A PATIENT MANAGEMENT SYSTEM FOR PAEDIATRIC CLINIC USING FLUTTER

BY

LAI CHIN SI

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

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: A patient management system for paediatric clinic using Flutter

Academic Session: Jan 2022

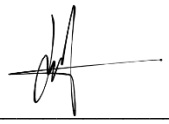
I LAI CHIN SI

(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.



(Author's signature)

Address:

10, Jalan Kebudayaan 30

Taman Universiti

81300 Skudai, Johor

Date: 21 April 2022

Verified by,

(Supervisor's signature)

Dr. Aun Yichiet

Supervisor's name

Date: 22 April 2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: __21 April 2022_____

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that _____***Lai Chin Si***_____ (ID No: __***18ACB04057***__) has completed this final year project entitled "_____*A patient management system for paediatric clinic using Flutter*_____" under the supervision of ___Dr. Aun Yichiet_____ (Supervisor) from the Department of _Computer and Communication Technology_____, Faculty of _Information and Communication Technology_____ .

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

Yours truly,

_____

LAI CHIN SI

DECLARATION OF ORIGINALITY

I declare that this report entitled "**A PATIENT MANAGEMENT SYSTEM FOR PAEDIATRIC CLINIC USING FLUTTER**" 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 : Lai Chin Si

Date : 21 April 2022

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor, Dr. Aun Yichiet and my moderator, Ts Lai Siew Cheng for the opportunity in involving in the study of web application development together with the web services. Along with the project, they have provided priceless guidance and assistance whenever I have any difficulties in the project. With their assistance, I am managed to deliver the final product along with the report within the schedule. Again, a million thanks to my supervisor and moderator.

Apart from them, I would also like to express my gratitude to the project teammate, Ong Zi Leong. Although we are having project scope, we have spent quite abundant amount of time together to discuss the capability and implementation of the project. Whenever I encountered any difficulties, he is willing to provide his help.

ABSTRACT

Recently, the medical field has received increasing concern from the public due to the emergence of various diseases. With respect to this circumstance, a well-functioning patient management system preserves a vital role in improving the clinic efficiency and ensuring smooth operation flow. This project introduces a widely applicable patient management system on various devices, by implementing the Flutter framework for the UI design and Firebase Cloud Services for variety of functionality including authentication and data storage. Spring Boot framework will be integrated as the main framework of the back-end application. The communication between the Flutter and Spring Boot application is performed using exposed APIs, and both of the applications will be encapsulated using Docker technology.

TABLE OF CONTENTS

LIST OF FIGURES

TITLE **PAGES**

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
Appt.	Appointment
DAO	Data Access Object
EHR	Electronic Health Record
EMR	Electronic Medical Record
etc.	Et cetera
GUI	Graphical User Interface
JDK	Java Development Kit
MVVM	Model-View-ViewModel
OS	Operating System
PC	Personal Computer
Q&A	Questions and Answers
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
VM	Virtual Machine

Chapter 1 Introduction

1.1 Problem Statement and Motivation

In recent years, the healthcare domain has been allocated much more attention from the public due to the critical disease, COVID-19. The emergence of this disease has brought heavy influences to the entire society, as the disease is highly contagious and may potentially leave irreversible damage to the human body. Public awareness of the importance of the healthcare system has been enhanced. More and more people start putting healthcare services as their first priority, leading to rapid growth in the medical field. To put it bluntly, the COVID-19 pandemic has acted as a stimulus to the growth as there are a large number of medical researches have been initiated and supported by the capital in order to eliminate the virus and cure the disease. In addition to the researches, the number of newly built field hospitals (eg. Huoshenshan Hospital and Leishenshan Hospital in China) and clinics are also increased to quarantine and treat the patients. In this case, a proper patient management system reserved a crucial role in managing and monitoring the patients so that they are under-controlled and managed to cooperate in the treatment provided by the hospital.

The requirement in the patient management system is inevitable along with the growth in the medical field. However, as the services provided and operation workflow are varied according to the hospital, it would be difficult for a patient management system to fulfil all the requirements of the hospital. For instance, a large-scale hospital might require an interface to manage its inventory, while some hospitals allow the online appointment service to their patients but some don’t. This means that each of the hospitals would most likely have its own tailored patient management system with different integrated features. Despite the requirement in a system that supports basic operations, the capability of the system to be expanded for new features in a short period has also be considered. In case of sudden unrevealed disease, the scalability of the system is significant whereby the short development and deployed time of the new features into the existing system facilitate in resolving society depression since the public could obtain the latest information about the disease at the first place while the other services including the appointment for vaccination would be available for later access.

The same scenario also applies to the paediatric clinic. Although the scale of the paediatric clinic is smaller than the hospital, it may have more specific features required in its patient management system as the paediatricians might trace the children's conditions from time to time. However, the paediatric clinic is lacking in a patient management system which could sufficiently support the daily operation while providing scalability characteristic in case of new features integration. With the scalability characteristic provided in the new system, the clinics are managed to adopt any essential features into the existing system in the future without any constraint. The clinics are also not required to abandon their previous system since the integration of the new features into the existing system is tolerated. With this, the clinics would be likely to reduce their extra spending on the new system and the difficulties encountered in familiarizing with the new patient management system.

1.2 Objectives

- To develop a web application for patient record management and appointment scheduling for a paediatric clinic using Flutter.
    - The project involves the development of an online web application with a variety of functionality, particularly the appointment and patient record administration, in order to optimize the operation flows of a paediatric clinic. It integrates Flutter as its front-end framework to enhance its cross-platform capability and the UI consistency of the application with a single codebase rather than heavily relying on the platform-specific UI components.

- To build and connect real-world patient database using Google Firebase that connects to the Web UI.
    - Firebase cloud services like the Firebase Authentication is used to handle the application logic.

1.3 Project Scope

In terms of project scope, the project develops an online web application that involves the development of several modules that support the patient management of a paediatric clinic. The modules including patient records, inventory management and online appointment are developed according to the requirements of the client clinic, and the requirement changes are taken into account in the development process of the project. The project will integrate several modern frameworks and cloud services, mainly Flutter for the user interface design and Firebase cloud services for the data and application logic management. Meanwhile, the flexibility of the application will be maintained, enhancing its future development potential into a complex clinic management system that applicable to various scale of clinic or hospital.

1.4 Contributions

The project is intended to contribute in maintaining the operations flow of various clinic by providing simple and neat user interface and well-maintained patient records. While adopting the system developed, the patients will be managed to schedule their appointments remotely, according to their preferred timeslot. This functionality will significantly reduce the workload of the clinic receptionist, with increasing importance when it comes to contagious diseases which requires least physical contacts and exposures.

On the other hand, with the extreme scalability of the web application, it can be duplicated and integrated with additional functionality according to requirement of each clinic. As the adoption of digital records becomes the current trend, the clinics may migrate their patient records from paper documentation to digital records, allowing the clinics to improve their efficiency in case of retrieving, updating and maintaining the patient records.

Meanwhile, the flexibility of the application will be preserved through developing reusable object model and UI components, enabling code reuse in the future development of additional functionalities. Using the system design of the project, the integration of the new functionality should require less effort, thereby indirectly improve the scalability of the system to adapt the adoption unit with various scale. In the current phase, a prototype of the application's user interface design, as well as the basic authentication method, are being developed.

1.5 Report Organization

The report is composed of 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology and Approach, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion and Chapter 7 Conclusion and Recommendation.

The first chapter of the report is the introduction, whereby it provides the problem statement and motivation of the project, the objectives, the project scope, the major contributions of the project and also the report organization. Literature review, the second chapter of the report, not only covers the review of the technologies applied in the project, but also includes a review of the existing similar systems and applications so that the advantages and drawbacks of these systems can be analyzed and improved in the proposed project. Then, the chapter 3 system methodology and approach describe the overall system capability and architecture adopted through several relevant diagrams such as System Architecture Diagram, Use Case Diagram and Activity Diagram.

After the system methodology and approach, the report is followed by chapter 4 system design. In this chapter, the design of the system is illustrated graphically using system block diagram, system components specifications, circuits and components design and system components interaction operations. The next chapter, chapter 5 system implementation states how the system is implemented according to the design made previously. With the system implemented, the chapter 6 system evaluation and discussion evaluate the system implemented for both validation and verification. The last chapter, which is the chapter 7 conclusion and recommendation, makes a conclusion for the project and also some recommendations on the future work on the project.

Chapter 2 Literature Review

In order to study the current trend of practices and provide a suitable solution to the problem, some existing systems relevant to clinic management have been reviewed. All of the patient management systems reviewed are available on the Internet for purchasing and implementation by the clinics. These existing systems are composed of general features to maintain the basic workflow of the clinic while supporting several specific features for implementation on-demand. Reviewing these existing systems especially the integrated unique modules provide a better understanding of the potential solutions to the problem mentioned in Chapter 1.

2.1 Review of the Technologies

2.1.1 Hardware Platform

Model	Laptop – Y520-151KBN
CPU	Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
GPU	NVIDIA GeForce GTX 1050 2 GB
RAM	Kingston 12.0 GB
Hard Disk	1 TB
Manufacturer	Lenovo

Table 2.1 Hardware platform utilized

The system can be executed either on the cloud using cloud hosting services or locally in own server. The machine should have minimum 4GB of RAM to ensure the services hosted can handle multiple requests concurrently and provide respective responses within short timeframe. Table 2.1 shows the machine specification used to launch the system in system development and testing.

2.1.2 Firmware/OS

Operating System	Windows 10
Edition	Windows 10 Home Single Language
Version	21H2

Table 2.2 Operating system required by the system

Table 2.2 shows the specification of the operating system used to test the system during system testing. Due to the fact that the system will be composed into Docker Container, the user should not have heavy concern on the operating system required. Generally, Docker is kind of a virtualization technology, where it reduces the burden in the software deployment and distribution by packaging all the required dependencies into standardized format, which is recognized as Docker Container [1]. With the dependencies included, the new users do not need to install extra environment component or SDK in order to execute the intended system.

2.1.3 Database

Database	Cloud Firestore
Type	Document-oriented NoSQL cloud database

Table 2.3 Database specification of the system

The system adopts Cloud Firestore, which is one of the Google Cloud Services as the solution for the data storage. With Cloud Firestore, it eliminates the need in physical server to store the data, reducing the upfront investment cost, which is a major discourage factor for the digitalization of the small-to-medium enterprise, including the paediatric clinic [2]. Unlike relational database, Cloud Firestore has a NoSQL structure, whereby the data are stored in documents in the form of Map. In order to have a better organization and optimize queries efficiency, these documents are categorized into different collections. Each of the collection can have nested collections if they are interrelated with strong dependency. Cloud Firestore also enforces security rules and cooperate with Firebase Authentication so that the data are only accessible with authorized users.

2.1.4 Programming Language

Front-end language	Dart
Front-end framework	Flutter
Backend	Java
Back-end framework	Spring Boot

Table 2.4 Programming language used to develop project

As the system is separated into front-end and back-end application, different programming languages are applied in the development. For the front-end application, as the Flutter framework is adopted, its native language, Dart, is utilized to develop the UI of the application. Flutter offers cross-platform design consistency to the application. As Flutter utilize its rendering engine, the GUI of the application is not dependent on the user rendering tools, therefore guaranteeing the exact same design for different platform users. Apart from the single codebase, Flutter also offers various widgets which provide elegant design and animation to the application. The developers could also customize the widgets to produce the intended layout effortlessly. In terms of the development process, the Flutter provides a hot-reload feature that allows the developers to view the results immediately after each change on the code. The hot-reload functionality avoids the process of recompiling and loading after the changes on the code, reducing the development time.

On the other hand, Spring Boot is adopted as the framework of the back-end application. Thus, Java is used to develop the entire business logic. Through the Spring Boot MVC structure, the core services are encapsulated into various REST APIs, so that these APIs are accessible by the front-end application. Apart from the inversion of control and dependency injection, Spring Boot also reduces the efforts required in the environment configuration by providing autoconfiguration for necessary components, facilitating in constructing a production grade system.

As the system does not integrate traditional relational database, no raw query is constructed using SQL. Instead, the designated Firestore Java SDK is applied to formulate the intended queries.

2.1.5 Algorithm

No specific algorithm is researched as it is not included in the project scope and is not part of the crucial element in the system.

2.1.6 Summary of the Technologies Review

As a summary, the system is mainly developed using Flutter framework with Dart language and Spring Boot framework with Java language. The cloud services, Cloud Firestore and Firebase Authentication are adopted as crucial components of the system, whereby the Cloud Firestore is used to store data while the Firebase Authentication is responsible for securing the data stored. After the system development is done, it will be dockerized into Docker Container to reduce the difficulty level and effort required on the deployment and distribution. At the meantime, the system is loosely coupled to both the hardware and OS requirement since the Docker Container will package all the essential dependencies and be deployed on the cloud using external cloud hosting services.

2.2 Review of the Existing Systems and Applications

2.2.1 Patient Manager

Patient Manager is a clinic management software that allows the staff team to exchange medical information and perform their necessary interaction efficiently. The workflow of the medical team such as doctors and nurse, receptionist and financial members are integrated via the built-in features in the software. A high level of security is also provided with the software whereby different access privileges are distributed to the patient, receptionist, doctors and etc [3].

The Patient Manager software has a patient management module to support the organization and management of the clinic patients. For those clinics with inpatients and outpatients, the software has a ward module so that the medical team are managed to monitor the admission and discharge of the patients. Along with that, a calendar API is also adopted in the software to enable the ward scheduling feature so that the ward occupied status is monitored. In addition, the medical records of each patient are also be recorded in the medical record module. The detailed information of each medical consultation and treatment results could be recorded in the system for future references.

A medical billing module is also integrated with the software. Through this module, the invoices of the patient treatments could be generated to perform payments at the clinic counter. The invoices would be able to retrieve the information of the treatments received by the patients. The management team of the clinic would also be managed to generate a monthly or yearly financial report using the report feature in the billing module.

In terms of the security aspect, the users of the software are separated into different roles with various permissions. For instance, the medical assistants are allowed to create a new patient record but the employee who responsible for the cashier don’t. The access to the user interface and sections in the software are limited through their respective permission so that the unauthorized users would have no access to the sensitive sections. Besides that, the flexibility in changing the permission setting is provided whereby the permission of the user group could be updated when necessary. To ease the management of the user permission to the software, the new user is added to a

predefined role group so that all role owns the same privileges within their respective group and no repeated permission setting is required.

Despite the fact that multiple useful functions are provided in the software, it is only available on the Windows OS. There is no mobile version for this software, bringing inconvenience to the medical team since they are required to utilize the software through the PC. The incapability of the patient to access the software is another weakness that exists in the software. This software aimed at building interconnection between the medical team, therefore not allowing access from the normal users which are the patients. Hence, the patients could not be able to examine their personal treatment records and billing history unless they visit the clinic, which is inconvenient for them.

2.2.2 eClinic Systems

The eClinic system is a clinic management web application that supports features including patient management, online appointment and also eBilling feature [4]. With the implementation of the eClinic system, all of the clinic operations aspects such as medical, administrative and financial status could be monitored more intuitively.

Apart from the common features, there is an integrated alert system that pushed the notifications via email and text messages to the patients. With the reminder, the risk of appointments being forgotten or neglected is minimized, enhancing the efficiency of patient care. The software is also integrated with the email service, whereby the administrators are capable to send out emails to the patients who have subscribed to update them about periodic events such as vaccination of newborn or body checkup.

The eClinic system contains several modules that are associated with the treatment provided by the clinic. The laboratory management module fulfils the requirement of some clinics with analysis activities in their lab and required additional documentation and management. There is also a discharge summary module for the medical team to manage the admission and discharge operation of their patients.

The system has a relatively wide availability compared to the first system reviewed. The eClinic system is available in the version of a web application, Android and iOS application, reducing the constraints on the requirement of the device since the access to the system is not limited to the PC only. The Android and iOS application also enable the patients to receive the notifications and reminders from the clinic without the demand on the phone-call which consume more labour-power. (Adroit Infosystems n.d.)

2.2.3 Jane App

Jane App is a clinic management software developed for healthcare services providers including the paediatric clinic. It served as a multipurpose web platform with several integrated features for clinic essential operations. As the Jane App is developed as a web-based program, all of the users are allowed to access the system as long as they are connected to the Internet. With the implementation of the Jane App, the patients are equipped with a reminder service whereby email reminder would be received before the real appointment [5].

The online booking of the appointments is provided for the patients. The patients who registered would obtain permission to access the schedule of the practitioner, facilitating the patients to book their appointments according to their respective available timeslot. The scheduling of the appointments also avoids the clashing of the meeting time, decreasing the extra efforts required to perform manual scheduling daily after the clinic operation. The practitioners are managed to view their next-day appointments directly since it is organized and synchronized with the booking of the patients. With this, the information of the patients could be obtained and viewed prior to the appointment, allowing the practitioners to revise the condition of the patients and perform necessary preparations.

The Jane App has a rather unique function which is the availability of the uploading of images and videos. These uploads are allowed for the practitioners whereby they could upload any images relevant to the patient’s conditions including the X-ray images for future reference. The videos could also be appended to the patients' records to provide the patients with clearer documentation of their healthcare conditions since the explanation of the diagnosis results and precautions could be reviewed by the patients. The explanation in the video also served as the summary for the patients’ treatment so that the patients are not required to study the paper documentation which normally filled with statistics and jargons.

In terms of weaknesses, the Jane App is more inclined to the online appointments and scheduling features and the security aspect is neglected. As there is no role separation and permission setting, all of the clinic staffs are provided with unlimited access to the database. This may result in the exposure of private personal information whereby the

receptionists who are responsible for patients' simple queries and appointments arrangement are also allowed to view the detailed information of the patients.

2.2.4 Encore Clinic Software

ENCORE Clinic Software by Encore eServices (2014) is a clinic management system available on the Windows OS. The ENCORE Clinic Software is composed of multiple management modules including queue management, patient records management and appointment management modules. The availability of these modules supports the business workflow of the clinic, whereby the patient registrations and the billing process are facilitated by the software.

A queue management module is developed in this software. There are 3 distinct queues, which are 'Wait Queue', 'Dispensary Queue' and 'Payment Queue' available for the different operations of the patients. With the wait queue, the doctors are managed to admit the patients who are waiting according to their arrival time and requested doctors' specialty.

Moreover, a report manager module is integrated into the ENCORE Clinic Software. The practitioners of the clinic may utilize this module to generate the reports according to the intended timeframe. The inventory status, including the shelf life of each drug, are appended in the report to act as a reminder. The financial reports could also be generated based on the services provided for the analysis purpose.

Several limitations exist in the software. First of all, there is no online appointment service provided to the patient. Therefore, the patients are required to either visit the clinic or make a phone call to the front desk of the clinic in order to make an appointment. This may bring inconvenience to the patients as they cannot preview the schedule of the doctors to make a suitable arrangement on the visit timeslot [6].

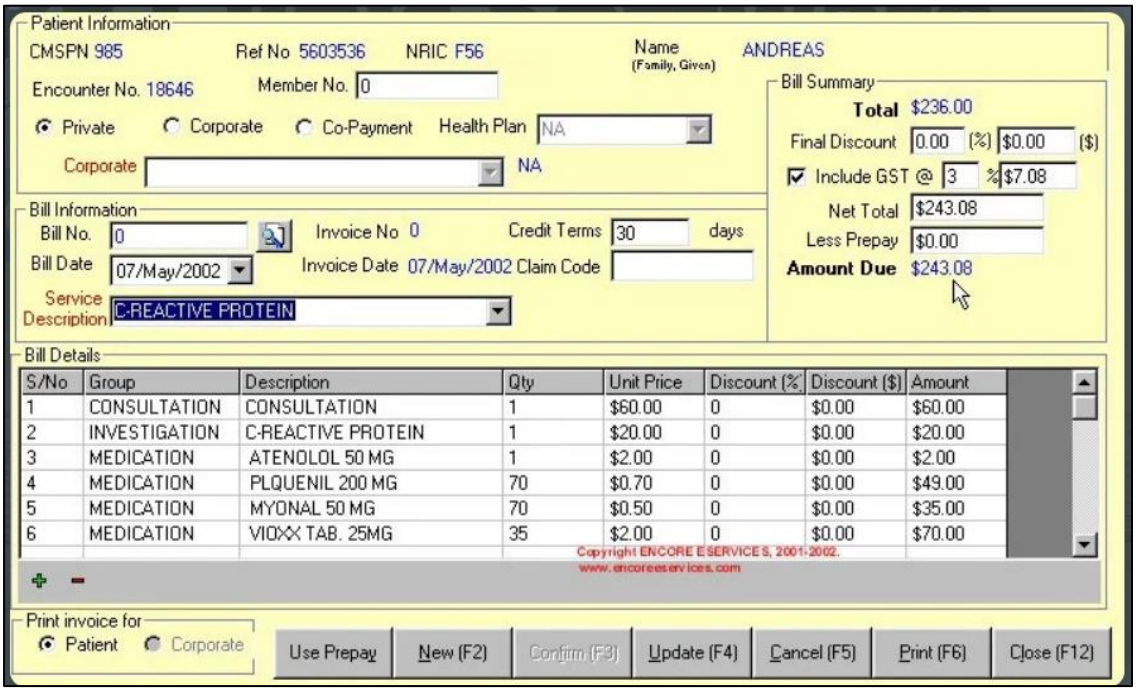


Figure 2.1 Billing interface of patient

In terms of the user interface, the GUI of the system could be further optimized as the patient information displayed on the user interface are compact. The lacking of highlight on function buttons and inconsistent in input field organization may be improved to enhance the work efficiency since the reading and entering of the patient information would be facilitated.

2.2.5 Kareo Clinical

Kareo Clinical is an EHR software that allows the clinic to handle the patients' records and records the prescriptions provided to the patients. Its targeted user domain is the healthcare services providers as the software are composed of a variety of features that support the business flow of the clinic [7].

A patient portal is integrated into the software to allow the patients to check their prescriptions and their treatment records. The lab results and all checkup-relevant information are available on the platform so that the patients could directly view the results using the portal, without the need in visiting the clinic again to obtain the results.

In addition, a communication channel between the patients and the medical staffs including doctors is provided in the software. The patients could request an explanation on the lab results, offering the patients an alternative method if they are unable to have face to face meeting with the doctors. The communication channel also enabled the patients to solve simple queries online efficiently.

There are some limitations in terms of the software features. Unlike other similar software, there is no online appointments feature provided to the patients. The patients are not allowed to perform appointments independently. They are required to contact the medical staffs whenever they are intended to make an appointment.

The feature which provides the reminder of appointments is also absent in the software. Hence, the patients need to pay extra attention to the appointment date and login into the portal to recall the appointments made. To resolve any overlook incident, the software should consider adopting the notifications pushing features in their software to remind their patients before the actual appointment date.

2.2.6 Juvonno

In the marketplace, there is also an EMR software which is recognized as Juvonno, to facilitate the clinic in providing medical services. The software is equipped with several functional modules, reducing the potential administrative issues that might occur under manual work operations. With this, the clinic staffs are managed to put their focus on helping out the patients, avoiding the distractions due to human errors when interacting with the database.

In the Juvonno software, there is an analytic tool available for the clinic. The analytic tool simplified the calculation of the data, compile the statistics obtained into a more intuitive format and display them in a chart. With this feature, the clinic management team are managed to gain insight into the clinic operations. This feature expressed its importance especially when the revenue report is required to determine the growth and future direction of the clinic. Furthermore, the data generated could be exported into .pdf format, assisting the complicated analysis and documentation process [8].

Apart from the general features and the analytic tool, there is also an audit trail and compliance feature integrated with the software. Whenever there is a user logged in to the system, its actions including viewing and updating records would be recorded. The monitoring and maintenance of the system would be easier along with this feature since there is a history that allows the tracing of the changes while providing a checkpoint for rollback when necessary.

In terms of limitations, the software is lacking in the communication channel between the patients and the practitioners. Hence, the patients are most likely to have a visit to the clinic by themselves or solve their queries through a phone call. To assist the patients who have questions on general information about the clinic such as the operating hours of the clinic, a chatbot could be considered to be implemented so that the simple Q&A session is simplified with the auto-reply.

2.2.7 e-Clinic 2

There exists a clinic management software named as e-clinic 2 that provides multiple features that suits the need of the clinic regardless the business size. This software provides basic features such as online appointments for patient and comprehensive documentation for the medical staffs.

The e-clinic 2 software also provides sophisticated clinical templates for the patient records whereby the doctors could even draw a diagram of the injury position and make annotation inside the treatment record. With this, the doctors are managed to have a quick recall about the patients at their next visit since the annotation and diagram are more intuitive than plain text records [9].

As the registered patients made appointments, the integrated communication module would monitor the appointment date and send out the confirmation messages prior to their appointment. The reminder template is customizable so that the clinics are managed to append any additional information into the reminder message. The clinics could also utilize the messages services to inform their patients about their events, indirectly improve the retention of the patients and the business performance.

Similar to the previous software reviewed, the online communication channel is not available in this software. A simple chatbot could be adopted to reduce the workload of the receptionist as the patients would be able to solve their queries directly with the chatbot without calling the receptionist.

2.2.8 Critical Remarks of Previous Works

Generally, all of the products are having different focus in terms of the functionality. For instance, the Patient Manager mentioned in 2.1 has taken access privileges into consideration, while offering a variety of features including the patient management module. It is also linked to a medical billing module which generates invoices for the patients. The eClinic System provides another feature, whereby an alert system is integrated to send notifications via email and text messages to the patients upon their appointments. Meanwhile, the Jane App has an online appointment booking features, which allows the patients to schedule their appointment remotely. It is also equipped with the images and video uploading services, so that the doctors can upload the X-Ray images of patients for future medical consultation references.

As indicated in 2.4, Encore clinic software is equipped with an unique feature among all, whereby it provide a report manager module to assists the inventory management of the clinic. Kareo clinic, on the other approach, established a communication channel between the patients and the medical staffs, so that the patients can conduct the

consultation online. Furthermore, Juvonno is more inclined to the analytic tool, which simplifies data processing and provides statistics visualisation to aid clinic operations. To extend the functionality, e-Clinic 2 even prepared advanced clinical templates that enable physicians to add annotations inside medical records.

In terms of weaknesses, the compact UI of the Encore Clinic Software should be enhanced to optimized the data visualization. Besides that, the absence of appointment modules as in Encore Clinical Software and Kareo Clinical should also be taken into account, since it substantially reduces the burden of the staffs while providing convenience to the patients. In addition, the narrow availability of the Patient Manager also limited the accessibility of the application, affecting the adoption of the application significantly since the patients' devices are comprised of various operating systems or platforms. The comparison of availability and functionality of all mentioned products are appended below for references.

Features / Software	Patient Records	Online Appt.	Role Division	Inventory	Notification
Patient Manager	√		√		
eClinic System	√	√	√	√	√
Jane App	√	√			√
ENCORE	√		√	√	√
Kareo Clinical	√				
Juvonno	√	√		√	√
e-clinic 2	√	√			√
Project Outcome	√	√	√	√	√

Table 2.5 The summary of the reviewed software features

Availability / Software	Web	Windows	Android	iOS
Patient Manager		√		
eClinic System	√		√	√
Jane App	√			
ENCORE		√		
Kareo Clinical	√			√
Juvonno	√			
e-clinic 2	√			
Project Outcome	√	√	√	√

Table 2.6 The summary on the availability of the reviewed software

Chapter 3 System Methodology and Approach

3.1 System Design Diagram

3.1.1 System Architecture Diagram

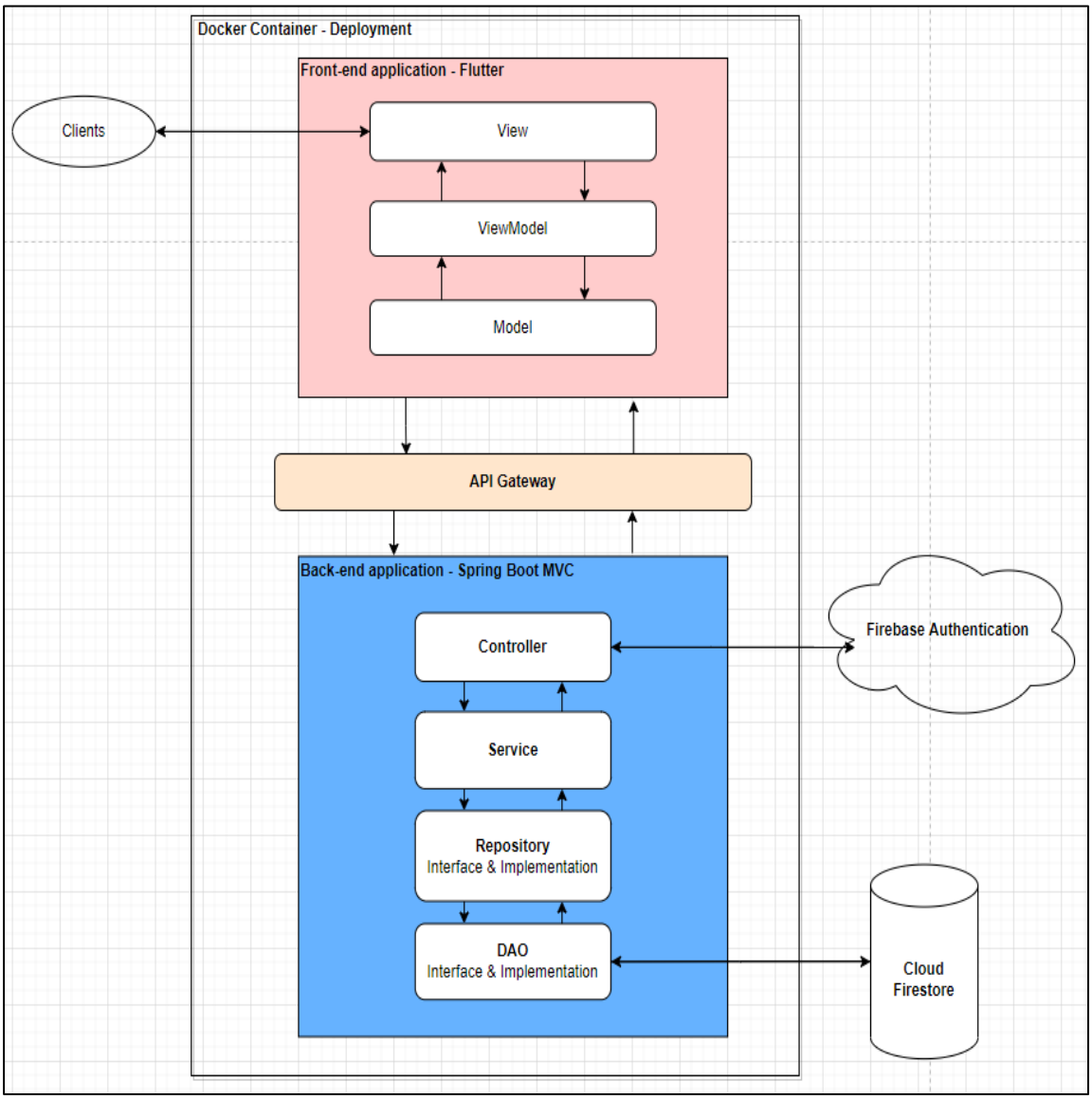


Figure 3.1 Architecture diagram of the system

Figure 3.1 illustrate the architecture of the system. Generally, the system is separated into 5 major parts, which are the UI presentation, RESTful APIs, business logic processing, data storage and authentication and authorization.

First of all, the front-end application which adopted Flutter framework is responsible to interact with the users, whereby it displays all relevant information in predesigned user interface and also waiting for requests from the users. The users can perform intended actions such as create dependent record and book appointments by clicking on the corresponding buttons.

On the back-end application, all of the services are encapsulated into APIs, waiting for any incoming requests. It will listen to incoming requests, mapping them into controllers and then trigger their corresponding services according to the business logic. With the exposed APIs, the front-end application is managed to connect to the back-end application, allowing the system works together to handle the users' requests. After the response is generated, it will be encoded into JSON format, and be sent back to the front-end application. The front-end application will then decode the JSON response, mapping it into corresponding objects, eventually updating the user interface and informing the users regarding the result of the actions.

In the data storage level, Cloud Firestore is implemented as the cloud storage solution. The adoption of Cloud Firestore not only eliminating heavy upfront investment cost in acquiring physical servers, while providing scalability to the system since it does not have fixed capacity and limitations on the data stored. As it is a pay-as-you-go service, the system owner is only being charged according to the usage rate, tailoring the cost incurred without bringing serious burden in terms of regular maintenance cost.

Before conducting any actual actions to the Cloud Firestore by the back-end application, the requests received will be validated through token authentication mechanism. For each of the API request, a Bearer token generated by the Firebase Authentication for authenticated users must be attached to the request header. When this request is received by the back-end application, it will first validate the requester identity by decoding the token attached, and decide whether to perform requested actions according to the validation result. In case of lacking authorization token, the requests will be rejected, securing the data stored by blocking access from unauthorized users.

To further improve the efficiency in the system deployment and distribution, the entire system including both front-end and back-end application will be composed into a Docker Container. Docker Container will bundle up all the necessary dependencies when building the container, so that the actual implementation in the production environment can be ease since no further environment setup (i.e., installation of JVM or installation of Flutter SDK) are required. Furthermore, Docker also offers flexibility in the deployment, as the composed Docker Container can either be uploaded to any cloud hosting services or be hosted locally in own physical servers.

In terms of system internal structure, the system has been separated into 2 sections, which are the front-end application that adopts Flutter framework and the back-end application that adopts Spring Boot framework. The adoption of these frameworks has contributed tremendously in simplifying the development process since they are managed to significantly reduce the boilerplate codes required. To further enhance the organization and module structures of the system, each of application has implemented additional architecture pattern, whereby the Flutter application will comply to Model-View-ViewModel (MVVM) architecture, while the Spring Boot application will implement both Repository and Data Access Object (DAO) Pattern to improve the module organization.

First of all, the MVVM is an architecture pattern that is suitable to standardize the modules structure during the development of user interface. In some cases, without standards, the development of the application will mix up all of its components, without concerning whether its content is related to business logic or UI presentation. This problem escalates rapidly with the scale of the application, leading to a system with poor maintainability and testability [10]. However, this issue can be against through segregating the application using MVVM architecture.

Generally, the MVVM architecture breaks down the entire front-end application into 3 layers: Model layer, View layer and the ViewModel layer, forming smaller pieces of manageable components. Among all three layers, the model layer is served as the business logic layer, whereby it composed all of the business logic that will be responsible for the CRUD operations to retrieve data. In the proposed system, this layer will interact with the APIs exposed by the back-end application, so that the real services which reside in the back-end application can be triggered and subsequently returns

requested data. With the data obtained from model layer, the ViewModel layer will further extracts the data and perform preprocessing for the user interface. It is responsible for the state management to notify the view layer to update their UI after the data is done prepared. The last layer, View layer contains the pure presentation codes that built up the UI which visible to the users of the application. With these layers, the separation of concern can be achieved, whereby any changes on one of the layers are isolated from the other layers, not only reducing the development effort required but also bringing down the risk level whenever a patch or fix is applied.

On the other hand, the Spring Boot application has complied with the standard that integrates Spring MVC, Repository pattern and DAO pattern. Among all these patterns, the DAO serves as the most underlying storage, which is an abstraction of data persistence. Each of the table will have a designated DAO, carrying all the CRUD queries that manipulate their respective table [11]. To distinguish the repository and DAO layer, the repository is recognized as a mechanism to manage storage encapsulation, data retrieval and looking up operation which handle a collection of objects [12]. In the system developed, the repository layer will be constructed as the upper layer of the DAO layer, so that any necessary DAO methods can be implemented to build a single domain. Then, the service layer will be the major component that decides program flow and exception handling process. These services are then encapsulated into APIs in the controller layer so that these services can be exposed to the front-end application. Before trigger any of the services, prior validation and verification on the user accessibility are made through token authentication offered by Firebase Authentication.

3.1.2 Use Case Diagram and Description

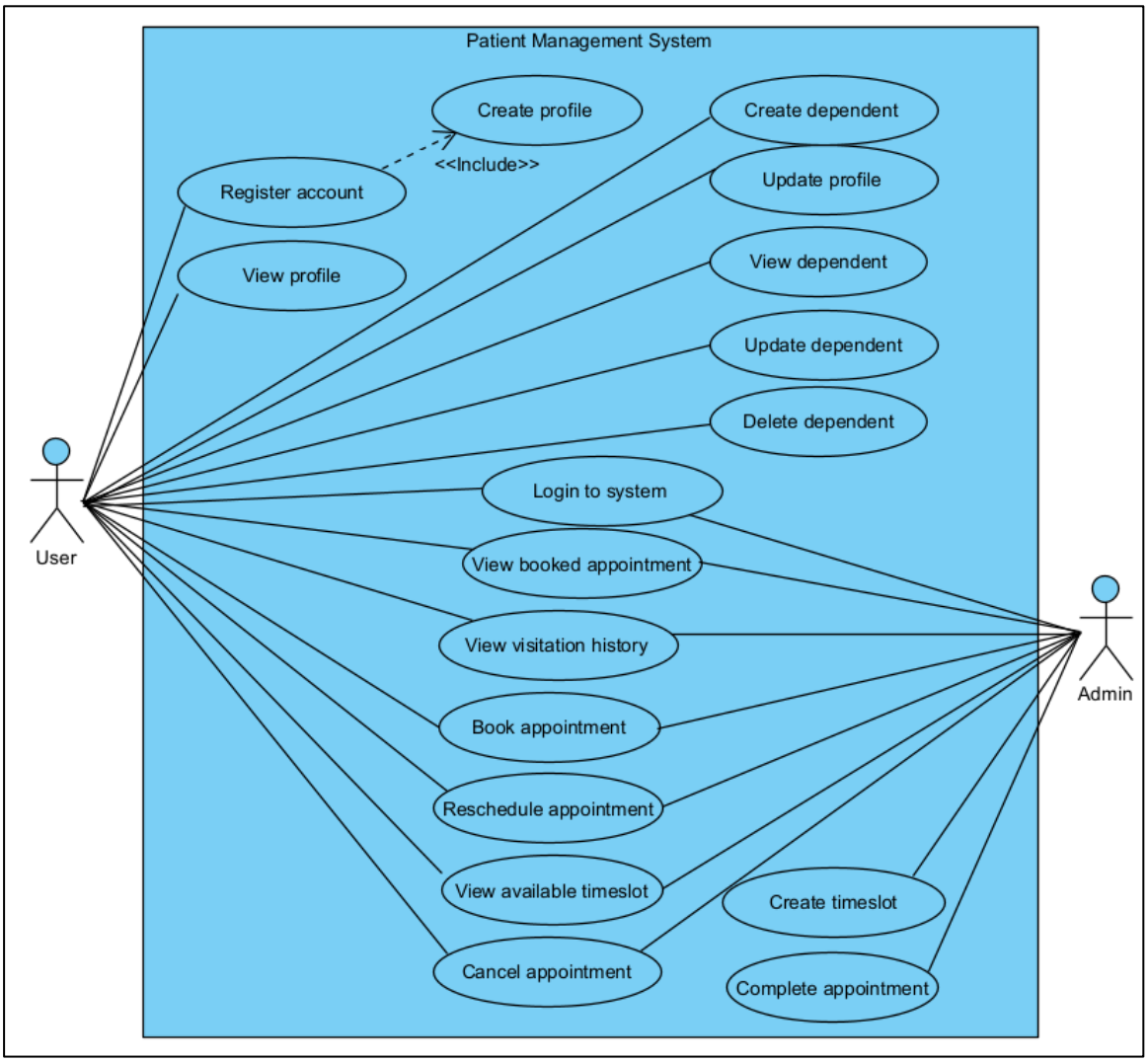


Figure 3.2 Use case diagram of the system

Use Case ID	UC001	**Version**	1.0
Use Case	Login to system		
Purpose	To authenticate user identity and avoid unauthorized access		
Actor	Admin, User		
Trigger	The user or admin visits the login page.		
Precondition	The user or admin is not login yet.		
Scenario name	**Step**	**Action**	
Main flow	1	Admin or user visits to the login page.	
	2	System displays the login form, waiting for email and password inputs.	
	3	Admin or user inputs email and password and submits.	
	4	System checks whether the credentials provided are valid.	
	5	System directs the admin or user to the main page of the system in case of valid credential.	
Alternate flow – Empty field for email or password	3a.1	Admin or user leave either email or password or both of the fields empty and the submit button is clicked.	
	3a.2	System prompts the admin or user to fill out the empty field.	
Alternate flow – Invalid email or password	3b.1	Admin or user enters invalid email or password.	
	3b.2	System checks whether the credentials provided are valid.	
	3a.3	System display access denied message.	
	3a.4	Back to Main Flow Step 2	

Table 3.1 Use case description for login to system

Use Case ID	UC002	**Version**	1.0
Use Case	Register account		
Purpose	To allow first time visitor registers for an account		
Actor	User		
Trigger	The user clicks on 'Don't have an account' button.		
Precondition	The user is not login yet and do not have an account.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the 'Don't have an account' button on the login form.	
	2	System displays the register form, waiting for email and password inputs.	
	3	User inputs email and password and submits.	
	4	System displays the user profile form, waiting for user input.	
	5	User inputs all required information in the form.	
	6	System checks whether the credentials provided are valid.	
	7	System registers an account using credentials provided.	
	8	System generates a user record in the database.	
	9	System directs the user to the main page of the system in case registration succeed.	
Alternate flow – Empty field for email or password	3a.1	User user leave either email or password or both of the fields empty and the submit button is clicked.	
	3a.2	System prompts the user to fill out the empty field.	
	3a.3	Back to Main Flow Step 3	
Alternate flow – Value in password field does not match with the one in confirm password field	3b.1	User inputs different value for the password and confirm password field and submits.	
	3b.2	System notifies the user for not matching value, waiting for new input.	
	3b.3	Back to Main Flow Step 3	
Alternate flow – Incorrect format of email or password	3c.1	User enters email or password in incorrect format.	
	3c.2	System checks whether the credentials provided passed validator requirements.	
	3c.3	System notifies the user and waits for new input.	
	3c.4	Back to Main Flow Step 2	

Alternate flow – Invalid value in user profile form	5.1	User inputs any of the profile field with value in incorrect format.
	5.2	User clicks on 'Register' button.
	5.3	System detects the errors and notifies the user, waiting for new input
	5.4	Back to Main Flow Step 5
Alternate flow – Email not valid	6.1	System detects the email provided is invalid.
	6.2	System notifies the user for the errors and wait for new input.
	6.3	Back to Main Flow Step 2

Table 3.2 Use case description for register account

Use Case ID	UC003	**Version**	1.0
Use Case	View profile		
Purpose	To allow the user to view his/her user profile		
Actor	User		
Trigger	The user clicks on ‘Profile’ tab in the side menu.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the ‘Profile’ tab in the side menu.	
	2	System directs the user to the user profile page.	
	3	System displays the user profile information to the user.	

Table 3.3 Use case description for view profile

Use Case ID	UC004	**Version**	1.0
Use Case	Update profile		
Purpose	To allow the user to update his/her profile		
Actor	User		
Trigger	The user clicks on ‘Edit’ icon in the profile page header.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the ‘Edit’ icon in the profile page header.	
	2	System displays a popup dialog containing the profile form, waiting for user input.	
	3	User inputs new value into the intended fields.	
	4	User submits the profile form with updated value.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs update through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Update success	7a.1	System displays a ‘Update Success’ dialog box.	
	7a.2	System updates the user interface with latest value.	
Sub flow – Update failed	7b.1	System displays a ‘Update Failed’ dialog box.	
Alternate flow – Incorrect format for provided update value	3.1	User inputs new value with incorrect format into the intended fields.	
	3.2	System detects the errors and notifies the user, waiting for new input.	
	3.3	Back to Main Flow Step 3	

Table 3.4 Use case description for update profile

Use Case ID	UC005	**Version**	1.0
Use Case	Create dependent		
Purpose	To allow the user to create dependent records		
Actor	User		
Trigger	The user clicks on ‘Add dependent’ button in the dependent section.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the ‘Add dependent’ button in the dependent section.	
	2	System displays a popup dialog containing the add dependent record form, waiting for user input.	
	3	User inputs information required.	
	4	User submits the add dependent form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs create dependent action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a ‘Action Success’ dialog box.	
	7a.2	System updates the dependent section with latest value.	
Sub flow – Action failed	7b.1	System displays a ‘Action Failed’ dialog box.	
Alternate flow – Incorrect format for value provided	3.1	User inputs information required in incorrect format.	
	3.2	System detects the errors and notifies the user, waiting for new input.	
	3.3	Back to Main Flow Step 3	

Table 3.5 Use case description for create dependent

Use Case ID	UC006	**Version**	1.0
Use Case	View dependent		
Purpose	To allow the user to view dependent records.		
Actor	User		
Trigger	The user clicks on ‘Profile’ tab in the side menu.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the ‘Profile’ tab in the side menu.	
	2	System displays the dependent records according to user dependent records.	
Sub flow – Don’t have any dependent record	2a.1	System displays ‘No dependent record found’ in the dependent record section.	
Sub flow – Has dependent record	2b.1	System displays the dependents records to the users.	

Table 3.6 Use case description for view dependent

Use Case ID	UC007	**Version**	1.0
Use Case	Update dependent		
Purpose	To allow the user to update dependent records		
Actor	User		
Trigger	The user clicks on ‘Edit’ icon in the dependent record.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on the ‘Edit’ icon in the dependent record.	
	2	System displays a popup dialog containing the update dependent record form, waiting for user input.	
	3	User inputs new value into intended fields.	
	4	User submits the update dependent form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs update dependent action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a ‘Action Success’ dialog box.	
	7a.2	System updates the dependent section with latest value.	
Sub flow – Action failed	7b.1	System displays a ‘Action Failed’ dialog box.	
Alternate flow – Incorrect format for value provided	3.1	User inputs new value in incorrect format.	
	3.2	System detects the errors and notifies the user, waiting for new input.	
	3.3	Back to Main Flow Step 3	

Table 3.7 Use case description for update dependent

Use Case ID	UC008	**Version**	1.0
Use Case	Delete dependent		
Purpose	To allow the user to remove dependent record		
Actor	User		
Trigger	The user clicks on ‘Remove’ icon in the dependent record.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	User clicks on ‘Remove’ icon in the dependent record.	
	2	System displays a popup dialog to confirm the delete action.	
	3	System performs delete dependent action through invoking relevant API.	
	4	System displays a dialog box according to the result of the action.	
Sub flow – Record removal success	4a.1	System displays a ‘Record removal success’ dialog box.	
	4a.2	System updates the dependent section with latest value.	
Sub flow – Record removal failed	4b.1	System displays a ‘Record removal failed’ dialog box.	

Table 3.8 Use case description for delete dependent

Use Case ID	UC009	**Version**	1.0
Use Case	Book appointment		
Purpose	To allow the user or admin to book for appointment		
Actor	User, admin		
Trigger	The user clicks on 'Book Appointment' button in the dashboard page or appointment page.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	The user clicks on 'Book Appointment' button in the dashboard page or appointment page.	
	2	System displays a popup dialog containing the book appointment form, waiting for user input.	
	3	User selects the intended timeslot, dependent and enter specific request.	
	4	User submits the book appointment form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs book appointment action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a 'Action Success' dialog box.	
	7a.2	System updates the incoming appointment section with the latest value.	
Sub flow – Action failed	7b.1	System displays a 'Action Failed' dialog box.	
Alternate flow – No dependent is selected	3a.1	User does not select dependent when submitting the form.	
	3a.2	System detects the errors and notifies the user, waiting for user selection.	
	3a.3	Back to Main Flow Step 3	
Alternate flow – No timeslot is selected	3b.1	User does not select timeslot when submitting the form.	
	3b.2	System detects the errors and notifies the user using alert dialog box.	
	3b.3	Back to Main Flow Step 3	

Table 3.9 Use case description for book appointment

Use Case ID	UC010	**Version**	1.0

Use Case	View booked appointment	
Purpose	To allow the user or admin to view incoming appointment	
Actor	User, admin	
Trigger	The user/admin clicks on the Appointment tab in the side menu.	
Precondition	The user/admin has logged into the system.	
Scenario name	**Step**	**Action**
Main flow	1	The user/admin clicks on the Appointment tab in the side menu.
	2	System displays the incoming appointment record according to user identity.
Sub flow – Identity is admin	2a.1	System retrieves all incoming appointment records.
	2a.2	System updates the incoming appointment section with the latest value.
Sub flow – Identity is user	2b.1	System only retrieves users' incoming appointment.
	2b.2	System updates the incoming appointment section with the retrieved value.

Table 3.10 Use case description for view incoming appointment

Use Case ID	UC011	**Version**	1.0
Use Case	Reschedule appointment		
Purpose	To allow the user or admin to reschedule appointment		
Actor	User, admin		
Trigger	The user/admin clicks on 'Reschedule' icon in the appointment record.		
Precondition	The user has logged into the system and has at least one appointment.		
Scenario name	**Step**	**Action**	
Main flow	1	The user clicks on 'Reschedule' icon in the appointment record.	
	2	System displays a popup dialog containing reschedule appointment form, waiting for user input.	
	3	User selects the new timeslot.	
	4	User submits the reschedule appointment form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs reschedule appointment action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a 'Action Success' dialog box.	
	7a.2	System updates the incoming appointment section with the latest value.	
Sub flow – Action failed	7b.1	System displays a 'Action Failed' dialog box.	

Table 3.11 Use case description for reschedule appointment

Use Case ID	UC012	**Version**	1.0
Use Case	Cancel appointment		
Purpose	To allow the user or admin to cancel appointment		
Actor	User, admin		
Trigger	The user clicks on ‘Remove’ icon in the appointment record.		
Precondition	The user has logged into the system and has at least one appointment.		
Scenario name	**Step**	**Action**	
Main flow	1	The user clicks on ‘Remove’ icon in the appointment record.	
	2	System displays a popup dialog to confirm the submit action.	
	3	System performs cancel appointment action through invoking relevant API.	
	4	System displays a dialog box according to the result of the action.	
Sub flow – Action success	4a.1	System displays a ‘Action Success’ dialog box.	
	4a.2	System updates the incoming appointment section with the latest value.	
Sub flow – Action failed	4b.1	System displays a ‘Action Failed’ dialog box.	

Table 3.12 Use case description for cancel appointment

Use Case ID	UC013	**Version**	1.0
Use Case	View available timeslot		
Purpose	To allow the user or admin to view available timeslot		
Actor	User, admin		
Trigger	The user clicks on ‘Timeslot’ tab in the side menu.		
Precondition	The user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	The user clicks on ‘Timeslot’ tab in the side menu.	
	2	System calls API to retrieve the available timeslots.	
	3	System displays a list of available timeslots	

Table 3.13Use case description for view available timeslot

Use Case ID	UC014	**Version**	1.0
Use Case	Create timeslot		
Purpose	To allow the admin to create timeslot for appointment		
Actor	Admin		
Trigger	The admin clicks on 'Create timeslot' button in the appointment page.		
Precondition	The admin has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	The admin clicks on 'Create timeslot' button in the appointment page.	
	2	System displays a popup dialog containing the create timeslot form, waiting for user input.	
	3	Admin specifies date range, weekly off day and holidays.	
	4	Admin submits the create timeslot form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs create timeslot action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a 'Action Success' dialog box.	
	7a.2	System updates the incoming appointment section with the latest value.	
Sub flow – Action failed	7b.1	System displays a 'Action Failed' dialog box.	
Alternate flow – Invalid value provided	3a.1	Admin does not specify either start or end date or both.	
	3a.2	System detects the errors and notifies the user, waiting for user inputs.	
	3a.3	Back to Main Flow Step 3	

Table 3.14 Use case description for create timeslot

Use Case ID	UC015	**Version**	1.0
Use Case	Complete appointment		
Purpose	To allow the admin to complete an appointment		
Actor	Admin		
Trigger	The admin clicks on ‘Complete’ icon in the appointment record.		
Precondition	The admin has logged into the system and there is at least one ongoing appointment.		
Scenario name	**Step**	**Action**	
Main flow	1	The admin clicks on ‘Complete’ icon in the appointment record.	
	2	System displays a popup dialog containing the complete appointment form, waiting for user input.	
	3	Admin states the diagnosis result and prescription.	
	4	Admin submits the complete appointment form.	
	5	System displays a popup dialog to confirm the submit action.	
	6	System performs complete appointment action through invoking relevant API.	
	7	System displays a dialog box according to the result of the action.	
Sub flow – Action success	7a.1	System displays a ‘Action Success’ dialog box.	
	7a.2	System updates the incoming appointment section with the latest value.	
Sub flow – Action failed	7b.1	System displays a ‘Action Failed’ dialog box.	

Table 3.15 Use case description for complete appointment

Use Case ID	UC016	**Version**	1.0
Use Case	View visitation history		
Purpose	To allow the admin or user to view visitation history		
Actor	Admin, User		
Trigger	The admin/user clicks on ‘Appointment’ tab in the side menu.		
Precondition	The admin/user has logged into the system.		
Scenario name	**Step**	**Action**	
Main flow	1	The admin/user clicks on ‘Appointment’ tab in the side menu.	
	2	System displays the visitation history section accordingly.	
Sub flow – No history record	2a.1	System displays a list of visitation history records.	
Sub flow – Has history record	2b.1	System displays a message to tell there is no history yet.	

Table 3.16 Use case description for view visitation history

3.1.3 Activity Diagram

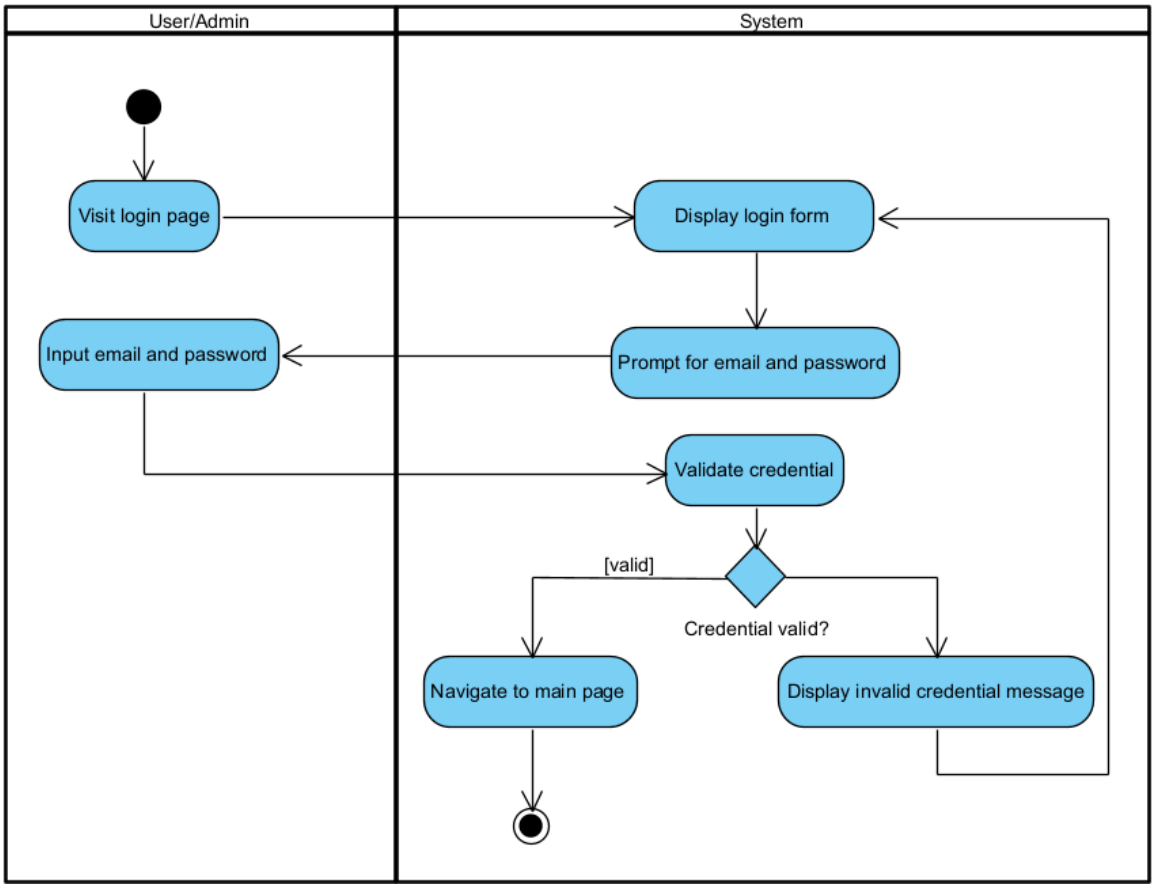


Figure 3.3 Activity diagram for login to system

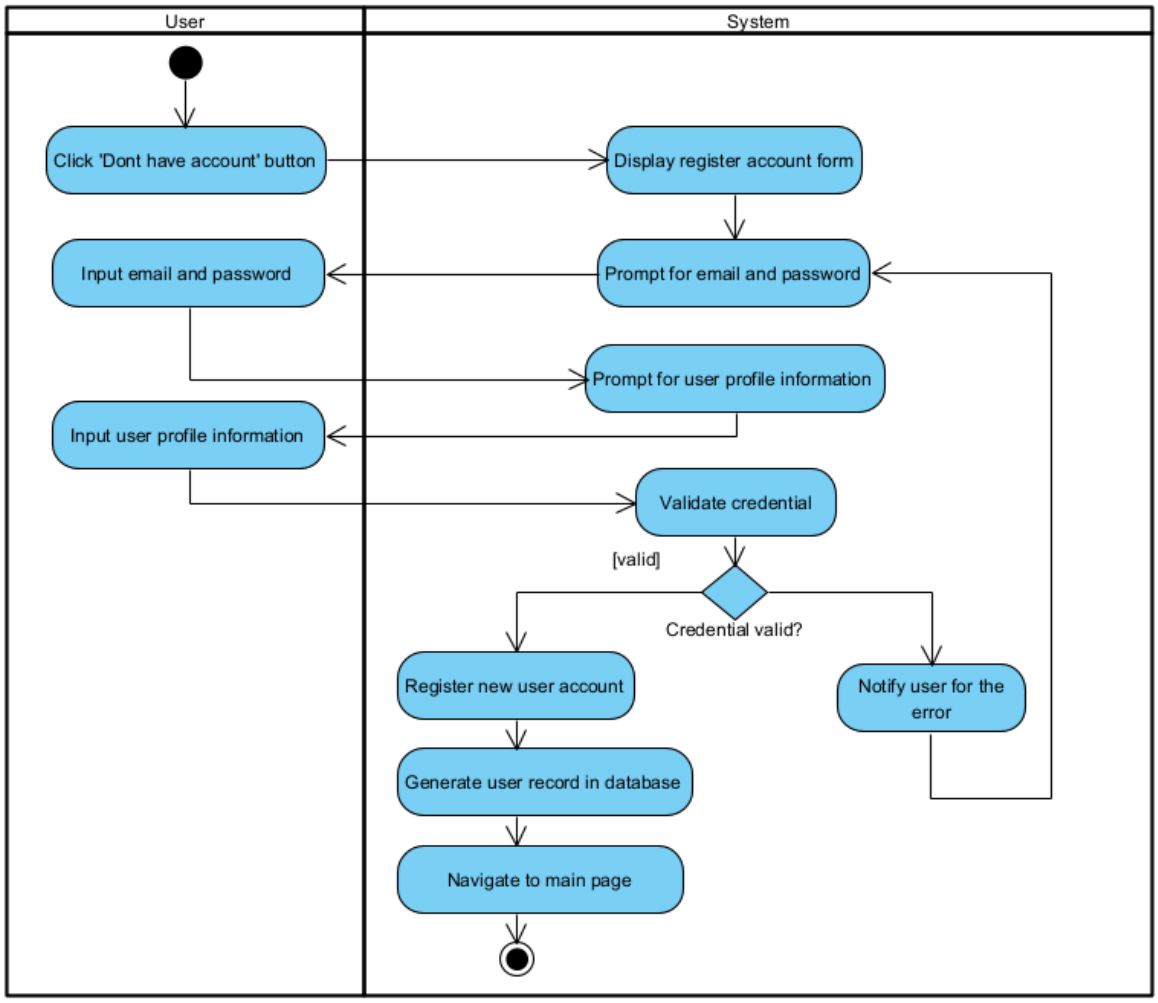


Figure 3.4 Activity diagram for register account

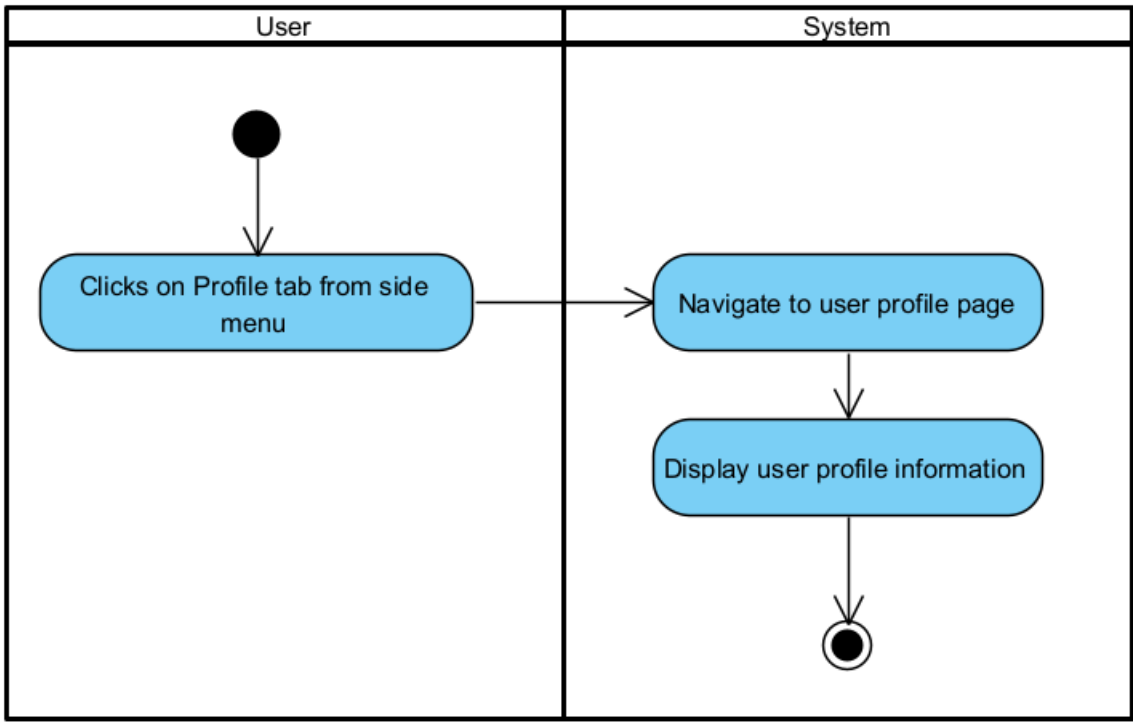


Figure 3.5 Activity diagram for view profile

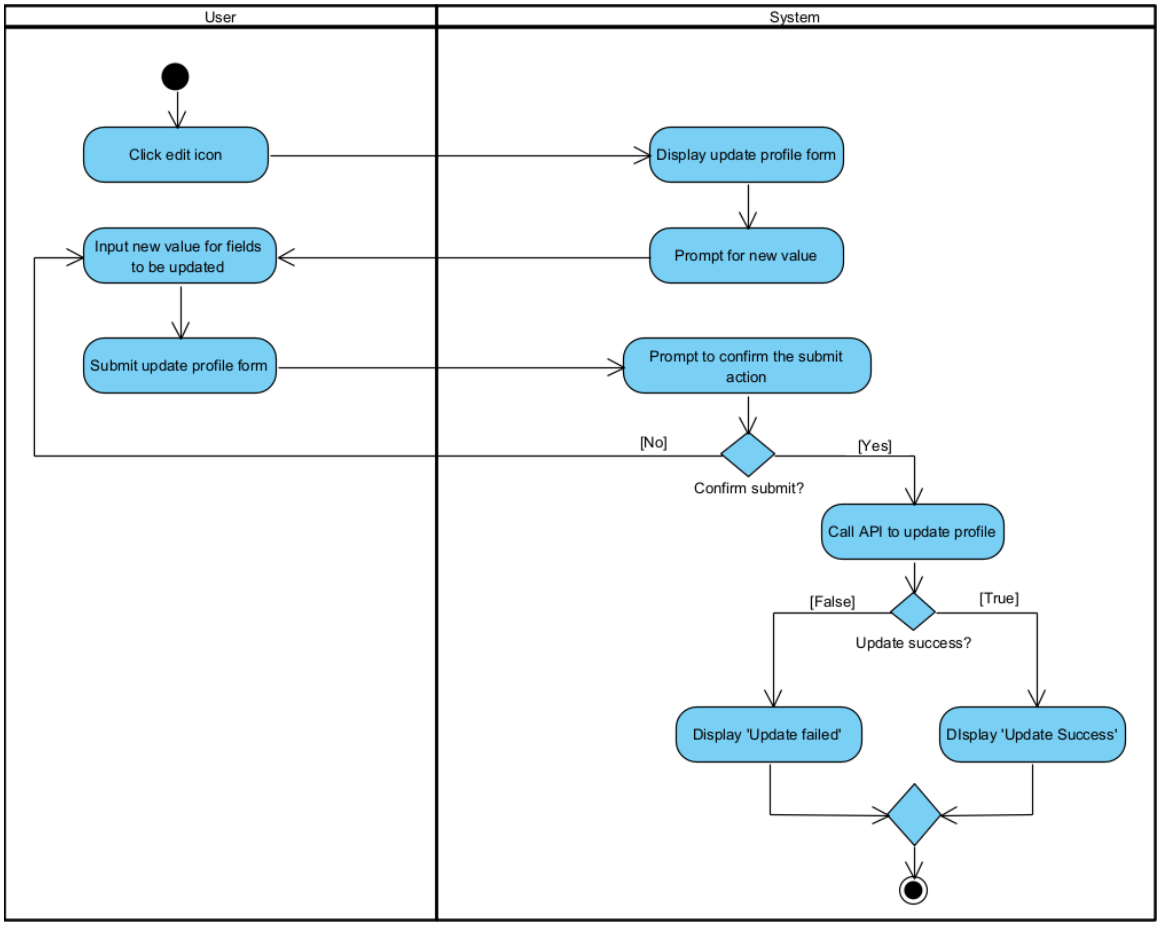


Figure 3.6 Activity diagram for update profile

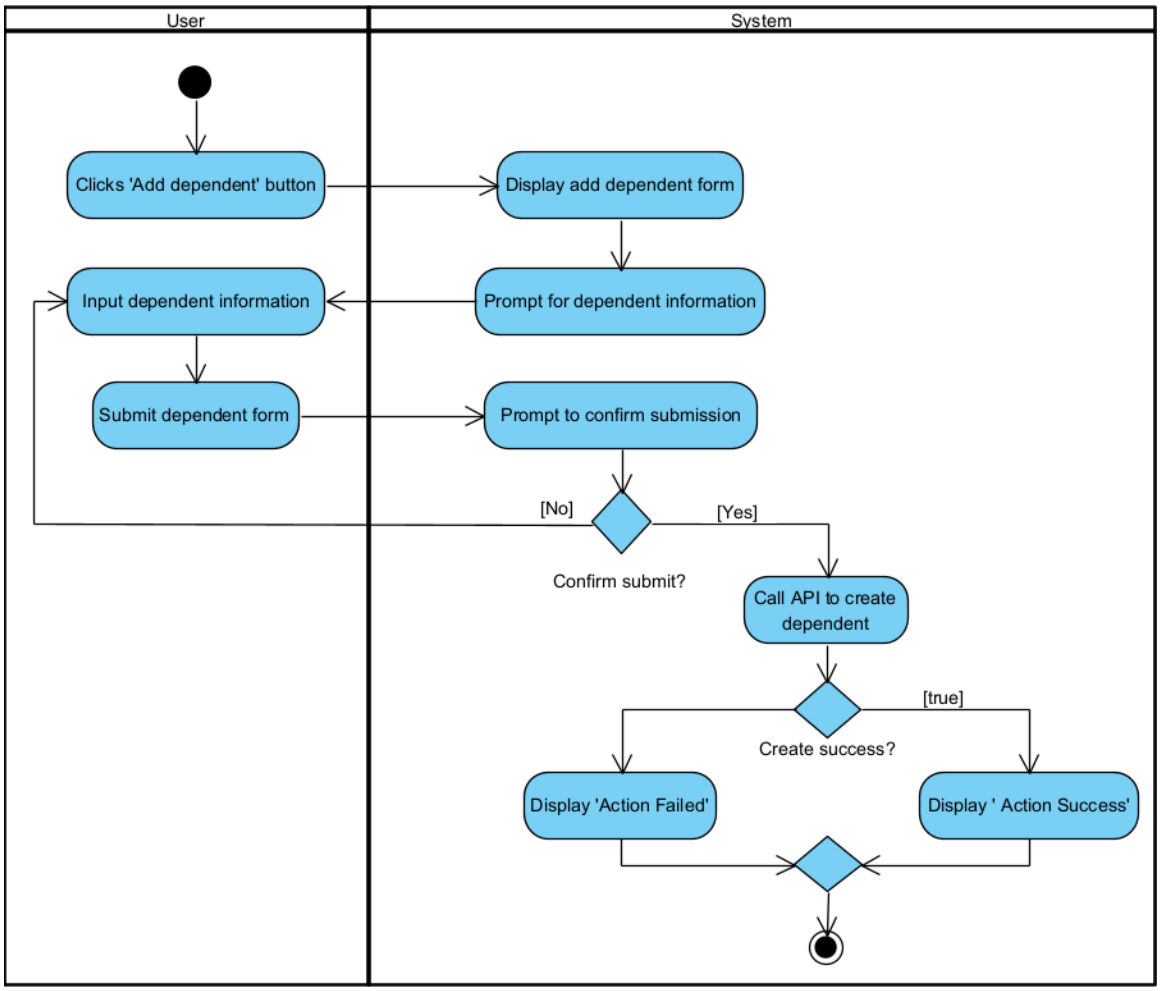


Figure 3.7 Activity diagram for create dependent

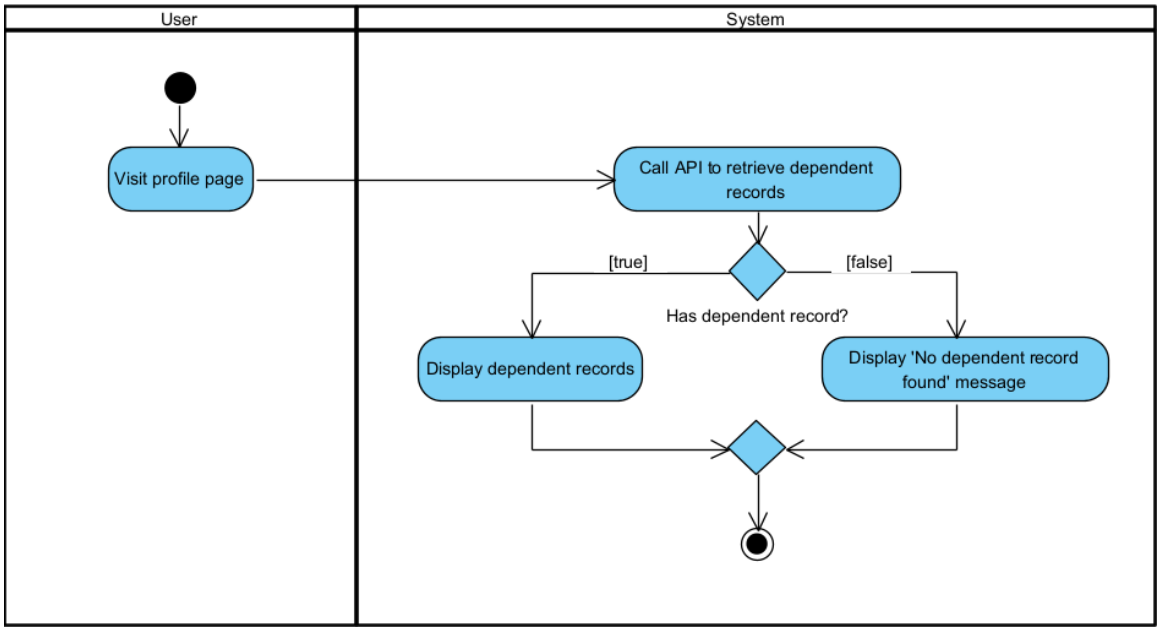


Figure 3.8 Activity diagram for view dependent

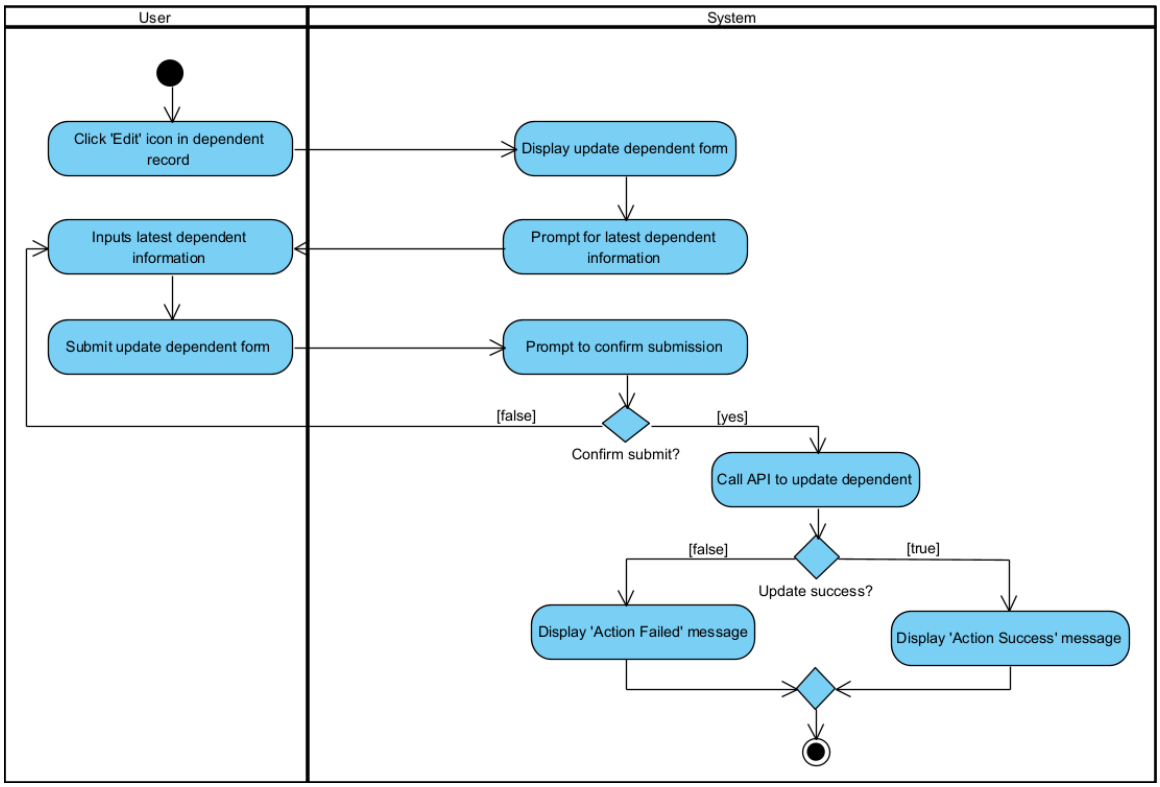


Figure 3.9 Activity diagram for update dependent

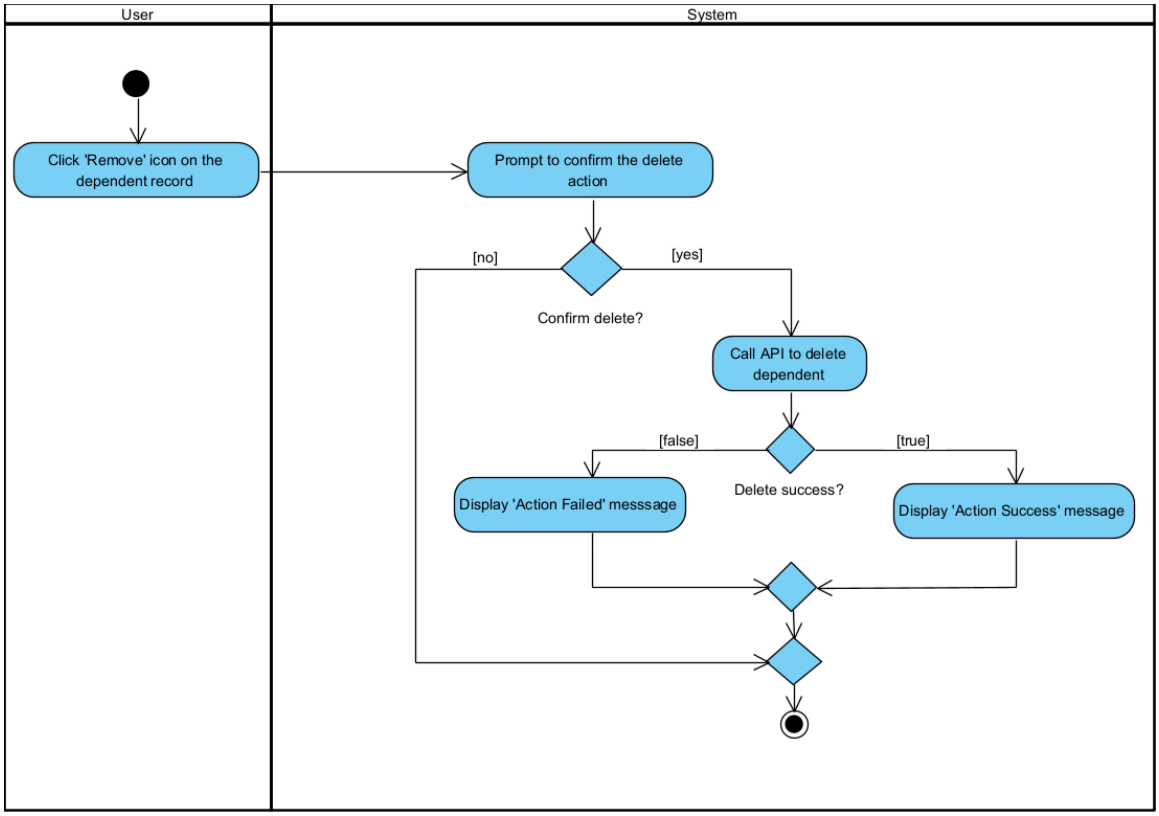


Figure 3.10 Activity diagram for delete dependent

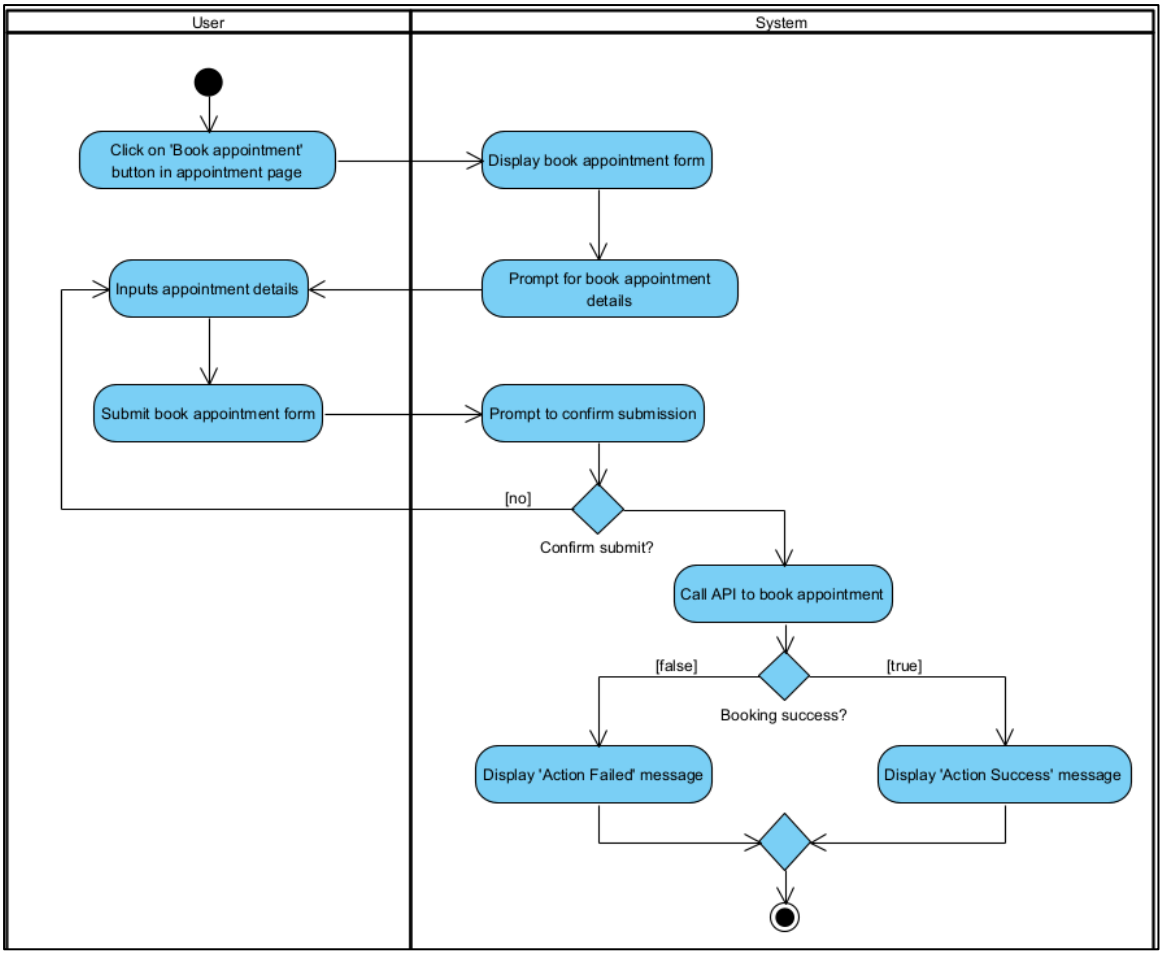


Figure 3.11 Activity diagram of book appointment

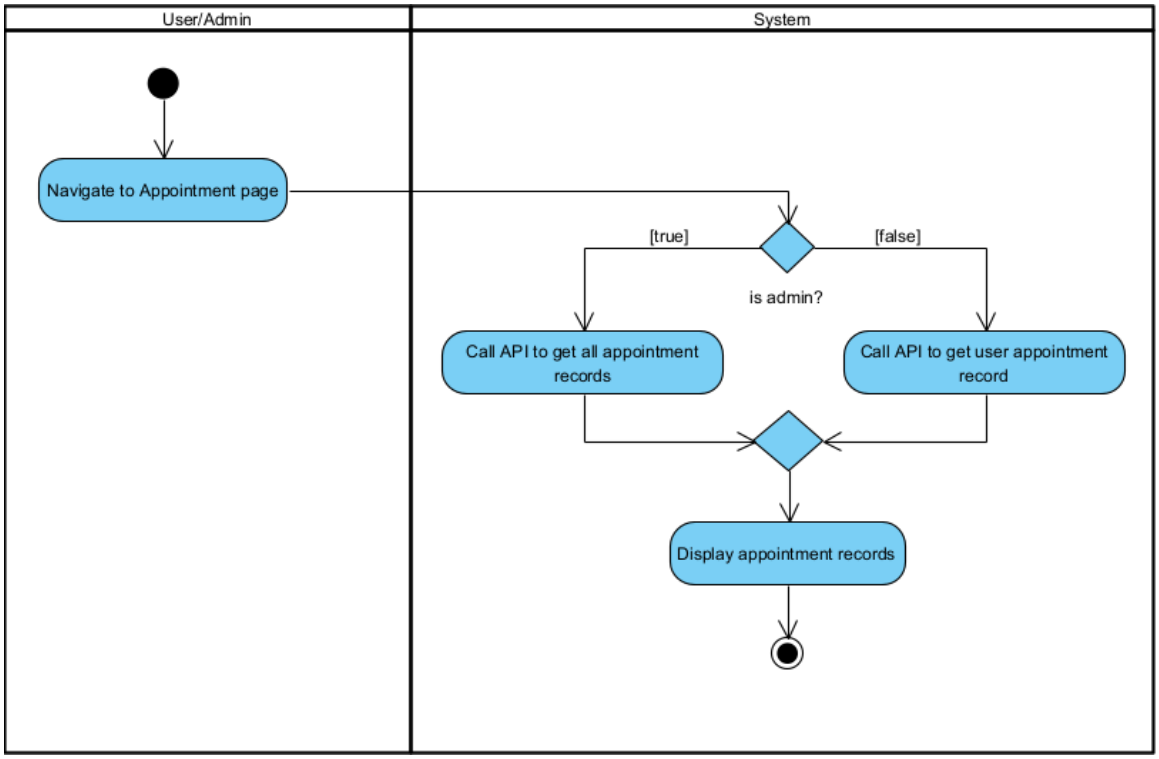


Figure 3.12 Activity diagram for view booked appointment

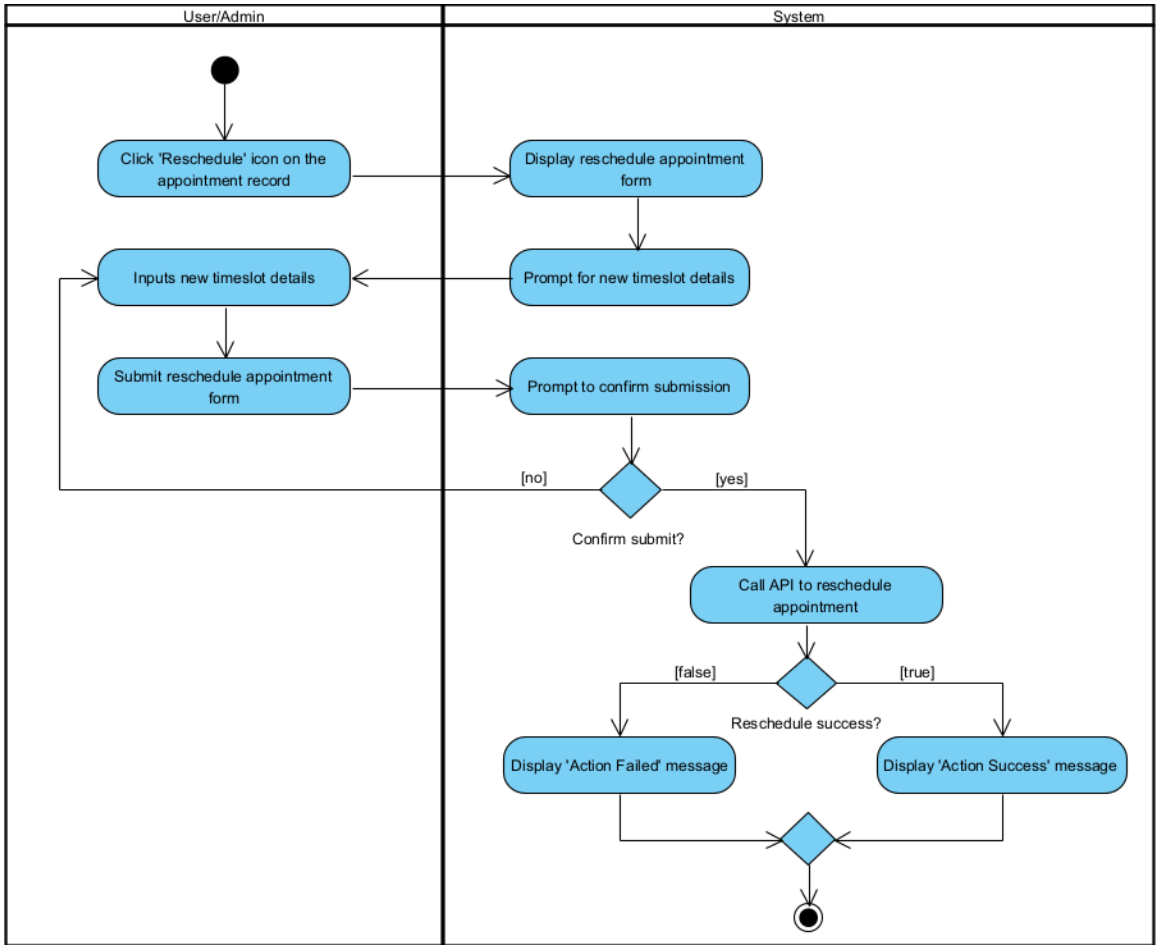


Figure 3.13 Activity diagram for reschedule appointment

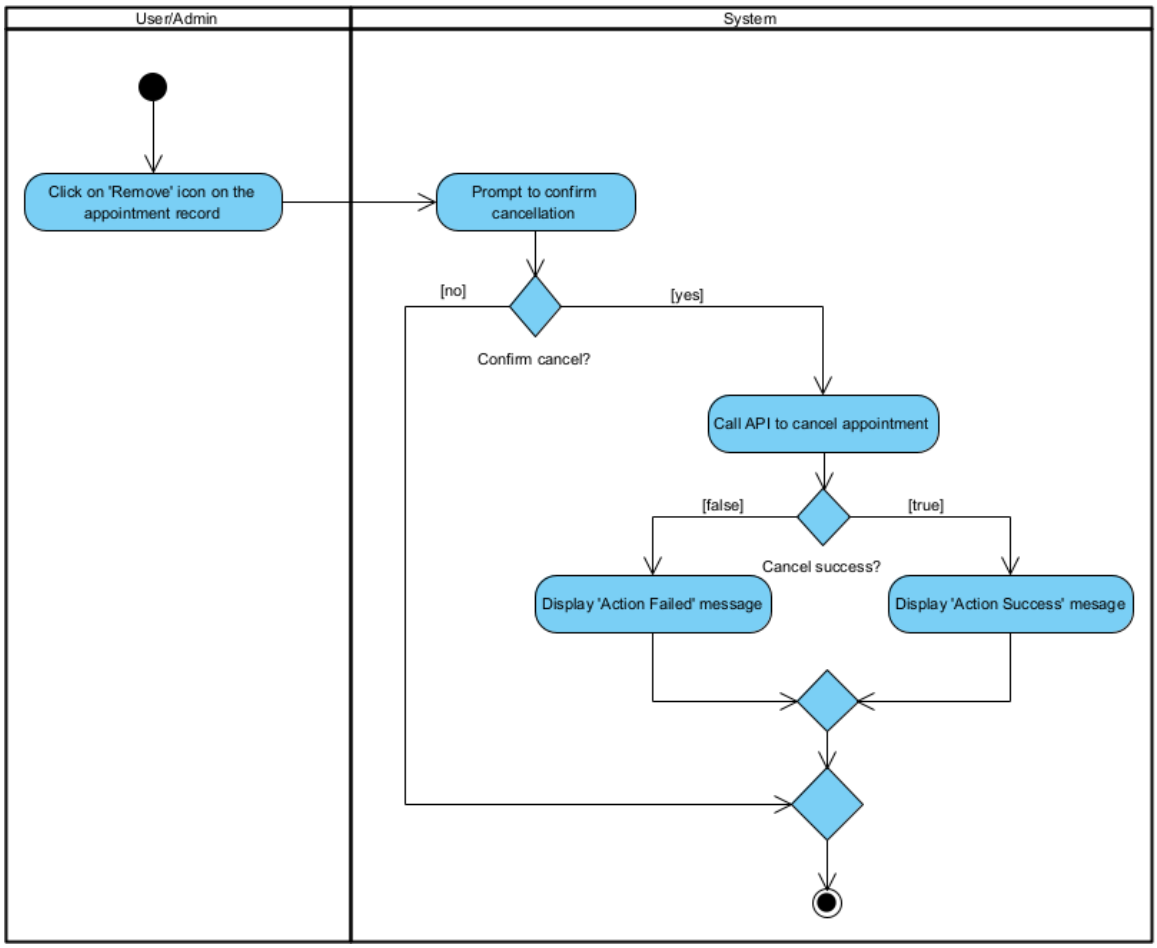


Figure 3.14 Activity diagram for cancel appointment

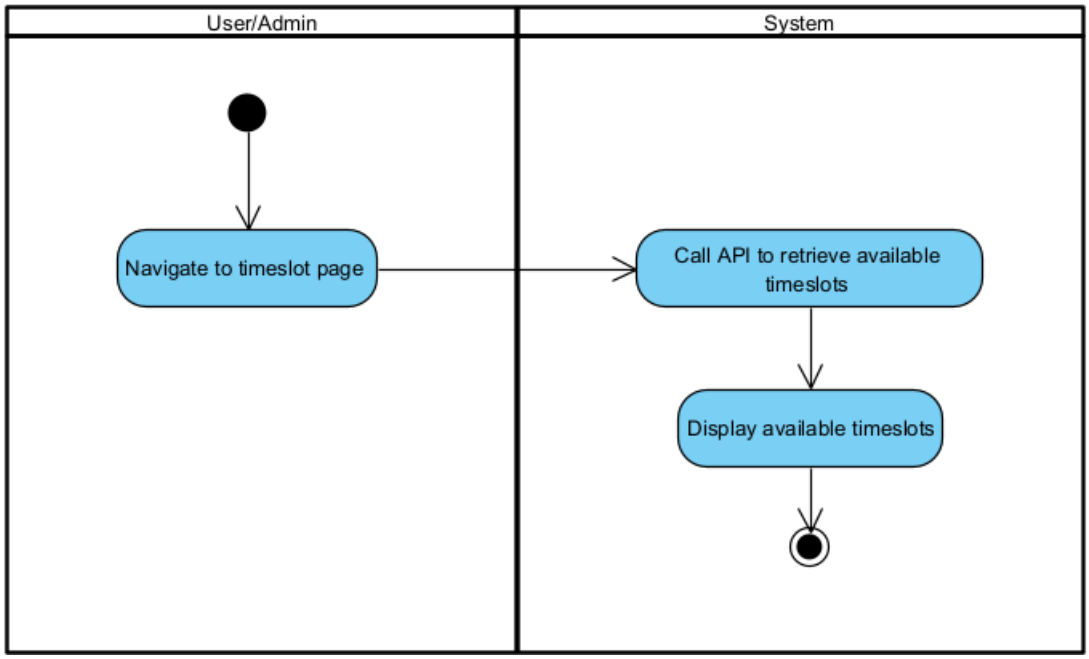


Figure 3.15 Activity diagram for view available timeslot

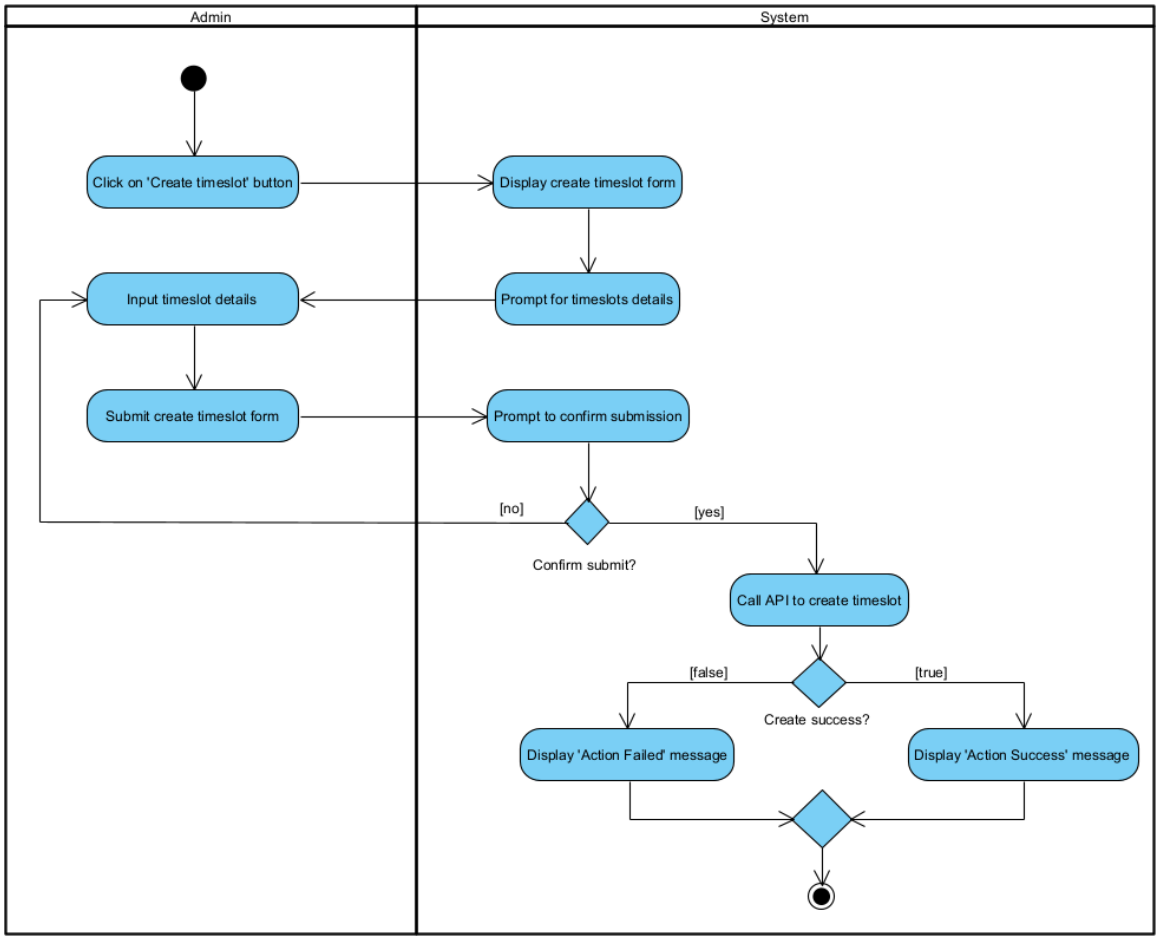


Figure 3.16 Activity diagram for create timeslot

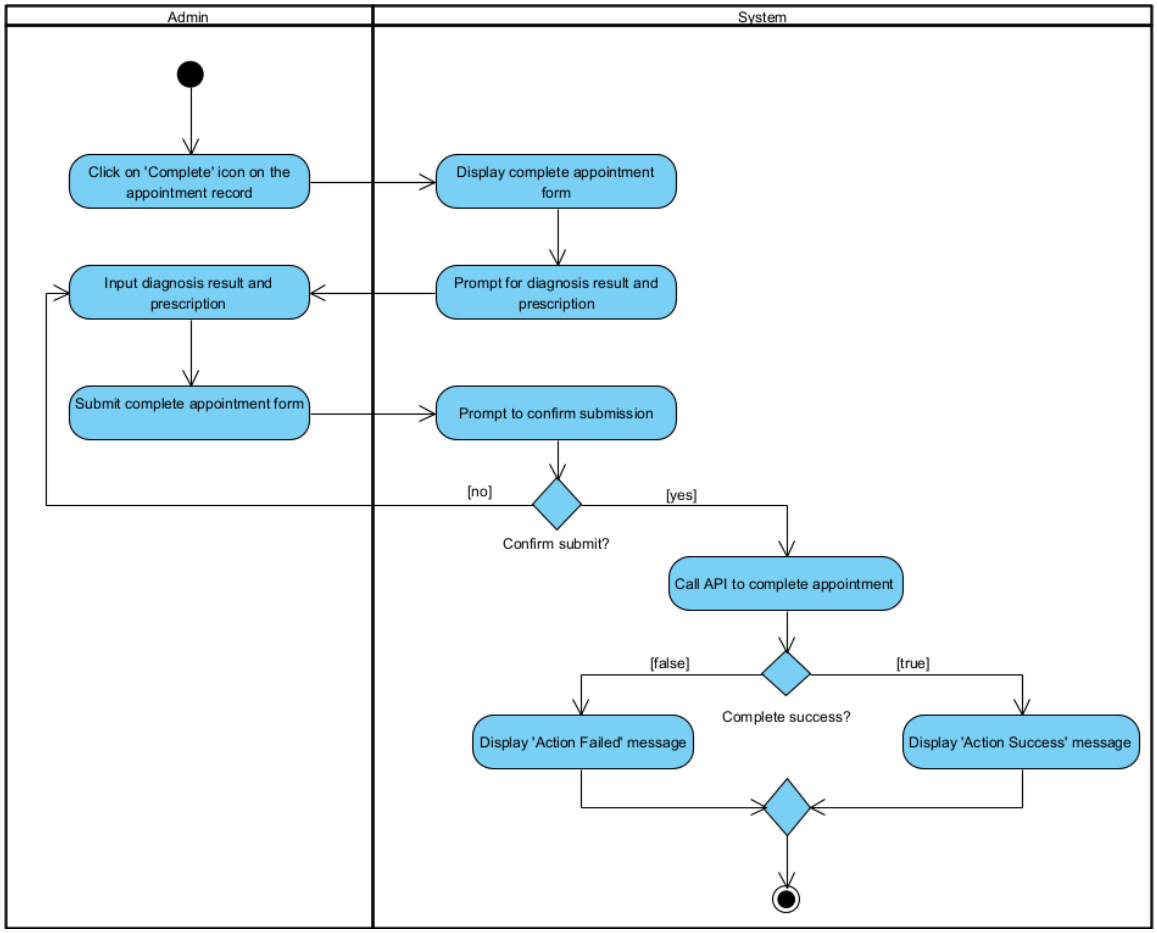


Figure 3.17 Activity diagram for complete appointment

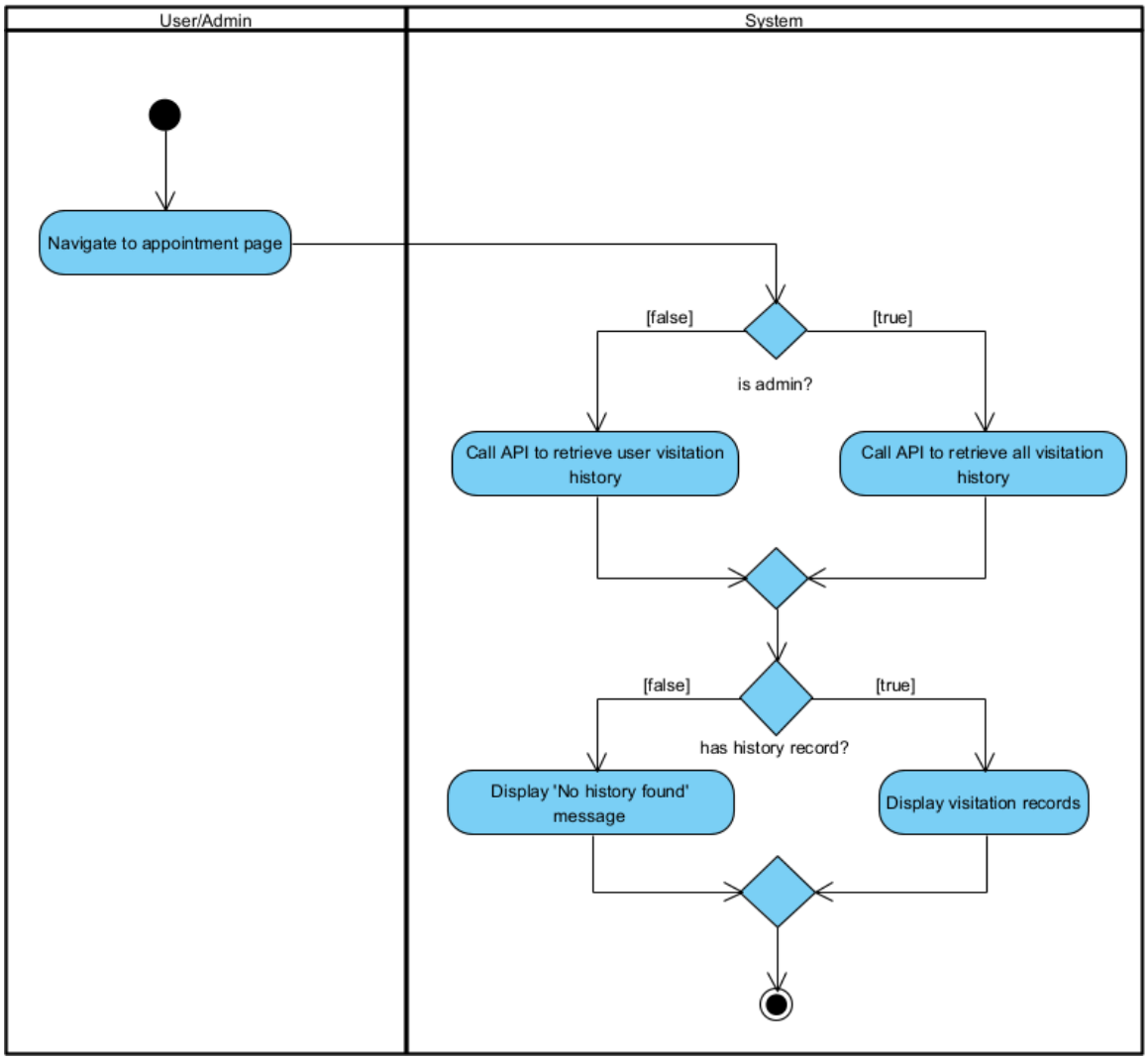


Figure 3.18 Activity diagram for view visitation record

Chapter 4 System Design

4.1 System Block Diagram

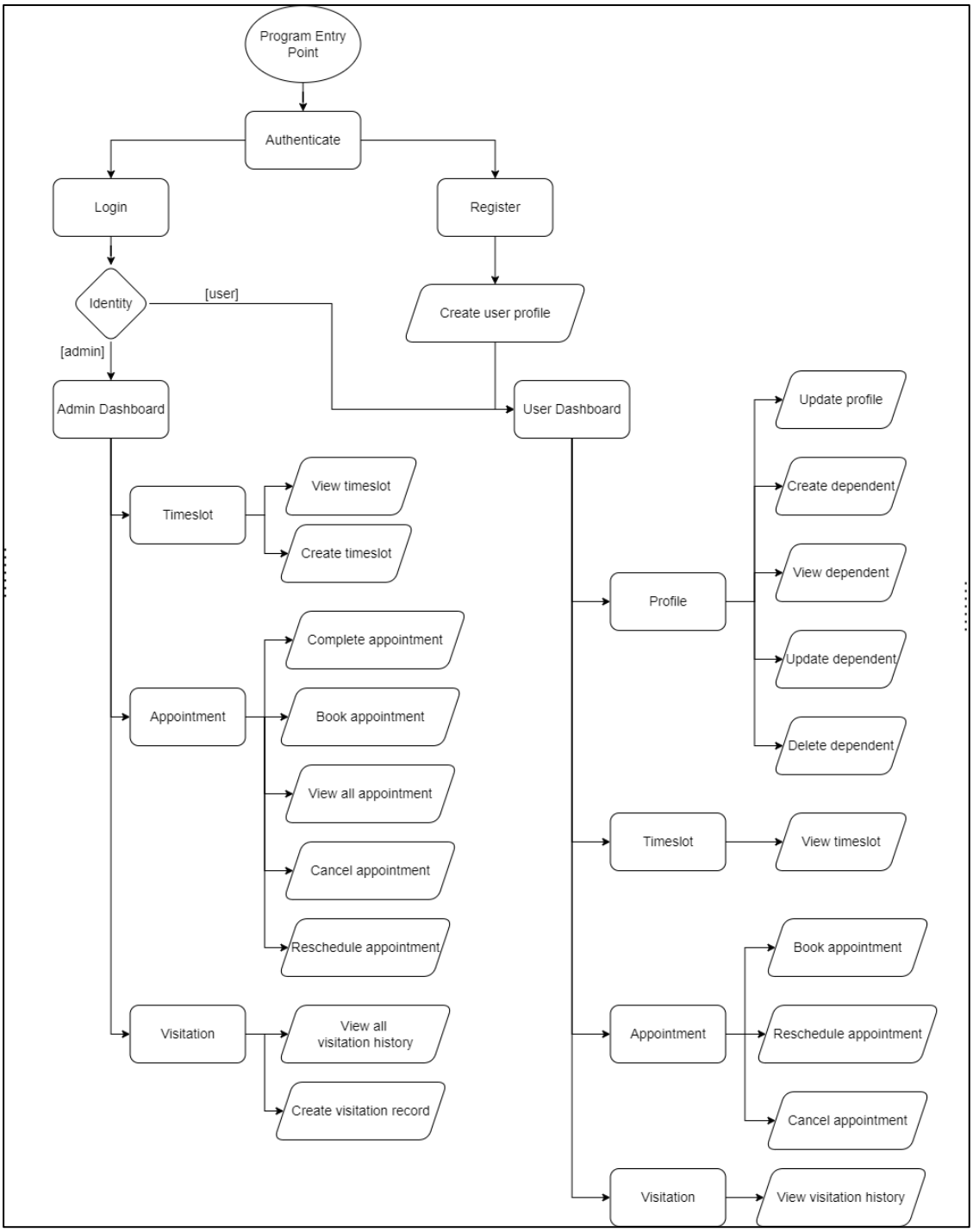


Figure 4.1 System block diagram

4.2 System Components Specifications

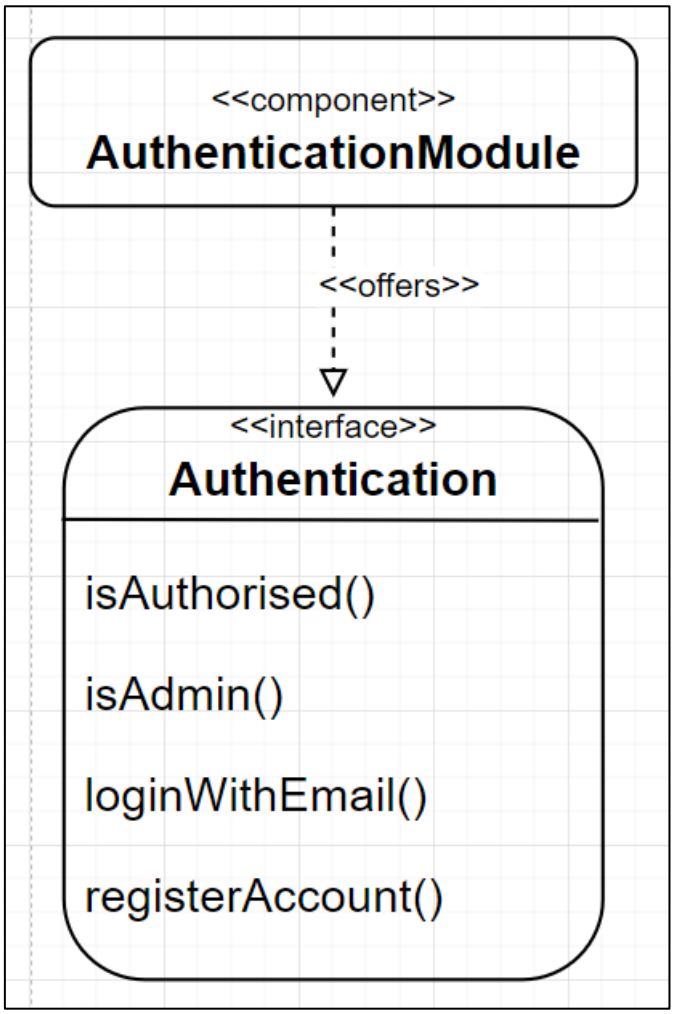


Figure 4.2 Component specification diagram of authentication module

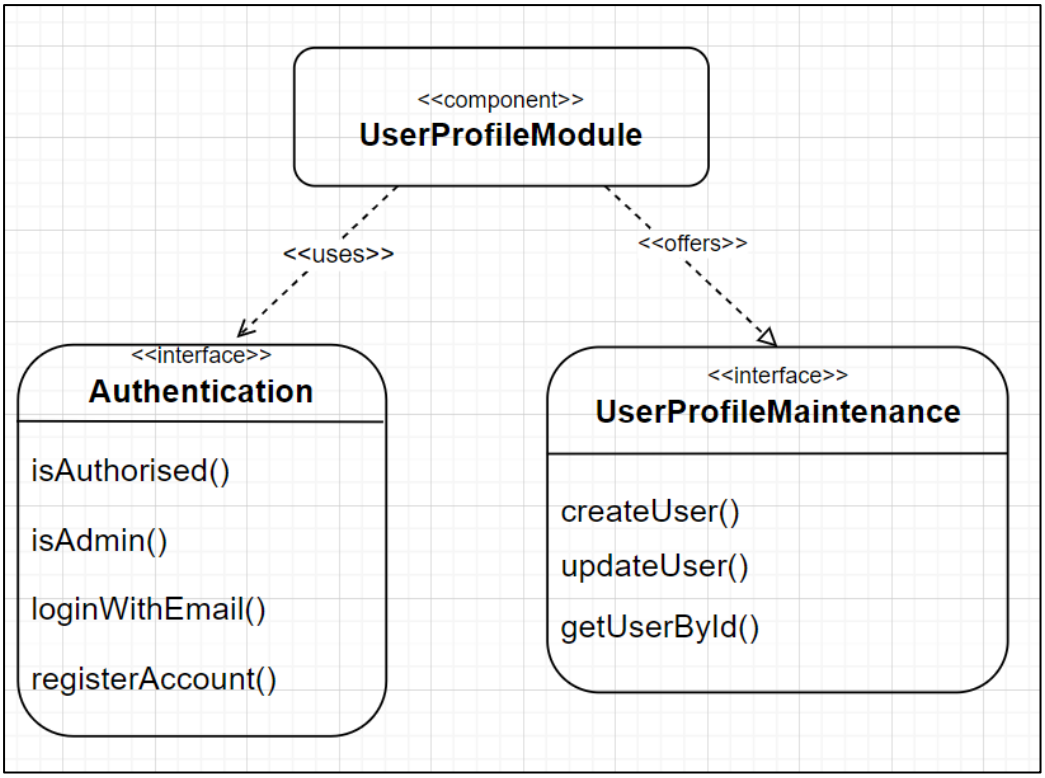


Figure 4.3 Component specification diagram of user profile module

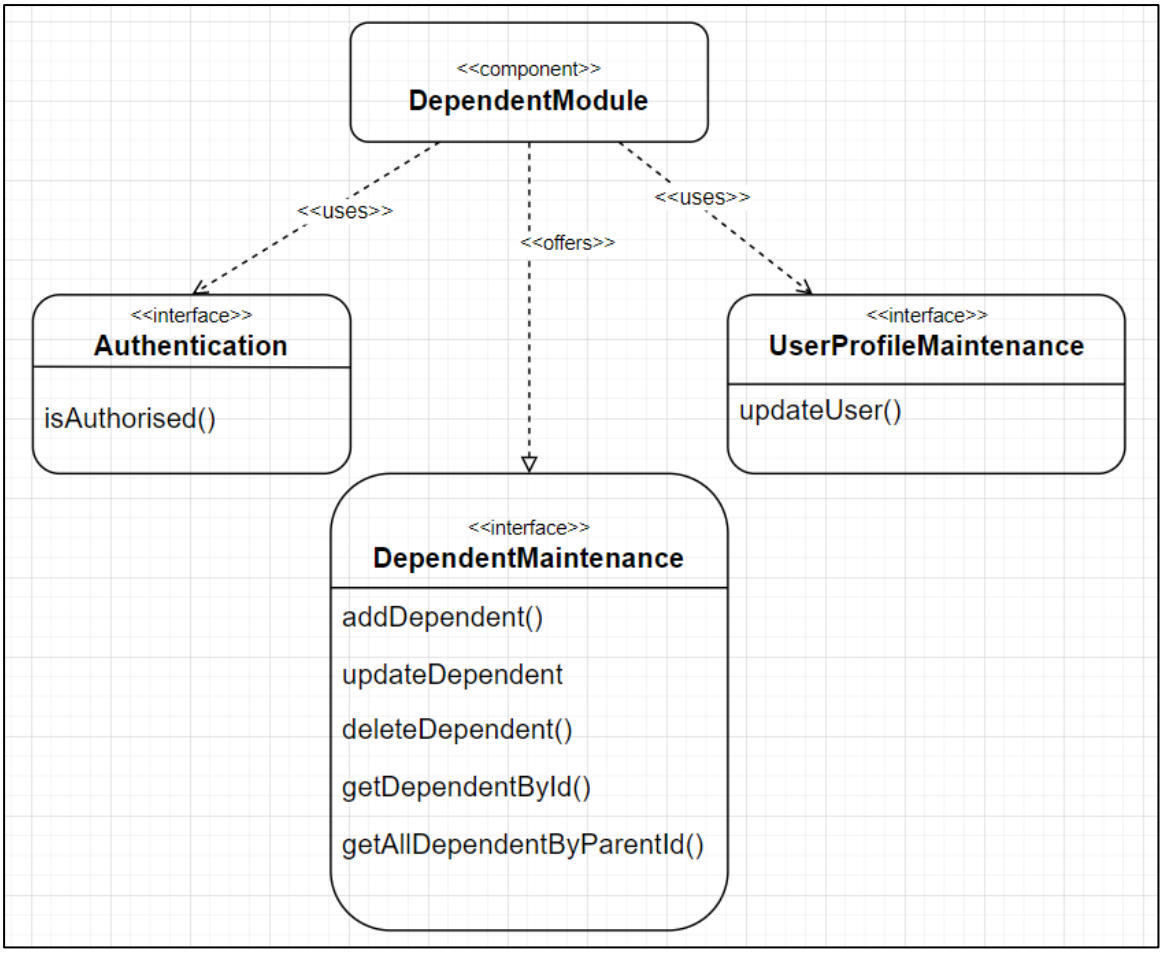


Figure 4.4 Component specification diagram of dependent module

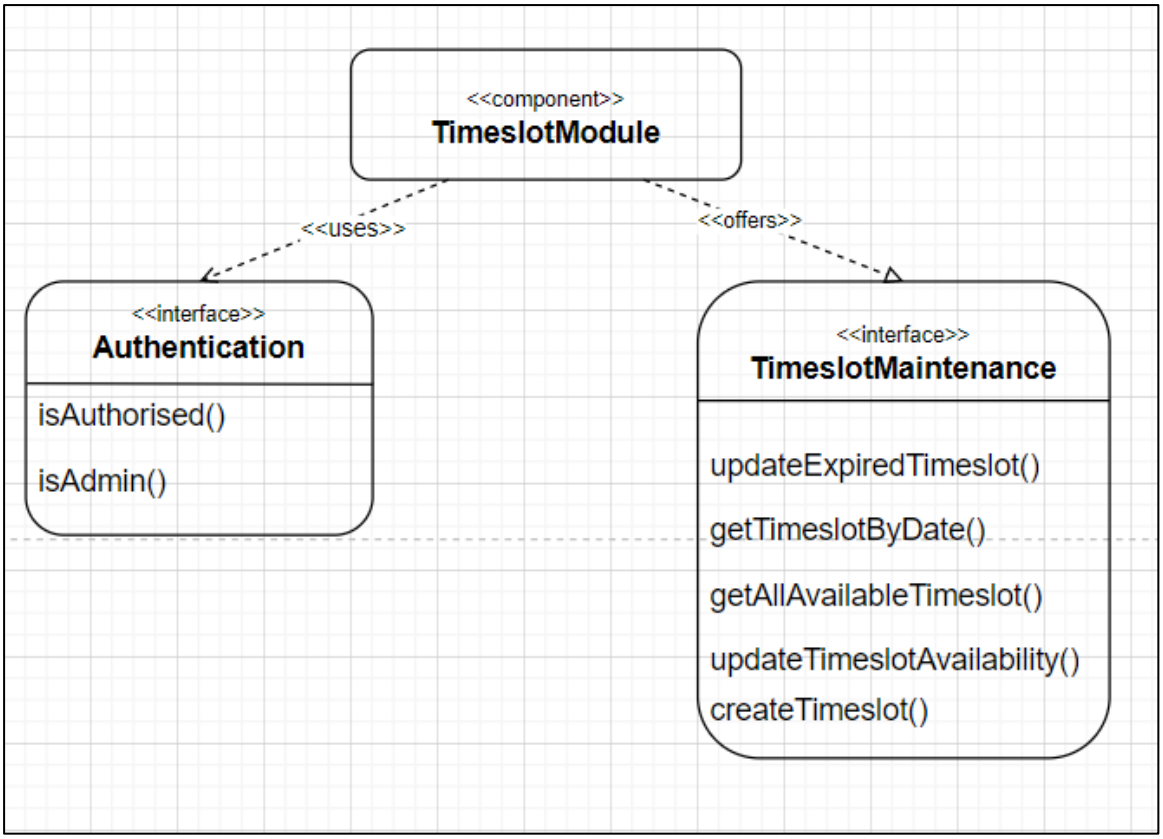


Figure 4.5 Component specification diagram of timeslot module

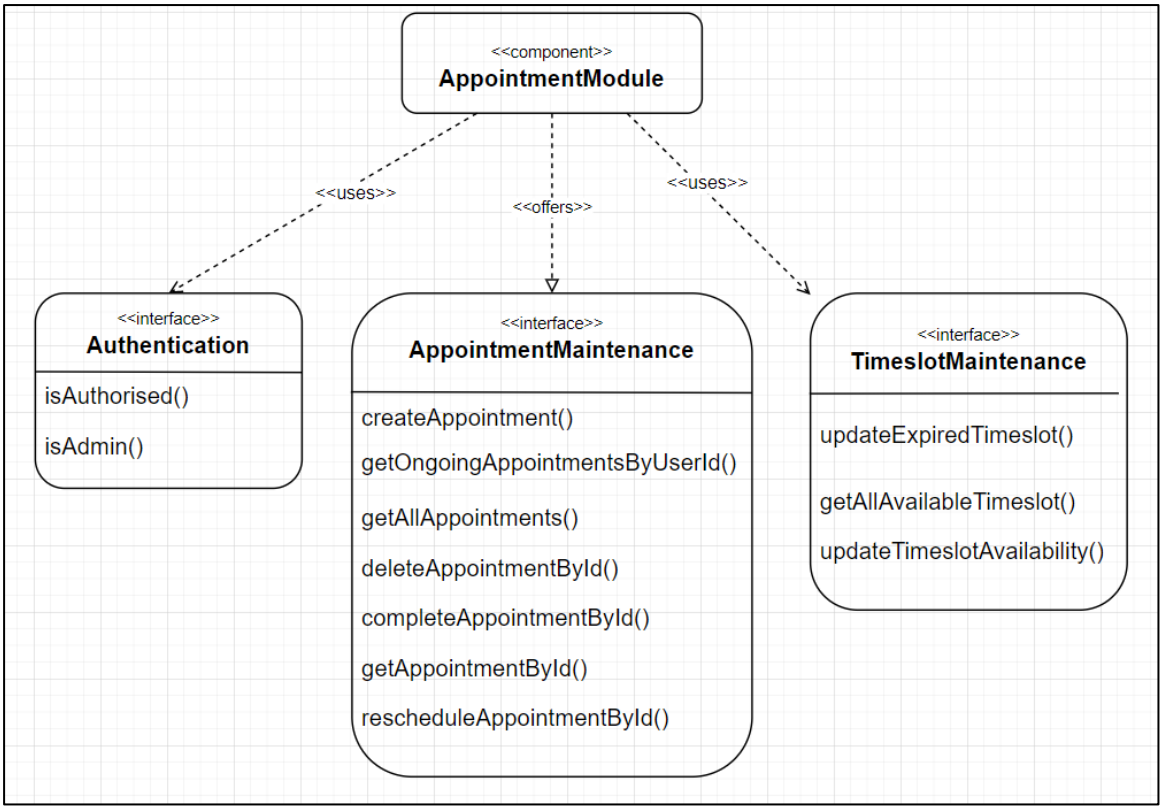


Figure 4.6 Component specification diagram of appointment module

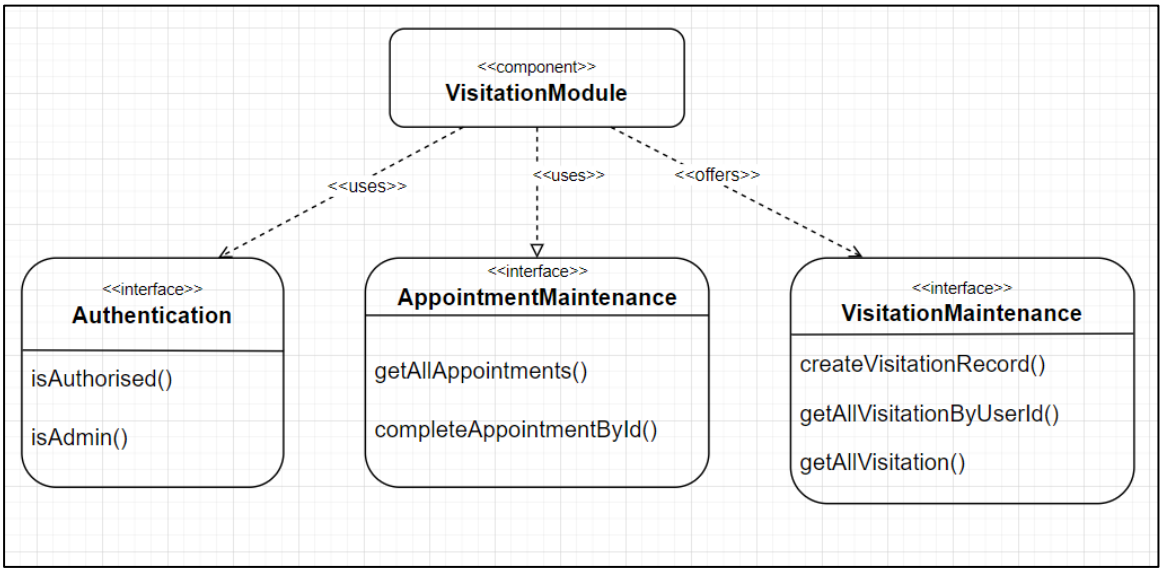


Figure 4.7 Component specification diagram of visitation module

4.3 System Components Interaction Operations

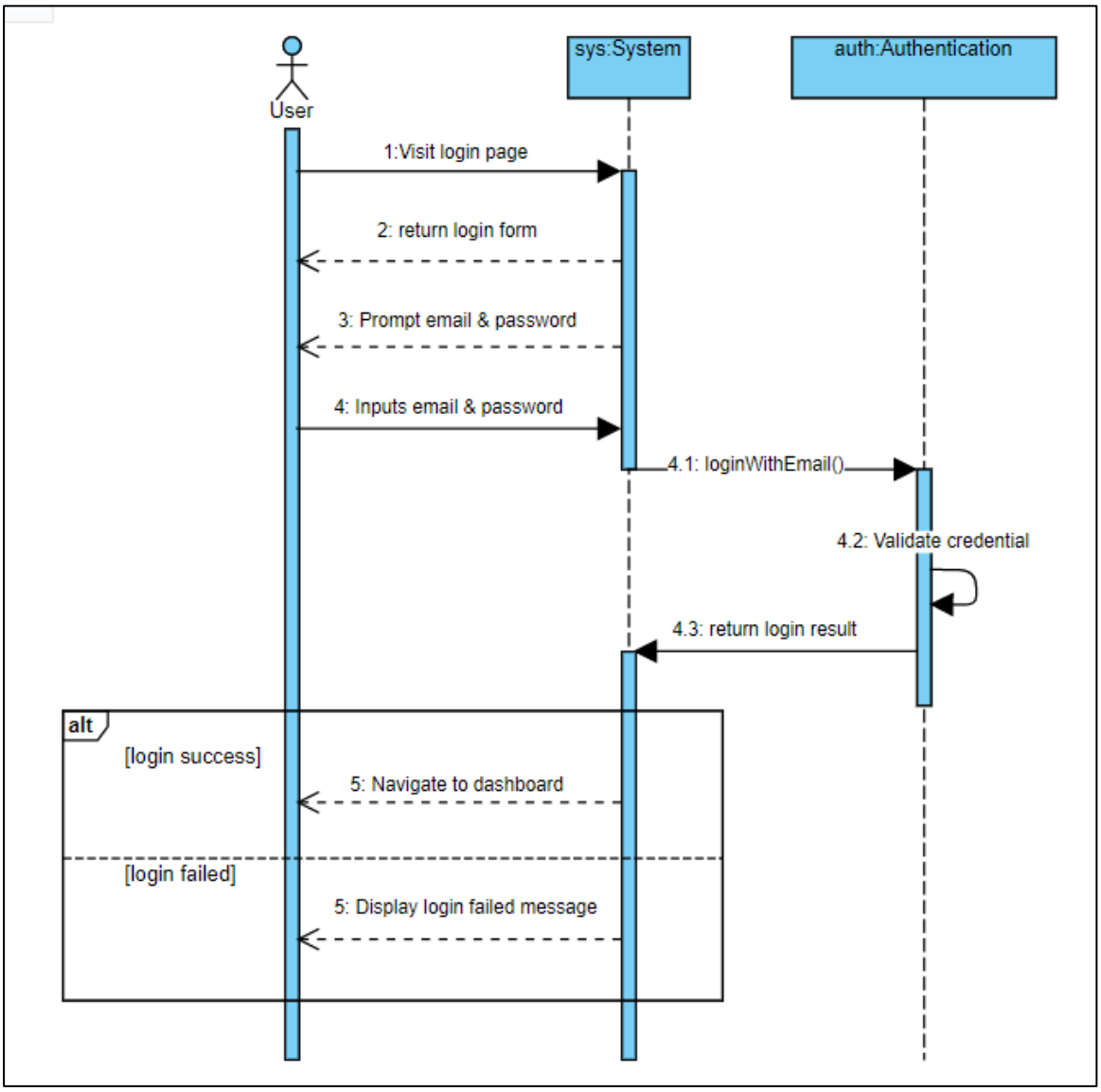


Figure 4.8 Component interaction diagram of login to system

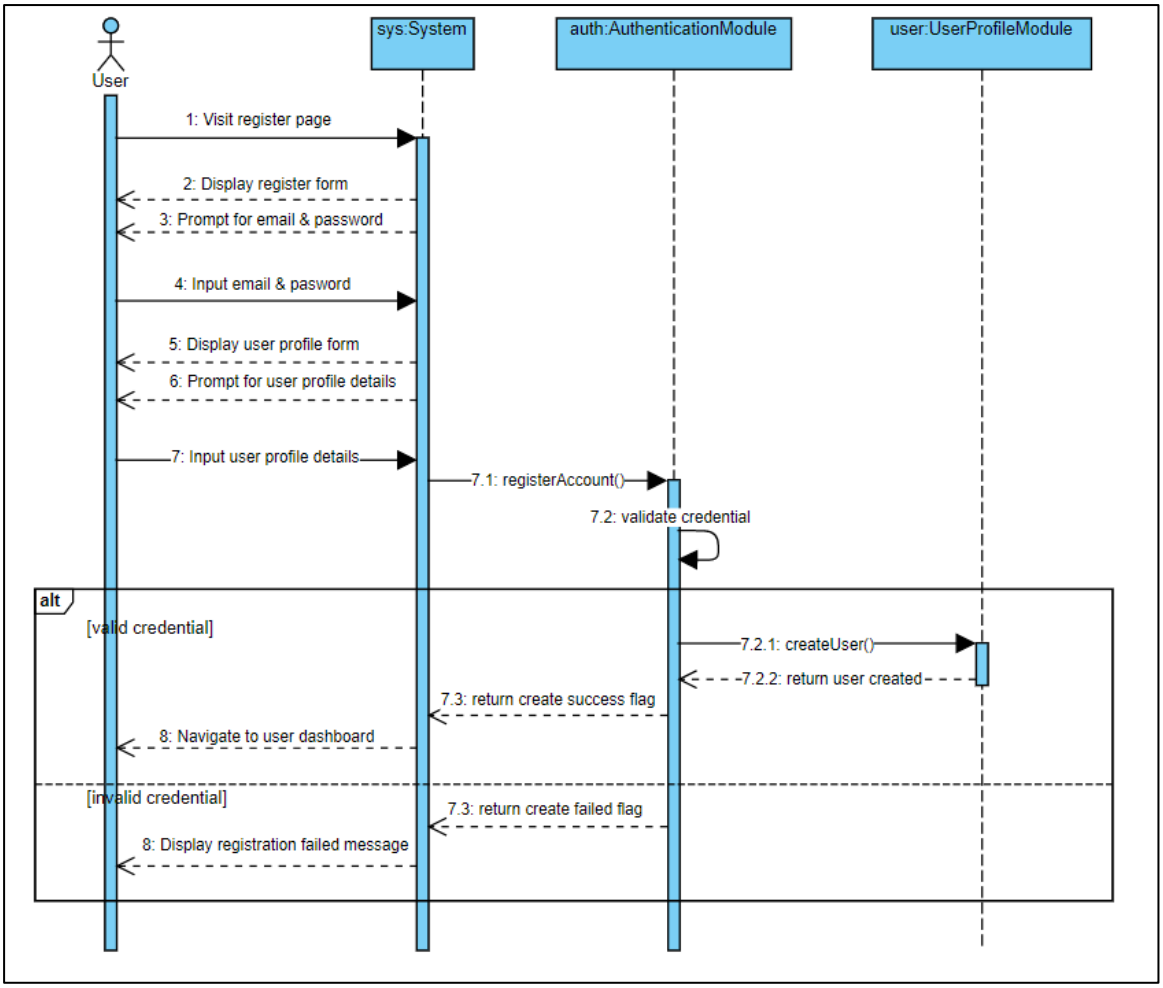


Figure 4.9 Component interaction diagram of register account

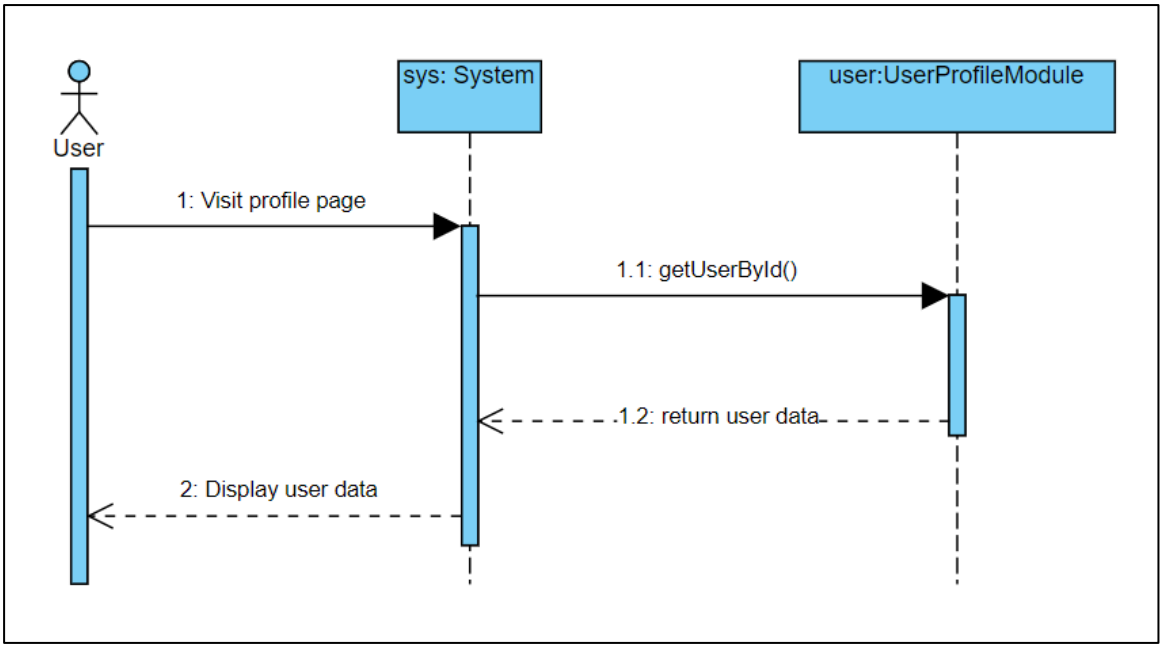


Figure 4.10 Component interaction diagram of view profile

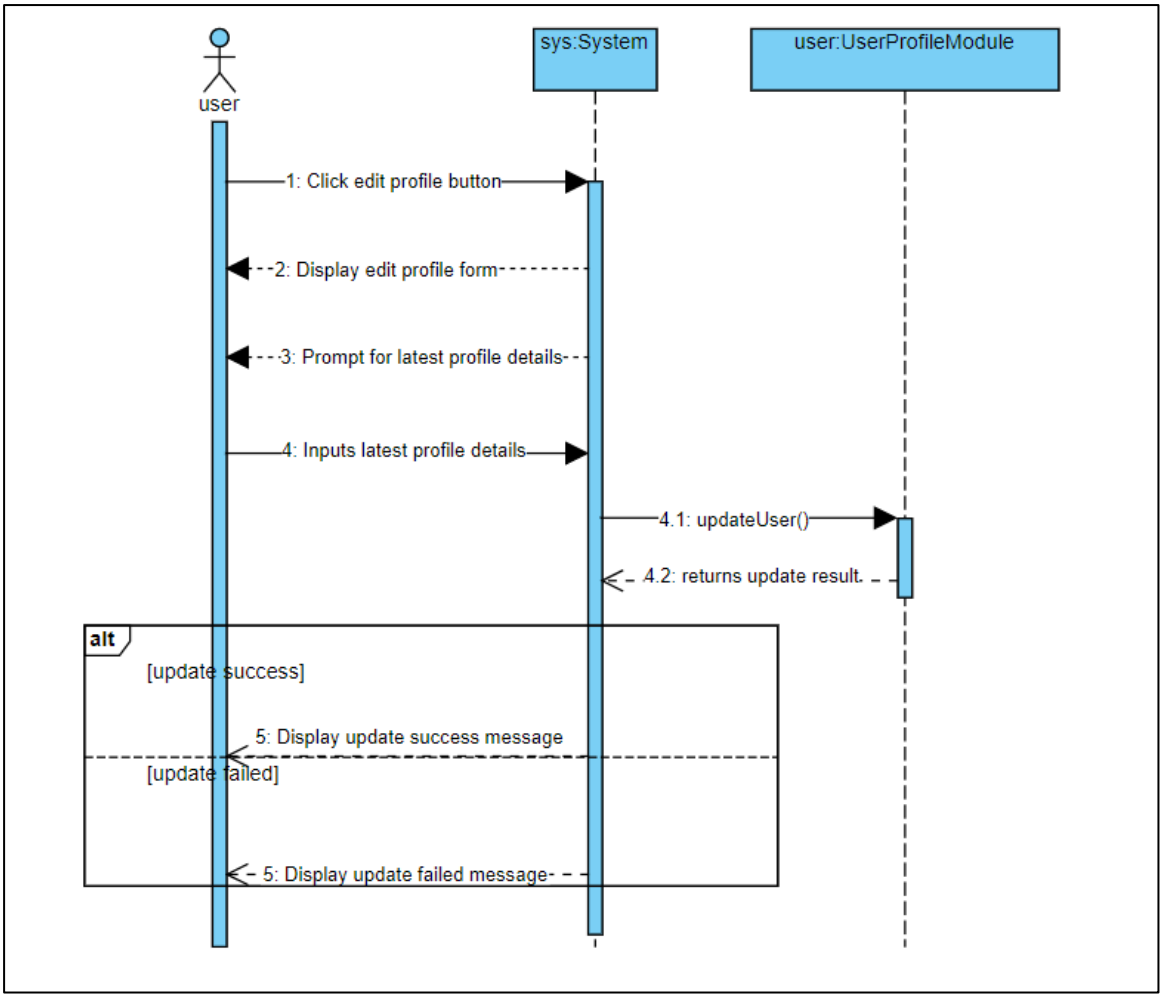


Figure 4.11 Component interaction diagram of update profile

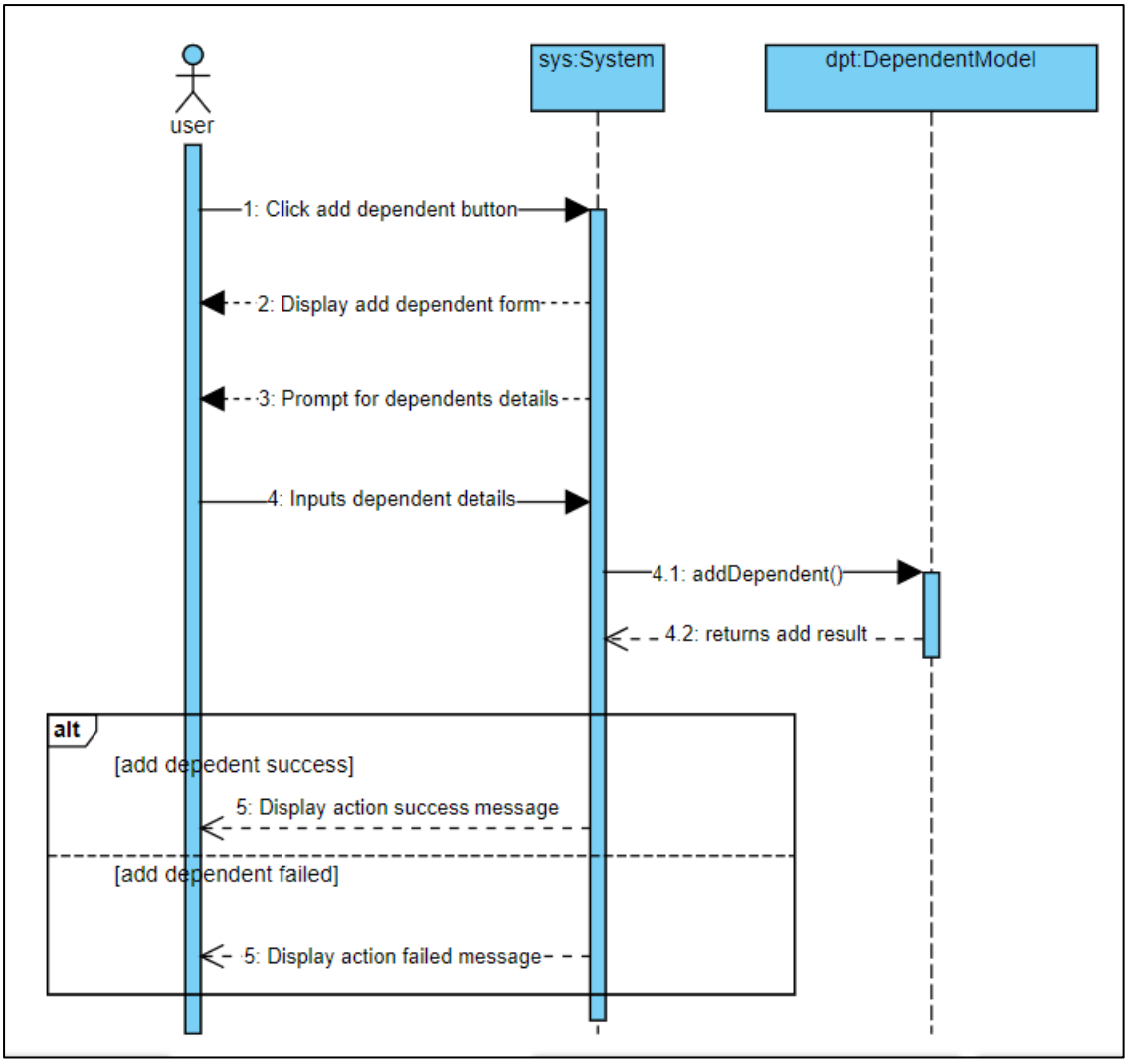


Figure 4.12 Component interaction diagram of create dependent

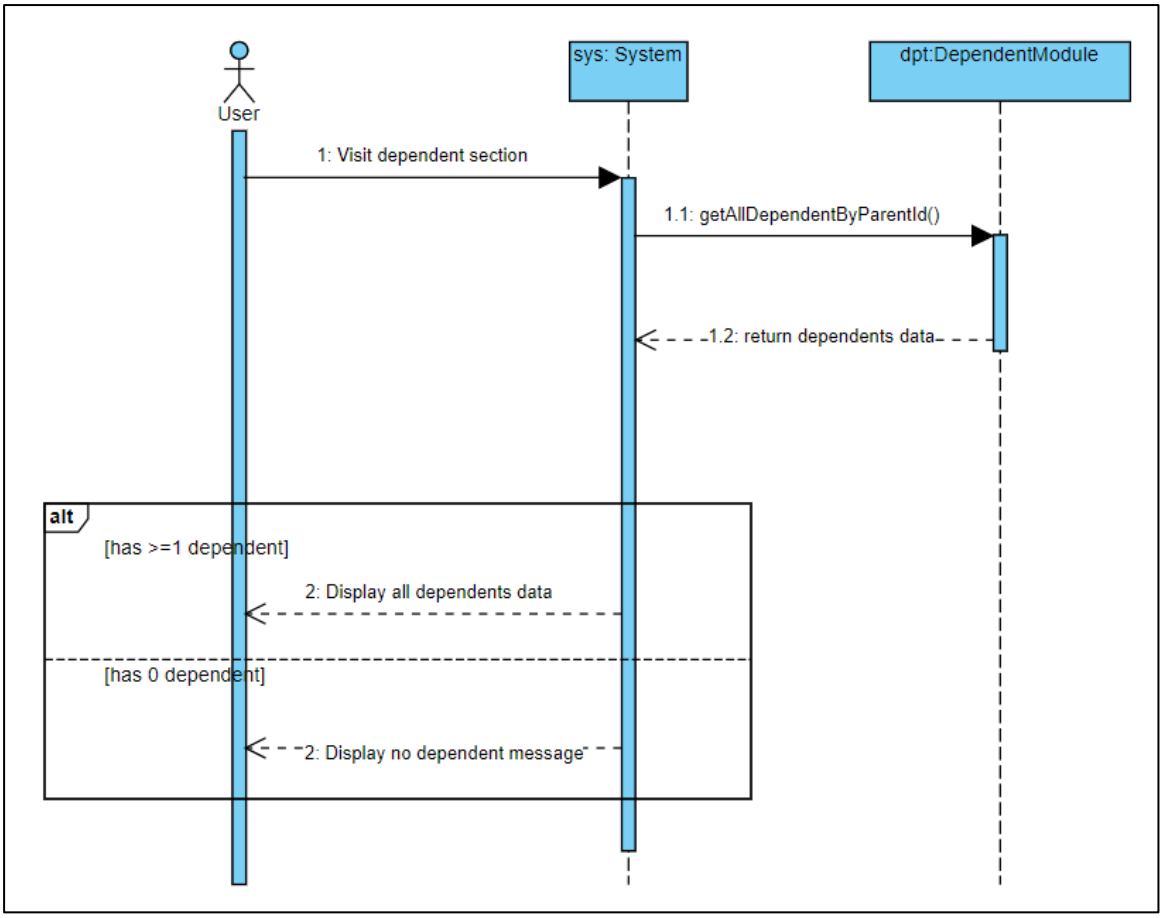


Figure 4.13 Component interaction diagram of view dependent

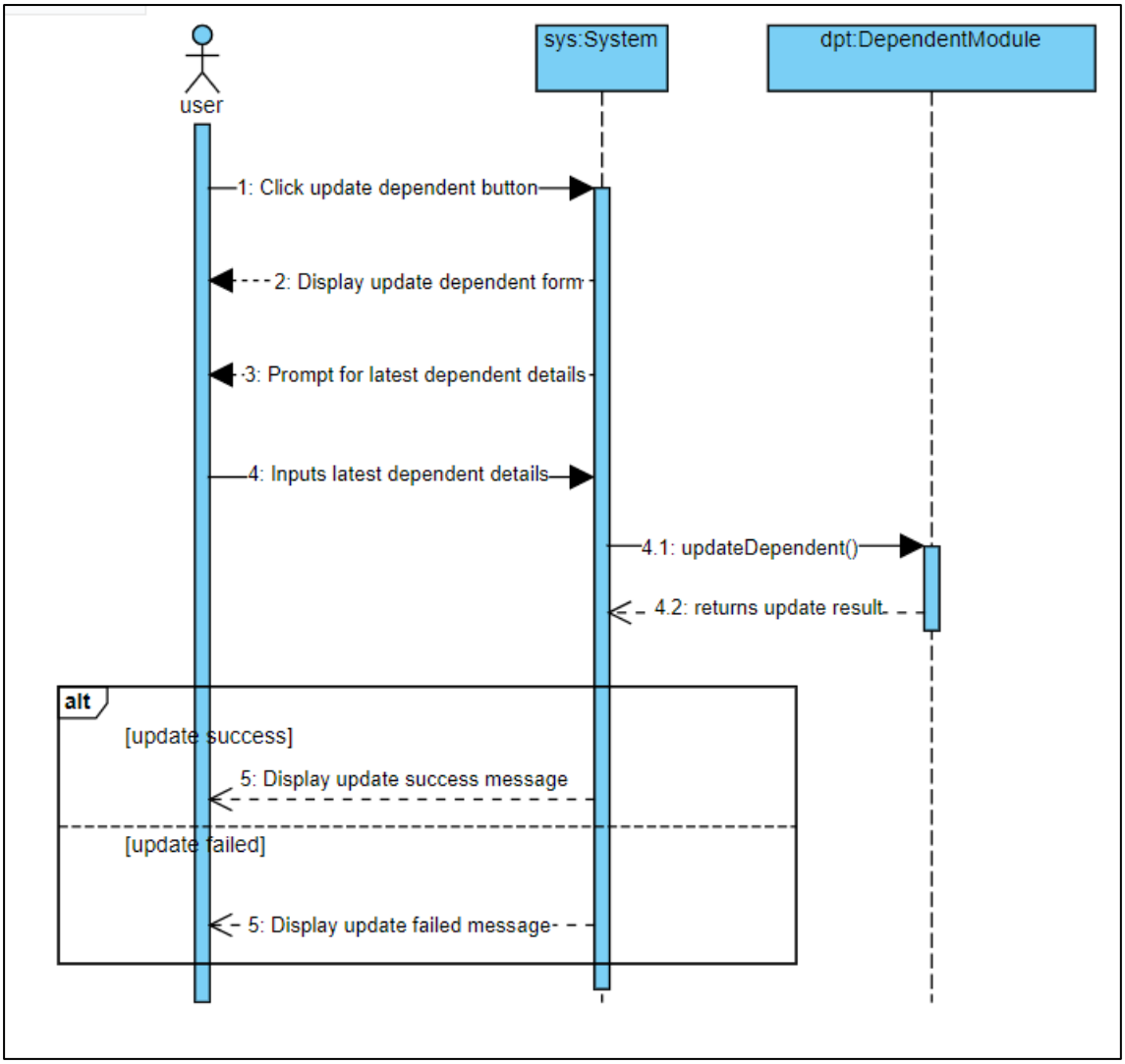


Figure 4.14 Component interaction diagram of update dependent

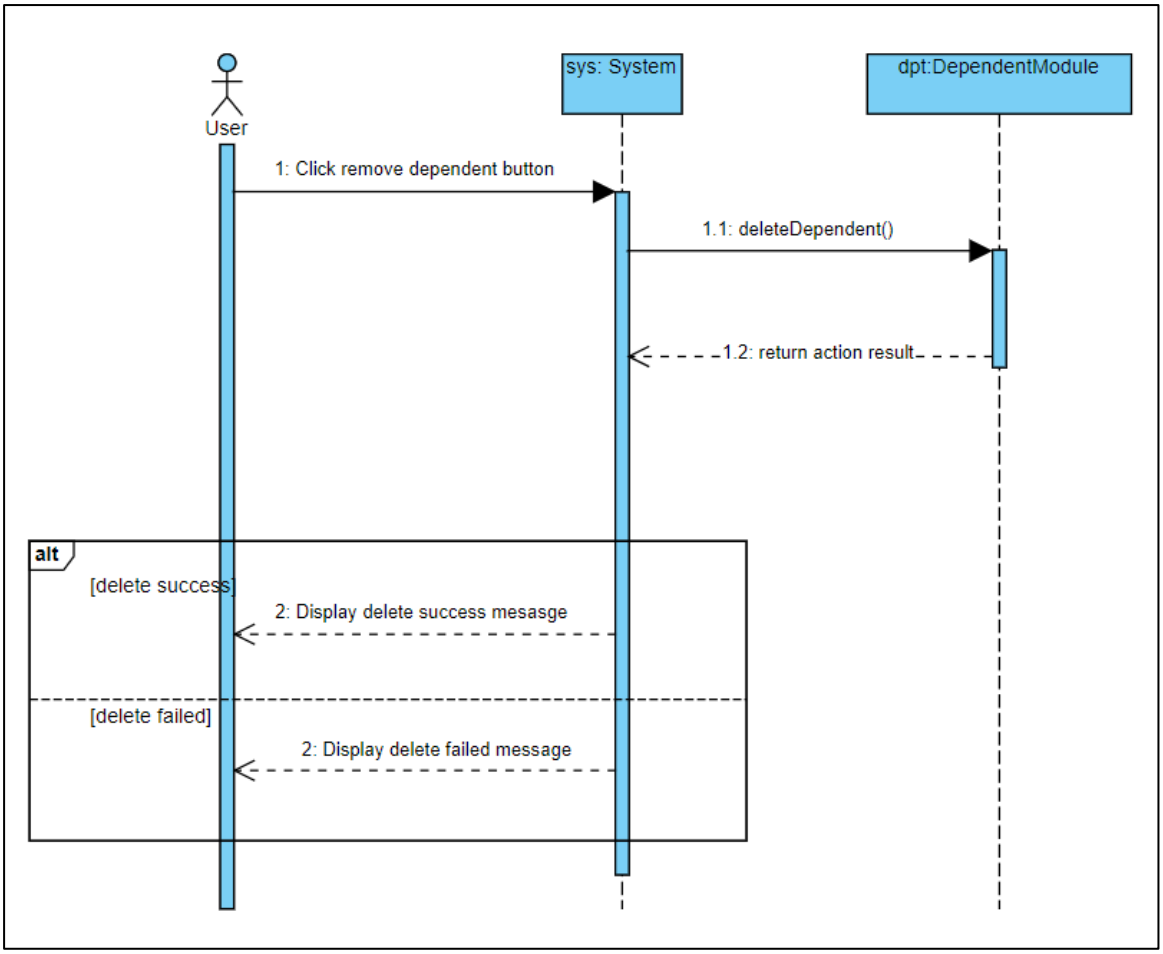


Figure 4.15 Component interaction diagram of delete dependent

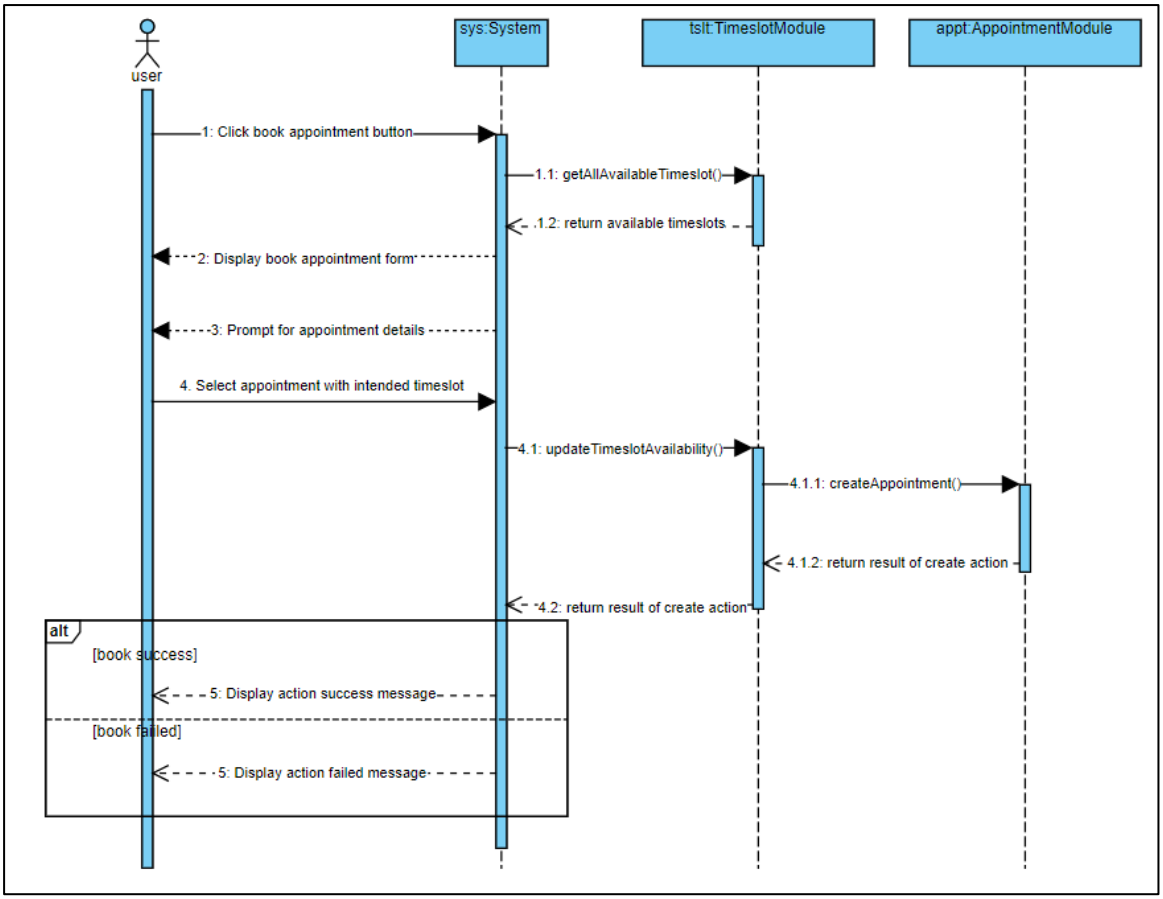


Figure 4.16 Component interaction diagram of book appointment

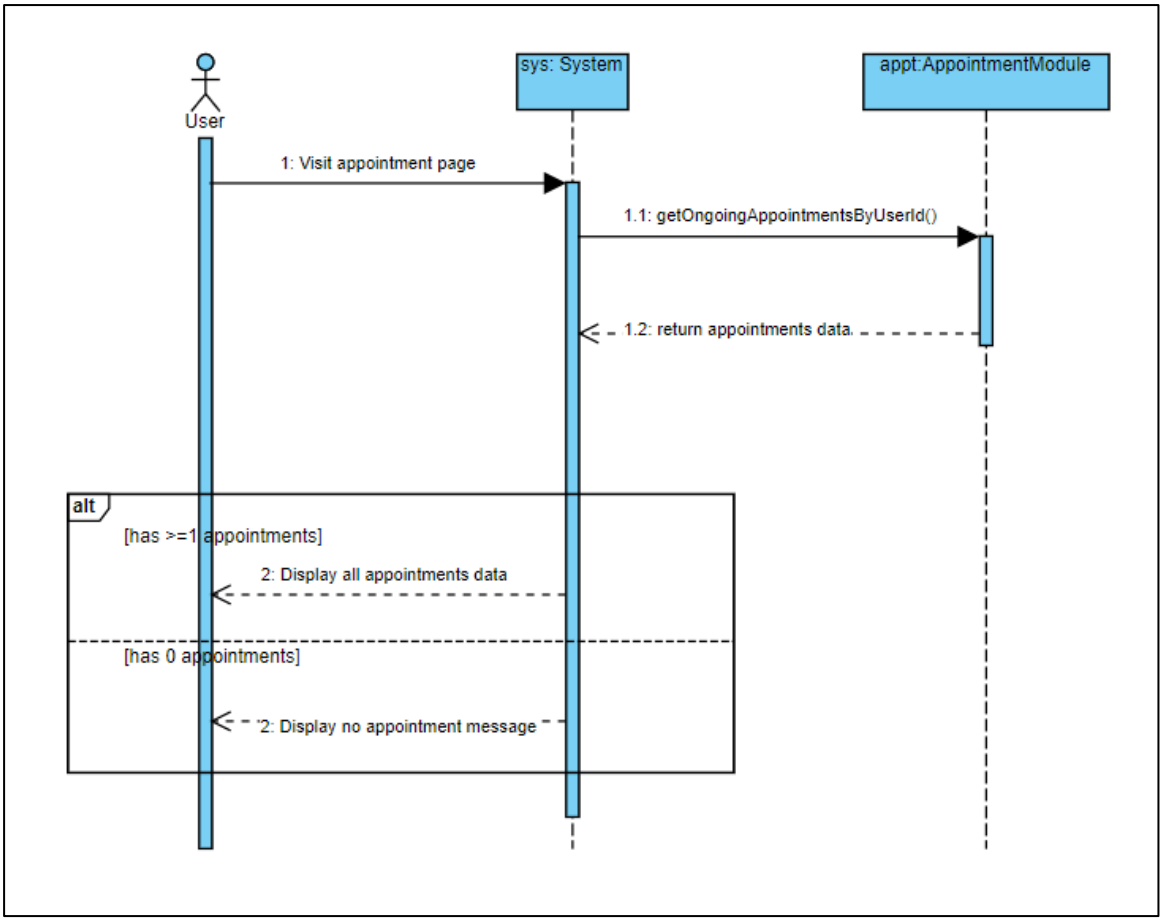


Figure 4.17 Component interaction diagram of view booked appointment

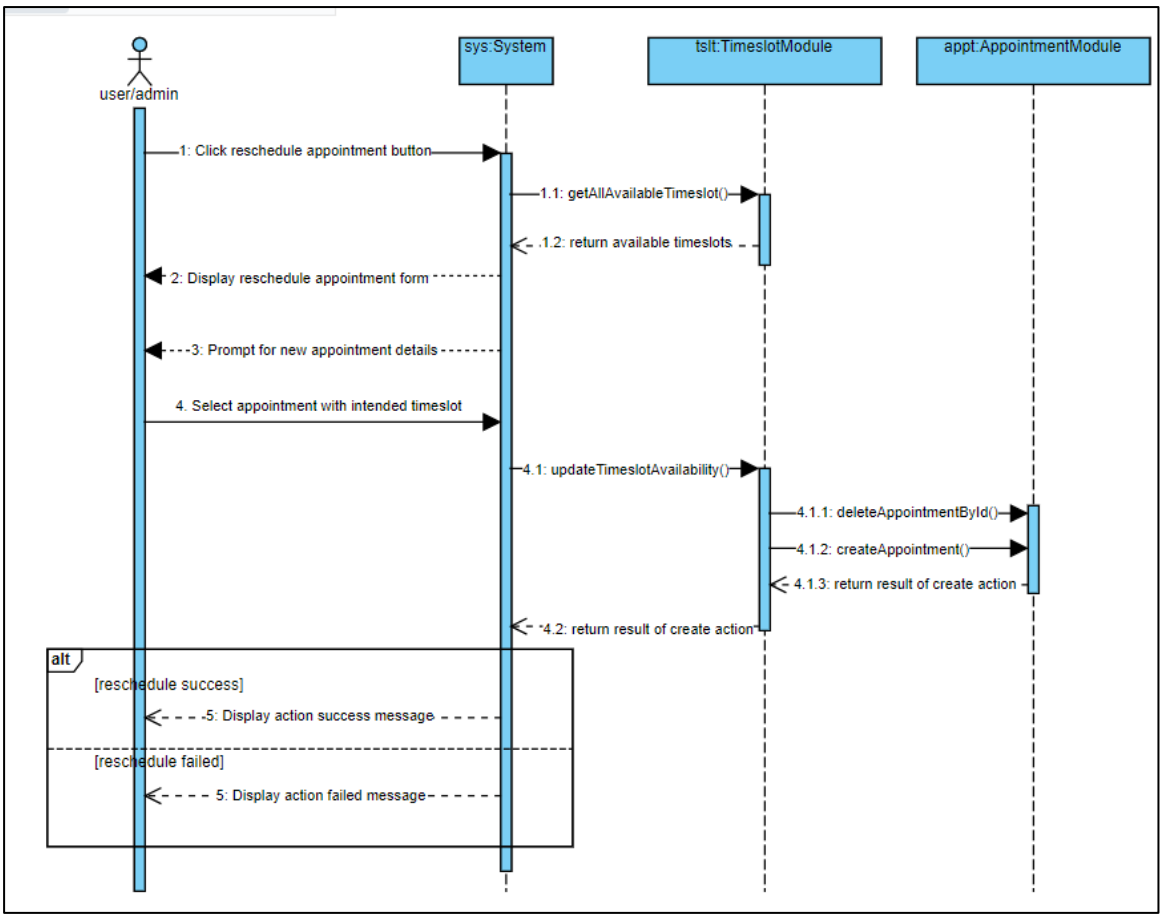


Figure 4.18 Component interaction diagram of reschedule appointment

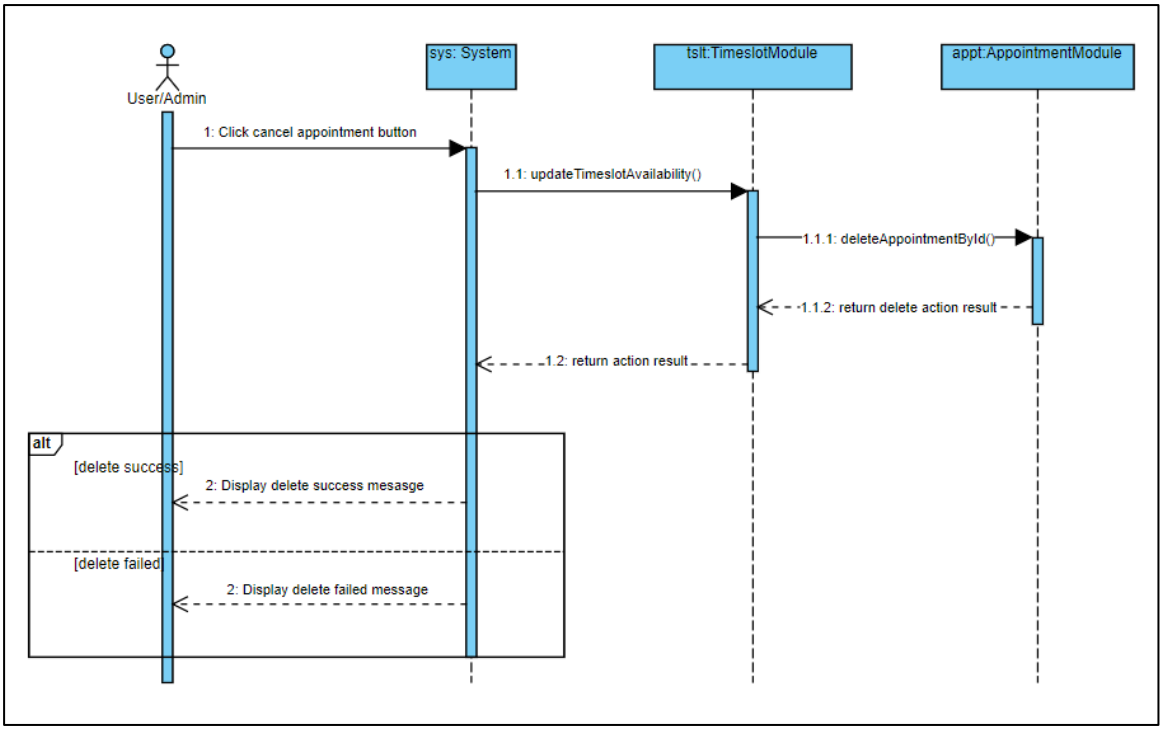


Figure 4.19 Component interaction diagram of cancel appointment

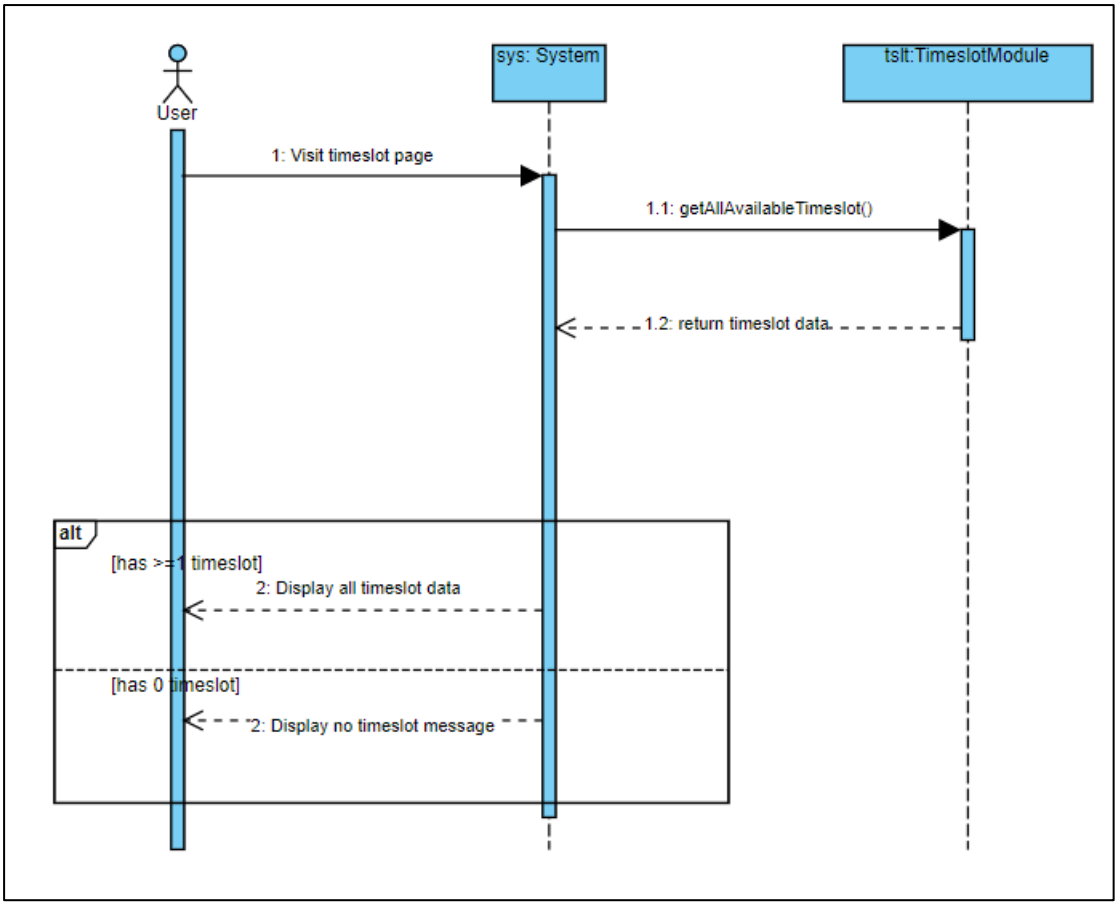


Figure 4.20 Component interaction diagram of view available timeslot

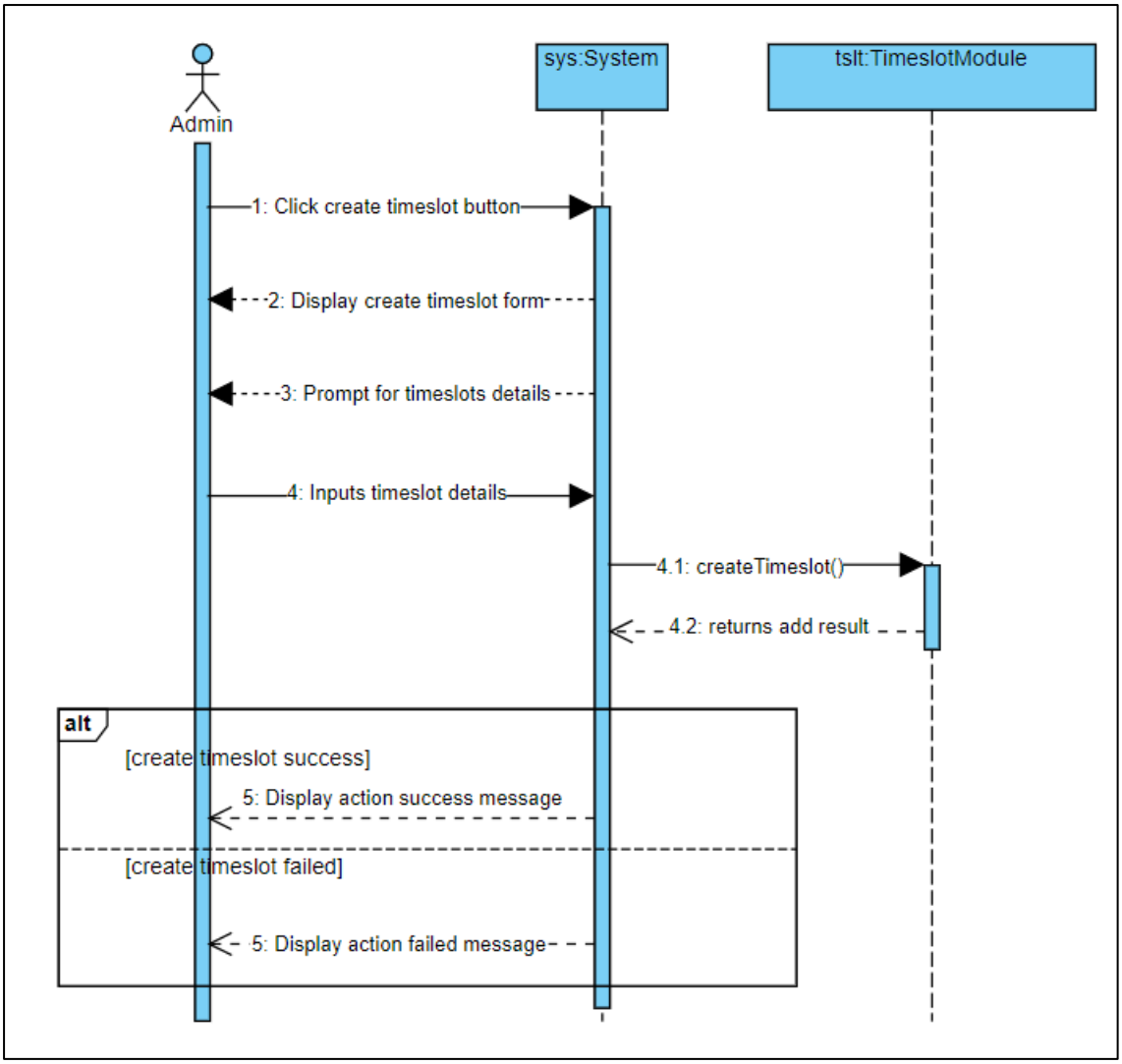


Figure 4.21 Component interaction diagram of create timeslot

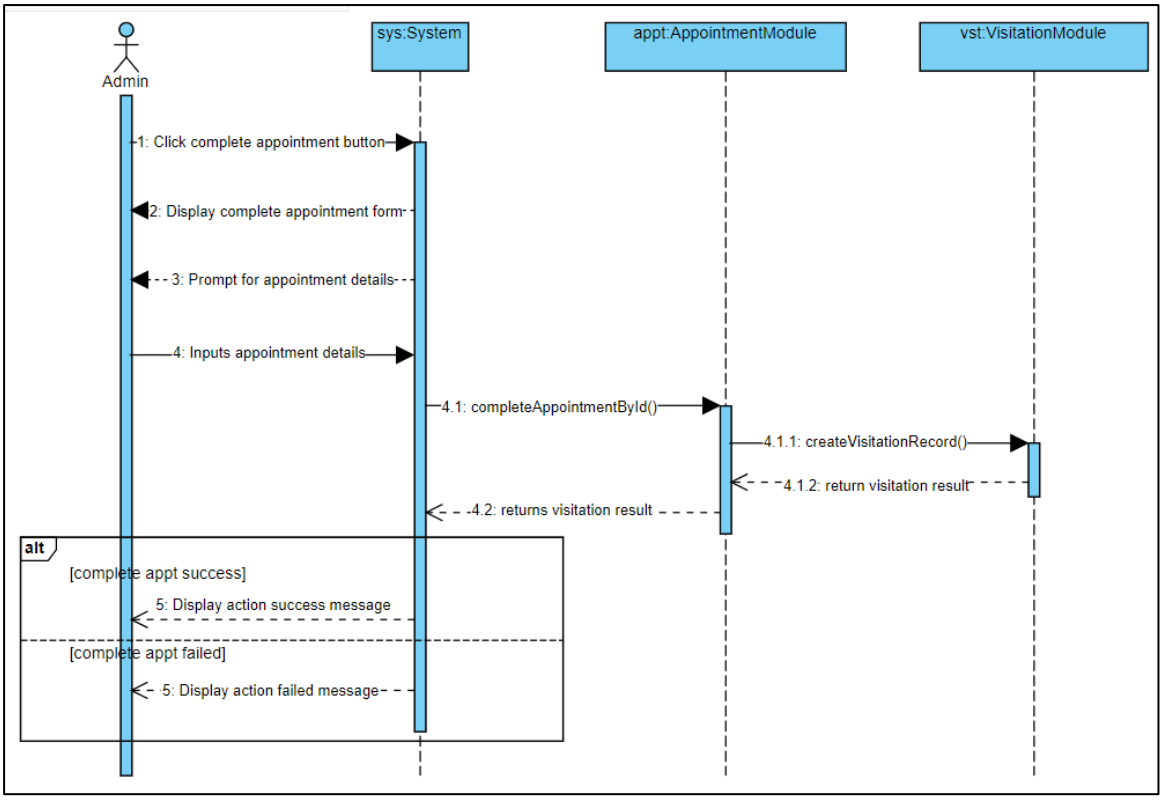


Figure 4.22 Component interaction diagram of complete appointment

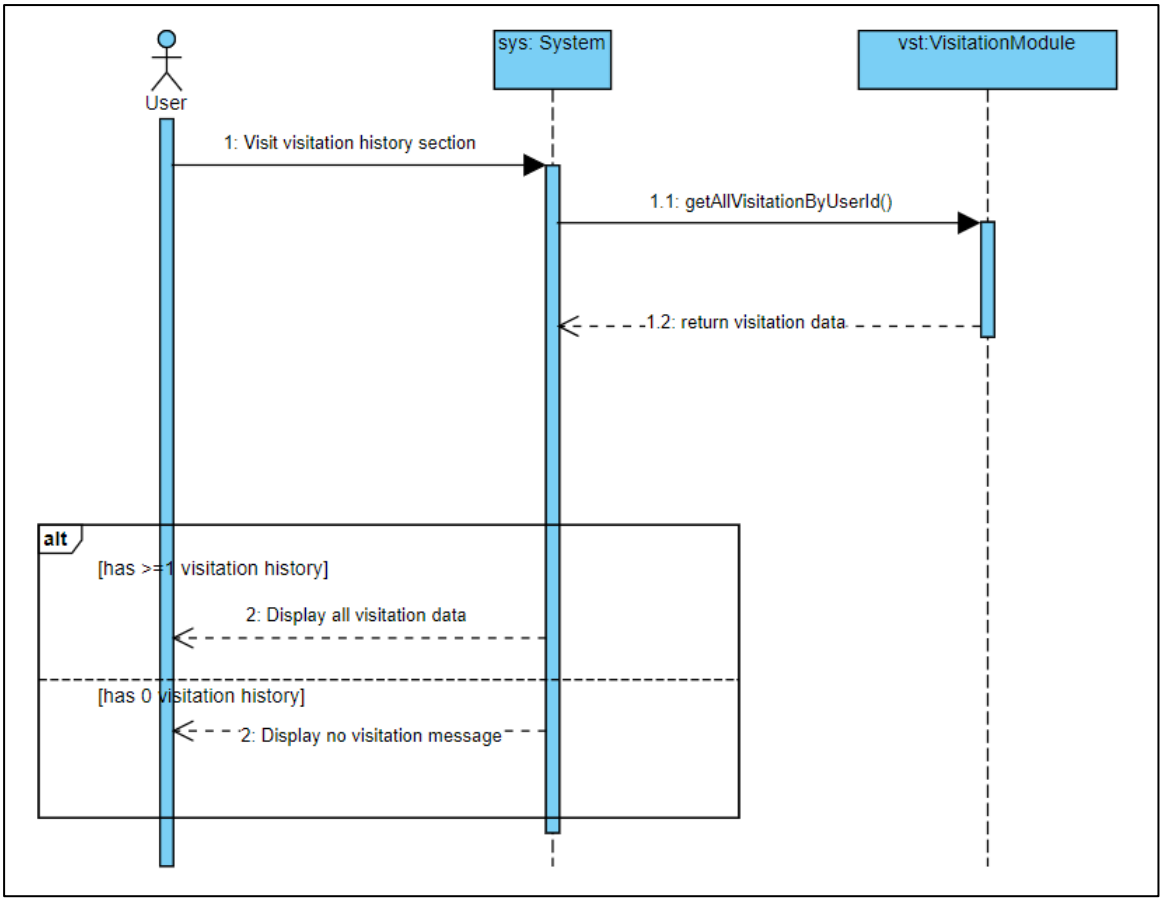


Figure 4.23 Component interaction diagram of view visitation record

4.4 Data Dictionary

The following tables illustrates the model structure of each collection in the Cloud Firestore. Each of the record will be identified using a random unique identifier generated by Cloud Firestore which is recognized as the documentId. An exception is applied to the user record where the user record will be uniquely identified by a random id generated by Firebase Authentication, which has a length of 28 characters instead of 20 characters as the one generated by Cloud Firestore. With respect to the nature of NoSQL database, there will be no foreign key relationship be enforced internally using database relationship. However, some dependencies between collections have been defined and be enforced as self-defined relationship to fulfill the system requirements.

Collection: User

Field	Type	Size	Null	Description	Example
userID (PK)	String	28	No	Unique identifier of user record	KaK3WFI63lTubqqMpgpZ6DxSfNv1
firstName	String	255	No	First name of the user	Marine
lastName	String	255	No	Last name of the user	Bob
gender	String	6	No	Gender of the user	Female
icNum	String	12	No	IC number of the user	000000112222
birthDate	Date	255	No	Birthdate of the user	1998-02-02T00:00:00.000+00:00
contactNumber	String	255	No	Contact number of the user	12345678910
email	String	255	No	Email address of the user	xyz@gmail.com
address1	String	255	No	First address of the user	10, Jalan XYZ
address2	String	255	Yes	Second address of the user	Village JKL
address3	String	255	Yes	Third address of the user	City P
iconUrl	String	255	No	Avatar URL of the user	https://picsum.photos/200/300
admin	boolean	-	No	The identity of the user	False
createTs	Timestamp	-	No	Audit field that records the timestamp when user is created	1998-02-02T00:00:00.000+00:00
createUser	String	255	No	Audit field that records the user id when user is created	KaK3WFI63lTubqqMpgpZ6DxSfNv1
updateTs	Timestamp	-	Yes	Audit field that records the timestamp when user is updated	1998-02-02T00:00:00.000+00:00
updateUser	String	255	Yes	Audit field that records the user id when user is updated	KaK3WFI63lTubqqMpgpZ6DxSfNv1

Table 4.1 Data dictionary of the User collection

Collection: Dependent

Field	Type	Size	Null	Description	Example
dependentId (PK)	String	20	No	Unique identifier of dependent record	aIoWZDiY74KKu8XVphu6
parentId (FK)	String	28	No	User id of the dependent parent	KaK3WFI63lTubqqMpgpZ6DxSfNv1
firstName	String	255	No	First name of the dependent	Zhen
lastName	String	255	No	Last name of the dependent	Ty
gender	String	6	No	Gender of the dependent	Male
icNum	String	12	No	IC number of the dependent	000000112222
birthDate	Date	255	No	Birthdate of the dependent	1998-02-02T00:00:00.000+00:00
avatarUrl	String	255	No	Avatar URL of the dependent	https://picsum.photos/200/300
allergies	String	255	No	Allergies of the dependent if any	Nut products
createTs	Timestamp	-	No	Audit field that records the timestamp when dependent is created	1998-02-02T00:00:00.000+00:00
createUser	String	255	No	Audit field that records the user id when dependent record is created	KaK3WFI63lTubqqMpgpZ6DxSfNv1
updateTs	Timestamp	-	Yes	Audit field that records the timestamp when dependent is updated	1998-02-02T00:00:00.000+00:00
updateUser	String	255	Yes	Audit field that records the user id when dependent is updated	KaK3WFI63lTubqqMpgpZ6DxSfNv1

Table 4.2 Data dictionary of the Dependent collection

Collection: Timeslot

Field	Type	Size	Null	Description	Example
timeslotId (PK)	String	20	No	Unique identifier of timeslot record	aIoWZDiY74KKu8XVphu6
isAvailable	boolean	-	No	Flag to indicate whether the timeslot is available	true
timeslotDateInMillis	number	13	No	Date of the timeslot recorded in milliseconds	1649692800000
sessionStartTimeInMillis	number	13	No	Timeslot session time recorded in milliseconds	1649692800000
createTs	Timestamp	-	No	Audit field that records the timestamp when timeslot is created	1998-02-02T00:00:00.000+00:00
createUser	String	255	No	Audit field that records the user id when timeslot record is created	KaK3WFI63lTubqqMpgpZ6DxSfNv1
updateTs	Timestamp	-	Yes	Audit field that records the timestamp when timeslot is updated	1998-02-02T00:00:00.000+00:00
updateUser	String	255	Yes	Audit field that records the user id when timeslot is updated	KaK3WFI63lTubqqMpgpZ6DxSfNv1

Table 4.3 Data dictionary of the Timeslot collection

Collection: Appointment

Field	Type	Size	Null	Description	Example
apptId (PK)	String	20	No	Unique identifier of appointment record	aIoWZDiY74KKu8XVphu6
timeslotId (FK)	String	20	No	Id of the appointment's timeslot	0PoT1DiY74KKu2GVp0PH
userId (FK)	String	28	No	User id to indicate the owner of the appointment	KaK3WFI63lTubqqMpgpZ6DxSfNv1
dependentId (FK)	String	20	No	Dependent id to indicate the main target of the appointment	aIoWZDiY74KKu8XVphu6
sessionStartTimeInMillis	number	13	No	Appointment session time recorded in milliseconds	1649692800000
isCompleted	boolean	-	No	Flag to indicate whether the appointment is completed	true
description	String	255	Yes	Represents any remark made when the user book the appointment	Body checkup
createTs	Timestamp	-	No	Audit field that records the timestamp when appointment is created	1998-02-02T00:00:00.000+00:00
createUser	String	255	No	Audit field that records the user id when appointment record is created	KaK3WFI63lTubqqMpgpZ6DxSfNv1
updateTs	Timestamp	-	Yes	Audit field that records the timestamp when appointment is updated	1998-02-02T00:00:00.000+00:00
updateUser	String	255	Yes	Audit field that records the user id when appointment is updated	KaK3WFI63lTubqqMpgpZ6DxSfNv1

Table 4.4 Data dictionary of the Appointment collection

Collection: Visitation

Field	Type	Size	Null	Description	Example
visitId (PK)	String	20	No	Unique identifier of visitation record	aIoWZDiY74KKu8XVphu6
apptId (FK)	String	20	yes	Appointment id that the visitation relevant to	0PoT1DiY74KKu2GVp0PH
userId (FK)	String	28	No	User id to indicate the owner of the visitation	KaK3WFI63lTubqqMpgpZ6DxSfNv1
dependentId (FK)	String	20	No	Dependent id to indicate the main target of the visitation	aIoWZDiY74KKu8XVphu6
sessionDateTime	number	13	No	Date and time of the visitation	1649692800000
diagnosisResult	String	255	No	Diagnosis result of the visitation session	Headache
prescription	String	255	Yes	The prescription given to the user according to the diagnosis result	Paracetamol
createTs	Timestamp	-	No	Audit field that records the timestamp when visitation is created	1998-02-02T00:00:00.000+00:00
createUser	String	255	No	Audit field that records the user id when visitation record is created	KaK3WFI63lTubqqMpgpZ6DxSfNv1
updateTs	Timestamp	-	Yes	Audit field that records the timestamp when visitation is updated	1998-02-02T00:00:00.000+00:00
updateUser	String	255	Yes	Audit field that records the user id when visitation is updated	KaK3WFI63lTubqqMpgpZ6DxSfNv1

Table 4.5 Data dictionary for Visitation collection

The visitation collection has a weak relationship with the appointment collection, whereby a visitation record can still be generated without prior appointment to cover those emergency treatment without appointment.

Chapter 5 System Implementation

5.1 Hardware Setup

Model	Laptop – Y520-151KBN
CPU	Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
GPU	NVIDIA GeForce GTX 1050 2 GB
RAM	Kingston 12.0 GB
Hard Disk	1 TB
Manufacturer	Lenovo

Table 5.1 The specification of the hardware required

5.2 Software Setup

No.	Software	Purpose
1.	Docker Desktop	To allow the building and pulling of Docker Images from the repository
2.	Edge Web Browser	To browse the developed web application

Table 5.2 Software required for the system implementation

5.3 Setting and Configuration

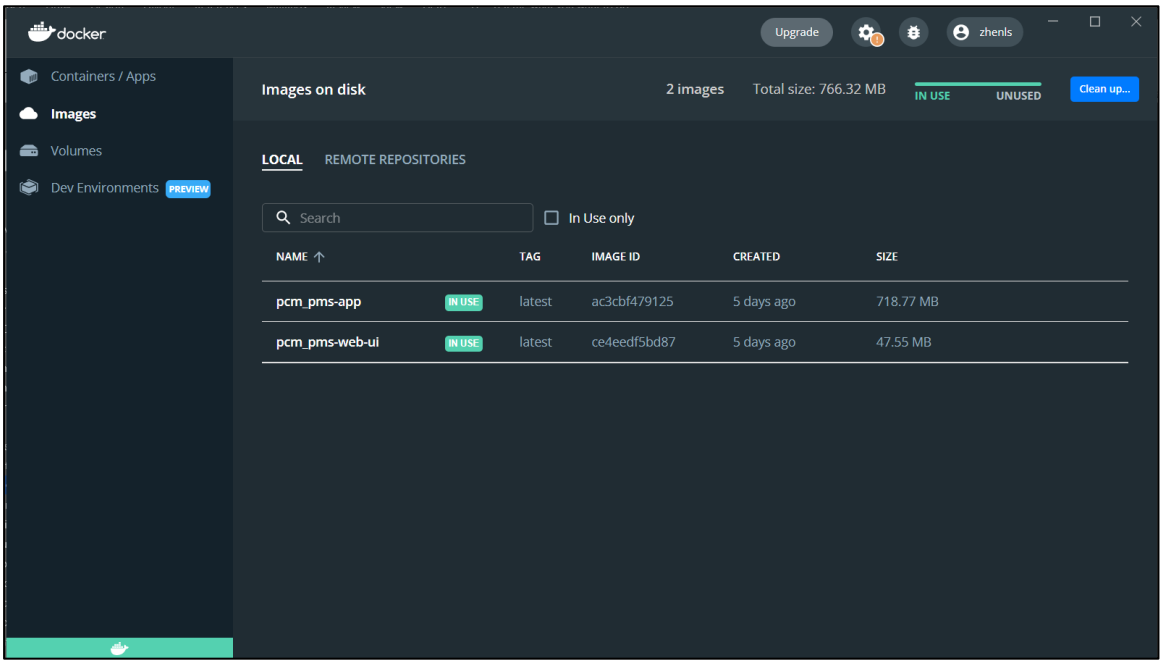


Figure 5.1 Docker images pulled from repository

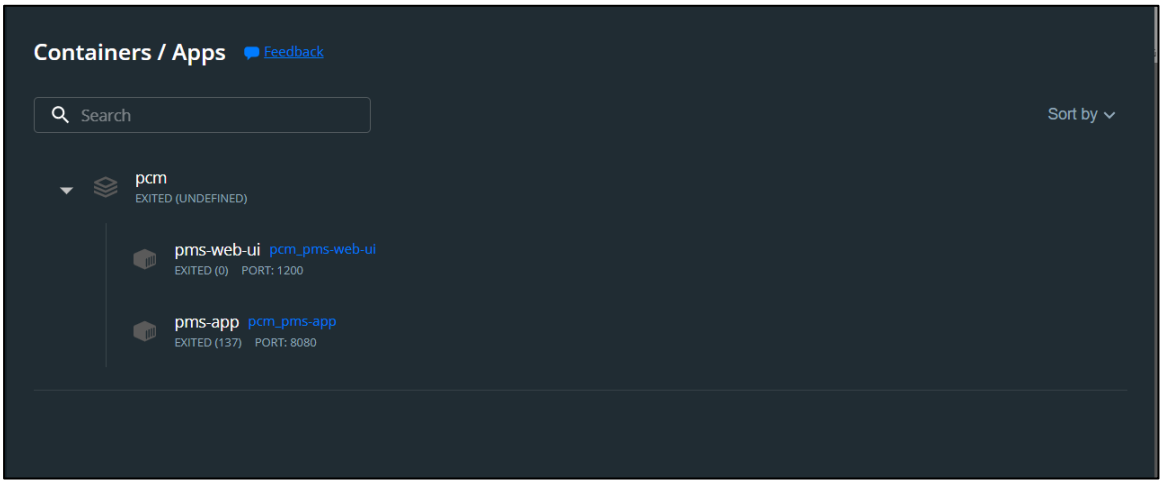


Figure 5.2 Docker container spawned from the pulled images

In order to execute the application from the Docker Images, the Docker Images have to be pulled from the Docker repository priorly. After that, new Docker Container can be spawned using the downloaded Docker images, making the application available in the local machine. With the container setup, the web application can be accessed using Edge browser using the URL of http://localhost:1200, as specified in the docker-compose file.

5.4 System Operation

With the system methodology and design prepared, the system has been undergoing an implementation phase. The system design made in the previous section has been realized through several technologies, ranging from Flutter Web, Spring Boot MVC to Docker. Each of these technologies reserved responsibility in different part, and the final application are formed through integrating these components. The following subsections will explain the functionality and UI of the application along with the relevant technologies applied.

5.4.1 Dockerization

As described in the system methodology, the final product, Patient Management System is encapsulated into single functional web application using Docker technologies. Docker significantly reduced the configuration and integration efforts when the application is intended to be deployed in the production stage. In order to take advantages from this characteristic, there are several settings and preparations have to be conducted in advance. First of all, Docker requires a Dockerfile which contains the configuration information to build an application template, which normally be recognized as Docker Images. With respect to the configuration stated in Dockerfile, the Docker Image built can be referred as a template to spawn multiple Docker Containers from it. Figures below illustrates the Dockerfile from both front-end and back-end application.

Dockerfile > ...

```
1   # Get base OS with openjdk
2   FROM openjdk:11
3
4   # Make the 'app' folder the current working directory
5   WORKDIR /app
6
7   # Copy artifact to the current working directory (i.e. 'app' folder)
8   COPY build/libs .
9
10  # When run the docker container, this command will execute and start the program
11  ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -jar ./*.jar;" ]
12
13  # EXPOSE listening port
14  EXPOSE 8080
```

Figure 5.3 Dockerfile of back-end application

pcs_frontend > pcs_frontend_application > Dockerfile > ...

```
#Stage 1 - Install dependencies and build the app
FROM debian:latest AS build-env

# Install flutter dependencies
RUN apt-get update
RUN apt-get install -y curl git wget unzip libgconf-2-4 gdb libstdc++6 libglu1-mesa fonts-droid-fallback lib32stdc++6 python3
RUN apt-get clean

# Clone the flutter repo
RUN git clone https://github.com/flutter/flutter.git /usr/local/flutter

# Set flutter path
# RUN /usr/local/flutter/bin/flutter doctor -v
ENV PATH="/usr/local/flutter/bin:/usr/local/flutter/bin/cache/dart-sdk/bin:${PATH}"

# Run flutter doctor
RUN flutter doctor -v
# Enable flutter web
RUN flutter channel master
RUN flutter upgrade
RUN flutter config --enable-web

# Copy files to container and build
RUN mkdir /app/
COPY . /app/
WORKDIR /app/
RUN flutter build web

# Stage 2 - Create the run-time image
FROM nginx:1.21.1-alpine
COPY --from=build-env /app/build/web /usr/share/nginx/html

# Expose port 80
EXPOSE 80
```

Figure 5.4 Dockerfile of front-end application

It can be observed that the configuration of the Dockerfile follows some syntax. Generally, the Dockerfile is started from the FROM command, whereby the base image used as the underlying layer is specified so that the subsequent configurations are managed to be built on top of the base image. In the back-end application, the OpenJDK environment has been utilized as the base image while Debian, one of the Linux distros has been selected as the base image of the front-end application. After the underlying layer has been setup, further dependencies installation is conducted so that the application can be executed later. In the front-end application, the Flutter SDK is installed from Github and be inserted into environment variable. On the other hand, dependencies installation is skipped in the back-end application since the packaged jar file will contain all necessary dependencies.

After the dependencies installed, the application should be installed into the images. This step is done using the COPY command, whereby the packaged jar file from back-end application and the source codes of front-end application is copied into the images. Since we will have 2 docker images, an additional configuration which is the port exposed has to be performed so that the containers spined from these 2 images will not occupied the same port, leading to the malfunction of the system. The last step of the

Dockerfile is to provide an entry point to the image to indicate the actions to be performed when an instance is created from the Docker Image and be executed.

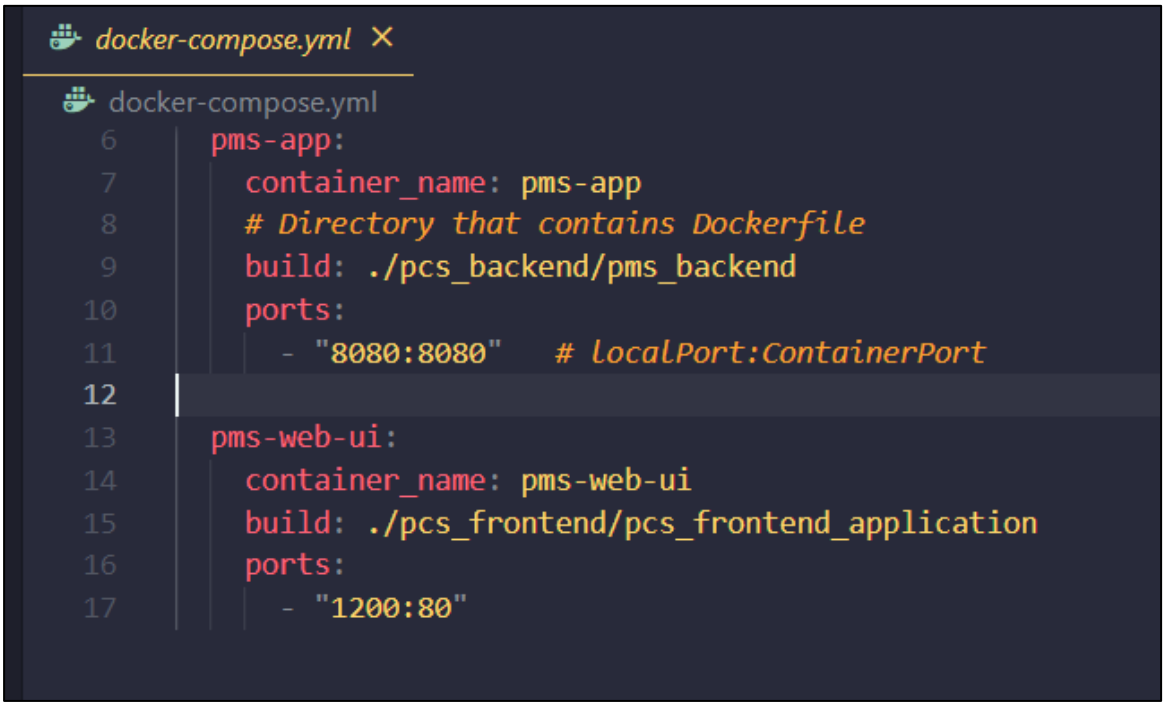


Figure 5.5 Docker-compose configuration

With the ready Dockerfile, the connectivity between front-end and back-end application can be taken into consideration. To ensure the publication of new Docker Container instances are placed into same network, an extra configuration file, Docker-Compose file is required. Using docker-compose configuration, the container name, location of Dockerfile and the environment variables can be identified. Then, a port mapping has to be conducted to map the container port in the Docker Network into local host port so that the application is accessible locally. For instance, the pms-web-ui can be accessed using the URL http://localhost:1200.

With the Dockerfile and docker-compose configuration files set up, the Docker Images is generated, followed by the spinning up of a container that encapsulates instances from both Docker Images. The following figure illustrates the interface of Docker Desktop which presenting the spawned container created from the Docker Images configured.

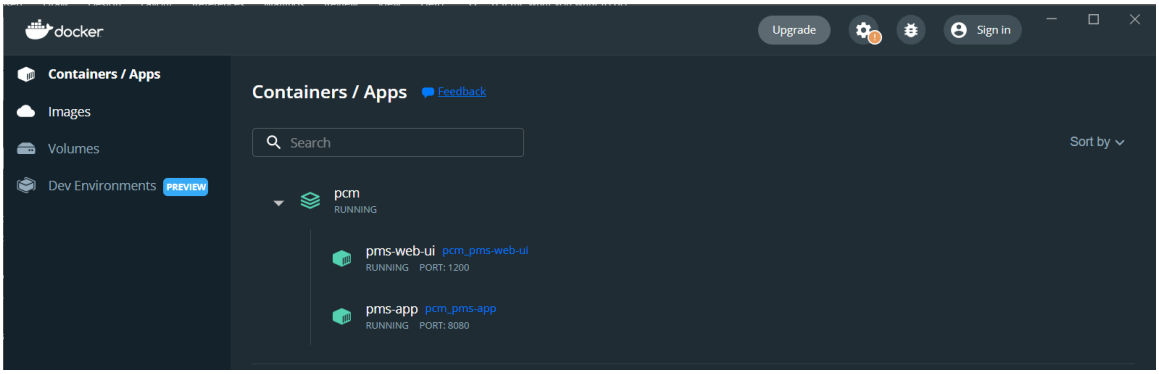


Figure 5.6 Container spawned using docker-compose

```
C:\Users\[redacted]>docker push zhenls/pcm_application:pms-web-ui
The push refers to repository [docker.io/zhenls/pcm_application]
e6a8fbe76d87: Pushed
45d993692050: Pushed
1ea998b95474: Pushed
95b99a5c3767: Pushed
fc03e3cb8568: Pushed
24934e5e6c61: Pushed
e2eb06d8af82: Pushed
pms-web-ui: digest: sha256:2fbe6e068f8eb8372f0bb1295cdfc1198bb640061bed5b20365a3a03ea47150d size: 1780
```

Figure 5.7 Pushing Docker Images to Docker repository

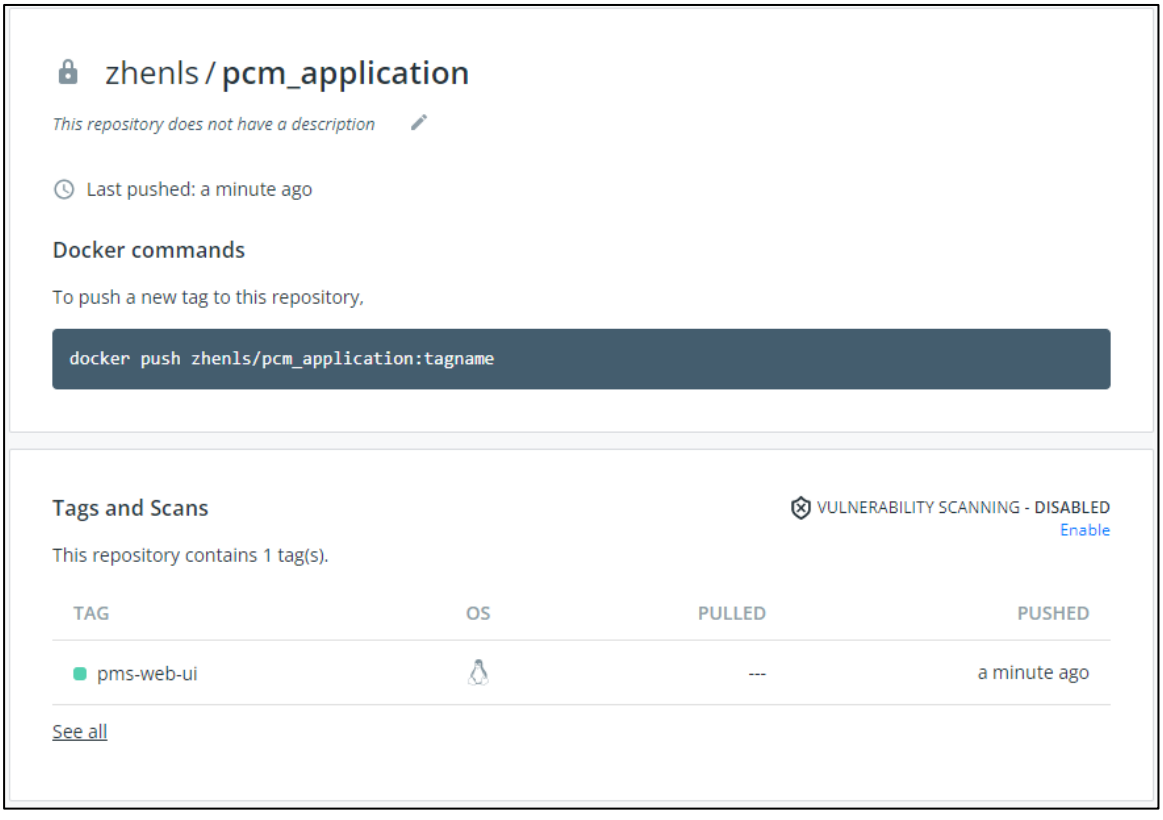


Figure 5.8 Docker repository with pushed Docker Image

After that, the Docker Images are pushed to Docker Repository where they may be downloaded remotely. The Docker Images may now be retrieved from the repository and used to launch new application instances. No further configuration is needed and the deployment of the application is straightforward without the concern on the system environment issues as all essential dependencies are included.

5.4.2 User Authentication

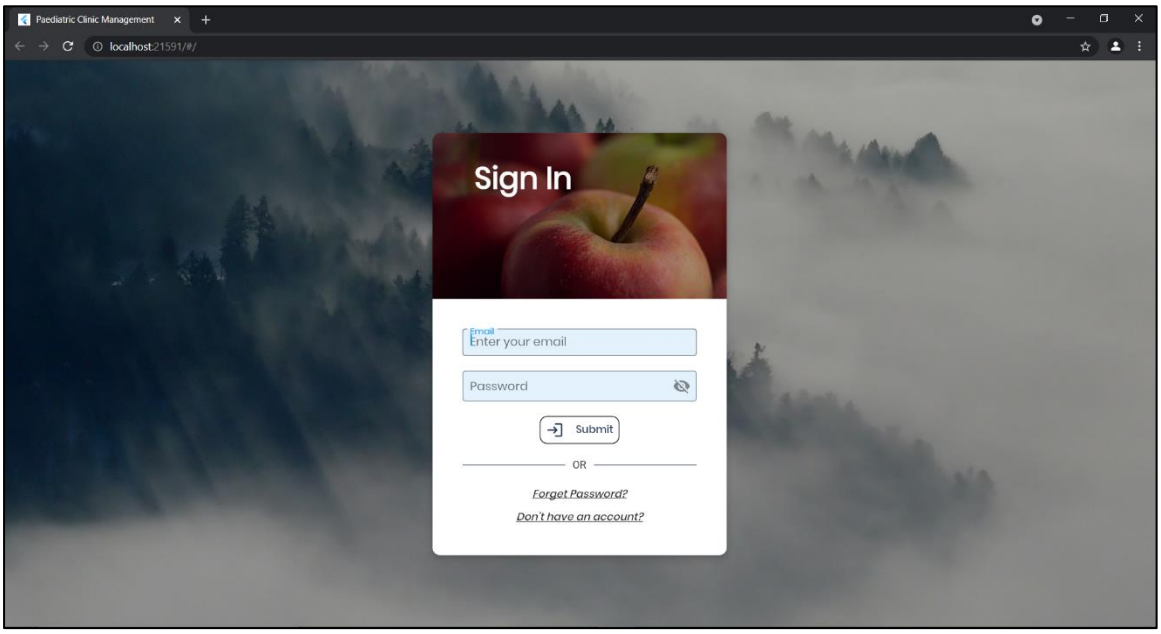


Figure 5.9 UI of sign-in page

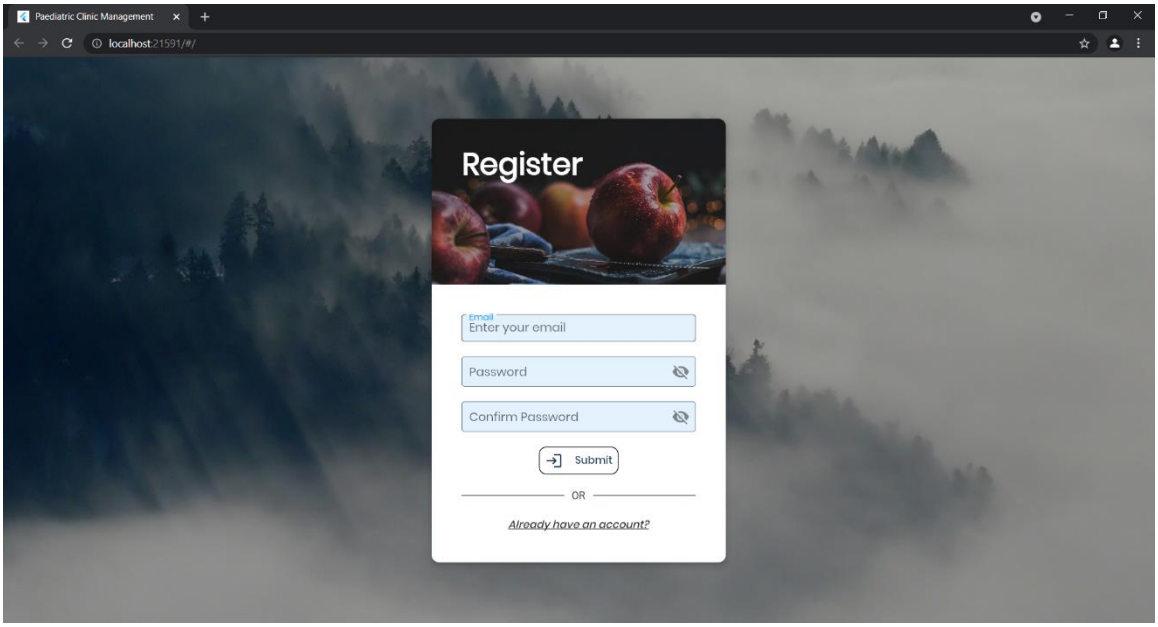


Figure 5.10 UI of user registration page

Figure 5.7 shows the sign page, which are the default page to the users upon their visit to the website. A sign in form is centered in the middle of the page, allowing the users to sign in to the application in case they have registered for an account previously. Otherwise, the users can click on the ‘Don’t have an account’ hyperlink, so that they will be navigated to the register page, which shown in the Figure 5.8. Account registration form requires 3 inputs from the users, ranging from email address, password to password confirmation. The users can also switch back to the Sign-In page through the ‘Already have an account’ hyperlink.

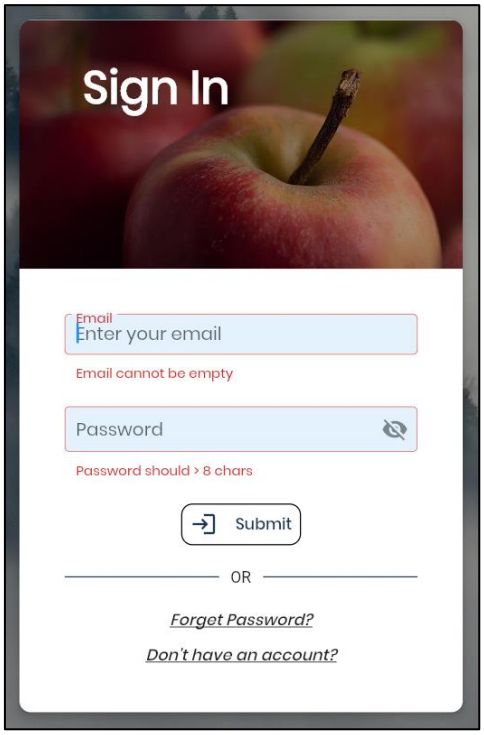


Figure 5.11 Sign-in form validation

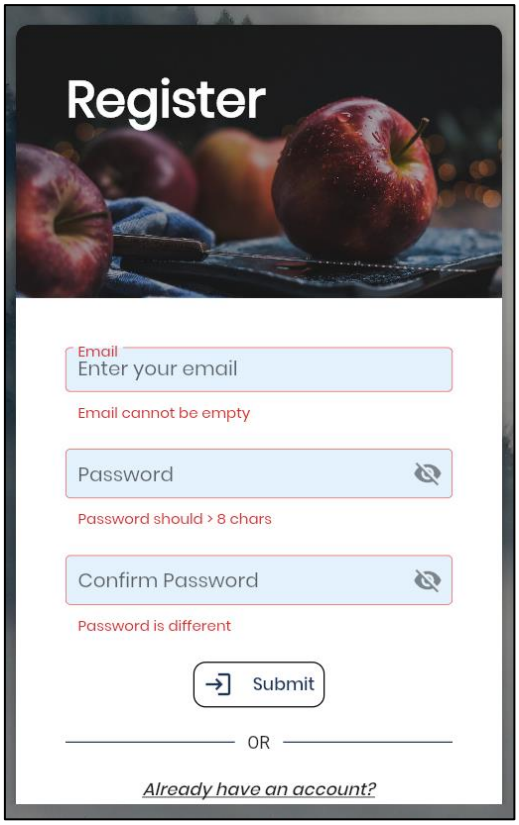


Figure 5.12 Registration form validation

Several validation requirements have been implemented into both of the sign-in and registration form to ensure the correctness of input submitted by the users. Through the validation error messages, the users will be aware of the input field that does not match requirement, thus update it instantly. In case of validation passed, the click on the submit button will trigger the corresponding methods to either sign-in the users into the application or create an account for them. After successfully creating an account, they will also be navigated into the application automatically.

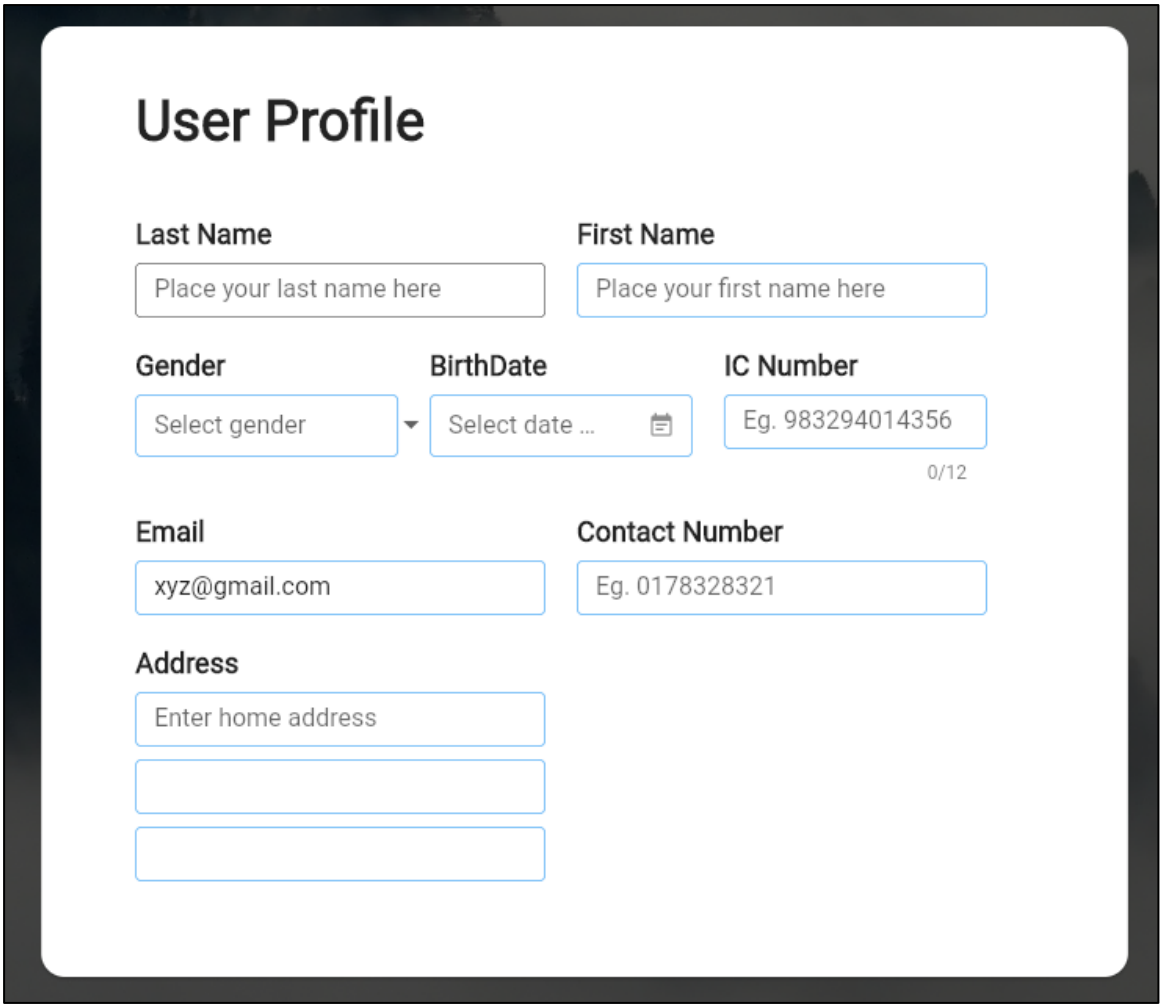


Figure 5.13 User profile form

An additional step is required to be performed by the users in the registration process, whereby they are required to provide their basic personal information in a popup dialog box which will be presented when the users submit their credentials. This information becomes essential in facilitating the clinic to have an improved knowledge on their clients. The user profile form has been equipped with various validators according to the fields. For instance, the IC number only accepts numeric characters and the value length must be 12 characters. These validators reduced the risks for having user data in inappropriate format, forming organized user records with high maintainability. When the users successfully registered their accounts, the data provided in the user profile form are recorded into Cloud Firestore as the user records.

5.4.3 Dashboard

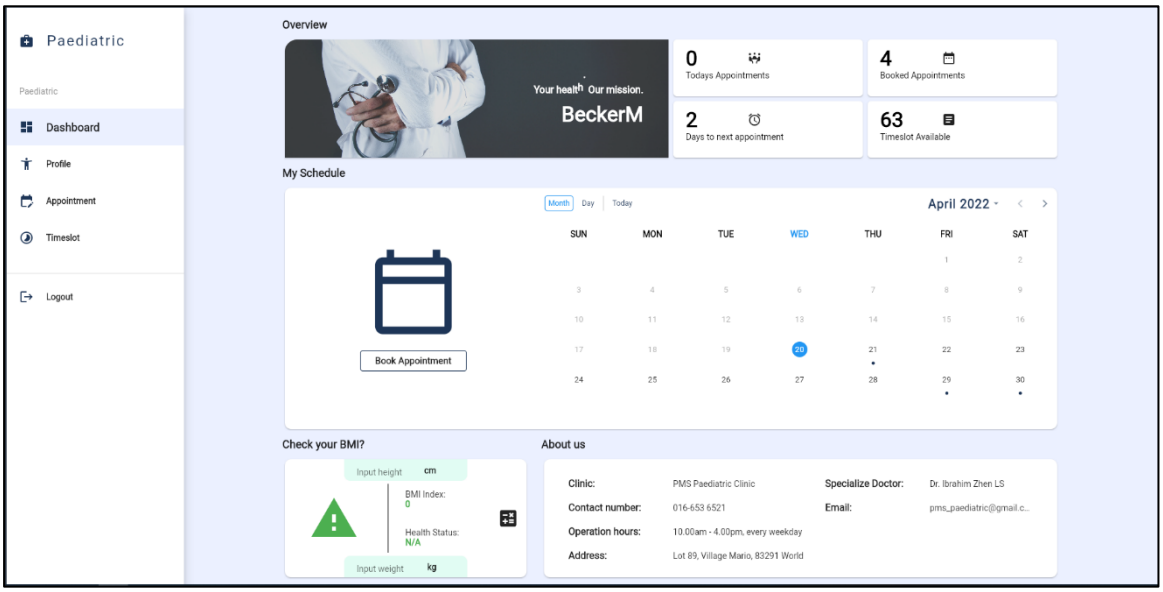


Figure 5.14 General layout of dashboard

After the login or succeed registration procedure, the system will navigate the users to the internal landing page, which presents a dashboard for the summarized information. To enforce consistency and simplicity, all of the pages will have a similar structure, whereby a side menu that provides navigation between each module is attached to the left of the screen, while the major section of the module is placed in the remaining space. In dashboard, there is a designated section to sum up the information of the appointments and timeslots, assisting the system users to have a brief knowledge regarding their status with single glance.

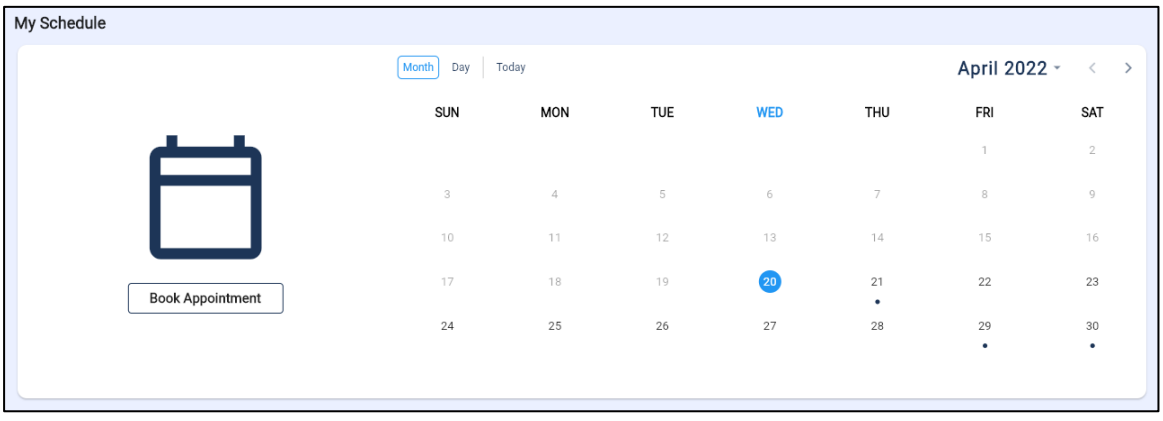


Figure 5.15 Schedule calendar in month view

The middle section of the dashboard illustrates a schedule section along with a calendar. Each of the incoming appointments of the users is indicated as a single dot on the calendar, indirectly reminding the users for the appointment sessions.

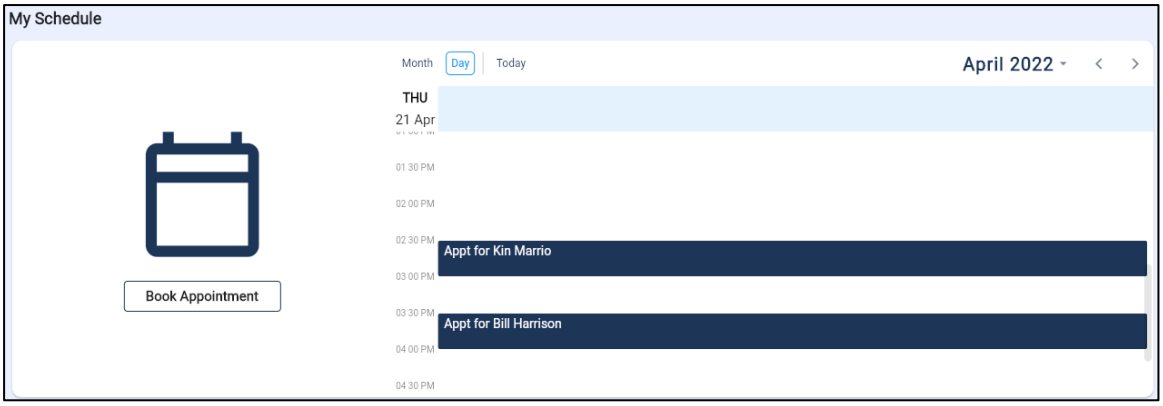


Figure 5.16 Schedule calendar in day view

Through clicking on the date, the users can further view the exact time of the appointment session, helping them to arrange their schedule. To the left of the calendar widget, there is also a button which reserved as a quick access for appointment booking. When the user interacts with the book appointment button, a dialog box will popup, allowing the user to perform appointment booking.

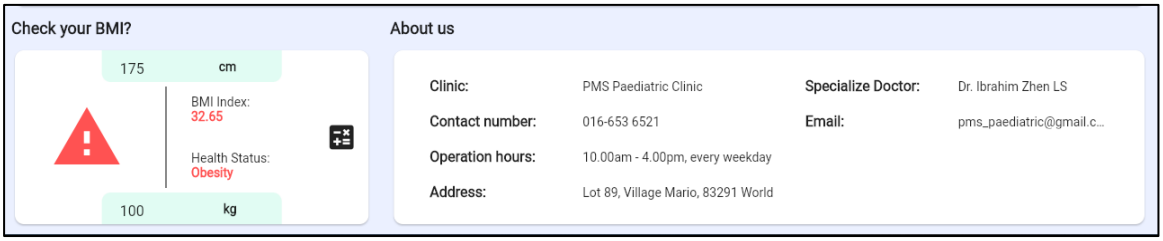


Figure 5.17 Footer of the dashboard

The bottom section of the dashboard page is composed of 2 widgets, which are the BMI Calculation widget and the about us information section. Throughout the BMI calculation widget, the users are allowed to compute their BMI index by providing their weight and height data. Instant response with the BMI index and the health status derived from the index is provided to the users so that they can have a better understanding on their current health status.

On the other hand, the about us section has stated several information of the clinic including contact methods, helping the users to locate the clinic physically or contact the doctor for consultations.

5.4.4 User Module

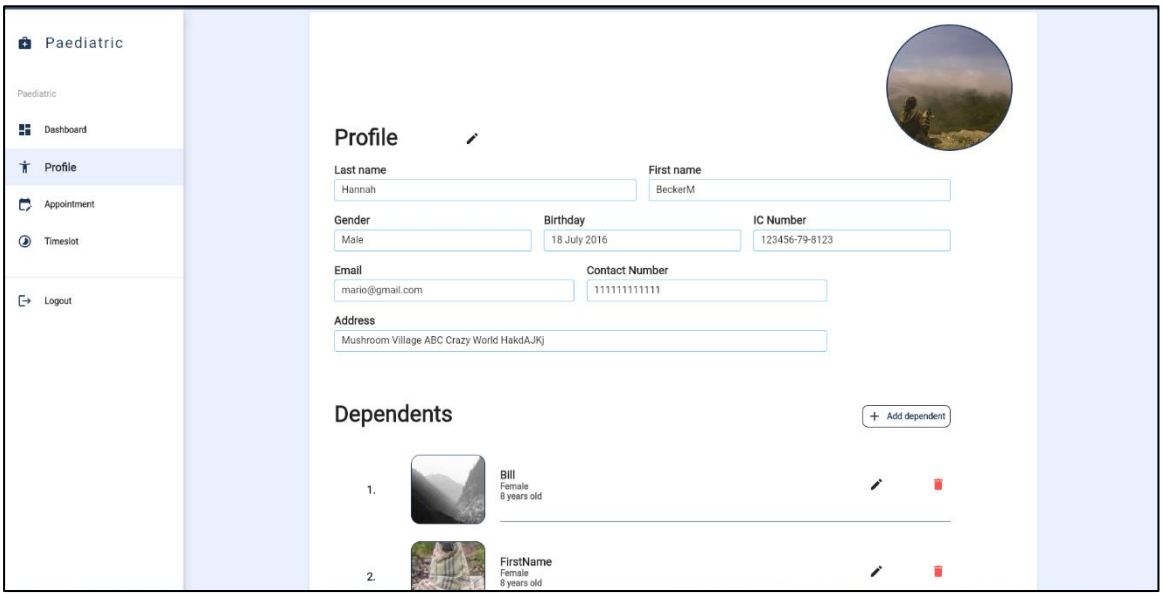


Figure 5.18 Layout of user profile page

The users are managed to view their profile data using the navigation button in the side menu. Once the profile tab is selected, the main body of the application will be switched to display the user supplied using the information provided during user registration. The profile page with a scrollable view is divided into 2 parts, with the upper portion illustrating the user's information and the bottom part listing out the data of the users' dependents.

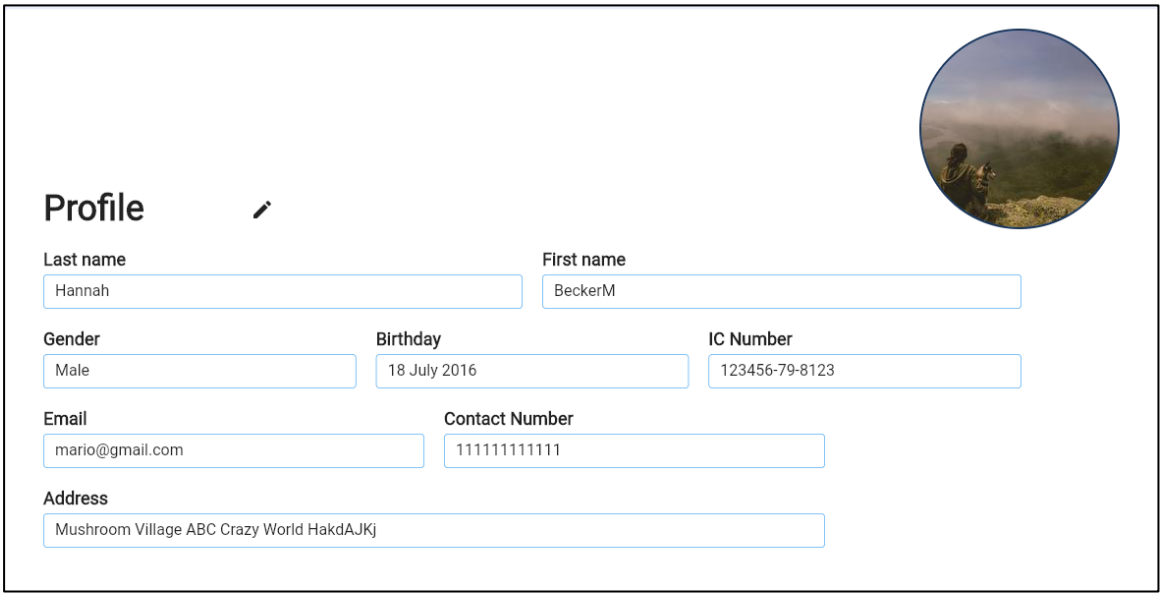


Figure 5.19 User profile section

In the upper part of the page, the basic information such as name, contact number and email address can be viewed. These fields are temporarily read-only so that the users will not update their data unintentionally. In order to update any of these fields, a single tap on the edit icon located to the profile header will trigger a popup dialog box with the update profile form.

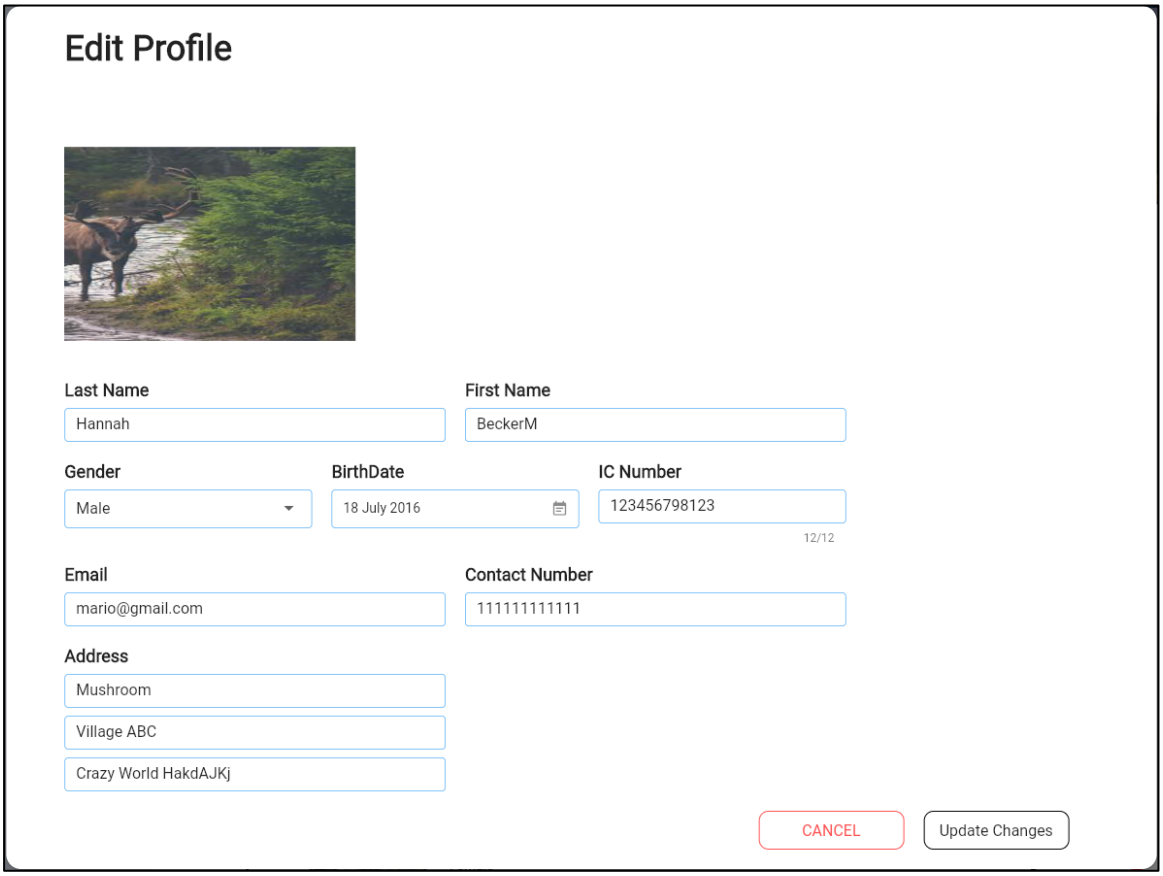


Figure 5.20 Update profile form

Initially, the update profile form carries the current user data by default. The values in these fields are modifiable to the latest data. Similar to the user registration form, several validators are implanted into the update profile form to avoid unintended data format.

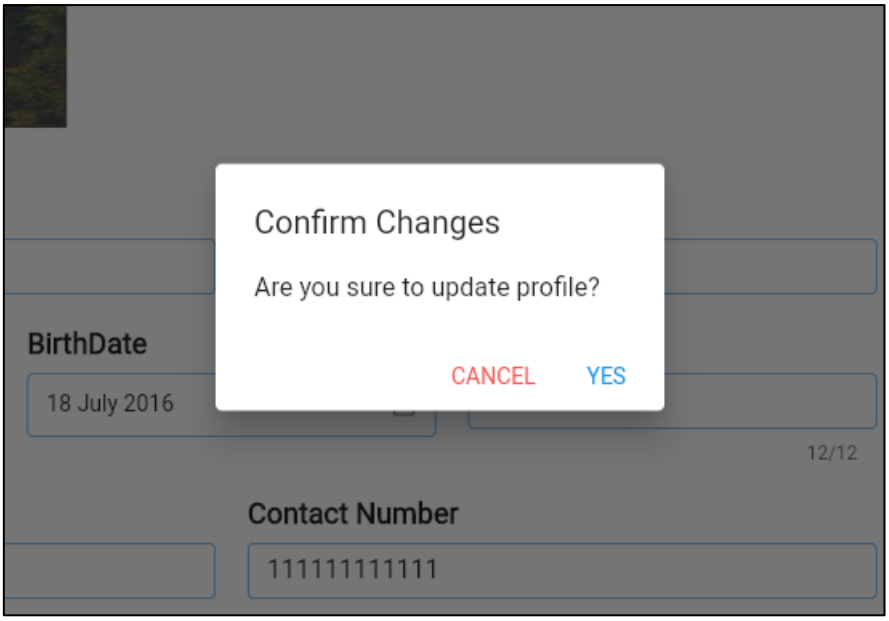


Figure 5.21 Dialog box for update confirmation

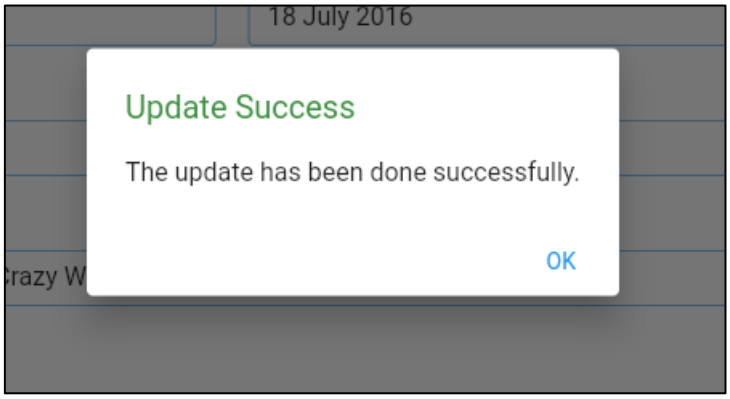


Figure 5.22 Dialog box to inform update result

After the user done inserting the latest value, the 'Update Changes' button can be clicked to trigger the update action. The values contained in the update profile form will be encapsulated into a domain model, parsed into JSON format, inserted as API request body and transferred to the back-end application for processing. After the processing is done, latest data will be returned to the front-end application. Eventually, the UI will reflect the result of the update action by dismissing the update profile form dialog and showing another dialog box with action result.

5.4.5 Dependent Module

Figure 5.23 Dependent table with no dependent record

The bottom part of the profile page consists the data of the users' dependents. Since the clinic is specialized to provide paediatric services, most of the patients will be in young age, thus requiring to be managed under their parents. By default, a fresh created user account does not own any dependent record, leading to the 'No dependent record found' message be shown in the dependent sections, as illustrated in Figure 5.21.

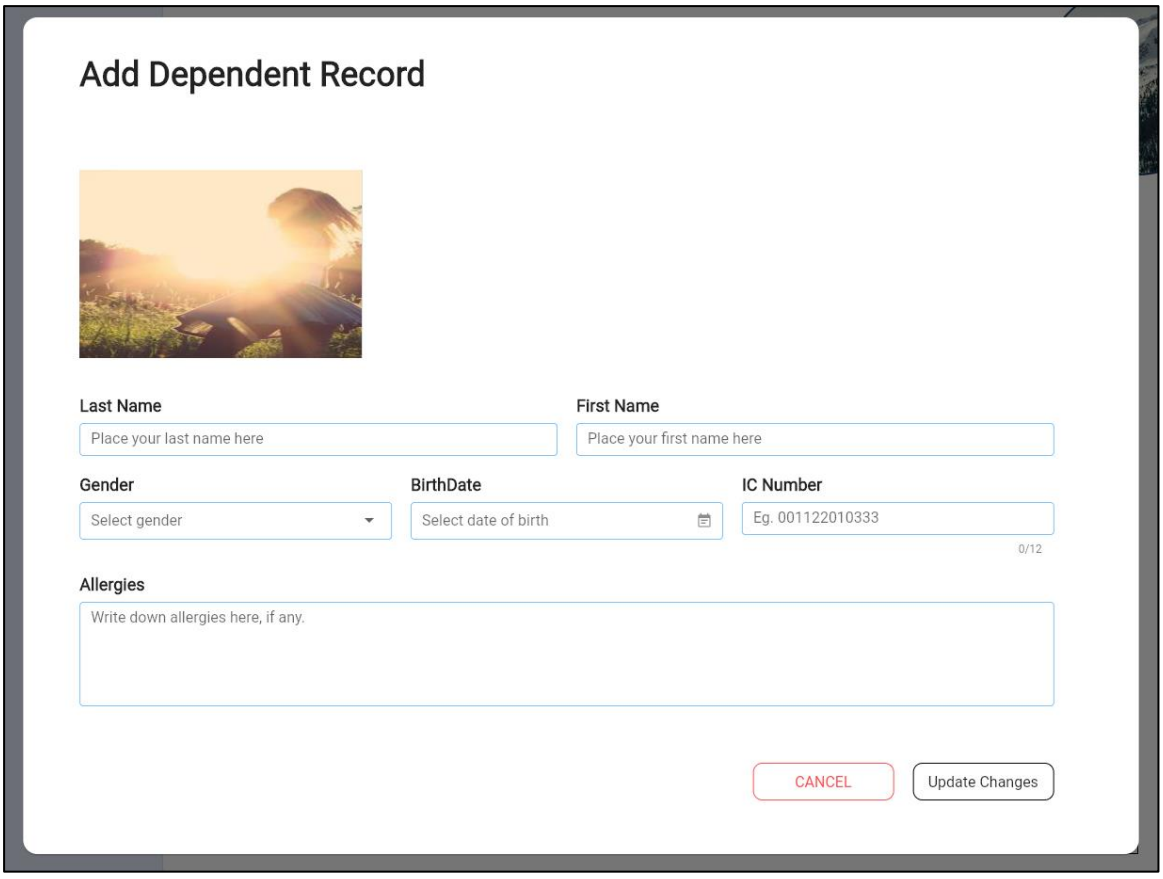


Figure 5.24 Form for create dependent record

The user can click on the ‘Add dependent’ button to trigger the presentation of a popup dialog box equipped with add dependent form. In this form, the user can specify their dependent details, including the last and first name, gender, date of birth, IC number and also their allergies if any. The dependent information will be stored in the Cloud Firestore collection so that they are accessible for other operations later. After the confirmation of dependent information submission, the latest dependent records will be displayed as a list in the dependent section.

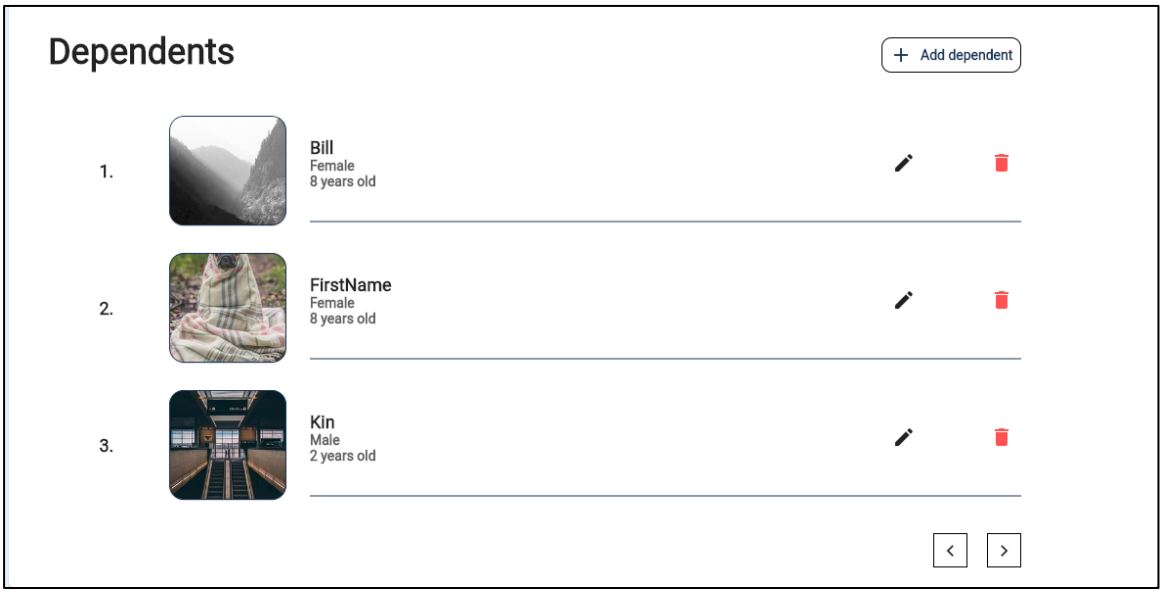


Figure 5.25 Table of dependent records

The dependents record listed will be sorted ascendingly with respect to the dependent name. In case the number of dependents exceed 3, a pagination will be applied on the dependents record as the users can navigate between each dependent page with the provided pagination button. Besides that, there are 2 buttons, which are the edit and remove button be integrated with each of the dependent record.

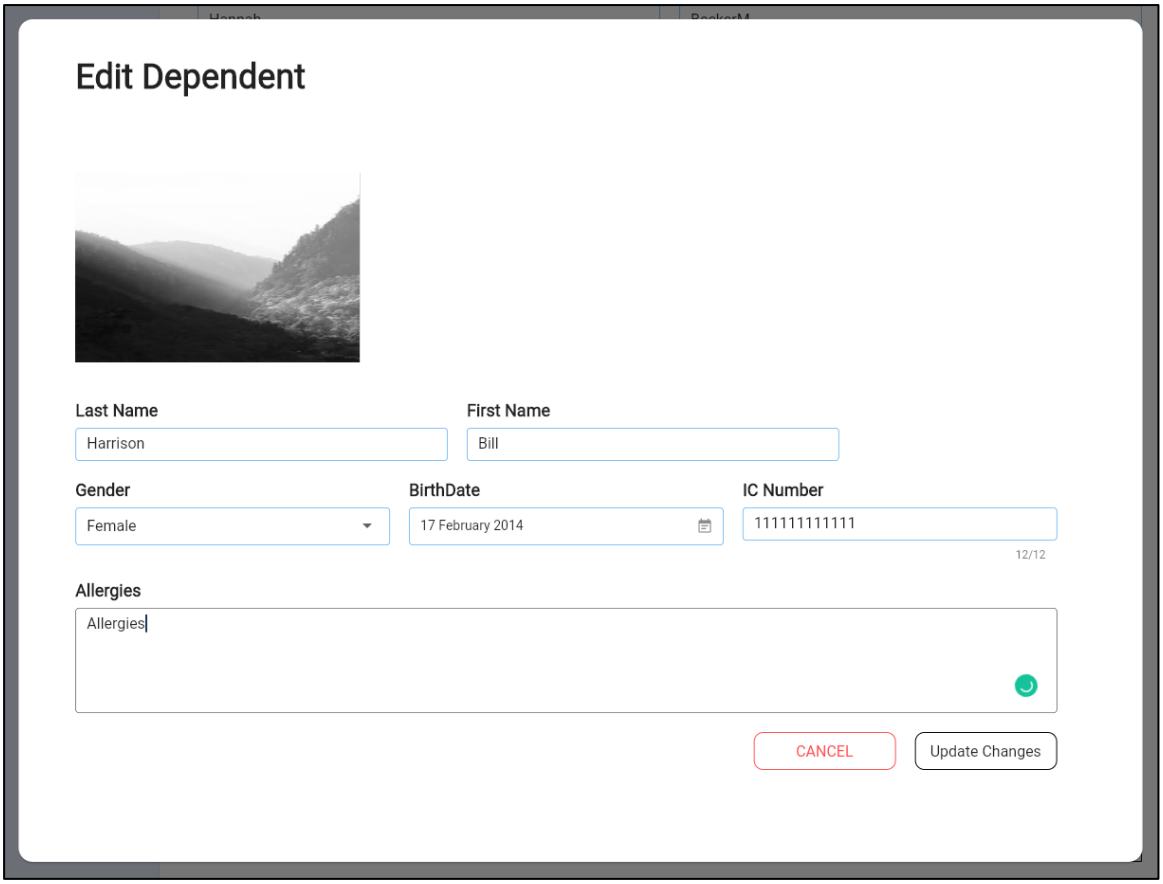


Figure 5.26 Form to update dependent

Through the edit button, the information of the dependent can be updated by providing latest values in the edit dependent form displayed. After submission, the selected dependent record will be updated in the database and then be reflected in the dependent record.

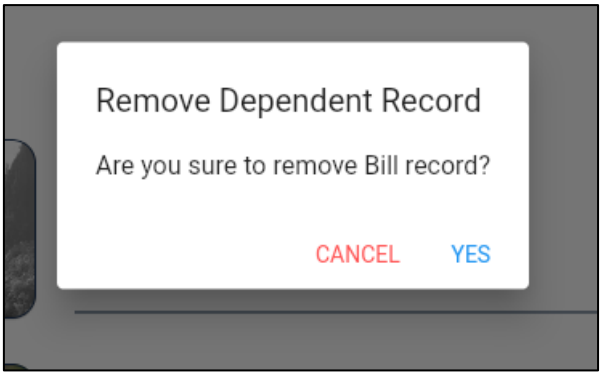


Figure 5.27 Alert dialog to confirm removal action

The dependent record can be completely removed with the remove button provided. In case of delete button is tapped, the system will prompt an alert message box to have a confirmation on the delete action, so that the dependent record will not be accidently removed due to misclicks. If the record is confirmed to be removed, a delete dependent API exposed by the back-end application will be invoked, dumping off the selected dependent record from the database. The user will also be informed regarding whether the removal is successful using a small dialog box.

5.4.6 Timeslot Module

Figure 5.28 Timeslot table without records

The timeslot page is accessible through the navigation tab available in the side menu. The timeslot page displayed all available timeslot to the users. In case the user has admin credential, a ‘Create timeslot’ button is visible in the header of the timeslot page. Single clicking on this button will call out the create timeslot form, prompting the admin for the timeslot details.

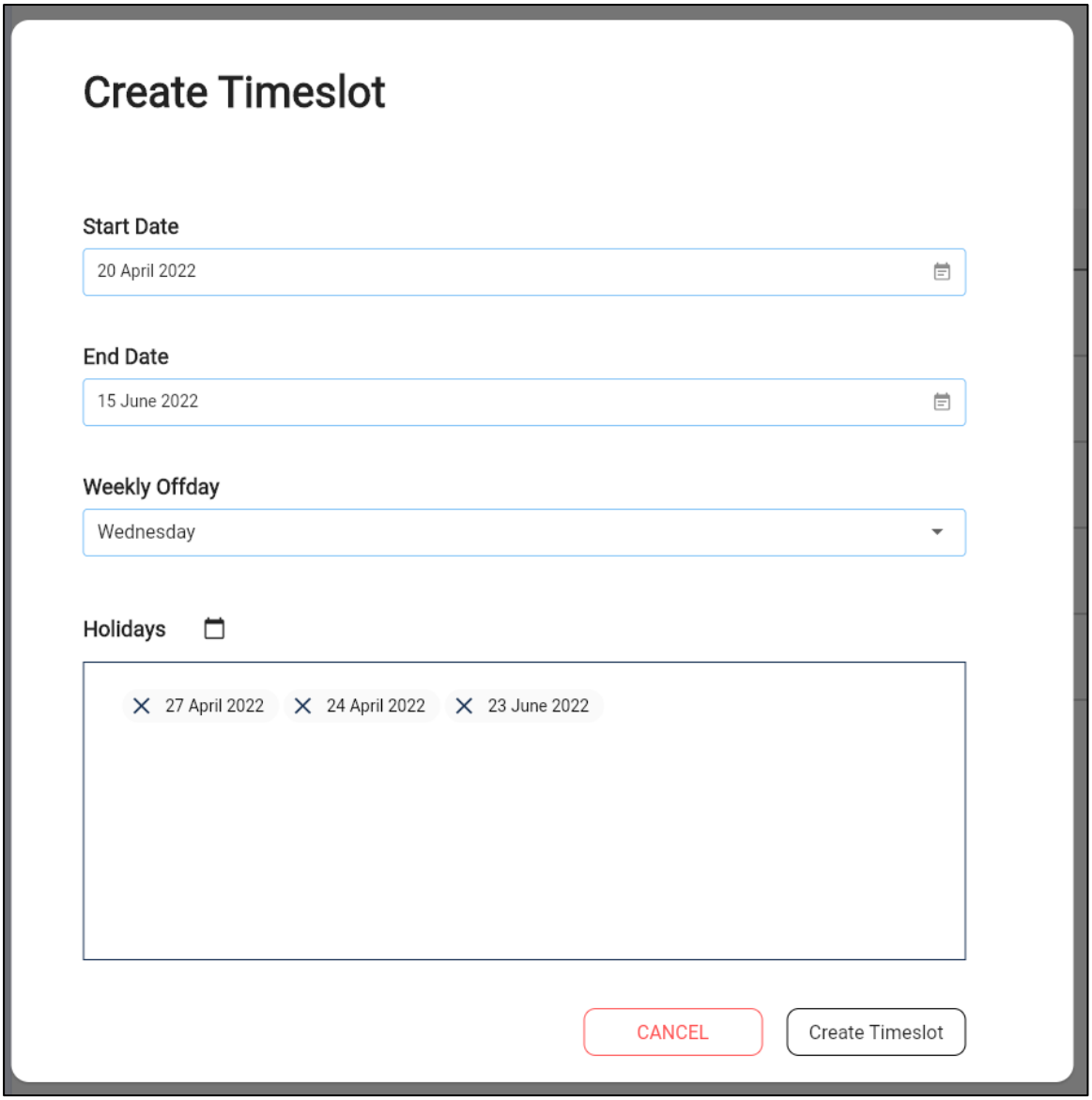


Figure 5.29 Form to create timeslot

To reduce the efforts required during timeslot setup, the admin can specify a date range, and then selects the weekly non-operation day and also the date of holidays. With these details, the system will filter out every occurrence of the non-operation day in the date range specified, as well as the holidays. After the processing, a total of 9 timeslot will be created for each of the leftover date, improving the efficiency in configuring the available timeslot since the admin does not keep repeating the create timeslot action for each operation day.

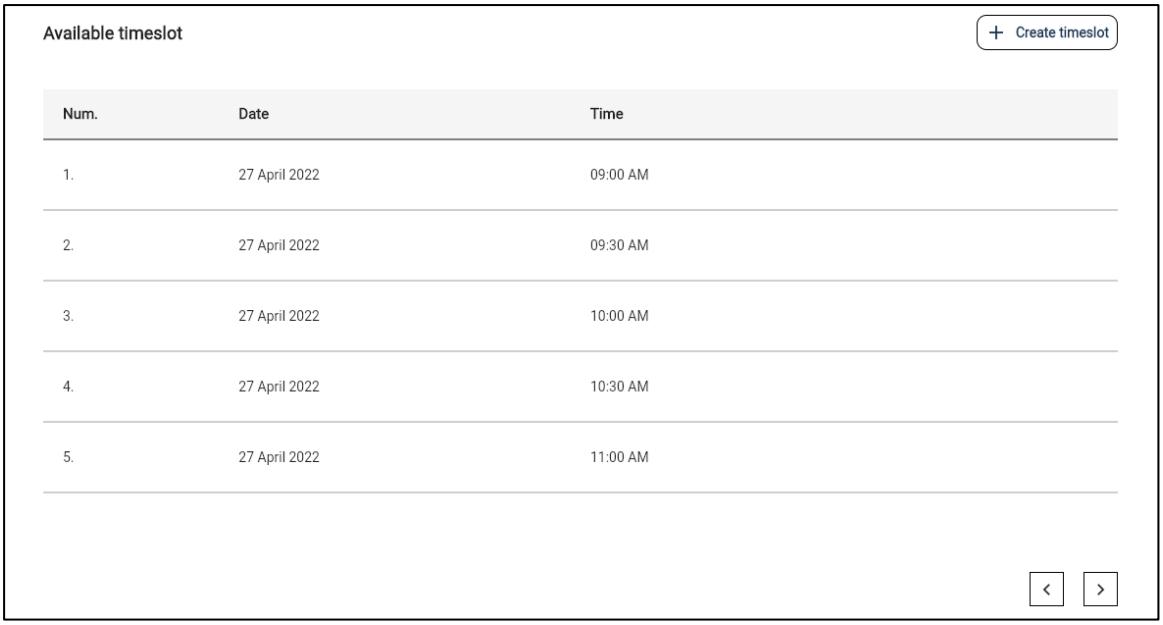


Figure 5.30 Timeslot table with created timeslot records

The created timeslot will be presented in a table in the timeslot section, allowing the users to look for the available timeslots in case they intend to book for an appointment slot.

5.4.7 Appointment Module

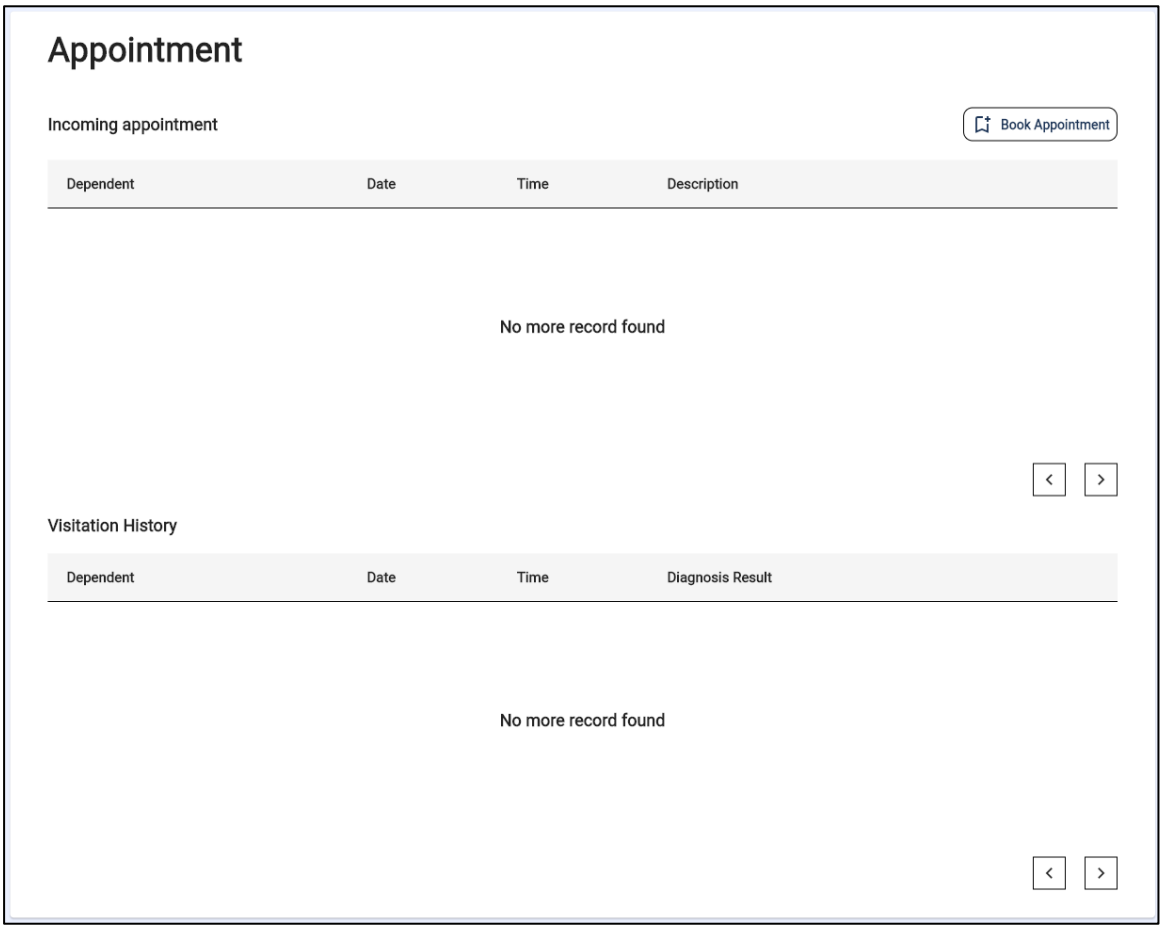


Figure 5.31 General layout of appointment page

The appointment page can also be accessed from the side menu. The appointment page is separated into upper and bottom layout, whereby the upper layout illustrates the incoming appointment section followed by a visitation history table in the bottom layout. The upper layout of the page listed every appointment booked by the user. Each of the table item will list out the dependent name, the date and time of the appointment and the note they made during the appointment booking.

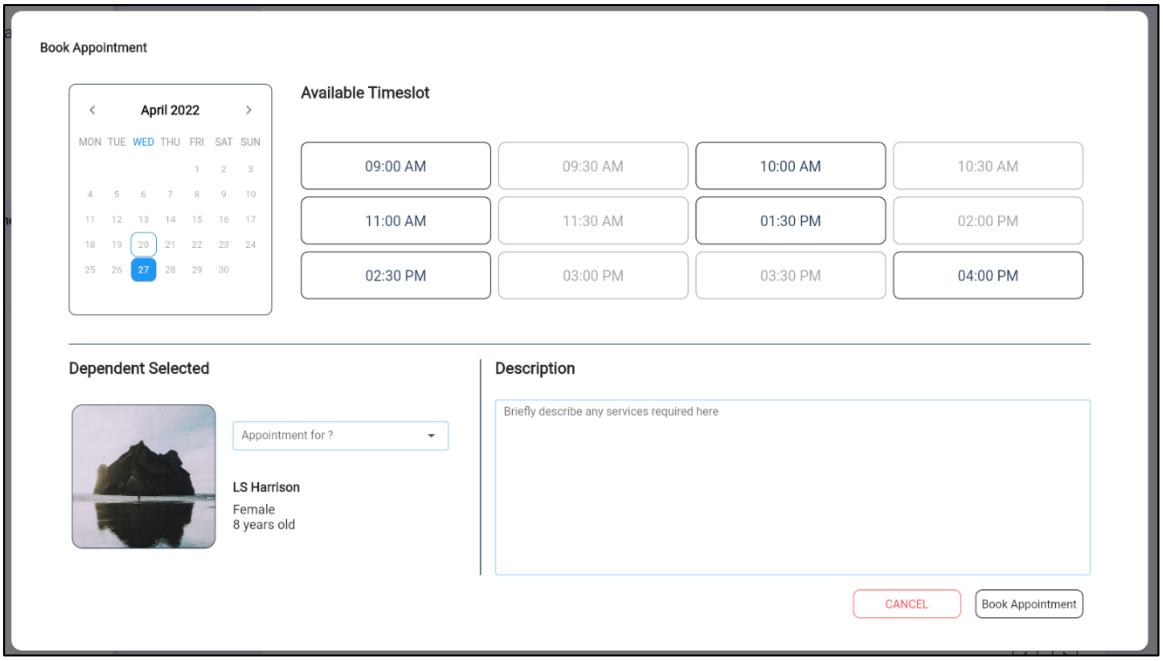


Figure 5.32 Form for appointment booking

In order to book for an appointment session, the user can click on the book appointment button. With this, a popup booking form will be presented to the user, waiting for the details of the appointment section. The entire booking form is integrated with 4 different widgets. First of all, a calendar is illustrated in the upper left corner of the form. In this calendar, all of the dates will be blacked out and unable to be selected, except for those dates with available timeslot. With this, the user is capable to view the available dates of the month. They are also managed to navigate to other months in case the dates in current month are not suitable.

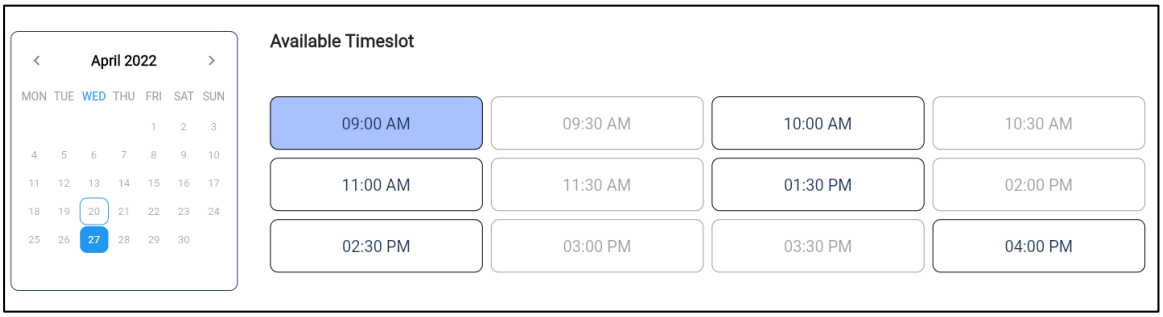


Figure 5.33 Appointment booking form with selected date and timeslot

After the selection of appointment date, the widget right next to it will display all of the timeslots of the day. Among all these timeslots, those timeslots that have been booked by other users will be grayed out, indicating they are not available at the current time. The user is free to choose any of the leftover timeslot, and their selected timeslot will

be changed to purple-blue in color, allowing the user to be informed regarding their selection.

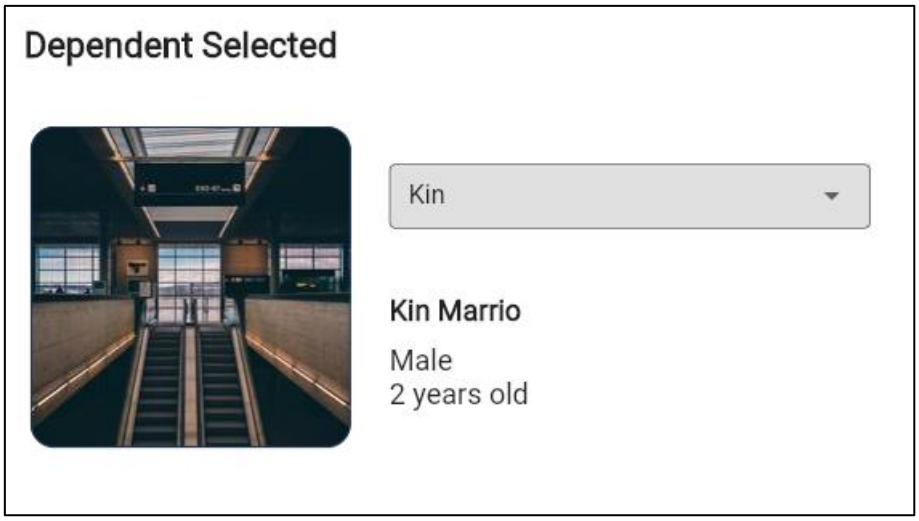


Figure 5.34 Widget for dependent selection in appointment booking form

After the selection of appointment date and time, the user is required to select the major target of the appointment. In the dependent selection widget, all of the dependent records created previously in the dependent module will be made available for selection in a dropdown menu button. Whenever a dependent is selected, the dependent name, gender, age and avatar will be displayed to the user.

Figure 5.35 Remark section for appointment booking form

The user is also allowed to provide some remarks regarding the appointment in the description fields. For instance, the user may request for the services such as medical body checkup or vaccination so that the required equipment can be prepared prior to the appointment session.

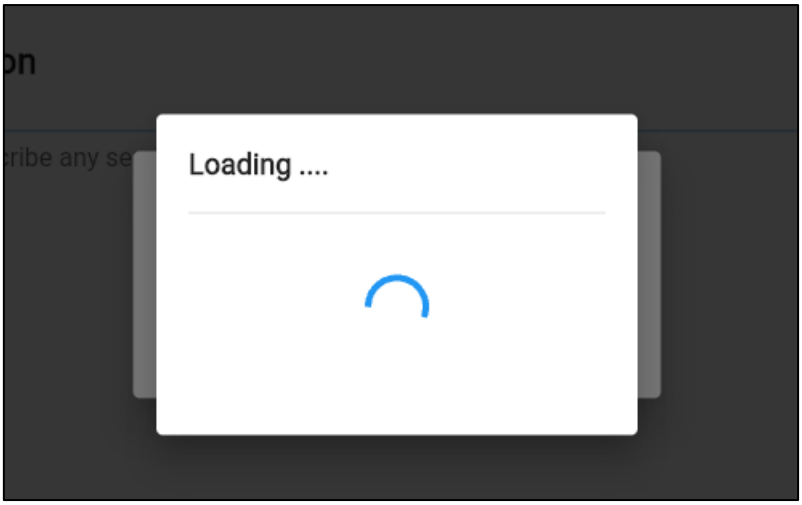


Figure 5.36 Loading widget

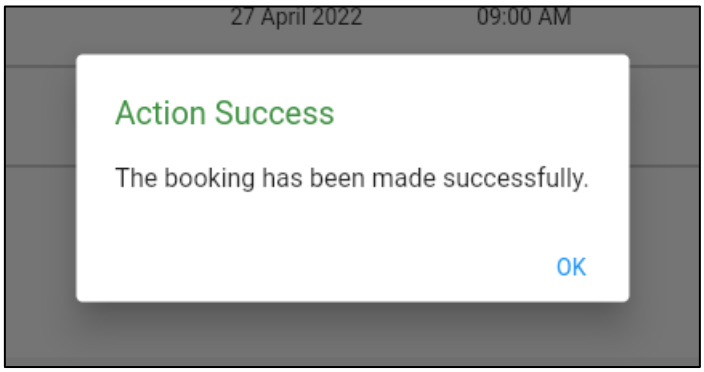


Figure 5.37 Dialog box that shows action success message

After the booking form is submitted, the booking request will be processed by the back-end application. A loading screen will be displayed at the mean time of processing. In case the booking success, a dialog box with action success message will be displayed. Otherwise, an alert dialog box will be shown and therefore reminds the user to try another booking.

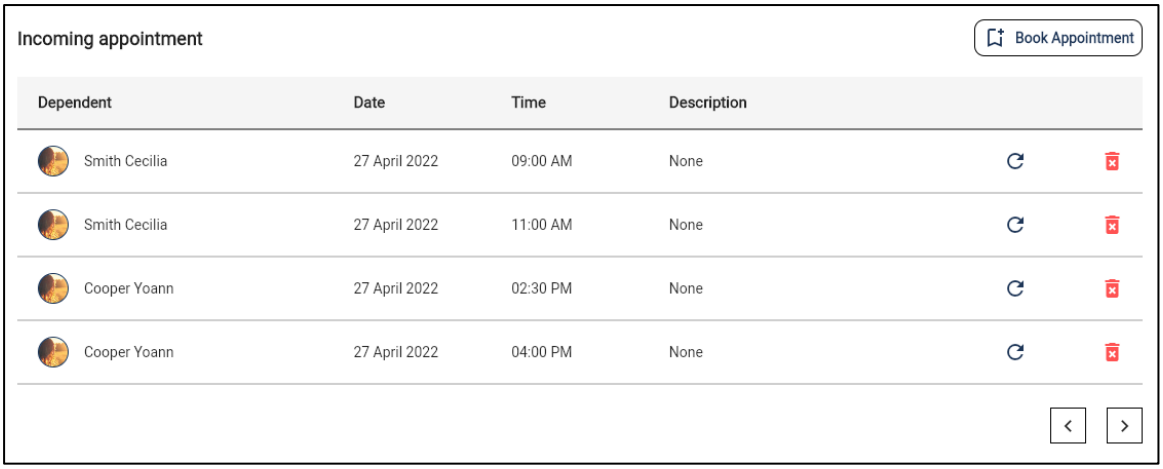


Figure 5.38 Incoming appointment section

Whenever the user has successfully booked for any appointment session, the appointment info will be illustrated in the appointment table form. Each of the appointment record is attached with the reschedule and cancellation button, allowing the user to modify their booked appointment on their own.

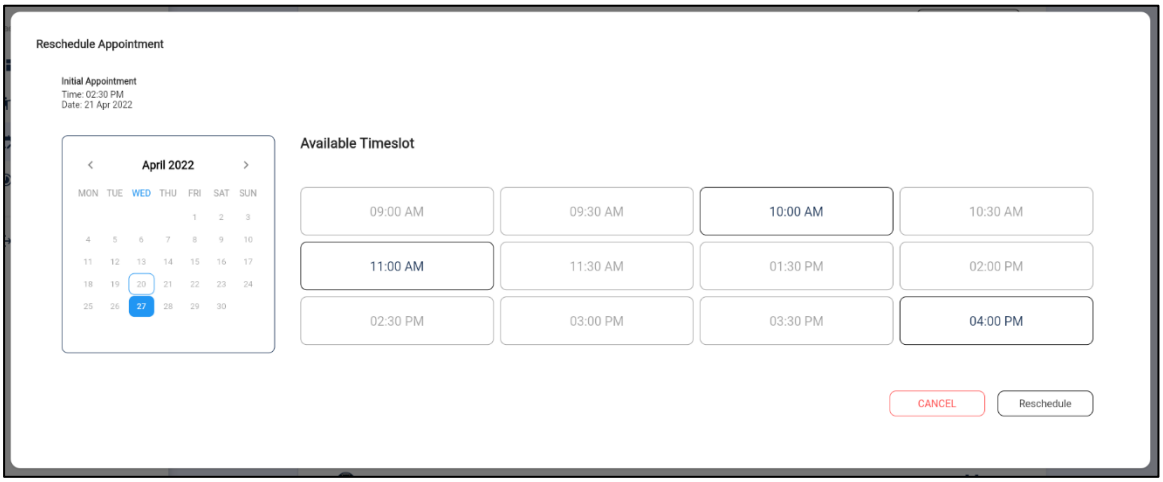


Figure 5.39 Form for appointment reschedule

In case the user intends to reschedule the appointment made before, a single click on the reschedule button will invoke the reschedule appointment form. A summary of the initial appointment info is constructed at the top of the form, followed by the similar calendar and available timeslot widgets are presented to the user, enabling the selection of new date and time. With the rescheduling, the initial timeslot booked will be released and available for booking.

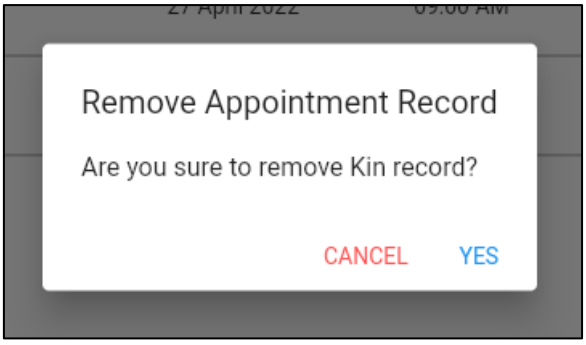


Figure 5.40 Alert dialog box to confirm appointment cancellation

The cancellation of booked appointment is also possible with the remove button on the appointment record. In case the user intends to give up the appointment, the system will prompt a confirmation on the removal action. With the user confirmation, the appointment selected will be cancelled and removed from the database. The occupied timeslot will also be released from the occupied status and thus be available again for selection for appointment booking.

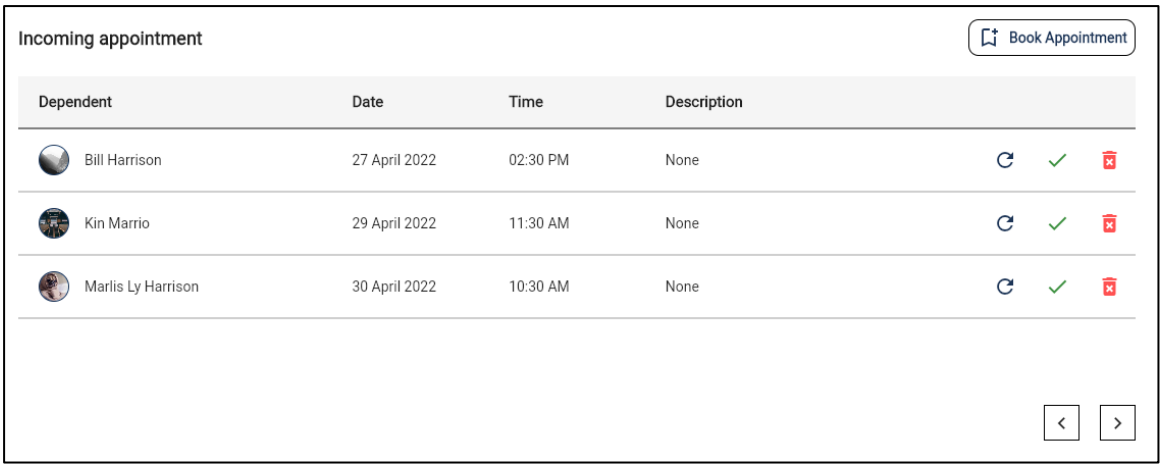


Figure 5.41 Second page of incoming appointment records

A pagination has been done to the appointment records listed in the table. While the appointment record is sorted ascendingly to show the nearest appointment session, the user is managed to view or modify other incoming appointment through the navigation button provided at the bottom of the table.

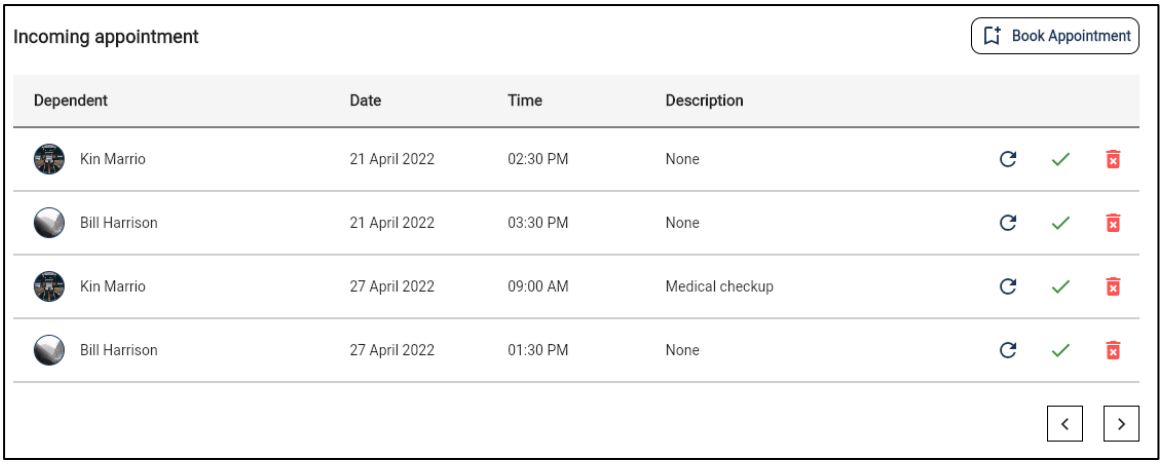


Figure 5.42 Admin view of incoming appointments section

From the admin perspective, the appointment table will show all booked appointments from various users, sorted with the nearest date. With this, the admin can have a brief knowledge on the schedule of the day. Besides that, there is also an additional button which is used to complete the appointment.

Figure 5.43 Form for appointment completion

Whenever an appointment session is attended, the appointment can be made completed through the complete appointment action. The complete appointment action requires the admin to specify the diagnosis result or doctor comments after the treatments. The prescription given to the patients can also be specified in the complete appointment

form. With this, the user will be managed to review their diagnosis result and prescription in the visitation history section, helping them to recall if necessary.

```java
@Component
public class DailyActivitiesScheduler {

    private final AppointmentDao appointmentDao;

    public DailyActivitiesScheduler(AppointmentDao appointmentDao) {
        this.appointmentDao = appointmentDao;
    }

    // Execute at the start of every day
    @Scheduled(cron = "0 0 0 * * *")
    public void updateExpiredAppointment() throws InterruptedException, ExecutionException {
        LocalDate yesterday = LocalDate.now().minusDays(1);

        List<Appointment> appointments = appointmentDao.getAppointmentsByDate(yesterday);
        for (Appointment appointment : appointments) {
            appointmentDao.completeAppointmentById("SYSTEM", appointment.getApptId());
        }
    }       You, 6 days ago • feat: added Daily Complete Appt Scheduler …

}
```

Figure 5.44 Scheduler to update expired appointments

In order to update the availability of the expired appointment automatically, a scheduler has been configured. At the start of everyday, all of the appointment of the previous day will be updated to expired, ensuring the appointment section will not display outdated data.

5.4.8 Visitation Module

Figure 5.45 Visitation section with no record

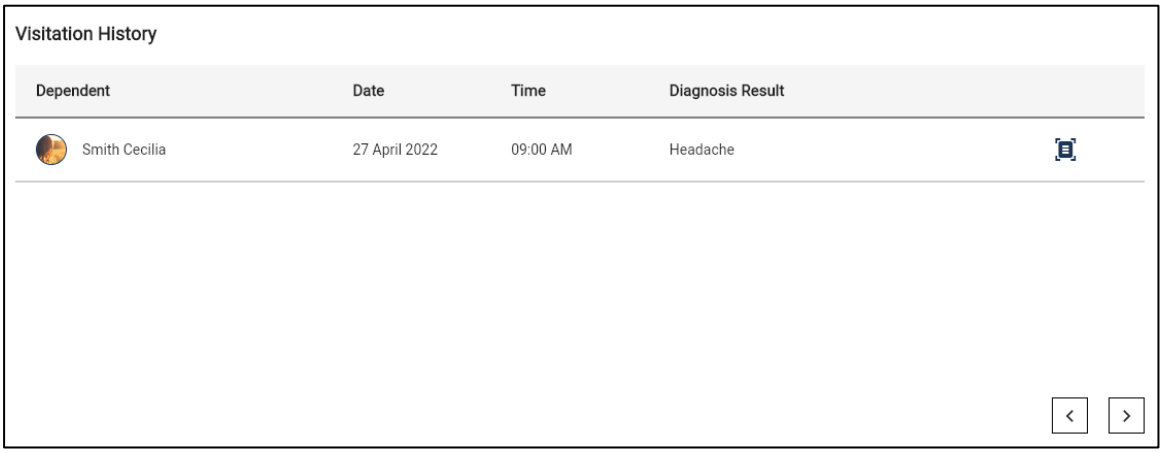


Figure 5.46 Visitation section with record

The visitation history section listed out the paginated visitation attended to the clinic. In case there is no visitation yet, a message that inform no visitation record will be displayed in the visitation table. From the admin perspective, all of the visitation records are displayed, allowing the admin to perform tracing on all of the visitation records.

Figure 5.47 Form to display diagnosis result and prescription

The diagnosis result and prescription of that visitation can be reviewed through the prescription button shown to the right of the visitation record.

5.4.9 Responsive Design

To enable accessibility across multiple devices and platforms, the web application UI has been made responsive so that the UI varies to adapt with the resolution of the user devices. Generally, the side menu will be collapsed into a menu button in the devices with small screen size, leaving the spaces for the major body of the page. Various forms for the CRUD operations are also repopulated with appropriate layout to ease the user action in small screen. The following figures illustrate several responsive designs of the widgets.

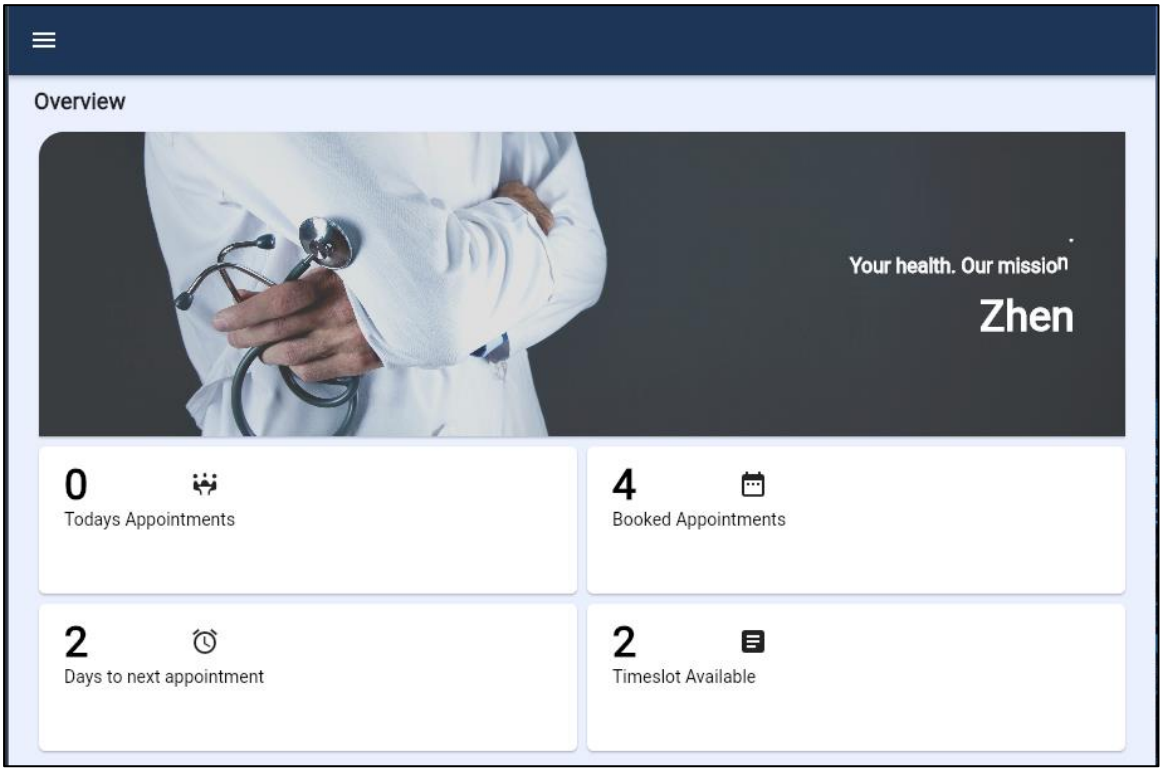


Figure 5.48 Responsive design of dashboard (part 1)

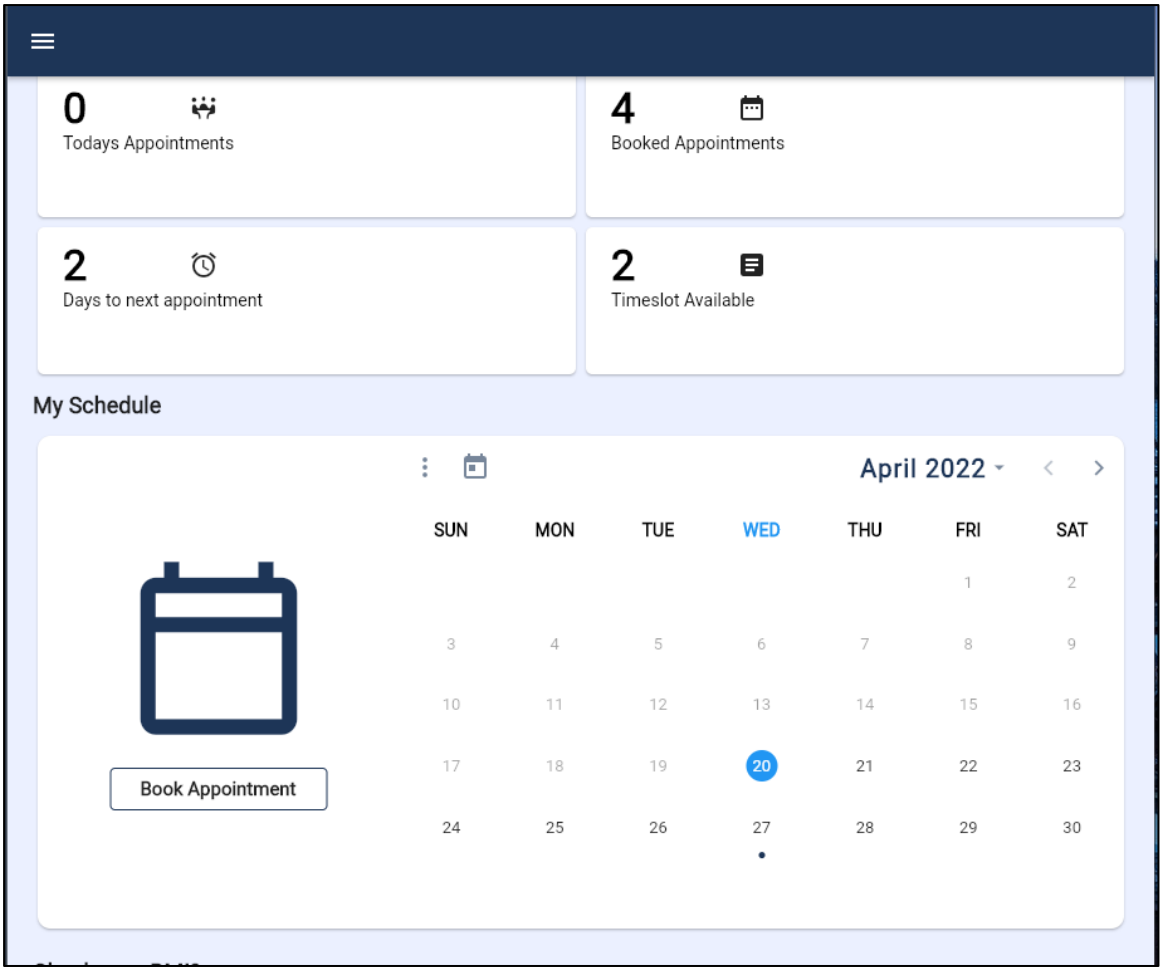


Figure 5.49 Responsive design of dashboard (part 2)

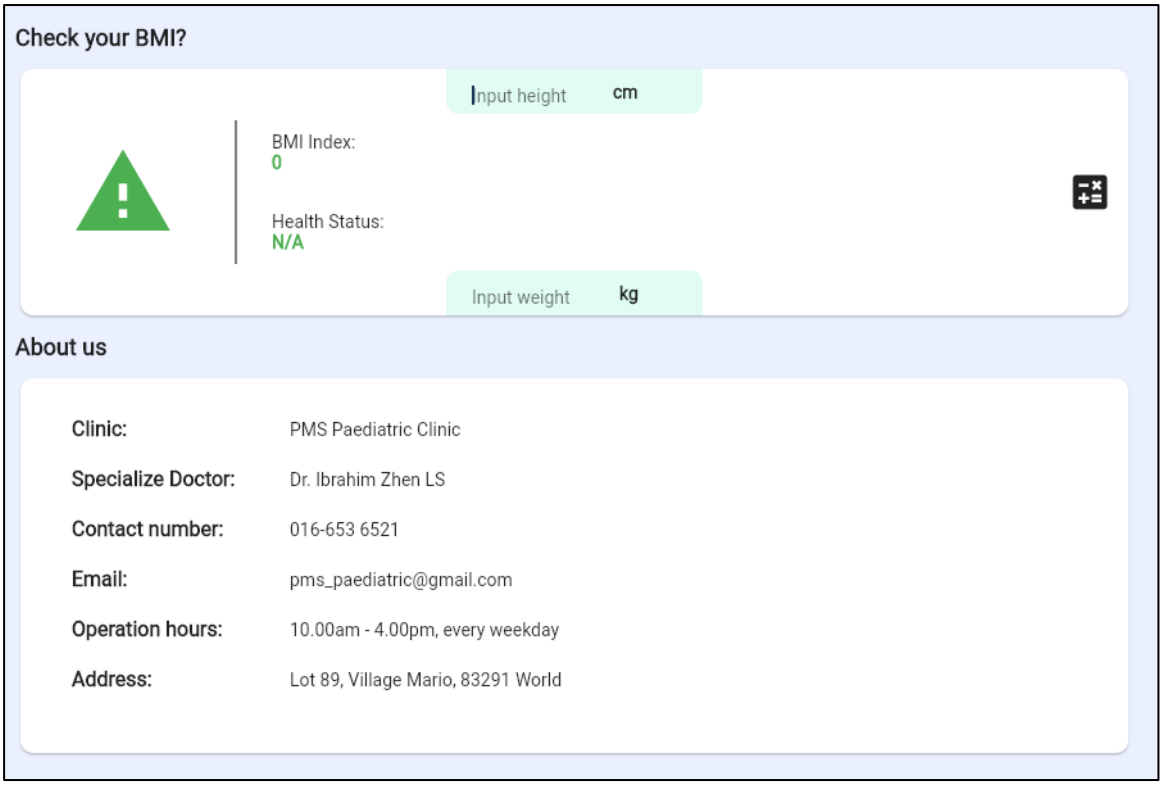


Figure 5.50 Responsive design of dashboard (part 3)

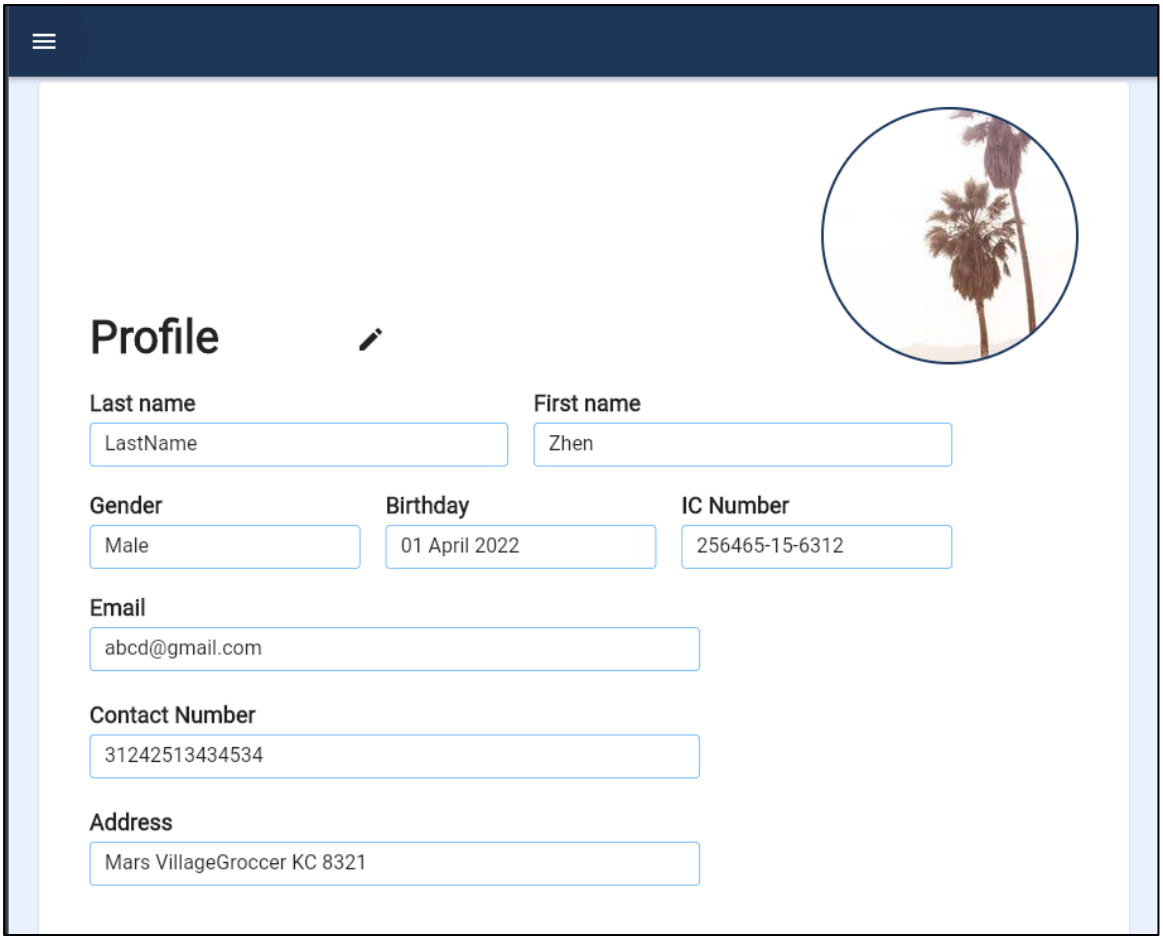


Figure 5.51 Responsive design of user profile

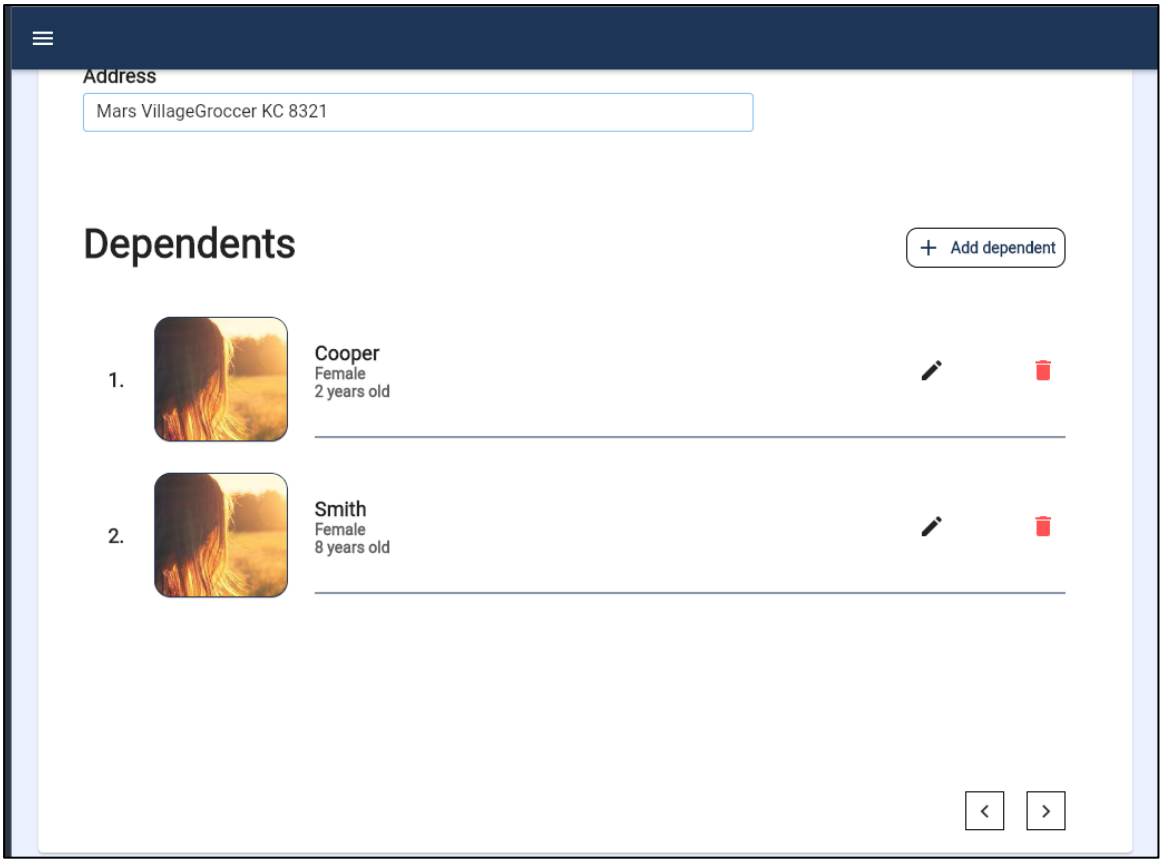


Figure 5.52 Responsive design of dependent section

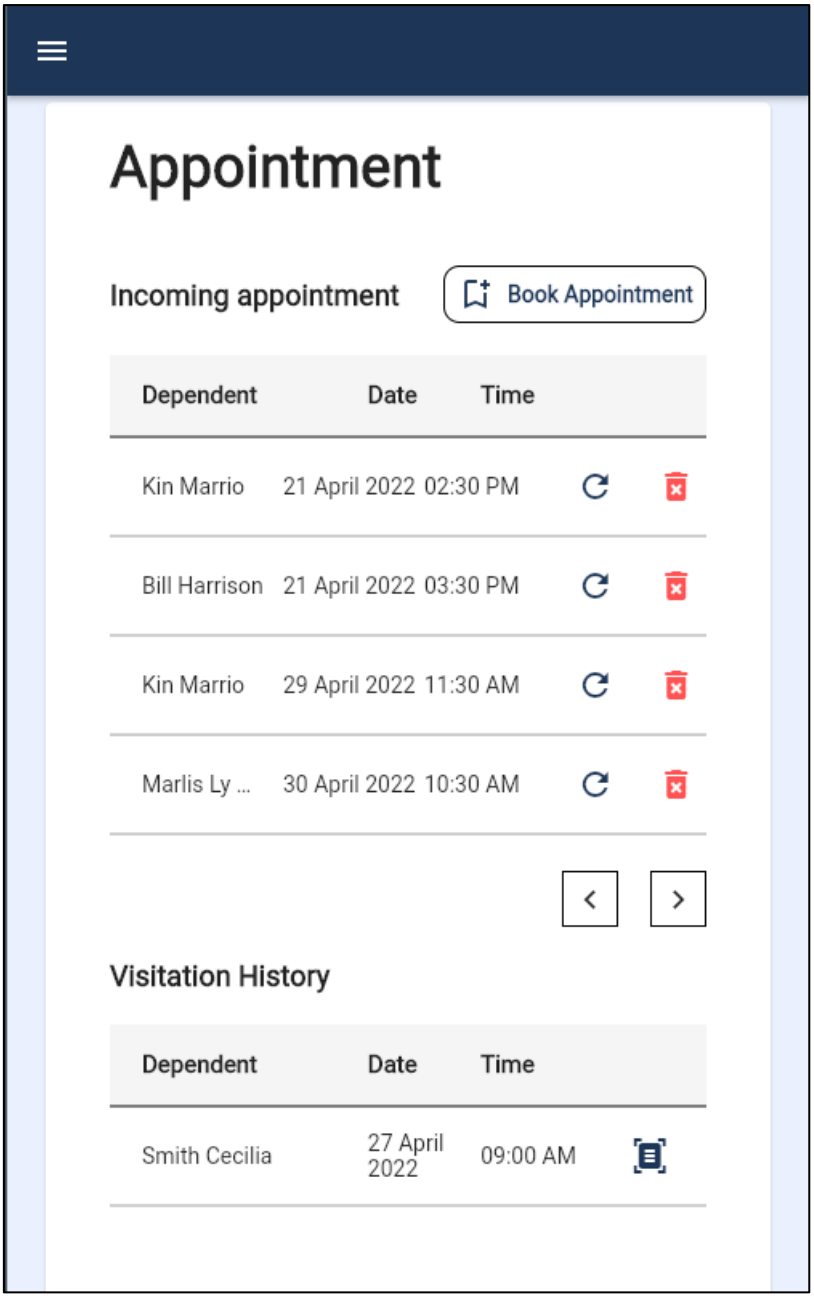


Figure 5.53 Responsive design of appointment and visitation section

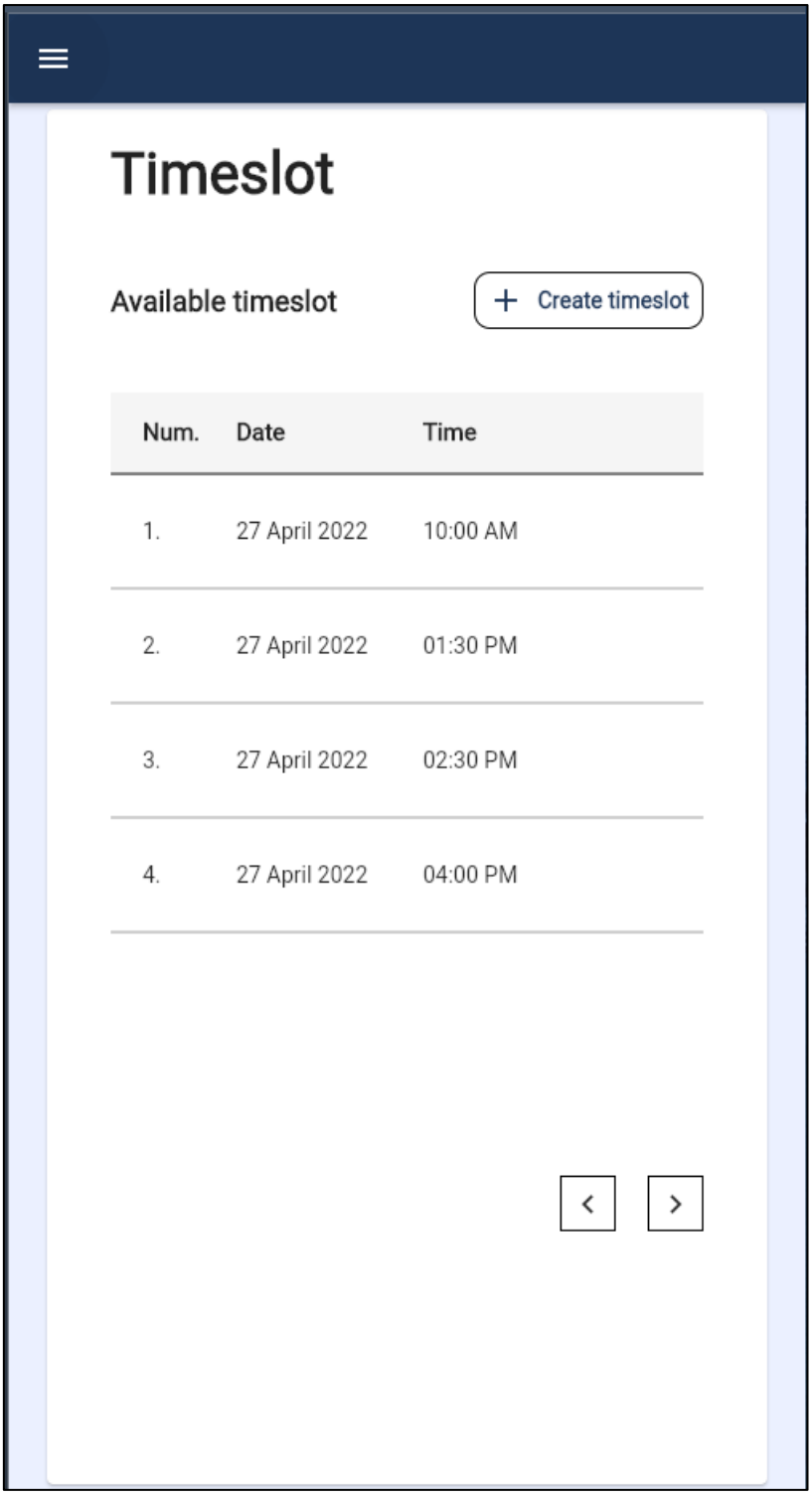


Figure 5.54 Responsive design of timeslot section

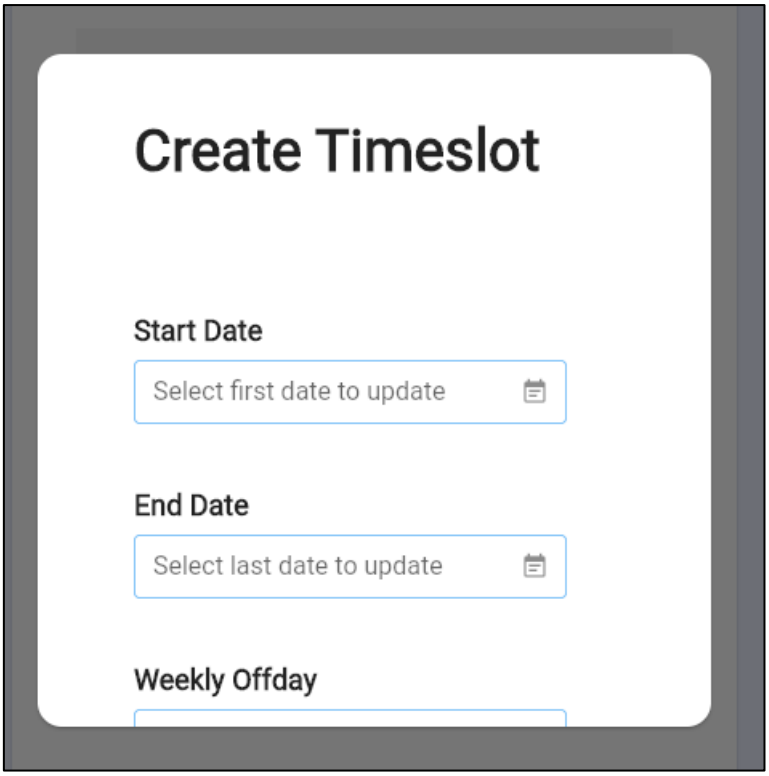


Figure 5.55 Responsive design of create timeslot form (part 1)

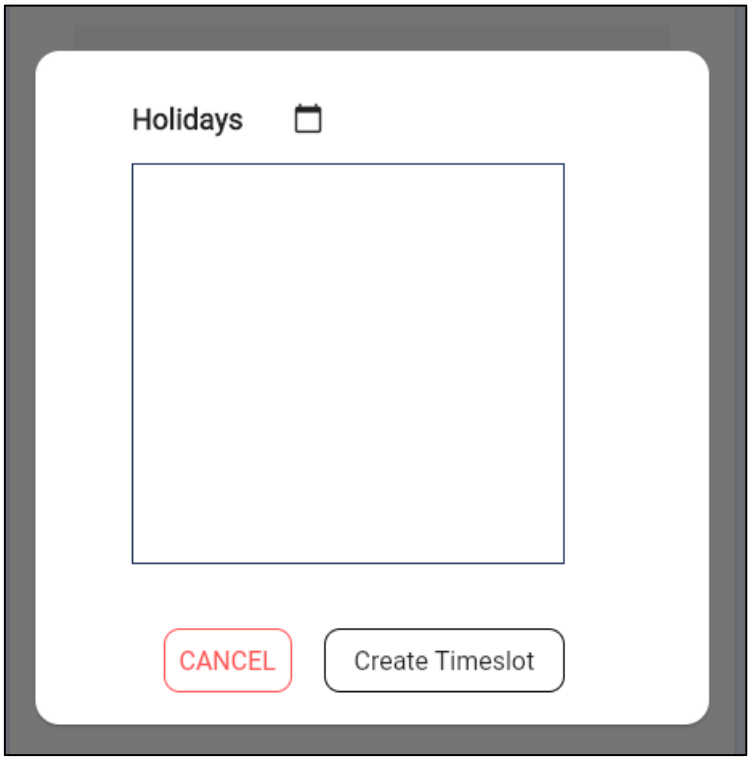


Figure 5.56 Responsive design of create timeslot form (part 2)

5.5 Concluding Remark

Through the implementation phase, a functional application is developed with respect to the system methodology and design proposed. The development of the application not only involves the design of the user interface, but also requires efforts in analyzing the system requirements and business logic to ensure the system performs expected behavior. The following list states the concluding remark for the application development:

- The development of the application is divided into 2 portions, which are the front-end Flutter application and back-end Spring Boot application.
- The Flutter application focuses on the UI presentation.
- The Spring Boot application focuses on the business logic processing and data manipulation.
- Both applications interact with each other through the developed APIs
- At the end of the development, both applications are dockerized to ease the deployment and distribution whenever necessary.

Chapter 6 System Evaluation and Discussion

6.1 System Testing and Performance Metrics

The system has been undergoing several testing approaches to evaluate the system quality. 3 testing methods, including use case testing, system performance testing and A/B testing are conducted to evaluate various system aspects.

In the use case testing, each of the use case will be conducted to test the behavior of the system. With the real outcomes collected, a comparison using the expected outcomes and real outcomes can be made, evaluating the correctness of the system functionality of that use case. In case the actual outcome varies with the expected outcome, a system defect may present in the system.

On the other hand, a system testing on the system performance is conducted by the actual execution of the system. The response time of each request-response cycle is measured and be averaged out to discover the average processing time required by the back-end application to return the expected outcome. Although the simulation through Postman has disregarded the potential delay caused by the front-end application, it is managed to reflect the system performance in terms of response speed and latency.

A/B testing facilitates the optimization of a system in terms of user experience, whereby the participants of the tests may help to explore the pain points of the system when they attempted to utilize the services provided by the system. With this, a A/B testing is conducted with 100 participants with survey approach. The participants are requested to rate the system from the perspective of UI consistency, density of data distributed at a page and simplicity for utilization after experiencing the various functions provided in the system.

6.2 Testing Setup and Result

6.2.1 Use Case Testing

In the use case testing, various test data are provided to the system to test out every potential flow of the use case.

Test Item	Login to system
Test Condition	Main flow
Test Data	(Valid credential) email: mario@gmail.com password: abcd1234
Expected Outcome	System approves the login and redirects the user to main page
Actual Outcome	System approves the login and redirects the user to main page
Result (Pass/Fail)	Pass

Table 6.1 Use case testing for main flow of login to system

Test Item	Login to system
Test Condition	Alternate flow – empty field for email or password
Test Data	email: password: abcd1234
Expected Outcome	System detects the empty field and prompt the user to fill out the empty field.
Actual Outcome	System detects the empty field and prompt the user to fill out the empty field.
Result (Pass/Fail)	Pass

Table 6.2 Use case testing for first alternate flow of login to system

Test Item	Login to system
Test Condition	Alternate flow – invalid email or password
Test Data	(Invalid credential) email: abc@gmail.com password: abcd1234
Expected Outcome	System detects invalid credential and display access denied message.
Actual Outcome	System detects invalid credential and display access denied message.
Result (Pass/Fail)	Pass

Table 6.3 Use case testing for second alternate flow of login to system

Test Item	Register account
Test Condition	Main flow
Test Data	(Valid credential) email: abc@gmail.com password: abcd1234 confirm password: abcd1234
Expected Outcome	System approves the registration and directs the user to the main page.
Actual Outcome	System approves the registration and directs the user to the main page.
Result (Pass/Fail)	Pass

Table 6.4 Use case testing for the main flow of register account

Test Item	Register account
Test Condition	Alternate flow – empty field or password
Test Data	email: password: abcd1234 confirm password: abcd1234
Expected Outcome	System detects the empty field and prompt the user to fill out the empty field.
Actual Outcome	System detects the empty field and prompt the user to fill out the empty field.
Result (Pass/Fail)	Pass

Table 6.5 Use case testing for first alternate flow of register account

Test Item	Register account
Test Condition	Alternate flow – non-matching password & confirm password value
Test Data	email: abcd@gmail.com password: abcd1234 confirm password: abcd
Expected Outcome	System detects the errors and prompt the user to review password field.
Actual Outcome	System detects the errors and prompt the user to review password field.
Result (Pass/Fail)	Pass

Table 6.6 Use case testing for second alternate flow of register account

Test Item	Register account
Test Condition	Alternate flow – incorrect format of email or password
Test Data	email: abcd (not an email) password: 3322 (< 8 characters) confirm password: 3322
Expected Outcome	System detects the errors and prompt the user to review value provided.
Actual Outcome	System detects the errors and prompt the user to review value provided.
Result (Pass/Fail)	Pass

Table 6.7 Use case testing for third alternate flow of register account

Test Item	Register account
Test Condition	Alternate flow – invalid value provided in user profile form
Test Data	No value is inputted into the user profile form
Expected Outcome	System detects the errors and prompt the user to fill out the empty fields.
Actual Outcome	System detects the errors and prompt the user to fill out the empty fields.
Result (Pass/Fail)	Pass

Table 6.8 Use case testing for fourth alternate flow of register account

Test Item	Register account
Test Condition	Alternate flow – email not exists
Test Data	Email: 3986129837@gmail.com Password: abcd1234 Confirm password: abcd1234
Expected Outcome	System detects the non-existence email and notifies the user regarding the error.
Actual Outcome	System detects the non-existence email and notifies the user regarding the error.
Result (Pass/Fail)	Pass

Table 6.9 Use case testing for fifth alternate flow of register account

Test Item	View profile
Test Condition	Main flow
Test Data	User clicks on the profile tab.
Expected Outcome	System navigates the user to profile and displays the user profile data.
Actual Outcome	System navigates the user to profile and displays the user profile data.
Result (Pass/Fail)	Pass

Table 6.10 Use case testing for main flow of view profile

Test Item	Update profile
Test Condition	Main flow
Test Data	Last name: ABC First name: DEF Gender: Male Birthdate: 1 April 2016 IC Number: 123456789101 Email: abcd@gmail.com Contact number: 0163598265 Address: Village Mario
Expected Outcome	System approves the update action and shows the updated user profile.
Actual Outcome	System approves the update action and shows the updated user profile.
Result (Pass/Fail)	Pass

Table 6.11 Use case testing for main flow of update profile

Test Item	Update profile
Test Condition	Alternate flow – Incorrect format of updated value
Test Data	Last name: First name: DEF Gender: Male Birthdate: 1 April 1978 IC Number: Email: abcd@gmail.com Contact number: 0163598265 Address: Village Mario
Expected Outcome	System detects the fields with invalid values and prompt user to update the values.
Actual Outcome	System detects the fields with invalid values and prompt user to update the values.
Result (Pass/Fail)	Pass

Table 6.12 Use case testing for alternate flow of update profile

Test Item	Create dependent
Test Condition	Main flow
Test Data	Last name: Smith First name: John Gender: Male Birthdate: 1 April 2017 IC Number: 125651239785 Allergies: None
Expected Outcome	System approves the creation and displays the created dependent in dependent section.
Actual Outcome	System approves the creation and displays the created dependent in dependent section.
Result (Pass/Fail)	Pass

Table 6.13 Use case testing for main flow of create dependent

Test Item	Create dependent
Test Condition	Alternate flow – incorrect format of value provided
Test Data	Last name: First name: John Gender: Male Birthdate: IC Number: Allergies: None
Expected Outcome	System detects the errors and prompts the user to update the error fields.
Actual Outcome	System detects the errors and prompts the user to update the error fields.
Result (Pass/Fail)	Pass

Table 6.14 Use case testing for alternate flow of create dependent

Test Item	View dependent
Test Condition	Main flow
Test Data	User clicks on the profile tab.
Expected Outcome	System navigates the user to profile page and displays the dependent data in the bottom layout.
Actual Outcome	System navigates the user to profile page and displays the dependent data in the bottom layout.
Result (Pass/Fail)	Pass

Table 6.15 Use case testing for main flow of view dependent

Test Item	Update dependent
Test Condition	Main flow
Test Data	Last name: Smith First name: John Gender: Male Birthdate: 1 April 2017 IC Number: 125651239785 Allergies: None
Expected Outcome	System approves the update and displays the updated dependent in dependent section.
Actual Outcome	System approves the creation and displays the updated dependent in dependent section.
Result (Pass/Fail)	Pass

Table 6.16 Use case testing for main flow of update dependent

Test Item	Update dependent
Test Condition	Alternate flow – incorrect format for value provided
Test Data	Last name: First name: Gender: Birthdate: 1 April 2017 IC Number: 125651239785 Allergies: None
Expected Outcome	System detects the errors and prompt the users to update the error fields.
Actual Outcome	System detects the errors and prompt the users to update the error fields.
Result (Pass/Fail)	Pass

Table 6.17 Use case testing of alternate flow of update dependent

Test Item	Delete dependent
Test Condition	Main flow
Test Data	The remove icon on a dependent record is clicked.
Expected Outcome	System removes the dependent record and update the dependent list.
Actual Outcome	System removes the dependent record and update the dependent list.
Result (Pass/Fail)	Pass

Table 6.18 Use case testing for main flow of delete dependent

Test Item	Book appointment
Test Condition	Main flow
Test Data	Date: 27 April 2022 Timeslot: 9.30 am Dependent: dependent 1 Description: None
Expected Outcome	System approves the booking and displays the booked appointment in the incoming appointment table.
Actual Outcome	System approves the booking and displays the booked appointment in the incoming appointment table.
Result (Pass/Fail)	Pass

Table 6.19 Use case testing for main flow of book appointment

Test Item	Book appointment
Test Condition	Alternate flow – No dependent is selected
Test Data	Date: 27 April 2022 Timeslot: 9.30 am Dependent: Description: None
Expected Outcome	System detects the error and prompt the user to select a dependent.
Actual Outcome	System detects the error and prompt the user to select a dependent.
Result (Pass/Fail)	Pass

Table 6.20 Use case testing for first alternate flow of book appointment

Test Item	Book appointment
Test Condition	Alternate flow – No timeslot is selected
Test Data	Date: 27 April 2022 Timeslot: Dependent: Mario Description: None
Expected Outcome	System detects the error and prompt the user to select a timeslot.
Actual Outcome	System detects the error and prompt the user to select a timeslot.
Result (Pass/Fail)	Pass

Table 6.21 Use case testing for second alternate flow of book appointment

Test Item	View booked appointment
Test Condition	Main flow
Test Data	The appointment tab in side menu is selected.
Expected Outcome	System retrieves the users' booked appointments and display them in incoming appointment table.
Actual Outcome	System retrieves the users' booked appointments and display them in incoming appointment table.
Result (Pass/Fail)	Pass

Table 6.22 Use case testing for main flow of view booked appointment

Test Item	Reschedule appointment
Test Condition	Main flow
Test Data	Date: 30 April 2022 Timeslot: 3.30 pm
Expected Outcome	System approves the reschedule and displays the updated appointment in the incoming appointment table.
Actual Outcome	System approves the reschedule and displays the updated appointment in the incoming appointment table.
Result (Pass/Fail)	Pass

Table 6.23 Use case testing for main flow of reschedule appointment

Test Item	Cancel appointment
Test Condition	Main flow
Test Data	The remove icon on an appointment record is clicked.
Expected Outcome	System approves the cancellation and displays the remaining appointment records in the incoming appointment table.
Actual Outcome	System approves the cancellation and displays the remaining appointment records in the incoming appointment table.
Result (Pass/Fail)	Pass

Table 6.24 Use case testing for main flow of cancel appointment

Test Item	View available timeslot
Test Condition	Main flow
Test Data	The timeslot tab is selected from the side menu.
Expected Outcome	System navigates to the timeslot page and displays all available timeslots in the timeslot table.
Actual Outcome	System navigates to the timeslot page and displays all available timeslots in the timeslot table.
Result (Pass/Fail)	Pass

Table 6.25 Use case testing for main flow of view available timeslot

Test Item	Create timeslot
Test Condition	Main flow
Test Data	Start date: 21 April 2022 End date: 30 April 2022 Weekly off-day: Monday Holidays: 25 April 2022
Expected Outcome	System creates the timeslots and displays them in available timeslot table.
Actual Outcome	System creates the timeslots and displays them in available timeslot table.
Result (Pass/Fail)	Pass

Table 6.26 Use case testing for main flow of create timeslot

Test Item	Complete appointment
Test Condition	Main flow
Test Data	Diagnosis result: Headache Prescription: paracetamol – 50mg
Expected Outcome	System completes the appointment with the data provided. A visitation record is generated and be shown in the visitation section.
Actual Outcome	System completes the appointment with the data provided. A visitation record is generated and be shown in the visitation section.
Result (Pass/Fail)	Pass

Table 6.27 Use case testing for main flow of complete appointment

Test Item	View visitation history
Test Condition	Main flow
Test Data	The appointment tab is selected from the side menu
Expected Outcome	System retrieves the user's visitation history and displays them in the visitation record table.
Actual Outcome	System retrieves the user's visitation history and displays them in the visitation record table.
Result (Pass/Fail)	Pass

Table 6.28 Use case testing for main flow of view visitation history

6.2.2 System Testing

Postman is used to simulate the process of sending API requests from front-end application and be processed and returned by the back-end application. Among all of the functionality provided by the system, the create dependent record and book appointment use cases are selected to be conducted, whereby their respective API endpoints are set as the target of API invocations.

```
1
2  let index = pm.collectionVariables.get("index");
3  let apiCounter = index + 1;
4  let totalResponseTime = pm.collectionVariables.get("totalResponseTime");
5
6  // Sum up total response time
7  totalResponseTime += pm.response.responseTime;
8
9  // Store current response time
10 pm.collectionVariables.set(apiCounter.toString(), pm.response.responseTime);
11 // Store total response time
12 pm.collectionVariables.set("totalResponseTime", totalResponseTime);
```

Figure 6.1 Postman test script to obtain response time

For each of the use cases selected, a total of 100 requests are initiated and sent to the back-end application and the response time of each request is recorded. To avoid any failure case, the data required by the API is prepared priorly and be attached to the API before sending. The distribution of the response time is illustrated with a bar chart so that any significant difference in the response time can be easily identified. With the bar chart, the consistency level of the back-end application performance can also be observed. Then, all response time are then be averaged out to obtain the average response time of the API requests. The average response time is used to compare with the average response time of the other similar application.

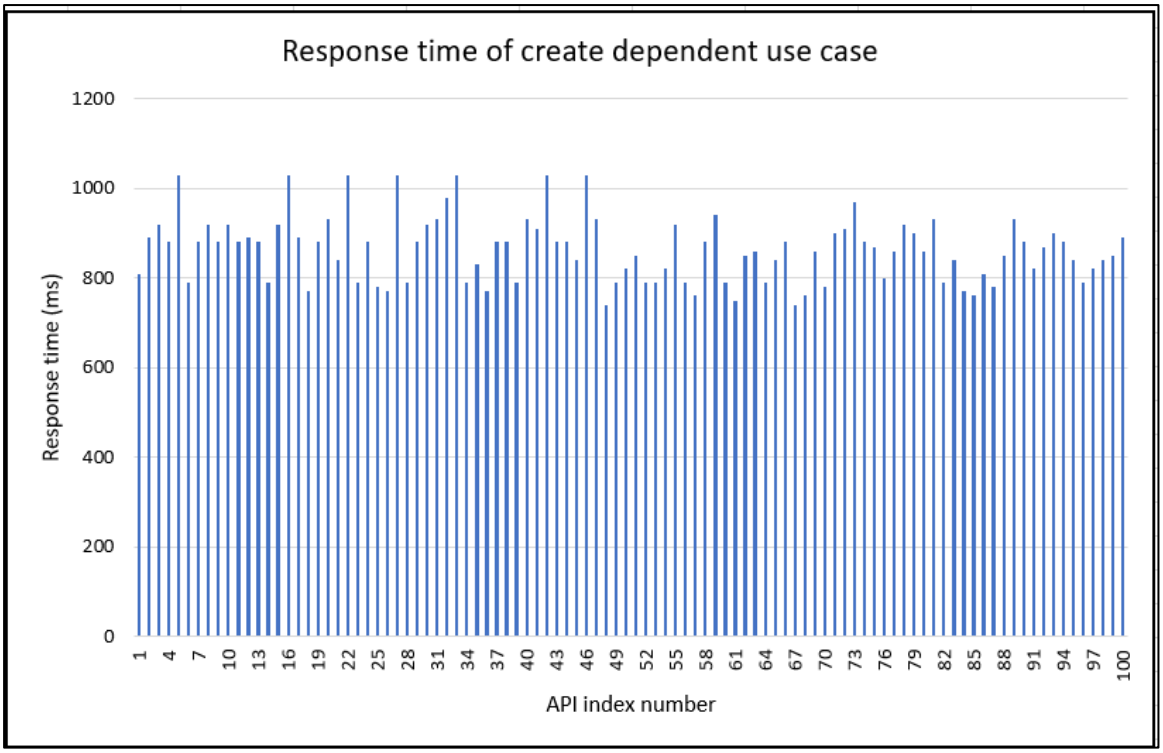


Figure 6.2 Chart for response time of create dependent

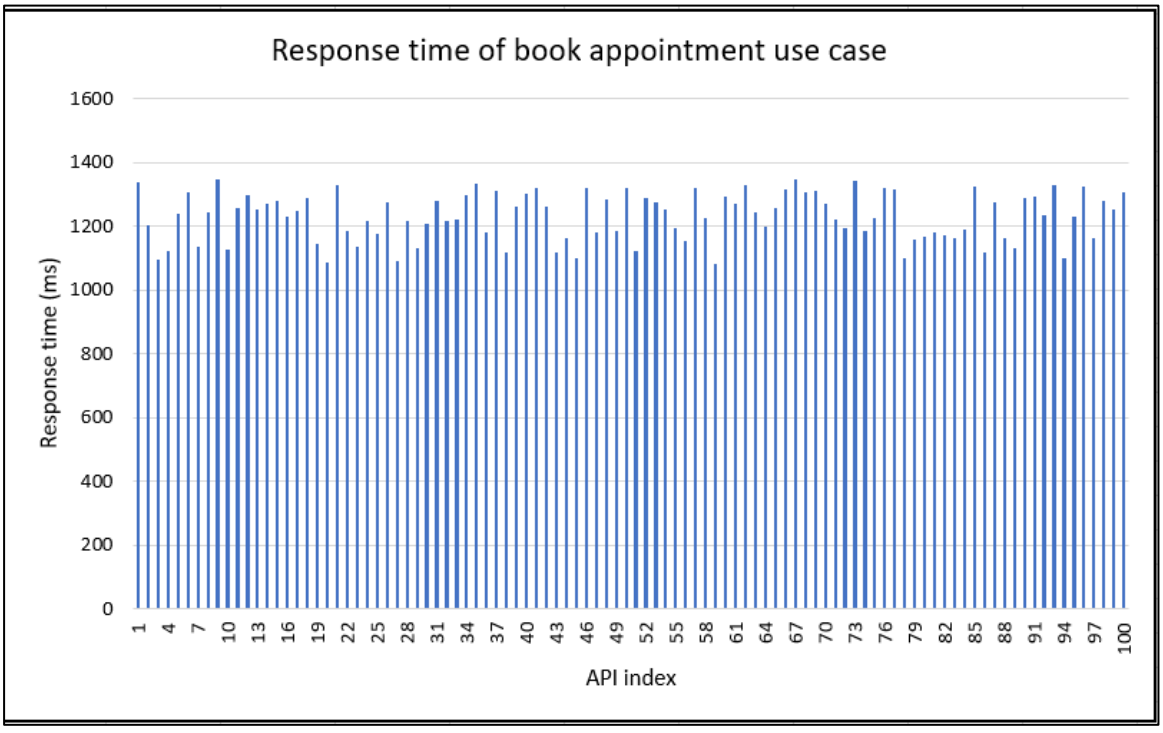


Figure 6.3 Chart for response time of book appointment

In terms of system performance consistency, it can be observed that the response time of each API request in both use cases are in average level, whereby there is no significant surge or drop in the response time among in the total 200 requests sent. According to the data collected, the create dependent has taken a maximum of 1030ms and a minimum of 740ms while the response time of book appointment has capped at 1348ms and grounded at 1080ms. In terms of average response time, the create dependent use case has taken an average of 863ms while the book appointment use case has a longer average time of 1215ms which may be caused by the more complex business logic as the booking of appointments involves an action of update timeslot availability internally.

6.2.3 A/B Testing

To perform the A/B testing, a survey form is prepared to allow the participants to rate their experience after trying out the application. A total of 100 participants are selected randomly to conduct the survey. To evaluate the user experience quantitatively, each participant is requested to create a user profile and perform an appointment booking using the provided system. Then, they are requested to answer the survey by giving rating score to the aspects of UI consistency, density of data distributed at a page and simplicity for utilization. The participants are allowed to refer to any similar system used before for the comparison purpose. The rate score is scaled as in Table 6.1.

Rating Score	**Description**
1	Very unsatisfied
2	Unsatisfied
3	Neutral
4	Satisfied
5	Very satisfied

Table 6.29 Rating scale of A/B testing survey

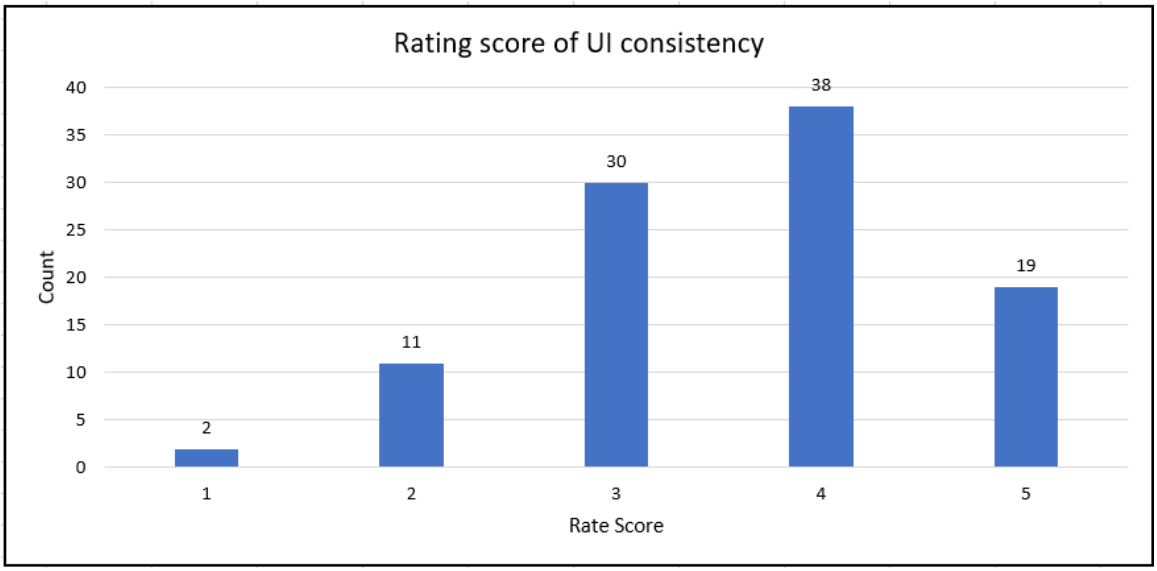


Figure 6.4 User rating for UI consistency of the system

Figure 6.5 User rating for the data density of the system

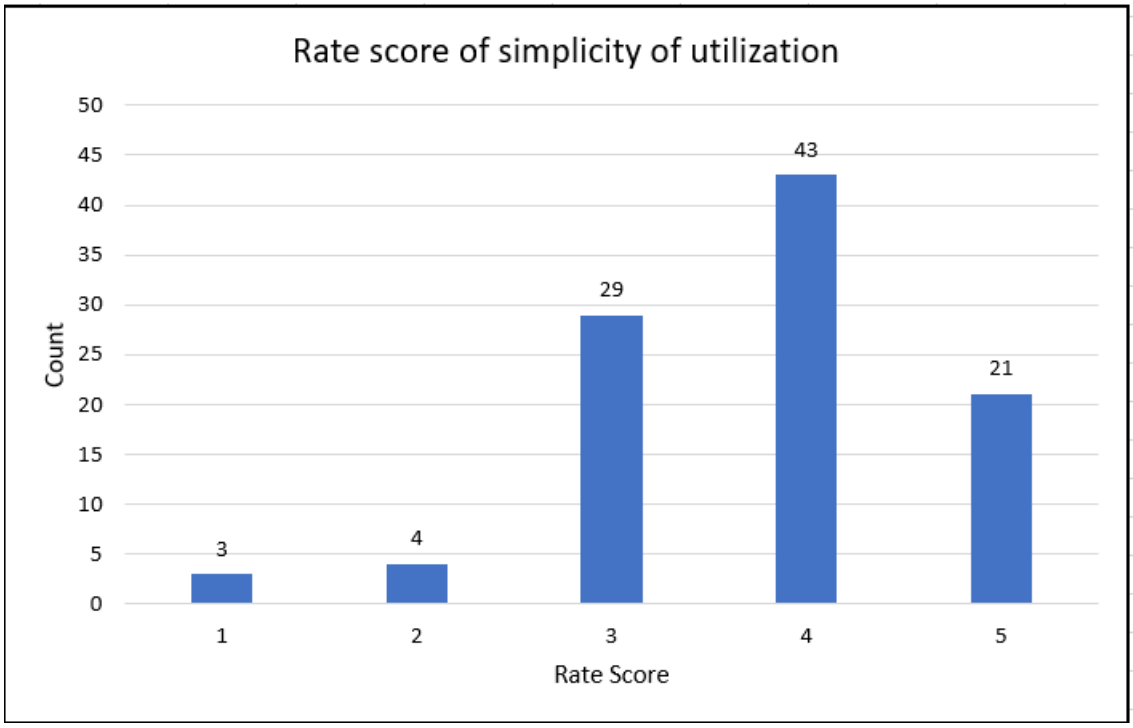


Figure 6.6 User rating for the simplicity of utilization of the system

As the result of the A/B testing, it can be observed that more than 50% of participants are satisfied for the system in terms of UI consistency, simplicity in utilization and data distributed in appropriate density. There are 19 out of 100 participants are very satisfied with the UI consistency presented by the system, allowing them to get familiar with the system in a quick manner. In terms of data density, a number of 19 participants are very satisfied with the distribution of the data in each page of the system, which can be interpreted as the users will not easily be overloaded since the huge amount data is not presented at once. Besides that, there are a total of 43 participants have responded with satisfied level and 21 participants are very satisfied for the simplicity of the system in performing the use cases requested.

6.3 Project Challenges

Several challenges are encountered during the development of the system. First of all, as the system will be adopted by a paediatric clinic for their business digitalization, the system developed must fulfilled their requirements. System implementation without prior planning is prohibited as it will certainly waste the development time due to insufficient understanding on the expected product. Hence, a huge effort has been invested into the planning and analysis phases of the software development lifecycle. In the planning phase, several similar systems are studied to analyze their advantages and drawbacks so that their merits can be applied to the planned system. Then, the functionality of the system is derived from the requirements and be transformed into various diagrams for reference during system implementation. The technology stack of the system has also been selected carefully so that the final product is capable to function while tolerating and adapting to various scale of traffic in the production environment. As a result, the final tech stack is formed by the Flutter Web, Spring Boot MVC and also the Docker technologies.

With this, another challenge faced is actually on the Flutter Web due to its unstable nature. As the Flutter Web has just freshly published, some widgets available in the base Flutter are not applicable in the Web environment. The external packages offered in the Flutter repository may also not be compatible with the Flutter Web due to its frequent update, increasing the difficulty level in searching for suitable packages. The Flutter SDK adopted must also be updated from time to time so that the bugs discovered in Flutter Web can be patched.

Moreover, as the front-end and back-end application of the system is separated, they required certain communication channels in order to perform the functionality of the system not only in the development stage but also in the production environment. Thus, a stable and reliable connection must be established for the communication purpose. This leveraged the importance of the Docker technologies which utilized prior configuration to ensure the stability of the system environment by preparing the exact same development environment for the system execution in the production stage. Hence, extra configuration and setting have to be done after the system development.

6.4 Objectives Evaluation

Objective	To develop a web application for patient record management and appointment scheduling for a paediatric clinic using Flutter
How to achieve	As in the final product, the patient record management are developed and handled by the user profile and dependent module. These modules not only allow the admin to manage the patient record but also allows the user to manipulate own data within given privilege. For the appointment scheduling, the timeslot module and appointment module are designed to ease the process. The timeslot module allows the admin to create timeslot for the appointment session. These timeslots will then be opened to the users for the booking process. The booking, rescheduling and cancellation of an appointment can then be performed by the users itself, reducing the workload significantly.

Table 6.30 Objective evaluation of the system (Part 1)

Objective	To build and connect real-world patient database using Google Firebase that connects to the Web UI
How to achieve	A Google cloud service, Cloud Firestore is selected to construct the patient database. Through Cloud Firestore, all of the patient data are stored in the cloud, making them having almost 24-hours availability with minimal risk of encountering disaster. To interact with the Cloud Firestore, the back-end Spring Boot application has implemented Firebase Admin SDK to develop intended features and API including the create patient record function. With the exposed APIs, the Web-UI presented by the Flutter application will then allow various operations offered by the clinic particularly appointment booking and patient record management to the users. To further enhance the security aspects, the Firebase Authentication has been implemented so that each patient is associated with proper credential. To invoke the services provided, the users must login to the system so that their API requests are equipped with the authentication token.With this, the risk of data exposure to unauthorized person can be minimized.

Table 6.31 Objective evaluation of the system (Part 2)

Chapter 7 Conclusion

7.1 Conclusion

Conclusively, a web application which served as patient management system with focuses on patient record management and appointment scheduling features are developed as the final product of the project. In terms of application architecture, the application has integrated the Flutter Web for web user interface development, Spring Boot for the business logic processing and Docker technology to simplify the deployment and distribution process.

Various modules in the application provides convenience to both the users and the admin. From the perspective of users, the user profile and dependent modules allow them to create and update their relevant information. They can maintain their children records through updating their dependent records whenever necessary. With the timeslot and appointment modules, the appointment scheduling process has been simplified whereby the users is managed to view the available timeslots directly and choose for the intended timeslot for appointment. The booked timeslot will be reserved and no longer be available for other users, reducing the effort required in the coordination between the clinic and the users. To recall their previous diagnosis result and the related prescription, the users can go through the visitation record to obtain the required information.

Through the implementation of the application, the digitalization of the clinic operations can be achieved while providing convenience in various operations especially appointment booking as these actions no longer required continuous maintenance or assistance from the admin when the users intend to perform operations remotely. As the digitalization of business becomes inevitable along with the global trend, the adoption of the application developed facilitates their digitalization at relatively small upfront costs, reducing the concern in terms of expensive investment cost in the transition phase from traditional business to the digitalization.

7.2 Recommendation

Although the current system has limited features, the application has great potential for the value-added services such as digital billing system and reminder system. The adoption of Spring Boot MVC with Repository and DAO layers as its internal architecture pattern allows the development and integration of new features with less effort due to the isolation of layers. The adding of new features into the system will bring extremely small impacts to the initial features in case there are not related. Besides that, the adoption of MVVM model in the Flutter application also improved the maintainability of the system as the code for presentation layer and business logic layer are segregated.

To further improve the quality of the system, several aspects especially the exception handling and value-added services can be taken into consideration. Instead of informing the users regarding the failure of the actions, the system may store the current state of the system temporarily so that these actions can be reattempted without requiring the user to repeat the form submission. In addition, some external services such as Camunda platform can be integrated to replace the Spring Boot scheduler. With the BPMN model constructed with Camunda, the automated tasks can be performed with better flexibility.

Bibliography

[1] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[2] H. Varshney, A. S. Allahloh, and M. Sarfraz, "IoT Based eHealth Management System Using Arduino and Google Cloud Firestore," in *2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2019, pp. 1–6.

[3] "Patient Manager," *Vertikal System*, 2021. https://www.vertikalsystems.com/en/products/pm/clinic-management.htm (accessed Aug. 11, 2021).

[4] "eClinic Systems," *Adroit InfoSystems*. https://www.adroitinfosystems.com/products/eclinic-systems (accessed Aug. 10, 2021).

[5] "Jane App - Practice Management Software for Health & Wellness Practitioners," 2021. https://jane.app/

[6] "ENCORE Clinic Management Software for medical and dental clinics," 2014. https://www.encoreeservices.com/clinic/clinicsoftware.htm

[7] "Kareo Clinical -EHR Software," 2021. https://www.kareo.com/ehr

[8] "Canada's Clinic Software for Managing Growing Clinics | Juvonno," 2021. https://juvonno.com/

[9] "The Smart Choice in Clinic Management Software," *e-clinic*. https://e-clinic.co.uk (accessed Aug. 10, 2021).

[10] C. Anderson, "The Model-View-ViewModel (MVVM) Design Pattern," in *Pro Business Applications with Silverlight 5*, C. Anderson, Ed. Berkeley, CA: Apress, 2012, pp. 461–499. doi: 10.1007/978-1-4302-3501-9_13.

[11] A. Bansal, "DAO vs Repository Patterns," *Baeldung*, Mar. 31, 2022. https://www.baeldung.com/java-dao-vs-repository (accessed Apr. 01, 2022).

[12] Eric Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, *1st edition*, 1st edition. Addison-Wesley Professional, 2003.

Appendices

None

Final Year Project Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.: 1**
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Performed simple requirements analysis.

2. WORK TO BE DONE

Continue to analysis the requirements.

3. PROBLEMS ENCOUNTERED

Lack of familiarity with the operations of the clinic. Thus, reference and study on other similar system has to be conducted.

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 2
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Partial completion on the requirements analysis.

2. WORK TO BE DONE

Continue to analysis the requirements.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 3
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Completed the system requirement analysis. Use case diagram has been developed.

2. WORK TO BE DONE

Produce use case description to further refine the system use cases.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 4
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Partial completion on the use case description and activity diagram

2. WORK TO BE DONE

Continue on the use case description and activity diagram drawing.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 5
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

System planning is done and currently be progressed to the system design.

2. WORK TO BE DONE

Determine the object model used with system components specifications and data dictionary.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 6
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Partial completion on data dictionary

2. WORK TO BE DONE

Continue to work on data dictionary

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 7
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Completed system design.

2. WORK TO BE DONE

Implements the system with code development.

3. PROBLEMS ENCOUNTERED

Packed schedule due to huge number of code development.

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently. Issues encountered during development are solved through online solution and supervisor assistance.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 8
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Partial completion on the system development

2. WORK TO BE DONE

Continue to work on system development

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 9
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Completed patient record management module and partial of the appointment scheduling module.

2. WORK TO BE DONE

Completes the development of appointment scheduling module.

3. PROBLEMS ENCOUNTERED

Packed schedule due to huge number of code development.

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently. Issues encountered during development are solved through online solution and supervisor assistance.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 10
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Partial completion on appointment scheduling module.

2. WORK TO BE DONE

Continue to work on system development on appointment scheduling.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 11
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

System implementation has been done.

2. WORK TO BE DONE

Starts the system testing and evaluation.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently with 80% of completion.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 13
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Done use case testing and system testing.

2. WORK TO BE DONE

Continue to work on A/B testing.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently with 80% of completion.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 13
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun Yichiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

System testing and evaluation has been completed.

2. WORK TO BE DONE

Prepare report of the project.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

Progressing fluently with 90% of completion.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan 2021	**Study week no.:** 13
Student Name & ID: Lai Chin Si, 18ACB04057	
Supervisor: Dr. Aun YiChiet	
Project Title: A patient management system for paediatric clinic using Flutter	

1. WORK DONE

Both system development and report are completed.

2. WORK TO BE DONE

None

3. PROBLEMS ENCOUNTERED

None

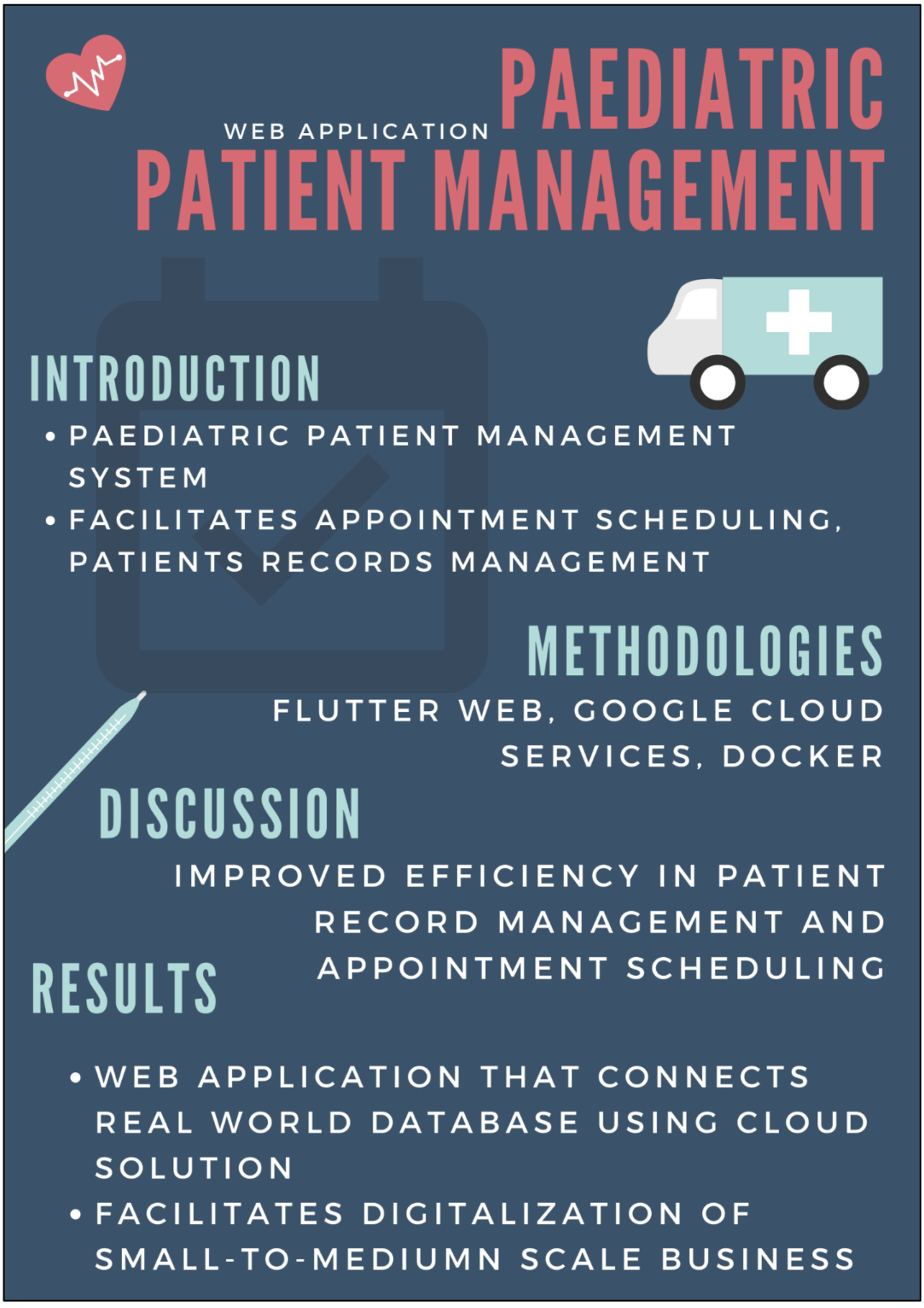
4. SELF EVALUATION OF THE PROGRESS

Completed the project and report within schedule.

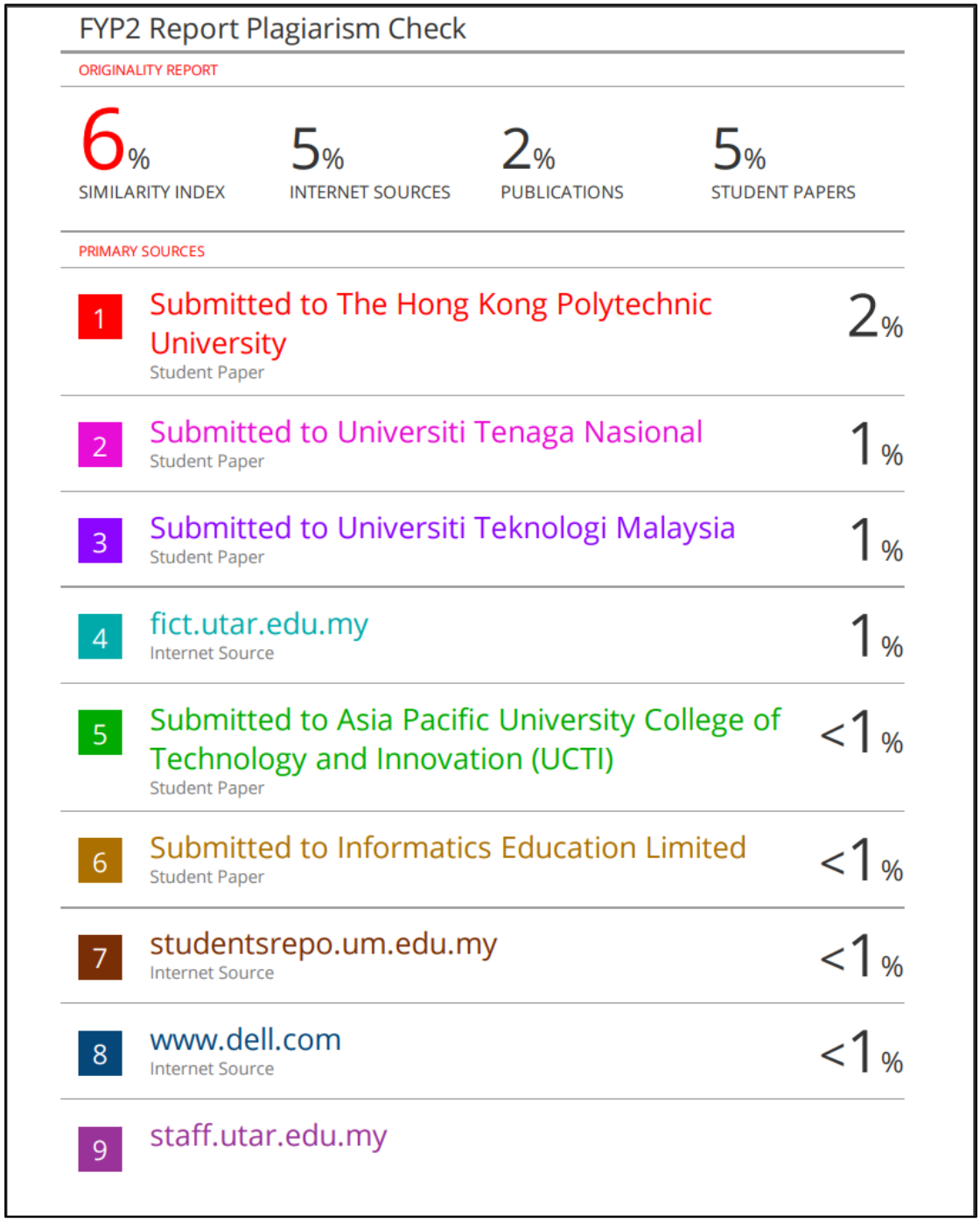
Supervisor's signature

Student's signature

Poster


WEB APPLICATION
PAEDIATRIC PATIENT MANAGEMENT
INTRODUCTION
• PAEDIATRIC PATIENT MANAGEMENT SYSTEM
• FACILITATES APPOINTMENT SCHEDULING, PATIENTS RECORDS MANAGEMENT
METHODOLOGIES
FLUTTER WEB, GOOGLE CLOUD SERVICES, DOCKER
DISCUSSION
IMPROVED EFFICIENCY IN PATIENT RECORD MANAGEMENT AND APPOINTMENT SCHEDULING
RESULTS
• WEB APPLICATION THAT CONNECTS REAL WORLD DATABASE USING CLOUD SOLUTION
• FACILITATES DIGITALIZATION OF SMALL-TO-MEDIUMN SCALE BUSINESS

Plagiarism Check Summary



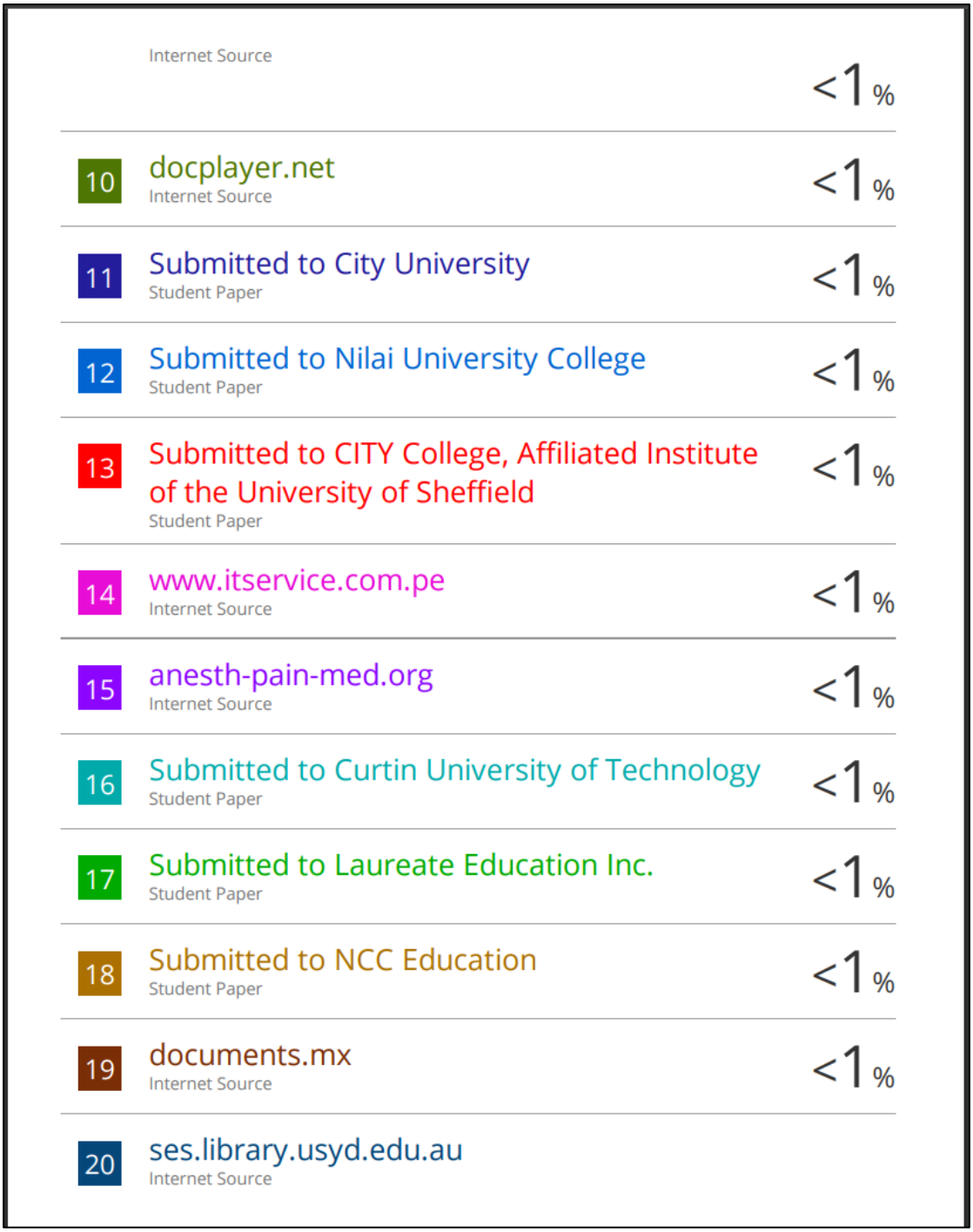
FYP2 Report Plagiarism Check

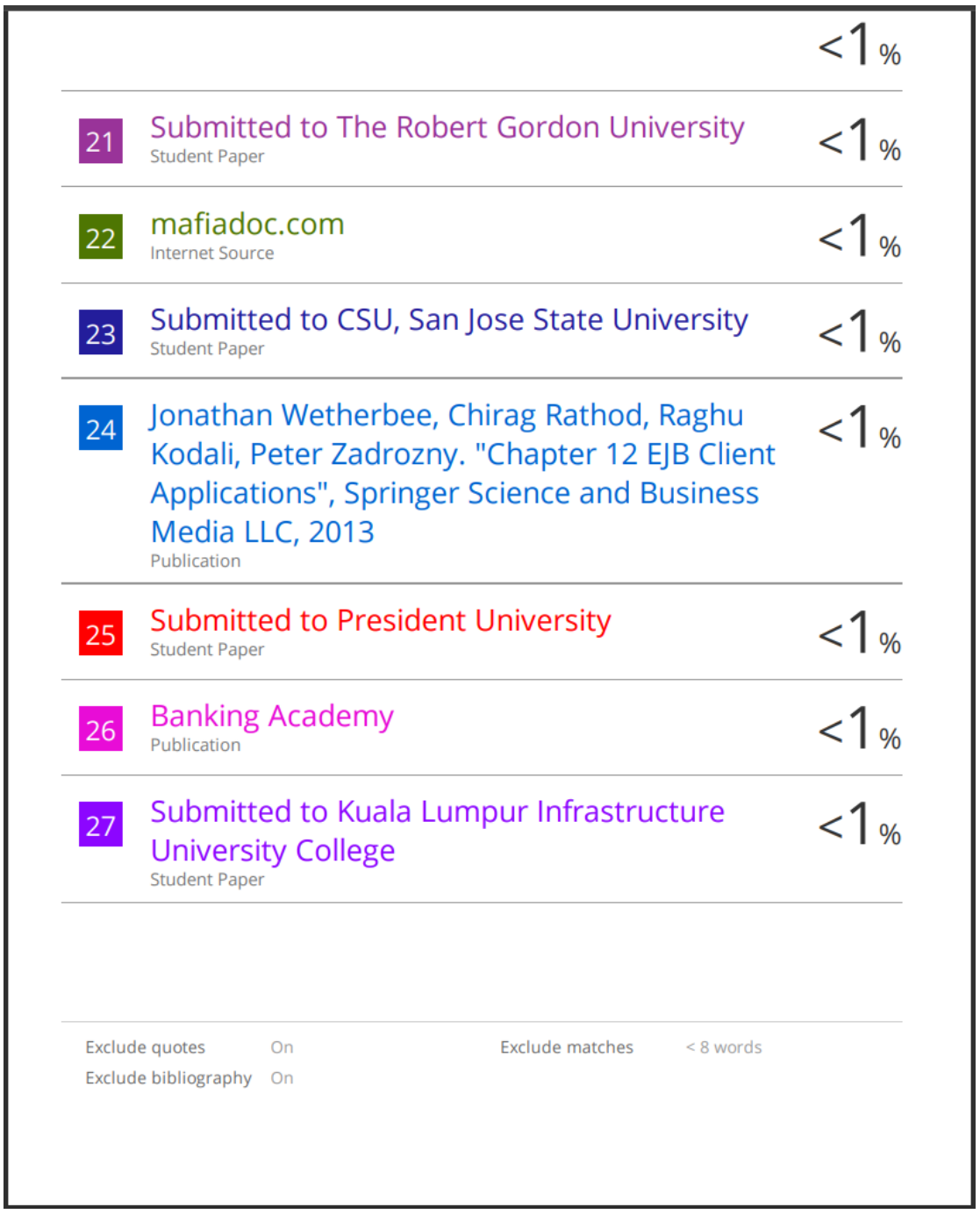
ORIGINALITY REPORT

6%	5%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to The Hong Kong Polytechnic University Student Paper	2%
2	Submitted to Universiti Tenaga Nasional Student Paper	1%
3	Submitted to Universiti Teknologi Malaysia Student Paper	1%
4	fict.utar.edu.my Internet Source	1%
5	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
6	Submitted to Informatics Education Limited Student Paper	<1%
7	studentsrepo.um.edu.my Internet Source	<1%
8	www.dell.com Internet Source	<1%
9	staff.utar.edu.my	


Internet Source
<1%
10 docplayer.net
Internet Source
<1%
11 Submitted to City University
Student Paper
<1%
12 Submitted to Nilai University College
Student Paper
<1%
13 Submitted to CITY College, Affiliated Institute of the University of Sheffield
Student Paper
<1%
14 www.itservice.com.pe
Internet Source
<1%
15 anesth-pain-med.org
Internet Source
<1%
16 Submitted to Curtin University of Technology
Student Paper
<1%
17 Submitted to Laureate Education Inc.
Student Paper
<1%
18 Submitted to NCC Education
Student Paper
<1%
19 documents.mx
Internet Source
<1%
20 ses.library.usyd.edu.au
Internet Source


<1%
21 Submitted to The Robert Gordon University
Student Paper
<1%
22 mafiadoc.com
Internet Source
<1%
23 Submitted to CSU, San Jose State University
Student Paper
<1%
24 Jonathan Wetherbee, Chirag Rathod, Raghu Kodali, Peter Zadrozny. "Chapter 12 EJB Client Applications", Springer Science and Business Media LLC, 2013
Publication
<1%
25 Submitted to President University
Student Paper
<1%
26 Banking Academy
Publication
<1%
27 Submitted to Kuala Lumpur Infrastructure University College
Student Paper
<1%
Exclude quotes On
Exclude bibliography On
Exclude matches < 8 words

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:	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	LAI CHIN SI
ID Number(s)	18ACB04057
Programme / Course	Bachelor of Computer Science (HONOURS)
Title of Final Year Project	A patient management system for paediatric clinic using Flutter

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 6 % **Similarity by source** Internet Sources: 5 % Publications: 2 % Student Papers: 5 %	
Number of individual sources listed of more than 3% similarity: 0	

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**

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

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

Name: Dr. Aun Yichiet

Date: 22 April 2022

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB04057
Student Name	LAI CHIN SI
Supervisor Name	Dr. Aun YiChiet

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.

I, the author, have checked and confirmed all the items listed in the table are included in my report.

_____

(Signature of Student)

Date: 21 April 2022