BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Course Code : CSE100
Course Title : Software Development Project
Course Teacher : Ashfiya Jannat Keya
Lecturer, Dept. of CSE

Project Report

Group Name : ByteBreakers
Project Name : Blood Bank Management System

Team Members:

1. Md. Miftahur Rahman Swapnil (22235103183)
2. Khorshed Alam Sadhin (22235103193)
3. Asif Ali (22235103194)
4. Md. Sabbir Ahmed (22235103185)
5. Md. Jahidul Kamal Islam (22235103214)

Abstract

The Blood Bank Management System code facilitates efficient management of blood donor information, providing features for adding, updating, and deleting donor records. Users, including administrators, can easily input, track, and retrieve donor details, ensuring a streamlined process for blood donation eligibility. This system improves the overall organization and accessibility of blood donor information, contributing to the swift identification of eligible donors and enhancing the efficiency of blood donation campaigns.

Acknowledgements

We would like to express our deepest gratitude to Ashfiya Jannat Keya, our project supervisor and esteemed Lecturer in the Department of Computer Science and Engineering. Her unwavering support, insightful guidance, and continuous encouragement have been invaluable throughout the development of the Blood Bank Management System project. Under her mentorship, we have not only honed our technical skills but also gained a profound understanding of software development principles.

Special thanks go to our classmates and friends for their collaborative spirit, constructive discussions, and shared enthusiasm, which played a crucial role in shaping the project and making it a success.

Thank you all for your contributions and support.

Declaration

We hereby declare that the Project on Blood Bank Management System in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering of Bangladesh University of Business and Technology (BUBT) is our own work and that it contains no material which has been accepted for the award to the candidate(s) of any other degree or diploma. To the best of our knowledge, it contains no materials previously published or written by any other person except where due reference is made in the project.

Contents

1. Introduction

1.1 Introduction

In the landscape of healthcare, the efficient and organized management of blood resources is of paramount importance to save lives and ensure the well-being of patients. The Blood Bank Management System (BBMS) presented in this project aims to provide a systematic solution to the challenges faced by blood banks in handling donor information, ensuring blood type availability, and maintaining a secure and accessible database. This system introduces features such as donor registration, display, search, update, and deletion, along with a robust authentication system to control access levels.

1.2 Objective

The primary objectives of the Blood Bank Management System project are as follows:

1. **Donor Management**: Implement a user-friendly interface to facilitate the addition, display, and modification of donor records within the blood bank database.

2. **Blood Type Search**: Enable efficient searching of donors based on blood types to streamline the process of matching donors with specific blood requirements.

3. **Access Control**: Implement a secure sign-up and login system, distinguishing between normal users and administrators. Normal users can add donors and view donor information, while administrators have access to all features, including updating and deleting donor records.

1.3 Project Scope

The project encompasses the following key features:

Donor Management:
Add Donor: Allow blood bank staff to register new donors, capturing essential information for the database.
Display Donor: Provide a user-friendly display of donor information for quick reference.
Search by Blood Type: Facilitate efficient searching of donors based on blood types, ensuring rapid response to blood type-specific requests.
Update Donor: Enable authorized users to modify donor information as needed.

Delete Donor: Allow administrators to remove outdated or incorrect donor records, maintaining the accuracy of the database.

1.4 Our Contributions

In the development of the Blood Bank Management System, our team has played a pivotal role in the following aspects:

1. User Interface and Experience Design:
Crafting an intuitive and user-friendly interface to enhance the ease of use for both blood bank staff and administrators.

2. Access Control Implementation:
Developing a robust authentication system to differentiate between normal users and administrators, ensuring the security and integrity of the donor database.

3. Feature Implementation:
Implementing the core features of the system, including donor registration, display, search, update, and delete functionalities.

4.Testing and Quality Assurance:
Conducting rigorous testing procedures to identify and rectify potential issues, ensuring the reliability and stability of the Blood Bank Management System.

Introduction

This project report aims to provide a comprehensive overview of the Blood Bank Management System, shedding light on its development, functionality, and the significant role it plays in optimizing blood bank operations.

2. Existing Literature

2.1 Introduction

Healthcare systems worldwide heavily rely on the accessibility of safe and adequate blood supplies to meet the demands of medical treatments, surgeries, and emergencies. Blood banks serve as vital repositories, ensuring the availability of various blood types for diverse patient needs. This literature review delves into the necessity of blood banks, emphasizing their critical role in healthcare, and explores the evolving landscape of blood donation culture.

2.2 Necessity of Blood Banks

1. Medical Emergencies and Treatments:
Blood is a life-saving resource in emergency situations, traumatic injuries, and surgical procedures. The organized storage and distribution facilitated by blood banks ensure timely access to compatible blood products, significantly improving patient outcomes.

2. Diverse Patient Requirements:
The diversity of medical conditions and patient blood types necessitates a comprehensive blood banking system. Blood banks act as central hubs,

maintaining a varied inventory to meet the specific needs of individuals undergoing treatments such as chemotherapy, organ transplants, and childbirth.

3. Blood Testing and Quality Assurance:

Blood banks play a crucial role in testing and screening donated blood for infectious diseases and ensuring its quality. Rigorous testing procedures guarantee the safety of both donors and recipients, mitigating the risk of transmitting infections through blood transfusions.

2.3 The Growing Culture of Blood Donation

1. Public Awareness and Education:

Over the years, concerted efforts in public awareness campaigns and educational initiatives have contributed to the growth of a culture of voluntary blood donation. Increased awareness about the impact of blood donation on saving lives has encouraged individuals to participate actively.

2. Community Engagement and Corporate Initiatives:

Community-based blood donation drives and corporate social responsibility initiatives have emerged as effective strategies to boost blood donation rates. Partnerships between blood banks and organizations foster a sense of

community responsibility, encouraging regular and voluntary blood contributions.

3. Technology and Convenience:
Advancements in technology have streamlined the blood donation process, making it more convenient for donors. Mobile apps, online scheduling, and real-time updates on blood needs have enhanced donor engagement and participation.

2.4 Conclusion

This literature review underscores the indispensable role of blood banks in healthcare and highlights the positive shift in societal attitudes towards blood donation. As we navigate a landscape where the necessity of blood banks is ever-growing, fostering a culture of voluntary donation becomes paramount for sustaining a robust and accessible blood supply system.

3. Proposed Model

3.1 Introduction

This exploration delves into the essence of blood bank management, assessing whether the system is manual, the array of features it offers, and its user-friendliness. We aim to examine the impact of automation on operational efficiency, the diversity of features provided for donor record-keeping, and the user-friendly design facilitating seamless interaction. In this brief overview, we set the stage for a comprehensive evaluation of a blood bank management system, considering its manual or automated nature, feature richness, and user accessibility.

3.2 Hardware and Software

● Programming Language: C++

Software:

● Development Environment: Any suitable IDE (e.g., Visual Studio, Code::Blocks)

Hardware :

● 1GHz or High processor

● 512 MB RAM (minimum)

- 500 MB Hard Disk(minimum)

3.3 SS of Encountered Interfaces

The Screenshots of the interfaces we are going to face inside the Program are shown below:

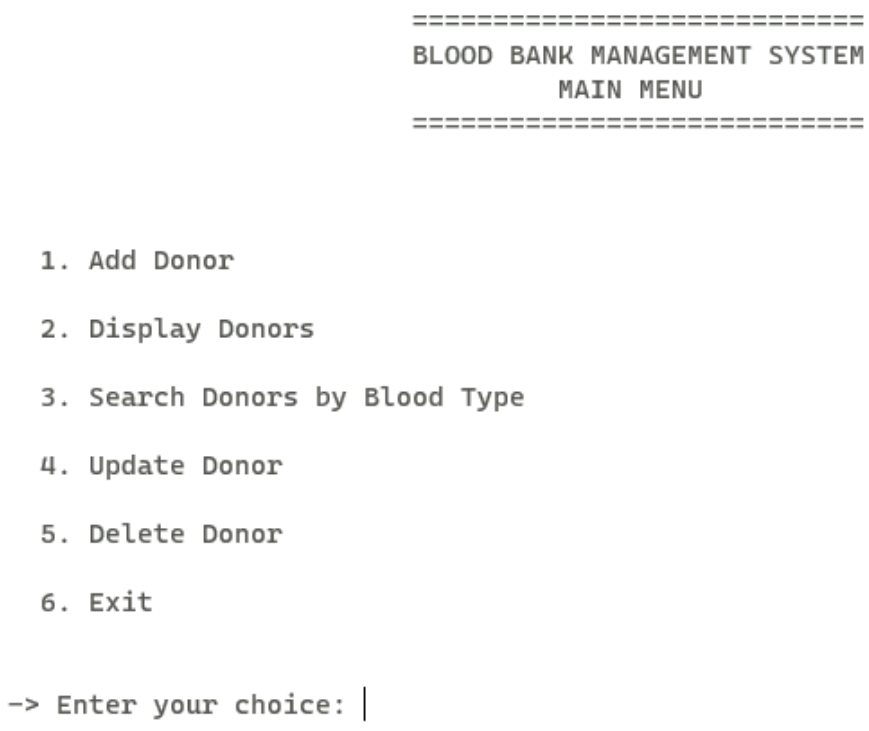

Figure 3.1 : Main Menu Interface

```
        >> Donor Information :
       ----------------------

    -> Enter Donor Name: Khorshed Alam Sadhin

    -> Enter Gender (M/F): M

    -> Enter Blood Type (A/B/AB/O +/-): B+

    -> Enter Contact Number: 015******86

    -> Enter Gmail address: khorshedsadhin81@gmail.com

    -> Enter last date of donating blood (e.g., 29-2-2023): 1-10-2022

    >> Donor added successfully! <<
```

Figure 3.2: Add Donor Interface

```
Donors loaded from donors.txt successfully!

         >> List of donors:
        --------------------

    -> Name: Khorshed Alam Sadhin
    -> Gender: M
    -> Blood Type: B+
    -> Contact Number: 015******86
    -> Gmail Address: khorshedsadhin81@gmail.com
    -> Last Donation Date: 1-10-2022
    -> Status: Eligible for Blood Donation.
```

Figure 3.3: Display Donors Interface

```
>> Enter blood type to search (A/B/AB/O +/-): B+

     >> Donors with blood type B+:
     --------------------------


-> Name: Khorshed Alam Sadhin
-> Gender: M
-> Blood Type: B+
-> Contact Number: 015******86
-> Gmail Address: khorshedsadhin81@gmail.com
-> Last Donation Date: 1-10-2022
-> Status: Eligible for Blood Donation.
```

Figure 3.4 : Search by Blood Type Interface

```
Enter the name of the donor you want to update: Khorshed Alam Sadhin
Updating donor information for Khorshed Alam Sadhin:

-> Enter new contact number: 0195618**45
-> Enter new Gmail address: sadhin@gmail.com
-> Enter new last donation date (e.g., 29-2-2023): 2-10-23

Donor information updated successfully!
```

Figure 3.5 : Update Donor Interface

```
Enter the name of the donor you want to delete: Khorshed Alam Sadhin
Donor Khorshed Alam Sadhin has been deleted from the database.


Enter the name of the donor you want to delete: Miftahur Rahman Swapnil
No donor found with the name Miftahur Rahman Swapnil.
```

Figure 3.6 : Delete Donor Interface

4. Implementation of Our System

4.1 Introduction

Explanation of Key functions :

1. Add Donor (addDonor function):

Allows the user to input information for a new donor, including name, gender, blood type, contact number, Gmail address, and the last date of blood donation.
Performs an eligibility check based on gender and the time elapsed since the last donation.
Updates the donor's eligibility status accordingly.
Appends the new donor information to the vector of donors.
Automatically saves the updated donor list to a file.

2. Display Donors (displayDonors function):

Retrieves and displays donor information stored in the vector.
Outputs relevant details such as name, gender, blood type, contact number, Gmail address, last donation date, and eligibility status.
Utilizes a user-friendly interface for easy readability.
Waits for user input before clearing the console.

3. Search Donors by Blood Type (searchDonorsByBloodType function):

Allows the user to input a specific blood type.
Searches the donor vector for donors matching the specified blood type.
Displays the details of matching donors, including eligibility status.

4. Update Donor (updateDonor function):

Prompts the user to enter the name of the donor to be updated.

Searches for the donor in the vector.

Updates the contact number, Gmail address, and last donation date for the specified donor.

Performs an eligibility check based on the updated information.

Saves the updated donor information to the file.

5. Delete Donor (deleteDonor function):

Prompts the user to enter the name of the donor to be deleted.

Searches for the donor in the vector and removes it if found.

Updates the donor list in the file to reflect the deletion.

5. A Code Walkthrough

5.1 Introduction

Here we have analysed all the results regarding to the project and also we have tried to show the source code of our program.

5.2 Result Analysis

Some screenshots of our program source code are shown below in order.

Starting With the Main Menu

```
int choice;
while (true)
{
    cout <<"\t\t\t\t\t\t\t\t\t\t\t ============================";
    cout <<"\n\t\t\t\t\t\t\t\t\t\t\t BLOOD BANK MANAGEMENT SYSTEM" << endl;
    cout<<"\t\t\t\t\t\t\t\t\t\t\t          MAIN MENU"<<endl;
    //cout<<"\t\t\t\t\t\t\t\t\t\t\t       By: ByteBreakers"<<endl;
    cout <<"\t\t\t\t\t\t\t\t\t\t\t ============================";

    cout <<endl<<endl<<endl<< "\n\t\t\t\t\t\t\t\t  1. Add Donor" << endl;
    cout << "\n\t\t\t\t\t\t\t\t  2. Display Donors" << endl;
    cout << "\n\t\t\t\t\t\t\t\t  3. Search Donors by Blood Type" << endl;
    cout << "\n\t\t\t\t\t\t\t\t  4. Update Donor" << endl;
    cout << "\n\t\t\t\t\t\t\t\t  5. Delete Donor" << endl;
    cout << "\n\t\t\t\t\t\t\t\t  6. Exit" << endl;
    cout << "\n\n\t\t\t\t\t\t\t\t-> Enter your choice: ";
    cin >> choice;
    system("cls");
```

Figure 5.1 : SS of Main Menu Code

```
//feature functions
void addDonor(vector<Donor>& donors, const string& filename)
{
    Donor newDonor;
    cout <<endl<< "\t\t      >> Donor Information :" << endl;
    cout << "\t\t     ------------------------" << endl;
    cout << "\n\n\t\t -> Enter Donor Name: ";
    cin.ignore();
    getline(cin, newDonor.name);
    cout << "\n\t\t -> Enter Gender (M/F): ";
    getline(cin, newDonor.gender);
    cout <<endl<< "\t\t -> Enter Blood Type (A/B/AB/O +/-): ";
    cin >> newDonor.bloodType;
    cin.ignore();
    cout << "\n\t\t -> Enter Contact Number: ";
    getline(cin, newDonor.contactNumber);
    cout << "\n\t\t -> Enter Gmail address: ";
    getline(cin, newDonor.gmailAddress);
    cout << "\n\t\t -> Enter last date of donating blood (e.g., 29-2-2023): ";
    newDonor.lastDonationDate.inputDate();

    int daysInBetween = getDifference(newDonor.lastDonationDate, currDate);
    if((newDonor.gender == "F" && daysInBetween >= 120) || (newDonor.gender == "M" && daysInBetween >= 90)
        newDonor.status = "Eligible for Blood Donation.";

    donors.push_back(newDonor);

    cout <<endl<<endl<< "\t\t >> Donor added successfully! <<";
    saveDonorsToFile(donors, filename); // Auto save after adding a donor
    cout << "\n" << endl;
}
```

Figure 5.2 : SS of Add Donor Feature Code

```
void displayDonors(const vector<Donor>& donors)
{
    const string filename = "donors.txt";
    if (donors.empty())
    {
        cout << "      >> No donors found. <<" << endl;
    }
    else
    {
        cout <<endl<< "Donors loaded from " << filename << " successfully!" << endl;
        cout <<endl<< "\t  >> List of donors:" << endl;
        cout << "\t ---------------------" << endl;
        for (const auto& donor : donors)
        {
            if(!donor.name.empty() && !donor.bloodType.empty() && !donor.contactNumber.empty())
            {
                cout <<endl<<endl<< "    -> Name: " << donor.name << endl
                     << "    -> Gender: " <<donor.gender<<endl
                     << "    -> Blood Type: " << donor.bloodType << endl
                     << "    -> Contact Number: " << donor.contactNumber << endl
                     << "    -> Gmail Address: " << donor.gmailAddress << endl
                     << "    -> Last Donation Date: " << donor.lastDonationDate.day<<"-"
                     <<donor.lastDonationDate.month<<"-"<<donor.lastDonationDate.year << endl
                     <<"    -> Status: " << donor.status << endl;
            }
        }
    }
    getch();
    system("cls");
}
```

Figure 5.3 : SS of Display Donors Feature Code

```
void searchDonorsByBloodType(const vector<Donor>& donors, const string& bloodType)
{
    bool found = false;
    cout<<endl << "\t  >> Donors with blood type " << bloodType << ":" << endl;
    cout << "\t  ---------------------------" << endl<<endl<<endl;
    for (const auto& donor : donors)
    {
        if (donor.bloodType == bloodType)
        {
            cout << "    -> Name: " << donor.name << endl
                 << "    -> Gender: " <<donor.gender<<endl
                 << "    -> Blood Type: " << donor.bloodType << endl
                 << "    -> Contact Number: " << donor.contactNumber << endl
                 << "    -> Gmail Address: " << donor.gmailAddress << endl
                 << "    -> Last Donation Date: " << donor.lastDonationDate.day<<"-"
                 <<donor.lastDonationDate.month<<"-"<<donor.lastDonationDate.year << endl
                 <<"    -> Status: " << donor.status <<endl<<endl<<endl;
            found = true;
        }
    }
    if (!found)
    {
        cout << "     >> No donors found with blood type " << bloodType << "." << endl;
    }
    getch();
    system("cls");
}
```

Figure 5.4 : SS of Search by Blood Type Feature Code

```
void updateDonor(vector<Donor>& donors, const string& filename)
{
    string donorName;
    cout <<endl<<endl<< "\tEnter the name of the donor you want to update: ";
    cin.ignore();
    getline(cin, donorName);

    bool donorFound = false;
    for (Donor& donor : donors)
    {
        if (donor.name == donorName)
        {
            cout << "\tUpdating donor information for " << donorName << ":" << endl<<endl;
            /*cout << "\t-> Enter new gender (M/F): ";
            cin >> donor.gender;
            cout << "\t-> Enter new blood type (A/B/AB/O +/-): ";
            cin >> donor.bloodType; */
            cout << "\t-> Enter new contact number: ";
            getline(cin, donor.contactNumber);
            cout << "\t-> Enter new Gmail address: ";
            getline(cin, donor.gmailAddress);
            cout << "\t-> Enter new last donation date (e.g., 29-2-2023): ";
            donor.lastDonationDate.inputDate();

            int daysInBetween = getDifference(donor.lastDonationDate, currDate);
            if ((donor.gender == "F" && daysInBetween >= 120) || (donor.gender == "M" && daysInBetween >= 90))
                donor.status = "Eligible for Blood Donation.";
            else
                donor.status = "Not Eligible";

            cout <<endl<< "\tDonor information updated successfully!" << endl;
            saveDonorsToFile(donors, filename); // Save updated donor information
            donorFound = true;
            break; // No need to continue searching
        }
    }

    if (!donorFound)
    {
        cout << "\tNo donor found with the name " << donorName << "." << endl;
        getch();
        system("cls");
    }
}
```

Figure 5.5 : SS of Update Donor Feature Code

```
void deleteDonor(vector<Donor>& donors, const string& filename)
{
    string donorName;
    cout <<endl<<endl<< "\tEnter the name of the donor you want to delete: ";
    cin.ignore();
    getline(cin, donorName);

    bool donorFound = false;
    for (auto it = donors.begin(); it != donors.end();)
    {
        if (it->name == donorName)
        {
            it = donors.erase(it); // Erase the donor from the vector
            donorFound = true;
        }
        else
        {
            ++it;
        }
    }

    if (donorFound)
    {
        cout << "\tDonor " << donorName << " has been deleted from the database." << endl;
        saveDonorsToFile(donors, filename); // Save the updated donor list without the deleted donor
    }
    else
    {
        cout << "\tNo donor found with the name " << donorName << "." << endl;
        getch();
        system("cls");
    }
}
```

Figure 5.6 : Delete Donor Feature Code

6. Conclusion

In conclusion, the development and implementation of the Blood Bank Management System mark a significant stride in enhancing the efficiency of donor record-keeping and streamlining blood bank operations. Throughout this project, we have successfully addressed key functionalities such as donor registration, eligibility checks, and data management.

However, it's crucial to acknowledge the limitations identified during the course of this project. These include the absence of a sophisticated user authentication system, the reliance on a console-based interface, and potential scalability concerns. These limitations serve as valuable insights for future improvements.

Looking ahead, the future aims for the Blood Bank Management System involve refining user roles, introducing a graphical user interface for improved usability, exploring data analytics capabilities, integrating with broader health systems, and considering the development of a mobile application.

The iterative nature of software development encourages ongoing enhancements and adaptations. This project lays the foundation for a more sophisticated and user-centric Blood Bank Management System, contributing to the broader goal of facilitating efficient blood donation processes and ultimately saving lives. As technology continues to advance, so too will the capabilities and impact of the Blood Bank Management System in the realm of healthcare technology.