PURBANCHAL UNIVERSITY

Gomendra Multiple College
Birtamode, Jhapa

BCA

PU Registration No.: 004-3-2-04788-2022

Project report on

"Blood Bank Management System"

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF COMPUTER APPLICATION

Submitted by

Sanam Acharya, Kewal Dahal, Sajak Basnet

Submitted to

Jhalnath Chapagain

Year/Semester: I/II

ACKNOWLEDGEMENT

According to the course of study of BCA Second semester determined by P.U. A computer project is to be carried out for the partial fulfilment of the requirement for the bachelor's in computer application. Therefore, as students of the course we will develop a small **COMMAND LINE APPLICATION** "Blood Bank Management System".

We feel glad for getting such an opportunity to accomplish the BCA. This second semester project and feeling the experience of the team environment and team spirit. This gave us insight knowledge about the practical aspect of the various stages and the procedures of the software developments projects. We are very thankful to **Gomendra Multiple College** for facilitating the completion of the entire project work.

We are grateful because we managed to complete our project within the time given by our principal, Mr. Rupak Khanal. We also thank our mentor/teacher Mr. Jhalnath Chapagain for the guidance and encouragement in finishing this project and teaching us this course and special thanks to our group members for co-operation and effort for this project.

Table of Contents

Introduction to Project

In today's healthcare landscape, efficient management of blood donor information is crucial for ensuring the availability of safe and adequate blood supplies. The "Blood Bank Management System" represents a pioneering effort to modernize and streamline the process of managing donor records and facilitating blood donation procedures.

This innovative system is designed to revolutionize how blood banks handle donor data, leveraging advanced technology to enhance efficiency, transparency, and accessibility throughout the blood donation process. With a user-centric approach and robust security measures, the Blood Bank Management System aims to optimize blood bank operations, ultimately saving lives and supporting healthcare initiatives in communities worldwide.

Key Features:

1. **Streamlined User Experience:** The system offers a user-friendly interface, simplifying the process of registering donors, managing donor records, and scheduling donation appointments.
2. **Security and Confidentiality:** Security is paramount. The system implements stringent security measures to protect donor information and ensure compliance with privacy regulations, safeguarding the confidentiality of donor data.
3. **Real-time Updates:** Stay informed in real-time. Blood bank staff can monitor blood inventory levels, track donation appointments, and receive alerts for critical blood supply needs, enabling timely responses to emergencies.
4. **Scalability:** The system is designed to scale seamlessly, accommodating blood banks of varying sizes and catering to the needs of both local community blood drives and large-scale donation events.
5. **Customization:** Blood banks have the flexibility to customize the system to align with their specific workflows, donation protocols, and branding requirements, ensuring seamless integration into existing operations.

Through its comprehensive set of features and functionalities, the Blood Bank Management System aims to elevate the efficiency, transparency, and effectiveness of blood donation processes, ultimately contributing to improved healthcare outcomes and enhanced community well-being.

Objectives

Project Overview:

The primary objective of the Blood Bank Management System is to create a robust and user-friendly platform that facilitates efficient management of blood donor information and blood donation processes.

Objectives:

1. **Enhance User Experience:** Develop an intuitive and easy-to-use interface for donors and blood bank staff to seamlessly navigate through donor registration, search, and deletion processes.
2. **Improve Security:** Implement stringent security protocols to safeguard donor data and ensure compliance with privacy regulations, preventing unauthorized access or data breaches.
3. **Enable Real-Time Updates:** Enable real-time updates on blood inventory levels, donor eligibility status, and donation appointments to facilitate prompt responses to blood supply demands and emergencies.
4. **Support Scalability:** Design the system to accommodate the needs of blood banks of varying sizes and complexities, with the flexibility to scale up as the organization grows or diversifies its services.
5. **Customization for Blood Banks:** Allow customization of the system to align with the specific workflows, policies, and procedures of individual blood banks, ensuring seamless integration into existing operations.
6. **Establish an Audit Trail:** Establish a comprehensive audit trail to track all donor interactions and system activities, ensuring transparency, accountability, and compliance with regulatory requirements.
7. **Leverage Data Insights:** Utilize data analytics tools to derive actionable insights from donor demographics, donation trends, and inventory management metrics, empowering blood banks to make informed decisions and optimize resource allocation.
8. **Ensure Accessibility:** Ensure that the system is accessible to all stakeholders, including donors, healthcare professionals, and administrative

staff, regardless of their technical expertise or physical abilities, through inclusive design and user-friendly interfaces.

9. **Streamline Blood Bank Operations:** Streamline administrative tasks such as donor registration, blood inventory tracking, and communication management to enhance operational efficiency and minimize administrative burden.
10. **Enhance Blood Donation Efforts:** Contribute to the promotion of regular blood donation and raise public awareness about the critical role of blood donation in saving lives and supporting healthcare services, ultimately improving community health outcomes.

Importance of Blood Bank Management System

1. **Efficient Doner Management:** This system streamlines the process of managing blood donor information by allowing the easy addition, searching, and deletion of donor records. This efficiency is crucial for ensuring that blood banks can quickly access and update donor information as needed.

2. **Enhanced Accessibility:** By providing a centralized database, the system ensures that donor information is easily accessible to healthcare providers. This accessibility can save valuable time during emergencies when locating a suitable donor is critical.

3. **Improved Accuracy:** The system minimizes human errors by automating the management of donor records. Accurate and up-to-date information about donors' blood types and availability is essential for successful blood transfusions.

4. **Resource Optimization:** With the ability to search and categorize donors effectively, blood banks can optimize their resources. They can ensure that blood supplies are managed efficiently, reducing wastage and ensuring that the right type of blood is available when needed.

5. **Data Security and Privacy:** The system ensures that donor information is stored securely, and access is restricted to authorized personnel. This is vital for maintaining the privacy and confidentiality of donors' personal information.

6. **Enhanced Communication:** The system can facilitate better communication between blood banks and donors. Automated notifications and reminders can be sent to donors for upcoming blood donation drives, thereby improving donor retention and participation rates.

7. **Support for Emergency Situations:** In case of emergencies, the system can quickly identify and contact suitable donors based on their blood type and location. This rapid response capability can be lifesaving in critical situations.

8. **Comprehensive Reporting:** The system can generate detailed reports on donor demographics, blood type distributions, and donation frequency. These reports provide valuable insights that can help in planning and improving blood donation strategies.

9. **Promoting Voluntary Donations:** By maintaining a user-friendly and efficient system, blood banks can encourage more individuals to become regular donors. The ease of registration and the assurance of a well-managed system can boost public trust and participation.

System Requirements

Hardware Requirements

To ensure optimal performance, the following hardware specifications are recommended:

- PC with an AMD or Intel processor clocked at 2.1 GHz or above.
- PC equipped with a minimum of 4 GB RAM or higher for efficient data processing.
- Any type of monitor and a standard keyboard for seamless data input and output.

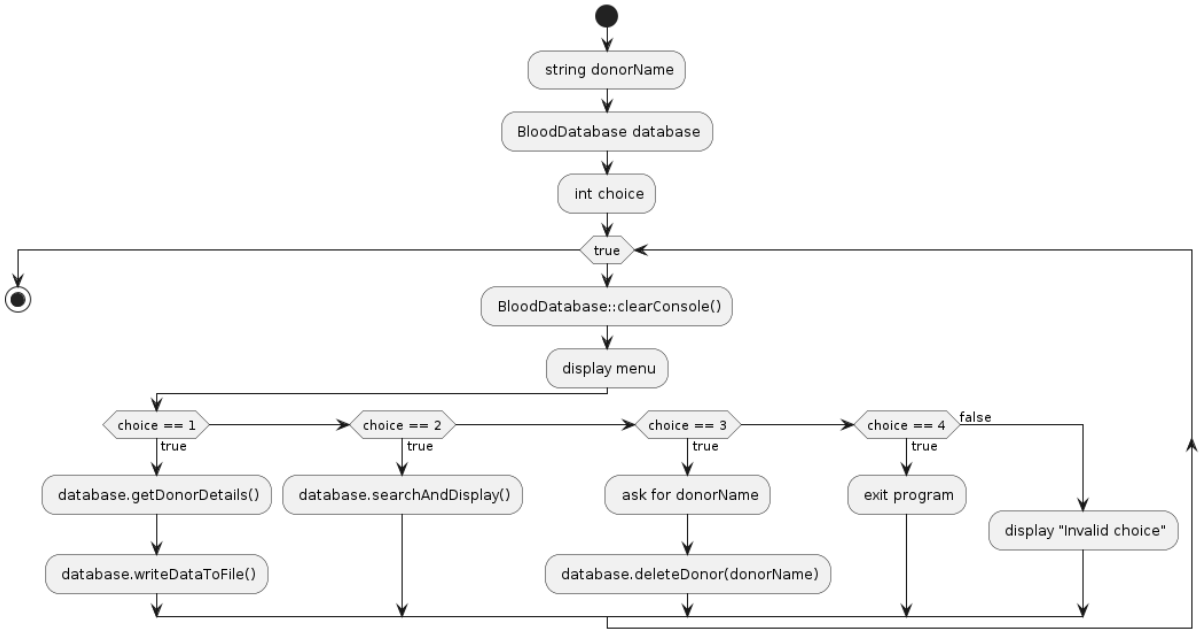
These specifications aim to provide a comfortable and efficient working environment for users, facilitating smoother operations and enhanced productivity.

Software Requirements

To effectively utilize this program and ensure compatibility, the following software prerequisites are recommended:

- **Operating System:** Compatible with Windows, macOS, or Linux platforms.
- **C++ Compiler:** An installed C++ compiler such as GCC (GNU Compiler Collection) for Linux and macOS, or Visual C++ for Windows.
- **Text Editor:** Any standard text editor like Notepad++, Visual Studio Code, or Sublime Text for writing and editing C++ source code files.
- **Terminal or Command Prompt:** A terminal or command prompt interface to compile and execute the C++ program.
- **File Explorer:** Necessary for navigating the file system and accessing input and output files.
- **Optional:** Version control software like Git for managing source code revisions, though not mandatory for basic functionality.

Flowchart


string donorName
BloodDatabase database
int choice
true
BloodDatabase::clearConsole()
display menu
choice == 1
true
database.getDonorDetails()
database.writeDataToFile()
choice == 2
true
database.searchAndDisplay()
choice == 3
true
ask for donorName
database.deleteDonor(donorName)
choice == 4
true
exit program
false
display "Invalid choice"

Source Code

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>
#include <limits>
#include <stdexcept>
#include <cctype>
#include <algorithm>

using namespace std;

class Donor {
public:
int donorId, district, number;
string name, address, bloodType;

void donorDetails() const {
cout << "Donor Name: " << name << endl;
cout << "Donor District: " << district << endl;
cout << "Donor Blood Type: " << bloodType << endl;
}

static Donor parseLine(const string& line) {
Donor d;
stringstream ss(line);
string token;

getline(ss, token, ','); d.donorId = stoi(token);
getline(ss, token, ','); d.name = token;
getline(ss, token, ','); d.address = token;
getline(ss, token, ','); d.district = stoi(token);
getline(ss, token, ','); d.bloodType = token;
getline(ss, token, ','); d.number = stoi(token);

return d;
}
};
```

```
class BloodDatabase {
private:
const string fileName = "data.txt";
vector<Donor> donors;

static void displayProvinces() {
cout << "Choose the province:\n";
cout << "1. Koshi\n";
cout << "2. Madhesh\n";
cout << "3. Bagmati\n";
cout << "4. Gandaki\n";
cout << "5. Lumbini\n";
cout << "6. Karnali\n";
cout << "7. Sudurpashchim\n";
}

public:
static void clearConsole() {
#ifdef _WIN32
system("cls");
#else
system("clear");
#endif
}

static void waitForKeyPress() {
cout << "Press any key to continue...";
cin.ignore();
cin.get();
}

static int getValidatedInput(const string& prompt) {
int value;
string input;
while (true) {
cout << prompt;
getline(cin, input);
try {
if (!all_of(input.begin(), input.end(), ::isdigit)) {
throw invalid_argument("Input contains non-numeric characters");
}
value = stoi(input);
```

```
break; // if conversion is successful, exit loop
} catch (const invalid_argument& e) {
cout << "Invalid input: " << e.what() << ". Please enter a valid number." << endl;
} catch (const out_of_range&) {
cout << "Input out of range. Please enter a valid number." << endl;
}
}
return value;
}

void getDonorDetails() {
clearConsole();
cout << "Enter donor details\n";

Donor newDonor;
newDonor.donorId = getValidatedInput("Id: ");
cout << "Name: ";
getline(cin, newDonor.name);
cout << "Address: ";
getline(cin, newDonor.address);
displayProvinces();
newDonor.district = getValidatedInput("Province (enter the corresponding number): ");
cout << "Blood Type: ";
getline(cin, newDonor.bloodType);
newDonor.number = getValidatedInput("Number: ");

donors.push_back(newDonor);
}

void writeDataToFile() {
ofstream outfile(fileName, ios::app);

if (!outfile) {
cout << "Error opening file for writing." << endl;
return;
}

Donor newDonor = donors.back();
outfile << newDonor.donorId << "," << newDonor.name << "," << newDonor.address << "," <<
newDonor.district << "," << newDonor.bloodType << "," << newDonor.number << endl;

outfile.close();
```

```
}

void searchAndDisplay() const {
clearConsole();
displayProvinces();
int provinceName = getValidatedInput("Enter the province number: ");

ifstream inFile(fileName);

if (!inFile) {
cout << "Error opening file for reading." << endl;
return;
}

vector<Donor> donors;
string line;
bool found = false;

while (getline(inFile, line)) {
Donor d = Donor::parseLine(line);
if (d.district == provinceName) {
donors.push_back(d);
found = true;
}
}

if (!found) {
cout << "No people found from province " << provinceName << endl;
} else {
cout << "People from province " << provinceName << ":" << endl;
for (const auto& d : donors) {
cout << "Name: " << d.name << endl;
cout << "Address: " << d.address << endl;
cout << "Province: " << d.district << endl;
cout << "Blood Type: " << d.bloodType << endl;
cout << "Mobile Number: " << d.number << endl;
cout << endl;
}
}

inFile.close();
waitForKeyPress();
```

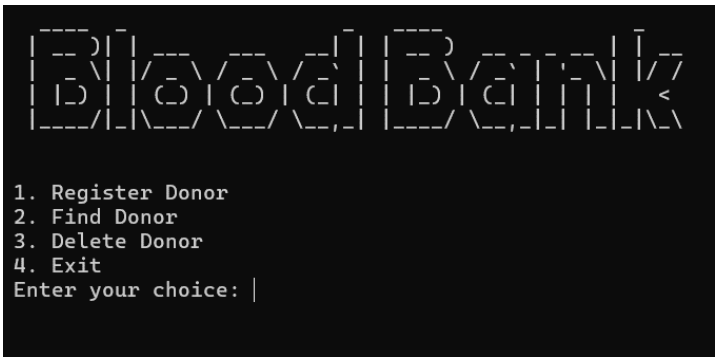
```
}

void deleteDonor(const string& donorName) {
ifstream inFile(fileName);
ofstream tempFile("temp.txt");

if (!inFile) {
cerr << "Error opening file " << fileName << endl;
return;
}

if (!tempFile) {
cerr << "Error creating temporary file" << endl;
return;
}

string line;
bool found = false;

while (getline(inFile, line)) {
Donor d = Donor::parseLine(line);
if (d.name == donorName) {
found = true;
cout << "Name: " << d.name << endl;
cout << "Address: " << d.address << endl;
cout << "Blood Type: " << d.bloodType << endl;
cout << "Mobile Number: " << d.number << endl;
cout << endl;
cout << "Are you sure you want to delete donor? [y/n]: ";
char sureChoice;
cin >> sureChoice;
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // discard any extra input

if (sureChoice == 'y' || sureChoice == 'Y') {
continue;
}
}

tempFile << d.donorId << "," << d.name << "," << d.address << "," << d.district << "," << d.bloodType
<< "," << d.number << endl;
}
```

```
inFile.close();
tempFile.close();

if (remove(fileName.c_str()) != 0) {
cerr << "Error removing original file" << endl;
return;
}

if (rename("temp.txt", fileName.c_str()) != 0) {
cerr << "Error renaming temporary file" << endl;
return;
}

if (!found) {
cout << "No donor found with the name " << donorName << endl;
}
}
};

int main() {
string donorName;
BloodDatabase database;
int choice;

while (true) {
BloodDatabase::clearConsole();
cout <<
" ____ _ _ ____ _ \n"
" | __ )| | ___ ___ __| | | __ ) __ _ _ __ | | __ \n"
" | _ \\| |/ _ \\ / _ \\ / _` | | _ \\ / _` | '_ \\| |/ / \n"
" | |_) | | (_) | (_) | (_| | | |_) | (_| | | | | < \n"
" |____/|_|\\___/ \\___/ \\__,_| |____/ \\__,_|_| |_|_|\\_\\ \n"
" \n\n";
cout << "1. Register Donor\n2. Find Donor\n3. Delete Donor\n4. Exit\nEnter your choice: ";
choice = BloodDatabase::getValidatedInput("");

switch (choice) {
case 1:
database.getDonorDetails();
database.writeDataToFile();
break;
case 2:
```

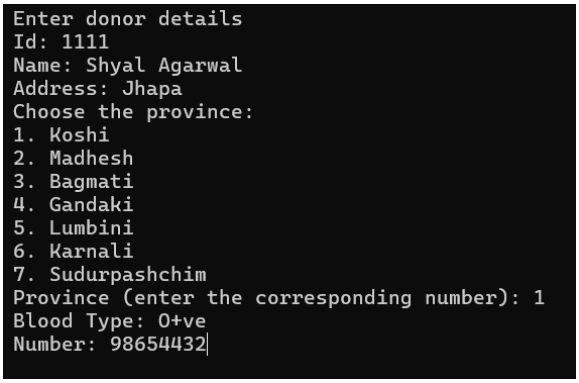
```
database.searchAndDisplay();
break;
case 3:
cout << "Enter the Name of Donor: ";
getline(cin, donorName);
database.deleteDonor(donorName);
break;
case 4:
exit(0);
default:
cout << "Invalid choice\n";
}
}
return 0;
}
```

Output Overview Windows

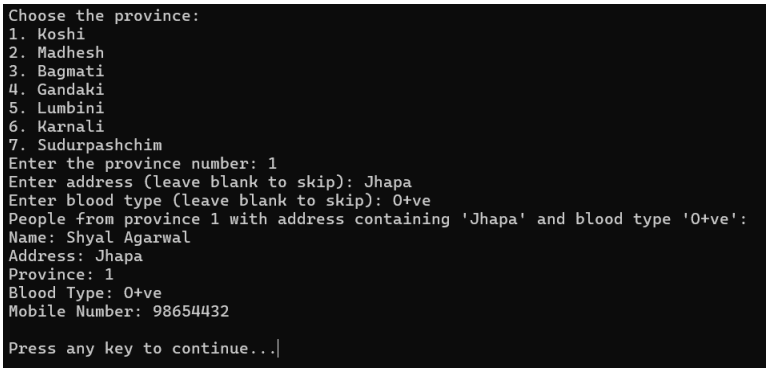
Welcome Menu



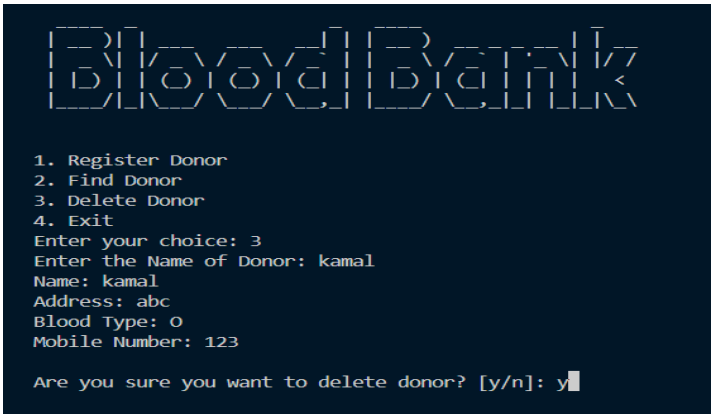
Registration Process



Finding Doner



Delete Doner



Future Use and Implementation

Enhanced User Interface: Implement a graphical user interface (GUI) using frameworks like Qt or wxWidgets to provide a more visually appealing and intuitive user experience.

Database Integration: Integrate the application with a robust database management system (DBMS) like MySQL or PostgreSQL to improve data storage, retrieval, and management capabilities.

Multi-Platform Support: Extend the application's compatibility to multiple platforms (Windows, macOS, Linux) by leveraging cross-platform development tools such as Qt or Electron.

Online Blood Donation Portal: Develop a web-based portal to allow donors to register online, schedule donation appointments, and access their donation history. This portal can also serve as a platform for educational resources and community engagement.

Mobile Application: Create a mobile application for iOS and Android devices to enable donors to conveniently register, locate nearby blood drives, and receive notifications about urgent blood needs.

Blood Inventory Management: Enhance the system to include features for tracking blood inventory levels, expiration dates, and transfusion histories, providing blood banks with real-time visibility and control over their inventory.

Donor Relationship Management (DRM): Implement DRM functionalities to maintain ongoing communication with donors, including sending reminders for upcoming donation appointments, expressing gratitude for donations, and providing updates on the impact of their contributions.

Data Analytics and Reporting: Integrate analytics tools to analyze donor demographics, donation trends, and campaign effectiveness, empowering blood banks to make data-driven decisions and optimize donation strategies.

Integration with Healthcare Systems: Establish interoperability with electronic health record (EHR) systems used by hospitals and healthcare providers to streamline the process of matching blood donors with patients in need of transfusions.

Compliance and Security Enhancements: Stay abreast of regulatory requirements and implement additional security measures to protect donor privacy and ensure compliance with data protection laws such as HIPAA (Health Insurance Portability and Accountability Act).

Community Engagement and Outreach: Utilize social media platforms, email newsletters, and community events to raise awareness about the importance of blood donation and encourage participation in donation drives.

Continuous Improvement and Feedback: Solicit feedback from users, blood bank staff, and healthcare professionals to identify areas for improvement and prioritize future development efforts to meet evolving needs and requirements.

Conclusion

The "Blood Bank Management System" project is a significant step toward enhancing the efficiency of blood donor information management. Designed as a command-line application, it focuses on core functionalities such as registering, searching, and deleting donor records, streamlining blood bank operations.

The system prioritizes user-friendliness and security, ensuring that users with basic computer skills can navigate it while keeping donor information confidential. The project followed a structured approach with stages like research, proposal drafting, coding, and testing, all completed within the allocated timeframe.

Key features include donor registration, search functionality, and deletion capabilities, addressing essential blood bank management needs. The hardware and software requirements were chosen to ensure a smooth user experience. The Gantt chart effectively guided the project schedule.

In conclusion, this project provided valuable insights into software development and teamwork. The "Blood Bank Management System" is a practical tool for blood banks, contributing to more efficient and accurate management of donor information.

Reference

- https://dev.to/tbhaxor/a-comprehensive-guide-to-file-handling-in-c-2p1b
- https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/
- https://stackoverflow.com/questions/1377078/problem-opening-file-c
- https://chenweixiang.github.io/docs/The_C++_Programming_Language_4th_Edition_Bjarne_Stroustrup.pdf