RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM -602 105

CS23333 OOPS Using Java

Laboratory Record Note Book

Name :

Year / Branch / Section :

University Register No. :

College Roll No. :

Semester :

Academic Year :

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

DHARAAN

Name: ……

Academic Year: ……2025-26…… Semester: ……III……
Branch: …CSE……

240701117

Register No.

2407011197

Certified that this is the bonafide record of work done by the above student in the.

Laboratory

during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on……

Internal Examiner External Examiner

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	

10		Collections	
11		Project	
12		Lambda	

BLOOD DONATION MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

BERT GAWIN D 240701075

DHARAAN M N 240701117

DHARSHANA SHRI S 240701119

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"BLOOD DONATION MANAGEMENT**

SYSTEM" is the bonafide work of **"BERT GAWIN D,DHARAAN MN ,DHARSHANA SHRI S"** who carried out the project work under my supervision.

SIGNATURE
B.DEEPA
ASSISTANT PROFESSOR SG

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The **Blood Donation Management System** is a comprehensive web-based application developed using JSP (Java Server Pages) for the frontend and Oracle Database for the backend, implemented in IntelliJ IDEA. It is designed to modernize and streamline the traditional blood donation and inventory management process by introducing automated

donor–recipient matching and real-time blood stock monitoring. When required blood units are unavailable, the system efficiently handles a priority-based FIFO waitlist for recipient requests and automatically notifies suitable donors when a matching requirement arises. The application includes features such as donor registration, eligibility verification, blood group–based search, appointment scheduling, donation history tracking, and secure record management, along with hospital-specific requests and emergency alerts to ensure smooth coordination. By integrating smart queue handling, digital record-keeping, and automated notifications, the system reduces delays in critical situations, enhances communication between donors and blood banks, optimizes blood unit utilization, and ultimately provides a seamless, reliable, and user-friendly platform that strengthens the blood supply chain and improves overall public health responsiveness.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and

our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **B.DEEPA,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. BERT GAWIN D

2. DHARAAN M N

3. DHARSHANASHRI S

TABLE OF CONTENTS

CH. NO.	TITLE	PAGE NO
1	**INTRODUCTION**	**1**
1.1	INTRODUCTION	8
1.2	SCOPE OF THE WORK	8

1.3 PROBLEM STATEMENT 8

1.4 AIM AND OBJECTIVES OF THE PROJECT 8

2 SYSTEM SPECIFICATIONS 9

2.1 HARDWARE SPECIFICATIONS 9

2.2 SOFTWARE SPECIFICATIONS 9

3 MODULE DESCRIPTION 10 4 CODING 11 5 SCREENSHOTS 16 6 CONCLUSION AND FUTURE ENHANCEMENT 18 7 REFERENCES 19

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	SIGN UP	22
5.2	MAIN MENU	22
5.3	DONOR REGISTERATION	23

5.4	**VIEW RECEPIENT**	**23**
5.5	**BLOOD INVENTORY**	**24**
5.6	**HOSPITAL FORM**	**24**

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Blood Donation Management System is a web-based platform that simplifies donor registration, blood requests, and inventory tracking. It helps donors schedule appointments, allows hospitals to request blood, and uses an automated matching and notification system to ensure timely availability of required blood units..

1.2 SCOPE OF THE WORK

The project covers donor registration, authentication, blood group–based search, appointment scheduling, donation tracking, and real-time stock monitoring. It includes a FIFO-based waitlist, automated alerts, emergency

request handling, and a simple, responsive interface for donors and blood banks.

1.3 PROBLEM STATEMENT

Existing blood donation systems rely heavily on manual processes, making it difficult to match donors quickly, check real-time availability,
or respond to emergencies. Without automated alerts or proper tracking, delays, shortages, and inefficient coordination often occur.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to create an efficient and user-friendly system for managing blood donations. The objectives include secure donor authentication, quick matching based on blood group, FIFO waitlist management, automated notifications, appointment scheduling, and real-time inventory updates through a responsive web interface.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel Core i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	JAVA SWING
Back - End	:	Oracle Database
Language	:	Java (Swing),SQL

CHAPTER 3

MODULE DESCRIPTION

The **Blood Donation Management System** consists of six primary modules that work together to streamline donor coordination, blood stock management, and emergency response. Each module handles specific functionalities and interacts

seamlessly with the others to ensure smooth and efficient system performance. The modules are described below:

1. USER AUTHENTICATION MODULE

This module manages secure registration, login, and logout for donors, hospitals, and administrators. It validates user credentials, maintains active sessions, and ensures that sensitive features such as blood requests and inventory updates are accessible only to authorized users.

2. DONOR MANAGEMENT MODULE

This module allows donors to register, update personal and medical information, check eligibility, and schedule donation appointments. It maintains donor history and ensures compliance with donation guidelines such as minimum interval between donations.

3. BLOOD INVENTORY MODULE

The inventory module tracks real-time availability of blood units by type and quantity. It updates stock levels automatically after donations or issue of units to hospitals, helping administrators monitor shortages and prevent mismatches during emergencies.

4. BLOOD REQUEST & MATCHING MODULE

This module enables hospitals or recipients to request specific blood types. It automatically searches the inventory and identifies matching donors when units

are not available. It ensures quick and accurate donor–recipient compatibility checks based on blood group and availability.

5. WAITLIST MANAGEMENT MODULE

When requested blood units are unavailable, this module manages a FIFO-based waitlist. It queues recipients in order of request time and sends automated notifications to donors or administrators when matching blood becomes available or when donors are needed urgently.

6. NOTIFICATION & ALERT MODULE

This module sends automated alerts through email or SMS for events such as donation reminders, emergency blood requirements, waitlist updates, appointment confirmations, and stock shortages. It ensures timely communication between donors, hospitals, and blood banks.

CHAPTER 4

SAMPLE CODING

```
import javax.swing.*;

import java.awt.*; import

java.sql.*;



public class SignUpForm extends JFrame {

    JTextField nameF, userF, emailF; JPasswordField passF, confirmF;
```

```
        public SignUpForm() {
setTitle("Sign Up"); setSize(380, 500);
setLocationRelativeTo(null); setDefaultCloseOperation(DISPOSE_ON_CLOSE);

            JPanel p = new JPanel(new GridLayout(0,1,8,8));

            nameF = new JTextField(); userF = new JTextField(); emailF = new JTextField();
            passF = new JPasswordField(); confirmF = new JPasswordField();

            p.add(new JLabel("Full Name"));   p.add(nameF); p.add(new
            JLabel("Username"));                p.add(userF); p.add(new
            JLabel("Email"));                   p.add(emailF); p.add(new
            JLabel("Password"));                p.add(passF); p.add(new
            JLabel("Confirm Password")); p.add(confirmF);

            JButton sign = new JButton("Sign Up");
            sign.addActionListener(e -> register());
            p.add(sign);

add(p); setVisible(true);
```

```
    }


    private void register() {
        try (Connection con = DBConnection.getConnection()) {
            PreparedStatement ps = con.prepareStatement(
                "INSERT INTO users(full_name,username,email,password)
VALUES(?,?,?,?)"
            );
            ps.setString(1, nameF.getText()); ps.setString(2,
            userF.getText()); ps.setString(3,
            emailF.getText());
            ps.setString(4, new String(passF.getPassword())); ps.executeUpdate();
            con.commit(); JOptionPane.showMessageDialog(this, "Account
            Created!"); dispose();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }
```

```
    public static void main(String[] a){ new SignUpForm(); }
}
```

Sample 1

Sample 1 shows the View Donor Form, which retrieves donor records from the database and displays them in a JTable using JDBC. It provides search filters for blood group and location with real-time results. The module loads all donors on startup and updates the total count dynamically.

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel; import
java.awt.*;
import java.sql.*;
public class ViewDonorsForm extends JFrame { private
DefaultTableModel model;
private JTable table;
private JTextField locationField;
private JComboBox<String> bgCombo; public
ViewDonorsForm() { setTitle("View Donors"); setSize(800,
550); setLocationRelativeTo(null);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
JPanel top = new JPanel(); bgCombo = new
JComboBox<>(new
String[]{"All","A+","A-","B+","B-","AB+","AB-","O+","O-"};
locationField = new JTextField(12);

      JButton searchBtn = new JButton("Search"); JButton
       refreshBtn = new JButton("Refresh"); top.add(new
       JLabel("Blood Group:")); top.add(bgCombo);
       top.add(new JLabel("Location:")); top.add(locationField);
       top.add(searchBtn); top.add(refreshBtn);

       String[] cols = {"ID","Name","Blood
Group","Age","Contact","Email","Address","Status"};
       model = new DefaultTableModel(cols, 0);
       table = new JTable(model);
       JScrollPane scroll = new JScrollPane(table);

       JLabel resultLabel = new JLabel("Total Donors: 0");

       JPanel bottom = new JPanel();
```

```
        JButton exportBtn = new JButton("Export CSV"); JButton
        closeBtn = new JButton("Close");
        bottom.add(exportBtn); bottom.add(closeBtn);

        add(top, BorderLayout.NORTH);
        add(scroll, BorderLayout.CENTER);
        add(resultLabel, BorderLayout.SOUTH);
        add(bottom, BorderLayout.SOUTH);
        searchBtn.addActionListener(e -> search(resultLabel));
        refreshBtn.addActionListener(e -> { bgCombo.setSelectedIndex(0);
locationField.setText(""); loadAll(resultLabel); });
        closeBtn.addActionListener(e -> dispose());
        exportBtn.addActionListener(e -> exportCSV());
        loadAll(resultLabel);
        setVisible(true);
    }
```

Sample 2

Sample 2 presents the Blood Inventory Management code, which efficiently loads, searches, filters, updates, and manages blood stock records using JDBC. It displays real-time inventory details in a Swing table, including units, expiry status, and locations.

```
import java.sql.*; import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class BloodInventory {
   Connection con; JTable table; DefaultTableModel model;

   public BloodInventory() { model = new DefaultTableModel(new
      String[]{"ID","Blood”}; table
      = new JTable(model); loadInventory();
      JFrame f = new JFrame("Blood Inventory");
      f.add(new JScrollPane(table)); f.setSize(400,300); }

   void loadInventory() { try
      {
con=DriverManager.getConnection("jdbc:mysql://localhost/bloodbank")
PreparedStatement ps = con.prepareStatement("SELECT * FROM
blood_inventory");

         ResultSet rs = ps.executeQuery(); while(rs.next())
         model.addRow(new Object[]{rs.getInt(1), rs.getString(2),
rs.getInt(3)});

      } catch(Exception e){ JOptionPane.showMessageDialog(null, e); } }
```

```
    public static void main(String[] args){ new BloodInventory(); }
}
```

Sample 3

Sample 3 shows the JDBC code used to connect the Java Swing application with the Oracle database, enabling secure retrieval and management of donor records. It uses prepared statements for safe querying, supports real-time search and filtering, and applies proper exception handling. The code ensures efficient data loading and updating, while maintaining accuracy and consistency.

```
import java.sql.*;

public class DBConnection {


    private static final String DRIVER =
"oracle.jdbc.driver.OracleDriver";

    private static final String URL =
"jdbc:oracle:thin:@localhost:1521/FREEPDB1"; private
    static final String USER = "system"; private static
    final String PASS = "avinash@2024";

    public static Connection getConnection() { try
        {
          Class.forName(DRIVER);
PASS);  Connection con = DriverManager.getConnection(URL, USER,
          con.setAutoCommit(false);        // manual transaction control
```

```
            return con;
        } catch (Exception e) {
            System.out.println("Database connection failed: " +
e.getMessage());
            return null;
        }
    }

    public static void close(Connection con) { try
        { if (con != null) con.close(); }
        catch (Exception e) { System.out.println("Close error: " + e.getMessage());
}
    }

    public static void main(String[] args) {
        Connection con = getConnection(); if
        (con != null) {
            System.out.println("Oracle DB Connected Successfully");
            close(con);
        }
    }
}
```

CHAPTER 5

SCREEN SHOTS

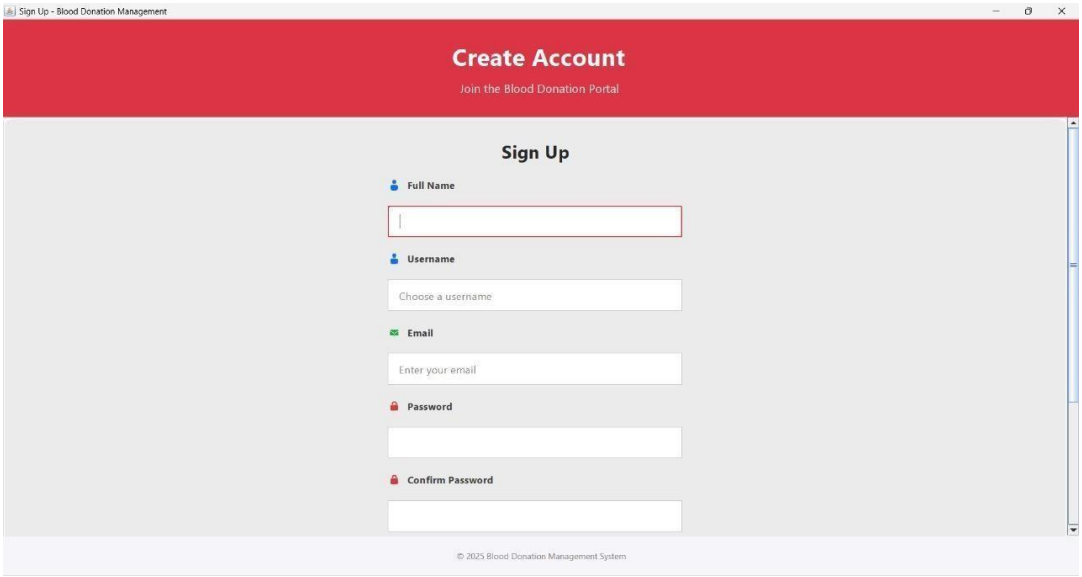

Fig 5.1 Sign Up

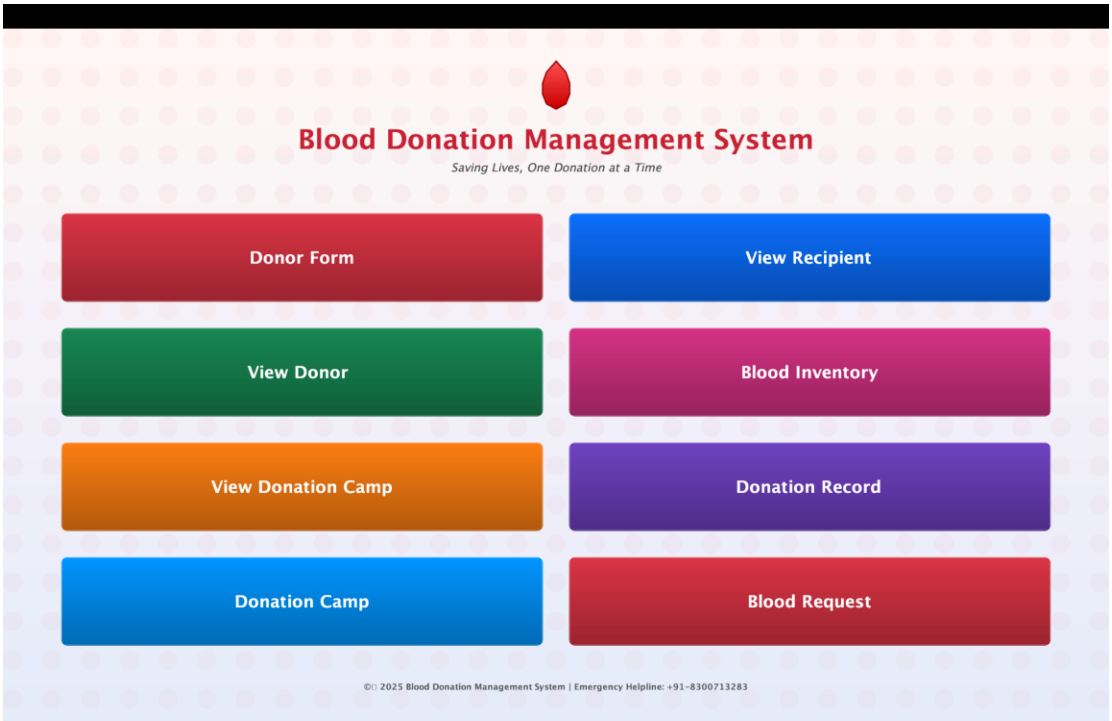

Fig 5.2 Main Menu

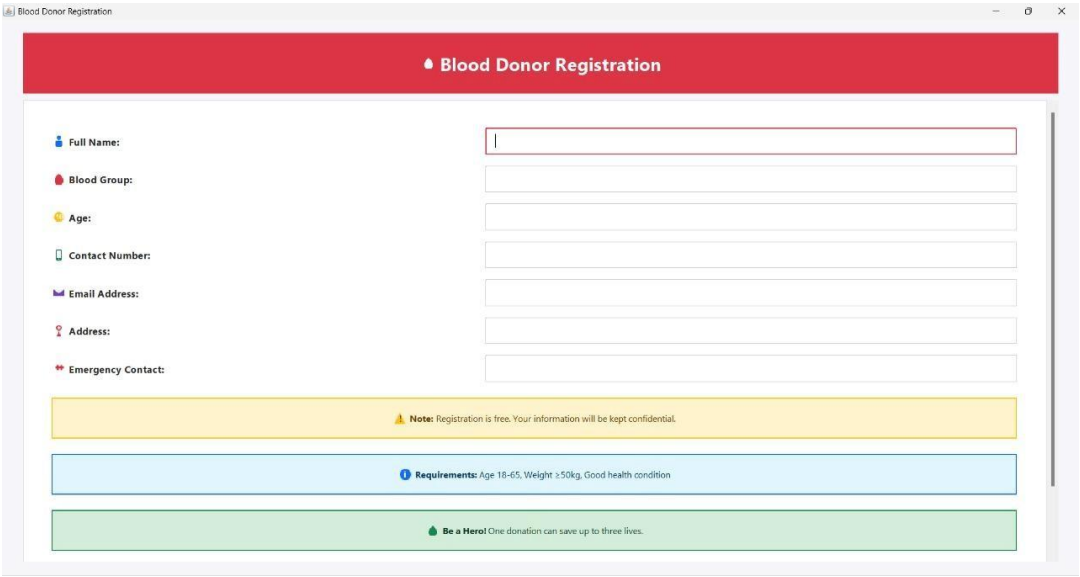

Fig 5.3 Donor Registeration

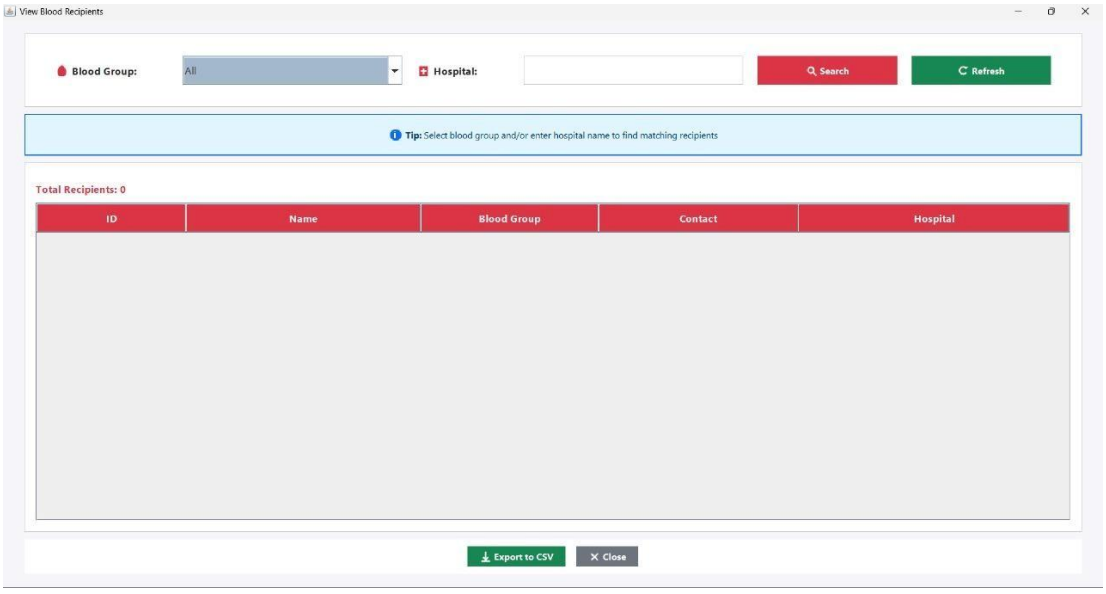

Fig 5.4 View Recepient

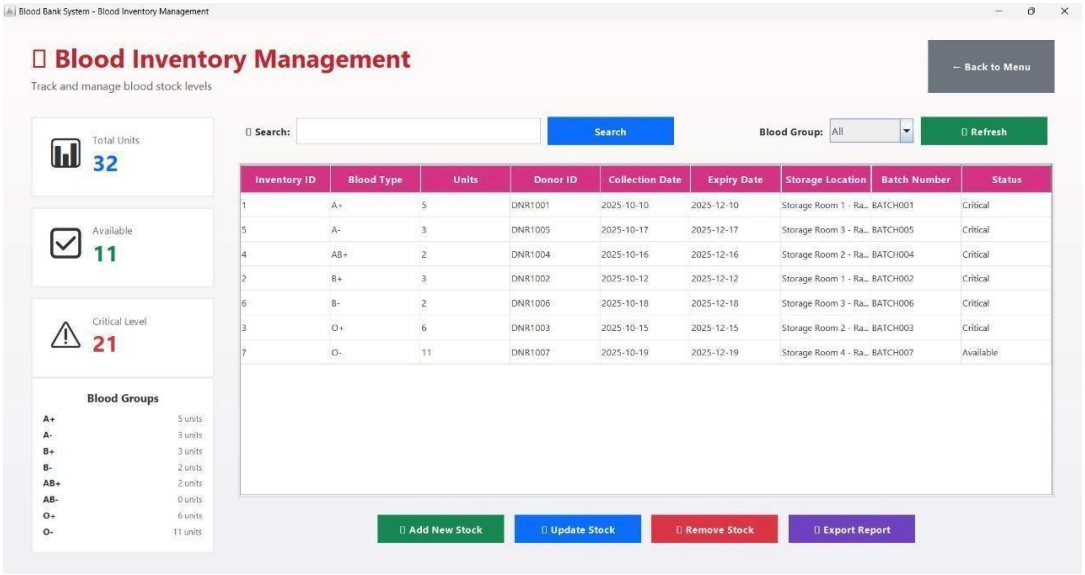

Fig 5.5 Blood Inventory

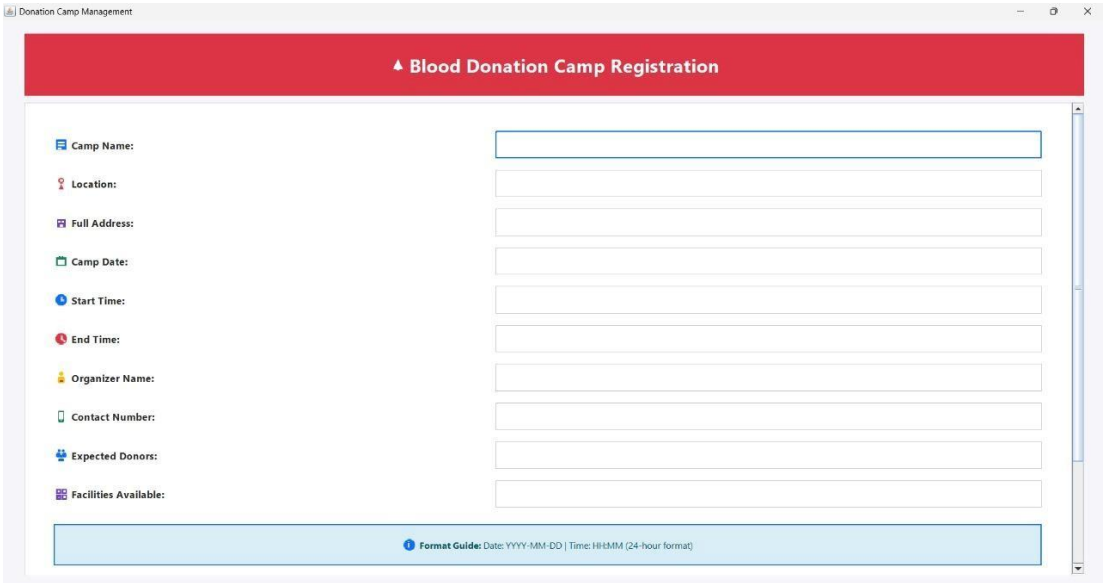

Fig 5.6 Donation Camp Register

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The **Blood Donation Management System** effectively addresses the challenges of traditional blood bank operations by introducing an automated, intelligent platform for managing donors, blood requests, and inventory. It streamlines donor registration, appointment scheduling, and blood stock monitoring while ensuring timely matching between donors and recipients through real-time updates. The integrated waitlist mechanism notifies users when required blood units become available, improving responsiveness during emergencies and reducing dependency on manual coordination. Developed using JSP, Java Servlets, Java Swing, and Oracle Database, the system provides a secure, scalable, and user-friendly solution for hospitals, donors, and administrators.In the future, the system can be enhanced with features such as real-time tracking of blood transport, mobile app integration, AI-based donor matching, automated eligibility checks, and advanced security for confidential medical data. These improvements will make the system more intelligent, efficient, and accessible, ensuring faster response during critical situations and enabling better resource management for blood banks and healthcare providers.

REFERENCES

1. https://www.w3schools.com/
2. https://tomcat.apache.org/tomcat-10.1-doc/
3. https://docs.oracle.com/en/database/
4. https://www.baeldung.com/
5. https://owasp.org/www-project-top-ten/