Sri Lanka Institute of Information Technology

Assignment 2

MLB_WE_01.01_06

Blood Donating System

Object Oriented Concepts – IT1050

B.Sc. (Hons) in Information Technology

Group Details

Group Number: 01.01_06

Project Title: Blood Donating System

	Student ID	Student Name	Email	Contact Number
1	IT21256196	WIJERATHNA N. T	it21256196@my.sliit.lk	0760063900
2	IT21304842	DISSANAYAKE A.S.H	it21304842@my.sliit.lk	0719045077
3	IT21272936	SILVA S. K. D. T	it21272936@my.sliit.lk	0716994576
4	IT21338120	WIJESINGHE S. A. A. K	it21338120@my.sliit.lk	0769827219
5	IT21292040	SASANKA M.W.K. L	it21292040@my.sliit.lk	0774181047

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.

CONTENT

➢ Case Scenario

In this system donors can create donor accounts and they can use the account to book a date, venue, and a time to donate blood. Then they can donate blood without staying in queues. They can acknowledge their selves through this system about the blood campaigns. They can request for a guider if they have any difficulties when using the system.

Through this system the receivers make an user account for themselves and check whether the blood they need is available on the blood bank or not, and they can make a reservation for it. They also can ask for a help of a guider when they are using the system. They can give feedbacks about the services they got.

As for the charity clubs and the other parties who organize blood campaigns, they can advertise their events on this website through the system admin. The system admin is already having an account which can manage all the other user accounts.

The donors should co-operate with the employees during the donation process. The donor should have covered at least the basic requirements. So, when they are registering to the system, they should fill a form which contains the details about their health condition, personal affairs etc. This form is directed to the lab doctor by the system, who has logged in with their employee account which has given by the system admin. If the lab doctor approves the registration the donor's registration becomes complete. The staff members who has logged in with their special credentials give the guidance for the donor and patient.

The numbered blood bags are stored in the blood bank and the blood bank maintains a report which mention the types of blood, the quantity on hand, the blood types which are out of stock etc.

Please note that in the above case scenario, parts in;

Yellow highlighted are Nouns

Green highlighted are Verbs

➢ Noun Verb Analysis

✓ Nouns

Donor	Class
Donor account	Class
Date	Attribute
Venue	Attribute
Time	Attribute
Blood	Class
Queues	Out of scope
Blood campaigns	Class
Guider	Redundant
Patient	Class
Patient account/user account	Class
Reservation	This is related to an action
Blood bank	Class
Feedbacks	Class
Charity clubs	Out of scope
System admin	Class
System admin account	Class
Employees	Class
Donation process	Meta language
Requirements	This is related to an action
Form	This is related to an action
Health condition	Attribute of a donor
Personal affairs	Attribute of a donor
Lab doctor	Redundant
Registration	This is related to an action
Staff member	Redundant
Guidance	This is related to an action
Blood bags	Attribute
Report	This is related to an action

✓ Verbs

- **Donors**
  - Create donor accounts
  - Use, book a date, venue, and a time
  - Donate
  - Acknowledge
  - Request for a guider

- **Patient**
  - Make a user account
  - Check whether the blood they need is available
  - Make a reservation
  - Ask for some help
  - Give feedbacks

- **System admin**
  - Advertise blood campaigns
  - Manage accounts

- **Employee**
  - approve the registration
  - give the guidance

- **Blood bank**
  - Maintain reports

➢ Identifying Classes

- **Donor**
- **Donor Account**
- **Patient**
- **Patient Account**
- **Employee**
- **Employee Account**
- **Admin**
- **Admin Account**
- **Feedback**
- **Blood Bank**
- **Blood camp**

➢ CRC Cards

Donor

Donor Class	
Responsibilities	**Collaborators**
• Make a user account	• Donor account

Donor Account

Donor Account Class	
Responsibilities	**Collaborators**
• Reserve date,time, venue for the donation • Getting guidance from system • Request for a guider	• Employee account • Employee account • Employee account

Patient

Patient Class	
Responsibilities	**Collaborators**
• Make a user account	• Patient account

Patient Account

Patient Account Class	
Responsibilities	**Collaborators**
• Check whether the blood they need is available	• Blood bank
• Make a reservation	• Employee account
• Ask for a help	• Employee account
• Get feedbacks	• Feedbacks

- **Employee**

Employee Class	
Responsibilities	**Collaborators**
• Make a user account	• Employee account

- **Employee Account**

Employee Account Class	
Responsibilities	**Collaborators**
• Should give guidance • Filling the form for health conditions • Directing the form	• Donor account • Patient account • Employee account

Admin

Admin Class	
Responsibilities	**Collaborators**
• Login with special credentials	• Admin account

Admin Account

Admin Account Class	
Responsibilities	**Collaborators**
• Give special credentials • Advertise blood campaigns • Manage accounts	• Employee account • Blood camp • Employee account, Donor account, Patient account

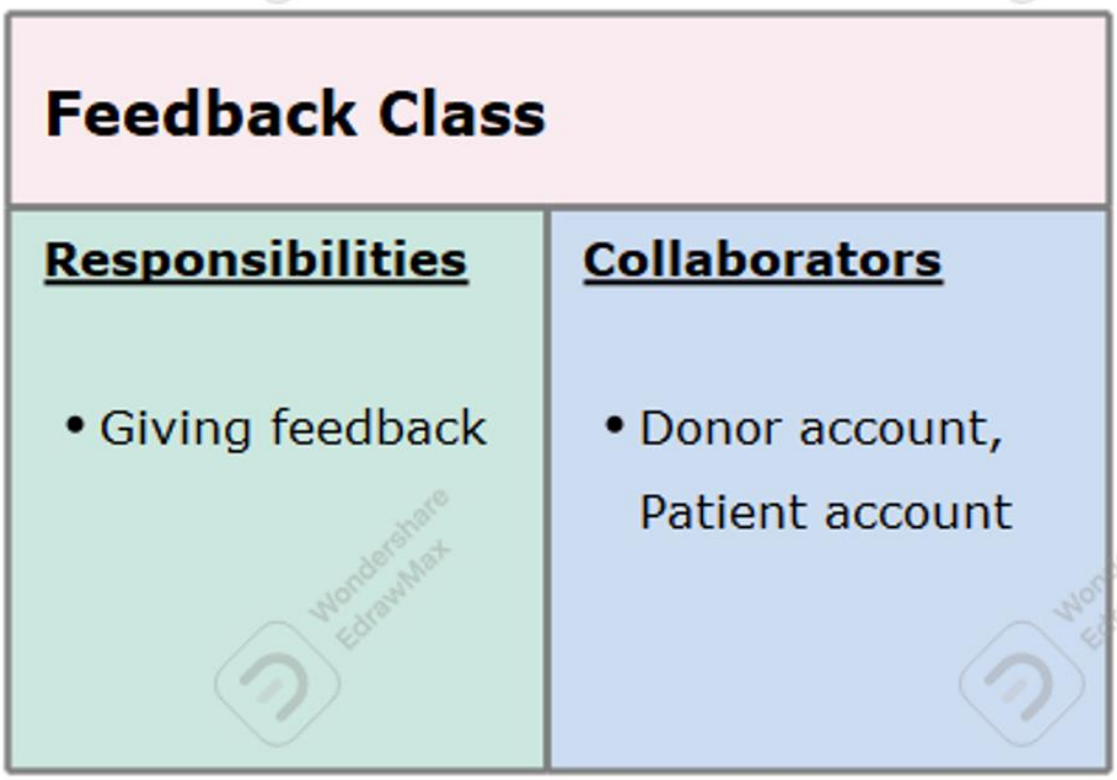
- **Blood Bank**

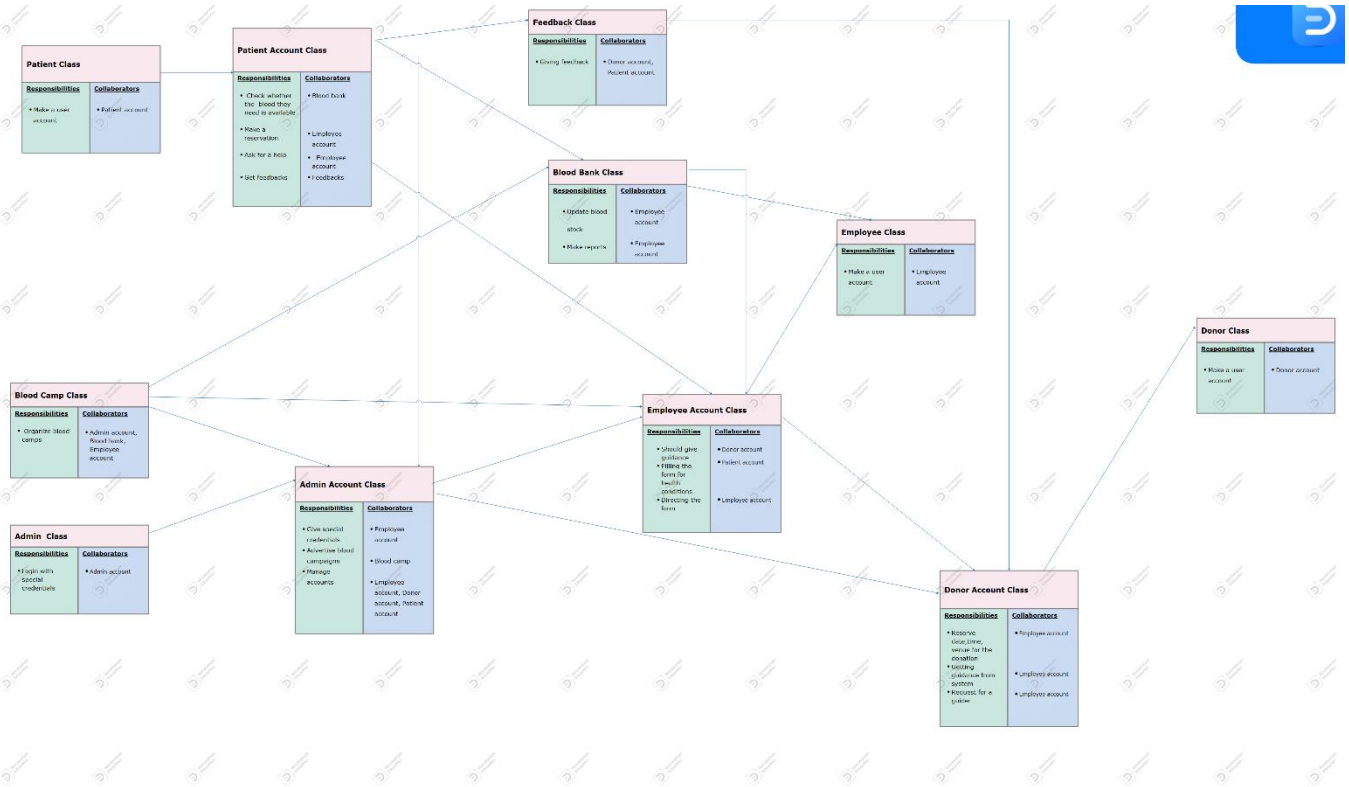
Blood Camp Class	
Responsibilities	**Collaborators**
• Organize blood camps	• Admin account, Blood bank, Employee account

- **Blood camp**

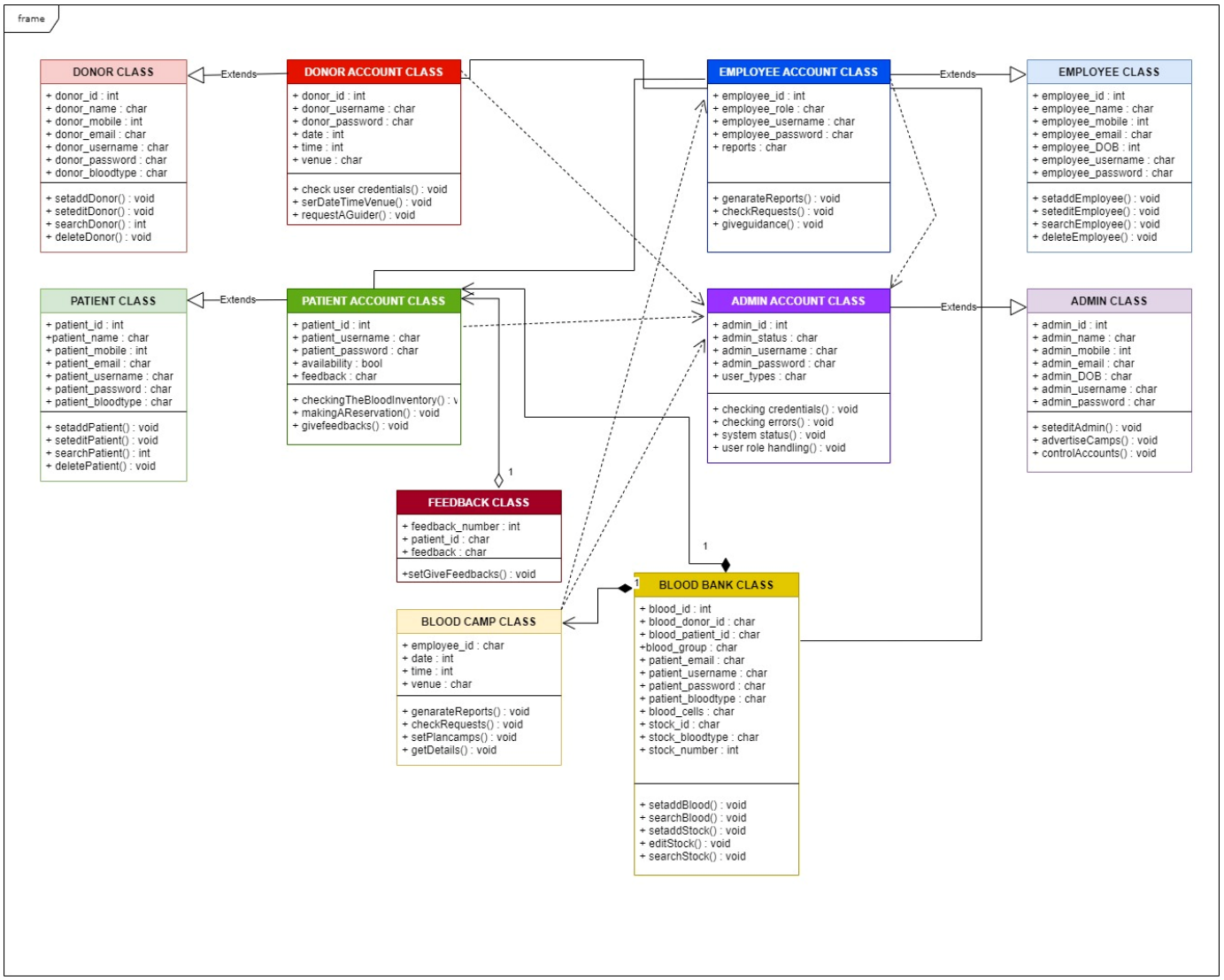
Blood Bank Class	
Responsibilities	**Collaborators**
• Update blood stock • Make reports	• Employee account • Employee account

Feedback


Feedback Class
Responsibilities
• Giving feedback
Collaborators
• Donor account,
Patient account


Patient Class
Responsibilities
Collaborators
Patient Account Class
Responsibilities
Collaborators
Feedback Class
Responsibilities
Collaborators
Blood Bank Class
Responsibilities
Collaborators
Employee Class
Responsibilities
Collaborators
Donor Class
Responsibilities
Collaborators
Blood Camp Class
Responsibilities
Collaborators
Employee Account Class
Responsibilities
Collaborators
Admin Account Class
Responsibilities
Collaborators
Admin Class
Responsibilities
Collaborators
Donor Account Class
Responsibilities
Collaborators

➢ Class Diagram



➢ Coding the Classes

```
// Main Program
#include <iostream>
#include <string.h>

#include "Admin.h"
#include "AdminAccount.h"
#include "Donor.h"
#include "donorAccount.h"
#include "Patient.h"
#include "patientAccount.h"
#include "Employee.h"
#include "employeeAccount.h"
#include "bloodBank.h"
#include "bloodCamp.h"
#include "Feedback.h"

using namespace std;

int main()
{
	//defining pointers
	Admin* systemAdmin_1;

	//create dynamic objects of Admin class using overloading constructor
	systemAdmin_1 = new Admin( "001", "Lakshan", "0774567099", "lakshan99@gmail.com", "A_Lakshan", "Abcd@1234", "1999-08-10" );

	//defining pointers
	Donor* donor_1;

	//create dynamic objects of Donor class using overloading constructor
	donor_1 = new Donor( "D001", "Ishini", "0764522376", "ish21@gmail.com", "D_Ishini", "Qwert@4321", "B+" );
```

```
        //defining pointers
        Patient* patient_1;

        //create dynamic objects of Patient class using overloading constructor
        patient_1 = new Patient( "1001", "Ravindu", "071342134", "Ravi2001@gmail.com", "P_ravindu","Asdf@2001", "O+" );


        //defining pointers
        Employee* employee_1;

        //create dynamic objects of Employee class using overloading constructor
        employee_1 = new Employee( "02001", "Nethmi", "0774127890", "nethu@gmail.com", "2001-07-12", "E_Nethmi", "NeTh@#2001" );



        //defining pointers
        AdminAccount* AA_1;

        //create dynamic objects of AdminAccount class using overloading constructor
        AA_1 = new AdminAccount( "001", "System_Admin of blood system", "A_Lakshan", "Abcd@1234", "System_Admin");



        //defining pointers
        DonorAccount* DA_1;

        //create dynamic objects of DonorAccount class using overloading constructor
        DA_1 = new DonorAccount( "DA001", "D_Ishini", "Qwert@4321", "2022-05-12", "14.00", "Panadura" );


        //defining pointers
        PatientAccount* PA_1;

        //create dynamic objects of PatientAccount class using overloading constructor
        PA_1 = new PatientAccount( "1001", "P_ravindu", "Asdf@2001", "True", "Good" );
```

```
//defining pointers
EmployeeAccount* EA_1;

//create dynamic objects of EmployeeAccount class using overloading constructor
EA_1 = new EmployeeAccount( "02001", "Lab-Docotor", "Blood-reports,daily-reports", "E_Nethmi", "NeTh@#2001" );



//defining pointers
bloodCamp* Camp_1;

//create dynamic objects of bloodCamp class using overloading constructor
Camp_1 = new bloodCamp( "02001", "24", "08.00", "Panadura", "Give blood save life" );

//defining pointers
bloodBank* BlooBank_1;

//create dynamic objects of bloodBank class using overloading constructor
BlooBank_1 = new bloodBank( "B112", "D001", "1001", "B+", "Ravi2001@gmail.com", "P_ravindu" , "Asdf@2001", "O+", "point11", "ST098", "AB+", "98" );



//defining pointers
Feedback* F1;

//create dynamic objects of Feedback class using overloading constructor
F1 = new Feedback( "100", "1001", "Good work" );
```

```
        //Manually (delete) Remove pointers

        delete systemAdmin_1;
        delete donor_1;
        delete patient_1;
        delete employee_1;
        delete AA_1;
        delete DA_1;
        delete PA_1;
        delete EA_1;
        delete Camp_1;
        delete BlooBank_1;
        delete F1;


    return 0;
}
```

```
//Class of Admin

class Admin
{
	protected: // Inheritence relationship

		int adminId;
		char adminName[50];
		int adminMobile[10];
		char adminEmail[20];
		char adminUsername[20];
		char adminPassword[20];
		char adminDOB[10];

	public:
		Admin(); //Default constructor
		Admin( int aId, const char aName[], int aMobile[], char aEmail[], char aUsername[], char aPwd[], const char aBOD[] ); // Overloading constructor

		void seteditAdmin(const int aMobile[],const char aEmail[],const char aUsername[],const char apwd[]);
		void advertiseCamps();
		void controlAccounts();

		~Admin(); //destructor
};
```

```
//Class of  Admin Account

class AdminAccount
{
        private:

                int adminId;
                char adminStatus[50];
                char adminUsername[20];
                char adminPassword[20];
                char userTypes[10];

        public:

                AdminAccount(); //Default constructor
                AdminAccount( int aAId, char aAStatus[], char aAUsername[], char aAPwd[], char typeUser[] ); // Overloading constructor

                void checkingCredentials();
                void checkingErrors();
                void systemStatus();
                void userRoleHandling();

                ~AdminAccount(); //destructor
};
```

```
//Class of donor

class Donor
{
	protected:// Inheritence relationship

		char donorId[20];
		char donorName[50];
		int dornMobile;
		char donorEmail[20];
		char donorUsername[20];
		char donorPassword[20];
		char donorBloodtype[3];

	public:

		Donor(); //Default constructor
		Donor( char dId[], char dName[], int dMobile, char dEmail[], char dUsername[], char dPwd[], char dBloodType[] ); // Overloading constructor

		void setaddDonor(const char dId[],const char dName[], int dMobile, char dEmail[],const char dBloodType[] );
		void seteditDonor(const int dMobile,const char dEmail[],const char dUsername[],const char dPassword[]);
		int searchDonor();
		void deleteDonor(const int did[]);

		~Donor(); //destructor

};
```

```
// Class of donor account

#define SIZE 2
class DonorAccount
{
	private:

		char donorId[20];
		char donorUsername[20];
		char donorPassword[20];
		char date[10];
		int time[5];
		char venue[20];

		EmployeeAccount* employees[SIZE]; //Bi-Directional Association relationship

	public:

		DonorAccount(); //constructor
		DonorAccount( char dAId[], char dAUsername[], char dAPwd[], int reserveDate[], int reserveTime[], char reserveVenue[] ); // Overloading constructor

		void CheckUserCredentials();
		void setDateTimeVenue( int reserveDate[], int reserveTime[], char reserveVenue[] );
		void requestAGuider( AdminAccount* a ); //Dependancy relationship

		~DonorAccount(); //destructor

};
```

```
// Class of patient

class Patient
{
        protected:// Inheritence relationship

                int patientId[20];
                char patientName[50];
                int patientMobile;
                char patientEmail[20];
                char patientUsername[20];
                char patientPassword[20];
                char patientBloodtype[3];

        public:

                Patient(); //constructor
                Patient( int pId, char pName[], int pMobile, char pEmail[], char pUsername[], char pPwd[], char pBloodType[] ); // Overloading constructor

                void setaddPatient( const int pId[],const char pName[], int pMobile, char pEmail[],const char pBloodType[] );
                void seteditPatient( int pMobile[], char pEmail[], const char pUsername[], char pPassword[]) ;
                int searchPatient();
                void deletePatient( const int pId[] );

                ~Patient(); //destructor
};
```

```
//Class of patient account

#define SIZE 2
#define SIZEF 3
class PatientAccount
{
	private:

		int patientId;
		char patientUsername[20];
		char patientPassword[20];
		bool availability[];
		char feedbacks[50];

		EmployeeAccount* employees[SIZE]; //Bi-Directional Association relationship
		Feedback* F[SIZEF]; //aggregation relationship

	public:

		PatientAccount(); //constructor
		PatientAccount( int pAId, char pAUsername[], char pAPwd[], bool bAvailability[], char pAFeedback[] );  // Overloading constructor

		void CheckingTheBloodInventory();
		void MakingAReservation( AdminAccount* a ); //Dependancy relationship

		void GiveFeedbacks( Feedback* F1, Feedback* F2, Feedback* F3 ) //Aggregation relationship
			{
				feedbacks[0] = F1;
				feedbacks[1] = F2;
				feedbacks[2] = F3;
			};


		~PatientAccount(); //destructor
};
```

```
//Class of employee

class Employee
{
	protected:// Inheritence relationship

		int employeeId[20];
		char employeeName[50];
		int employeeMobile;
		char employeeEmail[20];
		char employeeDOB[10];
		char employeeUsername[20];
		char employeePassword[20];

	public:

		Employee(); //constructor
		Employee( int eId[], char eName[], int eMobile, char eEmail[], char eDOB[], char eUsername[], char ePwd[] ); // Overloading constructor

		void setaddEmployee(const int eId[],const char eName[], int eMobile, char eEmail[],const char eBloodtype[] );
		void seteditEmployee( int Emobile,const char eMail[],const char eUsername[], char ePwd[]);
		int searchEmployee();
		void deleteEmployee(const int eId[]);

		~Employee(); //destructor

};
```

```
//Class of employee account

#define SIZEP 10
#define SIZED 10
#define SIZEBC 10
class EmployeeAccount
{
	private:

		int employeeId;
		char employeeRole[20];
		char reports[50];
		char employeeUsername[20];
		char employeePassword[20];

		PatientAccount*	patients[SIZEP];	//Bi-Directional Association relationship
		DonorAccount*	donors[SIZED];	//Bi-Directional Association relationship
		bloodCamp* camps[SIZEBC];  //Bi-Directional Association relationship

	public:

		EmployeeAccount(); //constructor
		EmployeeAccount( int eAId, char eARole[], char eAReports[], char eAUsername[], char eAPwd[] ); // Overloading constructor

		void generateReports( AdminAccount* a ); //Dependancy relationship
		void checkRequests();
		void giveGuidance();

		~EmployeeAccount(); //destructor

};
```

```
// Class of blood bank

class bloodBank
{
	private:

		char bloodId[20];
		char bloodDonorId[20];
		char bloodPatientId[20];
		char bloodGroup[3];
		char patientEmail[20];
		char patientUsername[20];
		char patientPassword[20];
		char patientBloodtype[3];
		char bloodCells[10];
		char stockID[20];
		char stockBloodtype[3];
		int stockNumber;

		PatientAccount* pAccount[]; //composition relationship
		bloodCamp* bCamp[]; //composition relationship

	public:

		bloodBank(); //constructor
		bloodBank( char bId[], char bDId[], char bPId[], char bGroup[], char pEmail[], char pUsername[], char pPwd[], char pBloodType[], char bCells[], char stId[], char stBloodType[], int sNo ); // Overloading constructor

		void setaddblood( char bId[], char bGroup[] );
		void searchblood();
		void setaddstocks( char stId[], char stBloodType[], int sNo, char bId[] );
		void editstocks();
		void searchstocks();

		void addPAccout(); //composition relationship
		void hostBloodCamp(); //composition relationship

		~bloodBank(); //destructor
	};
```

```
// Class of blood camp

#define SIZE 5
class bloodCamp
{
	private:

		char employee_id[20];
		int date;
		int time;
		char venue[20];
		char theme[100];

		EmployeeAccount* employees[SIZE]; //Bi-Directional Association relationship

	public:

		bloodCamp(); //constructor
		bloodCamp( char eId[], int campDate, int campTime, char campVenue[], char campTheme[] ); // Overloading constructor

		void setPlancamps( int campDate[], int campTime[], char campVenue[], char campTheme[] );
		void GetDetails( AdminAccount* a ); //Dependancy relationship
		void generateReports();
		void CheckRequests( employeeAccount* a ); //Dependancy relationship

		~bloodCamp(); //destructor

};
```

```
// Class of feedbacks

class Feedback
{
        private:

                int feedbackNumber[20];
                char patientId[20];
                char feedback[50];

        public:

                Feedback(); //constructor
                Feedback( int FBno[], char pId[], char fb[] ); // Overloading constructor

                void setGiveFeedbacks( int fBNo[], char pId[], char fb[] );

                ~Feedback(); //destructor
};
```

➢ Contribution

	Student ID	Student Name	Contribution
1	IT21256196	WIJERATHNA N. T	• Class diagram • Donor Class • Donor Account Class • Feedback Class
2	IT21304842	DISSANAYAKE A.S.H	• Noun Analysis • Blood Bank Class • Admin Account Class
3	IT21272936	SILVA S.K.D. T	• Verb Analysis • Employee Class • Employee Account Class
4	IT21338120	WIJESINGHE S.A.A. K	• CRC Cards drawing • Patient Class • Patient Account Class
5	IT21292040	SASANKA M.W.K. L	• Case Scenario • Admin Class • Blood Camp Class