**Blood Donation System**

**SPROJ Report**



**Ahmad Aslam 22100221  
Saad Azam 22100082  
Muhammad Bilal 22100224**

**Shehroz Atir 22100337**

**Advisor: Dr. Waqar Ahmad  
School of Science and Engineering  
Lahore University of Management Sciences  
Submission Date: 30 April 2022**

**Acknowledgement and Dedication**

**Certificate**

I certify that the senior project titled “**Blood Donation system**” was completed under my supervision by the following students:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# List of Figures

# Introduction

## Introduction

Blood Donating System is an Android, iOS and web-based app that provides a platform to blood donors and receivers for successful blood donation. It is going to make blood donation accessible to the one in need. The potential users of the app are blood donors, NGOs and blood recipients.

In the app, a recipient is going to ask for a donation of blood. The request will be pushed to a newsfeed where a donor will willingly respond to the request. A chat module will open between the donor and the receiver where they can communicate regarding the blood donation. On successful blood donation, the donor will be rewarded with points.

The app is going to be android, iOS and web-based. The main technologies that we are going to use for our app are Ruby on Rails, React and React Native.

## Objective and Scope

The overall objectives of the app are:

* Expedite the process of blood donation.
* Encourage NGOs to provide blood donors to the system.
* Through a system of ratings, points and rewards, encourage more and more donors to use the app.
* Provide an easy communication between blood donors and blood receivers.
* Make blood donation easily accessible to the one in need.

## Development Methodology

We will be using **Agile** methodology for project management and software development.

**Reason:**

Over methodology that we are using in this course is agile where we submit

deliverable and then receive feedback. Then as a whole team we incorporate the changes suggested in the feedback to improve quality of the deliverable. So, agile methodology is well aligned with the course because in upcoming we will be create a working prototype of the system and the improve it sequentially which is basically agile. Moreover, it will help us complete project quickly. Since we are developing a social app which require a lot of feedback from end users so agile is best suitable for us because it will be easy to incorporate changes. Agile is becoming more popular day by day so implementing agile on our project will also give us experience of industry standards. There will be less pressure of the deadlines considering the improvements that we can make in the next iteration.

We will use standard practices to implement agile methodology. We will use Trello for task management. We will divide each phase into small tasks and assign the tasks to different team member. We will also have scrum meeting to evaluate the progress of the project and define the sprints.

## Contributions

The main motivation of our project was to provide a very simple and minimalistic solution to a naive problem that is always present in the society. We carefully decided the design language and features to keep up the minimalistic aesthetic of the app.

# System Requirements

Brief introduction of this chapter in a paragraph highlighting the content

## System Actors

|  |  |  |
| --- | --- | --- |
| **Actor** | **Description** | |
| Admin | Admin has full knowledge and control of the system. Admin acts as a moderator and has direct access to the whole system. | |
| **Functionalities** | | |
| Admin can:   * Register NGOs * Authorize NGOs * Sign in to his/her account * View feedbacks/complaints/reports regarding the app * View logs in order to detect any system failure or error * Log out of the app * Make internal changes in the app. | | |
| NGOs | NGOs will provide blood donors to the system which will in result assist in blood donation. In result of their services, they will receive more points, ratings and popularity in the app. | |
| **Functionalities** | | |
| NGOs can:   * Register their accounts * If they are registered, sign in their accounts * Provide data about their blood donors to the systems and provide its blood donors their login details. * Create, Read, Update and Delete data of their blood donors * Notify blood donors about blood donations request according to their eligibility and location * Respond to association requests if a donor wants to join that NGO * View user activity of their blood donors * Edit their own profile * If they want to drop their services, request the admin to remove them from the system | | |
| Blood donors | Blood donors are the actual main service provider to our system. They are going to donate blood to someone in need of blood. On fulfilment of a request, they will receive rewards and points. | |
| **Functionalities** | | |
| Blood donors can:   * Sign up their accounts * If they are registered, sign in their accounts * Respond to blood requests on newsfeed * Chat with a blood receiver whose request they have accepted * View/Edit their profile * Request an NGO to join with it * Look up NGOs according to their names * Log out of their accounts * View requests of blood receivers | | |
| Blood receiver | | They are the beneficiary of our service as they are the one who are in need of donation of blood. |
| **Functionalities** | | |
| Blood receivers can:   * Sign up their accounts * If they are signed up, sign in their accounts * Push notification for blood request on newsfeed * Chat with blood donors * View/Edit their profile * View notifications regarding the blood donors who have accepted their request | | |

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Sr#** | **Actor** | **Requirement** |
| 1 |  | There will be login/logout system for users (admin, blood donors, blood receivers). |
| 2 | System will show updated news feed every time new request is made. |
| 3 | System will send push notifications to nearest donors of recipient, depending upon location. |
| 4 | System will automatically give points to NGOs and blood donors, on handling each request depending upon reviews/feedback/ratings. |
| 5 | **As an admin,** | I want to register and authorize NGOs to check authenticity of NGO. |
| 6 | I want to view feedback, complaints and logs to make constructive updates. |
| 7 | I want to notice any system failure/error to revise system issues. |
| 8 | I want to monitor users (NGO, blood donors, blood receivers) activity for security constraints. |
| 9 | I want to make internal changes in app, for any updates if required. |
| 10 | I want to handle termination requests of NGOs to clear their data. |
| 11 | **As an NGO,** | I want to CRUD blood donors so that I can handle updated blood donor data depending upon their availability. |
| 12 | I want to send login details to my registered blood donors so that they can respond to blood donation requests. |
| 13 | I want to respond to Onboarding requests, to handle associations by analyzing their health report. |
| 14 | I want to view user activity of my associated blood donors, so that check their eligibility for blood donation. |
| 15 | I want to view/edit my user profile to keep my profile up-to-date. |
| I want to request admin to terminate my account in case I want to end my services for the app. |
| 16 | **As a blood donor,** | I want to see blood donor information and respond to blood request on newsfeed/notification to respond blood requests. |
| 17 | I want to chat with blood recipient to stay in touch with the recipient. |
| 18 | I want to view and make changes to my profile to keep it up-do-date. |
| 19 | I want to look up for an NGO if I want to associate myself to it. |
| 20 | **As a blood recipient,** | I want to put a request for blood depending upon location. |
| 21 | I want to chat with blood donor to communicate about further details. |
| 22 | I want to view and edit my profile for any updates, if required. |
| 23 | I want to see notification of accepted blood requests. |
| 24 | I want to see blood donor information, when donor accepts the request. |
| 25 | I want to give feedback/reviews/rating to blood donor. |

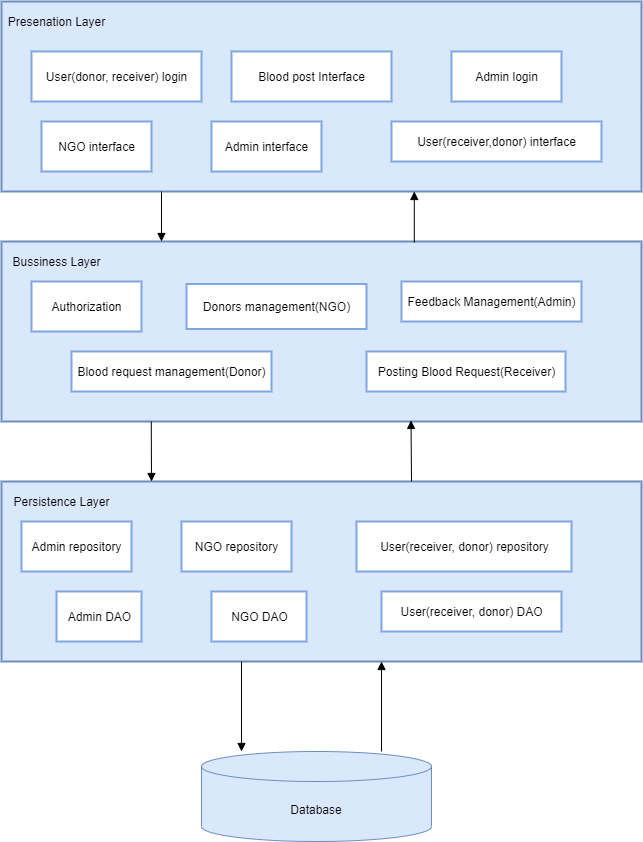
## Non-functional Requirements

|  |  |
| --- | --- |
| **Sr#** | **Requirements** |
| 1 | App will be available for Android and IOS users. Moreover, web-based App will also increase the availability of the system. Moreover, MTBF (Mean Time Between Failures) will be more than 48 hours which means that system should not fail more than 4 times a week and MTTR (Mean Time To Repair) will be less than 5 hours which means system should be operational within 5 hours after the failure is detected. |
| 2 | Download size of the Application will not be more than 150 Mbs. |
| 3 | Any requested page should not take more than 7 seconds to load with average internet speed of 10 Mbps. |
| 4 | Upon entering the correct login details, user should be able to access the app within 5 seconds. |
| 5 | System should be scalable which means it should be capable of handling at least 10000 users simultaneously. |
| 6 | System will force user to create strong password, at least 8 character long containing combination of alphabet, number and character, to ensure account security. System will use Password authenticator to determine the strength of the given passwords based on the number of alphabets, characters and digits and display pop up message accordingly. |
| 7 | System will display correct and understandable error messages to user if user is doing something wrong. For Example: entering wrong blood group. |
| 8 | Any personal data of the user sent to server will be encrypted. This will ensure that the data is not usable in case of any data theft. |
| 9 | System will be able to correctly declare log about module crashing. |
| 10 | System will be robust against cyber security attacks. Input validation attacks like SQL injection, Buffer overflow and XSS attacks will be prevented using centralized validation approach rather than relying on client-side validation. |
| 11 | Current state of the database will be backed up bi-weekly and there will also be option for emergency backup in case on any security threat or hardware failure. |
| 12 | Reusability of the code will be ensured with different techniques like modularity, high cohesion and loose coupling. |
| 13 | Since the users of the app will be average internet user, usability of the system will be given prime importance. An average user will be able to post, search and create blood donations request within 5-steps. Moreover, use of consistency standards will help understanding the content easily but help and documentation will also be available. |
| 14 | For the early stages of the software launch portability of the software will be limited to only Android, IOS apps and chrome web-app. Later, the support for other web browsers like Safari and Firefox will be available. |
| 15 | Chat between the blood donor and blood receiver will be encrypted so that the users feel safe while sharing contact information in the chat. |

# System Architecture

Brief introduction of this chapter in a paragraph highlighting the content

## Architecture Diagram



## Architecture Description

**Presentation Layer:**

Presentation layer is the top most layer of our layered architectural pattern. In our case, it consists of our blood donation web/app user interface (UI). So, our Users (receiver, donor, NGO, admin) will directly interact or communicate with this layer. But user interactions will be handled by business and persistence layer. For example, in case of, donor wants to connect to NGO, it will interact with our UI, and this interaction will be forwarded to Business layer to logically deal with donor. Afterwards, business layer will interact with persistence layer in order to read/write data to database.

We will implement this layer in our project using react/react native.

**Business Layer:**

Business layer is the second layer of our layered architectural pattern. In our case, it contains some basic functionalities such as feedback management and donor management etc.

We implement the logic of system in this layer. For example, if donor wants to join NGO, it will be selected on basis of its health reports. So, logic of donor selection will be implemented in this layer.

**Persistence Layer:**

Persistence layer is the third layer of our layered architectural pattern. In our case, it contains user repositories and data access objects etc. This layer will access data from our database using repositories and data access objects and will pass it to business layer above it. For example, if donor wants to view his profile, it will interact with UI, this interaction will be forwarded to business layer and then to persistence layer, persistence layer will then read data of donor from database. Persistence layer will send this read data to business layer and from there it will be sent to presentation layer, where data will be shown to user.

**Database:**

Database is the part of layered architectural pattern, where all user data is stored. Persistence layer access data from database using data access objects and repositories to handle user interactions.

We will use PostgreSQL as our Database for our project.

## Justification of the Architecture

We will use layered architectural pattern in our project, pros and cons of using this architecture in our system are discussed below.

**Pros**:

* **Ease of development:** Every team member will not need full knowledge of project stack. Every member can work on the skillset, it is comfortable with. For example, if some member is good at react native, it can work on presentation layer building front end. Similarly, if a team member is good at backend, it can work on persistence and database layer.
* **Maintainability:** since layers are separated by their functionalities. So, our code will be modular and changing code will be easy and would not impact other layers.
* **Ease of testing:** we can easily test each layer by making mock components at each layer.
* **Reusable code:** we will be able to reuse code with in same layer. For example, we can reuse react components within same layer.

**Cons:**

* **Performance issues:** Since each request needs to go through multiple layers in order to fulfill user requirement. So, Performance issues may arise, if application gets complicated.
* We may be **unable to achieve clean separation of layers** as system get complicated, which is drawback of layered architecture.
* May be Harder for our team to ensure **consistency** with layered architecture.

**Justification:**

1. We will be using layered architecture in our project, since there is possibility that some members may not have full knowledge and expertise in project stack, which can affect the overall progress of our project. As layered architecture separates the layers i.e., presentation layer, business layer and persistence layer. So, members can work on the layers they are comfortable with depending upon their skillset. For example, if some team member is good at backend, he can work on persistence and database layer. Also, team member can work in different layers, if they are comfortable with overall project stack.
2. We can check the **progress of our project**, by looking at how much has been implemented in each layer.
3. As in layered architecture, code will be modular and maintainable in each layer, so making changes to any layer of code will not be an issue and will not impact other layers and overall system. For example, if we want to implement full functionality, we will start by implementing smaller functions and then merging them would not be an issue since code is **modular and maintainable**. Also, we can add new features without compromising existing ones.
4. When we merge smaller functions into bigger functionalities, there are chances **of performance issues**. But as code will be modular in layered architecture, so we can make changes to smaller functions to reduce overhead, so that we do not run into performance issues when we merge them later on.
5. As layered architecture have separate database, so we can make database secure using different built in **security measures** such as middleware and SQL injection measure to handle security of system.

## Tools and Technologies

**Tech stack:** React Js, React Native, NodeJs, PostgreSQL

**React Native:**

For **frontend** we will be using React Native. React native is an open source UI software framework created by Facebook. It allows developers to develop applications for Android, iOs and web. Difference between react Js and react native is that, react Js is library and react native is a framework which use native components to develop responsive mobile applications.

Advantages of using react native for frontend development are:

* + It is really famous and backed by a very large community which increases the availability of solutions we might face while development.
  + Its uses Modular and intuitive architecture which allows developers to create independent reusable component. This approach contributes in clean and reusable code, enhanced flexibility and saving time and money.
  + It allows developers to create an app with cross platform support.

**Version:** we will be using latest react native version 0.66

**Express.js:**

For **backend** we will be using ExpressJS. Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.Ruby on Rails or Rails is a server-side web application framework written in Ruby. Ruby is highly used for the social app development.

Advantages of using ExpressJs as back end development are:

* + It makes the app scalable. It is easy to learn and also decreases the developer cost to maintain the app.
  + Express.JS is supported with the Google V8 engine with the help of which you would be able to get higher performance without any lag or error in the processing.
  + Express.js supports the caching feature, and the advantage of the catch is that you would not have to re-execute the codes again and again. Moreover, it will help web pages to load faster than ever.

**Version:** we will be using current stable version  4.17.1.

**MongoDB:**

We will be using **MongoDB** as **Database. MongoDB** is a powerful, open source object-relational database system. It is really mature database which provides following advantages:

* + High speed
  + Secure
  + reliability

**Version:** we will be using current version 13.3.

**Styling and Responsiveness:**

To enhance the usability of the app, responsiveness is really important. Bootstrap is CSS framework which provides grid layout for easier incorporation of responsiveness in the app. For better styling and responsiveness we will be using Bootstrap 5 along with HTML5 and CSS(2.1) .

**Design:**

For designing the screens and prototypes we will be using **Figma** version 102.9.

**Testing:**

For testing the backend API’s we will be using **Postman** (8.

**Coding Environment:**

We will be using **vsCode**(v.1.61) as Integrated Development Environment.

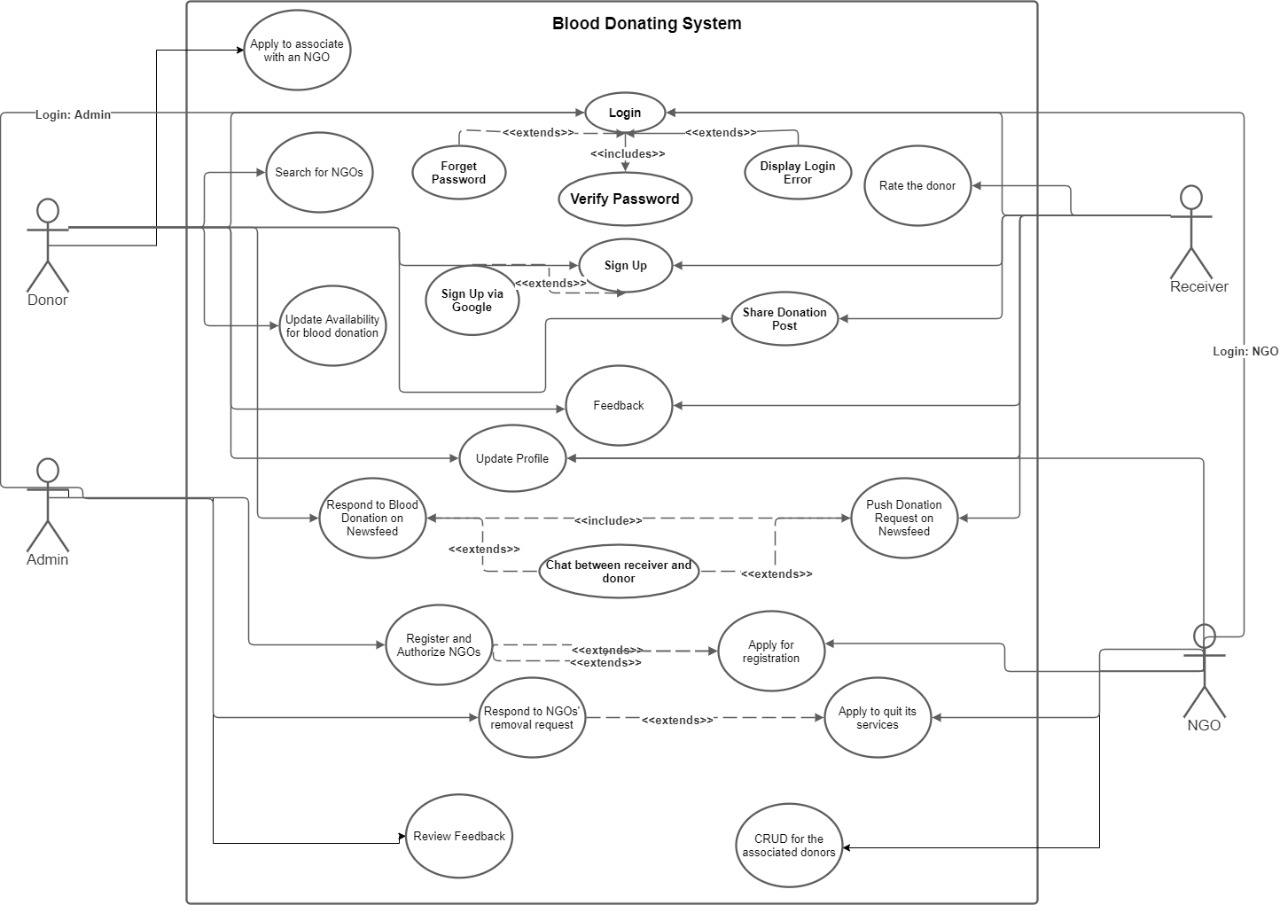
**Version Control:**

To track progress and version control, we will be using **GitHub.**

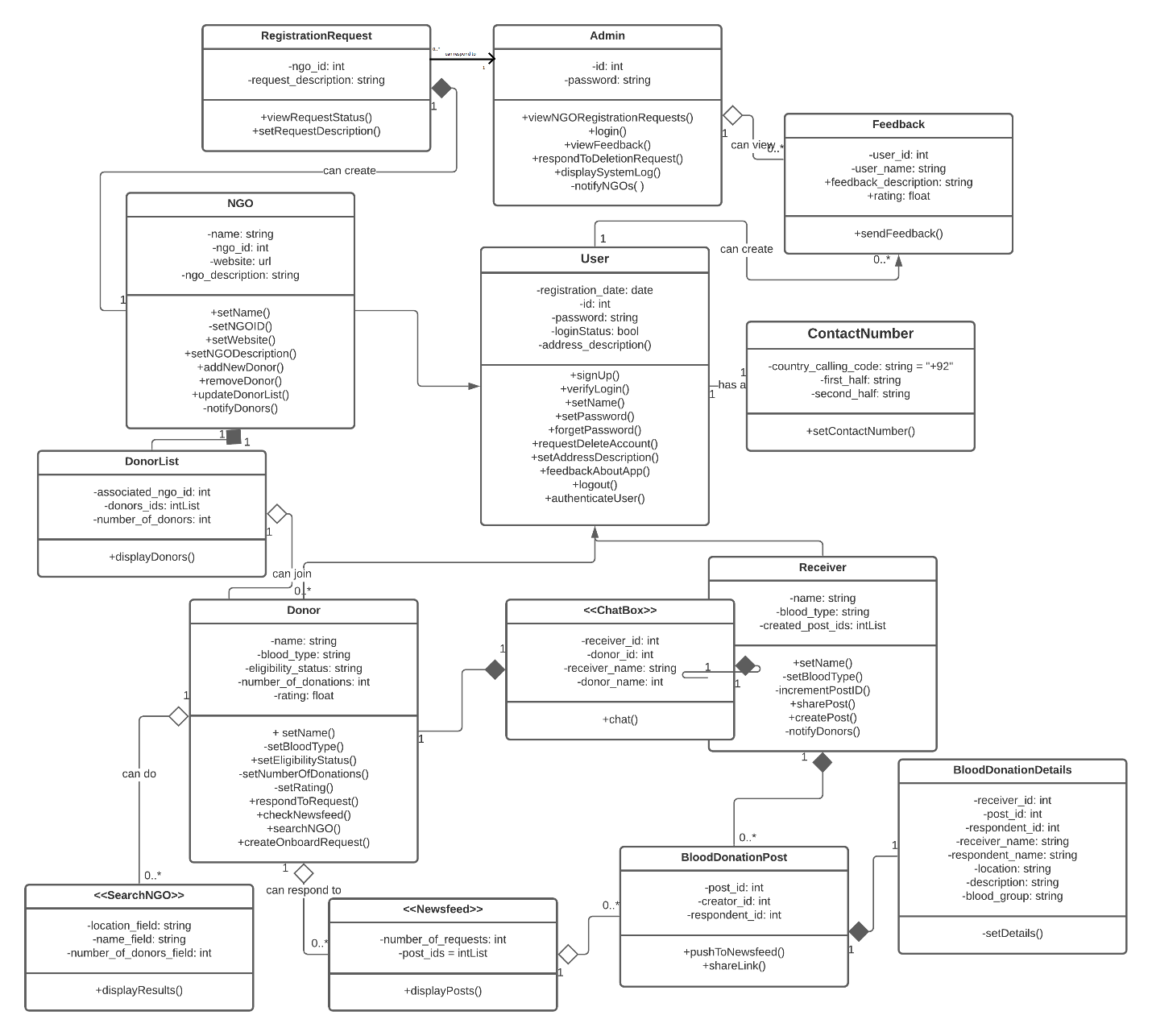
# Requirements Specifications

Brief introduction of this chapter in a paragraph highlighting the content

## Use Cases

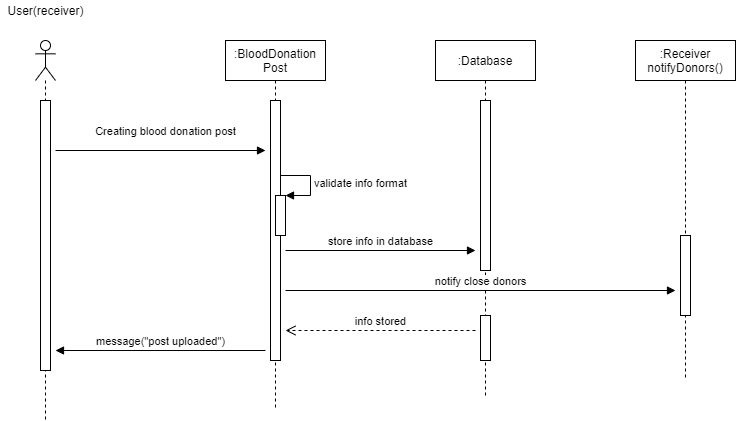


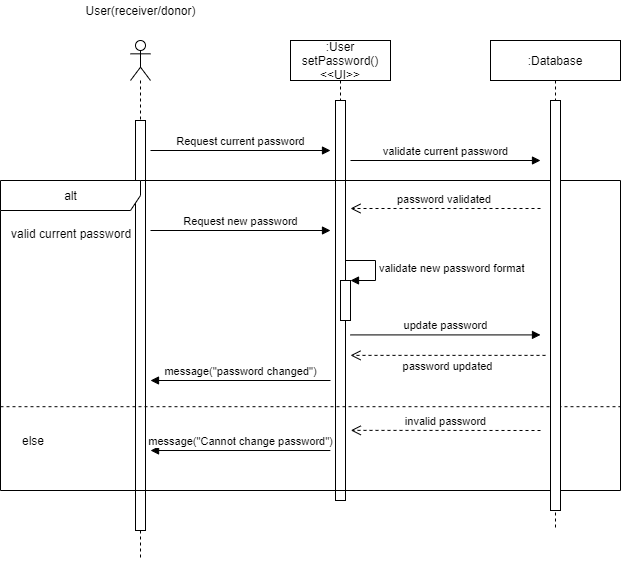
## Class Diagram



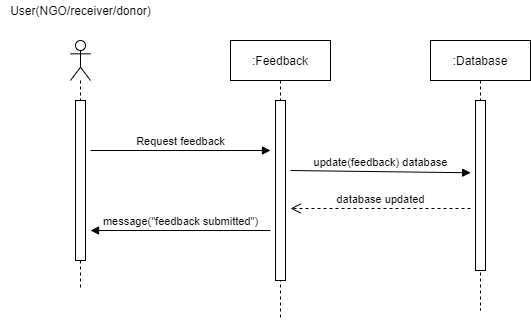
## Sequence Diagrams

1. **Create Blood Donation Request**

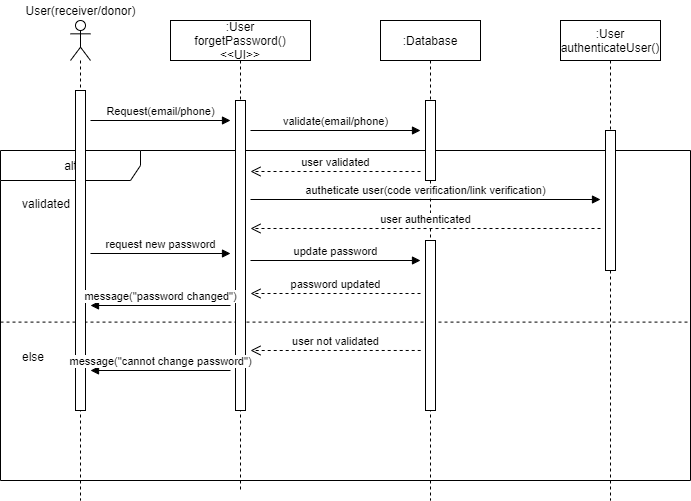
****

1. **Change password**

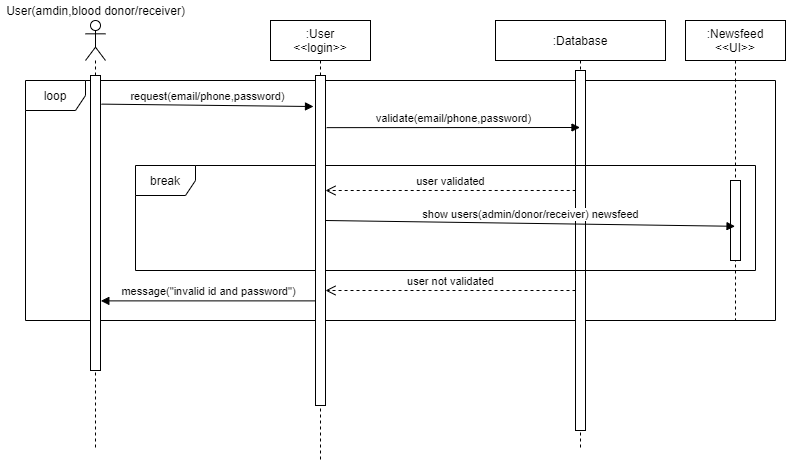
**3) Feedback**

****

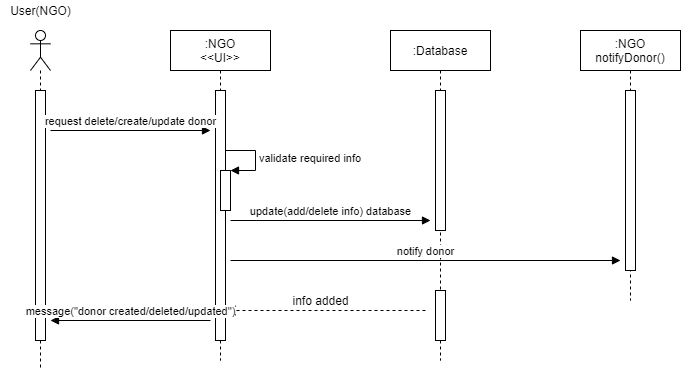
**4) Forget Password**

****

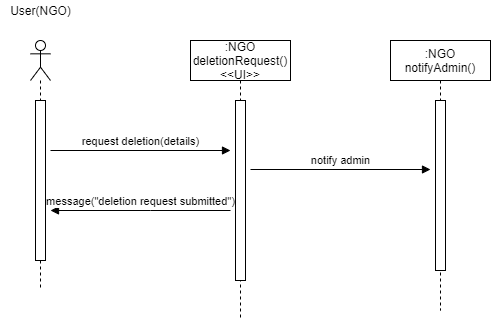
**5) Login**

****

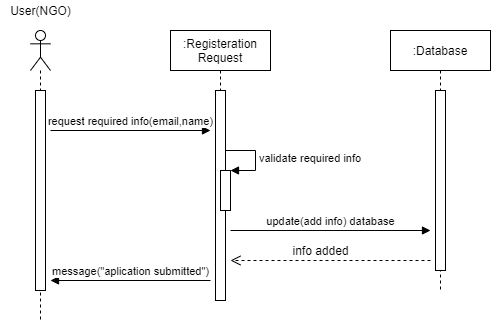
**6) Create, Update, Delete operations for NGO**

****

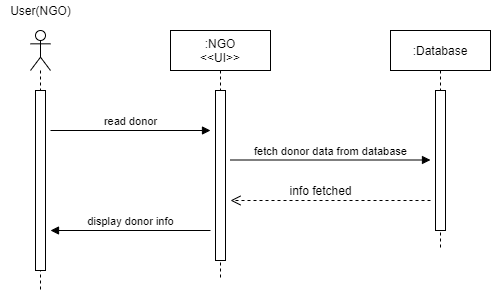
**7) NGO Account Termination Request**

****

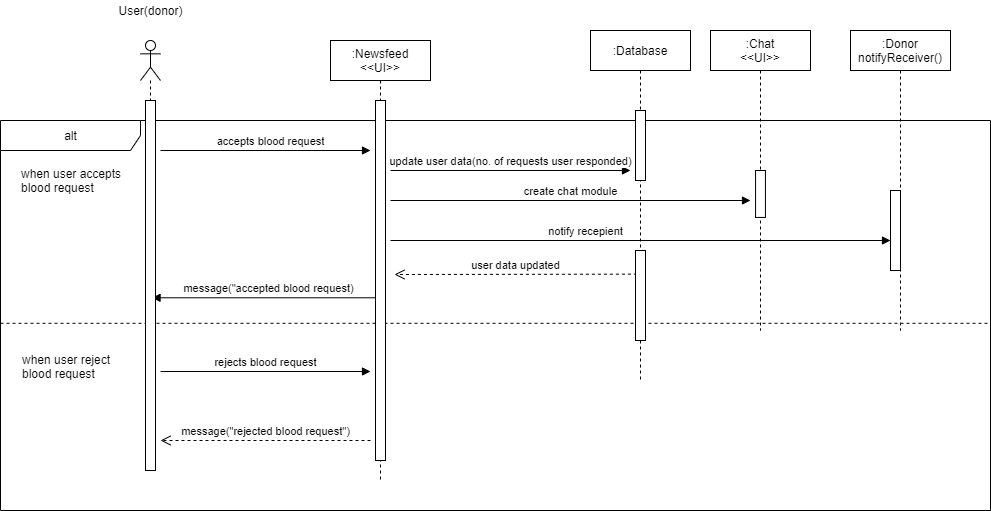
**8) NGO Registration Request**

****

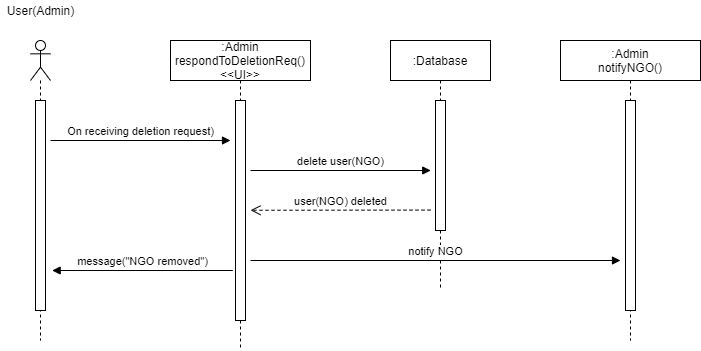
**9) Read Operation for NGO**

****

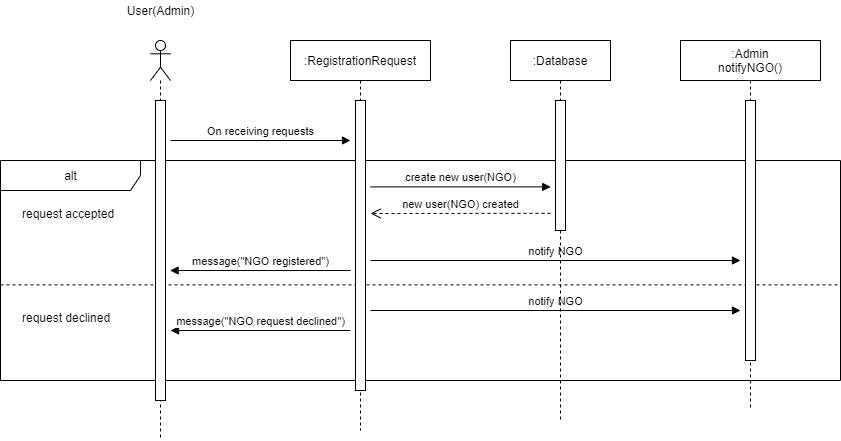
**10) Respond to Blood Request**

****

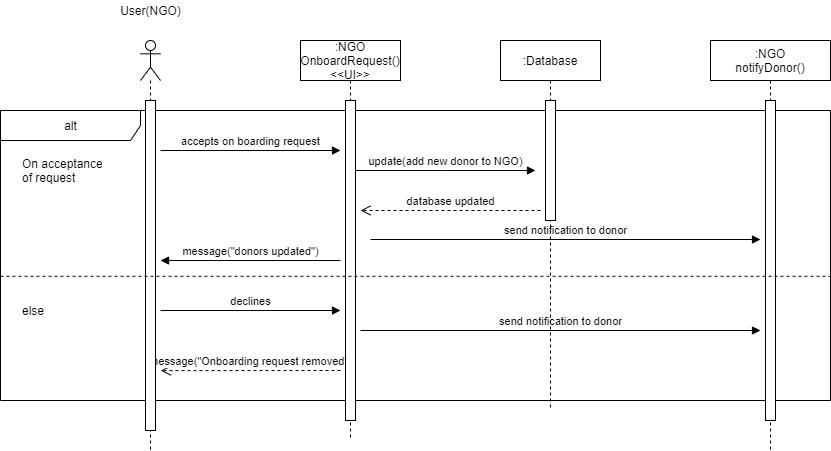
**11) Respond to NGO’s Deletion Request**

****

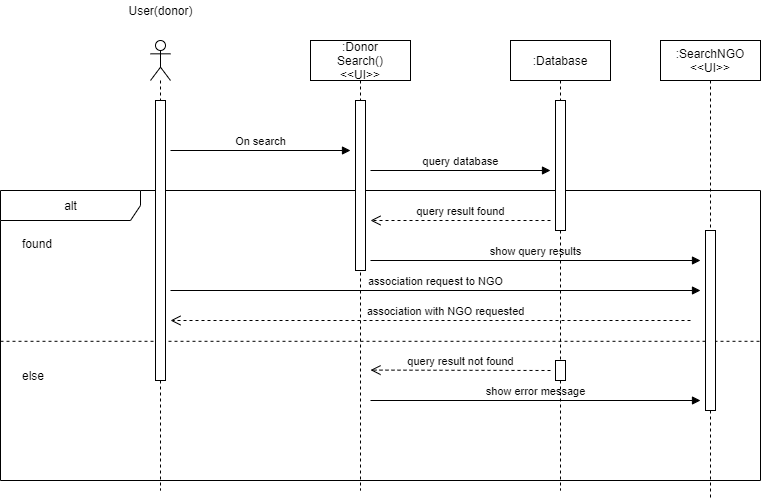
**12) Respond to NGO’s Registration Request**

****

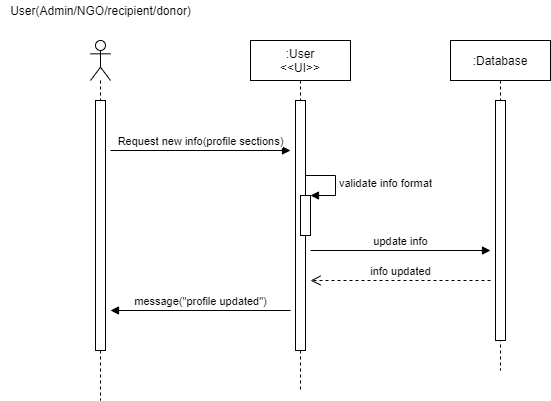
**13) Respond to Donor’s Onboarding Request**

****

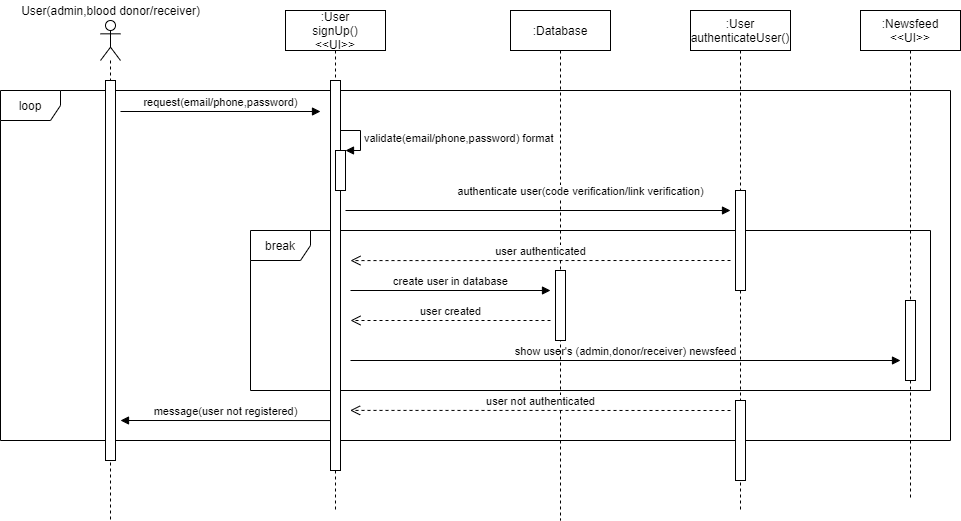
**14) Search NGOs**

****

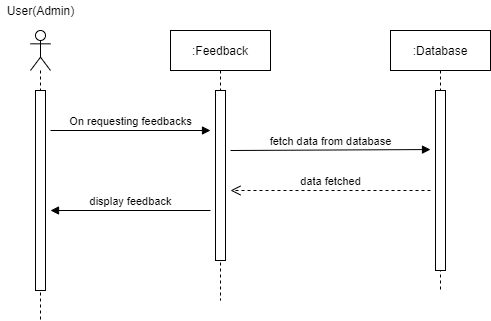
**15) Update Profile**

****

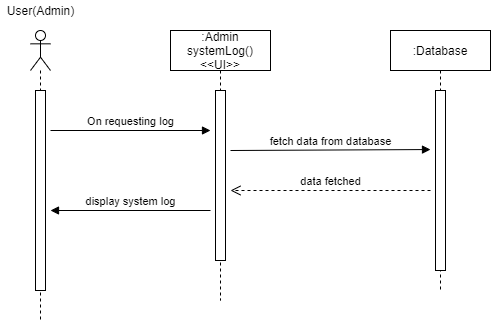
**16) Sign Up**

****

**17) View Feedback**

****

**18) View System Log**

****

# Software Development Methodology and Plan

Brief introduction of this chapter in a paragraph highlighting the content

## Software Process Selection

|  |
| --- |
| **Waterfall** |
| Waterfall is a traditional method of software development in which each phase depends on the deliverable of the previous phase. In other words, one phase is developed completely and then the next phase is developed. It is consist of following steps:   1. Requirement Analysis 2. System design 3. Implementation 4. Testing 5. Deployment 6. Maintenance   Sequence of these steps is strictly followed. |
| **PROS:**   * Waterfall focuses on the clear, well defined set of steps which makes its structure easy to understand and use. * Since all the requirements are defined beforehand, it is easy to manage. Time scales and deadlines can be set up easily because of the rigidity of the model. * Phases are completed one at a time which allows developers to focus more on specific tasks. * Waterfall focuses on the end goal from the beginning. Which eliminates the risk of loosing track and it also keeps the team organized. * Information is completely transferred from one phase to other in the form of documents which allows the incorporation of new developers and teams. In other words, it is possible that different phases are done by different teams. |
| **CONS:**   * Major disadvantage of using waterfall model is that it makes changes very difficult. Waterfall is based entirely on following fixes steps so it leaves almost no room for any unexpected changes. * No working software is produced until late during the life cycle. Because of this, there is almost no involvement of client and end user. * No involvement of the client and end user increases the risk and uncertainty. No feedback is available from client or end user until late in the life cycle. And at the end if client or end user dislikes the software and wants to make any changes these changes can hardly be accommodated. * Adjusting scope during life cycle can end the project. * It is difficult to measure progress within the phases. |

|  |
| --- |
| **Agile** |
| The agile methodology is an iterative approach to project management and software development. In agile, project is divided into several phases which are developed and improved continuously with the collaboration of the stakeholders. Agile contains following steps.   * Planning * Analyzing * Architecture design * Coding * Unit testing * Acceptance testing * Delivery * Feedback   In agile there can be many iteration in one specific phase of the life cycle. Every iteration solves some existing problems or incorporated the requirement changes. |
| **PROS:**   * The agile process allows to deliver working software quickly. It might contain a lot of bugs but the feedback of the stakeholders can be taken which helps a lot is improvement of the software. * Due to iterative approach, it is very easy for the project to solve problems and adapt itself to the changes requirements. * Agile methodology requires a lot of collaboration back and forth. The continuous feedback of client and customer and teams creates a positive environment which enhances the quality of project. * Since feedbacks are continuously being exchanged. It is very easy to detect problems in the project. * Risk and uncertainty is very low. |
| **CONS**:   * Lack of documentation is one main disadvantage of using agile methodology. Due to ever changing requirements of the project, document for one phase can become obsolete due to minor or major changes * Flexibility of the agile makes it very hard to manage. Time scales and deadlines for specifics phases are also very hard to set up. * In contrast to the waterfall model it is easy to lose track in agile and team can face difficulties to organize itself. * Due to continuous change in the requirement there is a lack or clarity in the project which leads to many problems. |

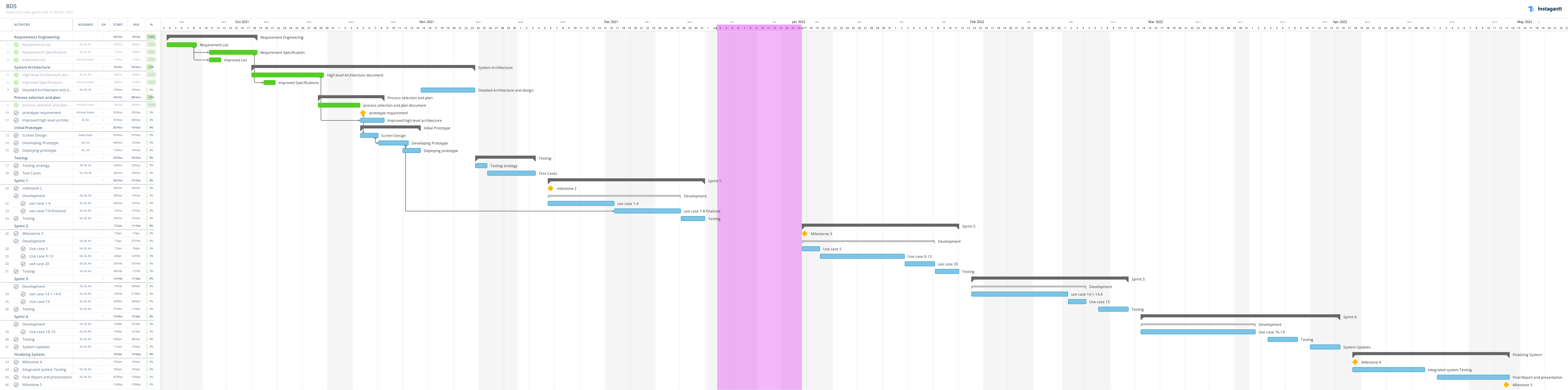
## Gantt Chart

Gantt chart link: <https://app.instagantt.com/shared/s/zdBlHQMXawLPcQLyhEx9/latest>

Please visit the link above to view gantt ch

|  |
| --- |
| **Process Selection** |
| We will be using **Agile** methodology for project management and software development.  **Reason:**  Over methodology that we are using in this course is agile where we submit  deliverable and then receive feedback. Then as a whole team we incorporate the changes suggested in the feedback to improve quality of the deliverable. So, agile methodology is well aligned with the course because in upcoming we will be create a working prototype of the system and the improve it sequentially which is basically agile. Moreover, it will help us complete project quickly. Since we are developing a social app which require a lot of feedback from end users so agile is best suitable for us because it will be easy to incorporate changes. Agile is becoming more popular day by day so implementing agile on our project will also give us experience of industry standards. There will be less pressure of the deadlines considering the improvements that we can make in the next iteration.  We will use standard practices to implement agile methodology. We will use Trello for task management. We will divide each phase into small tasks and assign the tasks to different team member. We will also have scrum meeting to evaluate the progress of the project and define the sprints. |

art in detail.

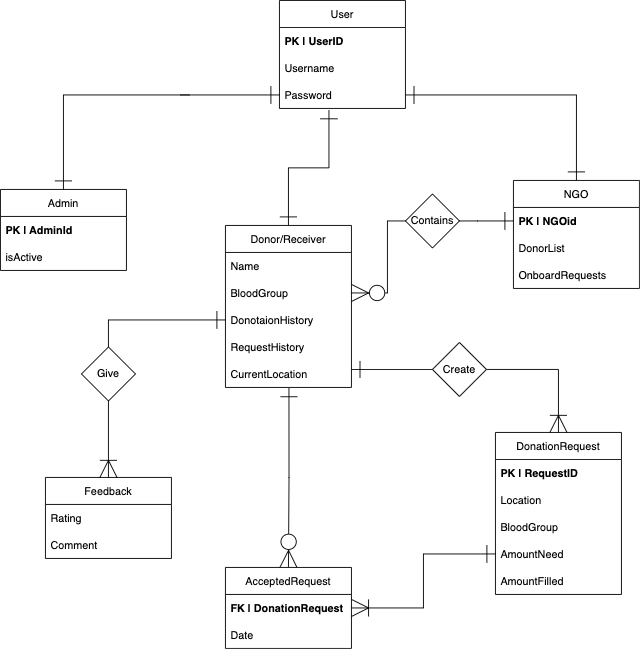


Caption

# Database Design and Web Services

Brief introduction of this chapter in a paragraph

## Database Design



# Description:

We have a general user class which contains username and password of the user. In the context of data model User can be of three types.

1. Blood receiver and donor

2. NGO which have one to many relationship with blood donor and receiver.

3. Admin which is the most privileged user of the system

**Data Types:**

username | varChar

password | varChar (hashed)

Name | varChar

isActive | bool

BloodGroup | char

DonationHistory | SET(val1, val2, val3,…)

RequestHistory | SET(val1, val2, val3,…)

CurrentLocation | varChar

DonorList | SET(val1, val2, val3,…)

OnboardRequest| SET(val1, val2, val3,…)

Rating | INT

Comment | varChar

Location | varChar

AmountNeeded | INT

AmountFilled | INT

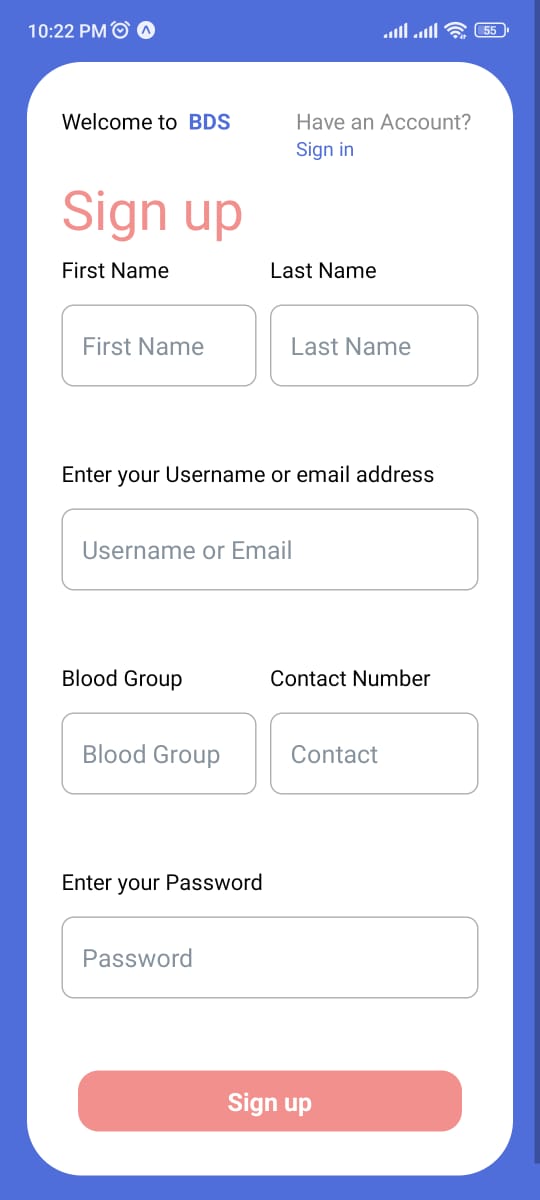
## API Specification

**Geocoding Api:**

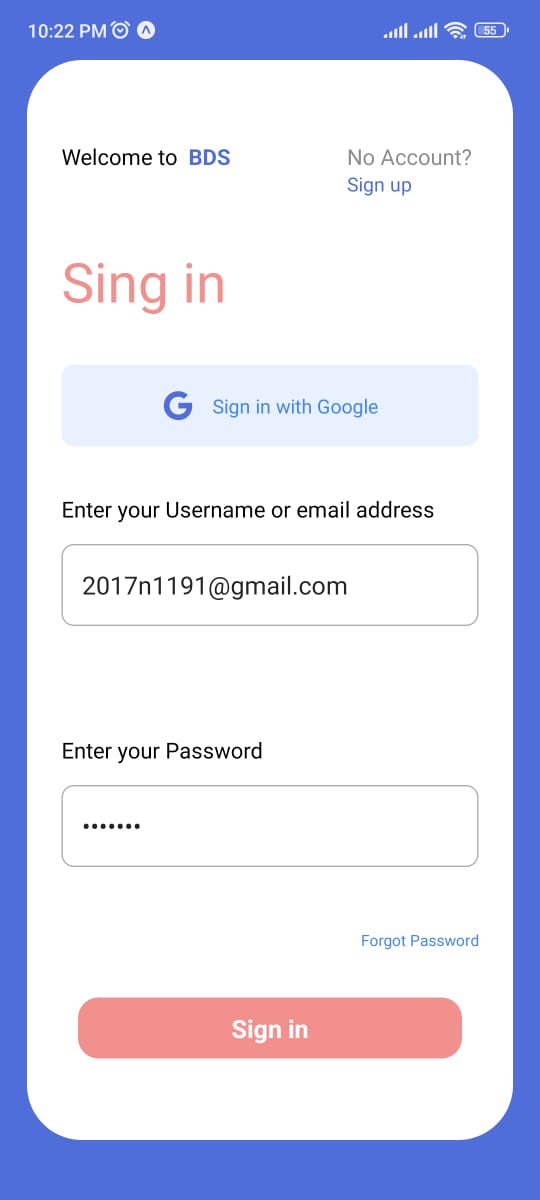
This is the gps api provided by the google for the location configuration in the in-app google map.

# System User Interface

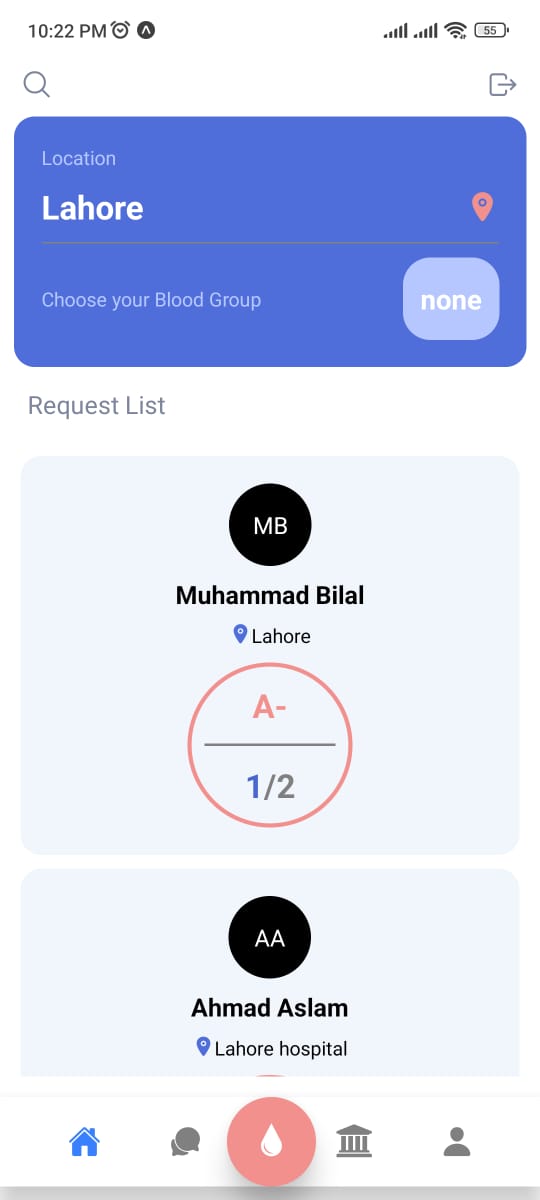
User can create his account by providing all the required credentials. If user already have account then he can go to sign in screen by clicking link on the top right corner.



User sign in to his account by entering username and password. If user does not have account then he can go to sign up screen from the “sign up” link.

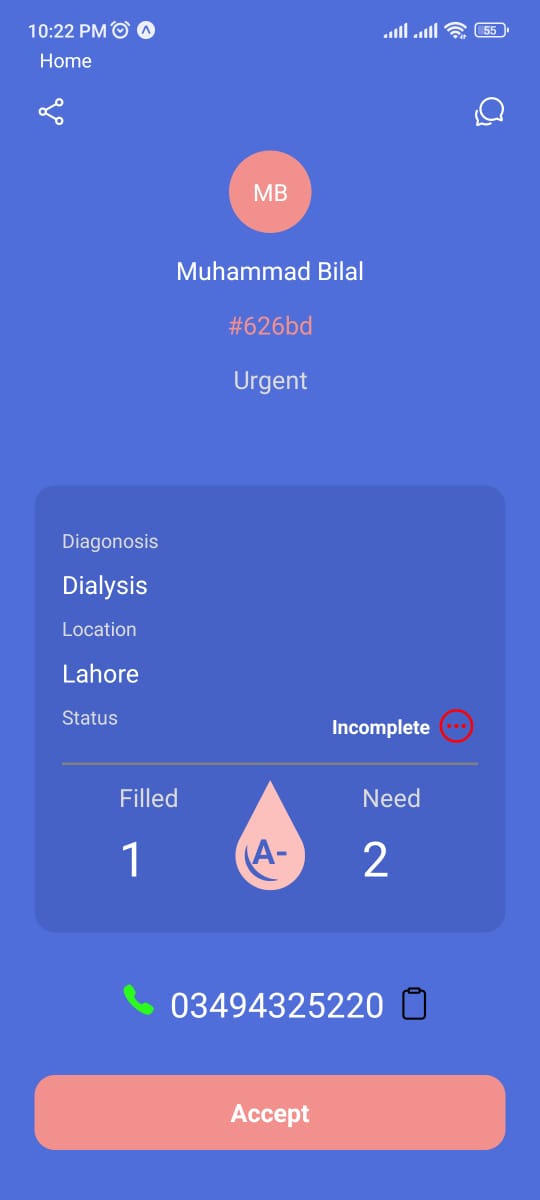


This is the page where user lands on after successful login. Top right corner is the log out button that redirects user to the login page. “Choose your blood group” is the filter which allows user to filter blood donation requests based on specific blood group. By default filter value is none so all the blood donation request are the listed.  
Menu bar contains buttons for the chat screen, Ngo screen and profile screen along with landing screen. Details of these screen are given ahead.



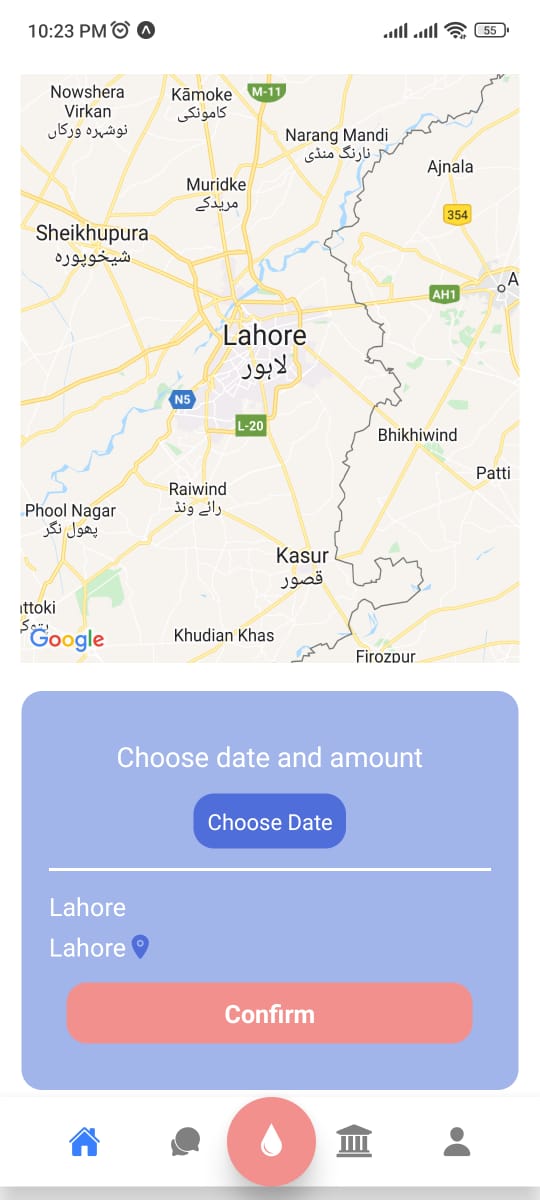
This screen opens when user click on any card in the request list. Screen displays complete detail of the request user clicked on. On top left corner home button redirects user back to home page. Share icon initiates the share functionality. Similarly chat icon initiates the chat with the person that posted the request.

Accept button redirects user to the confirmation screen.



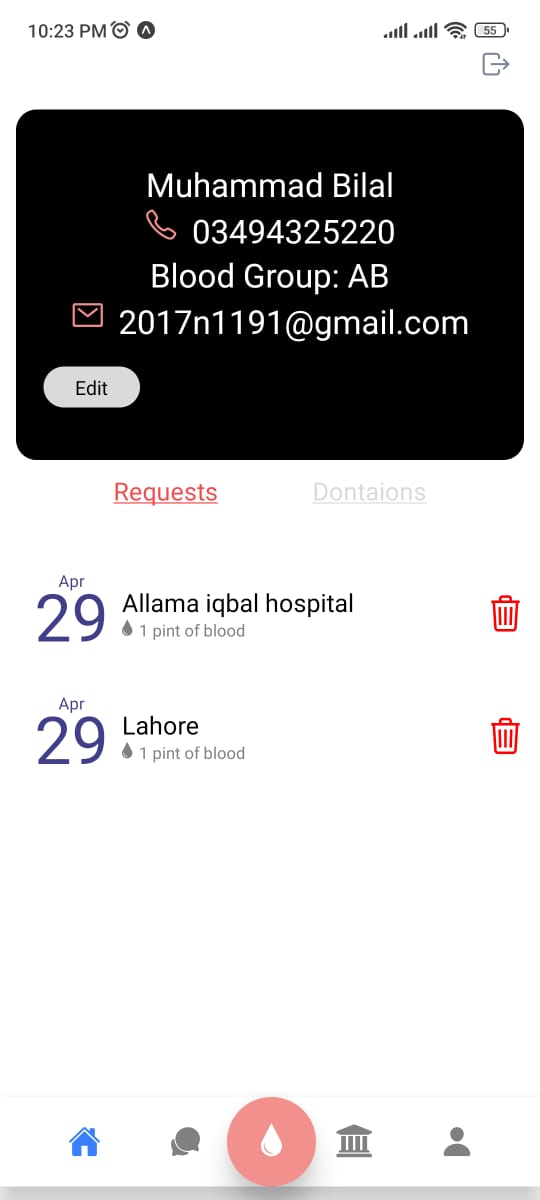
This is the confirmation screen. Map on the top section shows the location of the request. This the location that blood receiver has entered while making the request (More on this later).

Calendar below allows the blood donor to select the date on which he will be available for donation.

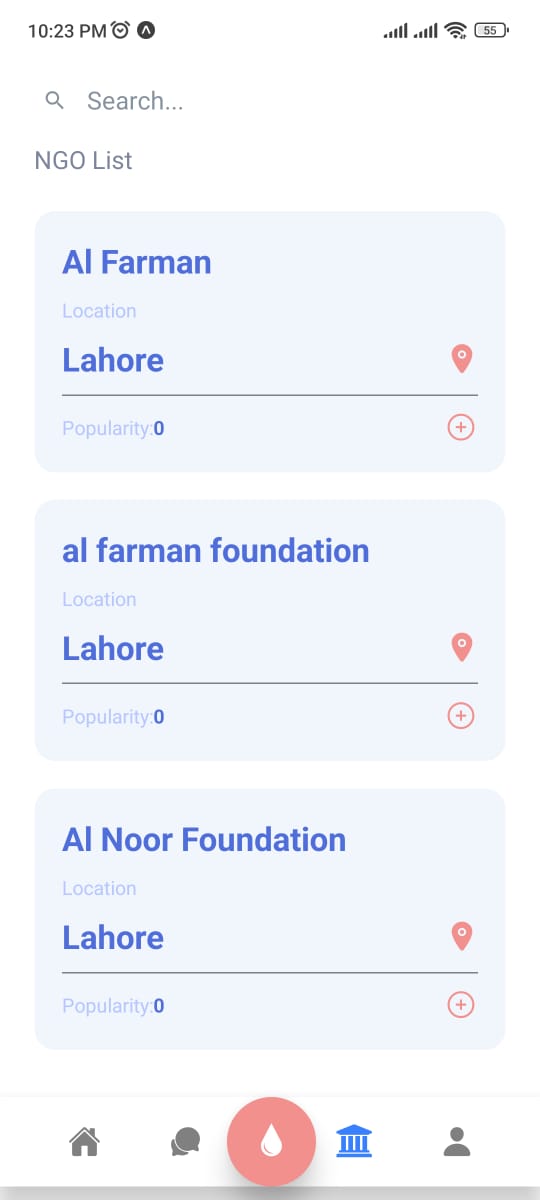


After clicking the confirm button user is redirected to the profile page where the confirmed donation is added to the donations section.

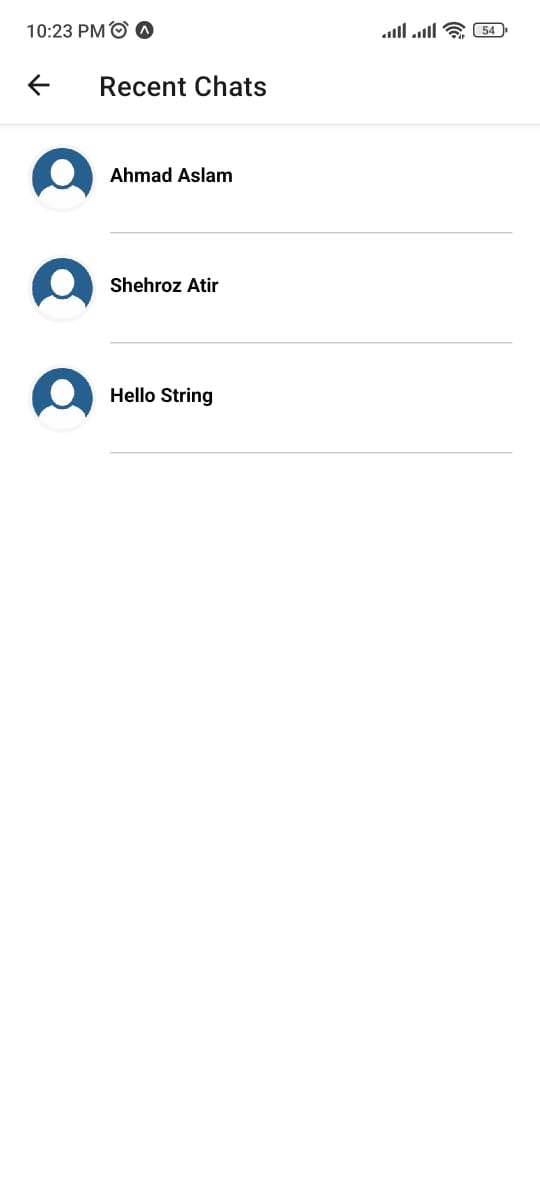
Profile screen shows the user details. In. Requests section all the requests that user has posted are visible with the delete icon which allows user to delete the request.



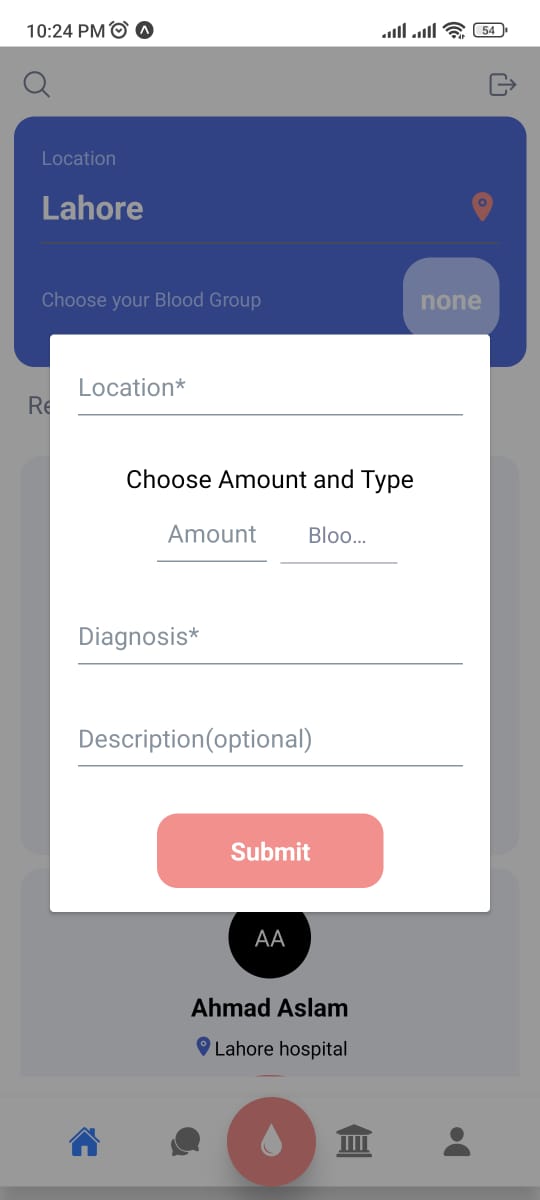
This page open when user clicks on the ngo icon in the menu bar or clicks on the search icon in the landing screen.   
Each card on this screen contains the detail of a specific ngo which is coming from ngo portal.



Chat screen gets open when chat icon on the menu bar is clicked. All previous chats are available here and user can have chat with any person simply by clicking on name.



On clicking the big red button at the centre of the menu bar this overlay form open. This allows user to post blood donation request. After filling all the required fields, user clicks on the submit button and request is posted on the landing screen. Now request will be visible to all other users.



# Project Security

It’s the identification of potential risks, and implementation of strategies which will protect or preserve the confidentiality, integrity, and availability of our project resources.

## Project Threats

* **Broken Authentication**

Authentication and user authorization issues emerge due to the insecure connection between the web client and the server. By tampering with authentication and authorization protocols, hackers can compromise user data.

* **Server-Side rendering**

We have employed server-side rendering. There are several benefits to server-side rendering, and it boosts app performance, speeds up webpage loading, and assures consistent SEO performance. However, this style of page rendering might pose certain security risks.

* **SQL injection**

This kind of attack targets the app's databases. Attackers can inject malicious SQL code into a database and get access to the data contained inside. They can get access to database data.

## Potential Losses

* + 1. Hackers can compromise user account data, passwords, session tokens, and other things
    2. Hackers can modify, remove, or create new records if they have admin privileges

## Security Controls

* + **Broken Authentication**
    1. Evaluate whether the domain "WWW" header has a valid attribute that aids in avoiding mismatches between user IDs and passwords.
    2. Use suitable authentication techniques, such as ensuring that a web app responds with a 401 status error page if authentication fails.
    3. Incorporate cloud-native authentication, such as Google Cloud Identity Platform or Azure Active Directory.
    4. Implement password strength and weakness tests.
  + **Server-Side rendering**
    1. Perform code reviews regularly and double-check the data in JSON.stringify().
    2. To avoid rendering JSON, use the serialize-JavaScript NPM package.
* **SQL injection**
  1. Implement whitelists to filter all types of inputs.
  2. When a web app may only utilize one statement (SELECT, UPDATE, INSERT, or DELETE) for particular actions, apply the concept of least privilege to all accounts.
  3. Assign database roles to various accounts.
  4. Employ vulnerability scanners.
  5. Validate all API methods following their API schemas.

## Static and Dynamic Security Scanning Tools

We have used **ImmuniWeb** for static and dynamic scanning that offers a unique combination of mobile app and its backend testing in a consolidated offer. It comprehensibly covers Mobile [OWASP Top 10](https://www.softwaretestinghelp.com/owasp-top-10-security-vulnerabilities/) for the mobile app. ImmuniWeb offers a free online mobile scanner for developers and SMEs, to detect privacy issues, verify application permissions and run holistic [DAST/SAST](https://www.softwaretestinghelp.com/differences-between-sast-dast-iast-and-rasp/) testing for OWASP Mobile Top 10.

# Risk Management

## Potential Risks and Mitigation Strategies

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | **Servers Unavailability risk:**  In case of high traffic, servers can become unavailable and user can get delayed response. | In such situation, owner will be suggested to move to high performing servers to handle heavy traffic, and make system available all time. |
|  | **Operational Risk:**  In case of an issue in working application, it can crash. | Team will identify problem and will work on solving it. It will make application work again within time constraint. |
|  | **Reusable component risk:**  There is risk of faults in reusable components of application, which can make certain application features unavailable. | Our team will identify the fault in component. It will work on resolving it and making it functional for future. |
|  | **Team members Unavailability Risk:**  There is possibility of team members unavailability, that can disturb the normal flow of activities done by team. | Available team will reorganize the tasks, and will do missing work. |
|  | **Requirement Change Risk:**  Change in requirement can happen in future, depending upon customer need. | Team will be notified of change in requirements. It will re-define time constraint for updated requirements.  Also, owner will be notified of a cost change. |
|  | **Team Skill Risk:**  In case of team skillset, there are chances that some team members will not be familiar with tool and technologies project requires. | Team will hold tutorials to get familiarize all team members with the tools and technologies project requires. |
|  | **Security Risk:**  There is risk of data breaching of application users. | Team will identify security issues and will suggest owner to move to more secure system. |
|  | **Performance Risk:**  There is the risk of low performance of application. | In such situation, team will make amendments in tools and technologies it used, for high performance of application. |
|  | **Database risk:**  There is risk of low database performance i.e., it cannot perform expected number of transactions. | In such situation, team will identify the problem and owner will be suggested to move to high performing database. |
|  | **System Feature Risk:**  There is risk that some system features stop working due to unexpected programmatic issue. | Team will get feedback from end users and will resolve the issues. |

# Deployment Guidelines

Github repository: <https://github.com/ahmadaslam1999/SPROJ>

**Backend:**

Our backend is hosted on heroku. To perform deployment we just need to push our latest working code to the main repository and click deploy button on heroku app.

Link: **<https://dashboard.heroku.com/apps/sproj-bds>**

Username: **[ahmadaslam1999@gmail.com](mailto:ahmadaslam1999@gmail.com)**

Password: D4x5@X=ywJcDF/F

**Frontend:**

Our Frontend is hosted on expo client. After making changes all you need to is pres publish button on the expo server. Everything else will be handled by expo.

Link: **<https://exp.host/@ahmadaslam1999/BDS>**

# Conclusion

## Summary

The main idea of the all this work was to understand the process. At the very beginning we had two options. Either try to solve a very complex problem in which we would most probably fail due to limited knowledge and experience. Or try to understand the process and the mindset through which problems in th IT world are solved. We decided to with the second approach and we learned a lot about processes.

## Challenges

Keeping up with all the team members was certainly a difficult task. During the time span of about 1 year every one is expected to have some hard days. Most important thing is that we stand with each other and motivated each other. But unfortunately, we could not do that. We faced many problems due to lack of communication.

## Future

There is a lot of room for improvement in this app. We will first add missing functionalities and then go for fixing the existing bugs. More functionality can be added in the ngo portal part.

# Review checklist

Before submission of this report, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Chapter/Section Name** | **Reviewer Name(s)** |
| All sections | Ahmad Aslam |
| All sections | Muhammad Bilal |

# References