



Fayoum University,

Faculty of Computers and Information

PATIENT SHARE

Graduation Project 2019

By:

Ahmed Emad Hamdy (CS)

Ahmed Sayed Anwer (IS)

Mohamed Ramadan Hassanein (CS)

Mostafa Mohamed Mohamed (CS)

Moataz Muhammad Al-Azzamy (CS)

Hadeer Kamel El-Sayed (CS)

Supervised by: Dr/ Mostafa Thabet

ACKNOWLEDGEMENT

First of All , we have to thank our God for helping us in all times through our Life, our college Dean Dr/Nabila Hasan, Vice Dean Dr/ Shereen Taie and Vice Dean Dr/ Mohammed Khafagy for supporting, helping and Teaching us throughout the study years .

Special thanks for our project supervisor, Dr/ Mostafa Thabet Mohamed for supporting and encouraging us to make this progress in our idea and finishing it as planned. Without his assistance and dedicated involvement in every step throughout the development process, this project would have never been accomplished. We would like to thank you very much for your support and understanding over these periods.

Getting through our success required more than academic support, and we have many people to thank for over the past four years. We cannot begin to express our gratitude and appreciation for their friendship. They have been unwavering in their personal and professional support during the time we spent at the university.

For many memorable evenings out and in, we have to thank everyone in our community.

Most importantly, none of this could have happened without our family. We want to thank every family member for his/her support and encouragement .The words cannot express enough your role in our life.

ABSTRACT

Patient Share contributes in solving some health sector problems in Egypt. First Stage, by solving blood shortage problem by providing those who need any type of blood with the type they need by showing the donors who can donate within their geographical area. Second Stage, we will evaluate the application to cover all medical donations in Egypt. Patient Share facilitates the process of blood donation by providing features like notifications, map monitoring and hospitals monitoring.

Generally, it's designed to improve the whole donation process in different and new way.

TABLE OF CONTENTS

List of Figures.....	VIII
List of Tables.....	X
1 INTRODUCTION	1
1.1 Overview	1
1.2 Motivations	2
1.3 Problem Definitions	2
1.4 Objectives	2
1.5 Chapters Outlines	3
2 LITERATURE REVIEW	4
2.1 Overview	4
2.2 Web Application Technology	5
2.2.1 React	5
2.2.1.1 Why Learn React?	5
2.2.2 Node.js	6
2.2.2.1 How Node.js Works?	7
2.3 Database	8
2.3.1 Firebase	8
2.4 Mobile Application Technology	10
2.4.1 iOS	10
2.4.1.1 Why iOS	11
2.4.2 Swift	11

2.4.2.1 Why swift	12
2.4.3 Mobile Application Examples	13
2.4.3.1 Apple maps	13
2.5 Project Competitors	14
2.5.1 Tradition ways	14
2.5.2 Facebook group	14
2.5.3 Life saver blood donation	15
2.5.4 Blood bank Bangladesh	15
2.5.5 Simply blood –find blood donor	16
2.5.6 بنك الدم الالكتروني	16
2.5.7 Life saver blood bank	17
3 Project Management	18
3.1 Overview	18
3.2 Development methodology	19
3.2.1 Waterfall model	19
3.3 Plan	21
3.4 Feasibility study	21
3.4.1 Executive Summary	21
3.4.2 Description of Products and Services	22
3.4.3 Technology Consideration	22
3.4.4 Product/Service Marketplace	22
3.4.5 Marketing Strategy	23
3.4.6 Organization and Staffing	23

3.4.7 Schedule	24
3.4.8 Financial Projections	24
3.4.9 Findings and Recommendations	25
3.4.10 Risk	25
3.5 Team structure	26
4 Analysis and design	27
4.1 Overview	27
4.2 Requirements and analysis	28
4.2.1 Functional requirements	28
4.2.2 Non-Functional requirements.....	28
4.3 Use Case Diagrams	29
4.4 Use case scenario	32
4.5 System sequence and sequence	39
4.5.1 System Sequence	39
4.5.2 Sequence	41
4.6 Activity Diagrams	46
4.7 Context Diagram	53
4.8 Domain Model	53
4.9 Data Flow Diagram	54
4.10 Class Diagram	55
4.11 Entity Relationship Diagram	56

5 System Architecture	57
5.1 Overview	57
5.2 Mobile Application	58
5.3 Database	58
5.4 Web Interface	58
5.5 System screenshot	59
6 Implementation & Testing	68
6.1 Overview	68
6.2 Testing	69
6.2.1 Database Testing	69
6.2.2 Map Testing	69
6.2.3 Application Testing	69
6.3 Code screenshots	70
7 Future work	77
7.1 Overview	77
7.2 Summary	78
7.3 Future work	78
REFERENCES	79

LIST OF FIGURES

Figure 2.1 how Firebase work	10
Figure 3.1: waterfall model	19
Figure 3.3 Plan	21
Figure 4.1: Use Case (Donor)	29
Figure 4.2: Use Case (Hospital)	30
Figure 4.3: Use Case (Requester)	31
Figure 4.4 System Sequence Diagram (Front-end)	39
Figure 4.5 Sequence Diagram (Backend)	40
Figure 4.6 Sequence Diagram (Login)	41
Figure 4.7 Sequence Diagram (Request blood)	42
Figure 4.8 Sequence Diagram (Request Answer)	43
Figure 4.9 Sequence Diagram (Navigation)	43
Figure 4.10 Sequence Diagram (Reports)	44
Figure 4.11 Sequence Diagram (change password)	44
Figure 4.12 Sequence Diagram (Block reason)	45
Figure 4.13 Activity Diagram (Create user)	46
Figure 4.14 Activity Diagram (Login)	47
Figure 4.15 Activity Diagram (request blood)	48
Figure 4.16 Activity Diagram (Accept request)	49
Figure 4.17 Activity Diagram (Follow view)	50
Figure 4.18 Activity Diagram (Navigation)	51
Figure 4.18 Activity Diagram (Generate report)	52

Figure 4.19 Context Diagram	53
Figure 4.20 Domain Model	53
Figure 4.21 DFD	54
Figure 4.22 Class Diagram	55
Figure 4.23 Entity Relationship Diagram	56
Figure 5.1 System Architecture	58
Figure 5.2 Login screen in mobile	59
Figure 5.3 Sign up screen in mobile app	60
Figure 5.4 Order donation screen in mobile app	61
Figure 5.5 Notification screen in mobile app	62
Figure 5.6 Donation requests from donor in mobile app	63
Figure 5.7 Follow view of donor in mobile app	64
Figure 5.8 Home page of web app	65
Figure 5.9 Register from web app	65
Figure 5.10 Who we are?	66
Figure 5.11 Download link from web app	66
Figure 5.12 How can project help you?	67
Figure 5.13 Authentication code	67
Figure 6.1 Login in mobile app	70
Figure 6.2 Signup in mobile app	71
Figure 6.3 order donation in mobile app	72
Figure 6.4 Donate in mobile app	73
Figure 6.5 Donors places in mobile app	74
Figure 6.6 Backend for web application	75

Figure 6.7 Login in web app75

Figure 6.8 Register in web app76

Figure 6.9 Authentication in web app76

LIST OF TABLES

Table 3-1: Financial Projections	24
Table 4-1: Use Case(Signup)	32
Table 4-2: Use Case(Login)	33
Table 4-3: Use Case(Request Blood)	33
Table 4-4: Use Case(Answer Request)	35
Table 4-5: Use Case(Get navigation)	35
Table 4-6: Use Case(Follow view)	36
Table 4-7: Use Case(Generate reports)	36
Table 4-8: Use Case(Block user)	37
Table 4-9: Use Case(Edit data)	37

LIST OF ACRONYMS/ABBREVIATIONS

JSX	JavaScript XML
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
DB	Database
NPM	Node Package Manager
Node.js	Node. JavaScript
iOS	iPhone operating system
GPS	Global Positioning System
OS	Operating System
UI	User Interface
GSM	Global System of Mobile
LLVM compiler	Low Level Virtual Machine compiler
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram

Chapter One

INTRODUCTION

1.1 Overview

Patient share system is designed for improving the medical experience in Egypt. It provides its services through a mobile and hospital use web application for administration operation. That system enables patients to reach the largest number of blood donors who have the same blood type as first stage then system turns up to donate any medical bid

This chapter is organized into 5 sections:

- Section 1.1 Overview.
 - Section 1.2 Motivations.
 - Section 1.3 Problem Definitions.
 - Section 1.4 Objectives.
 - Section 1.5 Chapters Outlines
-

1.2 Motivation

Egyptian Community service and solve problems of the health sector in Egypt. The targeted people are blood donors and blood transfusion recipients at first stage, then people who needs a medical donation at the rest of the stages. It will control a huge market share, because it's the first one of its kind in Egypt. Till now there are no restrictions or regulations that will limit the possibility of our work to continue or even better expand.

1.3 problem definition

The most sectors of the Egyptian state suffer from problems is health sector. The most frequent problems blood shortage and Blood transfusion was a serious problem we face every day in our country. The effect of this problem is the lack of communication between blood donors and recipients of blood transfusions. They cannot reach each other easily. If we take a look at the current situation in Egypt, we will know the magnitude of the problem.

- The rate of blood donation in Egypt is 1/6 the rate in developed countries.
- World Health Organization put Egypt from 120 countries is unsafe in the blood.
- Egyptians need 2.5 to 3 million blood bags, There are no more than 800,000 bags.
- The reduction rate covers less than 30% of the patients' needs.
- The blood stock fills 1/3 of the actual requirement.
- One out of 10 people entering the hospital needs blood vessels.

1.4 objectives

- Implement a system that provides patients with the type of blood they need.
- Facilitating access to as many blood donors as possible.
- Ensure that each patient receives a blood donor with the type of blood he needs in his geographical scope.
- Create a friendly user interface to interact with.
- Contribute to public awareness of the importance of technology in working life through the use of the application.
- Giving the hospital validity of administration of block user or allow him to donor

1.5 Chapters Outlines

- **Chapter2 Literature Review:**
This chapter includes more information about used fields and technologies in the implementation, and Competitors who have ideas that are similar to our idea and their advantages & disadvantages.
- **Chapter3 Project Management:**
This chapter provides the process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements.
- **Chapter4 Analysis and Design:**
System analysis and design deal with planning the development of information systems through understanding and specifying in detail what a system should do and how the components of the system.
- **Chapter5 System Architecture:**
This chapter provides general definitions and detailed information about system.
- **Chapter6 Implementation& Testing:**
This chapter includes experiment and its result and then validates this result.
- **Chapter7 Future work:**
This chapter will have a recap for whole idea and work that will be implemented in the future.

Chapter Two

LITERATURE REVIEW

2.1 OVERVIEW

This chapter will provide more detailed information about the technologies used in the project, examples and other important info about the vision to be clear.

Competitors and their strength, weakness, opportunities and more about them are also discussed in this chapter.

This chapter is organized into 5 sections:

- Section 2.1 Overview
- Section 2.2 Web Application Technology.
- Section 2.3 Database.
- Section 2.4 Mobile Application Technology.
- Section 2.5 Project Competitors

2.2 WEB APPLICATION TECHNOLOGY

Web applications refer to applications accessed via Web browser over a network and developed using browser-supported languages (e.g., HTML, JavaScript). For execution Web applications depend on Web browsers and include many familiar applications such as online retail sales, online auctions, and webmail. Web applications are needed in the area of business-to-business interaction over networks, e.g., for overseas companies that outsource projects to each other. The adoption of a Web applications infrastructure can provide vital processes such as transfer of funds and updates of pricing information. There are many technologies for building web applications (Front-end, Back-End) such as Angular, React, Node.js and etc.

2.2.1 React

React is a popular open source front-end JavaScript library developed by Facebook. React is widely popular among developer communities because of its simplicity and easy but effective developing process. React makes it easier to create interactive user interfaces. It efficiently updates through rendering the exact components to the view of each state and makes the data changes in the application. In React, every component manages their own state and composes them to the user interfaces. This concept of components instead of templates in JavaScript, plenty of data can easily be passed to the app and thus keep the state out of the DOM. Using Node React can also be rendered on the server side. Alongside web apps, to build mobile applications we can use React Native as well.

2.2.1.1 Why Learn React?

React was introduced to the world two years ago, and since then it has seen impressive growth, both inside and outside Facebook. New web projects at Facebook are commonly built using React in one form or another, and it is being broadly adopted across the industry. Developers and engineers are choosing React because it allows spending more time to focus more on the product development and less time spent on fighting and learning to the framework.

Short and Easy Learning Curve

Unlike some other JavaScript libraries where it takes a lot of time to learn about the frameworks, in React it does not take much of an effort to start building an application. React is comprised of many strong features. Readability is one of the greatest strength of React. It is easily readable even to those who are unfamiliar to it.

While other frameworks require learning many concepts about the framework itself, ignoring the language fundamentals, React does the absolute opposite.

React is fast and agile

React is featured with one-way unidirectional data flow between the states and layers in an application. This means data flows in single direction between the application states and layers. In two-way data binding like Angular, if a model is changed, the view also changes and vice-versa. React renders the updates in the DOM much quicker than alternative frameworks and it is a much smaller library. DOM means document object model. Thus, it is easy to choose the tools to get the job done.

React Introduced JSX

JSX is a language that lets you specify the DOM elements before the components right inside of JavaScript files. This means the logic behind the components and the visuals are all in one place. This is such a great idea when other frameworks are taking queues to place them.

Big Development Community

Big companies like New York Times, Facebook, and Netflix are using React in production. They are continuously contributing to develop the React core and building amazing third party libraries that work great with any React applications.

2.2.2 Node.js

Node.js (Node) is a cross platform runtime environment originally developed in 2009 by Ryan Dahl for developing server-side applications. It can be regarded as server-side JavaScript. It was created to address the issues platforms can have with the performance in network communication time dedicating excessive time processing web requests and responses. “Node.js is a platform built on Chrome’s JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.” Node.js Modules are plugins, add-ons, and extensions for Node to help with the development process. The Node module exposes a public API (Application Programming

Interface) that one can use after the module is imported into the current script. Node modules can be categorized as core modules, third party modules, and local modules.

2.2.2.1 How Node.js Works?

The main distinctive features of the Node architecture are the usage of non-blocking, event driven, asynchronous I/O calls that operate in a single thread. Conventional web servers handle concurrency by spawning new threads for each new request, which can max out the available memory. Node is lightweight, efficient, and different. It is able to support tens of thousands of concurrent connections because of its unique features. Even with limited memory and a single 8 thread, Node can achieve high concurrency rate without having to perform context switching between threads.

Non-Blocking Event Loop

Node is non-blocking in the sense that it is able to service multiple requests, and it doesn't waste clock cycles in I/O tasks as is the case in the conventional blocking model. The conventional blocking model tends to block subsequent requests sent to a server when it is performing I/O operations such as reading content from a database. In order to be non-blocking, Node uses an event loop, a software pattern that facilitates non-blocking I/O combined with event-driven I/O, a scheme where a registered event callback function is invoked when some action happens in the program.

Single-Threaded Model

Node is a process that runs in an event loop making use of a single thread to service any requests. Whereas other web servers like Apache spawn a new thread per request, which starts with a fresh state every time. Node is powerful considering the way it permits non-blocking I/O to occur in a single thread.

Asynchronous Programming

While the non-blocking part of Node makes it able to accept virtually all the requests made to it, its asynchronous programming makes it possible to handle the requests by effectively utilizing the limited clock cycles and memory available to its single-threaded architecture. Asynchrony is in the root of Node because almost all the APIs exposed through Node modules are asynchronous (although synchronous versions may exist). Node is able to achieve high concurrency by its asynchronous calls via a callback function to handle the tasks in its event loop. Node integrates asynchronous

programming in its architecture by means of asynchronous APIs with callback function.

2.3 DATABASE

Organizing data is an important activity in the process of developing informatics systems. The performances of the informatics system depend greatly on how data are organized. In the process of organizing data are followed mainly two major objective concerning optimizing computing resources: fast access to data and small memory space occupied. Sometimes these two objectives are contradictory. If data access is faster, it is necessary memory space in addition to provide additional information to ensure the access. These additional data can be links, files with addresses, pointers, index files etc.

2.3.1 Firebase

Firebase is Google's mobile and web application development platform that helps you build, improve, and grow your app and it a Backend-as-a-Service—BaaS—that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform.

Firebase component you need to create your app

- Authentication
- Real-time Database
- Cloud Firestore
- Cloud Storage
- Cloud Functions
- Firebase Hosting
- ML Kit

Firebase features

Real-time In-App Analytics - provides key metrics to help you understand how engaged your app users are, their activity over time, which device and OS they use and their geographic breakdown.

Crash Recordings - detects and records crashed sessions automatically. You can see the exact sequence of actions that resulted in a crash with your own eyes and identify the single UI element causing the crash.

Conversion Funnels - allows you to configure conversion funnels for relevant processes within your app, measure the conversion rates, and understand the reasons for the success or failure of each process. In addition, you will be able to see the average completion time for each step.

Actionable Insights - act on pinpoint insights from the App see solution to maximize customer engagement, conversions and in-app monetization.

Auto-Detect UI Problems - detects unresponsive actions made by your users, such as taps, swipes or pinches, and helps you to optimize the UI.

Easy Setup & Integration - just drop the App see SDK in your app and add a single line of code. Takes less than 1 minute!

Firestore Pros & Cons

Pros

- Email & password, Google, Facebook, and Github authentication
- Real-time data
- Ready-made API
- Built in security at the data node level
- File storage backed by Google Cloud Storage
- Static file hosting
- Treat data as streams to build highly scalable applications
- Don't worry about your infrastructure!

Cons

- Limited query abilities due to Firestore's data stream model
- Traditional relational data models are not applicable to NoSQL; therefore, your SQL chops will not transfer
- No on-premise installation

Figure show how Firebase work

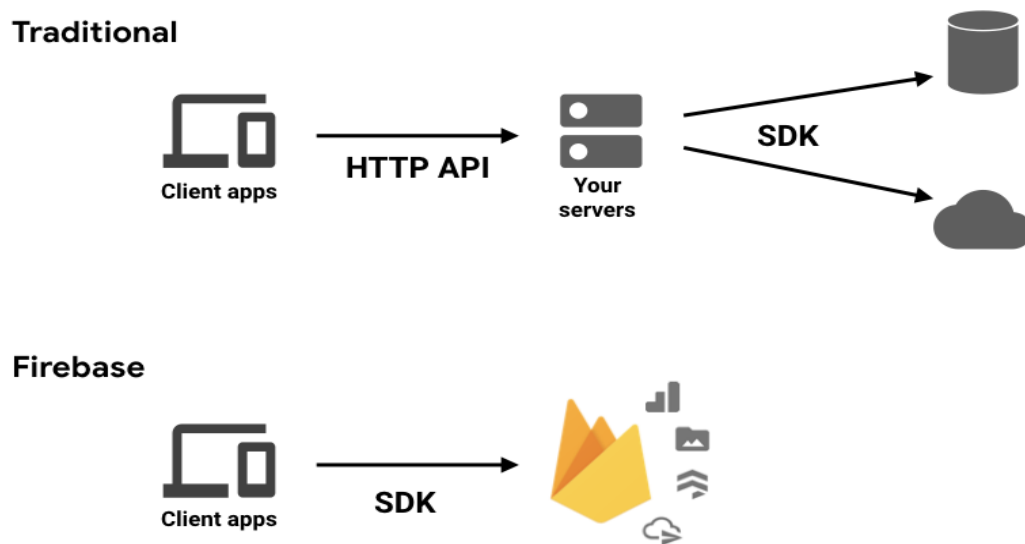


Figure 2.1

2.4 Mobile Application Technology

Mobile application development in recent times is growing exponentially. Today each person in this world has a smart-phone in his pocket. Smartphone's combine a range of functions such as media players, camera and GPS with advanced computing abilities and touch screens are enjoying ever-increasing popularity. Smartphone's help us to achieve a range of tasks through something known as applications or Apps to short. According to Gartner, Google's Android, and Apple's iOS.

2.4.1 iOS

iOS (formerly iPhone OS) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod Touch. It is the second most popular mobile operating system globally after Android. Originally unveiled in 2007 for the iPhone, iOS has been extended to support other

Apple devices such as the iPod Touch (September 2007) and the iPad (January 2010). As of March 2018, Apple's App Store contains more than 2.1 million iOS applications, 1 million of which are native for iPads. These mobile apps have collectively been downloaded more than 130 billion times. iOS utilizes many security features in both hardware and software like Secure Boot, Secure Enclave, Passcode, Touch ID and Two-Factor Authentication.

2.4.1.1 Why iOS

High Quality Emulators

As compared to other operating systems, iOS emulators are definitely faster and also offer great support. This decidedly makes the iOS emulators better, helping the developers in speeding up their app development process and in making it a whole lot easier.

Better Developer Support & Tools

The developer support and tools offered by Apple is superior as compared to all the other operating systems including one of the market leaders – Android.

Lesser Fragmentation

Even a perfectly developed Android app has a high chance of running into bugs simply because they have to deal with a huge number of brands, platforms, and screen sizes! However, the number of versions and types of devices are limited in case of iOS app development, thus making it relatively easier to build the apps and reducing the chances of unforeseen bugs in the app.

Higher Revenue

It has been established through research and surveys that iOS users are generally of a higher income group on an average. This means that the chances of them spending on and within an app are a lot higher than the average Android user. This means the developers and app development companies would definitely prefer to develop apps for iOS in place of Android.

2.4.2 Swift

Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. for iOS, macOS, watchOS, tvOS, Linux, and z/OS. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products. It is built with the open source LLVM compiler framework and has been included in Xcode since version 6, released in 2014. On Apple platforms, it uses the Objective-C runtime library which allows C, Objective-C, C++ and Swift code to run within one program.

Swift was introduced at Apple's 2014 Worldwide Developers Conference (WWDC). It underwent an upgrade to version 1.2 during 2014 and a more major upgrade to Swift 2 at WWDC 2015. Initially a proprietary language, version 2.2 was made open-source software under the Apache License 2.0 on December 3, 2015, for Apple's platforms and Linux. Through version 3.0 the syntax of Swift went through significant evolution, with the core team making source stability a focus in later versions. In the first quarter of 2018 Swift surpassed Objective-C in measured popularity.

Swift 4.0, released in 2017, introduced several changes to some built-in classes and structures. Code written with previous versions of Swift can be updated using the migration functionality built into Xcode. Swift 5, released in March 2019, introduced a stable binary interface on Apple platforms, allowing the Swift runtime to be incorporated into Apple operating systems. It is source compatible with Swift 4.

Swift won first place for Most Loved Programming Language in the Stack Overflow Developer Survey 2015 and second place in 2016.

2.4.2.1 Why swift?

First of all it is open source, safe and fast language. Its mainly advantages are:

Rapid development process

A clean and expressive language with a simplified syntax and grammar, Swift is easier to read and write. It is very concise, which means less code is required to perform the same task, as compared to Objective-C. Automatic Reference Counting (ARC) does all the work tracking and managing the app's memory usage, so developers don't need to spend time and effort doing that manually. Accordingly, it usually takes less time to build iOS apps with Swift.

Easier to scale the product and the team

You get a product that is future-proof and can be extended with new features as needed. Thus, Swift projects are typically easier to scale. The fact that Apple is more likely to support Swift than Objective-C should also get serious consideration for long-term investment.

Improved safety and performance

As suggested by its name, Swift is made to be... well, swift. With a focus on performance and speed, the language was initially designed to outperform its predecessor. Namely, the initial release claimed a 40 percent increase in performance, as compared to Objective-C. Over the years, multiple benchmarks and tests conducted by individual developers have proved that. Moreover, there are many ways to optimize Swift code for even better performance.

Automatic memory management

Swift uses Automatic Memory Counting (ARC) – a technology aimed to add a garbage collector function that wasn't introduced to iOS before. Languages like Java, C#, and Go use garbage collectors to delete class instances that are no longer used. They are useful to decrease your memory footprint but can add up to 20 percent to CPU. Before ARC, iOS developers had to manage memory manually and constantly manage retain counts of every class. Swift's ARC determines which instances are no longer in use and gets rid of them on your behalf. It allows you to increase your app's performance without lagging your memory or CPU.

2.4.3 Mobile Application Examples

Many applications overall world provide variety of services. These apps create a competition in the market and all try to be number one and create a competitive advantage. In this section, there is a comparison between PROJEC and its competitors in variety of elements.

2.4.3.1 Apple maps

Is a web mapping service developed by Apple Inc. It is the default map system of IOS, MACOS, and watch OS. It provides directions and estimated times of arrival for

automobile, pedestrian, and public transportation navigation. Apple Maps also features Flyover mode, a feature that enables a user to explore certain densely populated urban centers and other places of interest in a 3D landscape composed of models of buildings and structures.

2.5 Project competitors

2.5.1 Tradition ways

Strength point:

People you deal with are not anonymous so you can trust them

Weakness point:

You don't reach as many donators as through the application because the connections may be not enough.

Threats: Building strong relations.

Country: Worldwide

Target: Family and friends

Technology: mobile phone calling service

2.5.2 Facebook group

Strength point:

Facebook has too many users so it will be more reachable by Facebook more than any application

Weakness point:

You don't reach as many donators as through the application. People in the group may don't want to donate and may be far away from the required location.

Threats: Reach

Country: Worldwide

Target: People in the group

Technology: Facebook app

2.5.3 Life saver blood donation

Strength points:

Register your account and verify your number, then complete your profile for better experience.

Donate blood List of blood request there, search around your location and donate blood and save life.

All requests are verified through number.

Search Blood if you need blood simply search blood.

Search through Blood type or search through location. Add Blood Request If you can't find your blood through search simply add a blood request by Entering your blood group and your location.

Weakness point:

Not well marketed, users can see other user's data so it is not secure and it does not use Google maps and Google navigation so it is hard to reach the suitable users.

Threats:

Users does not want to donate blood through the internet because they don't feel secure using them

2.5.4 Blood bank Bangladesh

Strength point:

You can find Blood Banks List.

You can see the location of Blood banks in Map.

You can get address and contact of Blood Banks.

You can be a Blood Donor.

You can find Blood Donors.

You can search and filter Blood Donors by Blood Groups.

Weakness point:

Not well marketed, users can see other user's data so it is not secure and it does not use Google maps and Google navigation so it is hard to reach the suitable users.

Threats:

Users does not want to donate blood through the internet because they don't feel secure using them.

2.5.5 Simply blood –find blood donor

Strength point:

Take hours to find a suitable blood donor. Calling hundreds of blood donors is not required with Simply Blood. Just a single search allows you to reach maximum number of blood donors in minimum possible time and that too within just 5

KMs from where the blood is required.

- Privacy – apart from other blood donation apps available simply blood don't show donor's contact details to public. Only a verified user can see your contact that too to a limited number of contact details.
- pick a date – blood donors can make special days like birthday, anniversary, festival, tribute to someone, holidays, etc. even more special by donating on that day. Simply blood connects the donor to the needy on that particular day to make it special.
- Simply blood ambassador program (sap) - join the simply blood ambassador program and encourage others to be a part of it.
- Username - create username to avoid your number being visible to other member. Saves your blood donation history

Weakness point:

Not well marketed, and it is designed for people in India only.

Threats:

Users does not want to donate blood through the internet because they don't feel secure using them.

2.5.6 بنك الدم الإلكتروني

Strength point:

Collect database about donors to make get in touch easily with patient

Weakness point:

Not well marketed, users can see other user's data so it is not secure and it does not use Google maps and Google navigation so it is hard to reach the suitable users.

Threats:

Users does not want to donate blood through the internet because they don't feel secure using them.

2.5.7 Life saver blood bank

Strength point:

- List of blood request there, search around your location and donate blood and save life.
- All requests are verified through number.
- Add Blood Request If you can't find your blood through search simply add a blood request by
Entering your blood group and your location.
- If you have or you know any donation centers then add their location and Contact number.

Weakness point:

- Not well marketed, users can see other user's data so it is not secure and it does not use Google maps and Google navigation so it is hard to reach the suitable users.

Threats:

- Users does not want to donate blood through the internet because they don't feel secure using them.

Chapter Three

PROJECT MANAGEMENT

3.1 Overview

This chapter will discuss the used development methodology in the project and how it was accomplished; also discuss the plan that should be followed to achieve all tasks successfully. Team members are also included in this chapter and their roles.

This chapter is organized into 5 sections:

- Section 3.1: Overview.
- Section 3.2: Development Methodology.
- Section 3.3: Plan.
- Section 3.4: Feasibility Study.
- Section 3.5: Team Structure.

3.2 Development methodology

3.2.1 Waterfall model

The development methodology used in this project. The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

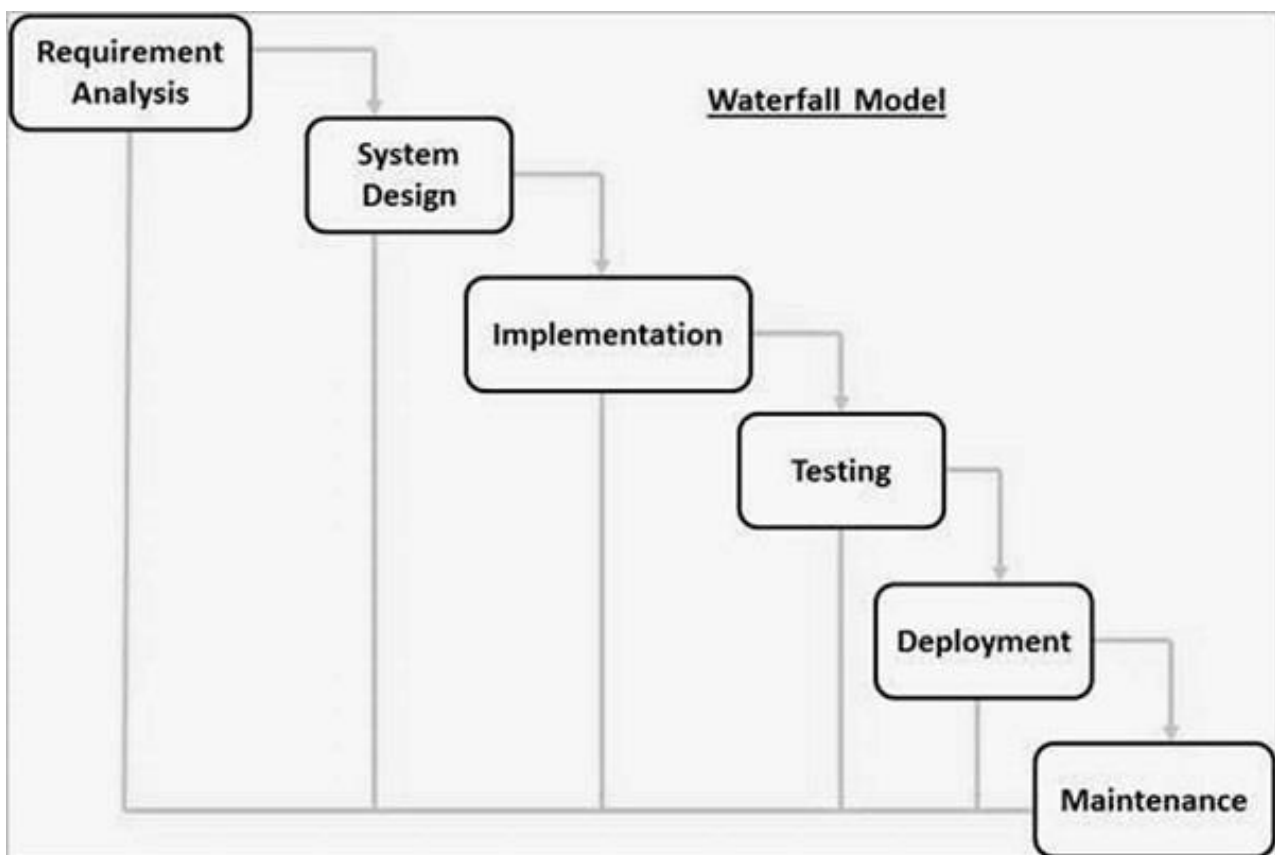


Figure 3-1: waterfall model.

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

3.3 Plan

analysis	14 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
System Study	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	ahmed sayed
feasibility study	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	mohamed ramadan
Requirement analysis	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	hadeer kamel
learning	30 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	all team member
mid term stop	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
design	60 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	ahmed sayed
physical design	10 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
digrams design	10 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
logical design	51 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
data base design	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	ahmed beledy
server architect and design	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	mostafa
front end fourms	14 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	ahmed sayed
final exam stop	22 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
coding	45 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	all team member
testing	28 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
program test	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	moutaaz elazamy
mid term 2 exam	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
integration test	14 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	hadeer kamel
Implementation	70 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
run software on server	14 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	mostafa
prouduct in app store and marketing	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	mohamed ramadan
final term 2 exam	22 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	
documentation complete	7 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	ahmed sayed
complete marking plan	20 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	moutaaz elazamy
Maintenance	10 days	١٤/١٠/٢٠١٨	١٤/١٠/٢٠١٨	mohamed ramadan

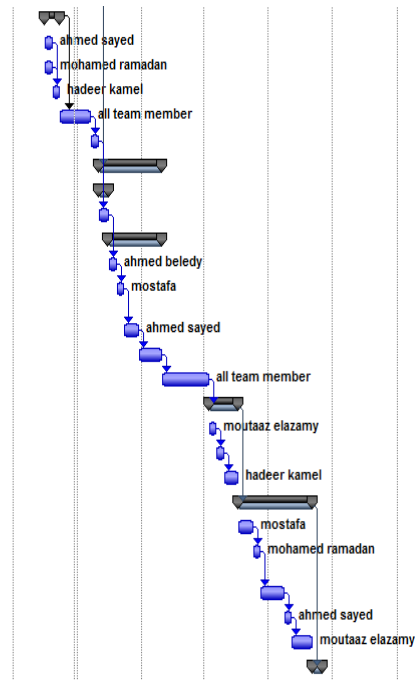


Figure 3-2: Plan.

3.4 Feasibility study

3.4.1 Executive Summary

Patient share contribute in solving one aspect of the most important problems in our country “The health sector problem” in first stage solve blood shortage by providing those who need any type of blood with the type they need by showing the closest persons who can donate their blood within his geographical area . Patient share has a leadership and surpassed its competitors by its new features and services. It will be in a good position in online marketplace and by performing an aggressive marketing plan and some awareness campaigns; we can ramp up the project’s growth projections for the near future.

3.4.2 Description of Products and Services

Patient share connects three entities together the patient, the donor and the hospital .it provides its users with many services. It connects the patient with the blood donors who can donate the same type of blood that the patient needs within the patient's geographical area. When the patient writes the type of blood he needs, it sends requests to the donors with the same type he needs within his geographical area so he can reach the largest number of donors. The hospital provides the patient with a code so the donor makes sure that the blood he donates reaches those who really need it. Patient share maintains a friendly user interface for the application and web interface for hospital.

3.4.3 Technology Consideration:

Upgraded technological capability will be required for patient share to execute its mission. It is a technological solution for blood shortage problem. The patients need an easy and simple way to reach the number of blood donors they need. Patient share maintains many functions whose targeted users need to know, such as the closest blood donors with the same type of blood the patient needs route and vehicles that are appropriate for their destination. Patient share currently maintains a friendly user interface for the application and for the web interface.

3.4.4 Product/Service Marketplace

The online marketplace for public applications is still booming and will prosper more. Our marketplace consists of mobile iOS application that provides many services, and targets the following:

- Patients who need blood.
- Blood donors.

There is no competitors in Egypt but there are some competitors in another countries such as "life saver blood donation", "blood bank Bangladesh", "بنك الدم الالكتروني" but they provide inefficient solutions, in our points of view .they are Not well marketed, users can see other users data so it is not secure and it does not use Google maps and Google navigation so it is hard to reach the suitable users.

. Patient share solved the big issues that face the competitors and provide more variety features that will help end-users and enhance their experience. Patient share will be placed in Play Store and apple store to be installed and used by users. It will be available anywhere and anytime when a user want to access its services.

3.4.5 Marketing Strategy

In order to be successful, patient share must differentiate itself from competitors in order to appeal to users in the online marketplace. To do this, patient share will utilize its solutions and services that solve the problems of competitors in addition to more features. Patient share uses Google maps so it shows the closest donors to the patient's geographical area. One of the big challenges is the trust between the users and the applications, many users almost do not trust in donating through applications, but this is solved in patient share .the hospital must give the patient a code he enters to make sure that the donors blood goes to those who really need it.it also hides the data of the donor so it is secure. That will achieve an important goal in reinforce the relationship between patient share and its users and that will help us more in the technical and marketing sides as the competitors do not provide these services in an efficient way. They show the donors data to the public so they are not secure and they don't use Google maps so they don't suggest the suitable donors . None of our competitors provides our services, so patient share has the leadership in this area. The revenue will come from ads that will be provided through the patient share Mobile App.

3.4.6 Organization and Staffing

Patient share has an integrated team each of whom has an effective role. Positions distributed according to each person's ability to perform the assigned tasks.

Position #1: Team Leader – He is who manages, distributes tasks, handles team members, solve internal problems and leads the team well.

Position #2: Back-end Developers – They are who responsible for technical side such as back-end and coding. Briefly, they are strong in the technical side and writing the system code.

Position #3: Front-end Developers – They are who create the user interface design and implement its code. Take into consideration the coordination of fonts, colors and make them in a good image to be suitable and easy to use. They make the application friendly for users.

Position #4: Database administrator – is a person who design the database and implement it. He has the authority to manage and maintain the database.

Position #5: System analyst is who reports and documents all the steps that happen in the project.

3.4.7 Schedule

The following is a high-level schedule of some significant milestones for this initiative:

Oct 1, 2018 : Initiate Analysis and requirements phase.

Nov 21, 2018: Project Design phase.

Jan 20, 2019: Initiate project implementation.

Mar 5, 2019: Complete implementation of application& web interface
Mar 6, 2019 : Testing trials of application& web interface

Jun 12, 2019: Go live with application launch

3.4.8 Financial Projections

The financial projections for the cost of patient share requirements are highlighted in the table below. These figures account for projected requirements, shipping, material, and insurance costs

Project Poster	100 LE
Documentation	500 LE
Tools license	100LE

Table 3-1: Financial Projections.

3.4.9 Findings and Recommendations:

Based on the information presented in this feasibility study, it is recommended that patient share approves the online awareness initiative and begins project initiation. The findings of this feasibility study show that this initiative will be highly beneficial to us and has a high probability of success. Key findings are as follows:

Technology:

Utilize new technologies and fields that will be helpful in patient share development make patient share more secure, Attention to security and privacy.

Marketing:

Patient share is able to differentiate itself from its competitors and will create new features to target new consumers. This initiative will allow patient share to reach large number of target groups electronically. Enhance marketing strategy according to persona .Working on how to be a big brand.

Organizational:

New facilities or capital investments are required.

Financial:

Patient share will be in position to capture greater market share by maintaining both services development and online presence.

3.4.10 Risk

The main risks types facing development of this system are the follows:

Time risk

this is due to compression of schedule of the project.

Optimistic evaluation of time estimates.

Dropping off important tasks from estimates.

Budget risk

this is due to un-predictable developing cost since project requires several experiments which can't be determined at earlier stages

Out time

This can happen due to incomplete, incorrect, inconsistent, volatile or unclear requirements

Culture risk

Peoples of the Middle East do not have the culture of donating and doing good by using application

3.5 Team structure

- **Moataz Muhammad Al-Azamy (Team leader and Mobile app)**
- **Mostafa Mohamed Mohamed (Node JS developer)**
- **Mohamed Ramadan Hassanein (Firebase)**
- **Hadeer kamel El-Sayed (UI –UX design)**
- **Ahmed Emad Hamdy (System analyst)**
- **Ahmed Sayed Anwer (Front end web)**

Chapter Four

SYSTEM ANALYSES AND DESIGN

4.1 OVERVIEW

This chapter includes system analysis and design. It describes how the system works through discuss functional and non-functional requirements, Use Case diagrams, Sequence diagrams, System sequence, Context diagram, etc...

This chapter is divided into 11 sections:

- Section 4.1 Overview.
- Section 4.2 Requirements and Analysis.
- Sections 4.3 use case diagram Design.
- Section 4.4 Use case Scenario.
- Section 4.5 System Sequence and Sequence.
- Section 4.6 Activity Diagrams.
- Section 4.7 Context Diagram.
- Section 4.8 Domain Model.
- Section 4.9 Data Flow Diagram (DFD).
- Section 4.10 Class Diagram.
- Section 4.11 Entity Relationship Diagram (ERD).

4.2 REQUIREMENTS AND ANALYSIS

This section discusses how the main functions work and includes its analysis to make the system clearer for readers and understand what happen.

4.2.1 Functional Requirements

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used. Functional requirements such as:

4.2.2 Non-Functional Requirements

A non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are often called "quality attributes" of a system. Non-Functional requirements such as:

- **Performance** Appears in your using for the app. Performance results from the good writing code and right algorithms, included fast and usability.
- **Maintainability** Appears in the maintenance of the application, maintain it, attention to all aspects related to it and ensure that it is the target designed for it.
- **Security** Shown in the protection of the application from hacker, detect the gaps first, and work to repair them and choose the correct algorithms that reduce the possibility of the hack.
- **Usability** App can respond within 5 seconds through app screen.
- **Reliability** App should be available all time.

4.3 Use Case Diagrams

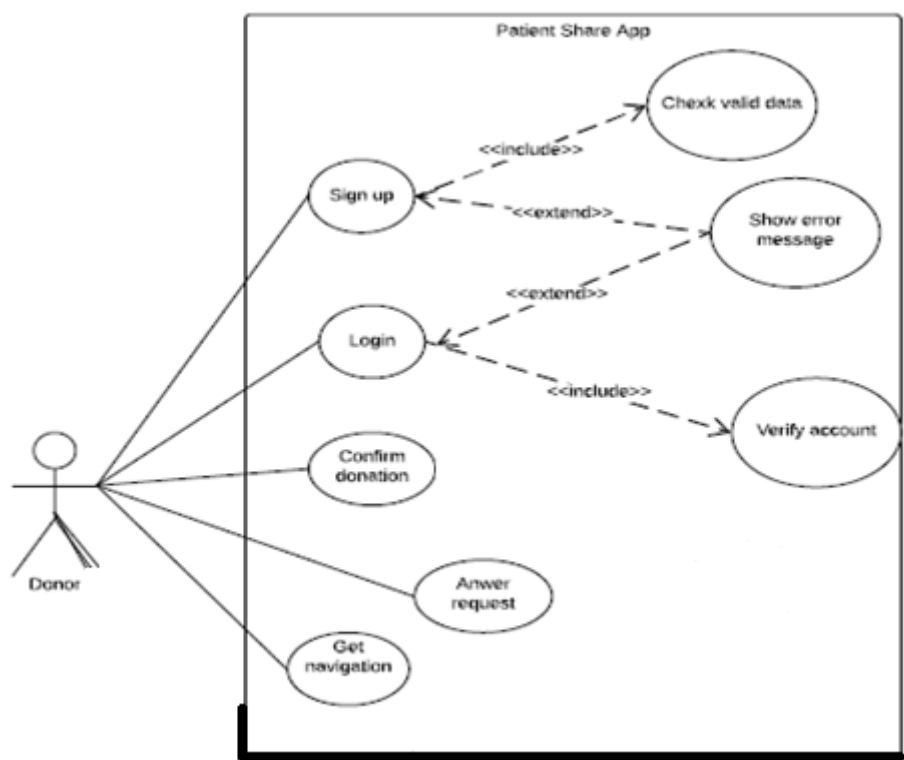


Figure 4.1: Use Case (Donor)

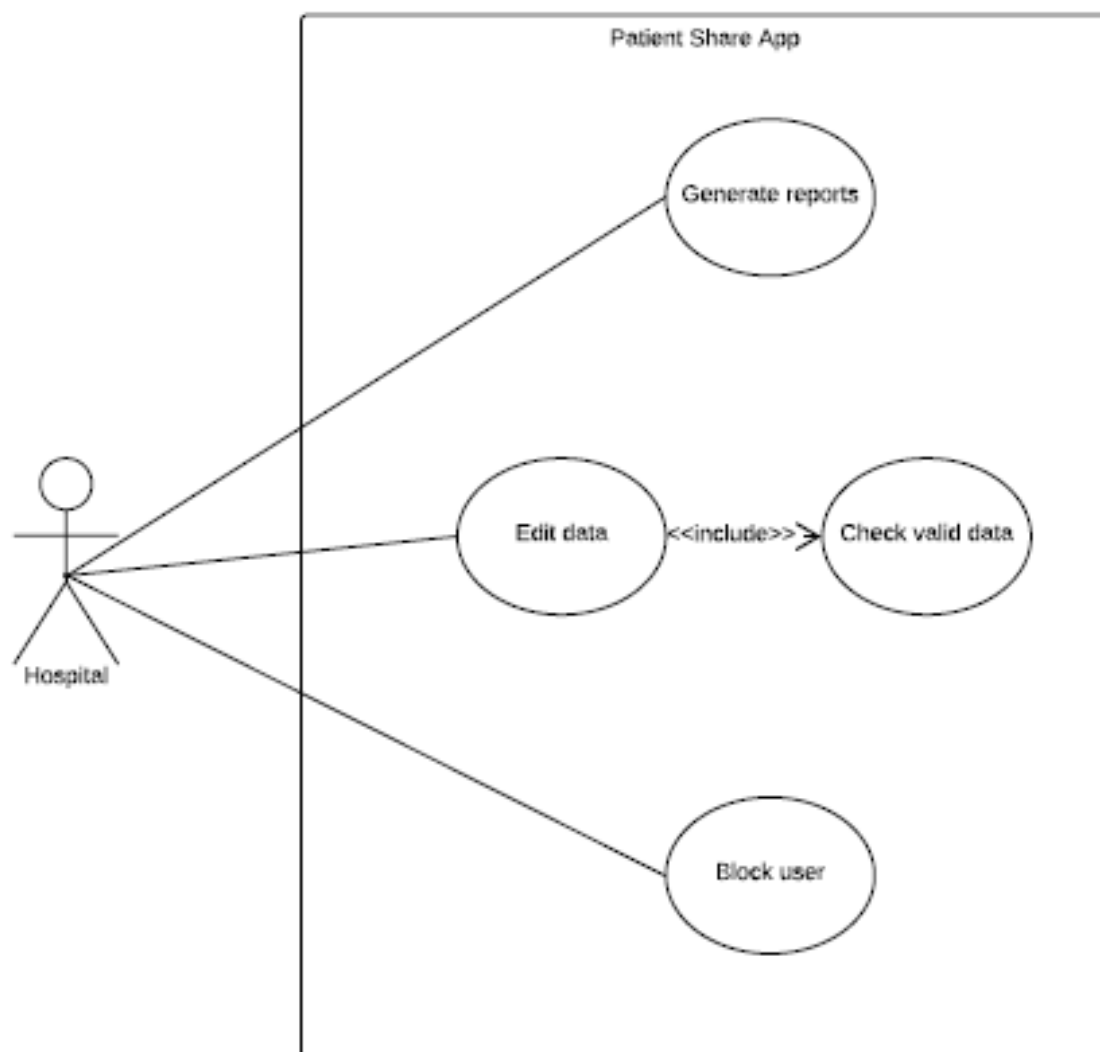


Figure 4.2: Use Case (Hospital)

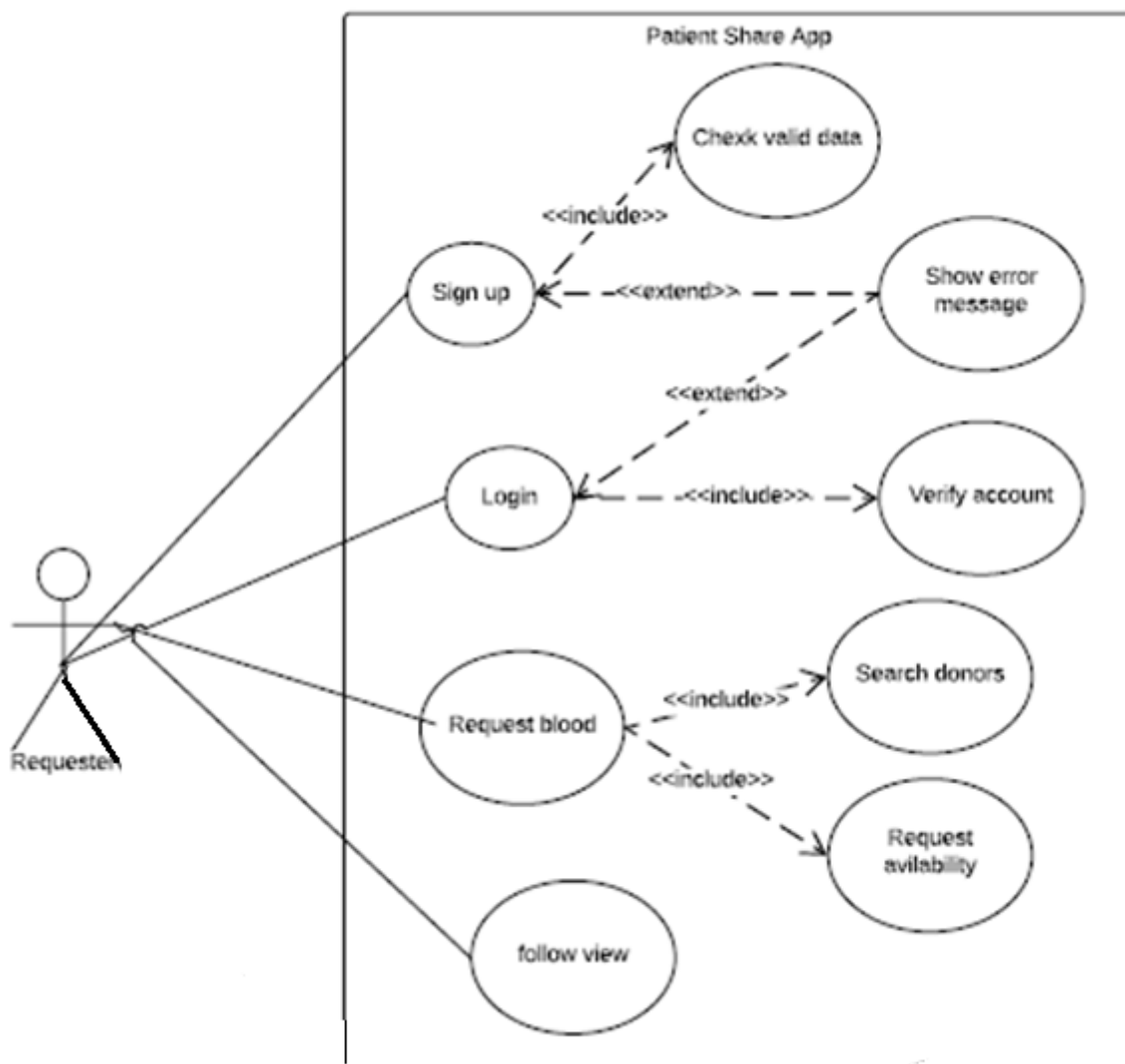


Figure 4.3: Use Case (Requester)

4.4 Use case Scenario

Use Case Name:		Sign up
Actor(s):	User	
Description:	A user can login to the application	
Flow of event	Actor Action	System Response
	Step1: download app for users then write URL of the site in browser	Step2: home page appear
		Step 4 : sign up forum
	Step3: clicks the Sign up button.	Step7: check that data valid and complete
	Step5: user fill sign up data	
	Hospital sign up by admin	
Alternative Flow:	Step6: clicks the confirm button.	
	Step7: data not complete or not valid show error message	
	And form appear again	
Precondition:	-	
Post condition:	- Has successfully sign up	
Assumption:		

Table 4-1: Use Case(Signup)

Use Case Name: Log in	
Actor(s):	User
Description:	A user can login to the application
Flow of event	Actor Action System Response
	Step1: open application or enter URL Step2: home page appear
	Step 4 : sign in form appear
	Step3: clicks the Sign in button. Step7: check and verify account data
	Step5: user enter name and password
	Step6: clicks the login button.
Alternative Flow:	Step7: user name or password error show error message to user and forget password button appear And form appear again
Precondition:	user sign up Once before
Post condition:	User account interface appear
Assumption	

Table 4-2: Use Case(Login)

Use Case Name: Request blood	
Actor(s):	User
Description:	A user can request blood from other user in the application
Flow of event	Actor Action System Response
	Step1: log in to app Step2: account interface appear
	Step3: clicks to request button
	Step 4 : fill the data Step5: system will send code to hospital to make sure that donor need blood

	<p>(data contain information about blood that he need and hospital) and confirm</p> <p>Step7:person who requests blood should enter the hospital confirmation code</p>	<p>Step 8: system will do some operation after confirmation code operation</p> <p>first operation system search on donors Who are located at the nearest distance from the patient that request blood</p> <p>Second operation system will check availability</p> <p>Check availability mean that system will review history of donor and make sure that donor not donate blood 56 day ago and donor not in black list (list generate from hospital)</p>
Alternative Flow:	<p>Step 4: data not valid send error message and form appear again</p> <p>Step 8: if user enter wrong code 3 times request operation canceled</p> <p>Step9:</p>	
Precondition:	<p>-have an account Existing in hospital</p>	
Post condition:	<p>System send notification to user that can donate blood</p>	
Assumption:		

Table 4-3: Use Case (Request Blood)

Use Case Name:	Answer request	
Actor(s):	User (donor)	
Description:	Answering blood request	
Flow of event	Actor Action	System Response
	Step1: user receive notification from system and answer accepting or not	Step3:system will calculate that acceptance request suitable for number of requests if not send to request function demand more blood Step4:system put people that accept to list called final list
Alternative Flow:		
Precondition:	System receive list sent from system contain all available user that can donate	
Post condition:	Valid and accept list generate called final list	
Assumption:		

Table 4-4: Use Case (Answer Request)

Use Case Name:	Get navigation	
Actor(s):	Donor	
Description:	Google navigation appear to donors	
Flow of event	Actor Action	System Response
	Step1: donor ask to know place that he can donate	Step2: system sent notifications to donor contain Google map
Alternative Flow:		
Precondition:	User in final list	
Post condition:	- donor receive Google map	
Assumption:		

Table 4-5: Use Case (Get navigation)

Use Case Name: Follow view		
Actor(s):	Requester	
Description:	Map sent to requester, map contain real time map of donors	
Flow of event	Actor Action	System Response
	Step1: request map of donor	Step3: follow and draw moving of member of list in screen to requester
Alternative Flow:		
Precondition:	Final list	
Post condition:	Request have a map contain movements of donors	
Assumption:		

Table 4-6: Use Case (Follow view)

Use Case Name: Generate reports		
Actor(s):	hospital ,requester	
Description:	Generate report about every donate process	
Flow of event	Actor Action	System Response
	Step1: hospital sent report about every donation IF complete successfully and donors status if donors have any issues put him in black list and sent to system	Step4:system take report and write some statistics and take issues and problem to administrator
	Step 2:requesrter give system feedback	Step 5: if any donator Failure to go on time more than once system send him a warning message and deduce from his score and if user missing donate system put him in black list
Alternative Flow:		
Precondition:	Donation complete	
Post condition:	System record what happen in every donate operation	
Assumption:		

Table 4-7: Use Case (Generate reports)

Use Case Name:	Block user	
Actor(s):	Hospital	
Description:	User cannot donate blood	
Flow of event	Actor Action	System Response
	Step2:user ask to know block reason	Step1:system stopped account from donation operation and sent notification
		Step3:system send to user the reason Account stopped until problem solve
		System sent notification include health status that block user
Alternative Flow:	Step 1:data not valid sent error dot make change until having valid data	
Precondition:	Block list	
Post condition:	User cannot donate blood in case of healthy reasons	
Assumption:		

Table 4-8: Use Case (Block user)

Use Case Name: Edit data		
Actor(s):	hospital	
Description:	Record new data	
Flow of event	Actor Action	System Response
	Step1: hospital change this user data build on information received from his donation operation	Step2:system accept any valid account from hospital
Alternative	Step1: data not valid system use the previous data	

Flow:	
Precondition:	Account exists
Post condition:	New data about user
Assumption:	

Table 4-9: Use Case (Edit data)

4.5 System Sequence and Sequence

4.5.1 System Sequence

A system sequence diagram (SSD) is a sequence diagram that shows, For a particular scenario of a use case, the events that external actors generate, Their order and possible inter-system events.

Front-end:

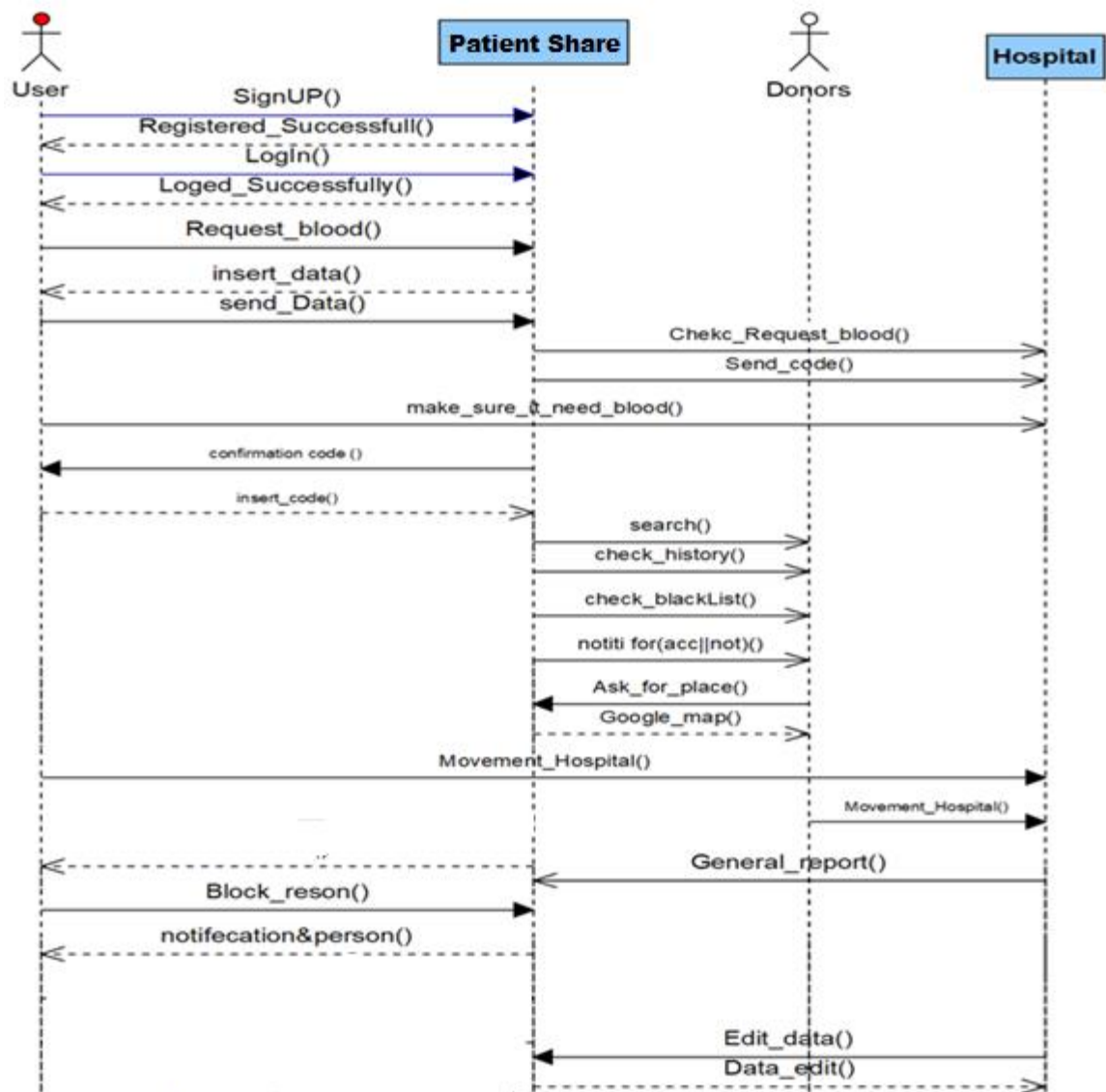


Figure 4.4 System Sequence Diagram (Front-end)

Backend:

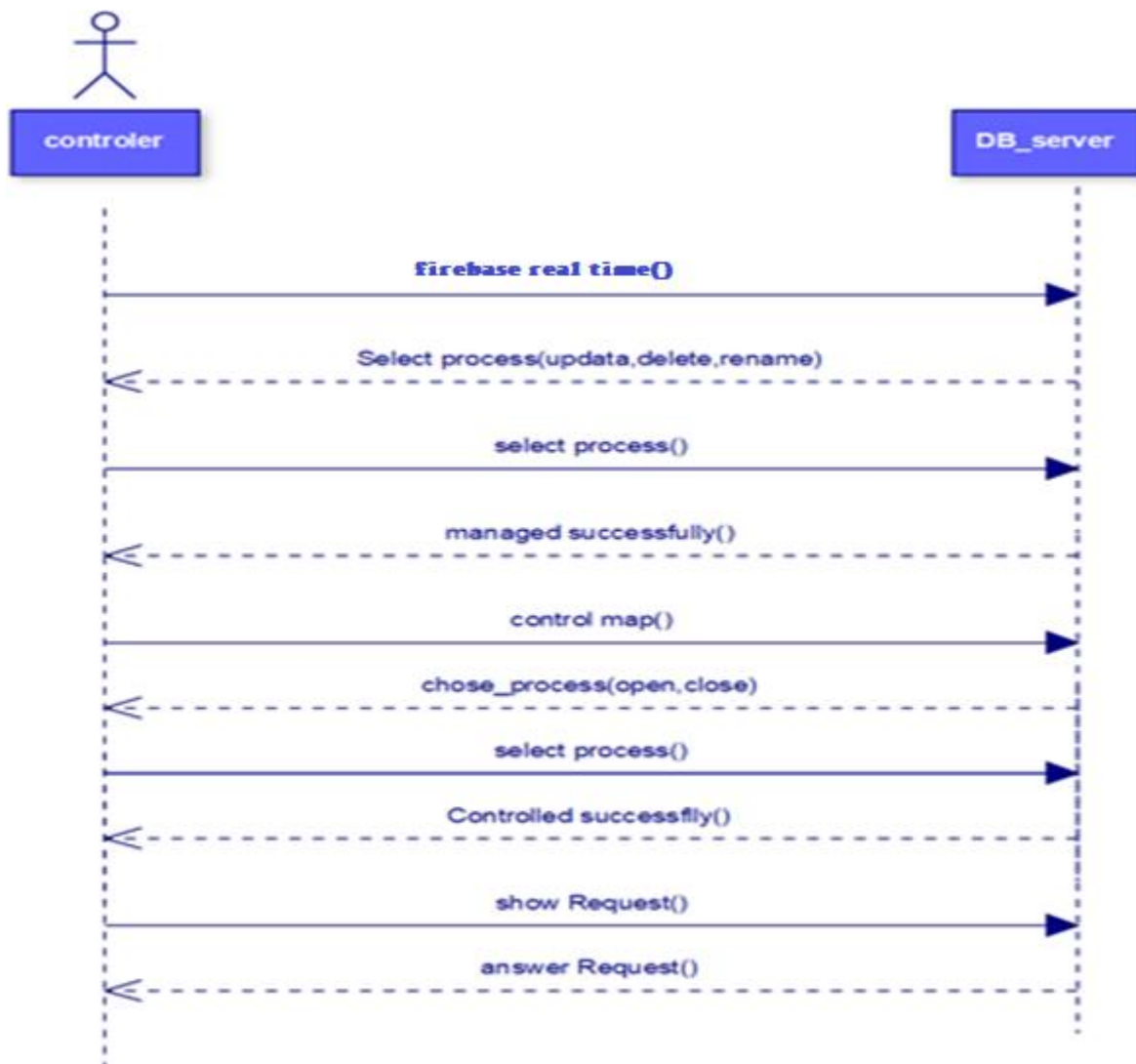


Figure 4.5 Sequence Diagram (Backend)

4.5.2 Sequence

A sequence diagram shows object interactions arranged in time sequence.

It depicts the objects and classes involved in the scenario and the sequence of

Messages exchanged between the objects needed to carry out the functionality of the scenario

Front end:

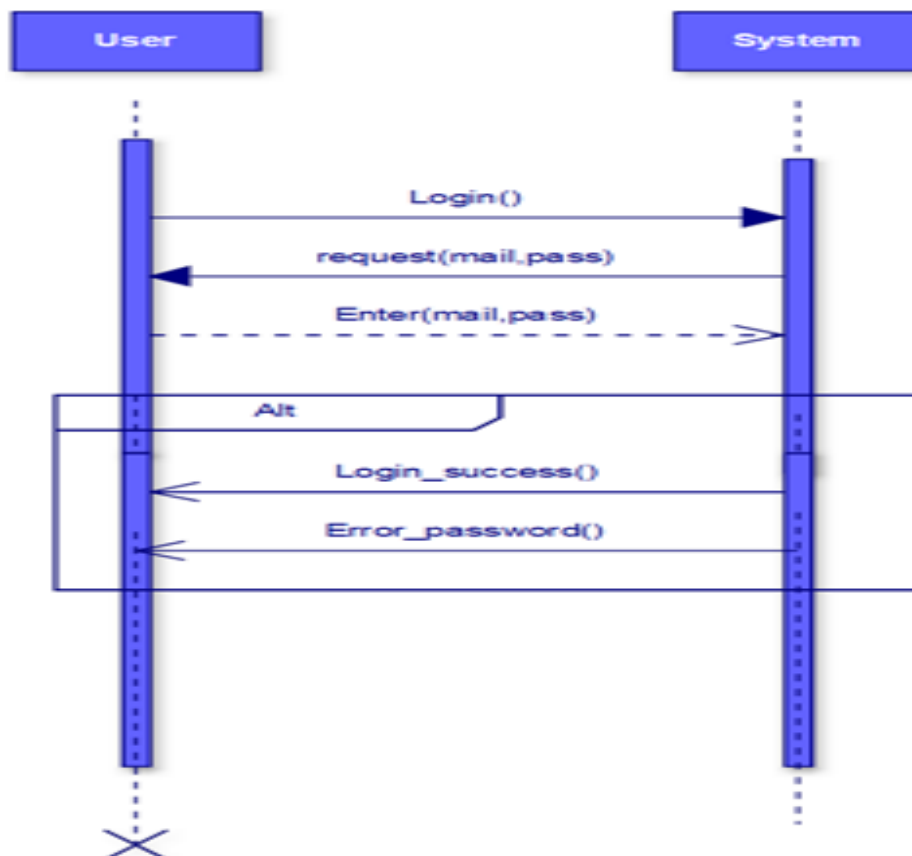


Figure 4.6 Sequence Diagram (Login)

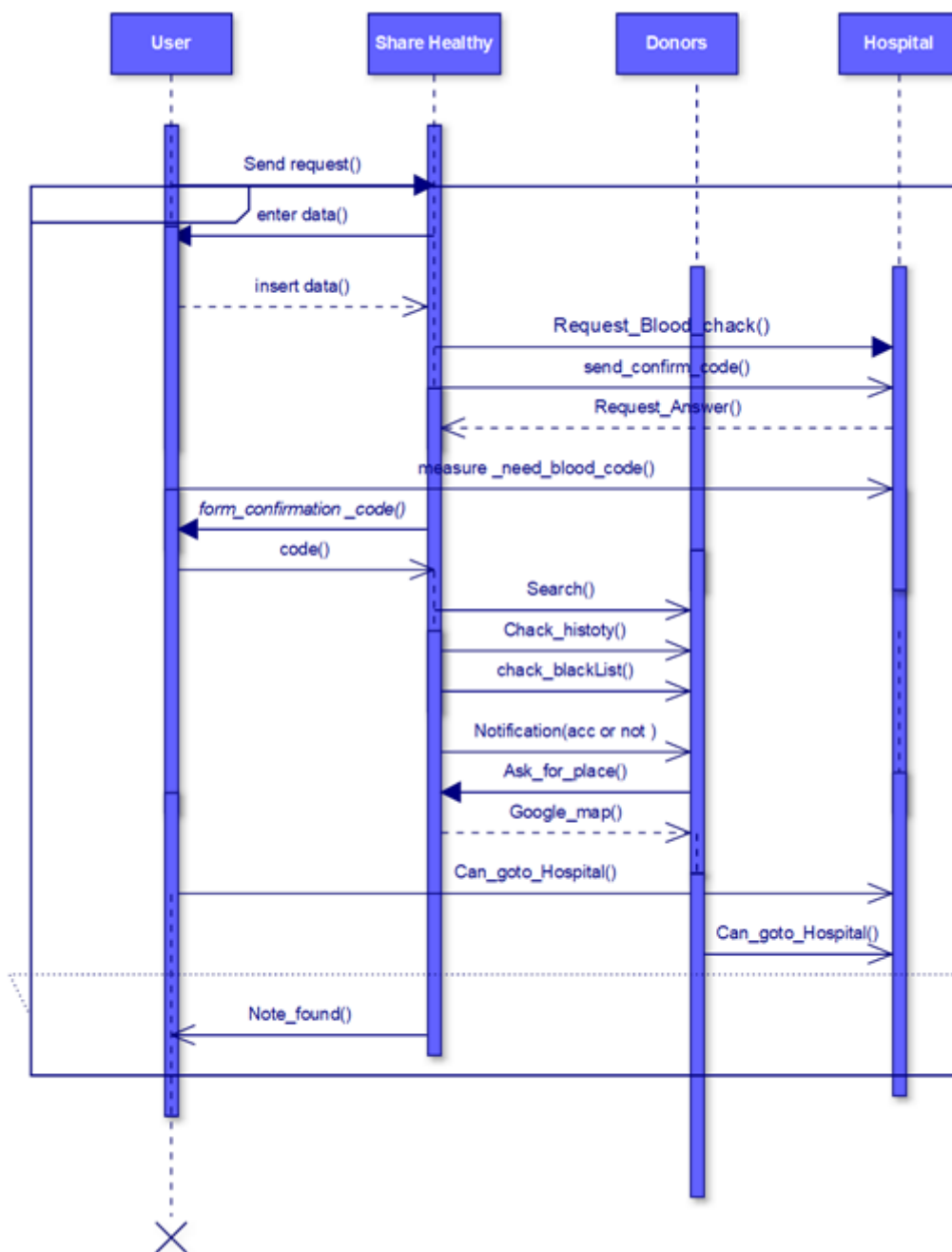


Figure 4.7 Sequence Diagram (Request blood)

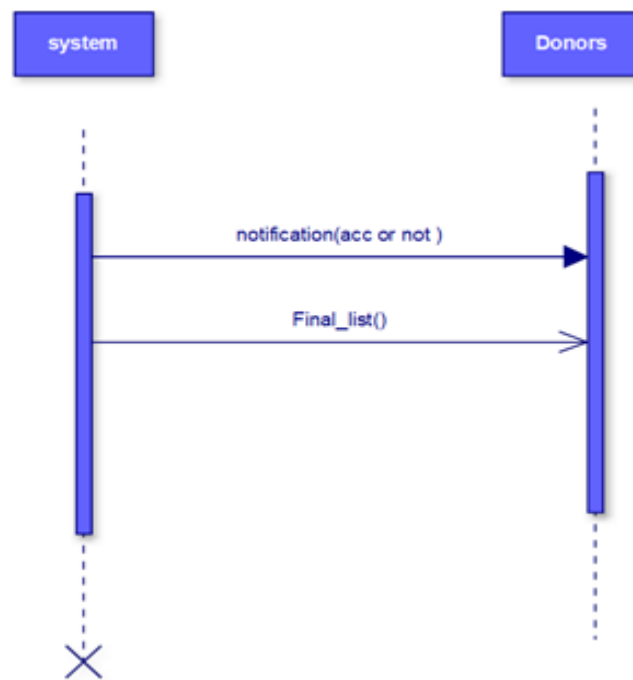


Figure 4.8 Sequence Diagram (Request Answer)

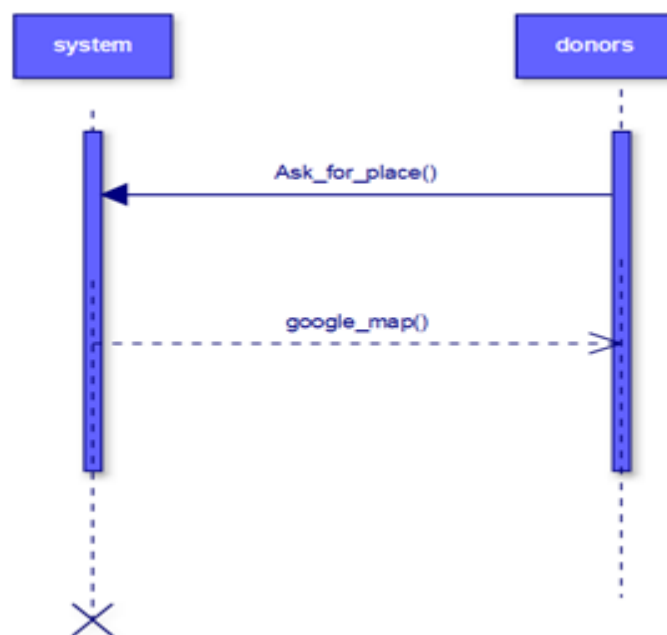


Figure 4.9 Sequence Diagram (Navigation)

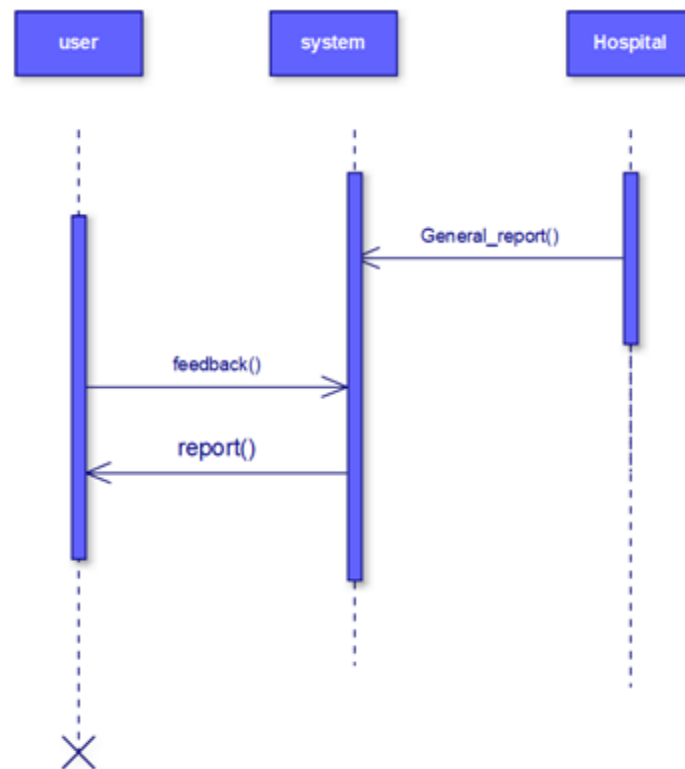


Figure 4.10 Sequence Diagram (Reports)

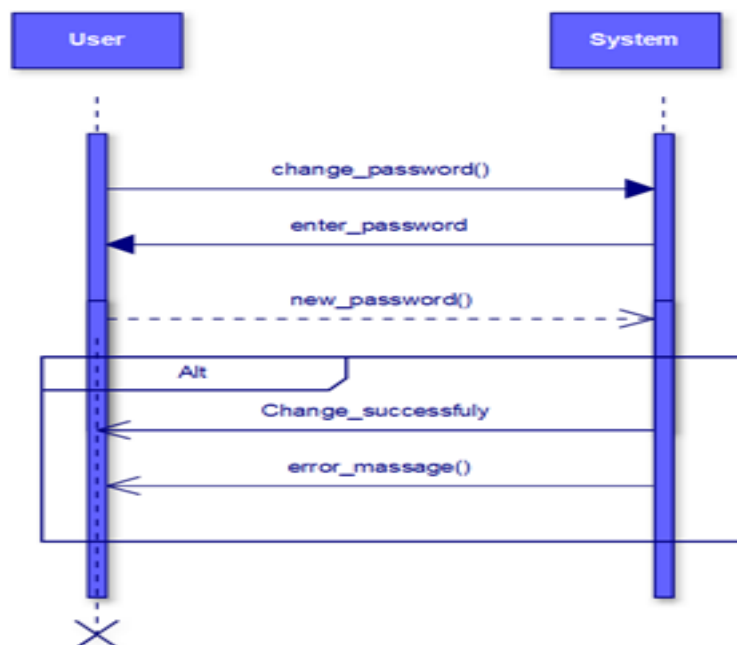


Figure 4.11 Sequence Diagram (change password)

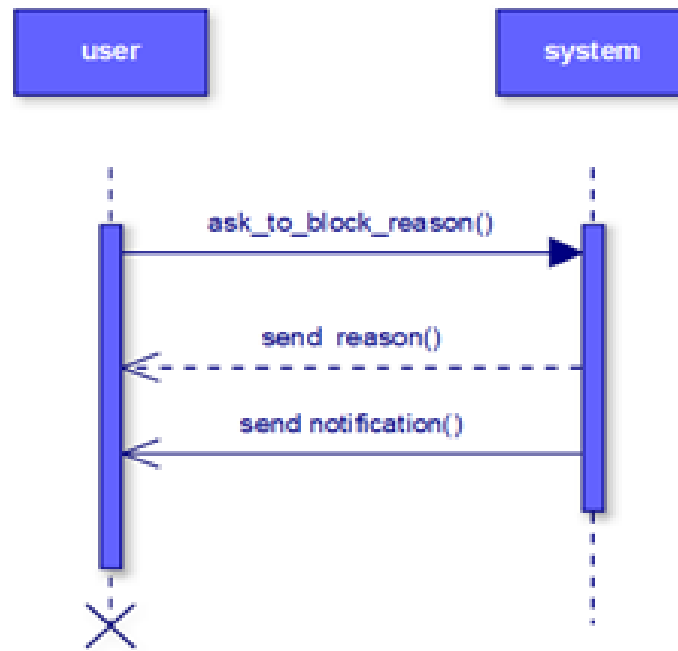


Figure 4.12 Sequence Diagram (Block reason)

4.6 Activity Diagrams

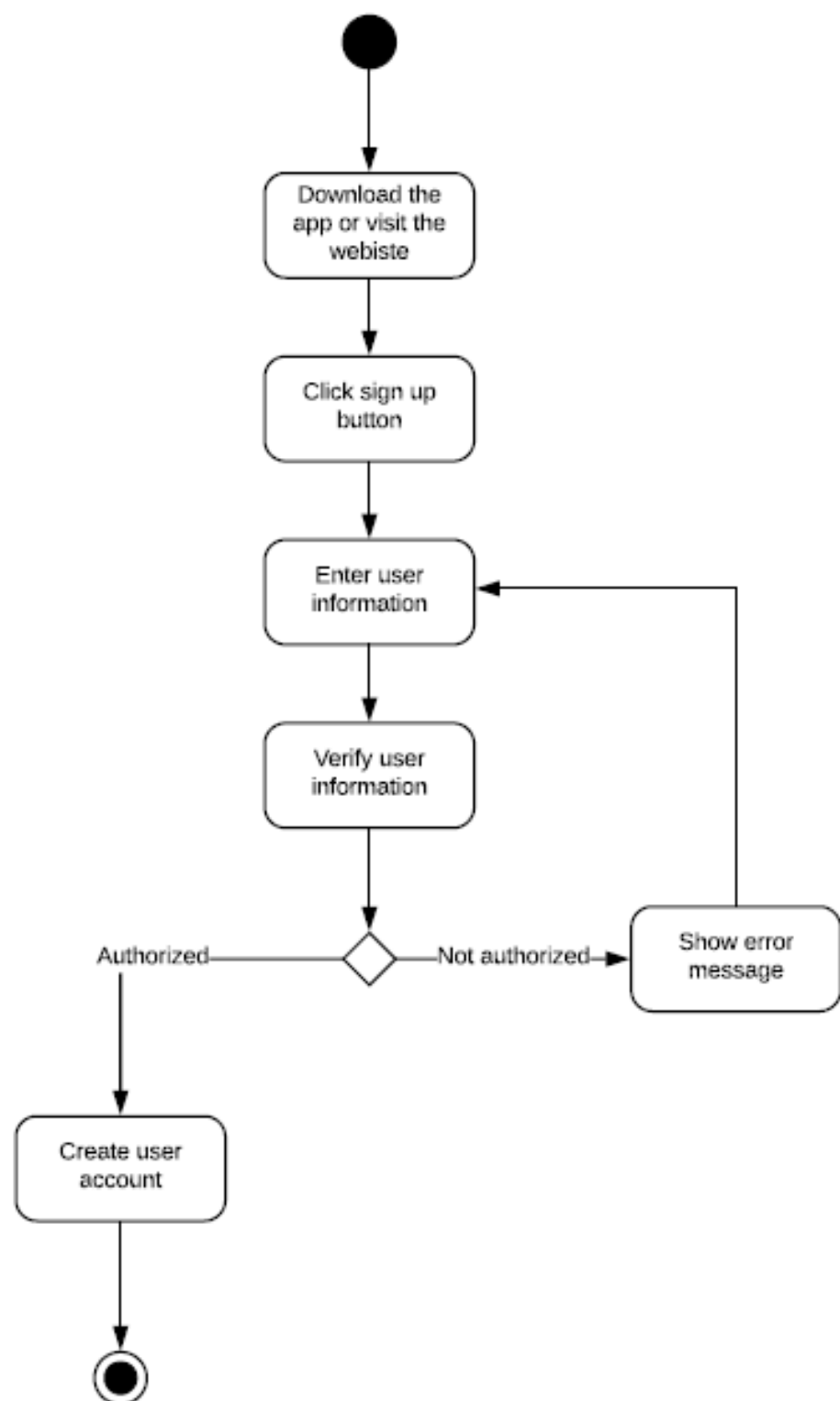


Figure 4.13 Activity Diagram (Create user)

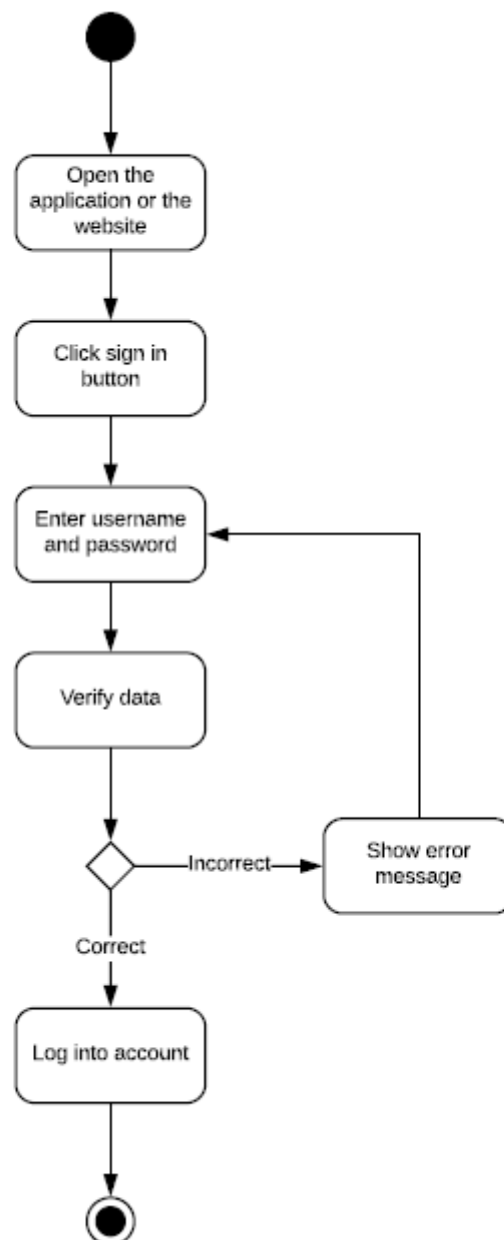


Figure 4.14 Activity Diagram (Login)

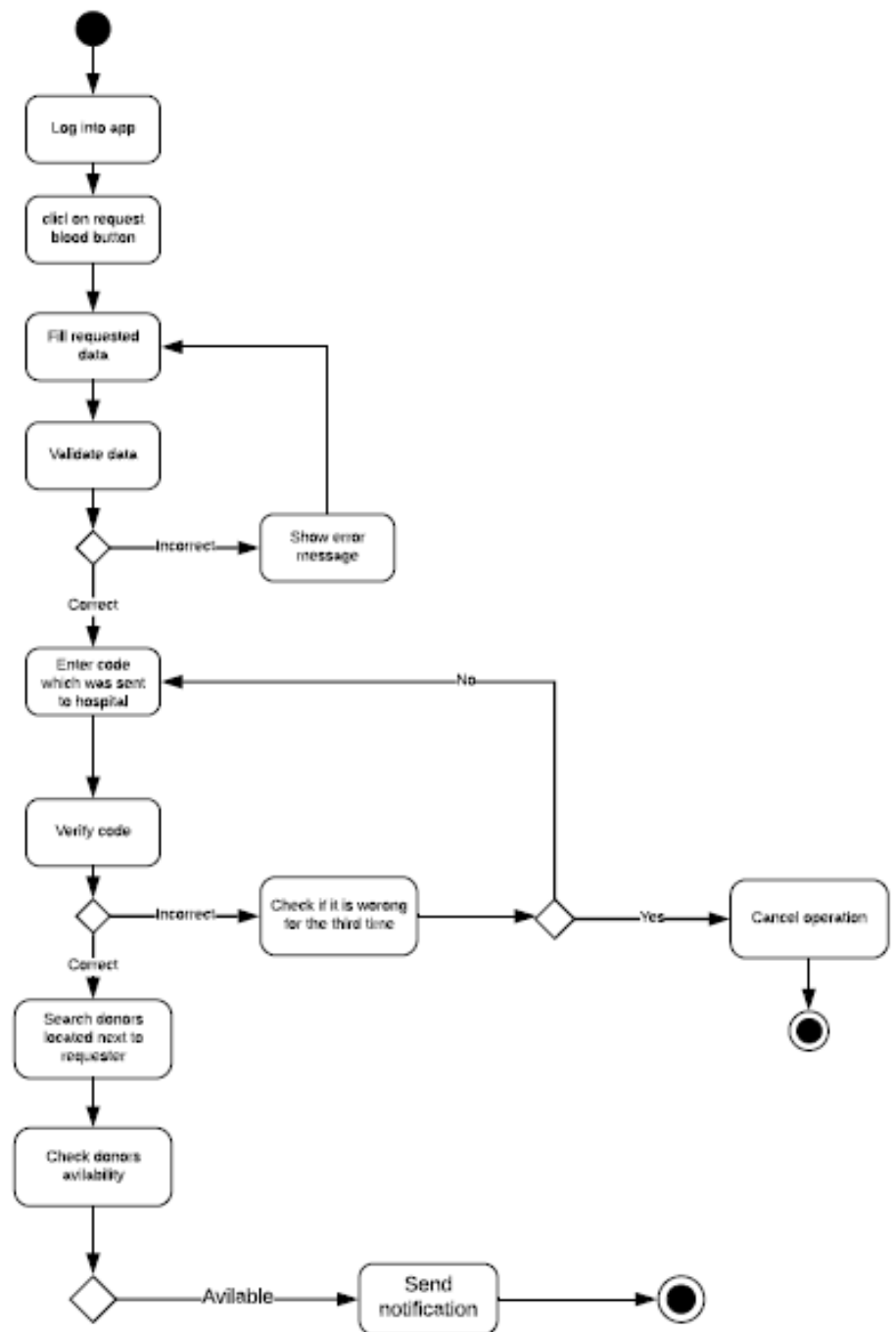


Figure 4.15 Activity Diagram (request blood)

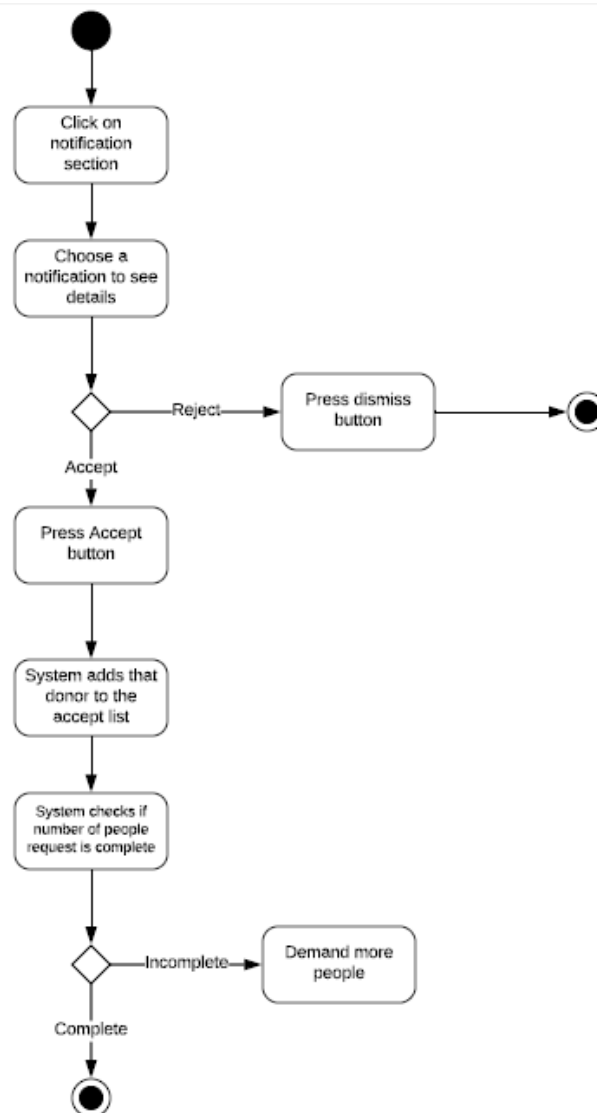


Figure 4.16 Activity Diagram (Accept request)

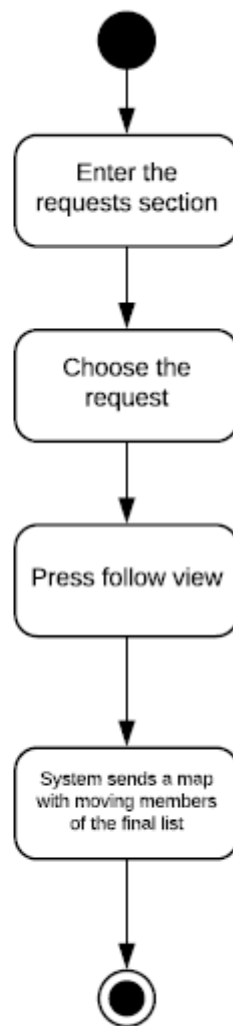


Figure 4.17 Activity Diagram (Follow view)

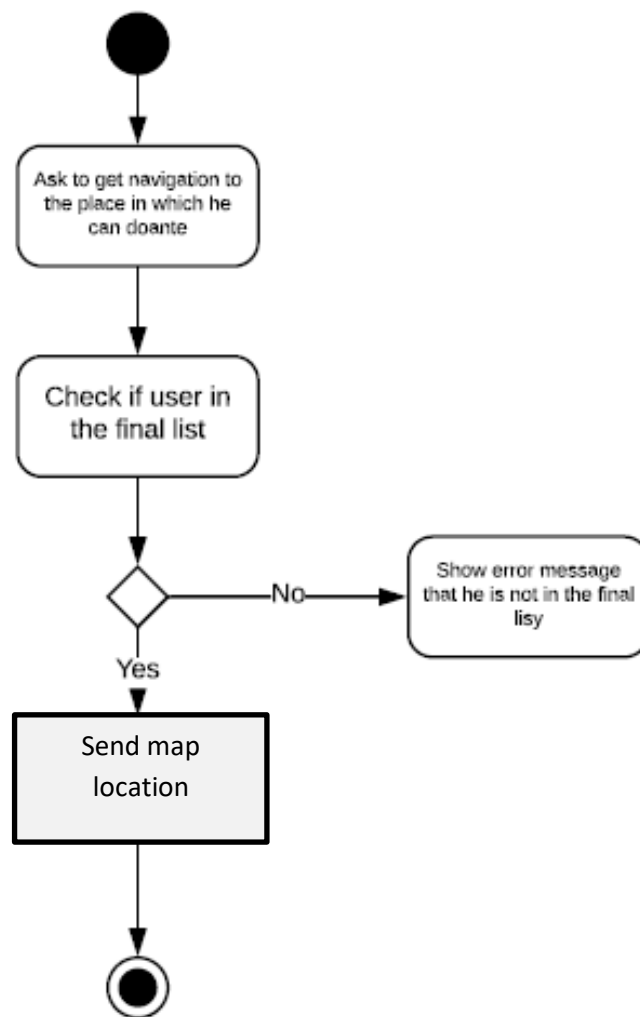


Figure 4.18 Activity Diagram (Navigation)

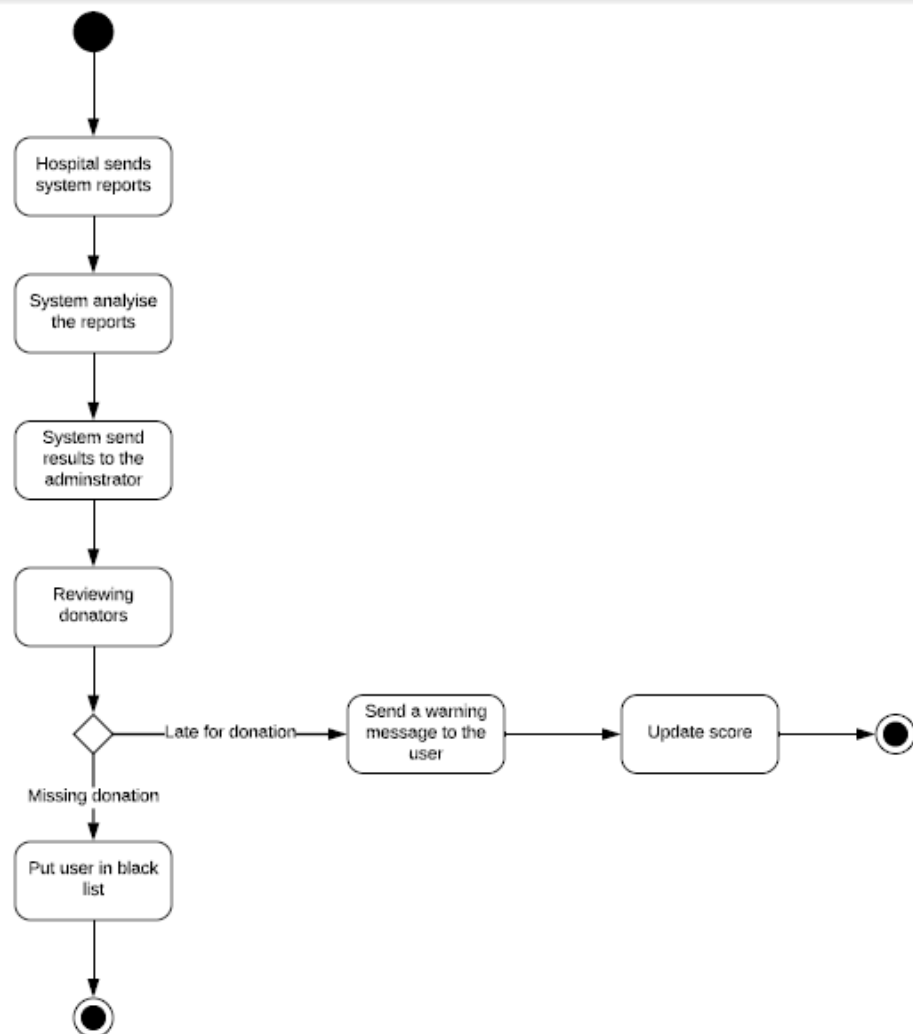


Figure 4.18 Activity Diagram (Generate report)

4.7 Context Diagram

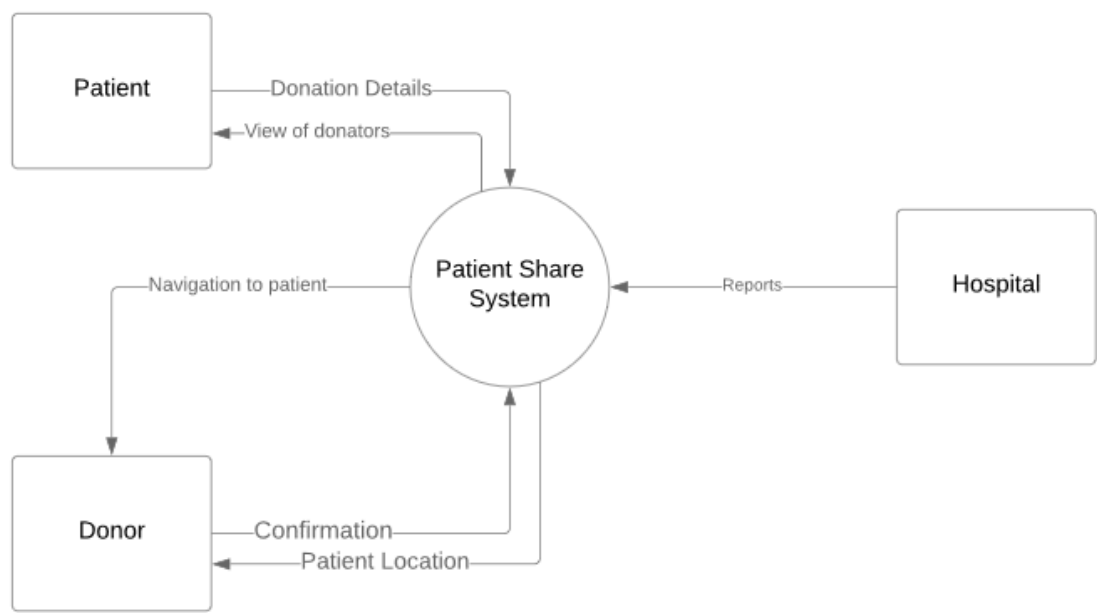


Figure 4.19 Context Diagram

4.8 Domain Model

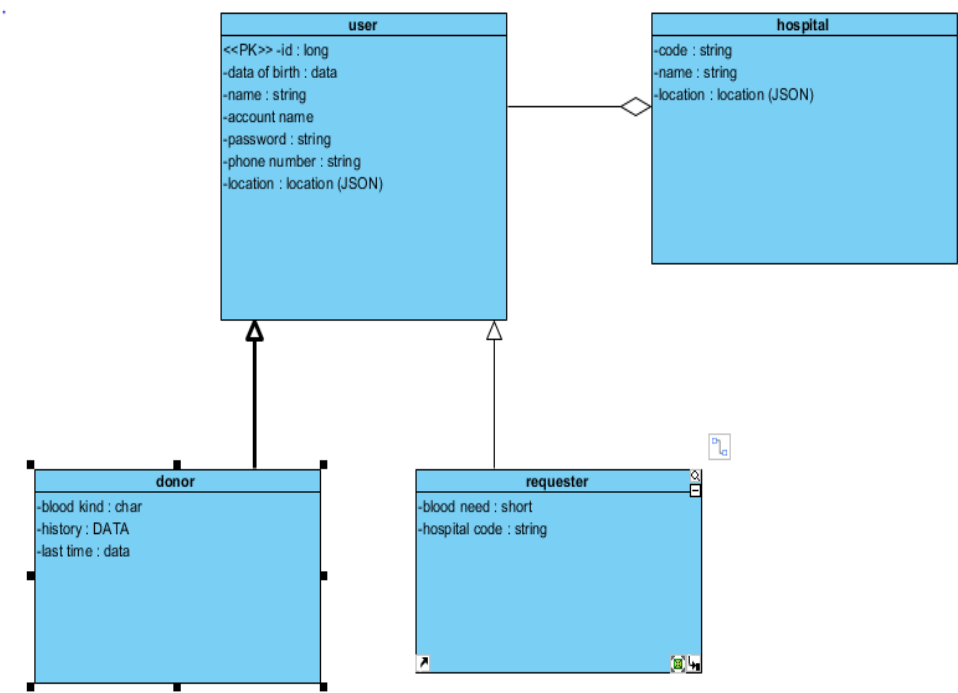


Figure 4.20 Domain Model

4.9 Data Flow Diagram

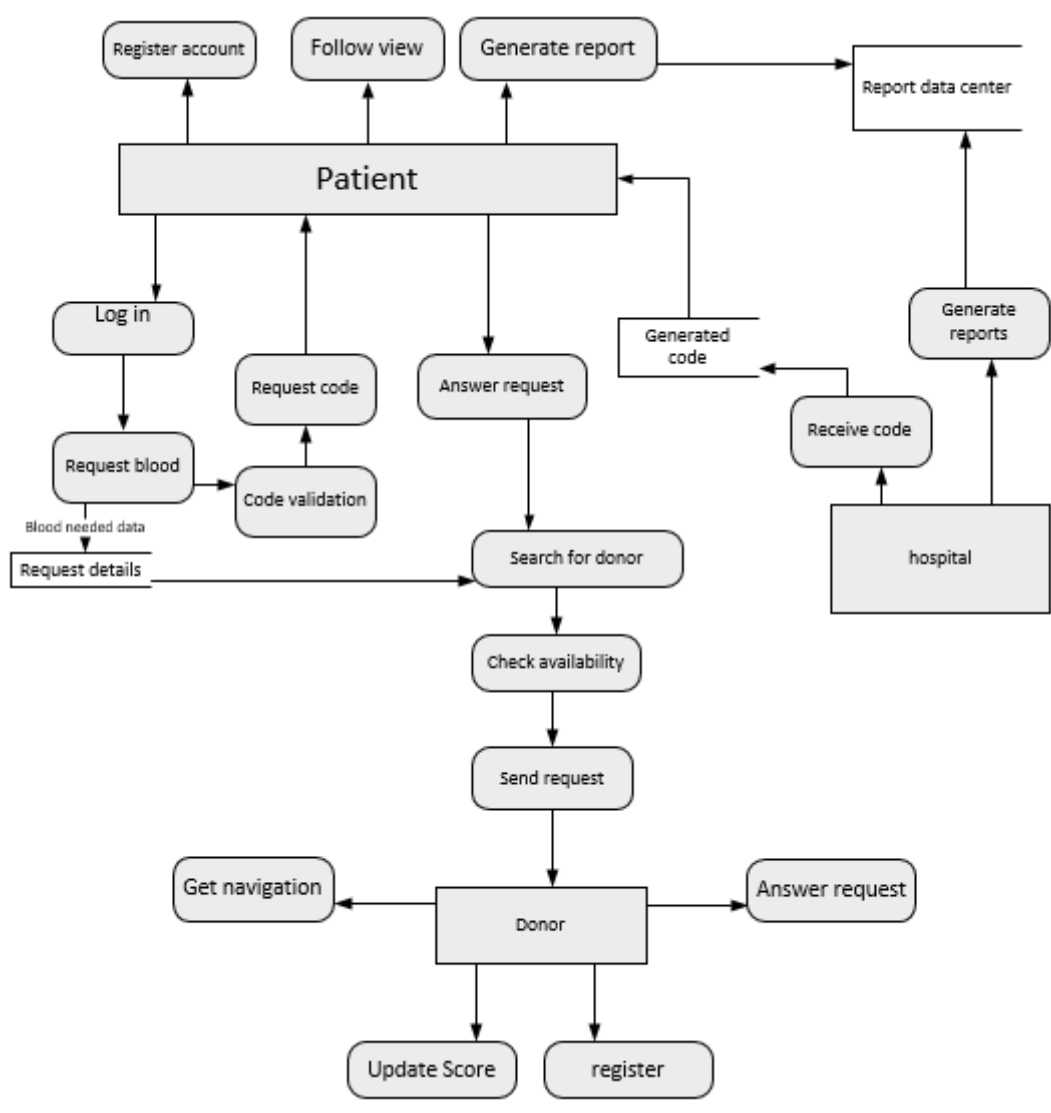


Figure 4.21 DFD

4.10 Class Diagram

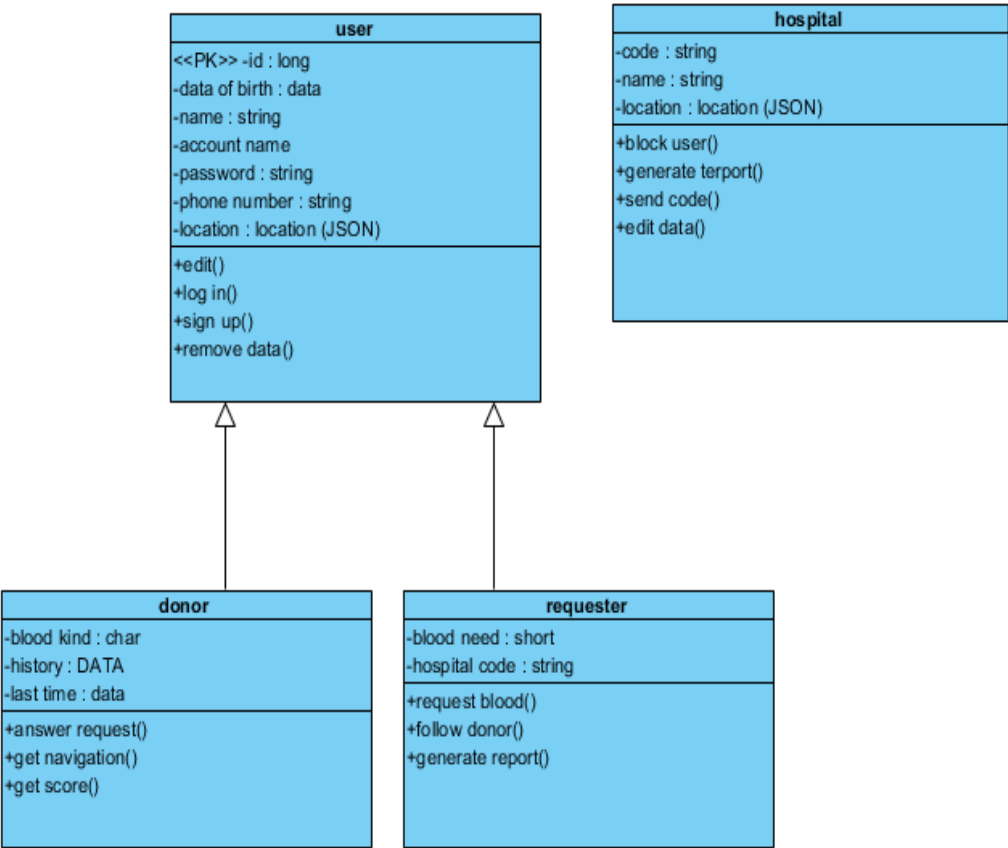


Figure 4.22 Class Diagram

4.11 Entity Relationship Diagram

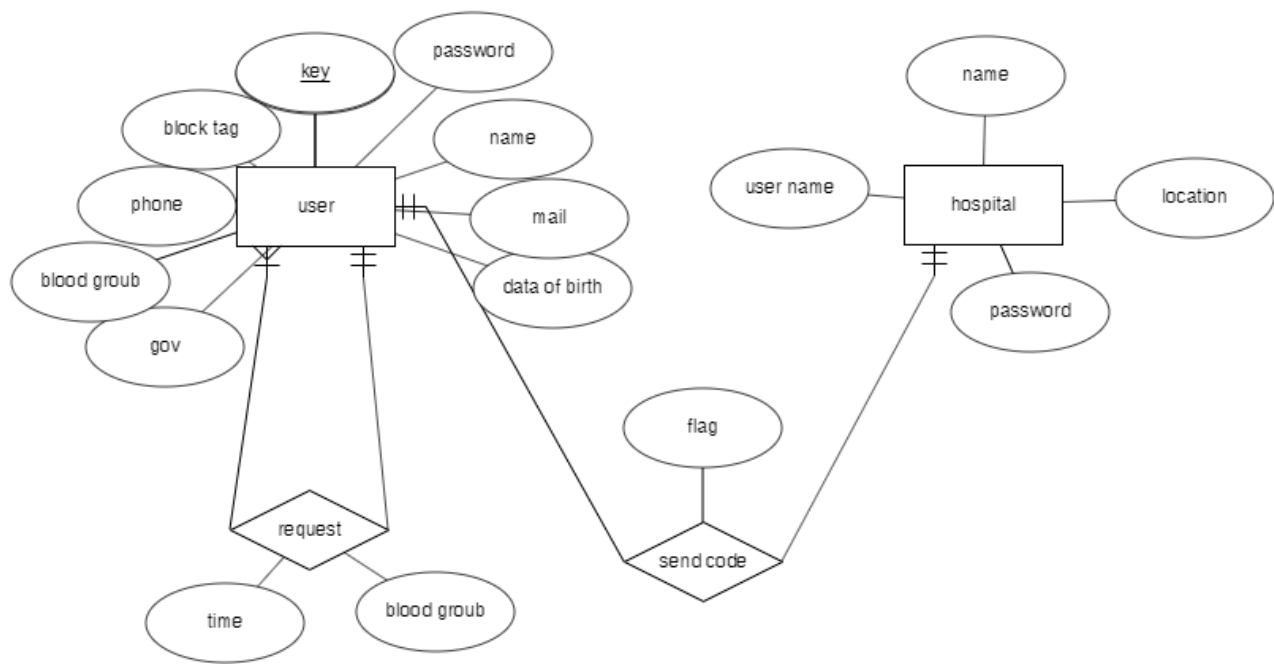


Figure 4.23 ERD

Chapter Five

SYSTEM ARCHITECTURE

5.1 Overview

This chapter includes the system architecture and how it works. Discuss the relationship between each component and how they integrate to provide a solution. User sends his location and the desired destination through a mobile application. User's data such as position and destination are stored in the database. Our algorithm will operate on these data to find the best candidate transportation line to take to reach their destination. Here, the entire process will be clearer through detailed sections.

This chapter is organized into 5 Sections

- Section 5.1 Overview
- Section 5.2 Mobile Application
- Section 5.3 database
- Section 5.4 Web Interface
- Section 5.5 system screenshot

System Architecture

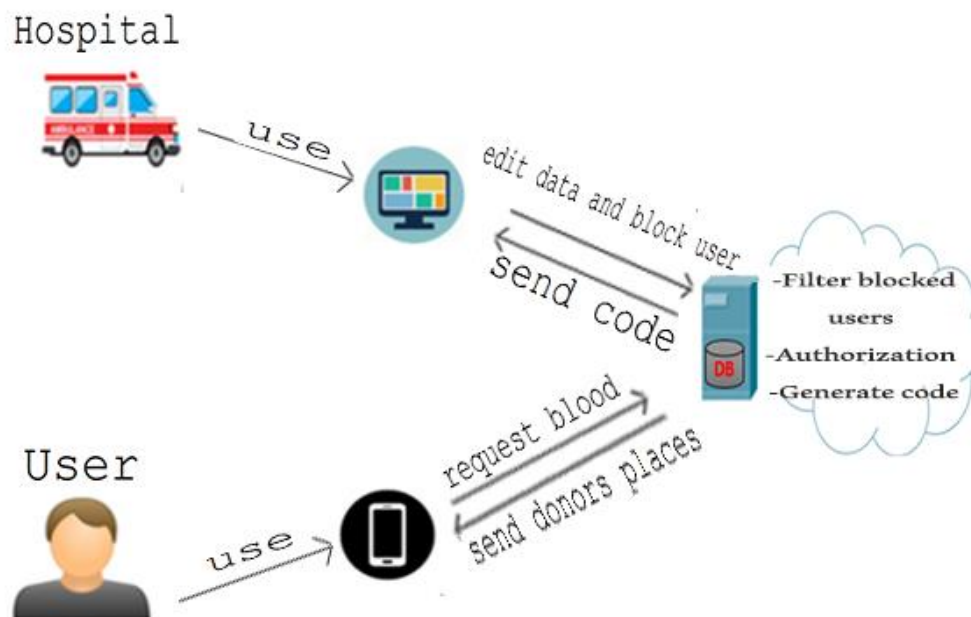


Figure 5.1

5.2 Mobile application

It's called patient share User can install it to access the services that the system provide. The main services that the systems provide blood donation. Patient request donation and system search for donors and back to patient its location

5.3 Database

Database is a big repository that consists of classes that contain data such as User's class that contains all data which belongs user (name, age, address, mail and etc...) ,donation request (id ,patient ,hospital ,donors location ,blood group) hospital information ,donation code

5.4 Web Interface

This part from system designed to hospital to modify data of donor and block donor if his health status prevent him from donation. User gets donation code from here and can download this from this portal

5.5 system screenshot

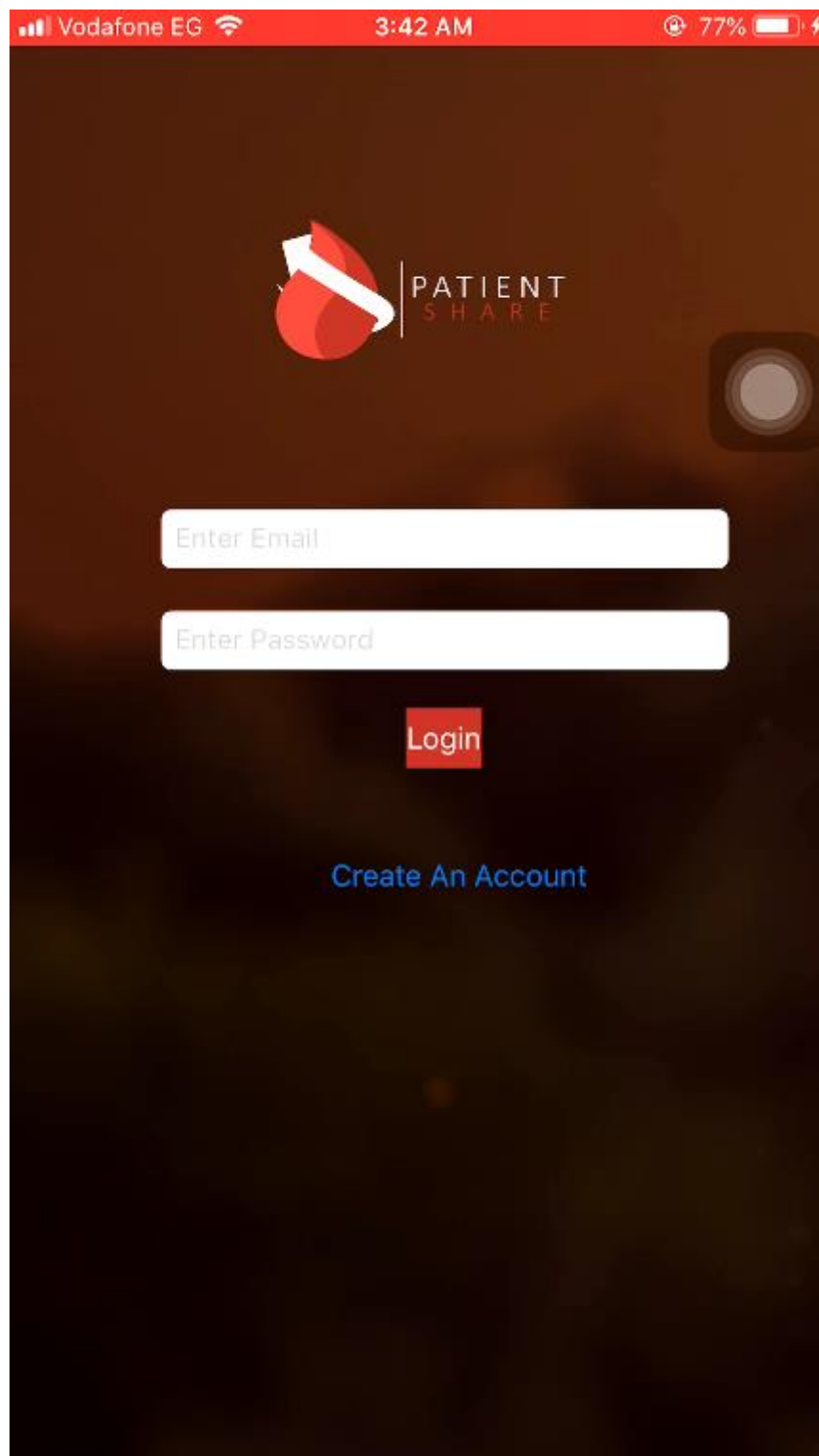
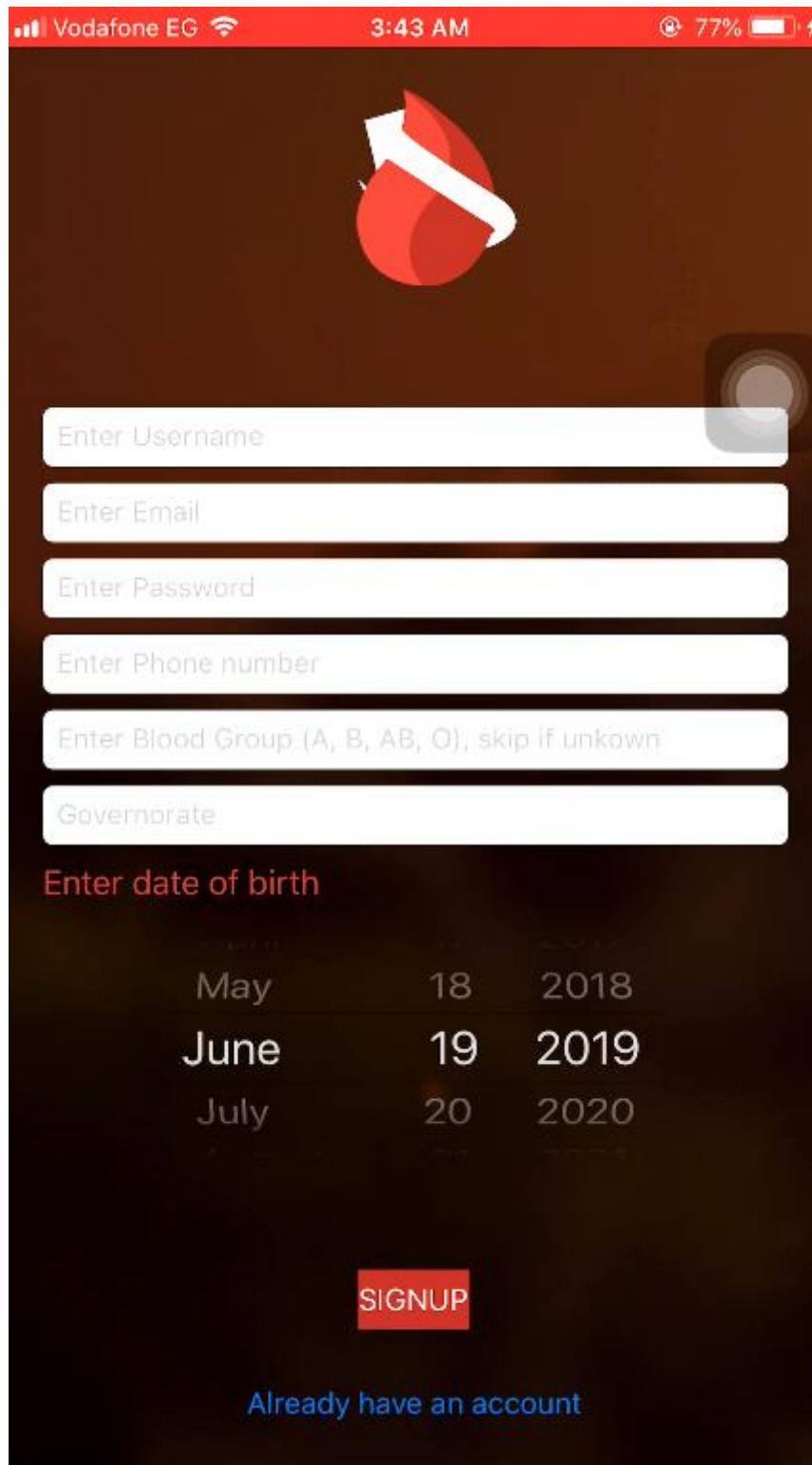


Figure 5.2
Login screen in mobile



The image shows a mobile application sign-up screen. At the top, a red status bar displays 'Vodafone EG', signal strength, Wi-Fi, the time '3:43 AM', and a 77% battery level. Below the status bar is a dark brown background with a red and white logo at the top center. A camera icon is visible on the right side. The form consists of several white input fields: 'Enter Username', 'Enter Email', 'Enter Password', 'Enter Phone number', 'Enter Blood Group (A, B, AB, O), skip if unknown', and 'Governorate'. Below these fields is a red text prompt 'Enter date of birth' followed by a date picker showing 'May 18 2018', 'June 19 2019', and 'July 20 2020'. At the bottom, there is a red 'SIGNUP' button and a blue link 'Already have an account'.

Vodafone EG 3:43 AM 77%

Enter Username

Enter Email

Enter Password

Enter Phone number

Enter Blood Group (A, B, AB, O), skip if unknown

Governorate

Enter date of birth

May 18 2018

June 19 2019

July 20 2020

SIGNUP

Already have an account

Figure 5.3

Sign up screen in mobile app

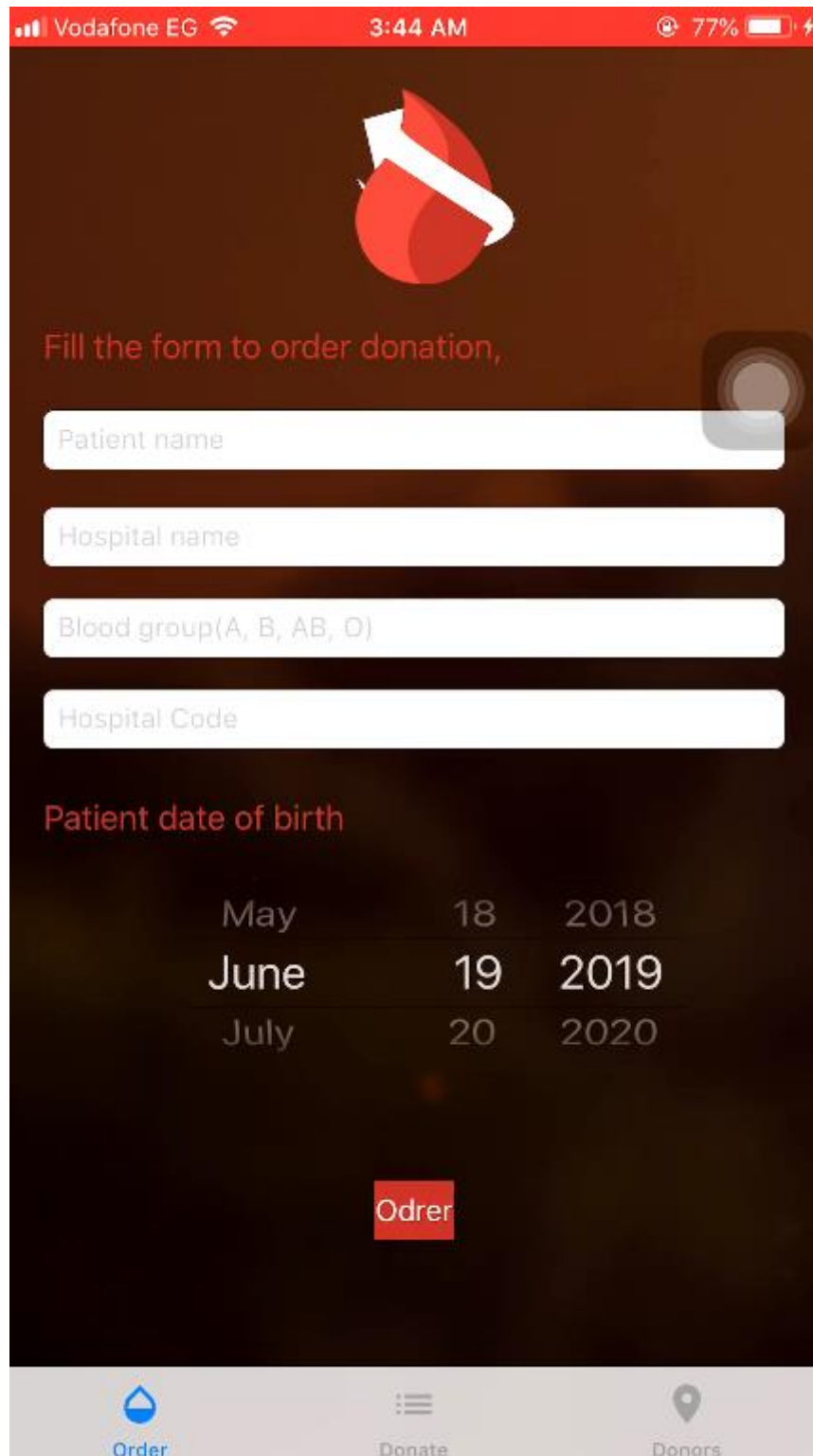


Figure 5.4
Order donation screen in mobile app

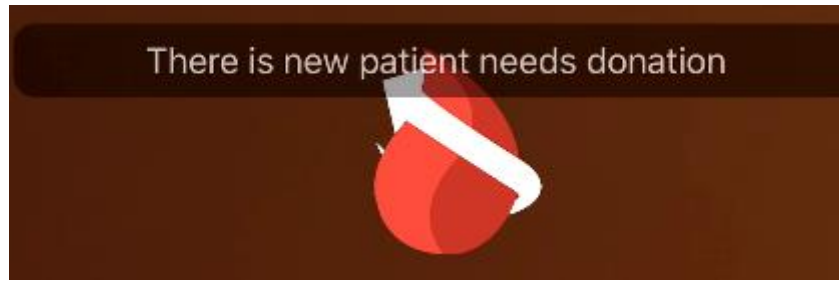


Figure 5.5
Notification screen in mobile app

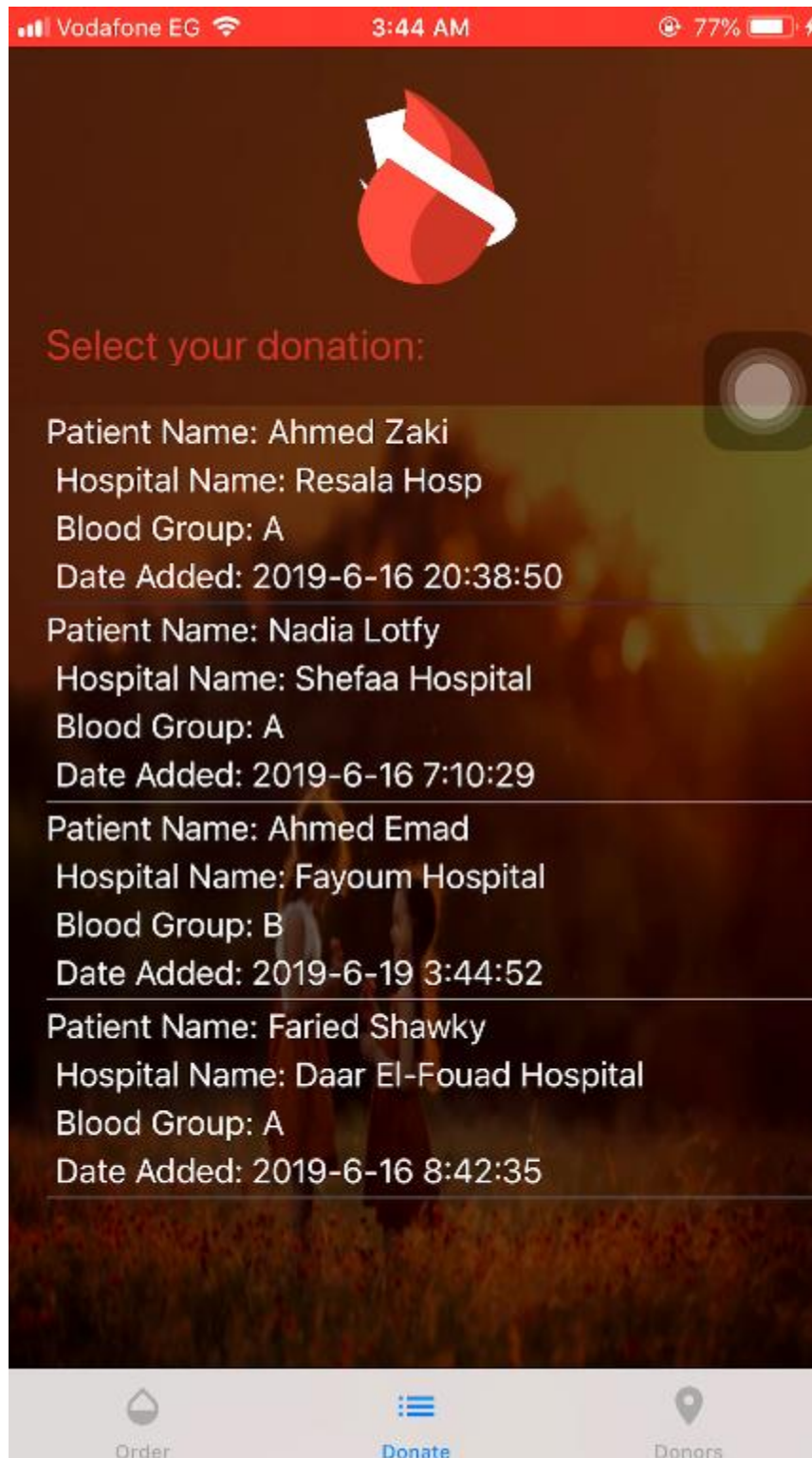


Figure 5.6

Donation requests from donor in mobile app

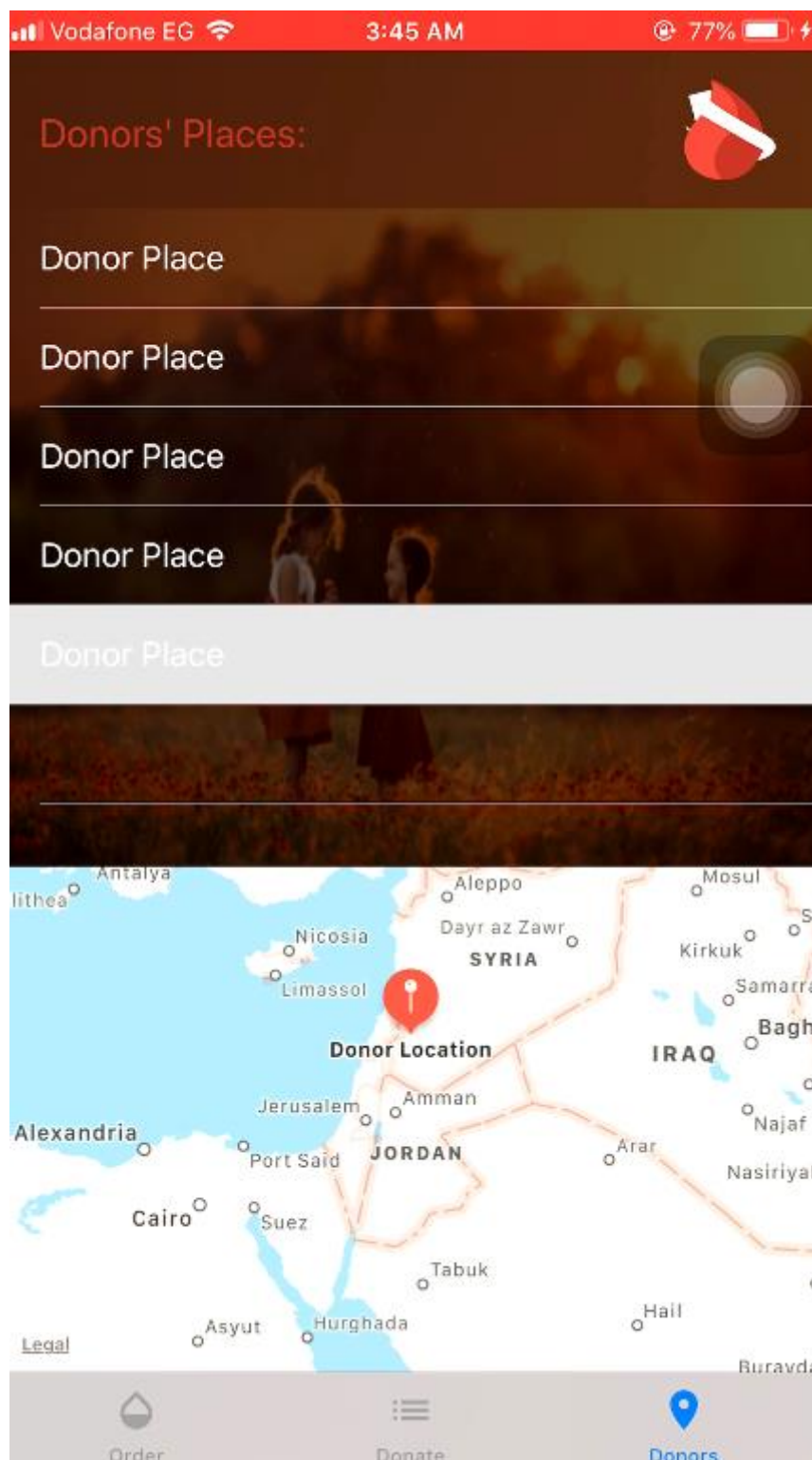


Figure 5.7

Follow view of donor in mobile app

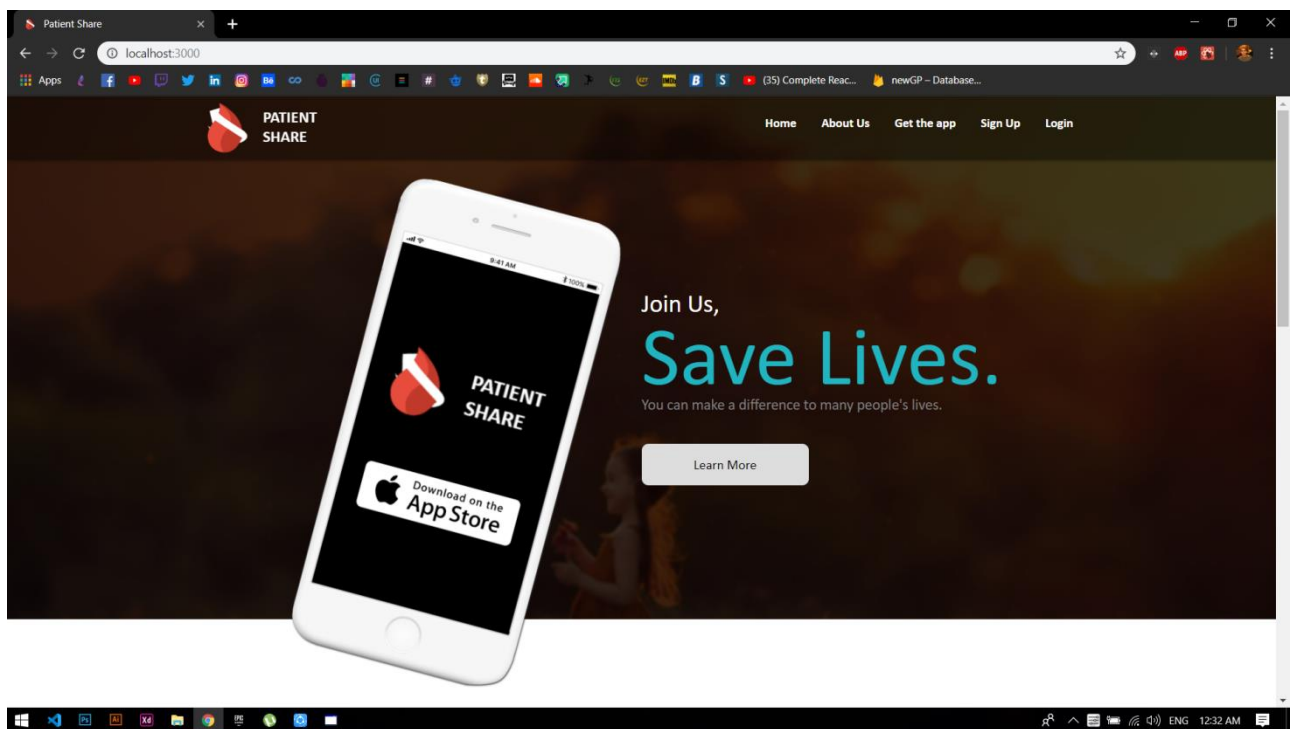


Figure 5.8
Home page of web app

Figure 5.9
Register from web app

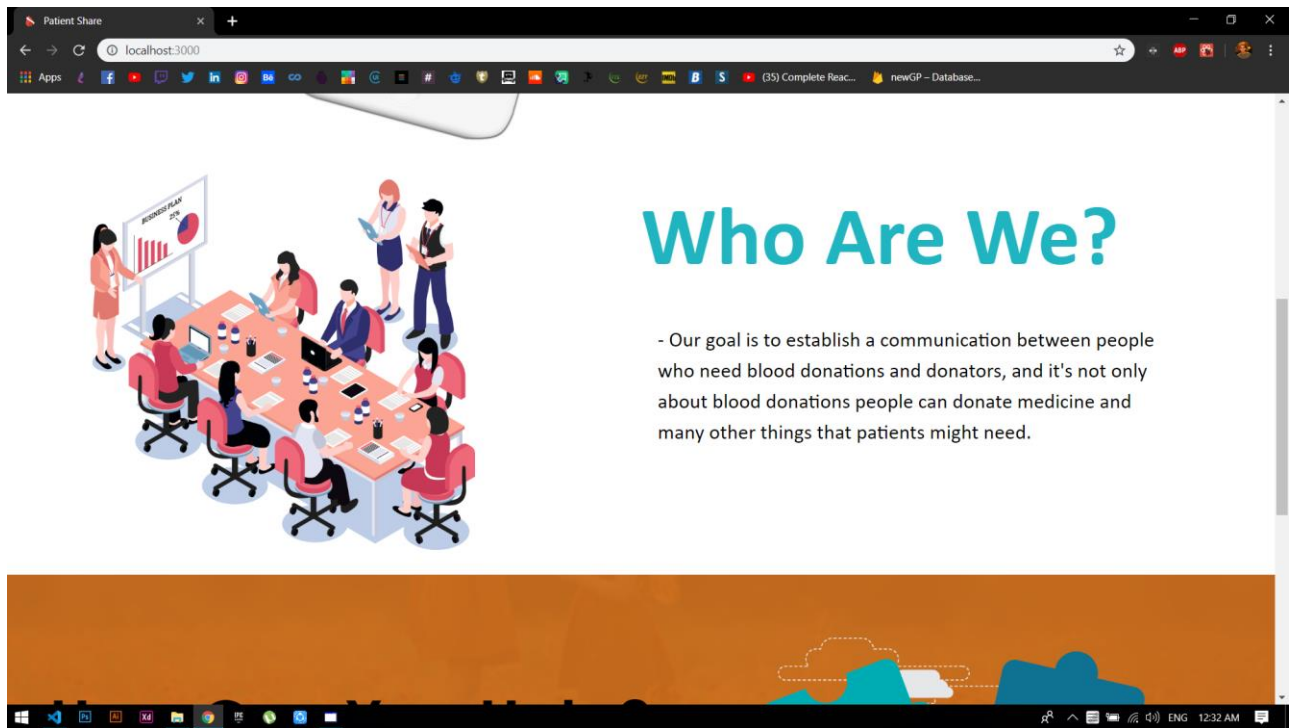


Figure 5.10

Who we are?

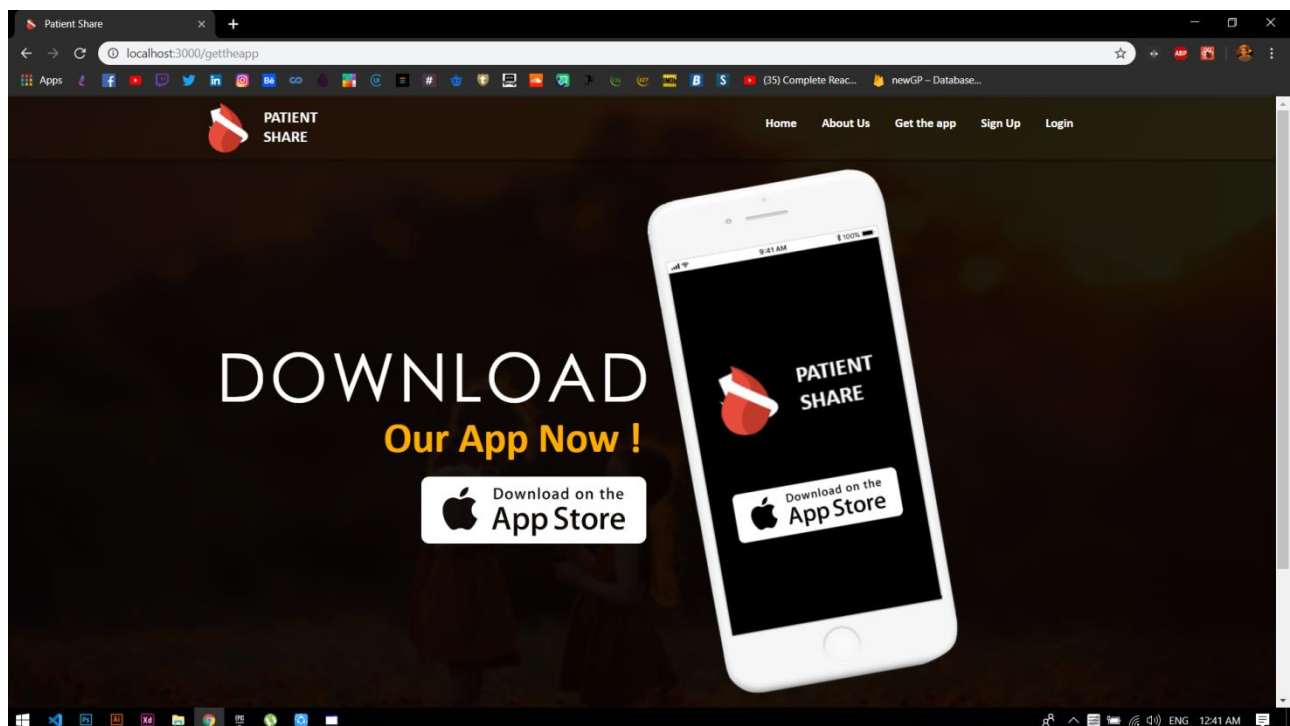


Figure 5.11

Download link from web app

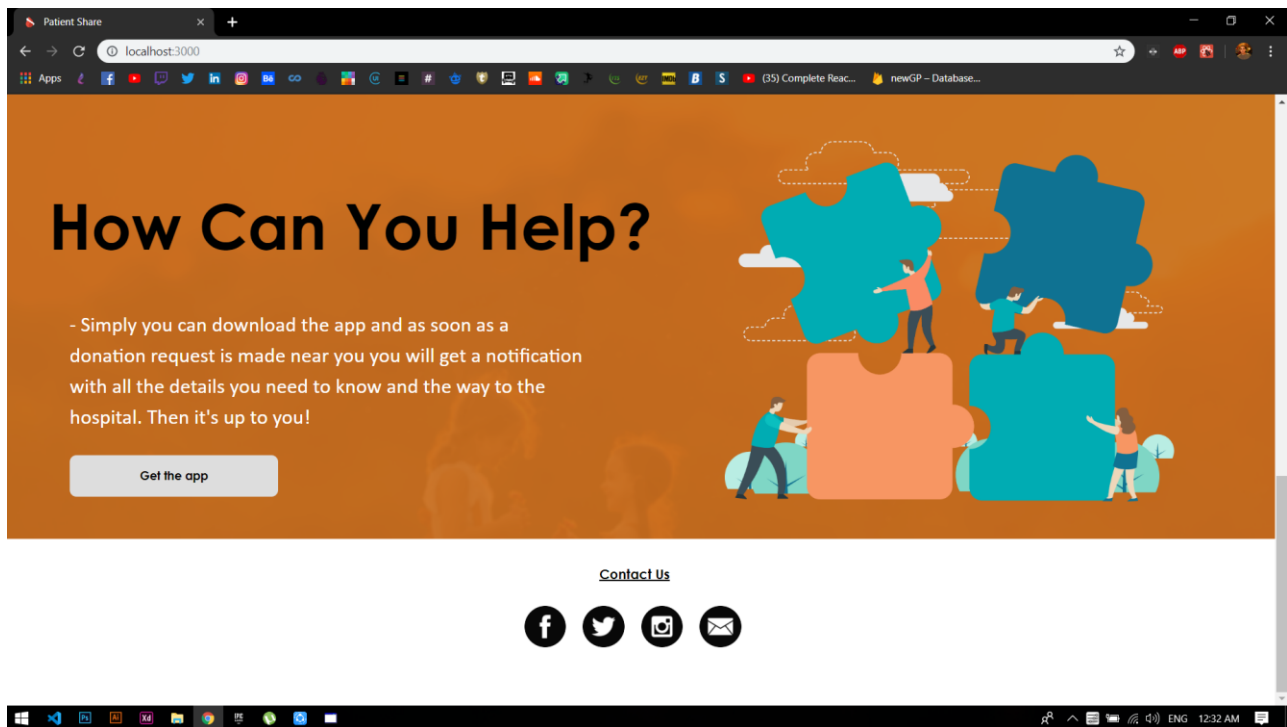


Figure 5.12

How can project help you?

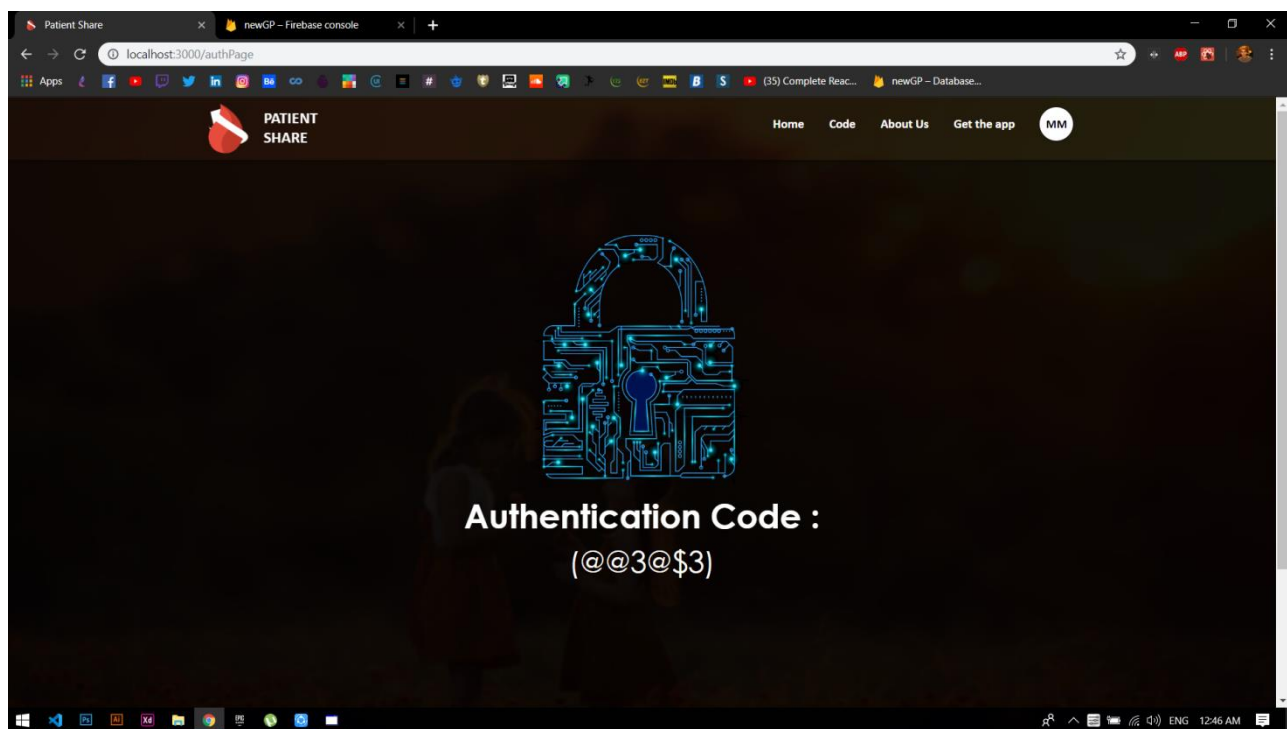


Figure 5.13

Authentication code

Chapter six

IMPLEMENTATION AND TESTING

6.1 overview

This chapter includes the implementation and testing of the system. How it was implemented? And how it was tested? It includes the testing operation of the entire system.

This chapter is organized into 3 sections:

- Section 6.1 Overview.
- Section 6.2 Testing.
- Section 6.3 code screenshots

6.2 Testing

Patient share testing required a smartphone with iOS as an operating system as it is implemented to be applicable for iOS devices, and also internet connection for accessing its services. This section discusses how the major components of the system are tested.

6.2.1 Database testing

Database testing includes that all management methods well work, data is added, updated and deleted successfully and search process retrieve the desired result. Ensuring the connection between database, the application and the web management interface is invoked.

6.2.2 Map Testing

Map testing is for ensuring that the methods that are related to the map work correctly, Map mode changes successfully, Marker appears in its right position on the map when the user chooses a donor cell and any information where is displayed within the map is done successfully.

6.2.3 Application Testing

This testing includes the test for the entire application components. Ensure that all layouts work successfully and well displayed for the user. UI is friendly use for the user and the different layouts are integrated correctly.

6.3 code screenshots

```
// ViewController.swift
// PatientShare
//
// Created by Moataz on 6/12/19.
// Copyright © 2019 Moataz. All rights reserved.
//

import UIKit
import Firebase
import FirebaseAuth
import FirebaseDatabase

class ViewController: UIViewController {

    @IBAction func loginB(_ sender: Any) {
        if let email = emailText.text, let password = passwordText.text {
            Auth.auth().signIn(withEmail: email, password: password, completion: { (user, error) in
                if user != nil {
                    //Successful Login
                    self.performSegue(withIdentifier: "segue", sender: self)
                }
                else{
                    //Failed Login
                    let alert = UIAlertController(title: "Alert", message: "Re-enter email and password",
                                                    preferredStyle: UIAlertControllerStyle.alert)
                    alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
                    self.present(alert, animated: true, completion: nil)

                    if let myError = error?.localizedDescription{
                        print(myError)}
                    else{
                        print ("Error")}
                }
            })
        }
    }

    @IBOutlet weak var emailText: UITextField!
    @IBOutlet weak var passwordText: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        self.view.backgroundColor = UIColor(patternImage: UIImage(named: "BGBack.png"))
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}
```

Figure 6.1

Login in mobile app

```

// SignupClass.swift
// PatientShare
//
// Created by Moataz on 6/12/19.
// Copyright © 2019 Moataz. All rights reserved.
//

import UIKit
import Firebase
import FirebaseDatabase
import FirebaseAuth

class SignupClass: UIViewController {

    @IBOutlet weak var govText: UITextField!
    @IBOutlet weak var dobPicker: UIDatePicker!
    @IBOutlet weak var bloodText: UITextField!
    @IBOutlet weak var phoneText: UITextField!
    @IBOutlet weak var passwordText: UITextField!
    @IBOutlet weak var emailText: UITextField!
    @IBOutlet weak var nameText: UITextField!
    var ref:DatabaseReference?
    var handle:DatabaseHandle?

    @IBAction func SignupAction(_ sender: Any) {
        Auth.auth().createUser(withEmail: self.emailText.text!, password: self.passwordText.text!, completion: {
            (user, error) in
            if self.emailText.text != "" && self.govText.text != "" && self.emailText.text != "" && self.
                passwordText.text != "" && self.nameText.text != "" {
                if self.bloodText.text == "A" || self.bloodText.text == "B" || self.bloodText.text == "AB" ||
                    self.bloodText.text == "O" || self.bloodText.text == "" {
                    if self.phoneText.text?.characters.count == 11{

                        //Successful SignUp
                        let dateFormatr = DateFormatter()
                        dateFormatr.dateFormat = "MM/dd/yyyy"
                        let strDate = dateFormatr.string(from: (self.dobPicker?.date)!)
                        self.ref?.child("myUsers").childByAutoId().setValue(["email": self.emailText.text as
                            Any,"gov": self.govText.text as Any,"name": self.nameText.text as Any,"phone": self.
                                phoneText.text as Any,"DOB": strDate as Any,"bloodGroup": self.bloodText.text as Any])

                        self.performSegue(withIdentifier: "seg1", sender: self)

                    }
                }
            } else {
                let Malert = UIAlertController(title: "Alert", message: "Please, Enter Valid mobile
                    number.", preferredStyle: UIAlertControllerStyle.alert)
                Malert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler:
                    nil))
                self.present(Malert, animated: true, completion: nil)

            }
        } else{
            let Balert = UIAlertController(title: "Alert", message: "Please, Enter Valid blood group or leave
                it empty.", preferredStyle: UIAlertControllerStyle.alert)
            Balert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
            self.present(Balert, animated: true, completion: nil)
        }
    }
    } else {
        let alert = UIAlertController(title: "Alert", message: "Please, Fill the form.", preferredStyle:
            UIAlertControllerStyle.alert)
        alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
        self.present(alert, animated: true, completion: nil)

        if let myError = error?.localizedDescription{
            print(myError)}
        else{
            print ("Error")}
    }
}

}

override func viewDidLoad() {
    super.viewDidLoad()

    // Do any additional setup after loading the view.
    ref = Database.database().reference()
    self.view.backgroundColor = UIColor(patternImage: UIImage(named: "BGBack.png")!)
    dobPicker.setValue(UIColor.white, forKeyPath: "textColor")

}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

}

```

Figure 6.2

Signup in mobile app

```

// OrderClass.swift
// PatientShare
//
// Created by Moataz on 6/13/19.
// Copyright © 2019 Moataz. All rights reserved.
//

import UIKit
import Firebase
import FirebaseDatabase
import FirebaseAuth
import FirebaseStorage

class OrderClass: UIViewController {

    var hospitalCode: String!

    @IBOutlet weak var pdobPicker: UIDatePicker!
    @IBOutlet weak var hospCodeText: UITextField!
    @IBOutlet weak var bloodText: UITextField!
    @IBOutlet weak var hospNameText: UITextField!
    @IBOutlet weak var patNameText: UITextField!
    var ref2:DatabaseReference?
    var handle2:DatabaseHandle?
    let childRef = Database.database().reference(withPath: "myOrders")

    @IBAction func orderAction(_ sender: Any) {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MM/dd/yyyy"
        let pdDate = dateFormatter.string(from: (self.pdobPicker?.date!))

        let postT : String!
        postT = getTodayString()
        let currUser = Auth.auth().currentUser?.email
        if patNameText.text != "" && hospCodeText.text != "" && hospNameText.text != "" && bloodText.text != ""{
            if hospCodeText.text == hospitalCode
            {
                self.childRef.childByAutoId().setValue(["PatientName": self.patNameText.text as Any,"HospName": self.hospNameText.text as Any,"postTime": postT as Any,"PatientDOB": pdDate as Any,"bloodGroup": self.bloodText.text as Any, "AddedBy": currUser as Any])

                showToast(message: "There is new patient needs donation")
            }
            else {
                let alert = UIAlertController(title: "Alert", message: "Please, Enter Valid hospital code", preferredStyle: UIAlertControllerStyle.alert)
                alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
                self.present(alert, animated: true, completion: nil)
            }
        }
        else {
            let alert = UIAlertController(title: "Alert", message: "Please fill the form", preferredStyle: UIAlertControllerStyle.alert)
            alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
            self.present(alert, animated: true, completion: nil)
        }
    }
}

func getTodayString() -> String{

    let date = Date()
    let calender = Calendar.current
    let components = calender.dateComponents([.year,.month,.day,.hour,.minute,.second], from: date)

    let year = components.year
    let month = components.month
    let day = components.day
    let hour = components.hour
    let minute = components.minute
    let second = components.second

    let today_string = String(year!) + "-" + String(month!) + "-" + String(day!) + " " + String(hour!) + ":" + String(minute!) + ":" + String(second!)

    return today_string
}

func showToast(message : String) {
    let toastLabel = UILabel(frame: CGRect(x: 5, y: 30, width: (self.view.frame.width-10), height: 35))
    toastLabel.numberOfLines = 0
    toastLabel.backgroundColor = UIColor.black.withAlphaComponent(0.6)
    toastLabel.textColor = UIColor.white
    toastLabel.font = UIFont(name: "IranSansMobile", size: 19)
    toastLabel.textAlignment = .center;
    toastLabel.text = message
    toastLabel.alpha = 1.0
    toastLabel.layer.cornerRadius = 10;
    toastLabel.clipsToBounds = true
    self.view.addSubview(toastLabel)
    UIView.animate(withDuration: 6.0, delay: 0.1, options: .curveEaseOut, animations: {
        toastLabel.alpha = 0.0
    }, completion: {(isCompleted) in
        toastLabel.removeFromSuperview()
    })
}

override func viewDidLoad() {
    super.viewDidLoad()

    // Do any additional setup after loading the view.
    self.view.backgroundColor = UIColor(patternImage: UIImage(named: "BGBack.png"))
    pdobPicker.setValue(UIColor.white, forKeyPath: "textColor")

    let rootRef = Database.database().reference()
    let ref = rootRef.child("code")
    ref.observe(.value, with: { snapshot in
        let value = snapshot.value as? NSDictionary
        self.hospitalCode = value?["authCode"] as? String ?? ""
    })

}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}
}

```

Figure 6.3 - order donation in mobile app

```

// DonationC.swift
// PatientShare
//
// Created by Moataz on 6/14/19.
// Copyright © 2019 Moataz. All rights reserved.
//

import UIKit
import Firebase
import FirebaseDatabase
import FirebaseAuth
import MapKit

class DonationC: UIViewController, UITableViewDelegate, UITableViewDataSource {

    var myLons:[String] = Array()
    var myLats:[String] = Array()

    let locationManager = CLLocationManager()
    var fireBaseRef: DatabaseReference!
    @IBOutlet weak var donateTV: UITableView!
    var donations: [everyDonation] = [everyDonation]()
    override func viewDidLoad() {
        super.viewDidLoad()
        fireBaseRef = Database.database().reference()
        // Do any additional setup after loading the view.
        self.view.backgroundColor = UIColor(patternImage: UIImage(named: "BGBack.png"))
        donateTV.backgroundColor = UIImageView(image: UIImage(named: "BGBack.png"))

        donateTV?.register(UITableViewCell.self, forCellReuseIdentifier: "cell")
        self.donateTV.estimatedRowHeight = 55
        self.donateTV.rowHeight = UITableViewAutomaticDimension

        donations = [everyDonation]()
        fireBaseRef.observe(.value, with: {snapshot in
            let dictRoot = snapshot.value as? [String:AnyObject]
            let dictDon = dictRoot?["myOrders"] as? [String: AnyObject]
            self.donations.removeAll()
            for key in Array(dictDon!.keys){
                self.donations.append(everyDonation(dictionary:dictDon?[key] as! [String:AnyObject], key: key))
            }
            self.donateTV?.reloadData()
        })
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
    func tableView(_ tableView: UITableView, numberOfRowsInSectionSection section: Int) -> Int{

        return donations.count
    }
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        let cell = donateTV.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as UITableViewCell
        let eachDon = donations[indexPath.row]
        cell.textLabel?.text = "Patient Name: \(eachDon.PN)\n Hospital Name: \(eachDon.HN)\n Blood Group: \(eachDon.BG)\n Date Added: \(eachDon.PT)"
        cell.textLabel?.numberOfLines = 0
        cell.backgroundColor = UIColor.clear
        cell.textLabel?.textColor = UIColor.white

        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let eachDon = donations[indexPath.row]
        let lat: Double = (locationManager.location?.coordinate.latitude)!
        let lon: Double = (locationManager.location?.coordinate.longitude)!
        let currUser = Auth.auth().currentUser?.email
        self.fireBaseRef?.child("donorsP").childByAutoId().setValue(["Latitude": lat, "Longitude": lon, "Hospital":eachDon.HN, "Donor": currUser as Any])

        let alert = UIAlertController(title: "Thank you", message: "Your donation is added", preferredStyle: UIAlertControllerStyle.alert)
        alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil))
        self.present(alert, animated: true, completion: nil)
    }

    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat{

        return 100
    }
}

```

Figure 6.4

Donate in mobile app

```

// mapV.swift
// PatientShare
//
// Created by Moataz on 6/16/19.
// Copyright © 2019 Moataz. All rights reserved.
//

import UIKit
import MapKit
import Firebase
import FirebaseDatabase
import CoreLocation

class mapV: UIViewController, UITableViewDelegate, UITableViewDataSource, CLLocationManagerDelegate {

    @IBOutlet weak var mymp: MKMapView!
    @IBOutlet weak var donorTV: UITableView!
    var handle: DatabaseHandle?
    var ref: DatabaseReference?
    var myHosps: [String] = []
    let locationManager = CLLocationManager()
    override func viewDidLoad() {
        super.viewDidLoad()
        self.view.backgroundColor = UIColor(patternImage: UIImage(named: "BGBack.png")!)

        donorTV.backgroundColor = UIImageView(image: UIImage(named: "BGBack.png"))

        mymp.delegate = self as? MKMapViewDelegate
        donorTV.register(UITableViewCell.self, forCellReuseIdentifier: "cell")
        self.donorTV.estimatedRowHeight = 22
        self.donorTV.rowHeight = UITableViewAutomaticDimension

        //FirebaseRetrieve
        donorTV.delegate = self
        donorTV.dataSource = self
        ref = Database.database().reference()
        handle = ref?.child("donorsP").observe(.childAdded, with: {(snapshot) in
            if (snapshot.value as? NSDictionary) != nil {
                self.myHosps.append("Donor Place")
                self.donorTV.reloadData()
            }
        })
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return myHosps.count
    }

    var myIndex = 0
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        myIndex = indexPath.row
        donorTV.delegate = self
        donorTV.dataSource = self
        ref = Database.database().reference()
        handle = ref?.child("donorsP").observe(.childAdded, with: {(snapshot) in
            if let value = snapshot.value as? NSDictionary {
                let longt = value["Longitude"] as? Double
                let latd = value["Latitude"] as? Double

                self.mymp.delegate = self as? MKMapViewDelegate
                let usrLocation = CLLocationCoordinate2DMake(latd!, longt!)
                // Drop a pin
                let annotationsToRemove = self.mymp.annotations.filter { $0 != self.mymp.userLocation }
                self.mymp.removeAnnotations(annotationsToRemove)

                let dropPin = MKPointAnnotation()
                dropPin.coordinate = usrLocation
                dropPin.title = "Donor Location"
                self.mymp.addAnnotation(dropPin)
                self.updateMapForCoordinate(coordinate: usrLocation)
            }
        })
    }

    func updateMapForCoordinate(coordinate: CLLocationCoordinate2D) {
        var center = coordinate;
        center.latitude -= self.mymp.region.span.latitudeDelta / 6.0;
        mymp.setCenter(center, animated: true);
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = UITableViewCell(style: .default, reuseIdentifier: "cell")
        cell.textLabel?.text = (myHosps[indexPath.row])
        cell.backgroundColor = UIColor.clear
        cell.textLabel?.textColor = UIColor.white
        return cell
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}

```

Figure 6.5

Donors places in mobile app

```

export const signIn = (creds) => {
  return (dispatch, getState, { getFirestore }) => {
    const firebase = getFirestore();
    firebase.auth().signInWithEmailAndPassword(
      creds.email,
      creds.password
    ).then(() => {
      dispatch({ type: 'SIGNIN_SUCCESS' });
    }).catch((err) => {
      dispatch({ type: 'SIGNIN_ERROR', err });
    });
  }
}

export const signOut = () => {
  return (dispatch, getState, { getFirestore }) => {
    const firebase = getFirestore();
    firebase.auth().signOut().then(() => {
      dispatch({ type: 'SIGNOUT_SUCCESS' });
    });
  }
}

export const signUp = (user) => {
  return (dispatch, state, { getFirestore, getFirestore }) => {
    const firebase = getFirestore();
    const firestore = getFirestore();

    firebase.auth().createUserWithEmailAndPassword(
      user.email,
      user.password
    ).then((resp) => {
      return firestore.collection('myUsers').doc(resp.user.uid).set({
        DOB: user.month + '/' + user.day + '/' + user.year,
        bloodGroup: user.bloodGroup,
        email: user.email,
        gov: user.gov,
        name: user.name,
        phone: user.phone
      });
    }).then(() => {
      dispatch({ type: 'SIGNUP_SUCCESS' });
    }).catch((err) => {
      dispatch({ type: 'SIGNUP_ERROR', err });
    });
  }
}

```

Figure 6.6

Backend for web application

```

const mapDispatchToProps = (dispatch) => {
  return {
    signIn: (creds) => dispatch(signIn(creds))
  }
}

const mapStateToProps = (state) => {
  return {
    authError: state.auth.authError,
    auth: state.firebase.auth
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(Login);

```

Figure 6.7

Login in web app


```

class Register extends Component {
  state = {
    email: '',
    password: '',
    day: '',
    bloodGroup: '',
    gov: '',
    name: '',
    phone: '',
    month: '',
    year: ''
  }

  handleChange = (e) => {
    this.setState({
      [e.target.id]: e.target.value
    });
  }

  handleSubmit = (e) => {
    this.props.signUp(this.state);
  }
}

const mapDispatchToProps = (dispatch) => {
  return {
    signUp: (user) => dispatch(signUp(user))
  }
}

const mapStateToProps = (state) => {
  return {
    auth: state.firebase.auth,
    authError: state.auth.authError
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(Register);

```

Figure 6.8

Register in web app

```

const mapDispatchToProps = (dispatch) => {
  return {
    signIn: (creds) => dispatch(signIn(creds))
  }
}

const mapStateToProps = (state) => {
  return {
    authError: state.auth.authError,
    auth: state.firebase.auth
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(Register);

```

Figure 6.9

Authentication in web app

Chapter seven

FUTURE WORK

7.1 Overview:

This chapter will discuss Patient share summary, system future, and the long-term vision. This chapter is organized into three sections.

This chapter is organized into 3 sections:

- Section 7.1: Overview.
- Section 7.2: Summary.
- Section 7.3: Future work.

7.2 Summary:

Patient share contribute in solving one aspect of the most important problems in our country “The health sector problem” in first stage solve blood shortage by providing those who need any type of blood with the type they need by showing the closest persons who can donate their blood within his geographical area . Patient share has a leadership and surpassed its competitors by its new features and services. It will be in a good position in online marketplace and by performing an aggressive marketing plan and some awareness campaigns; we can ramp up the project’s growth projections for the near future

7.3 future works:

-First stage:

- Develop android version from our project
- As social network becomes more powerful and has an effective role, so patient share app will not be used for guiding purposes, it will also include a “Social side”. The power of social media becomes stronger day after day so, the idea is developed to be more effective and attractive for more users.
- Making partnership with another community is one of the planned goals to increase our strength in the market place and become more popular within the market.
- Machine learning is booming in these days, there is an intent to use it for improving the algorithm to be more efficient and effective. Enhance the algorithm for better and accurate results

-Second stage:

- add features that enable user to donate and his Medicine to other users

-Third stage

- Transfer system from share blood and medicine system to more powerful share system allow user to donate and share anything related to health sector
- This system allow you to donate Medical tools, medical equipment

Money, Operations, donating organs and anything can improve medical sector in Egypt

REFERENCES

- [1] Hillegass, A & Conway, J (2010). iOS Programming: The Big Nerd Ranch Guide.
- [2] Mathias, M (2016). Swift Programming: The Big Nerd Ranch Guide.
- [3] Yahiaoui, H (2017). Firebase Cookbook: Over 70 Recipes to Help You Create Real-time Web and Mobile Applications with Firebase.
- [4] .Islam Naim, N. (2017). ReactJS: An Open Source JavaScript Library for Front-end Development.
- [5] Ali Syed, B. (2014). Beginning Node.js.
- [6] Subramaniaan, V. (2019). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node.