


# WOLKITE UNIVERSITY

## COLLEGE OF COMPUTING AND INFORMATICS

## DEPARTMENT OF COMPUTER SCIENCE

**Title: Android Based Patient Management System for Family Doctor**

**Team Members:**

Name……ID

Admasu Feleke…… ...CIR/403/11

Lidiya Fikadu……CIR/413/11

Yohannes Kebede……CIR/407/11

Advisor:

Mr. Ermiyas Nigatu

Wolkite, Ethiopia

Sunday, August 22, 2021.

## DECLARATION

This is to declare that this project work is done under the supervision of Mr. Ermiyas Nigatu and having the title patient management system is the contribution of Admasu Feleke, Lidiya Fikadu, Yohannes Kebede.

No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if a violation of this declaration is proven.

Date: _____

**Group Members:**

| Full name | Signature |
|---|---|
| 1. Admasu Feleke | _____ |
| 2. Lidiya Fikadu | _____ |
| 3. Yohannes kebede | _____ |

## APPROVAL FORM

This is to confirm that the project report entitled patient management system submitted to Wolkite University, College of Computing and Informatics Department of computer science by Admasu Feleke, Lidiya Fikadu, Yohannes Kebede is approved for submission.

| | | |
|---|---|---|
| Advisor Name | Signature | Date |
| Department Head Name | Signature | Date |
| Examiner 1 Name | Signature | Date |
| Examiner 2 Name | Signature | Date |
| Chairperson Name | Signature | Date |

## Acknowledgment

First of all, we would like to thank Almighty GOD for the strength, he has given us throughout our life and this project; nothing could happen without the help of GOD. Secondly, we would like to express our gratitude to our advisors **Mr. Ermiyas Nigatu (Msc)** for his help, willingness, and commitment in giving his precious time to help us to accomplish this work. We are also very grateful and would like to extend our sincere thanks to our staff members and students of our Department of Computer Science for sharing their ideas, suggestions, and support and especially for their commitment. We give great respect and credit to everyone who involved in our project task.

**TABLE OF CONTENTS**

DECLARATION ..... i
APPROVAL FORM ..... ii
Acknowledgment ..... iii
ACRONYMS AND ABREVIATIONS ..... ix
ABSTRACT ..... x
Chapter One ..... 1
1. Introduction ..... 1
1.1. Background ..... 1
1.2. Statement of the problem ..... 2
1.3. Objectives of the problem ..... 3
1.3.1. General objectives ..... 3
1.3.2. Specific objectives ..... 3
1.4. Feasibility Analysis ..... 3
1.4.1. Technical feasibility ..... 3
1.4.2. Operational feasibility ..... 4
1.4.3. Economic feasibility ..... 4
1.4.4. Time feasibility ..... 5
1.5. Project Scope and Limitation ..... 5
1.5.1. Scope ..... 5
1.5.2. Limitation ..... 6
1.6. Significance of the project ..... 6
1.7. Beneficiary of the project ..... 6
1.8. Methodology of the Project ..... 7
1.8.1. Data Collection Techniques ..... 7
1.8.2. System Analysis and Design ..... 7
1.8.3. System Development Model ..... 8
1.8.4. System Testing Methodology ..... 8
1.8.5. Development tools and technologies ..... 9
1.9. Team composition ..... 9
1.10. Organization of the document ..... 10

Chapter Two ..... 11
2. Description of the Existing System ..... 11
2.1. Introduction of the Existing System ..... 11
2.2. Users of Existing System ..... 11
2.2.1. Existing system description ..... 12
2.3. Major Function of the Existing System ..... 12
2.4. Drawback of the Existing System ..... 12
2.5. Business Rules of the Existing System. ..... 12
Chapter three ..... 14
3. Proposed system ..... 14
3.1. Introduction ..... 14
3.2. The Advantages of the proposed Application ..... 14
3.3. Disadvantage of the proposed Application ..... 15
3.4. Functional Requirements ..... 15
3.5. Non-functional Requirements ..... 15
3.5.1. User Interface and Human Factors ..... 15
3.5.2. Hardware Consideration (Android version, processor, memory) ..... 16
3.5.3. Security Issues ..... 16
3.5.4. Performance Consideration ..... 16
3.5.5. Error Handling and Validation ..... 16
3.5.6. Quality Issues ..... 16
3.5.7. Backup and Recovery ..... 16
3.5.8. Physical Environment ..... 17
3.5.9. Documentation ..... 17
Chapter Four: ..... 18
4. System Analysis ..... 18
4.1. System Modeling ..... 18
4.1.1. Use Case Model ..... 18
4.1.2. Class Diagram ..... 29
4.1.3. Data Dictionary ..... 29
4.2. Dynamic Model ..... 32
4.2.1. Sequence Diagram ..... 32

4.2.2. Activity Diagram ..... 39
4.2.3. State chart Diagram ..... 44
CHAPTER FIVE ..... 47
5. SYSTEM DESIGN ..... 47
5.1. Introduction ..... 47
5.2. Design Goals. ..... 47
5.3. Current Software Architecture ..... 48
5.4. Proposed Software Architecture ..... 48
5.4.1. Subsystem Decomposition ..... 48
5.4.2. Component Diagram ..... 49
5.4.3. Deployment Diagram ..... 51
5.4.4. Detailed Class Diagram ..... 52
5.4.5. Persistent Data Management ..... 53
5.4.6. Access Control and Security ..... 54
5.5. Package ..... 54
5.6. Algorithm Design ..... 55
5.7. User Interface Design ..... 57
CHAPTER SIX ..... 64
6. IMPLEMENTATION AND TESTING ..... 64
6.1. IMPLEMENTATION ..... 64
6.2. Implementation of the Database ..... 64
6.3. Implementation of the Class Diagram ..... 69
6.4. Configuration of the Application Server ..... 70
6.5. Configuration of Application Security ..... 70
6.6. Implementation of User Interface ..... 71
6.7. Testing ..... 71
6.7.1. Test Case. ..... 71
6.7.2. Testing Tools and Environment ..... 72
6.7.3. Unit Testing ..... 72
6.7.4. Integration Testing ..... 73
6.7.5. Acceptance Testing ..... 74
6.7.6. System Testing ..... 74

CHAPTER SEVEN ..... 75
7. CONCLUSION AND RECOMMENDATION ..... 75
7.1. Conclusion ..... 75
7.2. Recommendation ..... 75
Reference ..... 77
APPENDICES ..... 78
Appendix I ..... 78
Sample patient interview questions to the chatbot ..... 78
Appendix II ..... 79
Sample code for login ..... 79
Sample code for Signup ..... 84

**LIST TABLES**

Table 4.1 A use case description for signup for a system ..... 20
Table 4.2 A use case description of login use case ..... 20
Table 4.3 Description of registration of members ..... 20
Table 3. 4: Description of treatment use case ..... 21
Table 3.5.: Description of update member use case ..... 22
Table 3.6. Description for update treatment use case ..... 23
Table 3.7. Description for view information use case ..... 23
Table 3.8. Description for appointments use case ..... 23
Table 3.9. Description for view appointments use case ..... 24
Table 3.10 Description for update appointments use case ..... 25
Table 3.11 Description for delete appointments use case ..... 25
Table 3.12. Description for set reminder use case ..... 25
Table 3.13 Description for send message use case ..... 26
Table 3.14 Description for view message use case ..... 27
Table 3.15 Description for a chat with system chatbot use case ..... 27
Table 20. Data dictionary for signup table ..... 29
Table 21. Data dictionary for Customer Table ..... 30
Table 22 Data dictionary for Doctor table ..... 30
Table 23. Data dictionary for TreatmentInfo ..... 30
Table 24. Data dictionary of MemberDetails ..... 31
Table 25. Data dictionary of AppointmentDetails ..... 31
Table 26. Data dictionary of message ..... 32
Table 27. Data dictionary of chatbot ..... 32

Table 27 access control and security. ..... 54

## LIST FIGURES

Figure 4.1 System use case diagram for Android-based patient management system..... 19
Figure 2. Class diagram for the proposed system. ..... 29
Figure 3. sequence diagram for register member details ..... 33
Figure 4. Sequence Diagram for register treatment information. ..... 34
Figure 5. Sequence Diagram for update member information..... 35
Figure 9. Sequence Diagram for Set Appointment. ..... 36
Figure 16. A sequence Diagram for message for doctor. ..... 37
Figure 17. Sequence diagram for chatbot healthcare advisory. ..... 38
Figure 18. Activity Diagram for login ..... 40
Figure 19. Activity Diagram for registration of member. ..... 41
Figure 20. Activity Diagram for treatment record by the doctor. ..... 41
Figure 21. Activity Diagram for set Appointment. ..... 42
Figure 22. Activity Diagram for update member. ..... 43
Figure 23. Statechart diagram for login. ..... 45
Figure 24. Statechart diagram for register member. ..... 46
Figure 25. Proposed system architecture. ..... 48
Figure 26. Subsystem decomposition of Android-based patient management for family doctors. ..... 49
Figure 27. Component diagram of the system. ..... 50
Figure 28. Deployment diagram. ..... 51
Figure 29. Detailed class Diagram for the proposed system. ..... 52
Figure 30. Persistent data management. ..... 53
Figure 31. UML package diagram of the system. ..... 55
Figure 32. Patient signup user interface. ..... 59
Figure 33. User interface for login page. ..... 60
Figure 34. A user interfaces home page for doctor. ..... 61
Figure 35. User interface for health care chatbot system ..... 62
Figure 36. A user interfaces home page for patient. ..... 63

## ACRONYMS AND ABREVIATIONS

ABPMSFD: Android Based Patient Management System for Family Doctors.

ADT: Android Development Tool

GUI: Graphical User Interface.

IDE: Integrated Development Environment

OOSAD: Object-Oriented Software Analysis Design

SDK: Software Development Kit

UML: Unified Modeling Language

## ABSTRACT

*This project is an Android-based patient management system specially designed for family doctors. The application will be installed on smartphones. The application provides a user-friendly GUI in order to make it easy and efficient for the doctors to use the application. In this application, doctors can register members along with their address, patients in the family, keep track of appointments. It provides a chatting system for doctors and patients, which in turn enables easy communication. Supports knowledge base called chatbot system which assists patients in time if the doctor got busy. The system also provides time reminder for the doctor. In general, it enables doctors to have information about the families, family members, and patients in the family by installing this application on his/her mobile.*

# Chapter One

## 1. Introduction

Android is an open-source mobile operating system based on Linux with java support. It comes under free and open-source software licenses. A mobile application is gaining more and more popularity in the research world, even though it's a recent concept in information technology. This is because it is a way of collaboratively doing tasks or solving the problem. The mobile applications are developed in the Java language using the Android software development kit (SDK). The officially supported integrated development environment (IDE) is Android Studio using the Android Development Tools (ADT) plug-in. This project is designed and implement an Android-based patient management system for a family doctor. Mobile devices present many unique characteristics that make their use as mobile health particularly attractive, such as

- Ubiquity and convenience: mobile devices are portable, ubiquitous devices that come in many shapes and forms. Even though there are convenient portable computing devices and mobile phones are still the most widely employed ubiquitous computing devices readily available to doctors.
- Personalization: unlike PC's handheld devices are typically operated by single users, thereby enabling the provision of personalized service by various applications.

### 1.1. Background

Today, mobile phones are such a familiar part of our lives that we don’t only use them for simply talking or texting others but in many other aspects. Mobile health (using mobile devices in the health sector) is a young and dynamic field that could improve the well-being of all human beings around the world. This document is a report or a software requirement specification for an android based patient management system for a family doctor. Having a family doctor is not that much in practice in countries like Ethiopia, only those higher-class families engage themselves in this kind of scenarios, but with the growth in economy families will be able to have their family doctor, and also increase in the number of professionals (Doctors) on health sector will have a direct impact on the need to possess a family doctor.

In most of the areas across all over Ethiopia, since there is no such flexible application it's not familiar that family doctors use their cell phones to record patient's information. They use papers to record patient information.

Android-based patient management system for family doctor simply a mobile application that helps a family doctor to keep a record of medical history (which includes information about past illnesses of each and everyone in the families, information about blood relatives (for example parents, grandparents, children, and brothers or sisters)) of the intended families. Typically, this application is exceedingly useful by better record-keeping for a doctor when they have too many families that they are sighting into them.

## 1.2. Statement of the problem

Identifying the statement of the problem focuses on analyzing what problems the existing have and describe those problems and write out them understandably. In many cases, the mobile application system has brought new opportunities to the government, business companies, and educational institutions, and even to person. Now a day, mobile applications play an important role in the development of the country in respect of social, economic, and political levels.

Accordingly, users need to get enough information about every smart service using mobile applications. From these mobile applications, the user's need will be to get service like mobile health application

In some places over Ethiopia, the most presently existing system is that an intended family doctor sees people in families for all of their health problems, they record the necessary information in the paper-based form. As the number of families, the doctor's sighting increases, recording in paper-based format will become problematic.

The usage of traditional paper-based forms to record information has several drawbacks since it is erroneous, insecure, unorganized, and also less flexible. The following list shows the problems that a family doctor faces if he is going to record the family's medical information that he is going to sight.

- A paper-based recording system is not secure which means patient's information can be visible to others, it can also get damaged.
- It is difficult to whirl through papers to check the health mark of individuals in the intended family.

- It is also difficult to list all the appointments with patients on the paper, the paper may lose.
- The paper-based patient recording system is not flexible.
- A paper-based patient recording system isn't good enough for managing the patient's information.
- It is difficult for the doctor to remember every appointment he has with his patients.

## 1.3. Objectives of the problem

The general and specific objective of the project is described below.

### 1.3.1. General objectives

The general objective of this project is to develop a mobile-based patient management system for a family doctor.

### 1.3.2. Specific objectives

We listed the following aspects as specific objectives.

- To gather requirements and review related works.
- To determine how data will be entered into our system.
- To design a user-friendly interface.
- To develop a database that keeps the overall records associated with the management.
- To implement and test the system.

## 1.4. Feasibility Analysis

Since a mobile-based patient management system for a family doctor is very useful for the doctors in that it provides the prevailing betterment in their work, it is expected that this application will also grow faster and more productively if family doctors recognize the role of strategic financing from the system and interventions into using it. This application simplifies their work by providing much functionality by overcoming the described drawbacks above.

### 1.4.1. Technical feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system.

This project is feasible with the currently available hardware such as mobile devices and software's like an android (a mobile device OS). It is also feasible with the development areas such as the computer we used and the software such as android studio IDE and plug-ins.

### 1.4.2. Operational feasibility

The user should not misuse the system. To determine the operational feasibility of the system we should take into consideration the knowledge level of the users. Initially, this system is proposed to be designed for the novice user (for both doctors and patients who are capable of using android devices). This system is operationally feasible since the users are professional and known with the technologies and hence there is no need to come up with the user especially doctors too much to use the system. The application is expected to be very flexible for its users. And this is also one of the factors that make it operationally feasible.

### 1.4.3. Economic feasibility

The proposed system is going to be economically feasible in such a way that since the system will be developed for medical doctors so that doctors can afford to have smartphones and also to install this system and can benefit from it. Here, in determining benefits, there are two cases for benefit analysis such as Intangible and Tangible, and in determining cost analysis one time (initial) cost and recurring costs. The following are the tangible and intangible benefits and one-time and recurring costs that we have identified in this project.

**A. Tangible benefits: -** are those our project benefits that can convert into monetary values. For this project, we have identified the following tangible benefits (per year).

- ❖ Reduced Stationary Cost to record
- ❖ Reduction of error
- ❖ Increased speed of service
- ❖ Improvement in management planning and control.

**Intangible benefits: -** are those our project benefits that cannot convert into monetary values.

- ❖ Knowledge gain by the project developer.
- ❖ Increasing the competitiveness of the individual
- ❖ More timely, updated, and accurate information
- ❖ Improved productivity
- ❖ Improving the morale of our team.

- ❖ Faster decision making on the team member
- ❖ Facilitating information processing of our team

**B. One time (Initial) Cost**

These are costs incurred at the time of developing our project. For this project, we have come up with the following tangible costs and intangible costs**.**

- ✓ **Tangible cost**

Tangible costs are those costs that can be accurately put in the numeric and measurable form, i.e., in birr and with certainty. These costs include the cost of types of equipment such as computers, servers, terminals, and the costs of other resources which can be covered by the departments.

- ✓ **Intangible Costs**

Intangible costs are costs that will be incurred during the project but are difficult to be represented in monetary terms. The intangible costs involved in this project are: - Time and effort of the development team.

**C. Recurring Costs: -** cost incurred to use and maintain our project once developed.

### 1.4.4. Time feasibility

A time feasibility study will take into account the period in which the project is going to take up to its completion. A project will fail if it takes too long to be completed before it is useful. Typically, this means estimating how long the system will take to develop, and if it can be completed in a given period using some methods like payback period. Time feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable. So that, our project is feasible within the given period.

## 1.5. Project Scope and Limitation

### 1.5.1. Scope

The scope of this project is developing a mobile-based patient management system for family doctors. It supports only patient-related activities such as registering new patient families and their members, updating, searching, deleting, and accessing patient information.

- ❖ Easy family members and patient record management application.

- Set appointments with patient
- Set a reminder for each appointment
- Able to notification or alarming.
- Register patient details online.
- Record patient treatment information about patients.
- Chatting system
- Chatbot healthcare advisory.

### 1.5.2. Limitation

The following list shows some of the limitations of the project:

- Payment issue is manual
- Only support the English language.

The key challenging task for developing this mobile application is due to the limitations and numerous variations inherent to standard mobile devices.

## 1.6. Significance of the project

In the first instance, the healthcare application has become a boon for the doctors, nurses, and various other medical staff associated with health professionals. The application help to update the doctors about the health condition of their patients. It helps medical professionals to view all patient reports through the application which makes it easier to take quick and apt decisions in emergency cases.

The patients are the ones who receive the most benefits of the healthcare application whether it is scheduling a quick appointment with the doctor, finding an appropriate specialist with hands-on experience, viewing the medical test reports from labs online, know prescribed medicines.

It is also useful for the developer team to gain experience while developing the system.

## 1.7. Beneficiary of the project

Securely sharing electronic information with patients and other clinicians. Helping providers more effectively diagnose patients, reduce medical errors, and provide safer care. Improving patient and provider interaction and communication, as well as health care convenience. Enabling safer, more reliable prescribing.

## 1.8. Methodology of the Project

### 1.8.1. Data Collection Techniques

- For gathering requirements from users, we are going to have frequent interviews with users who are expected to be users of the system that we are going to develop.
- Observing the existing problem such that all patient history is manually accessed.
- By discussing and analyzing the problems with the project team.
- The other method is document analysis; we reviewed documents such as books, e-books, and some previously done project reports which are used as a reference to design the system we are going to develop.
- Internet

Internet is our main source of information for the requirement of our project

### 1.8.2. System Analysis and Design

A. **System Analysis:** Object-Oriented System Analysis and Design (OOSAD) using Unified Modeling Language (UML) and Visio Software. Because of the following reasons:

- These techniques enable to reduce the communication gap between users and designers.
- These techniques enable designers to model the real world accurately.
- These techniques have usability features (it allows to use of codes repeatedly on other systems).
- Allows full exploitation of the power of object-based and object-oriented programming languages
- Object-based models appeal to the workings of human cognition, and hence the human input into the development of a software system is likely to be more natural and less error-prone.

**B. Implementation**

- Graphics design tools (like Photoshop, paint)
- Database (like Firebase)
- Programming language (Java)
- Development and debugging tools (ADT plug-in and Android SDK).

### 1.8.3. System Development Model

The model that we will use in our system development life cycle is the waterfall model which constitutes the following phases:

- Project Initiation and Planning
- Requirement elicitation
- Analysis
- Design
- Implementation
- And Testing

### 1.8.4. System Testing Methodology

Software testing is an activity that should be done throughout the whole development process. Testing is a process to show the correctness of the program. Testing is checking the system workability in an attempt to discover errors and avoiding such errors from the system. In this, the team members will taste the entire system as a whole with all forms, code, modules. The following are different testing strategies: -

- **Unit testing**: - This project is going to be tested every module of the system separately It is often done by the team members of the project to test that whether the unit has been implemented or not in producing the expected output against the given input.
- **Integration testing: -** After unit testing, we integrate individual modules as a group and tested. We perform this testing after unit testing and before proceeding to acceptance testing.
- **Acceptance testing: -** At this level the project is the application is tested by the intended users of our system. A system is mainly developed for an end-user normally a customer of the organization. A system is said to be accepted if and only if the user of the system is satisfied.
- **System testing: -** It is the final step of testing. In this, the team members test the entire system as a whole with all forms, code, modules. This form of testing is popularly known as Black Box testing or System tests. In this, the team members test all the functionalities in the System. All errors in the forms, functions, modules are tested.

### 1.8.5. Development tools and technologies

The programming tools that the project concentrates on are the following:

#### 1.8.5.1. Frontend software requirements

- Android SDK 30
- Gradle 6.5
- Android Studio IDE
- Programming language (Java)

#### 1.8.5.2. Backend Software requirements

- Firebase real-time database, cloud Firestore, IBM Watson assistant, firebase authentication and authorization.

#### 1.8.5.3. Documentation and Modeling Tools

- Edraw Max-–to design Sequence Diagram, Class Diagram, Activity diagram, and Use case Diagram.
- MS word 2016: for documentation.
- Paint: for editing
- Window 10 pro Operating system
- MS PowerPoint for presentation

#### 1.8.5.4. Hardware requirements:

- Computer (8GB RAM)
- flash disk (16 GB)
- Hard disk: 500GB. etc.

## 1.9. Team composition

The project development team is organized by three members. Each of the members has full responsibility for the development system for timely and quality development. In the case of workload, there is high cooperation that overcomes the workload.

| ID | NAME | Responsibility |
|---|---|---|
| **403/11** | ADMASU FELEKE | All |
| **413/11** | LIDIYA FIKADU | All |
| **407/11** | YOHANNES KEBEDE | All |

## 1.10. Organization of the document

This document has five chapters including this chapter which gives highlight about the project we are going to develop. The second chapter is description of existing system; the third chapter is the proposed system; the fourth chapter is System Analysis and the fifth chapter is System design of the proposed system.

# Chapter Two

## 2. Description of the Existing System

### 2.1. Introduction of the Existing System

This section about the existing system of patient management system for a family doctor is going to be discussed. Having a family doctor is not that much in practice in countries like Ethiopia, only those higher-class families engage themselves in this kind of scenarios, but with the growth in economy families will be able to have their family doctor, and also increase in the number of professionals (Doctors) on health sector will have a direct impact on the need to possess a family doctor.

In most of the areas across all over Ethiopia, since there is no flexible application, it is done manually. The system that we are going to develop the most presently existing system is that an intended family doctor sees people in families for all of their health problems, they record the necessary information in the paper-based form. As the number of families, the doctor is sighting increases, recording is paper-based.

### 2.2. Users of Existing System

There are two users in the existing system that are doctor and patient.

Responsibilities of doctors include:-

- Register member manually
- Record treatment info manually
- Prescribe the right medication manually
- tell patients about the advantages, disadvantages, risks, and alternatives regarding a proposed treatment or operation, and.

Responsibilities of patients include:-

- provide correct and complete information about their health and past medical history.
- Patients are responsible for keeping appointments.

### 2.2.1. Existing system description

The existing system that the family doctors currently using is a manual system which means on paper-based manner. The information of the previous and new coming customers is recording on the paper and placed on the shelf. If the doctor needs to know some information about a specific family or patient, he/she has to refer to all the papers that are placed on the shelf. Therefore, developing an automated mobile application for family doctors is found to be important to address those problems.

## 2.3. Major Function of the Existing System

- The system would be able to register families that need to treat with the doctor manually.
- Record treatment information manually.
- Most of the time appointment is home visiting which makes difficult carrying huge medical equipment.
- The appointment of the patient is manual.
- Prescription of the drug also given manually

## 2.4. Drawback of the Existing System

The existing system or this manual system has many drawbacks like;

- It takes time while searching patient information
- The economy of the existing system needs high costs.
- The paper-based recording cannot be claimed as secure.
- The paper-based patient management system is not flexible.
- The performance of the existing system provides a slow response time because it is difficult to access data from the stored data.
- Data redundancy problem.

## 2.5. Business Rules of the Existing System.

The existing business rule of the patient management system has its own rules between the doctor and the patient. This rule must be fulfilled by a patient to give necessary data to the

doctor. The patient must be face to face communication with the doctor. And also, users need a pen, paper, or manual to use the system.

# Chapter three

## 3. Proposed system

### 3.1. Introduction

This chapter discusses topics like how the current system is operating and how the proposed application is going to replace the existing system will be discussed. Generally, this application will reduce the cost, time, and effort of the users, and also it will provide an easy way to use it for the user.

Our android-based patient management system for a family doctor is a mobile application designed to minimize the problem of the current system which is described in the problem statement. The application is effective at the time of registering, updating, searching, and deleting.

The main aim of the proposed application is:

- To reduce human errors by providing user-friendly input and output capabilities and record keeping.
- To store data into his/her mobile so, it can be easily accessed and used at any time.
- Perform crude operations in the easiest way
- To help the patient in case the doctor is unreachable by providing a knowledge base system such as a health assistant chatbot.
- To provide security to data.
- Able to perform easy communication through the system by providing a chatting system.

### 3.2. The Advantages of the proposed Application

The proposed application provides fast response time because it is to access data from the stored data. While using an android-based patient management application, there will be areas in which cost will be reduced. Reduce costs of paper, reduce transportation costs, and reduce time. The mobile-based patient management application by itself is clear and in this application, there is no duplication of data through the new application so it is powerful to manage things around patient information. The processing of data is a high speed and shorthand for

## 3.3. Disadvantage of the proposed Application

It needs an internet connection because its database crud operation is firebase database and also chat system working with IBM Watson assistant cloud server which provides an API to train the expert system.

## 3.4. Functional Requirements

The functional requirements of the application are concerned with the major activities that the application has to perform. Functional requirements describe the functionality or service provided by the new system. The proposed application can perform the following tasks: -

- The application would be able to register member details.
- The application would be able to manage the necessary information of each member.
- The application would be able to view, update, and also would be able to manage the patient treatment information and patient's history.
- The application would be able to set a reminder to medicine taking time to patient.
- The application is going to be used to record treatment information about patients.
- The application would be able to notification or alarming for appointments.
- The application provides a health care assistant (knowledge baes system) which helps patient self-treatment.
- The application is also able to perform chatting and appointment announcement through messages between doctors and members.

## 3.5. Non-functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of the system, rather than specific behaviors of the system which are specified under functional requirements of the system. The following are considered as the non-functional requirements for the project we are going to develop.

### 3.5.1. User Interface and Human Factors

The user interface of this system is easy and not difficult because the feature of each interface is a clear and readable interface to use so that anyone with basic knowledge of smartphones can easily interact with the system.

### 3.5.2. Hardware Consideration (Android version, processor, memory)

The application should run under android version 4.2.2 to recent with its and kernel version. The application can run android version 4.2.2 and above.

### 3.5.3. Security Issues

Any information system must be secure to fulfill the three main components of system security which are confidentiality, integrity, and availability, therefore to make the system that is going to develop secure, an authentication mechanism is going to be employed which is giving the user username and password so that no one apart from the authenticated user can get access to the application.

Since this application is going to be developed for mobile devices it is considered physically secure because the only person who is going to use the device is the one who is the owner of that device.

### 3.5.4. Performance Consideration

The system should give the right response for the user as fast as possible. Since the system that is going to be developed is for mobile phones, it will not be complicated to use and it will not take much time to handle responses.

### 3.5.5. Error Handling and Validation

When a user interacts with the application errors may occur. To control this kind of inaccuracies the application will generate different user-friendly messages. To do this, most of the system execution buttons will be controlled according to the sequence which the user is expected to follow, or this can be done by generating different system responses to the input of the user.

### 3.5.6. Quality Issues

The system quality issues also are important non-functional requirements. The system should give fast and efficient service to users. Availability and reliability are the key issues of this requirement. We will use suitable software and hardware to develop the system, will able to achieve this requirement.

### 3.5.7. Backup and Recovery

We will use removable flashes and disks such as CD-ROM, DVD for backup and recovery mechanism. Because the risks of data loss might be happening due to many reasons such as computer viruses, sudden accident on data storages, data theft, or power fluctuation.

### 3.5.8. Physical Environment

The application will be installed on any tablet and android based smartphones.

### 3.5.9. Documentation

User documentation and technical documentation will be provided to let them maintain the system if any failure happens, also the process of the documentation will be documented to let any developer upgrade the system.

# Chapter Four:

## 4. System Analysis

This chapter deals with the system use case diagrams, the use case descriptions (scenarios), sequence diagrams, activity diagrams, and state diagrams. After identifying the actors and use cases, the use cases are developed and descriptions of use cases are stated. The sequence diagram is drawn based on the use cases, which are developed for the proposed system. The activity diagrams will represent activities.

### 4.1. System Modeling

#### 4.1.1. Use Case Model

Use Cases are used to model the requirements of a system. A Requirements Model specifies all the required functionality of the system. A Use Case specifies the necessary functionality from a particular perspective. The creation of Use Cases can lead to better communication between users such as Doctors and patients and the software developer as the requirements are analyzed in a structured manner.

A Use Case is a descriptive textual document that describes a certain way of using the system. At a high level (or with a very simple system) the use case could describe the behaviors of the system as a whole. Use Cases need to be associated with at least one Actor.

##### 4.1.1.1. Use Case Diagram

In software and system engineering, a use case is defined as a list of steps, typically defining interaction between a role (known in UML as an “actor”) and a system, to achieve a goal. The actor can be a human or an external system. In system engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.

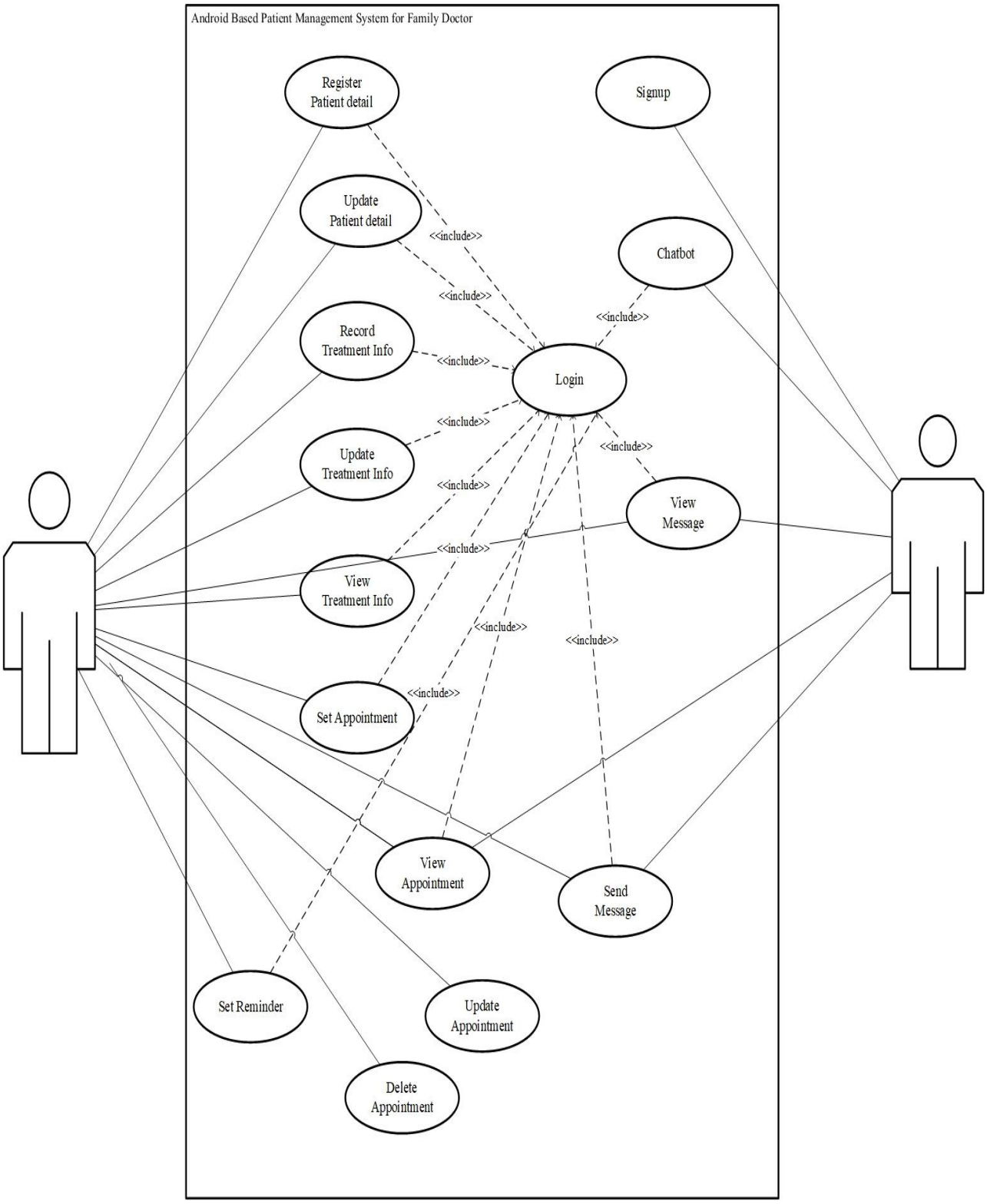



Figure 4.1 System use case diagram for Android-based patient management system

#### 4.1.1.2. Use Case Description

Table 4.1 A use case description for signup for a system

| | | |
|---|---|---|
| Name | Signup | |
| ID | UC#1 | |
| Description | It allows the customer to create an account | |
| Actor | Patient | |
| Precondition | Legitimate email is required | |
| Postcondition | User name and password created and login to home page | |
| Basic course of action: | **Actor action**<br>Step 1. Initiated when the user a create account button<br>**Step3.**The user fills his/her user's name, password, and confirm password. | **System response**<br>**Step2.** The system Displays create account Page.<br>**Step4.** Display created successfully<br>**Step5.** The use case ends. |
| An alternate course of action: | **A3.** The password and confirm password does not match.<br>**3.1.** The system displays the error message.<br>**3.2.** The system continues at step 3. | |

Table 4.2 A use case description of login use case

| | |
|---|---|
| Name | Login |
| ID | UC#2 |
| Description | It allows user to login into the system |
| Actor | Doctor and patient |
| Precondition | The doctor and customer should have an account |
| Post condition | The user will login in to his system home page |

Table 4.3 Description of registration of members

| | | |
|---|---|---|
| Name | Register Patient | |
| ID | UC#3 | |
| Description | It allows the doctor to register the necessary information of each members | |
| Actor | Doctor | |
| Precondition | The doctor must have login and family's the member registered | |
| Post condition | member information is recorded | |
| Basic course of action: | **Actor action** | **System response** |
| | **Step 1.** The doctor open member registration page form menu | **Step2.** The system display member registration form. |
| | **Step3.** The user fill the required information in the form | **Step5.** Display registration successful or failed. |
| | **Step4.** Press Register button | **Step6.** The use case ends. |
| Alternate course of action: | **B3.** The username/password is invalid.<br>**3.1.** The system displays error message.<br>**3.2.** The system continues at step 3. | |
| An alternate course of action: | **C3.** Invalid information entry<br>**3.1.** The system displays the error message.<br>**3.2.** The system continues at step 3. | |

Table 3. 4: Description of treatment use case

| | |
|---|---|
| Name | Record Treatment |
| ID | UC#4 |
| Description | It allows the doctor to register the necessary information of each Patient |
| Actor | Doctor |
| Precondition | The Patient first registered |
| Postcondition | Patient information is recorded |

| | | |
|---|---|---|
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open patient information registration page form menu<br>**Step3.**The user fills in the required information in the form<br>**Step4.** Press-Register button | **System response**<br>**Step2.** The system displays the patient registration form.<br>**Step5.** Display registration successful or failed.<br>**Step6.** The use case ends. |
| An alternate course of action: | **D3.** Invalid information entry<br>**3.1.** The system displays the error message.<br>**3.2.** The system continues at step. | |

Table 3.5.: Description of update member use case

| | | |
|---|---|---|
| Name | Update Patient details | |
| ID | UC#5 | |
| Description | It allows the doctor to update the member information | |
| Actor | Doctor | |
| Precondition | The doctor must log in and the family being updated is registered first | |
| Postcondition | Patient information is updated | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor opens the update form from the main page<br>**Step3.**Fill the update form<br>**Step4.** Click Update button | **System response**<br>**Step2.**The System Display the update form<br>**Step5.**The system Display Update Successful y<br>**Step6.** The use case ends |
| An alternative course of action | **E3** Invalid information entry.<br>**3.1.** The system displays the error message.<br>**3.2**. Go to **step 3** to fill again | |

Table 3.6. Description for update treatment use case

| | | |
|---|---|---|
| Name | Update Treatment | |
| ID | UC#6 | |
| Description | It allows the doctor to update the treatment's information | |
| Actor | Doctor | |
| Precondition | The doctor must login and the patient information being updated is registered first | |
| Post condition | treatment information is updated | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open update form from the main page<br>**Step3.** Fill the update form<br>**Step4.** Click Update button | **System response**<br>**Step2.**The System Display the update form<br>**Step5.**The system Display Update Successfully<br>**Step6.** The use case ends |
| An alternative course of action | **F3** Invalid information entry.<br>**3.1.** The system displays the error message.<br>**3.2**. Go to **step 3** to fill again | |

Table 3.7. Description for view information use case

| | |
|---|---|
| Name | View information |
| ID | UC#7 |
| Description | It allows the doctor to see the information of Patient |
| Actor | Doctor |
| Precondition | The information of those families and members first register |
| Postcondition | The doctor can see the information he/she needs |

Table 3.8. Description for appointments use case

| | | |
|---|---|---|
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open view page from main menu<br>**Step3.** the doctor presses Display button | **System response**<br>**Step2.**The System displays the view page.<br>**Step4.**The system displays the information we need.<br>**Step4.** The use case ends. |
| Name | ~~Appointment~~ | |
| ID | ~~UC#8~~ | |
| Description | It allows the doctor to record a ~~list of a~~ppointments | |
| Actor | ~~Doctor~~ | |
| Precondition | The doctor has to log in first | |
| Postcondition | Appointments are recorded | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor opens the appointment registration page<br>**Step3.**The user fills in the required information in the form | **System response**<br>**Step2.** The system displays the appointment registration form.<br>**Step4.** Display record successful or failed<br>**Step5.** The use case ends. |
| An alternate course of action: | **J3.** Invalid information entry<br>**3.1.** The system displays the error message.<br>**3.2.** The system continues at step 3 to fill in the correct information. | |

Table 3.9. Description for view appointments use case

| | | |
|---|---|---|
| Name | View appointments | |
| ID | UC#8 | |
| Description | It allows the doctor to see the list of appointments registered in the database | |
| Actor | Doctor | |
| Precondition | The appointments should be first registered in the database | |
| Post condition | The doctor can see the list of appointments | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open view page from main menu<br>**Step3.** the doctor presses Display button | **System response**<br>**Step2.**The System displays the view page.<br>**Step4.**The system displays the information we need.<br>**Step4.** The use case ends. |

Table 3.10 Description for update appointments use case

| Name | Update Appointment | |
|---|---|---|
| ID | UC#9 | |
| Description | It allows the doctor to update the list of appointment | |
| Actor | Doctor | |
| Precondition | The doctor first saves a list of appointments to the database | |
| Postcondition | The appointment's information is updated | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open update appointment page<br>**Step3.**Fill the update form and Click Update | **System response**<br>**Step2.**The System Display the update form<br>**Step4.**The system Display Update Successful y<br>**Step7.** The use case ends |
| An alternative course of action | **K3** Invalid information entry.<br>**3.1.** The system displays the error message.<br>**3.2**. Go to **step 3** to fill again | |

Table 3.11 Description for delete appointments use case

| Name | Delete Appointment |
|---|---|
| ID | UC#10 |
| Description | It allows the doctor to update a list of appointment |
| Actor | Doctor |
| Precondition | The doctor must log in and the appointment being deleted is registered first |
| Postcondition | Appointment is deleted |

Table 3.12. Description for set reminder use case

| | | |
|---|---|---|
| Name | ~~Set reminder~~ | |
| ID | ~~UC#11~~ | |
| Description | It allows user to set a reminder or alarm | |
| Actor | ~~Doctor~~ | |
| Precondition | Login to home page | |
| Postcondition | Set alarm | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor opens delete form from the main page<br>**Step3.** enter the ID of the appointment being deleted and press the delete button | **System response**<br>**Step2.**The System Display the delete form<br>**Step4.**The system Display delete Successful y<br>**Step7.** The use case ends |
| An alternative course of action: | **L3** Invalid ID entry.<br>**3.1.** The system displays the error message.<br>**3.2.** Go to **step 3** to fill again | |

| | | |
|---|---|---|
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open set reminder page<br>**Step3.** The user fill the required information in the form | **System response**<br>**Step2.** The system display form.<br>**Step4.** Display successfully saved or failed.<br>**Step5.** The use case ends. |
| Alternate course of action: | **M3.** Invalid information entry<br>**3.1.** The system displays error message.<br>**3.2.** The system continues at step 3. | |

Table 3.13 Description for send message use case

| | | |
|---|---|---|
| Name | Send message | |
| ID | UC#12 | |
| Description | It allows user to send a message | |
| Actor | Doctor and customer | |
| Precondition | Login to home page | |
| Postcondition | Send message | |
| Basic course of action: | **Actor action**<br>**Step 1.** The doctor open send notification page<br>**Step3.**The user fills in the required information in the form | **System response**<br>**Step2.** The system display form.<br>**Step4.** Display successfully sent or failed.<br>**Step5.** The use case ends. |

Table 3.14 Description for view message use case

| Name | view message | |
|---|---|---|
| ID | UC#13 | |
| Description | It allows user to send message | |
| Actor | Doctor and customer | |
| Precondition | Login to home page | |
| Post condition | view message | |
| Basic course of action: | **Actor action**<br>**Step 1.** The user open message page<br>**Step3.**The user fill the required information in the form | **System response**<br>**Step2.** The system display form.<br>**Step4.** Display successfully sent or failed.<br>**Step5.** The use case ends. |

Table 3.15 Description for a chat with system chatbot use case

| Name | Chatbot | |
|---|---|---|
| ID | UC#14 | |
| Description | It allows user to request help | |
| Actor | Customer | |
| Precondition | Login to home page | |
| Postcondition | view message | |
| Basic course of action: | **Actor action**<br>**Step 1.** The customer opens the chatbot page | **System response**<br>**Step2.** The system display form. |

| | | |
|---|---|---|
| | **Step3.**The user fills in the required information in the form | **Step4.** Display successfully sent and display response or failed.<br>**Step5.** The use case ends. |

#### 4.1.1.3. Use Case Scenarios

The following are described scenario of how the user use the systems to perform operations.

As we know the interaction between the system and the user there are different scenarios from these, we will discuss some parts of how the user uses the system. It tells who is using the system and what they are trying to accomplish.

Scenario name: login

Participant actor Doctor

Mr. Kebede is a doctor and as a user, if he wants to login into the system, first browse the home page and initiate the login page. Then after the login page is displayed fill in username and password and click the login button. The system checks the credentials and if the credentials are correct, he can sign in to his page.

After he logged in to his page, he can register members, update members, delete members, register treatment information, update treatment, view treatment, set appointment, view appointment, delete an appointment, set reminder, send a message, and view a message.

Scenario name: chat with system chatbot

Participant actor: patient(customer)

Mr. Yohannes is a patient and as user of the system if he wants help about his health condition in case the doctor got busy, first he signs in to the system and clicks health advisory help center which the system chatbot. By the time chatbot is initiated, it displays the first greeting to the user and the patient writes his request to the system. Then chatbot displays the requested solution to the user.

If the system does not recognize the requests, it displays corrections to rephrase the grammar.

### 4.1.2. Class Diagram

Class modeling is used to model the static structure of how the software will be built. In particular, class modeling shows classes, types, their internal structure, and their relationships. And also shows a collection of (static) declarative model elements, such as classes, interfaces, and their relationships, connected as a graph to each other and their contents.

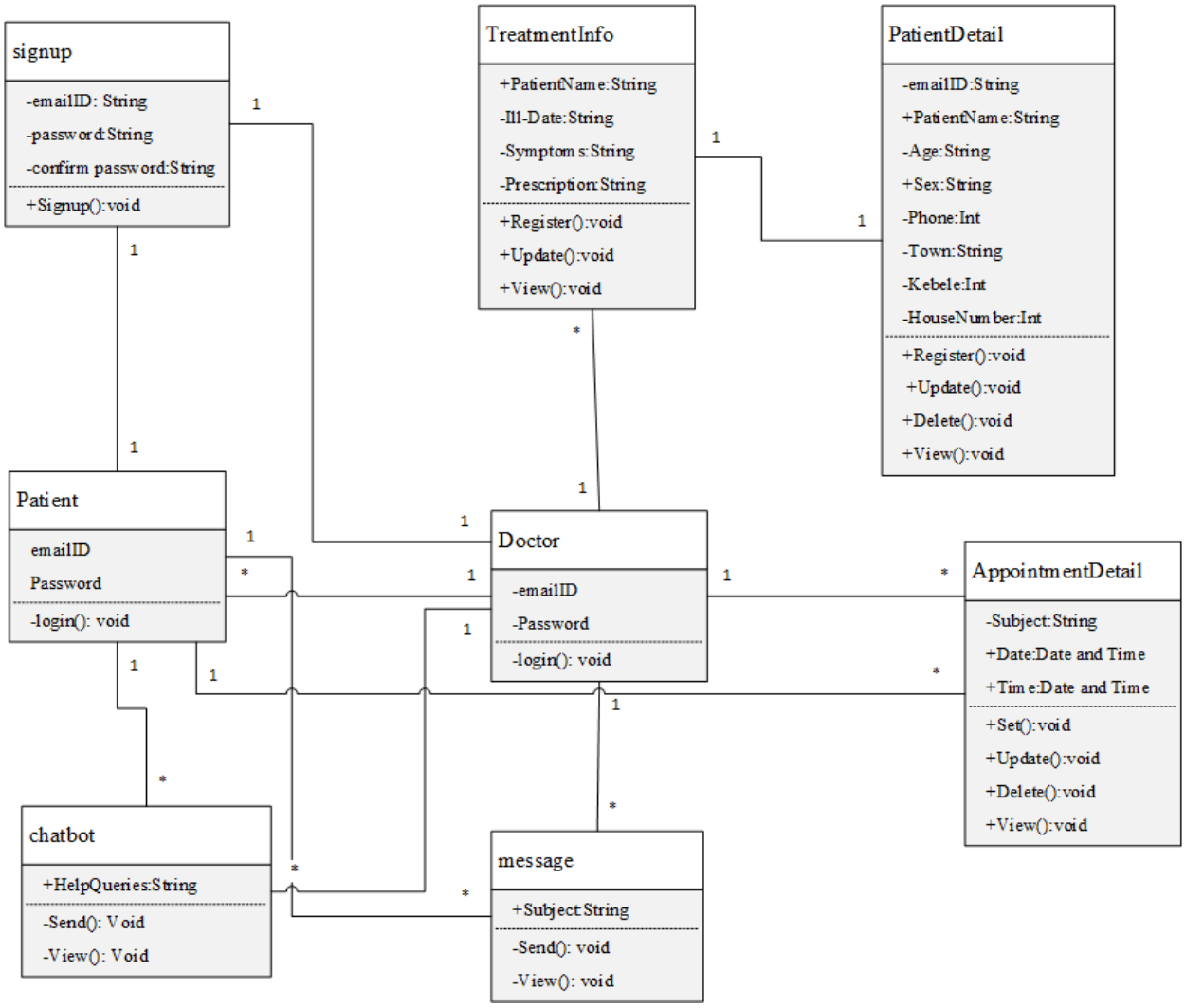



Figure 2. Class diagram for the proposed system.

### 4.1.3. Data Dictionary

Table 20. Data dictionary for signup table

| Field Name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| emailID | 50 | String | ….. | Email of patient |

| | | | | |
|---|---|---|---|---|
| Password | 25 | String | ….. | Password of patient |
| Confirm Password | 25 | String | ….. | Re password of patient |

Table 21. Data dictionary for Customer Table

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| emailID | 50 | String | … | Email of patient |
| Password | 25 | String | … | Password of patient |

Table 22 Data dictionary for Doctor table

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| emailID | 50 | String | … | Email of doctor |
| Password | 25 | String | … | Password of doctor |

Table 23. Data dictionary for TreatmentInfo

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| emailID | 25 | String | … | Email of patient |
| PatientName | 50 | String | … | Name of the patient |
| Ill-Date | 15 | Date and time picker | Mm/dd/yyyy | Beginning of the day patient became ill |
| Symptoms | 100 | String | …. | Symptoms of the case |
| Prescription | 100 | String | …. | The prescription of dosage |

| | | | | |
|---|---|---|---|---|
| | | | | instruction of medicine |

Table 24. Data dictionary of MemberDetails

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| emailID | 25 | String | …. | Email of the patient |
| PatientName | 50 | String | …. | Name of the patient |
| Age | 10 | Int | …. | Age of the patient |
| Sex | 5 | String | …. | Sex of the patient |
| Phone | 12 | Int | … | Phone number |
| Town | 15 | String | … | Living town of patient |
| Kebele | 15 | String | …. | Residence kebele of patient |
| HouseNumber | 10 | Int | … | House number of the patient |

Table 25. Data dictionary of AppointmentDetails

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| Appointment place | 20 | String | …. | Place where doctor and patient going to be met |
| Subject | 30 | String | … | Cause of appointment |
| Date | 15 | Date and time picker | Mm/dd/yyyy | Date of appointment to the patient |

| | | | | |
|---|---|---|---|---|
| Time | 5 | Date and time picker | hh/mm/ss | Time of appointment to the patient |

Table 26. Data dictionary of message

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| Subject | 1048576 | String | …. | Message content is written by a doctor or patient |

Table 27. Data dictionary of chatbot

| Field name | Size in byte | Data type | Format | Description |
|---|---|---|---|---|
| Subject | unlimited | String | …. | Help queries written by a patient |

## 4.2. Dynamic Model

The dynamic model represents the time-dependent aspects of a system. It is concerned with the temporal changes in the states of the objects in a system.

### 4.2.1. Sequence Diagram

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

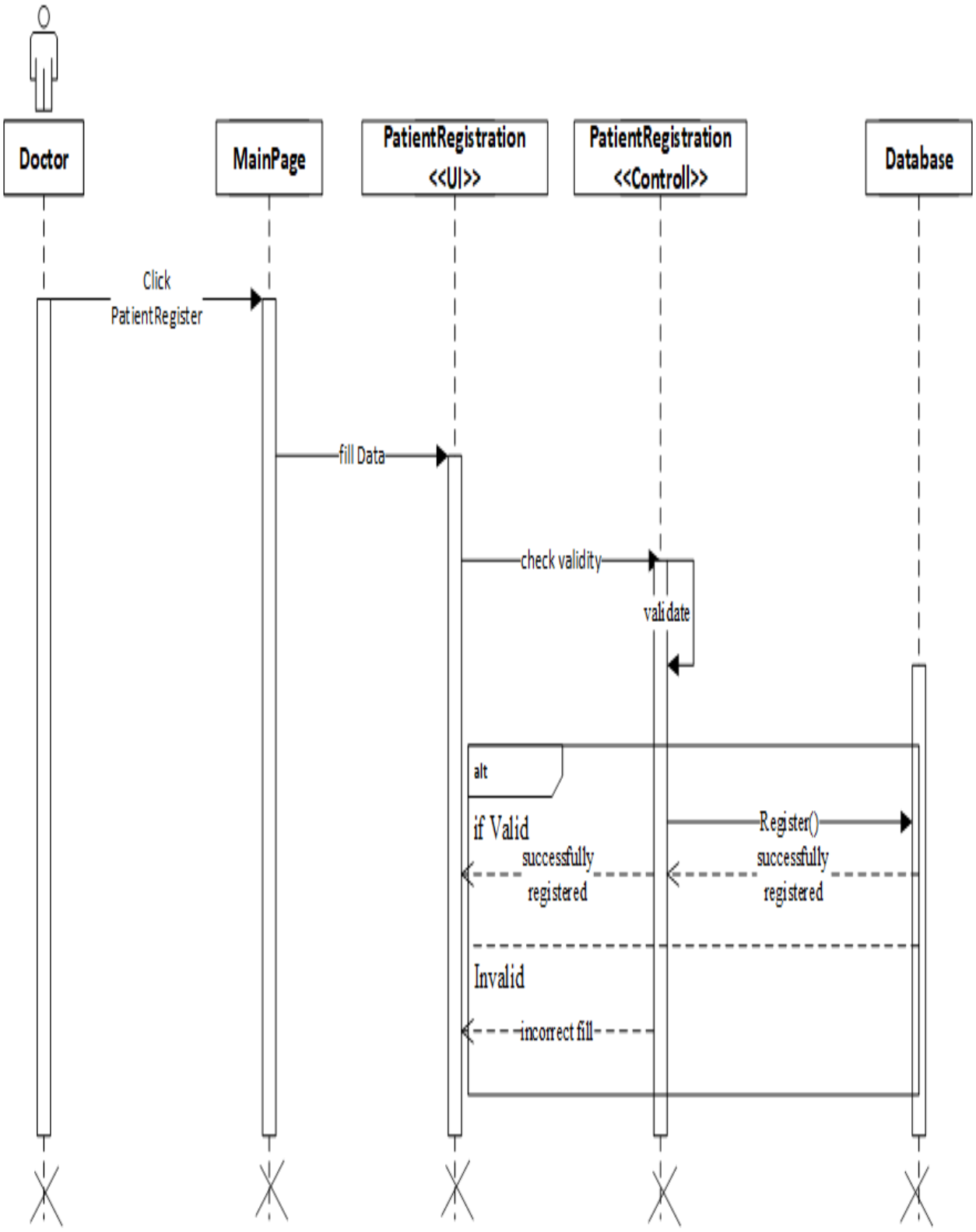



Figure 3. sequence diagram for register member details

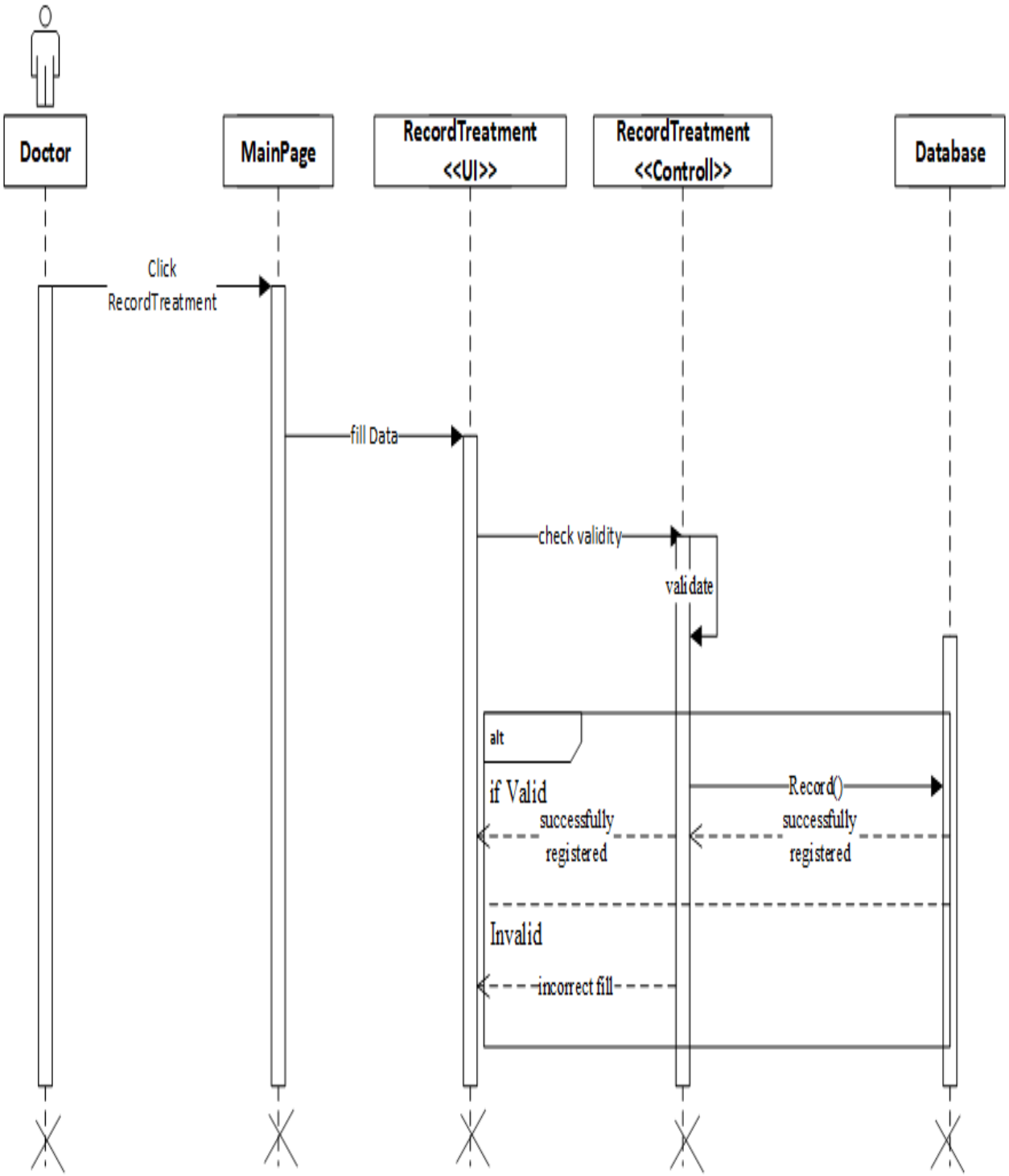



Figure 4. Sequence Diagram for register treatment information.

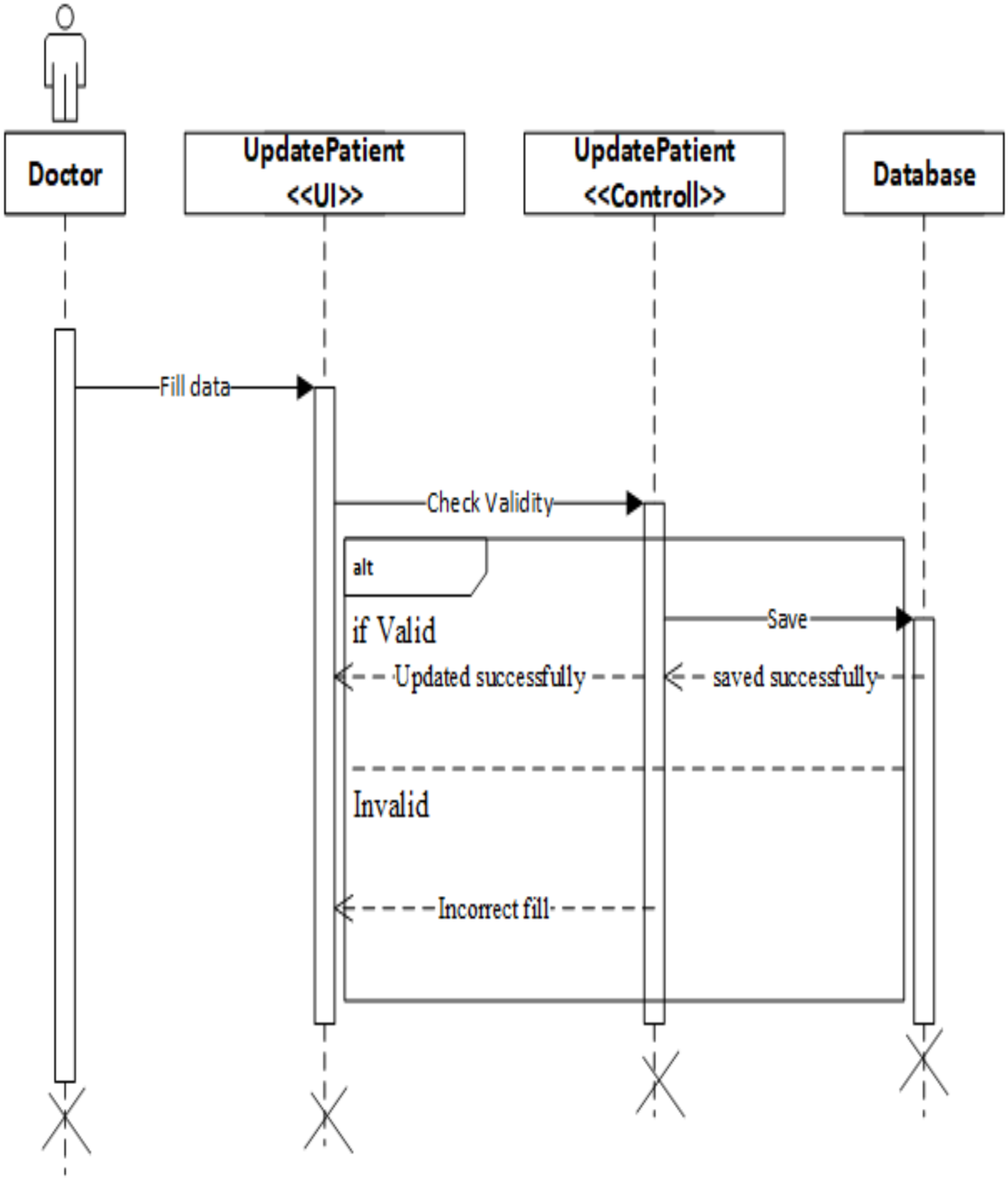



Figure 5. Sequence Diagram for update member information

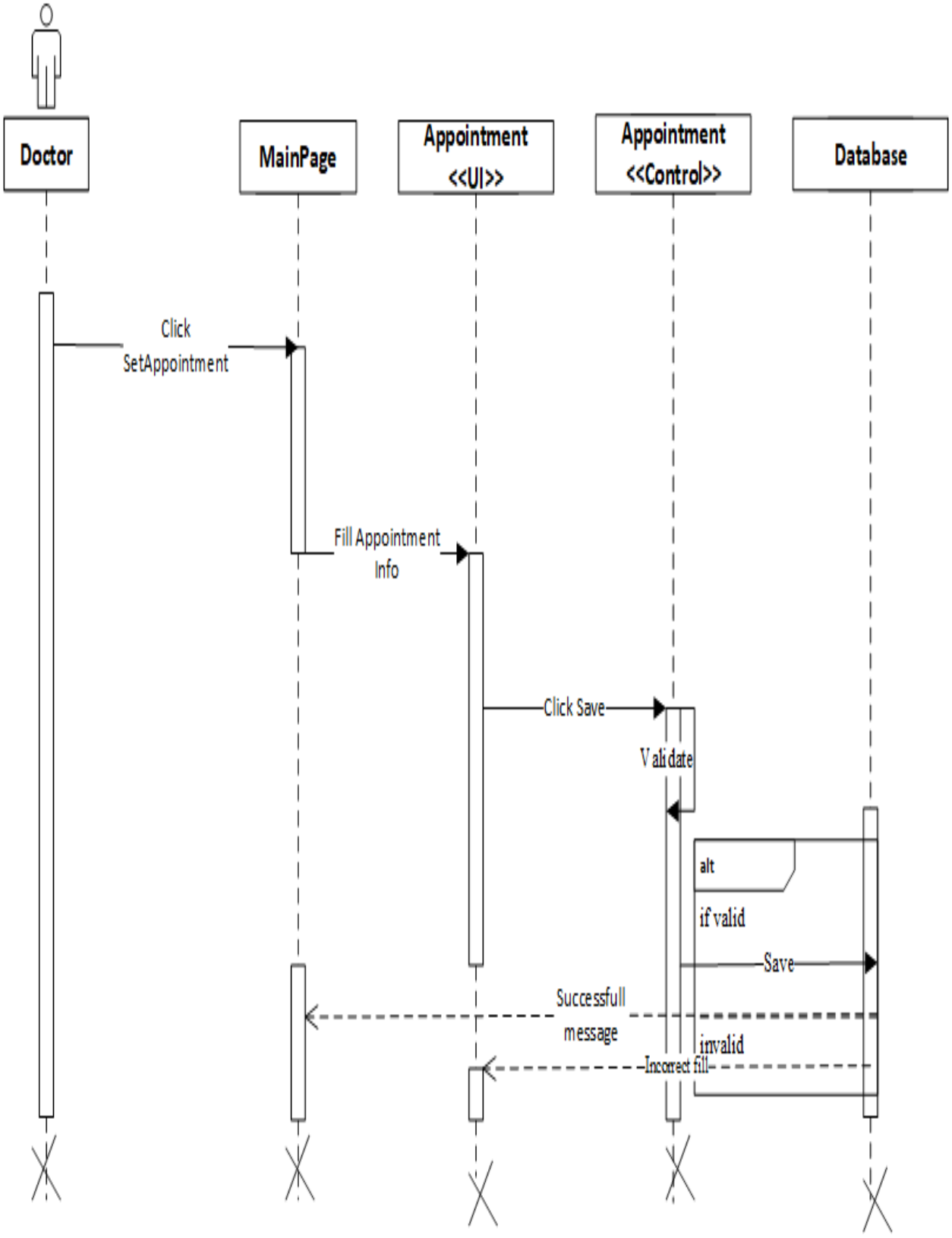



Figure 9. Sequence Diagram for Set Appointment.

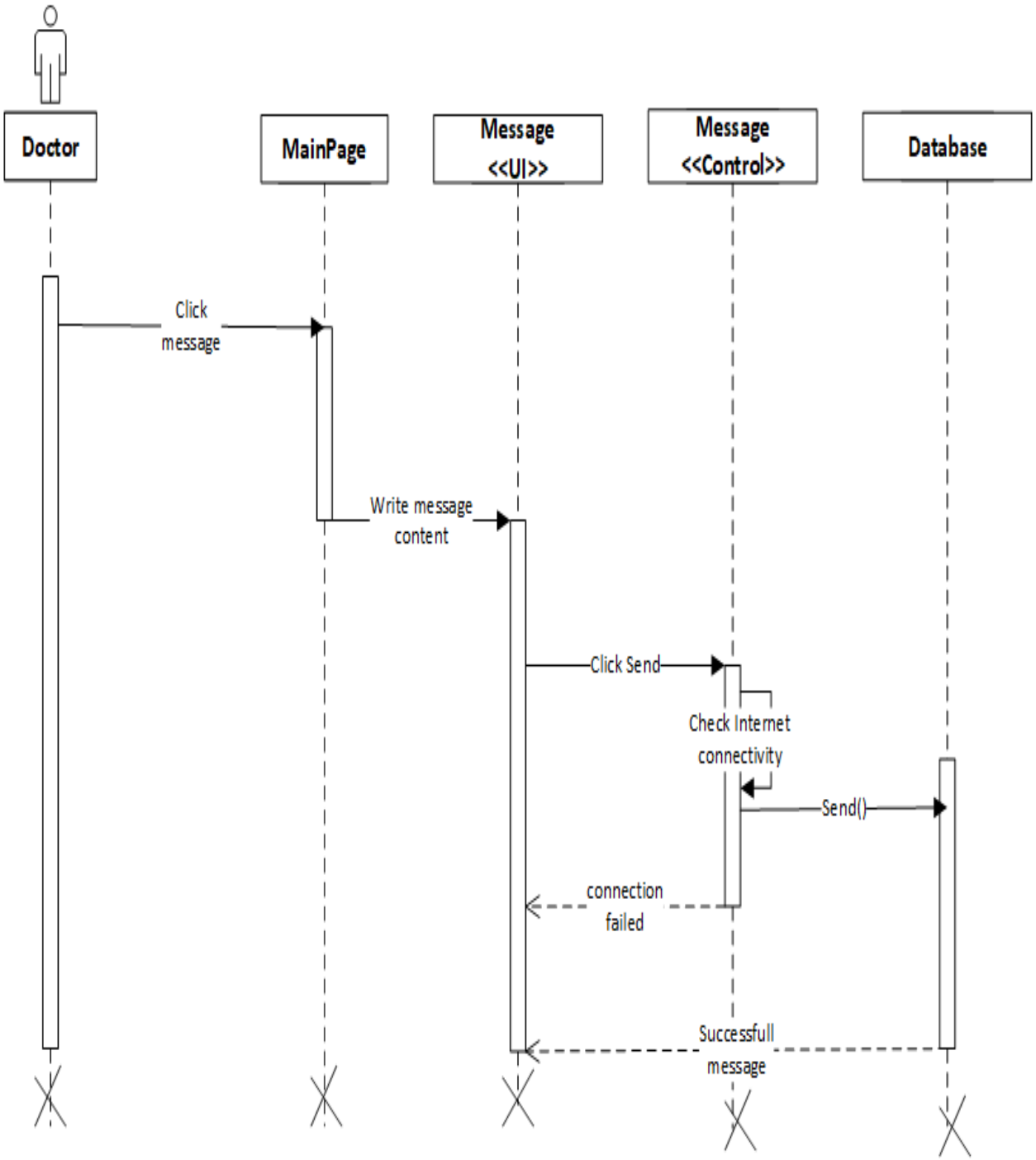



Figure 16. A sequence Diagram for message for doctor.

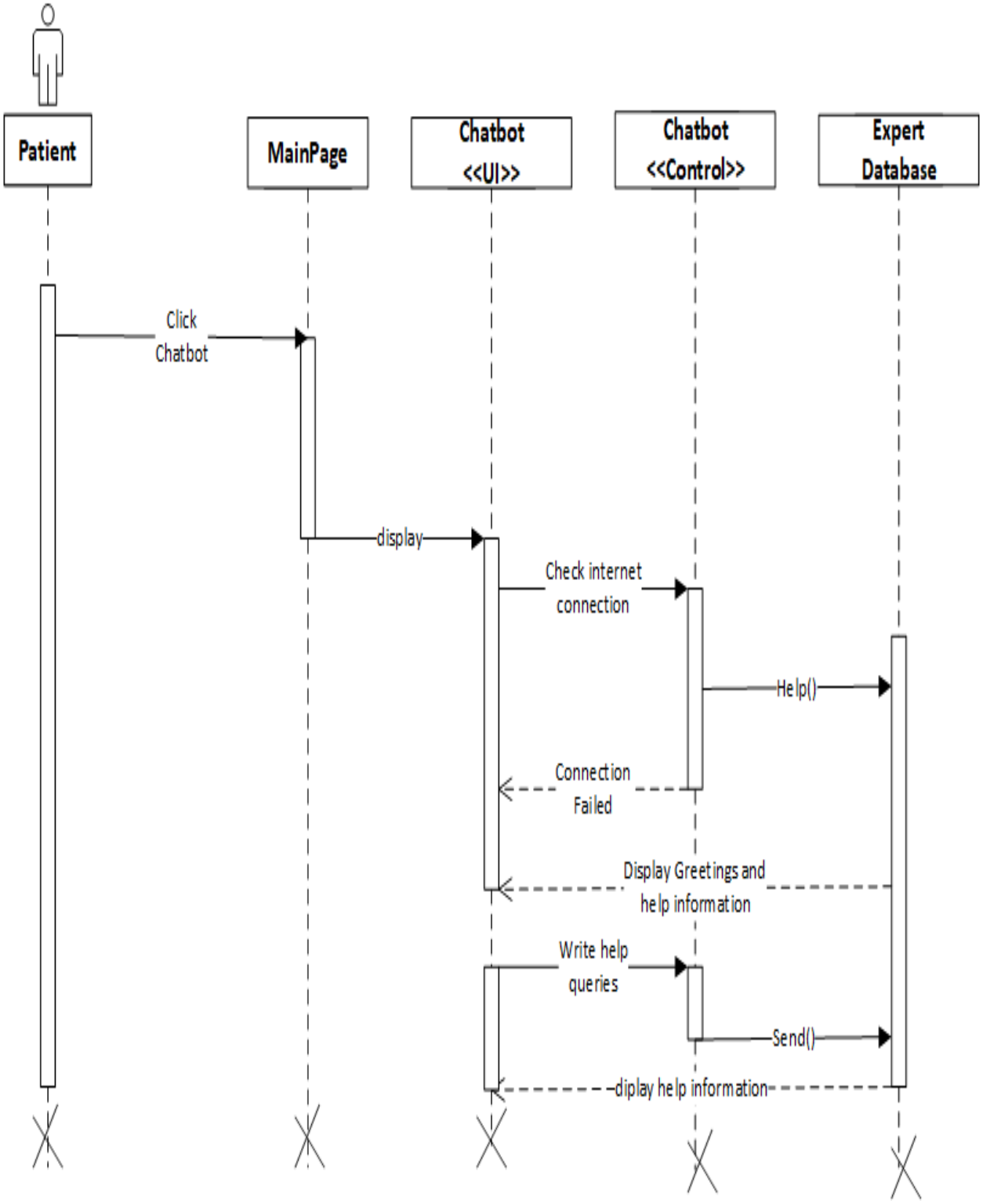



Figure 17. Sequence diagram for chatbot healthcare advisory.

### 4.2.2. Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the unified modeling language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. In its basic form an activity diagram is a simple and intuitive illustration of what happens in a workflow, what activities can be done in parallel, and whether there are alternative paths through the workflow.

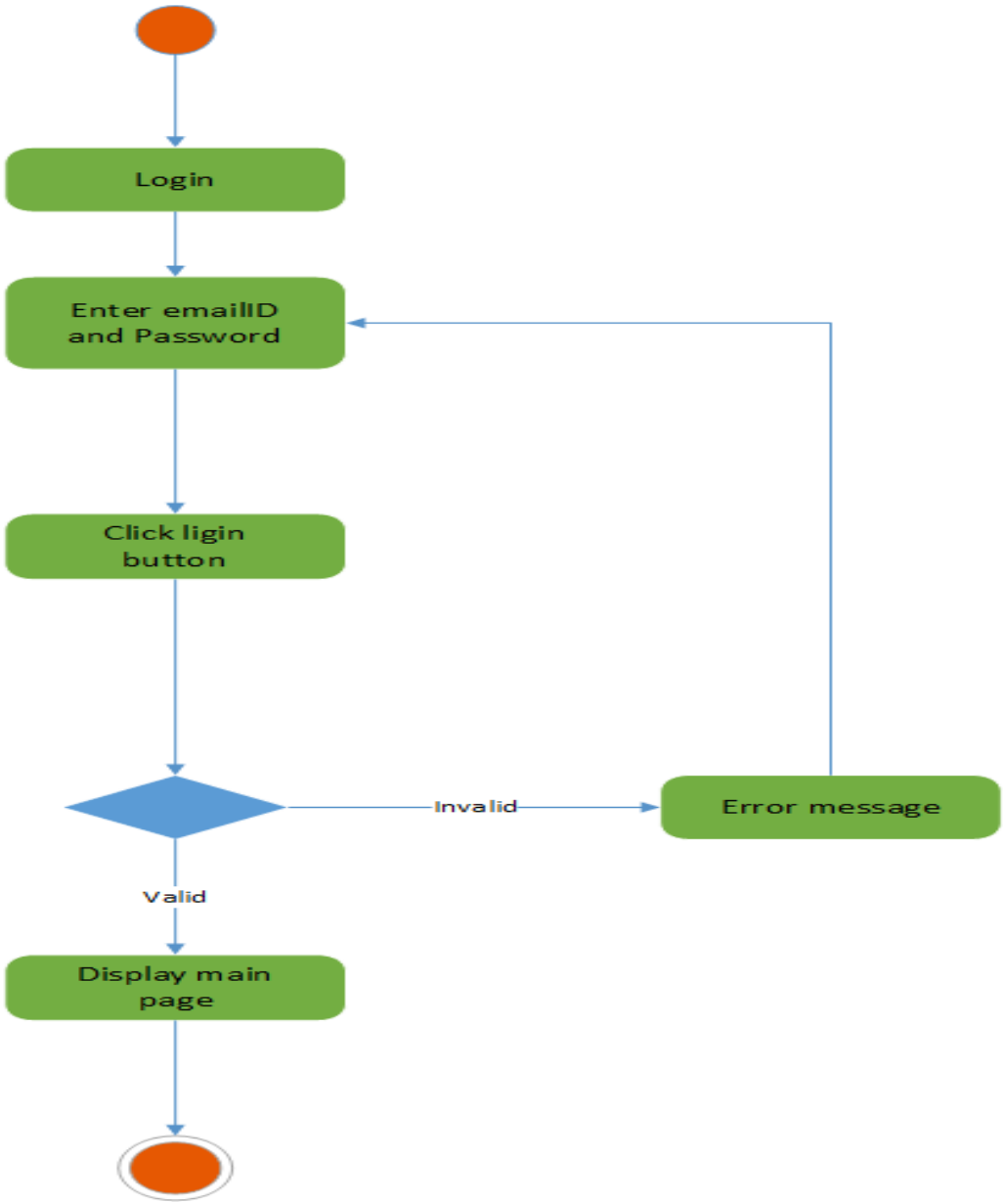


Figure 18. Activity Diagram for login

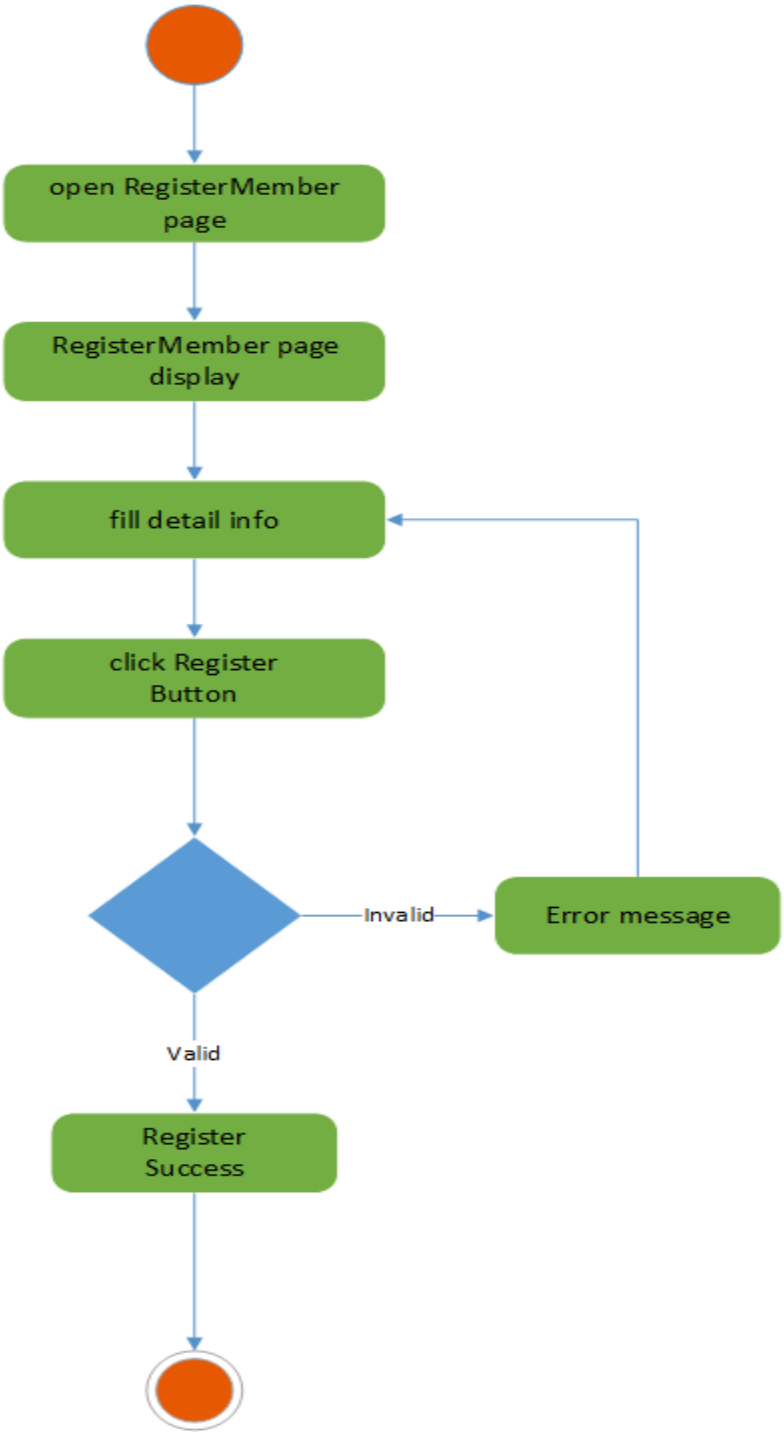


Figure 19. Activity Diagram for registration of member.

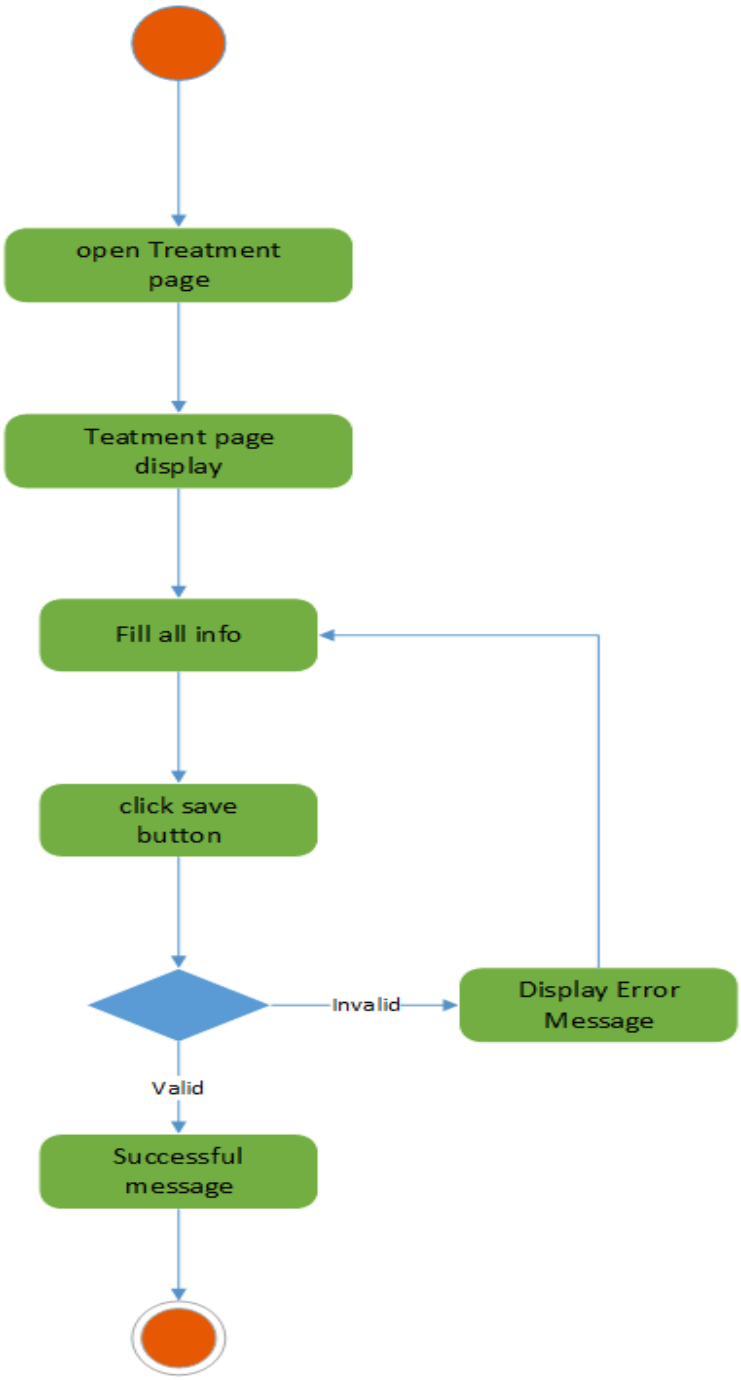



Figure 20. Activity Diagram for treatment record by the doctor.

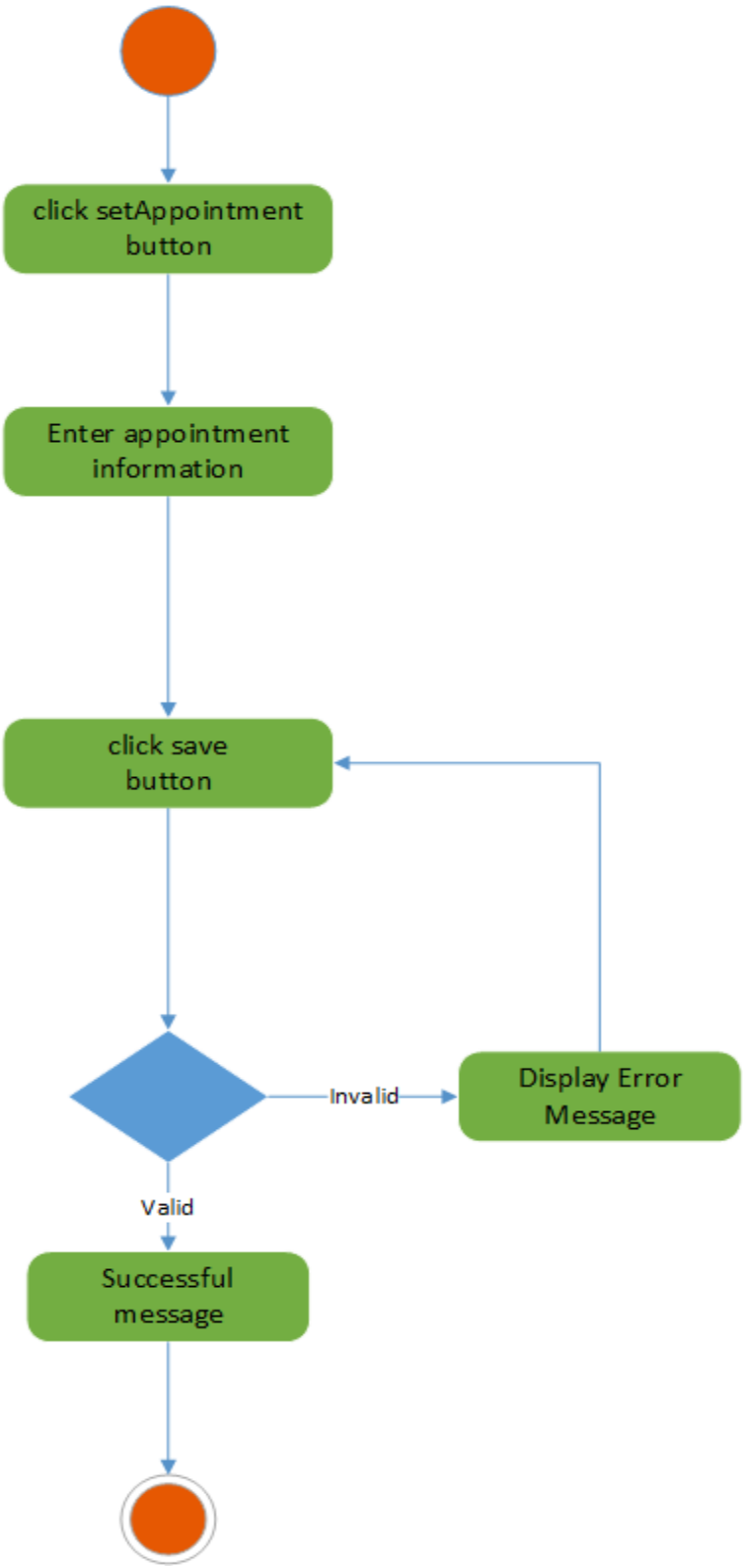



Figure 21. Activity Diagram for set Appointment.

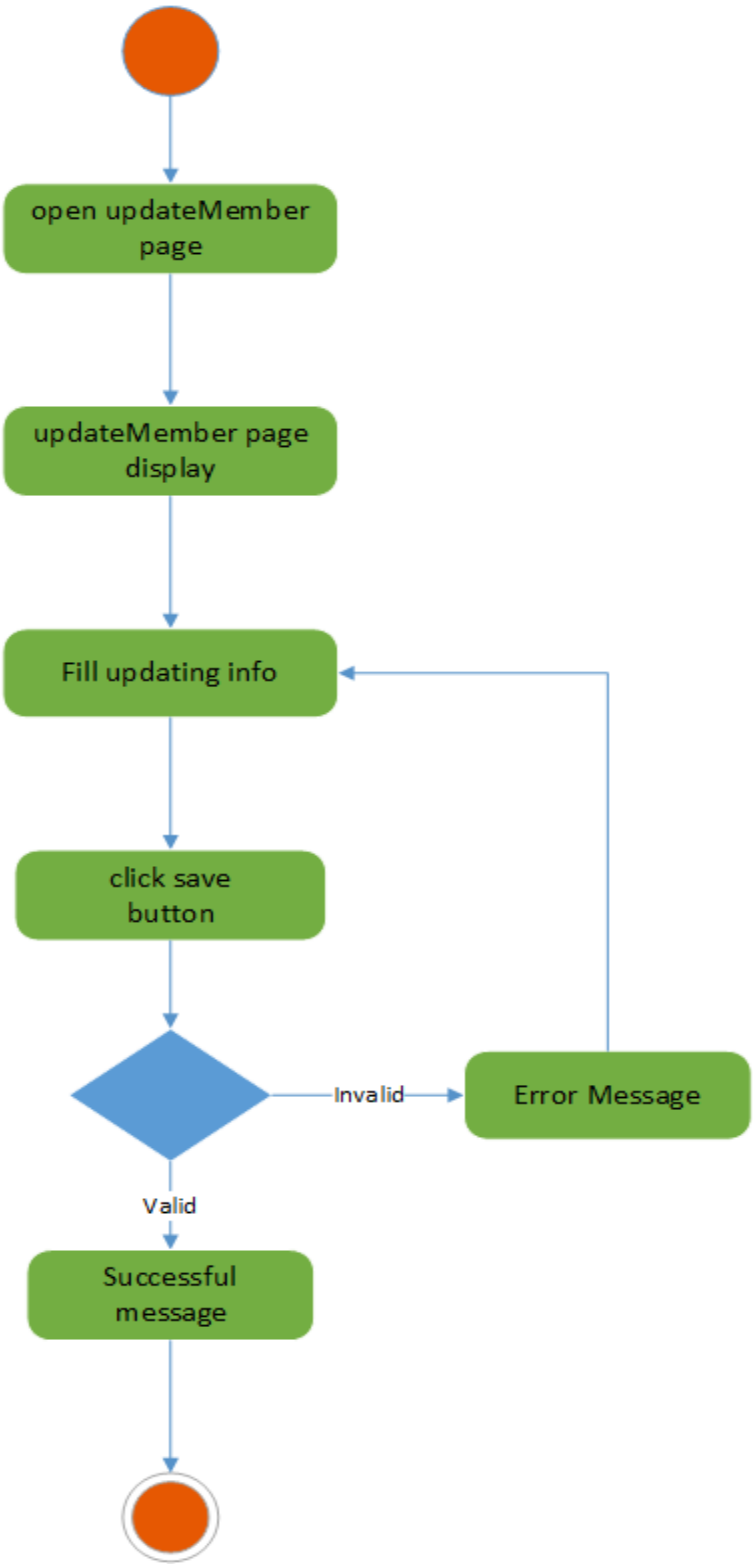



Figure 22. Activity Diagram for update member.

### 4.2.3. State chart Diagram

A state chart diagram is a view of a state machine that models the changing behavior of a state. State chart diagrams show the various states that an object goes through, as well as the events that cause a transition from one state to another.

The common model elements that state chart diagrams contain are:

- ✓ States
- ✓ Start and end states
- ✓ Transitions

A state represents a condition during the life of an object during which it satisfies some condition or waits for some event. Start and end states represent the beginning or ending of a process.

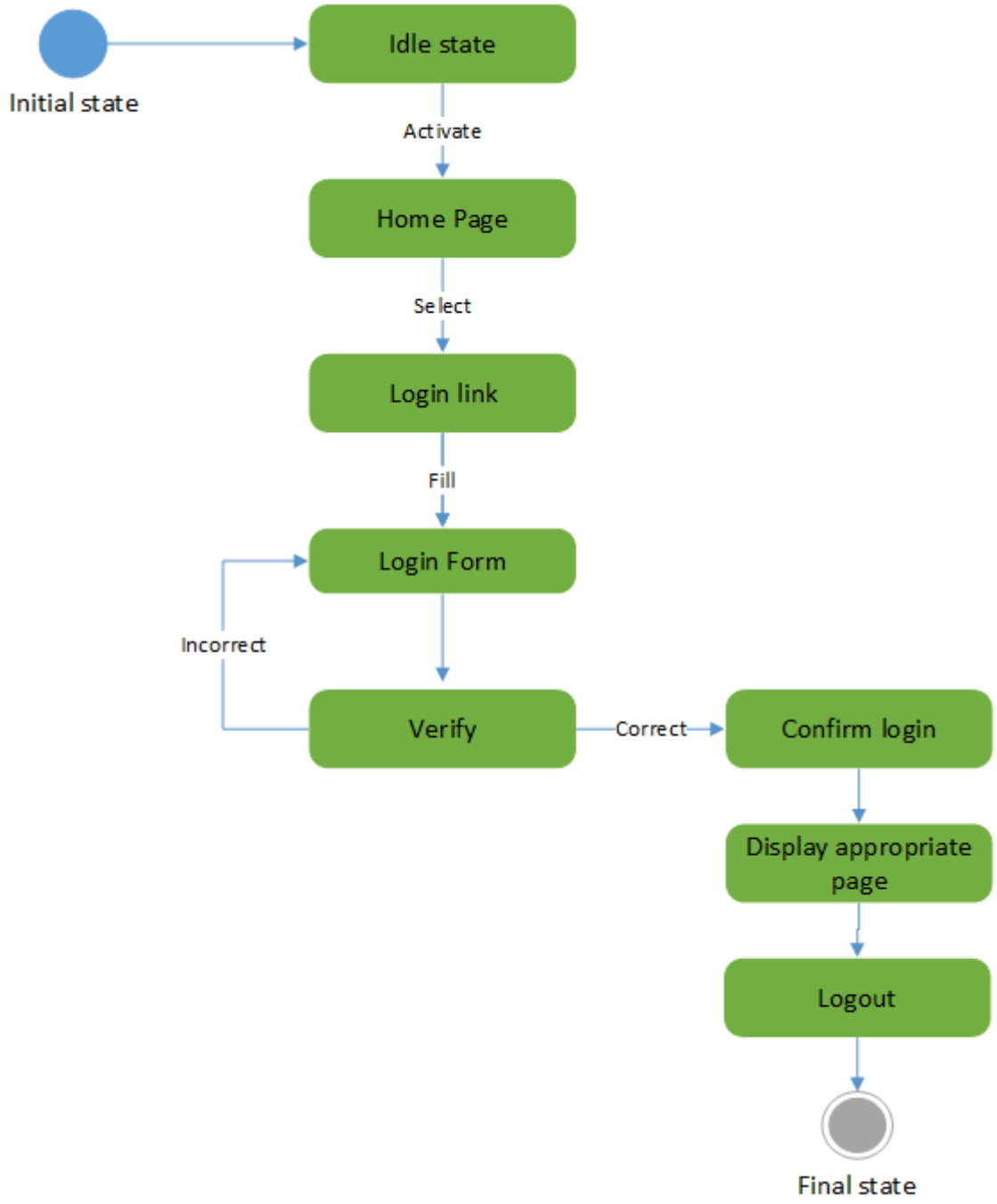



Figure 23. Statechart diagram for login.

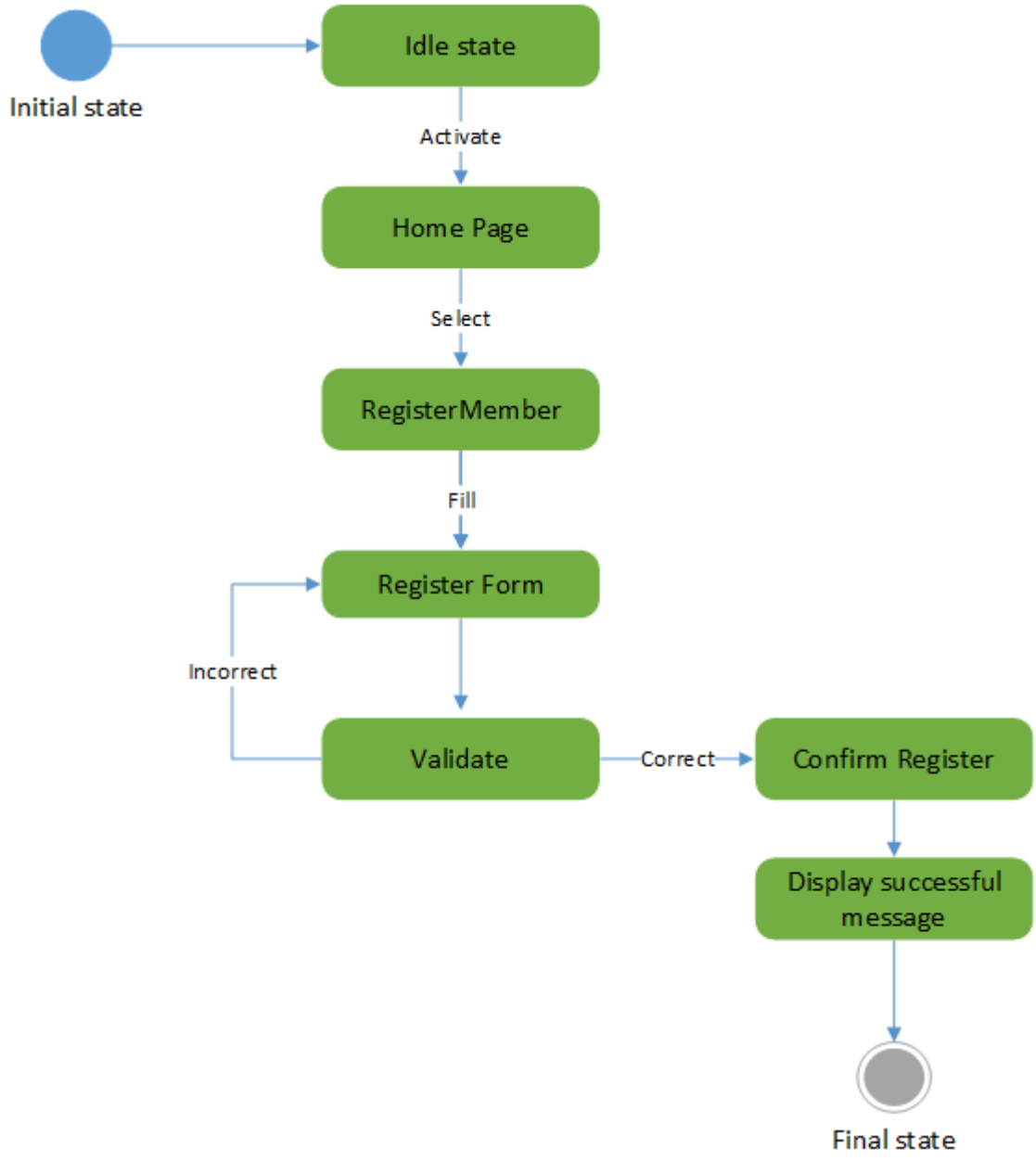



Figure 24. Statechart diagram for register member.

# CHAPTER FIVE

# 5. SYSTEM DESIGN

## 5.1. Introduction

In this section, the application is described by defining the design goals of the project, by decomposing the system into smaller subsystems that can be easily realized (proposed application architecture), and by selecting strategies for building the application, such as the hardware/software platform on which the application will run, the persistent data management strategy and the handling of boundary conditions. The result of system design is a clear description of each of these strategies, subsystem decomposition, and a deployment diagram representing the hardware/software mapping of the system.

## 5.2. Design Goals.

The design goals represent the desired qualities the system should have and provide a consistent set of criteria that would be taken into consideration when making design decisions. The following are mentioned as the design goals of the "Android-based patient management system for family doctors.

**Security**: the system should be secure to maintain data confidentiality. The system should authenticate its users by prompting them to enter their user's name and password to get access to the system.

**Extensibility**: the system should allow any additional services easily if needed, in other words, it should not be difficult to extend the system if additions are necessary.

**Availability**: the system should be available every time the user needs to access it.

**Usability**: the system should have a user-friendly user interface to allow the user to interact with the system easily.

**Portability:** the system should be able to run on any mobile that supports android mobile applications.

**Performance**: the main performance measure for a project is time, so the system should give fast responses to user requests.

## 5.3. Current Software Architecture

The existing system of Android-based patient recording system for family doctors is manual and hence there is no current software architecture that will be considered. As a result, we only describe the software architecture of the newly proposed system.

## 5.4. Proposed Software Architecture

The architecture which will be used for the proposed system is a three-tier architecture where the client or the user side is a mobile phone containing user interfaces like data entry interfaces, it is used to display information to the user. The user directly interacts with the system through the interfaces on this layer. The second layer is the synchronization layer which is the firebase.

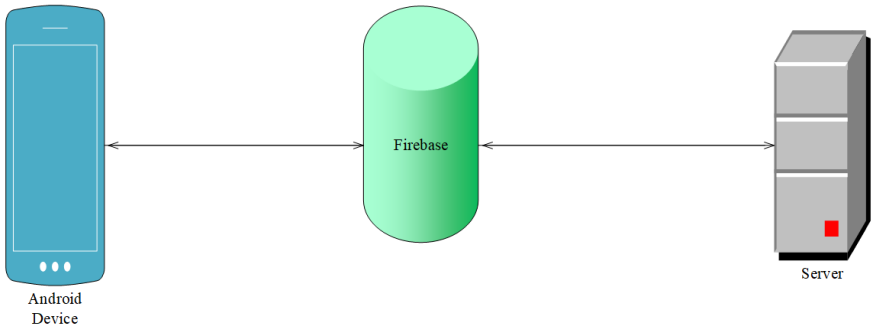



Figure 25. Proposed system architecture.

### 5.4.1. Subsystem Decomposition

A subsystem's decomposition is a general approach to solve a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. Subsystem decompositions will help reduce the complexity of the system. The subsystems can be considered as packages holding related classes or objects. The ABPMS is decomposed into the following subsystems.

- Signup subsystem
- Member Registration Subsystem
- Treatment Info Subsystem
- Appointment Subsystem
- Alarm set Subsystem
- Message Subsystem
- Chatbot Subsystem

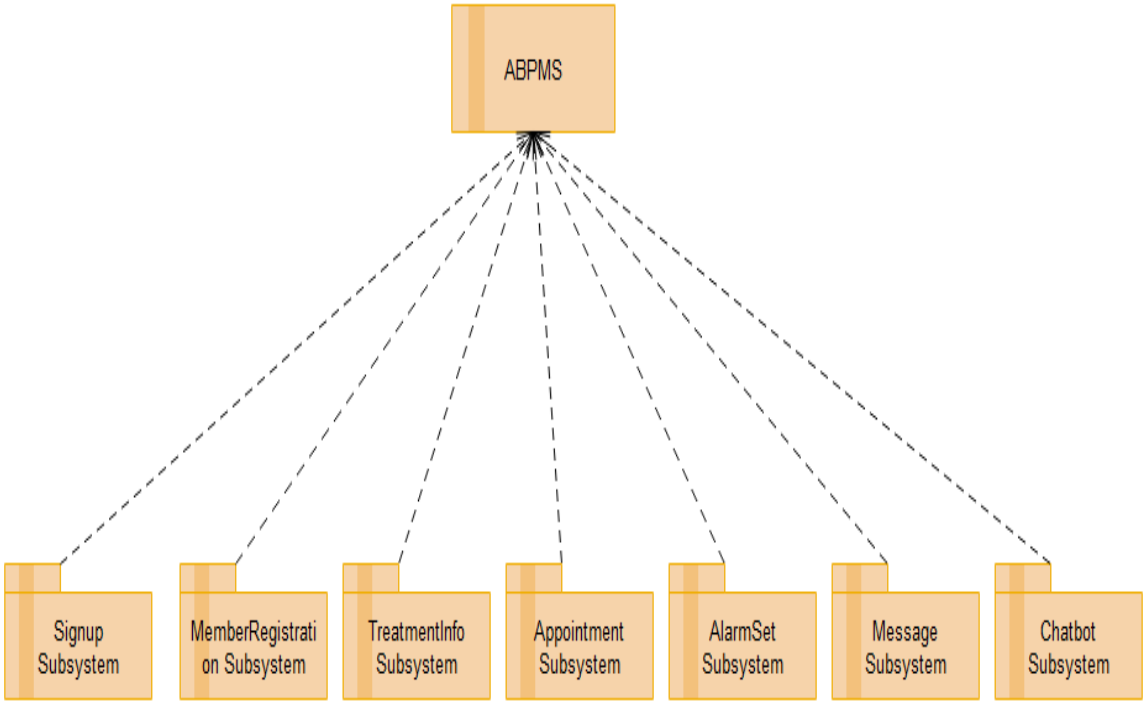



Figure 26. Subsystem decomposition of Android-based patient management for family doctors.

### 5.4.2. Component Diagram

Component diagrams show how the physical components of a system are organized. In this Diagram components of the system will be wired showing that there is a relation among components, management of the system, database, and operations performed on databases such as security issues. The diagram is simulated below.

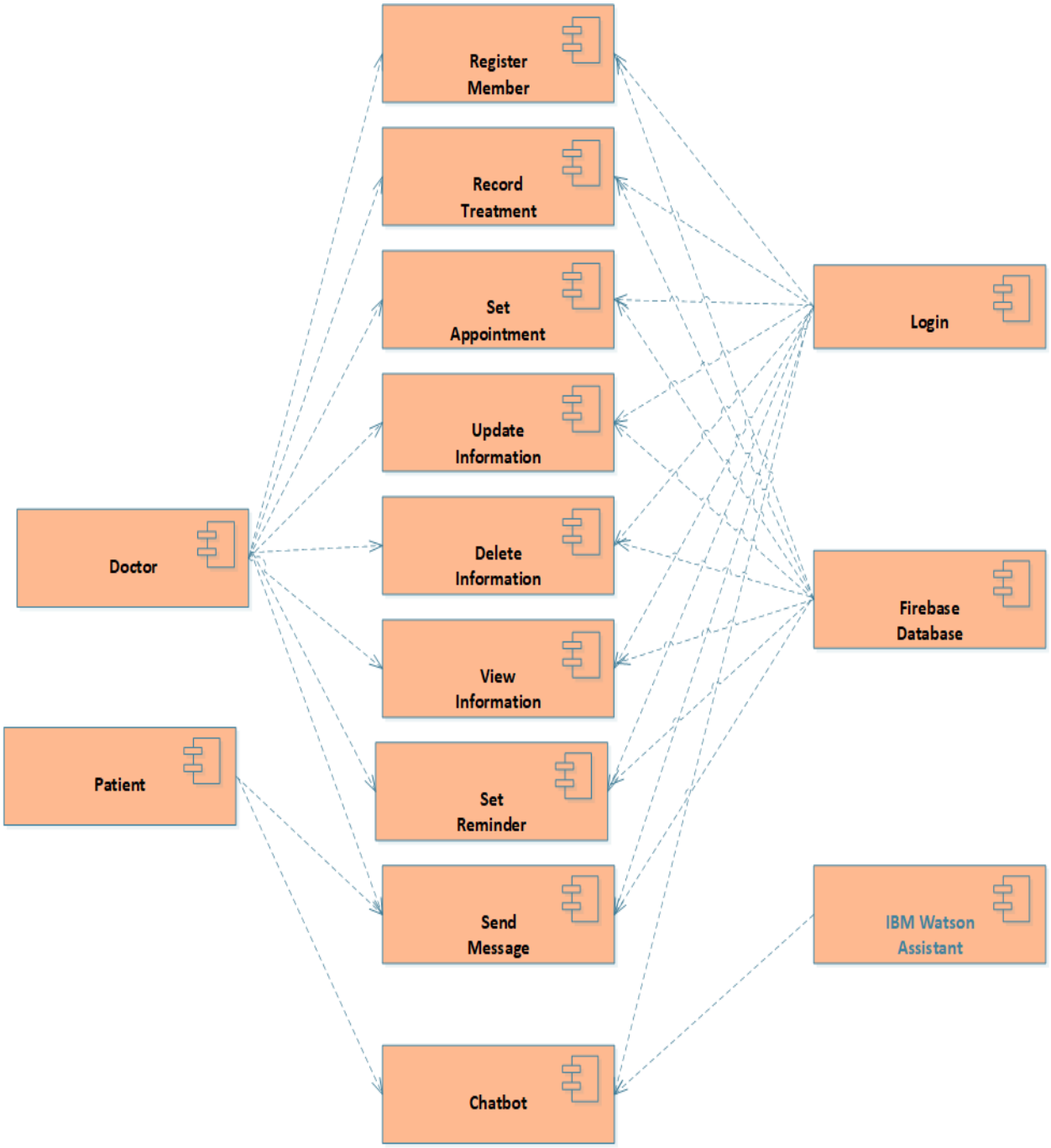



Figure 27. Component diagram of the system.

### 5.4.3. Deployment Diagram

**Android** is a software stack for mobile devices that includes an operating system, middleware, and key applications. Android relies on Linux OS for core system services such as security, memory management, process management, network stack, and driver model. The Linux kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

**Android applications** are written in Java. Android SDK tools compile and package the code along with any required data and resource files into the Android application **archive file** having a .apk suffix. The .apk file represents one Android application to be deployed to the Android-enabled mobile devices.

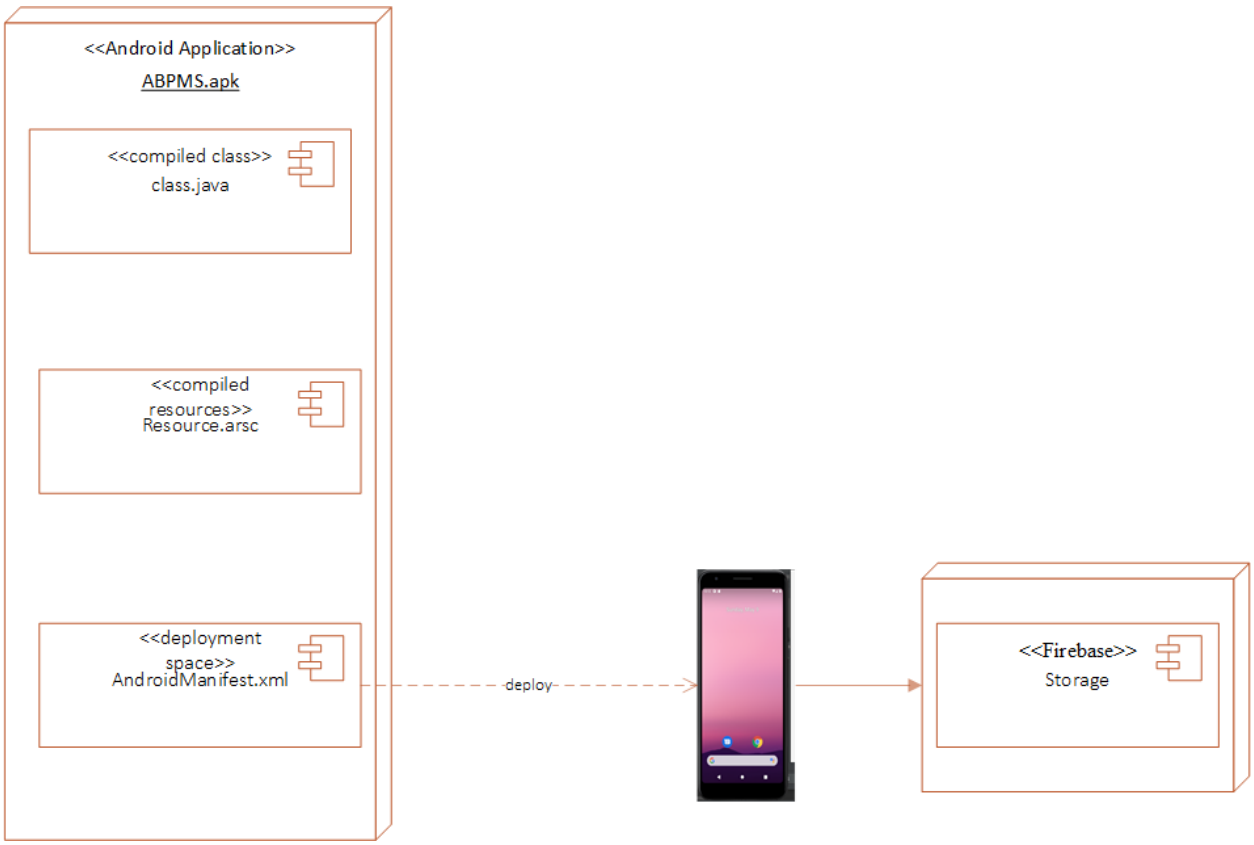



Figure 28. Deployment diagram.

### 5.4.4. Detailed Class Diagram

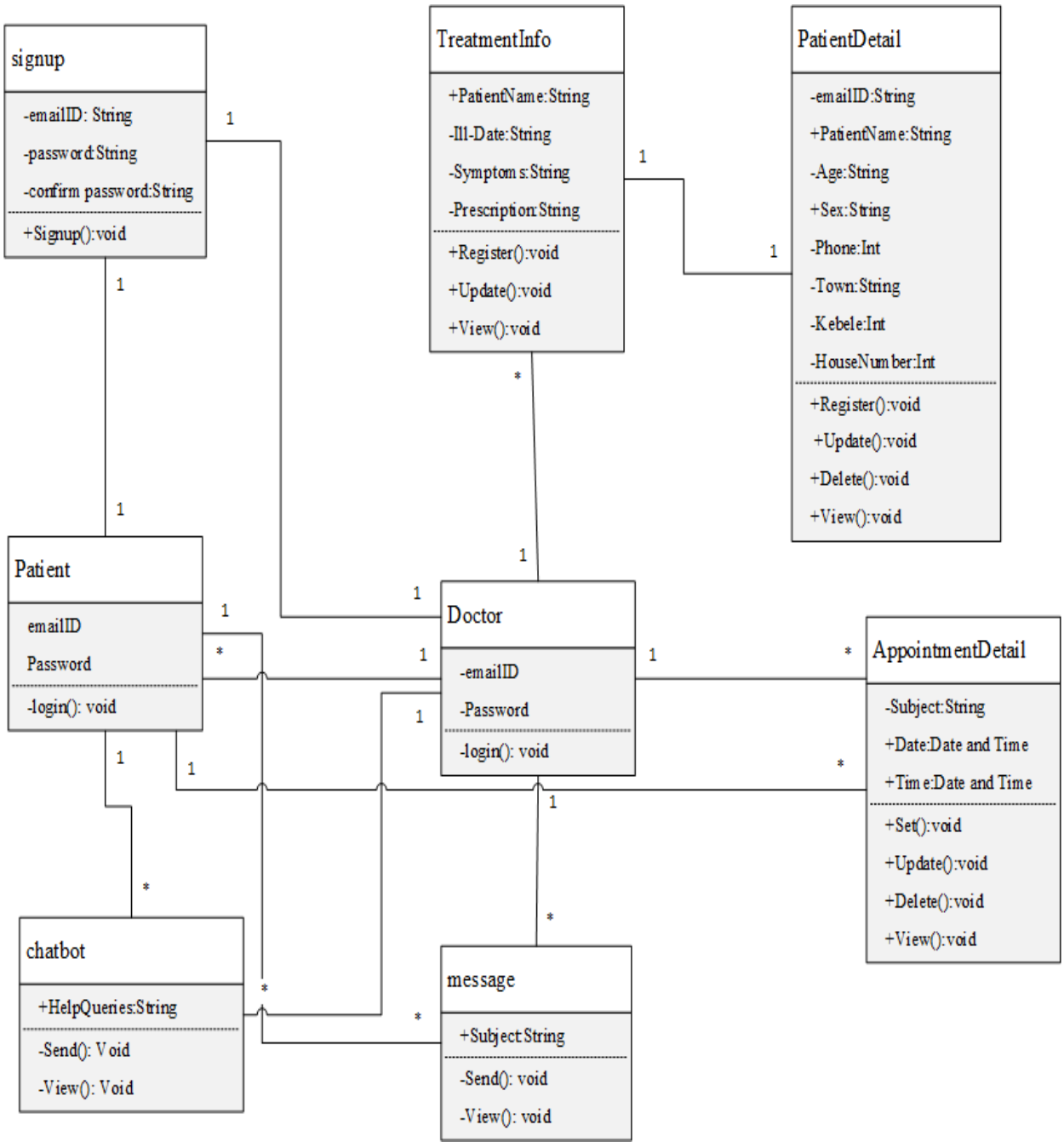



Figure 29. Detailed class Diagram for the proposed system.

### 5.4.5. Persistent Data Management

Persistence modeling is used to communicate the design of the database, usually to the developers. It is also used to describe the persistence data aspect of the system.

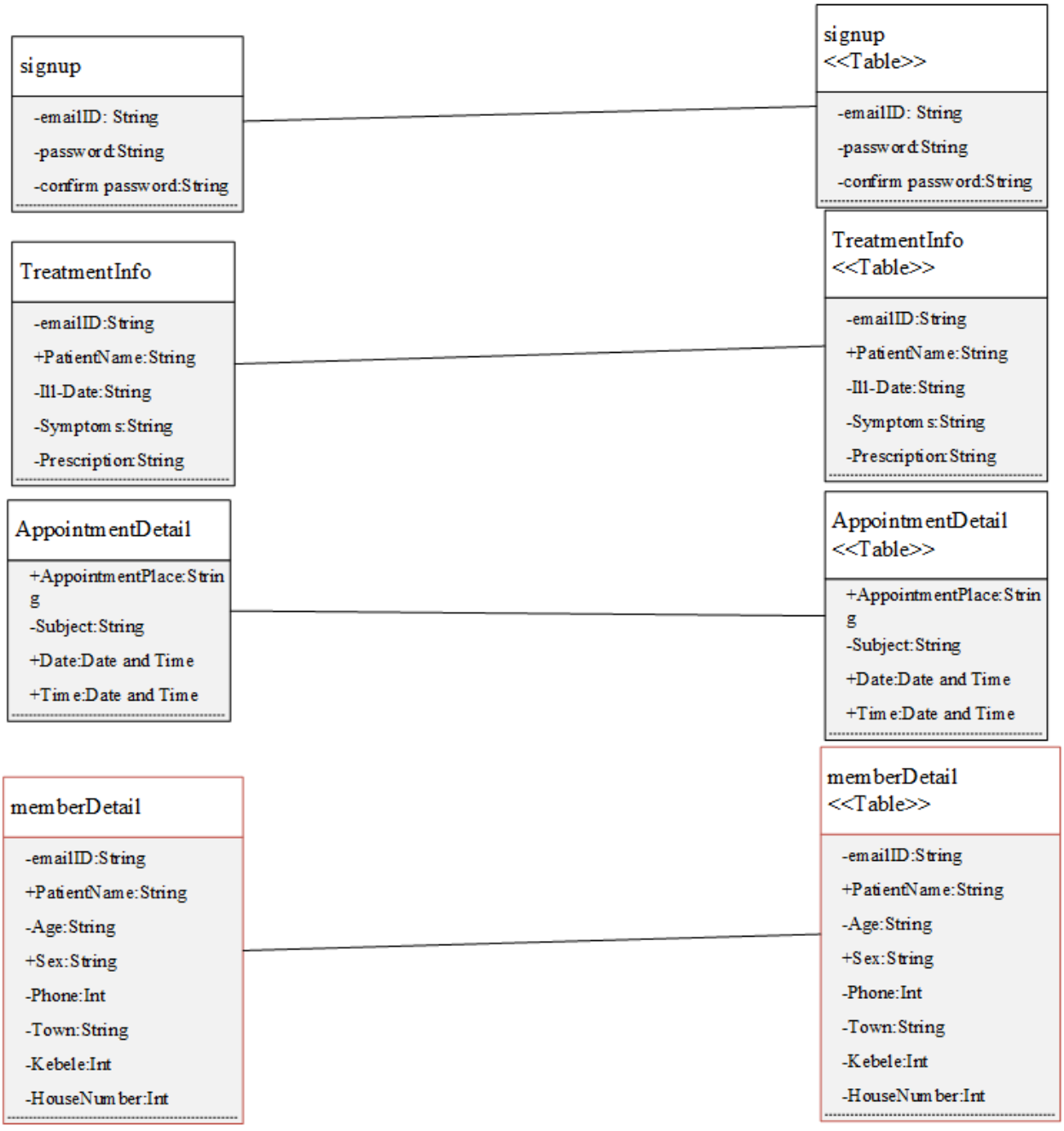



Figure 30. Persistent data management.

### 5.4.6. Access Control and Security

This new application has two users' access to different functionality and data since the application has only two actors. The access control is shown below by using an access control matrix in the following table.

Table 27 access control and security.

| Actor | Privilege |
|---|---|
| Doctor | Register();<br>Update();<br>Delete();<br>View();<br>setAlarm(); |
| Patient | Send()<br>View()<br>Signup() |

## 5.5. Package

A package diagram is a UML structure diagram that shows packages and dependencies between the packages. Package diagram enables you to gain a high-level understanding of the collaboration among model elements through analyzing the relationships among their parent package. This also helps explain the system's architecture from a broad view.

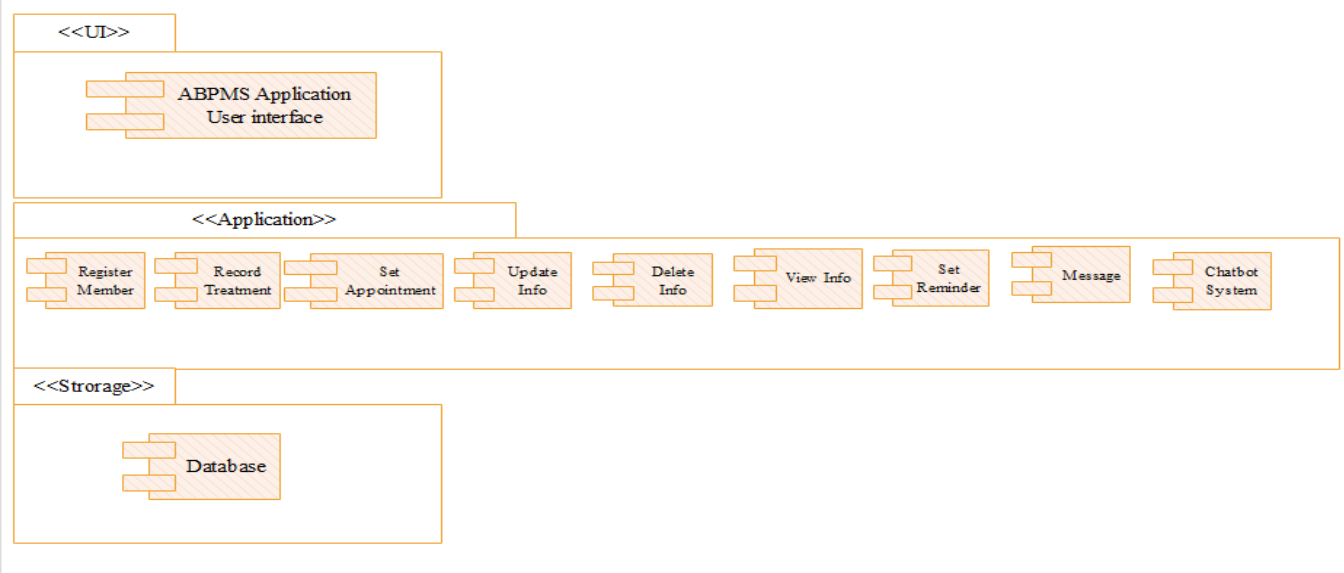



Figure 31. UML package diagram of the system.

## 5.6. Algorithm Design

Algorithm for login

//define credential variables

TextView tbtn;

EditText lEmail, lPass;

Button loginButton;

private FirebaseAuth lAauth;

private ProgressDialog mLoadingBar;

//initialize variables from XML file

protected void onCreate(Bundle savedInstanceState)

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_login);

lEmail = findViewById(R.id.logEmail);

lPass = findViewById(R.id.logPass);

loginButton = findViewById(R.id.logButton);

```
        tbtn = findViewById(R.id.noAccount);
        lAauth = FirebaseAuth.getInstance();
        mLoadingBar = new ProgressDialog(LoginActivity.this);
//implement the listener method for login
loginButton.setOnClickListener(new View.OnClickListener()
@Override
            public void onClick(View v)
//call method to implement credentials
checkCredentials();
//implement the method
            private void checkCredentials()
//convert the inputs received from the layout to strings
                String email = lEmail.getText().toString();
                String password = lPass.getText().toString();
//validate credentials
                if(email.isEmpty() || !email.contains("@"))
                {
                   showError(lEmail, "Email is not Valid");
                }
                else if(password.isEmpty()|| password.length()<7)
                {
                   showError(lPass, "Password must be 7 character");
                }
                else
```

```
{
    mLoadingBar.setTitle("Login");
    mLoadingBar.setMessage("Please wait while checking credentials!");
    mLoadingBar.setCanceledOnTouchOutside(false);
    mLoadingBar.show();

    //login
    lAauth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful())
        {
          mLoadingBar.dismiss();
          Intent = new Intent(LoginActivity.this, MainActivity.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
          startActivity(intent   }}
```

## 5.7. User Interface Design

User interface design is the overall process of designing how a user will be able to interact with an application.

User interface design is the specification of communication/interaction mechanism between the application users and a mobile. The process involves input mechanism design, output mechanism design, and navigation mechanism.

I. **A navigation mechanism** is part of the user interface that takes the user form one part of the application to the other user layout application. That includes buttons, toolboxes, icons, dialog boxes, etc. all controls should be clear and standardized.

II. **Input design** is about designing a form and its controls for a GUI system. That includes EditText, Spainer, Textarea, RadioBoton, CheckBox, etc...

III. **Output design** is about designing reports like detailed and summarized of our proposed application. e.g., Listview which can be displays details about the settled appointment.

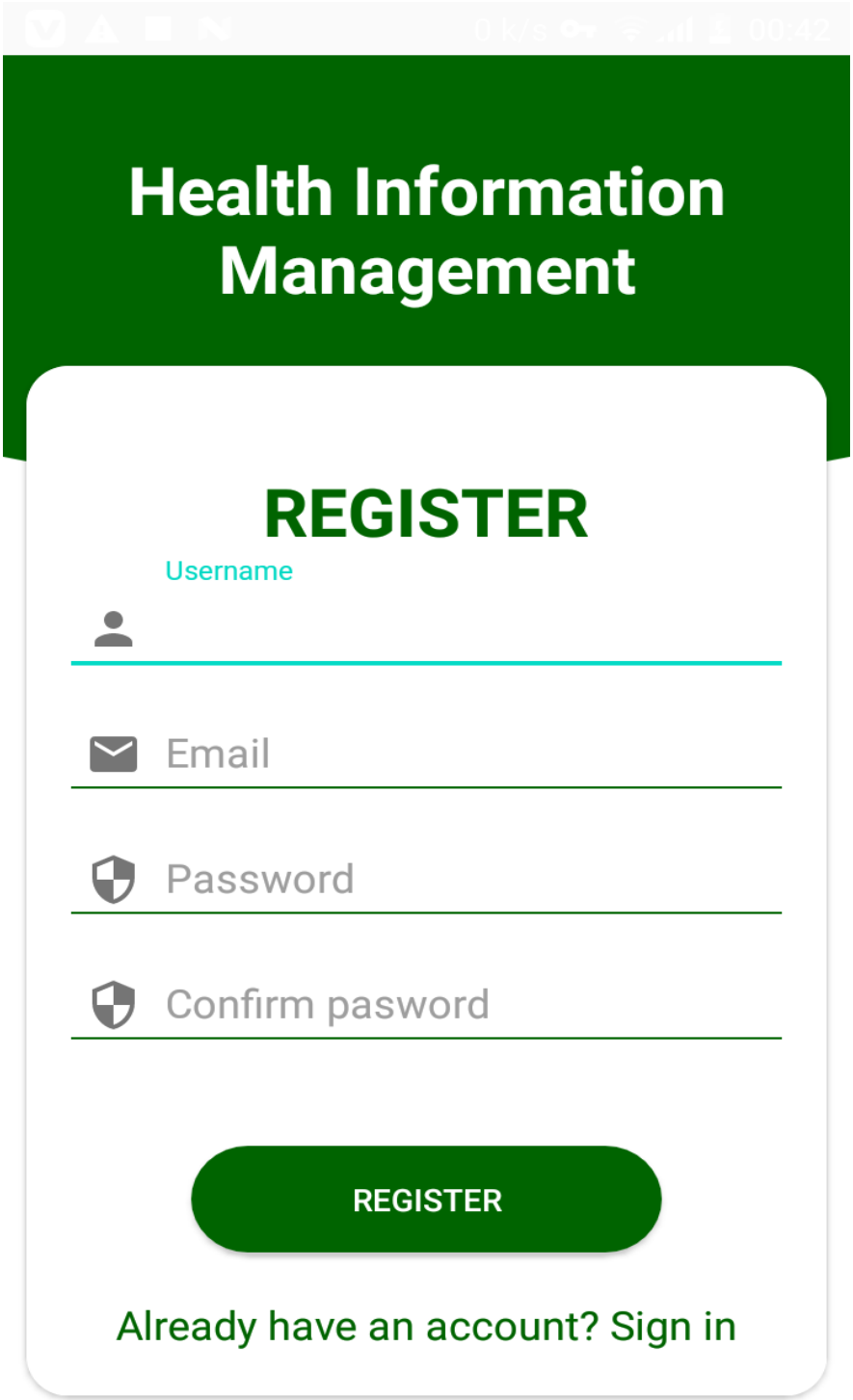



Figure 32. Patient signup user interface.

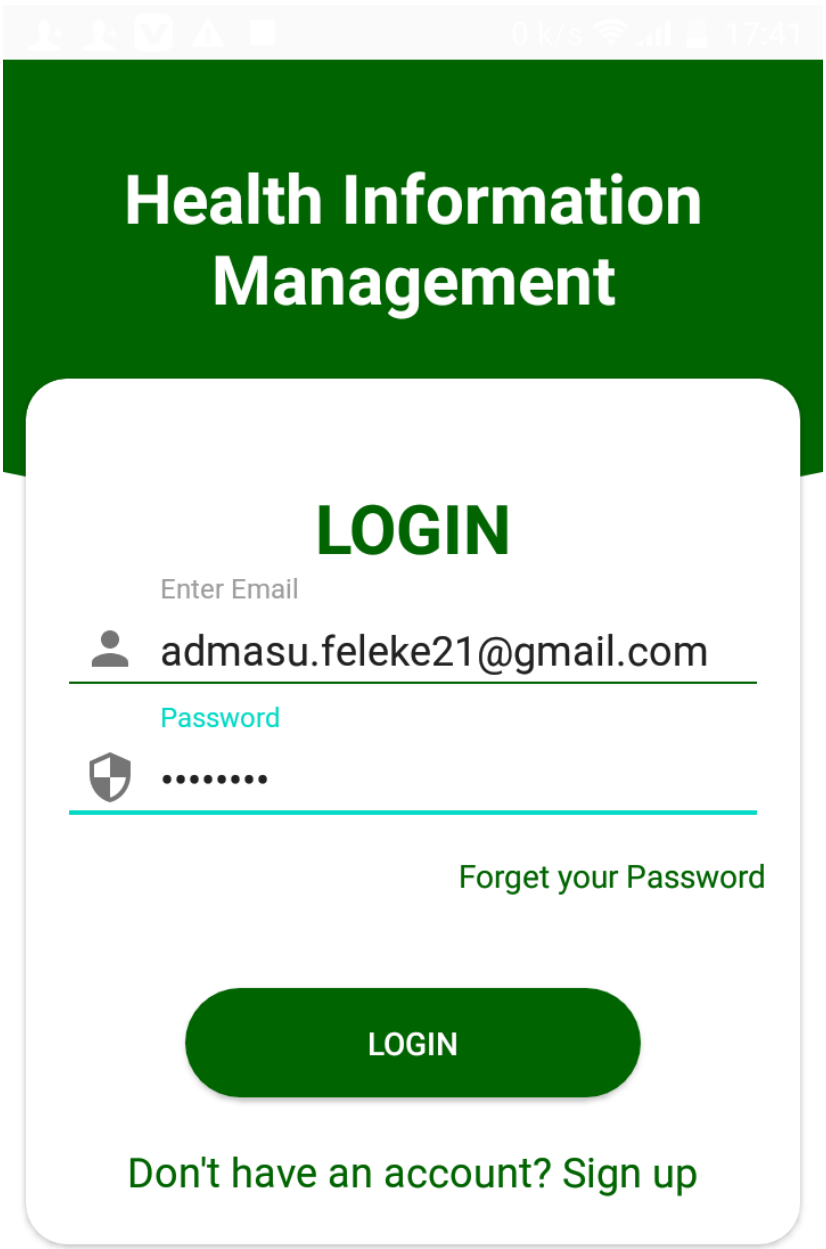



Figure 33. User interface for login page.

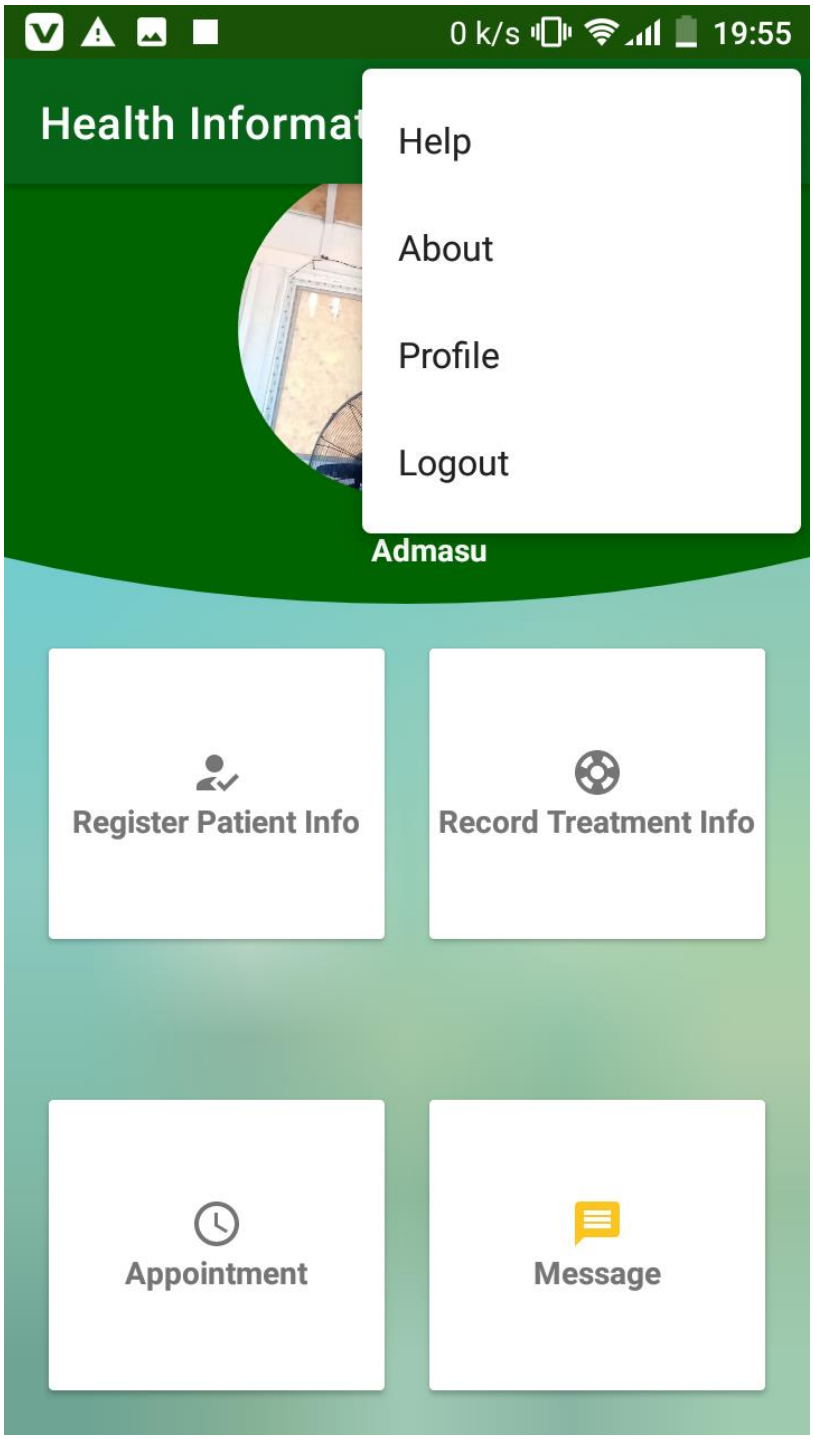



Figure 34. A user interfaces home page for doctor.

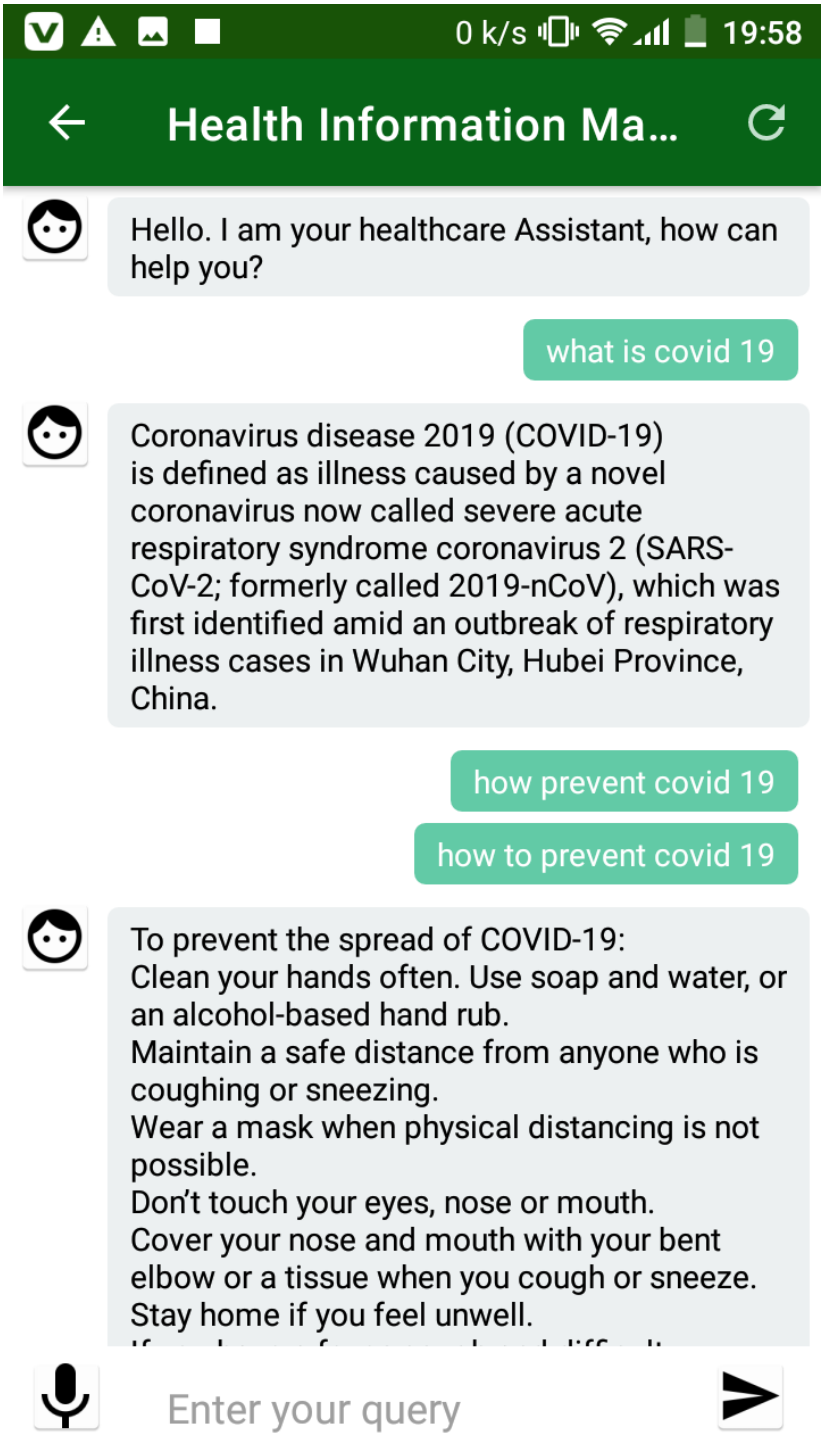



Figure 35. User interface for health care chatbot system

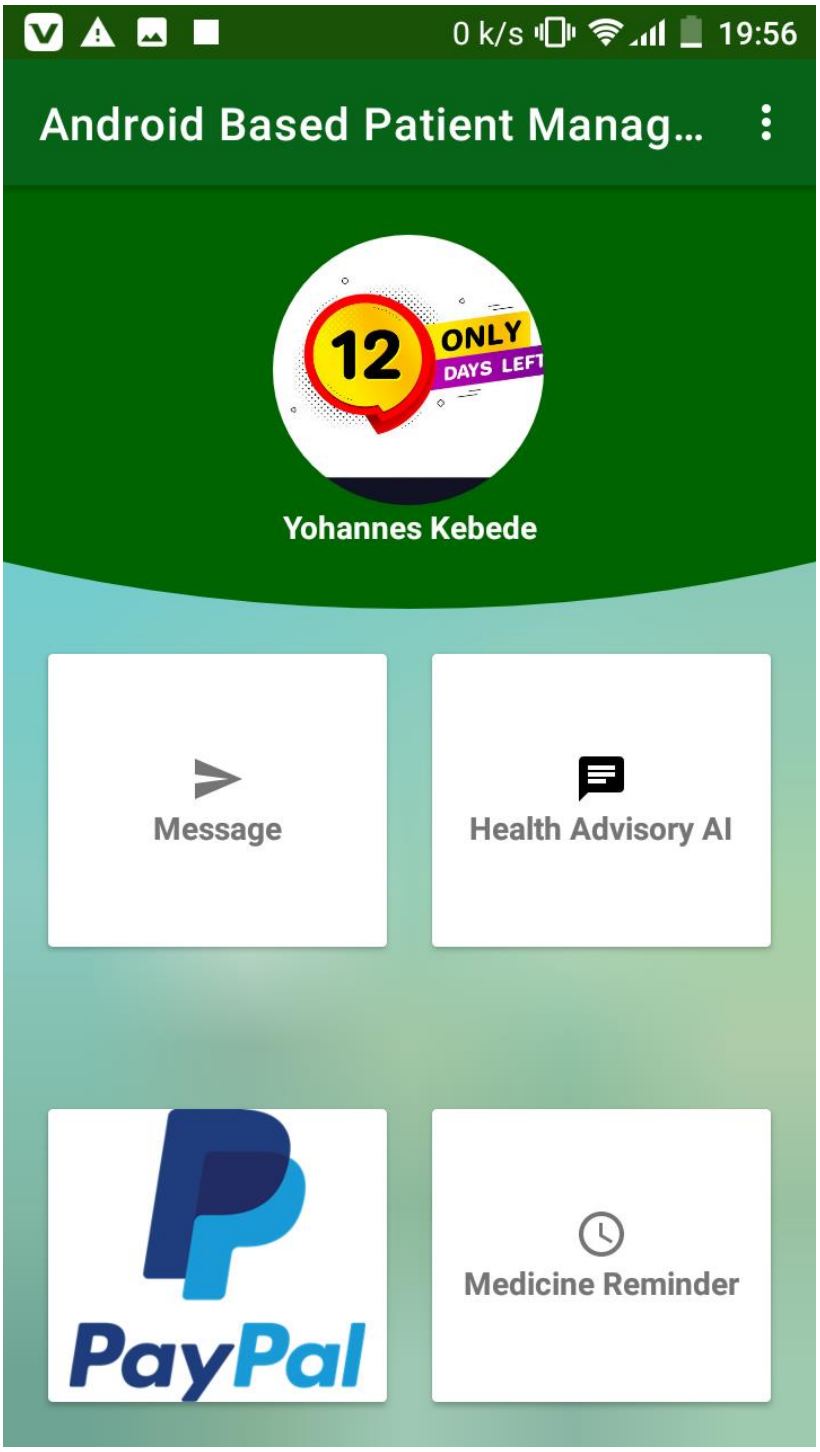



Figure 36. A user interfaces home page for patient.

# CHAPTER SIX

# 6. IMPLEMENTATION AND TESTING

## 6.1. IMPLEMENTATION

Implementation is the phase where objectives of physical operations of the system turned into reality i.e., real working model. The crucial phase in the system development life cycle is the successful implementation of the new system design. The process of converting as new system into an operational one is known as system implementation. This includes all those activities that take place to convert from an old system to a new system.

## 6.2. Implementation of the Database

In order to implement the application, we have used Firebase Realtime database to store text and integer values, and Firebase Storage to store images. We have selected these database because it supports real time data, it is tree structure just creating node and sub nodes, no SQL command is required. And also, we have implemented SQLite database which is default database in Android Studio to store value of appointment and reminder.

Sample Exported JSON file of our apps database


```
{
 "Appointment" : {
  "-MhD928-t8NsBTRDBD0K" : {
   "date" : "17/8/2021",
   "place" : "yyt",
   "subject" : "gdh",
   "time" : "1:50"
  },
  "appointment" : {
   "date" : "12/45/11",
   "place" : "home",
   "subject" : "home vising",
   "time" : "12:00/yuuhu"
  }
 },
 "Chats" : {
  "-MhS5fFrVZZtCa8yNR_a" : {
   "isseen" : true,
```

```
  "message" : "hi",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "status" : "offline"
},
"-MhS6jN7vOlZTgWwzrq7" : {
  "isseen" : true,
  "message" : "hello",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "status" : "offline"
},
"-MhS7uVABr3KgR7XUrSn" : {
  "isseen" : true,
  "message" : "hi",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "status" : "offline"
},
"-MhS7vziI5Wx4W9REKG0" : {
  "isseen" : true,
  "message" : "hello",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "status" : "offline"
},
"-MhS9GkORwQafBVYoDfU" : {
  "isseen" : true,
  "message" : "how",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
},
"-MhS9I1eXsSE8tfVHJTc" : {
  "isseen" : true,
  "message" : "yes",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
},
"-MhSCYJgjeOwVfMcsdpU" : {
```

```
  "isseen" : true,
  "message" : "what man",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2"
},
"-MhSHZkzn6A1MQP5_jIZ" : {
  "isseen" : true,
  "message" : "how are you",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
},
"-MhSqJHclgiwx_uJfZ8x" : {
  "isseen" : true,
  "message" : "pis",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
},
"-MhSr9HJ_YzVEKnW-IGA" : {
  "isseen" : true,
  "message" : "man",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2"
},
"-MhSst89eKhbPhCCYn_m" : {
  "isseen" : true,
  "message" : "man",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
  "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2"
},
"-MhYPWQrsvnUBSKpt8T6" : {
  "isseen" : true,
  "message" : "hi",
  "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
  "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
},
"-MhbcF8xIC9ZPm2-BNJH" : {
  "isseen" : true,
  "message" : "wanaw",
  "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
```

```
    "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2"
   },
   "-MhbcleMu46tY4X501f0" : {
    "isseen" : true,
    "message" : "ghbnxb",
    "receiver" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
    "sender" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2"
   },
   "-MhbgNigStgLoSK27aiK" : {
    "isseen" : true,
    "message" : "hi",
    "receiver" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
    "sender" : "6j1VThg8AYhblGZWWl80I2RGbyH2"
   }
  },
  "Patient" : {
   "-MhT5Ey0pklwF5DmceTA" : {
    "age" : "12",
    "email" : "ftt@.com",
    "houseNumber" : "67",
    "kebele" : "66",
    "name" : "hi",
    "phone" : "96366",
    "purl" : "1629371302600 jpg",
    "sex" : "Male",
    "town" : "yhgy"
   },
   "-MhT5G_5cNKdqKCiX3KL" : {
    "age" : "12",
    "email" : "ftt@.com",
    "houseNumber" : "mak",
    "kebele" : "66",
    "name" : "fygg",
    "phone" : "96366",
    "purl" : "1629371309209 jpg",
    "sex" : "Male",
    "town" : "yhgy"
   },
   "-MhTChikWVmMMIyvabdz" : {
```

```
    "age" : "23",
    "email" : "yy@g..com",
    "houseNumber" : "6776",
    "kebele" : "67",
    "name" : "yyh",
    "phone" : "+2519008867",
    "purl" : "1629373259630 jpg",
    "sex" : "Female",
    "town" : "yhjyy"
  }
 },
 "Treatment" : {
  "-Md3MWGW6WJ315Y8yz9b" : {
    "ill_date" : "Ill Dateq2/123",
    "prescription" : "Prescriptiontyjjdjjshheue7e7",
    "symptoms" : "colx"
  },
  "-MhSRns2wSk2g2NedqPR" : {
    "ill_date" : "19-8-2021",
    "prescription" : "jj^kkkjnnnnnnnbhjmmjj\nhjjkkkmbhjskkzkkznznzn",
    "symptoms" : "jardia"
  },
  "tttt" : {
    "ill_date" : "12/45/79",
    "prescription" : "uguyfytdyt",
    "symptoms" : "dkyr"
  }
 },
 "Users" : {
  "6j1VThg8AYhblGZWWl80I2RGbyH2" : {
    "as" : "user",
    "email" : "user@g.com",
    "id" : "6j1VThg8AYhblGZWWl80I2RGbyH2",
    "image" : "https://firebasestorage.googleapis.com/v0/b/abpmsfd-
787c4.appspot.com/o/Profile%20pic%2F6j1VThg8AYhblGZWWl80I2RGbyH2.jpg?alt=media&token=d70d77f2-87db-
4ecf-ab42-2eef5c14fcf9",
    "img_url" : "https://firebasestorage.googleapis.com/v0/b/abpmsfd-
787c4.appspot.com/o/Uploads%2F1629354948379.jpg?alt=media&token=0567a43e-8b1e-4852-a0bb-bf62b9611374",
    "purl" : "default",
```

```
    "status" : "Offline",
    "username" : "user1"
   },
   "DozCT3OSLQdAHW2QBhWWSpEs3ws2" : {
    "as" : "admin",
    "email" : "admin@g.com",
    "id" : "DozCT3OSLQdAHW2QBhWWSpEs3ws2",
    "image" : "https://firebasestorage.googleapis.com/v0/b/abpmsfd-
787c4.appspot.com/o/Profile%20pic%2FDozCT3OSLQdAHW2QBhWWSpEs3ws2.jpg?alt=media&token=66a39d90-
b68b-46d2-8a0e-c1fefeab6319",
    "img_url" : "https://firebasestorage.googleapis.com/v0/b/abpmsfd-
787c4.appspot.com/o/Uploads%2F1629355246788.jpg?alt=media&token=be5cca58-77d7-44df-b008-fc2ca7192080",
    "purl" : "default",
    "status" : "Offline",
    "username" : "Admas admin"
   }
  }
}
```


## 6.3. Implementation of the Class Diagram

| Class | Method | Variable |
|---|---|---|
| +class Signup | +register(); | registerEmail String<br>regUsername String<br>regPass String<br>comfirmPass String |
| +class PatientDetail | +Register();<br>+update();<br>+delete();<br>+view(); | Email String<br>PatientName String<br>Age int<br>Sex string |

| | | |
|---|---|---|
| | | Phone string<br>Town string<br>Kebele string<br>HouseNumber string |
| +class TreatmentInfo | +Register();<br>+update();<br>+view(); | Name String<br>Ill_Date string<br>Symptom's string<br>Prescription string |
| +class AppointmentDetail | +Record();<br>+update();<br>+delete();<br>+view(); | Subject string<br>Date string<br>Time string |
| +class chatbot | +send();<br>+view(); | HelpQueries string |
| +Message | +send();<br>+view | Subject string |

## 6.4. Configuration of the Application Server

Firebase is a Backend-as-a-Service and thus app-development platform on Google Cloud Platform. Firebase is used as server, API and datastore, all written so generically that we can modify it to suit most needs. In our project Firebase is a back-end and as well as server to our app as a server. It can be accessed through the application anywhere.

## 6.5. Configuration of Application Security

Firebase Authentication uses an internally modified version of scrypt to hash account passwords. Even when an account is uploaded with a password using a different algorithm, Firebase Auth will rehash the password the first time that account successfully logs in. Accounts downloaded from Firebase Authentication will only ever contain a password hash if one for this version of scrypt is available, or contain an empty password hash otherwise.

## 6.6. Implementation of User Interface

We implement the application we have developed has good and friendly user interface that user can easily navigate the app. The user interface is portion of an interactive computer system that user can communicate the application. Design of the interface includes any aspect of the system that is visible to user. The system has consistence user interface when we navigating through page.

## 6.7. Testing

Final phase of implementation is testing. Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item. Software testing is an activity that should be done throughout the whole development process

Testing is a process to show the correctness of the program. Testing is checking of the system workability in an attempt to discover errors and avoiding such errors from the system. In this the team members tested the entire system as a whole with all forms, code, modules. In this we tested all the functionalities in the System. All errors in the forms, functions, modules have been tested.

### 6.7.1. Test Case.

<table>
<tr><td colspan="3">Test Case ID = TestCase1</td></tr>
<tr><td colspan="3">Unit to Test = Authentication of login users</td></tr>
<tr><td colspan="3">Assumptions = Login into appropriate page<br><br>Test Data = User Name(empty , invalid user name, valid username)<br>Password (empty ,invalid password, valid password)</td></tr>
<tr><td>Steps to be Executed</td><td>Data</td><td>Expected Results</td></tr>
<tr><td>Empty Username and password and Click Login button</td><td>Any valid data for the other fields</td><td>“Please enter User name and password”</td></tr>
</table>

| | | |
|---|---|---|
| Enter invalid Username, password and Click Login button | Any valid data for the other fields | "Invalid Username or Password.<br>Please Enter Again ... " |
| Enter valid username ,empty Password and Click Login button | Any valid data for the other fields | **"**Please Enter Password**"** |
| Enter valid password ,empty user name and Click Login button | Any valid data for the other fields | "Please Enter user name" |
| Enter valid Username and Password then Click Login button | All fields Fulfill with valid data | "Login in to approprate page" |

### 6.7.2. Testing Tools and Environment

- Smart mobile phone
- Wi-Fi/Internet connection
- Computer
- Chrome

### 6.7.3. Unit Testing

Unit testing is every module of the System is separately tested. We have performed unit testing by project members to test that the unit has implemented is producing expected output against given input.

Table 6.8.1Unit Test Description for login

| |
|---|
| **Test Case ID** = TestCase1 |
| **Unit to Test** = Authentication of login users |

| | | |
|---|---|---|
| **Assumptions** = Login into appropriate page | | |
| **Test Data** = User Name(empty , invalid user name, valid username)<br>Password (empty ,invalid password, valid password) | | |
| **Steps to be Executed** | **Data** | **Expected Results** |
| Empty Username and password and Click Login button | Any valid data for the other fields | "Please enter User name and password" |
| Enter invalid Username, password and Click Login button | Any valid data for the other fields | "Invalid Username or Password.<br>Please Enter Again ... " |
| Enter valid username ,empty Password and Click Login button | Any valid data for the other fields | **"**Please Enter Password**"** |
| Enter valid password ,empty user name and Click Login button | Any valid data for the other fields | "Please Enter user name" |
| Enter valid Username and Password then Click Login button | All fields Fulfill with valid data | "Login in to approprate page" |

### 6.7.4. Integration Testing

Integration testing is the activity of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before acceptance testing. Thus, Integration testing is a logical extension of unit testing. In its simplest form, two units that we have already tested and combined them into a component and the interface between them and

tested. Here a component refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined into components, which are in turn aggregated into even larger parts of the program. The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups.

Integration testing can be done in a variety of ways but the following are three common strategies:

- The top-down approach to integration testing requires the highest-level modules be test and integrated first.
- The bottom-up approach requires the lowest-level units be tested and integrated first.
- The third approach, sometimes referred to as the umbrella approach, requires testing along functional data and control-flow paths.

### 6.7.5. Acceptance Testing

Acceptance testing is the process of testing system (e.g. software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery. A system is mainly developed for an end user normally a customer of the organization. A system is said to be accepted if and only if the user of the system is satisfied. In this perspective acceptance testing is widely used to prove that system performs as per the requirements.

In acceptance testing the customers provides the input data to validate the system operation. It is also known as functional testing, black-box testing, release acceptance, QA testing, application testing, confidence testing, final testing, validation testing, or factory acceptance testing.

### 6.7.6. System Testing

It is the final step of testing. In this the team members tests the entire system as a whole with all forms, code, modules. This form of testing is popularly known as Black Box testing or System tests. In this the team members tests all the functionalities in the System. All errors in the forms, functions, modules are tested.

# CHAPTER SEVEN

# 7. CONCLUSION AND RECOMMENDATION

## 7.1. Conclusion

In this Topic we summarized the result of the study that has been found in the analysis. Based on the result rational suggestions and recommendations have been forwarded. The project is partitioned into two phases and each phase has a specific deliverable which is essential and base for the next phase.

The first phase is the Introduction in this part the major works are: describing about background where the new application is intended to be built on, the problems in the Application have been identified, the feasibility part studies building a new application is possible or not from different perspectives, after the scope of the proposed application has been defined the selected programming tools , methods are stated ,analyzing Current applications and prior related work with their strengthen and wakens of related work, over view of proposed application with their Functional Requirements, Non Functional Requirements, Object Oriented System Modeling With Use case diagram and dynamic diagram and the project's phases with their respective deliverables are stated.

The second phase is conducted based on the first phase. In this phase we design the actual implementation of the application by using detailed class diagram, proposed architecture, component diagram, deployment diagram and persistent data modeling. After designing the next stage is implementation. In the implementation stage we change the above design into coding to make good interactive interface and store the data into the database. After we have written the actual code, we tested all controls and subsystems of the application by using different testing mechanisms. Such as unit testing, integration testing, acceptance testing and system testing.

By following the above steps, we have developed the application. This developed application has solved the problems that we have stated in the statement of problems.

## 7.2. Recommendation

The system that we are trying to develop is not a fully executed application. Because we were new to the system, we missed out some parts which should be included in the system. This is mainly due to limited development capacity, shortage of resource and shortage of time. Therefore, we

suggest the following features need to be incorporated in any further revision and extension attempt.

- All the necessary applications are not incorporated in this applications duet the shortage of time and lack of android coding experience; we recommend that to add the necessary applications. Such as Appointment announcement option application is not as included, so we recommend to including this task into the application.
- The user that is the doctor should be familiar with smart phone.
- The chatbot is not fully trained because we did not had enough time to consult with health experts.

## Reference


1. https://www.vogella.com/tutorials/Android/article.html [2021,Aug,22]
2. http://stackoverflow.com/
3. http://www.vogella.com/articles/AndroidSQLite/article.html
4. Books (2008_Ed_Burnette_Hello_Android, mobile computing principles text book and from_OO-SAD lecture note)
5. J. Runbaugh, I. Jacobson and G. booch,**The unified modelling language reference manual,** Addison Wesley, 1999
6. http://developer.android.com/index.html
7. v-softinc, 2013, **brief-history-of-mobile-applications.html,** October 26
8. Ian Sommerrvil. **Software Engineering**. Prentice Hall Inc, New York, 2004
9. http://www.vogella.com/tutorials/AndroidSQLite/article.html
10. ttp://agile.csc.ncsu.edu[2021, June, 05]
11. http://www.tutorialspoint.com/android/index.htm[2021, Aug,12]
12. http://www.javacodegeeks.com/2021/09/android-alarmmanager-tutorial.htm

# APPENDICES

## Appendix I

### Sample patient interview questions to the chatbot

- Where it is located on the body (such as pain)

  Does it radiate?

- Timing

  When did it start / how long has it been going on?

  Is this a new problem / first time having this problem?

  Intermittent or constant?

  If intermittent, how long does it last, and how often do it occur?

- Quality (particularly important if pain)

  Dull, sharp/knife- like, achy, pressure, tightness, tingling, etc.

- Severity or intensity
- What triggers the symptom

  What were you doing when it started?

  What causes it to occur

- Alleviating / aggravating factors

  What helps it o What makes it worse

- Any other symptoms that you have?

# Appendix II

## Sample code for login

```
package com.example.abpmsfd;


import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Patterns;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;


import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;


import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;


public class LoginActivity extends AppCompatActivity {


  EditText logEmail, lPass;
```

```
Button loginButton;
TextView forgot_password, tbtn;
//Switch active;
private ProgressDialog mLoadingBar;
FirebaseAuth firebaseAuth;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    logEmail = findViewById(R.id.logEmail);
    lPass = findViewById(R.id.logPass);
    loginButton = findViewById(R.id.logButton);
    tbtn = findViewById(R.id.noAccount);
    //active = findViewById(R.id.active);
    forgot_password = findViewById(R.id.forgot_password);

    mLoadingBar = new ProgressDialog(LoginActivity.this);

    forgot_password.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View v) {
          startActivity(new Intent(LoginActivity.this, ResetPasswordActivity.class));
       }
    });
    loginButton.setOnClickListener(new View.OnClickListener() {

       @Override
       public void onClick(View v) {
          if (logEmail.getText().toString().isEmpty()) {
             logEmail.setError("Empty email field");
```

```
        } else if (!Patterns.EMAIL_ADDRESS.matcher(logEmail.getText().toString()).matches()) {
            logEmail.setError("Invalid email address!");
        } else if (lPass.getText().toString().isEmpty()) {
            lPass.setError("Empty Password!");
        }
        else if (lPass.getText().toString().length()<8){
            lPass.setError("Enter 8 length password");
        }

        else
        {

        FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();

        if (!(logEmail.getText().toString().isEmpty() && lPass.getText().toString().isEmpty())) {
            mLoadingBar.setMessage("Authenticating...");
            mLoadingBar.setTitle("LOGIN");
            mLoadingBar.setCanceledOnTouchOutside(false);
            mLoadingBar.show();
            firebaseAuth.signInWithEmailAndPassword(logEmail.getText().toString(),
lPass.getText().toString())
                    .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                String uid = task.getResult().getUser().getUid();
                                FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
                                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                                if (user.isEmailVerified()) {
                                    firebaseDatabase.getReference().child("Users").child(uid).child("as")
```

```
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        String as1 = snapshot.getValue(String.class);
                        assert as1 != null;
                        if (as1.equals("admin")) {
                            logEmail.setText("");
                            lPass.setText("");
                            Intent signin = new Intent(LoginActivity.this,
MainActivity.class);
                            startActivity(signin);
                            mLoadingBar.dismiss();

                        }

                        if (as1.equals("user")) {
                            logEmail.setText("");
                            lPass.setText("");
                            Intent signin = new Intent(LoginActivity.this,
PatientDashboard.class);
                            startActivity(signin);
                            mLoadingBar.dismiss();
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });
            }
```

```
                        else {
                            Toast.makeText(LoginActivity.this, "Please verify your email!",
Toast.LENGTH_SHORT).show();
                        }
                    }
                    else {
                        Toast.makeText(LoginActivity.this, "Authentication Failed",
Toast.LENGTH_SHORT).show();
                        mLoadingBar.dismiss();
                    }
                }
            });
        }
    }
    }
});

    tbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(LoginActivity.this, Signup.class));
        }
    });
}

@Override
public void onBackPressed() {
    moveTaskToBack(true);
}
}
```

## Sample code for Signup

```
package com.example.abpmsfd;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;

public class Signup extends AppCompatActivity {

    private FirebaseAuth mAuth;
```

```
DatabaseReference reference;
private ProgressDialog mLoadingBar;
EditText registerEmail,regUsername, registerPass, confirmPassword;
Button registerButton;
TextView tbtn;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    registerEmail = findViewById(R.id.regEmail);
    regUsername = findViewById(R.id.regUsername);
    registerPass = findViewById(R.id.regPass);
    confirmPassword = findViewById(R.id.confirmPass);
    registerButton = findViewById(R.id.regButton);
    tbtn = findViewById(R.id.member);
    mAuth = FirebaseAuth.getInstance();
    mLoadingBar = new ProgressDialog(Signup.this);
    tbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(Signup.this, LoginActivity.class));
        }
    });

    registerButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (registerEmail.getText().toString().isEmpty())
            {
```

```
            registerEmail.setError("Empty email field");
        }
        else if (regUsername.getText().toString().isEmpty()){
            regUsername.setError("Empty username not allowed");
        }
        else if (registerPass.getText().toString().isEmpty()){
            registerPass.setError("Empty password not allowed!");
        }
        else if (confirmPassword.getText().toString().isEmpty()){
            confirmPassword.setError("Empty password not allowed!");
        }
        else if (!Patterns.EMAIL_ADDRESS.matcher(registerEmail.getText().toString()).matches()){
            registerEmail.setError("Invalid email, please insert valid email!");
        }

      else if (registerPass.getText().toString().length()<8 ||
confirmPassword.getText().toString().length()<8){
            Toast.makeText(Signup.this, "Password must be 8 characters long",
Toast.LENGTH_SHORT).show();
        }
        else if (!registerPass.getText().toString().equals(confirmPassword.getText().toString())){
            Toast.makeText(Signup.this, "Password and Confirm password do not match",
Toast.LENGTH_SHORT).show();
        }
        else {
          register(registerEmail.getText().toString(), registerPass.getText().toString(),
confirmPassword.getText().toString(), regUsername.getText().toString());
        }
    }
});
```

```
    }


    public void register(final String email, final String password, String confirmPassword, String
regUsername){
        mLoadingBar.setTitle("Registration");
        mLoadingBar.setMessage("Registering please wait!");
        mLoadingBar.setCanceledOnTouchOutside(false);
        mLoadingBar.show();
        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()){
                            FirebaseUser firebaseUser = mAuth.getCurrentUser();
                            FirebaseUser user = mAuth.getCurrentUser();
                            user.sendEmailVerification().addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    Toast.makeText(Signup.this, "Verification email has been sent, Please Verify your
Email!", Toast.LENGTH_SHORT).show();
                                }
                            }).addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    Toast.makeText(Signup.this, "Email not sent", Toast.LENGTH_SHORT).show();
                                }
                            });
                            assert firebaseUser != null;
```

```
                String userid = firebaseUser.getUid();
                reference = FirebaseDatabase.getInstance().getReference("Users").child(userid);

                HashMap<String, String> hashMap = new HashMap<>();
                hashMap.put("id", userid);
                hashMap.put("email", email);
                hashMap.put("purl", "default");
                hashMap.put("username", regUsername);
                hashMap.put("as", "user");
                hashMap.put("profession", "Patient");
                reference.setValue(hashMap).addOnCompleteListener(new
OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if(task.isSuccessful()){

                            Intent intent = new Intent(Signup.this, LoginActivity.class);
                            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|
Intent.FLAG_ACTIVITY_NEW_TASK);
                            startActivity(intent);
                            finish();
                            mLoadingBar.dismiss();
                        }
                    }
                });
            }

        }
    });
  }
```

```
}
```