

ELECTRINIC KIDNEY RECIPIENTS' REGISTRY

B GNANAKEETHAN

May 2022



ELECTRINIC KIDNEY RECIPIENTS’ REGISTRY

B GNANAKEETHAN

R1812154

18121548

Name of the Supervisor:

Ms. B. Yaalini

May 2022



**This dissertation is submitted in partial fulfilment of the requirement of the
Degree of Bachelor of Information Technology (external) of the University of
Colombo School of Computing**


Declaration

I certify that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and interlibrary loans, and for the title and abstract to be made available to outside organizations.

Signature of Candidate: . . .  . . . Date: 2022/04/30

Name of Candidate: Balasubramaniam Gnanakeethan

Countersigned by:

Signature of Supervisor(s) Advisor(s): . . .  . . . Date: 2022/04/30

Name(s) of Supervisor(s)/Advisor(s): Yaalini Balathasan

Abstract

Need for a digital health solution in managing end-stage renal disease patients who are awaiting kidney transplants, especially during workup for surgery and in finding a kidney donor was emerging during the last decade in the medical professorial unit in the national hospital of Sri Lanka. It extends toward the patients who got kidney transplants and on clinic follow-up. Lack of online data of kidney recipients when a cadaveric kidney is available, lack of interoperability and good coordination in information exchange between health institutions, difficulties of record-keeping and timely updating, etc were the main emerging difficulties for the health staff to manage. Design and development of an electronic renal registry were initiated along with the need from the stakeholders to provide a better solution for all mentioned difficulties.

The system is being developed using the Iterative Spiral Model of Software Development as the system requirements may change as the system is evolved and presented to the client. It uses Golang as the backend language and PostgreSQL as the database layer. It has a web-based frontend compiled using Svelte Library. The frontend and the backend are interfaced using a GraphQL Specification.

Acknowledgments

I would like to thank my supervisor Ms. Yaalini Balathasan for her continuous guidance and support throughout the development of the project. I am forever grateful for the appreciation and support from Dr. Roshan Hewapathirana. I also would like to thank Dr. Chamila Malsiri for providing me with valuable information for the implementation of the system. It was a great opportunity to work with all of them. Thank you.

Table of Contents

Declaration	i
Abstract	ii
Acknowledgments	iii
Table of Contents	v
List of Figures	vi
List of Tables	vi
1 Introduction	1
1.1 Introduction	1
1.2 Motivation for the project	1
1.3 Objectives of the project	1
1.4 Scope	2
1.5 Functional Requirements of the System	2
1.6 Non-Functional Requirements of the System	2
2 Requirement Gathering & Analysis	3
2.1 Existing Systems	3
2.1.1 Current Manual Process	3
2.1.2 Afflo.io	4
2.1.3 OpenTransplant	4
2.1.4 Otter Organ(CareDX)	4
2.2 Fact Finding Techniques	4
2.3 Feasibility Study for the Proposed System	5
2.3.1 Operational Feasibility	5
2.3.2 Technical Feasibility	5

2.3.3	Schedule Feasibility	5
2.4	Requirements Definition	6
2.4.1	Functional Requirements	6
2.4.2	Non Functional Requirements	6
2.4.3	Resource Requirements	7
3	Design of the solution	8
3.1	Methodology	8
3.2	Alternative Solutions	10
3.3	Design Process	10
3.3.1	Interface Design	11
3.3.2	Architecture Design	12
3.3.3	Detailed Design	12
4	Implementation	19
4.1	Implementation Environment	19
4.2	Development Environment	20
4.3	Code / Module Architecture	21
5	Evaluation	22
5.1	Testing Procedure	22
6	Conclusion	25
6.1	Lessons Learnt	25
6.2	Future work	25
	Appendices	25
A	System Manual	26
B	User Manual	28

List of Figures

3.1	Spiral / Iterative Method	9
3.2	Context Diagram	11
3.3	Architecture Diagram	12
3.4	ER Diagram	13
3.5	High-level Usecase Diagram of the proposed system	14
3.6	Sequence Diagram for Patient Registration	15
3.7	Sequence Diagram for Follow Ups	16
3.8	Sequence Diagram for Workups	17
3.9	Donor Finding	18
A.1	Directory Layout	27
B.1	Login Screen	28
B.2	Dashboard Screen	29
B.3	Search Patients Screen	29
B.4	List of Patients Screen	29
B.5	Patient History Screen	30

List of Tables

3.1	Alternate Solutions	10
5.1	Authentication and Authorization Tests	23
5.2	Patient Record Access	23
5.3	Workflow Management	24

Chapter 1

Introduction

1.1 Introduction

The necessity of Digital Health Solutions is emerging in the world due to technological shifts and digital transitions. The need for a digital health solution in the Nephrology Unit of the National Hospital of Sri Lanka was identified in the Postgraduate Institute of Medicine, University of Colombo. The workups for the End-stage renal disease patients and finding a donor for them is a complicated process. The patients who have received kidney transplant needs to have regular follow-ups and investigations to be managed.

1.2 Motivation for the project

The hospital system faces numerous troubles in the current environment due to the drawbacks of finding recipients when a cadaveric kidney donor is found. There is less interoperability and good coordination between health institutions that constitute a critical blocker. The main reason is the manual process that involves managing the records and these records are not searchable in the quickest possible manner. Designing and developing an electronic system that can provide these functionalities for the Medical Officers and Doctors is essential.

1.3 Objectives of the project

- The system should allow a medical officer to manage a patient
- The system should allow a consultant to derive conclusions from the analytical reports

of the patient

- The system should allow a medical officer to find a recipient for a cadaveric donor.
- The system should provide analytical information about a patient's progress in the system.

1.4 Scope

The scope of the project is to provide a kidney patients' registry that allows Consultants and Medical Officers to manage End-Stage Renal Disease patients efficiently. The system will ease manual the process of continuity of care.

1.5 Functional Requirements of the System

- The system should allow the recording of the diagnosis history of patients.
- The system should assist the Consultants and Medical Officers in taking critical decisions about patients' health.
- The system should alert patients on their potential days of clinic meetings.
- The system should alert Medical Officers of potential missing patients from a specific clinic day.

1.6 Non-Functional Requirements of the System

- The system should store information in a secure manner not exposed to malicious actors.
- The system should prevent record tampering.
- The system should be available and reliable throughout the process.

Chapter 2

Requirement Gathering & Analysis

System analysis is an essential phase of a software development project. The process allows us to understand the complexity of the system in concern by sections. Hence it allows for a better understanding of it. The currently available mechanisms are considered, and their pitfalls and advantages are measured. This provides a better way to achieve the project goals.

2.1 Existing Systems

2.1.1 Current Manual Process

Currently, the Medical Officers and Consultants process the patients through manual processes. There are multiple forms via which data is collected and stored about the patients. Patients are issued clinic books in which the patients' diagnosis records are recorded. The patients usually must carry them with them alongside the other laboratory reports. However, there are significant issues alongside the manual process and it is highly time-consuming for the doctors and the patients. A Serial number is issued for every patient and it is recorded into the registry in the clinic room. The registrar keeps track of the demographic information about the patients. The serial number can be used to trace previous appointments, but however, only clinic books have the medical history of the patient. If the patient loses the clinic book, there are no other means to trace the history of the patient.

2.1.2 Afflo.io

An off-the-shelf solution called Afflo.io is provided as an on-premise solution for Transplant Management requirements. It has features like Donor management, Patient management comprising of facilities for self-management by patients, Electronic Health Records integrations, and Organ matching features. The system is a fully proprietary system. There is no comparable information from the informational site. [1]

2.1.3 OpenTransplant

The solution is an open-source solution developed in a foreign country with Governmental Assistance. The project has received some recognition from the online community. However, the project primarily focuses on Cadaveric Transplants whereas our system has to focus on Live Donor transplants as well. Besides that our system strives to achieve a feature parity of a similar nature.[2]

2.1.4 Otter Organ(CareDX)

This is also a commercially available solution. The solution has a full stack of solutions needed to manage organ transplants. This also has necessary features like Organ transplant tracking, research & reporting features like others. In addition to that, it has a data visualization feature. [3]

2.2 Fact Finding Techniques

The client's requirements must be gathered properly to produce an efficient output. Hence there are multiple methods we can use to develop the system. The fact-finding techniques can be critical in the analysis phase of the software development lifecycle. Even though we have multiple methods of fact-finding mechanisms, The relevant information for the development of this system was found via

- Discussions with Open Ended Questions

There were discussions with the Consultants and the medical doctors who are supposed to use the system. Such that they were asked non-specific questions relevant to their needs in a proper digitized solution. Through their answers their requirements were defined.

- Semi Structured Interviews

The prospective users were asked questions on their scope of usage of the system and how they intend to interact in the system, via such methods, their inputs were conceived into a requirements specification.

Through observation of the existing systems and analyzing the consumption patterns, it was determined that the users are relying on them in a specific pattern.

2.3 Feasibility Study for the Proposed System

The feasibility study is a very important process to find out the strength, weaknesses, threats, and opportunities of the proposed system to fulfill the main requirements. The measure of how beneficial an information system will be to an organization. A feasibility study is used for making accurate decisions. A detailed feasibility study was carried out regarding this system in the following sections.

2.3.1 Operational Feasibility

The operational feasibility of this system is highly important since the system is required to be used by less technical people who would like to achieve their goals very quickly. The operational feasibility measures the context in which the system in question accomplishes the requirement. The hospital is also eager to implement

2.3.2 Technical Feasibility

The technical feasibility analyses the availability of technical resources and technical expertise to complete the product. The product requires significant technical expertise and effort.

2.3.3 Schedule Feasibility

The schedule feasibility analyses whether the completion of the product could occur within a defined timeframe providing an advantage to the benefactor of the system. The implementation of the system will highly benefit the hospital in managing its patients effectively.

2.4 Requirements Definition

The requirement definition is an important part of a software development project. It defines the final requirement that the project/software should address and complete. A clear requirement definition is required for a project to succeed in an effective manner.

2.4.1 Functional Requirements

The functional requirements comprise the very critical requirements which are of absolute necessity for the system to function. A system always has a set of functional requirements. The functional requirements finalized for the system which is being developed are as follows.

- The Medical Officers should be able to record patient information.
- The Medical Officers should be able to monitor patient progress.
- The system should assist the Nephrology Consultants to make proper decisions.
- The system should be able to provide assistance in matching donors to Medical officers and Nephrology consultants alike.

2.4.2 Non Functional Requirements

The non-functional requirements are for non-critical parts of the software. These requirements enhance the experience of the user and provide additional features, however, these features do not affect the core functionality of the system. The following features have been identified as non-functional requirements of the system.

- Security – The system must feature security protocols to prevent misuse from malicious users/actors.
- Usability – The system must be user-friendly.
- Reliability – The system must provide a reliable & consistent interface for medical officers to carry out their duties.
- Reusability and Maintainability – The system should be able to be reused or maintained / modifiable in the future to fit further requirement changes.

2.4.3 Resource Requirements

This system is designed to be on a central server. Minimum hardware requirements to run the backend system are as follows.

- Octa-Core 2.5GHz Processor
- 8GB of RAM
- 80GB of Storage Space or above.

The system does not have any operating system dependencies. It requires PostgreSQL 14+ Database. It can be consumed by either using a mobile or desktop device via any Internet Browser.

Chapter 3

Design of the solution

System design is an important process of defining interfaces, architecture, components, databases, and models necessary for the development of the solution while satisfying system requirements. The system design converts the requirements into a technical solution. The design process is more creative and knowledge-based than system analysis.

3.1 Methodology

The design of the system can be done using various tooling. The process is very complex, and the designer should have the means to organize a vast amount of information gathered during analysis and determine if there are any discrepancies in the flow of information.

There are different development methodologies for the software development cycle. Such as the Waterfall method, Spiral /Iterative Method, Rapid Application Development, and Prototyping method. However, in this project most of the requirements are stable, but certain changes are anticipated. It was chosen to use the Spiral method of software development.

The need to choose the spiral method of software development was necessary as the system developed in question has some unexplored areas. Hence it was necessary to revisit the design parameters on each iteration and fine-tune them to match the requirements of the client. Please see Figure 3.1

In the initial iteration, requirements gathering is done. A requirements plan is then generated, and a small prototype is enacted to demonstrate the capabilities of the software system. After that, in the second iteration, the requirements are further fine-tuned based on the

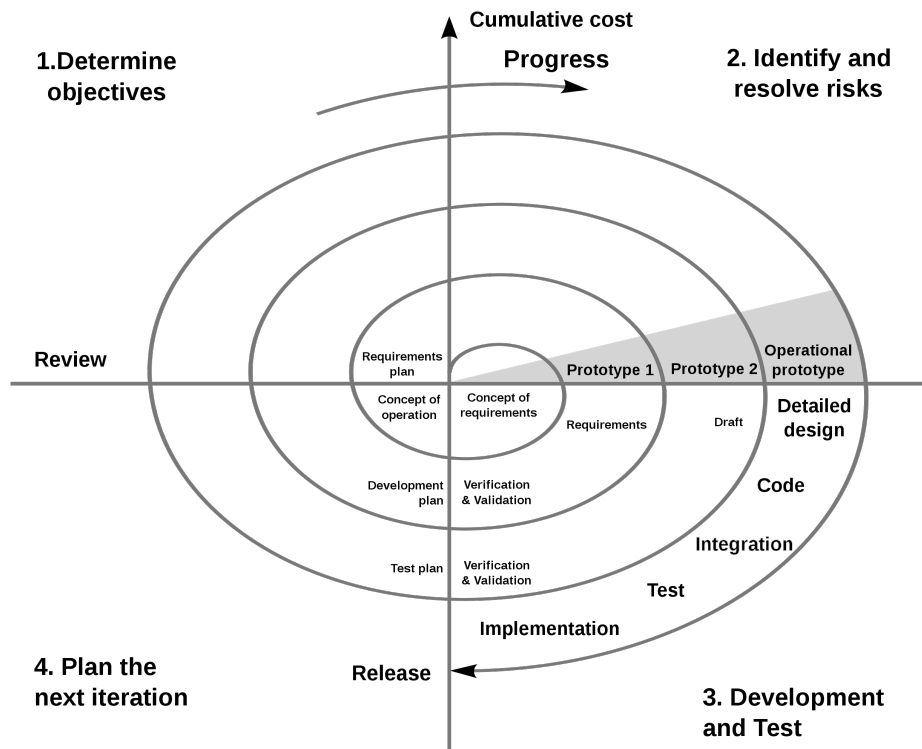


Figure 3.1: Spiral / Iterative Method

feedback from the client, and the requirements are validated and verified. The development plan is then formulated. At the end of the second iteration, a prototype is completed to have a design prototype. The development proceeds into the third iteration; once the design is validated and confirmed by the client, a test plan is enacted. During this phase, the design is converted into an operational prototype.

The operational prototype acts as the basis to which further development is done linearly—the next iteration results in the detailed design, coding, integration, test, and implementation. Then the product is released. After the release, the product is reviewed, and further changes are committed in the iterations to improve the effectiveness of the software.

The chosen development methodology for the project critically influences the design process used. Various modeling techniques can be used in system development. Few which are used in this design process are listed below.

1. Context Diagrams
2. ER Diagrams
3. Sequence Diagrams
4. Activity Diagrams

3.2 Alternative Solutions

Alternate Solutions	Mobile	Desktop	Web-based
Internet Required Always	No	No	Yes
Data Protection	Partial	Yes	Yes
Accessibility	Limited	Limited	Anywhere
Speed	Good	Best	Good
Content-Rich	Poor	Best	Good

Table 3.1: Alternate Solutions

In the above chart, it is evident that more convenient to access the content via the web interface considering the richness of the information to be shown. It should also be considered that the web interface can be also responsive and accessible via mobile for those who need it on the mobile.

3.3 Design Process

The design process results in a solution that is viable and conforms to the requirements of the clients. The following procedure has been followed to produce the final design of the software.

1. **Interface Design** In this phase, the interactions between the users and the systems are defined. This phase is done with a higher level of abstraction. The internal workings of the system are completely ignored during this phase. Attention is given to the interaction between the user and the system. A precise description of the messages that the system should produce. Specification of the ordering and timing of the incoming events or messages or outgoing messages must be detailed.
2. **Architectural Design** Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored. This covers the following topics,
 - Allocation of functional responsibilities of components

- Component Interfaces
 - Communication and interaction between components
3. Detailed Design Detailed design is the specification of the internal elements of all major components of the system and their properties, relationships, processing, and often their algorithms and the data structures.

3.3.1 Interface Design

The interface design can be done in various methods. One of the best mechanisms to show an interface design is to use a context diagram. The context diagram is a higher-level document detailing the environment and its interactions with the system. It provides an overview of the system in development.

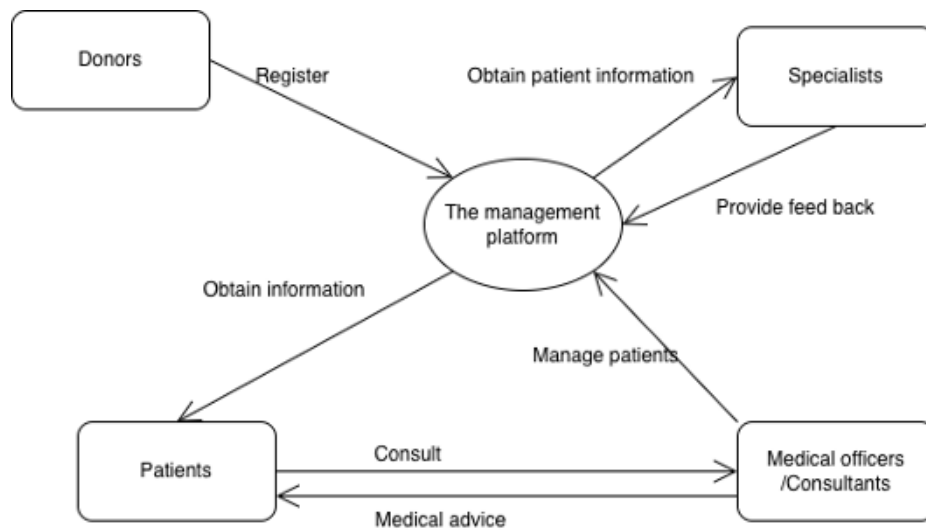


Figure 3.2: Context Diagram

3.3.2 Architecture Design

Architectural Design shows the major components of a system and depicts the inter-connectivity between each of them. The properties and interfaces are shown. The internal details specific to each component are ignored in the process.

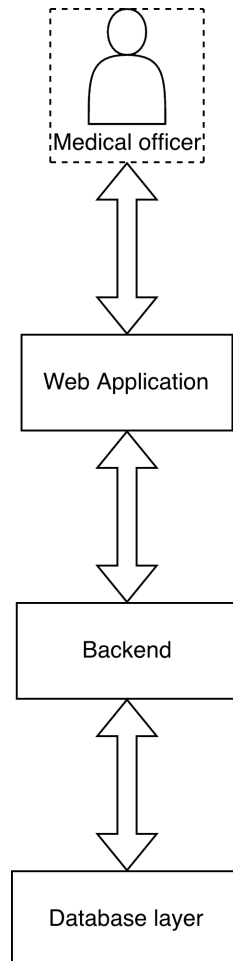


Figure 3.3: Architecture Diagram

3.3.3 Detailed Design

The detailed design of the system consists of ER Diagrams, Activity Diagrams, and Sequence diagrams primarily. Entity Relationship Diagram shows most of the internal data structures relevant to the system. It also shows the relationship of the objects situated within the system.

Entity Relationship Diagram

Figure 3.4 shows the entity-relationship diagram of the system. The system uses a sequential ID for users.

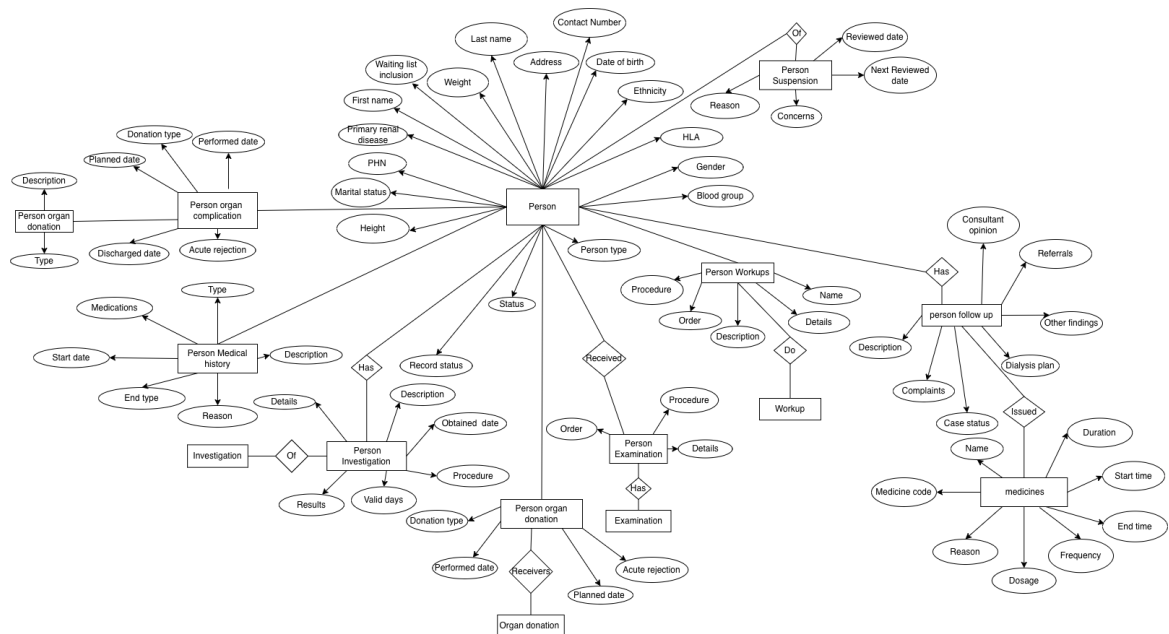


Figure 3.4: ER Diagram

Figure 3.5 depicts the interaction of the system with outside & internal components.

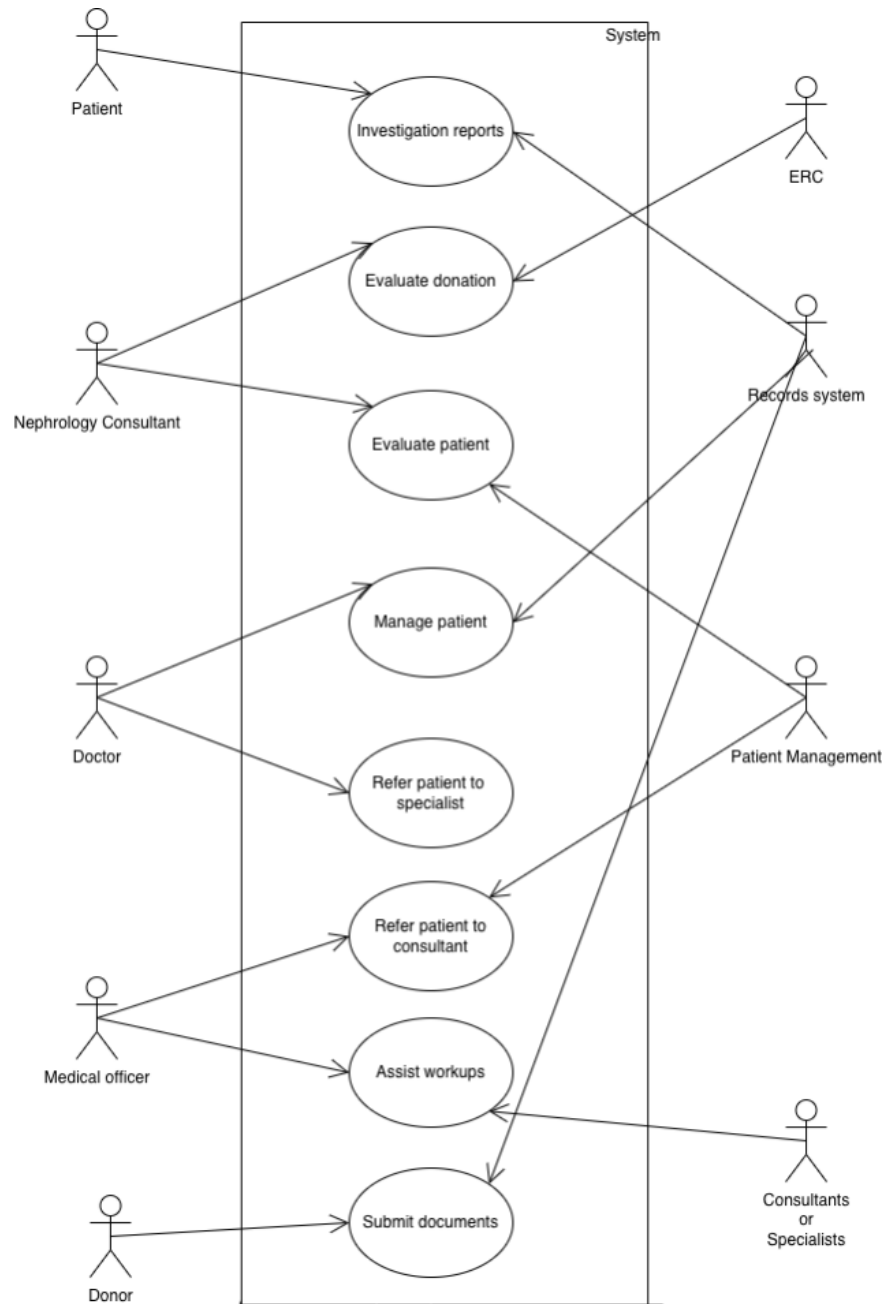


Figure 3.5: High-level Usecase Diagram of the proposed system

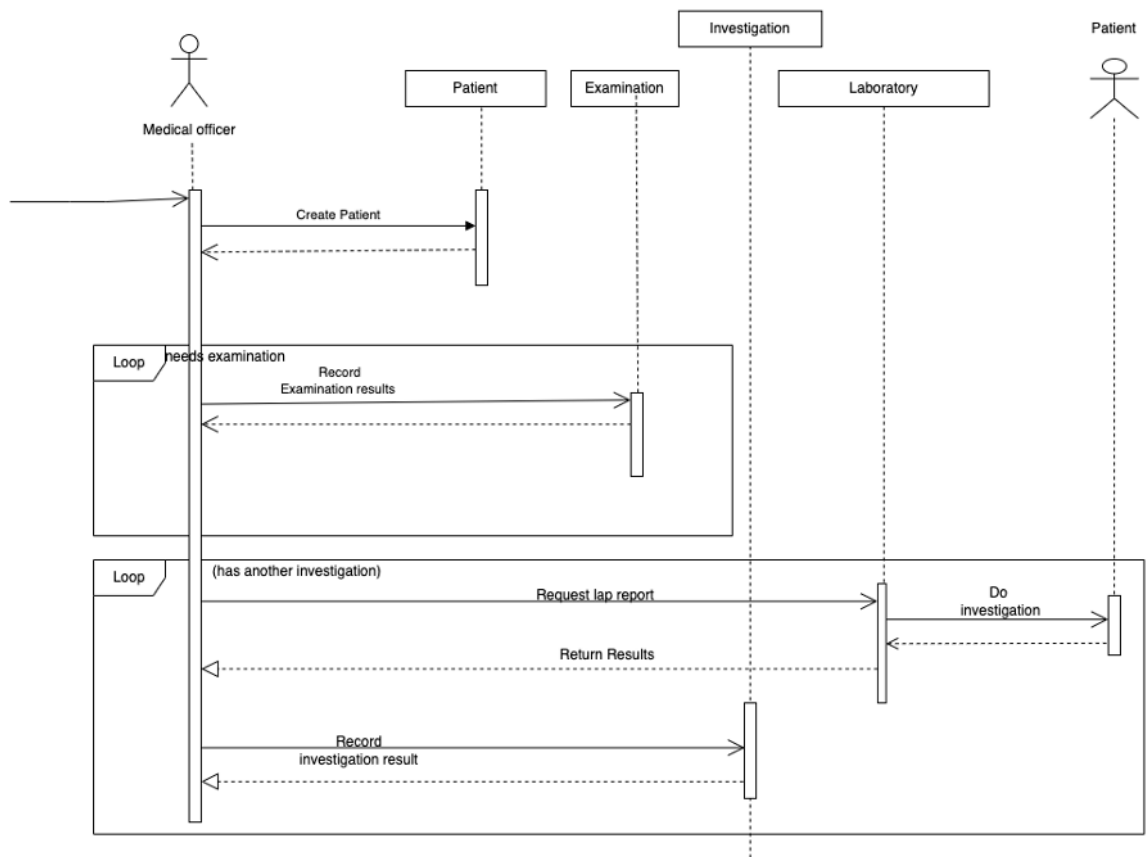


Figure 3.6: Sequence Diagram for Patient Registration

Figure 3.6 depicts the flow during which a new patient is added to the system. There are multiple steps involved in the process.

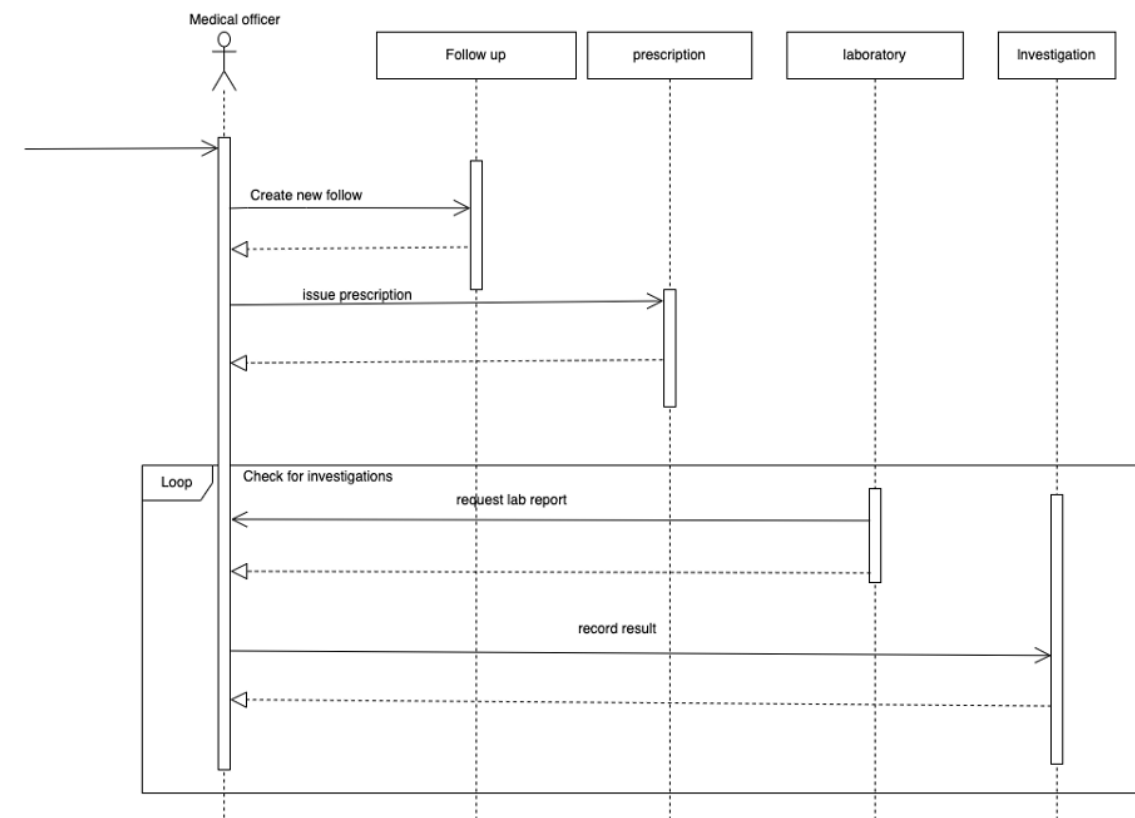


Figure 3.7: Sequence Diagram for Follow Ups

Figure 3.7 shows the follow-up procedure in the hospital. The patients are usually followed up on their regular clinic visits. The follow-up sequences are currently followed up in the clinic books but will be digitized in the new system.

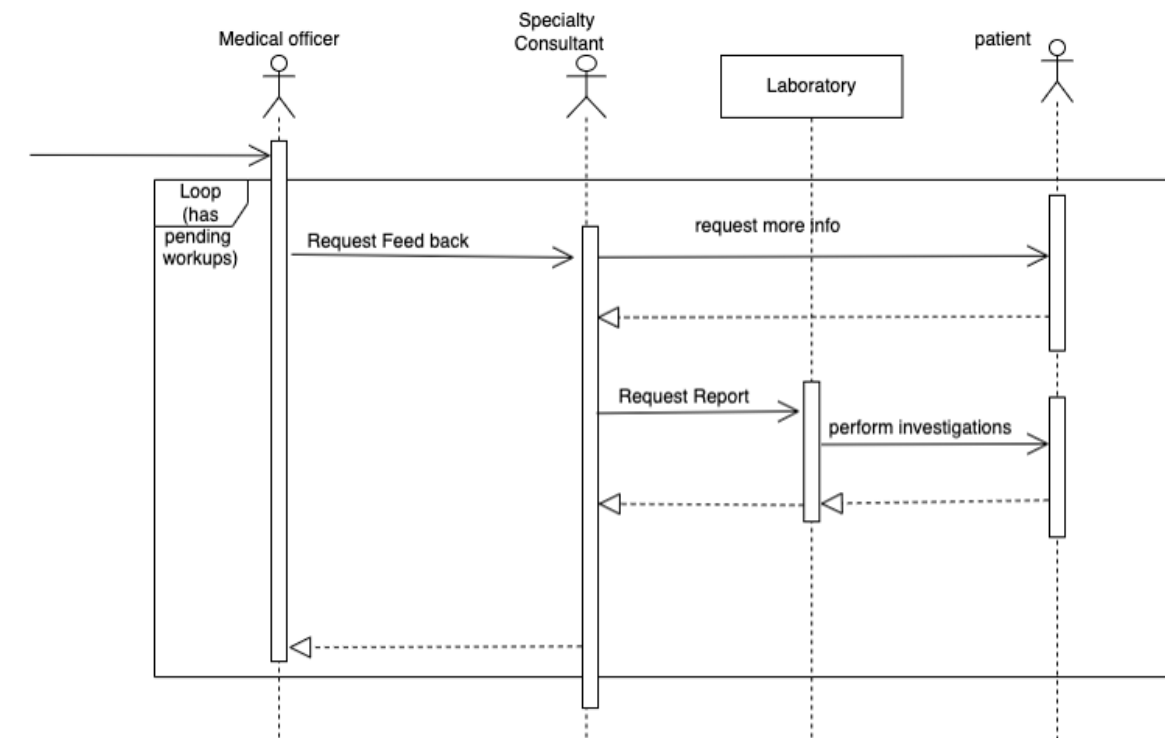


Figure 3.8: Sequence Diagram for Workups

Once the patient completes the necessary registration and passes the Kidney Transplant checklist, the patient is enrolled for workups. The workups are repeated until the patient has gone through all the Specialist doctors referred. Figure 3.8 shows the sequence diagram for workups.

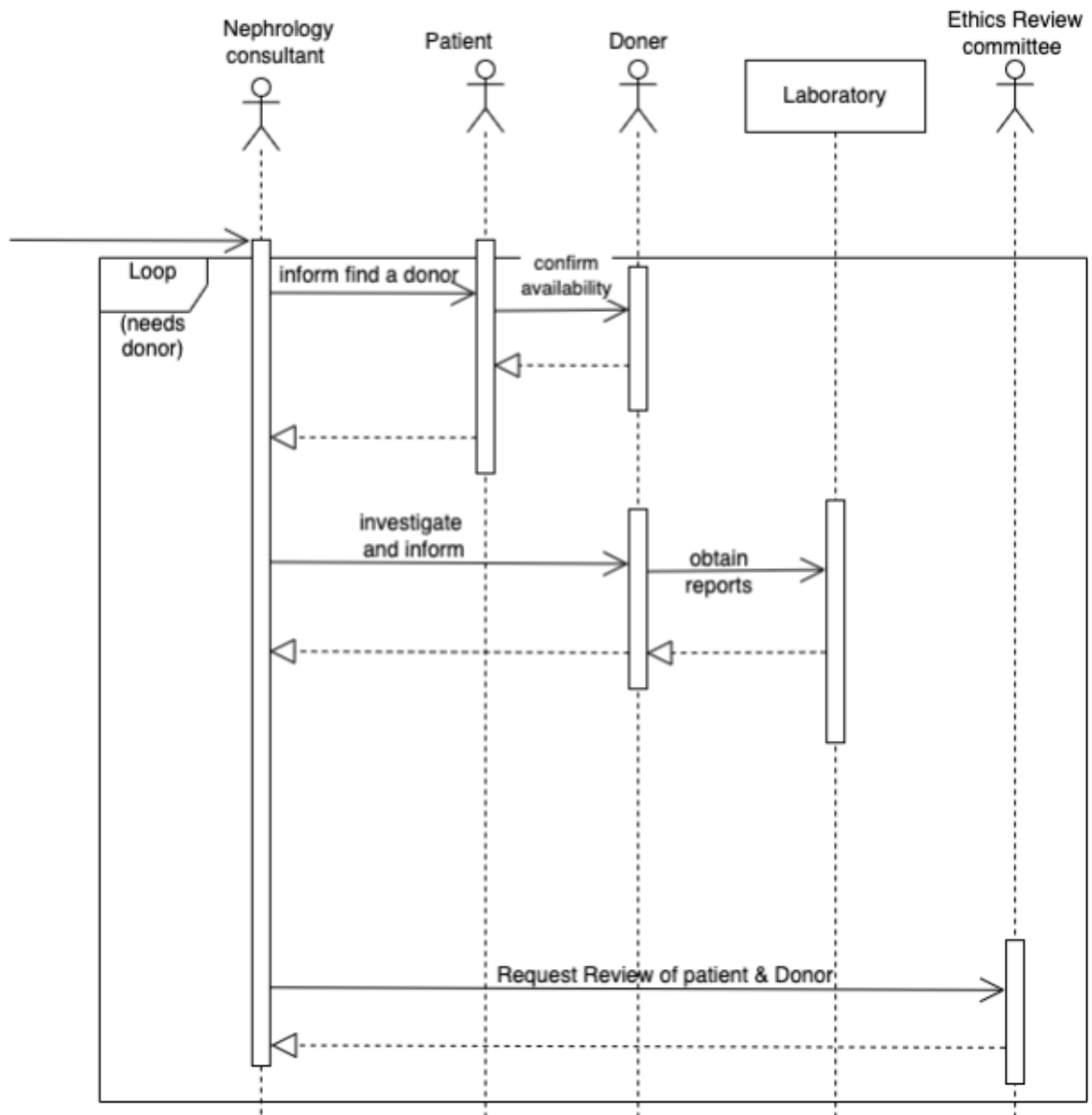


Figure 3.9: Donor Finding

The donors can be found by a Patient on their own if they need a kidney transplant. The donor needs to be fully informed of the decision and they need to go through similar steps of workups and investigations before they are allowed to go ahead with the surgery. In this manner, the donor is thoroughly reviewed. Figure 3.9 shows the procedure followed in Donor Finding.

Chapter 4

Implementation

The implementation phase is crucial in the software's lifecycle. The system is developed during this phase, according to the details gathered in the previous stages. It requires a significant amount of time to complete. The outcome of the implementation phase is deliverable. The deliverable should satisfy the requirements of the client and improve their work processes.

4.1 Implementation Environment

Software development has specific requirements for every product. Such that the implementation environment must be specified for the development. The system should be able to cater to the demand of the users as well. Following are the minimum hardware requirements for the project to successfully cater to 100 simultaneous users and store details for up to 1 year for 5000 patients.

The Application Server requires two parallel servers operating to obtain a highly available environment and to do rolling updates.

Hardware Requirements for Application Server: (2 Servers)

- 2.5GHz (4-core CPU)
- 4GB RAM
- 50GB SSD

Hardware Requirements for Database Server: (1 Server)

- 2.5GHz (4-core CPU)

- 4GB RAM
- 500GB SSD

In addition to the above requirements, in order to access the system users will require a computer running Google Chrome / Webkit browser released after 2018 for better performance gains. The system processes most of the data on the server. Hence it should work fine for most computers with an internet browser released later than 2017.

Technical Specifications

- Database Layer: PostgresDB
- Application Layer: Single binary file built from Golang

4.2 Development Environment

Hardware

- 2.5GHz-octa-core CPU (All-in-one Database, Application)
- 16GB RAM
- 10GB Storage

Software

- macOS
- JetBrains IntelliJ Idea 2022.1
- Golang 1.18
- PostgreSQL 14

4.3 Code / Module Architecture

Golang was chosen for the implementation of the backend. The latest version of 1.18 was used to develop the system since it provides some additional features like Generics. The backend is architected on top of a Golang framework called Beego. An extensible GraphQL Specification is implemented using a GraphQL Library called "GqlGen" which is available in Golang. The specification allows the system to be extensible without making many changes on the front end.

The system has been implemented with a custom Role-Based Access Control (RBAC). It allows different types of users to be assigned different permissions in the future as well. The permissions are predefined in the GraphQL Schema, which can be altered to ensure access control policies are sufficient.

The components of the system are as follows

- Users Management - Interface for managing access for the users and their permissions
- Patients Management - Interface for managing patients, includes components such as investigations, examinations, etc.
- Workflow Management - Interface for managing any customized workflow pertaining to their job role.

Chapter 5

Evaluation

The evaluation process in a software development life cycle is carried out to ensure that the system satisfies the user requirements. In this phase, the user requirements are verified to be available in the system. The system has to be tested with Sample data. This ensures that the features developed are free of issues, which is crucial for the successful delivery of a software system. Hence testing part should be done with full consideration of the features.

5.1 Testing Procedure

The system can be tested for issues in multiple manners. The testing procedures usually start from the Unit Level for each and every function and proceed up. Integration testing for each and every component ensures that the functionality is as expected when the components are working together. The system testing is carried out at the system level to ensure all of the components are functioning as intended. The system was allowed to be tested by the end-users to their satisfaction.

Unit Testing

Golang's built-in testing toolkit has been used for Unit testing.

User Acceptance Testing

These tests emulate real-world requirements and provide an environment for the software to be tested in real conditions. Hence it provides insight into how the system might perform under certain conditions. The following tests separated per module have been designed for the purpose.

Authentication and Authorization Tests

#	Description	Steps to Test	Expected Result
1	Log in without a Password	1. Visit /auth/login 2. Fill in a valid username 3. Click Login	Cannot log in to the system
2	Log in with a wrong password	1. Visit /auth/login 2. Fill in a valid username 3. Fill in an invalid password 4. Click Login	Cannot log in to the system
3	Log in with a correct password	1. Visit /auth/login 2. Fill in valid username 3. Fill in correct password 4. Click Login	Logged in successfully
4	Admin Access	1. Visit the /auth/login and valid admin credentials 2. Access Dashboard via /	Admin Dashboard Panel accessible.

Table 5.1: Authentication and Authorization Tests

Patient Record Access

#	Description	Steps to Test	Expected Result
1	Add a new Patient to the system	1. Login as a Consultant / Doctor 2. Click Patients list view 3. Click <New> on Top-bar	Add Patient view visible
2	View Patient History	1. Login as a Consultant / Doctor 2. Click Patients list view 3. Click View on a Patient 4. Open the History Tab	The recorded medical history of the patient is visible
3	Record a new consultation	1. Login as a Consultant / Doctor 2. Click Patients list view 3. Click View on a Patient 4. Open the new Follow up tab 5. Fill Details & Save	New Follow Up Form visible & Saveable
3	Record a new examination	1. Login as a Consultant / Doctor 2. Click Patients list view 3. Click View on a Patient 4. Open the new Examination tab	New Examination Form visible

Table 5.2: Patient Record Access

Workflow Management

#	Description	Steps to Test	Expected Result
1	Create new workflow	1. Log in as a Consultant/Doctor 2. Visit Dashboard 3. Click "Workflows" on Sidebar 4. Click "New" on Topbar	New Workflow Creation
2	Use Workflow	1. Log in as a Consultant/Doctor 2. Open a patient record 3. Click a workflow in the right sidebar 4. Process the workflow	Patient record presented according to the workflow and allows input to me made

Table 5.3: Workflow Management

Chapter 6

Conclusion

The project was successfully completed to achieve the scope which was determined during the design period as critical components. The client's User Acceptance Testing was conducted to ensure that client's fundamental requirements are met to use the system.

6.1 Lessons Learnt

The development of the platform resulted in great experience in developing GraphQL-based technology. It expanded knowledge and the technical know-how of project management principles.

6.2 Future work

Implementing more robust controls on the setup to record historic values of patients' health. Implementation of a mobile application for easier access.

Appendix A

System Manual

Figure A.1 shows the layout of the application.

In the root of the directory, Golang binary is built and run using the command "go build -o backend"

Within the "frontend" directory, the dependencies of the frontend has to be installed before building it. It can be installed by using "npm / pnpm.io".

















	<code>.idea</code>
	<code>conf</code>
	<code>database</code>
	<code>frontend</code>
	<code>graph</code>
	<code>models</code>
	<code>routers</code>
	<code>utilities/rand</code>
	<code>.gitignore</code>
	<code>Dockerfile</code>
	<code>docker-compose.yml</code>
	<code>go.mod</code>
	<code>go.sum</code>
	<code>gqlgen.yml</code>
	<code>main.go</code>
	<code>tools.go</code>

Figure A.1: Directory Layout

Appendix B

User Manual

Following Figure B.1 shows the login screen which the user is greeted with whenever they are accessing any part of the system.

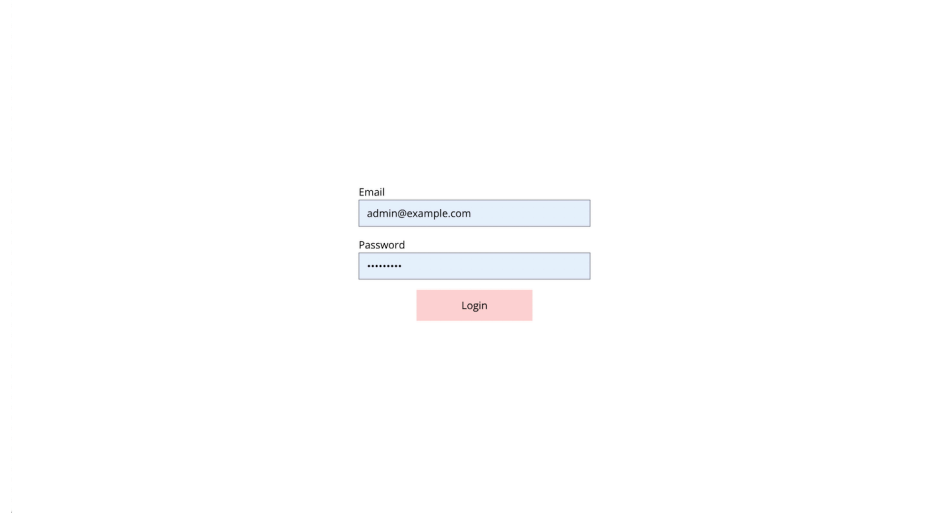
The image shows a login screen interface. It features two input fields: the first is labeled 'Email' and contains the text 'admin@example.com'; the second is labeled 'Password' and contains a series of dots. Below these fields is a red button labeled 'Login'. The entire interface is centered within a light gray rectangular frame.

Figure B.1: Login Screen

Figure B.2 shows Dashboard layout on which the user primarily lands for further actions. This layout is to be expanded with more charts in the future.

Figure B.3 shows screen that allows the user to search for patients in the system.

Figure B.4 shows the list of patients in the system in a tabular manner.

Figure B.5 shows a specific patient's historical records.

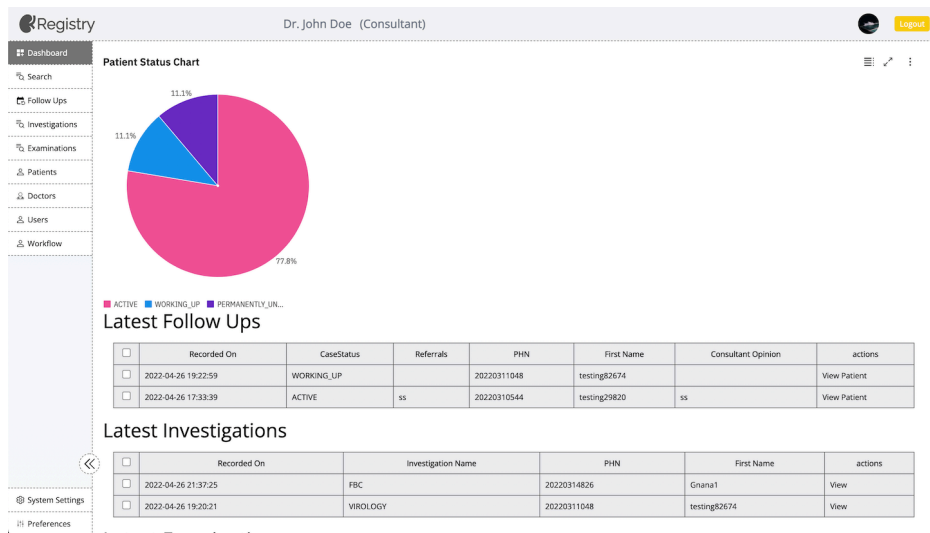


Figure B.2: Dashboard Screen

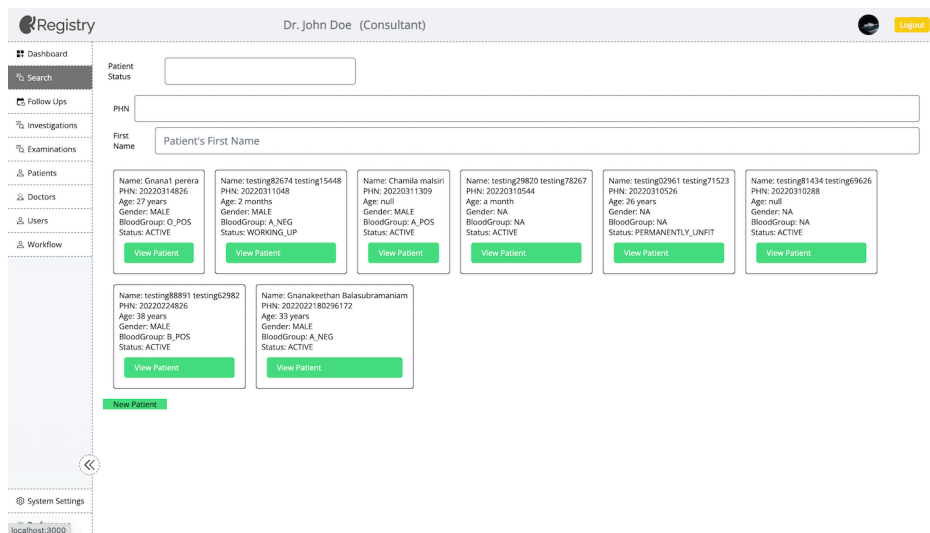


Figure B.3: Search Patients Screen

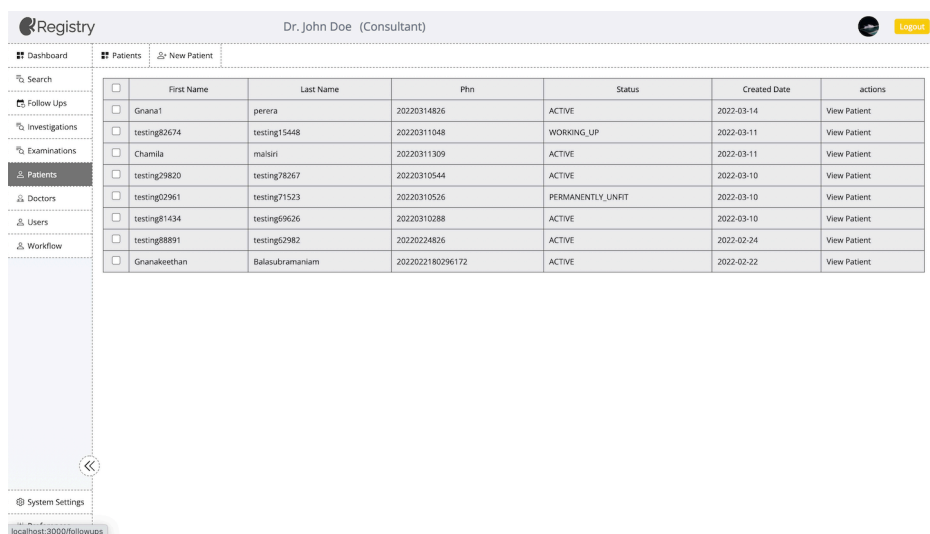


Figure B.4: List of Patients Screen

Registry

Dr. John Doe (Consultant)

Logout

Dashboard

Patients

New Patient

View Recipient

Search

Overview

History

Follow Ups

New Followup

Investigations

Examinations

New Examination

Workups

New Workup

Edit

Donor Details

Follow Ups

Investigations

Examinations

Patients

Doctors

Users

Workflow

System Settings

Preferences

RECIPIENT

Patient Name: Chamila mabisi

Date of Birth: null (null)

Phn: 20220311309

Blood Group: A_POS

Previous Records

New Complaint

Add Medical History

Add Surgical History

Add Social History

<input type="checkbox"/>	Recorded On	Occurance Date (f any)	Type	Reason	Description
<input type="checkbox"/>	2022-03-11		COMPLAINT	Fever	
<input type="checkbox"/>	2022-03-11	2022-03-10	MEDICAL	Diabetes mellitus	Defaulted
<input type="checkbox"/>	2022-03-11	2022-03-07	SURGICAL	Appendectomy	Acute appendicitis
<input type="checkbox"/>	2022-03-11	2021-11-10	SOCIAL	Smoking	Advised to stop

Available Workflows

Custom 1

Custom 2

Sample INFLAMM

SAMPLE 2

Custom Doct 1

Figure B.5: Patient History Screen

References

- [1] *Afflo.io Product Features*. <https://web.archive.org/web/20220428004926/https://afflo.io/en/features>. April 22, 2022 (accessed April 22, 2022).
- [2] <https://github.com/wiredsister/OpenTransplant/>. April 22, 2022.
- [3] <https://caredx.com/products-and-services/digital-transplant-solutions/ottr-organ/>. <https://web.archive.org/web/20210812072459/https://caredx.com/products-and-services/digital-transplant-solutions/ottr-organ/>. April 22, 2022.