Quick Care

Submitted By

Subhan Ahmed

Qazi Abdul Rahim

Session 2020-2024

Institute of Computer Science

Khwaja Fareed University of Engineering & Information Technology

Rahim Yar Khan

2024

Quick Care

Submitted to
Ms. Humaira Anwer

Institute of Computer Science

In partial fulfilment of the requirements
For the degree of

Bachelor's in Computer Science

By

Subhan Ahmed

COSC201101079

Qazi Abdul Rahim

COSC201101618

Session 2020-2024

Khwaja Fareed University of Engineering & Information Technology
Rahim Yar Khan
2024

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Khwaja Fareed University of engineering & Information Technology or other institutions.

Reg No	:	COSC201101079	Reg No	:	COSC201101618
Name	:	Subhan Ahmed	Name	:	Qazi Abdul Rahim
Signature	:	_____	Signature	:	_____
Date	:	_____	Date	:	_____

APPROVAL FOR SUBMISSION

I certify that this project report entitled ***Quick Care*** was prepared by **Subhan Ahmed** and **Qazi Abdul Rahim** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor's in Computer Science at Khwaja Fareed University of Engineering & Information Technology.

Approved by:

Signature : _____

Supervisor : Ms. Humaira Anwer

Date : _____

MEETING LOG

Project Name: Quick Care Supervisor: Ms. Humaira Anwer

Meeting No.	Date	Outcome	Signature of Supervisor
1.	05/09/23	Project Proposal	
2.	12/09/23	Finalizing functional requirements	
3.	19/09/23	Discuss Title Defence presentation	
4.	28/09/23	Discussed documentation and its format	
5.	03/10/23	Presented completed Chapter 1 for review	
6.	10/10/23	Discussed existing systems and did comparison	
7.	17/10/23	Presented completed Chapter 2 for review	
8.	24/10/23	Presented fixed Chapter 2 for review	
9.	07/11/23	Discussed needs of user and function requirements details	
10.	14/11/23	Shown progress of Chapter 3 and discussed feature mapping	
11.	16/11/23	Presented completed Chapter 3 for review	
12.	21/11/23	Discussed software process models suitable for project	
13.	28/11/23	Presented ER Diagram for review	
14.	05/12/23	Presented few of the UML Diagrams for review	
15.	08/12/23	Presented complete Chapter 4 for review	
16.	13/12/23	Presented FYP-I complete documentation for review	

MEETING LOG

Project Name: Quick Care Supervisor: Ms. Humaira Anwer

Meeting No.	Date	Outcome	Signature of Supervisor
17.	22/01/24	Overview for starting app development	
18.	25/01/24	Discussed the app flow	
19.	30/01/24	Discussed different databases that can be used	
20.	01/02/23	Presented the work done on the base project	
21.	06/02/23	Discussed structure of firebase firestore database	
22.	13/02/24	Presented implementation for appointment creation	
23.	20/03/24	Discussed the current status of the project	
24.	27/02/24	Completed and presented the admin dashboard	
25.	29/02/24	Discussed the issues in the app	
26.	05/03/24	Presented the fixed version	
27.	07/03/24	Completed and presented the patient dashboard	
28.	28/03/24	Presented the changes made in the backend	
29.	03/0424	Implemented and presented the payments API	
30.	16/04/24	Final discussion before internals	
31.	25/04/24	Discussed the present issues in the app	
32.	09/05/24	Presented the final app before externals	

ACKNOWLEDGEMENT

We would like to thank everyone who had contributed to this project. We would like to express our gratitude to our Project supervisor, *Ms. Humaira Anwer* for her invaluable advice, guidance, and her enormous patience throughout the development of the project.

In addition, we would also like to express our gratitude to our loving parents and friends who had helped and given us encouragement.

ABSTRACT

Long wait times and inefficient clinic management are a problem for both patients and doctors. Quick Care app solves this problem by making it easy for patients to book and manage appointments, and for doctors to see who their next patient is. This project makes it easier for doctors to manage their patients and for patients it saves a lot of their valuable time and increased convenience. The main benefit of this project is the time it will save and the crowd if will reduce in front of the doctor's rooms. Users can easily book and manage appointments from the app and before leaving for the doctor can check which patient's turn it is currently so that they arrive right on time for their appointment and there aren't any long queues, saving a lot of time for patients. The app also has the functionality to chat with patients to help them figure which type of doctor they should visit for their condition. To conclude, this app will manage the appointments for both patients and doctors saving their time and help patients by guiding them to the doctor they need.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1. Background

Patients face a lot of issues when they are visiting doctors for appointments. From long and frustrating queues to a lot of mismanagement, it's just chaos. Patients waiting the long queue and facing all the issues is one thing, the hospital/clinical staff is also under pressure for maintaining calm and managing all the patients. So, most of their attention is diverted to managing the queue and the crowd.

1.2. Introduction

The Quick Care app is a mobile app that makes it easy for patients to book and manage appointments, and for doctors to see who their next patient is. This helps to reduce wait times for patients and improve clinic efficiency.

The Quick Care app has the potential to revolutionize the healthcare system by making it easier and more efficient for patients to get the care they need.

1.3. Problem Statement

Long wait times and inefficient clinic management are a major problem for both patients and doctors. Patients often have to wait for hours to see their doctor, and doctors are often overwhelmed by the number of patients they need to see. This can lead to frustration and stress for both parties, and it can also compromise the quality of care that patients receive.

1.4. Objectives

The objectives of the Quick Care appointment management system are to:

1. **Streamline Appointment Scheduling:** Provide a user-friendly platform for patients to book, manage, and track their appointments efficiently, eliminating the hassles of traditional scheduling methods.
2. **Enhance Appointment Transparency:** Offer real-time appointment status updates, keeping patients informed about their scheduled times and any potential delays or changes, reducing unnecessary waiting and uncertainty.

3. **Empower Informed Decision-Making:** Provide comprehensive information about doctors, including qualifications, specialties, and patient reviews, enabling patients to make informed choices when selecting a healthcare provider.
4. **Facilitate Effective Communication:** Integrate AI-powered support and communication channels to address patient inquiries promptly and efficiently, ensuring timely assistance and improved patient satisfaction.
5. **Optimize Clinic Efficiency:** Optimize clinic workflows by streamlining appointment scheduling, reducing no-shows, and improving patient flow, leading to enhanced resource utilization and overall clinic productivity.
6. **Revolutionize Healthcare Experience:** Transform the healthcare experience by providing a seamless, patient-centric platform for appointment management, fostering a more positive and satisfying interaction between patients and healthcare providers.

1.5. Project Scope

To develop a mobile app that allows patients to book and manage appointments, and doctors to see who their next patient is:

Patient features:

- Ability to search for doctors by specialty, location, and insurance coverage
- Ability to book appointments in advance
- Ability to check the status of the queue before leaving for the doctor's office

Doctor features:

- Ability to see a list of all upcoming appointments
- Ability to see the status of each patient in the queue

1.6. Advantages of the System

Quick Care provides a lot of benefits for both patients and the doctors, revolutionizing the way healthcare services are accessed and managed. These include:

- Convenient, accessible and easy to use appointment system that reducing wait times and allows patients to make informed choices
- Streamlines appointment scheduling that allows better patient-doctor experience

- Empowers patients with more control over their healthcare and helps in better and well-informed doctor selection

1.7. Relevance to the Study Program

The development of Quick Care aligns with various aspects of the study program of Computer Science. Some of the relevant courses are mentioned below:

- Software Development
- Design and Analysis of Algorithm
- Database Systems
- Information Security
- Mobile Application Development
- Artificial Intelligence

1.8. Chapter Summary

Quick Care aims to revolutionize the patient experience and clinic efficiency. It supports features like live appointment status tracking, online appointment booking and management, doctor reviews, patient appointment history and Bard AI integration. Our goal is to help patients effortlessly book, manage and track their appointments, reduce their waiting times, allow them to plan their visit efficiently and for doctors to not worry about management and focus on the patients.

Chapter 2

EXISTING SYSTEMS

2.1. Existing Systems

In the healthcare sector, appointment management system plays an important role for streamlining patient-doctor interactions and enhancing overall healthcare delivery. They aim to bridge the gap between patients and the healthcare providers by facilitating seamless appointment scheduling and access to medical services.

A number of prominent appointment management systems have been developed, each offering a unique set of features and functionalities. Although these systems have huge contribution in improving healthcare accessibility and efficiency, they still face limitations that hinder their effectiveness. To effectively address the shortcomings, it is essential to understand the current landscape of existing solutions and their respective strengths and weaknesses.

2.2. Drawbacks in Existing Systems

Existing appointment management systems have several drawbacks that hinder their effectiveness and contribute to a suboptimal healthcare experience. These limitations include:

- **Lack of Live Appointment Status Updates:** Existing systems often fail to provide real-time updates on appointment status, leaving patients unaware of delays or changes in their scheduled appointment times, leading to unnecessary waiting and frustration.
- **Crowded and Confusing Interfaces:** Many systems suffer from cluttered and unintuitive user interfaces, making navigation and appointment management a cumbersome process, particularly for less tech-savvy individuals.
- **Inefficient Support Systems:** Existing support mechanisms often rely on time-consuming phone calls, leading to long wait times and delayed assistance. The lack of integrated AI-powered support further limits the accessibility of timely and personalized assistance.

Our proposed system addresses these drawbacks by providing live appointment status updates, a user-friendly interface, and integrated AI support for seamless and efficient healthcare appointment management.

2.3. Examples of Existing Systems

Some examples for the existing systems are:

1. Marham.pk
2. Oladoc
3. InstaCare

1. Marham.pk

Marham.pk is a prominent healthcare platform in Pakistan, offering a number of services to make healthcare more accessible [1]. Important features include online appointment booking, doctor search, teleconsultations, and access to health information resources. With a huge network of doctors and other healthcare professionals across various specialities, Marham.pk helps patients find suitable doctors and book appointments conveniently.

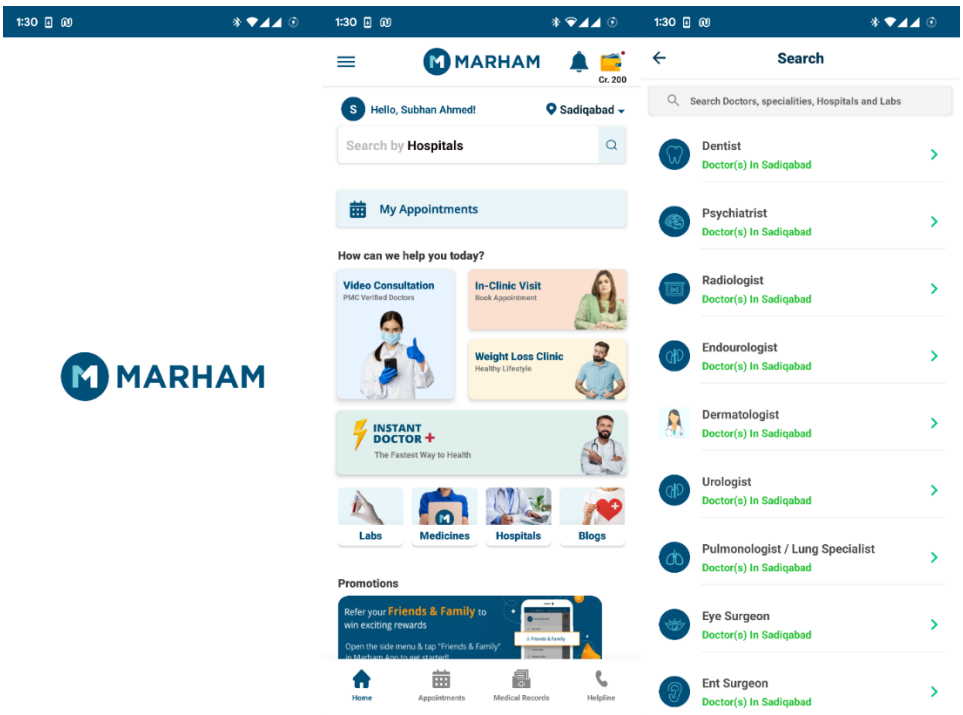

Figure 2.1 Marham Android App

2. Oladoc

Oladoc is a one of the leading healthcare providers in Pakistan. It is focused on streamlining online appointment bookings and doctor consultations. Oladoc was previously known as MyDoctor.pk and was rebranded to Oladoc when they expanded their services internationally

[2]. They have a user-friendly interface, enabling patients to search for doctors based on their location and speciality. They also offer online consultations.

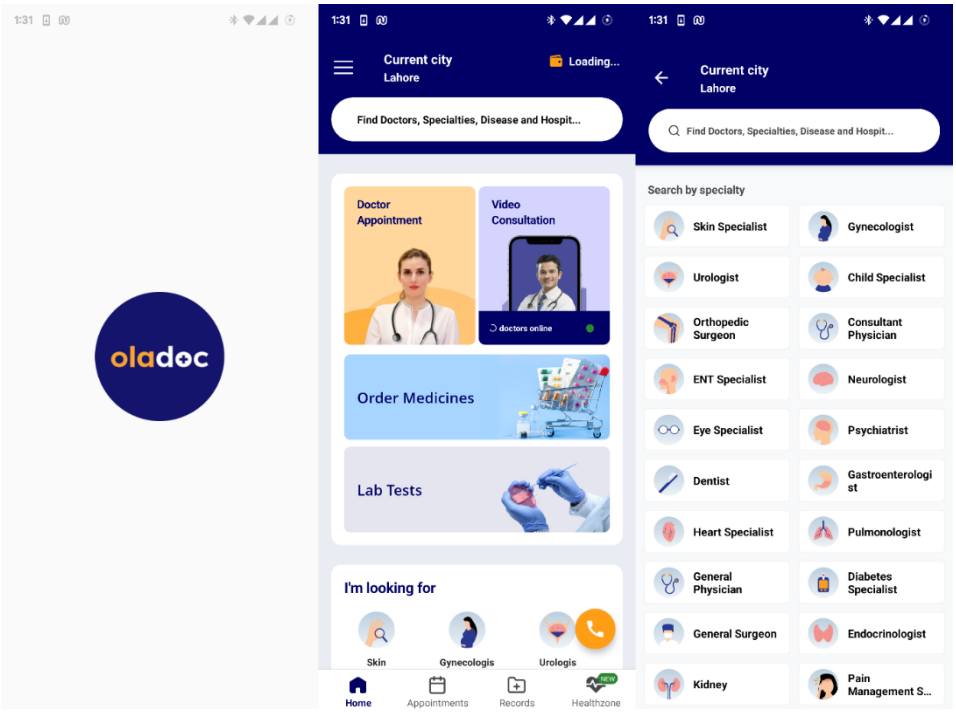

Figure 2.2 Oladoc Android App

3. InstaCare

InstaCare is a rapidly growing healthcare platform in Pakistan, offering services, including online appointment booking, doctor search, lab test bookings, and medicine delivery. It aims to provide a seamless healthcare experience by streamlining various aspects of patient care, from appointment scheduling to medication procurement.

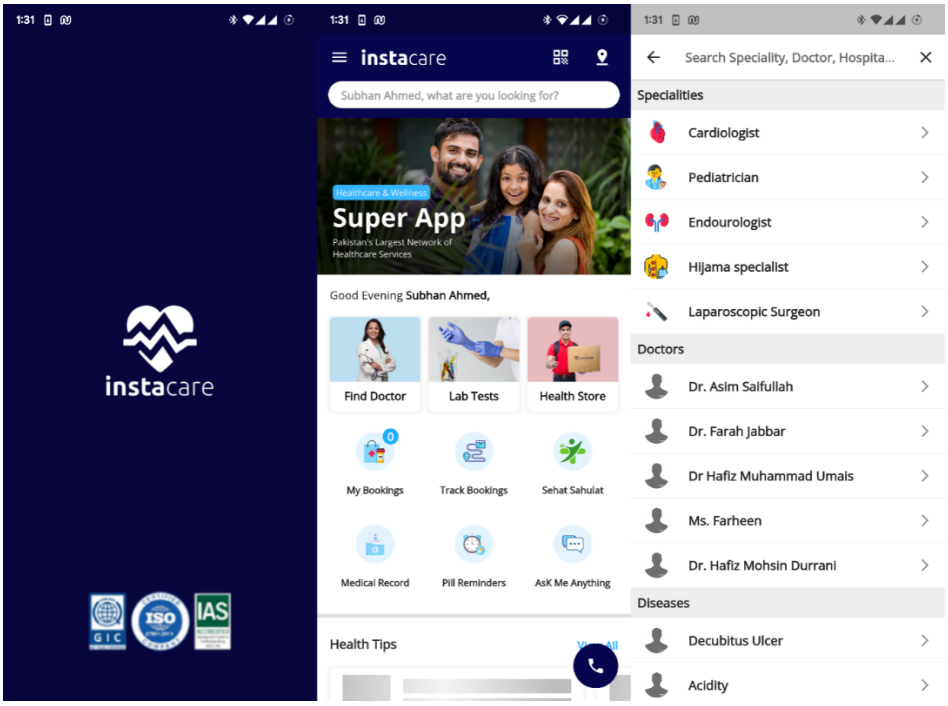

Figure 2.3 InstaCare Android App

2.4. Need to Replace Existing Systems

Existing appointment management systems often fall short in addressing the evolving needs of patients and healthcare providers, leading to inefficiencies, frustrations, and an overall suboptimal healthcare experience. The need to replace these systems stems from several critical shortcomings:

1. **Limited Functionality:** Existing systems often lack comprehensive features, limiting their ability to streamline appointment scheduling, provide real-time status updates, and facilitate effective communication between patients and providers.
2. **Inefficient User Interfaces:** Many existing systems suffer from cluttered and unintuitive user interfaces, making navigation and appointment management cumbersome, particularly for less tech-savvy individuals.
3. **Lack of Personalized Support:** Existing support mechanisms often rely on time-consuming phone calls or emails, leading to delayed assistance and frustration for patients. The lack of integrated AI-powered support further limits personalized assistance.

4. **Inadequate Clinic Optimization:** Existing systems often fail to optimize clinic workflows effectively, leading to scheduling conflicts, patient backlogs, and inefficient resource utilization, hindering overall clinic productivity.
5. **Suboptimal Patient Experience:** Existing systems often prioritize appointment scheduling without adequately addressing patient experience aspects such as appointment transparency, informed decision-making, and seamless communication, leading to dissatisfaction.

Replacing existing systems with a comprehensive and patient-centric appointment management solution is crucial to address these shortcomings and revolutionize the healthcare experience for both patients and providers.

2.5. Chapter Summary

This chapter delved into the landscape of existing appointment management systems, highlighting their limitations and the need for a revolutionary solution. Existing systems, such as Marham.pk, Oladoc, and InstaCare, have contributed to healthcare accessibility but face drawbacks, including inefficient scheduling, limited transparency, and inadequate support. These shortcomings hinder patient satisfaction and clinic efficiency, emphasizing the need for a comprehensive and patient-centric solution. The Quick Care system aims to address these gaps and transform the healthcare experience.

Chapter 3

REQUIREMENT ENGINEERING

3.1. Proposed System

The proposed system revolutionizes the healthcare experience by providing a patient-centric platform that streamlines appointment scheduling, enhances transparency, facilitates informed decision-making and optimizes clinic efficiency. Its user-friendly interface simplifies appointment booking and management while real-time updates keep patients informed about their scheduled times. Comprehensive doctor information empowers informed decision-making. By reducing no-shows and streamlining patient flow, Quick Care enhances clinic productivity, leading to improved patient care and overall satisfaction.

3.2. Understanding the System

In the dynamic healthcare landscape, the ability to navigate appointments effectively is crucial for both patients and providers. The Quick Care appointment management system emerges as a revolutionary solution, addressing the limitations of existing systems and redefining the healthcare experience. This chapter delves into the intricacies of the Quick Care system, exploring its functionalities, benefits, and transformative impact on the healthcare ecosystem.

3.2.1. User Involvement

Understanding the system through user eyes is crucial. We actively involve users from the start, conducting interviews and workshops to grasp their needs and pain points. By observing their workflows and iterating on prototypes, we ensure the final product resonates with them, fostering adoption and satisfaction. User involvement isn't just a formality; it's a strategic decision that builds trust, minimizes risk, and ultimately delivers a system that truly serves its purpose.

3.2.2. Stakeholders

The Quick Care appointment management system acknowledges that its success is not solely the responsibility of its developers but rather a collaborative endeavor involving key stakeholders. By actively engaging with the following stakeholders, the system can

revolutionize healthcare appointment management and deliver a seamless patient-centric experience for all:

- Patients
- Healthcare Providers
- System Developers
- Healthcare Organizations
- Policymakers and Regulatory Bodies

3.2.3. Domain

The Quick Care appointment management system is firmly rooted in the complexities of the healthcare domain, addressing the specific needs and challenges faced by healthcare providers and patients. Key areas of focus include:

- Effective patient management
- Streamlined appointment scheduling
- Comprehensive doctor profiles
- Optimized clinic workflows
- AI-powered communication and support

By understanding the nuances of the healthcare domain, Quick Care has developed a comprehensive system that revolutionizes the healthcare experience for all.

3.2.4. Needs of System

The Quick Care appointment management system is designed to meet the ever-changing needs of the healthcare ecosystem, ensuring its continued relevance and effectiveness. The system's patient-centric approach, data-driven insights, and adaptability to new technologies position it as a transformative solution, revolutionizing the patient experience and enhancing healthcare delivery.

Table 3.1 User Needs of System

SR #	Needs	Need ID
1	Secure Access	ID-01
2	Account Creation	ID-02

3	Doctor Accreditation	ID-03
4	Appointment Management	ID-04
5	Doctor Availability	ID-05
6	Search and Discovery	ID-06
7	Patient Data Management	ID-07
8	Personalization	ID-08
9	Communication	ID-09
10	Payment Processing	ID-10
11	Dispute Resolution	ID-11
12	Transparency and Feedback	ID-12

3.3. Requirement Engineering

The requirement engineering process for the Quick Care appointment management system involved a comprehensive approach to gathering, analysing, and documenting the system's functional and non-functional requirements. This process ensured that the system aligned with the needs and expectations of both patients and healthcare providers.

3.3.1. Functional Requirements

The Quick Care appointment management system is built upon a robust foundation of functional requirements that address the needs of both patients and healthcare providers. These requirements encompass a range of functionalities, including secure user authentication, streamlined appointment management, effective communication channels, comprehensive patient records management, and secure payment processing. Each requirement has been meticulously crafted to ensure a seamless and user-centric experience for all stakeholders. The comprehensive functional requirements serve as the cornerstone of the Quick Care system, ensuring its effectiveness in addressing the challenges and opportunities of the healthcare industry. By prioritizing user needs and streamlining processes, the Quick Care system is poised to revolutionize the way healthcare is delivered and experienced.

Table 3.2 Functional Requirement 01

Functional Requirement ID	FR-01
Name	Login
Description	This requirement provides the functionality to log in the user into the system
Input	Username and password
Output	Homepage will be displayed
Precondition	User must be registered
Postcondition	The user must be logged in

Table 3.3 Functional Requirement 02

Functional Requirement ID	FR-02
Name	Patient Registration
Description	This requirement provides the functionality to register the user into the system
Input	A new username and password
Output	New account registered
Precondition	User must not be registered already
Postcondition	Account must be created and user redirected to home

Table 3.4 Functional Requirement 03

Functional Requirement ID	FR-03
Name	Doctor registration

Description	This requirement allows the registration of doctors after verification to avoid spam
Input	Data required for verification
Output	Doctor account is sent for verification
Precondition	Correct and valid data must be provided
Postcondition	Account registration request has been accepted

Table 3.5 Functional Requirement 04

Functional Requirement ID	FR-04
Name	Doctor registration approval on admin dashboard
Description	This requirement allows the ability to approve or deny new doctor's registration
Input	Doctor's registration request
Output	Request is accepted or denied
Precondition	Admin must be logged in
Postcondition	Doctor is informed based on the response given

Table 3.6 Functional Requirement 05

Functional Requirement ID	FR-05
Name	Two Factor Authentication
Description	This allows extra layer of security by requiring One time password at login
Input	A unique one-time password
Output	Login allowed
Precondition	User must be registered using a phone number or email

Postcondition	User is logged in

Table 3.7 Functional Requirement 06

Functional Requirement ID	FR-06
Name	Authentication with Google
Description	This requirement provides the functionality to register & login using google account
Input	Allow using the google authentication pop-up
Output	Account register/login from google account
Precondition	User must have a google
Postcondition	User is logged in/Registered

Table 3.8 Functional Requirement 07

Functional Requirement ID	FR-07
Name	Book appointments
Description	This requirement provides the ability to book doctor appointments
Input	Doctor and time of choice
Output	Book appointment
Precondition	User must be registered and logged in
Postcondition	Appointment is booked or cancelled

Table 3.9 Functional Requirement 08

Functional Requirement ID	FR-08
Name	Cancel appointments

Description	This requirement provides the ability to cancel doctor appointments
Input	Appointment to be cancelled
Output	Appointment is cancelled
Precondition	User must be registered and logged in
Postcondition	Appointment is cancelled

Table 3.10 Functional Requirement 09

Functional Requirement ID	FR-09
Name	Display appointment status
Description	This requirement provides the live status of the doctor's current patient and the user's expected appointment time
Input	Booked appointment
Output	Show live status on home screen
Precondition	Appointment must be booked
Postcondition	Status of appointment gets displayed

Table 3.11 Functional Requirement 10

Functional Requirement ID	FR-10
Name	Location based status update for doctor
Description	This requirement allows the live status to start working and inform patients when doctor arrives at their hospital/clinic
Input	Location allowed
Output	Detects when doctor reaches or leaves the clinic
Precondition	Location access must be allowed

Postcondition	Starts/Stops the real-time appointments status

Table 3.12 Functional Requirement 11

Functional Requirement ID	FR-11
Name	Location based search
Description	This requirement allows patients to search doctors according to a specific location
Input	Location
Output	All doctors from that area
Precondition	User must be registered
Postcondition	Shows list of doctors available in the selected area

Table 3.13 Functional Requirement 12

Functional Requirement ID	FR-12
Name	Doctor-side local patient admission
Description	This functionality allows doctor to manually add local patients to the database so the live status in app shows correct status and time
Input	Local patient information
Output	Live status accounts the local patients as well
Precondition	Doctor & staff must be aware of the benefit of this requirement
Postcondition	The appointment list also contains locally booked patients

Table 3.14 Functional Requirement 13

Functional Requirement ID	FR-13

Name	Update User Profile for patients & doctors
Description	This requirement provides the ability to manage personal information for both patients and doctors
Input	Personal information
Output	Updated profile is displayed
Precondition	User must be registered
Postcondition	Personal information is updated

Table 3.15 Functional Requirement 14

Functional Requirement ID	FR-14
Name	Email reminder
Description	This requirement provides reminders to the patient for their appointments
Input	User consent
Output	Email reminder is sent
Precondition	User should have an email address
Postcondition	Users receives email reminders

Table 3.16 Functional Requirement 15

Functional Requirement ID	FR-15
Name	Chatbot integration
Description	This requirement allows user to talk to a chatbot to discuss any issues or get queries answered
Input	User's issue/query
Output	The most optimal response

Precondition	User must be registered
Postcondition	User must receive a response

Table 3.17 Functional Requirement 16

Functional Requirement ID	FR-16
Name	Payment method
Description	This requirement allows users to pay using debit card, credit card & Jazzcash wallet
Input	User's payment information
Output	Payment received via the selected method
Precondition	User must've booked an appointment
Postcondition	User must be charged

Table 3.18 Functional Requirement 17

Functional Requirement ID	FR-17
Name	Refund system
Description	This requirement allows users to refund payment if they cancel appointment in allowed time or if it gets cancelled by the doctor
Input	Appointment information
Output	Payment refund
Precondition	Appointment should be cancelled in allowed time or by doctor
Postcondition	Appointment is cancelled

Table 3.19 Functional Requirement 18

Functional Requirement ID	FR-18

Name	Review system
Description	This requirement allows users to review doctor based on their appointment to let other patients know what to expect
Input	Review
Output	Review will be displayed on doctor's profile
Precondition	User must've have booked and attended the appointment
Postcondition	Review is published

Table 3.20 Functional Requirement 19

Functional Requirement ID	FR-19
Name	Create records
Description	This requirement provides ability to create record of patient's old appointments
Input	The appointment's result or description
Output	Record can be accessed from profile anytime
Precondition	Patient must've attended an appointment
Postcondition	Patient record is saved and can be displayed

Table 3.21 Functional Requirement 20

Functional Requirement ID	FR-20
Name	Update records
Description	This requirement provides ability to update the saved records of the patient
Input	Changes in the result or description
Output	Record is updated

Precondition	Patient must've saved a record
Postcondition	Patient record is updated

Table 3.22 Functional Requirement 21

Functional Requirement ID	FR-21
Name	Doctor status update
Description	This requirement allows doctor to update his status especially if doctor is on leave and all patients can be informed
Input	Doctor's status
Output	Information is relayed to all patients
Precondition	Doctor must have booked appointments
Postcondition	All patients with that doctor's appointments are notified

Table 3.23 Functional Requirement 22

Functional Requirement ID	FR-22
Name	Notification System
Description	This requirement will show all notifications for the app
Input	NIL
Output	Shows all past notifications
Precondition	User must be registered
Postcondition	All notifications are displayed

3.3.2. Non-Functional Requirements

In addition to its comprehensive functional requirements, the Quick Care appointment management system adheres to a set of stringent non-functional requirements to ensure its reliability, performance, usability, and accessibility. These requirements address critical

aspects of the system's operation, ranging from security measures to performance optimization, ensuring a seamless experience for all users. Quick Care leverages the power of Flutter and Firebase to deliver a robust and user-friendly appointment management system.

- **Uninterrupted Operations:** Quick Care delivers a seamless and responsive experience across platforms thanks to its native code rendering. Schedule appointments, manage details, and access information without any lag or delay. Firebase Cloud Firestore keeps you updated with real-time data updates. Appointments, schedules, and changes are reflected instantly, ensuring everyone is always on the same page. Even when the internet is unavailable, you can still access and manage your appointments with Quick Care's offline functionality. Schedule, reschedule, and stay on top of your healthcare needs, regardless of your internet connection.
- **Security and Privacy:** Quick Care prioritizes security and privacy by adopting robust authentication protocols to secure your account. Sensitive information, from patient records to appointment details, is encrypted both in transit and at rest, ensuring utmost privacy and security. Granular access control further safeguards your data by allowing only authorized users to view and modify specific information.
- **Intuitive and User-friendly:** Quick Care boasts a user-friendly interface built with Flutter's rich UI components, making navigation intuitive and scheduling straightforward. Managing appointments becomes a breeze with this easy-to-use interface. Quick Care also features dynamic content updates driven by Firebase Remote Config, ensuring the best possible user experience tailored to your individual needs. Say goodbye to confusing error messages with Quick Care's clear and concise information within the app, guiding you through troubleshooting and resolving issues effectively.
- **Experience Consistency and Adaptability:** Access Quick Care from any Android or iOS device you prefer. Flutter's native code ensures consistent performance and functionality across platforms. Whether you're using a smartphone or a tablet, Quick Care adapts to your device's screen size, providing an optimal and comfortable experience. As your needs evolve and your appointment volume increases, Quick Care adapts seamlessly with its scalable architecture. Firebase's cloud-based

infrastructure automatically scales to accommodate your growth, ensuring smooth operation even during peak times.

- **Easy Maintenance and Ongoing Development:** Quick Care facilitates rapid implementation of new features and updates with Flutter's hot reload functionality, keeping it at the forefront of innovation. The system's modular code structure and clear separation of concerns make maintenance and modification a breeze, ensuring continuous improvement and bug fixes. Firebase's version control and deployment tools further simplify the update process, guaranteeing smooth and reliable operation with minimal downtime.
- **Accessible for Everyone:** Quick Care adheres to WCAG guidelines, ensuring accessibility for users with diverse needs, including those who rely on assistive technologies. Navigate the app and access information easily with keyboard navigation and built-in support for screen readers. Quick Care also provides alternative text descriptions for all images and non-text content, making information accessible to everyone, regardless of their visual abilities.
- **Optimized for Performance:** Quick Care utilizes code profiling and Flutter DevTools to identify and address potential performance bottlenecks, ensuring fast loading times and smooth operation. Images and content are loaded only when needed, minimizing resource utilization and optimizing performance for a seamless experience. Even in low-connectivity environments, Quick Care's offline functionality ensures responsiveness and performance, allowing you to manage your appointments without interruption.
- **Data Integrity and Reliability:** Firebase Realtime Database's data validation rules ensure the accuracy and consistency of information within Quick Care. Regular data backups and synchronization through Firebase Cloud Storage provide an extra layer of protection, minimizing the risk of data loss. Quick Care actively monitors performance and identifies potential data inconsistencies, guaranteeing the reliability and integrity of your information.

By combining the strengths of Flutter and Firebase, Quick Care delivers a robust, scalable, and user-friendly appointment management system that caters to the needs of both patients and healthcare providers. From scheduling appointments to managing your health information,

Quick Care provides a seamless and reliable experience, making it the ideal platform for managing your healthcare journey.

These non-functional requirements form the foundation of a robust and user-centered system, ensuring that the Quick Care appointment management system delivers a seamless and reliable experience for all stakeholders in the healthcare ecosystem.

3.3.3. Requirements Baseline

The requirements baseline serves as a crucial reference point throughout the development of the Quick Care appointment management system. It encapsulates the agreed-upon functional and non-functional requirements, defining the system's scope, features, and quality standards. This baseline establishes a shared understanding among stakeholders, ensuring alignment with the needs of both patients and healthcare providers.

This comprehensive documentation guides the development team, ensuring adherence to the agreed-upon specifications. The requirements baseline also facilitates effective communication among stakeholders and serves as the foundation for testing and validation activities.

3.3.4. Need to Feature Mapping

Feature mapping is a collaborative technique that helps teams break down complex features into smaller, more manageable tasks, identify potential edge cases, and uncover hidden assumptions. It is a valuable tool for ensuring that the Quick Care appointment management system meets the needs of its users and delivers a seamless experience.

Table 3.24 Need to Feature Mapping

Needs / Features	ID-01	ID-02	ID-03	ID-04	ID-05	ID-06	ID-07	ID-08	ID-09	ID-10	ID-11	ID-12
FR-01	✓	-	-	-	-	-	-	-	-	-	-	-
FR-02	-	✓	-	-	-	-	-	-	-	-	-	-
FR-03	-	✓	-	-	✓	-	-	-	-	-	-	-
FR-04	-	-	✓	-	-	-	-	-	-	-	-	-
FR-05	✓	-	-	-	-	-	-	-	-	-	-	-
FR-06	✓	✓	-	-	-	-	-	-	-	-	-	-
FR-07	-	-	-	✓	-	-	-	-	-	-	-	-
FR-08	-	-	-	✓	-	-	-	-	-	-	-	-
FR-09	-	-	-	✓	-	-	-	-	✓	-	-	-
FR-10	-	-	-	✓	✓	-	-	-	-	-	-	-
FR-11	-	-	-	-	-	✓	-	-	-	-	-	-

FR-12	-	-	-	-	-	-	✓	-	-	-	-	-
FR-13	-	-	-	-	-	-	-	✓	-	-	-	-
FR-14	-	-	-	-	-	-	-	-	✓	-	-	-
FR-15	-	-	-	-	-	-	-	-	✓	-	-	-
FR-16	-	-	-	-	-	-	-	-	-	✓	-	-
FR-17	-	-	-	-	-	-	-	-	-	✓	✓	-
FR-18	-	-	-	-	-	-	-	-	-	-	-	✓
FR-19	-	-	-	-	-	-	✓	-	-	-	-	-
FR-20	-	-	-	-	-	-	✓	-	-	-	-	-
FR-21	-	-	-	-	✓	-	-	-	✓	-	-	-
FR-22	-	-	-	-	-	-	-	-	✓	-	-	-

Feature mapping has proven effective in developing successful products that meet user needs. Its implementation in the Quick Care appointment management system would be highly beneficial.

3.4. Gantt Chart

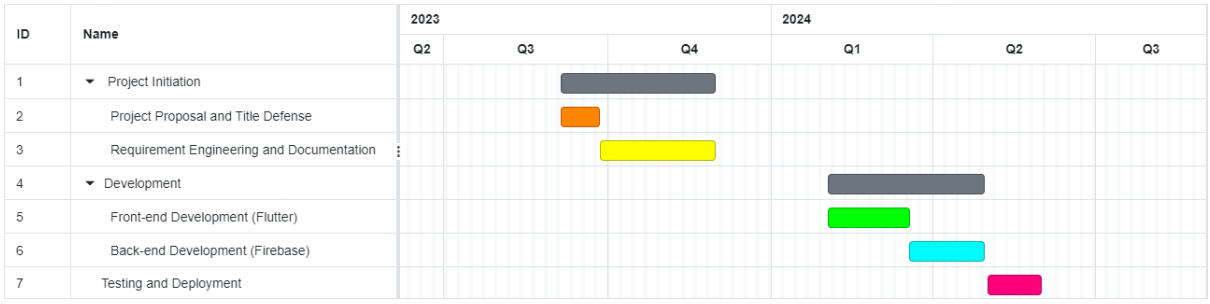

Figure 3.1 Gantt Chart

3.5. Hurdles in Optimizing the Current System

Despite the significant advancements in technology and healthcare practices, optimizing the current healthcare system remains a complex and multifaceted challenge. Several hurdles impede the path towards a truly efficient, patient-centred, and cost-effective healthcare delivery model.

- **Silos of Information and Lack of Interoperability:** Fragmented healthcare data across institutions, providers, and EHRs creates silos of information that hinder comprehensive patient care and data sharing.
- **Inefficient Processes and Workflows:** Outdated processes, manual data entry, paper-based records, and fragmented communication channels contribute to delays, errors, and unnecessary administrative burdens.

- **Limited Patient Engagement:** Lack of access to health records, unclear communication from providers, and inadequate health literacy hinder patient engagement and informed decision-making.
- **Fee-for-Service Payment Model:** Fee-for-service incentivizes volume over value, driving up costs without improving patient outcomes. It fails to address preventive care and chronic disease management.
- **Social Determinants of Health:** Social factors like socioeconomic status, education, and access to healthy food and housing significantly impact health outcomes. Addressing these factors requires collaboration beyond traditional healthcare delivery.

3.6. Chapter Summary

This chapter delves into the proposed healthcare management system, examining its core components, user involvement, stakeholders, domain, and underlying needs. It also explores the intricacies of requirement engineering, functional requirements, non-functional requirements, and the establishment of a requirements baseline. Furthermore, it highlights the importance of feature mapping and the utilization of Gantt charts for project management. Finally, it acknowledges the potential hurdles that may arise during system optimization, such as data security, integration complexities, high usage demands, and the need for adaptability.

Chapter 4

DESIGN

4.1. Software Process Model

Software Processes is a coherent set of activities for specifying, designing, implementing and testing software systems. A software process model is an abstract representation of a process that presents a description of a process from some particular perspective. There are many different software processes, but all involve:

1. Specification – defining what the system should do;
2. Design and implementation – defining the organization of the system and implementing the system;
3. Validation – checking that it does what the customer wants;
4. Evolution – changing the system in response to changing customer needs.

4.1.1. Incremental Model

The Quick Care appointment management system will be built using the incremental model, a process that emphasizes building software in manageable phases. Instead of tackling the entire system at once, it will be meticulously deconstructed into smaller increments, each delivering a distinct set of functionalities.

This approach allows for focused development, with the initial increment laying the foundation by establishing core features like user registration, appointment booking, doctor profiles, and basic appointment management tools. Subsequent increments will add features like online payments, medical record management, and advanced appointment management tools, ensuring a smooth and seamless user experience throughout the app's growth.

Each increment undergoes rigorous testing and evaluation, providing valuable user feedback that actively shapes the app's future. This continuous feedback loop fuels the development of future increments, ensuring that the Quick Care app adapts and improves alongside the changing needs of its users and the healthcare landscape.

By embracing the incremental model, Quick Care fosters a collaborative environment between developers and users, enabling the development team to continuously refine and

optimize the app to provide a convenient, efficient, and intuitive solution for managing healthcare appointments.

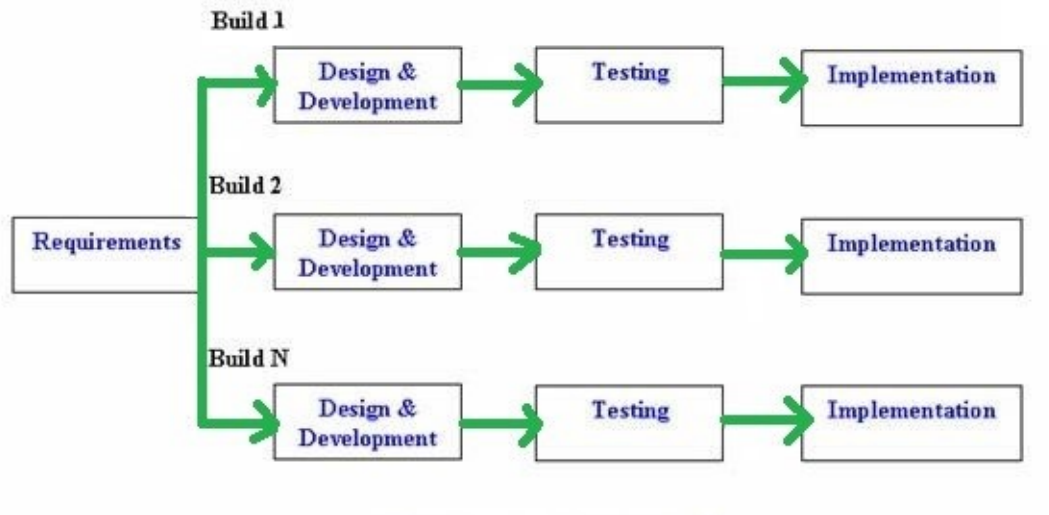

Figure 4.1 Incremental Model [3]

4.1.2. Benefits of Selected Model

The incremental model offers several distinct advantages for the development of the Quick Care appointment management system:

1. Reduced Risk: Dividing the project into smaller, independent increments significantly reduces the risk of major project failures. This allows for early identification and resolution of issues, minimizing their potential impact and ensuring smoother development progress.
2. Early User Feedback: Each increment provides valuable opportunities for user feedback, enabling developers to incorporate user needs and preferences into future iterations. This ensures the app continuously evolves to meet user expectations, leading to a user-centric and ultimately successful product.
3. Improved User Experience: Focusing on core functionalities in initial releases allows users to benefit from the app early on. This fosters trust and engagement, establishing a solid foundation for future enhancements and new features, ultimately leading to a more user-friendly experience.
4. Increased Agility: The incremental model promotes adaptability. It allows the app to readily respond to changing requirements and market conditions, ensuring its relevance and value to users in the evolving healthcare landscape.

4.1.3. Limitations of Selected Model

While the incremental model offers significant benefits, it also presents some limitations to consider:

1. Delayed Full Functionality: Delivering features in stages can lead to initial user frustration, as comprehensive functionality might not be available immediately. Carefully managing user expectations is crucial to mitigate this issue.
2. Requirement Management: Maintaining a clear vision and managing changing requirements across multiple increments can be challenging. This requires careful planning, communication, and documentation to ensure seamless integration of new features with existing functionalities.
3. Integration Complexity: Integrating new features seamlessly with existing functionalities requires meticulous planning and execution. Suboptimal integration strategies can lead to technical challenges and hinder user experience.

By acknowledging and effectively addressing these limitations, the Quick Care development team can leverage the strengths of the incremental model to deliver a user-centric and impactful healthcare appointment management system.

4.2. Design

4.2.1. Methodology of the Proposed System

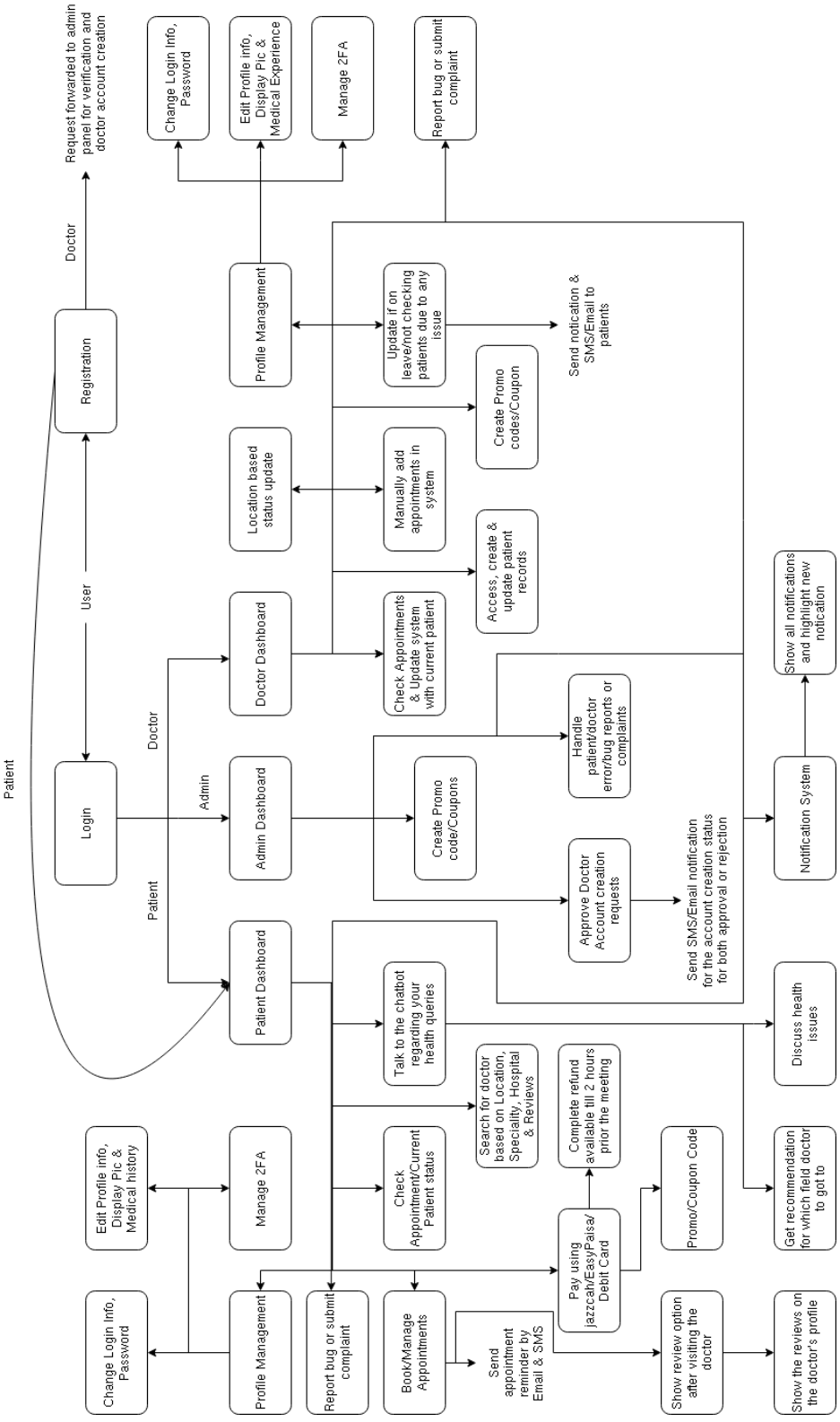

Figure 4.2 Methodology Diagram

4.2.2. Entity Relationship Diagram

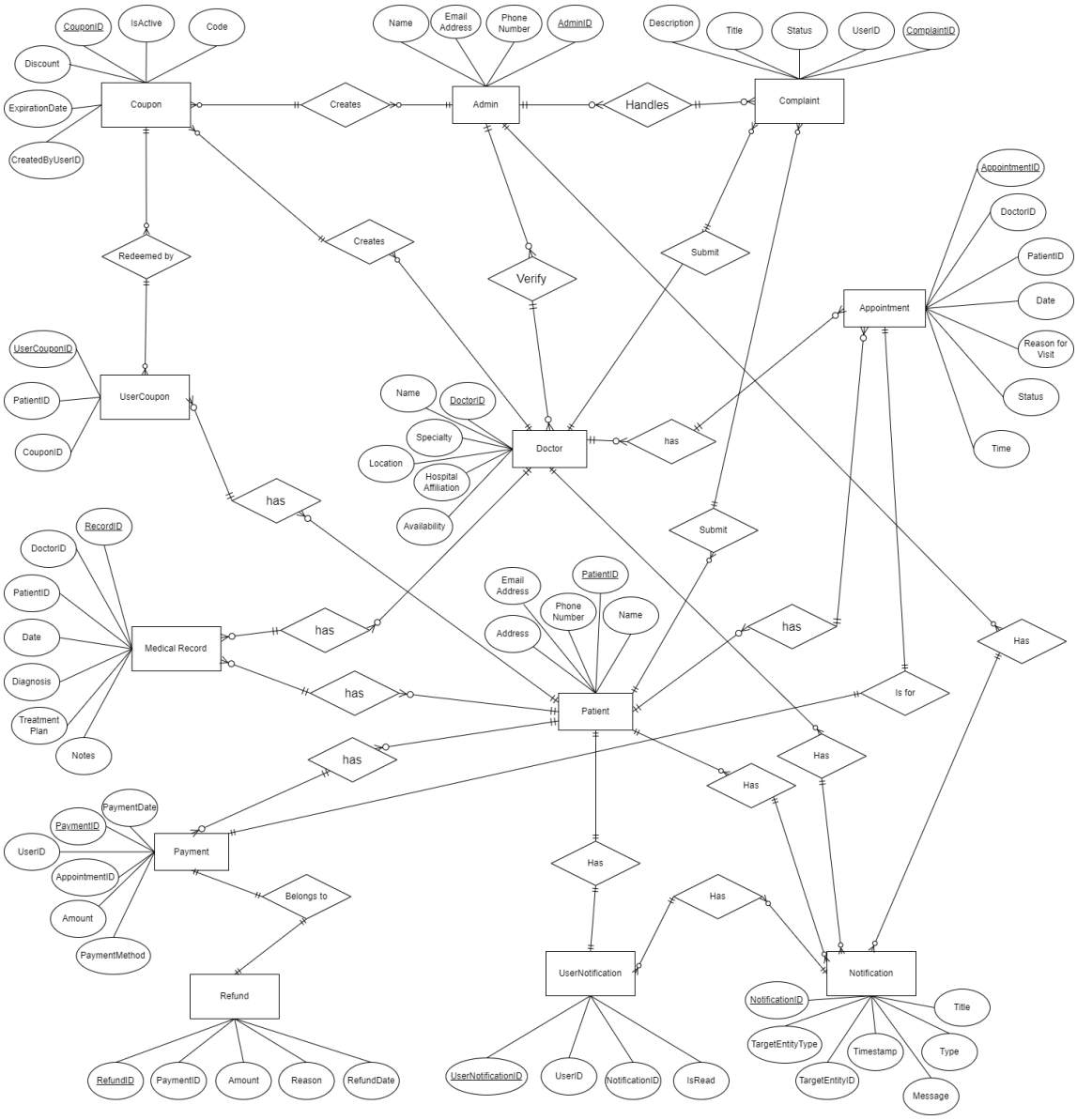

Figure 4.3 Entity Relationship Diagram

4.2.3. UML Diagrams

UML Diagrams, or Unified Modeling Language diagrams, are a standardized set of graphical notations used to visualize and communicate the design and structure of software systems. They serve as a valuable tool for developers, analysts, and other stakeholders to effectively understand, analyze, and improve software systems.

4.2.3.1. *Use Case Diagram of the System*

Figure 4.4 Use Case Diagram

4.2.3.2. *Class Diagram of the System*

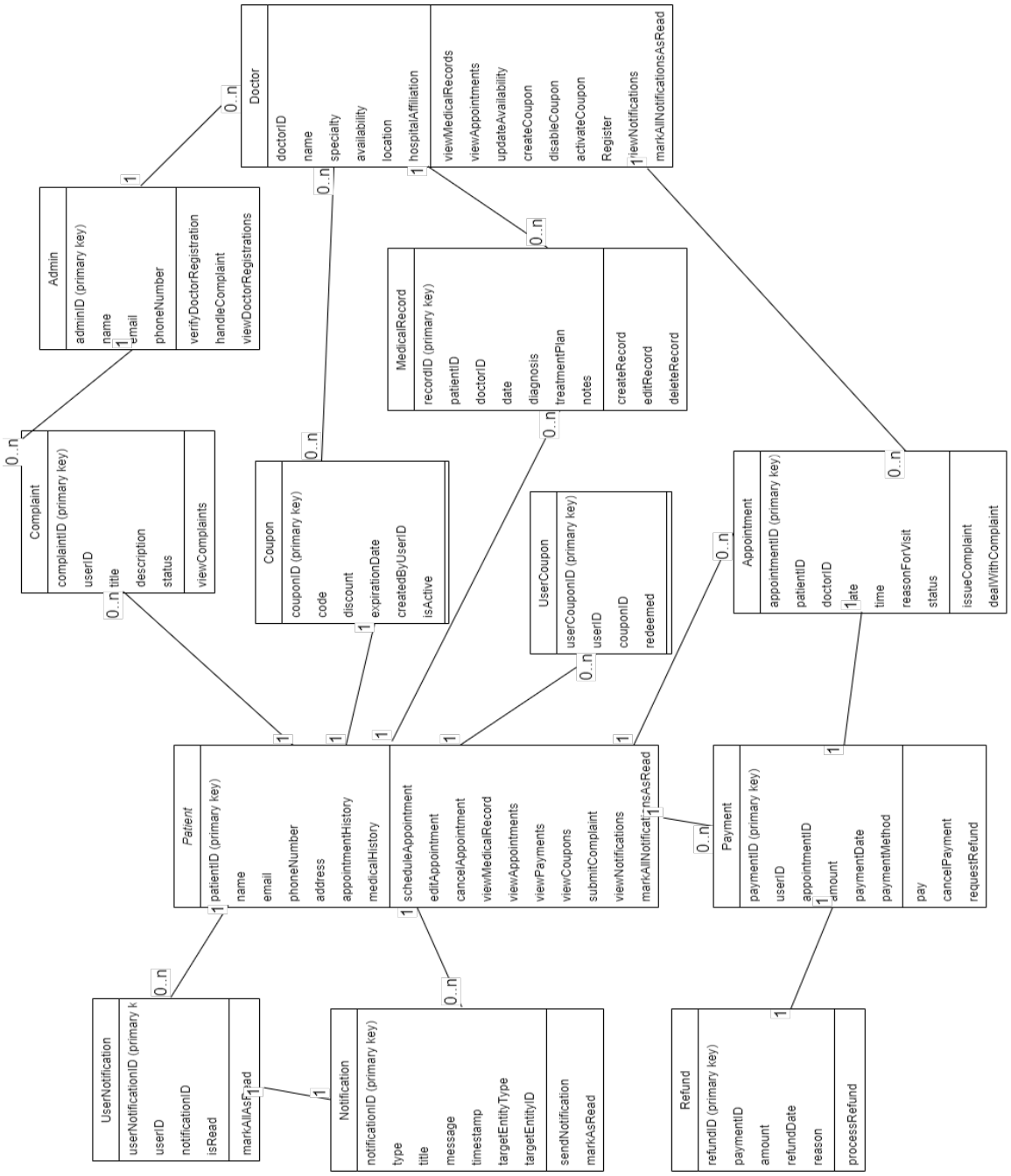

Figure 4.5 Class Diagram

4.2.3.3. *Activity Diagram of the System*

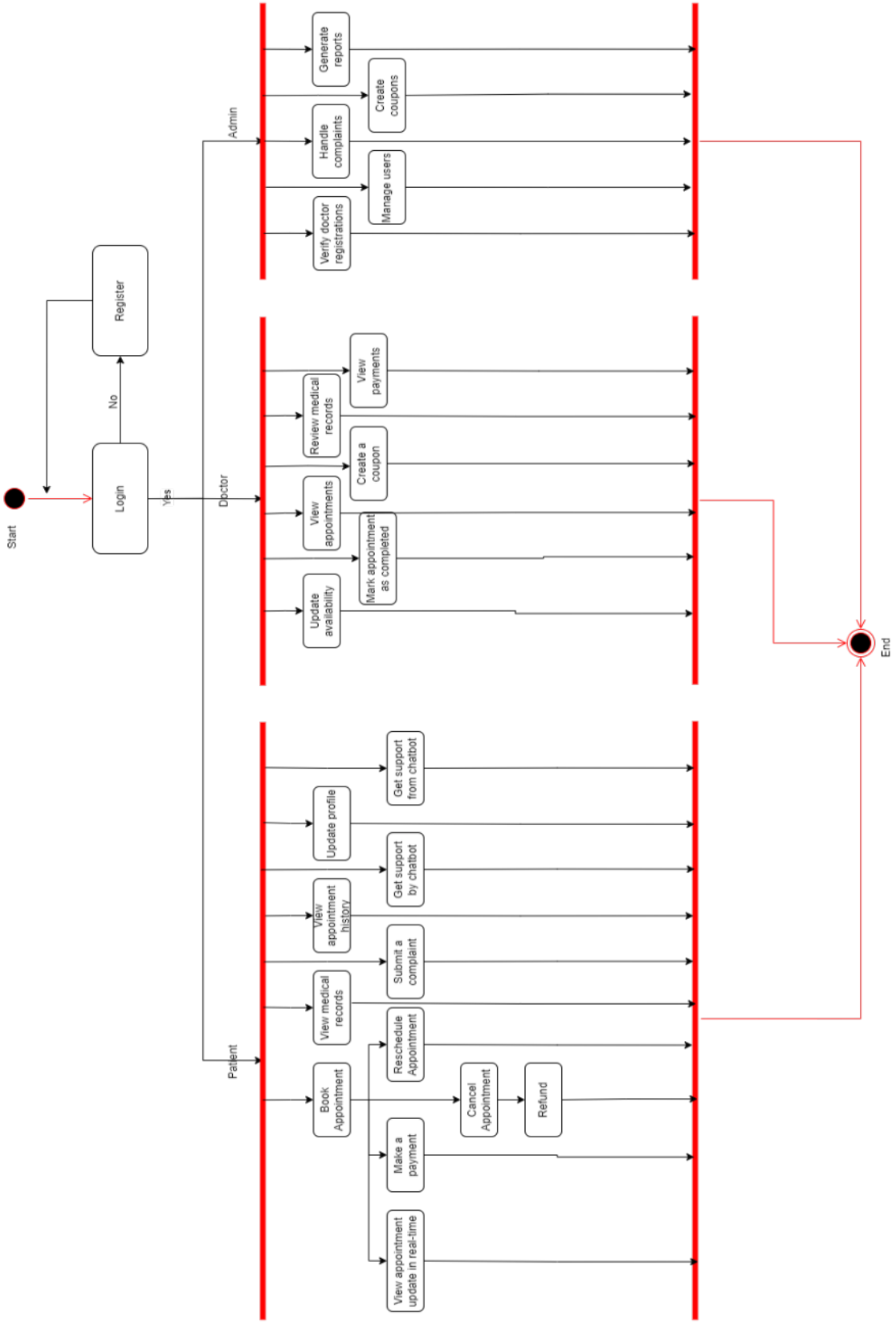

Figure 4.6 Activity Diagram

4.2.3.4. ***Sequence Diagram of the System***

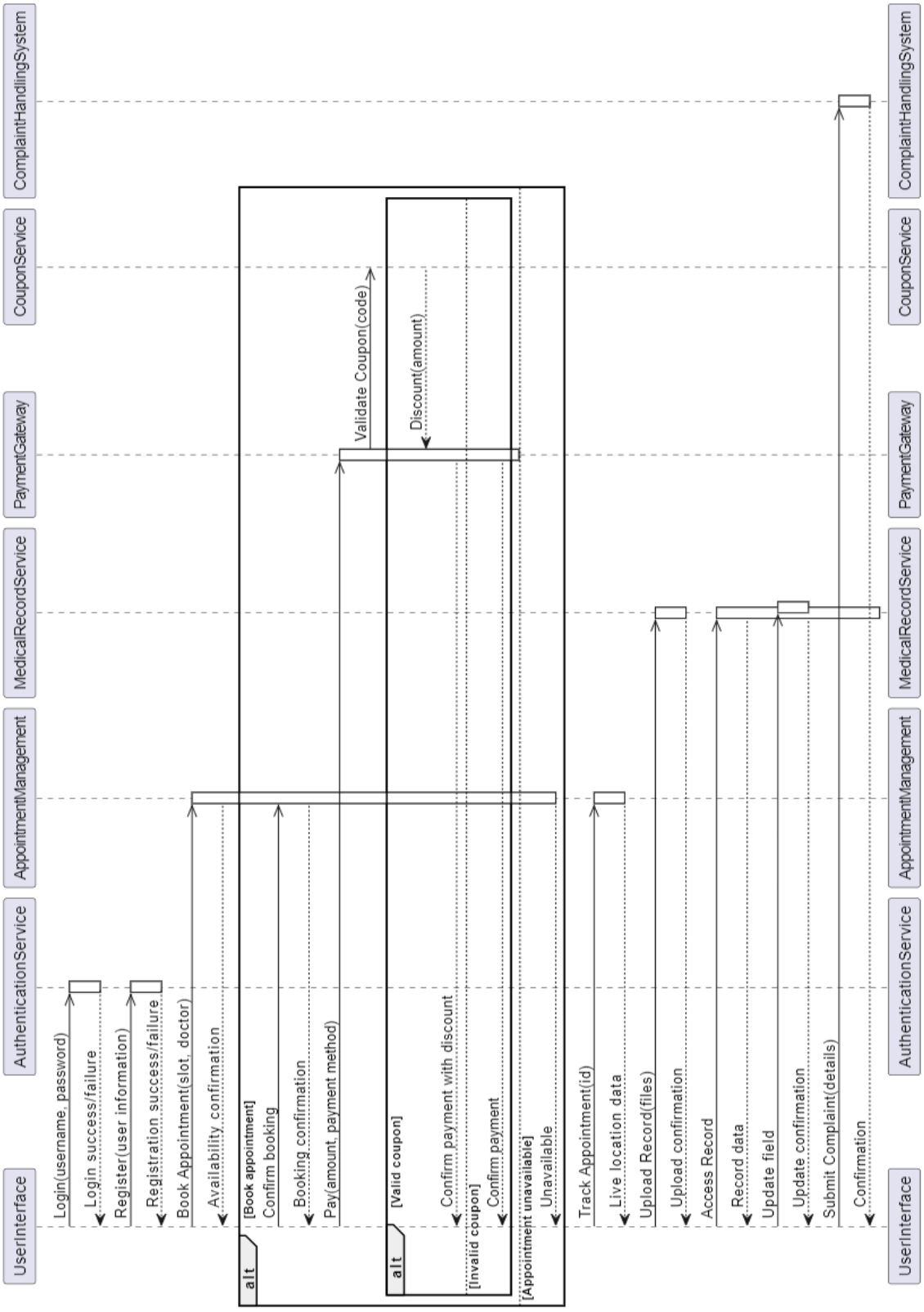

Figure 4.7 Sequence Diagram

4.2.3.5. *Component Diagram of the System*

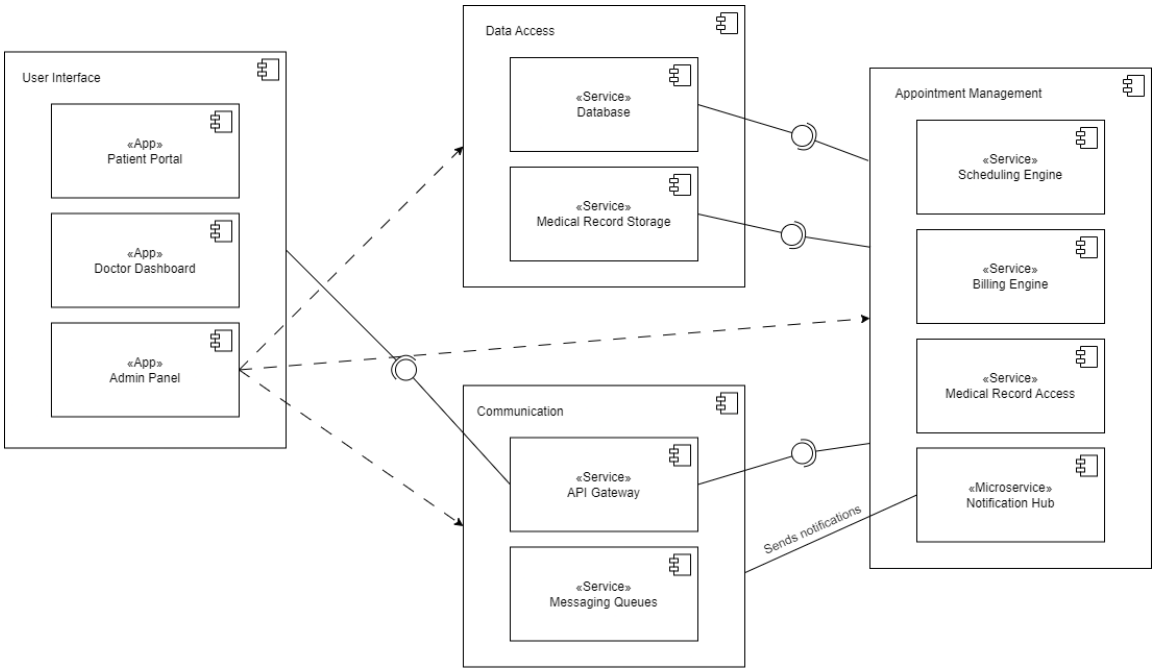

Figure 4.8 Component Diagram

4.3. Chapter Summary

This chapter explored the incremental software development model, highlighting its advantages like early feedback and flexibility, while acknowledging potential drawbacks like integration challenges and documentation burden. Following this approach, the chapter delved into the design of the proposed system, outlining its chosen methodology and providing essential UML diagrams like the use case, class, and sequence diagrams to illustrate functionality, structure, and interaction between components.

Chapter 5

DATABASE

5.1. Database Introduction

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modelled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data. But there are also others that use no structured query language (NoSQL) such as firebase and MongoDB.

5.2. Selected Database

Database: **Firebase Firestore** (NoSQL Document Database)

5.2.1. Reasons for Selection of the Database

- **Developer-Friendliness:** Firestore integrates seamlessly with Flutter, providing a smooth development experience within the Firebase ecosystem.
- **Real-time Updates:** Firestore's real-time synchronization ensures that any changes to appointment schedules, patient records, or payments are reflected instantly across all connected devices. This is vital for maintaining an up-to-date and consistent view of the system.
- **Scalability:** Firestore scales effortlessly, allowing your appointment system to handle a growing number of users and data without performance bottlenecks.
- **Offline Support:** Firestore enables offline functionality, meaning users can interact with crucial appointment information even with intermittent connectivity. Changes sync automatically when the connection is restored.
- **Security:** Firebase provides robust security features, including authentication, authorization, and data encryption to protect sensitive patient health information.

5.2.2. Benefits of the Selected Database

- **Rapid Development:** Firebase Firestore accelerates your development process, allowing you to focus on building the app's core features.
- **Cost-Effective:** Firestore's flexible pricing models and potential for a generous free tier can make it a cost-effective solution, especially during the early phases of your project.
- **Maintenance:** As a managed service, Firebase handles much of the database maintenance and infrastructure, reducing your operational overhead.

5.2.3. Limitations of the Selected Database

- **Complex Queries:** Firestore's querying capabilities, while suitable for many use cases, may have limitations when dealing with highly complex, relational data structures or aggregations.
- **Cost at Scale:** Costs can become a factor as your app scales significantly. Optimizing your data structure and queries will be important to manage costs.
- **Vendor Lock-In:** Firestore is a proprietary solution, meaning you're within the Firebase ecosystem. Migration to a different database can be potentially complex if needed in the future.

5.3. Database Queries

Store Patient record in Firestore:

```
FirebaseFirestore.instance.collection('patients')
        .doc(FirebaseAuth.instance.currentUser?.uid)
        .set({
          'uid': FirebaseAuth.instance.currentUser?.uid,
          'name': name,
          'email':FirebaseAuth.instance.currentUser?.email,
          'display_image': profilePic,
          'dob': dOB,
          'gender': isMale ? 'male' : 'female',
```

```
            'city': city,
            'diseases': diseases,
            'more_diseases': otherDiseases,
            'allergies': allergies,
            'creation_timestamp':DateTime.now()
          })
```

Store Doctor record in Firestore:

```
FirebaseFirestore.instance.collection('doctors')
          .doc(FirebaseAuth.instance.currentUser?.uid)
          .set({
            'uid': FirebaseAuth.instance.currentUser?.uid,
            'name': name,
            'email':FirebaseAuth.instance.currentUser?.email,
            'display_image': profilePic,
            'dob': dOB,
            'gender': isMale ? 'male' : 'female',
            'city': city,
            'specialization': specialization,
            'experience': experience,
            'qualification': qualification,
            'creation_timestamp':DateTime.now()
          })
```

Generate appointment from doctor:

```
FirebaseFirestore.instance.collection('appointments').add({
       'uid': FirebaseAuth.instance.currentUser?.uid,
```

```
        'city': city,
        'location': location,
        'position': position,
        'fee': fee,
        'start_time': startTime,
        'end_time': endTime,
        'days': days,
        'status':'active',
        'creation_timestamp': DateTime.now()
    })
```

Create coupon:

```
FirebaseFirestore.instance.collection('coupons').add({
      'uid': FirebaseAuth.instance.currentUser?.uid,
      'expiry': expiryDate,
      'type': selectedType,
      'amount': amount,
      'code': couponCode,
      'creation_timestamp': DateTime.now(),
      'created_by': getCurrentUserType(),
      'status': 'active',
    })
```

Retrieve all appointments from a city:

```
FirebaseFirestore.instance
                    .collection('appointments')
                    .where("city", isEqualTo: selectedCity)
```

```
.snapshots()
```

5.4. Database Tables

Due to the flexible nature of NoSQL databases, our patient appointment system won't have rigid tables. Instead, we'll use Firestore's concept of 'collections' to organize our data. Here's the list of collections in our database:

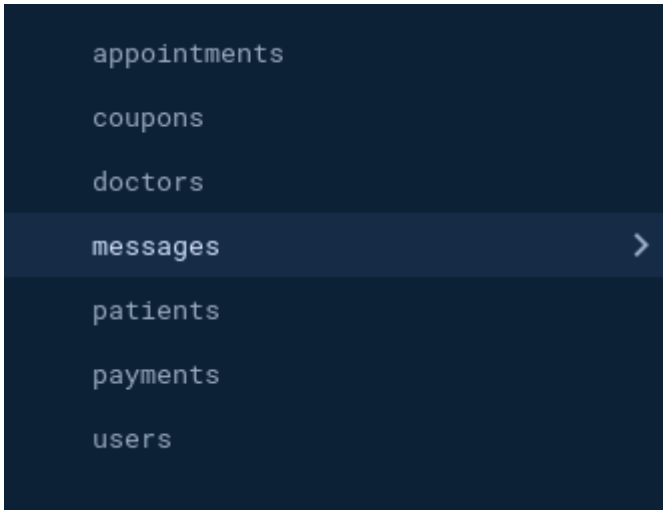

Figure 5.1 Database Collections

5.5. Database Schema Diagram

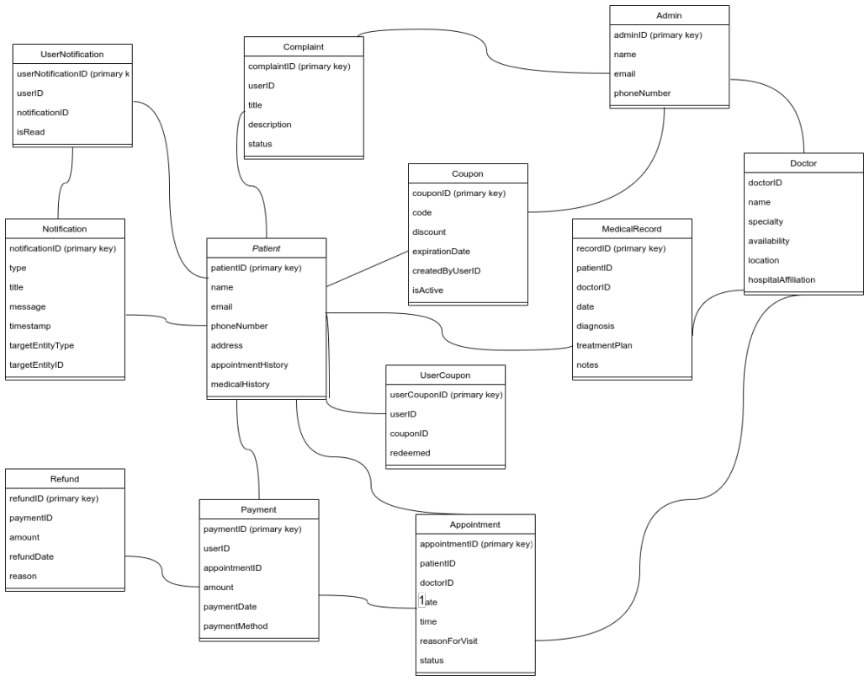

Figure 5.2 Database Schema Diagram

5.6. Chapter Summary

- Our system aligns well with a NoSQL document database like Firebase Firestore due to its scalability, real-time features, and ease of use within the Flutter/Firebase environment.
- Firestore queries focus on filtering and retrieving documents from collections based on their field values.
- Data is organized in collections rather than strict tables, providing flexibility in how you structure your patient, appointment, and provider information.
- While Firestore has limitations for highly complex queries, it suits the core needs of a patient appointment system well.

Important Considerations:

- **Security:** Thoroughly implement Firebase Authentication and Firestore security rules to safeguard sensitive patient information.
- **Data Modeling:** Carefully design your collections and document structures to optimize query performance and maintain data integrity.

Chapter 6

DEVELOPMENT AND IMPLEMENTATION

6.1. Development of the Computer Program

We have adopted the Scrum methodology (an Agile framework) for its iterative nature and adaptability to changing requirements. Sprints will be approximately 2 weeks long.

Development Phases:

- **Planning:** We've meticulously outlined all features (patient booking, doctor view, secure profiles, etc.), designed a Firestore schema, and created detailed UI wireframes.
- **Design:** UI focuses on simple navigation, clear calls to action, and accessible design.
- **Development:** We're building the core appointment logic, integrating Firestore, implementing robust input validation, and setting up real-time data sync.
- **Testing:** Unit tests, integration tests, and UI testing are in progress. User acceptance testing will be scheduled near the release phase.
- **Deployment:** We're preparing app packages for the Google Play Store and Apple App Store, strictly adhering to platform guidelines.
- **Maintenance:** A post-launch plan addresses bug fixes, feature requests based on user feedback, and regular security updates.

6.2. Implementation Strategy

Project Management: We're using a Kanban board (in Jira) to visualize task progress, assign work, and ensure everyone is on the same page.

Team Roles and Responsibilities:

Frontend Developer (1): Build the Flutter UI, handle state management (likely Riverpod), and implement Firestore interactions.

Backend Developer (1): Responsible for Firebase configuration, security rules, user authentication, and potentially Cloud Function development.

6.3. Tools Selection

6.3.1. Hardware

Development Environment: Developers are using MacBook Pro laptops (M1 processors, 16GB RAM) for smooth emulator use and to run Android Studio/VS Code efficiently.

Testing Devices: We have a range of devices:

- Android: Samsung Galaxy S22, Google Pixel 6, OnePlus 9 (different screen sizes/OS versions)
- iOS: iPhone 13, iPhone 12 Mini, iPad Pro

6.3.2. Software

Development Tools:

- IDEs: Android Studio and VS Code
- Flutter SDK
- Firebase CLI
- State Management: Riverpod

Design Tools:

- Figma for collaborative interface design.

Collaboration Tools:

- Git (for version control)
- Slack (for team communication)
- Jira (task management and Kanban tracking)

Testing Tools:

- Flutter's unit testing framework
- A suitable Firebase integration testing library

6.4. Coding

- **Language and Framework:** We'll use Dart for its strong typing and the Flutter framework for its cross-platform efficiency and expressive UI building capabilities.
- **Coding Standards:** The team will adhere to the official Dart style guide and Flutter's recommended patterns for clean, maintainable code.

- **Code Reviews:** A mandatory pull-request system will be implemented, requiring at least one other developer's review before code merges to ensure quality and shared knowledge.

6.5. User Interface

6.5.1. Description

- **Design Principles:** Paramount focus on usability, clear navigation, and visual hierarchy to guide users effortlessly. Accessibility standards (WCAG) will be considered for color contrast, text sizing, and screen reader compatibility.
- **Target Audience:** UI tailored to patients (potentially less tech-savvy), doctors (needing efficient workflows), and admin staff (who may manage complex data entry).
- **Aesthetic Considerations:** A clean, professional visual style suitable for a medical app. Color choices will be soothing yet informative.

6.5.2. Interface Screenshots

Figure 6.1 Start Screen

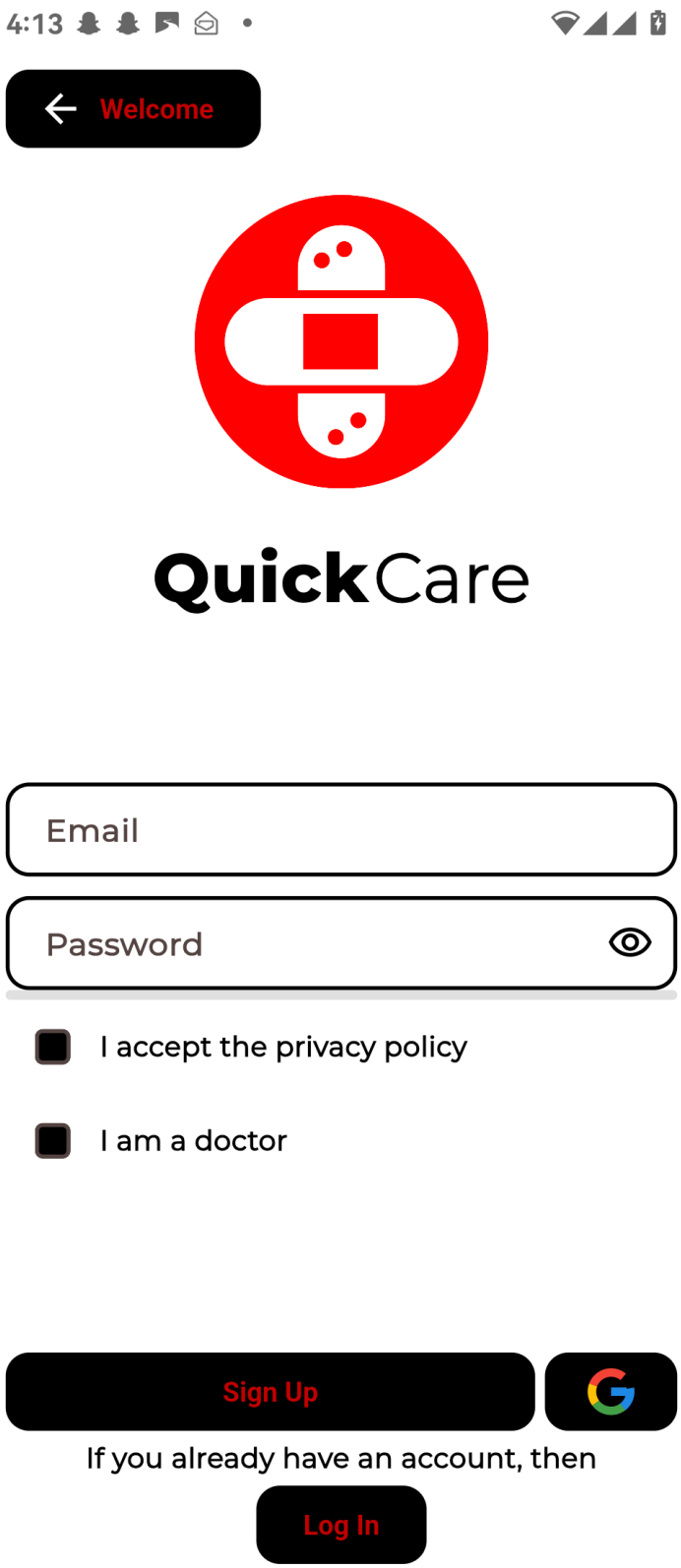

Figure 6.2 Sign-up Screen

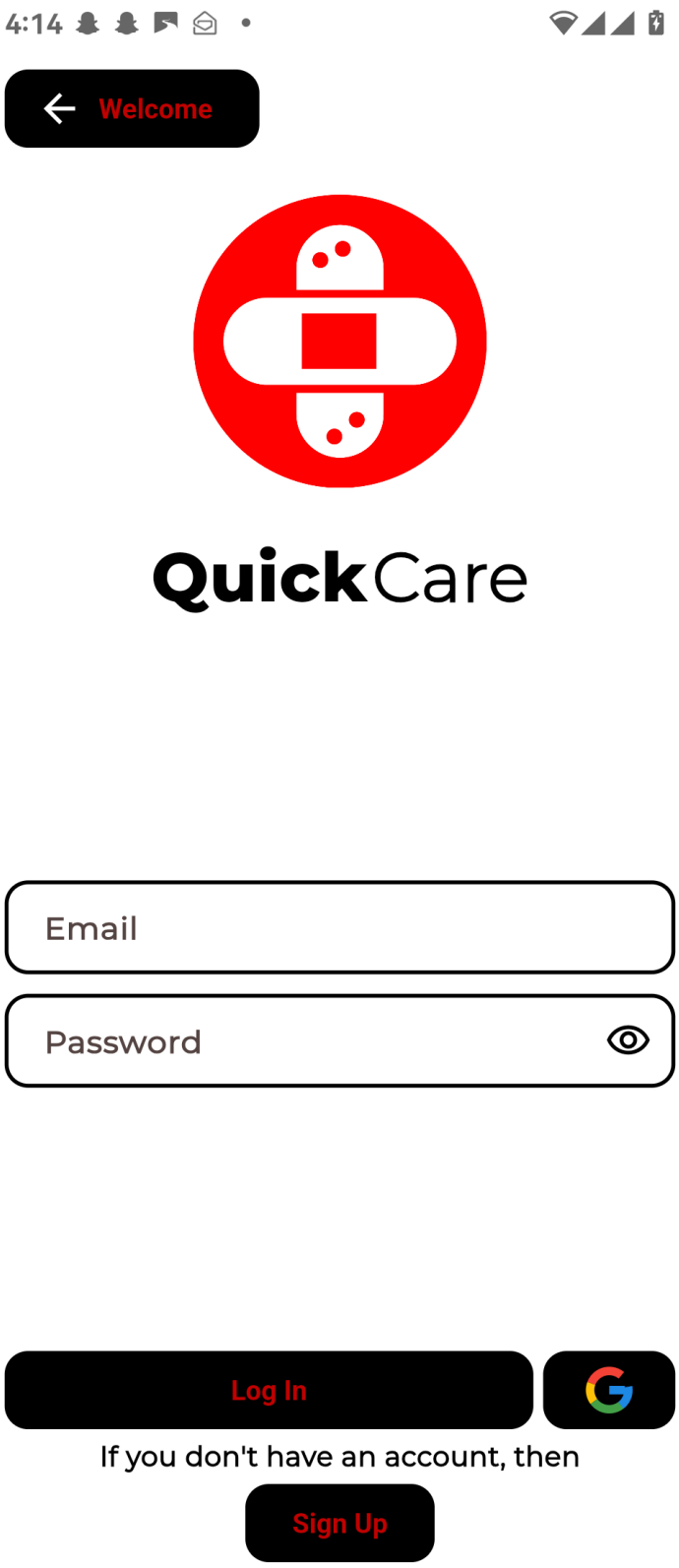

Figure 6.3 Log-in Screen

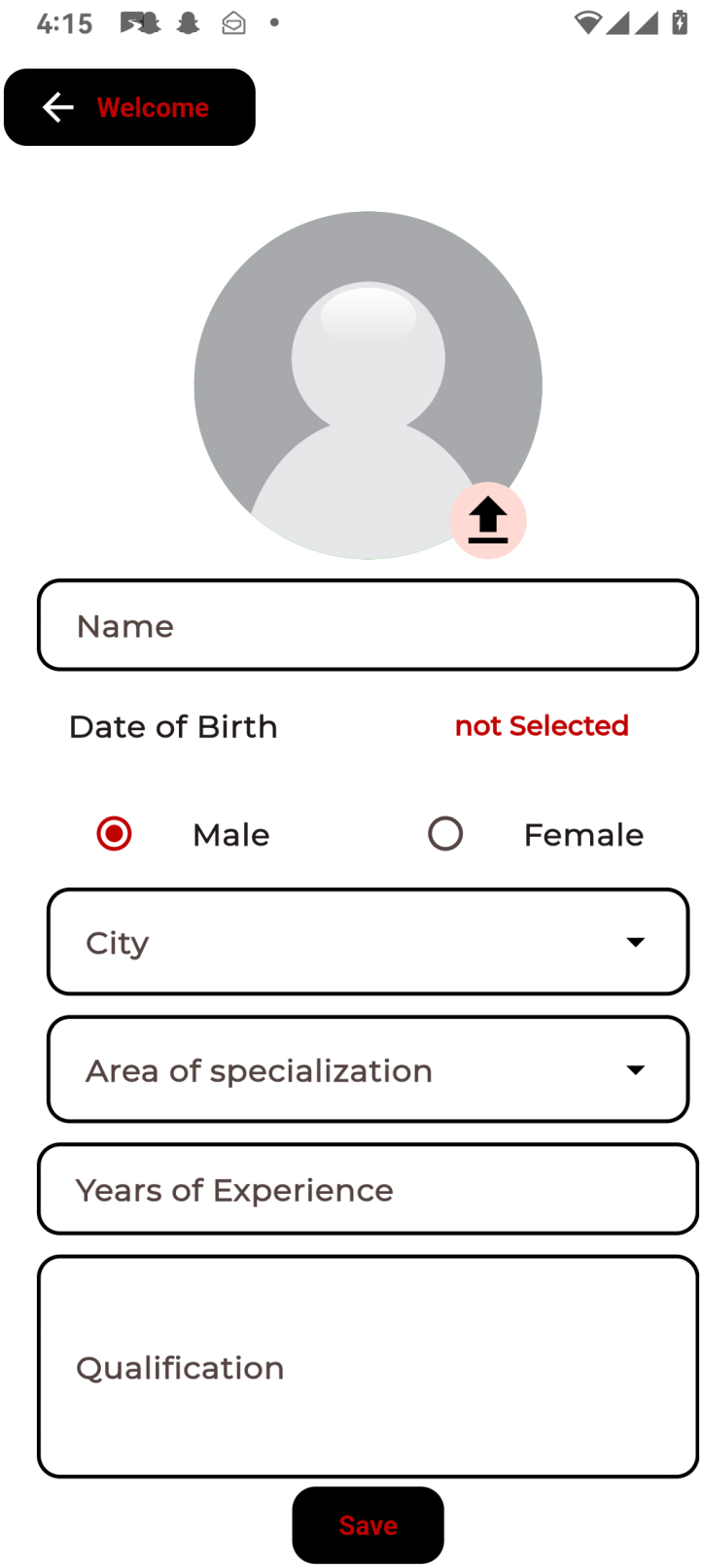

Figure 6.4 Doctor On-board Screen

Figure 6.5 Home Screen - Admin Dashboard

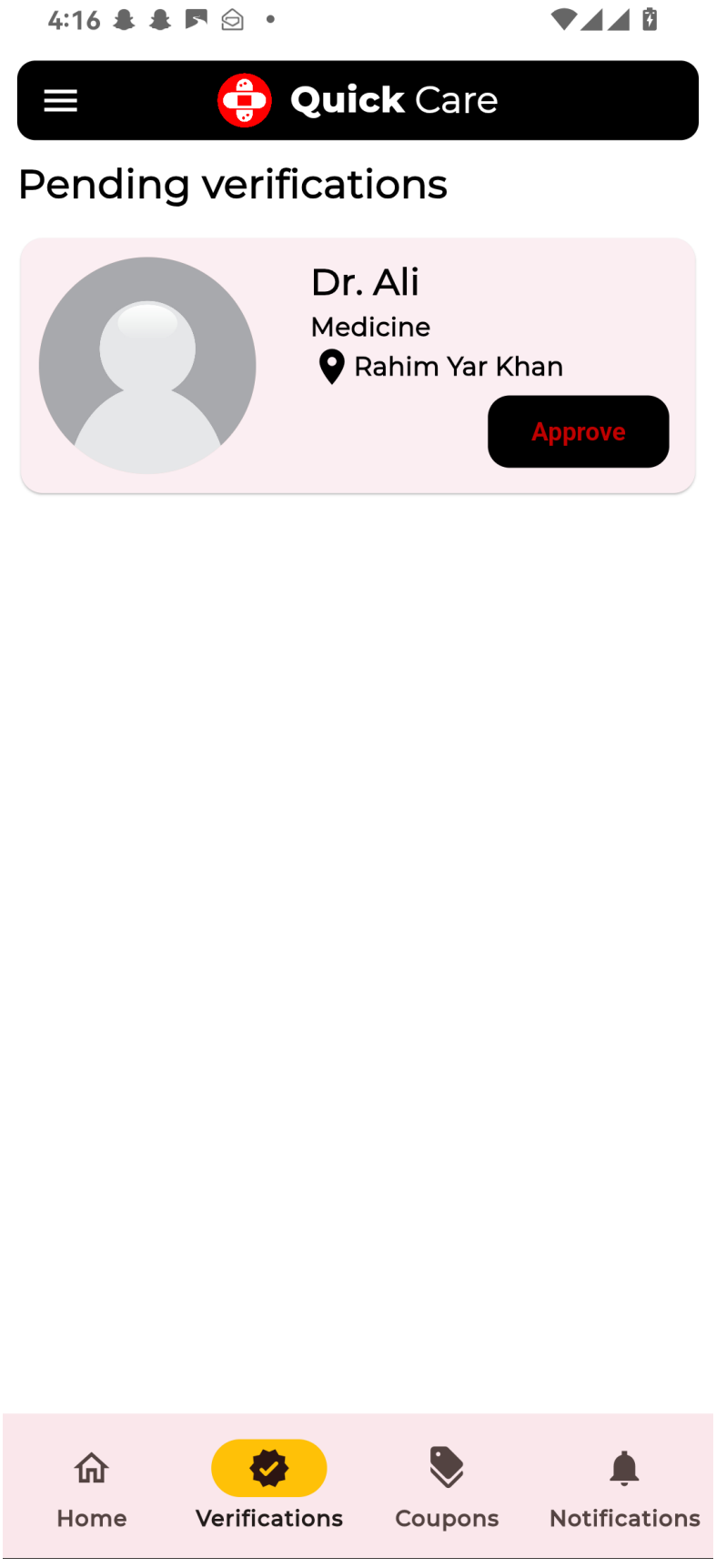

Figure 6.6 Verification Screen - Admin Dashboard

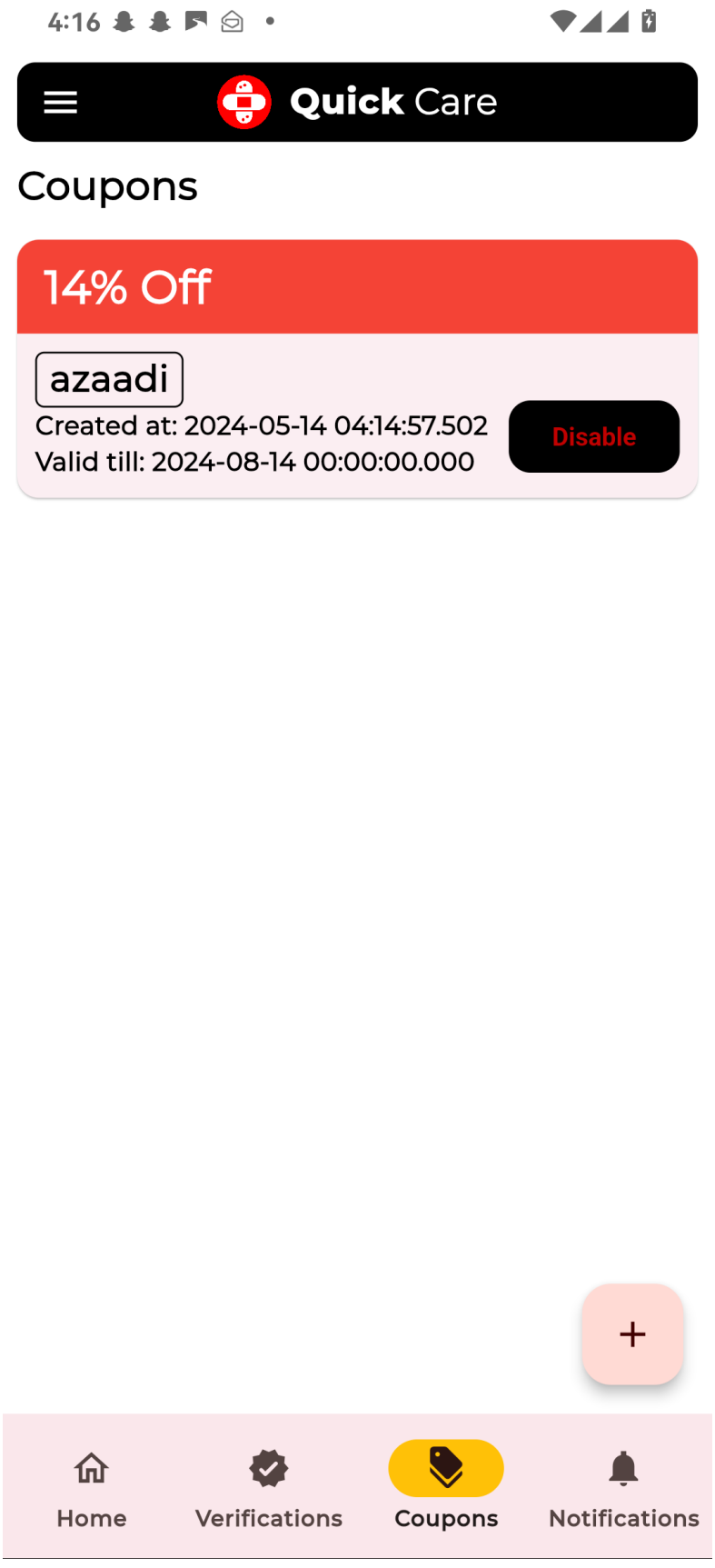

Figure 6.7 Coupons Screen - Admin Dashboard

Figure 6.8 Notifications Screen - Admin Dashboard

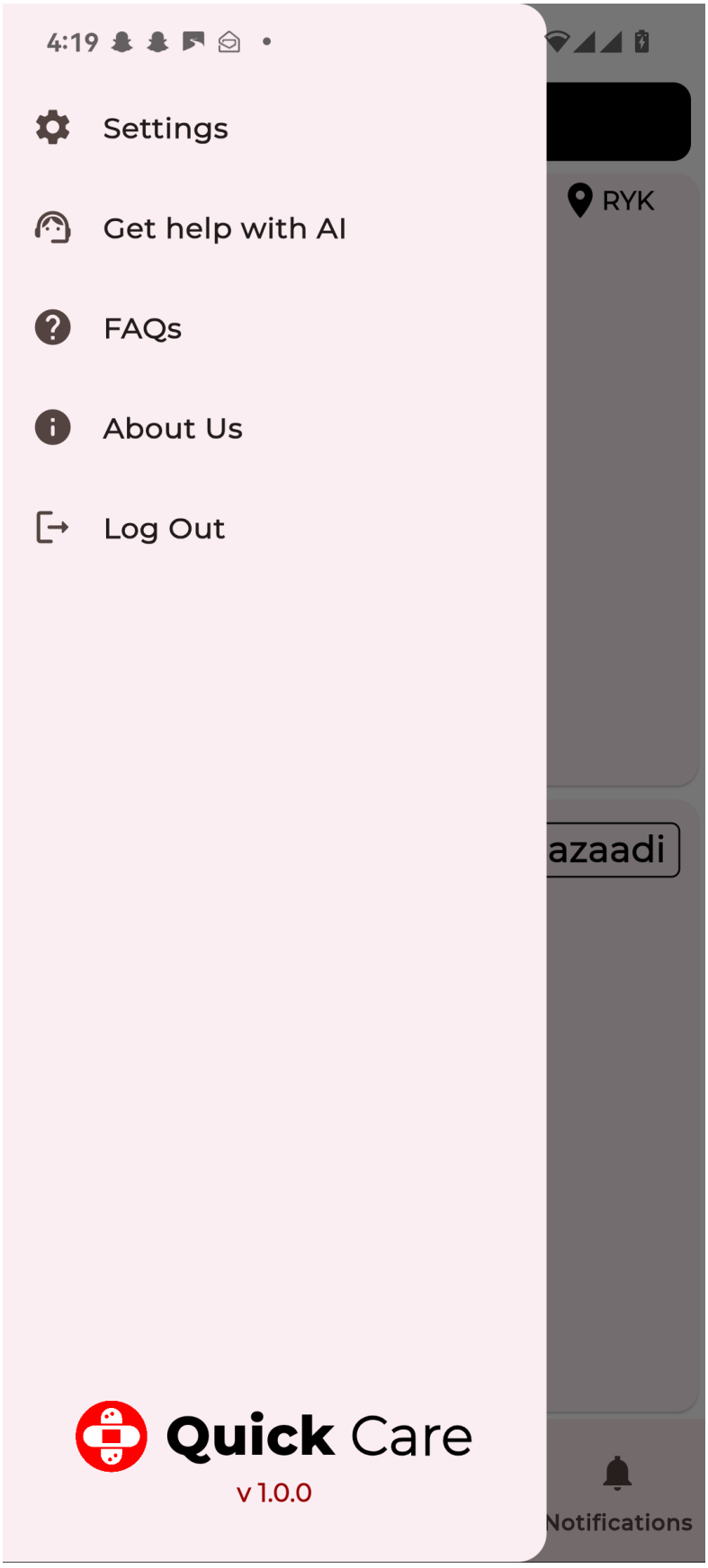

Figure 6.9 App Drawer - Admin Dashboard

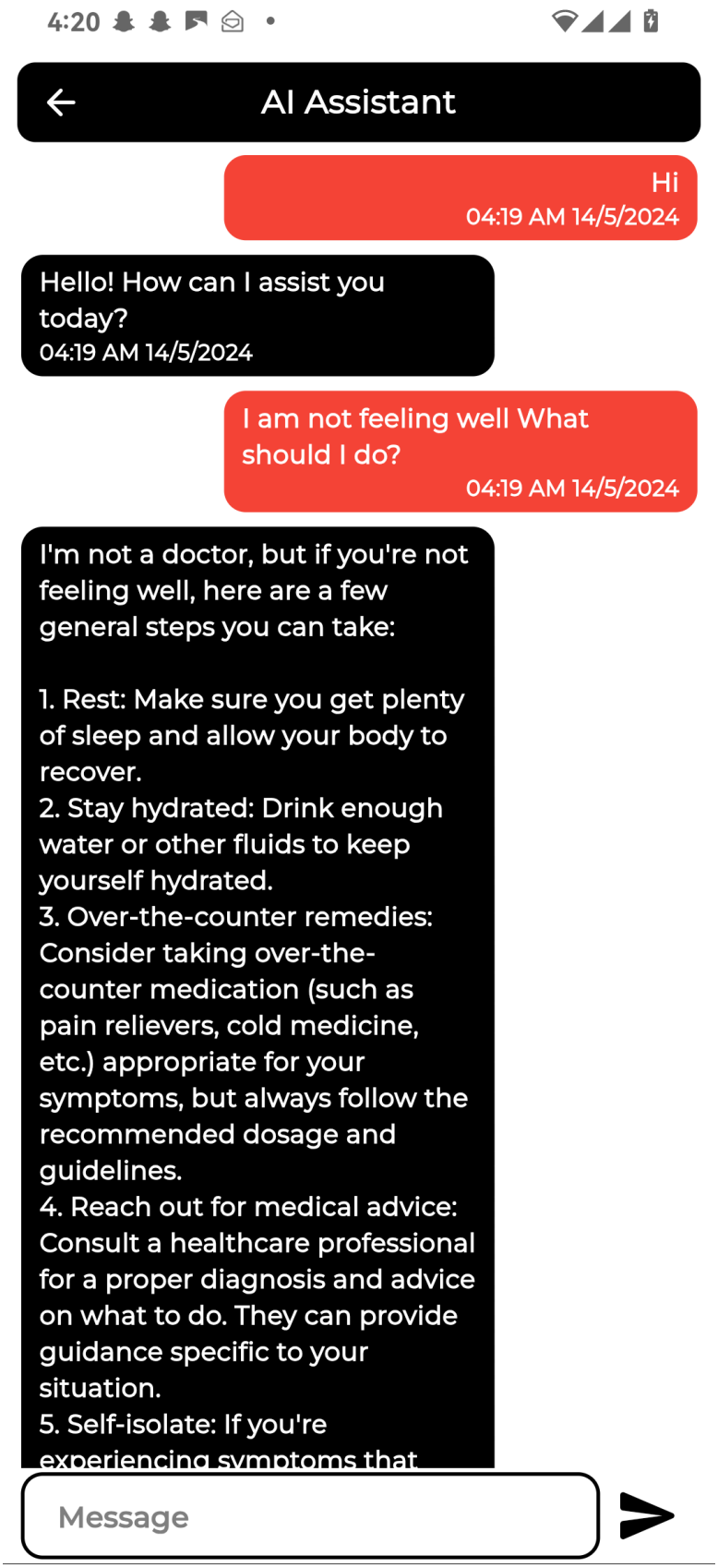

Figure 6.10 AI Assistant Chat Screen

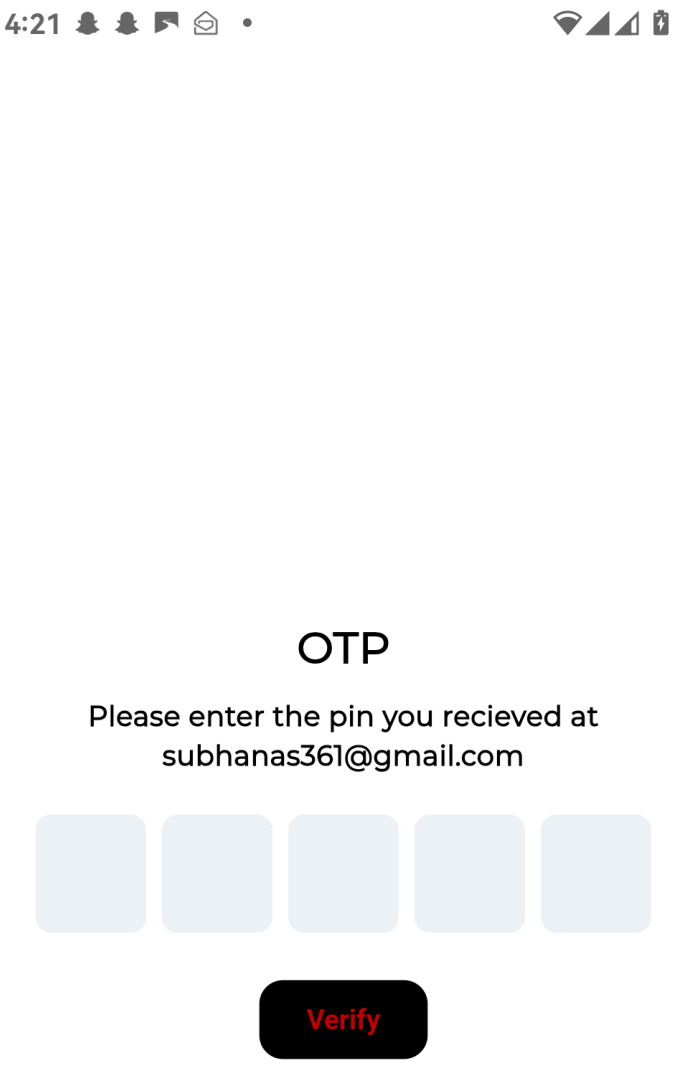

Figure 6.11 OTP Screen

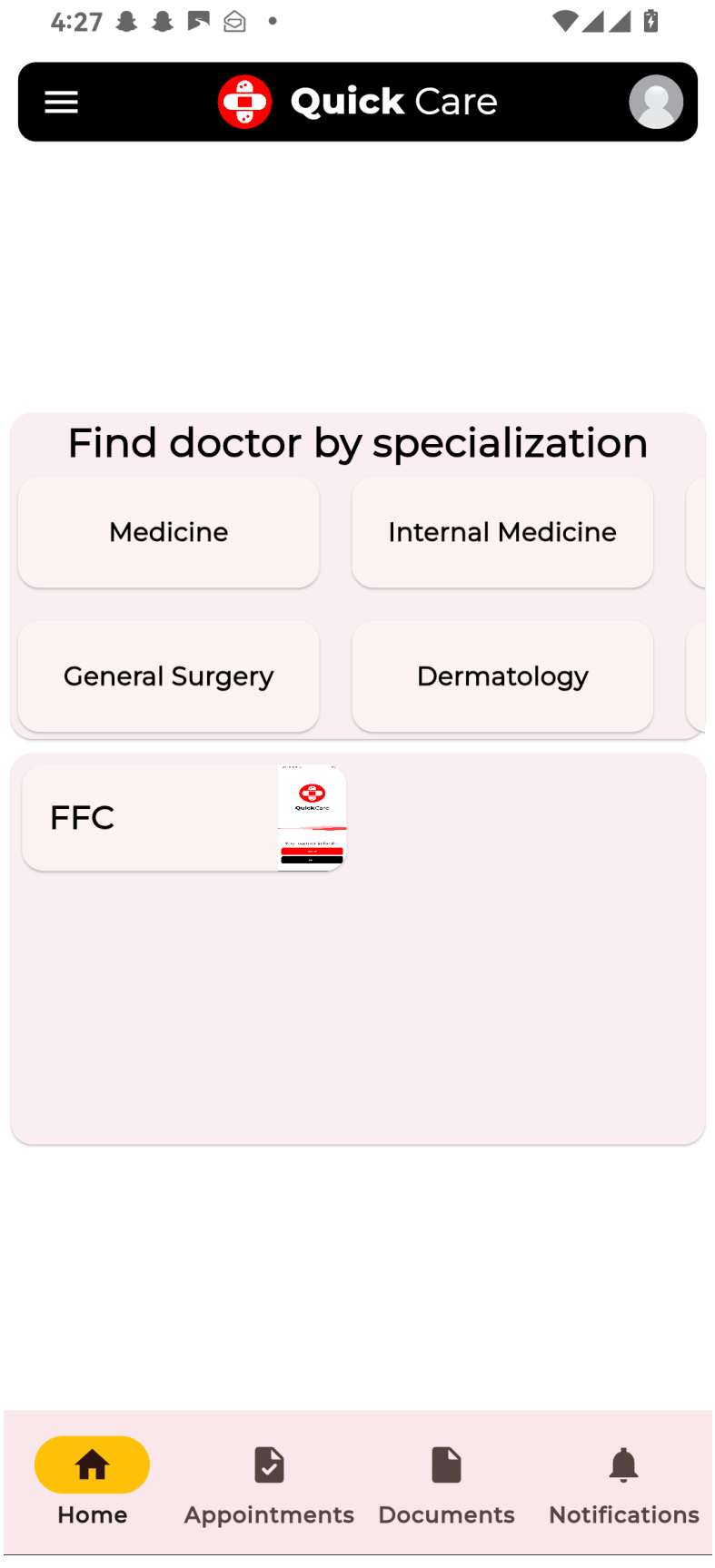

Figure 6.12 Home Screen - Patient Dashboard

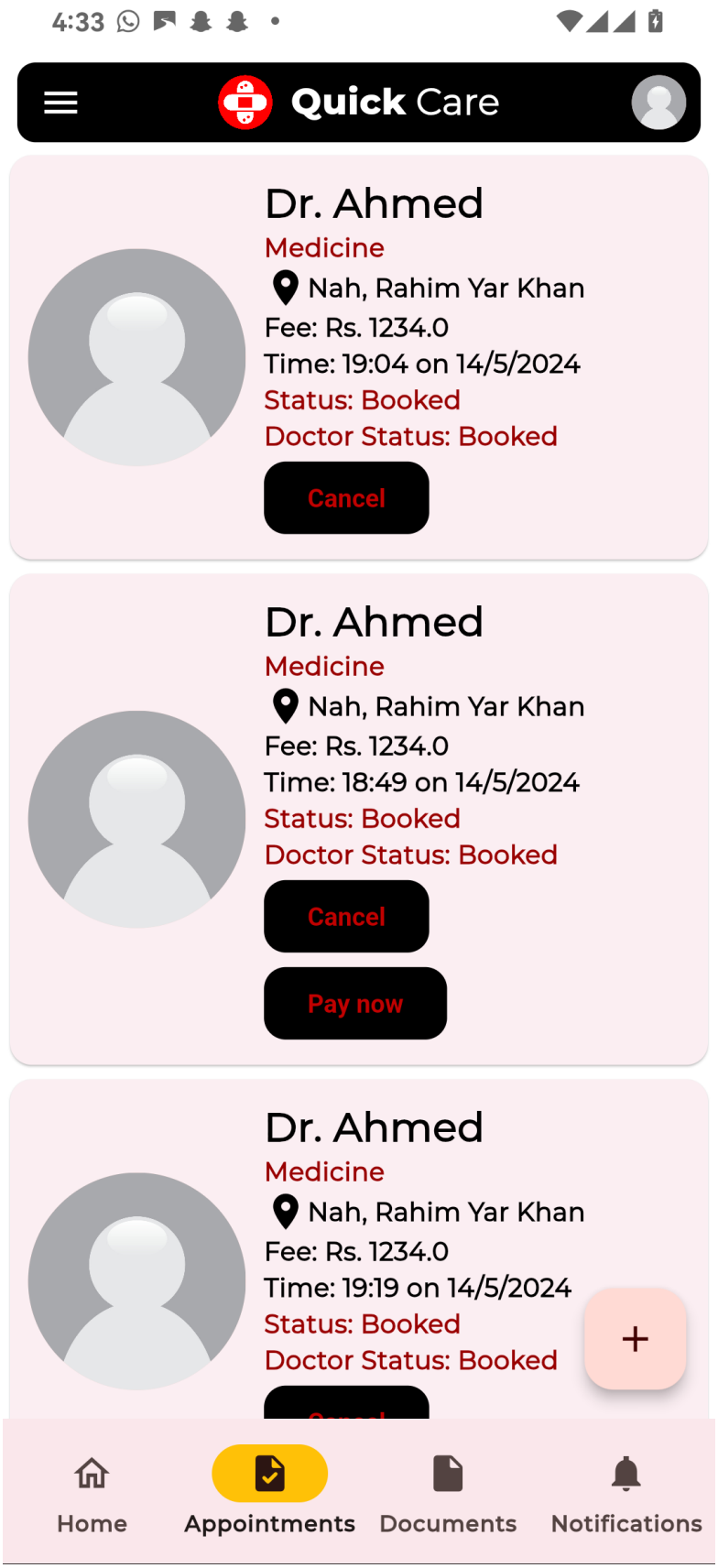

Figure 6.13 Appointment Screen - Patient Dashboard

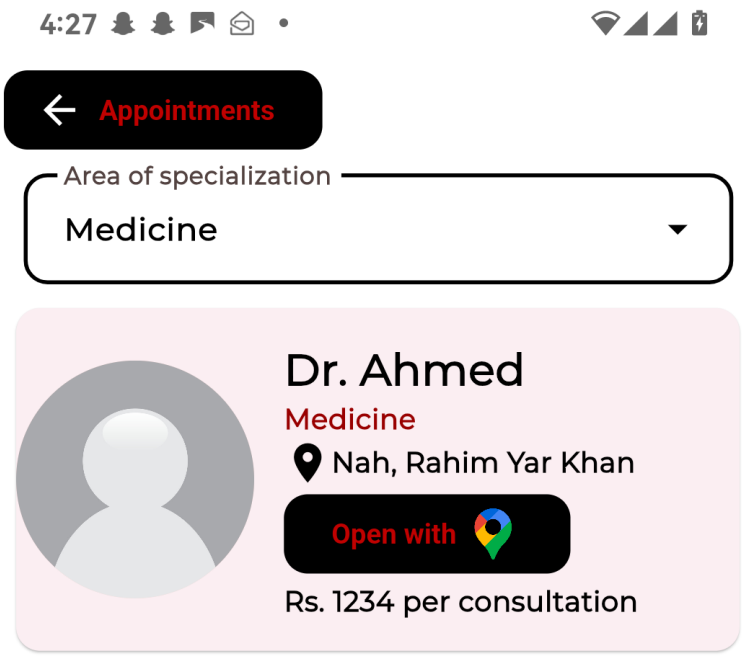

Figure 6.14 Schedules based of Specialization

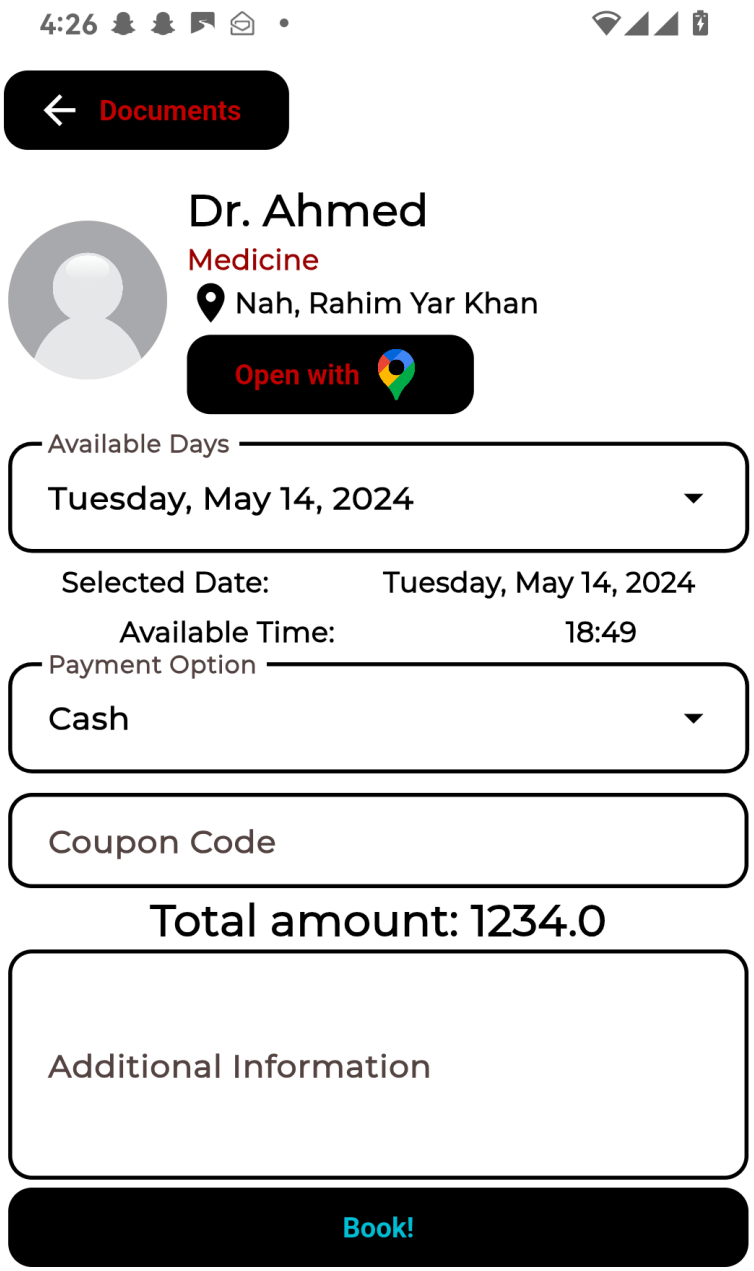

Figure 6.15 Appointment Booking Screen

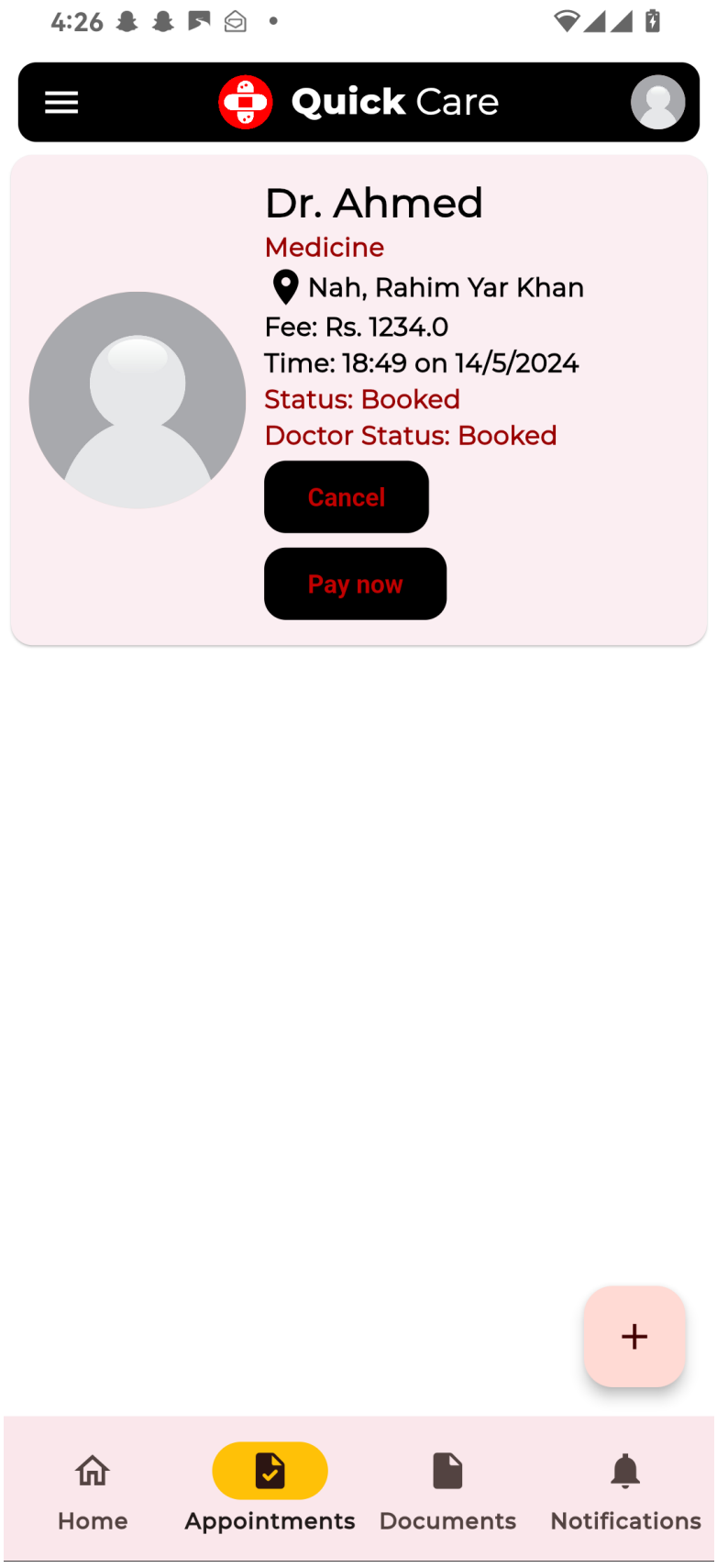

Figure 6.16 Patient Appointment Screen

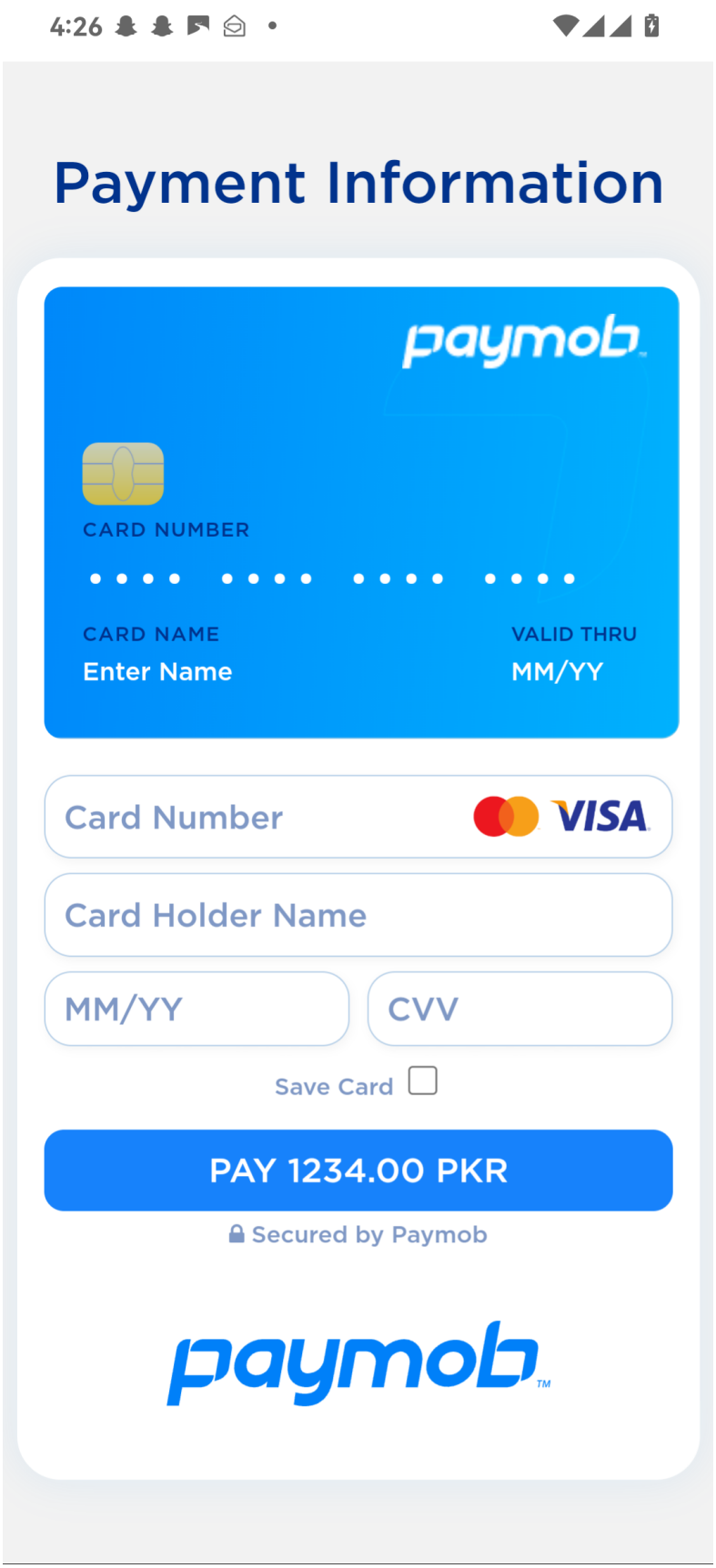

Figure 6.17 Online Payments Screen

Figure 6.18 Documents Screen - Patient Dashboard

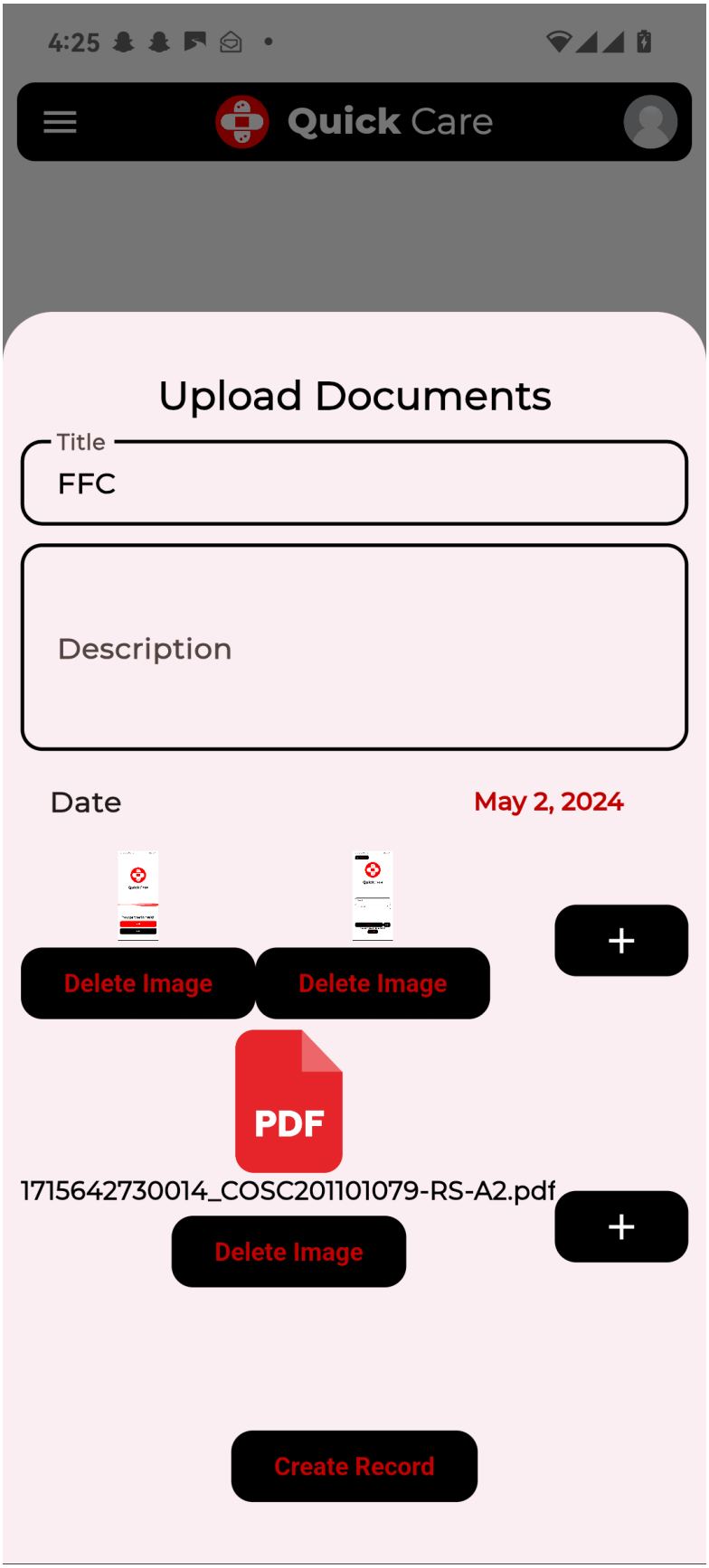

Figure 6.19 Documents Upload & Edit Bottom Sheet

Figure 6.20 Document Viewer

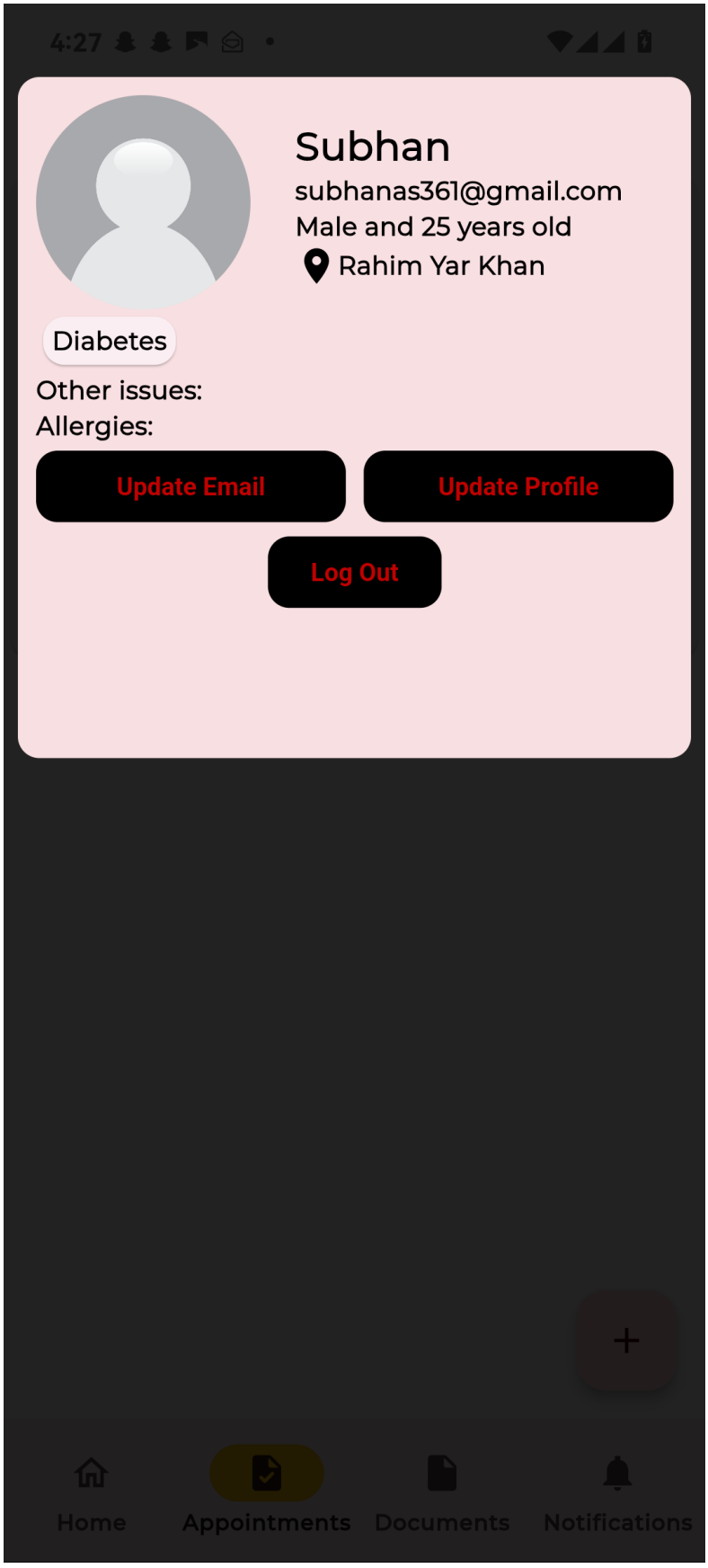

Figure 6.21 Profile view for patient

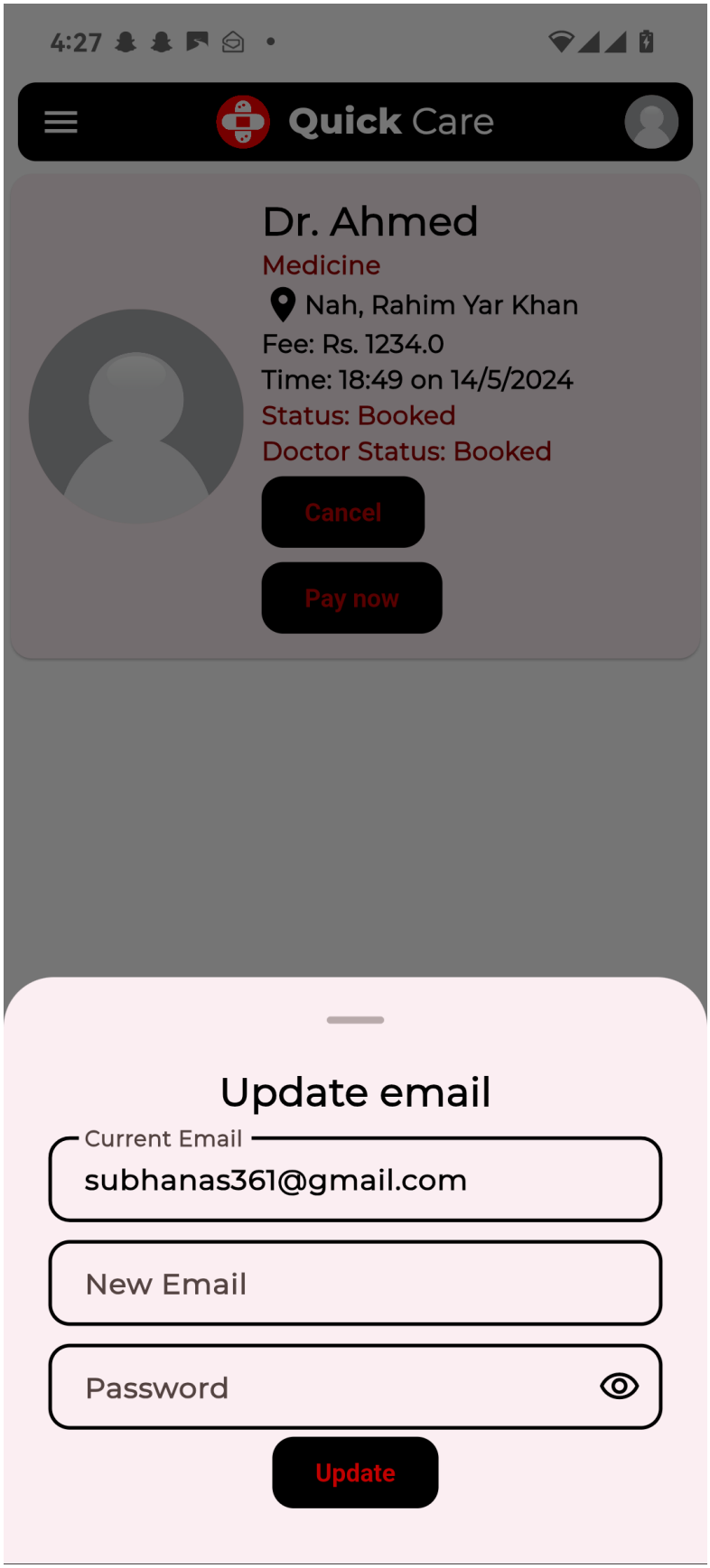

Figure 6.22 Email Update Bottom Sheet

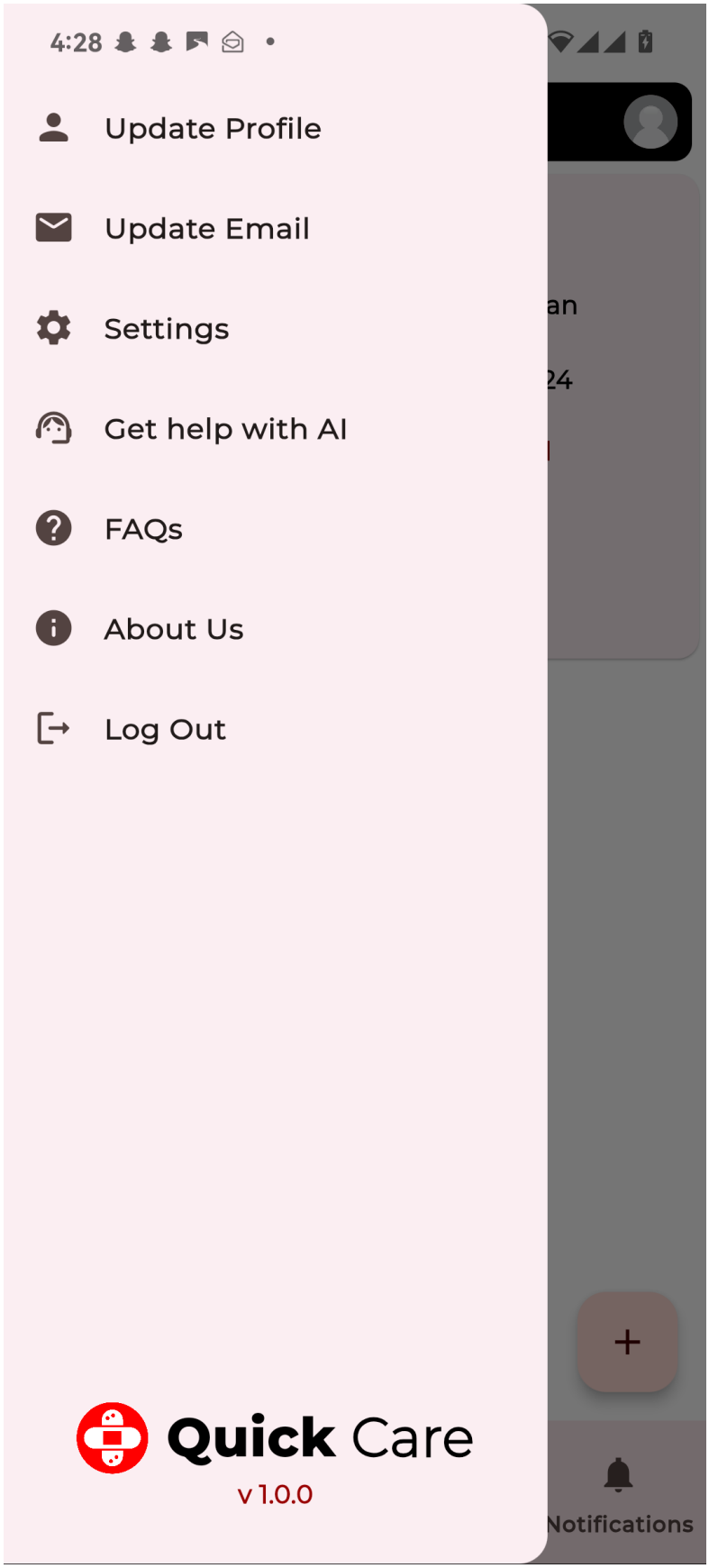

Figure 6.23 App Drwaer - Patient & Doctor Dashboard

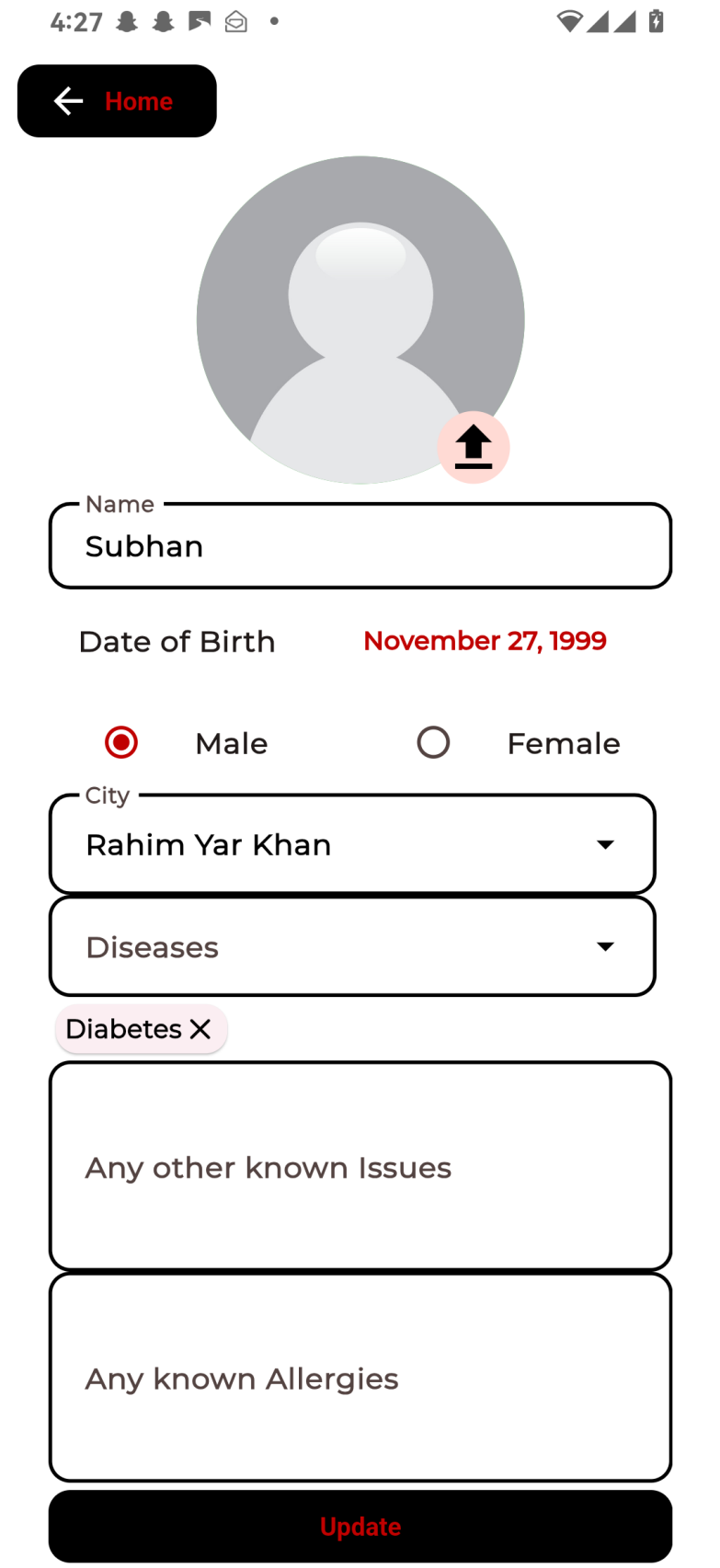

Figure 6.24 Oboard & Profile Edit Screen

Figure 6.25 Notification Screen - Patient Screen

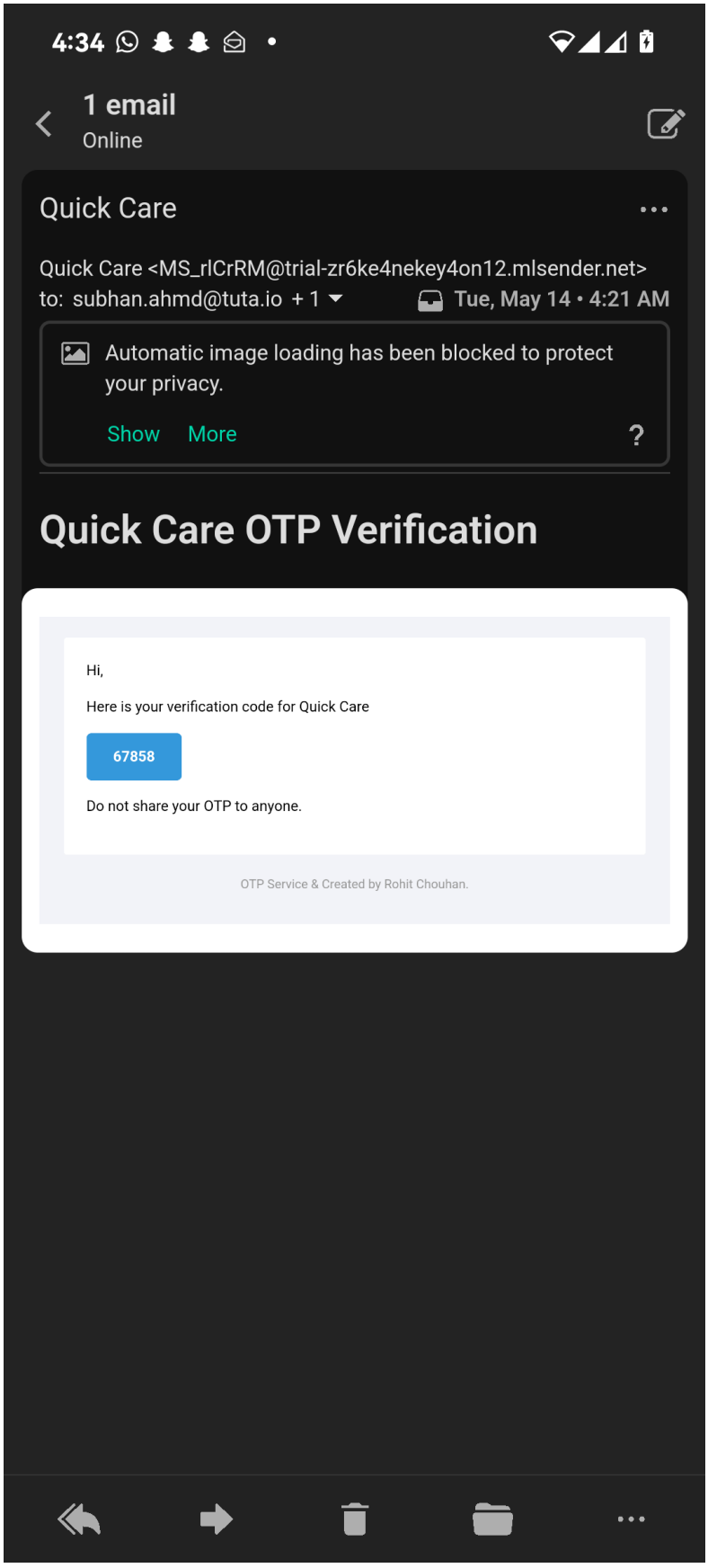

Figure 6.26 OTP Email Sample

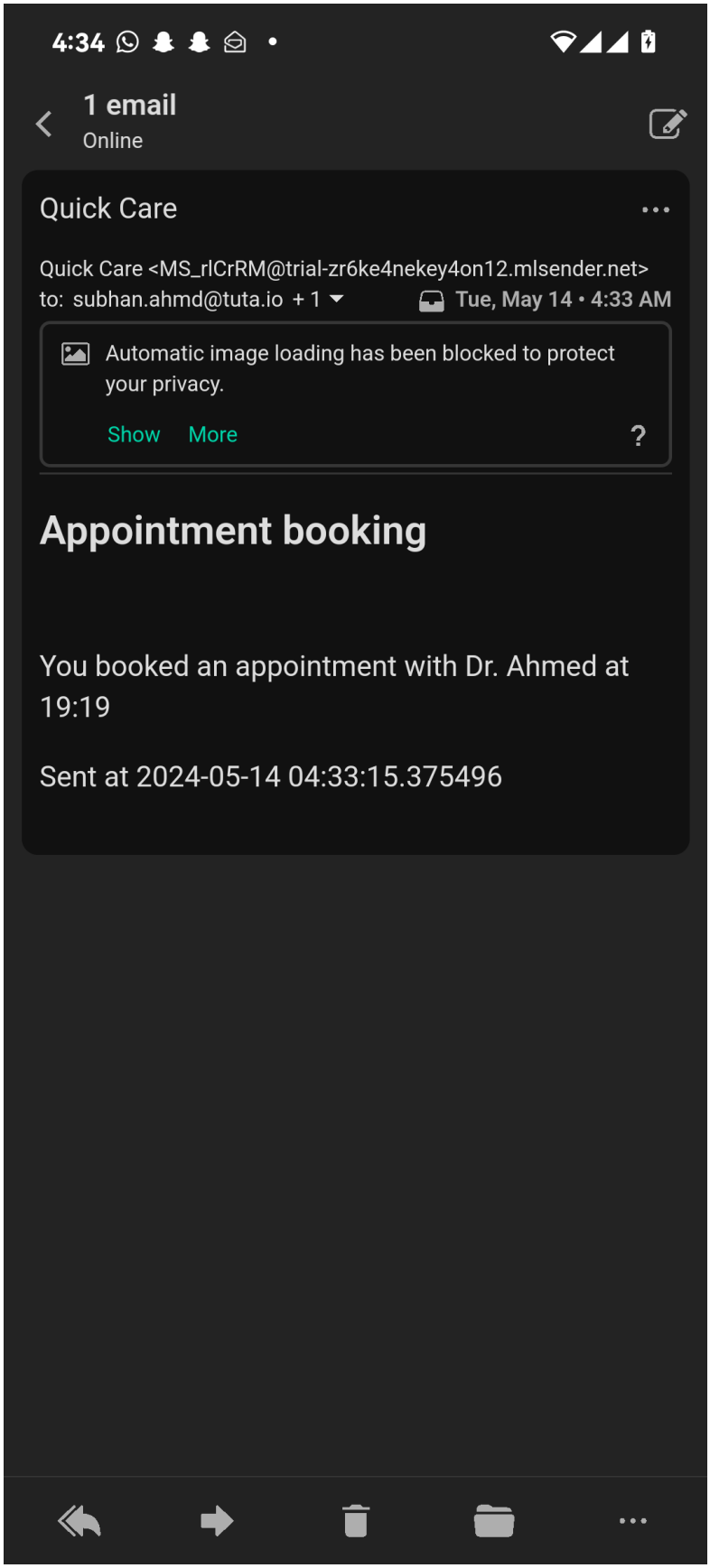

Figure 6.27 Notification Email Sample

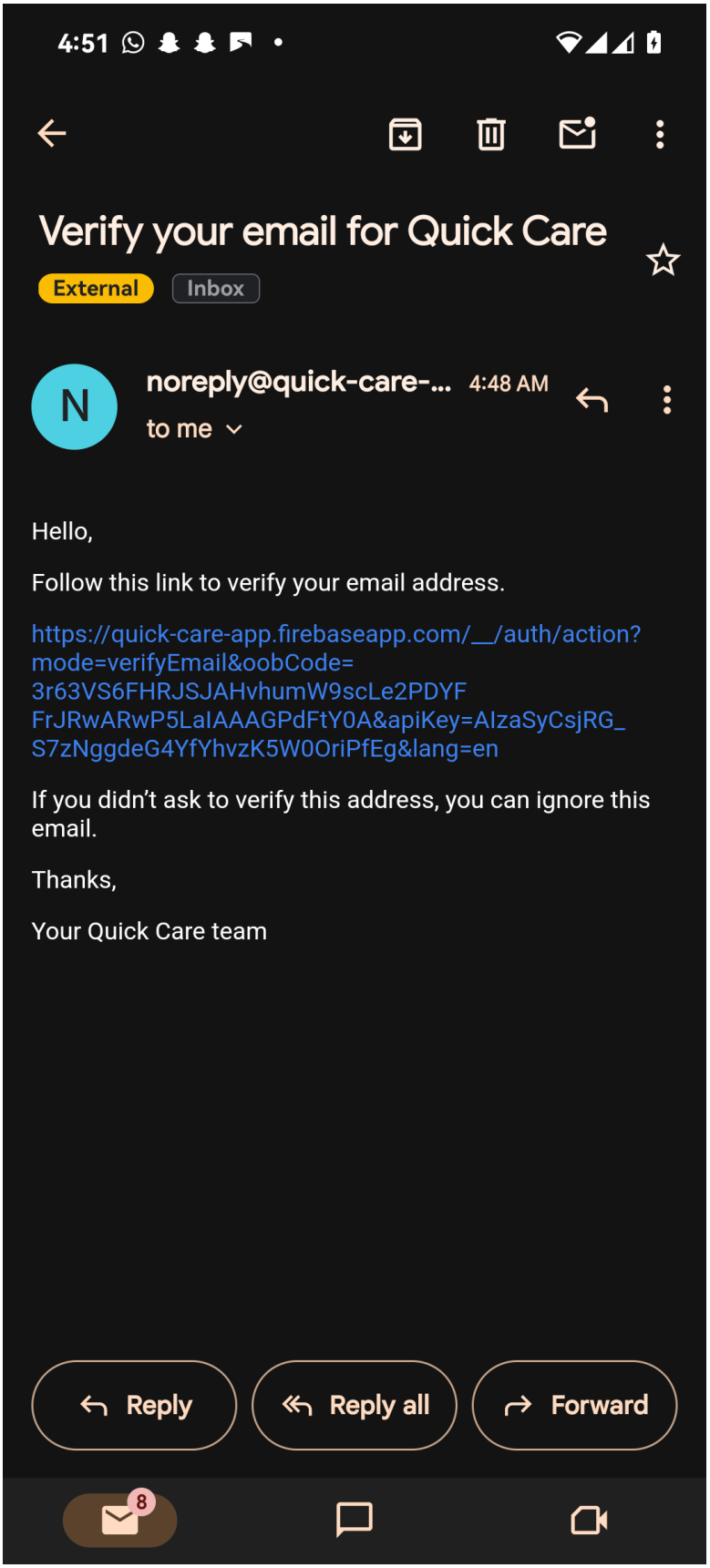

Figure 6.28 Verification Email Sample

Figure 6.29 Notification Screen - Doctor Dashboard

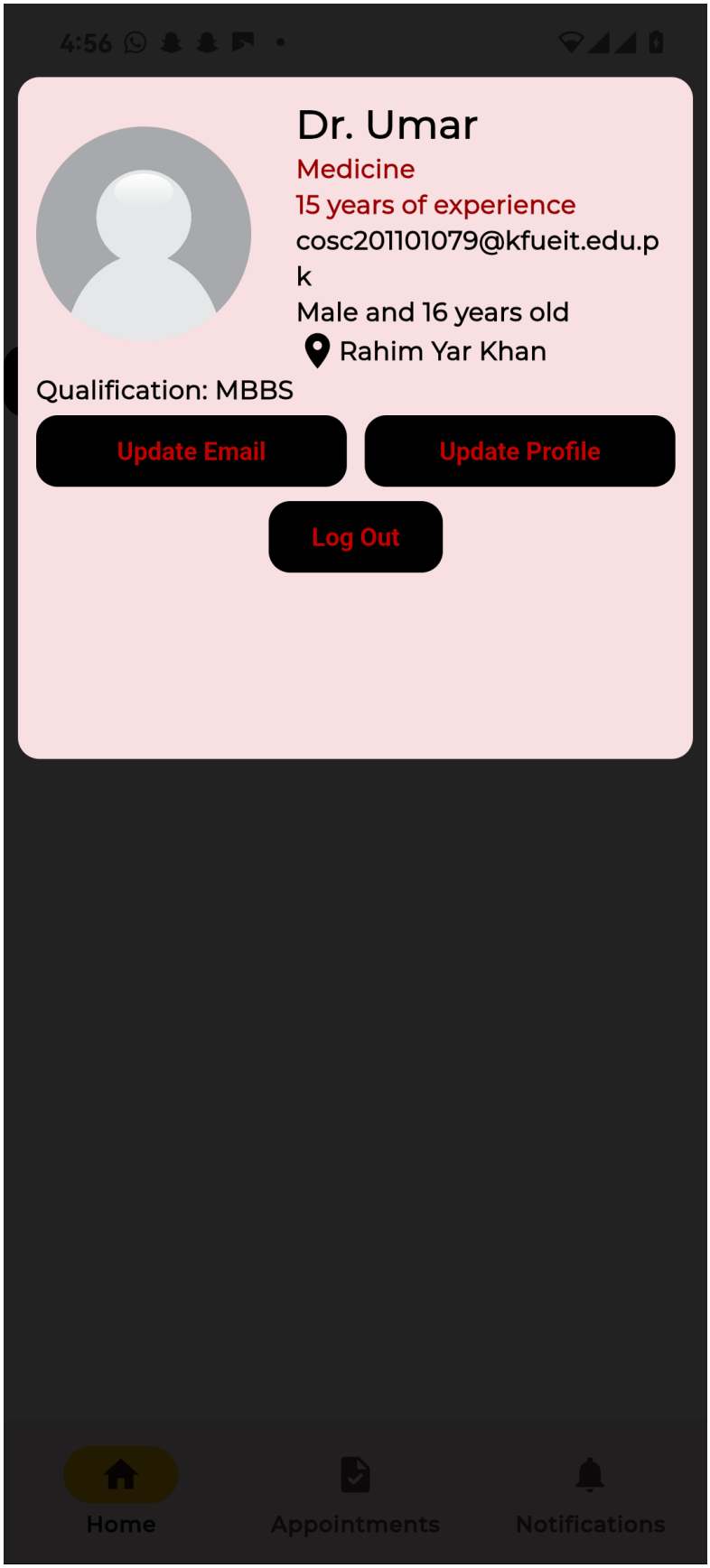

Figure 6.30 Profile View for Doctors

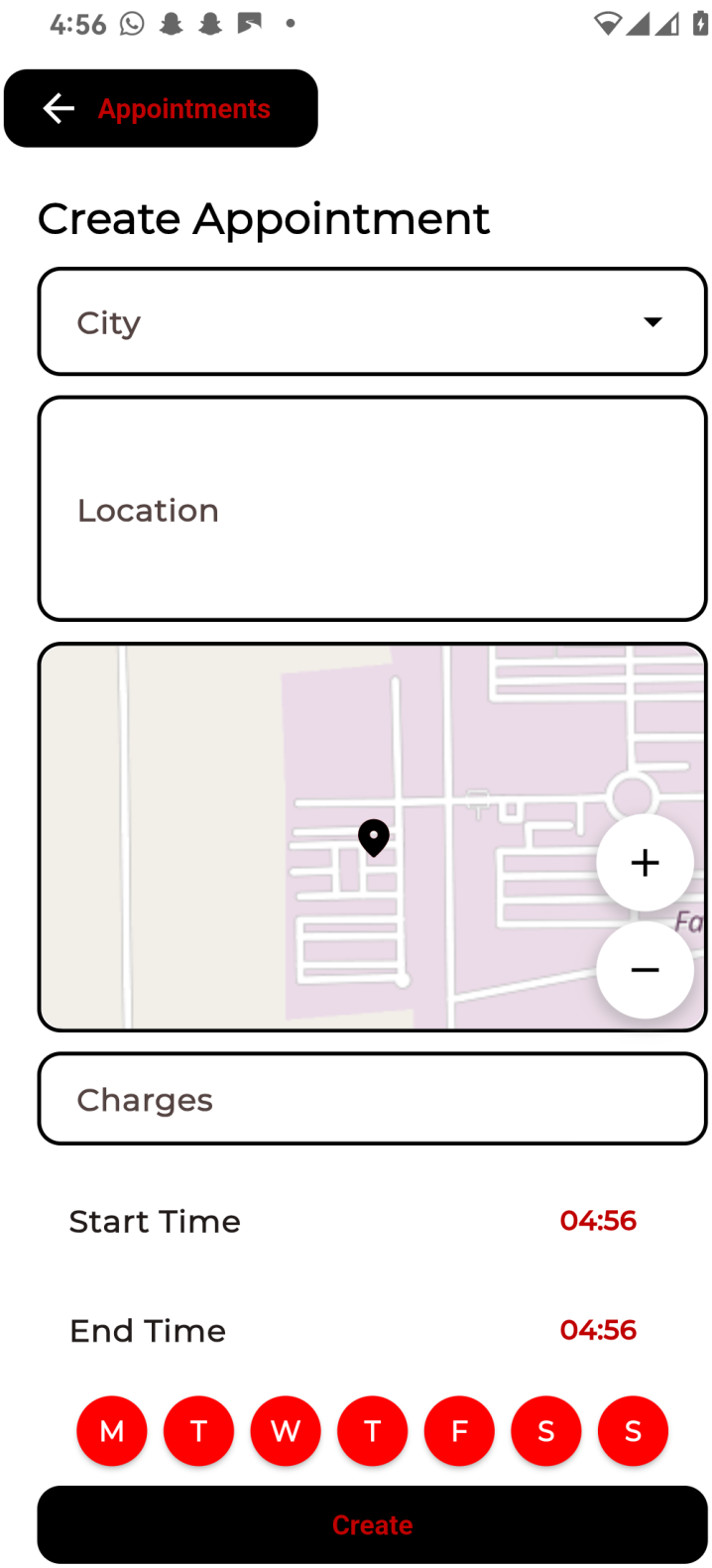

Figure 6.31 Appointment Creation Screen

6.6. Program Deployment

- **App Stores:** Apps will be prepared according to Google Play Store and Apple App Store guidelines (assets, metadata, compliance reviews). We'll factor in potential review time when establishing a release plan.
- **Maintenance and Updates:** We will have a dedicated process for collecting user feedback, prioritizing bug fixes, rolling out security updates, and introducing well-planned new features in an iterative manner.

6.7. Chapter Summary

An Agile development process, emphasis on iterative design, and a well-equipped toolset. Focus on secure data handling and user-centric UI. Continuous improvement via feedback and updates.

Chapter 7

TESTING

7.1. Introduction

Importance of Testing: A rigorously tested patient appointment system ensures that appointments are booked reliably, patient data remains secure, and the application functions smoothly across different devices. This leads to a positive user experience and builds trust in the system.

Scope of Testing: Our testing approach will encompass:

- **Unit Testing:** Verifying the behaviour of individual functions, such as appointment validation logic or Firestore data retrieval.
- **Integration Testing:** Ensuring that the frontend (Flutter), backend (Firebase), and any external services (e.g., payment gateway) work together seamlessly.
- **UI Testing:** Validating that the app's interface displays correctly on various screen sizes and handles user interactions as expected.
- **User Acceptance Testing:** Gathering feedback from potential users to uncover usability issues and to assess overall satisfaction with the system.

7.2. Testing Methods

Black Box Testing: We'll focus on testing the app from the user's perspective. For example:

- Can a patient successfully book a new appointment?
- Can a doctor view their upcoming appointments?
- Does the search function return accurate results?
- White Box Testing: Testing internal code logic for correctness:
- Do appointment validation rules correctly prevent overlaps or conflicts?
- Are data updates to Firestore executed securely and without race conditions?

Automated Testing: We'll automate unit and integration tests using Flutter's testing framework and suitable Firebase testing tools. UI test automation may be explored if resources permit.

Manual Testing: Manual testing will be essential for evaluating the overall user experience, conducting exploratory tests, and gathering feedback during user acceptance testing.

7.3. Comparison

Black Box vs. White Box: Black box testing ensures the app works as intended from the user's point of view, while white box testing helps identify and fix potential errors at the code level. Both are essential for overall quality.

Automated vs. Manual: Automated tests provide rapid, repeatable checks for core functionality, while manual testing allows for nuanced exploration and real-user evaluation, especially regarding the UI.

7.4. Software Evaluation

7.4.1. Testing Strategy

- **Risk-Based Testing:** We'll focus testing efforts on core functionalities (e.g., appointment booking synchronization), data security, and areas impacting the patient experience.
- **Test-Driven Development (TDD) Potential:** If feasible, we'll explore TDD for parts of the codebase. Writing unit tests first can encourage well-structured, testable code.
- **Regression Testing:** Every time new features are added, or changes are made, we'll run a regression suite to ensure existing functionality hasn't been broken.

7.4.2. Test Plans

- **Unit Testing Plan:** This plan will define unit tests for functions handling appointment logic, data validation, Firestore interactions, and state management (if applicable).
- **Integration Testing Plan:** Focus on tests verifying the flow of data between the Flutter UI, Firebase backend, and any additional external systems.
- **UI Testing Plan:** Tests will ensure UI renders properly on target devices, navigation works as intended, and interactive elements respond to user input correctly.

7.4.3. Test Cases

Table 7.1 Test Cases

ID	Feature	Description	Expected Outcome	Outcome
TS-01	User Registration	User registers with valid email and password	Pass	Pass
TS-02	User Registration	User registers with existing email	Pass	Pass
TS-03	User Login	User logs in with valid credentials	Pass	Pass
TS-04	User Login	User logs in with invalid credentials	Pass	Pass
TS-05	Appointment Booking	User searches for a doctor by specialty	Pass	Pass
TS-06	Appointment Booking	User selects a doctor and available time slot	Pass	Pass
TS-07	Online Payment	User enters valid payment information for appointment	Pass	Pass
TS-08	Online Payment	User enters invalid payment information	Pass	Pass
TS-09	Real-time Queue	User views the queue for an upcoming appointment	Pass	Pass
TS-10	Doctor Availability	User checks if a doctor is currently in the hospital	Pass	Pass
TS-11	Notification System	User receives a notification for an upcoming appointment	Pass	Pass
TS-12	Doctor Profile Management	Doctor updates their profile information	Pass	Pass

TS-13	Patient Review	Patient leaves a review for a doctor after an appointment	Pass	Pass
TS-14	Past Medical Record	Patient uploads a medical record document	Pass	Pass
TS-15	Past Medical Record Sharing	Patient shares a medical record with a doctor	Pass	Pass

7.4.4. Test Report

Table 7.2 Test Case 01

Test Case ID	TS-01
Feature	User Registration
Description	User registers with valid email and password
Expected Result	Account creation successful, confirmation email sent
Outcome	Pass
Notes	-

Table 7.3 Test Case 02

Test Case ID	TS-02
Feature	User Registration
Description	User registers with existing email
Expected Result	Error message indicating email already in use
Outcome	Pass
Notes	-

Table 7.4 Test Case 03

Test Case ID	TS-03
Feature	User Login
Description	User logs in with valid credentials
Expected Result	Successful login, redirection to user dashboard
Outcome	Pass
Notes	-

Table 7.5 Test Case 04

Test Case ID	TS-04
Feature	User Login
Description	User logs in with invalid credentials
Expected Result	Error message indicating invalid login attempt
Outcome	Pass
Notes	-

Table 7.6 Test Case 05

Test Case ID	TS-05
Feature	Appointment Booking
Description	User searches for a doctor by specialty
Expected Result	List of relevant doctors displayed
Outcome	Pass
Notes	-

Table 7.7 Test Case 06

Test Case ID	TS-06
Feature	Appointment Booking
Description	User selects a doctor and available time slot
Expected Result	Appointment booking confirmation and estimated wait time displayed
Outcome	Pass
Notes	-

Table 7.8 Test Case 07

Test Case ID	TS-07
Feature	Online Payment
Description	User enters valid payment information for appointment
Expected Result	Payment successful, confirmation email sent
Outcome	Pass
Notes	Test with different payment methods

Table 7.9 Test Case 08

Test Case ID	TS-08
Feature	Online Payment
Description	User enters invalid payment information
Expected Result	Error message indicating payment failure
Outcome	Pass
Notes	Simulate various payment errors

Table 7.10 Test Case 09

Test Case ID	TS-09
Feature	Real-time Queue
Description	User views the queue for an upcoming appointment
Expected Result	Live queue updates with estimated wait time displayed
Outcome	Pass
Notes	Test at different points in the queue

Table 7.11 Test Case 10

Test Case ID	TS-10
Feature	Doctor Availability
Description	User checks if a doctor is currently in the hospital
Expected Result	Accurate availability status displayed (present/absent)
Outcome	Pass
Notes	-

Table 7.12 Test Case 11

Test Case ID	TS-11
Feature	Notification System
Description	User receives a notification for an upcoming appointment reminder as email
Expected Result	Notification received on the user's email address
Outcome	Pass
Notes	-

Table 7.13 Test Case 12

Test Case ID	TS-12
Feature	Doctor Profile Management
Description	Doctor updates their profile information (education, specialization)
Expected Result	Updated information displayed accurately
Outcome	Pass
Notes	-

Table 7.14 Test Case 13

Test Case ID	TS-13
Feature	Patient Review
Description	Patient leaves a review for a doctor after an appointment
Expected Result	Review submitted and displayed on doctor's profile
Outcome	Pass
Notes	-

Table 7.15 Test Case 14

Test Case ID	TS-14
Feature	Past Medical Record
Description	Patient uploads a medical record document
Expected Result	Document uploaded successfully and accessible within the app
Outcome	Pass
Notes	Test with different file formats (Image and pdf)

Table 7.16 Test Case 15

Test Case ID	TS-02
Feature	Past Medical Record Sharing
Description	Patient shares a medical record with a doctor
Expected Result	Doctor receives the shared record securely
Outcome	Pass
Notes	Test with different sharing permissions

Bug Tracking: Issues will be logged in a tracking system (e.g., Jira) with priorities and assignments.

7.5. Chapter Summary

- **Comprehensive Approach:** Our testing combines black box, white box, automated, and manual approaches to maximize reliability.
- **Feedback Loop:** Test results will guide bug fixes, feature improvements, and ensure a positive user experience.
- **Patient Trust:** Rigorous testing demonstrates our commitment to safeguarding patient data and providing a trustworthy system.

REFERENCES

[1] Z. N. Paracha, "Marham raises $1 million seed to grow into a healthcare superapp for Pakistan," MenaBytes, 3 August 2021. [Online]. Available: https://www.menabytes.com/marham-seed/.

[2] S. Report, "MyDoctor.pk raises $1.1million funding, rebrands to oladoc.com," Profit Pakistan Today, 21 February 2018. [Online]. Available: https://profit.pakistantoday.com.pk/2018/02/21/mydoctor-pk-raises-1-1million-funding-rebrands-to-oladoc-com/#:~:text=LAHORE%3A%20MyDoctor.pk%2C%20Pakistan's,doctors%20and%20specialists%20near%20them..

[3] G. Joshi, "Incremental model: Advantages And Disadvantages," FindNerd, [Online]. Available: https://findnerd.com/list/view/Incremental-model-Advantages-And-Disadvantages/18426/.