Bangladesh University of Business and Technology (BUBT)

Project Report

Project Name : Blood Bank Management System

Course Title : Software Development II

Course Code : CSE 200

Submitted To:

Anusha Aziz

Course Instructor

Department of CSE

Bangladesh University of Business and Technology (BUBT)

Submitted by

Name	ID	Intake	Section
Md. Mehedi Hasan	21225103334	49	08
Mushfiqur Rahman Pulok	21225103525	49	08
Abdullah Al Hill Baki Anim	21225103341	49	08
Faiza Khandoker Fama	21225103338	49	08
Jannatul Ferdous	21225103350	49	08

Abstract

The "Blood Bank Management System" project is a desktop-based application that is designed to enhance the operations of blood bank and facilitate efficient management of blood donations, inventory, and distribution. It allows to store information about its users and the donors of the blood bank, and also store blood bags and update the details of its inventory. This project aims to address the critical challenges faced by blood banks, such as maintaining donor records, managing blood inventory etc. Key features of the system include a user-friendly interface for blood donation registration, update / delete donor information and manually update inventory.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Currently, dengue has become an epidemic in Bangladesh. Due to this many people are suffering from anemia and platelet deficiency. Finding the necessary blood has become difficult. People often post online or message their group for necessary blood but it doesn't help them much. Our university maintains a list of donors and their details but often they are not updated. It is one of the major reasons why we have decided to make a blood bank management system. Our purpose is to make an application where admins can login into the system and update the inventory information of blood bank, add or remove details of donors. In this way we can help patients with safe blood supply when they are needed.

4.1. CONCEPT ILLUSTRATION

A blood bank system is a desktop application that helps users find blood donors and blood banks in their area. The main purpose of these apps is to make the process of blood donation more convenient and efficient, and to help save lives. Blood bank management systems can provide information about registered blood donors, including their name, address, blood group, and other medical information and also blood stock information of the bank.

1.2. WORK MANAGEMENT

Admins can insert/update/delete any donor's information. Multiple admins can register in this application. The number of blood bags are updated as they are donated each time. However, the donors cannot update/insert/delete their information neither they can login/register in this system. The number of each blood group is always shown and updated whenever they are being donated.

CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system.

The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

2.1. TRADITIONAL BLOOD BANK MANAGEMENT SYSTEM

In traditional blood bank management system, the data are stored in physical space like register books. By doing this there is often a risk of security breach, data redundancy. The data may not be updated and it may cause problems when emergency blood is needed. In this process finding necessary data is also a hassle and time consuming. Anyone can view the data which causes data security issues. As the information are stored in register books, updating the existing information can make it complex. In our blood bank management project, these issues are fixed.

2.2. PROPOSED BLOOD BANK MANAGEMENT SYSTEM

In this management system there is a login system that allows the admins to log into the application. Admins can update their information and add new users into the system. This helps us to prevent others viewing information, thus secures our system. The users can add new donors in

the database and update their information. Blood stock information is also managed by the users. The use of relational database management system helps us prevent data redundancy and security breach unlike the traditional management system. The key features of this project are:

- **Login**: Users/admins can login to their accounts.
- **Update user details**: Admins can update their information of their accounts.
- **Update donor details**: Admins can add and update the donor's information.
- **Dashboard**: Number of donors of each blood group is shown.
- **Inventory**: Number of bags of each blood group is shown. The numbers are updated when new supply of blood comes in the blood bank or blood is donated to someone from the bank.

CHAPTER 3
SYSTEM ORGANIZATION

3.1. E-R DIAGRAM DESIGN

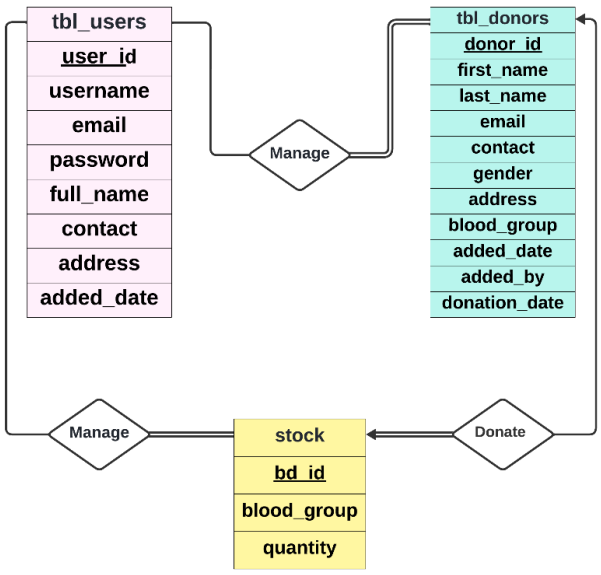


Fig: E-R Diagram

CHAPTER 4

SYSTEM ANALYSIS

4.1. PROPOSED SYSTEM

Effective blood bank management is essential for managing blood banks today. The main objective of this project is to create a secure desktop-based application by which we can store and manage necessary information of a blood bank and easily access them.

Disadvantages of existing system

- Provides minimum security mean for individual's authentication.
- The major disadvantages of this system low efficiency and time consumption.
- The maintenance cost of the existing system is high and it should be barred by the students only.

Advantages of proposed management system

- Secured login mechanisms are handled.
- Reducing the work load of blood banks.
- Time consumption is comparatively good.
- Comprehensive UI makes it easy to learn and be adapted.
- Implementation and maintenance cost is low.

CHAPTER 5
REQUIREMENT SPECIFICATION

5.1. SYSTEM REQUIREMENTS

CPU	Core i3 4rth Generation or equivalent or better
Disk Space	160 MB or more
Ram	512 Mb or better
Monitor	15 VGA Colour or better
Operating system	Windows 7/10/11
Technology Used	ASP.NET
Backend Used	MS SQL SERVER

5.2 MICROSOFT.NET FRAMEWORK

Features OF .Net

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

".NET" is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

THE .NET FRAMEWORK

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the "execution engine" of .NET. It provides the environment within which programs run. The most important features are

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.
- Loading and executing programs, with version control and other such features.
- The following features of the .NET framework are also worth description:

Managed Code

The code that targets .NET, and which contains certain extra Information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

THE CLASS LIBRARY

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

Other languages for which .NET compilers are available include

- FORTRAN
- COBOL
- Eiffel

<table>
<tr><td>ASP.NET
XML WEB SERVICES</td><td>Windows Forms</td></tr>
<tr><td colspan="2">Base Class Libraries</td></tr>
<tr><td colspan="2">Common Language Runtime</td></tr>
</table>

.Net Framework

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#. NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

CONSTRUCTORS AND DESTRUCTORS:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

GARBAGE COLLECTION

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use. In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

OVERLOADING

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

MULTITHREADING:

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

STRUCTURED EXCEPTION HANDLING

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

THE .NET FRAMEWORK

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

OBJECTIVES OF. NET FRAMEWORK

1. To provide a consistent object-oriented programming environment whether object codes are stored and executed locally on Internet-distributed, or executed remotely.
2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
3. Eliminates the performance problems.

5.3 SQL SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component

available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consists of six type of objects,

They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

TABLE:

A database is a collection of data about a specific topic.

VIEWS OF TABLE:

We can work with a table in two types,

1. Design View
2. Datasheet View

DESIGN VIEW

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

DATASHEET VIEW

To add, edit or analyses the data itself we work in tables datasheet view mode.

QUERY:

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited). Each time we run query; we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

CHAPTER 6

IMPLEMENTATION AND RESULT

6.1. CODING

Loading interface

```
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 BloodBankManagementSystem.UI
{
    public partial class frmSplash : Form
    {
        public frmSplash()
        {
            InitializeComponent();
        }
        int move = 0;

        private void timerSplash_Tick(object sender, EventArgs e)
        {
            //Write the code to show Loading Animation
            timerSplash.Interval = 20;
            panelMovable.Width += 5;

            move += 5;

            //If the loading is complete then display login form and close this
form
            if(move==640)
            {
                //Stop the Timer and Close this Form
                timerSplash.Stop();
                this.Hide();

                //Display the Login Form
                frmLogin login = new frmLogin();
                login.Show();
            }
        }

        private void frmSplash_Load(object sender, EventArgs e)
```

```
        {
            //Load the Timer
            timerSplash.Start();
        }
    }
}
```

Login panel

```
using BloodBankManagementSystem.BLL;
using BloodBankManagementSystem.DAL;
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 BloodBankManagementSystem.UI
{
    public partial class frmLogin : Form
    {
        public frmLogin()
        {
            InitializeComponent();
        }

        //Create the Object of BLL and DAL
        loginBLL l = new loginBLL();
        loginDAL dal = new loginDAL();

        //Create a Static String method to save the username
        public static string loggedInUser;

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            //Write the Code to Cose the Application
            this.Close();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            //Write the Code to Login our Application
            //1. Get the username and password from login form
            l.username = txtUsername.Text;
            l.password = txtPassword.Text;

            //Check the Login Credentials
            bool isSuccess = dal.loginCheck(l);

            //Check whehter the login is success or not
```

```
            //If login is success then isSuccess will be true else it will be
false
            if(isSuccess==true)
            {
                //Login Success
                //Display Success Message
                MessageBox.Show("Welcome to
Dashboard","Message",MessageBoxButtons.OK,MessageBoxIcon.Information);

                //Save the username in loggedInuser Stattic MEthod
                loggedInUser = l.username;

                //Display home Form
                frmHome home = new frmHome();
                home.Show();
                this.Hide(); //To CLose Login Form
            }
            else
            {
                //Login Failed
                //Display the Error Message
                MessageBox.Show("Login Failed. Try
Again.","Message",MessageBoxButtons.OK,MessageBoxIcon.Error);
            }
        }

        private void btnexit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void buttonexit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

DASHBOARD

```
using BloodBankManagementSystem.DAL;
using BloodBankManagementSystem.UI;
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 BloodBankManagementSystem
{
```

```
public partial class frmHome : Form
{
    public frmHome()
    {
        InitializeComponent();
    }

    //Create the Object of Donor Dal
    donorDAL dal = new donorDAL();

    private void pictureBoxClose_Click(object sender, EventArgs e)
    {
        //Code to Close this Application
        this.Hide();
    }

    private void usersToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //Open Users Form
        frmUsers users = new frmUsers();
        users.Show();
    }

    private void donorsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //Open Manage Donors Form
        frmDonors donors = new frmDonors();
        donors.Show();
    }


    private void frmHome_Load(object sender, EventArgs e)
    {
        //Load all the Blood Donors Count When Form is Loaded
        //Call allDonorCountMethod
        allDonorCount();

        //Display all the Donors
        DataTable dt = dal.Select();
        dgvDonors.DataSource = dt;

        //Display the username of Logged In user
        lblUser.Text = frmLogin.loggedInUser;
    }

    public void allDonorCount()
    {
        //Get the Donor Count from DAtabase and SEt in respective label
        lblOpositiveCount.Text = dal.countDonors("O+");
        lblOnegativeCount.Text = dal.countDonors("O-");
        lblApositiveCount.Text = dal.countDonors("A+");
        lblAnegativeCount.Text = dal.countDonors("A-");
        lblBpositiveCount.Text = dal.countDonors("B+");
        lblBnegativeCount.Text = dal.countDonors("B-");
        lblABpositiveCount.Text = dal.countDonors("AB+");
        lblABnegativeCount.Text = dal.countDonors("AB-");
    }
```

```
        private void frmHome_Activated(object sender, EventArgs e)
        {
            //Call allDonorCount Method
            allDonorCount();
            DataTable dt = dal.Select();
            dgvDonors.DataSource = dt;
        }

        private void txtSearch_TextChanged(object sender, EventArgs e)
        {
            //Get the Keywords from the SEarch TextBox
            string keywords = txtSearch.Text;

            //Check whether the TextBox is Empty or Not
            if(keywords != null)
            {
                //Filter the Donors based on Keywords
                DataTable dt = dal.Search(keywords);
                dgvDonors.DataSource = dt;
            }
            else
            {
                //DIsplay all the Donors
                DataTable dt = dal.Select();
                dgvDonors.DataSource = dt;
            }
        }


        private void AdvanceTeachinglinkLabel_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {

System.Diagnostics.Process.Start("https://www.youtube.com/channel/UCORMBWmoMqv5iTF
BazaeAYA/playlists?view_as=subscriber");
        }

        private void btnexit_Click_1(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void aboutToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            MessageBox.Show("This Application is Developed by Group-02 \n Intake-
49 \n section-08 \n Dept of CSE \n Bangladesh University of Business & Technology
\n Contact:
+8801537168991","Message",MessageBoxButtons.OK,MessageBoxIcon.Information);

        }

        private void menuStripTop_ItemClicked(object sender,
ToolStripItemClickedEventArgs e)
        {

        }

        private void toolStripMenuItem1_Click(object sender, EventArgs e)
```

```
        {

        }

        private void imventoryToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void addBloodUnitToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            addstock sd = new addstock();
            sd.Show();
        }

        private void removeBloodUnitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            deletestock si = new deletestock();
            si.Show();
        }

        private void detailsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            details sf = new details();
            sf.Show();
        }

        private void btnminimize_Click(object sender, EventArgs e)
        {

        }

        private void btnmaximize_Click(object sender, EventArgs e)
        {

        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm = new Bitmap(this.dgvDonors.Width, this.dgvDonors.Height);
            dgvDonors.DrawToBitmap(bm, new Rectangle(0, 0, this.dgvDonors.Width,
this.dgvDonors.Height));
            e.Graphics.DrawImage(bm, 0, 0);
        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }

        private void btnexit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
```

```
}
```

USER FORM

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

namespace BloodBankManagementSystem.UI
{
    public partial class frmUsers : Form
    {
        public frmUsers()
        {
            InitializeComponent();
        }

        //Create Objects of userBLL and userDAL
        userBLL u = new userBLL();
        userDAL dal = new userDAL();

        private void pictureBoxClose_Click(object sender, EventArgs e)
        {
            //Add functionality to close this form
            this.Close();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            //Step 1: Get the Values from UI
            if (txtFullName.Text.Trim() != string.Empty || txtEmail.Text.Trim() !=
string.Empty || txtUsername.Text.Trim() != string.Empty || txtPassword.Text.Trim()
!= string.Empty || txtPassword.Text.Trim() != string.Empty ||
txtContact.Text.Trim() != string.Empty || txtAddress.Text.Trim() != string.Empty)
            {
                u.full_name = txtFullName.Text;
                u.email = txtEmail.Text;
                u.username = txtUsername.Text;
                u.password = txtPassword.Text;
                u.contact = txtContact.Text;
                u.address = txtAddress.Text;
                u.added_date = DateTime.Now;

                //Step2: Adding the Values from UI to the Database
                //Create a Boolean Variable to check whether the data is inserted
successfully or not
```

```
                bool success = dal.Insert(u);

                //Step 3: Check whether the Data is Inserted Successfully or Not
                if (success == true)
                {
                    //Data or User Added Successfully
                    MessageBox.Show("New User added Successfully.", "Mesage",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    //Display the user in DataGrid View
                    DataTable dt = dal.Select();
                    dgvUsers.DataSource = dt;

                    //Clear TextBoxes
                    Clear();
                }
                else
                {
                    //Failed to Add User
                    MessageBox.Show("Failed to Add New User.", "Mesage",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
            }
            else
            {
                MessageBox.Show("All Fields are
required","Message",MessageBoxButtons.OK,MessageBoxIcon.Warning);
            }
        }

        //Method or Function to Clear TextBoxes
        public void Clear()
        {
            txtFullName.Text = "";
            txtEmail.Text = "";
            txtUsername.Text = "";
            txtContact.Text = "";
            txtAddress.Text = "";
            txtPassword.Text = "";
            txtUserID.Text = "";
        }

        private void dgvUsers_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            //Find the Row Index of the Row Clicked on Users Data Frid View
            int RowIndex = e.RowIndex;
            txtUserID.Text = dgvUsers.Rows[RowIndex].Cells[0].Value.ToString();
            txtUsername.Text = dgvUsers.Rows[RowIndex].Cells[1].Value.ToString();
            txtEmail.Text = dgvUsers.Rows[RowIndex].Cells[2].Value.ToString();
            txtPassword.Text = dgvUsers.Rows[RowIndex].Cells[3].Value.ToString();
            txtFullName.Text = dgvUsers.Rows[RowIndex].Cells[4].Value.ToString();
            txtContact.Text = dgvUsers.Rows[RowIndex].Cells[5].Value.ToString();
            txtAddress.Text = dgvUsers.Rows[RowIndex].Cells[6].Value.ToString();

        }

        private void frmUsers_Load(object sender, EventArgs e)
```

```
        {
            //Display the Users in DAtagrid View When the Form is Loaded
            DataTable dt = dal.Select();
            dgvUsers.DataSource = dt;
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            //Step1: Get the Values from UI
            if (txtUserID.Text.Trim() != string.Empty)
            {
                u.user_id = int.Parse(txtUserID.Text);
                u.full_name = txtFullName.Text;
                u.email = txtEmail.Text;
                u.username = txtUsername.Text;
                u.password = txtPassword.Text;
                u.contact = txtContact.Text;
                u.address = txtAddress.Text;
                u.added_date = DateTime.Now;

                //Step 2: Create a Boolean variable to check whether the data is
updated successfully or not
                bool success = dal.Update(u);

                //Let's check whether the data is updated or not
                if (success == true)
                {
                    //Data Udated Successfully
                    MessageBox.Show("User Updated
Successfully.","Mesage",MessageBoxButtons.OK,MessageBoxIcon.Information);

                    //Refresh DAta Grid View
                    DataTable dt = dal.Select();
                    dgvUsers.DataSource = dt;

                    //Clear the TextBoxes
                    Clear();
                }
            }
            else
            {
                MessageBox.Show("First Double Click on data that you want to
Update","Message",MessageBoxButtons.OK,MessageBoxIcon.Warning);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            //Step 1: Get the UserID from Text Box to Delete the User
            if (txtUserID.Text.Trim() != string.Empty)
            {
                u.user_id = int.Parse(txtUserID.Text);

                //Step Create the Boolean value to check whether the user deleted
or not
                bool success = dal.Delete(u);

                //Let's check whteher the user is Deleted or Not
```

```
                if (success == true)
                {
                    //User Deleted Successfully
                    MessageBox.Show("User Deleted
Successfully.","Mesage",MessageBoxButtons.OK,MessageBoxIcon.Information);

                    //Refresh DataGrid View
                    DataTable dt = dal.Select();
                    dgvUsers.DataSource = dt;

                    //Clear the TextBoxes
                    Clear();
                }
            }
            else
            {
                MessageBox.Show("First Double Click on data that you want to
Delete", "Message", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            //Call the user Function
            Clear();
        }

        private void btnSelectImage_Click(object sender, EventArgs e)
        {

        }

        private void txtSearch_TextChanged(object sender, EventArgs e)
        {
            //Write the Code to Get the users BAsed on Keywords
            //1. Get the Keywords from the TExtBox
            String keywords = txtSearch.Text;

            //Check whether the textbox is empty or not
            if(keywords!=null)
            {
                //TextBox is not empty, display users on DAta Grid View based on
the keywords
                DataTable dt = dal.Search(keywords);
                dgvUsers.DataSource = dt;
            }
            else
            {
                //TExtbox is Empty and display all the users on DAta Grid View
                DataTable dt = dal.Select();
                dgvUsers.DataSource = dt;
            }
        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm = new Bitmap(this.dgvUsers.Width, this.dgvUsers.Height);
```

```
            dgvUsers.DrawToBitmap(bm, new Rectangle(0, 0, this.dgvUsers.Width,
this.dgvUsers.Height));
            e.Graphics.DrawImage(bm, 0, 0);

        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }
    }
}
```

DONORS' FORM

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

namespace BloodBankManagementSystem.UI
{
    public partial class frmDonors : Form
    {
        public frmDonors()
        {
            InitializeComponent();
        }
        //Create object of Donor BLL and Donor DAL
        donorBLL d = new donorBLL();
        donorDAL dal = new donorDAL();
        userDAL udal = new userDAL();


        private void frmDonors_Load(object sender, EventArgs e)
        {
            //Display Donors in DataGrid View
            DataTable dt = dal.Select();
            dgvDonors.DataSource = dt;
        }

        private void pictureBoxClose_Click(object sender, EventArgs e)
        {
            //Close this form
            this.Close();
        }
```

```
        private void btnAdd_Click(object sender, EventArgs e)
        {
            //We will write the code to Add new Donor
            //Step 1. Get the DAta from Manage Donors Form
            if (txtFirstName.Text.Trim() != string.Empty ||
txtLastName.Text.Trim() != string.Empty || txtEmail.Text.Trim() != string.Empty ||
cmbGender.Text.Trim() != string.Empty || cmbBloodGroup.Text.Trim() != string.Empty
|| txtContact.Text.Trim() != string.Empty || txtAddress.Text.Trim() !=
string.Empty)
            {
                d.first_name = txtFirstName.Text;
                d.last_name = txtLastName.Text;
                d.email = txtEmail.Text;
                d.gender = cmbGender.Text;
                d.blood_group = cmbBloodGroup.Text;
                d.contact = txtContact.Text;
                d.address = txtAddress.Text;
                d.added_date = DateTime.Now;
                d.donation_date= DateTime.Now;

                //Get The ID of Logged In USer
                string loggedInUser = frmLogin.loggedInUser;
                userBLL usr = udal.GetIDFromUsername(loggedInUser);

                d.added_by = usr.user_id;

                //Step2: Inserting Data into Database
                //Create a Boolean Variable to Isnert DAta into DAtabase and check
whether the data inserted successfully of not
                bool isSuccess = dal.Insert(d);

                //if the Data is inserted successfully then the values of
isSuccess will be True else it will be false
                if (isSuccess == true)
                {
                    //Data Inserted Successfully
                    MessageBox.Show("New Donor Added Successfully", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    //Refresh Datagrid View
                    DataTable dt = dal.Select();
                    dgvDonors.DataSource = dt;

                    //Clear all the Textboxes
                    Clear();
                }
                else
                {
                    //FAiled to Insert Data
                    MessageBox.Show("Failed to Add new Donor.", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("All Fields are required", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
            }
        }

        //Create a Method to Clear all the Textboxes
        public void Clear()
        {
            //Clear all the TExtboxes
            txtFirstName.Text = "";
            txtLastName.Text = "";
            txtEmail.Text = "";
            txtDonorID.Text = "";
            cmbGender.SelectedIndex = -1;
            cmbBloodGroup.SelectedIndex = -1;
            txtContact.Text = "";
            txtAddress.Text = "";
        }

        private void dgvDonors_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            //SElect the DAta from DAtagrid View and Display in our Form

            //1. Find the Row Selected
            int RowIndex = e.RowIndex;

            txtDonorID.Text = dgvDonors.Rows[RowIndex].Cells[0].Value.ToString();
            txtFirstName.Text =
dgvDonors.Rows[RowIndex].Cells[1].Value.ToString();
            txtLastName.Text = dgvDonors.Rows[RowIndex].Cells[2].Value.ToString();
            txtEmail.Text = dgvDonors.Rows[RowIndex].Cells[3].Value.ToString();
            txtContact.Text = dgvDonors.Rows[RowIndex].Cells[4].Value.ToString();
            cmbGender.Text = dgvDonors.Rows[RowIndex].Cells[5].Value.ToString();
            txtAddress.Text = dgvDonors.Rows[RowIndex].Cells[6].Value.ToString();
            cmbBloodGroup.Text =
dgvDonors.Rows[RowIndex].Cells[7].Value.ToString();
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            //Add the Functionality to Update the Donors
            //1. Get the Values from Form
            if (txtDonorID.Text.Trim() != string.Empty)
            {
                d.donor_id = int.Parse(txtDonorID.Text);
                d.first_name = txtFirstName.Text;
                d.last_name = txtLastName.Text;
                d.email = txtEmail.Text;
                d.gender = cmbGender.Text;
                d.blood_group = cmbBloodGroup.Text;
                d.contact = txtContact.Text;
                d.address = txtAddress.Text;

                DateTime donationDate = donorDatetime.Value;
                d.donation_date = donationDate;

                //Get The ID of Logged In USer
                string loggedInUser = frmLogin.loggedInUser;
                userBLL usr = udal.GetIDFromUsername(loggedInUser);
```

```
                d.added_by = usr.user_id;

                //Create a Boolean Variable to Check whether the data updated
successfully or not
                bool isSuccess = dal.Update(d);

                //If the data updated successfully then the value of isSuccess
will be true else it will be false
                if (isSuccess == true)
                {
                    //Donor Updated Successfully
                    MessageBox.Show("Donor updated Successfully.", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    Clear();

                    //Refresh Datagridview
                    DataTable dt = dal.Select();
                    dgvDonors.DataSource = dt;

                }
                else
                {
                    //Failed to Update
                    MessageBox.Show("Failed to update donors.", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("First Double Click on data that you want to
Update", "Message", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            //Get the value from form
            if (txtDonorID.Text.Trim() != string.Empty)
            {
                d.donor_id = int.Parse(txtDonorID.Text);
                //Create a Boolean Variable to Check whether the donor deleted or
not
                bool isSuccess = dal.Delete(d);

                if (isSuccess == true)
                {
                    //Donor Deleted Successfully
                    MessageBox.Show("Donor Deleted Successfully.", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    Clear();

                    //Refresh Datagrid View
                    DataTable dt = dal.Select();
                    dgvDonors.DataSource = dt;
                }
                else
```

```
                {
                    //Failed to Delete Donor
                    MessageBox.Show("Failed to Delete Donor", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("First Double Click on data that you want to
Delete", "Message", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            //Clear the TExtboxes
            Clear();
        }

        private void btnSelectImage_Click(object sender, EventArgs e)
        {

        }

        private void txtSearch_TextChanged(object sender, EventArgs e)
        {
            //Let's Add the Dunctionality to Search the Donors

            //1. Get the Keywords Typed on the Search TExt Box
            string keywords = txtSearch.Text;

            // Check Whether the Search TExtBox is Empty or Not
            if(keywords != null)
            {
                //Display the information of Donors Based on Keywords
                DataTable dt = dal.Search(keywords);
                dgvDonors.DataSource = dt;
            }
            else
            {
                //DIsplay all the Donors
                DataTable dt = dal.Select();
                dgvDonors.DataSource = dt;
            }
        }

        private void dgvDonors_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm = new Bitmap(this.dgvDonors.Width, this.dgvDonors.Height);
            dgvDonors.DrawToBitmap(bm, new Rectangle(0, 0, this.dgvDonors.Width,
this.dgvDonors.Height));
```

```
            e.Graphics.DrawImage(bm, 0, 0);

        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void txtAddress_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

ADD STOCK

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

namespace BloodBankManagementSystem.UI
{
    public partial class addstock : Form
    {
        function fn = new function();
        string query;
        public addstock()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }
```

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

        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
        private void addstock_Load(object sender, EventArgs e)
        {
            query = "select blood_group,quantity from stock";
            DataSet ds = fn.getData(query);
            dataGridView1.DataSource = ds.Tables[0];

        }

        private void button1_Click(object sender, EventArgs e)
        {
            query = " update stock set quantity=quantity+ " + txtUnits.Text + "
where blood_group ='" + txtBloodGroup.Text + "'";
            fn.setDate(query);
            addstock_Load(this, null);
        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }

        private void btnclose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm = new Bitmap(this.dataGridView1.Width,
this.dataGridView1.Height);
            dataGridView1.DrawToBitmap(bm, new Rectangle(0, 0,
this.dataGridView1.Width, this.dataGridView1.Height));
            e.Graphics.DrawImage(bm, 0, 0);
        }
    }
}
```

DELETE STOCK

```
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 BloodBankManagementSystem.UI
{
    public partial class deletestock : Form
    {
        function fn = new function();
        string query;

        public deletestock()
        {
            InitializeComponent();
        }

        private void deletestock_Load(object sender, EventArgs e)
        {
            query = "select blood_group,quantity from stock";
            DataSet ds = fn.getData(query);
            dataGridView1.DataSource = ds.Tables[0];
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void btndelete_Click(object sender, EventArgs e)
        {
            query = " update stock set quantity=quantity- " + txtUnits.Text + "
where blood_group ='" + txtBloodGroup.Text + "'";
            fn.setDate(query);
            deletestock_Load(this, null);
        }

        private void btnclose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm = new Bitmap(this.dataGridView1.Width,
this.dataGridView1.Height);
```

```
            dataGridView1.DrawToBitmap(bm, new Rectangle(0, 0,
this.dataGridView1.Width, this.dataGridView1.Height));
            e.Graphics.DrawImage(bm, 0, 0);
        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }
    }
}
```

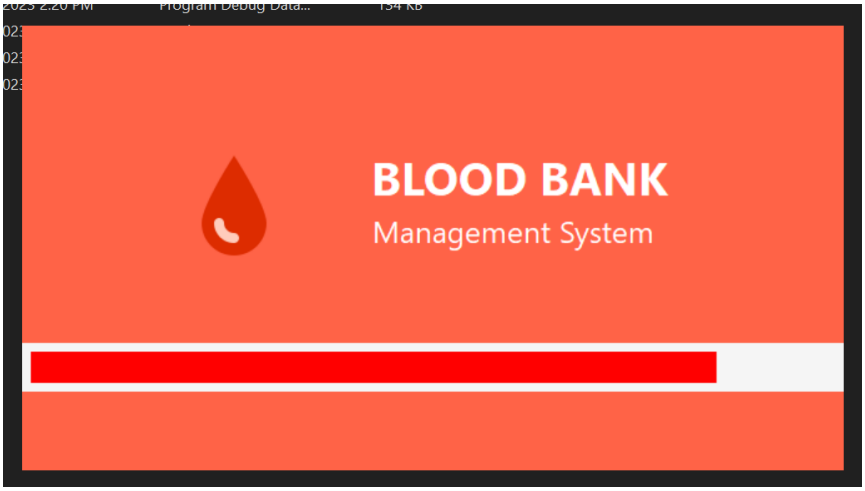
STOCK DETAILS

```
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 BloodBankManagementSystem.UI
{
    public partial class details : Form
    {
        function fn = new function();
        string query;
        public details()
        {
            InitializeComponent();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {


        }

        private void details_Load(object sender, EventArgs e)
        {
            query = "select * from stock";
            DataSet ds = fn.getData(query);
            dataGridView1.DataSource = ds.Tables[0];
        }

        private void btnclose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

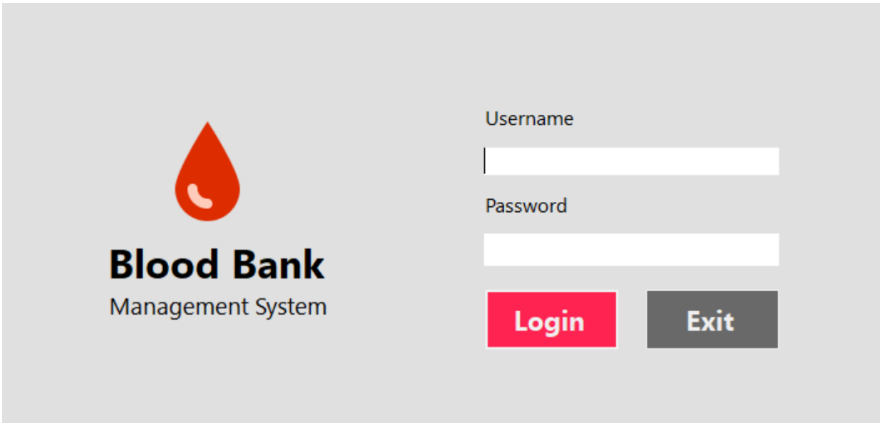
```
        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void lblunit_Click(object sender, EventArgs e)
        {

        }

        private void txtBloodGroup_SelectedIndexChanged(object sender, EventArgs
e)
        {

        }

        private void txtUnits_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void BloodGroup_Click(object sender, EventArgs e)
        {

        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            Bitmap bm= new Bitmap(this.dataGridView1.Width,
this.dataGridView1.Height);
            dataGridView1.DrawToBitmap(bm,new
Rectangle(0,0,this.dataGridView1.Width,this.dataGridView1.Height));
            e.Graphics.DrawImage(bm,0,0);
        }

        private void label3_Click(object sender, EventArgs e)
        {
            printDocument1.Print();
        }
    }
}
```

6.2. SCREENSHOTS

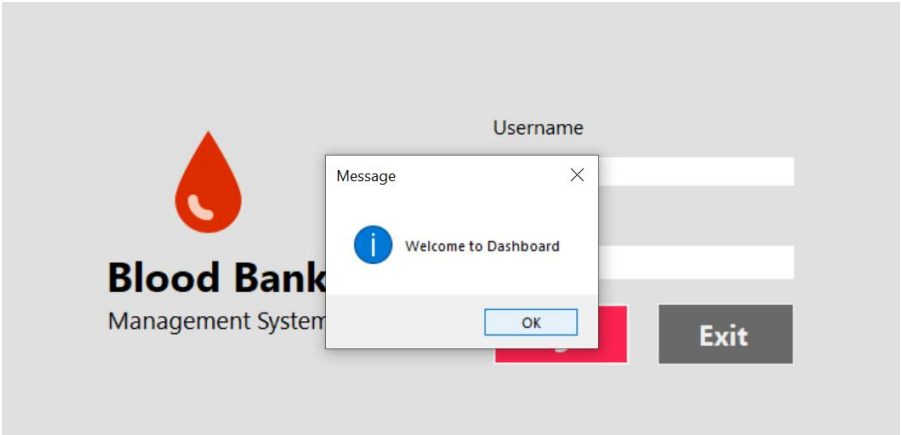
1. **Loading interface:**



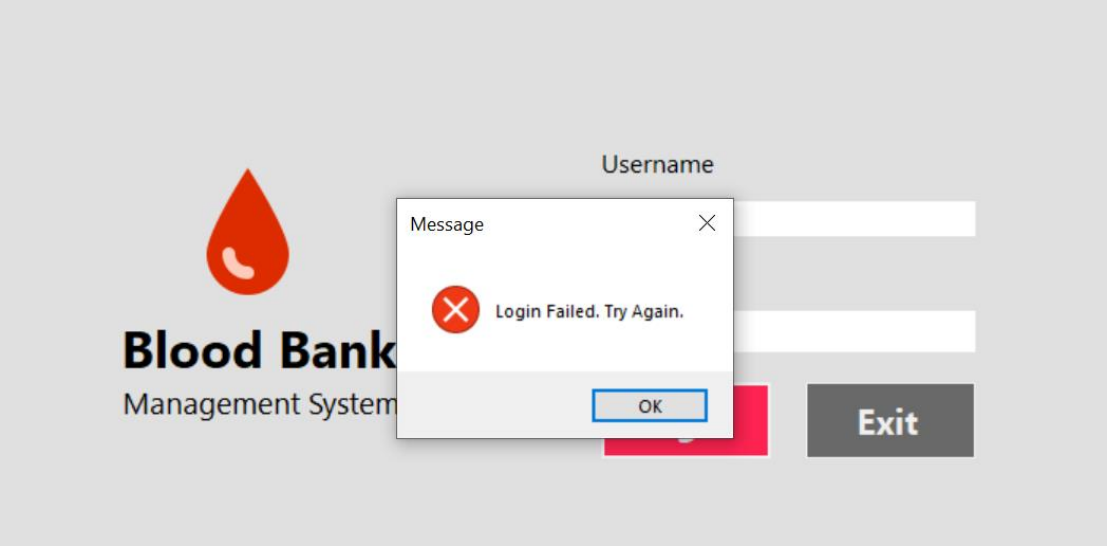
2. **Login panel:**



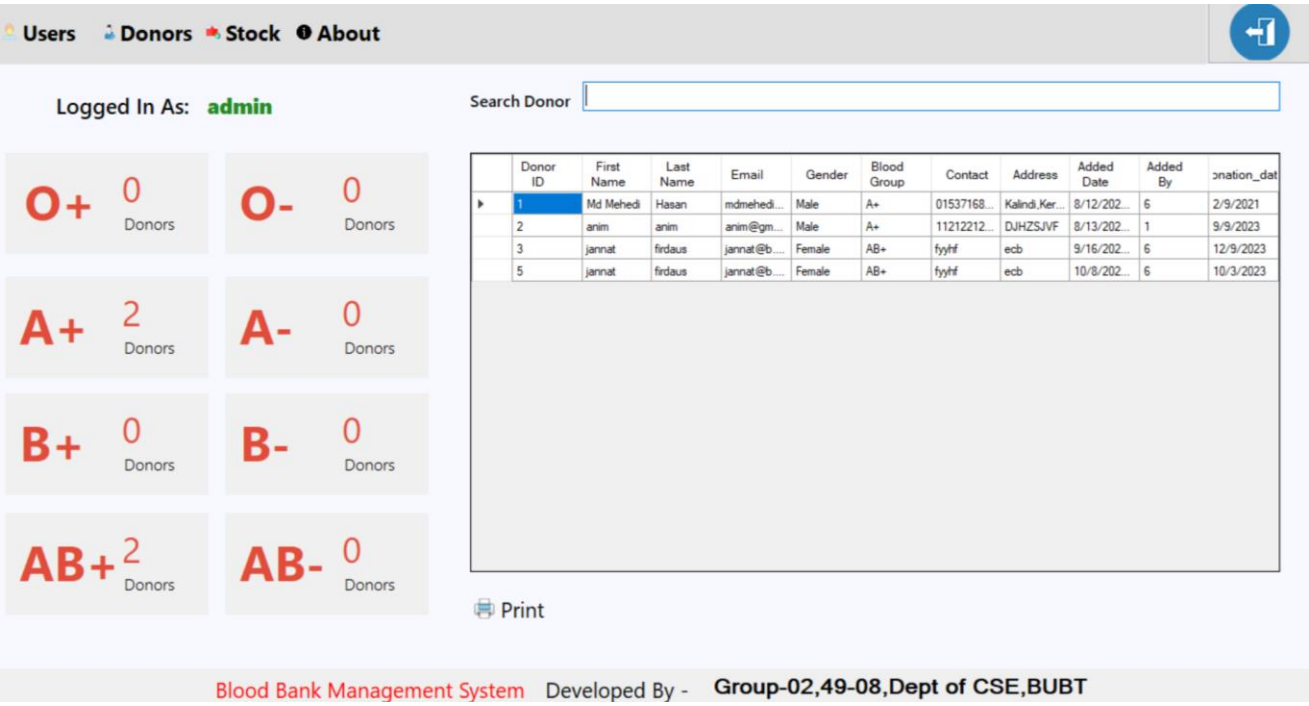
3. **Successfully login shows welcome message:**



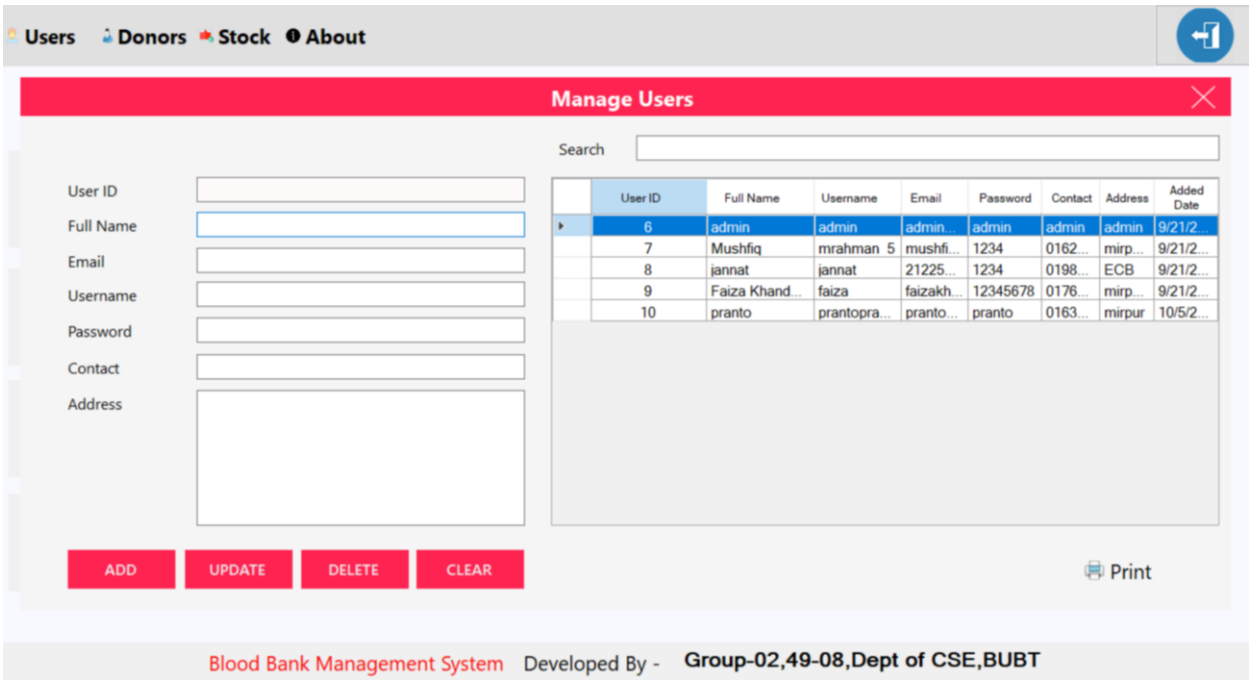
4. **Wrong password or username shows failed attempt message:**



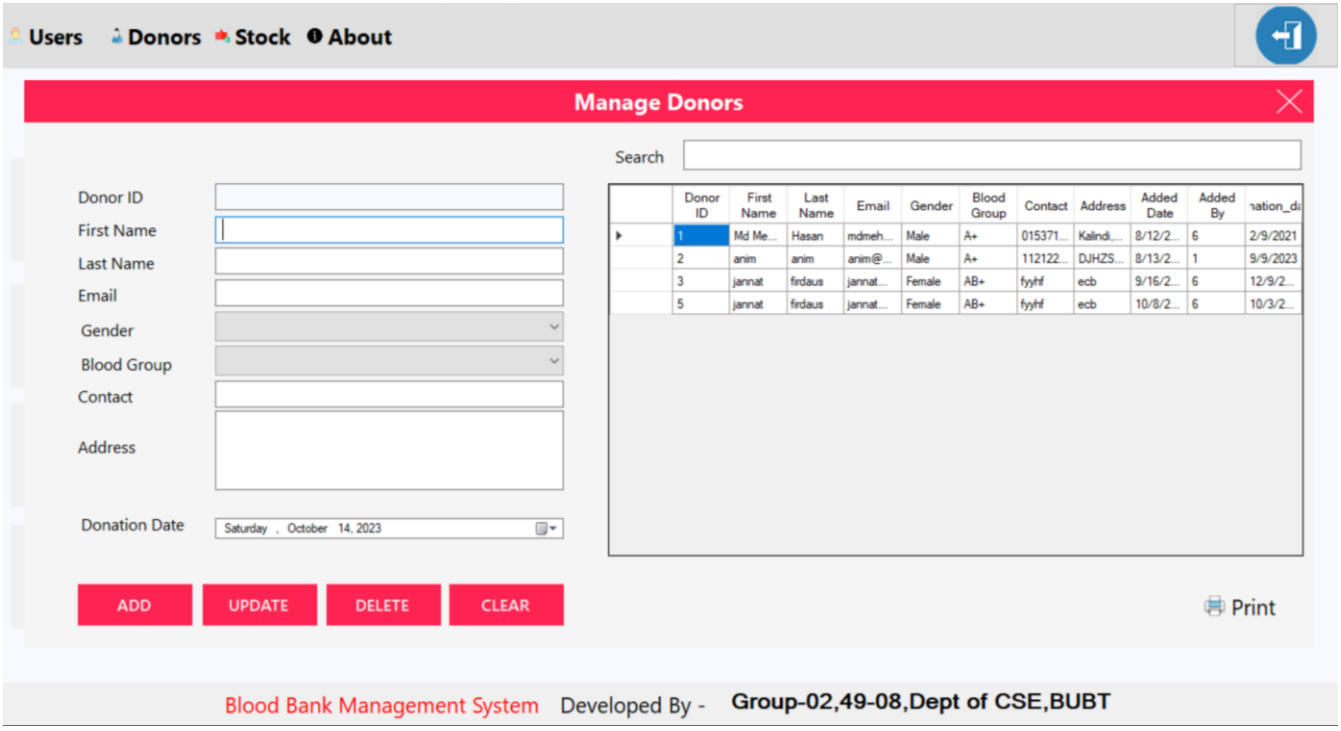
5. **Dashboard or Home page:**



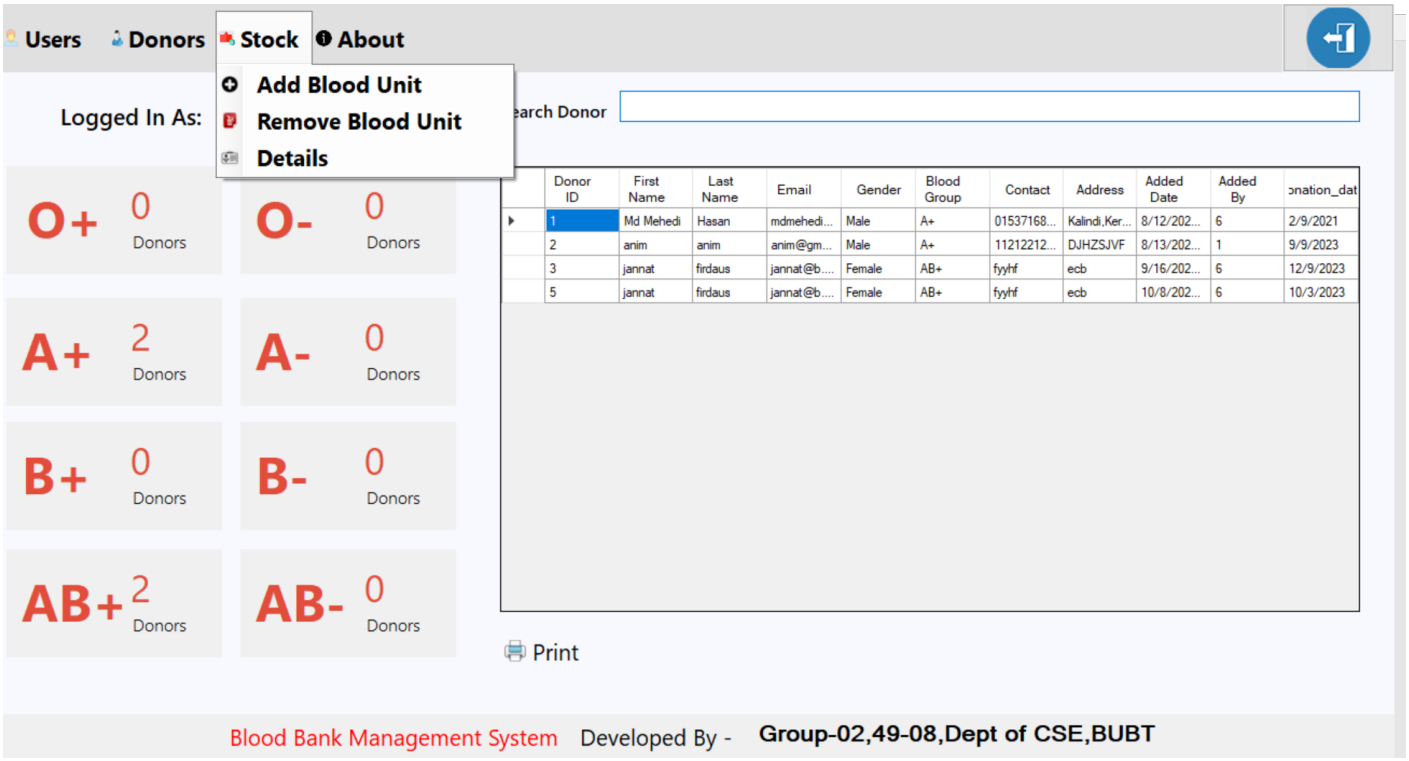
6. **Clicking on Users opens user form:**



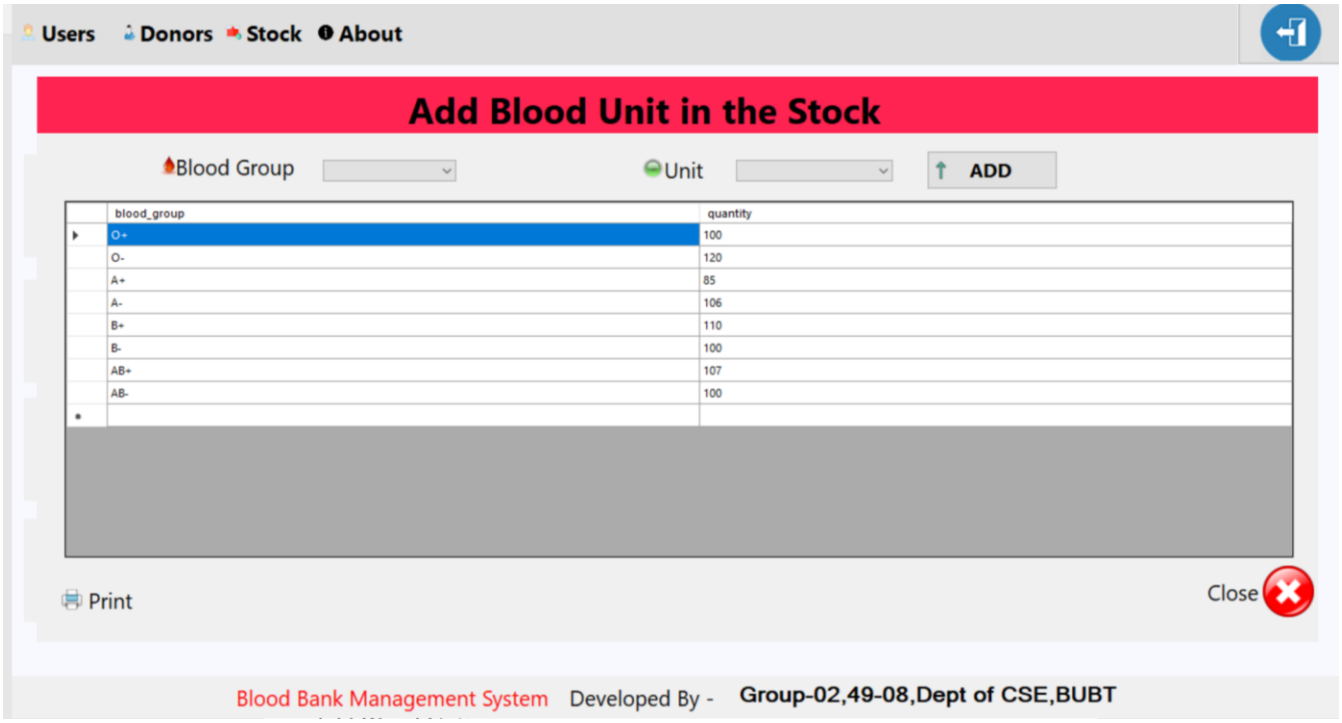
7. **Clicking on Donors will open Donor's form**



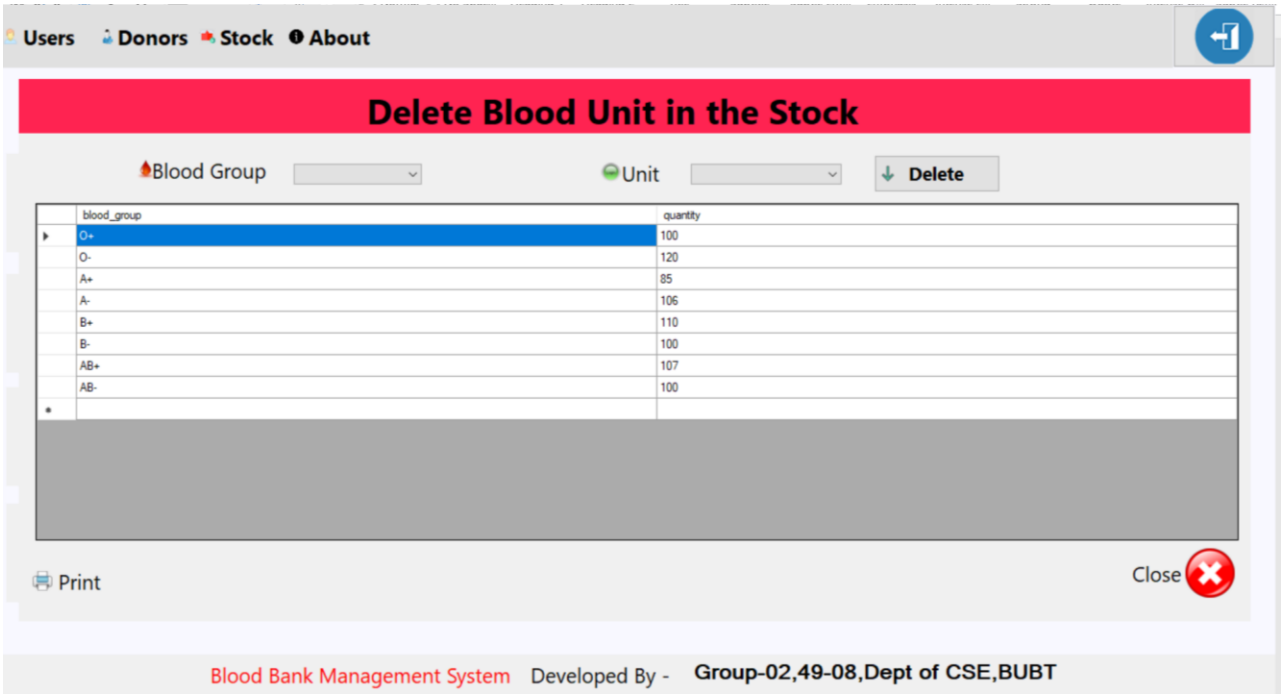
8. **Clicking on Stock:**



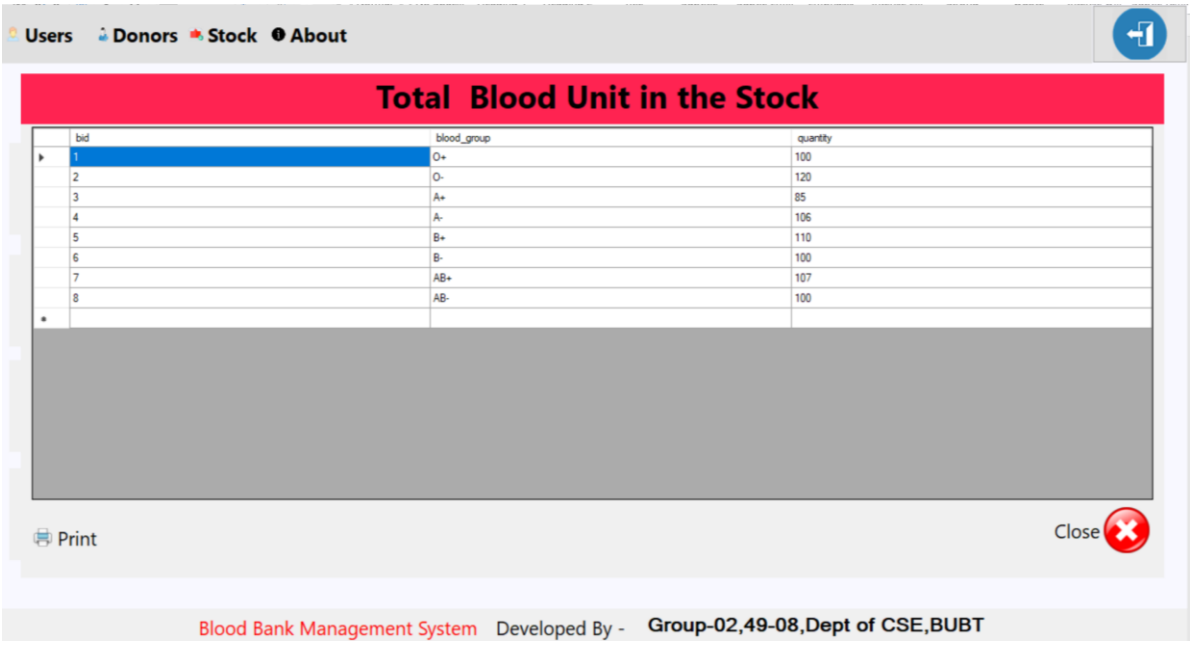
9. Add Blood Unit:



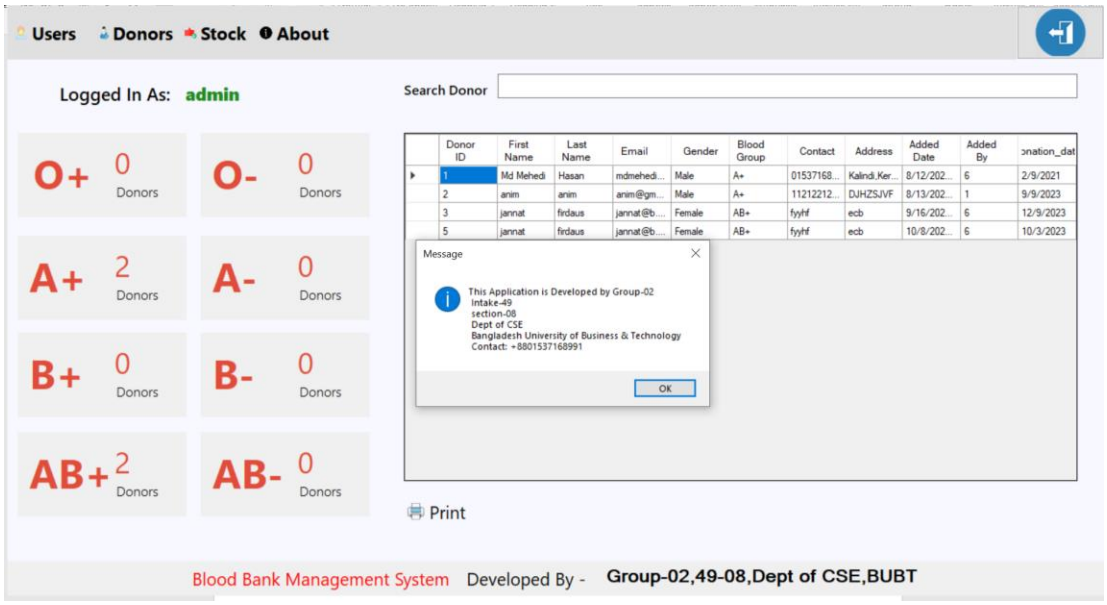
10. Remove Blood Unit:



11. Clicking on Details:



12. Clicking on About:



CHAPTER 7

CONCLUSION

This blood bank management system can be useful for both donors and recipients. Donors can use this system to find out where they can donate blood, while recipients can use them to find out where they can receive blood transfusions. This particular application can also help hospitals and clinics manage their blood supplies more efficiently by providing real-time information about available blood stocks.

In conclusion, a blood bank management application is an essential tool that can help save lives by making the process of blood donation more convenient and efficient. This application provides a platform for donors and recipients to connect with each other, and they also help hospitals and clinics manage their blood supplies more effectively. By using a blood bank management system, we can make a difference in someone's life by donating or receiving life-saving blood when it is needed the most.

CHAPTER 8

REFERENCES

1.Professional ASP.NET 1.0, Special Edition

Author(s) : Alex Homer, Brian Francis, David Sussman, Karli Watson, Richard Anderson, Robert Howard

2.eXtreme .NET: Introducing eXtreme Programming Techniques to .NET Developers

Author(s) : Dr. Neil Roodyn

3.Student's Essential Guide to .NET, 1st Edition

Author(s) : Tony Grimer

4.Professional DotNetNuke ASP.NET Portals

Author(s) : Shaun Walker, Patrick J. Santry, Joe Brinkman, Dan Caron,

Web References:

- http://www.microsoft.com
- http://www.vb101.com
- http://www.codeguru.com