



# Final Project Report

**SENG2011 19T3 Team EarlyDafny**

**Bowei Pan z5141863**

**Junjie Zhu z5158356**

**Tianyu Pan z5150836**

**Xiangyu Zeng z5137425**

**Yifan Dai z5172455**

# 1. Executive Summary

## 1. Who are we

WARMER is a blood management system which provides reliable and convenient services to publics and organisations. It will focus on handling the blood. The primary aim of the system to provide enough and absolutely safe blood when users request and relative information.

## 2. What we offer and the problem we solves.

Warmer handles sufficient and reliable blood to publics and organisations.

The system's clients are publics and organisations who want to donate blood and those who demand for blood.

Warmer offers a variety of services, for example:

### 1. Handle blood request

The request that stored in WARMER will be delivered to users with screening and testing by pathologies if the blood is fresh and valid.

### 2. Handle query

Users can also search relative information about blood.

### 3. Receive and test blood

It can receive blood from publics and organisations and testing them by pathologies.

### 4. Critical level actions

When stock is low can send warning messages when stock is nearly out or blood is out of date to Company staff and users at different levels, it can also send feedback of the blood to places where it comes from.

## 3. Target market

The system's main target is to help Vampire Pty Ltd solving continually arised problems such as:

1. blood going out-of-date before it can be used
2. running out of blood
3. poor record keeping
4. poor donor service.

## 4. Advantages

The system is a safety first system, it will always make sure the blood is fresh enough and have enough blood in stock.

It will receive safety blood tested by pathology.

It will keep track on blood received and delivered and record its information such as type of blood, arrival date, use-by-date, where it came from, and from whom.

When blood has some problems, it will send feedback to its providers.

When level of certain type of blood is going low, it will base on the level to take relative actions to warn Company Staff.

## 2. Requirements

For the purpose of this report, we define some stakeholders:

1. Warmer can be defined as this blood management system.
2. Organizations can be defined as any medical facilities such as hospitals and clinics.
3. Donors can be defined as anyone with a government issued id number that can uniquely identify the person.
4. Pathologies can be defined as the blood testing companies.
5. Bat-mobile can be defined as all organised mobile blood service which collect blood from donors.
6. Government employee can be defined as any donor who works for the government and likely to donate blood when there is insufficient blood.
7. Red Cross can be defined as a special organisation who can deposit blood to Vampire in case of critical blood amount.
8. MWHC can be defined as the medical waste handling company that can safely handle expired blood.

### Blood management system

**[priority 1]** Req1: handle blood requests from Organizations.

**[priority 1]** 1.1 Organizations can request blood.

- **[priority 1]** 1.1.1 Organizations can specify the type of blood to request.
- **[priority 1]** 1.1.2 Organizations can specify the number of bags of blood to request.
- **[priority 1]** 1.1.3 Organizations can specify their information (Name, Address, Phone number).

**[priority 1]** 1.2 Vampire must send good blood to organizations that have requested blood.

- **[priority 1]** 1.2.1 Vampire can obtain the organization name, address, phone number, and number of bags, type of blood from Warmer.
- **[priority 1]** 1.2.2 Warmer must ensure the blood to send is good blood and before use-by date.

**[priority 2]** Req2: handle donor's queries.

**[priority 3]** 2.1 Donors can query public donating information.

- **[priority 3]** 2.1.1 Donors can query donating location.
- **[priority 3]** 2.1.2 Donors can query donating time.

**[priority 2]** 2.2 Donors can query private donating information.

- **[priority 2]** 2.2.1 Donors can query historical donating location and time of themselves.
- **[priority 2]** 2.2.2 Donors can query whether their blood has passed the blood testing.
- **[priority 2]** 2.2.3 Donors can query the feedback of their donated blood.

**[priority 1]** Req3: blood donating and testing

**[priority 1]** 3.1 Bat-mobile can deposit untested blood to Vampire.

- **[priority 1]** 3.1.1 Vampire must mark the blood collected in this way as "untested" (untested blood).

**[priority 1]** 3.2 Vampire must check for blood quality of untested blood.

- **[priority 1]** 3.2.1 Vampire can send untested blood to pathologies for testing.
- **[priority 1]** 3.2.2 Pathologies can specify the blood quality (either “good blood” or “bad blood”) and blood information (blood type, use-by date).

**[priority 1]** 3.3 Pathologies and Red Cross can deposit good blood to Vampire.

- **[priority 1]** 3.3.1 Vampire can receive bags of good bloods from Pathologies and Red Cross with blood informations (blood type, use-by date).
- **[priority 1]** 3.3.2 Vampire must mark the blood collected in this way as “good blood”.

**[priority 1]** Req4: inventory management

**[priority 2]** 4.1 Vampire is able to check the status of the inventory.

- **[priority 2]** 4.1.1 Vampire can get the amount of each specific type of blood.
- **[priority 2]** 4.1.2 Vampire can get the number of bags of each specific type of blood.
- **[priority 2]** 4.1.3 Vampire can list all donors who have donated blood to Bat-mobile with their personal information and donation history.

**[priority 1]** 4.2 Vampire can update the inventory.

- **[priority 1]** 4.2.1 Vampire can add blood that comes from Bat-mobile, pathologies, Red Cross to the inventory.
- **[priority 1]** 4.2.2 Vampire can set the origin of a bag of blood (including state, test result, use-by date).

**[priority 2]** 4.3 Vampire can discard expired blood.

- **[priority 2]** 4.3.1 Vampire can list all the expired blood in the inventory.
- **[priority 2]** 4.3.2 MWHC can collect expired blood from Vampire and safely handle it.

**[priority 1]** 4.4 Warmer should take actions when the number of bags of a type of blood is critical.

- **[priority 1]** 4.4.1 lower than 200 bags: Vampire sends email to the donors whose blood type match the insufficient blood.
- **[priority 1]** 4.4.2 lower than 100 bags: Vampire sends email to government employees to advocate them to donate blood if they are that type of blood.
- **[priority 1]** 4.4.3 lower than 50 bags: Vampire sends blood request to Red Cross so that it could deposit Vampire blood.

**[priority 3]** Req5: Account and authentication

**[priority 3]** 5.1 Donors can register and login

- **[priority 3]** 5.1.1 Donors must be able to login if they provide correct username and password
- **[priority 3]** 5.1.2 Donors can register an account.

**[priority 3]** 5.2 Organization can register and login.

- **[priority 3]** 5.2.1 Organizations must be able to login if they provide correct username and password.
- **[priority 3]** 5.2.2 Organizations can register an account.

**[priority 4]** 5.3 Registered donors can update their account information, including username, password, email.

**[priority 4] 5.4 Registered organizations can update their account information, including username, password, organization name, email, address.**

### **Assumption:**

- 1. Bat-mobiles give Vamipre blood in unit of bag (1 bag per donor).**
- 2. The blood from Red Cross is reliable with the testing result “good” and not expired so doesn’t need to be tested.**
- 3. The origin of all the blood (donor’s name, time and location of collection) is known and will be correctly stored.**
- 4. Assume that the Red Cross is always be able to provide Vampire sufficient blood that can increase the blood level to non-critical level.**
- 5. The capacity of inventory to store blood is unlimited.**
- 6. The blood will be used as soon as organizations request it, so that it will not expire after it is requested.**
- 7. Assume that vampire staff correctly update blood states so that they remain consistent. (e.g, “disposed” blood cannot become “in inventory”)**

# 3. Use-Cases

## Use Case 1: (Req1)

Request blood

**Actor:**

Organisations, Vampire

**Post-condition:**

Organization has received the requested blood.

**Basic Flow:**

1. Organization goes to blood request page.
2. Warmer displays a page showing the number of bags of blood and all types of blood, with text boxes to accept inputs of organization's name, address, phone number, and volume of blood to request.
3. Organization enters all information (described above) and click request button.
4. Warmer displays another page that lists all bags of blood with their use-by date.
5. Warmer notify Vampire staff which bags of blood in the inventory to send, and the information of the organization that has requested the blood.
6. Vampire staff send blood according to the notification to the organization.

**Alternative Flow 1:**

2.1 Warmer displays an error message beside all the blank text boxes, asking organization to complete all the necessary information.

**Alternative Flow 2:**

2.2 Warmer displays an error message beside the volume text box if the volume is less than 1 or greater than the maximum number of bags of good blood that are before use-by date.

## Use Case 2: (Req3, Req4.2)

Donate blood

**Actor:**

Bat-mobile, Pathology, Vampire staff

**Pre-condition:**

Bat-mobile has blood donated from donors.

**Post-condition:**

Vampire has received tested blood which is "good blood".

**Basic Flow:**

1. Bat-mobile gives Vampire staff the bag of blood collected from donors.
2. Vampire staff adds the blood and set its origin to the inventory.
3. Warmer should send a notification to another Vampire staff to send the blood for testing, the notification indicates which bags of blood in the inventory will be sent.
4. Vampire staff find that blood from the inventory and send it to a Pathology for testing.
5. The pathology returns the good blood to Vampire.
6. Vampire changes the attribute of the blood from "untested" to the test result ("good blood")

**Alternative Flow 1:**

5.1 Pathology throws away the bad blood and send the test result to Vampire.

6.1 Vampire changes the attribute of the blood from "untested" to "bad blood".

7.1 Pathology contacts donor's GP and send him the test result.

## Use Case 3: (Req2)

Donor queries inventory

**Actor:**

Donors

**Post-condition:**

Donor gets requested information

**Basic Flow:**

1. Donor goes to donor information query page.
2. Warmer displays two links (public information, private information).
3. Donor clicks "private information".
4. Warmer displays a donor id textbox.
5. Donor enters his id and click submit button.
6. Warmer displays all donating history corresponding to this donors, each of which includes time and location of donating, test result and an optional feedback.

**Alternative Flow 1:**

- 2.1 Donor clicks "public information"
- 3.1 Warmer displays available time and location of all Bat-mobiles in the next 7 days.

**Alternative Flow 2:**

6.2 Warmer displays a page saying "You don't have any donating history" if the donor has never donated blood.

## Use Case 4: (Req4.1, Req4.4, Req3.3)

Critical level action

**Actor:**

Vampire staff

**Pre-condition:**

Blood level is critial.

**Post-condition:**

Blood level is no longer critical.

**Basic Flow:**

1. When the amount of blood of any type is lower than 2000 bags, Warmer raises an alarm to notify Vampire staff.
2. Vampire staff checks the blood amount of that type.
3. Vampire staff lists donors who matches the type of blood from the inventory and sends emails to them to advocate them to donate blood.

**Alternative Flow 1:**

3.1 If the blood level is lower than 1000 bags, Warmer sends emails to government employees to advocate them to donate blood if they are that type of blood.

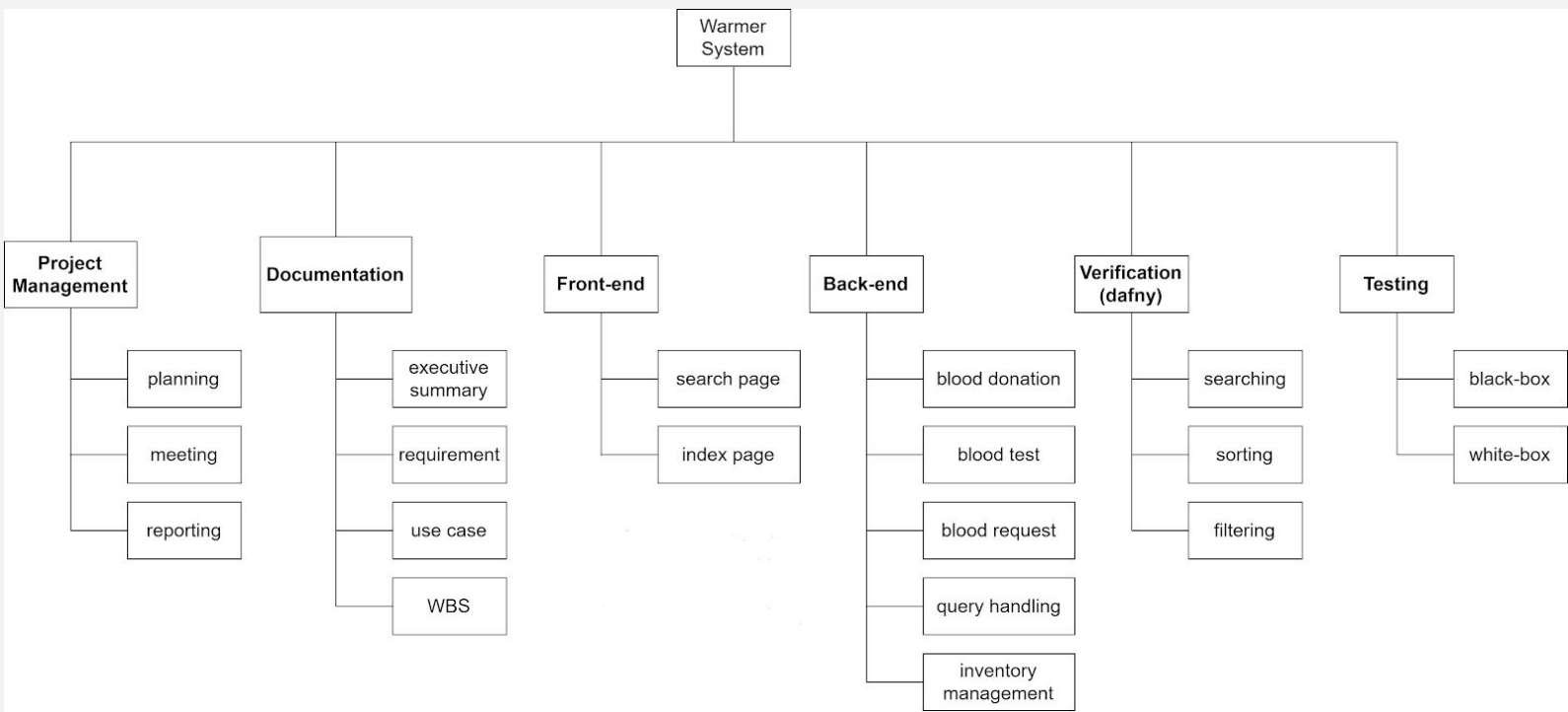
**Alternative Flow 2:**

3.2 When the blood level is lower than 500 bags, Vampire sends blood request to Red Cross.

4.2. Red Cross sends at lease 2000 bags of blood to Vampire.

5.2. Vampire adds these bags of blood to the inventory and specifies they are from Red Cross, so that Warmer marks them as good blood.

# 4. Work Breakdown Structure



Back-end breakdown verision (See initial report)



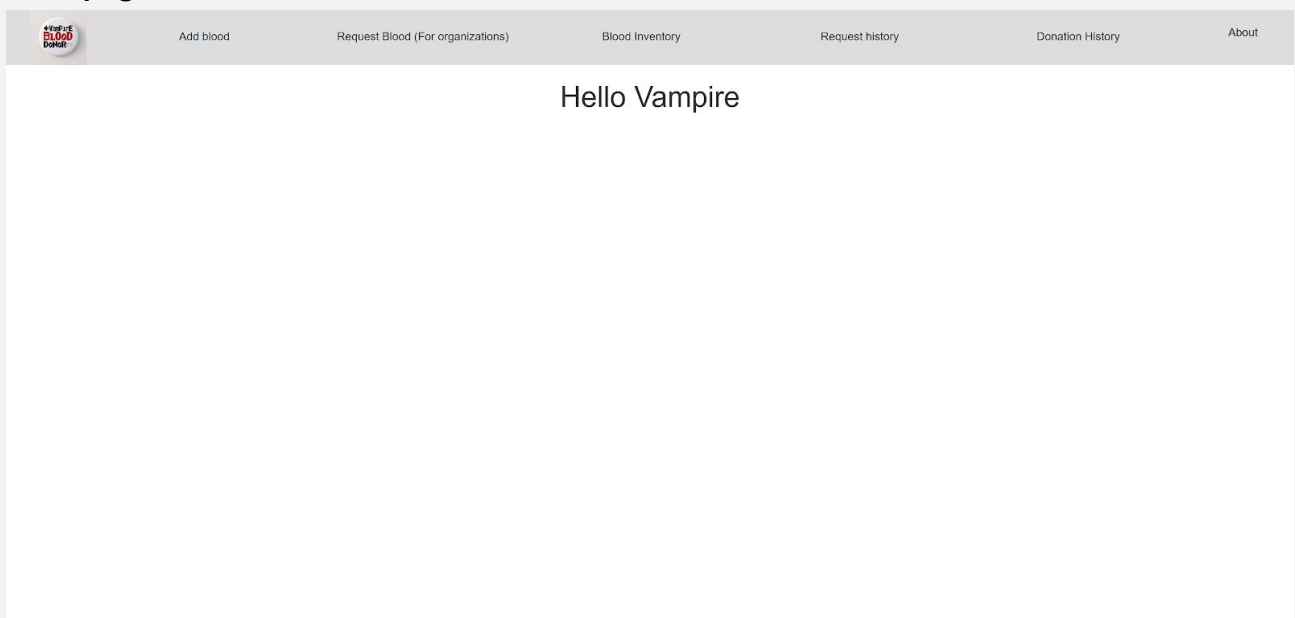
## 5. Front-End Implementation

HTML: HTML has been used as basic framework of WARMER, all pages, links, navigations has been implemented by HTML

CSS: CSS has been used to increase visual effect to the base HTML on the site. Bootstrap was used as visual framework to enhance the basic HTML.

JS: JQuery has been used to enhance the code style and other javascript functions have been developed to suit specific use and some HTML tags were generated by js. And we've tried to transfer UTC seconds into readable text. Ajax is used for dynamically loading at inventory page.

### Homepage




That is a welcome page for all users.

For Vampire staff, they can add blood and store the data in inventory.

Organization can request blood from Vampire.

Vampire staff can manage the blood in inventory and check the request history, with the quality of notifying MWHC to dispose the bad blood or expired blood.

## Add bat mobile blood



Add blood

Request Blood (For organizations)

Blood Inventory

Request history

Donation History

About


### Add blood

Donor name:

Donor ID:

Source: Bat-Mobile

This page is only for Vampire Staff to add untested blood from bat mobile to inventory by entering the information of donor such as donor name and donor ID.



Add blood

Request Blood (For organizations)

Blood Inventory

Request history

Donation History

About

### Add blood

Donor name:


Donor ID:

Source: Bat-Mobile

submitted, blood id: 212

After adding untested blood, the page would show the blood id in inventory.

## Add Red cross add blood



Add blood

Request Blood (For organizations)

Blood Inventory

Request history

Donation History

About

### Add blood

Donor name:

Hospital

Donor ID:

RedC

Blood type:

AB

Use by:

Jan 01 2020 10:00:00

Source:


Red Cross

Submit

submitted, blood id: 213

This page is also for Vampire Staff. Adding blood in this way, the blood would be marked as passing the test by entering the information of blood such as donor name, donor ID, Blood type and use by date.

## Request blood



Add blood

Request Blood (For organizations)

Blood Inventory

Request history

Donation History

About

### Request blood

Blood available for requesting

Type	Amount (bags)
A	110
B	2
AB	2

Type:

A

Amount:

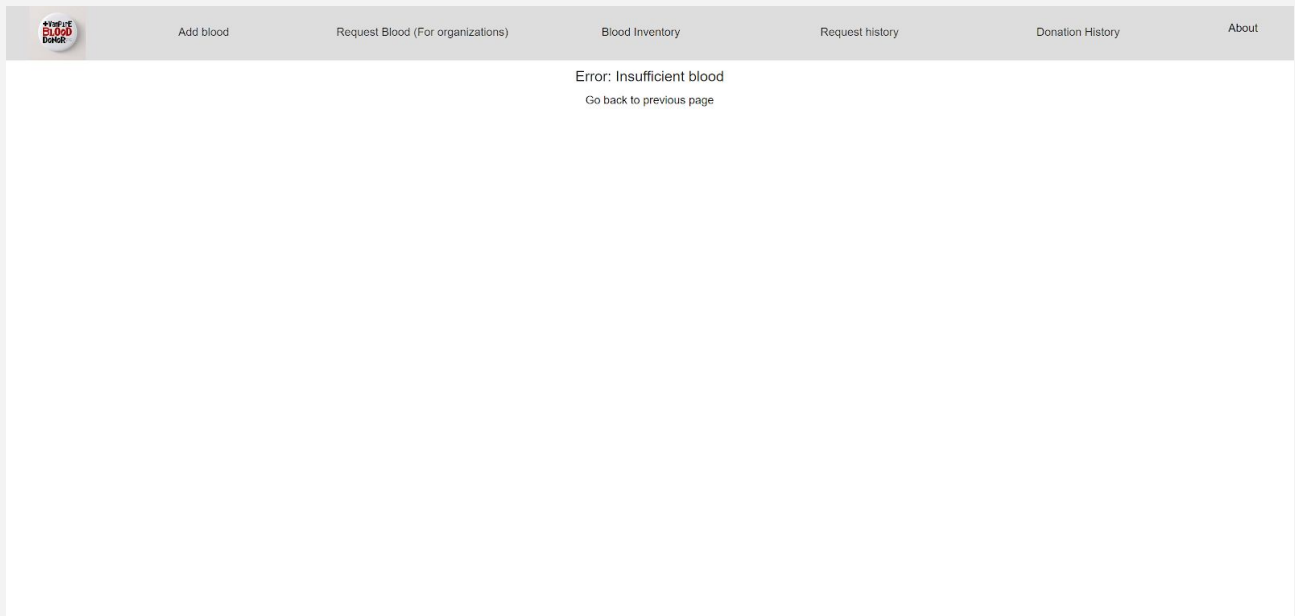
Organization name:

Organization address:

Organization phone:

Submit

For organisations, they can request blood by entering the amount of blood, blood type and the information of organization. If request is successful, the information of request would pass to the request history.



It would show error message, if the amount of request is bigger than that in inventory.

## Inventory

A screenshot of the 'Blood Inventory' page. The navigation bar is the same as in the previous image, with 'Blood Inventory' selected. The main content area has the title 'Blood Inventory' at the top. Below the title, there is a small table showing blood types and their amounts in bags. This is followed by a 'Filter by' section with input fields for 'ID' and 'Type', and dropdown menus for 'Expired', 'State', 'Test State', and 'Order by'. At the bottom, there is a large table listing individual inventory items with columns for Id, Add time, Use By, State, Test state, Feedback, Type, Donor name, and Donor ID.

Type	Amount (bags)
A	210
B	2
AB	2

Filter by

ID

Type

Expired ☐

State: None ▾

Test State: None ▾


Order by  
None ▾ None ▾

Id	Add time	Use By	State	Test state	Feedback	Type	Donor name	Donor ID
0	Nov 22 2019 12:17:01	Jan 03 2020 19:29:53	in inventory	Good blood		A	Miku0	0
1	Nov 22 2019 12:17:01	Jan 03 2020 19:29:54	in inventory	Good blood		A	Miku1	1
2	Nov 22 2019 12:17:01	Jan 03 2020 19:29:55	in inventory	Good blood		A	Miku2	2

Vampire staff can search blood by ID, Type, State, Test State

Vampire staff can also check the blood which is needed to be disposed in this page. For example, Vampire staff can filter the expired blood and bad blood that needed to be disposed, then notify MWHC to disposed the bad blood.

## Update Status



Add bloodRequest Blood (For organizations)Blood InventoryRequest historyDonation HistoryAbout

Blood 0

Use By Date:

Jan 03 2020 19:29:53

Feedback:

Everything is fine

Type:

A

State:

In inventory ▾

Test State:


Good ▾

Submit

Vampire staffs can update the information of blood.

For Vampire staffs, they should be able to mark the state to be disposed, which means the blood is bad and notified MWHC to disposed it.

## Request History



Add bloodRequest Blood (For organizations)Blood InventoryRequest historyDonation HistoryAbout


Request history

ID	time	Type	Num of bags
0	Nov 22 2019 12:28:13	A	100
1	Nov 22 2019 12:28:21	B	1
2	Nov 22 2019 12:28:28	AB	1

The record of blood requested by organizations stores in request history page.

If click on the ID number, it should direct to the detail of that record.

## Request Info

Add bloodRequest Blood (For organizations)Blood InventoryRequest historyDonation HistoryAbout

### Request


Requested blood

Blood id	Use By	State	Test state	Feedback	Type
210	Jan 01 2121 10:00:00	used	Good blood		B

Organization name: pc  
Organization address: 6a  
Organization phone: 123456

In this page, staff can check the detail of requested blood and the detail of organization such as name, address and phone number.

## Donation History

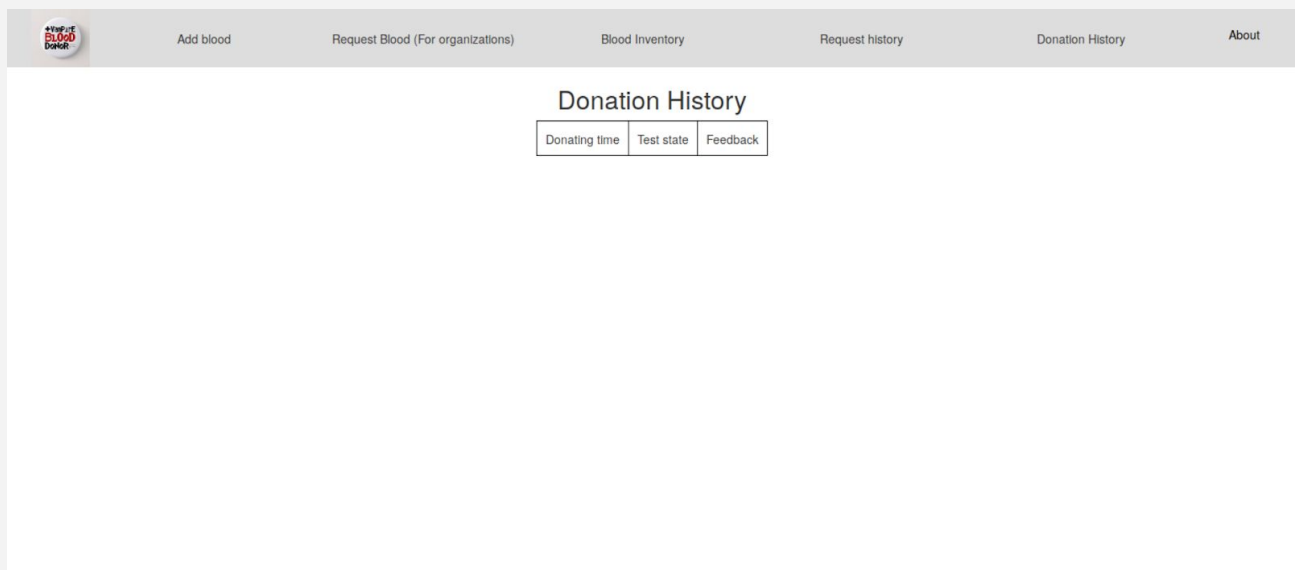
Add bloodRequest Blood (For organizations)Blood InventoryRequest historyDonation HistoryAbout

### Donation History

Donor name:

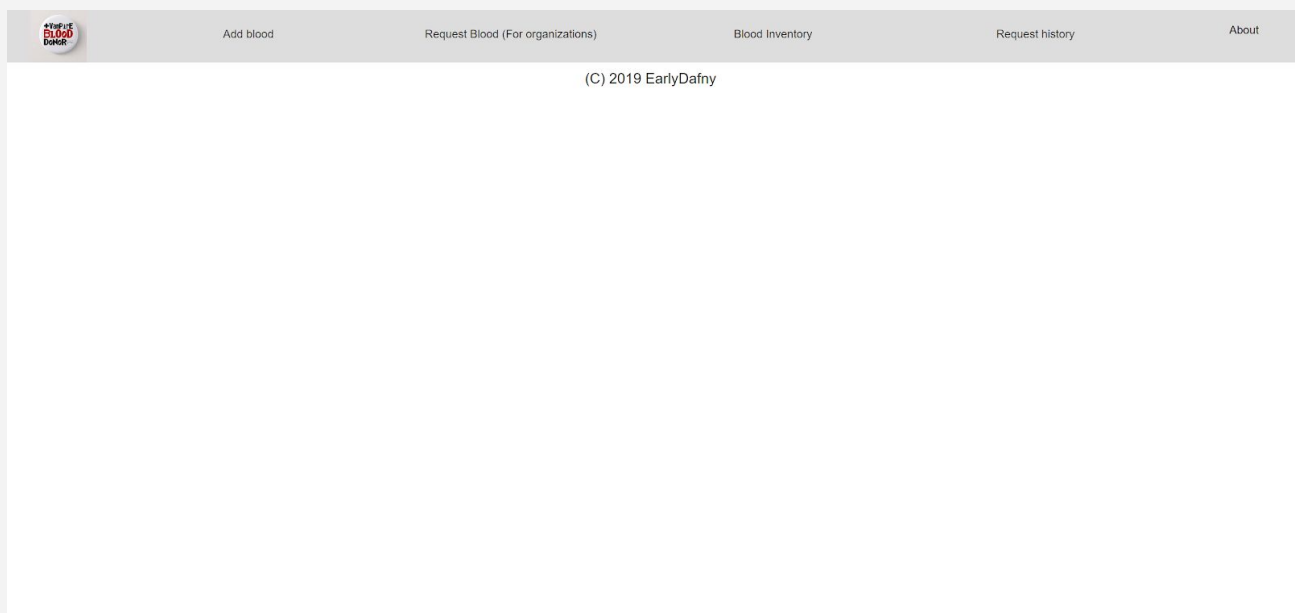
Donor ID:

In this page, donors can check their donation history by insert their names and donor IDs.(paritally implemented)



This page will show the donation history including donation time, test states(not tested, good, bad) and feedback.

## About



This page shows our team name.

## 6. Back-End Implementation

### Python

Since Python is one of the most useful language to implement the backend, we used it to class all the objects and provide all the necessary backend API. Using Flask which is a micro framework to implement our website.

### Flask

Flask was combined with Python and Jinja2 to create the dynamic website. It has been an effective tool to connect to and from back-end analysis data from frontend and back again through Jinja, its simplistic also help us in this project.

In order to ensure the correctness of sorting and searching functions in Python,Dafny was used to verify them.

### Architecture:

Blood class: refers to Blood.py

Represents a bag of blood.

Has the following attributes:

- donor: donor object
- add\_time: the time when this blood is added to the inventory
- use\_by: the time that this bag of blood will expire
- state: indicates the state of this blood. It could be:
  - IN\_INVENTORY (value = 1): the blood is inside the inventory
  - TESTING (value = 2): the blood is being testing
  - USED (value = 3): the blood is used by an organization
  - DISPOSED (value = 4): the blood is disposed by vampire or pathologies
- test\_state: indicates the test state of the blood:
  - NOT\_TESTED (value = 1): The blood is not tested
  - GOOD (value = 2): The test result is good blood
  - BAD (value = 3): The test result is bad blood
- feedback: optional feedback from pathologies
- type: blood type. Empty string implies unknown type (pathologies will determine the type)
- id: blood id

Donor class: refers to Donor.py

Represents a donor.

Has the following attributes:

- name: donor's name
- id: donor's id
- history: donor's donation history

Organization class: refers to Organization.py

Represents an organization.

Has the following properties:

- name: organization name
- address: organization address



- phone: organization's phone number

Request class: refers to Request.py

Represents a successful blood request

Has the following properties:

- org: Organization that requests the blood
- bloods: list of Blood object that is requested
- time: when this request is made
- id: id of this request

LevelChecker class: refers to LevelChecker.py

It can check the level of requestable blood of each known type in the inventory and take corresponding actions when the level is critical (described in the requirement). The actions is simulated by printing a line to the console.

Inventory class: refers to Inventory.py

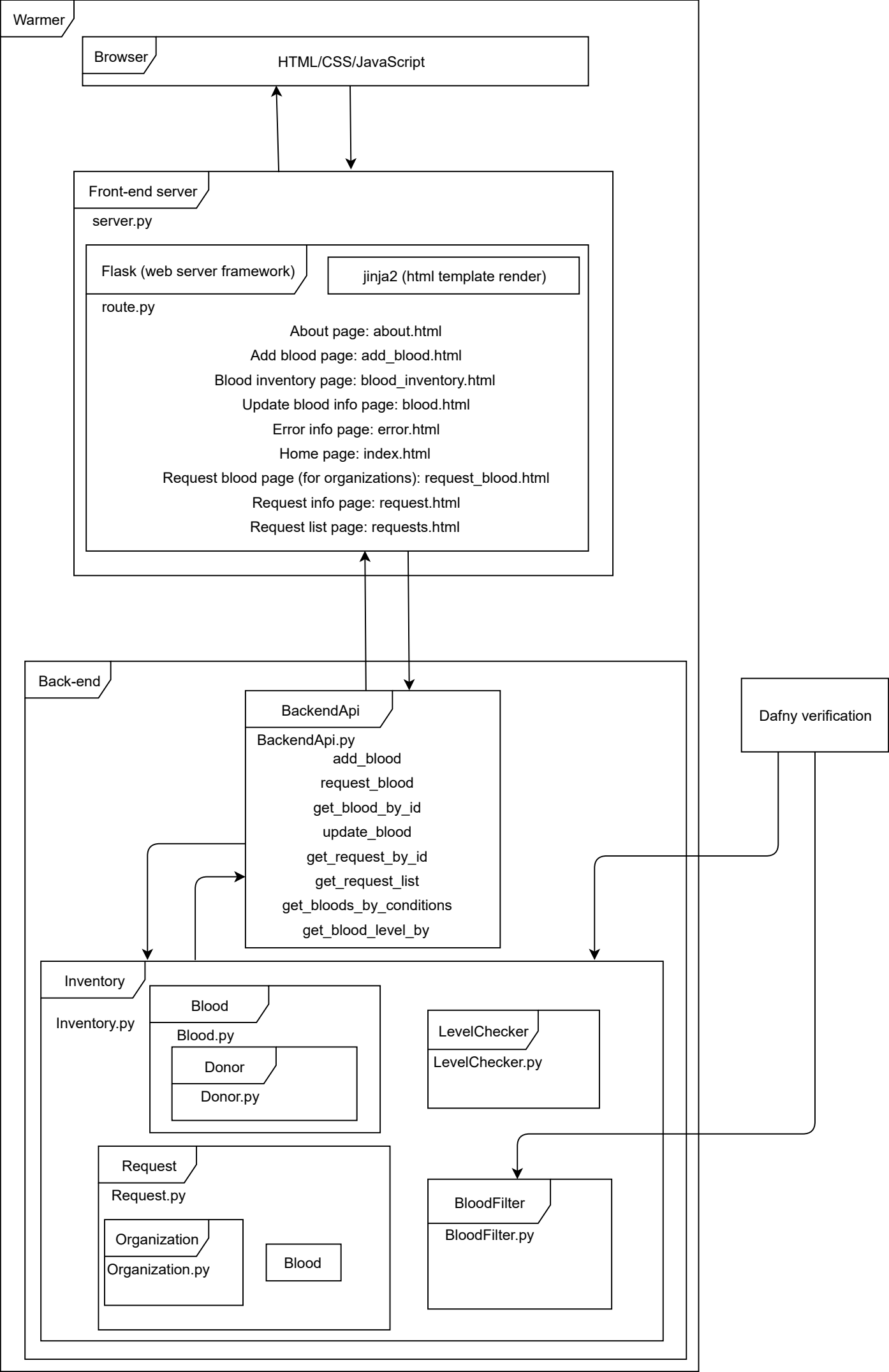
All the information (i.e. blood, donor, organization, request) is stored in the inventory. It implements all the functionality that the api is using. Blood Filter which has been verified by dafny is used to do the critical tasks. It also checks the blood level using Level Checker every a few seconds and when any blood is updated (e.g. add new blood, request blood), to make sure the level info is up-to-date. Assumption that the low level actions will be immediately taken so that the blood level is assumed to be restored above the critical level to ensure the blood supply.

BackendApi: refers to BackendApi.py

It exposes the following apis to the front-end:

1. add\_blood:
  - Allows front-end to add blood from source Batmobile or Red Cross (one bag each time).
  - Required input: donor\_name and donor\_id (both source), blood\_type and use\_by (Red cross only)
  - If success, it returns the id of the new blood, otherwise it returns error message if donor's info is missing
2. request\_blood:
  - Allows front-end to request blood (in unit of bags)
  - Required input: blood type (selected from a drop down list, available blood type only), number of bags, organization name, address, phone.
  - If success, it returns a list of blood objects. If organization's info is missing or invalid/insufficient blood amount, it returns an error message.
3. get\_blood\_by\_id:
  - Allows front-end to get blood info by blood id.
  - Required input: blood id
  - If success, it returns a serialized blood object (a python dictionary) to represent blood info, Otherwise, it returns an error message if blood of id is not found.
4. update\_blood:
  - Allows front-end to update the info of a specific bag of blood.
  - Required input: id, use\_by, state, test\_state, feedback, type
  - If fails, it returns an error message (blood not found), otherwise, it returns true
5. get\_request\_by\_id:

- Allows front-end to get a blood request by its id.
  - Required input: id of the request
  - If success, it returns a serialized request object (a python dictionary) to represent request info, Otherwise, it returns an error message if request of this id is not found.
6. `get_request_list`:
- Allows front-end to get a list of blood request
  - Required input: none
  - Returns a list of serialized request objects
7. `get_bloods_by_conditions`:
- Allows front-end to get a list of blood by some user-specific conditions (e.g. type, id)
  - Required input: filter options (type, isexpired, state, test\_state), sorting options (order by use by time or add time, ascending or descending)
  - Returns a list of serialized blood objects
8. `get_blood_level_by`:
- Allows front-end to get a summary of blood leve.
  - Required input: cat (group by, currently only support "type"), filter (no filter or filter the requestable blood).
  - Returns a level summary of each type of blood.
9. `get_blood_by_donor`:
- Allows front\_end get a list of blood which is the history of a donor's donation
  - Requires input: donor name, donor id
  - Returns a list of blood



## Verification

The important algorithmic functions in our project are verified using dafny. We defined a simplified blood class and inventory class to make sure the python code and dafny code are implemented as closed as possible. Non-critical data (like donor's name, organization address) has been removed. Since BloodFilter involves lots of algorithms, it has been implemented using dafny.

Req1 (Organizations request blood) has been verified. The corresponding function is implemented in Inventory.py request\_blood. It takes number of bags (n\_bags) and type of blood as parameter, and finds the requestable blood (good blood, not expired, in inventory), then marks these bags of blood as used. If n\_bags is greater than the level of requestable blood, it returns an empty list. We try to model this function using dafny in Inventory.dfy request\_blood, other functions in BloodFilter.dfy are also involved.

Req 4.2, 3.1.1, 3.2.2, 3.3.1, 3.3.2 (update inventory) has been verified. The corresponding function is implemented in Inventory.py update\_blood. It takes blood id and some new attributes of blood object (e.g. use\_by time, state) and update the blood object of the given id using these new attributes. Assumption has been made that the vampire staff only specify correct attributes and state so that the inventory remains consistent. Dafny code to verify this function is implemented in Inventory.dfy update\_blood.

Algorithms of searching, sorting and filtering are implemented using dafny in BloodFilter.dfy. The implementations match exactly the python functions of the same name. Most of the apis (and some requirement) are using these python functions so they are also partly verified.

## 7. Evaluation

### Requirements Status Table

Requirement	Y = implemented N = not implemented P = partially implemented	Y = Verified N = Not Verified P = partially verified
<b>1.1 Organizations can request blood.</b>	Y	Y
<b>1.1.1 Organizations can specify the type of blood to request.</b>	Y	Y
<b>1.1.2 Organizations can specify the number of bags of blood to request.</b>	Y	Y
<b>1.1.3 Organizations can specify their information (Name, Address, Phone number).</b>	Y	Y
<b>1.2 Vampire must send good blood to organizations that have requested blood.</b>	Y	Y
<b>1.2.1 Vampire can obtain the organization name, address, phone number, and number of bags, type of blood from Warmer.</b>	Y	Y
<b>1.2.2 Warmer must ensure the blood to send is good blood and before use-by date.</b>	Y	Y

Requirement	Y = implemented N = not implemented P = partially implemented	Y = Verified N = Not Verified P = partially verified
<b>2.1 Donors can query public donating information.</b>	N	N
<b>2.1.1 Donors can query donating location.</b>	N	N
<b>2.1.2 Donors can query donating time.</b>	N	N
<b>2.2 Donors can query private donating information.</b>	N	N
<b>2.2.1 Donors can query historical donating location and time of themselves.</b>	N	N
<b>2.2.2 Donors can query whether their blood has passed the blood testing.</b>	N	N
<b>2.2.3 Donors can query the feedback of their donated blood.</b>	N	N

Requirement	Y = implemented N = not implemented P = partially implemented	Y = Verified N = Not Verified P = partially verified
<b>3.1 Bat-mobile can deposit untested blood to Vampire.</b>	Y	N
<b>3.1.1 Vampire must mark the blood collected in this way as “untested” (untested blood).</b>	Y	Y
<b>3.2 Vampire must check for blood quality of untested blood.</b>	Y	N
<b>3.2.1 Vampire can send untested blood to pathologies for testing.</b>	Y	N
<b>3.2.2 Pathologies can specify the blood quality (either “good blood” or “bad blood”) and blood information (blood type, use-by date).</b>	Y	Y
<b>3.3 Pathologies and Red Cross can deposit good blood to Vampire.</b>	Y	N
<b>3.3.1 Vampire can receive bags of good bloods from Pathologies and Red Cross with blood informations (blood type, use-by date).</b>	Y	Y
<b>3.3.2 Vampire must mark the blood collected in this way as “good blood”.</b>	Y	Y

Requirement	Y = implemented N = not implemented P = partially implemented	Y = Verified N = Not Verified P = partially verified
<b>4.1 Vampire is able to check the status of the inventory.</b>	Y	N
<b>4.1.1 Vampire can get the amount of of each specific type of blood.</b>	Y	P
<b>4.1.2 Vampire can get the number of bags of each specific type of blood.</b>	Y	P
<b>4.1.3 Vampire can list all donors who have donated blood to Bat-mobile with their personal information and donation history.</b>	Y	P
<b>4.2 Vampire can update the inventory.</b>	Y	Y
<b>4.2.1 Vampire can add blood that comes from Bat-mobile, pathologies, Red Cross to the inventory.</b>	Y	Y
<b>4.2.2 Vampire can set the origin of a bag of blood (including state, test result, use-by date).</b>	Y	Y
<b>4.3 Vampire can discard expired blood.</b>	Y	N
<b>4.3.1 Vampire can list all the expired blood in the inventory.</b>	Y	P
<b>4.3.2 MWHC can collect expired blood from Vampire and safely handle it.</b>	Y	N



<b>4.4 Warmer should take actions when the number of bags of a type of blood is critical.</b>	Y	N
<b>4.4.1 lower than 200 bags: Vampire sends email to the donors whose blood type match the insufficient blood.</b>	Y	N
<b>4.4.2 lower than 100 bags: Vampire sends email to government employees to advocate them to donate blood if they are that type of blood.</b>	Y	N
<b>4.4.3 lower than 50 bags: Vampire sends blood request to Red Cross so that it could deposit Vampire blood.</b>	Y	N

Requirement	Y = implemented N = not implemented P = partially implemented	Y = Verified N = Not Verified P = partially verified
<b>5.1 Donors can register and login</b>	N	N
<b>5.1.1 Donors must be able to login if they provide correct username and password</b>	N	N
<b>5.1.2 Donors can register an account.</b>	N	N
<b>5.2 Organization can register and login.</b>	N	N
<b>5.2.1 Organizations must be able to login if they provide correct username and password.</b>	N	N
<b>5.2.2 Organizations can register an account.</b>	N	N
<b>5.3 Registered donors can update their account information, including username, password, email.</b>	N	N
<b>5.4 Registered organizations can update their account information, including username, password, organization name, email, address.</b>	N	N

## Reflection

We tried our best to focus on handling blood requests, blood donating and testing, inventory management and successfully implemented all priority 1 requirements. We first consider about blood inventory because most of our work needed to be done with the inventory, then we thought about handling request and donating & testing at the same time. We tried to make every states of blood clearly. (tested, untested, testing ...)

We also finished some of the the priority 2 requirements.

Since register and login are priority 3,4 requirements, we didn't put much effort on them.

For the implementation, we have too many states for blood and thus it is difficult to ensure the state is consistent (e.g. "disposed" blood cannot change state to "in inventory"). So the state consistency relies on the correct operation of vampire staff. Due to the limitation of time, not all important requirements are verified using dafny, and the test cases are not well-written.

We think we have tried our best to finish our job well, it is a little pity to us that we were kind of busy and didn't manage our time very well to make our project better.

## Teamwork

Junjie Zhu	Executive summary, requirement, use-cases, Dafny, final report
Tianyu Pan	Requirement, use-cases, front-end, Dafny
Xiangyu Zeng	Executive summary, requirement, use-cases, WBS, back-end, Dafny, final report
Yifan Dai	Executive summary, requirement, WBS, front-end test, final report
Bowei Pan	Requirement, use-cases, front-end