ABSTRACT

Blood Bank Management System is a browser based system that is designed to store, retrieve and analyze information concerned with the administrative management within a blood bank. It aims at automating the existing manual system of management of a blood bank effectively. It has been developed to override the problems existing in the prevailing manual system by keeping records for a longer time with ease of access and manipulation.

This system allows the administrator maintaining all the information to access details the blood donors, recipients of blood from the blood bank, hospitals blood has been sent to, different blood groups available in stock within the blood bank and those out of stock. The administer can also search for donors of any specific blood group, the contact details of previously visited donors having the blood group will be provided. This feature can prove extremely useful at the time of emergency. It is developed in a manner that is easily manageable, time saving and relieving one from manual works. This system will ultimately allow better usage of resources

Requirement Gathering

The requirement analysis stage of a software engineering project involves collecting and analyzing information about the part of the organization that is supported by the application. This information is then used to identify the users requirement of the new system. Identifying the required functionality of the system is very important as a system with incomplete functionality may lead to it being rejected

**Analysis of the Existing System**:

The operation of the blood bank still now is maintained in the manual system.

• The operation is tedious, time consuming and space consuming.

• It creates room for errors as the data is entered manually by the persons.

• It includes the risk of the documents being lost over years and maintenance of the records is difficult.

• The data recorded during testing or while acquiring the details of different aspects of blood bank management system is not so accurate and precise.

• Maintaining the stock of blood and the daily transactions without computerization also poses a challenge.

**PROBLEM STATEMENT**

• Scarcity of rare blood group.

• Unavailability of blood during emergency.

• Less awareness among people about blood donation and blood transfusion.

• Deaths due to lack of blood during operations.

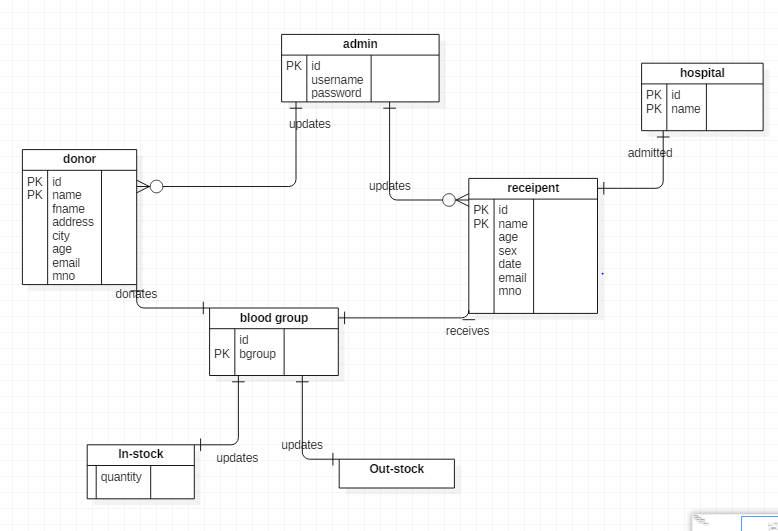
**Functional requirements:**

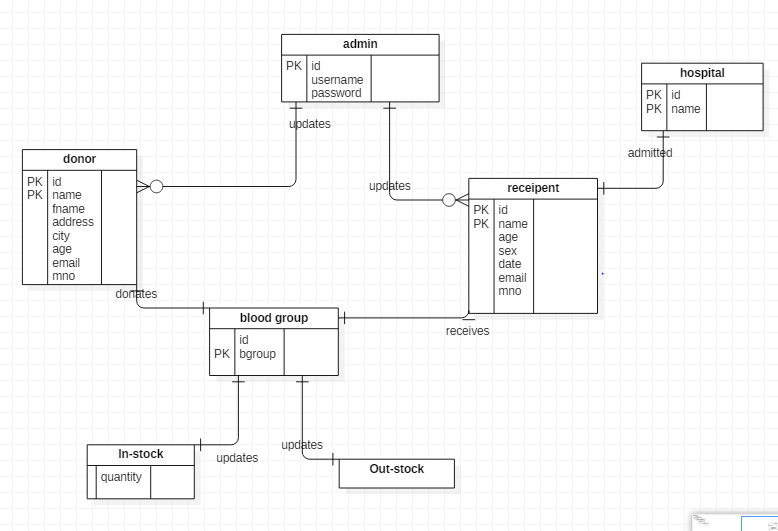
* The system must have xampp app installed in order to access the backend databases on phpmyadmin
* Login details of phpmyadmin (Username and password)
* Admin Login details of blood bank (Same as saved in backend)
* HTML has been used for creating the layout of the web application
* CSS has been used for creating the designing of the application
* Bootstrap has been used for modifying of the application
* JavaScript has been used as a client side scripting language.
* PHP has been used to connect frontend to backend.

**Non-Functional requirements:**

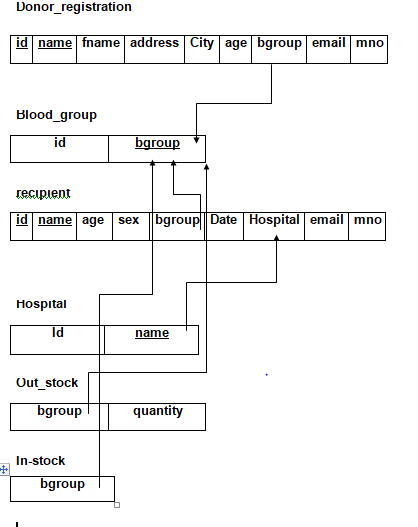
* Capacity, scalability and availability Maintainability
* Performance- System should be able to handle multiple admins at a time
* Reliability- Database updating should follow transaction processing to avoid data inconsistency
* Availability-The project will be deployed on a public shared server so it will be available all the time and will be accessible anywhere of the world using internet
* Security-implementation a lot of security mechanism to avoid hacking and data loss
* Maintainability- It is easy to maintain the system .
* Portability- System is portable and can switch servers.
* Browser Compatibility-The project being web based requires compatibility with at least the popular web browsers.
* Performance-The response time for access the database will be very less.

**ER Diagram**

****



**Relational Schema**



**Integrity Constraints**

* Integrity constraints are a set of rules. It is used to maintain the quality of information.
* Integrity constraints ensure that the data insertion, updating, and other processes have

to be performed in such a way that data integrity is not affected.

* Thus, integrity constraint is used to guard against accidental damage to the database

**Relationship-Cardinality**

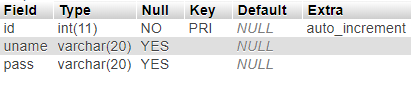
* Admin updates donor (1---N)
* Admin updates recipient (1---N)
* Donor donates blood group (1---1)
* Recipient receives blood group (1---1)
* Recipient admitted to hospital (N---1)

**DATABASE:-**

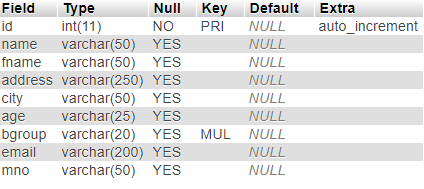
**The name of database is mypro\_bbms which consist of 7 tables, 2 views and 3 triggers as follows.**

* **Tables:-**

**1.admin table:**

****

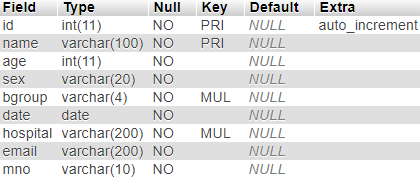
**2.donor\_registration table:**

****

**3.blood\_group table:**

****

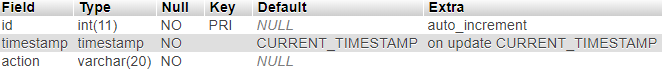
**4.recipient table:**

****

**5.Hospital table:**

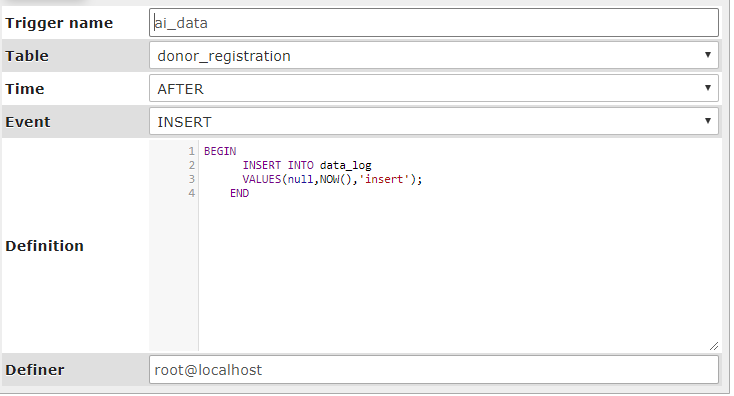
****

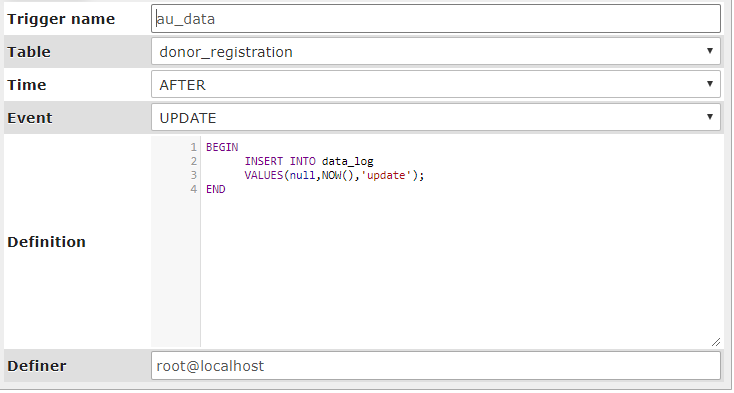
**6.The two data\_log tables:**

****

Values are inserted into the tables with the use of three triggers

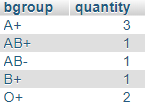
* **Triggers:**

a)

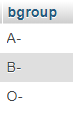
b)

c)

* **Views:**

**1.Stock\_blood\_list:**

Create View Stock\_blood\_list as select `mypro\_bbms`.`donor\_registration`.`bgroup` AS `bgroup`,count(`mypro\_bbms`.`donor\_registration`.`id`) AS `quantity` from `mypro\_bbms`.`donor\_registration` group by `mypro\_bbms`.`donor\_registration`.`bgroup`;

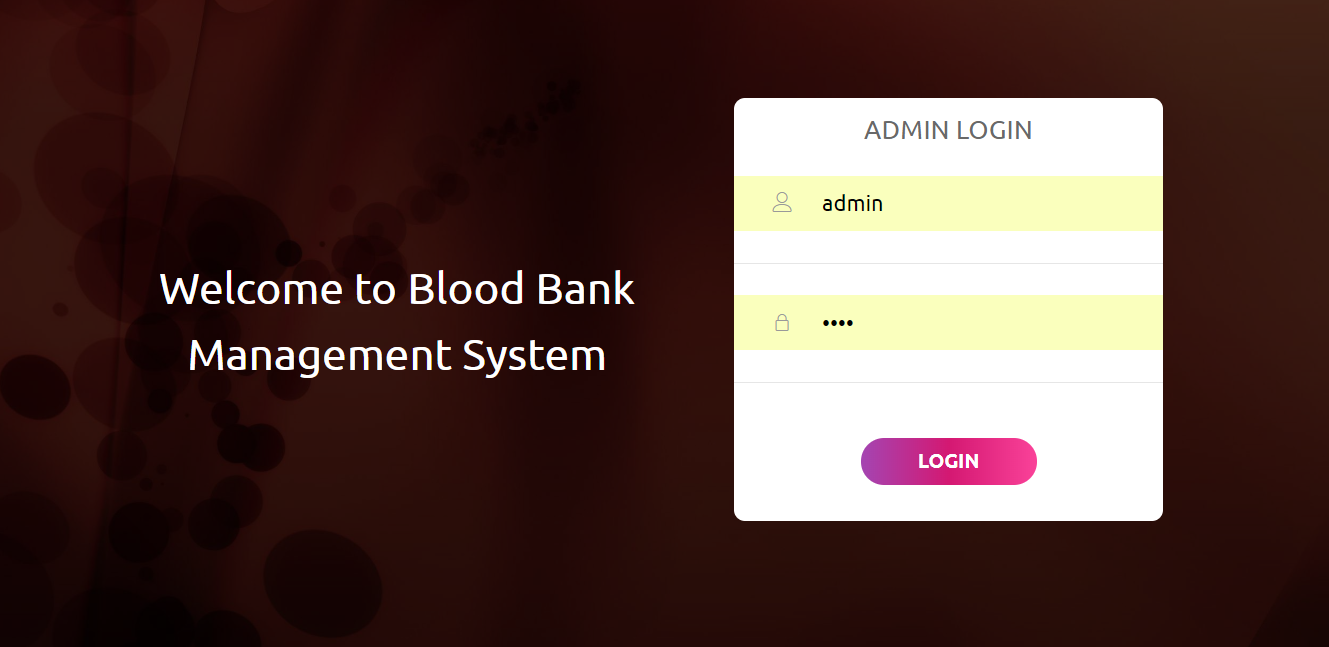
**2.Out\_stock\_blood\_list:**

Create View out\_stock\_blood\_list as select `a`.`bgroup` AS `bgroup` from `mypro\_bbms`.`blood\_group` `a` where (not(`a`.`bgroup` in (select `b`.`bgroup` from `mypro\_bbms`.`stock\_blood\_list` `b`)));

**Screenshots:**

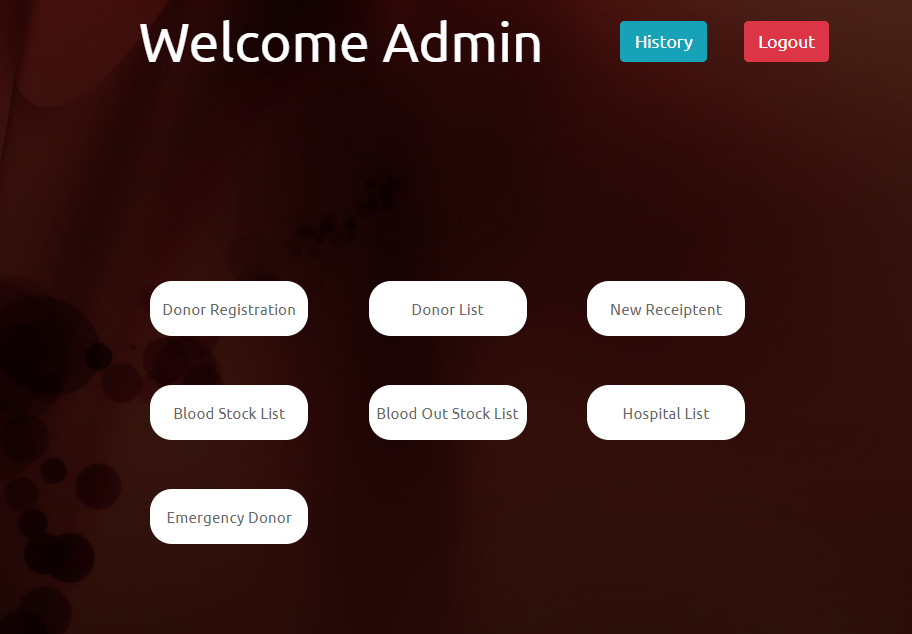
**Login Page**

The admin is required to first enter his login credentials .i.e. his username and password. The login credentials of the admin are saved in the admin table in the backend database.

****

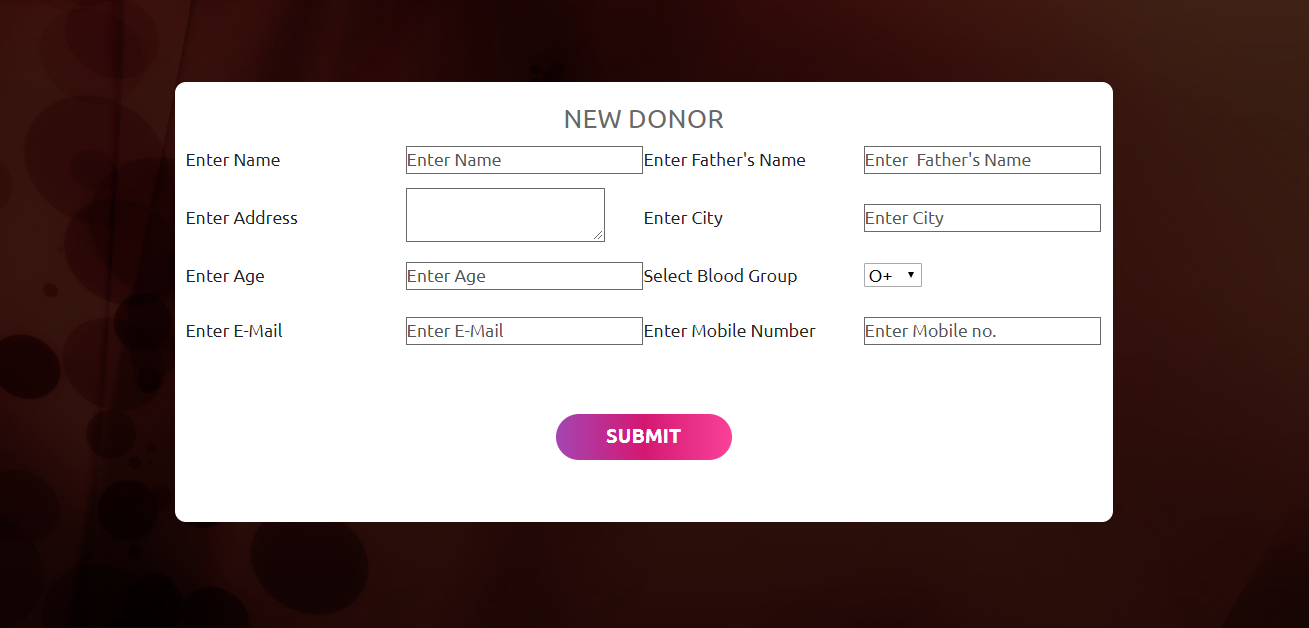
**Home Page**

The home page is a user friendly basic UI displaying all the various functions the admin can perform by the application.

****

**New Donor Registration**

Here, admin can fill the registration details of the donor. The data will be updated in the donor\_registration table in the backend database.

****

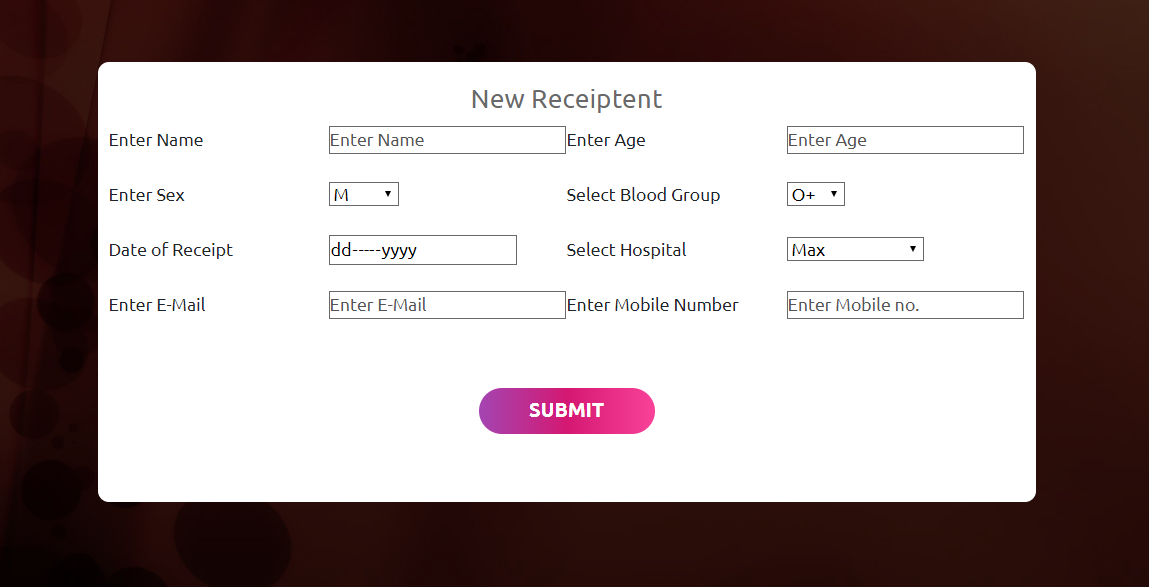
**Donor List**

Here, the list of all the donors is displayed from the donor\_registration table in the backend.

****

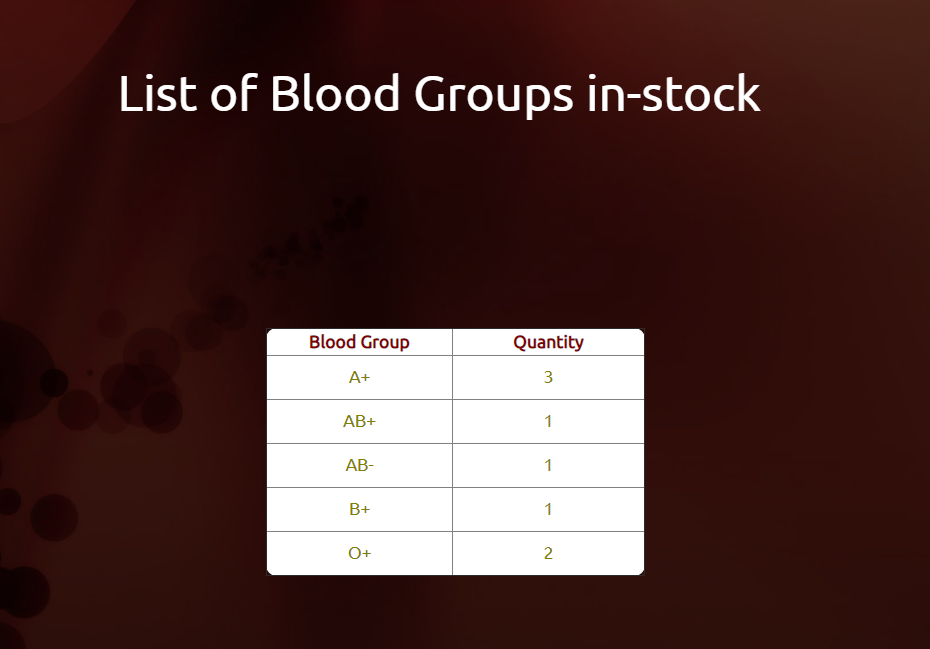
**New Recipient**

Here, the admin can make entries of new recipients (i.e.) patients who have received blood from the blood bank. The data will be saved in the recipient table in the backend database.



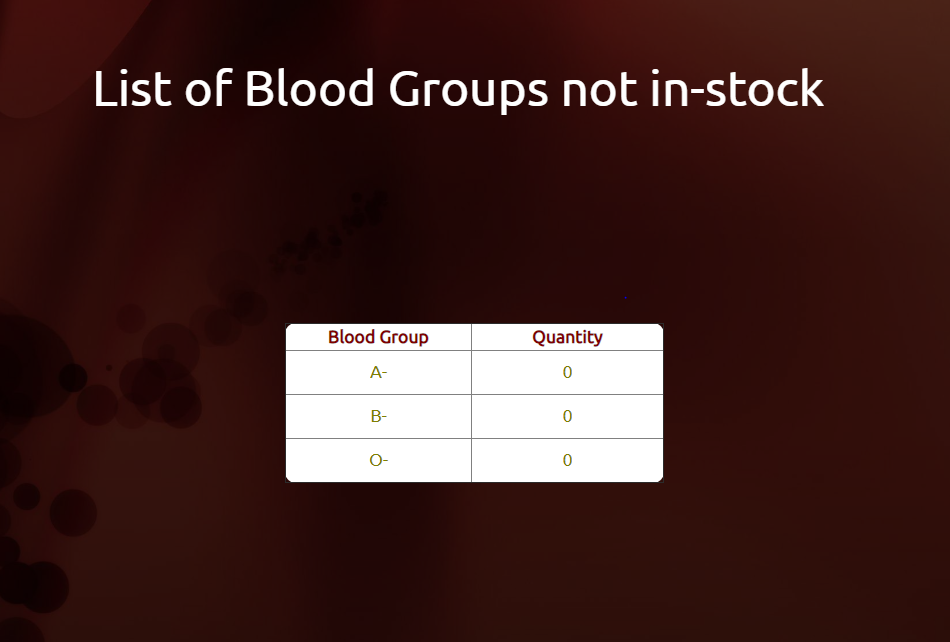
**Stock-Details**

Here, data from the stock view created in the backend database from the donor\_registration table is displayed.

****

**Out-Stock Details**

Here, data from the out\_stock view created in the backend database from the donor\_registration table is displayed.

****

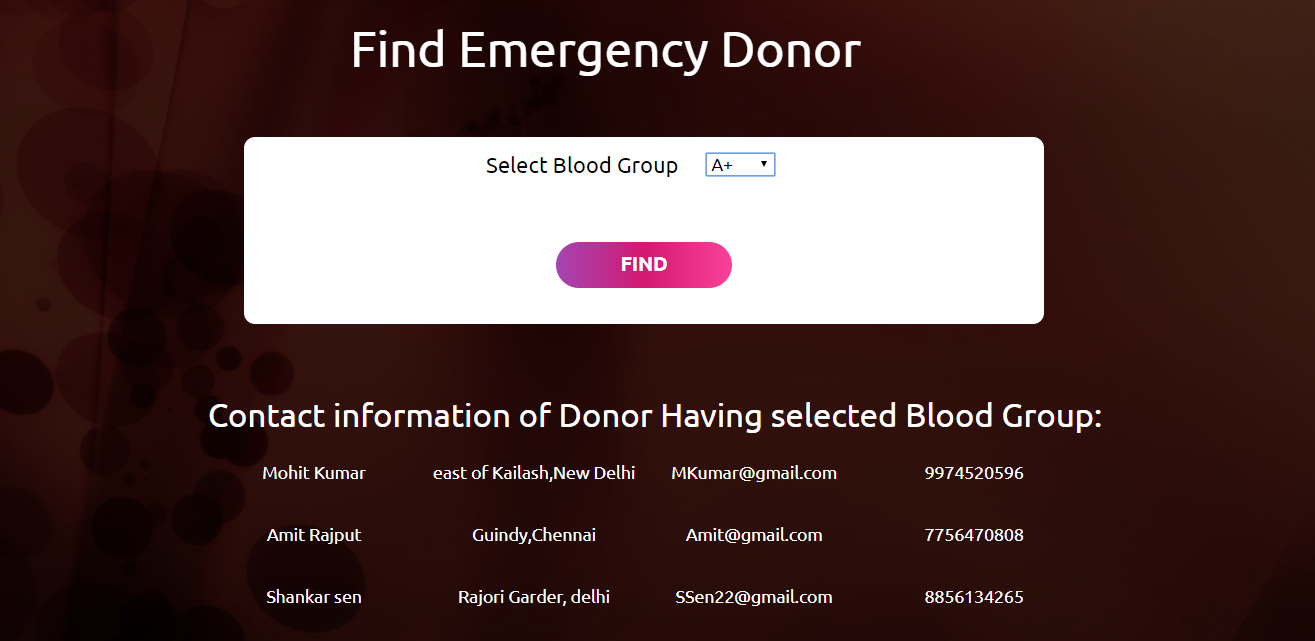
**List of Hospitals Blood Bank has provided to**

Here, the data from the hospital table is displayed. Data is retrieved from the recipient details and inserted into the hospital table while entering. Hospital acts as a foreign key in recipient table.

****

**Finding an emergency donor**

Here, donors are searched from the donor table in the database to find the donors having selected blood group. Their respective name, address, emails and phone number is displayed.

****

**View History**

Here, data of Data\_Log table is displayed which is auto filled by 3 triggers on donor table and recipient table, each, for after insert, after update and after delete. The admin can view his action history here.

****