Software Design Specification Document
(CS360)

BloodLink

Group Number: 21

Muhammad Ahmad Mahmood
Daim Armaghan
Muhammad Farrukh
Muhammad Hassnain
Sharjeel Ahmed

Course: Software Engineering CS360

Instructor: Suleman Shahid

University: Lahore University of Management Sciences (LUMS)

Version: 1.0
Date: (09/03/2022)
Number of hours spent on this document: 110

Table of Contents

1 Change Log

1.1 Project Scope

This project aims to design and develop a working mobile app that is compatible with android and iOS systems.

Work on this project has been started. The design phase is complete, and development will start from the third week of March. Five developers will work on the development and complete the development by the mid of April. Testing will be done right afterward, and the final product will be released at the end of April.

1.2 Change log

A major change was made in the development architecture of application. The BloodLink app was previously decided to be a web application, however, after consultation with Sir Mustansir, we found that some critical features of our app are more feasible to implement in Mobile architecture. We had a detailed meeting on this, and consultation with our clients, we made a joint decision to switch to mobile application. This application will be developed on a cross-platform i.e., flutter, and will be accessible on both android and IOS devices.

2 Introduction

2.1 Document Purpose

The purpose of this document is to help us build and maintain an application for the Bloodlink. The app being developed will allow initially people from the LUMS community to request and donate blood and eventually all over Pakistan. This document will serve as a guideline for developers to develop the application. It is also a contract between the client (Bloodlink) and us the developers.

This document will help us better identify our system architecture, development requirements. It's quite crucial for developers to understand the architectural division that will directly help in the developmental phase. The document first gives the application scope and detailed overview. It then highlights the key constraints along with architectural designs. The main focus of the document is to demonstrate the architectural division and flows using multiple UML diagrams. These UML diagrams will help viewers better understand the structural flows and relationships visually. The document also deeply discusses the User Interface designs, the key UI elements, Material design principles used, and accessibility measures incorporated.

2.2 Product Scope

2.2.1 Scope:

The intended user group of this software is the people across Pakistan who are in need of blood or want to donate it. It is expected that initially, the users of this software will mainly comprise students, faculty, and staff members of LUMS. Depending on the marketing strategy applied by Bloodlink (the client) and the success of this software, the user group is eventually expected to expand to the general population of Pakistan.

2.2.2 Objectives:

1. Automate blood request management.
2. Decrease time delays in finding a donor.
3. Eliminate the potential of requests getting lost due to human error.
4. Expand the scope of Bloodlink's services.

5. Find better donors based on location and availability.

Benefits:

1. The amount of time Bloodlink members serve in blood request management will be saved.
2. Blood requests will be managed more efficiently and reliably.
3. More people will be able to avail Bloodlink's services.
4. The time required for donors to travel to the hospitals where blood is required will be reduced.

2.3 Intended Audience and Document Overview

BloodLink-LCSS currently performs activities manually, which causes delays in request processing. These activities include managing requests, verifying cases and arranging donors, etc. Since these processes are dealt with manually, there are long processing times, more time, and effort by the Bloodlink team. According to BloodLink Director, "The biggest issue we are facing is handling urgent requests. Many times, we could not handle them as urgently as required, due to personal emergencies. Also, we are often overwhelmed with a huge volume of requests.".So, there's a gap to mitigate these delays by automating all BloodLink activities through a software solution. This project aims to automate the current manual process of the Bloodlink. It is also being developed as part of a course project therefore the intended audience not only includes our team, the bloodlink team, the people they will hire in the future to maintain it but also the instructor and teaching assistants of the course titled "Software Engineering." This document contains an overall description, the functionality, target users, their characteristics, and functional and non-functional requirements. Although the document is well organized, with appropriate placement of sections, readers might like to overview the user interface designs prior to architectural design, to get a concrete idea of the end product.

2.4 Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application programming interface. Set of protocols for building and integrating application software.
BloodLink	Our mobile app, named after BloodLink-LCSS, a department within LCSS bridging gaps between donors and requestors.
HIV	Human Immunodeficiency Virus
LCSS	Lums Community Service Society. An internal lums society working for human well-being and welfare.
LDF	Lums Discussion Forum. A Facebook group for LUMS student body.
LUMS	Lahore University of Management Sciences. A Not-for-profit university located in Lahore, Punjab, Pakistan
OTP	One Time Password
NPM	Node JS Package Manager.
SQL	Structured Query Language is used to communicate with a database
DB	Database
I/O	Input/Output

2.5 References and Acknowledgments

"Component Diagram Tutorial." Lucidchart, 2019, www.lucidchart.com/pages/uml-component-diagram.

Visual Paradigm. "What Is Sequence Diagram?" Visual-Paradigm.com, 2019, www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

"Material Design." Material Design, material.io/design.

"Architectural Design: Definition, Types and Examples." Www.mchmaster.com, www.mchmaster.com/news/architectural-design-definition-types-and-examples/.

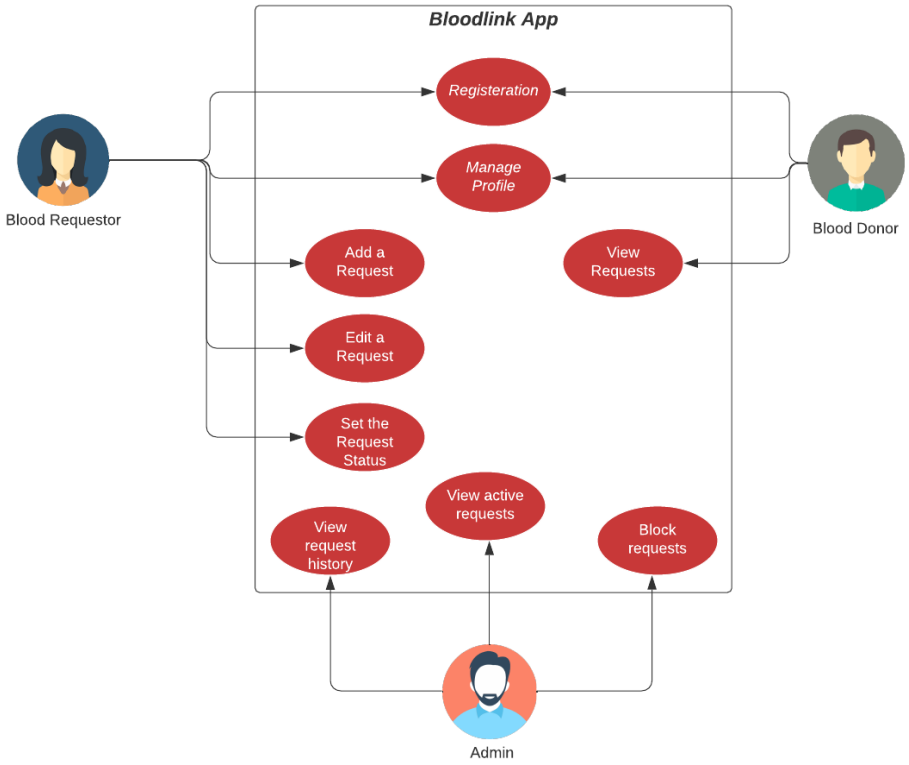
CDC. "Common Eye Disorders." Centers for Disease Control and Prevention, 2019, www.cdc.gov/visionhealth/basics/ced/index.html.

3 Overall Description

3.1 System overview

The Bloodlink app is a new system that aims to automate the manual blood request management of Bloodlink and make it more efficient. Bloodlink is a department of LUMS Community Service Society (LCSS) that receives blood donation requests from people across Pakistan, mainly the students, staff, and faculty of LUMS. Bloodlink app supports four distinct users: Blood requestor, blood donator, admin, and moderator. Each user has a different set of functionalities and privileges. The whole system can be viewed as the cycle of the following main interactions of the system with its environment:

- **Blood Request**: A requester can add blood requests by specifying different details and urgency of the request. On submission of the request, the system notifies the potential donor. The requestor can also set the status of the request as resolved or unresolved depending upon the progress of the request.
- **Accounting Management**: Admin can create moderator accounts, delete those accounts and block other user accounts in case of any discrepancy. Similarly, a requestor and a donor can create and manage their accounts accordingly.
- **Progress Tracking**: Once a request has been made, the admin can track the request whether it has been resolved or not. If the requestor doesn't find any donor such that only 6hours is left in the deadline, the system notifies the admin about the situation to take necessary actions.



3.2 System constraints

3.2.1 No Future Maintenance:

Our development team will graduate in 2023 from LUMS therefore, we will not be able to provide any technical maintenance after that. LCSS may need to develop an IT department with relevant skills to maintain the app in the future.

3.3.2 Impact on design:

Keeping the above constraint in mind, we have introduced the following features in the app:

- The admin can manually add details about the new donors
- Admin can create and delete moderator accounts to provide fewer privileges to maintain the app

- Admin can also initiate blood requests.

3.2.3 Financial Constraint:

LCSS comprises several departments with a limited fixed budget assigned to each department.

Impact on design:

- In case of peak load, there is a possibility of occurring performance issues because of the unavailability of a load balancer(cannot be used in the limited mentioned budget)
- A small hosting machine will be used because of budget constraints.

3.4 Architectural strategies

3.4.1 Frameworks:

Node JS:

Node Js will be used for the server side development of the system. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment. It uses Google Chrome's V8 engine to execute JavaScript code outside a web browser.

Selection Reasoning:

- **Non-Blocking I/O Model:** A Node JS app is run in a single process, without creating a new thread for every request. This is possible because of javascripts event loop functionality. Node JS provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking. It handles thousands of concurrent connections on a single thread without the need of thread concurrency.

- **Node Package Manager (npm):** npm is the package manager for the Node JavaScript platform. It provides you with already built-in packages with different functionalities instead of writing your own code. Some pre-built in plugins(modules) such as for sending an email will be used in our system to focus on the other aspects of development.
- **Performance and Popularity:** Node Js provides very good performance because of its single-threaded non-blocking I/O model. Since it is open-source, it is backed up by a large community and it has become a popular backend framework.

Express JS:

Express is a free and open-source Node js application framework that provides a robust set of features to develop applications.

Selection Reasoning:

- **Pre-Built Modules:** Express js allows to set up middlewares in order to respond to HTTP requests with minimal lines of code. Moreover, it also provides you with a routing component to perform different actions based on HTTP Method and URL.

3.4.2 External Plugins:

Nodemailer:

"Nodemailer" is a plugin for Node JS applications for sending emails. This plugin will be used to automate email sending tasks to notify potential donors about the blood requests

3.4.3 Databases:

MySQL

Our choice of Database is MySql. MySql is a relational database management system which uses structured query language (SQL). Data is organized in tables and relations. SQL is the language used to create, modify and extract data from the relational database, as well as control user access to the database.

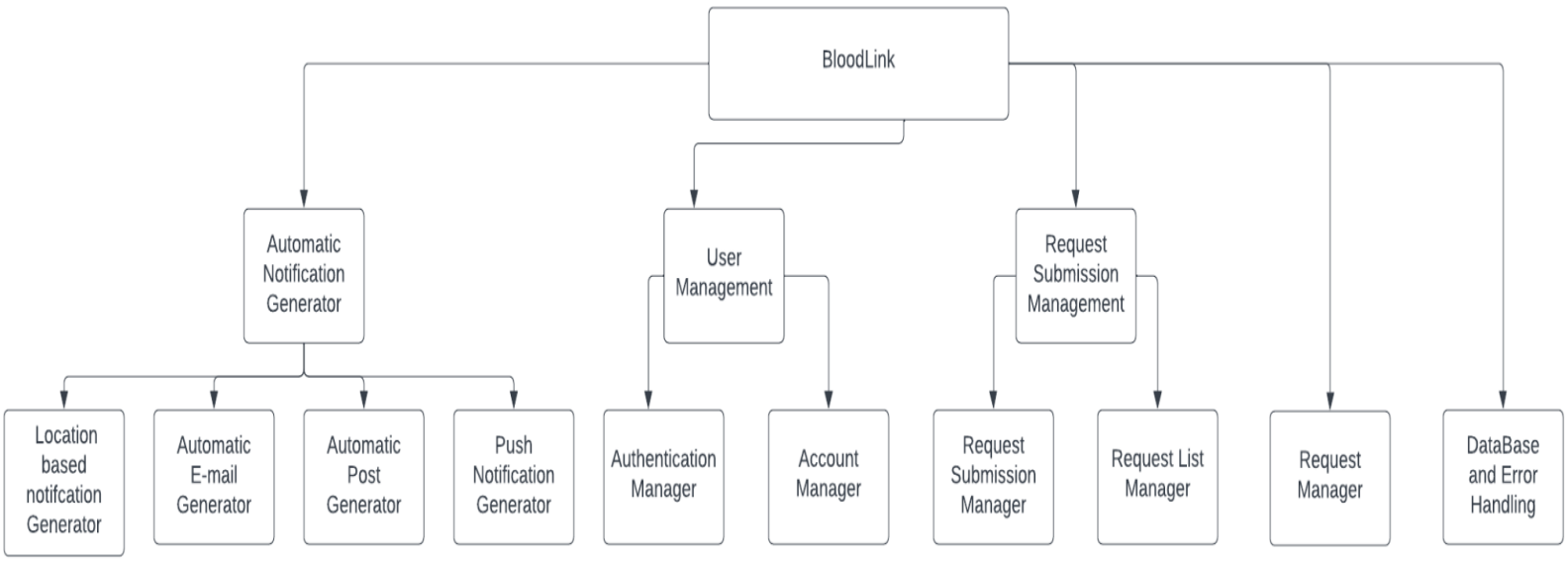
Selection Reasoning:

- We are using structured data, if we use noSQL we have to use an ORM on top of that to perform structured queries which slows down the performance of the overall system.
- We have relations between users and requests. We prefer having a relational database like MySQL.
- MYSQL has ACID (Atomicity, Consistency, Isolation, and Durability) properties which ensures reliability of transactions.
- Allows joins which minimizes duplication of data which keeps the database size small, which is preferable for mobile applications.

4 System Architecture

4.1 System Architecture

4.1.1 System Component Breakdown:



4.1.2 Decomposition Strategy:

First Level Decomposition:

- Identify high-level components from the user's perspective. We achieved this by looking at the user screens and use cases. This will help break the system into smaller components without which the system would not function as expected.

1. Authentication Manager
2. Account Manager
3. Request Submission Manager
4. Request List Manager
5. Request Manager
6. DataBase and error handling

7. Location-based notification generator
8. Automatic Email Generator
9. Automatic Post Generator
10. Push Notification Generator

Second Level Decomposition:

- Identify the components that have similar tasks and create a subsystem based on that.

1. User Management
2. Request Submission Management
3. Automatic Notification Generator

4.1.3 Component Functionality:

- **Authentication Manager**
    a. Login
    b. Password Encryption/Decryption

- **Account Manager**
    a. Create Account
    b. Delete Account
    c. Change Password
    d. Update Account information

- **Request Submission Manager**
    a. Enter Request Details
    b. Cancel Request
    c. Submit Request

- **Request List Manager**
    a. View List of Requests

b. Add Request
c. Delete Request

- **Request Manager**
    a. Edit Request
    b. DeleteRequest
    c. Update Request

- **DataBase and error handling**
    a. Retrieve, Add, Update, Delete data
    b. Handle errors and send error messages

- **Location-based notification generator**
    a. Retrieve Location of the Request and send SMS Request to donors within the specified range.

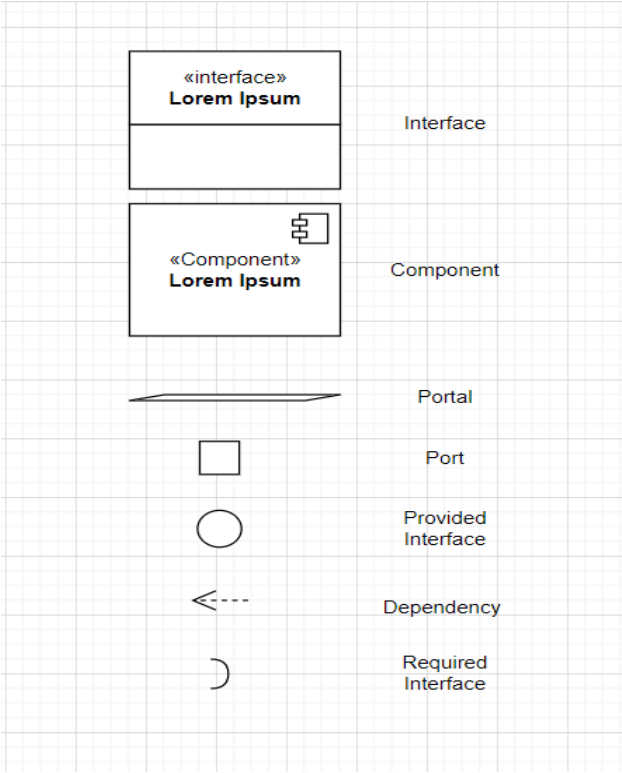
- **Automatic Email Generator**
    a. Send Request email when the request is urgent and the email counter is less than specified emails per day.

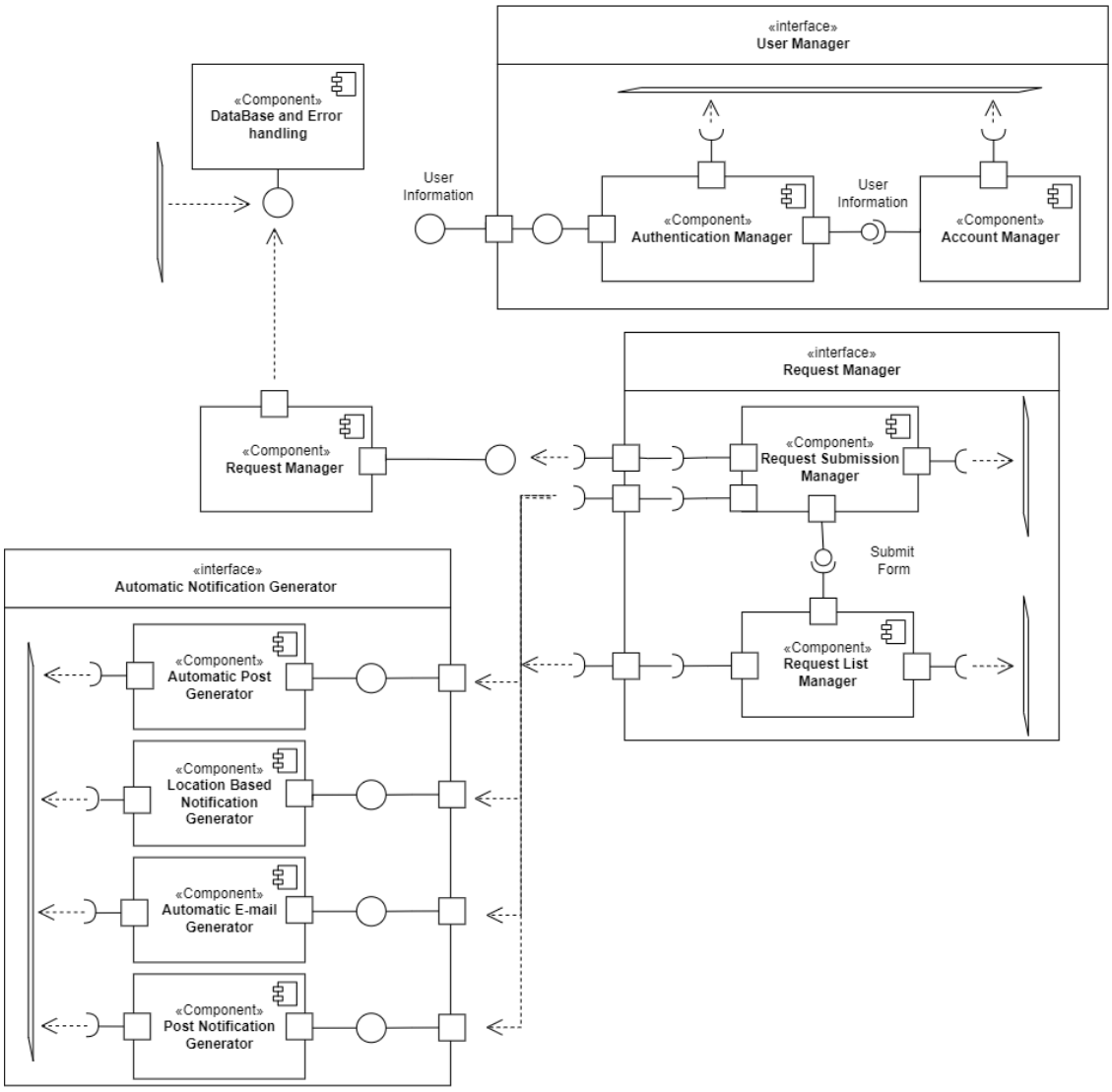
- **Automatic Post Generator**
    a. Post Request on Social Media

- **Push Notification Generator**
    a. Send Push Notifications of the Request to all registered donors.

4.1.4 Component diagram:

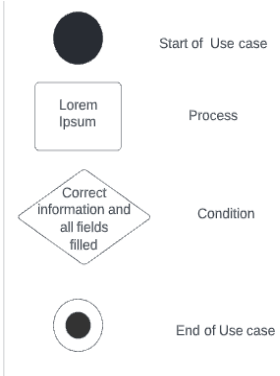
Key



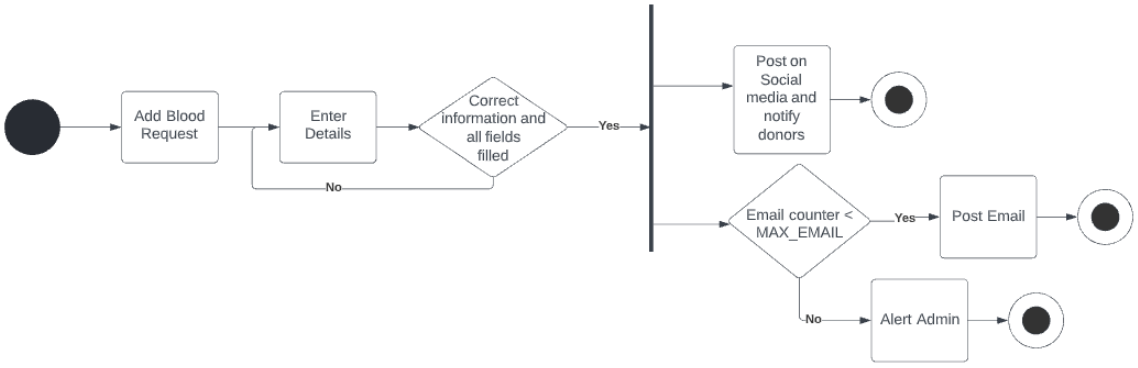


4.1.4 Activity diagrams:

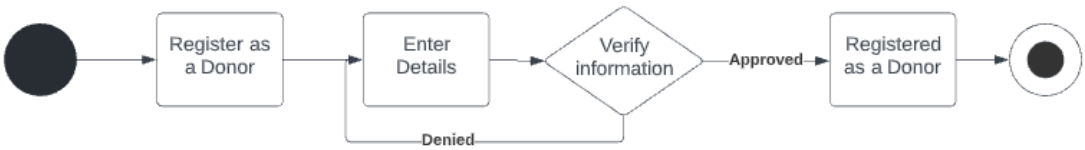
Key



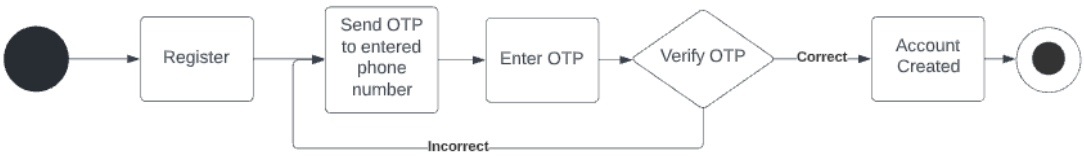
a. Create Blood Request



b. Register as a Donor



c. Phone number verification



4.2 Subsystem Architecture

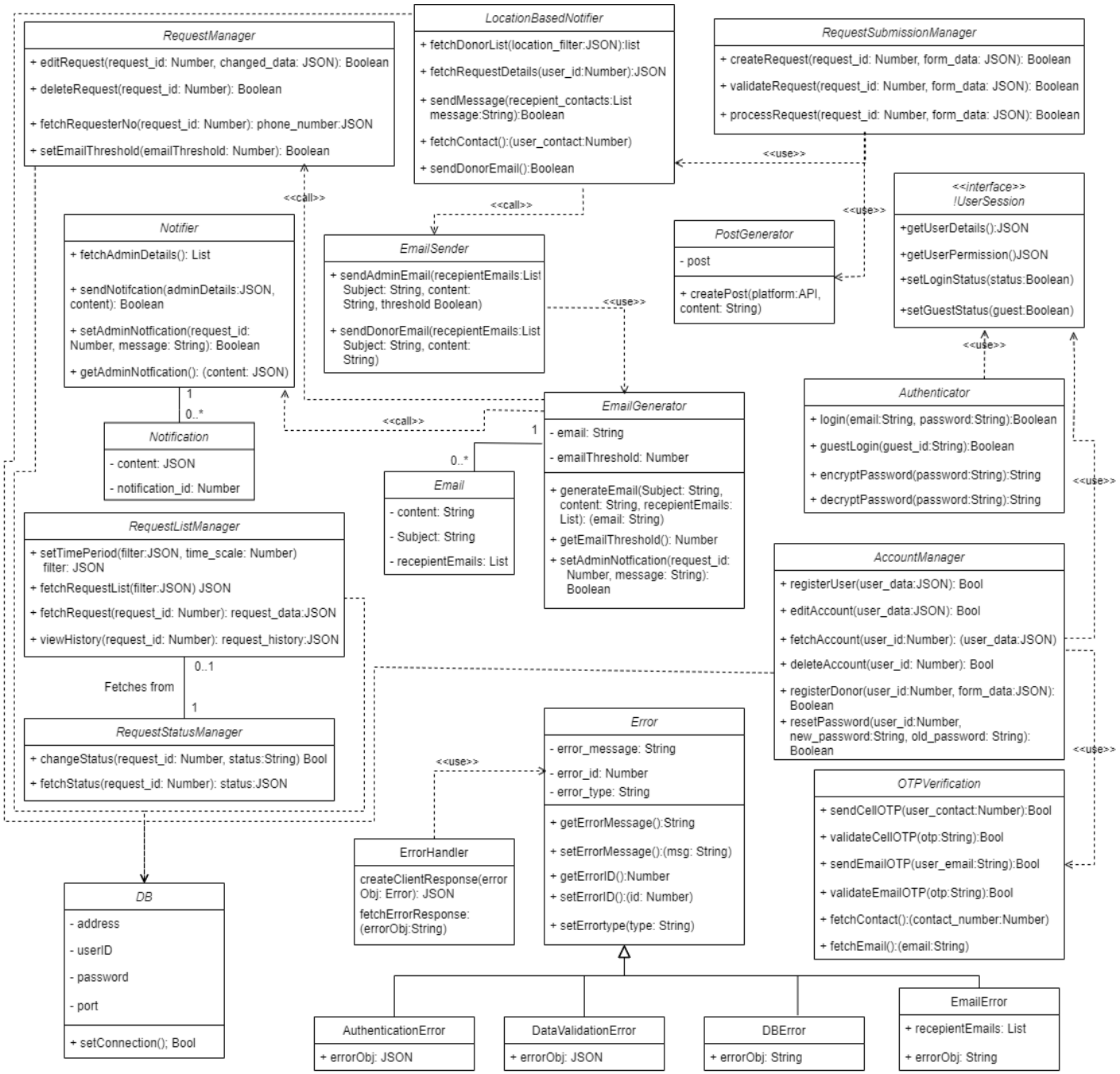
4.2.1 Decomposition of components in classes:

High level components of the system are divided into subcomponents and classes, which are kept coherent with class diagrams. Each class has its respective methods/functions as displayed in table below:

Components	Class(es)	Methods
Authentication Manager	Authenticator	login, otp Verification, encryptPassword, decryptPassword
Account Manager	AccountManager	registerUser, editAccount, deleteAccount, changePassword, fetchAccount, resetPassword, registerDonor
Request Submission Manager	RequestSubmissionManager PostGenerator	createRequest, validateRequest, processRequest, createPost, pushNotification
Request List Manager	RequestListManager	fetchRequest, fetchStatus, changeTimePeriod, viewHistory, fetchRequestList
Request Manager	RequestManager RequestStatusManager	editRequest, deleteRequest, fetchRequestNo, contactRequestor, setStatus, changeStatus
DataBase Manager	DB	startConnection
Error Handler	ErrorHandler Error - AuthenticationError - VerificationError - DBError - EmailError	errorMessage, setError, clientResponse, errorObj, recepientEmails
Automated Email Generation	EmailGenerator EmailSender Email	generateEmail, sendEmail, getAutomationRules, sendAdminEmail, sendDonorEmail
Automated Post Generator	PostGenerator	createPost, getMessage, setMessage

Location Based Notification Generator	LocationBasedNotifier Notifier	fetchDonorList, fetchRequestDetails, sendMessage, fetchContact, sendDonorEmail, fetchAdminDetails, sendNotification, sendAdminNotifcation, getAdminNotification

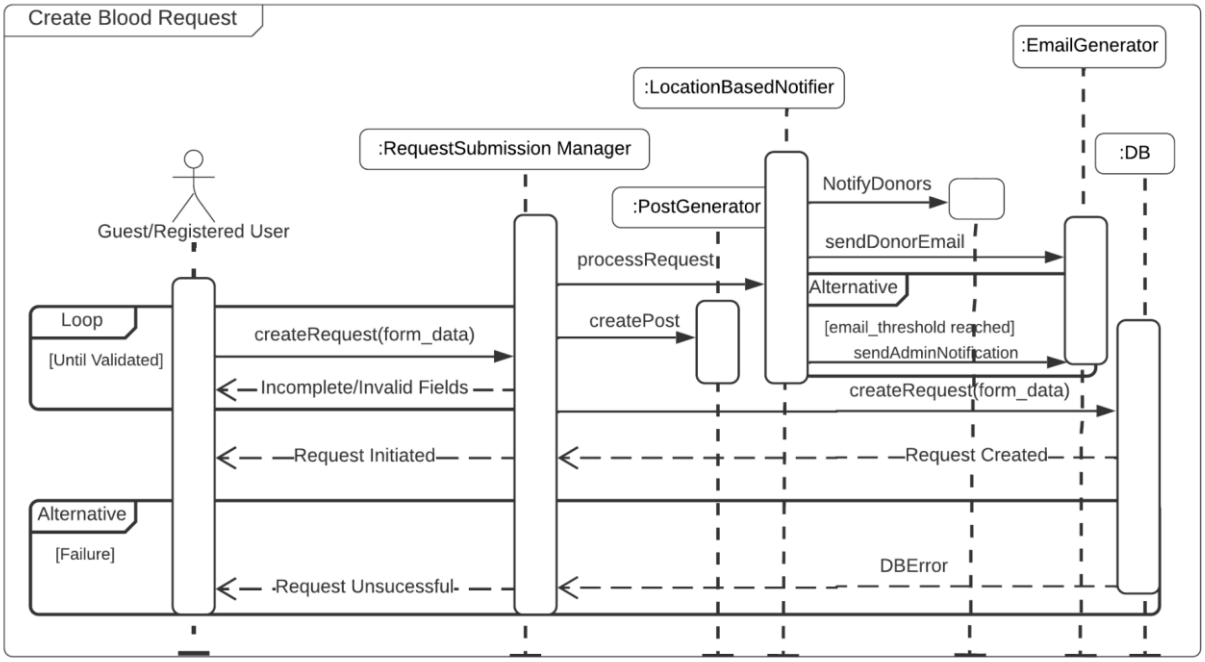
4.2.2 Class Diagram:



4.2.3 Sequence diagrams:

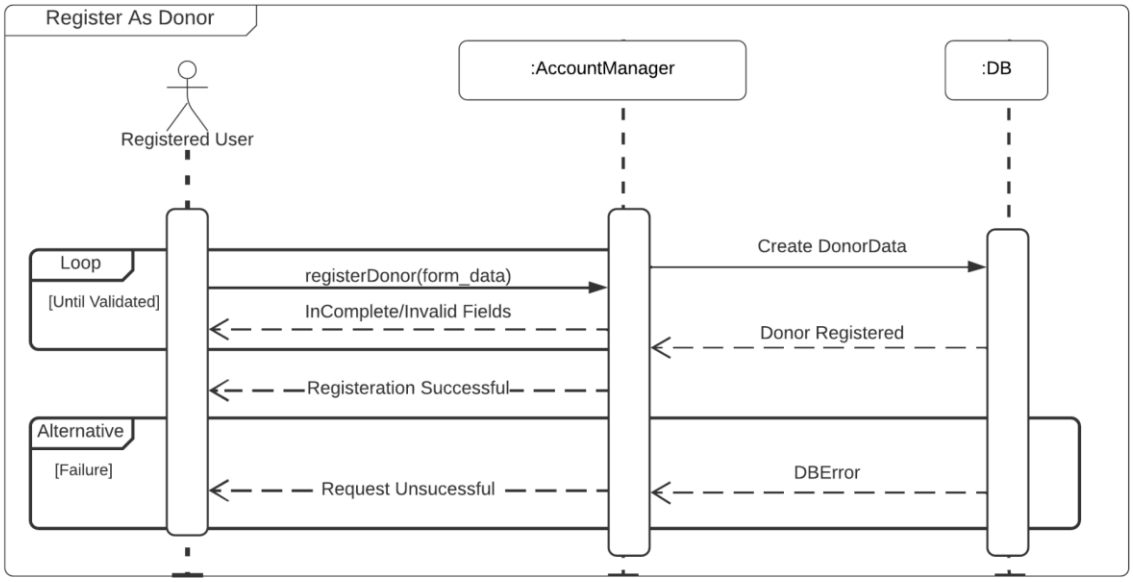
Sequence Diagrams of top three use cases.

a. **Create Blood Request:**



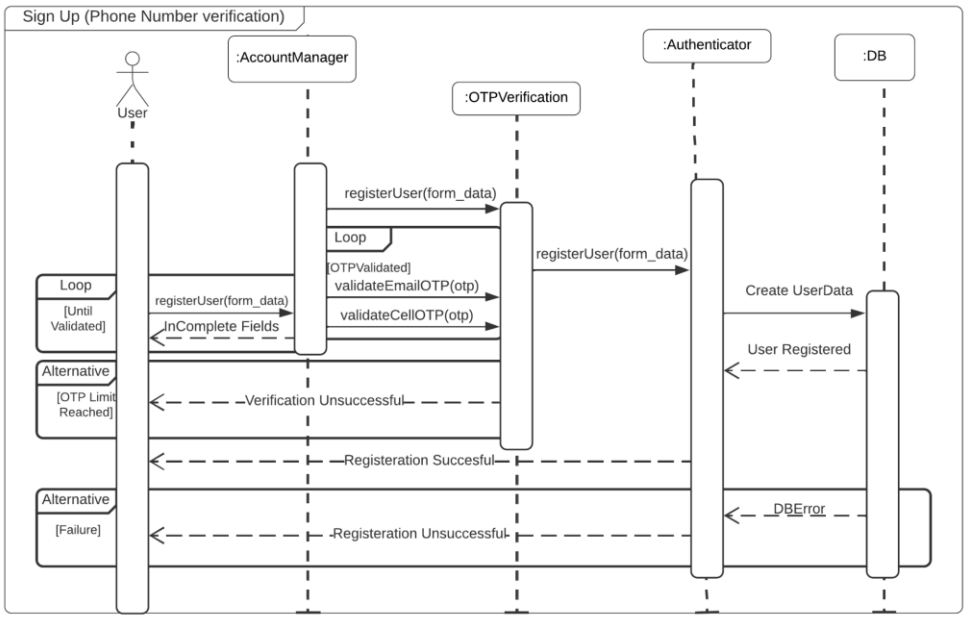
Guest/Registered users can create blood requests by filling required fields on the form and submitting the request. The form data will then process in the request submission manager where required fields are checked and data is sent to Database. An automated post is generated on social media platforms and automated notification is sent to location-based (neighboring) donors. An automated email is sent to all donors if the email threshold isn't reached. Else, a notification is sent to Admin to process email manually. In case of a DB error, a failure message gets displayed.

b. **Register as Donor**



Registered users can register themselves as donors, by providing the required information. The information is then validated in AccountManager and is forwarded to Database. In case of DB error, an error message gets displayed.

c. **SignUp (Phone Number Verification)**



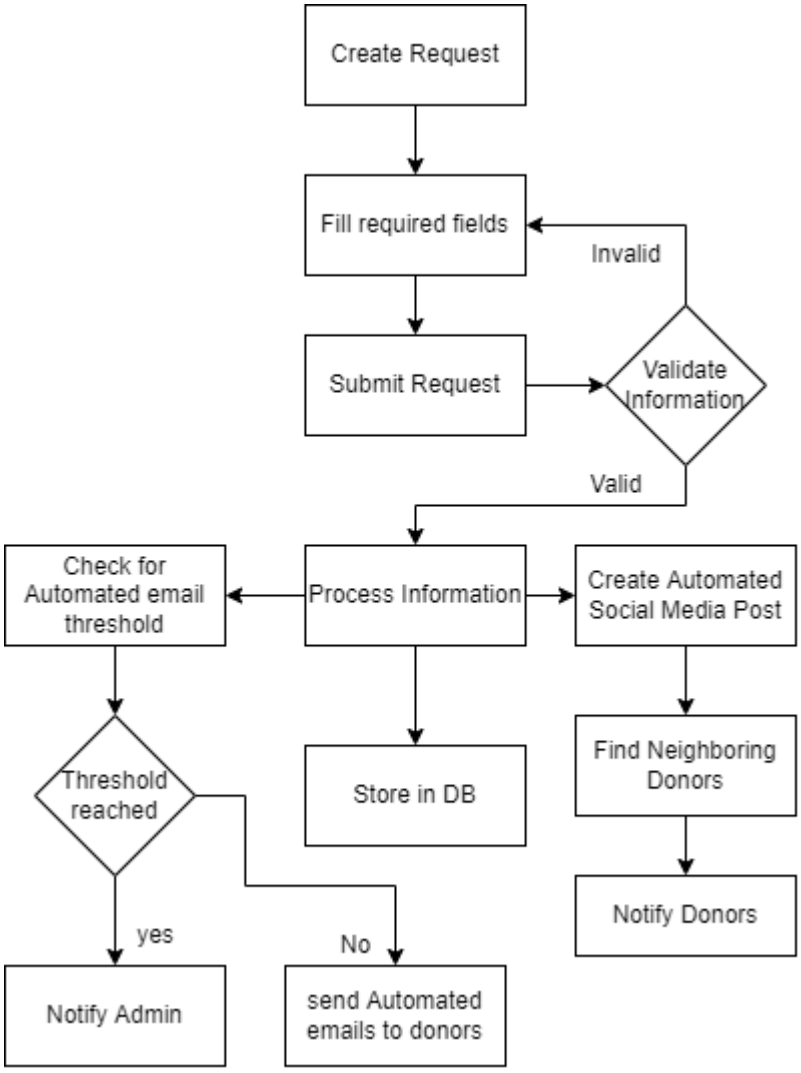
Users can register themselves by the Sign-Up method. For sign-up, users enter required information including phone numbers. This phone number is first validated by OTP authentication. An OTP is generated and sent to the number provided by the user and is validated. Upon authentication, user data is processed by Authenticator and sent to Database. In case of a database error, a failure message gets displayed.

4.2.4 Flow diagrams:

Following flow diagrams demonstrate detailed working:

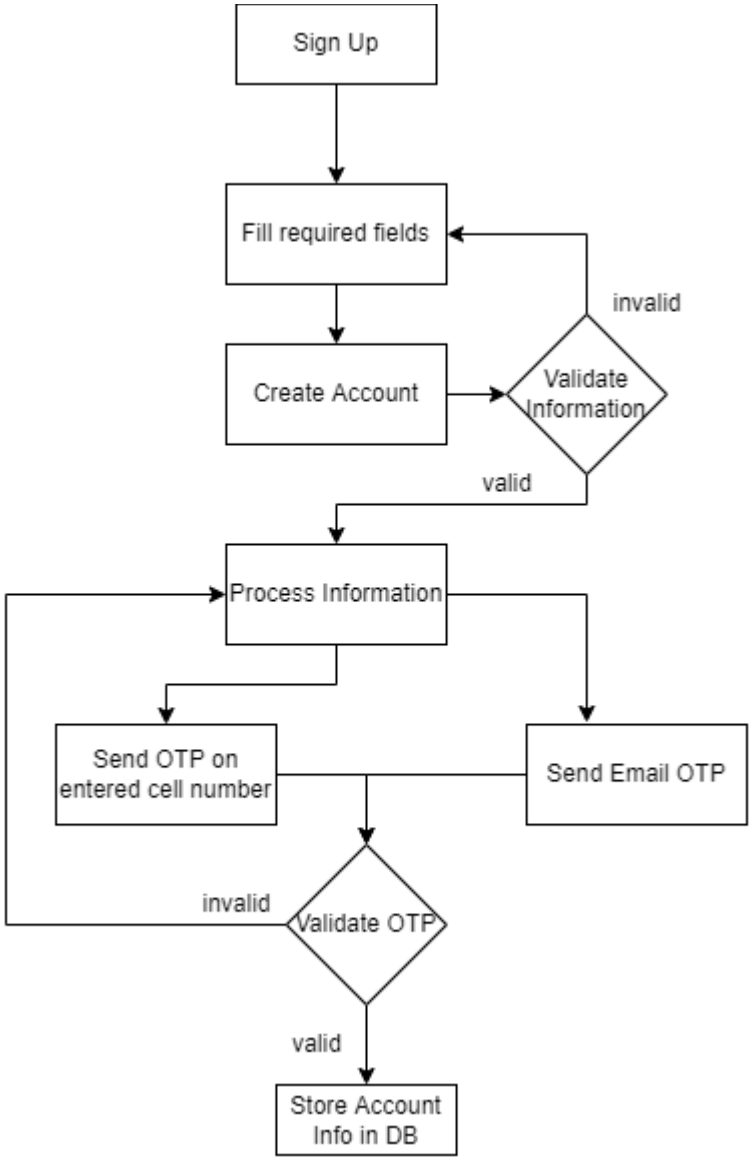
RequestSubmissionManager:

Request Submission Manager creates, validates, and processes new requests. The following diagram displays the flow for the request submission module:



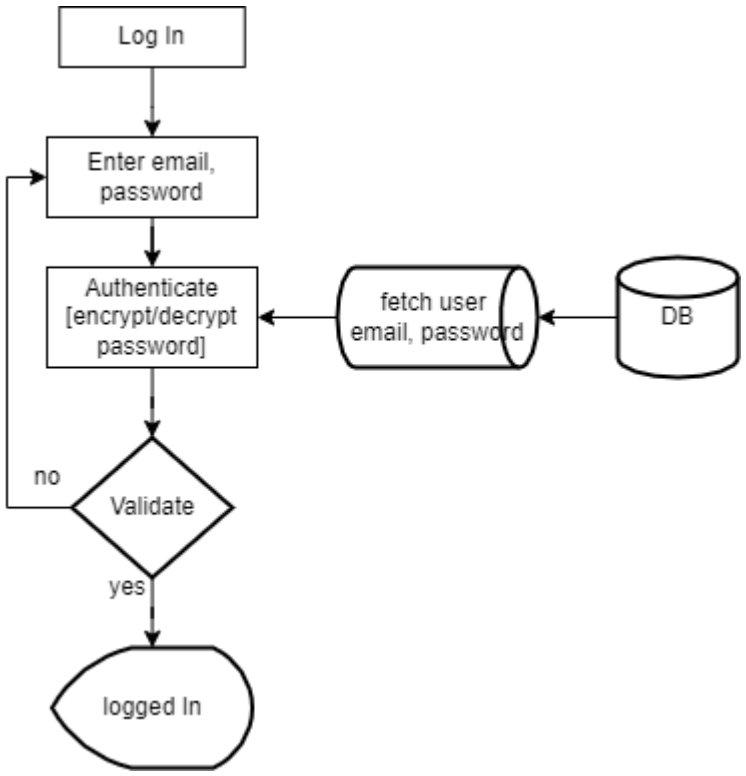
AccountManager:

Account Manager deals with creating an account, deleting an account, editing an account details. The following diagram demonstrates the flow of account creation.



Authenticator:

Authenticator encrypts and decrypts passwords and verifies user credentials. Login functionality uses both AccountManager Subsystem and Authentication Manager. The flow diagram is as follows:



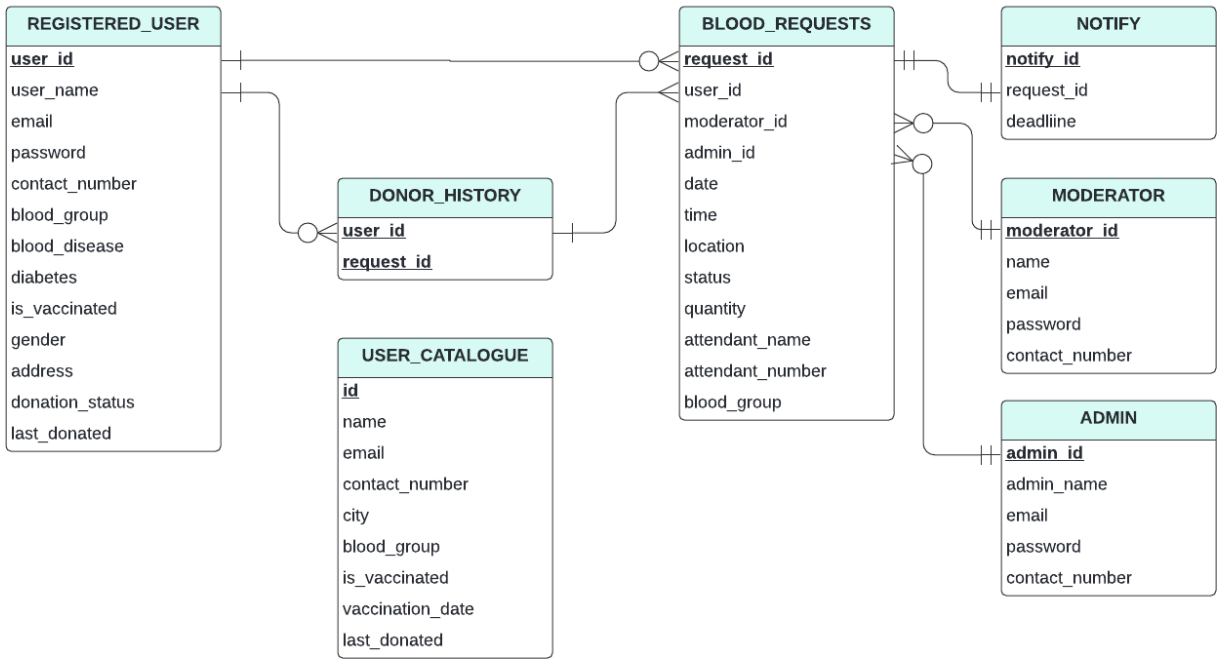
4.3 Data Structure

Bloodlink doesn't require any particular data structure to implement functional and non-functional requirements

4.4 Database Model

4.4.1 Database scheme and detailed description

Schema Description and details



REGISTERED_USER

Schema to store details when the user registered on the app. This information will also be used when the system finds a donor in response to a blood request.

One REGISTERED_USER can add Many BLOOD_REQUESTS

One REGISTERED_USER can donate blood for Many BLOOD_REQUESTS

Attribute	Type	Description
user_id	Number	Unique id to keep record of individual registered user details
user_name	String	Full name of the user
Email	String	Email ID

Password	String	Password of the registered user
contact_number	Number	Phone number of the user used to notify about the blood requests
blood_group	String	Blood group of the user
blood_disease	Boolean	Whether the user has any blood disease or not to identify donating eligibility
Diabetes	Boolean	Whether the user has diabetes or not to identify donating eligibility
is_vaccinated	Boolean	Whether the user is COVID vaccinated or not to identify donating eligibility
Gender	String	Gender of the user
donation_status	Boolean	To determine whether this user is open to donate blood or not
last_donated	DateTime	To check the donating eligibility if the time elapsed is less than 8 weeks
Address	String	Address of the user to implement geolocation request feature

USER_CATALOGUE

Bloodlink has a separate database which they used to find donors and it gets updated annually. This schema will be used when the system finds a donor in response to a blood request, however the users may or may not be registered with the app.

Attribute	Type	Description
Id	Number	Unique id to keep record of individual user details
Name	String	Full name of the user

Email	String	Email ID
contact_number	Number	Phone number of the user used to notify about the blood requests
blood_group	String	Blood group of the user
is_vaccinated	Boolean	Whether the user is COVID vaccinated or not to identify donating eligibility
vaccination_date	DateTime	To check the donating eligibility of the donor
last_donated	DateTime	To check the donating eligibility if the time elapsed is less than 8 weeks
City	String	City of the user to implement geolocation request feature

BLOOD_REQUESTS

Schema to store the details of a blood request initiated by the user to track progress and history

One REGISTERED_USER can have Many BLOOD_REQUESTS

One BLOOD_REQUESTS will have only One NOTIFY

One ADMIN can have Many BLOOD_REQUESTS

One MODERATOR can have Many BLOOD_REQUESTS

Attribute	Type	Description
request_id	Number	Unique id to keep record of individual requests
attendant_name	String	Full name of the point of contact incase a donor wants to donate blood
attendant_number	Number	Phone number of the point of contact in case a donor wants to donate blood

blood_group	String	Blood group required
Date	DateTime	Date when the blood is required
Time	DateTime	At which time the blood is required
Location	String	Location of the hospital
Quantity	String	Quantity of the blood required
Status	Boolean	To determine whether a request has been resolved or not
user_id	Number	user_id of the user who added the blood request. In case the user is a guest, it would be NULL
admin_id	Number	admin_id of the admin who added the blood request. In should be unique and could be NULL
moderator_id	Number	moderator_id of the moderator who added the blood request. In should be unique and could be NULL

ADMIN

Schema to store the details of the admin account

One ADMIN can have Many BLOOD_REQUESTS

Attribute	Type	Description

admin_id	Number	Unique id to keep record of individual admins
admin_name	String	Full name of admin
Email	String	Email ID
Password	String	Password of the admin
contact_number	Number	Phone number of the admin

MODERATOR

Schema to store the details of the moderator account which will be created by admins

One MODERATOR can have Many BLOOD_REQUESTS

Attribute	Type	Description
moderator_id	Number	Unique id to keep record of individual moderator
Name	String	Full name of the moderator
Email	String	Email ID
Password	String	Password of the moderator
contact_number	Number	Phone number of the moderator

DONOR_HISTORY

Schema to store the details of the requests on which a registered user has denoted blood

A single DONOR can have donated blood for MANY BLOOD_REQUESTS

A single REGISTERED_USER can have donated blood for MANY BLOOD_REQUESTS

Attribute	Type	Description
user_id	Number	Unique id to keep record of individual registered users
request_id	Number	Unique id to keep track of requests on which a registered user has denoted blood

NOTIFY

Schema to store the request id to keep track of time. Incase a request has not been resolved and the deadline is close admins will be notified to take actions accordingly

Only one BLOOD_REQUEST will have one NOTIFY

Attribute	Type	Description
user_id	Number	Unique id to keep record of individual registered users
request_id	Number	Unique id to keep track of requests on which a registered user has denoted blood

4.4.2 Database

MySQL

Our choice of Database is MySql. MySql is a relational database management system which uses structured query language (SQL). Data is organized in tables and relations. SQL is the language used to create, modify and extract data from the relational database, as well as control user access to the database.

Reasons for using MySQL:

- We are using structured data, if we use noSQL we have to use an ORM on top of that to perform structured queries which slows down the performance of the overall system.
- We have relations between users and requests. We prefer having a relational database like MySQL.
- MYSQL has ACID (Atomicity, Consistency, Isolation, and Durability) properties which ensures reliability of transactions.
- Allows joins which minimizes duplication of data which keeps the database size small, which is preferable for mobile applications.

4.5 External Interface Requirements

4.5.1 User Interfaces

Users will interact with the app using a GUI. The user interface is kept user-friendly and consistent throughout and can be accessed through both android and IOS devices. The UI elements, fonts, contents are in correspondence with material design guidelines. Following conventions were followed to enhance user accessibility and convenience:

- **Required Skill Level:**

The UI is minimalist and uses simple text. So a basic understanding of smartphone applications will suffice. The basic skills include understanding of login/Sign Up, email, OTP verification, and filling requirement fields, etc.

- **Supportive Graphics:**

  UI contains many supportive icons along with text and buttons that increase user convenience.

- **Terminologies:**

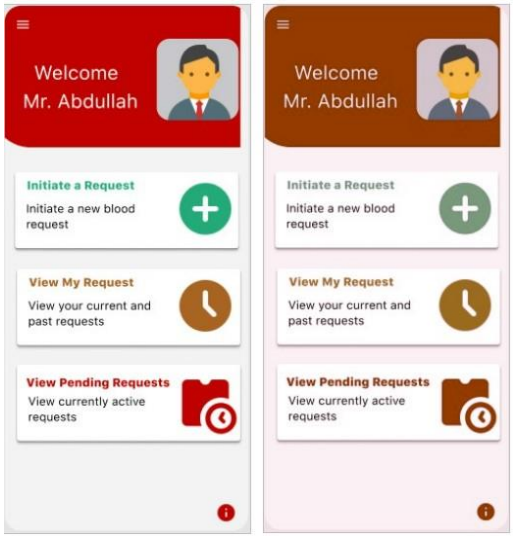
  UI uses self-descriptive, well-defined, and conventional terminologies that prevent user perplexity.

- **Accessible to users with disabilities:**

  Application usability remains the same even with no sound output, to accommodate users with auditory disabilities. We are making our application accessible to people who might have slight disabilities in sight, for example, color blindness, the screens view to people with different color blindnesses can be found below:

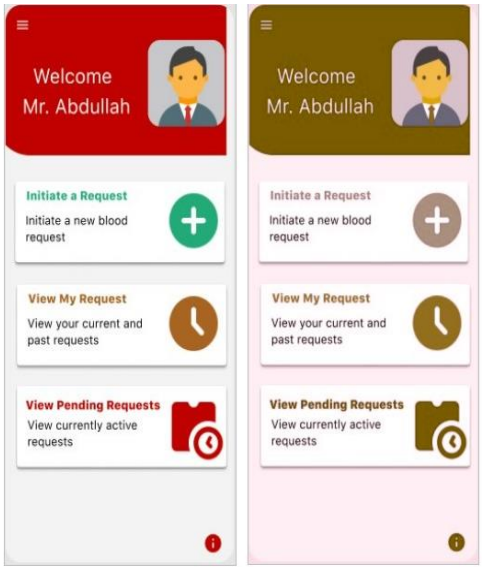
1. **Deuteranomaly**

   Deuteranomaly or green-week is color blindness in which a user sees less green. Thus, the eye cannot form the proper color of any object that contains green. The screens for normal and deuteronomic people are shown below. It can be seen that all the text and icons are still distinguishable and a deuteronomic person will not face any difficulty using this screen.



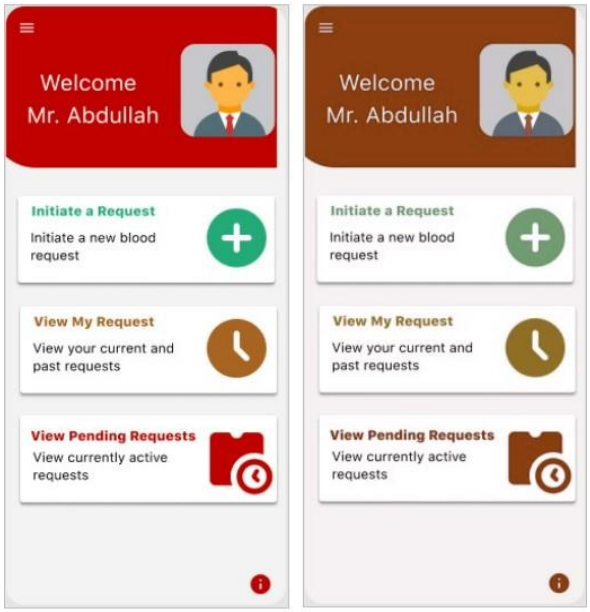
2. Deuteranopia

Deuteranopia or green blindness occurs when a user cannot see green color at all. The images formed are a mixture of red and blue and green is completely ignored. The screens for normal and deuteranopic people are shown below. It can be seen that all the text and icons are still distinguishable and a deuteranopic person will not face any difficulty using this screen.



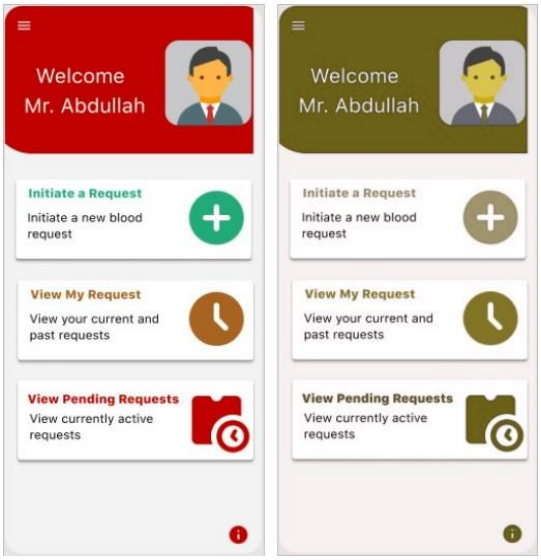
3. Protanomaly

Protanomaly or red-weak is color blindness in which the eye cannot perceive the red color properly. Thus, the eye cannot form the proper color of any object that contains red. The screens for normal and a protonamic person are shown in the image. It can be seen that all the text and icons are still distinguishable and a protonamic person will not face any difficulty using this screen.



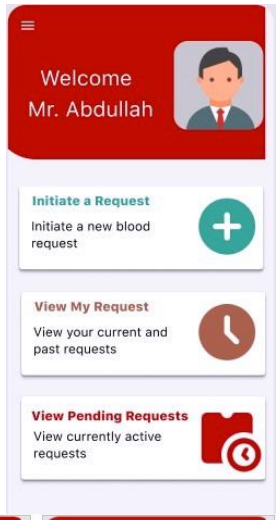
4. Protanopia

Protanopia or red-blind occurs when eye-cones to sense red light do not function. The person cannot see the red color at all. The images formed are a mixture of blue and green, while red is completely ignored. The screens for normal and a protanopia person are shown below. It can be seen that all the text and icons are still distinguishable and a protanopia person will not face any difficulty using this screen.



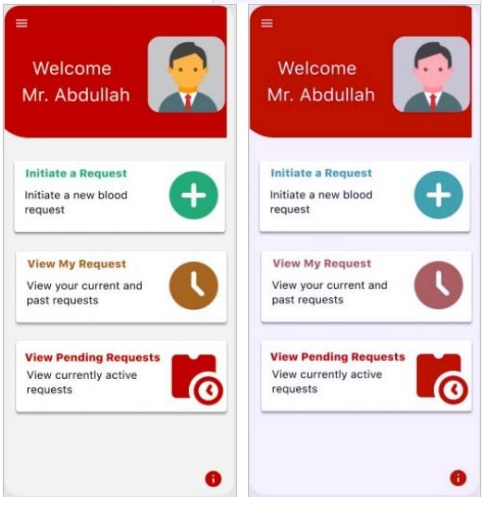
5. Tritanomaly

Tritanomaly or blue-weak is color blindness in which the eye cannot perceive the blue color properly. Thus, the eye cannot form the proper color of any object that contains blue. The screens for normal and a tritanomaly person are shown on the right. It can be seen that all the text and icons are still distinguishable and a triatomic person will not face any difficulty using this screen.



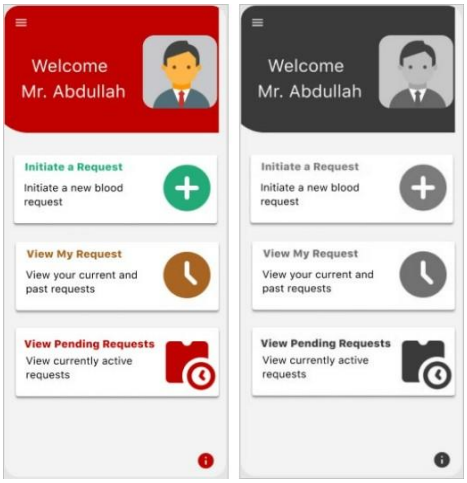
6. Tritanopia

i. Tritanopia or blue-blind occurs when eye-cones to sense blue light do not function. The person cannot see the color blue at all. The images formed are a mixture of red and green, while blue is completely ignored. The screens for normal and a tritanopia person are shown below.It can be seen that all the text and icons are still distinguishable and a tritanopia person will not face any difficulty using this screen.



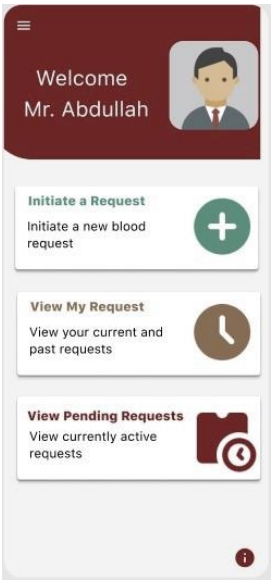
7. Monochromacy

Monochromacy occurs when the eye can perceive only one light intensity, so essentially, they see everything as black and white. The screens for a normal person and a monochromatic person can be seen on the right. It can be seen that the text and icons are still distinguishable and readable for a monochromatic person. So, a monochromatic person will not face any difficulty while using this application.



8. Blue Cone Monochromacy

Blue cone monochromacy occurs due to the deficiency of red and green cones in the eyes. It causes severely impaired color discrimination, low vision, nystagmus, and photophobia. The screens for a normal person and a person with blue cone monochromacy can be seen in the image. In image all the text and icons are still distinguishable and a blue cone monochromatic person will not face any difficulty using this screen.



4.5.2 Hardware Interfaces

BloodLink app can be accessed through any android or IOS device since it is developed through cross-platform i.e., flutter. Although it is not accessible on laptop/desktop through browsers, However users can still use mobile applications on PCs using emulators. The app requires a stable internet connection for smooth working.

4.5.3 Software Interface

The application will run on both android and IOS devices. It is to be noted that a user will need a smartphone to access it. The application might not be supported on older mobile phones.

5 User Interface Design

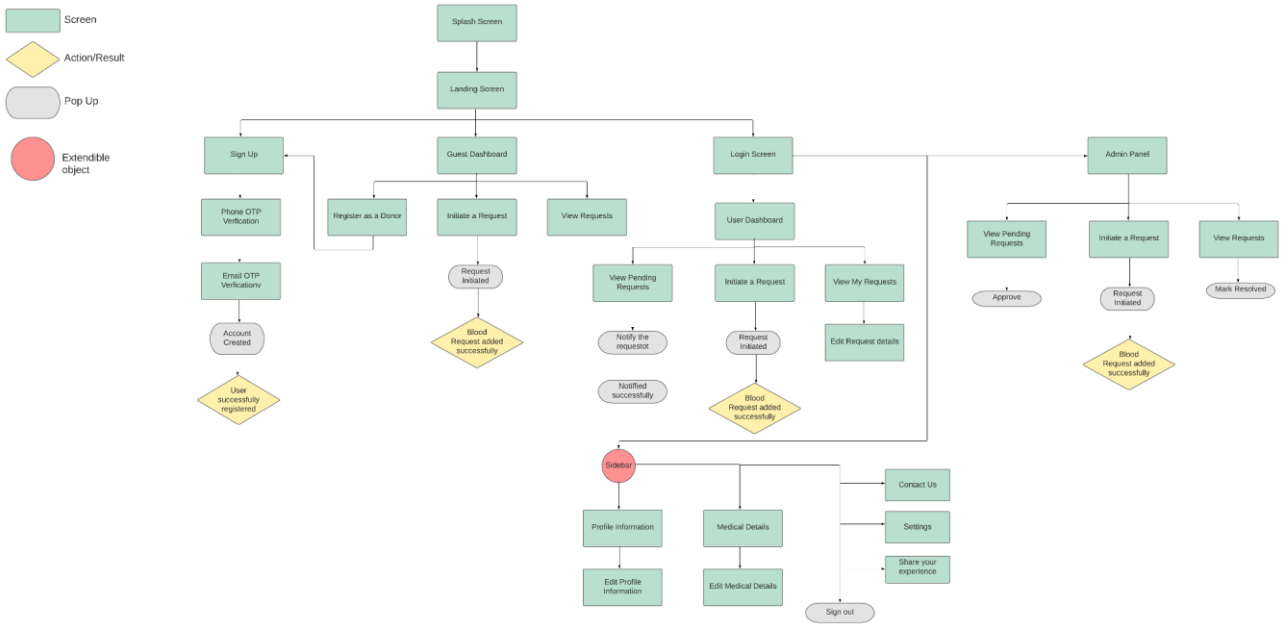
5.1 Description of the user interface

Flutter

Flutter is an open-source cross-platform instrument for the development of applications' UI. Other native app frameworks rely on their specific UI components to paint the scaffold. However, Flutter doesn't need platform-specific UI components to render the whole UI which makes it faster and easier to maintain consistency of design and logic across different platforms. Flutter apps are written in DART language which is similar to javascript.

We are going to use Flutter because of its ability to build cross-platform apps with reduced code development time compared to other native frameworks. It utilizes "hot reload" functionality to instantly rerender the UI components without losing the current application state. Moreover, flutter will give us the flexibility to implement demanding complex UI logics and handle component states because of its own powerful rendering engine. Besides that, Flutter provides pre-built plugins to deal with the implementation of fetching GPS coordinates, handling permissions etc.

5.2 Information architecture



Relationship between screens:

- When the user opens the app for the first time, it will be shown a splash screen having a bloodlink logo.
- The Landing Screen will allow the users to either login, continue as guest or make an account if the user is not registered.
- The Sign Up screen will allow the users to enter valid details and create an account. In this way, the user can donate blood as well
- Guest Dashboard will enable the user to add a request, view past and active request or the user can also register for the donation
- Login Screen will allow the users to their credentials and go to the user dashboard.
- User Dashboard provides the user with multiple options
  - initiates a blood request by adding request details.
  - View the pending blood requests for which a donor has not been found yet. The user can then contact the requestor to donate blood
  - View all requests added by the user. Through this screen, the user can edit the details of the current request

- The sidebar panel on the user dashboard provides the user with the following further option
  - A user can view his/her profile information and can edit the details as well
  - A user can view his/her medical details and can edit them as well
  - A user can go to contact page where he/she can submit a message to the admins
  - A user can share the app details with others

5.3 Screens

The screens along with their Goal, the UI Details, and user requirement are as follows:

1. **The Landing Page**

The landing page is the first screen a user sees after the loading page.

Goal:

The goals of this screen include allowing the user to get started with the application. This screen allows the user to either login, sign up, or continue as a guest. They can also view information about the application here.

User Interface:

The user interface is made by keeping in mind thus helping him to navigate the application easily by building a mental model. The labels used are standard which will minimize the gulf of execution. The icons are also used to support the text, this will not only be helpful for people who are not very much screen literature but also increases the visual appeal. The user can also view information about the application (like what we are, what we do etc, before logging into the application), this increases mutual trust.

Use Case:

The user can either login or continue as a guest. In case the user does not have an account they can either continue as a guest or sign up. The guest mode will allow the user to use the application in case of emergency, they will be able to initiate a blood request without creating an account.

2. The Login Screen

Goal:

The goal of this screen is to allow the user to login to their account. Since it is quite possible that a user can click on the login button and land on this page while he does not actually have an account. To account for such cases, we are also giving an option to sign up. Keeping in mind the current modern standards and user convenience we are also giving an option to continue with Google, if a user has created an account using Google they can login here.

User Interface:

The user interface of this screen follows the material design guidelines along with the conventions. The color scheme and mapping is kept consistent so that users can better understand and navigate through the application. A button is used for the login button, the dimensions, shadow are kept in accordance with material design. A card is used to continue with Google, making it prominent. A test field along with a label is used to username and password. In case the user forgot his password, he can click on forget password, to get his password. There is also an option to go to the Sign up page, in case the user does not have an account.

Use Case:

The user should be able to login to the application. So, this screen allows him to login. It is important to note that admin can enter their account details here, and they will then view the application as an admin after the entered credentials are verified from the database. Normal user

credentials are also verified, in case the user enters an incorrect information an error message is generated saying. “incorrect username or password”, and if he enters the correct information the user is taken to the home screen.

3. The Guest Mode

Goal:

The goal of this screen is to help the user in case of an emergency. It will allow the user to initiate and view their blood request. We also want a guest user to create an account later, so that our donor database increases, so there is also an option to register as a donor. There is also some functionality in the sidebar, accessed by clicking on the bar menu. (this functionality is explained later).

User Interface:

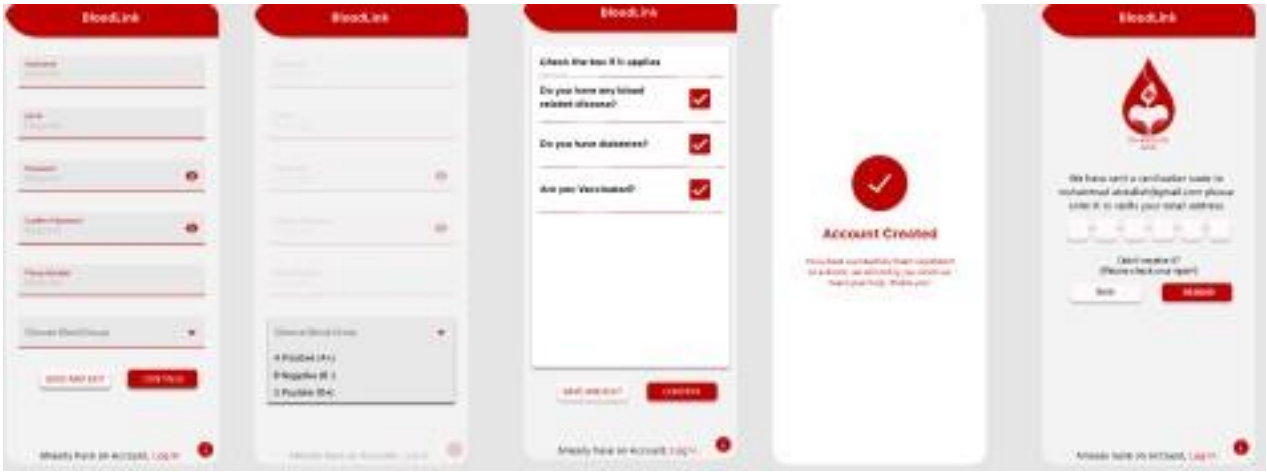
The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options , the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design. We have used a shade of green to initiate a request, because it is a soothing and peaceful color, we want the users to calm down first. The brown color is used to denote the past, so it is used for past requests. Red color is used for registering as a donor, because red color has very high affinity, we want users to click this button later. The placement of the sidebar menu is kept in accordance with the user mental model, the size is also within the material design. The component spacing is also within the material design.

Use Case:

This screen allows the user to initiate a blood request. The user can also view their past requests, as the requests are to be marked as resolved by the user themselves. The guest user might

also be interested in creating an account later, so we have given him the option to register as a donor.

4. User Login



Goal:

The goal of the above screens is to enable the user to create an account, there are some questions that a user must answer to become a donor, since we also email the user or send text messages, so we also verify the user's email and phone number. The goal is to make the user complete this process.

User Interface:

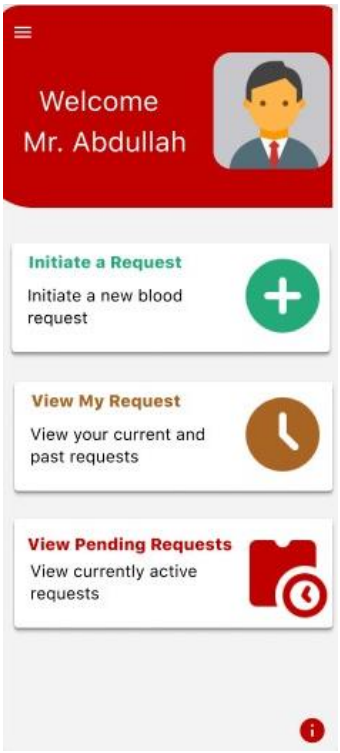
The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. The text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design**.**

Use Case:

It is possible that a user wants to create an account to use the application. In such cases, the user has two options, they can either continue with Google or create a new account. When the user

provides an information, it is verified using OTP (phone or email), once the verification is successful the account is created

5. Home Page



Goal:

The goal of this screen is to allow a logged in user to initiate a request and view his past and current requests. Since, only the logged in users can access this screen, they are also the donors, therefore, they can also see the pending requests. The user can also view more options by clicking on the side bar menu, this menu includes the profile information , medical details, settings, contact us, share your experience and sign out. This screen acts as a main page for the user to navigate through the application.

User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options , the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design.We have used a shade of green to initiate a request, because it is a soothing and peaceful color, we want the users to calm down first. The brown color is used to denote the past, so it is used for past requests. Red color is used to view pending requests, it is because red color has very high affinity, we want users to click this button. The placement of the sidebar menu is kept in accordance with the user mental model, the size is also within material design. The component spacing is also within the material design.

Use Case:

This screen allows the user to initiate a blood request. The user can also view their past requests, as the requests are to be marked as resolved by the user themselves. Since it is a registered user, so they act as donors, therefore, they can also view pending requests. All of these buttons take the user to new screens, where they can perform the required action.

6. Pending Request

Goal:

The goal of this screen is to show the pending requests to the user. The user who is a registered donor, can see requests posted by other people. The goal is to provide them with most of the information through display of minimum information. We want to show full details to only those users who are interested, so they can click on the view more to open more information. The share button is there to enable the user to share the request on their socials.

User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options , the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design. The back button is provided according to the user's mental model.

Use Case:

This screen is there to provide a user information about the pending blood requests. Here, they can see some information about a request, clicking on the view more on the card will provide them with more information about the request. The back button on the top will take the user to the home page. The share button will allow the user to copy and share the request details on some social media platforms.

7. Request Details

Goal:

The goal of this screen is to provide further details about a blood request. This screen also gives the user an option to contact the requestor. Keeping in mind the trends of emerging markets, where people usually do not have credit, we have provided a user to notify the requestor (via notification) that I am interested, kindly contact me.

User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options, the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design.

Use Case:

In case a user is interested to donate to a certain request, they click on view more and they will land on this page. Here, all the information is provided to the user in a structured pattern. The user can click on the contact button to contact the blood requestor via call. The user can also notify the blood requestor in case for some reason, they cannot contact them.

8. View Requests

Goal:

The goal of this screen is to enable the user to view their request. The user can also edit their request, in case they now require less or greater amounts of blood.

The user can also mark the request as resolved in case they do not need the blood anymore.

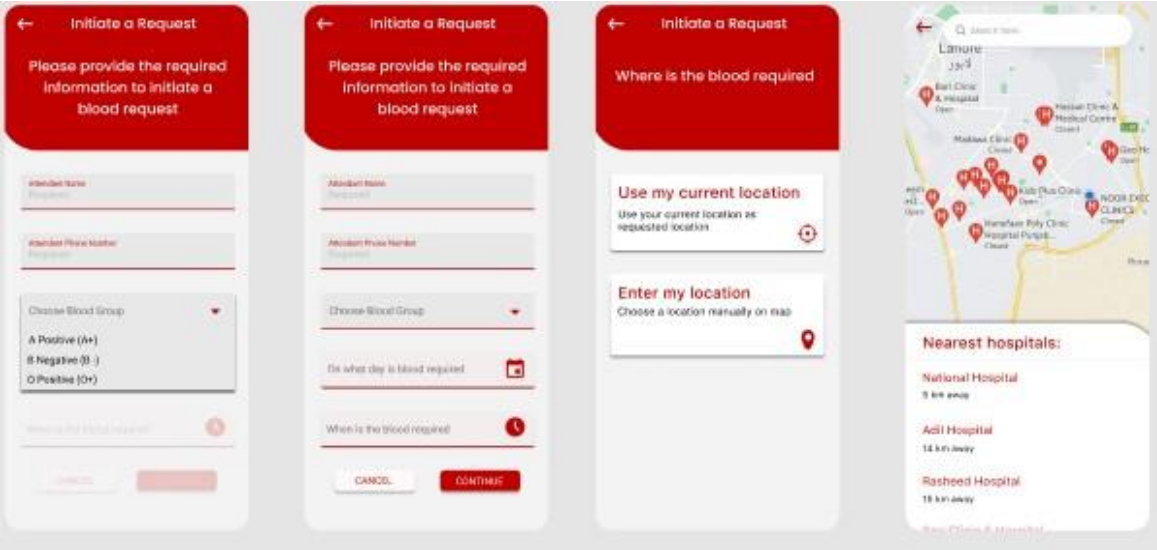
User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options, the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The sizes and dimensions are also according to the material design. Lines are used to separate content on the card.

Use Case:

If a user requests blood, it is quite possible that they require more than one units of blood and now they have received some units and now they require less amount of units, so they can edit their blood request, it is also possible that the blood has been arranged, so the user can mark their request as resolved, so that this request can be removed from currently active requests.

9. Initiate a Request



Goal:

These screens are there to take information from the user about the blood request.

User Interface:

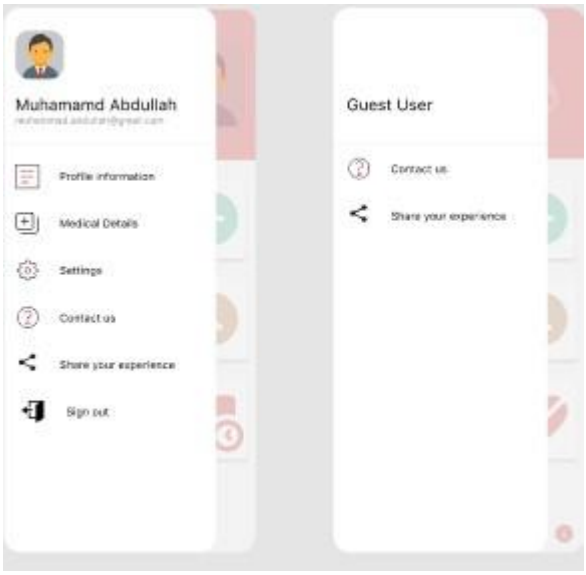
The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. Cards are used to display more options, the text is supported with icons to make it easier for users to understand. The standard icons are used, to make it consistent with the user mental model. The textboxes are also labeled to make it easier for the user to understand this.

Use Case:

These forms are to be filled when a user wants to initiate a blood request. The user provides the necessary information, and selects the location where blood is required.

10. SideBar

Goal:



The sidebar allows the user to do miscellaneous tasks. These screens are different depending upon whether the person is a logged in user or a guest user. This will allow the user to view and update the profile information, medical details and settings, the user can also contact us, share their experience or sign out.

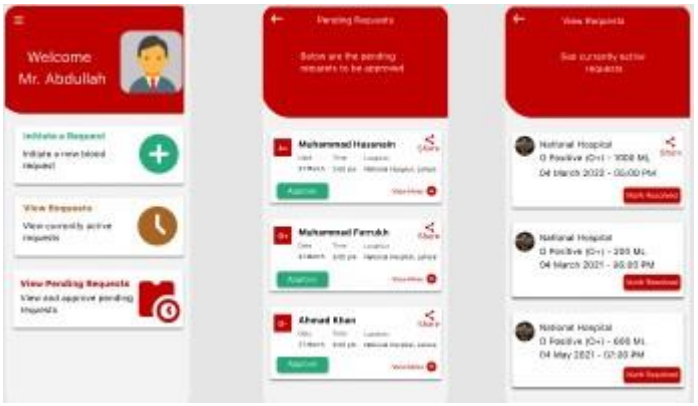
User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. A symbol which already exists in the user mental model is used to open this menu, here, each of the icons is followed by text, thus making it easier for users to understand, moreover, conventional symbols are used. The screen behind the menu (which can be seen on the right side of both screens) is blurred to improve the user focus on the side menu.

Use Case:

It is possible that a user wants to update their application settings or profile information, in such a case, they can open this side menu and click on the desired field to view. The requestors sometimes have certain conditions like, the donor should be vaccinated etc, the medical records information allows the user to update such information about them.

11. Admin Account



Goal:

This is an admin account, the goal of this screen is to allow the admin to initiate a request, view requests and view pending requests. It will serve as the main page to allow the admin to monitor the application. They can view currently active requests as well as the pending requests. The pending requests are required to be approved by the admin. The admin can also initiate a request and can also mark any request as resolved.

User Interface:

The user interface is again kept consistent, both internal consistency of the application and external consistency that comes from the user mental model are being followed. A symbol that already exists in the user mental model is used to open this menu, here, each of the icons is followed by text, thus making it easier for users to understand, moreover, conventional symbols are used. The screen behind the menu (which can be seen on the right side of both screens) is blurred to improve the user focus on the side menu.

Use Case:

The admin can add blood requests. The admin can also view currently active blood requests. The admin can set a threshold for maximum completely automated posts, that option can be accessed by side bar menu and update it in settings. Once this threshold has met, posts start getting added to the pending requests queue, and here the admin can approve them, once an admin approves it an email and text message is sent to registered donors.

12. Miscellaneous

Before initiating any action, we ask the user for confirmation, this prevents accidental requests or requests with false information. It can be seen in the screens below:

When user has filled a form, they see this screen to review the request once, before confirming, once the user clicks on confirm, the request is initiated, and a confirmation message is displayed on the screen. This feedback enables the user to better perceive what has happened with their click. Similarly, such screens with confirmation popups are also available like in the case of logout, as seen in the figure on the right. These small pop ups reduce the Gulf of Evaluation, which is important according to the Norman Design Principles.

5.4 User interface design rules

Following are the interface design rules we followed:

1. **Material Design Guidelines**

   We have completely followed material design guidelines while making our screens. The component suggested by material design for a certain task is used for that particular task, All the attributes of that component are kept as advised by the material design.

   - **Spacing**

     For all of the components used in the screen the spacing suggested by the material design is followed. For example, all the components align to an 8 dp squareline of the grid. The distance between two components is also a multiple of 8 (32)

   - **Aspect Ratio**

     For all the components used in the screens the aspect ratio suggested by the material design is followed. The aspect ratio we have used is 4:3,1:1,3:4.

   - **Icon Size**

     All the icons used within the components follow the material design guideline, that is each icon is 24*24. Even when the icon is placed on a card the height and width of a card are still kept a multiple of 12.

   - **Cards and Dividers**

     Whenever we wanted to display some information about something, we used a card as suggested. As suggested by the material design, our cards do not flip. We have also used dividers to separate the content on a card.

2. **Norman Design Principles**

   While designing the above screens we have also kept the Norman design principles in mind. The principles are listed below:

   - **Discoverability**

     Discoverability is helping users understand where to perform the action, it is catered by making the text and button components with proper signifiers while keeping in mind the conceptual model.

   - **Feedback**

     The feedback is to help users understand the response of the action they

performed. It is very important to keep the user engaged and interested.

- **Conceptual Method**

  The conceptual model is the user's understanding of how something works. If a user can build a good conceptual model of the application, it means they are understanding the flow and will be able to perform the required tasks, this is done by transferring the user mental model from other popular applications into this application.

- **Affordance**

  Affordance is the perceived action of an object. It is like what a user thinks, they can do with a certain object, this is done, by again using the user mental model and it is reinforced by making the application internally and externally consistent.

- **Signifiers**

  The signifier tells, where exactly to perform the action. This is done using the proper labeling and icons so the user can understand where to perform the required action.

- **Mapping**

  Mapping is the relation between the components and actions they perform, this is done by using the knowledge in the head and in the world and being consistent with the actions a certain component performs throughout the application.

- **Constraints**

  The constraints are actually the limitations that these actions cannot be performed. This is mostly done to avoid the “cognition overload”. There could be “physical, logical, cultural and semantic constraints:” according to Norman. In our case, we have logical constraints, like when a pop up or side bar menu appears the remaining screening is grayed out indicating that that component is no longer responsive to touch.

3. **Reduction of Gulfs**

There are two types of gulfs that can exist in a system, the Gulf of execution and the Gulf of evaluation. The Gulf of execution occurs when user wants to perform a certain task but they cannot understand how to perform the task. We have minimized the Gulf of execution by proper usage of signifiers, mapping while keeping in mind the user mental model. The second gulf is the Gulf of evaluation, this happens, when a user performs a certain task and they do not understand the consequences or result of their action, we have catered for this problem by providing user feedback and pop ups for the actions they perform.

4. Heuristic Evaluation

We also kept in mind Nielson's ten heuristic principles. These principles are used to check user interface. They are as follows:

- **Visibility of System Status**

  The visibility of system status is very important for the user to perform any action. We have catered this problem by making all the components visible for the user. The user can find the components with ease and perform the action due to effective use of signifiers and mappings.

- **Match between System and User World**

  We have made sure there is a match between system and user world. This is done by making the system internally and externally consistent. The transfer of learning is used here.

- **User Control and Freedom**

  The user control and freedom is also an important nielsen heuristic principle. We are not forcing a user to perform a certain task by limiting their options. The user has complete freedom and control, and has multiple options on each screen so they can control the flow from each screen and have freedom to perform an action.

- **Consistency and Standards**

  The consistency of the application and following the standards used in applications are very important. We have consistently followed a color scheme and components are also consistent. The icons and labels we have used are also the same as the standard used in other applications.

- **Error Prevention**

  Error prevention is very essential for smooth flow of the application. We have tried to minimize the errors by giving users a confirmation pop up before each operation is performed. For example, a user has to review the blood request form once before confirming, and once he has confirmed, a request is forwarded.

- **Recognition rather than Recall**

  We have used recognition rather than recall to make the step easier for users. We have ensured this by icons with the text labels, so the user can see an icons and instantly recognize what this button will do. The icons we have used are the standard ones used in applications throughout the world, so the user can recognize them due to his mental model.

- **Flexibility and Efficiency of Use**

  We have also ensured that the user has a flexibility to perform actions, they can approach a task in multiple ways to suit the way they prefer, for example, they could either click on their picture to view profile, or click on the hamburger menu to open a sidebar and then view profile.

- **Aesthetic and Minimalist Design**

  We have also made sure that our design appeals to the aesthetic sense of the user and our design is minimalistic, we are not bombarding the user with information and causing a cognition overload, we only show the relevant information and if a user is interested, they can choose to view more. (like in the case of requests).

- **Help, Diagnosis and Recovery from Error**

  We have made sure that if a user makes an error they can get help and recover from that error, for example, if a user does not have an account and they accidentally clicked on login, they can recover from this problem by clicking on don't have an account option on that page. Similarly, in case a user enters an invalid entry he is notified with an error message, so they can recover from that, the feedback enables them to diagnose the problem.

- **Documentation and Help**

  It is also very important to provide documentation and help to the user in case they are facing a problem. We are providing users with an option to contact us, in case they are facing some trouble and want help.

6 Other Non-Functional Requirements

6.1 Performance Requirements

6.2 Safety and Security Requirements

Safety Requirements:

Following safety requirements are associated with the use of BloodLink app:

1. **Privacy of Donor's address:**

   Our user survey showed that around 5 % of people are not comfortable in sharing detailed addresses, due to privacy and safety concerns. To counter this, BloodLink app

will allow requestors to only see the list of available donors in a certain region, and will hide address details of donors.

2. **Medical condition of donor:**

Donors must meet certain medical conditions, as regulated by PTBA (Punjab Blood Transfusion Authority). The major checks include

- Donor is at least 18 years old
- Has not received recent surgery or long-term medications
- Is free from blood transmittable diseases (HIV, syphilis, hepatitis).
- Has not donated blood for the last three months.

BloodLink app will require donors to enter these medical details while registering.

3. **Verification of User details:**

BloodLink app should verify user details such as contact numbers to prevent invalid entries.

4. **Donor Safety:**

BloodLink app will require donors to mark their location status, once they go for blood donation at the requestor's location. This will prevent kidnapping incidents and ensure the donor's safety.

5. **Feedback Mechanism:**

BloodLink app will allow requesters and donors to actively give feedback under the communication thread. This will mitigate incidents of misconduct and will help to improve the user experience.

Security Requirements:

1. **Communication security:**

BloodLink app should use end-to-end encrypted messages for communication.

2. **Data Security:**

   BloodLink app should use a secure hosting service to prevent data leakage.

3. **Database Backup:**

   BloodLink app should use a reliable hosting service that provides multiple data backups.

Expected Level of Security:
Users should expect that:

- Data shared on BloodLink app will stay secure and free from cyber attacks.
- Personal details including name, phone number, address, medical records will not be shared with some external party.
- Only necessary contact details of donors will be shared with requestors.
- Medical records and addiction details of donors will not be directly visible to requestor. Instead, requesters can add filters while searching for donors.

6.3 Software Quality Attributes

6.3.1 Availability:

The server will be available to all sorts of users with access to the internet on the latest browser. It will be available 24/7. It can be accessed using the internet.

.

6.3.2 Usability:

There is another important feature of our software that is usability, it is easy to understand for all sorts of users. The mental model is very easily developed. The UI is intuitive and easy to follow.

6.3.3 Correctness:

Correctness is another important feature of our application. It means that all of the requests are up to date. As soon as the requestor marks it complete it is updated on all clients, moreover, as soon as someone sends a request it is updated on all client ends.

6.3.4 Portability:

There is another important feature in our application that is portability. It can be accessed using a phone, it can also be accessed using a laptop. All you need is a device with the latest browser.

6.3.5 Testability:

Testability or modularity is another important feature of our application. The BloodLink application is easy to test both as a whole and module wise. The application will go through a test run to ensure that all modules work.

6.3.6 Reusability:

Reusability will be another software quality attribute of our software. It can be integrated into other applications to use the same functionality there. Similarly, other software can also be integrated in our application.

6.3.7 Adaptability:

Adaptability is another important feature of our software. It can be adapted depending upon the screen size. Certain features can also be excluded / included based on priorities at that particular time.

6.3.8 Interoperability:

Interoperability is another important feature of our software. It can communicate with other devices. This feature is essential for adding/removing requests.

6.3.9 Maintainability:

Maintainability is another important feature of our application. It can easily be maintained and updated due to extensive documentation, not only by us but also by any other developers. Similarly, all of the requirements can be traced back to their roots, therefore allowing us to maintain it easily.

Appendix A - Group Log

- Held group meetings over zoom to share progress and discuss the overall document.

  Meeting Log:

  - Meeting Duration: (11:45pm - 12:20 am), 23rd Feb

Discussion:

Decided on a deadline before the actual one, divided the tasks, discussed screen-making procedure

- Meeting Duration: (10:30 pm - 1:40 am), 1st march

  Mode: Zoom

  Discussion:

  Divided parts of SDS, Looked at first versions of screens, Planned next steps for SDS completion

- Meeting Duration: (9:30pm - 1:00am), 7th March

  Mode: Zoom

  Discussion:

  Reviewed Databases design, did refinement of screens and designed component and sequence diagrams

- Communicated with clients via Whatsapp to share progress and discuss any concerns.
- Discussed confusions with TA Maha and Sir Mustansar on weekly meetings and via Slack.

Appendix B – Contribution Statement

Name	***Contributions in this phase***	***Approx. Number of hours***	***Remarks***
Ahmad Mahmood	Database Design, Information Architecture	21	
Daim Akram	Class Diagram, Sequence Diagrams, Flow Diagrams	23	
Mohammad Farrukh	Information Architecture, Database Design	22	
Mohammad Hassnain	User Interface, Creating Screens	23	
Sharjeel Ahmed	Component Diagram, System Breakdown, Activity Diagram	21	