****

**Name : LOKESH BOOTU**

**Roll no : 1/23/SET/BCS/435**

**Course : CSE-AIML**

**Section : 3AIML-B**

**Subject : Azure Ai Fundamentals**

**Subject code : BCS –DS-375**

**Submitted to : Mrs. Swati Hans**

**Blood Bank Management System**

**Introduction**

The **Blood Bank Management System (BBMS)** is a software application designed to efficiently manage and streamline operations in a blood bank. The management of vital data, including donor and receiver records, blood supply levels, and the processing of blood requests, is made easier by this system. The BBMS seeks to improve the overall administration of blood donation and distribution procedures by combining several functionalities into a single platform, guaranteeing that individuals in need have easy access to life-saving supplies.

### Problem Statement: Blood Bank Management System

The **Blood Bank Management System (BBMS)** addresses the challenges faced by blood banks in maintaining efficient and accurate operations. Blood banks often struggle with managing donor records, tracking recipients, monitoring blood stock, and ensuring the timely fulfillment of blood requests. These challenges can lead to inefficiencies, stock mismatches, and delays during emergencies, ultimately impacting the ability to save lives.

### ****Challenges to Address****

1. **Inefficient Donor and Recipient Management:**
   * Difficulty in maintaining updated donor and recipient records.
   * Challenges in quickly locating suitable donors or recipients based on specific criteria (e.g., blood type).
2. **Blood Stock Monitoring:**
   * Manual tracking of blood stock, leading to potential errors.
   * Unavailability of real-time information about current stock levels for each blood group.
3. **Fulfillment Delays:**
   * Inefficient processes for matching blood requests with available stock.
   * Inability to address urgent blood requirements due to poor organization.
4. **Lack of Access Control:**
   * Unauthorized access to sensitive data and operational features.
   * Risk of data tampering or loss of critical information.

### ****Objectives of the Blood Bank Management System****

To solve these challenges, the system aims to:

1. Automate the management of donor and recipient records.
2. Provide real-time updates and tracking of blood stock levels for all blood types.
3. Enable fast and efficient search functionality for donors and recipients.
4. Streamline the process of fulfilling blood requests during emergencies.
5. Implement secure access control to prevent unauthorized usage.

### ****Proposed Solution****

The Blood Bank Management System will:

1. **Maintain Comprehensive Records:**
   * Store detailed information about donors (name, blood type, age, contact) and recipients (name, age, blood group, contact).
2. **Streamline Blood Stock Management:**
   * Record donations and update stock levels in real-time.
   * Monitor the availability of each blood group to ensure supply meets demand.
3. **Enhance Search and Retrieval:**
   * Allow admins to search donors and recipients based on blood type, name, or other criteria.
4. **Fulfill Requests Efficiently:**
   * Match blood requests to available stock and notify if stock levels are insufficient.
5. **Secure System Access:**
   * Authenticate admin users to restrict access to critical operations like adding or deleting records, updating stock, and fulfilling requests.

### ****Scope of the System****

The system is intended for:

* Blood banks looking to modernize their management processes.
* Hospitals and clinics that depend on blood banks for emergency supplies.
* Organizations managing blood donation camps to maintain donor information.

By implementing the Blood Bank Management System, organizations can overcome inefficiencies and delays, ensuring that life-saving blood is available when needed most.

**Requirements for Blood Bank Management System (BBMS)**

**1. Hardware Requirements**

**Server-Side Hardware (if hosted centrally):**

* **Processor:** Intel Core i5 or higher (or equivalent)
* **RAM:** 8 GB or more
* **Storage:** 500 GB HDD or 256 GB SSD (minimum)
* **Network:** Reliable internet connection for multi-user access
* **Operating System:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS

**Client-Side Hardware:**

* **Processor:** Intel Core i3 or higher
* **RAM:** 4 GB or more
* **Storage:** 250 GB HDD or higher
* **Monitor:** Minimum resolution of 1024x768
* **Input Devices:** Keyboard, mouse
* **Printer (Optional):** For generating reports or receipts

**2. Software Requirements**

**Development Environment:**

* **Programming Language:** C++
* **Compiler:** GCC or Visual C++ (any standard C++ compiler)
* **IDE:** Visual Studio, Code::Blocks, Eclipse CDT, or similar

**Operating System:**

* Windows, Linux, or macOS (compatible with chosen compiler and IDE)

**Database (optional, for advanced systems):**

* SQLite, MySQL, or PostgreSQL (if persistent storage is needed)

**Additional Software (optional):**

* Text editors like Notepad++ or Visual Studio Code for editing source files
* Git for version control

**3. Functional Requirements**

**Core Features:**

1. **Donor Management:**
   * Add, update, delete, and view donor details.
   * Search donors by blood type or name.
2. **Recipient Management:**
   * Add, update, delete, and view recipient details.
   * Search recipients by blood group or name.
3. **Blood Stock Management:**
   * View current blood stock levels for all blood types.
   * Update stock quantities (e.g., after donations or requests).
4. **Request Fulfillment:**
   * Process blood requests by matching with available stock.
   * Notify users if stock is insufficient.
5. **Authentication and Access Control:**
   * Admin login for access to critical operations.
   * Restrict unauthorized access to sensitive data.

**Additional Features:**

* Generate and display reports of donors, recipients, and stock.
* Provide user-friendly error messages for invalid inputs.

**4. Non-Functional Requirements**

1. **Performance:**
   * Fast and efficient searches for donors, recipients, and blood stock.
   * Real-time updates to stock levels after transactions.
2. **Usability:**
   * Simple, menu-driven interface for ease of use.
   * Input validation to minimize errors during data entry.
3. **Reliability:**
   * Accurate storage and retrieval of donor and recipient data.
   * Ensures no data is lost or corrupted during transactions.
4. **Scalability:**
   * Able to handle an increasing number of donors, recipients, and stock entries.
5. **Security:**
   * Admin login credentials to prevent unauthorized access.
   * Secure handling of sensitive information.
6. **Portability:**
   * The system should run on multiple platforms (Windows, Linux, or macOS).

These requirements will ensure the system operates smoothly, fulfills its objectives, and is robust enough to meet future needs.

**Actors:**

1. **Admin**:
   * Manages the blood bank system, including donors, recipients, stock, and fulfilling requests.
2. **General User** (Donors and Recipients):
   * Includes any user (non-admin) interacting with the system to view details or search records.

### ****Use Cases****:

#### Admin Use Cases:

1. **Login**
2. **Add Donor**
3. **Add Recipient**
4. **Delete Donor**
5. **Delete Recipient**
6. **Update Donor Details**
7. **Update Recipient Details**
8. **View Blood Stock**
9. **Update Blood Stock**
10. **Fulfill Blood Request**

#### General User Use Cases:

1. **View All Donors**
2. **View All Recipients**
3. **Search Donor by Blood Type**
4. **Search Donor by Name**
5. **Search Recipient by Blood Group**
6. **Search Recipient by Name**

### ****Relationships****:

1. **Admin** has access to **Admin Use Cases** after successful **Login**.
2. **General User** interacts with **General User Use Cases** directly without logging in.

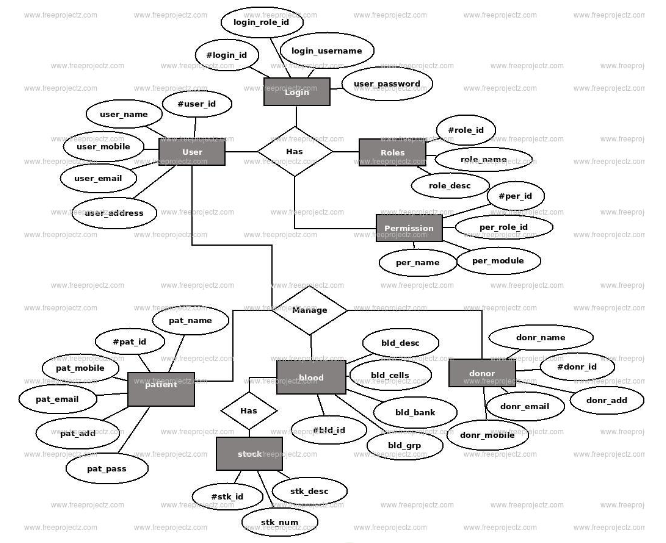
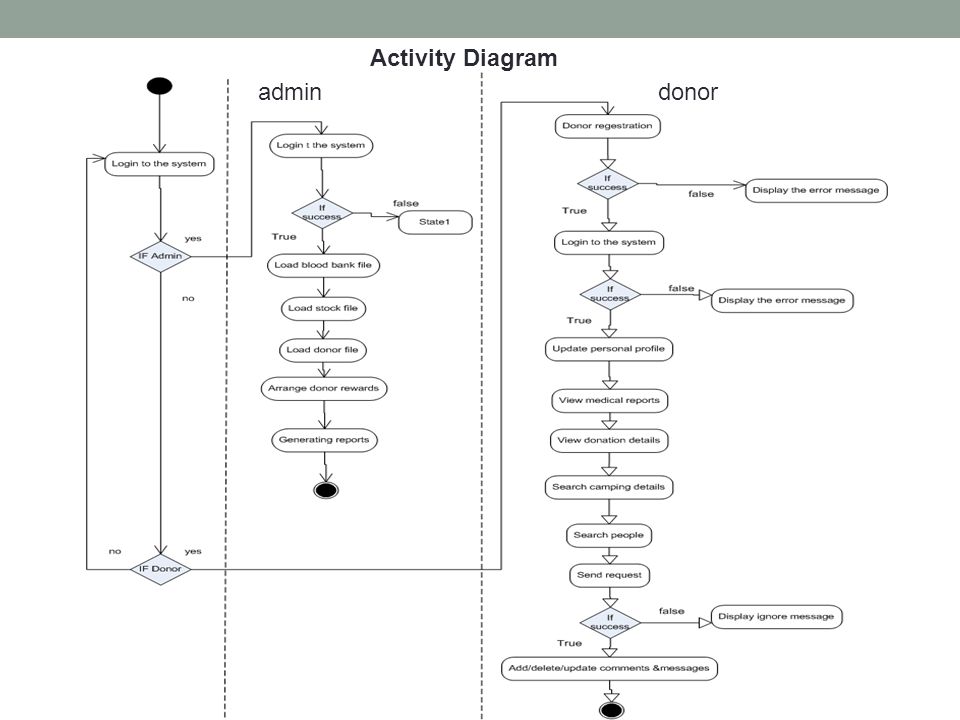
Use case diagrams

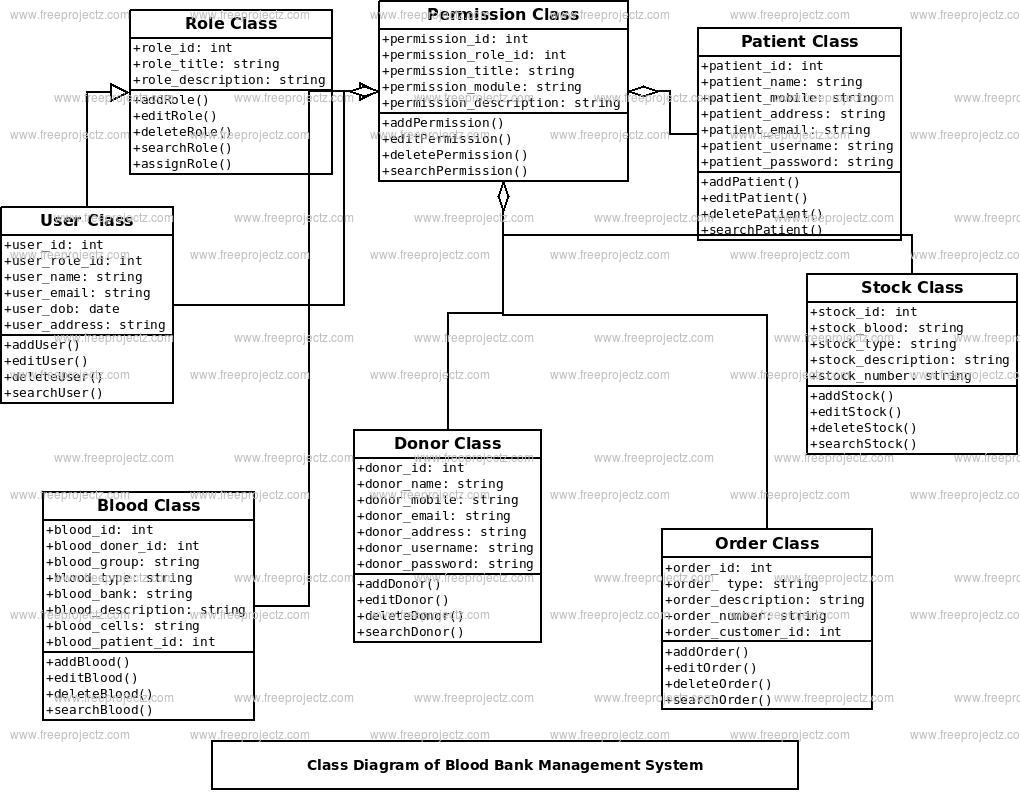
C

Admin

Recipient

Donor

Diagram

Modular diagram

Source code for implementation

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <map>

#include <iomanip>

using namespace std;

// Structure to store donor information

struct Donor {

int id;

string name;

string bloodType;

int age;

string contact;

};

// Structure to store recipient information

struct Recipient {

int id;

string name;

int age;

string bloodGroup;

string contact;

};

// Structure to manage blood stock

struct Stock {

string bloodGroup;

int quantity;

};

// Class for Blood Bank Management System

class BloodBank {

private:

vector<Donor> donors;

vector<Recipient> recipients;

vector<Stock> stock;

int nextDonorId;

bool isValidBloodType(const string &type) {

string validTypes[] = {"A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-"};

for (const auto &validType : validTypes) {

if (type == validType)

return true;

}

return false;

}

public:

BloodBank() : nextDonorId(1) {

// Initialize stock with blood groups

string bloodGroups[] = {"A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-"};

for (const auto &group : bloodGroups) {

stock.push\_back({group, 0});

}

}

// Add a new donor

void addDonor() {

Donor donor;

donor.id = nextDonorId++;

cout << "Enter donor's name: ";

cin.ignore();

getline(cin, donor.name);

do {

cout << "Enter donor's blood type (e.g., A+, B-, O+, AB-): ";

cin >> donor.bloodType;

if (!isValidBloodType(donor.bloodType)) {

cout << "Invalid blood type. Please enter again.\n";

}

} while (!isValidBloodType(donor.bloodType));

do {

cout << "Enter donor's age (must be 18 or older): ";

cin >> donor.age;

if (donor.age < 18) {

cout << "Donor age must be 18 or older.\n";

}

} while (donor.age < 18);

cout << "Enter donor's contact number: ";

cin >> donor.contact;

donors.push\_back(donor);

cout << "Donor added successfully! Assigned ID: " << donor.id << "\n";

}

// Update donor details

void updateDonor() {

int id;

cout << "Enter the ID of the donor to update: ";

cin >> id;

for (auto& donor : donors) {

if (donor.id == id) {

cout << "Updating details for Donor ID " << id << ":\n";

cout << "Enter new name (current: " << donor.name << "): ";

cin.ignore();

getline(cin, donor.name);

cout << "Enter new blood type (current: " << donor.bloodType << "): ";

string bloodType;

cin >> bloodType;

if (isValidBloodType(bloodType))

donor.bloodType = bloodType;

else

cout << "Invalid blood type. Keeping existing value.\n";

cout << "Enter new age (current: " << donor.age << "): ";

int age;

cin >> age;

if (age >= 18)

donor.age = age;

else

cout << "Invalid age. Keeping existing value.\n";

cout << "Enter new contact (current: " << donor.contact << "): ";

string contact;

cin >> contact;

donor.contact = contact;

cout << "Donor details updated successfully.\n";

return;

}

}

cout << "Donor not found.\n";

}

// Display all donors

void displayDonors() {

if (donors.empty()) {

cout << "No donors registered yet.\n";

return;

}

cout << "List of Donors:\n";

for (const auto& donor : donors) {

displayDonorDetails(donor);

}

}

// Display donor details

void displayDonorDetails(const Donor& donor) {

cout << "ID: " << donor.id << "\n"

<< "Name: " << donor.name << "\n"

<< "Blood Type: " << donor.bloodType << "\n"

<< "Age: " << donor.age << "\n"

<< "Contact: " << donor.contact << "\n"

<< "----------------------------------\n";

}

// Add a recipient

void addRecipient() {

Recipient recipient;

recipient.id = recipients.size() + 1;

cout << "Enter recipient's name: ";

cin.ignore();

getline(cin, recipient.name);

cout << "Enter recipient's age: ";

cin >> recipient.age;

do {

cout << "Enter recipient's blood group (e.g., A+, B-, O+, AB-): ";

cin >> recipient.bloodGroup;

if (!isValidBloodType(recipient.bloodGroup)) {

cout << "Invalid blood group. Please enter again.\n";

}

} while (!isValidBloodType(recipient.bloodGroup));

cout << "Enter recipient's contact number: ";

cin >> recipient.contact;

recipients.push\_back(recipient);

cout << "Recipient registered successfully! Assigned ID: " << recipient.id << "\n";

}

// Update Recipient details

void updateRecipient() {

int id;

cout << "Enter the ID of the recipient to update: ";

cin >> id;

for (auto& recipient : recipients) {

if (recipient.id == id) {

cout << "Updating details for recipient ID " << id << ":\n";

cout << "Enter new name (current: " << recipient.name << "): ";

cin.ignore();

getline(cin, recipient.name);

cout << "Enter new blood Group (current: " << recipient.bloodGroup << "): ";

string bloodGroup;

cin >> bloodGroup;

if (isValidBloodType(bloodGroup))

recipient.bloodGroup = bloodGroup;

else

cout << "Invalid blood Group. Keeping existing value.\n";

cout << "Enter new age (current: " << recipient.age << "): ";

int age;

cin >> age;

if (age >= 18)

recipient.age = age;

else

cout << "Invalid age. Keeping existing value.\n";

cout << "Enter new contact (current: " << recipient.contact << "): ";

string contact;

cin >> contact;

recipient.contact = contact;

cout <<"Recipient details updated successfully.\n";

return;

}

}

cout << "Recipient not found.\n";

}

// View all recipients

void viewRecipients() {

if (recipients.empty()) {

cout << "No recipients registered yet.\n";

return;

}

cout << "List of Recipients:\n";

for (const auto& recipient : recipients) {

displayrecipientDetails(recipient);

}

}

// Display recipient details

void displayrecipientDetails(const Recipient& recipient) {

cout << "ID: " << recipient.id << "\n"

<< "Name: " << recipient.name << "\n"

<< "Blood Type: " << recipient.bloodGroup << "\n"

<< "Age: " << recipient.age << "\n"

<< "Contact: " << recipient.contact << "\n"

<< "----------------------------------\n";

}

// Delete donor by ID

void deleteDonor() {

int id;

cout << "Enter the ID of the donor to delete: ";

cin >> id;

auto it = remove\_if(donors.begin(), donors.end(),

[&id](const Donor& donor) { return donor.id == id; });

if (it != donors.end()) {

donors.erase(it, donors.end());

cout << "Donor deleted successfully.\n";

} else {

cout << "Donor not found.\n";

}

}

// Delete Recipient by ID

void deleteRecipient() {

int id;

cout << "Enter the ID of the Recipient to delete: ";

cin >> id;

auto it = remove\_if(recipients.begin(), recipients.end(),

[&id](const Recipient& recipient) { return recipient.id == id; });

if (it != recipients.end()) {

recipients.erase(it, recipients.end());

cout << "Recipient deleted successfully.\n";

} else {

cout << "Recipient not found.\n";

}

}

// View blood stock

void viewStock() {

cout << left << setw(10) << "Blood Group" << "Quantity\n";

for (const auto &s : stock) {

cout << left << setw(10) << s.bloodGroup << s.quantity << "\n";

}

}

// Update blood stock

void updateStock() {

string bloodGroup;

int quantity;

cout << "Enter blood group to update: ";

cin >> bloodGroup;

cout << "Enter quantity to add: ";

cin >> quantity;

for (auto &s : stock) {

if (s.bloodGroup == bloodGroup) {

s.quantity += quantity;

cout << "Stock updated successfully!\n";

return;

}

}

cout << "Invalid blood group!\n";

}

void searchByBloodType(const string &bloodType) {

if (!isValidBloodType(bloodType)) {

cout << "Invalid blood type entered.\n";

return;

}

bool found = false;

for (const auto &donor : donors) {

if (donor.bloodType == bloodType) {

cout << "Donor ID: " << donor.id << ", Name: " << donor.name << "\n";

found = true;

}

}

if (!found) {

cout << "No donors found with blood type " << bloodType << ".\n";

}

}

//search recepient by blood group

void searchByBloodGroup(const string &bloodType) {

if (!isValidBloodType(bloodType)) {

cout << "Invalid blood group entered.\n";

return;

}

bool found = false;

for (const auto &recipient : recipients) {

if (recipient.bloodGroup == bloodType) {

cout << "Recipient ID: " << recipient.id << ", Name: " << recipient.name << "\n";

found = true;

}

}

if (!found) {

cout << "No recipients found with blood group " << bloodType << ".\n";

}

}

// search donar by name

void searchByName(const string &name) {

bool found = false;

for (const auto &donor : donors) {

if (donor.name == name) {

cout << "Donor ID: " << donor.id << ", Blood Type: " << donor.bloodType << "\n";

found = true;

}

}

if (!found) {

cout << "No donors found with the name " << name << ".\n";

}

}

// search recipient by name

void searchBy\_Name(const string &name) {

bool found = false;

for (const auto &recipient : recipients) {

if (recipient.name == name) {

cout << "recipient ID: " << recipient.id << ", Blood Group: " << recipient.bloodGroup << "\n";

found = true;

}

}

if (!found) {

cout << "No recipients found with the name " << name << ".\n";

}

}

// Fulfill a blood request

void fulfillRequest() {

string bloodGroup;

int quantity;

cout << "Enter blood group required: ";

cin >> bloodGroup;

cout << "Enter quantity required: ";

cin >> quantity;

for (auto &s : stock) {

if (s.bloodGroup == bloodGroup) {

if (s.quantity >= quantity) {

s.quantity -= quantity;

cout << "Request fulfilled successfully!\n";

} else {

cout << "Insufficient stock available.\n";

}

return;

}

}

cout << "Invalid blood group!\n";

}

};

// Admin Class for authentication

class Admin {

private:

string username = "admin";

string password = "admin123";

public:

bool login() {

string enteredUsername, enteredPassword;

cout << "Enter admin username: ";

cin >> enteredUsername;

cout << "Enter admin password: ";

cin >> enteredPassword;

if (enteredUsername == username && enteredPassword == password) {

cout << "Admin login successful!\n";

return true;

}

cout << "Invalid credentials. Access denied.\n";

return false;

}

};

// Main function

int main() {

BloodBank bloodBank;

Admin admin;

int choice;

do {

cout << "\n--- Blood Bank Management System ---\n";

cout << "1. Admin Login\n";

cout << "2. View All Donors\n";

cout << "3. Search Donor by Blood Type\n";

cout << "4. Search Donor by Name\n";

cout << "5. View All Recipients\n";

cout << "6. Search recepient by Blood group\n";

cout << "7. Search recepient by Name \n";

cout << "8. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

if (admin.login()) {

int adminChoice;

do {

cout << "\n--- Admin Menu ---\n";

cout << "1. Add Donor\n";

cout << "2. Add Recipient\n";

cout << "3. Delete Donor\n";

cout << "4. Delete Reciepient\n";

cout << "5. Update Donor Details\n";

cout << "6. Update Recipient Details\n";

cout << "7. view Blood Stock\n";

cout << "8. Update Blood Stock\n";

cout << "9. Fulfill Blood Request\n";

cout << "10. Logout\n";

cout << "Enter your choice: ";

cin >> adminChoice;

switch (adminChoice) {

case 1:

bloodBank.addDonor();

break;

case 2:

bloodBank.addRecipient();

break;

case 3:

bloodBank.deleteDonor();

break;

case 4:

bloodBank.deleteRecipient();

break;

case 5:

bloodBank.updateDonor();

break;

case 6:

bloodBank.updateRecipient();

break;

case 8:

bloodBank.updateStock();

break;

case 7:

bloodBank.viewStock();

break;

case 9:

bloodBank.fulfillRequest();

break;

case 10:

cout << "Admin logged out.\n";

break;

default:

cout << "Invalid choice! Try again.\n";

}

} while (adminChoice != 10);

}

break;

case 2:

bloodBank.displayDonors();

break;

case 3: {

string bloodType;

cout << "Enter blood type to search for: ";

cin >> bloodType;

bloodBank.searchByBloodType(bloodType);

break;

}

case 4: {

string name;

cout << "Enter donor's name to search for: ";

cin.ignore();

getline(cin, name);

bloodBank.searchByName(name);

break;

}

case 5:

bloodBank.viewRecipients();

break;

case 6: {

string bloodGroup;

cout << "Enter blood Group to search for: ";

cin >> bloodGroup;

bloodBank.searchByBloodGroup(bloodGroup);

break;

}

case 7: {

string name;

cout << "Enter recipients name to search for: ";

cin.ignore();

getline(cin, name);

bloodBank.searchBy\_Name(name);

break;

}

case 8:

cout << "Exiting the system. "<<"\n"<<" THANKYOU FOR VISITING Goodbye!"<<"\n"<<" SAVE HEALTH , SAVE LIFE";

break;

default:

cout << "Invalid choice! Try again.\n";

}

} while (choice != 8);

return 0;

}

Output

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

// admin login

Enter your choice: 1

Enter admin username: admin

Enter admin password: admin123

Admin login successful!

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

// adding donor information

Enter your choice: 1

Enter donor's name: lokesh

Enter donor's blood type (e.g., A+, B-, O+, AB-): B+

Enter donor's age (must be 18 or older): 20

Enter donor's contact number: 9874563210

Donor added successfully! Assigned ID: 1

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

// adding another donor information

Enter your choice: 1

Enter donor's name: hemanth

Enter donor's blood type (e.g., A+, B-, O+, AB-): A-

Enter donor's age (must be 18 or older): 19

Enter donor's contact number: 9874563210

Donor added successfully! Assigned ID: 2

// adding recipient

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 2

Enter recipient's name: upendra

Enter recipient's age: 19

Enter recipient's blood group (e.g., A+, B-, O+, AB-): O-

Enter recipient's contact number: 8745962130

Recipient registered successfully! Assigned ID: 1

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 2

Enter recipient's name: ganesh

Enter recipient's age: 18

Enter recipient's blood group (e.g., A+, B-, O+, AB-): AB+

Enter recipient's contact number: 7456981230

Recipient registered successfully! Assigned ID: 2

// deletion of donor information

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 3

Enter the ID of the donor to delete: 2

Donor deleted successfully.

// deletion of recipient information

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 4

Enter the ID of the Recipient to delete: 2

Recipient deleted successfully.

// updating donor’s information

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 5

Enter the ID of the donor to update: 1

Updating details for Donor ID 1:

Enter new name (current: lokesh): LOKESH

Enter new blood type (current: B+): B+

Enter new age (current: 20): 21

Enter new contact (current: 9874563210): 9876543210

Donor details updated successfully.

// updation of recipients information

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 6

Enter the ID of the recipient to update: 1

Updating details for recipient ID 1:

Enter new name (current: upendra): UPENDRA

Enter new blood Group (current: O-): O-

Enter new age (current: 19): 20

Enter new contact (current: 8745962130): 8745693210

Recipient details updated successfully.

// updation of blood stock

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 8

Enter blood group to update: A-

Enter quantity to add: 4

Stock updated successfully!

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 8

Enter blood group to update: B+

Enter quantity to add: 2

Stock updated successfully!

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 8

Enter blood group to update: O+

Enter quantity to add: 3

Stock updated successfully!

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 8

Enter blood group to update: AB+

Enter quantity to add: 1

Stock updated successfully!

// viewing stock information

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 7

Blood GroupQuantity

A+ 0

A- 4

B+ 2

B- 0

O+ 3

O- 0

AB+ 1

AB- 0

--- Admin Menu ---

1. Add Donor

2. Add Recipient

3. Delete Donor

4. Delete Reciepient

5. Update Donor Details

6. Update Recipient Details

7. view Blood Stock

8. Update Blood Stock

9. Fulfill Blood Request

10. Logout

Enter your choice: 10

Admin logged out.

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 2

List of Donors:

ID: 1

Name: LOKESH

Blood Type: B+

Age: 21

Contact: 9876543210

----------------------------------

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 3

Enter blood type to search for: B+

Donor ID: 1, Name: LOKESH

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 3

Enter blood type to search for: AB-

No donors found with blood type AB-.

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 4

Enter donor's name to search for: LOKESH

Donor ID: 1, Blood Type: B+

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 4

Enter donor's name to search for: KING

No donors found with the name KING.

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 5

List of Recipients:

ID: 1

Name: UPENDRA

Blood Type: O-

Age: 20

Contact: 8745693210

----------------------------------

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recipient by Name

8. Exit

Enter your choice: 6

Enter blood Group to search for: o+

Invalid blood group entered.

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 6

Enter blood Group to search for: O-

Recipient ID: 1, Name: UPENDRA

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recipient by Blood group

7. Search recipient by Name

8. Exit

Enter your choice: 7

Enter recipients name to search for: UPENDRA

recipient ID: 1, Blood Group: O-

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 7

Enter recipients name to search for: CHIRU

No recipients found with the name CHIRU.

--- Blood Bank Management System ---

1. Admin Login

2. View All Donors

3. Search Donor by Blood Type

4. Search Donor by Name

5. View All Recipients

6. Search recepient by Blood group

7. Search recepient by Name

8. Exit

Enter your choice: 8

Exiting the system.

THANKYOU FOR VISITING Goodbye!

SAVE HEALTH , SAVE LIFE

=== Code Execution Successful ===

CONCLUSION

The Blood Bank Management System efficiently manages donors, recipients, and blood stock, ensuring timely fulfillment of blood requests. With automated processes, secure admin access, and user-friendly features, the system improves service delivery, reduces errors, and bridges the gap between donors and recipients. It’s a scalable solution designed to support critical medical needs and save lives effectively.

REFERENCES

1. **"Health Informatics: Practical Guide for Healthcare and IT Professionals" by Robert E. Hoyt**: This book offers a broad understanding of healthcare IT systems, including blood bank management.
2. **"Biomedical Informatics: Computer Applications in Health Care and Biomedicine" by Edward H. Shortliffe**: Discusses the role of information technology in health care, with relevant sections on managing blood banks and medical resources.
3. **"Design and Implementation of Blood Bank Information Systems" (Journal Article)**: This academic paper provides an overview of blood bank information systems and their impact on clinical practice and hospital management.
4. **"Blood Bank Management System: A Review" (International Journal of Engineering Research and Applications)**: This article explores the design and development of BBMS and its implementation in various healthcare settings. It discusses how automation improves efficiency and reduces errors.
5. **"Design and Implementation of a Blood Bank Management System" (IEEE Xplore)**: A study that details the technical aspects of implementing a BBMS, covering system architecture and software solutions.
6. **IEEE Xplore (**[**https://ieeexplore.ieee.org/**](https://ieeexplore.ieee.org/)**)**: A comprehensive collection of papers and articles on healthcare management, including blood bank systems.
7. **SpringerLink (**[**https://link.springer.com/**](https://link.springer.com/)**)**: Hosts various research papers and journals that cover the implementation and advancements of blood bank systems.
8. **HealthIT.gov**: Provides resources for healthcare IT systems, which may include guidelines for managing blood banks through technology.
9. **"Blood Bank Management System Implementation and Evaluation" (University Thesis)**: Many university research projects focus on developing and evaluating BBMS, which you can find in university repositories.