BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Year 1, Semester 2

Topic : **Blood Donation System**

Group no : **MLB_WD_09.02_06**

Campus : **Malabe**

Submission Date : **19/05/2024**

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	Name	Contact Number
IT23389656	AHAMED M.N.I	0777179669
IT23382244	FERNANDO B.S.N	0740078131
IT23394124	BAGYA R.M.S	0721908494
IT23391468	PERERA K.D.S	0722970694
IT23388284	NETHSARANI P.A.T	0764820245

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Blood Donation System

1] Requirements

1. Blood donation system facilitates to the convenience of people by providing platform to donate blood and blood transfusion.
2. The system has three main type of users who can login to the system donors, blood recipients and staff members.
3. The donors are further clarified as regular donors and first-time donors.
4. The system allows first time donors to get registered to the system by providing user credentials such as name, age, blood type, passwords and etc.
5. Once registered donors can login to the system by entering valid user credentials.
6. Registered donors can donate blood upon request by a recipient or as a regular activity.
7. Donors donating blood books appointments.
8. Once the appointment is booked donor eligibility is again verified.
9. Record details of each donation and health records will appear on donor’s profile.
10. System manages the blood inventory by blood units, blood type and amount
11. Once a request initiated by a recipient system checks inventory for blood availability.
12. System notifies donors on critical blood type shortages, situations with immediate effect and about upcoming donation opportunities.
13. System enabled push notifications and reminders for donations and appointments.
14. System maintains records on blood recipient transfusions, recipient health status and on post transfusion details.

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

2]Classes Identified

1. User
2. Donor
3. User Database
4. Appointment Schedule
5. Blood Inventory
6. Health Record System
7. Eligibility Verifier
8. Notification System
9. Recipient
10. Staff

3]CRC cards for the Online Examination System

User	
Responsibility:	**Collaborators:**
Manage user login and authentication	User Database
Store and retrieve user credentials	User Database
Differentiate user roles	

Donor	
Responsibility:	**Collaborators:**
Register first-time donors	User Database
Login and manage donor profile	User Database
View donation history	User Database
Schedule blood donation appointments	Appointment scheduler
View health records	User Database

Recipient	
Responsibility:	**Collaborators:**
Login and manage profile	User Database
Initiate blood request	Notification system
View transfusion history	Blood Inventory

Staff	
Responsibility:	**Collaborators:**
Manage blood donation process	Blood Inventory
Update blood inventory records	User Database
Verify donor and recipient information	Eligibility Verifier

User Database	
Responsibility:	**Collaborators:**
Store user credentials and profiles	User/Donor/Recipient/staff
Verify login information	Notification system
Update user Information	Database System

Appointment Scheduler	
Responsibility:	**Collaborators:**
Manage available slots	Donor
Schedule and confirm donor appointment	Eligibility verifier
Notify donors of upcoming appointments	Notification system

Eligibility verifier	
Responsibility:	**Collaborators:**
Verify donor eligibility	Donor/Recipient
Check recipient eligibility	Health records database
Update eligibility status	Database system

Health Records Database	
Responsibility:	**Collaborators:**
Store health records	Donor/Recipient
Update health information	Eligibility verifier
Provide health status	Staff

Blood Inventory	
Responsibility:	**Collaborators:**
Track available blood units	Notification system
Update inventory after donation	Donor
Check blood availability	Database system

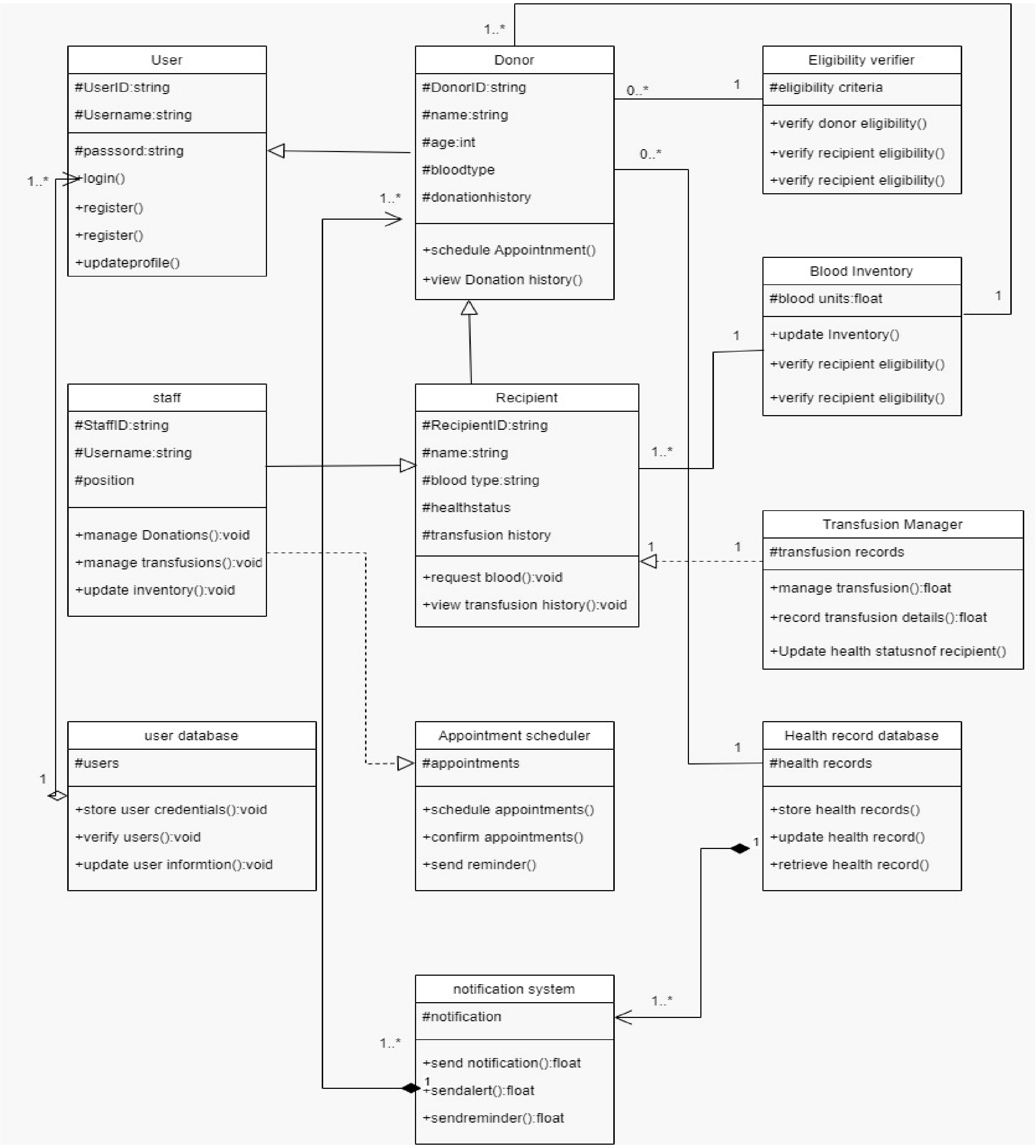
Notification System	
Responsibility:	**Collaborators:**
Send notifications and reminders	User/Donor/Recipient
Send shortage alerts	Blood Inventory
Notify recipients about status	Appointment reminders

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

4]Class Diagram



5]Coding for the classes

main.cpp

```
#include "donor.h"
#include "recipient.h"
#include "staff.h"
#include "userdatabase.h"
#include "appointmentscheduler.h"
#include "eligibilityverifier.h"
#include "healthrecordsDatabase.h"
#include "bloodinventory.h"
#include " notificationsystem.h"
#include " Transfusionmanager.h"
#include <iostream>

using namespace std;

int main() {
    Donor donor;
    donor.registerUser();

    eligibilityverifier Eligibilityverifier;
    if (Eligibilityverifier.verifydonorEligibility(donor)) {
        cout << "Donor is eligible." << endl;

        userdatabase userDB;
        userDB.storeUsercredentials(&donor);

        User* loggedInUser = userDB.verifyUser(donor.username, donor.password);
        if (loggedInUser != nullptr) {
            cout << "Login successful for user: " << loggedInUser->username << endl;

            appointmentscheduler scheduler;
            scheduler.scheduleappointment(donor.donorID);
        }
        else {
            cout << "Login failed." << endl;
        }
    }
    else {
        cout << "Donor is not eligible." << endl;
    }

    return 0;
}
```

user.h

```
#include <string>
using namespace std;
```

```
class user {
protected:
    string UserID;
    string Username;
    string Password;
    string Role;

public:
    virtual void login() = 0;
    virtual void registerUser() = 0;
    virtual void updateProfile() = 0;


    string getUserID() const { return userID; }
    string getUsername() const { return username; }
    string getPassword() const { return password; }
    string getRole() const { return role; }


    void setUserID(const string& id) { userID = id; }
    void setUsername(const string& uname) { username = uname; }
    void setPassword(const string& pass) { password = pass; }
    void setRole(const string& r) { role = r; }

};
```

user.cpp

```
#include "user.h"
string user::getUserID() const { return userID; }
string user::getUsername() const { return username; }
string user::getPassword() const { return password; }
string user::getRole() const { return role; }

void user::setUserID(const string& id) { userID = uid; }
void user::setUsername(const string& uname) { username = u_name; }
void user::setPassword(const string& pass) { password = pass; }
void user::setRole(const string& r) { role = r; }
```

donor.h

```
#include "user.h"
#include <string>
#include <vector>

using namespace std;

class donor : public user {
private:
    string DonorID;
    string Name;
    int Age;
    string Bloodtype;
    vector<string> donationHistory;

public:
    void login() override;
    void registeruser() override;
    void updateprofile() override;
    void scheduleappointment();
    void viewdonationhistory();

string getDonorID() const { return donorID; }
    string getName() const { return name; }
    int getAge() const { return age; }
    string getBloodType() const { return bloodType; }


    void setDonorID(const string& id) { donorID = id; }
    void setName(const string& n) { name = n; }
    void setAge(int a) { age = a; }
    void setBloodType(const string& bt) { bloodType = bt; }
};
```

donor.cpp

```
#include "donor.h"
#include <iostream>

using namespace std;

void donor::login() {
}

void donor::registeruser() {
    cout << "Enter donor ID: ";
    cin >> DonorID;
    cout << "Enter name: ";
    cin.ignore();
    getline(cin, name);
    cout << "Enter age: ";
    cin >> Age;
    cout << "Enter blood type: ";
    cin >> Bloodtype;
    cout << "Enter username: ";
    cin >> Username;
    cout << "Enter password: ";
    cin >> password;
    cout << "Registration successful for Donor: " << name << endl;
}

void donor::updateProfile() {
}

void donor::scheduleAppointment() {
}
```

```
void donor::viewDonationHistory() {
}
string donor::getDonorID() const { return donorID; }
string donor::getName() const { return name; }
int donor::getAge() const { return age; }
string donor::getBloodType() const { return bloodType; }

void donor::setDonorID(const string& id) { donorID = id; }
void donor::setName(const string& n) { name = n; }
void donor::setAge(int a) { age = a; }
void donor::setBloodType(const string& bt) { bloodType = bt; }
```

recipient.h

```
#include "user.h"
#include <string>
#include <vector>

using namespace std;

class recipient : public user {
private:
    string RecipientID;
    string Name;
    string Bloodtype;
    string Healthstatus;
    vector<string> transfusionHistory;

public:
    void login() override;
    void registeruser() override;
    void updateprofile() override;
    void requestblood();
    void Viewtransfusionhistory();

  string getRecipientID() const { return RrecipientID; }
    string getName() const { return Rname; }
    string getBloodType() const { return RbloodType; }
    string getHealthStatus() const { return RhealthStatus; }

    void setRecipientID(const string& id) { RrecipientID = rid; }
    void setName(const string& n) { Rname = rn; }
    void setBloodType(const string& bt) { RbloodType = rbt; }
    void setHealthStatus(const string& hs) { RhealthStatus = rhs; }
};
```

recipient.cpp

```
#include "Recipient.h"
#include <iostream>

using namespace std;
string recipient::getRecipientID() const { return RrecipientID; }
string recipient::getName() const { return Rname; }
string recipient::getBloodType() const { return RbloodType; }
string recipient::getHealthStatus() const { return RhealthStatus; }

void recipient::setRecipientID(const string& id) { RrecipientID = rid; }
void recipient::setName(const string& n) { Rname = rn; }
void recipient::setBloodType(const string& bt) { RbloodType = rbt; }
void recipient::setHealthStatus(const string& hs) { RhealthStatus = rhs; }
void recipient::login() {
}

void recipient::registerUser() {
}

void recipient::updateProfile() {
}

void recipient::requestBlood() {
}

void recipient::viewTransfusionHistory() {
}
```

staff.h

```
#include "user.h"
#include <string>

using namespace std;

class staff : public user {
private:
    string StaffID;
    string Name;
    string Position;

public:
    void login() override;
    void registeruser() override;
    void updateprofile() override;
    void managedonations();
    void managetransfusions();
    void updateinventory();

string getStaffID() const { return staffID; }
    string getName() const { return name; }
    string getPosition() const { return position; }

    void setStaffID(const string& id) { staffID = sid; }
    void setName(const string& n) { name = sn; }
    void setPosition(const string& p) { position = sp; }
};
```

staff.cpp

```
#include "staff.h"
#include <iostream>
```

```
using namespace std;

void staff::login() {
}

void staff::registeruser() {
}

void staff::updateprofile() {
}

void staff::managedonations() {
}

void staff::managetransfusions() {
}

void staff::updateinventory() {
}
string staff::getStaffID() const { return staffID; }
string staff::getName() const { return name; }
string staff::getPosition() const { return position; }

void staff::setStaffID(const string& id) { staffID = sid; }
void staff::setName(const string& n) { name = sn; }
void staff::setPosition(const string& p) { position = sp; }
```

userdatabase.h

```
#include "user.h"
#include <map>
#include <string>

using namespace std;

class userdatabase {
private:
   map<string, user*> users;

public:
   void storeUsercredentials(user* user);
   user* verifyUser(const string& username, const string& password);
   void updateUserInfo(user* user);
};
```

userdatabase.cpp

```
#include "userdatabase.h"

void userdatabase::storeUsercredentials(user* user) {
   users[user->username] = user;
}
```

```
user* userdatabase::verifyUser(const string& username, const string& password) {
    if (users.find(username) != users.end() && users[username]->password == password) {
        return users[username];
    }
    return nullptr;
}

void userdatabase::updateUserInfo(user* user) {
    users[user->username] = user;
}
```

appointmentscheduler.h

```
#include <string>
#include <vector>

using namespace std;

class appointmentscheduler {
private:
    vector<string> appointments;

public:
    void scheduleappointment(const string& DonorID);
    void confirmappointment(const string& AppointmentID);
    void sendreminder(const string& DonorID);
};
```

appointmentscheduler.cpp

```
#include "appointmentscheduler.h"
#include <iostream>

using namespace std;

void appointmentscheduler::scheduleappointment(const string& DonorID) {
    appointments.push_back(DonorID);
    cout << "Appointment scheduled successfully for Donor ID: " << DonorID << endl;
}

void appointmentscheduler::confirmappointment(const string& AppointmentID) {
}

void appointmentscheduler::sendreminder(const string& DonorID) {
}
```

eligibilityverifier.h

```
#include "donor.h"
#include "recipient.h"

using namespace std;
```

```cpp
class eligilibilityverifier
{
private:
    vector<string> Eligibilitycriteria;

public:
    bool verifydonorEligibility(const Donor& donor);
    bool verifyrecipientEligibility(const Recipient& recipient);
};
```

eligibilityverifier.cpp

```cpp
#include "eligibilityverifier.h"

bool eligibilityverifier::verifydonorEligibility(const Donor& donor) {
    return donor.age >= 18 && donor.age <= 65 && !donor.bloodType.empty();
}

bool eligibilityverifier::verifyrecipientEligibility(const Recipient& recipient) {
    return true;
}
```

healthrecordDatabase.h

```cpp
#include <map>
#include <string>

class healthrecordsDatabase {
public:
    map<string, string> Healthrecords;

    void storeHealthrecord(const string& userID, const string& healthRecord);
    void updateHealthrecord(const string& userID, const string& healthRecord);
    string retrieveHealthrecord(const string& userID);
};
```

healthrecordDatabase.cpp

```cpp
#include "healthrecordsDatabase.h"

void healthrecordsDatabase::storeHealthrecord(const string& userID, const string& healthRecord) {
    healthRecords[userID] = healthRecord;
}

void healthrecordsDatabase::updateHealthrecord(const string& userID, const string& healthRecord) {
    healthRecords[userID] = healthRecord;
}

string healthrecordsDatabase::retrieveHealthrecord(const string& userID) {
    return healthRecords[userID];
}
```

bloodinventory.h

```cpp
#include <map>
```

```
#include <string>

class bloodinventory {
public:
    map<string, int> Bloodunits;

    void updateinventory(const string& bloodType, int amount);
    bool checkavailability(const string& bloodType, int amount);
    void reserveBloodunits(const string& bloodType, int amount);
};
```

bloodinventory.cpp

```
#include "bloodinventory.h"

void bloodinventory::updateinventory(const string& bloodType, int amount) {
    Bloodunits[bloodType] += amount;
}

bool bloodinventory::checkAvailability(const string& bloodType, int amount) {
    return bloodUnits[bloodType] >= amount;
}

void bloodinventory::reserveBloodunits(const string& bloodType, int amount) {
    if (checkAvailability(bloodType, amount)) {
        bloodUnits[bloodType] -= amount;
    }
}
```

notificationsystem.h

```
#include <vector>
#include <string>

class notificationsystem {
public:
    vector<string> notifications;

    void sendnotification(const string& userID, const string& message);
    void sendalert(const string& userID, const string& alert);
    void sendreminder(const string& userID, const string& reminder);
};
```

notificationsystem.cpp

```
#include " notificationsystem.h"

void notificationsystem::sendnotification(const string& userID, const string& message) {
    notifications.push_back("To: " + userID + " - " + message);
}

void notificationsystem::sendalert(const string& userID, const string& alert) {
    notifications.push_back("To: " + userID + " - " + alert);
}
```

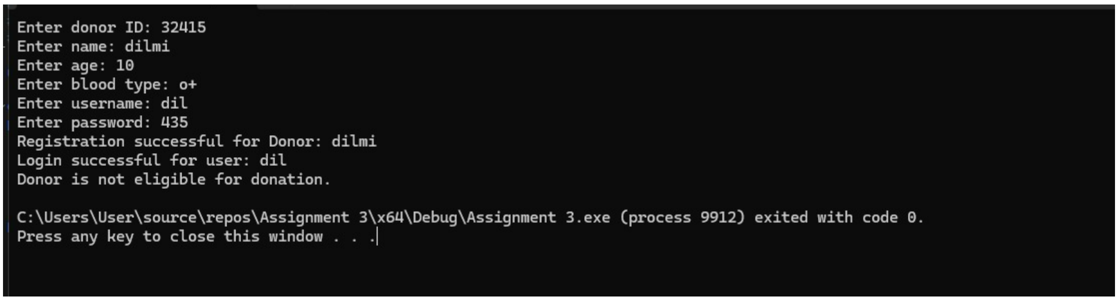
```
void notificationsystem::sendreminder(const string& userID, const string& reminder) {
    notifications.push_back("To: " + userID + " - " + reminder);
}
```

Transfusionmanager.h

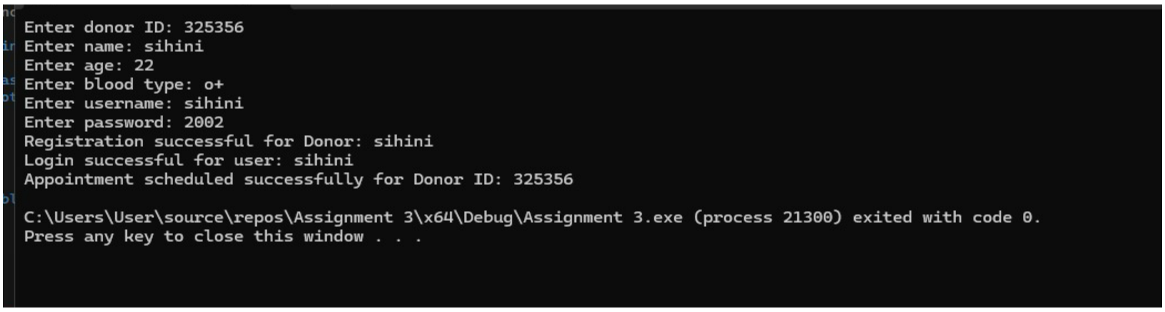
```
#include <vector>
#include <string>

class Transfusionmanager {
public:
    vector<string> transfusionrecords;

    void managetransfusion(const string& recipientID);
    void recordtransfusionDetails(const string& recipientID, const string& details);
    void updateRecipientHealthstatus(const string& recipientID, const string& healthStatus);
};
```

Transfusionmanager.cpp

```
#include " Transfusionmanager.h"
#include "healthrecordsDatabase.h"
#include <iostream>

using namespace std;

void Transfusionmanager:: recordtransfusionDetails (const string& recipientID, const string& details) {
    transfusionrecords.push_back("Recipient: " + RecipientID + " - Details: " + details);
}

void Transfusionmanager:: updateRecipientHealthstatus (const string& RecipientID, const string& healthStatus) {
    healthrecordsDatabase hrdb;
    hrdb.updateHealthrecord(RecipientID, healthStatus);
            }
```



```
Enter donor ID: 32415
Enter name: dilmi
Enter age: 10
Enter blood type: o+
Enter username: dil
Enter password: 435
Registration successful for Donor: dilmi
Login successful for user: dil
Donor is not eligible for donation.

C:\Users\User\source\repos\Assignment 3\x64\Debug\Assignment 3.exe (process 9912) exited with code 0.
Press any key to close this window . . .
```



```
Enter donor ID: 325356
Enter name: sihini
Enter age: 22
Enter blood type: o+
Enter username: sihini
Enter password: 2002
Registration successful for Donor: sihini
Login successful for user: sihini
Appointment scheduled successfully for Donor ID: 325356

C:\Users\User\source\repos\Assignment 3\x64\Debug\Assignment 3.exe (process 21300) exited with code 0.
Press any key to close this window . . .
```

BSc (Hons) in Information Technology

Object Oriented Concepts – IT1050

Assignment 2

Individual Contribution

- **IT23389656** AHAMED M.N.I
  - Created CRC Cards for Donor and User
  - Contributed to gather requirements for creating the Class Diagram.
  - Created the class diagram in drawio.
  - Implemented codes for Donor and User

- IT23382244 FERNANDO B.S.N
  - Implemented the codes for Recipient and Staff.
  - Created CRC Cards for Recipient and Staff.
  - Contributed to class diagram.
  - Wrote the user requirements and helped in report.

- IT23394124 BAGYA R.M.S
  - Implemented the codes for User database and Appointment Scheduler.
  - Created CRC Cards for User database and Appointment Scheduler.
  - Contributed to gather requirements for creating the class Diagram.
  - Implemented the main program.

- IT23391468 PERERA K.D.S
  - Implemented the codes for Eligibility verifier and Health records Database.
  - Created CRC Cards for Eligibility verifier and Health records Database.
  - Contributed to gather requirement for creating the Class Diagram.
  - Implemented the main program.

- IT23388284 **NETHASARANI P.A.T.**
  - Implemented the codes for Blood inventory and Notification system.
  - Created CRC Cards for Blood inventory and Notification system.
  - Contributed to gather requirements for creating the class Diagram.
  - Contributed to making the report.