OOP PROJECT

BLOOD BANK MANAGEMENT SYSTEM

COURSE INSTRUCTOR: MISS SAMIA

Group Members:

Sajeel Tariq (CTAI-028)

Faraz Ahmed (CTAI-029)

Faisal Shahid (CTAI-011)

Hunaiza Khan (CTAI-007)

Chapter 1

Introduction:

1.1 Purpose

Blood Bank System will manage the reservation, transport and storage of blood required for different purposes such as in hospital or medical testing for research. With increasing cases of diseases causing blood deficiency, this will keep a track of no. of patients that require certain blood groups and the reserves that we already have. It will help determine if we have blood reserves in excess or if we have shortage. It manages the transport of the reserves and also the different hospitals to which it is being supplied, and in case of emergency calls blood bank branch nearest to hospital.

1.2 Project's Objective

The Blood Bank System enables real-time inventory tracking, transportation and logistics management, emergency call system handling, and analytics and reporting. It manages blood reserves, transportation, and emergency calls, ensuring timely availability and coordination between blood banks, hospitals, and medical facilities. This system addresses blood deficiencies, optimizes resource allocation, and contributes to enhanced healthcare delivery. By implementing the system, organizations can efficiently manage blood reserves, improve coordination, and address deficiencies, ultimately contributing to better healthcare delivery.

The development of the Blood Bank System's main goal is to efficiently manage the ordering, delivery, and storage of blood to meet a variety of demands, including hospital treatments and medical testing for research.

The method is essential for keeping track of the number of patients who require particular blood groups and managing the available reserves given the increased incidences of diseases that cause blood shortages. It is a crucial tool for detecting whether there are too many or not enough blood reserves.

Chapter 2

Product Features:

2.1 Log in/Registration

The Blood Bank Management System implements a robust password encryption and decryption system to ensure the security of user accounts. The encryption and decryption techniques are described as global functions. This one-way encryption ensures that even if the database is compromised, the original passwords cannot be easily obtained. During the authentication process, the user's inputted password is hashed and compared with the stored hash for verification. This encryption-decryption system helps safeguard sensitive user information and maintain the integrity of the Blood Bank Management System

2.2 Admin

2.2.1 Search, Delete and View Records of Users

The Admin class is derived from the User class in the Blood Bank Management System. As the administrator, it holds the authority to grant access to the Menu class, which provides various options for managing the system.

One of the main functionalities of the Menu class is to allow the admin to search for specific records within the database. This feature enables the administrator to locate specific users or blood inventory information efficiently. Additionally, the Menu class allows the admin to delete records when necessary. This functionality is crucial for removing outdated or irrelevant data from the system, ensuring the accuracy and relevance of the stored information.

2.2.2 View Blood Inventory

The class facilitates access to the blood inventory, allowing the admin to view the available blood types, quantities, and other related information. This provides the admin with a real-time snapshot of the blood stock, aiding in efficient management and coordination of blood supply.

2.3 Citizen

The citizen class is designed to store and manage information related to a citizen, including their personal details and health metrics.

The registration method allows a citizen to register by entering a unique username and password. The method performs verification checks to ensure that the chosen username and password meet certain criteria.

2.3.1 Receiver

The receiver class extends the citizen class. It has two functions.

The function blood packet takes the number of blood packets required as input. The range here is set to be 1-5. Second function last received will check the last date of the person who received the blood from receiver file, If 7 days have been passed then the user can receive more blood.

2.3.2 Donor

The Donor class extends the citizen class and provides straightforward functions to validate a donor's health profile based on specific criteria and determine their eligibility for blood donation based on the last donation date.

CheckDonorHealthProfile: This function verifies the health profile of a donor by evaluating specific criteria such as gender, hemoglobin level, weight, pulse rate, age, and diastolic blood pressure. If the donor meets all the criteria, the function returns true; otherwise, it returns false.

checkLastDonate: This function checks the last donation date of a donor. It calculates the time difference between the current date and the last donation date. If the time difference is more than 56 days, the function indicates that the donor is eligible for donation. Otherwise, displays a waiting period message (58 days) indicating the donor cannot donate yet.

2.4 Hospital

The hospital class extends the user class and provides functionality for hospitals to authenticate themselves, set their location, and order blood packets. It manages blood reserves and handles exceptions for invalid blood packet quantities.

The setLocation function sets the geographical location of the hospital based on its name. It assigns latitude and longitude values to the corresponding hospital name and sets the EmergencyStatus attribute to false.

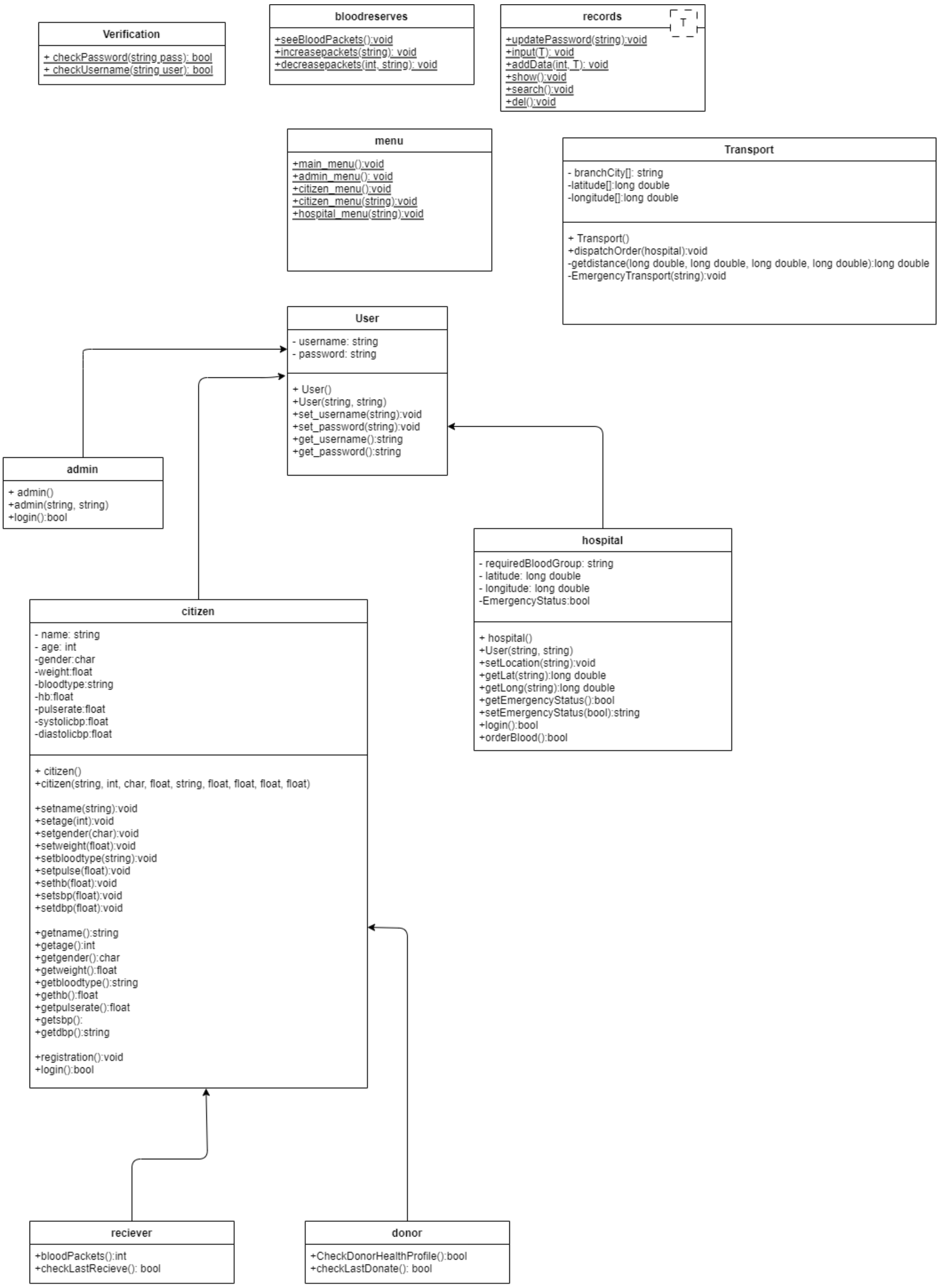
2.4.1 Request Blood Packets

The requestBlood function enables hospitals to place orders for specific blood types and quantities. It prompts the user to enter the desired blood group and the number of packets required. If the packet quantity is within the valid range of 1 to 10, the function decreases the corresponding number of packets from the blood reserves. If the packet quantity is invalid, it throws a BloodPacketException.

Chapter 3

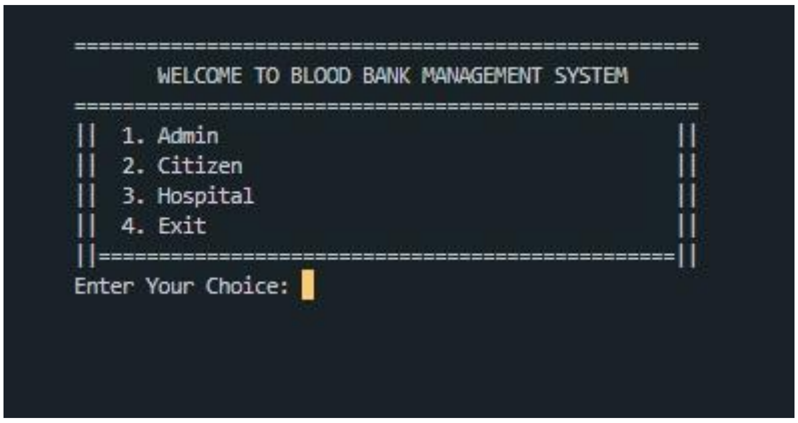
Design and Analysis:

3. 1 Class Diagram



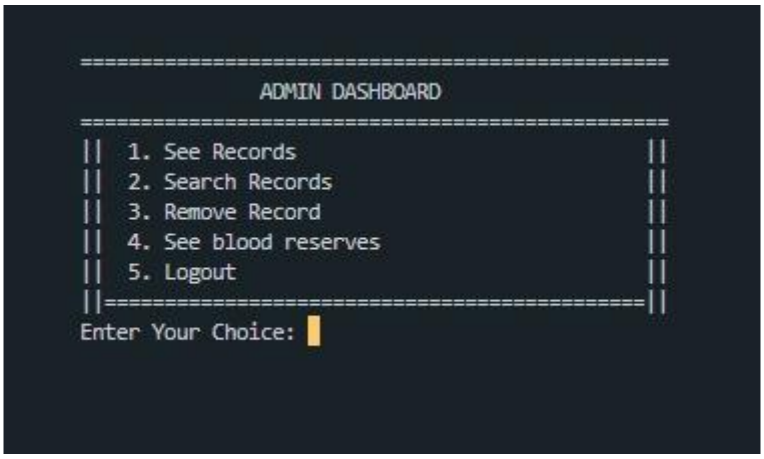
3.2 Outputs

3.2.1 Menu



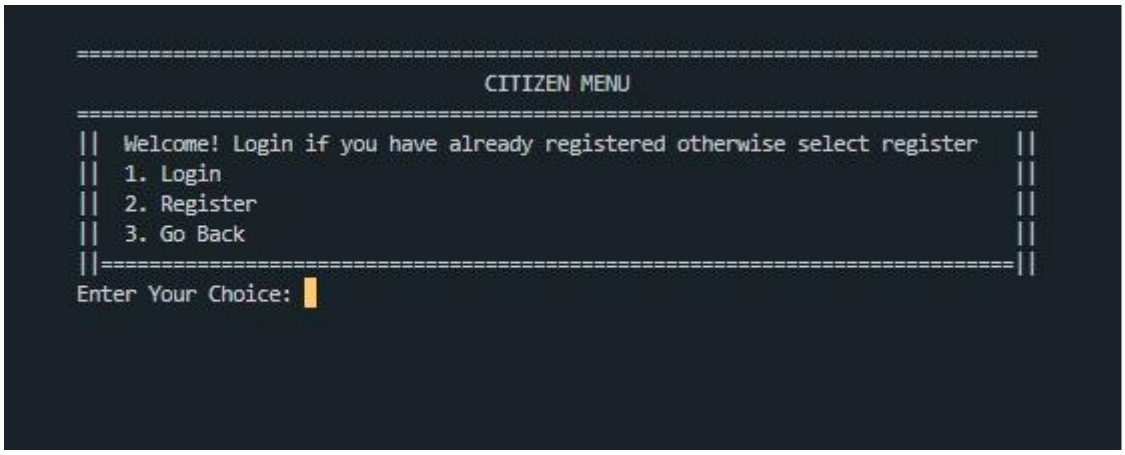
```
====================================================
     WELCOME TO BLOOD BANK MANAGEMENT SYSTEM
====================================================
||  1. Admin                                       ||
||  2. Citizen                                     ||
||  3. Hospital                                    ||
||  4. Exit                                        ||
||================================================||
Enter Your Choice:
```

3.2.2 Admin Menu



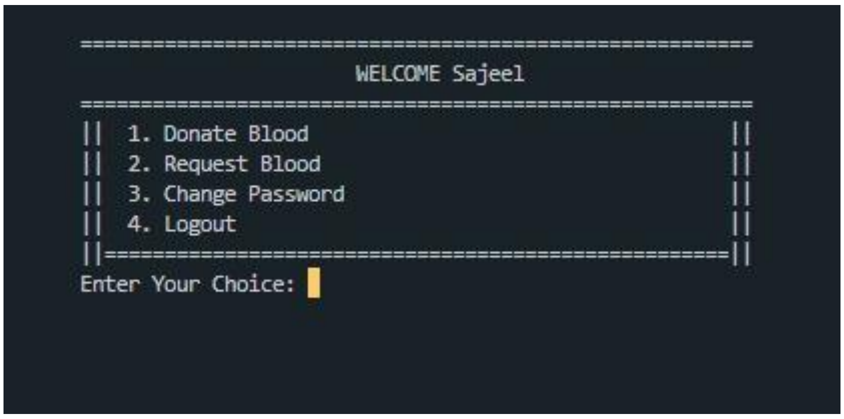
```
=================================================
               ADMIN DASHBOARD
=================================================
||  1. See Records                             ||
||  2. Search Records                          ||
||  3. Remove Record                           ||
||  4. See blood reserves                      ||
||  5. Logout                                  ||
||============================================||
Enter Your Choice:
```

3.2.3 Citizen Menu



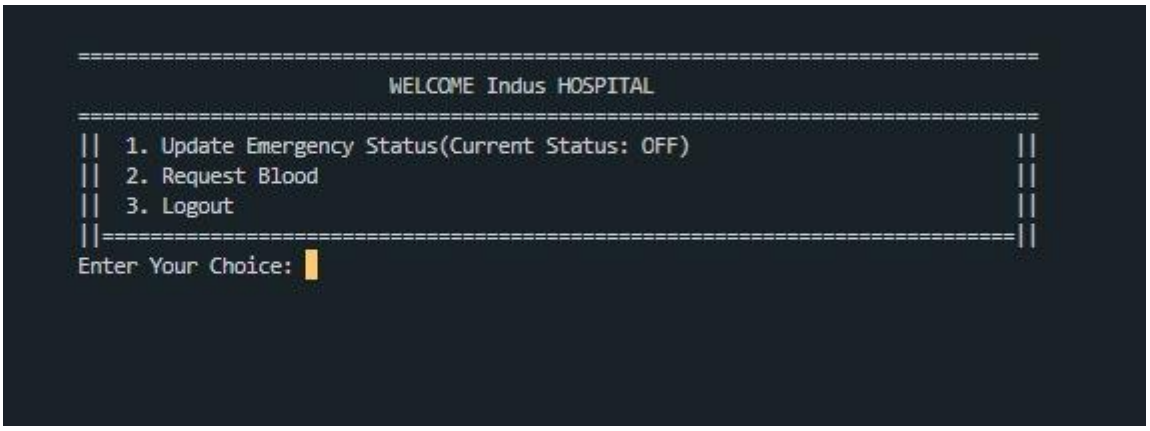
```
================================================================================
                                  CITIZEN MENU
================================================================================
||  Welcome! Login if you have already registered otherwise select register   ||
||  1. Login                                                                  ||
||  2. Register                                                               ||
||  3. Go Back                                                                ||
||===========================================================================||
Enter Your Choice:
```

3.2.3.1 Receiver & Donor Menu



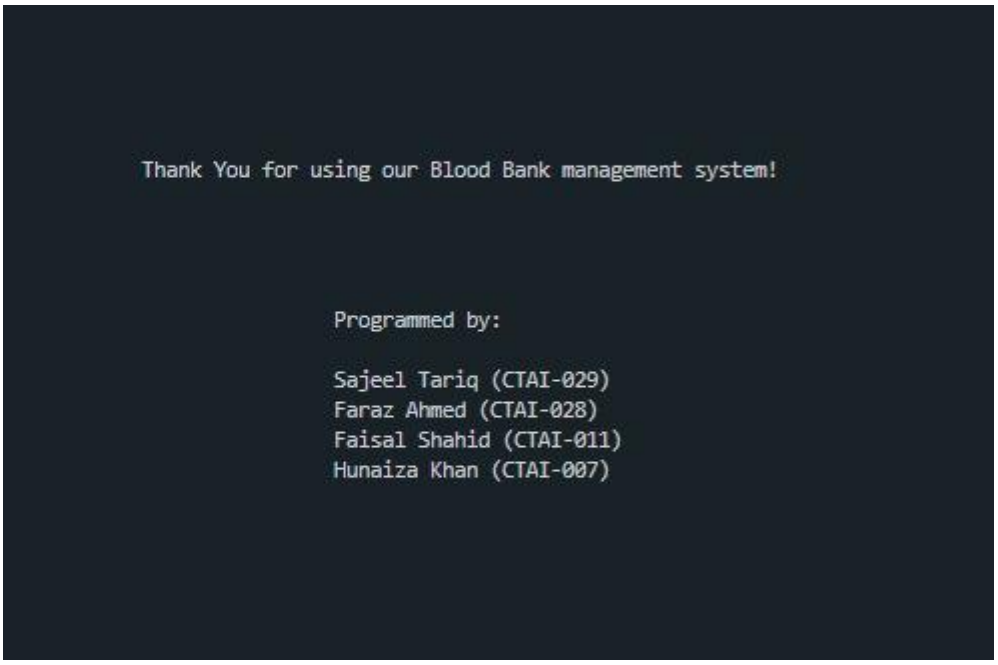
```
========================================================
                       WELCOME Sajeel
========================================================
||  1. Donate Blood                                   ||
||  2. Request Blood                                  ||
||  3. Change Password                                ||
||  4. Logout                                         ||
||====================================================||
Enter Your Choice:
```

3.2.4 Hospital Menu



```
================================================================================
                          WELCOME Indus HOSPITAL
================================================================================
||  1. Update Emergency Status(Current Status: OFF)                           ||
||  2. Request Blood                                                          ||
||  3. Logout                                                                 ||
||============================================================================||
Enter Your Choice:
```

3.2.6 End page



```
Thank You for using our Blood Bank management system!


                    Programmed by:

                    Sajeel Tariq (CTAI-029)
                    Faraz Ahmed (CTAI-028)
                    Faisal Shahid (CTAI-011)
                    Hunaiza Khan (CTAI-007)
```

3.3 Code

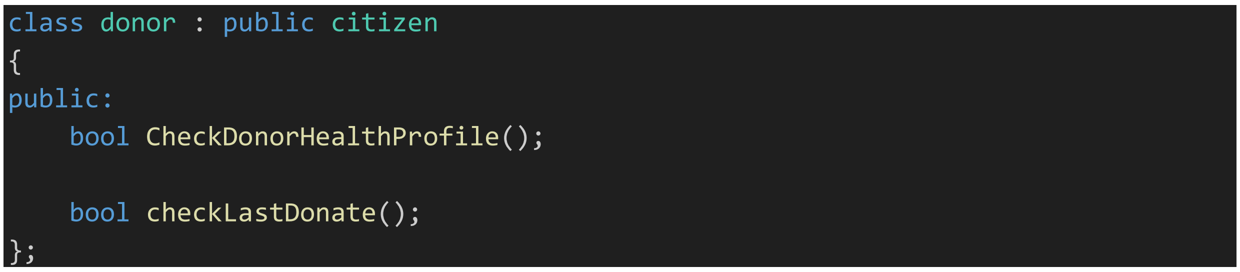
3.3.1 Class User

```
class user
{
private:
    string username;
    string password;

public:
    user() : username(""), password("") {}
    user(string un, string pass) : username(un), password(pass) {}

    void set_username(string un) { username = un; }
    void set_password(string pass) { password = pass; }

    string get_username() { return username; }
    string get_password() { return password; }
};
```

3.3.2 Class Citizen

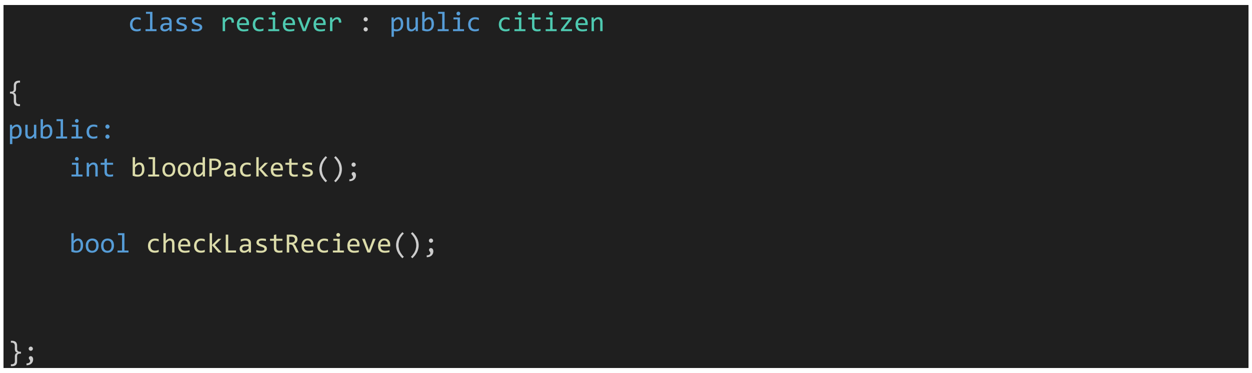
```
        class citizen : public user

{
private:
    string name;
    int age;
    char gender;
    float weight;
    string bloodtype;
    float hb;
    float pulse;
    float systolicbp;
    float diastolicbp;

public:
    citizen() : name(" "), age(0), gender(' '), weight(0.0), bloodtype(" "),
hb(0.0), pulse(0), systolicbp(0.0), diastolicbp(0.0) {}
    citizen(string n, int a, char g, float w, string bt, float hb, float pr,
float sbp, float dbp) : name(n), age(a), gender(g), weight(w), bloodtype(bt),
hb(hb), pulse(pr), systolicbp(sbp), diastolicbp(dbp) {}

    void setname(string n) { name = n; }
    void setage(int age);
    void setgender(char gender);
    void setweight(float weight);
    void setpulse(int pulse);
    void sethb(float hb);
    void setsbp(float SystolicBP);
    void setdbp(float DiastolicBP);
    void setbloodtype(string bloodgroup);


    string getname() { return name; }
    int getage() { return age; }
    char getgender() { return gender; }
    float getweight() { return weight; }
    string getbloodtype() { return bloodtype; }
    float gethb() { return hb; }
    float getpulserate() { return pulse; }
    float getsbp() { return systolicbp; }
    float getdbp() { return diastolicbp; }

    void registration();
    bool login();


};
```

3.3.3 Class Donor



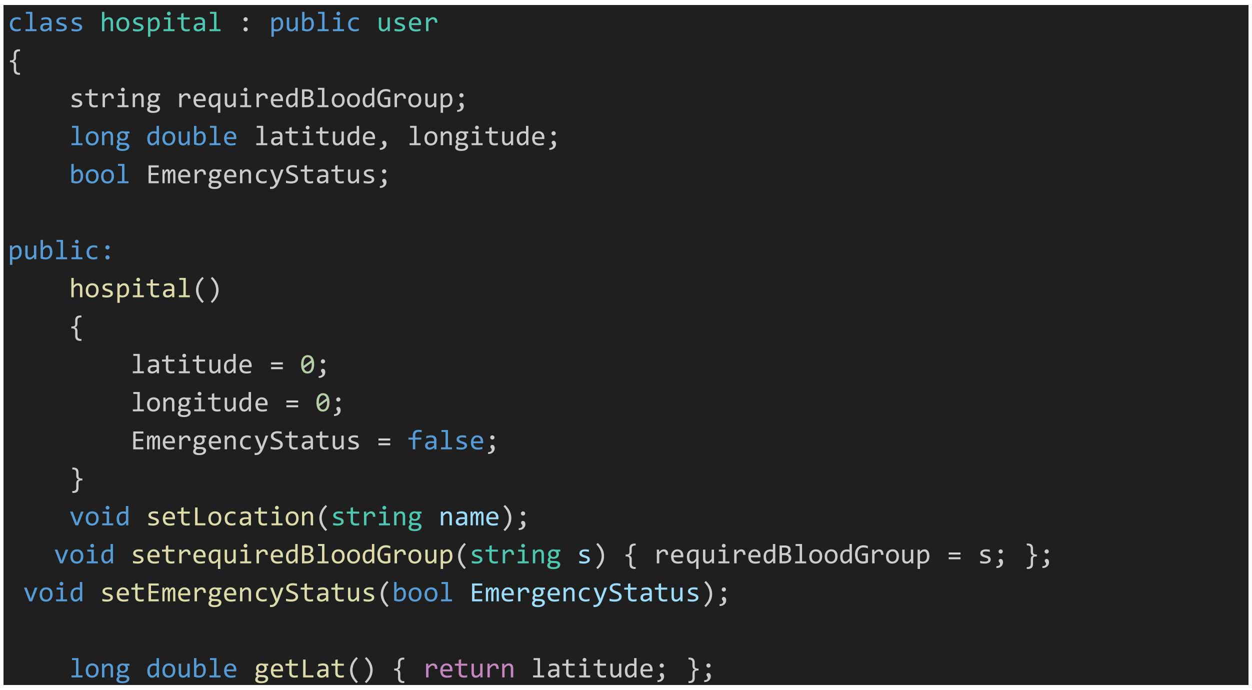
```
class donor : public citizen
{
public:
    bool CheckDonorHealthProfile();

    bool checkLastDonate();
};
```

3.3.4 Class Reciever

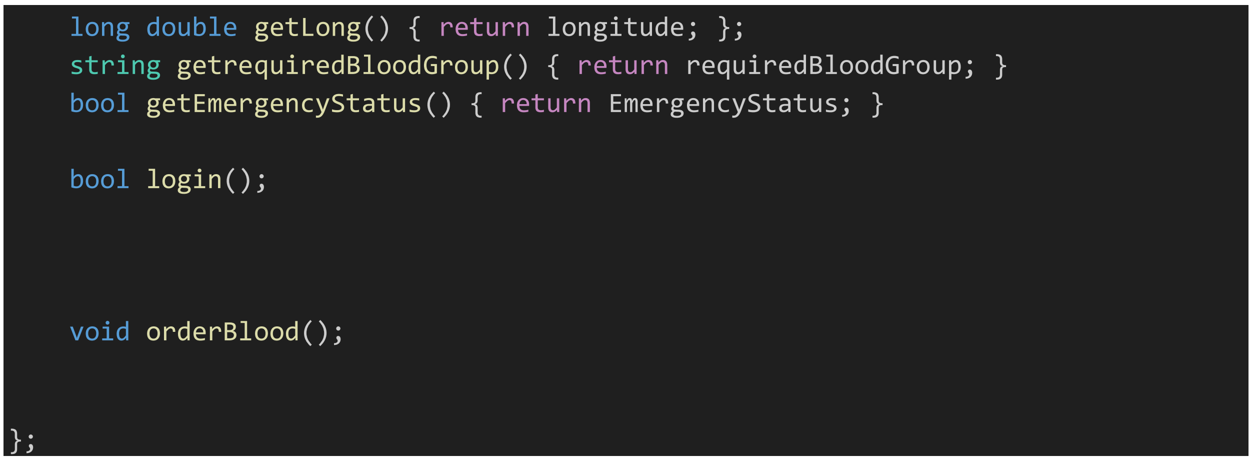


```
        class reciever : public citizen

{
public:
    int bloodPackets();

    bool checkLastRecieve();


};
```

3.3.5 Class Hospital

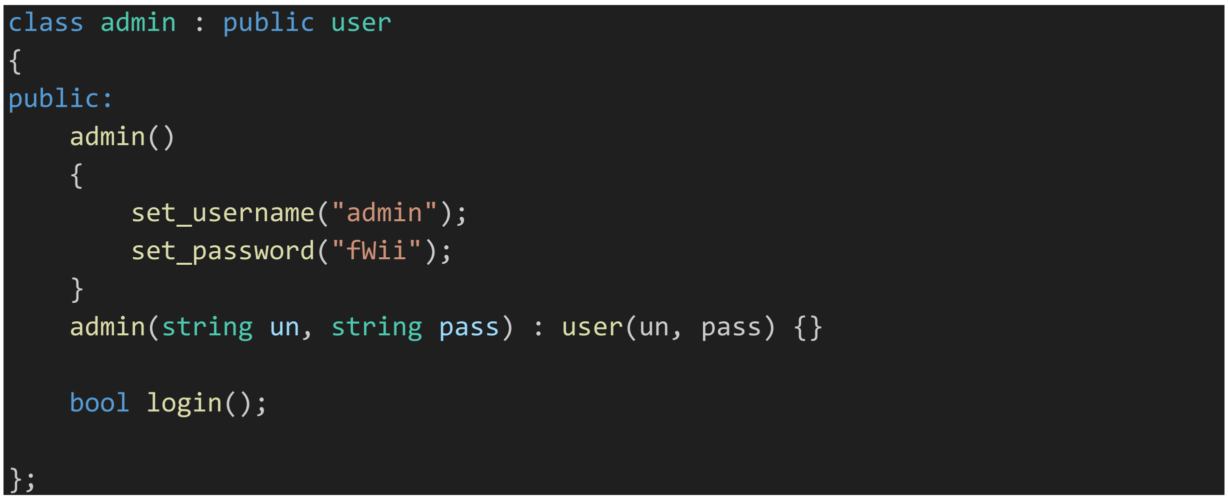


```
class hospital : public user
{
    string requiredBloodGroup;
    long double latitude, longitude;
    bool EmergencyStatus;

public:
    hospital()
    {
        latitude = 0;
        longitude = 0;
        EmergencyStatus = false;
    }
    void setLocation(string name);
   void setrequiredBloodGroup(string s) { requiredBloodGroup = s; };
 void setEmergencyStatus(bool EmergencyStatus);

    long double getLat() { return latitude; };
```



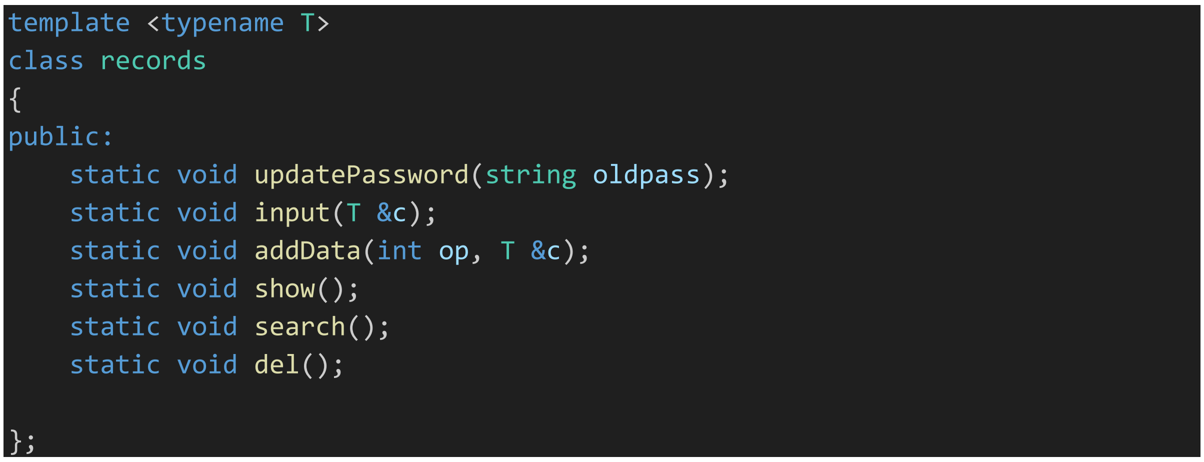
```
    long double getLong() { return longitude; };
    string getrequiredBloodGroup() { return requiredBloodGroup; }
    bool getEmergencyStatus() { return EmergencyStatus; }

    bool login();



    void orderBlood();


};
```

3.3.6 Class Admin



```
class admin : public user
{
public:
    admin()
    {
        set_username("admin");
        set_password("fWii");
    }
    admin(string un, string pass) : user(un, pass) {}

    bool login();

};
```

3.3.7 Class Records



```
template <typename T>
class records
{
public:
    static void updatePassword(string oldpass);
    static void input(T &c);
    static void addData(int op, T &c);
    static void show();
    static void search();
    static void del();

};
```

3.3.8 Class Verification

```cpp
class verification
{
public:
    static bool checkPassword(string pass);
    static bool checkUsername(string user);

};
```

3.3.9 Class Blood Reserves

```cpp
class bloodreserves
{
public:
    static void seeBloodPackets();
    static void increasepackets(string type);
    static void decreasepackets(int num, string type);
};
```

Chapter 4

Conclusion:

The Blood Bank Management System aims to develop a user-friendly and efficient solution that addresses the challenges faced by blood banks. It focuses on managing the reservation, transport, and storage of blood for various purposes, such as medical testing and hospital requirements. By tracking patient needs and existing blood reserves, the system helps identify surplus or shortage of specific blood groups. It ensures seamless transportation of blood reserves to different hospitals and provides emergency assistance by connecting with the nearest blood bank branch. This comprehensive approach tackles the increasing cases of blood deficiency diseases while optimizing the management of blood resources.
The testing of this system was done by our fellow classmates and seniors from whom it received positive feedback.