Topic : Blood Donation System

Group no : MLB_04.02_06

Campus : Malabe

Submission Date:

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**
IT21271946	Samarakoon S.M.B.P.B.	+94758283137
IT21273476	Satharasinghe S.A.C.M.	+94728184017
IT21269820	Munasinghe M.M.A.D.	+94715138565
IT21274466	Burah T.I.O.	+94710640320
IT21270406	Jayasinghe H.D.D.R.	+94725544490

System Requirements

- The system should provide services 24/7/365.
- Guest users can overview the system, to use the system, they must register with the system by providing correct details such as name, address, NIC, Email, and contact.
- Registered users are of two types called donors and organizers where they can log into the system by using the correct username and password.
- People who registered to the system can update user details and view current details.
- They can organize or join blood camps using the system.
- Organizers should be able to add blood camp details such as Location, date, time, and venue to the system.
- Details should be confirmed by the administrator staff.
- Staff can delete or update the status of the blood camp details and blood donation details.
- System should generate a unique id for the blood camp after confirming.
- Before joining blood camps on the system donors must upload their health reports to the system.
- Staff can check health reports of donors.
- Donors should be able to filter blood camp from location, time, and date.
- After confirming the donor details, time slot and unique donation ID is generated to the donate blood.
- Blood bank can request blood entering blood types, and quantity by using the system.
- After the request, system generates unique request ID.
- Administrator can generate reports such as donor records, blood camp details, and staff records.

Identify Classes

- Guest User
- Registered User
- Donor
- Organizer
- Staff
- Reports
- Blood Camp

Drawing CRC Cards

Guest User	
Responsibilities	Collaborations
Register to the system	
Allow to view the system	

Registered User	
Responsibilities	Collaborations
Update user details	
View profile	

Donor	
Responsibilities	Collaborations
Update user details	Registered User
View profile	Registered User
Filter Blood camps	Blood camp
Join Blood camps	Blood camp

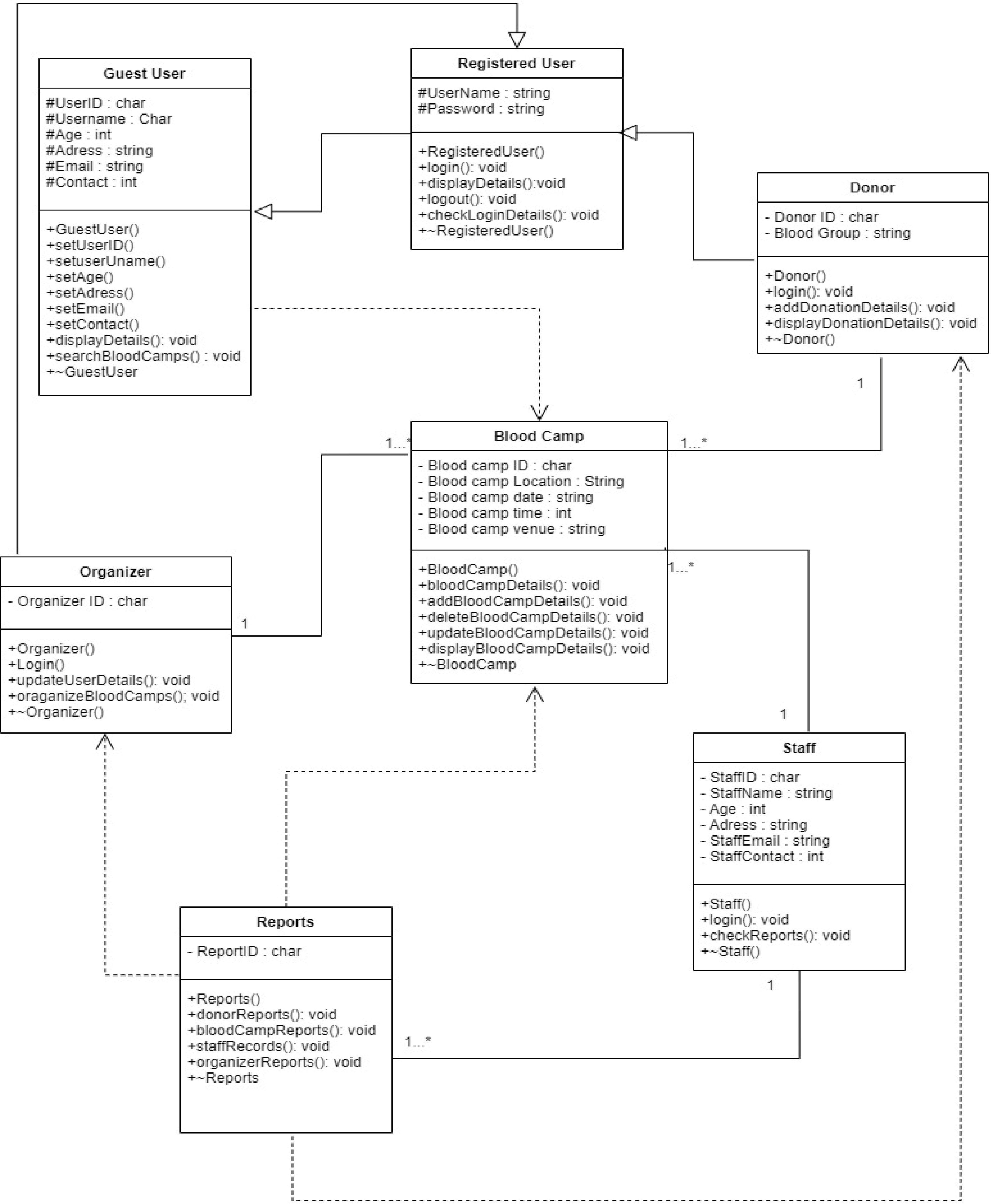
Organizer	
Responsibilities	Collaborations
Update user details	Registered User
View profile	Registered User
Organize Blood camps	Blood Camp

Staff	
Responsibilities	Collaborations
Confirm registration details	Registered user
Confirm blood camp details	Blood Camp
Check Health reports	Reports

Reports	
Responsibilities	Collaborations
Donor records	Donor
Blood camp details	Blood camp
Staff records	Staff
Health Reports	Donor

Blood Camp	
Responsibilities	Collaborations
Add blood camp details	Organizer
Delete blood camp details	Staff
Update blood camp details	Staff

Class Diagram (UML Notation)



```
};
```

Class Header Files

GuestUser.h

```
#include "BloodCamp.h"
class GuestUser
{
protected:
char userID;
char userName[20];
char userAddress[30];
char userEmail[30];
char userContact[10];
public:
GuestUser();
GuestUser(char puserid, const char puserName[], const char
puserAddress[], const char puserEmail[] ,const char puserContact[]);
void searchBloodCamp(BloodCamp* pBC);
void registerUser();
virtual void displayDetails();
~GuestUser();
};
```

RegisteredUser.h

```
#include "RegisteredUser.h"
#include <cstring>
RegisteredUser::RegisteredUser()
{
strcpy(userUsername, "");
strcpy(userPassword, "");
}
RegisteredUser::RegisteredUser(const char puserUsername[],
const char puserPassword[], char puserid, const char puserName[],
const char puserAddress[], const char puserEmail[], const char
puserContact[]) : GuestUser(puserid, puserName, puserAddress,
puserEmail, puserContact)
{
strcpy(userUsername, puserUsername);
strcpy(userPassword, puserPassword);
}
void RegisteredUser::displayDetails()
{
}
void RegisteredUser::login()
{
}
void RegisteredUser::logout()
{
}
char RegisteredUser::checkLoginDetails()
{
return 0;
}
RegisteredUser::~RegisteredUser()
{
//Destructor
}
```

Donor.h

```
#include "RegisteredUser.h"
#include "BloodCamp.h"
#define SIZE 5
class Donor :public RegisteredUser
{
private:
 char donorID;
 char bloodGroup;
public:
 Donor();
 Donor(const char userName[], const char userPwd[], char donorID,
const
char name[], const char address[], const char email[], const char
contact[]);
 void addDonationDetails();
 void login();
 void displayDonorDetails();
 ~Donor();
};
```

Staff.h

```
#include "BloodCamp.h"
#include "Report.h"
#define SIZE 5
class Staff
{
private:
char staffID;
char staffName[20];
char staffEmail[20];
char staffNumber[10];
char staffUsername[20];
char staffPassword[20];
BloodCamp* BC[SIZE];
public:
Staff();
Staff(char pstaffID, const char pstaffName[], const char
pstaffEmail[], const char pstaffNumber[], const char
pstaffUsername[], const char pstaffPassword[]);
void login(const char stfUsername, const char stfPsword );
void checkReport(Report* pRP);
void manage(BloodCamp*pBC);
~Staff();
};
```

Organizer.h

```
#include "RegisteredUser.h"
#include "BloodCamp.h"
#define SIZE 5
class Oraganizer : public RegisteredUser
{
private:
 char organizerID
public:
 Organizer();
 Organizer(const char userName[], const char userPwd[], char organizerID, const char name[], const char address[], const char email[], const char contact[]);
 void updateUserDetails;
 void login();
 void organizeBloodCamps();
 ~Organizer();
};
```

BloodCamp.h

```
#include "Donor.h"
#include "Organizer.h"
#include "Staff.h"
#define SIZE1 2
#define SIZE2 2
class BloodCamp
{
private:
char bloodCampID;
char bloodCampLoctaion[50];
char bloodCampDate[10];
char bloodCampTime[10];
char bloodCampVenue[20];
int count = 0;
Donor* donor;
Organizer* organizer;
Staff* staff;
public:
BloodCamp();
BloodCamp(Organizer* porganizer, Donor* pdonor,Staff* pstaff);
void bloodCampDetails(char bcID, const char bcLocation, const char bcTime, const char bcDate, const char bcVenue , Organizer* porganizer , Donor* pdonor , Staff* pstaff);
void BloodCampDetails();
void deleteBloodCampDetails();
void updateBloodCampDetails();
void addBloodCampDetails();
void displayBloodCampDetails();
~BloodCamp();
};
```

Report.h

```
#include"Donor.h"
#include"BloodCamp.h"
```

```
#include"Staff.h"

#include"Organizer.h"

#define SIZE1 5

#define SIZE2 5

#define SIZE3 5

#define SIZE4 5

class Report

{

private:

char reportID;

public:

Report();

Report(BloodCamp*pBC[], Staff*pstaff, Organizer*pOR[], Donor*pDO);

void donorReports();

void bloodCampReports();

void staffReports();

void organizerReports();

~Report();

};
```

Class Cpp Files

GuestUser.cpp

```
#include "GuestUser.h"

#include <cstring>

GuestUser::GuestUser()

{

userID = 0;

strcpy(userName, "");

strcpy(userAddress, "");

strcpy(userEmail, "");

strcpy(userContact, "0000000000");

}

GuestUser::GuestUser(char puserid, const char puserName[], const char

puserAddress[], const char puserEmail[], const char puserContact[])

{

userID = puserid;

strcpy(userName, puserName);

strcpy(userAddress, puserAddress);

strcpy(userEmail, puserEmail);

strcpy(userContact, puserContact);
}
void GuestUser::searchBloodCamp(BloodCamp* pBC)
{
}
void GuestUser::registeredUser()
{
}
void GuestUser::displayDetails()
{
}
GuestUser::~GuestUser()
{
//Destructor
}
```

RegisteredUser.cpp

```
#include "RegisteredUser.h"
#include <cstring>
RegisteredUser::RegisteredUser()
{
strcpy(userUsername, "");
strcpy(userPassword, "");
```

```
}
RegisteredUser::RegisteredUser(const char puserUsername[],
const char puserPassword[], char puserid, const char puserName[],
const char puserAddress[], const char puserEmail[], const char
puserContact[]) : GuestUser(puserid, puserName, puserAddress,
puserEmail, puserContact)
{
strcpy(userUsername, puserUsername);
strcpy(userPassword, puserPassword);
}
void RegisteredUser::displayDetails()
{
}
void RegisteredUser::login()
{
}
void RegisteredUser::logout()
{
}
char RegisteredUser::checkLoginDetails()
{
return 0;
```

```
}
RegisteredUser::~RegisteredUser()
{
//Destructor
}
```

Donor.cpp

```
#include "Donor.h"
Donor::Donor()
{
donorID = 0;
strcpy(bloodGroup,"");
}
Donor::Donor(const char userName[], const char userPwd[], char donorID, const char name[], const char address[], const char email[], const char
contact[],) :RegisteredUser(usName,
usPwd, id, name, address, email, contact)

void Donor::addDonationDetails()
{
}
void Donor::login()
```

```
{
}
void Donor::displayDonorDetails()
{
}
Donor::~Donor()
{
//Destructor
}
void Donor::displayDonorDetails()
{
}
Donor::~Donor()
{
//Destructor
}
```

Staff.cpp

```
#include "Staff.h"
#include<cstring>
Staff::Staff()
{
```

```
staffID = 0;
strcpy(staffName, "");
strcpy(staffEmail, "");
strcpy(staffContact, "0000000000");
strcpy(staffUsername, "");
strcpy(staffPassword, "");
}
Staff::Staff(char pstaffID, const char pstaffName[], const char
pstaffEmail[], const char pstaffContact[], const char
pstaffUsername[], const char pstaffPassword[])
{
staffID = pstaffID;
strcpy(staffName, pstaffName);
strcpy(staffEmail, pstaffEmail);
strcpy(staffContact, pstaffContact);
strcpy(staffUsername, pstaffUsername);
strcpy(staffPassword, pstaffPassword);
}
void Staff::login(const char stfUsername, const char stfPsword)
{
}
void Staff::manage(BloodCamp* pBC)
```

```
{
}
void Staff::checkReport(Report*pRP)
{
}
Staff::~Staff()
{
//Destructor
}
```

Organizer.cpp

```
#include "Organizer.h"
Organizer::Organizer()
{
organizerID = 0;
}
Organizer::Organizer(const char userName[], const char userPwd[], char organizerID, const char name[], const char address[], const char email[], const char contact[]):RegisteredCustomer(userName,userPwd, userid, name, address, email, telno)

void Organizer::updateUserDetails()
{
}
void Organizer::login()
{
}
void Organizer::organizeBloodCamps()
{
}
Organizer::~Organizer()
{
//Destructor
}
```

Bloodcamp.cpp

```
#include "BloodCamp.h"
#define SIZE1 2
#define SIZE2 2
BloodCamp::BloodCamp()
{
}
BloodCamp::BloodCamp(Organizer* porganizer, Donor* pdonor,Staff* pstaff)
{
donor = pdonor;
```

```
organizer = porganizer;

staff = pstaff;

}

void BloodCamp::bloodCampDetails(char bcID, const char
bcLocation, const char bcTime, const char bcDate, const char
bcVenue , Organizer* porganizer , Donor* pdonor , Staff* pstaff)

{

}

void BloodCamp::deleteBloodCampDetails()

{

}

void BloodCamp::updateBloodCampDetails()

{

}

void BloodCamp::displayBloodCampDetails()

{

}

void BloodCamp::addBloodCampDetails()

{

}

BloodCamp::~BloodCamp()

{
```

```
//Destructor
}
```

Report.cpp

```
#include "Report.h"
Report::Report()
{
  reportID =0;
}
Report::Report(BloodCamp*pBC[], Staff*pstaff, Organizer*pOR[], Donor*pDO)


void Report:donorReports()
{
}
void Report::bloodCampReports()
{
}
void Report::staffReports()
{
}
void Report::organizerReports()
{
}
Report::~Report()
{
//Destructor
}
```

Main Program

Main.cpp

```
#include "Donor.h"
#include "Organizer.h"
#include "Staff.h"
#include "BloodCamp.h"
#include "GuestUser.h"
#include "RegisteredUser.h"
#include "Report.h"
#include <iostream>
using namespace std;
int main()
{
        //---- Object creation ------
        GuestUser* rg = new RegisteredUser(); // Object - RegisteredUser class
        RegisteredUser* donor = new Donor(); // Object - Donor class
        RegisteredUser* organizer = new Organizer(); // Object - Organizer class
        BloodCamp* BC = new BloodCamp(); // Object - BloodCamp class
        Staff* staff = new Staff(); // Object - Staff class
        Report* report = new Report(); // Object - Report class

        //----Method Calling------
        rg->login();
        rg->displayDetails();
        donor->login();
        donor->displayDonorDetails();
        organizer->login();
        organizer->displayOrganizerDetails();
        BC->updateBloodCampDetails();
        BC->deleteBloodCampDetails();
        BC->displayBloodCampDetails();
        report->donorReports();
        report->bloodCampReports();
        report->staffReports();
        report->organizerReports();
```

```
    //----Delete Dynamic objects------

    delete rg;

    delete donor;

    delete organizer;

    delete BC;

    delete report;

    return 0;

}
```

Registration No	**Name**	**Contribution**
IT21271946	Samarakoon S.M.B.P.B.	Main.cpp RegisteredUser.h RegisteredUser.cpp
IT21273476	Satharasinghe S.A.C.M.	
IT21269820	Munasinghe M.M.A.D.	Organizer.h Organizer.cpp Report.h Report.cpp
IT21274466	Burah T.I.O.	BloodCamp.h BloodCamp.cpp Staff.h Staff.cpp
IT21270406	Jayasinghe H.D.D.R.	GuestUser.h GuestUser.cpp Donor.h Donor.cpp