**National University of Computer and Emerging Sciences**

# Software Engineering

# Blood Bank Management System

**DEGREE PROGRAM:** **BSCS**
**SECTION:** **D**
**DATE OF SUBMISSION:** **June 18, 2021**
**SUBMITTED TO:** **Ma'am Maryam Ghafoor**



| Names | Roll No. |
|---|---|
| Zeeshan Ikram | I18-0660 |
| Wajeeha Malik | I18-1587 |
| Sundus Khalid | I18-0666 |

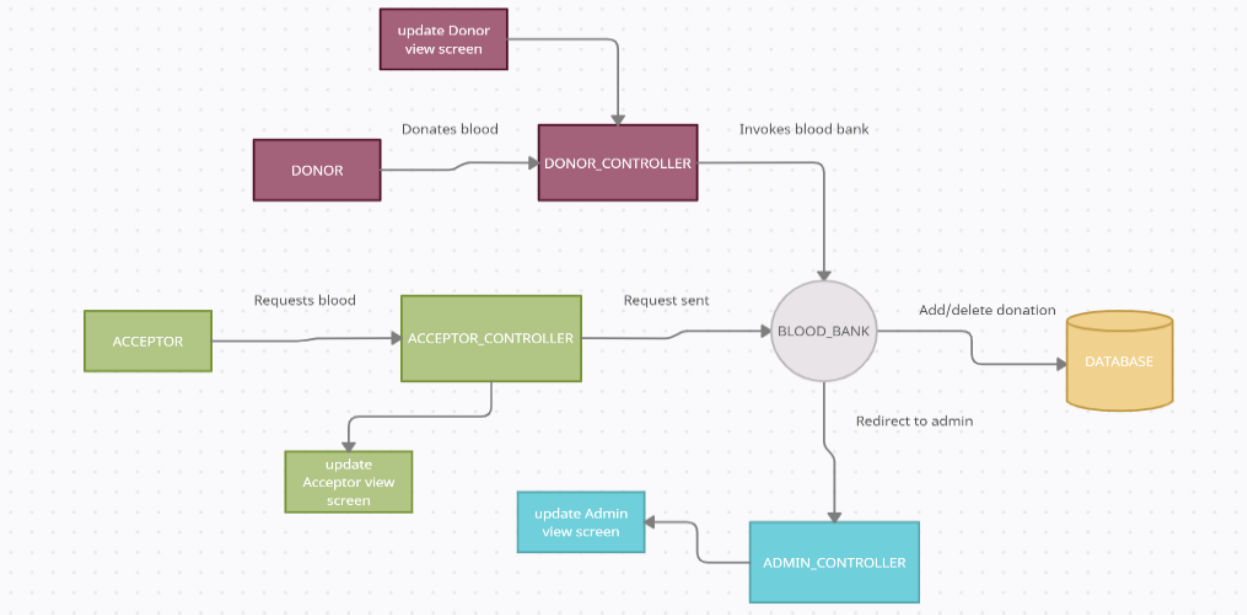
# Contents

## Introduction

The basic goal of Blood Bank Management System (BBMS) is to ***provide blood donation service***. BBMS is an automated ***Web-based*** application that is designed to store, process, retrieve and analyze information within the blood bank. This project aims at ***maintaining information*** about donors, acceptors, available blood groups and helps them manage in a better way. It will provide ***transparency*** in the process of obtaining blood from the blood bank. It makes the system of blood bank management ***hassle-free*** and ***effective***.

## Software Architecture

**(MVC)**




### Reasons:

- The reason we've chosen MVC architecture is because it makes software development smoother and robust as compared to other architectures as it allows reusing of code and develops simultaneous applications.
- Unlike traditional approach of programming, MVC approach helps you create applications that separate the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.
- The pattern specifies where each kind of logic should be located in the application.
- The UI logic belongs in the view. Input logic belongs in the controller. Business logic belongs in the model.
- This separation helps you manage complexity when you build an application, because it enables you to focus on one aspect of the implementation at a time.
- For example, you can focus on the view without depending on the business logic

## Benefits:

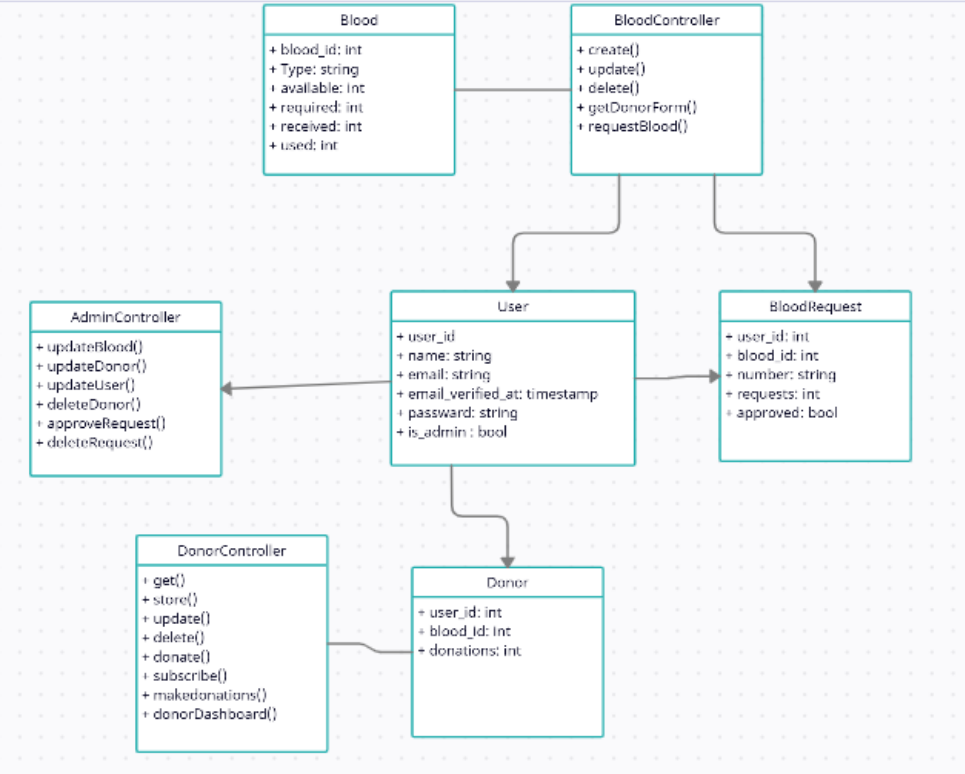
- MVC model returns the data without the need of formatting.
- Modification doesn't affect entire model.
- Multiple developers can work simultaneously on the model, controller and views.
- MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- The nature of the MVC framework is such that there is low coupling among models, views or controllers.
- Because of the separation of responsibilities, future development or modification is easier i.e. scalability of the product is increased.
- Models can have multiple views.

## Limitations:

- The framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria of MVC.
- Decomposing a feature into three artifacts causes scattering. Thus, requiring developers to maintain the consistency of multiple representations at once.

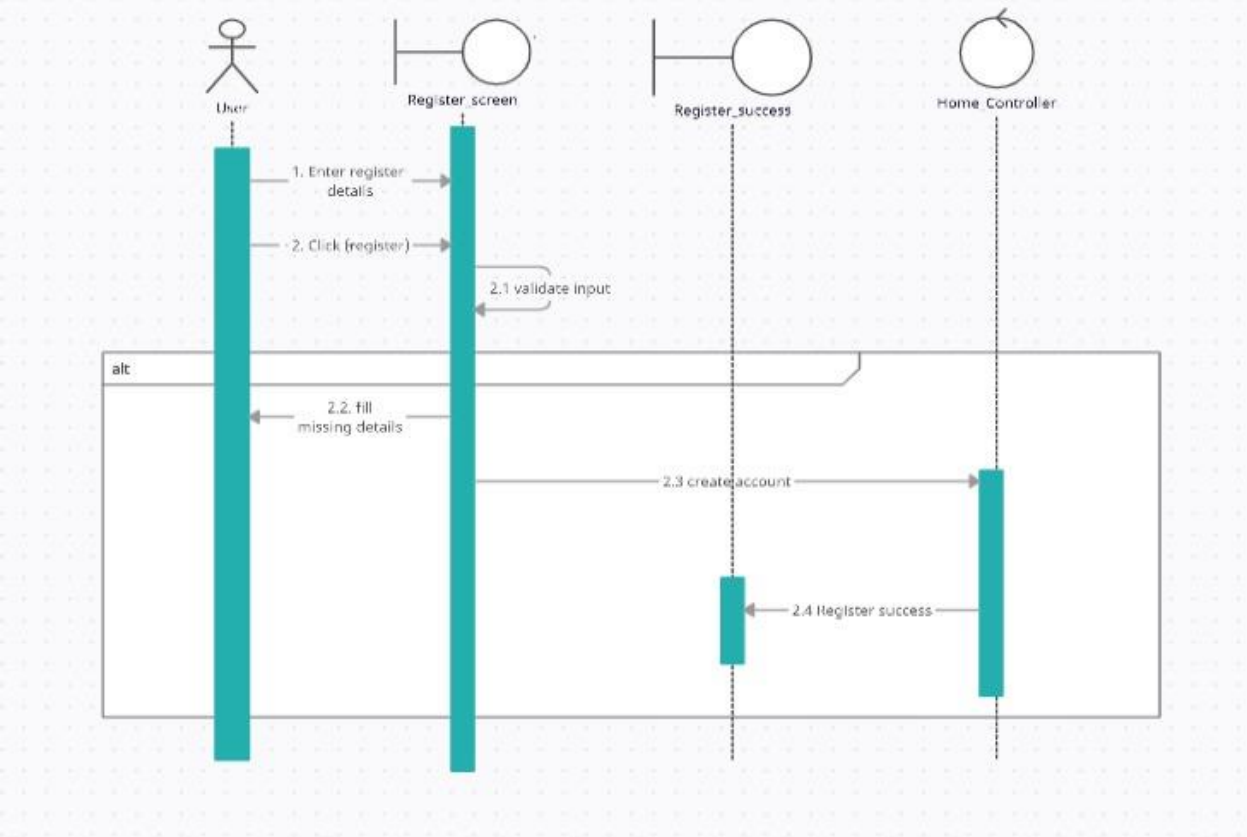
## UML diagram:-

**Class Diagram:**

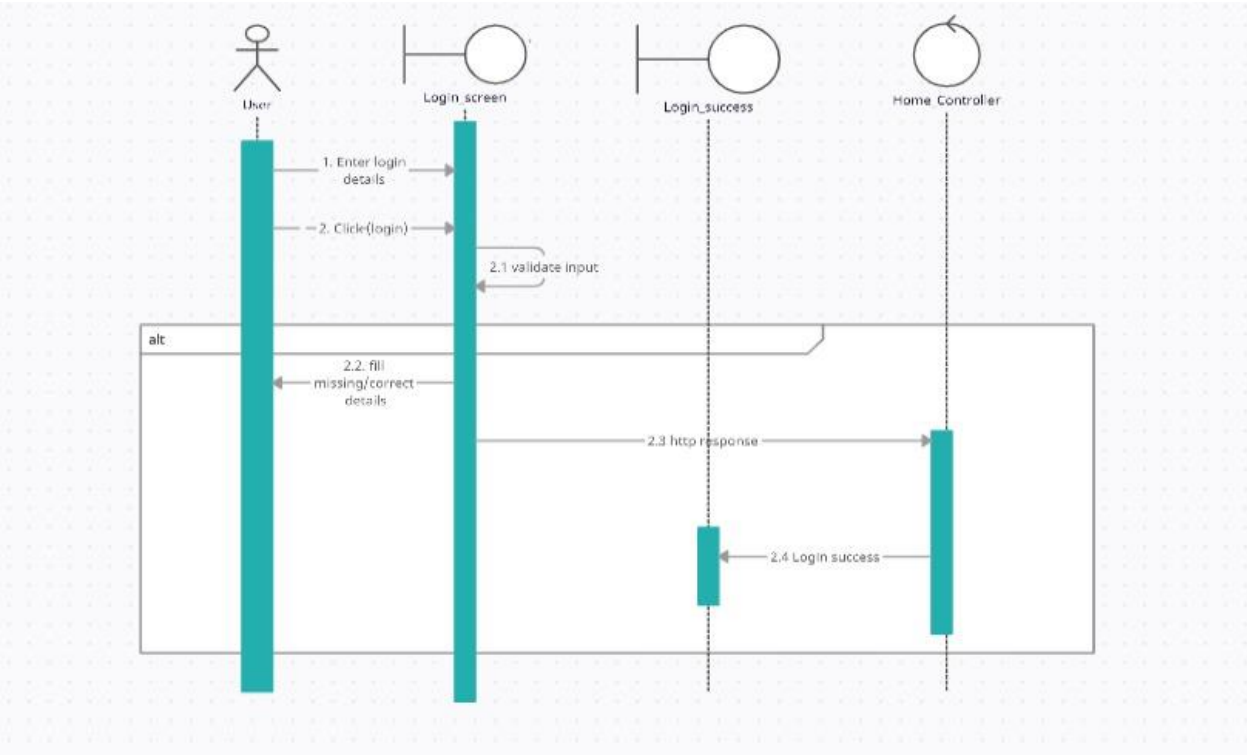



**Sequence Diagram:**

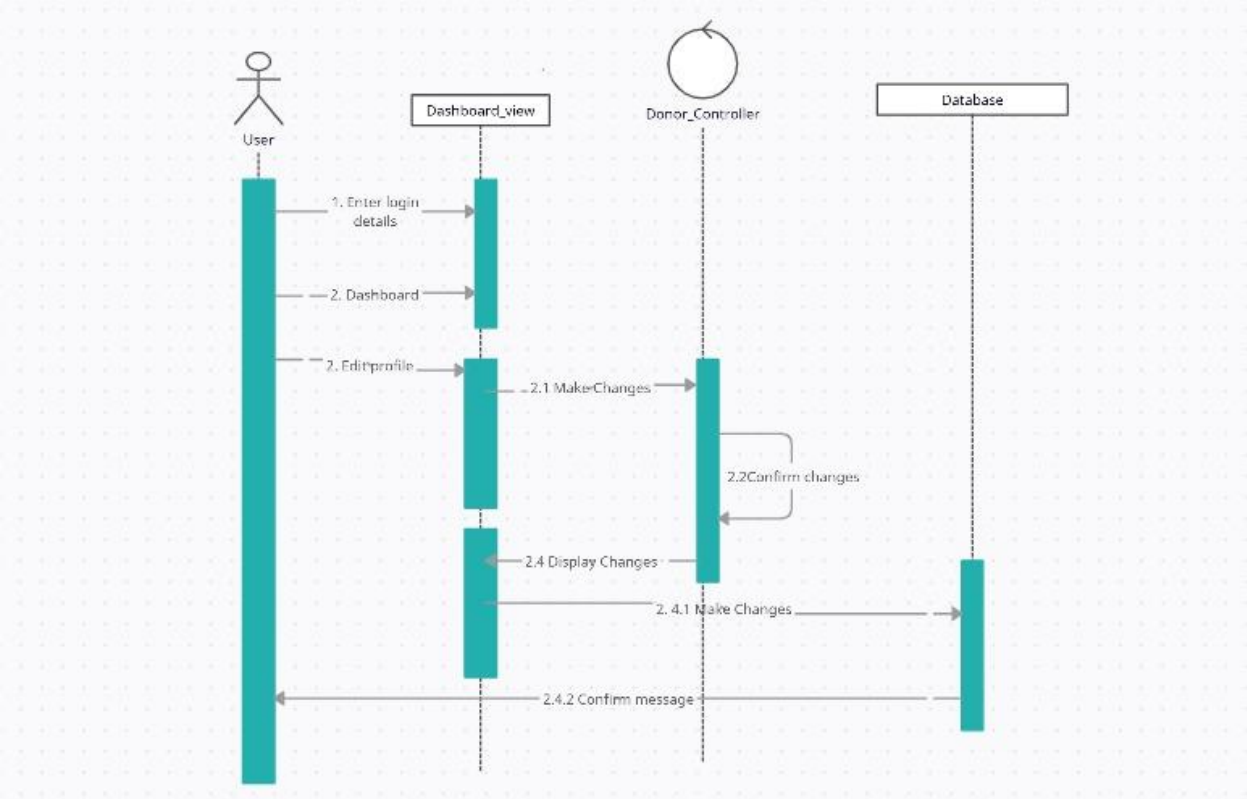
Registration:


User
Register_screen
Register_success
Home_Controller
1. Enter register details
2. Click (register)
2.1 validate input
alt
2.2. fill missing details
2.3 create account
2.4 Register success

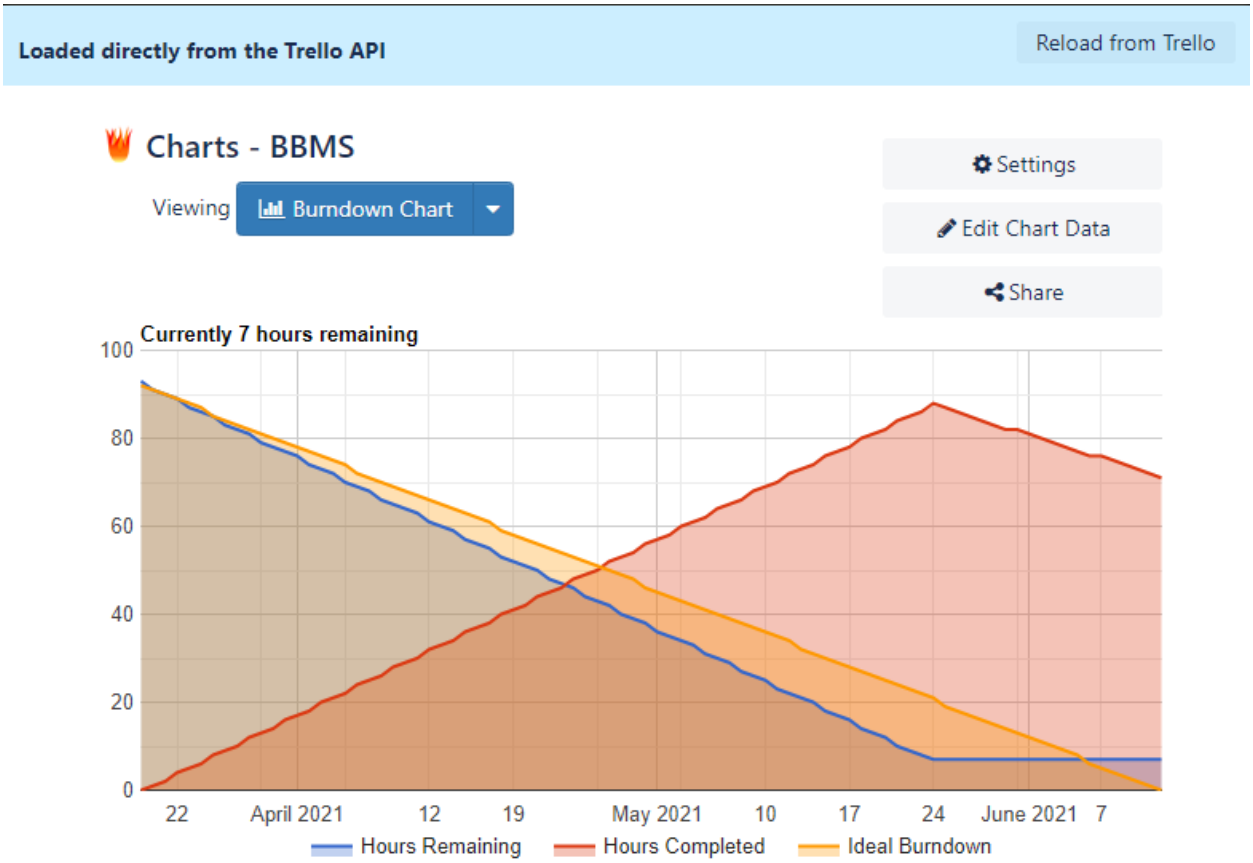
Login:


User
Login_screen
Login_success
Home_Controller
1. Enter login details
2. Click(login)
2.1 validate input
alt
2.2. fill missing/correct details
2.3 http response
2.4 Login success

Update profile:



# Burn-down Chart




# Product Review

## Improvements:

- The interface we are using is simple in design. It can be further improved with animations and effects.
- The application can also have a smartphone-supported version so that our users can use BBMS on both phone and desktop. This can also help us in increasing our users.
- The software can be further improved by adding a search by location functionality. So that a user can have an option to limit the search to an area to facilitate the search for blood.
- The software can be improved by adding a functionality of receiving notification on email and phone. This will help the users fasten the process of blood donation.

## Limitations/ Constraints:

- It is up to the admin to accept or decline blood requests. Also, only admin can delete the user profiles.
- Notifications can only be received by the users when they're logged-in in the system.

## Product Backlog:-

| User Story | Replaced By | Reason for change | Task | Developed in | Developed by | Time | Comments |
|---|---|---|---|---|---|---|---|
| As a user, I want to register in the system so that I can enter my information in BBMS. | Not changed. | - | 1) Design UI for registration<br>2) Code in Laravel , HTML + CSS<br>3) Maintain Database to store user information<br>4) Test Database connection | Sprint 1 | Wajeeha | 20 | It was completed in sprint 1 |
| As a user, I want to search available blood by blood group or by location so that I can find blood easily | As a user, I want to make blood request to admin so that I can receive the blood I need. | We decided not to implement the location functionality. | 1) Design UI for requesting blood<br>2) Code in Laravel<br>3) Check database for the available donors.<br>4) Test request constraints | Sprint 2 | Sundus | 15 | Couldn't implement location functionality because of the limited time. |
| As a user, I want to update/delete my profile so that I can change my details accordingly. | As a user, I want to modify my profile so that I can change my details according ly | We decided to give the authority of deletion to admin. | 1) Code in Laravel<br>2) Modify data in database<br>3)Make UI that helps in updating profile<br>4)Test update constraints | Part of sprint 2 but completed in sprint 3 | Sundus | 15 | We decided not to implement it because most of the system control was given to the admin. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| As a user, I want to login to the system so that I can use BBMS features. | Not changed. | - | 1) Design UI for registration<br>2) Code in Laravel , HTML + CSS<br>3) Test entered data validity | Sprint 1 | Wajeeha | 18 | - |
| As a donor, I want to be able to donate my blood so that people can accept blood. | Not changed | - | 1)Design UI for donation of blood<br>2)Code in Laravel<br>3)Maintain Database to store all the information<br>4)Test donation constraints | Sprint 3 | Zeeshan | 23 | - |
| As a user, I want to view my profile so that I can view my information. | Not changed. | - | 1) Code in laravel<br>2) Modify data in database<br>3) Make a UI for profile page. | Sprint 2 | Zeeshan | 19 | - |
| As a user I want to reset my password so that I can secure my account | Not changed. | - | 1) Code in laravel<br>2) Update database with new password<br>3) Make a UI for resetting password<br>4) Test password resetting constraints | Sprint 2 | Wajeeha+ Zeeshan | 11 | - |
| As an admin, I want to update/delete donor profile so that I can manage donors' details accordingly | Not changed. | - | 1) Code in Laravel<br><br>2) Modify data in database | Sprint 3 | Sundus+ Wajeeha | 25 | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 3) Make a UI that helps in updating / deleting profile<br>4) Test update/ delete constraints | | | | |
| As an admin, I want to accept or ignore blood requests so that I can manage requests accordingly. | Not changed. | - | 1) Code in Laravel<br><br>2) Modify data in database<br>3) Test ignore and accept constraints | Sprint 3 | Zeeshan + Wajeeha+Sundus | 17 | - |
| As an admin, I want to view inventory so that I can see inventory details accordingly. | Not changed. | - | 1) Code in Laravel<br><br>2) Modify data in database | Sprint 3 | Zeeshan | 19 | - |

# Scrum Retrospective

## Work division:

- For sprint 1 we had planned to make database connection(MySql). We wrote the code for login and registration. We created the interface so that users can register and login. After that, we created the test cases for login and registration.
- For sprint 2 we had planned to write code, create interface and create test cases of update, delete and donate. But we shifted the update and delete functionalities to sprint 3.
- For sprint 3 we planned to write code, create interface and create test cases for admin functionalities like accept or decline the request, view inventory, update/delete profile, search and donate blood.

## Process:

- **Conceptual planning:** First off we decide the rolls and responsibilities of the members. We decided the functionalities we have to implement. We assigned the tasks to the members

accordingly. We decided which database to use and which languages to use. Initially, we decided to use mongoDB as database and Django as a framework.

- **Planning requirements:** We gathered the requirements from the product owner. We made a product backlog. We assigned the tasks for each sprint. Every task was given a completion estimate. We made a daily sprint backlog as well to complete the work on time.
- **Development and Testing:** We established a database connection and made a user interface. After that we wrote the code for the tasks and functionalities required. We created the test cases using Katalon to test our code. We tested the functionalities of the code and updated our code accordingly.
- **Operations and Maintenance:** We monitored the performance of the software and repaired the defects or issues we found during this phase. Added functionalities that didn't affect the reliability of our software.

## Improvements in process:

We didn't feel any issues while following the process and how we carried out our work. So we don't feel the need to add any improvements in our process.

## Work Performance Improvement:

- The work can be improved by using a database of which we have a beforehand knowledge about. This can cause delays in sprint work. The database connection making an testing should be the first priority when it comes to the implementation.
- Work performance can be improved by the communication between the top and bottom hierarchy. The group leaders must keep their team up-to-date.

## Difficulties and solutions:

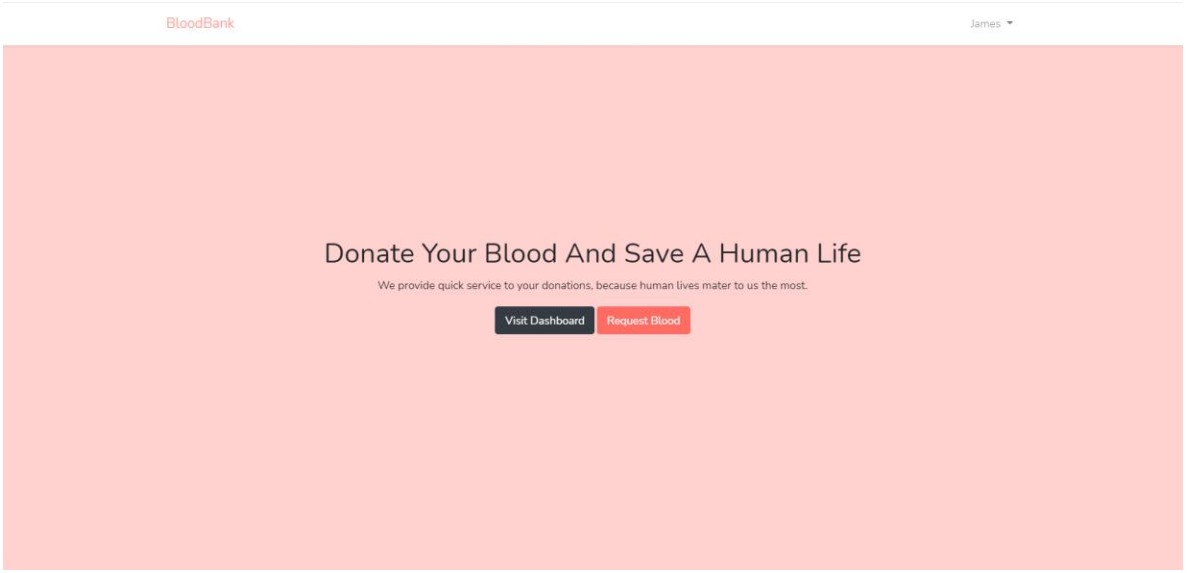
- Database connection issue: Initially we decided to work in Django but we faced the database connection issues so we had to switch to MySql which is easier to establish the connection.
- Update Delete issue: While updating the profiles the new information was not being updated in database. This issue was resolved when we added few constraints and tested the database connection.
- Password issue: Some of the test cases of passwords were failing because of some issue. After adding validity checks this issue was rectified.

## Lessons learned:

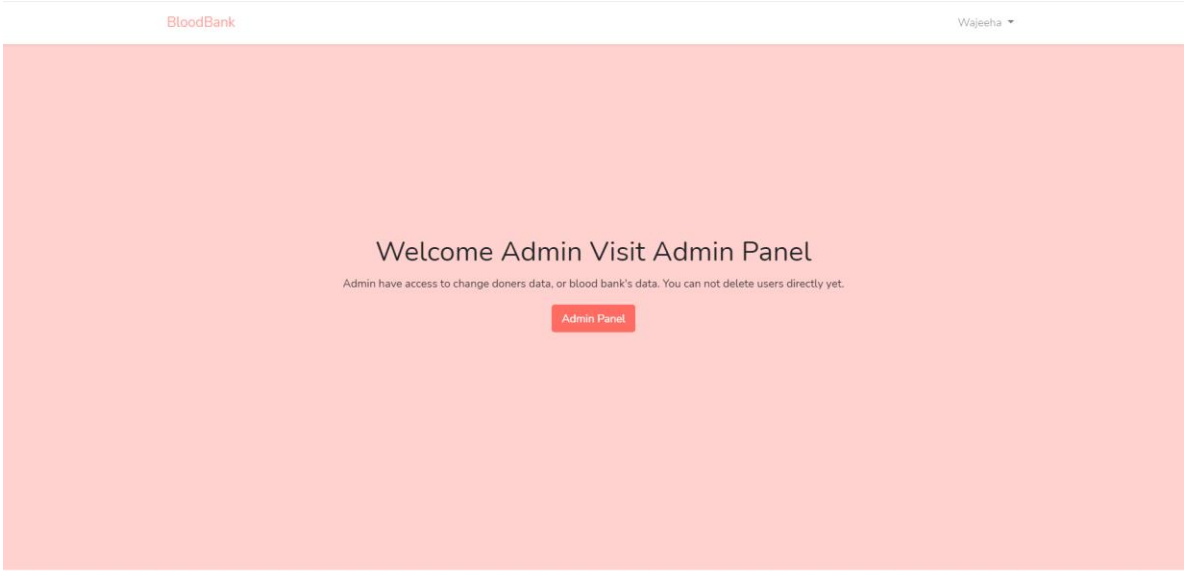
- Always make a backup after implementing something.
- Change one thing at a time so in case if we face any problem it is easier to know what caused the problem.
- A piece of code can take longer than anticipated so don't panic and be prepared for it.
- Be mentally prepared to face unexpected problems.

# Graphical User Interface

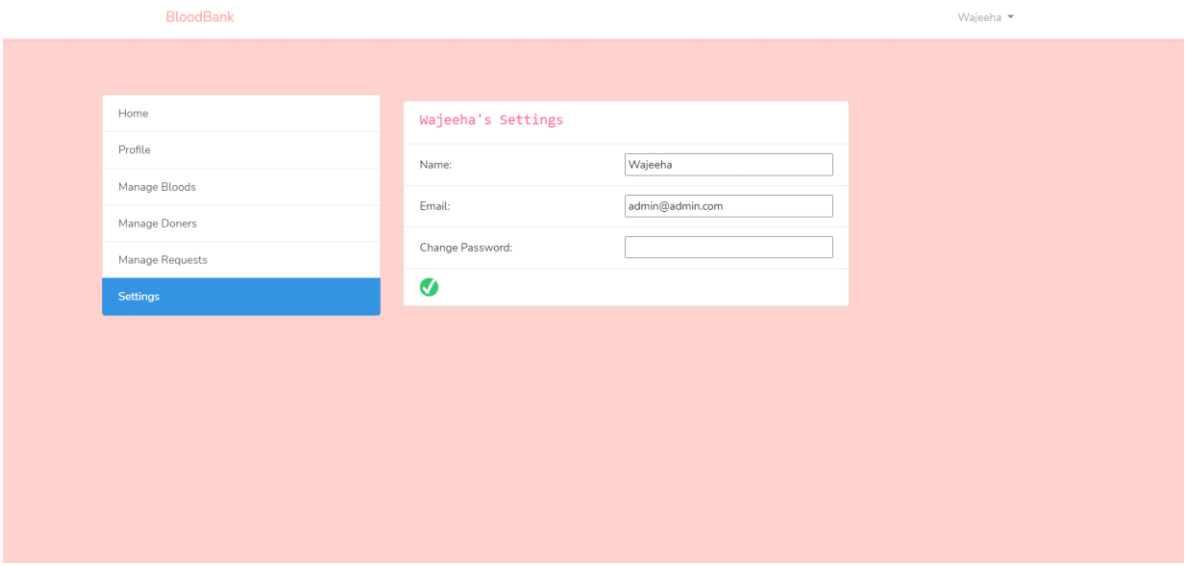
User Frontend:



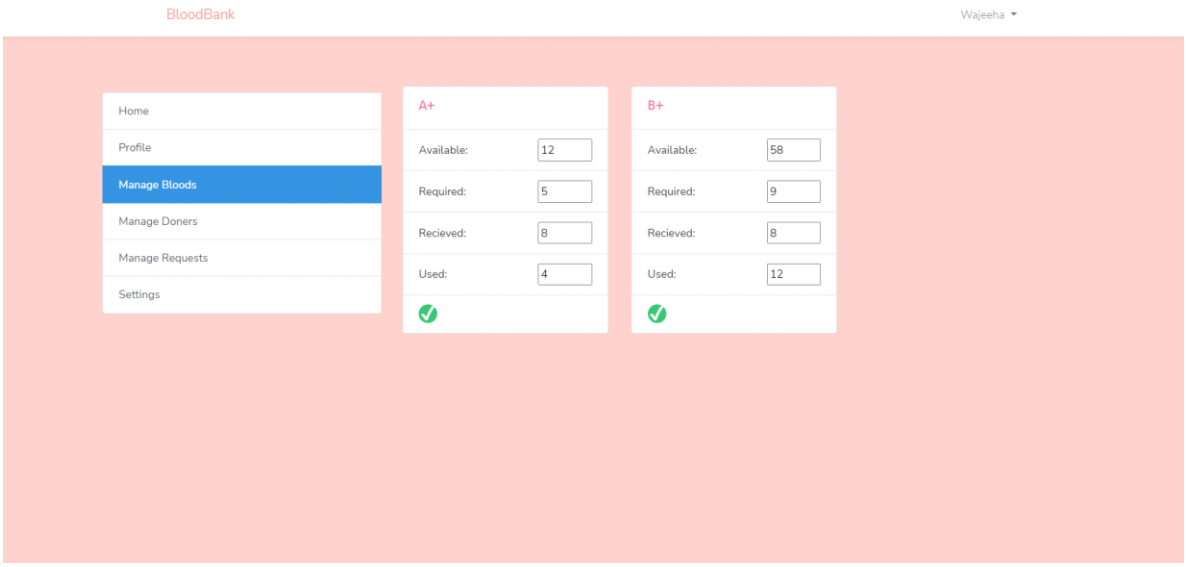

Admin Frontend:



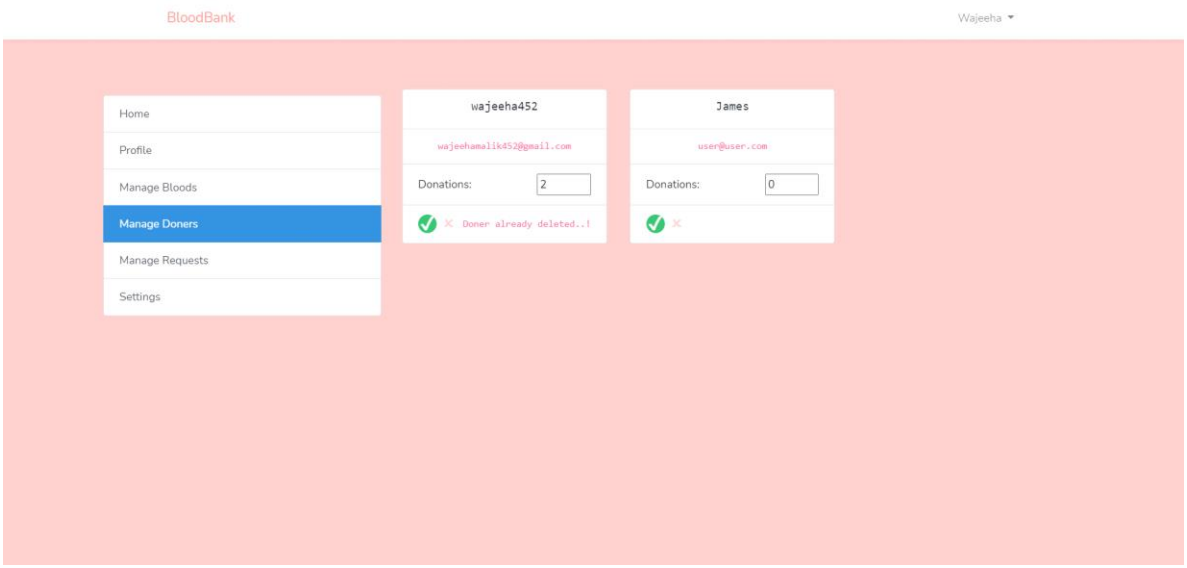

Settings:



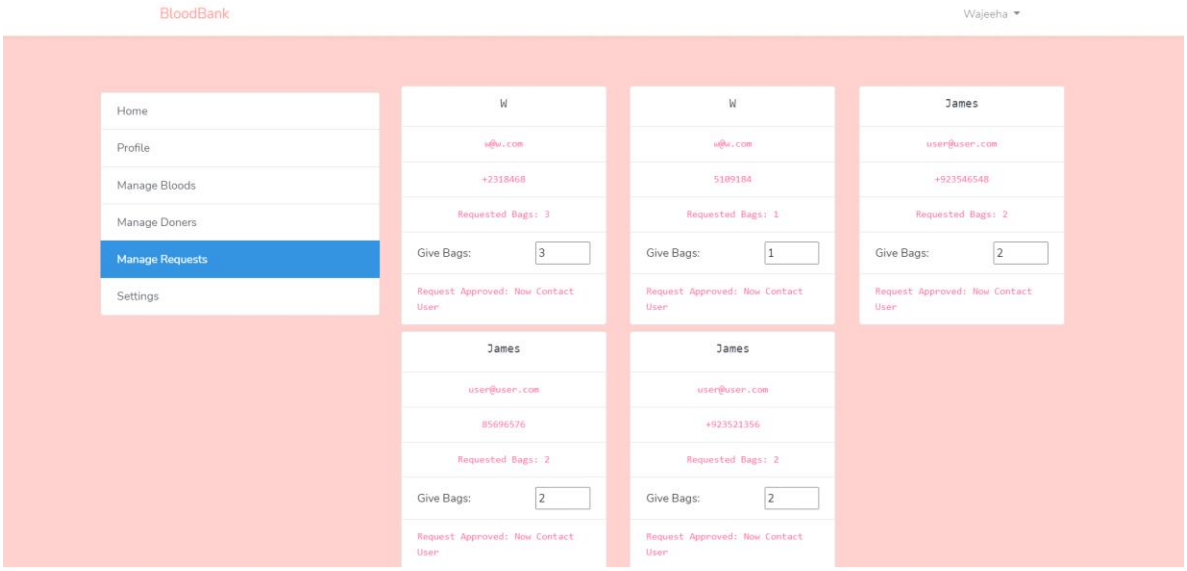

Blood Management:



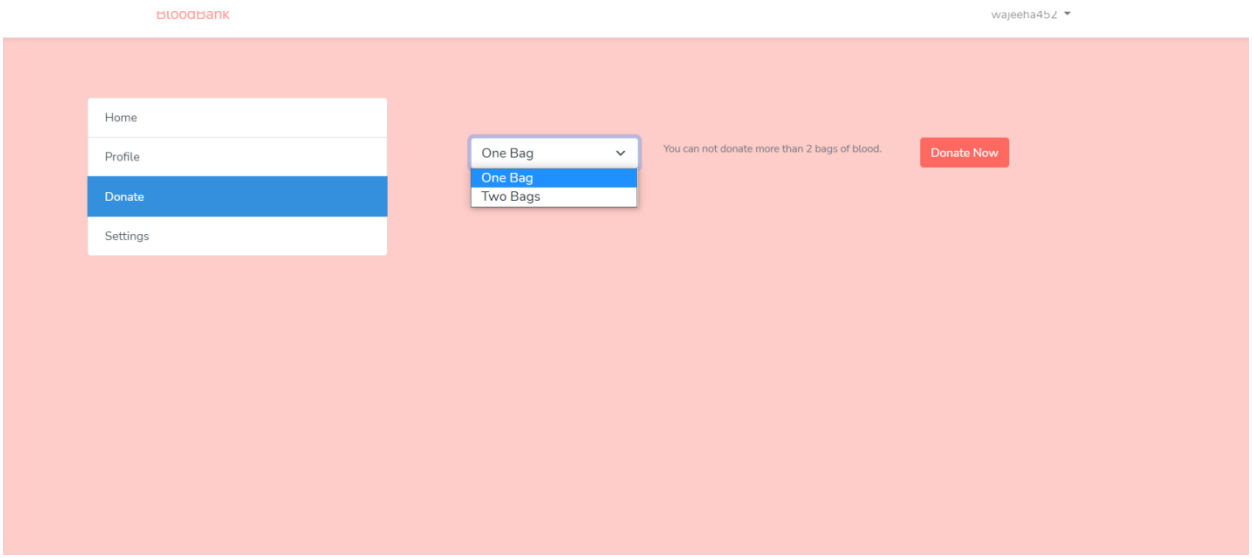

Donor Profiles Management:



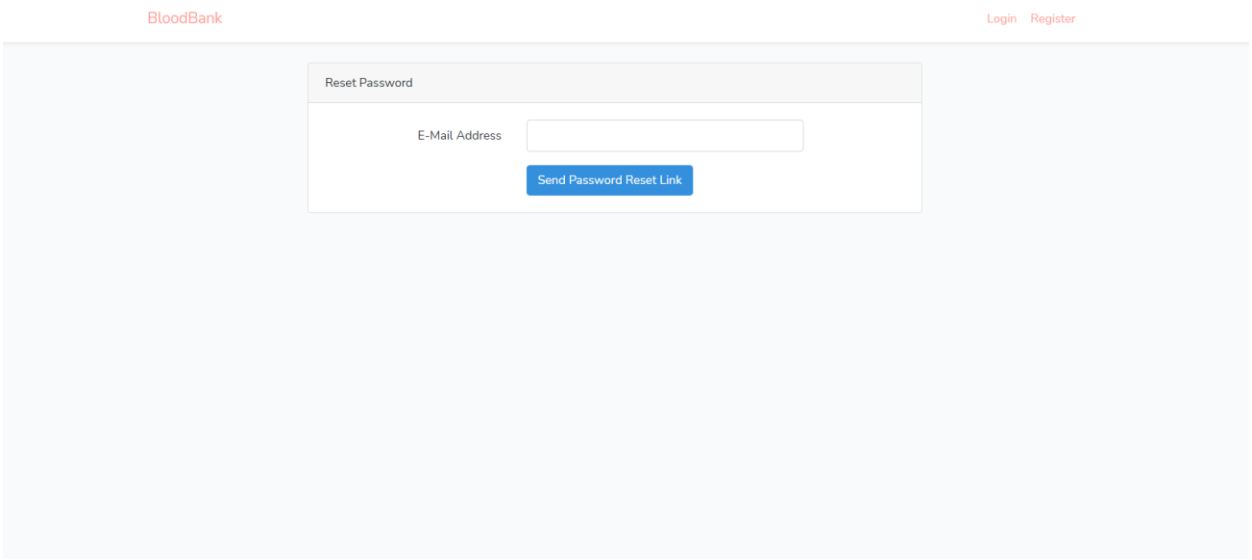

Blood Requests Management:



Donor profile:




Reset Password:



# Testing

## Boundary Values:

PASSWORD: Password field accepts minimum of 8 characters.

| Boundary value | |
|---|---|
| VALID (min, min+1) | INVALID (min-1) |
| 8, 9 | 7 |

REQUEST DONATIONS: User can only request blood bags inside available blood bags range

| Boundary value | |
|---|---|
| VALID (max, max-1) | INVALID (max+1) |
| available bags, available bags-1 | available bags+1 |

DONATIONS: User can only donate two blood bags due to health concerns.

| Boundary value | |
|---|---|
| VALID (min, min+1, max, max-1) | INVALID (min-1, max+1) |
| 1, 2 | 0,3 |

PHONE NUMBER: User can only enter 10 digits.

| Boundary value | |
|---|---|
| VALID (max) | INVALID (max+1, min-1) |
| 10 | 11,9 |

## Equivalence Partitioning:

PASSWORD: Password field accepts minimum of 8 characters.

| Equivalence partitioning | |
|---|---|
| VALID | INVALID |
| Characters > 8 | Characters <= 8 |

REQUEST DONATIONS: User can only request blood bags inside available blood bags range

| Equivalence partitioning | |
|---|---|
| VALID | INVALID |
| Request < available bags | Request > available bags |

DONATIONS: User can only donate two blood bags due to health concerns.

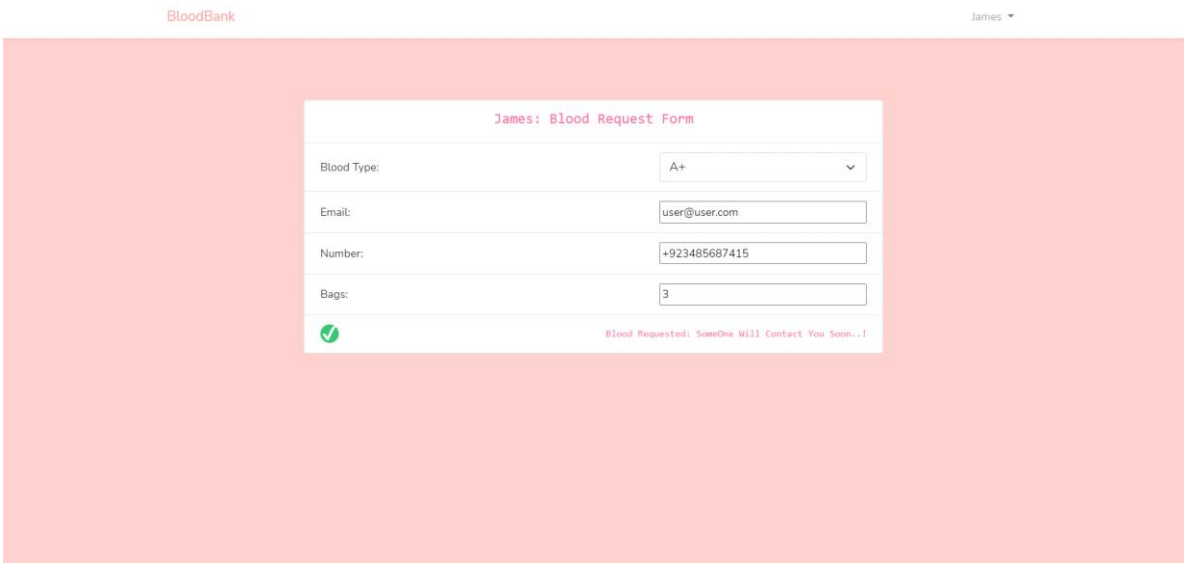
| Equivalence partitioning | |
|---|---|
| VALID | INVALID |
| 1 < Bags <= 2 | 2 < Bags <= 0 |

PHONE NUMBER: User can only enter 10 digits.

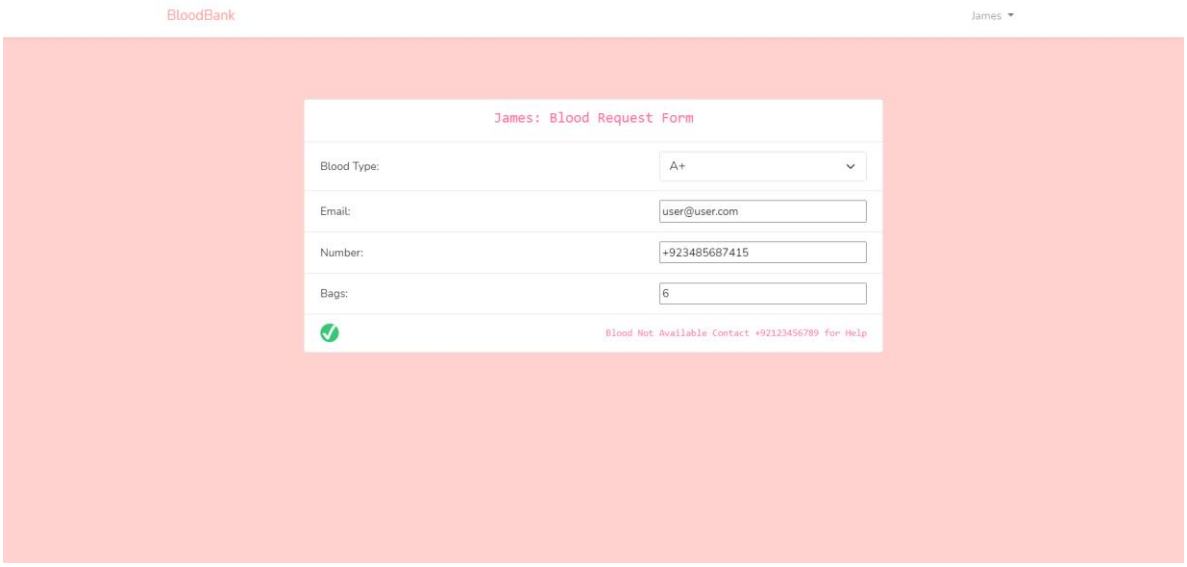
| Equivalence partitioning | |
|---|---|
| VALID | INVALID |
| Phone = 10 | 10 < Phone <= 0 |

## Snapshots:

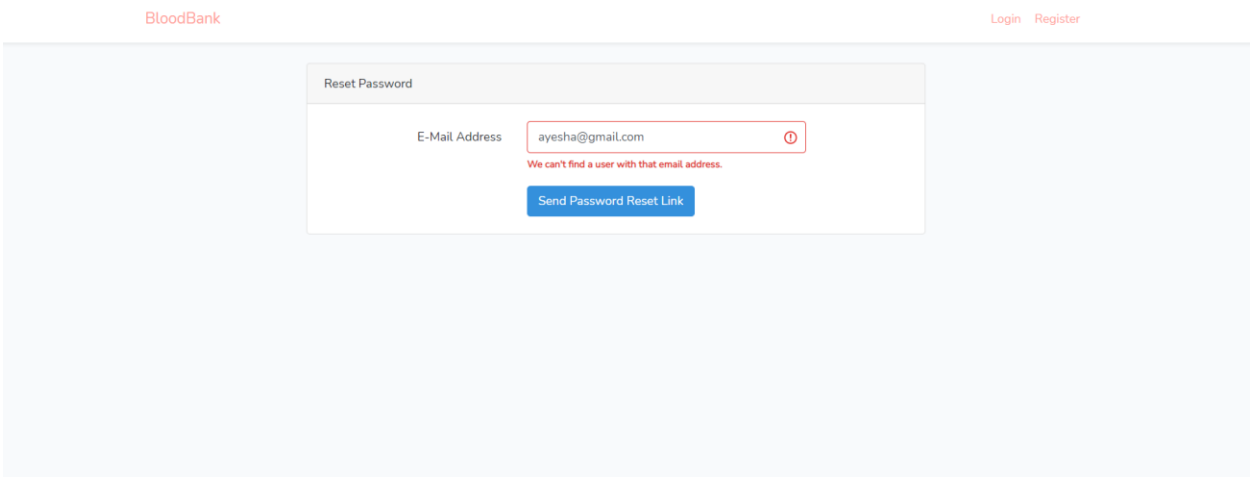
Blood Request 1:



Blood Request 2:




Email verification:



Input validation: