

SENG2011 19T3



Project Report

# Dafine

*Cher Ping Choy z5135745*

*Jiahui Luo z5158415*

*Ryan Cai z5210636*

*Wenlue Zhang z5158333*

*Yang Zheng z5210646*

# 0. Table of Contents

|  |           |
|--|-----------|
| <b>0. Table of Contents</b>                      | <b>2</b>  |
| <b>1. Executive Summary</b>                      | <b>4</b>  |
| 1.1 Background                                   | 4         |
| 1.2 Project Description                          | 4         |
| <b>2. Requirements</b>                           | <b>5</b>  |
| 2.0 Assumptions                                  | 5         |
| 2.1 Must-have                                    | 5         |
| 2.2 Should-have                                  | 5         |
| 2.3 Could-have                                   | 5         |
| 2.4 Won't-have                                   | 5         |
| <b>3. Use-Cases</b>                              | <b>7</b>  |
| 3.1 Must-have                                    | 7         |
| 3.1.1 Requirement 1                              | 7         |
| 3.1.2 Requirement 2                              | 8         |
| 3.1.3 Requirement 3                              | 10        |
| 3.1.4 Requirement 4                              | 12        |
| 3.2 Should-have                                  | 15        |
| 3.2.1 Requirement 1                              | 15        |
| 3.2.2 Requirement 2                              | 17        |
| 3.3 Could-have                                   | 21        |
| 3.3.1 Requirement 1                              | 21        |
| 3.3.2 Requirement 2                              | 23        |
| 3.3.3 Requirement 3                              | 25        |
| 3.4 Won't-have                                   | 27        |
| <b>4. Work Breakdown Structure</b>               | <b>28</b> |
| <b>5. Front-End Implementation</b>               | <b>29</b> |
| 5.1 Design and Implementation                    | 29        |
| 5.2 Front-end Walkthroughs:                      | 29        |
| 5.2.1 Top Level Menu                             | 29        |
| 5.2.2 Donation Manager                           | 29        |
| 5.2.3 Tester                                     | 32        |
| 5.2.4 Doctor                                     | 34        |
| 5.2.4.1 Doctor Reservation                       | 35        |
| 5.2.4.2 Doctor Cancel Reservation                | 36        |
| 5.2.4.3 Doctor Search For Blood by Type          | 36        |
| 5.2.5 Storehouse Manager                         | 37        |
| 5.2.5.1 Storehouse Manager Disposes Blood Sample | 37        |
| 5.2.5.2 Storehouse Manager Check Storage         | 38        |
| 5.2.5.3 Storehouse Manager Check Future Stock    | 39        |

|   |           |
|---|-----------|
| <b>6. Back-End Implementation</b>               | <b>40</b> |
| 6.1 Implementation                              | 40        |
| 6.2 Module Outline                              | 41        |
| 6.2.1 Blood                                     | 41        |
| 6.2.2 BloodList                                 | 42        |
| 6.2.3 Roles and Actions                         | 42        |
| 6.2.3.1 DonationManager                         | 42        |
| 6.2.3.2 Tester                                  | 42        |
| 6.2.3.3 Doctor                                  | 42        |
| 6.2.3.4 StorehouseManager                       | 43        |
| 6.3 Verification                                | 43        |
| 6.3.1 Coverage                                  | 43        |
| 6.3.2 Code Matching                             | 43        |
| 6.3.3 Limitations                               | 45        |
| <b>7. Evaluation</b>                            | <b>46</b> |
| 7.1 Requirements Status Table                   | 46        |
| 7.2 Work Distribution                           | 46        |
| 7.3 Challenge and Decisions                     | 47        |
| 7.3.1 Verification between Classes              | 47        |
| 7.3.2 User Interface                            | 47        |
| 7.3.3 Verification of List Operations           | 47        |
| 7.4 Further Improvements                        | 47        |
| 7.4.1 More Verification and Test Cases          | 47        |
| 7.4.2 Better User Interface                     | 48        |
| 7.4.3 More Functionalities and Less Assumptions | 48        |
| <b>8. Appendices</b>                            | <b>49</b> |
| 8.1 Use cases for Won't have                    | 49        |
| 8.1.1 Requirement 1                             | 49        |
| 8.1.2 Requirement 2                             | 50        |

# 1. Executive Summary

## 1.1 Background

Vampire Pty Ltd is a government-funded company that focuses on blood management. We receive blood from trusted medical facilities and mobile services. Donated blood is first sent to registered pathology clinics for testing. Once tested, the blood will be distributed to registered hospitals. Vampire ensures that all available blood is not kept past their expiry date and stock levels are updated in real-time. To better manage the blood, we have implemented state of the art verified software that ensures that all blood users will be safe when using blood from our blood banks.

## 1.2 Project Description

This project is designed around developing a system which can reliably manage the usage and storage of blood. The core purpose of this system is to ensure that the blood used by hospitals have been thoroughly tested and that all relevant information regarding the blood donation is recorded in the system accurately. Furthermore, our system is designed so that blood that is past its expiry date will be disposed of swiftly. Our system is also able to deal with queries on the level of supply, location and reservations for blood present in the blood bank.

In the first step of Vampire's process, blood will be retrieved from the donor via a **donation manager**. The donation manager will enter the required information regarding blood samples into the system. Once the blood has been collected, testing will be carried out by a registered pathology **tester**, who will enter the results of the test into the system. Samples that fail the test will be disposed of by the **storehouse manager**. This ensures that only blood which has passed the pathology tests will be used for transfusions. Samples that pass the test will be assigned an expiration date and blood type for further queries and usage. When there is a demand for blood, a **doctor** will make a query to the system for a specific type and volume of blood. To ensure the safety of the blood, our system will only provide samples within their expiration date. Blood that is past its expiration date will be removed from the storehouse by the **storehouse manager**.

## 2. Requirements

### 2.0 Assumptions

- User will always enter value correctly.
- Donation managers are assigned by third-party organisations.
- Transportation is conducted by third-party organisations.
- Transportation is always instantly finished.
- Tests are always reliable.
- Only registered medical facilities have access to request blood.
- Third-party will handle the disposal of wasted blood.
- When the storage of a certain type of blood is low, the donation manager will tend to retrieve more blood of that type.
- Users in the system are all registered and valid.
- Untested blood is transported to nearby available pathology for testing by third-party
- Tested blood is transported to storehouse by third-party.
- There is only one storehouse for the blood

### 2.1 Must-have

1. Donation managers can insert untested blood sample information into the system
2. Testers can test and fill in the details for blood samples (e.g. type, expiration date)
3. Doctors can view the storage of certain types of blood and obtain the blood from the blood bank.
4. Storehouse manager can remove disposed blood from the storehouse.

### 2.2 Should-have

1. Testers are able to start testing and fill in the results later.
2. When storage of blood is low, donor manager and storehouse manager can be notified by the system.

### 2.3 Could-have

1. Doctors can reserve the blood and reservation can be cancelled.
2. Donation manager can be notified if there is blood demonstrating a negative result in the test.
3. The amounts of blood should be measured in volume.

## 2.4 Won't-have

1. Donor could be recorded when they give blood and the donors can check their own record.
2. Patient can check the blood storage of certain types of blood and book the blood.
3. Donor will have more priority for using blood.

## 3. Use-Cases

### 3.1 Must-have

#### 3.1.1 Requirement 1

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | Insert untested blood.  |
| <b>Priority</b>           | Must-have   |
| <b>Requirement 1</b>      | Donation manager can insert a record of untested blood samples into the system  |
| <b>Actor</b>              | Donation Manager  |
| <b>Use Case Overview</b>  | Donation manager adds a record of untested blood into the system  |
| <b>Trigger</b>            | New blood arrives at the storehouse   |
| <b>Precondition 1</b>     | The blood has been transported to the storehouse  |
| <b>Basic Flow</b>         | <p>Main success scenario:</p> <ol style="list-style-type: none"><li>1. Donation manager transports the new blood into the storehouse</li><li>2. Donation manager enters the date of retrieval, donor information of blood</li><li>3. The system assigns a unique id and untested state.</li></ol>   |
| <b>Alternative Flow 1</b> | <p>Failed scenario when the donation manager enters invalid date format or invalid donor information:</p> <ol style="list-style-type: none"><li>1. Donation manager enters the invalid date of retrieval or donor information of blood</li><li>2. The system captures the error and informs the donation manager the incorrect field(s)</li></ol> |
| <b>Business Rules</b>     | <ul style="list-style-type: none"><li>• Every sample of blood in the storehouse has a unique id.</li><li>• Blood is untested</li></ul>  |
| <b>Postcondition 1</b>    | Successfully stored blood has a unique id and untested state.   |

|                      |  |
|----------------------|--|
| <b>Use Case 2</b>    | View the inserted blood  |
| <b>Priority</b>      | Must-have  |
| <b>Requirement 2</b> | Donation manager adds a record of untested blood into the system |

|                           |   |
|---------------------------|---|
| <b>Actor</b>              | Donor Manager   |
| <b>Use Case Overview</b>  | Donor manager can view a list of test records corresponding to the untested blood that just inserted into system  |
| <b>Trigger</b>            | Donor manager wants to know the blood inserted  |
| <b>Precondition 1</b>     | Donor manager knows his role  |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters command for untested blood</li> <li>2. The system returns a list of blood with id and retrieved date</li> </ol>   |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that no blood is related to the manager:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters command for untested blood</li> <li>2. The system notifies the donor manager that there is no blood to show</li> </ol> |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Only untested blood is displayed</li> <li>• Untested blood has status “untested”</li> </ul>  |
| <b>Postcondition 1</b>    | Donor manager learns ID of untested blood   |

### 3.1.2 Requirement 2

|                          |  |
|--------------------------|--|
| <b>Use Case 1</b>        | View and select untested blood   |
| <b>Priority</b>          | Must-have  |
| <b>Requirement 2</b>     | Tester can test and fill the details for blood samples (e.g. type, expiration date)  |
| <b>Actor</b>             | Tester   |
| <b>Use Case Overview</b> | Tester can view a list of untested blood and choose one to test  |
| <b>Trigger</b>           | Untested blood has been stored into the storehouse   |
| <b>Precondition 1</b>    | The tester is registered in the system   |
| <b>Basic Flow</b>        | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Tester enters a query to check for untested blood</li> <li>2. There should be a list of untested blood ordered by the date of retrieval</li> <li>3. The list also shows the id of blood</li> </ol> |



|                           |  |
|---------------------------|--|
| <b>Alternative Flow 1</b> | <p>Scenario when there is not any untested blood:</p> <ol style="list-style-type: none"> <li>1. Tester enters a query to check untested blood</li> <li>2. The system will display “No untested blood currently available”</li> </ol> |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Every blood sample will only be tested once.</li> <li>• Selected blood is in untested state</li> </ul>  |
| <b>Postcondition</b>      | Details of selected blood can be viewed by Tester  |

|                           |   |
|---------------------------|---|
| <b>Use Case 2</b>         | Test the blood  |
| <b>Priority</b>           | Must-have   |
| <b>Requirement 2</b>      | Tester can test and fill the details for blood samples (e.g. type, expiration date)   |
| <b>Actor</b>              | Tester  |
| <b>Use Case Overview</b>  | Tester tests the type and expiry date of blood  |
| <b>Trigger</b>            | Tester selected an untested blood   |
| <b>Precondition 1</b>     | The selected blood is entered   |
| <b>Precondition 2</b>     | Tester is registered in the system  |
| <b>Basic Flow</b>         | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Tester performs a test for selected blood that is currently un-tested</li> <li>2. If the test succeeds, tester adds type and expiry date of that blood</li> <li>3. The status of blood change to tested</li> <li>4. There should be a notice to indicate that the blood has been tested successfully</li> </ol> |
| <b>Alternative Flow 1</b> | <p>Scenario when the test result of the blood is negative:</p> <ol style="list-style-type: none"> <li>1. Tester add test command for certain blood</li> <li>2. The test fails, tester add command to dispose</li> <li>3. The status of blood change to disposed</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Blood type stays the same</li> <li>• Any blood that fails the test is disposed.</li> <li>• Any blood that passes the test should have known expiry date and type.</li> </ul>   |
| <b>Postcondition</b>      | Either the status of the blood is changed to “tested” with specified type and expiry date, or the blood has been disposed.  |

### 3.1.3 Requirement 3

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | View list of blood  |
| <b>Priority</b>           | Must-have   |
| <b>Requirement 3</b>      | Doctor can view the storage of certain types of blood and obtain the blood.   |
| <b>Actor</b>              | Doctor  |
| <b>Use Case Overview</b>  | Doctor can view a list of blood of certain type   |
| <b>Trigger</b>            | Doctor wants to check the storage of a certain type of blood  |
| <b>Precondition 1</b>     | The doctor is registered in the system  |
| <b>Basic Flow</b>         | <p>Main success scenario, there is enough fresh blood in the system:</p> <ol style="list-style-type: none"> <li>1. The doctor enters a query which contains the type of blood they want to view</li> <li>2. The list of tested blood of that type shows in the system ordered by date of expiry</li> <li>3. The id of blood also shows in the list</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation when no record of that type of blood is in the system:</p> <ol style="list-style-type: none"> <li>1. Doctor add the command which contains the type of blood to ask for a certain type of blood</li> <li>2. "No record" is displayed to inform the Doctor</li> </ol>   |
| <b>Alternative Flow 2</b> | <p>This scenario describes the situation when the doctor enters an invalid type of blood:</p> <ol style="list-style-type: none"> <li>1. Doctor add the command which contains the type of blood to ask for a certain type of blood</li> <li>2. "Invalid blood type" is displayed to inform the Doctor</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• The blood shows in the list must have been tested</li> <li>• The blood shows in the list must be unexpired</li> <li>• The blood shows in the list must be available to use</li> </ul>  |
| <b>Postcondition 1</b>    | Doctor knows how much blood of a certain type is currently in storage   |
| <b>Postcondition 2</b>    | Doctor knows the id of available blood  |

|                           |  |
|---------------------------|--|
| <b>Use Case 2</b>         | Obtain the blood   |
| <b>Priority</b>           | Must-have  |
| <b>Requirement 3</b>      | Doctor can view the storage of certain types of blood and obtain the blood.  |
| <b>Actor</b>              | Doctor   |
| <b>Use Case Overview</b>  | Doctor can obtain certain blood  |
| <b>Trigger</b>            | Doctor need to use certain blood   |
| <b>Precondition 1</b>     | Doctor already knows the id of the blood they want ( if don't know, view list of blood first   |
| <b>Basic Flow</b>         | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Doctor type in obtain command which includes the id of blood they want</li> <li>2. There is a notice that indicates that the obtain command succeeded</li> <li>3. Doctor can then take out the blood from the blood bank</li> </ol>                                  |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation when an invalid id of blood is entered:</p> <ol style="list-style-type: none"> <li>1. Doctor type in obtain command which includes the id of blood they want</li> <li>2. There should be a notice with the words "Invalid ID, please try again"</li> </ol>  |
| <b>Alternative Flow 2</b> | <p>This scenario describes the situation when the selected blood is no longer available:</p> <ol style="list-style-type: none"> <li>1. Doctor type in obtain command which includes the id of blood they want</li> <li>2. There should be a notice with the words "The required blood is used or disposed, please use another id"</li> </ol> |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• The blood that the doctor obtained must be unexpired</li> <li>• The blood that the doctor obtained must have been tested</li> </ul>   |
| <b>Postcondition 1</b>    | Doctor obtains the selected blood  |
| <b>Postcondition 2</b>    | The retrieved blood's status is changed to "used"  |

### 3.1.4 Requirement 4

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | View expired blood  |
| <b>Priority</b>           | Must-have   |
| <b>Requirement 4</b>      | Storehouse manager can remove expired blood from the storehouse   |
| <b>Actor</b>              | Storehouse manager  |
| <b>Use Case Overview</b>  | Storehouse manager can view a list of expired blood   |
| <b>Trigger</b>            | The storehouse manager come to the storehouse   |
| <b>Precondition 1</b>     | Storehouse manager is registered  |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. The storehouse type in command to check the list of expired blood</li> <li>2. The list which includes the id of blood should be shown</li> </ol>   |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that no expired blood is founded:</p> <ol style="list-style-type: none"> <li>1. The storehouse type in command to check the list of expired blood</li> <li>2. The system returns "No expired blood" as a result</li> </ol> |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Only expired blood samples are on the list</li> </ul>  |
| <b>Postcondition</b>      | Storehouse has remembered the id of blood they want to remove if they want to remove certain blood  |

|                          |  |
|--------------------------|--|
| <b>Use Case 2</b>        | Remove certain disposed blood                                    |
| <b>Priority</b>          | Must-have  |
| <b>Requirement 4</b>     | Storehouse manager can remove disposed blood from the storehouse |
| <b>Actor</b>             | Storehouse   |
| <b>Use Case Overview</b> | The storehouse manager can remove certain disposed blood         |
| <b>Trigger</b>           | The storehouse manager start to remove disposed blood            |
| <b>Precondition 1</b>    | The storehouse manager knows the id of blood they want to remove |

|                           |   |
|---------------------------|---|
| <b>Precondition 2</b>     | The storehouse manager is registered  |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager retrieves certain expired blood from the storehouse</li> <li>2. Storehouse manager adds dispose command for certain blood</li> <li>3. There should be a notice indicates which blood has been disposed</li> <li>4. The selected blood and its related record is removed from the system</li> <li>5. Storehouse manager removes the selected blood from storehouse and make it ready for disposal</li> </ol> |
| <b>Alternative Flow 1</b> | <p>Failed scenario that describe the disposal of fresh blood:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager adds dispose command for certain blood</li> <li>2. There should be a notice "You can't dispose unexpired blood from system" to storhouse manager</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Only expired blood can be disposed</li> <li>• The record of expired blood is removed from the system</li> </ul>  |
| <b>Postcondition 1</b>    | The selected blood is deleted from the system and removed from storehouse if it's expired   |
| <b>Postcondition 2</b>    | The selected blood is unchanged in all fields if it's not expired   |

|                          |  |
|--------------------------|--|
| <b>Use Case 3</b>        | Remove all expired blood   |
| <b>Priority</b>          | Must-have  |
| <b>Requirement 4</b>     | Storehouse manager can remove the disposed blood from the storehouse   |
| <b>Actor</b>             | Storehouse   |
| <b>Use Case Overview</b> | The storehouse manager can remove all expired blood at once  |
| <b>Trigger</b>           | The storehouse manager wants to remove all expired blood   |
| <b>Precondition 1</b>    | The storehouse manager has retrieved all expired blood physically  |
| <b>Basic Flow</b>        | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager enters dispose command to remove all of the disposed blood from the system</li> <li>2. The system displays a list of all expired blood to</li> </ol> |

|                           |  |
|---------------------------|--|
|                           | <p>storehouse manger</p> <ol style="list-style-type: none"> <li>3. The system warns the storehouse manager “Are you sure?”</li> <li>4. Storehouse manager enters “yes”</li> <li>5. All expired blood and its related record is removed from the system</li> <li>6. There should be a notice that indicates when all expired blood has been disposed</li> <li>7. Storehouse manager removes all expired blood from storehouse and make it ready for disposal</li> </ol>   |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that storehouse manager enters “no”:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager enters dispose command to remove all of the disposed blood from the system</li> <li>2. System displays a list of all expired blood to storehouse manager</li> <li>3. System warns the storehouse manager “Are you sure?”</li> <li>4. Storehouse manager enters “no”</li> <li>5. There should be a notice that indicates no action is done, back to the main page</li> </ol> |
| <b>Alternative Flow 2</b> | <p>This scenario describes the situation that no expired blood is founded in the system:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager add dispose command to remove all the disposed blood</li> <li>2. There should be a notice indicates no expired blood is founded in the system</li> </ol>   |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>● Only expired blood can be disposed</li> <li>● The record of expired blood is removed from the system</li> </ul>   |
| <b>Postcondition</b>      | <p>All expired blood with the related record is deleted from the system and removed from storehouse if processed</p>   |

## 3.2 Should-have

### 3.2.1 Requirement 1

|                          |  |
|--------------------------|--|
| <b>Use Case 1</b>        | Tester indicates the start of testing  |
| <b>Priority</b>          | Should-have  |
| <b>Requirement 1</b>     | Testers are able to start testing and fill in the results later.   |
| <b>Actor</b>             | Tester   |
| <b>Use Case Overview</b> | When the tester decided to test a blood sample, the status of the blood sample will be changed to "Testing"  |
| <b>Trigger</b>           | Tester selected an untested blood  |
| <b>Precondition 1</b>    | The selected blood is entered  |
| <b>Precondition 2</b>    | Tester is registered in the system   |
| <b>Basic Flow</b>        | <p>Main success scenario:</p> <ol style="list-style-type: none"><li>1. Tester add test command for certain blood</li><li>2. The status of blood change to "Testing"</li><li>3. The blood will be removed from the untested list</li><li>4. The blood will be moved to a testing list</li><li>5. There should be a notice to indicate that the blood is "testing"</li></ol> |
| <b>Business Rules</b>    | <ul style="list-style-type: none"><li>• Every blood sample will only be tested once.</li><li>• Every blood sample will only in one list.</li></ul>   |
| <b>Postcondition 1</b>   | The selected blood sample is in "testing" state.   |
| <b>Postcondition 2</b>   | The blood is in the testing list, not in the untested list   |

|                          |   |
|--------------------------|---|
| <b>Use Case 2</b>        | Tester view and select a testing blood sample                           |
| <b>Priority</b>          | Should-have   |
| <b>Requirement 1</b>     | Testers are able to start testing and fill in the results later.        |
| <b>Actor</b>             | Tester  |
| <b>Use Case Overview</b> | Tester can view a list of testing blood and choose one to enter details |

|                        |   |
|------------------------|---|
| <b>Trigger</b>         | Tester add a blood sample into the testing list   |
| <b>Precondition 1</b>  | The tester is registered in the system  |
| <b>Precondition 2</b>  | There is a blood sample in the testing list   |
| <b>Basic Flow</b>      | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Tester add a command to check testing blood</li> <li>2. There should be a list of testing blood ordered by the date of retrieval</li> <li>3. The list also shows the id of blood</li> <li>4. Tester enters the id of the blood sample which has been tested.</li> </ol> |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• Every blood sample will only be tested once.</li> <li>• Every blood sample will only in one list.</li> <li>• Viewing a list will not move the elements in the list</li> </ul>  |
| <b>Postcondition 1</b> | The selected blood sample is entered.   |
| <b>Postcondition 2</b> | The blood is still in the testing list.   |

|                           |  |
|---------------------------|--|
| <b>Use Case 3</b>         | Tester indicates the end of testing  |
| <b>Priority</b>           | Should-have  |
| <b>Requirement 2</b>      | Testers are able to start testing and fill in the results later.   |
| <b>Actor</b>              | Tester   |
| <b>Use Case Overview</b>  | Tester finishes the testing, enter the type and expiry date of a blood   |
| <b>Trigger</b>            | Tester selects a testing blood sample in the testing list.   |
| <b>Precondition 1</b>     | The selected blood is entered  |
| <b>Precondition 2</b>     | Tester is registered in the system   |
| <b>Basic Flow</b>         | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Test succeeds, tester add type and expiry date of that blood</li> <li>2. The status of blood change to tested</li> <li>3. There should be a notice to indicate that the blood has been tested successfully</li> <li>4. The selected blood sample is removed from the testing list and available to be used.</li> </ol> |
| <b>Alternative Flow 1</b> | Scenario when the test result of the blood is negative:  |



|                       |  |
|-----------------------|--|
|                       | <ol style="list-style-type: none"> <li>1. The test fails, tester add a command to dispose</li> <li>2. The status of blood change to disposed</li> <li>3. There should be a notice to indicate that the blood has been disposed successfully</li> <li>4. The selected blood sample is removed from the testing list and waiting to be disposed</li> </ol> |
| <b>Business Rules</b> | <ul style="list-style-type: none"> <li>• Any blood that fails the test should be disposed.</li> <li>• Any blood that passes the test should have known expiration date and type.</li> <li>• Every blood sample will only in one list.</li> </ul>   |
| <b>Postcondition</b>  | Either the status of the blood is changed to “tested” with specified type and expiration date, or the blood has been disposed.   |

### 3.2.2 Requirement 2

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | Notify donor managers when blood is insufficient  |
| <b>Priority</b>           | Should-have   |
| <b>Requirement 3</b>      | When storage of blood is low, donor managers and doctors can be notified by the system.   |
| <b>Actor</b>              | Donor Manager   |
| <b>Use Case Overview</b>  | When storage of blood is low, donor managers can be notified by the system and react to it.   |
| <b>Trigger</b>            | A obtain, reservation or disposal of blood  |
| <b>Precondition 1</b>     | Certain type of blood is lower than a set amount  |
| <b>Precondition 2</b>     | There is a valid email address of Donor manager   |
| <b>Basic Flow</b>         | <p>Main success full scenario:</p> <ol style="list-style-type: none"> <li>1. After reserve or obtain of blood, system noticed that a certain type of blood is not enough</li> <li>2. The current storage amount of blood is recorded</li> <li>3. An email is generated with detailed information about shortage blood</li> <li>4. Email is sent to the donor manager</li> <li>5. Donor manager will read the email and react to this issue</li> </ol> |
| <b>Alternative Flow 1</b> | This scenario describes the situation that the system have sent notification for shortage of certain type of blood:   |

|                        |  |
|------------------------|--|
|                        | <ol style="list-style-type: none"> <li>1. After reserve or obtain of blood, system noticed that a certain type of blood is not enough</li> <li>2. System learns that the notification of issue has been sent before</li> <li>3. No email or message is generated</li> </ol>                    |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• The issue of shortage of certain blood will captured</li> <li>• Donor Manager will get notified for once for each issue</li> <li>• A blood shortage after level of blood get recovered to the set amount will be considered as a new issue</li> </ul> |
| <b>Postcondition 1</b> | Donor Manager will noticed the shortage of blood   |
| <b>Postcondition 2</b> | Donor Manager will only got one message for an event   |

|                           |   |
|---------------------------|---|
| <b>Use Case 2</b>         | Notify storehouse manager when blood is insufficient  |
| <b>Priority</b>           | Should-have   |
| <b>Requirement 3</b>      | When storage of blood is low, donor manager and storehouse manager can be notified by the system.   |
| <b>Actor</b>              | storehouse manager  |
| <b>Use Case Overview</b>  | When storage of blood is low, storehouse manager can be notified by the system.   |
| <b>Trigger</b>            | A obtain or reservation of blood  |
| <b>Precondition 1</b>     | Certain type of blood is lower than a set amount  |
| <b>Precondition 1</b>     | There is a valid email address of storehouse manager  |
| <b>Basic Flow</b>         | <ol style="list-style-type: none"> <li>1. After reserve or obtain of blood, system noticed that a certain type of blood is not enough</li> <li>2. The current storage amount of blood is recorded</li> <li>3. An email is generated with detailed information about shortage blood</li> <li>4. Email is sent to the company manager</li> <li>5. storehouse manager will read the email and react to this issue</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that the system have sent notification for shortage of certain type of blood:</p> <ol style="list-style-type: none"> <li>1. After reserve or obtain of blood, system noticed that a certain type of blood is not enough</li> <li>2. System learns that the notification of issue has been sent before</li> </ol>   |

|                        |   |
|------------------------|---|
|                        | 3. No email or message is generated   |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• The issue of shortage of certain blood will captured</li> <li>• storehouse manager will get notified for once for each issue</li> <li>• A blood shortage after level of blood get recovered to the set amount will be considered as a new issue</li> </ul> |
| <b>Postcondition 1</b> | storehouse manager will noticed the shortage of blood   |

|                          |  |
|--------------------------|--|
| <b>Use Case 3</b>        | Storehouse manager can view the current storage of all blood   |
| <b>Priority</b>          | Should-have  |
| <b>Requirement 3</b>     | When storage of blood is low, donor manager and storehouse manager can be notified by the system.  |
| <b>Actor</b>             | Storehouse manager   |
| <b>Use Case Overview</b> | Storehouse manager can use the system to view the total amount of all types of blood currently restored in system  |
| <b>Trigger</b>           | The storehouse manager wants to view the current amount of storage of blood  |
| <b>Precondition 1</b>    | Storehouse manager is registered in the system   |
| <b>Basic Flow</b>        | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager types in command to request current amount of storage of all blood</li> <li>2. The system returns a list contains tuples in format (Type of blood, total amount, total count of samples) based on unexpired blood</li> <li>3. Storehouse manager reads the response form the system</li> </ol> |
| <b>Business Rules</b>    | <ul style="list-style-type: none"> <li>• The total amount and total total count of samples must be non-negative</li> <li>• Only unexpired blood is counted</li> <li>• No record of blood is altered</li> </ul>   |
| <b>Postcondition 1</b>   | Storehouse manager knows the current storage level of each type of blood   |

|                      |   |
|----------------------|---|
| <b>Use Case 4</b>    | Storehouse manager can view the storage of all blood in future                                    |
| <b>Priority</b>      | Should-have   |
| <b>Requirement 3</b> | When storage of blood is low, donor manager and storehouse manager can be notified by the system. |

|                           |  |
|---------------------------|--|
| <b>Actor</b>              | Storehouse manager   |
| <b>Use Case Overview</b>  | Storehouse manager can use the system to view the total amount of all types of blood restored in system in future scale to make provision  |
| <b>Trigger</b>            | The storehouse manager wants to preview the amount of storage of blood in a given future date  |
| <b>Precondition 1</b>     | Storehouse manager is registered in the system   |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager types in command to request amount of storage of all blood at given future date</li> <li>2. The system returns a list contains tuples in format (Type of blood, total amount, total count of samples) based on unexpired blood at given future date</li> <li>3. Storehouse manager reads the response form the system</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that the given date is in the past:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager types in command to request amount of storage of all blood with date in wrong format</li> <li>2. The system generates an error message "Invalid date format"</li> </ol>  |
| <b>Alternative Flow 2</b> | <p>Failed scenario describes the situation that the date format is wrong:</p> <ol style="list-style-type: none"> <li>1. Storehouse manager types in command to request amount of storage of all blood at given date in the past</li> <li>2. The system generates a warning message "The given date is in the past, return current storage instead" to storehouse manager</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• The total amount and total total count of samples must be non-negative</li> <li>• Only unexpired blood at given date is counted</li> <li>• No record of blood is altered</li> </ul>   |
| <b>Postcondition 1</b>    | Storehouse manager knows the predicted future storage level of each type of blood if enquire successfully  |

## 3.3 Could-have

### 3.3.1 Requirement 1

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | View list of reservation  |
| <b>Priority</b>           | Could-have  |
| <b>Requirement 3</b>      | Doctors can reserve the blood and reservation can be cancelled by doctor.   |
| <b>Actor</b>              | Doctor  |
| <b>Use Case Overview</b>  | Doctor can view the list of his reservations  |
| <b>Trigger</b>            | Doctor wants to check his reservations  |
| <b>Precondition 1</b>     | Doctor knows his own id   |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"><li>1. Doctor type in the command which including his own id</li><li>2. The list of reservations made by this doctor is shown sort by date</li><li>3. The list contains the date of use, the id of blood and the id of reservation</li></ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that the doctor haven't made any reservation yet:</p> <ol style="list-style-type: none"><li>1. Doctor type in the command which including his own id</li><li>2. The system returns message "No reservation made"</li></ol>   |
| <b>Business Rules</b>     | <ul style="list-style-type: none"><li>• The reservations keep the same when viewing the list</li><li>• All reservation corresponding to this doctor is displayed</li><li>• No record is altered</li></ul>   |
| <b>Postcondition 1</b>    | Doctor know all his reservations of blood   |

|                          |   |
|--------------------------|---|
| <b>Use Case 2</b>        | Reserve blood   |
| <b>Priority</b>          | Could-have  |
| <b>Requirement 3</b>     | Doctors can reserve the blood and reservation can be cancelled by doctor. |
| <b>Actor</b>             | Doctor  |
| <b>Use Case Overview</b> | Doctor can reserve the blood to use                                       |
| <b>Trigger</b>           | Doctor need certain blood   |

|                           |  |
|---------------------------|--|
| <b>Precondition 1</b>     | The doctor is registered in the system   |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Doctor add command which contain the id(s) of blood they want to take out</li> <li>2. Doctor also enters the date of taking out be recorded for this blood</li> <li>3. The status of blood change to “used”</li> <li>4. There should be a notice indicates that the blood has been taken out successfully</li> <li>5. Blood storage level decreases by the amount of blood taken out</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes that invalid id is entered:</p> <ol style="list-style-type: none"> <li>1. Doctor add command which contain the id(s) of blood they want to take out</li> <li>2. System inform the doctor that the id(s) of the blood is invalid/reserved</li> </ol>   |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Blood has been tested and is safe for transfusion</li> <li>• Blood type stays the same</li> <li>• Expiry date stays the same</li> </ul>   |
| <b>Postcondition 1</b>    | Blood level has decreased by the amount of blood taken out and status of blood is now set to “used” if request accepted  |
| <b>Postcondition 2</b>    | Selected blood(s) is ready to be transported by third-party operator if request accepted   |
| <b>Postcondition 3</b>    | Doctor is aware the problem of request if failed   |

|                          |   |
|--------------------------|---|
| <b>Use Case 3</b>        | Cancel reservation  |
| <b>Priority</b>          | Could-have  |
| <b>Requirement 3</b>     | Doctors can reserve the blood and reservation can be canceled by doctor.          |
| <b>Actor</b>             | Doctor  |
| <b>Use Case Overview</b> | Doctor cancel his own reservation   |
| <b>Trigger</b>           | Doctor wants to cancel exist reservation made by himself                          |
| <b>Precondition 1</b>    | The date is before the date of use for reservation                                |
| <b>Precondition 2</b>    | Doctor know the id of reservation ( if not, check the list of reservation first ) |
| <b>Basic Flow</b>        | Main successful scenario:   |

|                        |  |
|------------------------|--|
|                        | <ol style="list-style-type: none"> <li>1. The doctor type in the cancel command which include the id of reservation</li> <li>2. This reservation is removed</li> <li>3. There should be a notice indicate result of command</li> </ol> |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• Other reservations keep the same</li> <li>• Record of other blood is remained the same</li> </ul>   |
| <b>Postcondition 1</b> | The reservation of this id is removed from system  |
| <b>Postcondition 2</b> | The status of blood is “tested” and available to be reserved   |

### 3.3.2 Requirement 2

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | View the general test status  |
| <b>Priority</b>           | Could-have  |
| <b>Requirement 2</b>      | Donation manager can be notified if there is blood demonstrating negative result in the test  |
| <b>Actor</b>              | Donor Manager   |
| <b>Use Case Overview</b>  | Donor manager can view a list of test records corresponding to the blood inserted in the system by this donor manager   |
| <b>Trigger</b>            | Donor manager wants to find out the test status of blood  |
| <b>Precondition 1</b>     | Donor manager is registered in the system and have unique Id  |
| <b>Precondition 2</b>     | Donor manager knows his role  |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters command with his ID</li> <li>2. The system returns a list of blood that is taken by this donor manger</li> <li>3. A tuple in the list contains the id of the blood, the donor information and the status of the blood (untested, passed, negative)</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that no blood is related to the manager:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters command with his ID</li> <li>2. The system notifies the donor manager that there is no blood to show</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Only blood related to this donor manager ID is displayed</li> <li>• Untested blood has status “untested”</li> <li>• Tested blood has status either “passed” or “negative”</li> </ul>   |

|                        |  |
|------------------------|--|
| <b>Postcondition 1</b> | Donor manager learns the status and ID of some blood |
|------------------------|--|

|                           |  |
|---------------------------|--|
| <b>Use Case 2</b>         | View the specific test status of a donor   |
| <b>Priority</b>           | Could-have   |
| <b>Requirement 2</b>      | Donation manager can be notified if there is blood demonstrating negative result in the test   |
| <b>Actor</b>              | Donor Manager  |
| <b>Use Case Overview</b>  | Donor manager can view test record with test notes corresponding to a specific donor   |
| <b>Trigger</b>            | Donor wants to find the test result of his blood   |
| <b>Precondition 1</b>     | Donor manager is registered in the system  |
| <b>Basic Flow</b>         | <p>Main successful scenario:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters view command with valid blood ID</li> <li>2. The system returns the record of the blood</li> <li>3. The record includes the status of the blood, donor's information and test note from tester of the blood if provided</li> <li>4. The Donor manager read the information of the record to serve the donor</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation that the ID of the blood is invalid:</p> <ol style="list-style-type: none"> <li>1. Donor manager enters a command with invalid blood ID</li> <li>2. The system notifies the donor manager that the blood ID is invalid</li> </ol>   |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• Record of the blood remains the same</li> <li>• Untested blood has status "untested"</li> <li>• Tested blood has status either "passed" or "negative"</li> </ul>  |
| <b>Postcondition 1</b>    | Donor manager learns the test result of the blood  |

|                      |  |
|----------------------|--|
| <b>Use Case 3</b>    | Notify donor manager for negative test result  |
| <b>Priority</b>      | could-have   |
| <b>Requirement 2</b> | Donation manager can be notified if there is blood demonstrating negative result in the test |



|                          |   |
|--------------------------|---|
| <b>Actor</b>             | Donor manager   |
| <b>Use Case Overview</b> | Donor manager will be notified immediately if blood inserted by him demonstrate a negative result   |
| <b>Trigger</b>           | Tester update the status of blood to “negative”   |
| <b>Precondition 1</b>    | The test result of blood is negative  |
| <b>Precondition 2</b>    | Donor manager involved has email address  |
| <b>Basic Flow</b>        | <p>Main successful flow:</p> <ol style="list-style-type: none"> <li>1. System notices the update of the blood and the test result is negative</li> <li>2. System generates an email with Donor manager ID, blood ID, test status and test notes</li> <li>3. Email is sent to the donor manager who inserts this blood to the system</li> <li>4. Donor manager in charge reads the email and reacts to this issue</li> </ol> |
| <b>Business Rules</b>    | <ul style="list-style-type: none"> <li>• All tested blood with a negative result are captured</li> <li>• Email is sent only once for an issue</li> </ul>  |
| <b>Postcondition 1</b>   | Email with necessary information is sent to Donor manager who inserts the blood with negative results to the system   |

### 3.3.3 Requirement 3

|                          |  |
|--------------------------|--|
| <b>Use Case 1</b>        | Input as volume  |
| <b>Priority</b>          | could-have   |
| <b>Requirement 3</b>     | The amounts of blood should be measured in volume.   |
| <b>Actor</b>             | Donor Manager  |
| <b>Use Case Overview</b> | (This use case is supplementary of must-have - requirement 1 - use case 1.) Donor manager should be able to specify the volume of blood sample when adding untested blood record into the system   |
| <b>Trigger</b>           | New blood arrives at the storehouse  |
| <b>Precondition 1</b>    | The blood has been transported to the storehouse   |
| <b>Basic Flow</b>        | <p>Main success scenario:</p> <ol style="list-style-type: none"> <li>1. Donation manager enters the date of retrieval, donor information of blood and Donation manager's id</li> <li>2. Donation manager will be required to enter the volume</li> </ol> |

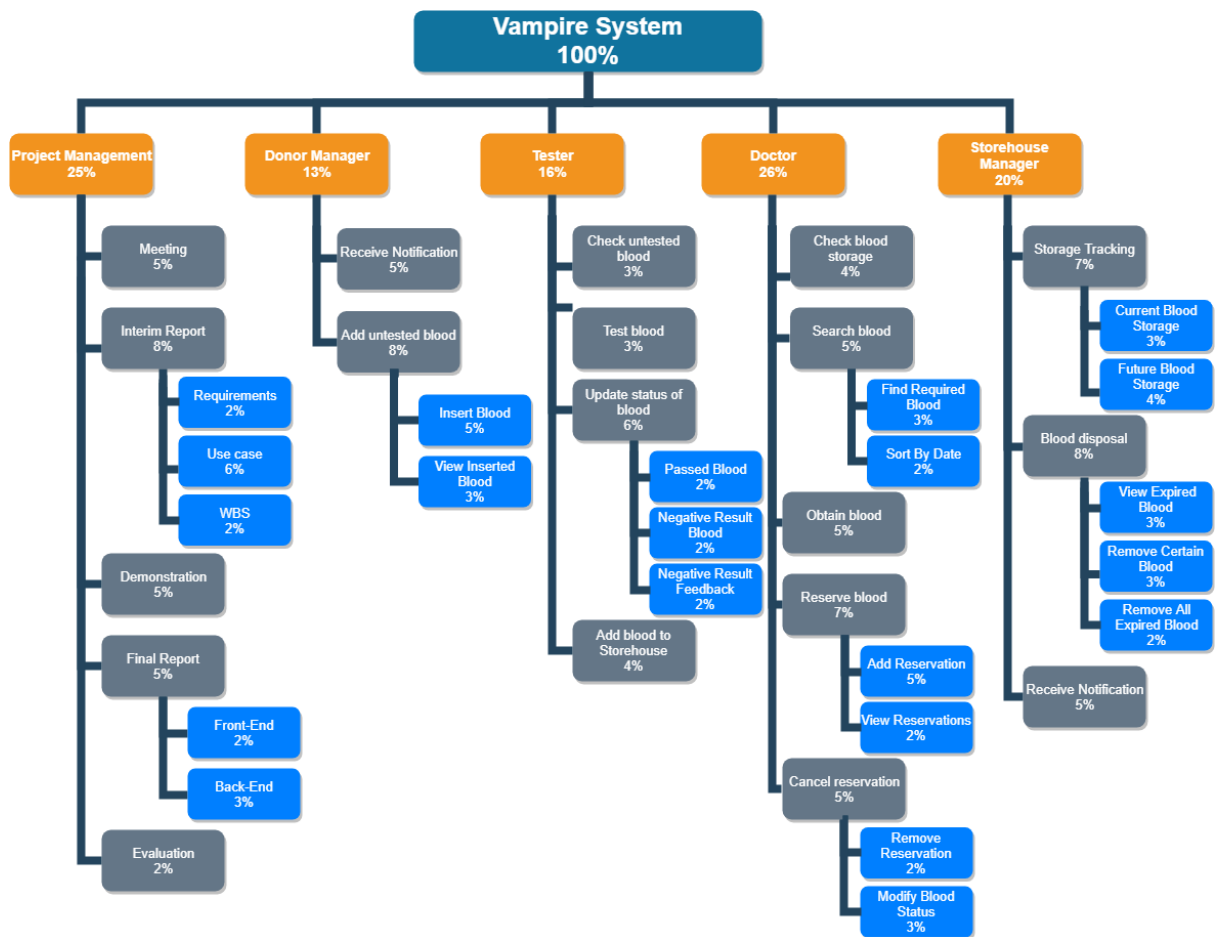
|                        |   |
|------------------------|---|
|                        | <p>of the blood sample.</p> <ol style="list-style-type: none"> <li>3. Donation manager enters the volume.</li> <li>4. The system assigns a unique id and untested state.</li> </ol> |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• Every sample of blood in the storehouse has a unique id.</li> <li>• Every newly added blood need to be tested.</li> </ul>                  |
| <b>Postcondition 1</b> | Successfully stored blood has a unique id and untested state.   |

|                           |  |
|---------------------------|--|
| <b>Use Case 2</b>         | View list of blood   |
| <b>Priority</b>           | could-have   |
| <b>Requirement 3</b>      | The amounts of blood should be measured in volume.   |
| <b>Actor</b>              | Doctor   |
| <b>Use Case Overview</b>  | (This use case is supplementary of must-have - requirement 3 - use case 1.) Doctor can view a list of blood of certain type with amount  |
| <b>Trigger</b>            | Doctor wants to check the storage of certain type of blood   |
| <b>Precondition 1</b>     | The doctor is registered in the system   |
| <b>Basic Flow</b>         | <p>Main success scenario, there is enough fresh blood in the system:</p> <ol style="list-style-type: none"> <li>1. Doctor add the command which contains the type of blood to ask for a certain type and a certain amount of blood</li> <li>2. The list of tested blood of that type shows in the system ordered by date of expiry</li> <li>3. The id of blood also shows in the list</li> </ol> |
| <b>Alternative Flow 1</b> | <p>This scenario describes the situation when there is no sample satisfying the search conditions in the system:</p> <ol style="list-style-type: none"> <li>1. Doctor add the command which contains the type of blood to ask for a certain type and a certain amount of blood</li> <li>2. "No record" is displayed to inform the Doctor</li> </ol>  |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• The blood shows in the list must have been tested</li> <li>• The blood shows in the list must be unexpired</li> <li>• The blood shows in the list must be available to use</li> </ul>   |
| <b>Postcondition 1</b>    | Doctor know the amount of storage of certain type of blood   |
| <b>Postcondition 2</b>    | Doctor remembered the id of blood they want if there is blood  |

## 3.4 Won't-have

For reading convenience, use cases for “won’t have” requirements have been moved to appendices in Section 8.1.

## 4. Work Breakdown Structure



## 5. Front-End Implementation

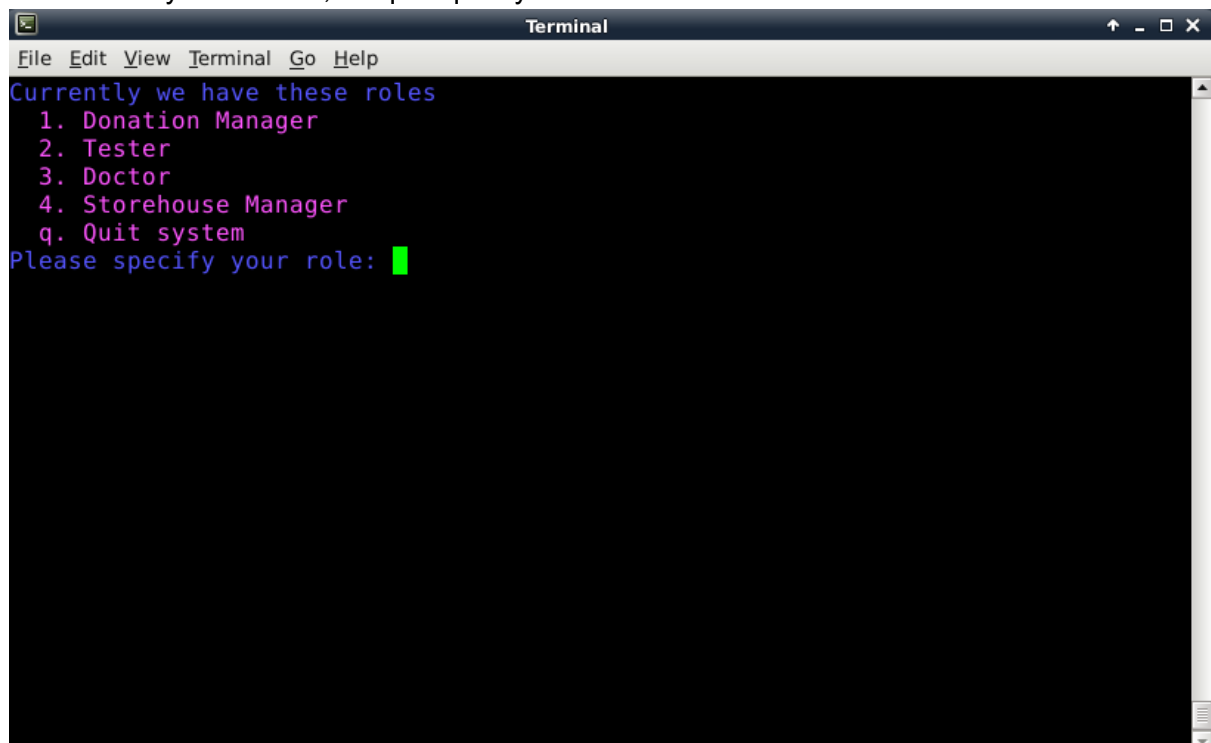
### 5.1 Design and Implementation

Since the project has been implemented as a command-line program, we tried to improve the user experience by simply displaying the text colourful. Also, the structure of the interaction has been levelised by abstracting the level into an abstract class, then statically or dynamically binding sub-classes, which is very similar to the structure of composite pattern in object-oriented programming. The basic usage for a user is to select a menu item accordingly by entering the number in front of it to the terminal.

### 5.2 Front-end Walkthroughs:

#### 5.2.1 Top Level Menu

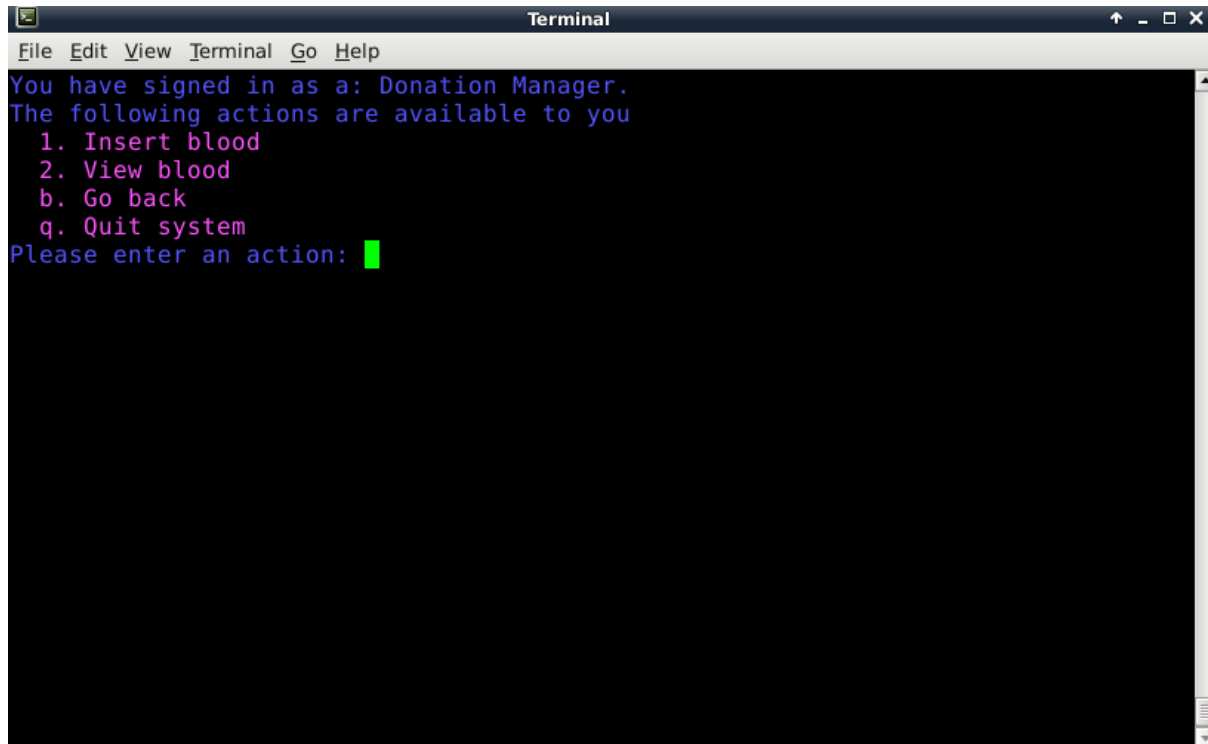
The Vampire System is designed to satisfy all the needs of different users. Therefore, functions are separated due to the user type. Therefore, there is a top menu level that users can choose which type of the role they are going to be. Users could type in the number of the role that they want to be, or 'q' to quit system.

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal content shows a list of roles: "Currently we have these roles", followed by a numbered list: "1. Donation Manager", "2. Tester", "3. Doctor", "4. Storehouse Manager", and "q. Quit system". Below the list, it says "Please specify your role:" followed by a green cursor. The terminal background is black, and the text is in various colors (blue, green, red, yellow).

#### 5.2.2 Donation Manager

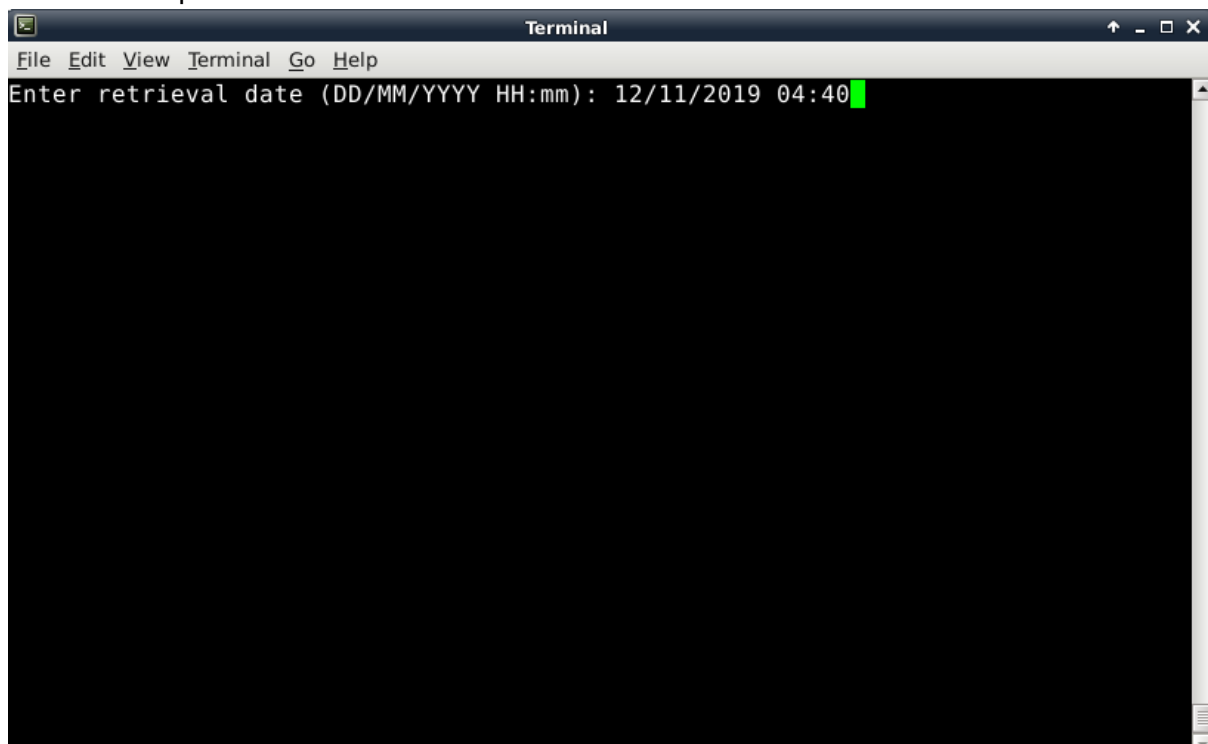
Input 1 into the command line to sign in as a donation manager. The submenu will show the actions that donation manager could do. Whether insert new blood or view the untested

blood list.



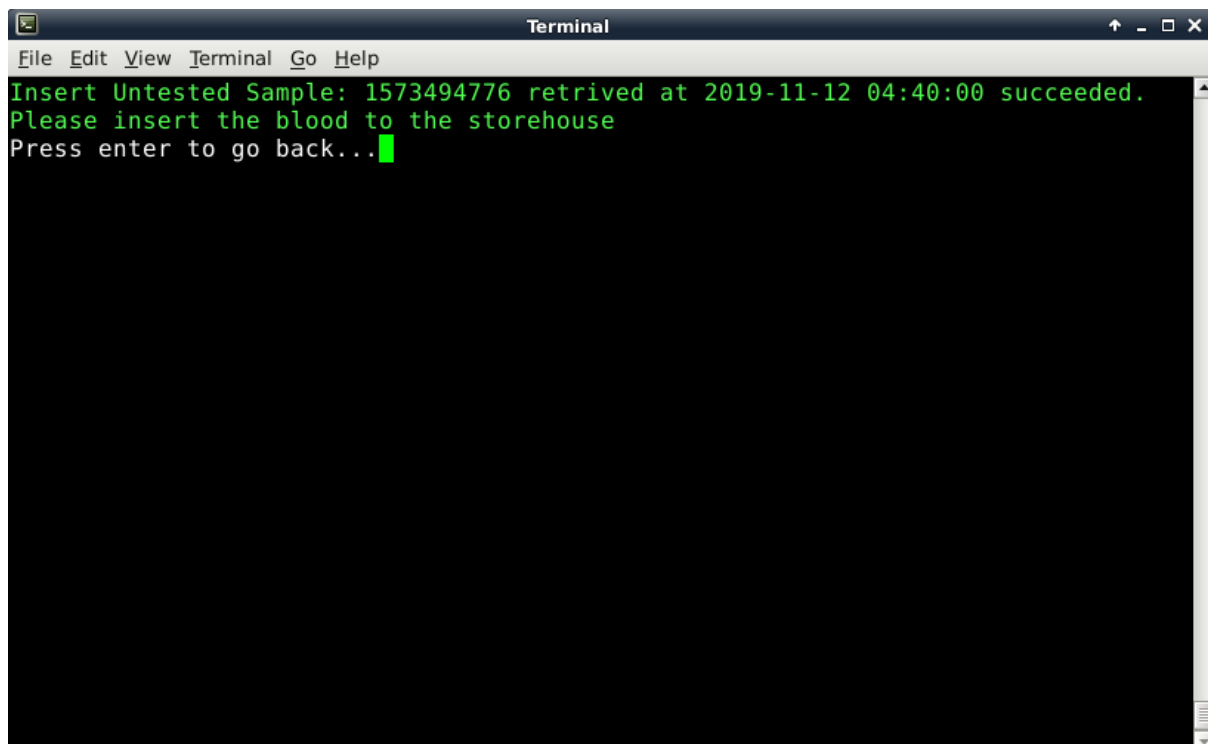
```
Terminal
File Edit View Terminal Go Help
You have signed in as a: Donation Manager.
The following actions are available to you
  1. Insert blood
  2. View blood
  b. Go back
  q. Quit system
Please enter an action: █
```

For inserting new blood into the system, the user is asked to provide a retrieval date for the blood. The input value should be in correct format.



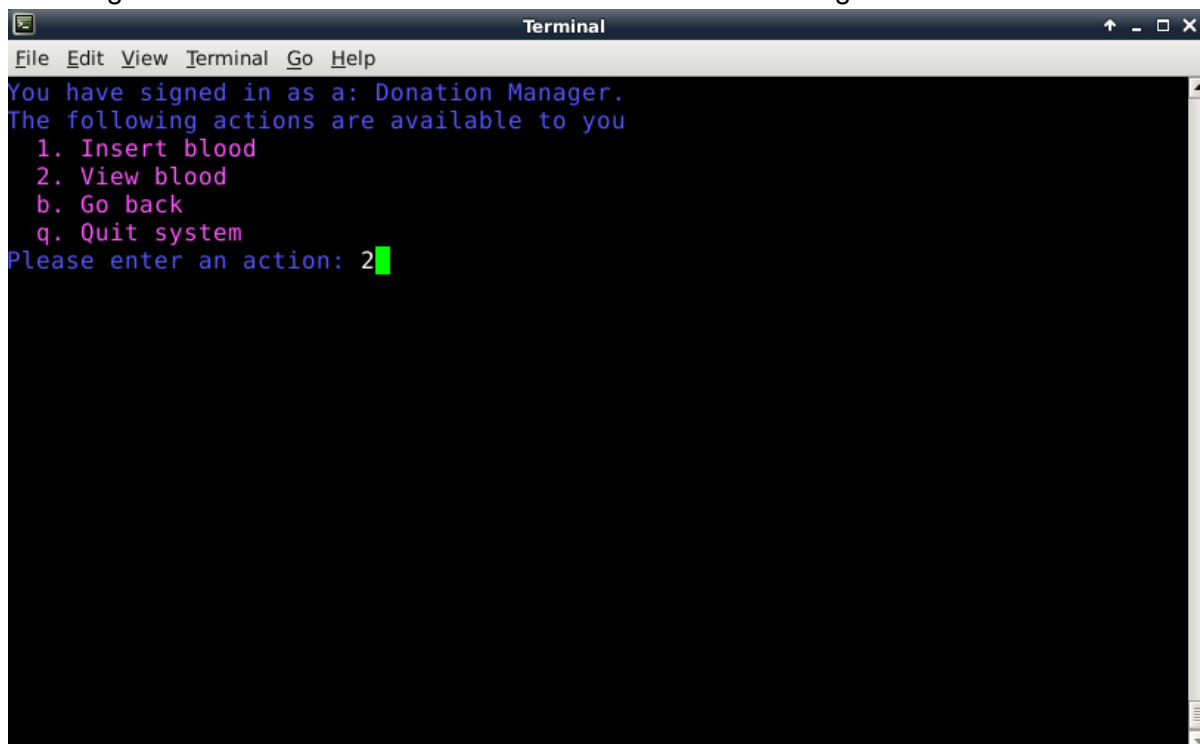
```
Terminal
File Edit View Terminal Go Help
Enter retrieval date (DD/MM/YYYY HH:mm): 12/11/2019 04:40 █
```

Then, if a correct date time input for the blood, a random id is generated for the blood. And a success information would be shown.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The output shows a green message: "Insert Untested Sample: 1573494776 retrived at 2019-11-12 04:40:00 succeeded. Please insert the blood to the storehouse Press enter to go back..." followed by a green cursor.

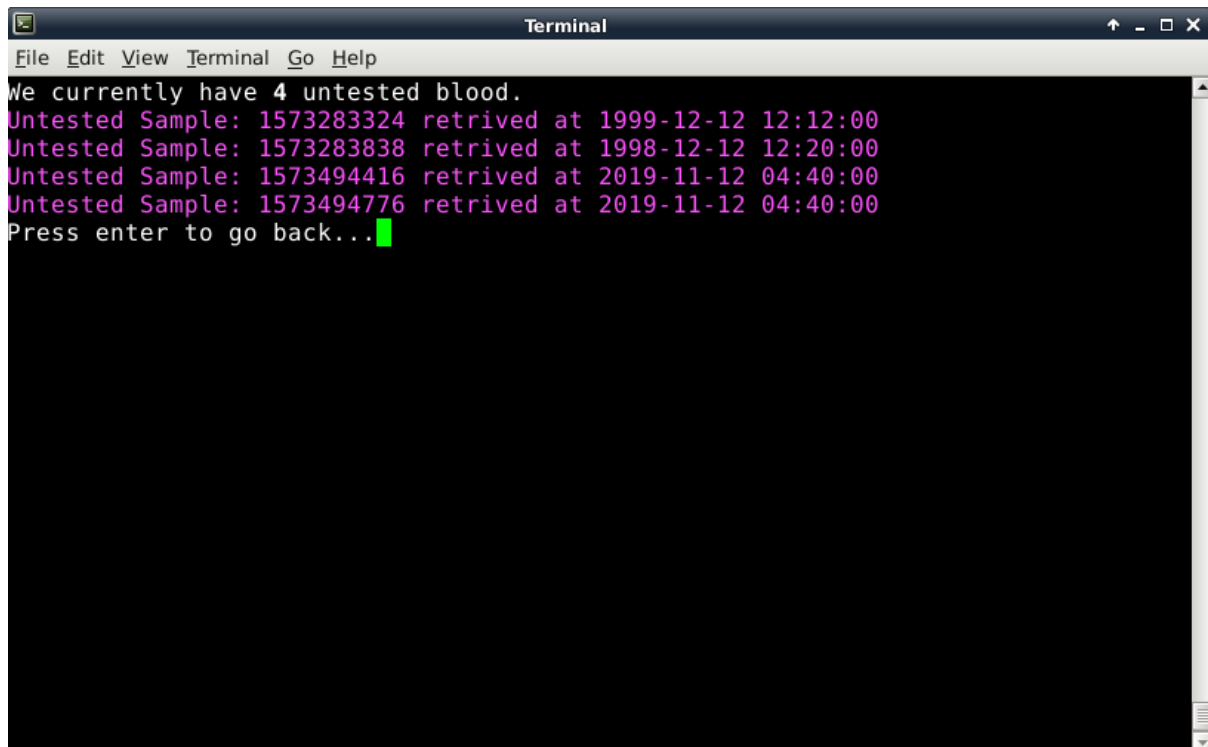
```
Terminal
File Edit View Terminal Go Help
Insert Untested Sample: 1573494776 retrived at 2019-11-12 04:40:00 succeeded.
Please insert the blood to the storehouse
Press enter to go back...
```

Pressing enter would redirect to the submenu of donation manager.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The output shows a blue message: "You have signed in as a: Donation Manager. The following actions are available to you" followed by a list of actions: "1. Insert blood", "2. View blood", "b. Go back", "q. Quit system". Below the list is the prompt "Please enter an action: 2" followed by a green cursor.

```
Terminal
File Edit View Terminal Go Help
You have signed in as a: Donation Manager.
The following actions are available to you
  1. Insert blood
  2. View blood
  b. Go back
  q. Quit system
Please enter an action: 2
```

For viewing the blood, all the untested blood stored in the database would be shown including its id and retrieval date.

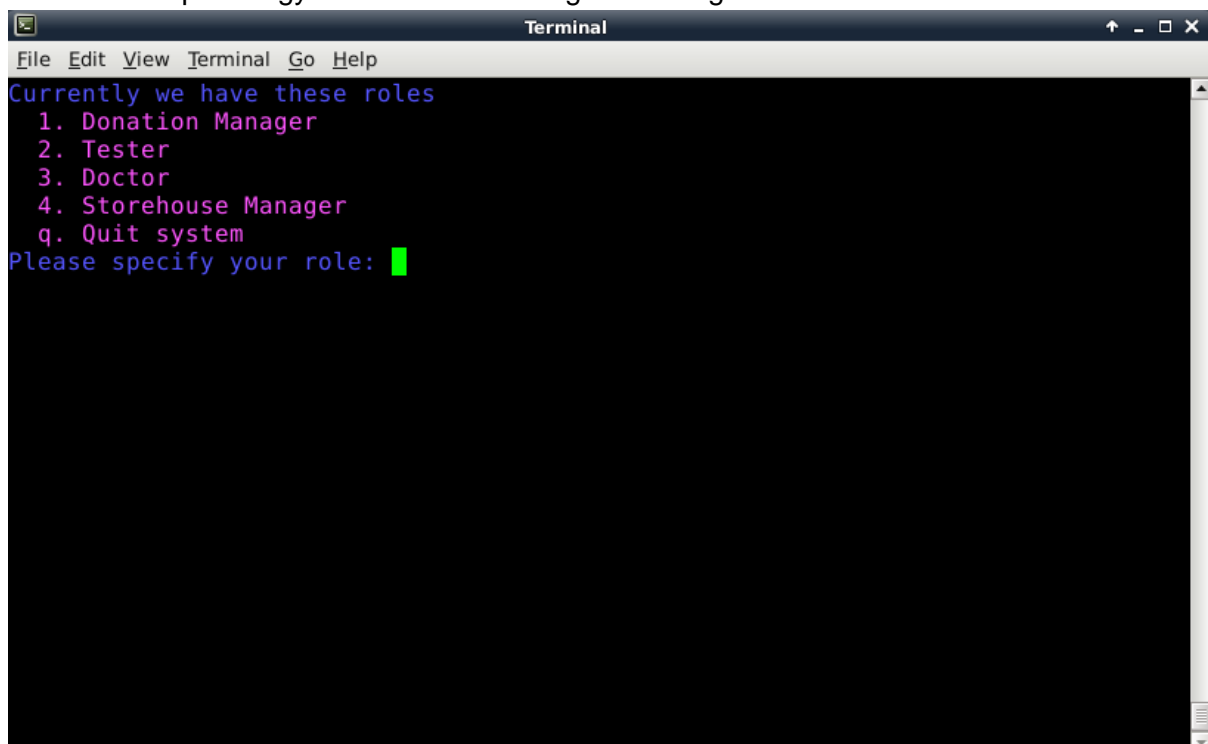
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The text inside the terminal is as follows:

```
We currently have 4 untested blood.  
Untested Sample: 1573283324 retrived at 1999-12-12 12:12:00  
Untested Sample: 1573283838 retrived at 1998-12-12 12:20:00  
Untested Sample: 1573494416 retrived at 2019-11-12 04:40:00  
Untested Sample: 1573494776 retrived at 2019-11-12 04:40:00  
Press enter to go back...
```

A green cursor is positioned at the end of the last line.

### 5.2.3 Tester

Inputting '2' into the input line will sign the user in as a Tester. The Tester is a registered member of a pathology clinic and is in charge of testing blood.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The text inside the terminal is as follows:

```
Currently we have these roles  
1. Donation Manager  
2. Tester  
3. Doctor  
4. Storehouse Manager  
q. Quit system  
Please specify your role:
```

A green cursor is positioned at the end of the last line.



The tester can do the following actions:

```
⬆️You are a tester now.  
1. Test blood  
2. Fill in result  
b. Go back  
q. Quit system  
What do you want to do?
```

A tester must first choose which blood they want to start testing. To do this, a tester would input '1' onto the command line and be presented with the following screen:

```
What do you want to do? 1  
⬆️We have these untested blood in the storehouse:  
1. Untested sample 1573974251  
b. Go back  
q. Quit system  
Select one of the blood you want to test:
```

Once the tester selects an untested blood sample they will be navigated back to the previous menu-level. Once the tester has finished testing the blood (in real life) they can fill in the results. To do this they input '2' into the command line.

```
⬆️You are a tester now.  
1. Test blood  
2. Fill in result  
b. Go back  
q. Quit system  
What do you want to do?
```

They will be presented with a list of pending blood whose details need to be filled in.

```
⬆️We have these testing blood in the pending:  
1. Pending sample 1573974251 | Testing Start Date 17-11-2019 18:05  
b. Go back  
q. Quit system  
Select one of the blood you want to fill in test result:
```

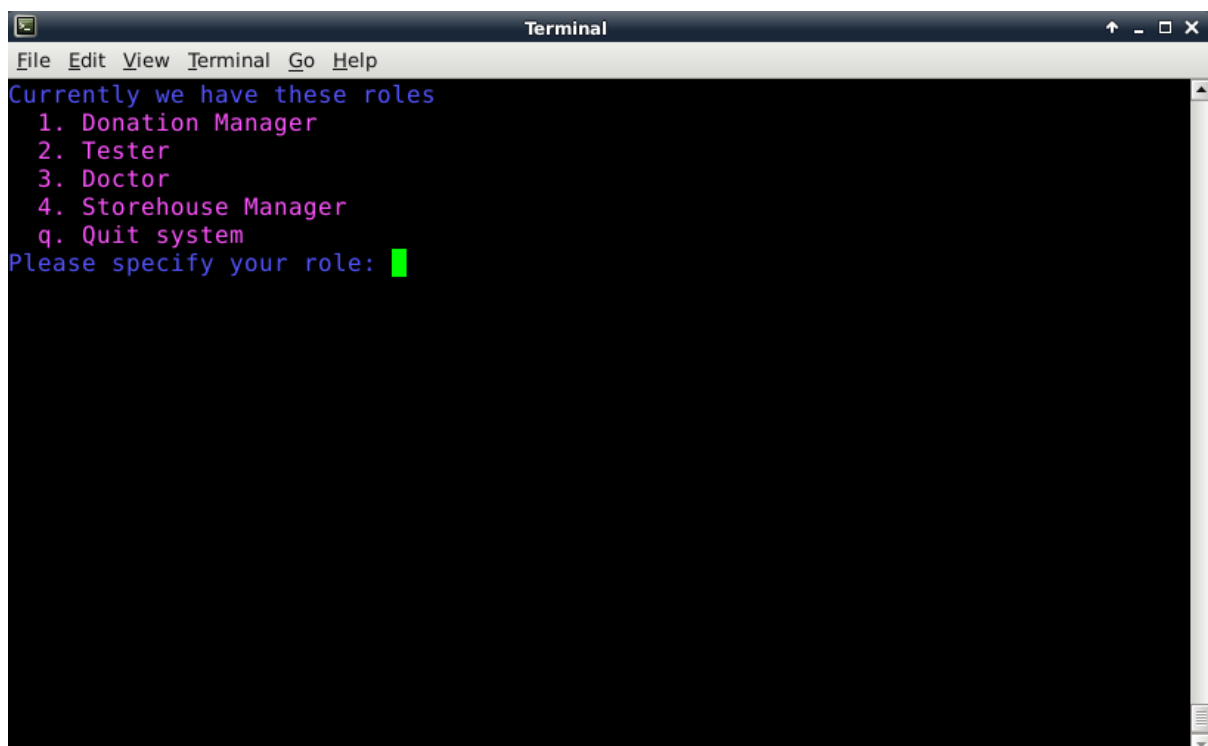
Once the Tester selects a blood sample, they will be asked if the blood has passed the test. If it has, then the Tester will need to enter the blood type and expiration date. If the blood fails, then it will be removed from the list and added to the disposed blood list.

```
↑Is the test passed (Y/N): Y
↑Valid Blood Types are: O, O+, O-, A+, A-, B+, B-, AB+, AB-
Enter blood type: O+
↑Enter expiration date (DD/MM/YYYY HH:mm): 31/12/2021 12:30
↑Blood sample 1573974251 has been tested.
Press enter to continue
```

The tester will then be navigated back to their home menu and can proceed to test other blood samples.

## 5.2.4 Doctor

Inputting '3' into the command line will sign in the user as a Doctor. The Doctor can view blood samples, search for blood samples by type, reserve blood and cancel blood reservations.

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal content shows a list of roles: "Currently we have these roles", followed by a numbered list: "1. Donation Manager", "2. Tester", "3. Doctor", "4. Storehouse Manager", and "q. Quit system". Below the list, it says "Please specify your role:" followed by a green cursor.

```
Terminal
File Edit View Terminal Go Help
Currently we have these roles
1. Donation Manager
2. Tester
3. Doctor
4. Storehouse Manager
q. Quit system
Please specify your role: █
```

The doctor will then be presented with the following screen:

```
⬆️You have signed in as a: Doctor
The following actions are available to you
  1. View Blood
  2. Reserve Blood
  3. Search By Blood Type
  4. View Reserve Blood List
  5. Cancel Blood Reservation
  b. Go back
  q. Quit system
Please enter an action:
```

#### 5.2.4.1 Doctor Reservation

```
⬆️You have signed in as a: Doctor
The following actions are available to you
  1. View Blood
  2. Reserve Blood
  3. Search By Blood Type
  4. View Reserve Blood List
  5. Cancel Blood Reservation
  b. Go back
  q. Quit system
Please enter an action:
```

If the user inputs '2' into the command-line they will be presented with a list of blood samples that they can reserve. Reserving a blood sample will remove that sample from the main blood list and add it to the reserved blood list

```
We have these tested blood in the storehouse:
  1. Blood ID: 1573280814 | Expiration Date: 12-12-2019 12:30
  2. Blood ID: 1572769082 | Expiration Date: 12-12-2020 12:30
  3. Blood ID: 1573974251 | Expiration Date: 31-12-2021 12:30
  4. Blood ID: 1573280644 | Expiration Date: 12-10-2100 12:30
  b. Go back
  q. Quit system
Select one of the blood you want to reserve:
```

```
Blood has been reserved.
```

The doctor is also able to view a list of all blood reservations:

```
We currently have 1 blood reserved.
ID:1573280814 | Blood Type:0 | Retrieval Date: 12-12-2018 12:30 | Expiration Date: 12-12-2019 12:30
Press enter to go back...
```

#### 5.2.4.2 Doctor Cancel Reservation

```
⬆️You have signed in as a: Doctor
The following actions are available to you
  1. View Blood
  2. Reserve Blood
  3. Search By Blood Type
  4. View Reserve Blood List
  5. Cancel Blood Reservation
  b. Go back
  q. Quit system
Please enter an action:
```

If the user inputs '5' into the command-line they will be presented with a list of blood reservations that they can cancel

```
We have these blood reserved in the storehouse
  1. Blood ID: 1573280814 | Expiration Date: 12-12-2019 12:30
  b. Go back
  q. Quit system
Select the blood reservation you want to cancel:
Select the blood reservation you want to cancel: 1
⬆️Blood reservation has been removed.
```

Once the reservation has been cancelled, the blood sample will be moved back into the main blood list.

#### 5.2.4.3 Doctor Search For Blood by Type

```
⬆️You have signed in as a: Doctor
The following actions are available to you
  1. View Blood
  2. Reserve Blood
  3. Search By Blood Type
  4. View Reserve Blood List
  5. Cancel Blood Reservation
  b. Go back
  q. Quit system
Please enter an action:
```

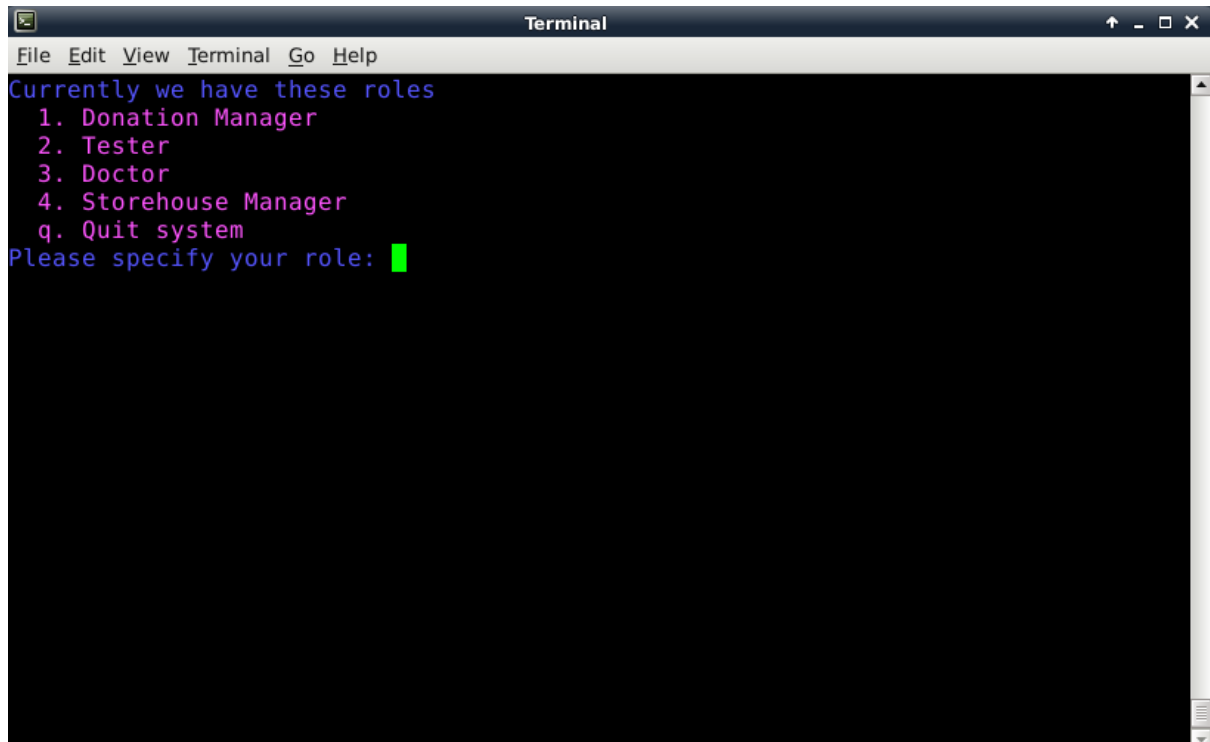
If the User inputs '3' into the command-line they will be presented with a list of valid blood-types to search for.

```
Valid Blood Types are: O, O+, O-, A+, A-, B+, B-, AB+, AB-
Enter blood type:
```

Once the user enters in their blood type. A list will appear showing all blood samples with that specific blood type in sorted order (based on expiration date)

```
⬆ID:1573280814 | Blood Type:0 | Retrieval Date: 12-12-2018 12:30 | Expiration Date: 12-12-2019 12:30
ID:1573280644 | Blood Type:0 | Retrieval Date: 12-12-1997 12:30 | Expiration Date: 12-10-2100 12:30
Press enter to go back...
```

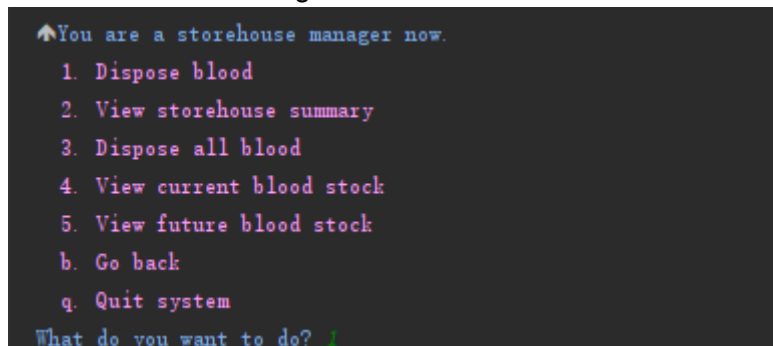
## 5.2.5 Storehouse Manager

A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal displays a list of roles: "Currently we have these roles", followed by a numbered list: "1. Donation Manager", "2. Tester", "3. Doctor", "4. Storehouse Manager", and "q. Quit system". Below the list, it says "Please specify your role:" followed by a green cursor.

```
File Edit View Terminal Go Help
Currently we have these roles
 1. Donation Manager
 2. Tester
 3. Doctor
 4. Storehouse Manager
 q. Quit system
Please specify your role: █
```

If the user inputs '4' into the command line they will sign in as a Storehouse Manager. The storehouse manager is in charge of disposing expired blood and managing the stock levels of blood

The storehouse manager will now be able to see a list of commands:

A terminal window showing a list of commands for the Storehouse Manager. It starts with "⬆You are a storehouse manager now." followed by a numbered list: "1. Dispose blood", "2. View storehouse summary", "3. Dispose all blood", "4. View current blood stock", "5. View future blood stock", "b. Go back", and "q. Quit system". At the bottom, it says "What do you want to do?" followed by a green cursor.

```
⬆You are a storehouse manager now.
 1. Dispose blood
 2. View storehouse summary
 3. Dispose all blood
 4. View current blood stock
 5. View future blood stock
 b. Go back
 q. Quit system
What do you want to do? █
```

### 5.2.5.1 Storehouse Manager Disposes Blood Sample

If the storehouse manager enters '1' into the command line he will be able to select a blood sample to dispose from a list of expired / failed blood

```
We have these expired blood in the storehouse:
```

1. Disposed sample 1572765435
2. Disposed sample 1572766273
3. Disposed sample 1572767453
4. Disposed sample 1573279121
5. Disposed sample 1573280769
- b. Go back
- q. Quit system

```
Select one of the blood you want to dispose:
```

```
Select one of the blood you want to dispose: 1
```

```
↑Blood sample 1572765435 succeeded.
```

```
Please remove the blood from the storehouse
```

```
Press enter to go back...
```

Furthermore, for ease of use the storehouse manager can choose to dispose all blood in the disposed list.

```
We have 4 expired blood.
```

```
ID:1572766273
```

```
ID:1572767453
```

```
ID:1573279121
```

```
ID:1573280769
```

```
Please collect all above blood from storehouse before delete them from system
```

```
Are you sure you want to delete them all (Y/N):
```

#### 5.2.5.2 Storehouse Manager Check Storage

If the storehouse manager enters '2' into the command line he will be able to see an overview of the storage levels of all types of blood. If there are less than 3 blood samples for a blood type, a notification will appear to get more stock. An email will also be sent to the storehouse manager.

```
Blood Type: O has 2 Blood stock remaining. Please get more stock.
```

```
Blood Type: O+ has 1 Blood stock remaining. Please get more stock.
```

```
Blood Type: O- has 0 Blood stock remaining. Please get more stock.
```

```
Blood Type: A+ has 0 Blood stock remaining. Please get more stock.
```

```
Blood Type: A- has 0 Blood stock remaining. Please get more stock.
```

```
Blood Type: B+ has 0 Blood stock remaining. Please get more stock.
```

```
Blood Type: B- has 0 Blood stock remaining. Please get more stock.
```

```
Blood Type: AB+ has 1 Blood stock remaining. Please get more stock.
```

```
Blood Type: AB- has 0 Blood stock remaining. Please get more stock.
```

```
We currently have 4 available blood supplies.
```

```
ID:1573280814 | Blood Type:O | Retrieval Date: 12-12-2018 12:30 | Expiration Date: 12-12-2019 12:30
```

```
ID:1572769082 | Blood Type:AB+ | Retrieval Date: 12-12-2019 12:30 | Expiration Date: 12-12-2020 12:30
```

```
ID:1573974251 | Blood Type:O+ | Retrieval Date: 12-12-1997 12:30 | Expiration Date: 31-12-2021 12:30
```

```
ID:1573280644 | Blood Type:O | Retrieval Date: 12-12-1997 12:30 | Expiration Date: 12-10-2100 12:30
```

### 5.2.5.3 Storehouse Manager Check Future Stock

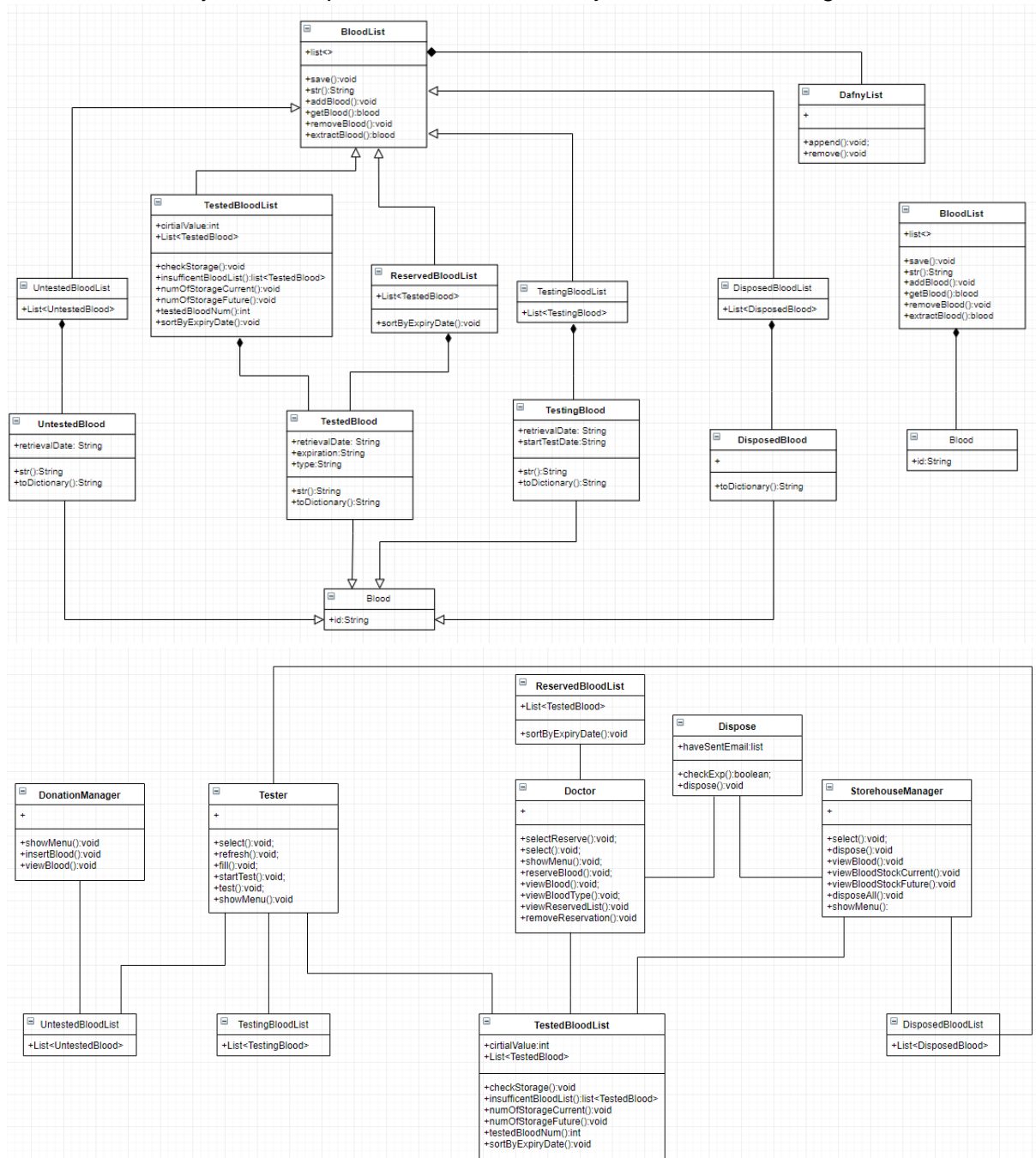
This function allows the storehouse manager to enter a date in the future. It will filter out all the blood that has expired by that date so that the storehouse manager can accurately gauge which blood will need to be replaced in the future.

```
Enter date in future (DD/MM/YYYY HH:mm): 12/12/2019 12:30
↑The estimated stock level at 12/12/2019 12:30 is:
  Blood Type: O has 2 Blood stock remaining
  Blood Type: O+ has 1 Blood stock remaining
  Blood Type: O- has 0 Blood stock remaining
  Blood Type: A+ has 0 Blood stock remaining
  Blood Type: A- has 0 Blood stock remaining
  Blood Type: B+ has 0 Blood stock remaining
  Blood Type: B- has 0 Blood stock remaining
  Blood Type: AB+ has 1 Blood stock remaining
  Blood Type: AB- has 0 Blood stock remaining
Press enter to go back...
```

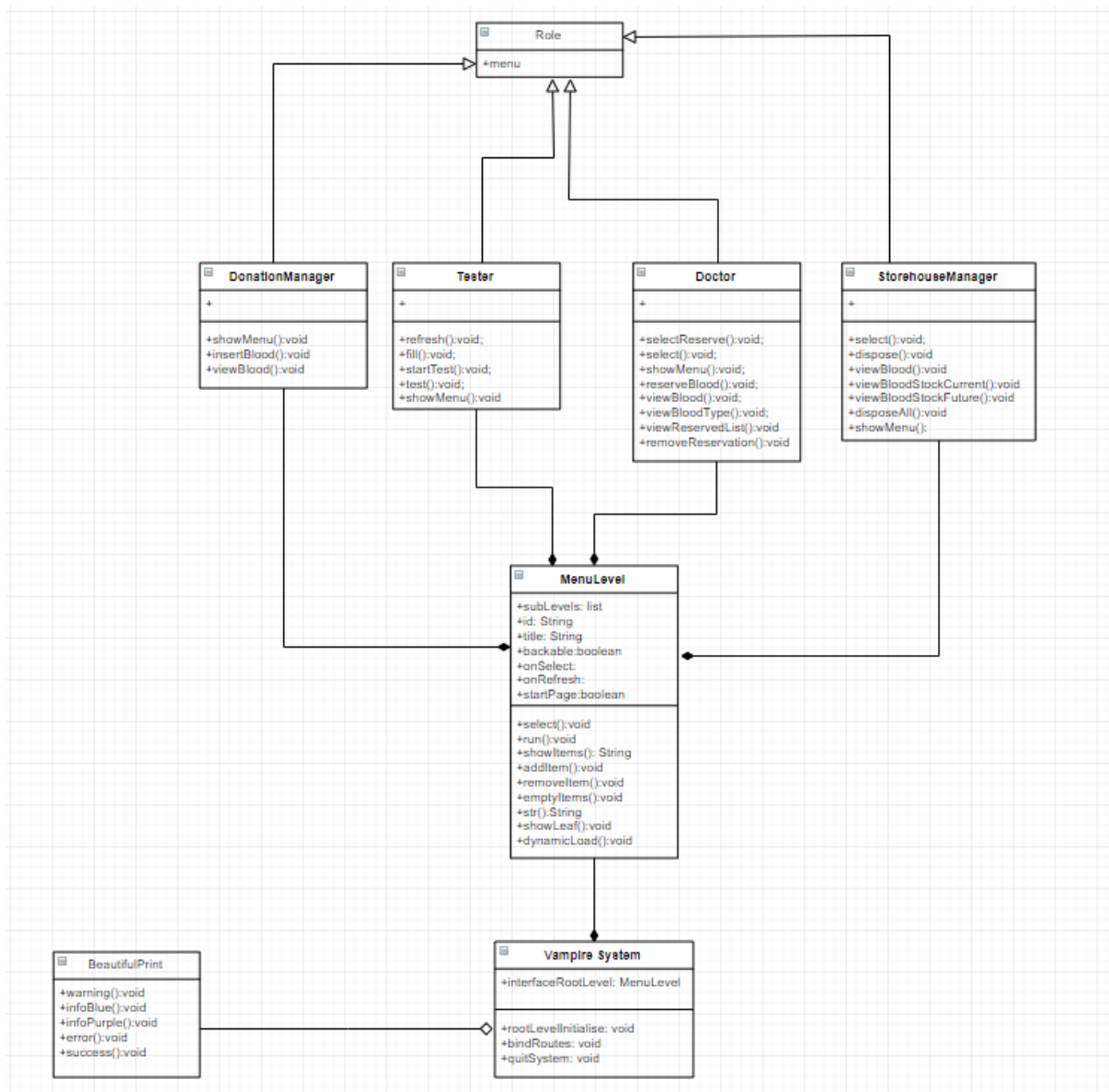
## 6. Back-End Implementation

### 6.1 Implementation

The backend of system is implemented based on Object Orientation Design.







## 6.2 Module Outline

### 6.2.1 Blood

The Blood class is an abstract class, which has four subclasses: UntestedBlood, TestingBlood, TestedBlood and DisposedBlood. The abstract blood class only contains the id of blood, which is generated by timestamp on creation to make sure it is unique. UntestedBlood adds the retrieval date of blood. TestingBlood adds starting date of testing. TestedBlood adds the expiration date and type of the blood. And DisposedBlood only contains the id of blood. Each object represents a blood sample in the real-world, and the state of the blood is corresponding to the type of class here.

## 6.2.2 BloodList

The Bloodlist class is an abstract class, which has five subclasses: UntestedBloodList, TestngBloodList, TestedBloodList, ReservedBloodList and DisposedList. Each list stores corresponding type of blood, the ReservedBloodList and TestedBloodList both stores TestedBlood. Similar to Blood in Section 6.2.1, the list simulates the collection of blood sample.

The data of list will be stored as different json files in dataset folder for persistence purposes.

When a state of blood changes, generally it will be removed from the current list and a new object will be instantiated based on the attributes of the original object and user input. After the instantiation, the new object will be added to the corresponding list.

All the list in this part is actually object of DafnyList, which use dafny way to implement append() and remove().

## 6.2.3 Roles and Actions

The operations of different users are implemented in different role classes, which are DonationManager, Tester, StorehouseManager and Doctor. Each role class contains the functions to get user input and also change the status of bloods. These functions will be described below.

In addition, there is another class Dispose containing the functions to check each blood in the list and compare the expiration date with current date. After dispose is called, all the expiration blood in the list should be disposed. This function is called whenever the doctor want to view or reserve blood, so that the expiration blood will not be taken by the doctor.

### 6.2.3.1 DonationManager

- `insertBlood()`: Retrieve information of untested blood sample, then add the blood to UntestedBloodList.
- `viewBlood()`: Visualise the list of all untested blood

### 6.2.3.2 Tester

- `startTest()`: Convert an UntestedBlood to TestingBlood indicating the start of a test.
- `fill()`: Convert a TestingBlood to TestedBlood along with some extra information if the test passed, or DisposedBlood if the test failed.

### 6.2.3.3 Doctor

- `viewBlood()`: Visualise the list of all tested blood
- `reserveBlood()`: Move a blood to ReservedBloodList, indicating it has been reserved.
- `removeReservation()`: Revert the operation made by reserveBlood(). Move a blood from ReservedBloodList back to TestedBloodList.
- `viewReservedList()`: Visualise the list of all reserved blood

- `viewBloodType()`: Visualise the list of all tested blood on specific type, i.e. search by type.

#### 6.2.3.4 StorehouseManager

- `dispose()`: Remove a blood from `DisposedBloodList`, indicating the sample has been properly destroyed.
- `disposeAll()`: Empty the `DisposedBloodList`, destroying all invalid blood sample.
- `viewBlood()`: Visualise the list of all tested blood, with out-of-stock warning where appropriate.
- `viewBloodStockFuture()`: Visualise the list of all tested blood at some future time, with out-of-stock warning where appropriate. Notice that the data here are just simply calculated by the expiration date of current blood sample in the storehouse, indicating the stock level when some blood expired in the future to provide a possibility to a Storehouse Manager act before blood expiration occurs.
- `viewBloodStockCurrent()`: Same as above, but the time is set to the moment that the function is called.

## 6.3 Verification

### 6.3.1 Coverage

The main usage of Dafny is to verify list operations, including `addBlood()`, `extractBlood()` and `sortBloodByDate()`. Also, we attempted to verify some essential operations to ensure unexpired blood such as `dispose()`, which checks the expiration date for the blood and removes expired blood. Note that the way of verifying the list and implementing are not able to be completely matched. The decision has been explained in Section 7.3.3.

### 6.3.2 Code Matching

The corresponding relationship between Python code and Dafny code has been written as comment in the actual code and listed in the table below. We attempted to verify these functions, although not all are successfully verified. (Note: functions with name in gray exist because of inheritance)

| Python  | Dafny  | Verified |
|---|--|----------|
| <code>BloodList.py::addBlood()</code>         | <code>BloodList.dfy::addBlood()</code>         | Yes      |
| <code>BloodList.py::getBlood()</code>         | <code>BloodList.dfy::getBlood()</code>         | Yes      |
| <code>BloodList.py::removeBlood()</code>      | <code>BloodList.dfy::removeBlood()</code>      | Yes      |
| <code>BloodList.py::extractBlood()</code>     | <code>BloodList.dfy::extractBlood()</code>     | Yes      |
| <code>UntestedBloodList.py::addBlood()</code> | <code>UntestedBloodList.dfy::addBlood()</code> | Yes      |
| <code>UntestedBloodList.py::getBlood()</code> | <code>UntestedBloodList.dfy::getBlood()</code> | Yes      |

|   |  |           |
|---|--|-----------|
| UntestedBloodList.py::removeBlood()       | UntestedBloodList.dfy::removeBlood()       | Yes       |
| UntestedBloodList.py::extractBlood()      | UntestedBloodList.dfy::extractBlood()      | Yes       |
| TestedBloodList.py::addBlood()            | TestedBloodList.dfy::addBlood()            | Yes       |
| TestedBloodList.py::getBlood()            | TestedBloodList.dfy::getBlood()            | Yes       |
| TestedBloodList.py::removetBlood()        | TestedBloodList.dfy::removeBlood()         | Yes       |
| TestedBloodList.py::extractBlood()        | TestedBloodList.dfy::extractBlood()        | Yes       |
| TestedBloodList.py::testedBloodNum()      | TestedBloodList.dfy::testedBloodNum()      | Yes       |
| TestedBloodList.py::sortByExpiryDate()    | TestedBloodList.dfy::sortByExpiryDate()    | Yes       |
| TestedBloodList.py::numOfStorageCurrent() | TestedBloodList.dfy::numOfStorageCurrent() | Partially |
| TestingBloodList.py::addBlood()           | TestingBloodList.dfy::addBlood()           | Yes       |
| TesingdBloodList.py::getBlood()           | TestingBloodList.dfy::getBlood()           | Yes       |
| TestingBloodList.py::removeBlood()        | TestingBloodList.dfy::removeBlood()        | Yes       |
| TestingBloodList.py::extractBlood()       | TestingBloodList.dfy::extractBlood()       | Yes       |
| ReservedBloodList.py::addBlood()          | ReservedBloodList.dfy::addBlood()          | Yes       |
| ReservedBloodList.py::getBlood()          | ReservedBloodList.dfy::getBlood()          | Yes       |
| ReservedBloodList.py::removeBlood()       | ReservedBloodList.dfy::removeBlood()       | Yes       |
| ReservedBloodList.py::extractBlood()      | ReservedBloodList.dfy::extractBlood()      | Yes       |
| ReservedBloodList.py::sortByExpiryDate()  | ReservedBloodList.dfy::sortByExpiryDate()  | Yes       |
| DisposedBloodList.py::addBlood()          | DisposedBloodList.dfy::addBlood()          | Yes       |
| DisposedBloodList.py::getBlood()          | DisposedBloodList.dfy::getBlood()          | Yes       |
| DisposedBloodList.py::removeBlood()       | DisposedBloodList.dfy::removeBlood()       | Yes       |
| DisposedBloodList.py::extractBlood()      | DisposedBloodList.dfy::extractBlood()      | Yes       |
| Dispose.py::dispose()                     | Dispose.dfy::dispose()                     | Partially |
| Dispose.py::checkExp()                    | Dispose.dfy::checkExp()                    | Yes       |
| Doctor.py::reserveBlood()                 | Doctor.dfy::reserveBlood()                 | Yes       |
| Doctor.py::removeReservation()            | Doctor.dfy::removeReservation()            | Yes       |
| DonationManager.py::insertBlood()         | DonationManager.dfy::insertBlood()         | Yes       |
| StorehouseManager.py::doDispose()         | StorehouseManager.dfy::dispose()           | Yes       |
| StorehouseManager.py::disposeAll()        | StorehouseManager.dfy::disposeAll()        | Yes       |

|                        |                         |     |
|------------------------|-------------------------|-----|
| Tester.py::startTest() | Tester.dfy::startTest() | Yes |
| Tester.py::test()      | Tester.dfy::test()      | Yes |

### 6.3.3 Limitations

- Due to the usage of abstraction and inheritance on Python code in implementation, which could not be applied to Dafny in verification, code for different types of lists and bloods in Dafny has been duplicated to simulate the process of inheritance.
- Only pure functions could be verified. The verification of impure functions such as printing warning and sending emails have been omitted in this project.

## 7. Evaluation

### 7.1 Requirements Status Table

| Requirements  | Status          |
|---|-----------------|
| <b>Must-have</b>  |                 |
| Donation managers can insert untested blood sample information into the system                    | Implemented     |
| Testers can test and fill in the details for blood samples (e.g. type, expiration date)           | Implemented     |
| Doctors can view the storage of certain types of blood and obtain the blood from the blood bank.  | Implemented     |
| Storehouse manager can remove disposed blood from the storehouse.                                 | Implemented     |
| <b>Should-have</b>  |                 |
| Testers are able to start testing and fill in the results later.                                  | Implemented     |
| When storage of blood is low, donor manager and storehouse manager can be notified by the system. | Implemented     |
| <b>Could-have</b>   |                 |
| Doctors can reserve the blood and reservation can be cancelled.                                   | Implemented     |
| Donation manager can be notified if there is blood demonstrating a negative result in the test.   | Not Implemented |
| The amounts of blood should be measured in volume.  | Not Implemented |

### 7.2 Work Distribution

- *Everyone*: Requirements, report
- *Cher Ping Choy*: Executive summary, proof-reading, back-end and front-end for reservation functions, Doctor, Storehouse Manager, frontend report
- *Jiahui Luo*: the majority part for use cases, verification for list, back-end for doctor, storehouse manager and donation manager, backend report
- *Jiajian Cai*: most of WBS, UML for back-end, program testing, part of back-end verification

- *Wenlue Zhang*: part of WBS and some use cases, front-end coding, back-end for storehouse manager, program testing, Structure of back-end
- *Yang Zheng*: the majority part for use cases, part of WBS, autotest Script, part of back-end for Doctor, Tester, Storehouse Manager, Verification of postconditions of methods

## 7.3 Challenge and Decisions

During the project, we met these challenges and made these decisions.

### 7.3.1 Verification between Classes

To match the python code, we decided to use more classes in dafny as well. But difficulties have been met when two or more classes cooperate. The methods can be verified by themselves but when being called in other classes, many errors occur. Overall, calling instances of classes from within another class proved to be extremely difficult when using dafny. However, we verify successful by adding a great number of requires and ensures.

### 7.3.2 User Interface

Since it has been mentioned in the project specification that the user interface is not the main concern of this project, we decided upon a command line user interface. However, to make information clearer and easier to read, it has been designed to be colourful. However, the display could be different on different devices with different configurations.

### 7.3.3 Verification of List Operations

Due to different representations of list in Python and Dafny, the way of verifying the list and implementing shows a slight difference. In Dafny, all the operation will be directly applied to an Dafny array. However, in Python, composition has been applied to BloodList, which mean a DafnyList object will be held by BloodList. The infrastructure of list will be exposed by getter and setter methods. Hence, for any list operation such as addBlood() and removeBlood(), the methods of BloodList will be called first, then it will “forward” the function call to the DafnyList().

## 7.4 Further Improvements

This quality of the project can be improved by these possible improvements below.

### 7.4.1 More Verification and Test Cases

More classes can be proved in this project, if there is no time constraint. Also, by performing more tests on unverified code, it builds more confidence regarding the correctness of this system.

## 7.4.2 Better User Interface

Due to the limitations of command line, some functionalities can still misleading users in the submission version. Information can be displayed in a more intuitive way by transforming the system to a website or an application with GUI design.

## 7.4.3 More Functionalities and Less Assumptions

Due to the time restriction, some of the functionalities has been given lower priorities to implement, even though they could be useful to have them. The scope of this project has also been shrunk to only implement the most critical requirements. Some functionalities that is related to too much external environment has been omitted or simplified such as transportation and blood sample collection. As a result of this, many assumptions have been made to ensure the environment matches the preconditions of some functions. However, better application should always aim at less assumptions and less preconditions to have a better coverage.



## 8. Appendices

### 8.1 Use cases for Won't have

#### 8.1.1 Requirement 1

|                          |   |
|--------------------------|---|
| <b>Use Case 1</b>        | Record donor  |
| <b>Priority</b>          | Won't have  |
| <b>Requirement 1</b>     | Donor could be recorded when they give blood and donors can check their own record.   |
| <b>Actor</b>             | Donation Manager  |
| <b>Use Case Overview</b> | The information of the donor is recorded by the donor manager   |
| <b>Trigger</b>           | New blood received by donor manager   |
| <b>Precondition 1</b>    | The donor is registered when the donate the blood   |
| <b>Precondition 2</b>    | The information of the donor will be attached with the blood  |
| <b>Basic Flow</b>        | <ol style="list-style-type: none"><li>1. Donation Manager receive the blood with donor information</li><li>2. Donation manager type in command to add the record for the donor</li><li>3. There should be a notice indicates the record has been added successfully</li></ol> |
| <b>Business Rules</b>    | Other recording keeps the same  |
| <b>Postcondition 1</b>   | The blood must be added to the system as well   |

|                          |   |
|--------------------------|---|
| <b>Use Case 2</b>        | Check donation record   |
| <b>Priority</b>          | Won't have  |
| <b>Requirement 1</b>     | Donor could be recorded when they give blood and donors can check their own record.                     |
| <b>Actor</b>             | Donor   |
| <b>Use Case Overview</b> | Donor can check the recording for his own donation  |
| <b>Trigger</b>           | The donor wants to check his recording  |
| <b>Precondition 1</b>    | The donor has at least one record for donation  |
| <b>Basic Flow</b>        | <ol style="list-style-type: none"><li>1. Donor type in the view command which includes the id</li></ol> |

|                       |   |
|-----------------------|---|
|                       | <p>of the donor to the system</p> <p>2. The list of donation record which includes the date of donation and the id and status of blood is shown</p> |
| <b>Business Rules</b> | The recording keeps the same  |

### 8.1.2 Requirement 2

|                           |   |
|---------------------------|---|
| <b>Use Case 1</b>         | Patient view list of blood  |
| <b>Priority</b>           | Won't have  |
| <b>Requirement 2</b>      | Patient can check the blood storage of certain types of blood and book the blood.   |
| <b>Actor</b>              | Patient   |
| <b>Use Case Overview</b>  | Patient can view a list of blood of certain type  |
| <b>Trigger</b>            | Patient wants to check the storage of certain type of blood   |
| <b>Basic Flow</b>         | <ol style="list-style-type: none"> <li>1. Patient add the command which contains the type of blood to ask for certain type of blood</li> <li>2. The list of tested blood of that type shows in the system ordered by date of expiry</li> <li>3. The id of blood also shows in the list</li> </ol> |
| <b>Alternative Flow 1</b> | <ol style="list-style-type: none"> <li>1. Patient add the command which contains the type of blood to ask for certain type of blood</li> <li>2. "No record" is displayed to inform the Patient</li> </ol>   |
| <b>Business Rules</b>     | <ul style="list-style-type: none"> <li>• The blood shows in the list must have been tested</li> <li>• The blood shows in the list must be unexpired</li> <li>• The blood shows in the list must be available to use</li> </ul>  |
| <b>Postcondition 1</b>    | Patient remembered the id of blood if they want to book   |

|                          |   |
|--------------------------|---|
| <b>Use Case 2</b>        | Patient book the blood  |
| <b>Priority</b>          | Won't have  |
| <b>Requirement 3</b>     | Patient can check the blood storage of certain types of blood and book the blood. |
| <b>Actor</b>             | Patient   |
| <b>Use Case Overview</b> | Patient can book the blood  |
| <b>Trigger</b>           | Patient wants to book the blood   |

|                        |  |
|------------------------|--|
| <b>Precondition 1</b>  | Patient already knows the id of the blood they want ( if don't know, view list of blood first  |
| <b>Precondition 2</b>  | The patient already got permission from the doctor for booking the blood   |
| <b>Basic Flow</b>      | <ol style="list-style-type: none"> <li>1. Patient type in book command which include the id of patient, id of blood, time for use.</li> <li>2. There should be a notice indicates that the obtain succeeded</li> </ol> |
| <b>Business Rules</b>  | <ul style="list-style-type: none"> <li>• The blood booked must be unexpired</li> <li>• The blood booked must have been tested</li> </ul>   |
| <b>Postcondition 1</b> | The status of this blood change to 'booked by patients'  |