HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Information Systems Department

Prepared by:

[Abdelrahman Mohamed Abdelazim (20170306)]

[Abdelrahman Gamal Fathy (20180311)]

[Fatma Ahmed Mohamed (20180416)]

[Ahmed Mustafa Mahmoud (20180731)]

[Omar Mohamed Hany (20150362)]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Information Systems Department, the Faculty ofComputers & Artificial Intelligence, Helwan University

Supervised by:

[Dr. Mohammed Marie]

January 2023

BLOOD BANK

Faculty of Computer and Artificial Intelligence
Helwan University.
Information System Department.

Supervised By:

Dr. Mohammed Marie

Team Members:

Abdelrahman Mohamed

Abdelrahman Gamal

Fatma Ahmed

Ahmed Mustafa

Omar Mohamed

Contents

Acknowledgements

The success of any project depends largely on the encouragement and guidelines of many others. At the first we thank ALLAH for giving us the power to achieve this work, and for the blessings from ALLAH to all of us.

we would like to express our heartfelt gratitude. gratitude and great joy at the opportunity to work with someone as dedicated as ***DR. Mohammed Marie*** *for her help, excellent advice, and reassurance throughout our graduation project.*

We would also like to thank our families, particularly our parents. parents, for their support, patience, and understanding over the years of assistance We will be eternally grateful to our parents, who have always said a prayer for us and propelled us to success.

Finally, anks to our faculty for providing the appropriate environment that drove us to be the best Consider Helwan's computer science graduates. University is supposed to stand for something.

THANK YOU

Chapter 1: Introduction

1.1 Overview:

The idea of the project is to facilitate the interaction between the user and the hospital in the part related to donation and purchase of blood bags This is by organizing the stock of blood bags in hospitals, so the user easily reaches the hospital and requests a blood bag Or enter the same hospital and make a request to donate this bag In this case it is a priority when dispensing with the rest of the users.

1.2 Objectives:

The application aims to solve a critical problem which is the lack and the difficulty in searching for blood type.

The main targets of blood bank application are:

1. Facilitate communication between user and hospital for donation or blood purchase.
2. Save money because the app is totally free.
3. It saves the user time because they're able to get the blood type to what they need, and make an order for a blood donation through the house without wasting time.
4. be available to service a large number of users at any time and from any location
5. produce adequate supplies of blood to address the issue of blood scarcity

1.3 Purpose:

The purpose is to provide time for the user and facilitate communication between the user and the hospital for purchase or donation.

1.4 Scope:

The application is supporting and helping

1- Admin who can Manage hospital and users in the system
2- User who need donation with blood by register on the application and create profile.
3- User who need purchasing with blood by register on the application and create profile.

1.5 General constraints:

They are factors that can impact quality, delivery, and overall project success and there are 6 factors like quality and time and cost and scope and benefits and risks.

1- <u>Developed process and team constraint:</u>

-Which is the time we will take to estimate the project Which are at least 5 months

2- <u>environmental and technology constraints</u>

-users must be connected to internet while registering.

-each user must have username and password.

-users cannot see the personal information of each other.

-the language of the app will be only English

2- software constraints

-we will use .net

-Availability on visual studio code

4- Hardware constraints

- android or IOS smartphone, tablet or an IOS device with an internet coverage.

5- delivery and deployment constraints

-We need a budget in order to launch the application on the play store and for marketing it to end users.

Chapter 2: Project "Planning and analysis"

2.1 Project Planning

2.1.1 Feasibility Study

Problem Statement :

the need for blood donation is increasing, many people face a lot of difficulties in searching for blood type in emergency and critical situations. because of the low quantities of blood, the highly demand and the hospitals and blood banks can't cover this highly demand. The system provides easy access to blood type and easy communication between the donor and the acceptors, and presents an alert system to the donor about requirement of their blood to a person need to meet the challenging requirement to efficiently collect blood during emergency and achieve communication between donors and acceptors, it will help us to find the blood group with its most time.

Project importance :

Cost: The application save cost because the application is completely free, the user can find a donor for his blood type easily without pay any cost.

Accessibility: Users will be able to access the application anytime and there will be no time constraints contrary to what usually happens in hospitals.

Communication: The application provides an easy way of communication between the donors the receivers

User friendly: Blood donation application is easy to use as any user can use and understand it easily.

Time: Save users time instead of spending a lot of time waiting their blood types be available in hospitals.

Awareness: spread awareness about the value of blood donation and its positive effect about health, which let to increasing the volunteering work among people and encourage them to help each us.

progress: the application participates in progress and improvement the medical services emergency: able to serve many users in emergency and critical situations

privacy and safety: the user can communicate with the donors in private and trustable way

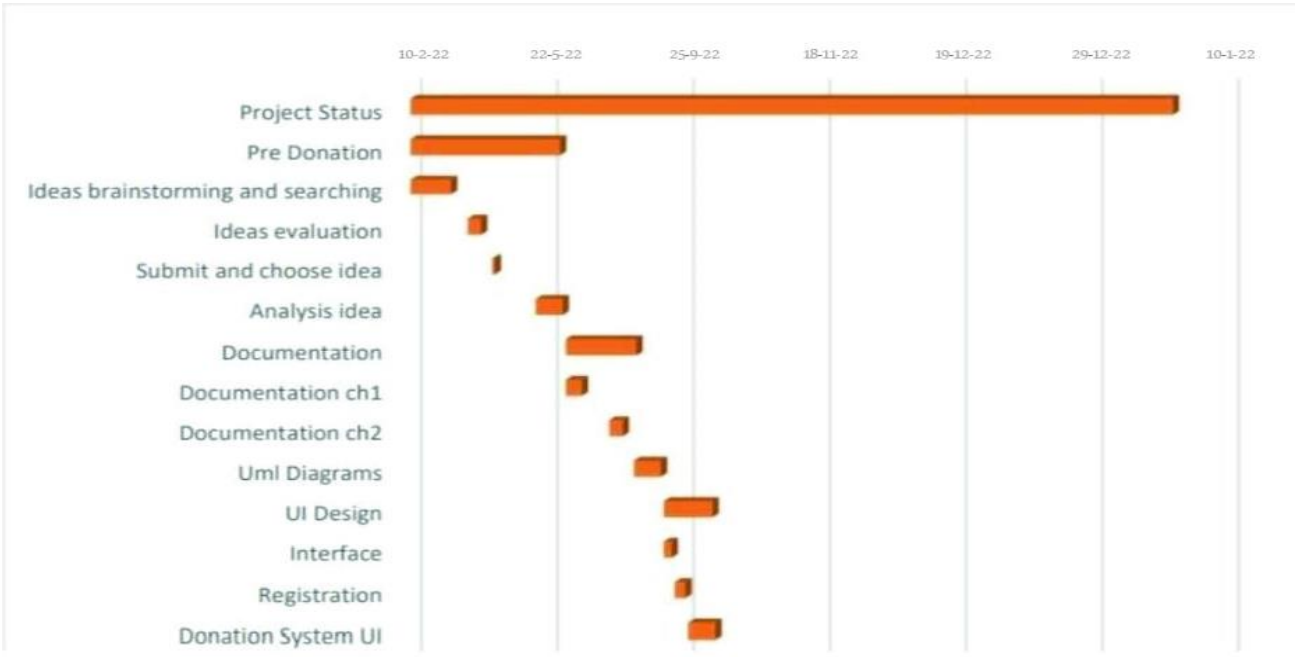
Reason for selecting this project:

This project acts as an important role in saving life of human beings and which is also its main aim. The project Android Blood donation system is developed so that users can view the information about registered blood donors such as name, donation status and other such personal information along with their details of blood -achieve the successful communication between users and blood donors -it can efficiently collect blood during emergency in limited time -The health and safety of the donor as well as the recipient must be safeguarded -Only individuals in good health should be accepted as donors of whole blood - The main aim of developing this application is to reduce the time to a great extent that is spent in searching for the right donor and the availability of blood required.

2.1.2 Gantt Chart

task	start	end	duration
Project Status	10/2/2022	7/1/2023	331
Pre-Donation	10/2/2022	12/4/2022	61
Ideas brainstorming and searching	12/4/2022	11/5/2022	29
Idea's evaluation	11/5/2022	20/5/2022	9
Submit and choose idea	23/5/2022	24/5/2022	1
Analysis idea	25/5/2022	12/6/2022	18
Documentation	12/6/2022	20/8/2022	69
Documentation ch1	21/8/2022	30/8/2022	9
Documentation ch2	1/9/2022	20/9/2022	19
UML Diagrams	1/10/2022	5/11/2022	35
UI Design	8/11/2022	20/11/2022	12
Interface	20/11/2022	28/11/2022	8
Registration	1/12/2022	28/12/2022	27
Donation System UI	28/12/2022	7/1/2023	10

Figure 1 "Gantt chart"



2.2 Analysis and Limitation of existing system:

1- Difficulty getting data out of the database :

It is quite difficult to keep track of the many fields, such as blood groups, members, latest blood donated, etc., when information is input manually. Now, maintaining the database is complex and there are many tasks to complete, which makes it error-prone and causes results to be delayed.

2-Time :

If a patient needs blood, the hospital first determines whether it is on hand; if not, they look through their records to find a donor. This procedure takes more time and is time-consuming.

3-leads to error prone results:

the systems are in-efficient to handle multiple requests at the same time and leads to error results.

4-Lack of data security:

The database contains no security for any of the data. 5- database not updated: There are a number of alternative websites designed specifically for blood donation. These platforms are ineffective since they don't routinely update the donor information.

2.3 Need for the new system:

In the new system, users can post in the event that a specific blood type is required to save time and save lives, and other users can interact with the post by giving it a like or leaving a comment. The post's author can then call or chat with those who have interacted with it. - In the new system, a user can look for people in their area who have the same blood type as them and communicate with them. -By responding to a few questions in the new system, the user can determine his donation status, and the system will notify him right away.

2.4 Analysis of the new system:

2.4.1 User Requirements

1. Requirements: - The user can search for the nearest donor and donate blood.

2- Desirable conditions

-User can look for a certain blood type

. User can make a request to find a donor; by responding to some questions, user can determine whether donor donation is likely to occur.

2.4.2 System Requirements

what hardware and software requirements does your computer have?

Operating System 1

The code for our app will run on visual studio code because it is built in. net.

Storage capacity

,our processor speed (GHz), memory (RAM), hard disc space (MB), and

Internet access

We require strong Wi-fi.

2.4.3 Domain Restrictions

• The database server should have enough memory to support any number of users and events they may have.

• Before making the update in the database, the program me must validate all values.

• The application needs to support updates for upcoming accessories and models.

• The database should periodically be backed up in case the original becomes corrupt.

2.4.4 Functional Requirement:

- Register

actor	User, Hospital
Input	Name, Location, Age, Email, Number phone, Blood Type, Password, and Confirm Password.
output	The Person has an account in the system.
Description	function enables you to create a system account.
Expected-Risk	Sign up with incorrect information and don't email you the code.
Precondition	Install the app on your device, launch it, and select "Sign up."
Postcondition	Sign up successfully
steps	Application downloads by user. 2 User launches the application. 3. Enter your name, age, blood type, email, phone number, password, and password confirmation. To obtain the user's current location, click on the icon's location. 5-Click the "Sign up" button. 6-System emails you a code. 7-Type the code that was supplied to you. 8 – Account creation.

- Login

actor	User, Hospital, Admin
input	Email Address and Password.
output	Log in of system.
Description	Function allows admin to login of system.
Expected-Risk	Login with wrong data.
Precondition	Admin already login in the system.
Postcondition	Admin login successfully.
steps	1-Enter right Email Address and Password. 2-Click on “Login” 3-Login to the system successfully.

➢ Change password

actor	User, Hospital
input	Enter old password New password and Confirm password.
output	Change current password.
Description	Function allows you to change password.
Expected-Risk	Password and confirm password not matched – Do not fill this data.
Precondition	User already login in the system.
Postcondition	User change password successfully.
steps	After opening this application: 1-Enter Drawer. 2-Enter “change password”. 3-Write a new password and Confirm password. 4-password is changed successfully.

➢ log out

actor	User, Hospital, Admin
input	-
output	Log out of system
Description	Function allows you to get out of system
Expected-Risk	-
Precondition	User already login in the system
Postcondition	User logout successfully
steps	1-Login in the system. 2-Enter Drawer. 3-Click in button “Logout”. 4-Logout of system successfully

➢ Add blood bag

actor	Admin
input	-
output	Add blood bag
Description	Function allows you to add blood bag
Expected-Risk	-
Precondition	User already login in the system as admin
Postcondition	Blood bag added successfully
steps	1-Login in the system. 2-Enter Details. 3-Click in button “ADD” 4-added successfully

➢ View balance

actor	Admin
input	-
output	view balance
Description	Function allows you to view balance
Expected-Risk	-
Precondition	User already login in the system as admin
Postcondition	balance view successfully
steps	1-Login in the system. 2-Enter Details. 3-Click in button “veiw balance”

➢ Add New Hospital

actor	Admin
input	-
output	Add new hospital
Description	Function allows you to add new hospital
Expected-Risk	-
Precondition	User already login in the system as admin
Postcondition	new hospital added successfully
steps	1-Login in the system. 2-Enter hospital Details. 3-Click in button “ADD” 4-added successfully

➢ View user profile

actor	User
input	Data retrieved from data base
output	Show any user’s profile
Description	Function allows you to view user’s profiles
Expected-Risk	-
Precondition	User already login and do Search function.
Postcondition	User can view user’s profile picture and user’s information.
steps	1-After user search for any blood type and location and show result 2-user can click on any user who show for him in result. 3-Show user’s Profile successfully.

➢ Search

actor	User
input	Blood Type and Location.
output	Users whom the same blood type and the same location.
Description	Function allows you to search for donor
Expected-Risk	-
Precondition	User already login.
Postcondition	User can search for donor.
steps	After opening this application: 1-Enter Drawer. 2-Click on Search. 3-Enter to Search page. 4-Choose blood type, and click on icon location to get current location. 5-Click on Search. 6-Search by blood type and location successfully and show result for this search

➢ Check cart

actor	User
input	-
output	Check expire, place and price
Description	Function allows you to check cart
Expected-Risk	-
Precondition	User already login in the system as user
Postcondition	Cart checked successfully
steps	1-Login in the system. 3-Click in button “cart” 4- check details

➢ Remove blood bags from your cart

actor	User
input	-
output	Delete blood bag
Description	Function allows you to Remove blood bags from your cart
Expected-Risk	-
Precondition	User already login in the system as user
Postcondition	Cart deleted successfully
steps	1-Login in the system. 3-Click in button “cart” 4- Click in button “delete”

➢ Confirm cart

actor	User
input	-
output	confirm blood bag
Description	Function allows you to confirm cart
Expected-Risk	-
Precondition	User already login in the system as user
Postcondition	Cart Submited successfully
steps	1-Login in the system. 3-Click in button “cart” 4- Click in button “confirm”

➢ Add new balance

actor	Hospital
input	-
output	Add balance
Description	Function allows you to add balance

Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	balance added successfully
steps	1-Login in the system. 2-Enter Details. 3-Click in button “ADD” 4-added successfully

➢ View balance details

actor	Hospital
input	-
output	view balance details
Description	Function allows you to balance details
Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	balance details view successfully
steps	1-Login in the system. 2- select balance details 3-Click in button “view balance details”

➢ View Total balance

actor	Hospital
input	-
output	View current balance
Description	Function allows you to view balance
Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	balance view successfully
steps	1-Login in the system. 2-choose one of them (expire-current-details)

	3-Click in button “Veiw Total”

➢ View Expired balance

actor	hospital
input	-
output	view balance
Description	Function allows you to view expire balance
Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	expire balance view successfully
steps	1-Login in the system. 2-select expire balance 3-Click in button “view”

➢ View Purchase orders

actor	hospital
input	-
output	view Purchase orders
Description	Function allows you to view Purchase orders
Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	Requests view successfully
steps	1-Login in the system. 2-select Purchase orders 3-Click in button “View Requests”

➢ View Donation Requests

actor	hospital
input	-
output	view Donor requests
Description	Function allows you to view Donor requests
Expected-Risk	-
Precondition	User already login in the system as hospital
Postcondition	Requests view successfully
steps	1-Login in the system. 2-select Donor requests 3-Click in button “view”

2.4.5 Non-Functional Requirement:

Performance: Defines how fast a system can respond to a particular user’s action under a certain workload.

Availability: Our application can be used in any time and it always

“uptime” is the amount of time that it is operational and available for use.

Usability: Our application is Ease to use. One of major things that user finds it friendly and easy to deal with.

Reliability: is the extent to which the app consistently performs the specified functions without failure.

Flexibility: is the ease in which the app can be modified to adapt to different environments, configurations, or user expectations.

Portability: is the ease with which the app can be moved from its current environment to another environment.

Maintainability: is the ease with which faults in the app can be found and repairs.

2.5 Advantages of the new system:

- The application saves money because it is entirely free the user can easily discover a donor who matches his blood type without having to spend any money.

The application offers a simple means of communication between the donors and the recipients, and it is simple enough for any user to use and comprehend.

- Save customers time by not having them wait an excessive amount of time for their blood types to be

available at hospitals.

- raising awareness of the importance of blood donation and its positive effects on health, which increased volunteer activity and inspired people to support one another.

- The application contributes to the development and enhancement of medical services.

- the application can assist numerous users in urgent and emergency situations.

2.6 Risk and Risk Managements:

Anything that has the potential to impact the project's schedule, performance, or money is considered a risk. In a project management setting, risks are potentialities that, if they materialize, are categorized as "problems" that need to be resolved. The following are some potential threats to our project:

1-poor password selection: Each new user must create an account on our system by providing a user name, password, etc. A weak password could pose a serious risk. Use a blend of upper-case, lower-case, numbers, and special characters as part of this risk management strategy. Make sure your password is lengthy—between 8 and 20 characters is ideal.

2-Password forgotten : Users may find it challenging to remember their passwords constantly, however the reset password option will help with this issue. Once they demonstrate that their email address is still in their possession or by providing their personal phone number, users will be able to enter a new password. -Application server downtime might be expensive. -The system's database can't handle as many transactions per second as it should

Chapter 3: Software Design

3.1 Use case Diagram:

Figure 2 “Use Case diagram”

Figure 3 “Admin Use Case diagram”

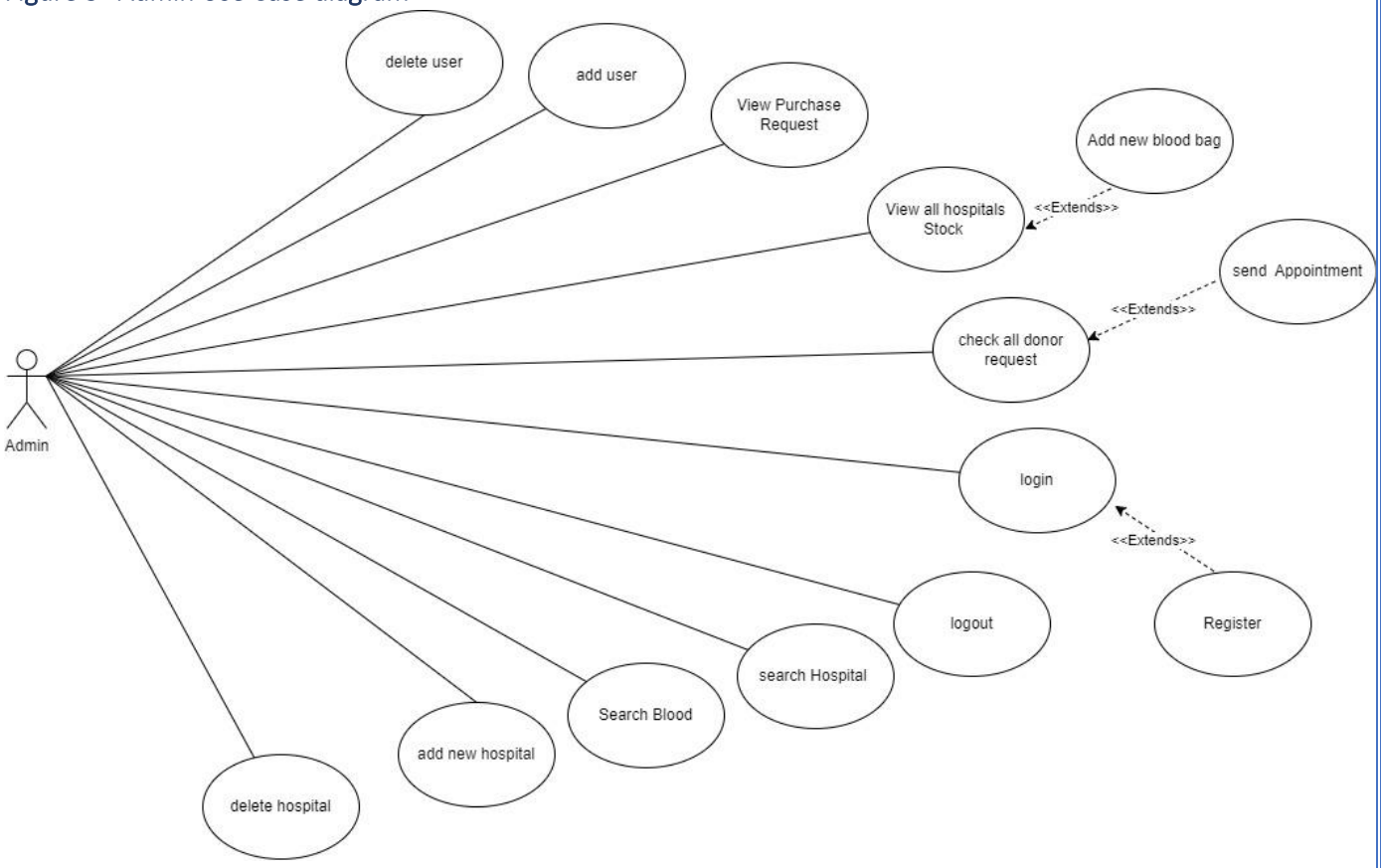


Figure 4 “User Use Case diagram”

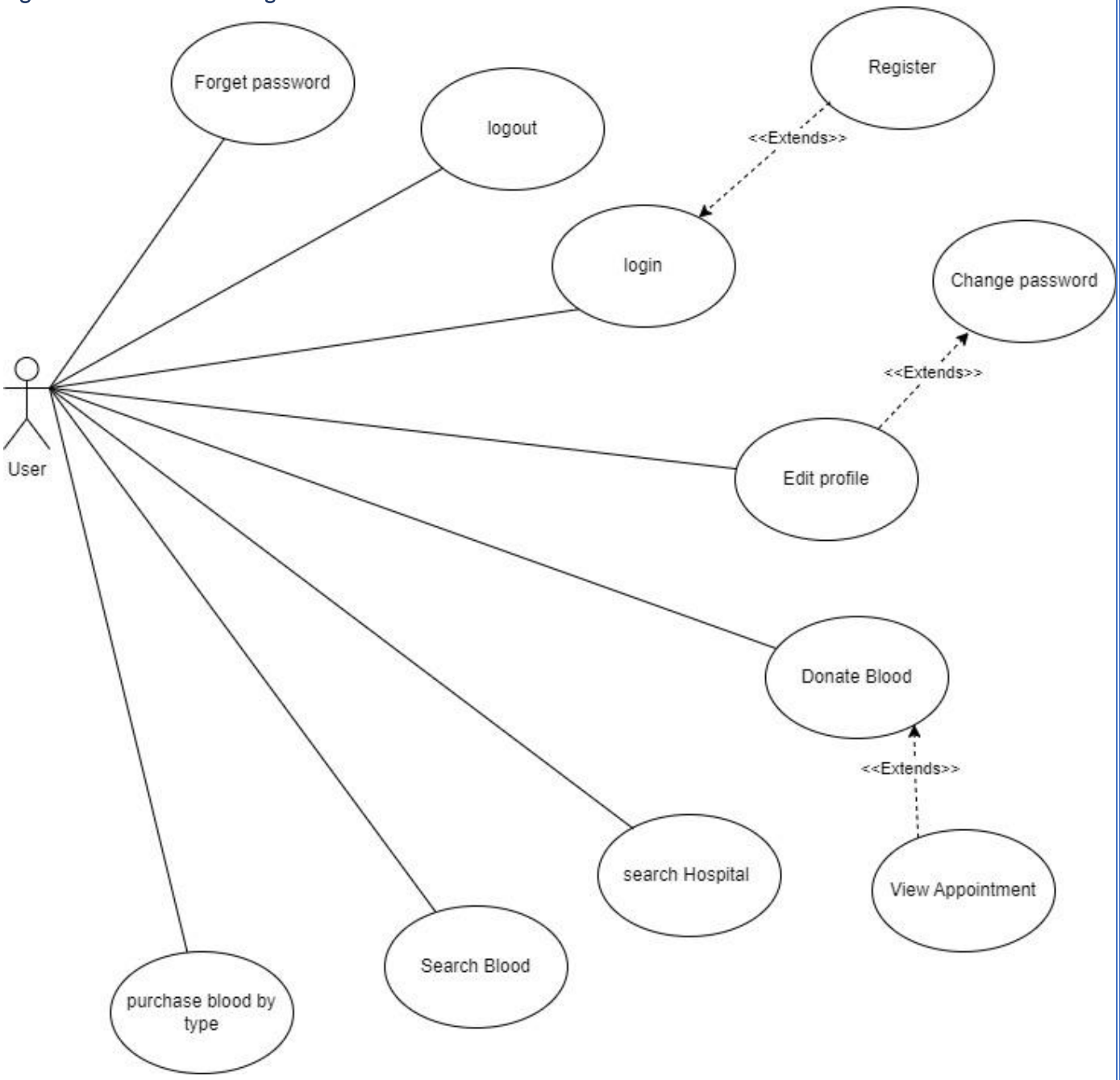
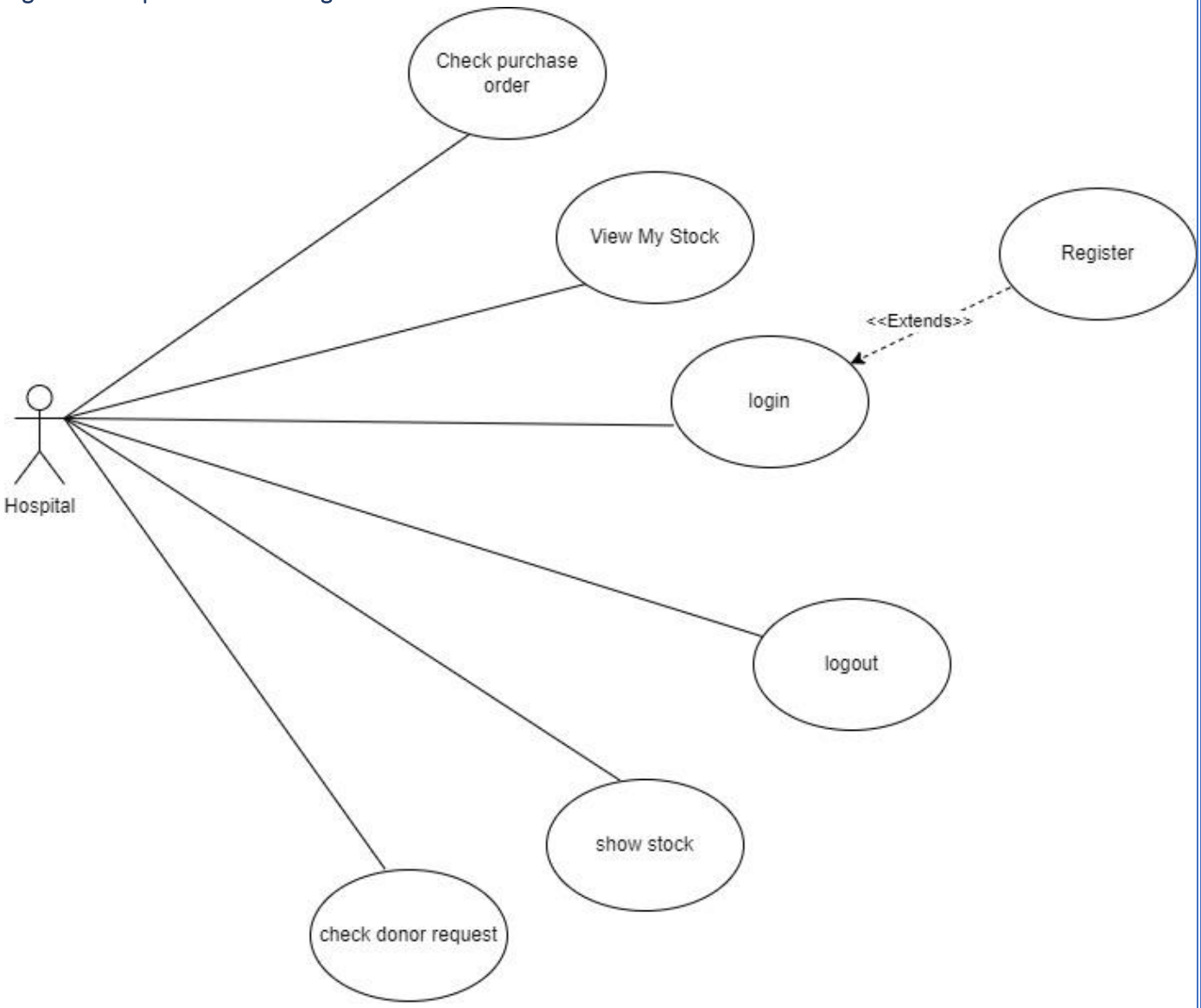


Figure 5 “Hospital Use Case diagram”



Use Case Scenario

“Register”

ID &Use Case Name	**1Register**
Initiator/Actors	The primary actor: (user) The secondary actor: (hospital).
Precondition	To enter the application and use its services.
Postcondition	The registration is saved in the database then he/she could login in the application and use its services.
Description of main sequence scenario	1-User download application. 2-User open application. 3-Click sign up button 4-Fill the required information in the registration form 5-Account is created.

“Login”

Use Case Name	**2 Login**
Initiator/Actors	The primary actor: (user) The secondary actor: (admin). The third actor hopital .
Goal	To enter the application and use its services.
Precondition	Any of the actors must be registered before.
Postcondition	They have an access to the system &they could use the application.
Description of main sequence scenario	1. Enter username and password. 2.user is verify to use the system

“Add new hospital”

Use Case Name	**Add new subject**
Initiator/Actors	The primary actor: Admin
Goal	New hospital will be added
Precondition	There is a data recorded in the database.
Postcondition	Hospital is successfully added

Description of main sequence scenario	1. Admin must login in the system. 2. Admin can add hospital information. 3. Hospital account will be created

"add new blood bag"

Use Case Name	**add new blood bag**
Initiator/Actors	The primary actor: Admin
Goal	New blood bag will be added
Precondition	View hospital stock
Postcondition	Blood bag is successfully added
Description of main sequence scenario	1. Admin must login in the system. 2. Admin can view hospital stock . 3. Admin can add blood bag

"search hospital"

Use Case Name	**search hospital**
Initiator/Actors	Admin, user
Goal	Found the hospital that I want
Precondition	He/she login to the system App.
Postcondition	Found the hospital
Description of main sequence scenario	1. Enter to search page 2. Enter hospital name 3. Click"search" icon

"check donor request"

Use Case Name	**check donor request**
Initiator/Actors	Admin, hospital

Goal	Actor check data for donor request.
Precondition	He/she is already on the system.
Postcondition	The request has been checked
Description of main sequence scenario	1. Actor must login system 2. Then check order

"Search blood"

Use Case Name	**Search blood**
Initiator/Actors	Admin , user
Goal	Get the correct blood type
Precondition	The user login to the system App.
Postcondition	Found the correct blood type
Description of main sequence scenario	1-Enter to search page 2-choose blood type. 3-click "search" icon

"Donates blood"

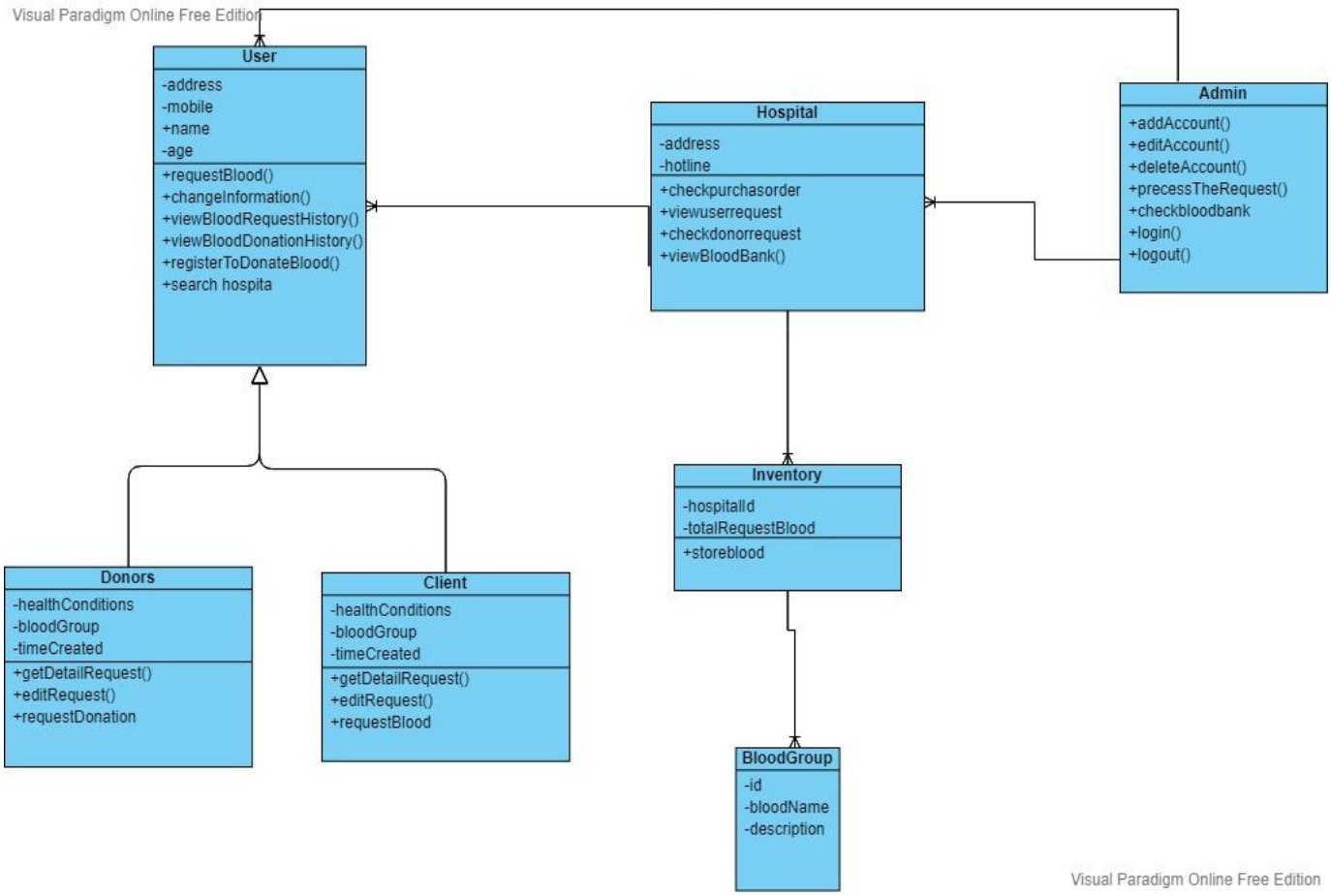
Use Case Name	**Donates blood**
Initiator/Actors	user
Goal	blood donation
Precondition	The user login to the system App.
Postcondition	make a request
Description of main sequence scenario	1. User must login in the system. 2. Search hospital 3. Make request

"Purchase blood”

Use Case Name	**Purchase blood**
Initiator/Actors	user
Goal	buy blood
Precondition	The user login to the system App.
Postcondition	make a request
Description of main sequence scenario	1. User must login in the system. 2. Search hospital 3. Search blood type 4. Make a request

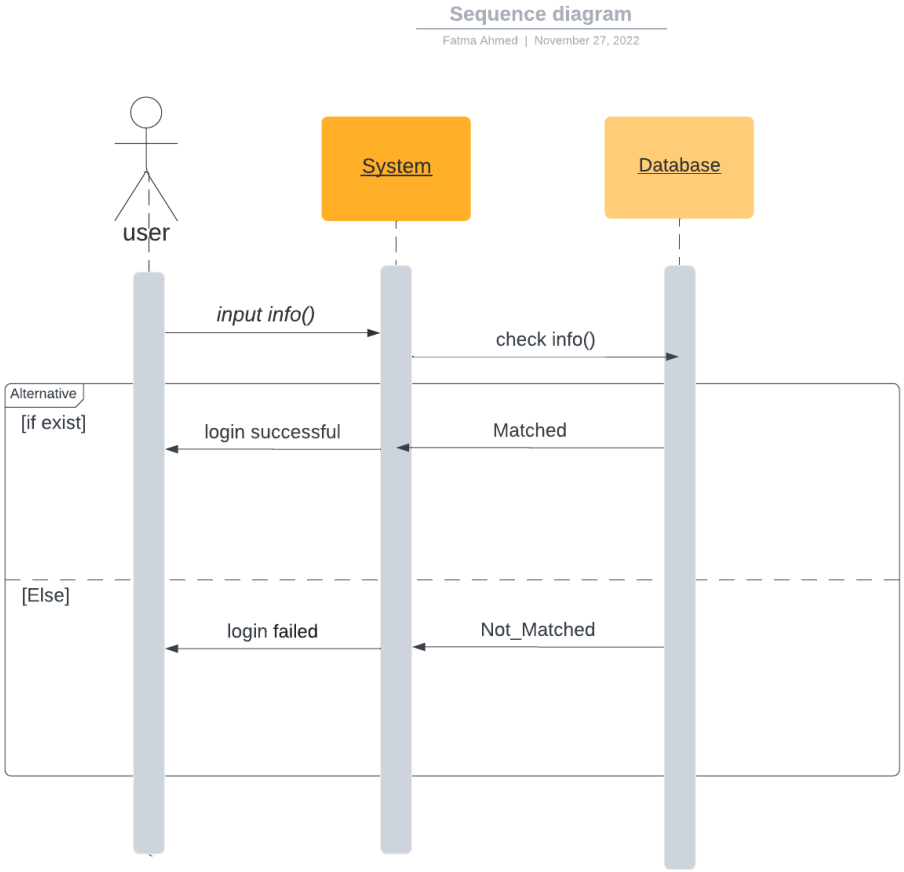
3.2 Class Diagram

Figure 6 “class diagram”



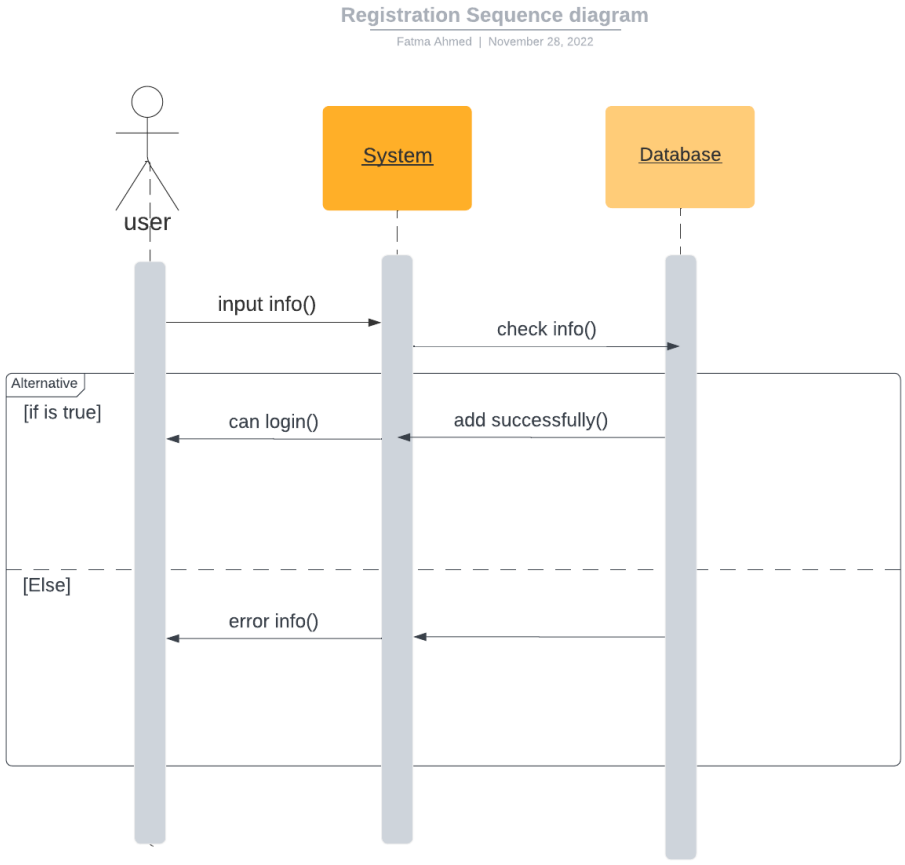
3.4 sequence diagram

Figure 7“User Sequence diagram 1”



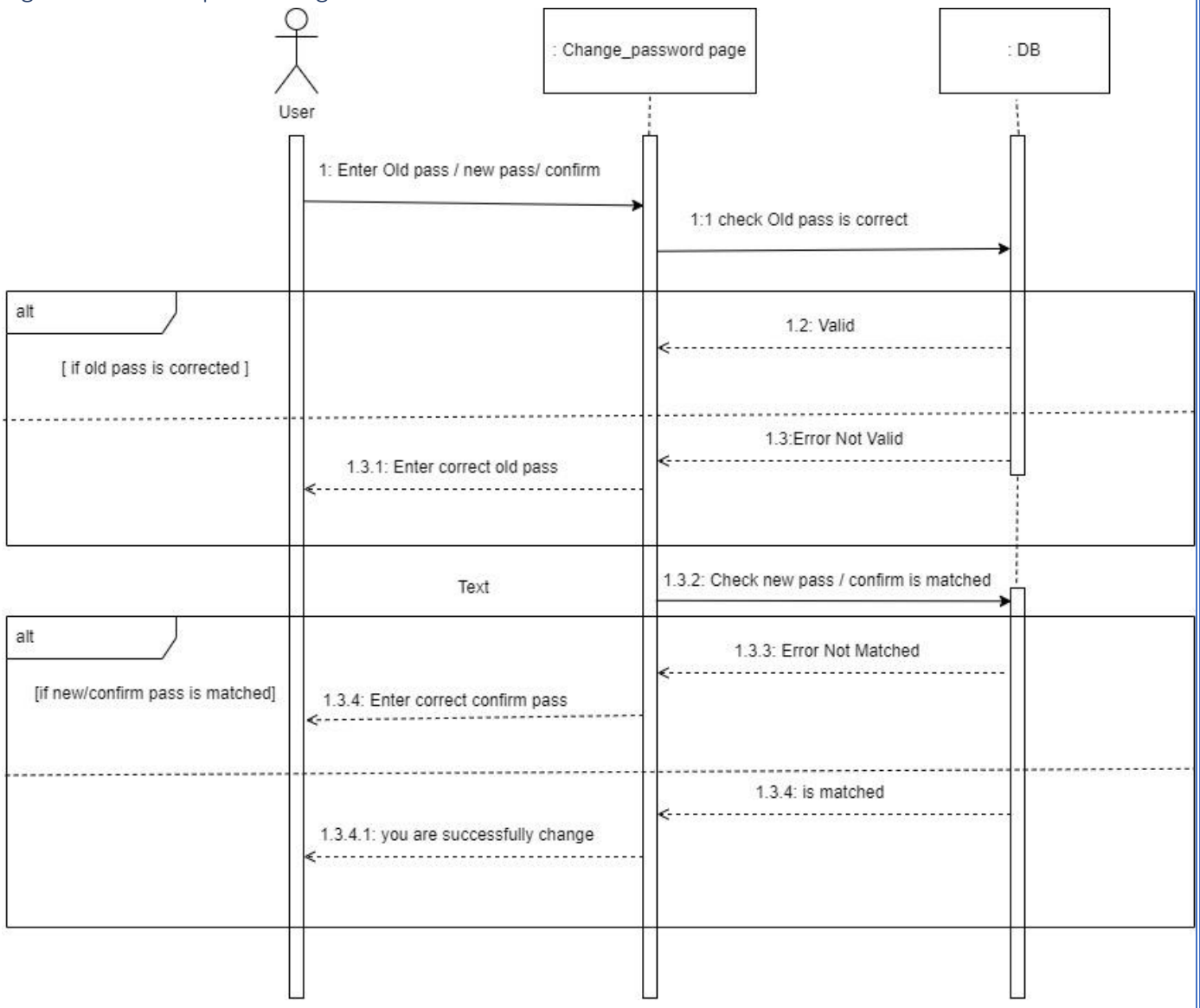
2-Register

Figure 8 “User Sequence diagram 2”



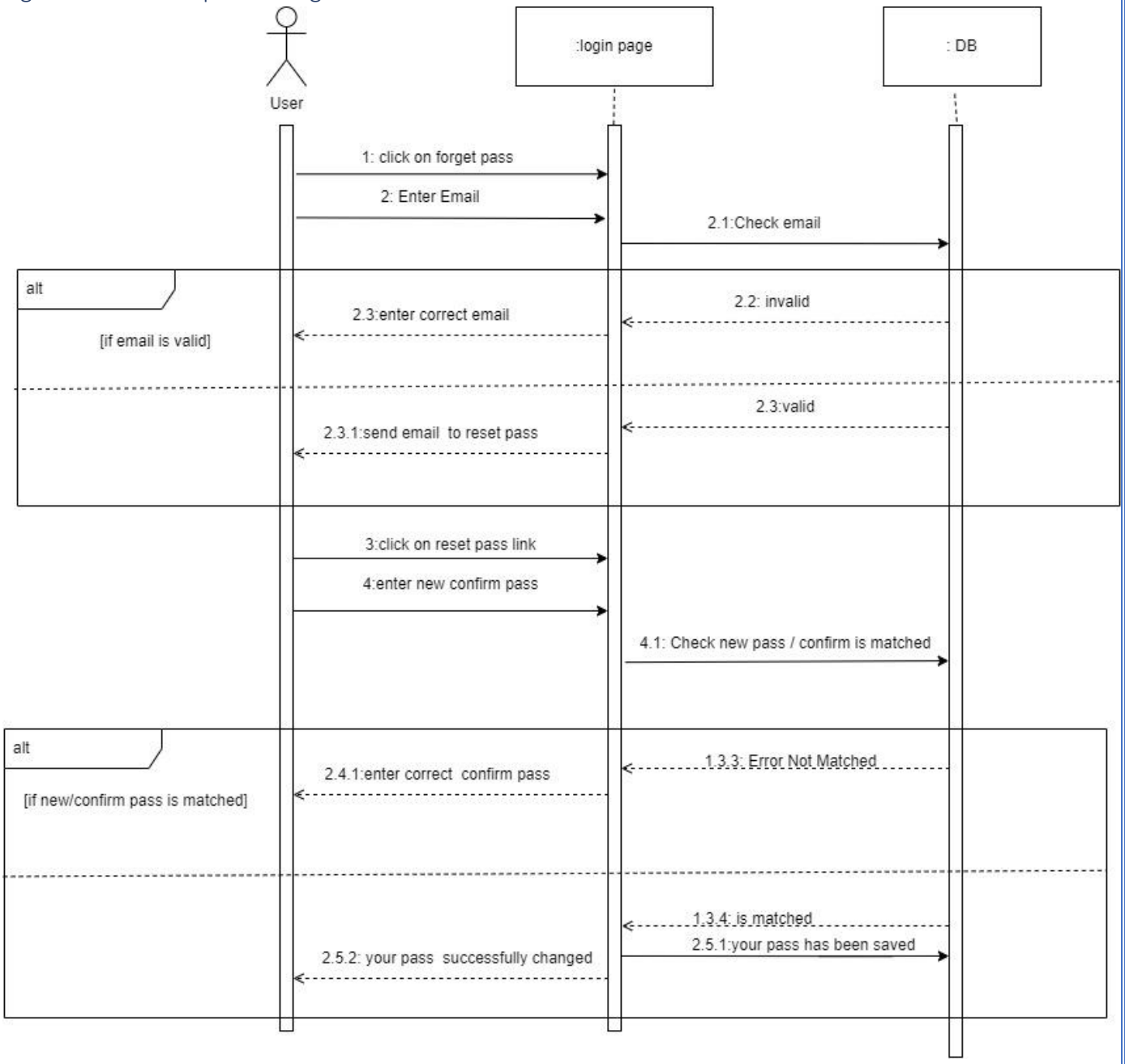
3-Change password

Figure 9 "User Sequence diagram 3"



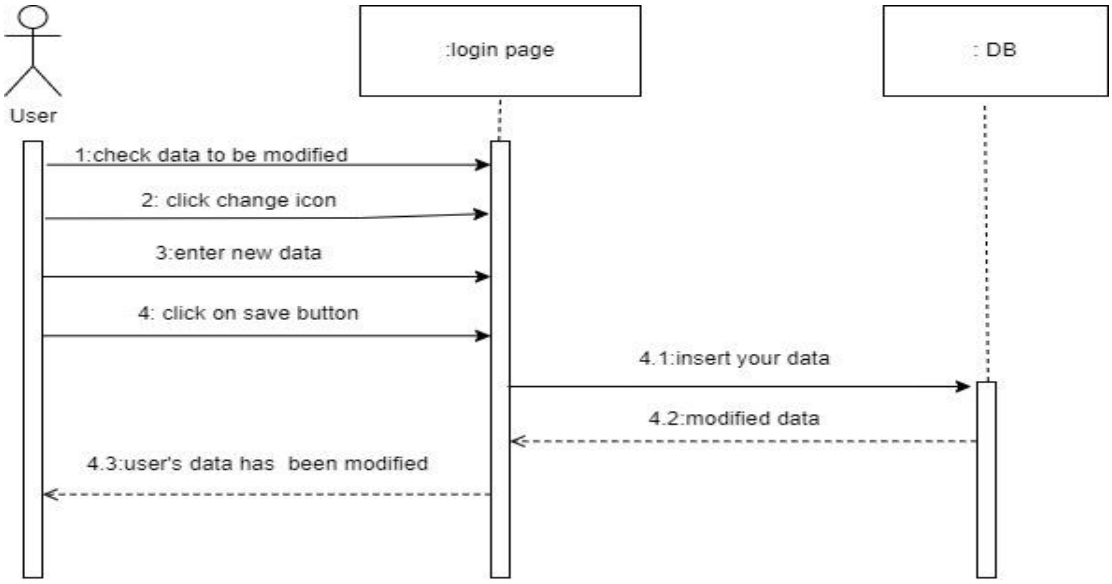
4-Forget password

Figure 10 “User Sequence diagram 4”



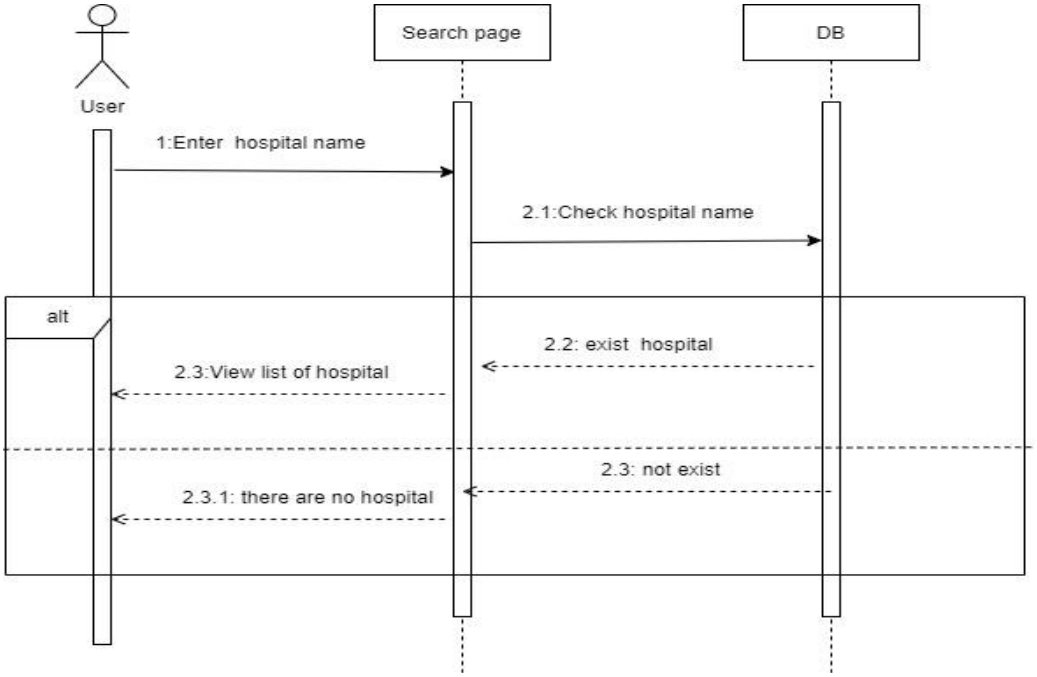
5-Edite profile

Figure 11 “User Sequence diagram 5”



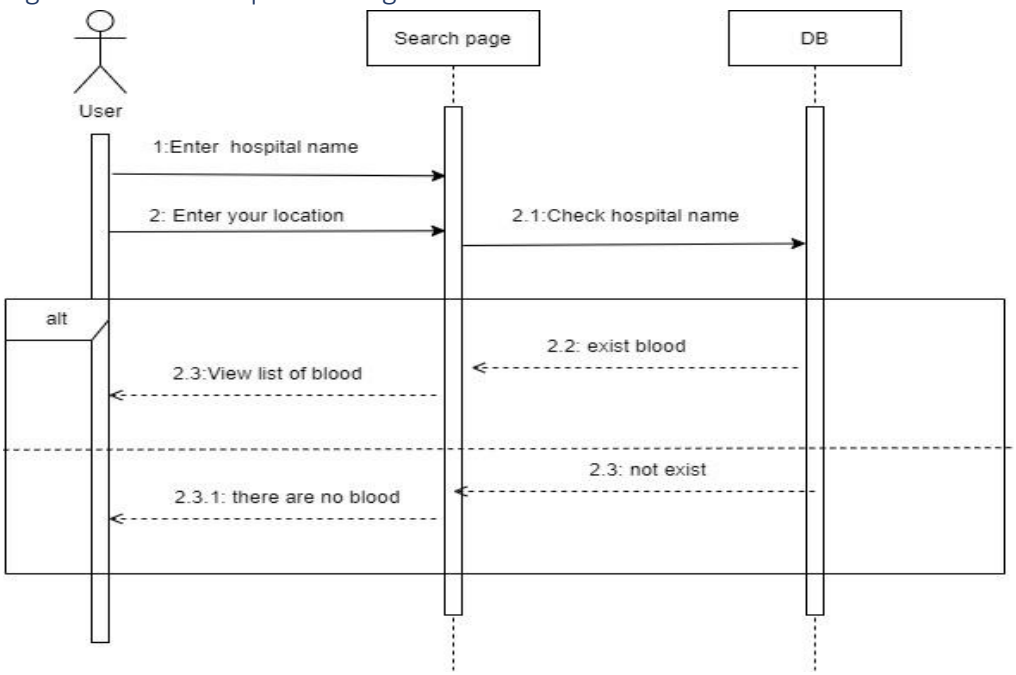
6-Search hospital

Figure 12 “User Sequence diagram 6”



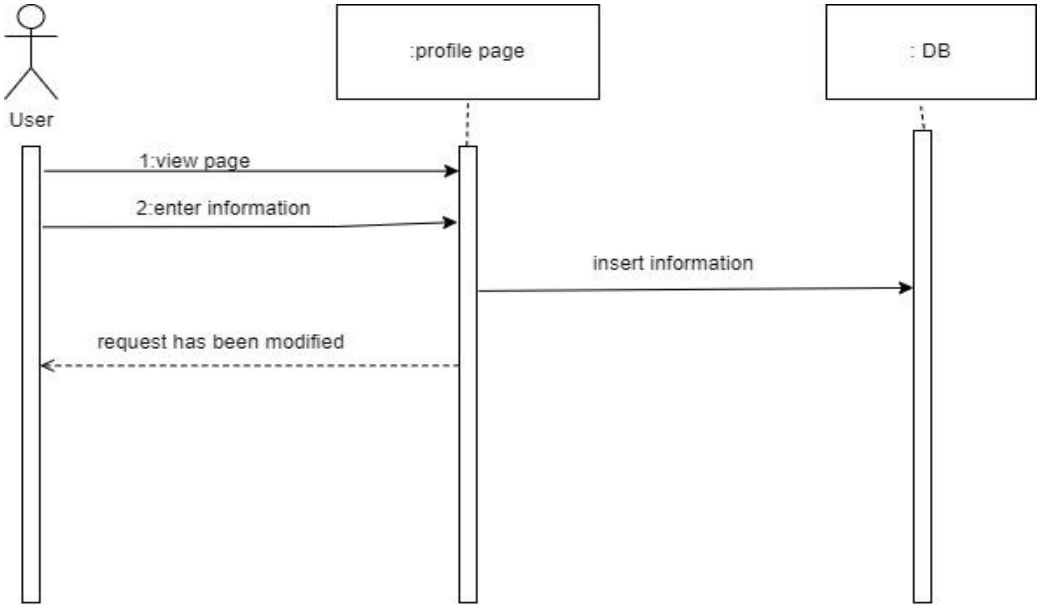
7-Search blood

Figure 13 “User Sequence diagram 7”



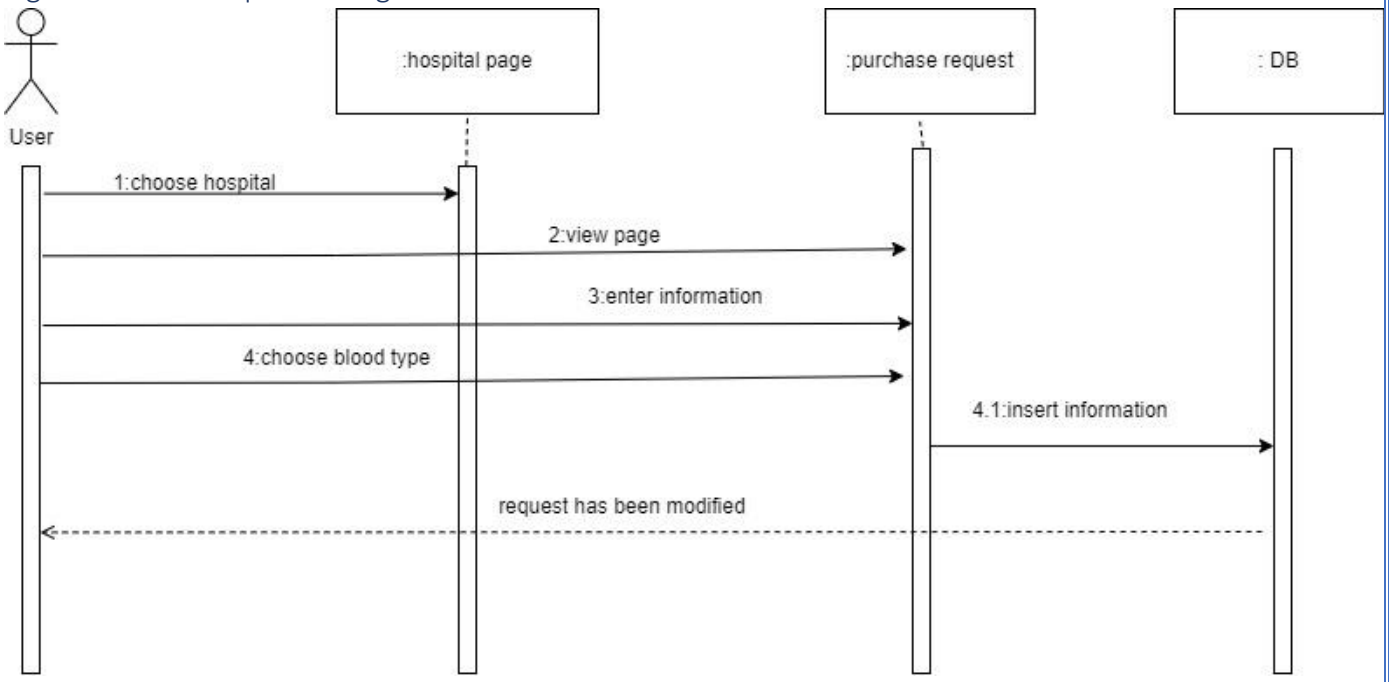
8-Request to donate

Figure 14 “User Sequence diagram 8”



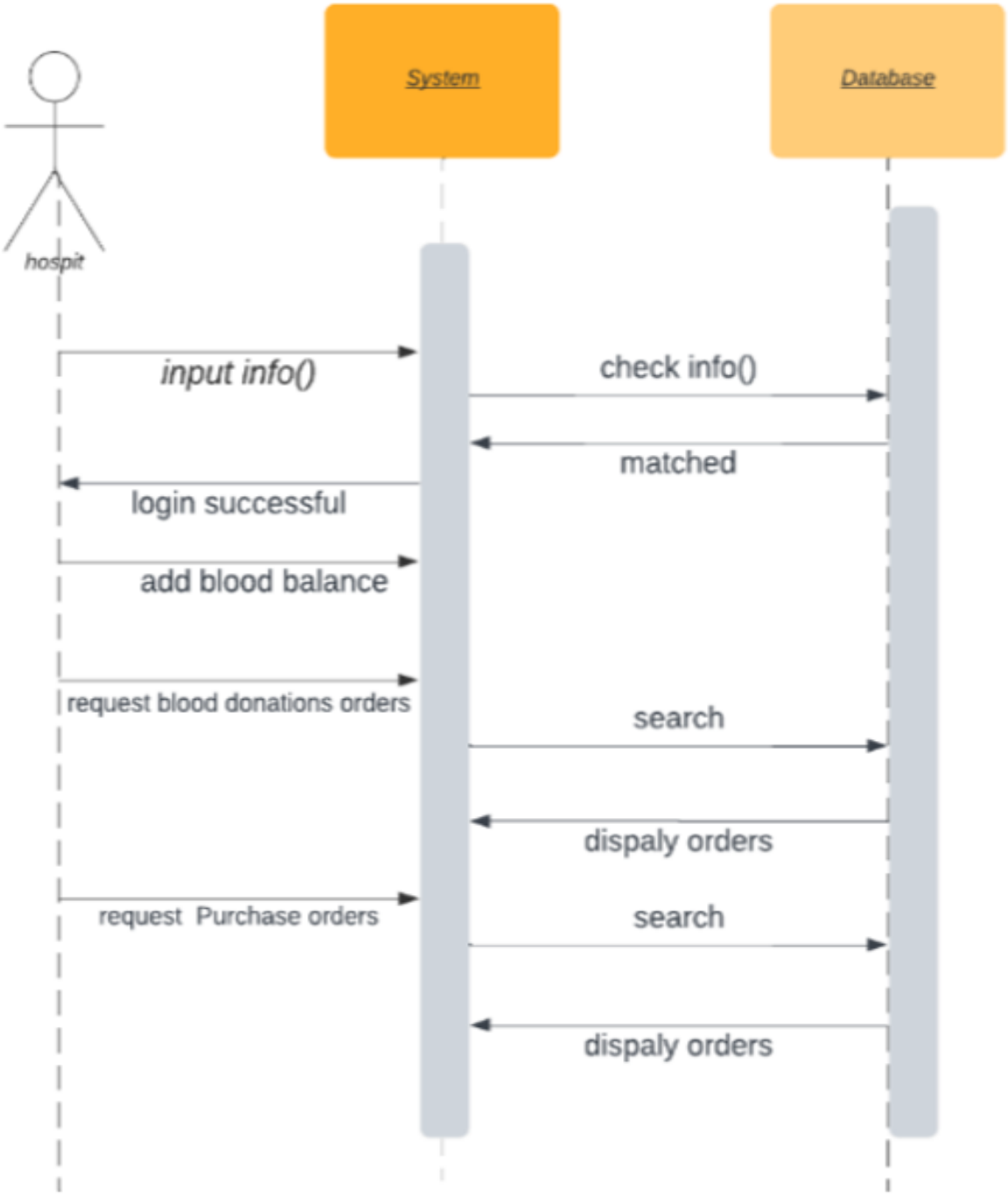
9-Purchase request

Figure 15“User Sequence diagram 9”



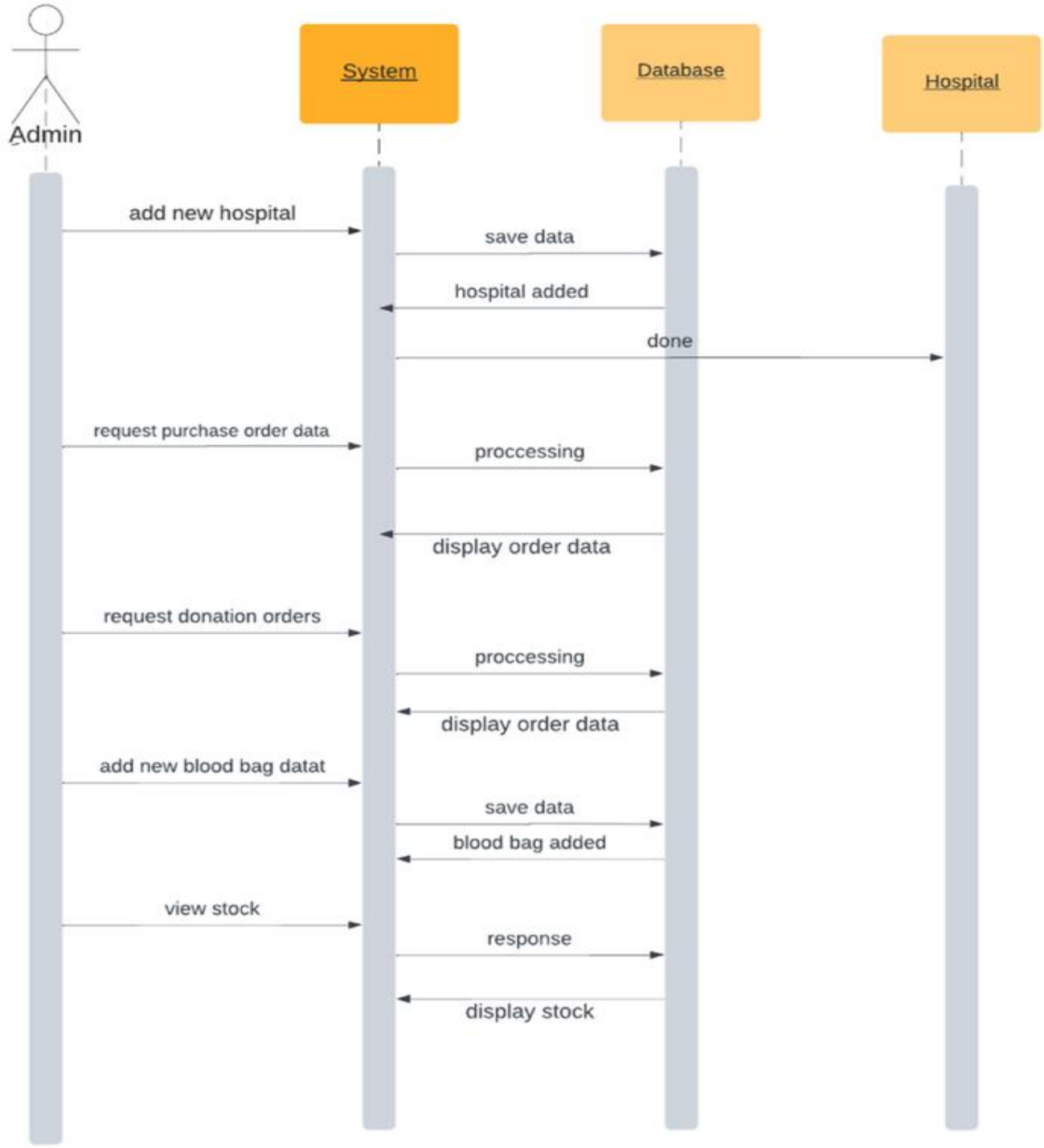
10-Hospital

Figure 17 “Hospital Sequence diagram 10”



11-Admin

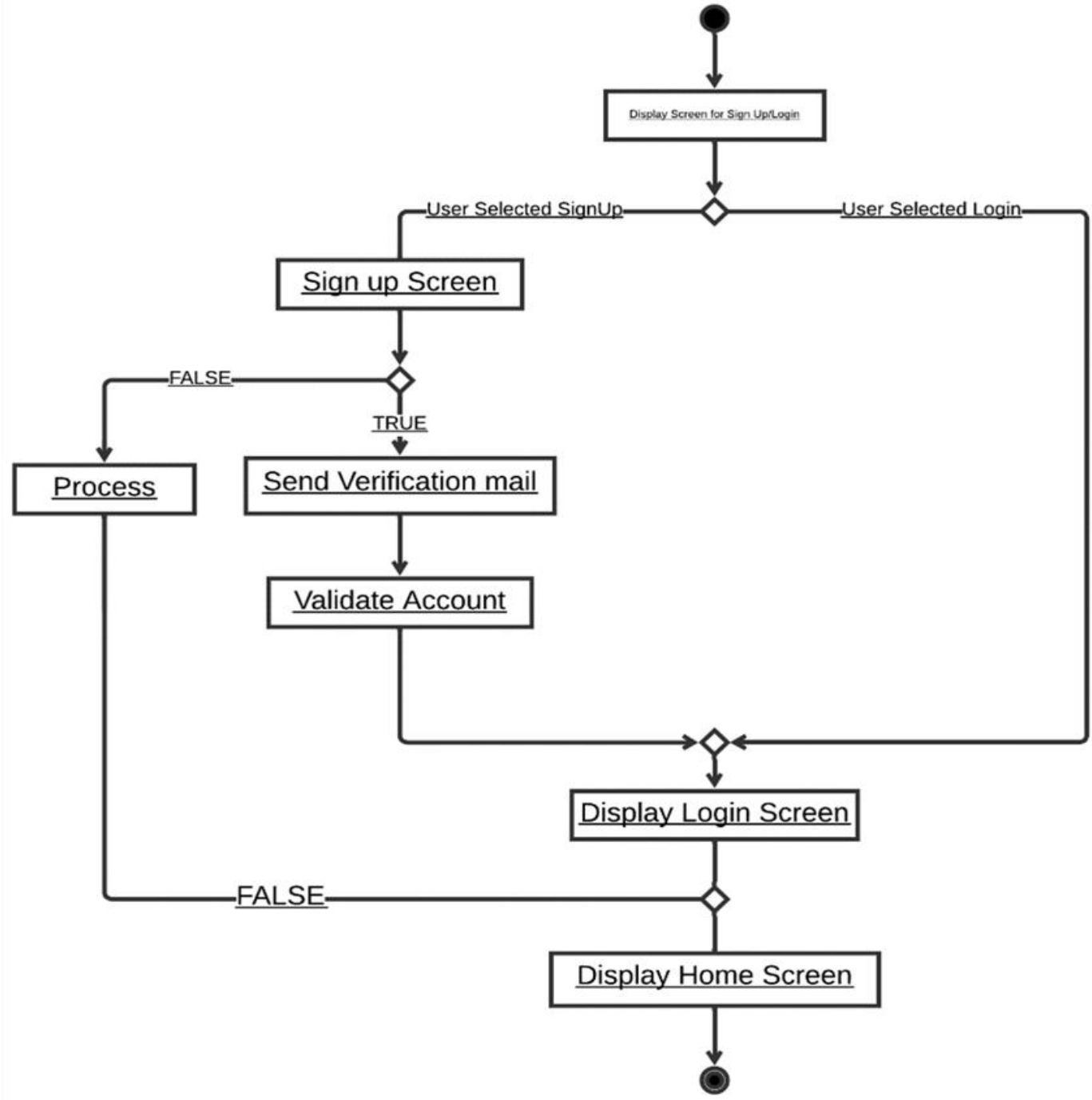
Figure 18 “Admin Sequence diagram 11”



Activity diagram

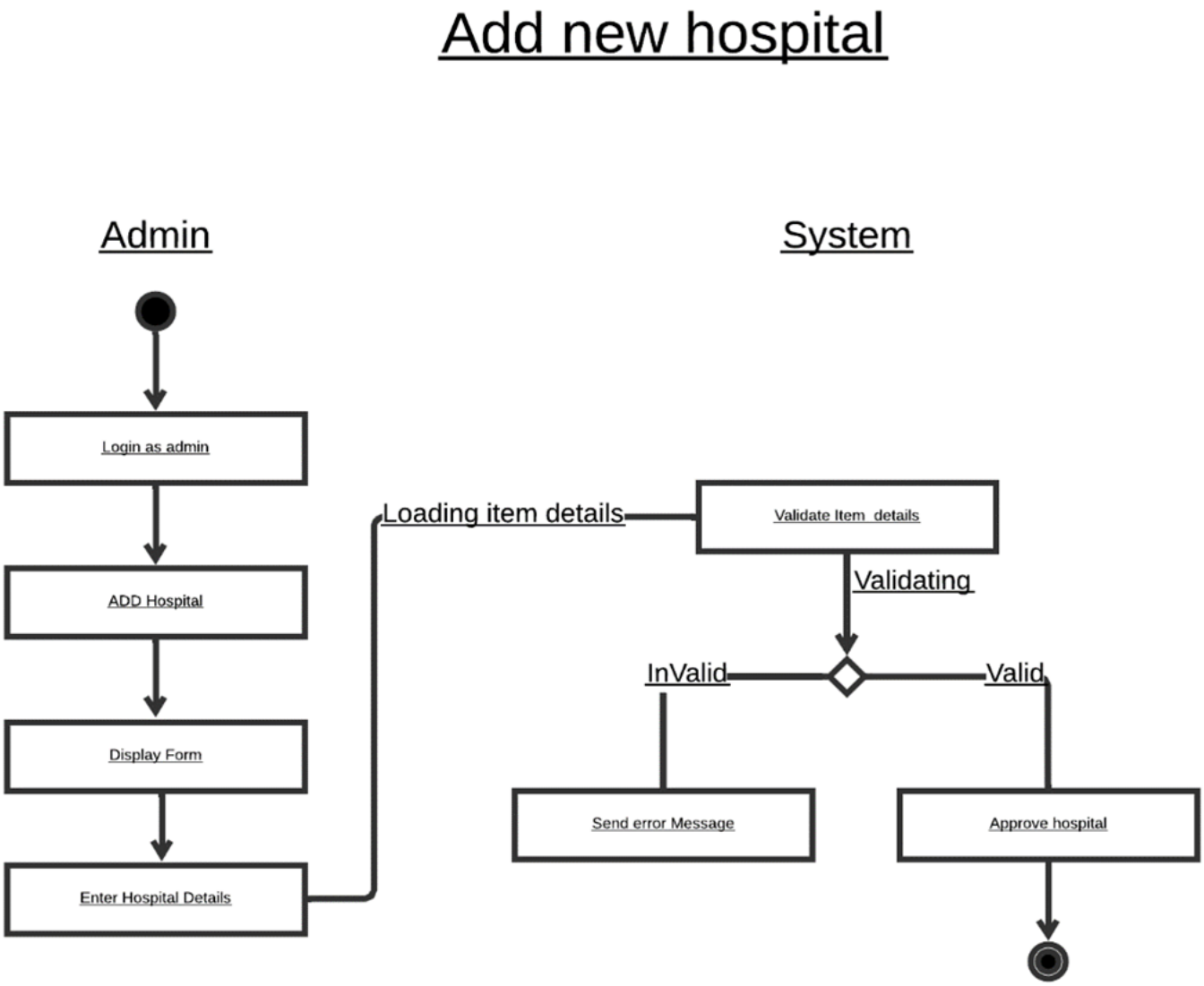
Sign in

Figure 19 “user activity diagram 12”



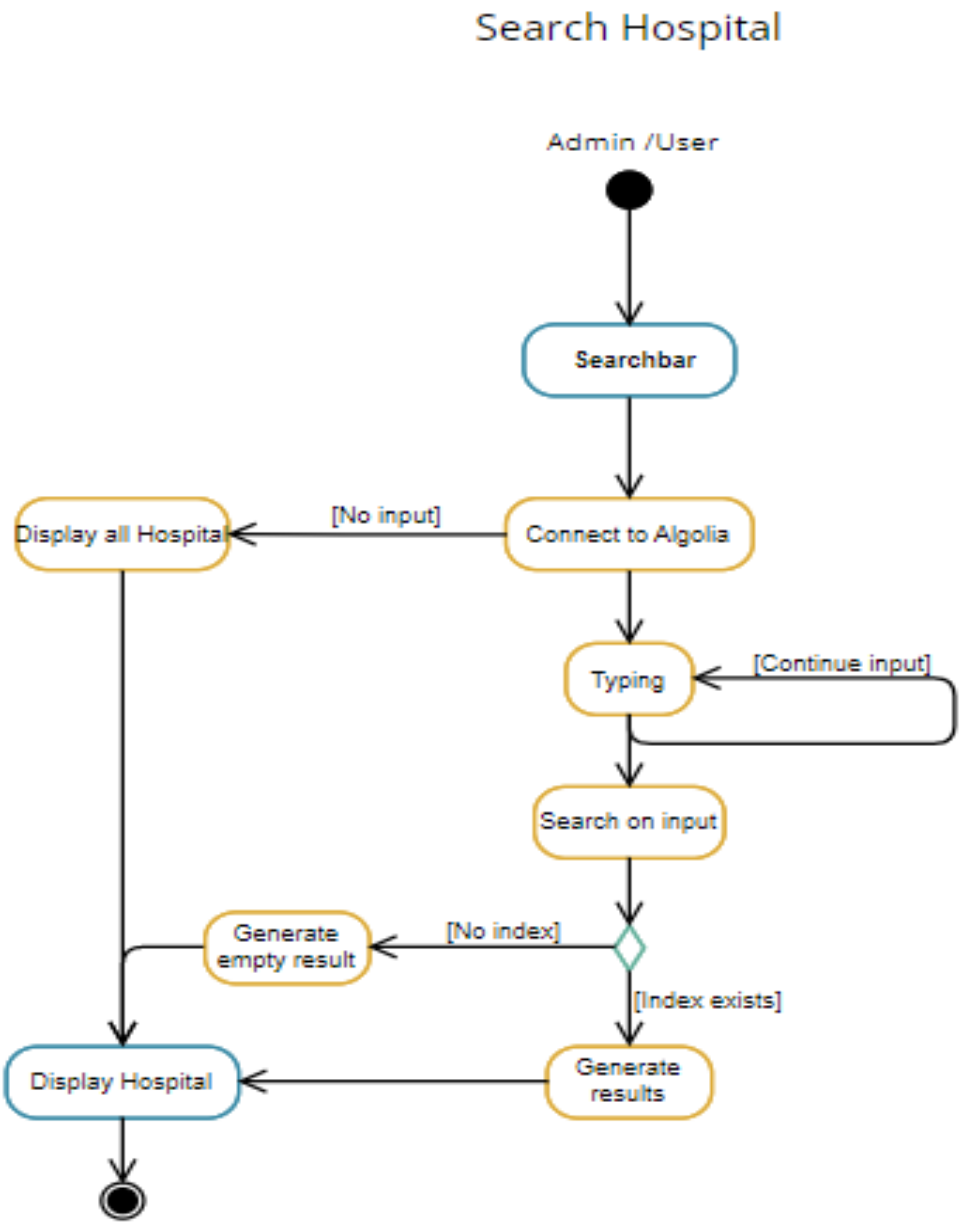
Add Hospital

figure 20 “user activity diagram 13”



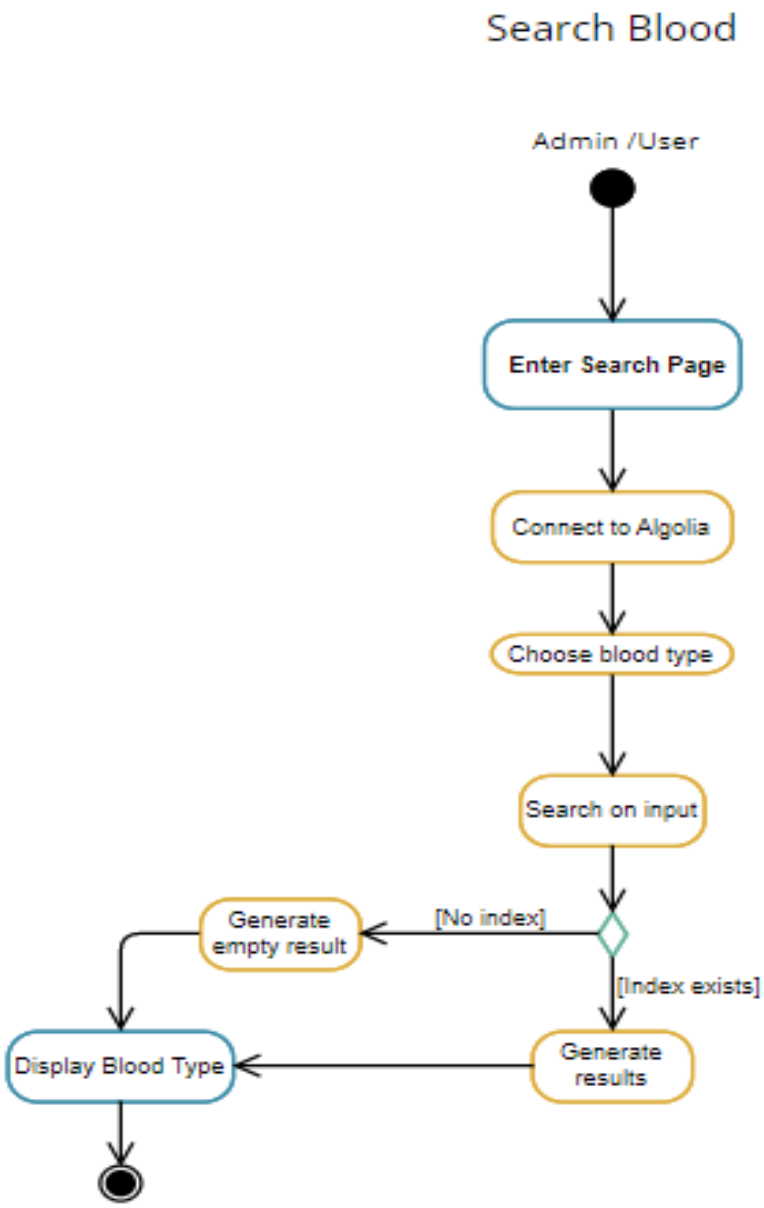
Search Hospital

figure 21 “user activity diagram 14”



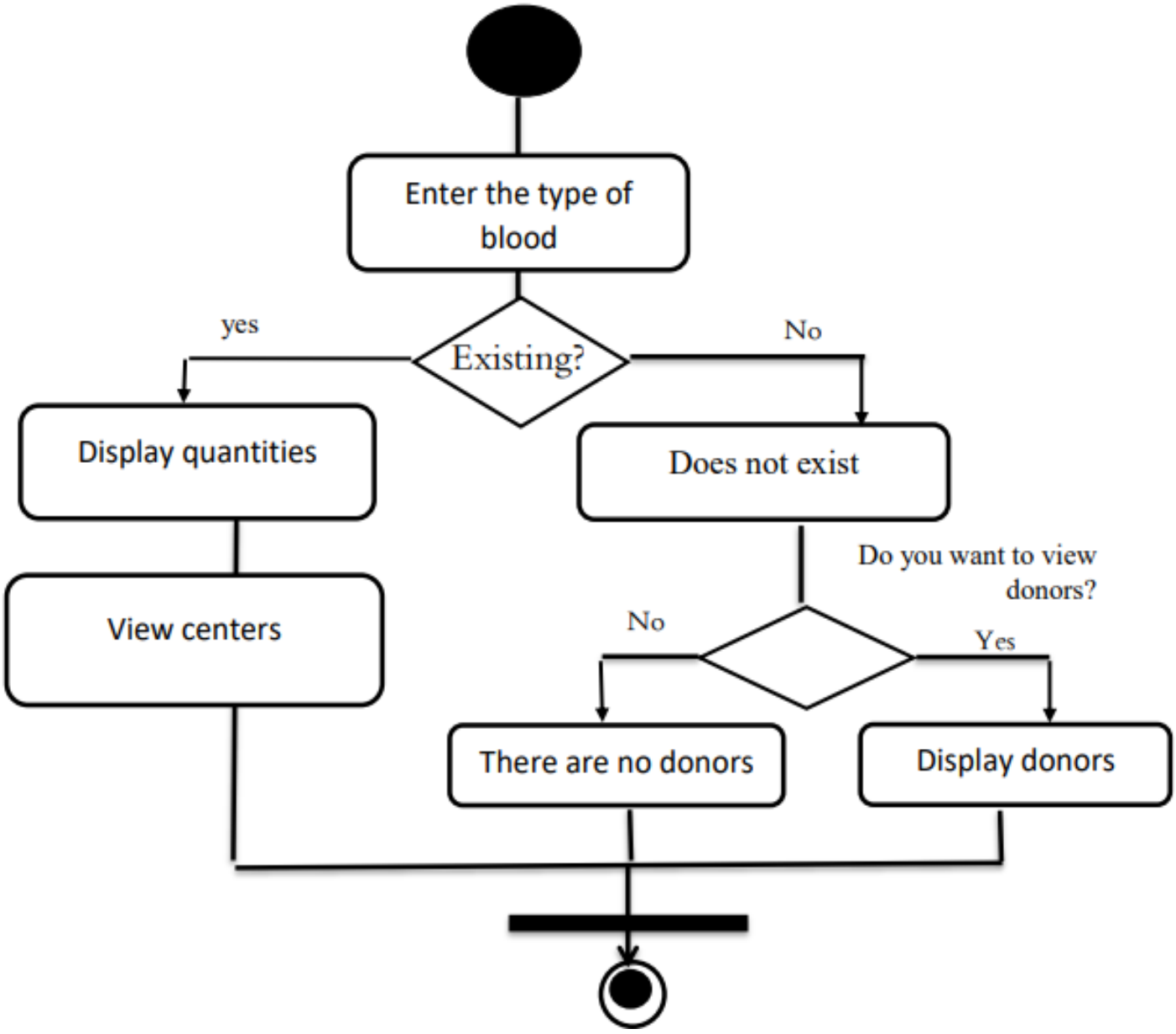
Search blood

figure 22 “user activity diagram 15”



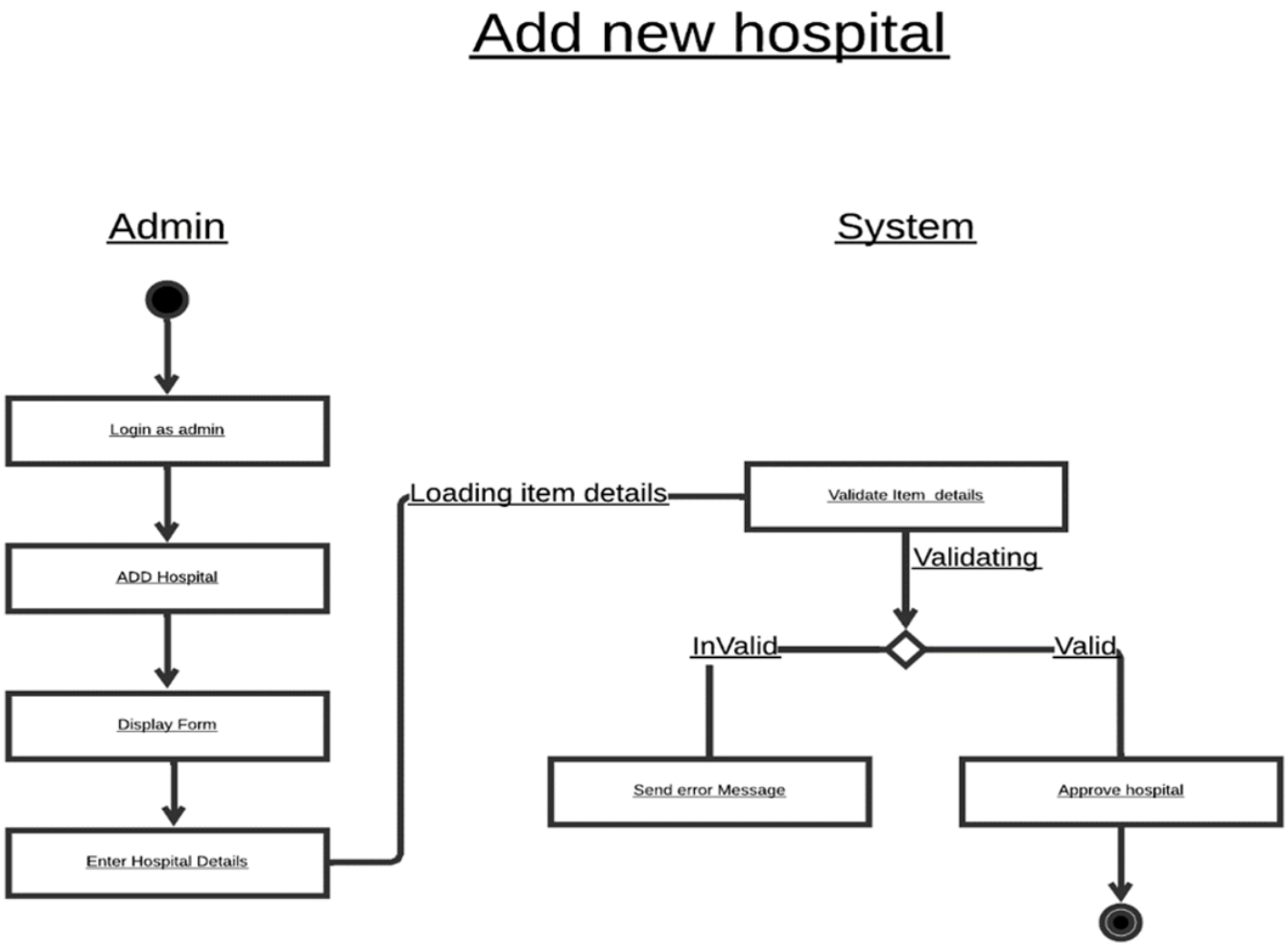
Search blood2

figure 23 “user activity diagram 16”



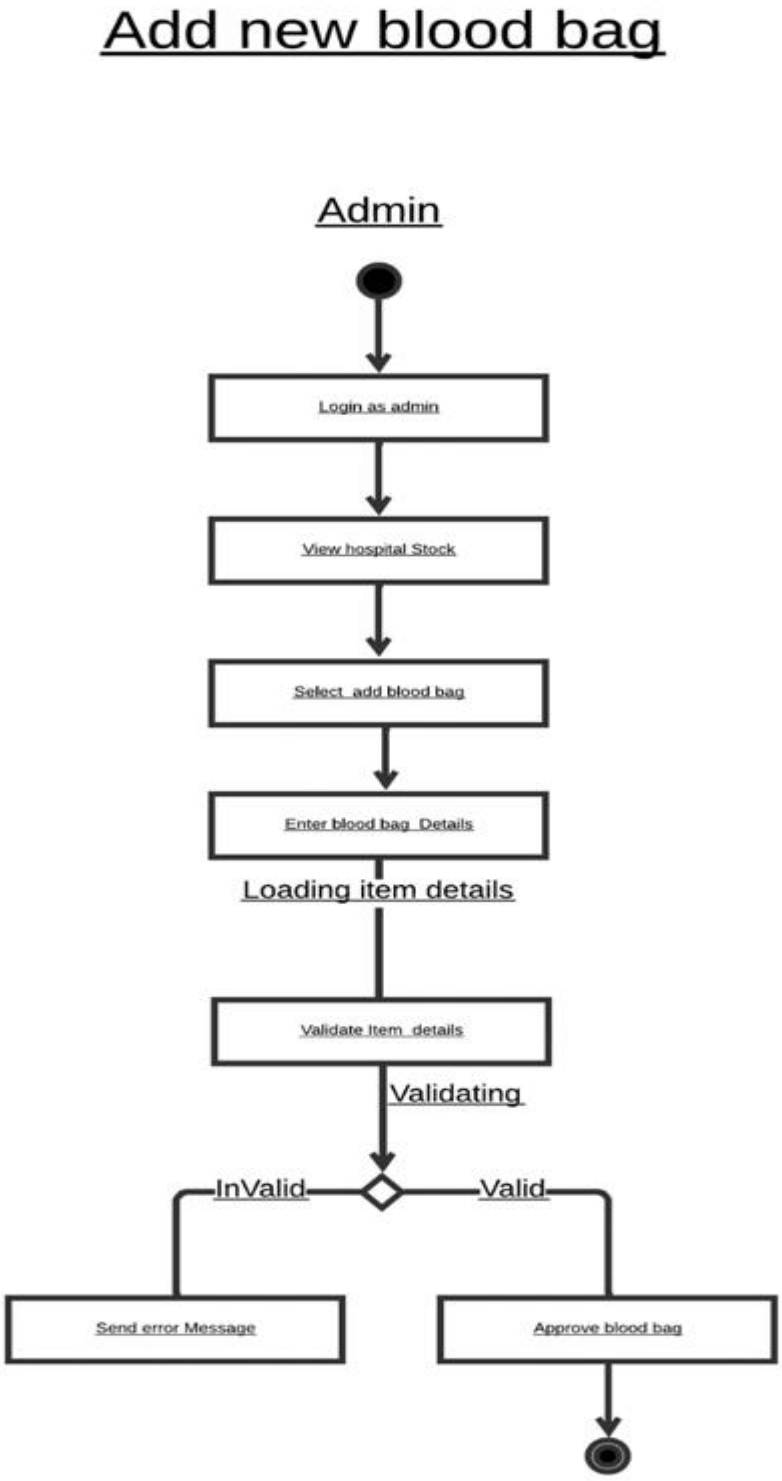
Add Hospital

figure 24 “Admin activity diagram 17”



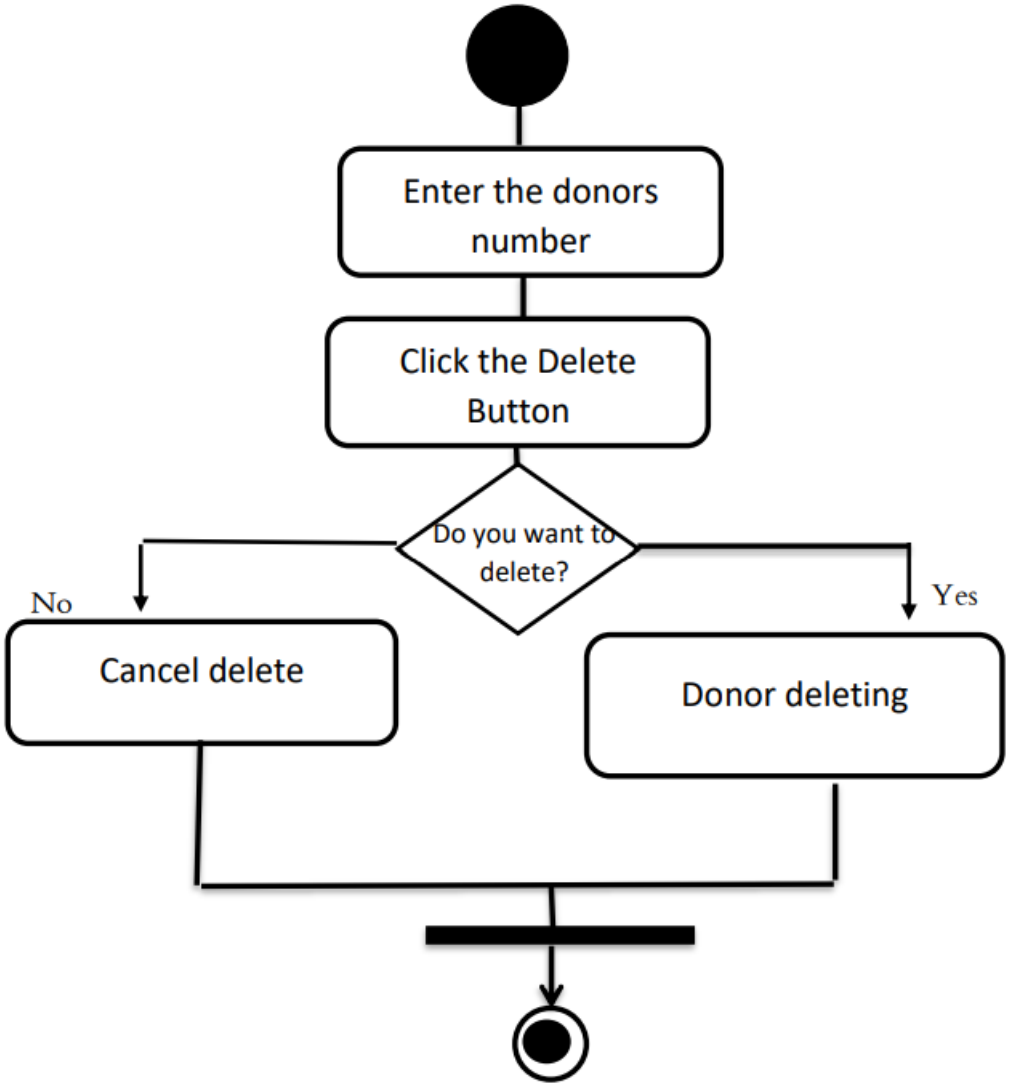
Add Blood Bag

Figure 25 “Admin activity diagram 18”



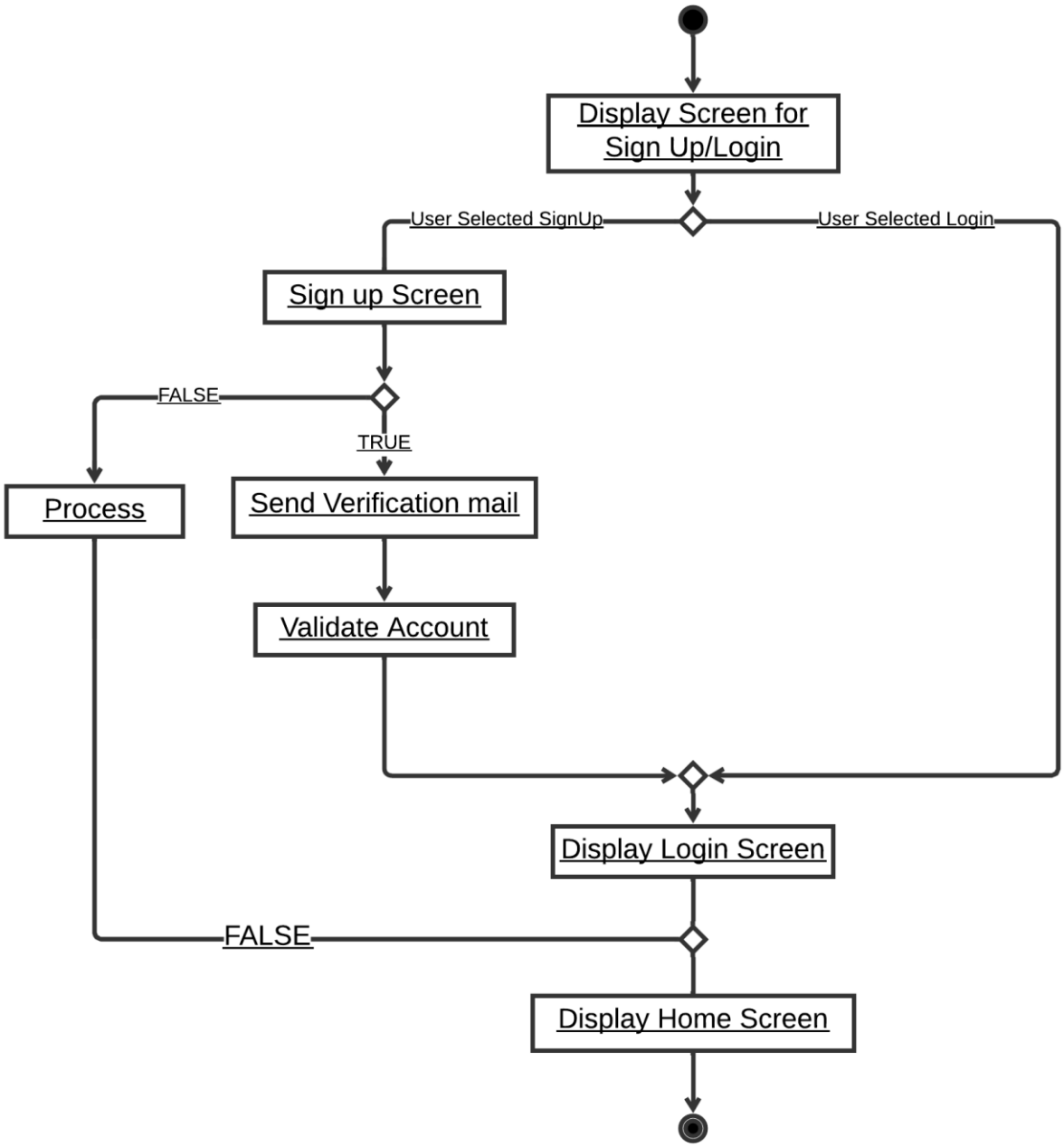
Delete User

figure 26 “Admin activity diagram 19”



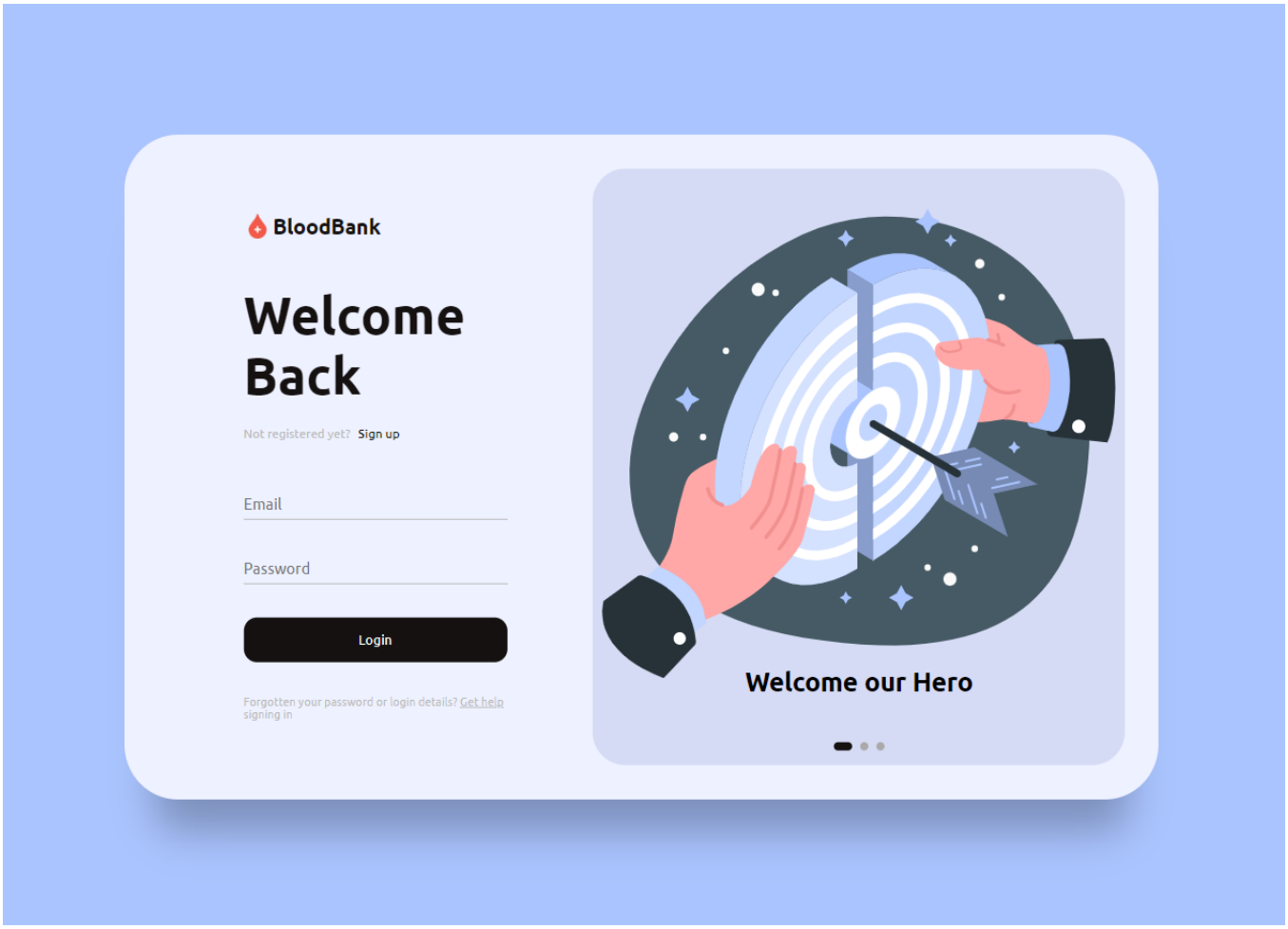
serve request

figure 27 “hospital activity diagram 20”



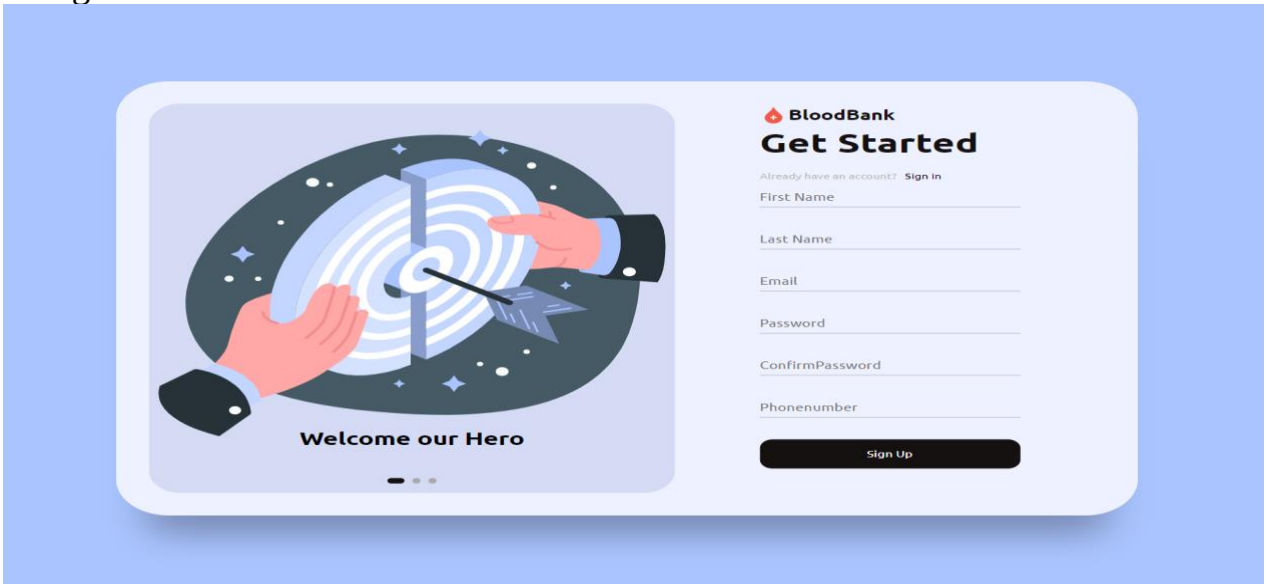
Chapter 4: Implementation

– Login:



```
[HttpPost]
[AllowAnonymous]
0 references
public async Task<IActionResult> Login(LoginAndRegisterViewModel user)
{
    if (user.Email != null && user.Password != null)
    {
        var User = await _userManager.FindByEmailAsync(user.Email);
        var Userid = _Context.USer.Where(s => s.Email == user.Email).FirstOrDefault();
        if (User != null)
        {
            var result = await _signInManager.CheckPasswordSignInAsync(User, user.Password, false);

            if (result.Succeeded)
            {
                await _signInManager.SignInAsync(User, isPersistent: false);

                var UserId = _Context.USer.FirstOrDefault(x => x.Email == user.Email);
                if (UserId.UserTypeId == 1)
                {
                    return RedirectToAction("Index", "Admin");
                }
                else if (UserId.UserTypeId == 2)
                {
                    return RedirectToAction("Index", "Hospital");
                }
                else
                {
                    return RedirectToAction("Shop", "User");
                }
            }
            else
            {
                ViewBag.Error = "User Not Found";
                ModelState.AddModelError(string.Empty, "User Not Found");
            }
        }
        else
        {
            ViewBag.Error = "User Not Found";
            ModelState.AddModelError(string.Empty, "User Not Found");
        }
    }
    return View(user);
}
```

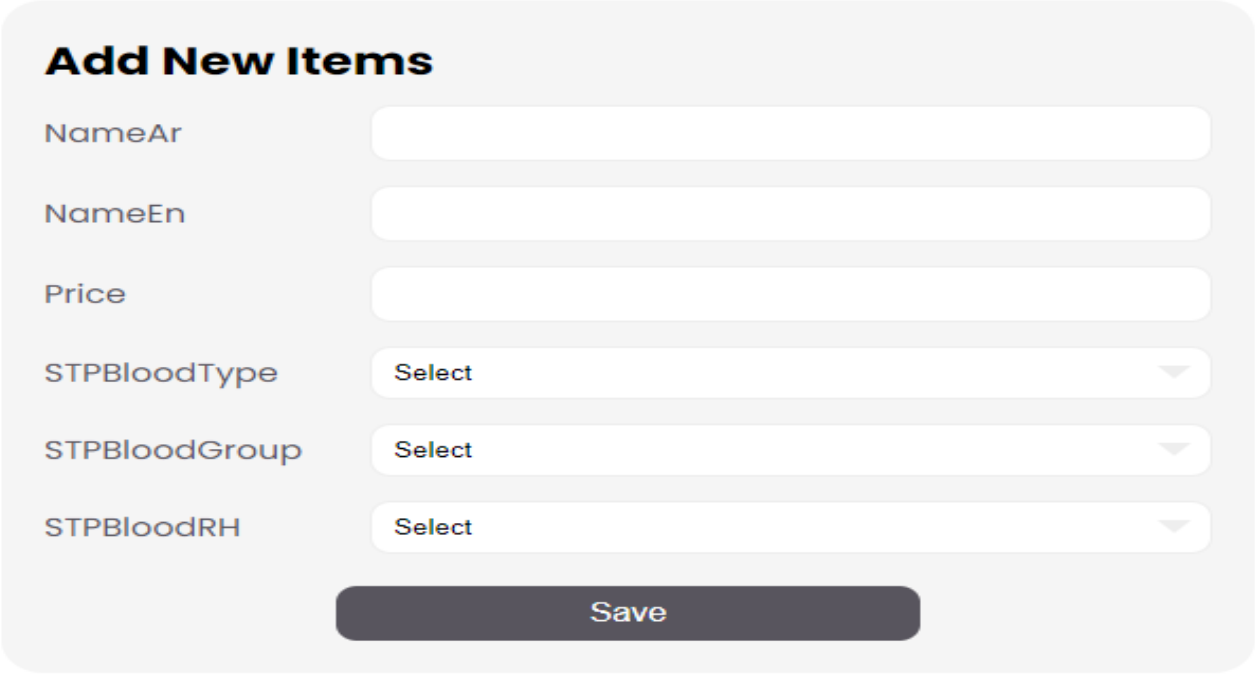
- Register :




```
[AllowAnonymous]
0 references
public IActionResult Register()
{
    return View();
}
[HttpPost]
0 references
public async Task<IActionResult> Register(LoginAndRegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new User
        {
            UserName = model.FirstName + model.LastName,
            Email = model.Email,
            EmailConfirmed = true,
            Type = "User",
            PhoneNumber = "0" + model.Phonenumber.ToString(),
            UserTypeId = 3,
        };

        var result = await _userManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            await _signInManager.SignInAsync(user, isPersistent: false);

            return RedirectToAction("Login", "Login", model);
        }

        foreach (var error in result.Errors)
        {
            ModelState.AddModelError("", error.Description);
        }
        ViewBag.Error = "Invalid Login Attempt";

        ModelState.AddModelError(string.Empty, "Invalid Login Attempt");

    }

    ViewBag.Error = "The password and confirmation password do not match";

    ModelState.AddModelError(string.Empty, "The password and confirmation password do not match");
    return RedirectToAction("Login","Login",model);
}
```

• Admin:

• Add new BloodBag :




```
0 references
public ActionResult _AddBloodBag()
{
    ViewBag.STPBloodGroup = _Context.STPBloodGroup.Select(x => new SelectListItem()
    {
        Text = x.NameAr,
        Value = x.Id.ToString(),
    });
    ViewBag.STPBloodType = _Context.STPBloodType.Select(x => new SelectListItem()
    {
        Text = x.NameAr,
        Value = x.Id.ToString(),
    });
    ViewBag.STPBloodRH = _Context.STPBloodRH.Select(x => new SelectListItem()
    {
        Text = x.Description,
        Value = x.Id.ToString(),
    });

    return PartialView();
}
0 references
```

- **Save BloodBag:**

```
0 references
public ActionResult SaveBloodBag(ItemsViewModel ItemsViewModel)
{
    if (ItemsViewModel.NameAr != null)
    {
        var UserId = _UserService.GetUserId();
        var product = new Items()
        {
            NameAr = ItemsViewModel.NameAr,
            NameEn = ItemsViewModel.NameEn,
            Price = ItemsViewModel.Price,
            Description = ItemsViewModel.Description,
            IsActive = true,
            IsDeleted = false,
            STPBloodGroupId = ItemsViewModel.STPBloodGroupId,
            STPBloodRHId = ItemsViewModel.STPBloodRHId,
            STPBloodTypeId = ItemsViewModel.STPBloodTypeId,
        };

        _Context.Items.Add(product);
        _Context.SaveChanges();
        return RedirectToAction("_AddBloodBag");
    }

    return View(ItemsViewModel);
}
```

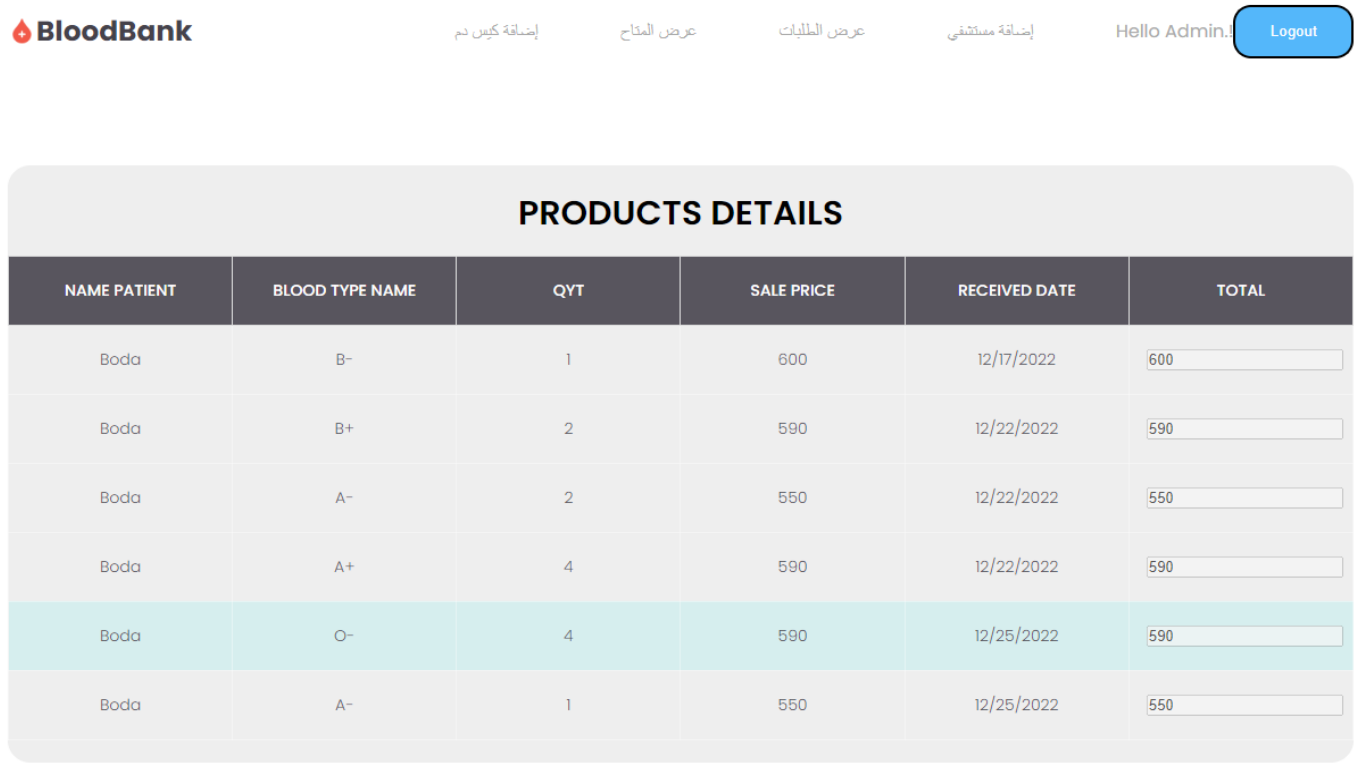
- **View Current Balance:**

CURRENT IN STORE

NAME HOSPITAL	BLOOD TYPE NAME	SALE PRICE	BLOODING DATE	ITEM EXPIRE DATE
Keeper	A+	590	12/12/2022 10:14:00 PM	1/6/2023 10:14:00 PM
Keeper	A-	550	12/12/2022 10:15:00 PM	1/7/2023 10:15:00 PM
Keeper	B+	590	12/12/2022 10:15:00 PM	1/6/2023 10:15:00 PM
Keeper	B-	600	12/12/2022 10:15:00 PM	1/4/2023 10:15:00 PM
Keeper	AB+	600	12/13/2022 10:16:00 PM	1/6/2023 10:16:00 PM
Keeper	AB-	600	12/12/2022 10:16:00 PM	1/7/2023 10:16:00 PM
Keeper	O+	550	12/14/2022 10:16:00 PM	1/7/2023 10:16:00 PM
Keeper	O-	590	12/13/2022 10:16:00 PM	1/7/2023 10:16:00 PM

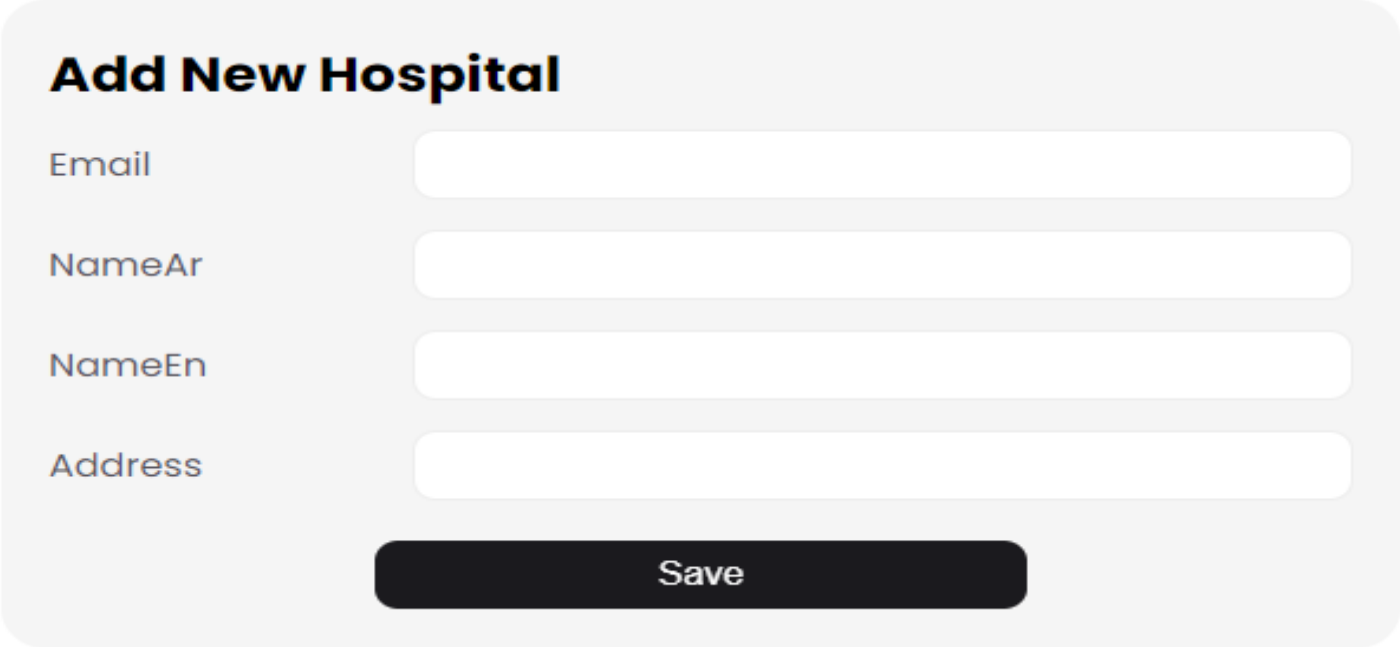
```
0 references
public ActionResult _ViewCurrentBalance()
{
    var TransactionItem = _Context.TransactionItem.Include(s => s.Item).Include(s => s.Transaction.Hospital).Where(x => x.ItemItemExpire ==
      false);
    var Transaction = TransactionItem.ToModel().ToList();

    return PartialView(Transaction);
}
```

•All Products Details:




```
0 references
public IActionResult AllProductsDetails()
{
    return View();
}
0 references
public IActionResult _AllProductsDetails()
{
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    if (UserId.UserTypeId == 1)
    {
        List<RequestProductsDetails> RequestProductsDetailsList = new List<RequestProductsDetails>();
        var cart = _Context.CartDetails.Include(x => x.Items).Include(x => x.TransactionItem).Include(x => x.CartMaster).Where(x =>
          x.IsActive == true && x.IsDeleted == false && x.IsPaid == true && x.AcceptFromHospital == false && x.RejectionFromHospital ==
          false).ToList();
        foreach (var item in cart)
        {
            RequestProductsDetails RequestProducts = new RequestProductsDetails();
            RequestProducts.Id = item.Id;
            RequestProducts.ItemName = item.Items.NameAr;
            RequestProducts.Price = item.TransactionItem.UnitSalePrice;
            RequestProducts.QYT = item.QYT;
            RequestProducts.ReceivedDate = item.AddedTime.AddDays(1).ToShortDateString();
            RequestProducts.Total = item.TransactionItem.UnitSalePrice;
            var user = _Context.USer.Where(x => x.Id == item.CartMaster.UserId).FirstOrDefault();
            RequestProducts.PatientName = user.UserName;

            RequestProductsDetailsList.Add(RequestProducts);
        }
        return PartialView("_AllProductsDetails", RequestProductsDetailsList);
    }

    return PartialView("_AllProductsDetails");
}
```

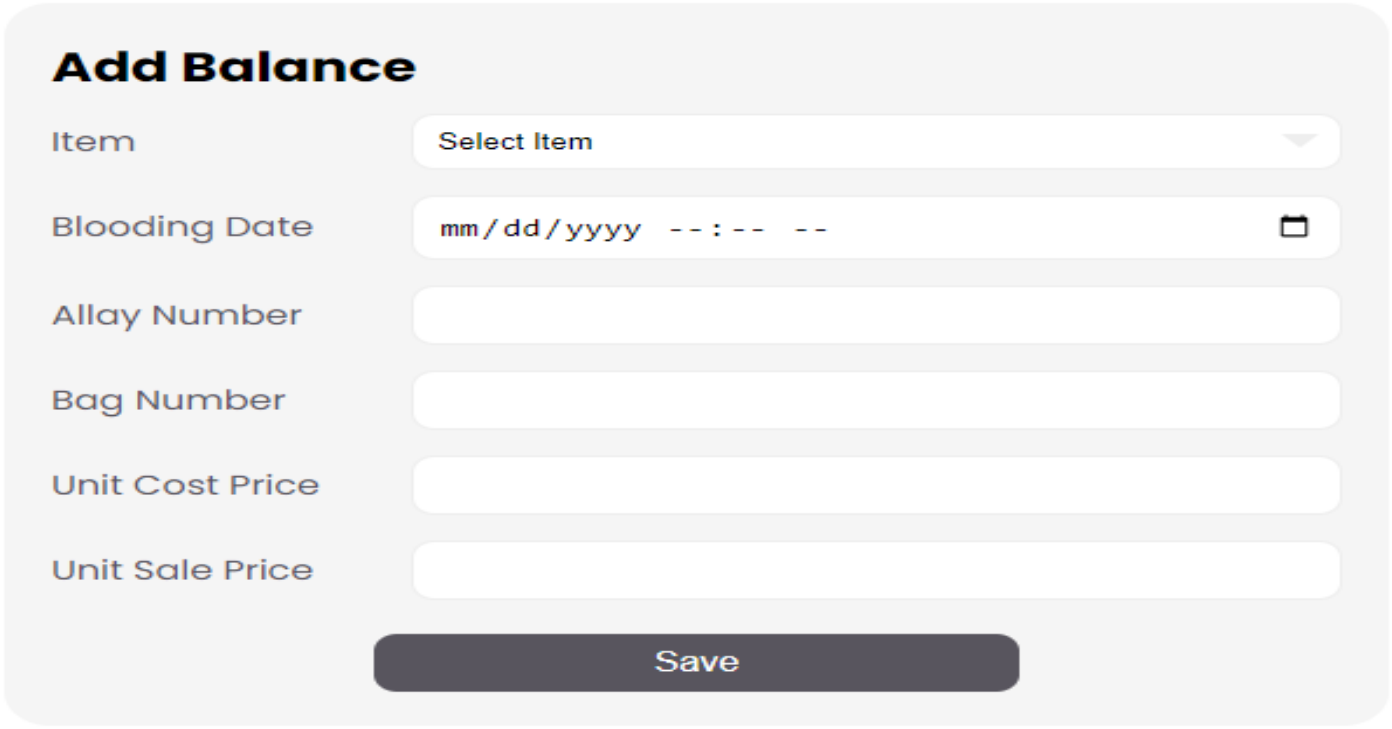
•Add Hospital:




```
0 references
public ActionResult _AddHospital()
{
    return PartialView();
}
0 references
public ActionResult AddNewHospital(HospitalModel Hospital)
{
    var user = _Context.USer.FirstOrDefault(x => x.Email == Hospital.Email);
    if (user != null)
    {
        Hospital hospital = new Hospital();
        hospital.NameAr = Hospital.NameAr;
        hospital.NameEn = Hospital.NameEn;
        hospital.Address = Hospital.Address;
        hospital.UserId = user.Id;
        hospital.IsActive = true;
        hospital.IsDeleted = false;

        _Context.Hospital.Add(hospital);
        _Context.SaveChanges();

        if (hospital.Id != 0 && hospital.Id != null)
        {
            UpdateUserType(user);
        }
    }

    return RedirectToAction("_AddHospital");
}
```

• Hospital:

•Add Balance:




```
0 references
public ActionResult _AddBalance()
{
    ViewBag.Items = _Context.Items.Select(x => new SelectListItem()
    {
        Text = x.NameAr,
        Value = x.Id.ToString(),
    });
    return PartialView();
}
```

```
0 references
public ActionResult SaveNewBalance(TransactionItemModel TransactionItem)
{
    if (ModelState.IsValid)
    {
        var UserId = _UserService.GetUserId();
        var UserName = GetUserName();
        var Quntity = _Context.TransactionItem.Where(x => x.ItemId == TransactionItem.ItemId && x.LatestStatusCode == 10);
        var HospitalId = GetHospitalId();
        TransactionItem TransactionI = new TransactionItem();

        TransactionI.ItemId = TransactionItem.ItemId;
        TransactionI.ItemExpireDate = TransactionItem.BloodingDate.Value.AddDays(45);
        TransactionI.AllayNumber = TransactionItem.AllayNumber;
        TransactionI.BagNumber = TransactionItem.BagNumber;
        TransactionI.UnitCostPrice = TransactionItem.UnitCostPrice;
        TransactionI.UnitSalePrice = TransactionItem.UnitSalePrice;
        TransactionI.LatestStatusCode = 10;
        TransactionI.LatestStatusName = "إضافة";
        TransactionI.CreatorName = UserName != null ? UserName : "0";
        TransactionI.IsActive = true;
        TransactionI.IsDeleted = false;
        TransactionI.HospitalId = HospitalId;
        TransactionI.TransactionEffectNameAr = "إضافة مباشرة";
        TransactionI.TransactionEffectNameEn = "Direct Addition";
        TransactionI.BenficaryNameAr = "المريض";
        TransactionI.BenficaryNameEn = "Patient";
        TransactionI.BloodingDate = TransactionItem.BloodingDate;
        TransactionI.TransactionDate = DateTime.Now;
        var item = _Context.Items.FirstOrDefault(x => x.Id == TransactionItem.ItemId);
        if (item.NameEn == "A+")...
        if (item.NameEn == "A-")...
        if (item.NameEn == "B+")...
        if (item.NameEn == "B-")...
        if (item.NameEn == "AB+")...
        if (item.NameEn == "AB-")...
        if (item.NameEn == "O+")...
        if (item.NameEn == "O-")...
        _Context.TransactionItem.Add(TransactionI);
        _Context.SaveChanges();
        return RedirectToAction("_AddBalance");
    }

    return RedirectToAction("_AddBalance");
}
```

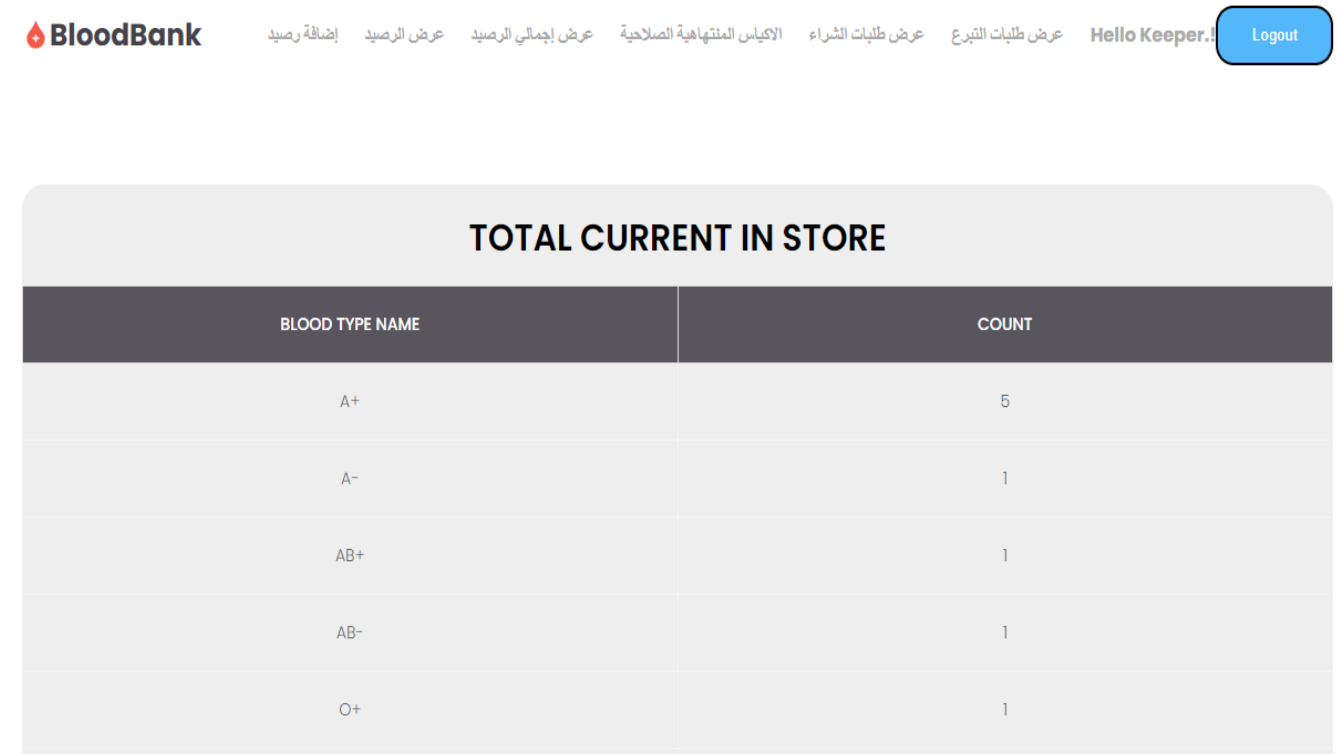
•View Current Balance:

CURRENT IN STORE

NAME HOSPITAL	BLOOD TYPE NAME	SALE PRICE	BLOODING DATE	ITEM EXPIRE DATE	DELETE
Keeper	AB+	600	12/13/2022 10:16:00 PM	1/6/2023 10:16:00 PM	
Keeper	AB-	600	12/12/2022 10:16:00 PM	1/7/2023 10:16:00 PM	
Keeper	O+	550	12/14/2022 10:16:00 PM	1/7/2023 10:16:00 PM	
Keeper	A+	590	12/16/2022 12:07:00 AM	1/6/2023 12:07:00 AM	

```
0 references
public ActionResult _ViewCurrentBalance()
{
    var HospitalId = GetHospitalId();
    var TransactionItem = _Context.TransactionItem.Include(s => s.Item).Include(s => s.Transaction.Hospital).Where(x => x.HospitalId == HospitalId &&
      x.ItemItemExpire == false && x.ItemItemExpire == false && x.IsPaid == false);
    var Transaction = TransactionItem.ToModel().ToList();

    return PartialView(Transaction);
}
```

•Total Blood Bags:




```
0 references
public ActionResult _TotalBloodBags()
{
    List<TransactionCountViewModel> TransactionCountList = new List<TransactionCountViewModel>();

    var HospitalId = GetHospitalId();
    var TransactionItem = _Context.TransactionItem.Include(s => s.Item).Include(s => s.Transaction.Hospital).Where(x => x.HospitalId == HospitalId &&
      x.ItemItemExpire == false && x.IsPaid == false).AsEnumerable();
    TransactionItem = TransactionItem.OrderBy(x => x.ItemId);
    foreach (var item in TransactionItem)
    {
        TransactionCountViewModel TransactionCount = new TransactionCountViewModel();
        var check = TransactionCountList.Where(x => x.ItemId == item.ItemId).FirstOrDefault();
        if (check == null)
        {
            TransactionCount.ItemName = item.Item.NameAr;
            TransactionCount.ItemId = item.ItemId;
            var Count = TransactionItem.Where(x => x.ItemId == item.ItemId).Count();
            TransactionCount.Count = Count;
            TransactionCountList.Add(TransactionCount);
        }
    }
    return PartialView(TransactionCountList);
}
```

•Show Expiration:

SHOW EXPIRATION

BLOOD TYPE NAME	SALE PRICE	ITEM EXPIRE DATE
O-	550	12/16/2022 4:55:00 PM

```
0 references
public ActionResult _ShowExpiration()
{
    ExpirationItem();
    List<TransactionExpirationViewModel> TransactionItemModel = new List<TransactionExpirationViewModel>();
    var HospitalId = GetHospitalId();
    var Date = DateTime.Now;
    var TransactionItem = _Context.TransactionItem.Include(s => s.Item).Include(s => s.Transaction.Hospital).Where(x => x.ItemExpireDate <= Date &&
      x.HospitalId == HospitalId).ToList();

    foreach (var item in TransactionItem)
    {
        TransactionExpirationViewModel TransactionModel = new TransactionExpirationViewModel();

        TransactionModel.ItemName = item.Item.NameAr;
        TransactionModel.ExpireDate = item.ItemExpireDate.ToString();
        TransactionModel.Price = item.UnitSalePrice;

        TransactionItemModel.Add(TransactionModel);
    }
    return PartialView(TransactionItemModel);
}
```

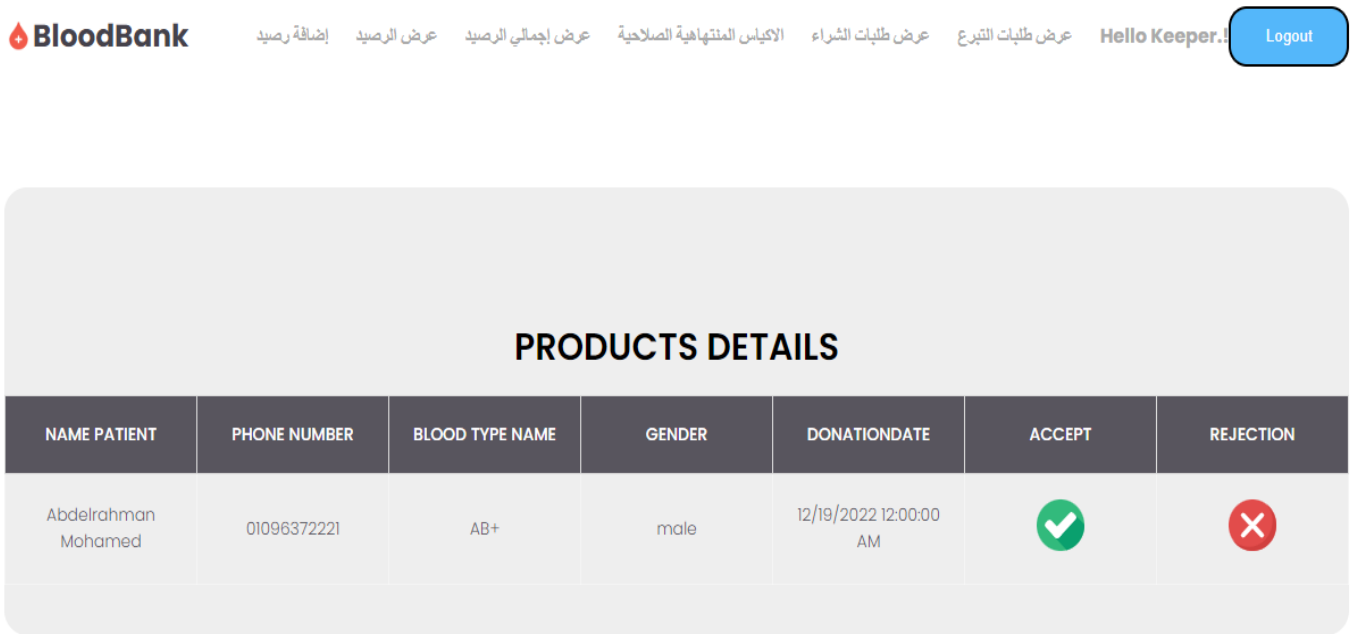
•All Products Details:

PRODUCTS DETAILS

NAME PATIENT	BLOOD TYPE NAME	QYT	SALE PRICE	RECEIVED DATE	ORDER STATUS	ACCEPT	REJECTION
Boda	B-	1	600	12/25/2022 9:37:36 AM	شراء		
Boda	B+	1	590	12/25/2022 9:37:36 AM	شراء		

```
2 references
public IActionResult _AllProductsDetails()
{
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    if (UserId.UserTypeId == 2)
    {
        List<RequestProductsDetails> RequestProductsDetailsList = new List<RequestProductsDetails>();

        var Donation = _Context.Donation.Include(x => x.UserData).Where(x => x.IsRollBack == false).ToList();
        Donation = Donation.Where(x => x.UserData.UserId == User).ToList();
        if (Donation.Count != 0)
        {
            var HospitalId = GetHospitalId();

            var Confirmation = _Context.Confirmation.Where(x => x.HospitalId == HospitalId && x.AcceptFromHospital == false && x.RejectionFromHospital ==
               false).ToList();
            foreach (var item in Confirmation)
            {
                RequestProductsDetails RequestProducts = new RequestProductsDetails();
                RequestProducts.Id = item.Id;
                RequestProducts.ItemName = item.ItemName;
                RequestProducts.Price = 0;
                RequestProducts.QYT = 1;
                RequestProducts.ReceivedDate = item.AddedTime.ToString();
                RequestProducts.Total = (int.Parse(item.Price) * Confirmation.Count());
                RequestProducts.PatientName = item.PatientName;
                var check = Confirmation.FirstOrDefault();
                if (check.Id == item.Id)
                {
                    RequestProducts.OrderStatus = "هدية لمتبرع";
                }
                else
                {
                    RequestProducts.OrderStatus = "شراء";
                }

                RequestProductsDetailsList.Add(RequestProducts);
            }
        }
        else...

        return PartialView("_AllProductsDetails", RequestProductsDetailsList);
    }

    return PartialView("_AllProductsDetails");
}
```

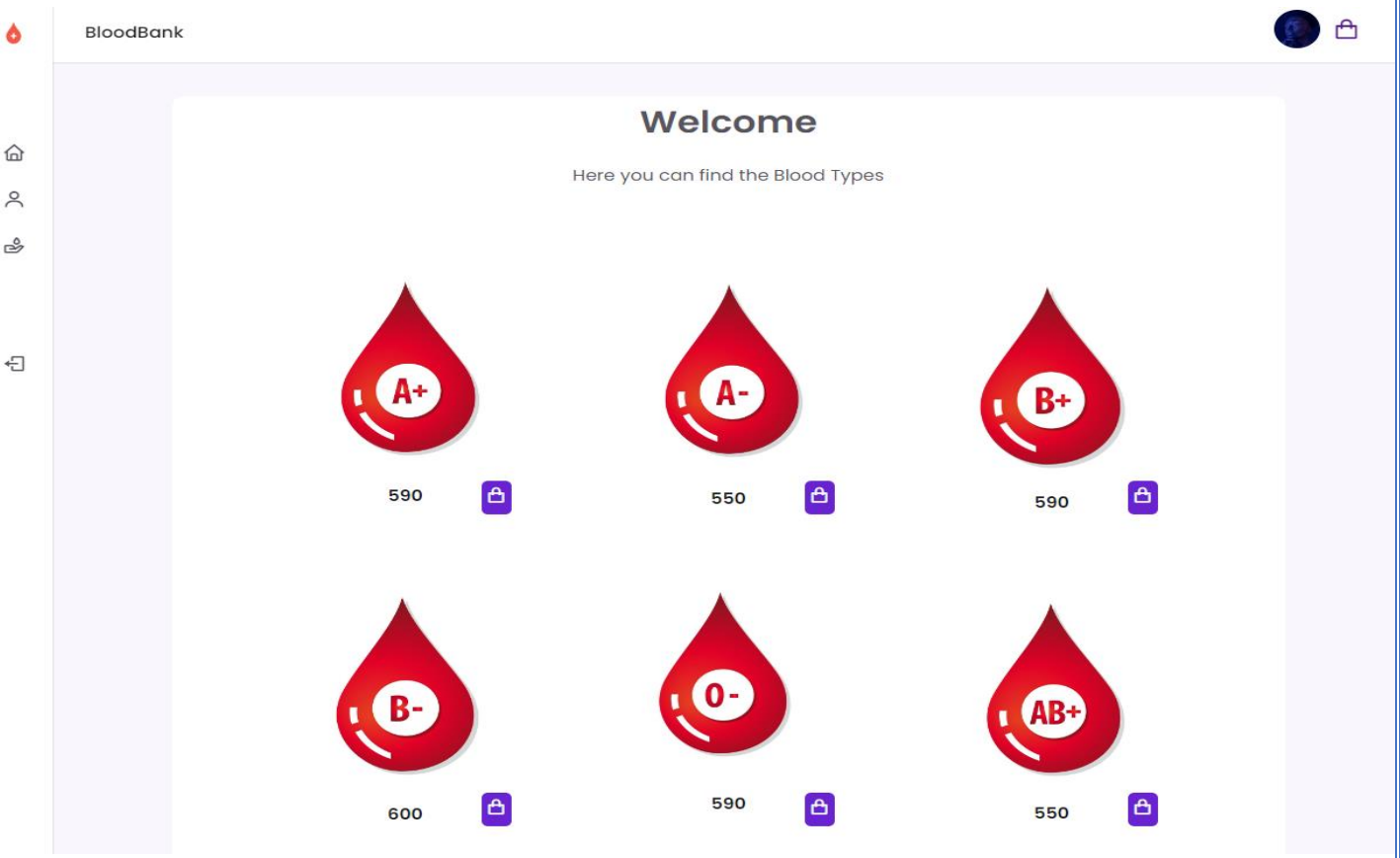
•View Donation Request:




```
public ActionResult Donation()
{
    return View();
}
2 references
public ActionResult _Donation()
{
    var Donation = _Context.Donation.Include(x => x.Hospital).Include(x => x.Item).Include(x => x.UserData).Where(x => x.Flag == false && x.IsDeleted ==
      false).ToList();
    return PartialView("_Donation", Donation);
}
```

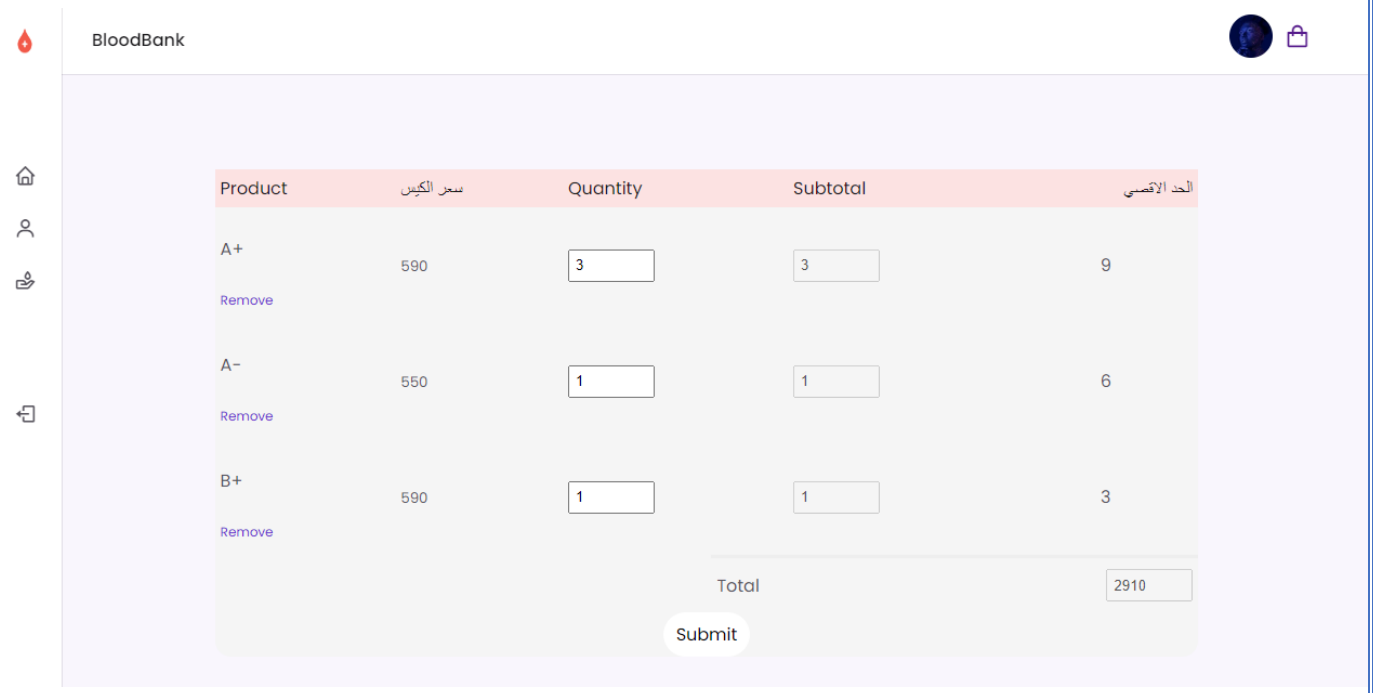
• Hospital:

• Shop:




```
0 references
public IActionResult Shop()
{
    ExpirationItem();
    List<TransactionItemModel> TransactionItemList = new List<TransactionItemModel>();
    var Trans = _Context.TransactionItem.Include(s => s.Item).Where(x => x.IsActive == true && x.IsDeleted == false && x.IsPaid == false &&
      x.ItemItemExpire == false).ToList();
    foreach (var item in Trans)
    {
        if (TransactionItemList.Any(x=>x.ItemId == item.ItemId))
        {}
        else
        {
            var itemz = _Context.Items.FirstOrDefault(s => s.Id == item.ItemId).ToModel();
            TransactionItemModel TransactionItem = new TransactionItemModel();
            TransactionItem.Item = itemz;
            TransactionItem.UnitSalePrice = item.UnitSalePrice;
            TransactionItem.Id = item.Id;
            TransactionItem.ItemId = item.ItemId;
            TransactionItem.IMG = item.IMG;

            TransactionItemList.Add(TransactionItem);
        }

    }
    return View(TransactionItemList);
}
```

• **Cart**:


BloodBank
Product
سعر الكيس
Quantity
Subtotal
الحد الاقصي
A+
590
3
3
9
Remove
A-
550
1
1
6
Remove
B+
590
1
1
3
Remove
Total
2910
Submit

```
public IActionResult _Cart()
{
    var User = _UserService.GetUserId();
    if (User != null)
    {
        var Donation = _Context.Donation.Include(x => x.UserData).Where(x=>x.IsRollBack == false).ToList();
        Donation = Donation.Where(x => x.UserData.UserId == User).ToList();
        if (Donation != null)
        {
            ViewBag.Massage = "لقد تبرعت بالدم من قبل و لهذا فيوجد كيس مجاني لك";
            List<CartViewModel> CartDetails = new List<CartViewModel>();
            var cart = _Context.CartDetails.Include(x => x.Items).Include(s => s.TransactionItem).Where(x => x.UserId == User && x.IsActive == true &&
              x.IsDeleted == false && x.IsPaid == false).ToList();
            float Total = 0;
            int HospitalId = 0;
            foreach (var item in cart)
            {
                int Max = 0;

                CartViewModel CartDetailsModel = new CartViewModel();
                CartDetailsModel.HospitalId = item.HospitalId;
                CartDetailsModel.ItemsId = item.ItemsId;
                CartDetailsModel.Id = item.Id;
                CartDetailsModel.QYT = item.QYT;
                CartDetailsModel.Items = item.Items;
                CartDetailsModel.TransactionItem = item.TransactionItem;
                HospitalId = item.HospitalId;

                CartDetailsModel.FlagMoreThanOneHospital = 2;
                Total = (Total + CartDetailsModel.TransactionItem.UnitSalePrice) * item.QYT;
                var Transaction = _Context.TransactionItem.Where(x => x.IsActive == true && x.IsDeleted == false && x.IsPaid == false && x.ItemId ==
                  item.ItemsId).ToList();

                foreach (var transaction in Transaction)
                {
                    if (item.ItemsId == transaction.ItemId)
                    {
                        Max++;
                    }
                }
                CartDetailsModel.MaxItems = Max;

                CartDetails.Add(CartDetailsModel);
            }
            if (cart.Count != 0)
            {
                var cartMaster = cart.FirstOrDefault().CartMasterId;
                ViewBag.Total = Total;
                ViewBag.CartMaster = cartMaster;
```


```
var CartMaster = $("#CartMaster").val();
if (Flag == 1) {
    swal("يوجد اكياس دم من اكثر من مستشفي", {
        buttons: {
            cancel: "cancel",
            catch: {
                text: "تأكيد باي حال !",
                value: "catch",
            },
            فحص: true,
        },
    })
    .then((value) => {
        switch (value) {
            case "فحص":
                $.ajax({
                    type: 'get',
                    url: '@Url.Action("_Check", "User")',
                    //dataType: 'json',
                    data: {
                        CartMaster: CartMaster,
                    },
                    success: function (data) {
                        debugger;
                        $("#CartFinally").html('');
                        $("#CartFinally").html(data);
                    },
                    error: function (XMLHttpRequest, textStatus, errorThrown) {
                    debugger;
                    }
                });
                //swal("Pikachu fainted! You gained 500 XP!");
                break;
            case "catch":
                $.ajax({
                    type: 'POST',
                    url: '@Url.Action("Confirm", "User")',
                    //dataType: 'json',
                    data: {
                        CartMaster: CartMaster,
                    },
                    success: function (data) {
```

- **Check**:

```
0 references
public ActionResult _Check(int CartMaster)
{
    var User = _UserService.GetUserId();
    if (User != null)
    { ≤ 144,611ms elapsed
        List<CheckCartViewModel> CheckCartViewModel = new List<CheckCartViewModel>();

        var DonationList = _Context.Donation.Include(x => x.UserData).Where(x => x.IsRollBack == false).ToList();
        var Donation = DonationList.Where(x => x.UserData.UserId == User && x.IsRollBack == false).FirstOrDefault();
        if (Donation != null)
        {
            List<CartDetails> CartDetails = new List<CartDetails>();
            var cart = _Context.CartDetails.Include(x => x.TransactionItem).Include(x => x.Items).Include(x => x.Hospital).Where(x => x.CartMasterId ==
              CartMaster && x.IsActive == true && x.IsDeleted == false && x.IsPaid == false).ToList();
            ViewBag.CartMaster = cart.FirstOrDefault().CartMasterId;

            foreach (var item in cart)
            {
                var Transaction = _Context.TransactionItem.Where(x => x.IsActive == true && x.IsDeleted == false && x.ItemId == item.ItemsId && x.IsPaid
                  == false).ToList();

                int Max = 0;
                foreach (var transaction in Transaction)
                {
                    CheckCartViewModel CheckCart = new CheckCartViewModel();
                    CheckCart.Id = item.Id;
                    //CheckCart.HospitalName = item.Hospital.NameAr;
                    CheckCart.ItemName = item.Items.NameAr;
                    if (Donation.ItemId == transaction.ItemId && Donation.HospitalId == transaction.HospitalId)
                    {
                        CheckCart.Price = "هذا الكيس هدية لكونك متبرع لهذه المستشفي من قبل";
                        CheckCart.QTY = "1";
                        if (item.ItemsId == transaction.ItemId)
                        {
                            Max++;
                            var Hospital = _Context.Hospital.FirstOrDefault(x => x.Id == transaction.HospitalId);
                            CheckCart.HospitalName = Hospital.NameAr;
                            CheckCart.HospitalAddress = Hospital.Address;
                            var QTYInHospital = Transaction.Where(x => x.ItemId == item.ItemsId && x.HospitalId == transaction.HospitalId).Count();
                            CheckCart.QTYInHospital = QTYInHospital.ToString();

                            var Check = CheckCartViewModel.Where(x => x.HospitalName == CheckCart.HospitalName && x.ItemName ==
                              item.Items.NameAr).FirstOrDefault();
                            if (Check == null)
                            {
                                CheckCartViewModel.Add(CheckCart);
                            }
```

• **Remove**:

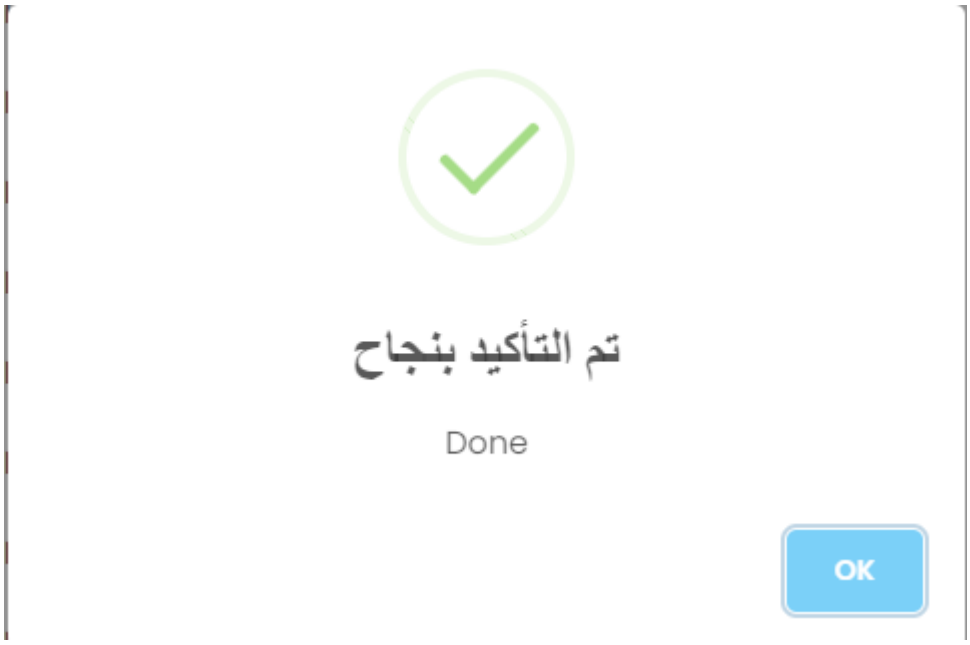
```
''
$("#CartGrid").on('click', '.DeleteRow', function () {
    debugger;
    $(this).parent().parent().remove();
});
''
```

• **Delete Item:**

```
function DeleteItem(id) {
    debugger;
    $.ajax({
        url: '@Url.Action("Delete", "User")',
        data: { Id: id },
        contentType: 'application/json; charset=utf-8',
        success: function (data) {
            debugger;
            $("#CartItemGrid").html('');
            $("#CartItemGrid").html(data);
        },
        error: function (XMLHttpRequest, textStatus, errorThrown) { }
    });
};
```

```
0 references
public ActionResult Delete(int id)
{
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    if (UserId != null)
    {
        var Cart = _Context.CartDetails.FirstOrDefault(x => x.Id == id && x.UserId == UserId.Id && x.IsDeleted == false);
        Cart.IsDeleted = true;
        Cart.IsActive = false;

        _Context.Entry(Cart).CurrentValues.SetValues(Cart);
        _Context.SaveChanges();

        return RedirectToAction("_Cart", "User");
    }
    else
    {
        return Json("Error");
    }
}
```

•Tostar:

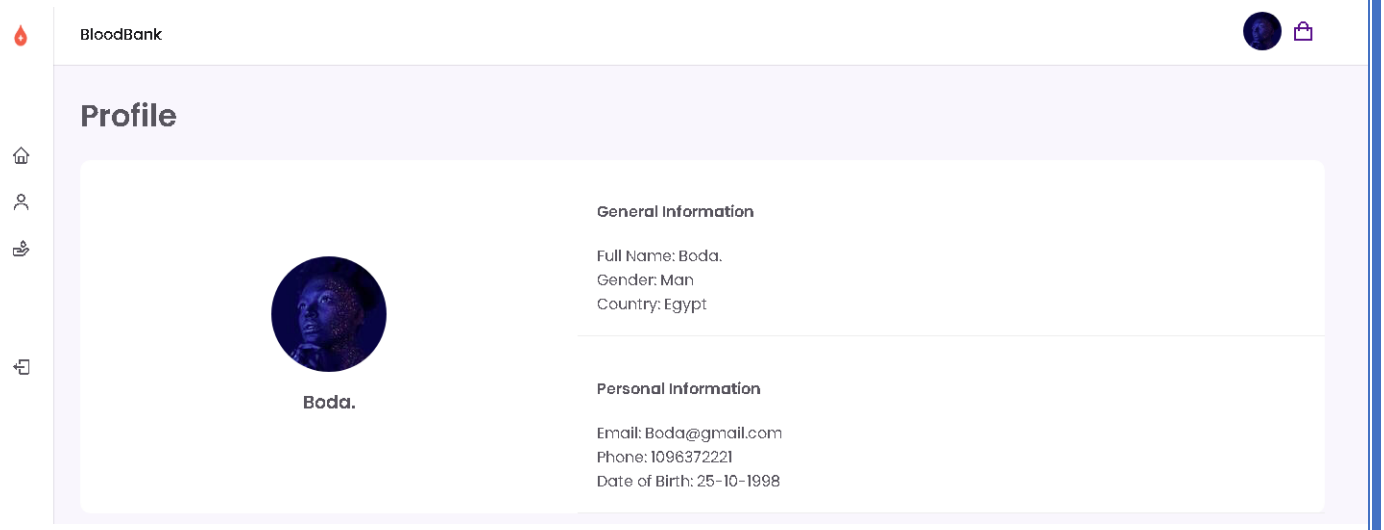



```
success: function (data) {
    debugger;
        swal("تم التأكيد بنجاح", "Done", "success");
        $("#CartFinally").html('');
        $("#CartFinally").html(data);
},
```

•**follow up with the order**:

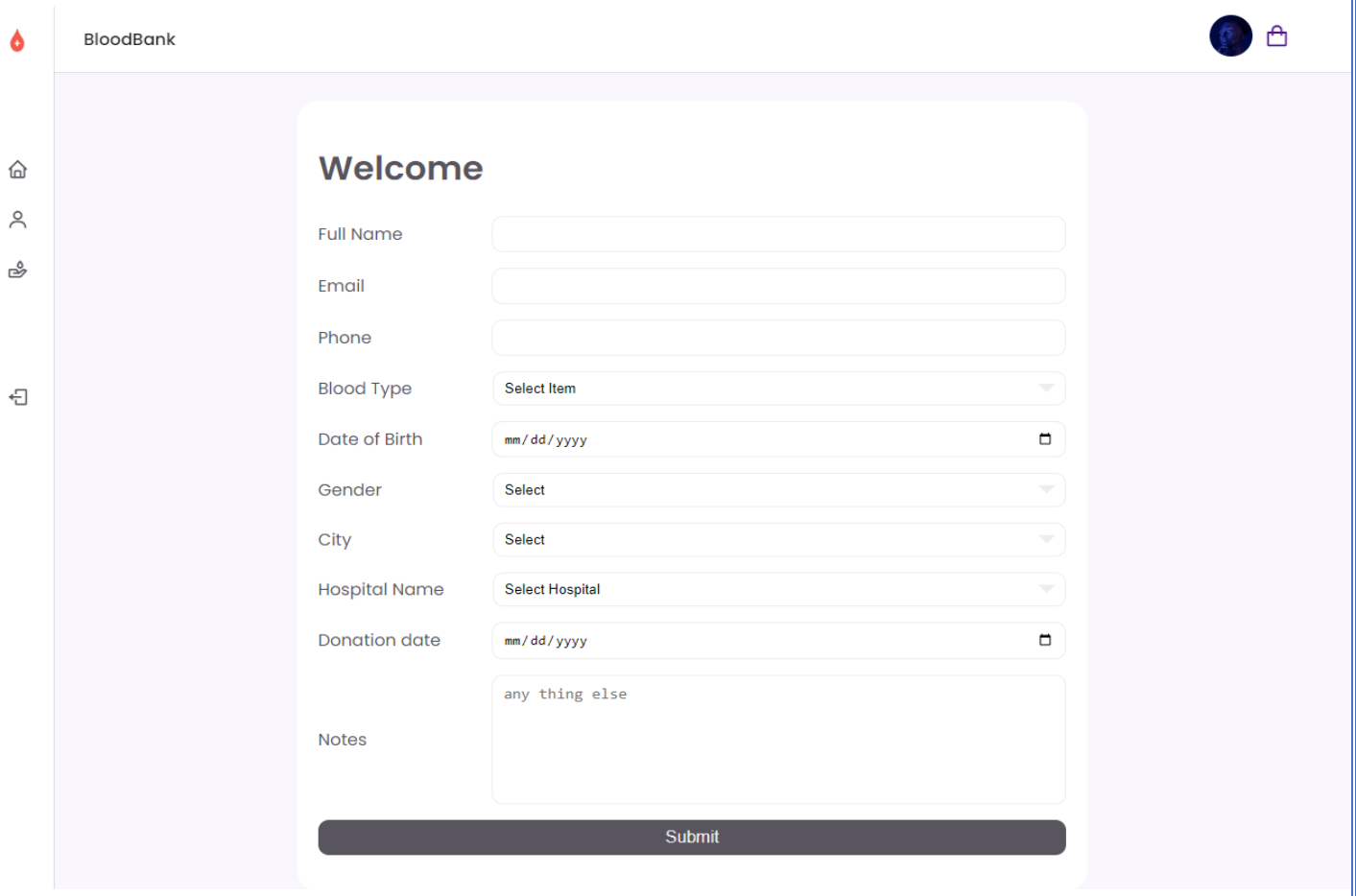

```
[HttpPost]
0 references
public ActionResult Confirm(int CartMaster)
{  ≤ 222,708ms elapsed
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    if (UserId != null)
    {
        var Donation = _Context.Donation.Include(x => x.UserData).Where(x => x.IsRollBack == false).ToList();
        Donation = Donation.Where(x => x.UserData.UserId == User).ToList();
        if (Donation != null)
        {
            CartViewModel CartViewModel = new CartViewModel();

            var Cart = _Context.CartDetails.Include(x => x.Hospital).Include(x => x.Items).Where(x => x.UserId == UserId.Id && x.CartMasterId ==
              CartMaster && x.IsDeleted == false && x.IsPaid == false).ToList();

            foreach (var item in Cart)
            {
                item.IsPaid = true;

                _Context.Entry(item).CurrentValues.SetValues(item);
                _Context.SaveChanges();

                Donation donation = new Donation();
                donation.Flag = true;
                donation.IsRollBack = true;

                _Context.Entry(donation).CurrentValues.SetValues(donation);
                _Context.SaveChanges();
                var transactionTset = _Context.TransactionItem.OrderByDescending(x => x.ItemExpireDate).Where(x => x.ItemId == item.ItemsId && x.IsPaid
                  == false && x.ItemItemExpire == false).ToList();
                for (int i = 0; i < item.QYT; i++)
                {
                    var transaction = transactionTset.Skip(i).FirstOrDefault();
                    transaction.IsPaid = true;
                    _Context.Entry(transaction).CurrentValues.SetValues(transaction);
                    _Context.SaveChanges();
                }
                ConfirmationRequest(item.Id);
            }
        }
        else...
        return _Confirmation();
    }
```

•**Profile**:




```
public IActionResult Profile()
{
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    InfoUserModel InfoUser = new InfoUserModel();
    if (UserId != null)
    {
        var UserData = _Context.UserData.Where(x => x.UserId == User).FirstOrDefault();
        if (UserData != null)
        {
            InfoUser.username = UserId.UserName;
            InfoUser.Gender = UserData.Gender;
            InfoUser.Country = UserData.Country;
            InfoUser.Email = UserId.Email;
            InfoUser.DateOfBirth = UserData.DateOfBirth;
            InfoUser.Phonenumber = Convert.ToInt64(UserId.PhoneNumber);
        }
        else...
    }

    return View(InfoUser);
}
```

•**Donate**:



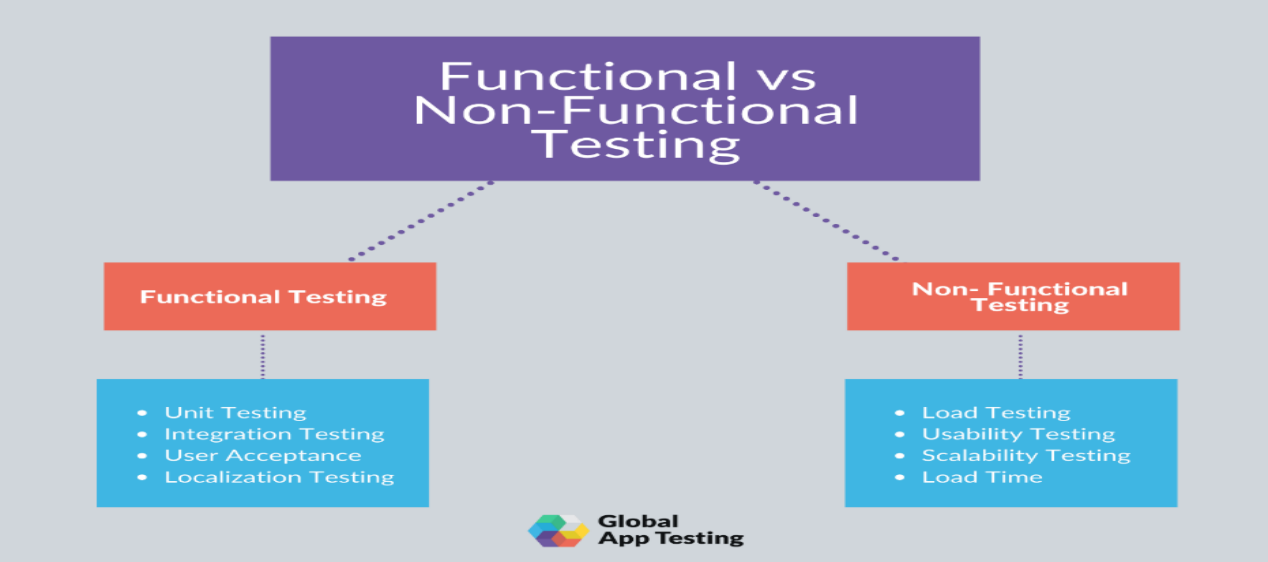

```
0 references
public IActionResult Donate()
{
    ViewBag.Items = _Context.Items.Select(x => new SelectListItem()
    {
        Text = x.NameAr,
        Value = x.Id.ToString(),
    });
    ViewBag.Hospital = _Context.Hospital.Select(x => new SelectListItem()
    {
        Text = x.NameAr,
        Value = x.Id.ToString(),
    });
    return View();
}
```

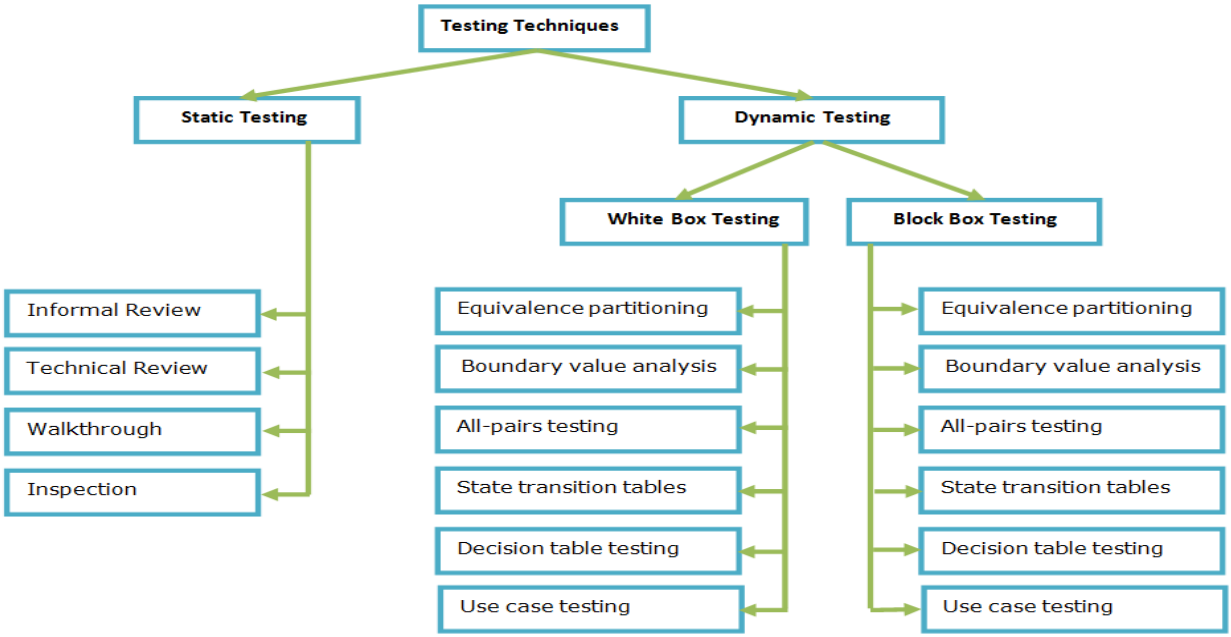
•**Submit Donate Request**:

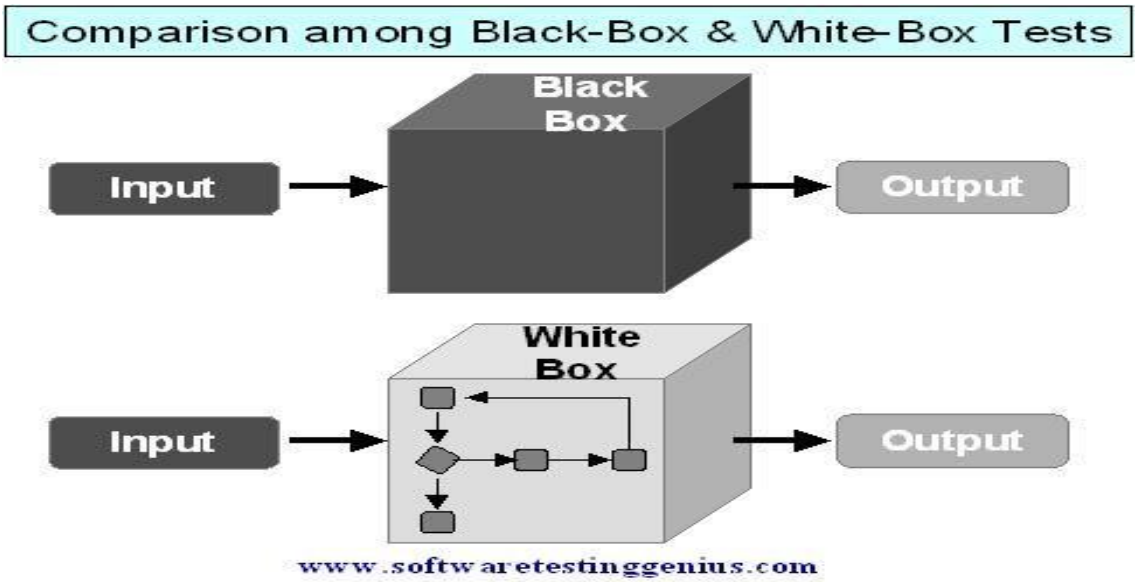
```
[HttpPost]
0 references
public ActionResult SaveDonate(DonateViewModel Donate)
{
    var User = _UserService.GetUserId();
    var UserId = _Context.USer.FirstOrDefault(s => s.Id == User);
    if (UserId != null)
    {
        Donation Donation = new Donation();
        UserData UserData = new UserData();
        UserData.FullName = Donate.FullName;
        UserData.PhoneNumber = Donate.PhoneNumber;
        UserData.DateOfBirth = Donate.DateOfBirth;
        UserData.Gender = Donate.Gender;
        UserData.CityAndArea = Donate.CityAndArea;
        UserData.Massage = Donate.Massage;
        UserData.UserId = User;
        UserData.UserType = "تبرع";
        _Context.UserData.Add(UserData);
        _Context.SaveChanges();

        Donation.ItemId = Donate.ItemId;
        Donation.DonationDate = Donate.DonationDate;
        Donation.UserDataId = UserData.Id;
        Donation.HospitalId = Donate.HospitalId;
        _Context.Donation.Add(Donation);
        _Context.SaveChanges();
    }

    return Json("Done");
}
```

```
function Save() {
    debugger;
    var Name = $("#Name").val();
    var Email = $("#Email").val();
    var Phone = $("#Phone").val();
    var ItemId = $("#ItemId").val();
    var DateOfBirth = $("#DateOfBirth").val();
    var DonationDate = $("#DonationOfBirth").val();
    var Gender = $("#Gender").val();
    var City = $("#City").val();
    var Notes = $("#Notes").val();
    var HospitalId = $("#HospitalId").val();
    var jDonate = {
        FullName: Name,
        Email: Email,
        PhoneNumber: Phone,
        ItemId: ItemId,
        DateOfBirth: DateOfBirth,
        Gender: Gender,
        CityAndArea: City,
        Massage: Notes,
        DonationDate: DonationDate,
        HospitalId: HospitalId,
    }
    $.ajax({
        type: 'POST',
        url: '@Url.Action("SaveDonate", "User")',
        dataType: 'json',
        //contentType: 'application/json; charset=utf-8',
        data: jDonate,
        success: function (data) {
            debugger;
            if (data == "Done") {
                $("#Donate").html('');
                $("#Name").html('');
                $("#Email").html('');
                $("#Phone").html('');
                $("#ItemId").html('');
                $("#DateOfBirth").html('');
                $("#Gender").html('');
                $("#City").html('');
                $("#Notes").html('');
            }
        },
        error: function (XMLHttpRequest, textStatus, errorThrown) {
            debugger;
        }
    });
}
```

Chapter 5: Testing







5.1 Unit Testing

Testing of individual items (e.g. modules, programs, objects, classes, etc.) As part of the coding phase, in isolation from other development items and the system as a whole

5.2 Integrated Testing

Testing is performed to expose defects in the interfaces and in the interactions between integrated components or systems.

5.3 System Testing

Testing a system behavior after it is fully integrated, as a whole when development is finished, hence, the system can be tested as a complete entity.

5.4 Regression Testing

To check older functionality after integrating new functionality.

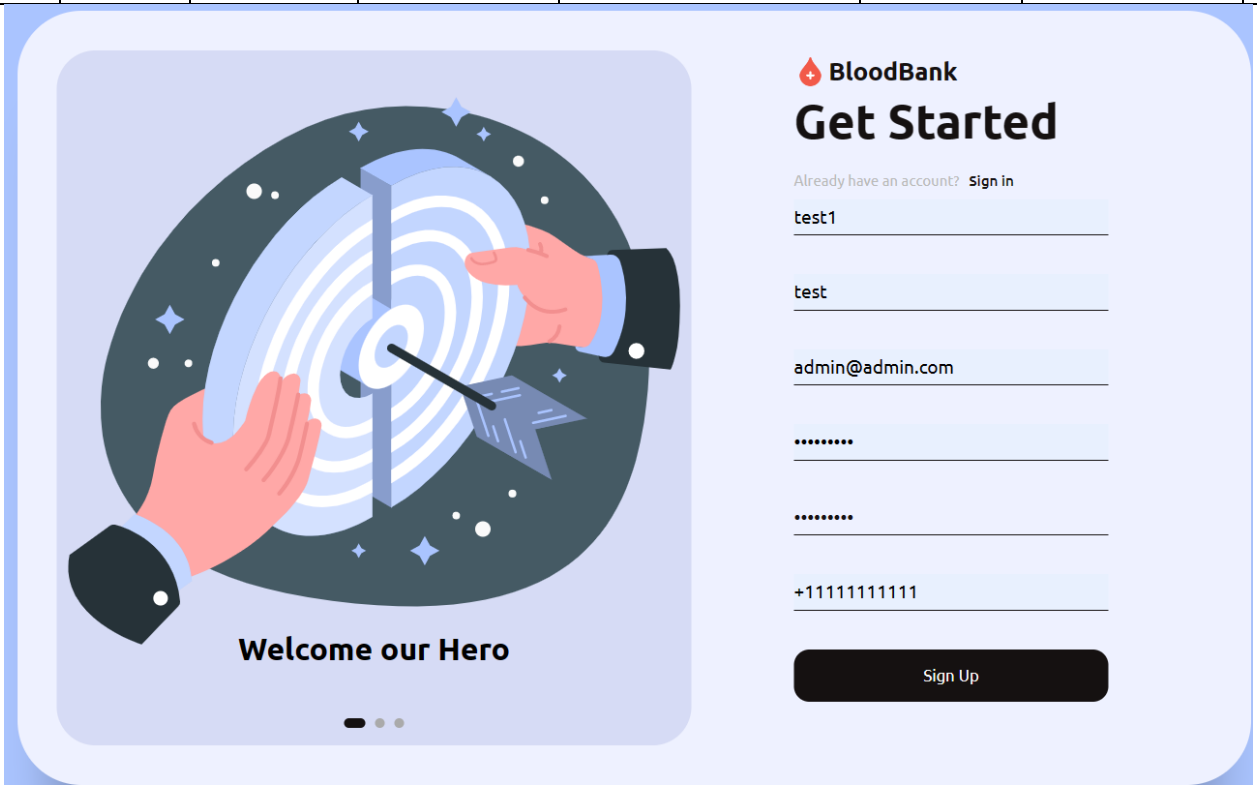
5.5 Acceptance Testing

Testing to evaluate the site compliance to ensure that it is ready to be deployed into the business, operational, or production environment.

User Register Test Case:

Test case ID	TC Type	TC Title	TC Prerequisites	Steps	Expected result	Actual result	Status
1	Positive	Check registering a new account	1-Open the website	1-Insert first name 2-Insert last name 3-Insert email 4-Insert password 5-Insert confirm password 6-Insert phone number	The user should be registered successfully	The user should be registered successfully	Pass
2	Negative	Check registering a new account without entering first name	1-Open the website	1-Keep first name Null 2-Insert last name 3-Insert email 4-Insert password 5-Insert confirm password 6-Insert phone number	The user shouldn't be registered and a validation message appears	The user shouldn't be registered and a validation message appears	Pass
3	Negative	Check registering a new account without entering last name	1-Open the website	1-Insert first name 2-Keep last name Null 3-Insert email 4-Insert password 5-Insert confirm password 6-Insert phone number	The user shouldn't be registered and a validation message appears	The user shouldn't be registered and a validation message appears	Pass
4	Negative	Check register with an invalid E-mail	1-Open the website	1-Insert first name 2- Insert last name 3-Insert invalid email 4-Insert password 5-Insert confirm password 6-Insert phone number	The user shouldn't be registered and a validation message appears	The user shouldn't be registered and a validation message appears	Pass
5	Negative	Check register with Mismatch password with confirm password	1-Open the website	1-Insert first name 2- Insert last name 3-Insert email 4-Insert password 5-Insert confirm password not equal the password 6-Insert phone number	The user shouldn't be registered and a validation message appears	The user shouldn't be registered and a validation message appears	Pass

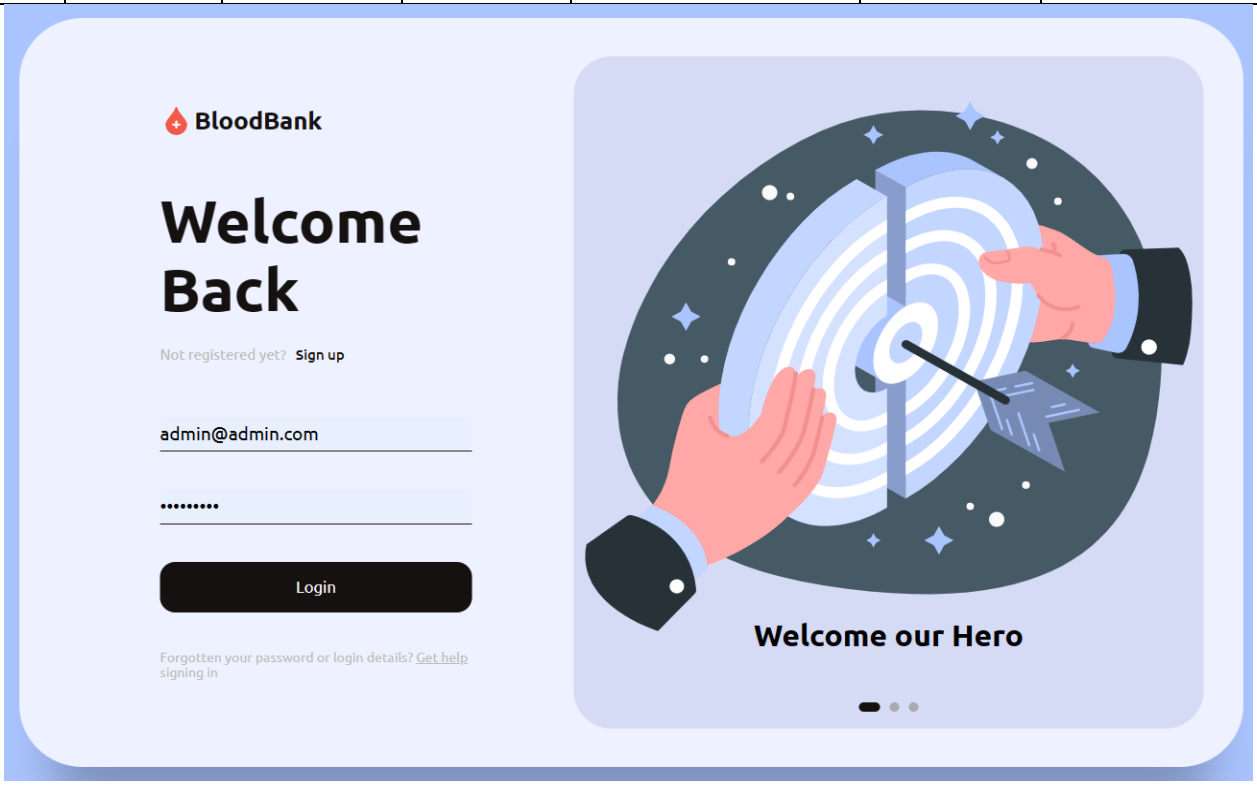
6	Negative	Check register with an invalid Phone number	1-Open the website	1-Insert first name 2- Insert last name 3-Insert email 4-Insert password 5-Insert confirm password 6-Insert Invalid phone number	The user shouldn't be registered and a validation message appears	The user shouldn't be registered and a validation message appears	Pass



User Login Test Case:

7	Positive	Login with a valid username and Password	1-Open the website 2- User should be registered	1-Insert valid username and Valid password 2- Click on Login button	User should be login successfully	User should be login successfully	Pass
8	Negative	Login with an invalid username and valid Password	1-Open the website	1-Insert invalid username and Valid password 2- Click on Login button	Validation message should be appear	Validation message should be appear	Pass

8	Negative	Login with a valid username and invalid Password	1-Open the website	1-Insert invalid username and Valid password 2- Click on Login button	Validation message should be appear	Validation message should be appear	Pass
9	Negative	Keep all fields empty	1-Open the website	Click on Login button	Validation message should be appear	Validation message should be appear	Pass

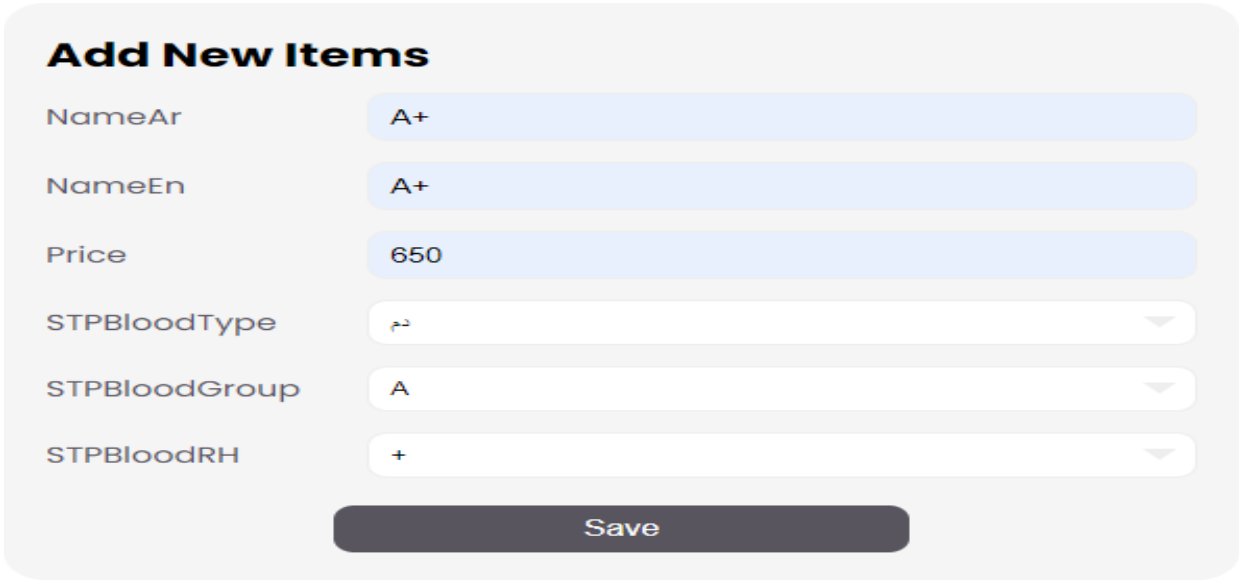


Add Hospital Test Case:

10	Positive	Add hospital	1-user should be login	1-Click on add hospital button 2-Insert Email 3-Insert NameAr 4-Insert NameEn 5-Insert Address 6-Click on Save button 7-Logout 8-Login with hospital email	Hospital should be added successfully	Hospital should be added successfully	Pass

Add blood Bag:

11	Positive	Add blood Bag	1-user should be login	1-Click on add blood bag 2-Insert NameAr 3-Insert NameEn 4-Insert price 5-Select STP Blood Type 6-Select STP Blood Group 7-Select STP Blood RH 8-Click on Save button	Blood bag should be added successfully	Blood bag should be added successfully	Pass



5.6 Test Automation

What is Automation Testing?

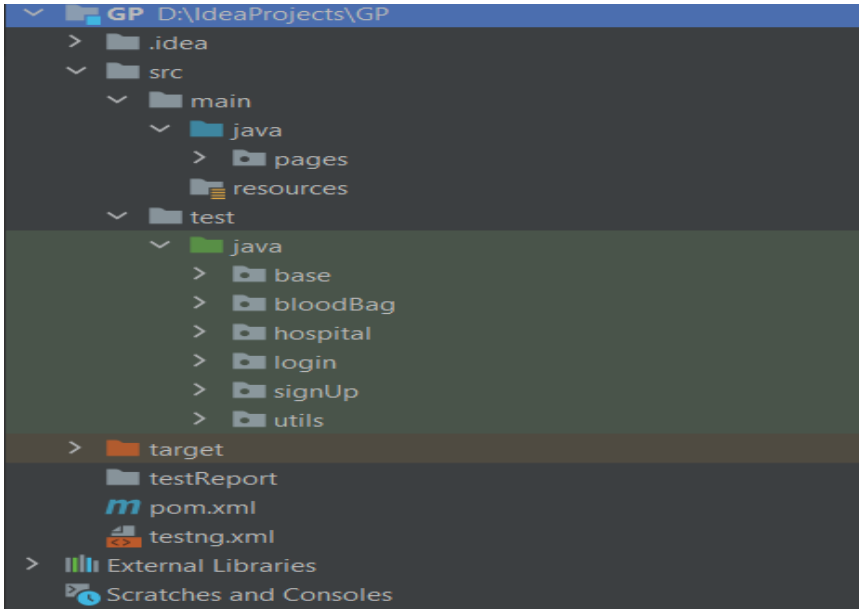
Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results, and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

Successive development cycles will require the execution of the same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and replay it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

How did we build these scripts?

We use Selenium WebDriver tool with Java to automate these scripts and use TestNG framework to handle my test suites, cases, preconditions, postconditions, and Page Object Model Design Pattern to follow the pattern Main directory and Test directory (Maven Project).



Pom.xml the heart of the project.

In automation projects, we use maven project not java to handle some issue we find it on java project the main one is pom.xml file, this file we put in it the dependencies and the project download it automatically.

We use in this project Selenium, TestNG, and Web Driver Manager

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.7.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>5.3.1</version>
</dependency>
```

Main (Framework) Directory :

In main directory we create a page as a package and put in it any page we locate all elements we want to make actions on it and its actions like insert, getText, click, select from drop-down list, and so on.

The page we create is Signup page, Login page, Home page, Blood Bank page, Add Hospital page, and Add Blood Bag page.

```java
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

17 usages
public class LoginPage {
    8 usages
    WebDriver driver;

    2 usages
    public LoginPage(WebDriver driver){
        this.driver = driver;
    }
    1 usage
    private WebElement signUpButton(){
        return driver.findElement(By.xpath( xpathExpression: "//a[contains(text(),'Sign up')]"));
    }

    1 usage
    private WebElement emailField() { return driver.findElement(By.xpath( xpathExpression: "(//input[@placeholder='Email'])[1]")); }
    1 usage
    private WebElement passwordField(){
        return driver.findElement(By.xpath( xpathExpression: "(//input[@id='password'])[1]"));
    }
```

```java
private WebElement errorValidationMessage(){
    return driver.findElement(By.xpath( xpathExpression: "(//p[@style='color:red'])"));
}

2 usages
public void insertEmail(String email) { emailField().sendKeys(email); }
2 usages
public void insertPassword(String password) { passwordField().sendKeys(password); }
3 usages
public BloodBankPage clickOnLoginButton(){
    loginButton().click();
    return new BloodBankPage(driver);
}
1 usage
public String getValidationMessage() { return errorValidationMessage().getText(); }
5 usages
public boolean loginButtonIsVisible(){
    return loginButton().isDisplayed();
}
4 usages
public SignUpPage clickOnSignUpButton(){
    signUpButton().click();
    return new SignUpPage(driver);
}
```

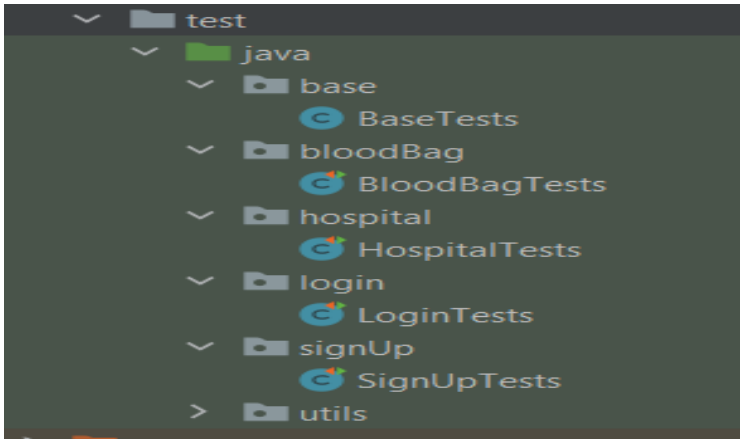
Test Directory:

In test directory we the main page (base), in base we put the preconditions and postconditions (before and after method and class)

In preconditions, we make a setup for our driver and handle all we want like maximize and implicate wait and so on.
In postconditions, we make a teardown to close all the browsers' drivers to save the memory.

```
public class BaseTests {
    6 usages
    WebDriver driver ;
    8 usages
    protected HomePage homePage;
    protected static ExtentReports extentReports;
    protected static ExtentTest logger;

    @BeforeClass
    public void setUp() throws Exception {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        goHome();
    }
    1 usage
    @BeforeMethod
    public void goHome() throws Exception {
//          logger = extentReports.startTest(method.getName());
        driver.get("http://bodamohammed-001-site1.itempurl.com/");
        homePage = new HomePage(driver);
//          ScreenRecorderUtil.startRecord(method.getName());
    }

    @AfterClass
    public void tearDown() { driver.quit(); }
}
```

The other directory is its own tests like Login Tests, Signup Tests, Hospital Tests, and Blood Bag Tests.



Login Tests :

```java
package login;

import base.BaseTests;
import org.openqa.selenium.WebDriver;
import org.testng.annotations.Test;
import pages.BloodBankPage;
import pages.LoginPage;
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

5 usages  2 inheritors
public class LoginTests extends BaseTests {

//     WebDriver driver;
    3 usages
    protected BloodBankPage bloodBankPage1;

    2 usages
    @Test
    public void testSuccessfulLogin(){
        String username = "admin@admin.com";
        String password = "AAss1212*";
        LoginPage loginPage = homePage.clickOnSignInButton();
        loginPage.insertEmail(username);
        loginPage.insertPassword(password);
        BloodBankPage bloodBankPage = loginPage.clickOnLoginButton();
        assertTrue(bloodBankPage.getHelloMassage().contains("Hello"));
        bloodBankPage1 = bloodBankPage;
    }
```

```java
@Test
public void testInvalidCredentials(){
    String username = "admin@admin.com";
    String password = "AAss1212";
    LoginPage loginPage = homePage.clickOnSignInButton();
    loginPage.insertEmail(username);
    loginPage.insertPassword(password);
    loginPage.clickOnLoginButton();
    String validationMessage = loginPage.getValidationMessage();
    assertEquals(validationMessage, expected: "User Not Found");
}
@Test
public void testEmptyFields(){
    LoginPage loginPage = homePage.clickOnSignInButton();
    loginPage.clickOnLoginButton();
    assertTrue(loginPage.loginButtonIsVisible());
}
```

Signup Tests:

```
package signUp;

import base.BaseTests;
import org.testng.annotations.Test;
import pages.LoginPage;
import pages.SignUpPage;
import java.util.Random;
import static org.testng.Assert.assertTrue;

1 usage
public class SignUpTests extends BaseTests {

    @Test
    public void testSuccessfulSignUp(){
        Random random = new Random();
        int randomNumber =random.nextInt();
        LoginPage loginPage = homePage.clickOnSignInButton();
        SignUpPage signUpPage = loginPage.clickOnSignUpButton();
        signUpPage.insertFirstName("Omar");
        signUpPage.insertLastName("Hani");
        signUpPage.insertEmail("Test@Test" + randomNumber + ".Test");
        signUpPage.insertPassword("Testtest123!");
        signUpPage.insertConfirmPassword("Testtest123!");
        signUpPage.insertPhoneNumber("123456789");
        LoginPage loginPage1 = signUpPage.clickOnSignUpButton();
        assertTrue(loginPage1.loginButtonIsVisible());
    }
```

```
@Test
public void testValidationMessage(){
    LoginPage loginPage = homePage.clickOnSignInButton();
    SignUpPage signUpPage = loginPage.clickOnSignUpButton();
    LoginPage loginPage1 = signUpPage.clickOnSignUpButton();
    assertTrue(loginPage1.loginButtonIsVisi
}
@Test
public void testPasswordNotEqualConfirmPassword(){
    Random random = new Random();
    int randomNumber =random.nextInt();
    LoginPage loginPage = homePage.clickOnSignInButton();
    SignUpPage signUpPage = loginPage.clickOnSignUpButton();
    signUpPage.insertFirstName("Omar");
    signUpPage.insertLastName("Hani");
    signUpPage.insertEmail("Test@Test" + randomNumber + ".Test");
    signUpPage.insertPassword("Testtest123!");
    signUpPage.insertConfirmPassword("Testtest123");
    signUpPage.insertPhoneNumber("123456789");
    LoginPage loginPage1 = signUpPage.clickOnSignUpButton();
    assertTrue(loginPage1.loginButtonIsVisible());
}
```

pages.SignUpPage
public LoginPage clickOnSignUpButton()
GP

Hospital Tests:

```java
package hospital;

import login.LoginTests;
import org.testng.annotations.Test;
import pages.AddHospitalPage;

import static org.testng.Assert.assertTrue;

1 usage
public class HospitalTests extends LoginTests {

    @Test
    public void testAddHospitalSuccessfully(){
        testSuccessfulLogin();
        AddHospitalPage addHospitalPage = bloodBankPage1.clickOnAddHospitalButton();
        addHospitalPage.insertEmail("test@test.test");
        addHospitalPage.insertNameAr("testttt");
        addHospitalPage.insertNameEn("anotherTest");
        addHospitalPage.insertAddress("15 May");
        addHospitalPage.clickOnSaveButton();
        assertTrue(addHospitalPage.buttonIsDisplayed());
    }
}
```

Blood Bag Tests:

```java
import login.LoginTests;
import org.testng.annotations.Test;
import pages.AddBloodBagPage;
import static org.testng.Assert.assertTrue;

1 usage
public class BloodBagTests extends LoginTests {

    @Test
    public void testAddBloodBag(){
        testSuccessfulLogin();
        AddBloodBagPage addBloodBagPage  = bloodBankPage1.clickOnAddBloodBagButton();
        addBloodBagPage.insertNameAr("عمر محمد هاني");
        addBloodBagPage.insertNameEn("Omar Mohamed Hani");
        addBloodBagPage.insertPrice("800");
        addBloodBagPage.selectBloodType( value: "دم");
        addBloodBagPage.selectBloodGroup( value: "A");
        addBloodBagPage.selectBloodRH( value: "+");
        addBloodBagPage.clickOnSaveButton();
        assertTrue(addBloodBagPage.getAddNewItemText());
    }
}
```

Chapter 6: Results and Discussion

6.1 Result:

we developed an efficient and active application that allow the interaction between many users who need a necessary blood transformation and those who can help them to get the specific type of blood that they need and convert our application from normal one to a platform and interactive app to help hospitals and blood administrations in their job too By this there are some steps that each one of them should follow. First the user can sign up as a donor or blood receiver then he can log in into the system and start searching for blood type or user who carry the same type that he needs and he is able to chat him too.

6.1.1 Expected result

First of all, we aim to help emergency situation and serious situations and operation that always need blood transformation of blood and .so we provide some tools and steps that make it easy for users to find the best donor who can help them There is a blood donation test and it aims to analysis people to two layers. if they are donors or receivers expected features and technology to make the application effective:

-the user can sign up and log in with the (phone number -name- email and password)

- the app displays the available users that carry the same blood type and the same location

-the app displays a view on the user profile so you can know the location and blood type

-the app displays a donation test for users to make sure that he is healthy and impatient enough to donate

6.1.2 The actual results

the actual results are not different from the expected results, but they also show how the admin and the users interact with the system to provide a complete help and benefits

1- the admin let the users sign up and logging in as they may be donors or receivers then he checks their ability to donate from the donation test which is provided in the application

2- the user can sign up and log in with the (phone number -name- email and password).

3-the application display a category of features that the users can use to find the same blood type as he wanted

4- those features are he can even search for user's location and profile by entering the location or the area he would search in

6.2 Discussion:

We manage to add all the techniques to help user and make it easy for him except the tool of call the users when you need to call them …but we add new features like the search field by choosing specific location user can also change his image and change personal information

Chapter 7: Conclusion

Technology is introducing new innovations day by day, thus reducing the time required to do things. The proposed system can be used to reduce the time required to deliver required blood to the needy in cases of emergency. The application can be used by people interested in donating their blood by locating their nearest blood bank. The web application provides a way of communication and synchronization between the hospitals and the blood banks. It also provides them with the facility of communicating with nearby donors in an emergency. The database is a vital aspect of the system. The database of the hospitals and the blood banks must be checked for consistency on regular basis for smooth working of the system.

Blood is the most important component of a person and in many cases, surgeries require a lot of blood like the process of giving birth and open heart, And we have a lot of blood types like O, A, B, etc. And since we have many types of blood required, they cannot overlap in the donation.

So we found a problem, and we want to solve it with a website to handle it automatically by making communication between the customer and the donor. To find out each other.

Based on the results, this study concluded that the online blood bank management system is much better than the manual system. The findings showed that respondents prefer to use an online blood bank management system rather than the manual system because it offers many advantages and benefits that lead to its effectiveness, and efficiency. Because of the increased confidence on the users on the system, it can be concluded that the online blood bank management system enhances blood transfusion safety because it provides better ways of handling the various processes in blood bank.

Chapter 8: Future Work

How can we benefit from this application in the future?

- The programmers may add some features that facilitate our application soon
- The donor can donate from his place instead of going to the hospital
- The user can buy blood bags instead of going to the hospital
- Some necessary tests before a donor can donate blood Such as blood sugar and pressure analysis
- All hospitals have reached the facility of blood donation abroad, if there is a shortage in some type: any country can help it
- In severe cases, the user can obtain blood bags for free
- Another future prospect is to develop artificial blood. Genetically modified blood cells are targeted for production in laboratory. Although it sounds a fantasy, progress can be expected within 10 or 15 years. Advantages in these types of blood components will be no risk of transfusion, neutral blood group and no fear of alloimmunization. However, the success and wide use of these components is questionable because of anticipated high price and risk of development in artificial culture medium.
- The application tells users when they can donate blood. It's important information because there must be an appropriate time interval between donations. Donation frequency depends on several factors, such as gender.
- App automated processes the client had to run, and as a result reduced the number of tasks and time consumed for arranging the users' donations.
- The client observed the increased blood donor's engagement in the idea of blood donation – people share their experiences and observations through social channels, donors claim they feel noticed and their involvement was appreciated.