IMPLEMENTATION OF PUBLIC HEALTH CONSULTATION USING FLUTTER FRAMEWORK

By

Yosua Nathanael Simbolon

001201800140

A Final Project
Submitted to the Faculty of Computing
President University
In Partial Fulfilment of the Requirements
For the Degree of Bachelor of Science
Information Technology

Cikarang, Bekasi, Indonesia

December 2021

IMPLEMENTATION OF PUBLIC HEALTH CONSULTATION USING FLUTTER FRAMEWORK

By

Yosua Nathanael Simbolon

Approved:

[illegible] [illegible]

Rila Mandala, Ph.D	Nur Hadisukmana, M.Sc
Final Project Advisor	Program Head of Information Technology

[illegible]

Rila Mandala, Ph.D
Dean of Faculty of Computing

STATEMENT OF ORIGINALITY

In my capacity as an active student at President University and as the author of the thesis/final project/business plan stated below:

Name	: Yosua Nathanael Simbolon
Student ID number	: 001201800140
Study Program	: Information Technology
Faculty	: Computing

I hereby declare that my thesis/final project/business plan entitled **" IMPLEMENTATION OF PUBLIC HEALTH CONSULTATION USING FLUTTER FRAMEWORK"** is to the best of my knowledge and belief, an original piece of work based on sound academic principles. If there is any plagiarism detected in this thesis/final project/business plan, I am willing to be personally responsible for the consequences of these acts of plagiarism and will accept the sanctions against these acts in accordance with the rules and policies of President University.

I also declare that this work, either in whole or in part, has not been submitted to another university to obtain a degree.

Cikarang, December 22, 2021

[illegible]

(Yosua Nathanael Simbolon)

SCIENTIFIC PUBLICATION APPROVAL FOR ACADEMIC INTEREST

As an academic community member of the President's University, I, the undersigned:

Name	: Yosua Nathanael Simbolon
Student ID Number	: 001201800140
Study Program	: Information Technology

for the purpose of development of science and technology, certify, and approve to give President University a non-exclusive royalty-free right upon my final report with the title:

IMPLEMENTATION OF PUBLIC HEALTH CONSULTATION USING FLUTTER FRAMEWORK

With this non-exclusive royalty-free right, President University is entitled to converse, to convert, to manage in a database, to maintain, and to publish my final report. There are to be done with the obligation from President University to mention my name as the copyright owner of my final report.

This statement I made in truth.

Cikarang, December 22, 2021

[illegible]

(Yosua Nathanael Simbolon)

ADVISOR APPROVAL FOR JOURNAL/INSTITUTION'S REPOSITORY

As an academic community member of the President's University, I, the undersigned:

Name	: Rila Mandala, Ph.D
ID Number	: 20020200021
Study Program	: Information Technology
Faculty	: Computing

declare that following final project:

Title of Final Project	: Implementation of Public Health Consultation Using Flutter Framework
Final Project Author	: Yosua Nathanael Simbolon
Student ID Number	: 001201800140

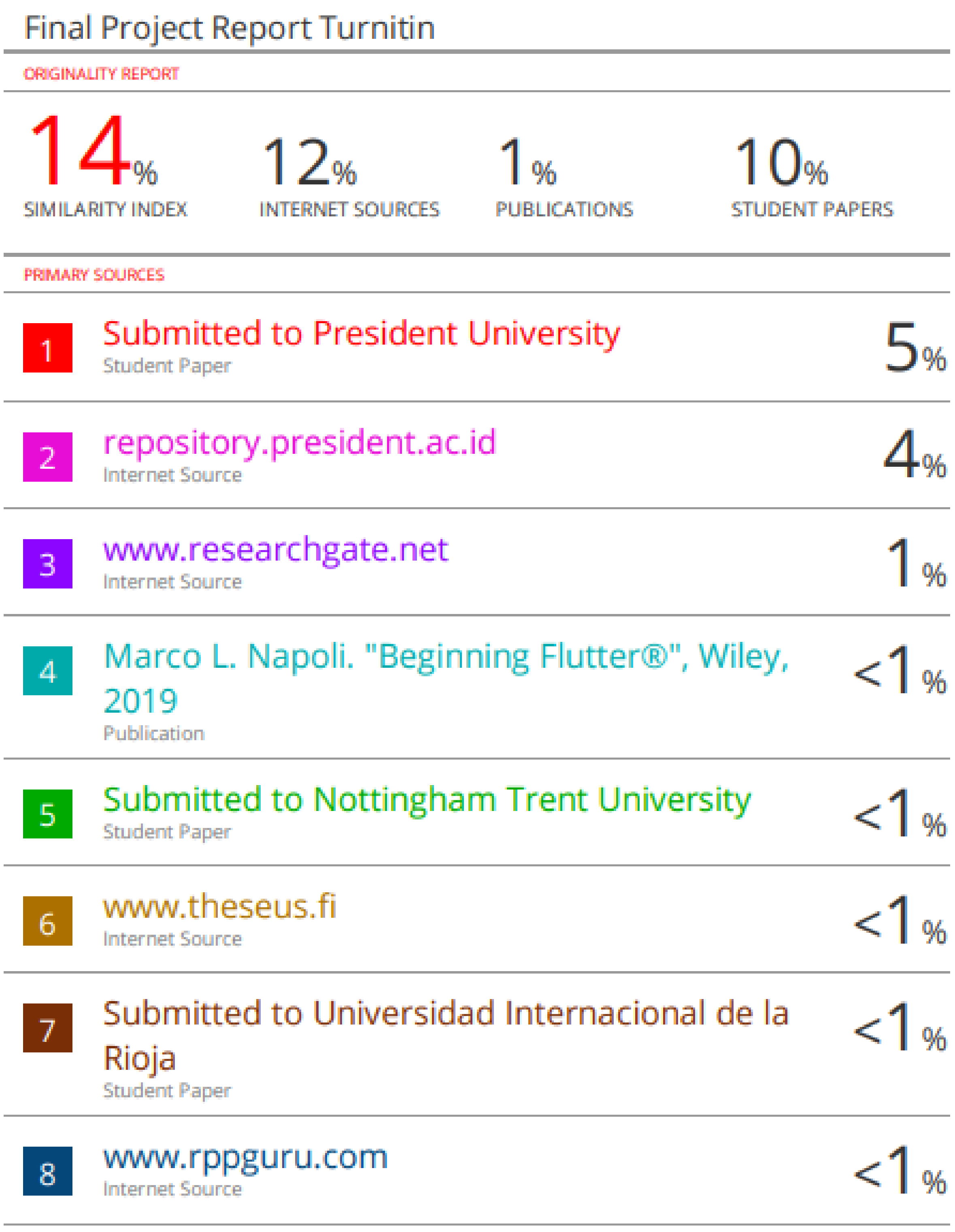
will be published in **journal / [✓ institution's repository] / proceeding / unpublish.**

Cikarang, December 17, 2021

[illegible]

(Rila Mandala, Ph.D)

SIMILARITY INDEX REPORT



Final Project Report Turnitin

ORIGINALITY REPORT

14%	12%	1%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to President University Student Paper	5%
2	repository.president.ac.id Internet Source	4%
3	www.researchgate.net Internet Source	1%
4	Marco L. Napoli. "Beginning Flutter®", Wiley, 2019 Publication	<1%
5	Submitted to Nottingham Trent University Student Paper	<1%
6	www.theseus.fi Internet Source	<1%
7	Submitted to Universidad Internacional de la Rioja Student Paper	<1%
8	www.rppguru.com Internet Source	<1%

ABSTRACT

The longer the rate of human growth in Indonesia is increasing, this situation makes the need for health increases for example the medicine and medical personnel. Many people are unable to go to the nearest hospital or medical clinic due to far distances, take a lot of time to come, and cost issues. The evolution of technology allows people to do everything instantly and faster, this is a very big opportunity to make something useful, especially in the human health sector. This final project will make a mobile application to make people easier to get a medical solution to solve their problems and absolutely cheaper than if people go to the hospital. This mobile application is database-based and made with Flutter framework. This application will divide into 2 users, the first user is people who need medical checks and the second user is the doctor. The user page will have a single button, the button will log in include registration using a google account, so the user did not register again, just using one google account, when login the user can search doctor and find the category of the doctor and directly doing health consultation. The doctor page will have the same application as the user but in the database, the doctor's role will be set by the user who want to be a doctor, by uploading their certificate and choosing their role in the application.

Keyword: technology, medical consultation, user, doctor

DEDICATION

I as the author of this final project, I would dedicate this final project to the Jesus Christ because of Him bless, so I can finish this final Project. I also would dedicate my final project to my parents, my cute girlfriend, and all of my friends. I hope this final project can be used for educational purpose in the academic field.

ACKNOWLEDGEMENTS

I as the author of this final project would like to say all of my gratitude to the Jesus Christ, because of Him bless and giving me healthiness. In this section, the author is very grateful to everyone who has provided assistance, guidance, encouragement, and support in making this final project. Therefore, the author would like to express my gratitude to:

1. Mr. Rila Mandala, Ph.D., as my final project advisor who always guide me and advise me when I working this final project, and give me a lot of knowledge while I am studying at university.
2. Mr. Rila Mandala, Ph.D., as the Dean of Faculty of Computing, Mr. Nur Hadisukmana, M.Sc., as the Program Head of Master of Science in Information Technology, Mrs. Inda as Secretary of Faculty of Computing, and other President University lecturers who provide facility to finish this final project.
3. My beloved parents (Mr. Parlindungan Simbolon and Ms. Tarida Siregar), my brother (Haikal Simbolon), and my sister (Meisya Simbolon), for all support and pray.
4. My beloved cute girlfriend, for all support and gofood.
5. My beloved *sementara* members (Rizky Asyam, Christopher Yusuf William Hia, M. Raihan, William T, Fuad M, Salman, Sultan, Naufal, and Bernardo), for all support
6. My beloved PROFEXENT, for all support and pray.
7. Lastly, all friends which I cannot mention one by one. I express my deepest gratitude.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1. Background

Health is one word but important, health is the most important part of every living thing in this world especially for humans. Humans can only do something if they are in good health and then health can be an indicator of humans can work efficiently or not. [1] Health is the main actor in this world because if health is disturbed so another part such as political, economic, or trade market will be disturbed too.

The health need is increasing day by day, many ways to keep humans healthy such as eating good food, having a good habit, or eating vitamins. And also, humans will pay more if they can keep their healthiness because of that reason many companies take this opportunity to make their business about health.

The most common if a person to take care of his health is that they have to know about his body and how one can find out about it, they have to consult and pay a doctor for a health check so that they will know about their body, they can improve their health and can more productive than anyone who doesn't check their body. The method is widely used by health companies to profit from people who are willing to pay more for health. This method causes many inequality problems in social equality because the only people who can get this method are people who have money, for those who can't pay the health consultation fees they can't get this method.

Technology is a wonderful thing because technology can make something impossible to possible. Technology is a tool that can help people from many aspects, for example in the health sector because we can make something using technology and connect many people to discuss about healthiness and talk about the health trend. With technology, we can help people to make an easy way to check their healthiness so everyone can increase their health and can more productive. Not only improve health but check-up our health can prevent us to more serious diseases.

This final project will be one of the factors that can make everyone able to carry out health consultations. The factor mentioned is making an application to conduct online consultations with doctors in general. By utilizing this application, people will easily get health consultations without having to be charged. By doing this online consultation using the application we use, we can improve our health, we can also prevent or find out our disease before it becomes more severe and difficult to treat.

Based on the background above, the authors try to overcome the above problems by creating applications that are suitable for their use and can be used by everyone, by making the application display simple and lightweight so that all generations can use the application easily and can be of benefit to the community. everyone. Therefore, this final project was submitted under the name "**Implementation of Public Health Consultation Using Flutter Framework**"

1.2. Problem Statement

Based on the issues and problems described in the background section, the researcher identifies research problems as follows:

- Analysing the benefits of using the Health Consultation method to doctors for people who can carry out the method compared to those who cannot carry out the method.
- The lack of an application that allows everyone to carry out free health consultations so that everyone without exception can feel and know their health condition.

1.3. Final Project Objectives

This study aims to solve the problem by creating an application that has benefits in the health sector such as online health consultation using a mobile application. Creating a mobile application can provide a more practical effect and still have an effective quality to solve this problem.

1.4. Scope and Limitation

This research focuses on developing a mobile application that can show online health consultations. So, the research will cover up as follows:

1. The user will receive data from the doctor after carrying out an online consultation using the mobile application.
2. The data will be inputted using the chat system directly to the doctor.
3. The data will be analysing by the doctor manually.

However, the limitations of this research are:

1. Data processing is a little time consuming because the data will be processed manually directly from the doctor.
2. The data provided to the user will be following the analysis of the doctor who handles that user.

1.5. Methodology

This application uses the Rapid Application Development methodology (RAD). RAD is one of the software development tools introduced by James Martin in 1991. This methodology is designed to reduce the emphasis and design and put more emphasis on the adaptive process.

In RAD there are four sections in the process of development:

1. Requirement Planning Section

   This section begins with planning what will be done and also planning what equipment will be used. This section also prepares project scope, system requirements, and constraints
2. User Design Section

   After finishing planning this project, it's time to start designing applications that will be made such as user interface, input, output design, and database design.
3. Development Section

   In this section where all designs and plans are developed, the development process is carried out according to what has been planned and planned
4. Cut Over Section

   In this last section, the finished application will carry out some testing to find out if there are bugs or errors and will also fix the existing errors.

1.6. Final Project Outline

This research contains seven chapters as listed below:

1. **CHAPTER I: INTRODUCTION**

   This chapter contains background, problem statement, research objectives, scope and limitation, research methodology, and research outline.

2. **CHAPTER II : LITERATURE STUDY**

   Literature study explain about the theories and concept related this final project application. This chapter consists description of what is mobile application, flutter, Firebase, ERD, UML, Black Box Testing, Health Consultation, Review of Relevant Studies.

3. **CHAPTER III: SYSTEM ANALYSIS**

   This chapter contains the description all of the requirements, analysis and function of the health consultation application using use case diagram and swim lane diagram.

4. **CHAPTER IV: SYSTEM DESIGN**

   This chapter explains how the health consultation is implemented on mobile application. This chapter consist of the user interface development and the mobile application details.

5. **CHAPTER V : SYSTEM IMPLEMENTATION**

   In this chapter will explain about the application page and layout, the application component, and application architecture. In this section will contain about User Interface and Database Design.

6. **CHAPTER VI: SYSTEM TESTING**

   This chapter will explain about the testing of the application’s system and function to running the application and doing the service as mobile application health application. This chapter include Testing Environment, Testing Scenario, and Limitation Testing.

7. **CHAPTER VII: CONCLUSION AND FUTURE WORK**

In this final chapter contains conclusion about this final project. The future work of this chapter describes about the next step to improve for the health consultation application in the future.

CHAPTER II

LITERATURE STUDY

The final project's related concept and theories that are used in the development of Health Consultation Mobile Application.

2.1. Theoretical Framework

2.1.1. Mobile Application

[2] A mobile application is a technology that consists of software that can perform many commands for users. Mobile application is a technology that is developing very quickly in line with the number of enthusiasts and also users of mobile applications. The factors that make mobile applications very attractive are practical, user-friendly, inexpensive, an application can be used on all mobile phones that support the technology. Mobile apps have many very useful uses such as calling, sending messages, surfing the internet, communicating, social media, audio, video, games, etc.

Most of the mobile applications are pre-installed before the mobile is used and some users can download the application from the internet or the mobile market application. The application market is also a place for developers, publishers, and also mobile application providers to gather and deliver their innovations. From a technical point of view, mobile applications can be run on various leading platforms such as iPhone and Android.

The usefulness of a mobile application is that it can connect one person to another in a very practical, inexpensive, and also easy way so that many ideas for developers can be much more utilized for everyday life, not just communication but can also influence factors. other.

2.1.2. Flutter

Flutter is a cross-platform framework that aims to develop high-performance mobile devices, why is it considered cross-platform because applications derived from the flutter framework can be used on Android as well as IOS. Flutter was released by Google Corporation in 2016, besides being able to be used on Android as well as IOS flutter can rely on OEM device widgets, Flutter renders every component of the high-performance rendering engine itself. With these properties, it is very supportive if flutter can produce applications that have high performance. Flutter also uses a C/C++ engine code structure compiled with NDK from Android as well as LLVM from IOS, and code from Dart Language is AOT compiled to native code during compilation and render the code.

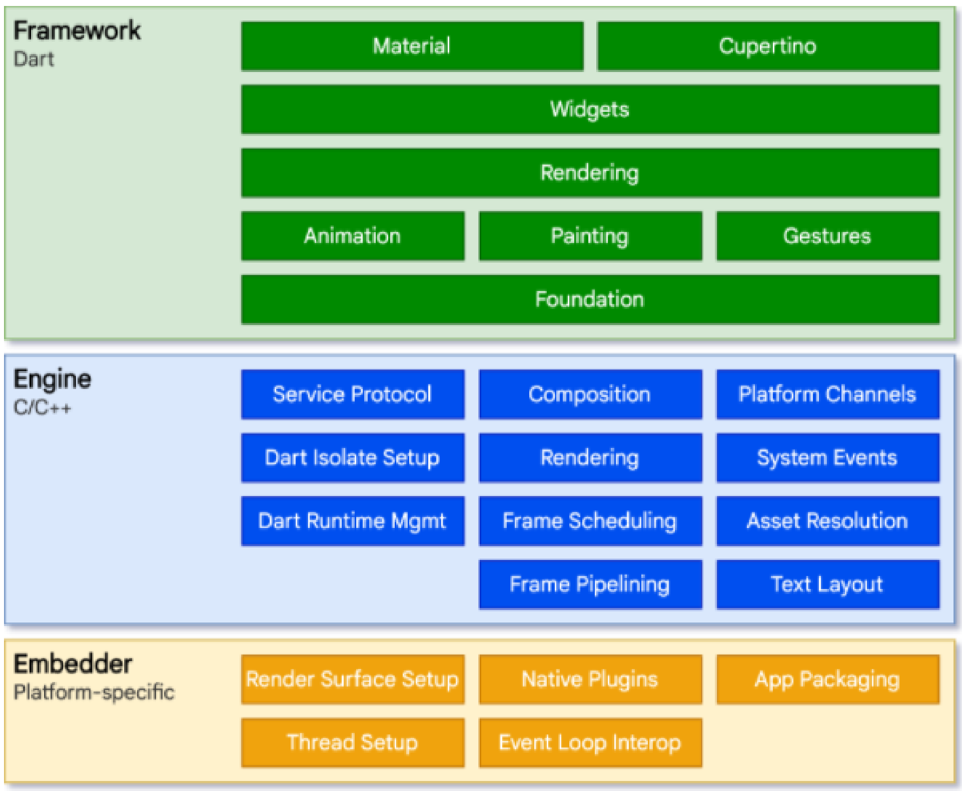


Figure 2.1 Flutter Environment

Widgets are a very important element in the Flutter Framework. Widgets have to be attractive and also make sense because users see and feel them first hand. Widgets not only control and affect the appearance of the app, but also respond to user actions. Therefore, the widget must be able to work quickly including in rendering and animation. Flutter moves widgets and renderers into a single system, which allows widgets to be more customizable for apps as well as extendable within apps. Widgets in Flutter are divided into 2 major, namely Stateless widgets and Stateful widgets. The table below will show the differences between the two widgets:

Table 2.1The differences between Stateless Widget and Stateful Widget

	Dynamic Composition	Itself immutable	Sub State Object mutable
Stateless Widget	False	True	False
Stateful Widget	True	True	True

2.1.3. Firebase

[4] Firebase is a web platform application. Firebase helps developers build high-quality apps. It stores data using the JavaScript Object Notation (JSON) format and Firebase does not use queries to insert, update, delete, or add data to it. Firebase is a back-end system that serves to provide data in the database. Firebase has several features such as:

1. Firebase analytics

   To analyse how users use the application, such as user habits when inside the application.

2. Firebase Cloud Messaging (FCM)

   Paid service on firebase which is a cross-platform solution for messages and notifications on Android, iOS, and web applications.

3. Firebase Auth

It is a service that can authenticate users using only client-side code and is a paid service. It also includes a user management system where developers can enable user authentication with email and password logins stored with Firebase.

4. Real-time Database

   This feature will provide an API for app developers so data can be synced across clients and the Firebase cloud. Client libraries will be provided by companies that have integrity with Android, iOS, and JavaScript applications.

5. Firebase Storage

   This feature facilitates easy and secure document transfer. Firebase Storage is powered by Google Cloud Storage which is a cost-effective object storage service from google. So, developers can store a lot of media such as photos, videos, audio, etc.

6. Firebase Test Lab for Android

   This feature provides cloud-based facilities for testing applications, especially Android. With this feature, developers can test their applications on various devices and configurations. The results of these tests are available in the Firebase Console.

7. Firebase Crash Reporting

   This feature is provided by firebase to be reported if there is an error from the application and can be sorted according to the level of complexity of the problem.

8. Firebase Notification

   This feature is useful for providing notifications to application developers and this service is freely available.

2.1.4. ERD (Entity Relationship Diagram)

Entity Relation Diagram (ERD) is a description of the data modelled in the form of a diagram that is used for data documentation by determining what is contained in each entity and how the relationship between one entity and another (Rahmayu, 2016:34)

2.1.5. UML (Unified Modelling Language)

[6] The Unified Modelling Language (UML) is a family of graphical notations supported by a single meta-model, which helps the description and design of software, especially programs that use the Object-Oriented Programming (OOP) system. The diagrams in UML include:

a) Use case diagram

   This diagram is used to describe the interaction between the user and the system itself, by telling which various systems are used.

b) Activity diagram / Swim-Lane Diagram

   This diagram is used to explain procedural logic, business processes, and work processes.

c) Class diagram

   This diagram illustrates the types of objects in the system and the different types of statistical relationships that exist between these objects.

d) Sequence diagram

   This diagram describes how an object can interact with other objects in the system. Usually used to show scenarios or steps taken in response to an event to produce a certain output.

e) Component diagram

This diagram is used to describe the organization of the system and the dependencies of the application components in the system and is used to show how the program code is divided into modules or components.

f) Deployment diagram

This diagram is used to describe the architecture in nodes for system software as well as other tools in building the architecture on a run-time basis.

2.1.6. Black Box Testing

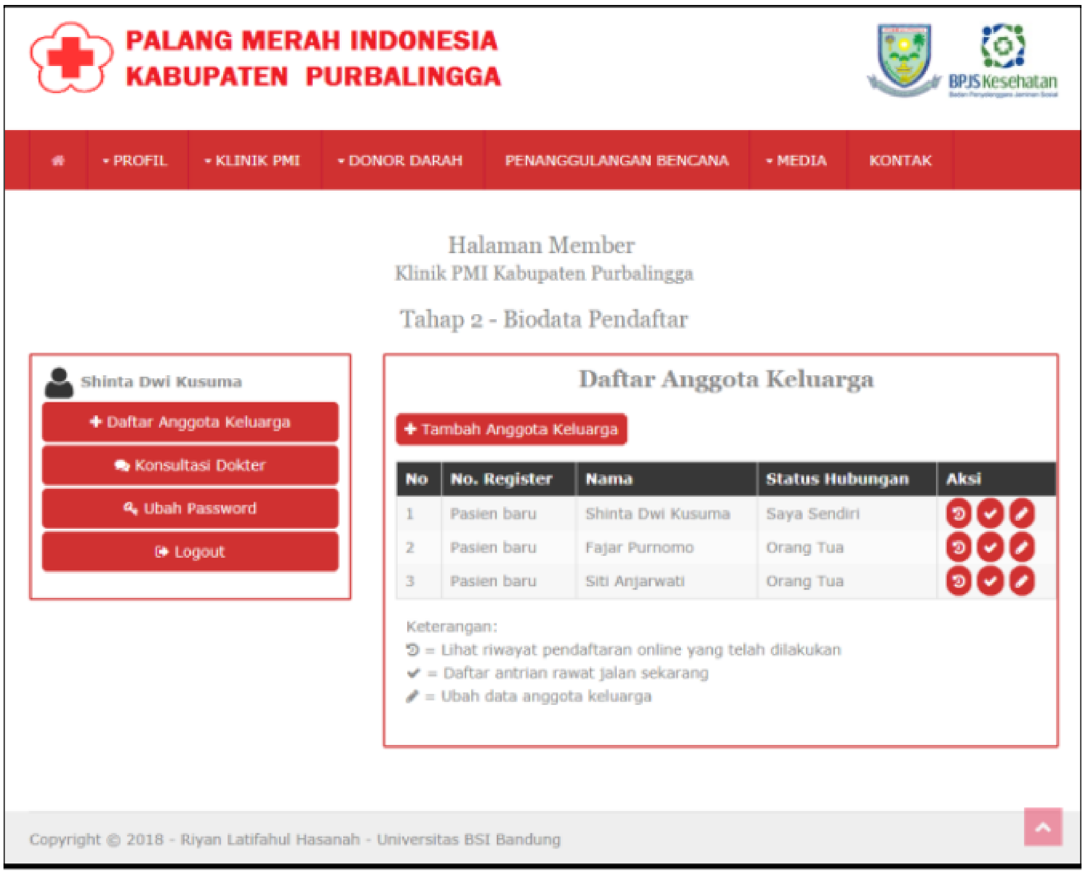
[7] Black Box Testing is a system test that is carried out by running the unit according to a predetermined process. Meanwhile, according to Pressman in Taufik and Ermawati (2018: 3), "Black Box Testing" is a test that allows a developer to get input conditions that fully use all the requirements of a functional program.

2.1.7. Health Consultation

[8] Health Consultation is a way to maintain health, improve health, or prevent the occurrence of a disease that is carried out individually or in groups. Health consultation usually will be done by the individual, family, group, or community.

2.2. Related Work

Several people or organizations in the field of Information Technology have made similar applications that have benefits in the field of Health. According to Khasanah and Wijianto (2018) in their journal entitled "Sistem Informasi Pelayanan Kesehatan Online Berbasis Web Pada PMI Kabupaten Purbalingga", the need for technology is currently very much needed, especially in the health sector because it can speed up registration and data entry so that people can easily get access. health.



Halaman Member
Klinik PMI Kabupaten Purbalingga

Tahap 2 - Biodata Pendaftar

Daftar Anggota Keluarga

No	No. Register	Nama	Status Hubungan	Aksi
1	Pasien baru	Shinta Dwi Kusuma	Saya Sendiri	
2	Pasien baru	Fajar Purnomo	Orang Tua	
3	Pasien baru	Siti Anjarwati	Orang Tua	

Keterangan:
= Lihat riwayat pendaftaran online yang telah dilakukan
✓ = Daftar antrian rawat jalan sekarang
= Ubah data anggota keluarga

Copyright © 2018 - Riyan Latifahul Hasanah - Universitas BSI Bandung

Figure 2.2. Sistem Informasi Pelayanan Kesehatan Online Berbasis Web Pada PMI Kabupaten Purbalingga

And medical consultation mobile application already made by company called Halodoc, the function of this application is to doing online consultation using mobile application.

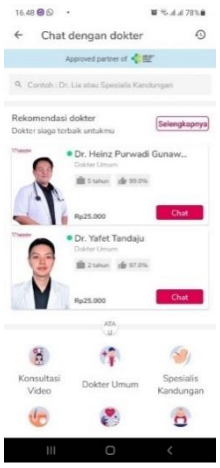


Figure 2.3. Halodoc user interface for mobile consultation

Table 2.2. Comparison with related work

No.	Program Feature	Pelayanan Kesehatan Web Pada PMI Kabupaten Purbalingga	Halodoc	This Application
1.	Sign in and Sign up	Yes	Yes	Yes
2.	Category doctor	No	Yes	Yes
3.	Track Record	No	No	Yes
4.	Update Profile	Yes	Yes	Yes
5.	Free	Yes	No	Yes
6.	Sign in method	Email and Password	Phone Number	Google Account
7.	Sing Out	Yes	Yes	Yes

Based on the two studies above, the authors conclude that the application of health consultations is needed to improve public health. With this application, it can also reduce long queues at the hospital, so that everyone can get health services at the right time and practice. and the advantage of this application is that it has a track record feature so users can see their progress at every doctor they visit.

CHAPTER III

SYSTEM ANALYSIS

In this chapter will explain about the technology function, how its work and how it develops. For this chapter, there will be some list of requirements used to complete the program, thus this chapter will consist of: System Overview, functional Analysis, development process analysis, hardware and software requirement, use case diagram, and many diagram that show the technology flows.

3.1. System Overview

The System will implement the Front-end service which is the screen of the application and then the Back-end service allowed user input the data thru the application to the database. There is no special algorithm used in this application because this application uses manual analysis carried out by doctor.

3.2. Functional Analysis

The functional analysis requirements are the basic for achieving the main goal of Implementation of Public Health Consultation Using Flutter Framework. The Functional Analysis requirements are the main goal to achieve the complete function of the application. The requirements for this final project are listed in Table 3.1.

Table 3.1. Function Description

No.	Function Description
1.	Allow user and doctor login to the application
2.	Allow user get the doctor and doing health consultation

3.	Allow doctor get the user’s health consultation data and store the analysis data to the user

3.3. Development process analysis

The development process analysis is the development process for monitoring system for Implementation of Public Health Consultation Using Flutter Framework is divided into some processes.

3.3.1. Application Development

In this process, the application is built. The application will use to user application include the doctor. Client side will handle the user interface and user experience while the server side handle the data flow to the database server.

3.3.2. Database Setup

In database setup, the developer will setup the database so database can connect to the application, so the application can add, delete, and update the data from database.

3.4. Hardware and Software Requirement

In this final project the application will be create using the following parts:

3.4.1. Hardware Requirements

To develop the mobile application there are several hardware needed to support the developer make the program. The hardware needed listed below:

a. Laptop

Laptop will the place where the application will be made, using Android Studio and using Framework Flutter.

b. Type C Cable

   The type C cable will use to connect the mobile phone to laptop so the program will debug and run directly to the phone.

c. Mobile Phone (Android)

   The developer uses android as the place for program running and debugging. Using mobile phone to check the application if there any bugs or some error.

3.4.2. Software Requirements

To build the application in this final project, there are several software needed to finish this project, the software used in this project is listed below:

a. Android Studio

   The android studio is the IDE will be used to develop the program. The Android Studio already installed the Framework flutter and ready to use.

b. Microsoft Office

   Microsoft Office is used to write the Final Project Documentation for whole process of develop the program.

c. Google Chrome

   Google Chrome is the web browser to open the firebase application and configuration the database.

3.5. Use Case Diagram

This diagram will be used to explain how the system works and the flow between the user and the system is provided in the diagram. Use case diagram is also explaining the system modelling and the process when it is used. The Use case diagram for Implementation of Public Health Consultation Using Flutter Framework is shown in fig. 3.1.

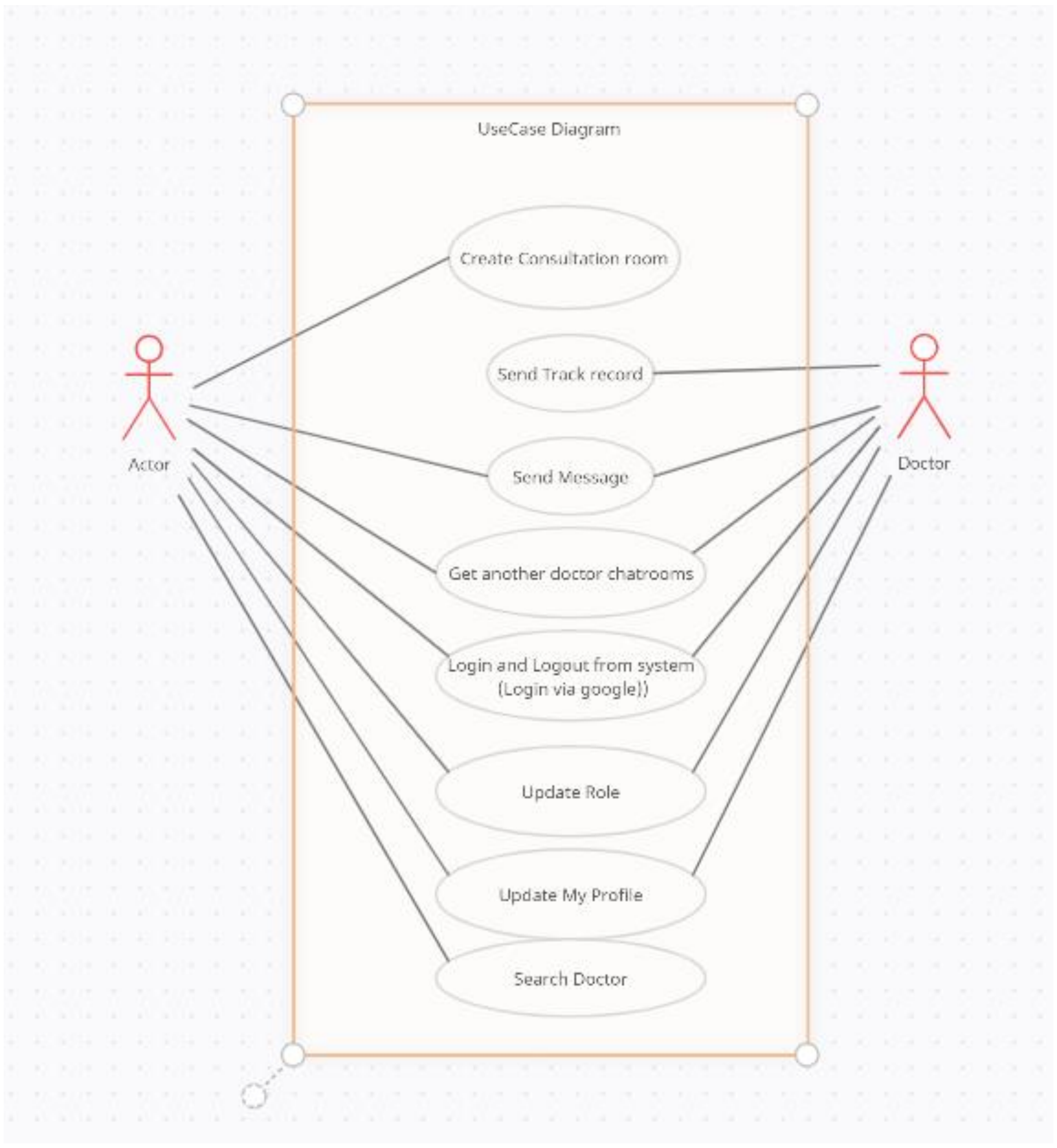


Figure 3.1. Application use case

3.6. Use Case Diagram with Narrative Descriptive

Use case diagram with narrative descriptive is the Use Case Diagram with the textual description. The narrative description is used to explain the mini process from the whole system of this final project application. The description will describe the process based on the functionality. The interaction description consists of several parts which are the case description, process name, precondition, business rule, trigger, conclusion, and post-condition. The use case diagram with narrative descriptive has a goal to explain the process and identify the idea more clearly before the developer starts developing the application.

Table 3.2. Use Case Narrative for User Registration

Use Case Name	User Registration	
Use Case Id	UC01	
Priority	High	
Primary Business Actor	User	
Primary System Actor	Mobile Phone	
Other Participating Actor	None	
Description	Use case explains the user registration to the application	
Trigger	Data successfully sent to the database	
Typical Course of Event	Action User	System Response

	Step 1: User press button login, and choose the google account	
		Step 2: System authentication the user
		Step 3: User automatically set up to database using google account data, and setup as user
	Step 4: User can use application and find the doctor.	
Alternate Course	None	
Post Condition	The data has already been sent to the database by google and automatically set up to database	
Implementation Constraint and Specifications	None	

Table 3.3. Use Case Narrative for Doctor Registration

Use Case Name	Doctor Registration	
Use Case Id	UC02	
Priority	High	
Primary Business Actor	Doctor	
Primary System Actor	Mobile Phone	
Other Participating Actor	None	
Description	Use case explains the doctor registration to the application.	
Trigger	Data successfully sent to the database.	
Typical Course of Event	Action Doctor	System Response
	Step 1: Doctor press button login, and choose the google account	
		Step 2: System authentication the doctor

		Step 3: User upload their certificate in the application and choose their role in the application.
	Step 4: Doctor can use application and find the doctor.	
Alternate Course	None	
Post Condition	The data has already sent to the database and already set up automatically from google account's data	
Implementation Constraint and Specifications	None	

Table 3.4. Use Case Narrative for User Using Application

Use Case Name	User using application
Use Case Id	UC03
Priority	High

Primary Business Actor	User	
Primary System Actor	Mobile Phone	
Other Participating Actor	None	
Description	Use case explains the user login to the application and use the consultation service.	
Trigger	Data successfully sent to doctor.	
Typical Course of Event	Action Doctor	System Response
	Step 1: User login to the application.	
		Step 2: System authentication the doctor
	Step 3: User search doctor in the category page and choose doctor.	
	Step 4: User start chat to the doctor	

		Step 5: System send the data to the doctor.
Alternate Course	None	
Post Condition	The data has already been sent to the doctor and will be analysed by a doctor	
Implementation Constraint and Specifications	None	

Table3.5. Use Case Narrative for Analysis the data

Use Case Name	Doctor Analysis the data user in the application
Use Case Id	UC04
Priority	High
Primary Business Actor	Doctor
Primary System Actor	Mobile Phone
Other Participating Actor	None

Description	Use case explains the doctor analysis the data and send it to the application	
Trigger	Doctor get the data from application	
Typical Course of Event	Action Doctor	System Response
	Step 1: Doctor login to the application	
		Step 2: System authentication the doctor
	Step 3: Doctor analysing the user’s health consultation data	
		Step 4: System send the result to the user.
Alternate Course	None	
Post Condition	The data has already been analysed by the doctor, and the data has been sent to the user.	
Implementation Constraint and Specifications	None	

3.7. Swim-Lane Diagram

A Swim-Lane diagram is a diagram used to visualize the component-based part of the process. This diagram visualizes the process in a view of lanes. The Swim-Lane diagram defines the input-output, main process, and subprocess of the application when the application is running. Swim-Lane diagram has two lanes that will be shown which are the User Lane, the Swim-Lane that will be used to define what users do, and the System Lane, the Swim-Lane Diagram that will be used to define what the System should do.

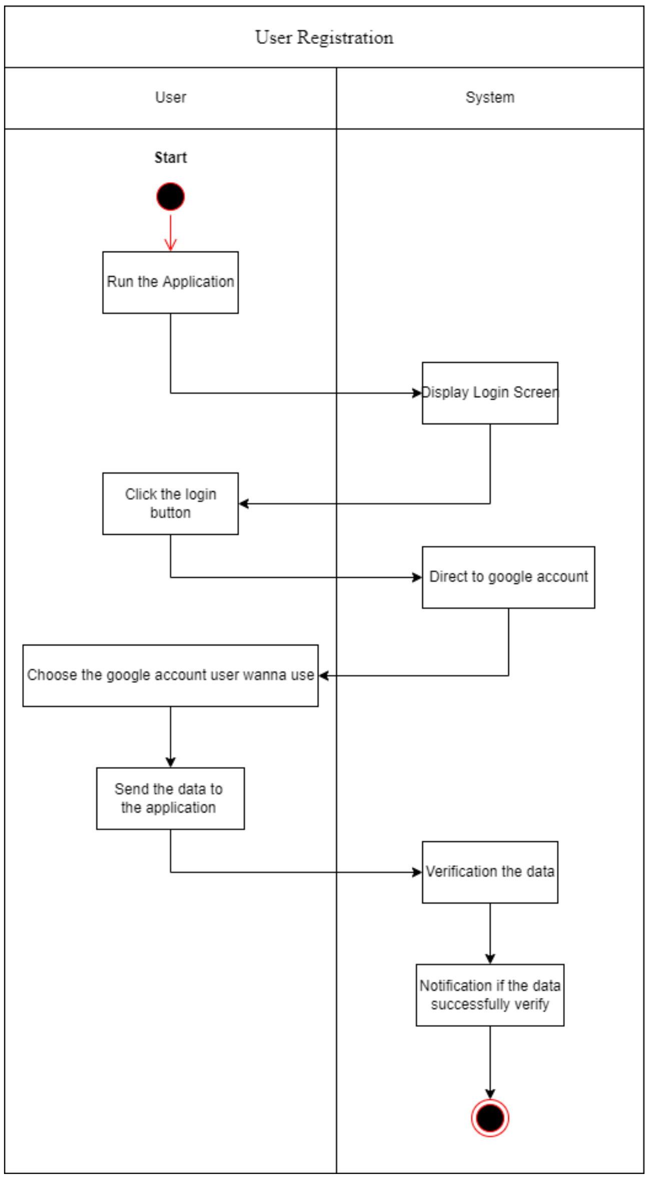


Figure 3.2. User's registration swim lane diagram

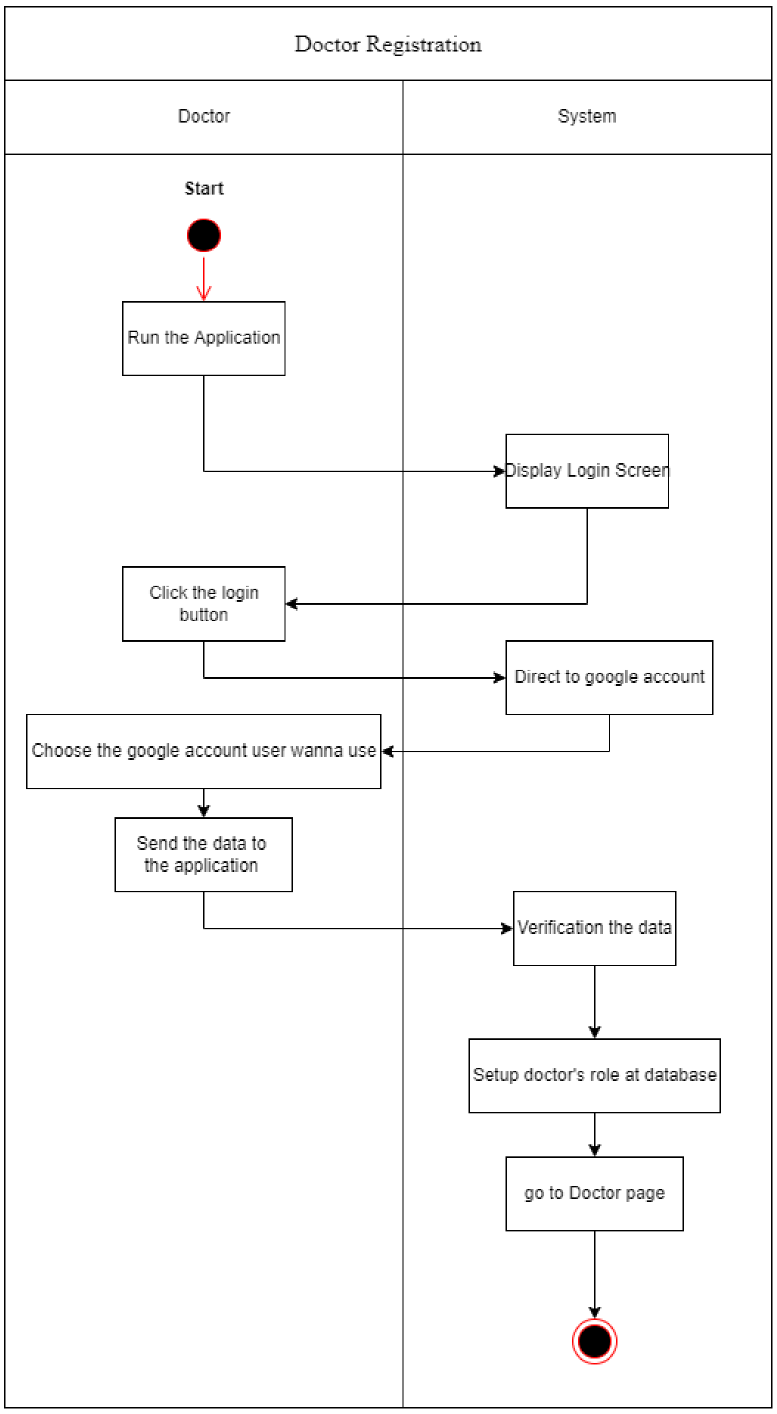


Figure 3.3. Doctor's registration swim lane diagram

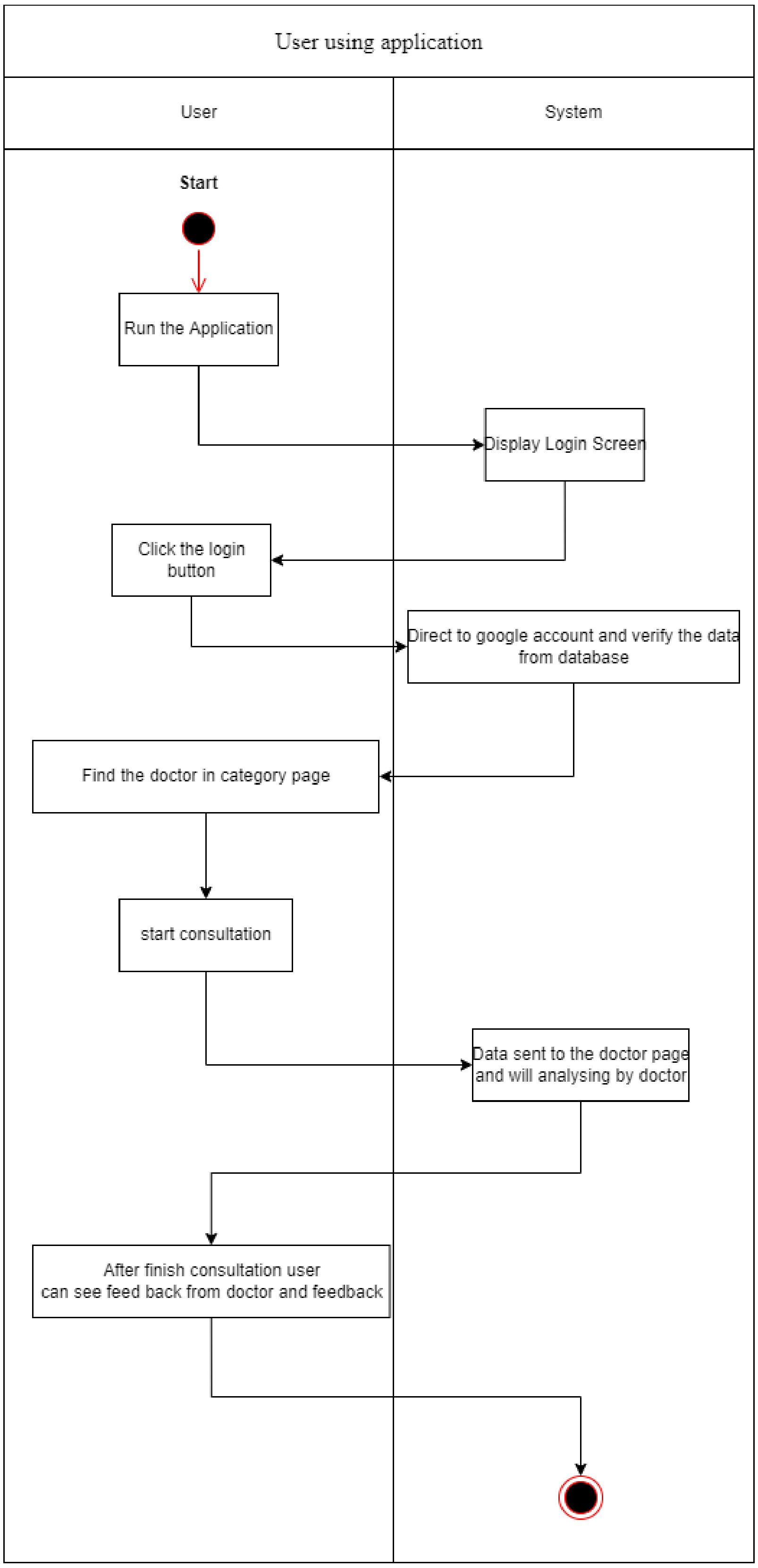

User using application
User
System
Start
Run the Application
Display Login Screen
Click the login button
Direct to google account and verify the data from database
Find the doctor in category page
start consultation
Data sent to the doctor page and will analysing by doctor
After finish consultation user can see feed back from doctor and feedback

Figure 3.4. Swim lane diagram for user using application

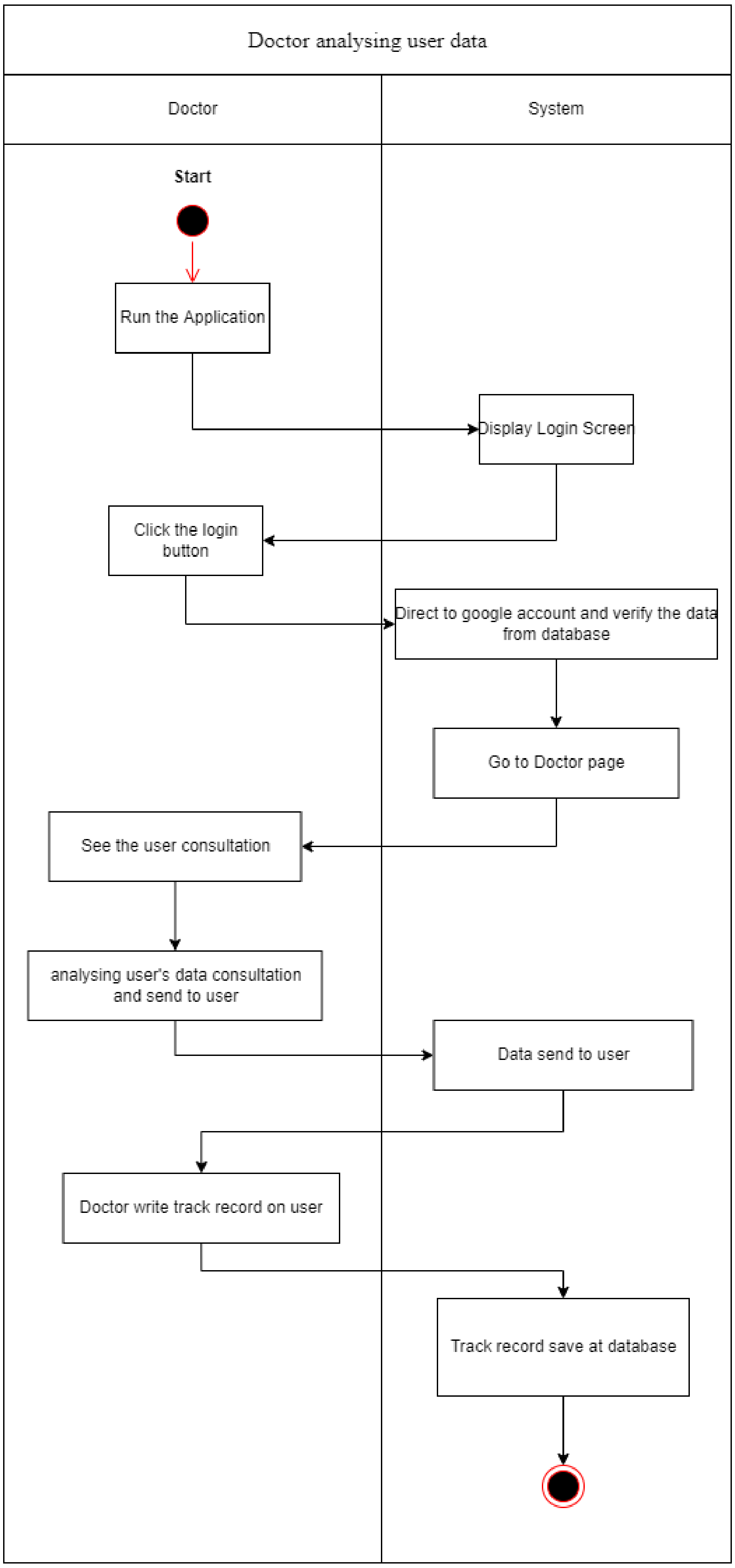

Doctor analysing user data
Doctor
System
Start
Run the Application
Display Login Screen
Click the login button
Direct to google account and verify the data from database
Go to Doctor page
See the user consultation
analysing user's data consultation and send to user
Data send to user
Doctor write track record on user
Track record save at database

Figure 3.5. Swim lane diagram for doctor analysing user data

CHAPTER IV

SYSTEM DESIGN

4.1. User Interface Design

The user interface (UI) for this mobile application is flexible, so the user can use any mobile operating system such as Android and iOS. For this application, there are 4 User interfaces which are Introduction Page, Login Page, User Page, and Doctor Page.

4.1.1. Introduction Page

In this page there are several screens start from splash screen as opening the application using animation after that going to introduction screen, the screen contains several messages from developer to users.

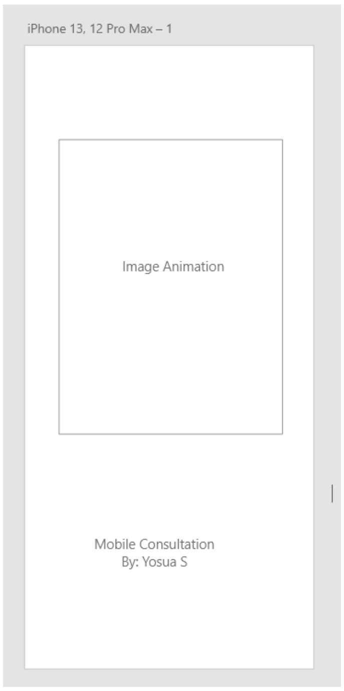


Figure 4.1. Splash screen

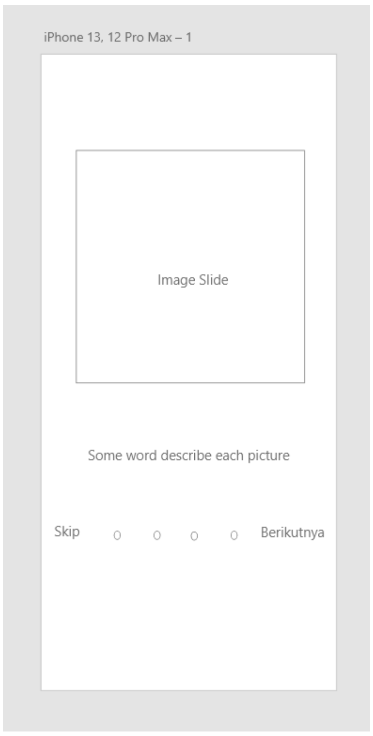


Figure 4.2. Introduction page

In Figure 4.1 and Figure 4.2 display the first page when the user start the application in the splash screen will automatically show when the application start, and the introduction page will show when the application first time installed in the user phone, after that the introduction will never show again.

4.1.2. Login Page

This page just has one button as login and register to the application, for the user

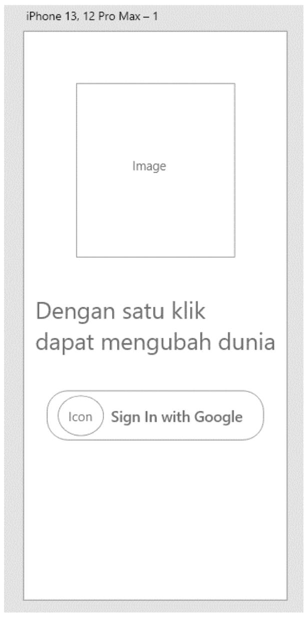

iPhone 13, 12 Pro Max – 1
Image
Dengan satu klik dapat mengubah dunia
Icon
Sign In with Google

Figure 4.3. Login Page Interface

and doctor, the function of the login button is getting the user’s Google account as the user’s data and then putting the data to the database. Then the database will verify the data from a google account and verify the user using that data.

4.1.3. User Page

The user page is the page just user, this page can be only accessed by the user, in this page the user gets some features which are, find a doctor in category screen, see the user’s track record data. User can choose their doctor by the user themselves. In category screen there are 4 category which is “Dokter Umum”,”Dokter Kandungan”,”Dokter Kecantikan”,and ”Dokter kelamin”. The user also can edit their profile in the user page.

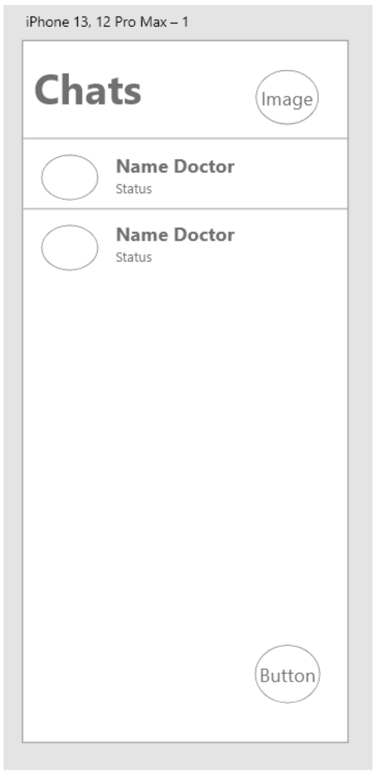


Figure 4.4. User Home Page

In the user's home page will show the last chat with doctors, and then in the corner of the home page will be a floating button, to navigate to maps and to category page.

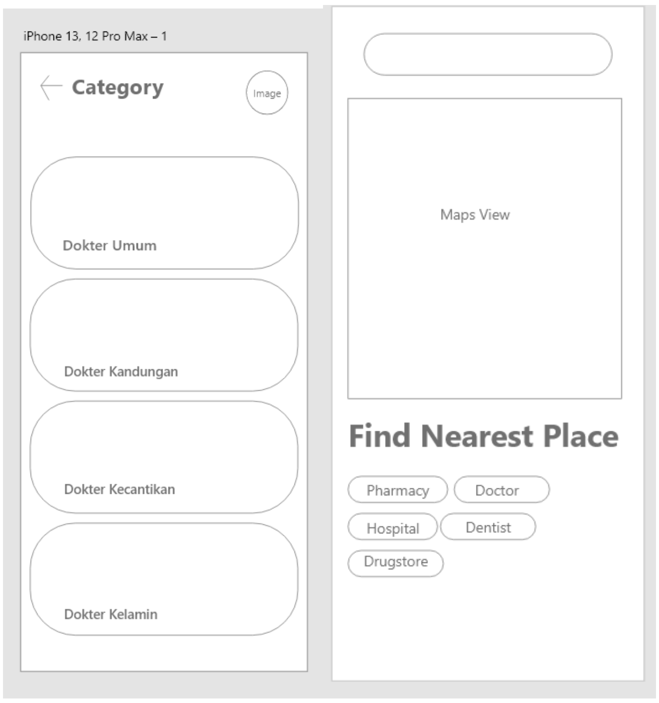


Figure 4.5. Category Screen and Maps Screen at User Interface

In figure 4.5, there is 2 screen the category screen will show the lists of categories of the doctor, when user push the category will show the lists of doctors with the same category. The maps screen will show the map and the location of the user, when user push the button below the maps will show the nearest category to the user.

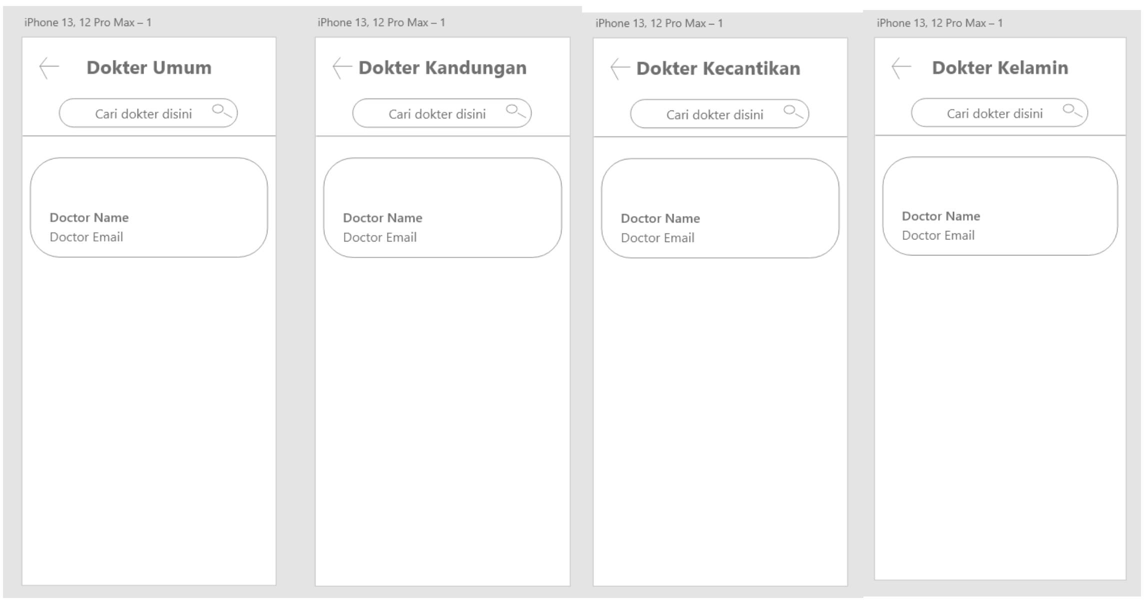


Figure 4.6. Inside Category Screen in User Page Interface

Doctor will be classification for each category in this screen, so the user can find the doctor easily.

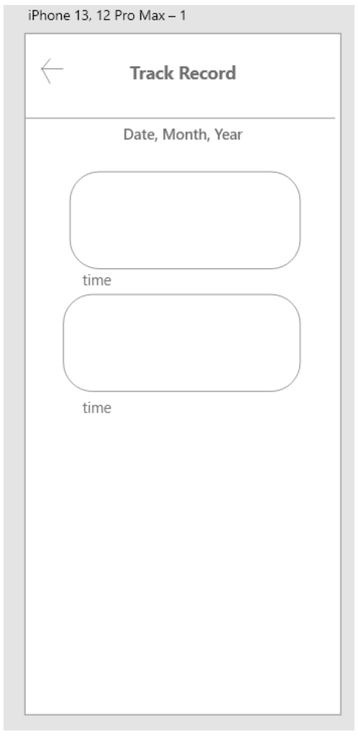


Figure 4.7. User's Track Record in User Page Interface

In this page will use after the user and doctor finish the consultation, and then the doctor will make the track record, so the user can read it.

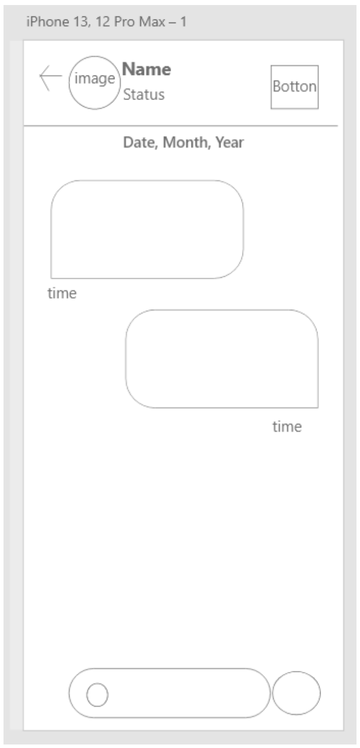


Figure 4.8. User Consultation room with doctor in user Page Interface

This page will use by user and the doctor, this page called consultation room, so the user will consultation with doctor in this page.

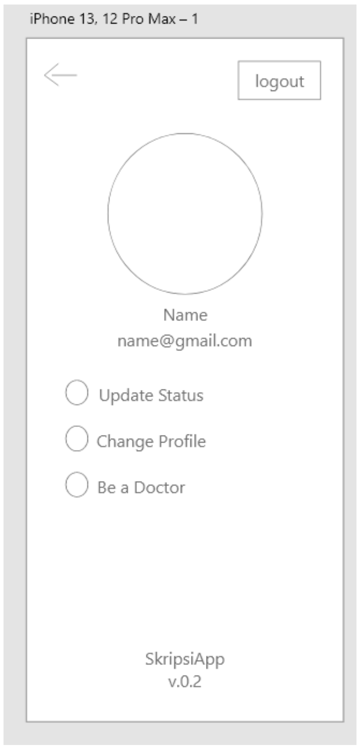


Figure 4.9. User Profile in User Page Interface

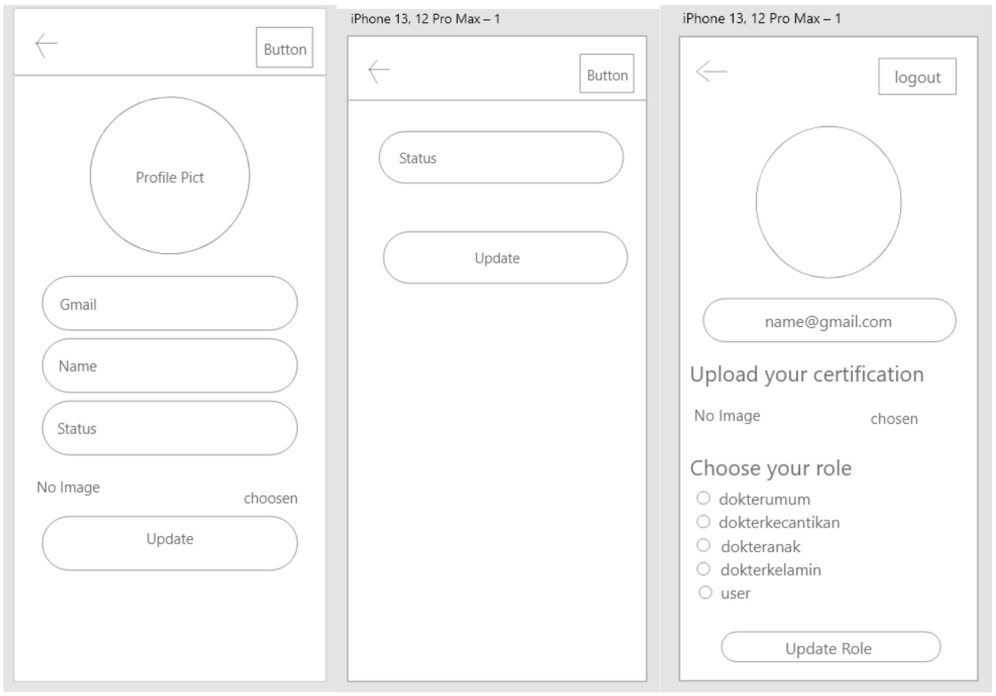


Figure 4.10. User Change profile, Status, be doctor in User Page Interface

In figure 4.9 and figure 4.10 is the user page in the page will be the palace user can maintain their user information.

4.1.4. Doctor Page

The doctor page is simpler than the user page just has 2 screens which are the Home Page for the doctor include the consultation room and only the doctor can write the track record.

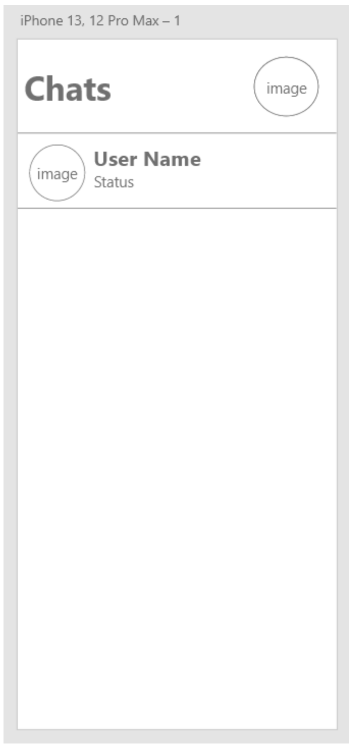


Figure 4.11. Doctor Home Page in Doctor Page Interface

In this page will show the user want to chat to the doctor.

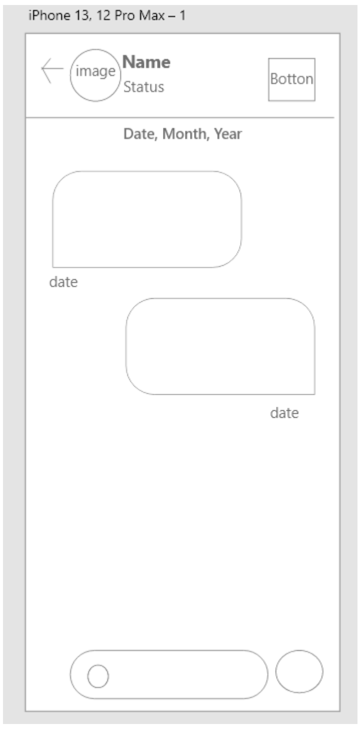


Figure 4.12. Doctor Consultation Room in Doctor Page Interface

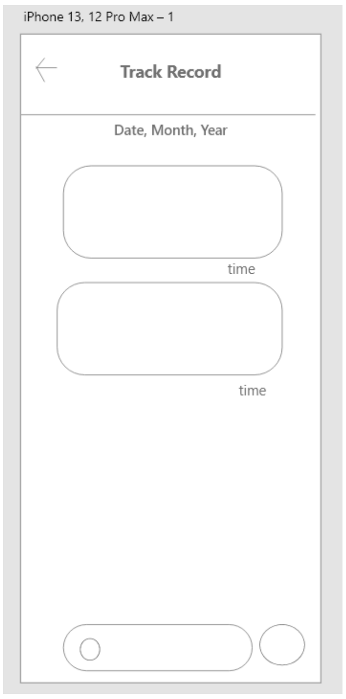


Figure 4.13. Doctor Track Record to User in Doctor Page Interface

In figure 4.12 and figure 4.13 there is consultation page and track record page, consultation page using by doctor to take the user consultation after the user consultation finish the doctor will give the user track record in the track record page

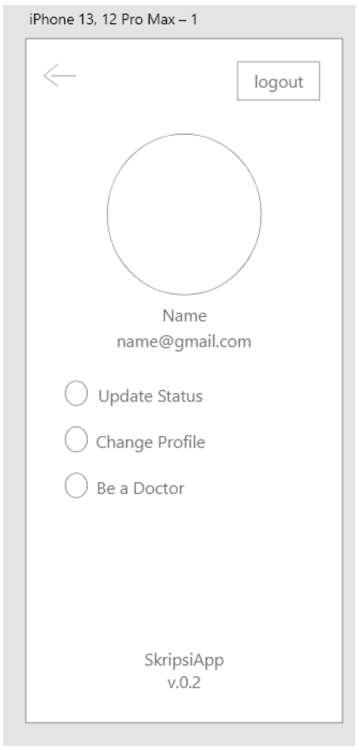


Figure 4.14. Doctor Profile Screen in Doctor Page Interface

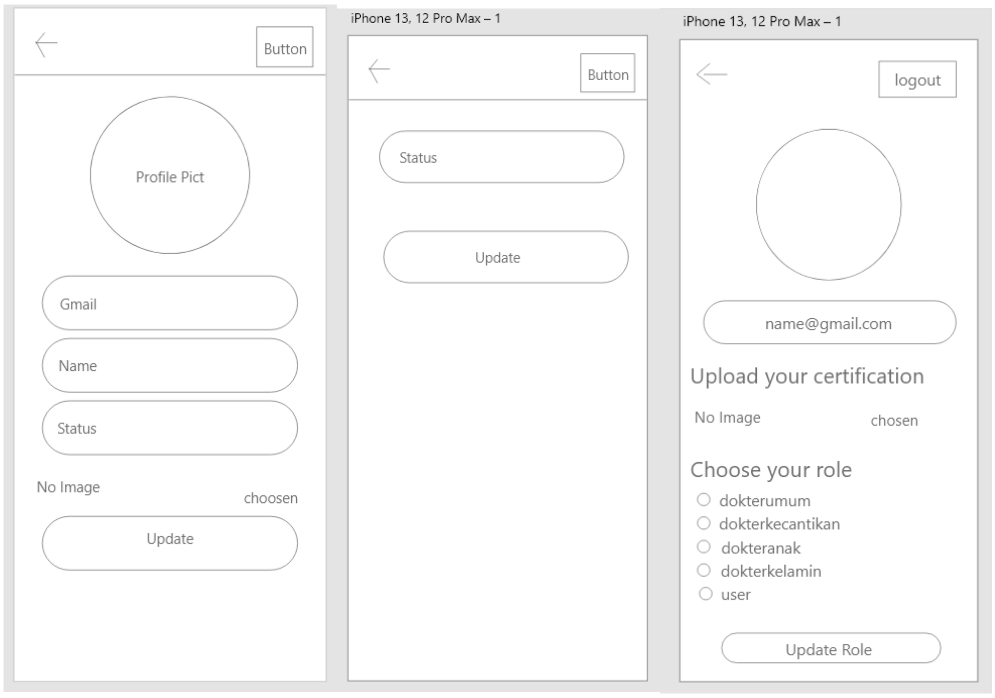
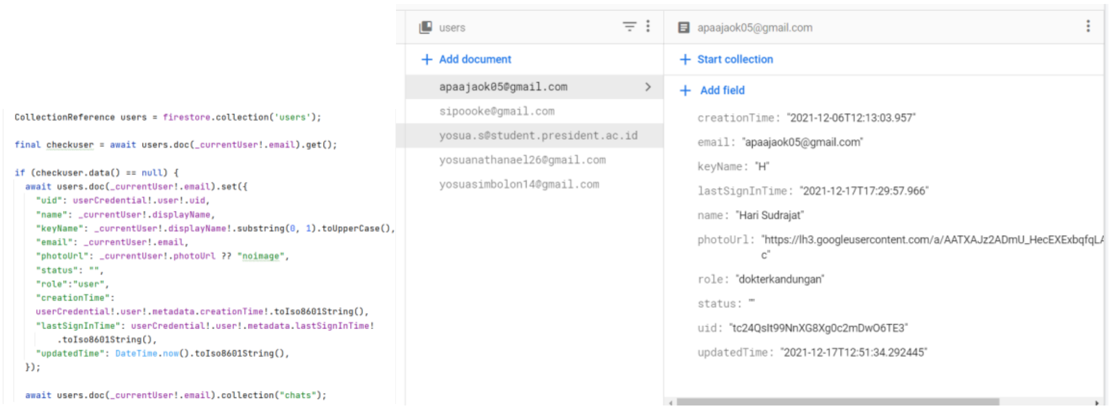


Figure 4.15. Doctor Change Profile and Status Screen in Doctor Page Interface

In figure 4.14 and figure 4.15 is the user page in the page will be the palace user can maintain their user information.

4.2. Database Design

In this application, the author uses the firebase database, which is different from SQL and Postgres. To create a table in the flutter, the database was created directly from the program.



```
CollectionReference users = firestore.collection('users');

final checkuser = await users.doc(_currentUser!.email).get();

if (checkuser.data() == null) {
  await users.doc(_currentUser!.email).set({
    "uid": userCredential!.user!.uid,
    "name": _currentUser!.displayName,
    "keyName": _currentUser!.displayName!.substring(0, 1).toUpperCase(),
    "email": _currentUser!.email,
    "photoUrl": _currentUser!.photoUrl ?? "noimage",
    "status": "",
    "role":"user",
    "creationTime":
    userCredential!.user!.metadata.creationTime!.toIso8601String(),
    "lastSignInTime": userCredential!.user!.metadata.lastSignInTime!
        .toIso8601String(),
    "updatedTime": DateTime.now().toIso8601String(),
  });

  await users.doc(_currentUser!.email).collection("chats");
```

users

Add document

apaajaok05@gmail.com
sipoooke@gmail.com
yosua.s@student.president.ac.id
yosuanathanael26@gmail.com
yosuasimbolon14@gmail.com

apaajaok05@gmail.com

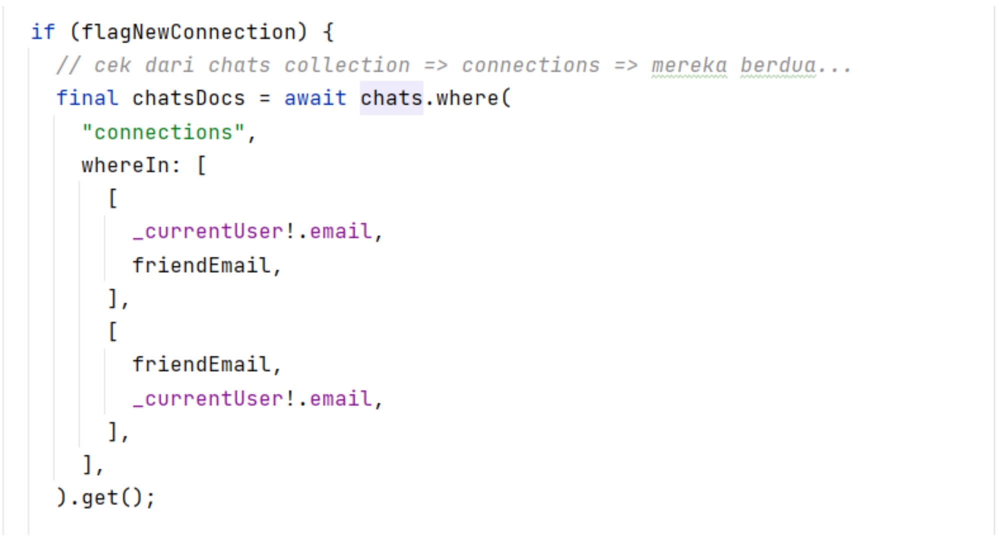
Start collection

Add field

```
creationTime: "2021-12-06T12:13:03.957"
email: "apaajaok05@gmail.com"
keyName: "H"
lastSignInTime: "2021-12-17T17:29:57.966"
name: "Hari Sudrajat"
photoUrl: "https://lh3.googleusercontent.com/a/AATXAJz2ADmU_HecEXExbqfqL
          c"
role: "dokterkandungan"
status: ""
uid: "tc24QsIt99NnXG8Xg0c2mDwO6TE3"
updatedTime: "2021-12-17T12:51:34.292445"
```

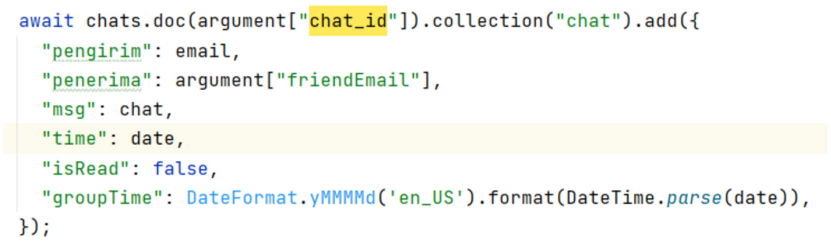
Figure 4.16 Code to make users collection in database

Figure 4.16 shows the code to create user collection. User collection has 10 attributes: *uid* as Primary Key (PK), *name*, *keyName*, *email*, *photoUrl*, *status*, *role*, *creationTime*, *lastSignInTime* and *updateTime.*



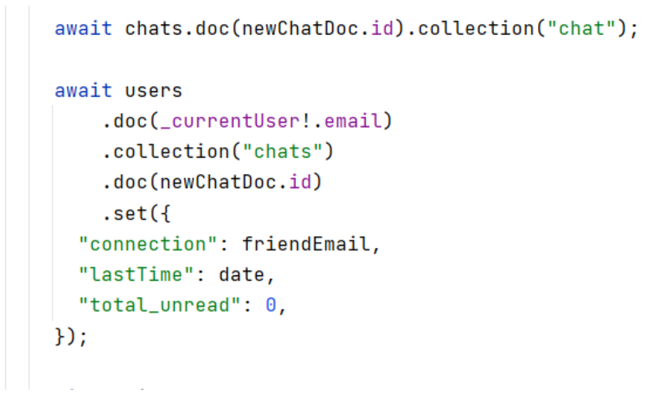
```
if (flagNewConnection) {
  // cek dari chats collection => connections => mereka berdua...
  final chatsDocs = await chats.where(
    "connections",
    whereIn: [
      [
        _currentUser!.email,
        friendEmail,
      ],
      [
        friendEmail,
        _currentUser!.email,
      ],
    ],
  ).get();
```

Figure 4.17. Create connections to chats collection



```
await chats.doc(argument["chat_id"]).collection("chat").add({
  "pengirim": email,
  "penerima": argument["friendEmail"],
  "msg": chat,
  "time": date,
  "isRead": false,
  "groupTime": DateFormat.yMMMMd('en_US').format(DateTime.parse(date)),
});
```

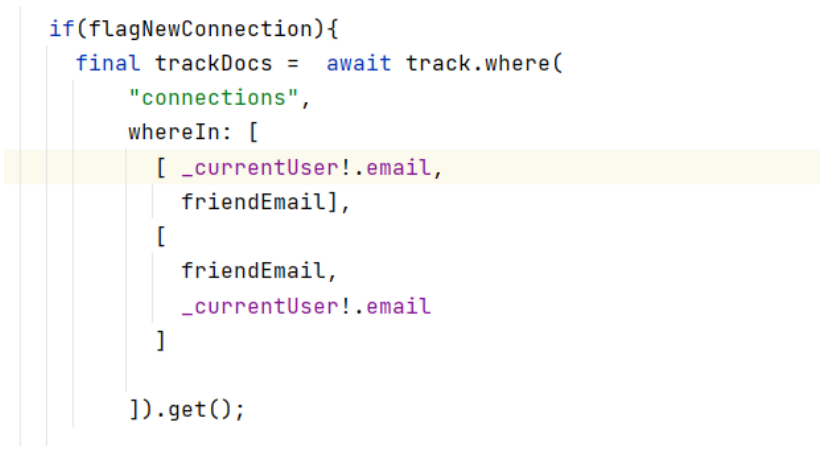
Figure 4.18. Create chat collection in firebase



```
await chats.doc(newChatDoc.id).collection("chat");

await users
    .doc(_currentUser!.email)
    .collection("chats")
    .doc(newChatDoc.id)
    .set({
  "connection": friendEmail,
  "lastTime": date,
  "total_unread": 0,
});
```

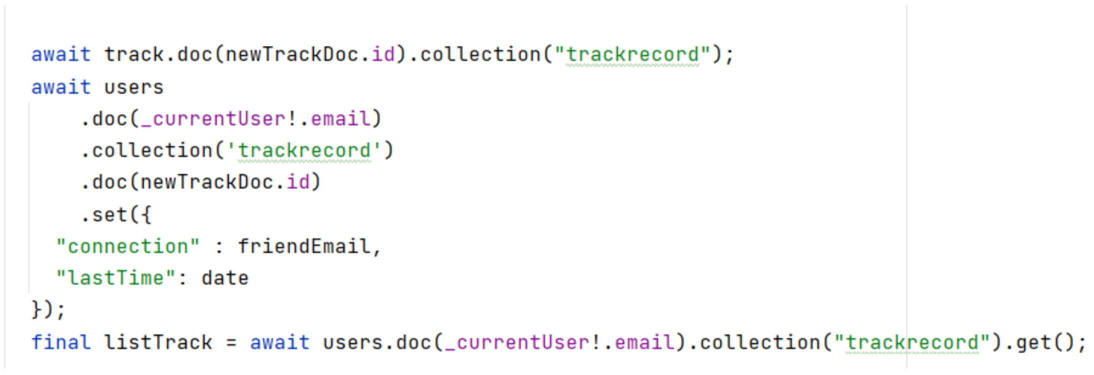
Figure 4.19. create chat collection in user collection in firebase

Above there are two images that have a relationship, the first in Figure 4.17. serves to make a connection between the sender of the message and the recipient of the message. this collection also serves to create a *chat_id* which will be useful in the next image. In figure 4.17 contain connections collection. In figure 4.18. code serves to create chat collections that are useful for storing chat data made by senders and users, in this collection contains *pengirim, penerima, msg, isRead, and groupTime*. In figure 4.19 create chat collection in user collection, so in user collection there is chat data consist of *connection*, *lastTime*, and total_unread.



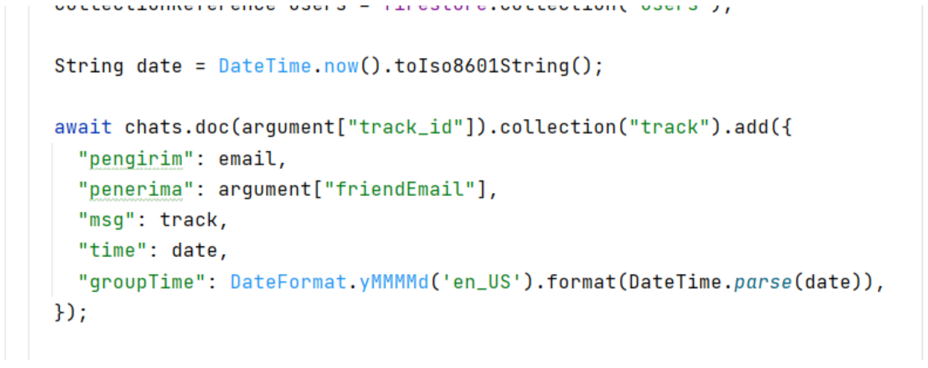
```
if(flagNewConnection){
  final trackDocs =  await track.where(
      "connections",
      whereIn: [
        [ _currentUser!.email,
          friendEmail],
        [
          friendEmail,
          _currentUser!.email
        ]

      ]).get();
```

Figure 4.20. Create connection in trackRecord collection



```
await track.doc(newTrackDoc.id).collection("trackrecord");
await users
    .doc(_currentUser!.email)
    .collection('trackrecord')
    .doc(newTrackDoc.id)
    .set({
  "connection" : friendEmail,
  "lastTime": date
});
final listTrack = await users.doc(_currentUser!.email).collection("trackrecord").get();
```

Figure 4.21. Set trackrecord data to users' collection



```
String date = DateTime.now().toIso8601String();

await chats.doc(argument["track_id"]).collection("track").add({
  "pengirim": email,
  "penerima": argument["friendEmail"],
  "msg": track,
  "time": date,
  "groupTime": DateFormat.yMMMMd('en_US').format(DateTime.parse(date)),
});
```

Figure 4.22. Create track collection in firebase

In three figures above, figure 4.20 is to create connection field in trackrecord collection, to connect the user to the doctor, so the doctor able to write the track record, trackrecord collection just have one field is just *connection*. In figure 4.21 is to input trackrecord connection to the user's data, in this collection consist of *connection* and *lastTime.* In figure 4.22 is to create a track collection the function of this collection to place a track record message be placed, so the data can easily put on UI.

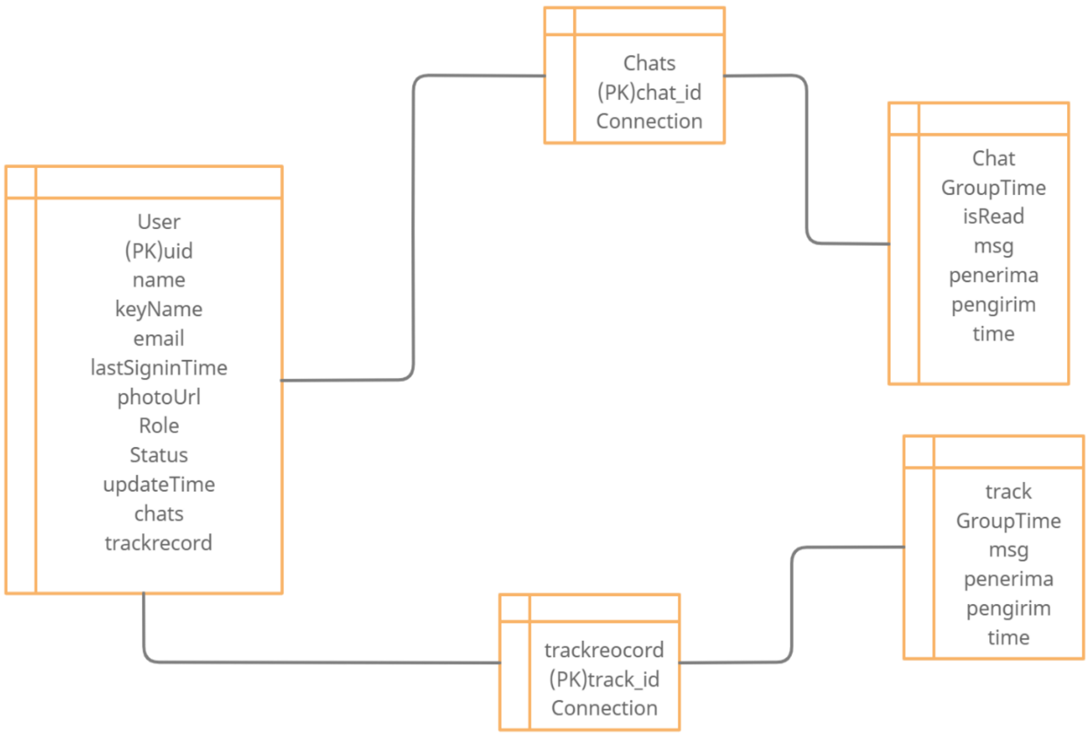


Figure 4.23. Database relationship for each collection

The figure above explains about collection relationship for each collection in the firebase database. User collection connect to Chats and track record collection, these collections have their id, and the for Chats collection have branched to contain Chat collection to be a chat data repository so that it is more structured, and same with a track record have a branch to contain track record collection to be a track record data repository.

CHAPTER V

SYSTEM DEVELOPMENT

In this chapter will explain about the implementation of the system to the application, The implementation of the chat from user to doctor will explain in this chapter. The application will develop using Flutter ver. 2.5.3, Dart 2.14.4, and Android Studio ver. 2020.3. In this chapter the system User Interface Development and Application Details are discussed.

5.1. User Interface Development

This application has many pages, but the pages can group into 4 main pages, there are Introduction Page, Login Page, User Page, and Doctor Page.

On the introduction page, the application will show the splash screen as the beginning screen when the user opens the application, after the splash screen there are two possibilities if the user never uses the application the introduction page will disappear and directly go to the login page, but if the user uses the application for the first time there will be introduction screen consist four pages intro to the application but if the user wants to skip it, it's can skipped because there is a button for skip the introduction. The introduction contains a message from the developer to the user.

In this application, there is no registration page because this application uses a google account as the account for the user, so in the Login page there is just have one button and the button will connect to the google account, so the user can choose the google account they want to use. If the user chooses the google account so the google account will send user data such as name, email, photo picture, etc so after pushing the button user can directly use the application without any verification.

In this application will divided by 2 users they are user and doctor, so the main page will divide by 2 too, first is user page, in user page user can find and choose their doctor in the category page. users can choose the doctor they want according to their needs, in the user page, there are 4 categories namely *"dokter umum"*, "*dokter kecantikan"*, *"dokter kandungan"*, and also *"dokter kelamin"*. At the time of selecting the user can also search for the doctor they want, if there is any doctor user want the system will display it. The consultation room make like chat app so the user can use the application easily. User can see their track record in the chat room and then the user can custom their name, status, and their profile picture.

On the doctor page the doctor just can wait for the user to contact them, after contact and doing consultation doctor can write the user's track record in the chat room so the user can see their track record. In doctor, page doctor can change their name, status, and their profile picture but cannot change the rule, because the rule can change only by developer.

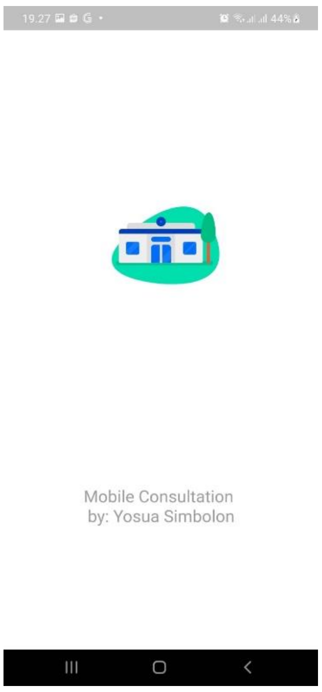
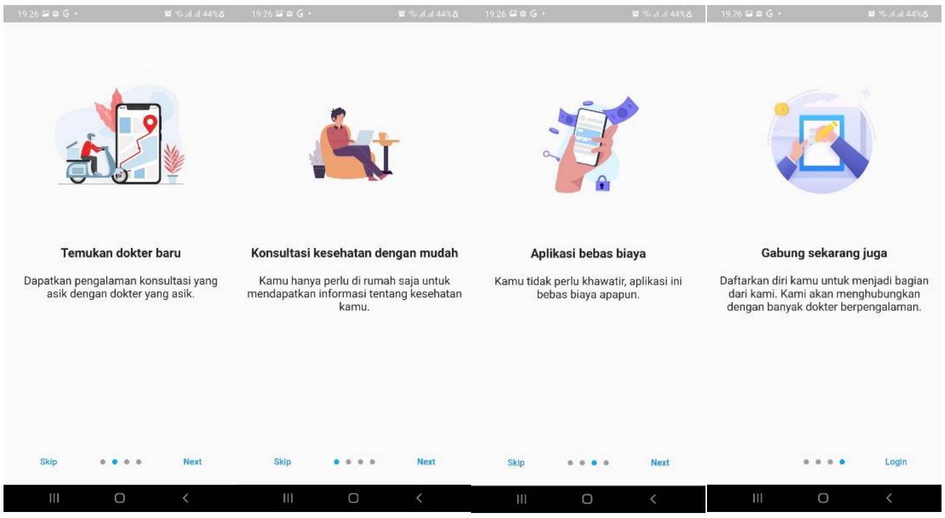


Figure 5.1. Splash screen



Temukan dokter baru

Dapatkan pengalaman konsultasi yang asik dengan dokter yang asik.

Konsultasi kesehatan dengan mudah

Kamu hanya perlu di rumah saja untuk mendapatkan informasi tentang kesehatan kamu.

Aplikasi bebas biaya

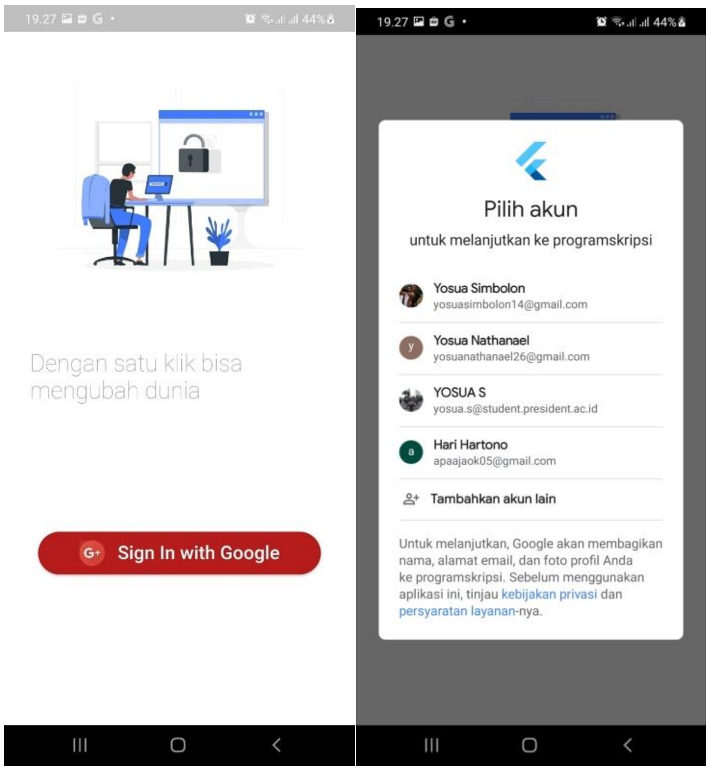
Kamu tidak perlu khawatir, aplikasi ini bebas biaya apapun.

Gabung sekarang juga

Daftarkan diri kamu untuk menjadi bagian dari kami. Kami akan menghubungkan dengan banyak dokter berpengalaman.

Figure 5.2. Introduction page

In figure 5.1 and figure 5.2 will show the splash screen and introduction scree, this page will be the first page will show when the user starts the application. Splash screen show when the user starts the application after that the splash screen will continue with introduction page.



Dengan satu klik bisa mengubah dunia

Pilih akun

untuk melanjutkan ke programskripsi

Yosua Simbolon
yosuasimbolon14@gmail.com

Yosua Nathanael
yosuanathanael26@gmail.com

YOSUA S
yosua.s@student.president.ac.id

Hari Hartono
apaajaok05@gmail.com

Tambahkan akun lain

Untuk melanjutkan, Google akan membagikan nama, alamat email, dan foto profil Anda ke programskripsi. Sebelum menggunakan aplikasi ini, tinjau kebijakan privasi dan persyaratan layanan-nya.

Figure 5.3. Login Page Interface

Figure 5.3 is the login page, in this application using login system by google, so the user will login automatically using google Gmail.

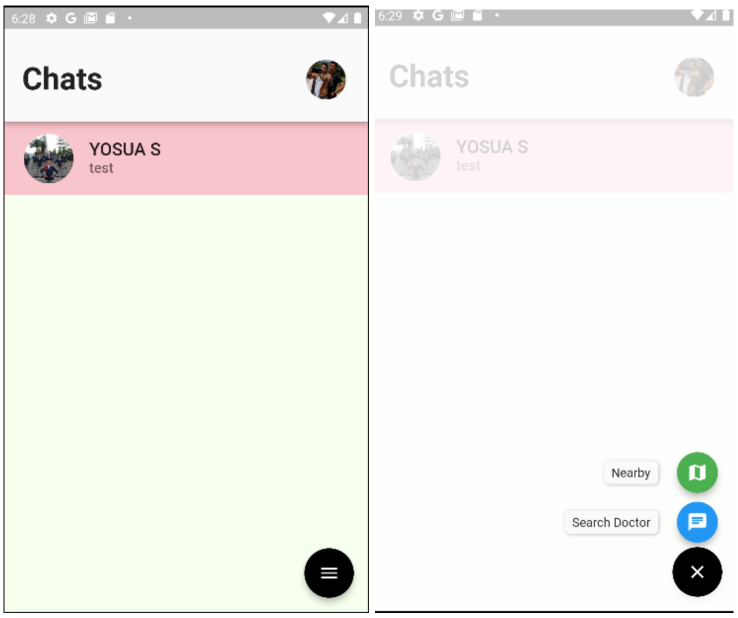


Figure 5.4. User Home Page

In the figure 5.4 user home page will show the history chat with the doctor, in this page the user can press the floating button and search doctor in the category page, not even search doctor the user can open the map page and find the nearest doctor from the user location.

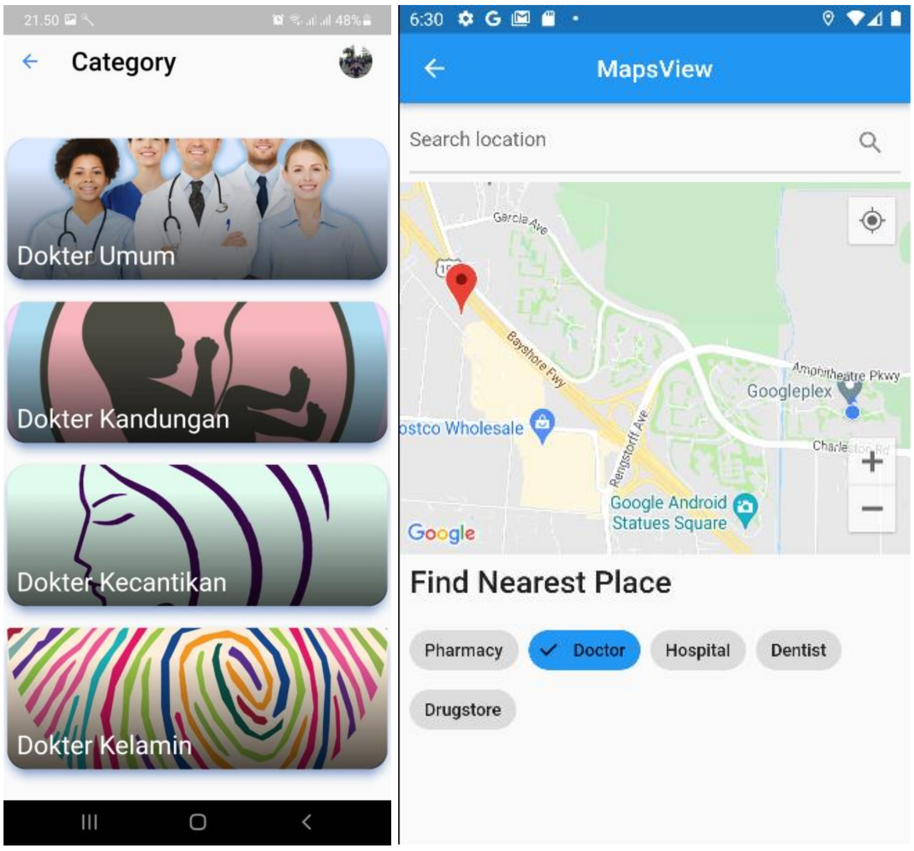


Figure 5.5. Category Screen at User Interface

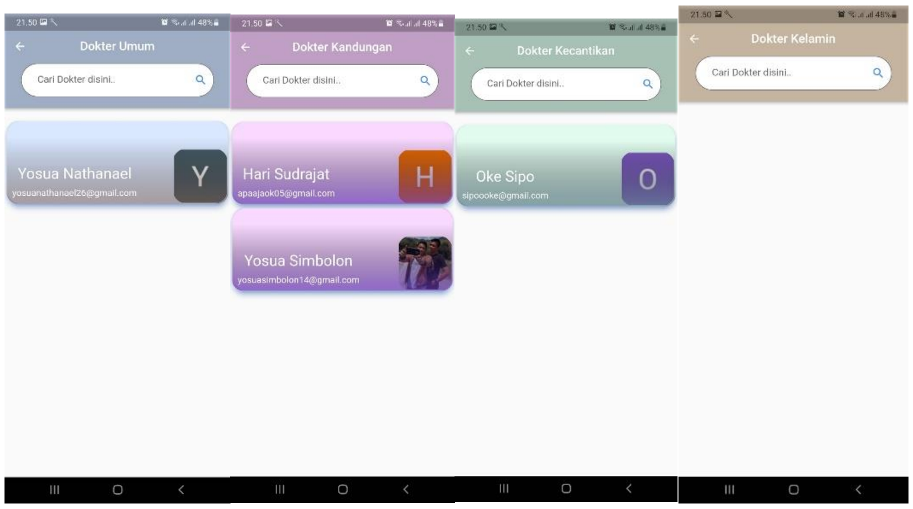
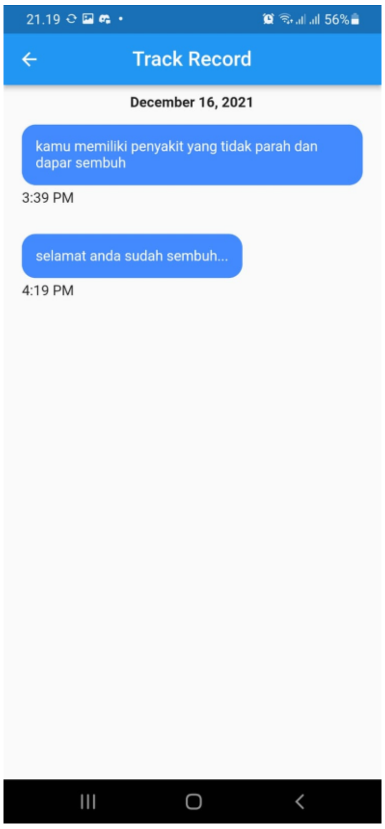


Figure 5.6. Inside Category Screen in User Page Interface

In the figure 5.5 and figure 5.6 there is the category page and maps page, the category page will show the doctor base the category and make the list. In the maps page will show the map and the user location, so the user can find the nearest place and doctor near user based on user location.



21.19 56%

Track Record

December 16, 2021

kamu memiliki penyakit yang tidak parah dan dapar sembuh

3:39 PM

selamat anda sudah sembuh...

4:19 PM

Figure 5.7. User's Track Record in User Page Interface

After finish the consultation the user can see the trac record given by the doctor, in this page the user just can see the track record, cannot edit or delete it.

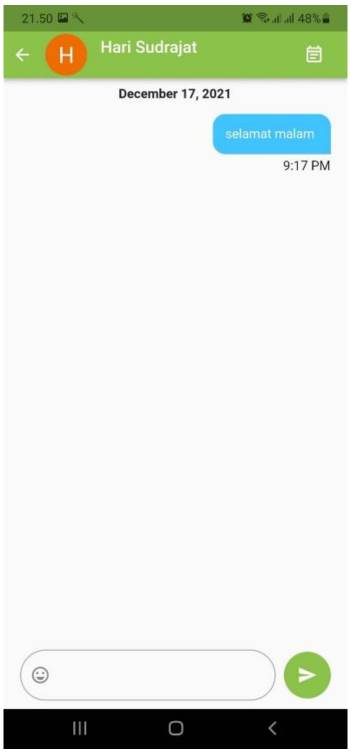


Figure 5.8. User Consultation room with doctor in user Page Interface

In this page the user doing consultation with the doctor, user can give their data to the doctor, so the doctor can analyse the disease.

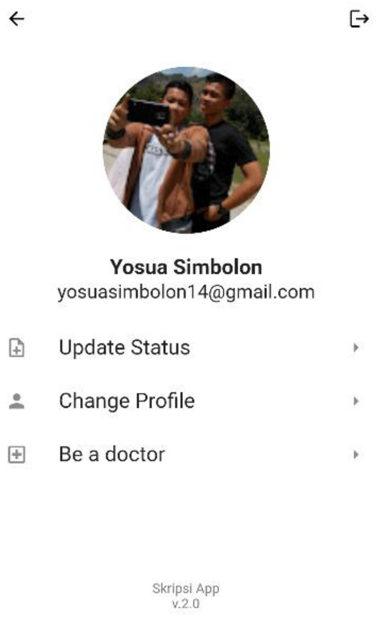
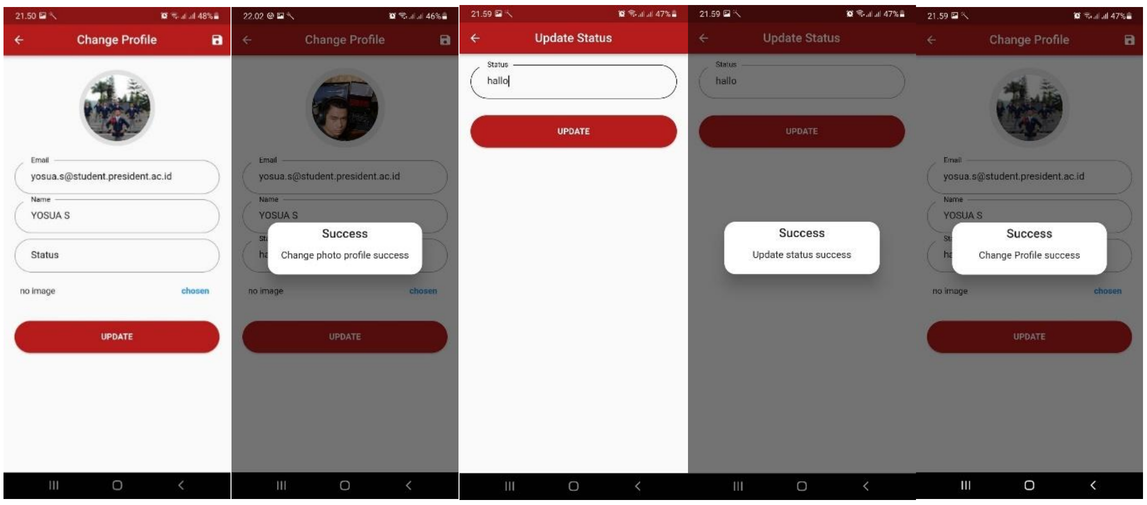


Figure 5.9. User Profile in User Page Interface

In figure 5.9 the user can maintain their data like change their status, manage their profile picture and their name, and also for user who want to be doctor can register in the be doctor page.


Change Profile

Email: yosua.s@student.president.ac.id

Name: YOSUA S

Status

no image chosen

UPDATE

Success
Change photo profile success

Update Status

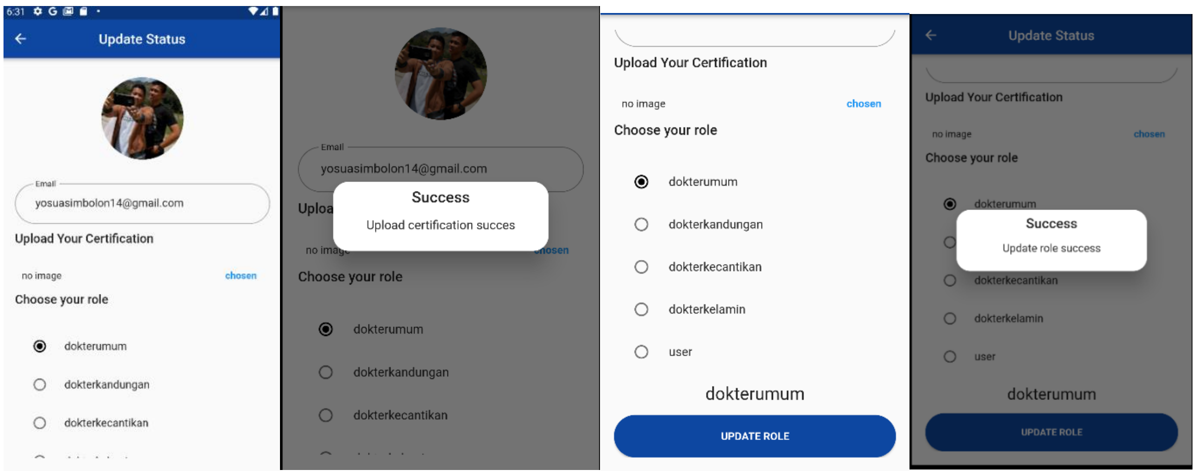
Status: hallo

UPDATE

Success
Update status success

Success
Change Profile success

Figure 5.10. User Change profile and Status in User Page Interface


Update Status

Email: yosuasimbolon14@gmail.com

Upload Your Certification

no image chosen

Choose your role

[✓ dokterumum]
dokterkandungan
dokterkecantikan
dokterkelamin
user

Success
Upload certification succes

dokterumum

UPDATE ROLE

Success
Update role success

Figure 5.11. User Change Role in User Page Interface

In figure 5.10 and figure 5.11 explain about the user maintain the user data, user can change the photo profile, status, and also name, but user cannot change their email, and in this page the user can also register to be doctor, the user can upload their certificate and choose their role and this change will be automatically change the user role at database.

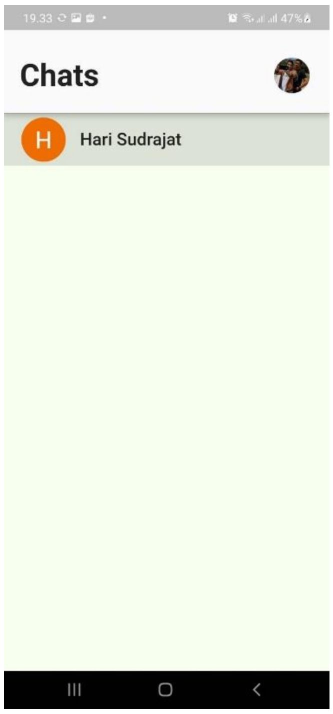
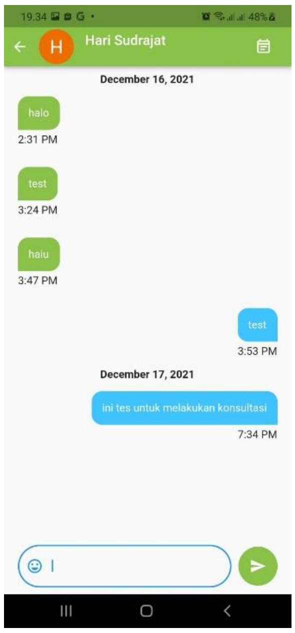


Figure 5.12. Doctor Home Page in Doctor Page Interface

In this page will show the doctor chat with the user. Doctor cannot search the user and also make the connection to the user, except the user itself. The doctor just waits the user chat them and then start the consultation.



December 16, 2021

halo
2:31 PM

test
3:24 PM

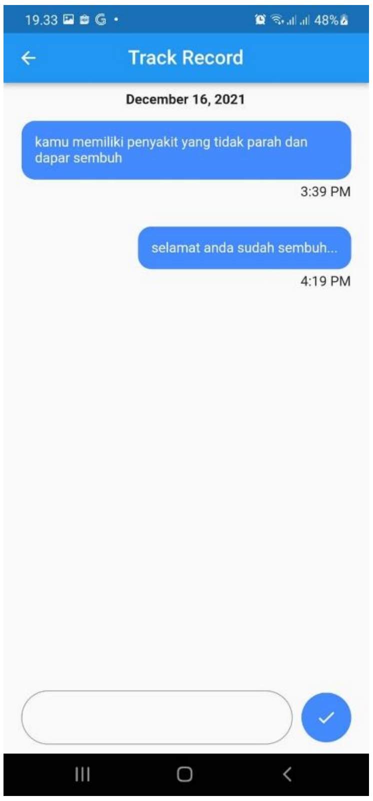
haiu
3:47 PM

test
3:53 PM

December 17, 2021

ini tes untuk melakukan konsultasi
7:34 PM

Figure 5.13. Doctor Consultation Room in Doctor Page Interface



December 16, 2021

kamu memiliki penyakit yang tidak parah dan dapar sembuh

3:39 PM

selamat anda sudah sembuh...

4:19 PM

Figure 5.14. Doctor Track Record to User in Doctor Page Interface

The doctor doing consultation in the chat room with the user, so the doctor can diagnose the disease of the user when user chat doctor, after finish the consultation the doctor will give the user track record, in this situation doctor can give the track record to the user.

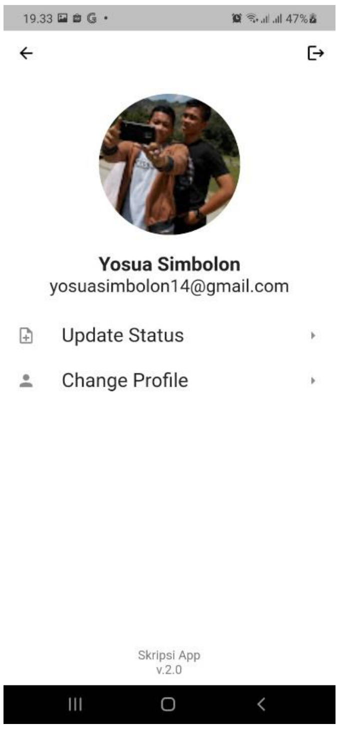


Figure 5.15. Doctor Profile Screen in Doctor Page Interface

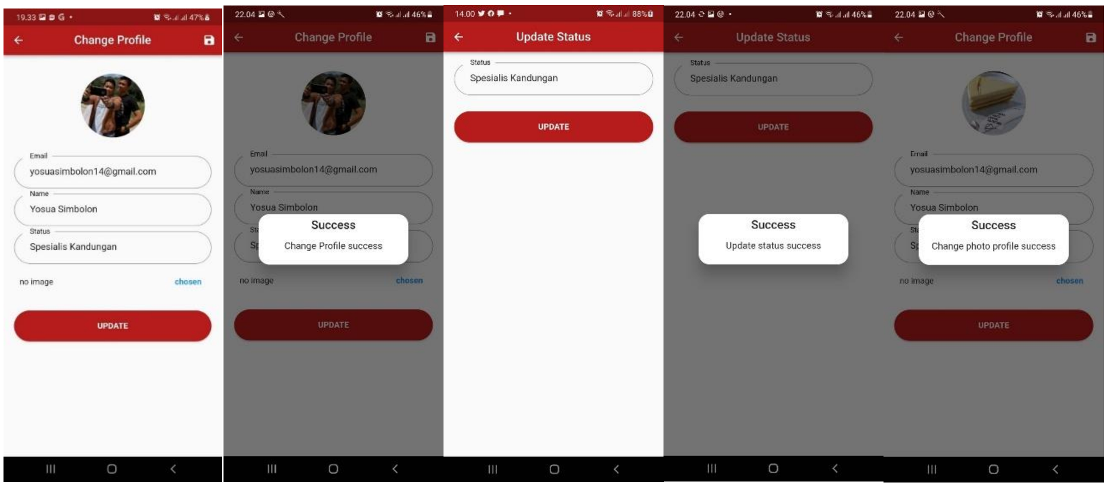


Figure 5.16. Doctor Change Profile and Status Screen in Doctor Page Interface

In this page the doctor can maintain their data such as change name, status and also their profile picture.

5.2. Application Details

This part explains about codes that are used to build in this mobile application. In this section will explain just the important code. This Mobile Application made using Flutter Frame work and using dart language. This application using GetX as State Management.

When made this application divided into 3 parts: the screen, the models, and the controllers.

5.2.1. Screens

When user open this application, the first thing user can get is the splash screen, splash screen build as the opening in this application. The splash screen called in the main activity by FutureBuilder() function, so when application running the splash screen will build and run as the first screen. In this application made by many widgets after the splash screen will continue with introduction screen, the introduction screen made by widget called IntroductionScreen()this widget generates the function index to be several screens and the style can be customized from the developer.

```
return Scaffold(
  body: IntroductionScreen(
    pages: [
       PageViewModel(
                                 …… ),
      PageViewModel(
                                 ….),
      PageViewModel(
                                 …. ),
      PageViewModel(

      ),
    ],
    onDone: () => function()
    showSkipButton: true,
    skip: Text(…)
    next: Text(….. ),
    done: const Text( ….)

);
```

Figure 5.17. Introduction widget in introduction page

if the page already finishes and the user want to move to another screen because this application using GetX state management so the navigator function called: Get.offAllNamed(Routes….) after that going to the next page. The most important widget in this application is ListView.builder() because this widget can change data in firebase to a list. ListView.builder() can get data if this widget wrap using StreamBuilder() go get the data.

```
StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
  stream: controller
        return ListView.builder(
        controller: controller
        itemCount: data.length,
        itemBuilder: (context, index) {
    ItemChat(  the style of the list ),
```

Figure 5.18 ListView widget

In this application screen using assets lottie, lottie is an image animation formatted by JSON, the Lottie usual put in the splash screen, introduction screen, and login screen. The Lottie called using Lottie.*asset*() function. The formation of this application using Column()and Row()to arrange widgets according too the design.

```
Column(
    children: [
    Container(
      child: Row(
        Children:[

                ]
        )
    ),
    ),)
  ],
),
```

Figure 5.19. Column and row function

To show give notification if some function doing well or there are some errors the application will pop up the dialog widget, the dialog function contains about error message or just a message to be indicator if the application doing well or not.

```
Get.defaultDialog(title: "……………", middleText: "………");
```

Figure 5.20. Dialog widget

In this application user also can search the doctor too, there is have search bar connected to the database, so when the user wants to search the doctor, the search bar will show it up. To show the search bar the program using ListView.builder() to show the doctor.

```
ListView.builder(
  padding: EdgeInsets.zero,
  itemCount: controller.tempSearch.length,
  itemBuilder: (context, index) => Card(
      elevation: 8,
      shadowColor: Colors.blueAccent,
      shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(24)),
      child: InkWell(
        onTap: () {
          authC.addNewConnection(
            controller.tempSearch[index]["email"],
          );
        },
        child: Container(
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(20),
            color: Colors.blueAccent.withOpacity(0.2),
          ),

          height: 150,
          width: double.infinity,
          child: Stack(
            children: [
              style….
                    child: Image.network(
                      controller.tempSearch[index]["photoUrl"],
                      height: 100,width: 100,
                    ),
                  )),
                    child: Column(
                      children: [
                        Text(
                          controller.tempSearch[index]["name"],
                          style: TextStyle(
                              color: Colors.white, fontSize: 25),
                        ),
                        SizedBox(height: 10,),
                        Text(
                          controller.tempSearch[index]["email"],
                          style: TextStyle(
                              color: Colors.white, fontSize: 15),
                        ),

                      ],
                    ),
                  ))
            ],
          ),
        ),
      )),
)
```

Figure 5.21. To show doctor in the search screen

5.2.2. Controllers

The application cannot show the data and send the data without the controllers because controllers have a function to manage the flow of the application, because this application using GetX state management the controllers file build automatically when developer create the file page using command "*get create page: #thepagename"* in the terminal. When user login to the application controllers will get data from google account and store the data to the data base.

```
Future<void> login() async {
  try {
        await _googleSignIn.signOut();
 await _googleSignIn.signIn().then((value) => _currentUser =
value);
final isSignIn = await _googleSignIn.isSignedIn();
    if (isSignIn) {
      print("LOGIN SUCCESS GO TO THE NEXT PAGE: ");
      print(_currentUser);

      final googleAuth = await _currentUser!.authentication;

      final credential = GoogleAuthProvider.credential(
        idToken: googleAuth.idToken,
        accessToken: googleAuth.accessToken,
      );

      await FirebaseAuth.instance
          .signInWithCredential(credential)
          .then((value) => userCredential = value);
 print("USER CREDENTIAL");
      print(userCredential);
      CollectionReference users = firestore.collection('users');

      final checkuser = await
users.doc(_currentUser!.email).get();

      if (checkuser.data() == null) {
        await users.doc(_currentUser!.email).set({
          set the user in database
      } else {
        Set username log history in database
              }

      user.refresh();

      isAuth.value = true;
      Get.offAllNamed(Routes.HOME);
    } else {
      print("TIDAK BERHASIL LOGIN");
    }
```

Figure 5.22. login function in controller

Everything in the backend will handle in the controllers, when user already login in the application the introduction will skipped so from splash screen will directly go to the login page because when login the login:

```
final box = GetStorage();
//box.write('skipIntro', true);
if (box.read('skipIntro') != null) {
  box.remove('skipIntro');
}
box.write('skipIntro', true);
```

Figure 5.23. GetStorate to create local storage in device

From picture 5.23. when user login for the first time the application will make a local storage and will write data “skipIntro”, so when the application running if there is a data called “skipIntro” the introduction page will skip and go directly to the home page if the user already login or goes to the login page if the user already logout. This local storage get control in controller and called in the main screen.

```
Future<void> firstInitialized() async {
await skipIntro().then((value) {
    if (value) {
      isSkipIntro.value = true;
    }
  });
}
```

Figure 5.24. Initialized the skipIntro function in controller

```
return FutureBuilder(
  future: authC.firstInitialized(),
  builder: (context, snapshot) => SplashScreen(),
```

Figure 5.25. Calling Initialized function in the main page

In figure 5.25. the controller function called in the main page so when running the application, the initialized function will check the local storage first.

In this application when user already login and close the application, and then open the application the application will still login as the user login before because in the controller already make auto login function, the function system is like skip intro but the auto login did not use local storage as the indicator, the auto login using google auth

```
Future<bool> autoLogin() async {
    try {
    final isSignIn = await _googleSignIn.isSignedIn();
    if (isSignIn) {
      await _googleSignIn
          .signInSilently()
          .then((value) => _currentUser = value);
      final googleAuth = await _currentUser!.authentication;

      final credential = GoogleAuthProvider.credential(
        idToken: googleAuth.idToken,
        accessToken: googleAuth.accessToken,
      );

      await FirebaseAuth.instance
          .signInWithCredential(credential)
          .then((value) => userCredential = value);

      print("USER CREDENTIAL");
      print(userCredential);


      CollectionReference users = firestore.collection('users');

      await users.doc(_currentUser!.email).update({
        "lastSignInTime":
userCredential!.user!.metadata.lastSignInTime!.toIso8601String(),
      });

      final currUser = await
users.doc(_currentUser!.email).get();
      final currUserData = currUser.data() as Map<String,
dynamic>;

      user(UsersModel.fromJson(currUserData));

      user.refresh();
```

Figure 5.26. autoLogin function in Controller

In figure 5.26. the autologin function will check if there any google account still sign in in this application. Type of this function is Boolean so if in this application there are already signed in the function will have value true, if not the function will have value false. Value will process in the Initialized function like skipIntro function and then the function will call in the main page.

```
Future<void> firstInitialized() async {
  await autoLogin().then((value) {
    if (value) {
      isAuth.value = true;
    }
```

Figure 5.27. initialized autoLogin function in controller

After initialized the autologin function the initialized function will call in the main page. User can login this application, the controller will take the function, it's called logout function in this function google account who already signed in application will remove and after remove the function will navigate the screen to the login screen.

```
Future<void> logout() async {
  await _googleSignIn.disconnect();
  await _googleSignIn.signOut();
  Get.offAllNamed(Routes.LOGIN);
}
```

Figure 5.28 logOut function in controller

The function will put on the button signOut in user page, and will automatically signOut from the application.

When user doing consultation with doctor the list of the doctor will show in list widget and get the data from controllers, the controllers called getAllUsers() this function will take all specific user data in database and the user’s data can process to the list.

```
Stream<List<UsersModel>>getAllUsers()=>
    collectionReference.where("role", isEqualTo: filter
data).snapshots().map((query) =>
        query.docs.map((item) =>
UsersModel.fromMap(item)).toList()
    );
```

Figure 5.29. getAllUsers function to get specific data in the database

This application able to change username and status these features get controlled by function called changeProfile()in controllers. This function can change the data that has been provided by the previous google account’s data.

```
void changeProfile(String name, String status) {
  String date = DateTime.now().toIso8601String();
  CollectionReference users = firestore.collection('users');

  users.doc(_currentUser!.email).update({
    "name": name,
    "keyName": name.substring(0, 1).toUpperCase(),
    "status": status,
    "lastSignInTime":

userCredential!.user!.metadata.lastSignInTime!.toIso8601String(),
    "updatedTime": date,
  });
  user.update((user) {
    user!.name = name;
    user.keyName = name.substring(0, 1).toUpperCase();
    user.status = status;
    user.lastSignInTime =

userCredential!.user!.metadata.lastSignInTime!.toIso8601String();
    user.updatedTime = date;
  });
  user.refresh();
  Get.defaultDialog(title: "Success", middleText: "Change Profile
success");
}
void updateStatus(String status) {
  String date = DateTime.now().toIso8601String();
  CollectionReference users = firestore.collection('users');

  users.doc(_currentUser!.email).update({
    "status": status,
    "lastSignInTime":

userCredential!.user!.metadata.lastSignInTime!.toIso8601String(),
    "updatedTime": date,
  });
```

Figure 5.30 Update User Function

In figure 5.30. the function updates the users' s database, the function will read the database first and then then in the function called ".update*()"* to update the data in firebase, and then in this function the data "updatedTime"will updated automatically, after update done the dialog widget will pop up and give message to user if update user already done and success.

After update name and status, this application can update photo, this application using upadateUrl function in controller.

```
void updatePhotoUrl(String url) async {
  String date = DateTime.now().toIso8601String();
  // Update firebase
  CollectionReference users = firestore.collection('users');

  await users.doc(_currentUser!.email).update({
    "photoUrl": url,
    "updatedTime": date,
  });

  // Update model
  user.update((user) {
    user!.photoUrl = url;
    user.updatedTime = date;
  });

  user.refresh();
  Get.defaultDialog(
      title: "Success", middleText: "Change photo profile
success");
}
```

Figure 5.31 updatePhotoUrl function in controller

```
void selectImage() async {
  try {
    final checkDataImage =
        await imagePicker.pickImage(source: ImageSource.gallery);

    if (checkDataImage != null) {
      print(checkDataImage.name);
      print(checkDataImage.path);
      pickedImage = checkDataImage;
    }
    update();
  } catch (err) {
    print(err);
    pickedImage = null;
    update();
  }
}
```

Figure 5.32 selectImage function in controllers to the firebase and then take the URL name. the selectImage function get the image URL and then send to the updatePhotoUrl function to replacte the newest URL from before.

Doing the consultation like chat with the doctor, but the first person to star contact is the user not the doctor, doctor just wait the user chat and start consultation with them. The consultation room or chat room build from ListView.builder(), but to make the list

```
void addNewConnection(String friendEmail) async {
  final docChats =
  await
users.doc(_currentUser!.email).collection("chats").get();

  if (docChats.docs.length != 0) {
    final checkConnection = await users
        .doc(_currentUser!.email)
        .collection("chats")
        .where("connection", isEqualTo: friendEmail)
        .get();

    if (checkConnection.docs.length != 0) {
      flagNewConnection = false;
      chat_id = checkConnection.docs[0].id;
    } else {
      flagNewConnection = true;
    }
  } else {
    flagNewConnection = true;
  }
```

Figure 5.32 selectImage function in controllers

```
if (chatsDocs.docs.length != 0) {
  // terdapat data chats (sudah ada koneksi antara mereka
berdua)
  final chatDataId = chatsDocs.docs[0].id;
  final chatsData = chatsDocs.docs[0].data() as Map<String,
dynamic>;

  await users
      .doc(_currentUser!.email)
      .collection("chats")
      .doc(chatDataId)
      .set({
    "connection": friendEmail,
    "lastTime": chatsData["lastTime"],
    "total_unread": 0,
  });

  final listChats =
  await
users.doc(_currentUser!.email).collection("chats").get();

  if (listChats.docs.length != 0) {
    List<ChatUser> dataListChats = List<ChatUser>.empty();
    listChats.docs.forEach((element) {
      var dataDocChat = element.data();
      var dataDocChatId = element.id;
      dataListChats.add(ChatUser(
        chatId: dataDocChatId,
        connection: dataDocChat["connection"],
        lastTime: dataDocChat["lastTime"],
        total_unread: dataDocChat["total_unread"],
      ));
    });
    user.update((user) {
      user!.chats = dataListChats;
    });
  } else {
    user.update((user) {
      user!.chats = [];
    });
  }

  chat_id = chatDataId;

  user.refresh();
}
```

Figure 5.34. If already have connections in chat collection

```
else {
  // buat baru , mereka berdua benar2 belum ada koneksi
  final newChatDoc = await chats.add({
    "connections": [
      _currentUser!.email,
      friendEmail,
    ],
  });

  await chats.doc(newChatDoc.id).collection("chat");

  await users
      .doc(_currentUser!.email)
      .collection("chats")
      .doc(newChatDoc.id)
      .set({
    "connection": friendEmail,
    "lastTime": date,
    "total_unread": 0,
  });

  final listChats =
  await
users.doc(_currentUser!.email).collection("chats").get();
```

Figure 5.35. Make new connection in Chat collections

```
final updateStatusChat = await chats
    .doc(chat_id)
    .collection("chat")
    .where("isRead", isEqualTo: false)
    .where("penerima", isEqualTo: _currentUser!.email)
    .get();

updateStatusChat.docs.forEach((element) async {
  await chats
      .doc(chat_id)
      .collection("chat")
      .doc(element.id)
      .update({"isRead": true});
});

await users
    .doc(_currentUser!.email)
    .collection("chats")
    .doc(chat_id)
    .update({"total_unread": 0});
```

Figure 5.36 UpdateStatus read or unRead in controller

In the figure 5.35. there is function make a connection from one user to another user. The user who wants to make connection to the doctor will take the doctor email, then make the collection in the database called “chats”, in this collection will show the id and the name of connection the user and the doctor, after that, the user able to start consultation to the doctor, because the connection already made and doing well.

```
void newChat(String email, Map<String, dynamic> argument, String chat) async {
  if (chat != "") {
    CollectionReference chats = firestore.collection("chats");
    CollectionReference users = firestore.collection("users");
   String date = DateTime.now().toIso8601String();
    await chats.doc(argument["chat_id"]).collection("chat").add({
      "pengirim": email,
      "penerima": argument["friendEmail"],
      "msg": chat,
      "time": date,
      "isRead": false,
      "groupTime": DateFormat.yMMMMd('en_US').format(DateTime.parse(date)),
    });
   Timer(
      Duration.zero,
      () => scrollC.jumpTo(scrollC.position.maxScrollExtent),
    );
    chatC.clear();
  await users
        .doc(email)
        .collection("chats")
        .doc(argument["chat_id"])
        .update({
      "lastTime": date,
    });
    final checkChatsFriend = await users
        .doc(argument["friendEmail"])
        .collection("chats")
        .doc(argument["chat_id"])
        .get();

    if (checkChatsFriend.exists) {
      final checkTotalUnread = await chats
          .doc(argument["chat_id"])
          .collection("chat")
          .where("isRead", isEqualTo: false)
          .where("pengirim", isEqualTo: email)
          .get();
      total_unread = checkTotalUnread.docs.length;
 await users
          .doc(argument["friendEmail"])
          .collection("chats")
          .doc(argument["chat_id"])
          .update({"lastTime": date, "total_unread": total_unread});
    } else {
      await users
          .doc(argument["friendEmail"])
          .collection("chats")
          .doc(argument["chat_id"])
          .set({
        "connection": email,
        "lastTime": date,
        "total_unread": 1,
      });
     }
```

Figure 5.37. newChat function in controllers

In figure 5.37. the user starts the chat and then the system will make new data in firebase according the sender, receiver, and the time. In this function sender and receiver will have same “Chats” collection with same ID. The chat collection will have the message from sender to receiver so that message will be show up it the chat room.

After finish consultation the doctor can give the track record to the user, only doctor can be giving the track record to the user, the user just can see and read the track record.

```
if(docTrack.docs.length != 0){
  final checkConnection = await users
      .doc(_currentUser!.email)
      .collection("trackrecord")
      .where("connection", isEqualTo: friendEmail)
      .get();

  if(checkConnection.docs.length != 0){
    //already have connection
    flagNewConnection  = false;
    track_id =  checkConnection.docs[0].id;
  }else{
    //make new connection
    flagNewConnection = true;
  }
}else{
  //never have connection, make new connection
  flagNewConnection =  true;
}

if(flagNewConnection){
  final trackDocs =  await track.where(
      "connections",
      whereIn: [
        [ _currentUser!.email,
          friendEmail],
        [
          friendEmail,
          _currentUser!.email
        ]

      ]).get();

  if(trackDocs.docs.length != 0){
    //already have connection and track
    final trackDataId = trackDocs.docs[0].id;
    final trackData = trackDocs.docs[0].data() as Map<String,
dynamic>;

    await users
        .doc(_currentUser!.email)
        .collection('trackrecord')
        .doc(trackDataId)
        .set({
      "connection":friendEmail,
      "lastTime": trackData["lastTime"],
    });
```

Figure 5.38. Check tarckRecord in trackRecord Collection

```
if(listTrack.docs.length != 0){
    List<TrackUser> dataListTrack = List<TrackUser>.empty();
    listTrack.docs.forEach((element) {
      var dataDocTrack = element.data();
      var dataDocTrackId = element.id;
      dataListTrack.add(TrackUser(
        trackId: dataDocTrackId,
        connection: dataDocTrack["connection"],
        lastTime: dataDocTrack["lastTime"],
      ));


    });

    user.update((user) {
      user!.track = dataListTrack;
    });

  }else{
    user.update((user) {
      user!.track = [];
    });
  }

  track_id = trackDataId;
  user.refresh();

}
```

Figure 5.39. if already have connection of trackRecord in trackRecord Collection

In figure 5.38 and figure 5.39 the code explains about the trackRecord function, first the system will check if there is any connection already build up by the user to doctor, if not the connection will made automatically and set up on the database.

```
{   final newTrackDoc = await track.add({
      "connections":[
        _currentUser!.email,
        friendEmail,
      ]
    });

    await track.doc(newTrackDoc.id).collection("trackrecord");
    await users
        .doc(_currentUser!.email)
        .collection('trackrecord')
        .doc(newTrackDoc.id)
        .set({"connection" : friendEmail,
      "lastTime": date
    });
    final listTrack = await
users.doc(_currentUser!.email).collection("trackrecord").get();

    if(listTrack.docs.length != 0 ){
      List<TrackUser> dataListTrack = List<TrackUser>.empty();
      listTrack.docs.forEach((element) {
        var dataDocTrack = element.data();
        var dataDocTrackId = element.id;
        dataListTrack.add(TrackUser(
          trackId: dataDocTrackId,
          connection: dataDocTrack["connection"],
          lastTime: dataDocTrack["lastTime"],
        ));
      });
      user.update((user) {
        user!.track = dataListTrack;
      });
    }else{
      user.update((user) {
        user!.chats = [];
      });
    }
    track_id =  newTrackDoc.id;
    user.refresh();

  }
}
print(track_id);
```

Figure 5.40 Build trackRecord in the database

5.2.3. Models

In this application there are 2 models the users models and the chats models, The models contains variables that have a relationship with the user or chat, so that when user or chat data is updated, there is no need to bother writing all the variables to be entered.

```
UsersModel usersModelFromJson(String str) =>
    UsersModel.fromJson(json.decode(str));

String usersModelToJson(UsersModel data) => json.encode(data.toJson());

class UsersModel {
  UsersModel({
    user data
  });

user data

  factory UsersModel.fromJson(Map<String, dynamic> json) => UsersModel(
        user data In json
      );

  Map<String, dynamic> toJson() => {
user data toJson
      };

  UsersModel.fromMap(DocumentSnapshot data){
   user data in document snapshot
  }
class ChatUser {
  chat user data
  });

  chat user data

  factory ChatUser.fromJson(Map<String, dynamic> json) => ChatUser(
       chat user data from json
      );

  Map<String, dynamic> toJson() => {
      chat user data to json
      };
}

class TrackUser {
  TrackUser({
    track user data
  });

  track user data


  factory TrackUser.fromJson(Map<String, dynamic> json) => TrackUser(
    track user data from json
  );

  Map<String, dynamic> toJson() => {
    track user data toJson
  };
}
```

Figure 5.41. Users Models

Figure 5.41 there is user models. To make developer easy input data to database, so the author just calls the user models function so the data will be set automatically and decrease the error of typing when code the program.

```
Chats chatsFromJson(String str) =>
Chats.fromJson(json.decode(str));

String chatsToJson(Chats data) => json.encode(data.toJson());

class Chats {
  Chats({
chat data
  });

chat data

  factory Chats.fromJson(Map<String, dynamic> json) => Chats(
        connections:
List<String>.from(json["connections"].map((x) => x)),
        chat: List<Chat>.from(json["chat"].map((x) =>
Chat.fromJson(x))),
      );

  Map<String, dynamic> toJson() => {
        "connections": List<dynamic>.from(connections!.map((x)
=> x)),
        "chat": List<dynamic>.from(chat!.map((x) =>
x.toJson())),
      };
}

class Chat {
  Chat({
chat data
  });

  chat data

  factory Chat.fromJson(Map<String, dynamic> json) => Chat(
chat data from json
      );

  Map<String, dynamic> toJson() => {
chat data toJson
      };
}
```

Figure 5.42. Chat Models

The chat models will use to make chat construction when make data chat in the database.

CHAPTER VI

SYSTEM DEVELOPMENT

1.7. Testing Environment

Testing environment is an explanation of the device that will be used on the mobile application testing, there are the things that used for testing:

1.1.1. Mobile Application

The mobile application test with 2 ways, the first is using emulator and then using mobile phone, the testing device have a specification as follow:

1. Samsung A50S
   a. Processor Exynos 9611
   b. RAM 4GB
   c. Android 11
2. Pixel 3a (emulator)
   a. Android 11

1.1.2. Laptop

The laptop used to monitor the database, if the database already inputted or not

1. Laptop MSI
   a. Processor i5-gen 10
   b. OS windows

1.8. Testing Result

This final project using prioritizes Blackbox testing to focusing in the application system functionality.

1.1.3. Black Box Testing Scenario

This method is useful for viewing and reviewing the usability of the application and also the course of the input or output process in the application.

Table 6.1. Black Box Testing

No	Features	Scenario	Expected Result	Result
1	Login	User press login button in the login screen	User successfully login to the application and go to the home page	As Expected,
		Doctor press login button in the login screen	Doctor successfully login to the application and go to the home page	As Expected,
2	Search Doctor	User press search button in the home page, after that user go the category page and choose the category after that choose the doctor and directly go to the	User can find the doctor and start the consultation	As Expected,

		consultation room or chat room		
3	Consultation	User chat the doctor and then the doctor reply the user	Users can send a message to the doctor and then the doctor can read the message and so on	As Expected,
4	Track Record	After finishing doing consulting, the doctor writes the track record on the track record page, and then the user can read the track record on the track record page.	A doctor can write the track record to the user and the user can read the track record.	As Expected,
5	Change Profile	User and doctor change their name and their status in the profile screen.	The user and doctor can change their name and status after that show pop up message success.	As Expected,
6	Change Profile Picture	User and doctor change their profile	The user and doctor can change their	As Expected,

		picture in the profile screen	profile picture in the profile screen	
7	Search Nearby doctor	User can search nearby doctor, hospital and drugstore in the application	The user can find the nearby doctor in the application	As Expected,
8	Change role to be a doctor	User can register to be a doctor, if user input their certificate and fill the role options.	The user can change their role in the application by uploading the certificate and fill the role in the application	As Expected,

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1. Conclusion

This final project aims to provide online consulting services using a mobile application to make it easier for people to get consultations instantly and can be carried out anywhere, this application is also free so users don't have to worry about the consultation fees that will be incurred.

The framework used is flutter which provides a lot of convenience in making this application, because flutter has a lot of libraries that can be used in the application creation process. In this application it takes about 16 libraries to finish the system needs to making this application.

7.2. Future Work and Suggestion

for the future work and future development in this final project, there are some features to improve this final project and suggestion:

1. User who wants access this application must using smartphone
2. Add doctors, so the number of medical personnel in this application can increase
3. Add new features to the user and to the doctor so the application can grow up such as video call or call.

REFERENCES

[1] Boyce, Tammy, Brown, Chris.(2019). economic and SOCIAL impacts and benefits of health systems

[2] Islam, Md. Rashedul.(2010). Mobile Application and Its Global Impact

[3] Wu, Wenhao.(2018). React Native vs Flutter, cross-platform mobile application frameworks

[4] Khawas, Chunnu.(2018). Application of Firebase in Android App Development-A Study.

[5] Khasanah, R. L., Kesuma, C., & Wijianto, R. (2018). Sistem Informasi Pelayanan Kesehatan Online Berbasis Web Pada PMI Kabupaten Purbalingga, 1(1), 74-75.

[6] Syukron, Akhmad. and Noor Hasan. 2015. Perancangan Sistem Informasi Rawat Jalan Berasis Web Pada Puskesmas Winog. ISSN: 2338-9761. Yogyakarta: Bianglala Informatika Vol. 3, No. 1, Maret 2015: 28–34. Diambil dari: http://ejournal.bsi.ac.id/jurnal/index.php/Bianglala/article/view/574/465.

[7] Setiyawati, Erwin dan Sardiarinto. 2016. Perancangan Sistem Informasi Berbasis Web Studi Kasus: KSU BMT Al-Ikhwan Yogyakarta. Yogyakarta: Indonesian Journal on Computer and Information Technology Vol. 1, No. 1 Mei 2016: 34–41. Diambil dari: http://ejournal.bsi.ac.id/jurnal/index.php/ijcit/ article/view/417/317.

[8] Levey, Samuel and Loomba, Paul, 1973, Health Care Administration: "A Managerial Prespectiv". Dalam: Azwar, Azrul, 1996, Pengantar Ilmu Kesehatan Masyarakat, Jakarta: FKUI

[9] Taufik, Andi dan Ermawati. 2017. Perancangan Sistem Informasi Pemesanan Pentas Seni Berbasis Web Pada Sanggar Seni Getar Pakuan Bogor. ISSN: 2461-0690. Jakarta: IJSE –

Indonesian Journal on Software Engineering Vol. 3, No. 2: 1–7. Diambil dari: http://ejournal.bsi.ac.id/jurnal/index.php/ijs e/article/view/2812/1836.

[10] Jati. K.D.N(2017).Rancang Bangun Aplikasi Konsultasi Kesehatan Online, 1(1), 2