

BLOOD DONATION SYSTEM

FINAL REPORT

Abstract

This paper includes a summary of system requirements, database design, general definition of the tools and an outline of the means of implementation of a blood donation management system. The objective of this project is the creation of a database management system capable of facilitating the stress-free management and organization of a blood donation database capable of being implemented easily by an NGO. The E-R diagram of the system, analysis/specifications, data flow diagram of the system, database schema with detailed tables, relational mapping, entity and relation descriptions, constraints and scripts of each table are outlined in detail. Subsequently, datasets and screenshots are provided to allow for visualization of the system for the users.

DONE BY : UDAYKUMAR KHEMEWAR

ROLL NO : 2K19/CO/417

Database Management Systems

Delhi Technological University

TABLE OF CONTENTS

LIST OF FIGURES	iv
LIST OF TABLES	v
SUMMARY	1
WHY THIS PROJECT?	1
SPECIFICATIONS.....	1
GENERAL DESCRIPTION	2
REQUIREMENT ANALYSIS.....	2
TOOLS/IDES	3
UML USE CASE DIAGRAM.....	4
E-R DIAGRAM.....	5
DESIGN PHILOSOPHY.....	5
NORMALIZATION	9
E-R TO RELATIONAL MAPPING	11
DATABASE SCHEMA	12

SCRIPTS.....	18
VIEWS	24
DATABASE APPLICATION.....	26
SCREENSHOTS AND DATASETS	28

LIST OF FIGURES

Figure 1: Tools/IDEs	3
Figure 2: UML diagram.....	4
Figure 3: Entity-Relationship diagram.....	5
Figure 4: Example of normalization step	10
Figure 5: DATABASE SCHEMA FROM MYSQL.....	17
Figure 6: START PAGE.....	28
Figure 7: DONOR/RECIPIENT SIGN UP PAGE.....	29
Figure 8: BLOOD BANK SIGN UP PAGE.....	30
Figure 9: BLOOD BANK ADMIN PAGE.....	31
Figure 10: BLOOD BANK PROFILE PAGE	31
Figure 11: LIST OF DONATIONS CENTRES	32
Figure 12: UPDATING DONATION CENTRE INFORMATION	33
Figure 13: DETAILES OF THE DONATION CENTRES	34

LIST OF TABLES

Table 1: ADDRESS TABLE	12
Table 2: STATUS TABLE	12
Table 3: USER TABLE	13
Table 4: BLOOD_GROUP TABLE	13
Table 5: DONOR TABLE	14
Table 6: MEDICAL_INFO TABLE	14
Table 7: RECIPIENT TABLE	15
Table 8: BLOOD_BANK TABLE.....	15
Table 9: GIVES_TO TABLE	16
Table 10: TAKES_FROM TABLE	16

SUMMARY

The importance of an efficient blood bank cannot be overstated and drives this project. The goal is to design a high-level database which would be easy to implement and effective. The design philosophy is created with three main users in mind; a **donor, recipient and blood-bank**. The entity-relationship diagram was created using **LucidChart**, a website that allows and assists users in designing various types of diagrams and charts. Finally, **JDBC** would be main tool used in this project due to the proficiency of the team using this.

WHY THIS PROJECT?

Investigating the methods and technologies used to connect blood-donors to potential patients and blood-banks, it could be said that advancements to these systems in the aspects of ease of storing data for easy retrieval and potential ease of access could be improved. Rather than naive paper means of collecting and storing information, a web interface supported by an efficient SQL database can be developed and implemented.

SPECIFICATIONS

Last year, the red cross visited our university and asked the students to help out by donating blood. However, they were asking the students to fill out long papers which many students including us did not think it was efficient for such a thing. So, we got the idea to create a database management system that is intended to connect the people who are willing to donate blood with the people who are in need of it. It will require the user to create his own account and enter his information (Name, email, blood type, etc.). There will be three options to choose from, either donor or recipient or blood bank administrator, and based on that, the system will look through the database which will check the needs of the blood bank and find the nearest donation center.

GENERAL DESCRIPTION

The blood donation system we aim to build with the implementation of our database would aim to facilitate the easy and efficient connection of blood donors and recipients to blood-banks. The system would register users in three groups namely, a donor, recipient and a blood bank. Our users will be classified into persons (donor and recipient) and organizations (blood-banks). We would record locations of all users and aim to provide the most appropriate solution, taking into account distance. Persons will have their medical history and blood group recorded and made available to aid the efficiency of the system. Organizations will have their activity history recorded for accountability and references.

REQUIREMENT ANALYSIS

An efficient blood bank database would aim to easily connect potential donors with blood-banks, blood-banks with potential recipients and vice versa. The potential donors would be connected with their nearest blood bank. The data of potential donors will be stored according to their blood types, disease history and proximity to nearest blood-bank in the database.

Another Objective of the database is to register and store blood-banks by their location in order for them to be connected to potential donors and recipient with ease. The blood banks aim to receive and give out blood from people in need (**recipients**) closest to them. The blood banks record the instances of blood received according to the blood group received, location and past illnesses associated with donor. The banks also record the instances of blood given out according to similar criteria, replacing past illnesses with reason for requesting blood.

The blood recipient would need to be connected to a bank which possesses his blood in the closest proximity to him. The recipient would provide details of the blood type requested, reason for request and location.

TOOLS/IDES

MySQL Workbench: This tool will be used to create the database tables and it will also be used to create extended entity relation diagram (EERD).

Eclipse: This IDE will be used to connect the database from MySQL to the main program.

Window Builder: This will be used to design the program and test it in a local environment.

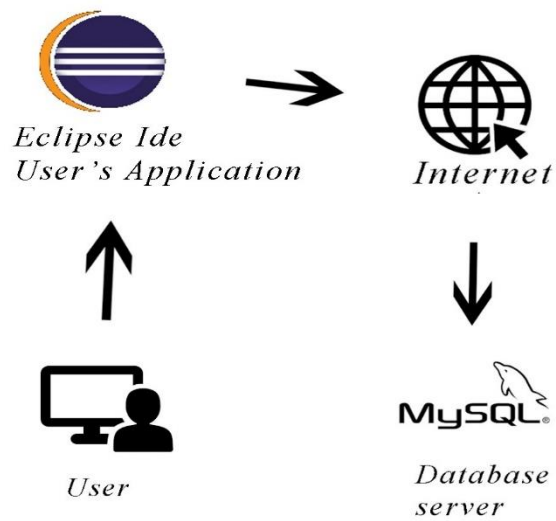


Figure 1: Tools/IDEs

UML USE CASE DIAGRAM

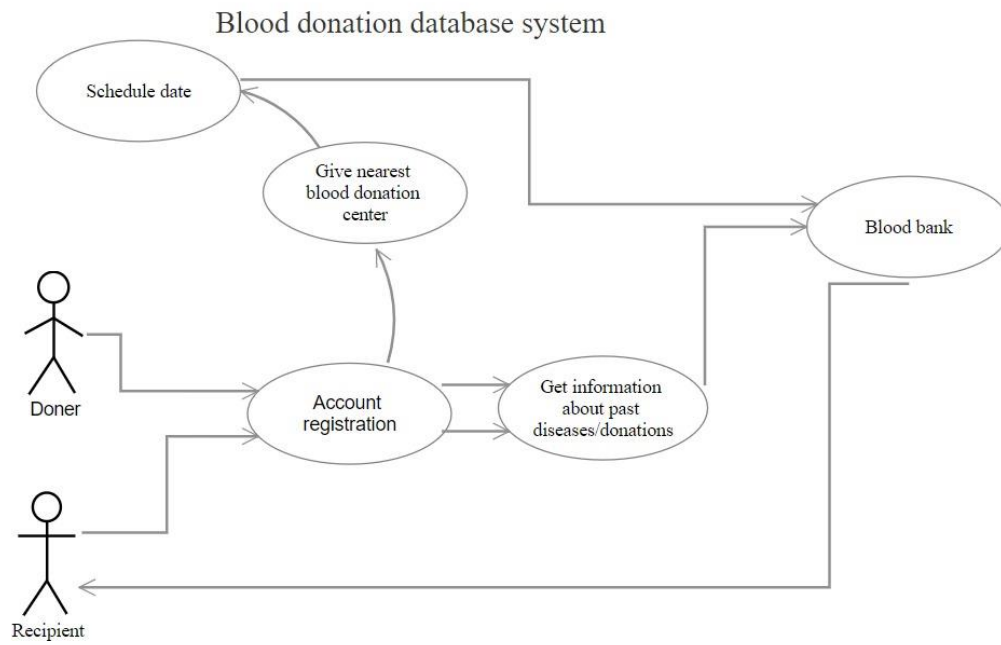


Figure 2: UML diagram

E-R DIAGRAM

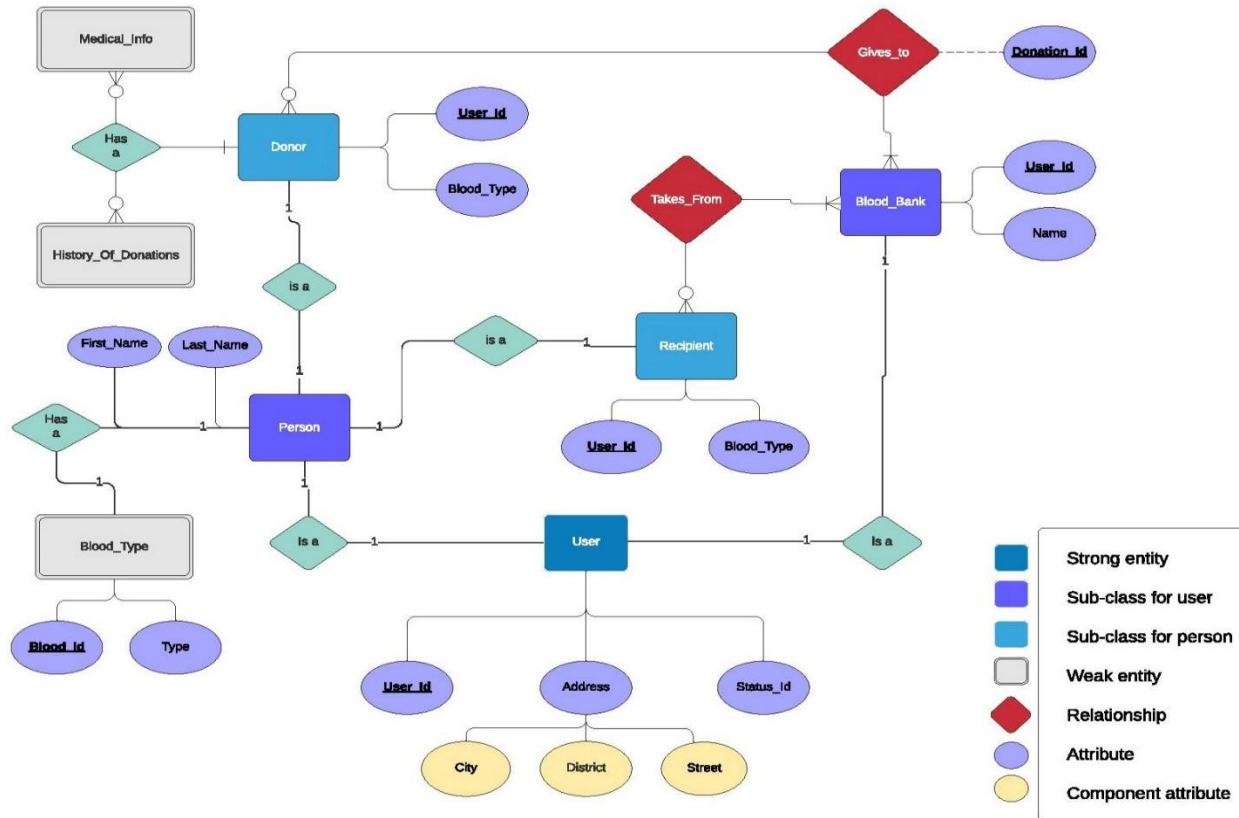


Figure 3: Entity-Relationship diagram

DESIGN PHILOSOPHY

In this part of the report, the ER diagram is going to be explained in detail to enhance the understanding of the system whole. While designing the diagram, we were trying to connect it to the system to be done in at the end. Therefore, it will be explained in this part the reasons behind some steps that may seem unnecessary.

Entities:

1. User
2. Blood_Bank
3. Person
4. Donor
5. Recipient
6. Medical_Info
7. History_Of_Donations
8. Blood_Type

The **User** table is a super class for **Blood_Bank** and **Person**. They share the same basic attributes with **Name** attribute and **First_Name**, **Last_Name** and **Blood_Id** attributes being added to the **Blood_Bank** and **Person** respectively.

Donor and **Recipient** are both sub-classes of **person** that share the same attributes and taking the **User_Id** as their Donor or Recipient Ids from the **User** where Ids will be generated automatically by the system according to the status of the User; Donor, Recipient or Blood bank.

Medical_Info table will be made to control and approve whether the donor will be able to donate blood or not according to his medical case. Therefore, **Medical_Info** table will store the medical reports of the donors and the date of them.

History_Of_Donations table will store the previous donations of donors so that the system will not accept them to donate any blood earlier than **56 days** from the previous donation.

Blood_Type table is made to monitor the blood types with some attributes to help automate the system for donation to make it easier and faster. The attributes will show which type can donate to which and vice versa. **Blood_Id** being the primary key of this table, it will be a critical attribute in the other tables to manage the process of donation and make it simpler.

Cardinalities:

ONE TO ONE

Between User and Blood_Bank

Between User and Person

Between Person and Donor

Between Person and Recipient

Between Person and Blood_Type

ONE TO MANY

Between Donor and Medical_Info

Between Donor and History_Of_Donation

MANY TO MANY

Between Donor and Blood_Bank

Between Recipient and Blood_Bank

User Permissions:

Donor:

Add: User, Blood Type, Medical Info

Delete: User, Blood Type, Medical Info

Update: User, Blood Type, Medical Info

View: User, Blood Type, Medical Info, Blood Bank

Recipient:

Add: User, Blood Type

Delete: User, Blood Type

Update: User, Blood Type

View: User, Blood Type, Blood Bank

Blood Bank:

Add: History of donations

Delete: History of donations,

Update: History of donations

View: History of donations, Donor, Recipient

PHASE 2

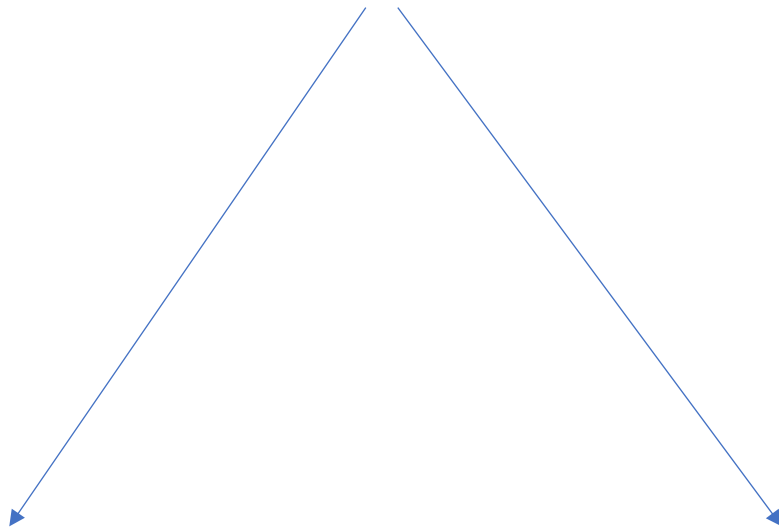
NORMALIZATION

Normalization is an approach that should be done for databases to minimize their redundancy and increase their efficiency. It consists of several steps and has many forms starting from 1NF (First Normal Form) which has basic and logical requirements such as not having two columns having the same name. Steps of normalization continue with one main goal to achieve; Minimize redundancies.

Figure 1 below is an example of a step that was taken to achieve the goal. It was noticed while inserting data that different users can have the **same address**. Which will cause some redundancy as the same address is repeated for different users. Not to mention the possibility that a name of a district or a neighborhood getting changed. In that case, a lot of time will be wasted editing the data of the users from that district or area. Therefore, as a solution for this redundancy, an **Address** table was created to save the addresses and defining an **Address_Id** as the primary key of the table in order to connect the addresses to a unique Id. After that, **Address_Id** was added as attribute in the user table as a foreign key connecting the user to their address. Now in this case, adding, deleting, and changing data is made **easier and time-saving**.

Furthermore, some functional dependencies were spotted between some attributes which are not primary keys. An entity was previously defined as History_Of_Donations to collect the data for donations that took place by the donors. However, it was noticed later that the attributes of the entity are dependent and it was decided to remove this table as its job can be done by some queries natural joining Donor entity with Gives_To relation.

User_Id	Status_Id	City	District	Neighborhood	Phone_No
1	3	Kayseri	Melikgazi	Alparslan	96565265
2	2	Kayseri	Melikgazi	AYDINLIKEVLER	63168498
3	3	Kayseri	Kocasinan	Erkilet	32489465
4	1	Kayseri	Talas	Anayurt	75468545
5	1	Kayseri	Melikgazi	Alparslan	86546848



User_Id	Status_Id	Address_Id	Phone_No
1	3	2	96565265
2	2	3	63168498
3	3	1	32489465
4	1	4	75468545
5	1	2	86546848

Address_Id	City	District	Neighborhood
1	Kayseri	Kocasinan	Erkilet
2	Kayseri	Melikgazi	Alparslan
3	Kayseri	Melikgazi	AYDINLIKEVLER
4	Kayseri	Talas	Anayurt
⋮	⋮	⋮	⋮

Figure 4: Example of normalization step

E-R TO RELATIONAL MAPPING

1. NORMAL ENTITIES

Address (Address_Id , City , District , Neighborhood)

Status (Status_Id , Status)

User (User_Id , Status_Id , Address_Id , Phone_No)

Blood_Type (Blood_Id , Blood_Code , Donates_to , Receives_from)

Donor (Donor_Id , First_Name , Last_Name , Blood_Id)

Recipient (Recipient_Id , First_Name , Last_Name , Blood_Id)

Blood_bank (Bank_Id , Name , Capacity)

2. WEAK ENTITIES

Medical_Info (Report_Id , Donor_Id , Date , Result)

3. RELATIONSHIPS

Gives_to (Donation_Id , Donor_Id , Bank_Id , Date , Amount)

Takes_from (Transfer_Id , Recipient_Id , Bank_Id , Date , Amount)

DATABASE SCHEMA

1. **ADDRESS:** This table where the main addresses are saved so that only the Address_Id is used in the user table to avoid redundancy.

	Address_Id	City	District	Neighborhood
TYPE	numeric	varchar(20)	varchar(20)	varchar(60)
KEY	PK			
EXAMPLE	1000000	Kayseri	Kocasinan	Mithatpaşa

Table 1: ADDRESS TABLE

2. **STATUS:** This table defines the users and categorizes them into the main 3 categories; blood donor, blood recipient and blood bank.

	Status_Id	Status
TYPE	numeric	varchar(20)
KEY	PK	
EXAMPLE	2	Donor

Table 2: STATUS TABLE

3. USER: This table saves the user's personal data in the database.

	User_Id	Status_Id	Address_Id	Phone_No	Password
TYPE	numeric	numeric	numeric	Varchar(10)	Varchar(45)
KEY	PK	FK	FK		
EXAMPLE	4	2	1000000	5539190967	123

Table 3: USER TABLE

4. BLOOD_TYPE: This table defines and saves all the blood groups with their donation and receipt features.

	Blood_Id	Blood_Code	Donates_to	Receives_from
TYPE	numeric	Varchar(5)	Varchar(45)	Varchar(45)
KEY	PK			
EXAMPLE	4	O	All	O

Table 4: BLOOD_GROUP TABLE

5. DONOR: This table saves the data of donors who registered to the system.

	Donor_Id	First_Name	Last_Name	Blood_Id
TYPE	numeric	Varchar(5)	Varchar(45)	Varchar(45)
KEY	PK,FK			FK
EXAMPLE	4	Ahmed	Alqershi	4

Table 5: DONOR TABLE

6. MEDICAL_INFO: This table saves the medical reports that state whether donors can donate blood or not.

	Report_Id	Donor_Id	Date	Result
TYPE	numeric	numeric	date	Varchar(45)
KEY	PK	FK		
EXAMPLE	150100	4	2020-05-02	No disease

Table 6: MEDICAL_INFO TABLE

7. RECIPIENT: This table saves the data of those recipients who registered to the system.

	Recipient_Id	First_Name	Last_Name	Blood_Id
TYPE	numeric	Varchar(5)	Varchar(45)	Varchar(45)
KEY	PK,FK			FK
EXAMPLE	3	Lekan	Aremu	1

Table 7: RECIPIENT TABLE

8. BLOOD_BANK: This table saves the data of blood banks which are registered to the system.

	Bank_Id	Name	Capacity
TYPE	numeric	Varchar(5)	numeric
KEY	PK,FK		
EXAMPLE	1	AGUBB	1000

Table 8: BLOOD_BANK TABLE

9. GIVES_TO: This table saves the blood donations information of donors in blood banks.

	Donation_Id	Donor_Id	Bank_Id	Date	Amount
TYPE	numeric	numeric	numeric	date	numeric
KEY	PK	FK	FK		
EXAMPLE	990001	4	1	2/19/2020	700

Table 9: GIVES_TO TABLE

10. TAKES_FROM: This table saves the blood donations information of donors in blood banks.

	Transfer_Id	Recipient_Id	Bank_Id	Date	Amount
TYPE	numeric	numeric	numeric	date	numeric
KEY	PK	FK	FK		
EXAMPLE	770001	3	1	3/12/2020	450

Table 10: TAKES_FROM TABLE

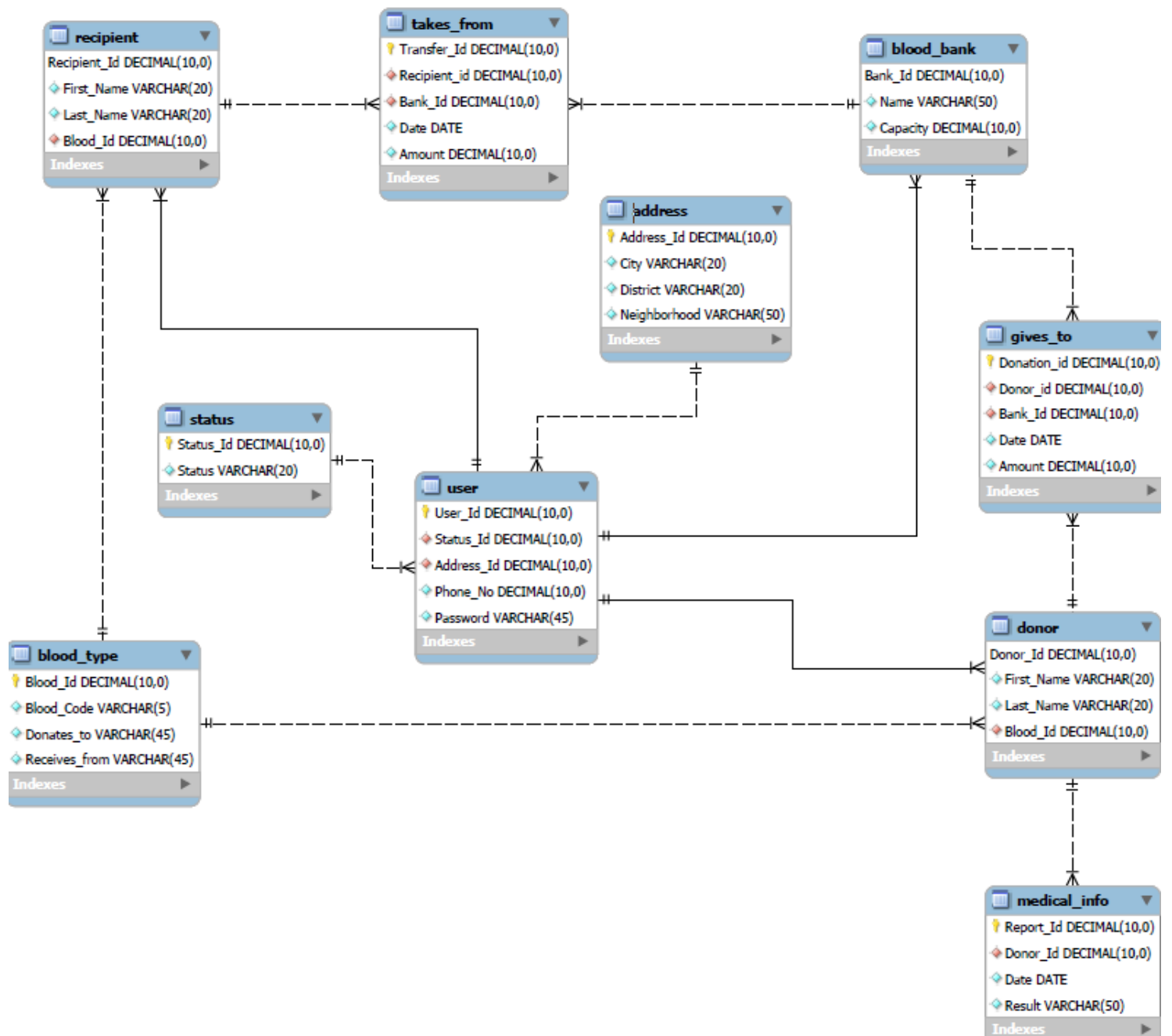


Figure 5: DATABASE SCHEMA FROM MYSQL

SCRIPTS

DDLs:

CREATE TABLE `Address` (

`Address_Id` numeric NOT NULL,

`City` varchar (20) NOT NULL,

`District` varchar (20) NOT NULL,

`Neighborhood` varchar (60) NOT NULL,

PRIMARY KEY (`Address_Id`)

);

CREATE TABLE `Status` (

`Status_Id` numeric NOT NULL,

`Status` varchar (20) NOT NULL,

PRIMARY KEY (`Status_Id`)

);

CREATE TABLE `User` (

`User_Id` numeric NOT NULL,

`Status_Id` numeric NOT NULL,

`Address_Id` numeric NOT NULL,

`Phone_No` varchar(10) NOT NULL,

```

`Password` varchar(45) NOT NULL,

PRIMARY KEY (`User_Id`),

FOREIGN KEY (`Status_Id`)

REFERENCES `blood_donation_database`.`Status` (`Status_Id`),

FOREIGN KEY (`Address_Id`)

REFERENCES `blood_donation_database`.`Address` (`Address_Id`)

);

```

CREATE TABLE `Blood_Type` (

```

`Blood_Id` numeric NOT NULL,

`Blood_Code` varchar (5) NOT NULL,

`Donates_to` varchar (45) NOT NULL,

`Receives_from` varchar (45) NOT NULL,

PRIMARY KEY (`Blood_Id`)

);

```

CREATE TABLE `Donor` (

```

`Donor_Id` numeric NOT NULL,

`First_Name` varchar (20) NOT NULL,

`Last_Name` varchar (20) NOT NULL,

`Blood_Id` numeric NOT NULL,

PRIMARY KEY (`Donor_Id`),

```



```
FOREIGN KEY (`Donor_Id`)

REFERENCES `blood_donation_database`.`user` (`User_Id`),

FOREIGN KEY (`Blood_Id`)

REFERENCES `blood_donation_database`.`Blood_Type` (`Blood_Id`)

);
```

```
CREATE TABLE `Medical_Info` (
```

```
  `Report_Id` numeric NOT NULL,

  `Donor_Id` numeric NOT NULL,

  `Date` date NOT NULL,

  `Result` varchar (50) NOT NULL,

  PRIMARY KEY (`Report_Id`),

  FOREIGN KEY (`Donor_Id`)

  REFERENCES `blood_donation_database`.`Donor` (`Donor_Id`)

);
```

```
CREATE TABLE `Recipient` (
```

```
  `Recipient_Id` numeric NOT NULL,

  `First_Name` varchar (20) NOT NULL,

  `Last_Name` varchar (20) NOT NULL,

  `Blood_Id` numeric NOT NULL,

  PRIMARY KEY (`Recipient_Id`),
```

```
FOREIGN KEY (`Recipient_Id`)

REFERENCES `blood_donation_database`.`user` (`User_Id`),

FOREIGN KEY (`Blood_Id`)

REFERENCES `blood_donation_database`.`Blood_Type` (`Blood_Id`)

);
```

```
CREATE TABLE `Blood_Bank` (
```

```
  `Bank_Id` numeric NOT NULL,

  `Name` varchar (50) NOT NULL,

  `Capacity` numeric NOT NULL,

  PRIMARY KEY (`Bank_Id`),

  FOREIGN KEY (`Bank_Id`)

  REFERENCES `blood_donation_database`.`user` (`User_Id`)

);
```

```
CREATE TABLE `Gives_to` (
```

```
  `Donation_id` numeric NOT NULL,

  `Donor_id` numeric NOT NULL,

  `Bank_Id` numeric NOT NULL,

  `Date` date NOT NULL,

  `Amount` numeric NOT NULL,

  PRIMARY KEY (`Donation_id`),
```

```

FOREIGN KEY (`Donor_Id`)

REFERENCES `blood_donation_database`.`Donor` (`Donor_Id`),

FOREIGN KEY (`Bank_Id`)

REFERENCES `blood_donation_database`.`Blood_Bank` (`Bank_Id`)

);

CREATE TABLE `Takes_From` (

  `Transfer_Id` numeric NOT NULL,

  `Recipient_id` numeric NOT NULL,

  `Bank_Id` numeric NOT NULL,

  `Date` date NOT NULL,

  `Amount` numeric NOT NULL,

  PRIMARY KEY (`Transfer_Id`),

  FOREIGN KEY (`Recipient_Id`)

REFERENCES `blood_donation_database`.`Recipient` (`Recipient_Id`),

  FOREIGN KEY (`Bank_Id`)

REFERENCES `blood_donation_database`.`Blood_Bank` (`Bank_Id`)

);

```

DMLs:

INSERT INTO **ADDRESS** VALUES (1000000 , 'Kayseri' , 'Kocasinan' , 'Mithatpasa');

INSERT INTO **ADDRESS** VALUES (1000001 , 'Kayseri' , 'Melikgazi' , 'Cumhuriyet Meydani');

INSERT INTO **ADDRESS** VALUES (1000002 , 'Kayseri' , 'Talas' , 'Anayurt');

INSERT INTO **STATUS** VALUES (1 , 'Blood Bank');

INSERT INTO **STATUS** VALUES (2 , 'Donor');

INSERT INTO **STATUS** VALUES (3 , 'Recipient');

INSERT INTO **USER** VALUES (1 , 1 , 1000002 , 5539190967);

INSERT INTO **USER** VALUES (2 , 1 , 1000001 , 558458625);

INSERT INTO **USER** VALUES (3 , 3 , 1000001 , 5582579655);

INSERT INTO **USER** VALUES (4 , 2 , 1000000 , 5539147827);

INSERT INTO **USER** VALUES (5 , 3 , 1000000 , 5539190954);

INSERT INTO **BLOOD_TYPE** VALUES (1 , 'AB' , 'AB' , 'ALL');

INSERT INTO **BLOOD_TYPE** VALUES (2 , 'A' , 'A & AB' , 'A & O');

INSERT INTO **BLOOD_TYPE** VALUES (3 , 'B' , 'B & AB' , 'B & O');

INSERT INTO **BLOOD_TYPE** VALUES (4 , 'O' , 'ALL' , 'O');

INSERT INTO **DONOR** VALUES (4 , 'Ahmed' , 'Alqershi' , 4);

INSERT INTO **RECIPIENT** VALUES (3 , 'Lekan' , 'Aremu' , 1);

INSERT INTO **RECIPIENT** VALUES (5 , 'Mohammed' , 'Shughri' , 3);

INSERT INTO **BLOOD_BANK** VALUES (1 , 'Melikgazi' , 1000000);

INSERT INTO **MEDICAL_INFO** VALUES (150100 , 4 , '2020-05-02' , 'No disease');

INSERT INTO **GIVES_TO** VALUES (990001 , 4 , 1 , '2020-03-12' , 350);

INSERT INTO **TAKES_FROM** VALUES (770001 , 5 , 1 , '2020-02-19' , 700);

VIEWS

Below are the views for our database system. Our system will have three main users, the donors, the recipient and the blood banks. These users have different needs and as a result would need to access different data but be unable to access others. These cases are listed below;

1. **Find_all_universal_donors** – This is a view to be used by the blood banks to receive a list of all universal donors to enable them serve potential recipients much better. Below is the SQL query for that view.

Create view find_all_universal_donors as select Blood_code, donor.First_Name, donor.Last_Name **from** Blood_Type, donor **where** Blood_Type.Blood_Id = donor.Blood_Id and Blood_code = 'O'

2. **Find_all_universal_receivers** - This is also a view to be used by blood banks to retrieve a list of all universal receivers. The SQL query is given below

Create view find_all_universal_receivers as select Blood_code, recipient.First_Name, recipient.Last_Name **from** Blood_Type, recipient **where** Blood_Type.Blood_Id = recipient.Blood_Id and Blood_code = 'AB'

3. **Find_all_receiver_contact_info** – This view retrieves all receiver contact information without displaying sensitive information and would be used by blood banks. The SQL query is below

Create view find_all_receiver_contact_info as select distinct recipient.First_Name, recipient.Last_Name, User.phone_no, address.city **from** user, recipient, address **where** user.user_Id = recipient.recipient_Id **or** user.user_id = address.Address_Id

4. **Final_all_donor_contact_info** – This view retrieves all donor contact info. SQL Query below -

Create view final_all_donor_contact_info as select distinct donor.First_Name, donor.Last_Name, User.phone_no, address.city **from** user, donor, address **where** user.user_Id = donor.donor_id **or** user.user_id = address.Address_Id

5. **Find_all_blood_bank_contact_info** – This view is to be used by donors and recipients. It shows all the blood banks and their location to enable ease of access. Below is the SQL Query.

Create view find_all_blood_bank_contact_info as select distinct Blood_bank.Name, User.phone_no, address.city from user, blood_bank, address where user.user_Id = blood_bank.bank_id or user.user_id = address.Address_Id

DATABASE APPLICATION

The software which we used to demonstrate the database was developed in Java, accompanied with the Java Database Connectivity (JDBC) for connection to the database and Swing for the graphical user interface (GUI). An external Jar file called 'rs2xml' for populating Jtables was used as well.

When the program is first run, the start page is displayed and it can be used to sign in or sign up. If the user chooses to sign up, they would need to select what kind of user they would be signing up as from our three choices, which are; the donor, recipient or a blood-bank and then click on the sign up button after which they will be transferred to the relevant sign up pages. Alternatively, if the user is registered into the system, they can sign in by giving their user id and their password.

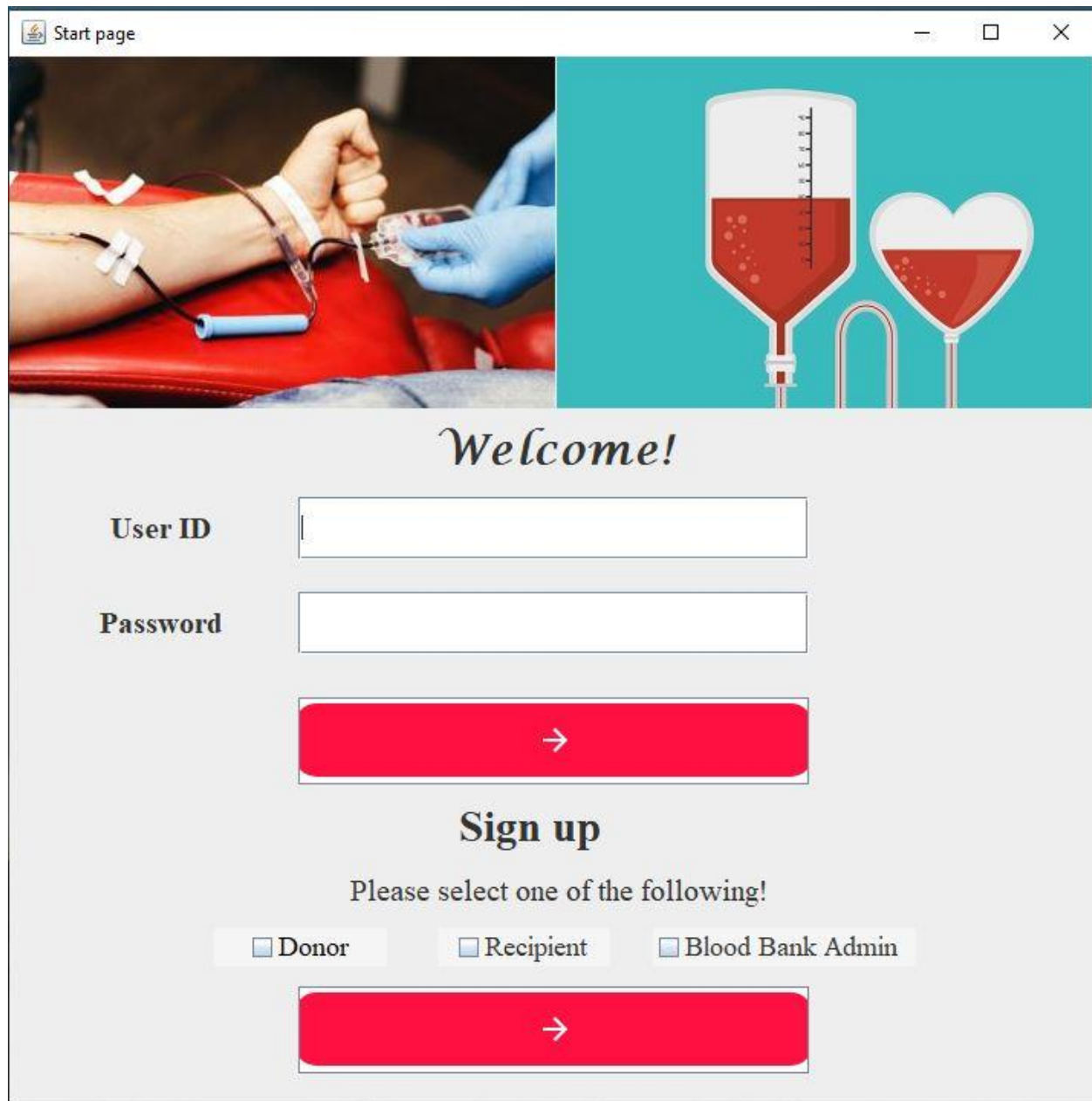
On the Donor or Recipient sign up page, they can enter their basic info and select their address and blood type from a combo-box with values populated from the Address and Blood Type tables of the database. After the user enters all their valid information, the user clicks the sign-up button and are given an id with which they use to sign in. Donors are given an id that begins with 200, recipients are given an id that begins with 300 and blood banks are given an id that begins with 100. The blood-bank sign-up page is similar to this, with some minor changes to accommodate the data with which a blood bank needs to provide according to our database design philosophy.

After the user signs up, they can now sign in. If they are a donor or recipient, they will be redirected to the dashboard where all the blood banks in the system are displayed to them. They can click on the 'next' button to see more blood-banks in the system if they are available and they can click on the 'contact us' button to see the full information of the blood-bank. The update button sends the

user to the update page when clicked and then the user can update their data in the system. There is also a log-out button present at the top right of the page.

On the other hand, when a blood-bank user signs in, they are taken to the admin page where all the donors and recipients in the system are displayed on a table. The blood-bank user can click on the 'profile' button to see their information as well.

SCREENSHOTS AND DATASETS



The screenshot shows a web browser window titled "Start page". The header features two images: a photograph of a person's arm with a blood donation needle and a graphic of a blood drip chamber and a heart. Below the header, the text "Welcome!" is displayed in a cursive font. The login section includes labels for "User ID" and "Password", each followed by a text input field. A red button with a right-pointing arrow is positioned below the password field. The "Sign up" section follows, with the text "Please select one of the following!". There are three radio button options: "Donor", "Recipient", and "Blood Bank Admin". A second red button with a right-pointing arrow is located at the bottom of the sign-up section.

Start page

Welcome!

User ID

Password

→


Sign up

Please select one of the following!

☐ Donor ☐ Recipient ☐ Blood Bank Admin

→

Figure 6: START PAGE



Name

Surname

Password

Confirm Password

City

Kayseri

▼

District

Kocasinan

▼

Neighborhood

Barbaros mahallesi

▼

Blood type

AB

▼

Phone number




Figure 7: DONOR/RECIPIENT SIGN UP PAGE

Admin Signup

←

Blood Bank Name

Password **Confirm Password**

Capacity **Phone Number**

City **District** **Neighborhood**

→

Figure 8: BLOOD BANK SIGN UP PAGE

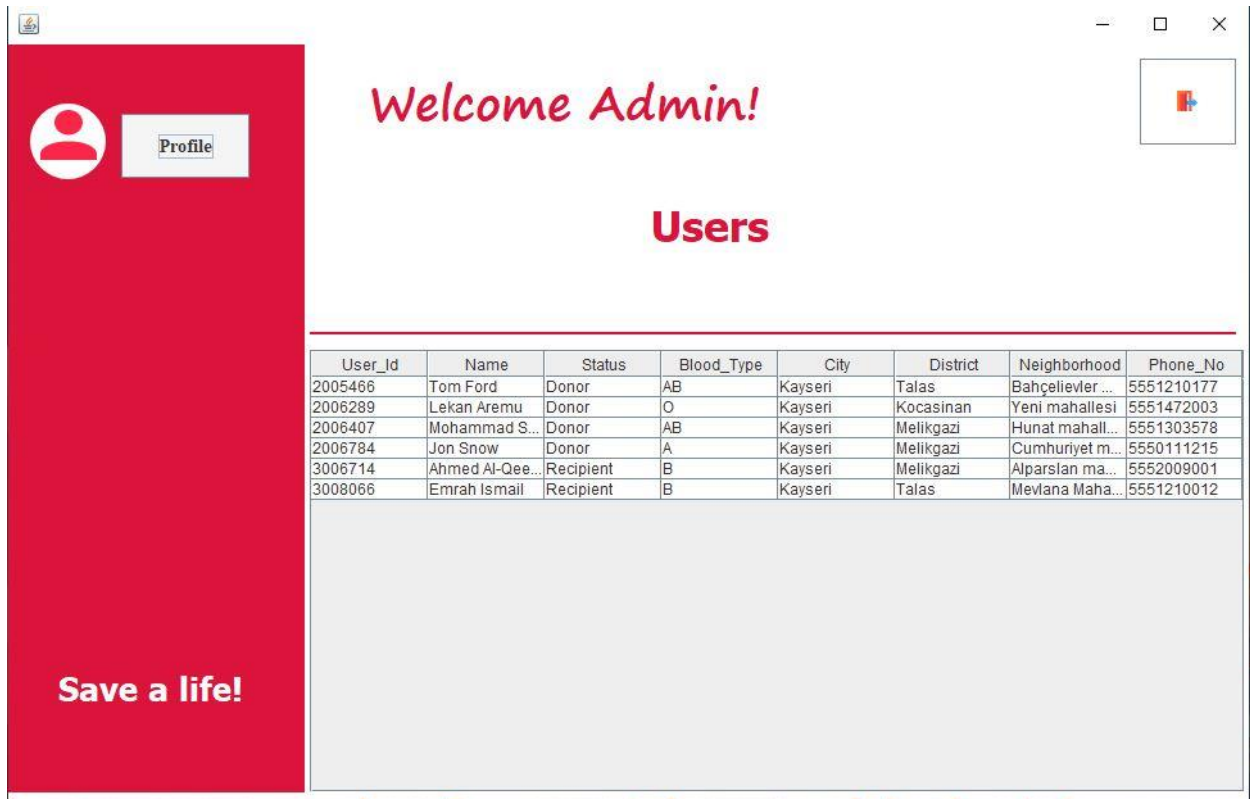


Figure 9: BLOOD BANK ADMIN PAGE

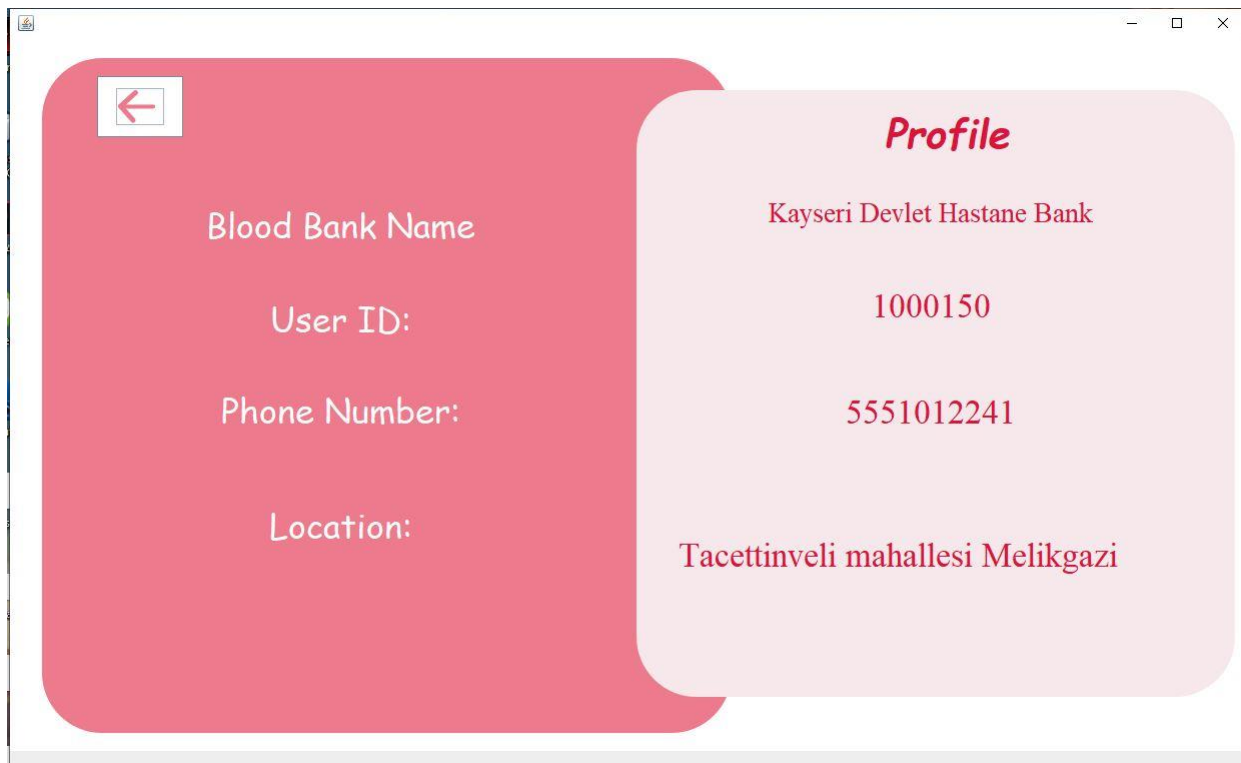


Figure 10: BLOOD BANK PROFILE PAGE



Figure 11: LIST OF DONATIONS CENTRES

The image shows a web browser window titled "Update Page". The browser's address bar is empty. The page has a red header bar with a white back arrow icon on the left. Below the header, the form is displayed on a light gray background. It consists of five input fields arranged in three rows: "Name" and "Surname" in the first row, "Password" and "Confirm Password" in the second row, and "Phone number" in the third row. A red button with a white right arrow icon is centered below the "Phone number" field. The page ends with a solid red footer bar.

Update Page

←

Name

Surname

Password

Confirm Password

Phone number

→

Figure 12: UPDATING DONATION CENTRE INFORMATION



Figure 13: DETAILES OF THE DONATION CENTRES