Blood Donation Management System

Session: 2022 – 2026

Submitted by:

Muhammad Wali Ahmad 2022-CS-65

Supervised by:

Maida Shahid

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

1. INTRODUCTION:

1.1 Overview:

Problem Statement: During emergency situations or surgeries, finding the required blood group quickly and efficiently can be a challenging task. This creates a need for a blood donation management system that simplifies the process and provides a platform for individuals to search for their desired blood group.

Solution: The blood donation management system aims to address the problem by utilizing technology to streamline the blood donation process. It provides an organized system that connects donors and individuals in need of specific blood types, making it easier to locate and acquire the required blood group in a timely manner.

Significance: The development and implementation of a blood donation management system can have a positive impact on both computer science and healthcare. By leveraging technology, the system improves the efficiency and accessibility of blood donation processes, ensuring the availability of safe and adequate blood supplies. This has the potential to significantly improve healthcare outcomes and save lives, making it an essential tool for patients, healthcare providers, and blood donors.

1.2 Objectives:

Enable Easy Blood Group Search: The primary objective of the blood donation management system is to provide a user-friendly platform where individuals can quickly and easily search for their desired blood group during emergency situations or surgeries.

Streamline Blood Donation Processes: The system aims to automate and streamline various aspects of the blood donation process, including managing donor information, maintaining records of donated blood units, and organizing blood drives or collection centers. This objective ensures efficient management of blood resources.

Improve Healthcare Outcomes: By ensuring the availability of safe and adequate blood supplies for transfusions, the system aims to contribute to better healthcare outcomes. This objective involves reducing the time and effort required to find matching blood donors, leading to improved patient care, and potentially saving lives.

1.3 Intended Functionality:

User-Friendly Interface: The blood donation management system will have a user-friendly interface that allows individuals to easily search for specific blood groups based on their requirements. The interface will be intuitive, responsive, and accessible to a wide range of users.

Real-Time Availability: The system will provide real-time information on the availability of different blood groups. This feature enables quick identification and location of the required blood type, ensuring timely access to blood donors during emergency situations or surgeries.

Donor and Donation Management: The system will facilitate the management of donor information, including registration, eligibility screening, and contact details. It will also maintain records of donated blood units, ensuring accurate tracking and inventory management.

Blood Drive Organization: The system will enable the organization and coordination of blood drives and collection centers. This functionality includes scheduling, volunteer management, and communication tools to optimize the efficiency of blood donation campaigns.

Data Security and Privacy: The blood donation management system will prioritize the security and privacy of donor and patient information. It will incorporate robust security measures and adhere to data protection regulations to ensure the confidentiality and integrity of sensitive data.

2. OOP-CONCEPTS:

In the context of the blood donation management system, object-oriented programming (OOP) principles and concepts can be applied to enhance the design, organization, and functionality of the system. Here are some examples of how OOP can be associated with various aspects of the system:

2.2 Association:

Association is of two types of aggregation and composition; both are applied in the management system. Aggregation is present between different classes like all classes of UI to classes of BL and DL. Also, there is aggregation between Employee class with Person class employee contain list of persons. While composition is present between DL classes to the respective BL classes like PersonCRUD.cs has composition with Person.cs.

2.2 Encapsulation:

Encapsulation is a fundamental principle of OOP that involves bundling data and related methods within a class, hiding the internal implementation details from other parts of the system. In the blood donation management system, encapsulation can be employed to encapsulate the properties and methods of classes such as Donor, Recipient and Employee protecting their internal state and providing controlled access to their data.

2.3 Inheritance:

Inheritance is another key concept in OOP that allows the creation of hierarchical relationships between classes. In the blood donation management system, inheritance can be utilized to establish relationships between several types of donors or recipients to a single person class. For example, a Donor class could inherit from a more general Person class, inheriting its properties and methods while adding specific characteristics or behaviors. Also, there is inheritance between Users of the systems.

2.4 Polymorphism:

Polymorphism enables objects of different classes to be treated interchangeably, allowing for flexibility and extensibility in the system. In the blood donation management system, polymorphism can be applied when handling different types of donors and recipients. Methods or functions that operate on donors or recipients can be designed to accept objects of the general class type named Person, enabling the system to handle various specific instances transparently. It is also helpful when storing it into a list.

3. COMPARE WITH PROCEDURAL PROGRAMING:

Following are the various aspects in which OOP is better than procedural programing:

3.1 Reusability and Code Sharing:

OOP promotes code reusability through the concept of inheritance. Inheritance allows classes to inherit properties and behaviors from other classes, reducing the need to rewrite code. This not only saves development time but also makes the codebase more efficient and easier to maintain. In procedural programming, reusing code requires copying and pasting or creating separate functions, which can lead to code redundancy and maintenance issues.

3.2 Maintainability and Scalability:

OOP promotes code maintainability and scalability. With encapsulation and modularity, making changes to a specific functionality or fixing issues becomes easier because the affected code is localized within the relevant class. Additionally, OOP's ability to extend existing classes through inheritance allows for the addition of new features without modifying the existing codebase extensively. In procedural programming, making changes or adding features often involves modifying multiple functions, increasing the likelihood of introducing errors and making maintenance more challenging.

3.3 Code Readability and Understandability:

OOP encourages a more natural representation of real-world entities and their relationships. This makes the codebase more readable and understandable, as classes and objects closely resemble the entities and interactions they model. Procedural programming, on the other hand, can lack this intuitive representation, making it harder to grasp the overall system design and the relationships between various parts of the code.

3.4 Data Security and Abstraction:

OOP allows for the implementation of data security through the concept of encapsulation. By hiding the internal implementation details of a class, OOP ensures that data can only be accessed and modified through designated methods, reducing the risk of unintended modifications or data corruption. Procedural programming typically lacks built-in mechanisms for data security and abstraction, making it more prone to data integrity issues and unauthorized access.

4. DESIGN PATTERN IMPLEMENTATION:

In the blood donation management system, the utilization of design patterns helps ensure modularity, separation of concerns, and maintainability. Here is how the project incorporates the Business Logic (BL), Data Access Layer (DL), and User Interface (UI) design patterns:

4.1 Business Logic (BL) Design Pattern:

The BL design pattern focuses on encapsulating the business rules and logic of the application. It ensures that the core functionality and operations of the system are independent of the underlying data storage or user interface.

In the blood donation management system: The BL design pattern can be implemented using the **Model-View-Controller** (MVC) architectural pattern. The model represents the business logic and operations of the system. It encapsulates functionalities such as single employee functionalities and his credentials, donor or recipient details, and their related functionality. The controllers handle the interaction between the user interface and the business logic. They receive user input, invoke appropriate operations in the model, and update the views accordingly. The views represent the user interface components, which display information to the users and receive their input. They communicate with the controllers to trigger actions and display updated information. By implementing the BL design pattern, the blood donation management system achieves a separation of concerns between the business logic and the user interface, making it easier to maintain and modify each component independently.

4.2 Data Access Layer (DL) Design Pattern:

The DL design pattern focuses on managing the interaction between the application and the underlying data storage or database. It abstracts the data access operations, ensuring that the business logic is decoupled from the specifics of the data storage implementation.

In the blood donation management system: The DL design pattern can be implemented using the **Repository pattern**. The repositories provide an abstraction layer between the business logic and the database. They handle data retrieval, storage, and update operations. The repositories encapsulate the details of the data access, such as querying the data, mapping data to objects, and ensuring data integrity and consistency. By applying the DL design pattern, the blood donation management system achieves separation between the business logic and the data storage implementation. This separation allows for easier maintenance, scalability, and the potential to switch or upgrade the underlying database without impacting the overall functionality of the system.

4.3 User Interface (UI) Design Pattern:

The UI design pattern focuses on structuring and organizing the user interface components to ensure a consistent and user-friendly experience. It separates the presentation layer from the underlying business logic and data access operations.

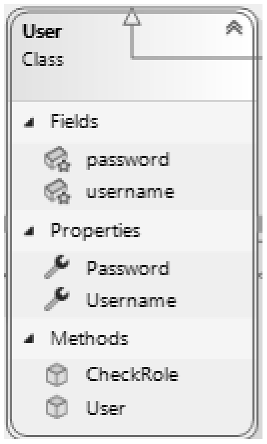
In the blood donation management system: The UI design pattern can be implemented using the **Model-View-ViewModel** (MVVM) architectural pattern. The model represents the data and business logic, while the view model acts as an intermediary between the model and the view. The view model exposes properties and commands that the view can bind to, facilitating data binding and handling user actions. The view represents the visual components of the user interface, displaying the data from the view model and capturing user input. By utilizing the UI design pattern, the blood donation management system separates the UI concerns from the underlying business logic and data access operations. This separation allows for easier modification, testing, and customization of the user interface, without impacting the core functionality of the system.

5. CLASSES DETAILS:

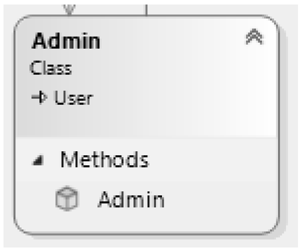
Explanations of classes in the blood donation management system along with class diagram to illustrate their responsibilities:

5.1 Business Logic (BL) Classes:

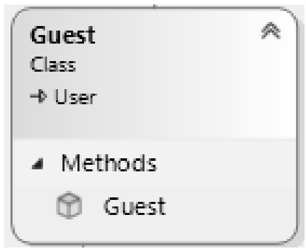
- **User:**
  This is the main class for login to the application. It is the parent class for three users i.e., admin, employee, and guest.



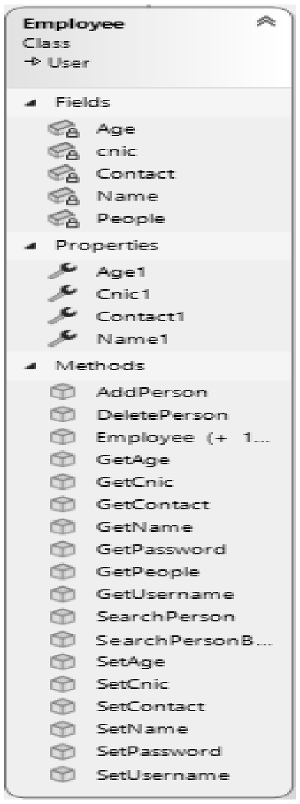
- **Admin:**
  Admin is the child class of user, and it is responsible for employee's operation.



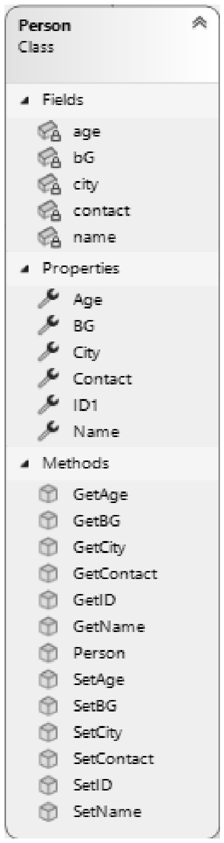
- **Guest:**
  Guest is the child class of user, and it is responsible add request for blood group.



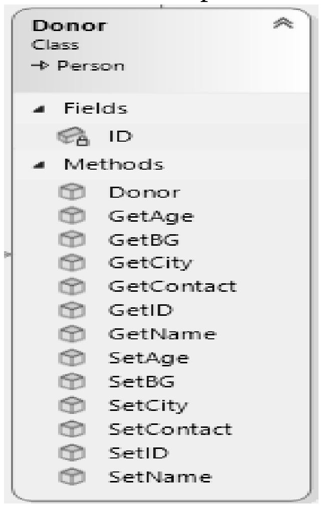
- **Employee:**
  Employee is the child class of user, and it is responsible for donors and recipient operation.



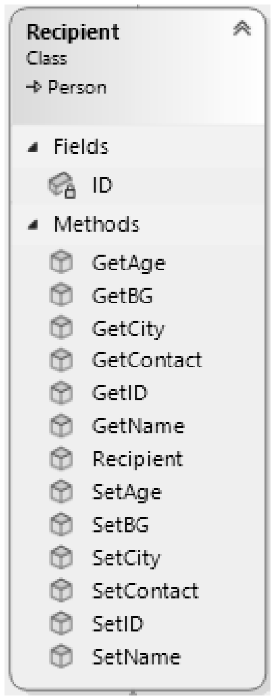
- **Person:**
  A person is also a parent class for donor and recipient. There is dynamic polymorphism present in this class with his child classes.



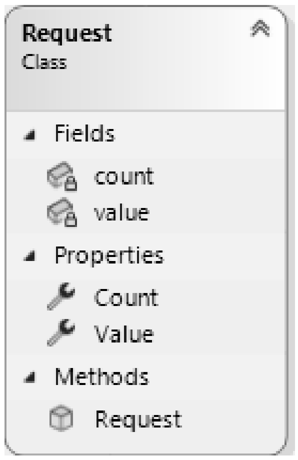
- **Donor:**
  It is the child class of person, and it contains respective attributes of the donor.



- **Recipient:**
  It is the child class of a person, and it contains respective attributes of the recipient.



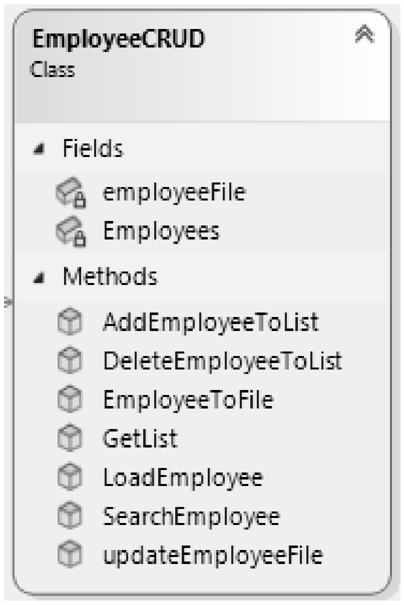
- **Request:**
  It is helping class to count the requests for respective blood group.



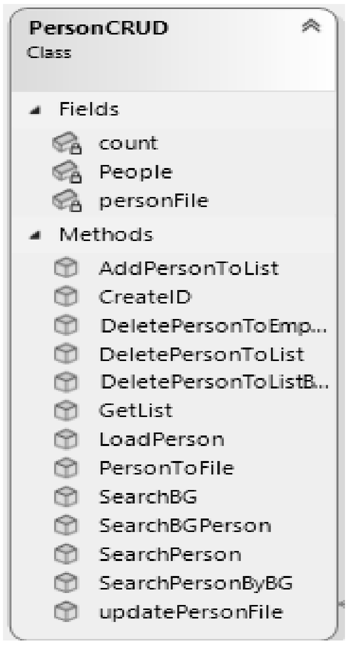
5.2 <u>**Data Access Layer (DL) Classes:**</u>

- **EmployeeCRUD:**

This class is responsible for data handling of employees. It stores data into file when changes are held or add new employees and loads data when application starts.

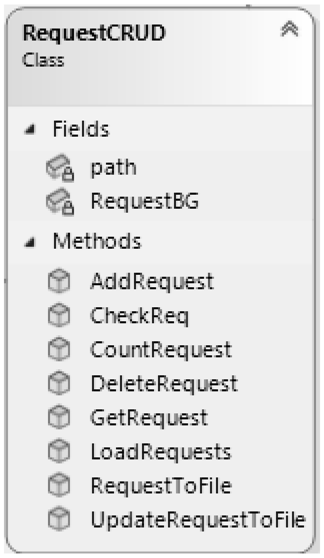


- **PersonCRUD:**
  This class is responsible for data handling of person i.e., donors and recipients it stores data into file when changes held or add new person and load data when application start.



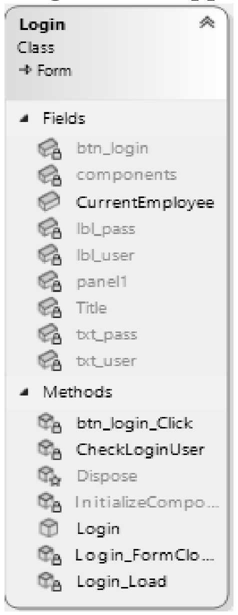
- **RequestCRUD:**

This class is responsible for data handling of requests made by guest user it stores request into file when request fulfill it deletes it or add new request when made and load data when application start.



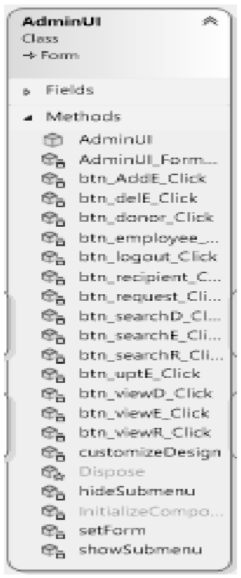
5.3 <u>User Interface (UI) Forms:</u>

- **Login:**
  This class provides user interface for login to the application.

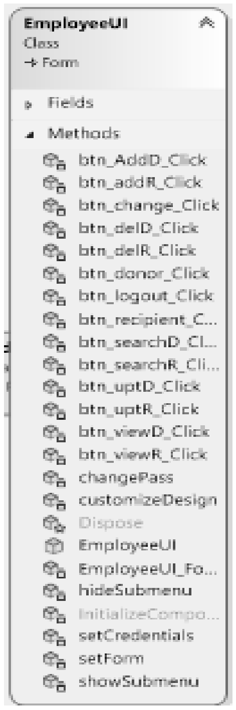


- **AdminUI:**

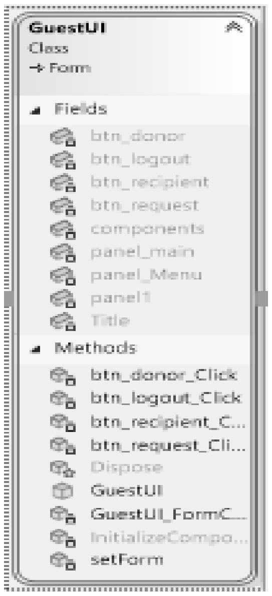
This class provides interface for the employee operations that is adding new employees and different other operations.



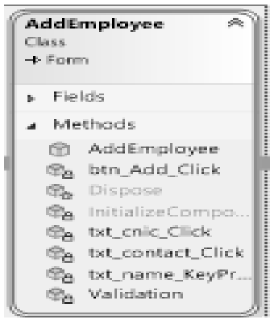
- **EmployeeUI:**
  This class provides interface for the person operations that is show all persons



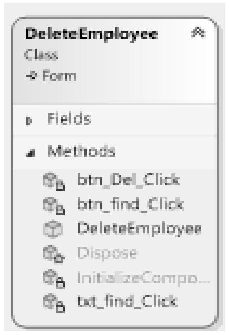
- **GuestUI:**
  This class provides interface for guest account user.



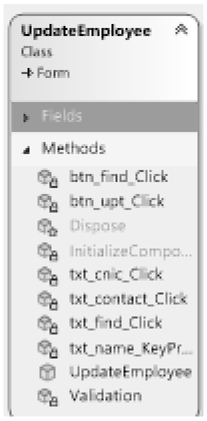
- **AddEmployee:**
  Provide interface for add employee.



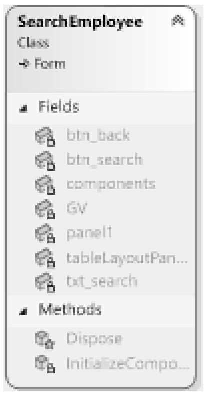
- **DeleteEmployee:**
  Provide interface for delete employee.



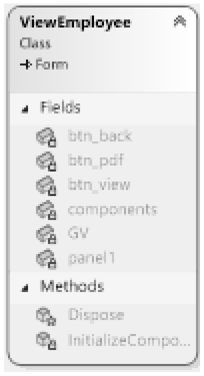
- **UpdateEmployee:**
  Provide interface for update employee.



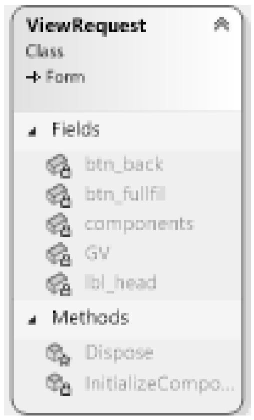
- **SearchEmployee:**
  Provide interface for search employee.



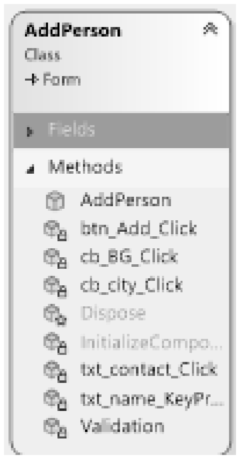
- **ViewEmployee:**
  Provide interface for view employee.



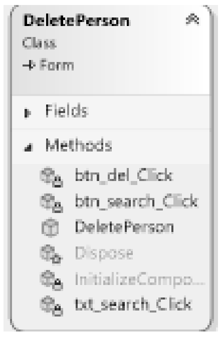
- **ViewRequest:**
  Provide interface for view requests.



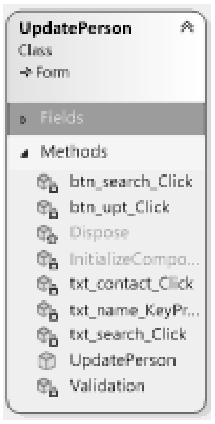
- **AddPerson:**
  Provide interface for add person.



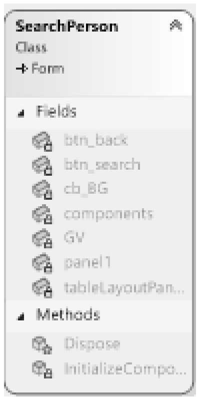
- **DeletePerson:**
  Provide interface for delete person.



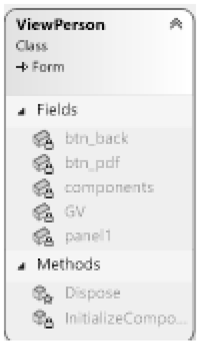
- **UpdatePerson:**
  Provide interface for update person.



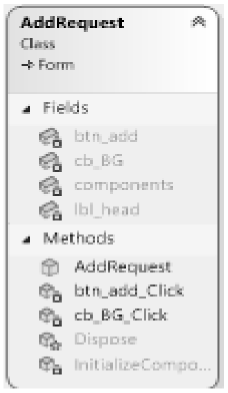
- **SearchPerson:**
  Provide interface for search person.



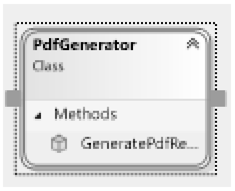
- **ViewPerson:**
  Provide interface for view person.



- **AddRequest:**



- **PdfGenerator:**
  Pdf Generator class



6. Wireframes:

6.1. Login screen:

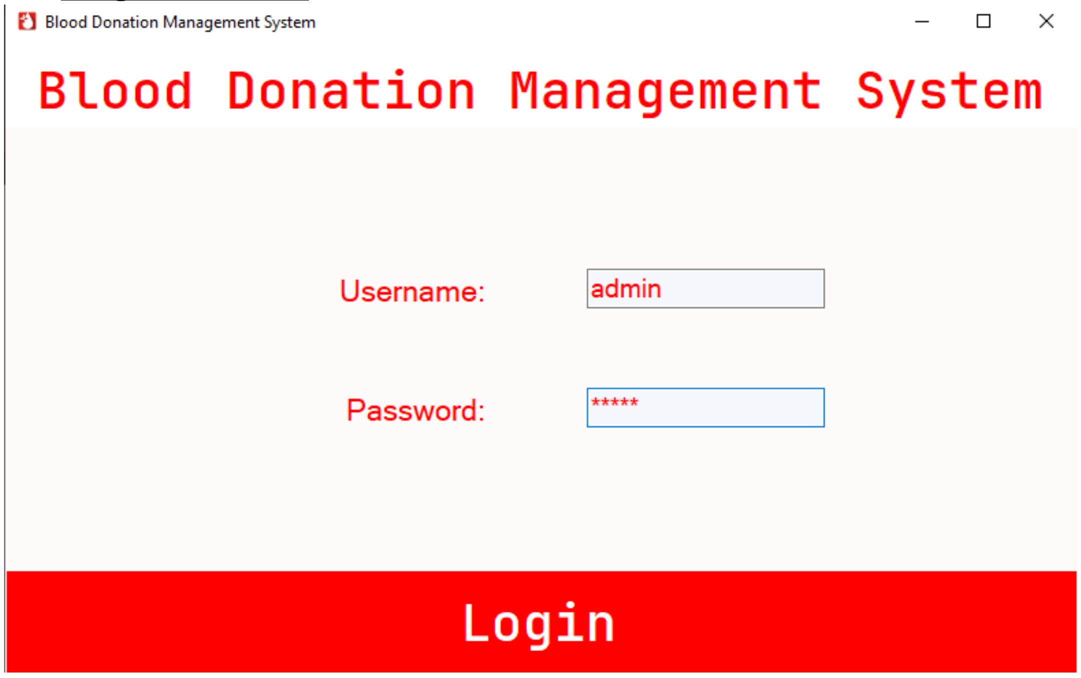

Blood Donation Management System
Blood Donation Management System
Username:
admin
Password:
*****
Login

Figure 1: Login Screen

6.2. Admin screen and Options:

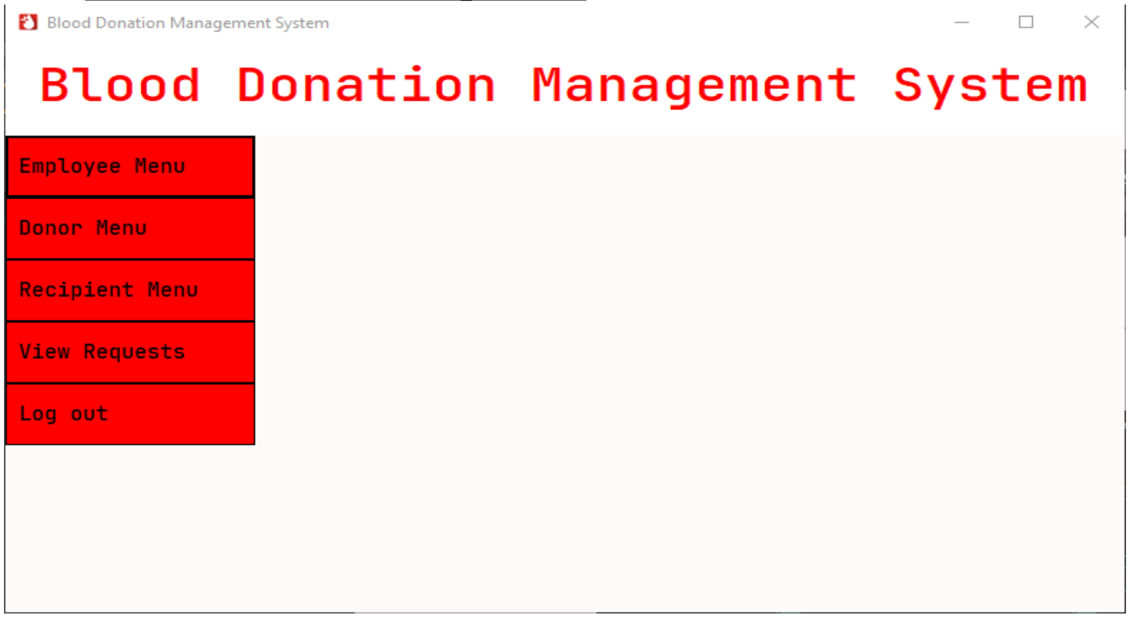
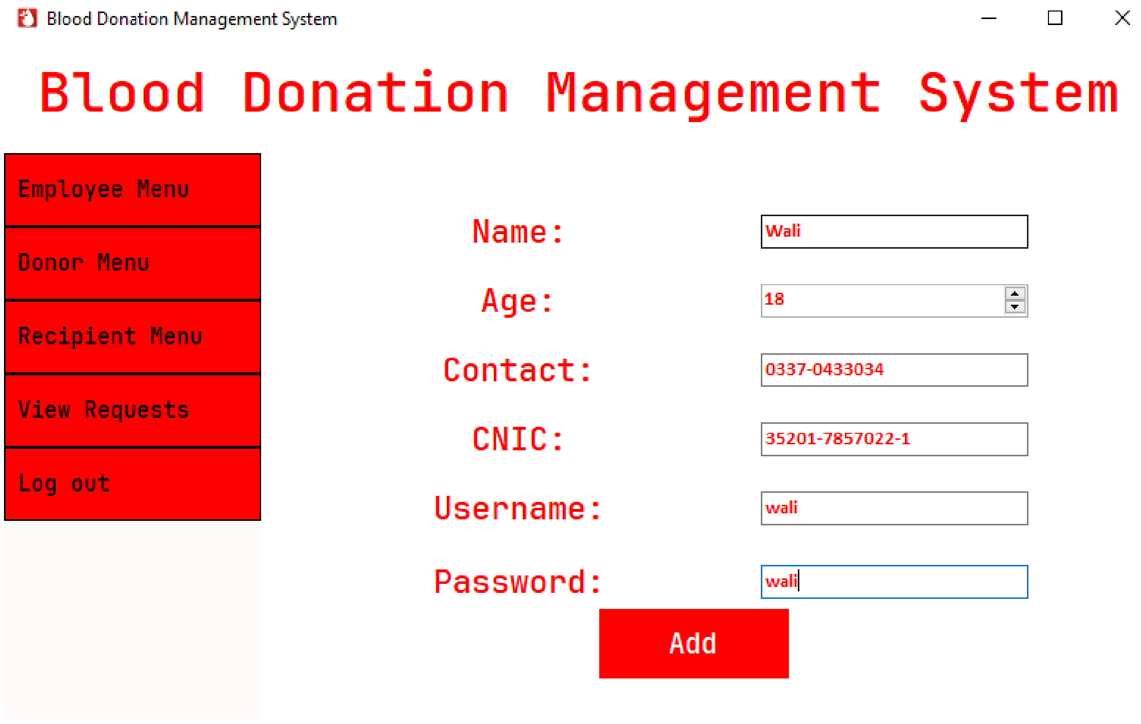
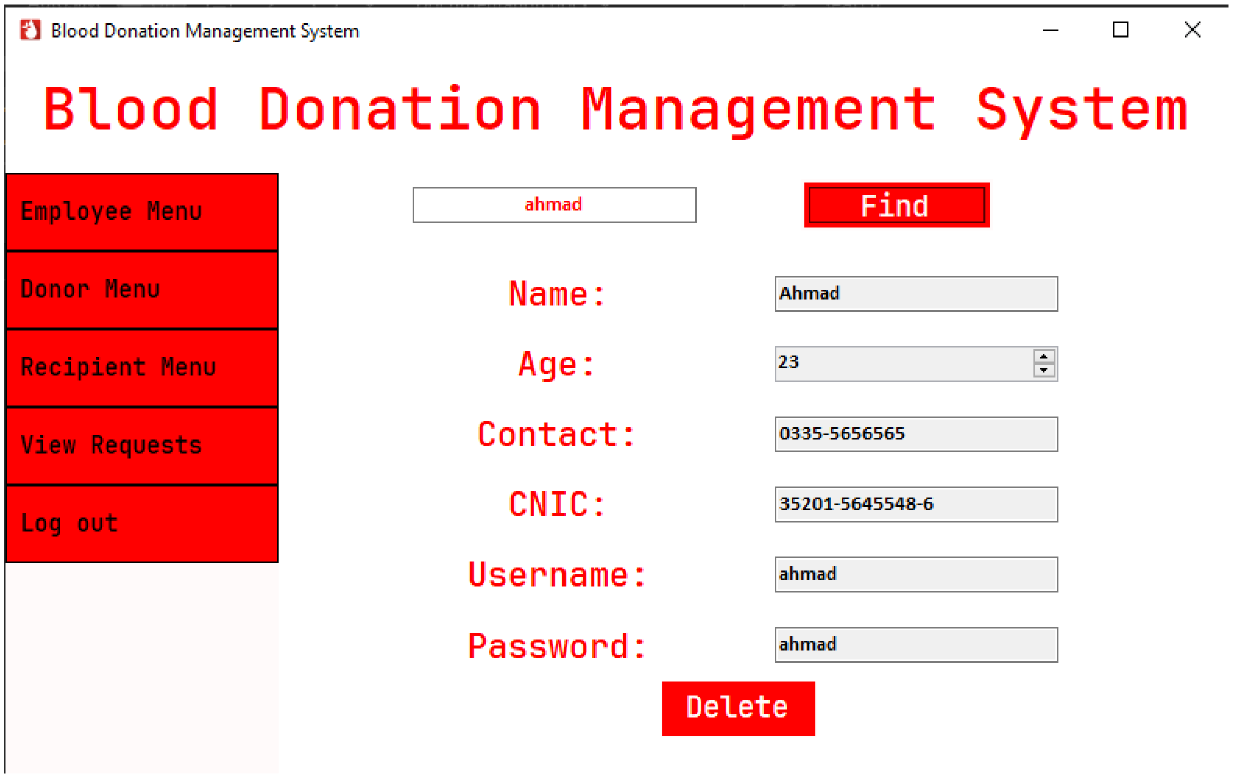

Blood Donation Management System
Blood Donation Management System
Employee Menu
Donor Menu
Recipient Menu
View Requests
Log out

Figure 2: Admin Main Menu Screen


Blood Donation Management System
Blood Donation Management System
Employee Menu
Donor Menu
Recipient Menu
View Requests
Log out
Add

Name:	Wali
Age:	18
Contact:	0337-0433034
CNIC:	35201-7857022-1
Username:	wali
Password:	wali

Figure 2.1: Add Employee Screen



ahmad

Name: Ahmad

Age: 23

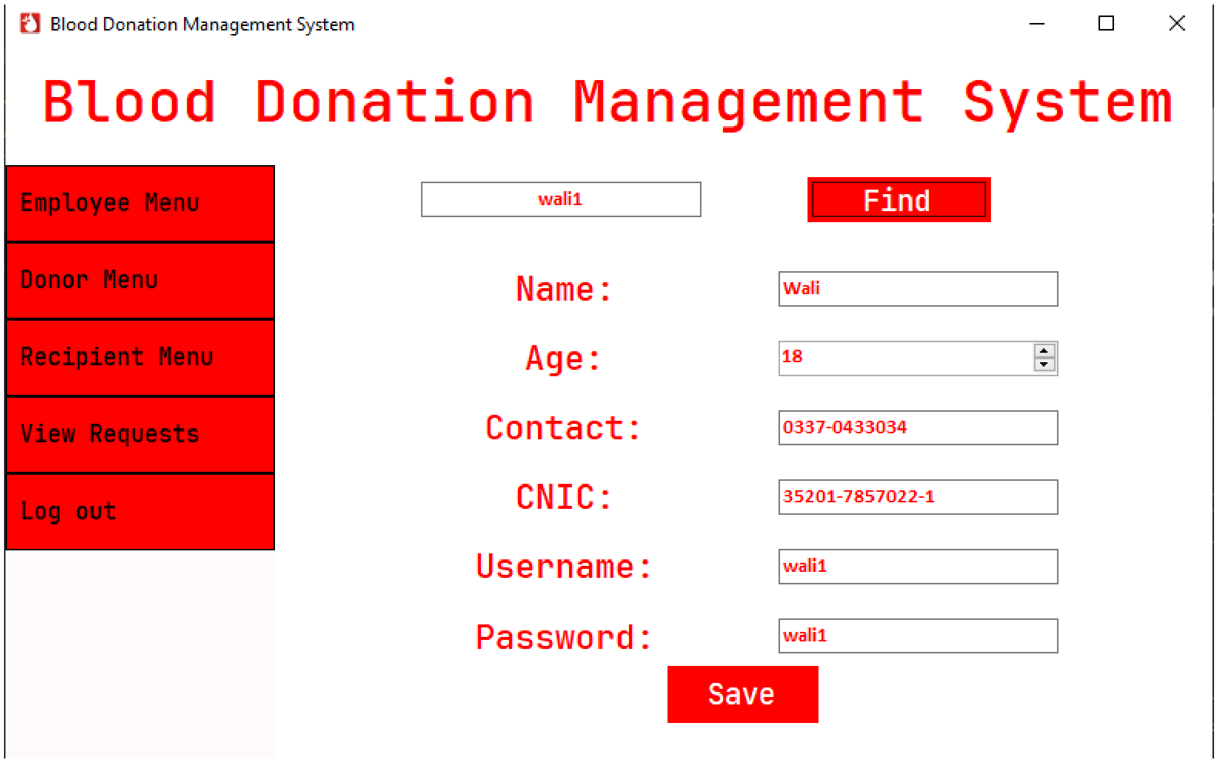
Contact: 0335-5656565

CNIC: 35201-5645548-6

Username: ahmad

Password: ahmad

Figure 2.2: Delete Employee Screen



wali1

Name: Wali

Age: 18

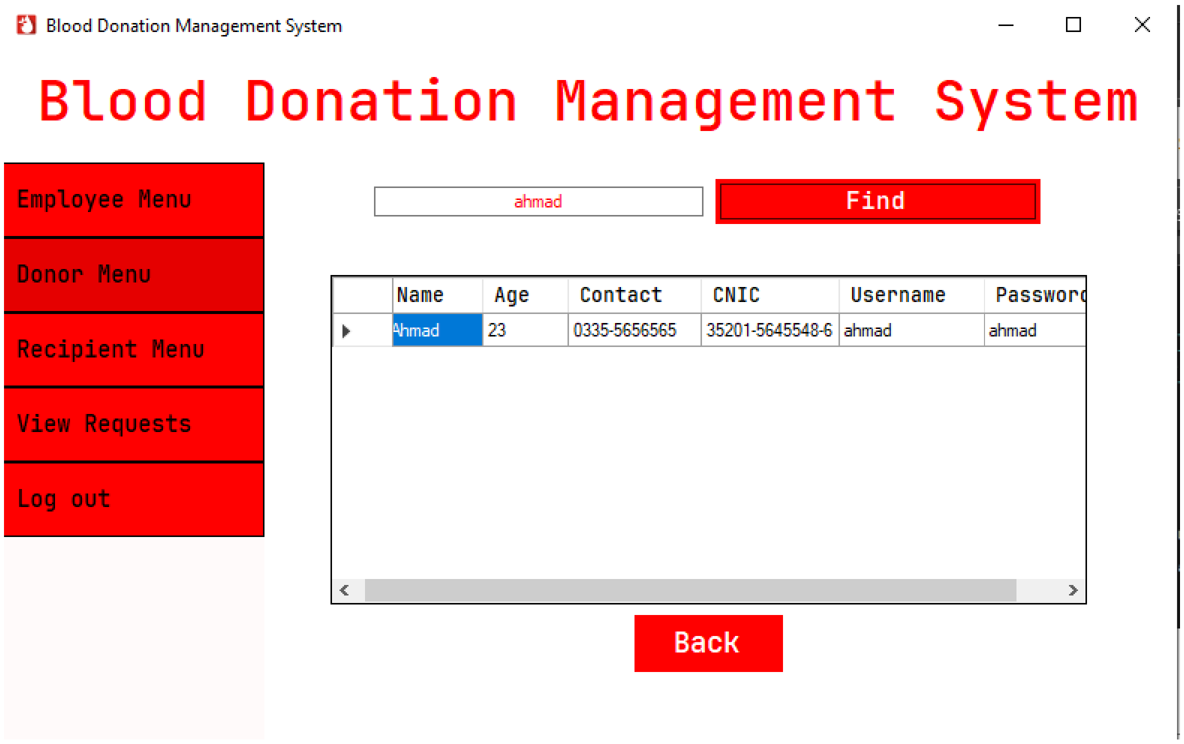
Contact: 0337-0433034

CNIC: 35201-7857022-1

Username: wali1

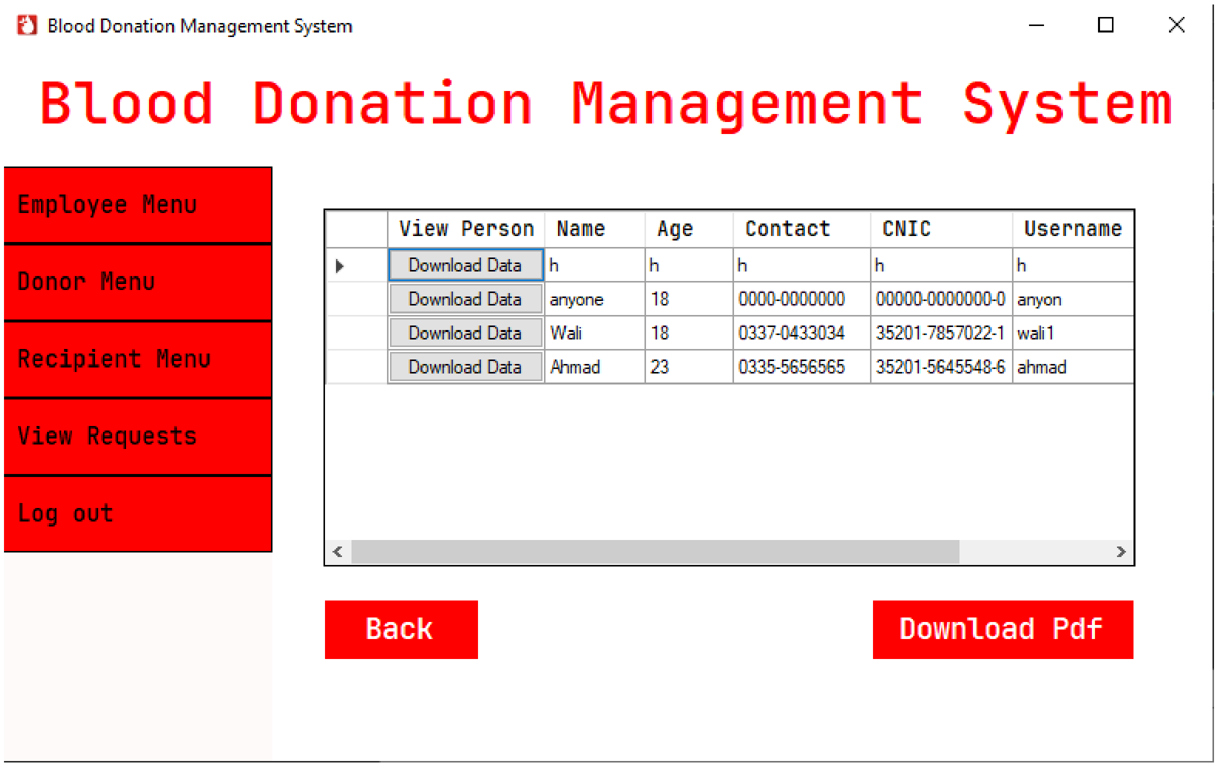
Password: wali1

Figure 2.3: Update Employee Screen



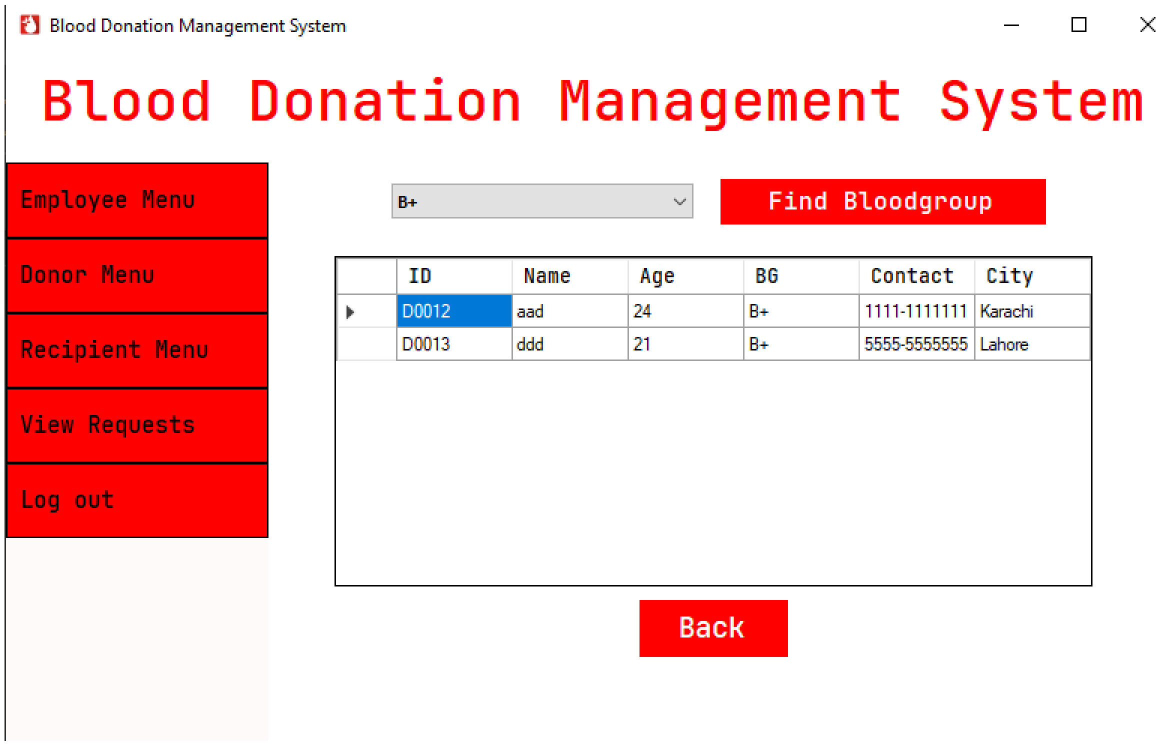
Name	Age	Contact	CNIC	Username	Passwor
Ahmad	23	0335-5656565	35201-5645548-6	ahmad	ahmad

Figure 2.4: Search Employee Screen



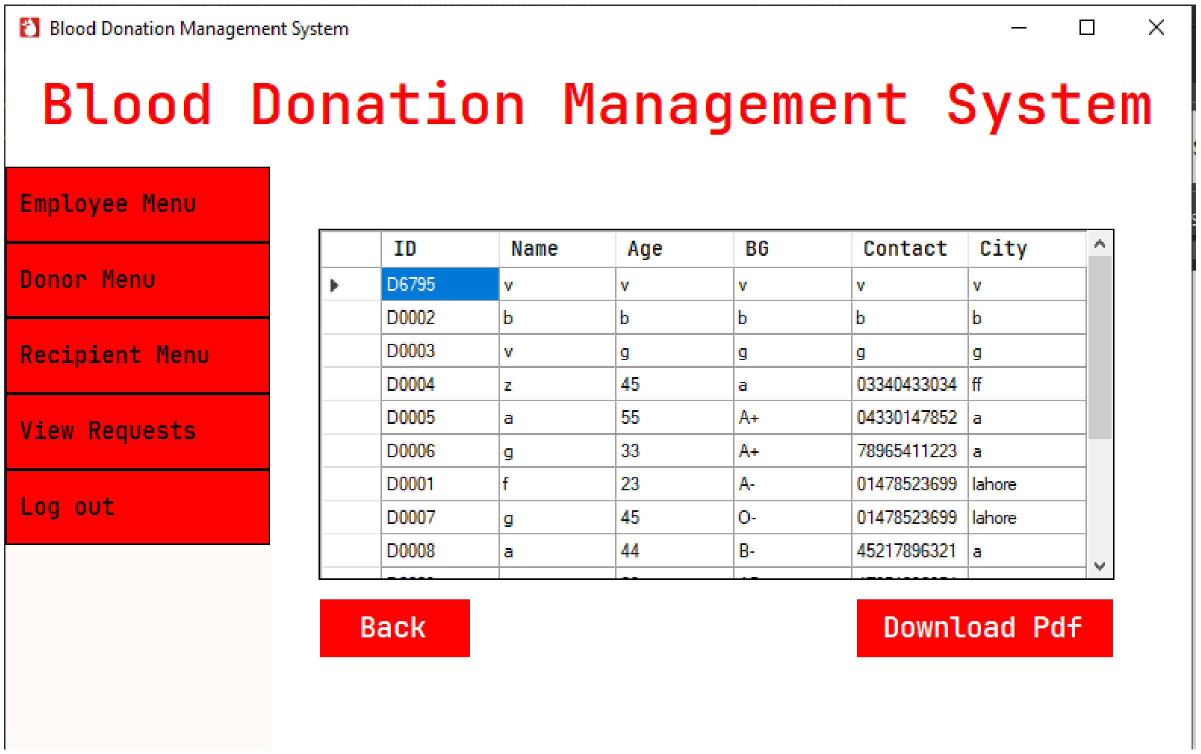
View Person	Name	Age	Contact	CNIC	Username
Download Data	h	h	h	h	h
Download Data	anyone	18	0000-0000000	00000-0000000-0	anyon
Download Data	Wali	18	0337-0433034	35201-7857022-1	wali1
Download Data	Ahmad	23	0335-5656565	35201-5645548-6	ahmad

Figure 2.5: View All Employee Screen



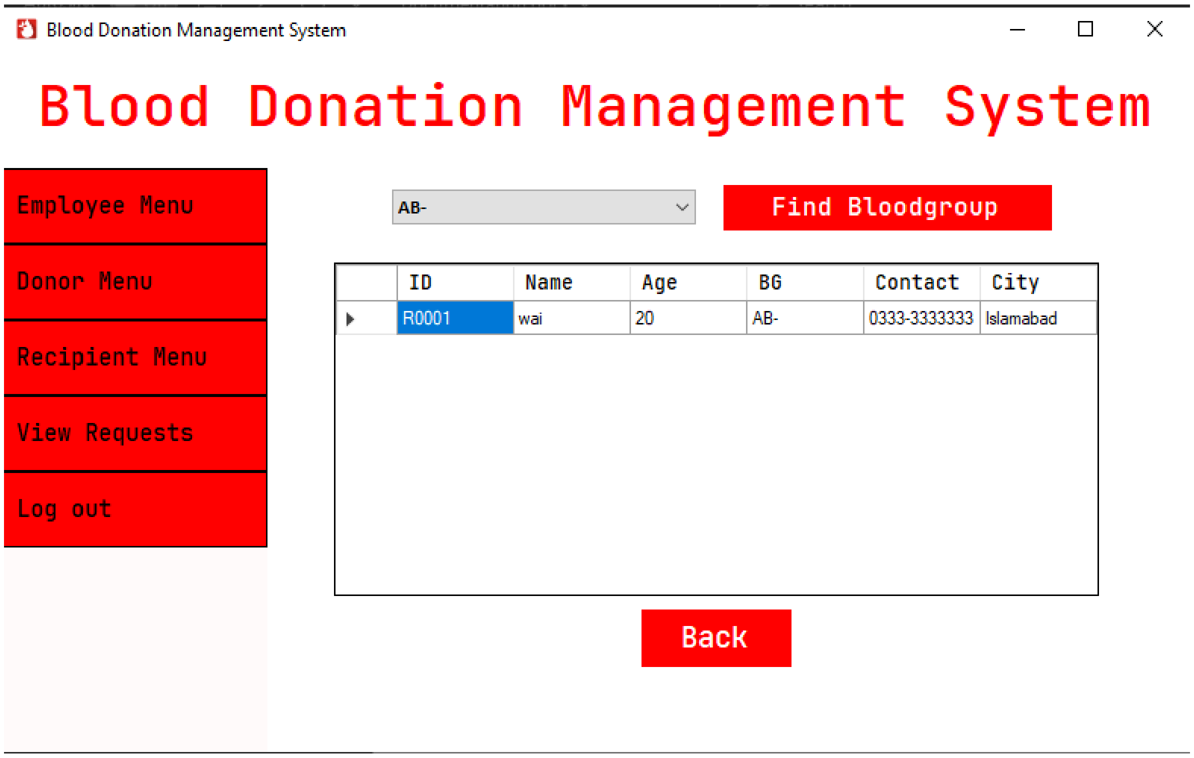
ID	Name	Age	BG	Contact	City
D0012	aad	24	B+	1111-1111111	Karachi
D0013	ddd	21	B+	5555-5555555	Lahore

Figure 2.6: Search Donor Screen



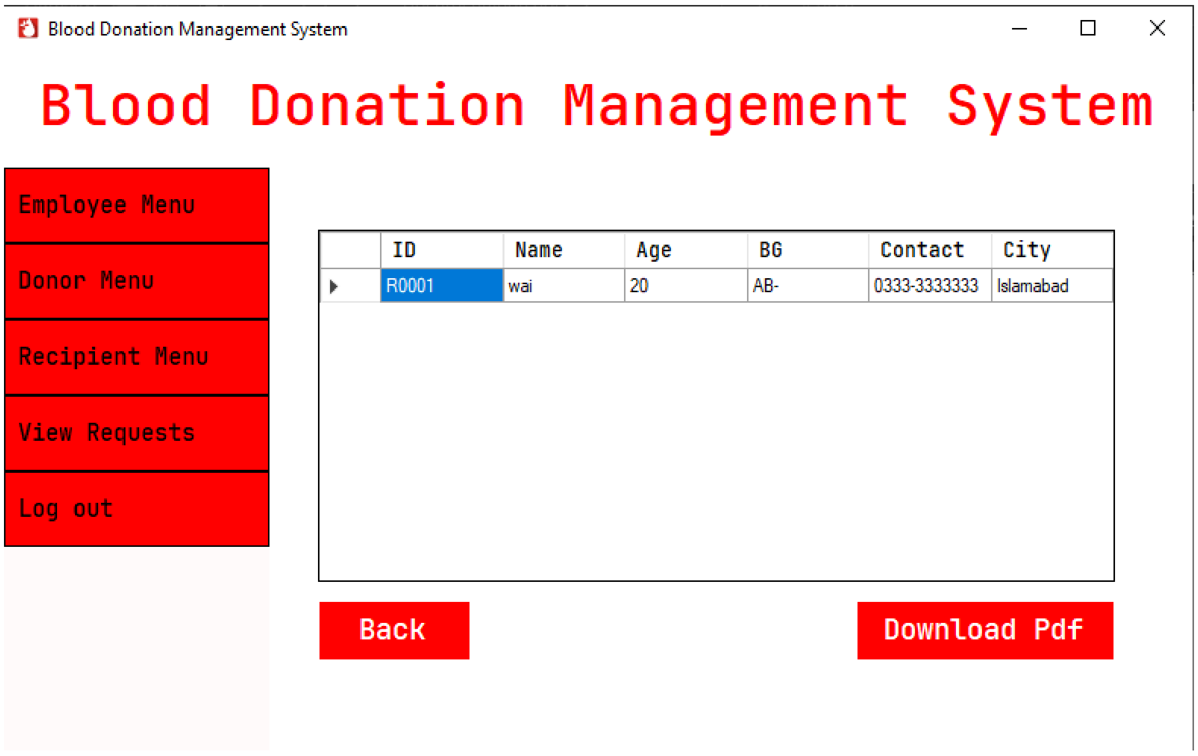
ID	Name	Age	BG	Contact	City
D6795	v	v	v	v	v
D0002	b	b	b	b	b
D0003	v	g	g	g	g
D0004	z	45	a	03340433034	ff
D0005	a	55	A+	04330147852	a
D0006	g	33	A+	78965411223	a
D0001	f	23	A-	01478523699	lahore
D0007	g	45	O-	01478523699	lahore
D0008	a	44	B-	45217896321	a

Figure 2.7: View All Donor Screen



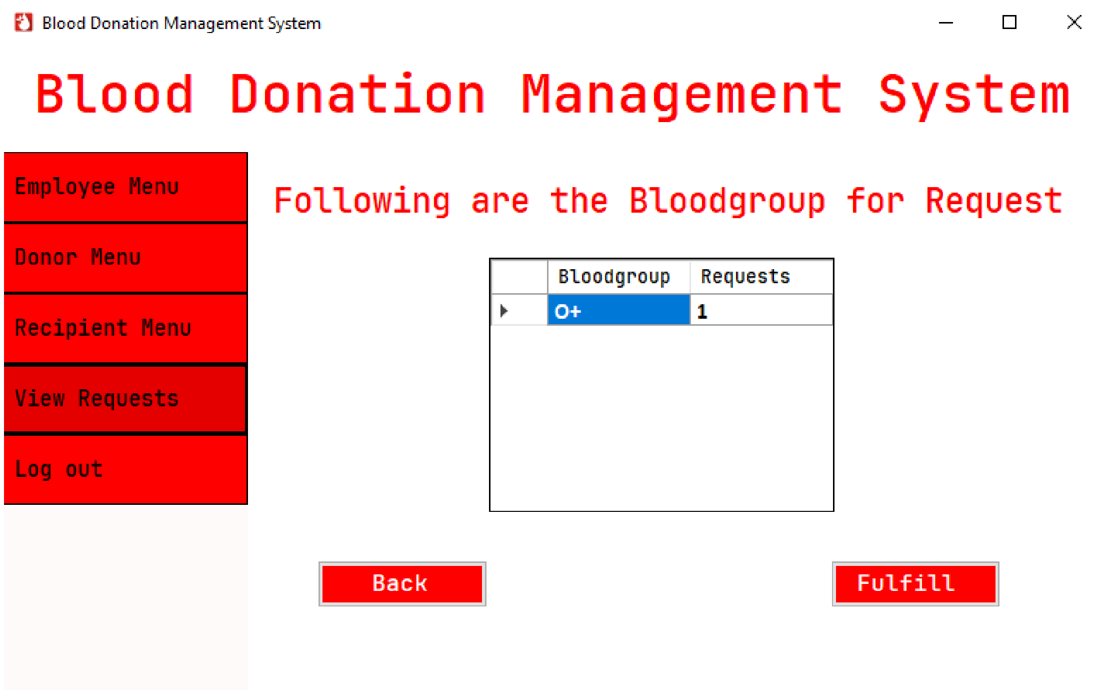
ID	Name	Age	BG	Contact	City
R0001	wai	20	AB-	0333-3333333	Islamabad

Figure 2.8: Search Recipient Screen



ID	Name	Age	BG	Contact	City
R0001	wai	20	AB-	0333-3333333	Islamabad

Figure 2.9: View All Recipients Screen


Blood Donation Management System
Blood Donation Management System
Employee Menu
Donor Menu
Recipient Menu
View Requests
Log out
Following are the Bloodgroup for Request
Back
Fulfill

Bloodgroup	Requests
O+	1

Figure 2.10: View All Requests Screen

6.3. Employee screen and Option:

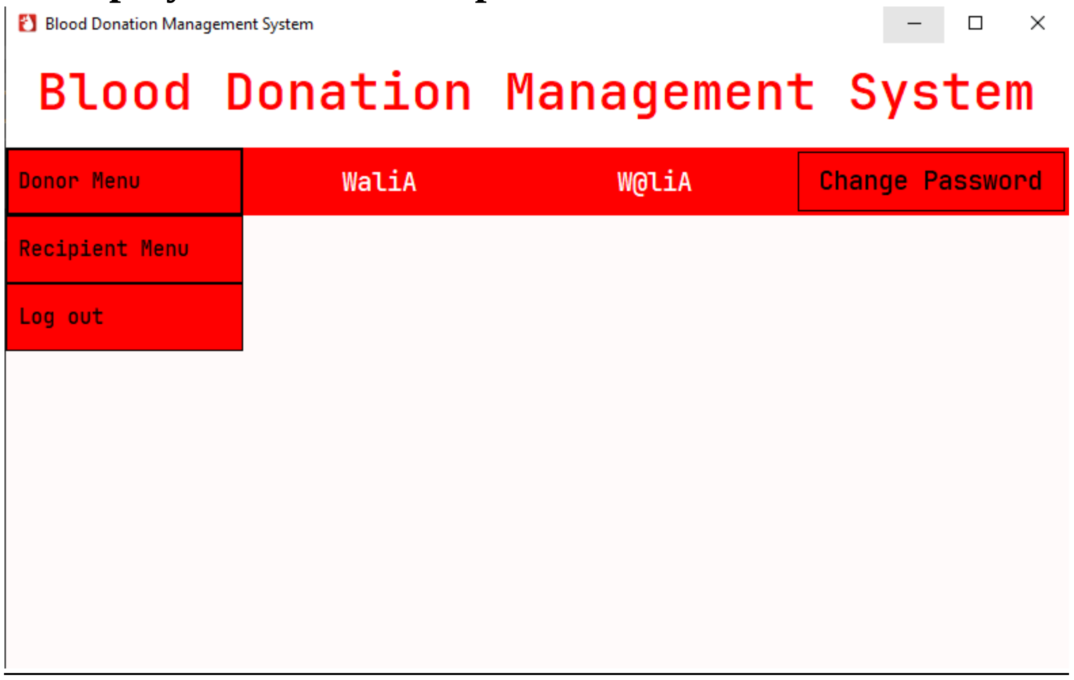
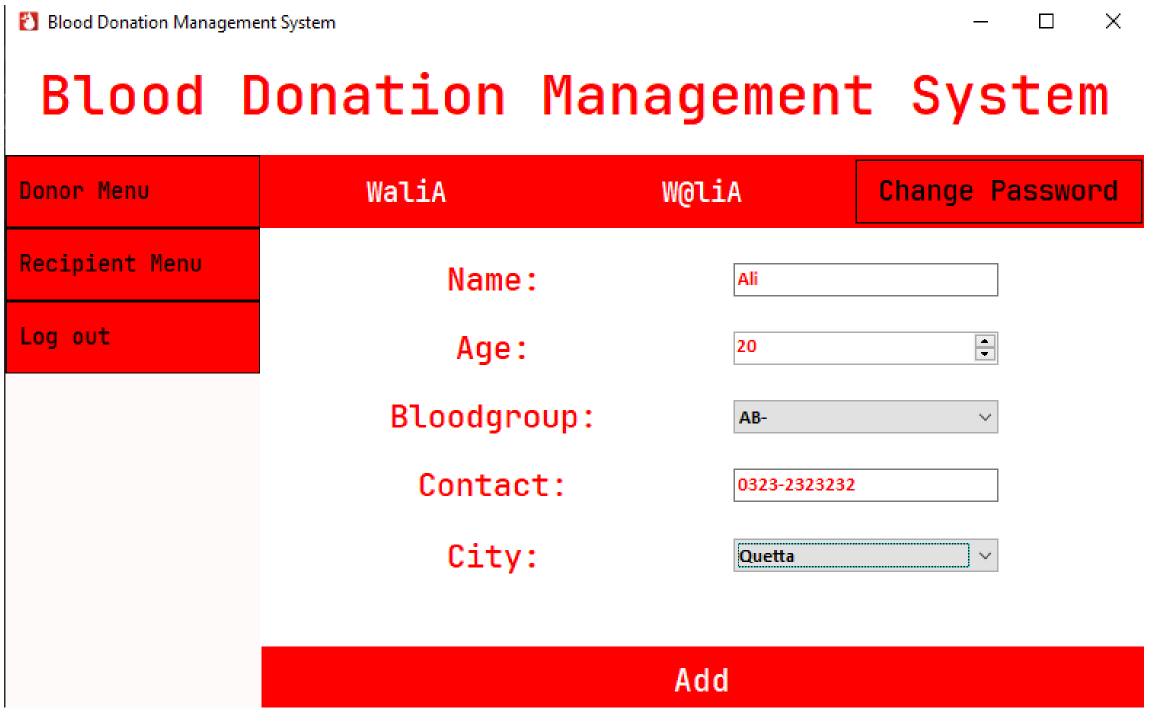
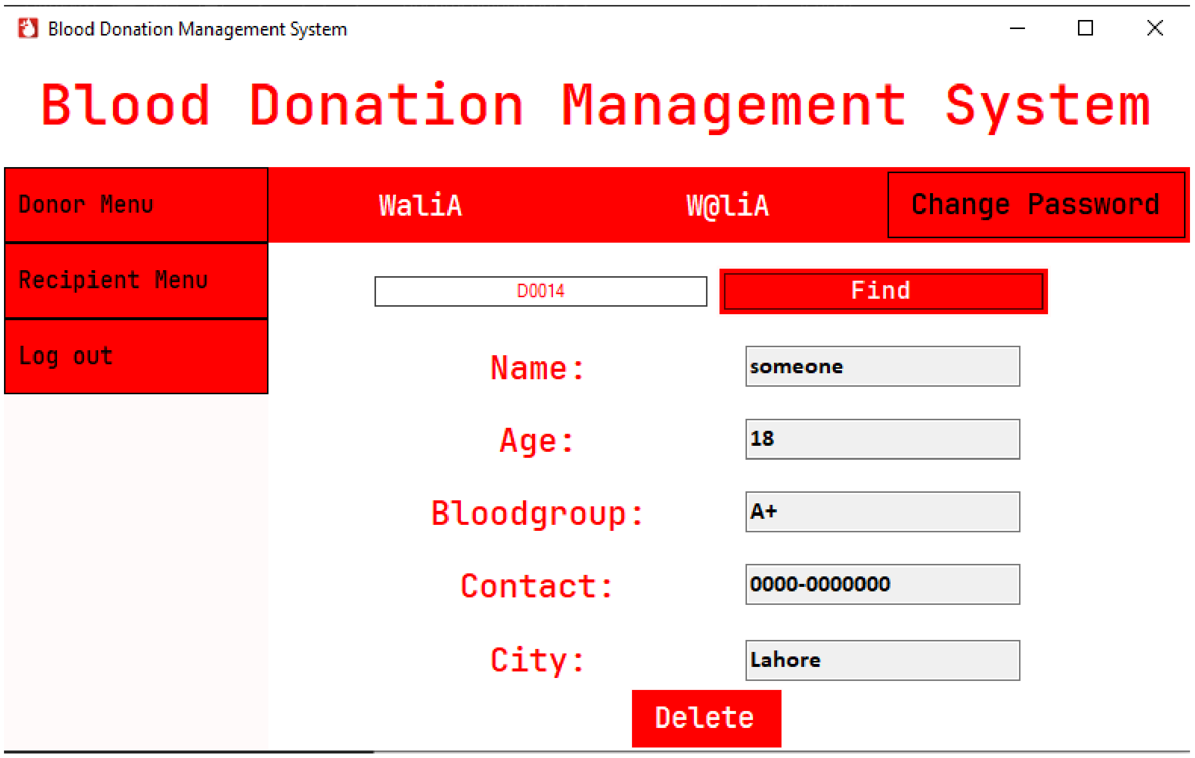

Blood Donation Management System
Blood Donation Management System
Donor Menu
Recipient Menu
Log out
WaliA
W@liA
Change Password

Figure 3: Employee Main Menu Screen


Blood Donation Management System
Blood Donation Management System
Donor Menu
Recipient Menu
Log out
WaliA
W@liA
Change Password
Add

Name: Ali
Age: 20
Bloodgroup: AB-
Contact: 0323-2323232
City: Quetta

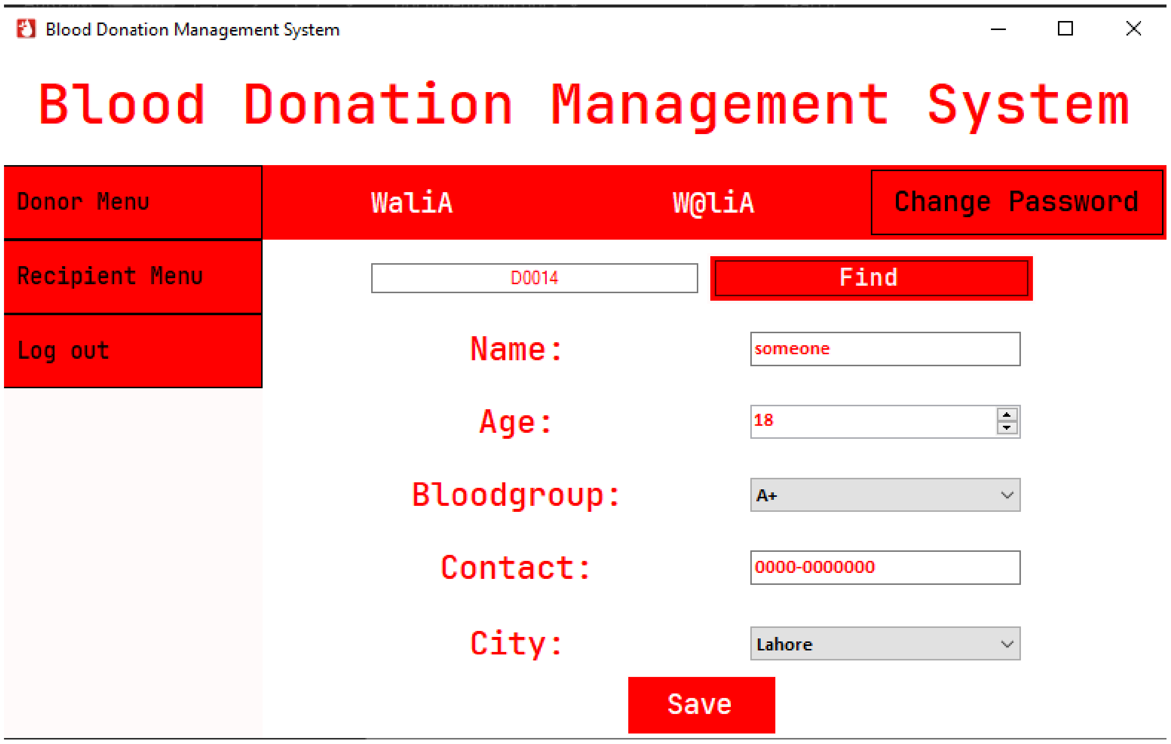
Figure 3.1: Add Donor Screen



D0014

Name: someone
Age: 18
Bloodgroup: A+
Contact: 0000-0000000
City: Lahore

Figure 3.2: Delete Donor Screen


Blood Donation Management System
Blood Donation Management System
Donor Menu
Recipient Menu
Log out
WaliA
W@liA
Change Password
Find
Save

D0014

Name: someone

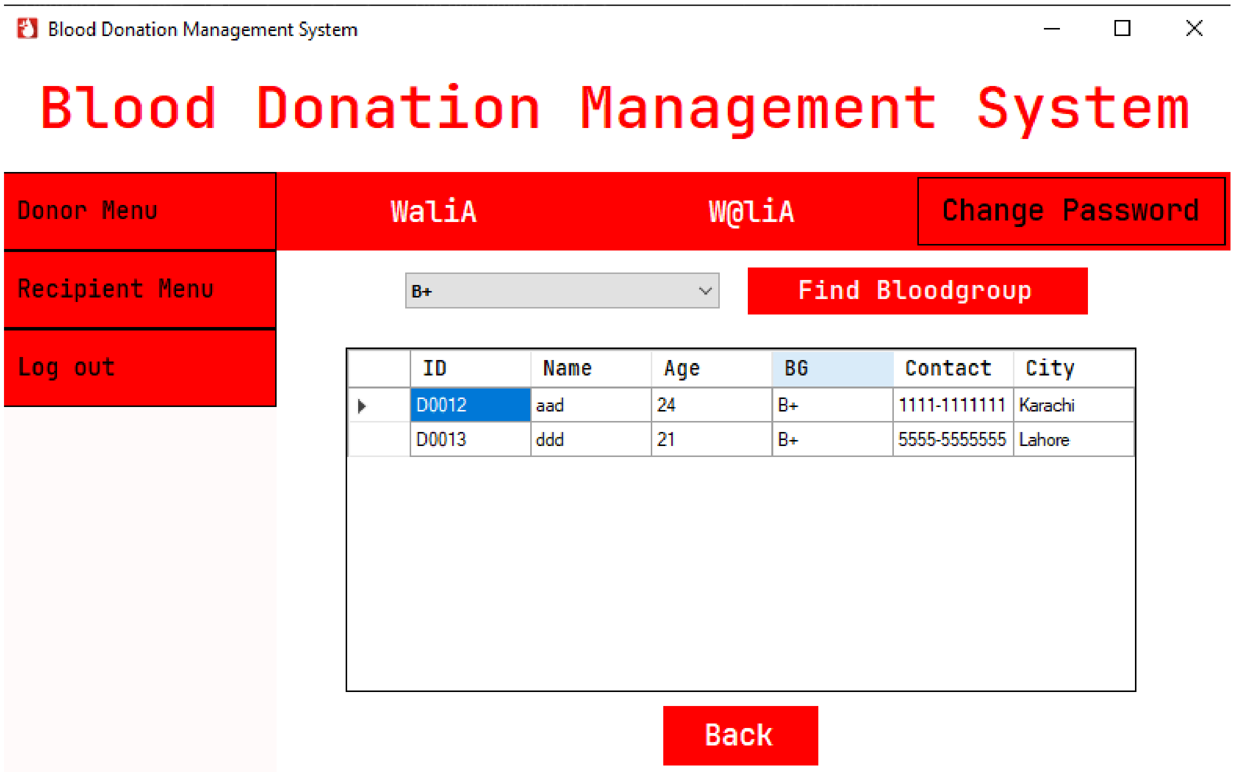
Age: 18

Bloodgroup: A+

Contact: 0000-0000000

City: Lahore

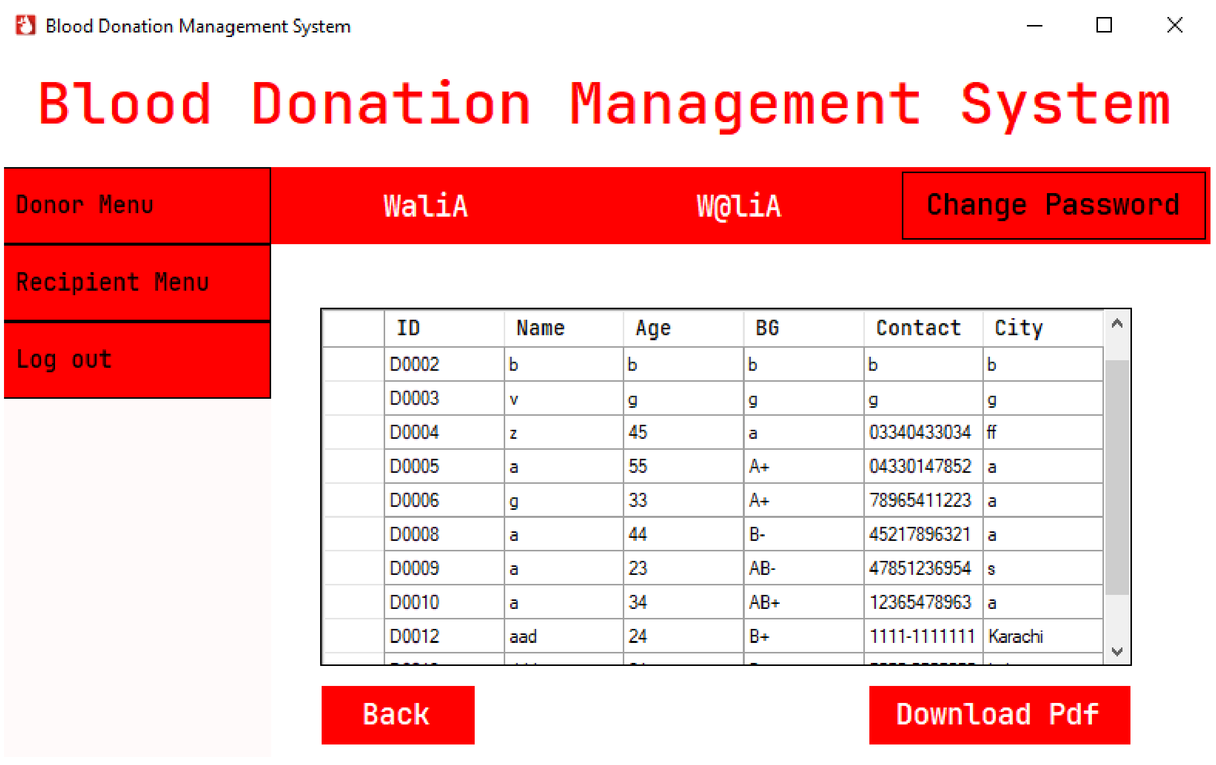
Figure 3.3: Update Donor Screen



B+

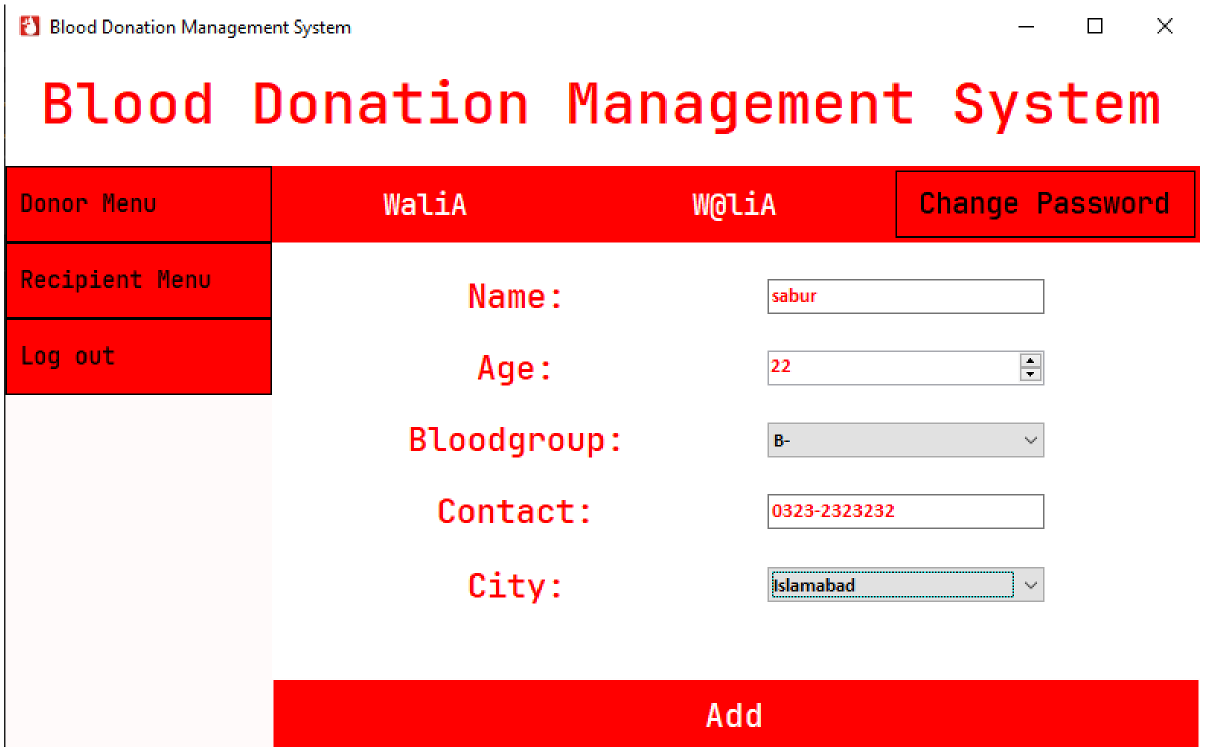
ID	Name	Age	BG	Contact	City
D0012	aad	24	B+	1111-1111111	Karachi
D0013	ddd	21	B+	5555-5555555	Lahore

Figure 3.4: Search Donor Screen



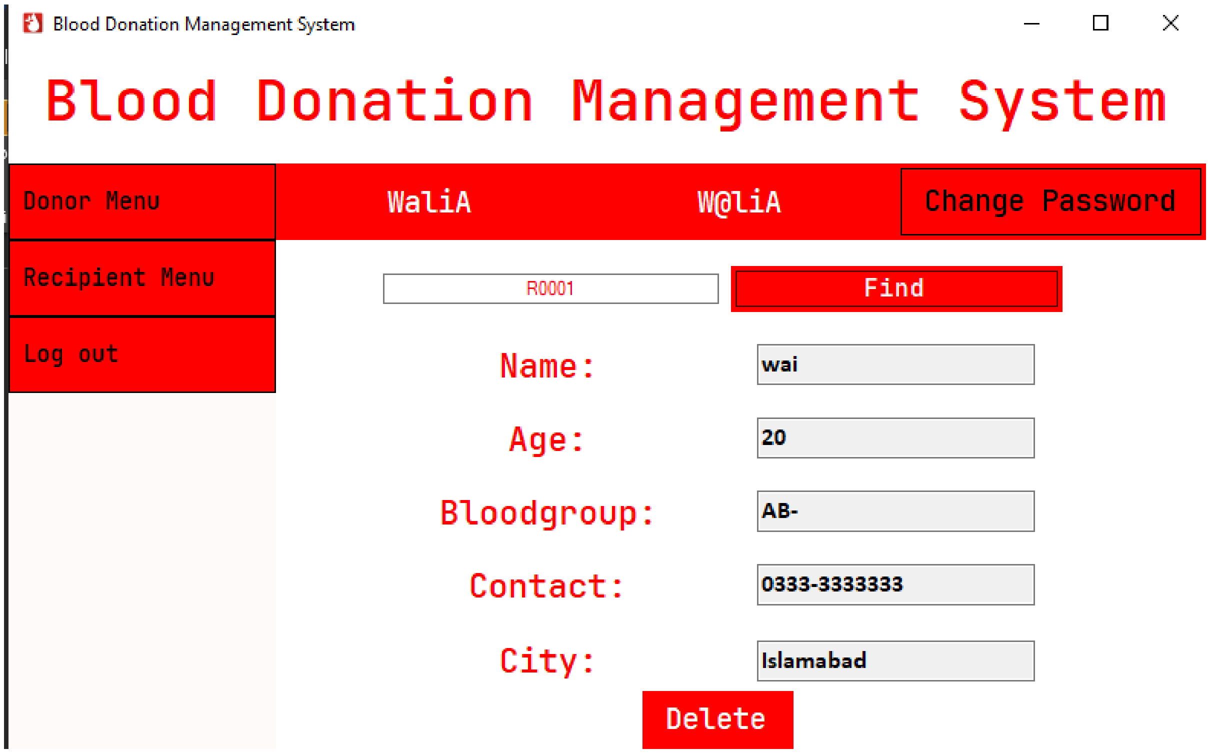
ID	Name	Age	BG	Contact	City
D0002	b	b	b	b	b
D0003	v	g	g	g	g
D0004	z	45	a	03340433034	ff
D0005	a	55	A+	04330147852	a
D0006	g	33	A+	78965411223	a
D0008	a	44	B-	45217896321	a
D0009	a	23	AB-	47851236954	s
D0010	a	34	AB+	12365478963	a
D0012	aad	24	B+	1111-1111111	Karachi

Figure 3.5: View All Donor Screen


Blood Donation Management System
Blood Donation Management System
Donor Menu
WaliA
W@liA
Change Password
Recipient Menu
Log out
Add

Name: sabur
Age: 22
Bloodgroup: B-
Contact: 0323-2323232
City: Islamabad

Figure 3.6: Add Recipient Screen


Blood Donation Management System

Blood Donation Management System

R0001

Name: wai

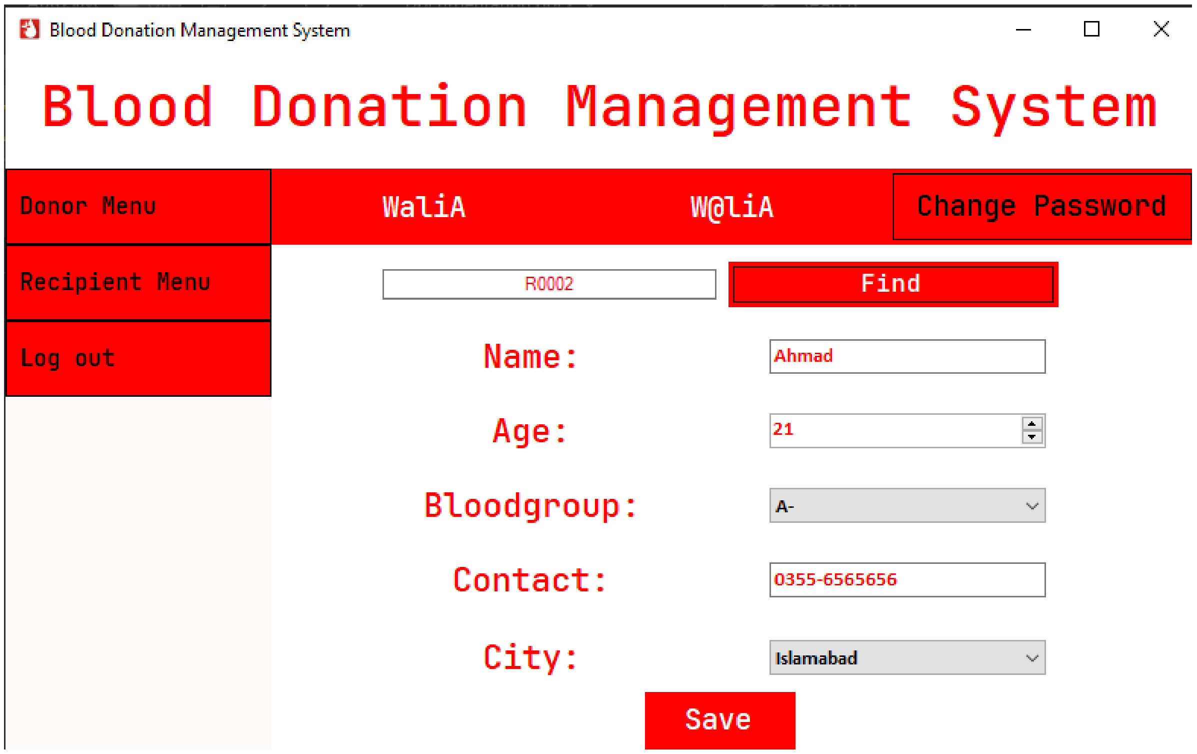
Age: 20

Bloodgroup: AB-

Contact: 0333-3333333

City: Islamabad

Figure 3.7: Delete Recipient Screen


Blood Donation Management System

Blood Donation Management System

R0002

Name: Ahmad

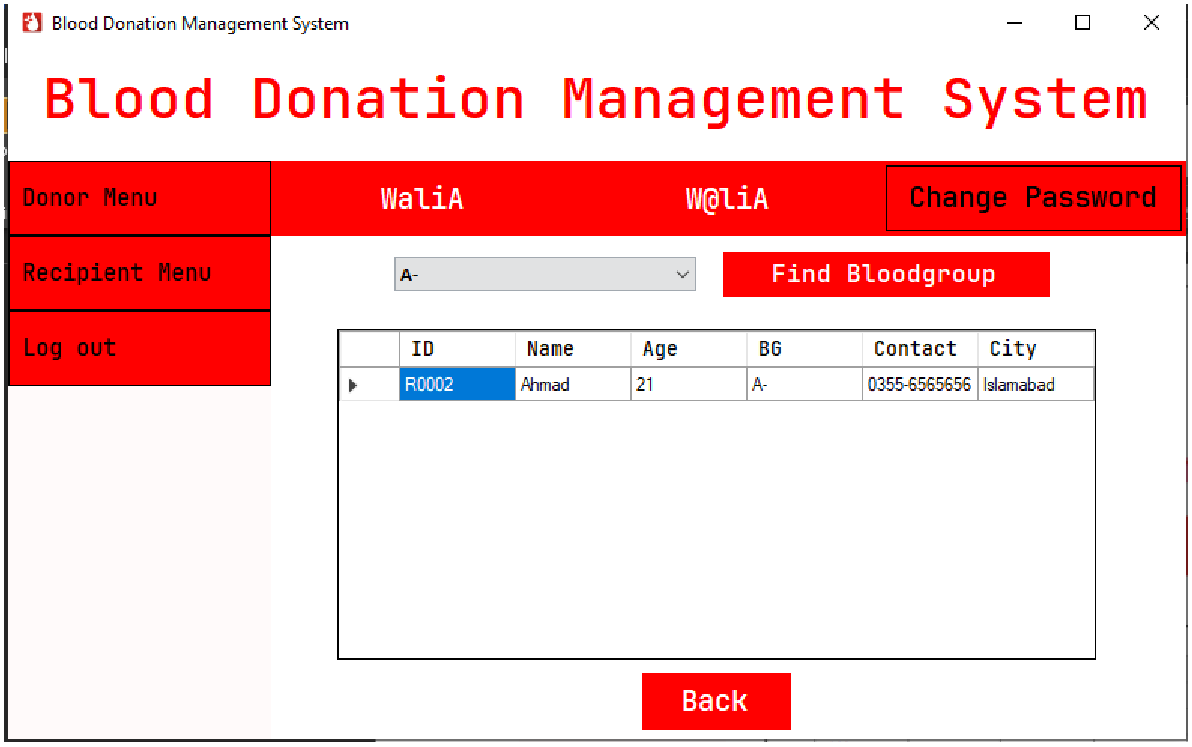
Age: 21

Bloodgroup: A-

Contact: 0355-6565656

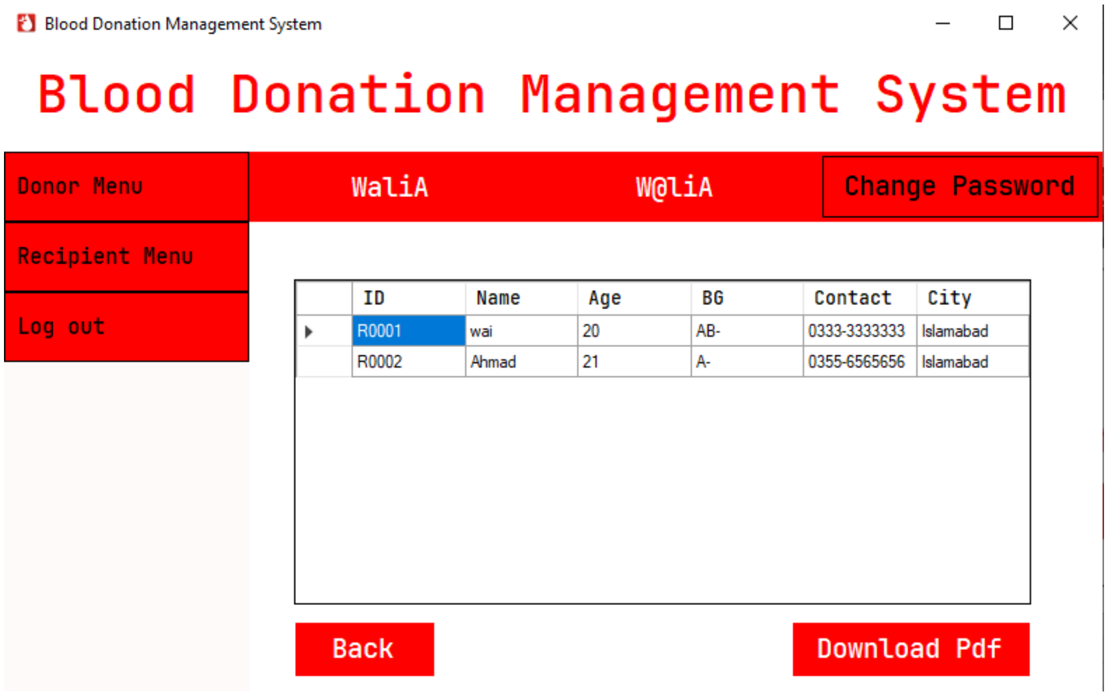
City: Islamabad

Figure 3.8: Update Recipient Screen



ID	Name	Age	BG	Contact	City
R0002	Ahmad	21	A-	0355-6565656	Islamabad

Figure 3.9: Search Recipient Screen



ID	Name	Age	BG	Contact	City
R0001	wai	20	AB-	0333-3333333	Islamabad
R0002	Ahmad	21	A-	0355-6565656	Islamabad

Figure 3.10: View All Recipients Screen

6.4. Guest screen and Option:

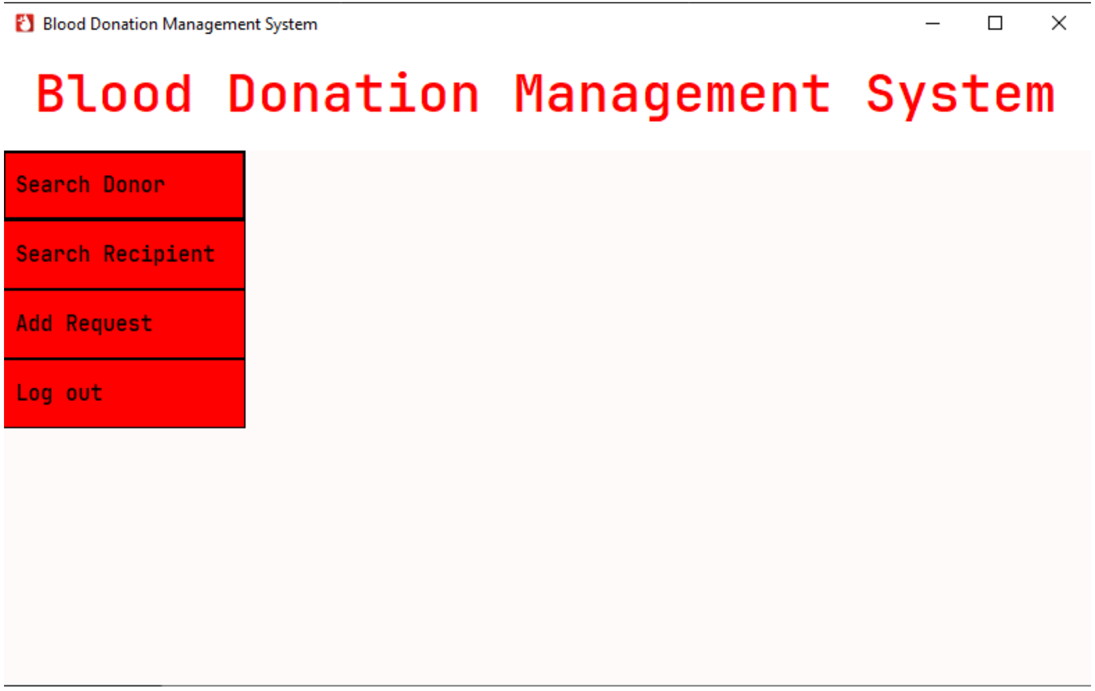
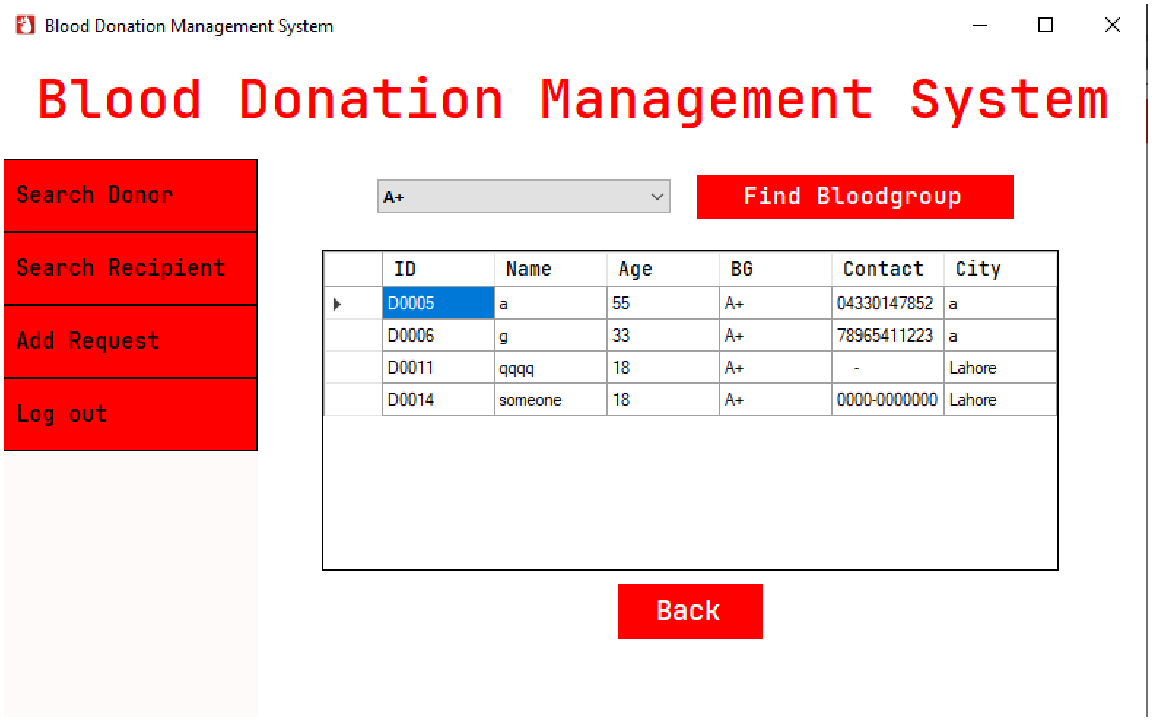
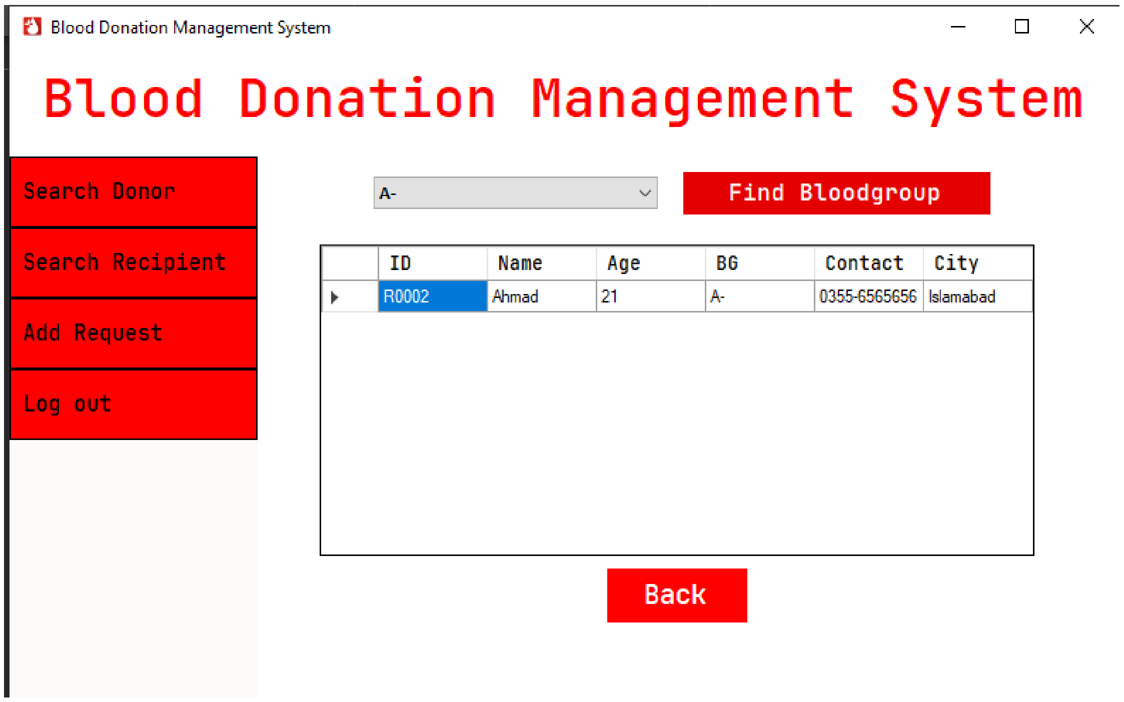

Blood Donation Management System
Blood Donation Management System
Search Donor
Search Recipient
Add Request
Log out

Figure 4: Guest Main Menu Screen



ID	Name	Age	BG	Contact	City
D0005	a	55	A+	04330147852	a
D0006	g	33	A+	78965411223	a
D0011	qqqq	18	A+	-	Lahore
D0014	someone	18	A+	0000-0000000	Lahore

Figure 4.1: Search Donor Screen



ID	Name	Age	BG	Contact	City
R0002	Ahmad	21	A-	0355-6565656	Islamabad

Figure 4.2: Search Recipient Screen

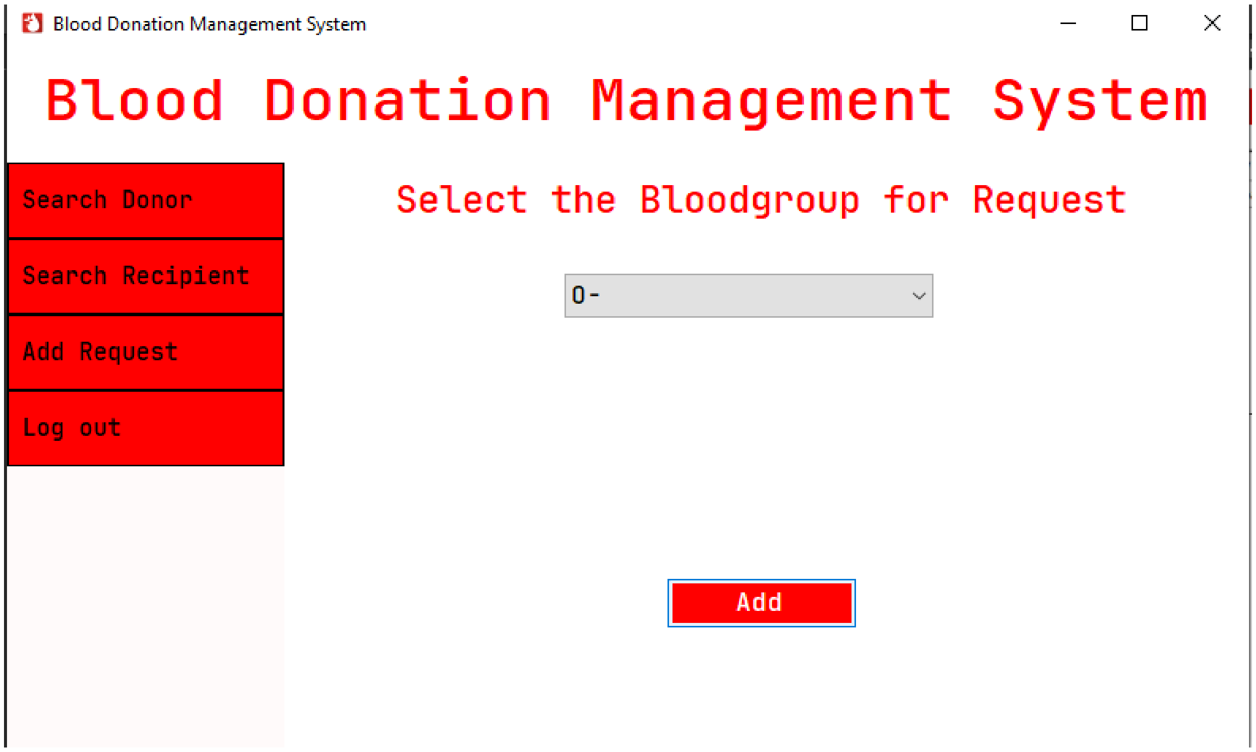

Blood Donation Management System
Blood Donation Management System
Search Donor
Search Recipient
Add Request
Log out
Select the Bloodgroup for Request
O-
Add

Figure 4.3: Add Request Screen

7. COMPLETE CODE:

7.1 Business Logic (BL) Classes:

- **User:**

```
public class User
    {
        public User(string Username, string Password)
        {
            this.Username = Username;
            this.Password = Password;
        }
        protected string username;
        protected string password;

        public string Username { get => username; set => username = value; }
        public string Password { get => password; set => password = value; }

        public User CheckRole()
        {
            if (Username == "admin" && Password == "admin")
            {
                return new Admin();
            }
            else if (Username == "guest" && Password == "guest")
            {
                return new Guest();
            }
            else
            {
                for (int i = 0; i < EmployeeCRUD.GetList().Count; i++)
                {
                    if (Username == EmployeeCRUD.GetList()[i].GetUsername()
&& Password == EmployeeCRUD.GetList()[i].GetPassword())
                    {
                        return EmployeeCRUD.GetList()[i]; // for employee
                    }
                }
            }
            return null;
        }
    }
```

- **Admin:**

```
public class Admin : User
    {
        public Admin() : base("admin", "admin")
        {
        }
    }
```

- **Employee:**

```
public class Employee : User
    {
        public Employee(string Username, string Password) : base(Username, 
Password)
        {

        }
        public Employee(string Name, string Age, string Contact, string Cnic, 
string Username, string Password) : this(Username, Password)
        {
            this.Name1 = Name;
            this.Age1 = Age;
            this.Cnic1 = Cnic;
            this.Contact1 = Contact;
            People = new List<Person>();
        }

        private string Name;
        private string Age;
        private string Contact;
        private string cnic;
        private List<Person> People;

        public string Name1 { get => Name; set => Name = value; }
        public string Age1 { get => Age; set => Age = value; }
        public string Contact1 { get => Contact; set => Contact = value; }
        public string Cnic1 { get => cnic; set => cnic = value; }

        public string GetName()
        {
            return Name1;
        }
        public void SetName(string Name)
        {
            this.Name1 = Name;
        }
        public string GetAge()
        {
            return Age1;
        }
        public void SetAge(string Age)
        {
            this.Age1 = Age;
        }
        public string GetCnic()
        {
            return Cnic1;
        }
        public void SetCnic(string Cnic)
        {
            this.Cnic1 = Cnic;
        }
        public string GetContact()
        {
            return Contact1;
        }
        public void SetContact(string Contact)
        {
            this.Contact1 = Contact;
        }
        public string GetUsername()
        {
            return Username;
        }
        public void SetUsername(string Username)
        {
            this.Username = Username;
        }
        public string GetPassword()
        {
            return Password;
        }
        public void SetPassword(string Password)
        {
            this.Password = Password;
        }
        public List<Person> GetPeople()
        {
            return People;
        }
        public void AddPerson(Person P)
        {
            People.Add(P);
        }
        public void DeletePerson(Person P)
        {
            People.Remove(P);
        }
        public Person SearchPerson(string ID)
        {
            for (int idx = 0; idx < People.Count; idx++)
            {
                if (ID == People[idx].GetID())
                {
                    return People[idx];
                }
            }
            return null;
        }
        public List<Person> SearchPersonByBG(string BG, char P)
        {
            List<Person> lst = new List<Person>();
            for (int i = 0; i < People.Count; i++)
            {
                if (People[i].GetBG() == BG && People[i].GetID()[0] == P)
                {
                    lst.Add(People[i]);
                }
            }
            return lst;
        }
    }
```

- **Guest:**

```
public class Guest : User
    {
        public Guest() : base("guest", "guest")
        {

        }

    }
```

- **Person:**

```
public class Person
    {
        public Person(string Name, string Age, string BG, string Contact,
string City)
        {
            this.Name = Name;
            this.Age = Age;
            this.BG = BG;
            this.Contact = Contact;
            this.City = City;
        }

        public virtual string ID1 { get; set; }
        public string Name { get => name; set => name = value; }
        public string Age { get => age; set => age = value; }
        public string BG { get => bG; set => bG = value; }
        public string Contact { get => contact; set => contact = value; }
        public string City { get => city; set => city = value; }

        private string name;
        private string age;
        private string bG;
        private string contact;
        private string city;
        public virtual string GetName()
        {
            return Name;
        }
        public virtual void SetName(string Name)
        {
            this.Name = Name;
        }
        public virtual string GetAge()
```

```
            {
                return Age;
            }
            public virtual void SetAge(string Age)
            {
                this.Age = Age;
            }
            public virtual string GetBG()
            {
                return BG;
            }
            public virtual void SetBG(string BG)
            {
                this.BG = BG;
            }
            public virtual string GetContact()
            {
                return Contact;
            }
            public virtual void SetContact(string Contact)
            {
                this.Contact = Contact;
            }
            public virtual string GetCity()
            {
                return City;
            }
            public virtual void SetCity(string City)
            {
                this.City = City;
            }
            public virtual void SetID(string ID)
            {
            }
            public virtual string GetID()
            {
                return "";
            }
        }
```

- **Donor:**

```
public class Donor : Person
    {
        public Donor(string Name, string Age, string BG, string Contact,
string City) : base(Name, Age, BG, Contact, City)
        {
        }
        private string ID;

        public override string ID1 { get => ID; set => ID = value; }

        public override string GetName()
        {
            return Name;
        }
        public override void SetName(string Name)
        {
            this.Name = Name;
        }
        public override string GetAge()
        {
            return Age;
        }
        public override void SetAge(string Age)
        {
            this.Age = Age;
        }
        public override string GetBG()
        {
            return BG;
        }
        public override void SetBG(string BG)
        {
            this.BG = BG;
        }
        public override string GetContact()
        {
            return Contact;
        }
        public override void SetContact(string Contact)
        {
            this.Contact = Contact;
        }
        public override string GetCity()
        {
            return City;
        }
        public override void SetCity(string City)
        {
            this.City = City;
        }
        public override string GetID()
        {
            return ID1;
        }
        public override void SetID(string ID)
        {
            this.ID1 = ID;
        }
 }
```

- **Recipient:**

```
public class Recipient : Person
    {
        public Recipient(string Name, string Age, string BG, string Contact,
string City) : base(Name, Age, BG, Contact, City)
        {
        }
        private string ID;
        public override string ID1 { get => ID; set => ID = value; }

        public override string GetName()
        {
            return Name;
        }
        public override void SetName(string Name)
        {
            this.Name = Name;
        }
        public override string GetAge()
        {
            return Age;
        }
        public override void SetAge(string Age)
        {
            this.Age = Age;
        }
        public override string GetBG()
        {
            return BG;
        }
        public override void SetBG(string BG)
        {
            this.BG = BG;
        }
        public override string GetContact()
        {
            return Contact;
        }
        public override void SetContact(string Contact)
        {
            this.Contact = Contact;
        }
        public override string GetCity()
        {
            return City;
        }
        public override void SetCity(string City)
        {
            this.City = City;
        }
        public override string GetID()
        {
            return ID1;
        }
        public override void SetID(string ID)
        {
            this.ID1 = ID;
        }
    }
```

- **Request:**

```
public class Request
```

```
        {
            private string value;
            private int count;
            public Request()
            {
            }
            public string Value { get => value; set => this.value = value; }
            public int Count { get => count; set => count = value; }
        }
```

7.2 Data Access Layer (DL) Classes:

- **EmployeeCRUD:**

```
    public class EmployeeCRUD
        {
            private static List<Employee> Employees = new List<Employee>();
            static string employeeFile = "Employee.txt";

            public static List<Employee> GetList()
            {
                return Employees;
            }
            public static Employee SearchEmployee(string name)
            {
                for (int idx = 0; idx < Employees.Count; idx++)
                {
                    if (name == Employees[idx].GetUsername())
                    {
                        return Employees[idx];
                    }
                }
                return null;
            }
            public static void AddEmployeeToList(Employee E1)
            {
                Employees.Add(E1);
            }
            public static void DeleteEmployeeToList(Employee E1)
            {
                Employees.Remove(E1);
            }

            public static void EmployeeToFile(Employee E)
            {
                StreamWriter employeeData = new StreamWriter(employeeFile, true);
                string IDs = "";
                employeeData.WriteLine(E.GetName() + "," + E.GetAge() + "," +
    E.GetCnic() + "," + E.GetContact() + "," + E.GetUsername() + "," +
    E.GetPassword() + "," + IDs);
                employeeData.Flush();
                employeeData.Close();
            }
            public static void updateEmployeeFile()
            {
                StreamWriter employeeData = new StreamWriter(employeeFile);
                for (int i = 0; i < Employees.Count; i++)
                {
                    string IDs = "";
                    if (Employees[i].GetUsername() != "")
                    {
                        if (Employees[i].GetPeople().Count > 0)
                        {
                            for (int x = 0; x < Employees[i].GetPeople().Count -
    1; x++)
                            {
                                IDs = IDs + Employees[i].GetPeople()[x].GetID() +
    ";";
                            }
                            IDs = IDs +
    Employees[i].GetPeople()[Employees[i].GetPeople().Count - 1].GetID();
                        }
                        employeeData.WriteLine(Employees[i].GetName() + "," +
    Employees[i].GetAge() + "," + Employees[i].GetCnic() + "," +
    Employees[i].GetContact() + "," + Employees[i].GetUsername() + "," +
    Employees[i].GetPassword() + "," + IDs);
                    }
                }
                employeeData.Flush();
                employeeData.Close();
            }
            public static bool LoadEmployee()
            {
                StreamReader employeeData = new StreamReader(employeeFile);
                string record;
                if (File.Exists(employeeFile))
                {
                    Employees.Clear();
                    while ((record = employeeData.ReadLine()) != null)
                    {
                        string[] splittedRecord = record.Split(',');
                        string name = splittedRecord[0];
                        string age = splittedRecord[1];
                        string contact = splittedRecord[2];
                        string cnic = splittedRecord[3];
                        string username = splittedRecord[4];
                        string password = splittedRecord[5];
                        string[] IDs = splittedRecord[6].Split(';');
                        Employee E = new Employee(name, age, cnic, contact,
    username, password);
                        // if(E.GetPeople().Count > 0)
                        for (int x = 0; x < IDs.Length; x++)
                        {
                            Person P = PersonCRUD.SearchPerson(IDs[x]);
                            if (P != null)
                            {
                                E.AddPerson(P);
                            }
                        }
                        AddEmployeeToList(E);
                    }
                    employeeData.Close();
                    return true;
                }
                else
                {
                    return false;
                }
            }
        }
```

- **PersonCRUD:**

```
public class PersonCRUD
    {
        private static List<Person> People = new List<Person>();
        private static int count = 1;
        static string personFile = "Person.txt";
        public static Person SearchPerson(string ID)
        {
            for (int idx = 0; idx < People.Count; idx++)
            {
                if (ID == People[idx].GetID())
                {
                    return People[idx];
                }
            }
            return null;
        }
        public static string SearchBGPerson(string BG)
        {
            for (int idx = 0; idx < People.Count; idx++)
            {
                if (BG == People[idx].GetBG())
                {
                    return People[idx].GetID();
                }
            }
            return null;
        }
        public static List<Person> SearchPersonByBG(string BG, char P)
        {
            List<Person> lst = new List<Person>();
            for (int i = 0; i < People.Count; i++)
            {
                if (People[i].GetBG() == BG && People[i].GetID()[0] == P)
                {
                    lst.Add(People[i]);
                }
            }
            return lst;
        }
        public static bool SearchBG(string BG)
        {
            for (int i = 0; i < People.Count; i++)
            {
                if (People[i].GetBG() == BG && People[i].GetID()[0] == 'D')
                {
                    return true;
                }
            }
            return false;
        }
        public static void PersonToFile(Person P)
        {
            StreamWriter personData = new StreamWriter(personFile, true);
            personData.WriteLine(P.GetID() + "," + P.GetName() + "," +
P.GetAge() + "," + P.GetBG() + "," + P.GetContact() + "," + P.GetCity());
            personData.Flush();
            personData.Close();
        }
        public static void updatePersonFile()
        {
            StreamWriter personData = new StreamWriter(personFile);
            for (int i = 0; i < People.Count; i++)
            {
                if (People[i].GetID() != "")
                {
                    personData.WriteLine(People[i].GetID() + "," +
People[i].GetName() + "," + People[i].GetAge() + "," + People[i].GetBG() +
"," + People[i].GetContact() + "," + People[i].GetCity());
                }
            }
            personData.Flush();
            personData.Close();
        }
        public static bool LoadPerson()
        {
            StreamReader personData = new StreamReader(personFile);
            string record;
            if (File.Exists(personFile))
            {
                People.Clear();
                while (!personData.EndOfStream)
                {
                    record = personData.ReadLine();
                    string[] splittedRecord = record.Split(',');
                    string id = splittedRecord[0];
                    string name = splittedRecord[1];
                    string age = splittedRecord[2];
                    string BG = splittedRecord[3];
                    string contact = splittedRecord[4];
                    string city = splittedRecord[5];
                    if (id[0] == 'D')
                    {
                        Donor P = new Donor(name, age, BG, contact, city);
                        P.SetID(id);
                        AddPersonToList(P);
                    }
                    else
                    {
                        Recipient P = new Recipient(name, age, BG, contact, 
city);
                        P.SetID(id);
                        AddPersonToList(P);
                    }
                }
                personData.Close();
                return true;
            }
            else
            {
                return false;
            }
        }
        public static void AddPersonToList(Person P)
        {
            People.Add(P);
        }
        public static void DeletePersonToList(Person P)
        {
            People.Remove(P);
        }
        public static void DeletePersonToListByID(string ID)
        {
            for (int i = 0; i < People.Count; i++)
            {
                if (People[i].GetID() == ID)
                {
                    DeletePersonToList(People[i]);
                }
            }
        }
        public static void DeletePersonToEmployeeList(string ID)
        {
            for (int i = 0; i < EmployeeCRUD.GetList().Count; i++)
            {
                for (int j = 0; j < 
EmployeeCRUD.GetList()[i].GetPeople().Count; j++)
                {
                    if (ID == 
EmployeeCRUD.GetList()[i].GetPeople()[j].GetID())
                    {

EmployeeCRUD.GetList()[i].DeletePerson(EmployeeCRUD.GetList()[i].GetPeople()[
j]);
                    }
                }
            }
        }
        public static List<Person> GetList()
        {
            return People;
        }
        public static string CreateID(Person P)
        {
            string ID;
            do
            {
                int number = count;
                if (P is Donor D)
                {
                    ID = "D" + number.ToString("D4");
                }
                else
                {
                    ID = "R" + number.ToString("D4");
                }
                count++;
            }
            while (SearchPerson(ID) != null);
            return ID;
        }
    }
```

- **RequestCRUD:**

```
public class RequestCRUD
    {
        private static List<string> RequestBG = new List<string>();

        private static string path = "Request.txt";
        public static List<string> GetRequest()
        {
            return RequestBG;
        }
        public static void AddRequest(string BG)
        {
            RequestBG.Add(BG);
        }
        public static void DeleteRequest(string BG)
        {
            RequestBG.Remove(BG);
        }
        public static void CheckReq()
        {
            for (int i = 0; i < RequestBG.Count; i++)
            {
                if (PersonCRUD.SearchBG(RequestBG[i]))
                {
                    string ID = PersonCRUD.SearchBGPerson(RequestBG[i]);
                    DeleteRequest(RequestBG[i]);
                    PersonCRUD.DeletePersonToListByID(ID);
                    PersonCRUD.DeletePersonToEmployeeList(ID);
                }
            }
```

```
                UpdateRequestToFile();
            }
            public static bool LoadRequests()
            {
                StreamReader requestData = new StreamReader(path);
                string record;
                if (File.Exists(path))
                {
                    RequestBG.Clear();
                    while ((record = requestData.ReadLine()) != null)
                    {
                        RequestBG.Add(record);
                    }
                    requestData.Close();
                    return true;
                }
                return false;
            }
            public static void RequestToFile(string BG)
            {
                StreamWriter reqData = new StreamWriter(path, true);
                reqData.WriteLine(BG);
                reqData.Flush();
                reqData.Close();
            }
            public static void UpdateRequestToFile()
            {
                StreamWriter reqData = new StreamWriter(path);
                for (int i = 0; i < RequestBG.Count; i++)
                {
                    reqData.WriteLine(RequestBG[i]);
                }
                reqData.Flush();
                reqData.Close();
            }
            public static List<Request> CountRequest()
            {
                List<Request> requests = RequestBG.GroupBy(x => x)
                          .Select(g => new Request { Value = g.Key, Count = 
    g.Count() })
                          .OrderByDescending(x => x.Count)
                          .ToList();
                return requests;
            }
        }
```

7.3 User Interface (UI) Forms:

- **Login:**

```
    public partial class Login : Form
        {
            public Employee CurrentEmployee;
```

```
public Login()
{
    InitializeComponent();
}
private void CheckLoginUser(User U)
{
    User loginUser = U.CheckRole();
    if (loginUser is Admin A)
    {
        AdminUI admin = new AdminUI();
        this.Hide();
        admin.Show();

    }
    else if (loginUser is Employee E)
    {
        CurrentEmployee = E;
        EmployeeUI employee = new EmployeeUI(CurrentEmployee);
        this.Hide();
        employee.Show();
    }
    else if (loginUser is Guest G)
    {
        GuestUI guest = new GuestUI();
        this.Hide();
        guest.Show();
    }
    else
    {
        MessageBox.Show("Wrong Input");
        txt_user.Clear();
        txt_pass.Clear();
    }
}

private void btn_login_Click(object sender, EventArgs e)
{
    string user = txt_user.Text;
    string pass = txt_pass.Text;
    User U = new User(user, pass);
    CheckLoginUser(U);
}

private void Login_FormClosing(object sender, FormClosingEventArgs e)
{
    Environment.Exit(0);
}

private void Login_Load(object sender, EventArgs e)
{
    bool flag1 = PersonCRUD.LoadPerson();
    bool flag2 = EmployeeCRUD.LoadEmployee();
    bool flag3 = RequestCRUD.LoadRequests();
    if (!(flag1 && flag2 && flag3))
    {
```

```
                    MessageBox.Show("Data Not Loaded");
                }
            }
        }
```

- **AdminUI:**

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

        private void customizeDesign()
        {
            panel_Emenu.Visible = false;
            panel_Dmenu.Visible = false;
            panel_Rmenu.Visible = false;
        }
        private void hideSubmenu()
        {
            if (panel_Emenu.Visible == true)
            {
                panel_Emenu.Visible = false;
            }
            if (panel_Dmenu.Visible == true)
            {
                panel_Dmenu.Visible = false;
            }
            if (panel_Rmenu.Visible == true)
            {
                panel_Rmenu.Visible = false;
            }
        }
        private void showSubmenu(Panel M)
        {
            if (M.Visible == false)
            {
                hideSubmenu();
                M.Visible = true;
            }
            else
            {
                M.Visible = false;
            }
        }



        private void btn_employee_Click(object sender, EventArgs e)
        {
            showSubmenu(panel_Emenu);
        }
```

```
private void btn_AddE_Click(object sender, EventArgs e)
{
    hideSubmenu();
    AddEmployee addEmployee = new AddEmployee();
    setForm(addEmployee);
}

private void btn_delE_Click(object sender, EventArgs e)
{
    hideSubmenu();
    DeleteEmployee deleteEmployee = new DeleteEmployee();
    setForm(deleteEmployee);
}

private void btn_uptE_Click(object sender, EventArgs e)
{
    hideSubmenu();
    UpdateEmployee updateEmployee = new UpdateEmployee();
    setForm(updateEmployee);
}

private void btn_searchE_Click(object sender, EventArgs e)
{
    hideSubmenu();
    SearchEmployee searchEmployee = new SearchEmployee();
    setForm(searchEmployee);
}

private void btn_viewE_Click(object sender, EventArgs e)
{
    hideSubmenu();
    ViewEmployee viewEmployee = new ViewEmployee();
    setForm(viewEmployee);
}

private void btn_donor_Click(object sender, EventArgs e)
{
    showSubmenu(panel_Dmenu);
}
private void btn_searchD_Click(object sender, EventArgs e)
{
    hideSubmenu();
    SearchPerson searchPerson = new SearchPerson(true, true);
    setForm(searchPerson);
}

private void btn_viewD_Click(object sender, EventArgs e)
{
    hideSubmenu();
    ViewPerson viewPerson = new ViewPerson(true, true);
    setForm(viewPerson);
}

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

```
            showSubmenu(panel_Rmenu);
        }
        private void btn_searchR_Click(object sender, EventArgs e)
        {
            hideSubmenu();
            SearchPerson searchPerson = new SearchPerson(false, true);
            setForm(searchPerson);
        }

        private void btn_viewR_Click(object sender, EventArgs e)
        {
            hideSubmenu();
            ViewPerson viewPerson = new ViewPerson(false, true);
            setForm(viewPerson);
        }
        private void btn_logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Login login = new Login();
            login.Show();
        }
        private void AdminUI_FormClosing(object sender, FormClosingEventArgs
e)
        {
            Environment.Exit(0);
        }
        private void setForm(Form form)
        {
            panel_main.Controls.Clear();
            form.TopLevel = false;
            form.FormBorderStyle = FormBorderStyle.None;
            panel_main.Controls.Add(form);
            form.Dock = DockStyle.Fill;
            panel_main.Tag = form;
            form.BringToFront();
            form.Show();
        }

        private void btn_request_Click(object sender, EventArgs e)
        {
            hideSubmenu();
            ViewRequest viewRequest = new ViewRequest();
            setForm(viewRequest);
        }
    }
```

- **EmployeeUI:**

```
public partial class EmployeeUI : Form
    {
        private Employee E;
        public EmployeeUI(Employee E)
        {
            this.E = E;
            InitializeComponent();
```

```
                customizeDesign();
                setCredentials();
            }
            private void setCredentials()
            {
                user.Text = E.GetUsername();
                pass.Text = E.GetPassword();
            }
            private void btn_change_Click(object sender, EventArgs e)
            {
                string input = "";
                if (changePass("Dialog Box", "Enter New Password", ref input) ==
        DialogResult.OK)
                {
                    E.SetPassword(input);
                    EmployeeCRUD.updateEmployeeFile();
                    setCredentials();
                }
            }
            private DialogResult changePass(string title, string promptText, ref
        string value)
            {
                Form form = new Form();
                Label label = new Label();
                TextBox textBox = new TextBox();
                Button buttonOk = new Button();
                Button buttonCancel = new Button();
                form.Text = title;
                label.Text = promptText;
                buttonOk.Text = "OK";
                buttonCancel.Text = "Cancel";
                buttonOk.DialogResult = DialogResult.OK;
                buttonCancel.DialogResult = DialogResult.Cancel;
                label.SetBounds(36, 36, 200, 13);
                textBox.SetBounds(36, 86, 200, 20);
                buttonOk.SetBounds(50, 160, 100, 60);
                buttonCancel.SetBounds(150, 160, 100, 60);
                label.AutoSize = true;
                form.ClientSize = new Size(300, 300);
                form.FormBorderStyle = FormBorderStyle.FixedDialog;
                form.StartPosition = FormStartPosition.CenterScreen;
                form.MinimizeBox = false;
                form.MaximizeBox = false;
                form.Controls.AddRange(new Control[] { label, textBox, buttonOk,
        buttonCancel });
                form.AcceptButton = buttonOk;
                form.CancelButton = buttonCancel;
                DialogResult dialogResult = form.ShowDialog();
                value = textBox.Text;
                return dialogResult;
            }
            private void customizeDesign()
            {
                panel_Dmenu.Visible = false;
                panel_Rmenu.Visible = false;
            }
            private void hideSubmenu()
            {
                if (panel_Dmenu.Visible == true)
                {
                    panel_Dmenu.Visible = false;
                }
                if (panel_Rmenu.Visible == true)
                {
                    panel_Rmenu.Visible = false;
                }
            }
            private void showSubmenu(Panel M)
            {
                if (M.Visible == false)
                {
                    hideSubmenu();
                    M.Visible = true;
                }
                else
                {
                    M.Visible = false;
                }
            }

            private void btn_donor_Click(object sender, EventArgs e)
            {
                showSubmenu(panel_Dmenu);
            }
            private void btn_AddD_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                AddPerson addPerson = new AddPerson(E, true);
                setForm(addPerson);
            }

            private void btn_delD_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                DeletePerson deletePerson = new DeletePerson(E, true);
                setForm(deletePerson);
            }

            private void btn_uptD_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                UpdatePerson updatePerson = new UpdatePerson(E, true);
                setForm(updatePerson);
            }

            private void btn_searchD_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                SearchPerson searchPerson = new SearchPerson(E, true);
                setForm(searchPerson);
            }

            private void btn_viewD_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                ViewPerson viewPerson = new ViewPerson(E, true, false);
                setForm(viewPerson);
            }

            private void btn_recipient_Click(object sender, EventArgs e)
            {
                showSubmenu(panel_Rmenu);
            }

            private void btn_addR_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                AddPerson addPerson = new AddPerson(E, false);
                setForm(addPerson);
            }

            private void btn_delR_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                DeletePerson deletePerson = new DeletePerson(E, false);
                setForm(deletePerson);
            }

            private void btn_uptR_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                UpdatePerson updatePerson = new UpdatePerson(E, false);
                setForm(updatePerson);
            }

            private void btn_searchR_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                SearchPerson searchPerson = new SearchPerson(E, false);
                setForm(searchPerson);
            }

            private void btn_viewR_Click(object sender, EventArgs e)
            {
                hideSubmenu();
                ViewPerson viewPerson = new ViewPerson(E, false, false);
                setForm(viewPerson);
            }

            private void btn_logout_Click(object sender, EventArgs e)
            {
                this.Hide();
                Login login = new Login();
                login.Show();
            }
            private void EmployeeUI_FormClosing(object sender,
    FormClosingEventArgs e)
            {
                Environment.Exit(0);
            }
            private void setForm(Form form)
            {
                panel_main.Controls.Clear();
                form.TopLevel = false;
                form.FormBorderStyle = FormBorderStyle.None;
                form.Dock = DockStyle.Fill;
                panel_main.Controls.Add(form);
                panel_main.Tag = form;
                form.BringToFront();
                form.Show();
            }

        }
```

- **GuestUI:**

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

        private void btn_logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Login login = new Login();
            login.Show();
        }

        private void GuestUI_FormClosing(object sender, FormClosingEventArgs
e)
        {
            Environment.Exit(0);
        }

        private void btn_donor_Click(object sender, EventArgs e)
        {
            SearchPerson viewPerson = new SearchPerson(true, true);
            setForm(viewPerson);
        }

        private void btn_recipient_Click(object sender, EventArgs e)
        {
            SearchPerson searchPerson = new SearchPerson(false, true);
            setForm(searchPerson);
        }
        private void setForm(Form form)
        {
            panel_main.Controls.Clear();
            form.TopLevel = false;
            form.FormBorderStyle = FormBorderStyle.None;
            panel_main.Controls.Add(form);
            form.Dock = DockStyle.Fill;
            panel_main.Tag = form;
            form.BringToFront();
            form.Show();
        }

        private void btn_request_Click(object sender, EventArgs e)
        {
            AddRequest addRequest = new AddRequest();
            setForm(addRequest);
        }
    }
```

- **AddEmployee:**

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

        private void btn_Add_Click(object sender, EventArgs e)
        {
            string name = txt_name.Text;
            string age = num_age.Value.ToString();
            string contact = txt_contact.Text;
            string cnic = txt_cnic.Text;
            string username = txt_user.Text;
            string password = txt_pass.Text;
            if (Validation())
            {
                if (EmployeeCRUD.SearchEmployee(username) == null)
                {
                    Employee E = new Employee(name, age, contact, cnic,
username, password);
                    EmployeeCRUD.AddEmployeeToList(E);
                    EmployeeCRUD.EmployeeToFile(E);
                    this.Hide();
                    MessageBox.Show("Added Successfully");
                }
                else
                {
                    MessageBox.Show("Username Already Exists");
                }
            }
            else
            {
                MessageBox.Show("Invalid or Incomplete Input");
            }
        }

        private void txt_contact_Click(object sender, EventArgs e)
        {
            if (txt_contact.Text == "    -")
                txt_contact.SelectionStart = 0;
            else
                txt_contact.SelectionStart = txt_contact.Text.Length;

        }

        private void txt_cnic_Click(object sender, EventArgs e)
        {
            if (txt_contact.Text == "     -       -")
                txt_contact.SelectionStart = 0;
            else
                txt_contact.SelectionStart = txt_contact.Text.Length;

        }

        private void txt_name_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
        }
        private bool Validation()
        {
            bool flag = true;
            if (txt_name.Text == null || txt_contact.Text.Length != 12 ||
txt_cnic.Text.Length != 15 || txt_user.Text.Length != 5 ||
txt_pass.Text.Length != 5)
            {
                flag = false;
            }

            return flag;
        }
    }
```

- **DeleteEmployee:**

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

        private void btn_find_Click(object sender, EventArgs e)
        {
            if (txt_find.Text != null)
            {
                Employee E = EmployeeCRUD.SearchEmployee(txt_find.Text);
                if (E != null)
                {
                    txt_name.Text = E.GetName();
                    num_age.Text = E.GetAge();
                    txt_contact.Text = E.GetContact();
                    txt_cnic.Text = E.GetCnic();
                    txt_user.Text = E.GetUsername();
                    txt_pass.Text = E.GetPassword();
                }
                else
                {
                    MessageBox.Show("Not Found");
                }
            }
        }

        private void btn_Del_Click(object sender, EventArgs e)
        {
            Employee E = EmployeeCRUD.SearchEmployee(txt_find.Text);
            if (E != null)
            {
                EmployeeCRUD.DeleteEmployeeToList(E);
                EmployeeCRUD.updateEmployeeFile();
                this.Hide();
                MessageBox.Show("Deleted Successfully");
            }
        }

        private void txt_find_Click(object sender, EventArgs e)
        {
            txt_find.Clear();
        }
    }
```

- **UpdateEmployee:**

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

        private void btn_find_Click(object sender, EventArgs e)
        {
            if (txt_find.Text != null)
            {
                Employee E = EmployeeCRUD.SearchEmployee(txt_find.Text);
                if (E != null)
                {
                    txt_name.Text = E.GetName();
                    num_age.Text = E.GetAge();
                    txt_contact.Text = E.GetContact();
                    txt_cnic.Text = E.GetCnic();
                    txt_user.Text = E.GetUsername();
                    txt_pass.Text = E.GetPassword();
                }
                else
                {
                    MessageBox.Show("Not Found");
                }
            }
        }

        private void txt_find_Click(object sender, EventArgs e)
        {
            txt_find.Clear();
        }

        private void btn_upt_Click(object sender, EventArgs e)
        {
            Employee E = EmployeeCRUD.SearchEmployee(txt_find.Text);
            if (E != null && Validation())
            {
                if (EmployeeCRUD.SearchEmployee(txt_user.Text) == null)
                {
                    E.SetName(txt_name.Text);
                    E.SetAge(num_age.Text);
                    E.SetContact(txt_contact.Text);
                    E.SetCnic(txt_cnic.Text);
                    E.SetUsername(txt_user.Text);
                    E.SetPassword(txt_pass.Text);

                    EmployeeCRUD.updateEmployeeFile();
                    this.Hide();
                    MessageBox.Show("Updated Successfully");
                }
                else
                {
                    MessageBox.Show("Username Already Exists");
                }
            }
            else
            {
                MessageBox.Show("Invalid or Incomplete Input");
            }
        }
        private bool Validation()
        {
            bool flag = true;
            if (txt_name.Text == null || txt_contact.Text.Length != 12 ||
txt_cnic.Text.Length != 15 || txt_user.Text.Length != 5 ||
txt_pass.Text.Length != 5)
            {
                flag = false;
            }

            return flag;
        }

        private void txt_contact_Click(object sender, EventArgs e)
        {
            if (txt_contact.Text == "    -")
                txt_contact.SelectionStart = 0;
            else
                txt_contact.SelectionStart = txt_contact.Text.Length;

        }

        private void txt_cnic_Click(object sender, EventArgs e)
        {

            if (txt_contact.Text == "     -       -")
                txt_contact.SelectionStart = 0;
            else
                txt_contact.SelectionStart = txt_contact.Text.Length;


        }

        private void txt_name_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
        }
    }
```

- **SearchEmployee:**

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

            public AdminUI AdminUI
            {
                get => default;
                set
                {
                }
            }

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

            private void btn_search_Click(object sender, EventArgs e)
            {
                List<Employee> employee = new List<Employee>();
                Employee E = EmployeeCRUD.SearchEmployee(txt_search.Text);
                if (E != null)
                    employee.Add(E);
                databind(employee);
            }
            private void databind(List<Employee> employee)
            {
                if (employee == null)
                {
                    MessageBox.Show("Not found");
                }
                else
                {
                    GV.DataSource = null;
                    foreach (Employee person in employee)
                    {
                        GV.DataSource = employee.Select(a => new { a.Name1,
    a.Age1, a.Contact1, a.Cnic1, a.Username, a.Password }).ToList();
                    }
                    GV.Columns[0].HeaderText = "Name";
                    GV.Columns[1].HeaderText = "Age";
                    GV.Columns[2].HeaderText = "Contact";
                    GV.Columns[3].HeaderText = "CNIC";
                    GV.Refresh();
                }
            }

            private void txt_search_Click(object sender, EventArgs e)
            {
                txt_search.Clear();
            }
        }
```

- **ViewEmployee:**

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

        public AdminUI AdminUI
        {
            get => default;
            set
            {
            }
        }

        private void btn_back_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void Databind()
        {
            List<Employee> list = EmployeeCRUD.GetList();
            if (list.Count > 0)
            {
                GV.DataSource = null;
                GV.DataSource = list;

                GV.Columns[1].HeaderText = "Name";
                GV.Columns[2].HeaderText = "Age";
                GV.Columns[3].HeaderText = "Contact";
                GV.Columns[4].HeaderText = "CNIC";

                GV.Refresh();
            }

            else
            {
                MessageBox.Show("Not Found Add First to View");
            }
        }
        private void btn_pdf_Click(object sender, EventArgs e)
        {
            PdfGenerator.GeneratePdfReport(EmployeeCRUD.GetList());
            MessageBox.Show("Downloaded Successfully");
            this.Hide();
        }

        private void GV_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (GV.Columns["btn_view"].Index == e.ColumnIndex)
            {
                int row = e.RowIndex;
                Employee E =
EmployeeCRUD.SearchEmployee(GV.Rows[row].Cells[5].Value.ToString());
                if (E.GetPeople().Count > 0)
                {
                    PdfGenerator.GeneratePdfReport(E.GetPeople());
                    MessageBox.Show("Downloaded Successfully");
                }
                else
                {
                    MessageBox.Show("This Employee has no DATA");
                }
            }
        }
    }
```

- **ViewRequest:**

```
public partial class ViewRequest : Form
    {
        public ViewRequest()
        {
            InitializeComponent();
            databind();
        }
        public AdminUI AdminUI
        {
            get => default;
            set
            {
            }
        }

        private void databind()
        {
            List<Request> list = RequestCRUD.CountRequest();
            GV.DataSource = null;
            GV.DataSource = list;
            GV.Columns[0].HeaderText = "Bloodgroup";
            GV.Columns[1].HeaderText = "Requests";
            GV.Refresh();
        }

        private void btn_back_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        private void btn_fullfil_Click(object sender, EventArgs e)
        {
            RequestCRUD.CheckReq();
            PersonCRUD.updatePersonFile();
            EmployeeCRUD.updateEmployeeFile();
            databind();
        }

    }
```

- **AddPerson:**

```
public partial class AddPerson : Form
    {
        private Employee E;
        private bool isDonor;
        public AddPerson(Employee E, bool isDonor)
        {
            this.E = E;
            this.isDonor = isDonor;
            InitializeComponent();
        }

        private void btn_Add_Click(object sender, EventArgs e)
        {
            string name = txt_name.Text;
            string age = num_age.Value.ToString();
            string bg = cb_BG.Text;
            string contact = txt_contact.Text;
            string city = cb_city.Text;

            if (Validation())
```

```
                {
                    if (isDonor)
                    {
                        Donor D = new Donor(name, age, bg, contact, city);
                        D.SetID(PersonCRUD.CreateID(D));
                        E.AddPerson(D);
                        PersonCRUD.AddPersonToList(D);
                        PersonCRUD.PersonToFile(D);
                    }
                    else
                    {
                        Recipient R = new Recipient(name, age, bg, contact,
city);
                        R.SetID(PersonCRUD.CreateID(R));
                        E.AddPerson(R);
                        PersonCRUD.AddPersonToList(R);
                        PersonCRUD.PersonToFile(R);
                    }


                    EmployeeCRUD.updateEmployeeFile();
                    this.Hide();
                    MessageBox.Show("Added Successfully");
                }
                else
                {
                    MessageBox.Show("Invalid or Incomplete Input");
                }
            }

            private void cb_BG_Click(object sender, EventArgs e)
            {
                cb_BG.DroppedDown = true;
            }

            private void cb_city_Click(object sender, EventArgs e)
            {
                cb_city.DroppedDown = true;
            }

            private void txt_contact_Click(object sender, EventArgs e)
            {
                if (txt_contact.Text == "    -")
                    txt_contact.SelectionStart = 0;
                else
                    txt_contact.SelectionStart = txt_contact.Text.Length;

            }

            private void txt_name_KeyPress(object sender, KeyPressEventArgs e)
            {
                e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
            }
            private bool Validation()
```

```
        {
            bool flag = true;
            if (txt_name.Text == null || txt_contact.Text.Length != 12 ||
cb_BG.SelectedIndex == -1 || cb_city.SelectedIndex == -1)
            {
                flag = false;
            }

            return flag;
        }
    }
```

- **DeletePerson:**

```
public partial class DeletePerson : Form
    {
        private Employee E;
        private bool isDonor;
        public DeletePerson(Employee E, bool isDonor)
        {
            this.E = E;
            this.isDonor = isDonor;
            InitializeComponent();
        }

        private void btn_del_Click(object sender, EventArgs e)
        {
            Person P = E.SearchPerson(txt_search.Text);
            E.DeletePerson(P);
            PersonCRUD.DeletePersonToList(P);
            PersonCRUD.updatePersonFile();
            EmployeeCRUD.updateEmployeeFile();

            this.Hide();
            MessageBox.Show("Deleted Successfully");
        }

        private void txt_search_Click(object sender, EventArgs e)
        {
            txt_search.Clear();
        }

        private void btn_search_Click(object sender, EventArgs e)
        {
            if (txt_search.Text != null)
            {
                Person P = E.SearchPerson(txt_search.Text);
                if (P is Donor && isDonor)
                {
                    txt_name.Text = P.GetName();
                    txt_age.Text = P.GetAge();
                    txt_BG.Text = P.GetBG();
                    txt_contact.Text = P.GetContact();
                    txt_city.Text = P.GetCity();
                }
```

```
                    else if (P is Recipient && !isDonor)
                    {
                        txt_name.Text = P.GetName();
                        txt_age.Text = P.GetAge();
                        txt_BG.Text = P.GetBG();
                        txt_contact.Text = P.GetContact();
                        txt_city.Text = P.GetCity();
                    }
                    else
                    {
                        MessageBox.Show("Not Found");
                    }
                }
            }
        }
```

- **UpdatePerson:**

```
public partial class UpdatePerson : Form
    {
        private Employee E;
        private bool isDonor;
        public UpdatePerson(Employee E, bool isDonor)
        {
            this.E = E;
            this.isDonor = isDonor;
            InitializeComponent();
        }

        private void txt_search_Click(object sender, EventArgs e)
        {
            txt_search.Clear();
        }

        private void btn_search_Click(object sender, EventArgs e)
        {
            if (txt_search.Text != null)
            {
                Person P = E.SearchPerson(txt_search.Text);
                if (P is Donor && isDonor)
                {
                    txt_name.Text = P.GetName();
                    num_age.Text = P.GetAge();
                    cb_BG.Text = P.GetBG();
                    txt_contact.Text = P.GetContact();
                    cb_city.Text = P.GetCity();
                }
                else if (P is Recipient && !isDonor)
                {
                    txt_name.Text = P.GetName();
                    num_age.Text = P.GetAge();
                    cb_BG.Text = P.GetBG();
                    txt_contact.Text = P.GetContact();
                    cb_city.Text = P.GetCity();
                }
                else
                {
                    MessageBox.Show("Not Found");
                }
            }
        }
        private void btn_upt_Click(object sender, EventArgs e)
        {
            Person P = E.SearchPerson(txt_search.Text);
            if (P != null && Validation())
            {
                P.SetName(txt_name.Text);
                P.SetAge(num_age.Value.ToString());
                P.SetBG(cb_BG.Text);
                P.SetContact(txt_contact.Text);
                P.SetCity(cb_city.Text);

                PersonCRUD.updatePersonFile();
                EmployeeCRUD.updateEmployeeFile();

                this.Hide();
                MessageBox.Show("Updated Successfully");
            }
            else
            {
                MessageBox.Show("Invalid or Incomplete Input");
            }
        }
        private bool Validation()
        {
            bool flag = true;
            if (txt_name.Text == null || txt_contact.Text.Length != 12 ||
cb_BG.SelectedIndex == -1 || cb_city.SelectedIndex == -1)
            {
                flag = false;
            }

            return flag;
        }

        private void txt_name_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = !(char.IsLetter(e.KeyChar) || e.KeyChar ==
(char)Keys.Back || e.KeyChar == (char)Keys.Space);
        }

        private void txt_contact_Click(object sender, EventArgs e)
        {
            if (txt_contact.Text == "    -")
                txt_contact.SelectionStart = 0;
            else
                txt_contact.SelectionStart = txt_contact.Text.Length;
        }
    }
```

- **SearchPerson:**

```
public partial class SearchPerson : Form
    {
        private Employee E;
        private bool isDonor;
        private bool isAdmin;
        public SearchPerson(Employee E, bool isDonor)
        {
            this.E = E;
            this.isDonor = isDonor;
            InitializeComponent();
        }
        public SearchPerson(bool isDonor, bool isAdmin)
        {
            InitializeComponent();
            this.isDonor = isDonor;
            this.isAdmin = isAdmin;
        }

        public AdminUI AdminUI
        {
            get => default;
            set
            {
            }
        }

        public GuestUI GuestUI
        {
            get => default;
            set
            {
            }
        }

        private void btn_search_Click(object sender, EventArgs e)
        {
            if (!isAdmin)
            {
                userBinding();
            }
            else
            {
                adminBinding();
            }
        }
        private void userBinding()
        {
            List<Person> people;
            if (isDonor)
            {
                people = E.SearchPersonByBG(cb_BG.Text, 'D');
            }
            else
```

```
                {
                    people = E.SearchPersonByBG(cb_BG.Text, 'R');
                }
                databind(people);
            }
            private void adminBinding()
            {
                List<Person> people;
                if (isDonor)
                {
                    people = PersonCRUD.SearchPersonByBG(cb_BG.Text, 'D');
                }
                else
                {
                    people = PersonCRUD.SearchPersonByBG(cb_BG.Text, 'R');
                }
                databind(people);
            }
            private void databind(List<Person> people)
            {
                if (people == null)
                {
                    MessageBox.Show("Not found");
                }
                else
                {
                    GV.DataSource = null;
                    /*foreach (Person person in people)
                    {
                        GV.DataSource = people.Select(a => new { a.ID1, a.Name,
    a.Age, a.BG, a.Contact, a.City }).ToList();
                    }*/
                    GV.DataSource = people;
                    GV.Columns[0].HeaderText = "ID";
                    GV.Refresh();
                }
            }
            private void cb_BG_Click(object sender, EventArgs e)
            {
                cb_BG.DroppedDown = true;
            }

            private void btn_back_Click(object sender, EventArgs e)
            {
                this.Hide();
            }
        }
```

- **ViewPerson:**

```
    public partial class ViewPerson : Form
        {
            private Employee E;
            private bool isDonor;
            private bool isAdmin;
```

```
public ViewPerson(Employee E, bool isDonor, bool isAdmin)
{
    this.E = E;
    this.isDonor = isDonor;
    this.isAdmin = isAdmin;
    InitializeComponent();

    databind();

}
public ViewPerson(bool isDonor, bool isAdmin)
{
    this.isDonor = isDonor;
    this.isAdmin = isAdmin;
    InitializeComponent();

    AdminDatabind();

}

public EmployeeUI EmployeeUI
{
    get => default;
    set
    {
    }
}

public AdminUI AdminUI
{
    get => default;
    set
    {
    }
}

private void AdminDatabind()
{
    List<Person> list = new List<Person>();
    List<Person> people = PersonCRUD.GetList();
    GV.DataSource = null;
    if (people.Count > 0)
    {
        for (int i = 0; i < people.Count; i++)
        {
            if (isDonor)
            {
                if (people[i] is Donor D)
                {
                    list.Add(people[i]);
                }
            }
            else
            {
                if (people[i] is Recipient R)
```

```
                        {
                            list.Add(people[i]);
                        }
                    }

                }
                GV.DataSource = list;
                GV.Columns[0].HeaderText = "ID";
                GV.Refresh();
            }
            else
            {
                MessageBox.Show("Not Found Add First to View");
            }
        }

        private void databind()
        {
            List<Person> list = new List<Person>();
            GV.Rows.Clear();
            if (E.GetPeople().Count > 0)
            {
                foreach (Person person in E.GetPeople())
                {
                    if (isDonor)
                    {
                        if (person is Donor D)
                        {
                            list.Add(person);
                        }
                    }
                    else
                    {
                        if (person is Recipient R)
                        {
                            list.Add(person);
                        }
                    }
                }

                GV.DataSource = list;
                GV.Columns[0].HeaderText = "ID";
                GV.Refresh();
            }
            else
            {
                MessageBox.Show("Not Found Add First to View");
            }
        }

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

```
private void btn_pdf_Click(object sender, EventArgs e)
{
    if (isAdmin)
    {
        List<Person> list = new List<Person>();
        List<Person> people = PersonCRUD.GetList();
        GV.DataSource = null;
        if (people.Count > 0)
        {
            for (int i = 0; i < people.Count; i++)
            {
                if (isDonor)
                {
                    if (people[i] is Donor D)
                    {
                        list.Add(people[i]);
                    }
                }
                else
                {
                    if (people[i] is Recipient R)
                    {
                        list.Add(people[i]);
                    }
                }

            }
            PdfGenerator.GeneratePdfReport(list);
        }
    }
    else
    {
        List<Person> list = new List<Person>();
        if (E.GetPeople().Count > 0)
        {
            foreach (Person person in E.GetPeople())
            {
                if (isDonor)
                {
                    if (person is Donor D)
                    {
                        list.Add(person);
                    }
                }
                else
                {
                    if (person is Recipient R)
                    {
                        list.Add(person);
                    }
                }
            }
            PdfGenerator.GeneratePdfReport(list);
        }
    }
```

```
                MessageBox.Show("Downloaded Successfully");
                this.Hide();
            }
        }
```

- **AddRequest:**

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

        private void cb_BG_Click(object sender, EventArgs e)
        {
            cb_BG.DroppedDown = true;
        }

        private void btn_add_Click(object sender, EventArgs e)
        {
            if (cb_BG.Text != null)
            {
                if (PersonCRUD.SearchBG(cb_BG.Text))
                {
                    MessageBox.Show("Your Required BloodGroup is Present in
our Bank You can get this");
                }
                else
                {
                    RequestCRUD.AddRequest(cb_BG.Text);
                    RequestCRUD.RequestToFile(cb_BG.Text);
                    this.Hide();
                    MessageBox.Show("Request Added");
                }
            }
            else
            {
                MessageBox.Show("Please Select the Bloodgroup");
            }
        }
    }
```

- **PdfGenerator:**

```
public class PdfGenerator
    {

        public static void GeneratePdfReport<T>(List<T> data)
        {
            Document document = new Document();


            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "PDF files (*.pdf)|*.pdf";
            saveFileDialog.DefaultExt = "pdf";


            DialogResult result = saveFileDialog.ShowDialog();
            string filePath = string.Empty;
            if (result == DialogResult.OK)
            {
                filePath = saveFileDialog.FileName;

            }


            PdfWriter writer = PdfWriter.GetInstance(document, new 
FileStream(filePath, FileMode.Create));


            document.Open();

            Phrase phrase = new Phrase();
            phrase.Add(new Chunk("Blood Donation Management System\n\n", 
FontFactory.GetFont("Arial", 16, Font.BOLD)));


            DataTable dataTable = new DataTable();
            var properties = typeof(T).GetProperties();
            foreach (var property in properties)
            {
                dataTable.Columns.Add(property.Name);
            }
            foreach (var item in data)
            {
                DataRow row = dataTable.NewRow();

                foreach (var property in properties)
                {
                    var value = property.GetValue(item);
                    row[property.Name] = value != null ? value.ToString() : 
string.Empty;
                }

                dataTable.Rows.Add(row);
            }
            PdfPTable pdfTable = new PdfPTable(dataTable.Columns.Count);

            pdfTable.WidthPercentage = 100;
            foreach (DataColumn column in dataTable.Columns)
            {
                PdfPCell cell = new PdfPCell(new Phrase(column.ColumnName));
                cell.BackgroundColor = BaseColor.LIGHT_GRAY;
                cell.HorizontalAlignment = Element.ALIGN_CENTER
                pdfTable.AddCell(cell);
            }
            foreach (DataRow row in dataTable.Rows)
            {
                foreach (var item in row.ItemArray)
```

```
                    {
                        PdfPCell cell = new PdfPCell(new
    Phrase(item.ToString()));
                        cell.HorizontalAlignment = Element.ALIGN_CENTER
                        pdfTable.AddCell(cell);
                    }
                }

                document.Add(pdfTable);
                document.Close();
                writer.Close();
            }
        }
```

8. CONCLUSION:

In conclusion I will summarize my project idea and its achievements, challenges faced during developments and the new things I learnt from this project.

8.1 Summarization and Achievements:

The blood donation management system is a technology-driven solution designed to address the challenges associated with finding specific blood groups during emergency situations or surgeries. By leveraging the principles of object-oriented programming, the system improves the efficiency and accessibility of the blood donation process, contributing to better healthcare outcomes.

Throughout the development of the project, several achievements have been made. The system provides a user-friendly interface that allows individuals to easily search for their desired blood group in a brief time. Real-time availability of blood group information ensures timely access to suitable blood donors, potentially saving lives. The system also incorporates features such as donor and donation management, blood drive organization, and data security measures, enhancing the overall functionality and usability of the system.

8.2 Challenges:

However, the project also encountered some challenges along the way. One significant challenge was communication between different classes of the system. Managing a large data of donors and recipients along with their blood groups and respective employees. while maintaining data maintainability and validation also posted its own set of challenges. Additionally, the most challenging moment for me is ORM during file-handling.

8.3 Lesson Learned:

Throughout the project, several valuable lessons were learned. It became evident that effective planning and requirement gathering are crucial for the successful development of such a complex system. Modularity and separation of concerns played a vital role in achieving maintainability and flexibility. The implementation of design patterns, such as the BL, DL, and UI patterns, helped in achieving a structured and organized codebase. Furthermore, continuous testing and quality assurance were imperative to identify and resolve any issues early on.