BLOOD BANK MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

JAMI SAI AKSHAYA **230701121**

DHARSHINI R S **230701076**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023 - 24

BONAFIDE CERTIFICATE

Certified that this project report "**BLOOD BANK MANAGEMENT SYSTEM**" is the bonafide work of **"JAMI SAI AKSHAYA(230701121), DHARSHINI R S (230701076)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs.Divya.M
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam,Chennai 602105

SIGNATURE

Mr. G Ragu
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam,Chennai 602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT:

The Blood Bank Management System is an advanced application built using Java Swing for the frontend and MySQL as the backend database, designed to streamline and automate blood donation and distribution processes. This system incorporates three distinct user roles: Admin, Donor, and Recipient, each serving a specific function to ensure efficient and secure management of blood resources.

The Admin role holds exclusive access to an administrative interface, where they oversee blood inventory, and review donation and blood request submissions. Donors are empowered to register on the platform and initiate requests to donate blood. These requests undergo Admin review, and upon approval, the corresponding blood units are added to the blood stock. Similarly, Recipients can create accounts to submit requests for specific blood types and quantities, detailing their requirements. The Admin evaluates these recipient requests, and upon approval, the requested units are deducted from the stock.

This system not only consolidates blood bank operations but also provides transparency, security, and accountability in the management of blood donations and distribution. The implementation of distinct user roles and permissions supports a structured workflow, reducing administrative burden and enhancing accessibility for all users, making this application a valuable asset for efficient blood bank management.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

○ INTRODUCTION

The Blood Bank Management System is an advanced application designed to simplify and optimize the intricate processes of blood donation, storage, and distribution. Leveraging the power of Java Swing for its graphical user interface and MySQL for its secure and reliable database management, this system ensures that critical blood bank operations are handled efficiently. It serves as a modern solution to address the challenges of manual management, which often results in delays and inaccuracies.

This system is structured around three distinct user roles: “Admin, Donor, and Recipient”. Each role is assigned specific permissions and functionalities to ensure accountability and efficiency. The “Admin” manages blood stocks, approves requests, and oversees the overall operations. “Donors” can register, provide their details, and request to donate blood, while “Recipients” can search for specific blood units and place requests as needed. This structured approach allows for clear role-based workflows, reducing the scope for errors and improving transparency.

A key feature of the system is its ability to provide real-time updates on blood stock availability. By keeping the inventory accurate and up-to-date, it ensures that emergency blood requests can be fulfilled promptly. This real-time tracking significantly improves the response time during medical emergencies, making the system a valuable tool in critical healthcare situations.

The system is user-friendly and scalable, catering to the needs of both small and large blood banks. Its intuitive interface encourages active participation from donors and recipients, fostering a sense of community involvement. The modular design also allows for future enhancements, such as integration with mobile applications or predictive analytics for blood demand trends.

By addressing the operational challenges faced by blood banks, the Blood Bank Management System plays a crucial role in supporting healthcare services. It provides a streamlined, accountable, and efficient platform for managing blood resources, ensuring timely and effective support for communities in need.

○ OBJECTIVES:

1.**Streamline Blood Management Processes**
- Automate the handling of blood donation, storage, and distribution to reduce manual efforts and errors.

2. **Enhance Accountability and Transparency**
- Implement role-based access for Admins, Donors, and Recipients to ensure secure and transparent operations.

3. **Ensure Real-Time Inventory Tracking**
- Maintain an up-to-date record of blood stocks to facilitate timely response to requests and emergencies.

4. **Support Healthcare Facilities**
- Provide a reliable platform for hospitals, blood banks, and healthcare organizations to manage blood resources efficiently.

5. **Encourage Donor Participation**
- Create a user-friendly interface to make donor registration and blood donation requests simple and accessible.

6. **Enable Future Scalability**
- Design the system to integrate with larger healthcare networks and accommodate additional features, such as analytics and automated alerts.

○ MODULES

1 Donor Registration Module

The Donor Registration Module is designed to capture and store detailed information about each donor. This module collects essential data such as the donor's name, contact information, blood type, medical history, and past donation records. It ensures that all donor information is accurately entered and securely stored in the database. This module includes an intuitive form that guides users through the registration process, making it easy for donors to input their details. Additionally, it verifies the eligibility of donors based on predefined criteria, ensuring that only eligible donors are registered.

2 Blood Type Search Functionality

The Blood Type Search Functionality is crucial for locating specific blood types within the inventory. This module allows administrators and users to search for available blood units based on blood type. It provides real-time updates on the

availability of different blood types, helping administrators manage stock efficiently and ensuring that the right type of blood is available when needed. This module includes functionalities for filtering search results by location, blood bank, and availability status, making it easier to quickly find the required blood type in emergency situations.

3 Donor Contact Module

The Donor Contact Module facilitates direct communication between the blood bank administrators and donors. It stores and manages donor contact information, enabling administrators to send notifications and reminders for upcoming donation events, blood shortages, or eligibility for the next donation. This module supports various communication channels such as email, SMS, and phone calls, ensuring that donors can be reached promptly. It also includes features for scheduling appointments and sending thank-you messages, fostering a positive relationship between the blood bank and donors.

4 Recipient Registration and Donor Matching Module

The Recipient Registration and Donor Matching Module is designed to register recipients who need blood transfusions and match them with suitable donors. This module collects detailed information about recipients, including their medical history, blood type, and specific blood needs. It then uses this information to find compatible donors from the database, ensuring a reliable match. This module also prioritizes matches based on urgency and criticality, ensuring that those in the most critical need are attended to first. By facilitating efficient and accurate matching, this module helps save lives and improve patient outcomes.

5 Admin Dashboard Module

The Admin Dashboard Module serves as a centralized hub for administrators to manage all aspects of the blood donation system. This module provides a comprehensive overview of donor registrations, blood inventory levels, recipient information, and ongoing communications. It includes tools for generating reports, monitoring system performance, and tracking key metrics such as donation rates and blood usage. The dashboard is designed with an intuitive interface, allowing administrators to easily navigate through different sections, access detailed records, and perform administrative tasks efficiently. It also features role-based access control to ensure that sensitive information is only accessible to authorized personnel.

6 Database Module

The Database Module is the backbone of the Blood Donation Bank Management System, responsible for storing and managing all data related to donors, recipients, blood inventory, and transactions. This module ensures data integrity, security, and efficient retrieval through structured queries and indexing. It uses SQL for database management, supporting complex queries and reporting needs.

Chapter 2 SURVEY OF TECHNOLOGIES

SOFTWARE DESCRIPTION

The Blood Bank Management System (BBMS) is a desktop application designed to efficiently manage blood donation processes, including donor registration, recipient requests, blood inventory, and transaction histories. The system employs Java Swing for its user-friendly graphical interface, a relational database for secure data storage, and SQL for efficient data management.

LANGUAGES AND TECHNOLOGIES USED

Java (Swing)

- **Role:** Java, along with its Swing library, is used for building the BBMS application, providing both backend logic and a graphical user interface (GUI).
- **Usage:**
  - Develops an intuitive desktop application interface for donors, recipients, and administrators.
  - Implements core functionalities such as donor registration, blood requests, and inventory management.
  - Integrates with the database through JDBC (Java Database Connectivity) to perform real-time data operations.
- **Advantages:**
  - **Platform Independence:** Java applications run on any device with a Java Virtual Machine (JVM).
  - **Rich GUI Components:** Swing offers advanced tools to create responsive and interactive user interfaces.
  - **Robust Performance:** Java ensures reliability and scalability for managing multiple transactions.

SQL

- **Role:** SQL (Structured Query Language) is used for managing and querying the relational database that stores all data for the system.
- **Usage:**
  - Handles CRUD (Create, Read, Update, Delete) operations for donor details, recipient requests, blood stock, and transaction histories.
  - Maintains relationships between tables to ensure data consistency.
- **Advantages:**
  - **Efficient Data Management:** Supports large-scale data processing and complex queries.
  - **Data Security:** Ensures the integrity of sensitive information like donor and recipient details.

JDBC (Java Database Connectivity)

- **Role:** JDBC acts as a bridge between the Java application and the relational database.
- **Usage:**
  - Establishes connections between the Java Swing application and the database.
  - Executes SQL queries directly from the Java application.
- **Advantages:**
  - **Seamless Integration:** Provides a standardized API for database communication.
  - **Flexibility:** Works with various RDBMS solutions, making the system adaptable.

KEY BENEFITS OF THE SYSTEM

User-Friendly Interface: The Swing GUI ensures smooth interaction for donors, recipients, and administrators.

- **Real-Time Data Management:** SQL and JDBC enable fast and efficient data retrieval and updates.
- **Reliability:** Java's robust framework ensures stability and scalability for growing system demands.

Chapter 3 REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1Functional Requirements

1. **User Authentication and Authorization**
   - **User Registration and Login:** Allow users to register, create accounts, and log in securely.
   - **Role-Based Access Control:** Assign specific permissions based on user roles (donor, recipient, admin).
2. **Donor Registration and Profile Management**
   - **Profile Creation and Update:** Enable donors to create and update their profiles with personal and medical information.
   - **Donation History:** Allow donors to view their past donation records.
3. **Blood Type Search Functionality**
   - **Search Capability:** Allow users to search for available blood units based on blood type.
   - **Real-Time Updates:** Provide up-to-date information on blood type availability in the inventory.
4. **Recipient Registration and Donor Matching**
   - **Recipient Data Collection:** Register recipients by collecting detailed information, including medical history and specific blood needs.
   - **Donor Matching:** Match recipients with suitable donors based on compatibility and urgency.
5. **Admin Dashboard Module**
   - **Comprehensive Overview:** Provide an overview of donor registrations, blood inventory levels, and recipient information.
   - **Management Tools:** Include tools for generating reports, monitoring system performance, and tracking key metrics.

3.1.2 Non-Functional Requirements

1. **Security**
   - **Data Encryption:** Ensure sensitive data is securely transmitted between the application and the database.
   - **Secure Authentication:** Implement secure user authentication mechanisms.
2. **Performance**
   - **Scalability:** Design the system to handle increasing numbers of users and transactions efficiently.
   - **Response Time:** Ensure prompt responses to user actions and queries.
3. **Reliability**
   - **Availability:** Guarantee high system availability with minimal downtime.
   - **Data Backup:** Regularly back up data to prevent loss and ensure data recovery.
4. **Usability**
   - **User-Friendly Interface:** Provide an intuitive and easy-to-navigate interface using Java Swing for all user roles.

- **Accessibility:** Ensure the application is accessible to users with basic computer skills.

5. **Maintainability**
   - **Modular Design:** Use a modular architecture to facilitate easy maintenance and future upgrades.
   - **Comprehensive Documentation:** Provide detailed documentation for users and developers.
6. **Interoperability**
   - **Database Integration:** Enable smooth interaction with the SQL database for data management.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

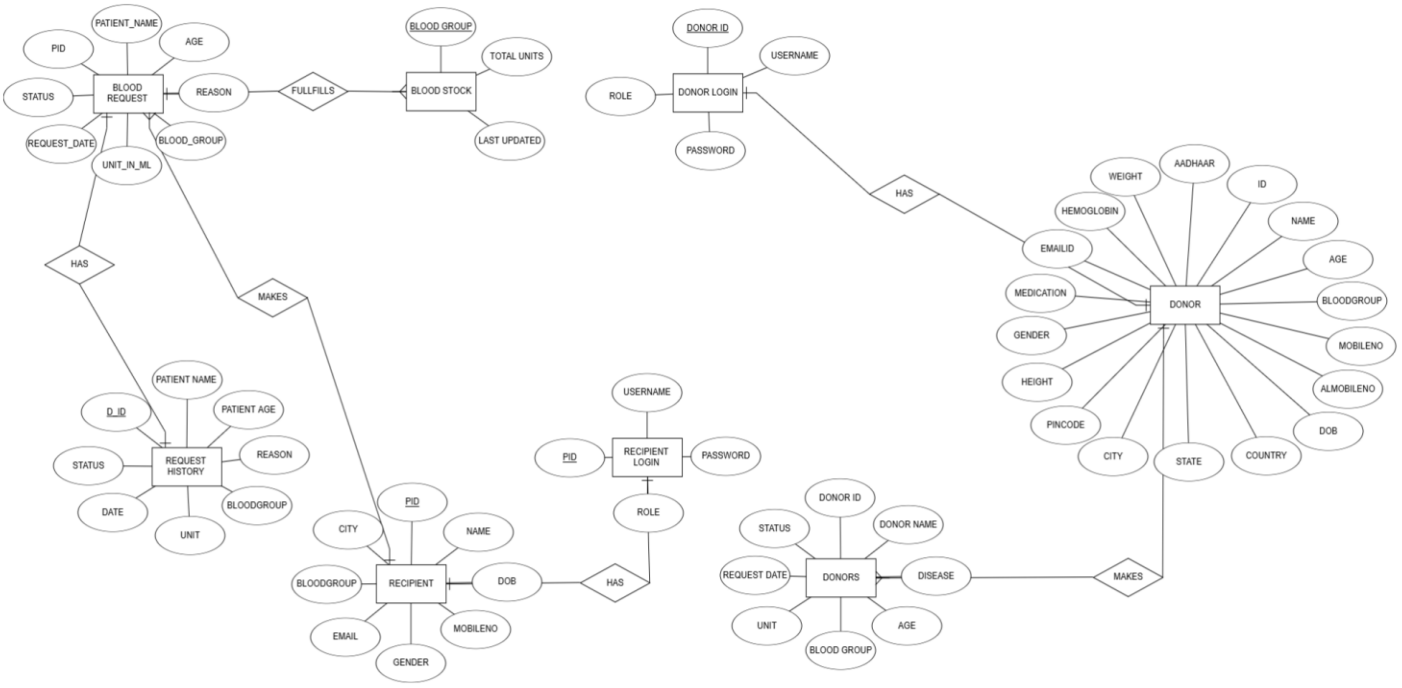
3.2.1 Hardware Requirements

- **Desktop PC or Laptop:** A reliable desktop PC or laptop to host the Blood Bank Management System.
- **Processor:** Intel® Core™ i3-6006U CPU @ 2.00GHz or equivalent for efficient processing.
- **RAM:** 4 GB or higher to handle concurrent database operations.
- **System Architecture:** 64-bit operating system for optimal performance.
- **Monitor Resolution:** 1024 x 768 or higher for clear display of the application interface.
- **Input Devices:** Keyboard and mouse for user interaction.

3.2.2 Software Requirements

- **Operating System:** Windows 10 or later.
- **IDE (Integrated Development Environment):** IntelliJ IDEA, Eclipse, or NetBeans for Java development.
- **Programming Language:** Java (Swing for GUI).
- **Database:** MySQL for relational database management.
- **Database Connectivity:** JDBC (Java Database Connectivity).
- **Version Control:** Git for version tracking and collaboration.
- **JDK (Java Development Kit):** Version 8 or later.

3.3 ER DIAGRAM



3.4 NORMALISATION

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The steps to normalize a database table for Blood Bank Management System are as follows.

Raw Database

Blood Requests Table

pid	patient_name	age	reason	blood_group	unit_in_ml	request_date	status
1	Alice	25	Surgery	A+	500	2024-11-01 10:00	Pending
2	Bob	40	Accident	B+	1000	2024-11-02 12:30	Approved

Blood Stock Table

blood_group	total_units	last_updated
A+	1500	2024-11-19 15:00
B+	1000	2024-11-19 15:00

Donor Login Table

donor_id	username	password	role
1	johndoe	1234pass	donor
2	janedoe	5678pass	donor

Donors Table

donor_id	donor_name	age	disease	blood_group	request_date	status
1	John Doe	30	None	A+	2024-11-18 09:00	Completed
2	Jane Doe	25	Diabetes	B+	2024-11-17 10:00	Pending

Login Table

pid	username	password	role
1	admin01	admin123	admin
2	alice_p	alicepass	patient

Request History Table

d_id	Patientname	Patientage	Reason	Bloodgroup	Unit	Date	Status
1	Alice	25	Surgery	A+	500	2024-11-01	Accepted
2	Bob	40	Accident	B+	1000	2024-11-02	Rejected

Recipient Table

pid	name	DOB	mobileno	gender	email	city
1	Alice	1999-01-01	9876543210	Female	alice@mail.com	New York
2	Bob	1984-12-12	8765432109	Male	bob@mail.com	Los Angeles

Second Normal Form (2NF)

To achieve **2NF**, we remove **partial dependencies** by breaking the tables into smaller tables such that each non-primary attribute depends on the **whole primary key**, not a part of it.

Updated Blood Requests Table

pid	patient_id	blood_group	unit_in_ml	request_date	status
1	1	A+	500	2024-11-01 10:00	Pending
2	2	B+	1000	2024-11-02 12:30	Approved

Patient Table

patient_id	patient_name	age	reason
1	Alice	25	Surgery
2	Bob	40	Accident

Blood Stock Table (No changes since it already adheres to 2NF)

blood_group	total_units	last_updated
A+	1500	2024-11-19 15:00
B+	1000	2024-11-19 15:00

Donor Login Table (No changes since it already adheres to 2NF)

donor_id	username	password	role
1	johndoe	1234pass	donor
2	janedoe	5678pass	donor

Donors Table

donor_id	donor_name	age	disease	blood_group	request_date	status
1	John Doe	30	None	A+	2024-11-18 09:00	Completed
2	Jane Doe	25	Diabetes	B+	2024-11-17 10:00	Pending

Login Table (No changes since it already adheres to 2NF)

pid	username	password	role
1	admin01	admin123	admin
2	alice_p	alicepass	patient

Request History Table

d_id	patient_id	Bloodgroup	Unit	Date	Status
1	1	A+	500	2024-11-01	Accepted
2	2	B+	1000	2024-11-02	Rejected

Recipient Table (No changes since it already adheres to 2NF)

pid	name	DOB	mobileno	gender	email	city
1	Alice	1999-01-01	9876543210	Female	alice@mail.com	New York
2	Bob	1984-12-12	8765432109	Male	bob@mail.com	Los Angeles

Third Normal Form (3NF)

To achieve **3NF**, we ensure there are no **transitive dependencies** (non-primary attributes depending on other non-primary attributes).

In this case, the database in **2NF** already adheres to **3NF**.

Chapter 4 SAMPLE PROGRAM CODE

DONOR LOGIN :

```
package bbms;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import pro.ConnectionProvider;
public class donor_login extends javax.swing.JFrame {
    public donor_login() {
        initComponents();
    }
    private void initComponents() {
        jPanel3 = new javax.swing.JPanel(); jPanel1 = new javax.swing.JPanel();
        jLabel7 = new javax.swing.JLabel(); jPanel2 = new javax.swing.JPanel();
        jLabel16 = new javax.swing.JLabel();jLabel26 = new javax.swing.JLabel();
        jLabel18 = new javax.swing.JLabel();jButton1 = new javax.swing.JButton();
        jLabel29 = new javax.swing.JLabel(); jLabel30 = new javax.swing.JLabel();
        username = new javax.swing.JTextField();
        password = new javax.swing.JPasswordField();jLabel20 = new javax.swing.JLabel();
        jLabel19 = new javax.swing.JLabel();jLabel2 = new javax.swing.JLabel();
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(25, 118, 242));
        setUndecorated(true);
        setResizable(false);
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
        jPanel3.setBackground(new java.awt.Color(255, 255, 255));
        jPanel3.setPreferredSize(new java.awt.Dimension(1440, 868));
        jPanel3.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
      jPanel1.setBackground(new java.awt.Color(255, 255, 255));
        jLabel7.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/Untitled
design.png")));
        jPanel2.setBackground(new java.awt.Color(250, 129, 135));
        jPanel2.setPreferredSize(new java.awt.Dimension(500, 371));
        jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
        jLabel16.setFont(new java.awt.Font("Segoe UI", 0, 13));
        jLabel16.setForeground(new java.awt.Color(255, 255, 255));
        jLabel16.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel16.setText("Hello! Let's get started");
        jPanel2.add(jLabel16, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, 440, -1));
        jLabel26.setFont(new java.awt.Font("Segoe UI", 0, 13));
        jLabel26.setForeground(new java.awt.Color(255, 255, 255));
        jLabel26.setText("Password");
        jPanel2.add(jLabel26, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 180, 586, -1));
```

```
jLabel18.setForeground(new java.awt.Color(255, 255, 255));
jLabel18.setText("_____");
jLabel18.setVerticalAlignment(javax.swing.SwingConstants.TOP);
jLabel18.setAlignmentY(0.0F);
jPanel2.add(jLabel18, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 210, 320, 30));
jButton1.setFont(new java.awt.Font("Segoe UI", 1, 14));
jButton1.setForeground(new java.awt.Color(255, 0, 0));
jButton1.setText("LOGIN");
jButton1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton1MouseClicked(evt);
    }
});
jPanel2.add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 300, 371, 39));
jLabel29.setFont(new java.awt.Font("Segoe UI", 0, 13));
jLabel29.setForeground(new java.awt.Color(255, 255, 255));
jLabel29.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel29.setText("Don't have an account? Sign Up");
jLabel29.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel29MouseClicked(evt);
    }
});
jPanel2.add(jLabel29, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 350, -1, -1));

jLabel30.setForeground(new java.awt.Color(255, 255, 255));
jLabel30.setText("_____");
jPanel2.add(jLabel30, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 144, 320, 30));
username.setBackground(new java.awt.Color(250, 129, 135));
username.setBorder(null);
username.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
jPanel2.add(username, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 140, 310, 30));
password.setBackground(new java.awt.Color(250, 129, 135));
password.setBorder(null);
jPanel2.add(password, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 200, 320, 30));
jLabel15.setFont(new java.awt.Font("Segoe UI", 0, 32));
jLabel15.setForeground(new java.awt.Color(255, 255, 255));
jLabel15.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel15.setText("Login");
jPanel2.add(jLabel15, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 30, 440, -1));
jLabel20.setFont(new java.awt.Font("Segoe UI", 0, 18));
jLabel20.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel20.setText("X");
jLabel20.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel20MouseClicked(evt); } });
```

```
        jPanel2.add(jLabel20, new org.netbeans.lib.awtextra.AbsoluteConstraints(394, 0, 46, -1));
        jLabel19.setFont(new java.awt.Font("Segoe UI", 0, 13));
        jLabel19.setForeground(new java.awt.Color(255, 255, 255));
        jLabel19.setText("Username");
        jPanel2.add(jLabel19, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 110, 361, -1));
        jPanel1.setLayout(new javax.swing.GroupLayout(jPanel1));
        jPanel3.add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(410, 340, -1, 400));
        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/login page
background.png")));
        jPanel3.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, -10, 1750, 1040));
        getContentPane().add(jPanel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1710,
1020));
        setSize(new java.awt.Dimension(1710, 1018));
        setLocationRelativeTo(null);}
    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
        String username1 = username.getText();
        String password1 = password.getText();
        try {
            Connection con = ConnectionProvider.getCon();
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT donor_id FROM donor_login WHERE username =
'" + username1 + "' AND password = '" + password1 + "'");
            if (rs.next()) {
                int d_id = rs.getInt("donor_id");
                dispose();
                new donorhomepage(d_id).setVisible(true);
            } else {
                JOptionPane.showMessageDialog(null, "Username or password is incorrect");
              username.setText(""); password.setText("");}
        } catch (Exception e) {JOptionPane.showMessageDialog(null, e);}  }
    private void jLabel20MouseClicked(java.awt.event.MouseEvent evt) {System.exit(0); }
    private void jLabel29MouseClicked(java.awt.event.MouseEvent evt) {
        new createprofile().setVisible(true);}
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(() -> new donor_login().setVisible(true));}
    private javax.swing.JButton jButton1;private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16; private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel29;private javax.swing.JLabel jLabel30;
    private javax.swing.JLabel jLabel7;private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2; private javax.swing.JPanel jPanel3;
    private javax.swing.JPasswordField password;private javax.swing.JTextField         username;
}
```

ADMIN HOME PAGE :

```
package bbms;
import java.sql.*;
import javax.swing.*;
import pro.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableModel;
import net.proteanit.sql.DbUtils;
public class blood_home extends javax.swing.JFrame {
    public blood_home()
    {
        initComponents();
        updateBloodStockLabels() ;
    }
    private void initComponents() {
        jPanel16 = new javax.swing.JPanel();jLabel31 = new javax.swing.JLabel();
        jLabel32 = new javax.swing.JLabel(); jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();jLabel1 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();aplus = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();jLabel2 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();bplus = new javax.swing.JLabel();
        jPanel4 = new javax.swing.JPanel();jLabel3 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();oplus = new javax.swing.JLabel();
        jPanel5 = new javax.swing.JPanel();abplus = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();jLabel12 = new javax.swing.JLabel();
       jPanel6 = new javax.swing.JPanel();jLabel5 = new javax.swing.JLabel();
        jLabel16 = new javax.swing.JLabel(); jLabel15 = new javax.swing.JLabel();
        aminus = new javax.swing.JLabel(); jPanel7 = new javax.swing.JPanel();
        jLabel6 = new javax.swing.JLabel();jLabel7 = new javax.swing.JLabel();
        bminus = new javax.swing.JLabel();jPanel8 = new javax.swing.JPanel();
      jLabel14 = new javax.swing.JLabel();ominus = new javax.swing.JLabel();
        jPanel9 = new javax.swing.JPanel();
        jLabel8 = new javax.swing.JLabel(); jLabel13 = new javax.swing.JLabel();
        abminus = new javax.swing.JLabel();
        jPanel10 = new javax.swing.JPanel();jPanel11 = new javax.swing.JPanel();
        jLabel18 = new javax.swing.JLabel();jLabel22 = new javax.swing.JLabel();
        jPanel12 = new javax.swing.JPanel();jLabel23 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();jPanel15 = new javax.swing.JPanel();
        jLabel24 = new javax.swing.JLabel(); jLabel26 = new javax.swing.JLabel();
        jPanel14 = new javax.swing.JPanel();jLabel20 = new javax.swing.JLabel();
        jLabel28 = new javax.swing.JLabel();jPanel17 = new javax.swing.JPanel();
        jLabel19 = new javax.swing.JLabel();jLabel29 = new javax.swing.JLabel();
        jPanel19 = new javax.swing.JPanel();jLabel25 = new javax.swing.JLabel();
        jLabel30 = new javax.swing.JLabel(); jPanel21 = new javax.swing.JPanel();
        jLabel21 = new javax.swing.JLabel(); jLabel33 = new javax.swing.JLabel();
```

```
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setUndecorated(true);
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel16.setBackground(new java.awt.Color(250, 129, 135));

        jLabel31.setFont(new java.awt.Font("Segoe UI Semibold", 1, 28)); // NOI18N
        jLabel31.setText("BLOOD BANK MANAGEMENT SYSTEM");

        jLabel32.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/photo 16.png"))); //
NOI18N
        jLabel32.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jLabel32MouseClicked(evt);
            }
        });

       javax.swing.GroupLayout jPanel16Layout = new javax.swing.GroupLayout(jPanel16);
jPanel16.setLayout(jPanel16Layout);
jPanel16Layout.setHorizontalGroup(
    jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel16Layout.createSequentialGroup()
        .addGap(277, 277, 277)
        .addComponent(jLabel31, javax.swing.GroupLayout.PREFERRED_SIZE, 727,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 888,
Short.MAX_VALUE)
        .addComponent(jLabel32)
        .addGap(22, 22, 22))
);
jPanel16Layout.setVerticalGroup(
    jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel16Layout.createSequentialGroup()

.addGroup(jPanel16Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jLabel32)
            .addComponent(jLabel31))
        .addGap(0, 9, Short.MAX_VALUE))
);
getContentPane().add(jPanel16, new org.netbeans.lib.awtextra.AbsoluteConstraints(-240, 0, 1950, -
1));

jPanel1.setPreferredSize(new java.awt.Dimension(1190, 200));
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jPanel2.setBackground(new java.awt.Color(255, 255, 255));
```

```
jPanel2.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel1.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel1.setText("A+");
jPanel2.add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 10, -1, -1));
jLabel9.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-
onlinepngtools.png")));
jPanel2.add(jLabel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 10, -1, 33));
aplus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
aplus.setText("0");
jPanel2.add(aplus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 30, -1, -1));

jPanel3.setBackground(new java.awt.Color(255, 255, 255));
jPanel3.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel3.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel2.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel2.setText("B+");
jPanel3.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 10, -1, -1));
jLabel10.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-
onlinepngtools.png")));
jPanel3.add(jLabel10, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 10, -1, 33));
bplus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
bplus.setText("0");
jPanel3.add(bplus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(340, 30, -1, -1));

jPanel4.setBackground(new java.awt.Color(255, 255, 255));
jPanel4.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel4.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel3.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel3.setText("O+");
jPanel4.add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 10, -1, -1));
jLabel11.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-
onlinepngtools.png")));
jPanel4.add(jLabel11, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 10, -1, 33));
oplus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
oplus.setText("0");
jPanel4.add(oplus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(620, 30, -1, -1));

jPanel5.setBackground(new java.awt.Color(255, 255, 255));
jPanel5.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel5.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel4.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel4.setText("AB+");
jPanel5.add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 0, -1, -1));
```

```
jLabel12.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-onlinepngtools.png")));
jPanel5.add(jLabel12, new org.netbeans.lib.awtextra.AbsoluteConstraints(180, 0, -1, 33));
abplus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
abplus.setText("0");
jPanel5.add(abplus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(920, 30, -1, -1));

jPanel6.setBackground(new java.awt.Color(255, 255, 255));
jPanel6.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel6.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel5.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel5.setText("A-");
jPanel6.add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(150, 10, -1, -1));
jLabel16.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-onlinepngtools.png")));
jPanel6.add(jLabel16, new org.netbeans.lib.awtextra.AbsoluteConstraints(180, 10, -1, 33));
aminus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
aminus.setText("0");
jPanel6.add(aminus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(37, 139, -1, -1));

jPanel7.setBackground(new java.awt.Color(255, 255, 255));
jPanel7.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel7.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel6.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel6.setText("B-");
jPanel7.add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 10, -1, -1));
jLabel15.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-onlinepngtools.png")));
jPanel7.add(jLabel15, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 10, -1, 33));
bminus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
bminus.setText("0");
jPanel7.add(bminus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(340, 140, -1, -1));

jPanel8.setBackground(new java.awt.Color(255, 255, 255));
jPanel8.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel8.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel7.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel7.setText("O-");
jPanel8.add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(150, 10, -1, -1));
jLabel14.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-onlinepngtools.png")));
jPanel8.add(jLabel14, new org.netbeans.lib.awtextra.AbsoluteConstraints(180, 10, -1, 33));
ominus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
ominus.setText("0");
```

```
jPanel8.add(ominus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel8, new org.netbeans.lib.awtextra.AbsoluteConstraints(620, 140, -1, -1));

jPanel9.setBackground(new java.awt.Color(255, 255, 255));
jPanel9.setPreferredSize(new java.awt.Dimension(200, 80));
jPanel9.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel8.setFont(new java.awt.Font("Maiandra GD", 1, 24));
jLabel8.setText("AB-");
jPanel9.add(jLabel8, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 0, -1, -1));
jLabel13.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/output-
onlinepngtools.png")));
jPanel9.add(jLabel13, new org.netbeans.lib.awtextra.AbsoluteConstraints(180, 0, -1, 33));
abminus.setFont(new java.awt.Font("Maiandra GD", 0, 24));
abminus.setText("0");
jPanel9.add(abminus, new org.netbeans.lib.awtextra.AbsoluteConstraints(6, 23, -1, 31));
jPanel1.add(jPanel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(920, 140, -1, -1));
  jLabel23.setBackground(new java.awt.Color(51, 51, 51));
jLabel23.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/house_28dp_FFFFFF_FILL0_wght400_GRA
D0_opsz24.png")));
jPanel12.add(jLabel23, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));

jLabel17.setBackground(new java.awt.Color(255, 255, 255));
jLabel17.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel17.setForeground(new java.awt.Color(255, 255, 255));
jLabel17.setText("Home");
jLabel17.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jLabel17MouseClicked(evt);
   }
});
jPanel12.add(jLabel17, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
jPanel10.add(jPanel12, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 170, 240, 50));

jPanel15.setBackground(new java.awt.Color(51, 51, 51));
jPanel15.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jPanel15MouseClicked(evt);
   }
});
jLabel24.setBackground(new java.awt.Color(255, 255, 255));
jLabel24.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel24.setForeground(new java.awt.Color(255, 255, 255));
jLabel24.setText("Recipients");
jLabel24.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jLabel24MouseClicked(evt);
```

```
    }
});
jPanel15.add(jLabel24, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
jLabel26.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/person_28dp_FFFFFF_FILL1_wght400_GRAD0_opsz24 (1).png")));
jPanel15.add(jLabel26, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));
jPanel10.add(jPanel15, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 270, 240, 50));

jPanel14.setBackground(new java.awt.Color(51, 51, 51));
jPanel14.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jPanel14MouseClicked(evt);
    }
});
jLabel20.setBackground(new java.awt.Color(255, 255, 255));
jLabel20.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel20.setForeground(new java.awt.Color(255, 255, 255));
jLabel20.setText("Donation Request");
jLabel20.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel20MouseClicked(evt);
    }
});
jPanel14.add(jLabel20, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
jLabel28.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/edit_square_28dp_FFFFFF_FILL0_wght400_GRAD0_opsz24.png")));
jPanel14.add(jLabel28, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));
jPanel10.add(jPanel14, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 320, 240, 50));

jPanel17.setBackground(new java.awt.Color(51, 51, 51));
jPanel17.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jPanel17MouseClicked(evt);
    }
});
jLabel19.setBackground(new java.awt.Color(255, 255, 255));
jLabel19.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel19.setForeground(new java.awt.Color(255, 255, 255));
jLabel19.setText("Donation History");
jLabel19.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel19MouseClicked(evt);
    }
});
jPanel17.add(jLabel19, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
```

```
jLabel29.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/manage_history_28dp_FFFFFF_FILL0_wght
400_GRAD0_opsz24.png")));
jPanel17.add(jLabel29, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));
jPanel10.add(jPanel17, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 370, 240, 50));

jPanel19.setBackground(new java.awt.Color(51, 51, 51));
jLabel25.setBackground(new java.awt.Color(255, 255, 255));
jLabel25.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel25.setForeground(new java.awt.Color(255, 255, 255));
jLabel25.setText("Blood Request");
jLabel25.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel25MouseClicked(evt);
    }
});
jPanel19.add(jLabel25, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
jLabel30.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/edit_square_28dp_FFFFFF_FILL0_wght400_
GRAD0_opsz24.png")));
jPanel19.add(jLabel30, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));
jPanel10.add(jPanel19, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 420, 240, 50));

jPanel21.setBackground(new java.awt.Color(51, 51, 51));
jLabel21.setBackground(new java.awt.Color(255, 255, 255));
jLabel21.setFont(new java.awt.Font("Segoe UI Semibold", 1, 18));
jLabel21.setForeground(new java.awt.Color(255, 255, 255));
jLabel21.setText("Request History");
jLabel21.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel21MouseClicked(evt);
    }
});
jPanel21.add(jLabel21, new org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 200, 50));
jLabel33.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icon/manage_history_28dp_FFFFFF_FILL0_wght
400_GRAD0_opsz24.png")));
jPanel21.add(jLabel33, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 10, -1, -1));
jPanel10.add(jPanel21, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 470, 240, 50));

getContentPane().add(jPanel10, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 240,
1020));
setBounds(0, 0, 1710, 1018);

 private void jLabel18MouseClicked(java.awt.event.MouseEvent evt) {
    new donorlist().setVisible(true);}
private void jPanel11MouseClicked(java.awt.event.MouseEvent evt) {
    new Recipient().setVisible(true);}
```

```
  private void jLabel17MouseClicked(java.awt.event.MouseEvent evt) {
     new blood_home().setVisible(true);}


  private void jLabel20MouseClicked(java.awt.event.MouseEvent evt) {
     new Donorrequest().setVisible(true);
  }private void jLabel20KeyPressed(java.awt.event.KeyEvent evt) {
  }
 private void jPanel14MouseClicked(java.awt.event.MouseEvent evt) {
     new donation().setVisible(true);}
  private void jLabel19MouseClicked(java.awt.event.MouseEvent evt) {
     new donation().setVisible(true);}
  private void jPanel17MouseClicked(java.awt.event.MouseEvent evt) {
     new Donorrequest().setVisible(true);}
  private void jLabel25MouseClicked(java.awt.event.MouseEvent evt) {
     new blood_request().setVisible(true);}
  private void jLabel21MouseClicked(java.awt.event.MouseEvent evt) {
     new request_history().setVisible(true);}
  private void jLabel24MouseClicked(java.awt.event.MouseEvent evt) {
     new Recipient().setVisible(true);}
 private void jLabel32MouseClicked(java.awt.event.MouseEvent evt) {
     int a = JOptionPane.showConfirmDialog(null, "Do you really want to Logout?", "Select",
  JOptionPane.YES_NO_OPTION);
     if (a == 0) {
        setVisible(false);
        new user_select().setVisible(true);
     }
  }


        }

    public static void main(String args[]) {
     try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
  javax.swing.UIManager.getInstalledLookAndFeels()) {
           if ("Nimbus".equals(info.getName())) {
              javax.swing.UIManager.setLookAndFeel(info.getClassName());
              break;
           }
        }
     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
  javax.swing.UnsupportedLookAndFeelException ex) {

  java.util.logging.Logger.getLogger(blood_home.class.getName()).log(java.util.logging.Level.SEVE
  RE, null, ex);
     }
```

```
    java.awt.EventQueue.invokeLater(() -> new blood_home().setVisible(true));
}

 private void updateBloodStockLabels() {
    try (Connection con = ConnectionProvider.getCon(); Statement stmt = con.createStatement()) {
        ResultSet rs = stmt.executeQuery("SELECT blood_group, total_units FROM blood_stock");
        while (rs.next()) {
            String bloodGroup = rs.getString("blood_group");
            int totalUnits = rs.getInt("total_units");
            switch (bloodGroup) {
                case "A+": aplus.setText("" + totalUnits); break;
                case "B+": bplus.setText("" + totalUnits); break;
                case "O+": oplus.setText("" + totalUnits); break;
                case "AB+": abplus.setText("" + totalUnits); break;
                case "A-": aminus.setText("" + totalUnits); break;
                case "B-": bminus.setText("" + totalUnits); break;
                case "O-": ominus.setText("" + totalUnits); break;
                case "AB-": abminus.setText("" + totalUnits); break;
            } }}
catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error updating blood stock labels: " + e.getMessage());
    }
}
 private javax.swing.JLabel abminus, abplus, aminus, aplus, bminus, bplus, jLabel1, jLabel2,
jLabel3, jLabel4, jLabel5, jLabel6, jLabel7, jLabel8, jLabel9, jLabel10, jLabel11, jLabel12,
jLabel13, jLabel14, jLabel15, jLabel16, jLabel17, jLabel18, jLabel19, jLabel20, jLabel21, jLabel22,
jLabel23, jLabel24, jLabel25, jLabel26, jLabel28, jLabel29, jLabel30, jLabel31, jLabel32, jLabel33,
ominus, oplus;
private javax.swing.JPanel jPanel1, jPanel2, jPanel3, jPanel4, jPanel5, jPanel6, jPanel7, jPanel8,
jPanel9, jPanel10, jPanel11, jPanel12, jPanel14, jPanel15, jPanel16, jPanel17, jPanel19, jPanel21; }
```

BLOOD REQUEST PAGE :

```
package bbms;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellRenderer;
import java.awt.*;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.table.JTableHeader;
import pro.ConnectionProvider;
import net.proteanit.sql.DbUtils;
```

```
public class blood_request extends javax.swing.JFrame {
    private JButton acceptButton;
    private JButton deleteButton;

    public blood_request() {
        initComponents();
        JTableHeader header = jTable1.getTableHeader();
        header.setBackground(new Color(25, 118, 255));
        header.setForeground(Color.black);
        header.setOpaque(true);
        header.setFont(new Font("Arial", Font.BOLD, 14));
        loadData();
    }
    private void initComponents() {
        jPanel3 = new javax.swing.JPanel();
        jLabel27 = new javax.swing.JLabel();
        jLabel32 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jTextField2 = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setUndecorated(true);
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
        jPanel3.setBackground(new java.awt.Color(250, 129, 135));
        jLabel27.setFont(new java.awt.Font("Segoe UI Semibold", 1, 28));
        jLabel27.setText("BLOOD BANK MANAGEMENT SYSTEM");
        jLabel32.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/photo 16.png")));
        jLabel32.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jLabel32MouseClicked(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(277, 277, 277)
                .addComponent(jLabel27)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 1141,
Short.MAX_VALUE)
                .addComponent(jLabel32)
                .addContainerGap())
```

```
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel32)
                    .addComponent(jLabel27))
                .addGap(0, 9, Short.MAX_VALUE))
        );

        getContentPane().add(jPanel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(-240, 0,
1950, -1));

        jPanel1.setBackground(new java.awt.Color(255, 255, 255));
        jPanel1.setPreferredSize(new java.awt.Dimension(1190, 800));
        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null}
            },
            new String [] {
                "Patient name", "age", "disease", "blood_group", "units", "Status", "Actions"
            }
        ));
        jTable1.setFocusTraversalPolicyProvider(true);
        jTable1.setGridColor(new java.awt.Color(250, 129, 135));
        jTable1.setPreferredSize(new java.awt.Dimension(475, 810));
        jTable1.setRowHeight(35);
        jTable1.setSelectionBackground(new java.awt.Color(51, 153, 255));
        jTable1.setShowHorizontalLines(true);
        jTable1.getTableHeader().setResizingAllowed(false);
        jTable1.getTableHeader().setReorderingAllowed(false);
        jScrollPane1.setViewportView(jTable1);

        jPanel1.add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 130, 1420,
990));

        jTextField2.setForeground(new java.awt.Color(153, 153, 153));
```

```
        jTextField2.setText("Search.....");
        jTextField2.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        jTextField2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField2ActionPerformed(evt);
            }
        });
        jTextField2.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                jTextField2KeyReleased(evt);
            }
        });
        jPanel1.add(jTextField2, new org.netbeans.lib.awtextra.AbsoluteConstraints(1240, 60, 140,
40));

        jLabel5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icon/search.png")));
        jLabel5.setPreferredSize(new java.awt.Dimension(45, 45));
        jPanel1.add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(1390, 60, -1, 40));

        jLabel2.setFont(new java.awt.Font("Segoe UI", 3, 24));
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("BLOOD REQUEST ");
        jPanel1.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 70, 1470, 40));

        getContentPane().add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(240, 0, 1470,
1020));
    }

    private void loadData() {
      DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.setRowCount(0); // Clear existing rows

String query = "SELECT  patient_name, age, reason, blood_group, unit_in_ml, status FROM
blood_requests WHERE status = 'Pending'";
    try (Connection con = ConnectionProvider.getCon(); Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(query)) {
        while (rs.next()) {

            String patientName = rs.getString("patient_name");
            int age = rs.getInt("age");
            String reason = rs.getString("reason");
            String bloodGroup = rs.getString("blood_group");
            int unitInMl = rs.getInt("unit_in_ml");

            String status = rs.getString("status");

            // Add row to the model
```

```
            model.addRow(new Object[]{ patientName, age, reason, bloodGroup, unitInMl, status});
    }
    private void jTextField2KeyReleased(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:

        String search=jTextField2.getText();
        try{
            Connection con=ConnectionProvider.getCon();
            Statement st=con.createStatement();
            ResultSet rs =st.executeQuery("select * from blood_requests where  pid like '%"+search+"%'
or patient_name like '%"+search+"%'or age like '%"+search+"%'or reason like '%"+search+"%'or
blood_group like '%"+search+"%'or unit_in_ml like '%"+search+"%'or request_date like
'%"+search+"%'or  status like '%"+search+"%'");
            //jTable1.setAutoResizeMode(jTable1.AUTO_RESIZE_OFF);
            jTable1.setModel(DbUtils.resultSetToTableModel(rs));
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(null, e);
        }
    }

    private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
    private void jLabel18MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new donorlist().setVisible(true);
    }
   private void jPanel11MouseClicked(java.awt.event.MouseEvent evt) {
      new Recipient().setVisible(true);
    }

    private void jPanel11KeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
    }

    private void jLabel17MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new blood_home().setVisible(true);
    }

    private void jPanel12MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void jLabel24MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
```

```
        new Recipient().setVisible(true);
    }

    private void jPanel15MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void jPanel15KeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        ;
    }

    private void jLabel20MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new Donorrequest().setVisible(true);

    }

    private void jLabel19MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new donation().setVisible(true);

    }

    private void jLabel19KeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
    }

    private void jPanel17MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new Donorrequest().setVisible(true);
    }

    private void jLabel25MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new blood_request().setVisible(true);
    }

    private void jLabel21MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new request_history().setVisible(true);
    }

    private void jLabel32MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        int a=JOptionPane.showConfirmDialog(null,"Do you really want to
Logout?","Select",JOptionPane.YES_NO_OPTION);
```

```
        if(a==0){
            setVisible(false);
            new user_select().setVisible(true);
           // new login().setVisible(true);
        }
    }
  public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(blood_request.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(blood_request.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(blood_request.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(blood_request.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        }
             java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new blood_request().setVisible(true);
            }
        });

    }
private void loadData() {
    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    model.setRowCount(0);
    String query = "SELECT patient_name, age, reason, blood_group, unit_in_ml, status FROM
 blood_requests WHERE status = 'Pending'";
    try (Connection con = ConnectionProvider.getCon(); Statement stmt = con.createStatement();
 ResultSet rs = stmt.executeQuery(query)) {
        while (rs.next()) {
            String patientName = rs.getString("patient_name");
```

```
            int age = rs.getInt("age");
            String reason = rs.getString("reason");
            String bloodGroup = rs.getString("blood_group");
            int unitInMl = rs.getInt("unit_in_ml");
            String status = rs.getString("status");
            model.addRow(new Object[]{patientName, age, reason, bloodGroup, unitInMl, status});
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    addButtonToTable(jTable1);
}

public void addButtonToTable(JTable table) {
    if (table.getColumn("Actions") != null) {
        table.getColumn("Actions").setCellRenderer(new ButtonRenderer());
        table.getColumn("Actions").setCellEditor(new ButtonEditor(new JCheckBox(), table));
    } else {
        System.out.println("Actions column not found!");}}
class ButtonRenderer extends JPanel implements TableCellRenderer {
    private final JButton acceptButton = new JButton("Accept");
    private final JButton deleteButton = new JButton("Reject");

    public ButtonRenderer() {
        setLayout(new FlowLayout());
        add(acceptButton);
        add(deleteButton);
        acceptButton.setBackground(new Color(25, 118, 255));
        acceptButton.setForeground(Color.WHITE);
        acceptButton.setOpaque(true);
        acceptButton.setBorderPainted(false);
        deleteButton.setBackground(Color.RED);
        deleteButton.setForeground(Color.WHITE);
        deleteButton.setOpaque(true);
        deleteButton.setBorderPainted(false);
    }

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus, int row, int column) {
        return this;
    }
}

class ButtonEditor extends DefaultCellEditor {
    private JPanel panel;
    private JTable table;
```

```
    public ButtonEditor(JCheckBox checkBox, JTable table) {
        super(checkBox);
        this.table = table;
        panel = new JPanel();
        panel.setLayout(new FlowLayout());
        acceptButton = new JButton("Accept");
        deleteButton = new JButton("Reject");
        acceptButton.setBackground(new Color(25, 118, 255));
        acceptButton.setForeground(Color.WHITE);
        acceptButton.setOpaque(true);
        acceptButton.setBorderPainted(false);
        deleteButton.setBackground(new Color(250, 129, 135));
        deleteButton.setForeground(Color.WHITE);
        deleteButton.setOpaque(true);
        deleteButton.setBorderPainted(false);
        panel.add(acceptButton);
        panel.add(deleteButton);
        acceptButton.addActionListener(e -> {
            fireEditingStopped();
            acceptRequest();
        });
        deleteButton.addActionListener(e -> {
            fireEditingStopped();
            deleteRequest();
        });
    }

    @Override
    public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected,
int row, int column) {
        return panel;
    }

    @Override
    public Object getCellEditorValue() {
        return null;
    }}
private void acceptRequest() {
    int currentRow = jTable1.getSelectedRow();
    String name = (String) jTable1.getValueAt(currentRow, 0);
    String bloodGroup = (String) jTable1.getValueAt(currentRow, 3);
    int age = (int) jTable1.getValueAt(currentRow, 1);
    int requestedUnits = (int) jTable1.getValueAt(currentRow, 4);
    try (Connection con = ConnectionProvider.getCon(); Statement stmt = con.createStatement()) {
        String query = "SELECT total_units FROM blood_stock WHERE blood_group = '" +
bloodGroup + "'";
```

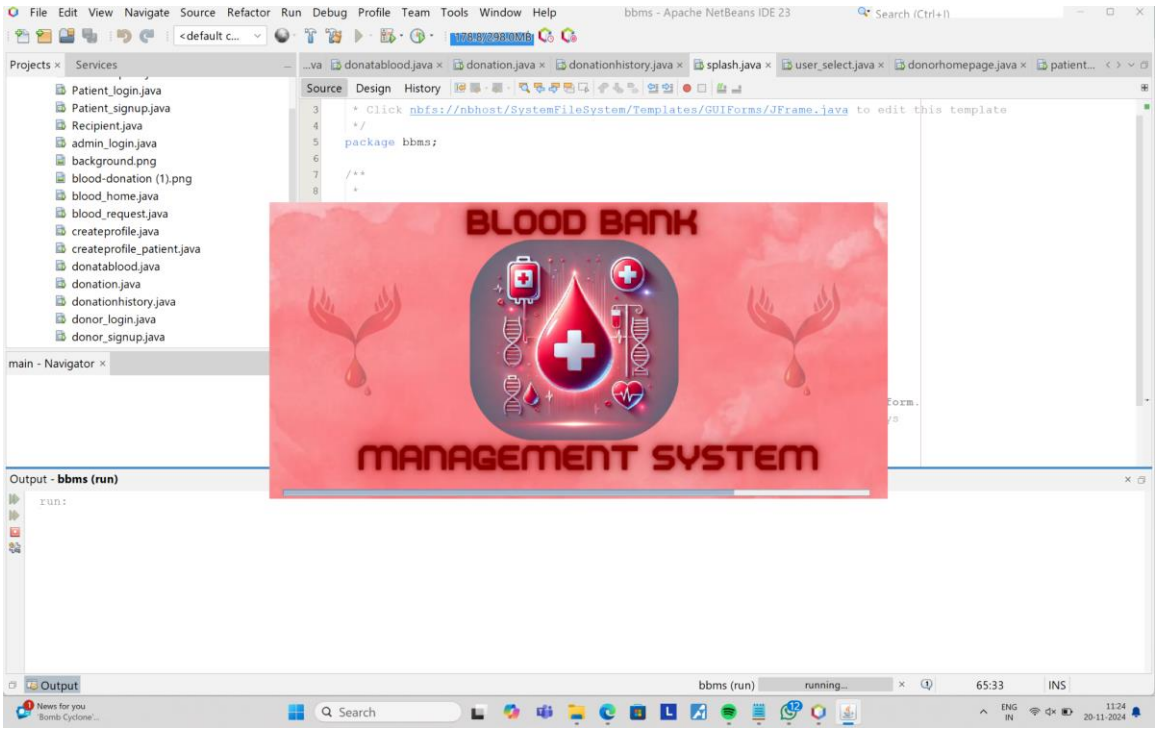
```
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next()) {
            int availableUnits = rs.getInt("total_units");
            if (requestedUnits > availableUnits) {
                JOptionPane.showMessageDialog(null, "Insufficient units available in the blood bank for
blood group " + bloodGroup + ".");
                return;
            }
        } else {
            JOptionPane.showMessageDialog(null, "Blood group " + bloodGroup + " not available in
stock.");
            return;
        }
        String updateQuery = "UPDATE blood_requests SET status = 'Approved' WHERE
blood_group = '" + bloodGroup + "' AND patient_name = '" + name + "' AND age = '" + age + "'
AND unit_in_ml = '" + requestedUnits + "'";
        stmt.executeUpdate(updateQuery);
        String stockQuery = "UPDATE blood_stock SET total_units = total_units - " + requestedUnits
+ ", last_updated = CURRENT_TIMESTAMP WHERE blood_group = '" + bloodGroup + "'";
        stmt.executeUpdate(stockQuery);
        JOptionPane.showMessageDialog(null, "Request approved and deducted from blood stock!");
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.removeRow(currentRow);
        model.fireTableDataChanged();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error approving request: " + e.getMessage());
        e.printStackTrace();
    }
}

private void deleteRequest() {
    int currentRow = jTable1.getSelectedRow();
    String name = (String) jTable1.getValueAt(currentRow, 0);
    String bloodGroup = (String) jTable1.getValueAt(currentRow, 3);
    int age = (int) jTable1.getValueAt(currentRow, 1);
    int requestedUnits = (int) jTable1.getValueAt(currentRow, 4);
    try (Connection con = ConnectionProvider.getCon(); Statement stmt = con.createStatement()) {
        String query = "UPDATE blood_requests SET status = 'Rejected' WHERE blood_group = '" +
bloodGroup + "' AND patient_name = '" + name + "' AND age = '" + age + "' AND unit_in_ml = '" +
requestedUnits + "'";
        stmt.executeUpdate(query);
        JOptionPane.showMessageDialog(null, "Request rejected!");
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.removeRow(currentRow);
        model.fireTableDataChanged();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error deleting request: " + e.getMessage());
        e.printStackTrace();
```

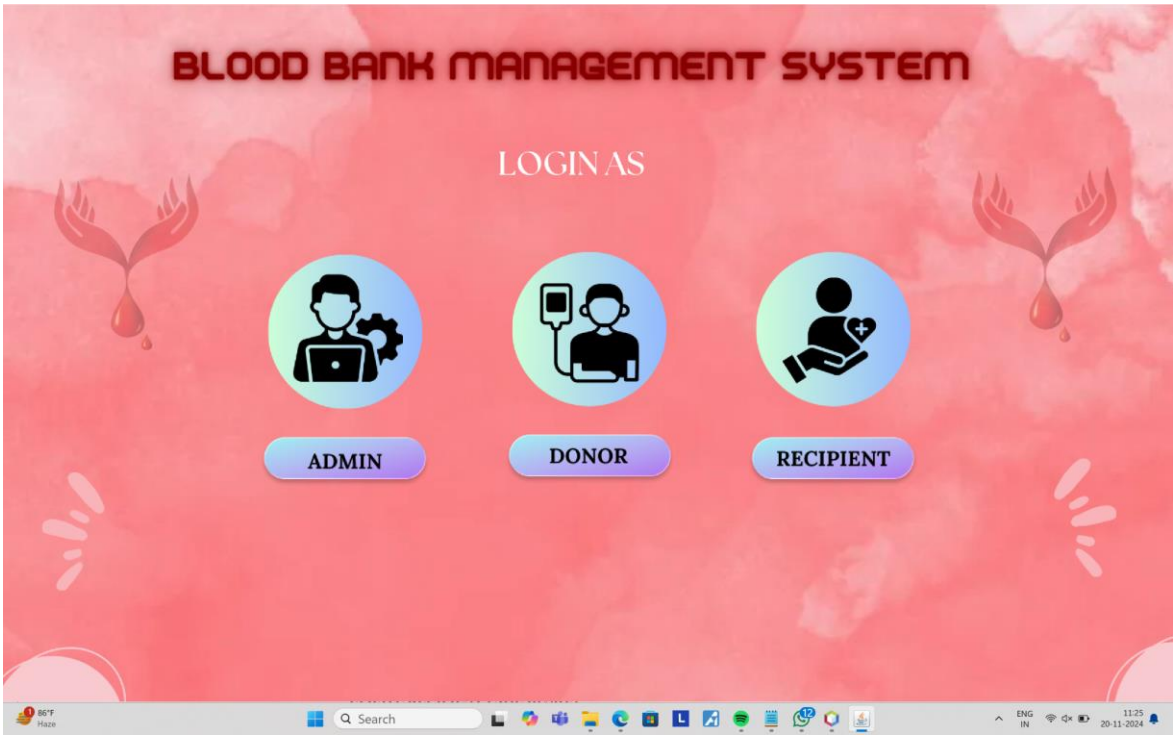
```
    }
}
private javax.swing.JLabel jLabel17, jLabel18, jLabel19, jLabel2, jLabel20, jLabel21, jLabel22,
jLabel23, jLabel24, jLabel25, jLabel27, jLabel29, jLabel30, jLabel31, jLabel32, jLabel33, jLabel34,
jLabel5;
private javax.swing.JPanel jPanel1, jPanel10, jPanel11, jPanel12, jPanel14, jPanel15, jPanel17,
jPanel19, jPanel21, jPanel3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField2;
```

Chapter 5 RESULTS AND DISCUSSION

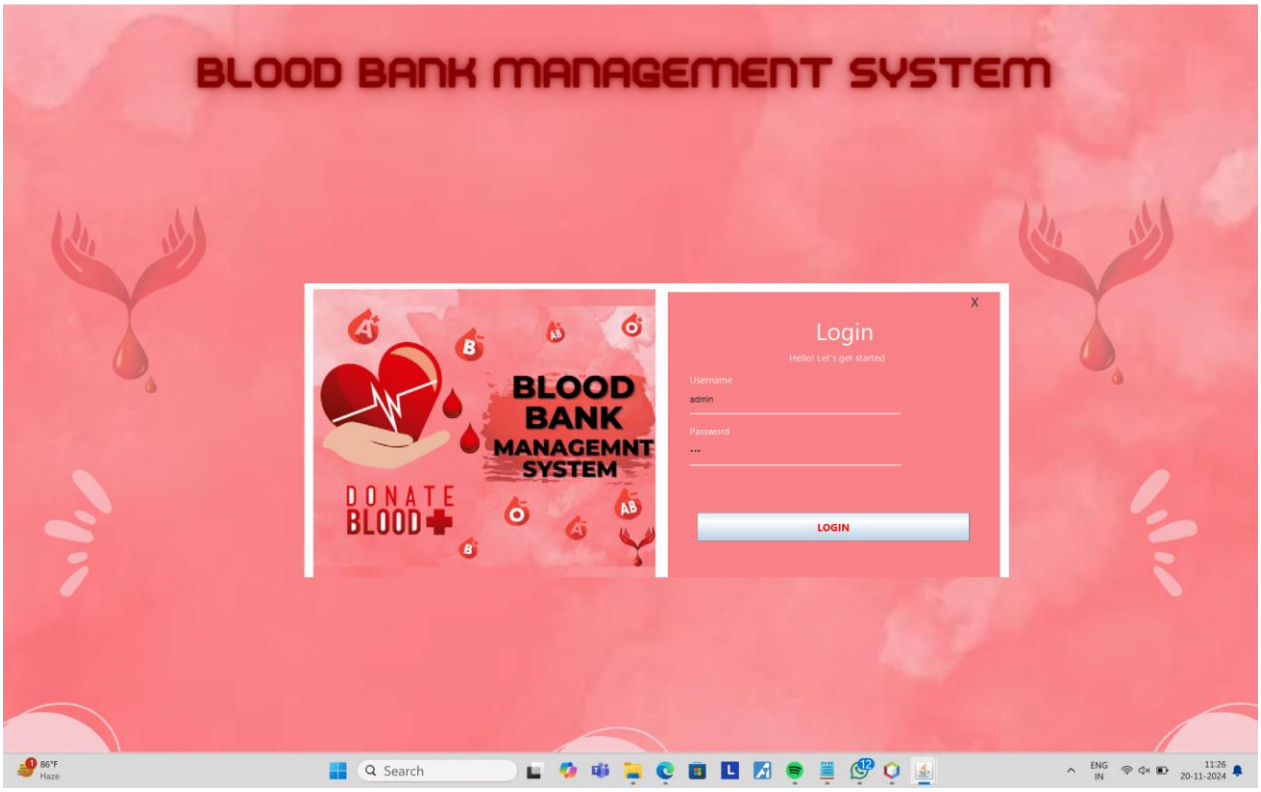
SPLASH PAGE DESIGN:



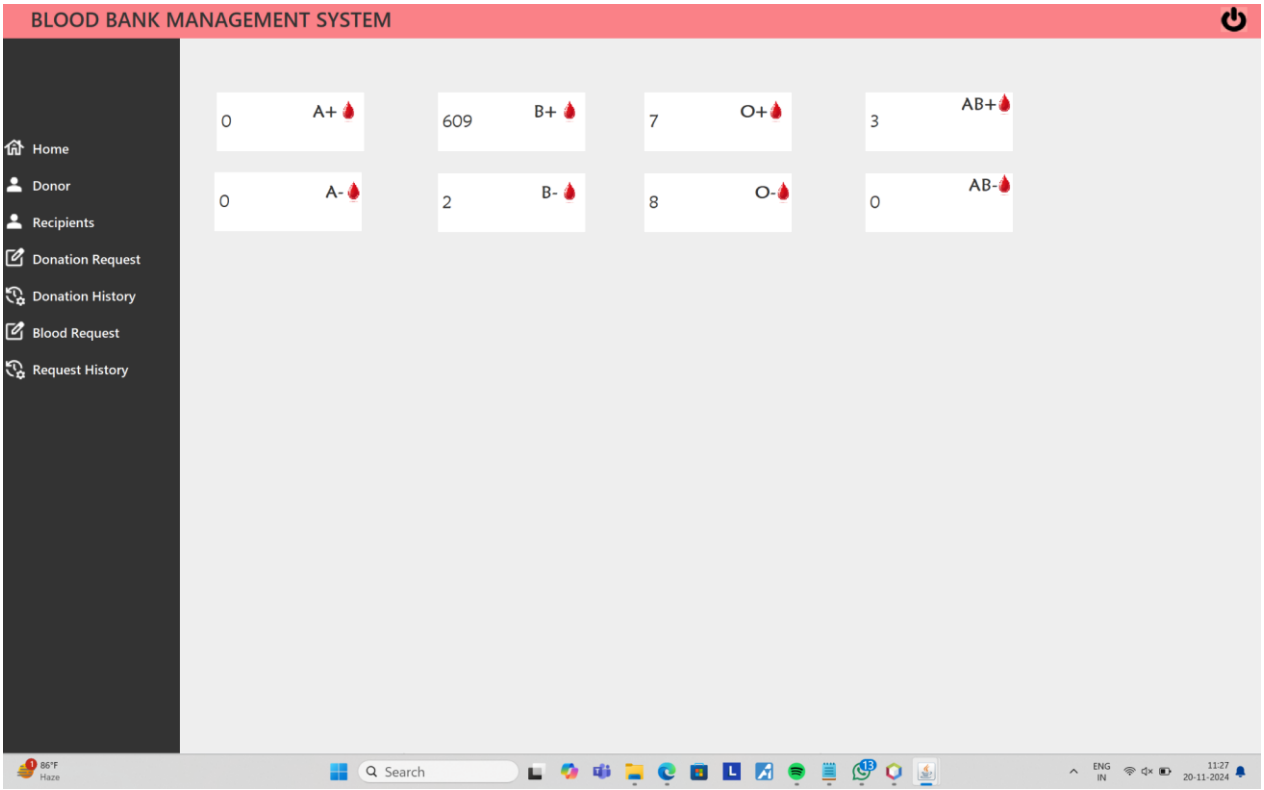
USER SELECT PAGE DESIGN:



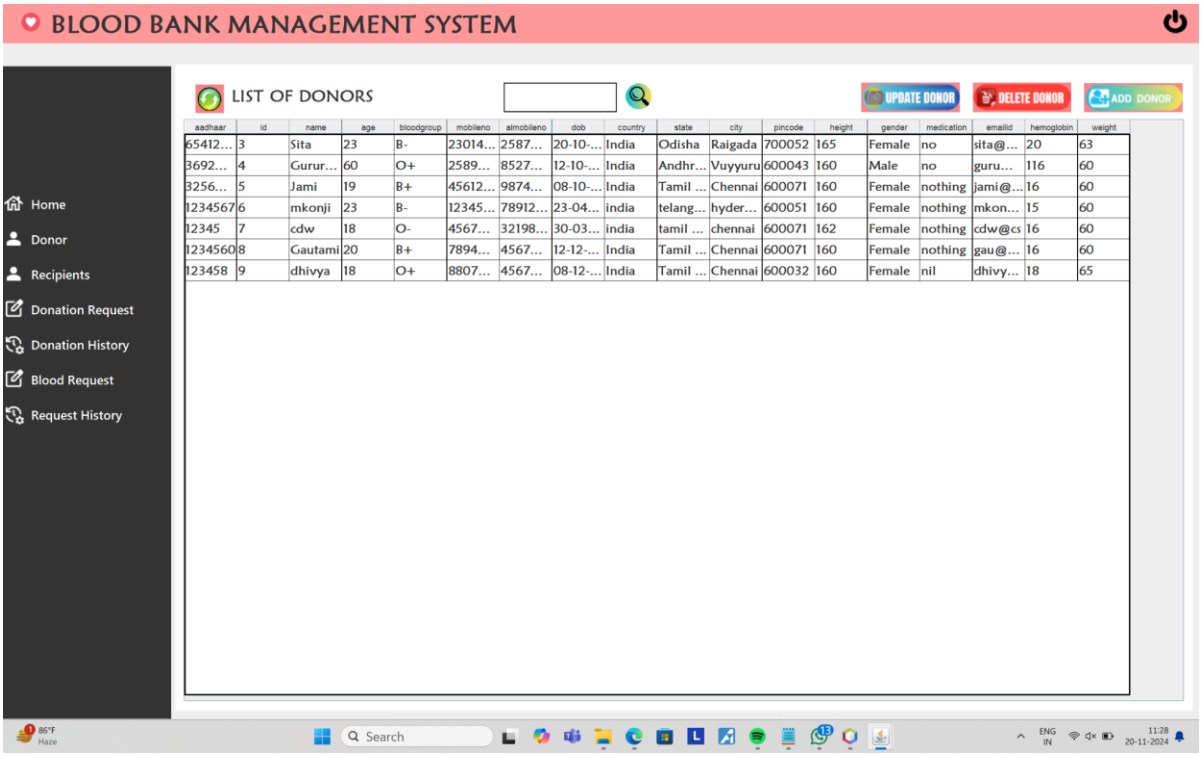
ADMIN LOGIN PAGE DESIGN:



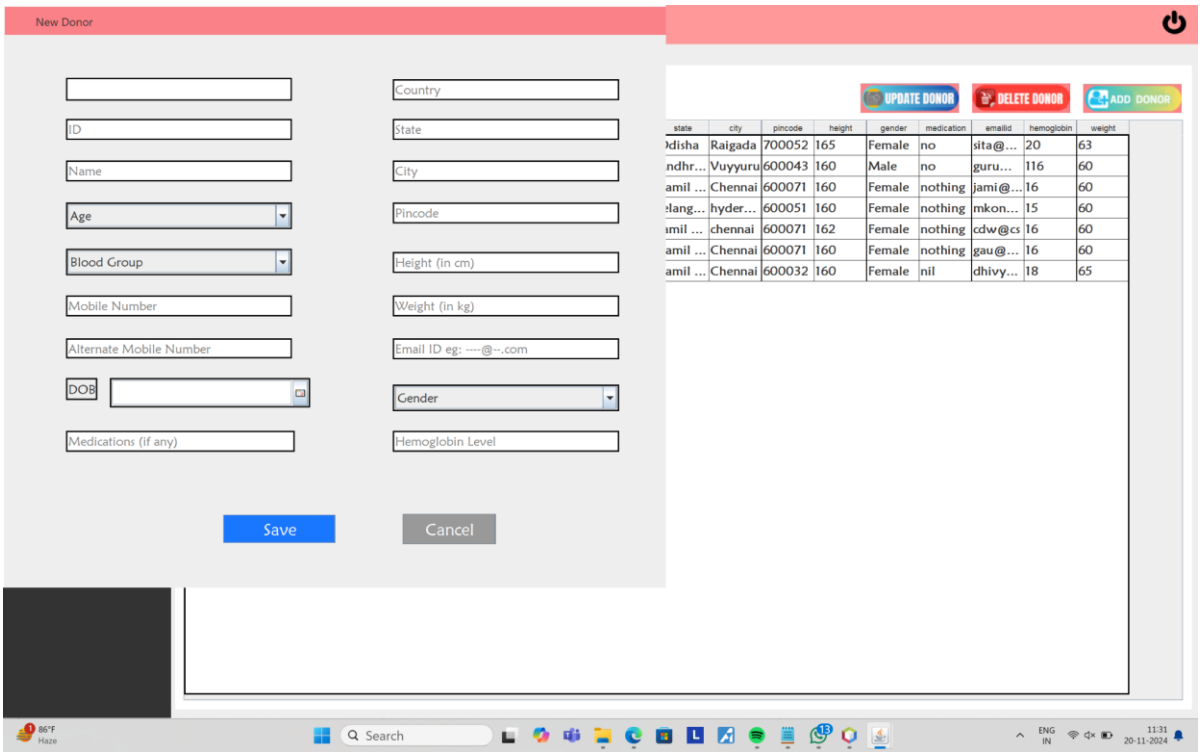
ADMIN HOME PAGE DESIGN:



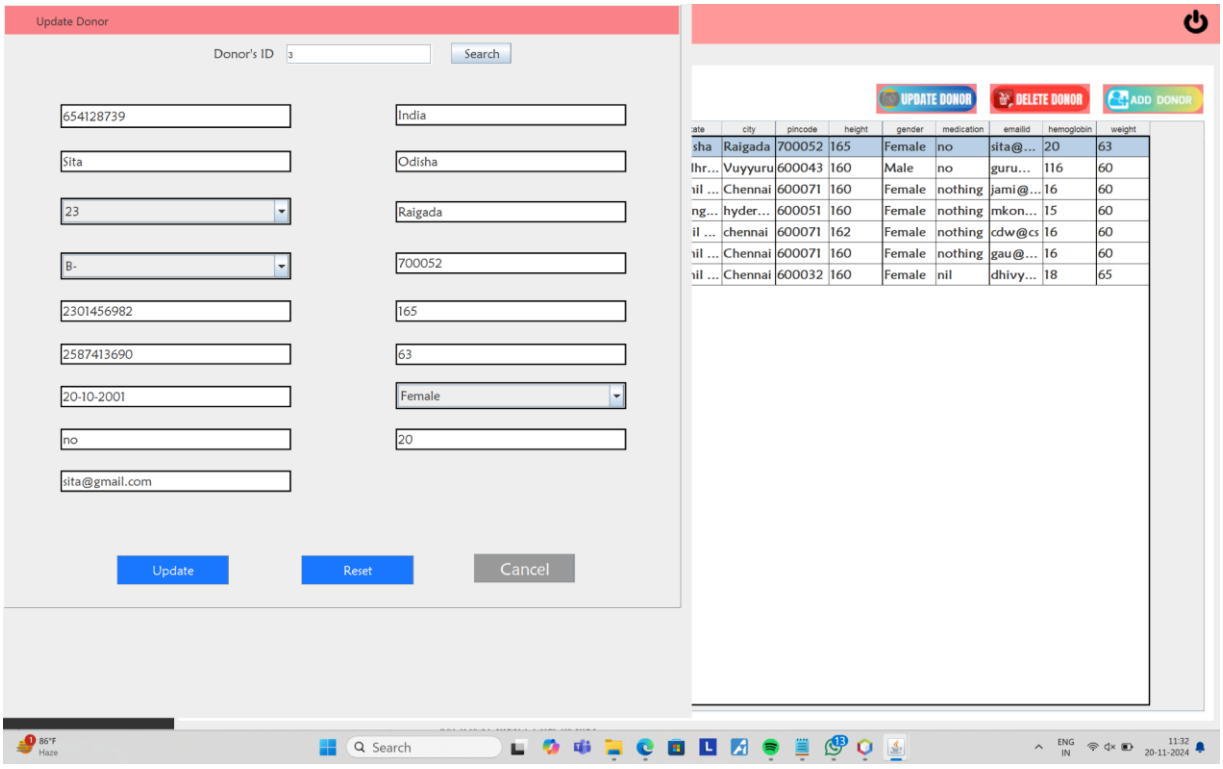
DONOR LIST PAGE DESIGN:



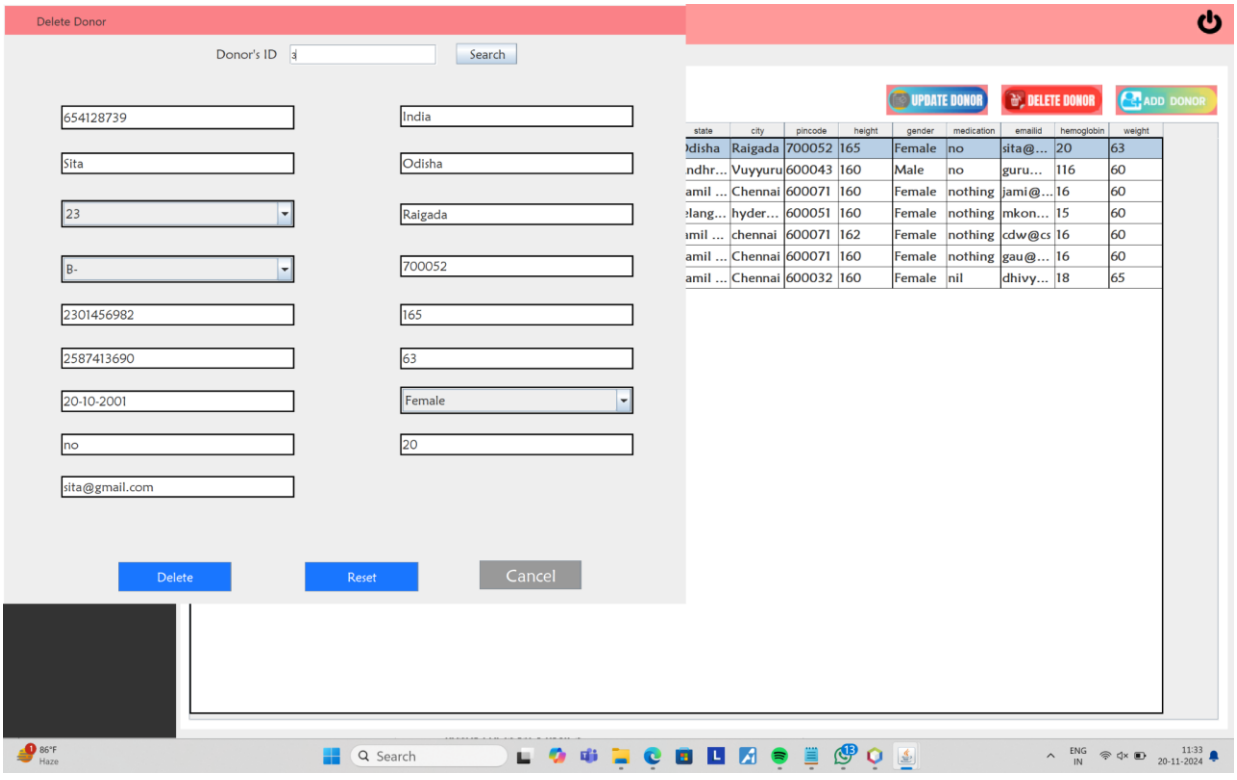
ADD DONOR PAGE DESIGN:



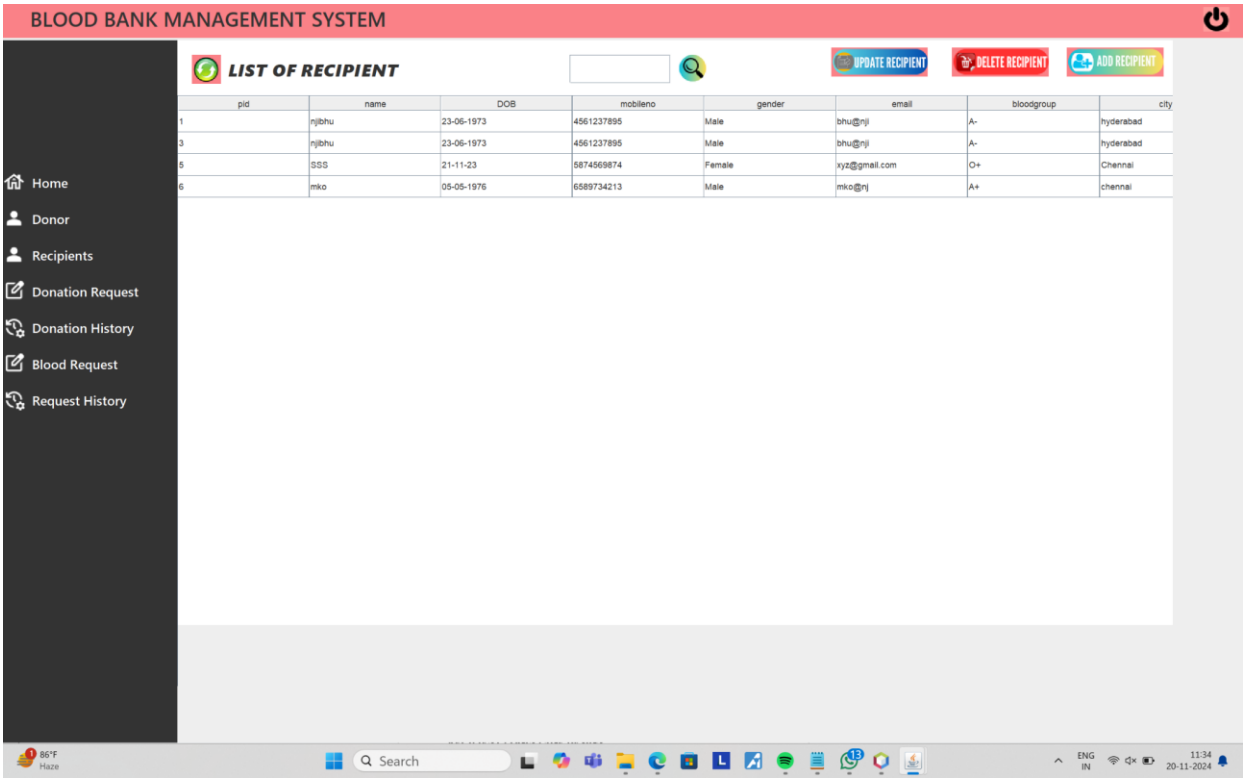
UPDATE DONOR PAGE DESIGN:



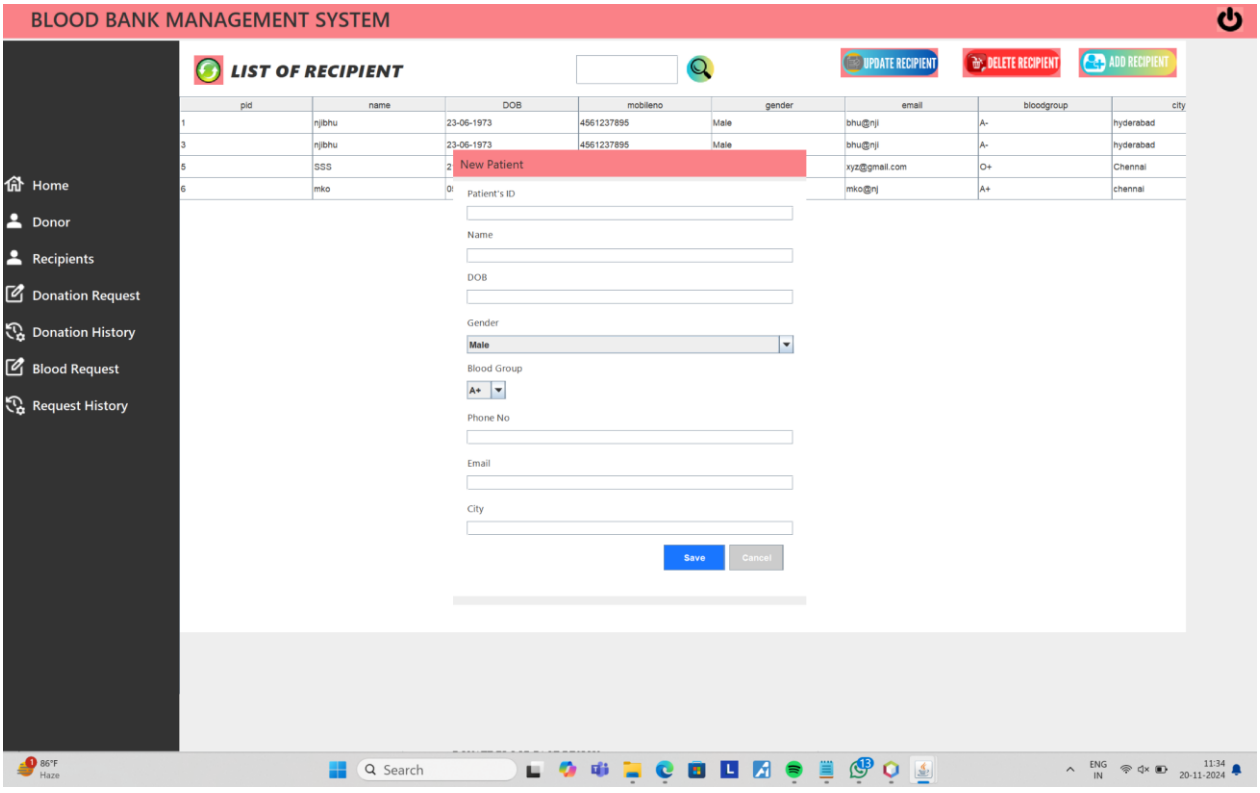
DELETE DONOR PAGE DESIGN:



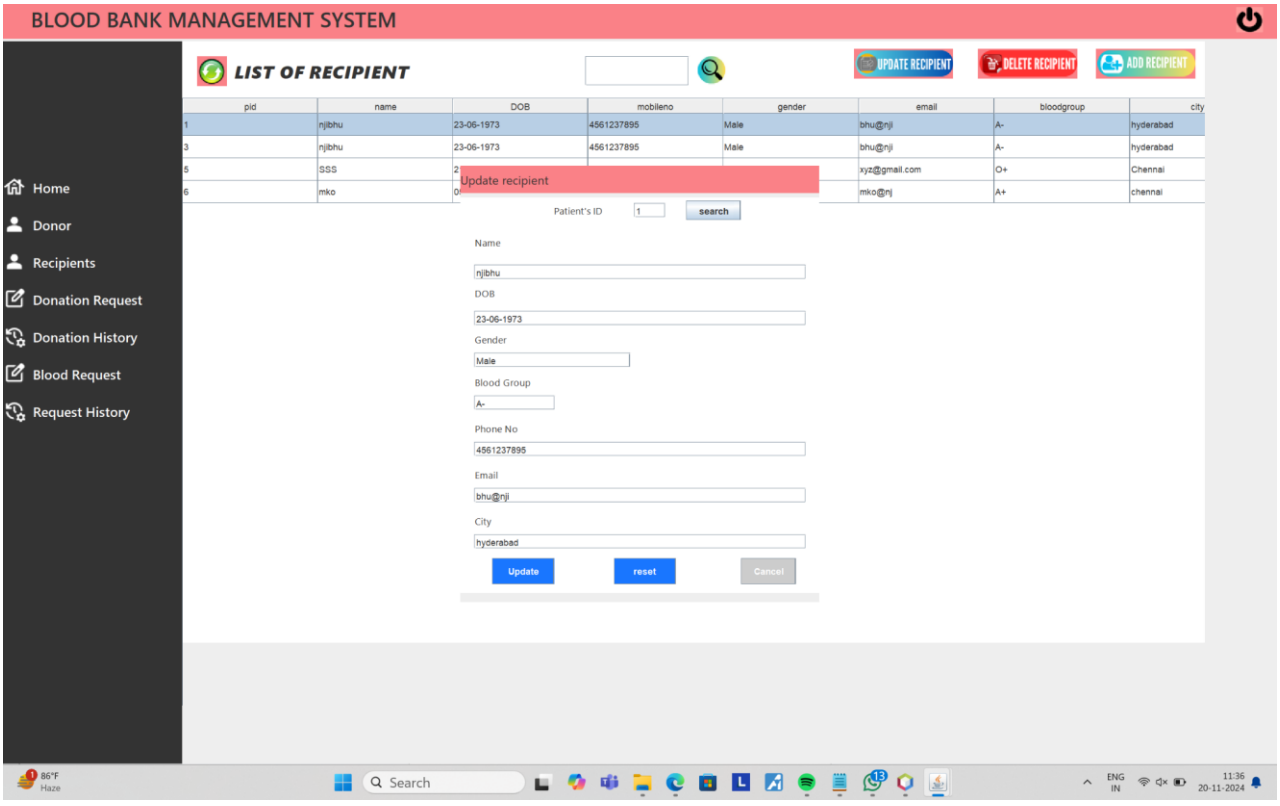
RECIPIENT LIST PAGE DESIGN:



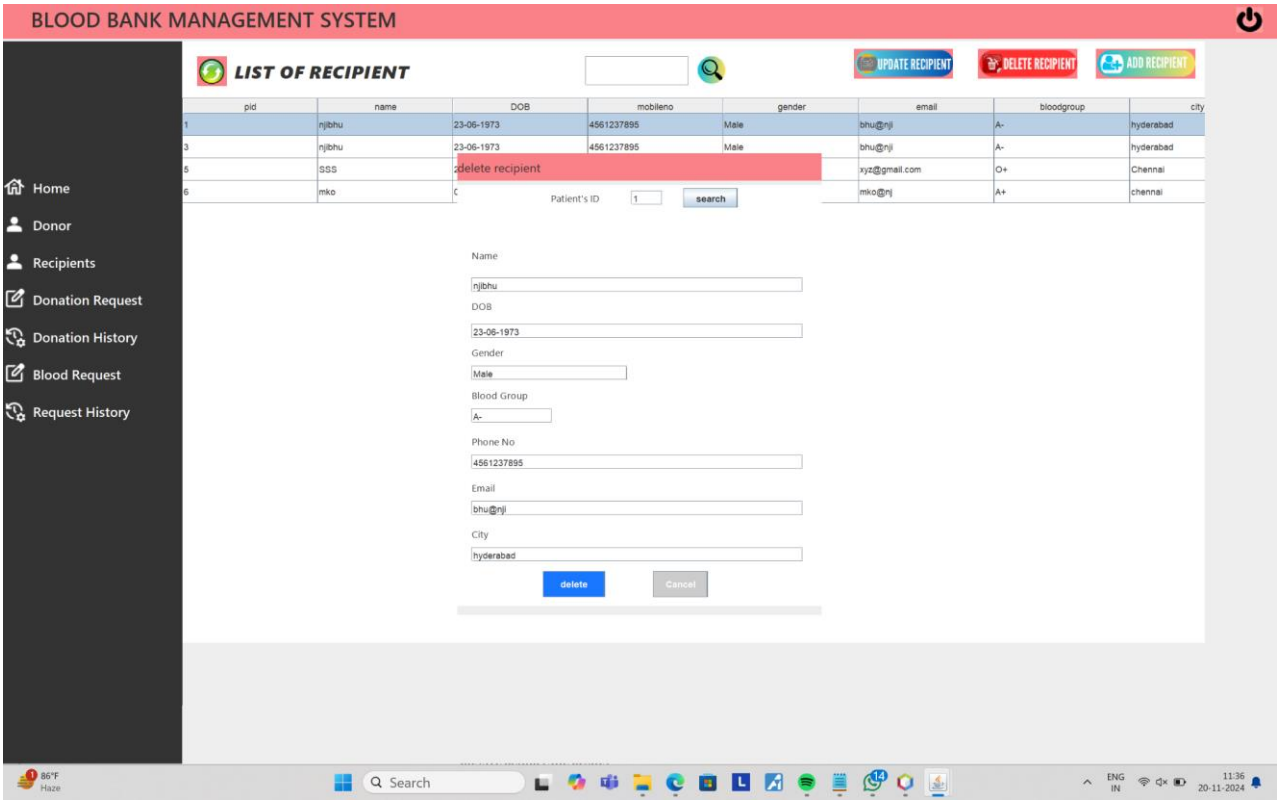
ADD RECIPIENT PAGE DESIGN:



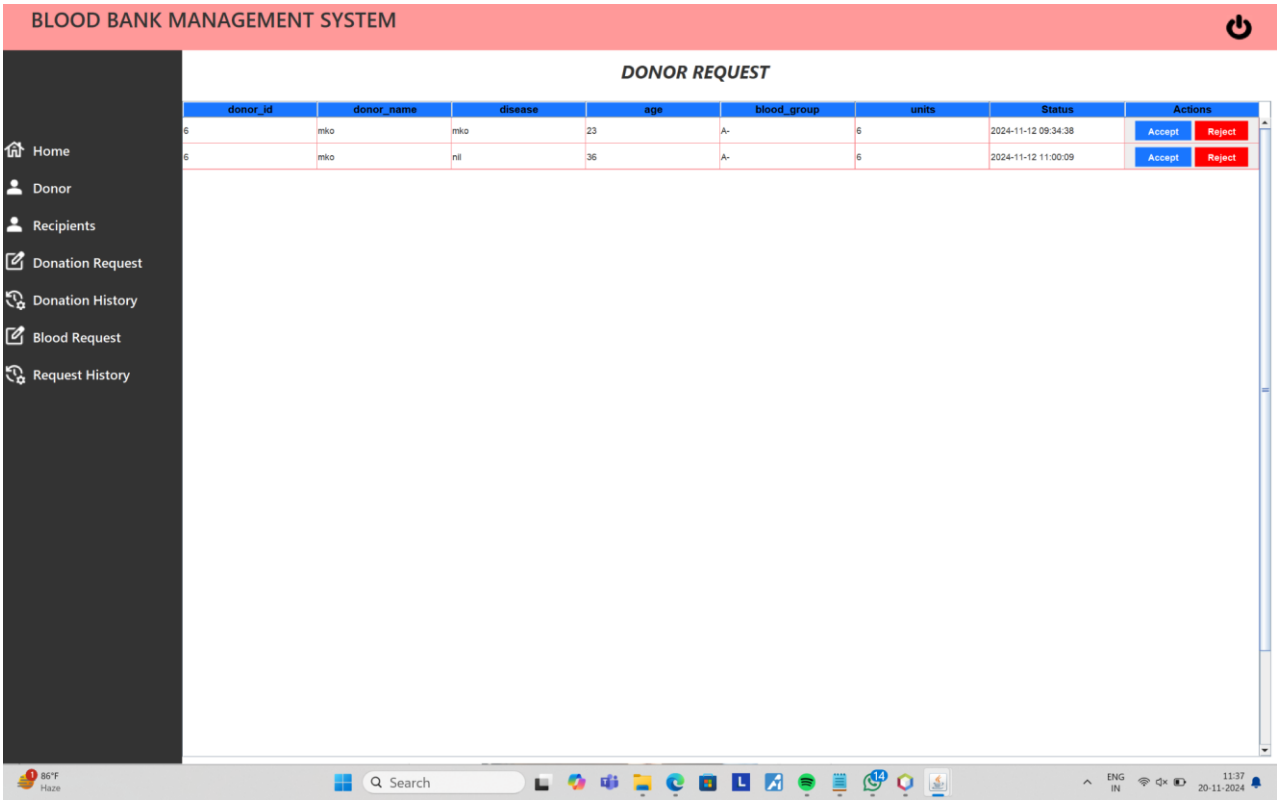
UPDATE RECIPIENT PAGE DESIGN:



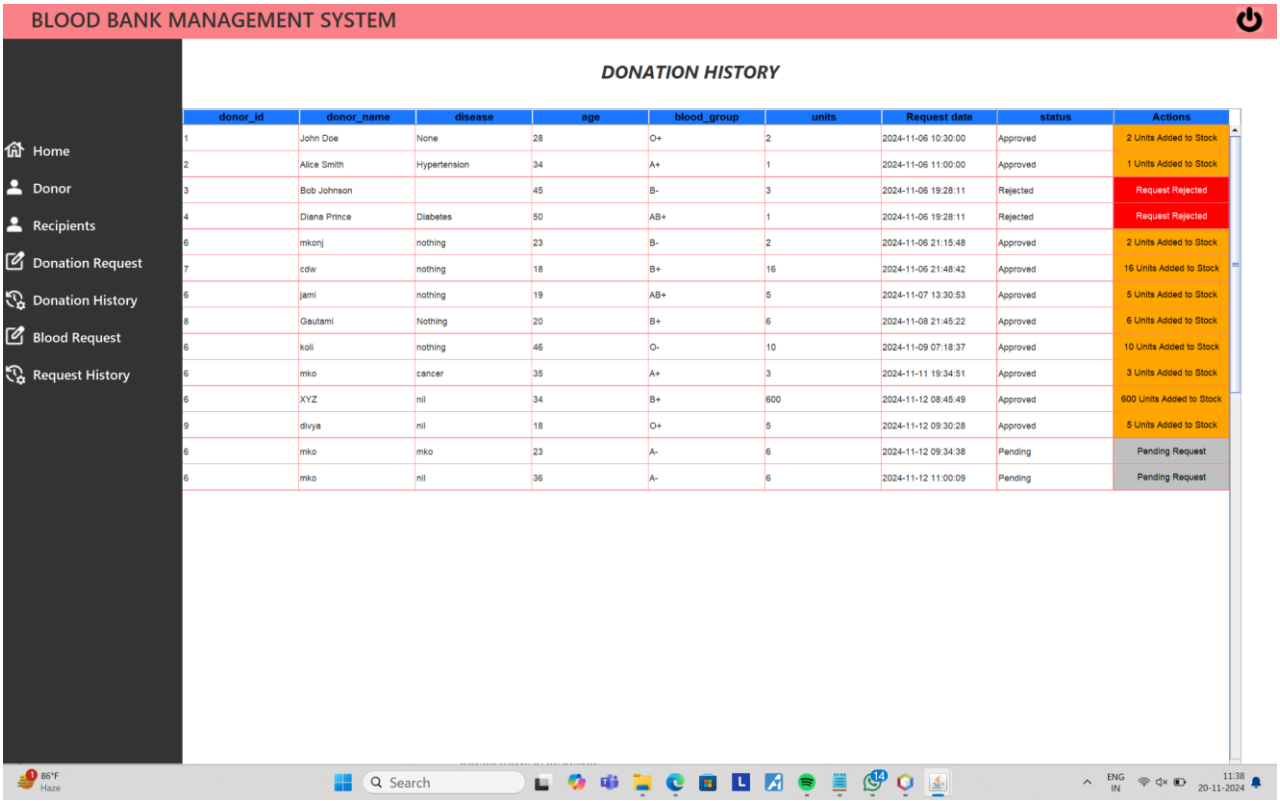
DELETE RECIPIENT PAGE DESIGN:



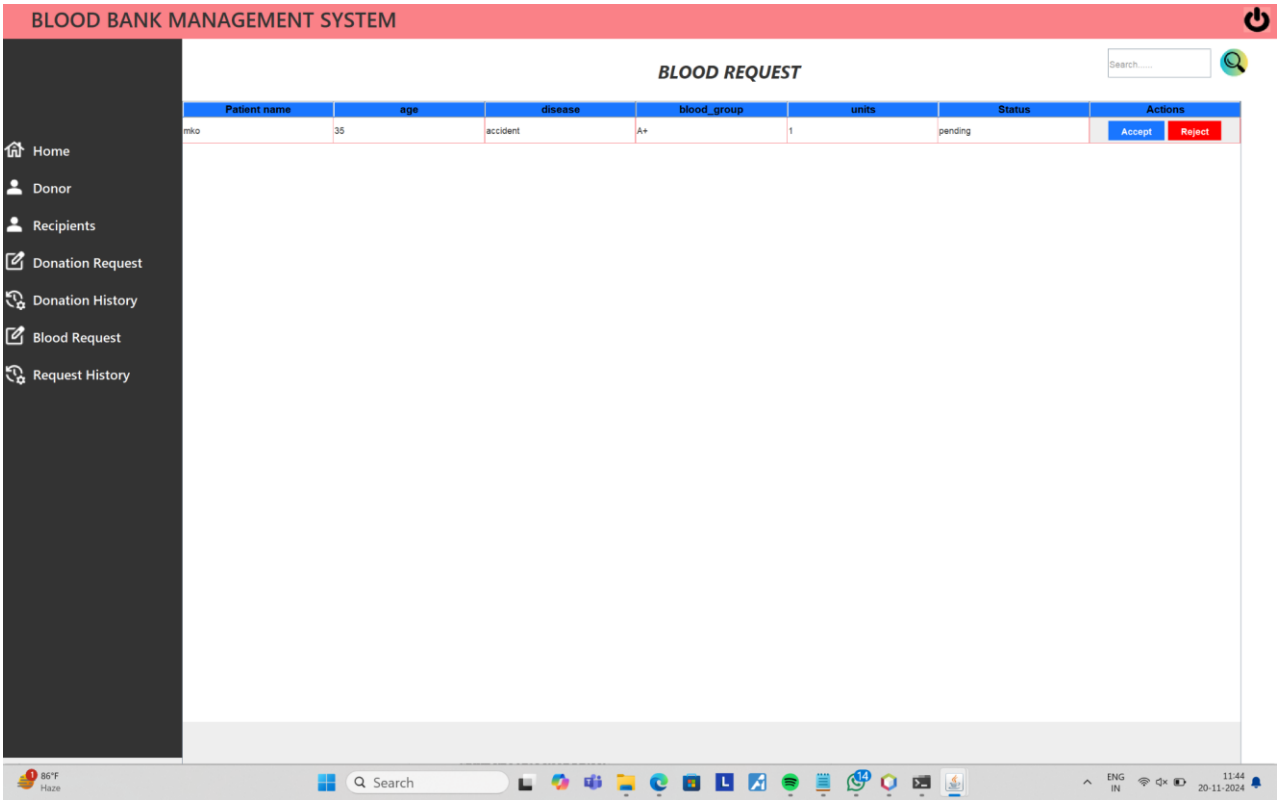
DONOR REQUEST PAGE DESIGN:



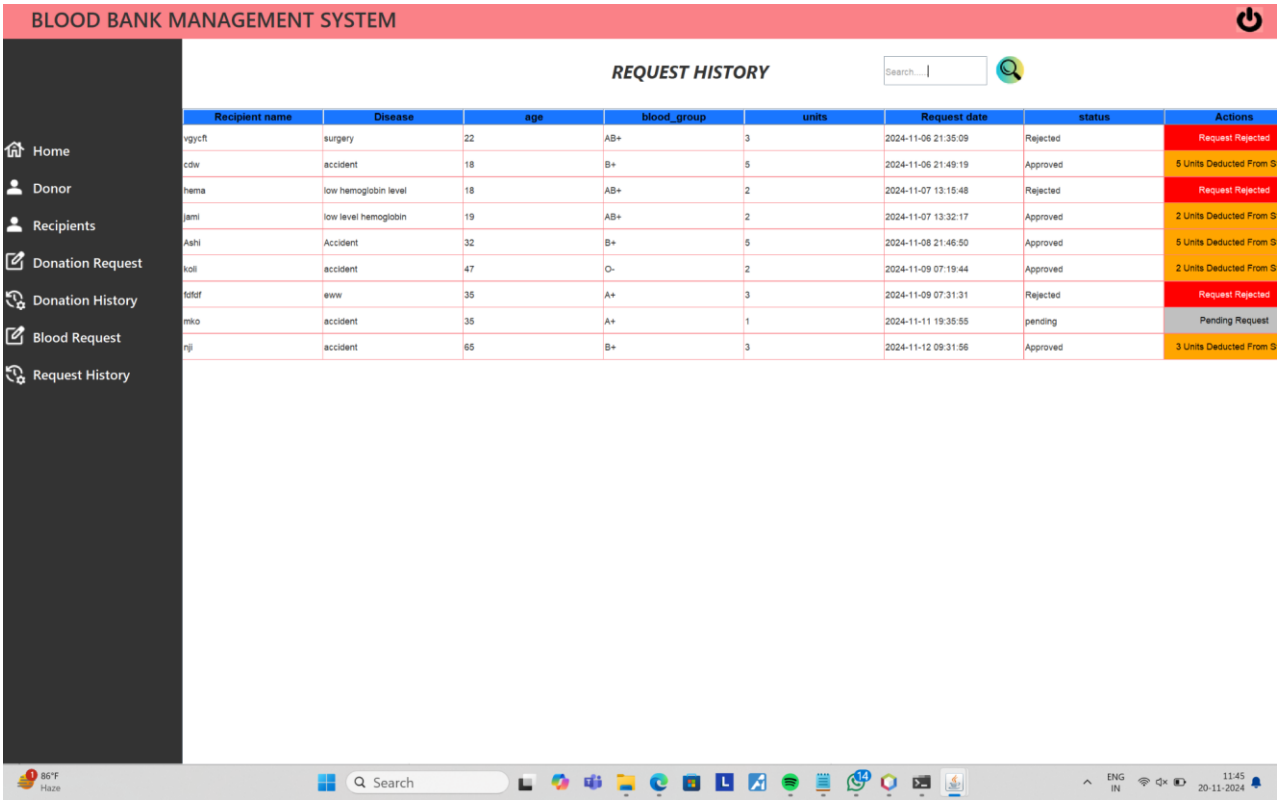
DONATE REQUEST HISTORY PAGE DESIGN:



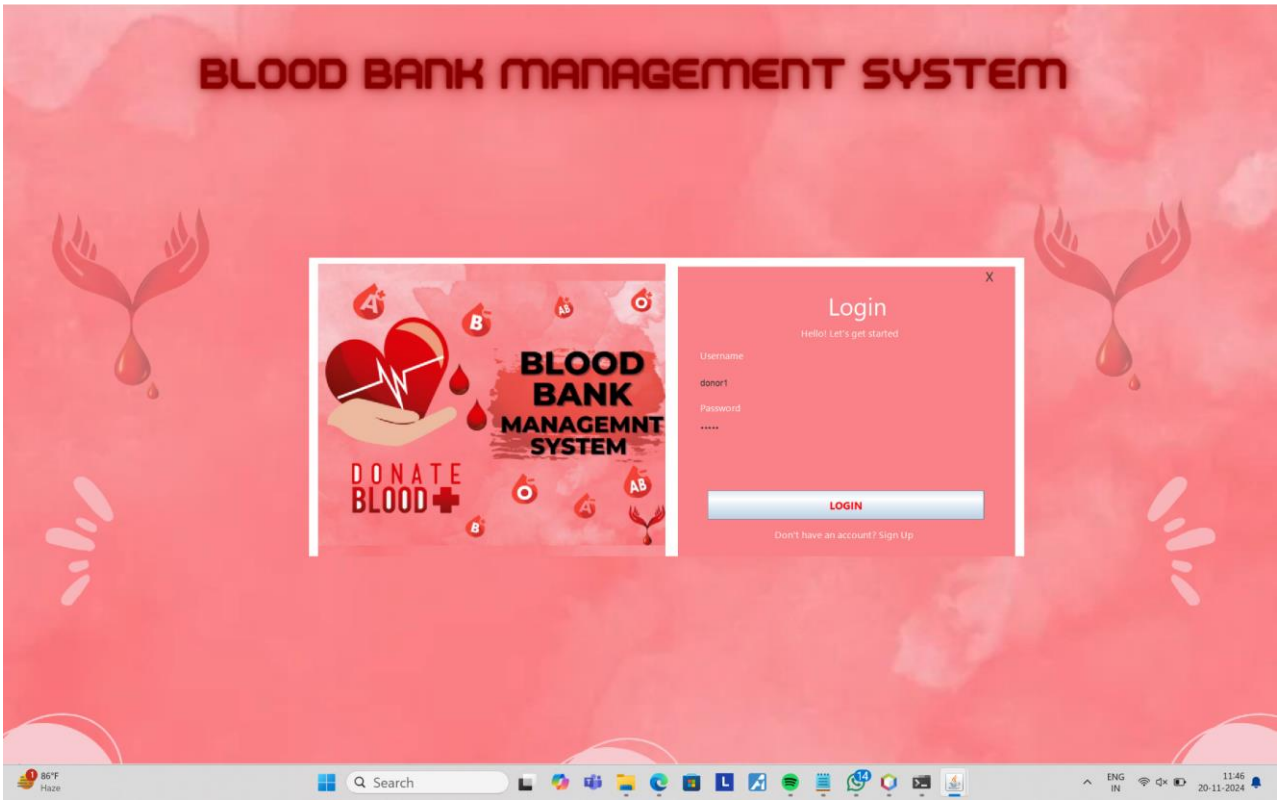
BLOOD REQUEST PAGE DESIGN:



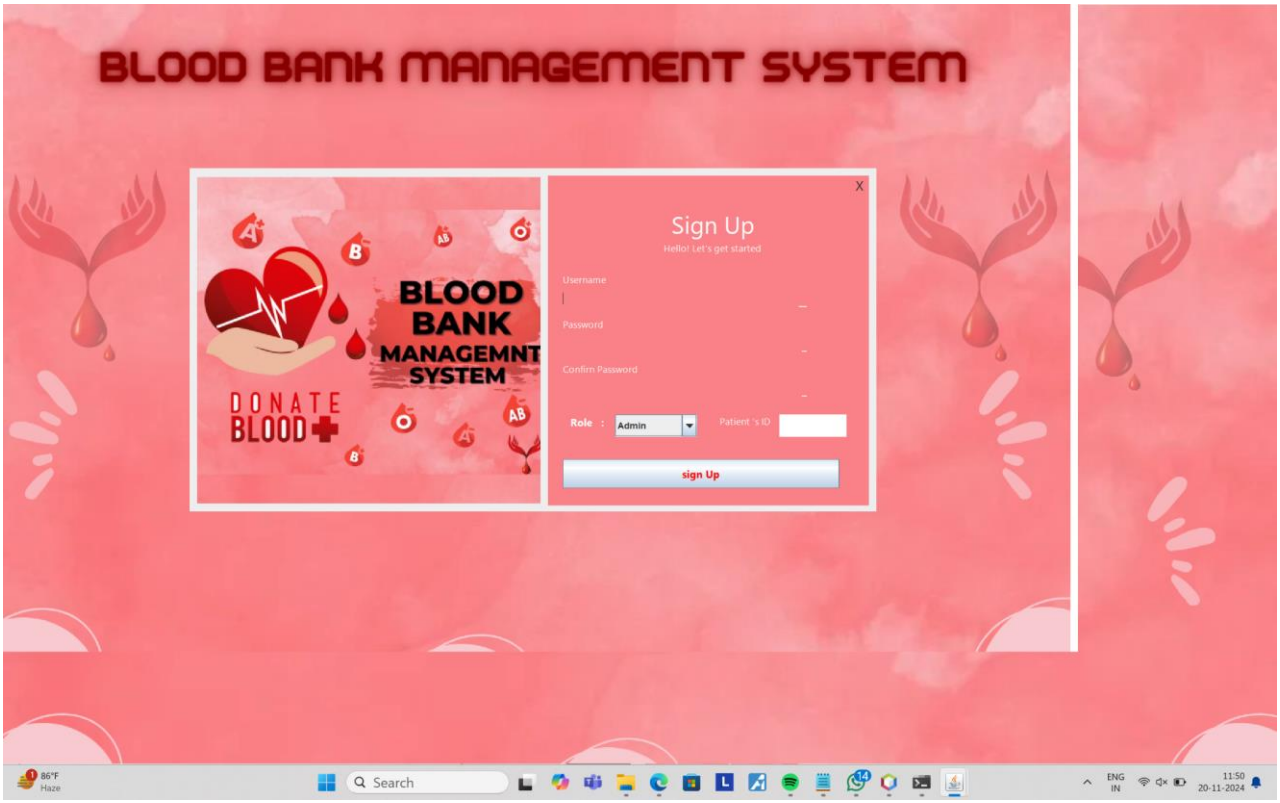
BLOOD REQUEST HISTORY PAGE DESIGN:



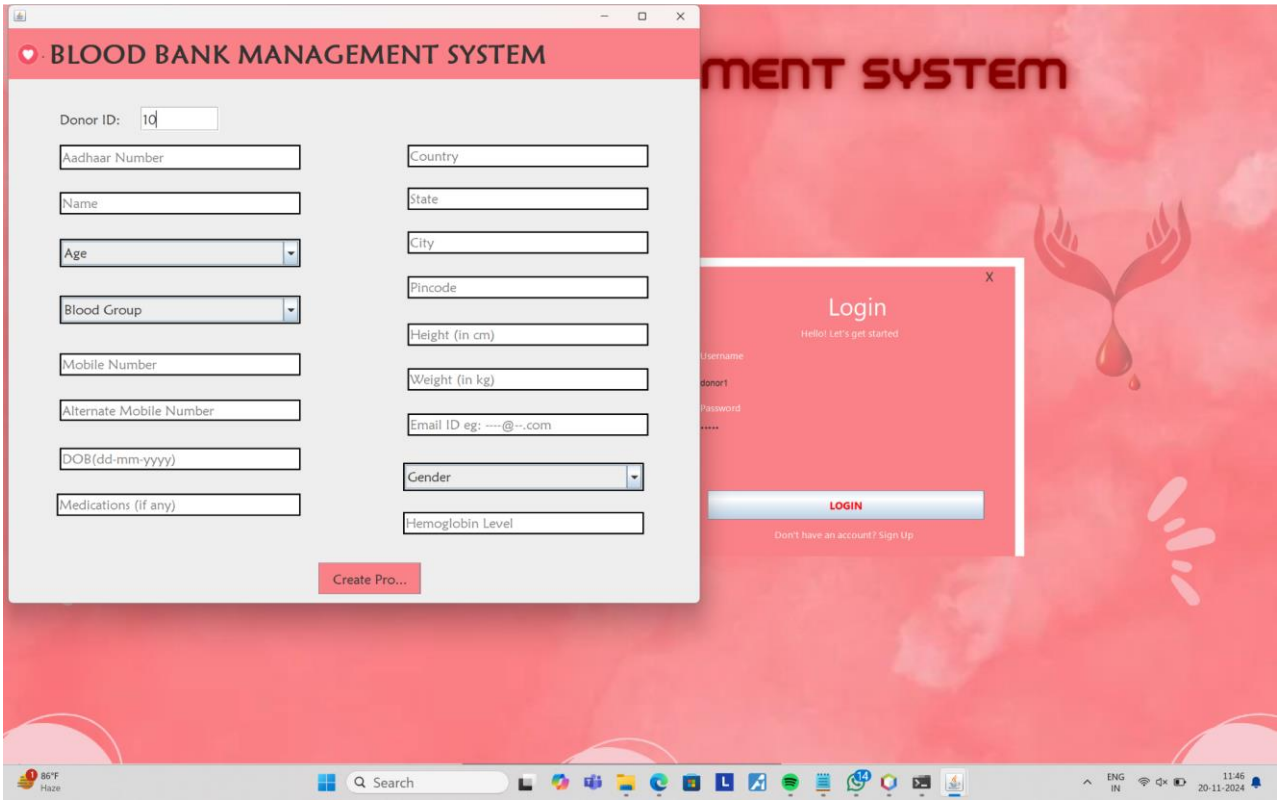
DONOR LOGIN PAGE DESIGN:



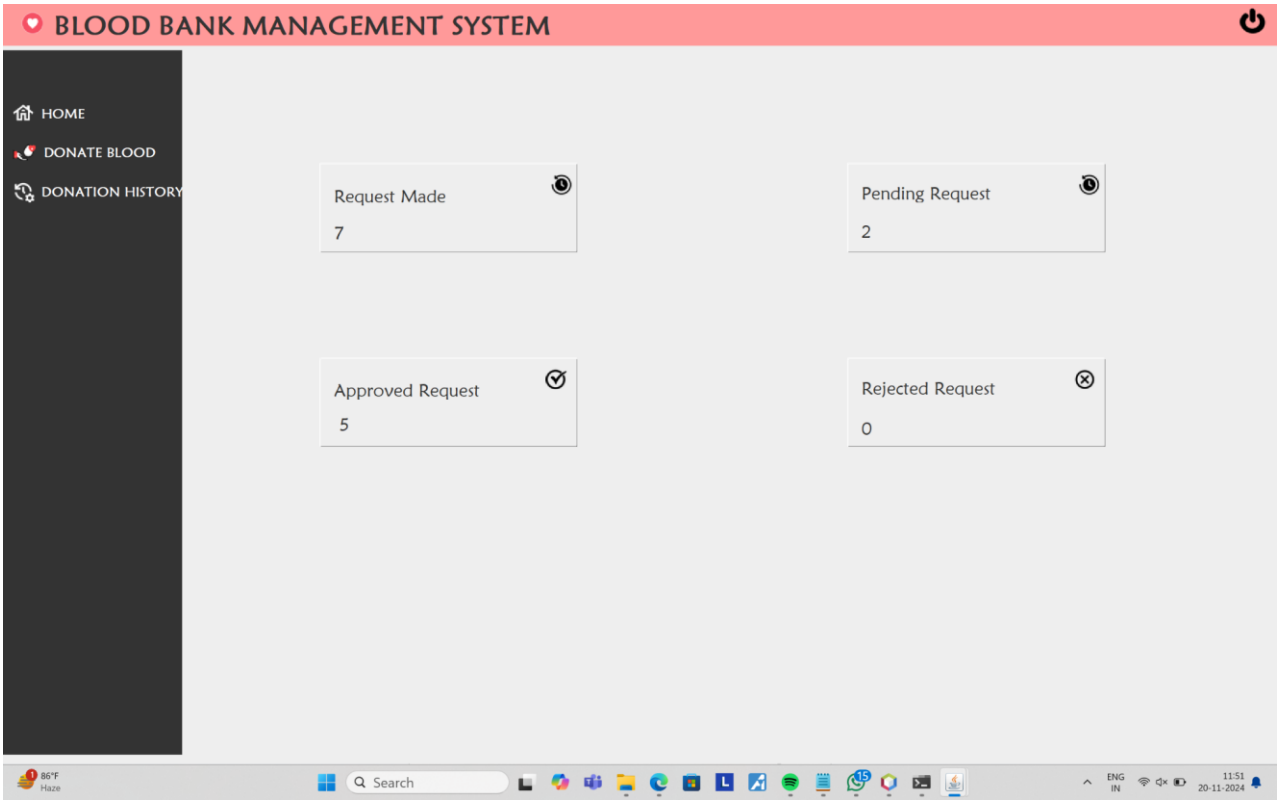
DONOR SIGNUP PAGE DESIGN:



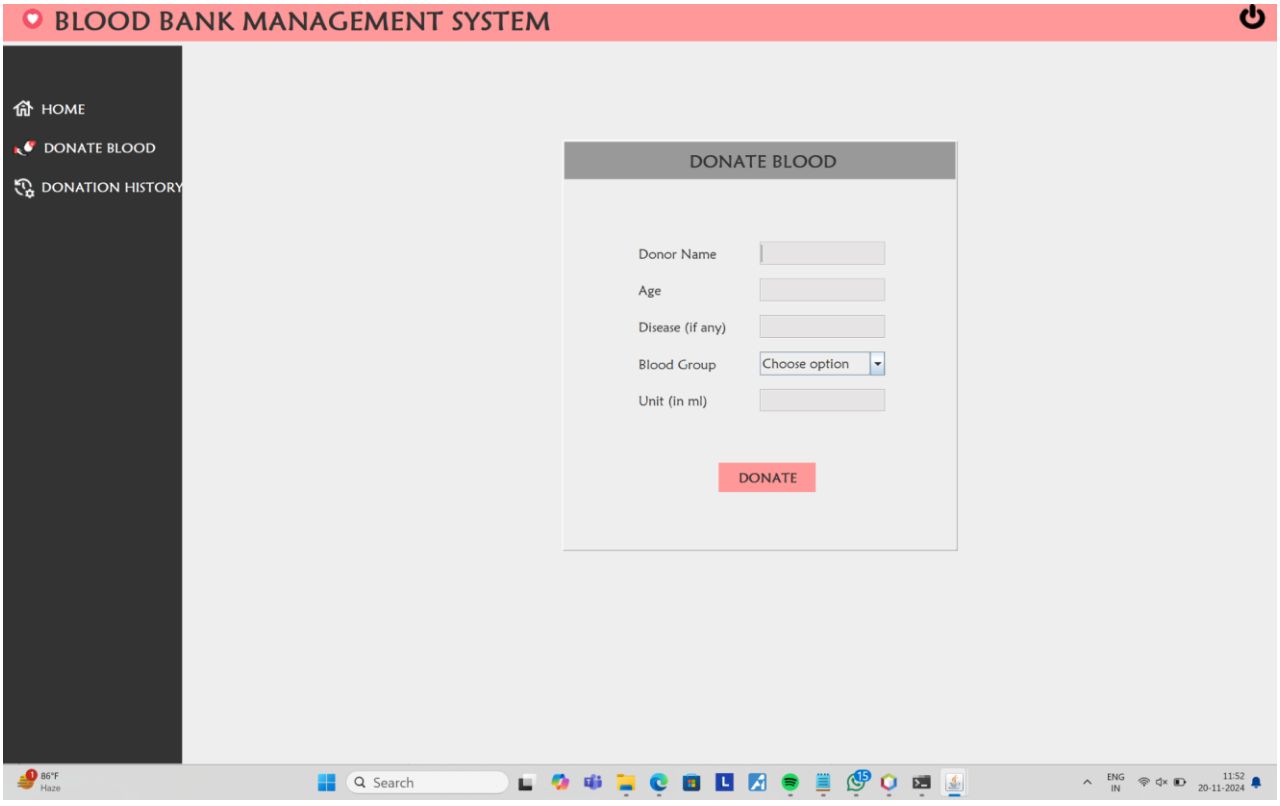
DONOR CREATE PROFILE PAGE DESIGN:



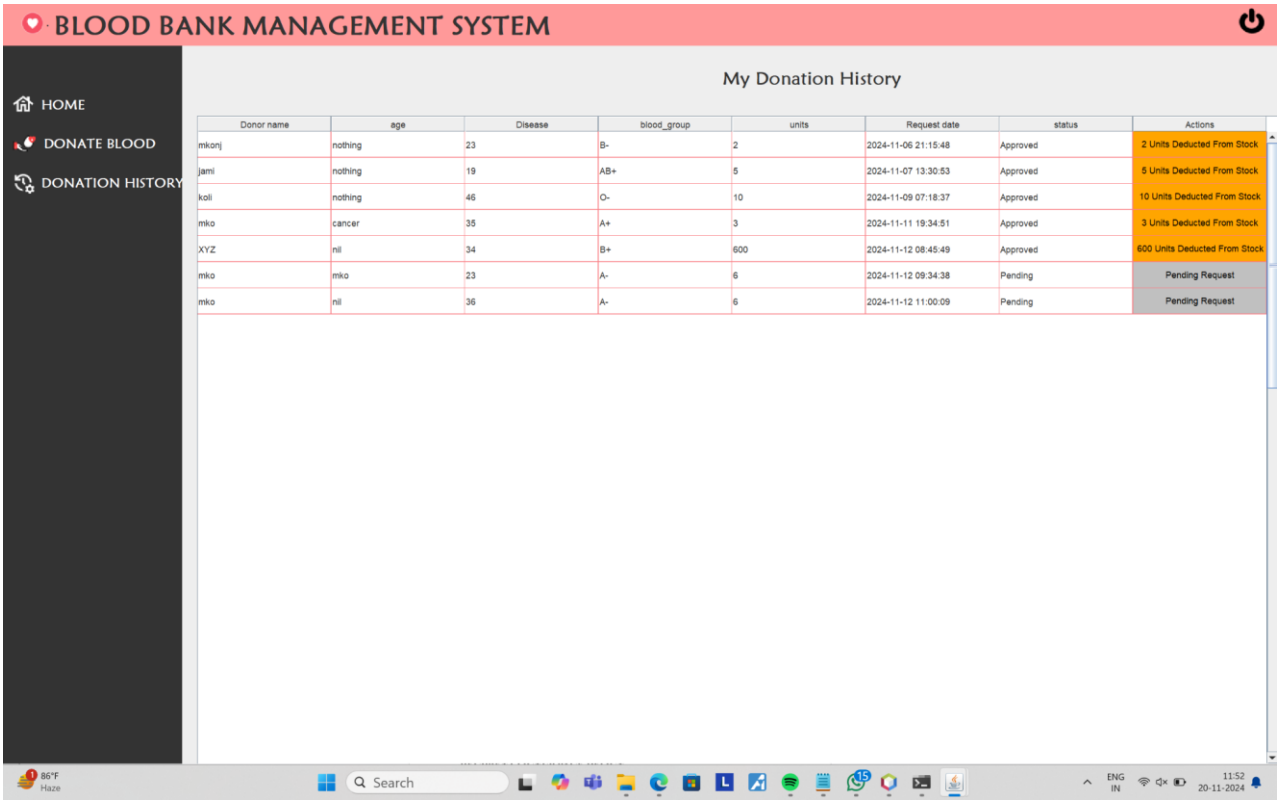
DONOR HOME PAGE DESIGN:



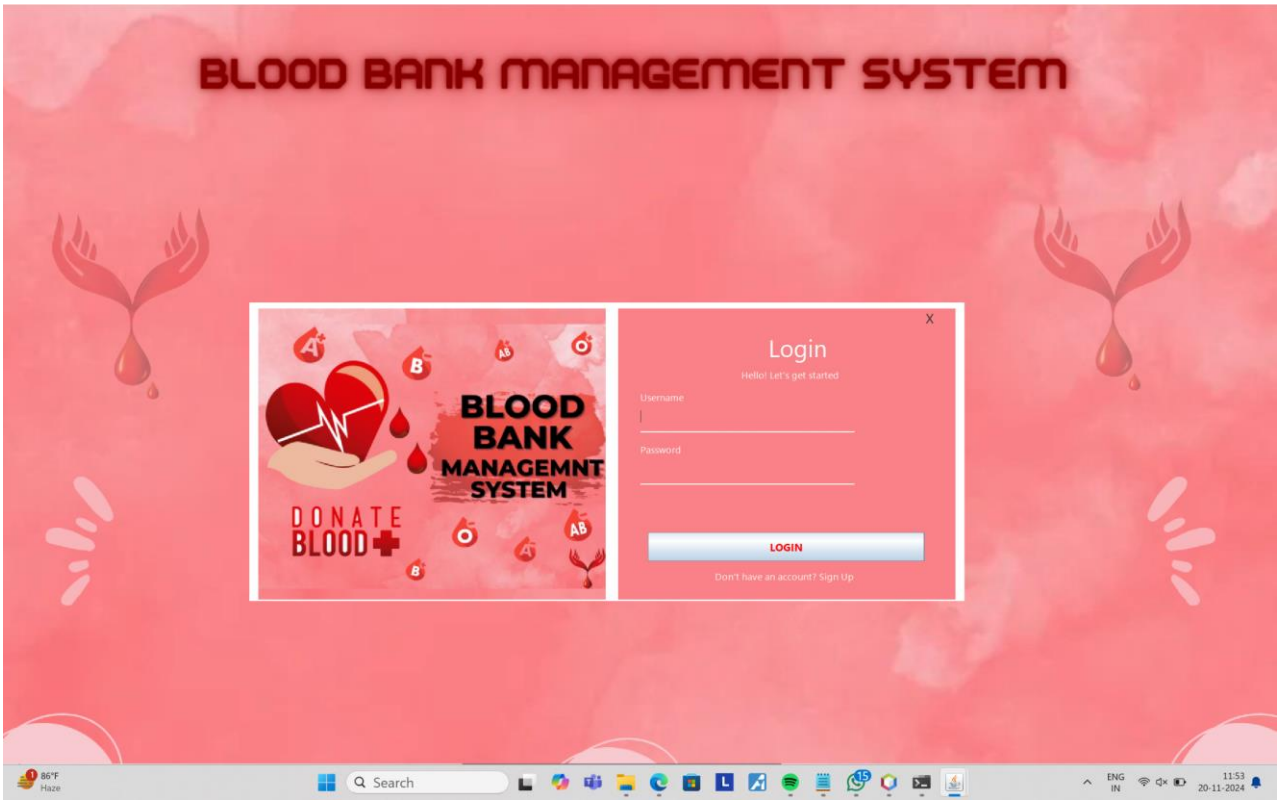
DONATE BLOOD PAGE DESIGN:



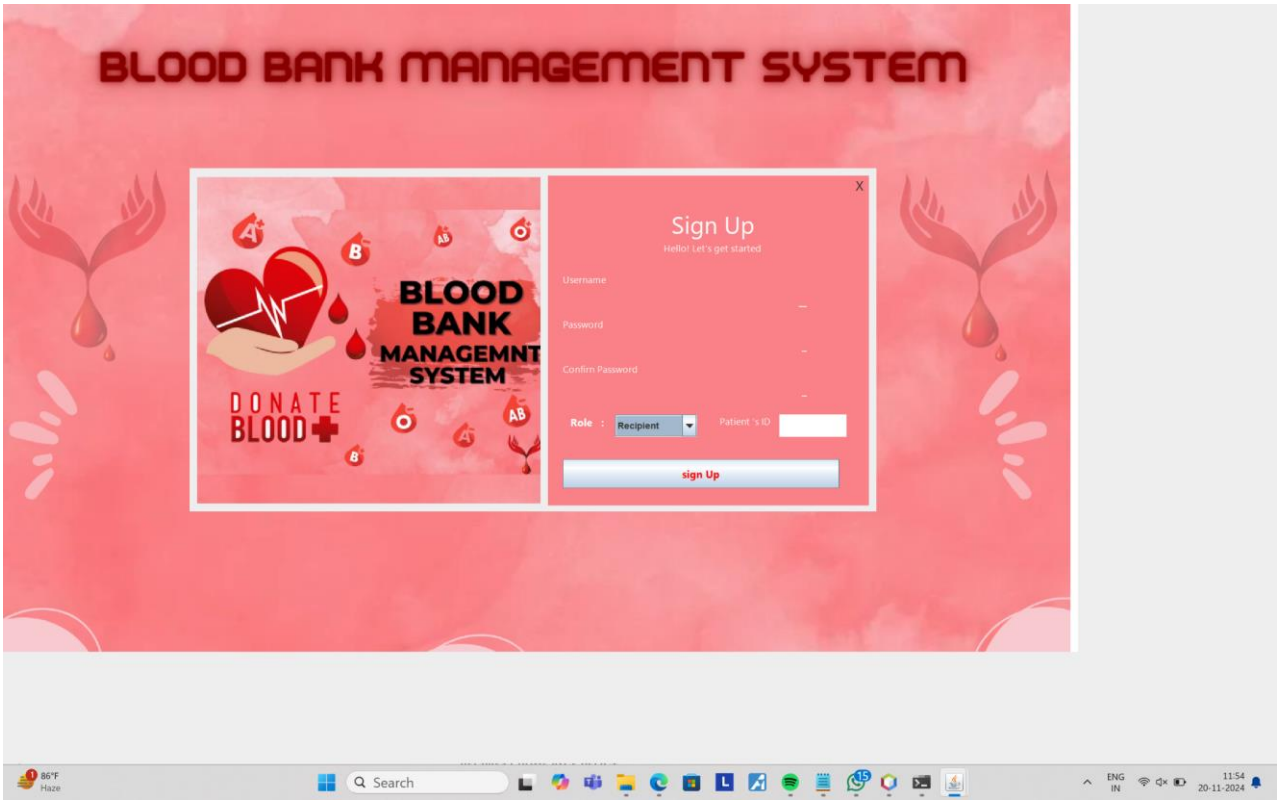
DONATION HISTORY PAGE DESIGN:



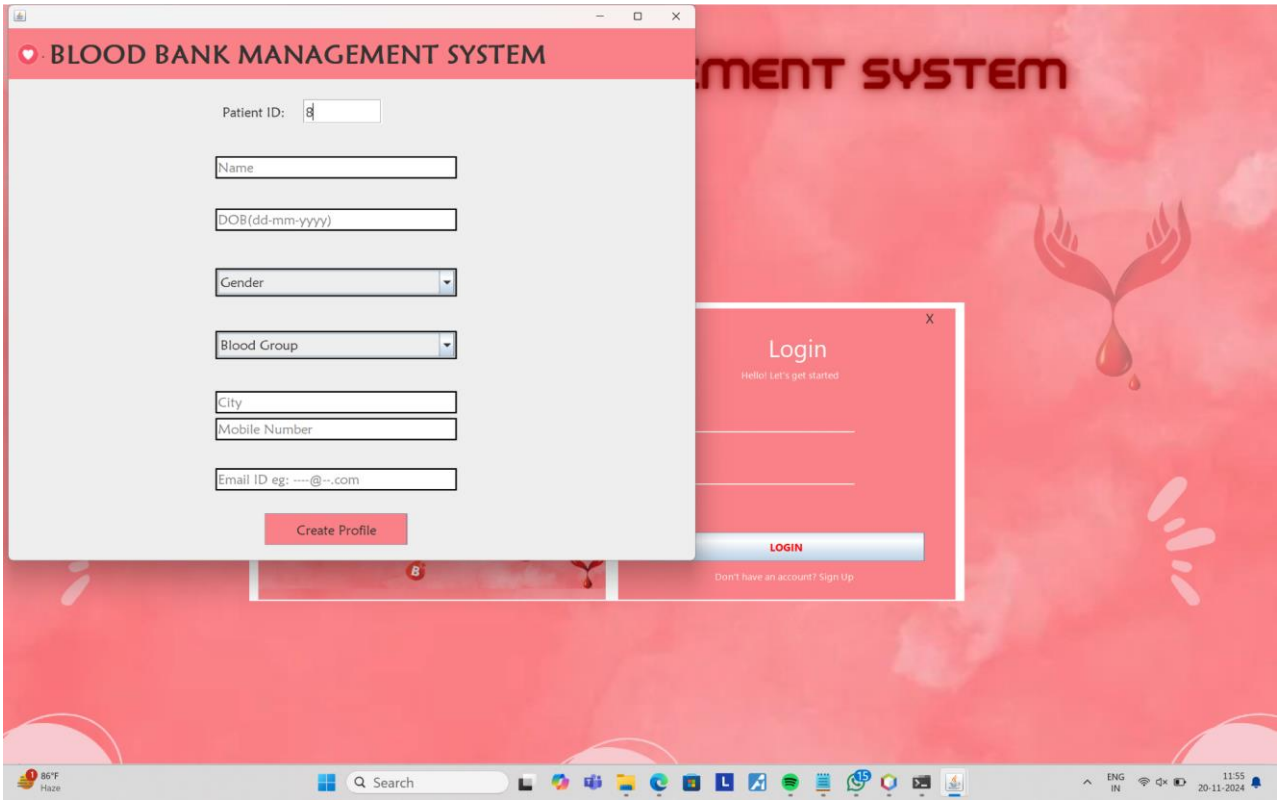
RECIPIENT LOGIN PAGE DESIGN:



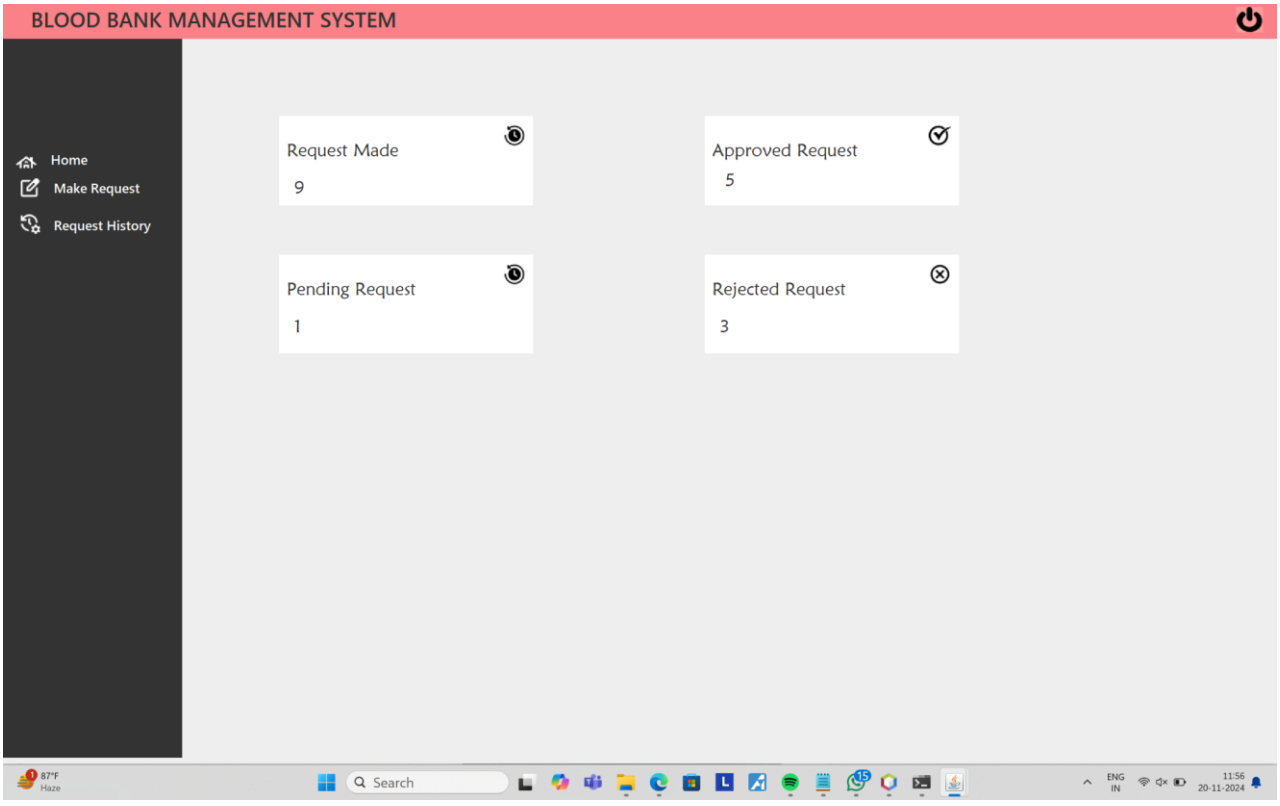
RECIPIENT SIGNUP PAGE DESIGN:



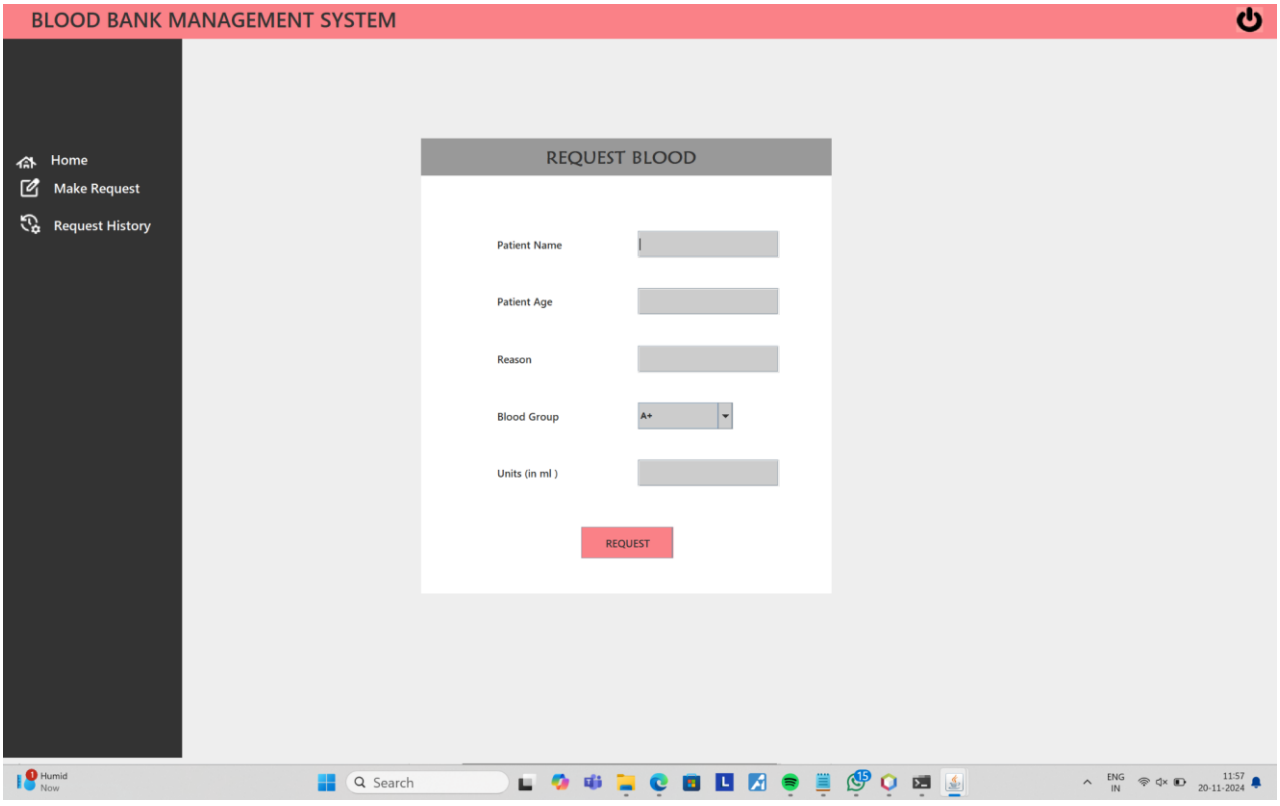
RECIPIENT CREATE PROFILE PAGE DESIGN:



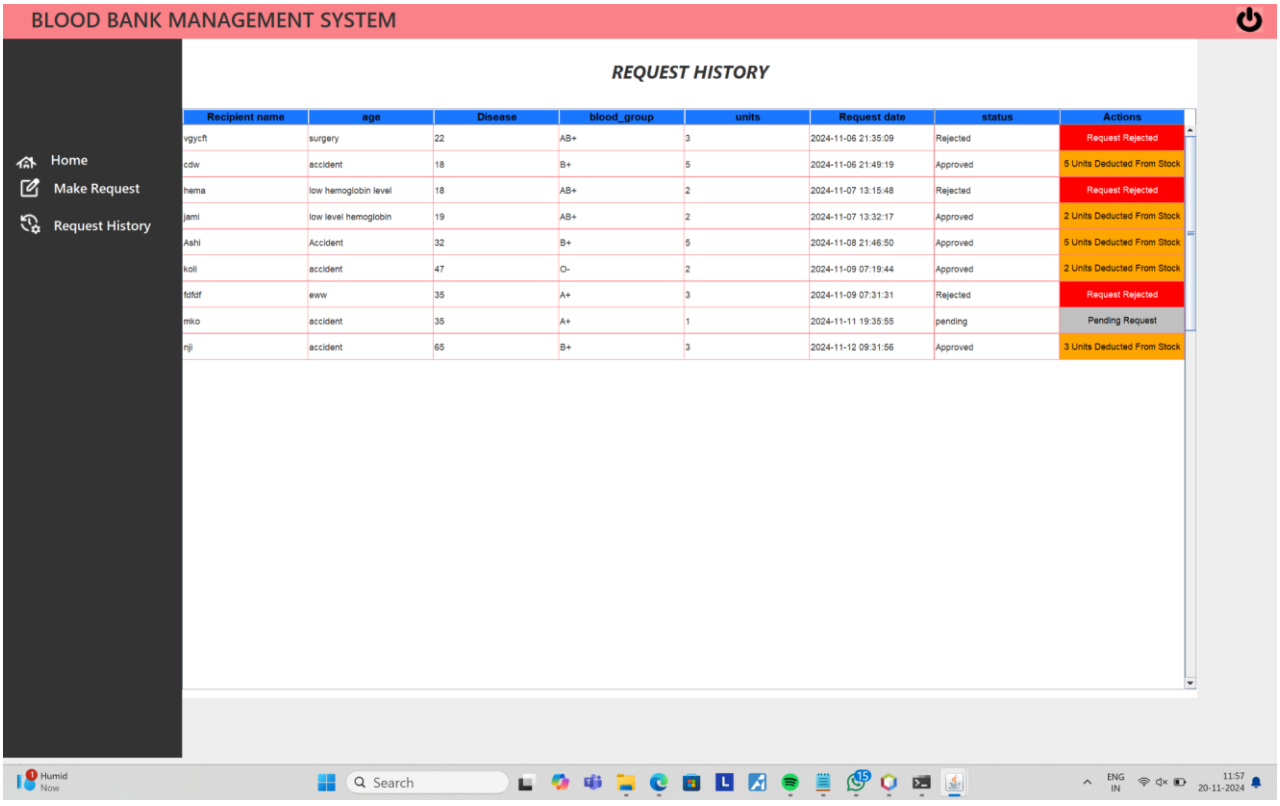
RECIPIENT HOME PAGE DESIGN:



MAKE REQUEST PAGE DESIGN:



REQUEST MADE HISTORY PAGE DESIGN:



Chapter 6 CONCLUSION

6.1 Conclusion

The “Blood Bank Management System” is designed to efficiently manage blood donations, requests, and stock levels, ensuring a streamlined workflow. Built using Java Swing for the frontend and MySQL for database management, the system provides a robust platform for handling essential operations in blood banks. By addressing the key challenges of manual management, the system ensures accuracy, timeliness, and reliability.

The system caters to three primary user roles: “Administrator, Donor, and Recipient”. Each role has distinct functionalities, enabling smooth operations and user satisfaction. Administrators manage blood stocks and approve requests, donors register and submit donation requests, and recipients can search and request required blood units. These role-specific features enhance the overall efficiency and effectiveness of the system.

Automation plays a significant role in the system by minimizing manual errors and delays. Real-time updates in blood stock ensure accurate tracking and timely decision-making. The system also facilitates seamless communication between donors and recipients, ensuring that blood donations are utilized effectively to meet urgent needs.

A user-friendly interface ensures easy access to the system, encouraging participation from donors and recipients alike. By simplifying operations and providing clear functionalities, the system fosters a better experience for all users while promoting transparency and accountability in managing blood stocks.

In summary, the Blood Bank Management System demonstrates the power of technology in transforming healthcare services. It provides an efficient, scalable, and user-centric solution to ensure that blood donations are managed effectively, contributing to saving lives and supporting healthcare infrastructure.

Chapter 7 REFERENCES

7.1 REFERENCES

1. Books:
- "Database System Concepts" by Silberschatz, Korth, and Sudarshan – Useful for database design principles.
- "Java: The Complete Reference" by Herbert Schildt – Comprehensive guide for Java programming, including Swing.

2. Online Documentation:
- Java Swing Documentation: https://docs.oracle.com/javase/tutorial/uiswing/
- MySQL Reference Manual: https://dev.mysql.com/doc/
- GeeksforGeeks - Java Swing Tutorials: https://www.geeksforgeeks.org/java-swing/

3. Research Papers:
- "A Novel Approach for Blood Bank Management System" – Discusses modern strategies for automating blood banks (Available on IEEE Xplore).
- "Automated Blood Bank System" – Published in IJCSIT, explores system design and features.

4. Case Studies:
- “E-RaktKosh”: https://www.eraktkosh.in/ – A national blood bank management system in India.
- “Indian Blood Donors”: https://www.indianblooddonors.com/ – Connects donors and recipients efficiently.

5. Additional Resources:
- GitHub Projects: https://github.com/ – For exploring similar blood bank management system projects.
- Stack Overflow: https://stackoverflow.com/ – For coding issue resolutions.

These references provide foundational knowledge, technical guidance, and real-world examples relevant to the Blood Bank Management System.