

HOSPITAL APPOINTMENT SYSTEM

BY

JONATHAN AARON JALLEH

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

HOSPITAL APPOINTMENT SYSTEM

BY

JONATHAN AARON JALLEH

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

REPORT STATUS DECLARATION FORM

Title: Hospital Appointment System

Academic Session: 1/2022

I JONATHAN AARON JALLEH
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

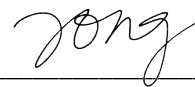
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

127, Jalan Meranti,
4, Taman Meranti,
09000, Kulim, Kedah

Dr Tong Dong Ling
Supervisor's name

Date: 19 April 2022

Date: 20 April 2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 19 APRIL 2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that JONATHAN AARON JALLEH (ID No: 18ACB01377) has completed this final year project entitled “Hospital Appointment System” under the supervision of Dr Tong Dong Ling (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.


Yours truly,



(Jonathan Aaron Jalleh)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**HOSPITAL APPOINTMENT SYSTEM**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : Jonathan Aaron Jalleh

Date : 19 April 2022

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr Tong Dong Ling who has given me this bright opportunity to engage in this hospital appointment system project. A million thanks to you.

I would also like to thanks to my friends for their unconditional support. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project is about developing a mobile based hospital appointment system for patients using the Flutter SDK. The motivation for this project is to create a mobile application for hospital appointment system to solve current appointment system problems in healthcare environment such as time-consuming patient appointment scheduling, troublesome patient registration, long waiting time and frequent patient no-shows. Patients currently are required to physically fill in registration form and must wait for their turn for an appointment with a doctor. In regard to these problems, the proposed application will allow patients to make hospital appointment easily through their mobile phone and check in swiftly by scanning their QR code to check in, which will contribute to increased patient satisfaction and reduced outpatient waiting time. Besides that, real time queue feature will allow patient to plan their arrival and appointment reminder feature will prevent patient no-shows. A chat feature will be implemented to allow patient to enquire any information regarding the appointment or illness via the application. Agile methodology, which is an incremental and mobile application development approach will be used for the development of this project. The proposed application will contain a front-end mobile application for the patient to schedule appointments and a backend web application for hospital administrators.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER 1: INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	4
1.2.1 Sub-Objectives	4
1.3 Project Scope	5
1.4 Contribution	7
1.5 Report Organization	8
CHAPTER 2: LITERATURE REVIEW	9
2.1 Traditional Appointment Scheduling	9
2.2. Online Appointment System	11
2.3 Mobile-based Appointment System	15
2.4 Summarized Table of Comparison	19
CHAPTER 3: SYSTEM DESIGN	20
3.1 System Framework	20
3.2 Use Case Diagram	21
3.3 Use Case Explanation	22
3.3 Activity Diagram	28

CHAPTER 4: SYSTEM METHODOLOGY/APPROACH	36
4.1 Design Specifications	36
4.1.1 Methodologies and General Work Procedures	36
4.1.2 Tools to use	39
4.1.3 User Requirements	41
CHAPTER 5: SYSTEM IMPLEMENTATION	46
5.1 User Interface (Mobile Application)	47
5.1.1 Sign in and Registration	47
5.1.2 Homepage	48
5.1.3 Make Appointment	50
5.1.4 Manage appointments	53
5.1.5 Appointment Details	55
5.1.6 Set appointment reminder	56
5.1.7 Manage reminders	57
5.1.8 User profile	58
5.1.9 Appointment history	60
5.1.10 Chat Enquiries	61
5.2 Admin Interface (Web Application)	62
5.2.1 Admin sign in	62
5.2.2 Homepage	63
5.2.3 Profile page	65
5.2.4 Add Appointments	67
5.2.5 Doctors List	68
5.2.6 Appointment List	69
5.2.7 Add Patients	71
5.2.8 Patients List	72
5.2.9 Chat Inquiries	74
5.3 Coding Explanation (User Mobile Application)	76
5.3.1 Create Account	76
5.3.2 Sign In	78
5.3.3 Homepage	79
5.3.4 Make Appointment	80
5.3.5 Manage Appointment	83
5.3.6 Appointment Details	86

5.3.7 Appointment Reminders	87
5.4.8 User Profile	90
5.4 Coding Explanation (Admin Web App)	92
5.4.1 Sign in	92
5.4.2 Add Appointments	93
5.4.3 Appointment List	95
5.4.4 Chat Enquiries	98
CHAPTER 6: SYSTEM TESTING	100
6.1.1 User and Admin Authentication	100
6.1.2 User Appointment Module	102
6.1.3 User Reminder Module	105
6.1.4 User Profile Module	106
6.1.5 Admin Module (Doctor)	107
6.1.6 Admin Module (Medical Receptionist)	109
CHAPTER 7: CONCLUSION AND RECOMMENDATION	111
7.1 Project Review and Discussion	111
7.2 Future Recommendation	112
REFERENCES	113
WEEKLY LOG	117
POSTER	122
PLAGIARISM CHECK RESULT	123
FYP2 CHECKLIST	127

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.2.1	Appointment scheduling function	11
Figure 2.2.2	Flowchart of outpatient registration method	12
Figure 2.2.3	Flowchart for online appointment registration	13
Figure 2.2.4	Flowchart for web application	14
Figure 2.3.1	Location-based hospital searching	15
Figure 2.3.2	Doctor availability feature	15
Figure 2.3.3	Block diagram for Mr Doc Application	16
Figure 2.3.4	Context Diagram of Medical Appointment Application	17
Figure 3.1.1	System framework	20
Figure 3.2.1	Use case diagram	21
Figure 3.3.1	Activity diagram (Register account)	28
Figure 3.3.2	Activity diagram (Sign in into system)	29
Figure 3.3.3	Activity diagram (Manage user profile)	29
Figure 3.3.4	Activity diagram (Make appointment)	30
Figure 3.3.5	Activity diagram (Cancel appointment)	31
Figure 3.3.6	Activity diagram (Set appointment reminders)	32
Figure 3.3.7	Activity diagram (Appointment check in)	32
Figure 3.3.8	Activity diagram (Manage patient records)	33
Figure 3.3.9	Activity diagram (View doctor schedule)	34
Figure 3.3.10	Activity diagram (Manage patient appointment)	35
Figure 4.1.1	Agile Methodology	36
Figure 4.2.1	Gantt Chart for Project 1	44
Figure 4.2.2	Gantt Chart for Project 2	25
Figure 5.0	Application Icon	46
Figure 5.1.1	Sign in page	47
Figure 5.1.2	Registration page	47
Figure 5.1.3	Application homepage	48
Figure 5.1.4	Category list	49

Figure 5.1.5	Appointment list	49
Figure 5.1.6	Doctor list page	50
Figure 5.1.7	Scheduled appointment	51
Figure 5.1.8	Same day appointment	52
Figure 5.1.9	Successfully booked page	52
Figure 5.1.10	Slide action on appointment	53
Figure 5.1.11	Cancel appointment	53
Figure 5.1.12	Reschedule appointment	54
Figure 5.1.13	Appointment details page	55
Figure 5.1.14	Set appointment reminder	56
Figure 5.1.15	Reminder list page	57
Figure 5.1.16	Remove reminder	57
Figure 5.1.17	Appointment reminder	57
Figure 5.1.18	Menu drawer	58
Figure 5.1.19	Account page	58
Figure 5.1.20	Edit profile page	59
Figure 5.1.21	Appointment history page	60
Figure 5.1.22	Appointment history details	60
Figure 5.1.23	Chat Enquiries page	61
Figure 5.2.1	Admin login	62
Figure 5.2.2	Homepage (medical receptionist)	63
Figure 5.2.3	Homepage (doctor)	63
Figure 5.2.4	Profile (Doctor)	65
Figure 5.2.5	Profile (Medical Receptionist)	66
Figure 5.2.6	Add appointments	67
Figure 5.2.7	Doctors list	68
Figure 5.2.8	Doctor's appointment list	69
Figure 5.2.9	Doctor's patient appointment list	70
Figure 5.2.10	Doctor's patient appointment details	70
Figure 5.2.11	Add patients	71
Figure 5.2.12	Patient list	72
Figure 5.2.13	Patient details	73
Figure 5.2.14	Chat list	74

Figure 5.2.15	Patient chat	75
Figure 5.3.1	Create Account (1)	76
Figure 5.3.2	Create Account (2)	77
Figure 5.3.3	Sign In (1)	78
Figure 5.3.4	Sign In (2)	78
Figure 5.3.5	Retrieve patient reminder data	79
Figure 5.3.6	Retrieve patient appointment data	79
Figure 5.3.7	Get doctor busy time	80
Figure 5.3.8	Check doctor available time	80
Figure 5.3.9	Book Appointment (1)	81
Figure 5.3.10	Book Appointment (2)	81
Figure 5.3.11	View current queue	82
Figure 5.3.12	Slidable actions	83
Figure 5.3.13	Update appointment (1)	84
Figure 5.3.14	Update appointment (2)	84
Figure 5.3.15	Cancel appointment (1)	85
Figure 5.3.16	Cancel appointment (2)	85
Figure 5.3.17	Retrieving appointment data	86
Figure 5.3.18	Creating QR information	86
Figure 5.3.19	Displaying QR	86
Figure 5.3.20	Set appointment reminder (1)	87
Figure 5.3.21	Set appointment reminder (2)	88
Figure 5.3.22	Set appointment reminder (3)	88
Figure 5.3.23	Delete appointment reminder	89
Figure 5.3.24	Edit profile (1)	90
Figure 5.3.25	Edit profile (2)	90
Figure 5.3.26	Edit profile (3)	91
Figure 5.3.27	Retrieve appointment history data	91
Figure 5.4.1	Sign In (1)	92
Figure 5.4.2	Sign In (2)	92
Figure 5.4.3	Add appointment (1)	93
Figure 5.4.4	Add appointment (2)	94
Figure 5.4.5	Doctor's appointment list	95

Figure 5.4.6	Retrieve Patient Appointment Data	96
Figure 5.4.7	Patient appointments details	97
Figure 5.4.8	Chats Enquiry List	98
Figure 5.4.9	Retrieve message data	98
Figure 5.4.10	Update message data	99

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Summary of traditional appointment scheduling	10
Table 2.2	Summary of the reviewed mobile-based appointment system	19
Table 3.3	Register Account explanation	22
Table 3.4	Sign in into system explanation	22
Table 3.5	View user profile explanation	22
Table 3.6	Update user profile explanation	23
Table 3.7	Make appointment explanation	23
Table 3.8	Update appointment explanation	24
Table 3.9	Cancel appointment explanation	24
Table 3.10	Set appointment reminders explanation	25
Table 3.11	Manage reminders explanation	25
Table 3.12	Use QR code to check in explanation	25
Table 3.13	Manage patient records explanation	26
Table 3.14	Update doctor schedule explanation	26
Table 3.15	Add patient explanation	26
Table 3.16	Set up follow up appointment explanation	27
Table 3.17	Update appointment details explanation	27
Table 4.1	Laptop Specification	39
Table 4.2	Phone Specification	39
Table 6.1	User and admin authentication test cases	100
Table 6.2	Appointment module test cases	102
Table 6.3	User reminder module test case	105
Table 6.4	User profile module test case	106
Table 6.5	Admin doctor module test case	107
Table 6.6	Admin medical receptionist module test case	109

LIST OF ABBREVIATIONS

<i>QR</i>	Quick Response
<i>SDK</i>	Software Development Kit
<i>IDE</i>	Integrated Development Environment
<i>UI</i>	User Interface
<i>UX</i>	User Experience

CHAPTER 1: INTRODUCTION

1.1 Problem Statement and Motivation

The motivation behind this project is to create a mobile application for hospital appointment system which makes scheduling medical appointments with doctor simple and dependable for users. This project is significant to patients as it will drastically reduce their waiting time in the hospital and increase patient satisfaction. Moreover, patient can manage their health more efficiently.

I. Patient registration and appointment scheduling is time consuming

Although some variant of appointment scheduling and management system is being used by majority of hospitals in Malaysia, the current patient registration and appointment scheduling is extremely time consuming and tedious, as the average waiting time is approximately one to two hours in hospital outpatient departments [1]. Patients are required to physically attend to hospitals to fill up hospital registration forms and schedule appointments with doctors. Moreover, this is not a simple and short task, it takes up much of a patient's time as they need to wait for a given scheduled date by the hospital after registration. To alleviate the problems, different appointment services were introduced.

The current appointment scheduling system whereby business practices employ a hospital personnel to manually record appointment will lead to patient wastage of time on the telephone while waiting to receive assistance. This process is repeated just to set up an appointment [2]. This manual appointment scheduling is prone to human errors, causing inefficiencies to the hospital and putting additional workload on hospital staffs. Besides that, finding a registered user past details and records will be time consuming.

Additionally, some hospitals use online appointment services as alternatives where patients can make an appointment via their hospital websites [3]. However, patients are still required to go to the reception counter to let them check with the hospital appointment slip card before being sent to the waiting room. This causes increased outpatient waiting time, leading to negative effects to delivery of services and experience of clinic by the patient. The factors of this issue are due to deficiency of hospital staff and long registration process. In addition, long waiting time leads to further patient disappointment and effecting patient health [4].

II. Long waiting time

On the other hand, even after making an appointment, patients are required to wait in line for their turn. For example, a patient's scheduled appointment may be at 8.00 a.m., but the real appointment during the agreed date may take a longer time before it is the patient's turn. A study showed that 53 percent of patients were registered within 15 minutes and the average total waiting time from registration to consulting a doctor was 41 minutes [1]. They stated that the reason for this delayed time may be due to outpatient staff are unable to locate the patients' medical record. The prime problem addressed is how to overcome and improve the current manual hospital appointment system. Patients are dissatisfied and disappointed with the duration of time spent between patient arrival and actual starting time of scheduled doctor's appointment.

Besides that, another factor of long waiting time could be due to appointed doctor has a longer consultation time with previous patient with chronic illness or with different visit purpose such as clinical examination [5]. Nonetheless, someone who is unwell does not enjoy the company of other people and want the process to quickly be over. Moreover, patients may get restless while waiting for their turn as they do not know how long they require to queue. During this Covid-19 pandemic, patients are visiting hospitals daily, and people waiting for their scheduled appointments will be uneasy as they are feared of getting infected with the coronavirus [6].

III. Patient no-shows

Patient no-shows are also one of the problems of the current hospital appointment system. Patient no-show refers to when a patient who scheduled an appointment missed or did not appear for the scheduled appointment, while not giving prior notice to the hospital staff [7]. A major reason for patient no-shows is patient perception that current hospital system does not respect their time and beliefs, which is caused by long indirect and direct waiting times of patient [8]. Indirect waiting time is the difference between the appointment request time and actual appointment time of patient, while direct waiting time is the difference between patient's appointment time and the time service is actually received.

Furthermore, forgetfulness is one of the primary reason patients miss their scheduled appointments. [9] stated that the highest percentage, which is 41.8 percent of patients missed scheduled appointments due to forgetfulness. The cause of this can be due to a

CHAPTER 1: INTRODUCTION

majority of patient relied on their memory or other methods such as appointment cards to remember their appointments. Another reason for this forgetfulness is no reminder call by hospital staffs. Missed appointments can be detrimental for patient health as they may not receive mandatory and timely health services, delaying provision of treatment and follow-ups.

Another issue leading to patients being delayed for appointment is due to inability or require long time to find a parking spot in the hospital. Kementerian Kesihatan Malaysia recorded that the total annual visit for outpatient attendances in government hospital all around Malaysia totalled up to 20,721,556 visits. The current parking facilities may be not sufficient for the rising number of patients. A report by [10] stated that approximately two-thirds of drivers felt stressed when finding a parking spot, and nearly 42 percent of them said they missed an appointment owing to this.

CHAPTER 1: INTRODUCTION

1.2 Objectives

The main objective of the proposed project is to develop a mobile application for hospital appointment system and to benefit patients by reducing their waiting time, increasing patient satisfaction and allowing efficient management of health.

1.2.1 Sub-Objectives

1. To add an appointment reminder feature to alert and notify patient of upcoming appointments and sudden appointment changes to prevent missed appointments.
2. To add a current patient queue for patient to see patient queue for same day appointment, preventing the patient to attend appointment with many peoples and reduce long outpatient waiting time.
3. To implement QR code for patient's to directly check in during appointment date, reducing need for manual queuing and check in.
4. To add a chat feature to allow patient to ask medical enquiries.

1.3 Project Scope

The scope for this project is to develop a mobile application for hospital appointment system to contribute to patient satisfaction and reduce outpatient waiting time. The mobile application will be able to book appointments for specific doctor on a date and timeslot selected by patients. A web application will also be developed for hospital administrator which are medical receptionist and doctor to view patient details and records and appointment details. In addition, a database will store the patient appointment details and medical record. This system can be used for any hospital setting and will bring convenience to the hospital administrator and patients. The mobile application will be equipped with multiple functionalities, overcoming limitations of current methods.

I. QR code check-in

The proposed application will have a QR code generated for patients to check in during appointment day after making an appointment. Flutter packages will be used to integrate the QR code generator in the application. The QR code will contain patient appointment details such as patient's info and appointment date and time. With this feature, patients can directly check in for the appointment without queuing, reducing waiting time and increasing patient satisfaction.

II. Patient queue

Furthermore, the proposed application will enable non-booked patient to view the current queue for same day appointments. This will prevent patients from waiting and queuing for a long time before their turn by accidentally attending at a time with many patients. Patients can view the patient queue and adjust their own time to leave their home, enabling them to consult with appointed doctor when there are less people.

III. Appointment reminder feature

The proposed application will include an appointment reminder feature to enable an automatic reminder to patients, thus reducing patient no-shows due to forgetting the appointment. In addition, the appointment reminder feature will be a customizable as each patient has their own preferences and may want to be notified on a different date prior to the appointment. Reminders will be automatically sent to patients' devices as notifications to alert them.

IV. Chat Feature

The proposed application will include a chat feature to enable patients to enquire further details regarding their illness or symptoms, thus increasing user friendliness of the app. This prevents them from making an appointment to the wrong specialist, which may cause them to reschedule an appointment. Moreover, they will not need to do their own research regarding their symptoms which may be difficult for them.

1.4 Contribution

The proposed application will be developed using Flutter SDK, allowing the application to be run on either iOS or Android devices and web. The proposed application will contain two parts: (A) the first is a front-end mobile application for the patient to make appointments, and (B) the backend web application for hospital administrators to manage all patient and doctor appointment schedules. The proposed project can enhance and overcome several issues related to traditional appointment scheduling methods such as long waiting time and patient no-shows, **enabling the task of making a hospital appointment to be simple and dependable** for patients.

In addition to making appointment, patient can also update and cancel any made appointment via the mobile application. This **saves patient's time** as it avoids the need to manually call hospital admins to amend the appointment. Moreover, an appointment reminder feature will be implemented at patient's appointments page, allowing him/her to set reminders based on their personal preference. Therefore, **missed appointments and no shows can be reduced** as patient will receive notification on upcoming appointments.

Secondly, a unique QR code will be provided for each appointment to allow patient to directly check-in on the day of the booked appointment. As compared to traditional check-in, this method would provide a contribution by tremendously saving patient's time **by minimizes the need for manual queueing**. Furthermore, patients can see a live queue of number of patients ahead of them for same day appointments, allowing them to **plan ahead and solve long outpatient waiting time issue**.

1.5 Report Organization

This report has 7 chapters, each chapter discusses different information. In Chapter 1, it contains the problem statement and motivation, project objectives and scope. Chapter 2 contains the literature review and there are 3 parts, which are traditional appointment scheduling, online appointment system and mobile-based appointment system. Besides that, there will be a table that shows the summary of the reviewed system. In Chapter 3, it consists of the system design and includes UML diagrams such as use case diagram and activity diagram. Chapter 4 on the other hand includes the methodology used in the project, along with the tools and requirements. Timeline of the project is also included. Chapter 5 is about the implementation of the project, which includes the project screenshot and the code explanation. Chapter 6 is the testing of the project, and unit testing is done on each component. Lastly, Chapter 7 is about the conclusion of the project, and it includes the project review, discussion and future recommendation.

CHAPTER 2: LITERATURE REVIEW

This chapter discusses the review of existing systems and literary work by researchers.

2.1 Traditional Appointment Scheduling

Traditional doctor appointment scheduling has a standard policy for patient to make an appointment in advance or few months ahead of the actual date [11]. But with no patient reminders, there is a high probability that patients will not show up for their appointments, leading to overbooked appointment timeslots and increasing patient waiting time. It is recommended that a patient reminder is vital to reduce patient no-shows.

Another traditional appointment scheduling is using paper-based system for the appointment registration process. It will require patients to complete and submit appointment forms or place identification card or appointment card in a specific box provided, waiting for their names to be called during their turn [3]. The drawback of this system is if patient or medical staff lost their appointment card or the card is replaced by unauthorised patient, causing waiting time of patient to increase.

[12] shows that clinics has switched to an open access system in which patients will share appointment time slots with patients wanting appointment on the same day to address patient no-show problem. The clinic allocates three to four appointment slots for same-day patients. However, this system is not effective when there is a large patient demand for a particular appointment time slot. Patient also do not know which time slot has less patients.

CHAPTER 2: LITERATURE REVIEW

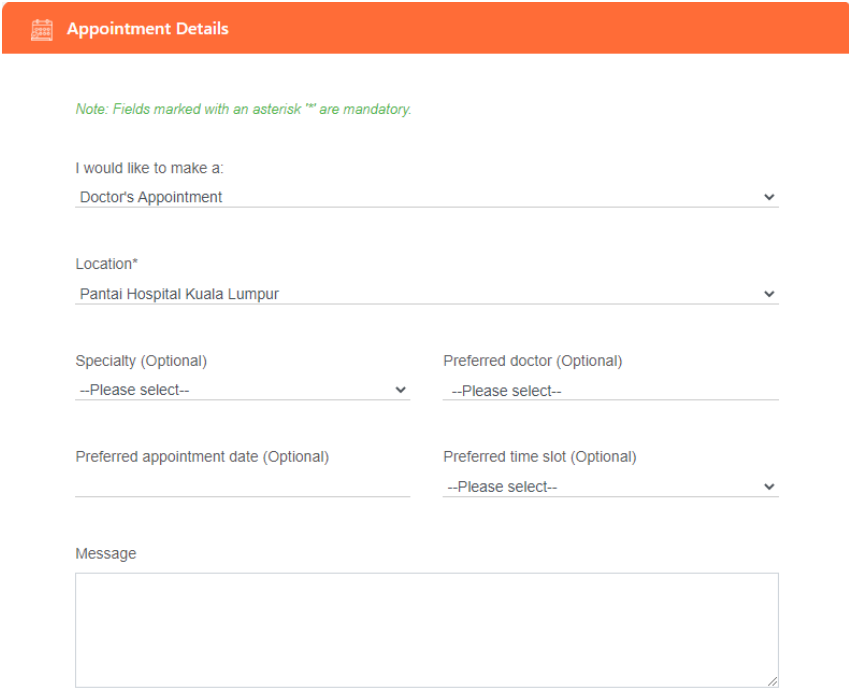
	Traditional Doctor Appointment [11]	Paper-based System [3]	Open Access System [12]
Description	Patient required to make appointment in advance or few months ahead of the actual date	Patient required to complete and submit appointment forms or place identification card or appointment card in boxes provided and wait their turn.	Patient required to share appointment time slots with patients wanting appointment on the same day by allocating extra slots.
Pros	Can plan ahead and make time for the appointment.	-	Can address patient no-show issue
Cons	High probability of patient no-shows as no reminding methods. Causing overbooked appointment timeslots and increased waiting time.	Patient or staff may lose the appointment card or replaced by unauthorised patient. This will cause unnecessary increased waiting time.	Ineffective when there is large patient demand for a particular appointment time slot as patient will not know which time slot is free.

Table 2.1 Summary of traditional appointment scheduling

2.2. Online Appointment System

[3] stated that online appointment registration and scheduling has been an alternative for patients to avoid the inconvenience of queuing and filling out registration forms, which also allows hospital staff to easily monitor and search for patients' medical records or personal information as it is available online. The inefficiency of the system is since it is accessible online, the risk of online hacking of patient personal data is huge. Besides that, there is no reminder system for appointments.

Pantai Holdings Sdn Bhd (PANTAI) introduced an online appointment system which allows patients to make a doctor's appointment via the website [13].



The screenshot shows a web form titled "Appointment Details" with an orange header. Below the header is a note: "Note: Fields marked with an asterisk * are mandatory." The form contains several dropdown menus and a text area:

- "I would like to make a:" dropdown menu with "Doctor's Appointment" selected.
- "Location*" dropdown menu with "Pantai Hospital Kuala Lumpur" selected.
- "Specialty (Optional)" dropdown menu with "--Please select--" selected.
- "Preferred doctor (Optional)" dropdown menu with "--Please select--" selected.
- "Preferred appointment date (Optional)" dropdown menu with "--Please select--" selected.
- "Preferred time slot (Optional)" dropdown menu with "--Please select--" selected.
- A "Message" text area at the bottom.

Figure 2.2.1 Appointment scheduling function

There is a direct registration feature which allows patients to directly register themselves online without having to queue physically at the hospital. The system also has appointment slot scheduling, allowing patients to select preferred appointment type and appointment location, together with preferred date and time slot. Nonetheless, there are inefficiencies found in this system [13]. Firstly, the system does not have the feature to allow view of appointment status, whereby staff need to contact the patient to reschedule appointment. Next the system requires patients' manual check in during the appointed date for the hospital administrator to check the patients' records.

CHAPTER 2: LITERATURE REVIEW

In 2009, all public tertiary hospitals in China has begun the use of web-based appointment system (WAS) to replace the traditional registration method which had tedious waiting times and caused great stress on clinic staff [14].

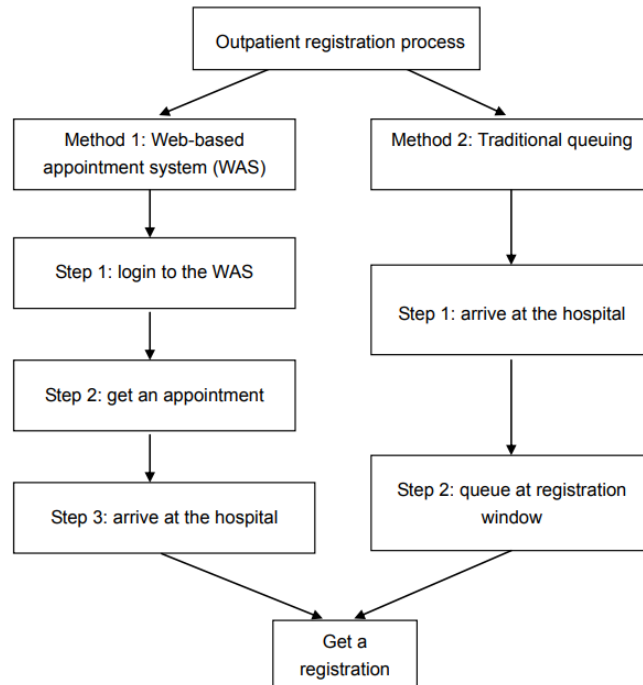


Figure 2.2.2 Flowchart of outpatient registration method

With this web-based appointment system, patients are required to login to the WAS, and schedule an appointment. After successful scheduling, patients will be given an appointment number. During the designated time on the scheduled appointment date, patients will directly get the registration that is allocated to their appointment number, thus removing the need to queue and wait for the registration. This system will significantly decrease total waiting time of patients, especially insignificant queue waiting time. One of the weaknesses of this system is lack of reminder system. It is suggested that the additional use of reminder system should be considered to reduce non-attendance rates of patients [14].

In addition, [2] proposed a web-based appointment system to minimize long patient waiting time and workload for hospital personnel.

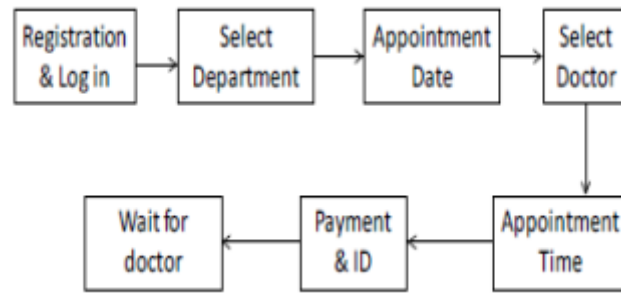


Figure 2.2.3 Flowchart for online appointment registration

The proposed system allows the patient to view appointment dates that are available and non-available and ability to view the doctor's scheduled calendar. Moreover, doctors or administrators are able to edit the appointment slot in the system during any emergency situation. The proposed system enables patient to view the status of the appointment, allowing them to know the best time to book an appointment when there are no pending appointments. This will reduce long waiting time and long queue for patients, while also having the freedom to fix and book an appointment based on their own preference [2]. There are some inefficiencies towards this proposed system, whereby the system does not have a reminder system to notify patients and no live queue system for walk in patients to view.

BruHIMS (Brunei Health Information Management System) is an ICT initiative where the management of patients' personal information in all medical facilities are done through an electronic patient record system and including appointment system and outpatient management system [15]. This system manages patients flow by scheduling appointments beforehand and providing early preparations of healthcare locations which will significantly decrease the wastage of time slots owing to non-availability of either doctors or patients. This is because it validates the available and non-available slots for both parties [3]. However, one of the drawbacks of this system is scheduled patients are required to present their appointment cards to be scanned before appointment to alert the system. Next, the system sets priority level of scheduled patients based on age and status of registration rather than the level of sickness.

Furthermore, another web application was proposed to minimize patient waiting time and allowing simpler way to find specific doctors for diseases [16].

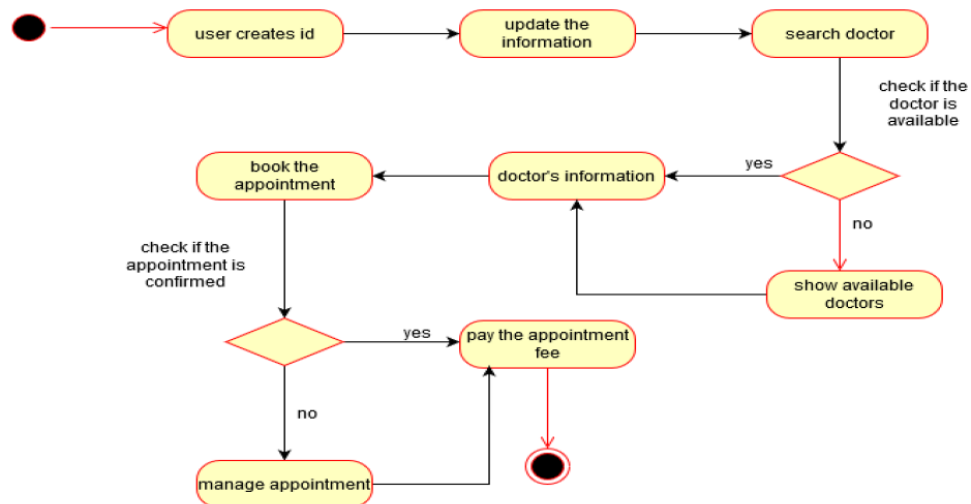


Figure 2.2.4 Flowchart for web application

The proposed work allows patient to search for doctor according to their location, hospital and specification of the doctors. Appointments are made by specifying the date and selecting available doctor. Besides that, patients will be given options to review and rate their doctor after appointments. One of the features proposed is using location services to lookup nearby hospitals, avoiding going far distance for doctor consultation, thus reduces time and money consumption. Besides that, data analytics is used to retrieve patient appointment details to aid in future appointment predictions.

[17] proposed a web-based appointments system to replace traditional queuing method for registration to enhance patient satisfaction and effectively decrease waiting time. For this system, patient registration and sign in is required, then they can book an appointment after selecting a doctor of their choice. However, admin will need to view patient's appointment request with doctor's availability before sending request confirmation. It is stated improvement can be made for a more efficient system and reduce patient no shows by promoting usage of online appointment registration and reminder system.

2.3 Mobile-based Appointment System

A mobile application for patient registration and appointment scheduling is one of the best alternatives as almost everyone has a smartphone. It is stated that approximately 90 percent of the population in Malaysia is using a smartphone in the year 2019 [18]. One of the current mobile appointments and queue system for clinics and hospitals use in Malaysia is Encore Med [19].



Figure 2.3.1 Location-based hospital searching

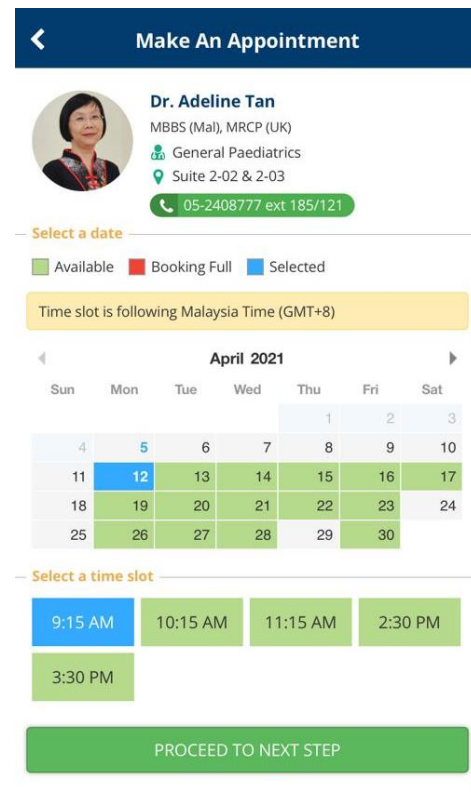


Figure 2.3.2 Doctor availability feature

CHAPTER 2: LITERATURE REVIEW

Patients can easily search for clinics or hospitals based on their preferred location. After selecting a particular hospital or clinic, patients will be able to view the list of doctors and services provided respectively. This application allows patients to check on a particular doctor's available date and time slot to schedule an appointment. After making an appointment successfully, the time slot will be reserved to prevent other bookings. This application has a check live queue feature, allowing patients to check the current queue of patients waiting for their turn. It is only available on actual appointment day and queue sequence along with estimated wait time is provided to patients. However, there are certain drawbacks such as no appointment reminder feature that allows patients to set automatic reminders to remind them of upcoming appointments. The system also still requires patients to manually check-in for hospital or clinic staff to search and access patients' date and medical records which may cause delay to the appointment consultation.

[20] proposed a Doctor Appointment Application System called Mr Doc, which uses an android platform to provide a simple and reliable way of making a doctor's appointment.

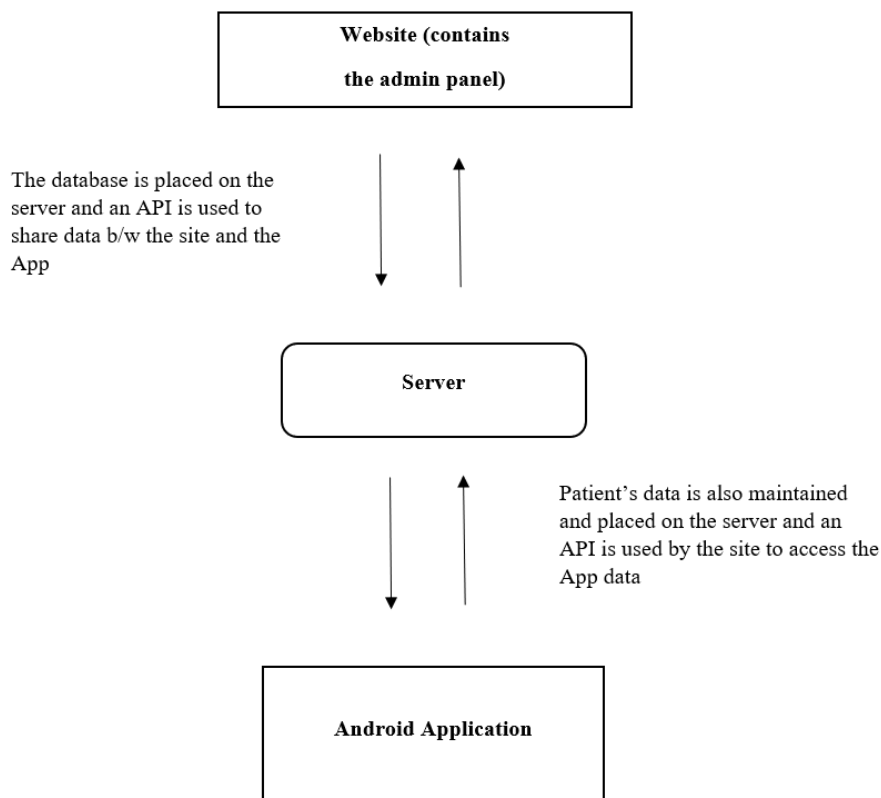


Figure 2.3.3 Block Diagram for Mr Doc Application

CHAPTER 2: LITERATURE REVIEW

Patients has the options to select particular doctor and view the doctor's details. Moreover, the patients can request for their preferred day and time for the appointment. After a successful appointment made, a notification will be sent to the patients and the selected day and time slot will be reserved for them. Next, there will also be a map feature showing the location of the hospital for patients to view. There is also an admin module that is designed on a website for admin to view all doctor and appointment details. [20] also stated various drawbacks that can be improved which includes adding an appointment reminder feature to notify patients of upcoming appointments and saving appointment date to application calendar or device calendar. They also consider adding a small payment amount to be charged to patients or users to prevent fake appointments by unethical users as this will cause wasted appointment time slot.

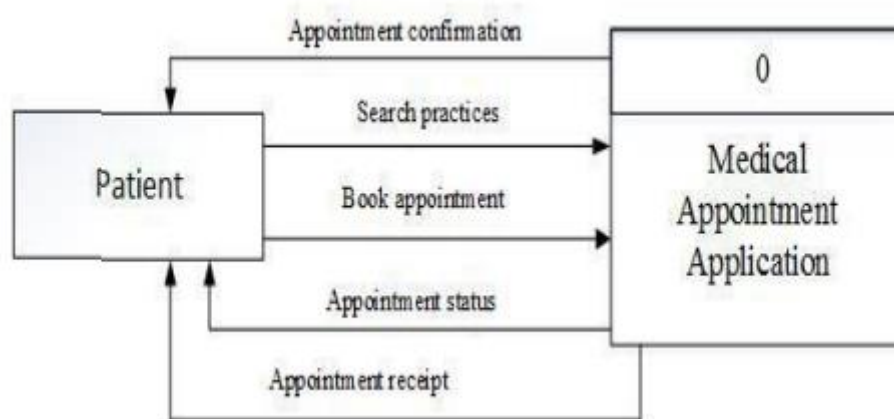


Figure 2.3.4 Context Diagram of Medical Appointment Application

Next, [21] developed a web-based mobile application, Medical Appointment Application, to manage appointment-booking process which will reduce the number of appointment calls and avoid rushing for an emergency appointment. The system will list available medical facilities with respective addresses for users to pick. Next, users will be required to fill in an appointment form which contains patient personal details and appointment details such as appointment time and date. After submitting, patients have an option to print the appointment details as physical proof. However, there are drawbacks to this system, whereby it is unable to display the doctors or clinics unavailable appointment time, patients can only fill in appointment date without knowing which time is available and unavailable [21]. They also propose multiple improvements for the system, such as displaying the available appointment time to prevent patients from accidentally selecting unavailable time. In addition, an email

CHAPTER 2: LITERATURE REVIEW

notification or other reliable notification feature should be added to notify patients for appointment rejection. It is also suggested to make the interface of the mobile application more interesting and creative to improve customer engagement.

On the other hand, [22] developed an application of intelligent agent-based system for hospital appointments negotiation and scheduling using Android 2.2. The system developed provides patients with the ability to select a specific appointment date and time, together with the procedure being scheduled. Moreover, the appointment scheduling procedure is based on priority level of patient and the system will check the appointment schedule of medical personnel and propose the most suitable appointment time based on fuzzy preferences. The system will also send out appointment confirmation. However, the drawback of this proposed system is that it has no automatic reminder system for patient before upcoming appointments [22]. Besides that, non-registered patients are required to make an appointment at least 24 to 48 hours before the actual appointment time. They proposed to develop a system in the future which can direct the appointment to other hospital or clinic that a same doctor works at and a feature to take care of least scheduling of emergency appointments. Another recommendation is to provide an automatic system calls as reminders before upcoming.

Lastly, [23] developed a mobile application for doctor appointment scheduling to provide patient with real time selection of appointments. The application requires patient to select what health problems they are facing from a list of health issues before inputting their current location. Next, a list of selected doctors is displayed to the patient based on their selection of health problems and location, providing the most suitable and relevant choices to the patient, reducing patient's time. It has a feature that will record down number of visitations, medication prescription and days taken if the patient is consulting the same doctor. Moreover, the application also provides a live video consultation with doctors.

2.4 Summarized Table of Comparison

Table 2.2 shows the summary of the reviewed mobile-based appointment system

Features	Encore Med	Mr Doc	Medical Appointment Application	[22]	[23]	Proposed Application
Location based hospital searching	✓		✓		✓	
Selection of preferred time and date	✓	✓	✓	✓	✓	✓
Appointment based on priority level				✓		
Hospital location map	✓	✓				
Doctor availability	✓					✓
List of health problems					✓	
Live patient queue	✓					✓
Customizable reminder feature						✓
QR code check in						✓
Chat feature						✓

Table 2.2 summary of the reviewed mobile-based appointment system

CHAPTER 3: SYSTEM DESIGN

The use case diagram, use case diagram explanation and activity diagram are done in this chapter to show the system overview. Besides that, the development process is briefly explained through the code of the project.

3.1 System Framework

Client server architecture is used for the development of the proposed project because resources are not shared between client and server and can only be accessed by client requesting data to the server through the internet. Hence, it is suited for the proposed project as patient or admin may access the data at any location with only the use of the internet.

According to [24], one of the advantages of this architecture is centralization, where all the resources are placed in a single location, thus easing the work of administrators to update and manage data and resources. Another benefit is accessibility, patients can easily access the system resources regardless of any location or platform. This architecture also offers high scalability, whereby the capacity of clients and servers can be increased easy without critical disturbance.

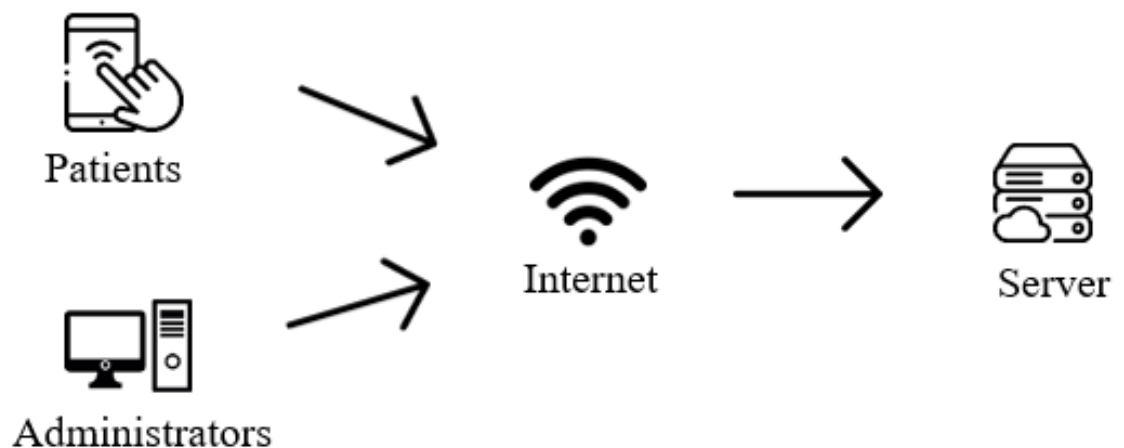


Figure 3.1.1 System framework

3.2 Use Case Diagram

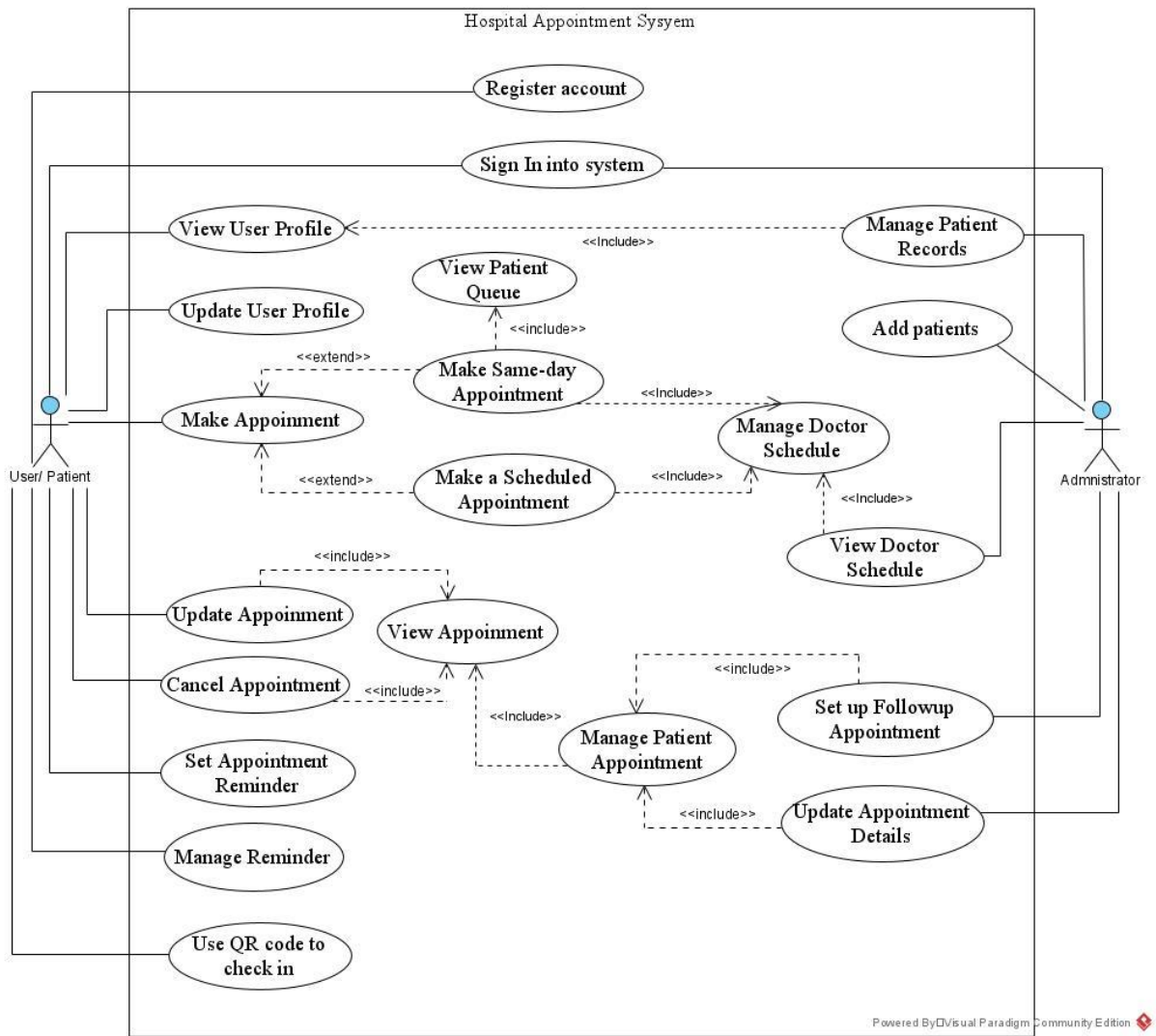


Figure 3.2.1 Use case diagram

3.3 Use Case Explanation

Register Account

Actor	Patient
Basic Flow	The use case describes the process of the patient and administrator register themselves as a user for the system. Patient will register an account via the mobile application, hospital administrator will register account via web application. Credentials needed are full name, I/C number, email address, phone number and password. Then, patient will submit the registration form, and it will be validated.
Alternate Flow	There are invalid credentials entered, patient need enter credentials again.

Table 3.3 Register Account explanation

Sign In into System

Actor	Patient, Administrator
Basic Flow	Patient/administrator will enter valid email and password. After submitting, the credentials will be validated by the system. They will be navigated to respective home page if sign in successful.
Alternate Flow	There are invalid credentials entered, patient/administrator need enter credentials again.

Table 3.4 Sign in into system explanation

View User Profile

Actor	Patient
Basic Flow	Patient navigates to account page, then select edit profile option. The system will display the user credentials stored in the database.

Table 3.5 View user profile explanation

Update User Profile

Actor	Patient
Basic Flow	Patient can select the fields displayed to update any of their credentials. After completed, patient can select a button to update their data to the database.

*Table 3.6 Update user profile explanation***Make Appointment**

Actor	Patient
Relationship	Extend: Make same day appointment, Make scheduled appointment
Basic Flow	System will display a list of categories of specific departments. Patient selects a specific category, navigating to that particular page. System will display a list of doctors in that department. User will have 2 option to choose from: (A) Making a same day appointment, (B) Making a scheduled appointment. The system will display successfully booked page.
Sub Flows	(A) The system will display the make appointment page, with listing the current date and time slot to choose from. Current patient queue will also be included to allow user to view queue of current appointments so patient can choose a suitable timeslot to prevent long waiting time. Patient will need to enter a time slot, then press the book appointment button. (B) The system will display the make appointment page. Patient will need to select a valid date and time slot that is available to the particular doctor. Then, patient need to press the book appointment button.
Alternate Flows	Patient selects an unavailable date; system will display time slot that cannot be selected. Patient will need to reselect an available date.

Table 3.7 Make appointment explanation

Update Appointment

Actor	Patient
Relationship	Include: View Appointment
Basic Flow	Patient navigates to appointment page. System will display the list of made appointments. Patient can select a specific appointment and slide it for more actions. The patient selects the update action. The patient will choose another date and time slot to change for the appointment. After patient confirm the choice to update, system will update the data to the database and display the updated appointment list.
Alternate Flows	Patient cancels the choice to update, system will cancel updated action and display the current appointment list.

*Table 3.8 Update appointment explanation***Cancel Appointment**

Actor	Patient
Relationship	Include: View Appointment
Basic Flow	Patient navigates to appointment page. System will display the list of made appointments. Patient can select a specific appointment and slide it for more actions. The patient selects the delete action. After patient confirm the choice to cancel, system will remove the data to the database. The system will display the updated appointment list.
Alternate Flows	Patient cancels the choice to delete, system will display the current appointment list.

Table 3.9 Cancel appointment explanation

Set Appointment Reminders

Actor	Patient
Basic Flow	Patient navigates to appointment page. System will display the list of made appointments. Patient selects a particular appointment. System displays appointment details page. Patient slides a button to navigate to set reminder page. Patient selects the reminder date and time. The patient presses the complete button. The system will navigate patient to reminder list page.

*Table 3.10 Set appointment reminders explanation***Manage Reminders**

Actor	Patient
Basic Flow	Patient navigates to reminder page. The system will display the list of scheduled reminders. Patient can: (A) cancel reminder (B) remove reminder.
Sub Flow	(A) If patient wants to cancel a scheduled reminder, he/she needs to press the cancel button to cancel the reminder, but the reminder details will still be present. (B) If patient wants to remove a reminder, he/she needs to select the radio button to remove the reminder details. The system will remove the reminder details from the database and display the updated list of reminders.

*Table 3.11 Manage reminders explanation***Use QR code to check in**

Actor	Patient
Basic Flow	Patient navigates to appointment page. System will display the list of made appointments. Patient selects a particular appointment, then the system will display appointment details page. A QR code will be generated containing patient's appointment details. Patient needs to scan the QR code for check in.

Table 3.12 Use QR code to check in explanation

Manage Patient Records

Actor	Administrator
Relationship	Include: View User Profile
Basic Flow	Medical receptionist will navigate to patient records page. System will display patient personal records. Administrator can view and retrieve data of specific patient.

*Table 3.13 Manage patient records explanation***View Doctor Schedule**

Actor	Administrator
Relationship	Include: Manage Doctor Schedule
Basic Flow	Doctor will navigate to their profile page. The page will display doctor's details and their busy timeslots.

*Table 3.14 View doctor schedule explanation***Add patient**

Actor	Administrator
Basic Flow	Medical receptionist will navigate to add patient page. Medical receptionist will need to enter valid patient details. Then, press the submit button to add the patient details to database.
Alternate Flow	Medical receptionists enter invalid details, system will display error message and request to re-enter details.

Table 3.15 Add patient explanation

Set Up Follow-up Appointment

Actor	Administrator
Relationship	Include: Manage Patient Appointment
Basic Flow	Doctor can navigate to add appointment page. They can add a follow up appointment for a specific patient. Doctor will enter follow up appointment date and time. The system will add the appointment to the database and display at patient's mobile application.
Alternate Flow	Doctor enters invalid details; the system will prompt doctor to enter details again.

*Table 3.16 set up follow up appointment explanation***Update Appointment Details**

Actor	Administrator
Relationship	Include: Manage Patient Appointment
Basic Flow	Doctor can navigate to patient appointment records. They can update patient completed appointment details and add any description or remarks. The doctor also can remove the completed appointment from patient appointment list and transfer it to the appointment history.

Table 3.17 Update appointment details explanation

3.3 Activity Diagram

Register Account

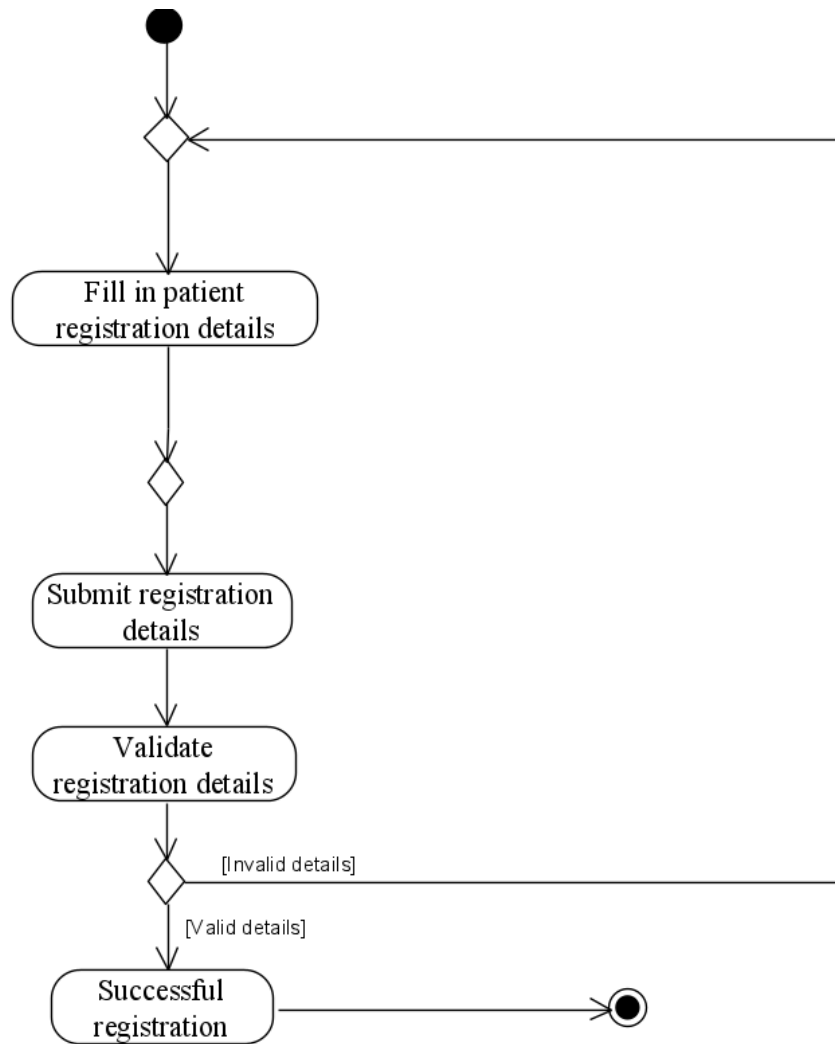


Figure 3.2.3 Activity diagram (Register account)

Sign In into system

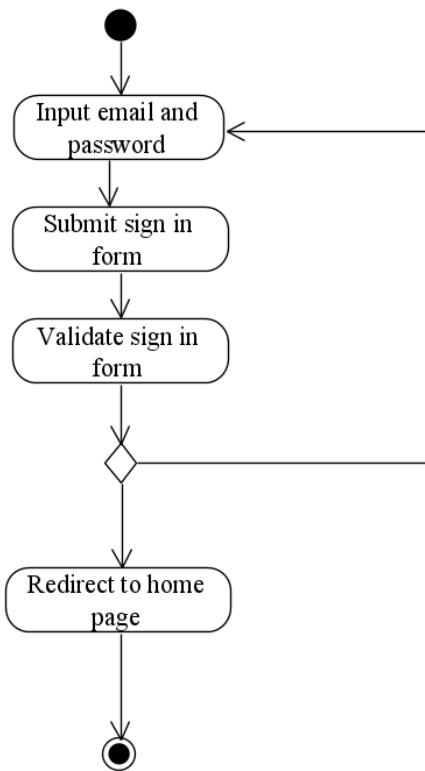


Figure 3.2.4 Activity diagram (Sign in into system)

Manage User Profile

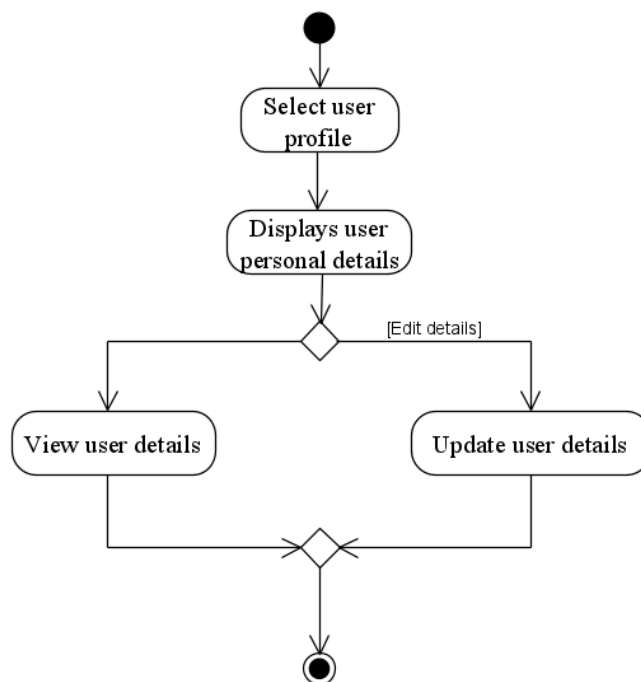


Figure 3.2.5 Activity diagram (Manage user profile)

Make Appointment

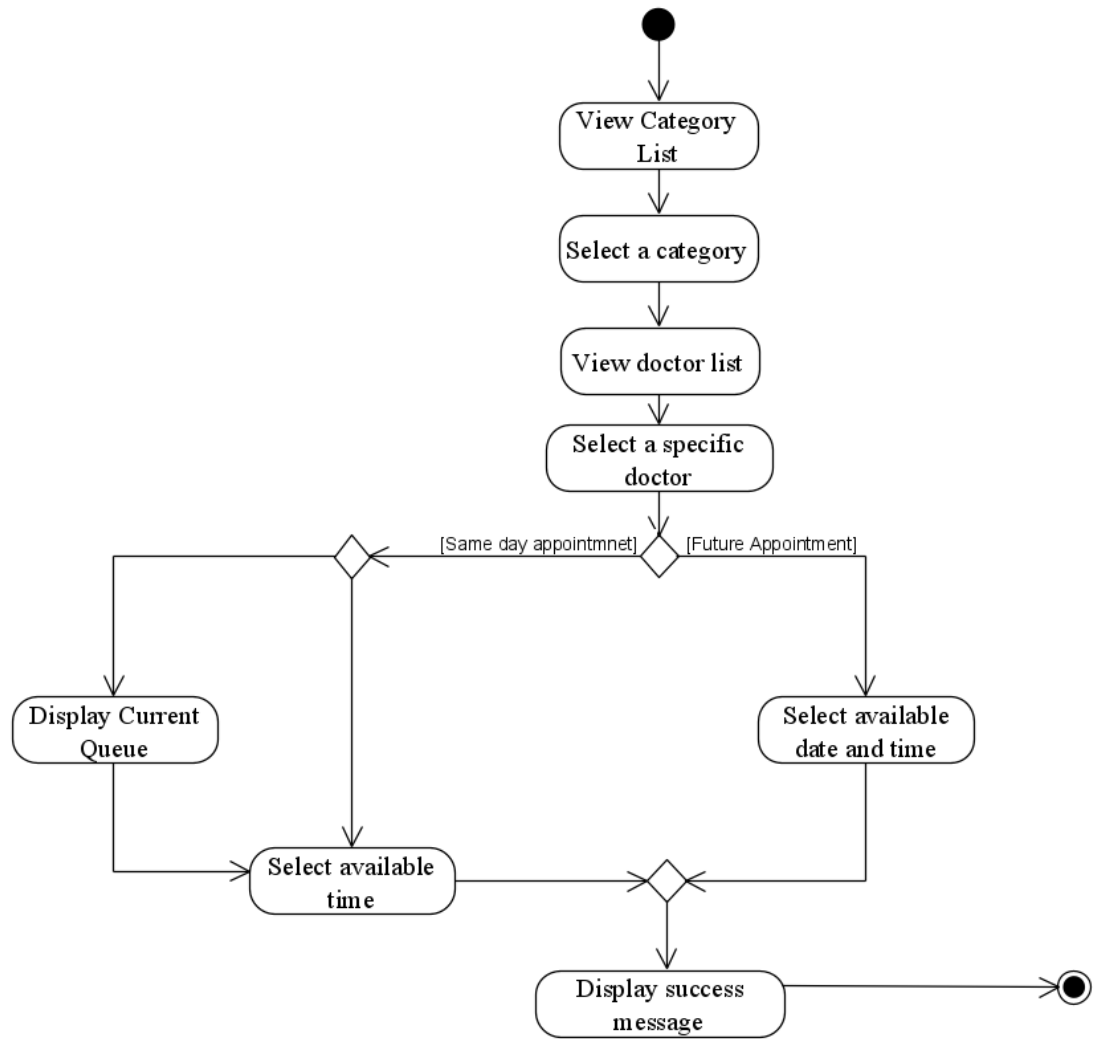


Figure 3.2.6 Activity diagram (Make appointment)

Cancel Appointment

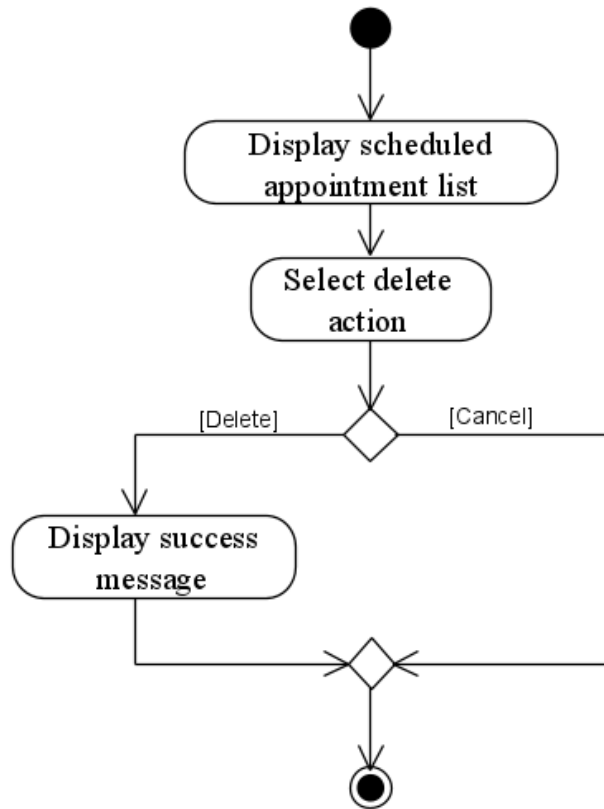


Figure 3.2.7 Activity diagram (Cancel appointment)

Set Appointment Reminder

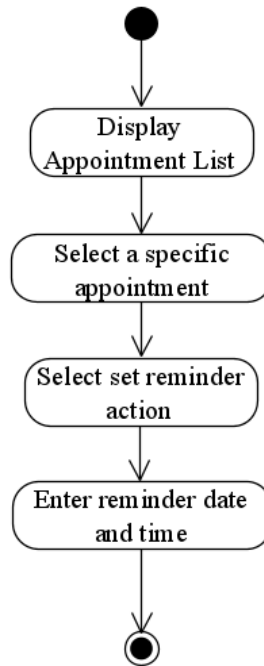


Figure 3.2.8 Activity diagram (Set appointment reminder)

Appointment Check In

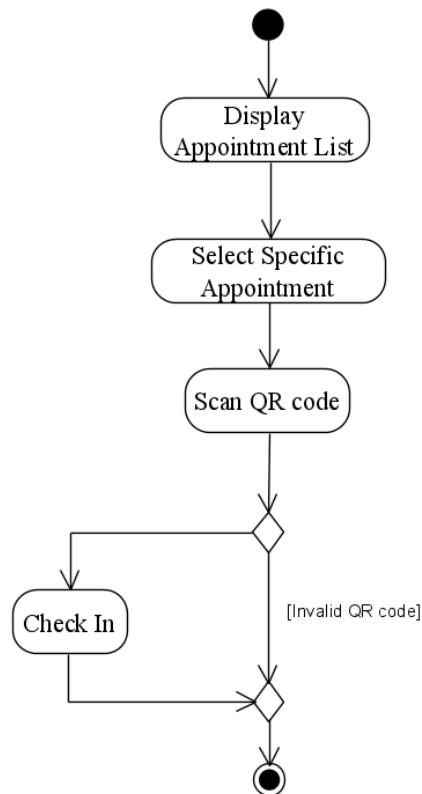


Figure 3.2.9 Activity diagram (Appointment check in)

Manage Patient Records

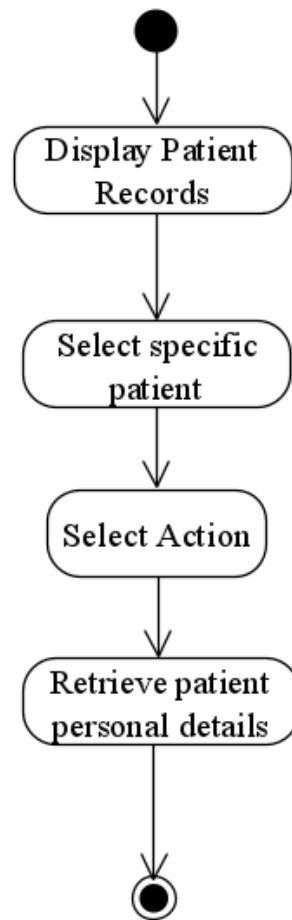


Figure 3.2.10 Activity diagram (Manage patient records)

View Doctor Schedule

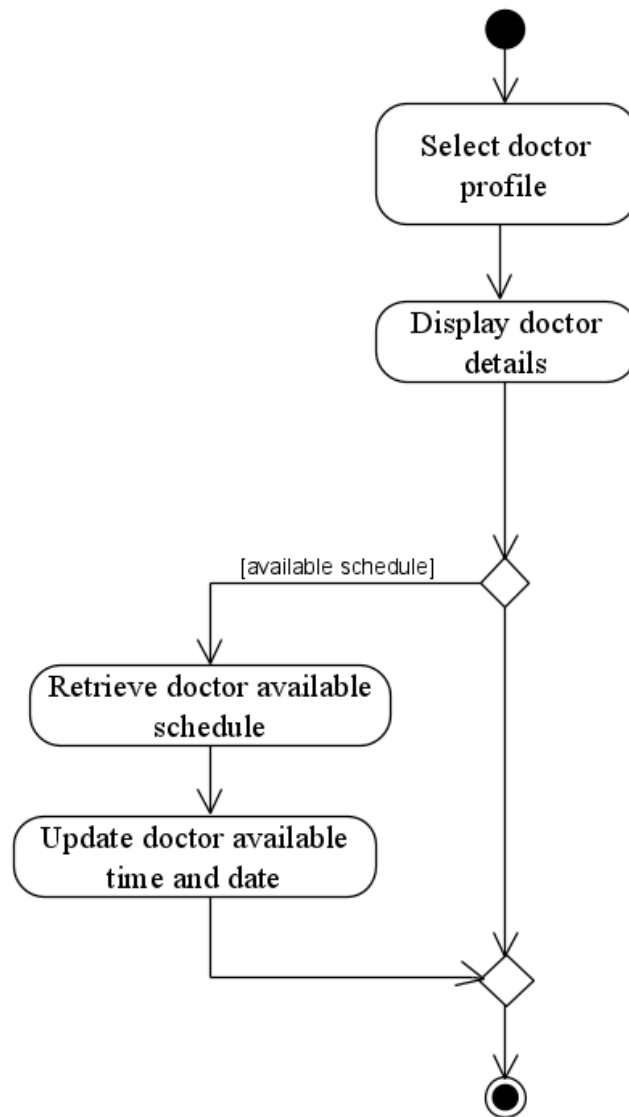


Figure 3.2.11 Activity diagram (View doctor schedule)

Manage Patient Appointment

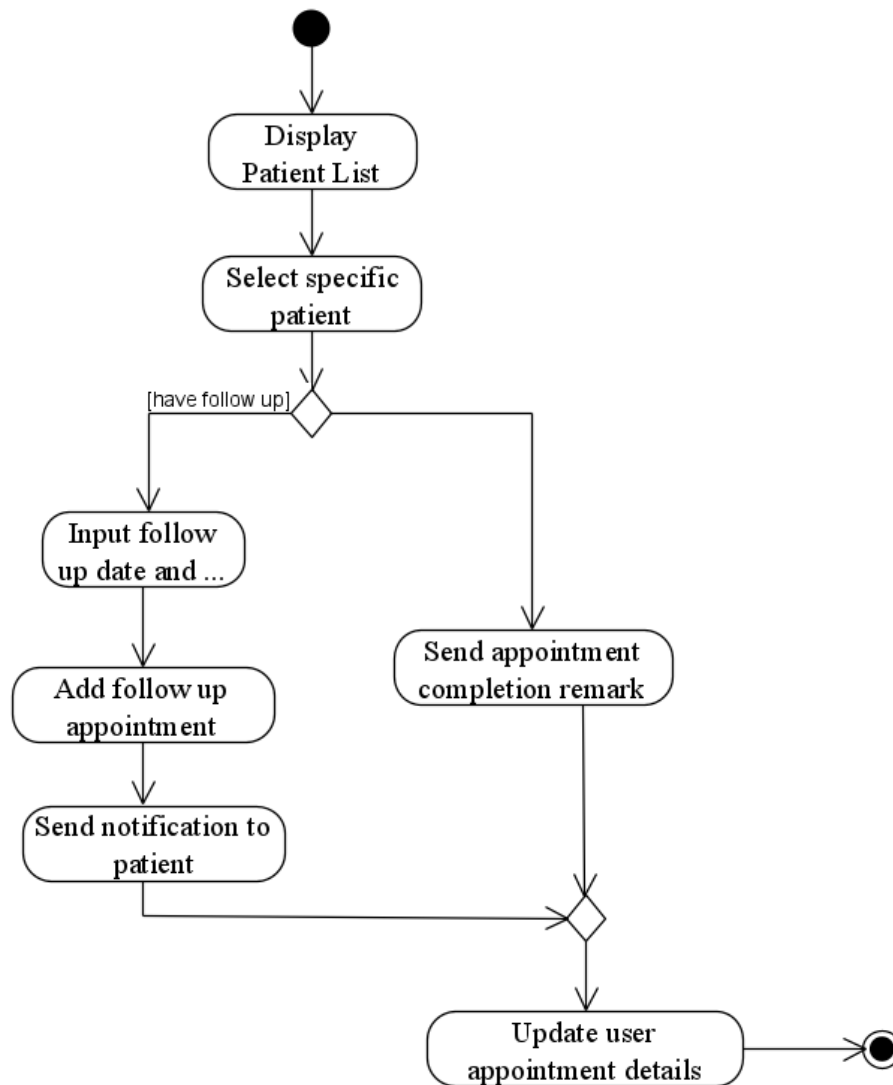


Figure 3.2.12 Activity diagram (Manage patient appointment)

CHAPTER 4: SYSTEM METHODOLOGY/APPROACH

4.1 Design Specifications

4.1.1 Methodologies and General Work Procedures

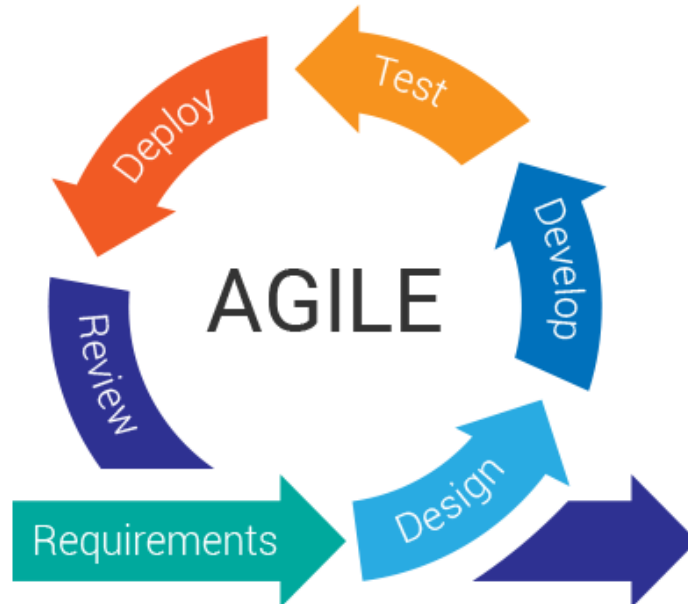


Figure 4.1 Agile Methodology

The methodology that will be involved for this proposed project is Agile methodology, which is an incremental and mobile application development approach. Agile methodology is suitable for this proposed project as it is a small-scale project and mobile applications require more flexibility to cater to continuous changes [25]. This methodology will break the project into several smaller adaptive phases or sub-modules, then delivered in various sprints or iteratively over the development cycle [26]. The incremental and iterative approach will help developer to identify problem with the features earlier in the development process. Agile method will also produce a better-quality mobile application as testing is undergone at every single module rather than at the end of the development phase, which reduces the chances of encountering a large number of bugs at the end of project [27]. This is because testing occurs concurrently with programming process. Adjustment of features can also be often done at each sprint and improve the final product.

Requirement Analysis

As the goal of this project is to develop a mobile-based hospital appointment system, the objectives of the project were detailed which include to benefit patients by reducing their waiting time, increasing patient satisfaction and allowing efficient management of health. Then, the user requirements are specified for the system.

Design

Based on the requirements gathered from the previous stage, tools needed for this hospital appointment system such as Android Studio and Flutter are identified. Moreover, client server architectural design is identified to be used for this project, whereby the client side is the user mobile application, and the server side is the hospital admin web application. Furthermore, UML diagrams that is used to model the project are use case and activity diagrams.

A low fidelity prototype of the mobile application and web application user interface is created to show a rough draft of the proposed app such as the homepage and make appointment page for the project. Further iterations are then done to improve and refine the design and improve user experience (UX) for the hospital appointment system throughout the project duration.

Development

This phase is generally known as coding phase, where the proposed design in the design phase is coded into an actual software using software development tools proposed which are Android Studio and Visual Studio Code, and the framework used is Flutter. The graphical user interface for the user mobile application is coded has been completed in FYP1, while the interface design for web application has been completed in FYP2. Then, backend functionality of the project is further implemented using Firebase to manage the data on the cloud, which is Cloud Firestore. Some examples of the functionality implemented is CRUD (Create, Read, Update, Delete) functions of the hospital appointment. On the other hand, backend such as storing, processing, and retrieving hospital appointment data from the database is coded and implemented.

Testing

Each of the project's functionality and module is tested to ensure they are working smoothly. Testing is done continuously throughout the project on any new functionalities or solution completed. For example, the make appointment functionality is tested in terms of how the patient will interact with it or the transactions that may be performed. The hospital appointment system will also need to be tested to prevent patients from having a slow and difficult application, which may reduce user experience. Moreover, since current mobile devices come in various sizes, testing of the application need to be done on multiple sizes so that it can be work as planned.

Deploy and Review

This phase is to implement and deploy the completed mobile application. After deploying the app, new bugs that arise must be resolved by further debugging and testing. After fixing bugs, further enhancement can be made to the features to suit user satisfaction. The progress is reviewed to ensure it achieves all the requirements. Then, a new iteration occurs, repeating all the previous phases and taking into consideration the new plan proposed to further improve the app. However, these two stages may not be conducted during the duration of the project as the app may not be deployed to Google Play Store or Apple App Store unless required.

4.1.2 Tools to use

Hardware Involved

I. Laptop

	Specification
Model	Asus ROG Strix G531G
Operating System	Windows 10 64-bit
Processor	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
RAM (Memory)	4.0GB DDR4 RAM
Storage	512GB M.2 SSD
GPU	NVIDIA GeForce GTX 1650

Table 4.1 Laptop Specification

II. Mobile Phone

The mobile phone described is used to run the application during development for easier viewing and interaction.

	Specification
Operating System	Android v.10
Processor	Huawei Kirin 990 5G
RAM (Memory)	8.0GB
Storage	128GB

Table 4.2 Phone Specification

Software Involved

- i. **Flutter** is a free and open-source cross-platform UI toolkit which allows creation of mobile application for both iOS and Android operating system by reusing the same code. Flutter uses a programming language called Dart, which uses C and C++ language as its core, giving access to platform-specific SDKs for Android and iOS [28]. It will be used to develop the front-end for both the mobile application and web application user interface.
- ii. **Android studio** is a free and open-source integrated development environment (IDE) to build a mobile app, providing a complete experience for developing Flutter applications. It will be used for this project in developing, testing, and debugging the mobile application for user or patients. Its official programming language is Java, but Flutter can be used by installing the Flutter and Dart plugins.
- iii. **Visual Studio Code** is a free open-source text editor by Microsoft, providing services to make developing and debugging modern web application easily. VS code supports many programming languages, such as Java, C++, and Python [29]. It will be used to develop, test, and debug the web application for hospital administrators.
- iv. **Firebase** is a Backend-as-a-service and provides various services for developers to create mobile applications. There are multiple features provided such as Firebase Authentication for phone numbers or Google and Firebase Hosting for web app hosting. On the other hand, Cloud Firestore is a NoSQL database stored in the cloud which offers great flexibility and scalability for storing data. It is mainly used for mobile, web and server development, keeping data in sync for client and server-side [30]. It will be used for user authentication and data storage for both the mobile and web application.

4.1.3 User Requirements

Functional Requirements

Client

Appointments

- I. Make online appointment via mobile application for specific doctor.
- II. View and select specific doctor's available timeslot for appointment.
- III. View appointment list of booked appointment which contain relevant appointment information.
- IV. Reschedule and cancel made appointments.
- V. View appointment history of previous completed appointments.
- VI. Check in hospital with QR code given during day of appointment.
- VII. View current patient queue for same day appointments.

Chat

- I. Enquire to medical receptionist about patient sickness and symptoms.

Reminders

- I. Set specific appointment reminder which is customizable.
- II. Cancel and remove reminders.

Profile

- I. View user profile information after registration.
- II. Edit and update user profile information.

Admin

- I. View patient records and details via web application.
- II. View and update doctor schedule
- III. View and update patient appointment list and details.
- IV. Cancel patient appointments
- V. Set up any follow up appointment.
- VI. Reply to patient medical enquiries.

Non-functional Requirements

- i. The system must be able to run in a Windows environment.
- ii. The system must be user-friendly and have a user graphical interface that is easily navigable by user of any ages.
- iii. The system must sync all data from database to application in real-time to provide accurate information.
- iv. The system should not break down or crash easily under normal circumstances.
- v. The system should process each request within 10 seconds.

4.2 Timeline

For Project 1, the tasks required to be done are literature review for further understanding of past works and research papers. System design for proposed project is also completed. Next, interface design and implementation of the functionality for mobile application are tasks to be done for Project 1 also, continues with report writing. On the other hand, for Project 2, tasks to be included is interface design and functionality implementation for the web application. Furthermore, testing and debugging, while refining the overall system are tasks to be done. Then, writing the report and compilation will be done. Further details are shown in the Figures 4.2 and 4.3 below.

CHAPTER 3: PROPOSED METHOD/APPROACH

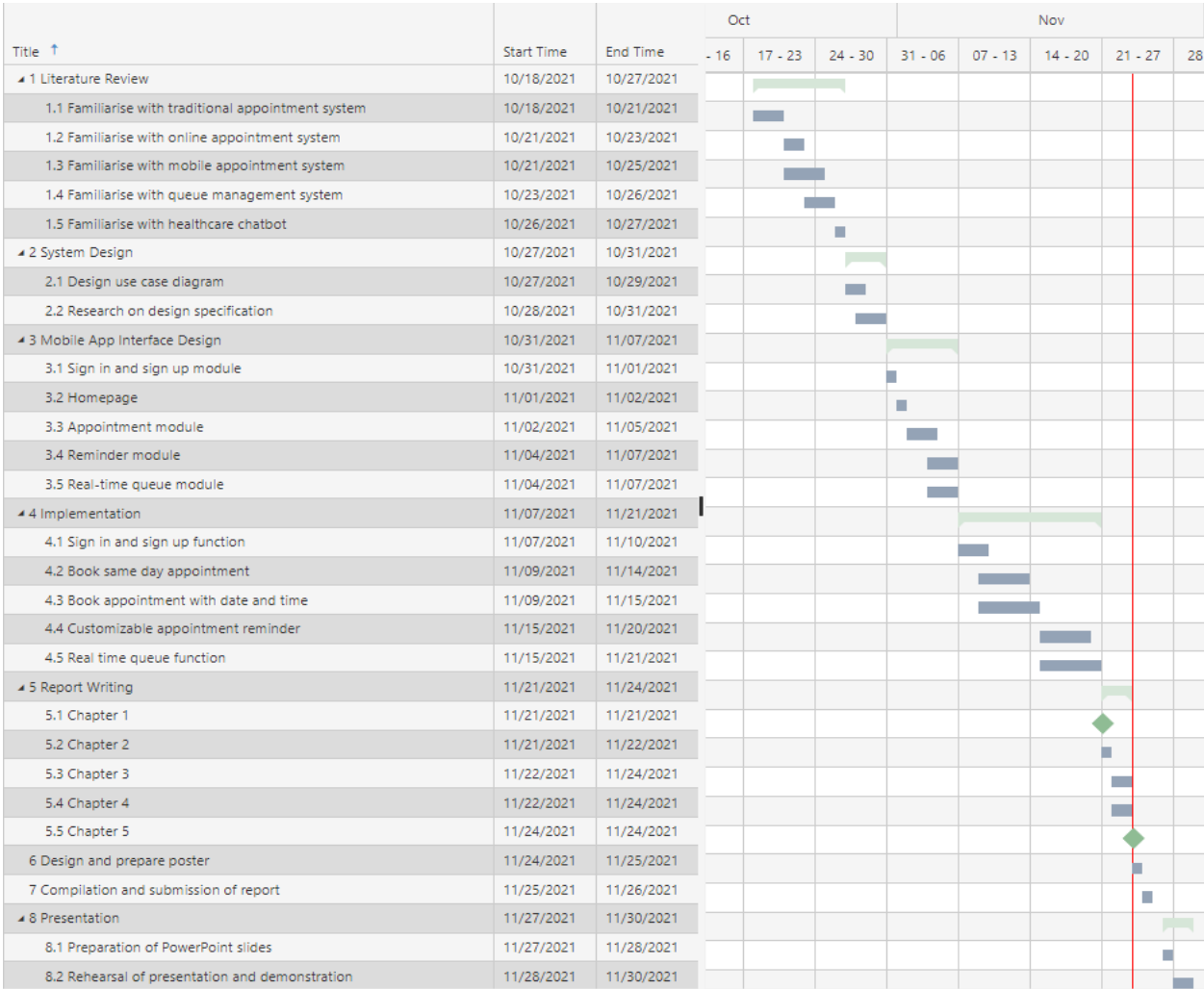


Figure 4.2 Gantt Chart for Project 1

CHAPTER 3: PROPOSED METHOD/APPROACH

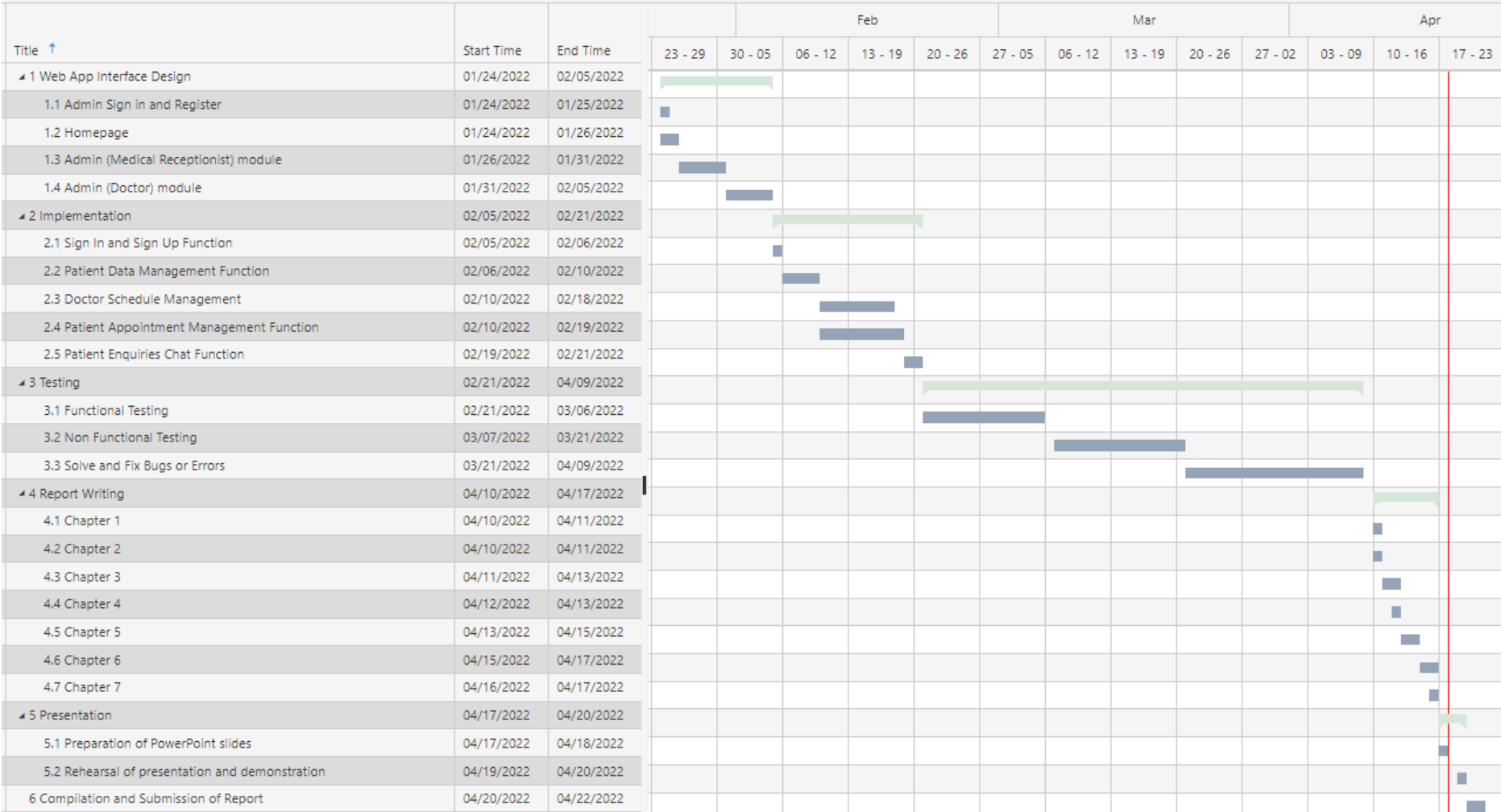


Figure 4.3 Gantt Chart for Project 2

CHAPTER 5: SYSTEM IMPLEMENTATION

This chapter will display the screenshots of the proposed application along with detailed explanation. The screenshots are taken from the laptop used for the project and the mobile application screenshots is taken from the mobile device, HUAWEI P40.

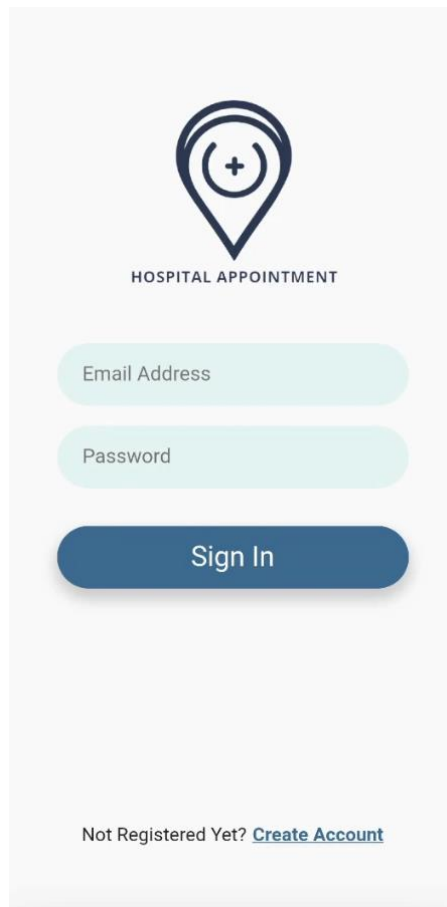
Firstly, the name of the application is “Hospital Appointment”, and the logo is a simple self-crafted one. The main theme used in this application is blue colour, as it is calm colour and is usually associated with professionalism and credibility. Figure 5.1.1 illustrates the icon of the application.



Figure 5.0 Application Icon

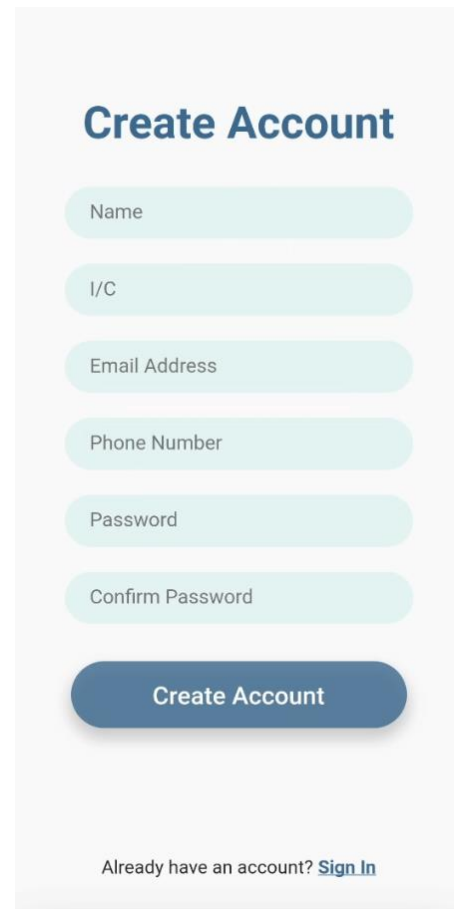
5.1 User Interface (Mobile Application)

5.1.1 Sign in and Registration



The sign-in page features a central logo consisting of a location pin with a plus sign inside, labeled "HOSPITAL APPOINTMENT". Below the logo are two light blue rounded rectangular input fields for "Email Address" and "Password". A dark blue rounded rectangular button labeled "Sign In" is positioned below the input fields. At the bottom of the page, there is a link that says "Not Registered Yet? [Create Account](#)".

Figure 5.1.1 Sign in page



The registration page is titled "Create Account" in a large, bold, dark blue font. It contains five light blue rounded rectangular input fields for "Name", "I/C", "Email Address", "Phone Number", and "Password". A second light blue rounded rectangular input field labeled "Confirm Password" is located below the "Password" field. A dark blue rounded rectangular button labeled "Create Account" is positioned below the "Confirm Password" field. At the bottom of the page, there is a link that says "Already have an account? [Sign In](#)".

Figure 5.1.2 Registration page

The figures above show the sign in and registration page for the patient mobile application. Patient can sign in into the application by entering a valid email and password. The entered credentials are then validated by Firebase Authentication. If patient's credentials are correct, patient will be redirected to the home page; if incorrect, an error message will be displayed. Besides that, if patient has not created an account, they can select "Create Account" and navigate to the registration page. Patient will be required to enter few fields, which will be validated before creating the account.

5.1.2 Homepage

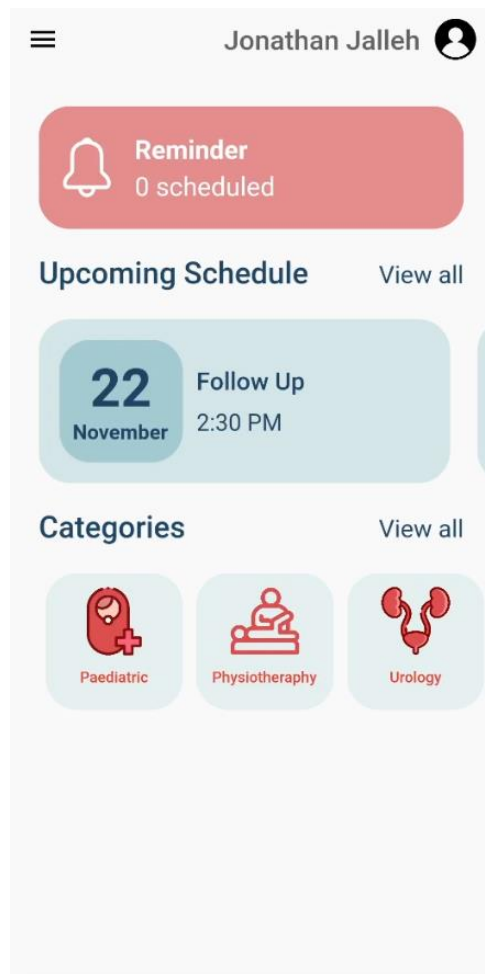


Figure 5.1.3 Application homepage

The figure above shows the homepage of the application. Patient can view scheduled reminders, upcoming appointments, and hospital categories. Patient can select “View all” of the upcoming appointments to view the list of appointments made, or directly select the appointment displayed to view appointment details as shown in Figure 5.1.4. Besides that, patient can select “View all” of the categories to view the list of categories offered as shown in Figure 5.1.5.



Figure 5.1.4 Category list

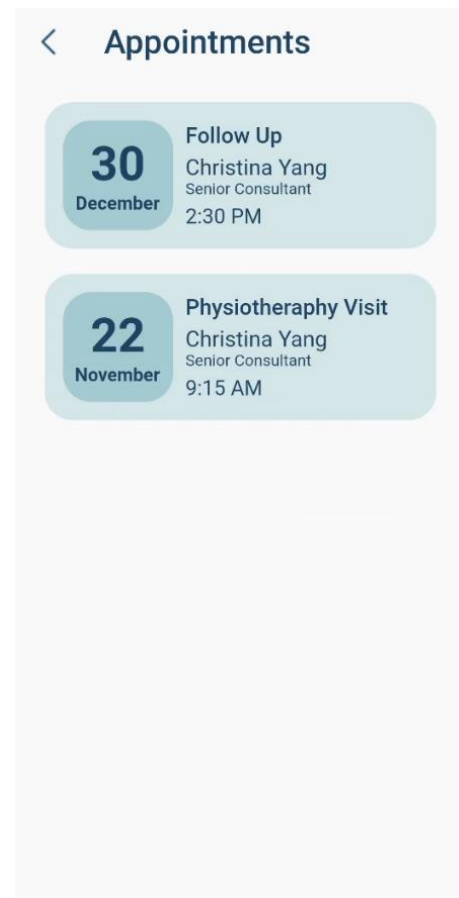


Figure 5.1.5 Appointment list

5.1.3 Make Appointment

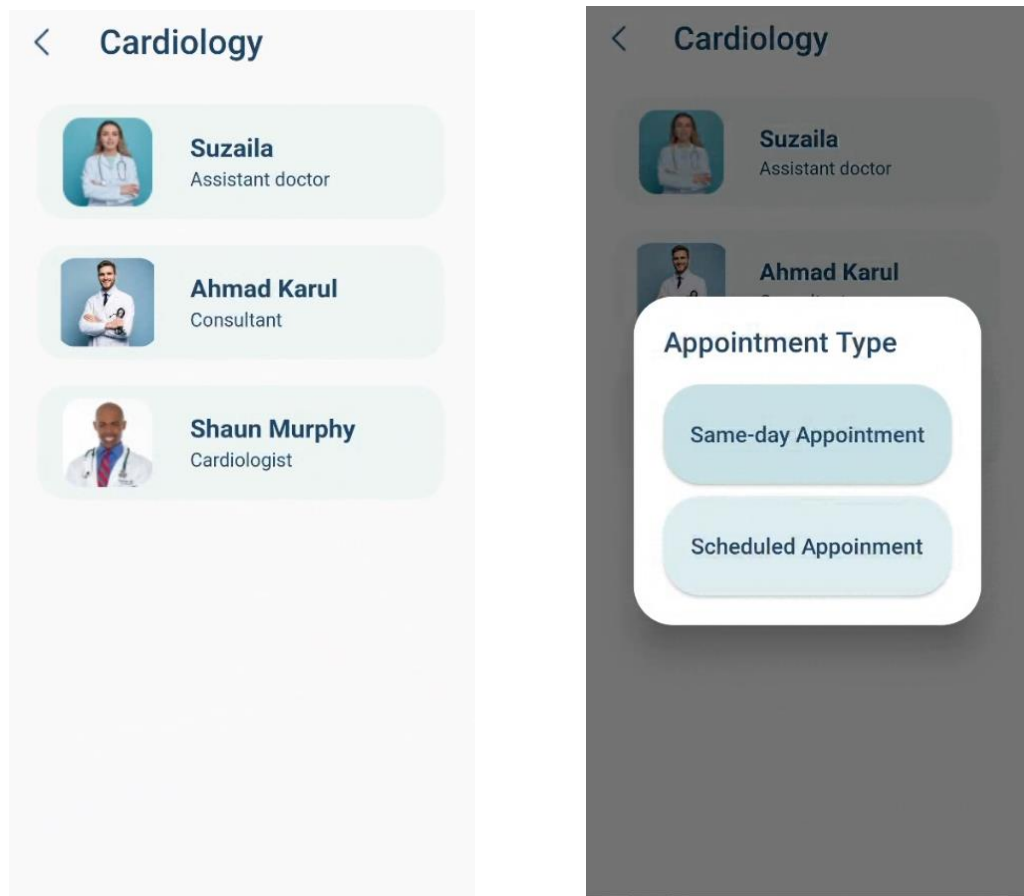


Figure 5.1.6 Doctor list page

Figure 5.1.6 illustrates the list of doctors for a specific category. After selecting a doctor, a dialog will pop up to allow patient to select an appointment type. After selecting an appointment type, it will redirect patient to either scheduled appointment page as shown in Figure 5.1.7 or same day appointment page as shown in Figure 5.1.8. Both have the same functionality except scheduled appointment requires patient to select a date. Same day appointment will display the current queue of patient ahead for the specific doctor. After selecting a date, the application will grey out timeslots which the doctor is unavailable to prevent patient from selecting it. After pressing the book appointment button, a success page will be displayed as shown in Figure 5.1.9 indicating the appointment is successfully booked and a confirmation email will be sent to the patient.

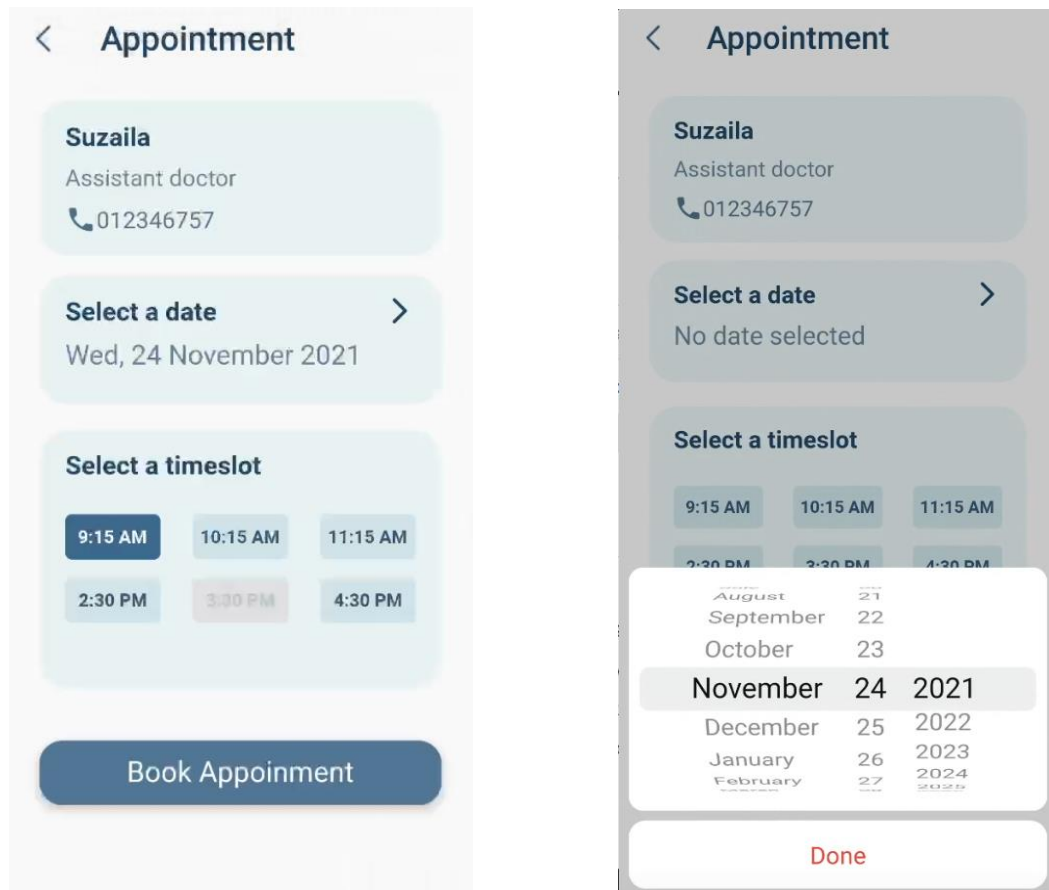


Figure 5.1.7 Scheduled appointment

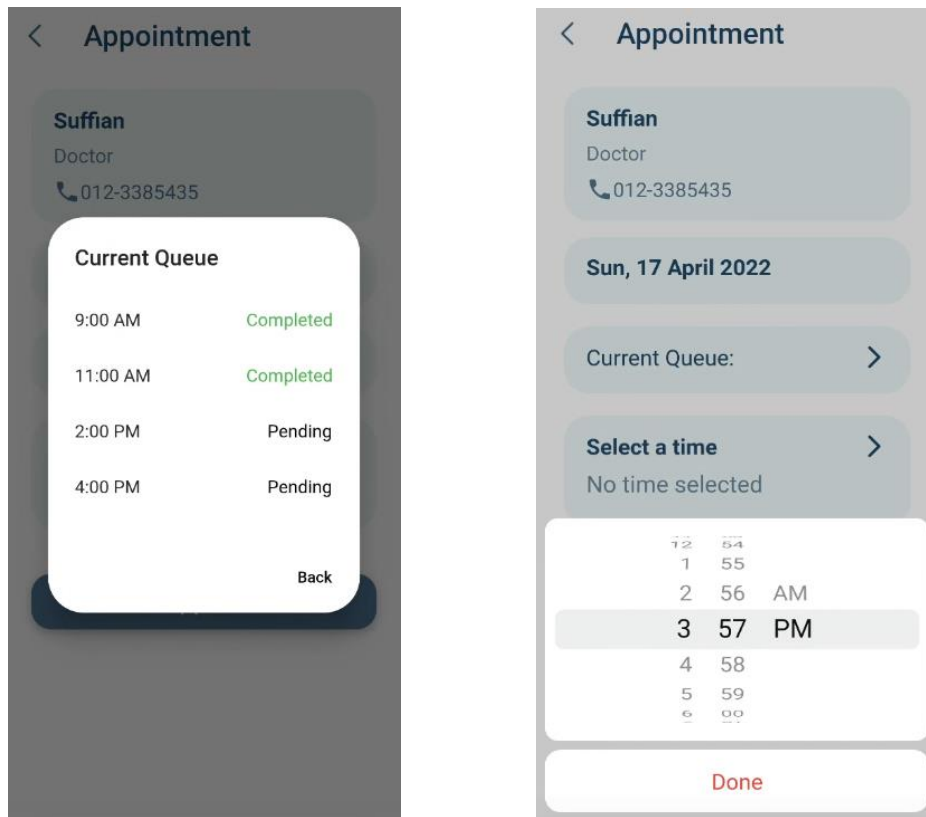


Figure 5.1.8 Same day appointment

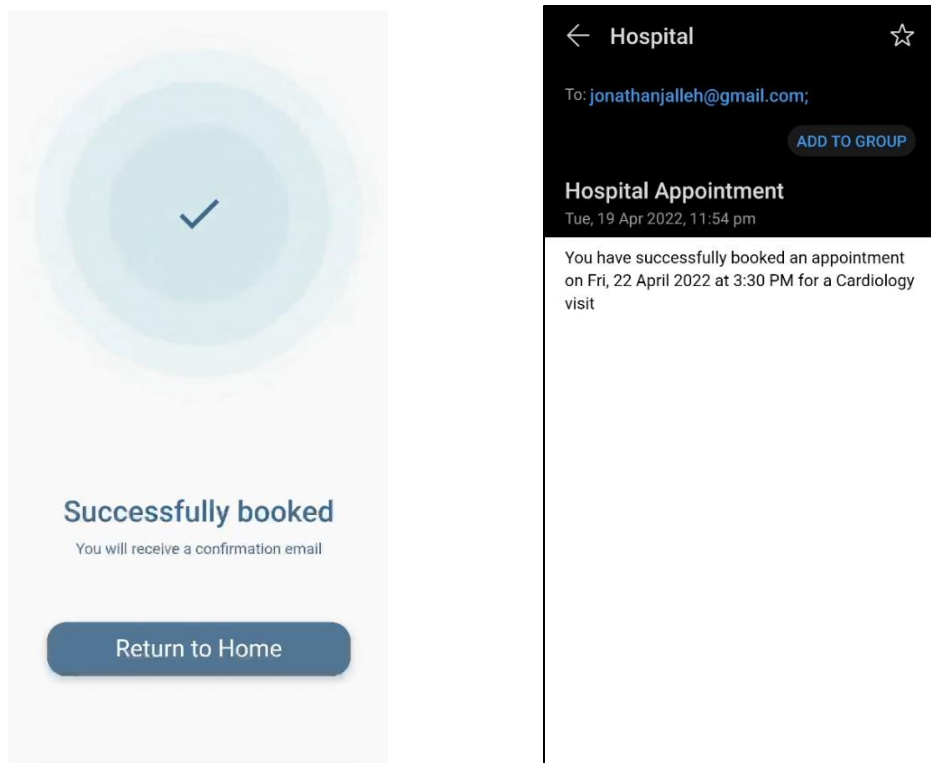


Figure 5.1.9 Successfully booked page

5.1.4 Manage appointments

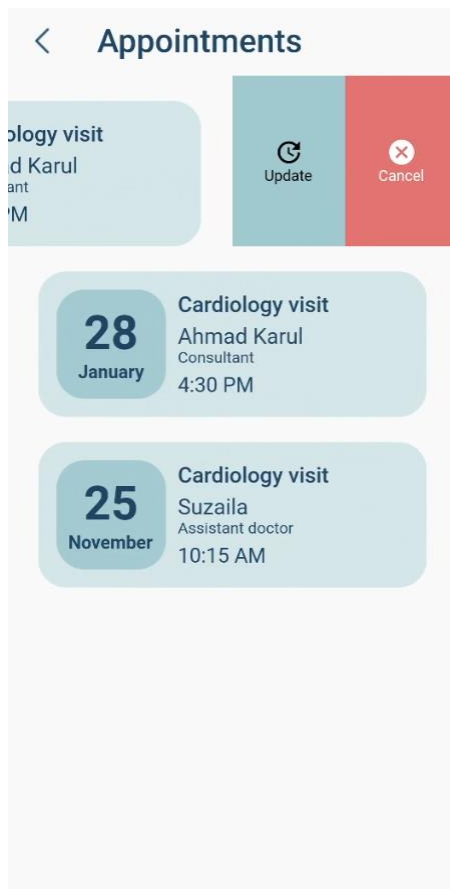


Figure 5.1.10 Slide action on appointment

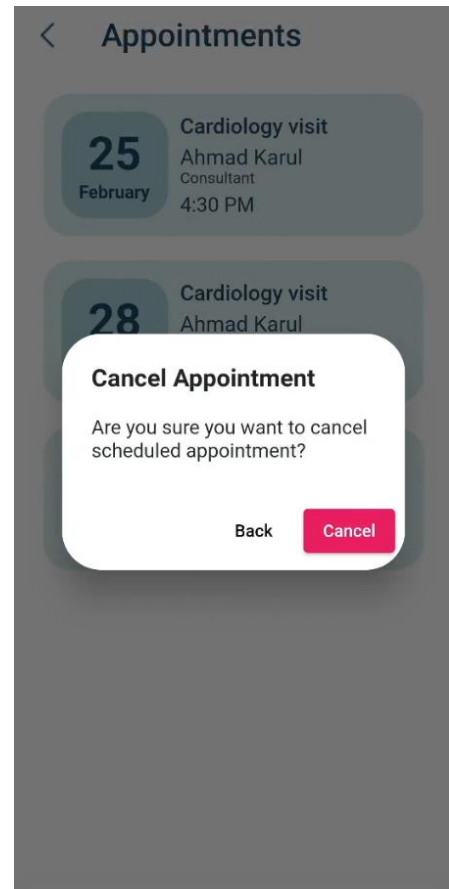


Figure 5.1.11 Cancel appointment

Figure 5.1.10 illustrates slide action on made appointments, which will allow patient to manage their appointments. If patient select the cancel appointment action, a confirmation dialog will be prompted to confirm patient's action as shown in Figure 5.1.11. If patient press the cancel button, the appointment will be cancelled and remove from the database. Then it will be removed from patient's appointment list.

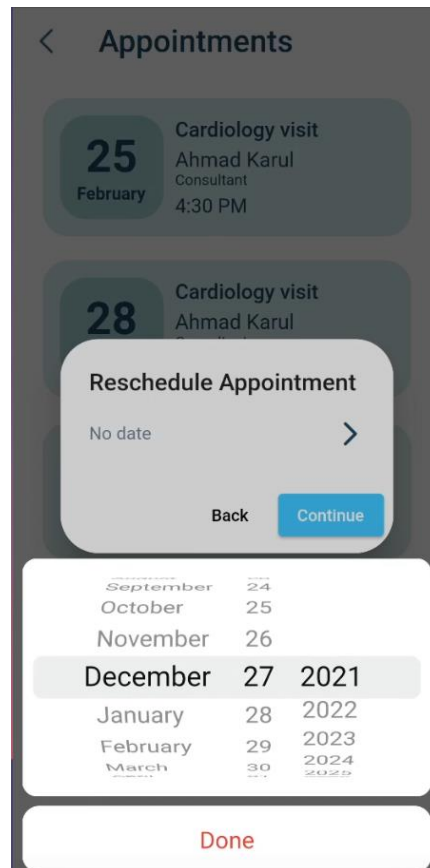


Figure 5.1.12 Reschedule appointment

If patient select update appointment action, a confirmation dialog will be prompted to get the rescheduled date as shown in Figure 5.1.12. After selecting a date, patient can press the continue button to reschedule the appointment to the new date. However, if the specific doctor for that date is not available, patient will receive an email from admin to reschedule to another suitable date.

5.1.5 Appointment Details

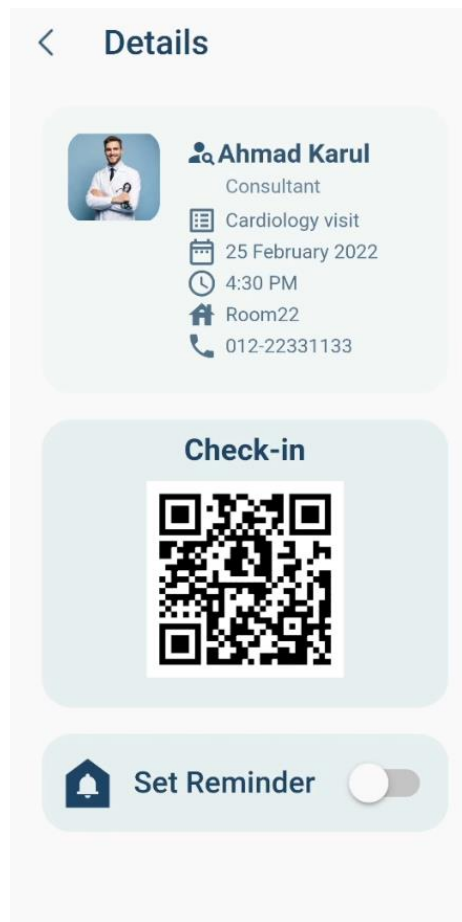


Figure 5.1.13 Appointment details page

If patient select an appointment from their appointment list, it will redirect them to the appointment detail page as illustrated in Figure 5.1.13. The page will contain details of the appointment and a generated QR code that can be scanned for check-in during appointment day. The QR code will contain patient and appointment info. Furthermore, patient can have the option to set customizable appointment reminders to notify themselves prior to the appointment.

5.1.6 Set appointment reminder

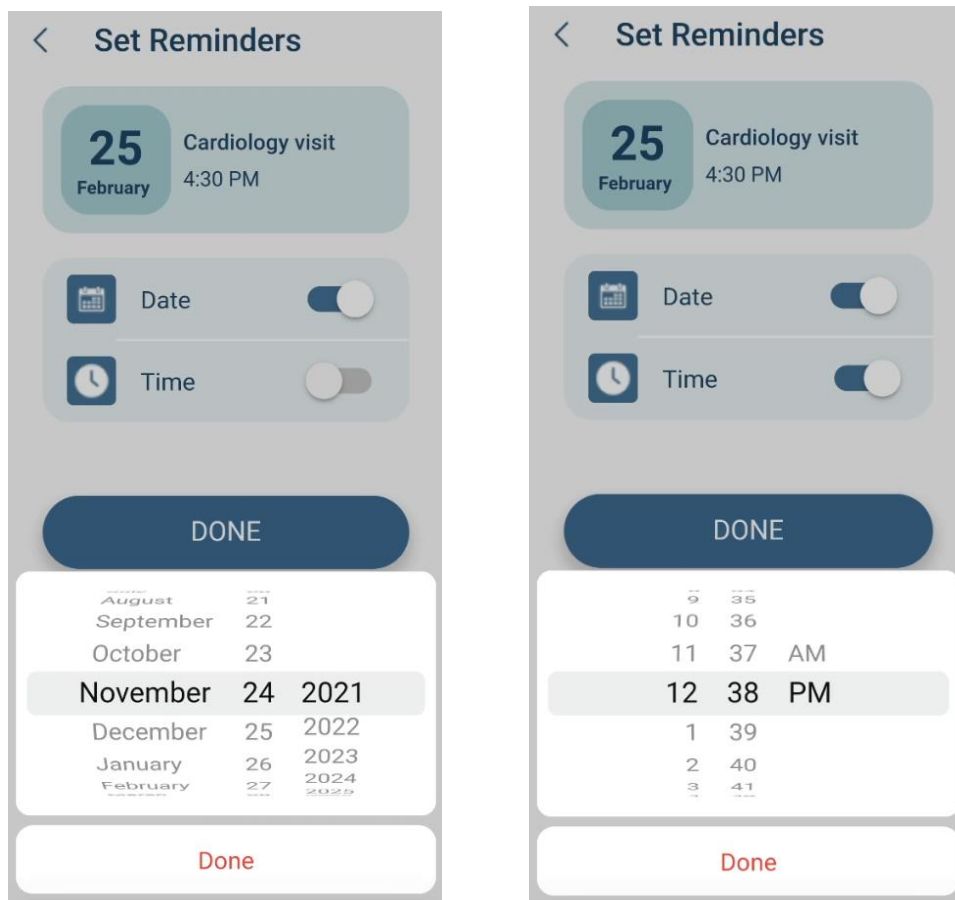


Figure 5.1.14 Set appointment reminder

After selecting the set reminder option, patient will be redirected to the set reminder page. Patient can then select specific date and time for the reminder. After pressing done, patient will be redirected to the reminder list page containing all appointment reminders.

5.1.7 Manage reminders

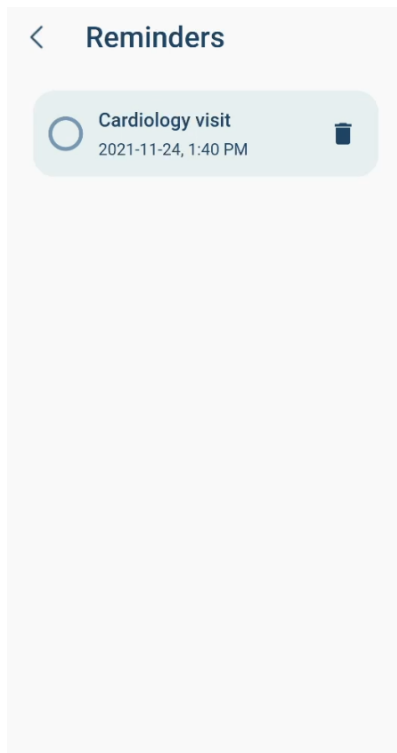


Figure 5.1.15 Reminder list page

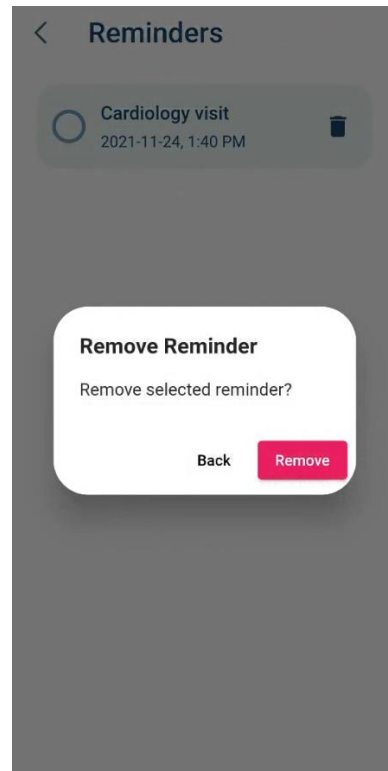


Figure 5.1.16 Remove reminder

Figure 5.1.15 illustrates the reminder list page which stores all the patient made reminders. Patient can check the radio button to prompt the confirmation dialog to confirm patient action to remove the reminder from the reminder list. Moreover, patient can press the delete icon to delete the reminder before it notifies the patient. The figure below shows the example of the reminder sent to patient's device.

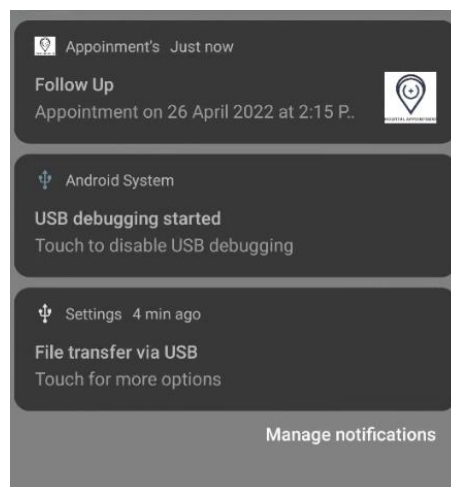


Figure 5.1.17 Appointment reminder

5.1.8 User profile

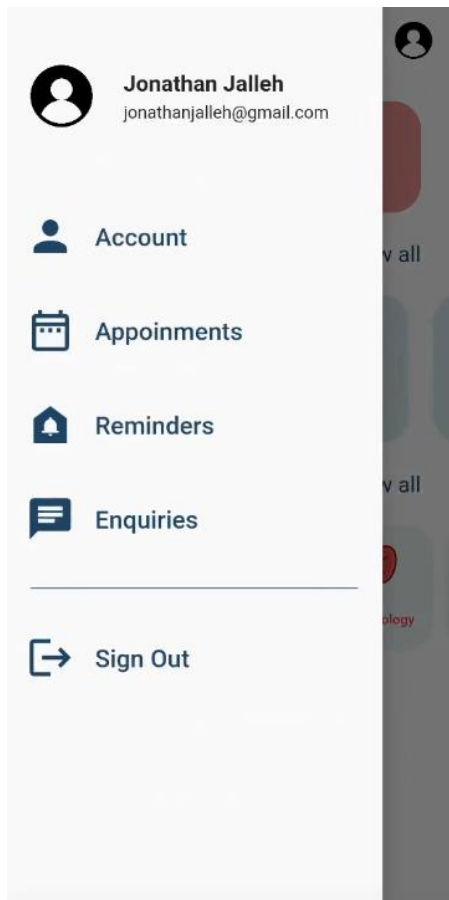


Figure 5.1.18 Menu drawer

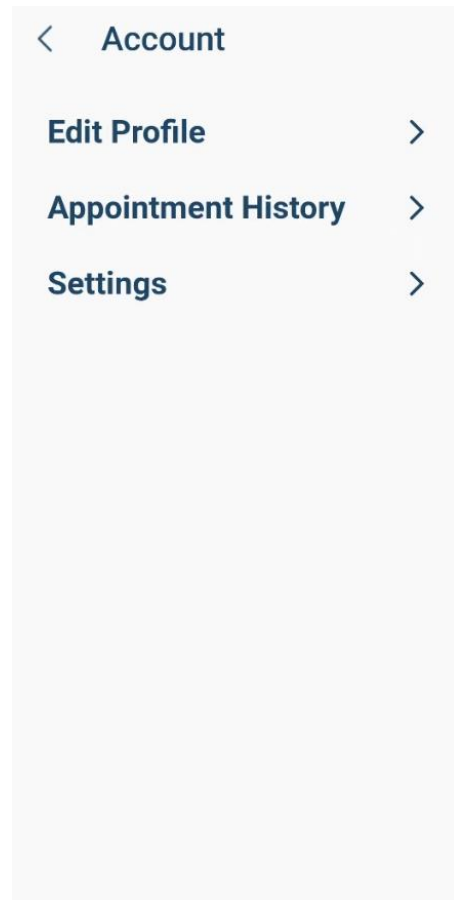


Figure 5.1.19 Account page

Besides that, by pressing the “hamburger” icon in the homepage, it will open the menu drawer which contains navigation to the account, appointment, reminders and enquiries page as shown in Figure 5.1.18. The sign out button is also at the bottom of the menu drawer to allow easy process of signing out. Figure 5.1.19 illustrates the account page after patient has press the account option from menu drawer. If patient select edit profile option, they will be redirected to edit profile page. Patient can be able to update their credentials here by selecting any of the fields and press done after confirming the credentials.

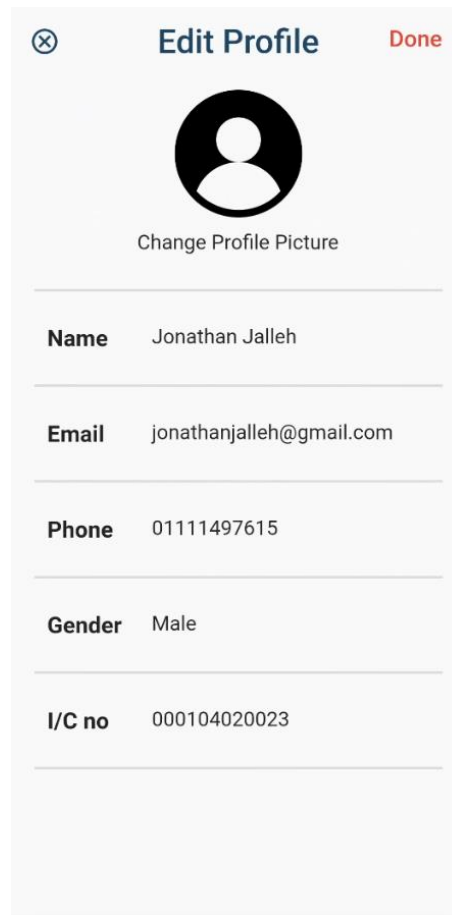


Figure 5.1.20 Edit profile page

5.1.9 Appointment history

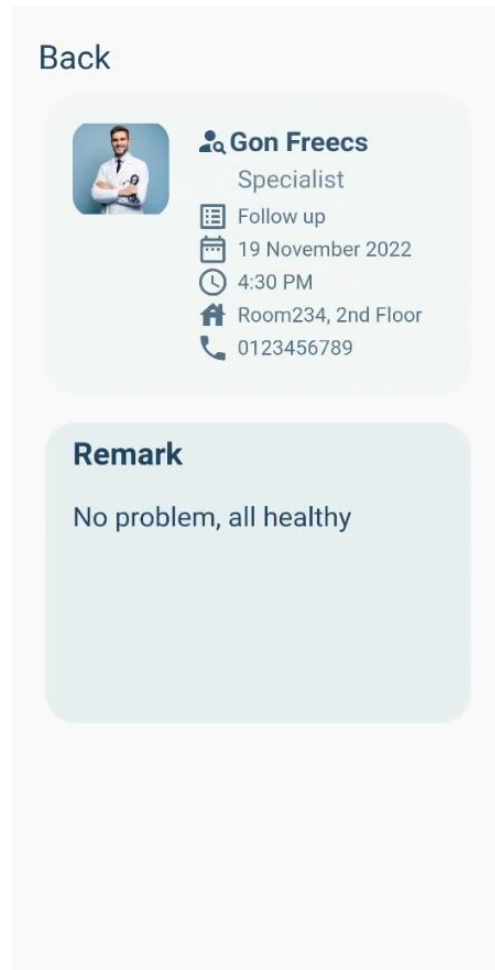
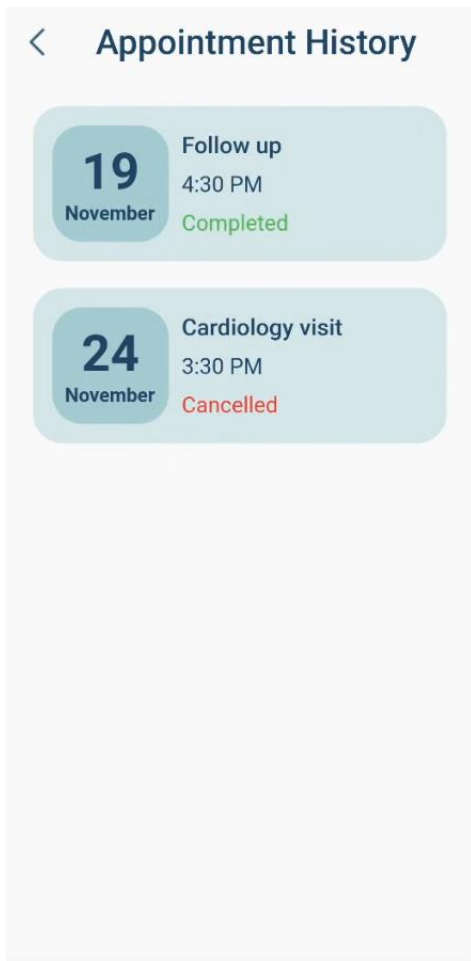


Figure 5.1.21 Appointment history page

Figure 5.1.22 Appointment history details

If patient select appointment history in the account page, they will be redirected to the appointment history page as shown in Figure 5.1.21. The page will contain all of patient past appointments, both completed and cancelled. After selecting the appointment, the details of the appointment containing with any remarks will be displayed for patient to view.

5.1.10 Chat Enquiries

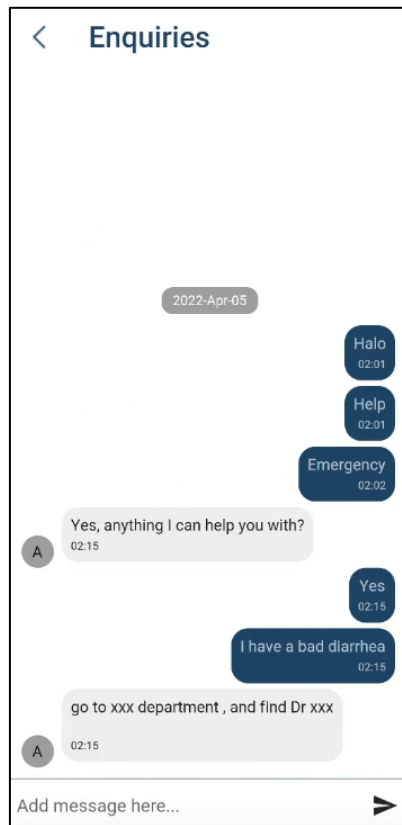


Figure 5.1.23 Chat Enquiries page

Patient can press the “Enquiries” option in menu drawer to navigate to the chat enquiries page as shown in the figure above. This page allows patient to enquire any details about the sickness they have, and which department is suitable to visit by the medical receptionist.

5.2 Admin Interface (Web Application)

5.2.1 Admin sign in

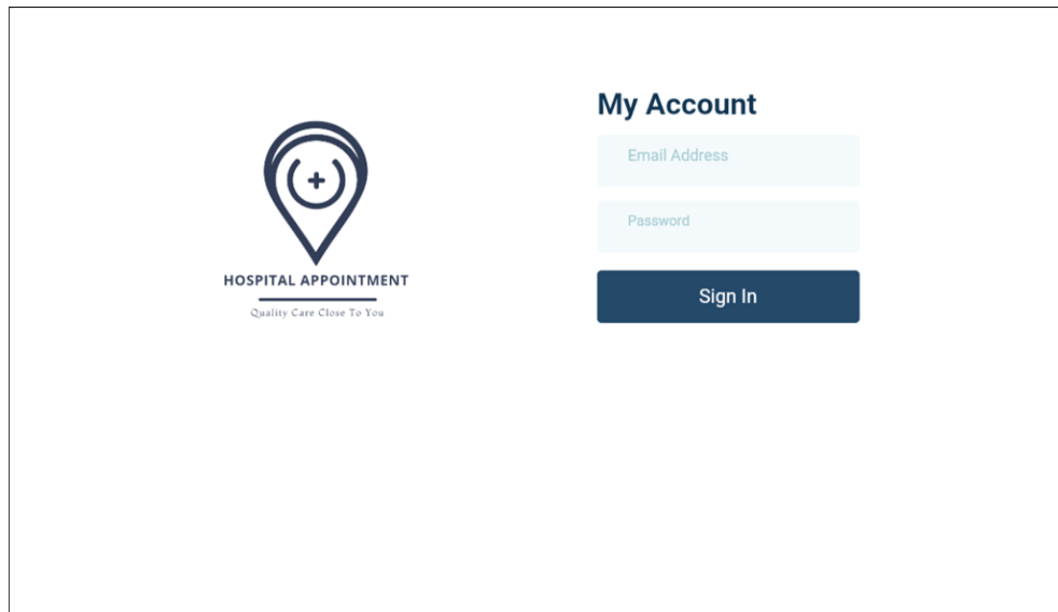


Figure 5.2.1 Admin login

The figure above shows the admin login page for admin web app. Admin can sign into the web app by entering a valid email and password that will be given from database administrator which will fill their credentials too. This is to prevent admins creating an account with fake or offensive email or credentials. The entered credentials will be validated by Firebase Authentication, an error message will be displayed if incorrect credentials are entered. Then, system will check if admin is under doctor or medical receptionist and will redirect them to the according homepage. This is because doctor and medical receptionist both have own privileges and access to different functions.

5.2.2 Homepage

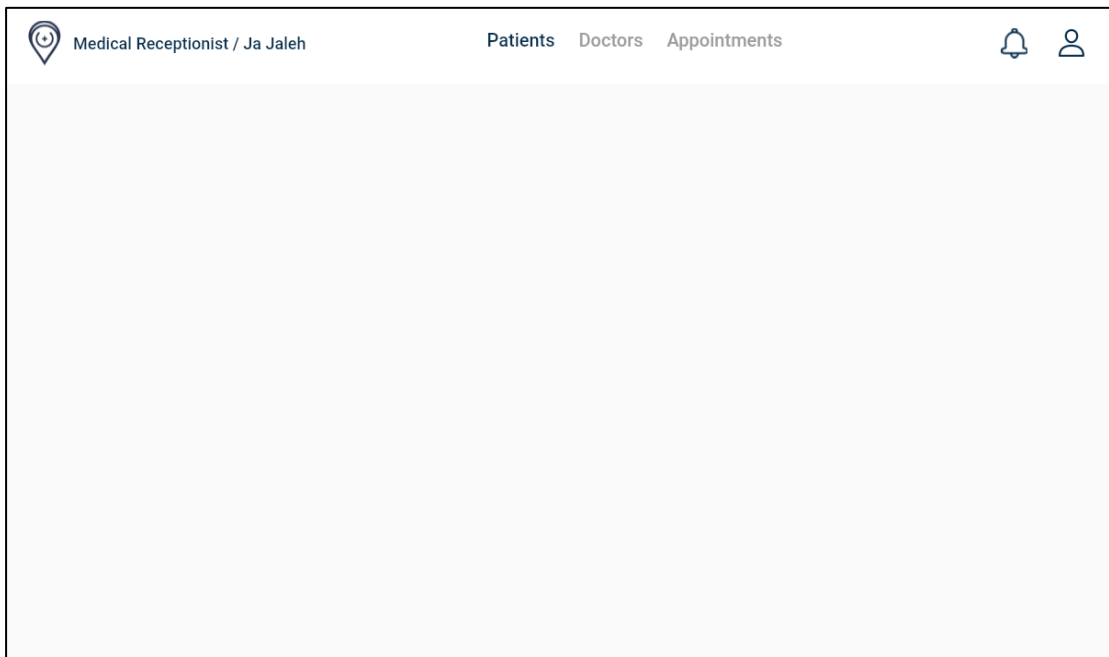


Figure 5.2.2 Homepage (medical receptionist)

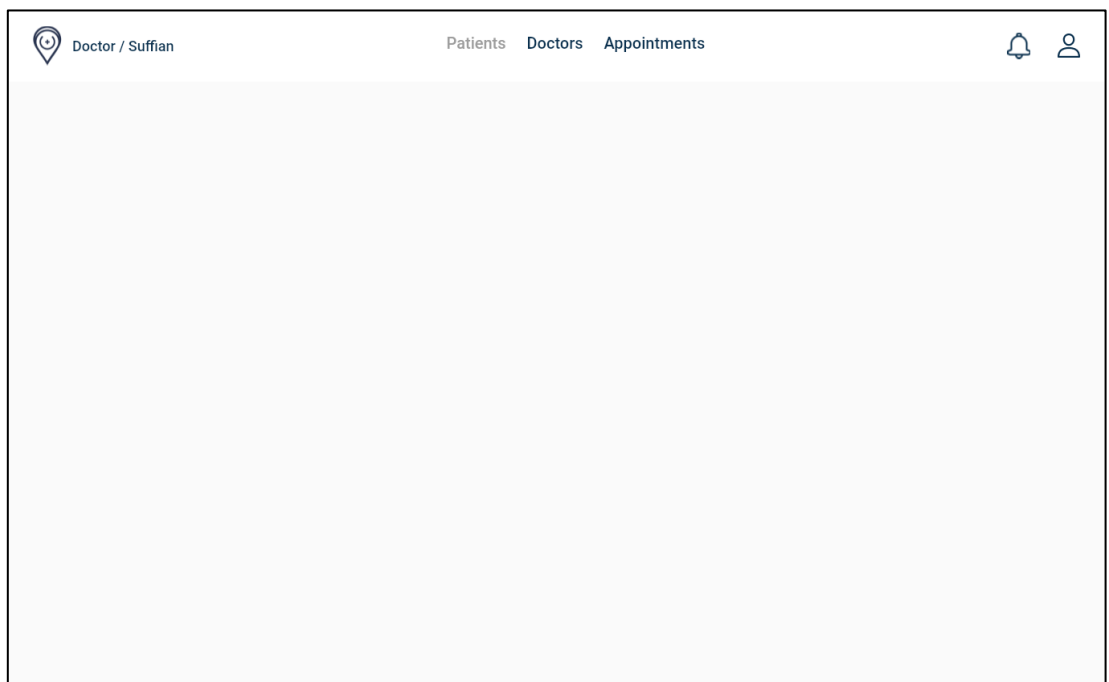


Figure 5.2.3 Homepage (Doctor)

CHAPTER 5: SYSTEM IMPLEMENTATION

Figure 5.2.2 illustrates the homepage for medical receptionist, while Figure 5.2.3 shows the homepage for doctor. Each admin role will have different privileges as medical receptionist will be able to access the patient menu and chat inquiries, while doctor will have privilege to the doctors and appointments menu.

5.2.3 Profile page

Doctor / Suffian Patients Doctors Appointments

Doctor Details

Dr. Suffian

Specialization Cardiology

Email Address Suffian@mail.com

Gender Male

Phone Number 012-3385435

Room No Room2312

Timeslot/Calendar

Select Date

Add available time

Add

Figure 5.2.4 Profile (Doctor)

In the profile page for doctor as shown in Figure 5.2.4, it will display the picture of doctor along with their details. Besides that, they can select a date to see current available time and may add any new available time so user/patient by pressing the add button to update the available timeslot in the database.

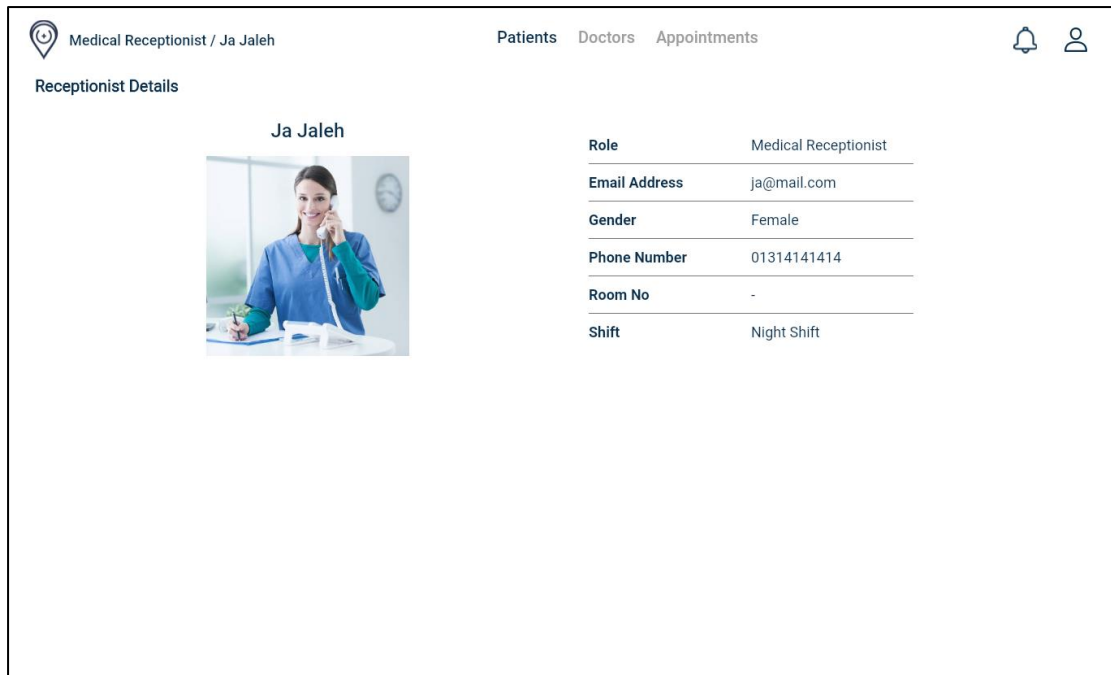
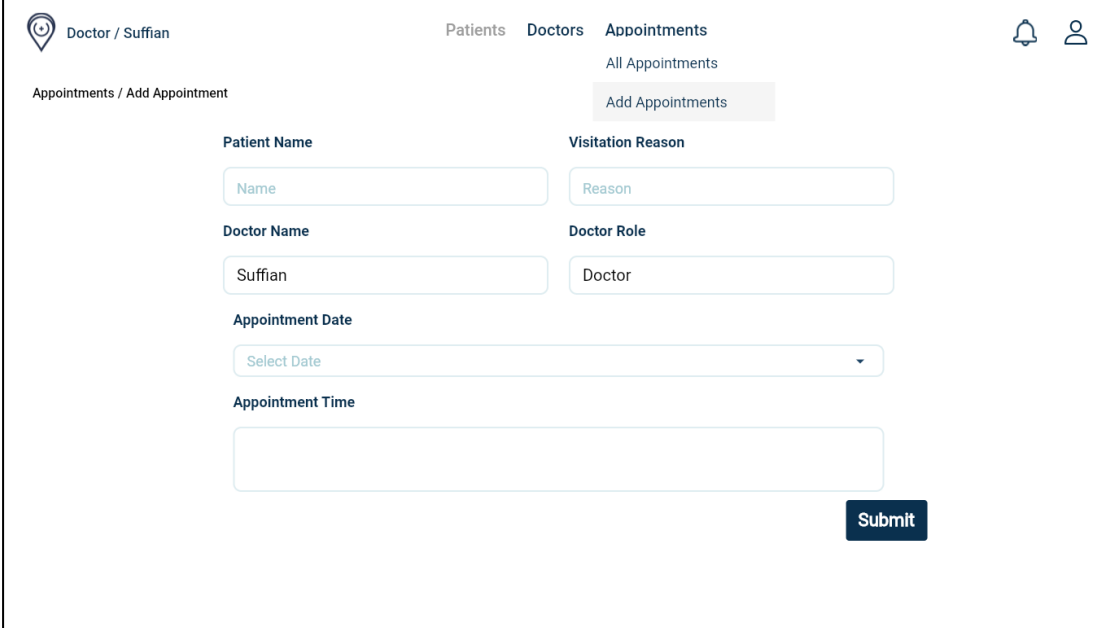


Figure 5.2.5 Profile (Medical Receptionist)

However, for medical receptionist profile as shown in Figure 5.2.5, it will only display the profile picture and their details only.

5.2.4 Add Appointments



The screenshot shows a web interface for a doctor named Suffian. The top navigation bar includes 'Patients', 'Doctors', and 'Appointments'. Under 'Appointments', there are options for 'All Appointments' and 'Add Appointments'. The main form area is titled 'Appointments / Add Appointment' and contains the following fields:

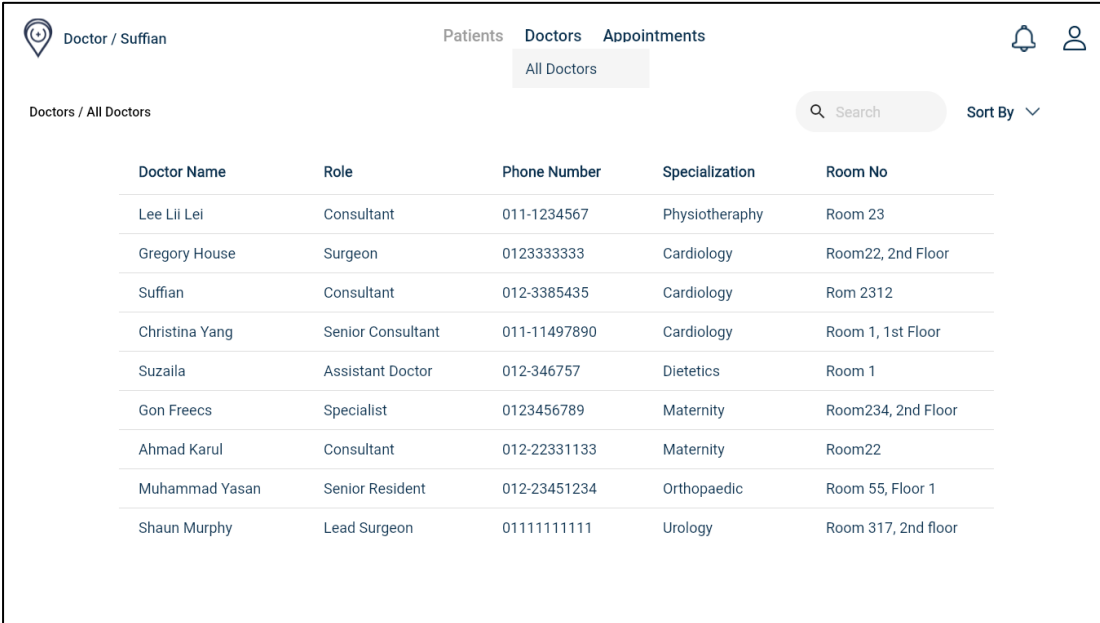
- Patient Name:** A text input field with the placeholder 'Name'.
- Visitation Reason:** A text input field with the placeholder 'Reason'.
- Doctor Name:** A text input field with the pre-filled value 'Suffian'.
- Doctor Role:** A text input field with the pre-filled value 'Doctor'.
- Appointment Date:** A date selection dropdown menu with the placeholder 'Select Date'.
- Appointment Time:** A large empty text input field.

A dark blue 'Submit' button is located at the bottom right of the form.

Figure 5.2.6 Add Appointments

Figure 5.2.6 illustrates the add appointment page. Doctor can be navigated to this page after selecting the “Add Appointments” option in the Appointment menu. After that, the pre-filled name and role of doctor will be displayed. Doctor will then have to enter the patient’s name, visitation reason, appointment date and time. If there are empty fields, an error dialog will be displayed and navigated back to the same page. If it is successful, the system will display an alert dialog that when closed brings back doctor to homepage.

5.2.5 Doctors List

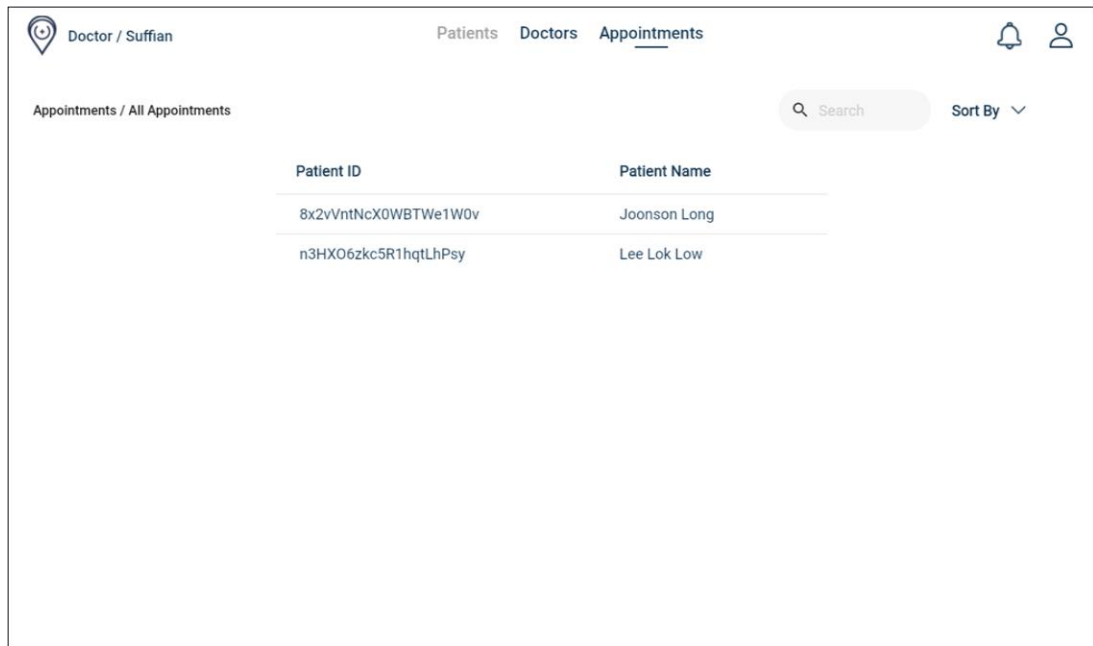


Doctor Name	Role	Phone Number	Specialization	Room No
Lee Lii Lei	Consultant	011-1234567	Physiotherapy	Room 23
Gregory House	Surgeon	0123333333	Cardiology	Room22, 2nd Floor
Suffian	Consultant	012-3385435	Cardiology	Rom 2312
Christina Yang	Senior Consultant	011-11497890	Cardiology	Room 1, 1st Floor
Suzaila	Assistant Doctor	012-346757	Dietetics	Room 1
Gon Freecs	Specialist	0123456789	Maternity	Room234, 2nd Floor
Ahmad Karul	Consultant	012-22331133	Maternity	Room22
Muhammad Yasan	Senior Resident	012-23451234	Orthopaedic	Room 55, Floor 1
Shaun Murphy	Lead Surgeon	01111111111	Urology	Room 317, 2nd floor

Figure 5.2.7 Doctors List

Figure 5.2.7 shows the list of doctors together with their role and specialization. It will be displayed after doctor has selected the “All Doctors” option from the Doctors menu. This list will be for reference when doctors want to set up a follow up appointment for their patients to others specialist doctors.

5.2.6 Appointment List



Patient ID	Patient Name
8x2vVntNcX0WBTWe1W0v	Joonson Long
n3HX06zkc5R1hqtLhPsy	Lee Lok Low

Figure 5.2.8 Doctor's appointment list

Figure 5.2.8 illustrates the appointment list of the doctor containing the specific patient's name and patient ID. Doctor will be navigated to this page by selecting on the "All appointments" option of Appointment menu. Then, selection of the patient will then redirect doctor to the patient's appointment list as shown in Figure 5.2.9. This page shows all the appointments the patient has had with the doctor or any future appointments. If doctor selects an appointment from the list, it will navigate to the patient appointment details page as displayed in Figure 5.2.10. This page will display the data retrieved from the database such as patient and appointment details along with the description or remarks given by the doctor.

The screenshot shows a web interface for a doctor named Suffian. At the top, there are navigation tabs for 'Patients', 'Doctors', and 'Appointments', with 'Appointments' being the active tab. The patient's name 'Joonson Long' is displayed. Below this is a table with three columns: 'Appointment Name', 'Appointment Date', and 'Appointment Status'. The table contains two rows of data.

Appointment Name	Appointment Date	Appointment Status
Consultation Visit	Wed, 24 November 2021	Completed
Follow Up	Wed, 2 February 2022	Completed

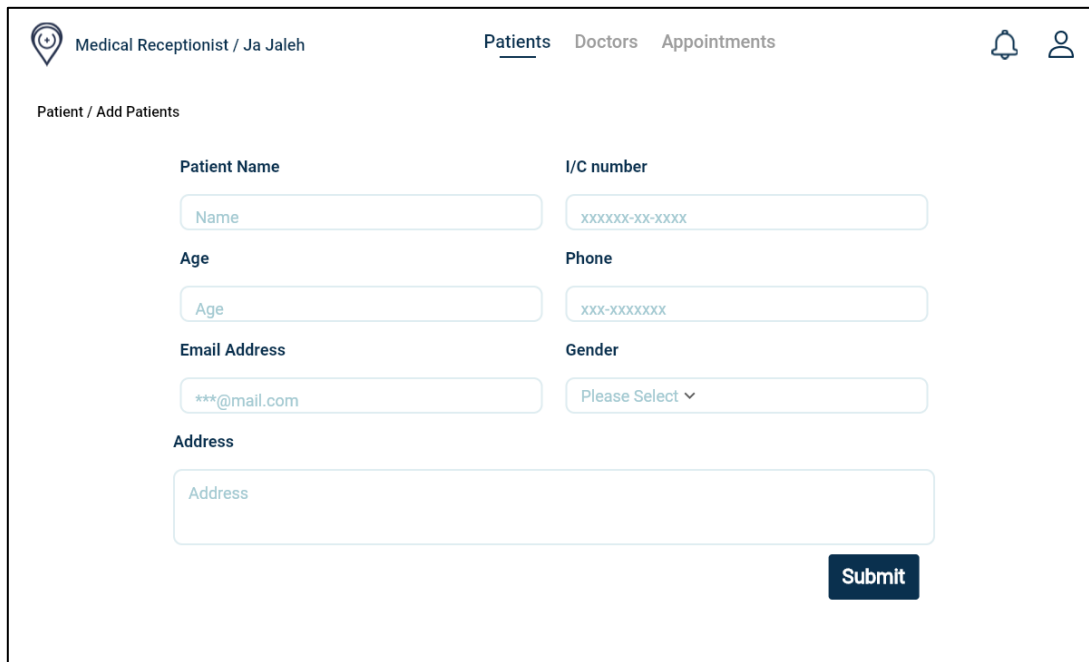
Figure 5.2.9 Doctor's patient appointment list

The screenshot shows the 'Appointment Details' page for the same patient, Joonson Long. It displays various appointment details in a key-value format. At the bottom right, there is an 'Edit' button.

Appointments / Appointment Details	
Patient Name	Joonson Long
Patient ID	8x2vVntNcX0WBTWe1W0v
Appointment Date	Wed, 24 November 2021
Time Slot	9:15 AM
Descriptions	All is well, no cold water for one week

Figure 5.2.10 Doctor's patient appointment details

5.2.7 Add Patients



The screenshot shows a web interface for a medical receptionist. At the top, the user is identified as 'Medical Receptionist / Ja Jaleh'. The navigation menu includes 'Patients', 'Doctors', and 'Appointments'. There are also notification and user profile icons. The main heading is 'Patient / Add Patients'. The form contains the following fields:

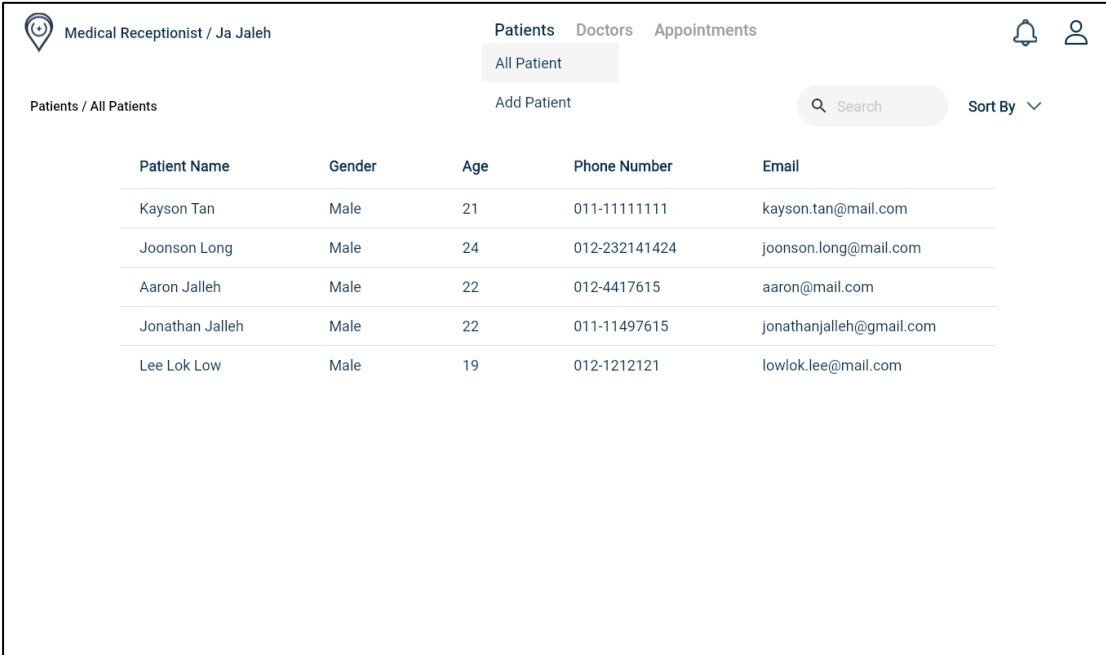
- Patient Name:** A text input field with the placeholder 'Name'.
- I/C number:** A text input field with the placeholder 'xxxxxx-xx-xxxx'.
- Age:** A text input field with the placeholder 'Age'.
- Phone:** A text input field with the placeholder 'xxx-xxxxxxx'.
- Email Address:** A text input field with the placeholder '***@mail.com'.
- Gender:** A dropdown menu with the placeholder 'Please Select' and a downward arrow.
- Address:** A large text input field with the placeholder 'Address'.

A dark blue 'Submit' button is located at the bottom right of the form.

Figure 5.2.11 Add Patients

Figure above shows the adds patients page, which is for medical receptionist to add patients that does not have the application. They will be redirected to this page by selecting the “Add Patients” option in the Patients Menu. Error dialog will be displayed if there are empty fields when the submit button is pressed. However, if it is successful, the system will display an alert dialog that will bring back to homepage when closed.

5.2.8 Patients List



Patient Name	Gender	Age	Phone Number	Email
Kayson Tan	Male	21	011-11111111	kayson.tan@mail.com
Joonson Long	Male	24	012-232141424	joonson.long@mail.com
Aaron Jalleh	Male	22	012-4417615	aaron@mail.com
Jonathan Jalleh	Male	22	011-11497615	jonathanjalleh@gmail.com
Lee Lok Low	Male	19	012-1212121	lowlok.lee@mail.com

Figure 5.2.12 Patient List

If medical receptionist selects “All Patient” option, they will be navigated to the patients list page. This page will retrieve data of all the patients from the database, then display the patient details in a table. After selecting a specific patient, it will be navigated to the patient details page as shown in Figure 5.2.13. This page will display the patient profile which shows patient information in details and patient past visits.

CHAPTER 5: SYSTEM IMPLEMENTATION

The screenshot displays a web application interface for a medical receptionist. At the top, the user is identified as 'Medical Receptionist / Ja Jaleh'. The navigation menu includes 'Patients' (which is underlined), 'Doctors', and 'Appointments'. There are also notification and user profile icons in the top right corner.

The main content area is titled 'Patients / Patient Details' and features a 'Patient Profile' section. On the left, there is a placeholder for a patient photo. To the right, the patient's details are listed in a two-column format:

Name	Joonson Long	Address	123, Taman Hati 4, Jalan Mati
Age	22	Weight	-
Gender	Male	Height	-
Phone Number	012-232141424	Emergency Contact	011-112347897
Email	joonson.long@mail.com		

Below the profile section is a 'Patient Visits' table:

Doctor Name	Visit Date	Appointment Name
Suffian	24-11-2021	Consultation Visit
Suffian	02-02-2022	Follow Up

Figure 5.2.13 Patient Details

5.2.9 Chat Inquiries

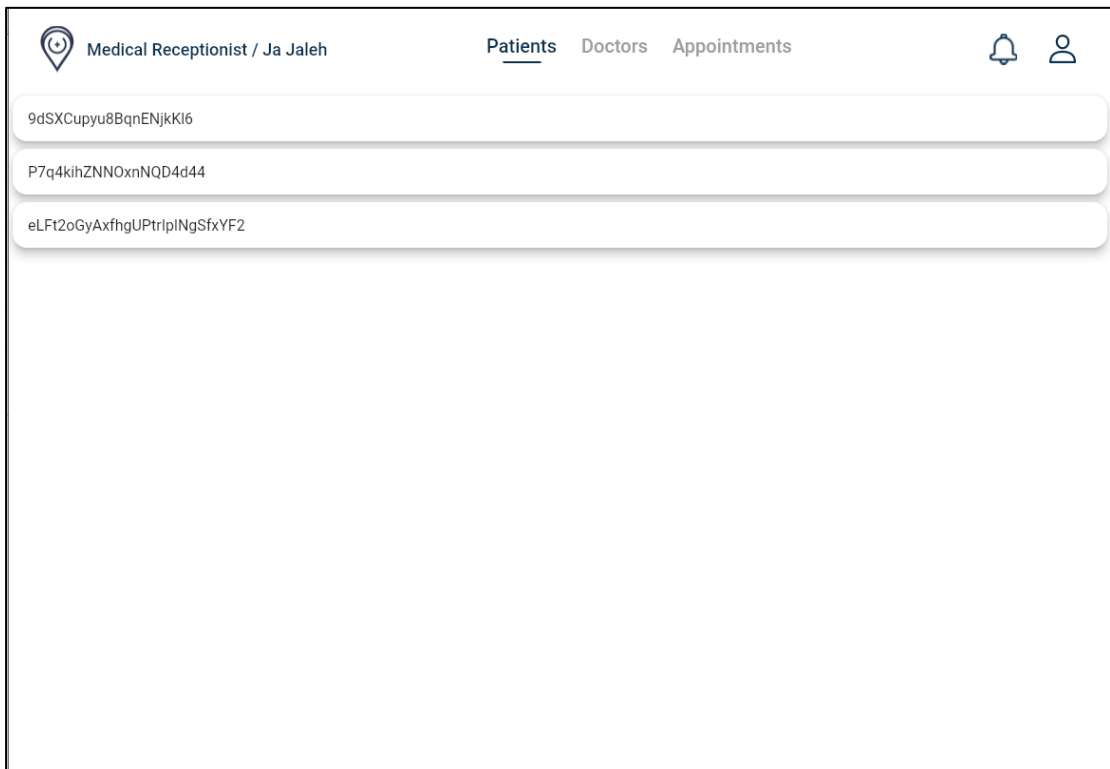


Figure 5.2.14 Chat List

Figure 5.2.14 illustrates the chat list page, which consists of chat ID that are created when a patient sent an inquiry from their mobile application. It is anonymous and no name of the patient will appear. Then, if one of the lists is pressed, it will be redirected to the patient chat page as illustrated in Figure 5.2.15. Medical receptionist can reply to any enquiry by the patient and patient will receive it in a short time.

CHAPTER 5: SYSTEM IMPLEMENTATION

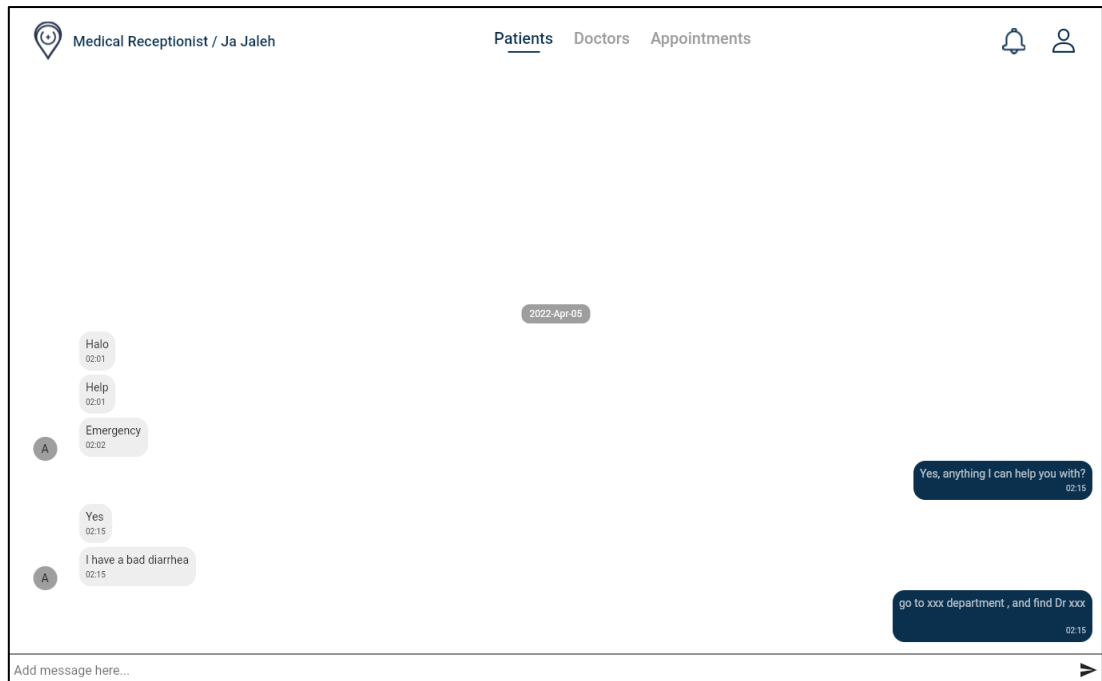


Figure 5.2.15 Patient Chat

5.3 Coding Explanation (User Mobile Application)

A new Flutter project is created in Android Studio to start the project development for the user mobile application. Then, the project will be connected to Firebase to use its features such as user authentication and Cloud Firestore. Furthermore, package dependencies are added to the pubspec.yaml file to use extra Dart libraries and features.

5.3.1 Create Account

```

createAcc() {
  if (formKey.currentState.validate()) {
    String gender;
    int number = int.parse(icController.text.substring(11));
    number % 2 == 0 ? gender = "Female": gender = "Male";
    Map<String, String> userInfo = {
      "name": nameController.text,
      "email": emailController.text,
      "ic": icController.text,
      "phone": phoneController.text,
      "gender" : gender
    };
    setState(() {
      isLoading = true;
    });
    authMethods
      .signInWithEmailAndPassword(
        emailController.text,
        passwordController.text)
      .then((val) {
        databaseMethods.uploadUserInfo(userInfo, val.uid);
        Navigator.pushReplacement(context,
          MaterialPageRoute(
            builder: (context) => homepage(val.uid))); // MaterialPageRoute
      });
  }
}

```

Figure 5.3.1 Create Account (1)

```
Future signUpwithEmailAndPassword(String email, String password) async {  
  try {  
    UserCredential result = await _auth.createUserWithEmailAndPassword(  
      email: email, password: password);  
    User firebaseUser = result.user;  
    return _userFromFirebaseUser(firebaseUser);  
  } catch (e) {  
    return null;  
  }  
}
```

Figure 5.3.2 Create Account (2)

Figure 5.3.1 illustrates the code for create account function. After patient press the create account button, their inputs which are their personal details will be retrieved and validated. If there are invalid inputs, an error message will be displayed and request user to input again. If the inputs are valid, the system will create user with Firebase Auth `createUserWithEmailAndPassword()` method as shown in Figure 5.3.2. If user is created successfully, the user data will be inserted to the database, then the system will direct user to the homepage.

5.3.2 Sign In

```

signIn() async {
  if (formKey.currentState.validate()) {
    setState(() {
      isLoading = true;
    });

    dynamic result = await authMethods.signInWithEmailAndPassword(
      emailController.text,
      passwordController.text);

    if (result == null) {
      setState(() {
        isLoading = false;
        final snackBar = SnackBar(
          content: const Text("INVALID CREDENTIALS",textAlign: TextAlign.center,),
          behavior: SnackBarBehavior.floating,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(24)
          ), // RoundedRectangleBorder
          backgroundColor: Colors.red,
        ); // SnackBar
        ScaffoldMessenger.of(context).showSnackBar(snackBar);
      });
    }
    if(result!=null){
      Navigator.pushReplacement(context, MaterialPageRoute(
        builder: (context) =>
          homepage(result.uid)
      )); // MaterialPageRoute
    }
  }
}

```

Figure 5.3.3 Sign In (1)

```

Future signInWithEmailAndPassword(String email, String password) async {
  try {
    UserCredential result = await _auth.signInWithEmailAndPassword(
      email: email, password: password);
    User firebaseUser = result.user;
    return _userFromFirebaseUser(firebaseUser);
  } catch (e) {
    print(e.toString());
  }
}

```

Figure 5.3.4 Sign In (2)

Figure 5.3.3 shows the coding of the sign in function. When user clicks the sign in button, the function is called. The system will pass the user inputs which are email and password and validate it. If invalid email or password are input, system will display an error message to notify user. If valid email and password are entered, system will call the Firebase Auth `signInWithEmailAndPassword()` function as shown in figure 5.3.4. This function will authenticate the user, and if successful will navigate user to the home page. If authenticate fail, then user will be navigated back to sign in page.

5.3.3 Homepage

```

getReminderInfo(String uid) async{
  return await UserCollection.doc(uid).collection("Reminders")
    .get();
}
QuerySnapshot remindInfo;
Future getRemindInfo() async{
  await dbs.getReminderInfo(widget.uid)
    .then((value){
      setState(() {
        remindInfo = value;
      });
    });
}

```

Figure 5.3.5 Retrieve patient reminder data

```

return StreamBuilder(
  stream: FirebaseFirestore.instance.collection("Users").doc(user.uid)
    .collection("Appointments").snapshots(),
  builder: (context, snapshots){
    if(snapshots.hasData && !snapshots.hasError && snapshots.data.docs.length>0){

```

Figure 5.3.6 Retrieve patient appointment data

Figure 5.3.5 shows the function to retrieve patient reminder data using the `get()`. The data is then stored in a `QuerySnapshot` to be used later. Moreover, the code in Figure 5.3.6 is used to retrieve the patient appointment data and stored in `snapshots`. Then, the system will validate whether the snapshots have any data and does not have any error before it proceeds to display the patient appointment data.

5.3.4 Make Appointment

```

getTimeBusy(String dateID) {
    FirebaseFirestore.instance.collection("Doctors")
        .doc(docInfo.docs[0].id).collection("date")
        .doc(dateID).get().then((val) {
            if (val.data() == null) {
                timeslot = <String>[];
            }
            try {
                if (val.data().isNotEmpty) {
                    setState(() {
                        timeslot = List.from(val['timeBusy']);
                    });
                }
            } catch (exception) {
                if (kDebugMode) {print(exception);}
            }
        });
}

```

Figure 5.3.7. Get doctor busy time

```

if (timeslot.isNotEmpty) {
    for (String times in timeslot) {
        if (times == choice.title) {
            flag = true;
        }
    }
}
if (flag) {
    return Container(
        height: 35.0,
        width: 80.0,
        decoration: BoxDecoration(
            color: Colors.grey[300],
            borderRadius: BorderRadius.circular(5),
        ), // BoxDecoration
        child: Center(
            child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    Text(
                        choice.title,
                        style: const TextStyle(
                            color: Color(0xffB6C4CE), fontWeight: FontWeight.bold),
                    ), // Text
                ],
            ),
        ),
    );
}

```

Figure 5.3.8 Check doctor available time

After patient have select the scheduled appointment option when selecting a specific specialist, they will be directed to the make appointment page. Then after selecting a

date, the above function shown in Figure 5.3.7 is run to check if there are any time that doctor is busy at that specific date. If there are any times that doctors are busy, then the data will be stored in the timeslot array list. Then, the values in the timeslot array list is looped and compared to the available time (choice.title). The timeslots which doctors are busy will change its color to grey and will remove its selection function to prevent patient to select it.

```

bookAppointment(String uid, String date) {
  if (dateSelected.isNotEmpty) {
    Map<String, dynamic> appointmentMap = {
      "docName": widget.docName,
      "docRole": widget.docRole,
      "apptName": widget.category + " visit",
      "apptDate": dateSelected,
      "apptTime": timeSelected
    };
    dbs.addAppointment(uid, appointmentMap);
    dd.addTime(docInfo.docs[0].id, dateID, timeSelected);
    dateSelected = "";
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) =>
          SuccessAppointment()
      )); // MaterialPageRoute
  }
}

```

Figure 5.3.9 Book Appointment (1)

```

addReminders(uid, reminderMap){
  UserCollection.doc(uid).collection("Reminders").add(reminderMap)
    .catchError((e){
      print(e.toString());
    });
}

addTime(String docID, String date, String apptTime) async {
  var list = [apptTime];
  DocCollection.doc(docID).collection("date").doc(date).set({
    "timeBusy" : apptTime
  });
  DocCollection.doc(docID).collection("date").doc(date).update({
    "timeBusy": FieldValue.arrayUnion(list)
  });
}

```

Figure 5.3.10 Book Appointment (2)

After selecting a valid date and time, patient will press the book appointment button. The system will then trigger the function in Figure 5.3.9. The function will first store the required data into a Map, then it will add the reminder data to Firestore and update the doctor unavailable date with the functions shown in Figure 5.3.10. After date, the date selected will be reset to empty and user will be navigated to the appointment successful page. The Figure below displays the code to display the current patient queue for same day appointments. It will retrieve data from the Cloud Firestore and order it based on the ascending time. Then, it will display the data in an alert dialog for the patient to view.

```

- return AlertDialog(
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.all(Radius.circular(32.0))), // RoundedR
  title: const Text("Current Queue"),
  content: Builder(
    builder: (context) {
      return Container(
        height: 200,
        child: StreamBuilder(
          stream: FirebaseFirestore.instance
            .collection("Doctors")
            .doc(docid)
            .collection("Appointment")
            .doc(dateid)
            .collection("timeslot")
            .orderBy("time")
            .snapshots(),
          builder: (context, snapshots) {
            if (snapshots.hasData && !snapshots.hasError) {
              return ListView.builder(
                itemCount: snapshots.data.docs.length,
                itemBuilder: (context, index) {
                  return Container(
                    height: 50,
                    child: Row(
                      mainAxisAlignment:
                        MainAxisAlignment.spaceBetween,
                      children: [
                        Text(DateFormat.jm().format(snapshots
                          .data.docs[index]["time"]
                          .toDate())), // Text
                        Text(snapshots.data.docs[index]["status"],
                          style: snapshots.data.docs[index]
                            ["status"] ==
                              "Completed"
                                ? TextStyle(color: Colors.green)
                                : TextStyle(color: Colors.black)),
                      ],
                    )); // Row, Container
                },
              ); // ListView.builder
            }
          },
        );
      }
    );

```

Figure 5.3.11 View current queue

5.3.5 Manage Appointment

```

secondaryActions:[
  — IconSlideAction(
    caption: 'Update',
    color: const Color(0xffA1C8D1),
    icon: Icons.update,
    onTap: () {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return
            updateDialog(
              user.uid,
              snapshots.data.docs[index].id,
              snapshots.data.docs[index]["apptTime"],
              snapshots.data.docs[index]["docName"],
              snapshots.data.docs[index]["docRole"],
              snapshots.data.docs[index]["apptName"],
            );
        });
    },
  ), // IconSlideAction
  — IconSlideAction(
    caption: 'Cancel',
    color: Colors.red[300],
    icon: Icons.cancel,
    onTap: () {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return cancelDialog(
            user.uid, snapshots.data.docs[index].id,
            context,docInfo.docs[0].id, dateID );
        });
    });
}

```

Figure 5.3.12 Slidable actions

In the appointment list page, patient can view all their made appointments, and do multiple actions on it by sliding the appointment to the right as shown in the figure above.

Update appointment

```

ElevatedButton(
  child: const Text("Continue"),
  style: ElevatedButton.styleFrom(
    primary: Colors.lightBlueAccent,
  ),
  onPressed: () {
    DatabaseService(uid: userID).rescheduleAppt(
      apptID, dateSelected, apptTime, docName, docRole, apptName);
    final snackBar = SnackBar(
      content: Container(
        height: 30.0,
        child: const Text(
          "Appointment Rescheduled",
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 20.0),
        )), // Text, Container
      behavior: SnackBarBehavior.floating,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(24)), // RoundedRectangleBorder
      backgroundColor: Colors.lightBlueAccent,
    ); // SnackBar
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
    Navigator.pop(context);
  },
) // ElevatedButton

```

Figure 5.3.13 Update appointment (1)

```

Future rescheduleAppt(String apptID, String newDate, String newTime,
  String docName, String docRole, String apptName) async{
  return await UserCollection.doc(uid).collection("Appointments").doc(apptID).set({
    'apptDate': newDate,
    'apptTime' : newTime,
    'docName' : docName,
    'docRole' : docRole,
    'apptName': apptName
  });
}

```

Figure 5.3.14 Update appointment (2)

If patient select the update option, an update dialog will appear after passing some arguments to it such as appointment time and patient ID. After selecting a new updated date, patient can select the Continue button to proceed to reschedule the appointment as shown in Figure 5.3.13. The code in Figure 5.3.14 will be called to update the data in the Cloud Firestore. After that, a snack bar containing the pop-up message will be displayed to notify patient of successful appointment reschedule.

Cancel appointment

```

ElevatedButton(
  child: const Text("Cancel"),
  style: ElevatedButton.styleFrom(
    primary: Colors.pink,
  ),
  onPressed: () {
    DatabaseService(uid: userid).cancelAppointment(apptid);
    DoctorDetails(uid: docID).deleteTime(dateID);
    final snackBar = SnackBar(
      content: Container(
        height: 30.0,
        child: const Text(
          "Appointment Cancelled",
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 20.0),
        )), // Text, Container
      behavior: SnackBarBehavior.floating,
      shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
      backgroundColor: Colors.red,
    ); // SnackBar
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
    Navigator.pop(context);
  },
) // ElevatedButton

```

Figure 5.3.15 Cancel appointment (1)

```

Future cancelAppointment(String apptID) async{
  await UserCollection.doc(uid).collection("Appointments").doc(apptID).get().then((val){
    UserCollection.doc(uid).collection("AppointmentHistory").add({
      'apptDate': val["apptDate"],
      'apptTime': val["apptTime"],
      'apptName': val["apptName"],
      'docName': val["docName"],
      'docRole': val["docRole"],
      'reason': "Cancelled",
      'docRemark': "",
    });
    val.reference.delete();
  });
}

deleteTime( String dateID){
  DocCollection.doc(uid).collection("date").doc(dateID).delete();
}

```

Figure 5.3.16 Cancel appointment (2)

If patient select cancel option, a cancel dialog will appear after passing the required arguments as shown in Figure 5.2.15. If user select the cancel button, then the system will proceed to cancel the appointment by calling the method as shown in Figure 5.2.16 to remove the appointment data from the Firestore “Appointment” collection and move it to the “Appointment History” collection as a Cancelled Appointment. Moreover, the date will be removed from the doctor’s busy timeslot from Cloud Firestore so that the

timeslot will be available again. A snack bar will notify the patient of the appointment cancellation.

5.3.6 Appointment Details

```

dbs.getAppointmentInfo(widget.apptID, widget.uid).then((value) {
  setState(() {
    apptInfo = value;
  });
  getAppointmentInfo(String apptID, String uid) async{
    return await UserCollection.doc(uid).collection("Appointments").doc(apptID)
      .get();
  }
}

```

Figure 5.3.17 Retrieving appointment data

The figure above shows the code for retrieving the patient's appointment data from Cloud Firestore and it is retrieved from the "Appointments" collection. The system will store the retrieved data in a DocumentSnapshot which will be used to display the data.

```

String appointmentQR = apptInfo["apptDate"] + ";" +apptInfo["apptTime"]
  + ";" +apptInfo["docName"]+ ";" +apptInfo["docRole"];

```

Figure 5.3.18 Creating QR information

```

— QImage(
  data: appointmentQR,
  size: 150,
  backgroundColor: Colors.white,
) // QImage

```

Figure 5.3.19 Displaying QR

Then, appointment data is concatenate into a string that will be used as the data for the generated QR code as shown in Figure 5.2.18. The QR code will then be displayed as shown in Figure 5.2.19 by setting the data for the QR, the size of the QR image along with the background color for the generated QR code.

5.3.7 Appointment Reminders

Set appointment reminder

```

if (dateSelected == null || timeSelected == null) {
  final snackBar = SnackBar(
    content: const Text(
      "Please select a date and time",
      textAlign: TextAlign.center,
      style: TextStyle(fontSize: 18.0),
    ), // Text
    behavior: SnackBarBehavior.floating,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(24)), // RoundedR
    backgroundColor: Colors.red,
  ); // SnackBar
  ScaffoldMessenger.of(context).showSnackBar(snackBar);
} else {
  final String dateTimeString =
    dateSelected + " " + timeSelected;
  final DateFormat format = DateFormat("yyyy-MM-dd hh:mm a");
  // if (kDebugMode) {
  //   print(format.parse(dateTimeString));
  // }
  DateTime reminderTime = format.parse(dateTimeString);
  final difference =
    reminderTime.difference(DateTime.now()).inMilliseconds;
  scheduleAlarm(difference);

  addReminder(user.uid, difference.toString());
}

```

Figure 5.3.20 Set appointment reminder (1)

If patient did not select a valid date and time for the reminder, a snack bar will be pop out to notify user to select a date and time as shown in Figure 5.2.20. If patient have selected a valid date and time for the reminder, the system will run the codes as shown below. In Figure 5.2.21, the code will schedule a reminder by calling the `flutterLocalNotificationsPlugin` method along with its parameters, which are notification identifier, notification title, notification message, scheduled date to send notification and platform specific notifications details. Then, the code in Figure 5.2.22 will be called to add the reminder data to the Cloud Firestore by using the `add()` method add the Map containing the data.


```

void scheduleAlarm(int difference) async {

  var scheduleNotificationDateTime = DateTime.now().add(Duration(
    milliseconds: difference)); //time of notification trigger

  String notiMessage = "Appointment on " +
    widget.apptDate.toString() +
    " at " +
    widget.apptTime.toString();

  var androidPlatformChannelSpecifics = const AndroidNotificationDetails(
    'alarm_notif',
    'alarm_notif',

    icon: 'codex_logo',
    sound: RawResourceAndroidNotificationSound('a_long_cold_sting'),
    largeIcon: DrawableResourceAndroidBitmap('codex_logo'),
  ); // AndroidNotificationDetails

  var iOSPlatformChannelSpecifics = const IOSNotificationDetails(
    sound: 'a_long_cold_sting.wav',
    presentAlert: true,
    presentBadge: true,
    presentSound: true);

  var platformChannelSpecifics = NotificationDetails(
    android: androidPlatformChannelSpecifics,
    iOS: iOSPlatformChannelSpecifics);

  await flutterLocalNotificationsPlugin.schedule(
    difference,
    widget.apptName,
    notiMessage,
    scheduleNotificationDateTime,
    platformChannelSpecifics,
  );
}

```

Figure 5.3.21 Set appointment reminder (2)

```

addReminder(String uid, String diffID) {
  if (dateSelected.isNotEmpty) {
    Map<String, dynamic> reminderMap = {
      "appName": widget.apptName,
      "remindDate": dateSelected,
      "remindTime": timeSelected,
      "isChecked": isChecked,
      "reminderID": diffID
    };
    dbs.addReminders(uid, reminderMap);
    dateSelected = "";
  }
}

addReminders(uid, reminderMap){
  UserCollection.doc(uid).collection("Reminders").add(reminderMap)
  .catchError((e){
    print(e.toString());
  });
}

```

Figure 5.3.22 Set appointment reminder (3)

Delete appointment reminder

```

- GestureDetector(
  onTap: (){
    int remindID = int.parse(reminderID);
    cancelReminder(remindID);
    final snackBar = SnackBar(
      content: Container(
        height: 30.0,
        child: const Text(
          "Reminder Cancelled",
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 20.0),
        )), // Text, Container
      duration: const Duration(seconds: 1),
      behavior: SnackBarBehavior.floating,
      shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
      backgroundColor: Colors.red,
    ); // SnackBar
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  },

  child: const Icon(
    Icons.delete,
    color: Color(0xff1E4465),
    size: 30.0,
  ), // Icon
), // GestureDetector

void cancelReminder(int remindID) async {
  //cancel reminder based on id
  await flutterLocalNotificationsPlugin.cancel(remindID);
}

```

Figure 5.3.23 Delete appointment reminder

After pressing the delete icon, it will automatically cancel the reminder so it will not notify the patient by calling the `flutterLocalNotificationPlugin.cancel()` method using its unique notification identifier as shown in the figure above. After successfully cancelling the appointment reminder, a snack bar will pop to notify that the reminder has been cancelled.

5.4.8 User Profile

Edit profile

```
Expanded(
  child: TextFormField(
    decoration: InputDecoration(
      hintText: text2,
      hintStyle: TextStyle(color: Colors.grey[600]),
      border: OutlineInputBorder(
        borderSide: BorderSide.none,
        borderRadius: BorderRadius.circular(20),
      ), // OutlineInputBorder
    ), // InputDecoration
    style: TextStyle(fontSize: 16.0),
    controller: selectedItem(index, text2),
  ), // TextFormField
) // Expanded
```

Figure 5.3.24 Edit profile (1)

```
TextEditingController selectedItem(int index, String initial) {
  switch (index) {
    case 0:
      nameController.text = initial;
      return nameController;
      break;
    case 1:
      emailController.text = initial;
      return emailController;
      break;
    case 2:
      phoneController.text = initial;
      return phoneController;
      break;
    case 3:
      genderController.text = initial;
      return genderController;
      break;
    case 4:
      icController.text = initial;
      return icController;
      break;
  }
}
```

Figure 5.3.25 Edit profile (2)

Patient can select one of their profile details to prompt the keyboard as shown in Figure 5.2.24. After entering a new value, it will update the controller of that specific details as shown in Figure 5.2.25. If there is nothing else to update, patient can press the “Done” button to proceed to update their details as shown in Figure 5.2.26. The data will then be updated in the Cloud Firestore using the set() method. After successfully updating the patient profile, a snack bar will be displayed to notify the user.

```

GestureDetector(
  onTap: () {
    DatabaseService(uid: user.uid).updateProfile(
      nameController.text,
      emailController.text,
      phoneController.text,
      genderController.text,
      icController.text);
    final snackBar = SnackBar(
      content: Container(
        height: 30.0,
        child: const Text(
          "Updated Profile",
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 20.0
          ), // TextStyle
        ), // Text
      ), // Container
      behavior: SnackBarBehavior.floating,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(24)), // Rounded
      backgroundColor: Colors.lightBlue,
    ); // SnackBar
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  },

Future updateProfile(String name, String email, String phone,
  String gender, String ic) async{
  return await UserCollection.doc(uid).set({
    'name' : name,
    'email': email,
    'phone':phone,
    'gender':gender,
    'ic': ic
  });
}
}

```

Figure 5.3.26 Edit profile (3)

Appointment History

```

body: StreamBuilder(
  stream: FirebaseFirestore.instance
    .collection("Users")
    .doc(user.uid)
    .collection("AppointmentHistory")
    .snapshots(),

```

Figure 5.3.27 Retrieve appointment history data

Figure above illustrates the coding to retrieve the data from “Appointment History” collection in Cloud Firestore and stored in QuerySnapshots. Then, system will use the data stored in the QuerySnapshots to display the appointment history details in the appointment history page.

5.4 Coding Explanation (Admin Web App)

A new Flutter project is created in Visual Studio Code to start the project development for the admin web application. Then, the project will also be connected to Firebase to use its features. Then, various package dependencies are also added to use its Dart libraries and features.

5.4.1 Sign in

```

signIn(double widths, double heights) async {
  if (formKey.currentState.validate()) {
    setState(() {
      isLoading = true;
    });

    dynamic result = await authMethods.signInWithEmailAndPassword(
      emailController.text,
      pwController.text);

    if (result == null) {
      setState(() {
        isLoading = false;
        final snackBar = SnackBar(
          content: const Text("INVALID CREDENTIALS"),
          behavior: SnackBarBehavior.floating,
          width: widths/2,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(24)
          ), // RoundedRectangleBorder
          backgroundColor: Colors.red,
        ); // SnackBar
        ScaffoldMessenger.of(context).showSnackBar(snackBar);
      });
    }
    if(result!=null){
      Navigator.pushReplacement(context, MaterialPageRoute(
        builder: (context) =>
          homepage()
      )); // MaterialPageRoute
    }
  }
}

```

Figure 5.4.1 Sign In (1)

```

Future signInWithEmailAndPassword(String email, String password) async {
  try {
    UserCredential result = await _auth.signInWithEmailAndPassword(
      email: email, password: password);
    User firebaseUser = result.user;
    return _userFromFirebaseUser(firebaseUser);
  } catch (e) {
    if (kDebugMode) {
      print(e.toString());
    }
  }
}

```

Figure 5.4.2 Sign In (2)

Figure 5.4.1 above shows the code of the sign in function. If empty or invalid email and password entered, the system will show error message and request admin to re-enter a valid email address and password. After valid credentials entered and the sign in button is pressed, the sign in function will be invoked. Then, the Firebase Auth `signInWithEmailAndPassword()` as shown in Figure 5.4.2 will authenticate the admin credentials. If authentication fails, then a snack bar will pop displaying the error message. If authentication successful, then the system will redirect admin to homepage of their respective role.

5.4.2 Add Appointments

```

bookAppointment(String patientName, String apptName, String date) {
    if (apptDate.isNotEmpty) {
        Map<String, dynamic> appointmentMap = {
            "docName": widget.docName,
            "docRole": widget.docRole,
            "apptName": apptName,
            "apptDate": apptDate,
            "apptTime": timeSelected
        };
        dbs.addAppointment(patientName, appointmentMap);
        dbs.addTime(widget.docID, dateId, timeSelected);
        apptDate = "";
        dateId = "";
        choices.clear();
    }
}

PatientCollection.doc(patientInfo.docs[0].id.toString())
    .collection("Appointments")
    .add(appointmentMap)
    .catchError((e) {
        print(e.toString());
    });
}

Future addTime(String docID, String date, String apptTime) async {
    var list = [apptTime];
    DocCollection.doc(docID)
        .collection("date")
        .doc(date)
        .update({"timeBusy": FieldValue.arrayUnion(list)});
}

```

Figure 5.4.3 Add appointment (1)

```

TextButton(
  onPressed: () {
    apptDate = DateFormat('EEE, dd MMMM yyyy')
      .format(dateTime)
      .toString();
    bookAppointment(nameController.text, reasonController.text, apptDate);
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          content: Stack(
            clipBehavior: Clip.none,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.max,
                crossAxisAlignment: CrossAxisAlignment.spaceBetween,
                children: [
                  const Text("Added Successfully", style: TextStyle(color: textColor)),
                  InkResponse(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) =>
                            homepage()); // MaterialPageRoute
                    },
                    child: const Icon(Icons.close),

```

Figure 5.4.4 Add appointment (2)

Doctor will first input the appointment details such as patient name, visitation reason, appointment date and time. After that, they can press the submit button, which will then run the code as shown in Figure 5.4.3. The appointment details will be stored in a Map, then added to the “Appointment” collection using the add() method. Besides that, the appointment time will be added to the doctors “timeBusy” array list based on the appointment date.

5.4.3 Appointment List

```

AxisSize(
  width: screenSize.width / 2,
  child: DataTable(showCheckboxColumn: false, columns: <DataColumn>[
    aptColumn("Patient ID"),
    aptColumn("Patient Name"),
  ], rows: patientMap.map( // <DataColumn>[]
    ((element) => DataRow(
      selected: isSelected,
      onSelectChanged: (val) {
        setState(() {
          isSelected = val;
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) =>
                patientAppointment(
                  element["patientName"],
                  element["patientID"])); // patientAppoint
            ));
        });
      },
      cells: <DataCell>[
        DataCell(aptRow(element["patientID"])),
        DataCell(aptRow(element["patientName"])),
      ], // <DataCell>[]
    )), // DataRow
  ),
  .toList(),

```

Figure 5.4.5 Doctor's appointment list

The code above illustrates how the DataTable stores data into the row and column. Patient's details of the doctor will be retrieved from the Cloud Firestore and stored in the Map, which will then be used to fill the data cells. Then, doctor can select a row to navigate to the doctor's patient appointment page.

```

Future getApptInfo() async {
  if (widget.patientName.toString().isNotEmpty) {
    await db.getPatientByName(widget.patientName.toString()).then((val) {
      setState(() {
        patientInfo = val;
      });
    });
    await FirebaseFirestore.instance
      .collection("Users").doc(patientInfo.docs[0].id)
      .collection("Appointments").get().then((val) {
        setState(() {
          apptInfo = val;
        });
      });
    await FirebaseFirestore.instance
      .collection("Users").doc(patientInfo.docs[0].id)
      .collection("AppointmentHistory").get().then((val) {
        setState(() {
          historyInfo = val;
        });
      });
  }
}

if (apptInfo.docs.isNotEmpty) {
  for (int i = 0; i <= apptInfo.docs.length; i++) {
    Map<String, String> patientMap2 = {
      "apptName": apptInfo.docs[i]["apptName"].toString(),
      "apptDate": apptInfo.docs[i]["apptDate"].toString(),
      "apptTime": apptInfo.docs[i]["apptTime"].toString(),
      "pid": apptInfo.docs[i].id.toString(),
      "docRemark": "-",
      "apptStat": "Pending",
    };
    apptMap.add(patientMap2);
  }
}

if (historyInfo.docs.isNotEmpty) {
  for (int i = 0; i < historyInfo.docs.length; i++) {
    Map<String, String> patientMap3 = {
      "apptName": historyInfo.docs[i]["apptName"].toString(),
      "apptDate": historyInfo.docs[i]["apptDate"].toString(),
      "apptTime": historyInfo.docs[i]["apptTime"].toString(),
      "pid": historyInfo.docs[i].id.toString(),
      "docRemark": historyInfo.docs[i]["docRemark"].toString(),
      "apptStat": "Completed",
    };
    apptMap.add(patientMap3); //add the map to the List
  }
}

```

Figure 5.4.6 Retrieve Patient Appointment Data

In the patient appointment page, the system will call the code as shown in Figure 5.2.6 in the initState() method. The function will first get the patient information from the Cloud Firestore by using the where() and the patient's name. Then, the system will retrieve the patient's appointment data from the "Appointment" and "AppointmentHistory" collection and stored in respective QuerySnapshots. Consequently, the patient's appointment information is then stored in the Map using add() to add the Map to the List. Following from that, the data stored in the Map is then displayed in the DataTable.


```

if (status == apptStat) {
  return Row(
    children: [
      SizedBox(
        width: 300,
        child: Text("Descriptions",
          textAlign: TextAlign.start, style: tableStyle), // Text
      ), // SizedBox
      Container(
        width: 500,
        height: 200,
        padding: const EdgeInsets.all(20),
        decoration:
          BoxDecoration(border: Border.all(color: Colors.black54)),
        child: TextField(
          controller: descController,
          decoration: InputDecoration.collapsed(
            hintText: 'Write here',
            hintStyle: GoogleFonts.roboto(color: textColor, fontSize: 18),
          ), // InputDecoration.collapsed
          keyboardType: TextInputType.multiline, //for multiline input
          maxLines: null,
          textAlign: TextAlign.start,
          style: valueStyle,
        ), // TextField
      ) // Container
    ],
  ); // Row
} else {
  return Row(
    children: [
      SizedBox(
        width: 300,
        child: Text("Descriptions",
          textAlign: TextAlign.start, style: tableStyle), // Text
      ), // SizedBox
      SizedBox(
        width: 300,
        child: Text(
          widget.docRemark,
          textAlign: TextAlign.start,
          style: valueStyle,
        ), // Text
      ) // SizedBox
    ]
  );
}

```

Figure 5.4.7 Patient appointments details

Then, in the patient appointment details page, the appointment data such as patient name and ID, appointment date and time will be passed as arguments and displayed. If the appointment status is “Pending”, there will be a TextField for doctor to enter any remarks or prescriptions after the appointment. However, if the appointment status is “Completed”, then the doctor description or remark from the appointment will be retrieved and displayed.

5.4.4 Chat Enquiries

```

future: dbs.getChatList(widget.adminID),
builder: (context, snapshot) {
  switch (snapshot.connectionState) {
    case ConnectionState.waiting:
      return Loading();
    default:
      if (snapshot.hasData && !snapshot.hasError) {
        return ListView.builder(
          itemCount: apptList.docs.length,
          itemBuilder: (context, index) {
            return GestureDetector(
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) =>
                      chat(widget.adminID, apptList.docs[index].id),
                  )); // MaterialPageRoute
              },
              child: Card(
                elevation: 10.0,
                color: white,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(15.0)), // RoundedR
                child: ListTile(
                  title: Text(apptList.docs[index].id),
                ), // ListTile
              ), // Card
            ); // GestureDetector
          },
        ); // ListView.builder
      }
    }
};

```

Figure 5.4.8 Chats Enquiry List

The code shown above will first retrieve chat data from the “Message” collection in Cloud Firestore. Then, a `ListView.builder` is created, with the data displayed in a `Card()`. Medical receptionist can select a `Card` to redirect to the chat page.

```

stream: FirebaseFirestore.instance
  .collection('messages')
  .doc("admin")
  .collection('patient1')
  .doc(widget.chatID)
  .collection("patient")
  .orderBy("createdAt")
  .snapshots(),
builder: (context, snapshot) {
  if (!snapshot.hasData) {
    return Loading();
  } else {
    List<DocumentSnapshot> items = snapshot.data.docs;
    var messages = items
      .map((i) => ChatMessage.fromJson(i.data()))
      .toList();
  }
};

```

Figure 5.4.9 Retrieve message data

```
void onSend(ChatMessage message) {  
    FirebaseFirestore.instance  
        .collection('messages')  
        .doc("admin")  
        .collection("patient1")  
        .doc(widget.chatID)  
        .collection("patient")  
        .doc(DateTime.now().millisecondsSinceEpoch.toString())  
        .set(message.toJson());  
}
```

Figure 5.4.10 Update message data

The Figure 5.4.9 shows the coding to retrieve the message data from Cloud Firestore and order it the time the message was created. Next, the data is then stored in List of DocumentSnapshot to later be displayed by the system. The code in Figure 5.4.10 shows the code to update message data. Once a message is sent, the code will be invoked to update the data in Cloud Firestore using the set() method along with the time the message was created and sent.

CHAPTER 6: SYSTEM TESTING

CHAPTER 6: SYSTEM TESTING

6.1 Unit Testing

Unit testing is a type of software testing to test the individual units or components in isolation to validate each unit perform as expected.

6.1.1 User and Admin Authentication

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Verify user and admin login with valid credentials	Email: jonathanjalleh@gmail.com Password: abcd1234	Successful login	As expected	Success
2	Verify user and admin login with invalid credentials	Email: jonathanJalleh@gmail.com Password: abcd1234	Unsuccessful login and error message displayed	As expected	Success
3	Verify user and admin login with invalid format of credentials	Email: jonathanjalleh (invalid email format) Password: abc (less than 6 characters)	Unsuccessful login	As expected	Success
4	Verify patient account creation with correct field format	Name: Jonathan Jalleh I/C: 00001-01-0001 Email: jonathanjalleh@gmail.com Address:	Successful account creation	As expected	Success

CHAPTER 6: SYSTEM TESTING

		Phone Number: 012-3456789 Password: jonathan123 Confirm Password: jonathan123			
5	Verify patient account creation with incorrect field format	Name: Jon (less than 4 char) I/C: 00001-01-0001 Email Address: jonathanjalle (invalid format) Phone Number: 012-3456789 Password: jona (less than 6 char) Confirm Password: jonathan (not same as password)	Unsuccessful account creation	As expected	Success

Table 6.1 User and admin authentication test cases

6.1.2 User Appointment Module

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Check for the response when patient to view the doctors in a category	Action: Enter cardiology category	System displays all doctors under that category	As expected	Success
2	Check for the response when patient want to book a new scheduled appointment with valid input fields	Doctor Selected: Suffian Date selected: 23 April 2022 Time selected: 9:15 AM Action: Press the book appointment button	System displays the successfully booked page	As expected	Success
3	Check for the response when patient want to book a new scheduled appointment with empty fields	Doctor Selected: Suffian Date selected: - Time selected: - Action: Press the book appointment button	System displays snack bar containing error message	As expected	Success

CHAPTER 6: SYSTEM TESTING

4	Check for the response when patient want to book a same day appointment with valid input fields	Doctor Selected: Suffian Time selected: 9:15 AM Action: Press the book appointment button	System displays the successfully booked page	As expected	Success
5	Check for the response when patient wants to view all appointment booked.	Action: Enter appointment list page	System displays all patient booked appointments	As expected	Success
6	Check for the response when patient does not have any appointment booked	Action: Enter appointment list page	System display “You have no appointments”	As expected	Success
7	Check response when patient want to reschedule appointment.	Action: slide to left an appointment and select update action. Then, select a new date and continue button.	System should update the appointment and show snack bar containing success message	As expected	Success

CHAPTER 6: SYSTEM TESTING

8	Check response when patient want to cancel appointment.	Action: slide to left an appointment and select cancel action. Then, press cancel button	System should remove the appointment and show snack bar containing success message	As expected	Success
9	Check for the response when patient want to view a specific appointment	Action: Press an appointment list	System should display appointment details along with generated appointment QR code	As expected	Success
10	Check for response when patient wants to view the appointment history	Action: Enter appointment history page	The system displays all patient appointment history.	As expected	Success
11	Check for response when patient want to view a specific appointment history.	Action: Press an appointment history list	The system displays the appointment history details.	As expected	Success

Table 6.2 Appointment module test cases

6.1.3 User Reminder Module

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Check for the response when patient set reminder with valid date and time.	Date selected: 23 April 2022 Time selected: 8:00 AM Action: Press done button	The system displays the reminder set.	As expected	Success
2	Check for the response when patient set reminder with empty date and time.	Date selected: - Time selected: - Action: Press done button	System displays snack bar containing error message.	As expected	Success
3	Check for response when patient want to view reminders set.	Action: Enter reminder list page	The system displays all the reminders set by patient.	As expected	Success
4	Check for the response when patient want to cancel reminder.	Action: Press the delete icon	The system displays snack bar containing success message.	As expected	Success
5	Check for the response when patient want to delete the reminder.	Action: Select the radio button. Press the remove button.	The system removes the reminder from the reminder list.	As expected	Success

Table 6.3 User reminder module test case

6.1.4 User Profile Module

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Check response when patient wants to view their profile	Action: Press the profile icon in the homepage	The system should display the patient details.	As expected	Success
2	Check for response when patient input new value to edit current fields.	Name: Jonathan Aaron Jalleh Phone: 012-23456789 Action: Press the Done button	The system should update the patients details with the new values.	As expected	Success

Table 6.4 User profile module test case

6.1.5 Admin Module (Doctor)

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Check response when doctor wants to view their profile	Action: Enter the profile page	The system should display the doctor details.	As expected	Success
2	Check response when doctor want to add available time	Date selected: 23 April 2022 Time selected: 12:01 PM	The system should add the available time to doctor's timeslot list.	As expected	Success
3	Check response when doctor want to add available time with empty fields	Date selected: - Time selected: -	The system should display error message and ask doctor to re-enter.	As expected	Success
4	Check response when doctor want to add appointment with valid inputs.	Patient Name: Jonathan Jalleh Visitation Reason: Follow up Appointment Date: 23 April 2022 Appointment Time: 12:00 PM	The system should add the appointment to patient lists of appointment.	As expected	Success
5	Check response when doctor wants to add	Patient Name: - Visitation Reason: - Appointment Date: -	The system should display error message.	As expected	Success

CHAPTER 6: SYSTEM TESTING

	appointments with empty fields.	Appointment Time: -			
6	Check response when doctor wants to view list of doctors	Action: Enter doctors list page	The system should display a list of doctors with brief details.	As expected	Success
7	Check response when doctor wants to view their patient list	Action: Enter patient list page	The system should display the patient list of the doctor.	As expected	Success
8	Check response when doctor wants to view the appointment list of a patient	Action: Enter patient appointment list page	The system should display the appointment list of the selected patient.	As expected	Success
9	Check response when doctor wants to view the appointment details of a patient	Action: Enter patient appointment details page	The system should display the appointment details of the selected patient.	As expected	Success

Table 6.5 Admin doctor module test case

6.1.6 Admin Module (Medical Receptionist)

Test Case	Test Case Description	Test Data	Expected Output	Actual Output	Success/Fail
1	Check response when medical receptionist wants to view their profile	Action: Enter the profile page	The system should display the medical receptionist details.	As expected	Success
2	Check response when medical receptionist wants to add a new patient	Patient Name: Jonathan Aaron I/C number: 000002-02-02 Age: 22 Phone: 012-22222222 Email Address: jonathan.aaron@gmail.com Gender: Male Address: 123, Jalan ABC, 4, Taman ABC	The system should add the patient details to patient list	As expected	Success
3	Check response when medical receptionist wants to add a new patient with invalid credentials	Patient Name: aaro I/C number: 000000 Age: - Phone: - Email Address: jonathan.aaron Gender: -	The system should display error message.	As expected	Success

CHAPTER 6: SYSTEM TESTING

		Address: -			
4	Check response when medical receptionist wants to view all list of patients	Action: Enter patient list page	The system should display list of patients.	As expected	Success
5	Check response when medical receptionist wants to view a patient's details	Action: Enter patient details page	The system should display the details of the specific patients	As expected	Success
6	Check response when medical receptionist wants to view the chat inquiries list.	Action: Enter chat inquiries list page	The system should display list of chat inquiries ID.	As expected	Success
7	Check response when medical receptionist wants to view patient enquiry	Action: Enter chat inquiries page	The system should display the chat inquiries of a patient.	As expected	Success

Table 6.5 Admin medical receptionist module test case

CHAPTER 7: CONCLUSION AND RECOMMENDATION

7.1 Project Review and Discussion

In summary, the current hospital appointment system in Malaysia is tedious and time-consuming owing to various physical process required to make an appointment. Besides that, it also causes long waiting time due to multiple factors such as long waiting line and long doctor consultation time. Patient no-shows is also a problem as it causes longer direct and indirect waiting times, which results from patient forgetfulness and difficulty in finding parking spot. This proposed project aims to develop a mobile application for hospital appointment system which makes scheduling medical appointment simple and dependable for patients. It is significant as it will reduce patient waiting time and increase patient satisfaction.

In this project, a front-end mobile application for patient to make appointments has been developed. The mobile application can be used to schedule appointments by selecting a date and time, then saving it to patient's list of appointments. Patient will be able to view doctor's available time slot to prevent them from picking slots with many people, reducing their waiting time. The customizable appointment reminder feature has been implemented too, so patient can schedule a reminder for each appointment at their own preference to prevent patient no-shows from occurring. The appointment can also be cancelled or remove after notifying patients. Moreover, for each appointment made by patient, a unique QR code will be generated containing the appointment information. This QR code can be used to check in during appointment day, preventing the need for manual queueing.

A backend web application for hospital admin for doctors and medical receptionist is developed. The web app is used by doctors to manage their respective patient's appointment such as set up any follow up appointment and update patient appointment details. Moreover, doctors can use it to update their schedule and timeslot such as viewing their time available and busy for a specific date. Furthermore, doctor can also add any new available timeslot which will be displayed in the patient mobile application. On the other hand, medical receptionist can use the application to view patient list and details, along with adding any new patient that may walk-in. In addition, there medical receptionist can reply to any patient enquiries through the chat feature.

7.2 Future Recommendation

There are still few enhancements that can be added to the hospital appointment system so that it can be improved further in the future. The first recommendation is addition of online payment such as credit card, online banking and e-wallet. This is because the system now requires patient to physically make payment at the hospital after the appointment. Nonetheless, it surely will be more convenient if patient can directly pay for the appointment via the mobile application as it has a faster transaction speed and can be done anywhere. Therefore, the future recommendation for the application is the integration of an online payment feature.

On the other hand, another enhancement that can be added is AI chatbot for the patient inquiries. This can be more effective and efficient as it can automatically reply to the patient in a short time and is online all the time. Compared to AI chatbot, medical receptionist may overlook a patient inquiry or may take a longer time to reply, this may be risky in an emergency situation when patient required immediate reply. Hence, for future recommendation, an AI chatbot feature to reply to patient enquiries automatically can be added.

In addition, a feedback and rating system can be added to further boost the credibility of the hospital appointment system. After an appointment is completed, patient should be able to give rating and write feedback on the appointed doctor. This will increase the credibility of the doctor and gives patient an increase confidence when booking a hospital appointment with that respective doctor. Moreover, it is a great way for patient interaction and influence patient decisions.

REFERENCES

REFERENCES

- [1] BA. Ahmad, K. Khairatul, and A. Farnaza, "An Assessment of Patient Waiting and Consultation Time in a Primary Healthcare Clinic," *Malaysian Family Physician*, vol. 12, no. 1, pp. 14–15, Apr. 2017.
- [2] J. Akinode and S. Oloruntoba, "Design and implementation of a patient appointment and scheduling system international advanced," *Research Journal in Science, Engineering and Technology*, vol. 4, no. 12, Dec. 2017, doi: 10.17148/IARJSET.2017.41203.
- [3] Y. Symey, S. Sankaranarayanan, and SN. Sait, "Application of Smart Technologies for Mobile Patient Appointment System," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 2, no. 9, pp. 74–75, Aug. 2013.
- [4] S. Sumaira and A. Shafquat, "Factors associated patient waiting time at outpatient department in allied hospital Faisalabad," *Journal of Biology, Agriculture and healthcare*, vol. 17, no. 7, pp. 14–20, 2017.
- [5] M. Petek Šter, I. Švab, and G. Živčec Kalan, "Factors related to consultation time: Experience in Slovenia," *Scandinavian Journal of Primary Health Care*, vol. 26, no. 1, pp. 29–34, Jan. 2008, doi: 10.1080/02813430701760789.
- [6] W. Feuer, "Doctors worry the coronavirus is keeping patients away from US hospitals as ER visits drop: 'Heart attacks don't stop,'" *CNBC*, Apr. 14, 2020. <https://www.cnbc.com/2020/04/14/doctors-worry-the-coronavirus-is-keeping-patients-away-from-us-hospitals-as-er-visits-drop-heart-attacks-dont-stop.html#:~:text=Health%20and%20Science-> (accessed Apr. 1, 2022).
- [7] "Patient No-Shows: Everything Practice Managers Need to Know," *Relatient*. <https://www2.relatient.net/resources/patient-no-shows/#:~:text=A%20patient%20no%2Dshow%20refers> (accessed Apr. 01, 2022).

REFERENCES

- [8] D. Gupta and B. Denton, "Appointment scheduling in health care: Challenges and opportunities," *IIE Transactions*, vol. 40, no. 9, pp. 800–819, Jul. 2008, doi: 10.1080/07408170802165880.
- [9] S. A. AlSadhan, "Frequency of missed and cancelled appointments in King Saud University orthodontic clinic," *King Saud University Journal of Dental Sciences*, vol. 4, no. 2, pp. 77–80, Jul. 2013, doi: 10.1016/j.ksujds.2013.04.001.
- [10] K. McCoy, "Drivers spend an average of 17 hours a year searching for parking spots," *USA TODAY*, Jul. 12, 2017. <https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/>
- [11] L. W. Robinson and R. R. Chen, "A Comparison of Traditional and Open-Access Policies for Appointment Scheduling," *Manufacturing & Service Operations Management*, vol. 12, no. 2, pp. 330–346, Apr. 2010, doi: 10.1287/msom.1090.0270.
- [12] S. Lee, D. Min, J. Ryu, and Y. Yih, "A simulation study of appointment scheduling in outpatient clinics: Open access and overbooking," *SIMULATION*, vol. 89, no. 12, pp. 1459–1473, Oct. 2013, doi: 10.1177/0037549713505332.
- [13] "Make An Appointment," *Pantai.com.my*, 2022. <https://www.pantai.com.my/kuala-lumpur/make-appointment> (accessed Apr. 1, 2022).
- [14] W. Cao *et al.*, "A web-based appointment system to reduce waiting for outpatients: A retrospective study," *BMC Health Services Research*, vol. 11, no. 1, Nov. 2011, doi: 10.1186/1472-6963-11-318.
- [15] "Bru-HIMS," *Moh.gov.bn*, 2019. <http://www.moh.gov.bn/SitePages/Bru-HIMS.aspx>

REFERENCES

- [16] H. Karthikraj, V. Savitha, M. Pavithra, K M. Mohammed-Fayyaz, and K. Sangeetha, "Doctor Appointment System Using Cloud," *International Research Journal on Advanced Science Hub*, vol. 3, no. Special Issue ICARD 3S, pp. 13–17, Mar. 2021, doi: 10.47392/irjash.2021.053.
- [17] V. Jogalekar, R. Pal, A. Yadav, and A. Ingle, "Development of Online Hospital Booking Portal for Patient," *VIVA-Tech International Journal for Research and Innovation*, vol. 1, no. 4, pp. 1–6, 2021.
- [18] "Malaysia smartphone penetration 2019-2023," *Statista*. <https://www.statista.com/statistics/625418/smartphone-user-penetration-in-malaysia/>
- [19] "Encore Med - Advanced Healthcare Appointment & Queue System," *encoremed.io*. <https://encoremed.io/> (accessed Apr. 1, 2022).
- [20] S. Malik, N. Bibi, S. Khan, R. Sultana, and S. Abdul-Rauf, "Mr. Doc: A Doctor Appointment Application System," *International Journal of Computer Science and Information Security*, vol. 14, no. 12, pp. 452–460, Dec. 2016.
- [21] N. Ismail, S. Kasim, Y. Yah Jusoh, R. Hassan, and A. Alyani, "MEDICAL APPOINTMENT APPLICATION," *Acta Electronica Malaysia*, vol. 1, no. 2, pp. 05–09, Nov. 2017, doi: 10.26480/aem.02.2017.05.09.
- [22] A. Hylton and S. Sankaranarayanan, "Application of Intelligent Agents in Hospital Appointment Scheduling System," *International Journal of Computer Theory and Engineering*, vol. 4, no. 4, pp. 625–630, 2012, doi: 10.7763/ijcte.2012.v4.545.
- [23] S. Usharani, S. Prithivi, S. Sharmila, P. M. Bala, T. Ananth Kumar, and R. Rajmohan, "Mobile Application for Doctor Appointment Scheduling," *IEEE Xplore*, Jul. 01, 2021. https://ieeexplore.ieee.org/abstract/document/9526398?casa_token=P6WYN0hBDKYAAAAA:jtw9E356YZcH6bAn980b8In6VEddf14luwuJ7pcVxipcbeZ411PPn_YBVozqC6mG9Yg24oHqcKQ (accessed Mar. 22, 2022).

REFERENCES

- [24] M. Roomi, “5 Advantages and Disadvantages of Client Server Network | Drawbacks & Benefits of Client Server Network,” *www.hitechwhizz.com*, Nov. 12, 2020. <https://www.hitechwhizz.com/2020/11/5-advantages-and-disadvantages-drawbacks-benefits-of-client-server-network.html>
- [25] S. Sharma, “What is Agile Methodology in Mobile App Development,” *Medium*, Aug. 20, 2020. <https://medium.flutterdevs.com/what-is-agile-methodology-in-mobile-app-development-4fa83ed6ac09>
- [26] V. Patil, S. Panicker, and M. Kv, “Use of Agile Methodology for Mobile Applications,” *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)*, vol. 5, no. 10, pp. 73–77, 11AD.
- [27] “9 reasons to choose Agile Methodology for Mobile App Development | Packt Hub,” *Packt Hub*, Oct. 2018. <https://hub.packtpub.com/9-reasons-to-choose-agile-methodology-for-mobile-app-development/> (accessed Apr. 1, 2022).
- [28] S. Cherednichenko, “Top 7 Reasons to Choose Flutter for Your Cross-platform App Development Project,” *Mobindustry - App Development*, Jun. 16, 2020. <https://www.mobindustry.net/top-7-reasons-to-choose-flutter-for-your-cross-platform-app-development-project/>
- [29] A. Mustafeez, “What is Visual Studio Code?,” *Educative: Interactive Courses for Software Developers*. <https://www.educative.io/edpresso/what-is-visual-studio-code>
- [30] G. Thomas, “What is Flutter and Why You Should Learn it in 2020,” *freeCodeCamp.org*, Dec. 12, 2019. <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020> (accessed Apr. 17, 2022).

WEEKLY LOG

FINAL YEAR PROJECT WEEKLY REPORT*(Project II)*

Trimester, Year: Y3S3	Study week no.: 1, 2
Student Name & ID: JONATHAN AARON JALLEH 18ACB01377	
Supervisor: DR TONG DONG LING	
Project Title: HOSPITAL APPOINTMENT SYSTEM	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

I reviewed back what I had done in FYP1.

2. WORK TO BE DONE

- Sketch draft of admin web app interface design.
- Complete web app user interface.

3. PROBLEMS ENCOUNTERED

No problem encountered.

4. SELF EVALUATION OF THE PROGRESS

Slow progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 3, 4
Student Name & ID: JONATHAN AARON JALLEH 18ACB01377	
Supervisor: DR TONG DONG LING	
Project Title: HOSPITAL APPOINTMENT SYSTEM	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- I have completed the draft of admin web app interface design and proceeded to complete the web app user interface.
- I have also done the sign in and sign-up function for the web app.

2. WORK TO BE DONE

- Implement patient data management function.
- Implement doctor schedule and patient appointment function.
- Implement patient enquiries chat function.

3. PROBLEMS ENCOUNTERED

No problem encountered.

4. SELF EVALUATION OF THE PROGRESS

I am still on track.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 5, 6
Student Name & ID: JONATHAN AARON JALLEH 18ACB01377	
Supervisor: DR TONG DONG LING	
Project Title: HOSPITAL APPOINTMENT SYSTEM	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- I have completed the patient data management module (view and add patient)
- I have completed the doctor schedule (view, add, update timeslot) and patient appointment (View, add, update and delete patient appointment).

2. WORK TO BE DONE

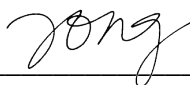
- Complete completed the patient enquiries list and chat page.
- Testing the overall system of the project (patient mobile app and admin web app)
- Check the overall performance of the system.

3. PROBLEMS ENCOUNTERED

Some error with the data management from Cloud Firestore.

4. SELF EVALUATION OF THE PROGRESS

I should improve my progress and manage my time more efficiently. However, I am able to solve the problem encountered successfully.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 7, 8
Student Name & ID: JONATHAN AARON JALLEH 18ACB01377	
Supervisor: DR TONG DONG LING	
Project Title: HOSPITAL APPOINTMENT SYSTEM	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete the patient chat enquiries module.
- Testing for the project and detecting bugs has been done.

2. WORK TO BE DONE

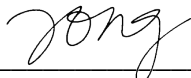
- Solve and fix the bugs encountered in the system.
- Adjust the user interface for better design and user experience.

3. PROBLEMS ENCOUNTERED

There was problem with irresponsive interface design and some layout issue.

4. SELF EVALUATION OF THE PROGRESS

I am on track and plan to proceed to report writing.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 9, 10
Student Name & ID: JONATHAN AARON JALLEH 18ACB01377	
Supervisor: DR TONG DONG LING	
Project Title: HOSPITAL APPOINTMENT SYSTEM	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Fixed majority of the bugs detected.
- Adjusted user interface.

2. WORK TO BE DONE

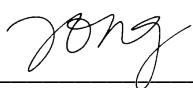
- Complete the FYP2 report from chapter 1 to 7.
- Design and create the poster
- Compile and check format for report.
- Prepare PowerPoint slides for presentation.

3. PROBLEMS ENCOUNTERED

No problem encountered

4. SELF EVALUATION OF THE PROGRESS

Planning to complete everything before due date.



Supervisor's signature



Student's signature

POSTER

Hospital Appointment System

The proposed mobile application will make scheduling hospital appointment simple and reliable by reducing patient waiting time and increasing patient satisfaction.

Project by: Jonathan Aaron Jalleh • Project Supervisor: Dr. Tong Dong Ling

Problem Statements

- Patient registration and appointment scheduling is time consuming
- Long waiting time
- Patient no-shows



Objectives

- To develop a mobile application for hospital appointment system.
- To benefit patients by reducing their waiting time, increasing patient satisfaction and allowing efficient management of health.

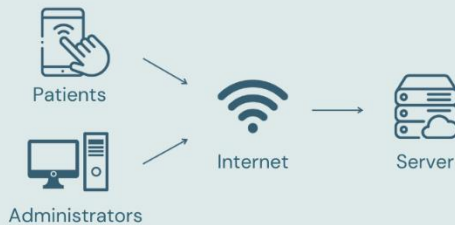


Project Scope

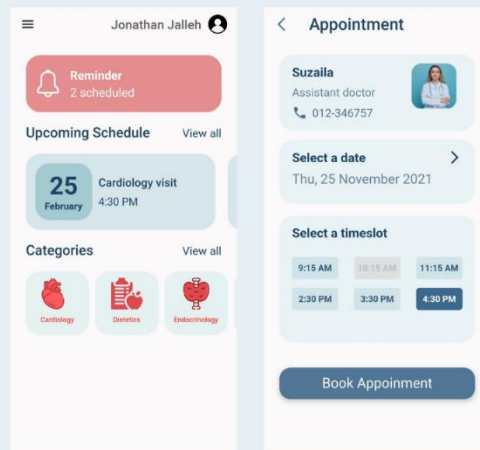
- QR code check in
- Current patient queue
- Appointment reminder feature
- Chat feature



System Framework



Results



Home Page

Appointment Page

Methodology



Conclusion

The proposed mobile application will make scheduling hospital appointment simple and reliable with features like customizable appointment reminder, QR code check in, real time queue and chat feature.



Faculty of Information and Communication Technology • Bachelor of Computer Science (Honours)

PLAGIARISM CHECK RESULT

PLAGIARISM CHECK RESULT

FYP2_1801377			
ORIGINALITY REPORT			
6%	4%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	eprints.utar.edu.my Internet Source		3%
2	www.coursehero.com Internet Source		1%
3	Submitted to Universiti Tunku Abdul Rahman Student Paper		<1%
4	Wenjun Cao, Yi Wan, Haibo Tu, Fujun Shang, Danhong Liu, Zhijun Tan, Caihong Sun, Qing Ye, Yongyong Xu. "A web-based appointment system to reduce waiting for outpatients: A retrospective study", BMC Health Services Research, 2011 Publication		<1%
5	Submitted to Our Lady of Fatima University Student Paper		<1%
6	Suren Machiraju, Suraj Gaurav. "BizTalk", Walter de Gruyter GmbH, 2018 Publication		<1%
7	Submitted to Universiti Teknologi MARA Student Paper		<1%

PLAGIARISM CHECK RESULT

8	Submitted to Westminster International College - Kuala Lumpur Student Paper	<1 %
9	ir.unimas.my Internet Source	<1 %
10	Maria Van Zyl-Cillié, Derya Demirtas, Erwin Hans. "Wait!What does that mean?: Eliminating ambiguity of delays in healthcare from an OR/MS perspective", Health Systems, 2022 Publication	<1 %
11	Submitted to Informatics Education Limited Student Paper	<1 %
12	30secondsofcode.org Internet Source	<1 %
13	Submitted to INTI International College Subang Student Paper	<1 %
14	Submitted to University of Bahrain Student Paper	<1 %
15	dev.lucee.org Internet Source	<1 %
16	"Chapter 5 Topological and Monotonicity Methods", Springer Science and Business Media LLC, 2007 Publication	<1 %

PLAGIARISM CHECK RESULT

17	Jie Zhou, Jun Li, Peng Guo, Xuemei Lin. "The booking problem of a diagnostic resource with multiple patient classes and emergency interruptions", Computers & Industrial Engineering, 2017 Publication	<1 %
18	rucforsk.ruc.dk Internet Source	<1 %
19	roserose-iloveu.blogspot.com Internet Source	<1 %
20	Diwakar Gupta, Brian Denton. "Appointment scheduling in health care: Challenges and opportunities", IIE Transactions, 2008 Publication	<1 %
21	P. Anantha Prabha, N. Arjun, J. Gogul, S. Divya Prasanth. "Chapter 36 Two-Way Economical Smart Device Control and Power Consumption Prediction System", Springer Science and Business Media LLC, 2022 Publication	<1 %

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Jonathan Aaron Jalleh
ID Number(s)	18ACB01377
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	Hospital Appointment System

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)
Overall similarity index: <u>6</u> % Similarity by source Internet Sources: <u>4</u> % Publications: <u>1</u> % Student Papers: <u>3</u> %	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required, and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.



 Signature of Supervisor

 Signature of Co-Supervisor

Name: Dr Tong Dong Ling

Name: _____

Date: 20 Apr 2022

Date: _____

FYP2 CHECKLIST



UNIVERSITI TUNKU ABDUL RAHMAN


**FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB1377
Student Name	Jonathan Aaron Jalleh
Supervisor Name	Dr Tong Dong Ling

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
✓	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;">  </div> <p>(Signature of Student) Date: 19 April 2022</p>
--