BLOOD BANK MANAGEMENT SYSTEM

SOFTWARE ENGINNERING PROJECT REPORT

submitted by

S. LOKESH KUMAR

(18BCE1180)

in partial fulfillment for the award of the completion of

SOFTWARE ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

NOVEMBER & 2019

ACKNOWLEDGEMENT

I am thankful to my teacher for giving us support and encouragement during the completion of this project. I am also thankful to VIT University for giving me the opportunity for working on this project.

I thank to my faculty Dr.Justus for guiding me during the completion of this project.

TABLE OF CONTENTS

I. Purpose

The purpose of the blood bank management system is to simplify and automate the process of searching for blood in case of emergency and maintain the records of blood donors, recipients, blood donation programs and blood stocks in the bank.

II. Background

A) Problem Statement

At present, the public can only know about the blood donation events through conventional media means such as radio, newspaper or television advertisements. There is no information regarding the blood donation programs available on any of the portal.

The current system that is using by the blood bank is manual system. With the manual system, there are problems in managing the donors' records. The records of the donor might not be kept safely and there might be missing of donor's records due to human error or disasters. Besides that, errors might occur when the staff keeps more than one record for the same donor.

There is no centralized database of volunteer donors. So, it becomes really tedious for a person to search blood in case of emergency. The only option is to manually search and match donors and then make phone calls to every donor.

There is also no centralized database used to keep the donors' records. Each bank is having their own records of donors. If a donor makes donation in different hospital, no previous records can be traced except if the donor brings along the donation certificate. Hence, the donor is considered to be a first-timer if they make blood donation in a new place.

Without an automated management system, there are also problems in keeping track of the actual amount of each and every blood type in the blood bank. In addition, there is also no alert available when the blood quantity is below its par level or when the blood in the bank has expired.

B) Project Goals and Objectives

The goals and objectives of the Blood Bank Management System are as follows:

1. To provide a means for the blood bank to publicize and advertise blood donation programs.
2. To allow the probable recipients to make search and match the volunteer donors, and make request for the blood.
3. To provide an efficient donor and blood stock management functions to the blood bank by recording the donor and blood details.
4. To improve the efficiency of blood stock management by alerting the blood bank staffs when the blood quantity is below it par level or when the blood stock has expired.
5. To provide synchronized and centralized donor and blood stock database.
6. To provide immediate storage and retrieval of data and information.

C) Product Description

The system that is going to be developed is Blood Bank Management System (BBMS). This is a database application system that is to be used by the blood banks or blood centers as a means to advertise the nationwide blood donation events to the public and at the same time allow the public to make reservation and request for the blood.

The system keeps the record of all the donors, recipients, blood donation programs, rejected bloods.

This system also has the ability to keep track of the donor's donation records and the blood stock in the blood bank. This project intends to computerize the blood and donor management system in a blood bank in order to improve the record management efficiency due to the grown size of records of data.

III. Scope

The system is used for maintaining all the process and activities of blood bank management system.

The system can be extended to be used for maintaining records of hospital, organ donation and other similar sectors. While developing the system, there shall be space for further modification. There shall be a proper documentation so that further enhancement becomes easy.

As a whole the system is focused to work with blood bank management system and on additional modification it can be also used as management systems of similar organizations.

A) Stakeholders

1) System Owner: The Blood Bank

2) System Users:

- Administrators: has full privilege on the system's functions
- Staffs of Blood Bank: has privilege on the system's functions as assigned by the administrator

B) Data

1. Data about Donor and recipients

  - ➢ Donor/ Recipient Id
  - ➢ Name
  - ➢ Date of Birth
  - ➢ Sex
  - ➢ Blood Group
  - ➢ Address
  - ➢ Contact Number
  - ➢ Email Address

2. Donation program

- Organizer
- Event Id
- Date of Donation
- Venue
- Volunteers
- Amount of blood collected

3. Blood

- Blood Id
- Blood Group
- Date of collection
- Expiry date

4. Staff

- User Name
- Password

C) Processes

- **Login**
  The system provides security features through username-password matching where only authorized user can access the system with different authorization level.

- **Advertisements of blood donation event**
  This function allows the blood bank staff to publicize the blood donation events online. The public can view the venue and time of the blood donation programs to be held.

- **Donor Profile Registration**
  This allows healthy public to register as volunteer donor.

- **Online Request for fresh blood**
  This allows the probable recipients to make online request to the donor. After the request has been filed donors are matched and the request is sent via SMS with necessary details.

- **Blood Stock Management**
  The blood bank staffs can manage the blood stock starting from the blood collection, to blood screening, processing, storage, transference and transfusion through this system. Each process or work-flow can be traced from the database. The system will also raise alert to the staff whenever the blood quantity is below its par level or when the blood in stock has expired.

- **Donor/Recipient Management**
  The records of all donors/recipient and their history are kept in one centralized database and thus reducing duplicate data in the database. The record of donation is maintained by the system.

- **Reporting**
  The system is able to generate pre-defined reports such as the list of donors, recipients, staffs, the blood quantity in the bank and charts.

Project Approach

Route:

- Problem Identification
- System Design
- System Building
- Testing and Implementation

Deliverables:

The main deliverables of the projects are as follows:

- Requirement Specification
  - ❖ Use-Case Model
- Analysis Model will be used to show the realization of all use-cases conceptually
- Design specification will be used to specify the design for the realization of all use-cases including class diagrams
- Implementation model
  - ❖ Agile
- Documentation and Manual

Managerial Approach

- Team Building Consideration:
  - ❖ Each of the team member will be given a job
  - ❖ The work division shall be on the basis of expertise
  - ❖ The progress shall be synchronized on weekly basis

- Training requirements:
    - ❖ C# (C Sharp)
    - ❖ Sql
    - ❖ Visual Studio Code

Meeting Schedules

The meeting of the working team members shall be done on weekly basis. This shall follow an objective of synchronized working and progress description.

Reporting Methods

Every one should prepare a report on the module upon which he is working. This report shall be used for documentation and manual preparation.

FUNCTIONAL REQUIREMENTS:

REQUIREMENT ID	DESCRIPTION
F1	Login
F1.1	Validation of credentials
F2	Donor registration
F2.1	Donor details
F3	Blood request
F3.1	Request details
F3.2	Check data base for availability.
F3.3	Process request
F4	Blood management
F4.1	Availability of blood
F4.2	Shortage alert
F4.3	Donation details
F5	Camp info
F5.1	Camp details insertion
F5.2	Camp details view
F6	Logout

Description of functional requirements

Login module:

Fid	F1.1 Validation of credentials
Input	Login credentials
Output	Logged in
Action	Checks whether the credentials are valid
Pre-condition	Must not be logged in
Post condition	Login successful

Donor registration module:

Fid	F2.1 Donor details
Input	Name,age,sex,blood group,contact no
output	Donor registration successful
Action	Add donor to database
Pre-condition	Check whether user is not already registered.
Post condition	Add the donor to database for later use.

Blood request module:

Fid	F3.1 Recipient's details
Input	Name,age,sex,blood group,contact no
Output	Blood request successful
Action	Process request
Pre-condition	Check whether request is not duplicated.
Post-condition	Request forwarded to next stage

Fid	F3.2 Check database availability
Input	Blood request
Output	Available or not available
Action	Check data base
Pre-condition	Check whether the request is valid.
Post-condition	User get to know if blood is available or not.

Fid	F3.3 Process request
Input	Blood request
Output	Blood donation successful
Action	Proceed for blood donation.
Pre-condition	Check if request is valid
Post-condition	Donate blood and maintain database

Blood management module:

Fid	F4.1 Check availability
Input	Blood group
Output	Blood availability
Action	Check data base.
Pre-condition	Ensure user is valid
Post-condition	User gets to know the availablity of blood.

Fid	F4.2 Shortage alert

Input	Data base
Output	Blood shortage alert
Action	Alerts the user when blood stock is less than minimum quantity
Pre-condition	Check if blood available
Post-condition	User is alerted when blood shortage occurs

Fid	F4.3 Donation details
Input	Data base
Output	Blood donation details
Action	Checks the data base for donation details
Pre-condition	Check if the data base is correct
Post-condition	User gets to know the donation details

Camp module:

Fid	F5.1 Camp details insertion
Input	Organization,venue,date,timing

Output	Details added successfully
Action	Add details to data base
Pre-condition	Check if data is already present
Post-condition	User can enter the camp details.

Fid	F5.2 Camp details view
Input	View camps key
Output	Organization,venue,date,timing
Action	Check data base for camp details
Pre-condition	Check if data is already present
Post-condition	User get to know the camp details

Logout module:

Fid	F6.1 Logout
Input	Logout key
Output	Logged out succesfully

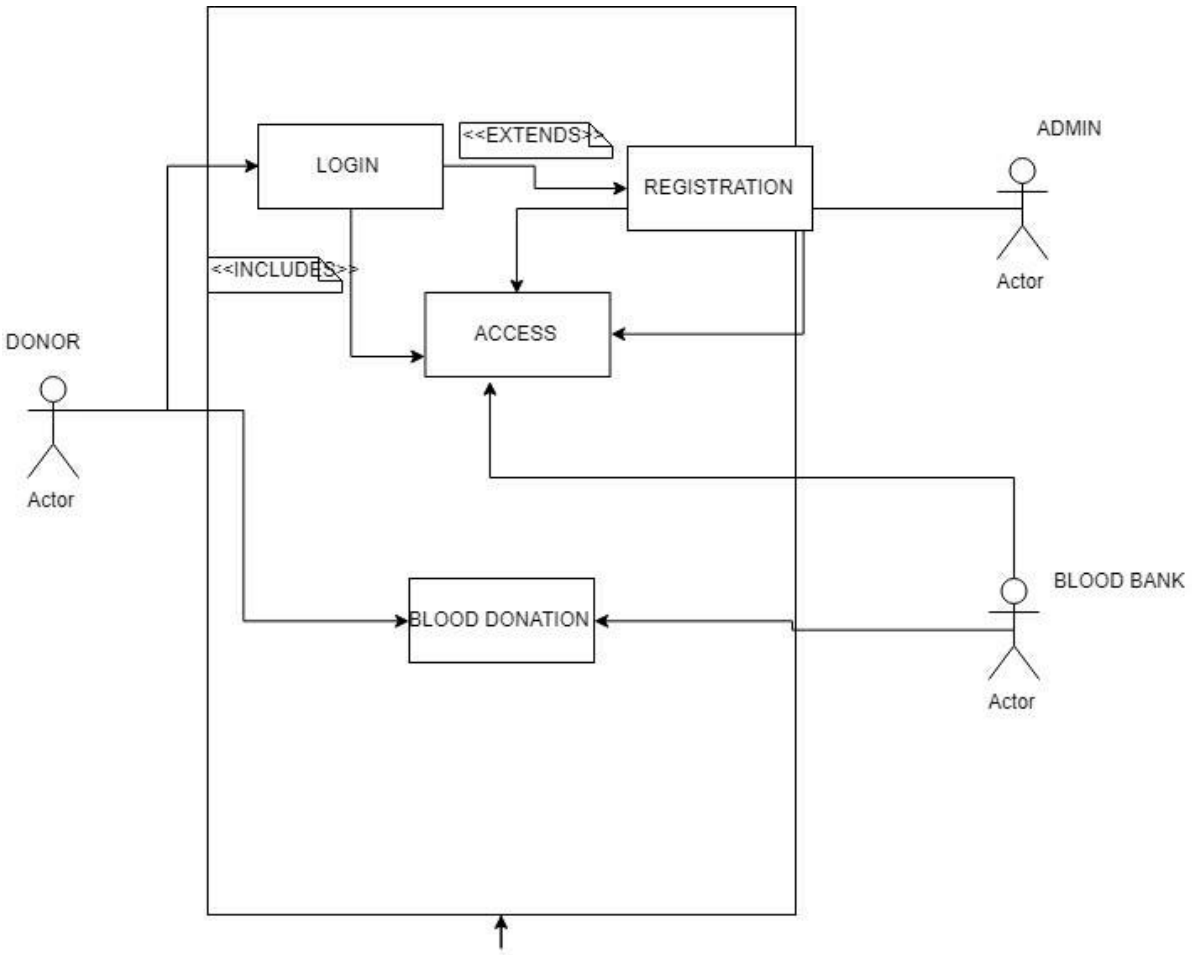
Action	Logs out of the session
Pre-condition	Checks if user is logged in
Post-condition	User logs out of the portal

Non Functional Requirements:

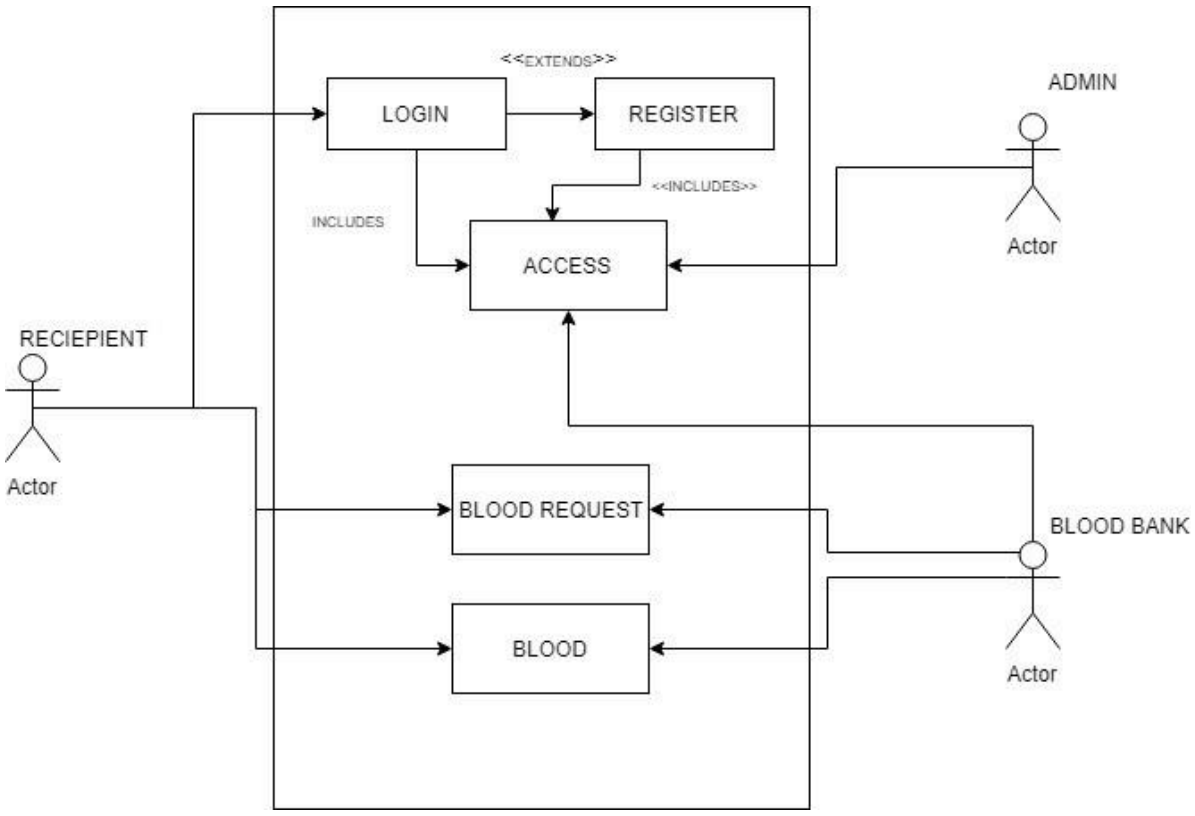
- *Friendly and simply interface*
- *The color is elegant, not flashy*
- *Language is English*
- *Scalability system*
- *Security of the system*
- *Performance*
- *User satisfaction*
- *Backup*
- *Availability*

UML DIAGRAMS:

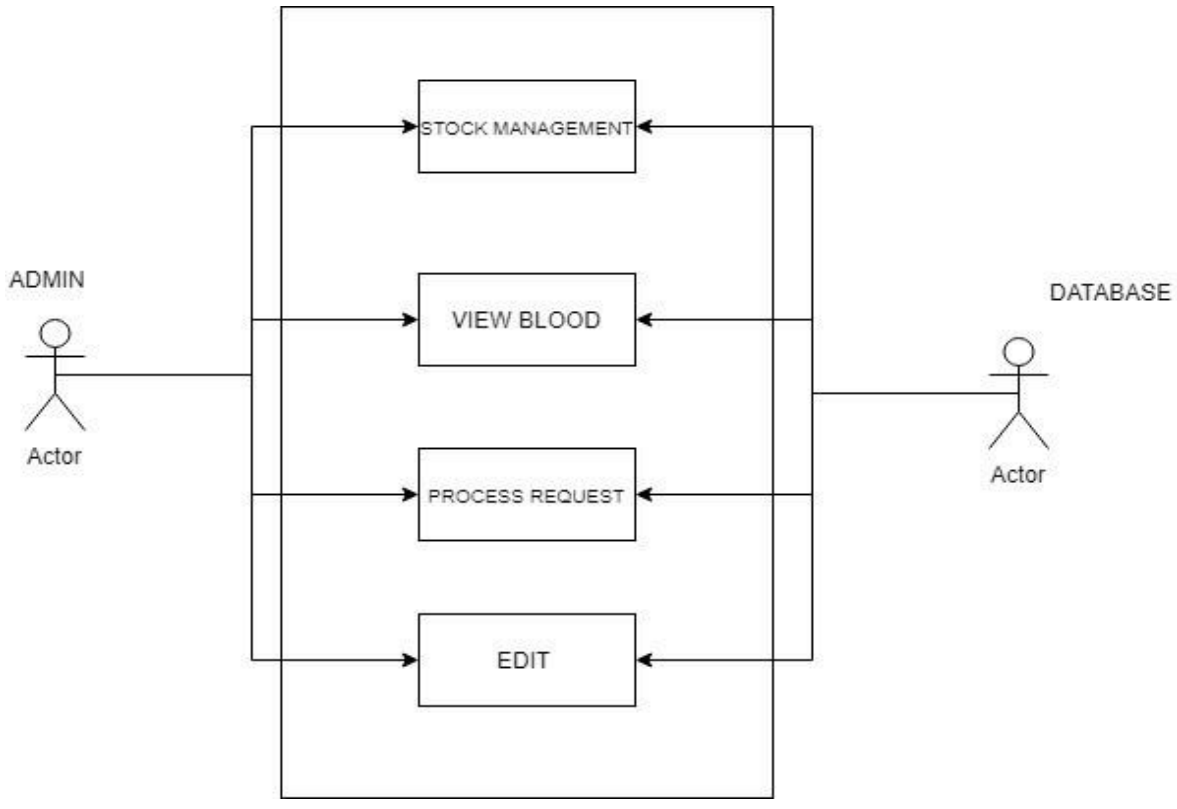
1.Donor Module:



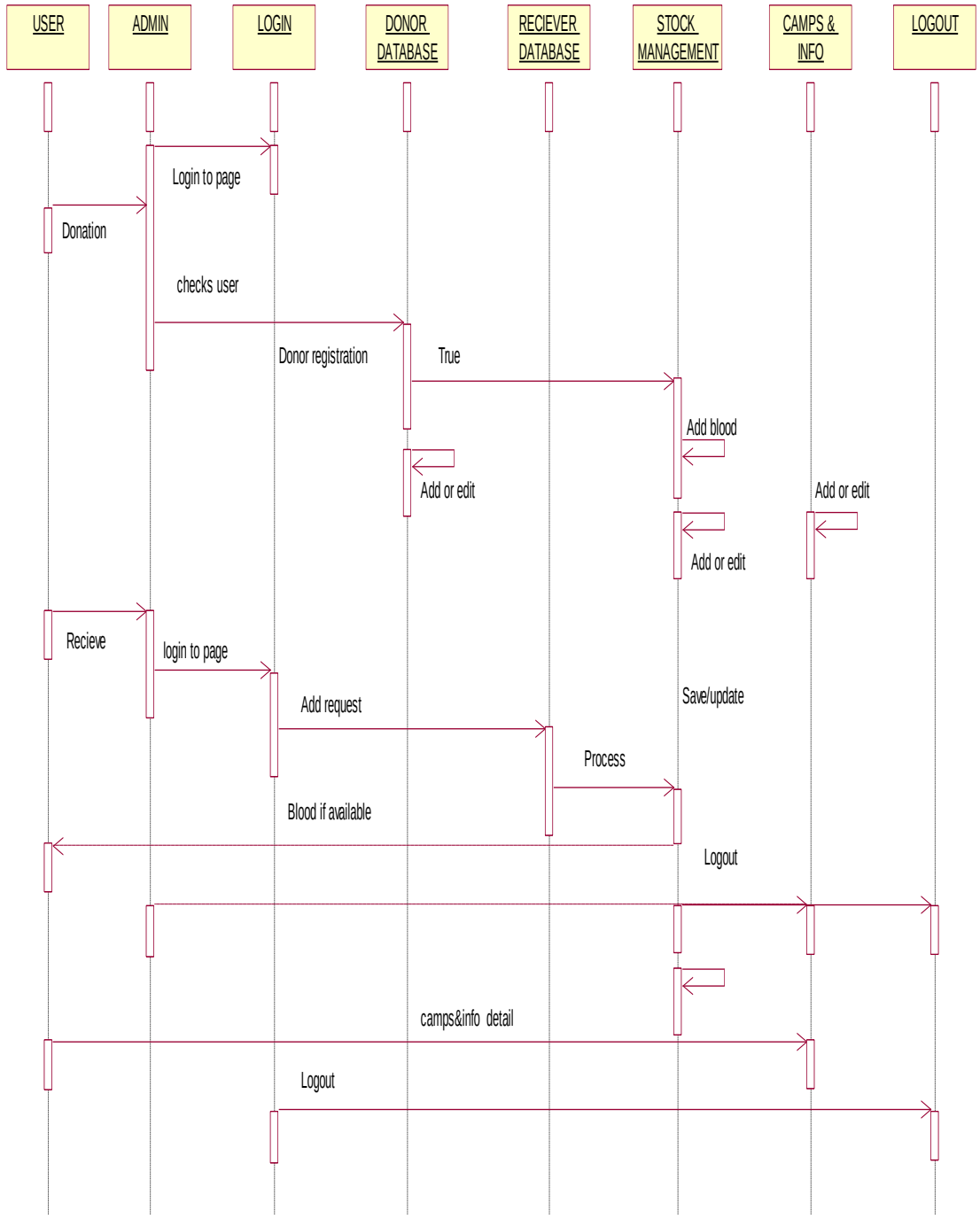
2. Recipient Module:



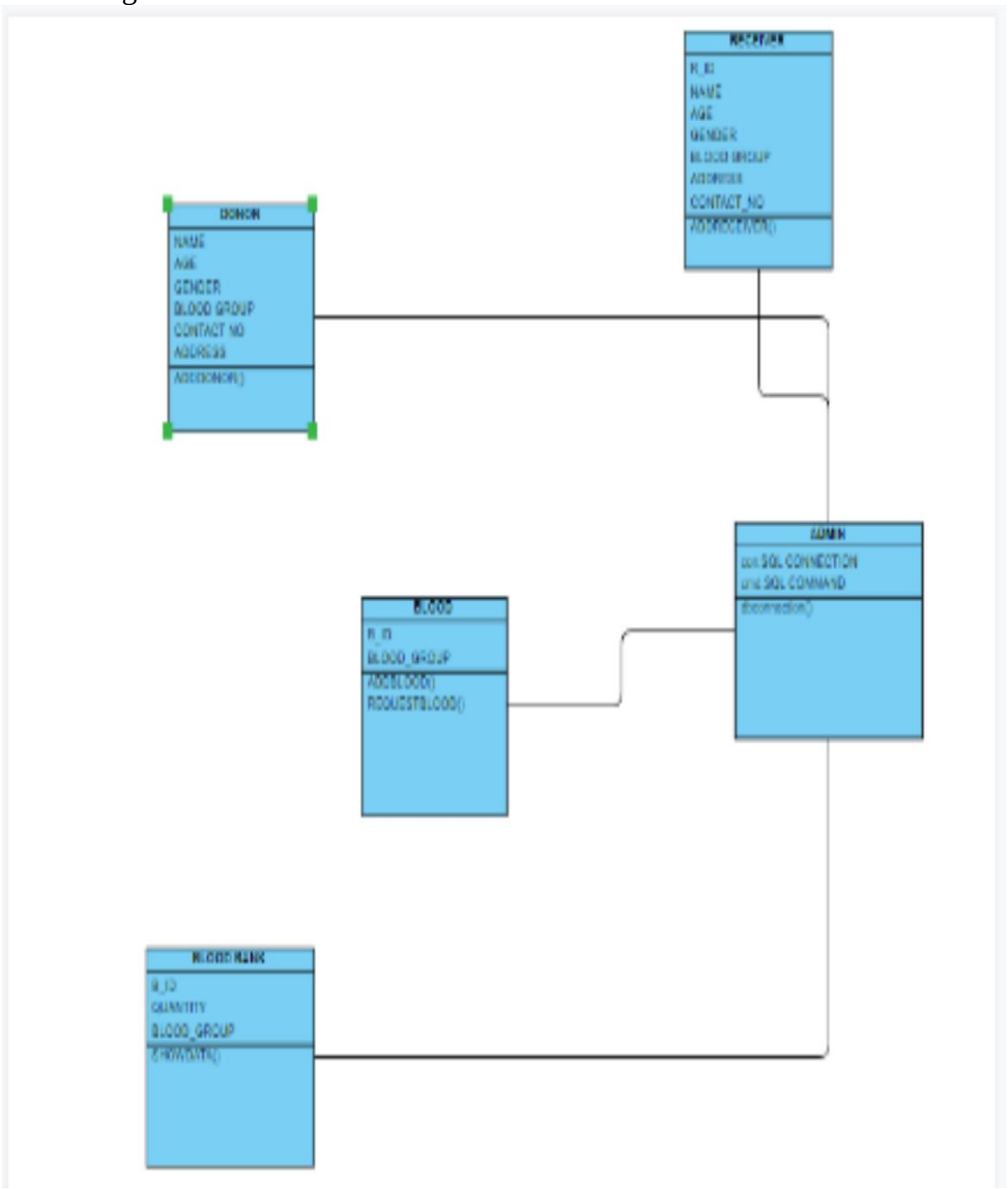
3. Admin Module:



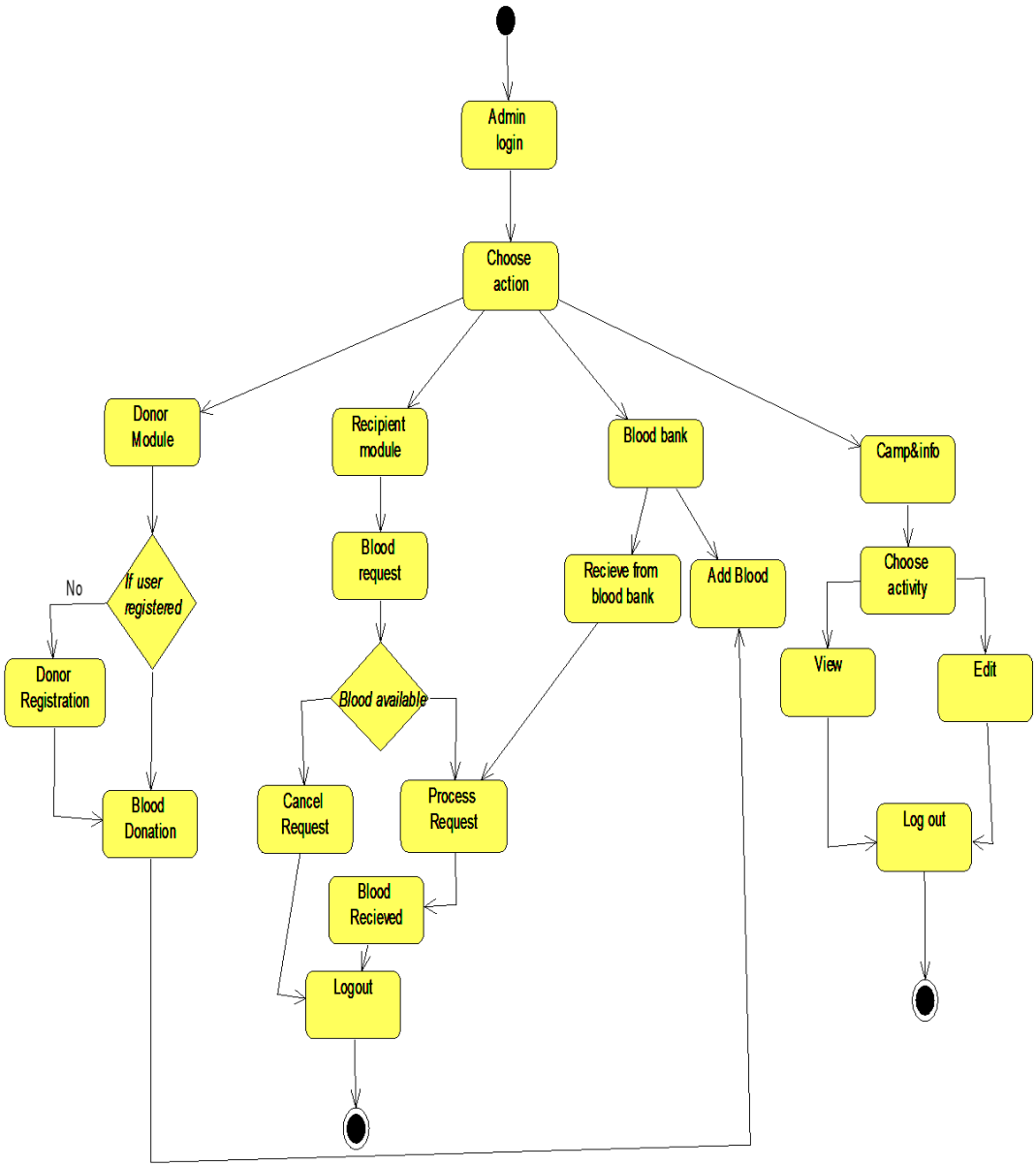
Sequence Diagram:



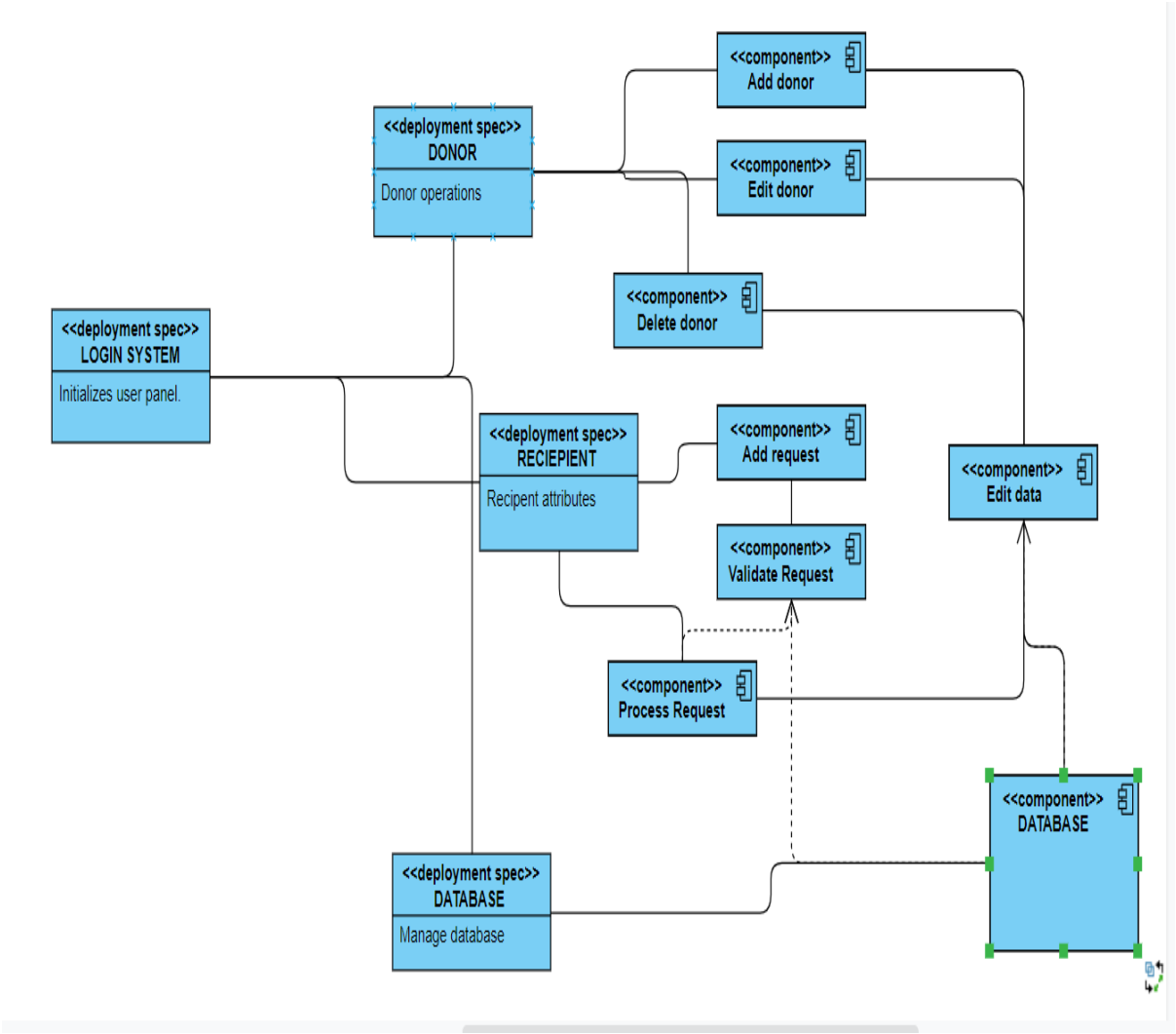
Class Diagram:



Activity Diagram:



DEPLOYNMENT DIAGRAM:



SAMPLE CODE

USER INTERFACE:

1.USER:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace WindowsFormsApp1
{
    public partial class USER : Form
    {
        public object DONOR_REGISTARTION { get; private set; }

        public USER()
        {
            InitializeComponent();
        }

        private void DONOR_Load(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(HOME.Instance))
            {
                contentpannel.Controls.Add(HOME.Instance);
                HOME.Instance.Dock = DockStyle.Fill;
                HOME.Instance.BringToFront();
            }
            else
            {
                HOME.Instance.BringToFront();
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }
```

```
private void panel1_Paint(object sender, PaintEventArgs e)
{

}



private void contentpannel_Paint(object sender, PaintEventArgs e)
{

}

private void button4_Click(object sender, EventArgs e)
{
    if (!contentpannel.Controls.Contains(DONOR_REGISTRATION.Instance))
    {
        contentpannel.Controls.Add(DONOR_REGISTRATION.Instance);
        DONOR_REGISTRATION.Instance.Dock = DockStyle.Fill;
        DONOR_REGISTRATION.Instance.BringToFront();
    }
    else
    {
        DONOR_REGISTRATION.Instance.BringToFront();
    }
}

private void button5_Click(object sender, EventArgs e)
{
    if (!contentpannel.Controls.Contains(RECEIVER.Instance))
    {
        contentpannel.Controls.Add(RECEIVER.Instance);
        RECEIVER.Instance.Dock = DockStyle.Fill;
        RECEIVER.Instance.BringToFront();
    }
    else
    {
        RECEIVER.Instance.BringToFront();
    }
}

private void button6_Click(object sender, EventArgs e)
{
    if (!contentpannel.Controls.Contains(BLOOD_BANK.Instance))
    {
        contentpannel.Controls.Add(BLOOD_BANK.Instance);
        BLOOD_BANK.Instance.Dock = DockStyle.Fill;
        BLOOD_BANK.Instance.BringToFront();
    }
    else
    {
        BLOOD_BANK.Instance.BringToFront();
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
```

```
            Form1 obj = new Form1();
            this.Hide();
            obj.Show();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(CAMPS.Instance))
            {
                contentpannel.Controls.Add(CAMPS.Instance);
                CAMPS.Instance.Dock = DockStyle.Fill;
                CAMPS.Instance.BringToFront();
            }
            else
            {
                CAMPS.Instance.BringToFront();
            }

        }


        private void button9_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(HOME.Instance))
            {
                contentpannel.Controls.Add(HOME.Instance);
                HOME.Instance.Dock = DockStyle.Fill;
                HOME.Instance.BringToFront();
            }
            else
            {
                HOME.Instance.BringToFront();
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            Form0 obj = new Form0();
            this.Hide();
            obj.Show();

        }

        private void button8_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(ABOUT_US.Instance))
            {
                contentpannel.Controls.Add(ABOUT_US.Instance);
                ABOUT_US.Instance.Dock = DockStyle.Fill;
                ABOUT_US.Instance.BringToFront();
            }
            else
            {
                ABOUT_US.Instance.BringToFront();
            }
        }
    }
}
```

2.ADMIN:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class ADMIN : Form
    {
        public ADMIN()
        {
            InitializeComponent();
        }

        private void panel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(DONORINFO.Instance))
            {
                contentpannel.Controls.Add(DONORINFO.Instance);
                DONORINFO.Instance.Dock = DockStyle.Fill;
                DONORINFO.Instance.BringToFront();
            }
            else
            {
                DONORINFO.Instance.BringToFront();
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form0 obj = new Form0();
            this.Hide();
            obj.Show();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form1 obj = new Form1();
            this.Hide();
            obj.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void contentpaNnel_Paint(object sender, PaintEventArgs e)
```

```
        {

        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(BLOOD_BANK.Instance))
            {
                contentpannel.Controls.Add(BLOOD_BANK.Instance);
                BLOOD_BANK.Instance.Dock = DockStyle.Fill;
                BLOOD_BANK.Instance.BringToFront();
            }
            else
            {
                BLOOD_BANK.Instance.BringToFront();
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(REQUESTS.Instance))
            {
                contentpannel.Controls.Add(REQUESTS.Instance);
                REQUESTS.Instance.Dock = DockStyle.Fill;
                REQUESTS.Instance.BringToFront();
            }
            else
            {
                REQUESTS.Instance.BringToFront();
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            if (!contentpannel.Controls.Contains(CAMPDETAILS.Instance))
            {
                contentpannel.Controls.Add(CAMPDETAILS.Instance);
                CAMPDETAILS.Instance.Dock = DockStyle.Fill;
                CAMPDETAILS.Instance.BringToFront();
            }
            else
            {
                CAMPDETAILS.Instance.BringToFront();
            }
        }

        private void ADMIN_Load(object sender, EventArgs e)
        {

        }
    }
}
```

DONAR REGISTRATION MODULE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApp1
{
    public partial class DONOR_REGISTRATION: UserControl
    {
        private static DONOR_REGISTRATION _instance;
        public static DONOR_REGISTRATION Instance
        {
            get
            {
                if (_instance == null)
                {
                    _instance = new DONOR_REGISTRATION();
                }
                return _instance;
            }
        }
        public DONOR_REGISTRATION()
        {
            InitializeComponent();
        }



        SqlConnection CON = new SqlConnection(@" Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\DONOR.mdf;Integrated
Security=True");



        private void REGISTER_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = new SqlCommand("Adddonor", CON);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@NAME", NAME.Text);
            cmd.Parameters.AddWithValue("@AGE", AGE.Text);
            cmd.Parameters.AddWithValue("@SEX", SEX.Text);
            cmd.Parameters.AddWithValue("@BLOOD_GRP", BLOOD_GRP.Text);
            cmd.Parameters.AddWithValue("@ADDRESS", ADDRESS.Text);
            cmd.Parameters.AddWithValue("@CONTACT_NO", CONTACT_NO .Text);
            CON.Open();
            try
            {
                cmd.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                MessageBox.Show("<<<INVALID SQL OPERATION>>>: \n" + ex);
            }
            MessageBox.Show("SUCCESSFULLY ADDED");
            CON.Close();

        }



        private void button1_Click(object sender, EventArgs e)
        {
```

```
            NAME.Text = "";
            AGE.Text = "";
            CONTACT_NO.Text = "";
            SEX.Text = "";
            BLOOD_GRP.Text = "";
            ADDRESS.Text = "";
        }

        private void CUS_ID_TextChanged(object sender, EventArgs e)
        {

        }
        private void DONOR_REGISTRATION_Load(object sender, EventArgs e)
        {


        }
    }
}
```

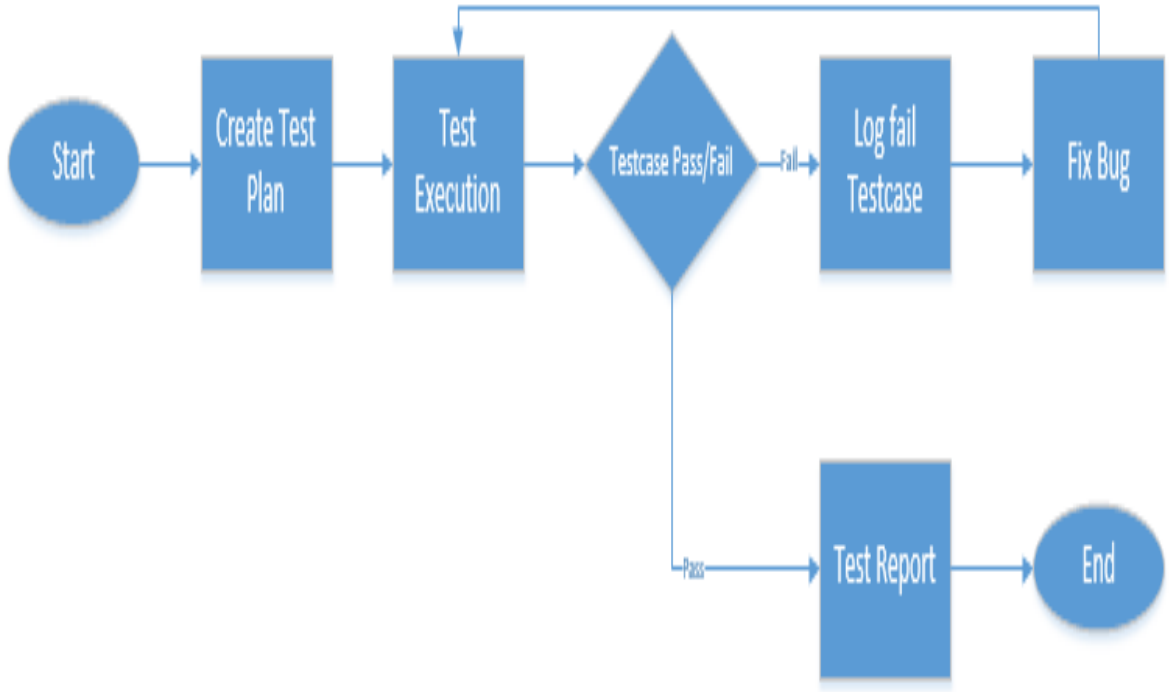
LOGIN MODULE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            label4.Hide();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string  upass = "adminblood", name, pass;
            name = textBox1.Text;
            pass = textBox2.Text;
            if (name.Equals("18bce1180") || name.Equals("18bce1273") &&
pass.Equals(upass))
            {
                label4.Hide();
```

```
            ADMIN obj = new ADMIN();
            this.Hide();
            obj.Show();
        }
        else
        {
            label4.Show();
            MessageBox.Show("Unsuccessful Login");
        }
    }

    private void textBox2_TextChanged(object sender, EventArgs e)
    {

    }

    private void label4_Click(object sender, EventArgs e)
    {

    }

    private void button2_Click(object sender, EventArgs e)
    {
        Environment.Exit(0);
    }

    private void button3_Click(object sender, EventArgs e)
    {
        this.WindowState = FormWindowState.Minimized;
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Form0 obj = new Form0();
        this.Hide();
        obj.Show();
    }}}
```

TESTING

Test plan:



BLACK BOX TESTING:

DONOR MODULE:

TEST CASE NO	FEATURE OF THE TEST	EVENT	EXPECTED RESULT
1	Check the credentials of the user	When user enters correct values and clicks on login	If the user is valid t then the user is logged in and display blood bank page.
		When the user invalid credentials	Word invalid credentials is printed
2	Redirecting to donor registration.	Clicks on donor registration.	Display donor registration form.
3.	Redirecting to view donor.	Clicks on view donor.	Display donor data form.
4.	Search donors	Types the correct name of the donor and clicks on search button	Donor information is displayed
		Types incorrect name of the donor and clicks on search button	Donor information is not displayed
5	Search donor by blood type.	Chooses the category and clicks on the search	Donors in the same category are displayed

RECIPIENT MODULE:

Test case no	Feature of the test	Event	Expexted result
1	Check the credentials of the user.	When user enters correct values and clicks on login	Login to system.
		When the user invalid credentials	Word invalid is printed
2	Redirecting to add request to blood.	Click on add request.	Add request window is displayed.
3	Redirecting to validate request.	After adding request in request window	Display whether request is duplicate or not.
4	Redirect to request status.	Click on view request.	Display request status.
5	Redirect to delete request	Click on delete request.	Display delete request screen
6	Delete request for blood.	Click on request and press Delete button	Display screen deleted request.

DATABASE MODULE:

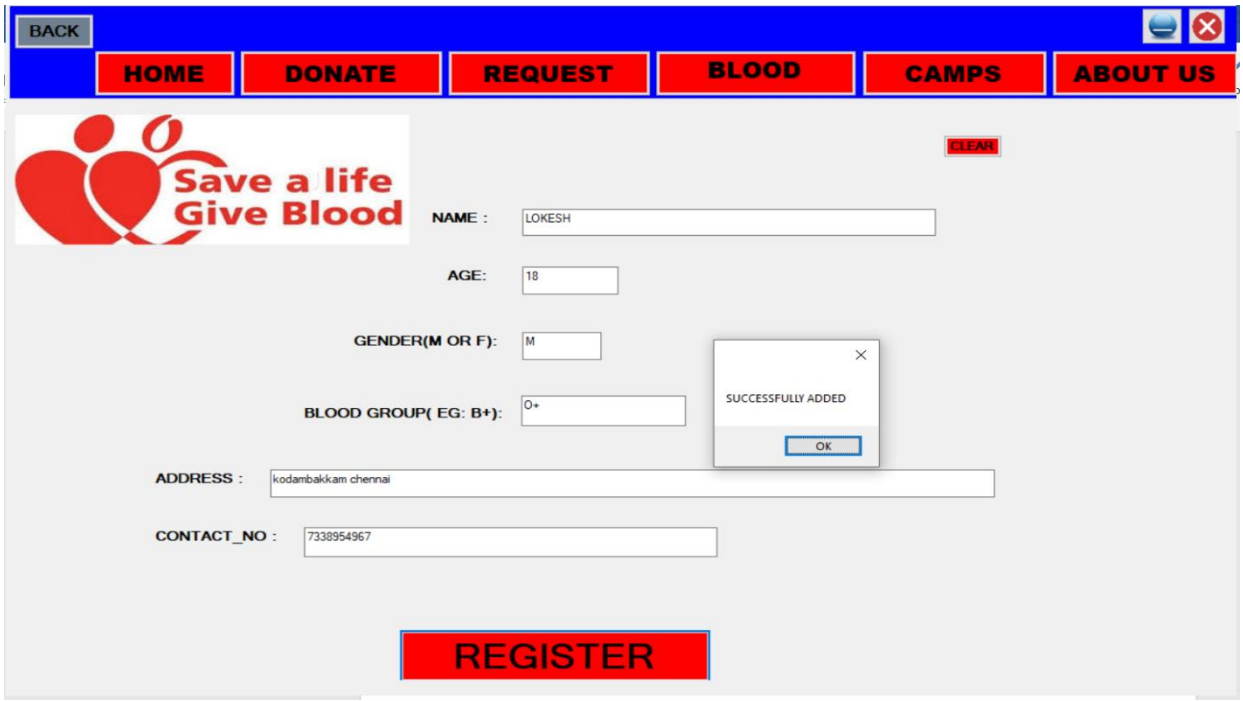
Test case no	Feature of the test	Event	Expexted result
1	Check the credentials of the user.	When user enters correct values and clicks on login	Login to system.
		When the user invalid credentials	Word invalid is printed
2	Redirecting to add data.	Click on add data.	Add data window is displayed.
3	Redirecting to edit data.	Click on edit data button.	Display edit database form.
5	Redirect to delete data.	Click on delete data.	Display delete data

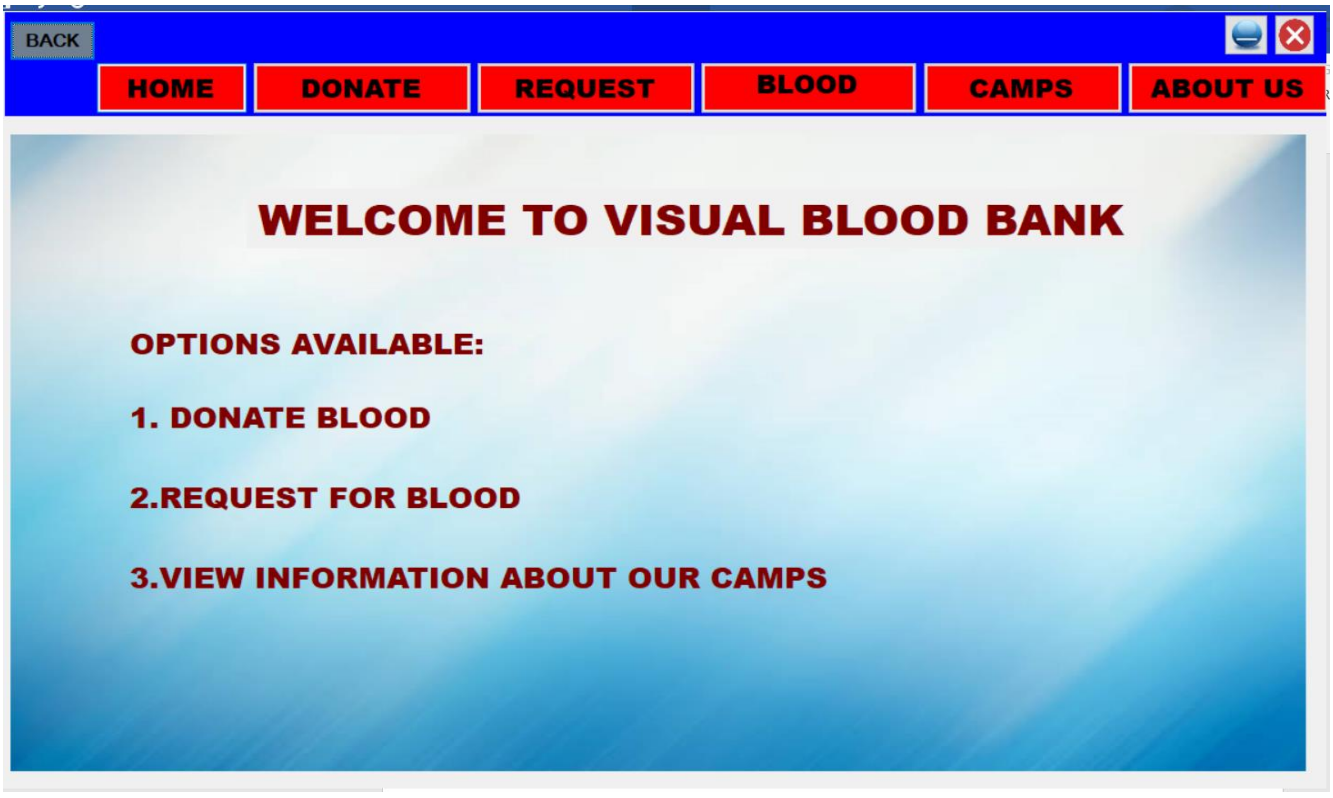
			form.
6	Delete data from database.	Click on data and press Delete button	Display screen deleted data.

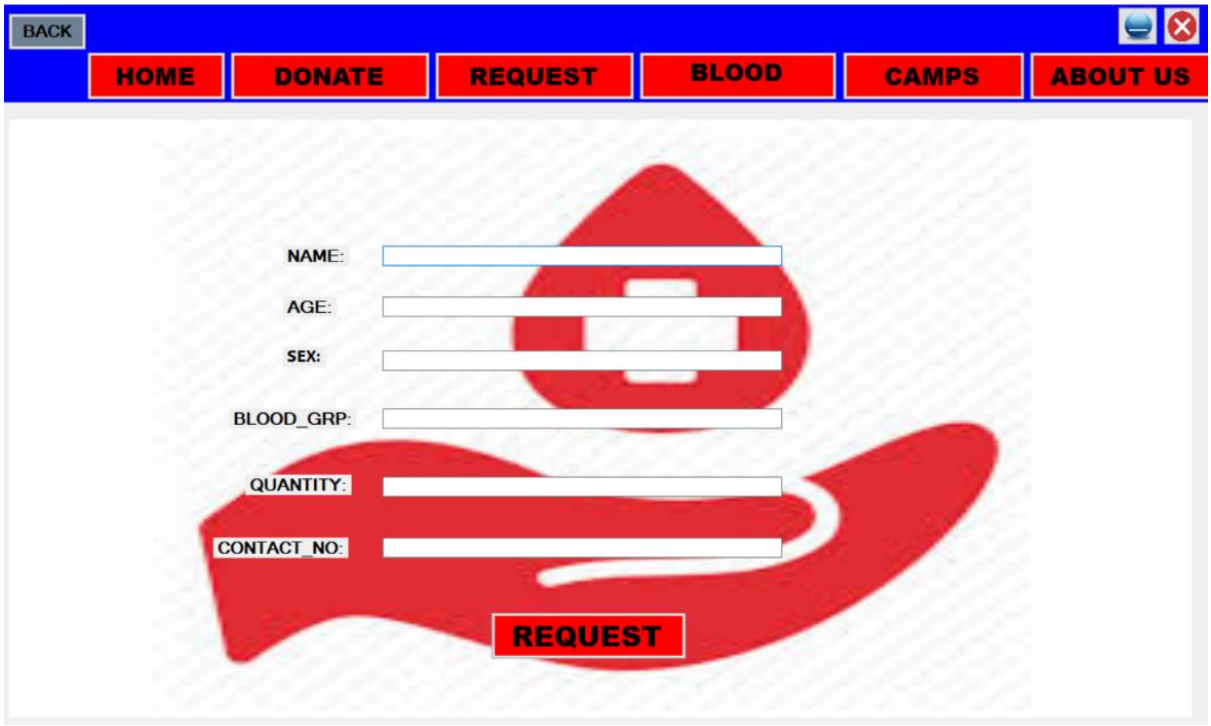
WHITE BOX TESTING:

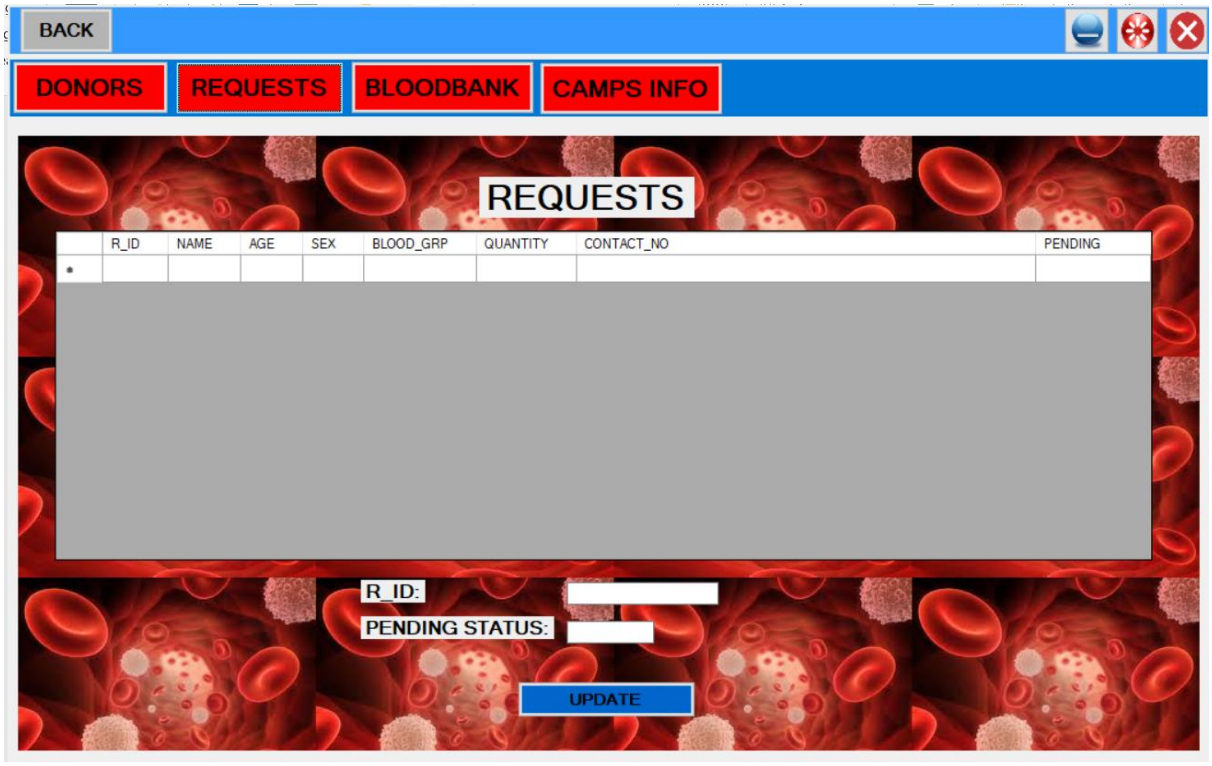
	C	D	E	F	G	H
1	Test Requirement	T.No	Test Case	Expected Result	Actual Result	Status
2	The user should be a authorized person	T1	Correct Username&Password	Logged in	Logged in	Pass
3		T2	Correct Username& Wrong Password	Invalid credentials	Invalid credentials	Pass
4		T3	Incorrect Username & Correct password	Invalid credentials	Invalid credentials	Pass
5		T4	Incorrect Username & Incorrect password	Invalid credentials	Invalid credentials	Pass
6						
7						
8	There should be no duplicate donors	T1	Donor name and data are already available in the database	User Exists	Registered Succesfully	Fail
9		T2	Donor name and data are not available in the database	Registered Succesfully	Registered Succesfully	Pass
10						
11						
12	There should be no irrelevant data	T1	Relevant input of alpha and numeric values in the required place	Registered Succesfully	Registered Succesfully	Pass
13		T2	Irrelevant input of alpha and numeric values in the required place	Invalid Data	Invalid Data	Pass
14						
15						
16	Test whether there is no duplicate requests	T1	Blood request already available in the database	Request Exists	Requests exists	Pass
17		T2	Blood request not available in the database	Request Succesfull	Request Succesfull	Pass
18						
19	Test whether the donor is present	T1	Correct donor id	User found	User found	Pass
20		T2	Incorrect donor id	User not found	User not found	Pass
21						
22	Test for adding blood request	T1	Given request informations are not already present in the database.	Request added	Request added	Pass
23		T2	Given request informations are already present in the database.	Duplicate request	Duplicate request	Pass
24						
25	Test for search donor	T1	Given donor name or blood group present in the database.	Display donor information	Display donor information	Pass
26		T2	Given donor name or blood group not present in the database.	Display no data found	Display no data found	Pass
27						
28	Test for add camps	T1	Given camp informations are already present in the database.	Camp already registered	Camp already registered	Pass
29		T2	Given camp informations are not already present in the database.	Camp registered	Camp registered	Pass
30						

USER INTERFACE

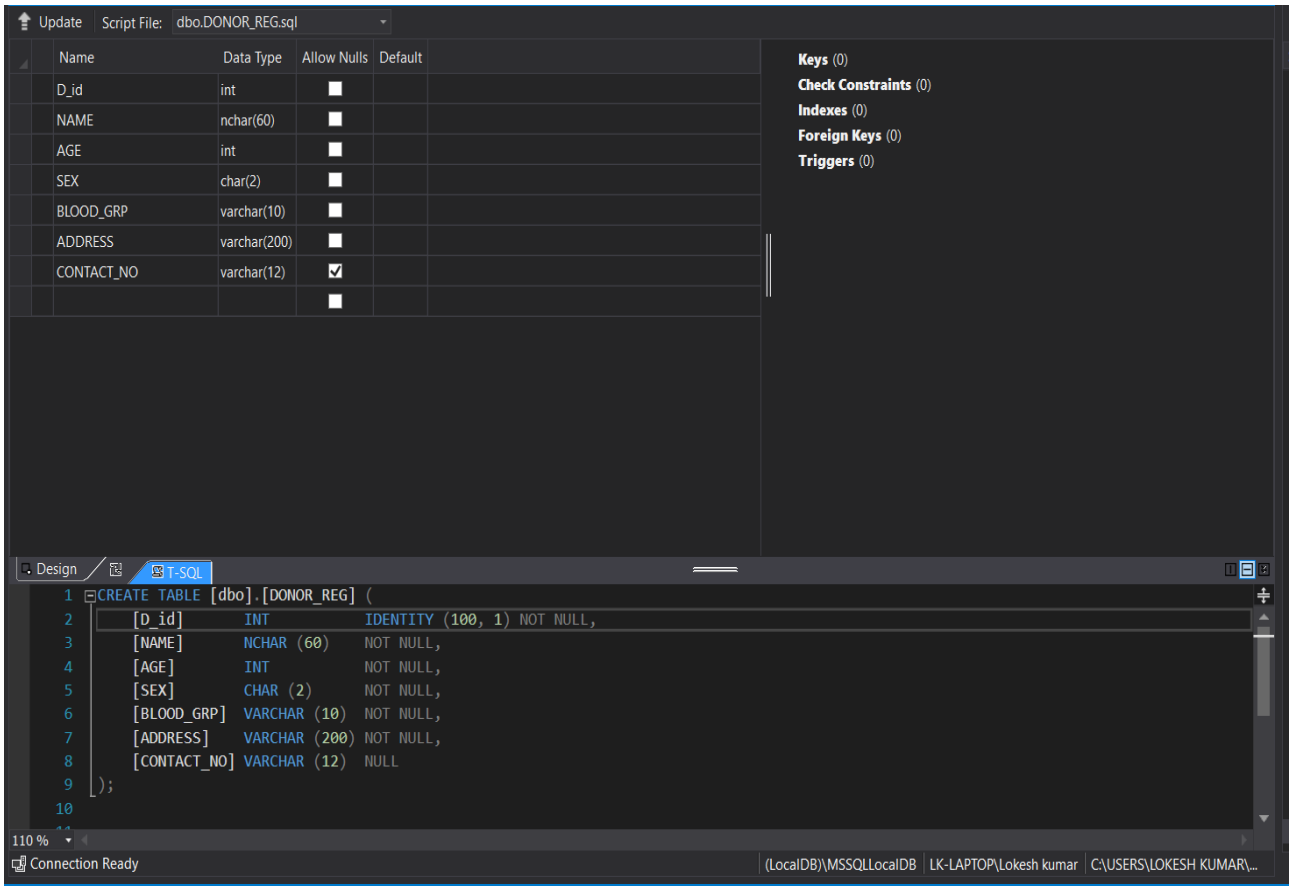


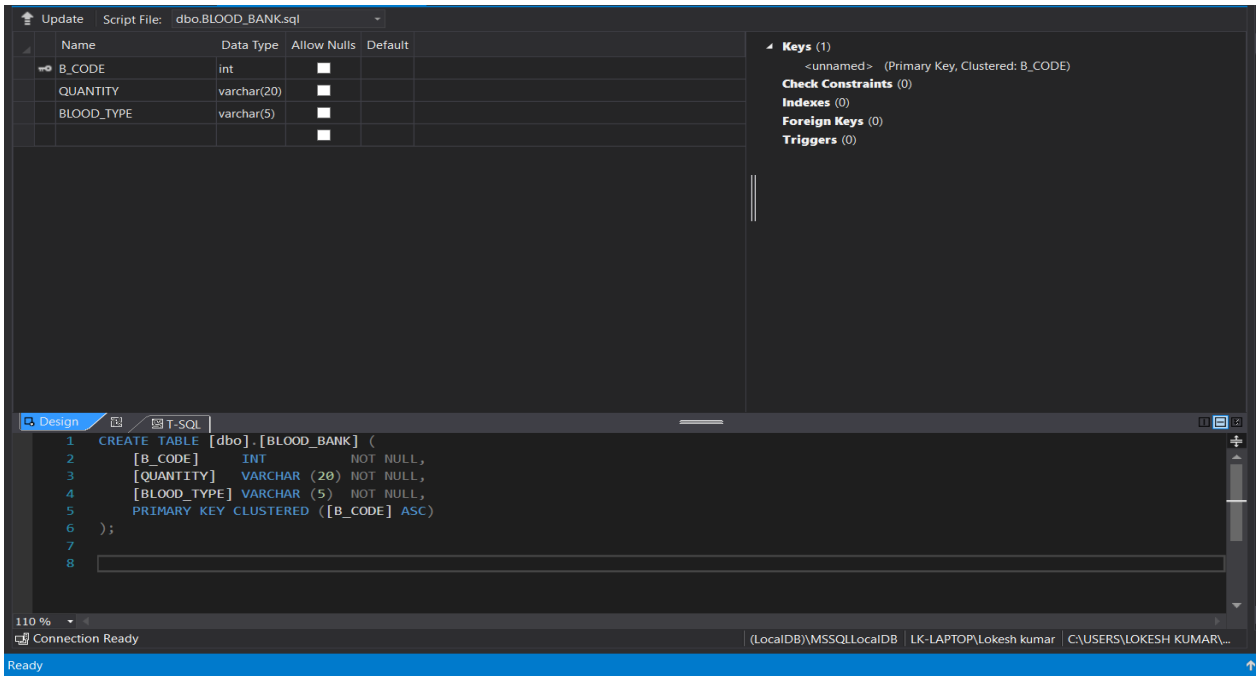



BACK
HOME
DONATE
REQUEST
BLOOD
CAMPS
ABOUT US
NAME:
AGE:
SEX:
BLOOD_GRP:
QUANTITY:
CONTACT_NO:
REQUEST


BACK
DONORS
REQUESTS
BLOODBANK
CAMPS INFO
REQUESTS
R_ID
NAME
AGE
SEX
BLOOD_GRP
QUANTITY
CONTACT_NO
PENDING
R_ID:
PENDING STATUS:
UPDATE

DATABASE:


Update
Script File: dbo.DONOR_REG.sql
Name
Data Type
Allow Nulls
Default
D_id
int
NAME
nchar(60)
AGE
int
SEX
char(2)
BLOOD_GRP
varchar(10)
ADDRESS
varchar(200)
CONTACT_NO
varchar(12)
Keys (0)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)
Design
T-SQL
CREATE TABLE [dbo].[DONOR_REG] (
[D_id] INT IDENTITY (100, 1) NOT NULL,
[NAME] NCHAR (60) NOT NULL,
[AGE] INT NOT NULL,
[SEX] CHAR (2) NOT NULL,
[BLOOD_GRP] VARCHAR (10) NOT NULL,
[ADDRESS] VARCHAR (200) NOT NULL,
[CONTACT_NO] VARCHAR (12) NULL
);
110 %
Connection Ready
(LocalDB)\MSSQLLocalDB
LK-LAPTOP\Lokesh kumar
C:\USERS\LOKESH KUMAR\...


Update
Script File: dbo.BLOOD_BANK.sql
Name
Data Type
Allow Nulls
Default
B_CODE
int
QUANTITY
varchar(20)
BLOOD_TYPE
varchar(5)
Keys (1)
<unnamed> (Primary Key, Clustered: B_CODE)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)
Design
T-SQL
CREATE TABLE [dbo].[BLOOD_BANK] (
[B_CODE] INT NOT NULL,
[QUANTITY] VARCHAR (20) NOT NULL,
[BLOOD_TYPE] VARCHAR (5) NOT NULL,
PRIMARY KEY CLUSTERED ([B_CODE] ASC)
);
110 %
Connection Ready
(LocalDB)\MSSQLLocalDB
LK-LAPTOP\Lokesh kumar
C:\USERS\LOKESH KUMAR\...
Ready

Maintain and Future Development

- ❑ Maintain
  - ▪ Fix GUI
  - ▪ Update and fix code
- ❑ Future Development
  - • Develop more interactivity between users (share, follow,…)
  - • Adding support for other language
  - • Integrate Google location
  - • Support for searching by blood constituents