RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CSD 334 MINI PROJECT REPORT

BLUD - The Blood Donation App

Submitted by

Adarsh P Sunil (Reg. No: KTE20CS005)
Jinash Jaleel (Reg. No: KTE20CS035)
Saurav K S (Reg. No: KTE20CS052)
Vyshnav C J (Reg. No: KTE20CS060)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRUVANANTHAPURAM

JUNE 2023

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

To be a centre of excellence for nurturing the young minds to become innovative computing professionals for the empowerment of society.

Mission of the Department

- To offer a solid foundation in computing and technology for crafting competent professionals.
- To promote innovative and entrepreneurial skills of students by exposing them to the forefront of developments in the field of computing.
- To inculcate strong ethical values in the young minds to work with commitment for the progress of the nation.

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this report entitled ***BLUD - The Blood Donation App*** *is an authentic report of the project done by the team consisting of* ***Adarsh P Sunil(Reg. No: KTE20CS005), Jinash Jaleel(Reg. No: KTE20CS035), Saurav K S(Reg. No: KTE20CS052), and Vyshnav C J(Reg. No: KTE20CS060)*** *during the academic year* ***2022-23****, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University, Thiruvananthapuram.*

GUIDE COORDINATOR

HEAD OF THE DEPARTMENT

Acknowledgement

Every successful project is the outcome of hard work and strong support and guidance given by a number of people. We sincerely appreciate the inspiration, support and guidance of all those people who have been instrumental in making this project a success.

We express our sincere gratitude to **Dr. Jalaja M. J., Former Principal** and **Dr. Prince A., Principal** for making the resources available at the right time without which this project would not have been a success.

We take this opportunity to express a deep sense of gratitude to **Prof. Kavitha N., Associate Professor & Head, Department of Computer Science and Engineering**.

We also take this opportunity to express our profound gratitude and deep regards to our guide **Prof. Mini Joswin, Assistant Professor** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her from time to time shall carry us a long way in the journey of life on which we are about to embark.

We also express our gratitude to Project Coordinators **Prof. Anil Kumar S** and **Prof. Nisha K.K** for the cordial support, valuable information and guidance, which helped us in completing this task through various stages.

Last, but not least, we thank **almighty, our parents** and **friends** for their constant encouragement without which this project would not have been possible.

Adarsh P Sunil
Jinash Jaleel
Saurav K S
Vyshnav C J

Declaration

We, the undersigned hereby declare that the project report entitled Project **BLUD - The Blood Donation App**, submitted for partial fulfilment of the requirements for the award of the degree of Bachelor of Technology of the **APJ Abdul Kalam Technological University, Kerala** is a bonafide work done by us under the supervision of **Prof. Mini Joswin, Assistant Professor**. This submission represents our idea in our own words where ideas or words of others have not been included; we have adequately and accurately cited and referenced the original sources. We have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea or on our submission. We understand that any violation of the above can result in disciplinary actions from the institute and/ or University and can evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed as the basis for awarding any degree, diploma or similar title of any other university.

Name of Students: Signatures:

Adarsh P Sunil (Reg. No.: KTE20CS005)

Jinash Jaleel (Reg. No.: KTE20CS035)

Saurav K S (Reg. No.: KTE20CS052)

Vyshnav C J (Reg. No.: KTE20CS060)

Batch: 2020 - 2024 August 8, 2023

Abstract

The purpose of our project, **"BLUD - The Blood Donation App"** is to develop a mobile application that facilitates blood donation by connecting donors with those in need based on location and through WhatsApp integration. The app aims to bridge the gap between blood donors and recipients in an efficient and user-friendly manner.

BLUD utilizes a digital platform to create a network of voluntary blood donors. By incorporating location-based services, the app enables users to search for nearby donors, ensuring timely assistance during emergencies. Additionally, we have integrated WhatsApp functionality into the app, streamlining the communication process between donors and recipients. This feature allows users to directly connect with potential donors through messaging, ensuring a smooth and direct communication channel.

Our mobile app provides an easy-to-navigate interface, making it accessible to a wide range of users, including all age groups and the general public. By employing this interactive and engaging approach to finding blood donors, BLUD aims to raise awareness about the significance of donating blood and emphasizes the importance of saving lives through simple yet impactful acts.

Overall, BLUD seeks to revolutionize the way blood donation is approached by utilizing modern technology to facilitate quick and efficient connections between donors and recipients. By providing a simple yet effective solution, we hope to contribute to the larger goal of ensuring adequate blood availability for those in need, ultimately saving lives and making a positive impact on society.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols and Abbreviations

MVC	Model View Controller Model
API	Application Program Interface
OTP	One Time Password
NoSQL	Not only Structured Query Language
HTTP	Hypertext Transfer Protocol

Chapter 1

Introduction

The "BLUD" project aims to revolutionize the blood donation process by providing a robust and efficient platform that connects blood donors with individuals or organizations in need of blood. The project simplifies the process of finding suitable donors, scheduling appointments, and live tracking of donors to ensure timely and effective blood supply to recipients.

1.1 Objectives

The primary objectives of the "BLUD" project are as follows:

- Develop an efficient WhatsApp donation confirmation system: The platform will send confirmation messages to blood donors via WhatsApp before their scheduled donation, providing essential information and reminders to enhance donor engagement.
- Implement live tracking of donors: The system will enable real-time tracking of available blood donors, allowing blood banks to quickly locate donors and manage their inventory effectively.
- Create a live blood request feed: The platform will display a real-time feed of current blood requests, enabling potential donors to see where their help is needed and encouraging them to donate.

1.2 Motivation

The motivation behind the "BLUD" project lies in addressing the challenges and inefficiencies in the blood donation process. Existing systems often struggle to maintain accurate and up-to-date information about blood donors and recipients, leading to delays and mismatches. By providing a comprehensive and user-friendly platform, the project aims to streamline the process, saving lives through timely blood supply.

1.3 Scope of the Project

The "BLUD" project focuses on:

- Developing a scalable and user-friendly mobile application for blood donors and recipients.
- Utilizing WhatsApp Business API for efficient donor communication and confirmation.
- Implementing real-time tracking of available donors to optimize blood supply management.
- Displaying a live blood request feed to connect donors with recipients promptly.

1.4 Prerequisites for the Reader

To understand the contents of this report, the reader is expected to meet the following prerequisites:

- Software Development Understanding: Basic knowledge of software development concepts and methodologies, including SDLC and requirements gathering.
- Familiarity with Mobile Applications: Understanding of mobile app functionalities and usage on smartphones.
- Knowledge of Blood Donation Process: Awareness of blood donation criteria, blood groups, and the importance of blood donation.
- Programming Concepts (Optional): Basic understanding of programming terminology and concepts.
- Basic Database Understanding: Awareness of databases and data storage concepts.
- General Domain Knowledge: General knowledge related to healthcare, blood donation, and social welfare.

1.5 Organization of the Report

Chapter 2 deals with the Feasibility Study(section 2.4.1), explores the Existing Systems(section 2.1), continues to the Gap Analysis(section 2.2) and then proceeds to the Proposed System(section 2.3), and Requirements Engineering(section 2.4). Chapter 3 deals with the design of the project. The System Design(section 3.1) which provides a basic overview of different modules in the system in sections 3.1.1, 3.1.2, 3.1.3, and 3.1.4, and the system architecture in section 3.1. In section 3.2 of this chapter, we have the detailed design, where the UI design in section 3.2.1, algorithms related to different functions in the second and fourth Modules in sections 3.2.2.2, 3.2.2.4, and 3.2.2.5, and the design of the Database(section 3.2.2.4) and Microservices(section 3.2.2.5) are given. Chapter 4 explores the Tools Used which are explained in section 4.1 and the Implementation of Modules which are shown in section 4.2(module 1 in 4.2.1, module 2 in 4.2.3, module 3 in 4.2.2 and module 4 in 4.2.4). Chapter 5 describes the Testing Strategies Used in section 5.1 which includes Unit Testing in sections 5.1.1 and Integration Testing is in section 5.1.2 followed by the Testing Results in section 5.2 for each strategy given. Chapter 6 contains the Results(section 6.1) of the project. The report is summarised in Chapter 7

Chapter 2

System Study and Requirement Engineering

A reality check is a fundamental starting point for every project. It often entails addressing the following inquiries:
1. The purpose of the project
2. Potential impact of the project
3. Estimated cost

To obtain clear answers to these questions, conducting a thorough system study via literature survey and requirement engineering is essential. This process involves analyzing existing literature, research, and similar platforms to refine the project's purpose and identify unique attributes.

Moreover, understanding the project's impact necessitates gathering and documenting detailed stakeholder requirements during requirement engineering. This helps anticipate the effects of the platform on users, customers, the organization, and society at large.

2.1 Existing Systems

Existing systems refer to the current operational setups utilized to perform tasks and fulfill the majority of the project's objectives. These systems are scrutinized to understand the various approaches and techniques currently in practice, along with their limitations. The following are the existing systems.

2.1.1 Simply Blood App

'Simply Blood' [1] is a mobile application that promotes blood donation with a social media model implementation in interface and services. The app provides E-mail with OTP validation for login.

Advantages

- Provision of certificates for donors
- Pre-selection of date and location for donors

2.1.2 UBlood App

UBlood app [2] is another mobile application that provides details of potential blood donors and blood requests within a defined amount of time. The app had a structured user interface with numerous pages for functionalities and services.

Advantages

- Predefined location selection system
- Tatoo, HIV, Covid vaccine checks and confirmations to ensure the possibility of donating blood

2.1.3 Blood Friends App

Blood Friends [3] is one of the latest mobile applications that contained the linking of social media accounts with the profile in app to provide quicker exposure to blood requests. The app comes with an emergency contact authorization feature that helps in receiving blood in emergency cases.

Advantages

- Lets the user choose request type (Plasma, Hospital, Blood Banks, etc.)
- Donation of Plasma and Platelet

2.1.4 Literature Review

A literature survey is a comprehensive study of the existing systems and their details related to the topic of the project. The literature review surveys scholarly articles, books, and other sources relevant to a particular area of the project. The review should enumerate, describe, summarize, objectively evaluate and clarify the existing system and it helps to form a set of requirements for the new system which we propose by analyzing the gaps in the existing system. Following is an explanatory website for undergoing a study of conditions for a person for donating blood.

2.1.4.1 National Blood Transfusion Council

The website of NBTC [4] which is concerned with the Ministry of Health and Family Welfare, Government of India, gives a detailed matter on the eligibility requirements for a person to donate blood. It provides complete details of the eligibility criteria for people in all conditions like pregnancy period, blood pressure, childbirth, post-vaccination, etc. This universal data is provided by the Government of India so as to acquire efficient blood donation.

Advantages

- Describes each and every eligibility requirement for blood donation.
- Contains a contact support system to facilitate the user to provide feedback. or ask any queries regarding blood donation.
- The website also contains the data of complete procedure of a blood donation process

2.2 Gap Analysis

Existing Systems	Analysis
Simply Blood App	The app had delay in generating OTP for login services. However,after login there are errors in the live updating of blood requests and the app had a poor notification system.
UBlood App	Although the app has location selection system, there are multiple bugs while providing location information. The app is also reluctant to find the status of blood banks in specific areas.
Blood Friends App	The number of hospitals and blood banks listed in the app is very less and doesn't provide status of local hospitals and small scale blood banks in an area.

Table 2.1: Gap Analysis of Existing Systems

2.3 Proposed System

The system proposed in this project is a user-friendly mobile application that supports and promotes blood donation among the people. Differentiated from the existing systems, the app facilitates alert messaging system through Whatsapp of the donor or receiver. Alert notifications for blood requests are provided from the app through WhatsApp to the phone number of the potential donor.To be more specific, the message is sent to only donors that had not donated blood in the previous 6 months of the date and not suffering from other health conditions, if stated in the profile of the app. The donor and receiver can contact through voice call from the app if they're in agreement of the blood donation and update the status of donation in the app. The user can also view live requests in the app interface to accept the blood request directly from the app if applicable.

Advantages

- Live updation of blood requests
- Efficient WhatsApp notification system
- Minimizing the time to find a potential donor
- Enabling direct communication with potential donors to increase efficiency and reduce the need for broadcast requests through other groups
- Simplifying the blood donation process for donors
- Utilising GPS technology to track live location of the donor
- The app will be utilisable for blood banks, hospitals, donor and requestors who register with the system

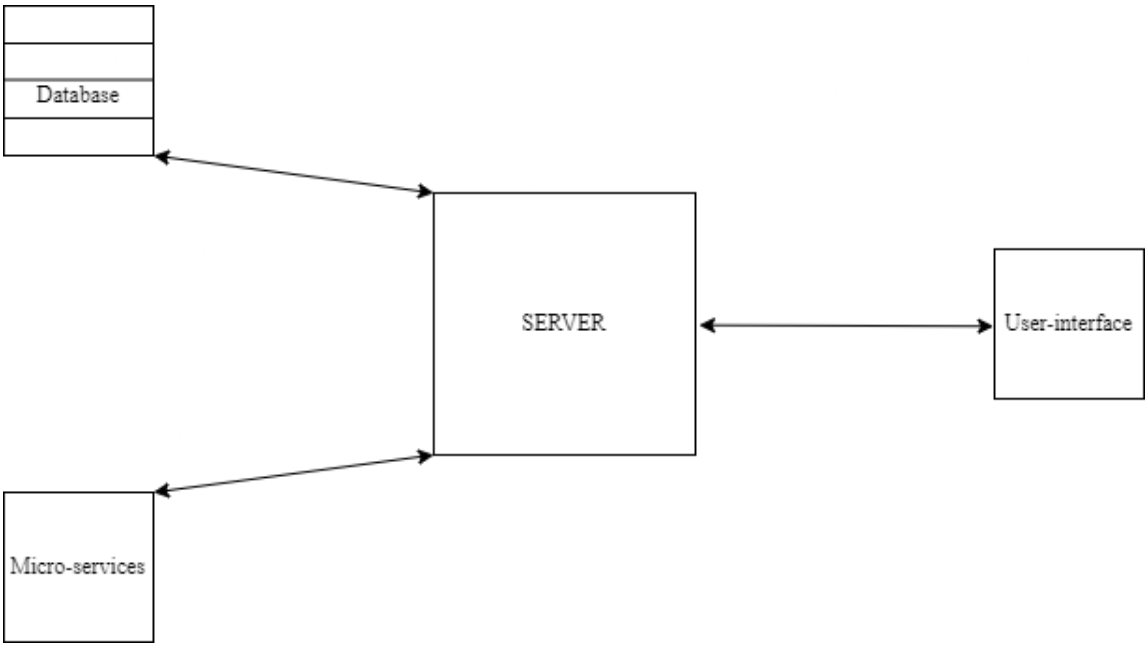

Figure 2.1: Architecture diagram

Disadvantages

- The user should have a WhatsApp account linked to the registering phone number

2.3.1 Problem Statement

To develop a user-friendly and efficient mobile application for blood donation activities with direct alerting and confirmation system and to promote blood donation as a social service among the people.

2.3.2 System Model

The system mainly consists of App interface, server, database and micro-services. The user interacts with the app interface which consists of functional activities of blood donation. The server acts as the main domain which receives inputs from the user and runs the operations accordingly by accessing the database and the micro-services which makes the extraction of information faster.

2.4 Requirements Engineering

The process of gathering software requirements from clients and then analysing and documenting them is known as requirements engineering. The goal of requirement engineering is to develop and maintain a sophisticated and descriptive 'System Requirements Specification' document. It essentially involves the following five activities.

1. Feasibility Study
2. Requirements Elicitation
3. Requirements Analysis

4. Requirements Specification
5. Requirements Validation

2.4.1 Feasibility Study

Feasibility check [5, 6] is a procedure that identifies, describes and evaluates the proposed systems and selects the best system for the task under consideration. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost-effective from a business point of view and if it can develop given existing budgetary constraints. The key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility

2.4.1.1 Economic Feasibility

The most commonly employed technique for assessing the efficacy of a system is economic analysis [5, 6], widely recognized as Cost/Benefit Analysis. Within this approach, specific measures are employed to identify the projected benefits and savings associated with the system and subsequently contrast them against the incurred costs. The outcomes of this comparison are then analyzed and adjusted, if necessary. This iterative process continuously enhances accuracy throughout the system's life cycle. Should the benefits prove superior to the costs, decisions are taken to proceed with the system's design and implementation. Conversely, if the costs outweigh the benefits, further justification or alterations to the proposed system are considered.

The system proposed in this project utilizes free application and development tools that are free of cost. The devices that run the application are commonly possessed by people, like a smartphone or tablets.

2.4.1.2 Technical Feasibility

Technical Feasibility [5, 6] focuses on evaluating the capabilities of the current system to accommodate the proposed additions. For instance, if the existing system is already operating at 80% capacity, introducing another application might overload it, necessitating additional hardware and software upgrades. This integration of financial considerations with technical improvements becomes crucial. If budgetary constraints are significant, it may render the project to improve the current system unfeasible.

The system's feasibility is supported by the fact that it utilizes hardware already owned by end users, eliminating the need for additional setups. The software and associated assets are developed using well-tested tools known for their reliability, further enhancing the project's technical feasibility.

2.4.1.3 Behavioural Feasibility

This aspect is commonly referred to as *Operational Feasibility* [5, 6]. People often exhibit resistance to change, and introducing new systems can facilitate such changes. Consequently, it becomes crucial to comprehend the efforts needed to educate and train users when implementing the proposed system.

This system uses an interactive interface to the end user which can be understood quickly and easily use the application provided. Since the software is run on devices already owned by them, an extra level of familiarity is offered.

2.4.2 Requirements Elicitation

Requirement elicitation begins with the collection of requirements from the stakeholders. A stakeholder is generally a person or group of people directly or indirectly concerned with the software.

1. *Study of existing systems and gap analysis*: The existing systems were studied as part of the literature survey. For each of the systems, we analysed the pros and cons and these are given in section 2.1 with a gap analysis in section 2.2.

2. *Survey and interviews*: Informal interviews were held with various representatives from the medical field and common people who are eager to involve in blood donation activities. Most of the people pointed out the difficulty of finding the right donor before the required time or finding them as early as possible, even through the vast spreading of blood request messages to numerous social media groups and associations. A short survey was conducted to find anomalies in existing apps that were used by users to improve the efficiency of the project.

3. *Through self logic*: We used an iterative approach of self logic which helped in gaining more insights and requirements and develop a solution that improves the aspirations of the end-users and stakeholders. This approach also fostered collaboration and engagement between the project team and end-users.

2.4.3 Requirements Analysis

Requirements Analysis [5, 6] defines the user expectations for developing or modifying an application. It involves various tasks aimed at identifying the diverse needs of stakeholders. As such, requirements analysis entails analyzing, documenting, validating, and managing software or system requirements.

Through the examination of data gathered from interviews and the study of existing systems, we extracted valuable information. This information played a pivotal role in shaping the app's development and determining the requirements.

1. *WhatsApp messaging*: Direct alert messages to the user's Whatsapp number on blood requests that can be donated by the person in his/her current location. The potential donor can accept the request directly through WhatsApp or contact the requestor through the app and accept the request.

2. *Real-time location*: The real-time location of the donors can be tracked through the app to facilitate quick and efficient blood exchange operations through direct/indirect contact between the user and potential donor. The closer the donor, the quicker the donation can be performed.

2.4.4 Requirements Specification

When a requirements specification is written, it is important to remember that the main goal is to deliver the best product possible, and not to produce a perfect requirements specification. There are many good definitions that describe how a requirements specification should be written, but all have at least one part in common, which is the essence of requirements specification, namely the requirements. Requirements are divided into *functional requirements* and *non-functional requirements* [5, 6]. Functional requirements describe the functionality of the desired system that usually consists of some kind of calculation and results, given specific inputs. Non-functional requirements [5, 6] describe how quickly these calculations should be and how quickly the system will respond when its functionality is used. The SRS [5, 6] document is given in the Appendix.

2.4.4.1 Functional Requirements

Functional requirements [5, 6] refer to the essential product features that developers must incorporate to facilitate users in accomplishing their objectives. These requirements outline the fundamental system behavior under all conditions. The functional requirements of the project are:

- Initial User registration for new users to the app
- One-time Password validation by phone number provided by the user
- Details of blood banks and hospitals with addresses to be stored in the database to provide access quickly
- Location information of the user
- Pinpoint tracking of the donors
- WhatsApp integrated alert/notification system

2.4.4.2 Non-functional Requirements

Non-functional requirements impose constraints on the implementation of the functional requirements.

Interface requirements

- Pages and flow of the app must be interactive and good looking with minimal designs
- Menus and interfaces must be easy to understand

Software Requirements

- The app must be connected with a stable network for live services.
- The app must have access to storage, GPS, messages and call functions.

Hardware Requirements

- The app should be utilizable on all range of smartphones
- Required RAM of 2 GB or above

Security Requirements

- Two factor authentication to login the app for users
- The app should be certified with SSL(Secure Sockets Layer)
- End-to-end encryption to be enabled in the app to provide complete privacy to calls and messages between the donors and receivers.

2.4.5 Requirements Validation

Requirement validation is done to make sure that the requirements are complete and consistent according to user requirements. It ensures that requirement specification meets prescribed quality standards. An organised manual analysis of the project's needs was carried out. The app's ability to find donors quickly and carry out the operations related to alerting, contacting and confirming the donations were validated. Non-functional requirements like interface requirements, software requirements, hardware requirements and security requirements were also checked.

2.5 Summary

In this chapter, we discussed the systems that are existing which operates on blood donation, their drawbacks, performance issues, as well as the system proposed as part of the project. We discussed requirements engineering which consists of the feasibility study, requirement elicitation, analysis, specification and validation processes in which we examine the different methods for collecting the requirements and the standard document specifying the requirements. The chapter ends with the set of functional and non-functional requirements of the project and its validation.

Chapter 3

Design

The design phase serves as a vital link between recognizing challenges and crafting solutions in software development. It navigates us from identifying project requirements to outlining a well-defined strategy for fulfillment.
Its pivotal role cannot be overstated, significantly shaping the software's ultimate quality and influencing subsequent stages like testing and maintenance, thus determining project success.
At the phase's culmination, a comprehensive blueprint emerges—a guiding document for implementation, testing, and maintenance. This blueprint divides the design process into:

- System Design
- Detailed Design

These distinct design stages will be elaborated upon in subsequent sections, offering insight into their objectives and processes, aiding a comprehensive grasp of effective software development planning.

3.1 System Design

System Design aims to identify the modules that should be present in the system, the specification of these modules and how they interact with each other to produce the desired results. At the end of the system design, all the major data structures, file formats, output formats and major modules in the system and their specifications are decided.

In our app design, the application is split into a series of modules, each focusing on a typical app's main part. The main modules are:

- **UI**: This module handles the app's user interface, including the design and layout of the user interface elements.
- **Server**: This module is responsible for the app's backend logic, handling requests from clients, and managing data processing.
- **Database**: The database module deals with data storage and retrieval, managing the app's data and ensuring data integrity.

In addition to these main modules, we also have a microservice dedicated to implementing third-party features. This microservice handles integration with external services or APIs to provide additional functionalities to the app.

By organizing the app into these modules, we ensure a more structured and maintainable design, with each module focusing on its specific role and responsibility.

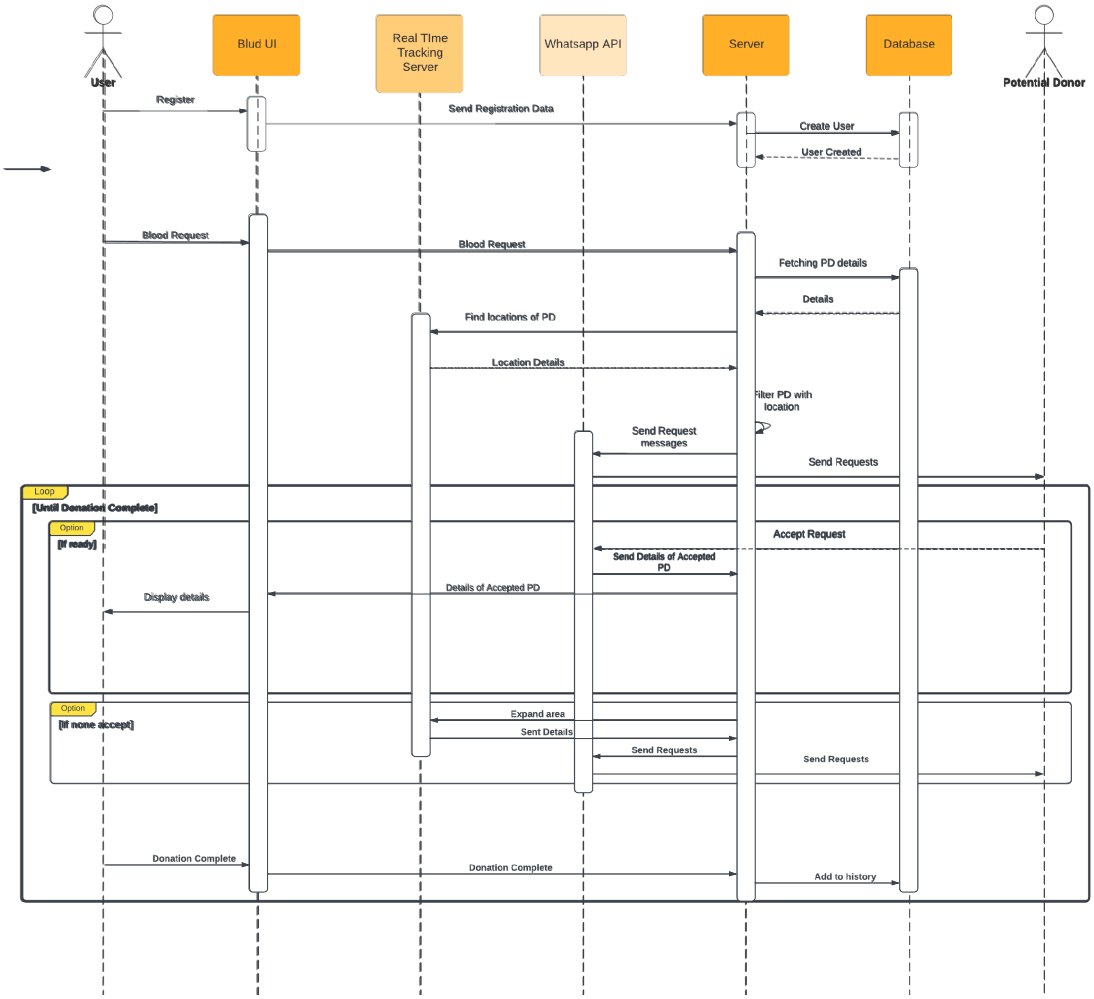

Figure 3.1: Sequence Diagram

3.1.1 BLUD UI

The UI module is responsible for interacting with the user. Its design is tailored to fit the app's theme, ensuring a cohesive and visually appealing experience for the users. The flow of pages within the app follows the process of requesting blood, with the UI module handling the modification and beautification of these pages.

In essence, the UI module ensures that the app's user interface is user-friendly, intuitive, and aesthetically pleasing. It guides users through the blood request process, making it easy for them to navigate and interact with the app effectively. Additionally, the module focuses on enhancing the visual elements and layout to create a seamless and enjoyable user experience. The main parts of the include:

- **Login**: This part enables users to log into the app, providing their credentials such as username/email and password to access their accounts.
- **Signup**: In this section, new users can create accounts by providing necessary information like their name, email, password, and other relevant details.

- **OTP Verification**: After signup or certain actions, the app may require users to verify their phone numbers or email addresses through a One-Time Password (OTP) sent to them for security purposes.
- **Profile Edit**: Users can modify and update their profiles, including personal information, contact details, and any other relevant data.
- **Blood Request**: This part allows users to make requests for blood when they are in need. They can specify their blood type, location, and other relevant information to connect with potential blood donors.
- **Accept Request**: When someone requests blood, this section allows potential donors to review the request and decide whether to accept or decline it.

3.1.2 Server

Server serves as the backbone of the app, playing a crucial role in handling various tasks. It receives data from the UI module, processes it accordingly, and then stores the processed data in the database. Some of the key processes that the server handles include:

- **Creating User and Storing in Database**: When a user signs up or registers through the UI, the server processes the provided information, creates a new user account, and stores the user's details in the database for future reference.
- **Blood Request Creation**: When a user submits a blood request via the UI, the server processes the request, validates it, and creates a new entry for the request in the database. This entry contains relevant details such as the blood type, location, and other essential information.
- **Profile Edit**: Whenever a user edits their profile through the UI, the server receives the updated information, processes it, and modifies the corresponding user data in the database.
- **Interacting with Microservices**: The server is responsible for interacting with various microservices integrated into the app. These microservices could be third-party features or external APIs that provide additional functionalities to the app. The server manages communication with these microservices to fetch or send data as needed.

The server is designed to be scalable, meaning it can handle increased traffic and user activity without compromising performance. This scalability is crucial to accommodate the app's growth and ensure a smooth experience for users even during peak usage periods.

Moreover, the server is developed following the Model-View-Controller (MVC) architectural pattern. The MVC model allows for a clear separation of concerns, making the server's codebase more organized, maintainable, and easier to extend or modify in the future. The Model represents the data and business logic, the View handles the UI and presentation layer, and the Controller manages the communication and flow between the Model and View components. This design pattern enhances code reusability, promotes clean code practices, and facilitates collaboration among developers working on the project.

3.1.3 Database

Database is a crucial component in any data-driven app, and in this case, a scalable NoSQL database is being used to store the app's data. To optimize data management, two distinct databases are employed:

- **Static Data Database**: This database is specifically dedicated to storing static data, such as configuration settings, reference data, or any data that remains relatively unchanged throughout the app's operation. By using a separate database for static data, the app can efficiently retrieve this information without affecting the performance of real-time data operations.
- **Real-time Data Database**: The second database is utilized for storing real-time data, including user-generated content, live updates, and dynamic information that constantly changes based on user interactions and app activities. A NoSQL database is well-suited for handling this kind of data due to its flexibility and scalability.

Both databases are designed with well-built schemas tailored to the specific use cases of the app. A well-designed schema ensures efficient data organization and retrieval, supporting the app's functionality without compromising on performance. Additionally, the schemas are optimized to handle the app's data needs and allow for seamless scaling as the user base and data volume grow.

By segregating static and real-time data into separate databases with optimized schemas, the app can maintain high performance and responsiveness, providing users with a smooth and enjoyable experience while ensuring the scalability to accommodate future growth.

3.1.4 Microservices

The app's microservices include:
Real-Time User Tracking Service: Tracks users in real-time to enable features like locating blood donors and monitoring blood requests.

Messaging Service: Facilitates seamless communication between users, allowing them to exchange information related to blood requests and coordinate donations.

3.1.4.1 RealTime tracking

Real-time tracking is achieved by periodically obtaining the user's location every 15 minutes. The user's location data is then stored in a real-time database, associated with their unique user ID. This approach ensures that the app can continuously track the user's movements and provide up-to-date location information for various functionalities, contributing to a seamless and dynamic user experience.

3.1.4.2 Whatsapp API

By utilizing the API provided by the platform, the app can directly send request messages to the user. This functionality effectively assists in finding potential blood donors. When a blood request is made, the app leverages the API to reach out to potential donors and notify them about the urgent need for blood. This direct communication streamlines the process

and increases the chances of finding suitable donors quickly, contributing to more effective and timely assistance for those in need of blood donations.

3.2 Detailed Design

In this detailed design, we will delve deeper into the designs of each module, including the tasks performed by the Server, UI, Database, and Microservices. We will provide comprehensive explanations for each module, accompanied by flowcharts and algorithms to illustrate their functionalities.

3.2.1 BLUD UI

The UI design is the most critical aspect of the app since it directly interacts with the users. The visual appearance and user experience of the UI design are essential for ensuring a successful app.

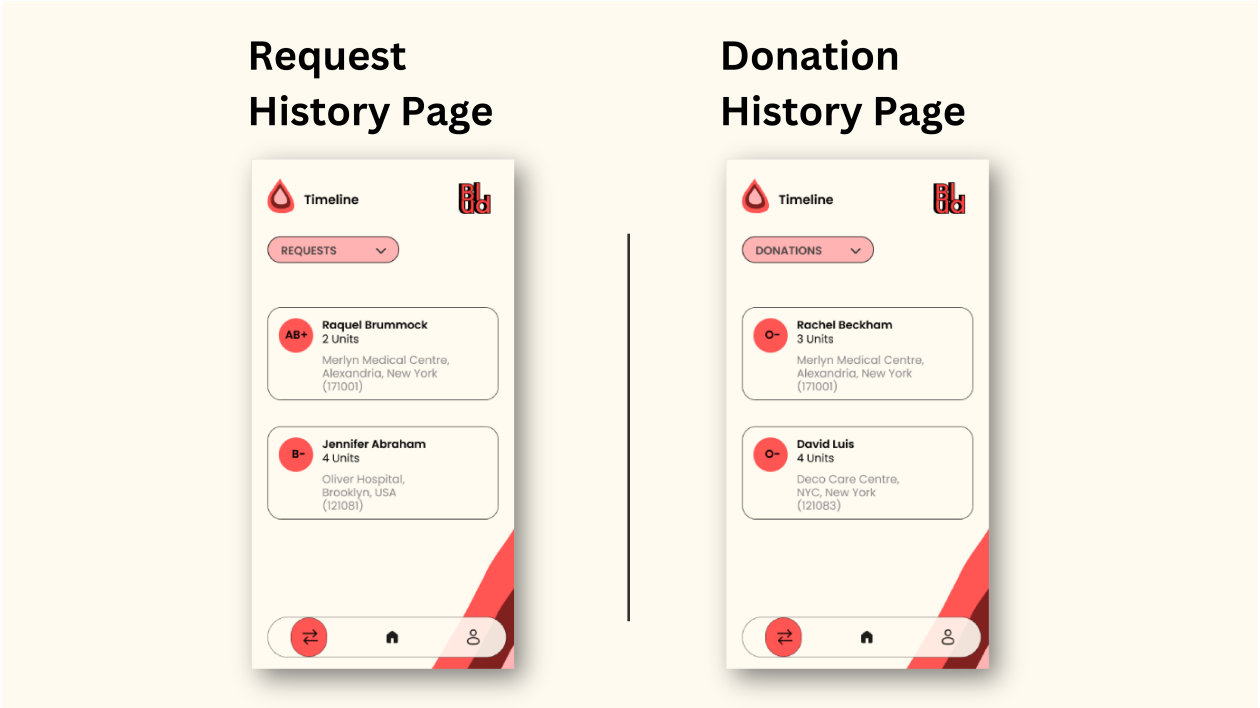

Figure 3.2: History page

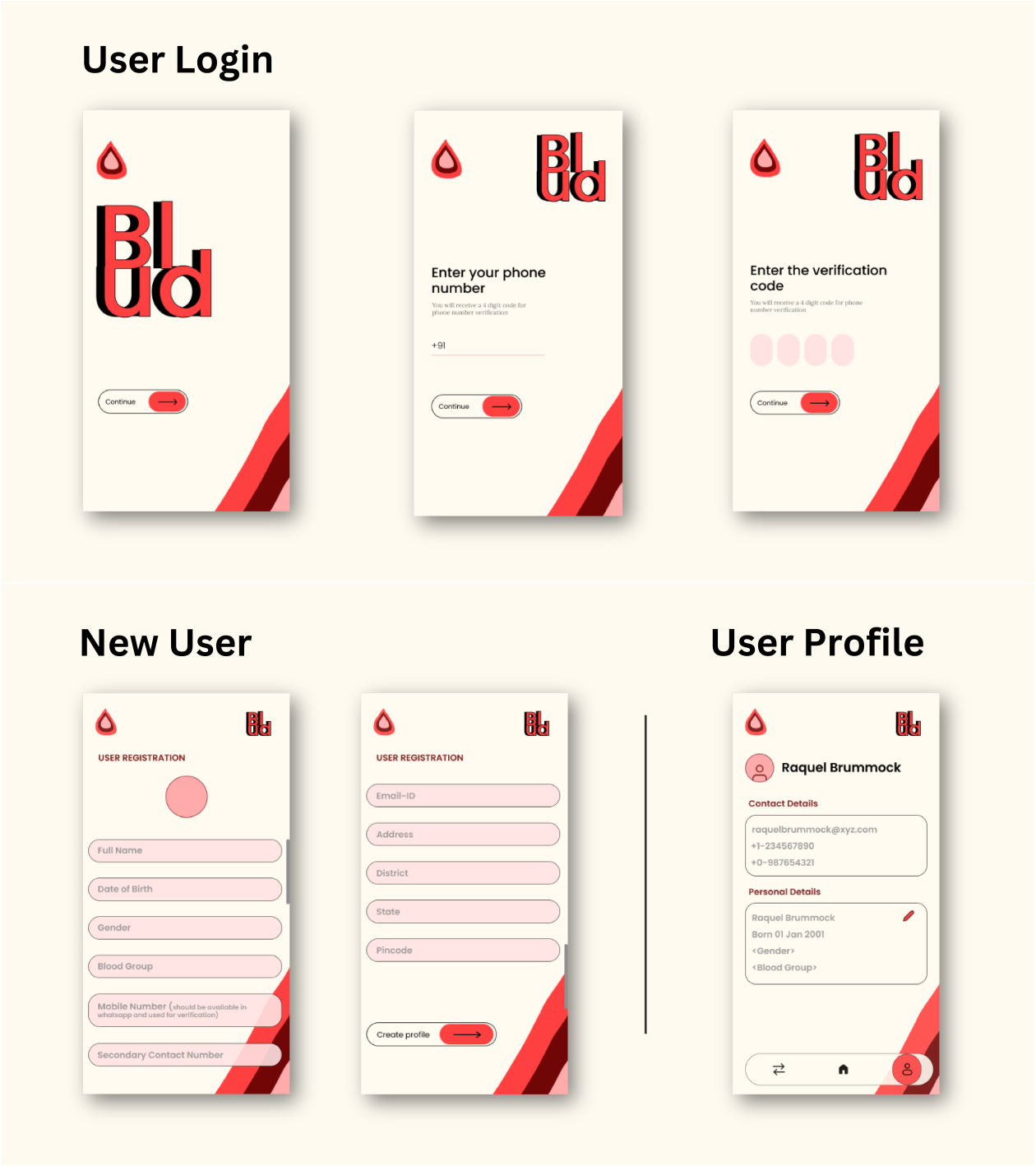

Figure 3.3: User login and create page

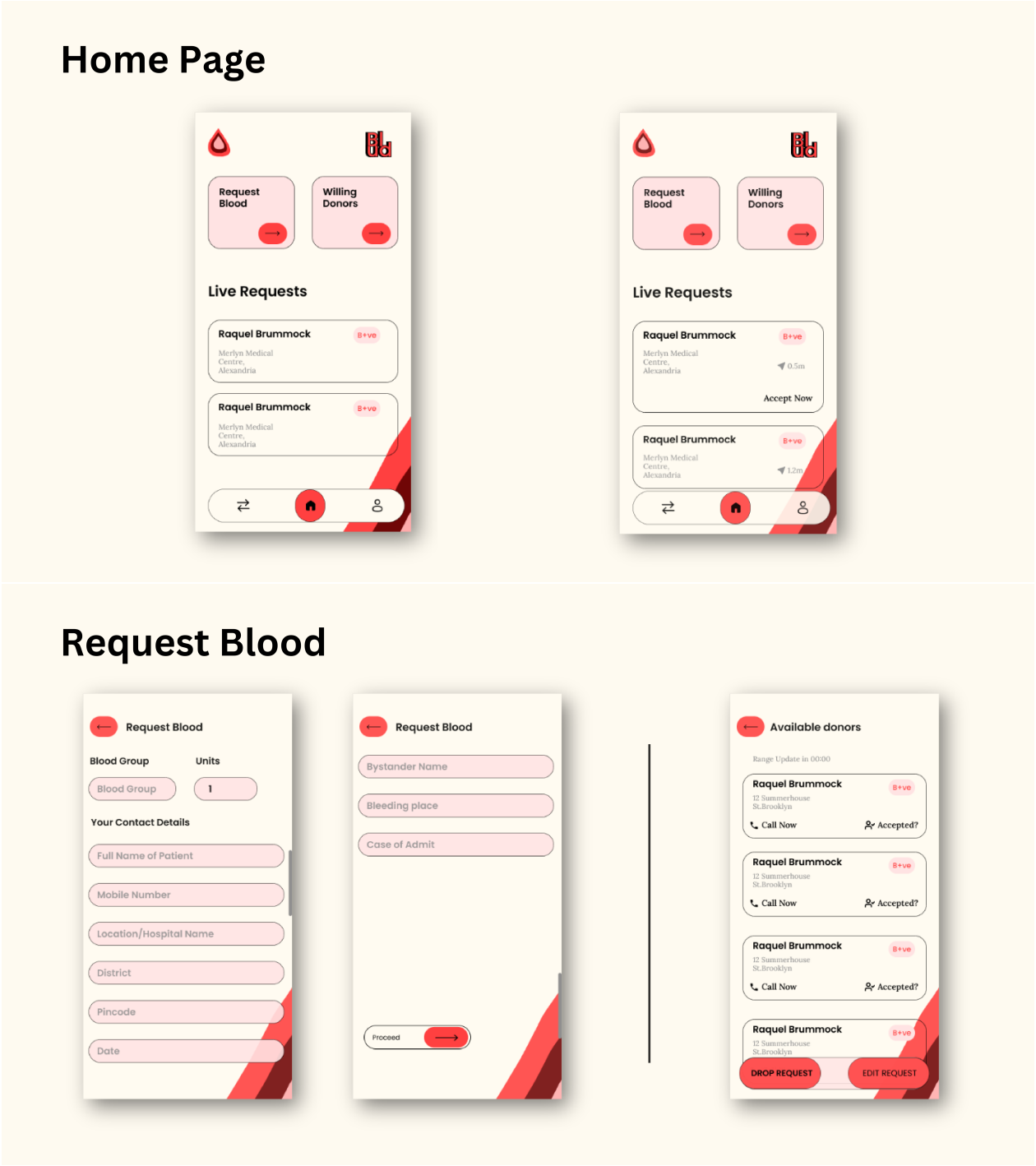

Figure 3.4: Home and Blood request page

3.2.2 Server

The Server serves as the backbone of the app, performing the main processing tasks. There are several tasks handled by the server, and we will discuss each of them separately to provide a comprehensive understanding of its functionalities.

3.2.2.1 Sign In/ Sign Up

The fundamental functionalities of the app include user registration and security. To achieve this, we implement an OTP authentication model to verify users during the registration process. Each time a user attempts to log in, the app utilizes the OTP mechanism for authentication. If the user does not exist in the system, we will create a new user account for them. This approach ensures that only authorized users can access the app, enhancing security and user data protection.

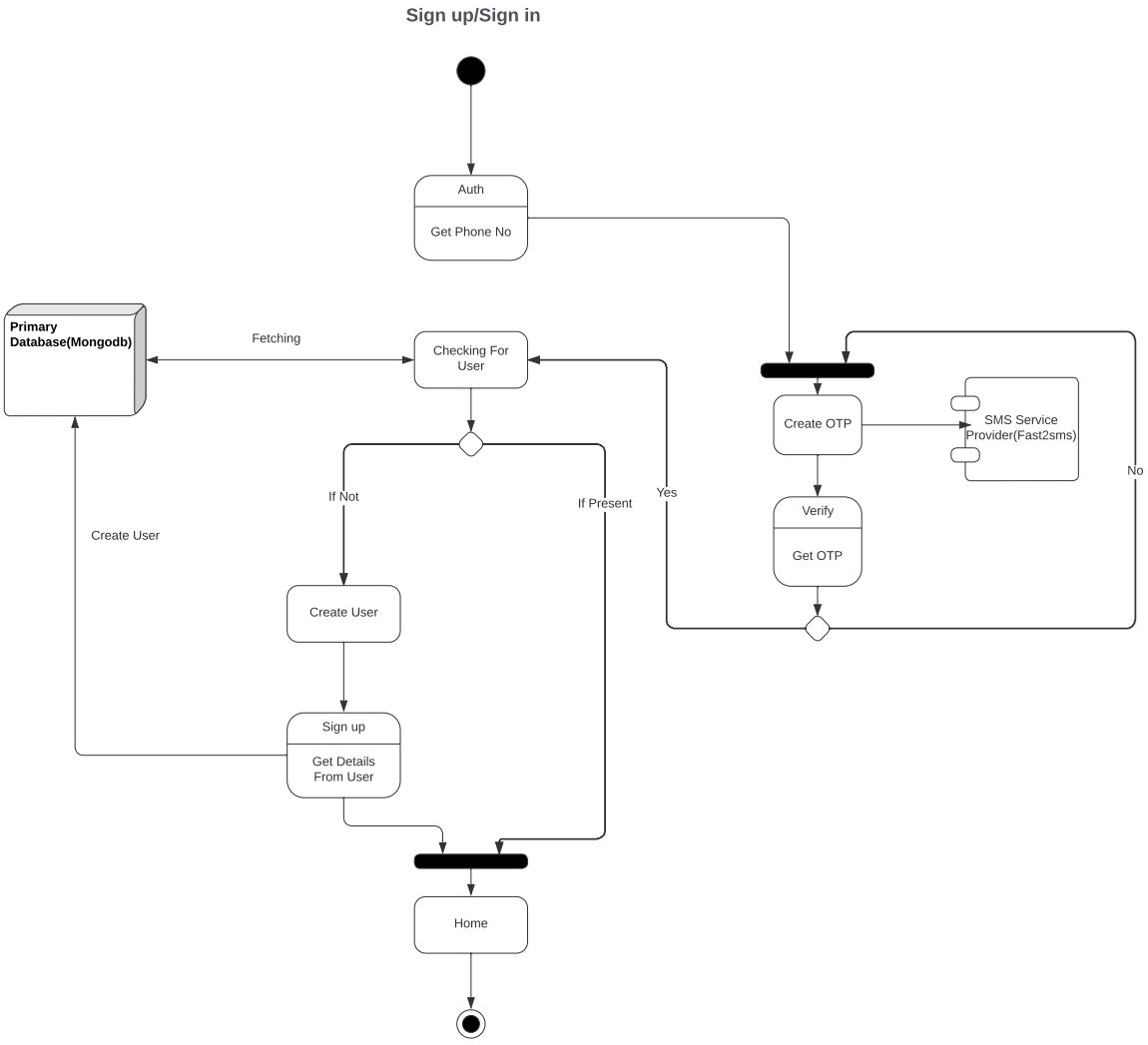

Figure 3.5: Sign In/Sign up flow chart

3.2.2.2 Available Request

One of the main functions of the app is to display available blood donation requests to users based on their location. Users can view the requests from nearby locations and choose to donate if they are eligible. If a user is not eligible to donate for any particular request,

Algorithm 1: Algorithm for Sign In/Sign up

Data: Phone number and OTP
Result: User present or not

1 Start
2 Get the Phone Number
3 Create OTP
4 Send OTP with SMS service provider
5 If OTP is not verified go to Step 3 otherwise go to Step 5
6 If the User is present with this phone number go to Step 7 otherwise go to Step 6
7 Create the User with essential details and store them in the User database
8 Go to the Home page
9 Stop

the app will provide relevant information accordingly. This feature facilitates effective blood donation coordination and ensures that donors can easily find and contribute to the requests that match their eligibility and location.

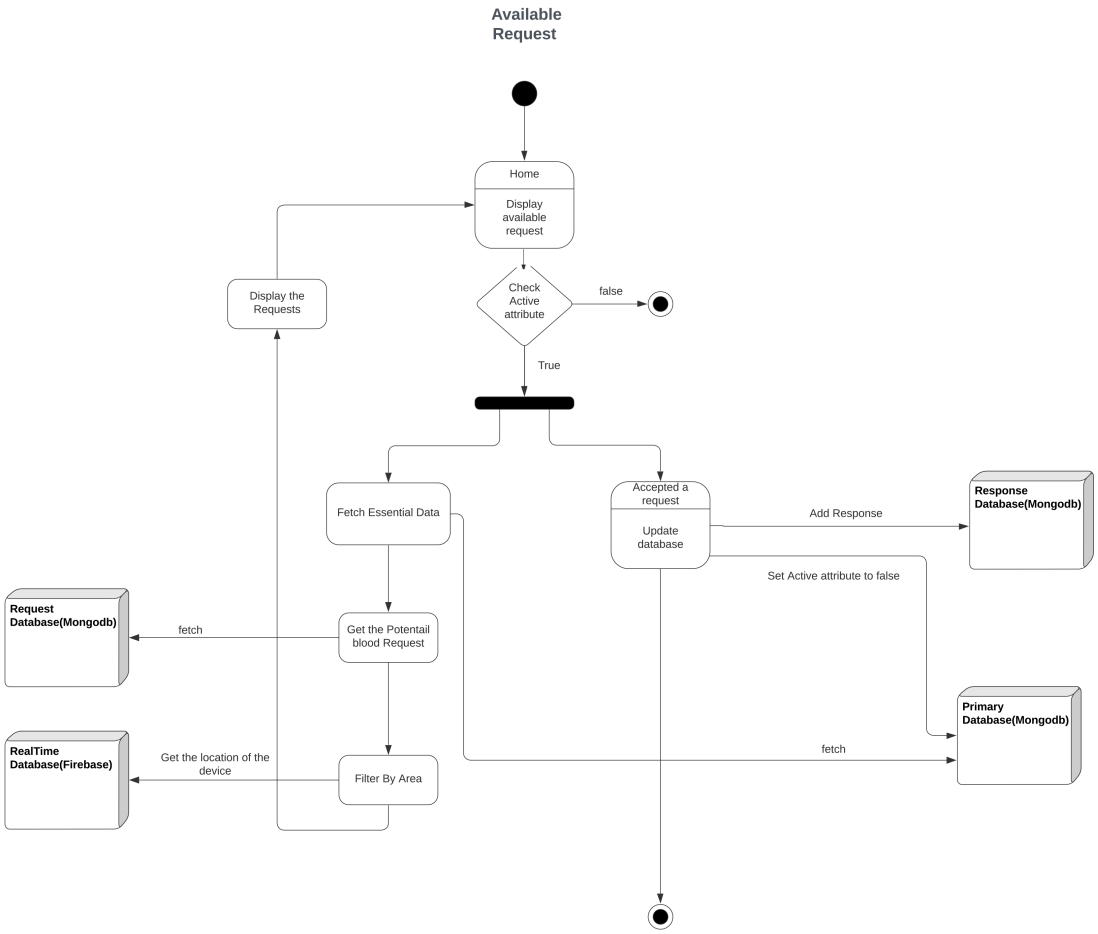

Figure 3.6: Available-request flow chart

Algorithm 2: Algorithm for Available-request

Data: Request Chosen
Result: Successful or Not

1 Start
2 If User.Active = false, go to step 9 otherwise go to Step 3
3 Get the User data
4 Use the User data to find potential requests
5 Filter the request by area by fetching real-time device location from RealTime database
6 Display the requests
7 Create the User with essential details and store them in the User database
8 Go to the Home page
9 Stop

3.2.2.3 Blood Request

In Blud, requesting blood is the primary and essential feature of the blood donation app. To enhance communication and improve the donor finding process, we leverage WhatsApp as a means of interaction. This integration allows us to seamlessly connect with potential donors. We use a real-time tracking module to identify the nearest available donors in the vicinity of the blood requester. Once potential donors are identified, the person making the blood request can choose a suitable donor and directly contact them through WhatsApp. This approach streamlines the process, enabling quicker and more efficient coordination between blood donors and recipients, ultimately contributing to timely and effective blood donation efforts.

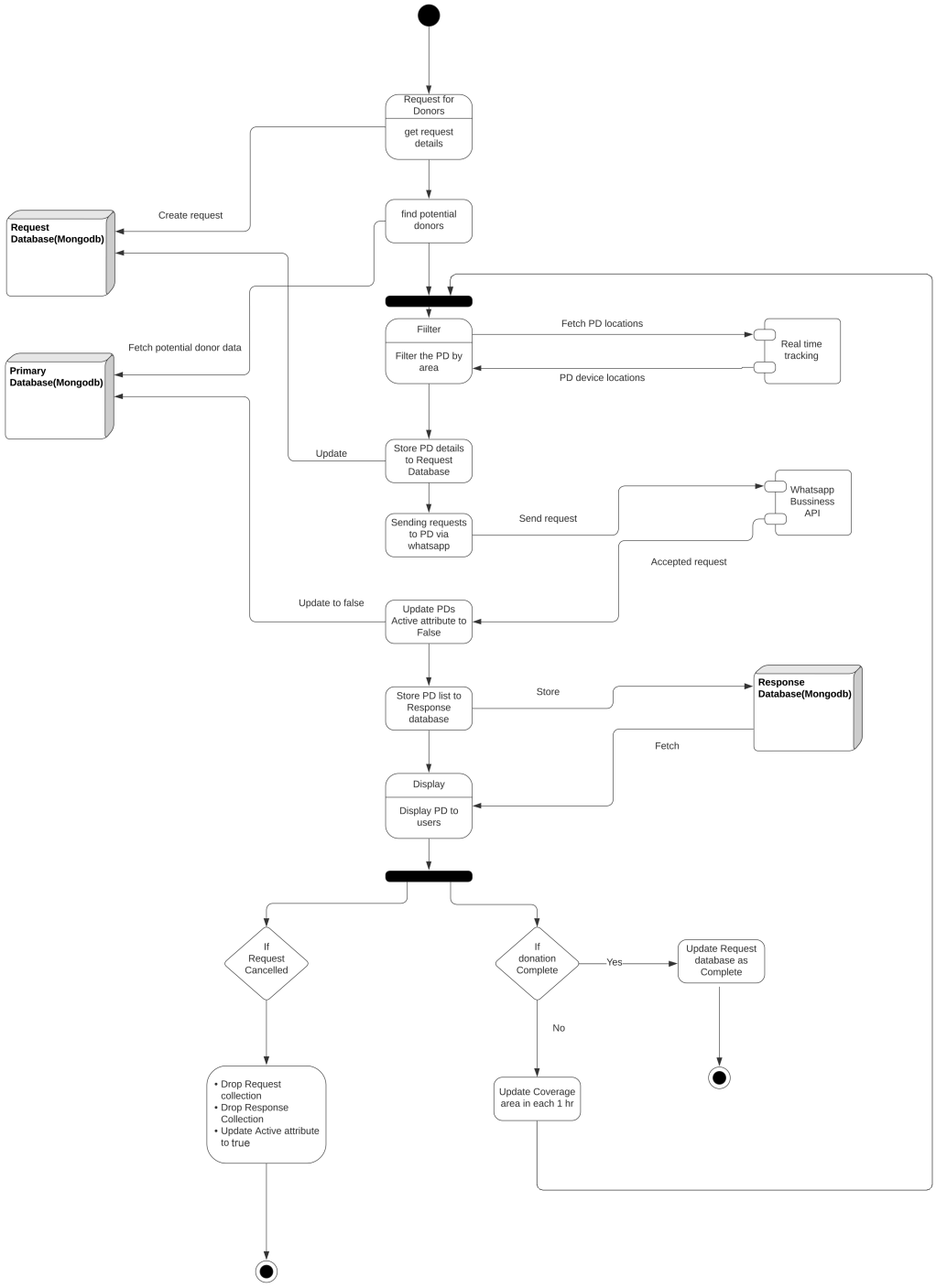

Figure 3.7: Blood request flow chart

3.2.2.4 Database

The Blud app employs two distinct databases for data storage. A NoSQL database is utilized for storing static data, while a real-time database is employed for real-time location tracking. Different schemas are employed for data storage to ensure accurate data retrieval, and to maintain the ACID (Atomicity, Consistency, Isolation, Durability) properties.

Algorithm 3: Algorithm for Blood Request

Data: Request details
Result: List of potential donors

1 Start
2 Get the request details and store them in the Requeset database
3 Find the potential donors from the User database
4 Filter the potential donors with their device location stored in the RealTime database and the location of the request
5 Update the Request database with Potential donors
6 Sent the request to Potential donors via Whatsapp Business API
7 If a potential Donor accepts the request set User.Active as false
8 Store the request accepted donor list in Response Database
9 Display the request accepted potential donors to the User
10 If the request gets canceled drop Request collection, response collection, and User.Active to true and go to Step 14
11 If the Donation is Complete go to Step 12 otherwise go to Step 13
12 Update the request database as complete and go to step 14
13 Update coverage area in each 1 hr and go to Step 4
14 Stop

By segregating static data and real-time location information into separate databases with appropriate schemas, the app can efficiently manage data and maintain data integrity. The NoSQL database is ideal for handling static data, such as user information and configuration settings, while the real-time database efficiently manages dynamic location data to facilitate real-time tracking of users. This architectural design ensures optimized performance, scalability, and reliability, enhancing the overall user experience and ensuring data consistency across the application.

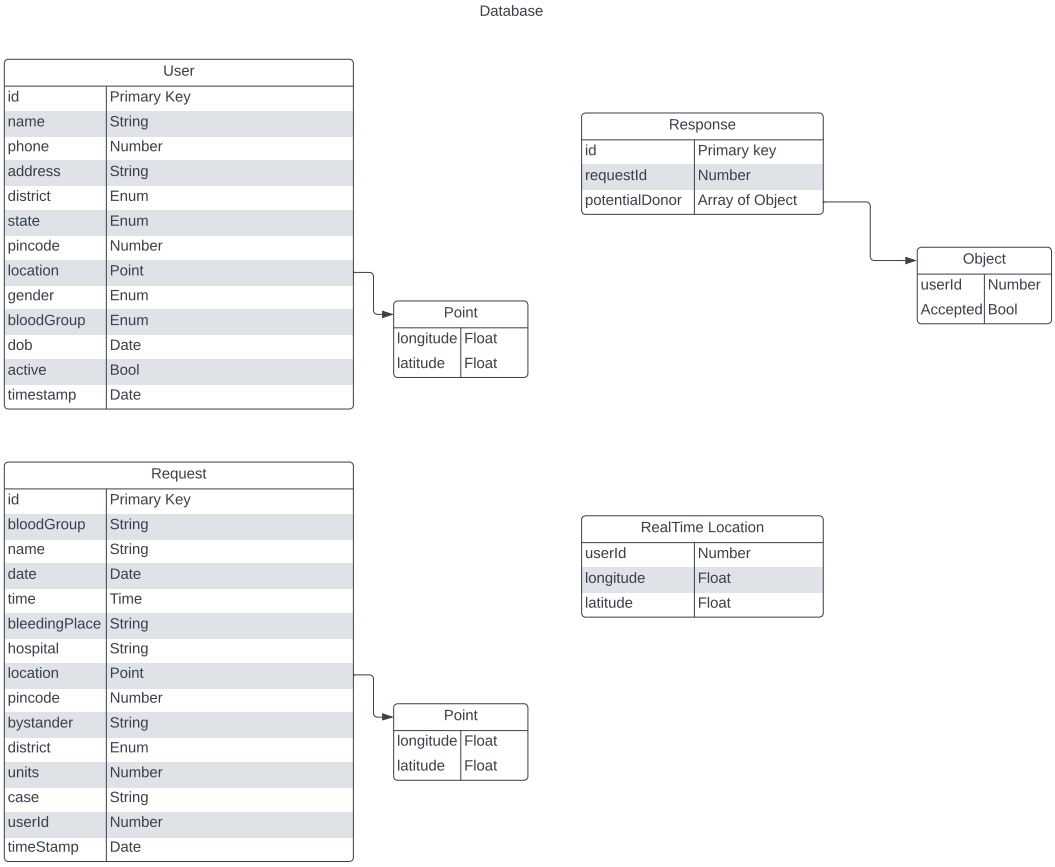

Figure 3.8: Database Schema

3.2.2.5 Microservices

Blud effectively utilizes two essential microservices that play a critical role in its functioning. The first microservice integrates with the WhatsApp API, enabling seamless message exchange and facilitating direct communication between users. This feature proves vital for efficient coordination between potential blood donors and recipients, enhancing the overall effectiveness of the system.

The second microservice involves real-time tracking, which is pivotal for the app's success. This microservice periodically captures and stores the location of devices every 15 minutes in the real-time database. By continuously updating user locations, the app can provide real-time tracking functionalities, optimizing the process of finding the nearest available blood donors for urgent requests.

Both microservices contribute significantly to Blud's operations, ensuring smooth communication and precise real-time tracking, key components that enhance the app's effectiveness in connecting those in need of blood donations with potential donors.

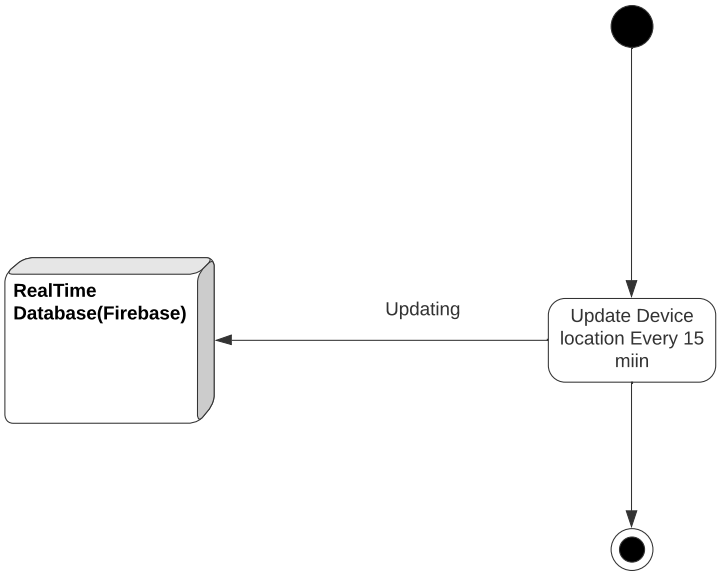

Figure 3.9: Real Time tracking flow chart

Algorithm 4: Algorithm for Real-Time Tracking

Data: Location of the device
Result: Success or failed

1 Start
2 Loop
3 Get Device Location and Update RealTime database every 15 min
4 End loop
5 Stop

3.3 Summary

In this chapter, we delve into the design of all modules, including the system design, four main components: Server, UI, Database, and Microservices. The Server handles tasks like user authentication, blood request processing, and Microservices interaction. The UI module enables user registration, blood request submission, and profile editing. The app employs two databases: NoSQL for static data and real-time for location tracking. Microservices integrate WhatsApp API for communication and real-time tracking of potential donors based on location, ensuring efficient blood donation coordination.

Chapter 4

Implementation

The development or implementation stage [5, 6] is the stage where the development of the code and modules are performed. The development and integration of the units and creation of the project build are done according to the design documents' specifications. The goal of this phase is to translate the design of the proposed system into a working model. Since this phase affects the testing phase, the coding for the project is performed in such a way as to maintain simplicity and clarity.

4.1 Tools Used

4.1.1 Flutter

Flutter [7] is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. It is an UI software development kit (SDK), enabling developers to create natively compiled apps for mobile, web, and desktop. It offers fast performance, a rich set of customizable widgets, and features like Hot Reload for efficient development. With access to native device functionalities and a supportive community, Flutter has become a popular choice for building cross-platform applications with native-like user experiences.[1]

4.1.2 Figma

Figma [8] brings together powerful design tools with multiplayer collaboration, allowing teams to explore ideas while capturing quality feedback in real-time—or anytime. It Establishes a shared visual language with design systems that anyone across your collaboration can access. It also enables everyone to create consistent and cohesive experiences with shared libraries and reusable assets. Figma allows for working together in real time and empowering designers to create in new ways. It keeps workflows efficient with tools that give every team visibility throughout the process.
Basically it allows to creation a design that acts as a model or plan that the developer builds the app on. [2]

4.1.3 NodeJS

Node.js [9] is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser. Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities.
It allows us to build the logic of the application that runs behind the UI, that is backend of the application. [3]

4.1.4 Visual Studio Code

Visual Studio Code] [10] is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C, Java, Python, PHP, Go, .NET, Dart). [4]

4.1.5 MongoDB

MongoDB [11] is a document database designed for ease of application development and scaling. In MongoDB we can store and query our data, transform data with aggregations, secure access to our data and deploy and scale our database. [5]

4.1.6 Firebase

Firebase [12] is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++. It also provides a realtime database that has a fast store and access operations that allows dynamic data to be stored efficiently. [6]

4.1.7 Postman

Postman [13] is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster. Easily store, catalog, and collaborate around all your API artifacts on one central platform. Postman can store and manage API specifications, documentation, workflow recipes, test cases and results, metrics, and everything else related to APIs. The Postman platform includes a comprehensive set of tools that help accelerate the API lifecycle—from design, testing, documentation, and mocking to the sharing and discoverability of your APIs. Postman integrates with the most important tools in your software development pipeline to enable API-first practices. The Postman platform is also extensible through the Postman API and through open source technologies. [7]

4.1.8 Android Studio

Android Studio] [14] is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.
Android Studio provides the application emulator such as a mobile device with an android operating system on which the developer can run and test their flutter codebase. [8]

4.2 Module Implementation

In this section of our project report, the implementation of each module is explained in detail. Here we discuss several modules such as User Interface, Database, Server and Micro Services. The interaction between each of these modules are important and it forms the backbone of the application.

4.2.1 User Interface

User Interface of our application deals with the implementation and design of pages using Figma and Flutter. The basic user interface of our BLUD app consist of the home page where live donation requests corresponding to the blood group of the user and their status of donation. It also consist two buttons for requesting blood and to see willing donors. The app also consist of pages to view History of both donations and requests. There is also a profile page to edit user details and to logout from the app.

4.2.2 Database

The database module of our BLUD application deals with the storage of data that is crucial to our application. The module is implemented using MongoDB. It facilitates the storage and provision of data such as user details that is collected by UI on registration phase, storing the details of potential donors when a user blood request is active for the UI to access and display, storing the record of all request and donations in a history page for the UI to access. The firebase realtime DB also facilitates the storage of more dynamic realtime data such as willing donors.

4.2.3 Server

Server module in our application was implemented using NodeJS which provided the environment to build the logic based on which the application works. This module deals with integrating the WhatsApp API, checking the location constraints and tracking location in realtime, increase the radius of the location search if the previous search finds unsuccessful, filtering the donor data, registration and login logic, OTP implementation and handling blood requests.

4.2.4 Micro Services

Micro services consist of two important components - WhatsApp API and Realtime Tracking Server. WhatsApp API is basically used to sent the request message to filtered donors from the server through WhatsApp messaging and the allow the user to accept a blood request. This is implemented using the 'Twillio' package in combination with WhatsApp business chat.
The other component is the Realtime Tracking server which is used to find the locations of the potential donors and send the details back to the server for filtering and displaying it to the users. It can be also used to expand the area of search for donors and find more potential candidates for donation.

4.3 Summary

In this chapter, we discussed the implementation of the BLUD application using several app development tools and environments that facilitate the usage of APIs and other packages that allows for effective performance of what the app intends to do. The chapter also explains each modules used in the application that makes front-end, back-end and database communication effective.

Chapter 5

Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is Defect free. It involves the execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

5.1 Testing Strategies Used

In this section, the strategies that were employed to test our project will be discussed. The following list outlines the testing strategies that were utilized in our project to ensure its quality and effectiveness.

5.1.1 Unit Testing

At the heart of our testing regimen lies unit testing—a software testing technique aimed at meticulously scrutinizing individual units or components of the software. Within the codebase, each unit, be it a function, method, or class, underwent rigorous evaluation in isolation. Through a suite of meticulously designed test cases, we verified the correctness of these units and ensured that they produced the desired output. The essence of unit testing lies in independently validating the accuracy and reliability of each code unit. By scrutinizing individual units in isolation, unit testing allowed us to pinpoint and rectify bugs and errors at an early stage of development. To achieve this, we employed two primary techniques for unit testing:

5.1.1.1 Black Box Testing

Among the techniques used was black box testing—an approach that focuses solely on the system's observable behaviors, without delving into its internal workings. Testers evaluated the system based on inputs and outputs, without any knowledge of the underlying implementation or code. In the black box testing phase, we treated the Blud system as a "black box," emphasizing validating its behavior and functionality against specified requirements without concerning ourselves with the intricacies of its implementation

5.1.1.2 White Box Testing

In contrast, white box testing, another crucial technique, delves into the internal structure, code, and implementation details of the Blud system. Armed with this internal knowledge, we designed test cases to explore specific paths, conditions, or branches within the code, enabling comprehensive validation of the internal logic. By gaining insight into the system's internal workings, white box testing enhanced our ability to achieve thorough test coverage and uncover potential logical errors or vulnerabilities in the code.

5.1.2 Integration Testing

5.1.2 Integration Testing Moving beyond individual units, we undertook integration testing to assess the seamless collaboration and interactions between various components of the Blud project. Through systematic integration and testing, we aimed to identify any interfacing-related issues and ensure that all integrated modules functioned harmoniously. Integration testing allowed us to catch and resolve bugs early in the development process, thereby improving the overall reliability and robustness of the Blud system. Additionally, this testing phase expanded our test coverage, adding an extra layer of assurance to the overall quality. By implementing these testing strategies, our Blud project underwent extensive examination, leading to a high-quality and well-tested software application that aligns with our intended requirements. The testing efforts were instrumental in enhancing user satisfaction by delivering a reliable and functional software product.

5.2 Testing Results

5.2.1 Results of Black Box Testing

Black box testing, a crucial part of the testing process, involved examining the external behavior of the Blud system without considering its internal structure or implementation details. Testers focused solely on the inputs and outputs of the system, treating it as an opaque "black box" to assess its functionalities and adherence to specified requirements. The black box testing phase for the Blud project encompassed various scenarios, simulating different user interactions and input combinations. Testers meticulously evaluated the responses and outputs produced by the system to verify its compliance with the expected behaviors. This form of testing ensured that the Blud system correctly processed inputs and generated appropriate outputs, regardless of its underlying implementation.

5.2.2 Results of White Box Testing

White box testing, also known as clear box testing or structural testing, involved a deeper inspection of the internal structure, code, and implementation details of the Blud system. Testers had access to the source code, design documents, and architectural information, which enabled them to design test cases to exercise specific paths, conditions, or branches within the code. During white box testing, the goal was to achieve thorough coverage of the internal logic and control flow of the Blud project. Testers strived to identify any discrepancies, logical errors, or potential code vulnerabilities that might exist within the implementation. By evaluating the system at this level, the team ensured that the underlying codebase was robust, stable, and free from hidden defects.

5.2.3 Results of Integration Testing

Integration testing was a critical phase aimed at assessing the interactions and collaborations between individual components of the Blud project when integrated into a cohesive whole. The primary focus was to identify any issues related to interfacing between modules and to ensure seamless interoperability. Integration testing involved constructing test scenarios where multiple components were combined and tested together. The objective was to validate

that the integrated modules cooperated as expected and that data and information were exchanged accurately between them. This ensured that the various parts of the Blud project worked harmoniously, supporting the overall system's robustness and functionality.

5.3 Summary

Through rigorous testing strategies, including unit testing, black box testing, and white box testing, as well as comprehensive integration testing, the Blud project underwent thorough examination to ensure its quality, functionality, and reliability. The various testing phases helped uncover and resolve potential defects, ensuring that the software product was defect-free and aligned with the expected requirements. By adhering to these testing methodologies, the Blud project achieved a higher level of confidence in its performance and behavior

Chapter 6

Results

6.1 Achievement

The following objectives were achieved through the course of this project:

- Finding potential donors with ease using the WhatsApp
- Helping users to request blood easily

6.2 Become a user

Becoming a Blud user is a simple and secure process. The app employs OTP for login, ensuring enhanced security during authentication. Additionally, the registration process only requires essential details, making it easy and convenient for users to create an account swiftly. This approach strikes a balance between user-friendliness and data protection, making it seamless for users to join the Blud platform and participate in blood donation efforts.

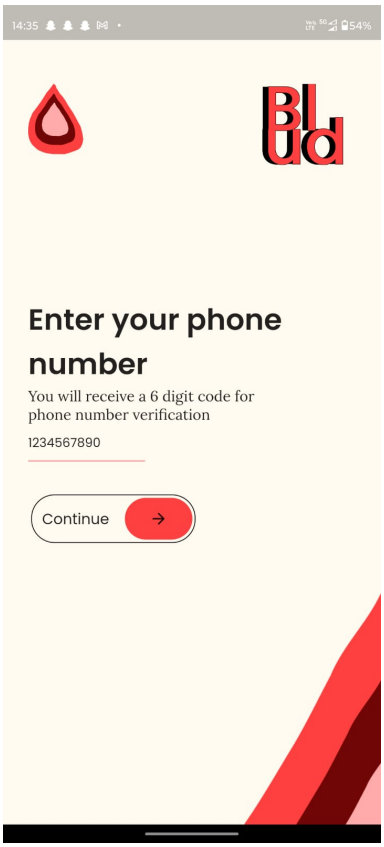

Figure 6.1: Entering the number

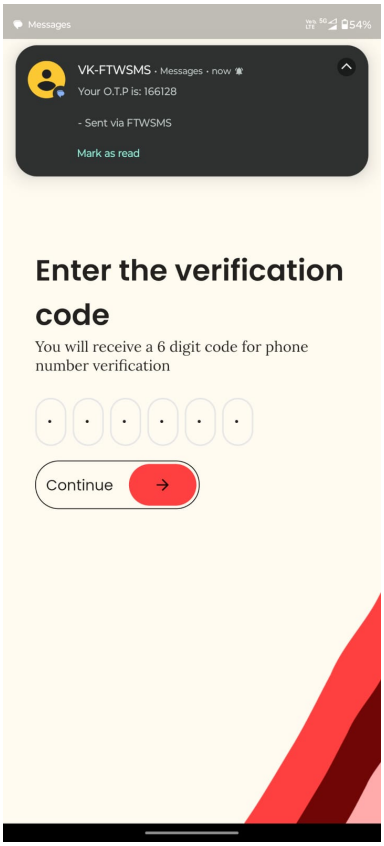

Figure 6.2: OTP verification

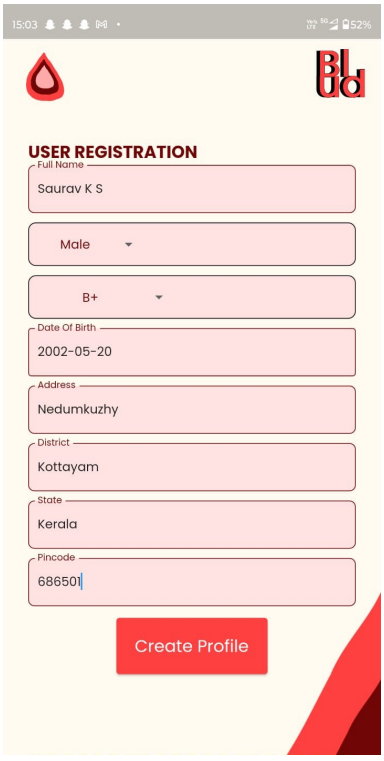

Figure 6.3: Entering User Details

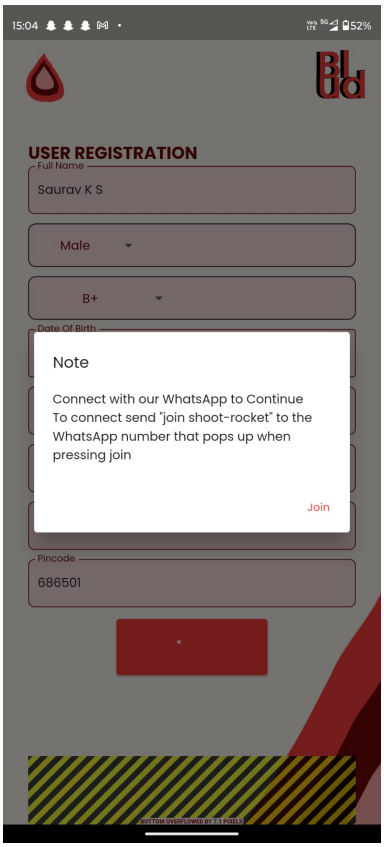

Figure 6.4: Notification for giving access to the Whatsapp

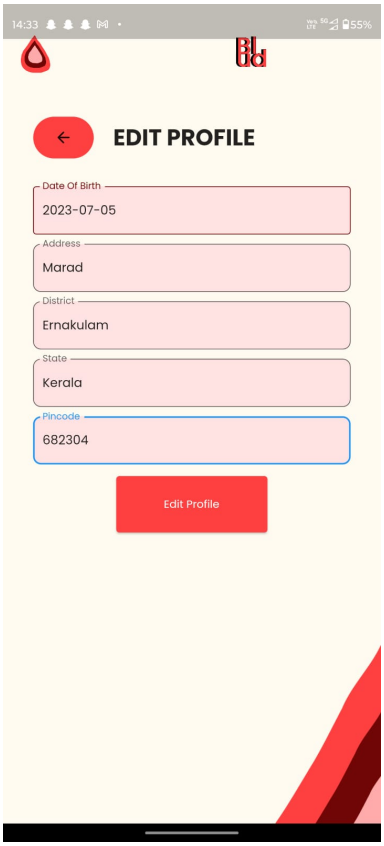

Figure 6.5: Editing profile

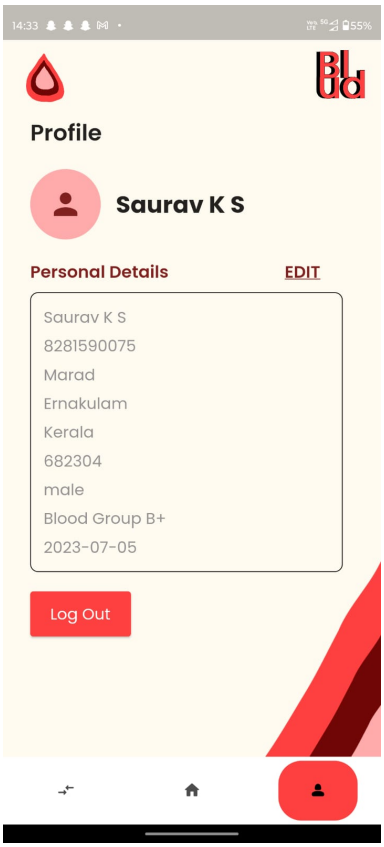

Figure 6.6: Edited profile

6.3 Requesting and Accepting process

Requesting blood through the Blud app is a straightforward process. Users need to provide the necessary blood request details, and the app will automatically find potential donors based on the user's location. The app will then send WhatsApp messages to these potential donors, notifying them of the blood request. After a donor accepts the request, users can conveniently view the donation history, ensuring transparency and keeping track of previous successful donations. This streamlined and efficient process simplifies blood donation coordination, making it easier for users to find donors and receive much-needed assistance promptly.

Figure 6.7: Blood request details

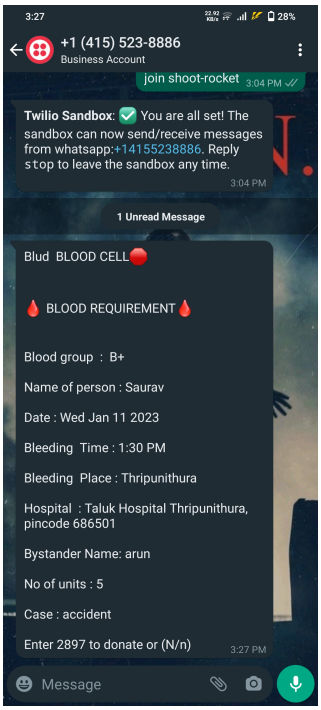

Figure 6.8: Whatsapp message from app

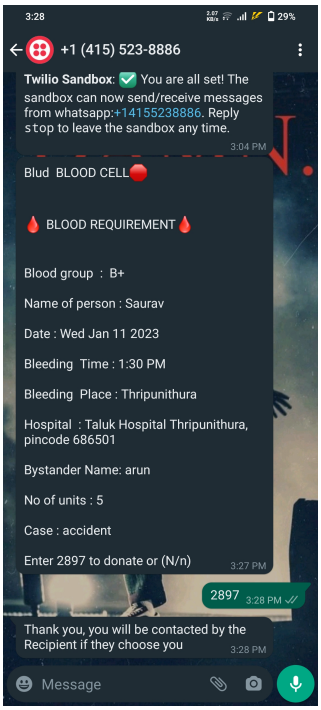

Figure 6.9: User accepting the donation

Figure 6.10: History of donor

Figure 6.11: History of Recipient

6.4 Summary

In this chapter, the results of the work are explained in detail. It begins with the objectives that was achieved through the project in section 6.1 and also how these objectives were achieved as well through the use of the outputs obtained and responses from the user

Chapter 7

Conclusion

7.1 Summary

Throughout the course of this report, the "BLUD" project has been comprehensively detailed. Chapter 1 introduces the "BLUD" project, an ambitious endeavor aimed at revolutionizing the blood donation process by creating a platform connecting donors and recipients. The project focuses on implementing real-time tracking of donors and a live blood request feed to address inefficiencies and ensure timely blood supply, ultimately saving lives. The report's structured organization covers various stages, including feasibility study, system design, implementation, testing, and results analysis, ensuring a comprehensive and systematic approach to project development. By adopting this well-organized methodology, the "BLUD" project strives to make a meaningful impact on the blood donation ecosystem, fostering a more efficient and life-saving process.

In Chapter 2, the analysis of existing blood donation systems sheds light on their limitations and performance issues. The proposed system for the "BLUD" project presents a compelling solution to overcome these challenges. The chapter also encompasses requirements engineering, including a thorough feasibility study, requirement elicitation, analysis, specification, and validation processes. This meticulous understanding of the project's needs and objectives ensures that the "BLUD" platform aligns with its intended goals, providing a robust and effective solution to streamline blood donation procedures and improve overall efficiency.

Chapter 3 delves into the design aspect of the "BLUD" project, presenting key components like Server, UI, Database, and Microservices. The Server handles critical tasks such as user authentication and blood request processing, while the UI module enables user registration and blood request submission. The integration of various app development tools and environments ensures optimized API usage and package integration, resulting in efficient front-end, back-end, and database communication. This chapter provides detailed insights into how these modules synergize to create a functional and user-friendly blood donation platform.

Additionally, Chapter 4 explores the implementation process, utilizing various app development tools and environments to facilitate API usage and package integration for peak performance. The chapter elaborates on how these modules work cohesively to create a seamless and effective blood donation platform, connecting donors and recipients efficiently and making the blood donation process more accessible and convenient.

Rigorous testing strategies, including unit testing, black box testing, white box testing, and comprehensive integration testing are showcased in chapter 5. These methodologies are vital in identifying and resolving potential defects, ultimately resulting in a defect-free software product that instills confidence in the "BLUD" project's performance and behavior.

Finally, Chapter 6 presents detailed results, highlighting the successful achievement of objectives and goals. The outcomes are supported by user responses and obtained outputs, demonstrating the effectiveness and fulfillment of the project's aspirations. Overall, the "BLUD" project emerges as a promising platform, streamlining the blood donation process and positively impacting the healthcare domain through efficient coordination between donors

and recipients.

7.2 Recommendations for Future Works

For performing future research on this work, we recommend the following.

1. **Live location of available donors on the map**: To further enhance the "BLUD" platform, integrating a live location feature that displays the real-time locations of available donors on a map would be highly beneficial. This functionality would enable blood banks and recipients to quickly identify nearby donors, thereby optimizing blood supply management and ensuring timely donations. The integration of GPS tracking or location-based services can be explored to achieve this feature, allowing the platform to offer up-to-date and accurate donor availability information.

2. **Blood Donation Drives in collaboration with Blood Banks**: Collaborating with established blood banks to organize blood donation drives can significantly boost the impact and outreach of the "BLUD" project. By partnering with blood banks, the platform can promote awareness about donation opportunities, mobilize more donors, and create a sense of community involvement in the blood donation process. Coordinating regular donation drives can help maintain a steady supply of blood and contribute to saving more lives. Integrating a feature that displays upcoming donation drives and encourages donors to participate actively in such events can be a valuable addition to the platform.

Appendix A

Software Requirement Specification

A.1 Introduction

A.1.1 Purpose

This project aims to connect blood donors with individuals or organizations in need of blood. It simplifies finding a suitable donor, scheduling appointments, and live tracking.

A.1.2 Document Conventions

- Convention for the main title:
  - Font Face: Computer Modern.
  - Font Style: Bold
  - Font Size: 32
- Convention for subtitle:
  - Font Face: Computer Modern.
  - Font Style: Bold
  - Font Size: 22
- Convention for the body:
  - Font Face: Computer Modern.
  - Font Style: Normal
  - Font Size: 10

A.1.3 Intended Audience and Reading Suggestions

- Blood Recipients who need to find eligible donors quickly and efficiently.
- Blood donors can donate accordingly based on the blood requests and check their successful donation history.
- The engineers and developers will be in charge of designing and putting the program into action.
- Project managers who are in charge of overseeing the development process must comprehend the requirements.
- Database Handlers.

A.1.4 Project Scope

The purpose of this project is to increase the efficiency of the blood donation process. Through this system, we can ease the process of finding suitable blood donors, enhance the tracking process and incorporate social media platforms like WhatsApp for advanced communication.

A.2 Overall Description

A.2.1 Product Perspective

The Blud is a replacement for certain existing systems that are aimed at enhancing the efficacy of blood donation procedures. The conventional system performs rather inadequately in its maintenance and updating. Blud approaches the task of conquering this dilemma and proffering a resolution by providing an efficient notification system and utilizing social media platforms like WhatsApp for efficacious communication.

A.2.2 Product Features

The blood donation app offers live tracking, effective communication, and a notification system to enhance the user experience.

A.2.3 User Classes and Characteristics

1. Recipients- The user can request blood through the app to avail donors.
2. Donors - The donors can confirm the donation of blood and provide the location of the donor.

A.2.4 Operating Environment

1. Hardware Environment - Smartphone.
2. Operating System - iOS, Android 8.0, and above.
3. Software components - Dart, Nodejs, MongoDB, Firebase.

A.2.5 Design and Implementation Constraints

1. Reduced availability of RAM.
2. Possible unavailability of Whats App in users.
3. Permission to access location for tracking features.

A.2.6 User Documentation

1. Quick Start guide.
2. Demonstration video.

A.2.7 Assumptions and Dependencies

Assumptions

1. The data provided by the user is accurate including the blood group and location of the user.
2. Selection of Whats App as our primary communication platform owing to its frequent usage amongst the general population.

Dependencies

1. The app is dependent on the network connection, like the internet.
2. The app is dependent on the accuracy of the data provided by the user.
3. The app depends on third-party API (like Whats app Business API) for communication.
4. The proper functioning of the app depends on the operating system on the device.

A.3 System Features

A.3.1 Efficient Whats App donation confirmation system

A.3.1.1 Description

The "Efficient Whats App donation confirmation system before donation" feature of the blood donation app enables the app to send donors a confirmation message via Whats App before they donate. This feature is designed to ensure that donors have all the necessary information and reminders before the donation and to encourage them to follow through with their donation commitment.

A.3.1.2 Priority

The priority of this feature is medium to high, as it can help to improve donor engagement and retention, but it is not as critical to the functioning of the app as other features such as donor tracking and communication

A.3.1.3 Stimulus/Response Sequences

Stimulus: The donor schedules a blood donation appointment.
Response: The app sends a confirmation message to the donor via WhatsApp.

A.3.1.4 Functional Requirements

- The app must have access to the Whats App API to enable the sending of confirmation messages.
- The confirmation message must include the date, time, and location of the donation, as well as any relevant information about the preparation for the donation and donor eligibility criteria.
- The app must ensure the privacy and security of donor information, under data protection laws and regulations.
- The app must provide donors with the option to opt out of receiving confirmation messages via WhatsApp.

A.3.2 Live Tracking of Donors

A.3.2.1 Description

The "Live tracking of donors" feature of the blood donation app enables the tracking of donors in real-time, allowing blood banks to quickly and easily locate donors who are currently available to donate blood. This feature is designed to enhance the efficiency and effectiveness of blood donation by providing an up-to-date and accurate list of donors who are ready to donate.

A.3.2.2 Priority

The priority of this feature is high, as it is critical to the functioning of the app and the overall success of the blood donation program. The feature helps blood banks quickly locate donors and manage their inventory, ensuring that blood is always available to those who need it.

A.3.2.3 Stimulus/Response Sequences

Stimulus: A request for the current location of available donors.
Response: The app displays a list of donors who are currently available to donate, along with their location and contact information

A.3.2.4 Functional Requirements

- The app must be able to collect and store donor location data in real time.
- A current list of available donors for blood banks must be available from the app.
- Based on their availability and location, the app must be able to filter donations.
- According to data protection laws and regulations, the app must guarantee the confidentiality and security of donor information.
- To help blood banks quickly find available donors, the app must be able to show their locations on a map.

- When a donor's blood type is urgently needed, the app must be able to notify them.
- The app must be able to keep track of how many donations each donor has made and notify donors when they are next eligible to give.

A.3.3 Live Blood Request Feed

A.3.3.1 Description

The "Live blood request feed, showing current blood requests" feature of the blood donation app enables users to view a real-time feed of current blood requests in their area. This feature is designed to make it easier for potential donors to see where their help is needed and to encourage them to donate

A.3.3.2 Priority

The priority of this feature is high, as it is critical to the functioning of the app and is the main purpose of the app, which is to facilitate blood donation by connecting donors and recipients.

A.3.3.3 Stimulus/Response Sequences

Stimulus: The user opens the app and navigates to the blood request feed.
Response: The app displays a real-time feed of current blood requests in the user's area.

A.3.3.4 Functional Requirements

- The app must have access to a database of current blood requests in the user's area.
- The app must be able to display blood requests in a real-time feed, with the most recent requests displayed first.
- The app must be able to filter blood requests by location, blood type, urgency, and other relevant criteria.
- The app must provide users with a way to respond to blood requests by indicating their willingness to donate and providing their contact information.
- The app must ensure the privacy and security of user information, under data protection laws and regulations.
- The app must be able to send notifications to users when a new blood request is posted in their area.
- Overall, the "Live blood request feed, showing current blood requests" feature is a critical component of a blood donation app, enabling users to see where their help is needed and to connect with potential recipients in real-time. This feature can help to encourage more blood donations and ultimately save lives, making it a high-priority feature for any blood donation app.

A.4 External Interface Requirements

A.4.1 User Interfaces

- Login Page- The user logs into his/her account using the provided OTPs
- Registration Page- User enters the necessary details
- Survey Page-Takes the user through a set of questions to check the eligibility of the user to donate blood.
- Home Page- This page contains the live feed of blood requests and the navigational functionalities to move to required pages
- Blood Request Page- The user can request blood of any group by giving the essential details
- Available Donors' list-Once the user requests blood, this page displays the available list of Donor sorted by distance and availability
- Profile Page-User Profile can be viewed on this page and can be edited by the user
- History Page- This page views the history of blood donations or requests done previously by the user throughout the usage of the app

A.4.2 Hardware Interfaces

- Device: Smartphone
- Operating System - iOS, Android 8.0 and above
- RAM: 2 GB or above
- Storage space: 32 GB

A.4.3 Software Interfaces

- VS Code-text editor for development purposes
- MongoDB compass-Document database for storing app data
- Firebase-for using tools like real-time notification and real-time tracking
- NodeJS-for creating the back-end server by using JavaScript
- Dart-creating the app
- WhatsApp Business API-API is used for connecting the app with Whats app to communicate with potential donors.

A.4.4 Communications Interfaces

- The user must have an internet connection
- Must have GPS feature turned on
- Web socket for real-time data sharing using HTTP handshake protocol

A.5 Other Nonfunctional Requirements

A.5.1 Performance Requirements

1. Higher speed of network connection to provide quick response and update for blood activities.
2. The latest version of Android for the smoother response of the app

A.5.2 Safety Requirements

The user must confirm in the app if a donor approves the request for donation so that the request is revoked from the database. If the confirmation is not given, the request continues to circulate in the live feed until a donor is found.

A.5.3 Security Requirements

1. The system uses a secure database that can only be accessed by the admin.
2. An advanced database system is used that has its own readily available backup system so that the user's data is not lost.
3. OTP system used for registration provides an extra layer of security from hackers that may access user profiles.

A.5.4 Software Quality Attributes

1. Usability- The app provides a user-friendly interface for easier functioning
2. Maintainability - The app will be provided with regular updates to remove bugs or to add additional features
3. Portability-The app can be installed and used in any of the smartphones of the current technology.

A.6 Other Requirements

A No SQL database is required for efficient access to data.
The user should be able to donate blood according to their eligibility and health conditions.

A.7 Appendix A: Glossary

- User: Donors and receivers that use the app for blood activities
- No SQL: Not only Structured Query Language
- HTTP handshake: connecting server and client using an HTTP request and sharing data in full duplex mode.
- OTP: One-time password for authentication
- API: Application programming interface that provides the readily available features that are beyond the scope of development

A.8 Appendix B: Analysis Models

No Analysis Models developed

A.9 Appendix C: Issues List

No issues discovered at this stage

References

[1] Simply blood app. https://play.google.com/store/apps/details?id=com.simplyblood, 2023.

[2] Ublood app. https://play.google.com/store/apps/details?id=com.ubld.ublood, 2023.

[3] Blood friends app. https://play.google.com/store/apps/details?id=com.posl.bloodfriends, 2023.

[4] Blood donation eligibility criteria. http://nbtc.naco.gov.in/page/eligibility, 2023.

[5] Ian Sommerville. *Software Engineering (tenth edn. global edition)*. Boston, MA: Pearson Education, Inc, 2016.

[6] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[7] Flutter. https://flutter.dev/, 2023.

[8] Figma. https://www.figma.com/, 2023.

[9] Nodejs. https://en.wikipedia.org/wiki/Node.js, 2023.

[10] Visual studio code. https://code.visualstudio.com/docs, 2023.

[11] Mongodb. https://www.mongodb.com/docs/manual/, 2023.

[12] Firebase. https://en.wikipedia.org/wiki/Firebase, 2023.

[13] Postman. https://www.postman.com/product/what-is-postman/, 2023.

[14] Android studio. https://en.wikipedia.org/wiki/Android_Studio, 2023.

Index