

A PROJECT REPORT
ON
“HEALTH RECORD CLIENT BASED ON ANDROID
PLATFORM ”

Submitted to
UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN
COMPUTER ENGINEERING

BY

ALMAS ABDUL RAHIM SAUDAT	18DCO01
MATTE MEHVISH FARRUKH RUBINA	18DCO11
SHIRGAONKAR IFFAT SALIM NASEEMA	18DCO21

UNDER THE GUIDANCE OF
PROF. REHAAL QURESHI



DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam’s Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY

Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2020-2021

AFFILIATED TO
UNIVERSITY OF MUMBAI

**A PROJECT II REPORT
ON**

“HEALTH RECORD CLIENT BASED ON ANDROID PLATFORM”

**Submitted to
UNIVERSITY OF MUMBAI**

In Partial Fulfilment of the Requirement for the Award of

**BACHELOR’S DEGREE IN
COMPUTER ENGINEERING**

BY

ALMAS ABDUL RAHIM SAUDAT	18DCO01
MATTE MEHVISH FARRUKH RUBINA	18DCO11
SHIRGAONKAR IFFAT SALIM NASEEMA	18DCO21

**UNDER THE GUIDANCE OF
PROF. REHAAL QURESHI**



**DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam’s Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206**

**2020-2021
AFFILIATED TO**



UNIVERSITY OF MUMBAI

Anjuman-i-Islam's Kalsekar Technical Campus

Department of Computer Engineering

SCHOOL OF ENGINEERING & TECHNOLOGY

Plot No. 2 3, Sector - 16, Near Thana Naka,

Khandagaon, New Panvel - 410206



CERTIFICATE

This is certify that the project entitled

**“HEALTH RECORD CLIENT BASED ON ANDROID
PLATFORM“**

submitted by

ALMAS ABDUL RAHIM SAUDAT	18DCO01
MATTE MEHVISH FARRUKH RUBINA	18DCO11
SHIRGAONKAR IFFAT SALIM NASEEMA	18DCO21

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at *Anjuman-I-Islam's Kalsekar Technical Campus, Navi Mumbai* under the University of MUMBAI. This work is done during year 2020-2021, under our guidance.

Date: / /

(Prof. REHAAL QURESHI)
Project Supervisor

(Prof. KALPANA BODKE)
Project Coordinator

(Prof. TABREZ KHAN)
TAGI
HOD, Computer Department

DR. ABDUL RAZAK HONNU-
Director

External Examiner

Acknowledgements

I would like to take the opportunity to express my sincere thanks to my guide **REHAAL QURESHI**, Assistant Professor, Department of Computer Engineering, AIKTC, School of Engineering, Panvel for her invaluable support and guidance throughout my project research work. Without her kind guidance & support this was not possible.

I am grateful to her for her timely feedback which helped me track and schedule the process effectively. Her time, ideas and encouragement that she gave is help me to complete my project efficiently.

We would like to express deepest appreciation towards **DR. ABDUL RAZAK HONNUTAGI**, Director, AIKTC, Navi Mumbai, **Prof. TABREZ KHAN**, Head of Department of Computer Engineering and **Prof. KALPANA BODKE**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

ALMAS ABDUL RAHIM SAUDAT
MATTE MEHVISH FARRUKH RUBINA
SHIRGAONKAR IFFAT SALIM NASEEMA

Project I Approval for Bachelor of Engineering

This project entitled *Project Title* by *Students Name* is approved for the degree of *Bachelor of Engineering in Department of Computer Engineering*.

Examiners

1.

2.

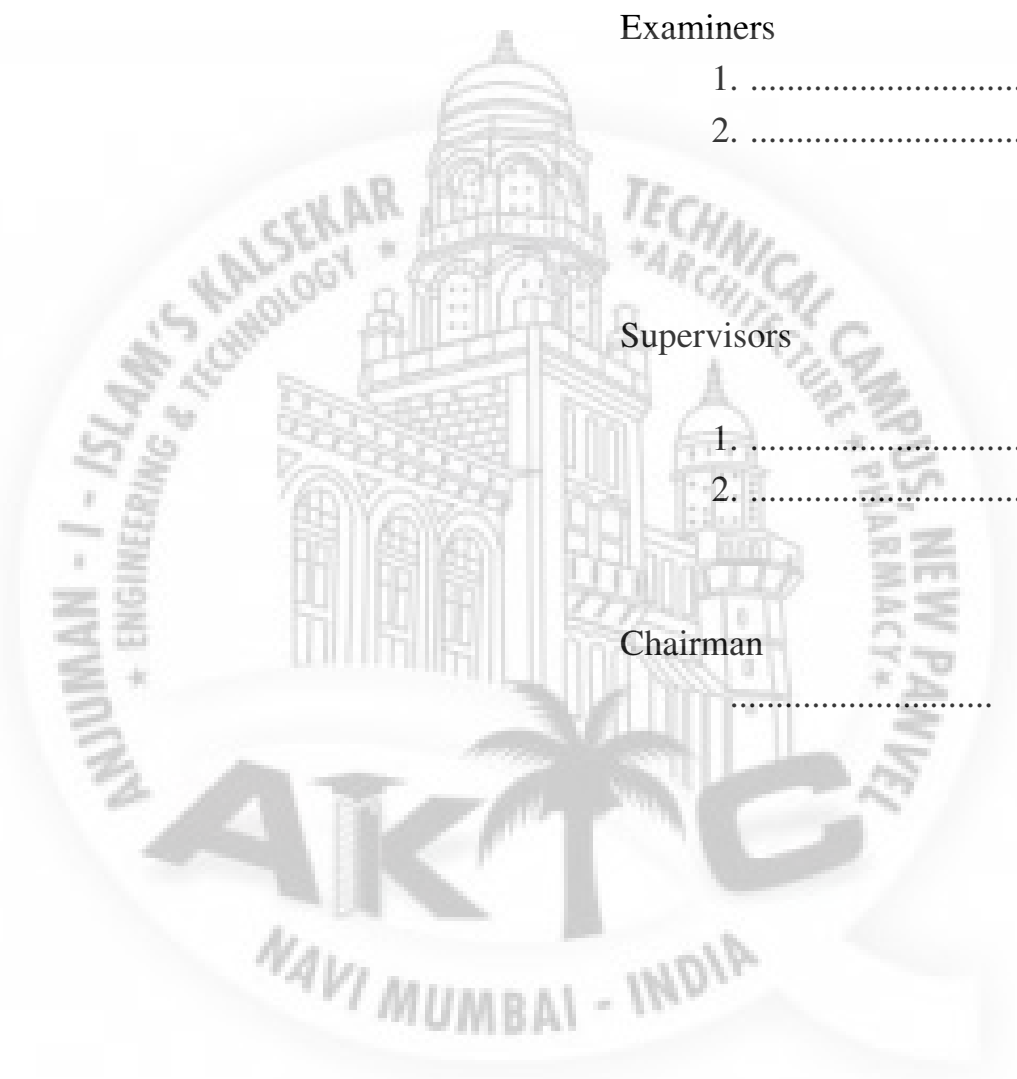
Supervisors

1.

2.

Chairman

.....



Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

ALMAS ABDUL RAHIM
18DCO01

MATTE MEHVISH FARRUKH
18DCO11

SHIRGAONKAR IFFAT SALIM
18DCO21

ABSTRACT

Title: HEALTH RECORD CLIENT BASED ON ANDROID PLATFORM

Over the past few decades our medical knowledge has increased. More investigative and treatment options are available; as a result our patients are living longer and we are dealing with more chronic conditions. Family physicians cannot “know all things” nor can we be “all things to all patients.” To adequately address our patients’ complex needs, we need good sources of information and good relationships, including access to a multidisciplinary team of professionals and other specialists. We need tools that improve access to information and relationships. We have had to transform how we practise, and the EMR, with its associated information technology, has facilitated that transformation.

Keywords :

Appointment, Online application, Android, Hospital, Scheduling, Track, Healthcare, Physician.

Abbreviation :

EHR(Electronic Health Record)

OS(Operating System)

HCP(Health Care Professional)

NP(Nurse Practitioner)

IDE(Integrated Development Environment)

Glossary :

A:

Account: Account is a location on a network server used to store a computer username, password, and other information.

Admin: Admin is the role with the highest level of access to your website.

Android: Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.

Android Studio: Android Studio is the official integrated development environment for Google’s Android operating system, built on JetBrains’ IntelliJ IDEA software and designed specifically for Android development.

API: A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

Appointment:An arrangement to consult doctor at a particular time and place.

C:

Consultation:A meeting with an expert, such as a medical doctor, in order to seek advice.

D:

Deployment:Make system available to use.

Doctor:A person who is qualified to treat people who are ill.

E:

Electronic Health Record:It is a digital version of a patient's paper chart.

F:

FireBase:Firebase is a platform developed by Google for creating mobile and web applications.

Function:Function is the most useful part for any programming language because with the help of function developer can define various methods, tasks into a single set of instructions and by calling this function you can perform simple defined task.

H:

Handset:A mobile phone.

J:

Java:Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

L:

Linux:Linux is a family of open-source Unix-like operating systems based on the Linux kernel.

M:

Medical Consultancy:An individual to whom one refers for expert advice or services.

*Mobile Application:*A mobile application, also referred to as a mobile app or simply an app, is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch.

O:

OS:An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

P:

Physician:A person qualified to practise medicine, especially one who specializes in diagnosis and medical treatment a person qualified to practise medicine, especially one who specializes in diagnosis and medical treatment.

Patient:A person receiving or registered to receive medical treatment.

R:

Record:To keep information for the future, by writing it down or storing it on a computer.

S:

Smart Phone:a mobile phone that performs many of the functions of a computer, having a touchscreen interface, internet access, and an operating system capable of running downloaded apps.

W:

Windows:Microsoft Windows, commonly referred to as Windows, is a group of several proprietary graphical operating system families, all of which are developed and marketed by Microsoft. Each family caters to a certain sector of the computing industry.

Contents

Acknowledgement	ii
Project I Approval for Bachelor of Engineering	iii
Declaration	iv
Abstract	v
Table of Contents	x
1 Introduction	2
1.1 Purpose	3
1.2 Project Scope	3
1.3 Project Need and Objectives	3
1.3.1 Need	3
1.3.2 Objectives	4
1.4 Innovativeness and Usefulness	5
1.4.1 Live Consultation	5
1.4.2 Real-Time Data and Accurate Decision Making	5
1.4.3 Patient Privacy	5
1.4.4 Patient Engagement	5
2 Literature Survey	6
2.1 Paper Title 1- 'Doctor Who?'- A Customizable Android Application for Integrated Health Care.	6
2.1.1 Advantages of Paper	6
2.1.2 Disadvantages of Paper	6
2.1.3 How to overcome the problems mentioned in Paper	6
2.2 Smart Healthcare'- A medical Record system for effective health services.	7
2.2.1 Advantages of Paper	7
2.2.2 Disadvantages of Paper	7
2.2.3 How to overcome the problems mentioned in Paper	7
2.3 Mr.Doc: A Doctor Appointment Application System	8
2.3.1 Advantages of Paper	8
2.3.2 Disadvantages of Paper	8
2.3.3 How to overcome the problems mentioned in Paper	8

2.4	Open ERP for Creating Medical Record Management System in Smart Hospital	9
2.4.1	Advantages of Paper	9
2.4.2	Disadvantages of Paper	9
2.5	Design of a Web-based and Electronic Health Record Management System for Medical Tele-consultation	10
2.5.1	Advantages of Paper	10
2.5.2	Disadvantages of Paper	10
2.6	Technical Review	12
2.6.1	Advantages of Technology	12
2.6.2	Dart	13
2.6.3	Firestore	14
2.6.4	Reasons to use this Technology	15
3	Project Planning	16
3.1	Members and Capabilities	16
3.2	Roles and Responsibilities	16
3.3	Assumptions and Constraints	16
3.3.1	Assumptions	16
3.3.2	Constraints	17
3.4	Project Management Approach	17
3.5	Ground Rules for the Project	18
3.6	Project Budget	19
3.7	Project Timeline	19
4	Software Requirements Specification	22
4.1	Overall Description	22
4.1.1	Product Perspective	22
4.1.2	Product Features	22
4.1.3	User Classes and Characteristics	22
4.1.4	Operating Environment	23
4.1.5	Design and Implementation Constraints	23
4.2	System Features	23
4.2.1	System Feature	23
4.3	External Interface Requirements	24
4.3.1	User Interfaces	24
4.3.2	Hardware Interfaces	24
4.3.3	Software Interfaces	24
4.4	Nonfunctional Requirements	24
4.4.1	Performance Requirements	24
4.4.2	Safety Requirements	25
4.4.3	Security Requirements	25

5	System Design	26
5.1	System Requirements Definition	26
5.1.1	Software Requirements	26
5.1.2	Hardware Requirements	26
5.1.3	System requirements (non-functional requirements)	28
5.2	System Architecture Design	29
5.3	Sub-system Development	29
5.3.1	Patient Profile Module	31
5.3.2	Doctor Profile Module	32
5.3.3	Sequence Diagram	33
6	Implementation	35
6.1	Doctor View	35
6.2	Prescription by Doctor	42
6.3	Patient View	48
6.4	Patient's Medical records	55
6.5	Consulting the Doctor	62
7	System Testing	68
7.1	Test Cases and Test Results	68
7.2	Test Cases	69
7.2.1	Software Quality Attributes	70
8	Screenshots of Project	71
8.1	Patient View	71
8.1.1	Registration	72
8.1.2	Login	73
8.1.3	Home Page	74
8.2	Doctor View	75
8.2.1	All Appointments	76
8.2.2	Appointment Scheduled	77
8.2.3	Video Calling	78
8.2.4	Prescription	79
8.2.5	Payment	80
9	Conclusion and Future Scope	81
9.1	Conclusion	81
9.2	Future Scope	82
	References	82

List of Figures

2.1	Flutter	12
3.1	Iterative Approach	17
3.2	Project Task Assigned to members	19
3.3	Project Task Assigned to members	20
3.4	Project Task Assigned to members	20
3.5	Project Task Assigned to members	21
5.1	Zero Level DFD	27
5.2	Level One DFD	28
5.3	System Architecture Design	29
5.4	Use case Diagram	30
5.5	Patient Profile Module	31
5.6	Patient function Module	31
5.7	Doctor Profile Module	32
5.8	Doctor Function Module	32
5.9	Sequence Diagram of Patient	33
5.10	Sequence Diagram of Patient	34
8.1	Patient Sign up	72
8.2	Home Page	74

List of Tables

3.1	Table of Capabilities	16
3.2	Table of Responsibilities	16



Chapter 1

Introduction

The title of this project is HEALTH RECORD CLIENT BASED ON ANDROID PLATFORM(Medico) , is a computerized collection of patient information in a digital format.EHR is real-time, patient-centred record that make information available instantly and securely to authorized users.

While an EHR does contain the medical and treatment histories of patients, an EHR system is built to go beyond standard clinical data collected in a provider's office and can be inclusive of a broader view of a patient's care. EHRs are a vital part of health IT and it: Contain a patient's medical history, diagnoses, medications, treatment plans, immunization dates, allergies, radiology images, and laboratory and test results and allow access to evidence-based tools that providers can use to make decisions about a patient's care The mechanisms of safety for electronic medical records in terms of both patient diagnosis and the security of their health records is one of the main elements that electronic medical records companies design into their software systems.The mechanisms of safety for electronic medical records in terms of both patient diagnosis and the security of their health records is one of the main elements that electronic medical records companies design into their software systems.

The main Aim To develop a system that allows users to: Minimize the risks of duplicate medical records and errors by providing access to real-time shared data.To Contribute to the delivery of integrated care.Digital health solutions can monitor patients in real time by collecting and analysing their health metrics through smart wearable devices.Connected medical solutions are of great value for preventive care, remote clinical monitoring, and chronic disease management.To Make healthcare more accessible.With the android application, people living in rural areas, physically disabled people, and elderly people can easily gain access to medical care. As for healthcare practitioners, they can reach wider audiences (even at the international level).Allow patients to take control of their health, communicate with their doctors when necessary, and monitor their own medical data.

This EHR Android Application is more than just a computerized version of a paper chart in a provider's office. It's a digital record that can provide comprehensive health information to doctors about their patients. EHR systems are built to share information with other health care providers and organizations – such as laboratories, specialists, medical imaging facilities, pharmacies, emergency facilities, and school and workplace clinics – so they contain information from all clinicians involved in a patient's care.

1.1 Purpose

In the coming years, EHR is better focusing on reporting the patient's presentation. It is assessing the flow of the patient through the hospital and stays that provide a better sense of the journey of the patient. The purpose of this project is that EHR stores the patient's information in an organized digital format. This information is available in real-time for authorized users, which makes the workflow easy and way more convenient. It has thus become an essential tool for hospitals as well as health practitioners across the world. EHR will provide the history of the patient. The user will check the level of details and the previous history of diagnosis, procedure or condition linking to recent records.

1.2 Project Scope

In the coming years, EHR is better focusing on reporting the patient's presentation. It is assessing the flow of the patient through the hospital and stays that provide a better sense of the journey of the patient. The goal of this project is that EHR stores the patient's information in an organized digital format. This information is available in real-time for authorized users, which makes the workflow easy and way more convenient. It has thus become an essential tool for hospitals as well as health practitioners across the world. EHR will provide the history of the patient. The user will check the level of details and the previous history of diagnosis, procedure or condition linking to recent records.

1.3 Project Need and Objectives

1.3.1 Need

- 1.Covid-19 virus has wreaked havoc around the world, infecting millions and raising fears of uncertainty. As a consequence, Governments have imposed stringent social distancing and lockdown measures to control the spread and contain the pandemic. This has left, those with medical priorities for routine check-ups and other chronic diseases in a quandary.

- 2. Amidst rising fears of disease infectivity from contaminated surfaces, people have turned their dependency on technology.
- 3. Technology powered virtual consultation has stepped up into the spotlight aiding healthcare provider organizations, doctors, and patients with digital consultations aired at the safe environs of a hospital/clinic straight to a patient's home.
- 4. Coronavirus pandemic has forced lockdowns and increased reliance in the field of telemedicine, which includes specialized consultations not only with general practitioners but with the specialist's into Oncology, Cardiology, and others.

1.3.2 Objectives

The Core objectives of Electronic Health records may be as follow:

1. Electronic Health Record (EHR) improve quality, safety and efficiency reduces costs.
2. Through Electronic Health Records Systems, physician and health practitioner improves care coordination.
3. To develop a system that allows users to have control over their appointment making service.
4. To facilitate the patients with real time healthcare scheduling.
5. To manage staff resources needed for managing appointments.
6. To maximize operation hours.
7. To make the use of online platform for less customer inconvenience and high productivity among staff.
8. To optimize time savings and monetary savings as both staff time and services translate into expenses and revenue.
9. It enhances the provided services to patients by making their records available online for physician to follow up the case easily with less effort, and their history would be available.
10. The physician can make their researches by using the advance search. Archiving and securing electronic records considered more reliable and trusted than paper-based records.

1.4 Innovativeness and Usefulness

1.4.1 Live Consultation

- People are facing difficulties in consulting with the doctors as they are unable to have physical communication or contact with their doctors so this would be a beneficial interface for patients to consult with their doctors with these unique feature.

1.4.2 Real-Time Data and Accurate Decision Making

- Real-time data is accessible from anywhere at any time. Commonly-needed data is virtually in every window view. There is no time wasted searching for pertinent patient data.

1.4.3 Patient Privacy

- Patients not only have easy access to their medical information, but their personal health information is not exposed to the general office staff who don't have anything to do with patient care.

1.4.4 Patient Engagement

- Patients can get alerts to schedule an appointment, and they can get an alert their results are ready for review.

Chapter 2

Literature Survey

2.1 Paper Title 1- 'Doctor Who?'- A Customizable Android Application for Integrated Health Care.

Author1:Abdur Razzak Author2:Robin Roy

This Online application has been designed and developed, which makes an easy and effective communications with the users to the doctors and hospitals. Using this app, patients can make appointment with doctors of different specialties in different locations and can take help for a 24/7 online medical consultancy and emergency ambulance service. This application will also provide users a list of registered blood donors with contact details for all existing blood groups. System provides the facility to the patient to choose the specialities of doctor required and select on the basis of rating or location .

2.1.1 Advantages of Paper

- a. The application can keep a record of the user's history.
- b. It can send a reminder exactly before two hours from the appointment time and send notifications if the doctor is unavailable on the appointed date and time.
- c. Application will keep record of date and time of donation and set an automatic remainder three months from the blood donation .

2.1.2 Disadvantages of Paper

- a. The system doesn't have any online features such as consultation through chats or video calls.

2.1.3 How to overcome the problems mentioned in Paper

- a. Once getting the appointment the patient can select the mode of consultation i.e; online or offline and online consultation can be done through application.

2.2 Smart Healthcare'- A medical Record system for effective health services.

Author1:Erwim Halim Author2:Diyurman Gea

This application of medical records that have good quality will affect the level of service to patients. Utilization of a good medical record management system, the orderly administration of health services becomes effective and efficient. The proposed systems are web-based and mobile application. This system allows the users to choose the doctor based on the location or speciality and also allow the line tracking of appointments and their queue status.

2.2.1 Advantages of Paper

- a. Patients are facilitated to register online through a mobile.
- b. application, get medical history records quickly, and get services that are fast and measurable.
- c. Doctors and nurses get the patient's medical record information quickly through a web-based application.
- d. Doctors conduct patient assessments and collect data through an online system, and the results can be read by all medical teams through an integrated system.

2.2.2 Disadvantages of Paper

- a. This system doesn't have features of consultation paper where all the prescribed medicines and the consultation from the doctor can be recorded and maintained with the user.

2.2.3 How to overcome the problems mentioned in Paper

- a. We can introduce the feature where the user has the record of previous consultation as well as the medicines prescribed. We can also allow the user to book and order medicine online.

2.3 Mr.Doc: A Doctor Appointment Application System

Author1:Shafaq Malik Author2:Nargis Bibi

The main idea of this work is to provide ease and comfort to patients while taking appointment from doctors and it also resolves the problems that the patients has to face while making an appointment. The android application Mr.Doc acts as a client whereas the database containing the doctor's details, patient's details and appointment details is maintained by a website that acts as a server. An Online Hospital Management Application that uses an android platform that makes the task of making an appointment from the doctor easy and reliable for the users. An intelligent agent based appointment system has been proposed in which a scheduling system is provided for patients.

2.3.1 Advantages of Paper

- a. An Online Hospital Management Application that uses an android platform that makes the task of making an appointment from the doctor easy and reliable for the users.
- b. An intelligent agent based appointment system has been proposed in which a scheduling system is provided for patients.

2.3.2 Disadvantages of Paper

- a. Many users only register themselves just for fun and has no concern by making an appointment.

2.3.3 How to overcome the problems mentioned in Paper

- a. A payment or some amount may be charged to the users/patients while making an appointment to avoid the unethical users.

2.4 Open ERP for Creating Medical Record Management System in Smart Hospital

Dickson Perdanakusuma, Warih Puspitasari, Muhardi Sapurti

A system to manage medical record data is a solution that must be immediately applied. The use of ERP systems can be a solution to the problems that exist at XYZ Hospital. ERP systems offer the integration of processes and data using a single database. The ERP system design at XYZ hospitals uses Odoo Software which is an open-source ERP software. ERP system design at XYZ hospitals uses the QuickStart method which is one of the fastest and cheapest methods for implementing ERP. The implementation of the ERP system at XYZ Hospital aims to create a patient medical record management system. Proposed business processes are made as solutions to problems that arise in the current business processes. The proposed business process design aims to eliminate problems that arise due to processes that are carried out manually. Proposed Business processes are designed in the form of block diagrams in which each block represents each process. The processes contained in this proposed business process are patient registration, medical service registration, and medical service billing.

2.4.1 Advantages of Paper

- a. ERP system can be a solution to the problems that exist in XYZ hospital, ERP system offers the concept of process and data integration so that it connects business functions comprehensively and coherently.
- b. there are many ERP solutions offered by software service providers, odoo is open source software, which means that the use of this software is free of charge.
- c. ○ Odoo software was chosen as the basis for system development at XYZ hospital because Odoo offers complete functionality and ease of development compared to others ERP software.

2.4.2 Disadvantages of Paper

- a. In the patient registration process, there is already a system used to store data but does not yet have an integration with the medical record.
- b. In the business process of billing medical services, there is not yet a system that integrates patient medical record data with billing functions. This causes the medical person to have to work extra to write data in the medical record file archive and do input to the billing system.

2.5 Design of a Web-based and Electronic Health Record Management System for Medical Tele-consultation

Nicole Jillian B. Day, Karmelo Antonio Lazaro R. Carranza, Lawrence Matthew S. Lin, Albert R. Ponce, Wilbur Rex O. Reyes, Nilo T. Bugtai and Renann G. Baldivino

The use of a telepresence system is a viable solution as doctors can still provide healthcare services even his patients are situated remotely. the study integrated a wireless data hub that hosts the web application and electronic health record (EHR) management system. The database structure was created using MySQL. Using HTML5 and Flask, the web application was developed as a mean for users to create, store, and access the user and patient information, remotely control the movement of the robot, allow two-way video telecommunication. It consists of the user accounts database (both for the patient and the doctor), patient information, consultation records, and medical files which can be accessed through a web application. The application serves as the main user interface for the patient's side and doctor's side functions. On the patient's side, the functionalities of the application include adding and viewing of patient information and medical files. To store and manage the data gathered in web application forms, MySQL database was used as the database system and was connected to the web-application using Flask. Different data tables were made for user information, patient information, pre-consultation record, consultation record, and medical files. A unique patient ID and consultation control number were generated to provide a unique relationship among tables.

2.5.1 Advantages of Paper

- a. A web application, using Flask, serves as the main user interface for all functionalities in the patient's and doctor's side.
- b. The developed web application has a responsive HTML template that could adapt to numerous screen sizes and was tested on an ASUS VivoStick and other operating systems.

2.5.2 Disadvantages of Paper

- a. One limitation of medical teleconsultation is the lack of medical features.
- b. Improvement of the web application and EHR security through different encryption and data validation forms, and to seek advice from hospitals that are currently using telepresence systems isn't resolved.

- c. Mobile browsers may not support some features of form fields. In this case, Mozilla Firefox, which is what was used as the mobile browser in the tablet, does not allow data in the date form field to be typewritten by the user, and only gives an option of clicking the date from the pop-up calendar.



2.6 Technical Review

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia,[4] and the web from a single codebase. The major components of Flutter include: Dart platform, Flutter engine, Foundation library, Design-specific widgets. Flutter apps are written in the Dart language and make use of many of the language's more advanced features. Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Skia graphics library. The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine. The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines.



Figure 2.1: Flutter

2.6.1 Advantages of Technology

Flutter:

Same UI and Business Logic in All Platforms. Reduced Code Development Time. Increased Time-to-Market Speed. Similar to Native App Performance. Custom, Animated UI of Any Complexity Available. Own Rendering Engine. Simple Platform-Specific Logic Implementation. The Potential Ability to Go Beyond Mobile.

2.6.2 Dart

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks. Languages are defined by their technical envelope — the choices made during development that shape the capabilities and strengths of a language. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hot reload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and desktop). Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.



Advantages of Technology

Dart The first advantage is that it is easy to learn. Any JavaScript programmer can quickly relearn how to write code in Dart. To do this, they only need to familiarize themselves with the basic principles of this programming language. The second thing is about its availability of documentation. Since Google is developing the interpreter for Dart, all the features of the language are described in detail. This allows you to quickly get answers to almost any questions that may arise during the training process, or directly while writing code. The third advantage is its high performance. Programs written in Dart tend to run faster than programs created in JavaScript. Dart is very stable and it can be used to build production quality real-time applications. It is an object-oriented programming language with support for inheritance, interfaces and optional typing features. It uses AOT and JIT compilation – Dart has the unique capability to handle both Ahead of time and Just in time compiling. In AOT, the Dart code can be directly converted into native machine code. While in the mode of JIT, it can be compiled for exceptionally fast development cycles and game-changing workflow. If you want to start writing your first Dart program without any installation or configuration, there is DartPad for you.

2.6.3 Firebase

Firestore is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firestore is categorized as a NoSQL database program, which stores data in JSON-like documents. It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firestore Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app. Data is synced across all clients in realtime and remains available even when an app goes offline. Firestore Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

Advantages of Technology

Firestore The cloud-hosted NoSQL database is offered by Firestore real-time database that helps you store and synchronize data between the clients. This indeed makes it easier for the developers to access the data using any of the devices and helps developing collaborative feature. Another advantage of a real-time database for the developers is that they do not need the support of backend to build apps as it comes with SDKs for various platforms, including Android, iOS and Web. It assists in the execution of backend code responding to events activated by databases. Furthermore, it is optimized for offline use too. It has often been seen that a lot of apps suffer due to bug issues, which tends to slow down navigation speed and users opt out of it. The result is that rating of the app also declines. However, you have to credit Firestore as now it is offering the facility of crash reporting to fix the bugs at the quicker pace and with ease. The app developers and QA testers can identify the problems in the stages, whether it is the app version, the device or the OS.

2.6.4 Reasons to use this Technology

a. Flutter:

Flutter provides easy and straightforward documentation with a large number of high-quality examples for reference. Developers who want to learn a new framework or a toolkit can opt for Flutter as it is easy to use and user-friendly language. Flutter allows you to build mobile apps for both platforms, including iOS and Android, with a single code base. Startups with a limited budget can spread their wings on all the major platforms with low development costs of Flutter apps.

b. FIREBASE:

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync between your users in realtime.

With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data.

Realtime syncing makes it easy for users to access their data from any device, be it web or mobile. Realtime Database also helps users collaborate with one another.

Another amazing benefit of Realtime Database is that it ships with mobile and web SDKs, allowing us to build apps without the need for servers.

c. Dart:

Dart is a very flexible programming language in that you can write the code and then run it anywhere without any limitations whatsoever. Mobile apps written in Dart with Flutter are cross-platform native apps; so they can run on both Android, iOS (like React Native, Xamarin, etc.).

Chapter 3

Project Planning

3.1 Members and Capabilities

Table 3.1: Table of Capabilities

SR. No	Name of Member	Capabilities
1	Almas Rahim	UI Design,Front-end Development
2	Mehvish Matte	Back-end and Front-end Development
3	Iffat Shirgaonkar	UI Design,Firestore

3.2 Roles and Responsibilities

Table 3.2: Table of Responsibilities

SR. No	Name of Member	Role	Responsibilities
1	Almas Rahim	Team Leader	UI Design,Back-end and Front-end Development,Documentation
2	Mehvish Matte	Member	UI Design,Back-end and Front-end Development,Documentation
3	Iffat Shirgaonkar	Member	Database Design,Documentation

3.3 Assumptions and Constraints

3.3.1 Assumptions

- A system that allows users to have control over their appointment making service.
- It can manage staff resources needed for managing appointments.
- The application can enhance the provided services to patients by making their records available online for physician to follow up the case easily with less effort, and their history would be available.
- The physician can make their researches by using the advance search.
- Patient can consult the doctor during emergency services.

3.3.2 Constraints

- Registered user can only book an appointment.
- After successful registration, user can view the list of Doctors.
- Doctors must be associated with the particular hospital for consultation.
- Calling feature from Patient side without registration on mobile application is forbidden.

3.4 Project Management Approach

We are following Iterative approach in our project

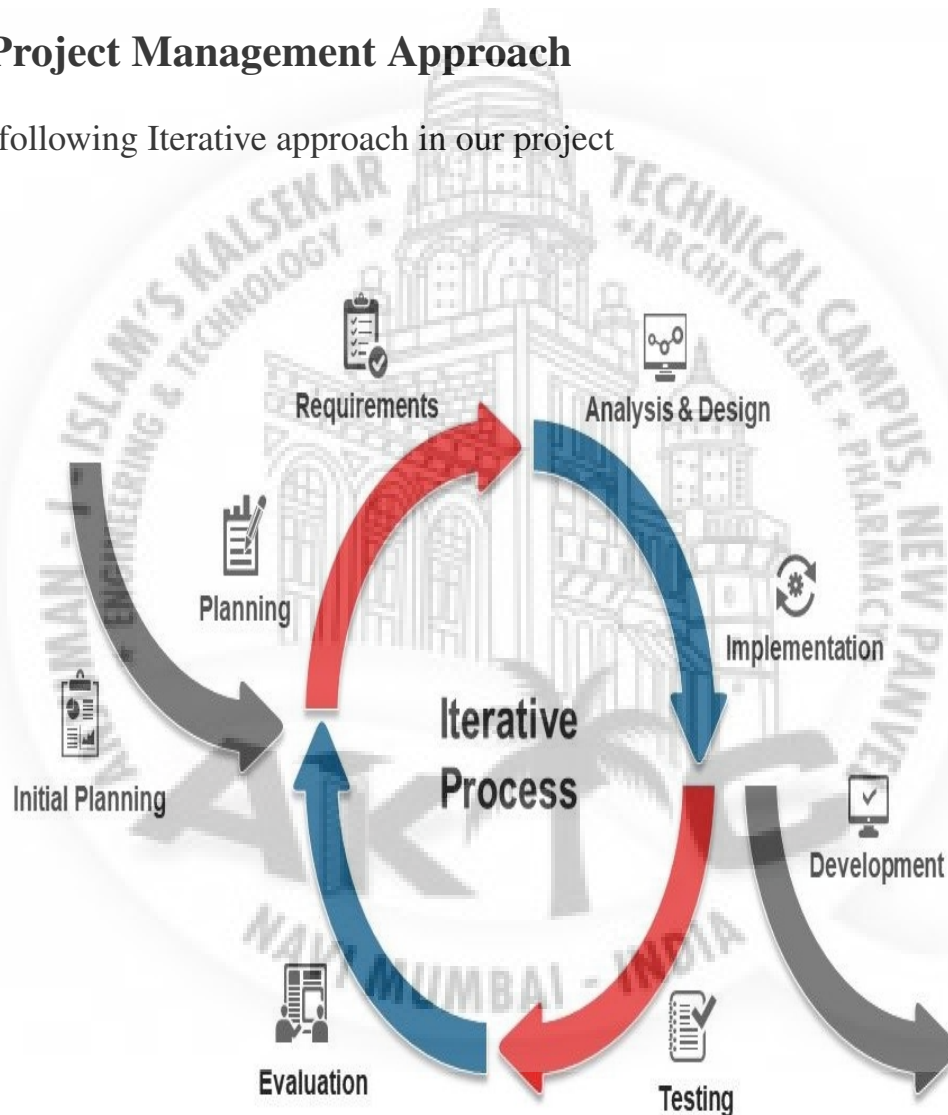


Figure 3.1: Iterative Approach

- Phases of Iterative approach are:

1. Planning Requirements:

In this phase, we planned the project approach and gathered requirements considering the patients convenience for booking the appointment and consulting

the doctor. In our project, application will provide services to patients by making their records available online for physician to follow up the case easily with less effort, and their history would be available. Patients requirements are effective UI and also optimize time savings and monetary savings as both staff time and services translate into expenses and revenue.

2. Analysis Design:

After the planning is complete, an analysis is performed to nail down the appropriate business logic, database models. In this phase, we perform analysis and design a right framework of the web application and mobile application that is to be developed.

3. Implementation:

With the planning and analysis out of the way, the actual implementation and coding process is done in this phase. All planning, specification, and design up to this point is implemented and coded here.

4. Testing:

Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have cropped up.

5. Evaluation:

Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on.

3.5 Ground Rules for the Project

1. Each team member have to work together with others member.
2. Each team member can share past experience with other members.
3. Each team member have to work on assigned task.
4. Members can share their idea.
5. Team members have to report daily to respective leader.
6. Talk softly with other members.
7. Participate in meeting.
8. Inform the leader about unavailability.

3.6 Project Budget

The budget for this project is very low as most of the tools we use are open source. Following are the budget for the project.

1. Operating System : Windows 10
2. Programming Languages : Dart, Flutter (Framework)

3.7 Project Timeline

Task views		Resource views		Other views	Sub-views		Filters
	Name	Duration	Start	Finish	Predecessors	Resource Names	
1	Project Initialization	12 days?	26/08/20, 8:00 AM	10/09/20, 5:00 PM		Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
2	Identify Problem Stat	4 days?	26/08/20, 8:00 AM	31/08/20, 5:00 PM		Matte Mehvish;Almas Rahim;Shirgaonkar Iffat	
3	Markey Survey	7 days?	01/09/20, 8:00 AM	09/09/20, 5:00 PM	2	Almas Rahim[50%];Matte Mehvish[50%];Shirgaonkar Iffat[50%]	
4	Milestone: Topic Finali	1 day?	10/09/20, 8:00 AM	10/09/20, 5:00 PM	3	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
5	Requirement Gather	24 days?	11/09/20, 8:00 AM	14/10/20, 5:00 PM	1	Shirgaonkar Iffat[80%];Matte Mehvish[80%];Almas Rahim[80%]	
6	Discuss with clients	7 days?	11/09/20, 8:00 AM	21/09/20, 5:00 PM	4	Matte Mehvish;Almas Rahim;Shirgaonkar Iffat	
7	Prepare SRS	5 days?	22/09/20, 8:00 AM	28/09/20, 5:00 PM	6	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
8	Prepare Prototype	8 days?	29/09/20, 8:00 AM	08/10/20, 5:00 PM	7	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
9	Milestone: SRS approv	4 days?	09/10/20, 8:00 AM	14/10/20, 5:00 PM	8	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
10	Design	50.6 days?	15/10/20, 8:00 AM	24/12/20, 1:48 PM		Almas Rahim;Shirgaonkar Iffat;Matte Mehvish	
11	Prepare front end and	20 days?	15/10/20, 8:00 AM	11/11/20, 5:00 PM	9	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
12	Prepare test cases	7.6 days?	12/11/20, 8:00 AM	23/11/20, 1:48 PM	11	Shirgaonkar Iffat[50%];Matte Mehvish[50%];Almas Rahim[50%]	
13	Design Database	7 days?	23/11/20, 1:48 PM	02/12/20, 1:48 PM	12	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
14	Prepare UML diagram	6 days?	02/12/20, 1:48 PM	10/12/20, 1:48 PM	13	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
15	Milestone: UML approv	10 days?	10/12/20, 1:48 PM	24/12/20, 1:48 PM	14	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
16	Coding	67 days?	24/12/20, 1:48 PM	29/03/21, 1:48 PM	10	Almas Rahim;Matte Mehvish;Shirgaonkar Iffat	
17	Develop Front end	50 days?	24/12/20, 1:48 PM	04/03/21, 1:48 PM	15	Shirgaonkar Iffat;Matte Mehvish;Almas Rahim	
18	link with database	17 days?	04/03/21, 1:48 PM	29/03/21, 1:48 PM	17	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
19	Milestone: Code Appro	4 days?	24/12/20, 1:48 PM	30/12/20, 1:48 PM		Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
20	Testing	19 days?	29/03/21, 1:48 PM	23/04/21, 1:48 PM	16	Matte Mehvish;Almas Rahim;Shirgaonkar Iffat	
21	Unit testing	7 days?	29/03/21, 1:48 PM	07/04/21, 1:48 PM		Matte Mehvish[50%];Shirgaonkar Iffat[50%];Almas Rahim[50%]	
22	Integration testing	10 days?	29/03/21, 1:48 PM	12/04/21, 1:48 PM		Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
23	System Testing	7 days?	07/04/21, 1:48 PM	16/04/21, 1:48 PM	21	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
24	Alpha/Beta testing	5 days?	12/04/21, 1:48 PM	19/04/21, 1:48 PM	22	Matte Mehvish[80%];Shirgaonkar Iffat[80%];Almas Rahim[80%]	
25	Milestone: testing app	5 days?	16/04/21, 1:48 PM	23/04/21, 1:48 PM	23	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
26	Maintenance	20 days?	23/04/21, 1:48 PM	21/05/21, 1:48 PM	20	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	
27	Denlov Online	3 days?	23/04/21, 1:48 PM	28/04/21, 1:48 PM	25	Matte Mehvish;Shirgaonkar Iffat;Almas Rahim	

Figure 3.2: Project Task Assigned to members

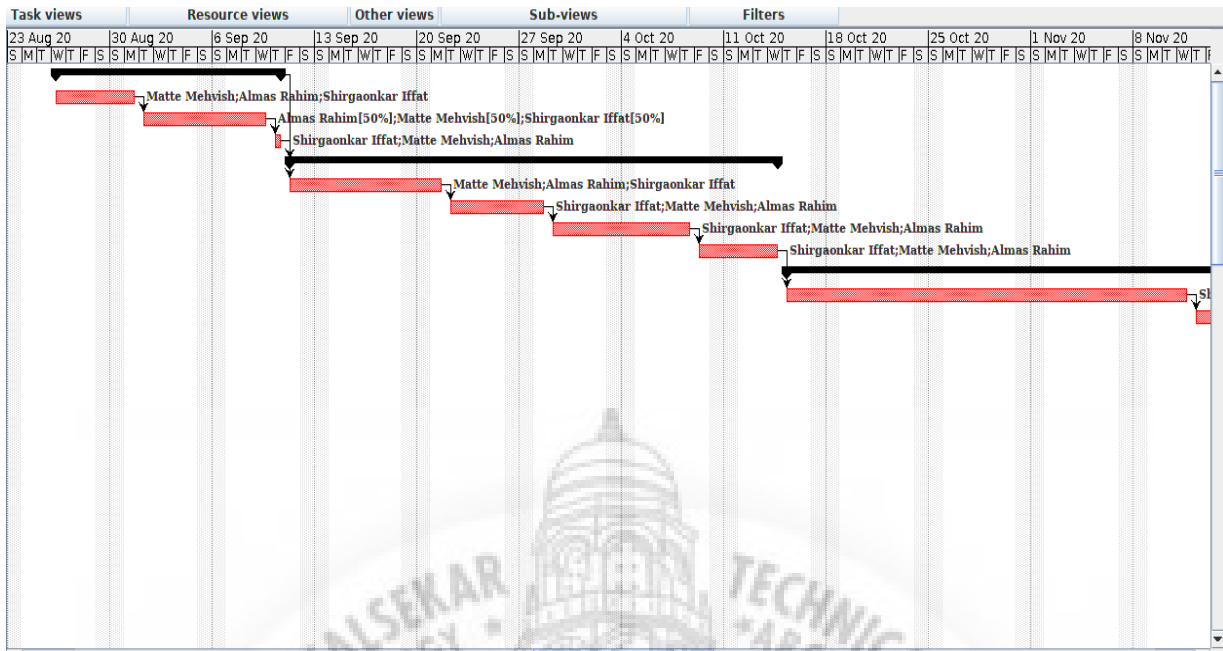


Figure 3.3: Project Task Assigned to members

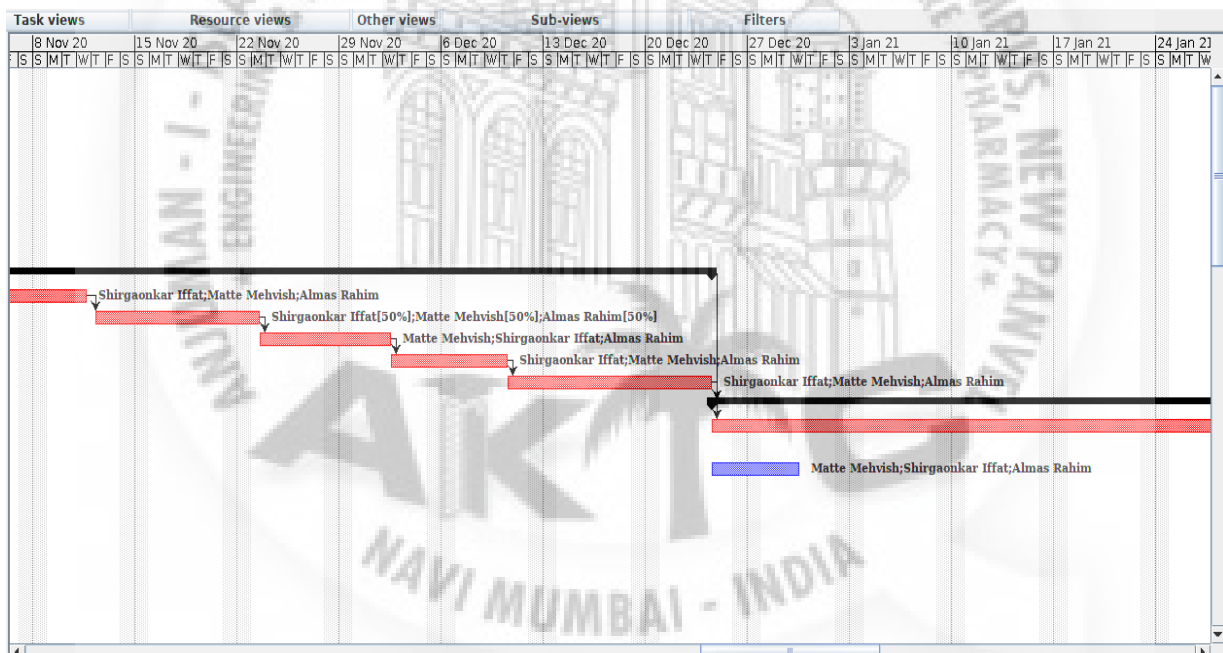


Figure 3.4: Project Task Assigned to members

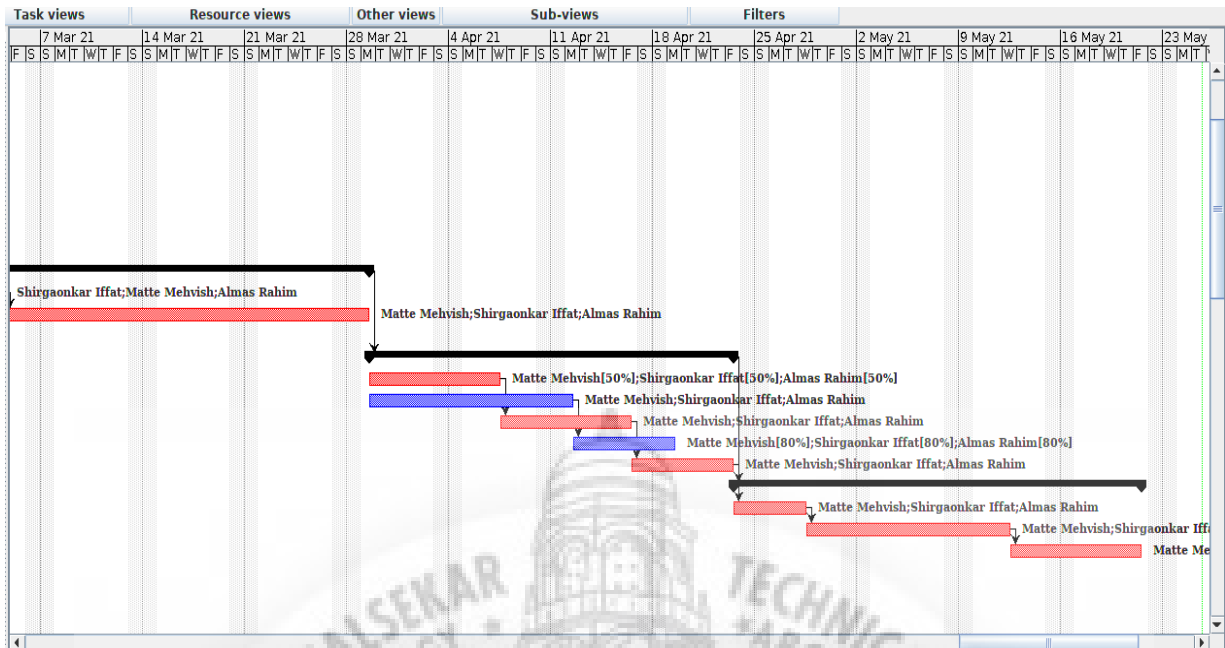


Figure 3.5: Project Task Assigned to members

Chapter 4

Software Requirements Specification

4.1 Overall Description

4.1.1 Product Perspective

the product is an Android application for Patient and Doctor side. The goal of this product is to provide an interactive application for the android market place. Medico is composed of two main components: A patient side application which will run on android handset and a doctor side application which will support and interact with various patient side features. The system is designed to provide features such as booking appointment. Consulting doctor, etc. The above proposed model is easy to implement considering the available technology infrastructure. The model is simple source and scalable.

4.1.2 Product Features

For patient user can select a doctor based on the requirement and rating of the doctor. And book a appointment with the doctor.

The patient can consult the doctor online through video calling and fan also make a payment online.

Patient can also upload his/her medical report and view the prescription uploaded by the doctor.

4.1.3 User Classes and Characteristics

The project is an android application for patient user and doctor user. User of project includes patient user and doctor user.

All the user should have knowledge of Internet and should have knowledge about how to use an android phone.

4.1.4 Operating Environment

1. Software Requirements:

- . Operating System: Linux or Windows (7 and above)
- . Android studio (Flutter)
- . Firebase

2. Hardware Requirements:

- . Processor i5
- . Memory 8GB
- . Hard Disk 1TB

3. Usage Requirements:

- . An android mobile API 19 and above (kitkat)
- . Internet Connectivity

4.1.5 Design and Implementation Constraints

The product is made using android studio hence, only android phone users can use this application m users may access the product using any android device m

The information of all the users, appointment histories, doctor data must be stored in database. Internet connectivity is the main source to use the product.

4.2 System Features

Booking appointment and Consulting a doctor online is main feature of the system. The patient has to choose a doctor by either searching specific doctor or as per requirement can choose from any specialists. And book an appointment with the doctor.

4.2.1 System Feature

Following are the mode of operation provided by the system:

For patient user: Register/login, Create his/her profile, choose a doctor as per requirement, book an appointment, view previous prescription or medical report.

For Doctor user: Login in the system, View the appointments, have a video call with patient, upload prescription.

Description and Priority

The highest priority is given to online consulting feature. As it is the main objective of our application.

Second prior feature is booking the appointment. Upon the booking of appointment, based on the time selected by the patient the consultation takes place.

Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary.

REQ-1: User are limited to android handset.

REQ-2: Access to Database.

REQ-3: Access to Internet.

4.3 External Interface Requirements**4.3.1 User Interfaces**

Android Provides a variety of pre-built UI components such as structured layout objects and UI controls that allow to build the graphical user interface for app. Android Provides other UI modules for special interface such as dialogues, navigation drawer and menu.

4.3.2 Hardware Interfaces

The application works in android handset with API 19 and above (kitkat) version.

4.3.3 Software Interfaces

Since this application is a mobile application, it will only need an Android KitKat version API 19 or higher in order to perform. Database is maintained in Firebase.

4.4 Nonfunctional Requirements**4.4.1 Performance Requirements**

Performance of overall system is very efficient and will optimize the time taken to show Doctor would take 3-5sec.

Process and everything is well organized.

The appointment booking will take approx 10-15sec.

4.4.2 Safety Requirements

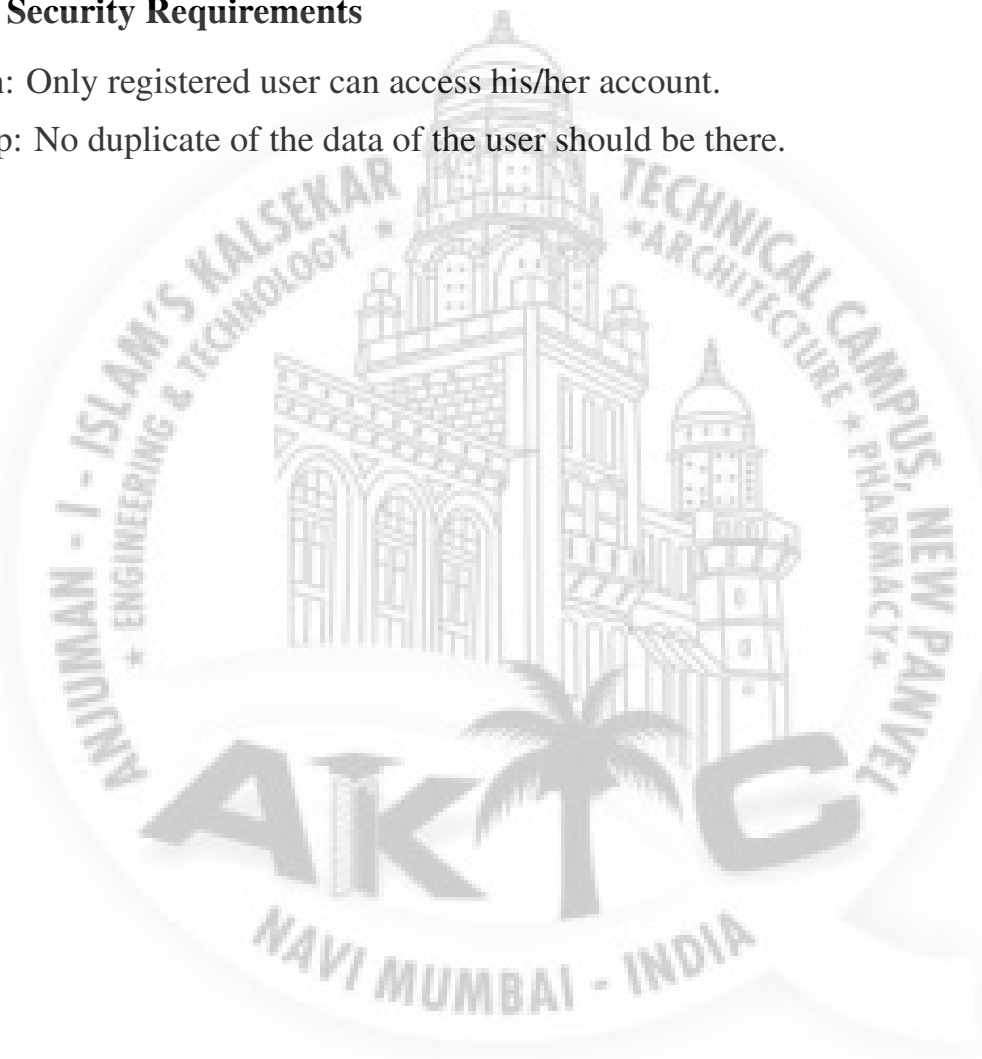
Login and sign-up must be authenticated for the pre-existing users.

Data of every user should be maintained.

4.4.3 Security Requirements

Sign-in: Only registered user can access his/her account.

Sign-up: No duplicate of the data of the user should be there.



Chapter 5

System Design

5.1 System Requirements Definition

5.1.1 Software Requirements

1. **Firestore:** Firestore Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firestore's cloud. The product assists software developers in building real-time, collaborative applications.

2. **Operating System (Linux 64-bit/Windows 64-bit):** An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

3. **Android Studio:** Android Studio provides a unified environment where developer can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow the developer to divide the project into units of functionality that developer can independently build, test, and debug.

5.1.2 Hardware Requirements

1. **Processor i5:** Core i5 processors provides improved performance for heavier usage needs. At the lower speeds, battery usage is pretty conservative and can reach up to five hours or usage on a single charge.

2. **Memory 8GB:** The minimal amount of RAM for running Android Studio is 8GB.

3. **Hard Disk 1TB:** The larger the hard drive, the more data and files it can store on it.

Data-flow Diagram

A data-flow diagram is a way of representing a flow of a data of a process or a system. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality. The physical data flow diagram describes the implementation of the logical data flow. The DFD also provides information about the outputs and inputs of each entity and the process itself. Given below is Level 0 and Level 1 system.

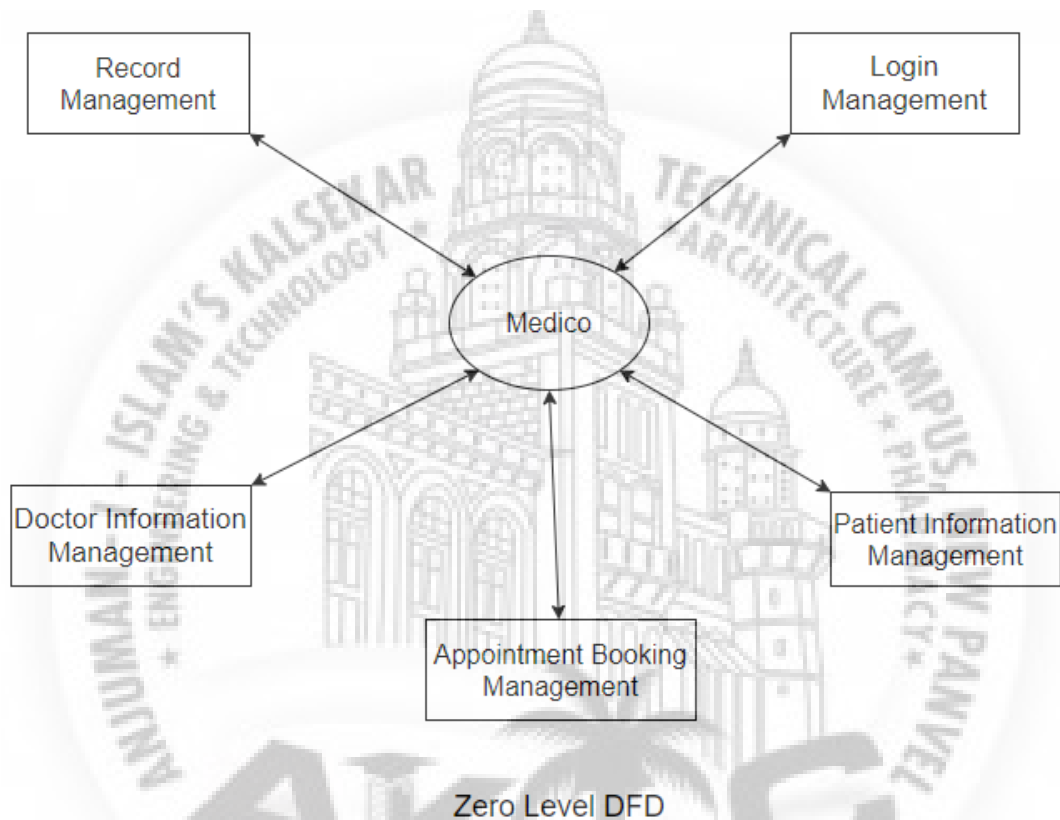


Figure 5.1: Zero Level DFD

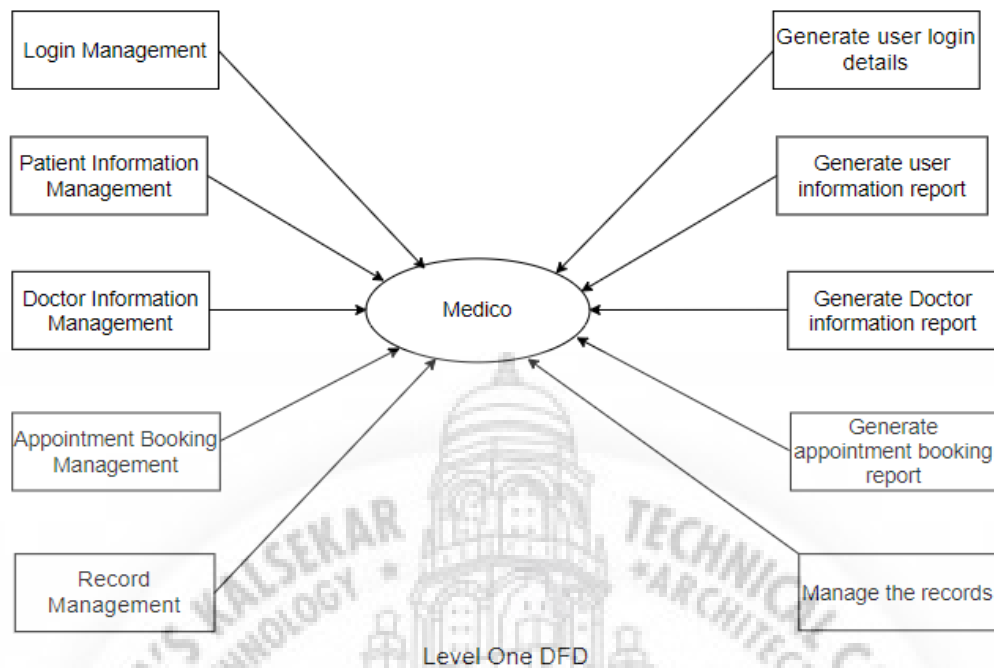


Figure 5.2: Level One DFD

5.1.3 System requirements (non-functional requirements)

These are non-functional system properties such as availability, performance and safety etc. They define functions of a system, services and operational constraints in detail.

- a. Usability - Application implementation is feasible using technologies that are accessible to the end-users.
- b. Portability - The interfaces are compatible with Web View and Mobile view
- c. Performance Efficiency - Application is able to perform well in a proper time constraint.
- d. Multi User System - Application is able to consider the presence of more than one user in the same environment. All the features of the system operates properly for all users and provides proper transparency.

5.2 System Architecture Design

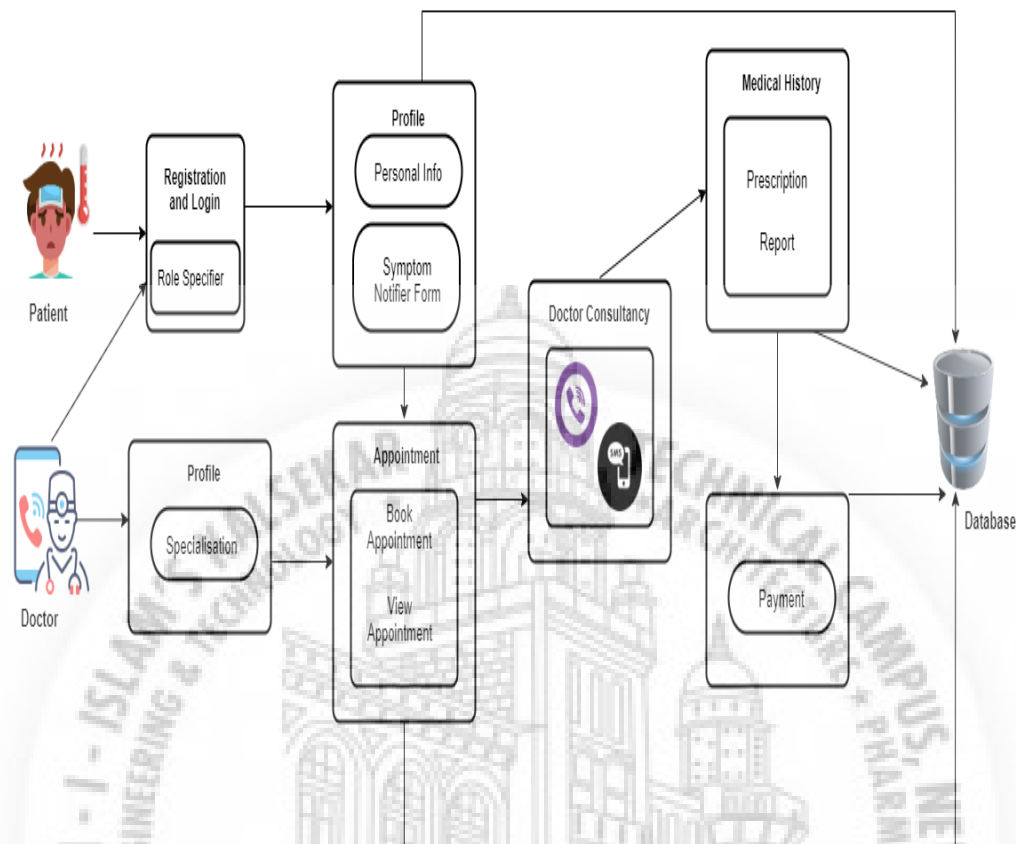


Figure 5.3: System Architecture Design

5.3 Sub-system Development

The system consist three user for each user we have user profile module for Patient side which consist of personal information such as Patient information, Patient prescription, Appointment history. User can select doctors, according to that patient can book appointment, view prescription after which they can make payment and upload medical history and can also view medical history. The doctor can view the patient's information, and also the appointment history and the patient's medical history. Functions performed in this module are: the doctor can view the appointments which are scheduled, view the prescription and also can send the prescription after checkup, can make video calls and view patients previous history.

Use case Diagram

Use case diagram are usually referred to as behaviour diagram used to describe a set of actions (use case) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

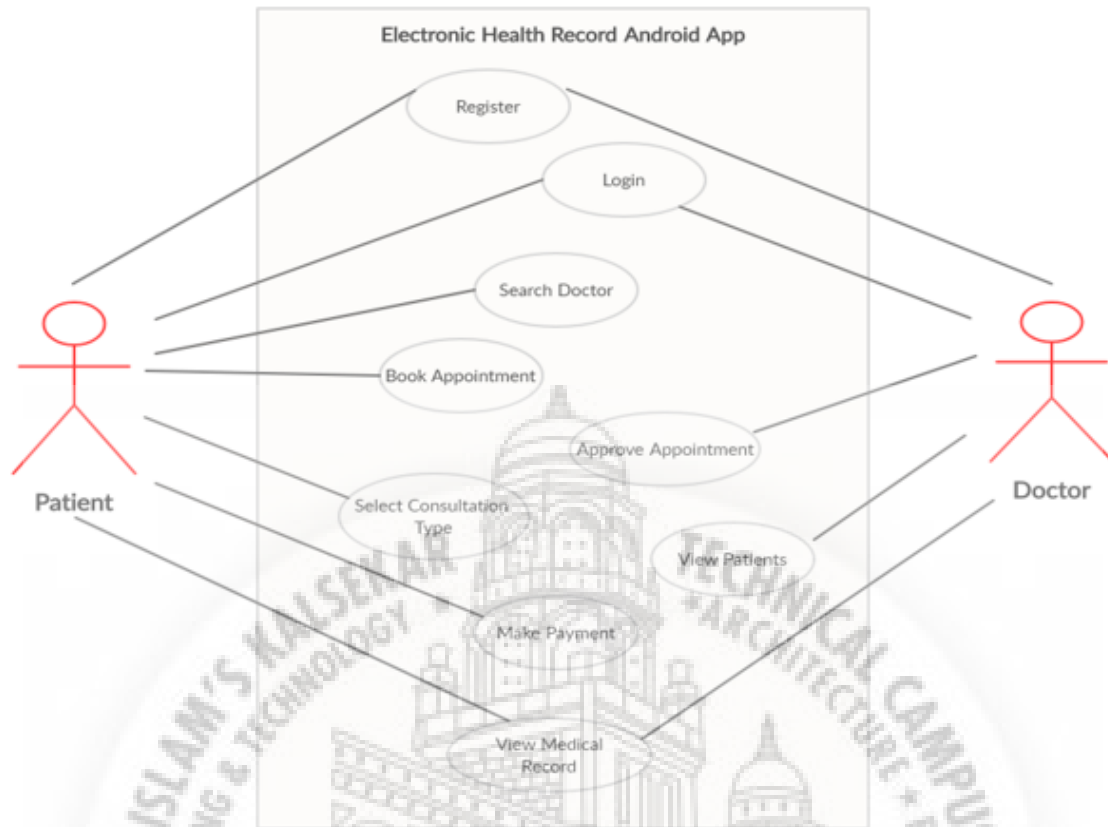


Figure 5.4: Use case Diagram

5.3.1 Patient Profile Module

The user will get registered themselves and then he/she will login, This consist of personal information such as Patient information, Patient prescription, Appointment history .

Patient Profile Module

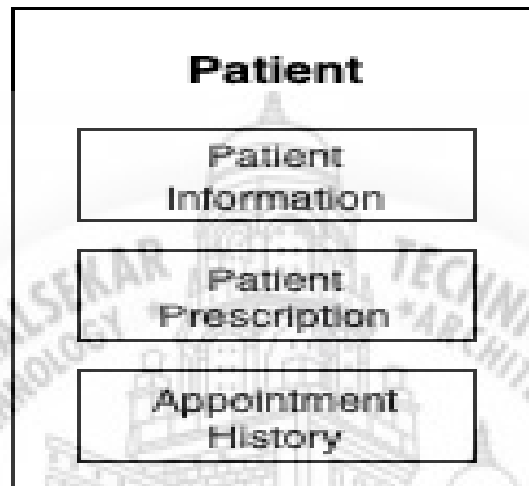


Figure 5.5: Patient Profile Module

Patient Function Module

User can select doctors, according to that patient can book appointment, view prescription after which they can make payment and upload medical history and can also view medical history.

Patient Function Module

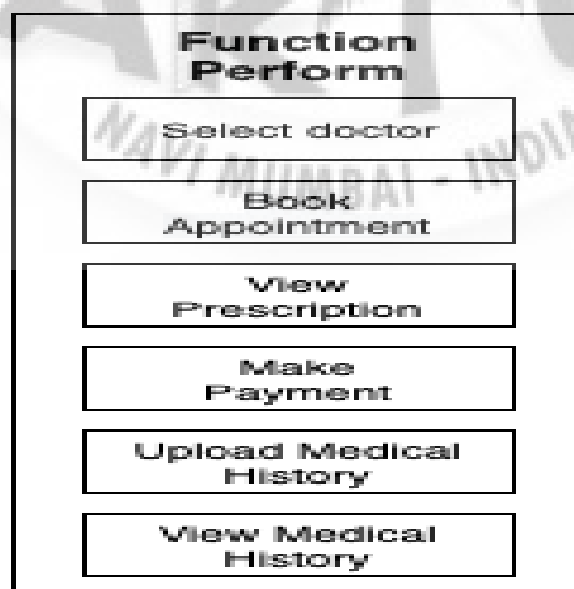


Figure 5.6: Patient function Module

5.3.2 Doctor Profile Module

The doctor can view the patient's information, and also the appointment history and the patient's medical history.

Doctor Profile Module

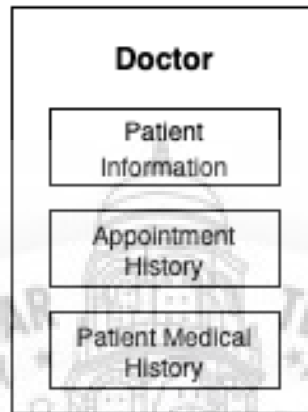


Figure 5.7: Doctor Profile Module

Doctor Function Module

Functions performed in this module are: the doctor can view the appointments which are scheduled, view the prescription and also can send the prescription after checkup, can make video calls and view patients previous history.

Doctor Function Module

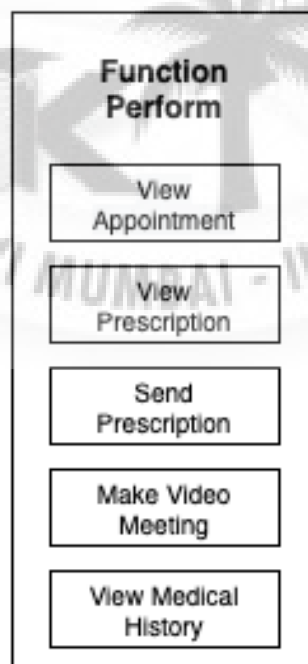
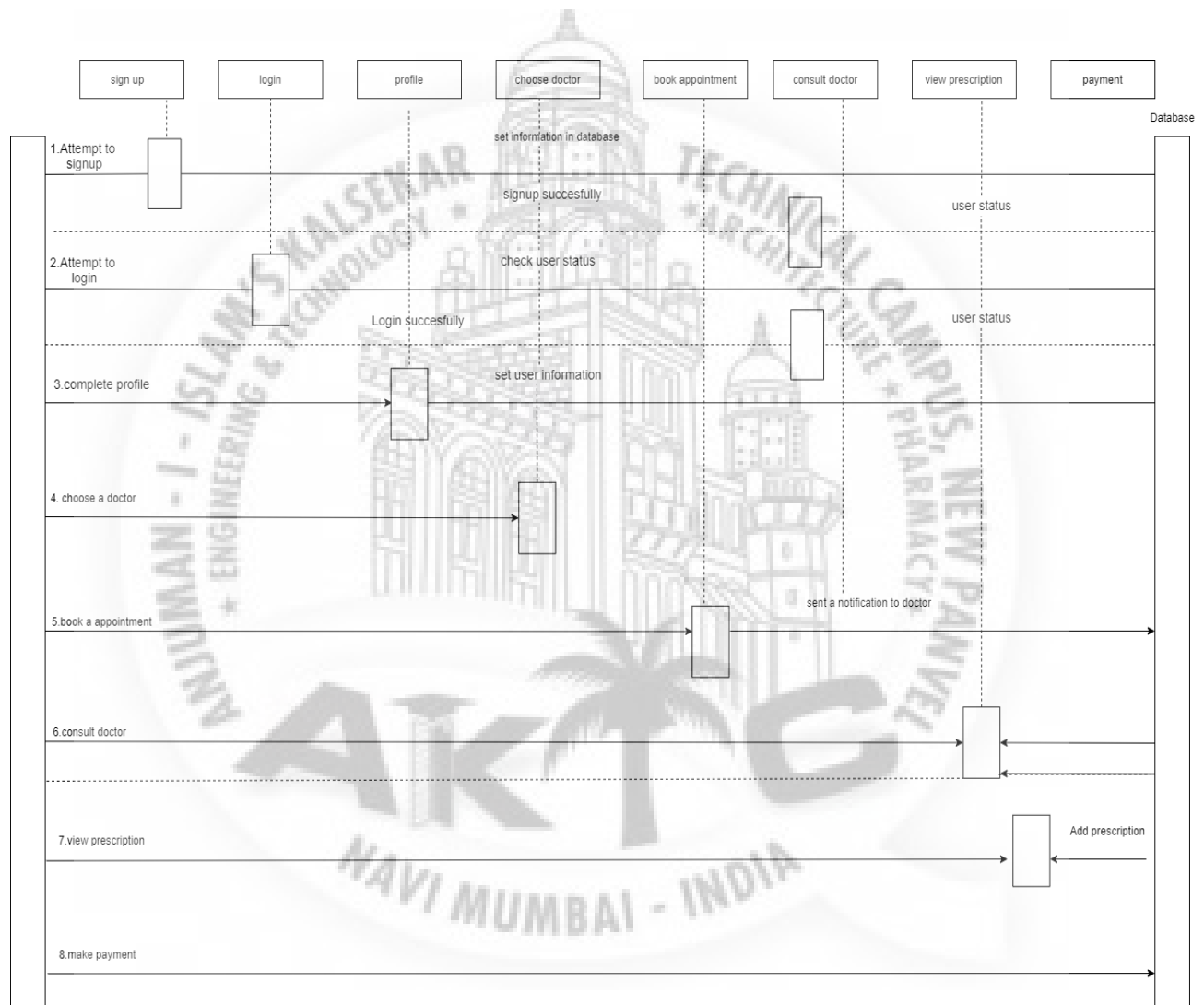


Figure 5.8: Doctor Function Module

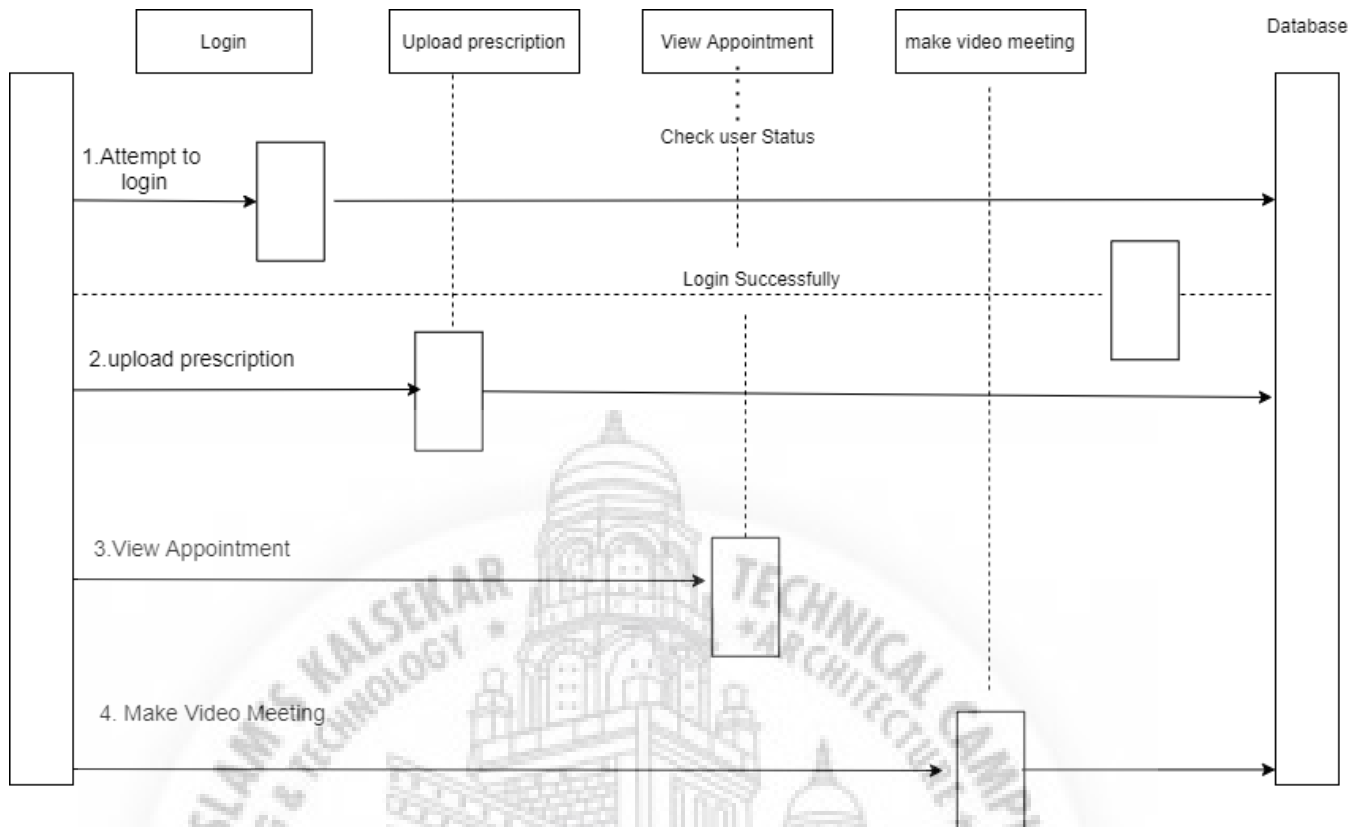
5.3.3 Sequence Diagram

A sequence diagram shows object interactions arranged in 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. The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams) and high-level interactions between user of the system and the system, between the system and other systems, or between subsystems.



Sequence Diagram of Patient

Figure 5.9: Sequence Diagram of Patient



Sequence Diagram of Doctor

Figure 5.10: Sequence Diagram of Patient

Chapter 6

Implementation

6.1 Doctor View

```

1 import 'dart:async';
2 import 'dart:ui';
3 import 'package:cloud_firestore/cloud_firestore.dart';
4 import 'package:flutter/painting.dart';
5 import 'package:google_fonts/google_fonts.dart';
6 import 'package:health_and_doctor_appointment/screens/prescription.dart';
7 import 'package:intl/intl.dart';
8 import 'package:firebase_auth/firebase_auth.dart';
9 import 'package:flutter/material.dart';
10 //import 'package:google_fonts/google_fonts.dart';
11
12 // ignore: camel_case_types 'assets/vector-doc2.jpg',
13
14 class Doctorpage extends StatefulWidget {
15   @override
16   _DoctorpageState createState() => _DoctorpageState();
17 }
18
19 TextEditingController comment = new TextEditingController();
20
21 class _DoctorpageState extends State<Doctorpage> {
22   FirebaseAuth _auth = FirebaseAuth.instance;
23   User user;
24   String _documentID;
25
26   Future<void> _getUser() async {
27     user = _auth.currentUser;
28   }
29
30   Future _signOut() async {
31     await _auth.signOut();
32   }
33   //fetch data frm fb
34
35   Future<void> deleteAppointment(String docID) {
36     return FirebaseFirestore.instance
37       .collection('appointments')
38       .doc("test@mail.com")
39       .collection('pending')
40       .doc(docID)
41       .delete();
42   }
43   // to show the app per dr

```

```

44 Future<void> deleteAppointmentDoc(String docID) {
45     return FirebaseFirestore.instance
46         .collection("appointments-doc")
47         .where("docid", isEqualTo: docID)
48         .get()
49         .then((value) {
50     value.docs.forEach((element) {
51         FirebaseFirestore.instance
52             .collection("appointments-doc")
53             .doc(element.id)
54             .delete()
55             .then((value) {
56                 print("Success!");
57             });
58     });
59 });
60 }
61
62 String _dateFormatter(String _timestamp) {
63     String formattedDate =
64         DateFormat('dd-MM-yyyy').format(DateTime.parse(_timestamp));
65     return formattedDate;
66 }
67
68 String _timeFormatter(String _timestamp) {
69     String formattedTime =
70         DateFormat('kk:mm').format(DateTime.parse(_timestamp));
71     return formattedTime;
72 }
73
74 showAlertDialog(BuildContext context) {
75     // set up the buttons
76     Widget cancelButton = TextButton(
77         child: Text("No"),
78         onPressed: () {
79             Navigator.of(context).pop();
80         },
81     );
82     Widget continueButton = TextButton(
83         child: Text("Yes"),
84         onPressed: () {
85             deleteAppointment(_documentID);
86             Navigator.of(context).pop();
87         },
88     );
89
90     // set up the AlertDialog
91     AlertDialog alert = AlertDialog(
92         title: Text("Complete Appointment"),
93         content: Text("Are you sure you want to delete this Appointment?"),
94         actions: [
95             cancelButton,
96             continueButton,
97         ],
98     );
99
100     // show the dialog
101     showDialog(
102         context: context,
103         builder: (BuildContext context) {
104             return alert;

```

```

105     },
106   );
107 }
108 // delete after 2 hrs
109 _checkDiff(DateTime _date) {
110   var diff = DateTime.now().difference(_date).inHours;
111   if (diff > 2) {
112     return true;
113   } else {
114     return false;
115   }
116 }
117
118 _compareDate(String _date) {
119   if (_dateFormatter(DateTime.now().toString())
120       .compareTo(_dateFormatter(_date)) ==
121       0) {
122     return true;
123   } else {
124     return false;
125   }
126 }
127
128 @override
129 void initState() {
130   super.initState();
131   _getUser();
132 }
133
134 @override
135 Widget build(BuildContext context) {
136   return Scaffold(
137     backgroundColor: Colors.white,
138     appBar: AppBar(
139       actions: [
140         PopupMenuButton(
141           itemBuilder: (BuildContext bc) => [
142             PopupMenuItem(child: Text("All Appointments"), value: "prev"),
143             PopupMenuItem(child: Text("Sign out"), value: "/login"),
144           ],
145           onSelect: (route) {
146             print(route);
147
148             // Note You must create respective pages for navigation
149             if (route == '/login') {
150               Navigator.pushNamedAndRemoveUntil(
151                 context, route, ModalRoute.withName('/login'));
152               _signOut();
153             } else if (route == 'prev') {
154               Navigator.pushNamed(context, '/prevDoctor');
155             }
156           },
157         ),
158       ],
159     backgroundColor: Colors.white,
160     title: Container(
161       padding: EdgeInsets.symmetric(vertical: 10),
162       child: Text(
163         'Welcome',
164         style: GoogleFonts.lato(
165           color: Colors.black,

```

```

166         fontWeight: FontWeight.bold,
167     ),
168 ),
169 ),
170     iconTheme: IconThemeData(color: Colors.black),
171 ),
172     body: SafeArea(
173     child: Column(
174     children: [
175     Container(
176     padding: const EdgeInsets.only(
177     left: 15,
178     top: 20,
179     ),
180     child: Row(
181     children: [
182     Icon(
183     Icons.account_circle_outlined,
184     size: 40,
185     ),
186     SizedBox(
187     width: 10,
188     ),
189     Text(
190     user.displayName,
191     style: GoogleFonts.lato(
192     fontSize: 18, fontWeight: FontWeight.bold),
193     )
194     ],
195     ),
196     ),
197     Container(
198     alignment: Alignment.centerLeft,
199     padding: EdgeInsets.only(top: 20, left: 15),
200     child: Text(
201     "Your Appointments,",
202     style: GoogleFonts.lato(
203     fontSize: 18,
204     fontWeight: FontWeight.bold,
205     ),
206     ),
207     ),
208     ),
209     Padding(
210     padding: const EdgeInsets.all(8.0),
211     child: StreamBuilder(
212     stream: FirebaseFirestore.instance
213     .collection('appointments-doc')
214     .where('email', isEqualTo: user.email)
215     .snapshots(),
216     builder: (BuildContext context,
217     AsyncSnapshot<QuerySnapshot> snapshot) {
218     if (!snapshot.hasData) {
219     return Center(
220     child: CircularProgressIndicator(),
221     );
222     }
223     return snapshot.data.size == 0
224     ? Center(
225     child: Text(
226     'No Appointment Scheduled',

```

```

227         style: GoogleFonts.lato(
228             color: Colors.grey,
229             fontSize: 18,
230         ),
231     ),
232 )
233 : ListView.builder(
234     scrollDirection: Axis.vertical,
235     physics: ClampingScrollPhysics(),
236     shrinkWrap: true,
237     itemCount: snapshot.data.size,
238     itemBuilder: (context, index) {
239         DocumentSnapshot document =
240             snapshot.data.docs[index];
241
242         print(_checkDiff(document['date'].toDate()));
243         if (_checkDiff(document['date'].toDate())) {
244             deleteAppointment(document.id);
245             deleteAppointmentDoc(document.id);
246         }
247         return Card(
248             elevation: 2,
249             child: InkWell(
250                 onTap: () {},
251                 child: ExpansionTile(
252                     title: Row(
253                         mainAxisAlignment:
254                             MainAxisAlignment.spaceBetween,
255                         children: [
256                             Padding(
257                                 padding:
258                                     const EdgeInsets.only(left: 5),
259                                 child: Text(
260                                     document['name'],
261                                     style: GoogleFonts.lato(
262                                         fontSize: 16,
263                                         fontWeight: FontWeight.bold,
264                                     ),
265                                 ),
266                             ),
267                             Text(
268                                 _compareDate(document['date']
269                                     .toDate()
270                                     .toString())
271                                     ? "TODAY"
272                                     : "",
273                                 style: GoogleFonts.lato(
274                                     color: Colors.green,
275                                     fontSize: 18,
276                                     fontWeight: FontWeight.bold),
277                             ),
278                             SizedBox(
279                                 width: 0,
280                             ),
281                         ],
282                     ),
283                     subtitle: Padding(
284                         padding: const EdgeInsets.only(left: 5),
285                         child: Text(
286                             _dateFormatter(document['date']
287                                 .toDate()

```

```

288         .toString()),
289         style: GoogleFonts.lato(),
290     ),
291 ),
292 children: [
293     Padding(
294         padding: const EdgeInsets.only(
295             bottom: 20, right: 10, left: 16),
296         child: Row(
297             mainAxisAlignment:
298                 MainAxisAlignment.spaceBetween,
299             children: [
300                 Column(
301                     crossAxisAlignment:
302                         CrossAxisAlignment.start,
303                     children: [
304                         Row(
305                             children: [
306                                 Text(
307                                     "Description: ",
308                                     style: GoogleFonts.lato(
309                                         fontSize: 16,
310                                         fontWeight:
311                                             FontWeight.w600),
312                                 ),
313                                 Container(
314                                     width:
315                                         MediaQuery.of(context)
316                                             .size
317                                             .width /
318                                             3,
319                                     child: Text(
320                                         document['description']
321                                             ==
322                                             ""
323                                             ? "Not Available"
324                                             : document[
325                                                 'description'],
326                                         style: GoogleFonts.lato(
327                                             fontSize: 16,
328                                         ),
329                                     ),
330                                 ],
331                             ),
332                             SizedBox(
333                                 height: 10,
334                             ),
335                             Text(
336                                 "Time: " +
337                                     _timeFormatter(
338                                         document['date']
339                                             .toDate()
340                                             .toString(),
341                                     ),
342                                 style: GoogleFonts.lato(
343                                     fontSize: 16,
344                                 ),
345                             ),
346                         ],
347                     ),

```

```

348     IconButton(
349         tooltip: 'Add Prescription',
350         icon: Icon(
351             Icons.pending_actions_outlined,
352             color: Colors.black87,
353         ),
354         onPressed: () {
355             Navigator.push(
356                 context,
357                 MaterialPageRoute(
358                     builder: (context) =>
359                     Prescription(
360                         useremail: document[
361                             'useremail'],
362                         name: document[
363                             'name'],
364                         docName: document[
365                             'doctor'],
366                         time: document[
367                             'date'],
368                         docID: document[
369                             'docid'],
370                     )),
371             );
372         },
373     ),
374     ),
375     ),
376     ),
377     ),
378     ),
379     ),
380     ),
381     ),
382     ),
383     ),
384     ),
385     ),
386     ),
387     ),
388     ));
389 }
390 }

```


6.2 Prescription by Doctor

```

1  import 'package:cloud_firestore/cloud_firestore.dart';
2  import 'package:firebase_auth/firebase_auth.dart';
3  import 'package:flutter/material.dart';
4  import 'package:google_fonts/google_fonts.dart';
5  import 'package:health_and_doctor_appointment/screens/Doctorpage.dart';
6  import 'package:health_and_doctor_appointment/screens/prevPrescription.dart';
7  import 'package:health_and_doctor_appointment/screens/seeMedicalRecord.dart';
8  import 'package:health_and_doctor_appointment/screens/videoCall.dart';
9  import 'package:health_and_doctor_appointment/screens/videoCallDoctor.dart';
10
11 class Prescription extends StatefulWidget {
12   final useremail;
13   final name;
14   final docName;
15   final time;
16   final docID;
17
18   const Prescription(
19     {Key key, this.useremail, this.name, this.docName, this.time, this.docID})
20     : super(key: key);
21   @override
22   _PrescriptionState createState() => _PrescriptionState();
23 }
24
25 class _PrescriptionState extends State<Prescription> {
26   final TextEditingController _controller = TextEditingController();
27   final TextEditingController _controllerFee = TextEditingController();
28   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
29
30   FirebaseAuth _auth = FirebaseAuth.instance;
31   User user;
32
33   Future<void> _getUser() async {
34     user = _auth.currentUser;
35   }
36
37   @override
38   void initState() {
39     super.initState();
40     _getUser();
41   }
42
43   showAlertDialog(BuildContext context) {
44     // set up the button
45     Widget okButton = TextButton(
46       child: Text(
47         "OK",
48         style: GoogleFonts.lato(fontWeight: FontWeight.bold),
49       ),
50       onPressed: () {
51         Navigator.pushReplacement(
52           context,
53           MaterialPageRoute(
54             builder: (context) => Doctorpage(),
55           ),
56         );
57       },
58     );
59

```

```

60 // set up the AlertDialog
61 AlertDialog alert = AlertDialog(
62     title: Text(
63         "Done!",
64         style: GoogleFonts.lato(
65             fontWeight: FontWeight.bold,
66         ),
67     ),
68     content: Text(
69         "Prescription added successfully.",
70         style: GoogleFonts.lato(),
71     ),
72     actions: [
73         okButton,
74     ],
75 );
76
77 // show the dialog
78 showDialog(
79     context: context,
80     builder: (BuildContext context) {
81         return alert;
82     },
83 );
84 }
85
86 Future<void> addPrescription() async {
87     FirebaseFirestore.instance
88         .collection('appointments')
89         .doc(widget.useremail)
90         .collection('prescriptions')
91         .doc()
92         .set({
93             'prescription': _controller.text,
94             'fees': _controllerFee.text,
95             'docID': widget.docID,
96             'doctor': user.displayName,
97             'time': Timestamp.now()
98         }, SetOptions(merge: true));
99
100     showAlertDialog(context);
101 }
102
103 @override
104 Widget build(BuildContext context) {
105     var meetName = widget.name;
106     meetName = widget.name.replaceAll(RegExp('\\s+'), '-');
107
108     print(Timestamp.now());
109
110     return Scaffold(
111         appBar: AppBar(
112             title: Text("Prescription",
113                 style: GoogleFonts.lato(
114                     color: Colors.black, fontWeight: FontWeight.bold)),
115             backgroundColor: Colors.white,
116             iconTheme: IconThemeData(color: Colors.black),
117         ),
118         body: SingleChildScrollView(
119             child: Padding(
120                 padding: const EdgeInsets.all(15.0),

```

```

121     child: Column(
122       children: [
123         Container(
124           child: StreamBuilder(
125             stream: FirebaseFirestore.instance
126               .collection('appointments')
127               .doc(widget.useremail)
128               .collection('prescriptions')
129               .orderBy('docID')
130               .startAt([widget.docID]).endAt(
131                 [widget.docID + '\uf8ff']).snapshots(),
132             builder: (BuildContext context,
133               AsyncSnapshot<QuerySnapshot> snapshot) {
134               if (!snapshot.hasData) return Container();
135               if (snapshot.data.size != 0) {
136                 return Container(
137                   child: TextButton(
138                     onPressed: () {
139                       Navigator.push(
140                         context,
141                         MaterialPageRoute(
142                           builder: (context) => PrevPrescription(
143                             email: widget.useremail,
144                             docID: widget.docID,
145                           ),
146                         ),
147                       );
148                     },
149                     child: Text(
150                       "See Previous Prescriptions",
151                       style: GoogleFonts.lato(
152                         fontSize: 16,
153                         fontWeight: FontWeight.bold,
154                       ),
155                     ),
156                   ),
157                 );
158               } else {
159                 return Container();
160               }
161             },
162           ),
163         ),
164         Container(
165           margin: EdgeInsets.only(top: 10),
166           child: Form(
167             key: _formKey,
168             child: Column(
169               crossAxisAlignment: CrossAxisAlignment.start,
170               children: [
171                 Text(
172                   "Patient Name",
173                   style: GoogleFonts.lato(
174                     fontSize: 18, fontWeight: FontWeight.bold),
175                 ),
176                 Container(
177                   alignment: Alignment.centerLeft,
178                   margin: EdgeInsets.only(top: 10),
179                   width: MediaQuery.of(context).size.width,
180                   height: 40,
181                   color: Colors.grey[200],

```

```

182         child: Text(
183             widget.name,
184             style: GoogleFonts.lato(fontSize: 18),
185         ),
186     ),
187     TextButton(
188         onPressed: () {
189             Navigator.push(
190                 context,
191                 MaterialPageRoute(
192                     builder: (context) => SeeMedicalRecord(
193                         email: widget.useremail,
194                     )),
195             );
196         },
197         child: Text(
198             "See Patient's Medical Record",
199             style: GoogleFonts.lato(
200                 fontWeight: FontWeight.bold,
201             ),
202         ),
203     ),
204     SizedBox(
205         height: 10,
206     ),
207     Text(
208         "Time",
209         style: GoogleFonts.lato(
210             fontSize: 18, fontWeight: FontWeight.bold),
211     ),
212     Container(
213         alignment: Alignment.centerLeft,
214         margin: EdgeInsets.only(top: 10),
215         width: MediaQuery.of(context).size.width,
216         height: 40,
217         color: Colors.grey[200],
218         child: Text(
219             widget.time.toDate().toString(),
220             style: GoogleFonts.lato(fontSize: 18),
221         ),
222     ),
223     SizedBox(
224         height: 25,
225     ),
226     Text(
227         "Add Prescription",
228         style: GoogleFonts.lato(
229             fontSize: 18, fontWeight: FontWeight.bold),
230     ),
231     TextFormField(
232         controller: _controller,
233         keyboardType: TextInputType.multiline,
234         maxLines: 5,
235         style: GoogleFonts.lato(
236             fontSize: 18, fontWeight: FontWeight.bold),
237         decoration: InputDecoration(
238             contentPadding:
239                 EdgeInsets.only(left: 10, top: 10, bottom: 10),
240             border: OutlineInputBorder(
241                 borderSide: BorderSide.none,
242             ),

```

```

243         filled: true ,
244         fillColor: Colors.grey[200],
245         hintText: 'Add Prescription' ,
246         hintStyle: GoogleFonts.lato(
247             color: Colors.black26 ,
248             fontSize: 18,
249             fontWeight: FontWeight.w800,
250         ),
251     ),
252     textInputAction: TextInputAction.next ,
253     validator: (value) {
254         if (value.isEmpty)
255             return 'Please Add the Prescription';
256         return null;
257     },
258 ),
259 SizedBox(
260     height: 25,
261 ),
262 Text(
263     "Fees" ,
264     style: GoogleFonts.lato(
265         fontSize: 18, fontWeight: FontWeight.bold),
266 ),
267 TextFormField(
268     controller: _controllerFee ,
269     keyboardType: TextInputType.number ,
270     style: GoogleFonts.lato(
271         fontSize: 18, fontWeight: FontWeight.bold),
272     decoration: InputDecoration(
273         contentPadding:
274             EdgeInsets.only(left: 10, top: 10, bottom: 10),
275         border: OutlineInputBorder(
276             borderSide: BorderSide.none ,
277         ),
278         filled: true ,
279         fillColor: Colors.grey[200],
280         hintText: 'Add fees in ' ,
281         hintStyle: GoogleFonts.lato(
282             color: Colors.black26 ,
283             fontSize: 18,
284             fontWeight: FontWeight.w800,
285         ),
286     ),
287     textInputAction: TextInputAction.next ,
288     validator: (value) {
289         if (value.isEmpty) return 'Please Add the Fees';
290         return null;
291     },
292 ),
293 SizedBox(
294     height: 25,
295 ),
296 Container(
297     width: MediaQuery.of(context).size.width ,
298     decoration: BoxDecoration(
299         color: Colors.blue[900],
300         borderRadius: BorderRadius.circular(12)),
301     child: TextButton(
302         onPressed: () {
303             if (_formKey.currentState.validate()) {

```

```

304         addPrescription ();
305     }
306 },
307 child: Text(
308     "Add Prescription",
309     style: GoogleFonts.lato(
310         color: Colors.white,
311         fontSize: 16,
312         fontWeight: FontWeight.bold,
313     ),
314 ),
315 ),
316 ),
317 SizedBox(
318     height: 15,
319 ),
320 Divider(
321     color: Colors.grey,
322 ),
323 SizedBox(
324     height: 15,
325 ),
326 Container(
327     width: MediaQuery.of(context).size.width,
328     height: 50,
329     decoration: BoxDecoration(
330         borderRadius: BorderRadius.circular(12),
331     ),
332     child: MaterialButton(
333         shape: RoundedRectangleBorder(
334             borderRadius: new BorderRadius.circular(12.0)),
335         onPressed: () {
336             Navigator.push(
337                 context,
338                 MaterialPageRoute(
339                     builder: (context) => MeetingDoctor(
340                         roomID: meetName,
341                     )),
342             );
343         },
344         color: Colors.blue[900],
345         child: Text(
346             "Connect Video Call",
347             style: GoogleFonts.lato(
348                 color: Colors.white,
349                 fontSize: 16,
350                 fontWeight: FontWeight.w600),
351         ),
352     ),
353 ),
354 SizedBox(
355     height: 25,
356 ),
357 ),
358 ),
359 ),
360 ),
361 ),
362 ),
363 }
364 }

```

6.3 Patient View

```

1  import 'dart:ui';
2  import 'package:cloud_firestore/cloud_firestore.dart';
3  import 'package:firebase_auth/firebase_auth.dart';
4  import 'package:flutter/cupertino.dart';
5  import 'package:flutter/material.dart';
6  import 'package:google_fonts/google_fonts.dart';
7  import 'package:flutter_icons/flutter_icons.dart';
8  import 'package:health_and_doctor_appointment/firestore-data/
    appointmentHistoryList.dart';
9  import 'package:health_and_doctor_appointment/screens/prevMedRecords.dart';
10 import 'package:health_and_doctor_appointment/screens/prevPrescriptionUser.dart'
    ;
11 import 'package:health_and_doctor_appointment/screens/userSettings.dart';
12
13 class UserProfile extends StatefulWidget {
14   const UserProfile({Key key}) : super(key: key);
15
16   @override
17   _UserProfileState createState() => _UserProfileState();
18 }
19
20 class _UserProfileState extends State<UserProfile> {
21   final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();
22   FirebaseAuth _auth = FirebaseAuth.instance;
23   User user;
24
25   Future<void> _getUser() async {
26     user = _auth.currentUser;
27   }
28
29   @override
30   void initState() {
31     super.initState();
32     _getUser();
33   }
34
35   @override
36   Widget build(BuildContext context) {
37     return Scaffold(
38       body: SafeArea(
39         child: NotificationListener<OverscrollIndicatorNotification>(
40           onNotification: (OverscrollIndicatorNotification overscroll) {
41             overscroll.disallowGlow();
42             return;
43           },
44           child: ListView(
45             physics: ClampingScrollPhysics(),
46             shrinkWrap: true,
47             children: <Widget>[
48               Stack(
49                 alignment: Alignment.center,
50                 children: <Widget>[
51                   Column(

```



```

52     children : [
53         Container(
54             decoration : BoxDecoration(
55                 gradient : LinearGradient(
56                     begin : Alignment.topCenter ,
57                     end : Alignment.bottomCenter ,
58                     stops : [0.1, 0.5],
59                     colors : [
60                         Colors.indigo ,
61                         Colors.indigoAccent ,
62                     ],
63                 ),
64             ),
65             height : MediaQuery.of(context).size.height / 5,
66             child : Container(
67                 padding : EdgeInsets.only(top : 10, right : 7),
68                 alignment : Alignment.topRight ,
69                 child : IconButton(
70                     icon : Icon(
71                         FlutterIcons.gear_faw ,
72                         color : Colors.white ,
73                         size : 20,
74                     ),
75                     onPressed : () {
76                         Navigator.push(
77                             context ,
78                             MaterialPageRoute(
79                                 builder : (context) => UserSettings() ,
80                             ),
81                         );
82                     },
83                 ),
84             ),
85         ),
86         Container(
87             alignment : Alignment.center ,
88             height : MediaQuery.of(context).size.height / 5,
89             padding : EdgeInsets.only(top : 75),
90             child : Text(
91                 user.displayName ,
92                 style : GoogleFonts.lato(
93                     fontSize : 25,
94                     fontWeight : FontWeight.bold ,
95                 ),
96             ),
97         ),
98     ],
99 ),
100 Container(
101     child : CircleAvatar(
102         radius : 80,
103         backgroundColor : Colors.white ,
104         backgroundImage : AssetImage('assets/person.jpg') ,
105     ),
106     decoration : BoxDecoration(
107         border : Border.all(
108             color : Colors.teal[50],
109             width : 5,
110         ),
111         shape : BoxShape.circle),
112 ),

```

```

113     ],
114     ),
115     Container(
116         margin: EdgeInsets.only(left: 15, right: 15),
117         padding: EdgeInsets.only(left: 20),
118         height: MediaQuery.of(context).size.height / 7,
119         width: MediaQuery.of(context).size.width,
120         decoration: BoxDecoration(
121             borderRadius: BorderRadius.circular(10),
122             color: Colors.blueGrey[50],
123         ),
124         child: Column(
125             mainAxisAlignment: MainAxisAlignment.center,
126             children: <Widget>[
127                 Row(
128                     crossAxisAlignment: CrossAxisAlignment.center,
129                     children: [
130                         ClipRect(
131                             borderRadius: BorderRadius.circular(30),
132                             child: Container(
133                                 height: 27,
134                                 width: 27,
135                                 color: Colors.red[900],
136                                 child: Icon(
137                                     Icons.mail_rounded,
138                                     color: Colors.white,
139                                     size: 16,
140                                 ),
141                             ),
142                         ),
143                         SizedBox(
144                             width: 10,
145                         ),
146                         Text(
147                             user.email,
148                             style: GoogleFonts.lato(
149                                 fontSize: 16,
150                                 fontWeight: FontWeight.w600,
151                                 color: Colors.black54,
152                             ),
153                         ),
154                     ],
155                 ),
156                 SizedBox(
157                     height: 15,
158                 ),
159                 Row(
160                     crossAxisAlignment: CrossAxisAlignment.center,
161                     children: [
162                         ClipRect(
163                             borderRadius: BorderRadius.circular(30),
164                             child: Container(
165                                 height: 27,
166                                 width: 27,
167                                 color: Colors.blue[800],
168                                 child: Icon(
169                                     Icons.phone,
170                                     color: Colors.white,
171                                     size: 16,
172                                 ),
173                             ),

```

```

174         ),
175         SizedBox(
176             width: 10,
177         ),
178         Text(
179             user?.phoneNumber?.isEmpty ?? true
180                 ? "Not Added"
181                 : user.phoneNumber,
182             style: GoogleFonts.lato(
183                 fontSize: 16,
184                 fontWeight: FontWeight.w600,
185                 color: Colors.black54,
186             ),
187         ),
188     ],
189 ),
190 ],
191 ),
192 ),
193 Container(
194     margin: EdgeInsets.only(left: 15, right: 15, top: 20),
195     padding: EdgeInsets.only(left: 20, top: 20, bottom: 20),
196     width: MediaQuery.of(context).size.width,
197     decoration: BoxDecoration(
198         borderRadius: BorderRadius.circular(10),
199         color: Colors.blueGrey[50],
200     ),
201     child: InkWell(
202         onTap: () {
203             Navigator.push(
204                 context,
205                 MaterialPageRoute(builder: (context) => PrevMedRecords()),
206             );
207         },
208         child: Row(
209             crossAxisAlignment: CrossAxisAlignment.start,
210             children: [
211                 Text(
212                     "See Previous Medical Records",
213                     style: GoogleFonts.lato(
214                         fontSize: 16,
215                         fontWeight: FontWeight.bold,
216                         color: Colors.black,
217                     ),
218                 ),
219                 SizedBox(
220                     width: 5,
221                 ),
222                 Icon(Icons.chevron_right_rounded),
223             ],
224         ),
225     ),
226 ),
227 Container(
228     margin: EdgeInsets.only(left: 15, right: 15, top: 20),
229     padding: EdgeInsets.only(left: 20, top: 20, bottom: 20),
230     width: MediaQuery.of(context).size.width,
231     decoration: BoxDecoration(
232         borderRadius: BorderRadius.circular(10),
233         color: Colors.blueGrey[50],
234     ),

```

```

235     child: InkWell(
236       onTap: () {
237         Navigator.push(
238           context,
239           MaterialPageRoute(
240             builder: (context) => PrevPrescriptionUser()),
241         );
242       },
243       child: Row(
244         crossAxisAlignment: CrossAxisAlignment.start,
245         children: [
246           Text(
247             "See Previous Prescriptions",
248             style: GoogleFonts.lato(
249               fontSize: 16,
250               fontWeight: FontWeight.bold,
251               color: Colors.black,
252             ),
253         ),
254         SizedBox(
255           width: 5,
256         ),
257         Icon(Icons.chevron_right_rounded)
258       ],
259     ),
260   ),
261 ),
262 Container(
263   margin: EdgeInsets.only(left: 15, right: 15, top: 20),
264   padding: EdgeInsets.only(left: 20, top: 20),
265   height: MediaQuery.of(context).size.height / 2,
266   width: MediaQuery.of(context).size.width,
267   decoration: BoxDecoration(
268     borderRadius: BorderRadius.circular(10),
269     color: Colors.blueGrey[50],
270   ),
271   child: Column(
272     children: [
273       Row(
274         children: [
275           ClipRRect(
276             borderRadius: BorderRadius.circular(30),
277             child: Container(
278               height: 27,
279               width: 27,
280               color: Colors.green[900],
281               child: Icon(
282                 FlutterIcons.history_faw,
283                 color: Colors.white,
284                 size: 16,
285               ),
286             ),
287           ),
288           SizedBox(
289             width: 10,
290           ),
291           Text(
292             "Appointment History",
293             style: GoogleFonts.lato(
294               fontSize: 16,
295               fontWeight: FontWeight.bold,

```

```

296         color: Colors.black,
297     ),
298 ),
299 ],
300 ),
301 SizedBox(
302     height: 10,
303 ),
304 Expanded(
305     child: Scrollbar(
306         child: Container(
307             padding: EdgeInsets.only(left: 35, right: 15),
308             child: SingleChildScrollView(
309                 child: AppointmentHistoryList(),
310             ),
311         ),
312     ),
313 ),
314 ],
315 ),
316 ),
317 SizedBox(
318     height: 30,
319 ),
320 ],
321 ),
322 ),
323 ),
324 );
325 }
326 }
327 import 'package:cloud_firestore/cloud_firestore.dart';
328 import 'package:firebase_auth/firebase_auth.dart';
329 import 'package:flutter/material.dart';
330 import 'package:google_fonts/google_fonts.dart';
331 import 'package:health_and_doctor_appointment/firestore-data/myAppointmentList.
    dart';
332
333 class MyAppointments extends StatefulWidget {
334     @override
335     _MyAppointmentsState createState() => _MyAppointmentsState();
336 }
337
338 class _MyAppointmentsState extends State<MyAppointments> {
339     FirebaseAuth _auth = FirebaseAuth.instance;
340     User user;
341
342     Future<void> _getUser() async {
343         user = _auth.currentUser;
344     }
345
346     @override
347     void initState() {
348         super.initState();
349         _getUser();
350     }
351
352     @override
353     Widget build(BuildContext context) {
354         return Scaffold(
355             backgroundColor: Colors.white,

```

```
356     appBar: AppBar(  
357       backgroundColor: Colors.white,  
358       title: Container(  
359         alignment: Alignment.center,  
360         padding: EdgeInsets.symmetric(vertical: 10),  
361         child: Text(  
362           'My Appointments',  
363           style: GoogleFonts.lato(  
364             color: Colors.black,  
365             fontWeight: FontWeight.bold,  
366           )),  
367         ),  
368       ),  
369       iconTheme: IconThemeData(color: Colors.black),  
370     ),  
371     body: Container(  
372       padding: EdgeInsets.only(right: 10, left: 10, top: 10),  
373       child: MyAppointmentList(),  
374     ),  
375   );  
376 }  
377 }
```



6.4 Patient's Medical records

```

1  import 'package:cloud_firestore/cloud_firestore.dart';
2  import 'package:firebase_auth/firebase_auth.dart';
3  import 'package:flutter/material.dart';
4  import 'package:google_fonts/google_fonts.dart';
5  import 'package:health_and_doctor_appointment/screens/Doctorpage.dart';
6  import 'package:health_and_doctor_appointment/screens/prevPrescription.dart';
7  import 'package:health_and_doctor_appointment/screens/seeMedicalRecord.dart';
8  import 'package:health_and_doctor_appointment/screens/videoCall.dart';
9  import 'package:health_and_doctor_appointment/screens/videoCallDoctor.dart';
10
11 class Prescription extends StatefulWidget {
12   final useremail;
13   final name;
14   final docName;
15   final time;
16   final docID;
17
18   const Prescription(
19     {Key key, this.useremail, this.name, this.docName, this.time, this.docID})
20     : super(key: key);
21   @override
22   _PrescriptionState createState() => _PrescriptionState();
23 }
24
25 class _PrescriptionState extends State<Prescription> {
26   final TextEditingController _controller = TextEditingController();
27   final TextEditingController _controllerFee = TextEditingController();
28   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
29
30   FirebaseAuth _auth = FirebaseAuth.instance;
31   User user;
32
33   Future<void> _getUser() async {
34     user = _auth.currentUser;
35   }
36
37   @override
38   void initState() {
39     super.initState();
40     _getUser();
41   }
42
43   showAlertDialog(BuildContext context) {
44     // set up the button
45     Widget okButton = TextButton(
46       child: Text(
47         "OK",
48         style: GoogleFonts.lato(fontWeight: FontWeight.bold),
49       ),
50       onPressed: () {
51         Navigator.pushReplacement(
52           context,
53           MaterialPageRoute(
54             builder: (context) => Doctorpage(),
55           ),
56         );
57       },
58     );
59

```



```

60 // set up the AlertDialog
61 AlertDialog alert = AlertDialog(
62     title: Text(
63         "Done!",
64         style: GoogleFonts.lato(
65             fontWeight: FontWeight.bold,
66         ),
67     ),
68     content: Text(
69         "Prescription added successfully.",
70         style: GoogleFonts.lato(),
71     ),
72     actions: [
73         okButton,
74     ],
75 );
76
77 // show the dialog
78 showDialog(
79     context: context,
80     builder: (BuildContext context) {
81         return alert;
82     },
83 );
84 }
85
86 Future<void> addPrescription() async {
87     FirebaseFirestore.instance
88         .collection('appointments')
89         .doc(widget.useremail)
90         .collection('prescriptions')
91         .doc()
92         .set({
93             'prescription': _controller.text,
94             'fees': _controllerFee.text,
95             'docID': widget.docID,
96             'doctor': user.displayName,
97             'time': Timestamp.now()
98         }, SetOptions(merge: true));
99
100     showAlertDialog(context);
101 }
102
103 @override
104 Widget build(BuildContext context) {
105     var meetName = widget.name;
106     meetName = widget.name.replaceAll(RegExp('\\s+'), '-');
107
108     print(Timestamp.now());
109
110     return Scaffold(
111         appBar: AppBar(
112             title: Text("Prescription",
113                 style: GoogleFonts.lato(
114                     color: Colors.black, fontWeight: FontWeight.bold)),
115             backgroundColor: Colors.white,
116             iconTheme: IconThemeData(color: Colors.black),
117         ),
118         body: SingleChildScrollView(
119             child: Padding(
120                 padding: const EdgeInsets.all(15.0),

```

```

121     child: Column(
122       children: [
123         Container(
124           child: StreamBuilder(
125             stream: FirebaseFirestore.instance
126               .collection('appointments')
127               .doc(widget.useremail)
128               .collection('prescriptions')
129               .orderBy('docID')
130               .startAt([widget.docID]).endAt(
131                 [widget.docID + '\uf8ff']).snapshots(),
132             builder: (BuildContext context,
133               AsyncSnapshot<QuerySnapshot> snapshot) {
134               if (!snapshot.hasData) return Container();
135               if (snapshot.data.size != 0) {
136                 return Container(
137                   child: TextButton(
138                     onPressed: () {
139                       Navigator.push(
140                         context,
141                         MaterialPageRoute(
142                           builder: (context) => PrevPrescription(
143                             email: widget.useremail,
144                             docID: widget.docID,
145                           ),
146                         ),
147                       );
148                     },
149                     child: Text(
150                       "See Previous Prescriptions",
151                       style: GoogleFonts.lato(
152                         fontSize: 16,
153                         fontWeight: FontWeight.bold,
154                       ),
155                     ),
156                   ),
157                 );
158               } else {
159                 return Container();
160               }
161             },
162           ),
163         ),
164         Container(
165           margin: EdgeInsets.only(top: 10),
166           child: Form(
167             key: _formKey,
168             child: Column(
169               crossAxisAlignment: CrossAxisAlignment.start,
170               children: [
171                 Text(
172                   "Patient Name",
173                   style: GoogleFonts.lato(
174                     fontSize: 18, fontWeight: FontWeight.bold),
175                 ),
176                 Container(
177                   alignment: Alignment.centerLeft,
178                   margin: EdgeInsets.only(top: 10),
179                   width: MediaQuery.of(context).size.width,
180                   height: 40,
181                   color: Colors.grey[200],

```

```

182         child: Text(
183             widget.name,
184             style: GoogleFonts.lato(fontSize: 18),
185         ),
186     ),
187     TextButton(
188         onPressed: () {
189             Navigator.push(
190                 context,
191                 MaterialPageRoute(
192                     builder: (context) => SeeMedicalRecord(
193                         email: widget.useremail,
194                     )),
195             );
196         },
197         child: Text(
198             "See Patient's Medical Record",
199             style: GoogleFonts.lato(
200                 fontWeight: FontWeight.bold,
201             ),
202         ),
203     ),
204     SizedBox(
205         height: 10,
206     ),
207     Text(
208         "Time",
209         style: GoogleFonts.lato(
210             fontSize: 18, fontWeight: FontWeight.bold),
211     ),
212     Container(
213         alignment: Alignment.centerLeft,
214         margin: EdgeInsets.only(top: 10),
215         width: MediaQuery.of(context).size.width,
216         height: 40,
217         color: Colors.grey[200],
218         child: Text(
219             widget.time.toDate().toString(),
220             style: GoogleFonts.lato(fontSize: 18),
221         ),
222     ),
223     SizedBox(
224         height: 25,
225     ),
226     Text(
227         "Add Prescription",
228         style: GoogleFonts.lato(
229             fontSize: 18, fontWeight: FontWeight.bold),
230     ),
231     TextFormField(
232         controller: _controller,
233         keyboardType: TextInputType.multiline,
234         maxLines: 5,
235         style: GoogleFonts.lato(
236             fontSize: 18, fontWeight: FontWeight.bold),
237         decoration: InputDecoration(
238             contentPadding:
239                 EdgeInsets.only(left: 10, top: 10, bottom: 10),
240             border: OutlineInputBorder(
241                 borderSide: BorderSide.none,
242             ),

```

```

243         filled: true ,
244         fillColor: Colors.grey[200],
245         hintText: 'Add Prescription' ,
246         hintStyle: GoogleFonts.lato(
247             color: Colors.black26 ,
248             fontSize: 18,
249             fontWeight: FontWeight.w800,
250         ),
251     ),
252     textInputAction: TextInputAction.next ,
253     validator: (value) {
254         if (value.isEmpty)
255             return 'Please Add the Prescription';
256         return null;
257     },
258 ),
259 SizedBox(
260     height: 25,
261 ),
262 Text(
263     "Fees" ,
264     style: GoogleFonts.lato(
265         fontSize: 18, fontWeight: FontWeight.bold),
266 ),
267 TextFormField(
268     controller: _controllerFee ,
269     keyboardType: TextInputType.number ,
270     style: GoogleFonts.lato(
271         fontSize: 18, fontWeight: FontWeight.bold),
272     decoration: InputDecoration(
273         contentPadding:
274             EdgeInsets.only(left: 10, top: 10, bottom: 10),
275         border: OutlineInputBorder(
276             borderSide: BorderSide.none ,
277         ),
278         filled: true ,
279         fillColor: Colors.grey[200],
280         hintText: 'Add fees in ' ,
281         hintStyle: GoogleFonts.lato(
282             color: Colors.black26 ,
283             fontSize: 18,
284             fontWeight: FontWeight.w800,
285         ),
286     ),
287     textInputAction: TextInputAction.next ,
288     validator: (value) {
289         if (value.isEmpty) return 'Please Add the Fees';
290         return null;
291     },
292 ),
293 SizedBox(
294     height: 25,
295 ),
296 Container(
297     width: MediaQuery.of(context).size.width ,
298     decoration: BoxDecoration(
299         color: Colors.blue[900],
300         borderRadius: BorderRadius.circular(12)),
301     child: TextButton(
302         onPressed: () {
303             if (_formKey.currentState.validate()) {

```

```

304         addPrescription ();
305     }
306 },
307 child: Text(
308     "Add Prescription",
309     style: GoogleFonts.lato(
310         color: Colors.white,
311         fontSize: 16,
312         fontWeight: FontWeight.bold,
313     ),
314 ),
315 ),
316 ),
317 SizedBox(
318     height: 15,
319 ),
320 Divider(
321     color: Colors.grey,
322 ),
323 SizedBox(
324     height: 15,
325 ),
326 Container(
327     width: MediaQuery.of(context).size.width,
328     height: 50,
329     decoration: BoxDecoration(
330         borderRadius: BorderRadius.circular(12),
331     ),
332     child: MaterialButton(
333         shape: RoundedRectangleBorder(
334             borderRadius: new BorderRadius.circular(12.0)),
335         onPressed: () {
336             Navigator.push(
337                 context,
338                 MaterialPageRoute(
339                     builder: (context) => MeetingDoctor(
340                         roomID: meetName,
341                     )),
342             );
343         },
344         color: Colors.blue[900],
345         child: Text(
346             "Connect Video Call",
347             style: GoogleFonts.lato(
348                 color: Colors.white,
349                 fontSize: 16,
350                 fontWeight: FontWeight.w600),
351         ),
352     ),
353 ),
354 SizedBox(
355     height: 25,
356 ),
357 ),
358 ),
359 ),
360 ),
361 ),
362 ),
363 }
364 }

```



6.5 Consulting the Doctor

```

1  import 'dart:io';
2
3  import 'package:flutter/cupertino.dart';
4  import 'package:flutter/foundation.dart';
5  import 'package:flutter/material.dart';
6  import 'package:google_fonts/google_fonts.dart';
7  import 'package:jitsi_meet/jitsi_meet.dart';
8
9  class Meeting extends StatefulWidget {
10   final roomID;
11   const Meeting({Key key, this.roomID}) : super(key: key);
12
13   @override
14   _MeetingState createState() => _MeetingState();
15 }
16
17 class _MeetingState extends State<Meeting> {
18   final serverText = TextEditingController();
19   final roomText = TextEditingController();
20   final subjectText = TextEditingController(text: "My Plugin Test Meeting");
21   final nameText = TextEditingController(text: "Plugin Test User");
22   final emailText = TextEditingController(text: "test@email.com");
23   final iosAppBarRGBAColor =
24     TextEditingController(text: "#0080FF80"); //transparent blue
25   bool isAudioOnly = true;
26   bool isAudioMuted = true;
27   bool isVideoMuted = true;
28
29   @override
30   void initState() {
31     super.initState();
32     JitsiMeet.addListener(JitsiMeetingListener(
33       onConferenceWillJoin: _onConferenceWillJoin,
34       onConferenceJoined: _onConferenceJoined,
35       onConferenceTerminated: _onConferenceTerminated,
36       onError: _onError));
37     roomText.text = widget.roomID + "_consultation";
38   }
39
40   @override
41   void dispose() {
42     super.dispose();
43     JitsiMeet.removeAllListeners();
44   }
45
46   @override
47   Widget build(BuildContext context) {
48     double width = MediaQuery.of(context).size.width;
49     return Scaffold(
50       appBar: AppBar(
51         backgroundColor: Colors.white,
52         iconTheme: IconThemeData(color: Colors.black),
53         title: Text('Connect Video call',
54           style: GoogleFonts.lato(
55             fontSize: 18,
56             fontWeight: FontWeight.bold,
57             color: Colors.black,
58           )),
59     ),

```

```

60 body: Container(
61   padding: const EdgeInsets.symmetric(
62     horizontal: 16.0,
63   ),
64   child: kIsWeb
65     ? Row(
66       mainAxisAlignment: MainAxisAlignment.spaceBetween,
67       children: [
68         Container(
69           width: width * 0.30,
70           child: meetConfig(),
71         ),
72         Container(
73           width: width * 0.60,
74           child: Padding(
75             padding: const EdgeInsets.all(8.0),
76             child: Card(
77               color: Colors.white54,
78               child: SizedBox(
79                 width: width * 0.60 * 0.70,
80                 height: width * 0.60 * 0.70,
81                 child: JitsiMeetConferencing(
82                   extraJS: [
83                     // extraJs setup example
84                     '<script>function echo(){console.log("echo
85                       !!!")};</script>',
86                     '<script src="https://code.jquery.com/jquery
87                       -3.5.1.slim.js" integrity="sha256-
88                       DrT5NfxfbHyMHux3lLkxhg42LY6of8TaYyK50jnxRnM
89                       =" crossorigin="anonymous"></script>'
90                   ],
91                 ),
92               ),
93             ),
94           ),
95         ],
96       ),
97       : meetConfig(),
98     ),
99   );
100 }
101
102 Widget meetConfig() {
103   return SingleChildScrollView(
104     child: Column(
105       children: <Widget>[
106         SizedBox(
107           height: 16.0,
108         ),
109         TextField(
110           controller: serverText,
111           decoration: InputDecoration(
112             border: OutlineInputBorder(),
113             labelText: "Server URL",
114             hintText: "Hint: Leave empty for meet.jitsi.si"),
115         ),
116         SizedBox(
117           height: 14.0,
118         ),
119         TextField(
120           controller: roomText,
121           decoration: InputDecoration(

```



```

117         border: OutlineInputBorder(),
118         labelText: "Room",
119     ),
120 ),
121 SizedBox(
122     height: 14.0,
123 ),
124 TextField(
125     controller: subjectText,
126     decoration: InputDecoration(
127         border: OutlineInputBorder(),
128         labelText: "Subject",
129     ),
130 ),
131 SizedBox(
132     height: 14.0,
133 ),
134 TextField(
135     controller: nameText,
136     decoration: InputDecoration(
137         border: OutlineInputBorder(),
138         labelText: "Display Name",
139     ),
140 ),
141 SizedBox(
142     height: 14.0,
143 ),
144 TextField(
145     controller: emailText,
146     decoration: InputDecoration(
147         border: OutlineInputBorder(),
148         labelText: "Email",
149     ),
150 ),
151 SizedBox(
152     height: 14.0,
153 ),
154 TextField(
155     controller: iosAppBarRGBAColor,
156     decoration: InputDecoration(
157         border: OutlineInputBorder(),
158         labelText: "AppBar Color(IOS only)",
159         hintText: "Hint: This HAS to be in HEX RGBA format"),
160 ),
161 SizedBox(
162     height: 14.0,
163 ),
164 CheckboxListTile(
165     title: Text("Audio Only"),
166     value: isAudioOnly,
167     onChanged: _onAudioOnlyChanged,
168 ),
169 SizedBox(
170     height: 14.0,
171 ),
172 CheckboxListTile(
173     title: Text("Audio Muted"),
174     value: isAudioMuted,
175     onChanged: _onAudioMutedChanged,
176 ),
177 SizedBox(

```

```

178         height: 14.0,
179     ),
180     CheckboxListTile(
181         title: Text("Video Muted"),
182         value: isVideoMuted,
183         onChanged: _onVideoMutedChanged,
184     ),
185     Divider(
186         height: 30.0,
187         thickness: 2.0,
188     ),
189     SizedBox(
190         height: 50.0,
191         width: double.maxFinite,
192         child: ElevatedButton(
193             onPressed: () {
194                 _joinMeeting();
195             },
196             child: Text(
197                 "Join Meeting",
198                 style: GoogleFonts.lato(
199                     color: Colors.white,
200                     fontSize: 18,
201                     fontWeight: FontWeight.w600,
202                 ),
203             ),
204             style: ButtonStyle(
205                 backgroundColor: MaterialStateColor.resolveWith(
206                     (states) => Colors.indigo[600]),
207             ),
208         ),
209     ),
210     SizedBox(
211         height: 48.0,
212     ),
213 ],
214 );
215 }
216
217 _onAudioOnlyChanged(bool value) {
218     setState(() {
219         isAudioOnly = value;
220     });
221 }
222
223 _onAudioMutedChanged(bool value) {
224     setState(() {
225         isAudioMuted = value;
226     });
227 }
228
229 _onVideoMutedChanged(bool value) {
230     setState(() {
231         isVideoMuted = value;
232     });
233 }
234
235 _joinMeeting() async {
236     String serverUrl = serverText.text.trim().isEmpty ? null : serverText.text;
237
238     // Enable or disable any feature flag here

```

```

239 // If feature flag are not provided, default values will be used
240 // Full list of feature flags (and defaults) available in the README
241 Map<FeatureFlagEnum, bool> featureFlags = {
242   FeatureFlagEnum.WELCOME_PAGE_ENABLED: false,
243 };
244 if (!kIsWeb) {
245   // Here is an example, disabling features for each platform
246   if (Platform.isAndroid) {
247     // Disable ConnectionService usage on Android to avoid issues (see
248     // README)
249     featureFlags [FeatureFlagEnum.CALL_INTEGRATION_ENABLED] = false;
250   } else if (Platform.isIOS) {
251     // Disable PIP on iOS as it looks weird
252     featureFlags [FeatureFlagEnum.PIP_ENABLED] = false;
253   }
254 }
255 // Define meetings options here
256 var options = JitsiMeetingOptions(room: roomText.text)
257   ..serverURL = serverUrl
258   ..subject = subjectText.text
259   ..userDisplayName = nameText.text
260   ..userEmail = emailText.text
261   ..iosAppBarRGBAColor = iosAppBarRGBAColor.text
262   ..audioOnly = isAudioOnly
263   ..audioMuted = isAudioMuted
264   ..videoMuted = isVideoMuted
265   ..featureFlags.addAll(featureFlags)
266   ..webOptions = {
267     "roomName": roomText.text,
268     "width": "100%",
269     "height": "100%",
270     "enableWelcomePage": false,
271     "chromeExtensionBanner": null,
272     "userInfo": {"displayName": nameText.text}
273   };
274 debugPrint("JitsiMeetingOptions: $options");
275 await JitsiMeet.joinMeeting(
276   options,
277   listener: JitsiMeetingListener(
278     onConferenceWillJoin: (message) {
279       debugPrint("${options.room} will join with message: $message");
280     },
281     onConferenceJoined: (message) {
282       debugPrint("${options.room} joined with message: $message");
283     },
284     onConferenceTerminated: (message) {
285       debugPrint("${options.room} terminated with message: $message");
286     },
287     genericListeners: [
288       JitsiGenericListener(
289         eventName: 'readyToClose',
290         callback: (dynamic message) {
291           debugPrint("readyToClose callback");
292         },
293       ),
294     ],
295   );
296 }
297 void _onConferenceWillJoin(message) {
298   debugPrint("_onConferenceWillJoin broadcasted with message: $message");

```

```
299     }
300
301     void _onConferenceJoined(message) {
302         debugPrint("_onConferenceJoined broadcasted with message: $message");
303     }
304
305     void _onConferenceTerminated(message) {
306         debugPrint("_onConferenceTerminated broadcasted with message: $message");
307     }
308
309     _onError(error) {
310         debugPrint("_onError broadcasted: $error");
311     }
312 }
```



Chapter 7

System Testing

System Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Below shows the test cases of our system.

7.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	User Registration	All Valid Input	User Registered Successfully/ Unsuccessfully	Success/ Failed
T02	User Login	Username and Password Required	User login Successfully/ Unsuccessfully	Success/ Failed
T03	Select Diseases	Display Dr. List	Display Dr. name successfully	Successfully Display Dr. name
T04	Doctor List	Display Doctor Details	Select Doctor successfully	Successfully Selected Doctor
T05	Book Appointment	Enter valid Input	Display Appointment Booking Page successfully	Successfully Fill Appointment Details
T06	Time and Date	Display Time and Date	Successfully Display correct Clock and Calendar	Successfully Book Appointment
T07	All Appointments	Data Fetch and show all Booked Appointments	Successfully Display all Appointments	Successful Display all Appointments

T08	Consult Doctor	Connect to Video Meeting	Connect to Call Successfully/ Unsuccessfully	Success/ Failed Call
T09	Doctor Login	Username and Password Required	Doctor login Successfully/ Unsuccessfully	Success/ Failed
T10	Dr. View Ap- pointments	All Appointments	Display All Ap- pointments Suc- cessfully	Successfully Display All Ap- pointments
T11	View Medical History	Data fetch and Dis- play	Display Medical History of a Patient Successfully	Successfully Dis- play Medical History of a Patient
T12	Call Patient	Connect to Call Successfully/ Unsuccessfully	Success/ Failed Call	
T13	Prescription	Send Prescription to Patient	Sent the Prescrip- tion	Successfully Sent the Prescription
T14	Make Payment	Select Payment Mode	Navigate to the se- lected Payment Ap- plication	Successfully Redi- rected to the Pay- ment App Applica- tion

7.2 Test Cases

Title: Login Page – Authenticate Successfully on gmail.com

Description: A registered user should be able to successfully login at gmail.com.

Precondition: the user must already be registered with an email address and password.

Assumption: Read/Write permissions are given in firebase.

Test Steps:

1. Navigate to gmail.com
2. In the 'email' field, enter the email of the registered user.
3. Enter the password of the registered user
4. Click 'Sign In'

Expected Result: System Should take the user to Home Page.

Actual Result:

Successfully Redirected to the Home Page

Title: Book Appointment – Book Appointment Successfully

Description: A registered user should be able to successfully book an appointment using book appointment feature.

Precondition: the user must already be logged in into the system.

Assumption: Read/Write permissions are given in firebase.

Test Steps:

1. Navigate to Book Appointment feature.
2. Fill all the details on the page.
3. Select Time and Date.
4. Click ‘Book Appointment’.

Expected Result: System Should take the user to All Appointments Page.

Actual Result:

Successfully Redirected to the All Appointments Page

7.2.1 Software Quality Attributes

Quality characteristics for the product that will be important to either the Patient and the Doctor. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability.

Chapter 8

Screenshots of Project

8.1 Patient View



8.1.1 Registration



Figure 8.1: Patient Sign up

8.1.2 Login



Login

group18@mail.com

.....

Sign In

Forgot Password?



Don't have an account? [Signup here](#)

8.1.3 Home Page

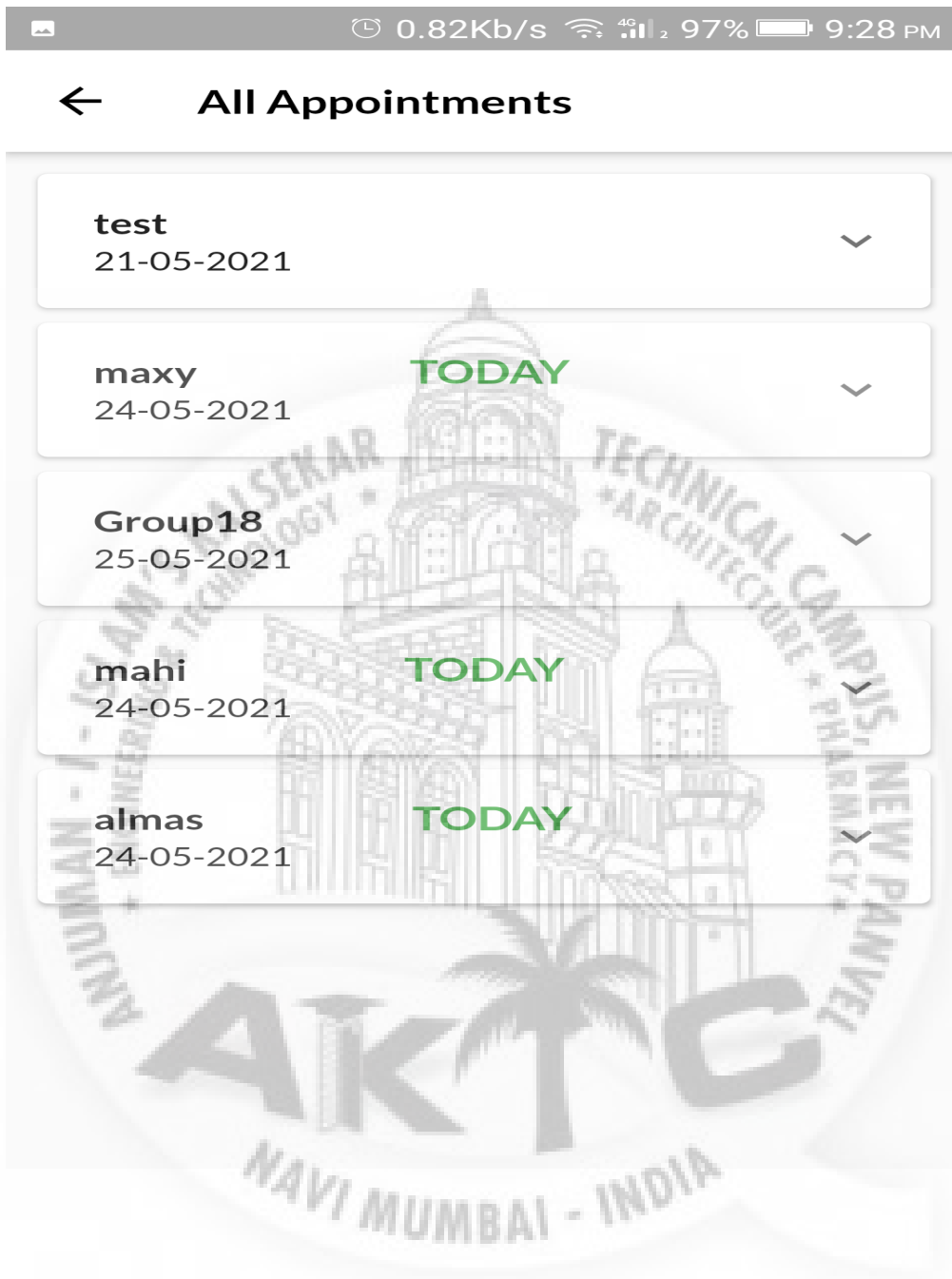


Figure 8.2: Home Page

8.2 Doctor View




8.2.1 All Appointments



8.2.2 Appointment Scheduled

3.53Kb/s 98% 9:22 PM

Welcome

 **Kartik**

Your Appointments,

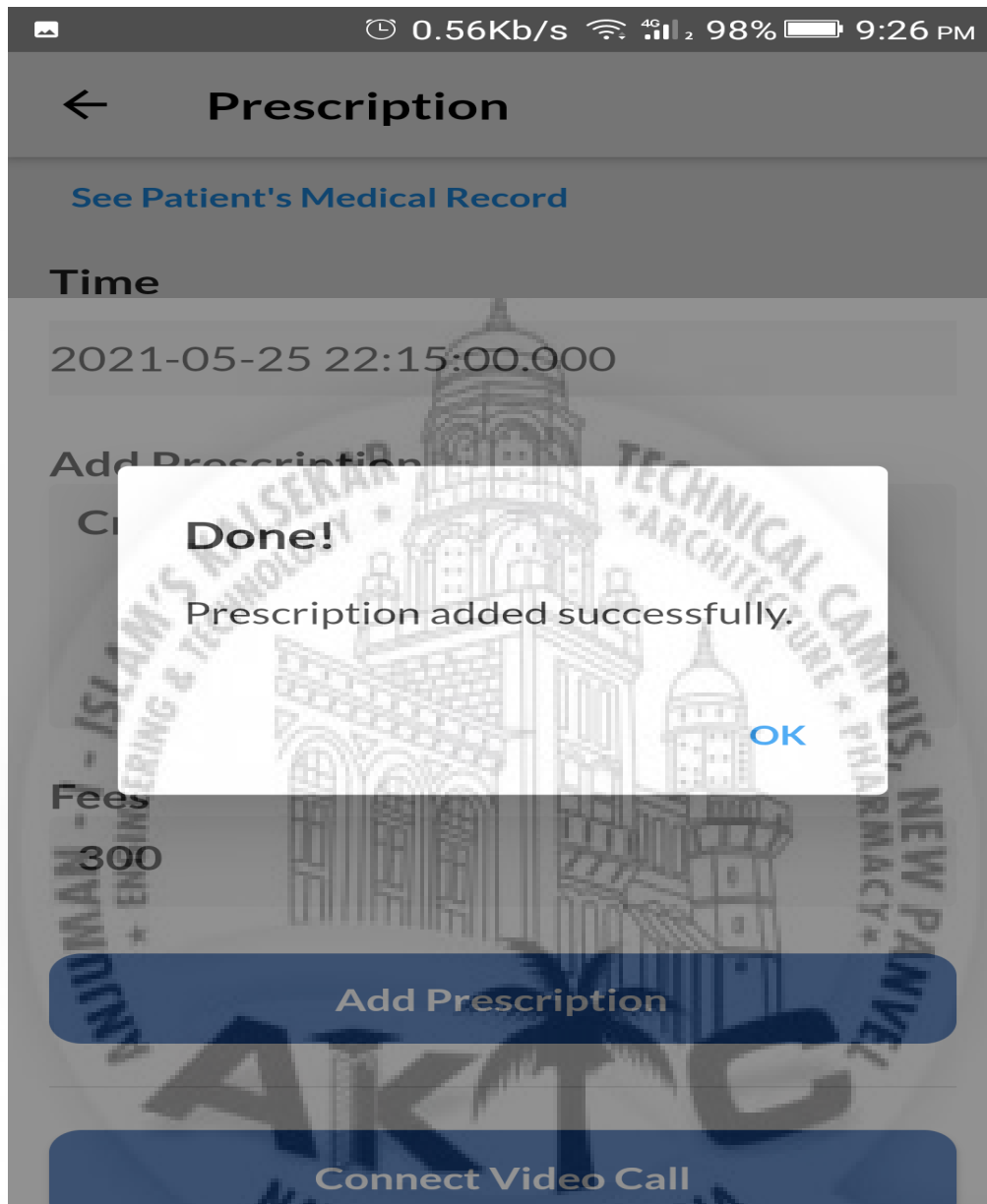
- mehhhh**
23-05-2021
- Group18**
25-05-2021
Description: I've Fever from past 4 days.
Time: 22:15
- maxy** **TODAY**
24-05-2021
- almas** **TODAY**
24-05-2021

AKTC
NAVI MUMBAI - INDIA

8.2.3 Video Calling



8.2.4 Prescription



8.2.5 Payment



Chapter 9

Conclusion and Future Scope

9.1 Conclusion

In this pandemic, safety has become the first priority. Thus, with the proposed system one can assume about his/her safety by consulting being at home.

Securing and archiving the paper-based records is difficult and it can be stolen, burned or modified, so the need for such a system was very essential. Also it is considered time and cost effective to healthcare.

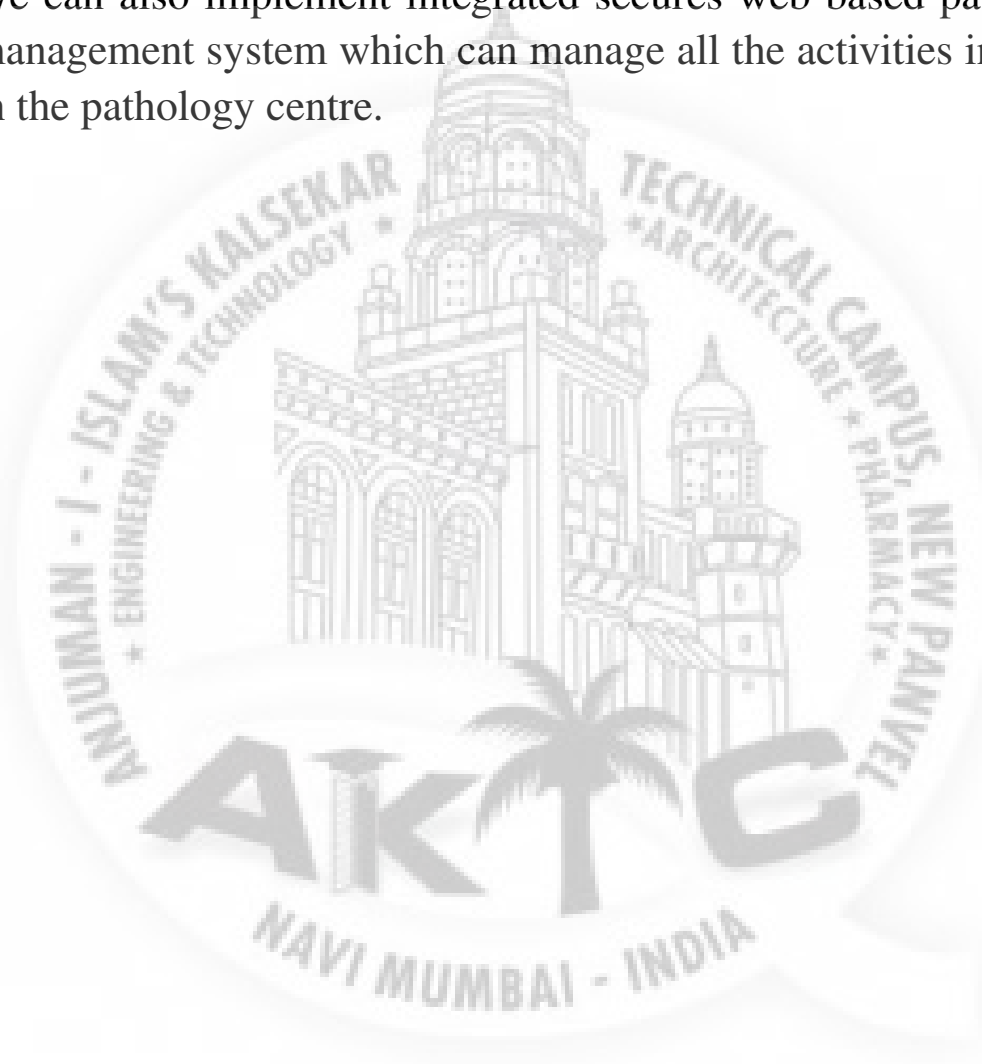
This system will also help to maintain the medical records and the patients data which will be helpful for both the patient as well as doctor reducing the paperwork.

Even the booking an appointment and getting reminder for the same becomes easy and efficient.

In future we can also add up some more features in our application like analysis of the patients based on the medical records stored and also on online pharmacy feature within the application.

9.2 Future Scope

- We can implement the feature for pharmacy application which will facilitate the user to get access to the medicines without walking through every pharmacy in the tracked location.
- We can also implement integrated secure web based pathology management system which can manage all the activities involved in the pathology centre.



References

- [1] 'Doctor Who?'(2019); NAME OF AUTHORS:Author1:Abdur Razzak Author2:Robin Roy,A customizable Android Application for Integrated Health Care,<https://ieeexplore.ieee.org/document/8944501>
- [2] 'Smart Healthcare'(2009) *NAME OF site*; NAME OF AUTHORS:Author1:Erwim Halim Author2:Diyurman Gea,<https://ieeexplore.ieee.org/document/8822574>
- [3] ' Mr.Doc; NAME OF AUTHORS:Author1:Author1:Shafaq Malik Author2:Nargis Bibi ,A Doctor Appointment Application System
- [4] <https://www.medscape.com/viewarticle/935981>
- [5] <https://www.bain.com/insights/why-telehealth-will-outlast-the-pandemic/>
- [6] <https://arxiv.org/ftp/arxiv/papers/1701/1701.08786.pdf>