓ Createadiverse,fully-engaged,learner-centriccampusenvironmenttoprovideQuality education tothe students.
- ✓ Maintainmutuallybeneficialpartnershipswithouralumni,industry,andProfessionalassociations.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

VISION

- ✓ To create competent Electrical and Electronics Engineers who will work in Emerging technologies thereby contributing value to their career and society

MISSION

- ✓ **DM1:** Impart quality education to enhance knowledge and skills in emerging technologies.
- ✓ **DM2:** Create an innovation and product development ecosystem in core and Interdisciplinary Industrial applications leading to patents.
- ✓ **DM3:** Augment student core competency and soft skills in handling technical projects.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

- ✓ **PEO 1 – Core Engineering Proficiency**

  Apply strong fundamentals in electrical and electronics engineering to solve complex problems in power systems, drives and instrumentation.
- ✓ **PEO 2 – Research and Emerging Technologies**

  Pursue research and adopt emerging technologies to innovate in core and interdisciplinary domains.
- ✓ **PEO 3 – Industry Readiness**

  Demonstrate technical and professional skills for successful careers in industry and entrepreneurship.

- ✓ **PEO 4 – Ethics and Societal Responsibility**
  Practice engineering with ethics, sustainability and a commitment to societal well-being.
- ✓ **PEO 5 – Lifelong Learning and Global Competence**
  Engage in continuous learning and adapt to global technological advancements.

PROGRAMMEOUTCOMES(POs)

AfterthesuccessfulcompletionoftheB.E.ElectricalandElectronicsEngineeringdegreeprogram,t hestudents will beable to:

- ✓ **PO1 Engineering knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals, and an engineering specialization to develop solutions to complex engineering problems.
- ✓ **PO2 Problem analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development.
- ✓ **PO3 Design/development of solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required.
- ✓ **PO4 Conduct investigations of complex problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.
- ✓ **PO5 Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.
- ✓ **PO6 The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment.
- ✓ **PO7 Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws.
- ✓ **PO8 Individual and CollaborativeTeam work**: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

✓ **PO9 Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

✓ **PO10 Project management and finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

✓ **PO11 Life-long learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

ThefollowingaretheProgramSpecificOutcomesofEngineeringStudents:

✓ **PSO 01: Power Systems Engineering**

Apply comprehensive knowledge of electrical power systems to analyze, design and manage generation, transmission, and distribution networks, integrating emerging technologies and sustainable practices to contribute effectively to industry and societal needs.

✓ **PSO 02: Power Electronics and Drives**

Design and develop efficient, reliable power electronic systems and motor drives with an emphasis on sustainable energy use, and inclusive technology solutions that address societal and environmental challenges.

✓ **PSO 03: Electronics and Instrumentation Engineering**

Develop intelligent instrumentation and embedded systems for accurate measurements, monitoring and control in interdisciplinary domains addressing societal needs.

ABSTRACT

The Blood Donation Management System is a software application designed to efficiently manage blood donors, blood inventory, and blood requests. The system provides a centralized platform to register donors, track blood availability, and facilitate blood donations and requests in a timely manner. It ensures proper record-keeping of donor information, blood groups, and quantity available, reducing manual errors and improving response during emergencies. Implemented using Java and optionally integrated with a database, the system enhances accessibility, maintains updated inventories, and streamlines communication between donors, recipients, and blood banks. This project aims to promote organized blood donation management, improve resource utilization, and support healthcare services in saving lives.

.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
The Blood Donation Management System is a software tool to manage donor and recipient details, maintain blood stock, and quickly match blood groups. It reduces manual work, saves time in emergencies, and ensures the timely availability of blood through an efficient and user-friendly platform.	POs1 POs2 POs3 POs4 POs5 POs6 POs7 POs8 POs9 POs10 POs11	PSO1 PSO2 PSO3

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

SUPERVISORHEAD OF THE DEPARTMENT

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Blood Donation Management System is to maintain a centralized digital platform that manages donor details, blood stock, and hospital requests efficiently. It aims to ensure quick donor–blood group matching, reduce manual errors, automate emergency alerts, and improve coordination between donors, blood banks, and hospitals for faster and safer blood availability.

1.2 Overview

The Blood Donation Management System is designed to streamline and digitize the entire process of managing blood donations by providing a centralized platform for donor information, blood stock levels, and hospital requests. Its main objective is to ensure quick and accurate matching of donors with required blood groups while reducing manual errors and delays. The system improves coordination between donors, blood banks, and hospitals through real-time updates, automated notifications, and efficient record handling. By maintaining accurate data and enabling faster decision-making, it helps ensure timely, safe, and reliable availability of blood during both regular needs and emergencies.

1.3 Java Programming Concepts

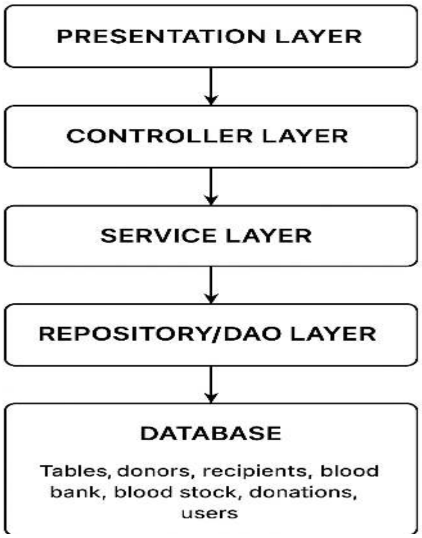
The Blood Donation Management System uses core Java concepts such as OOP (classes, objects, inheritance, polymorphism) to model system entities, encapsulation to protect donor data, exception handling to manage errors, collections to store and access records efficiently, and file handling or JDBC for storing donor and blood inventory information. GUI frameworks like Swing/JavaFX may also be used to create the user interface.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work aims to develop an efficient and user-friendly Blood Donation Management System that digitizes and automates the entire process of managing donor information, blood availability, and hospital requests. The system will provide a centralized database to store donor details, track blood units, and monitor donation history in real time. It will include features for quick donor–blood group matching, automated alerts during emergencies, and easy communication between donors, blood banks, and hospitals. The platform will also generate essential reports to support decision-making and planning of donation drives. By integrating Java-based technologies, the system will ensure accuracy, security, and faster access to critical blood-related information, ultimately improving the speed and reliability of blood management services.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Donor Registration & Information Collection Module

This module collects basic donor details such as name, age, blood group, contact, and medical history. It serves as the starting point of the system and ensures accurate information is stored for future donations**.**

3.2 Blood Stock & Inventory Management Module

This module tracks the available blood units based on blood type and quantity. It updates the inventory after each donation or issue, helping maintain real-time stock for emergency and routine needs.

3.3 Donor Eligibility & Screening Module

This module checks whether a donor is medically fit by verifying age, health status, and donation intervals. It ensures only safe and eligible donors proceed to donate blood.

3.4 Request Handling & Blood Allocation Module

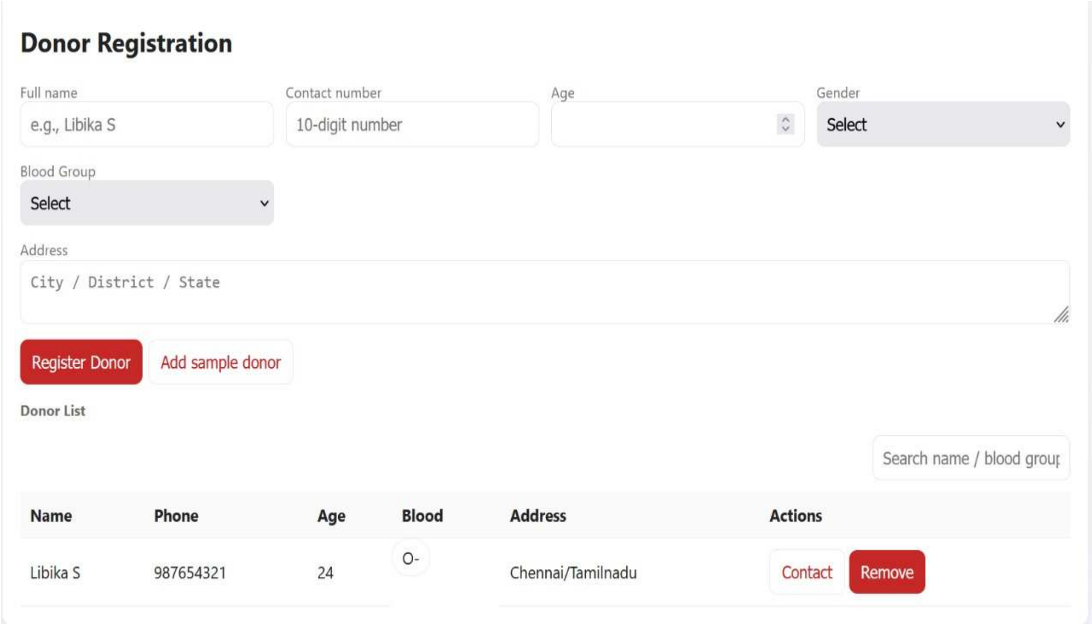
This module manages blood requests from hospitals or patients. It checks stock availability matches required blood groups, and allocates blood units. If stock is low, it alerts the admin for further action.

3.5. Notification & Communication Module

This module sends automated reminders for donation dates, emergency alerts, and blood camp updates, ensuring smooth communication with donors and improving blood availability.

CHAPTER 4

RESULTS AND DISCUSSION



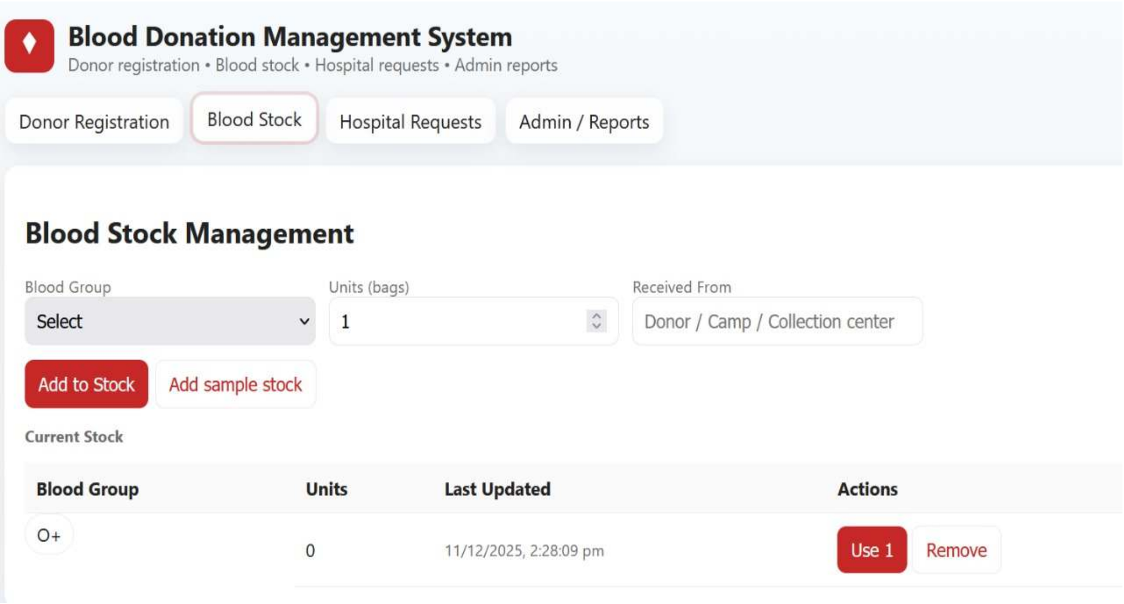
Donor Registration

Full name: e.g., Libika S
Contact number: 10-digit number
Age:
Gender: Select
Blood Group: Select
Address: City / District / State

Donor List

Name	Phone	Age	Blood	Address	Actions
Libika S	987654321	24	O-	Chennai/Tamilnadu	Contact Remove

;



Blood Donation Management System

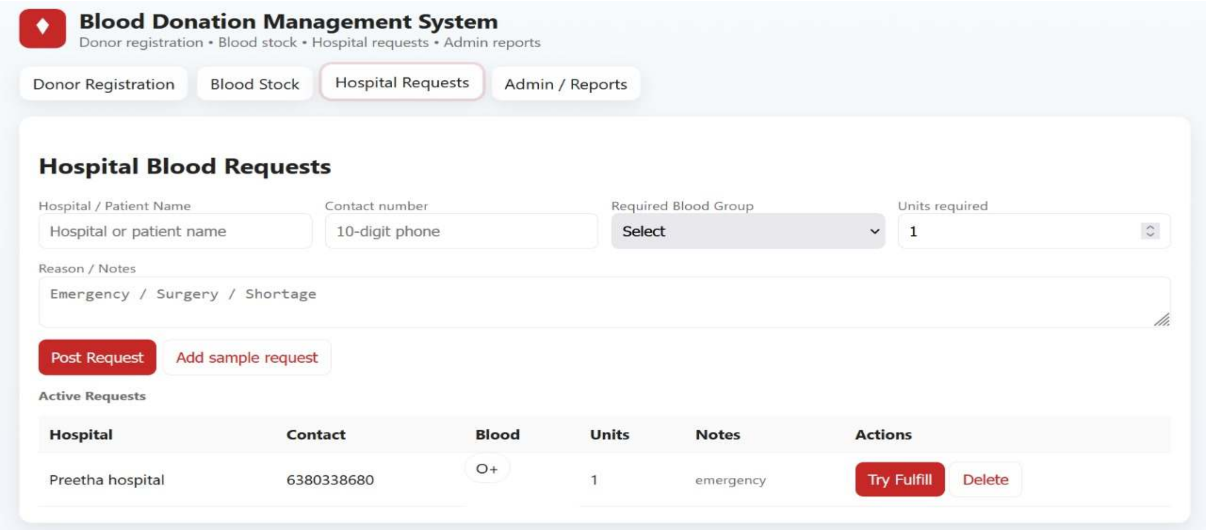
Donor registration • Blood stock • Hospital requests • Admin reports

Blood Stock Management

Blood Group: Select
Units (bags): 1
Received From: Donor / Camp / Collection center

Current Stock

Blood Group	Units	Last Updated	Actions
O+	0	11/12/2025, 2:28:09 pm	Use 1 Remove



Hospital Blood Requests

Hospital / Patient Name: Hospital or patient name

Contact number: 10-digit phone

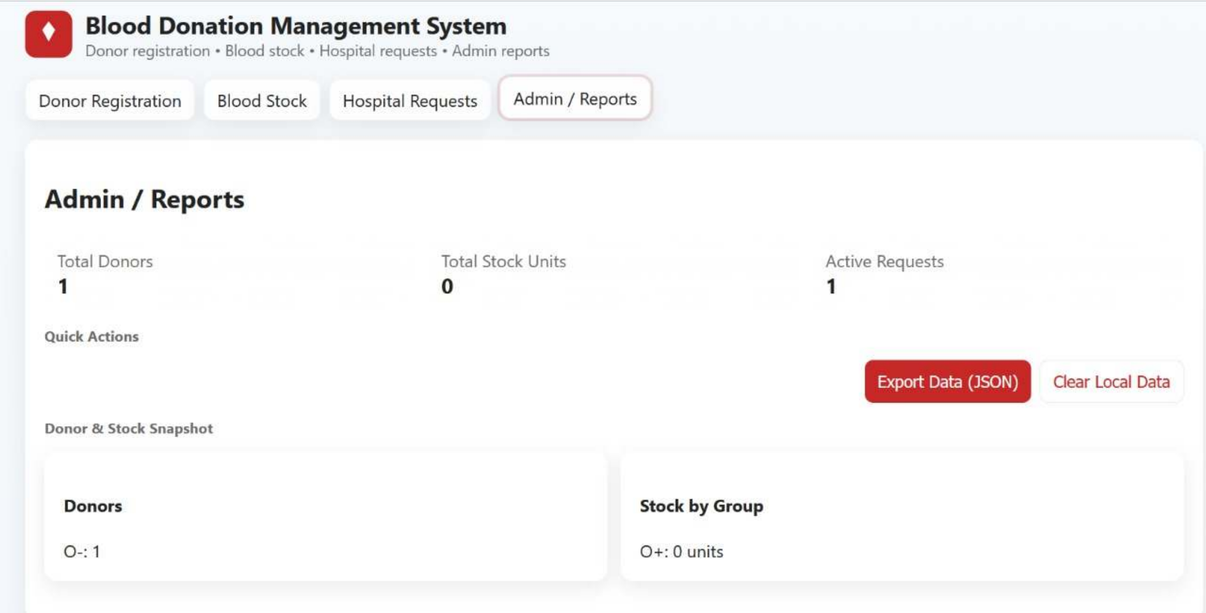
Required Blood Group: Select

Units required: 1

Reason / Notes: Emergency / Surgery / Shortage

Active Requests

Hospital	Contact	Blood	Units	Notes	Actions
Preetha hospital	6380338680	O+	1	emergency	Try Fulfill Delete


Blood Donation Management System
Donor registration • Blood stock • Hospital requests • Admin reports
Donor Registration
Blood Stock
Hospital Requests
Admin / Reports
Export Data (JSON)
Clear Local Data

Admin / Reports

Total Donors	Total Stock Units	Active Requests
1	0	1

Quick Actions

Donor & Stock Snapshot

Donors

O-: 1

Stock by Group

O+: 0 units

DESCRIPTION

The Blood Donation Management System is a digital platform designed to manage and streamline the process of blood donation efficiently. It stores and organizes donor information, blood group details, stock levels, and hospital requests in a secure and centralized database. The system helps users quickly find suitable donors, track blood availability, and maintain accurate donation records. It also automates notifications, ensures proper donor eligibility checks, and improves coordination between donors, blood banks, and hospitals. By providing real-time information and reducing manual work, the system makes the blood management process faster, safer, and more reliable.

CHAPTER 5
CONCLUSION

The development of the Blood Donation Management System successfully addresses the challenges faced by hospitals and blood banks in managing donor records and blood inventory. Through digital automation, the system simplifies donor registration, maintains accurate blood group data, and enables quick retrieval of information during emergencies. It also strengthens communication between donors and medical institutions, ensuring timely updates and efficient utilization of available blood units. By improving accuracy, transparency, and response time, this system plays a vital role in supporting healthcare services and enhancing the overall effectiveness of blood donation and distribution processes.

REFERENCES:

1. Horstmann, C.S., *Core Java Volume I – Fundamentals*, 11th Edition, Prentice Hall, 2019.
2. Schildt, H., *Java: The Complete Reference*, 12th Edition, McGraw-Hill, 2021.
3. Liang, Y.D., *Introduction to Java Programming*, 11th Edition, Pearson, 2018.
4. GeeksforGeeks, *Java Programming Tutorials*, https://www.geeksforgeeks.org/java/.
5. W3Schools, *Java Tutorial*, https://www.w3schools.com/java/.
6. TutorialsPoint, *Java Programming Tutorial*, https://www.tutorialspoint.com/java/index.htm.
7. GitHub, *Blood Bank Management System Projects*, https://github.com/topics/blood-bank-management.
8. Stack Overflow, *Java Programming Q&A*, https://stackoverflow.com/.
9. MySQL Documentation, https://dev.mysql.com/doc/.
10. JavatPoint, *JDBC Example in Java*, https://www.javatpoint.com/example-of-jdbc.

APPENDIX

(Coding)

```
import java.util.ArrayList;
import java.util.Scanner;

// Donor class
class Donor {
    String name;
    int age;
    String bloodGroup;
    String contact;

    Donor(String name, int age, String bloodGroup, String contact) {
        this.name = name;
        this.age = age;
        this.bloodGroup = bloodGroup;
        this.contact = contact;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Age: " + age + ", Blood Group: " + bloodGroup + ", Contact: " +
contact;
    }
}

// Blood Inventory class
class BloodInventory {
    String bloodGroup;
    int quantity;

    BloodInventory(String bloodGroup, int quantity) {
        this.bloodGroup = bloodGroup;
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Blood Group: " + bloodGroup + ", Quantity: " + quantity + " units";
    }
}

public class BloodDonationManagementSystem {
    static ArrayList<Donor> donors = new ArrayList<>();
    static ArrayList<BloodInventory> inventory = new ArrayList<>();

    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        // Initialize blood inventory
        inventory.add(new BloodInventory("A+", 10));
        inventory.add(new BloodInventory("A-", 5));
        inventory.add(new BloodInventory("B+", 8));
        inventory.add(new BloodInventory("B-", 4));
        inventory.add(new BloodInventory("O+", 12));
        inventory.add(new BloodInventory("O-", 6));
        inventory.add(new BloodInventory("AB+", 3));
        inventory.add(new BloodInventory("AB-", 2));

        while (true) {
            System.out.println("\n--- Blood Donation Management System ---");
            System.out.println("1. Add Donor");
            System.out.println("2. View Donors");
            System.out.println("3. View Blood Inventory");
            System.out.println("4. Donate Blood");
            System.out.println("5. Request Blood");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (choice) {
              case 1:
                addDonor();
                break;
              case 2:
                viewDonors();
                break;
              case 3:
                viewInventory();
                break;
              case 4:
                donateBlood();
                break;
              case 5:
                requestBlood();
                break;
              case 6:
                System.out.println("Exiting system...");
                System.exit(0);
                break;
              default:
                System.out.println("Invalid choice! Try again.");
            }
        }
    }

    // Add donor details
```

```
    static void addDonor() {
        System.out.print("Enter Name: ");
        String name = sc.nextLine();
        System.out.print("Enter Age: ");
        int age = sc.nextInt();
        sc.nextLine();
        System.out.print("Enter Blood Group: ");
        String bloodGroup = sc.nextLine();
        System.out.print("Enter Contact Number: ");
        String contact = sc.nextLine();

        donors.add(new Donor(name, age, bloodGroup, contact));
        System.out.println("Donor added successfully!");
    }

    // View all donors
    static void viewDonors() {
        if (donors.isEmpty()) {
            System.out.println("No donors available.");
        } else {
            System.out.println("\n--- Donor List ---");
            for (Donor donor : donors) {
                System.out.println(donor);
            }
        }
    }

    // View blood inventory
    static void viewInventory() {
        System.out.println("\n--- Blood Inventory ---");
        for (BloodInventory b : inventory) {
            System.out.println(b);
        }
```

```
}

// Donate blood (increase inventory)
static void donateBlood() {
    System.out.print("Enter Blood Group to donate: ");
    String bg = sc.nextLine();
    System.out.print("Enter Quantity (units): ");
    int qty = sc.nextInt();
    sc.nextLine();

    for (BloodInventory b : inventory) {
        if (b.bloodGroup.equalsIgnoreCase(bg)) {
            b.quantity += qty;
            System.out.println(qty + " units of " + bg + " blood added.");
            return;
        }
    }
    System.out.println("Blood group not found in inventory!");
}

// Request blood (decrease inventory)
static void requestBlood() {
    System.out.print("Enter Blood Group to request: ");
    String bg = sc.nextLine();
    System.out.print("Enter Quantity (units): ");
    int qty = sc.nextInt();
    sc.nextLine();

    for (BloodInventory b : inventory) {
        if (b.bloodGroup.equalsIgnoreCase(bg)) {
            if (b.quantity >= qty) {
                b.quantity -= qty;
                System.out.println("Request approved. " + qty + " units of " + bg + " blood
provided.");
                } else {
                    System.out.println("Insufficient blood units! Available: " + b.quantity);
                }
                return;
            }
        }
        System.out.println("Blood group not found in inventory!");
    }
}
```