

***Team:***

*Echipa Plasmă*

***Members:***

* *Lazăr Maria*
* *Simion Alexandra-Maria*
* *Todoran Ana-Corina*
* *Turcu Ema*
* *Ungur Maria*
* *Uscat Mihai Andrei*

***Team leader:***

*Simion Alexandra-Maria*

***Coordinator:***

*Borodi Flavia*

*Contents*

[*Presentation of the project requirements* 3](#_Toc516435761)

[*Used technologies* 4](#_Toc516435762)

[*Intallation manual* 5](#_Toc516435763)

[*Diagrams* 7](#_Toc516435764)

[*Use case diagrams* 8](#_Toc516435765)

[*Sequence diagrams* 11](#_Toc516435766)

[*Communication diagram* 13](#_Toc516435767)

[*Architecture diagrams* 14](#_Toc516435768)

[*Class diagrams* 15](#_Toc516435769)

[*Database diagrams* 17](#_Toc516435770)

[*Activity diagram* 19](#_Toc516435771)

[*Application Testing* 20](#_Toc516435772)

[*Help manual* 33](#_Toc516435773)

# *Presentation of the project requirements*

We have been tasked with the development of a blood donation application to enable its users to more easily manage the entire process of the donation. The application is to be designed to benefit each actor that may be involved in the donation process; therefore it is an application that can be used by donors and doctors alike.

The implementation of said application began by thoroughly investigating the given task in order to outline the actors and their roles in the flow of the app. There are three main actors which have to be accounted for: the donor, the doctor and the staff of the donation center, each with their own roles and restrictions. (e.g. a donor cannot donate if they do not meet certain health requirements, a doctor will manage blood requests and the donation center personnel will manage donors and their blood’s journey)

The development of the application was simultaneous, in the way that tasks were assigned in order for the project to progress on all fronts. Team members would take care of issues regarding the front end, the back end, the testing and the specification based on assigned tasks in order for the application to evolve evenly and in an efficient manner.

We are proud to have developed a user friendly, flexible application that molds based on the type of user that is logged in.

It enables anyone willing to donate blood to create an account, answer a number of general questions, fill in their credentials, and then have the rest of the issues taken care of. The application will inform its users of their ability to donate, the next recommended time for them to do so, and where to do it. If any questions regarding donation conditions were to arise, a variety of them are answered through a FAQ page, and if the questions remain unanswered, a contact page provides a direct way to reach an answer.

Furthermore, the application brings benefits to specialized personnel, as they can much more easily manage donations, as they can request blood based on their needs, and also see its availability. Doctors can easily communicate with donation personnel in order to have access to blood requests and their status. The donation center staff is able to see the closest compatible donor and attribute donations based on the information.

The application is the ideal solution for any user, regardless of their computer skills or expertise, as it is intuitive, friendly and fairly simple to use. We have solved the tasks as a team, and we are proud to have done it BloodyFast.

# *Used technologies*

1. **Database server**: *MySQL*
2. **Programming language**:
3. **Front-end**: *JavaScript with React, HTML, CSS*
4. **Back-end**: *Java, Spring*
5. **ORM**: *Hibernate*
6. **Tools used for diagram creation**: *draw.io and StarUML*
7. **Version control**: *GitHub*
8. **Task management**: *Trello*
9. **GUI creation**: *JavaScript, React, HTML, CSS*
10. **Client-server communication**: *Axios*
11. **Testing - manual or automated**:
12. **Manual testing**: *bug reports in Trello, Test Cases and Final Test Report docs*
13. **Automation testing**: *C# with Selenium Webdriver*

# *Intallation manual*

create new folder for project --> open GitBash there:

1. git clone https://github.com/HaremUscat/blood-donation.git

va va cere username si parola de la contul personal de github

2. git init

3.git remote add origin https://HaremUscat:haremuscat10@github.com/HaremUscat/blood-donation

4.git pull

5.git checkout develop

***First Contact With Node.js, React, Bootstrap***

Pentru a rula proiectul “A Basic Submit Project” (asta e doar parte de front-end, nu e conectata la niciun server si nu e functionala, that you gotta figure out how to do on your own in functie de tehnologie):

- instalati Node.js ( <https://nodejs.org/en/> )

- va recomand din SUFLET WebStorm ca IDE. E incredibil, completeaza foarte multe chestii automat, gaseste multe erori in JavaScript, HTML si CSS. E ca un extra developer care te ajuta la nevoie. E gratuit cu contul de student.

- descarcati arhiva cu proiectul, intrati in folder unde vedeti cele 3 subfoldere: node\_modules, public, src si niste fisiere .json

- deschideti aici un command line

- rulati comanda “npm start”

- in browser, veti vedea componenta Home si componenta SubmitToDatabase. Explorati codul, am pus comentarii, ca sa va familiarizati cu React. Am folosit si putin Bootstrap pentru asezarea in pagina si pentru partea de “mini-form”, cu textbox si buton.- ca sa explorati efectiv componentele, dati click dreapta si “Inspect”. Puteti explora html-ul si css-ul foarte usor asa. In partea de “Console” veti vedea mesaje cand modificati textul din textbox si cand dati click pe buton. Mai multa functionalitate nu am adaugat la proiectul basic, doar la cel cu tot cu server voi adauga, pentru a ma lega de server

- pentru a opri rularea, in command line dati “Ctrl+C” si apoi “exit”.

***Cum functioneaza?***

- browserul va intra mai intai in fisierul “index.js” specificat. Index.js va incerca sa randeze componenta App, pe care o gasiti in app.js, in containerul marcat “root”, pe care il gasiti in index.html. In index.html nu vom face multe modificari, majoritatea html-ului va fi inclus in componentele React.

- ce vedeti in folderul “modules” sunt componente React. App are niste rute care sunt, de fapt, mapari de la linkul afisat sus in browser la componente React scrise de noi (“pagini”, sa zicem).

- orice componenta React are o metoda “render”, in care punem html-ul si la unele componente vom specifica functii care sa se apeleze in urma unor evenimente (de exemplu, la buton avem “onClick”). Functiile de comportament le putem defini langa functia “render”, tot in componenta noastra. Daca ne uitam la componenta “SubmitToDatabase”, vedem ca are un constructor cu o “stare”. Acolo, in state, vom memora diverse lucruri (alea sunt ca niste fielduri private cu informatii).

- Sub state veti vedea niste binduri. Fara acele binduri, functiile noastre nu vor sti ce inseamna cuvantul “this”, adica in ce context sa ruleze. Practic, cand componenta noastra se randeaza si apare pe ecran, functiile acelea vor fi bind-uite, “legate”, de instanta tocmai creata a componentei, si vor sti ce inseamna “this”. Fara acel bind, o sa avem erori :D

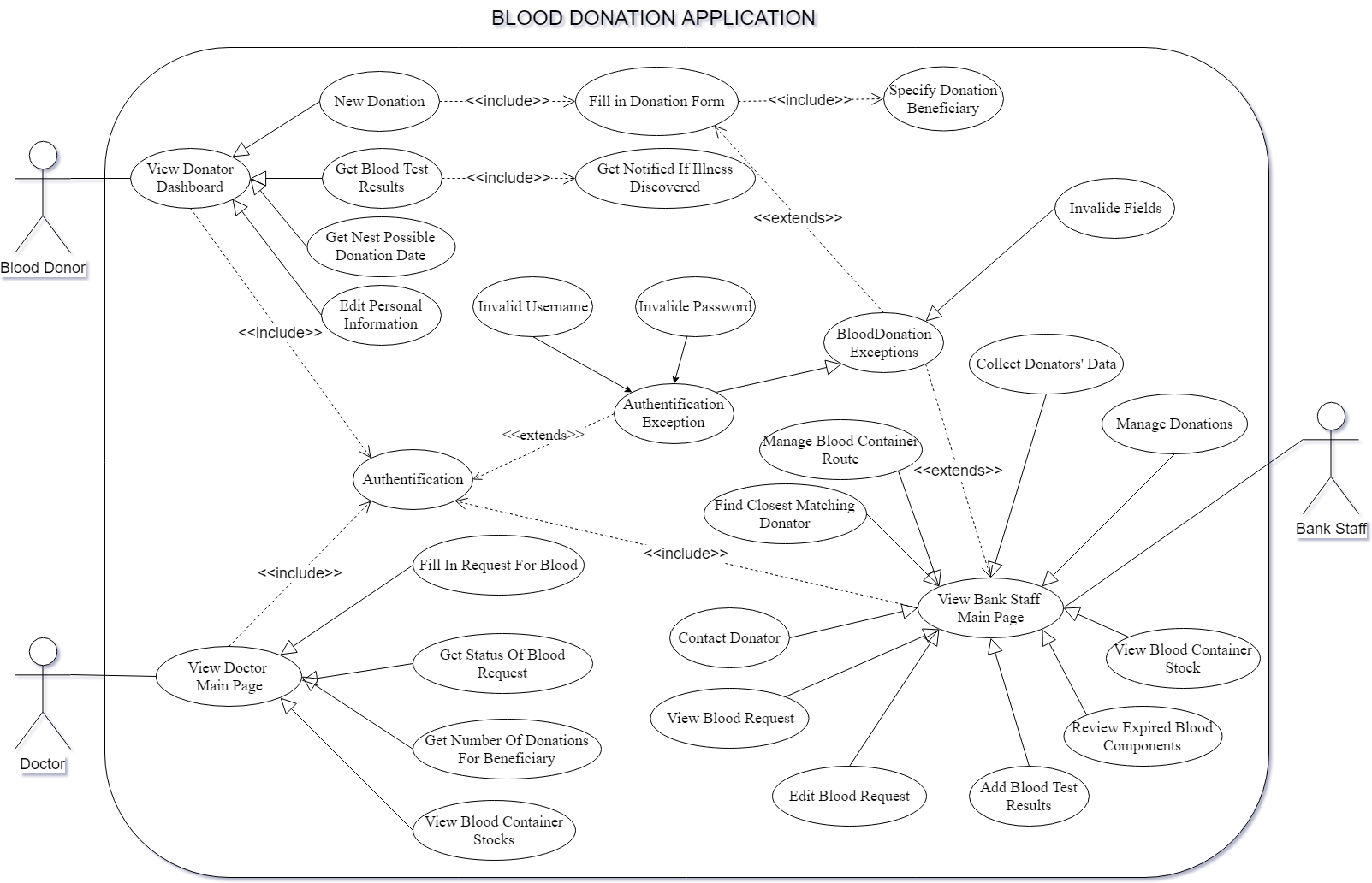
- in node\_modules sunt librarii/pachete de Node.js. Basically, cod scris de altii, cum e si React, de altfel. Le-am instalat eu, dar va trebui sa va arat/scriu un mini-tutorial ca sa stiti sa instalati si sa folositi pachete Node. Sunt EXTREM de variate, de la calendare, la loadere animate, la pachete care fac tot felul de minuni. Trebuie doar instalate in proiect si importate in fisierele javascript in care vreti sa le folositi.

- explorati codul si cititi comentariile. For any questions, feel free to ask. In curand va trimit si varianta conectata la server si la baza de date.

# *Diagrams*

* Use case diagram(s)
* Sequence diagram(s)
* Collaboration diagram(s)
* Architecture diagram(s)
* Class diagram(s)
* Database diagram
* Activity diagram

# *Use case diagrams*



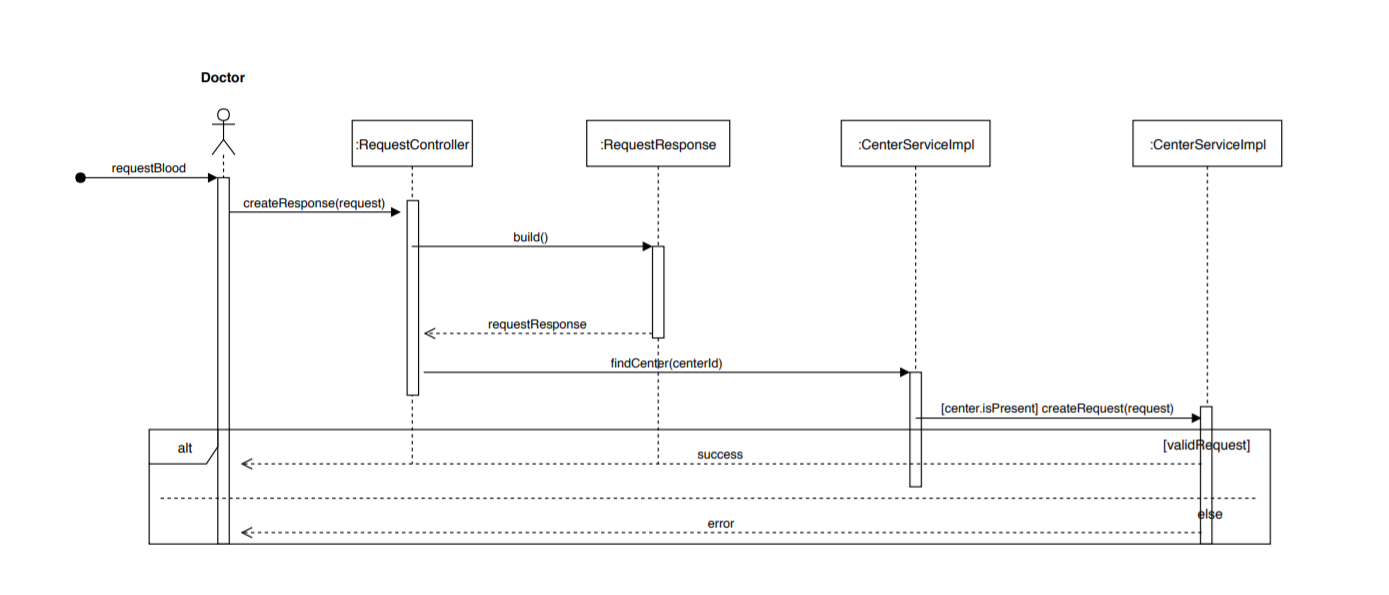
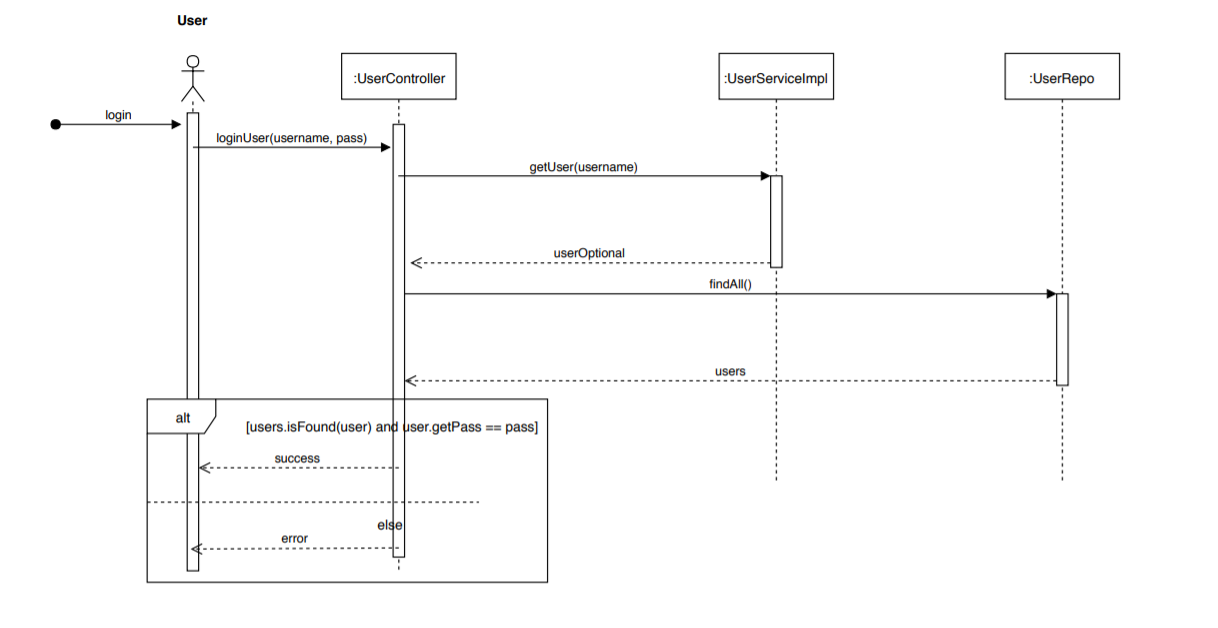
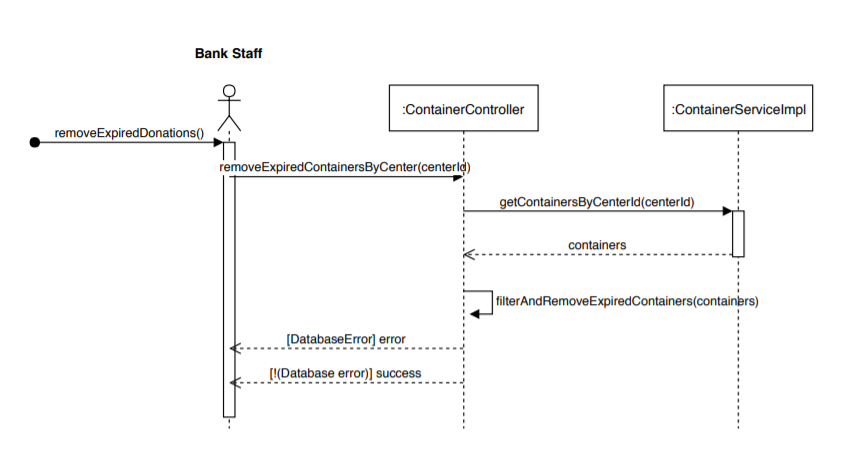
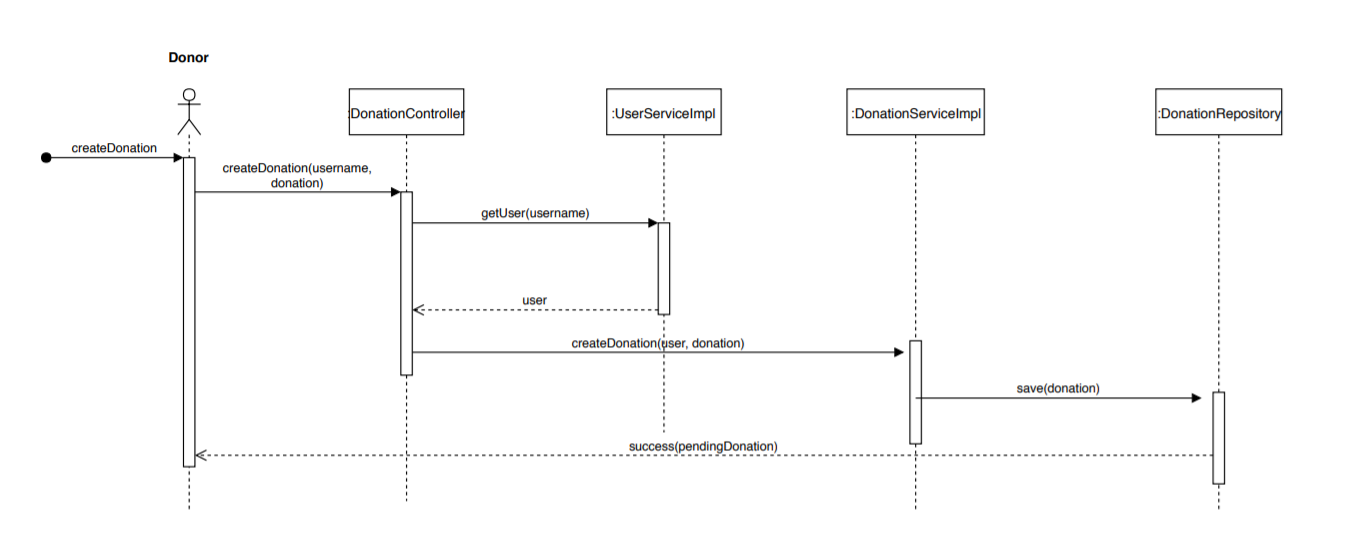
|  |  |
| --- | --- |
| ***Use case name*** | Authentification |
| ***Participating actors*** | Donator / Doctor / Bank Staff |
| ***Flow of events*** | 1. Enter username 2. Enter password 3. Click login button |
| ***Entry condition*** | None |
| ***Exit condition*** | * The user is logged into its corresponding dashboard, OR * A message of error is displayed |
| ***Quality requirements*** | * Fields for username and password must be filled in |

|  |  |
| --- | --- |
| ***Use case name*** | Get Blood Test Results |
| ***Participating actors*** | Donor |
| ***Flow of events*** | 1. Select test results link from navbar 2. All tests will be displayed 3. Download pdf to show a certain result details |
| ***Entry condition*** | Donor must be logged in |
| ***Exit condition*** | None |
| ***Quality requirements*** | Donor must have previous donations to be displayed otherwise this page will contain 0 results |

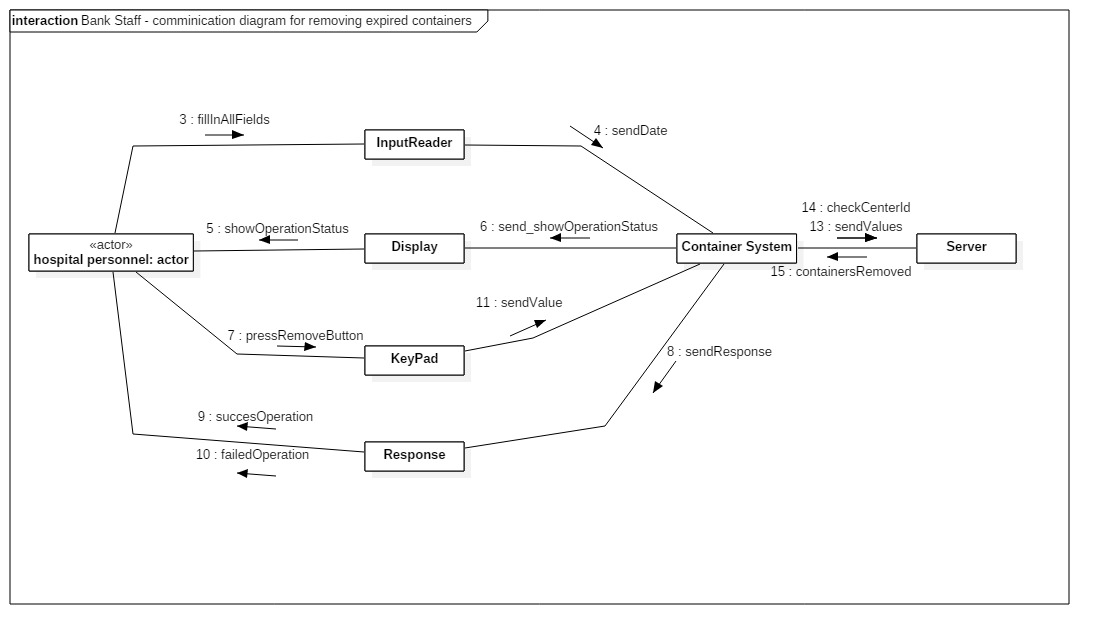
|  |  |
| --- | --- |
| ***Use case name*** | Fill in Request for Blood |
| ***Participating actors*** | Doctor |
| ***Flow of events*** | 1. Select blood group 2. Select Rh 3. Fill no. of thrombocyte units needed 4. Fill no. of red cell units needed 5. Fill no. of plasma units needed 6. Fill id of the donation center 7. Fill address of requesting hospital 8. Fill beneficiary full name 9. Select urgency level 10. Select agreement 11. Press send request button |
| ***Entry condition*** | Doctor must be logged in |
| ***Exit condition*** | None |
| ***Quality requirements*** | New request created if all fields are filled in |

|  |  |
| --- | --- |
| ***Use case name*** | Remove Expired Blood Components |
| ***Participating actors*** | Bank Staff |
| ***Flow of events*** | 1. Select stock link from navbar 2. Press “REMOVE ALL EXPIRED” button |
| ***Entry condition*** | Staff must be logged in |
| ***Exit condition*** | None |
| ***Quality requirements*** | All expired containers will be removed |

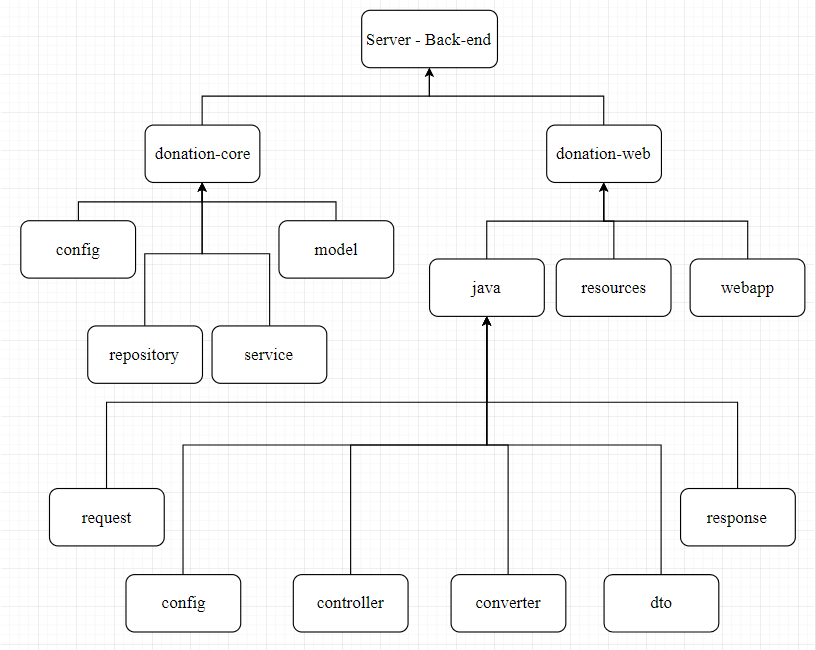
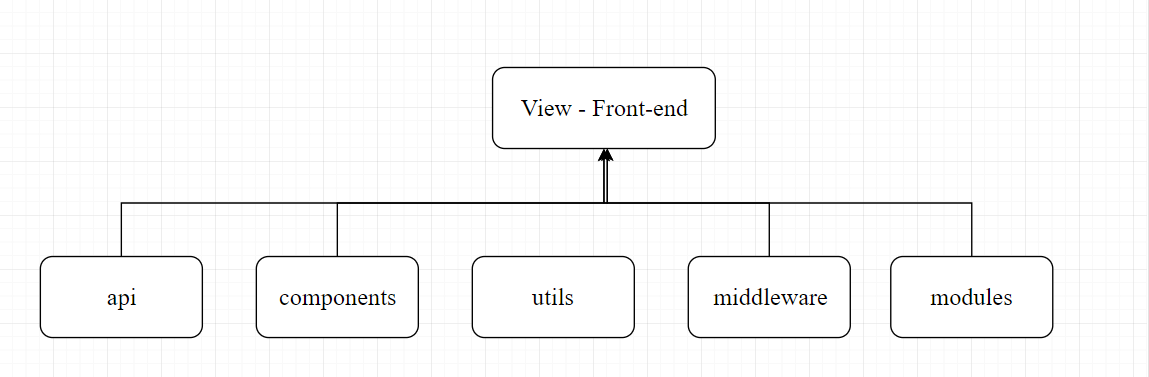
# *Sequence diagrams*



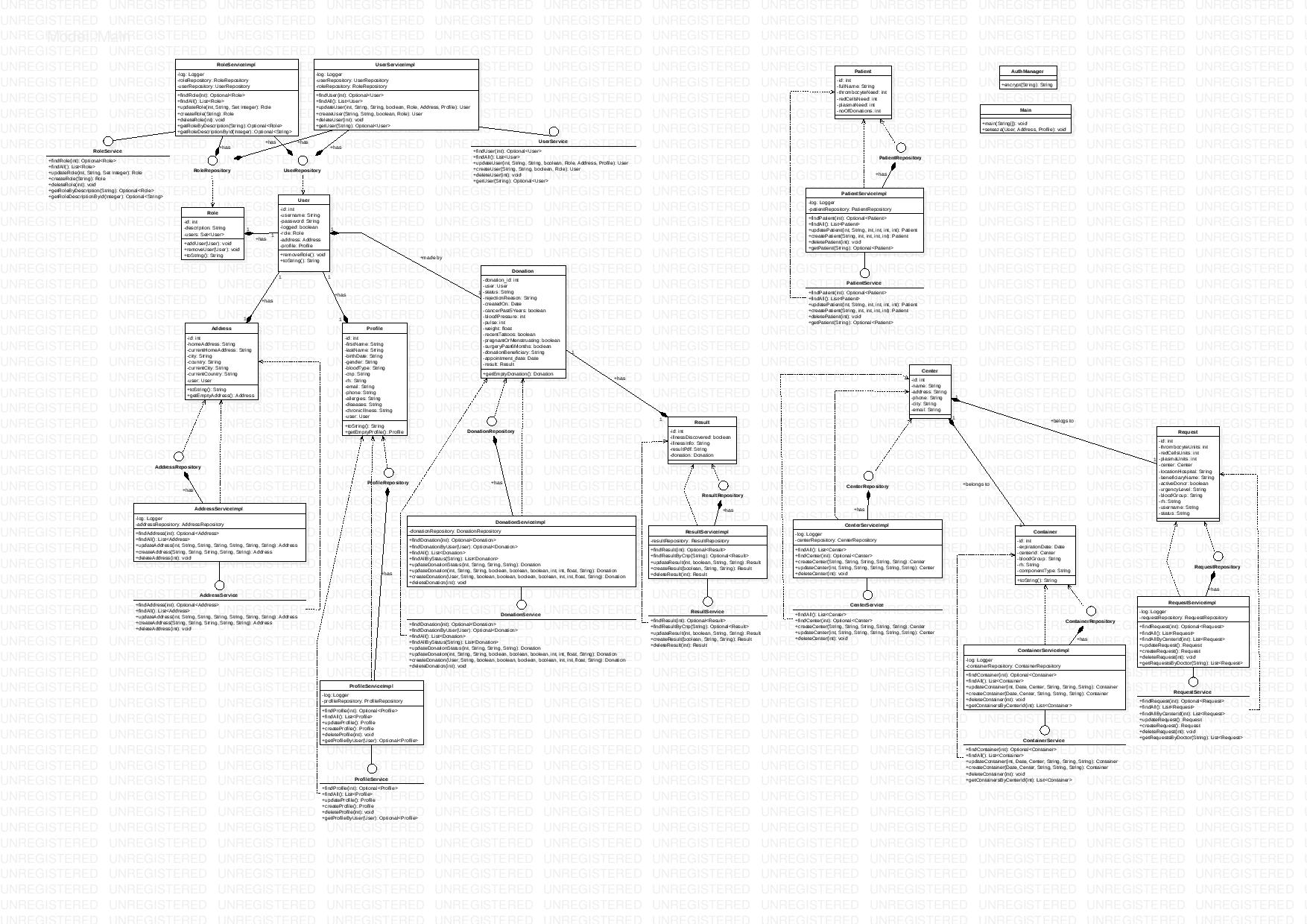
# *Communication diagram*



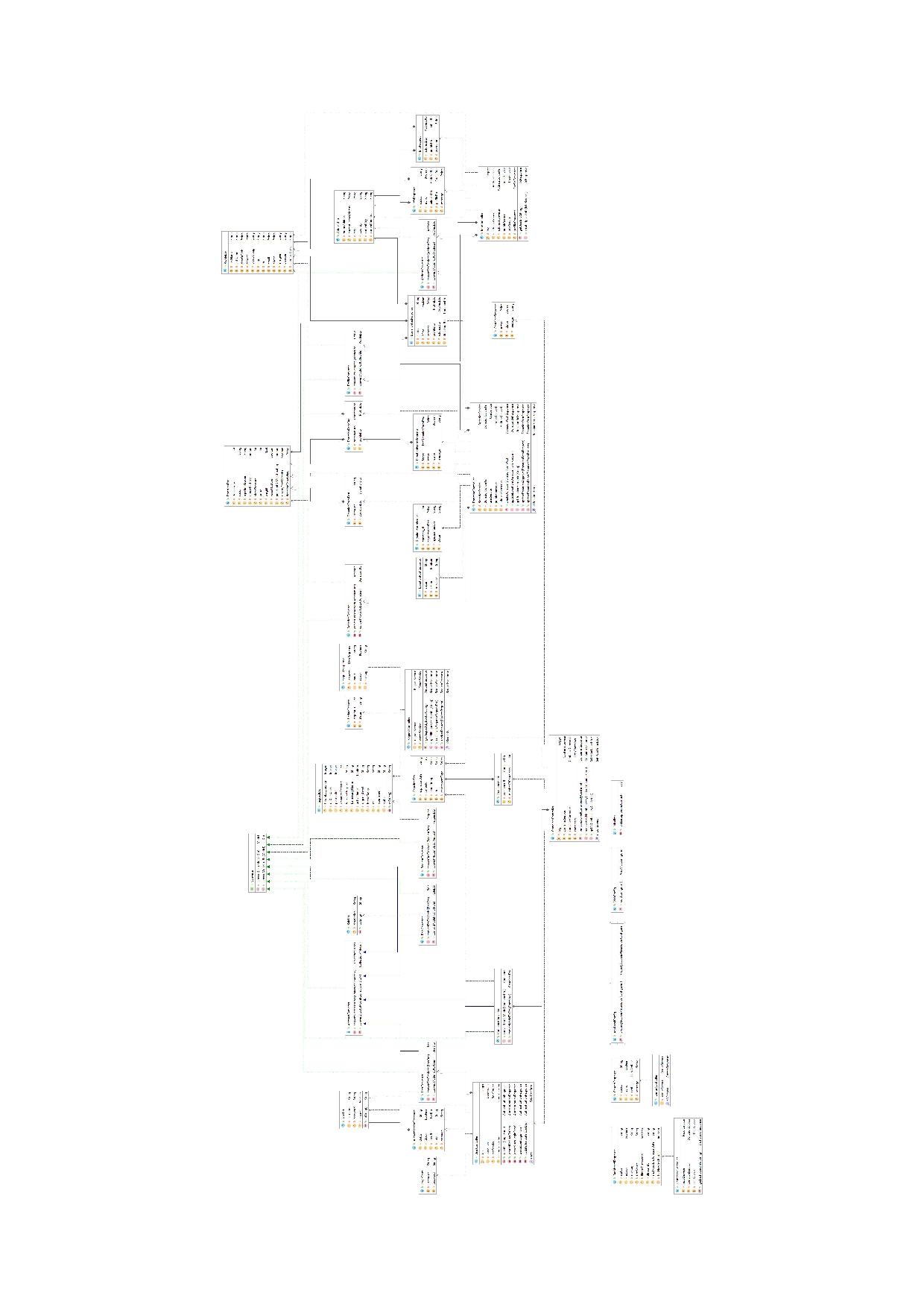
# *Architecture diagrams*



# *Class diagrams*

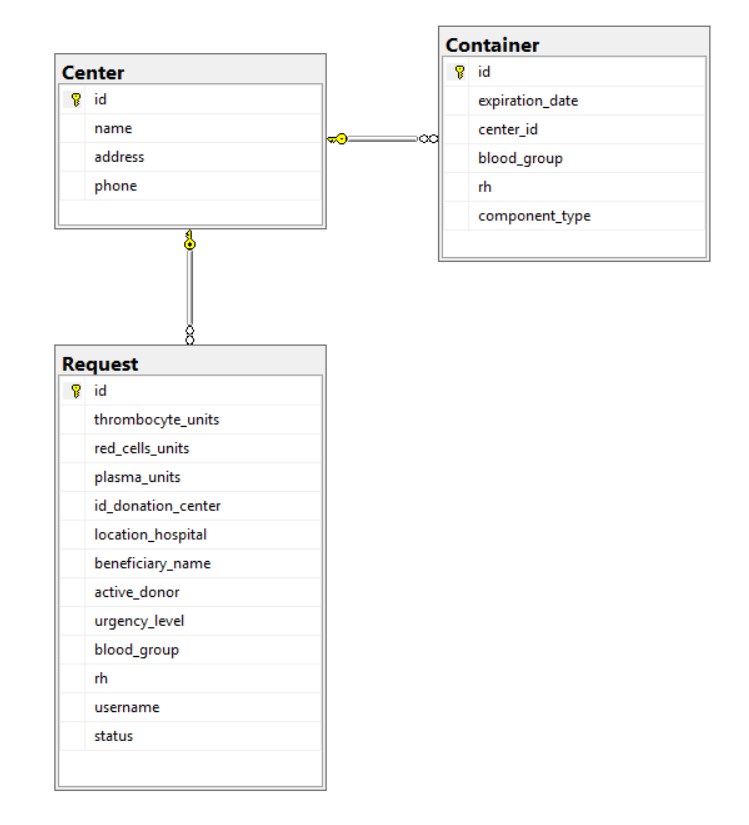


Go to Diagrams/Class diagram/CoreClassDiagramA4.pdf from DocumentationISS folder for zooming in.



Go to Diagrams/Class diagram/WebClassDiagramA4.pdf from DocumentationISS folder for zooming in.

# *Database diagrams*diagram1.pngdiagram2.png



# *Activity diagram*

# *ActivityDiagram.png*

# *Application Testing*

**BloodyFast**

**Test Report**

**Functionality**

**Test for – all the links in web pages, database connection, forms used for submitting or getting information from the user in the web pages, Cookie testing etc.**

**Check all the links:**

* Test the outgoing links from all the pages to specific domain under test.
* Test all internal links.
* Test links jumping on the same pages.
* Test links used to send email to admin or other users from web pages.
* Test to check if there are any orphan pages.
* Finally, link checking includes, check for broken links in all above-mentioned links.

**Test forms on all pages:**

**Forms are an integral part of any website. Forms are used for receiving information from users and to interact with them. So what should be checked in these forms?**

* First, check all the validations on each field.
* Check for default values of the fields.
* Wrong inputs in the forms to the fields in the forms.
* Options to create forms if any, form delete, view or modify the forms.

So sign up flow should get executed correctly. There are different field validations like email Ids, User financial info validations etc. All these validations should get checked in manual or automated web testing.

**Cookies Testing:**

Cookies are small files stored on the user machine. These are basically used to maintain the session- mainly the login sessions. Test the application by enabling or disabling the cookies in your browser options.

Test if the cookies are encrypted before writing to the user machine. If you are testing the session cookies (i.e. cookies that expire after the session ends) check for login sessions and user stats after the session ends. Check effect on application security by deleting the cookies.

**Validate your HTML/CSS:**

If you are optimizing your site for Search engines then HTML/CSS validation is the most important one. Mainly validate the site for HTML syntax errors. Check if the site is crawlable to different search engines.

**Database testing:**

Data consistency is also very important in a web application. Check for data integrity and errors while you edit, delete, modify the forms or do any DB related functionality.

Check if all the database queries are executing correctly, data is retrieved and also updated correctly.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Donor** | **Doctor** | **Staff** |
| **Link Check** | **Failed** | **Failed** | **Failed** |
| **Forms on all pages** | **Failed** | **Failed** | **Failed** |
| **Cookies** | **Failed** | **Failed** | **Failed** |
| **HTML/CSS** | **Passed** | **Passed** | **Passed** |
| **Database** | **Passed** | **Passed** | **Passed** |

**Important issues found:**

* **Donor**

1. Upon registering, the email field is not mandatory and can be left empty
2. Clicking on the "Test Results" will redirect the user to the test results page of the Staff members. He will have the Staff navigation bar, but he cannot access any of

the pages – Fixed

1. Fields on the "My Information" page are not validated
   1. The donor can submit a donation form without answering the mandatory Required Information (the first Yes/No questions)

* **Doctor**
  1. Pages not implemented: Pacients in Need, City Blood Stocks
  2. When the doctor fills in a request with information that is wrong, gets the "Request sent" message, but the browser returns an error message and the request is not saved
* **Staff**
  1. Pages not implemented: Stocks - Fixed, City Stocks, Find Donor, Tests

1. No validation on the appointment date(can be set in the past, not respecting the format etc.)
   1. When clicking on "Add/Remove from Stock" button without information filled in, "Error 404" error message is displayed

* **General**
  1. The application crashes if the user disables cookies from the web browser

**Compatibility**

**Browser compatibility:**

Some applications are very dependent on browsers. Different browsers have different configurations and settings that your web page should be compatible with.

Test the web application on different browsers like Internet Explorer, Firefox, Netscape Navigator, AOL, Safari, Opera browsers with different versions.

**OS compatibility:**

Some functionality in your web application is that it may not be compatible with all operating systems. All new technologies used in web development like graphic designs, interface calls like different API’s may not be available in all Operating Systems.

Hence test your web application on different operating systems like Windows, Unix, MAC, Linux, Solaris with different OS flavors.

**Mobile browsing:**

We are in the new technology era. So in future Mobile browsing will rock. Test your web pages on mobile browsers.

Compatibility issues may be there on mobile devices as well.

**Printing options:**

If you are giving page-printing options then make sure fonts, page alignment, page graphics etc., are getting printed properly. Pages should fit to the paper size or as per the size mentioned in the printing option.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Donor** | **Doctor** | **Staff** |
| **Browser** | **Failed** | **Failed** | **Failed** |
| **OS** | **Could not test** | **Could not test** | **Could not test** |
| **Mobile** | **Passed** | **Passed** | **Passed** |
| **Printing options** | **Passed** | **Passed** | **Passed** |

**Important issues found:**

* **General**

1. The application is not compatible with Safari, hence it is not compatible with Apple users
2. Linux and macOS X compatibility could not be tested as we did not have the resources necessary for in-depth tests

**Usability**

**Test for navigation:**

Navigation means how a user surfs the web pages, different controls like buttons, boxes or how the user uses the links on the pages to surf different pages.

**Usability testing includes the following:**

* The website should be easy to use.
* Instructions provided should be very clear.
* Check if the instructions provided are perfect to satisfy its purpose.
* The main menu should be provided on each page.
* It should be consistent enough.

**Content checking:**

Content should be logical and easy to understand. Check for spelling errors. Usage of dark colors annoys the users and should not be used in the site theme.

You can follow some standard colors that are used for web page and content building. These are the commonly accepted standards like what I mentioned above about annoying colors, fonts, frames etc.

Content should be meaningful. All the anchor text links should be working properly. Images should be placed properly with proper sizes.

These are some of the basic important standards that should be followed in web development. Your task is to validate all for UI testing.

**Other user information for user help:**

Like search option, sitemap also helps files etc. The sitemap should be present with all the links in websites with a proper tree view of navigation. Check for all links on the sitemap.

“Search on the site” option will help users to find content pages that they are looking for easily and quickly. These are all optional items and if present they should be validated.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Donor** | **Doctor** | **Staff** |
|  |  |  |  |
| **Test for navigation** | **Passed** | **Passed** | **Passed** |
| **Content checking** | **Failed** | **Passed** | **Passed** |
| **User information** | **Failed** | **Failed** | **Failed** |

**Important issues found:**

* **General**

1. The application does not have a "Search on the site" option
2. A small spelling error can be noticed after the user has the "approved for

donation" status: "**In** the morning before **the/your** blood donation, you may drink[…]"

1. In the "Welcome,[username]" screen, a space before the name should be placed o According to the documentation, the application is available for Romanian

users, but the site is only available in English at the moment

5

**Interface**

This is done by verifying that communication is done properly. Compatibility of the server with software, hardware, network, and the database should be tested.

**The main interfaces are:**

* Web server and application server interface
* Application server and Database server interface

Check if all the interactions between these servers are executed and errors are handled properly. If database or web server returns an error message for any query by application server then application server should catch and display these error messages appropriately to the users.

Check what happens if the user interrupts any transaction in-between? Check what happens if the connection to the web server is reset in between?

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Donor** | **Doctor** | **Staff** |
| **Web server andapplication serverinterface** | **Passed** | **Passed** | **Passed** |
| **Application server andDatabase serverinterface** | **Passed** | **Passed** | **Failed** |

**Important issues found:**

* **Staff**

1. When clicking on "Add/Remove from Stock" button without information filled in, "Error 404" error message is displayed

**Security**

Following are some of the test cases for web security testing:

* Test by pasting internal URL directly into the browser address bar without login. Internal pages should not open.
* If you are logged in using username and password and browsing internal pages then try changing URL options directly. I.e. If you are checking some publisher site statistics with publisher site ID= 123. Try directly changing the URL site ID parameter to different site ID which is not related to the logged in user. Access should be denied for this user to view others stats.
* Try some invalid inputs in input fields like login username, password, input text boxes etc. Check the system’s reaction to all invalid inputs.
* Web directories or files should not be accessible directly unless they are given download option.
* Test the CAPTCHA for automating script logins.
* All transactions, error messages, security breach attempts should get logged in log files somewhere on the web server.

The primary reason for testing the security of a web is to identify potential vulnerabilities and subsequently repair them.

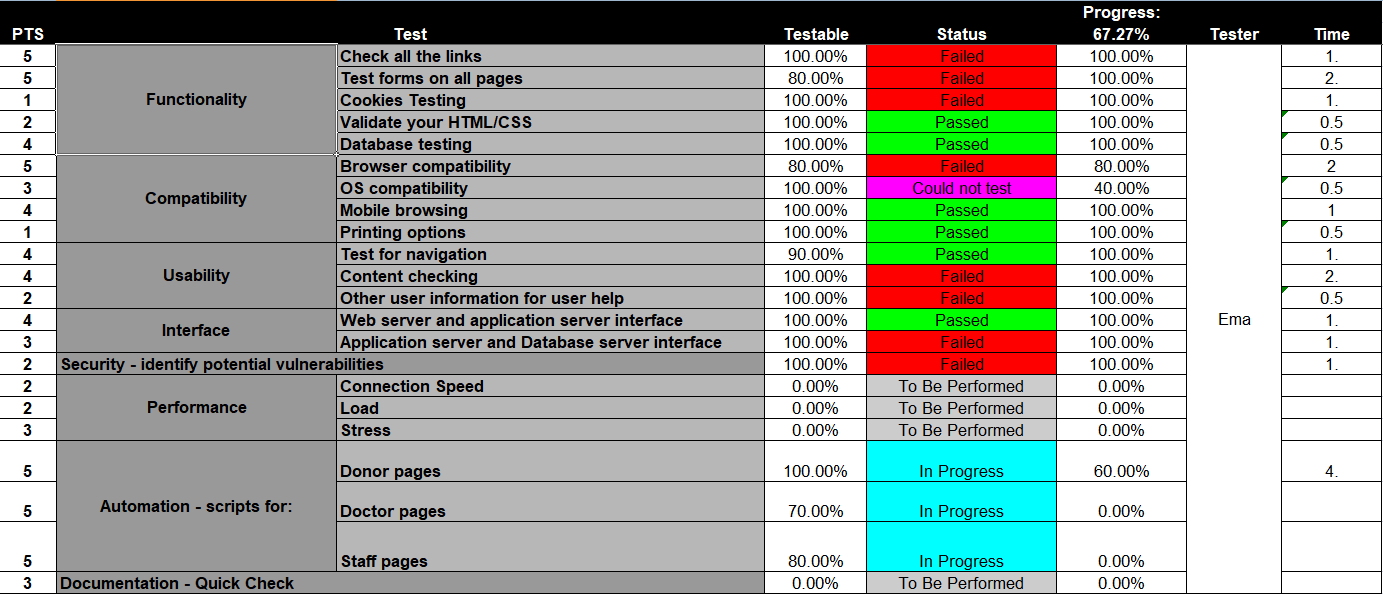
* + Network Scanning
  + Vulnerability Scanning
  + Password Cracking
  + Log Review
  + Integrity Checkers
  + Virus Detection

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Donor** | **Doctor** | **Staff** |
| **Security – identify vulnerability issues** | **Failed** | **Passed** | **Passed** |

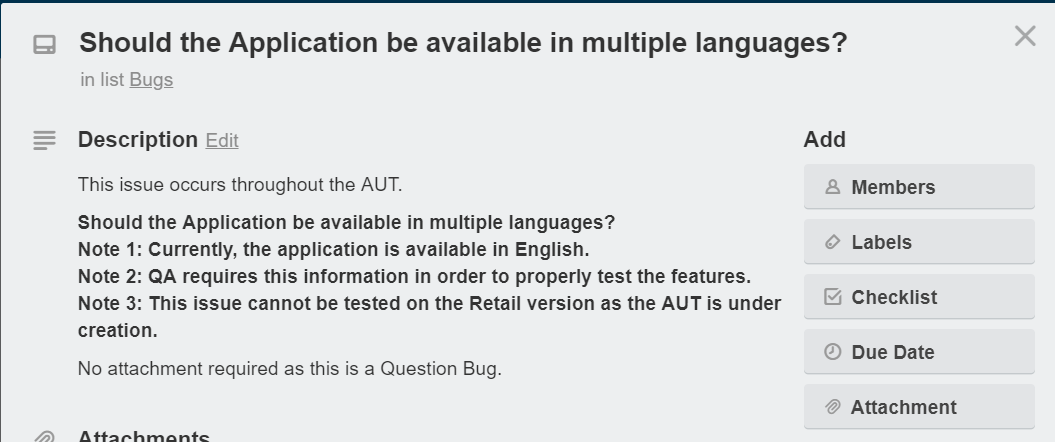
**Important issues found:**

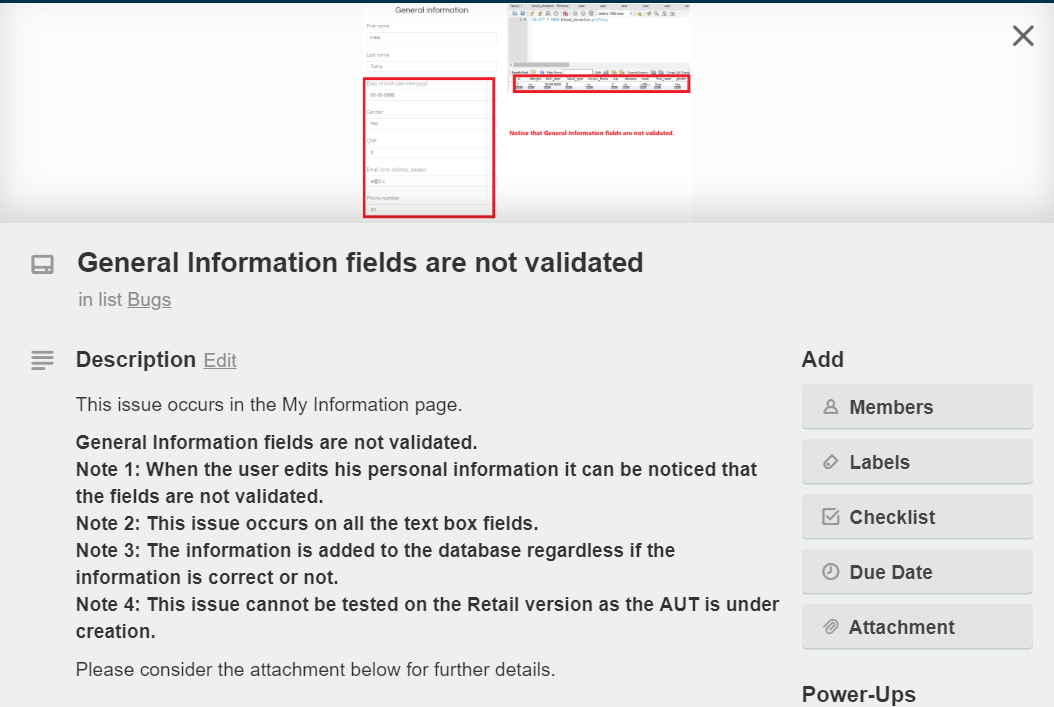
* **Donor**

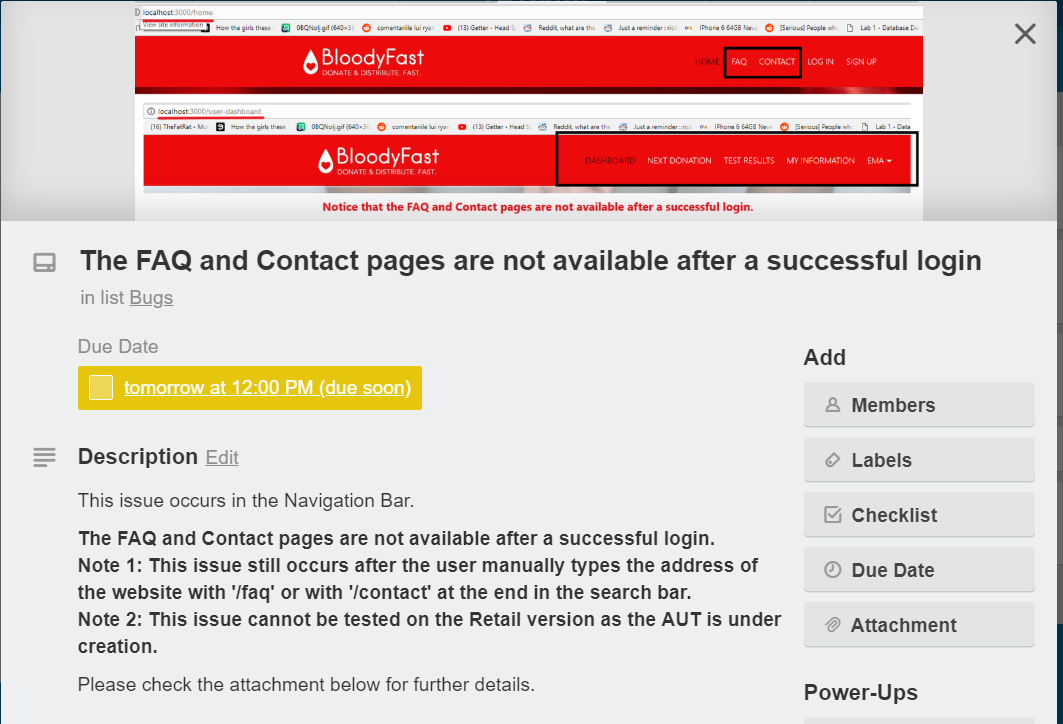
1. No email verification when signing up, there is a probability of spamming the server and bot users
2. Captcha verification would be a plus

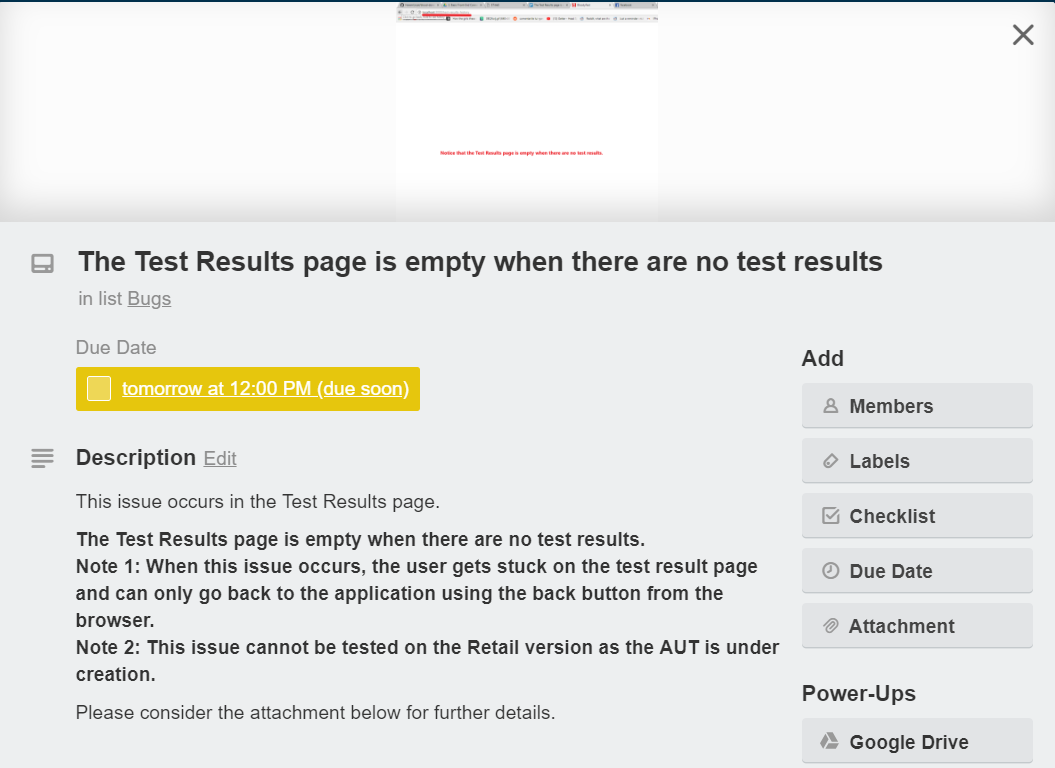


Some manual testing examples from Trello







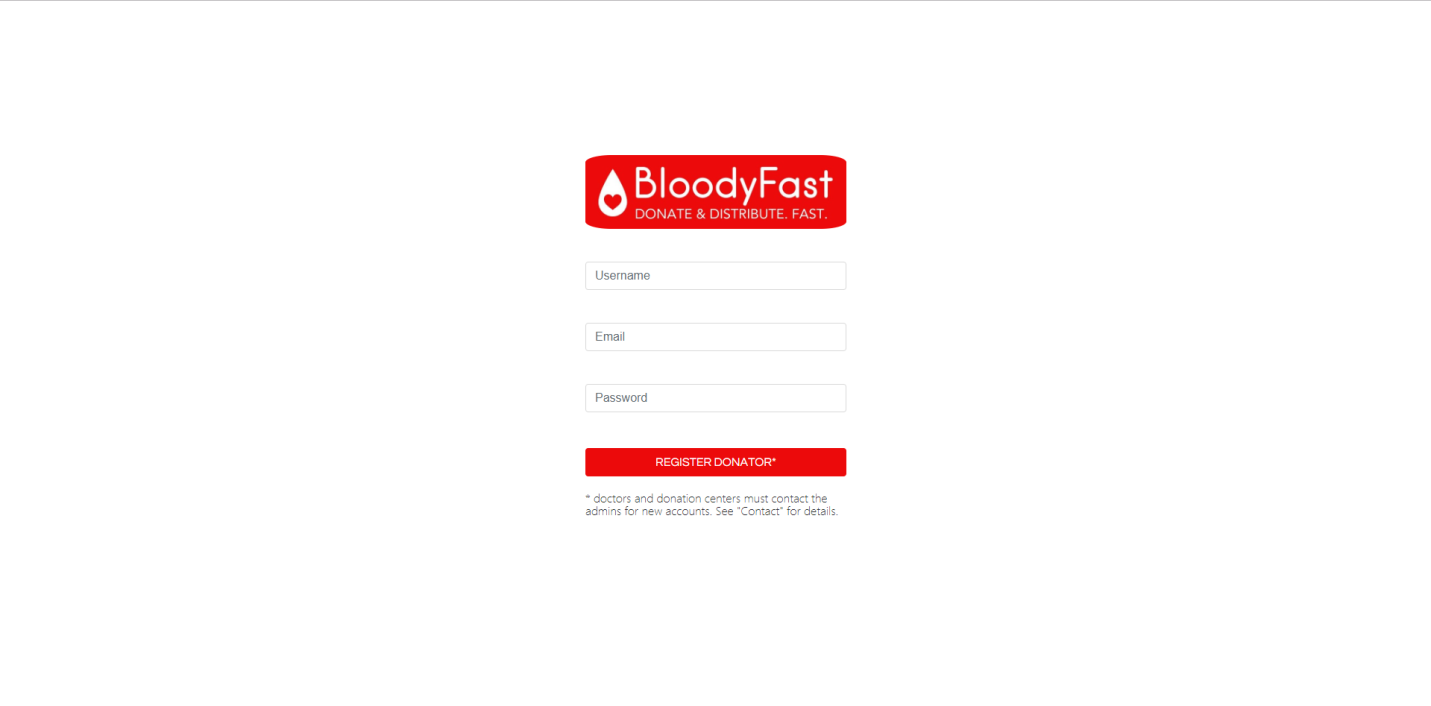


# *Help manual*

Bloody fast is an application meant to ease the entire process of blood donation, for all its partakers. Here’s what you need to know to get you started:

**Donor**

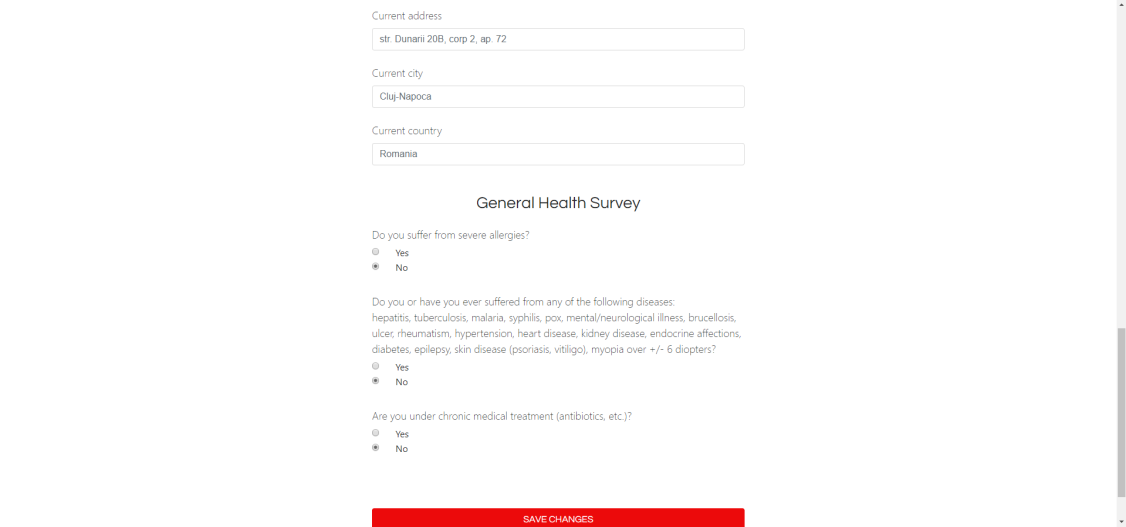
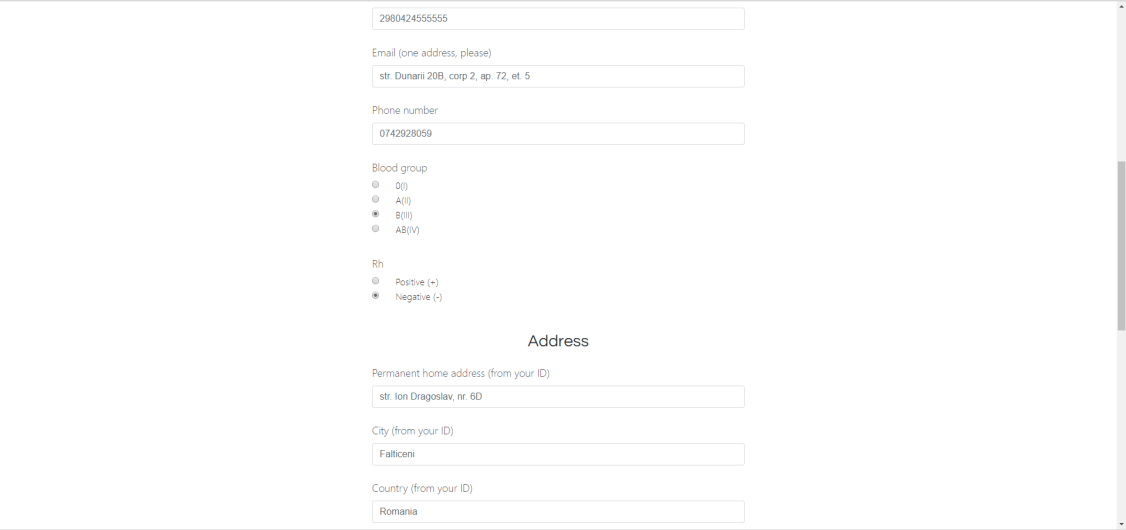
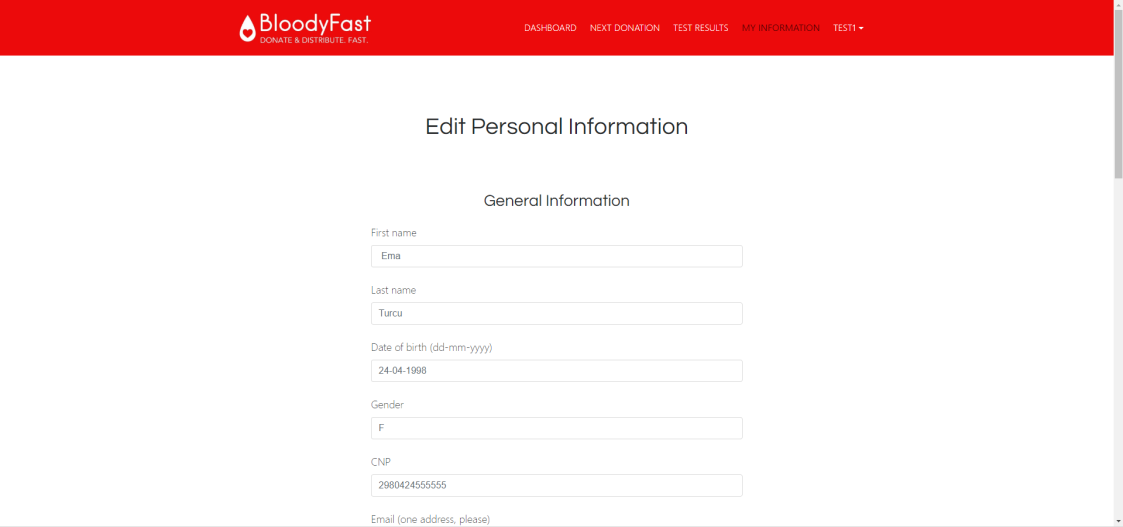
If you are eager to donate blood, look no further. Our application walks you through all of the steps, you simply need to create an account through our registration page by entering the required information.



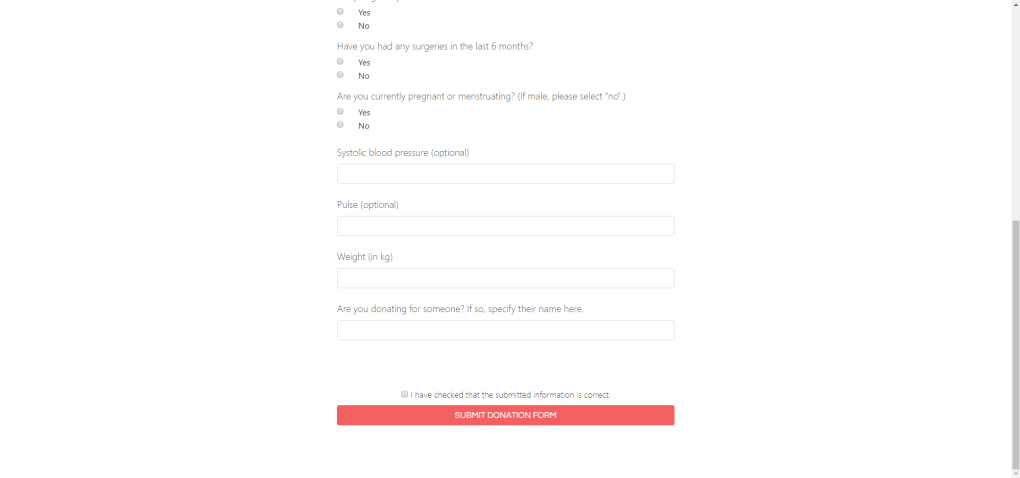
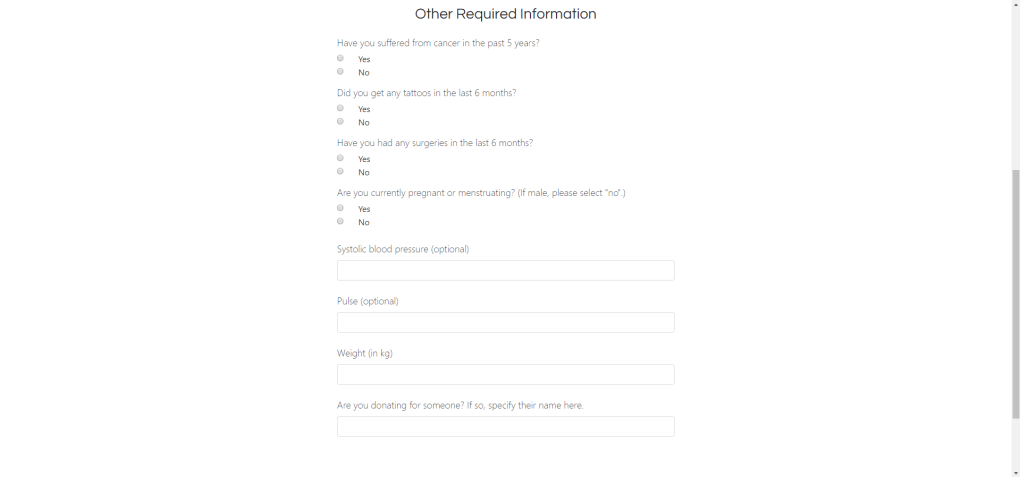
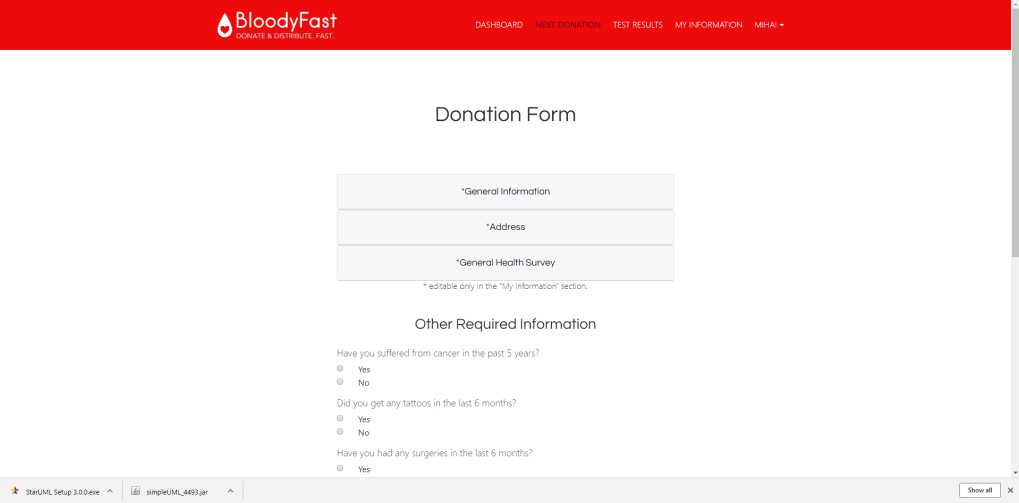
Already have an account? Enter your credentials and log in to find your very own user dashboard where you can see your current test and donation situation (test results and upcoming possible donation date).



If you are new, however, we kindly ask you to fill in some personal information in order for us to improve the efficiency of your future donations. Do this by clicking on the My Information section in the navigation bar and inserting the required information.



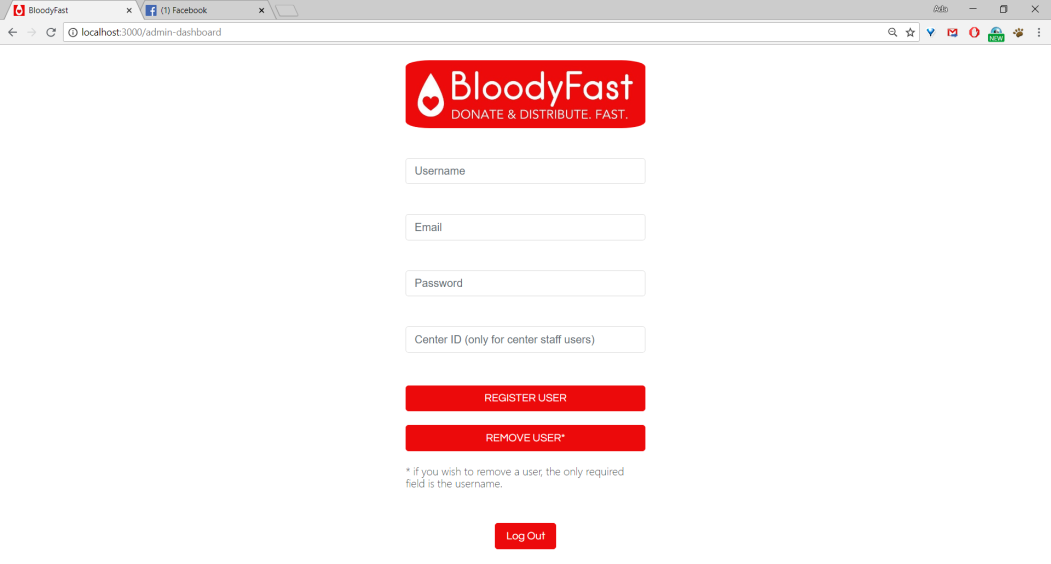
After this step you are free to fill in a donation form, if you are willing to donate blood. Do this by clicking the Next Donation tab, that will redirect you to the donation form page where you can fill some critical fields that will determine whether you are suitable to donate blood or not.



Once ready, the results can be accessed in your Test Results tab.

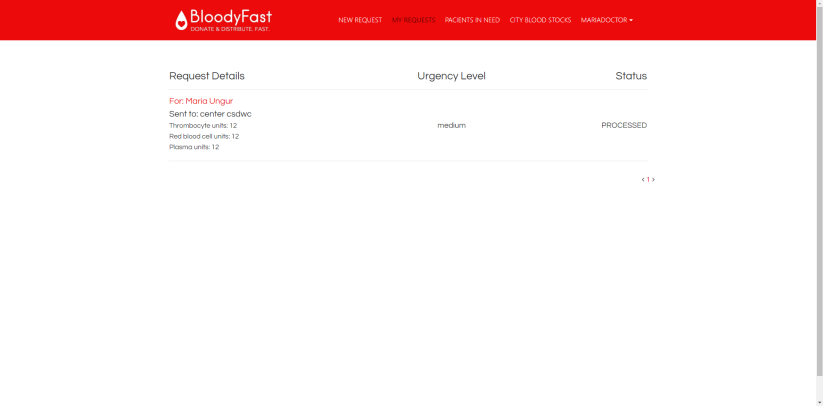
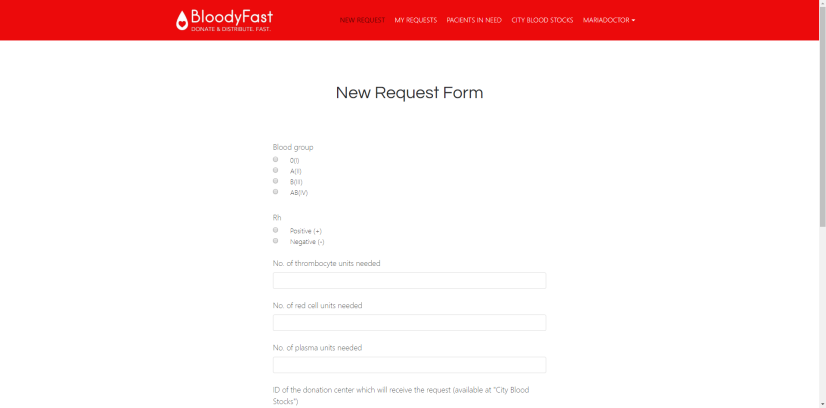
**Doctor & Bank Staff**

Official personnel involved in the process of donation do not have to create their own accounts as they are created for them by an admin.

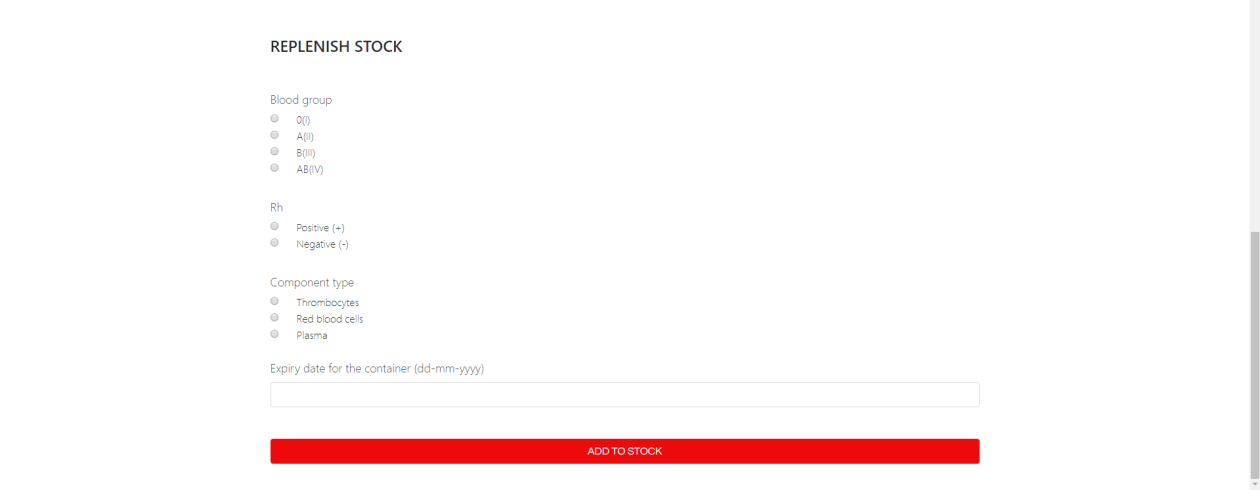
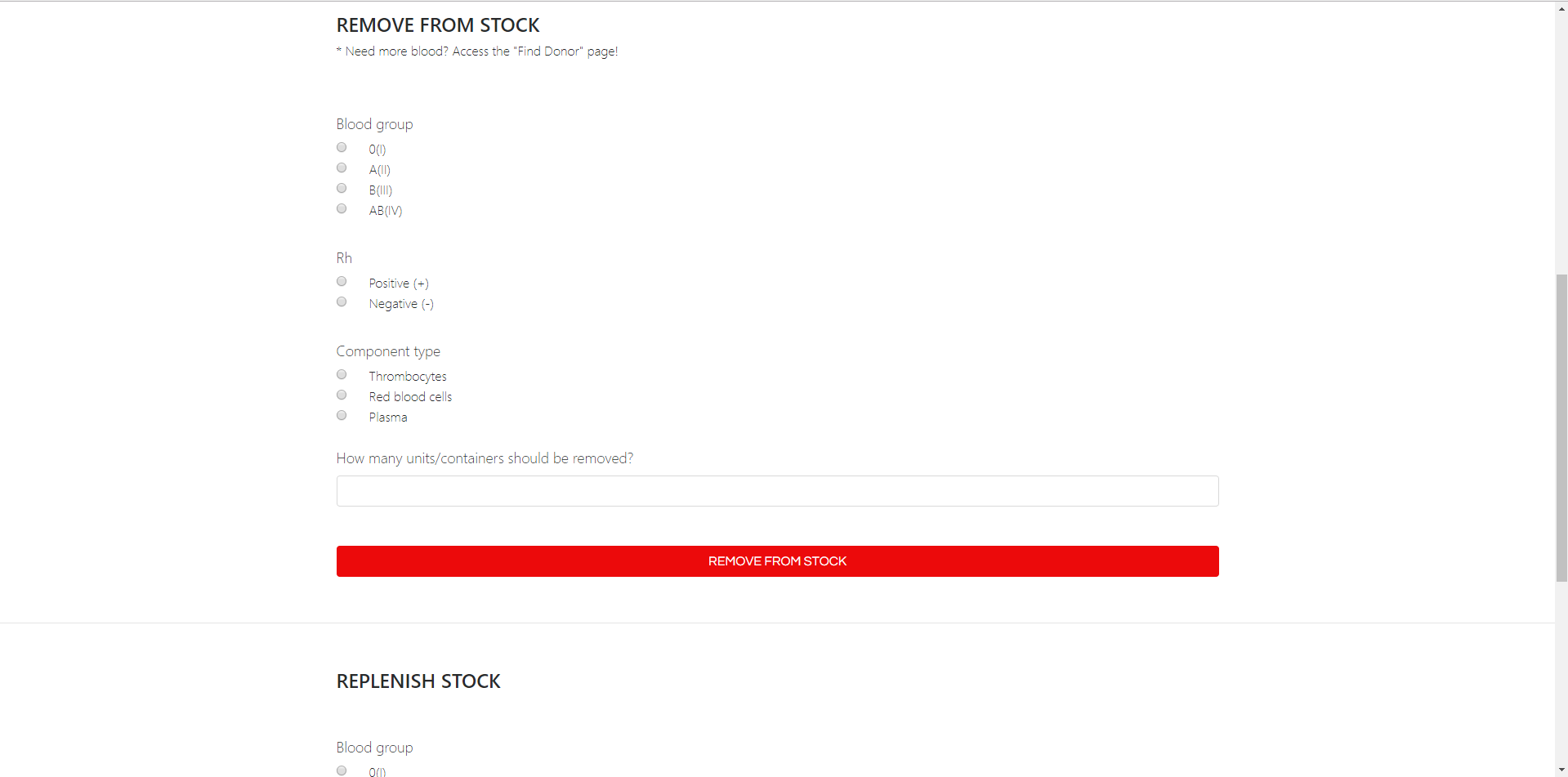
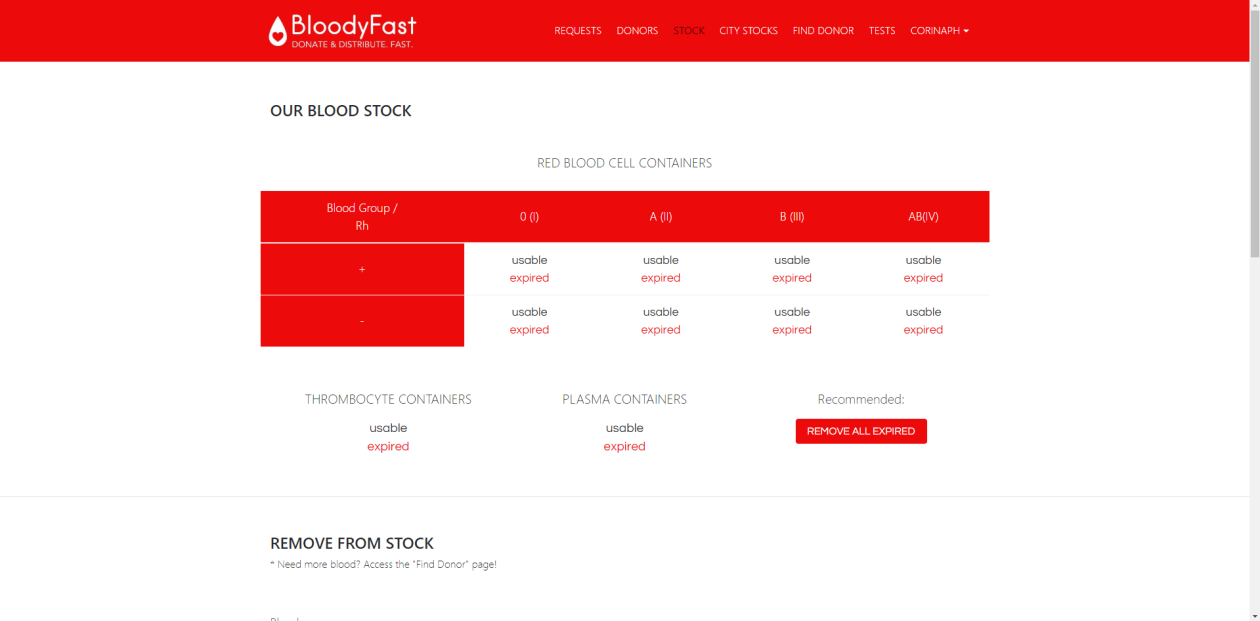


On the other hand, you are able to interact with one another in order to reach the best possible outcomes when it comes to blood donation.

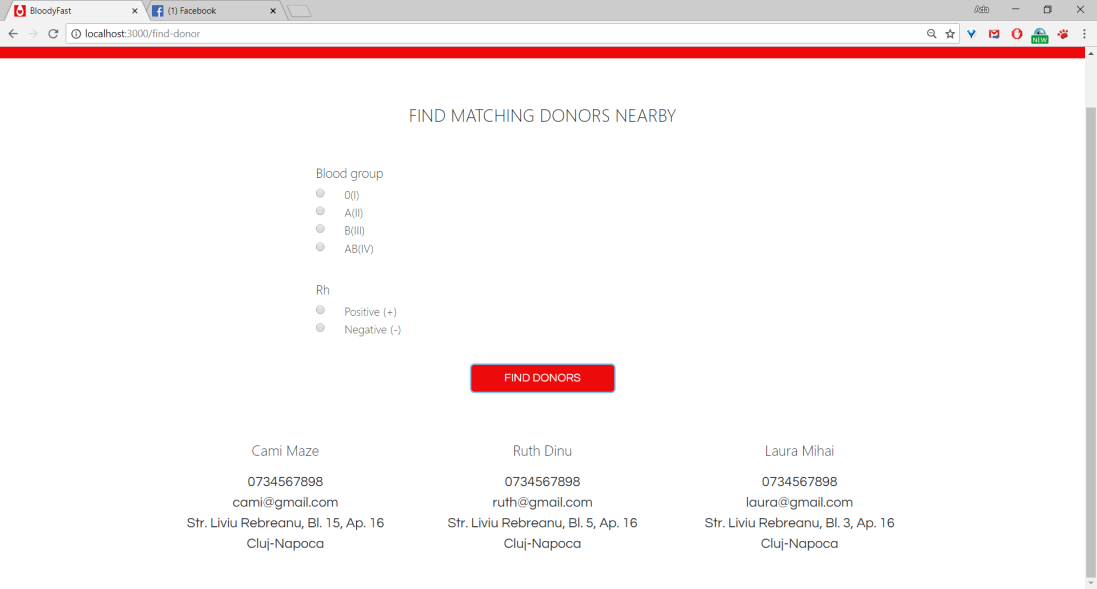
As a doctor you can click the New Request tab based on your needs for a specific type and quantity of blood; these requests can be managed through the My Requests page.



If you are part of the blood bank staff, you can manage the requests submitted by the doctors through the Requests tab, and supply them with the required blood, based on their needs and the blood’s availability. You can check that in the Stock section of your page, you can also see if the blood is usable, and discard it otherwise, or, if there has been a recent donation, add it to the stock.



Furthermore, if a certain resource is unavailable, click the Find Donor button in your navigation bar in order to find the nearest registered donor that matches your needs.



Lastly, as amember of the blood bank personnel, it is your responsibility to manage the results of donors’ blood tests, and that is available via the Tests page, where you can upload a patients’ results (either by selecting the files or via drag-and-drop from your computer), and they will be notified of them.

