

**Project Documentation**

**Blood Bank Management System**

**Team Members:**

1. Tasneem Adel.
2. Basma Mohamed.
3. Yomna Alaa.
4. Yasmin Shaaban.
5. Nehal A. Ali.
6. Yousra Desoukey

**Project Brief Description :**

Blood Bank Management System is a system that is designed to store, process, retrieve and analyze information concerned with the administrative and inventory management within a blood bank. We were working on blood donation and blood bank database management system.

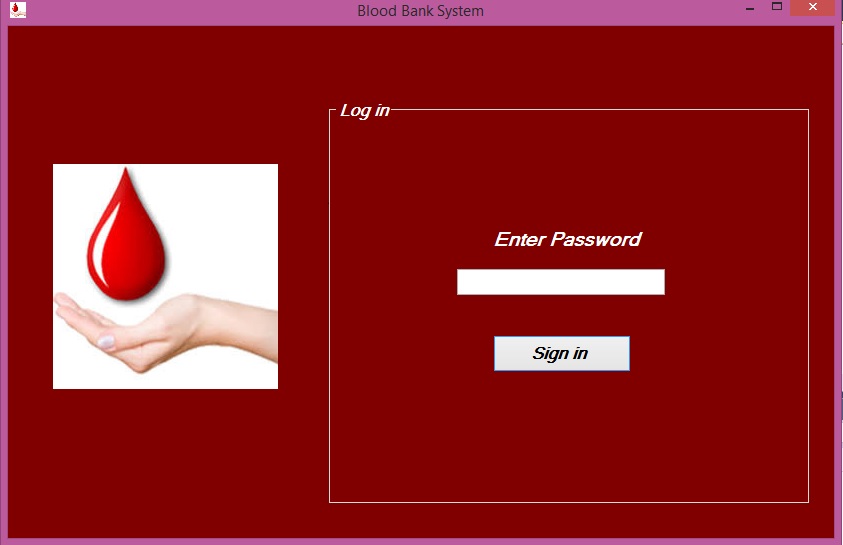
Our project establishes an easy communication between blood donors and blood acceptors according to their blood group. The user makes registration for Donor or Acceptor in the system and can search for matching blood groups for any blood Acceptor and gets the details of available Donors and thus the Acceptor can seek blood.

In our system, only donors whose age is between 18 and 40 can register; otherwise, they’re not allowed. We have two users who interact with the system: the admin and staff, each user has a specific password to access the system. The admin can update and delete data; on the other hand, staff can search by ID, by blood group or for matching blood groups. When staff searches by ID, all information about this donor appears with his/her medical history, but when staff searches by blood group, all information of the donors who have the same blood group appears. Last but not least, when staff searches for matching blood group a list of matching blood groups appears.

**User Guide:**

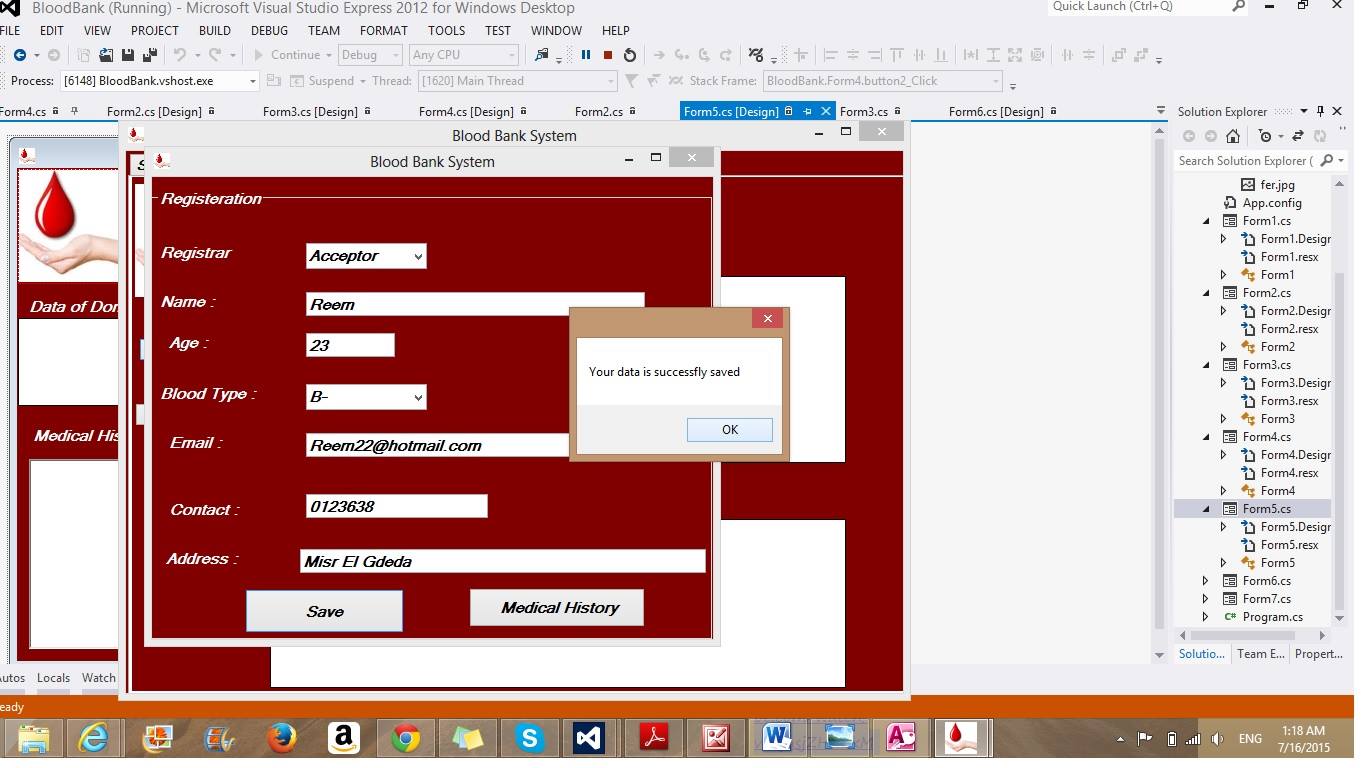
This is a step-by-step User Guide to the system.

Before everything we need to sign in first in the log in page. The user (administrator/normal user) will enter their password in the Textfield shown in the figure. Of course, the administrator has a password different from the user.



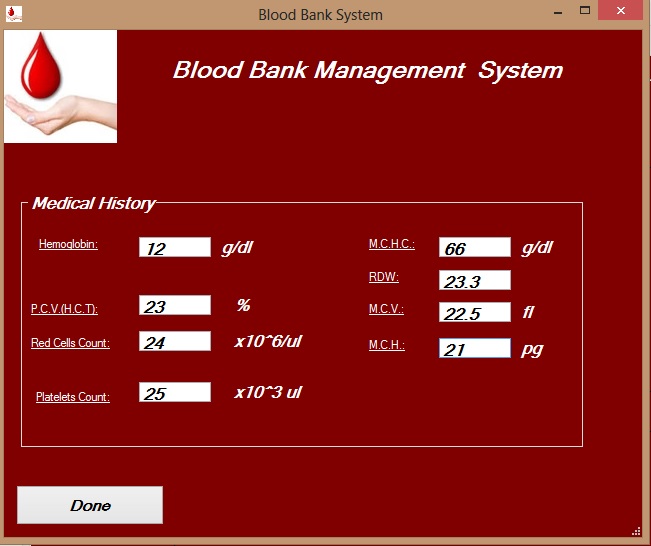
**First : let’s save new registrar in our database**

To save a new registrar, we simply fill the fields shown in the figure. To save the medical history associated with donor/acceptor, we press the medical history button.

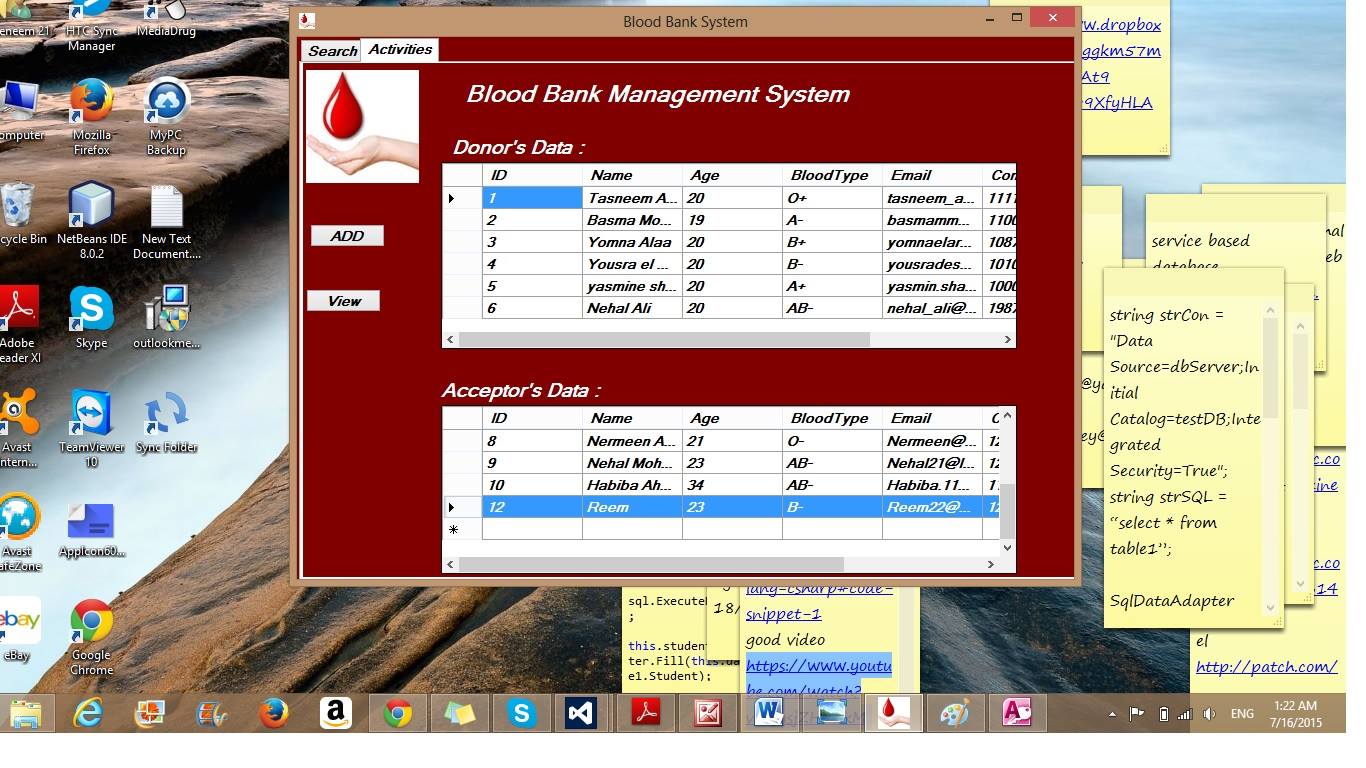
****

**To save the medical history of our registrar**

We fill the fields with the information associated with donor/acceptor, each information in its specified field as shown in the figure.

****

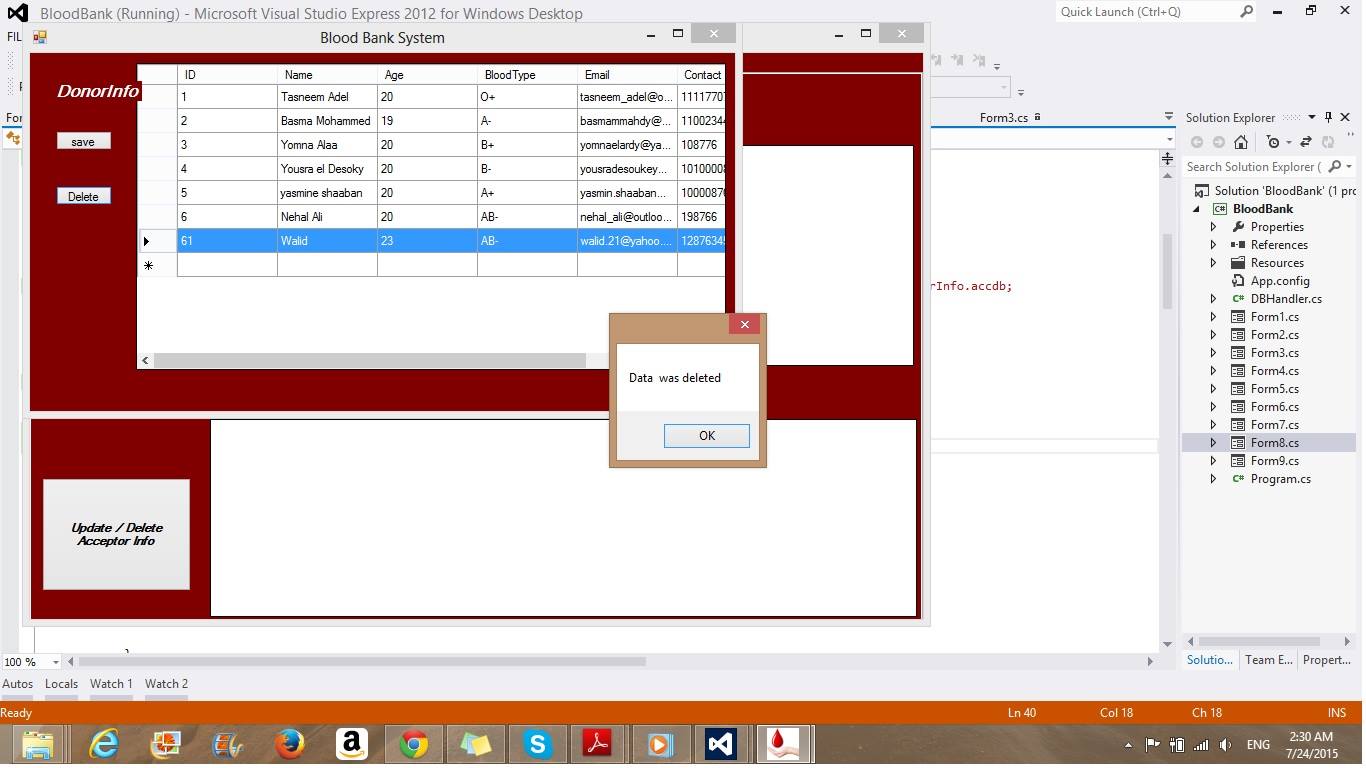
**We want to view all of our data**

****

We press the “View” button and all of the data appears instantly in table forms as shown.

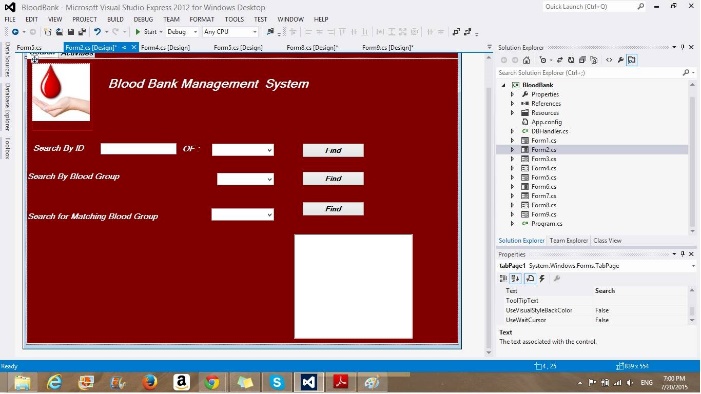
**If we wanted to edit or delete anything in our data**

We basically highlight the required by clicking on the left side of the data then click the “Delete” button.

****

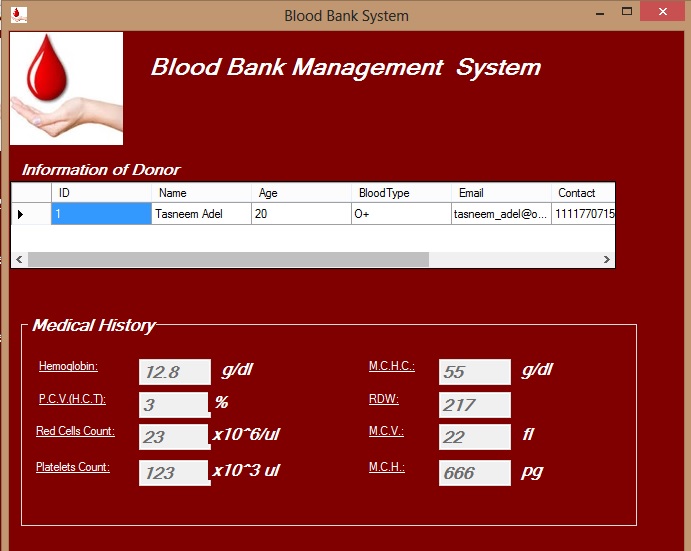
**To search**

We have few search options; by ID, by Blood Group, and for matching blood group. To start the search process, we fill the field (in case of ID) and/or choose from the list what we want to search for. To complete the search process we press the “Find” button.

****

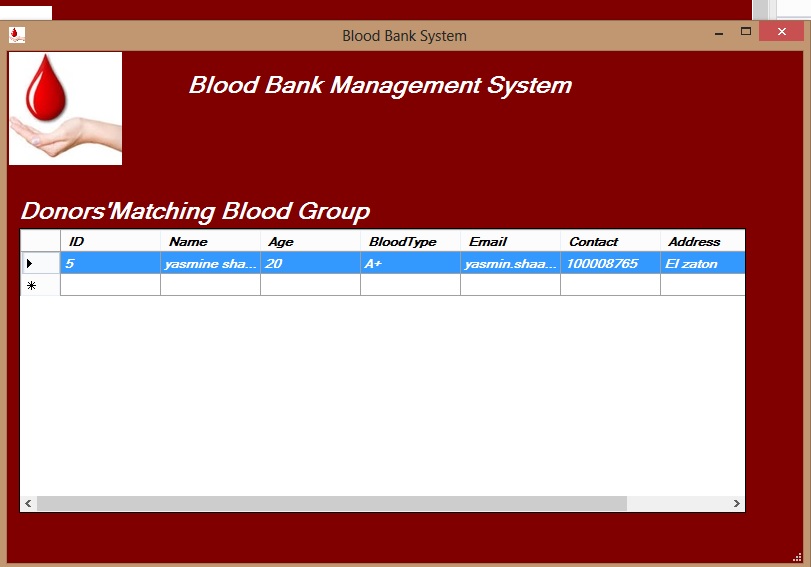
**To Search by ID**

Once we filled the field of ID we would like to search for and choose whether it’s a donor or an acceptor then click “Find”, we get this page. The info of the required appears in the RichTextBox in form of a table as follows. The medical history associated with the donor/acceptor appears in the TextFields shown in its specified place.



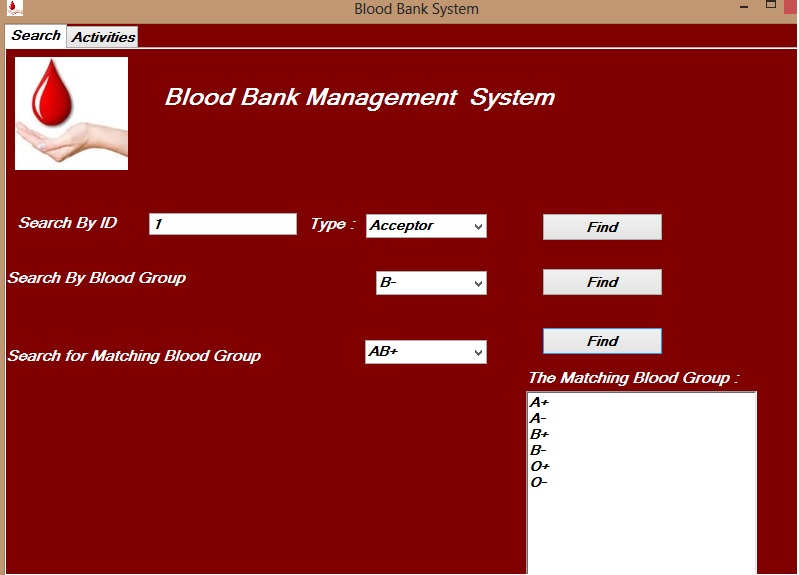
**To Search by Blood Group**

After choosing the blood group required to search for and clicking “Find”, this page appears with all people having the same blood type in form of a table.



**To find Matching Blood Type**

To search for matching blood groups, we pick one from the list and click “Find”. Instantly, the matching blood groups appear in the shown RichTextBox.



**\*\*P.S:**

Numbers and information in the snapshots are just a mere guessing to show a simple example.

**The Work Splitting:**

The work was split among us as the following:

**Basma Mohamed handled**

-Updating Data (Edit).

-Deleting Data from Database.

**Nehal A. Ali handled**

-Retrieving Medical History for Donor from Database (search by ID).

-Retrieving Medical History for Acceptor from Database (search by ID).

**Tasneem Adel handled**

- The login for the system.

-The Registration for Donor or Acceptor.

-The view option (view all data for Donors and Acceptors).

-Matching blood type.

**Yasmine Shaaban handled**

-Searching for all donors that have the required blood group.

**Yomna Alaa handled**

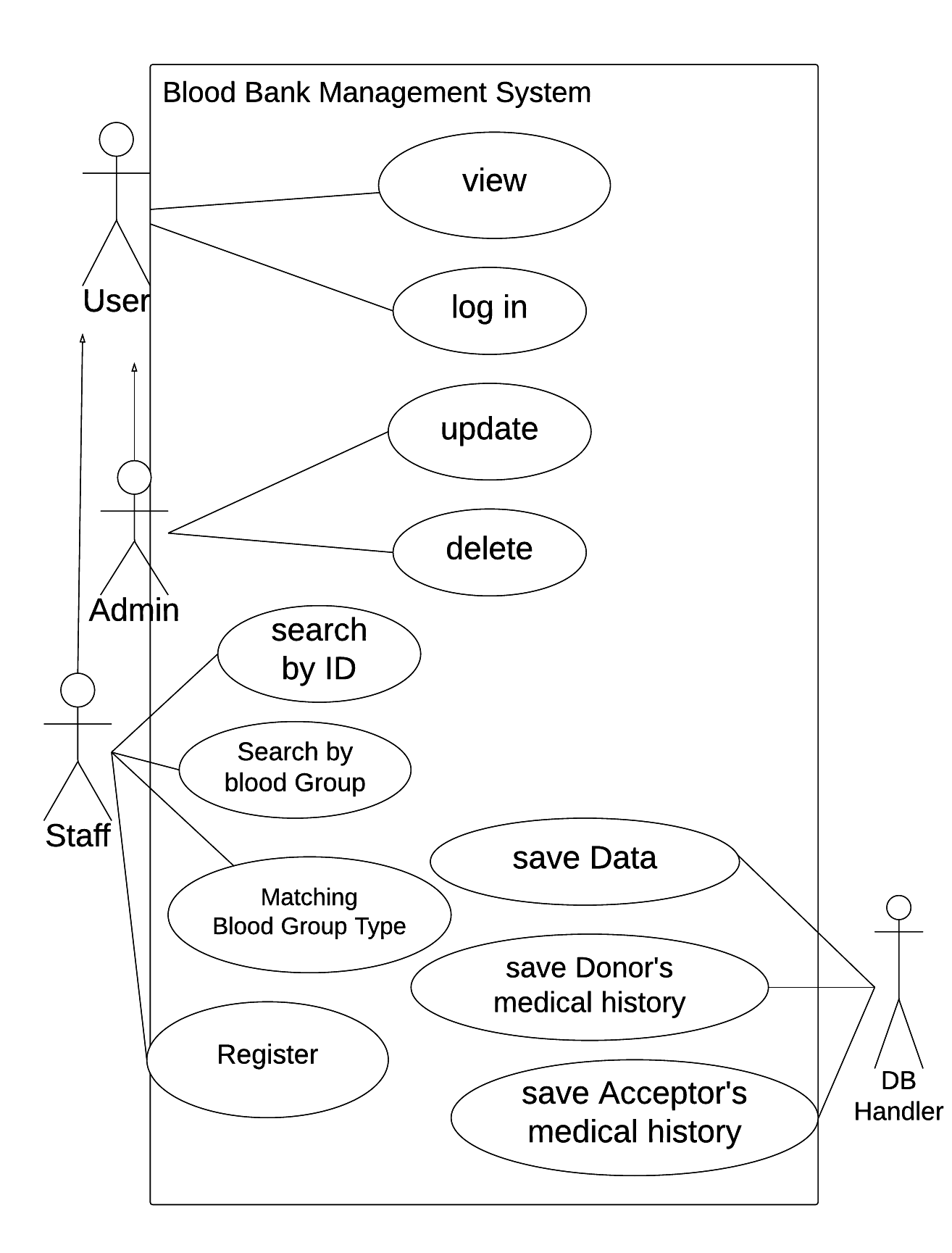
-Searching for Specific Donor in Database by his/her ID.

**Yousra el Desouky handled**

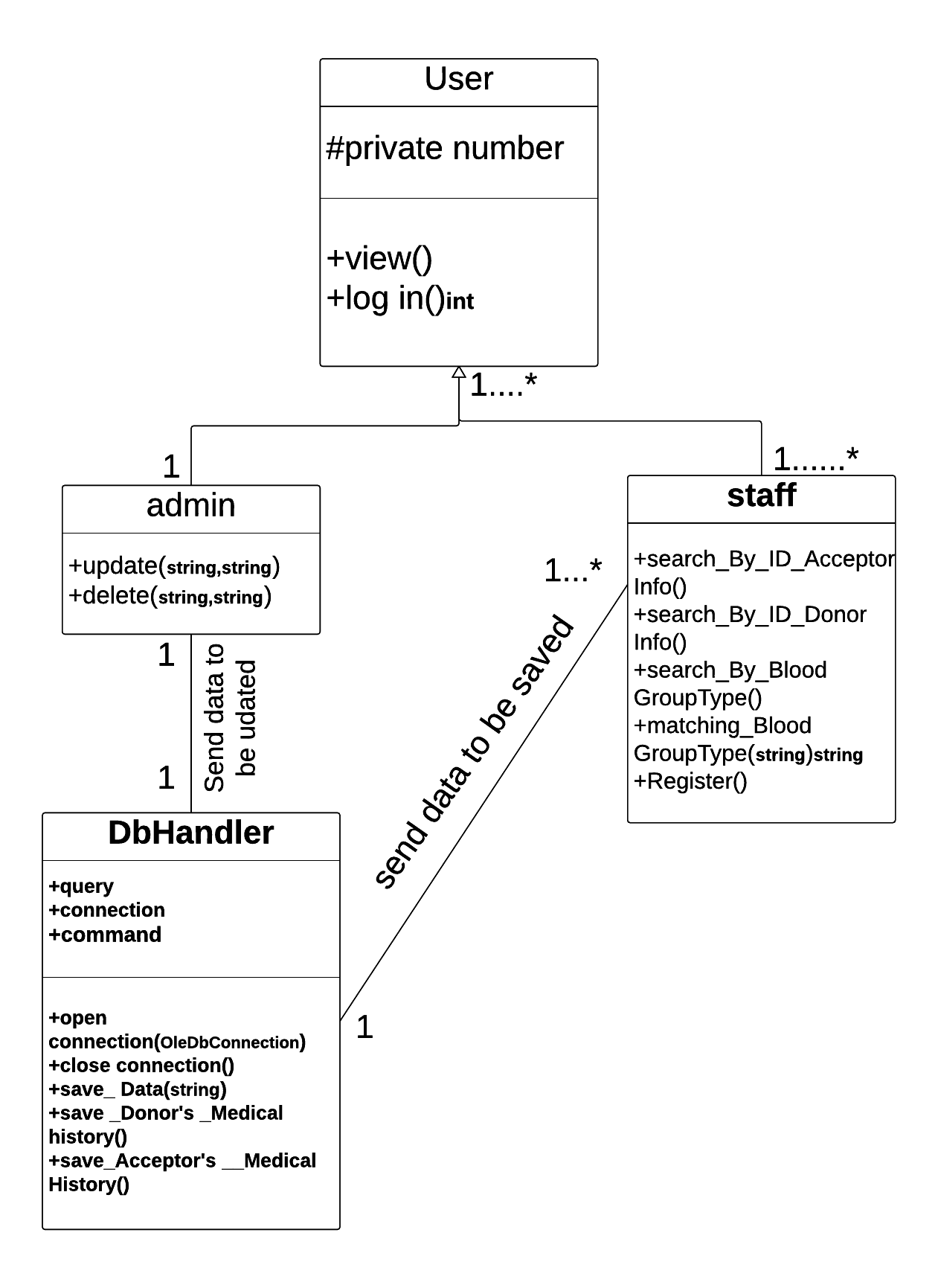
-Saving the Medical History of Donor in Database.-Saving Medical History of Acceptor in Database.   
\*Note: The System requirements, Design and Integration of work were split among us.

**Final Design:**

**Use Case**

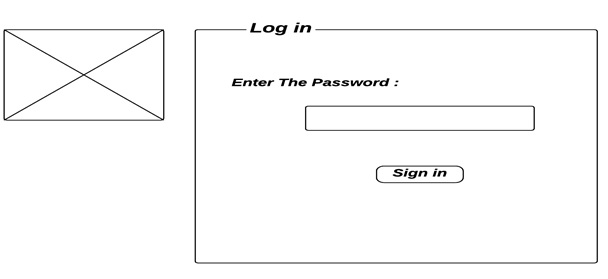


**Class Diagram**



**UI WireFrame**

**Generally, Log in Screen:**



**-Log-in Screen:** enables the administrator and the staff to sign in to the system by entering their passwords.

**-TextBox:** enables the administrator and the staff to type their passwords (maximum number of trials is three times).

**-Sign in Button:**

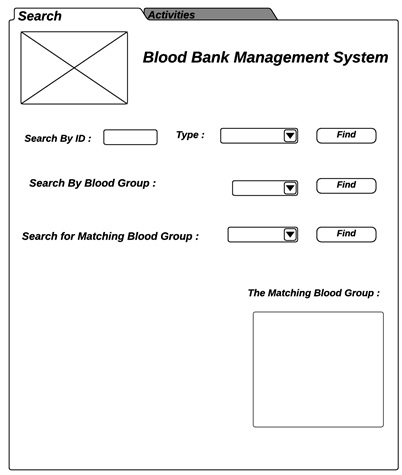
**1-Staff:** When pressed, the entered password is checked -if it is valid- then a new screen (main activity screen) appears with **Search** and **Activities** tabs .

**2-Administrator:** When pressed, the entered password is checked -if it is valid- a new screen (main activity screen) appears with **Activities** tab.

**\*Hint: If password is invalid for three times, screen will be exited automatically.**

***Firstly, Staff:***

**1-Search Screen:**

****

**-Search Screen:** enables the staff to perform the desired search operation.

**-ComboBox (Search By Blood Group):** shows the staff all the existed blood groups.

**-ComboBox (Search for Matching Blood Group):** shows all blood types to enable the staff to choose from them.

**-ComboBox (Type):** shows two options (Donor and Acceptor options) and so, it enables the staff to search for the required donor or acceptor by ID.

**-TextBox:** enables the staff to type the required ID he/she is searching for.

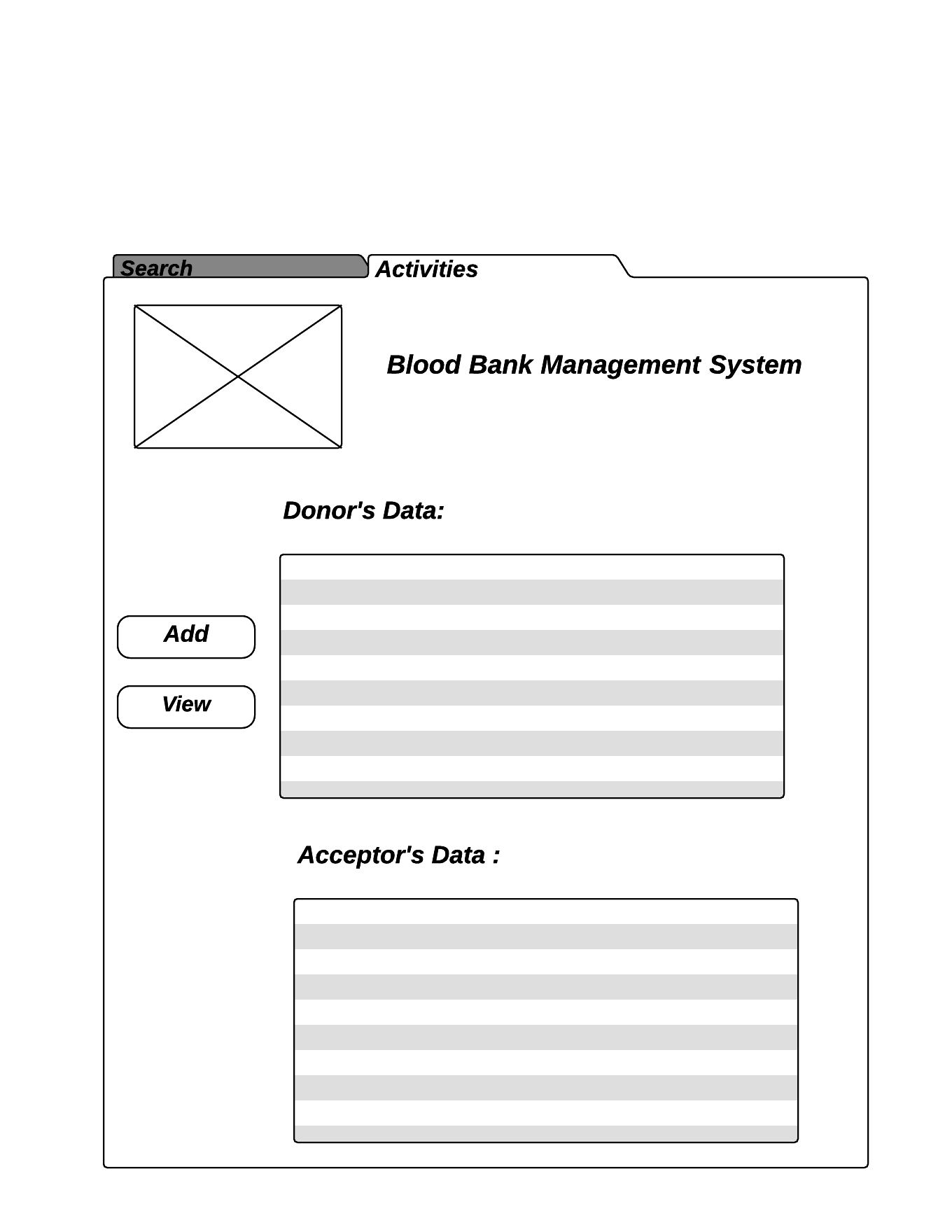
**-RichTextBox:** shows all the matching blood groups with the required one.

**-Find Button (Search By Blood Group):** when pressed, a screen **(Donors’ Matching Blood Group Screen)** appears with DataGridView showing all the information of donors who have the same blood group.

**-Find Button (Search for Matching Blood Group):** when pressed all matching blood groups appear in the RichTextBox.

**-Find Button (Search By ID):** when pressed, a screen **(Info and Medical History Showing Screen)** appears showing all the information of the specified donor with his/her medical history.

**2-Activities Screen:**

****

**-Activities Screen:** enables the staff to perform the desired activity (Add and View activities)

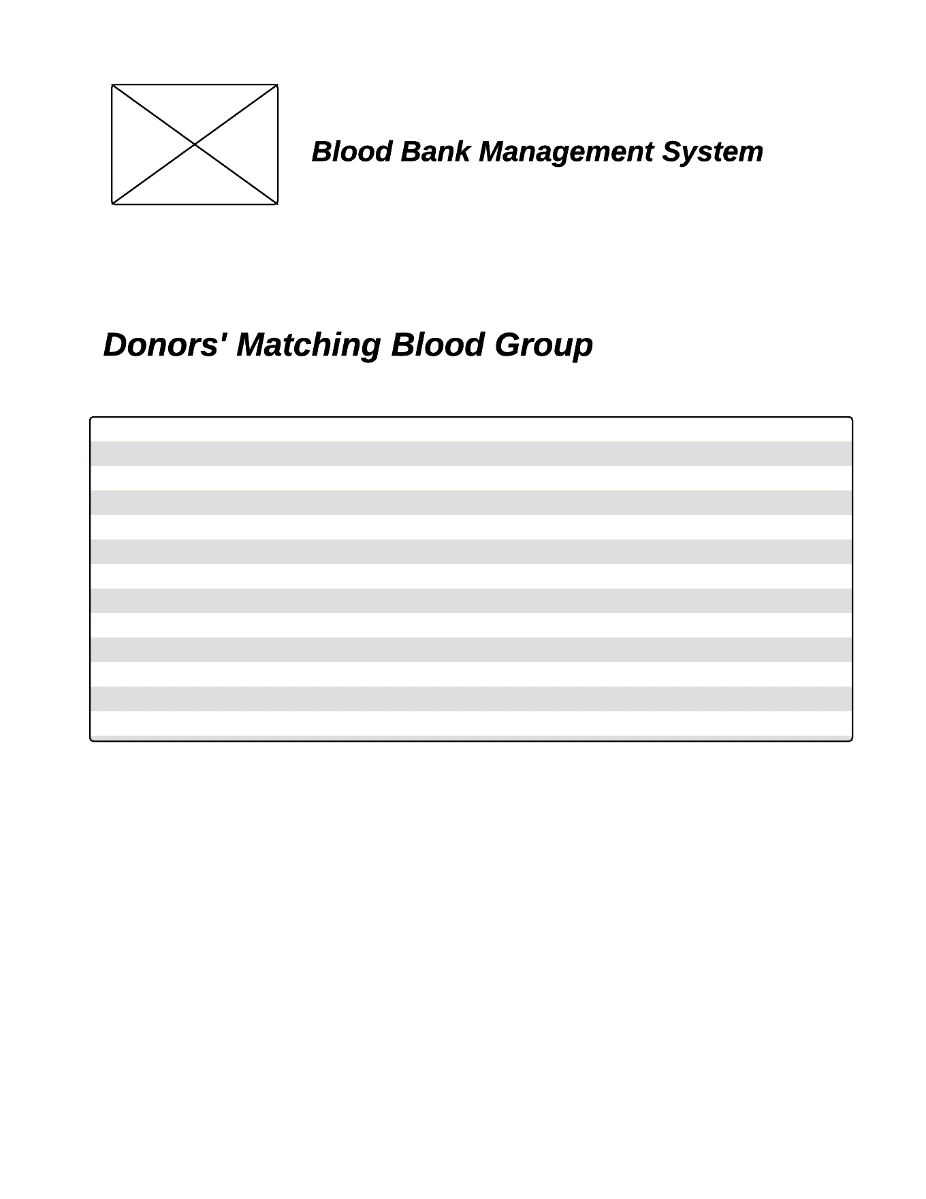
**-DataGridView (Donor’s Data):** shows all information (ID, Name, Age, Email, Address, Contact and Blood-Type) of donors in form of table.

**-DataGridView (Acceptor’s Data):** shows all information (ID, Name, Age, Email, Address, Contact and Blood-Type) of acceptors in form of table.

**-Add Button:** when pressed, a new screen (Registration Screen) appears that enables the staff to add information of new donor or acceptor.

**-View Button:** When pressed, the saved data of all donors and acceptors appears in the two DataGridViews.

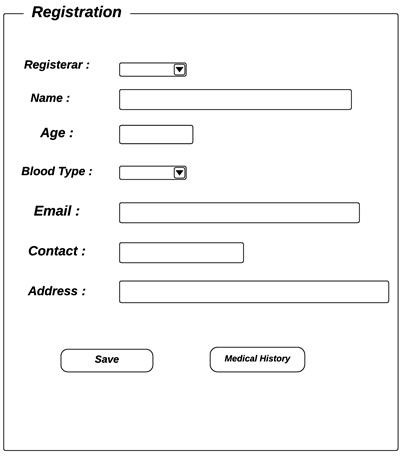
**3-Donors’ Matching Blood Group Screen:**

****

**-Donors’ Matching Blood Group Screen:** enables the staff to get the information of donors who have the required blood group.

**-DataGridView:** shows all information (ID, Name, Age, Email, Address, Contact and Blood-Type) of donors who have the required blood group in form of table.

**4-Registration Screen:**

****

**-Registration Screen**: enables the staff to add the data of new donor or acceptor.

**-ComboBox (Registrar)**: shows two options (acceptor and donor options).

**-ComboBox (Blood Type)**: shows all the blood types.

**TextBoxs**: the new donor’s or acceptor’s information

(Name, Age, Address, Contact and Email).

**-Save Button**: when pressed, the new donor’s info will be saved automatically in database.

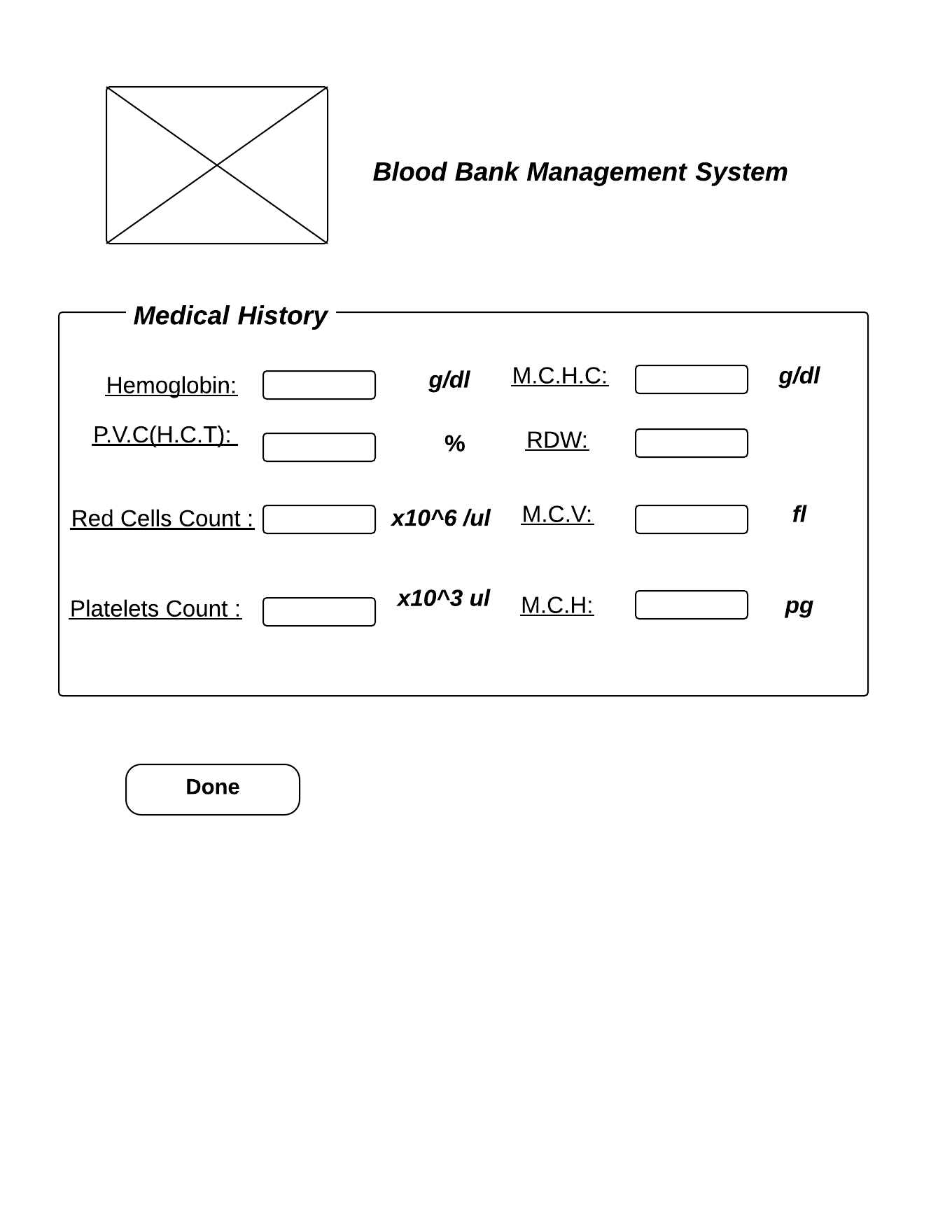
**-Medical History Button**: when pressed, a new screen (Medical History Saving Screen) appears with several TextBoxes that enables the staff to type the medical history of the new donor or acceptor in each field (each Textbox).

**\*\*Hint:**

**-ID of donor is incremented automatically and saved in database.**

**-Age must be from 18 to 40.**

**5- Medical History Saving Screen:**

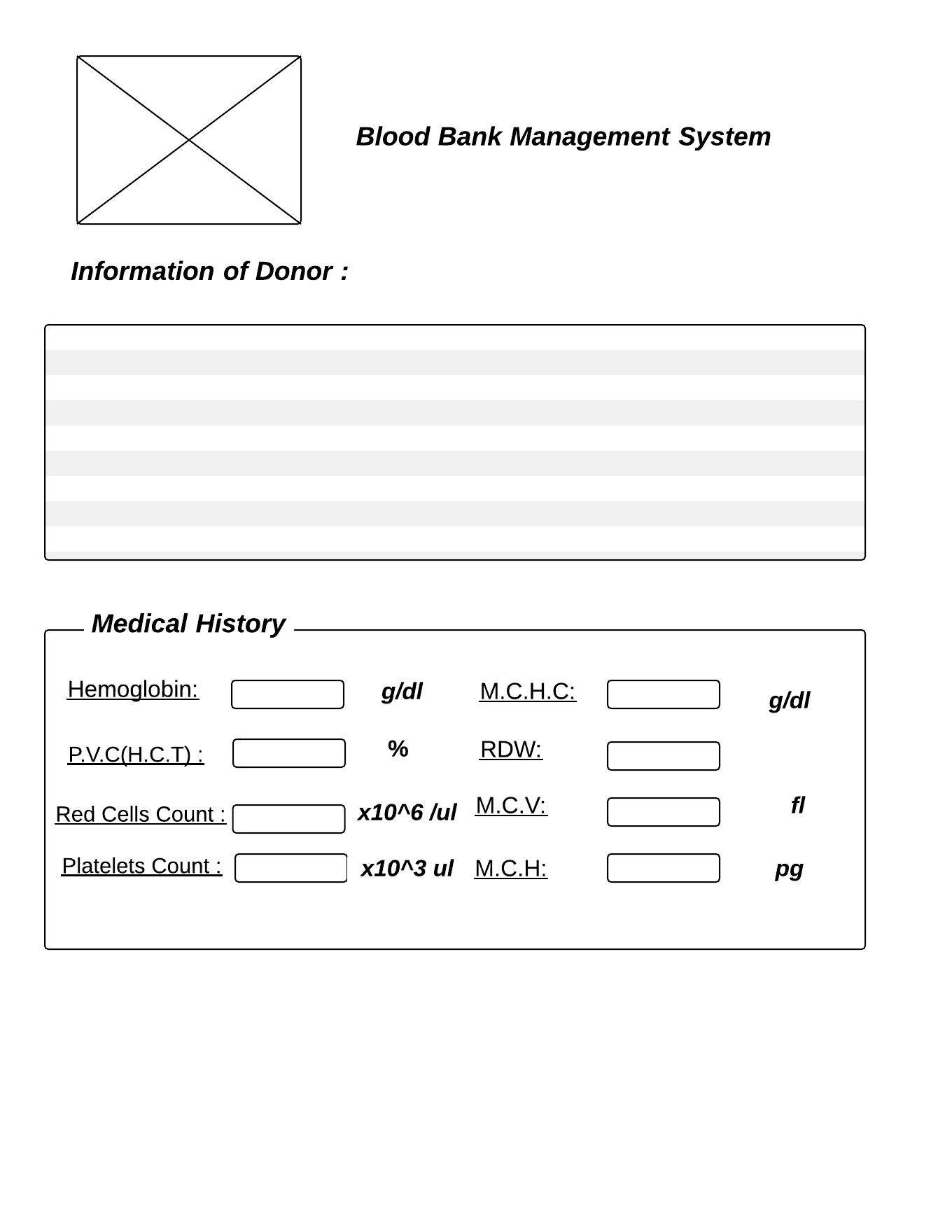
****

**-Medical History Saving Screen**: enables the staff to save the medical history of the new added Donor or Acceptor.

**-TextBox:** enables the staff to type the medical history of the new added Donor or Acceptor. Each piece of information is filled in its specified TextBox field (Hemoglobin, Red Cells Count, M.C.H, M.C.H.C., RDW, M.V.C., P.C.V, and Platelets Count).

**-Done Button:** when pressed medical history is saved in database.

**6- Info and Medical History Showing Screen:**



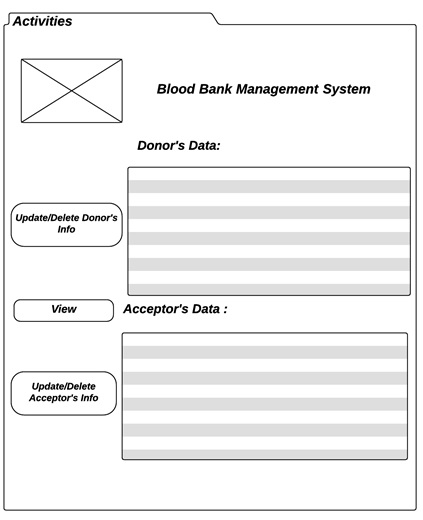
**Info and Medical History Showing Screen**: shows the both and medical history of required donor or acceptor (searching by ID).

**-DataGridView**: shows the information of the required donor or acceptor in form of table.

**-TextBoxes**: show the medical history of required donor or acceptor, each piece of information in its specified field (Hemoglobin, Red Cells Count, M.C.H, M.C.H.C., RDW, M.V.C., P.C.V, Platelets Count).

***Secondly, Admin:***

**1-Activities Screen:**

****

**-Activities Screen :** enables the administrator to perform the desired activity (Update, Delete and View activities).

**-Update/Delete Acceptor’s Info Button:** when pressed, a new screen **(Update and Delete Acceptor’s Info Screen)** appears.

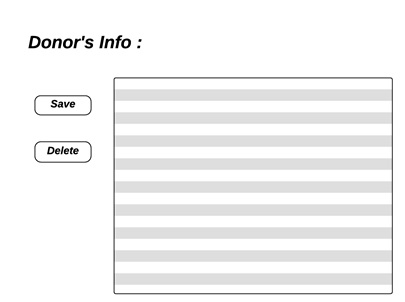
**-Update/Delete Donor’s Info Button:** when pressed, a new screen **(Update and Delete Donor’s Info Screen)** appears.

**-DataGridView (Donor’s Data)**: shows the information of all donors in form of table .

**-DataGridView (Acceptor’s Data):** shows the information of all acceptors in form of table.

**-View Button:** when pressed, the DataGridViews show the information of all donors and acceptors.

**2-Update and Delete Donor’s Info Screen :**

****

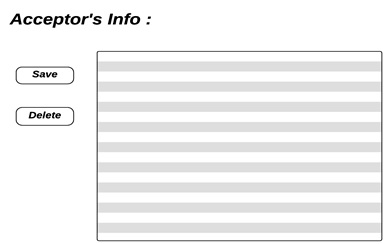
**-Update and Delete Donor’s Info Screen:** enables the administrator to update and delete the desired information of any added donor.

**-Delete Button:** when pressed, the information of any unwanted donor is deleted from database by selecting the ID cell of the row of the DataGridView .

**-Save Button:** when pressed, the information of any wanted donor is updated in database.

-**DataGridView:** shows the already saved information of donors and enables administrator to delete or update any information of any donor.

**3-Update and Delete Acceptor’s Info Screen:**

****

**-Update and Delete Acceptor’s Info Screen:** enables the administrator to update and delete the desired information of any added acceptor.

**-Delete Button:** when pressed, the information of any unwanted acceptor is deleted from database by selecting the ID cell of the row of the DataGridView.

**-Save Button:** when pressed, the information of any wanted acceptor is updated in database.

-**DataGridView:** shows the already saved information of acceptors and enables administrator to delete or update any information of any acceptor.

**Code Documentation:**

**The access of the system:** (Form1)

When the user enter the password ,the password is taken and checked if it’s valid or not .there’s a static counter that incremented when the user enter the password wrong and if it exceeds 3 times trial the counter set to zero and the form closed automatically .

**The Matching blood group:** (search tab)

When the user select a choice (blood type), the text of ComboBox is taken and compared with the saved choices till find its match then the text of rich box is updated with matching blood group.

**Searching by ID:** (search tab)

In this part the user should choose from the list Donor\Acceptor and write the ID in the text box in form 2, if the choice was Donor the integer cout “in form 2 code” will be assigned to “1” so the connection to the database “DonorInfo” will be opened and then we select the fields from the table which is equal to the ID written in the textbox in form 2, we wrote “\*” which means that we want all the fields, and then show it in the dataGridView in form 5.

And if the choice was Acceptor then cout will be equal to 2 and the connection of database “AcceptorInfo” will be opened and the fields in it and of the number of ID will be copied in the dataGridView as we said before.

**Searching by blood group:** (search tab)

In this part , the user chooses the required blood group from the comboBox (A-, A+, B+, B-, AB-, O-, O+ and AB+) which is in Form2 then by pressing the Find button, the program connects to Microsoft Access Database and the information (ID, Age, Name, Contact, Email, Address and BloodType -which is the required one-) about donors who have the required blood group appears in form of table in the dataGridView which is in Form6.

**View Option:** (Activties tab)

When button view is pressed the program opens connection to database and connects to it .the package that used is System.Data.OleDb.   
The classes that used are the oleDbConnection,  
aoleDbcommand , OleDbDataAdapter and DataTable.

After the connection is opened an object is created from class aoleDbcommand and the query that select all the data from database (“Select \* from tablename”) set to its object using property (commandText) ,That will be done twice since the data will be got from two tables (DonoInfo and AcceptorInfo).

The data is got using OleDbDataAdapter’s object and that object fill the table that created from DataTable class and finally the object that created from DataTable that is filled with data from database is set to DataGridView.

**ADD option:**  (Form 4)

When Add Button pressed the registration form appear to provide the user with the facility to enter his/her data .After the user pressed save button , the text in combobox is taken and checked if it’s Donor or acceptor to specify which table will be Selected to save the data in. After checking either it’s Donor or Acceptor a static variable (cin) set to whether “1” for Donor or “2” for Acceptor to facilitate for detecting the type of Registrar when his/her Medical History added for him/her. The program open connection with database and execute the query “insert “using aoleDbcommand’s object.commandText.

The values that provided by user in text boxes are taken and set to query to be saved in database.   
The method ExceuteNonquery will be used since the query is insertion.  
 If the registrar was Donor the text of Age’s textbox is taken to be checked if the age of registrar is between (18) and (40) years old or not. If the age of the donor out of the required range the program will throw AgeException and a messagebox will appear “The age must be between 18 and 40 “ ,another one ”You can’t Donate” then the form is closed. Also if the Data is saved correctly a static variable (ct) is set for “1” to allow for user to save Medical History if it doesn’t set to “1” The Medical History Form won’t appear instead an error Message to alert that the Registration must done first.

**Edit (save button):** (Form 8/Form9)

The admin log-in by the password which is only for the admin when button modify is pressed the program opens connection to database and connects to it .the package that used is System.Data.OleDb.

The classes that used are the oleDbConnection, aoleDbcommand, OleDbDataAdapter and DataTable.

After the connection is opened an object is created from class aoleDbcommand and the query that select all the data from database (“Select \* from tablename”) set to its object using property (commandText).

The data is got using OleDbDataAdapter’s object and that object fill the table that created from DataTable class and finally the object that created from DataTable that is filled with data from database is set to DataGridView and it will be opened in a new form this will be done either for the acceptor or donor's info tables after that the user will click on any value he wants to edit in the table then he will click save, the database will be updated with the edited data.

**Delete(delete button):** (Form 8/Form9)

The admin logs in by the password which is only for the admin, when button modify is pressed the program opens connection to database and connects to it .the package that used is System.Data.OleDb.

The classes that used are the oleDbConnection, aoleDbcommand , OleDbDataAdapter and DataTable.

After the connection is opened an object is created from class aoleDbcommand and the query that select all the data from database (“Select \* from tablename”) set to its object using property (commandText).

The data is got using OleDbDataAdapter’s object and that object fill the table that created from DataTable class and finally the object that created from DataTable that is filled with data from database is set to DataGridView and it will be opened in a new form this will be done either for the acceptor or donor's info tables after that the user will click on any user he wants to delete in the table then he will click delete, the database will be updated with the edited data.

**Done button** : (Form 7)

When the “done” button is pressed the program connects to the data base using “System.Data.OleDb.” package.

“aoleDbcommand” class is used.

Firstly, the connection to table (DonorInfo/AcceptorInfo) is established and query is established that retrieve The Id of registrar and saved in local variable (Regid).

The query that used “select ID from tablename where Name=”+Form4.text+” ”;

Where Name is used in query to get ID of Donor/Acceptor since it ‘s unique as each Registrar has a unique Name. The text is a static field in Form4 class that gets the Name of Registrar from textbox and accessed by its class.After the query is executed (by ExecuteScalar()) .The retrieved ID saved in Regid .Then the connection to Database is closed.

Secondly,After the connection is opened another object is created from class aoleDbcommand and the query that insert the medical history into the database (“insert into MedicalHistory(Hemoglobine,RedCellsCount,PCV,MCV,MCH,MCHC,RDW,PlateletsCount,RegID) values('" + hemog.Text + "','" + rcc.Text + "','" + pcv.Text + "','" + mcv.Text + "','" + mch.Text + "','" + mchc.Text + "','" + rdw.Text + "','" + pcount.Text +"','"+Regid+ "')"”) set to its object using property (commandText) , and this will be done twice since the medical history will be inserted into two tables either for Donor or Acceptor. After the execution of query (by ExecuteNonQuery()) the Data is saved in database and the connection is closed and a confirmation message appears.